APLIKASI TEKNIK *AUTOMATIC NUMBER PLATE RECOGNITION* BERBASIS FPGA UNTUK MENDETEKSI PLAT NOMOR KENDARAAN

SKRIPSI

TEKNIK ELEKTRO KONSENTRASI ELEKTRONIKA

Ditujukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

DIANNATA RAHMAN YULIANSYAH
NIM. 125060300111035

UNIVERSITAS BRAWIJAYA
FAKULTAS TEKNIK
MALANG
2016

LEMBAR PENGESAHAN

APLIKASI TEKNIK *AUTOMATIC NUMBER PLATE RECOGNITION* BERBASIS FPGA UNTUK MENDETEKSI PLAT NOMOR KENDARAAN

SKRIPSI

KONSENTRASI ELEKTRONIKA

Ditujukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

DIANNATA RAHMAN YULIANSYAH
NIM. 125060300111035

Skripsi ini telah direvisi dan disetujui oleh dosen pembimbing
Pada 19 Agustus 2016

Dosen Pembimbing I	Dosen Pembimbing II
Ir. Nanang Sulistiyanto, M.T.	Rahmadwati, S.T., M.T., Ph.D
NIP : 19700113 199403 1 002	NIP : 19771102 200604 2 003

Mengetahui
Ketua Jurusan Teknik Elektro

M. Aziz Muslim, S.T, M.T., Ph.D.
NIP : 19741203 200012 1 001

JUDUL SKRIPSI :

APLIKASI TEKNIK *AUTOMATIC NUMBER PLATE RECOGNITION* BERBASIS FPGA UNTUK MENDETEKSI PLAT NOMOR KENDARAAN

Nama Mahasiswa : Diannata Rahman Yuliansyah

NIM : 125060300111035

Program Studi : Teknik Elektro

Konsentrasi : Teknik Elektronika

KOMISI PEMBIMBING :

Ketua : Ir. Nanang Sulistiyanto, M.T. .....

Anggota : Rahmadwati, S.T., M.T., Ph.D .....

TIM DOSEN PENGUJI :

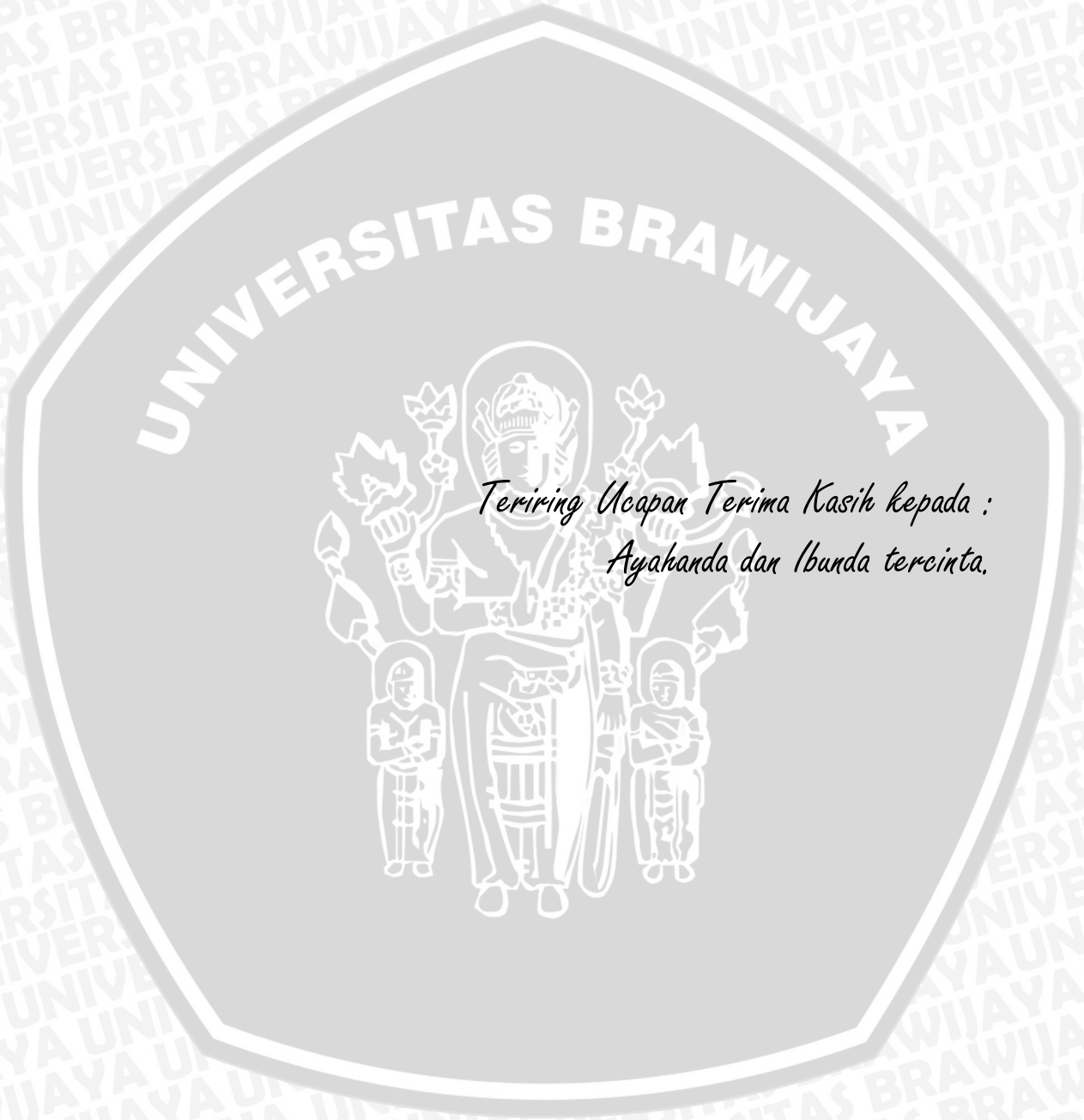
Dosen Penguji 1 : Dr.Ir. Ponco Siwindarto, M.Eng.Sc. .....

Dosen Penguji 2 : Ir. Nurussa'adah, M.T. .....

Dosen Penguji 3 : Eka Maulana, S.T., M.T., M.Eng .....

Tanggal Ujian : 9 Agustus 2016

SK Penguji : No. 974/UN10.6/SK/2016



Teriring Ucapan Terima Kasih kepada :
Ayahanda dan Ibunda tercinta.

PERNYATAAN ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya dan berdasarkan hasil penulusuran berbagai karya ilmiah, gagasan dan masalah ilmiah yang diteliti dan diulas didalam Naskah Skripsi adalah asli dari pemikiran saya, tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik disuatu perguruan tinggi dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata didalam naskah Skripsi ini dibuktikan terdapat unsur-unsur jiplakan, saya bersedia Skripsi dibatalkan, serta diproses sesuai peraturan perundang-undangan yang berlaku (UU No.20 Tahun 2003, pasal 25 ayat 2 dan pasal 70).

Malang, 19 Agustus 2016

Mahasiswa,

Diannata Rahman Yuliansyah

NIM. 125060300111035

PENGANTAR

Puji syukur ke hadirat Allah SWT atas rahmat dan karunia-Nya dan perkenan-Nya penulis dapat menyelesaikan penulisan skripsi ini. Karya ini tidak mungkin selesai tanpa restu dan dukungan dari berbagai pihak. Untuk itu penulis mengucapkan terima kasih sedalam-dalamnya yang tidak terhingga kepada:

- Kedua orang tua penulis, Alm. Tugino dan Ibu Sri Wahyuni atas pengorbanan dan doa restunya sehingga penulis dapat menuntut ilmu sampai jenjang sarjana. Serta seluruh keluarga di rumah atas segala doa, dukungan, dan motivasi dalam mengayomi penulis hingga saat ini.
- Bapak M. Aziz Muslim, S.T., M.T., Ph.D selaku Ketua Jurusan Teknik Elektro Universtias Brawijaya.
- Bapak Hadi Suyono, S.T., M.T., Ph.D selaku Sekretaris Jurusan Teknik Elektro Universtias Brawijaya.
- Bapak Ali Mustofa, S.T., M.T., selaku ketua Program Studi Sarjana Teknik Elektro Universtias Brawijaya.
- Ibu Ir. Nurussa'adah, M.T. selaku Ketua Kelompok Dosen Keahlian Elektronika Jurusan Teknik Elektro Universitas Brawijaya yang selalu memberi semangat dan motivasi untuk cepat menyelesaikan skripsi.
- Bapak Ir. Nanang Sulistiyanto, M.T. sebagai dosen pembimbing pertama dan Ibu Rahmadwati, S.T., M.T., Ph.D sebagai dosen pembimbing kedua sekaligus dosen pembimbing akademik, ditengah kesibukan beliau berdua selalu memberikan waktu untuk diskusi dengan tulus, sabar memberikan masukan yang sungguh berharga.
- Para Dosen Pengajar Program Studi Teknik Elektro Universitas Brawijaya, yang tidak dapat penulis sebutkan satu per satu yang telah memberikan bekal ilmu pada penulis dalam menyelesaikan studi.
- Teman-teman UNPAR, Topan, Firman, Seto, Gigih, Aldi, Risal, Ariskun, Satrio, Akbar, Bobby, Inar, Indro, Ibon, Ali, Bagus, Ilham, Valdy, Thoriq, Baskara atas segala motivasi dan semangat serta dukungan dalam pengerjaan skripsi.
- Teman – teman asisten laboratorium desain dan prototype, Ronny, Sirojuddin, Wildan, Bidin, Bagus, Guntoro, Rifqa, dan seluruh asisten atas segala dukungan dan bantuan dalam pengerjaan skripsi.

- Teman-teman VOCTRON konsentrasi Elektronika 2012 dan VOLTAGE angkatan 2012 atas segala dukungan dalam pembuatan skripsi.

Sekiranya Allah SWT membalas kebaikan semua pihak yang turut membantu skripsi ini terselesaikan. Akhirnya, penulis menyadari bahwa skripsi ini masih jauh dari sempurna, namun semoga skripsi ini dapat memberikan manfaat bagi kita semua. Amin, Terima kasih.

Malang, Agustus 2016

Penulis

RINGKASAN

Diannata Rahman Yuliansyah, Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya, April 2016, *Aplikasi Teknik Automatic Number Plate Recognition Berbasis FPGA untuk Mendeteksi Plat Nomor Kendaraan*, Dosen Pembimbing: Nanang Sulistiyanto dan Rahmadwati.

Teknik *Automatic Number Plate Recognition* (ANPR) adalah salah satu metode yang diandalkan untuk identifikasi kendaraan di era modern saat ini. Sistem ANPR yang ada saat ini kebanyakan menggunakan perangkat lunak pada komputer. Desain perangkat keras khusus yang dapat bekerja sebagai pemroses sistem ANPR diperlukan karena lebih unggul dari segi dimensi, konsumsi daya dan harga. Untuk perancangan sistem yang *low-cost* dan efisien, maka FPGA (*Field Programmable Gate Arrays*) adalah pilihan yang tepat untuk mengimplementasikan sistem. Sistem dirancang dengan menggunakan modul sensor kamera OV7670 dan *board* FPGA Nexys3 Spartan-6. Citra diproses pada sistem melalui tiga tahap, yaitu tahap lokalisasi plat nomor menggunakan metode deteksi tepi vertikal dan analisis proyeksi, tahap segmentasi karakter menggunakan metode pengambangan dan analisis proyeksi, dan tahap pengenalan karakter optis menggunakan metode KNN (*K-Nearest Neighbor*) dengan nilai K=1. Hasil perancangan dan pengujian menunjukkan bahwa sistem dapat mengenali 82% dari total karakter yang diuji dengan waktu eksekusi terlama hingga 2,3 ms dan kesalahan maksimum 3 karakter per plat nomor, akan tetapi tingkat keberhasilan identifikasi plat nomor hanya 19%. Hal ini menunjukkan bahwa perancangan sistem dapat dikembangkan lebih lanjut sehingga ke depannya dapat diaplikasikan untuk sistem identifikasi kendaraan otomatis.

Kata Kunci: Kendaraan, ANPR, FPGA

UNIVERSITAS BRAWIJAYA

SUMMARY

Diannata Rahman Yuliansyah, Department of Electrical Engineering, Faculty of Engineering Brawijaya University, April 2016, *FPGA Based Application of Automatic Number Plate Recognition Technique to Detect Number Plate Vehicles*, Academic Supervisor: Nanang Sulistiyanto and Rahmadwati.

Automatic Number Plate Recognition (ANPR) technique is a reliable method for identifying vehicles in today's modern era. ANPR systems that exist today mostly use the software on a computer. Design of special hardware that can work as the ANPR system processor is required because it is superior in terms of power consumptions, dimensions and price. For a low-cost and efficient system design, then FPGA (Field Programmable Gate Arrays) is the right choice to implement the system. The system was designed using the OV7670 camera sensor module and Nexys3 Spartan-6 FPGA board. Images are processed on the system through three stages, namely the number plate localization stage using vertical edge detection method and projection analysis, the character segmentation stage using thresholding method and projection analysis, and the optical character recognition stage using KNN (K-Nearest Neighbor) method with the value of K = 1. Design and test results have shown that the system could recognize 82% of the total characters tested with a longest execution time up to 2,3 ms and a maximum error of 3 characters per number plate, but the successful rate of number plate identification was only 19%. This shows that the design of the system can be developed further so that in the future it can be applied for automatic vehicle identification system.

Keywords: Vehicle, ANPR, FPGA

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

UNIVERSITAS BRAWIJAYA

BAB I
PENDAHULUAN

1.1 Latar Belakang

Metode pengenalan plat nomor otomatis atau yang dikenal dengan teknik *Automatic Number Plate Recognition* (ANPR) adalah salah satu metode yang diandalkan untuk identifikasi kendaraan di era modern saat ini. Teknologi ini masih terus dikembangkan dan diaplikasikan pada berbagai aspek penting dalam sistem transportasi dan manajemen lalu lintas. Dengan ANPR maka kendaraan dapat diidentifikasi, dilacak ataupun dimonitor pergerakannya dengan mengekstrak plat nomor secara otomatis.

Sebuah sistem ANPR terdiri dari komponen perangkat keras dan perangkat lunak yang memproses masukan berupa sinyal grafis baik berupa citra statis maupun citra dinamis (video), dan mengenali karakter plat nomor dari citra tersebut. Bagian perangkat keras dari sistem ANPR biasanya terdiri dari kamera, pemroses citra, pemicu kamera, komunikasi, dan unit penyimpanan (Martinsky, 2007:2). Sistem ANPR yang ada saat ini kebanyakan menggunakan perangkat lunak pada komputer. Komputer yang digunakan biasanya adalah komputer untuk tujuan umum (*general purpose*). Untuk mendapatkan performansi maksimum, maka dibutuhkan komputer dengan spesifikasi tinggi. Semakin tinggi spesifikasi komputer tentunya biaya menjadi semakin mahal. Oleh karena itu, kebanyakan sistem yang ada saat ini memiliki kekurangan dari performansi ataupun biaya. Sebuah penelitian yang telah dilakukan sebelumnya menunjukkan implementasi sistem ANPR menggunakan perangkat lunak pada komputer dapat mencapai waktu eksekusi hingga 1,5 detik (Hermawati, 2010). Hal ini menunjukkan bahwa performansi implementasi sistem pada perangkat lunak masih belum ideal untuk aplikasi *real-time*.

Untuk mengatasi kekurangan tersebut maka diperlukan desain perangkat keras khusus yang dapat bekerja sebagai pemroses sistem ANPR. Keuntungan dari desain perangkat keras adalah penghematan biaya dan performansi maksimum. Performansi maksimum dapat dicapai karena perangkat telah didesain secara optimal, khusus untuk pemrosesan ANPR. Selain itu, keseluruhan sistem diimplementasikan cukup pada sebuah rangkaian terintegrasi, sehingga didapat keunggulan dari segi dimensi, konsumsi daya dan harga.

FPGA merupakan rangkaian terintegrasi yang dapat dikonfigurasi oleh pengguna setelah dibuat (Zhai, 2013:6). Keunggulan FPGA di antaranya yaitu mendukung komputasi

paralel dan penghematan biaya produksi, sehingga FPGA adalah alat yang tepat untuk desain perangkat keras sistem yang *low-cost* dan efisien (Zhai, 2013:14).

Secara umum teknik ANPR dibagi menjadi 3 tahap, yaitu lokalisasi plat nomor, segmentasi karakter, dan pengenalan karakter optis (Zhai, 2013:2). Untuk tahap lokalisasi plat nomor dipilih metode deteksi tepi vertikal (*vertical edge detection*) berdasarkan pada fakta bahwa perubahan kecerahan pada area plat nomor lebih sering terjadi dibandingkan dengan area lainnya (Zhai, 2013:16). Untuk tahap segmentasi karakter dipilih metode analisis proyeksi untuk citra plat nomor yang telah dikonversi menjadi citra biner berdasarkan pada fakta bahwa warna dasar plat nomor kontras dengan warna karakter plat nomor, dan hanya 2 warna tersebut yang mendominasi warna plat nomor (Zhai, 2013:23). Untuk tahap pengenalan karakter optis dapat dipilih metode *K-Nearest Neighbor* (KNN) dengan nilai K = 1. Metode ini dilakukan dengan membandingkan citra masukan dengan citra referensi yang ada, kemudian dari nilai perbandingan tersebut citra diklasifikasikan berdasarkan citra referensi yang paling mendekati citra input (Martinsky, 2007:2). Metode ini sederhana dan tidak membutuhkan proses komputasi yang rumit, sehingga cocok untuk diaplikasikan pada FPGA.

Untuk dapat mengimplementasikan metode-metode tersebut sebagai sebuah sistem ANPR pada FPGA, maka dapat diilustrasikan sebuah *state diagram* yang menggambarkan tahapan proses sistem secara keseluruhan, kemudian pada skala yang lebih kecil setiap metode tersebut diterapkan dengan merancang blok diagram rangkaian masing-masing. Sehingga pada akhirnya didapatkan sistem yang dapat bekerja untuk ketiga tahap ANPR pada FPGA.

Pada penelitian ini dirancang sebuah sistem ANPR berbasis FPGA untuk mendeteksi plat nomor kendaraan. Perancangan yang dilakukan adalah merancang sistem ANPR dengan kamera untuk mengambil citra dan FPGA sebagai pengontrol utama, pemroses citra dan unit penyimpanan. Perancangan inti dari sistem ANPR adalah merancang *state diagram* dan diagram blok rangkaian untuk dapat mengimplementasikan ketiga tahap ANPR pada FPGA.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dibahas sebelumnya, maka dapat disusun rumusan masalah sebagai berikut:

1. Bagaimana merancang sistem yang dapat menerima input citra dari kamera ke FPGA?
2. Bagaimana mengaplikasikan teknik *Automatic Number Plate Recognition* pada FPGA?
3. Bagaimana tingkat akurasi sistem yang telah dirancang?

1.3 Batasan Masalah

Dengan mengacu pada rumusan masalah yang telah dibuat, maka hal-hal yang berkaitan dengan perancangan akan diberi batasan sebagai berikut:

1. Pengambilan citra dilakukan secara tidak langsung.
2. Sampel yang dipakai adalah plat nomor standar Indonesia dengan warna dasar hitam.

1.4 Tujuan

Tujuan dari penelitian ini adalah merancang dan melakukan analisis akurasi sistem identifikasi kendaraan otomatis menggunakan teknik *Automatic Number Plate Recognition* berbasis FPGA untuk mendeteksi plat nomor kendaraan.

BAB II
TINJAUAN PUSTAKA

Tinjauan pustaka berisi penjelasan dan uraian dari teori penunjang yang digunakan dalam perancangan berikut ini. Di sini akan dibahas mengenai teknik ANPR beserta metode-metode yang ada, FPGA dengan fitur-fiturnya, serta teori-teori umum yang mendukung.

2.1 Citra Digital

Citra dapat didefinisikan sebagai fungsi 2 dimensi $f(x, y)$, dengan x dan y adalah posisi koordinat. Nilai amplitudo f pada titik (x, y) adalah intensitas atau tingkat keabuan citra pada titik tersebut. Jika x, y, dan nilai amplitudo f adalah kuantitas diskrit yang berhingga, maka citra tersebut dinyatakan sebagai citra digital. Setiap elemen $f(x, y)$ disebut dengan piksel (Gonzalez & Woods, 2008:1).

Citra digital dapat dinyatakan sebagai matriks dengan jumlah baris M dan jumlah kolom N. x dinyatakan sebagai bilangan bulat positif dengan $0 \leq x \leq M-1$ dan y dinyatakan sebagai bilangan bulat positif dengan $0 \leq y \leq N-1$. Representasi matriks berukuran $M \times N$ dinyatakan dalam bentuk persamaan sebagai berikut:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & & \\ \vdots & & \ddots & \vdots \\ f(M-1,0) & & \cdots & f(M-1,N-1) \end{bmatrix} \quad (2\text{-}1)$$

2.2 Format Warna YUV

Standar format warna YUV dibuat oleh *European Broadcast Union* (EBU). Standar ini dipakai pada sistem televisi warna PAL dan SECAM. Standar ini dibuat supaya kompatibel dengan televisi hitam putih. Dalam perkembangannya, format warna YUV juga dipakai untuk konversi citra digital dari citra berwarna menjadi citra hitam putih / *grayscale* (Pratt, 2007:81).

Format RGB membagi informasi warna menjadi 3 elemen warna yaitu elemen R (warna merah), G (warna hijau), dan B (warna biru). Sedangkan format YUV membagi informasi warna menjadi 3 elemen warna yaitu elemen Y (*brightness/luminance*) serta elemen U dan V (*color/chrominance*). Untuk mendapatkan nilai tiap elemen warna YUV dari warna RGB digunakan persamaan-persamaan berikut:

$$Y = 0{,}29889531R + 0{,}58662247G + 0{,}11448223B \quad (2\text{-}2)$$

$$U = \frac{B-Y}{2.03} \quad (2\text{-}3)$$

$$V = \frac{R-Y}{1.14} \quad (2\text{-}4)$$

Pada penelitian ini format yang digunakan adalah format warna YUV. Hanya elemen Y saja yang akan diproses pada sistem, karena untuk metode yang akan digunakan hanya membutuhkan informasi tingkat kecerahan. Dengan kata lain, citra yang akan diproses merupakan citra *grayscale*.

2.3 Teknik Automatic Number Platform Recognition (ANPR)

Automatic Number Plate Recognition (ANPR) memiliki banyak istilah alternatif, seperti *Automatic License Plate Recognition* (ALPR), *Automatic Vehicle Identification* (AVI), *Car Plate Recognition* (CPR) dan *License Plate Recognition* (LPR). Seluruh istilah tersebut mengacu pada pengenalan karakter optis menggunakan kamera untuk membaca plat nomor pada kendaraan. ANPR ditemukan pada tahun 1976 oleh *Home Office Scientific Development Branch* di Inggris (saat ini dikenal sebagai *Home Office Centre for Applied Science and Technology*, CAST) dan telah sukses untuk beberapa kasus, seperti mendeteksi kendaraan yang dicuri, atau mendeteksi kendaraan yang melanggar aturan. Contohnya adalah zona "*Ring of Steel*" yang berada di kota London, Inggris. Pada zona tersebut terdapat sistem ANPR dengan 1.500 kamera pendeteksi plat nomor di seluruh area yang aktif bekerja 24 jam penuh (Zhai, 2013:1). Pada Gambar 2.1 diilustrasikan mengenai aplikasi dari ANPR.

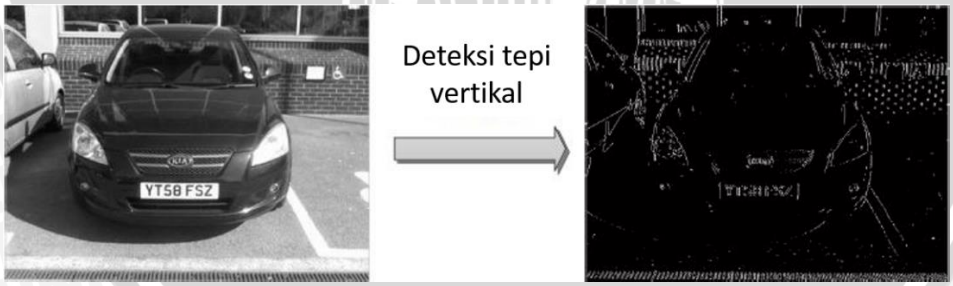
Gambar 2.1 Aplikasi ANPR
Sumber: Zhai (2013:1)

Secara umum teknik ANPR dibagi menjadi 3 tahap, yaitu lokalisasi plat nomor (*Number Plate Localisation*), segmentasi karakter (*Character Segmentation*), dan pengenalan karakter optis (*Optical Character Recognition*) (Zhai, 2013:2). Setiap tahap akan dibahas secara lebih detail. Pada setiap tahap juga dijelaskan metode-metode yang akan dipakai untuk tahap tersebut.

2.3.1 Lokalisasi Plat Nomor dengan Metode Deteksi Tepi Vertikal

Pada tahap ini dilakukan pemrosesan citra sehingga dapat dideteksi area plat nomor. Persoalan yang dihadapi adalah bagaimana menentukan area persegi panjang plat nomor dari citra aslinya. Performansi pada tahap ini sangat penting untuk sistem keseluruhan, karena mempengaruhi akurasi dan efisiensi pada tahap-tahap berikutnya (Zhai, 2013:16).

Pada citra kendaraan bermotor, perubahan kecerahan pada area plat nomor lebih sering terjadi dibandingkan dengan area lainnya. Deteksi tepi (*edge detection*) adalah proses untuk mendeteksi perubahan tingkat kecerahan tersebut. Deteksi tepi dilakukan dengan menghitung selisih kecerahan di area lokal citra (Zhai, 2013:16). Jika dilakukan deteksi tepi untuk mengetahui area plat nomor, maka dapat dilakukan analisis untuk mengetahui letak area plat nomor tersebut. Atas dasar tersebut maka pada tahap ini digunakan metode deteksi tepi vertikal (*vertical edge detection*). Tepi yang dideteksi adalah tepi vertikal karena tepi vertikal untuk karakter plat nomor lebih banyak terdeteksi daripada objek lainnya. Untuk mendeteksi area plat nomor, setelah dilakukan metode deteksi tepi vertikal dilakukan metode analisis proyeksi horizontal dan vertikal (Martinsky, 2007:7). Supaya lebih jelas dapat dilihat contoh hasil proses deteksi tepi vertikal pada Gambar 2.2.



Gambar 2.2 Hasil proses deteksi tepi vertikal (*vertical edge detection*)
Sumber: Zhai (2013:17)

2.3.2 Segmentasi Karakter dengan Metode Pengambangan dan Analisis Proyeksi

Tahap ini merupakan tahap *pre-processing* yang penting untuk tahap selanjutnya. Setelah mendapatkan citra area plat nomor, maka setiap karakter dari citra tersebut disegmentasi. Karakter yang tersegmentasi dengan benar dan akurat akan lebih memungkinkan untuk dapat dikenali dengan benar (Zhai, 2013:22).

Warna dasar plat nomor kontras dengan warna karakter plat nomor, dan hanya 2 warna tersebut yang mendominasi warna plat nomor. Jika plat nomor yang dideteksi hanya dibatasi pada plat dasar hitam tulisan putih, maka nilai keabuan yang tinggi (mendekati putih) akan terdeteksi pada tiap karakter dari plat nomor. Sehingga untuk memperjelas batas-batas setiap karakter dapat dilakukan proses pengambangan (*thresholding*). Proses ini akan memisahkan bagian karakter dengan bagian hitam plat (Martinsky, 2007:20). Atas dasar tersebut maka metode ini digunakan pada segmentasi karakter plat nomor.

Pengambangan (*global thresholding*) adalah pemrosesan citra digital dengan mengkonversi citra menjadi citra biner (*binary image*), yaitu citra yang hanya memiliki 2 jenis informasi warna (hitam dan putih) menurut nilai ambang global tertentu. Jika nilai sebuah piksel di bawah nilai ambang C_T, maka piksel baru bernilai 0 / hitam, jika sebaliknya maka piksel baru bernilai 1 / putih (Martinsky, 2007:27). Persamaan dari metode pengambangan adalah sebagai berikut:

$$T(x,y) = \begin{cases} 1 \; jika \; f(x,y) \geq C_T \\ 0 \; lainnya \end{cases} \quad (2\text{-}5)$$

dengan $T(x,y)$ adalah nilai piksel baru hasil proses pengambangan.

Setelah dilakukan metode pengambangan dilakukan metode analisis proyeksi horizontal dan vertikal. Metode ini dilakukan dengan menjumlahkan piksel untuk satu kolom atau baris untuk mendapatkan dua proyeksi (horizontal dan vertikal), kemudian dianalisis berdasarkan histogram proyeksi untuk mengetahui batas-batas segmentasi karakter (Zhai, 2013:23). Ilustrasi proses analisis proyeksi ditunjukkan pada Gambar 2.3. Persamaan proyeksi horizontal dan vertikal adalah sebagai berikut:

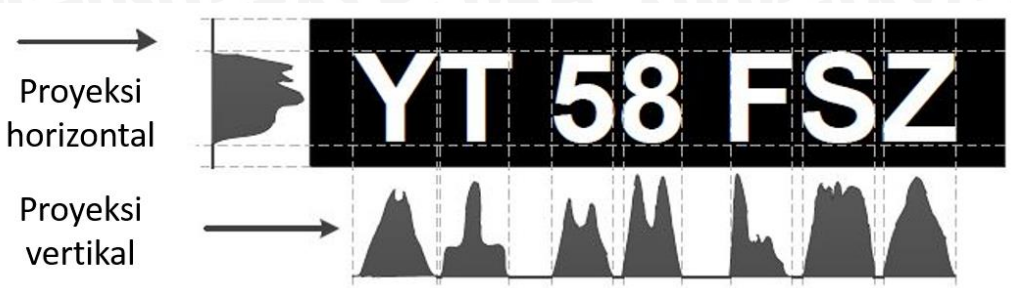
$$p_x(x) = \sum_{j=0}^{h-1} f(x,j) \quad (2\text{-}6)$$

$$p_y(y) = \sum_{i=0}^{w-1} f(i,y) \quad (2\text{-}7)$$

dengan $p_x(x)$ adalah nilai proyeksi horizontal dan $p_y(y)$ adalah nilai proyeksi vertikal.

2.3.3 Pengenalan Karakter Optis dengan Metode *K-Nearest Neighbor*

Teknologi pengenalan karakter optis digunakan untuk mengenali karakter yang telah tersegmentasi sebelumnya pada tahap akhir dari sistem ANPR (Zhai, 2013:22). Setelah mendapatkan citra karakter plat nomor, maka citra tersebut diidentifikasi satu persatu menjadi teks plat nomor. Metode yang akan dipakai adalah metode *K-Nearest Neighbor*. Sebelum dilakukan metode tersebut, dilakukan *pre-processing* dengan mengubah dimensi citra input.



Gambar 2.3 Ilustrasi proses analisis proyeksi
Sumber: Zhai (2013:23)

1) Metode K-Nearest Neighbor (KNN)

Metode *K-Nearest Neighbor* adalah metode klasifikasi data berdasarkan jarak tetangga terdekat antara data input dengan data *set* referensi yang tersedia. Dengan kata lain, metode ini adalah metode untuk mencocokkan sebuah data dengan data yang dianggap paling mirip pada referensi.

Sebagai contohnya sebuah data yang didefinisikan dengan 2 variabel, yaitu sebuah titik (x, y) pada bidang xy (2 dimensi). Tiap dimensi mewakili sebuah variabel. Untuk menghitung jarak dari sebuah titik ke titik lainnya dapat diketahui menggunakan persamaan *Euclidean distance*:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (2\text{-}8)$$

Pada kasus klasifikasi citra digital, maka setiap piksel pada citra mewakili sebuah variabel. Karena itu, kedua data yang dibandingkan harus memiliki dimensi yang sama. Untuk menghitung nilai jarak / *distance* supaya lebih sederhana dapat menggunakan persamaan *distance* absolut seperti berikut ini:

$$d = \sum_{y=0}^{h-1} \sum_{x=0}^{w-1} |f_s(x, y) - f_i(x, y)| \quad (2\text{-}9)$$

dengan $f_s(x, y)$ adalah nilai sebuah piksel pada citra referensi, sedangkan $f_i(x, y)$ adalah nilai piksel citra input.

Selanjutnya proses klasifikasi dilakukan dengan mengambil K data dengan nilai d terkecil dari seluruh perbandingan dengan data referensi. Data input akan diklasifikasikan sesuai klasifikasi terbanyak dari K data sampel referensi. Jika nilai $K = 1$, maka data input akan diklasifikasikan sesuai dengan klasifikasi data dengan nilai *d* terkecil (Peterson, 2009).

2) Metode Weighted Average Downsampling

Metode *Weighted Average Downsampling* merupakan metode untuk mengubah dimensi dari citra digital. Setiap piksel baru akan dihitung berdasarkan sejumlah piksel yang letaknya

bersesuaian pada citra yang lama. Persamaan untuk menghitung piksel baru adalah sebagai berikut:

$$f'(x',y') = \frac{1}{(x_{max}-x_{min})(y_{max}-y_{min})} \sum_{j=y_{min}}^{y_{max}} \sum_{i=x_{min}}^{x_{max}} f(i,j) \quad (2\text{-}10)$$

$$x_{min} = \left|\frac{x\prime}{r_x}\right| \quad (2\text{-}11)$$

$$y_{min} = \left|\frac{y\prime}{r_y}\right| \quad (2\text{-}12)$$

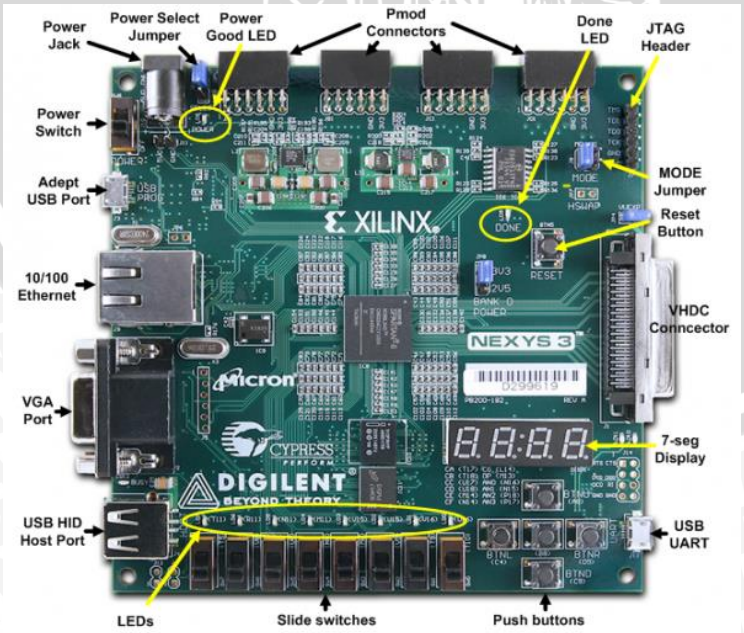
$$x_{max} = \left|\frac{x'+1}{r_x}\right| \quad (2\text{-}13)$$

$$y_{max} = \left|\frac{y'+1}{r_y}\right| \quad (2\text{-}14)$$

dengan r_x adalah rasio horizontal dan r_y adalah rasio vertikal citra yang telah diubah dimensinya (Martinsky, 2007:30). Metode ini akan dipakai untuk menyamakan dimensi antara citra input dan citra referensi.

2.4 *Board* Nexys3 FPGA Xilinx Spartan-6 XC6XLS16

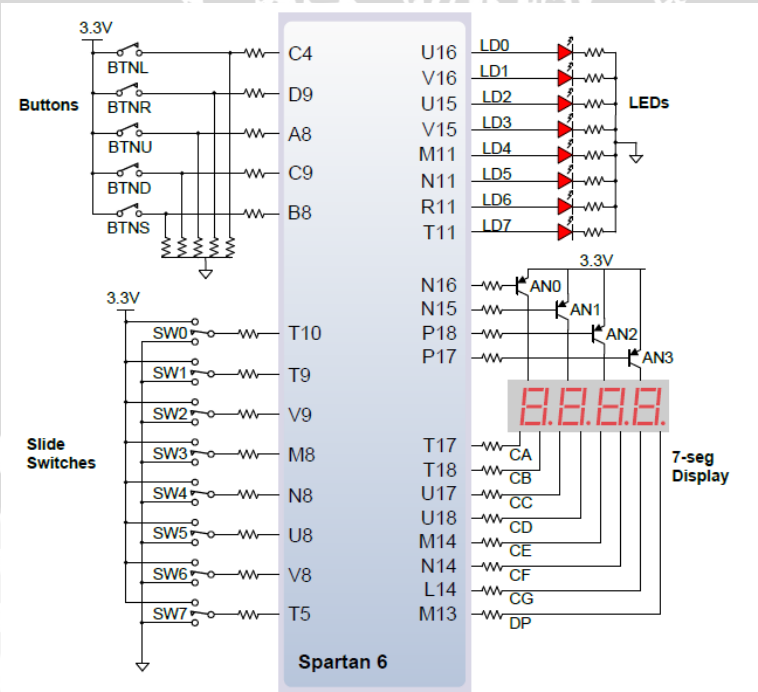
Field Programmable Gate Arrays (FPGA) adalah komponen elektronika dan semikonduktor yang mempunyai komponen gerbang terprogram dan sambungan rangkaian terprogram. Komponen gerbang terprogram yang dimiliki meliputi jenis gerbang logika dasar (AND, OR, XOR, NOT) maupun jenis fungsi matematis dan kombinasional yang lebih kompleks (*decoder*, *adder*, *subtractor*, *multiplier*, dll). Blok-blok komponen di dalam FPGA juga mengandung elemen memori mulai dari *flip-flop* hingga RAM.



Gambar 2.4 *Board* Nexys3
Sumber: Digilent (2011:3)

Board Nexys3 adalah *development board* produk dari Xilinx, berbasis FPGA Xilinx Spartan-6 XC6XLS16. *Board* ini juga dilengkapi dengan *input/output* dasar berupa konektor, port, *switch*, *push button*, LED, *seven-segment-display* dan sebagainya untuk memudahkan antarmuka dengan pengguna. Konfigurasi pin *input/output* dasar dapat dilihat pada Gambar 2.5. Arsitektur FPGA Xilinx Spartan-6 terdiri dari bagian fungsional berikut:

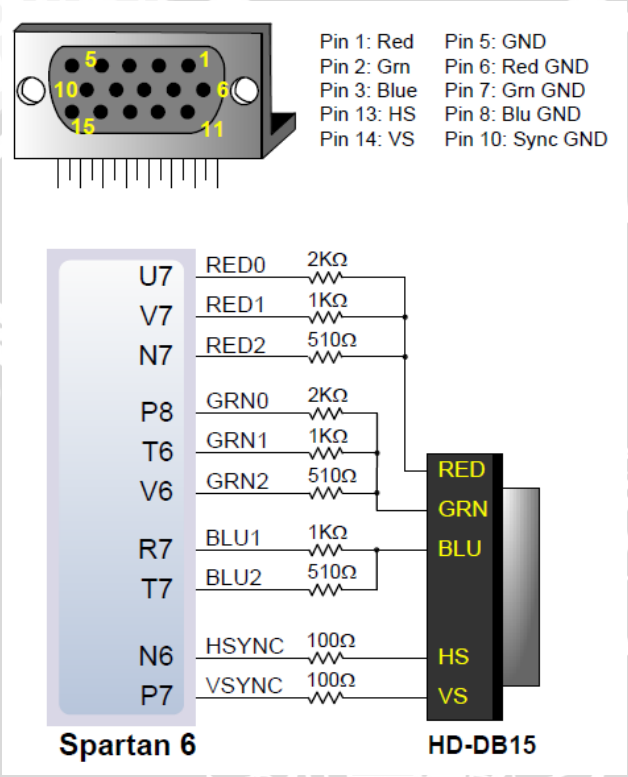
1. *Configurable Logic Block* (CLB), setiap CLB mempunyai 2 *slice*, masing-masing *slice* memiliki 4 *Look-Up Tables* (LUT) dan 8 *flip-flop*. LUT dapat mengimplementasikan fungsi logika dengan 6 input 1 output atau 5 input 2 output (Xilinx, 2011).
2. *Input/Output*, mengatur aliran data antara pin *input/output* dan logika internal rangkaian.
3. *Block* RAM, menyediakan penyimpanan data dalam bentuk blok. Tiap blok memori adalah *dual-port block* RAM yang berkapasitas 18 Kb. Untuk tipe XC6XLS16 memiliki jumlah total 32 blok memori.
4. *Memory Controller Block*, untuk memudahkan akses kontrol memori.
5. *Clock Management Tile* (CMT), mendistribusikan, menunda, menjamak, membagi, dan menggeser fase sinyal *clock*.
6. DSP48A1 *Slice*, untuk aplikasi operasi aritmatika dan *Digital Signal Processing* (DSP) (Xilinx, 2011).



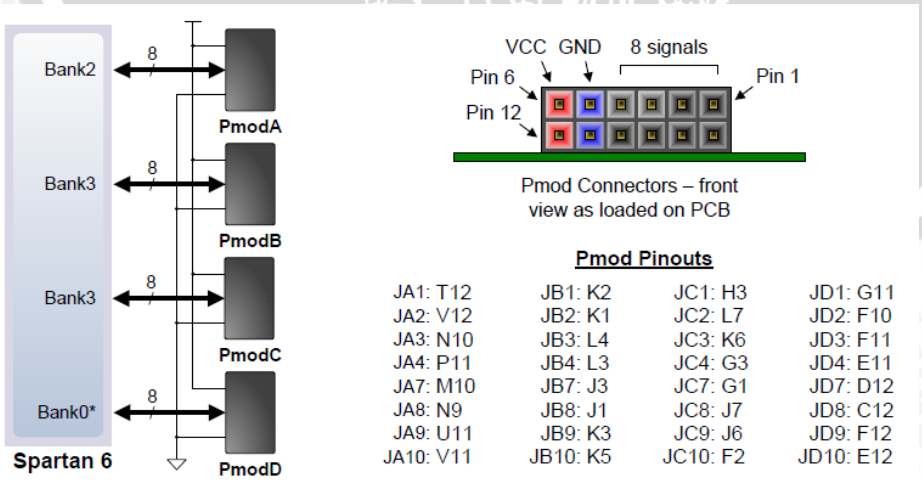
Gambar 2.5 Konfigurasi pin *input/output* dasar pada *Board* Nexys3
Sumber: Digilent (2011:18)

Board Nexys3 memiliki port VGA dengan 10 sinyal, yaitu 8 sinyal warna (3 bit sinyal warna merah, 3 bit sinyal warna hijau dan 2 bit sinyal warna biru), sinyal HS (*Horizontal*

Sync) dan VS (*Vertical Sync*). Konfigurasi pin port VGA dapat dilihat pada Gambar 2.6. Pada *board* Nexys3 juga terdapat konektor Pmod. Konektor ini berupa 12 konektor *female*, terdiri dari 2 sinyal VCC 3,3 V, 2 sinyal GND, dan 8 sinyal logika. Konektor ini dapat digunakan untuk menyambungkan pin-pin pada sensor kamera OV7670. Konfigurasi pin konektor Pmod dapat dilihat pada Gambar 2.7. Sumber tegangan *board* Nexys3 berasal dari port USB Adept, yang juga berfungsi untuk memprogram FPGA. Tabel 2.1 menunjukkan sumber tegangan pada rangkaian *board* Nexys3.



Gambar 2.6 Konfigurasi pin Port VGA pada *Board* Nexys3
Sumber: Digilent (2011:15)



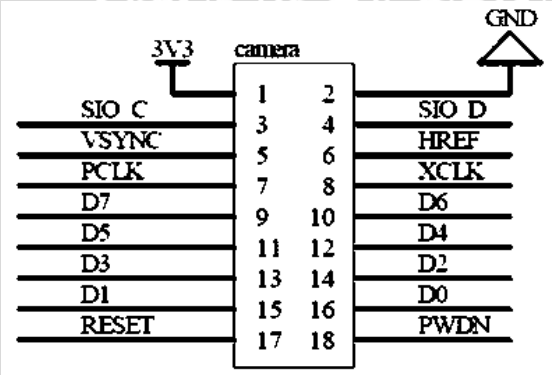
Gambar 2.7 Konfigurasi pin konektor Pmod pada *Board* Nexys3
Sumber: Digilent (2011:21)

Tabel 2.1 Sumber tegangan rangkaian pada *board* Nexys3

Supply	Circuits	Device	Amps (max/typ)
3.3V	FPGA I/O, USB ports, Clocks, ROM & RAM I/O, Ethernet	IC13: LTC3633	3A / 200mA
2.5V	Optional voltage for Bank0 and VHDC connector	IC14: LTC3619	800mA / 0mA
1.2V	FPGA Core	IC13: LTC3633	3A / 0.2 to 1.0A
1.8V	RAM and ROM core	IC14: LTC3619	400mA / 0.1 to 0.3A

2.5 Sensor Kamera OV7670

Sensor kamera OV7670 adalah sensor kamera VGA dengan ukuran *array* sensor citra aktif 640×480. Sensor dapat bekerja dengan *frame rate* maksimum 30 fps. Pengaturan kamera dapat dilakukan melalui antarmuka SCCB. Pengaturan ini dapat berupa format warna, resolusi, *frame rate*, pemrosesan citra, dan sebagainya. Gambar 2.8 menunjukkan konfigurasi pin modul sensor kamera OV7670. Penjelasan konfigurasi pin ditunjukkan pada Tabel 2.2.

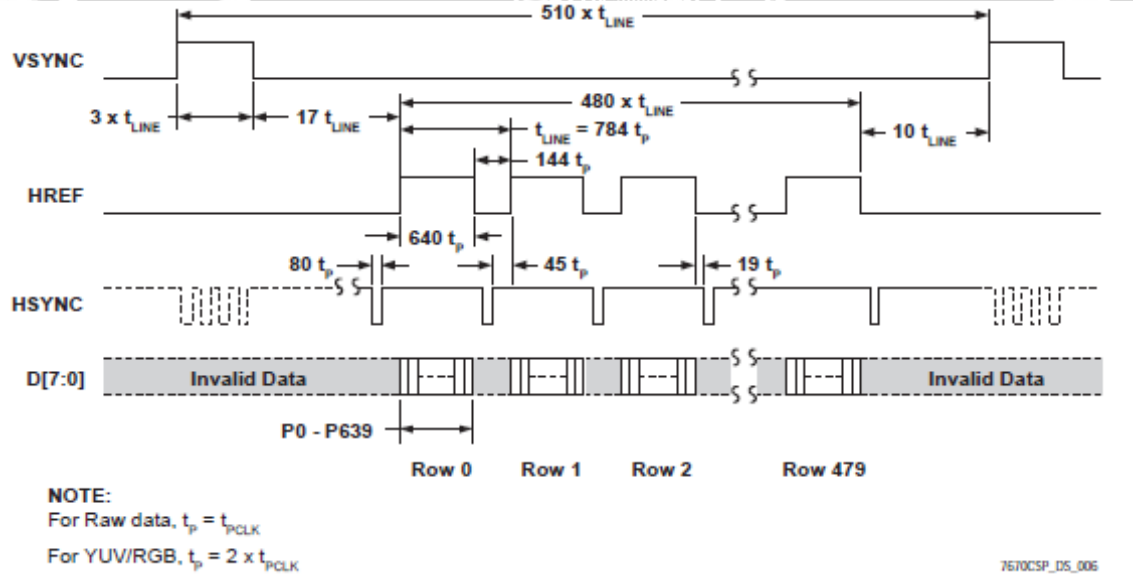


Gambar 2.8 Konfigurasi pin modul sensor kamera OV7670
Sumber: Electrodragon (2013)

Tabel 2.2 Penjelasan Konfigurasi Pin modul sensor kamera OV7670

Nama Pin	Fungsi
3V3	Sumber tegangan sebesar 3,3 V
GND	*Ground*
SIO_C	*Clock* pengontrol antarmuka SCCB
SIO_D	*Input/Output* data serial antarmuka SCCB (*bidirectional*)
VSYNC	Sinyal sinkronisasi *frame*
HREF	Sinyal penanda pengiriman data aktif untuk tiap baris
PCLK	*Pixel clock*
D0-D7	Port data piksel
RESET#	Sinyal *reset* (aktif saat *low*)
PWDN	Sinyal seleksi *Power Down Mode*

Pengiriman data sensor kamera OV7670 dilakukan secara *parallel synchronous*. Untuk mendapat data dari OV7670, pin XCLK diberikan masukan *clock* dengan frekuensi antara 10 – 48 MHz. Setelah itu maka OV7670 akan mulai menjalankan VSYNC, HREF dan D0-D7. Data D0 – D7 diambil saat tepi naik dari sinyal PCLK, dan saat sinyal dari pin HREF berlogika *high*. Tepi naik dari HREF menandai mulainya suatu baris, dan tepi turun dari sinyal HREF menandai akhir dari baris. Semua byte data yang tersampel saat HREF berlogika *high* merupakan data semua piksel dalam satu baris. Satu *frame* didapat saat VSYNC berlogika *low*. Tepi turun dari VSYNC menunjukkan awal dari *frame* dan tepi naik dari VSYNC menunjukkan akhir dari sebuah *frame*. Gambar 2.9 menunjukkan *timing diagram* pengiriman *frame* VGA. Dari Gambar 2.9 dapat diketahui *frame rate* dari kamera berdasarkan frekuensi PCLK dan format warna yang dipakai (YUV), yaitu sebesar PCLK / (510 × 784 × 2). Urutan pengiriman data yaitu U Y V Y untuk 2 piksel (4 byte). Karena yang akan diproses hanya elemen Y saja, maka data diambil pada pengiriman data Y untuk setiap piksel.



Gambar 2.9 *Timing diagram frame* VGA sensor kamera OV7670
Sumber: OmniVision Technologies (2006:7)

Pada Tabel 2.1 dapat dilihat bahwa sumber tegangan pada port USB dan *input/output* FPGA adalah sebesar 3,3 V. Karena sumber tegangan untuk modul sensor kamera juga sebesar 3,3 V, maka modul dapat langsung disambungkan ke *board* Nexys3 tanpa rangkaian khusus. Perlu diketahui bahwa pada modul telah dirancang rangkaian pengendali sinyal untuk menyesuaikan tegangan pin *input/output* dengan chip (tidak melebihi batas *absolute maximum rating* yang ditunjukkan pada Tabel 2.3), sehingga cukup menggunakan tegangan 3,3 V sesuai dengan pin sumber tegangan modul.

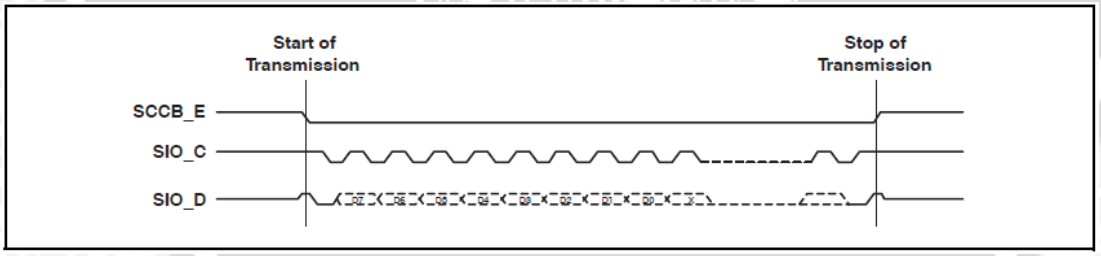
Tabel 2.3 *Absolute Maximum Rating* sensor kamera OV7670

Parameter	Value	Unit
Ambient Storage Temperature	-40 to +95	^{0}C
Supply Voltage (V_{DD-A})	4.5	V
Supply Voltage (V_{DD-C})	3	V
Supply Voltage (V_{DD-IO})	4.5	V
I/O Voltages	-0.3 to V_{DD-IO}+0.5	V
Lead-free temperature, Surface-mount process	245	^{0}C

2.6 Antarmuka SCCB

Antarmuka SCCB (*Serial Camera Control Bus*) adalah antarmuka yang digunakan oleh sensor kamera OV7670 (sebagai *slave device*) untuk berkomunikasi dengan *board* Nexys3 (sebagai *master device*). Komunikasi ini berjalan secara serial (mirip I2C) dengan pin-pin yang digunakan adalah SIO_C, SIO_D, dan PWDN. Seperti yang telah dijelaskan sebelumnya, antarmuka SCCB berfungsi untuk melakukan pengaturan kamera. Pengaturan ini dilakukan dengan cara memberi atau mengubah data pada *register* internal kamera (*Device Control Registers*). *Register* inilah yang menentukan bagaimana kamera akan bekerja.

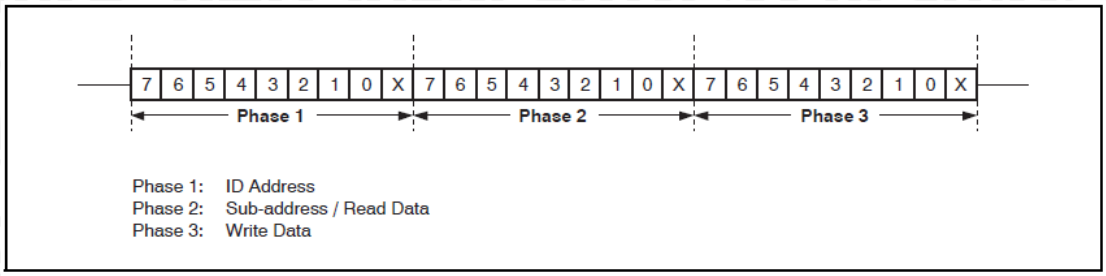
Komunikasi berjalan melalui pin SIO_C dan SIO_D. Pada saat komunikasi tidak berlangsung (*idle*), pin SIO_C diberi nilai logika 1 dan pin SIO_D *tri-state*. Transmisi data dimulai saat SIO_C diberi logika 0 untuk pertama kalinya. Pada saat transmisi data, transmisi bit dari sinyal SIO_D ditandai dengan SIO_C bernilai 1. Sehingga pergantian tiap bit pada SIO_D dilakukan saat SIO_C bernilai 0. Periode minimum SIO_C adalah 10 µs (frekuensi maksimum 100 kHz). Supaya lebih jelas dapat dilihat pada Gambar 2.10.



Gambar 2.10 *Timing diagram* transmisi data antarmuka SCCB
Sumber: OmniVision Technologies (2003:7)

Pada setiap transmisi data terdapat fase transmisi. Setiap fase transmisi adalah 9 bit transmisi, dengan 8 bit data dan bit ke-9 dapat berupa bit *Don't Care* atau bit NA. Transmisi data bertujuan untuk melakukan operasi baca (*read*) atau tulis (*write*) pada *register* internal. Untuk melakukan operasi tulis, maka dilakukan transmisi data 3 fase (*ID Address*, *Sub-*

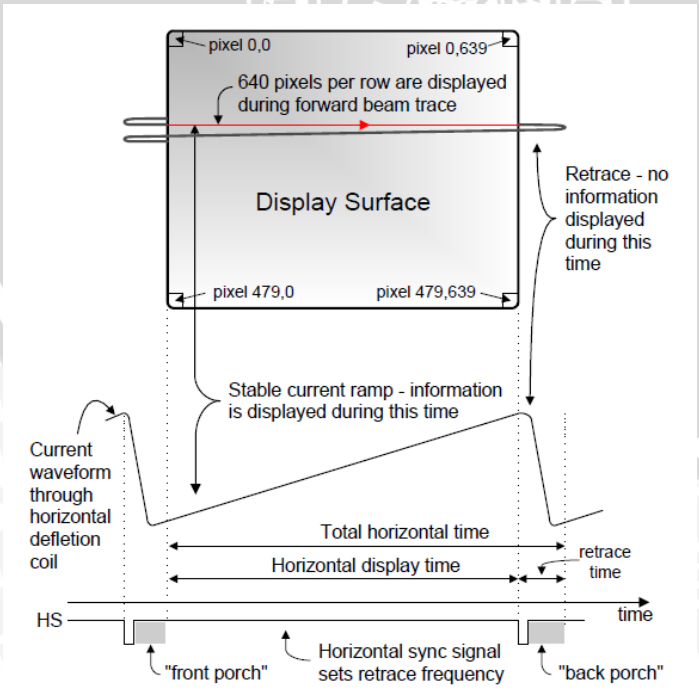
address dan *Write Data*). Sedangkan untuk operasi baca, maka dilakukan transmisi data 2 fase sebanyak 2 kali (*ID Address* dan *Sub-address*, kemudian *ID Address* dan *Read Data*).



Gambar 2.11 Fase transmisi data antarmuka SCCB
Sumber: OmniVision Technologies (2003:9)

2.7 Monitor VGA

Monitor VGA merupakan media tampilan yang menggunakan standar VGA. Monitor VGA bekerja dengan menggerakkan *electron beam* (dari kiri ke kanan serta atas ke bawah) untuk menampilkan informasi pada layar. Pada konektor VGA terdapat 5 sinyal yang digunakan untuk mengatur tampilan, yaitu sinyal RED (elemen warna merah), GREEN (elemen warna hijau), BLUE (elemen warna biru), HSYNC (*horizontal sync*, untuk mengembalikan posisi ke kiri layar pada baris di bawah yang sebelumnya), dan VSYNC (*vertical sync*, untuk mengembalikan posisi ke kiri atas layar). Resolusi yang akan digunakan adalah 800×600, dengan frekuensi *pixel clock* adalah 50 MHz. Pada Tabel 2.4, 2.5 dan 2.6 disajikan data *timing* untuk monitor VGA yang akan digunakan (SECONS, 2008).



Gambar 2.12 *Display timing* monitor VGA
Sumber: Digilent (2011:16)

Tabel 2.4 *General timing* 800×600 @ 72 Hz

Parameter	Value	Unit
Screen refresh rate	72	Hz
Vertical refresh	48.076923	kHz
Pixel frequency	50	MHz

Tabel 2.5 *Horizontal timing* 800×600 @ 72 Hz

Scanline part	Pixels	Time (µs)
Visible area	800	16
Front porch	56	1.12
Sync pulse	120	2.4
Back porch	64	1.28
Whole line	1040	20.8

Tabel 2.6 *Vertical timing* 800×600 @ 72 Hz

Frame part	Lines	Time (ms)
Visible area	600	12.48
Front porch	37	0.7696
Sync pulse	6	0.1248
Back porch	23	0.4784
Whole frame	666	13.8528

UNIVERSITAS BRAWIJAYA

BAB III
METODE PENELITIAN

Metode penelitian yang digunakan pada skripsi ini adalah metode studi kepustakaan dan penelitian laboratorium. Studi kepustakaan dilakukan sebagai penunjang yang berupa data-data literatur dari masing-masing komponen, informasi dari internet dan konsep-konsep teoritis dari buku-buku penunjang. Penelitian laboratorium berupa perancangan perangkat keras dan perangkat lunak. Langkah-langkah yang dilakukan untuk merealisasikan alat yang dirancang adalah penentuan spesifikasi, perancangan, metode pengujian dan pengujian sistem.

3.1 Spesifikasi Perancangan Sistem

Penentuan spesifikasi sistem bertujuan agar dapat dibuat sesuai yang diinginkan dan dapat bekerja dengan efektif serta efisien. Dalam perancangan ini, spesifikasi yang dibutuhkan adalah sebagai berikut:

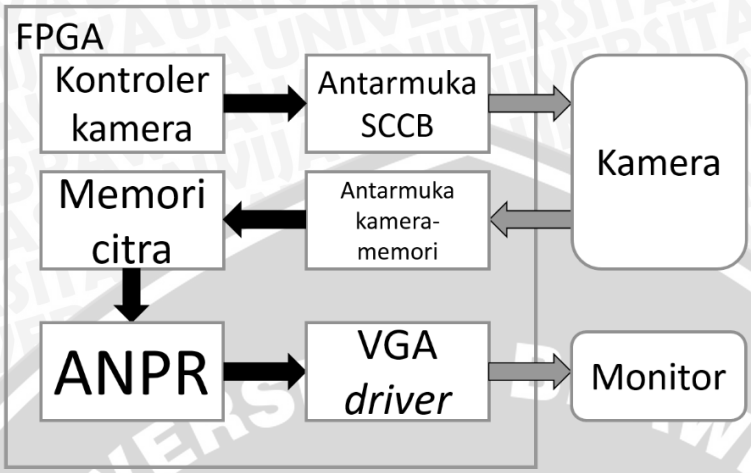
1. Sensor kamera OV7670 berfungsi mengambil citra dengan *setting* kamera 640 × 480 piksel YUV dan *frame rate* 15 fps.
2. *Board* Nexys3 FPGA Xilinx Spartan-6 XC6SLX16 berfungsi menjalankan sistem secara keseluruhan.
3. Monitor VGA berfungsi menampilkan data pemrosesan yang kemudian dapat dianalisis.
4. Pemrograman pada komputer menggunakan *software* Eclipse IDE untuk membuat program simulasi.
5. Pemrograman untuk desain implementasi sistem pada komputer menggunakan *software* Xilinx ISE Project Navigator.

3.2 Perancangan Sistem

3.2.1 Diagram Blok

Pembuatan diagram blok merupakan dasar dari perancangan sistem agar perancangan berjalan secara sistematis. Diagram blok ditunjukkan pada Gambar 3.1. Perancangan sistem dibagi menjadi 2, yaitu perancangan perangkat keras dan perancangan perangkat lunak (algoritma dan implementasi sistem). Perancangan perangkat keras membahas tentang

alokasi pin dari kamera dan monitor ke *board* FPGA. Perancangan perangkat lunak membahas tentang program simulasi, perancangan algoritma dan implementasinya pada sistem.



Gambar 3.1 Diagram blok sistem

Dilihat dari blok diagram, perangkat keras terdiri dari FPGA, kamera, dan monitor. Sedangkan blok-blok di dalam FPGA merupakan implementasi sistem. Kamera menerima input dari antarmuka SCCB. Input ini adalah data serial untuk mengatur setting kamera. Kamera memberikan output pada antarmuka kamera-memori. Output berupa data piksel dengan *timing frame* VGA seperti yang telah ditunjukkan pada bab sebelumnya (Gambar 2.9). Blok VGA *driver* memberikan output pada monitor, sehingga monitor dapat memberikan tampilan visual data pemrosesan untuk dianalisis. Prinsip kerja sistem keseluruhan adalah bagaimana FPGA mendapatkan data dari kamera kemudian menyimpannya untuk diproses melalui sistem ANPR, kemudian hasil berupa karakter plat nomor dikirim untuk ditampilkan pada monitor. Oleh karena itu, kerja sistem bergantung pada perancangan implementasi sistem pada FPGA. Blok-blok fungsional di dalam FPGA dapat dijelaskan secara singkat sebagai berikut:

1. Kontroler kamera: mengatur *setting* kamera supaya dapat bekerja mengambil citra sesuai yang diinginkan.
2. Antarmuka SCCB: sebagai antarmuka komunikasi serial ke kamera dengan menerima perintah dari kontroler kamera.
3. Antarmuka kamera-memori: melakukan pemrosesan data piksel dari kamera dan menyalurkannya untuk disimpan pada memori.
4. Memori citra: unit penyimpanan citra.
5. ANPR: Blok sistem ANPR, pemroses citra untuk dikonversi menjadi data karakter plat nomor.

6. VGA *driver*: menerima data hasil pemrosesan dari ANPR dan mengontrol tampilan monitor untuk menampilkannya data tersebut.

3.2.2 Perancangan Perangkat Keras

Perancangan perangkat keras meliputi alokasi pin sensor kamera OV7670 dan port VGA. Pin-pin pada sensor kamera akan disambungkan melalui konektor Pmod (B dan C) pada *board* Nexys3. Monitor akan disambungkan ke *board* Nexys3 melalui port VGA yang telah tersedia. Tabel 3.1 menunjukkan alokasi pin pada *board* Nexys3. Alokasi pin digunakan untuk menentukan pin mana saja yang digunakan pada *board* FPGA supaya sesuai dengan port dan konektor yang akan disambungkan.

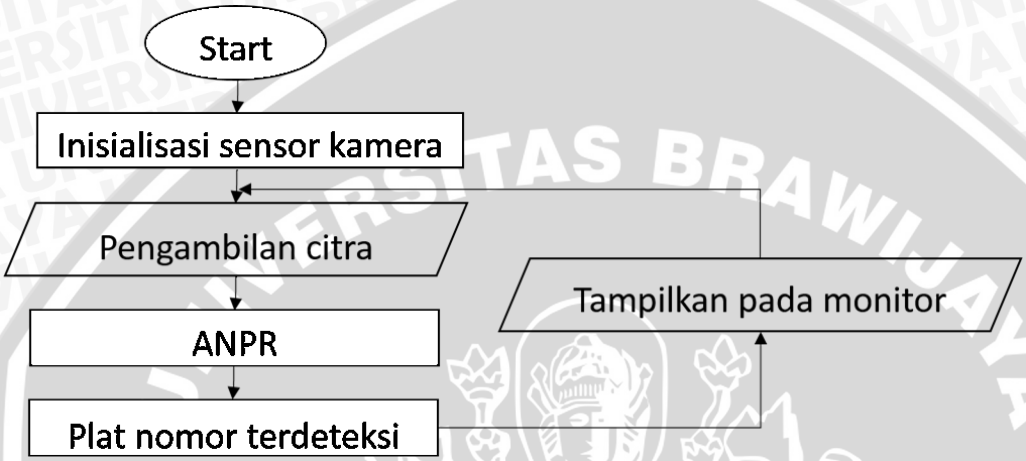
Tabel 3.1 Alokasi pin *board* Nexys3

No.	Unit	Pin	Pin Nexys3	Nama Skematik Pin
1.	Modul sensor kamera OV7670	3V3	3V3	JC12
		GND	GND	JC5
		SIO_C	F2	JC10
		SIO_D	G3	JC4
		VSYNC	J6	JC9
		HREF	K6	JC3
		PCLK	J7	JC8
		D0	K1	JB2
		D1	J1	JB8
		D2	L4	JB3
		D3	K3	JB9
		D4	L3	JB4
		D5	K5	JB10
		D6	H3	JC1
		D7	G1	JC7
		RESET#	J3	JB7
		PWDN	K2	JB1
2.	Monitor VGA	1 (Red)	U7	RED0
			V7	RED1
			N7	RED2
		2 (Grn)	P8	GRN0
			T6	GRN1
			V6	GRN2
		3 (Blue)	R7	BLU1
			T7	BLU2
		13 (HS)	N6	HSYNC
		14 (VS)	P7	VSYNC

3.2.3 Perancangan Algoritma dan Implementasi Sistem pada FPGA

Di sini akan dibahas mengenai bagaimana mendesain sistem sesuai yang diinginkan dengan merancang algoritma yang tepat dan efisien serta implementasinya pada FPGA. Pada Gambar 3.2 ditunjukkan diagram alir yang mendeskripsikan cara kerja keseluruhan dari

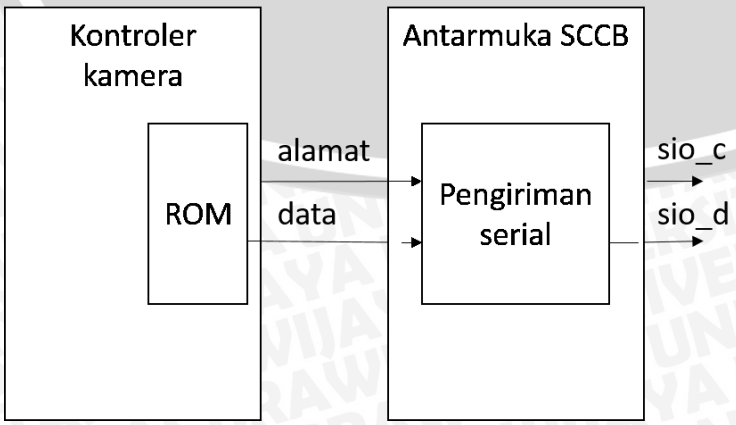
sistem. Diagram alir dari sistem berkaitan langsung dengan diagram blok, karena setiap blok berfungsi untuk melaksanakan setiap proses pada diagram alir. Pada tahap inisialisasi sensor kamera, blok yang bekerja adalah kontroler kamera dan antarmuka SCCB. Kemudian pada tahap pengambilan citra blok yang bekerja adalah antarmuka kamera-memori untuk menulis data pada memori citra. Blok VGA *driver* bekerja menampilkan data pada monitor. Tahap ANPR yang merupakan bagian inti dari sistem akan dibahas secara lebih mendalam. Berikutnya akan dijelaskan fungsi blok-blok pada diagram blok sistem (Gambar 3.1).



Gambar 3.2 Diagram alir sistem keseluruhan

1) Kontroler Kamera dan Antarmuka SCCB

Kedua blok ini berfungsi melakukan inisialisasi pada sensor kamera, yaitu mengatur *setting* pada kamera. Pengaturan kamera dilakukan dengan mengganti nilai *register* internal pada sensor kamera OV7670. Di dalam kontroler kamera terdapat ROM yang berisi pasangan alamat dan data pada *register* kamera. Alamat dan data tersebut dikirimkan secara berurutan ke blok antarmuka SCCB. Hal ini dapat dianalogikan seperti CPU pada komputer yang mengeksekusi program secara berurutan. Setelah didapat alamat dan data, blok antarmuka SCCB mengirimkan ke sensor kamera secara serial sesuai spesifikasi SCCB.



Gambar 3.3 Diagram blok kontroler kamera dan antarmuka SCCB

Sebelum melakukan pengaturan kamera dilakukan *software reset* terlebih dahulu dengan memberikan nilai *register* COM7 (0x12) = 0x80. Pengaturan kamera yang akan diterapkan pada sistem adalah sebagai berikut:

1. Pengaturan *window* dengan resolusi VGA (640×480) dilakukan dengan memberi nilai untuk setiap *register* berikut: HREF (0x32) = 0xF6, HSTART (0x17) = 0x13, HSTOP (0x18) = 0x01, VREF (0x03) = 0x0A, VSTRT (0x19) = 0x02, VSTOP (0x1A) = 0x7A (OmniVision Technologies, 2005).
2. Pengaturan format warna YUV dengan urutan pengiriman data U Y V Y dilakukan dengan memberi nilai untuk setiap *register* berikut: COM7 (0x12) = 0x00 (*default*), TSLB (0x3A) = 0x0C, COM13 (0x3D) = 0xC0 (OmniVision Technologies, 2005).
3. Pengaturan *timing frame rate* yaitu dengan mengatur frekuensi PCLK. PCLK sama dengan frekuensi internal sensor kamera, yang bergantung pada pengaturan di *register* internal serta frekuensi *clock input* (XCLK). XCLK yang diberikan adalah sebesar 25 MHz, supaya mendekati frekuensi tipikal 24 MHz (OmniVision Technologies, 2006). Persamaan untuk mendapatkan frekuensi PCLK adalah sebagai berikut:

   PCLK = XCLK × PLL_Multiplier / Divider

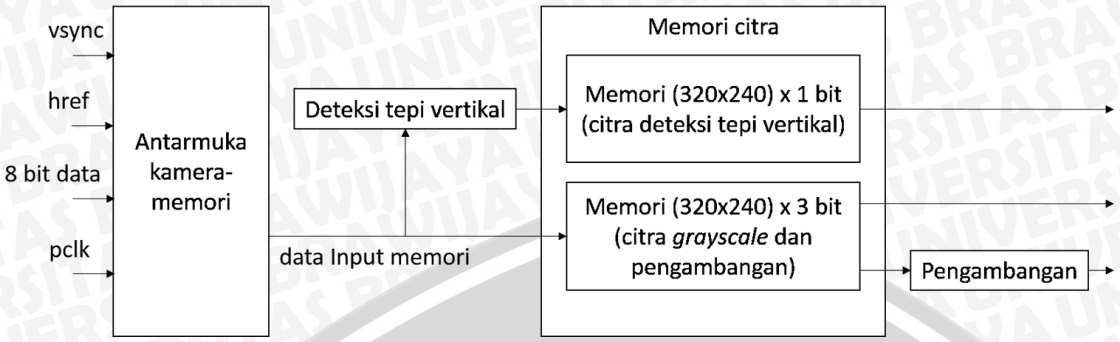
   Dengan memberi nilai pada *register* DBLV (0x6B) = 0x00 (*default*) dan CLKRC (0x11) = 0x00 (*default*) maka didapat nilai PLL_Multiplier = 1 dan Divider = 2, sehingga frekuensi PCLK sebesar 12,5 MHz (OmniVision Technologies, 2005). Maka didapatkan *frame rate* = 12.500.000 / (510 × 784 × 2) ≈ 15 fps.
4. Pengaturan untuk kontrol pemrosesan citra dengan memberi nilai pada beberapa *register* lainnya.

2) Antarmuka Kamera-Memori dan Memori Citra

Blok antarmuka kamera-memori berfungsi untuk mengambil data citra dari kamera dan menulisnya pada memori. Setiap data piksel dikirim dari kamera secara terus menerus tanpa jeda, sehingga proses pada blok ini tidak dapat dijeda hingga seluruh piksel selesai diterima. Oleh karena itu, blok ini segera memproses tiap piksel yang diterima kemudian ditulis pada memori.

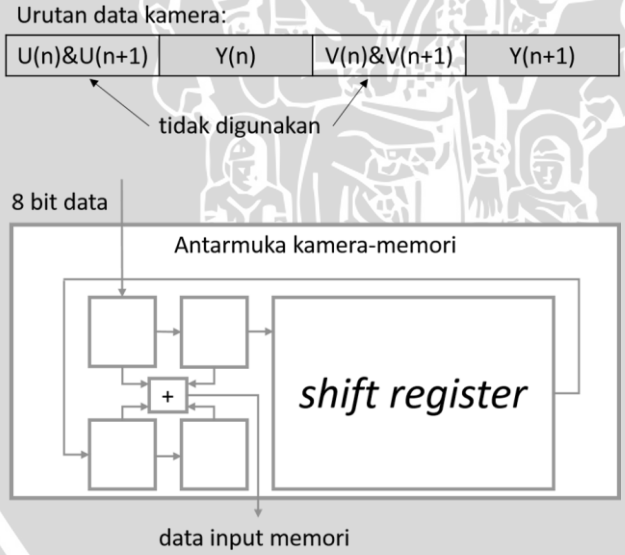
Terdapat 2 blok memori yang akan digunakan pada sistem untuk menyimpan citra, yaitu memori citra berukuran 320×240 × 3 bit serta 320×240 × 1 bit. Pemilihan ukuran 320×240 disebabkan oleh blok memori yang terbatas pada FPGA Spartan-6 (32 × 18 Kb), sehingga dipilih ukuran tersebut untuk menghemat alokasi memori. Supaya mendapat kualitas citra

yang bagus, dilakukan proses *resize* pada citra yaitu setiap piksel yang diterima merupakan hasil rata-rata dari 4 piksel yang diterima disertai teknik *pipelining*.



Gambar 3.4 Diagram blok antarmuka kamera-memori dan memori citra

Seperti diketahui sebelumnya hanya elemen kecerahan / *brightness* saja yang akan diproses, sehingga citra yang akan digunakan adalah citra *grayscale*. Untuk keperluan tampilan pada monitor, diketahui bahwa sinyal port VGA *board* Nexys3 hanya terdiri dari 3 bit merah, 3 bit hijau dan 2 bit biru. Oleh karena itu, dari 8 bit elemen Y yang diterima hanya 3 bit MSB (*Most Significant Bits*) yang akan disimpan (pada memori 320×240 × 3 bit), untuk selanjutnya ditampilkan sama baik pada sinyal merah, hijau maupun biru (untuk sinyal biru hanya 2 MSB).



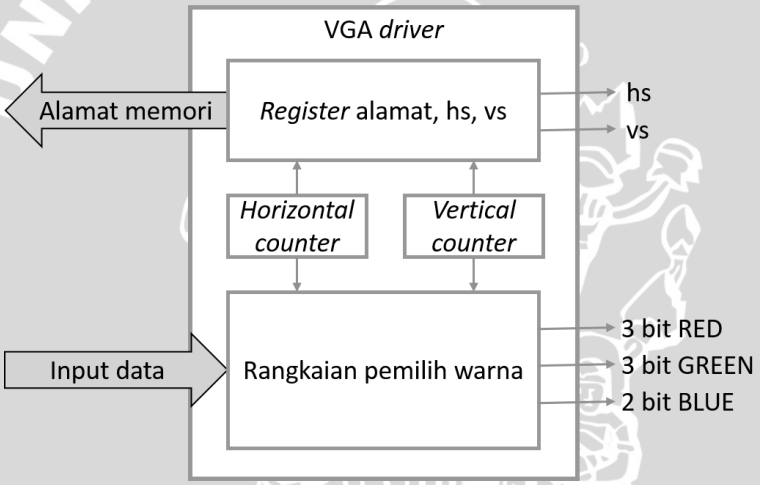
Gambar 3.5 Diagram blok antarmuka kamera-memori (detail)

Untuk keperluan teknik ANPR, diperlukan 2 macam memori untuk menyimpan citra, yaitu untuk proses deteksi tepi vertikal dan pengambangan. Supaya efisien, proses deteksi tepi vertikal segera dijalankan untuk setiap piksel yang diterima dari kamera sehingga segera tersimpan pada memori. Dengan kata lain, proses deteksi tepi vertikal telah dijalankan terlebih dahulu sebelum data disimpan pada memori. Sedangkan untuk proses pengambangan membutuhkan input dari memori citra *grayscale* sebelumnya (320×240 × 3

bit). Pada proses deteksi tepi vertikal hasilnya disimpan pada memori tersendiri (320×240 × 1 bit).

3) VGA *Driver*

Blok ini berfungsi sebagai pengontrol tampilan pada monitor VGA, kemudian monitor akan memberikan tampilan visual data pemrosesan untuk dianalisis. Karena berkaitan dengan analisis dan pengujian, maka blok ini didesain supaya dapat mendukung proses analisis pula. Pada dasarnya, blok ini menerima seluruh input yang diperlukan untuk ditampilkan (baik itu data pada memori, *register* dan sebagainya). Kemudian setiap input akan ditampilkan pada posisi yang diinginkan pada monitor dengan mengatur *timing* yang tepat berdasarkan *counter* posisi piksel pada monitor. Untuk mendapatkan input dari memori, pengalamatan untuk setiap memori juga dilakukan berdasarkan *counter* posisi piksel.



Gambar 3.6 Diagram blok VGA *driver*

4) Metode ANPR

Seperti yang telah dijelaskan sebelumnya bahwa teknik ANPR dibagi menjadi 3 tahap, yaitu lokalisasi plat nomor, segmentasi karakter, dan pengenalan karakter optis. Berikut ini akan dijelaskan secara singkat mengenai metode ANPR yang akan diterapkan berdasarkan tiap tahap tersebut. Kemudian akan diberikan penjelasan yang lebih mendalam mengenai cara kerja setiap bagian dari metode tersebut serta teori yang mendasari.

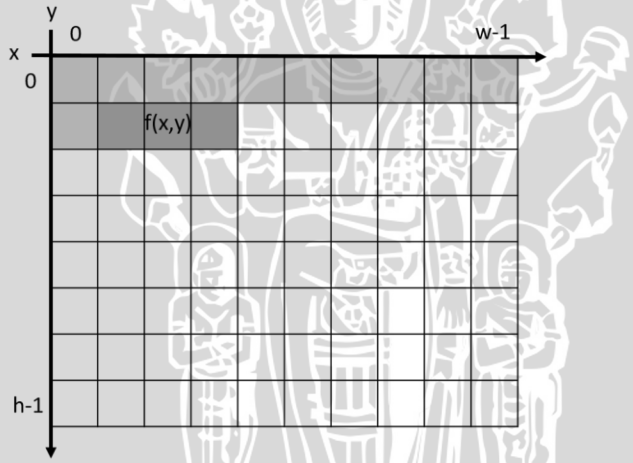
Pada tahap lokalisasi plat nomor dilakukan proses deteksi tepi vertikal (*vertical edge detection*). Kemudian dilanjutkan dengan analisis proyeksi vertikal untuk mengetahui batas vertikal area plat nomor (*band clipping*). Setelah itu dilakukan analisis proyeksi horizontal untuk mengetahui batas horizontal area plat nomor (*plate clipping*).

Pada tahap segmentasi karakter dilakukan proses pengambangan (*thresholding*) pada citra area plat nomor. Kemudian dilakukan analisis yang sama seperti sebelumnya yaitu analisis proyeksi horizontal untuk mengetahui batas-batas horizontal setiap karakter. Kemudian pada tiap karakter dilanjutkan dengan analisis proyeksi vertikal untuk mengetahui batas vertikal karakter tersebut.

Pada tahap pengenalan karakter optis dilakukan proses *resize* citra sehingga sama dengan citra sampel, kemudian dilakukan metode *K-Nearest Neighbor*. Setelah itu seluruh karakter dapat diidentifikasi menjadi teks plat nomor.

4.1) Lokalisasi Plat Nomor

Karena sistem akan diterapkan pada FPGA, maka target aplikasi sistem adalah kecepatan proses optimal dan penggunaan memori minimal supaya sistem dapat memungkinkan untuk diimplementasikan secara keseluruhan. Supaya target tersebut tercapai, maka sedapat mungkin setiap blok rangkaian dasar yang melakukan sebuah bagian proses tertentu dibuat sederhana.



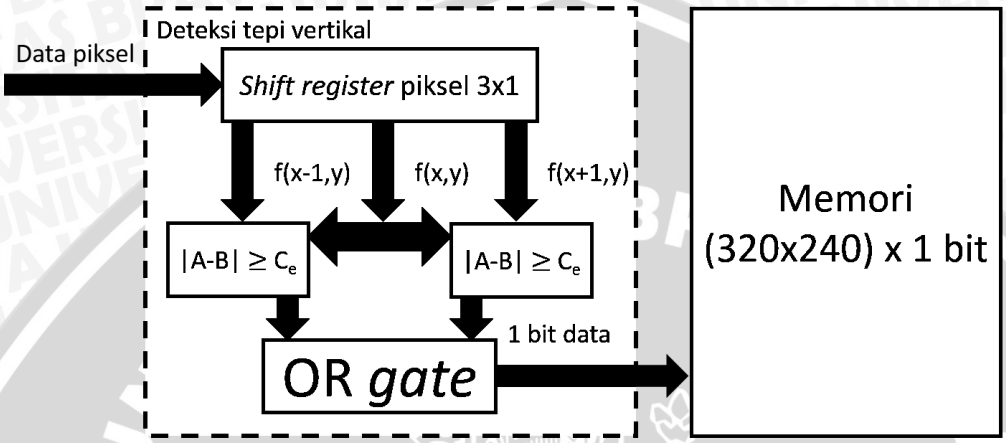
Gambar 3.7 Ilustrasi proses deteksi tepi vertikal

Untuk dapat lebih mudah dalam memahami proses deteksi tepi vertikal, dapat dilihat ilustrasi pada Gambar 3.7. Tiap piksel dibaca dari posisi kiri atas ($x = 0,\ y = 0$) hingga posisi piksel kanan bawah ($x = w - 1,\ y = h - 1$). Untuk mengetahui nilai tepi vertikal suatu piksel, maka digunakan persamaan berikut:

$$e(x,y) = \begin{cases} 1\ jika\ |f(x,y) - f(x-1,y)| \geq C_e \\ 1\ jika\ |f(x,y) - f(x+1,y)| \geq C_e \\ 0\ lainnya \end{cases} \quad (3\text{-}1)$$

dengan $f(x,y)$ adalah nilai kecerahan/keabuan citra pada piksel (x,y), $e(x,y)$ adalah nilai tepi vertikal piksel dan C_e adalah konstanta untuk menentukan sensitivitas deteksi tepi vertikal.

Gambar 3.8 menunjukkan diagram blok implementasi proses deteksi tepi vertikal pada FPGA. Setiap data piksel dibaca satu-persatu, disimpan dalam 3 *shift register* untuk menyimpan data piksel yang akan diproses nilai tepi vertikalnya beserta data piksel yang bersebelahan. Proses berlangsung melalui rangkaian logika yang terdiri pengurang (*subtractor*), pembanding (*comparator*) dan gerbang OR (OR *gate*). Kemudian keluaran dari proses berupa 1 bit piksel disimpan dalam memori.



Gambar 3.8 Diagram blok implementasi proses deteksi tepi vertikal pada FPGA

Karena hasil deteksi tepi vertikal hanya bernilai antara 0 dan 1, maka untuk mendapatkan proyeksi vertikal maupun horizontal juga menjadi sederhana. Untuk analisis pertama yaitu proyeksi vertikal untuk menentukan batas vertikal area plat nomor (*band clipping*). Tahap yang dilakukan pertama adalah menentukan titik y yang memiliki nilai proyeksi vertikal maksimum (y_M):

$$y_M = \arg\max_{0 \le y \le h-1} \{p_y(y)\} \tag{3-2}$$

Tahap berikutnya yaitu menentukan batas atas (y_0) dan batas bawah (y_1):

$$y_0 = \max_{0 \le y \le y_M} \{y | p_y(y) \le C_y \cdot p_y(y_M)\} \tag{3-3}$$

$$y_1 = \min_{y_M \le y \le h-1} \{y | p_y(y) \le C_y \cdot p_y(y_M)\} \tag{3-4}$$

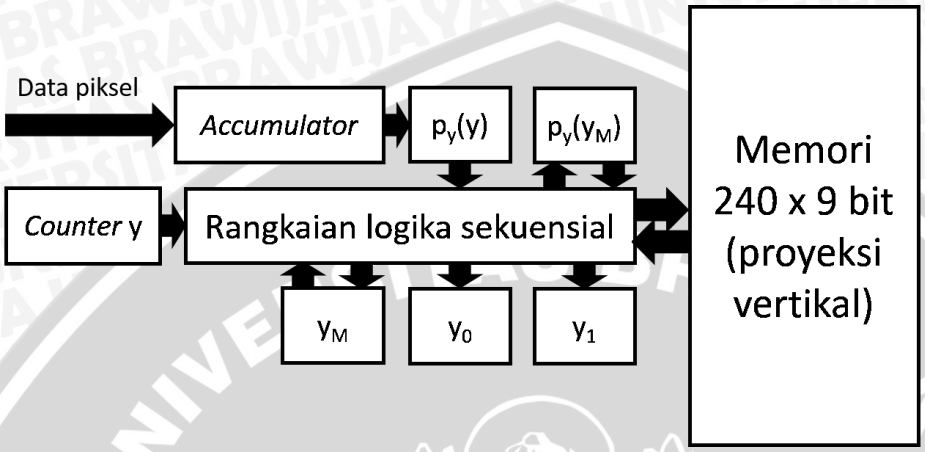
dengan C_y adalah konstanta penentu kaki puncak nilai maksimum proyeksi. Setelah itu dilakukan pengecekan tinggi ($h = y_1 - y_0 + 1$) supaya mempermudah seleksi plat nomor. Jika h tidak memenuhi kriteria minimum, maka proses kembali ke tahap awal untuk citra selanjutnya. Kriteria minimum tersebut adalah:

$$h \ge C_{hm} \tag{3-5}$$

dengan C_{hm} adalah batas tinggi minimum area plat nomor.

Gambar 3.9 menunjukkan diagram blok analisis proyeksi vertikal pada FPGA. Setiap data piksel yang terbaca menjadi input dari *accumulator*. Ketika sudah terbaca 1 baris piksel,

nilai *accumulator* akan disimpan sebagai nilai $p_y(y)$ dan *accumulator* mulai menghitung nilai proyeksi pada baris piksel baru. Untuk menghitung nilai y_M, setiap data proyeksi dibaca dan dicari yang terbesar. Untuk nilai y_0, dicari nilai y maksimum yang tidak melampaui y_M dan sesuai syarat $p_y(y) \leq C_y$. Untuk nilai y_1, dicari nilai y minimum yang terdekat dari y_M dan sesuai syarat pula.



Gambar 3.9 Diagram blok analisis proyeksi vertikal (*band clipping*) pada FPGA

Setelah ditemukan nilai-nilai batas vertikal, dilanjutkan pada analisis berikutnya yaitu proyeksi horizontal (*plate clipping*) melalui tahap berikut:

$$A(x_a, x_b) = \sum_{i=x_a}^{x_b} p_x(i) \tag{3-6}$$

$$x_{AM} = \underset{0 \leq x \leq w-1}{\arg\max} \{A(x, x + \Delta(x))\} \tag{3-7}$$

$$x_0 = \min_{x_{AM} \leq x \leq x_{AM} + \Delta(x_{AM})} \{x | p_x(x) \leq C_x \cdot p_x(x_M)\} \tag{3-8}$$

$$x_1 = \max_{x_{AM} \leq x \leq x_{AM} + \Delta(x_{AM})} \{x | p_x(x) \leq C_x \cdot p_x(x_M)\} \tag{3-9}$$

dengan:

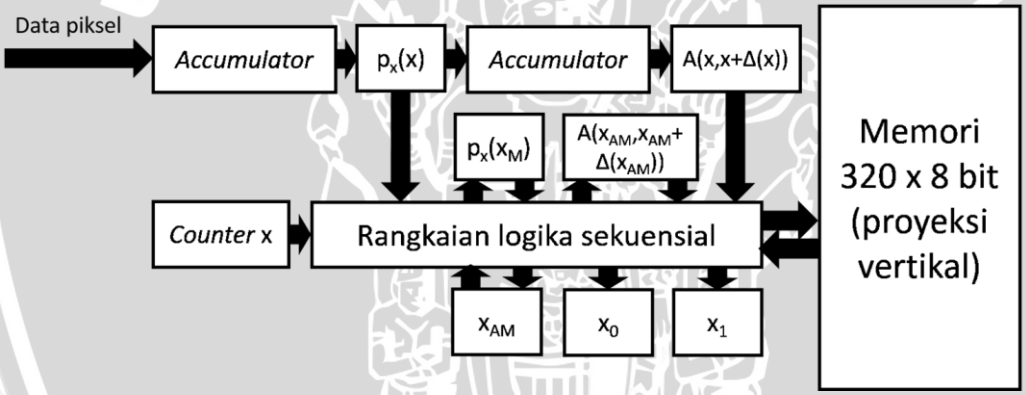
$$\Delta(x) = \begin{cases} 2C_{rM} \cdot h - 1 \; jika \; x + 2C_{rM} \cdot h < w \\ w - 1 - \text{x } lainnya \end{cases} \tag{3-10}$$

x_{AM} adalah nilai x dengan $A(x)$ maksimum, x_M adalah nilai x dengan $p_x(x)$ maksimum, x_0 adalah batas kiri dan x_1 adalah batas kanan. Sedangkan C_{rM} adalah konstanta rasio lebar : tinggi plat nomor maksimum kendaraan bermotor, yang didapat dari ukuran tulisan plat nomor pada sepeda motor (25 cm × 4 cm), sehingga $C_{rM} = 6.25$. $A(x_a, x_b)$ adalah fungsi luas area kurva dari titik x_a hingga x_b. Pada analisis proyeksi horizontal ini citra yang dianalisis adalah citra yang dipotong pada $y = y_0$ dan $y = y_1$ dari citra awal. Setelah itu dilakukan pengecekan lebar ($w = x_1 - x_0 + 1$) supaya mempermudah seleksi plat nomor. Jika w tidak memenuhi kriteria minimum, maka proses kembali ke tahap awal untuk citra selanjutnya. Kriteria minimum tersebut adalah:

$$w \geq C_{rm} \cdot h \tag{3-12}$$

dengan C_{rm} adalah konstanta rasio lebar : tinggi plat nomor minimum kendaraan bermotor, yang didapat dari ukuran tulisan plat nomor pada mobil (40 cm × 8 cm), sehingga $C_{rm} = 5$.

Gambar 3.10 menunjukkan diagram blok analisis proyeksi horizontal (*plate clipping*) pada FPGA. Setiap data piksel yang terbaca menjadi input dari *accumulator* pertama. Ketika sudah terbaca 1 kolom piksel, nilai *accumulator* pertama akan disimpan sebagai nilai $p_x(x)$ dan *accumulator* mulai menghitung nilai proyeksi pada kolom piksel baru. Sementara itu, nilai $p_x(x)$ yang telah didapat menjadi input dari *accumulator* kedua. Nilai *accumulator* kedua akan disimpan sebagai nilai $A(x, x + \Delta(x))$. Untuk menghitung nilai x_{AM}, setiap data $A(x, x + \Delta(x))$ dibaca dan dicari yang terbesar. Untuk nilai x_0, dicari nilai x minimum yang terdekat dari x_{AM} dan sesuai syarat $p_x(x) \geq C_x$. Untuk nilai x_1, dicari nilai x maksimum yang tidak melampaui $x_{AM} + \Delta(x_{AM})$ dan sesuai syarat pula. Setelah ditemukan batas-batas tersebut, maka didapatkan area plat nomor.

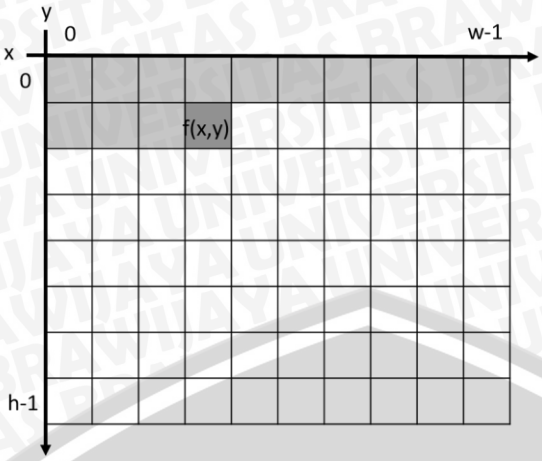


Gambar 3.10 Diagram blok analisis proyeksi horizontal (*plate clipping*) pada FPGA

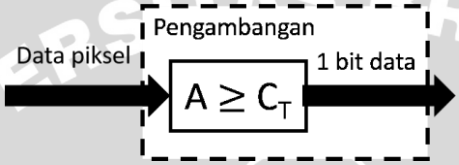
4.2) Segmentasi Karakter

Proses pengambangan (*thresholding*) pada citra area plat nomor diilustrasikan pada Gambar 3.11. Tiap pixel dibaca dari posisi kiri atas ($x = 0$, $y = 0$) hingga posisi piksel kanan bawah ($x = w - 1$, $y = h - 1$). Untuk mengetahui nilai hasil pengambangan suatu piksel, maka digunakan persamaan pengambangan yang telah dijelaskan sebelumnya.

Gambar 3.12 menunjukkan diagram blok implementasi proses pengambangan pada FPGA. Proses berlangsung melalui rangkaian logika pembanding (*comparator*), kemudian keluaran dari proses berupa 1 bit piksel disimpan dalam memori. Implementasi hanya memerlukan sebuah rangkaian dasar, karena persamaan pengambangan hanya melakukan perbandingan nilai input dengan sebuah nilai ambang C_T.



Gambar 3.11 Ilustrasi proses pengambangan (*thresholding*)



Gambar 3.12 Diagram blok implementasi proses pengambangan pada FPGA

Setelah dilakukan proses pengambangan, maka dilanjutkan dengan analisis proyeksi horizontal. Pada analisis ini dicari sekelompok titik x yang menjadi kaki puncak pada setiap area karakter, yaitu dengan persamaan berikut:

$$X_{c0} = \{x | p_x(x) \leq C_{xc} \cdot h \; dan \; p_x(x-1) > C_{xc} \cdot h, 0 \leq x \leq w-1\} \quad (3\text{-}13)$$

$$X_{c1} = \{x | p_x(x) \geq C_{xc} \cdot h \; dan \; p_x(x-1) < C_{xc} \cdot h, 0 \leq x \leq w-1\} \quad (3\text{-}14)$$

dengan C_{xc} adalah konstanta penentu kaki puncak nilai maksimum proyeksi. Untuk menyeleksi yang mana karakter dan bukan karakter, maka ditentukan kriteria minimum berikut:

$$A(x_{c0}, x_{c1}) \geq C_{rcm} \cdot h^2 \quad (3\text{-}15)$$

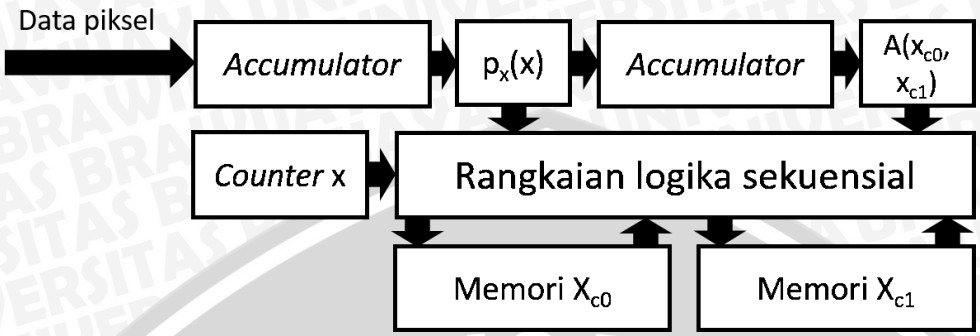
dan

$$A(x_{c0}, x_{c1}) \leq C_{rcM} \cdot h^2 \quad (3\text{-}16)$$

dengan C_{rcm} dan C_{rcM} adalah konstanta rasio lebar : tinggi karakter minimum dan maksimum kendaraan bermotor. Rasio minimum didapat dari ukuran ketebalan karakter plat nomor pada mobil (6 mm × 75 mm), sehingga dipilih yang mendekati: $C_{rcm} = \frac{3}{32} = 0.09375$, sedangkan rasio maksimum disesuaikan dengan ukuran karakter plat nomor pada sepeda motor (2 cm × 4 cm), sehingga $C_{rcM} = 0{,}5$.

Gambar 3.13 menunjukkan diagram blok analisis proyeksi horizontal (segmentasi) pada FPGA. Fungsi dari kedua *accumulator* serupa dengan analisis proyeksi horizontal sebelumnya, yaitu untuk menghitung nilai $p_x(x)$ dan $A(x_{c0}, x_{c1})$. Setiap nilai x_{c0} dicari saat

syarat $p_x(x) \geq C_{xc}$ terpenuhi hingga tak terpenuhi. Setelah itu nilai x_{c1} dicari saat syarat tak terpenuhi hingga terpenuhi. Setelah itu dimulai lagi mencari nilai x_{c0} untuk karakter berikutnya, dan seterusnya.



Gambar 3.13 Diagram blok analisis proyeksi horizontal (segmentasi) pada FPGA

Setelah ditemukan nilai-nilai batas-batas horizontal dilanjutkan pada analisis berikutnya yaitu proyeksi vertikal. Analisis proyeksi vertikal dilakukan sama seperti analisis proyeksi vertikal sebelumnya, hanya saja berbeda pada kriteria minimumnya. Kriteria tersebut dijabarkan sebagai berikut:

$$h_c \geq C_{rhm} \cdot h \tag{3-17}$$

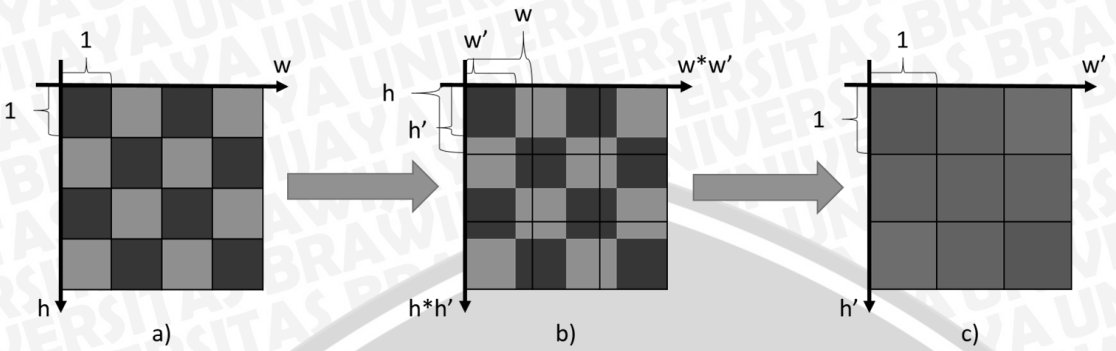
dengan C_{rhm} adalah konstanta untuk menentukan ketinggian minimum karakter, h_c adalah tinggi karakter dan h adalah tinggi plat nomor. Setelah ditemukan batas-batas tersebut, maka didapatkan segmentasi karakter plat nomor.

4.3) Pengenalan Karakter Optis

Citra karakter diubah dimensinya terlebih dahulu (*resize*) sebelum penerapan metode KNN. Proses ini diperlukan karena metode KNN hanya dapat diterapkan pada citra yang berdimensi sama. Pengubahan dimensi menggunakan metode *Weighted Average Downsampling* seperti disebutkan pada bab sebelumnya, dengan ilustrasi seperti yang terlihat pada Gambar 3.14. Pada metode ini, setiap piksel pada citra baru merupakan hasil rata-rata terbobot dari sejumlah piksel yang bersesuaian pada citra lama.

Pada Gambar 3.14 diilustrasikan bagaimana cara menentukan rata-rata terbobot yang dimaksud. Semisal $f(x, y)$ adalah nilai piksel pada koordinat (x, y). Pada ilustrasi bagian a), setiap piksel digambarkan dengan sebuah area dengan luas 1×1, sedangkan keseluruhan citra adalah area seluas $w \times h$. Warna pada setiap area piksel menunjukkan nilai piksel. Pada ilustrasi bagian b) dimensi telah diubah dengan mengalikan konstanta w' pada sumbu horizontal dan h' pada sumbu vertikal. Selain itu, satuan area piksel diubah menjadi seluas $w \times h$. Dengan satuan tersebut, maka terbentuklah citra dengan dimensi sesuai dengan yang

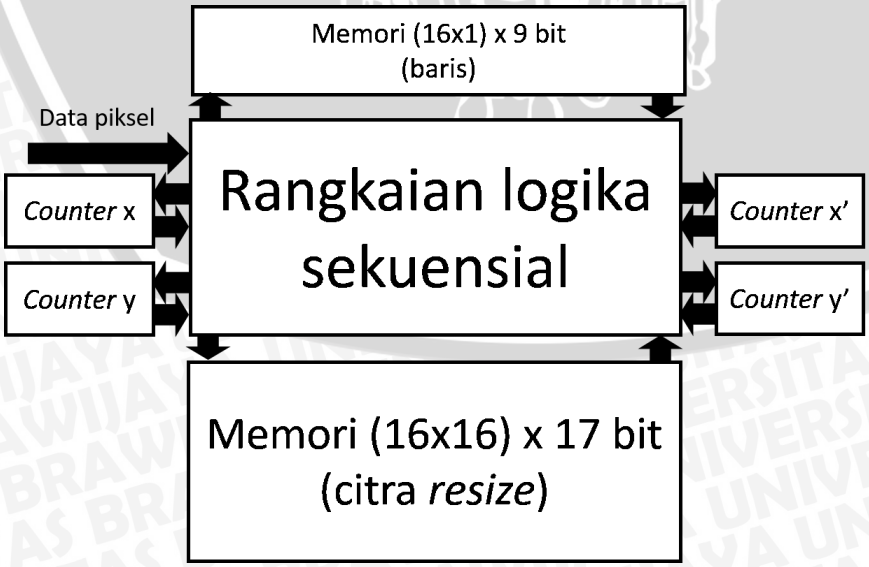
baru, sehingga tinggal mengetahui nilai piksel untuk setiap satuan area yang baru dengan menghitung hasil rata-rata nilai secara proporsional pada area tersebut.



Gambar 3.14 Ilustrasi proses *resize* dengan metode *Weighted Average Downsampling*

Jika memperhatikan area pojok kiri atas (piksel pada posisi $x' = 0, y' = 0$), maka nilai rata-rata terbobot pada area tersebut adalah $\{(f(0,0) \times w' \times h') + (f(0,1) \times w' \times (h - h')) + (f(1,0) \times (w - w') \times h') + (f(1,1) \times (w - w') \times (h - h'))\}/(w \times h)$. Hasil rata-rata ditunjukkan pada ilustrasi bagian c). Persamaan untuk menghitung nilai rata-rata terbobot digeneralisir menjadi Persamaan (2-10).

Gambar 3.15 menunjukkan diagram blok implementasi proses *resize* pada FPGA. Data piksel untuk satu baris diproses terlebih dahulu dan disimpan hasilnya pada memori baris (16×1) × 9 bit, setelah itu diakumulasikan pada baris-baris yang sesuai di memori citra *resize* (16×16) × 17 bit. Setiap lokasi memori (untuk tiap piksel baru) sebesar 17 bit untuk menyimpan data penjumlahan piksel, dengan kemungkinan nilai terbesarnya $w \times h$. Karena nilai piksel baru adalah hasil rata-rata, sementara piksel baru nilainya adalah 1 bit, maka cukup diambil batas pada $(w \times h)/2$ untuk menentukan piksel baru bernilai 0 atau 1.



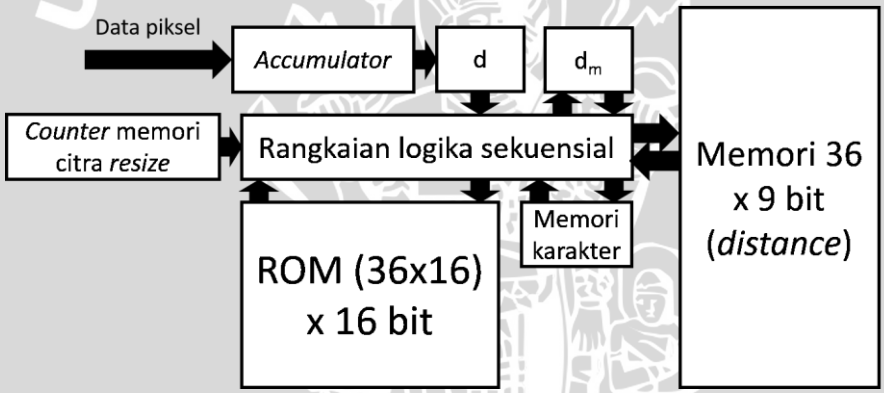
Gambar 3.15 Diagram blok implementasi proses *resize* pada FPGA

Setelah itu dilakukan metode *K-Nearest Neighbor* dengan nilai $K = 1$. Metode yang diterapkan telah disederhanakan, dengan persamaan untuk menghitung *distance* adalah sebagai berikut:

$$d = \sum_{y=0}^{h-1} \sum_{x=0}^{w-1} f_i(x,y)\ XOR\ f_s(x,y) \quad (3\text{-}18)$$

dengan $f_i(x,y)$ adalah piksel pada citra input dan $f_s(x,y)$ adalah piksel pada citra referensi. Data *set* citra referensi berjumlah 36, masing-masing mewakili setiap karakter huruf dan angka. Nilai d terkecil yang didapat menunjukkan karakter yang dikenali.

Gambar 3.16 menunjukkan diagram blok implementasi metode KNN pada FPGA. Cara kerja pada diagram blok kali ini serupa dengan diagram blok pada analisis proyeksi. Setiap data piksel yang terbaca dengan data pada ROM (data referensi) diproses (dilakukan operasi XOR antara data input dan referensi) dan hasilnya menjadi input dari *accumulator*. Ketika sudah terbaca seluruh citra karakter, nilai *accumulator* akan disimpan sebagai nilai *distance* (d) dan *accumulator* mulai membaca citra input lagi dari awal untuk karakter berikutnya pada ROM, untuk menghitung nilai *distance* terhadap karakter tersebut.



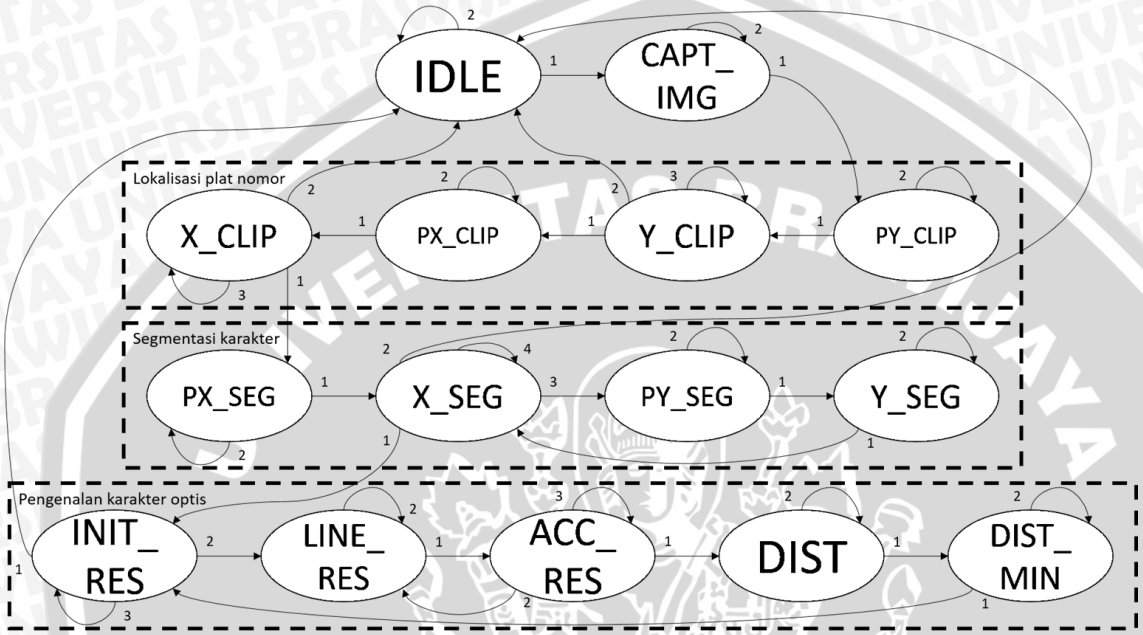
Gambar 3.16 Diagram blok implementasi metode KNN pada FPGA

Untuk menghitung nilai *distance* minimum (d_m), setiap data proyeksi dibaca dan dicari yang terkecil. Setelah diketahui d_m, maka karakter yang bersangkutan disimpan pada memori. Setelah itu kembali ke proses *resize* untuk citra input berikutnya. Proses *resize* dan KNN berulang seterusnya hingga seluruh citra input selesai diproses. Setelah seluruh citra input selesai diproses, maka didapat seluruh karakter plat nomor. Proses ini adalah tahap akhir dari sistem, sehingga setelah melalui tahap ini sistem kembali pada tahap awal untuk mempersiapkan deteksi plat nomor kendaraan berikutnya.

4.4) *State Diagram* Sistem ANPR

Untuk memperjelas cara kerja implementasi sistem ANPR secara keseluruhan, maka dibuatlah *state diagram* sistem ANPR seperti ditunjukkan pada Gambar 3.17. Sistem ANPR

bekerja secara terus menerus selama sinyal pemicu aktif. Keadaan (*state*) awal sistem disebut IDLE (tidak bekerja). Jika sinyal *reset* aktif, maka sistem akan berpindah ke keadaan ini. Pada keadaan ini, sistem tidak bekerja hingga sinyal pemicu aktif (tanda panah 2). Ketika sinyal pemicu aktif, maka sistem akan mulai bekerja (tanda panah 1) dan tidak akan berhenti hingga selesai (kembali ke keadaan IDLE) atau sinyal reset diaktifkan. Tanda panah 1 menunjukkan perpindahan keadaan sistem dari IDLE ke CAPT_IMG.



Gambar 3.17 *State diagram* sistem ANPR

Pada keadaan CAPT_IMG blok antarmuka kamera-memori bekerja dengan menyalurkan data ke memori citra sehingga didapat citra terbaru dari kamera, baik citra *grayscale*, deteksi tepi vertikal maupun pengambangan. Sistem memberi sinyal perintah pada blok tersebut setelah berpindah ke keadaan ini, sehingga blok tersebut telah siap dan menunggu tepi turun sinyal *vsync*. Setelah selesai, maka sistem akan menjalankan tahap-tahap selanjutnya secara berurutan (tanda panah 1) sesuai dengan 3 tahap ANPR yang telah dijelaskan sebelumnya yaitu lokalisasi plat nomor, segmentasi karakter, dan pengenalan karakter optis.

Tahap lokalisasi plat nomor dibagi menjadi 2 sub proses yaitu *band clipping* dan *plate clipping* sesuai dengan penjelasan sebelumnya. Sub proses ini dibagi lagi masing-masing dengan 2 keadaan. Pada proses *band clipping* keadaan dibagi menjadi PY_CLIP (proyeksi vertikal) dan Y_CLIP (menentukan titik-titik batas vertikal), sedangkan pada proses *plate clipping* keadaan dibagi menjadi PX_CLIP (proyeksi horizontal) dan X_CLIP (menentukan titik-titik batas horizontal). Setiap keadaan memiliki tanda panah menuju keadaan itu sendiri yang menandakan bahwa proses belum selesai dijalankan pada tahap tersebut, serta tanda

panah yang menuju ke keadaan selanjutnya. Pada saat keadaan X_CLIP dan Y_CLIP, tanda panah 2 menuju ke keadaan IDLE yang menunjukkan situasi ketika titik-titik batas tidak memenuhi kriteria minimum, sehingga proses berhenti dan kembali lagi untuk deteksi selanjutnya.

Tahap segmentasi karakter dibagi menjadi 4 keadaan, yaitu PX_SEG (proyeksi horizontal), X_SEG (menentukan titik-titik batas horizontal), PY_SEG (proyeksi vertikal) dan Y_SEG (menentukan titik-titik batas vertikal). Tiga keadaan terakhir membentuk *loop* yang menandakan perulangan proses penentuan titik-titik batas untuk setiap karakter. Setelah seluruh koordinat selesai diproses, maka terdapat 2 kondisi untuk menuju ke keadaan selanjutnya (tanda panah 1 dan 2 pada keadaan X_SEG). Tanda panah 1 berlaku jika terdapat sejumlah karakter untuk diproses lebih lanjut (menuju tahap pengenalan karakter optis). Sedangkan jika jumlah karakter terdeteksi adalah nol, maka tanda panah 2 yang berlaku (kembali ke awal).

Tahap pengenalan karakter optis dibagi menjadi 2 sub proses yaitu *resize* dan KNN. Pada proses *resize* keadaan dibagi menjadi INIT_RES (inisialisasi memori dan register), LINE_RES (*resize* baris) dan ACC_RES (akumulasi untuk satu baris). Pada proses KNN keadaan dibagi menjadi DIST (menghitung seluruh nilai *distance*) dan DIST_MIN (menentukan karakter dengan *distance* terkecil). Keadaan LINE_RES dan ACC_RES membentuk *loop* yang menandakan perulangan proses *resize* untuk setiap baris. Keseluruhan tahap pengenalan karakter optis membentuk *loop* yang menandakan perulangan proses pengenalan setiap karakter yang terdeteksi. Jika seluruh karakter telah diproses, maka sistem akan kembali ke keadaan awal (tanda panah 1 pada INIT_RES).

3.3 Metode Pengujian Sistem

Terdapat 3 macam metode yang akan digunakan pada pengujian sistem, yaitu simulasi *timing diagram*, simulasi algoritma sistem secara *software*, dan pengujian langsung pada FPGA. Ketiga macam metode ini penting untuk mengukur tingkat keakuratan dan keandalan sistem secara keseluruhan. Setiap prosedur pengujian akan menggunakan salah satu dari ketiga metode ini.

3.3.1 Simulasi *Timing Diagram*

Simulasi *timing diagram* dilakukan untuk mengetahui bagaimana kerja blok perangkat keras sistem yang telah didesain sebelum diimplementasikan untuk pengujian langsung. Simulasi ini dilakukan dengan memberikan input untuk dites pada blok yang diuji. Jika dari

penerimaan data pada port input blok tersebut dapat mengeluarkan data pada port output dengan benar, maka desain blok tersebut dinyatakan siap untuk diimplementasikan. Blok-blok yang perlu diuji dengan simulasi *timing diagram* adalah blok-blok antarmuka, karena blok-blok tersebut berkomunikasi dengan rangkaian luar. Kesalahan pada komunikasi dapat menyebabkan kesalahan pengiriman data yang sulit dideteksi.

Simulasi dilakukan dengan mengeksekusi kode VHDL *test bench* menggunakan ISim Simulator pada *software* Xilinx ISE Project Navigator. Pada kode inilah dijelaskan bagaimana pemicuan sinyal pada port input blok yang dites. Setelah kode dieksekusi oleh ISim Simulator, maka akan ditampilkan *timing diagram* sinyal seluruh port pada blok tersebut. Dari tampilan tersebut dapat dianalisis apakah sinyal output yang keluar sudah benar sesuai dengan target aplikasi. Blok-blok yang akan diuji pada pengujian sistem ini adalah blok antarmuka SCCB dan blok VGA *driver*.

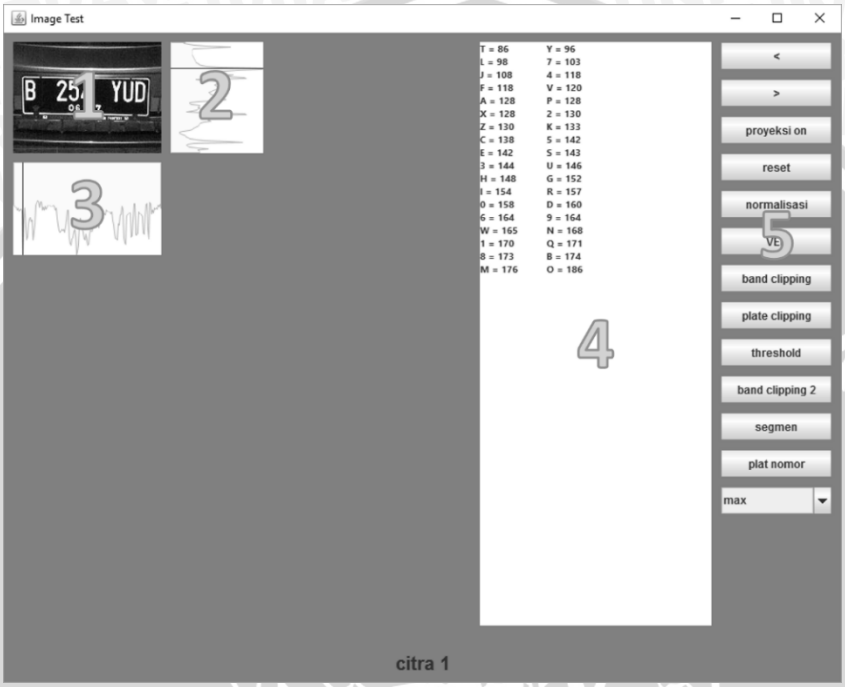
3.3.2 Simulasi Algoritma Sistem pada Program

Simulasi algoritma sistem pada program dilakukan untuk mengetahui bagaimana kerja sistem berdasarkan perancangan algoritmanya. Simulasi ini dilakukan pada program yang telah dibuat sebelumnya menggunakan bahasa Java. Pembuatan program ini sengaja dibuat khusus untuk melakukan simulasi terhadap algoritma yang digunakan untuk mendeteksi plat nomor. Perancangan sistem yang telah dijelaskan sebelumnya diterapkan dalam kode program, kemudian dibuatlah interaksi dengan pengguna sehingga dapat memberikan input citra, mengeksekusi algoritma sistem dan menghasilkan output yang diinginkan. Dari hasil pengujian pada simulasi nantinya dapat dilakukan analisis dan perbandingan terhadap pengujian langsung pada FPGA. Gambar 3.18 menunjukkan tampilan *Global User Interface* (GUI) dari program simulasi.

Terdapat beberapa bagian dari tampilan yang ditandai dengan nomor pada Gambar 3.18. Nomor 1 menunjukkan tampilan citra yang akan diproses. Pada bagian ini citra input maupun citra hasil pemrosesan dapat dilihat. Nomor 2 dan 3 menunjukkan tampilan histogram proyeksi horizontal dan vertikal. Pada bagian ini dapat dilihat hasil proyeksi sehingga dapat dilakukan analisis secara visual. Nomor 4 berfungsi menampilkan teks keterangan. Pada bagian ini ditampilkan teks berupa nilai *distance* citra karakter, plat nomor terdeteksi, serta waktu rata-rata eksekusi. Nomor 5 menunjukkan sekumpulan tombol yang melakukan fungsi-fungsi tertentu. Pada setiap tombol inilah dilakukan simulasi bagian-bagian proses sistem pada citra. Jika sebuah tombol ditekan program akan mengeksekusi kode yang berisi algoritma dari proses dan metode yang digunakan untuk sistem ANPR yang

akan dirancang, kemudian hasilnya ditampilkan secara visual pada nomor 1, 2 dan 3 maupun secara tekstual pada nomor 4.

Setiap prosedur pengujian yang menggunakan metode ini dimulai dengan memilih citra untuk sampel uji. Kemudian dilanjutkan dengan menekan salah satu tombol pada nomor 5 yang sesuai dengan pengujian tersebut, sehingga diperoleh data yang diinginkan pada nomor 1, 2, 3 atau 4 untuk diamati dan dilakukan analisis lebih lanjut.



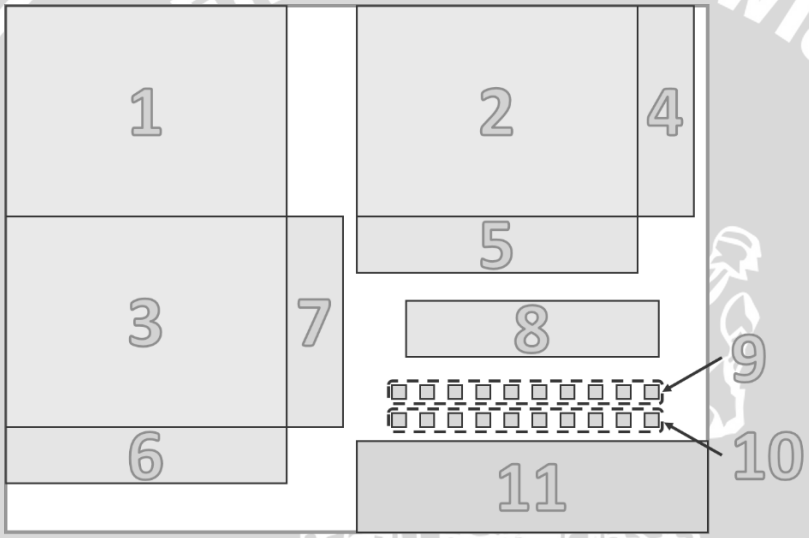
Gambar 3.18 Tampilan GUI program simulasi menggunakan Java

3.3.3 Pengujian Langsung pada FPGA

Pengujian langsung pada FPGA dilakukan untuk mengetahui bagaimana kerja perancangan sistem sesungguhnya setelah diimplementasikan pada FPGA. Sebelumnya implementasi dilakukan menggunakan *software* Xilinx ISE Project Navigator. Setelah proses implementasi selesai akan diperoleh *report* berupa skematik rangkaian hasil implementasi dan penggunaan *resource* FPGA, dan *board* FPGA siap untuk pengujian. Perangkat keras berupa modul kamera dan port VGA monitor disambungkan pada *board* FPGA. Program implementasi FPGA telah dirancang agar dapat mendukung dalam proses pengambilan data dan analisis. Hal ini meliputi penggunaan monitor dan *switch* pada board FPGA.

Tampilan monitor untuk pengujian dibagi dalam beberapa bagian seperti yang terlihat pada Gambar 3.19. Nomor 1 menunjukkan tampilan citra *grayscale* yang didapat dari kamera. Nomor 2 menunjukkan tampilan citra hasil deteksi tepi vertikal. Nomor 3 menunjukkan tampilan citra hasil pengambangan. Nomor 4 dan 5 menunjukkan proyeksi

vertikal dan horizontal dari citra deteksi tepi vertikal, sehingga nomor 2, 4 dan 5 menunjukkan hasil pengujian tahap lokalisasi plat nomor. Nomor 6 dan 7 menunjukkan proyeksi horizontal dan vertikal dari citra pengambangan, sehingga nomor 3, 6 dan 7 menunjukkan hasil pengujian tahap segmentasi karakter. Nomor 8 menunjukkan nilai *distance* salah satu segmen citra karakter dari hasil tahap segmentasi karakter. Nomor 9 menunjukkan tampilan hasil proses *resize* seluruh segmen citra karakter. Nomor 10 menunjukkan tampilan teks plat nomor hasil pemrosesan metode KNN, sehingga nomor 8, 9 dan 10 menunjukkan hasil pengujian tahap pengenalan karakter optis. Nomor 11 menunjukkan tampilan teks keterangan untuk mendukung dalam melakukan analisis, di antaranya adalah menampilkan angka periode *pixel clock* (PCLK), waktu eksekusi, nilai grafik di sebuah titik, dan sebagainya.



Gambar 3.19 Skema tampilan monitor untuk pengujian sistem

3.4 Pengujian Sistem

Untuk menganalisis kinerja sistem apakah sesuai dengan yang direncanakan maka dilakukan sejumlah simulasi dan pengujian. Simulasi dan pengujian dilakukan pada masing-masing blok dan kemudian secara keseluruhan sistem. Hal ini diperlukan untuk mengetahui kinerja dari masing-masing blok serta memudahkan kita untuk mendeteksi kesalahan dari implementasi sistem. Secara garis besar pengujian sistem dilakukan sebagai berikut:

3.4.1 Simulasi *Timing Diagram* Blok Antarmuka SCCB

Simulasi *timing diagram* blok antarmuka SCCB dilakukan untuk mengetahui apakah listing program blok tersebut dapat menghasilkan sinyal output dengan benar berdasarkan spesifikasi antarmuka SCCB. Pada kode *test bench*, sinyal "reset" diberikan logika 1 selama periode waktu tertentu. Kemudian pemberian perintah untuk memulai komunikasi oleh blok

kontroler kamera dimulai dengan pengiriman sinyal diaktifkan (“send” = 1), operasi tulis (“rw” = 0), dengan alamat (“addr”) 0xAB dan data input (“din”) 0xCD. Sinyal keluaran “sioc” dan “siod” dibandingkan dengan *datasheet* spesifikasi SCCB melalui tampilan ISim Simulator.

3.4.2 Simulasi *Timing Diagram* Blok VGA *Driver*

Simulasi *timing diagram* blok VGA *driver* dilakukan untuk mengetahui apakah listing program blok tersebut dapat menghasilkan sinyal output dengan benar berdasarkan spesifikasi *timing* VGA 800×600 @ 72 Hz. Pada kode *test bench*, sinyal “reset” diberikan logika 1 selama periode waktu tertentu. Sementara itu, seluruh sinyal input diberikan nilai *default* 0. Sinyal keluaran “hs” dan “vs” dibandingkan dengan tabel spesifikasi *timing* VGA 800×600 @ 72 Hz melalui tampilan ISim Simulator.

3.4.3 Implementasi Sistem

Seperti dijelaskan sebelumnya bahwa setelah dilakukan implementasi sistem diperoleh *report* berupa skematik rangkaian hasil implementasi dan penggunaan *resource* FPGA. Pemeriksaan *report* dilakukan untuk mengetahui apakah blok-blok sistem telah diimplementasikan dengan benar dan untuk menganalisis tingkat efisiensi implementasi sistem dari penggunaan *resource* FPGA.

3.4.4 Pengujian Perangkat Keras

Pengujian perangkat keras dilakukan untuk mengetahui apakah kamera dan monitor sudah bekerja sesuai spesifikasi perancangan. Pengujian dilakukan dengan merangkai keseluruhan sistem (FPGA, kamera dan monitor) kemudian melakukan analisis dari hasil tampilan monitor. Pada saat pengujian dapat diamati apakah tampilan monitor telah sesuai dengan skema seperti yang ditunjukkan pada Gambar 3.19, sehingga siap untuk dilakukan pengujian-pengujian selanjutnya. Diamati pula apakah gambar yang ditangkap kamera dapat menampilkan tulisan plat nomor dengan jelas. Periode PCLK ditampilkan pada monitor di bagian teks keterangan (nomor 11 pada Gambar 3.19). Jika periode PCLK ditampilkan sebesar 80 ns pada monitor (target frekuensi PCLK 12,5 MHz) maka kamera sudah sesuai dengan *setting* yang telah ditetapkan.

3.4.5 Pengujian Sistem Tahap Lokalisasi Plat Nomor

Pengujian ini dilakukan untuk mengetahui bagaimana kerja sistem pada tahap lokalisasi plat nomor. Pengujian dilakukan menggunakan 4 citra sampel uji dengan plat nomor yang

berbeda. Pengujian dibagi menjadi simulasi program dan pengujian langsung. Pada simulasi program citra sampel uji diakses dari memori komputer, sedangkan pada pengujian langsung citra sampel uji yang sama dicetak pada kertas HVS (ukuran plat yang tercetak tidak sesuai dengan ukuran plat nomor aslinya) dan dilakukan percobaan sebanyak 4 kali untuk setiap variasi jarak (antara 30-60 cm dengan selisih tiap 10 cm). Pada simulasi program, setiap proses dari tahap lokalisasi plat nomor (deteksi plat nomor, *band clipping* dan *plate clipping*) dieksekusi dengan menekan tombol yang berbeda pada program. Pada pengujian langsung, setiap proses dapat diamati hasilnya pada monitor (nomor 2, 4 dan 5 pada Gambar 3.19). Analisis yang dapat dibahas berupa pengaruh variasi jarak terhadap kerja sistem, dan perbandingan kerja sistem pada pengujian langsung dengan pada simulasi program. Dari hasil pengujian dilakukan analisis tingkat keberhasilan sistem dalam memisahkan area plat nomor dari citra sampel uji.

3.4.6 Pengujian Sistem Tahap Segmentasi Karakter

Pengujian ini dilakukan untuk mengetahui bagaimana kerja sistem pada tahap segmentasi karakter. Pengujian ini merupakan lanjutan dari tahap lokalisasi plat nomor dengan prosedur yang sama. Pada pengujian langsung, hasilnya dapat diamati pada bagian nomor 3, 6 dan 7 di monitor (Gambar 3.19). Dari hasil pengujian dilakukan analisis tingkat kesalahan sistem dalam melakukan segmentasi karakter dari area plat nomor.

3.4.7 Pengujian Sistem Tahap Pengenalan Karakter Optis

Pengujian ini dilakukan untuk mengetahui bagaimana kerja sistem pada tahap pengenalan karakter optis. Pengujian ini merupakan lanjutan dari tahap segmentasi karakter dengan prosedur yang sama. Pada pengujian langsung, hasilnya dapat diamati pada bagian nomor 8, 9 dan 10 di monitor (Gambar 3.19). Dari hasil pengujian dilakukan analisis tingkat kesalahan sistem dalam menentukan karakter dari setiap segmen citra karakter. Pengujian ini juga merupakan pengujian tahap akhir dari sistem, sehingga dilakukan pula analisis tambahan untuk mengetahui performa sistem dari waktu eksekusinya. Waktu eksekusi dari simulasi program maupun pengujian langsung dapat dilihat pada teks keterangan masing-masing.

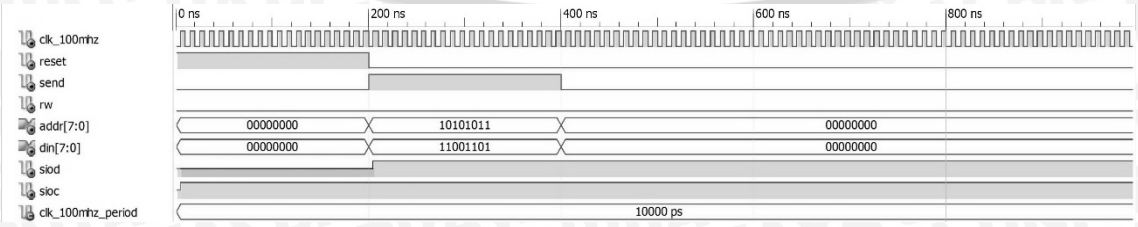
BAB IV
HASIL DAN PEMBAHASAN

Hasil dari penelitian ini diperoleh dari tahapan simulasi dan pengujian. Setelah itu, dilakukan analisis pembahasan dari hasil yang diperoleh. Hasil dari simulasi dan pengujian ini nantinya dijadikan sebagai acuan dalam pengambilan kesimpulan. Adapun simulasi dan pengujian tersebut meliputi simulasi *timing diagram* blok antarmuka SCCB, simulasi *timing diagram* blok VGA *driver*, implementasi sistem, pengujian perangkat keras, pengujian sistem tahap lokalisasi plat nomor, pengujian sistem tahap segmentasi karakter dan pengujian sistem tahap pengenalan karakter optis.

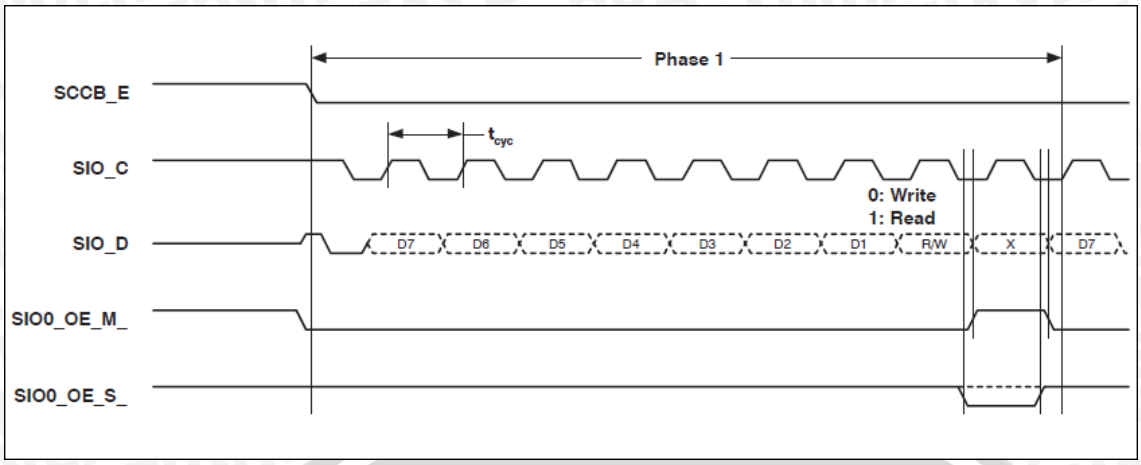
4.1 Hasil Simulasi *Timing Diagram* Blok Antarmuka SCCB

Analisis dari *timing diagram* blok antarmuka SCCB dibagi menjadi 3 fase untuk 1 siklus transmisi sesuai dengan *datasheet* spesifikasi antarmuka SCCB. Gambar 4.1 menunjukkan *timing diagram* awal mula pengaktifan sinyal "reset" dan perintah untuk memulai komunikasi oleh blok kontroler kamera dengan alamat 0xAB dan data 0xCD. Terlihat bahwa selama sinyal "reset" aktif (bernilai 1) sinyal "siod" bernilai Z (mengambang / impedansi tinggi), kemudian saat sinyal "reset" mulai tidak aktif (bernilai 0) sinyal "siod" nilai logikanya menjadi 1. Hal ini sesuai dengan spesifikasi pada *datasheet*.

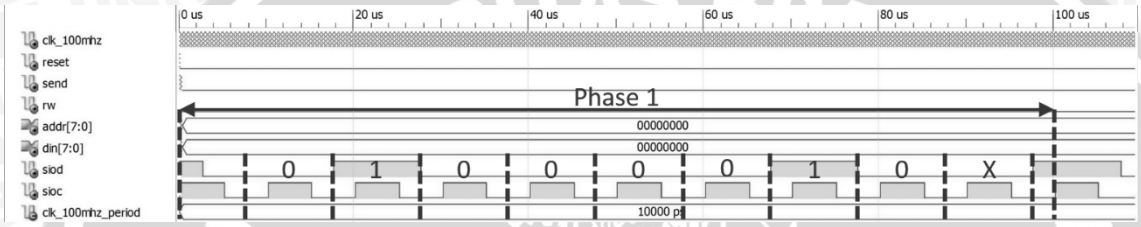
Timing diagram pada *datasheet* untuk fase 1 dapat dilihat pada Gambar 4.2, sedangkan hasil simulasi dapat dilihat pada Gambar 4.3. Dapat dilihat bahwa *timing diagram* hasil simulasi menunjukkan pengiriman data sebanyak 9 bit, dengan pergantian siklus pengiriman bit terjadi saat sinyal "sioc" bernilai 0, sehingga sudah cocok dengan *timing diagram* pada *datasheet*. Fase 1 adalah fase pengiriman alamat ID. Diketahui alamat ID operasi tulis sensor kamera OV7670 adalah 0x42. Dari Gambar 4.3 dapat dilihat bahwa pada fase 1 blok antarmuka SCCB telah menghasilkan sinyal output dengan benar (0b01000010=0x42).



Gambar 4.1 *Timing diagram* perintah untuk memulai komunikasi oleh blok kontroler kamera

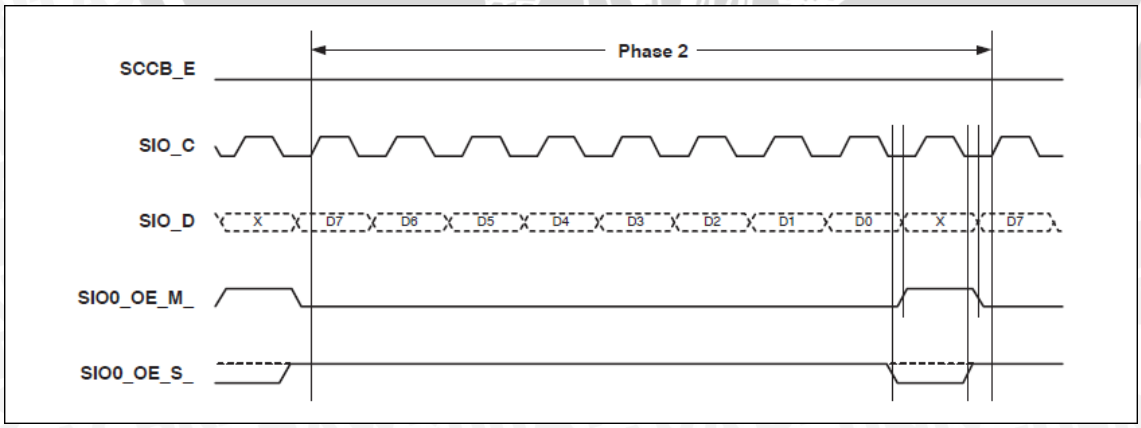


Gambar 4.2 *Timing diagram* pada *datasheet* untuk fase 1 (alamat ID)

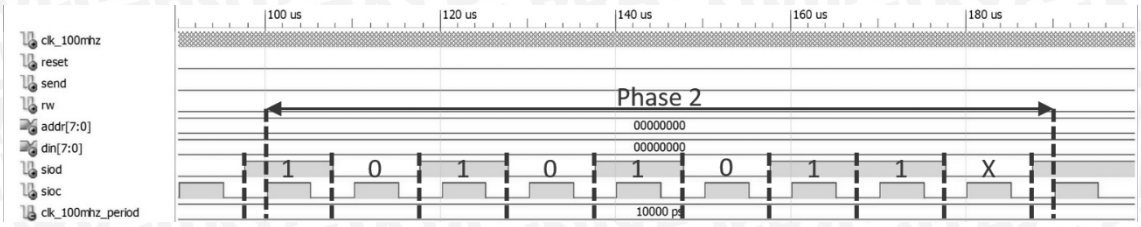


Gambar 4.3 *Timing diagram* hasil simulasi untuk fase 1 (alamat ID)

Timing diagram pada *datasheet* untuk fase 2 dapat dilihat pada Gambar 4.4, sedangkan hasil simulasi dapat dilihat pada Gambar 4.5. Dapat dilihat bahwa *timing diagram* hasil simulasi menunjukkan pengiriman data sebanyak 9 bit, dengan pergantian siklus pengiriman bit terjadi saat sinyal "sioc" bernilai 0, sehingga sudah cocok dengan *timing diagram* pada *datasheet*. Fase 2 adalah fase pengiriman alamat *register*. Sesuai perintah dari blok kontroler kamera, alamat *register* yang dituju adalah 0xAB. Dari Gambar 4.5 dapat dilihat bahwa pada fase 2 blok antarmuka SCCB telah menghasilkan sinyal output dengan benar (0b10101011=0xAB).

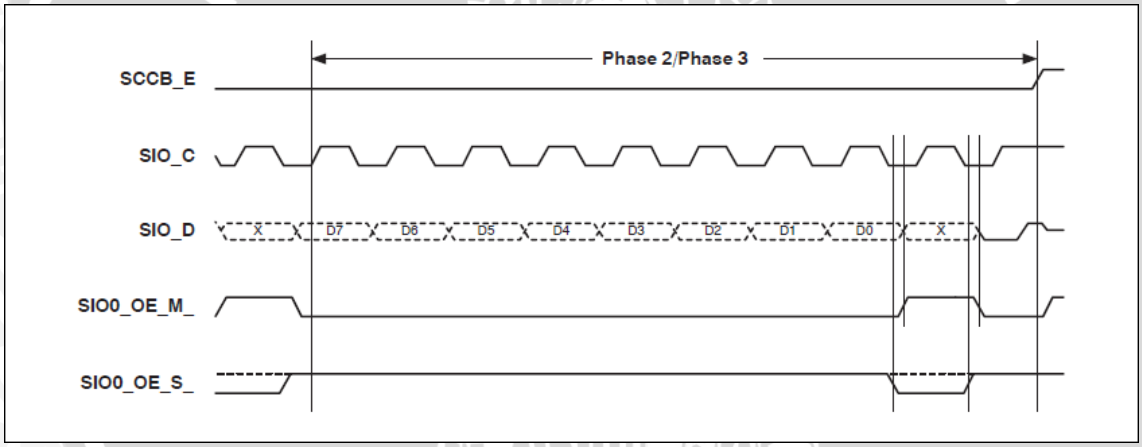


Gambar 4.4 *Timing diagram* pada *datasheet* untuk fase 2 (alamat *register*)

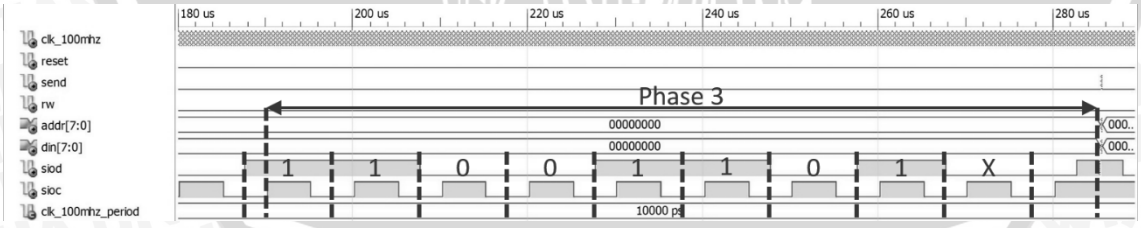


Gambar 4.5 *Timing diagram* hasil simulasi untuk fase 2 (alamat *register*)

Timing diagram pada *datasheet* untuk fase 3 dapat dilihat pada Gambar 4.6, sedangkan hasil simulasi dapat dilihat pada Gambar 4.7. Dapat dilihat bahwa *timing diagram* hasil simulasi menunjukkan pengiriman data sebanyak 9 bit, dengan pergantian siklus pengiriman bit terjadi saat sinyal "sioc" bernilai 0, sehingga sudah cocok dengan *timing diagram* pada *datasheet*. Fase 3 adalah fase pengiriman data tulis. Sesuai perintah dari blok kontroler kamera data tulis yang akan dikirim adalah 0xCD. Dari Gambar 4.7 dapat dilihat bahwa pada fase 3 blok antarmuka SCCB telah menghasilkan sinyal output dengan benar (0b11001101=0xCD).



Gambar 4.6 *Timing diagram* pada *datasheet* untuk fase 3 (data tulis)

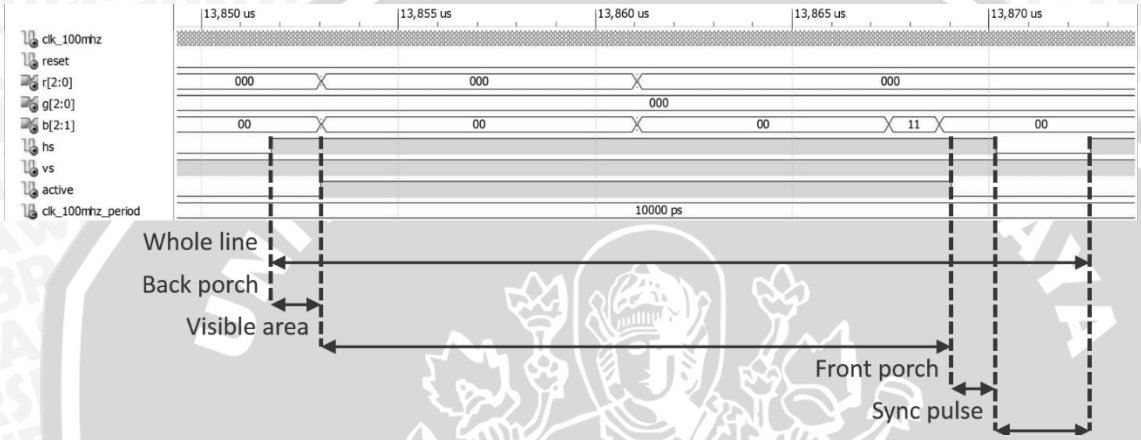


Gambar 4.7 *Timing diagram* hasil simulasi untuk fase 3 (data tulis)

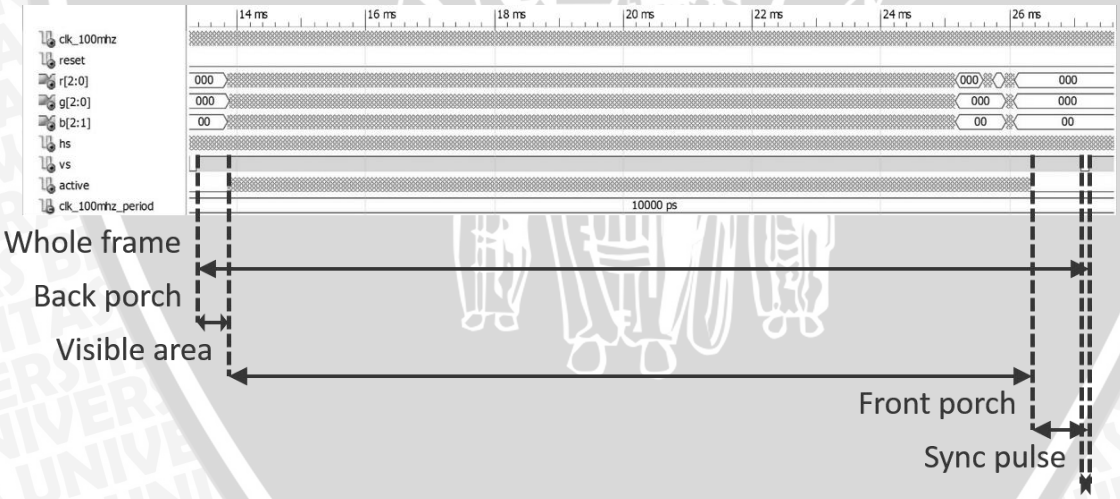
Dari hasil simulasi *timing diagram* dapat dilihat bahwa sinyal "siod" dan "sioc" telah sesuai dengan *datasheet*. Dapat diperhatikan pula pada hasil simulasi bahwa periode sinyal "sioc" sebesar 10 µs (frekuensi sebesar 100 KHz). Hal ini juga menunjukkan bahwa frekuensi sinyal "sioc" telah sesuai dengan *datasheet*. Dapat disimpulkan bahwa blok antarmuka SCCB mampu berkomunikasi sesuai spesifikasi antarmuka SCCB.

4.2 Hasil Simulasi *Timing Diagram* Blok VGA *Driver*

Gambar 4.8 dan Gambar 4.9 menunjukkan *timing diagram* sinyal "hs" dan "vs" pada blok VGA *driver*. Pada Tabel 4.1 dan Tabel 4.2 disajikan data secara lebih ringkas untuk membandingkan data simulasi dengan spesifikasi *timing* VGA 800×600 @ 72 Hz. Dapat dilihat dari tabel bahwa hasil simulasi menunjukkan data *timing* yang sama dengan spesifikasi *timing* VGA 800×600 @ 72 Hz. Dari hasil simulasi *timing diagram* dengan melihat bahwa data tabel sama dengan spesifikasi *timing* VGA 800×600 @ 72 Hz maka dapat disimpulkan bahwa blok VGA *driver* mampu bekerja sesuai yang diinginkan.



Gambar 4.8 *Timing diagram* sinyal "hs" pada blok VGA *driver*



Gambar 4.9 *Timing diagram* sinyal "vs" pada blok VGA *driver*

Tabel 4.1 Data *timing* horizontal VGA *driver*

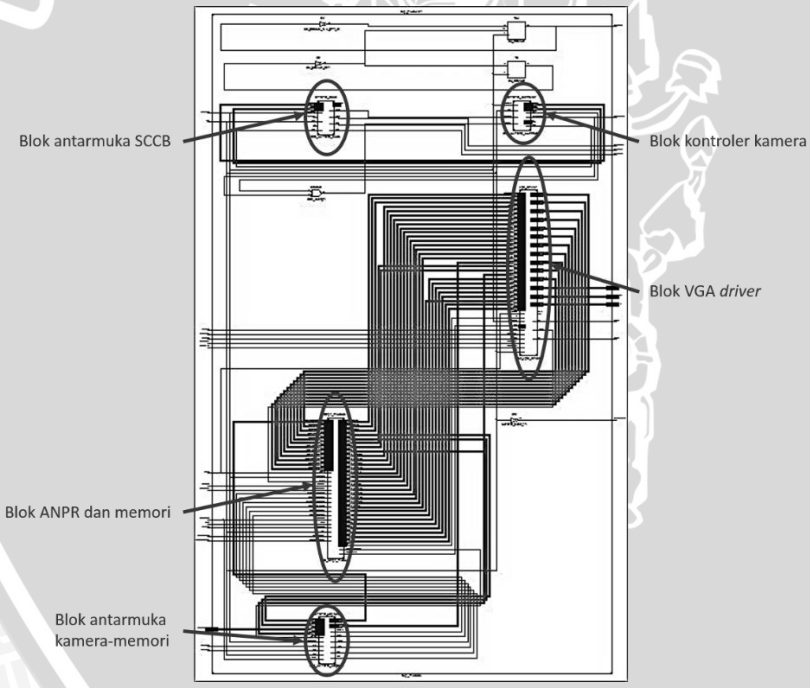
Scanline part	Hasil Simulasi (µs)	Spesifikasi (µs)
Visible area	16	16
Front porch	1,12	1,12
Sync pulse	2,4	2,4
Back porch	1,28	1,28
Whole line	20,8	20,8

Tabel 4.2 Data *timing* vertikal VGA *driver*

Frame part	**Hasil Simulasi (ms)**	**Spesifikasi (ms)**
Visible area	12,48	12,48
Front porch	0,7696	0,7696
Sync pulse	0,1248	0,1248
Back porch	0,4784	0,4784
Whole frame	13,8528	13,8528

4.3 Hasil Implementasi Sistem

Setelah dilakukan implementasi sistem, didapatkan hasil rangkaian skematik dari keseluruhan sistem seperti ditunjukkan pada Gambar 4.10. Selain itu penggunaan *resource* FPGA dari implementasi sistem dijelaskan pada Tabel 4.3. Pada Gambar 4.10 dapat dilihat blok-blok utama dari sistem yang sesuai dengan diagram blok perancangan yang telah direncanakan sebelumnya. Dari Tabel 4.3 terlihat bahwa hasil implementasi sistem masih menyisakan banyak *resource*. Penggunaan *resource* terbanyak adalah RAMB16BWER. *Resource* ini merupakan memori yang sebagian besar digunakan untuk menyimpan citra.



Gambar 4.10 Rangkaian skematik keseluruhan sistem

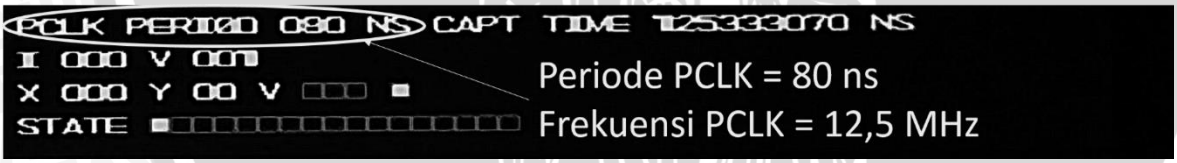
Tabel 4.3 Data penggunaan *resource* FPGA Nexys3 Spartan-6

Penggunaan *resource* FPGA	**Dipakai**	**Tersedia**	**Utilisasi**
Slice registers	1491	18224	8%
Slice LUTs	4831	9112	53%
RAMB16BWERs	20	32	62%
RAMB8BWERs	4	64	6%
BUFG/BUFGMUXs	2	16	12%
DSP48A1s	6	32	18%

4.4 Hasil Pengujian Perangkat Keras

Setelah seluruh perangkat keras sistem dirangkai dan diaktifkan, didapatkan tampilan monitor seperti yang terlihat pada Gambar 4.11. Terlihat bahwa tampilan monitor sudah sesuai dibandingkan dengan skema pada Gambar 3.19. Tampilan citra juga memperlihatkan tulisan plat nomor "AG 2920 LB" dengan jelas. Gambar 4.12 difokuskan pada teks keterangan (nomor 11) untuk memperlihatkan tampilan tulisan periode PCLK. Terlihat bahwa periode PCLK yang ditampilkan adalah sebesar 80 ns, sehingga telah sesuai dengan *setting* yang ditetapkan.

Gambar 4.11 Tampilan monitor saat pengujian sistem



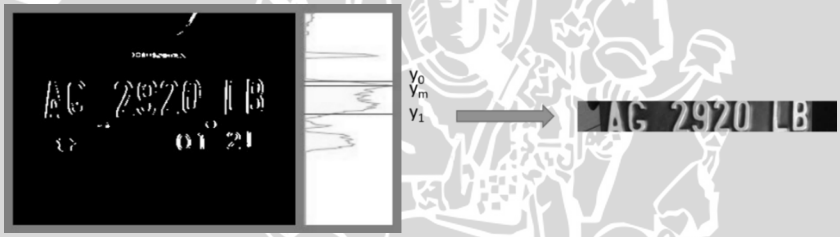
Gambar 4.12 Tampilan tulisan periode PCLK

4.5 Hasil Pengujian Sistem Tahap Lokalisasi Plat Nomor

Gambar 4.13 menunjukkan hasil pengujian deteksi tepi vertikal pada simulasi program. Dapat diamati bahwa tepi vertikal tulisan plat nomor dapat terlihat dengan jelas. Terlihat pula bahwa tulisan tampak tidak utuh pada citra deteksi tepi vertikal. Hal ini disebabkan tepi yang dideteksi hanya tepi vertikal saja, sehingga tepi horizontal tidak terlihat (dapat dilihat pada huruf L yang tampak seperti huruf I). Hal ini tidak menjadi masalah pada sistem karena memang hanya dibutuhkan tepi vertikal saja untuk mendeteksi keseluruhan area plat nomor. Dipilih nilai sensitivitas deteksi tepi vertikal $C_e = 48$ (disubstitusikan ke Persamaan (3-1)).

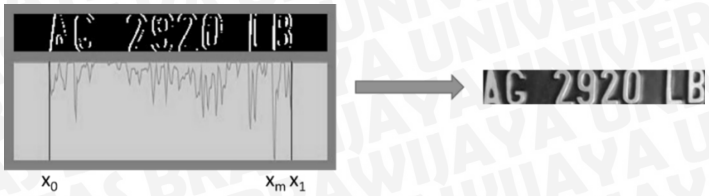
Gambar 4.13 Hasil pengujian deteksi tepi vertikal pada simulasi program

Gambar 4.14 menunjukkan hasil pengujian *band clipping* pada simulasi program. Pada grafik proyeksi vertikal di sebelah kanan citra hasil deteksi tepi vertikal, di titik-titik y tertentu ditandai dengan y_M, y_0 dan y_1 yang merupakan titik proyeksi maksimum, batas atas dan batas bawah plat nomor. Penentuan titik-titik ini terbukti berhasil ditandai dengan letaknya pada puncak grafik yang bersesuaian dengan letak tulisan plat nomor pada citra deteksi tepi vertikal, sehingga didapat citra hasil pemotongan batas atas dan batas bawah seperti yang terlihat di sebelah kanan grafik pada Gambar 4.14. Dipilih nilai kaki puncak grafik $C_y = 0{,}25$ (disubstitusikan ke Persamaan (3-3) dan (3-4)) dan tinggi minimum plat nomor $C_{hm} = 4$ (disubstitusikan ke Persamaan (3-5)).



Gambar 4.14 Hasil pengujian *band clipping* pada simulasi program

Gambar 4.15 menunjukkan hasil pengujian *plate clipping* pada simulasi program. Menggunakan area hasil pemotongan pada proses sebelumnya, citra deteksi tepi vertikal diproses menjadi grafik proyeksi horizontal di bawahnya. Di titik-titik x tertentu ditandai dengan x_M, x_0 dan x_1 yang merupakan titik proyeksi maksimum, batas kiri dan batas kanan plat nomor. Penentuan titik-titik ini terbukti berhasil ditandai dengan letaknya pada rentang grafik yang bersesuaian dengan letak tulisan plat nomor pada citra deteksi tepi vertikal, sehingga didapat citra hasil pemotongan batas kiri dan batas kanan seperti yang terlihat di sebelah kanan grafik pada Gambar 4.15. Dipilih nilai kaki puncak grafik $C_x = 0{,}125$ (disubstitusikan ke Persamaan (3-8) dan (3-9)).



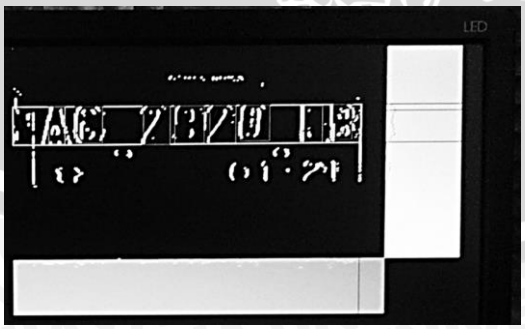
Gambar 4.15 Hasil pengujian *plate clipping* pada simulasi program

Tabel 4.4 menunjukkan data pengujian tahap lokalisasi plat nomor pada simulasi program. Dari Tabel 4.4 terlihat bahwa seluruh percobaan pada citra sampel uji berhasil mendeteksi area plat nomor. Didapatkan persentase kesalahan keseluruhan adalah sebesar 0% (persentase keberhasilan 100%). Dapat disimpulkan bahwa algoritma yang telah dirancang untuk tahap lokalisasi plat nomor bekerja dengan baik dan dapat diandalkan.

Tabel 4.4 Data pengujian tahap lokalisasi plat nomor pada simulasi program

Sampel	Keterangan
N 6203 AB	Berhasil
N 6173 AO	Berhasil
AG 2920 LB	Berhasil
N 1478 AZ	Berhasil

Gambar 4.16 menunjukkan hasil pengujian tahap lokalisasi plat nomor pada pengujian langsung. Pada citra deteksi tepi vertikal dapat diamati bahwa tepi vertikal tulisan plat nomor terlihat dengan jelas seperti pada simulasi program. Pada grafik proyeksi vertikal dapat ditemukan tanda berupa garis yang masing-masing menandai titik proyeksi maksimum, batas atas dan batas bawah plat nomor, sehingga terlihat bahwa proses *band clipping* telah berhasil menentukan batas atas dan batas bawah plat nomor seperti pada simulasi program. Pada grafik proyeksi horizontal juga ditandai letak batas kiri dan batas kanan plat nomor, sehingga terlihat bahwa proses *plate clipping* telah berhasil menentukan batas kiri dan batas kanan plat nomor seperti pada simulasi program. Pada citra deteksi tepi vertikal terdapat persegi panjang yang menandai area plat nomor hasil tahap lokalisasi plat nomor secara keseluruhan. Persegi panjang ini semakin mempertegas hasil akhir tahap lokalisasi plat nomor. Pemilihan nilai konstanta sama seperti pada simulasi program, kecuali pada deteksi vertikal dipilih nilai $C_e = 8$ untuk menyesuaikan dengan kontras pencahayaan yang rendah pada saat pengujian langsung.



Gambar 4.16 Hasil pengujian tahap lokalisasi plat nomor pada pengujian langsung

Tabel 4.5, Tabel 4.6, Tabel 4.7 dan Tabel 4.8 menunjukkan data pengujian tahap lokalisasi plat nomor pada pengujian langsung, masing-masing untuk jarak 30 cm, 40 cm, 50 cm dan 60 cm. Dari tabel-tabel tersebut terlihat bahwa seluruh percobaan pada citra

sampel uji berhasil mendeteksi area plat nomor seperti pada simulasi program. Persentase kesalahan untuk setiap variasi jarak dapat dilihat pada Tabel 4.9. Variasi jarak akan menimbulkan variasi ukuran plat nomor pada citra, sehingga jarak pada percobaan dibuat supaya ukuran plat nomor tidak kurang dari kriteria minimum dan maksimum sistem. Dari ukuran plat terkecil (jarak 60 cm) hingga terbesar (jarak 30 cm) sistem masih mampu mendeteksi area plat nomor, sehingga dapat disimpulkan bahwa jarak tidak berpengaruh pada sistem dalam mendeteksi area plat nomor asalkan memenuhi kriteria ukuran plat nomor. Didapatkan persentase kesalahan keseluruhan adalah sebesar 0%.

Tabel 4.5 Data pengujian tahap lokalisasi plat nomor pada pengujian langsung dengan jarak 30 cm

Sampel	Keterangan
N 6203 AB	Berhasil
N 6173 AO	Berhasil
AG 2920 LB	Berhasil
N 1478 AZ	Berhasil

Tabel 4.6 Data pengujian tahap lokalisasi plat nomor pada pengujian langsung dengan jarak 40 cm

Sampel	Keterangan
N 6203 AB	Berhasil
N 6173 AO	Berhasil
AG 2920 LB	Berhasil
N 1478 AZ	Berhasil

Tabel 4.7 Data pengujian tahap lokalisasi plat nomor pada pengujian langsung dengan jarak 50 cm

Sampel	Keterangan
N 6203 AB	Berhasil
N 6173 AO	Berhasil
AG 2920 LB	Berhasil
N 1478 AZ	Berhasil

Tabel 4.8 Data pengujian tahap lokalisasi plat nomor pada pengujian langsung dengan jarak 60 cm

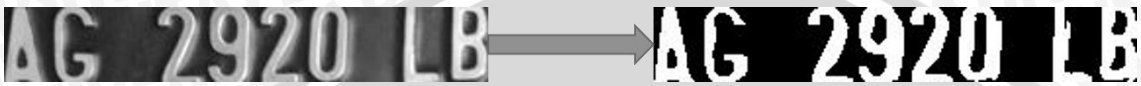
Sampel	Keterangan
N 6203 AB	Berhasil
N 6173 AO	Berhasil
AG 2920 LB	Berhasil
N 1478 AZ	Berhasil

Tabel 4.9 Data persentase kesalahan tahap lokalisasi plat nomor pada pengujian langsung

Jarak (cm)	Persentase kesalahan (%)
30	0%
40	0%
50	0%
60	0%
Persentase keseluruhan	0%

4.6 Hasil Pengujian Sistem Tahap Segmentasi Karakter

Gambar 4.17 menunjukkan hasil pengujian pengambangan pada simulasi program. Dapat diamati bahwa tulisan plat nomor berwarna putih dan latar berwarna hitam. Terlihat pula bahwa tulisan tampak utuh dan tidak ada bagian yang hilang. Keutuhan tulisan dari proses ini penting karena mempengaruhi kesempurnaan hasil segmentasi karakter, sehingga mempengaruhi keberhasilan pada tahap pengenalan karakter pula. Dipilih nilai ambang $C_T = 128$ (disubstitusikan ke Persamaan (2-5)).



Gambar 4.17 Hasil pengujian pengambangan pada simulasi program

Gambar 4.18 menunjukkan hasil pengujian segmentasi karakter pada simulasi program. Pada grafik proyeksi horizontal di bawah citra hasil pengambangan, di titik-titik x tertentu telah ditandai, yang merupakan batas-batas horizontal untuk setiap karakter (X_{c0} dan X_{c1}). Penentuan titik-titik ini terbukti berhasil ditandai dengan letaknya pada puncak grafik yang bersesuaian dengan letak tulisan setiap karakter pada citra plat nomor hasil pengambangan, sehingga didapat citra hasil pemotongan batas atas dan batas bawah karakter. Setelah diketahui letak setiap karakter yang dapat dideteksi secara horizontal, ditambah dengan pemrosesan proyeksi vertikal untuk setiap karakter supaya batas atas dan bawah karakter dapat tersegmentasi dengan tepat. Pada umumnya perbedaan batas atas dan bawah karakter dengan batas awal tidak terlalu signifikan, karena karakter pada plat nomor letaknya sejajar dan tingginya sama. Akan tetapi pemrosesan secara vertikal tetap diperlukan untuk kondisi-kondisi tertentu yang dapat menyebabkan karakter tidak tersegmentasi dengan tepat, karena hasil segmentasi sangat berpengaruh pada performansi di tahap selanjutnya. Setelah tahap segmentasi selesai didapat citra hasil segmentasi setiap karakter seperti yang terlihat di sebelah kanan grafik pada Gambar 4.18. Dipilih nilai kaki puncak grafik proyeksi horizontal $C_{xc} = 0{,}0625$ (disubstitusikan ke Persamaan (3-13) dan (3-14)), kaki puncak grafik proyeksi vertikal $C_{yc} = 0{,}0625$ (disubstitusikan ke Persamaan (3-3) dan (3-4)) dan $C_{rhm} = 0{,}5$ (disubstitusikan ke Persamaan (3-17)).

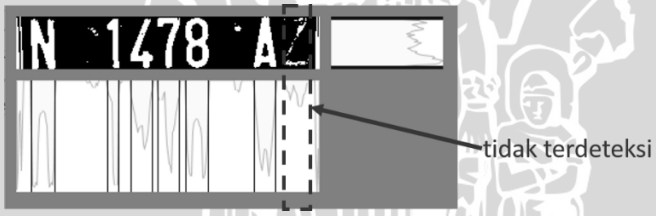
Gambar 4.18 Hasil pengujian segmentasi karakter pada simulasi program

Tabel 4.10 menunjukkan data pengujian tahap segmentasi karakter pada simulasi program. Terdapat 1 kesalahan yang dapat dibahas dari Tabel 4.10 dan telah diilustrasikan pada Gambar 4.19. Dilihat dari sampel plat N 1478 AZ pada Gambar 4.19, warna karakter 'Z' memudar karena plat tergores dan karakter 'Z' terlihat tidak utuh dibanding lainnya pada citra pengambangan, akibatnya karakter tersebut tidak tersegmentasi. Dapat diambil kesimpulan bahwa kondisi fisik plat dapat mempengaruhi kerja sistem.

Untuk menganalisis tingkat kesalahan pada tahap ini, kesalahan yang dihitung meliputi kesalahan karakter yang tidak tersegmentasi, area bukan karakter yang tersegmentasi, maupun karakter yang tidak tersegmentasi sempurna. Didapatkan kesalahan maksimum tahap segmentasi karakter pada simulasi program adalah 1 segmen. Dari 4 kali percobaan, 3 percobaan berhasil melakukan segmentasi karakter dengan sempurna. Dapat disimpulkan bahwa algoritma yang telah dirancang untuk tahap segmentasi karakter dapat bekerja meskipun belum 100% akurat.

Tabel 4.10 Data pengujian tahap segmentasi karakter pada simulasi program

Sampel	Keterangan
N 6203 AB	Berhasil
N 6173 AO	Berhasil
AG 2920 LB	Berhasil
N 1478 AZ	Karakter 'Z' tidak terdeteksi



Gambar 4.19 Kesalahan segmentasi karakter pada simulasi program

Gambar 4.20 menunjukkan hasil pengujian tahap segmentasi karakter pada pengujian langsung. Pada citra pengambangan dapat diamati bahwa tulisan plat nomor berwarna putih dan latar berwarna hitam seperti pada simulasi program. Proses segmentasi karakter telah berhasil menentukan batas-batas setiap karakter seperti pada simulasi program, terlihat dari garis penanda pada grafik yang menunjukkan batas-batas tersebut. Pada citra pengambangan di setiap area karakter ditandai dengan persegi panjang. Persegi panjang ini semakin mempertegas hasil akhir tahap segmentasi karakter. Pemilihan nilai konstanta sama seperti pada simulasi program.

Tabel 4.11, Tabel 4.12, Tabel 4.13 dan Tabel 4.14 menunjukkan data pengujian tahap segmentasi karakter pada pengujian langsung, masing-masing untuk jarak 30 cm, 40 cm, 50 cm dan 60 cm. Dari tabel-tabel tersebut terdapat beberapa kesalahan dengan penyebab yang

sama seperti pada simulasi program. Karakter ‘Z’ pada sampel plat N 1478 AZ tetap tidak dapat disegmentasi dengan baik, karena memang kondisi fisiknya yang cacat. Karakter ‘O’ pada sampel plat N 6173 AO dengan jarak pengujian 60 cm tidak dapat tersegmentasi karena jarak yang jauh dan karakter yang dekat dengan garis tepi plat menyebabkan karakter O terlihat menempel dengan garis tepi, sehingga tidak dapat terbaca sebagai karakter.

Kesalahan total untuk setiap variasi jarak dapat dilihat pada Tabel 4.15. Dari kesalahan total tersebut terlihat bahwa performa sistem pada tahap segmentasi karakter tidak jauh berbeda dari jarak terdekat hingga terjauh. Dapat disimpulkan bahwa jarak tidak berpengaruh pada sistem dalam melakukan segmentasi karakter, sama seperti pada tahap lokalisasi plat nomor. Hal ini terkait dengan cara kerja kedua tahap yang sama-sama menggunakan metode analisis proyeksi horizontal dan vertikal, sehingga dapat diambil kesimpulan bahwa metode ini tidak terpengaruh dengan ukuran objek pada citra. Didapatkan kesalahan maksimum tahap segmentasi karakter pada pengujian langsung adalah 2 segmen. Dari 16 kali percobaan, 11 percobaan berhasil melakukan segmentasi karakter dengan sempurna.

Gambar 4.20 Hasil pengujian tahap segmentasi karakter pada pengujian langsung

Tabel 4.11 Data pengujian tahap segmentasi karakter pada pengujian langsung dengan jarak 30 cm

Sampel	Keterangan
N 6203 AB	Berhasil
N 6173 AO	Berhasil
AG 2920 LB	Berhasil
N 1478 AZ	Karakter ‘Z’ tidak terdeteksi

Tabel 4.12 Data pengujian tahap segmentasi karakter pada pengujian langsung dengan jarak 40 cm

Sampel	Keterangan
N 6203 AB	Berhasil
N 6173 AO	Berhasil
AG 2920 LB	Berhasil
N 1478 AZ	Karakter ‘Z’ tidak terdeteksi, kesalahan deteksi setelah karakter ‘Z’

Tabel 4.13 Data pengujian tahap segmentasi karakter pada pengujian langsung dengan jarak 50 cm

Sampel	Keterangan
N 6203 AB	Berhasil
N 6173 AO	Berhasil
AG 2920 LB	Berhasil
N 1478 AZ	Karakter ‘Z’ terpotong

Tabel 4.14 Data pengujian tahap segmentasi karakter pada pengujian langsung dengan jarak 60 cm

Sampel	Keterangan
N 6203 AB	Berhasil
N 6173 AO	Karakter ‘O’ tidak terdeteksi
AG 2920 LB	Berhasil
N 1478 AZ	Karakter ‘Z’ terpotong

Tabel 4.15 Data kesalahan total tahap segmentasi karakter pada pengujian langsung

Jarak (cm)	Kesalahan total (segmen)
30	1
40	2
50	1
60	2

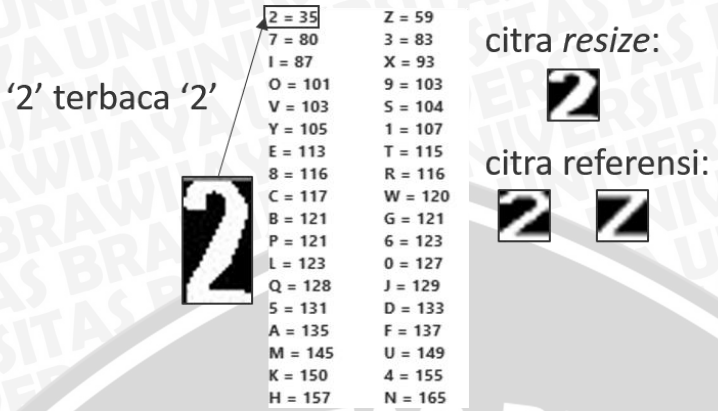
4.7 Hasil Pengujian Sistem Tahap Pengenalan Karakter Optis

Gambar 4.21 menunjukkan hasil pengujian *resize* pada simulasi program. Hasil pengujian menunjukkan bahwa karakter dapat diubah ukurannya dengan benar. Dimensi karakter diubah secara proporsional dengan bentuk aslinya. Setiap citra karakter yang telah diubah ukurannya akan dibandingkan dengan citra referensi yang berdimensi sama (16 × 16 piksel) pada metode KNN.

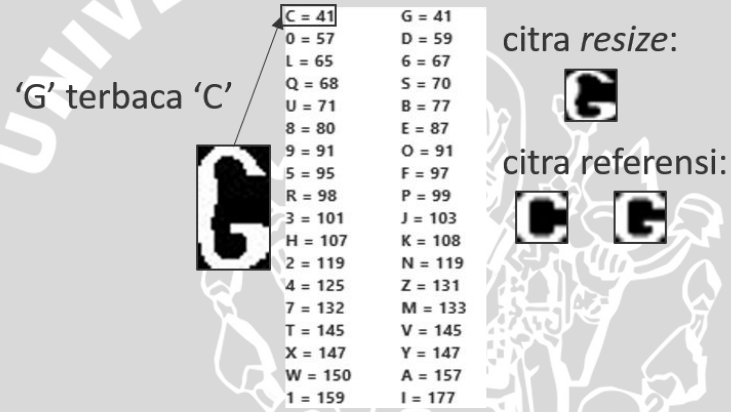
Gambar 4.21 Hasil pengujian *resize* pada simulasi program

Gambar 4.22 dan Gambar 4.23 menunjukkan hasil tampilan *distance* dari metode KNN pada simulasi program yang telah diurutkan dari *distance* terkecil ke terbesar. Nilai *distance* menunjukkan jumlah piksel yang berbeda nilainya pada setiap koordinat antara citra masukann dengan citra referensi. Gambar 4.22 menunjukkan nilai-nilai *distance* untuk karakter ‘2’ pada plat nomor (karakter kelima dari kiri). Terlihat bahwa *distance* terkecil adalah perbandingan dengan citra referensi karakter ‘2’, sehingga karakter telah berhasil dikenali dengan benar. Karakter terdekat berikutnya adalah ‘Z’ dengan terpaut selisih nilai sebesar 59-35=24, karena memang bentuk kedua karakter tersebut mirip. Dengan melihat

selisih yang besar antara karakter ‘2’ dengan karakter lainnya, maka pada kasus ini metode KNN telah dapat bekerja dengan baik untuk mengenali karakter.



Gambar 4.22 Hasil tampilan *distance* metode KNN pada simulasi program untuk karakter kelima sampel plat AG 2920 LB

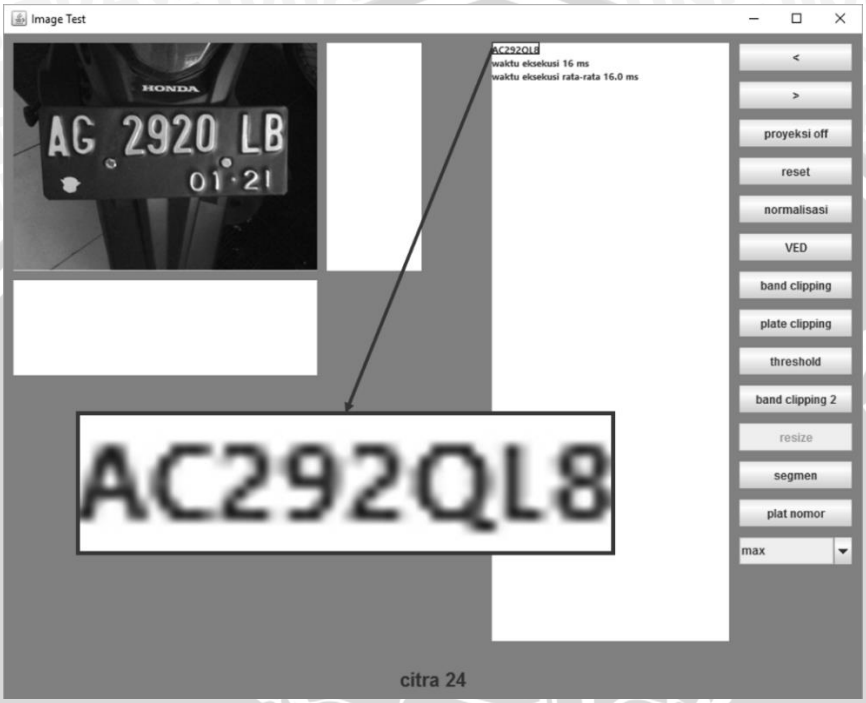


Gambar 4.23 Hasil tampilan *distance* metode KNN pada simulasi program untuk karakter kedua sampel plat AG 2920 LB

Gambar 4.23 menunjukkan nilai-nilai *distance* untuk karakter ‘G’ pada plat nomor (karakter kedua dari kiri). Telah terjadi kesalahan pengenalan karakter pada kasus ini karena *distance* terkecil adalah perbandingan dengan citra referensi karakter ‘C’. Karakter ‘G’ berada di urutan kedua, akan tetapi dengan nilai *distance* yang sama. Hal ini terjadi karena sistem menentukan karakter dengan mencari *distance* terkecil mulai dari urutan terendah. Urutan yang dimaksud disini adalah urutan karakter pada sistem, yaitu dimulai dari angka 0-9 kemudian dilanjutkan huruf A-Z. Jika dalam pencarian ditemukan nilai yang sama, maka sistem tetap memilih karakter yang sebelumnya. Dari Gambar 4.22 dan Gambar 4.23 dapat diambil kesimpulan bahwa karakter yang mirip cenderung memiliki selisih *distance* yang sedikit, sehingga mempengaruhi akurasi sistem dalam mengenali karakter.

Gambar 4.24 menunjukkan hasil pengujian tahap pengenalan karakter optis pada simulasi program. Setelah menekan tombol untuk mendeteksi plat nomor maka perintah untuk menjalankan sistem ANPR dieksekusi, hingga pada tahap akhir yaitu pengenalan

karakter optis ditampilkan hasil deteksi karakter plat nomor pada teks keterangan, serta waktu eksekusi dan waktu rata-rata eksekusi. Dapat dilihat bahwa dari total 8 karakter terdapat 3 kesalahan pengenalan karakter, seluruhnya disebabkan oleh kemiripan dengan karakter lain. Semakin jelas terlihat bahwa metode KNN sensitif terhadap kemiripan satu karakter dengan lainnya, meskipun sebelumnya tahap segmentasi karakter sudah berjalan dengan benar.



Gambar 4.24 Hasil pengujian tahap pengenalan karakter optis pada simulasi program

Tabel 4.16 menunjukkan data pengujian tahap pengenalan karakter optis pada simulasi program. Terdapat beberapa kesalahan yang dapat dilihat pada Tabel 4.16. Penyebabnya tidak lain adalah seperti yang telah dijelaskan sebelumnya, bahwa kemiripan karakter mempengaruhi proses pengenalan karakter. Terlihat bahwa karakter ‘0’ muncul sebanyak 2 kali dan keduanya terbaca menjadi ‘Q’. Dapat disimpulkan bahwa sistem masih kesulitan untuk membedakan beberapa karakter.

Untuk menganalisis tingkat kesalahan pada tahap ini, kesalahan yang dihitung meliputi kesalahan pengenalan karakter pada segmen karakter yang benar, tidak termasuk segmen yang bukan karakter. Didapatkan kesalahan maksimum tahap pengenalan karakter optis adalah 3 karakter. Dari 4 percobaan hingga tahap akhir, terdapat 1 percobaan yang berhasil mengenali karakter dengan sempurna, yaitu sampel plat N 6173 AO. Dari total 28 karakter plat nomor, 24 karakter dapat dikenali dengan benar. Dapat disimpulkan bahwa algoritma yang telah dirancang untuk tahap pengenalan karakter optis dapat bekerja meskipun masih perlu pengembangan supaya sistem lebih akurat.

Tabel 4.16 Data pengujian tahap pengenalan karakter optis pada simulasi program

Sampel	Output
N 6203 AB	N 62Q3 AB
N 6173 AO	N 6173 AO
AG 2920 LB	AC 292Q L8
N 1478 AZ	N 1478 A

Gambar 4.25 menunjukkan hasil pengujian tahap pengenalan karakter optis pada pengujian langsung. Hasil *resize* setiap segmen karakter diperlihatkan untuk dibandingkan dengan hasil pengenalan karakter. Terlihat bahwa citra hasil *resize* setiap segmen berukuran sama dengan citra hasil pengenalan karakter, selain itu karakter yang diperlihatkan tampak utuh pertanda bahwa karakter telah disegmentasi dengan baik pada proses sebelumnya. Pada grafik ditunjukkan besarnya *distance* setiap citra hasil segmentasi dengan citra referensi setiap karakter. Terdapat penanda pada grafik yang menunjukkan karakter dengan nilai *distance* terkecil. Terlihat bahwa karakter pertama terbaca dengan karakter ‘A’ karena memilki nilai *distance* terkecil. Dari seluruh karakter yang dikenali terdapat 2 karakter yang salah, yaitu karakter ‘0’ menjadi ‘Q’ dan karakter ‘B’ menjadi ‘8’. Kesalahan ini terjadi karena kemiripan karakter seperti yang terjadi pada simulasi program.

Gambar 4.25 Hasil pengujian tahap pengenalan karakter optis pada pengujian langsung

Tabel 4.17, Tabel 4.18, Tabel 4.19 dan Tabel 4.20 menunjukkan data pengujian tahap pengenalan karakter optis pada pengujian langsung, masing-masing untuk jarak 30 cm, 40 cm, 50 cm dan 60 cm. Dari tabel-tabel tersebut terdapat beberapa kesalahan dengan penyebab yang sama seperti pada simulasi program. Kesalahan total untuk setiap variasi jarak dapat dilihat pada Tabel 4.21. Dari kesalahan total tersebut terdapat kecenderungan kesalahan yang semakin besar dengan jarak yang semakin jauh. Dapat disimpulkan bahwa semakin jauh jarak maka semakin rendah tingkat akurasi tahap pengenalan karakter optis. Hal ini disebabkan oleh ukuran plat yang semakin kecil jika jarak semakin jauh, sehingga area piksel segmen terlalu sedikit dan kualitas citra hasil *resize* menjadi rendah. Area piksel yang luas akan menghasilkan kualitas citra yang lebih baik sehingga lebih akurat ketika diproses dengan metode KNN. Didapatkan kesalahan maksimum tahap pengenalan karakter

optis adalah 3 karakter. Dari 16 percobaan hingga tahap akhir, terdapat 3 percobaan yang berhasil mengenali karakter dengan sempurna, dan semuanya adalah sampel plat yang sama yaitu N 6173 AO. Dari total 111 karakter plat nomor, 95 karakter dapat dikenali dengan benar.

Tabel 4.17 Data pengujian tahap pengenalan karakter optis pada pengujian langsung dengan jarak 30 cm

Sampel	Output
N 6203 AB	N 62Q3 AB
N 6173 AO	N 6173 AO
AG 2920 LB	AG 292Q L8
N 1478 AZ	N 1478 A

Tabel 4.18 Data pengujian tahap pengenalan karakter optis pada pengujian langsung dengan jarak 40 cm

Sampel	Output
N 6203 AB	N 62Q3 AB
N 6173 AO	N 6173 AO
AG 2920 LB	AC 292Q L8
N 1478 AZ	N 1478 A1

Tabel 4.19 Data pengujian tahap pengenalan karakter optis pada pengujian langsung dengan jarak 50 cm

Sampel	Output
N 6203 AB	N 62Q3 AB
N 6173 AO	N 6173 AO
AG 2920 LB	AC 292Q LB
N 1478 AZ	N 1470 A4

Tabel 4.20 Data pengujian tahap pengenalan karakter optis pada pengujian langsung dengan jarak 60 cm

Sampel	Output
N 6203 AB	W 62Q3 AB
N 6173 AO	N 6171 A
AG 2920 LB	AG 292Q L8
N 1478 AZ	N 1478 A0

Tabel 4.21 Data kesalahan total tahap pengenalan karakter optis pada pengujian langsung

Jarak (cm)	Kesalahan total (karakter)
30	3
40	4
50	4
60	5

4.8 Analisis Keseluruhan Sistem

Pada simulasi program, dapat disimpulkan bahwa sistem dapat bekerja akan tetapi masih diperlukan pengembangan untuk meningkatkan akurasi. Dari 4 percobaan yang telah dilakukan, sistem telah mendeteksi sebanyak 24 karakter dengan benar dari total 29 karakter

plat nomor. Waktu eksekusi untuk setiap percobaan dapat dilihat pada Tabel 4.22. Rata-rata waktu eksekusi pada simulasi program adalah 15,25 ms. Hasil analisis simulasi program dapat dilihat pada Tabel 4.23.

Tabel 4.22 Data waktu eksekusi sistem pada simulasi program

Sampel	Waktu eksekusi (ms)
N 6203 AB	15
N 6173 AO	14
AG 2920 LB	17
N 1478 AZ	15
Rata-rata	15,25

Tabel 4.23 Data hasil analisis simulasi program

Parameter	Nilai
Persentase keberhasilan lokalisasi plat nomor	100% (4 dari 4 percobaan)
Persentase keberhasilan segmentasi karakter	75% (3 dari 4 percobaan)
Kesalahan maksimum segmentasi karakter	1 segmen
Persentase karakter tersegmentasi	96,55% (28 dari 29 karakter)
Persentase keberhasilan pengenalan karakter optis	25% (1 dari 4 percobaan)
Kesalahan maksimum pengenalan karakter optis	3 karakter
Persentase karakter dikenali (pengenalan karakter optis)	85,71% (24 dari 28 karakter)
Persentase karakter dikenali (sistem)	82,76% (24 dari 29 karakter)
Rata-rata waktu eksekusi	15,25 ms

Hasil yang didapat secara keseluruhan pada pengujian langsung tidak berkontradiksi dengan simulasi program, sehingga dapat disimpulkan bahwa pengujian langsung telah bekerja sejalan dengan simulasi program. Dari 16 percobaan yang telah dilakukan, sistem telah mendeteksi sebanyak 95 karakter dengan benar dari total 116 karakter plat nomor. Waktu eksekusi untuk setiap percobaan dapat dilihat pada Tabel 4.24. Dapat dilihat dari tabel bahwa semakin jauh jarak waktu eksekusi semakin singkat. Hal ini disebabkan semakin jauh jarak semakin kecil ukuran plat nomor pada citra, sehingga piksel yang diproses juga semakin sedikit. Semakin sedikit piksel yang diproses, maka waktu eksekusi juga semakin singkat, akan tetapi karakter juga semakin sulit terbaca, sehingga kemungkinan kesalahan juga semakin besar. Dapat diamati pada percobaan sampel plat N 1478 AZ dengan jarak 30 cm waktu eksekusinya lebih singkat dibandingkan dengan jarak 40 cm. Hal ini disebabkan oleh jumlah segmen karakter yang terdeteksi lebih sedikit. Sehingga dapat disimpulkan bahwa semakin dekat jarak dan/atau semakin banyak jumlah karakter akan menyebabkan waktu eksekusi menjadi lebih lama. Rata-rata waktu eksekusi pada pengujian langsung (1,833680625 ms) lebih singkat daripada simulasi program (15,25 ms). Dapat disimpulkan bahwa implementasi sistem pada FPGA lebih unggul dibandingkan dengan implementasi pada program dari segi performansi. Hasil analisis pengujian langsung dapat dilihat pada

Tabel 4.25. Analisis pengujian langsung menunjukkan hasil data yang secara kualitatif menurun dibandingkan dengan pada simulasi program. Hal ini disebabkan oleh frekuensi percobaan yang ditingkatkan menghasilkan kesalahan baru yang lebih bervariasi dibandingkan pada simulasi program.

Tabel 4.24 Data waktu eksekusi sistem pada pengujian langsung

Jarak (cm)	Waktu eksekusi (ns)				Rata-rata waktu eksekusi (ms)
	N 6203 AB	N 6173 AO	AG 2920 LB	N 1478 AZ	
30	2083450	1738340	2317180	1770830	1,977450
40	1813160	1734100	2176310	1833860	1,8893575
50	1790400	1610290	1982210	1773450	1,7890875
60	1733140	1338070	1906710	1737390	1,6788275
Rata-rata keseluruhan					1,833680625

Tabel 4.25 Data hasil analisis pengujian sistem keseluruhan

Parameter	Nilai
Persentase keberhasilan lokalisasi plat nomor	100% (16 dari 16 percobaan)
Persentase keberhasilan segmentasi karakter	68,75% (11 dari 16 percobaan)
Kesalahan maksimum segmentasi karakter	2 segmen
Persentase karakter tersegmentasi	95,69% (111 dari 116 karakter)
Persentase keberhasilan pengenalan karakter optis	18,75% (3 dari 16 percobaan)
Kesalahan maksimum pengenalan karakter optis	3 karakter
Persentase karakter dikenali (pengenalan karakter optis)	85,59% (95 dari 111 karakter)
Persentase karakter dikenali (sistem)	81,90% (95 dari 116 karakter)
Rata-rata waktu eksekusi	1,83 ms

Untuk melakukan analisis lebih jauh, maka hasil pengujian ini dibandingkan dengan hasil pengujian dari penelitian yang sudah dilakukan sebelumnya. Perbandingan ini merujuk kepada beberapa referensi. Sistem yang pertama diimplementasikan pada software Java menggunakan metode yang sama untuk ketiga tahap ANPR (Mellolo, 2012). Perbedaannya adalah sistem yang telah didesain penulis menggunakan metode yang telah disederhanakan lebih jauh dan tidak menggunakan metode-metode tambahan dibandingkan dengan sistem tersebut. Sistem yang kedua diimplementasikan pada software OpenCV dengan menggunakan metode yang berbeda (Lim, 2003). Sistem yang ketiga adalah sistem yang terdiri dari subsistem pendeteksi kendaraan, kamera *webcam*, dan komputer, diimplementasikan pada software Java dengan menggunakan metode yang berbeda (Hermawati, 2010). Perbandingan untuk setiap sistem dapat dilihat pada Tabel 4.26 dan Tabel 4.27.

Sistem 1 adalah sistem yang paling mirip dengan sistem yang telah dirancang, karena menggunakan dasar metode yang sama. Hasil pengujian keduanya menunjukkan kecenderungan yang sama, yaitu sistem mengalami penurunan performa dari tahap lokalisasi

plat nomor hingga pengenalan karakter optis. Penyebabnya sama, yaitu dikarenakan sistem masih kesulitan membedakan karakter yang mirip. Tingkat keberhasilan sistem yang telah dirancang lebih tinggi dari sistem 1 dilihat dari karakter yang berhasil dikenali, walaupun sistem yang dirancang memakai metode yang telah disederhanakan untuk menyesuaikan *resource* FPGA yang terbatas, akan tetapi sampel uji masih terlalu sedikit. Jika sistem dikembangkan dari segi memori penyimpanan citra dan *database* karakter sehingga dapat mengimbangi tingkat pemrosesan pada PC, tentunya akurasi akan dapat lebih ditingkatkan lagi.

Tabel 4.26 Spesifikasi implementasi sistem

Sistem	Metode			*Tools/ Software*	Alat Pemroses/Frekuensi Kerja
	Lokalisasi Plat Nomor	Segmentasi Karakter	Pengenalan Karakter Optis		
Simulasi Program	Deteksi tepi vertikal dan analisis proyeksi	Pengambangan dan analisis proyeksi	*Resampling* dan 1-Nearest Neighbor	Java	PC Intel Core i3/2,2 GHz
Pengujian Langsung	Deteksi tepi vertikal dan analisis proyeksi	Pengambangan dan analisis proyeksi	*Resampling* dan 1-Nearest Neighbor	Xilinx XST	*Board* Nexys3 FPGA Spartan-6/100 MHz
Sistem 1	Deteksi tepi vertikal dan analisis proyeksi	Pengambangan dan analisis proyeksi	*Resampling, medial axis transform* dan 1-Nearest Neighbor	Java	PC/-
Sistem 2	Pengambangan dan *Bounding Box Checking*	Pengambangan dan *Bounding Box Checking*	*Resampling* dan 1-Nearest Neighbor	OpenCV	PC Pentium II/400 MHz
Sistem 3	Pengambangan dan FFT	Proses morfologi dan analisis proyeksi	*Hidden Markov Model*	Java	PC/-

Tabel 4.27 Perbandingan Hasil implementasi sistem

Sistem	Keberhasilan (plat) / (karakter)			Waktu eksekusi
	Lokalisasi Plat Nomor	Segmentasi Karakter	Sistem / Pengenalan Karakter Optis	
Simulasi Program	100% (4/4)	75% (3/4) / 96,55% (28/29)	25% (1/4) / 85,71% (24/28)	1,833680625 ms
Pengujian Langsung	100% (16/16)	68,75% (11/16) / 95,69% (111/116)	18,75% (3/16) / 85,59% (95/111)	15,25 ms
Sistem 1	100% (25/25)	- / 94,48%	- / 69,52%	-
Sistem 2	69,26% (178/257)	58,91% (119/202) / 85,87% (1051/1224)	42,80% (110/257) / 94,67% (995/1051)	-
Sistem 3	95% (57/60)	- / 94,48%	- / 93,20%	1,522 s

Sistem 2 menggunakan *software* OpenCV yang telah dikenal untuk aplikasi *image processing*, dan dapat dikatakan yang paling teruji karena sampel ujinya paling banyak dan bervariasi. Dengan metode yang berbeda, sistem ini dapat mengenali karakter dengan

akurasi yang tinggi. Penyebab ketidakakuratan sistem sebagian besar disebabkan oleh pemrosesan pada tahap lokalisasi plat nomor. Melihat dari hasil pengujian sistem ini, maka perlu dilakukan pengujian lebih lanjut pada sistem yang dirancang dengan sampel yang lebih banyak seperti pada sistem ini, sehingga lebih teruji dan dapat lebih diketahui tingkat kehandalannya untuk dievaluasi dan dikembangkan.

Sistem 3 adalah sistem purwarupa *real-time* untuk akses parkir. Metode yang digunakan juga berbeda dari sistem yang telah dirancang. Dengan metode yang kompleks, sistem ini dapat mencapai akurasi tertinggi dari sistem lainnya, terlihat dari tingkat keberhasilan yang memuaskan pada setiap tahapnya. Akan tetapi kekurangan sistem terlihat dari tingkat kecepatan pemrosesan sebesar 1,522 detik, jauh lebih lama dibandingkan hasil perancangan sistem yang telah bekerja dalam periode milidetik.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil perancangan dan pengujian yang telah dilakukan selama penelitian ini, dapat disimpulkan hal-hal berikut:

1. Perangkat keras yang terdiri dari FPGA Nexys3, sensor kamera OV7670 dan monitor VGA mampu bekerja sesuai spesifikasi perancangan sistem. Hal ini terbukti dari hasil implementasi sistem yang menunjukkan skematik blok rangkaian yang sesuai dengan desain diagram blok sistem dan penggunaan *resource* yang sesuai dengan kebutuhan unit memori dan rangkaian logika. Selain itu hasil pengujian perangkat keras menunjukkan sensor kamera OV7670 telah bekerja dengan *setting* kamera 640 × 480 piksel YUV dan *frame rate* 15 fps.
2. Sistem ANPR berhasil diimplementasikan baik pada simulasi program maupun pengujian langsung dengan hasil pengujian yang sejalan. Baik hasil dari simulasi maupun pengujian menunjukkan sistem telah berhasil menerapkan metode yang dipakai dengan benar pada ketiga tahap ANPR.
3. Sistem dapat mengenali 82% dari total karakter yang diuji dengan waktu eksekusi terlama hingga 2,3 ms dan kesalahan maksimum 3 karakter per plat nomor, akan tetapi tingkat keberhasilan identifikasi plat nomor hanya 19%. Tingkat keberhasilan yang kecil untuk identifikasi plat nomor disebabkan karena pada tahap pengenalan karakter optis sistem masih kesulitan membedakan karakter yang mirip.

5.2 Saran

Saran untuk penelitian selanjutnya adalah sebagai berikut:

1. Menggunakan FPGA yang memiliki spesifikasi lebih tinggi dan *resource* yang lebih banyak dari tipe Spartan-6 sehingga perancangan sistem dapat ditingkatkan.
2. Pengujian menggunakan sejumlah sampel plat nomor dengan kemunculan setiap jenis karakter pada keseluruhan sampel plat nomor minimal 5 karakter untuk mengetahui tingkat akurasi sistem lebih lanjut.

DAFTAR PUSTAKA

Digilent. 2011. *Nexys3™ Board Reference Manual.* Pullman, Washington: Digilent. http://www.digilentinc.com/Data/Products/NEXYS3/Nexys3_rm.pdf (diakses 14 Januari 2016).

Electrodragon. 2013. *OV7670 Module.* Newark, New Jersey: Electrodragon. http://www.electrodragon.com/w/OV7670_Module (diakses 14 Januari 2016).

Gonzalez, R. C. & Woods, R. E. 2008. *Digital Image Processing.* Upper Saddle River, New Jersey: Pearson Education.

Hermawati, F. A. 2010. A Real-Time License Plate Detection System for Parking Access. *Telkomnika*. 8(2):97-106.

Lim, R. 2003. Sistem Pengenalan Plat Nomor Mobil dengan Metode Principal Component Analysis. *Jurnal Teknik Elektro*. 3(1):31-38.

Martinsky, O. 2007. *Algorithmic and Mathematical Principles of Automatic Number Plate Recognition Systems.* Brno: Brno University of Technology.

Mellolo, O. 2012. Pengenalan Plat Nomor Polisi Kendaraan Bermotor. *Jurnal Ilmiah Sains*. 12(1):35-42.

OmniVision Technologies. 2003. *OmniVision Serial Camera Control Bus (SCCB) Functional Specification.* Santa Clara, California: OmniVision Technologies. http://www.ovt.com/download_document.php?type=document&DID=63 (diakses 14 Januari 2016).

OmniVision Technologies. 2005. *OV7670/OV7171 CMOS VGA (640x480) CameraChip™ Implementation Guide.* Santa Clara, California: OmniVision Technologies. http://web.mit.edu/6.111/www/f2015/tools/OV7670app.pdf (diakses 14 Januari 2016).

OmniVision Technologies. 2006. *OV7670/OV7171CMOS VGA (640x480) CameraChip™ Sensor with OmniPixel® Technology.* Santa Clara, California: OmniVision Technologies. http://www.voti.nl/docs/OV7670.pdf (diakses 14 Januari 2016).

Peterson, L. E. 2009. K-nearest neighbor. *Scholarpedia*. 4(2):1883.

Pratt, W. K. 2007. *Digital Image Processing.* Hoboken, New Jersey: John Wiley & Sons.

SECONS. 2008. *VESA Signal 800 x 600 @ 72 Hz timing.* Prague: SECONS. http://tinyvga.com/vga-timing/800x600@72Hz (diakses 14 Januari 2016).

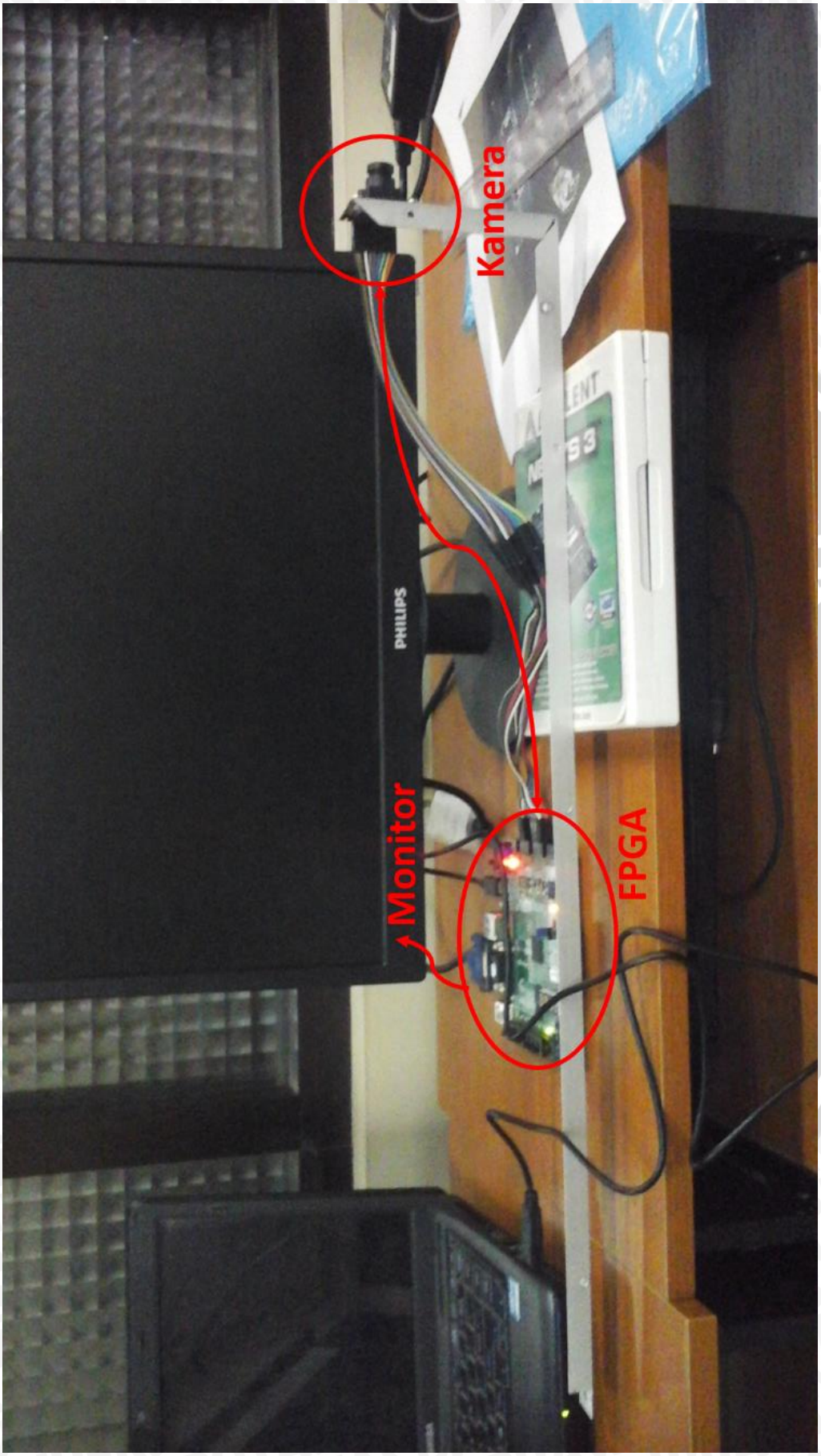
Xilinx. 2011. *Spartan-6 Family Overview.* San Jose, California: Xilinx. http://www.xilinx.com/support/documentation/data_sheets/ds160.pdf (diakses 14 Januari 2016).

Xilinx. 2011. *Spartan-6 FPGA Configurable Logic Blocks User Guide.* San Jose, California: Xilinx. http://www.xilinx.com/support/documentation/user_guides/ug384.pdf (diakses 14 Januari 2016).

Zhai, X. 2013. *Automatic Number Plate Recognition on FPGA.* Hertfordshire: Science and Technology Research Institute, University of Hertfordshire.

LAMPIRAN

Lampiran 1 Foto Alat Pengujian

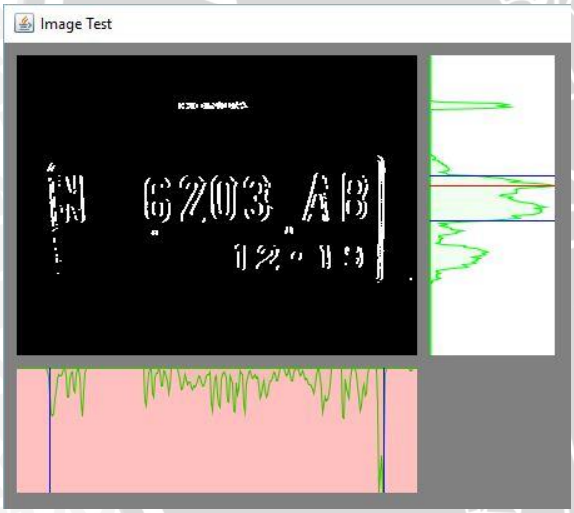


Lampiran 2 Tampilan Simulasi Program

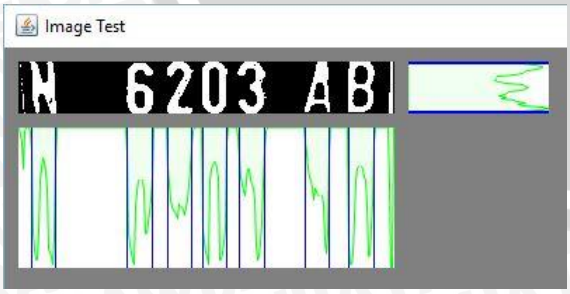
1. Sampel Plat Nomor N 6203 AB

Citra awal

Hasil lokalisasi plat nomor



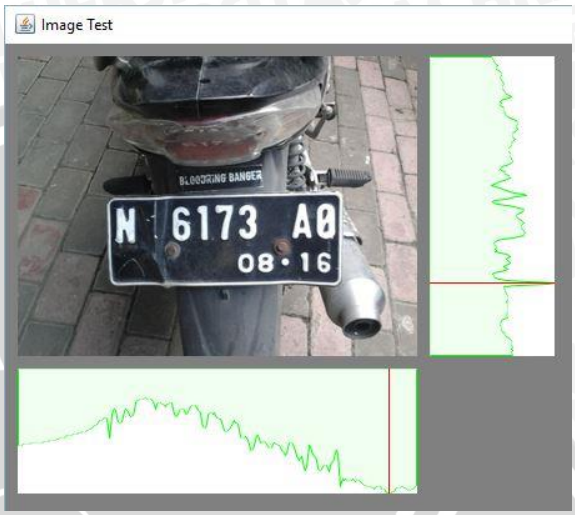
Hasil segmentasi karakter



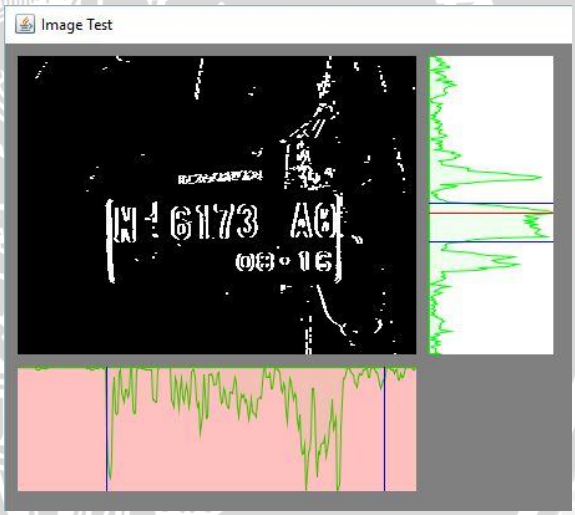
Hasil pengenalan karakter optis

2. Sampel Plat Nomor N 6173 AO

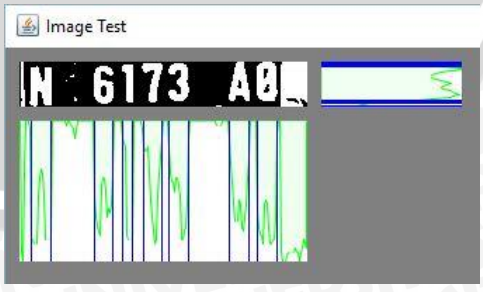
Citra awal



Hasil lokalisasi plat nomor



Hasil segmentasi karakter

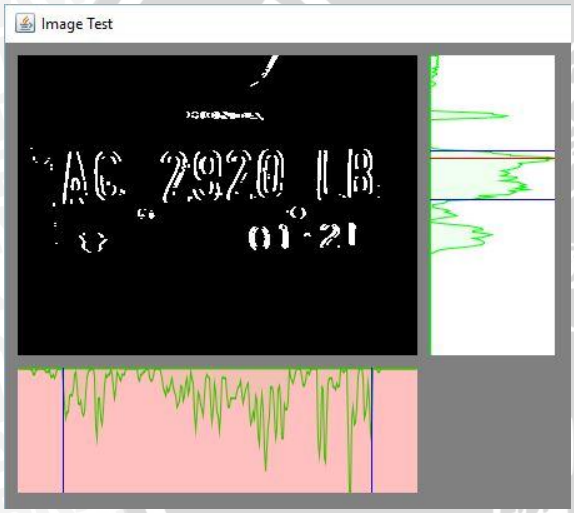


Hasil pengenalan karakter optis

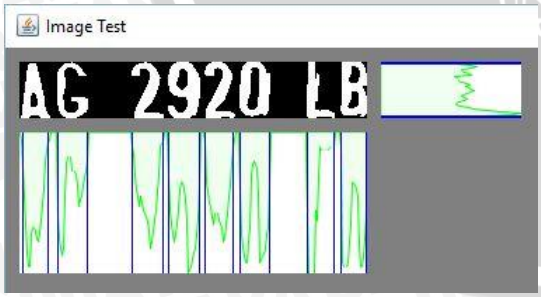
3. Sampel Plat Nomor AG 2920 LB

Citra awal

Hasil lokalisasi plat nomor



Hasil segmentasi karakter



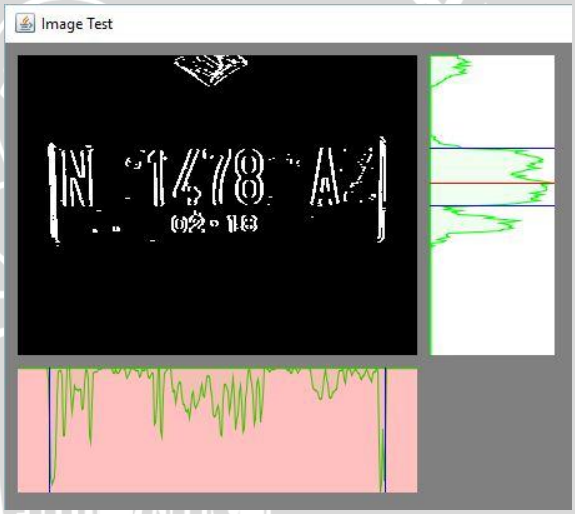
Hasil pengenalan karakter optis

AC292QL8
waktu eksekusi 17 ms

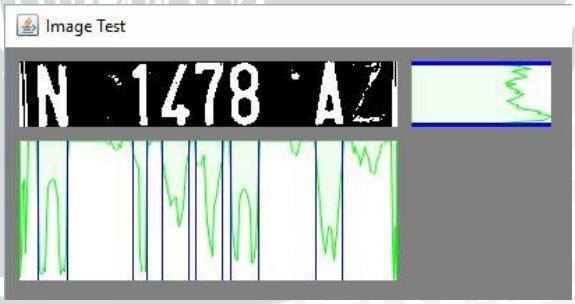
4. Sampel Plat Nomor N 1478 AZ

Citra awal

Hasil lokalisasi plat nomor



Hasil segmentasi karakter



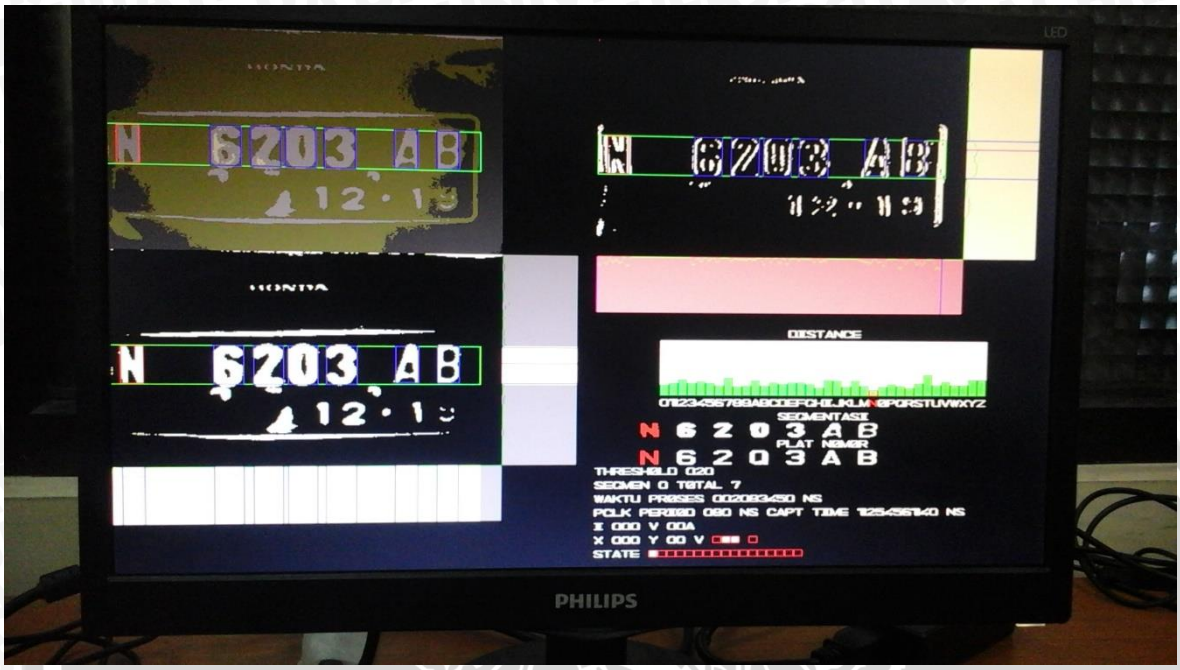
Hasil pengenalan karakter optis

N1478A
waktu eksekusi 15 ms

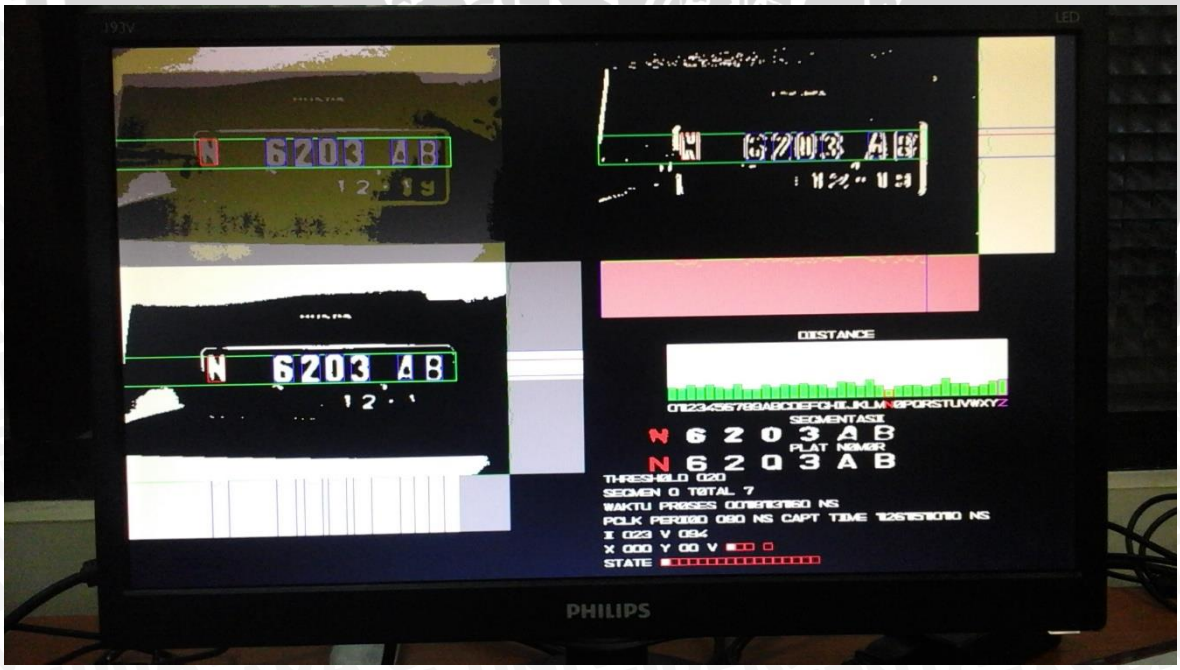
Lampiran 3 Pengujian Langsung

1. Tampilan Monitor Sampel Plat N 6203 AB

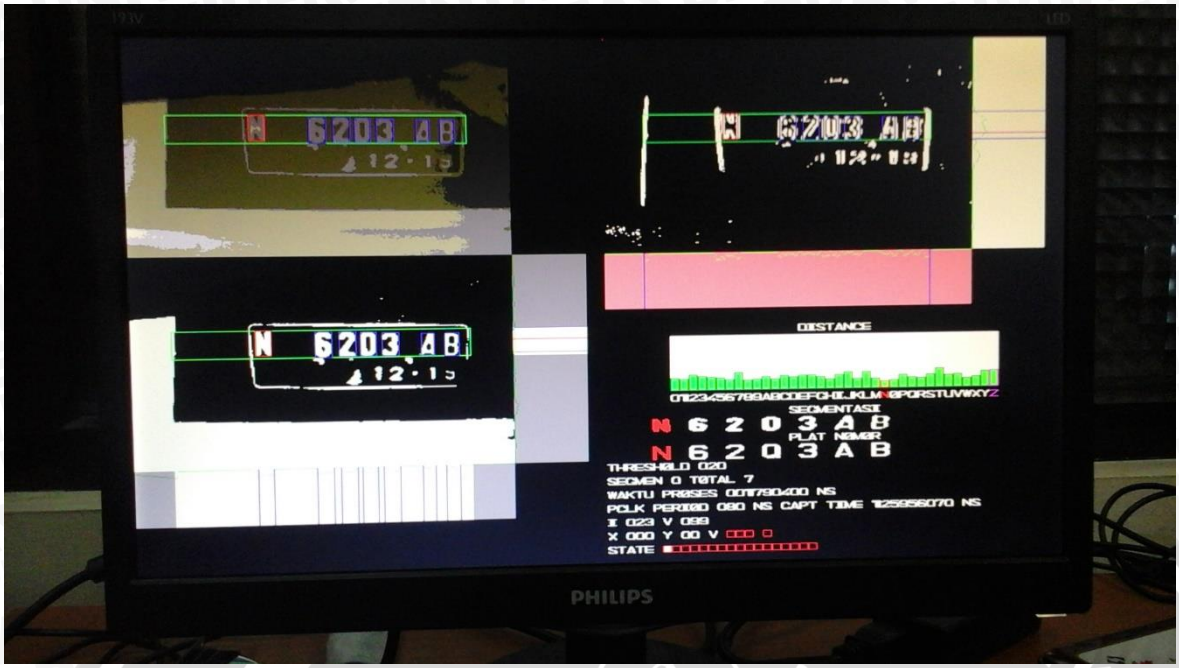
Jarak 30 cm



Jarak 40 cm



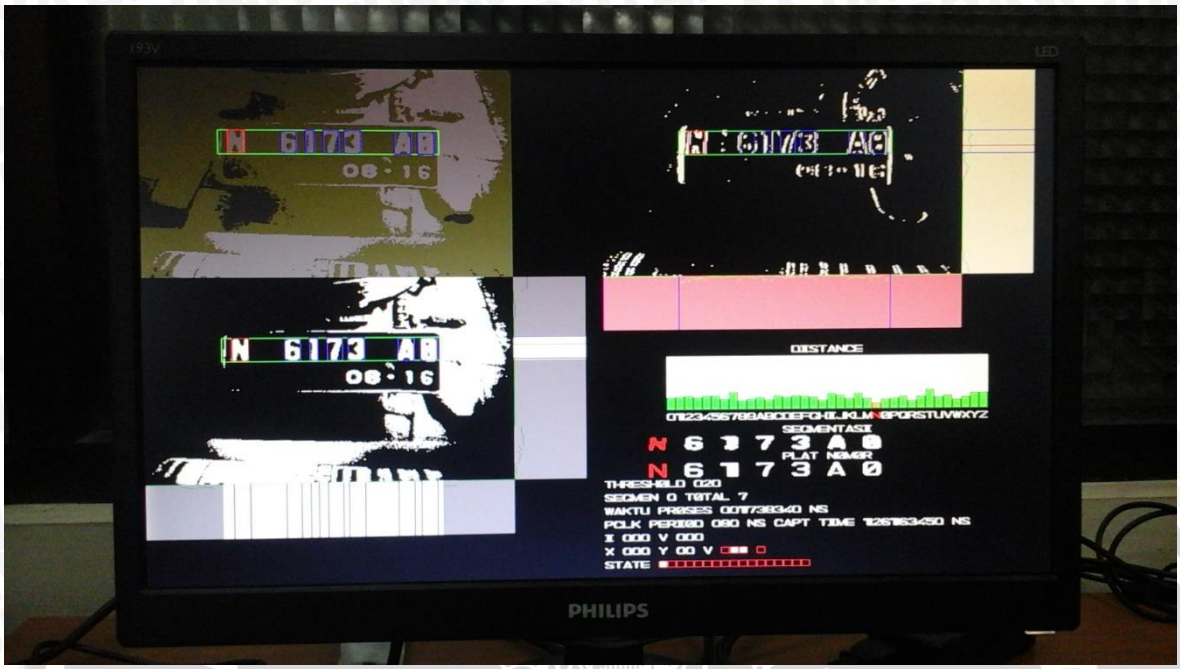
Jarak 50 cm



Jarak 60 cm

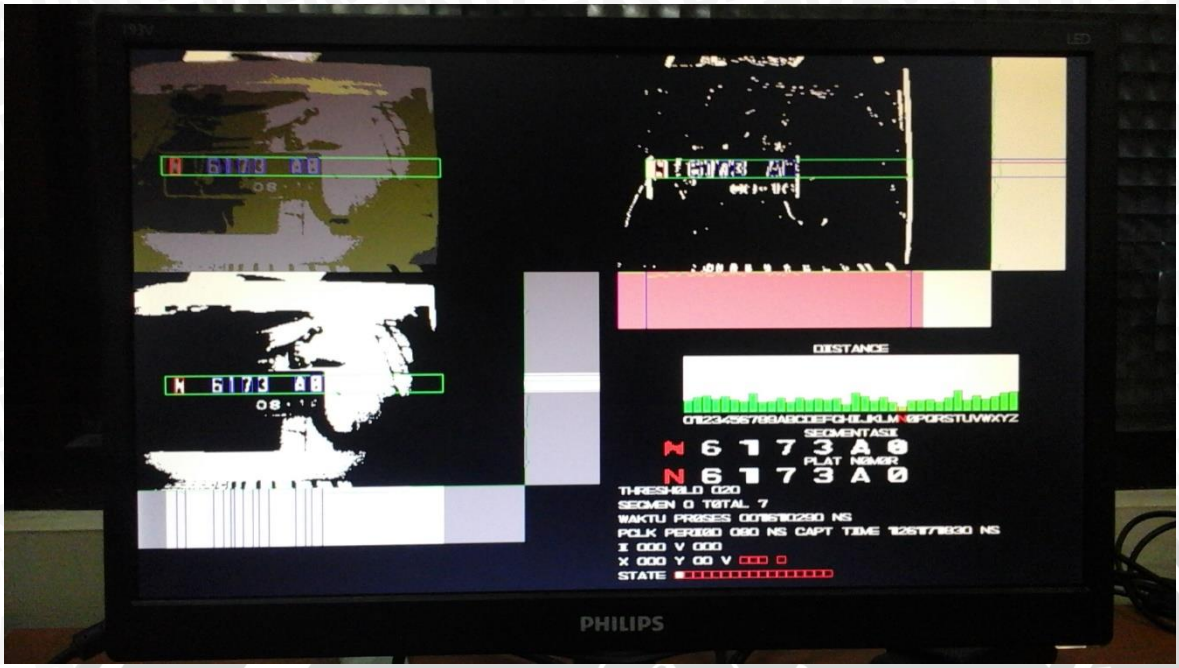
2. Tampilan Monitor Sampel Plat N 6173 AO

Jarak 30 cm

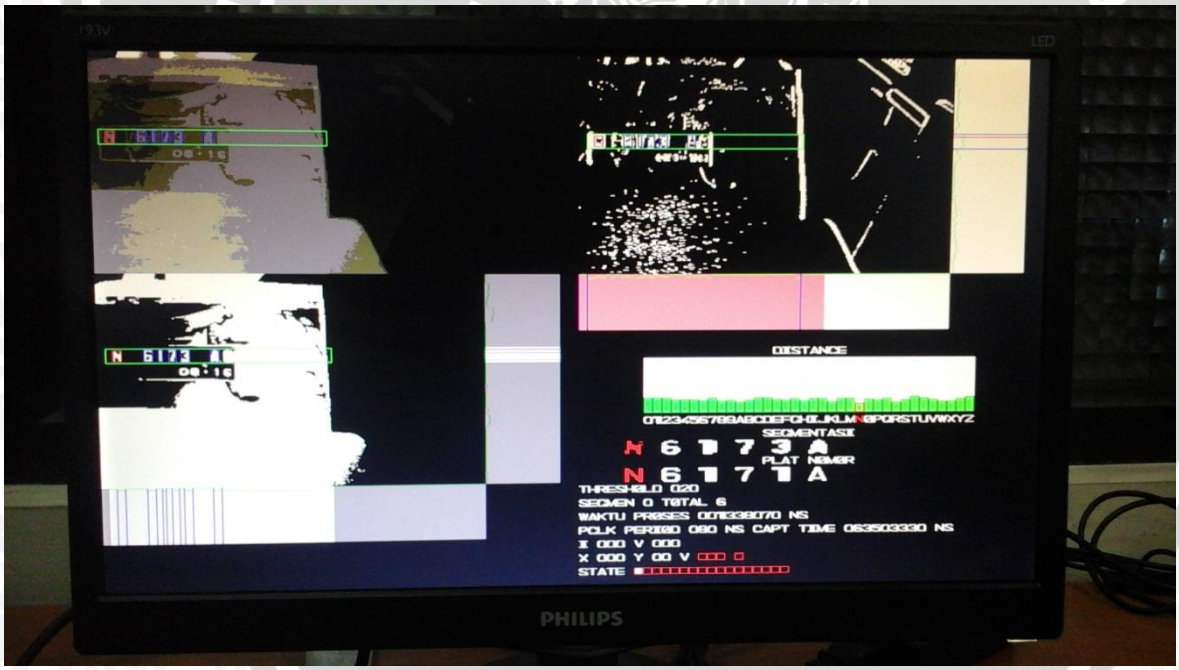


Jarak 40 cm

Jarak 50 cm

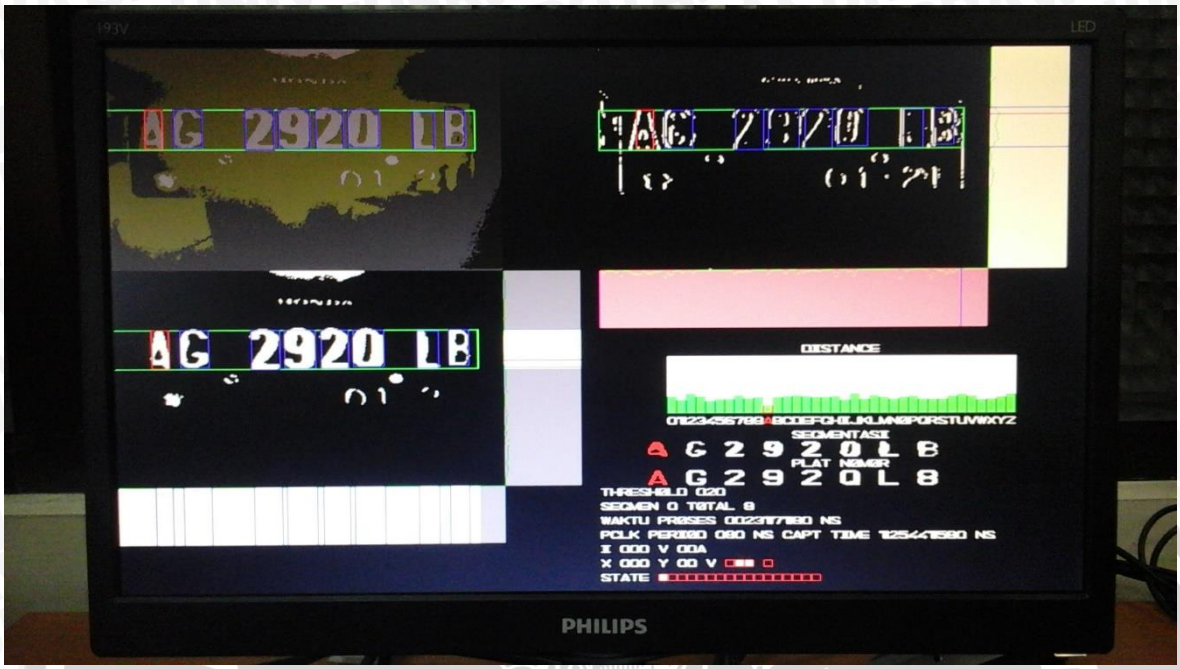


Jarak 60 cm



3. Tampilan Monitor Sampel Plat AG 2920 LB

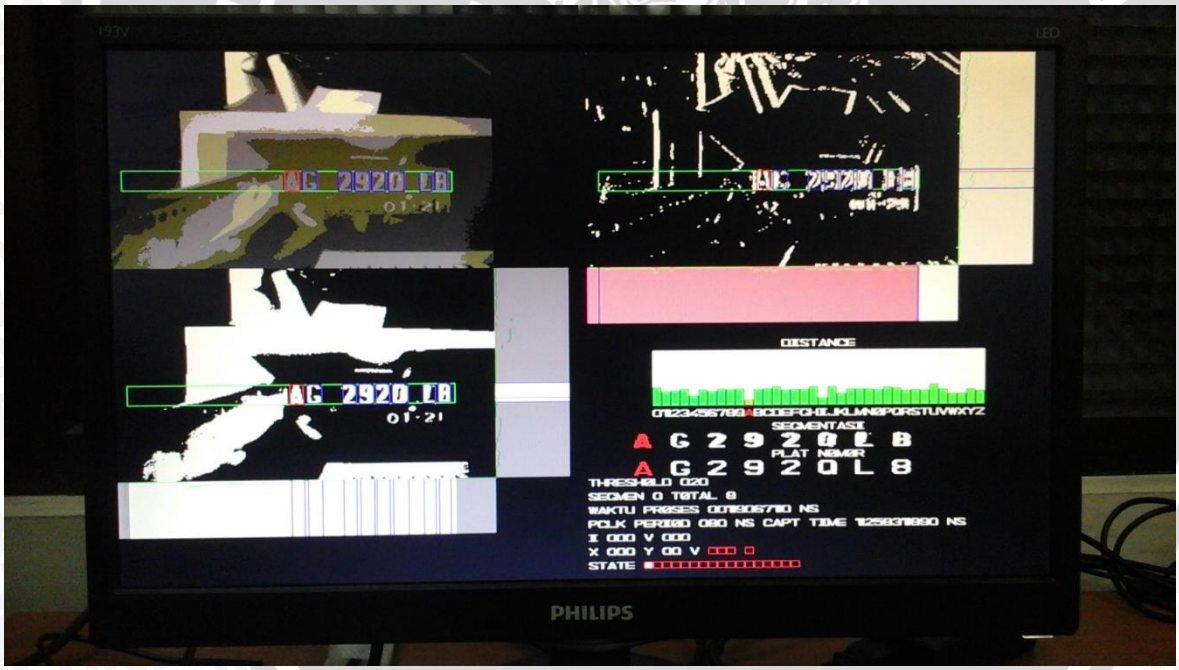
Jarak 30 cm



Jarak 40 cm

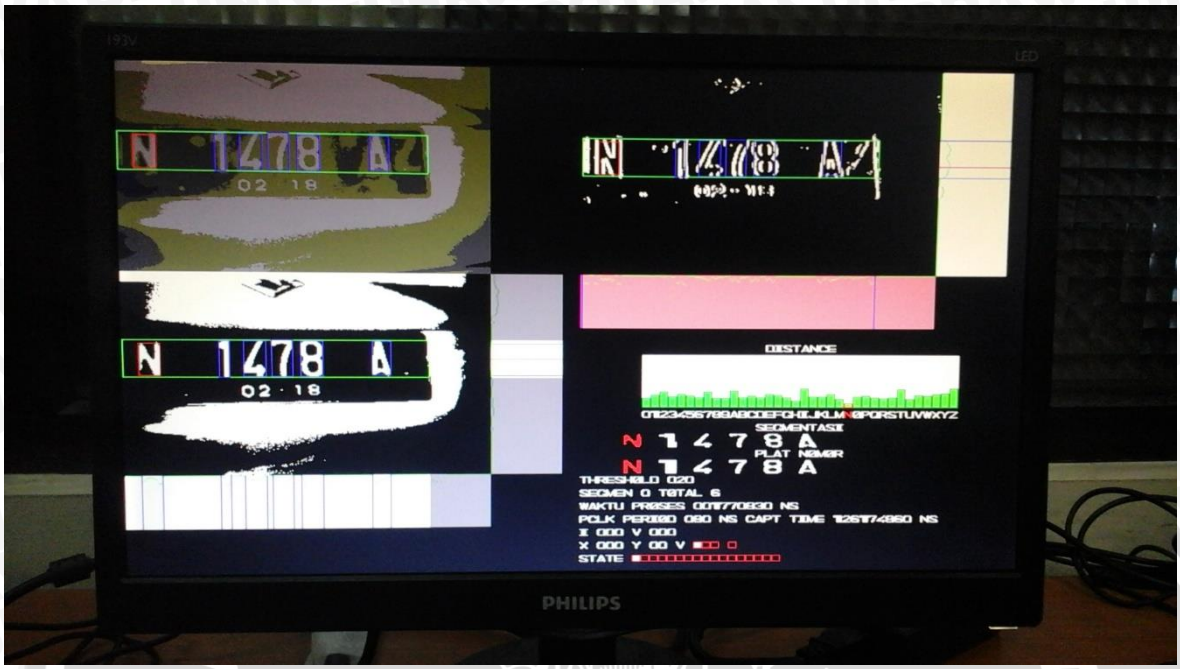
Jarak 50 cm

Jarak 60 cm

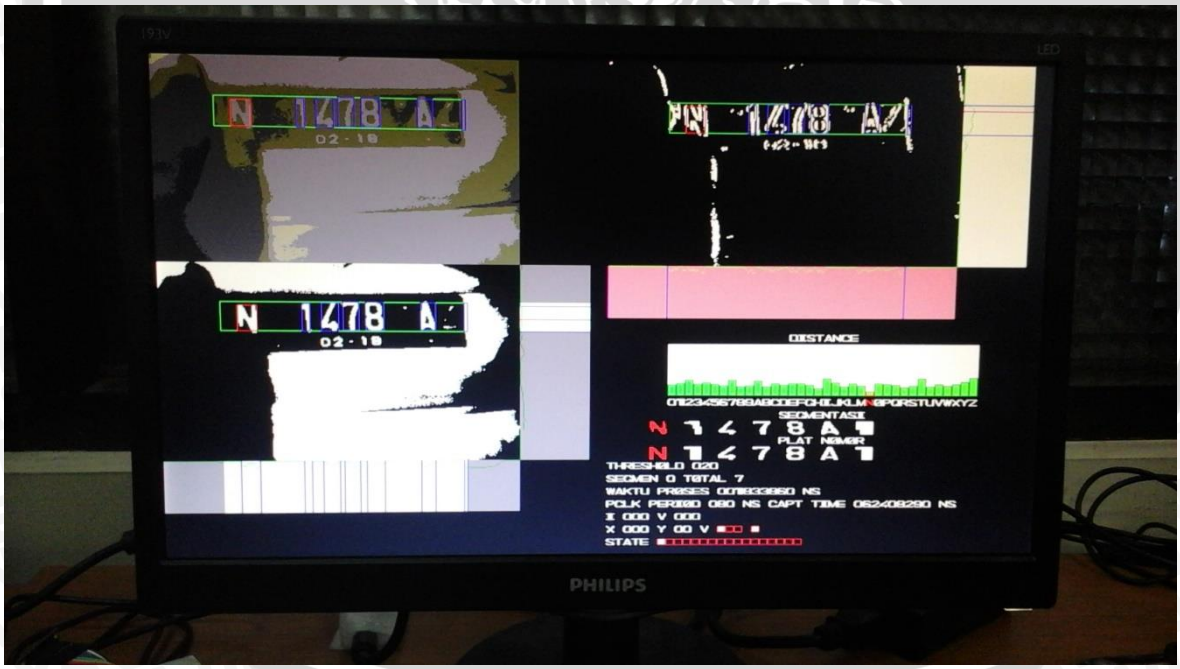


4. Tampilan Monitor Sampel Plat N 1478 AZ

Jarak 30 cm



Jarak 40 cm

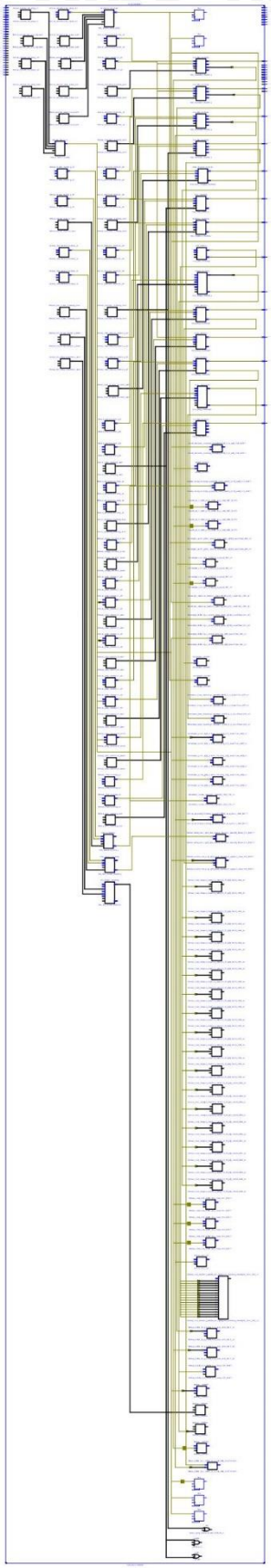


Jarak 50 cm

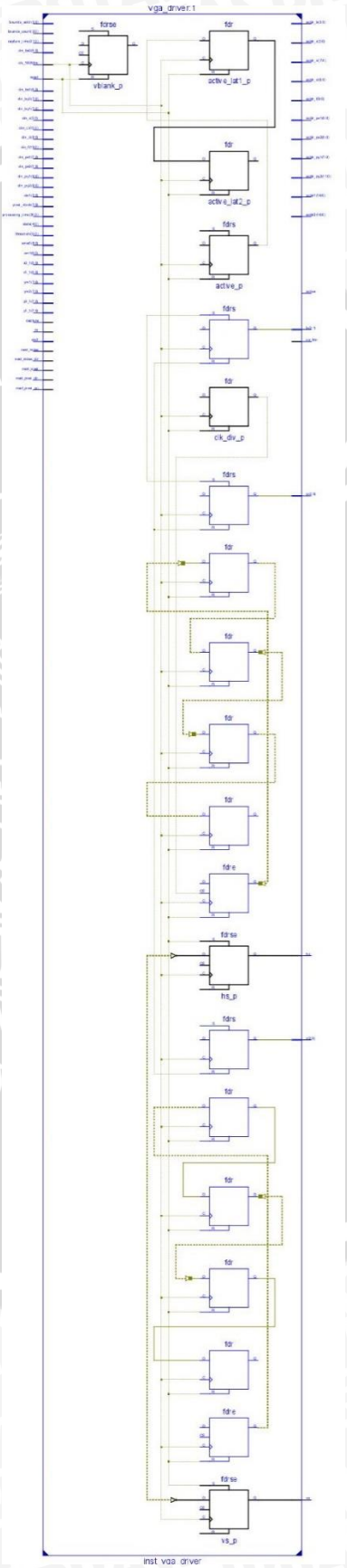
Jarak 60 cm

Lampiran 4 Rangkaian Skematik

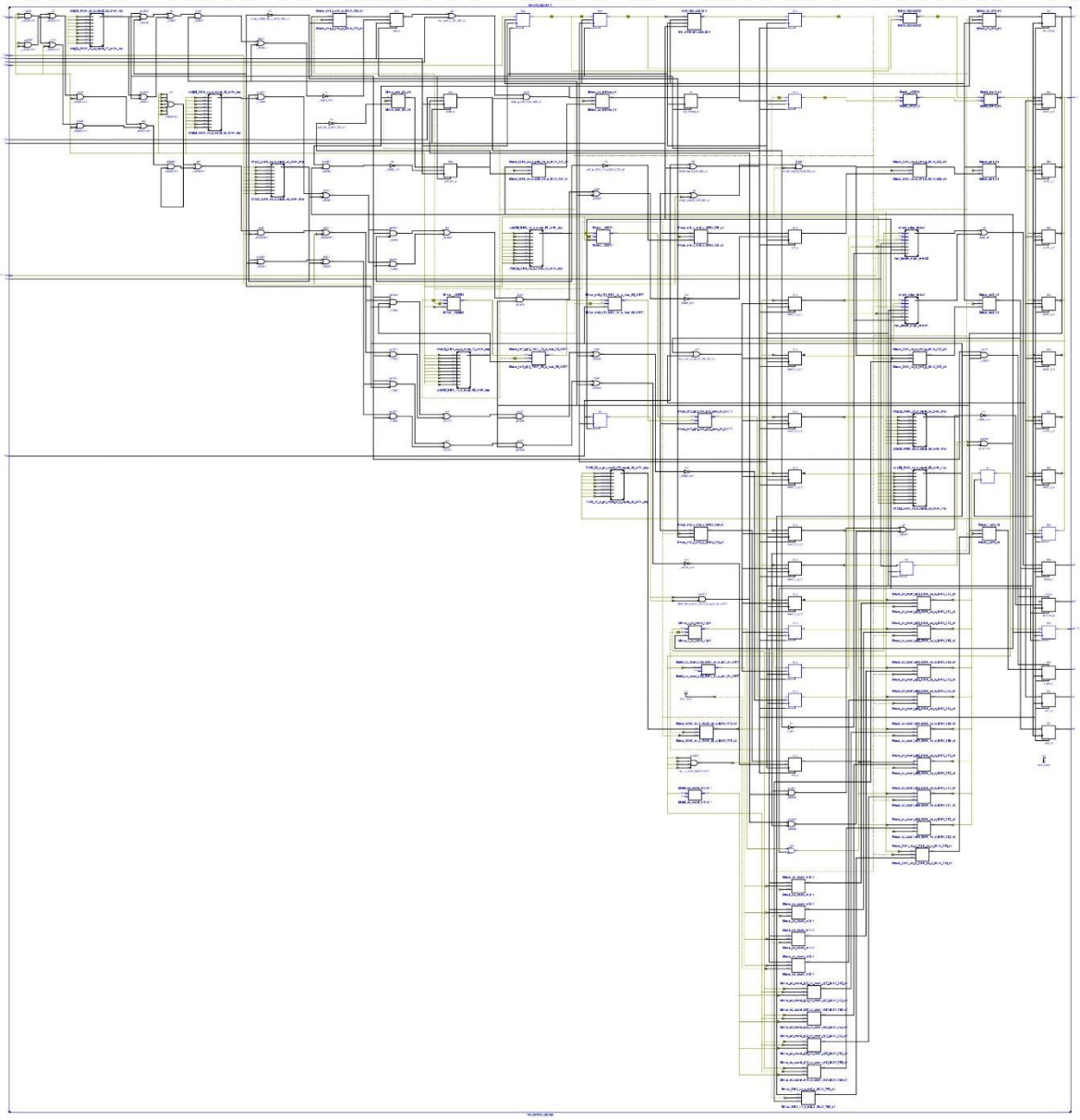
1. Blok Sistem ANPR



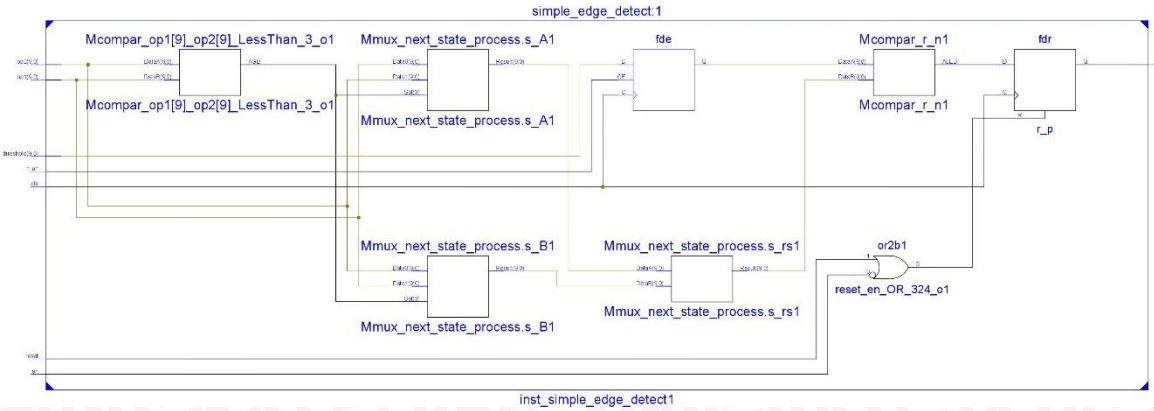
2. Blok VGA *Driver*



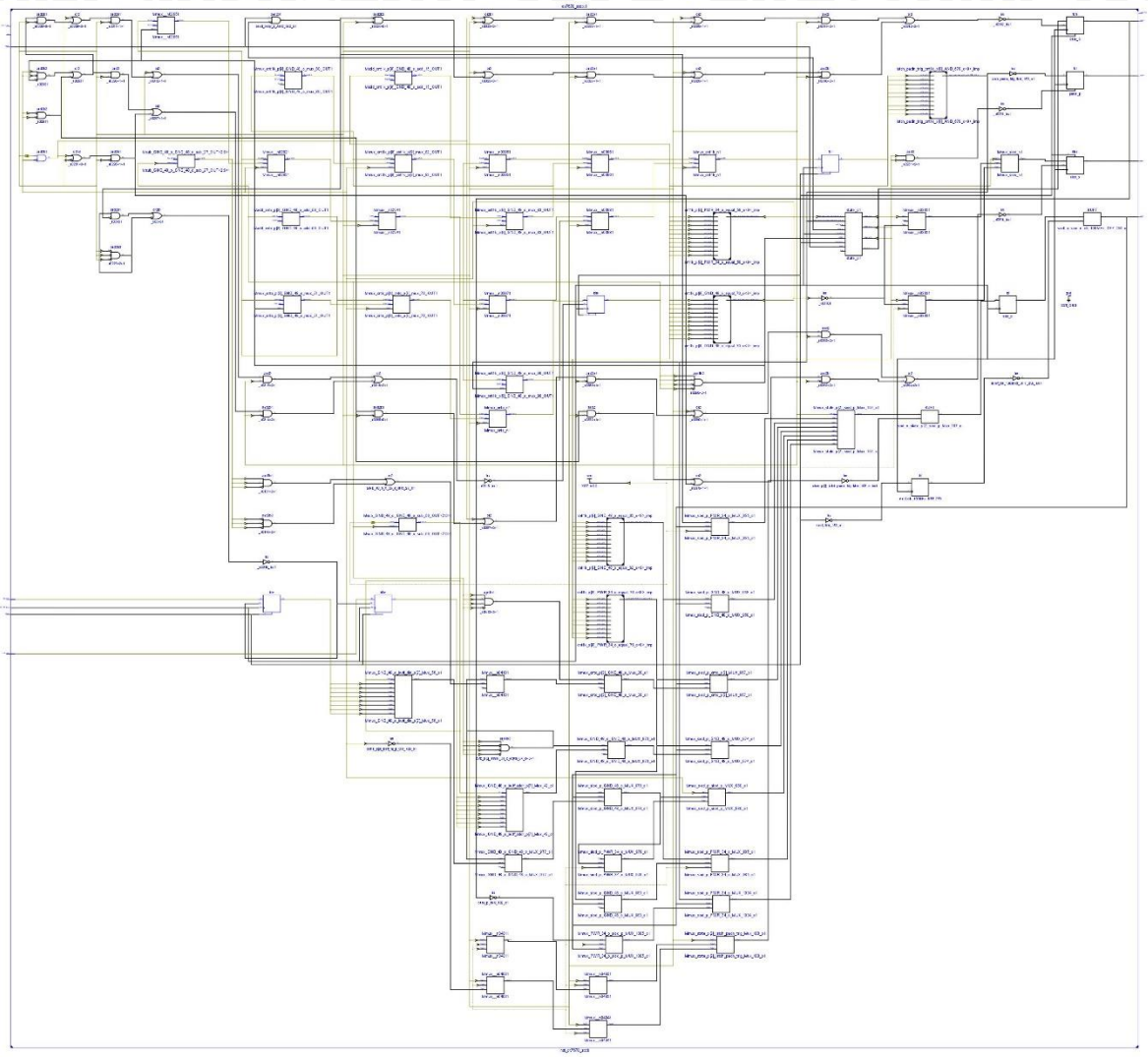
3. Blok Antarmuka Kamera-Memori



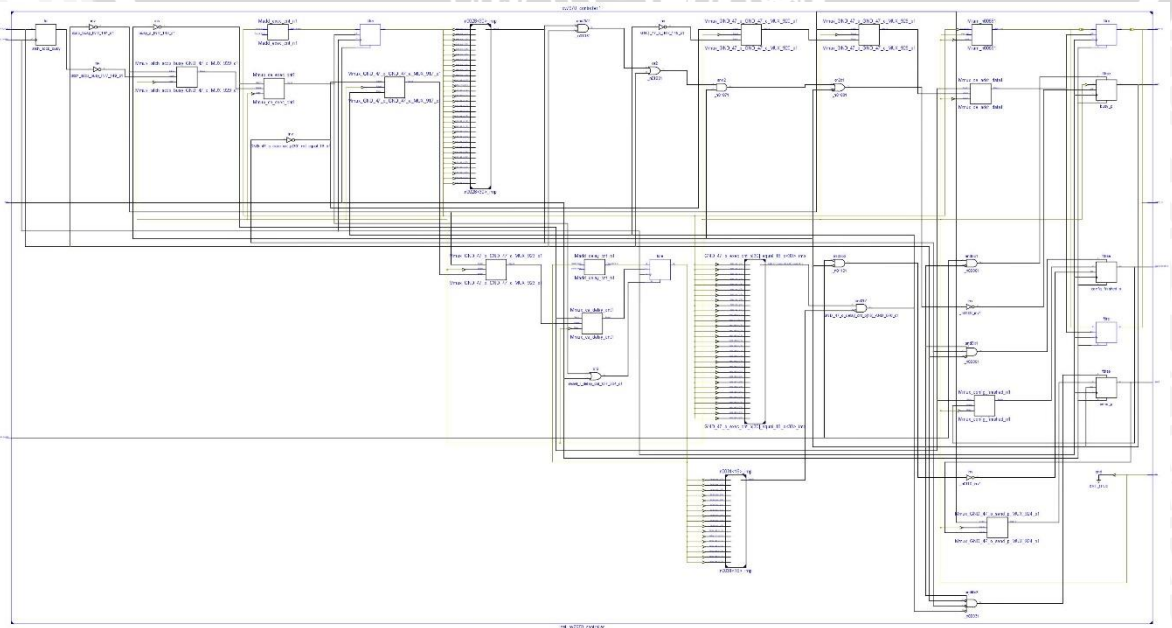
4. Blok Deteksi Tepi Vertikal



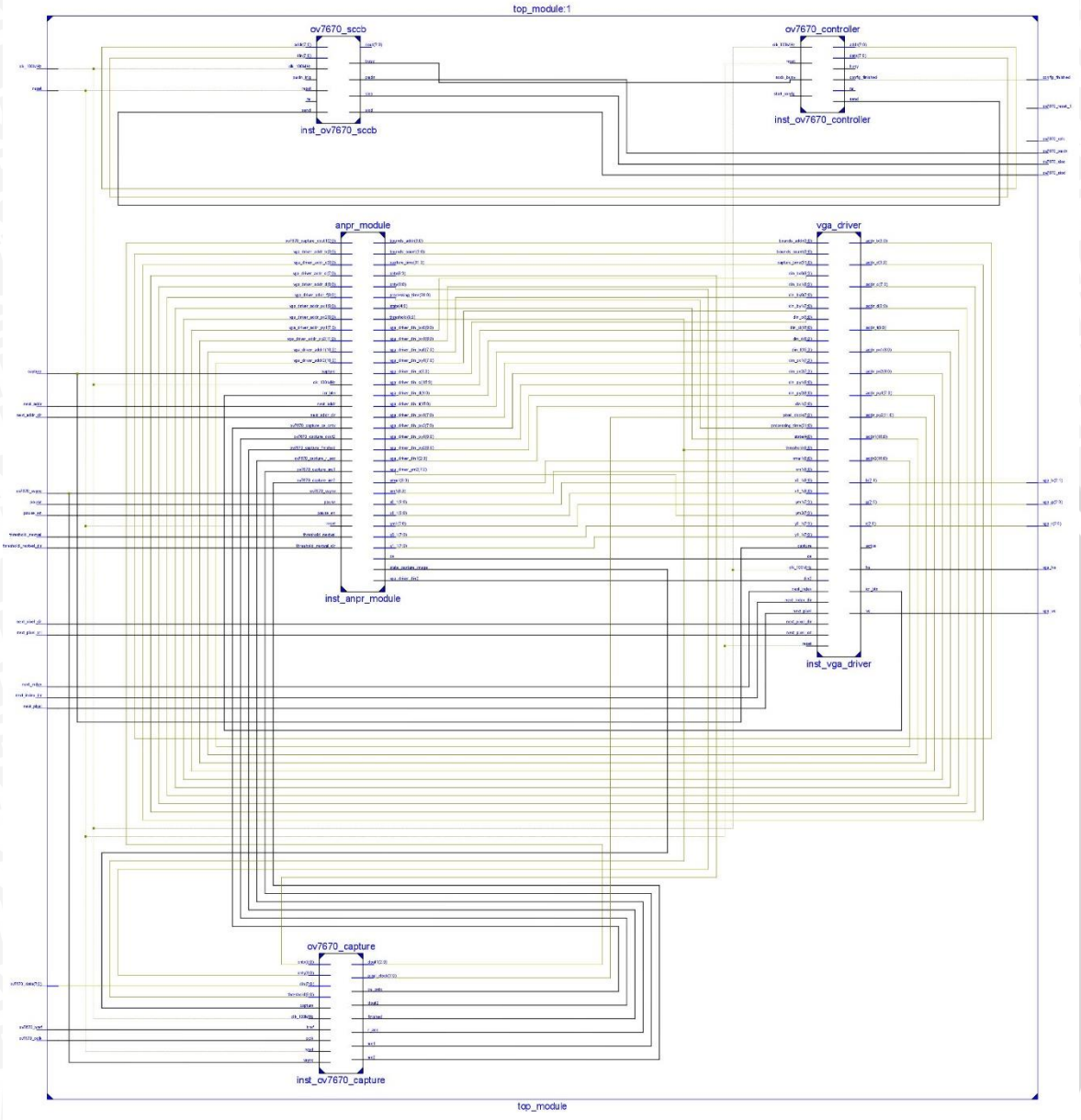
5. Blok Antarmuka SCCB



6. Blok Kontroler Kamera



7. Blok Sistem Keseluruhan



Lampiran 5 Laporan Hasil Implementasi Sistem

1. Ringkasan Perancangan Sistem

top_module Project Status (06/15/2016 - 04:49:38)			
Project File:	ov7670_nexys3.xise	Parser Errors:	No Errors
Module Name:	top_module	Implementation State:	Programming File Generated
Target Device:	xc6slx16-3csg324	Errors:	
Product Version:	ISE 14.7	Warnings:	
Design Goal:	Balanced	Routing Results:	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	Timing Constraints:	All Constraints Met
Environment:	System Settings	Final Timing Score:	0 (Timing Report)

Device Utilization Summary [-]				
Slice Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Registers	1,491	18,224	8%	
Number used as Flip Flops	1,487			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	4			
Number of Slice LUTs	4,831	9,112	53%	
Number used as logic	4,066	9,112	44%	
Number using O6 output only	3,172			
Number using O5 output only	244			
Number using O5 and O6	650			
Number used as ROM	0			
Number used as Memory	705	2,176	32%	
Number used as Dual Port RAM	472			
Number using O6 output only	412			
Number using O5 output only	0			
Number using O5 and O6	60			
Number used as Single Port RAM	73			
Number using O6 output only	69			
Number using O5 output only	0			
Number using O5 and O6	4			
Number used as Shift Register	160			
Number using O6 output only	160			
Number using O5 output only	0			
Number using O5 and O6	0			
Number used exclusively as route-thrus	60			
Number with same-slice register load	21			
Number with same-slice carry load	39			
Number with other load	0			
Number of occupied Slices	1,549	2,278	67%	
Number of MUXCYs used	1,056	4,556	23%	
Number of LUT Flip Flop pairs used	5,017			
Number with an unused Flip Flop	3,625	5,017	72%	

Number with an unused LUT	186	5,017	3%	
Number of fully used LUT-FF pairs	1,206	5,017	24%	
Number of unique control sets	107			
Number of slice register sites lost to control set restrictions	424	18,224	2%	
Number of bonded IOBs	41	232	17%	
Number of LOCed IOBs	41	41	100%	
Number of RAMB16BWERs	20	32	62%	
Number of RAMB8BWERs	4	64	6%	
Number of BUFIO2/BUFIO2_2CLKs	0	32	0%	
Number of BUFIO2FB/BUFIO2FB_2CLKs	0	32	0%	
Number of BUFG/BUFGMUXs	2	16	12%	
Number used as BUFGs	2			
Number used as BUFGMUX	0			
Number of DCM/DCM_CLKGENs	0	4	0%	
Number of ILOGIC2/ISERDES2s	0	248	0%	
Number of IODELAY2/IODRP2/IODRP2_MCBs	0	248	0%	
Number of OLOGIC2/OSERDES2s	0	248	0%	
Number of BSCANs	0	4	0%	
Number of BUFHs	0	128	0%	
Number of BUFPLLs	0	8	0%	
Number of BUFPLL_MCBs	0	4	0%	
Number of DSP48A1s	6	32	18%	
Number of ICAPs	0	1	0%	
Number of MCBs	0	2	0%	
Number of PCILOGICSEs	0	2	0%	
Number of PLL_ADVs	0	2	0%	
Number of PMVs	0	1	0%	
Number of STARTUPs	0	1	0%	
Number of SUSPEND_SYNCs	0	1	0%	
Average Fanout of Non-Clock Nets	4.83			

Performance Summary			[-]
Final Timing Score:	0 (Setup: 0, Hold: 0, Component Switching Limit: 0)	Pinout Data:	Pinout Report
Routing Results:	All Signals Completely Routed	Clock Data:	Clock Report
Timing Constraints:	All Constraints Met		

Detailed Reports					[-]
Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Wed Jun 15 08:26:49			
Translation Report	Current	Wed Jun 15 08:27:16	0	0	0
Map Report	Current	Wed Jun 15 08:30:14	0	3 Warnings (0 new)	98 Infos (92 new)
Place and Route Report	Current	Wed Jun 15 08:31:55	0	0	0
Power Report					
Post-PAR Static Timing	Current	Wed Jun 15 08:32:27	0	0	3 Infos (0 new)
Bitgen Report	Current	Wed Jun 15 08:33:32	0	2 Warnings (0 new)	1 Info (0 new)

Secondary Reports		[-]
Report Name	Status	Generated
ISIM Simulator Log	Current	Wed Jul 20 00:51:22 2016
WebTalk Report	Current	Wed Jun 15 08:33:36 2016
WebTalk Log File	Current	Wed Jun 15 08:33:37 2016

Date Generated: 07/31/2016 - 02:35:16

2. Ringkasan Laporan *Timing*

```
Timing summary:
 ---------------

 Timing errors: 0  Score: 0  (Setup/Max: 0, Hold: 0)

 Constraints cover 779187 paths, 0 nets, and 25763 connections

 Design statistics:
    Minimum period:   9.584ns{1}   (Maximum frequency: 104.341MHz)
```

3. Laporan Sintesis HDL

```
Synthesizing Unit <top_module>.
    Summary:
     inferred   2 D-type flip-flop(s).
Unit <top_module> synthesized.

Synthesizing Unit <anpr_module>.
    Summary:
     inferred   8 Multiplier(s).
     inferred  71 Adder/Subtractor(s).
     inferred 637 D-type flip-flop(s).
     inferred  36 Comparator(s).
     inferred 720 Multiplexer(s).
     inferred   1 Finite State Machine(s).
Unit <anpr_module> synthesized.

Synthesizing Unit <ov7670_capture>.
    Summary:
     inferred   5 Adder/Subtractor(s).
     inferred  96 D-type flip-flop(s).
     inferred  37 Multiplexer(s).
Unit <ov7670_capture> synthesized.

Synthesizing Unit <simple_edge_detect>.
    Summary:
     inferred   1 Adder/Subtractor(s).
     inferred  11 D-type flip-flop(s).
     inferred   2 Comparator(s).
     inferred   2 Multiplexer(s).
Unit <simple_edge_detect> synthesized.

Synthesizing Unit <ov7670_controller>.
    Summary:
     inferred   1 RAM(s).
     inferred   2 Adder/Subtractor(s).
     inferred  68 D-type flip-flop(s).
     inferred  10 Multiplexer(s).
Unit <ov7670_controller> synthesized.

Synthesizing Unit <ov7670_sccb>.
    Summary:
     inferred   4 Adder/Subtractor(s).
     inferred  52 D-type flip-flop(s).
     inferred  41 Multiplexer(s).
     inferred   2 Tristate(s).
     inferred   1 Finite State Machine(s).
Unit <ov7670_sccb> synthesized.

Synthesizing Unit <vga_driver>.
    Summary:
     inferred   1 RAM(s).
     inferred   4 Multiplier(s).
     inferred 169 Adder/Subtractor(s).
     inferred 422 D-type flip-flop(s).
     inferred 153 Comparator(s).
     inferred 348 Multiplexer(s).
Unit <vga_driver> synthesized.
=========================================
HDL Synthesis Report

Macro Statistics
# RAMs                                : 2
 32x5-bit single-port Read Only RAM  : 1
 64x16-bit single-port Read Only RAM
                                     : 1
# Multipliers                         : 12
 10x9-bit multiplier                 : 1
 11x10-bit multiplier                : 1
 16x2-bit multiplier                 : 1
 6x4-bit multiplier                  : 1
 8x3-bit multiplier                  : 1
 8x4-bit multiplier                  : 1
 8x5-bit multiplier                  : 1
 8x8-bit multiplier                  : 1
 9x8-bit multiplier                  : 4
# Adders/Subtractors                  : 253
 10-bit adder                        : 7
 10-bit addsub                       : 1
 10-bit subtractor                   : 6
 11-bit adder                        : 3
 11-bit subtractor                   : 18
 12-bit adder                        : 113
 12-bit subtractor                   : 14
 13-bit adder                        : 2
 14-bit adder                        : 1
 16-bit adder                        : 1
 17-bit adder                        : 10
 18-bit adder                        : 2
 19-bit adder                        : 3
 2-bit subtractor                    : 1
 20-bit adder                        : 1
 21-bit adder                        : 1
 26-bit adder                        : 3
 3-bit adder                         : 1
 3-bit subtractor                    : 4
 31-bit adder                        : 1
 4-bit adder                                : 16
 4-bit addsub                        : 1
 4-bit subtractor                    : 1
 6-bit adder                         : 2
 6-bit subtractor                    : 1
```

```
 7-bit adder                       : 2
 8-bit adder                       : 6
 8-bit addsub                      : 2
 8-bit subtractor                  : 6
 9-bit adder                       : 14
 9-bit addsub                      : 2
 9-bit subtractor                  : 7
# Registers                        : 160
 1-bit register                    : 36
 10-bit register                   : 15
 11-bit register                   : 6
 12-bit register                   : 5
 14-bit register                   : 2
 16-bit register                   : 2
 17-bit register                   : 9
 19-bit register                   : 1
 2-bit register                    : 4
 26-bit register                   : 3
 3-bit register                    : 2
 31-bit register                   : 1
 32-bit register                   : 3
 4-bit register                    : 11
 6-bit register                    : 5
 7-bit register                    : 1
 8-bit register                    : 31
 9-bit register                    : 23
# Comparators                      : 193
 10-bit comparator equal           : 22
 10-bit comparator greater         : 35
 10-bit comparator lessequal       : 35
 11-bit comparator equal           : 16
 11-bit comparator greater         : 14
 11-bit comparator lessequal       : 15
 12-bit comparator equal           : 1
 12-bit comparator greater         : 1
 12-bit comparator lessequal       : 2
 17-bit comparator greater         : 3
 17-bit comparator lessequal       : 1
 17-bit comparator not equal       : 1
 4-bit comparator equal            : 4
 4-bit comparator greater          : 1
 6-bit comparator equal            : 1
 6-bit comparator greater          : 1
 6-bit comparator not equal        : 1
 8-bit comparator equal            : 4
 8-bit comparator greater          : 15
 8-bit comparator not equal        : 2
 9-bit comparator equal            : 9
 9-bit comparator greater          : 5
 9-bit comparator lessequal        : 2
 9-bit comparator not equal        : 2
# Multiplexers                     : 1160
 1-bit 14-to-1 multiplexer         : 1
 1-bit 15-to-1 multiplexer         : 1
 1-bit 16-to-1 multiplexer         : 7
 1-bit 2-to-1 multiplexer          : 578
 1-bit 4-to-1 multiplexer          : 1
 1-bit 7-to-1 multiplexer          : 1
 1-bit 8-to-1 multiplexer          : 2
 10-bit 2-to-1 multiplexer         : 138
 10-bit 8-to-1 multiplexer         : 1
 11-bit 2-to-1 multiplexer         : 1
 12-bit 2-to-1 multiplexer         : 18
 16-bit 2-to-1 multiplexer         : 1
 17-bit 2-to-1 multiplexer         : 28
 19-bit 2-to-1 multiplexer         : 5
 2-bit 2-to-1 multiplexer          : 64
 3-bit 2-to-1 multiplexer          : 114
 4-bit 2-to-1 multiplexer          : 22
 6-bit 2-to-1 multiplexer          : 29
 8-bit 2-to-1 multiplexer          : 37
 9-bit 2-to-1 multiplexer          : 111
# Tristates                        : 2
 1-bit tristate buffer             : 2
# FSMs                             : 2
# Xors                             : 1
 1-bit xor2                        : 1

=========================================

=========================================
Advanced HDL Synthesis Report

Macro Statistics
# RAMs                             : 2
 32x5-bit single-port distributed Read Only
RAM                                : 1
 64x16-bit single-port distributed Read Only
RAM                                : 1
# MACs                             : 3
 11x10-to-17-bit MAC               : 1
 6x4-to-9-bit MAC                  : 1
 8x4-to-12-bit MAC                 : 1
# Multipliers                      : 9
 10x9-bit multiplier               : 1
 16x2-bit multiplier               : 1
 8x3-bit registered multiplier     : 1
 8x5-bit multiplier                : 1
 8x8-bit registered multiplier     : 1
 9x8-bit multiplier                : 2
 9x8-bit registered multiplier     : 2
# Adders/Subtractors               : 227
 10-bit adder                      : 114
 10-bit subtractor                 : 6
 11-bit subtractor                 : 12
 12-bit adder                      : 4
 12-bit subtractor                 : 6
 17-bit adder                      : 12
 18-bit adder                      : 2
 19-bit adder                      : 1
 2-bit subtractor                  : 1
 3-bit adder                       : 1
 3-bit subtractor                  : 4
 4-bit adder                       : 13
 4-bit addsub                      : 1
 4-bit subtractor                  : 1
 6-bit adder                       : 1
 6-bit subtractor                  : 1
 8-bit adder                       : 6
 8-bit addsub                      : 2
 8-bit subtractor                  : 6
 9-bit adder                       : 12
 9-bit addsub                      : 1
 9-bit subtractor                  : 20
# Adder Trees                      : 1
 10-bit / 4-inputs adder tree      : 1
# Counters                         : 18
 10-bit up counter                 : 1
 10-bit updown counter             : 1
 11-bit up counter                 : 1
 14-bit up counter                 : 1
 17-bit up counter                 : 1
 26-bit up counter                 : 3
 31-bit up counter                 : 1
 4-bit up counter                  : 5
 6-bit up counter                  : 1
 8-bit down counter                : 1
 8-bit up counter                  : 1
 9-bit updown counter              : 1
# Accumulators                     : 2
 12-bit up accumulator             : 1
 9-bit up accumulator              : 1
# Registers                        : 974
 Flip-Flops                        : 974
# Comparators                      : 193
 10-bit comparator equal           : 22
 10-bit comparator greater         : 35
 10-bit comparator lessequal       : 35
 11-bit comparator equal           : 16
 11-bit comparator greater         : 14
 11-bit comparator lessequal       : 15
 12-bit comparator equal           : 1
 12-bit comparator greater         : 1
 12-bit comparator lessequal       : 2
```

```
 17-bit comparator greater        : 3
 17-bit comparator lessequal      : 1
 17-bit comparator not equal      : 1
 4-bit comparator equal           : 4
 4-bit comparator greater         : 1
 6-bit comparator equal           : 1
 6-bit comparator greater         : 1
 6-bit comparator not equal       : 1
 8-bit comparator equal           : 4
 8-bit comparator greater         : 15
 8-bit comparator not equal       : 2
 9-bit comparator equal           : 9
 9-bit comparator greater         : 5
 9-bit comparator lessequal       : 2
 9-bit comparator not equal       : 2
# Multiplexers                    : 1173
 1-bit 14-to-1 multiplexer        : 1
 1-bit 15-to-1 multiplexer        : 1
 1-bit 16-to-1 multiplexer        : 7
 1-bit 2-to-1 multiplexer         : 596
 1-bit 4-to-1 multiplexer         : 1
 1-bit 7-to-1 multiplexer         : 1
 1-bit 8-to-1 multiplexer         : 2
 10-bit 2-to-1 multiplexer        : 137
 10-bit 8-to-1 multiplexer        : 1
 12-bit 2-to-1 multiplexer        : 18
 16-bit 2-to-1 multiplexer        : 1
 17-bit 2-to-1 multiplexer        : 28
 19-bit 2-to-1 multiplexer        : 5
 2-bit 2-to-1 multiplexer         : 64
 3-bit 2-to-1 multiplexer         : 114
 4-bit 2-to-1 multiplexer         : 22
 6-bit 2-to-1 multiplexer         : 29
 8-bit 2-to-1 multiplexer         : 36
 9-bit 2-to-1 multiplexer         : 109
# FSMs                            : 2
# Xors                            : 1
 1-bit xor2                       : 1

=========================================
```

Lampiran 6 *Datasheet*

1. *Datasheet* sensor kamera OV7670

OmniVision

Advanced Information
Preliminary Datasheet

OV7670/OV7171 CMOS VGA (640x480) CAMERACHIP™ Sensor with OmniPixel® Technology

General Description

The OV7670/OV7171 CAMERACHIP™ image sensor is a low voltage CMOS device that provides the full functionality of a single-chip VGA camera and image processor in a small footprint package. The OV7670/OV7171 provides full-frame, sub-sampled or windowed 8-bit images in a wide range of formats, controlled through the Serial Camera Control Bus (SCCB) interface.

This product has an image array capable of operating at up to 30 frames per second (fps) in VGA with complete user control over image quality, formatting and output data transfer. All required image processing functions, including exposure control, gamma, white balance, color saturation, hue control and more, are also programmable through the SCCB interface. In addition, OmniVision sensors use proprietary sensor technology to improve image quality by reducing or eliminating common lighting/electrical sources of image contamination, such as fixed pattern noise (FPN), smearing, blooming, etc., to produce a clean, fully stable color image.

Note: The OV7670/OV7171 uses a lead-free package.

Features

- High sensitivity for low-light operation
- Low operating voltage for embedded portable apps
- Standard SCCB interface compatible with I2C interface
- Output support for Raw RGB, RGB (GRB 4:2:2, RGB565/555/444), YUV (4:2:2) and YCbCr (4:2:2) formats
- Supports image sizes: VGA, CIF, and any size scaling down from CIF to 40x30
- VarioPixel® method for sub-sampling
- Automatic image control functions including: Automatic Exposure Control (AEC), Automatic Gain Control (AGC), Automatic White Balance (AWB), Automatic Band Filter (ABF), and Automatic Black-Level Calibration (ABLC)
- Image quality controls including color saturation, hue, gamma, sharpness (edge enhancement), and anti-blooming
- ISP includes noise reduction and defect correction
- Supports LED and flash strobe mode
- Supports scaling
- Lens shading correction
- Flicker (50/60 Hz) auto detection
- Saturation level auto adjust (UV adjust)
- Edge enhancement level auto adjust
- De-noise level auto adjust

Ordering Information

Product	Package
OV07670-VL2A (Color, lead-free)	24 pin CSP2
OV07171-VL2A (B&W, lead-free)	24 pin CSP2

Applications

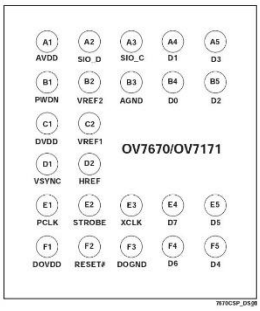
- Cellular and Picture Phones
- Toys
- PC Multimedia
- Digital Still Cameras

Key Specifications

Active Array Size		640 x 480
Power Supply	**Digital Core**	1.8VDC ±10%
	Analog	2.45V to 3.0V
	I/O	1.7V to 3.0V[a]
Power Requirements	**Active**	60 mW typical (15fps VGA YUV format)
	Standby	< 20 µA
Temperature Range	**Operation**	-30°C to 70°C
	Stable Image	0°C to 50°C
Output Formats (8-bit)		• YUV/YCbCr 4:2:2 • RGB565/555/444 • GRB 4:2:2 • Raw RGB Data
Lens Size		1/6"
Chief Ray Angle		25°
Maximum Image Transfer Rate		30 fps for VGA
Sensitivity		1.3 V/(Lux • sec)
S/N Ratio		46 dB
Dynamic Range		52 dB
Scan Mode		Progressive
Electronics Exposure		Up to 510:1 (for selected fps)
Pixel Size		3.6 µm x 3.6 µm
Dark Current		12 mV/s at 60°C
Well Capacity		17 K e
Image Area		2.36 mm x 1.76 mm
Package Dimensions		3785 µm x 4235 µm

a. I/O power should be 2.45V or higher when using the internal regulator for Core (1.8V); otherwise, it is necessary to provide an external 1.8V for the Core power supply.

Figure 1 OV7670/OV7171 Pin Diagram (Top View)



Version 1.4, August 21, 2006

Pin Description

Table 1 Pin Description

Pin Number	Name	Pin Type	Function/Description
A1	AVDD	Power	Analog power supply
A2	SIO_D	I/O	SCCB serial interface data I/O
A3	SIO_C	Input	SCCB serial interface clock input
A4	D1[a]	Output	YUV/RGB video component output bit[1]
A5	D3	Output	YUV/RGB video component output bit[3]
B1	PWDN	Input (0)[b]	Power Down Mode Selection 0: Normal mode 1: Power down mode
B2	VREF2	Reference	Reference voltage - connect to ground using a 0.1 µF capacitor
B3	AGND	Power	Analog ground
B4	D0	Output	YUV/RGB video component output bit[0]
B5	D2	Output	YUV/RGB video component output bit[2]
C1	DVDD	Power	Power supply (+1.8 VDC) for digital logic core
C2	VREF1	Reference	Reference voltage - connect to ground using a 0.1 µF capacitor
D1	VSYNC	Output	Vertical sync output
D2	HREF	Output	HREF output
E1	PCLK	Output	Pixel clock output
E2	STROBE	Output	LED/strobe control output
E3	XCLK	Input	System clock input
E4	D7	Output	YUV/RGB video component output bit[7]
E5	D5	Output	YUV/RGB video component output bit[5]
F1	DOVDD	Power	Digital power supply for I/O (1.7V ~ 3.0V)
F2	RESET#	Input	Clears all registers and resets them to their default values. 0: Reset mode 1: Normal mode
F3	DOGND	Power	Digital ground
F4	D6	Output	YUV/RGB video component output bit[6]
F5	D4	Output	YUV/RGB video component output bit[4]

a. D[7:0] for 8-bit YUV or RGB (D[7] MSB, D[0] LSB)
b. Input (0) represents an internal pull-down resistor.

Electrical Characteristics

Table 2 Absolute Maximum Ratings

Ambient Storage Temperature		-40°C to +95°C
Supply Voltages (with respect to Ground)	V_{DD-A}	4.5 V
	V_{DD-C}	3 V
	V_{DD-IO}	4.5 V
All Input/Output Voltages (with respect to Ground)		-0.3V to V_{DD-IO}+0.5V
Lead-free Temperature, Surface-mount process		245°C

NOTE: Exceeding the Absolute Maximum ratings shown above invalidates all AC and DC electrical specifications and may result in permanent device damage.

Table 3 DC Characteristics (-30°C < T_A < 70°C)

Symbol	Parameter	Condition	Min	Typ	Max	Unit
V_{DD-A}	DC supply voltage – Analog	–	2.45	2.75	3.0	V
V_{DD-C}	DC supply voltage – Digital Core	–	1.62	1.8	1.98	V
V_{DD-IO}	DC supply voltage – I/O power	–	1.7	–	3.0	V
I_{DDA}	Active (Operating) Current	See Note [a]		10 + 8[b]		mA
$I_{DDS-SCCB}$	Standby Current	See Note [c]		1		mA
$I_{DDS-PWDN}$	Standby Current			10	20	µA
V_{IH}	Input voltage HIGH	CMOS	0.7 x V_{DD-IO}			V
V_{IL}	Input voltage LOW				0.3 x V_{DD-IO}	V
V_{OH}	Output voltage HIGH	CMOS	0.9 x V_{DD-IO}			V
V_{OL}	Output voltage LOW				0.1 x V_{DD-IO}	V
I_{OH}	Output current HIGH	See Note [d]	8			mA
I_{OL}	Output current LOW		15			mA
I_L	Input/Output Leakage	GND to V_{DD-IO}			± 1	µA

a. V_{DD-A} = 2.5V, V_{DD-C} = 1.8V, V_{DD-IO} = 2.5V
$I_{DDA} = \Sigma\{I_{DD-IO} + I_{DD-C} + I_{DD-A}\}$, f_{CLK} = 24MHz at 30 fps YUV output, no I/O loading

b. I_{DD-C} = 10mA, I_{DD-A} = 8mA, without loading

c. V_{DD-A} = 2.5V, V_{DD-C} = 1.8V, V_{DD-IO} = 2.5V
$I_{DDS-SCCB}$ refers to a SCCB-initiated Standby, while $I_{DDS-PWDN}$ refers to a PWDN pin-initiated Standby

d. Standard Output Loading = 25pF, 1.2KΩ

Table 4 Functional and AC Characteristics (-30°C < T_A < 70°C)

Symbol	Parameter	Min	Typ	Max	Unit
Functional Characteristics					
	A/D Differential Non-Linearity		± 1/2		LSB
	A/D Integral Non-Linearity		± 1		LSB
	AGC Range			30	dB
	Red/Blue Adjustment Range			12	dB
Inputs (PWDN, CLK, RESET#)					
f_{CLK}	Input Clock Frequency	10	24	48	MHz
t_{CLK}	Input Clock Period	21	42	100	ns
$t_{CLK:DC}$	Clock Duty Cycle	45	50	55	%
$t_{S:RESET}$	Setting time after software/hardware reset			1	ms
$t_{S:REG}$	Settling time for register change (10 frames required)			300	ms
SCCB Timing (see Figure 4)					
f_{SIO_C}	Clock Frequency			400	KHz
t_{LOW}	Clock Low Period	1.3			µs
t_{HIGH}	Clock High Period	600			ns
t_{AA}	SIO_C low to Data Out valid	100		900	ns
t_{BUF}	Bus free time before new START	1.3			µs
$t_{HD:STA}$	START condition Hold time	600			ns
$t_{SU:STA}$	START condition Setup time	600			ns
$t_{HD:DAT}$	Data-in Hold time	0			µs
$t_{SU:DAT}$	Data-in Setup time	100			ns
$t_{SU:STO}$	STOP condition Setup time	600			ns
t_R, t_F	SCCB Rise/Fall times			300	ns
t_{DH}	Data-out Hold time	50			ns
Outputs (VSYNC, HREF, PCLK, and D[7:0] (see Figure 5, Figure 6, Figure 7, Figure 9, and Figure 10)					
t_{PDV}	PCLK[↓] to Data-out Valid			5	ns
t_{SU}	D[7:0] Setup time	15			ns
t_{HD}	D[7:0] Hold time	8			ns
t_{PHH}	PCLK[↓] to HREF[↑]	0		5	ns
t_{PHL}	PCLK[↓] to HREF[↓]	0		5	ns
AC Conditions:	• V_{DD}: V_{DD-C} = 1.8V, V_{DD-A} = 2.5V, V_{DD-IO} = 2.5V • Rise/Fall Times: I/O: 5ns, Maximum SCCB: 300ns, Maximum • Input Capacitance: 10pf • Output Loading: 25pF, 1.2KΩ to 2.5V • f_{CLK}: 24MHz				

2. *Datasheet* Spesifikasi SCCB

2 Pin Functions

Refer to Table 2-1 and Table 2-2 for pin descriptions of the master and slave devices, respectively, used in SCCB communications.

Table 2-1. Master Device Pin Descriptions

Signal Name	Signal Type	Description
SCCB_E[a]	Output	Serial Chip Select Output - master drives SCCB_E at logical 1 when the bus is idle. Drives at logical 0 when the master asserts transmissions or the system is in Suspend mode.
SIO_C	Output	Serial I/O Signal 1 Output - master drives SIO_C at logical 1 when the bus is idle. Drives at logical 0 and 1 when SCCB_E is driven at 0. Drives at logical 0 when the system is Suspend mode.
SIO_D	I/O	Serial I/O Signal 0 Input and Output - remains floating when the bus is idle and drives to logical 0 when the system is in Suspend mode.
PWDN	Output	Power down output

a. Where SCCB_E is not present on the CAMERACHIP, this signal is by default enabled and held high.

Table 2-2. Slave Device Pin Descriptions

Signal Name	Signal Type	Description
SCCB_E[a]	Input	Serial Chip Select Input - input pad can be shut down when the system is in Suspend mode.
SIO_C	Input	Serial I/O Signal 1 Input - input pad can be shut down when the system is in Suspend mode.
SIO_D	I/O	Serial I/O Signal 0 Input and Output - input pad can be shut down when the system is in Suspend mode.
PWDN	Input	Power down input

a. Where SCCB_E is not present on the CAMERACHIP, this signal is by default enabled and held high.

2.1 SCCB_E Signal

The SCCB_E signal is a single-directional, active-low, control signal that must be driven by the master device. It indicates the start or stop of the data transmission. A high-to-low transition of the SCCB_E indicates a start of a transmission, while the low-to-high transition of the SCCB_E indicates a stop of a transmission. SCCB_E must remain at logical 0 during a data transmission. A logical 1 of SCCB_E indicates that the bus is idle.

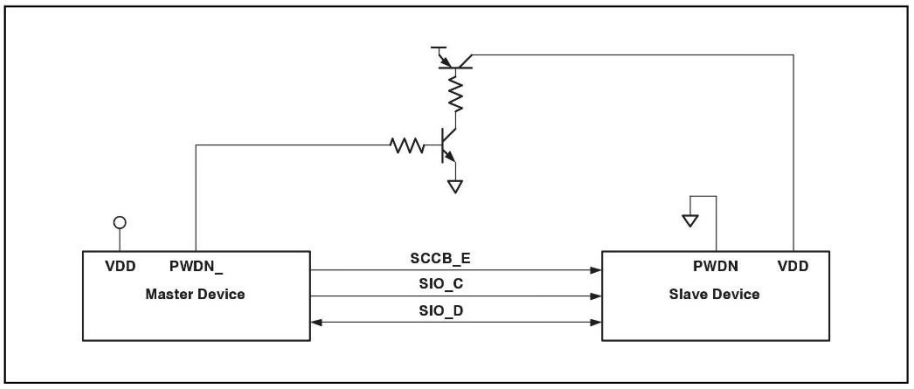
2.2 SIO_C

The SIO_C signal is a single-directional, active-high, control signal that must be driven by the master device. It indicates each transmitted bit. The master must drive SIO_C at logical 1 when the bus is idle. A data transmission starts when SIO_C is driven at logical 0 after the start of transmission. A logical 1 of SIO_C during a data transmission indicates a single transmitted bit. Thus, SIO_D can occur only when SIO_C is driven at 0. The period of a single transmitted bit is defined as t_{CYC} as shown in Figure 3-8. The minimum of t_{CYC} is 10 µs.

4.4.2 Switch Mode

The PWDN circuit of the slave(s) is always connected to logical 0. A power switch circuit is required for each slave. The power of each slave is OFF during suspend mode cycles. In suspend mode operation, there is no leakage current present as no power is provided to the slave(s).

Figure 4-5 Suspend Circuit - Switch Mode



5 Electrical Characteristics

Table 5-1. SCCB Electrical Characteristics

Symbol	Parameter	Condition	Min	Max	Unit
t_{cyc}	Single bit transmission cycle time		10		µs
t_{prc}	Pre-charge time of SIO_D		15		ns
t_{pra}	Pre-active time of SCCB_E		1.25		µs
t_{psc}	Post-charge time of SIO_D		15		µs
t_{psa}	Post-active time of SCCB_E		0		µs
t_{mack}	SIO_D_OE_M_ transition time		1.25		µs
t_{sack}	SIO_D_OE_S_ transition time		370		ns
t_{sup}	PWDN_ pre/post-charge time		50		ns

3. *Datasheet* Nexys3

Nexys3™ Board Reference Manual

Revision: December 28, 2011

DIGILENT
www.digilentinc.com
1300 Henley Court | Pullman, WA 99163
(509) 334 6306 Voice and Fax

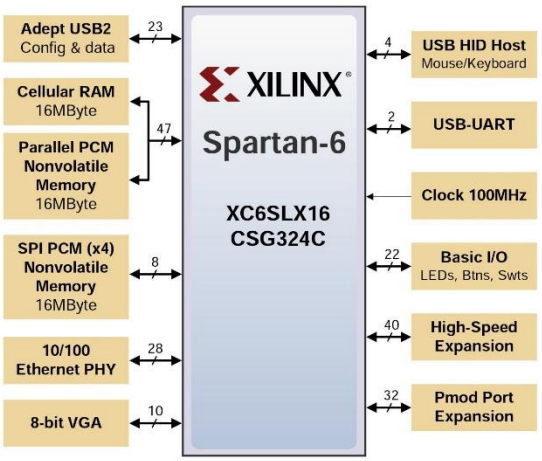
Overview

The Nexys3 is a complete, ready-to-use digital circuit development platform based on the Xilinx Spartan-6 LX16 FPGA. The Spartan-6 is optimized for high performance logic, and offers more than 50% higher capacity, higher performance, and more resources as compared to the Nexys2's Spartan-3 500E FPGA. Spartan-6 LX16 features include:

- 2,278 slices each containing four 6-input LUTs and eight flip-flops
- 576Kbits of fast block RAM
- two clock tiles (four DCMs & two PLLs)
- 32 DSP slices
- 500MHz+ clock speeds

In addition to the Spartan-6 FPGA, the Nexys3 offers an improved collection of peripherals including 32Mbytes of Micron's latest Phase Change nonvolatile memory, a 10/100 Ethernet PHY, 16Mbytes of Cellular RAM, a USB-UART port, a USB host port for mice and keyboards, and an improved high-speed expansion connector. The large FPGA and broad set of peripherals make the Nexys3 board an ideal host for a wide range of digital systems, including embedded processor designs based on Xilinx's MicroBlaze.

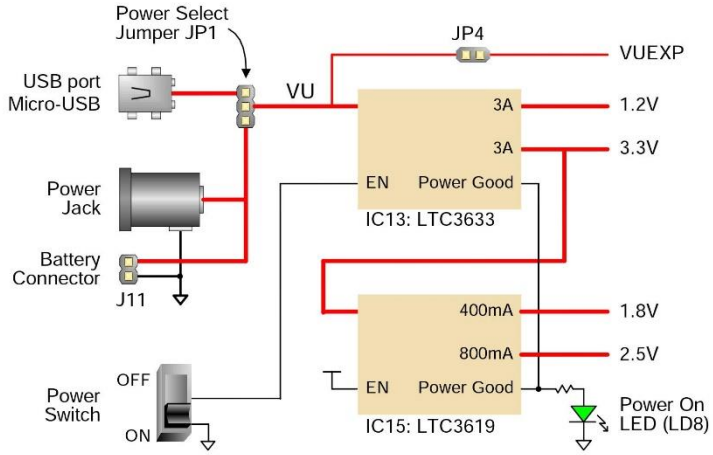
Nexys3 is compatible with all Xilinx CAD tools, including ChipScope, EDK, and the free WebPack. The Nexys3 uses Digilent's newest Adept USB2 system that offers FPGA and ROM programming, automated board tests, virtual I/O, and simplified user-data transfer facilities.



- *Xilinx Spartan-6 LX16 FPGA in a 324-pin BGA package*
- *16Mbyte Cellular RAM (x16)*
- *16Mbytes SPI (quad mode) PCM non-volatile memory*
- *16Mbytes parallel PCM non-volatile memory*
- *10/100 Ethernet PHY*
- *On-board USB2 port for programming & data xfer*
- *USB-UART and USB-HID port (for mouse/keyboard)*
- *8-bit VGA port*
- *100MHz CMOS oscillator*
- *72 I/O's routed to expansion connectors*
- *GPIO includes 8 LEDs, 6 buttons,8 slide switches and 4-digit seven-segment display*
- *USB2 programming cable included*

A comprehensive collection of board support IP and reference designs, and a large collection of add-on boards are available on the Digilent website. Please see the Nexys3 page at www.digilentinc.com for more information.

Nexys3 Power Supplies			
Supply	**Circuits**	**Device**	**Amps (max/typ)**
3.3V	FPGA I/O, USB ports, Clocks, ROM & RAM I/O, Ethernet	IC13: LTC3633	3A / 200mA
2.5V	Optional voltage for Bank0 and VHDC connector	IC14: LTC3619	800mA / 0mA
1.2V	FPGA Core	IC13: LTC3633	3A / 0.2 to 1.0A
1.8V	RAM and ROM core	IC14: LTC3619	400mA / 0.1 to 0.3A



The Nexys3 power supplies are enabled (or turned on) by a logic-level Power switch (SW8). A power-good LED (LD8), driven by the wired-OR of the "power good" outputs on the supplies, indicates that all supplies are operating within 10% of nominal.

The VU output of the main power jumper (JP1) is available to the VHDC expansion connector if jumper JP4 is loaded. Care must be taken to ensure the VUEXP delivered to any attached expansion board is the correct voltage – since VU is driven directly from an attached supply, this means a supply of the proper voltage must be used (e.g., 5V).

Memory

The Nexys3 board contains three external memories, all from Micron: a 128Mbit Cellular RAM (pseudo-static DRAM); a 128Mbit parallel non-volatile PCM (phase-change memory); and a 128Mbit serial PCM device. The Cellular RAM and parallel PCM device share a common bus, and the serial PCM is on a dedicated quad-mode (x4) SPI bus. The non-volatile PCM memories are byte and bit alterable without requiring a block erase, so they are faster and more versatile than conventional Flash in most applications. Reference designs available on the Digilent website show examples of using the Cellular RAM in asynchronous and synchronous modes, as well as examples of reading and writing both PCM devices.

Lampiran 7 Listing Program

1. Program Simulasi Algoritma Sistem

File: ImageTest.java

```
package imagetest;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ComponentEvent;
import java.awt.event.ComponentListener;
import java.awt.event.FocusEvent;
import java.awt.event.FocusListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.geom.Path2D;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.util.Arrays;

import javax.imageio.ImageIO;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextArea;

public class ImageTest extends JPanel
implements ActionListener, KeyListener,
FocusListener, MouseListener,
ComponentListener, ItemListener {
 private static final long serialVersionUID
= 1L;
 private final BufferedImage[] testImages;
 private final BufferedImage[] fontImages;
 private BufferedImage image;
 private BufferedImage[] segmentedImages;
 private Graph[] xgraphs, ygraphs;
 private BufferedImage xProj, yProj;
 private int refreshCount = 0;
 private boolean drawProj;
 private boolean segmentation = false;
 private final JButton prevButton;
 private final JButton nextButton;
 private final JButton projButton;
 private final JButton resetButton;
 private final JButton segmentButton;
 private final JButton plateNumberButton;
 private final JComboBox<String> comboBox;
 private final JButton saveImageButton;
 private final Filter[] filters;
 private final JLabel imgLabel;
 private final JTextArea summaryTextArea;
 private int index;
 private int indexSegmented;
 private long timeCheck;
 private long timeExecution;
 private long timeExecutionTotal;
 private int executionCount;
 private int rangeValueGraph = 256;
 private static final double Kr = 0.2126, Kg
= 0.7152, Kb = 0.0722;
 private static final int Cmb = 1, Cmp = 1,
Cms = 1, Cmas0 = 3, Cmas1 = 1, Cmc = 1, Cmac
= 1, Csb = 2, Csp = 3, Css = 4, Csas0 = 5,
Csas1 = 1, Csc = 4, Csac = 1, Ct = 2, Ce =
2, Cw = 320, Ch = 240, Csat0t = 0, Csat1t =
255, Csat0e = sat(48, Ce, false), Csat1e =
sat(48, Ce, true);
 private static final char[]
NUMBERS_AND_LETTERS =
{'0','1','2','3','4','5','6','7','8','9','A'
,'B','C','D','E','F','G','H','I','J','K','L'
,'M','N','O','P','Q','R','S','T','U','V','W'
,'X','Y','Z'};
 private static final boolean[] R;
 private static final boolean[] SE;
 private static final boolean[] R2;
 private static final boolean[] SE2;
 private static final boolean[] SE3;
 private static final boolean[] R3;
 private static final boolean[] R4;
 private static final int IMAGE_COUNT = 31;
 private static final int PANEL_WIDTH_MIN =
900;
 private static final int PANEL_HEIGHT_MIN =
700;
 private static final int BUTTON_WIDTH =
120;
 private static final int BUTTON_HEIGHT =
30;
 private static final int SPACE = 10;
 static {
  R = new boolean[9];
  SE = new boolean[39];
  Arrays.fill(R, true);
  Arrays.fill(SE, true);
  R2 = new boolean[25];
  Arrays.fill(R2, true);
  SE2 = new boolean[90];
  Arrays.fill(SE2, true);
  SE3 = new boolean[3];
  Arrays.fill(SE3, true);
  R3 = Arrays.copyOf(R, R.length);
  R3[0] = false;
  R3[2] = false;
  R3[6] = false;
  R3[8] = false;
  R4 = Arrays.copyOf(R, R.length);
  R4[4] = false;
 }

 private static final class Graph {
  private int[] curve;
  private int[] uPoints;
  private int[] points;
  private int[] uBoundOrigins;
  private int[] boundOrigins;
  private int[] uBoundLengths;
  private int[] boundLengths;
 }

 private static abstract class Filter {
  private final JButton button;

  Filter(String s) {
   button = new JButton(s);
  }
```

```
  public abstract BufferedImage
filter(BufferedImage image);
 }

 private static abstract class RGBFilter
extends Filter {

  RGBFilter(String s) {
   super(s);
  }

  public final BufferedImage
filter(BufferedImage image) {
   int[] d = new int[2], pixels =
getPixels(image, d);
   pixels = process(image.getRGB(0, 0,
image.getWidth(), image.getHeight(), null,
0, image.getWidth()), d);
   BufferedImage filteredImage = new
BufferedImage(d[0], d[1],
BufferedImage.TYPE_INT_RGB);
   filteredImage.setRGB(0, 0, d[0], d[1],
pixels, 0, d[0]);
   return filteredImage;
  }

  public abstract int[] process(int[]
pixels, int[] d);
 }

 public static void main(String[] args) {
  JFrame window = new JFrame("Image Test");
  Toolkit dt = Toolkit.getDefaultToolkit();
  ImageTest imgtest = new ImageTest();
  window.setContentPane(imgtest);
  window.pack();
  Dimension ukuranLayar =
dt.getScreenSize();
  window.setLocation((ukuranLayar.width -
window.getWidth()) / 2, (ukuranLayar.height
- window.getHeight()) / 2);

window.setDefaultCloseOperation(JFrame.EXIT_
ON_CLOSE);
  window.setVisible(true);
  window.setMinimumSize(new
Dimension(PANEL_WIDTH_MIN+20,
PANEL_HEIGHT_MIN+40));
 }

 private static int sat(int t, int l,
boolean b) {
  int temp;
  if (b) {
   temp = l*t > 256 ? 255 : l*t-1;
  }
  else {
   temp = l*t > 256 ? t-(int)Math.ceil((256-
t)/(l-1.0)) : 0;
  }
  return temp;
 }

 public ImageTest() {
  setLayout(null);
  setPreferredSize(new
Dimension(PANEL_WIDTH_MIN,
PANEL_HEIGHT_MIN));
  setSize(PANEL_WIDTH_MIN,
PANEL_HEIGHT_MIN);
  setBackground(Color.GRAY);
  ClassLoader cl =
getClass().getClassLoader();
  testImages = new
BufferedImage[IMAGE_COUNT];
  for (int i=0; i<IMAGE_COUNT; i++) {
   try {
    testImages[i] =
ImageIO.read(cl.getResource("test (" + i +
").jpg"));
   } catch (IOException e) {
    e.printStackTrace();
    System.exit(1);
   }
  }
  fontImages = new BufferedImage[36];
  for (int i=0; i<10; i++) {
   try {
    fontImages[i] =
ImageIO.read(cl.getResource("font - " + i +
".jpg"));
   } catch (IOException e) {
    e.printStackTrace();
    System.exit(1);
   }
  }
  for (int i=0; i<26; i++) {
   try {
    fontImages[i+10] =
ImageIO.read(cl.getResource("font - " +
(char)('A'+i) + ".jpg"));
   } catch (IOException e) {
    e.printStackTrace();
    System.exit(1);
   }
  }
  image = testImages[index];
  xgraphs = new Graph[]{new Graph()};
  ygraphs = new Graph[]{new Graph()};
  prevButton = new JButton("<");
  nextButton = new JButton(">");
  projButton = new JButton("proyeksi off");
  resetButton = new JButton("reset");
  segmentButton = new JButton("segmen");
  plateNumberButton = new JButton("plat
nomor");
  comboBox = new JComboBox<String>(new
String[]{"max", "top"});
  saveImageButton = new JButton("save
image");
  filters = new Filter[7];
  filters[0] = new RGBFilter("normalisasi")
{
   @Override
   public int[] process(int[] pixels, int[]
d) {
    return (int[]) numberPlate(pixels, d,
1);
   }
  };
  filters[1] = new RGBFilter("VED") {
   @Override
   public int[] process(int[] pixels, int[]
d) {
    pixels = (int[]) numberPlate(pixels, d,
1);
    pixels =
simpleEdge(getBrightness(pixels, d), d, Ce,
Csat0e, Csat1e);
    brightnessToRGB(pixels);
    return pixels;
   }
  };
  filters[2] = new RGBFilter("band
clipping") {
   @Override
   public int[] process(int[] pixels, int[]
d) {
    return (int[]) numberPlate(pixels, d,
2);
   }
  };
  filters[3] = new RGBFilter("plate
clipping") {
```

```
    @Override
    public int[] process(int[] pixels, int[]
d) {
     return (int[]) numberPlate(pixels, d,
3);
    }
   };
   filters[4] = new RGBFilter("threshold") {
    @Override
    public int[] process(int[] pixels, int[]
d) {
     return (int[]) numberPlate(pixels, d,
4);
    }
   };
   filters[5] = new RGBFilter("band clipping
2") {
    @Override
    public int[] process(int[] pixels, int[]
d) {
     pixels = (int[]) numberPlate(pixels, d,
2);
     pixels =
simpleEdge(getBrightness(pixels, d), d, Ce,
Csat0e, Csat1e);
     brightnessToRGB(pixels);
     return pixels;
    }
   };
   filters[6] = new RGBFilter("resize") {
    @Override
    public int[] process(int[] pixels, int[]
d) {
     return resizeTo(pixels, d, 16, 16);
    }
   };
   imgLabel = new JLabel("citra " + index,
JLabel.CENTER);
   imgLabel.setFont(new Font("Arial",
Font.BOLD, 20));
   summaryTextArea = new JTextArea("");
   summaryTextArea.setFont(new Font("Segoe
UI", Font.BOLD, 10));
   add(prevButton);
   add(nextButton);
   add(projButton);
   add(resetButton);
   add(segmentButton);
   add(plateNumberButton);
   add(comboBox);
   add(saveImageButton);
   for (Filter f : filters) {
    add(f.button);
   }
   add(imgLabel);
   add(summaryTextArea);
   computeBounds();
   prevButton.addActionListener(this);
   prevButton.setFocusable(false);
   nextButton.addActionListener(this);
   nextButton.setFocusable(false);
   projButton.addActionListener(this);
   projButton.setFocusable(false);
   resetButton.addActionListener(this);
   resetButton.setFocusable(false);
   segmentButton.addActionListener(this);
   segmentButton.setFocusable(false);
   plateNumberButton.addActionListener(this);
   plateNumberButton.setFocusable(false);
   comboBox.setFocusable(false);
   saveImageButton.addActionListener(this);
   saveImageButton.setFocusable(false);
   for (Filter f : filters) {
    f.button.addActionListener(this);
    f.button.setFocusable(false);
   }
   filters[6].button.setEnabled(false);
   summaryTextArea.setFocusable(false);
   comboBox.addItemListener(this);
   addKeyListener(this);
   addFocusListener(this);
   addMouseListener(this);
   addComponentListener(this);
   updateGraphDefault();
   index = 5;
   do {
    numberPlate();
   } while (timeExecution > 15);
   index = 0;
   timeExecutionTotal = 0;
   executionCount = 0;
  }

  @Override
  public void paintComponent(Graphics g) {
   super.paintComponent(g);
   g.drawImage(image, SPACE, SPACE, this);
   g.drawImage(xProj, SPACE,
image.getHeight(this)+SPACE*2, this);
   g.drawImage(yProj,
image.getWidth(this)+SPACE*2, SPACE, this);
  }

  private static int[] getColorSet(int in,
int length) {
   length = length < 1530 ? length : 1530;
   in %= length;
   int set = traverseSet(in, length);
   set = (int) (set*1530.0/length + 0.5);
   int[] colorSet = new int[3];
   colorSet[0] = getHSBColorFullSB(set+510);
   colorSet[1] = getHSBColorFullSB(set+1020);
   colorSet[2] = getHSBColorFullSB(set);
   return colorSet;
  }

  private static int traverseSet(int set, int
length) {
   if (length<=1)
    return 0;
   if (set%2==0)
    return traverseSet(set/2, length-
length/2);
   else
    return length-length/2+traverseSet(set/2,
length/2);
  }

  private static int getHSBColorFullSB(int
in) {
   in %= 1530;
   int rgb;
   switch (in/255) {
   case 0 :
    rgb = 0xffff0000 | (in%255)<<8; break;
   case 1 :
    rgb = 0xff00ff00 | (255-(in%255))<<16;
break;
   case 2 :
    rgb = 0xff00ff00 | (in%255); break;
   case 3 :
    rgb = 0xff0000ff | (255-(in%255))<<8;
break;
   case 4 :
    rgb = 0xff0000ff | (in%255)<<16; break;
   default :
    rgb = 0xffff0000 | (255-(in%255)); break;
   }
   return rgb;
  }

  @Override
  public void actionPerformed(ActionEvent
evt) {
```

```
  Object src = evt.getSource();
  if (src == prevButton) {
   previousImage();
   int[] dist = computeKNN(image);

summaryTextArea.setText(distanceText(dist));
  }
  else if (src == nextButton) {
   nextImage();
   int[] dist = computeKNN(image);

summaryTextArea.setText(distanceText(dist));
  }
  else if (src == projButton) {
   drawProj = !drawProj;
   projButton.setText(drawProj ? "proyeksi
on" : "proyeksi off");
   refreshCount = 0;
  }
  else if (src == resetButton) {
   if (segmentation)
    image = segmentedImages[indexSegmented];
   else
    image = testImages[index];
   refreshCount = 0;
   int[] dist = computeKNN(image);

summaryTextArea.setText(distanceText(dist));
  }
  else if (src == segmentButton) {
   segmentation = !segmentation;
   for (Filter f : filters) {

f.button.setEnabled(!f.button.isEnabled());
   }
   segmentButton.setText(segmentation ?
"citra" : "segmentasi");
   if (segmentation) {
    int[] d = new int[2], pixels =
getPixels(testImages[index], d);
    segmentedImages = (BufferedImage[])
numberPlate(pixels, d, 5);
    indexSegmented = 0;
    if (segmentedImages.length == 0) {
     summaryTextArea.setText("tidak ada
segmen!");
     segmentation = !segmentation;
     segmentButton.setText(segmentation ?
"citra" : "segmentasi");
    }
    else {
     image =
segmentedImages[indexSegmented];
     int[] dist = computeKNN(image);

summaryTextArea.setText(distanceText(dist));
     summaryTextArea.append("waktu eksekusi
" + timeExecution + " ms");
    }
   }
   else {
    image = testImages[index];
    int[] dist = computeKNN(image);

summaryTextArea.setText(distanceText(dist));
   }
   refreshCount = 0;
  }
  else if (src == plateNumberButton) {
   summaryTextArea.setText(numberPlate());
   summaryTextArea.append("\nwaktu eksekusi
" + timeExecution + " ms");
   summaryTextArea.append("\nwaktu eksekusi
rata-rata " +
(timeExecutionTotal/(double)executionCount)
+ " ms");
  }
  else if (src == saveImageButton) {
   summaryTextArea.setText("image saved");
   try {
            ImageIO.write(image, "JPG", new
File("C:/Users/Diannata
Rahman/Desktop/test_image.JPG"));
        } catch (Exception e) {
    summaryTextArea.setText(e.getMessage());
        }
  }
  for (Filter f : filters) {
   if (src == f.button) {
    summaryTextArea.setText("");
    if (segmentation) {
     image =
f.filter(segmentedImages[indexSegmented]);
     int[] dist = computeKNN(image);

summaryTextArea.setText(distanceText(dist));
    }
    else
     image = f.filter(testImages[index]);
    refreshCount = 0;
    summaryTextArea.append("waktu eksekusi "
+ timeExecution + " ms");
    break;
   }
  }
  if (src == filters[1].button || src ==
filters[5].button) {
   updateGraphCrop();
  }
  else if (src == filters[4].button) {
   updateGraphSegments();
  }
  else {
   updateGraphDefault();
  }
  if (segmentation)
   imgLabel.setText("segmen " +
indexSegmented);
  else
   imgLabel.setText("citra " + index);
  repaint();
 }

 private void previousImage() {
  if (segmentation) {
   if (indexSegmented>0) {
    indexSegmented--;
    refreshCount = 0;
    image = segmentedImages[indexSegmented];
   }
  }
  else {
   if (index>0) {
    index--;
    refreshCount = 0;
    image = testImages[index];
   }
  }
 }

 private void nextImage() {
  if (segmentation) {
   if
(indexSegmented<segmentedImages.length-1) {
    indexSegmented++;
    refreshCount = 0;
    image = segmentedImages[indexSegmented];
   }
  }
  else {
   if (index<IMAGE_COUNT-1) {
    index++;
    refreshCount = 0;
    image = testImages[index];
```

```
    }
   }
  }

  private String distanceText(int[] dist) {
   int[] in = sort(dist);
   String s = "";
   for (int i=0; i<36; i++) {
    s += NUMBERS_AND_LETTERS[in[i]] + " = " +
dist[in[i]];
    i++;
    s += "\t" + NUMBERS_AND_LETTERS[in[i]] +
" = " + dist[in[i]];
    s += "\n";
   }
   return s;
  }

  private static int[] getProjection(int[]
pixels, int[] d, boolean h, int l) {
   int[] pr = new int[h ? d[0] : d[1]];
   for (int y=0; y<d[1]; y++) {
    for (int x=0; x<d[0]; x++) {
     int px = pixels[y*d[0]+x];
     px = (int)(px/255.0*(l-1.0)+0.5);
     if (h)
      pr[x] += px;
     else
      pr[y] += px;
    }
   }
   return pr;
  }

  private BufferedImage[]
createSegmentedImages(int[] pixels, int[] d)
{
   int[] bounds =
segmentedImageBounds(pixels, d);
   int count = bounds.length/4;
   BufferedImage[] si = new
BufferedImage[count];
   for (int i=0; i<count; i++) {
    int[] dd = Arrays.copyOf(d, d.length);
    int[] fc = crop(pixels, dd, bounds[i*4],
bounds[i*4+2], bounds[i*4+1]-bounds[i*4]+1,
bounds[i*4+3]-bounds[i*4+2]+1);
    si[i] = new BufferedImage(dd[0], dd[1],
BufferedImage.TYPE_INT_RGB);
    si[i].setRGB(0, 0, dd[0], dd[1], fc, 0,
dd[0]);
   }
   return si;
  }

  private int[] segmentedImageBounds(int[]
pixels, int[] d) {
   int[] px =
getProjection(getBrightness(pixels, d), d,
true, Ct);
   int[] bounds = new int[80];
   boolean valid = false;
   boolean up = true;
   int count = 0, area = 0, t = (d[1]*(Ct-
1)*Cms)>>Css, ta0 = (d[1]*d[1]*(Ct-
1)*Cmas0)>>Csas0, ta1 = (d[1]*d[1]*(Ct-
1)*Cmas1)>>Csas1;
   for (int i=0; ; i++) {
    if (count == bounds.length)
     break;
    if (i == d[0]) {
     if (valid) {
      int[] dd = Arrays.copyOf(d, d.length);
      int[] fc = crop(pixels, dd,
bounds[count*4], 0, bounds[count*4+1]-
bounds[count*4]+1, d[1]);
      int[] b = vBounds2(fc, dd, Cmc, Csc);
      bounds[count*4+2] = b[0];
      bounds[count*4+3] = b[1];
      if (b[1]-b[0]+1 > (dd[1]*Cmac>>Csac))
       count++;
     }
     break;
    }
    boolean c = px[i] <= t;
    if (up) {
     if (!c) {
      bounds[count*4] = i;
      bounds[count*4+1] = i;
      area = px[i];
      up = false;
     }
    }
    else {
     if (c) {
      up = true;
      if (valid) {
       valid = false;
       area = 0;
       int[] dd = Arrays.copyOf(d, d.length);
       int[] fc = crop(pixels, dd,
bounds[count*4], 0, bounds[count*4+1]-
bounds[count*4]+1, d[1]);
       int[] b = vBounds2(fc, dd, Cmc, Csc);
       bounds[count*4+2] = b[0];
       bounds[count*4+3] = b[1];
       if (b[1]-b[0]+1 > (dd[1]*Cmac>>Csac))
        count++;
      }
     }
     else {
      area += px[i];
      bounds[count*4+1] = i;
      valid = area > ta0 && area <= ta1;
     }
    }
   }
   return Arrays.copyOf(bounds, count*4);
  }

  private static int[] vBounds(int[] pixels,
int[] d, int cm, int cs) {
   int[] dd = Arrays.copyOf(d, d.length);
   int[] f = simpleEdge(getBrightness(pixels,
dd), dd, Ce, Csat0e, Csat1e);
   int[] py = getProjection(f, dd, false,
Ce);
   int ym = maxIndex(py), pym = py[ym], y0,
y1, t = (pym*cm)>>cs;
   for (y0=ym; y0>0; y0--) {
    if (py[y0-1] <= t)
     break;
   }
   for (y1=ym; y1<dd[1]-1; y1++) {
    if (py[y1+1] <= t)
     break;
   }
   return new int[]{y0, y1, ym};
  }

  private static int[] vBounds2(int[] pixels,
int[] d, int cm, int cs) {
   int[] py =
getProjection(getBrightness(pixels, d), d,
false, Ct);
   int ym = maxIndex(py), pym = py[ym], y0,
y1, t = ((pym*cm)>>cs);
   for (y0=ym; y0>0; y0--) {
    if (py[y0-1] <= t)
     break;
   }
   for (y1=ym; y1<d[1]-1; y1++) {
    if (py[y1+1] <= t)
     break;
```

```
  }
  return new int[]{y0, y1, ym};
 }

 private static int[] plateBounds(int[]
pixels, int[] d, int cm, int cs) {
  int[] dd = Arrays.copyOf(d, d.length);
  int[] f = simpleEdge(getBrightness(pixels,
dd), dd, Ce, Csat0e, Csat1e);
  int[] px = getProjection(f, dd, true, Ce);
  final int w = dd[1]*25/2;
  int xam = 0, am = 0, a = 0, xamw=0;
  for (int i=0; i < px.length; i++) {
   a += px[i];
   if (i >= w) {
    a -= px[i-w];
    if (a > am) {
     xam = i-w;
     xamw = i;
     am = a;
    }
   }
   else
    xamw = i;
  }
  int pxm = px[maxIndex(px)], x0, x1, t =
((pxm*cm)>>cs);
  boolean b1 = false, b2 = false, b3 =
false;
  for (x0=xam, x1=xamw; x0<x1 && !(b2&&b3);
) {
   if (b1) {
    if (px[x0] >= t)
     b2=true;
    else
     x0++;
    if (!b3)
     b1 = !b1;
   }
   else {
    if (px[x1] >= t)
     b3=true;
    else
     x1--;
    if (!b2)
     b1 = !b1;
   }
  }
  return new int[]{x0, x1, xam, xamw};
 }

 private int[] computeKNN(BufferedImage
image) {
  int[] d = new int[2], pixels =
getPixels(image, d);
  return computeKNN(pixels, d);
 }

 private int[] computeKNN(int[] pixels,
int[] d) {
  int[] fr = getBrightness(resizeTo(pixels,
d, 16, 16), d);
  int[] dd = new int[2];
  int[] dist = new int[36];
  for (int i=0; i<36; i++) {
   int[] font =
getBrightness(getPixels(fontImages[i], dd),
dd);
   for(int y=0; y<16; y++) {
    for(int x=0; x<16; x++) {
     if (fr[y*16+x] < 128 != font[y*16+x] <
128)
      dist[i]++;
    }
   }
  }
  return dist;
 }

 private void updateGraphDefault() {
  int[] d = new int[2];
  xgraphs[0].curve =
getProjection(getBrightness(getPixels(image,
d), d), d, true, 256);
  ygraphs[0].curve =
getProjection(getBrightness(getPixels(image,
d), d), d, false, 256);
  xgraphs[0].points = null;
  ygraphs[0].points = null;
  xgraphs[0].uPoints = new
int[]{maxIndex(xgraphs[0].curve)};
  ygraphs[0].uPoints = new
int[]{maxIndex(ygraphs[0].curve)};
  xgraphs[0].boundOrigins = null;
  ygraphs[0].boundOrigins = null;
  xgraphs[0].boundLengths = null;
  ygraphs[0].boundLengths = null;
  xgraphs[0].uBoundOrigins = null;
  ygraphs[0].uBoundOrigins = null;
  xgraphs[0].uBoundLengths = null;
  ygraphs[0].uBoundLengths = null;
  rangeValueGraph = 256;
  updateGraph();
 }

 private void updateGraphCrop() {
  int[] d = new int[2], pixels =
getPixels(testImages[index], d);
  pixels = (int[]) numberPlate(pixels, d,
1);
  int[] vb = vBounds(pixels, d, Cmb, Csb);
  int[] pb = plateBounds(pixels, d, Cmp,
Csp);
  xgraphs[0].curve =
getProjection(simpleEdge(getBrightness(pixel
s, d), d, Ce, Csat0e, Csat1e), d, true, Ce);
  ygraphs[0].curve =
getProjection(simpleEdge(getBrightness(pixel
s, d), d, Ce, Csat0e, Csat1e), d, false,
Ce);
  xgraphs[0].points = new int[]{pb[0],
pb[1]};
  ygraphs[0].points = new int[]{vb[0],
vb[1]};
  xgraphs[0].uPoints = null;
  ygraphs[0].uPoints = new int[]{vb[2]};
  xgraphs[0].boundOrigins = null;
  ygraphs[0].boundOrigins = null;
  xgraphs[0].boundLengths = null;
  ygraphs[0].boundLengths = null;
  xgraphs[0].uBoundOrigins = new
int[]{pb[2]};
  ygraphs[0].uBoundOrigins = null;
  xgraphs[0].uBoundLengths = new
int[]{pb[3]-pb[2]+1};
  ygraphs[0].uBoundLengths = null;
  rangeValueGraph = Ce;
  updateGraph();
 }

 private void updateGraphSegments() {
  int[] d = new int[2], pixels =
getPixels(testImages[index], d);
  pixels = (int[]) numberPlate(pixels, d,
4);
  int[] b = segmentedImageBounds(pixels, d);
  xgraphs[0].curve =
getProjection(getBrightness(pixels, d), d,
true, 256);
  ygraphs[0].curve =
getProjection(getBrightness(pixels, d), d,
false, 256);
  int[] xp = new int[b.length/2];
  int[] yp = new int[b.length/2];
```

```
  int count = b.length/4;
  for (int i=0; i<count; i++) {
   xp[i*2] = b[i*4];
   xp[i*2+1] = b[i*4+1];
   yp[i*2] = b[i*4+2];
   yp[i*2+1] = b[i*4+3];
  }
  xgraphs[0].points = xp;
  ygraphs[0].points = yp;
  xgraphs[0].uPoints = null;
  ygraphs[0].uPoints = null;
  xgraphs[0].boundOrigins = null;
  ygraphs[0].boundOrigins = null;
  xgraphs[0].boundLengths = null;
  ygraphs[0].boundLengths = null;
  xgraphs[0].uBoundOrigins = null;
  ygraphs[0].uBoundOrigins = null;
  xgraphs[0].uBoundLengths = null;
  ygraphs[0].uBoundLengths = null;
  rangeValueGraph = Ce;
  updateGraph();
 }

 private Path2D createPath(int[] array,
boolean h) {
  Path2D path = new Path2D.Double();
  path.moveTo(0.0, 0.0);
  for (int i=0; i<array.length; i++) {
   if (h)
    path.lineTo(i, array[i]);
   else
    path.lineTo(array[i], i);
  }
  if (h)
   path.lineTo(array.length-1.0, 0.0);
  else
   path.lineTo(0.0, array.length-1.0);
  path.closePath();
  return path;
 }

 private void updateGraph() {
  int[] d = new int[2];
  do {
   d[0] = image.getWidth(null);
   d[1] = image.getHeight(null);
  } while (d[0] <= 0 || d[1] <= 0);
  Graphics2D g;
  if (refreshCount==0 || !drawProj) {
   xProj = new BufferedImage(d[0], 100,
BufferedImage.TYPE_INT_ARGB);
   g = xProj.createGraphics();
   g.setColor(Color.WHITE);
   g.fillRect(0,0,d[0],100);
  }
  else
   g = xProj.createGraphics();

g.setRenderingHint(RenderingHints.KEY_ANTIAL
IASING, RenderingHints.VALUE_ANTIALIAS_ON);
        int in =
comboBox.getSelectedIndex();
        int tx = 0;
        if (xgraphs != null) {
         for (Graph gr : xgraphs) {
          int temp =
gr.curve[maxIndex(gr.curve)];
          tx = tx > temp ? tx : temp;
         }
        }
        switch (in) {
        case 1 :
         tx = (rangeValueGraph-1)*d[1];
         break;
        default :
         break;
        }
        g.scale(1.0, 100.0/tx);
        if (drawProj && xgraphs != null) {
         for (int i=0; i<xgraphs.length;
i++) {
          int[] c =
getColorSet(i+refreshCount,
xgraphs.length+refreshCount);
          if (xgraphs[i].curve != null) {
              Path2D path =
createPath(xgraphs[i].curve, true);
        g.setColor(new Color(0x0fffffff &
c[0], true));
        g.fill(path);
        g.setColor(new Color(c[0]));
        g.draw(path);
          }
          if (xgraphs[i].boundOrigins !=
null && xgraphs[i].boundLengths != null) {
        g.setColor(new Color(0x3fffffff &
c[1], true));
        for (int j=0;
j<xgraphs[i].boundOrigins.length; j++) {

g.fillRect(xgraphs[i].boundOrigins[j],0,xgra
phs[i].boundLengths[j],tx);
        }
          }
          if (xgraphs[i].uBoundOrigins !=
null && xgraphs[i].uBoundLengths != null) {
        g.setColor(new Color(0x3fffffff &
c[2], true));
        for (int j=0;
j<xgraphs[i].uBoundOrigins.length; j++) {

g.fillRect(xgraphs[i].uBoundOrigins[j],0,xgr
aphs[i].uBoundLengths[j],tx);
        }
          }
          if (xgraphs[i].points != null) {
        g.setColor(new Color(c[1]));
        for (int j=0;
j<xgraphs[i].points.length; j++) {

g.drawLine(xgraphs[i].points[j],0,xgraphs[i]
.points[j],tx);
        }
          }
          if (xgraphs[i].uPoints != null) {
        g.setColor(new Color(c[2]));
        for (int j=0;
j<xgraphs[i].uPoints.length; j++) {

g.drawLine(xgraphs[i].uPoints[j],0,xgraphs[i
].uPoints[j],tx);
        }
          }
         }
  }
  g.dispose();
  if (refreshCount==0 || !drawProj) {
   yProj = new BufferedImage(100, d[1],
BufferedImage.TYPE_INT_ARGB);
   g = yProj.createGraphics();
   g.setColor(Color.WHITE);
   g.fillRect(0,0,100,d[1]);
  }
  else
   g = yProj.createGraphics();

g.setRenderingHint(RenderingHints.KEY_ANTIAL
IASING, RenderingHints.VALUE_ANTIALIAS_ON);
        int ty = 0;
        if (ygraphs != null) {
         for (Graph gr : ygraphs) {
          int temp =
gr.curve[maxIndex(gr.curve)];
          ty = ty > temp ? ty : temp;
```

```
          }
         }
         switch (in) {
         case 1 :
          ty = (rangeValueGraph-1)*d[0];
          break;
         default :
          break;
         }
         g.scale(100.0/ty, 1.0);
  if (drawProj && ygraphs != null) {
   for (int i=0; i<ygraphs.length; i++) {
            int[] c =
getColorSet(i+refreshCount,
ygraphs.length+refreshCount);
           if (ygraphs[i].curve != null) {
               Path2D path =
createPath(ygraphs[i].curve, false);
         g.setColor(new Color(0x0fffffff &
c[0], true));
         g.fill(path);
         g.setColor(new Color(c[0]));
         g.draw(path);
           }
           if (ygraphs[i].boundOrigins !=
null && ygraphs[i].boundLengths != null) {
         g.setColor(new Color(0x3fffffff &
c[1], true));
         for (int j=0;
j<ygraphs[i].boundOrigins.length; j++) {

g.fillRect(0,ygraphs[i].boundOrigins[j],ty,y
graphs[i].boundLengths[j]);
         }
           }
           if (ygraphs[i].uBoundOrigins !=
null && ygraphs[i].uBoundLengths != null) {
         g.setColor(new Color(0x3fffffff &
c[2], true));
         for (int j=0;
j<ygraphs[i].uBoundOrigins.length; j++) {

g.fillRect(0,ygraphs[i].uBoundOrigins[j],ty,
ygraphs[i].uBoundLengths[j]);
         }
           }
           if (ygraphs[i].points != null) {
         g.setColor(new Color(c[1]));
         for (int j=0;
j<ygraphs[i].points.length; j++) {

g.drawLine(0,ygraphs[i].points[j],ty,ygraphs
[i].points[j]);
         }
           }
           if (ygraphs[i].uPoints != null) {
         g.setColor(new Color(c[2]));
         for (int j=0;
j<ygraphs[i].uPoints.length; j++) {

g.drawLine(0,ygraphs[i].uPoints[j],ty,ygraph
s[i].uPoints[j]);
         }
           }
          }
  }
  g.dispose();
  refreshCount++;
 }

 private static int maxIndex(int[] array) {
  int mi = 0;
  for(int i=1; i<array.length; i++) {
   if (array[i] > array[mi]) {
    mi = i;
   }
  }
  return mi;
 }

 private static int minIndex(int[] array) {
  int mi = 0;
  for(int i=1; i<array.length; i++) {
   if (array[i] < array[mi]) {
    mi = i;
   }
  }
  return mi;
 }

 private static int[]
getPixels(BufferedImage image, int[] d) {
  d[0] = image.getWidth();
  d[1] = image.getHeight();
  return image.getRGB(0, 0,
image.getWidth(), image.getHeight(),
    null, 0, image.getWidth());
 }

 private static int[] getBrightness(int[]
pixels, int[] d) {
  int[] pb = new int[d[0]*d[1]];
  for(int i=0; i<pixels.length; i++) {
   pb[i] = getBrightness(pixels[i]);
  }
  return pb;
 }

 private static int threshold(int px, int l,
boolean b, int sat0, int sat1) {
  int temp = b ? getBrightness(px) : px;
  if (temp < sat0)
   return 0;
  else if (temp > sat1)
   return 255;
  temp = (temp-sat0)*l/(sat1-sat0+1);
  temp = (int)(temp*255/(l-1.0)+0.5);
  return temp;
 }

 private static int[] getAlpha(int[] pixels,
int[] d) {
  int[] pb = new int[d[0]*d[1]];
  for(int i=0; i<pixels.length; i++) {
   pb[i] = (pixels[i]>>24) & 0xff;
  }
  return pb;
 }

 private static int[] getRed(int[] pixels,
int[] d) {
  int[] pb = new int[d[0]*d[1]];
  for(int i=0; i<pixels.length; i++) {
   pb[i] = (pixels[i]>>16) & 0xff;
  }
  return pb;
 }

 private static int[] getGreen(int[] pixels,
int[] d) {
  int[] pb = new int[d[0]*d[1]];
  for(int i=0; i<pixels.length; i++) {
   pb[i] = (pixels[i]>>8) & 0xff;
  }
  return pb;
 }

 private static int[] getBlue(int[] pixels,
int[] d) {
  int[] pb = new int[d[0]*d[1]];
  for(int i=0; i<pixels.length; i++) {
   pb[i] = pixels[i] & 0xff;
  }
  return pb;
```

```
 }

 private static int[] mixRGB(int[] pa, int[]
pr, int[] pg, int[] pb, int[] d) {
  int[] pm = new int[d[0]*d[1]];
  normalizeBrightness(pa);
  normalizeBrightness(pr);
  normalizeBrightness(pg);
  normalizeBrightness(pb);
  for(int i=0; i<pm.length; i++) {
   pm[i] = (pa[i]<<24 | pr[i]<<16 | pg[i]<<8
| pb[i]);
  }
  return pm;
 }

 private static int getBrightness(int rgb) {
  int r = (rgb>>16) & 0xff;
  int g = (rgb>>8) & 0xff;
  int b = rgb & 0xff;
  return (int) (Kr*r + Kg*g + Kb*b + 0.5);
 }

 private static void
normalizeBrightness(int[] pixels) {
  for(int i=0; i<pixels.length; i++) {
   int px = pixels[i];
   if (px > 255)
    px = 255;
   else if (px < 0)
    px = 0;
   pixels[i] = px;
  }
 }

 private static void brightnessToRGB(int[]
pixels) {
  normalizeBrightness(pixels);
  for(int i=0; i<pixels.length; i++) {
   int px = pixels[i];
   pixels[i] = (0xff000000 | px<<16 | px<<8
| px);
  }
 }

 private static int[] simpleEdge(int[] f,
int[] d, int t, int sat0, int sat1) {
  int[] fe = new int[d[0]*d[1]];
  for (int y=0; y<d[1]; y++) {
   for (int x=0; x<d[0]; x++) {
    int temp = 0;
    if (x > 0) {
     temp = Math.abs(f[y*d[0]+x]-f[y*d[0]+x-
1]);
    }
    if (x < d[0]-1) {
     int temp2 = Math.abs(f[y*d[0]+x]-
f[y*d[0]+x+1]);
     if (temp2 > temp)
      temp = temp2;
    }
    fe[y*d[0]+x] = threshold(temp, t, false,
sat0, sat1);
   }
  }
  return fe;
 }

 private static int[] resizeTo(int[] f,
int[] d, int w, int h) {
  int[] fr, fra = new int[w*h], frr = new
int[w*h], frg = new int[w*h], frb = new
int[w*h],
    pa = getAlpha(f, d), pr = getRed(f, d),
pg = getGreen(f, d), pb = getBlue(f, d);
  int cy = 0, ya = 0, yb = 0, ly = d[1];
  int[] pya = new int[w], pyr = new int[w],
pyg = new int[w], pyb = new int[w];
  while (yb<h) {
   if (cy==0) {
    cy = h;
    for (int i=0; i<pyr.length; i++) {
     pya[i] = 0;
    }
    for (int i=0; i<pyr.length; i++) {
     pyr[i] = 0;
    }
    for (int i=0; i<pyg.length; i++) {
     pyg[i] = 0;
    }
    for (int i=0; i<pyb.length; i++) {
     pyb[i] = 0;
    }
    int cx = 0, xa = 0, xb = 0, pxa = 0, pxr
= 0, pxg = 0, pxb = 0, lx = d[0];
    while (xb<w) {
     if (cx==0) {
      cx = w;
      pxa = pa[ya*d[0]+xa];
      pxr = pr[ya*d[0]+xa];
      pxg = pg[ya*d[0]+xa];
      pxb = pb[ya*d[0]+xa];
      xa++;
     }
     else {
      int dx = cx<lx ? cx : lx;
      cx -= dx;
      lx -= dx;
      pya[xb] += pxa*dx;
      pyr[xb] += pxr*dx;
      pyg[xb] += pxg*dx;
      pyb[xb] += pxb*dx;
      if (lx==0) {
       lx = d[0];
       xb++;
      }
     }
    }
    ya++;
   }
   else {
    int dy = cy<ly ? cy : ly;
    cy -= dy;
    ly -= dy;
    for (int xb=0;xb<w;xb++) {
     fra[yb*w+xb] += pya[xb]*dy;
    }
    for (int xb=0;xb<w;xb++) {
     frr[yb*w+xb] += pyr[xb]*dy;
    }
    for (int xb=0;xb<w;xb++) {
     frg[yb*w+xb] += pyg[xb]*dy;
    }
    for (int xb=0;xb<w;xb++) {
     frb[yb*w+xb] += pyb[xb]*dy;
    }
    if (ly==0) {
     ly = d[1];
     yb++;
    }
   }
  }

  int s = d[0]*d[1];
  for(int y=0; y<h; y++) {
   for(int x=0; x<w; x++) {
    int a = fra[y*w+x];
    int r = frr[y*w+x];
    int g = frg[y*w+x];
    int b = frb[y*w+x];
    fra[y*w+x] = (a+s/2)/s;
    frr[y*w+x] = (r+s/2)/s;
    frg[y*w+x] = (g+s/2)/s;
```

```
    frb[y*w+x] = (b+s/2)/s;
   }
  }

  d[0] = w;
  d[1] = h;
  fr = mixRGB(fra, frr, frg, frb, d);

  return fr;
 }

 private static int[] crop(int[] f, int[] d,
int px, int py, int w, int h) {
  int[] fc = new int[w*h];
  for(int y=0; y<h; y++) {
   for(int x=0; x<w; x++) {
    int q = y+py, p = x+px;
    if (q >= 0 && q < d[1] && p >= 0 && p <
d[0])
     fc[y*w+x] = f[q*d[0]+p];
   }
  }
  d[0] = w;
  d[1] = h;
  return fc;
 }

 private static int[] sort(int[] value) {
  int[] index = new int[value.length];
  for (int i=0; i<index.length; i++) {
   index[i] = i;
  }
  value = Arrays.copyOf(value,
value.length);
  sort(index, value, 0, value.length);
  return index;
 }

 private static void sort(int[] index, int[]
value, int start, int length) {
  if (length <= 1)
   return;
  sort(index, value, start, length/2);
  sort(index, value, start+length/2, length-
length/2);
  int[] c1 = Arrays.copyOfRange(index,
start, start+length/2);
  int[] c2 = Arrays.copyOfRange(index,
start+length/2, start+length);
  int[] i1 = Arrays.copyOfRange(value,
start, start+length/2);
  int[] i2 = Arrays.copyOfRange(value,
start+length/2, start+length);
  for (int j=0,k=0; j+k<length;) {
   if (k == i2.length || (j < i1.length &&
i1[j] <= i2[k])) {
    value[start+j+k] = i1[j];
    index[start+j+k] = c1[j];
    j++;
   }
   else {
    value[start+j+k] = i2[k];
    index[start+j+k] = c2[k];
    k++;
   }
  }
 }

 @Override
 public void focusGained(FocusEvent evt) {
  repaint();
 }

 @Override
 public void focusLost(FocusEvent evt) {
  repaint();
 }
```

```
 @Override
 public void keyPressed(KeyEvent evt) {
  int key = evt.getKeyCode();
  if (key == KeyEvent.VK_LEFT) {
   previousImage();
  }
  else if (key == KeyEvent.VK_RIGHT) {
   nextImage();
  }
  int[] dist = computeKNN(image);

summaryTextArea.setText(distanceText(dist));
  updateGraphDefault();
  if (segmentation)
   imgLabel.setText("segmen " +
indexSegmented);
  else
   imgLabel.setText("citra " + index);
  repaint();
 }

 @Override
 public void keyReleased(KeyEvent evt) {
  repaint();
 }

 private String numberPlate() {
  int[] d = new int[2], pixels =
getPixels(testImages[index], d);
  return (String) numberPlate(pixels, d, 0);
 }

 private Object numberPlate(int[] pixels,
int[] d, int stage) {
  String s = "";
  double d0 = (double) Cw/d[0], d1 =
(double) Ch/d[1];
  timeExecution = 0;
  timeCheck = System.currentTimeMillis();
  if (d0 < d1)
   pixels = resizeTo(pixels, d, Cw,
d[1]*Cw/d[0]);
  else
   pixels = resizeTo(pixels, d,
d[0]*Ch/d[1], Ch);
  pixels = crop(pixels, d, (d[0]-Cw)/2,
(d[1]-Ch)/2, Cw, Ch);
  if (stage == 1) {
   timeExecution =
System.currentTimeMillis()-timeCheck;
   return pixels;
  }
  int[] vb = vBounds(pixels, d, Cmb, Csb);
  pixels = crop(pixels, d, 0, vb[0], d[0],
vb[1]-vb[0]+1);
  if (stage == 2) {
   timeExecution =
System.currentTimeMillis()-timeCheck;
   return pixels;
  }
  int[] pb = plateBounds(pixels, d, Cmp,
Csp);
  pixels = crop(pixels, d, pb[0], 0, pb[1]-
pb[0]+1, d[1]);
  if (stage == 3) {
   timeExecution =
System.currentTimeMillis()-timeCheck;
   return pixels;
  }
  for (int i=0; i<pixels.length; i++) {
   int k = threshold(pixels[i], Ct, true,
Csat0t, Csat1t);
   pixels[i] = (0xff000000 | k<<16 | k<<8 |
k);
  }
  if (stage == 4) {
```

```
    timeExecution =
System.currentTimeMillis()-timeCheck;
    return pixels;
   }
   BufferedImage[] si =
createSegmentedImages(pixels, d);
   if (stage == 5) {
    timeExecution =
System.currentTimeMillis()-timeCheck;
    return si;
   }
   for (BufferedImage img : si) {
    int[] dist = computeKNN(img);
    s += NUMBERS_AND_LETTERS[minIndex(dist)];
   }
   timeExecution =
System.currentTimeMillis()-timeCheck;
   timeExecutionTotal += timeExecution;
   executionCount++;
   return s;
  }

  @Override
  public void keyTyped(KeyEvent evt) {
   char key = evt.getKeyChar();
   if (key >= '0' && key <= '9') {
    int[] dist = computeKNN(fontImages[key-
'0']);
    summaryTextArea.setText("distance relatif
" + key + "\n" + distanceText(dist));
   }
   else if (key >= 'a' && key <= 'z') {
    int[] dist = computeKNN(fontImages[key-
'a'+10]);
    key = (char) (key-'a'+'A');
    summaryTextArea.setText("distance relatif
" + key + "\n" + distanceText(dist));
   }
   else if (key == ' ') {
    int[] a = new int[256];
    Arrays.fill(a, 0xffffffff);
    int[] dist = computeKNN(a, new
int[]{16,16});
    summaryTextArea.setText("distance relatif
'" + key + "'\n" + distanceText(dist));
   }
   else if (key == '\n') {
    summaryTextArea.setText(numberPlate());
    summaryTextArea.append("\nwaktu eksekusi
" + timeExecution + " ms");
    summaryTextArea.append("\nwaktu eksekusi
rata-rata " +
(timeExecutionTotal/(double)executionCount)
+ " ms");
   }
   else if (key == '.') {
    int[] d = new int[2];
    int[][] fonts = new int[36][];
    for (int i=0; i<36; i++) {
     fonts[i] =
getBrightness(getPixels(fontImages[i], d),
d);
    }
    String s =
"memory_initialization_radix=2;\nmemory_init
ialization_vector=\n";
    for (int i=0; i<576; i++) {
     for (int j=15; j>=0; j--) {
      s += fonts[i/16][(i%16)*16+j] < 128 ?
"0" : "1";
     }
     if (i<575)
      s += ",\n";
     else
      s += ";";
    }
    System.out.println(s);
   }
   else if (key == ',') {
    int[] d = new int[2];
    int[][] fonts = new int[36][];
    for (int i=0; i<36; i++) {
     fonts[i] = getPixels(fontImages[i], d);
    }
    for (int i=0; i<36; i++) {
     if (i < 10)
      System.out.println(i);
     else
      System.out.println((char)('A'+(i-10)));
     for (int j=0; j<16; j++) {
      for (int k=0; k<16; k++) {
       System.out.printf("%8x ",
fonts[i][j*16+k]);
      }
      System.out.println();
     }
     System.out.println();
    }
   }
   else if (key == '\\') {
    timeExecutionTotal = 0;
    executionCount = 0;
   }
   repaint();
  }

  @Override
  public void mouseClicked(MouseEvent evt) {
   if (!hasFocus())
    requestFocus();
   repaint();
  }

  @Override
  public void mouseEntered(MouseEvent evt) {
   if (!hasFocus())
    requestFocus();
   repaint();
  }

  @Override
  public void mouseExited(MouseEvent evt) {
   if (!hasFocus())
    requestFocus();
   repaint();
  }

  @Override
  public void mousePressed(MouseEvent evt) {
   if (!hasFocus())
    requestFocus();
   repaint();
  }

  @Override
  public void mouseReleased(MouseEvent evt) {
   if (!hasFocus())
    requestFocus();
   repaint();
  }

  @Override
  public void componentHidden(ComponentEvent
evt) {

  }

  @Override
  public void componentMoved(ComponentEvent
evt) {

  }

  @Override
```

```
 public void componentResized(ComponentEvent
evt) {
  computeBounds();
 }

 private void computeBounds() {
  prevButton.setBounds(getWidth()-
(BUTTON_WIDTH+SPACE)*1, SPACE, BUTTON_WIDTH,
BUTTON_HEIGHT);
  nextButton.setBounds(getWidth()-
(BUTTON_WIDTH+SPACE)*1,
SPACE+(BUTTON_HEIGHT+SPACE), BUTTON_WIDTH,
BUTTON_HEIGHT);
  projButton.setBounds(getWidth()-
(BUTTON_WIDTH+SPACE)*1,
SPACE+(BUTTON_HEIGHT+SPACE)*2, BUTTON_WIDTH,
BUTTON_HEIGHT);
  resetButton.setBounds(getWidth()-
(BUTTON_WIDTH+SPACE)*1,
SPACE+(BUTTON_HEIGHT+SPACE)*3, BUTTON_WIDTH,
BUTTON_HEIGHT);
  segmentButton.setBounds(getWidth()-
(BUTTON_WIDTH+SPACE)*1,
SPACE+(BUTTON_HEIGHT+SPACE)*(4+filters.lengt
h), BUTTON_WIDTH, BUTTON_HEIGHT);
  plateNumberButton.setBounds(getWidth()-
(BUTTON_WIDTH+SPACE)*1,
SPACE+(BUTTON_HEIGHT+SPACE)*(5+filters.lengt
h), BUTTON_WIDTH, BUTTON_HEIGHT);
  comboBox.setBounds(getWidth()-
(BUTTON_WIDTH+SPACE)*1,
SPACE+(BUTTON_HEIGHT+SPACE)*(6+filters.lengt
h), BUTTON_WIDTH, BUTTON_HEIGHT);
  saveImageButton.setBounds(getWidth()-
(BUTTON_WIDTH+SPACE)*1,
SPACE+(BUTTON_HEIGHT+SPACE)*(7+filters.lengt
h), BUTTON_WIDTH, BUTTON_HEIGHT);
  for (int i=0; i<filters.length; i++) {
   filters[i].button.setBounds(getWidth()-
(BUTTON_WIDTH+SPACE)*1,
SPACE+(BUTTON_HEIGHT+SPACE)*(i+4),
BUTTON_WIDTH, BUTTON_HEIGHT);
  }
  imgLabel.setBounds(0, getHeight()-40,
getWidth(), 40);
  summaryTextArea.setBounds(getWidth()-
(BUTTON_WIDTH+SPACE)*3, SPACE,
BUTTON_WIDTH*2+SPACE,
(BUTTON_HEIGHT+SPACE)*16-SPACE);
 }

 @Override
 public void componentShown(ComponentEvent
evt) {

 }

 @Override
 public void itemStateChanged(ItemEvent evt)
{
  refreshCount = 0;
  updateGraph();
 }
}
```

2. Program Modul Blok Sistem ANPR

File: anpr_module.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use work.work package.all;

entity anpr_module is
 port (
  clk_100mhz : in std_logic;
  reset : in std_logic;
        capture : in std_logic;
  pause : in std_logic;
  pause_en : in std_logic;
  next_addr : in std_logic;
  next_addr_dir : in std_logic;
  threshold_nextval : in std_logic;
  threshold_nextval_dir : in std_logic;
  ov7670_vsync : in std_logic;
  state_capture_image : out std_logic;
  cntx : out unsigned(9 downto 0);
  cnty : out unsigned(8 downto 0);
  ov7670_capture_ce_cntx : in std_logic;
  ov7670_capture_r_acc : in std_logic;
  ov7670_capture_dout1 : in
std_logic_vector(2 downto 0);
  ov7670_capture_we1 : in std_logic;
  ov7670_capture_dout2 : in std_logic;
  ov7670_capture_we2 : in std_logic;
  ov7670_capture_finished : in std_logic;
  vga_driver_addr1 : in std_logic_vector(16
downto 0);
  vga_driver_din1 : out std_logic_vector(2
downto 0);
  vga driver addr2 : in std logic vector(16
downto 0);
  vga_driver_din2 : out std_logic;
  vga_driver_addr_py1 : in
std_logic_vector(7 downto 0);
  vga_driver_din_py1 : out
std_logic_vector(8 downto 0);
  ym1 : out std_logic_vector(7 downto 0);
  y0 1 : out std logic vector(7 downto 0);
  y1_1 : out std_logic_vector(7 downto 0);
  vga_driver_addr_px1 : in
std_logic_vector(8 downto 0);
  vga_driver_din_px1 : out
std_logic_vector(7 downto 0);
  xm1 : out std_logic_vector(8 downto 0);
  xmw1 : out std_logic_vector(8 downto 0);
  x0_1 : out std_logic_vector(8 downto 0);
  x1_1 : out std_logic_vector(8 downto 0);
  vga_driver_addr_px2 : in
std_logic_vector(8 downto 0);
  vga_driver_din_px2 : out
std_logic_vector(7 downto 0);
  vga_driver_addr_py2 : in
std_logic_vector(11 downto 0);
  vga_driver_din_py2 : out
std_logic_vector(8 downto 0);
  vga_driver_ym2 : out std_logic_vector(7
downto 0);
  bounds_addr : out std_logic_vector(3
downto 0);
  bounds_count : out std_logic_vector(3
downto 0);
  vga_driver_addr_b : in std_logic_vector(3
downto 0);
  vga_driver_din_bx0 : out
std_logic_vector(8 downto 0);
  vga_driver_din_bx1 : out
std logic vector(8 downto 0);
  vga_driver_din_by0 : out
std_logic_vector(7 downto 0);
  vga_driver_din_by1 : out
std_logic_vector(7 downto 0);
```

```
  vga_driver_addr_f : in std_logic_vector(9
downto 0);
  vga_driver_din_f : out std_logic_vector(15
downto 0);
  vga driver addr c : in std logic vector(3
downto 0);
  vga_driver_din_c : out std_logic_vector(5
downto 0);
  vga_driver_addr_ci : in std_logic_vector(7
downto 0);
  vga_driver_din_ci : out
std_logic_vector(15 downto 0);
  vga_driver_addr_d : in std_logic_vector(8
downto 0);
  vga_driver_din_d : out std_logic_vector(8
downto 0);
  state : out state_type;
  ce : out std_logic;
        threshold : out std_logic_vector(9
downto 0);
        processing_time : out
std_logic_vector(31 downto 0);
        capture_time : out
std_logic_vector(31 downto 0);
        icr_btn : in std_logic
 );
end anpr_module;

architecture behavioral of anpr_module is
component image_memory is
 port (
  clka : in std_logic;
  wea : in std_logic_vector(0 downto 0);
  addra : in std_logic_vector(16 downto 0);
  dina : in std_logic_vector(2 downto 0);
  douta : out std_logic_vector(2 downto 0);
  clkb : in std_logic;
  web : in std_logic_vector(0 downto 0);
  addrb : in std_logic_vector(16 downto 0);
  dinb : in std_logic_vector(2 downto 0);
  doutb : out std_logic_vector(2 downto 0)
 );
end component;

component bin_image_memory is
 port (
  clka : in std_logic;
  wea : in std_logic_vector(0 downto 0);
  addra : in std_logic_vector(16 downto 0);
  dina : in std_logic_vector(0 downto 0);
  douta : out std_logic_vector(0 downto 0);
  clkb : in std_logic;
  web : in std_logic_vector(0 downto 0);
  addrb : in std_logic_vector(16 downto 0);
  dinb : in std_logic_vector(0 downto 0);
  doutb : out std_logic_vector(0 downto 0)
 );
end component;

component accum_bin_proj is
 port (
  b : in std_logic_vector(0 downto 0);
  clk : in std_logic;
  ce : in std_logic;
  bypass : in std_logic;
  sclr : in std_logic;
  q : out std_logic_vector(8 downto 0)
 );
end component;

component accum_area_projx is
 port (
  b : in std_logic_vector(7 downto 0);
  clk : in std_logic;
  add : in std_logic;
  ce : in std_logic;
  bypass : in std_logic;
  sclr : in std_logic;
  q : out std_logic_vector(16 downto 0)
 );
end component;

component font_memory is
 port (
  clka : in std_logic;
  addra : in std_logic_vector(9 downto 0);
  douta : out std_logic_vector(15 downto 0);
  clkb : in std_logic;
  addrb : in std_logic_vector(9 downto 0);
  doutb : out std_logic_vector(15 downto 0)
 );
end component;

component projx_memory is
 port (
  a : in std_logic_vector(8 downto 0);
  d : in std_logic_vector(7 downto 0);
  dpra : in std_logic_vector(8 downto 0);
  clk : in std_logic;
  we : in std_logic;
  qspo : out std_logic_vector(7 downto 0);
  qdpo : out std_logic_vector(7 downto 0)
 );
end component;

component projy_memory is
 port (
  a : in std_logic_vector(7 downto 0);
  d : in std_logic_vector(8 downto 0);
  dpra : in std_logic_vector(7 downto 0);
  clk : in std_logic;
  we : in std_logic;
  qspo : out std_logic_vector(8 downto 0);
  qdpo : out std_logic_vector(8 downto 0)
 );
end component;

component boundx_memory is
 port (
  a : in std_logic_vector(3 downto 0);
  d : in std_logic_vector(8 downto 0);
  dpra : in std_logic_vector(3 downto 0);
  clk : in std_logic;
  we : in std_logic;
  spo : out std_logic_vector(8 downto 0);
  dpo : out std_logic_vector(8 downto 0);
  qspo : out std_logic_vector(8 downto 0);
  qdpo : out std_logic_vector(8 downto 0)
 );
end component;

component boundy_memory is
 port (
  a : in std_logic_vector(3 downto 0);
  d : in std_logic_vector(7 downto 0);
  dpra : in std_logic_vector(3 downto 0);
  clk : in std_logic;
  we : in std_logic;
  spo : out std_logic_vector(7 downto 0);
  dpo : out std_logic_vector(7 downto 0);
  qspo : out std_logic_vector(7 downto 0);
  qdpo : out std_logic_vector(7 downto 0)
 );
end component;

component resize_memory is
 port (
  a : in std_logic_vector(7 downto 0);
  d : in std_logic_vector(16 downto 0);
  dpra : in std_logic_vector(7 downto 0);
  clk : in std_logic;
  we : in std_logic;
  qspo : out std_logic_vector(16 downto 0);
  qdpo : out std_logic_vector(16 downto 0)
```

```
 );
end component;

component line_resize_memory is
 port (
  a : in std_logic_vector(3 downto 0);
  d : in std_logic_vector(8 downto 0);
  dpra : in std_logic_vector(3 downto 0);
  clk : in std_logic;
  we : in std_logic;
  qspo : out std_logic_vector(8 downto 0);
  qdpo : out std_logic_vector(8 downto 0)
 );
end component;

component char_memory is
 port (
  a : in std_logic_vector(3 downto 0);
  d : in std_logic_vector(5 downto 0);
  dpra : in std_logic_vector(3 downto 0);
  clk : in std_logic;
  we : in std_logic;
  qspo : out std_logic_vector(5 downto 0);
  qdpo : out std_logic_vector(5 downto 0)
 );
end component;

component distance_memory is
 port (
  a : in std_logic_vector(8 downto 0);
  d : in std_logic_vector(8 downto 0);
  dpra : in std_logic_vector(8 downto 0);
  clk : in std_logic;
  we : in std_logic;
  qspo : out std_logic_vector(8 downto 0);
  qdpo : out std_logic_vector(8 downto 0)
 );
end component;

component char_images_memory is
 port (
  a : in std_logic_vector(7 downto 0);
  d : in std_logic_vector(15 downto 0);
  dpra : in std_logic_vector(7 downto 0);
  clk : in std_logic;
  we : in std_logic;
  qspo : out std_logic_vector(15 downto 0);
  qdpo : out std_logic_vector(15 downto 0)
 );
end component;

component projy_memory2 is
 port (
  clka : in std_logic;
  wea : in std_logic_vector(0 downto 0);
  addra : in std_logic_vector(11 downto 0);
  dina : in std_logic_vector(8 downto 0);
  douta : out std_logic_vector(8 downto 0);
  clkb : in std_logic;
  web : in std_logic_vector(0 downto 0);
  addrb : in std_logic_vector(11 downto 0);
  dinb : in std_logic_vector(8 downto 0);
  doutb : out std_logic_vector(8 downto 0)
 );
end component;

signal image_memory_wea : std_logic_vector(0
downto 0);
signal image_memory_addra_p,
image_memory_addra_n : std_logic_vector(16
downto 0);
signal ce_image_memory_addra,
r_image_memory_addra : std_logic;
signal image_memory_addrb :
std_logic_vector(16 downto 0);
signal image_memory_dina :
std_logic_vector(2 downto 0);
signal image_memory_douta :
std_logic_vector(2 downto 0);
signal image_memory_doutb :
std_logic_vector(2 downto 0);
signal bin image memory wea :
std_logic_vector(0 downto 0);
signal bin_image_memory_addra_p,
bin_image_memory_addra_n :
std_logic_vector(16 downto 0);
signal ce_bin_image_memory_addra,
r_bin_image_memory_addra : std_logic;
signal bin_image_memory_addrb :
std_logic_vector(16 downto 0);
signal bin_image_memory_dina :
std_logic_vector(0 downto 0);
signal bin_image_memory_douta :
std_logic_vector(0 downto 0);
signal bin_image_memory_doutb :
std_logic_vector(0 downto 0);
signal accum_bin_proj_b : std_logic_vector(0
downto 0);
signal accum_bin_proj_ce : std_logic;
signal accum_bin_proj_bypass : std_logic;
signal accum_bin_proj_sclr : std_logic;
signal accum_bin_proj_q : std_logic_vector(8
downto 0);
signal accum_area_projx_b :
std_logic_vector(7 downto 0);
signal accum_area_projx_add : std_logic;
signal accum_area_projx_ce : std_logic;
signal accum_area_projx_bypass : std_logic;
signal accum_area_projx_sclr : std_logic;
signal accum_area_projx_q :
std logic vector(16 downto 0);
signal font_memory_addra_p,
font_memory_addra_n : std_logic_vector(13
downto 0);
signal ce_font_memory_addra,
r_font_memory_addra : std_logic;
signal font_memory_addra_lat1_p,
font_memory_addra_lat1_n :
std_logic_vector(13 downto 0);
signal ce_font_memory_addra_lat1,
r_font_memory_addra_lat1 : std_logic;
signal font_memory_addrb :
std_logic_vector(9 downto 0);
signal font_memory_douta :
std_logic_vector(15 downto 0);
signal font_memory_doutb :
std_logic_vector(15 downto 0);
signal projx_memory1_a : std_logic_vector(8
downto 0);
signal projx_memory1_d : std_logic_vector(7
downto 0);
signal projx_memory1_dpra :
std_logic_vector(8 downto 0);
signal projx_memory1_we : std_logic;
signal projx_memory1_qspo :
std_logic_vector(7 downto 0);
signal projx_memory1_qdpo :
std_logic_vector(7 downto 0);
signal projy_memory1_a : std_logic_vector(7
downto 0);
signal projy_memory1_d : std_logic_vector(8
downto 0);
signal projy_memory1_dpra :
std_logic_vector(7 downto 0);
signal projy_memory1_we : std_logic;
signal projy_memory1_qspo :
std_logic_vector(8 downto 0);
signal projy_memory1_qdpo :
std_logic_vector(8 downto 0);
signal projx_memory2_a : std_logic_vector(8
downto 0);
signal projx_memory2_d : std_logic_vector(7
downto 0);
```

```
signal projx_memory2_dpra :
std_logic_vector(8 downto 0);
signal projx_memory2_we : std_logic;
signal projx_memory2_qspo :
std_logic_vector(7 downto 0);
signal projx_memory2_qdpo :
std_logic_vector(7 downto 0);
signal projy_memory2_wea :
std_logic_vector(0 downto 0);
signal projy_memory2_addra_p,
projy_memory2_addra_n : std_logic_vector(11
downto 0);
signal ce_projy_memory2_addra,
r_projy_memory2_addra : std_logic;
signal projy_memory2_addrb :
std_logic_vector(11 downto 0);
signal projy_memory2_dina :
std_logic_vector(8 downto 0);
signal projy_memory2_douta :
std_logic_vector(8 downto 0);
signal projy_memory2_doutb :
std_logic_vector(8 downto 0);
signal boundx0_memory_a : std_logic_vector(3
downto 0);
signal boundx0_memory_d : std_logic_vector(8
downto 0);
signal boundx0_memory_dpra :
std_logic_vector(3 downto 0);
signal boundx0_memory_we : std_logic;
signal boundx0_memory_spo :
std_logic_vector(8 downto 0);
signal boundx0_memory_dpo :
std_logic_vector(8 downto 0);
signal boundx0 memory qspo :
std_logic_vector(8 downto 0);
signal boundx0_memory_qdpo :
std_logic_vector(8 downto 0);
signal boundx1_memory_a : std_logic_vector(3
downto 0);
signal boundx1_memory_d : std_logic_vector(8
downto 0);
signal boundx1_memory_dpra :
std_logic_vector(3 downto 0);
signal boundx1_memory_we : std_logic;
signal boundx1_memory_spo :
std_logic_vector(8 downto 0);
signal boundx1_memory_dpo :
std_logic_vector(8 downto 0);
signal boundx1_memory_qspo :
std_logic_vector(8 downto 0);
signal boundx1_memory_qdpo :
std_logic_vector(8 downto 0);
signal boundy0_memory_a : std_logic_vector(3
downto 0);
signal boundy0_memory_d : std_logic_vector(7
downto 0);
signal boundy0_memory_dpra :
std_logic_vector(3 downto 0);
signal boundy0_memory_we : std_logic;
signal boundy0_memory_spo :
std_logic_vector(7 downto 0);
signal boundy0_memory_dpo :
std_logic_vector(7 downto 0);
signal boundy0_memory_qspo :
std_logic_vector(7 downto 0);
signal boundy0_memory_qdpo :
std_logic_vector(7 downto 0);
signal boundy1_memory_a : std_logic_vector(3
downto 0);
signal boundy1_memory_d : std_logic_vector(7
downto 0);
signal boundy1_memory_dpra :
std_logic_vector(3 downto 0);
signal boundy1_memory_we : std_logic;
signal boundy1_memory_spo :
std_logic_vector(7 downto 0);
signal boundy1_memory_dpo :
std_logic_vector(7 downto 0);
signal boundy1_memory_qspo :
std_logic_vector(7 downto 0);
signal boundy1 memory qdpo :
std_logic_vector(7 downto 0);
signal ym2_memory_a : std_logic_vector(3
downto 0);
signal ym2_memory_d : std_logic_vector(7
downto 0);
signal ym2_memory_dpra : std_logic_vector(3
downto 0);
signal ym2_memory_we : std_logic;
signal ym2_memory_spo : std_logic_vector(7
downto 0);
signal ym2_memory_dpo : std_logic_vector(7
downto 0);
signal ym2_memory_qspo : std_logic_vector(7
downto 0);
signal ym2_memory_qdpo : std_logic_vector(7
downto 0);
signal resize_memory_a : std_logic_vector(7
downto 0);
signal resize_memory_d : std_logic_vector(16
downto 0);
signal resize_memory_dpra :
std_logic_vector(7 downto 0);
signal resize_memory_we : std_logic;
signal resize_memory_qspo :
std_logic_vector(16 downto 0);
signal resize_memory_qdpo :
std_logic_vector(16 downto 0);
signal line_resize_memory_a :
std logic vector(3 downto 0);
signal line_resize_memory_d :
std_logic_vector(8 downto 0);
signal line_resize_memory_dpra :
std_logic_vector(3 downto 0);
signal line_resize_memory_we : std_logic;
signal line_resize_memory_qspo :
std_logic_vector(8 downto 0);
signal line_resize_memory_qdpo :
std_logic_vector(8 downto 0);
signal char_memory_a : std_logic_vector(3
downto 0);
signal char_memory_d : std_logic_vector(5
downto 0);
signal char_memory_dpra : std_logic_vector(3
downto 0);
signal char_memory_we : std_logic;
signal char_memory_qspo : std_logic_vector(5
downto 0);
signal char_memory_qdpo : std_logic_vector(5
downto 0);
signal distance_memory_a :
std_logic_vector(8 downto 0);
signal distance_memory_d :
std_logic_vector(8 downto 0);
signal distance_memory_dpra :
std_logic_vector(8 downto 0);
signal distance_memory_we : std_logic;
signal distance_memory_qspo :
std_logic_vector(8 downto 0);
signal distance_memory_qdpo :
std_logic_vector(8 downto 0);
signal char_images_memory_a :
std_logic_vector(7 downto 0);
signal char_images_memory_d :
std_logic_vector(15 downto 0);
signal char_images_memory_dpra :
std_logic_vector(7 downto 0);
signal char_images_memory_we : std_logic;
signal char_images_memory_qspo :
std_logic_vector(15 downto 0);
signal char_images_memory_qdpo :
std_logic_vector(15 downto 0);
signal state_p, state_n : state_type;
```

```
signal proj_addr_p, proj_addr_n :
std_logic_vector(8 downto 0);
signal ce_proj_addr, r_proj_addr :
std_logic;
signal bounds count p, bounds count n :
std_logic_vector(3 downto 0);
signal ce_bounds_count, r_bounds_count :
std_logic;
signal bounds_addr_p, bounds_addr_n :
std_logic_vector(3 downto 0);
signal ce_bounds_addr, r_bounds_addr :
std_logic;
signal bounds_addr2_p, bounds_addr2_n :
std_logic_vector(3 downto 0);
signal ce_bounds_addr2, r_bounds_addr2 :
std_logic;
signal ba_240_p, ba_240_n :
std_logic_vector(11 downto 0);
signal ce_ba_240, r_ba_240 : std_logic;
signal ba_36_p, ba_36_n : std_logic_vector(8
downto 0);
signal ce_ba_36, r_ba_36 : std_logic;
signal latch_next_addr : std_logic_vector(7
downto 0);
signal ym1_p, ym1_n : std_logic_vector(7
downto 0);
signal pym1_p, pym1_n : std_logic_vector(8
downto 0);
signal y0_1_p, y0_1_n : std_logic_vector(7
downto 0);
signal y1_1_p, y1_1_n : std_logic_vector(7
downto 0);
signal h1_p, h1_n : std_logic_vector(7
downto 0);
signal h2_p, h2_n : std_logic_vector(7
downto 0);
signal h1sq_p, h1sq_n : std_logic_vector(15
downto 0);
signal y0_1_320_p, y0_1_320_n :
std_logic_vector(16 downto 0);
signal y0_2_320_p, y0_2_320_n :
std_logic_vector(16 downto 0);
signal wmax_p, wmax_n : std_logic_vector(11
downto 0);
signal wmin_p, wmin_n : std_logic_vector(10
downto 0);
signal ce_ym1, r_ym1 : std_logic;
signal ce_y0_1, r_y0_1 : std_logic;
signal ce_y1_1, r_y1_1 : std_logic;
signal ce_h1, r_h1 : std_logic;
signal ce_h2, r_h2 : std_logic;
signal ce_h1sq, r_h1sq : std_logic;
signal ce_y0_1_320, r_y0_1_320 : std_logic;
signal ce_y0_2_320, r_y0_2_320 : std_logic;
signal ce_w, r_w : std_logic;
signal xm1_p, xm1_n : std_logic_vector(8
downto 0);
signal xmw1_p, xmw1_n : std_logic_vector(8
downto 0);
signal pxm1_p, pxm1_n : std_logic_vector(7
downto 0);
signal axm1_p, axm1_n : std_logic_vector(16
downto 0);
signal x0_1_p, x0_1_n : std_logic_vector(8
downto 0);
signal x1_1_p, x1_1_n : std_logic_vector(8
downto 0);
signal w1_p, w1_n : std_logic_vector(8
downto 0);
signal w2_p, w2_n : std_logic_vector(8
downto 0);
signal ce_xm1, r_xm1 : std_logic;
signal ce_pxm1, r_pxm1 : std_logic;
signal ce_x0_1, r_x0_1 : std_logic;
signal ce_x1_1, r_x1_1 : std_logic;
signal ce_w1, r_w1 : std_logic;
signal ce_w2, r_w2 : std_logic;
```

```
signal ym2_p, ym2_n : std_logic_vector(7
downto 0);
signal ym2_2_p, ym2_2_n : std_logic_vector(7
downto 0);
signal pym2 p, pym2 n : std logic vector(8
downto 0);
signal ce_ym2, r_ym2 : std_logic;
signal ce_ym2_2, r_ym2_2 : std_logic;
signal ff_p, ff_n : std_logic_vector(5
downto 0);
signal temp_dat1_p, temp_dat1_n :
std_logic_vector(8 downto 0);
signal temp_addr1_p, temp_addr1_n :
std_logic_vector(16 downto 0);
signal temp_dat2_p, temp_dat2_n :
std_logic_vector(8 downto 0);
signal cnty_p, cnty_n : unsigned(8 downto
0);
signal cntx_p, cntx_n : unsigned(9 downto
0);
signal cntyn_p, cntyn_n : unsigned(18 downto
0);
signal cntxn_p, cntxn_n : unsigned(9 downto
0);
signal cntyr_p, cntyr_n : unsigned(3 downto
0);
signal cntxr_p, cntxr_n : unsigned(3 downto
0);
signal ce_cntx, r_cntx : std_logic;
signal ce_cnty, r_cnty : std_logic;
signal ce_cntyn, r_cntyn : std_logic;
signal ce_cntxn, r_cntxn : std_logic;
signal ce_cntyr, r_cntyr : std_logic;
signal ce cntxr, r cntxr : std logic;
signal rx1_p, rx1_n : std_logic_vector(8
downto 0);
signal rx2_p, rx2_n : std_logic_vector(8
downto 0);
signal ry1_p, ry1_n : std_logic_vector(7
downto 0);
signal ry2_p, ry2_n : std_logic_vector(7
downto 0);
signal ce_rx1, r_rx1 : std_logic;
signal ce_rx2, r_rx2 : std_logic;
signal ce_ry1, r_ry1 : std_logic;
signal ce_ry2, r_ry2 : std_logic;
signal dm_p, dm_n : std_logic_vector(8
downto 0);
signal dc_addr_p, dc_addr_n :
std_logic_vector(5 downto 0);
signal cdm_p, cdm_n : std_logic_vector(5
downto 0);
signal ce_dm, r_dm : std_logic;
signal ce_dc_addr, r_dc_addr : std_logic;
signal status_active : std_logic_vector(0
downto 0);
signal ce_p, ce_n : std_logic;
signal latch_pause : std_logic_vector(7
downto 0);
signal threshold_p, threshold_n :
std_logic_vector(9 downto 0);
signal ce_threshold, r_threshold :
std_logic;
signal latch_threshold_nextval :
std_logic_vector(7 downto 0);
signal hold_ti_p, hold_ti_n : unsigned(25
downto 0);
signal hold_td_p, hold_td_n : unsigned(25
downto 0);
signal ce_hold_ti, r_hold_ti : std_logic;
signal ce_hold_td, r_hold_td : std_logic;
signal time_count_p, time_count_n :
std_logic_vector(31 downto 0);
signal ce_time_count_digit,
r_time_count_digit : std_logic_vector(7
downto 0);
```

```
signal processing_time_p, processing_time_n
: std_logic_vector(31 downto 0);
signal ce_processing_time, r_processing_time
: std_logic;
signal capture time p, capture time n :
std_logic_vector(31 downto 0);
signal ce_capture_time, r_capture_time :
std_logic;
begin

 inst_image_memory: image_memory port map (
  clka => clk_100mhz,
  clkb => clk_100mhz,
  wea => image_memory_wea,
  web => (others => '0'),
  addra => image_memory_addra_p,
  addrb => image_memory_addrb,
  dina => image_memory_dina,
  dinb => (others => '0'),
  douta => image_memory_douta,
  doutb => image_memory_doutb
 );

 inst_bin_image_memory: bin_image_memory
port map (
  clka => clk_100mhz,
  clkb => clk_100mhz,
  wea => bin_image_memory_wea,
  web => (others => '0'),
  addra => bin_image_memory_addra_p,
  addrb => bin_image_memory_addrb,
  dina => bin_image_memory_dina,
  dinb => (others => '0'),
  douta => bin image memory douta,
  doutb => bin_image_memory_doutb
 );

 inst_accum_bin_proj: accum_bin_proj port
map (
  b => accum_bin_proj_b,
  clk => clk_100mhz,
  ce => accum_bin_proj_ce,
  bypass => accum_bin_proj_bypass,
  sclr => accum_bin_proj_sclr,
  q => accum_bin_proj_q
 );

 inst_accum_area_projx: accum_area_projx
port map (
  b => accum_area_projx_b,
  clk => clk_100mhz,
  add => accum_area_projx_add,
  ce => accum_area_projx_ce,
  bypass => accum_area_projx_bypass,
  sclr => accum_area_projx_sclr,
  q => accum_area_projx_q
 );

 inst_font_memory: font_memory port map (
  clka => clk_100mhz,
  addra => font_memory_addra_lat1_p(13
downto 4),
  douta => font_memory_douta,
  clkb => clk_100mhz,
  addrb => font_memory_addrb,
  doutb => font_memory_doutb
 );

 inst_projx_memory1: projx_memory port map (
  a => projx_memory1_a,
  d => projx_memory1_d,
  dpra => projx_memory1_dpra,
  clk => clk_100mhz,
  we => projx_memory1_we,
  qspo => projx_memory1_qspo,
  qdpo => projx_memory1_qdpo
 );

 inst_projy_memory1: projy_memory port map (
  a => projy_memory1_a,
  d => projy_memory1_d,
  dpra => projy memory1 dpra,
  clk => clk_100mhz,
  we => projy_memory1_we,
  qspo => projy_memory1_qspo,
  qdpo => projy_memory1_qdpo
 );

 inst_projx_memory2: projx_memory port map (
  a => projx_memory2_a,
  d => projx_memory2_d,
  dpra => projx_memory2_dpra,
  clk => clk_100mhz,
  we => projx_memory2_we,
  qspo => projx_memory2_qspo,
  qdpo => projx_memory2_qdpo
 );

  inst_projy_memory2: projy_memory2 port map
(
   clka => clk_100mhz,
   clkb => clk_100mhz,
   wea => projy_memory2_wea,
   web => (others => '0'),
   addra => projy_memory2_addra_p,
   addrb => projy_memory2_addrb,
   dina => projy_memory2_dina,
   dinb => (others => '0'),
   douta => projy_memory2_douta,
   doutb => projy_memory2_doutb
  );

 inst_boundx0_memory: boundx_memory port map
(
  a => boundx0_memory_a,
  d => boundx0_memory_d,
  dpra => boundx0_memory_dpra,
  clk => clk_100mhz,
  we => boundx0_memory_we,
  spo => boundx0_memory_spo,
  dpo => boundx0_memory_dpo,
  qspo => boundx0_memory_qspo,
  qdpo => boundx0_memory_qdpo
 );

 inst_boundx1_memory: boundx_memory port map
(
  a => boundx1_memory_a,
  d => boundx1_memory_d,
  dpra => boundx1_memory_dpra,
  clk => clk_100mhz,
  we => boundx1_memory_we,
  spo => boundx1_memory_spo,
  dpo => boundx1_memory_dpo,
  qspo => boundx1_memory_qspo,
  qdpo => boundx1_memory_qdpo
 );

 inst_boundy0_memory: boundy_memory port map
(
  a => boundy0_memory_a,
  d => boundy0_memory_d,
  dpra => boundy0_memory_dpra,
  clk => clk_100mhz,
  we => boundy0_memory_we,
  spo => boundy0_memory_spo,
  dpo => boundy0_memory_dpo,
  qspo => boundy0_memory_qspo,
  qdpo => boundy0_memory_qdpo
 );

 inst_boundy1_memory: boundy_memory port map
(
  a => boundy1_memory_a,
```

```
  d => boundy1_memory_d,
  dpra => boundy1_memory_dpra,
  clk => clk_100mhz,
  we => boundy1_memory_we,
  spo => boundy1_memory_spo,
  dpo => boundy1_memory_dpo,
  qspo => boundy1_memory_qspo,
  qdpo => boundy1_memory_qdpo
 );

 inst_ym2_memory: boundy_memory port map (
  a => ym2_memory_a,
  d => ym2_memory_d,
  dpra => ym2_memory_dpra,
  clk => clk_100mhz,
  we => ym2_memory_we,
  spo => ym2_memory_spo,
  dpo => ym2_memory_dpo,
  qspo => ym2_memory_qspo,
  qdpo => ym2_memory_qdpo
 );

 inst_resize_memory: resize_memory port map
(
  a => resize_memory_a,
  d => resize_memory_d,
  dpra => resize_memory_dpra,
  clk => clk_100mhz,
  we => resize_memory_we,
  qspo => resize_memory_qspo,
  qdpo => resize_memory_qdpo
 );

 inst line resize memory: line resize memory
port map (
  a => line_resize_memory_a,
  d => line_resize_memory_d,
  dpra => line_resize_memory_dpra,
  clk => clk_100mhz,
  we => line_resize_memory_we,
  qspo => line_resize_memory_qspo,
  qdpo => line_resize_memory_qdpo
 );

 inst_char_memory: char_memory port map (
  a => char_memory_a,
  d => char_memory_d,
  dpra => char_memory_dpra,
  clk => clk_100mhz,
  we => char_memory_we,
  qspo => char_memory_qspo,
  qdpo => char_memory_qdpo
 );

 inst_distance_memory: distance_memory port
map (
  a => distance_memory_a,
  d => distance_memory_d,
  dpra => distance_memory_dpra,
  clk => clk_100mhz,
  we => distance_memory_we,
  qspo => distance_memory_qspo,
  qdpo => distance_memory_qdpo
 );

 inst_char_images_memory: char_images_memory
port map (
  a => char_images_memory_a,
  d => char_images_memory_d,
  dpra => char_images_memory_dpra,
  clk => clk_100mhz,
  we => char_images_memory_we,
  qspo => char_images_memory_qspo,
  qdpo => char_images_memory_qdpo
 );

 boundx0_memory_dpra <= vga_driver_addr_b;
 boundx1_memory_dpra <= vga_driver_addr_b;
 boundy0_memory_dpra <= vga_driver_addr_b;
 boundy1_memory_dpra <= vga_driver_addr_b;
 ym2_memory_dpra <= bounds_addr2_p;
 resize_memory_dpra <= (others => '0');
 line_resize_memory_dpra <= (others => '0');
 image_memory_addrb <= vga_driver_addr1;
 bin_image_memory_addrb <= vga_driver_addr2;
 projy_memory1_dpra <= vga_driver_addr_py1;
 projx_memory1_dpra <= vga_driver_addr_px1;
 projx_memory2_dpra <= vga_driver_addr_px2;
 projy_memory2_addrb <= vga_driver_addr_py2;
 font_memory_addrb <= vga_driver_addr_f;
 char_memory_dpra <= vga_driver_addr_c;
 char_images_memory_dpra <=
vga_driver_addr_ci;
 distance_memory_dpra <=vga_driver_addr_d;

 clock_process: process(clk_100mhz)
 begin
  if rising_edge(clk_100mhz) then
   if reset = '1' then
    latch_pause <= (others => '0');
    latch_next_addr <= (others => '0');
    latch_threshold_nextval <= (others =>
'0');
    ce_p <= '0';
   else
    latch_pause <= latch_pause(6 downto 0) &
pause;
    latch_next_addr <= latch_next_addr(6
downto 0) & next_addr;
    latch_threshold_nextval <=
latch threshold nextval(6 downto 0) &
threshold_nextval;
    ce_p <= ce_n;
   end if;
   if reset = '1' then
    state_p <= idle;
   elsif ce_p = '1' then
    state_p <= state_n;
   end if;
   if reset = '1' or r_ym1 = '1' then
    ym1_p <= (others => '0');
    pym1_p <= (others => '0');
   elsif ce_ym1 = '1' then
    ym1_p <= ym1_n;
    pym1_p <= pym1_n;
   end if;
   if reset = '1' or r_y0_1 = '1' then
    y0_1_p <= (others => '0');
   elsif ce_y0_1 = '1' then
    y0_1_p <= y0_1_n;
   end if;
   if reset = '1' or r_y1_1 = '1' then
    y1_1_p <= (others => '0');
   elsif ce_y1_1 = '1' then
    y1_1_p <= y1_1_n;
   end if;
   if reset = '1' or r_h1 = '1' then
    h1_p <= (others => '0');
   elsif ce_h1 = '1' then
    h1_p <= h1_n;
   end if;
   if reset = '1' or r_h2 = '1' then
    h2_p <= (others => '0');
   elsif ce_h2 = '1' then
    h2_p <= h2_n;
   end if;
   if reset = '1' or r_h1sq = '1' then
    h1sq_p <= (others => '0');
   elsif ce_h1sq = '1' then
    h1sq_p <= h1sq_n;
   end if;
   if reset = '1' or r_y0_1_320 = '1' then
    y0_1_320_p <= (others => '0');
   elsif ce_y0_1_320 = '1' then
```

```
 y0_1_320_p <= y0_1_320_n;
end if;
if reset = '1' or r_y0_2_320 = '1' then
 y0_2_320_p <= (others => '0');
elsif ce_y0_2_320 = '1' then
 y0_2_320_p <= y0_2_320_n;
end if;
if reset = '1' or r_w = '1' then
 wmax_p <= (others => '0');
 wmin_p <= (others => '0');
elsif ce_w = '1' then
 wmax_p <= wmax_n;
 wmin_p <= wmin_n;
end if;
if reset = '1' or r_xm1 = '1' then
 xm1_p <= (others => '0');
 xmw1_p <= "100111111";
 axm1_p <= (others => '0');
elsif ce_xm1 = '1' then
 xm1_p <= xm1_n;
 xmw1_p <= xmw1_n;
 axm1_p <= axm1_n;
end if;
if reset = '1' or r_pxm1 = '1' then
 pxm1_p <= (others => '0');
elsif ce_pxm1 = '1' then
 pxm1_p <= pxm1_n;
end if;
if reset = '1' or r_x0_1 = '1' then
 x0_1_p <= (others => '0');
elsif ce_x0_1 = '1' then
 x0_1_p <= x0_1_n;
end if;
if reset = '1' or r_x1_1 = '1' then
 x1_1_p <= (others => '0');
elsif ce_x1_1 = '1' then
 x1_1_p <= x1_1_n;
end if;
if reset = '1' or r_w1 = '1' then
 w1_p <= (others => '0');
elsif ce_w1 = '1' then
 w1_p <= w1_n;
end if;
if reset = '1' or r_w2 = '1' then
 w2_p <= (others => '0');
elsif ce_w2 = '1' then
 w2_p <= w2_n;
end if;
if reset = '1' or r_ym2 = '1' then
 ym2_p <= (others => '0');
 pym2_p <= (others => '0');
elsif ce_ym2 = '1' then
 ym2_p <= ym2_n;
 pym2_p <= pym2_n;
end if;
if reset = '1' or r_ym2_2 = '1' then
 ym2_2_p <= (others => '0');
elsif ce_ym2_2 = '1' then
 ym2_2_p <= ym2_2_n;
end if;
if reset = '1' then
 ff_p <= (others => '0');
 temp_dat1_p <= (others => '0');
 temp_addr1_p <= (others => '0');
 temp_dat2_p <= (others => '0');
elsif ce_p = '1' then
 ff_p <= ff_n;
 temp_dat1_p <= temp_dat1_n;
 temp_addr1_p <= temp_addr1_n;
 temp_dat2_p <= temp_dat2_n;
end if;
if reset = '1' or r_proj_addr = '1' then
 proj_addr_p <= (others => '0');
elsif ce_proj_addr = '1' then
 proj_addr_p <= proj_addr_n;
end if;
   if reset = '1' or r_bounds_count = '1'
then
    bounds_count_p <= (others => '0');
   elsif ce_bounds_count = '1' then
    bounds_count_p <= bounds_count_n;
   end if;
   if reset = '1' or r_bounds_addr = '1'
then
    bounds_addr_p <= (others => '0');
   elsif ce_bounds_addr = '1' then
    bounds_addr_p <= bounds_addr_n;
   end if;
   if reset = '1' or r_bounds_addr2 = '1'
then
    bounds_addr2_p <= (others => '0');
   elsif ce_bounds_addr2 = '1' then
    bounds_addr2_p <= bounds_addr2_n;
   end if;
   if reset = '1' or r_ba_240 = '1' then
    ba_240_p <= (others => '0');
   elsif ce_ba_240 = '1' then
    ba_240_p <= ba_240_n;
   end if;
   if reset = '1' or r_ba_36 = '1' then
    ba_36_p <= (others => '0');
   elsif ce_ba_36 = '1' then
    ba_36_p <= ba_36_n;
   end if;
   if reset = '1' or r_image_memory_addra =
'1' then
    image_memory_addra_p <= (others => '0');
   elsif ce_image_memory_addra = '1' then
    image_memory_addra_p <=
image_memory_addra_n;
   end if;
   if reset = '1' or
r_bin_image_memory_addra = '1' then
    bin_image_memory_addra_p <= (others =>
'0');
   elsif ce_bin_image_memory_addra = '1'
then
    bin_image_memory_addra_p <=
bin_image_memory_addra_n;
   end if;
   if reset = '1' or r_font_memory_addra =
'1' then
    font_memory_addra_p <= (others => '0');
   elsif ce_font_memory_addra = '1' then
    font_memory_addra_p <=
font_memory_addra_n;
   end if;
   if reset = '1' or
r_font_memory_addra_lat1 = '1' then
    font_memory_addra_lat1_p <= (others =>
'0');
   elsif ce_font_memory_addra_lat1 = '1'
then
    font_memory_addra_lat1_p <=
font_memory_addra_lat1_n;
   end if;
   if reset = '1' or r_projy_memory2_addra =
'1' then
    projy_memory2_addra_p <= (others =>
'0');
   elsif ce_projy_memory2_addra = '1' then
    projy_memory2_addra_p <=
projy_memory2_addra_n;
   end if;
   if reset = '1' or r_cnty = '1' then
    cnty_p <= (others => '0');
   elsif ce_cnty = '1' then
    cnty_p <= cnty_n;
   end if;
   if reset = '1' or r_cntx = '1' then
    cntx_p <= (others => '0');
   elsif ce_cntx = '1' then
    cntx_p <= cntx_n;
```

```
    end if;
    if reset = '1' or r_cntyn = '1' then
     cntyn_p <= (others => '0');
    elsif ce_cntyn = '1' then
     cntyn p <= cntyn n;
    end if;
    if reset = '1' or r_cntxn = '1' then
     cntxn_p <= (others => '0');
    elsif ce_cntxn = '1' then
     cntxn_p <= cntxn_n;
    end if;
    if reset = '1' or r_cntyr = '1' then
     cntyr_p <= (others => '0');
    elsif ce_cntyr = '1' then
     cntyr_p <= cntyr_n;
    end if;
    if reset = '1' or r_cntxr = '1' then
     cntxr_p <= (others => '0');
    elsif ce_cntxr = '1' then
     cntxr_p <= cntxr_n;
    end if;
    if reset = '1' or r_rx1 = '1' then
     rx1_p <= (others => '0');
    elsif ce_rx1 = '1' then
     rx1_p <= rx1_n;
    end if;
    if reset = '1' or r_rx2 = '1' then
     rx2_p <= (others => '0');
    elsif ce_rx2 = '1' then
     rx2_p <= rx2_n;
    end if;
    if reset = '1' or r_ry1 = '1' then
     ry1_p <= (others => '0');
    elsif ce ry1 = '1' then
     ry1_p <= ry1_n;
    end if;
    if reset = '1' or r_ry2 = '1' then
     ry2_p <= (others => '0');
    elsif ce_ry2 = '1' then
     ry2_p <= ry2_n;
    end if;
    if reset = '1' or r_dm = '1' then
     dm_p <= (others => '0');
     cdm_p <= (others => '0');
    elsif ce_dm = '1' then
     dm_p <= dm_n;
     cdm_p <= cdm_n;
    end if;
    if reset = '1' or r_dc_addr = '1' then
     dc_addr_p <= (others => '0');
    elsif ce_dc_addr = '1' then
     dc_addr_p <= dc_addr_n;
    end if;
    if reset = '1' or r_hold_ti = '1' then
     hold_ti_p <= (others => '0');
    elsif ce_hold_ti = '1' then
     hold_ti_p <= hold_ti_n;
    end if;
    if reset = '1' or r_hold_td = '1' then
     hold_td_p <= (others => '0');
    elsif ce_hold_td = '1' then
     hold_td_p <= hold_td_n;
    end if;
    if reset = '1' or r_threshold = '1' then
     threshold_p <= "0011000000";
    elsif ce_threshold = '1' then
     threshold_p <= threshold_n;
    end if;
    for i in 0 to 7 loop
     if reset = '1' or r_time_count_digit(i)
= '1' then
      time_count_p((i*4)+3 downto i*4) <=
(others => '0');
     elsif ce_time_count_digit(i) = '1' then
      time_count_p((i*4)+3 downto i*4) <=
time_count_n((i*4)+3 downto i*4);
     end if;
    end loop;
    if reset = '1' or r_processing_time = '1'
then
     processing_time_p <= (others => '0');
    elsif ce processing time = '1' then
     processing_time_p <= processing_time_n;
    end if;
    if reset = '1' or r_capture_time = '1'
then
     capture_time_p <= (others => '0');
    elsif ce_capture_time = '1' then
     capture_time_p <= capture_time_n;
    end if;
   end if;
 end process;

 next_state_process: process(capture, reset,
pause_en, latch_pause, ov7670_vsync,
ov7670_capture_finished, state_p,
proj_addr_p, latch_next_addr,
latch_threshold_nextval, next_addr_dir,
threshold_nextval_dir, threshold_p,
hold_ti_p, hold_td_p, image_memory_addrb,
cntx_p, cnty_p, cntxn_p, cntyn_p, cntxr_p,
cntyr_p, ce_p, rx1_p, rx2_p, ry1_p, ry2_p,
image_memory_addra_p,
bin_image_memory_addra_p,
font_memory_addra_p,
font_memory_addra_lat1_p,
projy_memory2_addra_p, ov7670_capture_we1,
ov7670_capture_dout1, ov7670_capture_r_acc,
ov7670_capture_we2, ov7670_capture_dout2,
ov7670_capture_ce_cntx, projx_memory1_qspo,
projx memory2 qspo, projy memory1 qspo,
projy_memory2_douta, boundx0_memory_qspo,
boundx1_memory_qspo, boundy0_memory_qspo,
boundy1_memory_qspo, ym2_memory_qspo,
resize_memory_qspo, line_resize_memory_qspo,
char_memory_qspo, distance_memory_qspo,
char_images_memory_qspo,
bin_image_memory_douta, image_memory_douta,
font_memory_douta, ym1_p, pym1_p, y0_1_p,
y1_1_p, h1_p, h2_p, h1sq_p, y0_1_320_p,
y0_2_320_p, wmax_p, wmin_p, bounds_count_p,
bounds_addr_p, bounds_addr2_p, ba_240_p,
ba_36_p, xm1_p, xmw1_p, pxm1_p, axm1_p,
x0_1_p, x1_1_p, w1_p, w2_p, ym2_p, ym2_2_p,
pym2_p, ff_p, temp_dat1_p, temp_addr1_p,
temp_dat2_p, time_count_p,
processing_time_p, accum_bin_proj_q,
accum_area_projx_q, dm_p, cdm_p, dc_addr_p,
icr_btn)
 variable bool_temp1, bool_temp2 : boolean
:= false;
 variable int_temp1, int_temp2 : integer :=
0;
 variable temp1 : std_logic_vector(11 downto
0) := (others => '0');
 begin
  --default:
  state_n <= state_p;
  accum_bin_proj_b <= (others => '0');
  accum_bin_proj_ce <= '0';
  accum_bin_proj_bypass <= '0';
  accum_bin_proj_sclr <= reset;
  accum_area_projx_b <= (others => '0');
  accum_area_projx_add <= '1';
  accum_area_projx_ce <= '0';
  accum_area_projx_bypass <= '0';
  accum_area_projx_sclr <= reset;
  image_memory_wea <= (others => '0');
  image_memory_addra_n <=
std_logic_vector(unsigned(image_memory_addra
_p)+1);
  ce_image_memory_addra <= '0';
  r_image_memory_addra <= '0';
  image_memory_dina <= (others => '0');
```

```
  bin_image_memory_wea <= (others => '0');
  bin_image_memory_addra_n <=
std_logic_vector(unsigned(bin_image_memory_a
ddra_p)+1);
  ce_bin_image_memory_addra <= '0';
  r_bin_image_memory_addra <= '0';
  bin_image_memory_dina <= (others => '0');
  font_memory_addra_n <=
std_logic_vector(unsigned(font_memory_addra_
p)+1);
  ce_font_memory_addra <= '0';
  r_font_memory_addra <= '0';
  font_memory_addra_lat1_n <=
font_memory_addra_p;
  ce_font_memory_addra_lat1 <= ce_p;
  r_font_memory_addra_lat1 <= '0';
  proj_addr_n <=
std_logic_vector(unsigned(proj_addr_p)+1);
  ce_proj_addr <= '0';
  r_proj_addr <= '0';
  bounds_count_n <=
std_logic_vector(unsigned(bounds_count_p)+1)
;
  ce_bounds_count <= '0';
  r_bounds_count <= '0';
  bounds_addr_n <=
std_logic_vector(unsigned(bounds_addr_p)+1);
  ce_bounds_addr <= '0';
  r_bounds_addr <= '0';
  bounds_addr2_n <= bounds_addr2_p;
  ce_bounds_addr2 <= '0';
  r_bounds_addr2 <= '0';
  ba_240_n <=
std_logic_vector(unsigned(ba_240_p)+240);
  ce_ba_240 <= '0';
  r_ba_240 <= '0';
  ba_36_n <=
std_logic_vector(unsigned(ba_36_p)+36);
  ce_ba_36 <= '0';
  r_ba_36 <= '0';
  projx_memory1_a <= (others => '0');
  projx_memory1_d <= (others => '0');
  projx_memory1_we <= '0';
  projy_memory1_a <= (others => '0');
  projy_memory1_d <= (others => '0');
  projy_memory1_we <= '0';
  projx_memory2_a <= (others => '0');
  projx_memory2_d <= (others => '0');
  projx_memory2_we <= '0';
  projy_memory2_wea <= (others => '0');
  projy_memory2_addra_n <=
std_logic_vector(unsigned(projy_memory2_addr
a_p)+1);
  ce_projy_memory2_addra <= '0';
  r_projy_memory2_addra <= '0';
  projy_memory2_dina <= (others => '0');
  boundx0_memory_a <= (others => '0');
  boundx0_memory_d <= (others => '0');
  boundx0_memory_we <= '0';
  boundx1_memory_a <= (others => '0');
  boundx1_memory_d <= (others => '0');
  boundx1_memory_we <= '0';
  boundy0_memory_a <= (others => '0');
  boundy0_memory_d <= (others => '0');
  boundy0_memory_we <= '0';
  boundy1_memory_a <= (others => '0');
  boundy1_memory_d <= (others => '0');
  boundy1_memory_we <= '0';
  ym2_memory_a <= (others => '0');
  ym2_memory_d <= (others => '0');
  ym2_memory_we <= '0';
  resize_memory_a <= (others => '0');
  resize_memory_d <= (others => '0');
  resize_memory_we <= '0';
  line_resize_memory_a <= (others => '0');
  line_resize_memory_d <= (others => '0');
  line_resize_memory_we <= '0';
```

```
char_memory_a <= (others => '0');
char_memory_d <= (others => '0');
char_memory_we <= '0';
distance_memory_a <= (others => '0');
distance_memory_d <= (others => '0');
distance_memory_we <= '0';
char_images_memory_a <= (others => '0');
char_images_memory_d <= (others => '0');
char_images_memory_we <= '0';
cnty_n <= cnty_p+1;
cntx_n <= cntx_p+1;
cntyn_n <= cntyn_p+320;
cntxn_n <= cntxn_p+1;
cntyr_n <= cntyr_p+1;
cntxr_n <= cntxr_p+1;
ce_cnty <= '0';
r_cnty <= '0';
ce_cntx <= '0';
r_cntx <= '0';
ce_cntyn <= '0';
r_cntyn <= '0';
ce_cntxn <= '0';
r_cntxn <= '0';
ce_cntyr <= '0';
r_cntyr <= '0';
ce_cntxr <= '0';
r_cntxr <= '0';
rx1_n <= rx1_p;
rx2_n <= rx2_p;
ry1_n <= ry1_p;
ry2_n <= ry2_p;
ce_rx1 <= '0';
r_rx1 <= '0';
ce_rx2 <= '0';
r_rx2 <= '0';
ce_ry1 <= '0';
r_ry1 <= '0';
ce_ry2 <= '0';
r_ry2 <= '0';
ym1_n <= ym1_p;
pym1_n <= pym1_p;
y0_1_n <= y0_1_p;
y1_1_n <= y1_1_p;
h1_n <= h1_p;
h2_n <= h2_p;
h1sq_n <= h1sq_p;
y0_1_320_n <= y0_1_320_p;
y0_2_320_n <= y0_2_320_p;
wmax_n <= wmax_p;
wmin_n <= wmin_p;
ce_ym1 <= '0';
r_ym1 <= '0';
ce_y0_1 <= '0';
r_y0_1 <= '0';
ce_y1_1 <= '0';
r_y1_1 <= '0';
ce_h1 <= '0';
r_h1 <= '0';
ce_h2 <= '0';
r_h2 <= '0';
ce_h1sq <= '0';
r_h1sq <= '0';
ce_y0_1_320 <= '0';
r_y0_1_320 <= '0';
ce_y0_2_320 <= '0';
r_y0_2_320 <= '0';
ce_w <= '0';
r_w <= '0';
xm1_n <= xm1_p;
xmw1_n <= xmw1_p;
pxm1_n <= pxm1_p;
axm1_n <= axm1_p;
x0_1_n <= x0_1_p;
x1_1_n <= x1_1_p;
w1_n <= w1_p;
w2_n <= w2_p;
ff_n <= (others => '0');
```

```
  temp_dat1_n <= (others => '0');
  temp_addr1_n <= (others => '0');
  temp_dat2_n <= (others => '0');
  ce_xm1 <= '0';
  r_xm1 <= '0';
  ce_pxm1 <= '0';
  r_pxm1 <= '0';
  ce_x0_1 <= '0';
  r_x0_1 <= '0';
  ce_x1_1 <= '0';
  r_x1_1 <= '0';
  ce_w1 <= '0';
  r_w1 <= '0';
  ce_w2 <= '0';
  r_w2 <= '0';
  ym2_n <= ym2_p;
  pym2_n <= pym2_p;
  ce_ym2 <= '0';
  r_ym2 <= '0';
  ym2_2_n <= ym2_2_p;
  ce_ym2_2 <= '0';
  r_ym2_2 <= '0';
  dm_n <= dm_p;
  cdm_n <= cdm_p;
  ce_dm <= '0';
  r_dm <= '0';
  dc_addr_n <=
std_logic_vector(unsigned(dc_addr_p)+1);
  ce_dc_addr <= '0';
  r_dc_addr <= '0';
  ce_n <= ce_p;
  threshold_n <= threshold_p;
  ce_threshold <= '0';
  r_threshold <= '0';
  hold_ti_n <= hold_ti_p+1;
  ce_hold_ti <= latch_threshold_nextval(0);
  r_hold_ti <= not
latch_threshold_nextval(0);
  if hold_ti_p = 5e7 then
   ce_hold_ti <= '0';
  end if;
  hold_td_n <= hold_td_p+1;
  ce_hold_td <= latch_threshold_nextval(0);
  r_hold_td <= not
latch_threshold_nextval(0);
  if hold_td_p = 5e7 then
   ce_hold_td <= '0';
  end if;
  for i in 0 to 7 loop
   time_count_n((i*4)+3 downto i*4) <=
std_logic_vector(unsigned(time_count_p((i*4)
+3 downto i*4))+1);
   ce_time_count_digit <= (others => '0');
   r_time_count_digit <= (others => '0');
  end loop;
  if state_p /= idle then
   ce_time_count_digit(0) <= ce_p;
  end if;
  if time_count_p(3 downto 0) = x"9" then
   ce_time_count_digit(1) <= ce_p;
   r_time_count_digit(0) <= ce_p;
  end if;
  if time_count_p(7 downto 0) = x"99" then
   ce_time_count_digit(2) <= ce_p;
   r_time_count_digit(1) <= ce_p;
  end if;
  if time_count_p(11 downto 0) = x"999" then
   ce_time_count_digit(3) <= ce_p;
   r_time_count_digit(2) <= ce_p;
  end if;
  if time_count_p(15 downto 0) = x"9999"
then
   ce_time_count_digit(4) <= ce_p;
   r_time_count_digit(3) <= ce_p;
  end if;
  if time_count_p(19 downto 0) = x"99999"
then
   ce_time_count_digit(5) <= ce_p;
   r_time_count_digit(4) <= ce_p;
  end if;
  if time_count_p(23 downto 0) = x"999999"
then
   ce_time_count_digit(6) <= ce_p;
   r_time_count_digit(5) <= ce_p;
  end if;
  if time_count_p(27 downto 0) = x"9999999"
then
   ce_time_count_digit(7) <= ce_p;
   r_time_count_digit(6) <= ce_p;
  end if;
  if time_count_p(31 downto 0) = x"99999999"
then
   r_time_count_digit(7) <= ce_p;
  end if;
  processing_time_n <= time_count_p;
  ce_processing_time <= '0';
  r_processing_time <= '0';
  capture_time_n <= time_count_p;
  ce_capture_time <= '0';
  r_capture_time <= '0';

  --main:
  if latch_next_addr = "00001111" then
   if next_addr_dir = '1' then
    bounds_addr2_n <=
std_logic_vector(unsigned(bounds_addr2_p)+1)
;
    if unsigned(bounds_addr2_p) = 9 then
     r_bounds_addr2 <= '1';
    else
     ce bounds addr2 <= '1';
    end if;
   else
    ce_bounds_addr2 <= '1';
    if unsigned(bounds_addr2_p) = 0 then
     bounds_addr2_n <= "1001";
    else
     bounds_addr2_n <=
std_logic_vector(unsigned(bounds_addr2_p)-
1);
    end if;
   end if;
  end if;

  if latch_threshold_nextval = "00001111" or
(hold_ti_p = 5e7 and icr_btn = '1') then
   ce_threshold <= '1';
   if threshold_nextval_dir = '1' then
    threshold_n <=
std_logic_vector(unsigned(threshold_p)+1);
   else
    threshold_n <=
std_logic_vector(unsigned(threshold_p)-1);
   end if;
  end if;

  case state_p is
   when idle =>
    if capture = '1' then
     state_n <= capture_image;
     r_time_count_digit <= (others => ce_p);
     if pause_en = '1' then
      ce_n <= '0';
     end if;
    end if;
    if ff_p(0) = '1' then
     ce_processing_time <= ce_p;
    end if;
   when capture_image =>
    if ov7670_vsync = '1' or
ov7670_capture_finished = '1' then
     r_image_memory_addra <= '1';
     r_bin_image_memory_addra <= '1';
     r_cnty <= '1';
```

```
      r_cntx <= '1';
     end if;
     if ov7670_vsync = '1' then
      r_ym1 <= '1';
      r_h1 <= '1';
      r_h1sq <= '1';
      r_proj_addr <= '1';
     end if;
     ce_cntx <= ov7670_capture_ce_cntx;
     if cntx_p = 639 and
ov7670_capture_ce_cntx = '1' then
      ce_cnty <= '1';
      r_cntx <= '1';
     end if;
     image_memory_wea(0) <=
ov7670_capture_we1;
     ce_image_memory_addra <=
ov7670_capture_we1;
     image_memory_dina <=
ov7670_capture_dout1;
     bin_image_memory_wea(0) <=
ov7670_capture_we2;
     ce_bin_image_memory_addra <=
ov7670_capture_we2;
     bin_image_memory_dina(0) <=
ov7670_capture_dout2;
     accum_bin_proj_b(0) <=
ov7670_capture_dout2;
     accum_bin_proj_ce <= ov7670_capture_we2;
     accum_bin_proj_sclr <= reset or
ov7670 vsync or ov7670 capture r acc or
ov7670_capture_finished;
     projy_memory1_we <=
ov7670 capture r acc;
     ce_proj_addr <= ov7670_capture_r_acc;
     projy_memory1_a <= proj_addr_p(7 downto
0);
     projy_memory1_d <= accum_bin_proj_q;
     ym1_n <= proj_addr_p(7 downto 0);
     pym1_n <= accum_bin_proj_q;
     bool_temp1 := unsigned(accum_bin_proj_q)
> unsigned(pym1_p);
     h1_n <=
std_logic_vector(unsigned(h1_p)+1);
     if bool_temp1 then
      y0_1_n <= proj_addr_p(7 downto 0);
      y1_1_n <= proj_addr_p(7 downto 0);
     else
      y0_1_n <= ym1_p;
      y1_1_n <= ym1_p;
     end if;
     if ov7670_capture_r_acc = '1' then
      if bool_temp1 then
       ce_ym1 <= '1';
      end if;
     end if;
     if ov7670_capture_finished = '1' then
      state_n <= y0_first;
      if pause_en = '1' then
       ce_n <= '0';
      end if;
      ce_y0_1 <= '1';
      ce_y1_1 <= '1';
      ce_h1 <= '1';
      ce_proj_addr <= '1';
      if bool_temp1 then
       proj_addr_n <=
std_logic_vector(unsigned(proj_addr_p)-1);
      else
       proj_addr_n <= "0" &
std_logic_vector(unsigned(ym1_p)-1);
      end if;
      r_cntxn <= '1';
     end if;
    when y0_first =>
     y0_1_n <=
std_logic_vector(unsigned(y0_1_p)-1);
     h1_n <=
std_logic_vector(unsigned(h1_p)+1);
     if unsigned(proj_addr_p) =
unsigned(ym1_p)-1 then
      r_time_count_digit <= (others => ce_p);
      r_time_count_digit(0) <= '0';
      ce_time_count_digit(0) <= ce_p;
      time_count_n(3 downto 0) <= "0001";
      ce_capture_time <= ce_p;
     else
      ce_y0_1 <= ce_p;
      ce_h1 <= ce_p;
     end if;
     ce_proj_addr <= ce_p;
     proj_addr_n <=
std_logic_vector(unsigned(proj_addr_p)-1);
     projy_memory1_a <= proj_addr_p(7 downto
0);
     bool_temp1 := unsigned(y0_1_p) = 0;
     bool_temp2 :=
unsigned(projy_memory1_qspo) <=
unsigned(pym1_p)/4;
     y0_1_320_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(y0_1_p))*320, 17));
     if unsigned(proj_addr_p) /=
unsigned(ym1_p)-1 and (bool_temp1 or
bool_temp2) then
      state_n <= y1_first;
      if pause_en = '1' then
       ce_n <= '0';
      end if;
      ce_y0_1 <= '0';
      ce y0 1 320 <= ce p;
      ce_h1 <= '0';
      proj_addr_n <= "0" &
std_logic_vector(unsigned(ym1_p)+1);
     end if;
    when y1_first =>
     y1_1_n <=
std_logic_vector(unsigned(y1_1_p)+1);
     h1_n <=
std_logic_vector(unsigned(h1_p)+1);
     if unsigned(proj_addr_p) /=
unsigned(ym1_p)+1 then
      ce_y1_1 <= ce_p;
      ce_h1 <= ce_p;
     end if;
     ce_proj_addr <= ce_p;
     projy_memory1_a <= proj_addr_p(7 downto
0);
     bool_temp1 := unsigned(y1_1_p) = 239;
     bool_temp2 :=
unsigned(projy_memory1_qspo) <=
unsigned(pym1_p)/4;
     cnty_n <= "0" & unsigned(y0_1_p);
     bin_image_memory_addra_n <= y0_1_320_p;
     h1sq_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(h1_p))*to_integer(unsigned(h1_p)),16));
     wmax_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(h1_p))*25/2,12));
     wmin_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(h1_p))*5,11));
     if unsigned(proj_addr_p) /=
unsigned(ym1_p)+1 and (bool_temp1 or
bool_temp2) then
      state_n <= px1;
      if pause_en = '1' then
       ce_n <= '0';
      end if;
      ce_y1_1 <= '0';
      ce_h1 <= '0';
      ce_h1sq <= ce_p;
      r_w1 <= ce_p;
```

```
      r_xm1 <= ce_p;
      r_pxm1 <= ce_p;
      ce_cnty <= ce_p;
      ce_bin_image_memory_addra <= ce_p;
      ce w <= ce p;
      r_proj_addr <= ce_p;
      ce_cntxn <= ce_p;
      if unsigned(h1_p) < 4 then
       state_n <= idle;
      end if;
     end if;
    when px1 =>
     ff_n(4 downto 1) <= ff_p(3 downto 0);
     ff_n(0) <= '0';
     ce_cnty <= ce_p;
     ce_bin_image_memory_addra <= ce_p;
     bin_image_memory_addra_n <=
std_logic_vector(unsigned(bin_image_memory_a
ddra_p)+320);
     temp_dat1_n <= temp_dat1_p;
     if cnty_p = unsigned(y1_1_p) then
      ce_cntx <= ce_p;
      ce_cntxn <= ce_p;
      cnty_n <= "0" & unsigned(y0_1_p);
      bin_image_memory_addra_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(y0_1_320_p))+to_integer(cntxn_p),17));
      ff_n(0) <= '1';
     end if;
     accum_bin_proj_b <=
bin_image_memory_douta;
     if bin_image_memory_addra_p /=
y0_1_320_p then
      accum bin proj ce <= ce p;
     end if;
     accum_area_projx_b <= accum_bin_proj_q(7
downto 0);
     projx_memory1_a <= proj_addr_p;
     projx_memory1_d <= accum_bin_proj_q(7
downto 0);
     if unsigned(proj_addr_p) <=
unsigned(wmax_p) then
      temp1 := (others => '0');
     else
      temp1 :=
std_logic_vector(unsigned(proj_addr_p)-
unsigned(wmax_p));
     end if;
     w1_n <=
std_logic_vector(unsigned(w1_p)+1);
     xm1_n <= temp1(8 downto 0);
     xmw1_n <=
std_logic_vector(unsigned(proj_addr_p)-1);
     pxm1_n <= accum_bin_proj_q(7 downto 0);
     axm1_n <= accum_area_projx_q;
     bool_temp1 :=
unsigned(accum_bin_proj_q(7 downto 0)) >
unsigned(pxm1_p);
     bool_temp2 :=
unsigned(accum_area_projx_q) >
unsigned(axm1_p);
     if ff_p(1) = '1' then
      if unsigned(ff_p(4 downto 2)) /= 0 then
       ce_cnty <= '0';
       ce_bin_image_memory_addra <= '0';
       ce_cntx <= '0';
       ce_cntxn <= '0';
       accum_bin_proj_ce <= '0';
       ff_n(1 downto 0) <= ff_p(1 downto 0);
      else
       accum_bin_proj_bypass <= '1';
       ce_proj_addr <= ce_p;
       ce_w1 <= ce_p;
       if unsigned(w1_p) = unsigned(wmax_p)
then
        ce_w1 <= '0';
       end if;
      if unsigned(proj_addr_p) >
unsigned(wmax_p) then
       temp_dat1_n <=
std_logic_vector(unsigned(temp_dat1_p)+1);
      end if;
      projx_memory1_we <= ce_p;
      accum_area_projx_ce <= ce_p;
      if bool_temp1 then
       ce_pxm1 <= ce_p;
      end if;
     end if;
    end if;
    if ff_p(2) = '1' and cntx_p >
unsigned(wmax_p) then
     projx_memory1_a <= temp_dat1_p;
    end if;
    if ff_p(3) = '1' and cntx_p >
unsigned(wmax_p) then
     accum_area_projx_b <=
projx_memory1_qspo;
     accum_area_projx_add <= '0';
     accum_area_projx_ce <= ce_p;
    end if;
    if ff_p(4) = '1' then
     if cntx_p >= unsigned(wmax_p) then
      if bool_temp2 then
       ce_xm1 <= ce_p;
      end if;
     end if;
     if cntx_p = 320 then
      r_bin_image_memory_addra <= ce_p;
      r_cnty <= ce_p;
      r_cntx <= ce_p;
      r cntxn <= ce p;
      ce_proj_addr <= ce_p;
      state_n <= x_first;
      if pause_en = '1' then
       ce_n <= '0';
      end if;
      ce_x0_1 <= ce_p;
      ce_x1_1 <= ce_p;
      if bool_temp2 then
       proj_addr_n <= temp1(8 downto 0);
      else
       proj_addr_n <= xm1_p;
      end if;
      accum_bin_proj_sclr <= ce_p or reset;
      accum_area_projx_sclr <= ce_p or
reset;
      temp_dat1_n <= (others => '0');
      ff_n <= (others => '0');
     end if;
    end if;
    if bool_temp2 then
     x0_1_n <= temp1(8 downto 0);
     x1_1_n <=
std_logic_vector(unsigned(proj_addr_p)-1);
    else
     x0_1_n <= xm1_p;
     x1_1_n <= xmw1_p;
    end if;
   when x_first =>
    ff_n(3) <= ff_p(3);
    ff_n(2) <= '1';
    ff_n(1 downto 0) <= ff_p(1 downto 0);
    x0_1_n <=
std_logic_vector(unsigned(x0_1_p)+1);
    x1_1_n <=
std_logic_vector(unsigned(x1_1_p)-1);
    w1_n <= std_logic_vector(unsigned(w1_p)-
1);
    ce_proj_addr <= ce_p;
    if ff_p(2) = '1' then
     ce_w1 <= ce_p;
     if ff_p(3) = '0' then
      ce_x0_1 <= ce_p;
     else
```

```
      ce_x1_1 <= ce_p;
     end if;
    end if;
    if ff_p(3) = '1' then
     proj addr n <=
std_logic_vector(unsigned(proj_addr_p)-1);
    end if;
    projx_memory1_a <= proj_addr_p;
    bool_temp1 :=
unsigned(projx_memory1_qspo) >=
unsigned(pxm1_p)/8;
    if ff_p(2) = '1' and bool_temp1 then
     if ff_p(3) = '0' then
      ff_n(0) <= '1';
      ce_x0_1 <= '0';
     else
      ff_n(1) <= '1';
      ce_x1_1 <= '0';
     end if;
     ce_w1 <= '0';
     if (ff_p(3) = '0' and ff_p(1) = '1') or
(ff_p(3) = '1' and ff_p(0) = '1') then
      state_n <= x_second;
      if pause_en = '1' then
       ce_n <= '0';
      end if;
      ce_cntx <= ce_p;
      ce_cnty <= ce_p;
      ce_cntxn <= ce_p;
      proj_addr_n <= x0_1_p;
      ce_image_memory_addra <= ce_p;
      ff_n <= (others => '0');
      r_bounds_count <= ce_p;
      r ba 240 <= ce p;
     end if;
    end if;
    if ff_p(2) = '1' and ((ff_p(3) = '0' and
ff_p(1) = '0') or (ff_p(3) = '1' and ff_p(0)
= '0')) then
     ff_n(3) <= not ff_p(3);
     ff_n(2) <= '0';
     if ff_p(3) = '0' then
      proj_addr_n <= x1_1_p;
     else
      proj_addr_n <= x0_1_p;
     end if;
    end if;
    bool_temp2 := unsigned(w1_p) <
unsigned(wmin_p);
    if bool_temp2 then
     state_n <= idle;
     if pause_en = '1' then
      ce_n <= '0';
     end if;
    end if;
    cntx_n <= "0" & unsigned(x0_1_p);
    cnty_n <= "0" & unsigned(y0_1_p);
    cntxn_n <= "0" & (unsigned(x0_1_p)+1);
    image_memory_addra_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(y0_1_320_p))+to_integer(unsigned(x0_1_p
)),17));
   when x_second =>
    ff_n(5) <= '0';
    ff_n(4) <= ff_p(4);
    ff_n(3 downto 1) <= ff_p(2 downto 0);
    ff_n(0) <= '0';
    ce_cnty <= ce_p;
    ce_image_memory_addra <= ce_p;
    image_memory_addra_n <=
std_logic_vector(unsigned(image_memory_addra
_p)+320);
    if cnty_p = unsigned(y1_1_p) then
     ce_cntx <= ce_p;
     ce_cntxn <= ce_p;
     cnty_n <= "0" & unsigned(y0_1_p);
     image_memory_addra_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(y0_1_320_p))+to_integer(cntxn_p),17));
     ff_n(0) <= '1';
    end if;
    accum_bin_proj_b(0) <=
image_memory_douta(2);
    if not (cntx_p = unsigned(x0_1_p) and
cnty_p = unsigned(y0_1_p)) then
     accum_bin_proj_ce <= ce_p;
    end if;
    accum_area_projx_b <= accum_bin_proj_q(7
downto 0);
    if ff_p(4) = '0' then
     accum_area_projx_bypass <= '1';
    end if;
    projx_memory2_a <= proj_addr_p;
    projx_memory2_d <= accum_bin_proj_q(7
downto 0);
    boundx0_memory_a <= bounds_count_p;
    boundx1_memory_a <= bounds_count_p;
    boundx0_memory_d <= proj_addr_p;
    boundx1_memory_d <= proj_addr_p;
    bool_temp1 := unsigned(accum_bin_proj_q)
> unsigned(h1_p)/16;
    bool_temp2 :=
unsigned(accum_area_projx_q) >
unsigned(h1sq_p)*3/32 and
unsigned(accum_area_projx_q) <=
unsigned(h1sq_p);
    if ff_p(1) = '1' then
     if unsigned(ff_p(3 downto 2)) /= 0 then
      ce_cnty <= '0';
      ce image memory addra <= '0';
      ce_cntx <= '0';
      ce_cntxn <= '0';
      accum_bin_proj_ce <= '0';
      ff_n(1 downto 0) <= ff_p(1 downto 0);
     else
      accum_bin_proj_bypass <= '1';
      ce_proj_addr <= ce_p;
      projx_memory2_we <= ce_p;
      accum_area_projx_ce <= ce_p;
      if bool_temp1 then
       if ff_p(4) = '0' then
        ff_n(4) <= '1';
        boundx0_memory_we <= ce_p;
       end if;
       boundx1_memory_we <= ce_p;
      else
       ff_n(5) <= '1';
      end if;
     end if;
    end if;
    if ff_p(2) = '1' then
     if ff_p(4) = '1' and (ff_p(5) = '1' or
unsigned(proj_addr_p) = unsigned(x1_1_p)+1)
then
      if pause_en = '1' then
       ce_n <= '0';
      end if;
      if bool_temp2 then
       state_n <= py2;
       accum_bin_proj_sclr <= ce_p or reset;
       accum_area_projx_sclr <= ce_p or
reset;
       ce_cntx <= ce_p;
       ce_cnty <= ce_p;
       ce_cntyn <= ce_p;
       ce_proj_addr <= ce_p;
       ce_projy_memory2_addra <= ce_p;
       temp_dat1_n <= proj_addr_p;
       ff_n <= (others => '0');
       ce_image_memory_addra <= ce_p;
       cntx_n <= "0" &
unsigned(boundx0_memory_qspo);
       proj_addr_n <= "0" & y0_1_p;
```

```
        projy_memory2_addra_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(ba_240_p))+to_integer(unsigned(y0_1_p))
,12));
        cnty_n <= "0" & unsigned(y0_1_p);
        cntyn_n <=
to_unsigned(to_integer(unsigned(y0_1_320_p))
+320,19);
        image_memory_addra_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(y0_1_320_p))+to_integer(unsigned(boundx
0_memory_qspo)),17));
        r_ym2 <= ce_p;
        r_h2 <= ce_p;
       else
        ff_n(4) <= '0';
       end if;
      end if;
     end if;
     if ff_p(3) = '1' and
(unsigned(proj_addr_p) = unsigned(x1_1_p)+1
or unsigned(bounds_count_p) = 10) then
      state_n <= init_res;
      if pause_en = '1' then
       ce_n <= '0';
      end if;
      r_image_memory_addra <= ce_p;
      r_cnty <= ce_p;
      r_cntx <= ce_p;
      r_cntxn <= ce_p;
      r_proj_addr <= ce_p;
      accum_bin_proj_sclr <= ce_p or reset;
      accum_area_projx_sclr <= ce_p or reset;
      resize_memory_we <= ce_p;
      line_resize_memory_we <= ce_p;
      r_bounds_addr <= ce_p;
      r_ba_36 <= ce_p;
      ff_n <= (others => '0');
      if unsigned(bounds_count_p) = 0 then
       state_n <= idle;
      end if;
     end if;
    when py2 =>
     ff_n(1) <= ff_p(0);
     ff_n(0) <= '0';
     ce_cntx <= ce_p;
     temp_dat1_n <= temp_dat1_p;
     ce_image_memory_addra <= ce_p;
     boundx0_memory_a <= bounds_count_p;
     boundx1_memory_a <= bounds_count_p;
     ym2_memory_a <= bounds_count_p;
     if cntx_p =
unsigned(boundx1_memory_qspo) then
      ce_cnty <= ce_p;
      ce_cntyn <= ce_p;
      cntx_n <= "0" &
unsigned(boundx0_memory_qspo);
      image_memory_addra_n <=
std_logic_vector(to_unsigned(to_integer(cnty
n_p)+to_integer(unsigned(boundx0_memory_qspo
)),17));
      ff_n(0) <= '1';
     end if;
     accum_bin_proj_b(0) <=
image_memory_douta(2);
     if not (cntx_p =
unsigned(boundx0_memory_qspo) and cnty_p =
unsigned(y0_1_p)) then
      accum_bin_proj_ce <= ce_p;
     end if;
     projy_memory2_dina <= accum_bin_proj_q;
     ym2_n <= proj_addr_p(7 downto 0);
     pym2_n <= accum_bin_proj_q;
     h2_n <=
std_logic_vector(unsigned(h2_p)+1);
     bool_temp1 := unsigned(accum_bin_proj_q)
> unsigned(pym2_p);
     if bool_temp1 then
      temp_dat2_n <= proj_addr_p;
      ym2_memory_d <= proj_addr_p(7 downto
0);
     else
      temp_dat2_n <= "0" & ym2_p;
      ym2_memory_d <= ym2_p;
     end if;
     if ff_p(1) = '1' then
      accum_bin_proj_bypass <= '1';
      ce_proj_addr <= ce_p;
      ce_projy_memory2_addra <= ce_p;
      projy_memory2_wea(0) <= ce_p;
      if bool_temp1 then
       ce_ym2 <= ce_p;
      end if;
      if cnty_p = unsigned(y1_1_p)+1 then
       r_image_memory_addra <= ce_p;
       ce_h2 <= ce_p;
       r_cnty <= ce_p;
       r_cntyn <= ce_p;
       r_cntx <= ce_p;
       ce_proj_addr <= ce_p;
       ce_projy_memory2_addra <= ce_p;
       ym2_memory_we <= ce_p;
       if bool_temp1 then
        proj_addr_n <=
std_logic_vector(unsigned(proj_addr_p)-1);
        projy_memory2_addra_n <=
std_logic_vector(unsigned(projy_memory2_addr
a_p)-1);
       else
        proj_addr_n <= "0" &
std_logic_vector(unsigned(ym2_p)-1);
        projy_memory2_addra_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(ba_240_p))+to_integer(unsigned(ym2_p))
- 1,12));
       end if;
       state_n <= y0_second;
       if pause_en = '1' then
        ce_n <= '0';
       end if;
       accum_bin_proj_sclr <= ce_p or reset;
       ff_n <= (others => '0');
      end if;
     end if;
    when y0_second =>
     temp_dat1_n <= temp_dat1_p;
     boundy0_memory_a <= bounds_count_p;
     temp_dat2_n <= temp_dat2_p;
     boundy0_memory_d <= temp_dat2_p(7 downto
0);
     h2_n <=
std_logic_vector(unsigned(h2_p)+1);
     if unsigned(proj_addr_p) /=
unsigned(ym2_p)-1 then
      temp_dat2_n <=
std_logic_vector(unsigned(temp_dat2_p)-1);
      ce_h2 <= ce_p;
     end if;
     ce_proj_addr <= ce_p;
     ce_projy_memory2_addra <= ce_p;
     proj_addr_n <=
std_logic_vector(unsigned(proj_addr_p)-1);
     projy_memory2_addra_n <=
std_logic_vector(unsigned(projy_memory2_addr
a_p)-1);
     bool_temp1 := unsigned(temp_dat2_p(7
downto 0)) = unsigned(y0_1_p);
     bool_temp2 :=
unsigned(projy_memory2_douta) <=
unsigned(pym2_p)/16;
     if unsigned(proj_addr_p) /=
unsigned(ym2_p)-1 and (bool_temp1 or
bool_temp2) then
      state_n <= y1_second;
```

```
      if pause_en = '1' then
       ce_n <= '0';
      end if;
      boundy0_memory_we <= ce_p;
      ce_h2 <= '0';
      temp_dat2_n <= "0" & ym2_p;
      proj_addr_n <= "0" &
std_logic_vector(unsigned(ym2_p)+1);
      projy_memory2_addra_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(ba_240_p))+to_integer(unsigned(ym2_p))
+ 1,12));
     end if;
    when y1_second =>
     temp_dat1_n <= temp_dat1_p;
     boundx1_memory_a <= bounds_count_p;
     boundy0_memory_a <= bounds_count_p;
     boundy1_memory_a <= bounds_count_p;
     temp_dat2_n <= temp_dat2_p;
     boundy1_memory_d <= temp_dat2_p(7 downto
0);
     h2_n <=
std_logic_vector(unsigned(h2_p)+1);
     if unsigned(proj_addr_p) /=
unsigned(ym2_p)+1 then
      temp_dat2_n <=
std_logic_vector(unsigned(temp_dat2_p)+1);
      ce_h2 <= ce_p;
     end if;
     ce_proj_addr <= ce_p;
     ce_projy_memory2_addra <= ce_p;
     bool_temp1 := unsigned(temp_dat2_p(7
downto 0)) = unsigned(y1_1_p);
     bool temp2 :=
unsigned(projy_memory2_douta) <=
unsigned(pym2_p)/16;
     cntx_n <= "0" & unsigned(temp_dat1_p);
     cnty_n <= "0" & unsigned(y0_1_p);
     cntxn_n <= "0" &
(unsigned(temp_dat1_p)+1);
     image_memory_addra_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(y0_1_320_p))+to_integer(unsigned(temp_d
at1_p)),17));
     if unsigned(proj_addr_p) /=
unsigned(ym2_p)+1 and (bool_temp1 or
bool_temp2) then
      state_n <= x_second;
      if pause_en = '1' then
       ce_n <= '0';
      end if;
      r_projy_memory2_addra <= ce_p;
      boundy1_memory_we <= ce_p;
      ce_h2 <= '0';
      ce_cntx <= ce_p;
      ce_cnty <= ce_p;
      ce_cntxn <= ce_p;
      ce_proj_addr <= ce_p;
      proj_addr_n <= temp_dat1_p;
      temp_dat1_n <= (others => '0');
      temp_dat2_n <= (others => '0');
      ce_image_memory_addra <= ce_p;
      ff_n <= "001000";
      if unsigned(h2_p) > unsigned(h1_p)/2
then
       ce_bounds_count <= ce_p;
       ce_ba_240 <= ce_p;
      end if;
     end if;
    when init_res =>
     resize_memory_a <=
std_logic_vector(cntyr_p & cntxr_p);
     boundy0_memory_a <= bounds_addr_p;
     boundy1_memory_a <= bounds_addr_p;
     boundx0_memory_a <= bounds_addr_p;
     boundx1_memory_a <= bounds_addr_p;
     h2_n <=
std_logic_vector(unsigned(temp_dat1_p(7
downto 0))+1);
     w2_n <=
std_logic_vector(unsigned(temp_dat2_p)+1);
     ry2_n <=
std_logic_vector(unsigned(temp_dat1_p(7
downto 0))+1);
     cnty_n <= "0" &
unsigned(boundy0_memory_qspo);
     cntyn_n <=
to_unsigned(to_integer(unsigned(y0_2_320_p))
+320,19);
     y0_2_320_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(boundy0_memory_qspo))*320, 17));
     cntx_n <= "0" &
unsigned(boundx0_memory_qspo);
     image_memory_addra_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(y0_2_320_p))+to_integer(unsigned(boundx
0_memory_qspo)),17));
     if unsigned(bounds_addr_p) =
unsigned(bounds_count_p) then
      state_n <= idle;
      ff_n(0) <= '1';
      if pause_en = '1' then
       ce_n <= '0';
      end if;
     else
      ce_cntxr <= ce_p;
      if cntxr_p = 15 then
       ce_cntyr <= ce_p;
       if cntyr p = 15 then
        state_n <= res;
       end if;
      end if;
      resize_memory_we <= ce_p;
      temp_dat1_n <= "0" &
std_logic_vector(unsigned(boundy1_memory_qsp
o)-unsigned(boundy0_memory_qspo));
      ce_h2 <= ce_p;
      temp_dat2_n <=
std_logic_vector(unsigned(boundx1_memory_qsp
o)-unsigned(boundx0_memory_qspo));
      ce_w2 <= ce_p;
      r_ry1 <= ce_p;
      ce_ry2 <= ce_p;
      ce_cnty <= ce_p;
      ce_cntyn <= ce_p;
      ce_y0_2_320 <= ce_p;
      ce_cntx <= ce_p;
      ce_image_memory_addra <= ce_p;
     end if;
    when init_l_res =>
     ce_cntxr <= ce_p;
     if cntxr_p = 15 then
      state_n <= l_res;
     end if;
     line_resize_memory_we <= ce_p;
     line_resize_memory_a <=
std_logic_vector(cntxr_p);
     r_rx1 <= ce_p;
     ce_rx2 <= ce_p;
     rx2_n <= w2_p;
    when res =>
     ce_ry1 <= ce_p;
     if unsigned(ry1_p) = 0 then
      ry1_n <= "00010000";
      state_n <= init_l_res;
     else
      ce_ry2 <= ce_p;
      if unsigned(ry1_p) < unsigned(ry2_p)
then
       ry1_n <= (others => '0');
```

```
      ry2_n <=
std_logic_vector(unsigned(ry2_p)-
unsigned(ry1_p));
      temp_dat1_n <= "0" & ry1_p;
     else
      ry1_n <=
std_logic_vector(unsigned(ry1_p)-
unsigned(ry2_p));
      ry2_n <= (others => '0');
      temp_dat1_n <= "0" & ry2_p;
     end if;
     state_n <= l_res2;
    end if;
   when l_res =>
    ff_n(0) <= ff_p(0);
    temp_dat1_n <= temp_dat1_p;
    boundx0_memory_a <= bounds_addr_p;
    line_resize_memory_a <=
std_logic_vector(cntxr_p);
    if unsigned(rx1_p) = 0 then
     ce_rx1 <= ce_p;
     rx1_n <= "000010000";
     ce_image_memory_addra <= ce_p;
     ff_n(0) <= image_memory_douta(2);
     ce_cntx <= ce_p;
    else
     ce_rx1 <= ce_p;
     ce_rx2 <= ce_p;
     if unsigned(rx1_p) < unsigned(rx2_p)
then
      rx1_n <= (others => '0');
      rx2_n <=
std_logic_vector(unsigned(rx2_p)-
unsigned(rx1 p));
      if ff_p(0) = '1' then
       temp_dat1_n <=
std_logic_vector(unsigned(temp_dat1_p)+unsig
ned(rx1_p));
      end if;
     else
      rx1_n <=
std_logic_vector(unsigned(rx1_p)-
unsigned(rx2_p));
      rx2_n <= w2_p;
      ce_cntxr <= ce_p;
      ff_n(1) <= '0';
      if cntxr_p = 15 then
       state_n <= res;
       ce_cntx <= ce_p;
       ce_cnty <= ce_p;
       ce_cntyn <= ce_p;
       ce_image_memory_addra <= ce_p;
       cntx_n <= "0" &
unsigned(boundx0_memory_qspo);
       image_memory_addra_n <=
std_logic_vector(to_unsigned(to_integer(cnty
n_p)+to_integer(unsigned(boundx0_memory_qspo
)),17));
       ff_n <= (others => '0');
      end if;
      line_resize_memory_we <= ce_p;
      line_resize_memory_d <= temp_dat1_p;
      temp_dat1_n <= (others => '0');
      if ff_p(0) = '1' then
       line_resize_memory_d <=
std_logic_vector(unsigned(temp_dat1_p)+unsig
ned(rx2_p));
      end if;
     end if;
    end if;
   when l_res2 =>
    ff_n <=
std_logic_vector(unsigned(ff_p)+1);
    temp_dat1_n <= temp_dat1_p;
    boundy0_memory_a <= bounds_addr_p;
    boundy1_memory_a <= bounds_addr_p;
    resize_memory_a <=
std_logic_vector(cntyr_p & cntxr_p);
    line_resize_memory_a <=
std_logic_vector(cntxr_p);
    temp addr1 n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(temp_dat1_p(7 downto
0)))*to_integer(unsigned(line_resize_memory_
qspo)),17));
    resize_memory_d <=
std_logic_vector(unsigned(resize_memory_qspo
)+unsigned(temp_addr1_p));
    if unsigned(ff_p) = 2 then
     ff_n <= (others => '0');
     resize_memory_we <= ce_p;
     ce_cntxr <= ce_p;
     if cntxr_p = 15 then
      state_n <= res;
      if unsigned(ry2_p) = 0 then
       ce_ry2 <= ce_p;
       ry2_n <= h2_p;
       ce_cntyr <= ce_p;
       if cntyr_p = 15 then
        state_n <= dist;
        if pause_en = '1' then
         ce_n <= '0';
        end if;
        r_image_memory_addra <= ce_p;
        r_cnty <= ce_p;
        r_cntyn <= ce_p;
        r_cntx <= ce_p;
        temp_dat1_n <= (others => '0');
        temp_addr1_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(w2_p))*to_integer(unsigned(h2_p)),17));
        r_font_memory_addra <= ce_p;
       end if;
      end if;
     end if;
    end if;
   when dist =>
    if pause_en = '1' then
     ce_n <= '0';
    end if;
    ff_n(1) <= not ff_p(1);
    temp_addr1_n <= temp_addr1_p;
    if ff_p(1) = '1' then
     ce_font_memory_addra <= ce_p;
    end if;
    temp_dat1_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(ba_36_p))+to_integer(unsigned(font_memo
ry_addra_p(13 downto 8))),9));
    resize_memory_a <= font_memory_addra_p(7
downto 0);
    distance_memory_a <= temp_dat1_p;
    distance_memory_d <=
std_logic_vector(unsigned(distance_memory_qs
po)+1);
    char_images_memory_a <= bounds_addr_p &
font_memory_addra_lat1_p(7 downto 4);
    char_images_memory_d <=
char_images_memory_qspo;

char_images_memory_d(to_integer(unsigned(fon
t_memory_addra_lat1_p(3 downto 0)))) <=
ff_p(0);
    dm_n <= distance_memory_qspo;
    cdm_n <=
std_logic_vector(unsigned(font_memory_addra_
lat1_p(13 downto 8))-1);
    if unsigned(font_memory_addra_lat1_p(7
downto 0)) = 0 then
     distance_memory_d <= "000000000";
    end if;
    char_memory_a <= bounds_addr_p;
```

```
    if unsigned(resize_memory_qspo) <
unsigned(temp_addr1_p)/2 then
     ff_n(0) <= '0';
    else
     ff n(0) <= '1';
    end if;
    if unsigned(font_memory_addra_p) /= 0
and ff_p(1) = '0' then
     if ff_p(0) = '1' xor
font_memory_douta(to_integer(unsigned(font_m
emory_addra_lat1_p(3 downto 0)))) = '1' then
      distance_memory_we <= ce_p;
      if unsigned(font_memory_addra_lat1_p(7
downto 0)) = 0 then
       distance_memory_d <= "000000001";
      end if;
     end if;
     if unsigned(font_memory_addra_lat1_p(7
downto 0)) = 0 then
      distance_memory_we <= ce_p;
     end if;
     char_images_memory_we <= ce_p;
    end if;
    if unsigned(font_memory_addra_p) = 9216
then
     state_n <= dm;
     temp_addr1_n <= (others => '0');
     temp_dat1_n <= ba_36_p;
     ff_n <= (others => '0');
     r_font_memory_addra <= ce_p;
     ce_dm <= ce_p;
     char_memory_we <= ce_p;
     r_dc_addr <= ce_p;
    end if;
   when dm =>
    ce_dc_addr <= ce_p;
    temp_dat1_n <=
std_logic_vector(unsigned(temp_dat1_p)+1);
    char_memory_a <= bounds_addr_p;
    char_memory_d <=
std_logic_vector(unsigned(dc_addr_p)-1);
    distance_memory_a <= temp_dat1_p;
    dm_n <= distance_memory_qspo;
    cdm_n <=
std_logic_vector(unsigned(dc_addr_p)-1);
    bool_temp1 :=
unsigned(distance_memory_qspo) <
unsigned(dm_p);
    if unsigned(dc_addr_p) /= 0 and
bool_temp1 then
     char_memory_we <= ce_p;
     ce_dm <= ce_p;
    end if;
    if unsigned(dc_addr_p) = 36 then
     r_dc_addr <= ce_p;
     temp_dat1_n <= (others => '0');
     ce_bounds_addr <= ce_p;
     ce_ba_36 <= ce_p;
     state_n <= init_res;
     if pause_en = '1' then
      ce_n <= '0';
     end if;
    end if;
  end case;

  if pause_en = '0' or latch_pause =
"00001111" then
   ce n <= '1';
  end if;
 end process;

 output_process: process(state_p,
threshold_p, cntx_p, cnty_p, ym1_p, y0_1_p,
y1_1_p, xm1_p, xmw1_p, x0_1_p, x1_1_p,
bounds_addr2_p, bounds_count_p, ce_p,
processing_time_p, capture_time_p,
image_memory_doutb,
bin_image_memory_doutb(0),
projy_memory1_qdpo, projx_memory1_qdpo,
projx_memory2_qdpo, projy_memory2_doutb,
ym2_memory_dpo, boundx0_memory_dpo,
boundx1_memory_dpo, boundy0_memory_dpo,
boundy1_memory_dpo, font_memory_doutb,
char_memory_qdpo, char_images_memory_qdpo,
distance_memory_qdpo)
 begin
  if state_p = capture_image then
   state_capture_image <= '0';
  else
   state_capture_image <= ce_p;
  end if;
  cntx <= cntx_p;
  cnty <= cnty_p;
  vga_driver_din1 <= image_memory_doutb;
  vga_driver_din2 <=
bin_image_memory_doutb(0);
  vga driver din py1 <= projy memory1 qdpo;
  ym1 <= ym1_p;
  y0_1 <= y0_1_p;
  y1_1 <= y1_1_p;
  vga_driver_din_px1 <= projx_memory1_qdpo;
  xm1 <= xm1_p;
  xmw1 <= xmw1_p;
  x0_1 <= x0_1_p;
  x1_1 <= x1_1_p;
  vga_driver_din_px2 <= projx_memory2_qdpo;
  vga_driver_din_py2 <= projy_memory2_doutb;
  vga_driver_ym2 <= ym2_memory_dpo;
  bounds_addr <= bounds_addr2_p;
  bounds_count <= bounds_count_p;
  vga_driver_din_bx0 <= boundx0_memory_dpo;
  vga_driver_din_bx1 <= boundx1_memory_dpo;
  vga_driver_din_by0 <= boundy0_memory_dpo;
  vga_driver_din_by1 <= boundy1_memory_dpo;
  vga_driver_din_f <= font_memory_doutb;
  vga_driver_din_c <= char_memory_qdpo;
  vga_driver_din_ci <=
char_images_memory_qdpo;
  vga_driver_din_d <= distance_memory_qdpo;
  state <= state_p;
  ce <= ce_p;
  threshold <= threshold_p;
  processing_time <= processing_time_p;
  capture_time <= capture_time_p;
 end process;
end behavioral;
```

3. Program Modul Blok Antarmuka Kamera-Memori

File: ov7670_capture.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric std.all;

entity ov7670_capture is
 port (
  clk_100mhz : in std_logic;
  pclk : in std_logic;
  reset : in std_logic;
  vsync : in std_logic;
  href : in std logic;
  capture : in std_logic;
  threshold : in std_logic_vector(9 downto
0);
  din : in std_logic_vector(7 downto 0);
  cntx : in unsigned(9 downto 0);
```

```
  cnty : in unsigned(8 downto 0);
  ce_cntx : out std_logic;
  r_acc : out std_logic;
  dout1 : out std_logic_vector(2 downto 0);
  we1 : out std logic;
  dout2 : out std_logic;
  we2 : out std_logic;
  finished : out std_logic;
  pixel_clock : out std_logic_vector(7
downto 0)
 );
end ov7670_capture;

architecture behavioral of ov7670_capture is
component shift_ram_638_8bit is
 port (
  d : in std_logic_vector(7 downto 0);
  clk : in std_logic;
  ce : in std_logic;
  sclr : in std_logic;
  q : out std_logic_vector(7 downto 0)
 );
end component;

component simple_edge_detect is
 port (
  clk : in std_logic;
  reset : in std_logic;
  threshold : in std_logic_vector(9 downto
0);
  t_en : in std_logic;
  op1 : in std_logic_vector(9 downto 0);
  op2 : in std_logic_vector(9 downto 0);
  en : in std logic;
  r : out std_logic
 );
end component;

signal dout1_p, dout1_n : std_logic_vector(9
downto 0);
signal we1_p, we1_n : std_logic;
signal ce_dout1 : std_logic;
signal dout1_1_p, dout1_1_n :
std_logic_vector(9 downto 0);
signal dout1_2_p, dout1_2_n :
std_logic_vector(9 downto 0);
signal dout2_p, dout2_n : std_logic;
signal we2_p, we2_n : std_logic;
signal ce_dout2 : std_logic;
signal ce_cntx_p, ce_cntx_n : std_logic;
signal ce_shiftreg_p, ce_shiftreg_n :
std_logic;
signal pclk_div_p, pclk_div_n : std_logic;
signal latch_pclk : std_logic_vector(1
downto 0);
signal latch_vsync : std_logic_vector(1
downto 0);
signal din0_p, din0_n : std_logic_vector(7
downto 0);
signal din1_p, din1_n : std_logic_vector(7
downto 0);
signal din2_p, din2_n : std_logic_vector(7
downto 0);
signal din3_p, din3_n : std_logic_vector(7
downto 0);
signal shift_ram_d : std_logic_vector(7
downto 0);
signal shift_ram_q : std_logic_vector(7
downto 0);
signal shift_ram_sclr : std_logic;
signal en1_p, en1_n : std_logic;
signal en2_p, en2_n : std_logic;
signal en3_p, en3_n : std_logic_vector(1
downto 0);
signal en4_p, en4_n : std_logic;
signal r : std_logic_vector(1 downto 0);
signal r_acc_p, r_acc_n : std_logic;
signal finished_p, finished_n : std_logic;
type state_type is (inactive, active);
signal state_p, state_n : state_type;
signal t_en : std_logic;
signal pc count p, pc count n :
std_logic_vector(7 downto 0);
signal ce_pc_count_digit, r_pc_count_digit :
std_logic_vector(1 downto 0);
signal pc_p, pc_n : std_logic_vector(7
downto 0);
signal ce_pc, r_pc : std_logic;
begin

 inst_shift_ram_638_8bit: shift_ram_638_8bit
port map (
  d => shift_ram_d,
  clk => clk_100mhz,
  ce => ce_shiftreg_p,
  sclr => reset,
  q => shift_ram_q
 );

 inst_simple_edge_detect1:
simple_edge_detect port map (
  clk => clk_100mhz,
  reset => reset,
  threshold => threshold,
  t_en => t_en,
  op1 => dout1_1_p,
  op2 => dout1_p,
  en  => en3_p(0),
  r => r(0)
 );

 inst_simple_edge_detect2:
simple_edge_detect port map (
  clk => pclk,
  reset => reset,
  threshold => threshold,
  t_en => t_en,
  op1 => dout1_1_p,
  op2 => dout1_2_p,
  en  => en3_p(1),
  r => r(1)
 );

 clock_process1: process(clk_100mhz)
 begin
  if rising_edge(clk_100mhz) then
   if reset = '1' then
    state_p <= inactive;
    finished_p <= '0';
    latch_pclk <= (others => '0');
    latch_vsync <= (others => '0');
   else
    state_p <= state_n;
    finished_p <= finished_n;
    latch_pclk <= latch_pclk(0) & pclk;
    latch_vsync <= latch_vsync(0) & vsync;
   end if;
  end if;
 end process;

 clock_process2: process(clk_100mhz)
 begin
  if rising_edge(clk_100mhz) then
   if reset = '1' or vsync = '1' then
    we1_p <= '0';
    we2_p <= '0';
    ce_cntx_p <= '0';
    ce_shiftreg_p <= '0';
    r_acc_p <= '0';
    pclk_div_p <= '0';
    din0_p <= (others => '0');
    din1_p <= (others => '0');
    din2_p <= (others => '0');
    din3_p <= (others => '0');
```

```
    dout1_p <= (others => '0');
    dout1_1_p <= (others => '0');
    dout1_2_p <= (others => '0');
    dout2_p <= '0';
    en1_p <= '0';
    en2_p <= '0';
    en3_p <= (others => '0');
    en4_p <= '0';
   else
    we1_p <= we1_n;
    we2_p <= we2_n;
    ce_cntx_p <= ce_cntx_n;
    ce_shiftreg_p <= ce_shiftreg_n;
    r_acc_p <= r_acc_n;
    pclk_div_p <= pclk_div_n;
    en1_p <= en1_n;
    en2_p <= en2_n;
    en3_p <= en3_n;
    en4_p <= en4_n;
    if ce_shiftreg_p = '1' then
     din0_p <= din0_n;
     din1_p <= din1_n;
     din2_p <= din2_n;
     din3_p <= din3_n;
    end if;
    if ce_dout1 = '1' then
     dout1_p <= dout1_n;
    end if;
    if ce_dout2 = '1' then
     dout2_p <= dout2_n;
    end if;
    if ce_dout1 = '1' or ce_dout2 = '1' then
     dout1_1_p <= dout1_1_n;
     dout1_2_p <= dout1_2_n;
    end if;
   end if;
   for i in 0 to 1 loop
    if reset = '1' or r_pc_count_digit(i) =
'1' then
     pc_count_p((i*4)+3 downto i*4) <=
(others => '0');
    elsif ce_pc_count_digit(i) = '1' then
     pc_count_p((i*4)+3 downto i*4) <=
pc_count_n((i*4)+3 downto i*4);
    end if;
   end loop;
   if reset = '1' or r_pc = '1' then
    pc_p <= (others => '0');
   elsif ce_pc = '1' then
    pc_p <= pc_n;
   end if;
  end if;
 end process;

 next_state_process: process(dout1_p, we1_p,
dout2_p, we2_p, href, latch_vsync,
latch_pclk, r, r_acc_p, din0_p, din1_p,
din2_p, din3_p, dout1_1_p, dout1_2_p, din,
capture, ce_cntx_p, ce_shiftreg_p, state_p,
shift_ram_q, pclk_div_p, en1_p, en2_p,
en3_p, en4_p, finished_p, cnty, cntx,
pc_count_p, pc_p)
 begin
  --default:
  din0_n <= din;
  din1_n <= din0_p;
  shift_ram_d <= din1_p;
  din2_n <= shift_ram_q;
  din3_n <= din2_p;
  dout1_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(din0_p))+to_integer(unsigned(din1_p))+t
o_integer(unsigned(din2_p))+to_integer(unsig
ned(din3_p)),10));
  we1_n <= '0';
  dout1_1_n <= dout1_p;
  dout1_2_n <= dout1_1_p;
  dout2_n <= r(0) or r(1);
  we2_n <= '0';
  en1_n <= en1_p;
  en2_n <= en2_p;
  en3_n <= en3_p;
  en4_n <= en4_p;
  ce_cntx_n <= '0';
  ce_shiftreg_n <= '0';
  pclk_div_n <= '0';
  state_n <= state_p;
  r_acc_n <= '0';
  finished_n <= finished_p;
  ce_dout1 <= '0';
  ce_dout2 <= '0';
  if state_p = active then
   t_en <= '0';
  else
   t_en <= '1';
  end if;
  for i in 0 to 1 loop
   pc_count_n((i*4)+3 downto i*4) <=
std_logic_vector(unsigned(pc_count_p((i*4)+3
downto i*4))+1);
   ce_pc_count_digit <= (others => '0');
   r_pc_count_digit <= (others => '0');
  end loop;
  ce_pc_count_digit(0) <= '1';
  if pc_count_p(3 downto 0) = x"9" then
   ce_pc_count_digit(1) <= '1';
   r_pc_count_digit(0) <= '1';
  end if;
  if pc_count_p(7 downto 0) = x"99" then
   r_pc_count_digit(1) <= '1';
  end if;
  pc_n <= pc_count_p;
  ce_pc <= '0';
  r_pc <= '0';

  --main:
  if latch_pclk = "10" then
   r_pc_count_digit <= (others => '1');
   r_pc_count_digit(0) <= '0';
   pc_count_n(3 downto 0) <= "0001";
   ce_pc <= '1';
  end if;

  if latch_vsync = "01" then
   if capture = '1' then
    state_n <= active;
   else
    state_n <= inactive;
   end if;
   finished_n <= '0';
  end if;

  if state_p = active and (href = '1' or
en4_p = '1') then
   pclk_div_n <= pclk_div_p;
   if latch_pclk = "10" then
    pclk_div_n <= not pclk_div_p;
    ce_shiftreg_n <= pclk_div_p;
   end if;
   ce_cntx_n <= ce_shiftreg_p;
  end if;

  if ce_cntx_p = '1' then
   if cnty = 1 then
    if cntx = 1 then
     en1_n <= '1';
    end if;
    if cntx = 4 then
     en2_n <= '1';
    end if;
   end if;
   if cnty(0) = '1' then
    if cntx = 4 then
     en3_n <= "01";
```

```
     if en2_p = '1' then
      r_acc_n <= '1';
     end if;
    end if;
    if cntx = 6 then
     en3_n <= "11";
    end if;
    if cntx = 2 then
     en3_n <= "10";
    end if;
   end if;
   if cnty = 479 then
    en4_n <= '1';
   end if;
   if cnty = 480 then
    if cntx = 0 then
     en1_n <= '0';
    end if;
    if cntx = 2 then
     en3_n <= "10";
    end if;
    if cntx = 4 then
     en2_n <= '0';
     en3_n <= "00";
     en4_n <= '0';
     r_acc_n <= '1';
     finished_n <= '1';
    end if;
   end if;
  end if;
```

```
  if ce_shiftreg_p = '1' then
   if cntx(0) = '0' and (cnty(0) = '1' or
cnty = 480) then
    if en1_p = '1' then
     ce_dout1 <= '1';
     we1_n <= '1';
    end if;
    if en2_p = '1' then
     ce_dout2 <= '1';
     we2_n <= '1';
    end if;
   end if;
  end if;
 end process;

 output_process: process(ce_cntx_p, r_acc_p,
dout1_p, we1_p, dout2_p, we2_p, finished_p,
pc_p)
 begin
  ce_cntx <= ce_cntx_p;
  r_acc <= r_acc_p;
  dout1 <= dout1_p(9 downto 7);
  we1 <= we1_p;
  dout2 <= dout2_p;
  we2 <= we2_p;
  finished <= finished_p;
  pixel_clock <= pc_p;
 end process;
end behavioral;
```

4. Program Modul Blok Kontroler Kamera

File: ov7670_controller.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity ov7670_controller is
 port (
  clk 100mhz : in std logic;
  sccb_busy : in std_logic;
  start_config : in std_logic;
  reset : in std_logic;
  send : out std_logic;
  rw : out std_logic;
  addr : out std_logic_vector(7 downto 0);
  data : out std_logic_vector(7 downto 0);
  busy : out std_logic;
  config_finished : out std_logic
 );
end ov7670_controller;

architecture behavioral of ov7670_controller
is
signal send_p, send_n : std_logic;
signal rw_p, rw_n : std_logic;
signal addr_p, addr_n : std_logic_vector(7
downto 0);
signal data_p, data_n : std_logic_vector(7
downto 0);
signal busy_p, busy_n : std_logic;
signal config_finished_p, config_finished_n
: std_logic;
signal latch_sccb_busy : std_logic;
signal delay_cnt_p, delay_cnt_n :
unsigned(16 downto 0);
signal ce_delay_cnt : std_logic;
signal r_delay_cnt : std_logic;
signal exec cnt p, exec cnt n : natural;
signal ce_exec_cnt : std_logic;
signal ce_addr_data : std_logic;
type config_mem_type is array(natural range
<>) of std_logic_vector(15 downto 0);
constant config_mem : config_mem_type := (
```

```
 --software reset register
 x"12" & "10000000",
 --set resolusi vga
 x"32" & "11110110",
 x"17" & "00010011",
 x"18" & "00000001",
 x"19" & "00000010",
 x"1a" & "01111010",
 x"03" & "00001010",
 --set format warna yuv
 x"04" & "00000000",
 x"14" & "00111000",
 x"4f" & "01000000",
 x"50" & "00110100",
 x"51" & "00001100",
 x"52" & "00010111",
 x"53" & "00101001",
 x"54" & "01000000",
 x"3d" & "11000000",
 x"3a" & "00001100",
 x"1e" & "00000111",
 x"74" & "00010000",
 x"b1" & "00001100",
 x"b3" & "10000010",
 x"0c" & "00000000",
 x"3e" & "00000000",
 x"58" & "00011110",
 x"11" & "00000000",
 x"0e" & "01100001",
 x"0f" & "01001011",
 x"16" & "00000010",
 x"21" & "00000010",
 x"22" & "10010001",
 x"29" & "00000111",
 x"33" & "00001011",
 x"35" & "00001011",
 x"37" & "00011101",
 x"38" & "01110001",
 x"39" & "00101010",
 x"3c" & "01111000",
 x"4d" & "01000000",
```

```
 x"4e" & "00100000",
 x"69" & "00000000",
 x"74" & "00010000",
 x"8d" & "01001111",
 x"8e" & "00000000",
 x"8f" & "00000000",
 x"90" & "00000000",
 x"91" & "00000000",
 x"96" & "00000000",
 x"9a" & "00000000",
 x"b0" & "10000100",
 x"b2" & "00001110",
 x"b8" & "00001010"
);
begin

 clock_process1: process(clk_100mhz)
 begin
  if rising_edge(clk_100mhz) then
   if reset = '1' then
    send_p <= '0';
    rw_p <= '0';
    addr_p <= (others => '0');
    data_p <= (others => '0');
    busy_p <= '0';
    config_finished_p <= '0';
    exec_cnt_p <= 0;
   else
    send_p <= send_n;
    rw_p <= rw_n;
    if ce_addr_data = '1' then
     addr_p <= addr_n;
     data_p <= data_n;
    end if;
    busy_p <= busy_n;
    config_finished_p <= config_finished_n;
    if ce_exec_cnt = '1' then
     exec_cnt_p <= exec_cnt_n;
    end if;
   end if;
   latch_sccb_busy <= sccb_busy;
  end if;
 end process;

 clock_process2: process(clk_100mhz)
 begin
  if rising_edge(clk_100mhz) then
   if reset = '1' or r_delay_cnt = '1' then
    delay_cnt_p <= (others => '0');
   elsif ce_delay_cnt = '1' then
    delay_cnt_p <= delay_cnt_n;
   end if;
  end if;
 end process;

 next_state_process: process(send_p, rw_p,
addr_p, data_p, busy_p, config_finished_p,
sccb_busy, start_config, delay_cnt_p,
exec_cnt_p, latch_sccb_busy)
 begin
  --default:
  send_n <= send_p;
  rw_n <= rw_p;
  addr_n <= config_mem(exec_cnt_p)(15 downto
8);
  data_n <= config_mem(exec_cnt_p)(7 downto
0);
  busy_n <= busy_p;
  config_finished_n <= config_finished_p;
  delay_cnt_n <= delay_cnt_p + 1;
  exec_cnt_n <= exec_cnt_p + 1;
  ce_delay_cnt <= '0';
  r_delay_cnt <= '0';
  ce_exec_cnt <= '0';
  ce_addr_data <= '0';

  --main:
  if busy_p = '0' then
   if start_config = '1' then
    config_finished_n <= '0';
    busy_n <= '1';
   end if;
  else
   if sccb_busy = '0' then
    if exec_cnt_p /= config_mem'length-1
then
     if exec_cnt_p = 1 and delay_cnt_p /=
99999 then
      ce_delay_cnt <= '1';
     else
      send_n <= '1';
      rw_n <= '0';
      ce_addr_data <= '1';
     end if;
    else
     config_finished_n <= '1';
     busy_n <= '0';
    end if;
   else
    if latch_sccb_busy = '0' then
     ce_exec_cnt <= '1';
     r_delay_cnt <= '1';
    end if;
    send_n <= '0';
   end if;
  end if;
 end process;

 output_process: process(send_p, rw_p,
addr_p, data_p, busy_p, config_finished_p)
 begin
  send <= send_p;
  rw <= rw_p;
  addr <= addr_p;
  data <= data_p;
  busy <= busy_p;
  config_finished <= config_finished_p;
 end process;
end behavioral;
```

5. Program Modul Blok Antarmuka SCCB

File: ov7670_sccb.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity ov7670_sccb is
 port (
  clk_100mhz : in std_logic;
  reset : in std_logic;
  pwdn_trig : in std_logic;
  send : in std_logic;
  rw : in std_logic;
  addr : in std_logic_vector(7 downto 0);
  din : in std_logic_vector(7 downto 0);
  dout : out std_logic_vector(7 downto 0);
  sioc : out std_logic;
  siod : inout std_logic;
  busy : out std logic;
  pwdn : out std_logic
 );
end ov7670_sccb;

architecture behavioral of ov7670_sccb is
```

```
signal cnt1k_p, cnt1k_n : unsigned(9 downto
0);
signal cntc_p, cntc_n : unsigned(3 downto
0);
signal pwdn_p, pwdn_n : std_logic;
signal sioc_p, sioc_n : std_logic;
signal siod_p, siod_n : std_logic;
signal busy_p, busy_n : std_logic;
signal dout_p, dout_n : std_logic_vector(7
downto 0);
signal sreg_p, sreg_n : std_logic_vector(6
downto 0);
signal latch_pwdn_trig : std_logic;
signal buff_addr_p, buff_addr_n :
std_logic_vector(7 downto 0);
signal buff_din_p, buff_din_n :
std_logic_vector(7 downto 0);
signal buff_rw_p, buff_rw_n : std_logic;
type state_type is (idle, start, id,
address, data, stop, suspend);
signal state_p, state_n : state_type;
type transmission_type is (tr_write,
tr_read);
signal tr_p, tr_n : transmission_type;
constant ov7670_id : std_logic_vector(6
downto 0) := "0100001";
begin

 clock_process: process(clk_100mhz)
 begin
  if rising_edge(clk_100mhz) then
   if reset = '1' then
    cnt1k_p <= (others => '0');
    dout_p <= (others => '0');
    sreg_p <= (others => '0');
    cntc_p <= (others => '0');
    sioc_p <= '1';
    siod_p <= 'z';
    state_p <= idle;
    busy_p <= '0';
    buff_addr_p <= (others => '0');
    buff_din_p <= (others => '0');
    buff_rw_p <= '0';
    tr_p <= tr_write;
   else
    cnt1k_p <= cnt1k_n;
    dout_p <= dout_n;
    sreg_p <= sreg_n;
    cntc_p <= cntc_n;
    sioc_p <= sioc_n;
    siod_p <= siod_n;
    state_p <= state_n;
    busy_p <= busy_n;
    buff_addr_p <= buff_addr_n;
    buff_din_p <= buff_din_n;
    buff_rw_p <= buff_rw_n;
    tr_p <= tr_n;
   end if;
   latch_pwdn_trig <= pwdn_trig;
   pwdn_p <= pwdn_n;
  end if;
 end process;

 next_state_process:
process(latch_pwdn_trig, pwdn_trig, pwdn_p,
sioc_p, siod_p, state_p, busy_p, send, tr_p,
siod, dout_p, cntc_p, buff_addr_p,
buff_din_p, buff_rw_p, cnt1k_p, addr, din,
rw, sreg_p)
 begin
  --default:
  if cnt1k_p = 999 then
   cnt1k_n <= (others => '0');
  else
   cnt1k_n <= cnt1k_p + 1;
  end if;
  dout_n <= dout_p;
  sreg_n <= sreg_p;
  cntc_n <= cntc_p;
  pwdn_n <= '0';
  sioc_n <= sioc_p;
  siod_n <= siod_p;
  state_n <= state_p;
  busy_n <= busy_p;
  buff_addr_n <= buff_addr_p;
  buff_din_n <= buff_din_p;
  buff_rw_n <= buff_rw_p;
  tr_n <= tr_p;

  --main:
  case state_p is
   when idle =>
    if send = '1' and busy_p = '0' then
     siod_n <= '1';
     buff_addr_n <= addr;
     if rw = '0' then
      buff_din_n <= din;
     end if;
     buff_rw_n <= rw;
     state_n <= start;
     busy_n <= '1';
    end if;
    cnt1k_n <= (others => '0');
   when start =>
    if cnt1k_p = 249 then
     siod_n <= '0';
    end if;
    if cnt1k_p = 499 then
     sioc_n <= '0';
     state_n <= id;
     cntc_n <= (others => '0');
    end if;
   when id =>
    if cnt1k_p = 749 then
     case cntc_p is
      when "1000" =>
       siod_n <= '0';
      when "0111" =>
       if tr_p = tr_read then
        siod_n <= '1';
       else
        siod_n <= '0';
       end if;
      when others =>
       siod_n <= ov7670_id(to_integer(6-
cntc_p));
     end case;
    end if;
    if cnt1k_p = 999 then
     sioc_n <= '1';
     cntc_n <= cntc_p + 1;
    end if;
    if cnt1k_p = 499 then
     sioc_n <= '0';
     if cntc_p = 9 then
      cntc_n <= (others => '0');
      if tr_p = tr_write then
       state_n <= address;
      else
       state_n <= data;
       siod_n <= 'z';
      end if;
     end if;
    end if;
   when address =>
    if cnt1k_p = 749 then
     if cntc_p = 8 then
      siod_n <= '0';
     else
      siod_n <= buff_addr_p(to_integer(7-
cntc_p));
     end if;
    end if;
    if cnt1k_p = 999 then
```

```
      sioc_n <= '1';
      cntc_n <= cntc_p + 1;
     end if;
     if cnt1k_p = 499 then
      sioc n <= '0';
      if cntc_p = 9 then
       cntc_n <= (others => '0');
       if buff_rw_p = '1' then
        state_n <= stop;
       else
        state_n <= data;
       end if;
      end if;
     end if;
    when data =>
     if cnt1k_p = 749 and tr_p = tr_write
then
      if cntc_p = 8 then
       siod_n <= '0';
      else
       siod_n <= buff_din_p(to_integer(7-
cntc_p));
      end if;
     end if;
     if cnt1k_p = 999 then
      if tr_p = tr_read and cntc_p = 8 then
       siod_n <= '1';
      end if;
      sioc_n <= '1';
      cntc_n <= cntc_p + 1;
     end if;
     if cnt1k_p = 249 then
      if buff_rw_p = '1' and cntc_p /= 9 then
       if cntc p = 8 then
        dout_n <= sreg_p(6 downto 0) & siod;
       end if;
       sreg_n <= sreg_p(5 downto 0) & siod;
      end if;
     end if;
     if cnt1k_p = 499 then
      sioc_n <= '0';
      if cntc_p = 9 then
       cntc_n <= (others => '0');
       state_n <= stop;
      end if;
     end if;
    when stop =>
     if cnt1k_p = 749 then
      siod_n <= '0';
     end if;
     if cnt1k_p = 999 then
      sioc_n <= '1';
     end if;
     if cnt1k_p = 249 then
      siod_n <= '1';
     end if;
     if cnt1k_p = 499 then
      if buff_rw_p = '1' and tr_p = tr_write
then
       tr_n <= tr_read;
       state_n <= start;
       cnt1k_n <= (others => '0');
      else
       tr n <= tr write;
       busy_n <= '0';
       siod_n <= 'z';
       state_n <= idle;
      end if;
     end if;
    when others =>
     pwdn_n <= '1';
     if latch_pwdn_trig = '1' and pwdn_trig =
'0' then
      sioc_n <= '1';
      siod_n <= 'z';
      cnt1k_n <= (others => '0');
     end if;
     if latch_pwdn_trig = '1' and cnt1k_p = 9
then
      sioc_n <= '0';
      siod_n <= '0';
      dout_n <= (others => '0');
      sreg_n <= (others => '0');
      tr_n <= tr_write;
     end if;
     if latch_pwdn_trig = '0' and cnt1k_p = 9
then
      pwdn_n <= '0';
      if busy_p = '0' then
       state_n <= idle;
      else
       state_n <= start;
       siod n <= '1';
      end if;
      cnt1k_n <= (others => '0');
     end if;
   end case;

   if latch_pwdn_trig = '0' and pwdn_trig =
'1' then
    pwdn_n <= '1';
    cnt1k_n <= (others => '0');
    cntc_n <= (others => '0');
    state_n <= suspend;
   end if;
 end process;

 output_process: process(pwdn_p, sioc_p,
siod_p, busy_p, dout_p)
 begin
  pwdn <= pwdn_p;
  sioc <= sioc_p;
  siod <= siod_p;
  busy <= busy_p;
  dout <= dout_p;
 end process;
end behavioral;
```

6. Program Modul Blok Deteksi Tepi Vertikal

File: simple_edge_detect.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity simple_edge_detect is
 port (
  clk : in std_logic;
  reset : in std_logic;
  threshold : in std_logic_vector(9 downto
0);
  t_en : in std_logic;
  op1 : in std_logic_vector(9 downto 0);
  op2 : in std_logic_vector(9 downto 0);
  en : in std_logic;
  r : out std_logic
 );
end simple_edge_detect;

architecture behavioral of
simple_edge_detect is
signal r_p, r_n : std_logic;
signal latch_threshold : std_logic_vector(9
downto 0);
begin
```

```
 clock_process: process(clk)
 begin
  if rising_edge(clk) then
   if reset = '1' or en = '0' then
    r_p <= '0';
   else
    r_p <= r_n;
   end if;
   if t_en = '1' then
    latch_threshold <= threshold;
   end if;
  end if;
 end process;

 next_state_process:
process(latch_threshold, op1, op2)
 variable s : unsigned(9 downto 0);
 begin
  if unsigned(op1) < unsigned(op2) then
   s := unsigned(op2) - unsigned(op1);
  else
   s := unsigned(op1) - unsigned(op2);
  end if;
  if s < unsigned(latch_threshold) then
   r_n <= '0';
  else
   r_n <= '1';
  end if;
 end process;

 r <= r_p;
end behavioral;
```

7. Program Modul Blok VGA *Driver*

File: vga_driver.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use work.work_package.all;

entity vga_driver is
 port (
  clk_100mhz : in std_logic;
  reset : in std_logic;
  next_index : in std_logic;
  next_index_dir : in std_logic;
  next_pixel : in std_logic;
  next_pixel_ori : in std_logic;
  next_pixel_dir : in std_logic;
  r : out std_logic_vector(2 downto 0);
  g : out std_logic_vector(2 downto 0);
  b : out std_logic_vector(2 downto 1);
  addr1 : out std_logic_vector(16 downto 0);
  din1 : in std_logic_vector(2 downto 0);
  addr2 : out std logic vector(16 downto 0);
  din2 : in std_logic;
  addr_py1 : out std_logic_vector(7 downto
0);
  din_py1 : in std_logic_vector(8 downto 0);
  ym1 : in std_logic_vector(7 downto 0);
  y0_1 : in std_logic_vector(7 downto 0);
  y1_1 : in std_logic_vector(7 downto 0);
  addr_px1 : out std_logic_vector(8 downto
0);
  din_px1 : in std_logic_vector(7 downto 0);
  xm1 : in std_logic_vector(8 downto 0);
  xmw1 : in std_logic_vector(8 downto 0);
  x0_1 : in std_logic_vector(8 downto 0);
  x1_1 : in std_logic_vector(8 downto 0);
  addr_px2 : out std_logic_vector(8 downto
0);
  din_px2 : in std_logic_vector(7 downto 0);
  addr_py2 : out std_logic_vector(11 downto
0);
  din_py2 : in std_logic_vector(8 downto 0);
  ym2 : in std_logic_vector(7 downto 0);
  bounds_addr : in std_logic_vector(3 downto
0);
  bounds_count : in std_logic_vector(3
downto 0);
  addr_b : out std_logic_vector(3 downto 0);
  din_bx0 : in std_logic_vector(8 downto 0);
  din_bx1 : in std_logic_vector(8 downto 0);
  din by0 : in std logic vector(7 downto 0);
  din_by1 : in std_logic_vector(7 downto 0);
  addr_f : out std_logic_vector(9 downto 0);
  din_f : in std_logic_vector(15 downto 0);
  addr_c : out std_logic_vector(3 downto 0);
  din_c : in std_logic_vector(5 downto 0);
  addr_ci : out std_logic_vector(7 downto
0);
  din_ci : in std_logic_vector(15 downto 0);
  addr_d : out std_logic_vector(8 downto 0);
  din_d : in std_logic_vector(8 downto 0);
  state : in state_type;
  ce : in std_logic;
  capture : in std_logic;
  threshold : in std_logic_vector(9 downto
0);
  processing_time : in std_logic_vector(31
downto 0);
  capture_time : in std_logic_vector(31
downto 0);
  icr_btn : out std_logic;
  pixel_clock : in std_logic_vector(7 downto
0);
  hs : out std_logic;
  vs : out std_logic;
  active : out std logic
 );
end vga_driver;

architecture behavioral of vga_driver is
signal hs_p, hs_n : std_logic;
signal vs_p, vs_n : std_logic;
signal active_p, active_n : std_logic;
signal active_lat1_p, active_lat1_n :
std_logic;
signal active_lat2_p, active_lat2_n :
std_logic;
signal active_lat3_p, active_lat3_n :
std_logic;
signal vblank_p, vblank_n : std_logic;
signal clk_div_p, clk_div_n : std_logic;
signal hcnt_p, hcnt_n : unsigned(10 downto
0);
signal vcnt_p, vcnt_n : unsigned(9 downto
0);
signal hcnt_lat1_p, hcnt_lat1_n :
unsigned(10 downto 0);
signal vcnt_lat1_p, vcnt_lat1_n : unsigned(9
downto 0);
signal hcnt_lat2_p, hcnt_lat2_n :
unsigned(10 downto 0);
signal vcnt_lat2_p, vcnt_lat2_n : unsigned(9
downto 0);
signal hcnt_lat3_p, hcnt_lat3_n :
unsigned(10 downto 0);
signal vcnt_lat3_p, vcnt_lat3_n : unsigned(9
downto 0);
signal hcnt_lat4_p, hcnt_lat4_n :
unsigned(10 downto 0);
```

```
signal vcnt_lat4_p, vcnt_lat4_n : unsigned(9
downto 0);
signal ce_hcnt : std_logic;
signal ce_vcnt : std_logic;
signal pick r : std logic vector(2 downto
0);
signal pick_g : std_logic_vector(2 downto
0);
signal pick_b : std_logic_vector(2 downto
1);
signal r_p, r_n : std_logic_vector(2 downto
0);
signal g_p, g_n : std_logic_vector(2 downto
0);
signal b_p, b_n : std_logic_vector(2 downto
1);
signal addr1_p, addr1_n :
std_logic_vector(16 downto 0);
signal addr2_p, addr2_n :
std_logic_vector(16 downto 0);
signal addr_py1_p, addr_py1_n :
std_logic_vector(7 downto 0);
signal addr_px1_p, addr_px1_n :
std_logic_vector(8 downto 0);
signal addr_px2_p, addr_px2_n :
std_logic_vector(8 downto 0);
signal addr_py2_p, addr_py2_n :
std_logic_vector(11 downto 0);
signal addr_py2s_p, addr_py2s_n :
std_logic_vector(7 downto 0);
signal addr_py1_lat1_p, addr_py1_lat1_n :
std_logic_vector(7 downto 0);
signal addr_px1_lat1_p, addr_px1_lat1_n :
std logic vector(8 downto 0);
signal addr_px2_lat1_p, addr_px2_lat1_n :
std_logic_vector(8 downto 0);
signal addr_py2_lat1_p, addr_py2_lat1_n :
std_logic_vector(7 downto 0);
signal addr_proj_p, addr_proj_n :
std_logic_vector(11 downto 0);
signal proj_p, proj_n : std_logic_vector(8
downto 0);
signal pixel_x_p, pixel_x_n :
std_logic_vector(8 downto 0);
signal pixel_y_p, pixel_y_n :
std_logic_vector(7 downto 0);
signal pixel_dat_p, pixel_dat_n :
std_logic_vector(3 downto 0);
signal latch_next_index : std_logic_vector(7
downto 0);
signal hold_ni_p, hold_ni_n : unsigned(25
downto 0);
signal ce_hold_ni : std_logic;
signal latch_next_pixel : std_logic_vector(7
downto 0);
signal hold_np_p, hold_np_n : unsigned(25
downto 0);
signal ce_hold_np : std_logic;
signal addr_b_p, addr_b_n :
std_logic_vector(3 downto 0);
signal ce_addr_b, r_addr_b : std_logic;
signal y0_p, y0_n : std_logic_vector(7
downto 0);
signal y1_p, y1_n : std_logic_vector(7
downto 0);
signal cdm_p, cdm_n : std_logic_vector(5
downto 0);
signal addr_b_lat1_p, addr_b_lat1_n :
std_logic_vector(3 downto 0);
signal addr_f_p, addr_f_n :
std_logic_vector(9 downto 0);
signal addr_ci_p, addr_ci_n :
std_logic_vector(7 downto 0);
signal addr_c_p, addr_c_n :
std_logic_vector(3 downto 0);
signal addr_d_p, addr_d_n :
std_logic_vector(8 downto 0);
signal addr_ds_p, addr_ds_n :
std_logic_vector(5 downto 0);
signal addr_c_lat1_p, addr_c_lat1_n :
std_logic_vector(3 downto 0);
signal addr d lat1 p, addr d lat1 n :
std_logic_vector(5 downto 0);
begin

 color_process: process(hcnt_lat4_p,
vcnt_lat4_p, din1,din2, din_py1, ym1, y0_1,
y1_1, din_px1, xm1, xmw1, x0_1, x1_1, state,
din_px2, din_py2, ym2, din_bx0, din_bx1,
din_by0, din_by1, bounds_count, din_f,
din_ci, din_c, din_d, cdm_p, pixel_x_p,
pixel_y_p, pixel_dat_p, addr_proj_p, proj_p,
bounds_addr, addr_b_p, y0_p, y1_p, ce,
capture)
 variable temp : integer := 0;
 begin
  pick_r <= (others => '0');
  pick_g <= (others => '0');
  pick_b <= (others => '0');

  if hcnt_lat4_p < 320 then
   if vcnt_lat4_p < 240 then
    pick_r <= din1(2 downto 0);
    pick_g <= din1(2 downto 0);
    pick_b <= din1(2 downto 1);
    if ((vcnt_lat4_p = unsigned(y0_1) or
vcnt_lat4_p = unsigned(y1_1)) and
hcnt_lat4_p >= unsigned(x0_1) and
hcnt_lat4_p <= unsigned(x1_1)) or
((hcnt_lat4_p = unsigned(x0_1) or
hcnt lat4 p = unsigned(x1 1)) and
vcnt_lat4_p >= unsigned(y0_1) and
vcnt_lat4_p <= unsigned(y1_1)) then
     pick_r <= (others => '0');
     pick_g <= (others => '1');
     pick_b <= (others => '0');
    end if;
    if unsigned(bounds_count) /= 0 and
(((vcnt_lat4_p = unsigned(din_by0) or
vcnt_lat4_p = unsigned(din_by1)) and
hcnt_lat4_p >= unsigned(din_bx0) and
hcnt_lat4_p <= unsigned(din_bx1)) or
((hcnt_lat4_p = unsigned(din_bx0) or
hcnt_lat4_p = unsigned(din_bx1)) and
vcnt_lat4_p >= unsigned(din_by0) and
vcnt_lat4_p <= unsigned(din_by1))) then
     pick_r <= (others => '0');
     pick_g <= (others => '0');
     pick_b <= (others => '1');
     if addr_b_p = bounds_addr then
      pick_r <= (others => '1');
      pick_g <= (others => '0');
      pick_b <= (others => '0');
     end if;
    end if;
    if (ce = '0' or capture = '0') and
hcnt_lat4_p = unsigned(pixel_x_p) and
vcnt_lat4_p = unsigned(pixel_y_p) then
     pick_r <= (others => '1');
     pick_g <= (others => '0');
     pick_b <= (others => '1');
    end if;
   elsif vcnt_lat4_p < 480 then
    pick_r <= (others => din1(2));
    pick_g <= (others => din1(2));
    pick_b <= (others => din1(2));
    if ((vcnt_lat4_p-240 = unsigned(y0_1) or
vcnt_lat4_p-240 = unsigned(y1_1)) and
hcnt_lat4_p >= unsigned(x0_1) and
hcnt_lat4_p <= unsigned(x1_1)) or
((hcnt_lat4_p = unsigned(x0_1) or
hcnt_lat4_p = unsigned(x1_1)) and
vcnt_lat4_p-240 >= unsigned(y0_1) and
vcnt_lat4_p-240 <= unsigned(y1_1)) then
```

```
      pick_r <= (others => '0');
      pick_g <= (others => '1');
      pick_b <= (others => '0');
     end if;
     if unsigned(bounds count) /= 0 and
(((vcnt_lat4_p-240 = unsigned(din_by0) or
vcnt_lat4_p-240 = unsigned(din_by1)) and
hcnt_lat4_p >= unsigned(din_bx0) and
hcnt_lat4_p <= unsigned(din_bx1)) or
((hcnt_lat4_p = unsigned(din_bx0) or
hcnt_lat4_p = unsigned(din_bx1)) and
vcnt_lat4_p-240 >= unsigned(din_by0) and
vcnt_lat4_p-240 <= unsigned(din_by1))) then
      pick_r <= (others => '0');
      pick_g <= (others => '0');
      pick_b <= (others => '1');
      if addr_b_p = bounds_addr then
       pick_r <= (others => '1');
       pick_g <= (others => '0');
       pick_b <= (others => '0');
      end if;
     end if;
     if (ce = '0' or capture = '0') and
hcnt_lat4_p = unsigned(pixel_x_p) and
vcnt_lat4_p-240 = unsigned(pixel_y_p) then
      pick_r <= (others => '1');
      pick_g <= (others => '0');
      pick_b <= (others => '1');
     end if;
    elsif vcnt_lat4_p < 544 then
     pick_r <= (others => '1');
     pick_g <= (others => '1');
     pick_b <= (others => '1');
     if vcnt lat4 p-480 = unsigned(din px2(7
downto 2)) then
      pick_r <= (others => '0');
      pick_g <= (others => '1');
      pick_b <= (others => '0');
     end if;
     if hcnt_lat4_p = unsigned(din_bx0) or
hcnt_lat4_p = unsigned(din_bx1) then
      pick_r <= (others => '0');
      pick_g <= (others => '0');
      pick_b <= (others => '1');
     end if;
     if (ce = '0' or capture = '0') and
addr_proj_p(11 downto 9) = "010" and
hcnt_lat4_p = unsigned(addr_proj_p(8 downto
0)) then
      pick_r <= (others => '1');
      pick_g <= (others => '0');
      pick_b <= (others => '1');
     end if;
     if hcnt_lat4_p < unsigned(x0_1) or
hcnt_lat4_p > unsigned(x1_1) then
      pick_r(1 downto 0) <= (others => '0');
      pick_g(1 downto 0) <= (others => '0');
      pick_b(1) <= '0';
     end if;
    end if;
   elsif hcnt_lat4_p < 400 then
    if hcnt_lat4_p < 384 and vcnt_lat4_p >=
240 and vcnt_lat4_p < 480 then
     pick_r <= (others => '1');
     pick_g <= (others => '1');
     pick_b <= (others => '1');
     if hcnt_lat4_p-320 = unsigned(din_py2(8
downto 3)) then
      pick_r <= (others => '0');
      pick_g <= (others => '1');
      pick_b <= (others => '0');
     end if;
     if vcnt_lat4_p-240 = unsigned(ym2) then
      pick_r <= (others => '1');
      pick_g <= (others => '0');
      pick_b <= (others => '0');
     end if;
     if vcnt_lat4_p-240 = unsigned(y0_p) or
vcnt_lat4_p-240 = unsigned(y1_p) then
      pick_r <= (others => '0');
      pick_g <= (others => '0');
      pick b <= (others => '1');
     end if;
     if (ce = '0' or capture = '0') and
addr_proj_p(11 downto 9) = "011" and
vcnt_lat4_p-240 = unsigned(addr_proj_p(8
downto 0)) then
      pick_r <= (others => '1');
      pick_g <= (others => '0');
      pick_b <= (others => '1');
     end if;
     if vcnt_lat4_p-240 < unsigned(y0_1) or
vcnt_lat4_p-240 > unsigned(y1_1) then
      pick_r(1 downto 0) <= (others => '0');
      pick_g(1 downto 0) <= (others => '0');
      pick_b(1) <= '0';
     end if;
    end if;
   elsif hcnt_lat4_p < 720 then
    if vcnt_lat4_p < 240 then
     pick_r <= (others => din2);
     pick_g <= (others => din2);
     pick_b <= (others => din2);
     if ((vcnt_lat4_p = unsigned(y0_1) or
vcnt_lat4_p = unsigned(y1_1)) and
hcnt_lat4_p-400 >= unsigned(x0_1) and
hcnt_lat4_p-400 <= unsigned(x1_1)) or
((hcnt_lat4_p-400 = unsigned(x0_1) or
hcnt_lat4_p-400 = unsigned(x1_1)) and
vcnt_lat4_p >= unsigned(y0_1) and
vcnt lat4 p <= unsigned(y1 1)) then
      pick_r <= (others => '0');
      pick_g <= (others => '1');
      pick_b <= (others => '0');
     end if;
     if unsigned(bounds_count) /= 0 and
(((vcnt_lat4_p = unsigned(din_by0) or
vcnt_lat4_p = unsigned(din_by1)) and
hcnt_lat4_p-400 >= unsigned(din_bx0) and
hcnt_lat4_p-400 <= unsigned(din_bx1)) or
((hcnt_lat4_p-400 = unsigned(din_bx0) or
hcnt_lat4_p-400 = unsigned(din_bx1)) and
vcnt_lat4_p >= unsigned(din_by0) and
vcnt_lat4_p <= unsigned(din_by1))) then
      pick_r <= (others => '0');
      pick_g <= (others => '0');
      pick_b <= (others => '1');
      if addr_b_p = bounds_addr then
       pick_r <= (others => '1');
       pick_g <= (others => '0');
       pick_b <= (others => '0');
      end if;
     end if;
     if (ce = '0' or capture = '0') and
hcnt_lat4_p-400 = unsigned(pixel_x_p) and
vcnt_lat4_p = unsigned(pixel_y_p) then
      pick_r <= (others => '1');
      pick_g <= (others => '0');
      pick_b <= (others => '1');
     end if;
    elsif vcnt_lat4_p < 304 then
     pick_r <= (others => '1');
     pick_g <= (others => '1');
     pick_b <= (others => '1');
     if hcnt_lat4_p-400 >= unsigned(xm1) and
hcnt_lat4_p-400 <= unsigned(xmw1) then
      pick_r <= (others => '1');
      pick_g <= "100";
      pick_b <= "10";
     end if;
     if vcnt_lat4_p-240 = unsigned(din_px1(7
downto 2)) then
      pick_r <= (others => '0');
      pick_g <= (others => '1');
```

```
      pick_b <= (others => '0');
     end if;
     if hcnt_lat4_p-400 = unsigned(x0_1) or
hcnt_lat4_p-400 = unsigned(x1_1) then
      pick r <= (others => '0');
      pick_g <= (others => '0');
      pick_b <= (others => '1');
     end if;
     if (ce = '0' or capture = '0') and
addr_proj_p(11 downto 9) = "000" and
hcnt_lat4_p-400 = unsigned(addr_proj_p(8
downto 0)) then
      pick_r <= (others => '1');
      pick_g <= (others => '0');
      pick_b <= (others => '1');
     end if;
    end if;
   elsif hcnt_lat4_p < 784 then
    if vcnt_lat4_p < 240 then
     pick_r <= (others => '1');
     pick_g <= (others => '1');
     pick_b <= (others => '1');
     if hcnt_lat4_p-720 = unsigned(din_py1(8
downto 3)) then
      pick_r <= (others => '0');
      pick_g <= (others => '1');
      pick_b <= (others => '0');
     end if;
     if vcnt_lat4_p = unsigned(ym1) then
      pick_r <= (others => '1');
      pick_g <= (others => '0');
      pick_b <= (others => '0');
     end if;
     if vcnt lat4 p = unsigned(y0 1) or
vcnt_lat4_p = unsigned(y1_1) then
      pick_r <= (others => '0');
      pick_g <= (others => '0');
      pick_b <= (others => '1');
     end if;
     if (ce = '0' or capture = '0') and
addr_proj_p(11 downto 9) = "001" and
vcnt_lat4_p = unsigned(addr_proj_p(8 downto
0)) then
      pick_r <= (others => '1');
      pick_g <= (others => '0');
      pick_b <= (others => '1');
     end if;
    end if;
   end if;

   if hcnt_lat4_p >= 400 and vcnt_lat4_p >=
320 then
    if vcnt_lat4_p >= 324 and vcnt_lat4_p <
332 and hcnt_lat4_p >= 568 and hcnt_lat4_p <
632 and din_f(((to_integer(hcnt_lat4_p)-568)
mod 8)*2) = '1' then
     pick_r <= (others => '1');
     pick_g <= (others => '1');
     pick_b <= (others => '1');
    end if;
    if vcnt_lat4_p >= 336 and vcnt_lat4_p <
400 and hcnt_lat4_p >= 456 and hcnt_lat4_p <
744 then
     pick_r <= (others => '1');
     pick_g <= (others => '1');
     pick_b <= (others => '1');
     if vcnt_lat4_p-336 > unsigned(not
(din_d(8 downto 3))) then
      pick_r <= (others => '0');
      pick_g <= (others => '1');
      pick_b <= (others => '0');
      if (hcnt_lat4_p-456) mod 8 = 0 or
(hcnt_lat4_p-456) mod 8 = 7 then
       pick_r <= (others => '0');
       pick_g <= "100";
       pick_b <= (others => '0');
      if (hcnt_lat4_p-456)/8 =
unsigned(cdm_p) then
       pick_r <= (others => '1');
       pick_g <= (others => '0');
       pick b <= (others => '0');
      end if;
      if (ce = '0' or capture = '0') and
addr_proj_p(11 downto 9) = "100" and
(hcnt_lat4_p-456)/8 = unsigned(addr_proj_p(8
downto 0)) then
       pick_r <= (others => '1');
       pick_g <= (others => '0');
       pick_b <= (others => '1');
      end if;
     end if;
    end if;
    if vcnt_lat4_p = 399 or vcnt_lat4_p-336
= unsigned(not (din_d(8 downto 3))) then
     pick_r <= (others => '0');
     pick_g <= "100";
     pick_b <= (others => '0');
     if (hcnt_lat4_p-456)/8 =
unsigned(cdm_p) then
      pick_r <= (others => '1');
      pick_g <= (others => '0');
      pick_b <= (others => '0');
     end if;
     if (ce = '0' or capture = '0') and
addr_proj_p(11 downto 9) = "100" and
(hcnt_lat4_p-456)/8 = unsigned(addr_proj_p(8
downto 0)) then
      pick_r <= (others => '1');
      pick_g <= (others => '0');
      pick b <= (others => '1');
     end if;
    end if;
   end if;
   if vcnt_lat4_p >= 404 and vcnt_lat4_p <
412 and hcnt_lat4_p >= 456 and hcnt_lat4_p <
744 and din_f(((to_integer(hcnt_lat4_p)-456)
mod 8)*2) = '1' then
    pick_r <= (others => '1');
    pick_g <= (others => '1');
    pick_b <= (others => '1');
    if (hcnt_lat4_p-456)/8 = unsigned(cdm_p)
then
     pick_r <= (others => '1');
     pick_g <= (others => '0');
     pick_b <= (others => '0');
    end if;
    if (ce = '0' or capture = '0') and
addr_proj_p(11 downto 9) = "100" and
(hcnt_lat4_p-456)/8 = unsigned(addr_proj_p(8
downto 0)) then
     pick_r <= (others => '1');
     pick_g <= (others => '0');
     pick_b <= (others => '1');
    end if;
   end if;
   if vcnt_lat4_p >= 420 and vcnt_lat4_p <
428 and hcnt_lat4_p >= 560 and hcnt_lat4_p <
640 and din_f(((to_integer(hcnt_lat4_p)-560)
mod 8)*2) = '1' then
    pick_r <= (others => '1');
    pick_g <= (others => '1');
    pick_b <= (others => '1');
   end if;
   if vcnt_lat4_p >= 432 and vcnt_lat4_p <
448 and hcnt_lat4_p >= 440 and hcnt_lat4_p <
760 and (hcnt_lat4_p-440)/32 <
unsigned(bounds_count) and ((hcnt_lat4_p-
440)/16) mod 2 = 0 and
din_ci((to_integer(hcnt_lat4_p)-440) mod 16)
= '1' then
    pick_r <= (others => '1');
    pick_g <= (others => '1');
    pick_b <= (others => '1');
```

```
    if unsigned(bounds_addr) = (hcnt_lat4_p-
440)/32 then
     pick_r <= (others => '1');
     pick_g <= (others => '0');
     pick b <= (others => '0');
    end if;
   end if;
   if vcnt_lat4_p >= 452 and vcnt_lat4_p <
460 and hcnt_lat4_p >= 560 and hcnt_lat4_p <
640 and (hcnt_lat4_p-560)/8 /= 4 and
din_f(((to_integer(hcnt_lat4_p)-560) mod
8)*2) = '1' then
    pick_r <= (others => '1');
    pick_g <= (others => '1');
    pick_b <= (others => '1');
   end if;
   if vcnt_lat4_p >= 464 and vcnt_lat4_p <
480 and hcnt_lat4_p >= 440 and hcnt_lat4_p <
760 and (hcnt_lat4_p-440)/32 <
unsigned(bounds_count) and ((hcnt_lat4_p-
440)/16) mod 2 = 0 and
din_f((to_integer(hcnt_lat4_p)-440) mod 16)
= '1' then
    pick_r <= (others => '1');
    pick_g <= (others => '1');
    pick_b <= (others => '1');
    if unsigned(bounds_addr) = (hcnt_lat4_p-
440)/32 then
     pick_r <= (others => '1');
     pick_g <= (others => '0');
     pick_b <= (others => '0');
    end if;
   end if;
   if vcnt lat4 p >= 484 and vcnt lat4 p <
492 and (hcnt_lat4_p-400)/8 /= 9 and
(hcnt_lat4_p-400)/8 < 13 and
din_f(((to_integer(hcnt_lat4_p)-400) mod
8)*2) = '1' then
    pick_r <= (others => '1');
    pick_g <= (others => '1');
    pick_b <= (others => '1');
   end if;
   if vcnt_lat4_p >= 500 and vcnt_lat4_p <
508 and (hcnt_lat4_p-400)/8 /= 6 and
(hcnt_lat4_p-400)/8 /= 8 and (hcnt_lat4_p-
400)/8 /= 14 and (hcnt_lat4_p-400)/8 < 16
and din_f(((to_integer(hcnt_lat4_p)-400) mod
8)*2) = '1' then
    pick_r <= (others => '1');
    pick_g <= (others => '1');
    pick_b <= (others => '1');
   end if;
   if vcnt_lat4_p >= 516 and vcnt_lat4_p <
524 and (hcnt_lat4_p-400)/8 /= 5 and
(hcnt_lat4_p-400)/8 /= 12 and (hcnt_lat4_p-
400)/8 /= 22 and (hcnt_lat4_p-400)/8 < 25
and din_f(((to_integer(hcnt_lat4_p)-400) mod
8)*2) = '1' then
    pick_r <= (others => '1');
    pick_g <= (others => '1');
    pick_b <= (others => '1');
   end if;
   if vcnt_lat4_p >= 532 and vcnt_lat4_p <
540 and (hcnt_lat4_p-400)/8 /= 4 and
(hcnt_lat4_p-400)/8 /= 11 and (hcnt_lat4_p-
400)/8 /= 15 and (hcnt_lat4_p-400)/8 /= 18
and (hcnt_lat4_p-400)/8 /= 23 and
(hcnt_lat4_p-400)/8 /= 28 and (hcnt_lat4_p-
400)/8 /= 38 and (hcnt_lat4_p-400)/8 < 41
and din_f(((to_integer(hcnt_lat4_p)-400) mod
8)*2) = '1' then
    pick_r <= (others => '1');
    pick_g <= (others => '1');
    pick_b <= (others => '1');
   end if;
   if (ce = '0' or capture = '0') then
    if vcnt_lat4_p >= 548 and vcnt_lat4_p <
556 and (hcnt_lat4_p-400)/8 /= 1 and
(hcnt_lat4_p-400)/8 /= 5 and (hcnt_lat4_p-
400)/8 /= 7 and (hcnt_lat4_p-400)/8 < 11 and
din f(((to_integer(hcnt lat4 p)-400) mod
8)*2) = '1' then
     pick_r <= (others => '1');
     pick_g <= (others => '1');
     pick_b <= (others => '1');
    end if;
    if vcnt_lat4_p >= 564 and vcnt_lat4_p <
572 then
     if (hcnt_lat4_p-400)/8 /= 1 and
(hcnt_lat4_p-400)/8 /= 5 and (hcnt_lat4_p-
400)/8 /= 7 and (hcnt_lat4_p-400)/8 /= 10
and (hcnt_lat4_p-400)/8 < 12 and
din_f(((to_integer(hcnt_lat4_p)-400) mod
8)*2) = '1' then
      pick_r <= (others => '1');
      pick_g <= (others => '1');
      pick_b <= (others => '1');
     end if;
     if hcnt_lat4_p >= 504 and (hcnt_lat4_p-
504)/8 /= 3 and (hcnt_lat4_p-504)/8 < 5 then
      if ((((vcnt_lat4_p-564) mod 8 = 0 or
(vcnt_lat4_p-564) mod 8 = 7) and
(hcnt_lat4_p-504) mod 8 >= 0 and
(hcnt_lat4_p-504) mod 8 <= 7) or
(((hcnt_lat4_p-504) mod 8 = 0 or
(hcnt_lat4_p-504) mod 8 = 7) and
(vcnt_lat4_p-564) mod 8 >= 0 and
(vcnt_lat4_p-564) mod 8 <= 7)) then
       pick_r <= (others => '1');
       pick g <= (others => '0');
       pick_b <= (others => '0');
      else
       if ((hcnt_lat4_p-504)/8 < 3 and
pixel_dat_p(3-to_integer((hcnt_lat4_p-
504)/8)) = '1') or ((hcnt_lat4_p-504)/8 = 4
and pixel_dat_p(0) = '1') then
        pick_r <= (others => '1');
        pick_g <= (others => '1');
        pick_b <= (others => '1');
       end if;
      end if;
     end if;
    end if;
    if vcnt_lat4_p >= 580 and vcnt_lat4_p <
588 then
     if (hcnt_lat4_p-400)/8 < 5 and
din_f(((to_integer(hcnt_lat4_p)-400) mod
8)*2) = '1' then
      pick_r <= (others => '1');
      pick_g <= (others => '1');
      pick_b <= (others => '1');
     end if;
     if hcnt_lat4_p >= 448 and (hcnt_lat4_p-
448)/8 < 17 then
      if ((((vcnt_lat4_p-580) mod 8 = 0 or
(vcnt_lat4_p-580) mod 8 = 7) and
(hcnt_lat4_p-448) mod 8 >= 0 and
(hcnt_lat4_p-448) mod 8 <= 7) or
(((hcnt_lat4_p-448) mod 8 = 0 or
(hcnt_lat4_p-448) mod 8 = 7) and
(vcnt_lat4_p-580) mod 8 >= 0 and
(vcnt_lat4_p-580) mod 8 <= 7)) then
       pick_r <= (others => '1');
       pick_g <= (others => '0');
       pick_b <= (others => '0');
      else
       case state is
        when idle => temp := 0;
        when capture_image => temp := 1;
        when y0_first => temp := 2;
        when y1_first => temp := 3;
        when px1 => temp := 4;
        when x_first => temp := 5;
```

```
        when x_second => temp := 6;
        when py2 => temp := 7;
        when y0_second => temp := 8;
        when y1_second => temp := 9;
        when init_res => temp := 10;
        when init_l_res => temp := 11;
        when res => temp := 12;
        when l_res => temp := 13;
        when l_res2 => temp := 14;
        when dist => temp := 15;
        when dm => temp := 16;
       end case;
       if (hcnt_lat4_p-448)/8 = temp then
        pick_r <= (others => '1');
        pick_g <= (others => '1');
        pick_b <= (others => '1');
       end if;
      end if;
     end if;
    end if;
   end if;
 end if;

end process;

clock_process1: process(clk_100mhz)
begin
 if rising_edge(clk_100mhz) then
  if reset = '1' then
   hs_p <= '1';
   vs_p <= '1';
   active_p <= '0';
   active_lat1_p <= '0';
   active_lat2_p <= '0';
   active_lat3_p <= '0';
   vblank_p <= '0';
   clk_div_p <= '0';
   hcnt_p <= (others => '0');
   vcnt_p <= (others => '0');
   hcnt_lat1_p <= (others => '0');
   vcnt_lat1_p <= (others => '0');
   hcnt_lat2_p <= (others => '0');
   vcnt_lat2_p <= (others => '0');
   hcnt_lat3_p <= (others => '0');
   vcnt_lat3_p <= (others => '0');
   hcnt_lat4_p <= (others => '0');
   vcnt_lat4_p <= (others => '0');
   addr_proj_p <= (others => '0');
   proj_p <= (others => '0');
   pixel_x_p <= (others => '0');
   pixel_y_p <= (others => '0');
   pixel_dat_p <= (others => '0');
   y0_p <= (others => '0');
   y1_p <= (others => '0');
   cdm_p <= (others => '0');
   addr_b_lat1_p <= (others => '0');
   addr_py1_lat1_p <= (others => '0');
   addr_px1_lat1_p <= (others => '0');
   addr_px2_lat1_p <= (others => '0');
   addr_py2_lat1_p <= (others => '0');
   addr_c_lat1_p <= (others => '0');
   addr_d_lat1_p <= (others => '0');
  else
   hs_p <= hs_n;
   vs_p <= vs_n;
   active_p <= active_n;
   active_lat1_p <= active_lat1_n;
   active_lat2_p <= active_lat2_n;
   active_lat3_p <= active_lat3_n;
   vblank_p <= vblank_n;
   clk_div_p <= clk_div_n;
   hcnt_lat1_p <= hcnt_lat1_n;
   vcnt_lat1_p <= vcnt_lat1_n;
   hcnt_lat2_p <= hcnt_lat2_n;
   vcnt_lat2_p <= vcnt_lat2_n;
   hcnt_lat3_p <= hcnt_lat3_n;
   vcnt_lat3_p <= vcnt_lat3_n;
   hcnt_lat4_p <= hcnt_lat4_n;
   vcnt_lat4_p <= vcnt_lat4_n;
   if ce_hcnt = '1' then
    hcnt_p <= hcnt_n;
   end if;
   if ce_vcnt = '1' then
    vcnt_p <= vcnt_n;
   end if;
   addr_proj_p <= addr_proj_n;
   proj_p <= proj_n;
   pixel_x_p <= pixel_x_n;
   pixel_y_p <= pixel_y_n;
   pixel_dat_p <= pixel_dat_n;
   y0_p <= y0_n;
   y1_p <= y1_n;
   cdm_p <= cdm_n;
   addr_b_lat1_p <= addr_b_lat1_n;
   addr_py1_lat1_p <= addr_py1_lat1_n;
   addr_px1_lat1_p <= addr_px1_lat1_n;
   addr_px2_lat1_p <= addr_px2_lat1_n;
   addr_py2_lat1_p <= addr_py2_lat1_n;
   addr_c_lat1_p <= addr_c_lat1_n;
   addr_d_lat1_p <= addr_d_lat1_n;
  end if;
  if reset = '1' or (ce = '1' and capture =
'1') then
   latch_next_index <= (others => '0');
   latch_next_pixel <= (others => '0');
  else
   latch_next_index <= latch_next_index(6
downto 0) & next_index;
   latch_next_pixel <= latch_next_pixel(6
downto 0) & next_pixel;
  end if;
 end if;
end process;

clock_process2: process(clk_100mhz)
begin
 if rising_edge(clk_100mhz) then
  if reset = '1' or latch_next_index(0) =
'0' then
   hold_ni_p <= (others => '0');
  elsif ce_hold_ni = '1' then
   hold_ni_p <= hold_ni_n;
  end if;
  if reset = '1' or latch_next_pixel(0) =
'0' then
   hold_np_p <= (others => '0');
  elsif ce_hold_np = '1' then
   hold_np_p <= hold_np_n;
  end if;
 end if;
end process;

clock_process3: process(clk_100mhz)
begin
 if rising_edge(clk_100mhz) then
  if reset = '1' or active_lat2_p = '0'
then
   r_p <= (others => '0');
   g_p <= (others => '0');
   b_p <= (others => '0');
  else
   r_p <= r_n;
   g_p <= g_n;
   b_p <= b_n;
  end if;
  if reset = '1' or active_p = '0' then
   addr1_p <= (others => '0');
   addr2_p <= (others => '0');
   addr_py1_p <= (others => '0');
   addr_px1_p <= (others => '0');
   addr_px2_p <= (others => '0');
   addr_py2_p <= (others => '0');
   addr_py2s_p <= (others => '0');
   addr_f_p <= (others => '0');
```

```
    addr_c_p <= (others => '0');
    addr_ci_p <= (others => '0');
    addr_d_p <= (others => '0');
    addr_ds_p <= (others => '0');
   else
    addr1_p <= addr1_n;
    addr2_p <= addr2_n;
    addr_py1_p <= addr_py1_n;
    addr_px1_p <= addr_px1_n;
    addr_px2_p <= addr_px2_n;
    addr_py2_p <= addr_py2_n;
    addr_py2s_p <= addr_py2s_n;
    addr_f_p <= addr_f_n;
    addr_c_p <= addr_c_n;
    addr_ci_p <= addr_ci_n;
    addr_d_p <= addr_d_n;
    addr_ds_p <= addr_ds_n;
   end if;
   if reset = '1' or active_p = '0' or
r_addr_b = '1' then
    addr_b_p <= (others => '0');
   elsif ce_addr_b = '1' then
    addr_b_p <= addr_b_n;
   end if;
  end if;
 end process;

 next_state_process: process(hs_p, vs_p,
active_p, active_lat1_p, active_lat2_p,
clk_div_p, vblank_p, r_p, g_p, b_p, hcnt_p,
vcnt_p, hcnt_lat1_p, vcnt_lat1_p,
hcnt_lat2_p, vcnt_lat2_p, hcnt_lat3_p,
vcnt_lat3_p, hcnt_lat4_p, vcnt_lat4_p,
din_px1, din_py1, din_px2, din_py2, addr1_p,
addr2_p, addr_py1_p, addr_px1_p, addr_px2_p,
addr_py2_p, addr_py2s_p, addr_py1_lat1_p,
addr_px1_lat1_p, addr_px2_lat1_p,
addr_py2_lat1_p, addr_b_p, addr_b_lat1_p,
addr_f_p, addr_c_p, addr_ci_p, addr_d_p,
addr_ds_p, addr_c_lat1_p, addr_d_lat1_p,
din_bx1, bounds_count, hcnt_p, vcnt_p,
pick_r, pick_g, pick_b, bounds_addr,
threshold, latch_next_index, next_index_dir,
latch_next_pixel, next_pixel_ori,
next_pixel_dir, hold_ni_p, hold_np_p,
addr_proj_p, proj_p, pixel_x_p, pixel_y_p,
pixel_dat_p, y0_p, y1_p, din_bx0, din_bx1,
din_by0, din_by1, din1, din2, din_ci, din_c,
din_d, cdm_p, processing_time, pixel_clock,
capture_time)
 begin
  --default:
  hs_n <= hs_p;
  vs_n <= vs_p;
  active_n <= active_p;
  active_lat1_n <= active_p;
  active_lat2_n <= active_lat1_p;
  active_lat3_n <= active_lat2_p;
  clk_div_n <= not clk_div_p;
  hcnt_lat1_n <= hcnt_p;
  vcnt_lat1_n <= vcnt_p;
  hcnt_lat2_n <= hcnt_lat1_p;
  vcnt_lat2_n <= vcnt_lat1_p;
  hcnt_lat3_n <= hcnt_lat2_p;
  vcnt_lat3_n <= vcnt_lat2_p;
  hcnt_lat4_n <= hcnt_lat3_p;
  vcnt_lat4_n <= vcnt_lat3_p;
  vblank_n <= vblank_p;
  if hcnt_p = 1039 then
   hcnt_n <= (others => '0');
  else
   hcnt_n <= hcnt_p + 1;
  end if;
  if vcnt_p = 665 then
   vcnt_n <= (others => '0');
  else
   vcnt_n <= vcnt_p + 1;
  end if;
  r_n <= pick_r;
  g_n <= pick_g;
  b_n <= pick_b;
  if vcnt_lat2_p < 240 then
   addr1_n <=
std_logic_vector(to_unsigned(to_integer(vcnt
_lat2_p)*320 + to_integer(hcnt_lat2_p),17));
  else
   addr1_n <=
std_logic_vector(to_unsigned((to_integer(vcn
t_lat2_p)-240)*320 +
to_integer(hcnt_lat2_p),17));
  end if;
  addr2_n <=
std_logic_vector(to_unsigned(to_integer(vcnt
_lat2_p)*320+(to_integer(hcnt_lat2_p)-
400),17));
  addr_py1_n <=
std_logic_vector(vcnt_lat2_p(7 downto 0));
  addr_px1_n <=
std_logic_vector(hcnt_lat2_p(8 downto 0)-
400);
  addr_px2_n <=
std_logic_vector(hcnt_lat2_p(8 downto 0));
  addr_py2_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(bounds_addr))*240 +
to_integer(vcnt_lat2_p(7 downto 0)-
240),12));
  addr_py2s_n <=
std_logic_vector(vcnt_lat2_p(7 downto 0)-
240);
  addr_py1_lat1_n <= addr_py1_p;
  addr_px1_lat1_n <= addr_px1_p;
  addr_px2_lat1_n <= addr_px2_p;
  addr_py2_lat1_n <= addr_py2s_p;
  addr_b_n <=
std_logic_vector(unsigned(addr_b_p)+1);
  ce_addr_b <= '0';
  r_addr_b <= '0';
  addr_d_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(bounds_addr))*36 +
to_integer((hcnt_lat2_p(8 downto 0)-
456)/8),9));
  addr_ds_n <=
std_logic_vector(to_unsigned(to_integer((hcn
t_lat2_p(8 downto 0)-456)/8),6));
  addr_b_lat1_n <= addr_b_p;
  addr_c_lat1_n <= addr_c_p;
  addr_d_lat1_n <= addr_ds_p;
  addr_f_n <= addr_f_p;
  if vcnt_lat2_p >= 324 and vcnt_lat2_p <
332 then
   case (hcnt_lat2_p-568)/8 is
    when "00000000000" => addr_f_n <=
std_logic_vector(to_unsigned(13*16 +
(to_integer(vcnt_lat2_p)-324)*2,10));
    when "00000000001" => addr_f_n <=
std_logic_vector(to_unsigned(18*16 +
(to_integer(vcnt_lat2_p)-324)*2,10));
    when "00000000010" => addr_f_n <=
std_logic_vector(to_unsigned(28*16 +
(to_integer(vcnt_lat2_p)-324)*2,10));
    when "00000000011" => addr_f_n <=
std_logic_vector(to_unsigned(29*16 +
(to_integer(vcnt_lat2_p)-324)*2,10));
    when "00000000100" => addr_f_n <=
std_logic_vector(to_unsigned(10*16 +
(to_integer(vcnt_lat2_p)-324)*2,10));
    when "00000000101" => addr_f_n <=
std_logic_vector(to_unsigned(23*16 +
(to_integer(vcnt_lat2_p)-324)*2,10));
    when "00000000110" => addr_f_n <=
std_logic_vector(to_unsigned(12*16 +
(to_integer(vcnt_lat2_p)-324)*2,10));
```

```
    when "00000000111" => addr_f_n <=
std_logic_vector(to_unsigned(14*16 +
(to_integer(vcnt_lat2_p)-324)*2,10));
    when others => addr_f_n <= addr_f_p;
   end case;
  elsif vcnt_lat2_p >= 404 and vcnt_lat2_p <
412 then
   addr_f_n <=
std_logic_vector(to_unsigned(to_integer((hcn
t_lat2_p-456)/8)*16 +
(to_integer(vcnt_lat2_p)-404)*2,10));
  elsif vcnt_lat2_p >= 420 and vcnt_lat2_p <
428 then
   case (hcnt_lat2_p-560)/8 is
    when "00000000000" => addr_f_n <=
std_logic_vector(to_unsigned(28*16 +
(to_integer(vcnt_lat2_p)-420)*2,10));
    when "00000000001" => addr_f_n <=
std_logic_vector(to_unsigned(14*16 +
(to_integer(vcnt_lat2_p)-420)*2,10));
    when "00000000010" => addr_f_n <=
std_logic_vector(to_unsigned(16*16 +
(to_integer(vcnt_lat2_p)-420)*2,10));
    when "00000000011" => addr_f_n <=
std_logic_vector(to_unsigned(22*16 +
(to_integer(vcnt_lat2_p)-420)*2,10));
    when "00000000100" => addr_f_n <=
std_logic_vector(to_unsigned(14*16 +
(to_integer(vcnt_lat2_p)-420)*2,10));
    when "00000000101" => addr_f_n <=
std_logic_vector(to_unsigned(23*16 +
(to_integer(vcnt_lat2_p)-420)*2,10));
    when "00000000110" => addr_f_n <=
std logic vector(to_unsigned(29*16 +
(to_integer(vcnt_lat2_p)-420)*2,10));
    when "00000000111" => addr_f_n <=
std_logic_vector(to_unsigned(10*16 +
(to_integer(vcnt_lat2_p)-420)*2,10));
    when "00000001000" => addr_f_n <=
std_logic_vector(to_unsigned(28*16 +
(to_integer(vcnt_lat2_p)-420)*2,10));
    when "00000001001" => addr_f_n <=
std_logic_vector(to_unsigned(18*16 +
(to_integer(vcnt_lat2_p)-420)*2,10));
    when others => addr_f_n <= addr_f_p;
   end case;
  elsif vcnt_lat2_p >= 452 and vcnt_lat2_p <
460 then
   case (hcnt_lat2_p-560)/8 is
    when "00000000000" => addr_f_n <=
std_logic_vector(to_unsigned(25*16 +
(to_integer(vcnt_lat2_p)-452)*2,10));
    when "00000000001" => addr_f_n <=
std_logic_vector(to_unsigned(21*16 +
(to_integer(vcnt_lat2_p)-452)*2,10));
    when "00000000010" => addr_f_n <=
std_logic_vector(to_unsigned(10*16 +
(to_integer(vcnt_lat2_p)-452)*2,10));
    when "00000000011" => addr_f_n <=
std_logic_vector(to_unsigned(29*16 +
(to_integer(vcnt_lat2_p)-452)*2,10));
    when "00000000101" => addr_f_n <=
std_logic_vector(to_unsigned(23*16 +
(to_integer(vcnt_lat2_p)-452)*2,10));
    when "00000000110" => addr_f_n <=
std_logic_vector(to_unsigned(24*16 +
(to_integer(vcnt_lat2_p)-452)*2,10));
    when "00000000111" => addr_f_n <=
std_logic_vector(to_unsigned(22*16 +
(to_integer(vcnt_lat2_p)-452)*2,10));
    when "00000001000" => addr_f_n <=
std_logic_vector(to_unsigned(24*16 +
(to_integer(vcnt_lat2_p)-452)*2,10));
    when "00000001001" => addr_f_n <=
std_logic_vector(to_unsigned(27*16 +
(to_integer(vcnt_lat2_p)-452)*2,10));
    when others => addr_f_n <= addr_f_p;
   end case;
  elsif vcnt_lat2_p >= 464 and vcnt_lat2_p <
480 then
   addr_f_n <=
std logic vector(to_unsigned(to_integer(unsi
gned(din_c))*16 + (to_integer(vcnt_lat2_p)-
464),10));
  elsif vcnt_lat2_p >= 484 and vcnt_lat2_p <
492 then
   case (hcnt_lat2_p-400)/8 is
    when "00000000000" => addr_f_n <=
std_logic_vector(to_unsigned(29*16 +
(to_integer(vcnt_lat2_p)-484)*2,10));
    when "00000000001" => addr_f_n <=
std_logic_vector(to_unsigned(17*16 +
(to_integer(vcnt_lat2_p)-484)*2,10));
    when "00000000010" => addr_f_n <=
std_logic_vector(to_unsigned(27*16 +
(to_integer(vcnt_lat2_p)-484)*2,10));
    when "00000000011" => addr_f_n <=
std_logic_vector(to_unsigned(14*16 +
(to_integer(vcnt_lat2_p)-484)*2,10));
    when "00000000100" => addr_f_n <=
std_logic_vector(to_unsigned(28*16 +
(to_integer(vcnt_lat2_p)-484)*2,10));
    when "00000000101" => addr_f_n <=
std_logic_vector(to_unsigned(17*16 +
(to_integer(vcnt_lat2_p)-484)*2,10));
    when "00000000110" => addr_f_n <=
std_logic_vector(to_unsigned(24*16 +
(to_integer(vcnt_lat2_p)-484)*2,10));
    when "00000000111" => addr_f_n <=
std_logic_vector(to_unsigned(21*16 +
(to_integer(vcnt lat2 p)-484)*2,10));
    when "00000001000" => addr_f_n <=
std_logic_vector(to_unsigned(13*16 +
(to_integer(vcnt_lat2_p)-484)*2,10));
    when "00000001010" => addr_f_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(threshold(9 downto 8)))*16 +
(to_integer(vcnt_lat2_p)-484)*2,10));
    when "00000001011" => addr_f_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(threshold(7 downto 4)))*16 +
(to_integer(vcnt_lat2_p)-484)*2,10));
    when "00000001100" => addr_f_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(threshold(3 downto 0)))*16 +
(to_integer(vcnt_lat2_p)-484)*2,10));
    when others => addr_f_n <= addr_f_p;
   end case;
  elsif vcnt_lat2_p >= 500 and vcnt_lat2_p <
508 then
   case (hcnt_lat2_p-400)/8 is
    when "00000000000" => addr_f_n <=
std_logic_vector(to_unsigned(28*16 +
(to_integer(vcnt_lat2_p)-500)*2,10));
    when "00000000001" => addr_f_n <=
std_logic_vector(to_unsigned(14*16 +
(to_integer(vcnt_lat2_p)-500)*2,10));
    when "00000000010" => addr_f_n <=
std_logic_vector(to_unsigned(16*16 +
(to_integer(vcnt_lat2_p)-500)*2,10));
    when "00000000011" => addr_f_n <=
std_logic_vector(to_unsigned(22*16 +
(to_integer(vcnt_lat2_p)-500)*2,10));
    when "00000000100" => addr_f_n <=
std_logic_vector(to_unsigned(14*16 +
(to_integer(vcnt_lat2_p)-500)*2,10));
    when "00000000101" => addr_f_n <=
std_logic_vector(to_unsigned(23*16 +
(to_integer(vcnt_lat2_p)-500)*2,10));
    when "00000000111" => addr_f_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(bounds_addr))*16 +
(to_integer(vcnt_lat2_p)-500)*2,10));
```

```
    when "00000001001" => addr_f_n <=
std_logic_vector(to_unsigned(29*16 +
(to_integer(vcnt_lat2_p)-500)*2,10));
    when "00000001010" => addr_f_n <=
std logic vector(to_unsigned(24*16 +
(to_integer(vcnt_lat2_p)-500)*2,10));
    when "00000001011" => addr_f_n <=
std_logic_vector(to_unsigned(29*16 +
(to_integer(vcnt_lat2_p)-500)*2,10));
    when "00000001100" => addr_f_n <=
std_logic_vector(to_unsigned(10*16 +
(to_integer(vcnt_lat2_p)-500)*2,10));
    when "00000001101" => addr_f_n <=
std_logic_vector(to_unsigned(21*16 +
(to_integer(vcnt_lat2_p)-500)*2,10));
    when "00000001111" => addr_f_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(bounds_count))*16 +
(to_integer(vcnt_lat2_p)-500)*2,10));
    when others => addr_f_n <= addr_f_p;
   end case;
  elsif vcnt_lat2_p >= 516 and vcnt_lat2_p <
524 then
   case (hcnt_lat2_p-400)/8 is
    when "00000000000" => addr_f_n <=
std_logic_vector(to_unsigned(32*16 +
(to_integer(vcnt_lat2_p)-516)*2,10));
    when "00000000001" => addr_f_n <=
std_logic_vector(to_unsigned(10*16 +
(to_integer(vcnt_lat2_p)-516)*2,10));
    when "00000000010" => addr_f_n <=
std_logic_vector(to_unsigned(20*16 +
(to_integer(vcnt_lat2_p)-516)*2,10));
    when "00000000011" => addr f n <=
std_logic_vector(to_unsigned(29*16 +
(to_integer(vcnt_lat2_p)-516)*2,10));
    when "00000000100" => addr_f_n <=
std_logic_vector(to_unsigned(30*16 +
(to_integer(vcnt_lat2_p)-516)*2,10));
    when "00000000110" => addr_f_n <=
std_logic_vector(to_unsigned(25*16 +
(to_integer(vcnt_lat2_p)-516)*2,10));
    when "00000000111" => addr_f_n <=
std_logic_vector(to_unsigned(27*16 +
(to_integer(vcnt_lat2_p)-516)*2,10));
    when "00000001000" => addr_f_n <=
std_logic_vector(to_unsigned(24*16 +
(to_integer(vcnt_lat2_p)-516)*2,10));
    when "00000001001" => addr_f_n <=
std_logic_vector(to_unsigned(28*16 +
(to_integer(vcnt_lat2_p)-516)*2,10));
    when "00000001010" => addr_f_n <=
std_logic_vector(to_unsigned(14*16 +
(to_integer(vcnt_lat2_p)-516)*2,10));
    when "00000001011" => addr_f_n <=
std_logic_vector(to_unsigned(28*16 +
(to_integer(vcnt_lat2_p)-516)*2,10));
    when "00000001101" => addr_f_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(processing_time(31 downto 28)))*16 +
(to_integer(vcnt_lat2_p)-516)*2,10));
    when "00000001110" => addr_f_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(processing_time(27 downto 24)))*16 +
(to_integer(vcnt_lat2_p)-516)*2,10));
    when "00000001111" => addr_f_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(processing_time(23 downto 20)))*16 +
(to_integer(vcnt_lat2_p)-516)*2,10));
    when "00000010000" => addr_f_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(processing_time(19 downto 16)))*16 +
(to_integer(vcnt_lat2_p)-516)*2,10));
    when "00000010001" => addr_f_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(processing_time(15 downto 12)))*16 +
(to_integer(vcnt_lat2_p)-516)*2,10));
    when "00000010010" => addr_f_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(processing_time(11 downto 8)))*16 +
(to_integer(vcnt_lat2_p)-516)*2,10));
    when "00000010011" => addr f n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(processing_time(7 downto 4)))*16 +
(to_integer(vcnt_lat2_p)-516)*2,10));
    when "00000010100" => addr_f_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(processing_time(3 downto 0)))*16 +
(to_integer(vcnt_lat2_p)-516)*2,10));
    when "00000010101" => addr_f_n <=
std_logic_vector(to_unsigned(0*16 +
(to_integer(vcnt_lat2_p)-516)*2,10));
    when "00000010111" => addr_f_n <=
std_logic_vector(to_unsigned(23*16 +
(to_integer(vcnt_lat2_p)-516)*2,10));
    when "00000011000" => addr_f_n <=
std_logic_vector(to_unsigned(28*16 +
(to_integer(vcnt_lat2_p)-516)*2,10));
    when others => addr_f_n <= addr_f_p;
   end case;
  elsif vcnt_lat2_p >= 532 and vcnt_lat2_p <
540 then
   case (hcnt_lat2_p-400)/8 is
    when "00000000000" => addr_f_n <=
std_logic_vector(to_unsigned(25*16 +
(to_integer(vcnt_lat2_p)-532)*2,10));
    when "00000000001" => addr_f_n <=
std_logic_vector(to_unsigned(12*16 +
(to_integer(vcnt_lat2_p)-532)*2,10));
    when "00000000010" => addr_f_n <=
std logic vector(to_unsigned(21*16 +
(to_integer(vcnt_lat2_p)-532)*2,10));
    when "00000000011" => addr_f_n <=
std_logic_vector(to_unsigned(20*16 +
(to_integer(vcnt_lat2_p)-532)*2,10));
    when "00000000101" => addr_f_n <=
std_logic_vector(to_unsigned(25*16 +
(to_integer(vcnt_lat2_p)-532)*2,10));
    when "00000000110" => addr_f_n <=
std_logic_vector(to_unsigned(14*16 +
(to_integer(vcnt_lat2_p)-532)*2,10));
    when "00000000111" => addr_f_n <=
std_logic_vector(to_unsigned(27*16 +
(to_integer(vcnt_lat2_p)-532)*2,10));
    when "00000001000" => addr_f_n <=
std_logic_vector(to_unsigned(18*16 +
(to_integer(vcnt_lat2_p)-532)*2,10));
    when "00000001001" => addr_f_n <=
std_logic_vector(to_unsigned(24*16 +
(to_integer(vcnt_lat2_p)-532)*2,10));
    when "00000001010" => addr_f_n <=
std_logic_vector(to_unsigned(13*16 +
(to_integer(vcnt_lat2_p)-532)*2,10));
    when "00000001100" => addr_f_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(pixel_clock(7 downto 4)))*16 +
(to_integer(vcnt_lat2_p)-532)*2,10));
    when "00000001101" => addr_f_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(pixel_clock(3 downto 0)))*16 +
(to_integer(vcnt_lat2_p)-532)*2,10));
    when "00000001110" => addr_f_n <=
std_logic_vector(to_unsigned(0*16 +
(to_integer(vcnt_lat2_p)-532)*2,10));
    when "00000010000" => addr_f_n <=
std_logic_vector(to_unsigned(23*16 +
(to_integer(vcnt_lat2_p)-532)*2,10));
    when "00000010001" => addr_f_n <=
std_logic_vector(to_unsigned(28*16 +
(to_integer(vcnt_lat2_p)-532)*2,10));
    when "00000010011" => addr_f_n <=
std_logic_vector(to_unsigned(12*16 +
(to_integer(vcnt_lat2_p)-532)*2,10));
```

```
    when "00000010100" => addr_f_n <=
std_logic_vector(to_unsigned(10*16 +
(to_integer(vcnt_lat2_p)-532)*2,10));
    when "00000010101" => addr_f_n <=
std_logic_vector(to_unsigned(25*16 +
(to_integer(vcnt_lat2_p)-532)*2,10));
    when "00000010110" => addr_f_n <=
std_logic_vector(to_unsigned(29*16 +
(to_integer(vcnt_lat2_p)-532)*2,10));
    when "00000011000" => addr_f_n <=
std_logic_vector(to_unsigned(29*16 +
(to_integer(vcnt_lat2_p)-532)*2,10));
    when "00000011001" => addr_f_n <=
std_logic_vector(to_unsigned(18*16 +
(to_integer(vcnt_lat2_p)-532)*2,10));
    when "00000011010" => addr_f_n <=
std_logic_vector(to_unsigned(22*16 +
(to_integer(vcnt_lat2_p)-532)*2,10));
    when "00000011011" => addr_f_n <=
std_logic_vector(to_unsigned(14*16 +
(to_integer(vcnt_lat2_p)-532)*2,10));
    when "00000011101" => addr_f_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(capture_time(31 downto 28)))*16 +
(to_integer(vcnt_lat2_p)-532)*2,10));
    when "00000011110" => addr_f_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(capture_time(27 downto 24)))*16 +
(to_integer(vcnt_lat2_p)-532)*2,10));
    when "00000011111" => addr_f_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(capture_time(23 downto 20)))*16 +
(to_integer(vcnt_lat2_p)-532)*2,10));
    when "00000100000" => addr f n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(capture_time(19 downto 16)))*16 +
(to_integer(vcnt_lat2_p)-532)*2,10));
    when "00000100001" => addr_f_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(capture_time(15 downto 12)))*16 +
(to_integer(vcnt_lat2_p)-532)*2,10));
    when "00000100010" => addr_f_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(capture_time(11 downto 8)))*16 +
(to_integer(vcnt_lat2_p)-532)*2,10));
    when "00000100011" => addr_f_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(capture_time(7 downto 4)))*16 +
(to_integer(vcnt_lat2_p)-532)*2,10));
    when "00000100100" => addr_f_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(capture_time(3 downto 0)))*16 +
(to_integer(vcnt_lat2_p)-532)*2,10));
    when "00000100101" => addr_f_n <=
std_logic_vector(to_unsigned(0*16 +
(to_integer(vcnt_lat2_p)-532)*2,10));
    when "00000100111" => addr_f_n <=
std_logic_vector(to_unsigned(23*16 +
(to_integer(vcnt_lat2_p)-532)*2,10));
    when "00000101000" => addr_f_n <=
std_logic_vector(to_unsigned(28*16 +
(to_integer(vcnt_lat2_p)-532)*2,10));
    when others => addr_f_n <= addr_f_p;
   end case;
  elsif vcnt_lat2_p >= 548 and vcnt_lat2_p <
556 then
   case (hcnt_lat2_p-400)/8 is
    when "00000000000" => addr_f_n <=
std_logic_vector(to_unsigned(18*16 +
(to_integer(vcnt_lat2_p)-548)*2,10));
    when "00000000010" => addr_f_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(addr_proj_p(8 downto 8)))*16
+(to_integer(vcnt_lat2_p)-548)*2,10));
    when "00000000011" => addr_f_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(addr_proj_p(7 downto 4)))*16 +
(to_integer(vcnt_lat2_p)-548)*2,10));
    when "00000000100" => addr_f_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(addr proj p(3 downto 0)))*16 +
(to_integer(vcnt_lat2_p)-548)*2,10));
    when "00000000110" => addr_f_n <=
std_logic_vector(to_unsigned(31*16 +
(to_integer(vcnt_lat2_p)-548)*2,10));
    when "00000001000" => addr_f_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(proj_p(8 downto 8)))*16 +
(to_integer(vcnt_lat2_p)-548)*2,10));
    when "00000001001" => addr_f_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(proj_p(7 downto 4)))*16 +
(to_integer(vcnt_lat2_p)-548)*2,10));
    when "00000001010" => addr_f_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(proj_p(3 downto 0)))*16 +
(to_integer(vcnt_lat2_p)-548)*2,10));
    when others => addr_f_n <= addr_f_p;
   end case;
  elsif vcnt_lat2_p >= 564 and vcnt_lat2_p <
572 then
   case (hcnt_lat2_p-400)/8 is
    when "00000000000" => addr_f_n <=
std_logic_vector(to_unsigned(33*16 +
(to_integer(vcnt_lat2_p)-564)*2,10));
    when "00000000010" => addr_f_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(pixel_x_p(8 downto 8)))*16 +
(to_integer(vcnt_lat2_p)-564)*2,10));
    when "00000000011" => addr f n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(pixel_x_p(7 downto 4)))*16 +
(to_integer(vcnt_lat2_p)-564)*2,10));
    when "00000000100" => addr_f_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(pixel_x_p(3 downto 0)))*16 +
(to_integer(vcnt_lat2_p)-564)*2,10));
    when "00000000110" => addr_f_n <=
std_logic_vector(to_unsigned(34*16 +
(to_integer(vcnt_lat2_p)-564)*2,10));
    when "00000001000" => addr_f_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(pixel_y_p(7 downto 4)))*16 +
(to_integer(vcnt_lat2_p)-564)*2,10));
    when "00000001001" => addr_f_n <=
std_logic_vector(to_unsigned(to_integer(unsi
gned(pixel_y_p(3 downto 0)))*16 +
(to_integer(vcnt_lat2_p)-564)*2,10));
    when "00000001011" => addr_f_n <=
std_logic_vector(to_unsigned(31*16 +
(to_integer(vcnt_lat2_p)-564)*2,10));
    when others => addr_f_n <= addr_f_p;
   end case;
  elsif vcnt_lat2_p >= 580 and vcnt_lat2_p <
588 then
   case (hcnt_lat2_p-400)/8 is
    when "00000000000" => addr_f_n <=
std_logic_vector(to_unsigned(28*16 +
(to_integer(vcnt_lat2_p)-580)*2,10));
    when "00000000001" => addr_f_n <=
std_logic_vector(to_unsigned(29*16 +
(to_integer(vcnt_lat2_p)-580)*2,10));
    when "00000000010" => addr_f_n <=
std_logic_vector(to_unsigned(10*16 +
(to_integer(vcnt_lat2_p)-580)*2,10));
    when "00000000011" => addr_f_n <=
std_logic_vector(to_unsigned(29*16 +
(to_integer(vcnt_lat2_p)-580)*2,10));
    when "00000000100" => addr_f_n <=
std_logic_vector(to_unsigned(14*16 +
(to_integer(vcnt_lat2_p)-580)*2,10));
    when others => addr_f_n <= addr_f_p;
   end case;
```

```
  end if;
  addr_ci_n <=
std_logic_vector(to_unsigned(((to_integer(hc
nt_lat2_p)-440)/32)*16 +
(to_integer(vcnt lat2 p)-432),8));
  addr_c_n <=
std_logic_vector(to_unsigned(((to_integer(hc
nt_p)-440)/32),4));

  ce_hcnt <= '0';
  ce_vcnt <= '0';

  addr_proj_n <= addr_proj_p;
  if latch_next_index = "00001111" or
(hold_ni_p = 5e7 and hcnt_p = 0 and vcnt_p =
0) then
   if next_index_dir = '1' then
    if addr_proj_p(11) = '0' and
addr_proj_p(9) = '0' and
unsigned(addr_proj_p(8 downto 0)) = 319 then
     addr_proj_n(11 downto 9) <=
std_logic_vector(unsigned(addr_proj_p(11
downto 9))+1);
     addr_proj_n(8 downto 0) <= (others =>
'0');
    elsif addr_proj_p(11) = '0' and
addr_proj_p(9) = '1' and
unsigned(addr_proj_p(8 downto 0)) = 239 then
     addr_proj_n(11 downto 9) <=
std_logic_vector(unsigned(addr_proj_p(11
downto 9))+1);
     addr_proj_n(8 downto 0) <= (others =>
'0');
    elsif addr proj p(11 downto 9) = "100"
and unsigned(addr_proj_p(8 downto 0)) = 35
then
     addr_proj_n(11 downto 9) <= (others =>
'0');
     addr_proj_n(8 downto 0) <= (others =>
'0');
    else
     addr_proj_n(8 downto 0) <=
std_logic_vector(unsigned(addr_proj_p(8
downto 0))+1);
    end if;
   else
    if addr_proj_p(11 downto 9) = "000" and
unsigned(addr_proj_p(8 downto 0)) = 0 then
     addr_proj_n(11 downto 9) <= "100";
     addr_proj_n(8 downto 0) <=
std_logic_vector(to_unsigned(35, 9));
    elsif addr_proj_p(11) = '0' and
addr_proj_p(9) = '1' and
unsigned(addr_proj_p(8 downto 0)) = 0 then
     addr_proj_n(11 downto 9) <=
std_logic_vector(unsigned(addr_proj_p(11
downto 9))-1);
     addr_proj_n(8 downto 0) <=
std_logic_vector(to_unsigned(319, 9));
    elsif (addr_proj_p(11 downto 9) = "010"
or addr_proj_p(11 downto 9) = "100") and
unsigned(addr_proj_p(8 downto 0)) = 0 then
     addr_proj_n(11 downto 9) <=
std_logic_vector(unsigned(addr_proj_p(11
downto 9))-1);
     addr_proj_n(8 downto 0) <=
std_logic_vector(to_unsigned(239, 9));
    else
     addr_proj_n(8 downto 0) <=
std_logic_vector(unsigned(addr_proj_p(8
downto 0))-1);
    end if;
   end if;
  end if;

  pixel_x_n <= pixel_x_p;
  pixel_y_n <= pixel_y_p;
  if latch_next_pixel = "00001111" or
(hold_np_p = 5e7 and hcnt_p = 0 and vcnt_p =
0) then
   if next_pixel_dir = '1' then
    if next pixel ori = '1' then
     if unsigned(pixel_y_p) = 239 then
      pixel_y_n <= (others => '0');
     else
      pixel_y_n <=
std_logic_vector(unsigned(pixel_y_p)+1);
     end if;
    else
     if unsigned(pixel_x_p) = 319 then
      pixel_x_n <= (others => '0');
     else
      pixel_x_n <=
std_logic_vector(unsigned(pixel_x_p)+1);
     end if;
    end if;
   else
    if next_pixel_ori = '1' then
     if unsigned(pixel_y_p) = 0 then
      pixel_y_n <=
std_logic_vector(to_unsigned(239, 8));
     else
      pixel_y_n <=
std_logic_vector(unsigned(pixel_y_p)-1);
     end if;
    else
     if unsigned(pixel_x_p) = 0 then
      pixel_x_n <=
std_logic_vector(to_unsigned(319, 9));
     else
      pixel x n <=
std_logic_vector(unsigned(pixel_x_p)-1);
     end if;
    end if;
   end if;
  end if;

  proj_n <= proj_p;
  case unsigned(addr_proj_p(11 downto 9)) is
   when "000" =>
    if addr_proj_p(8 downto 0) =
addr_px1_lat1_p then
     proj_n <= "0" & din_px1;
    end if;
   when "001" =>
    if addr_proj_p(7 downto 0) =
addr_py1_lat1_p then
     proj_n <= din_py1;
    end if;
   when "010" =>
    if addr_proj_p(8 downto 0) =
addr_px2_lat1_p then
     proj_n <= "0" & din_px2;
    end if;
   when "011" =>
    if addr_proj_p(7 downto 0) =
addr_py2_lat1_p then
     proj_n <= din_py2;
    end if;
   when "100" =>
    if addr_proj_p(5 downto 0) =
addr_d_lat1_p then
     proj_n <= din_d;
    end if;
   when others =>
    proj_n <= proj_p;
  end case;

  pixel_dat_n <= pixel_dat_p;
  if hcnt_lat4_p < 320 then
   if vcnt_lat4_p < 240 then
    if unsigned(pixel_x_p) = hcnt_lat4_p and
unsigned(pixel_y_p) = vcnt_lat4_p then
     pixel_dat_n(3 downto 1) <= din1;
```

```
    end if;
   elsif vcnt_lat4_p < 480 then
    if unsigned(pixel_x_p) = hcnt_lat4_p and
unsigned(pixel_y_p) = vcnt_lat4_p-240 then
     pixel dat n(3 downto 1) <= din1;
    end if;
   end if;
  elsif hcnt_lat4_p >= 400 and hcnt_lat4_p <
720 and vcnt_lat4_p < 240 then
   if unsigned(pixel_x_p) = hcnt_lat4_p-400
and unsigned(pixel_y_p) = vcnt_lat4_p then
    pixel_dat_n(0) <= din2;
   end if;
  end if;

  y0_n <= y0_p;
  y1_n <= y1_p;
  if bounds_addr = addr_b_lat1_p then
   y0_n <= din_by0;
   y1_n <= din_by1;
  end if;

  cdm_n <= cdm_p;
  if bounds_addr = addr_c_lat1_p then
   cdm_n <= din_c;
  end if;

  hold_ni_n <= hold_ni_p+1;
  ce_hold_ni <= latch_next_index(0);
  if hold_ni_p = 5e7 then
   ce_hold_ni <= '0';
  end if;

  hold np n <= hold np p+1;
  ce_hold_np <= latch_next_pixel(0);
  if hold_np_p = 5e7 then
   ce_hold_np <= '0';
  end if;

  --main:
  if hcnt_lat1_p = 0 then
   if vblank_p = '0' then
    active_n <= '1';
   end if;
  end if;
  if hcnt_lat1_p = 800 then
   active_n <= '0';
  end if;
  if hcnt_lat4_p = 856 then
   hs_n <= '0';
  end if;
  if hcnt_lat4_p = 976 then
   hs_n <= '1';
  end if;

  if vcnt_p = 0 then
   vblank_n <= '0';
  end if;
  if vcnt_p = 600 then
   vblank_n <= '1';
  end if;
  if vcnt_lat4_p = 637 then
   vs_n <= '0';
  end if;
  if vcnt_lat4_p = 643 then
   vs_n <= '1';
  end if;

  if hcnt_lat3_p < 320 then
   if unsigned(addr_b_p) <
unsigned(bounds_count)-1 then
    if hcnt_lat3_p = unsigned(din_bx1)+1
then
     ce_addr_b <= '1';
    end if;
   end if;
  elsif hcnt_lat3_p >= 400 and hcnt_lat3_p <
720 then
   if unsigned(addr_b_p) <
unsigned(bounds_count)-1 then
    if hcnt_lat3_p-400 = unsigned(din_bx1)+1
then
     ce_addr_b <= '1';
    end if;
   end if;
  else
   r_addr_b <= '1';
  end if;

  if clk_div_p = '1' then
   if hcnt_p = 1039 then
    ce_vcnt <= '1';
   end if;
   ce_hcnt <= '1';
  end if;

  if hcnt_p = 0 and vcnt_p = 0 then
   icr_btn <= '1';
  else
   icr_btn <= '0';
  end if;

 end process;

 output_process: process(hs_p, vs_p,
active_lat3_p, r_p, g_p, b_p, addr1_p,
addr2_p, addr_py1_p, addr_px1_p, addr_px2_p,
addr_py2_p, addr_b_p, addr_f_p, addr_c_p,
addr_ci_p, addr_d_p)
 begin
  hs <= hs_p;
  vs <= vs_p;
  active <= active_lat3_p;
  r <= r_p;
  g <= g_p;
  b <= b_p;
  addr1 <= addr1_p;
  addr2 <= addr2_p;
  addr_py1 <= addr_py1_p;
  addr_px1 <= addr_px1_p;
  addr_px2 <= addr_px2_p;
  addr_py2 <= addr_py2_p;
  addr_b <= addr_b_p;
  addr_f <= addr_f_p;
  addr_c <= addr_c_p;
  addr_ci <= addr_ci_p;
  addr_d <= addr_d_p;
 end process;

end behavioral;
```

8. Program Modul Blok Sistem Keseluruhan

File: work_package.vhd

```
library ieee;
use ieee.std_logic_1164.all;

package work_package is
 type state_type is (idle, capture_image,
y0_first, y1_first, px1, x_first, x_second,
py2, y0 second, y1 second, init res,
init_l_res, res, l_res, l_res2, dist, dm);
end work_package;

package body work_package is
end work_package;
```

File: top_module.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use work.work_package.all;

entity top_module is
 port (
  clk_100mhz : in std_logic;
  reset : in std_logic;
  capture : in std_logic;
  pause : in std_logic;
  pause_en : in std_logic;
  next_addr : in std_logic;
  next_addr_dir : in std_logic;
  next_index : in std_logic;
  next_index_dir : in std_logic;
  next_pixel : in std_logic;
  next_pixel_ori : in std_logic;
  next_pixel_dir : in std_logic;
  threshold_nextval : in std_logic;
  threshold_nextval_dir : in std_logic;
  ov7670_sioc : out std_logic;
  ov7670_siod : inout std_logic;
  ov7670_pclk : in std_logic;
  ov7670_xclk : out std_logic;
  ov7670_vsync : in std_logic;
  ov7670_href : in std_logic;
  ov7670_data : in std_logic_vector(7 downto
0);
  ov7670_reset_1 : out std_logic;
  ov7670_pwdn : out std_logic;
  vga_r : out std_logic_vector(2 downto 0);
  vga_g : out std_logic_vector(2 downto 0);
  vga_b : out std_logic_vector(2 downto 1);
  vga_hs : out std_logic;
  vga_vs : out std_logic;
  config_finished : out std_logic
 );
end top_module;

architecture behavioral of top_module is
component anpr_module is
 port (
  clk_100mhz : in std_logic;
  reset : in std_logic;
  capture : in std_logic;
  pause : in std_logic;
  pause_en : in std_logic;
  next_addr : in std_logic;
  next_addr_dir : in std_logic;
  threshold_nextval : in std_logic;
  threshold_nextval_dir : in std_logic;
  ov7670_vsync : in std_logic;
  state_capture_image : out std_logic;
  cntx : out unsigned(9 downto 0);
  cnty : out unsigned(8 downto 0);
  ov7670_capture_ce_cntx : in std_logic;
  ov7670_capture_r_acc : in std_logic;
  ov7670_capture_dout1 : in
std_logic_vector(2 downto 0);
  ov7670_capture_we1 : in std_logic;
  ov7670_capture_dout2 : in std_logic;
  ov7670_capture_we2 : in std_logic;
  ov7670_capture_finished : in std_logic;
  vga_driver_addr1 : in std_logic_vector(16
downto 0);
  vga_driver_din1 : out std_logic_vector(2
downto 0);
  vga_driver_addr2 : in std_logic_vector(16
downto 0);
  vga_driver_din2 : out std_logic;
  vga_driver_addr_py1 : in
std_logic_vector(7 downto 0);
  vga_driver_din_py1 : out
std_logic_vector(8 downto 0);
  ym1 : out std_logic_vector(7 downto 0);
  y0_1 : out std_logic_vector(7 downto 0);
  y1_1 : out std_logic_vector(7 downto 0);
  vga_driver_addr_px1 : in
std_logic_vector(8 downto 0);
  vga_driver_din_px1 : out
std_logic_vector(7 downto 0);
  xm1 : out std_logic_vector(8 downto 0);
  xmw1 : out std_logic_vector(8 downto 0);
  x0_1 : out std_logic_vector(8 downto 0);
  x1_1 : out std_logic_vector(8 downto 0);
  vga_driver_addr_px2 : in
std_logic_vector(8 downto 0);
  vga_driver_din_px2 : out
std_logic_vector(7 downto 0);
  vga_driver_addr_py2 : in
std_logic_vector(11 downto 0);
  vga_driver_din_py2 : out
std_logic_vector(8 downto 0);
  vga_driver_ym2 : out std_logic_vector(7
downto 0);
  bounds_addr : out std_logic_vector(3
downto 0);
  bounds_count : out std_logic_vector(3
downto 0);
  vga_driver_addr_b : in std_logic_vector(3
downto 0);
  vga_driver_din_bx0 : out
std_logic_vector(8 downto 0);
  vga_driver_din_bx1 : out
std_logic_vector(8 downto 0);
  vga_driver_din_by0 : out
std_logic_vector(7 downto 0);
  vga_driver_din_by1 : out
std_logic_vector(7 downto 0);
  vga_driver_addr_f : in std_logic_vector(9
downto 0);
  vga_driver_din_f : out std_logic_vector(15
downto 0);
  vga_driver_addr_c : in std_logic_vector(3
downto 0);
  vga_driver_din_c : out std_logic_vector(5
downto 0);
  vga_driver_addr_ci : in std_logic_vector(7
downto 0);
  vga_driver_din_ci : out
std_logic_vector(15 downto 0);
  vga_driver_addr_d : in std_logic_vector(8
downto 0);
  vga_driver_din_d : out std_logic_vector(8
downto 0);
  state : out state_type;
  ce : out std_logic;
  threshold : out std_logic_vector(9 downto
0);
  processing_time : out std_logic_vector(31
downto 0);
  capture_time : out std_logic_vector(31
downto 0);
  icr_btn : in std_logic
 );
end component;

component ov7670_capture is
 port (
  clk_100mhz : in std_logic;
  pclk : in std_logic;
  reset : in std_logic;
  vsync : in std_logic;
  href : in std_logic;
  capture : in std_logic;
```

```
  threshold : in std_logic_vector(9 downto
0);
  din : in std_logic_vector(7 downto 0);
  cntx : in unsigned(9 downto 0);
  cnty : in unsigned(8 downto 0);
  ce_cntx : out std_logic;
  r_acc : out std_logic;
  dout1 : out std_logic_vector(2 downto 0);
  we1 : out std_logic;
  dout2 : out std_logic;
  we2 : out std_logic;
  finished : out std_logic;
  pixel_clock : out std_logic_vector(7
downto 0)
 );
end component;

component ov7670_controller is
 port (
  clk_100mhz : in std_logic;
  sccb_busy : in std_logic;
  start_config : in std_logic;
  reset : in std_logic;
  send : out std_logic;
  rw : out std_logic;
  addr : out std_logic_vector(7 downto 0);
  data : out std_logic_vector(7 downto 0);
  busy : out std_logic;
  config_finished : out std_logic
 );
end component;

component ov7670_sccb is
 port (
  clk_100mhz : in std_logic;
  reset : in std_logic;
  pwdn_trig : in std_logic;
  send : in std_logic;
  rw : in std_logic;
  addr : in std_logic_vector(7 downto 0);
  din : in std_logic_vector(7 downto 0);
  dout : out std_logic_vector(7 downto 0);
  sioc : out std_logic;
  siod : inout std_logic;
  busy : out std_logic;
  pwdn : out std_logic
 );
end component;

component vga_driver is
 port (
  clk_100mhz : in std_logic;
  reset : in std_logic;
  next_index : in std_logic;
  next_index_dir : in std_logic;
  next_pixel : in std_logic;
  next_pixel_ori : in std_logic;
  next_pixel_dir : in std_logic;
  r : out std_logic_vector(2 downto 0);
  g : out std_logic_vector(2 downto 0);
  b : out std_logic_vector(2 downto 1);
  addr1 : out std_logic_vector(16 downto 0);
  din1 : in std_logic_vector(2 downto 0);
  addr2 : out std_logic_vector(16 downto 0);
  din2 : in std_logic;
  addr_py1 : out std_logic_vector(7 downto
0);
  din_py1 : in std_logic_vector(8 downto 0);
  ym1 : in std_logic_vector(7 downto 0);
  y0_1 : in std_logic_vector(7 downto 0);
  y1_1 : in std_logic_vector(7 downto 0);
  addr_px1 : out std_logic_vector(8 downto
0);
  din_px1 : in std_logic_vector(7 downto 0);
  xm1 : in std_logic_vector(8 downto 0);
  xmw1 : in std_logic_vector(8 downto 0);
  x0_1 : in std_logic_vector(8 downto 0);
  x1_1 : in std_logic_vector(8 downto 0);
  addr_px2 : out std_logic_vector(8 downto
0);
  din_px2 : in std_logic_vector(7 downto 0);
  addr py2 : out std logic vector(11 downto
0);
  din_py2 : in std_logic_vector(8 downto 0);
  ym2 : in std_logic_vector(7 downto 0);
  bounds_addr : in std_logic_vector(3 downto
0);
  bounds_count : in std_logic_vector(3
downto 0);
  addr_b : out std_logic_vector(3 downto 0);
  din_bx0 : in std_logic_vector(8 downto 0);
  din_bx1 : in std_logic_vector(8 downto 0);
  din_by0 : in std_logic_vector(7 downto 0);
  din_by1 : in std_logic_vector(7 downto 0);
  addr_f : out std_logic_vector(9 downto 0);
  din_f : in std_logic_vector(15 downto 0);
  addr_c : out std_logic_vector(3 downto 0);
  din_c : in std_logic_vector(5 downto 0);
  addr_ci : out std_logic_vector(7 downto
0);
  din_ci : in std_logic_vector(15 downto 0);
  addr_d : out std_logic_vector(8 downto 0);
  din_d : in std_logic_vector(8 downto 0);
  state : in state_type;
  ce : in std_logic;
  capture : in std_logic;
  threshold : in std_logic_vector(9 downto
0);
  processing_time : in std_logic_vector(31
downto 0);
  capture time : in std logic vector(31
downto 0);
  icr_btn : out std_logic;
  pixel_clock : in std_logic_vector(7 downto
0);
  hs : out std_logic;
  vs : out std_logic;
  active : out std_logic
 );
end component;

signal ov7670_sccb_send : std_logic;
signal ov7670_sccb_rw : std_logic;
signal ov7670_sccb_addr : std_logic_vector(7
downto 0);
signal ov7670_sccb_din : std_logic_vector(7
downto 0);
signal ov7670_sccb_busy : std_logic;
signal start_config : std_logic;
signal ov7670_capture_ce_cntx : std_logic;
signal ov7670_capture_r_acc : std_logic;
signal ov7670_capture_dout1 :
std_logic_vector(2 downto 0);
signal ov7670_capture_we1 : std_logic;
signal ov7670_capture_dout2 : std_logic;
signal ov7670_capture_we2 : std_logic;
signal ov7670_capture_finished : std_logic;
signal ov7670_controller_config_finished :
std_logic;
signal ov7670_controller_busy : std_logic;
signal clk_25mhz : std_logic := '0';
signal clk_50mhz : std_logic := '0';
signal pixel_clock : std_logic_vector(7
downto 0);
signal state_capture_image : std_logic;
signal cntx : unsigned(9 downto 0);
signal cnty : unsigned(8 downto 0);
signal vga_driver_addr1 :
std_logic_vector(16 downto 0);
signal vga_driver_din1 : std_logic_vector(2
downto 0);
signal vga_driver_addr2 :
std_logic_vector(16 downto 0);
signal vga_driver_din2 : std_logic;
```

```
signal vga_driver_addr_py1 :
std_logic_vector(7 downto 0);
signal vga_driver_din_py1 :
std_logic_vector(8 downto 0);
signal ym1 : std_logic_vector(7 downto 0);
signal y0_1 : std_logic_vector(7 downto 0);
signal y1_1 : std_logic_vector(7 downto 0);
signal vga_driver_addr_px1 :
std_logic_vector(8 downto 0);
signal vga_driver_din_px1 :
std_logic_vector(7 downto 0);
signal xm1 : std_logic_vector(8 downto 0);
signal xmw1 : std_logic_vector(8 downto 0);
signal x0_1 : std_logic_vector(8 downto 0);
signal x1_1 : std_logic_vector(8 downto 0);
signal vga_driver_addr_px2 :
std_logic_vector(8 downto 0);
signal vga_driver_din_px2 :
std_logic_vector(7 downto 0);
signal vga_driver_addr_py2 :
std_logic_vector(11 downto 0);
signal vga_driver_din_py2 :
std_logic_vector(8 downto 0);
signal vga_driver_ym2 : std_logic_vector(7
downto 0);
signal bounds_addr : std_logic_vector(3
downto 0);
signal bounds_count : std_logic_vector(3
downto 0);
signal vga_driver_addr_b :
std_logic_vector(3 downto 0);
signal vga_driver_din_bx0 :
std_logic_vector(8 downto 0);
signal vga driver din bx1 :
std_logic_vector(8 downto 0);
signal vga_driver_din_by0 :
std_logic_vector(7 downto 0);
signal vga_driver_din_by1 :
std_logic_vector(7 downto 0);
signal vga_driver_addr_f :
std_logic_vector(9 downto 0);
signal vga_driver_din_f :
std_logic_vector(15 downto 0);
signal vga_driver_addr_c :
std_logic_vector(3 downto 0);
signal vga_driver_din_c : std_logic_vector(5
downto 0);
signal vga_driver_addr_ci :
std_logic_vector(7 downto 0);
signal vga_driver_din_ci :
std_logic_vector(15 downto 0);
signal vga_driver_addr_d :
std_logic_vector(8 downto 0);
signal vga_driver_din_d : std_logic_vector(8
downto 0);
signal state : state_type;
signal ce : std_logic;
signal threshold : std_logic_vector(9 downto
0);
signal processing_time : std_logic_vector(31
downto 0);
signal capture_time : std_logic_vector(31
downto 0);
signal icr_btn : std_logic;
begin

 inst_anpr_module: anpr_module port map (
  clk_100mhz => clk_100mhz,
  reset => reset,
  capture => capture,
  pause => pause,
  pause_en => pause_en,
  next_addr => next_addr,
  next_addr_dir => next_addr_dir,
  threshold_nextval => threshold_nextval,
  threshold_nextval_dir =>
threshold_nextval_dir,
  ov7670_vsync => ov7670_vsync,
  state_capture_image =>
state_capture_image,
  cntx => cntx,
  cnty => cnty,
  ov7670_capture_ce_cntx =>
ov7670_capture_ce_cntx,
  ov7670_capture_r_acc =>
ov7670_capture_r_acc,
  ov7670_capture_dout1 =>
ov7670_capture_dout1,
  ov7670_capture_we1 => ov7670_capture_we1,
  ov7670_capture_dout2 =>
ov7670_capture_dout2,
  ov7670_capture_we2 => ov7670_capture_we2,
  ov7670_capture_finished =>
ov7670_capture_finished,
  vga_driver_addr1 => vga_driver_addr1,
  vga_driver_din1 => vga_driver_din1,
  vga_driver_addr2 => vga_driver_addr2,
  vga_driver_din2 => vga_driver_din2,
  vga_driver_addr_py1 =>
vga_driver_addr_py1,
  vga_driver_din_py1 => vga_driver_din_py1,
  ym1 => ym1,
  y0_1 => y0_1,
  y1_1 => y1_1,
  vga_driver_addr_px1 =>
vga_driver_addr_px1,
  vga_driver_din_px1 => vga_driver_din_px1,
  xm1 => xm1,
  xmw1 => xmw1,
  x0_1 => x0_1,
  x1_1 => x1_1,
  vga_driver_addr_px2 =>
vga_driver_addr_px2,
  vga_driver_din_px2 => vga_driver_din_px2,
  vga_driver_addr_py2 =>
vga_driver_addr_py2,
  vga_driver_din_py2 => vga_driver_din_py2,
  vga_driver_ym2 => vga_driver_ym2,
  bounds_addr => bounds_addr,
  bounds_count => bounds_count,
  vga_driver_addr_b => vga_driver_addr_b,
  vga_driver_din_bx0 => vga_driver_din_bx0,
  vga_driver_din_bx1 => vga_driver_din_bx1,
  vga_driver_din_by0 => vga_driver_din_by0,
  vga_driver_din_by1 => vga_driver_din_by1,
  vga_driver_addr_f => vga_driver_addr_f,
  vga_driver_din_f => vga_driver_din_f,
  vga_driver_addr_c => vga_driver_addr_c,
  vga_driver_din_c => vga_driver_din_c,
  vga_driver_addr_ci => vga_driver_addr_ci,
  vga_driver_din_ci => vga_driver_din_ci,
  vga_driver_addr_d => vga_driver_addr_d,
  vga_driver_din_d => vga_driver_din_d,
  state => state,
  ce => ce,
  threshold => threshold,
  processing_time => processing_time,
  capture_time => capture_time,
  icr_btn => icr_btn
 );

 inst_ov7670_capture: ov7670_capture port
map (
  clk_100mhz => clk_100mhz,
  pclk => ov7670_pclk,
  reset => reset,
  vsync => ov7670_vsync,
  href => ov7670_href,
  capture => state_capture_image,
      threshold => threshold,
  din => ov7670_data,
  cntx => cntx,
  cnty => cnty,
  ce_cntx => ov7670_capture_ce_cntx,
```

```
  r_acc => ov7670_capture_r_acc,
  dout1 => ov7670_capture_dout1,
  we1 => ov7670_capture_we1,
  dout2 => ov7670_capture_dout2,
  we2 => ov7670 capture we2,
  finished => ov7670_capture_finished,
  pixel_clock => pixel_clock
 );

 inst_ov7670_controller: ov7670_controller
port map (
  clk_100mhz => clk_100mhz,
  reset => reset,
  send => ov7670_sccb_send,
  rw => ov7670_sccb_rw,
  addr => ov7670_sccb_addr,
  data => ov7670_sccb_din,
  sccb_busy => ov7670_sccb_busy,
  start_config => start_config,
  busy => ov7670_controller_busy,
  config_finished =>
ov7670 controller config finished
 );

 inst_ov7670_sccb: ov7670_sccb port map (
  clk_100mhz => clk_100mhz,
  reset => reset,
  send => ov7670_sccb_send,
  rw => ov7670_sccb_rw,
  addr => ov7670_sccb_addr,
  din => ov7670_sccb_din,
  busy => ov7670_sccb_busy,
  pwdn_trig => '0',
  dout => open,
  sioc => ov7670_sioc,
  siod => ov7670_siod,
  pwdn => ov7670_pwdn
 );

 inst_vga_driver: vga_driver port map (
  clk_100mhz => clk_100mhz,
  reset => reset,
  next_index => next_index,
  next_index_dir => next_index_dir,
  next_pixel => next_pixel,
  next_pixel_ori => next_pixel_ori,
  next_pixel_dir => next_pixel_dir,
  r => vga_r,
  g => vga_g,
  b => vga_b,
  addr1 => vga_driver_addr1,
  din1 => vga_driver_din1,
  addr2 => vga_driver_addr2,
  din2 => vga_driver_din2,
  addr_py1 => vga_driver_addr_py1,
  din_py1 => vga_driver_din_py1,
  ym1 => ym1,
  y0_1 => y0_1,
  y1_1 => y1_1,
  addr_px1 => vga_driver_addr_px1,
  din_px1 => vga_driver_din_px1,
  xm1 => xm1,
  xmw1 => xmw1,
  x0_1 => x0_1,
  x1_1 => x1_1,
  addr_px2 => vga_driver_addr_px2,
  din_px2 => vga_driver_din_px2,
  addr_py2 => vga_driver_addr_py2,
  din_py2 => vga_driver_din_py2,
  ym2 => vga_driver_ym2,
  bounds_addr => bounds_addr,
  bounds_count => bounds_count,
  addr_b => vga_driver_addr_b,
  din_bx0 => vga_driver_din_bx0,
  din_bx1 => vga_driver_din_bx1,
  din_by0 => vga_driver_din_by0,
  din_by1 => vga_driver_din_by1,
  addr_f => vga_driver_addr_f,
  din_f => vga_driver_din_f,
  addr_c => vga_driver_addr_c,
  din_c => vga_driver_din_c,
  addr_ci => vga_driver_addr_ci,
  din_ci => vga_driver_din_ci,
  addr_d => vga_driver_addr_d,
  din_d => vga_driver_din_d,
  state => state,
  ce => ce,
  capture => capture,
  threshold => threshold,
  processing_time => processing_time,
  capture_time => capture_time,
  icr btn => icr btn,
  pixel_clock => pixel_clock,
  hs => vga_hs,
  vs => vga_vs,
  active => open
 );

 ov7670_reset_1 <= not reset;
 start_config <= (not
ov7670_controller_config_finished) and (not
ov7670_controller_busy);
 config_finished <=
ov7670_controller_config_finished;
 ov7670_xclk <= clk_25mhz;

 clock_process: process(clk_100mhz)
 begin
  if rising_edge(clk_100mhz) then
   if clk_50mhz = '0' then
    clk_25mhz <= not clk_25mhz;
   end if;
   clk_50mhz <= not clk_50mhz;
  end if;
 end process;
end behavioral;
```

9. Alokasi Pin Nexys3

File: Nexys3_Master.ucf

```
net "clk 100mhz" loc=v10 |
iostandard=lvcmos33;
net "clk_100mhz" tnm_net = sys_clk_pin;
timespec ts sys clk pin = period sys clk pin
100000 khz;

net "ov7670 pclk" clock dedicated route =
false;

## leds
net "config finished" loc = u16 | iostandard
= lvcmos33;

## switches
net "reset" loc = t10 | iostandard =
lvcmos33;
net "capture" loc = t9 | iostandard =
lvcmos33;
net "pause en" loc = v9 | iostandard =
lvcmos33;
net "next addr dir" loc = m8 | iostandard =
lvcmos33;
net "threshold_nextval_dir" loc = n8 |
iostandard = lvcmos33;
net "next index dir" loc = u8 | iostandard =
lvcmos33;
```

```
net "next_pixel_ori" loc = v8 | iostandard =
lvcmos33;
net "next pixel dir" loc = t5 | iostandard =
lvcmos33;

## buttons
net "pause" loc = b8 | iostandard =
lvcmos33;
net "threshold nextval" loc = a8 |
iostandard = lvcmos33;
net "next_addr" loc = c4 | iostandard =
lvcmos33;
net "next_pixel" loc = c9 | iostandard =
lvcmos33;
net "next index" loc = d9 | iostandard =
lvcmos33;

## vga connector
net "vga_r<0>" loc = u7 | iostandard =
lvcmos33;
net "vga r<1>" loc = v7 | iostandard =
lvcmos33;
net "vga_r<2>" loc = n7 | iostandard =
lvcmos33;
net "vga_g<0>" loc = p8 | iostandard =
lvcmos33;
net "vga g<1>" loc = t6 | iostandard =
lvcmos33;
net "vga_g<2>" loc = v6 | iostandard =
lvcmos33;
net "vga b<1>" loc = r7 | iostandard =
lvcmos33;
net "vga b<2>" loc = t7 | iostandard =
lvcmos33;

net "vga_hs" loc = n6 | iostandard =
lvcmos33;
net "vga vs" loc = p7 | iostandard =
lvcmos33;
```

```
## 12 pin connectors

##jb
net "ov7670 pwdn" loc = k2 | iostandard =
lvcmos33;
net "ov7670_data<0>" loc = k1 | iostandard =
lvcmos33 | pulldown;
net "ov7670 data<2>" loc = l4 | iostandard =
lvcmos33 | pulldown;
net "ov7670_data<4>" loc = l3 | iostandard =
lvcmos33 | pulldown;
net "ov7670_reset_1" loc = j3 | iostandard =
lvcmos33;
net "ov7670 data<1>" loc = j1 | iostandard =
lvcmos33 | pulldown;
net "ov7670_data<3>" loc = k3 | iostandard =
lvcmos33 | pulldown;
net "ov7670_data<5>" loc = k5 | iostandard =
lvcmos33 | pulldown;

##jc
net "ov7670_data<6>" loc = h3 | iostandard =
lvcmos33 | pulldown;
net "ov7670_xclk" loc = l7 | iostandard =
lvcmos33 | pulldown;
net "ov7670 href" loc = k6 | iostandard =
lvcmos33;
net "ov7670_siod" loc = g3 | iostandard =
lvcmos33 | pullup;
net "ov7670 data<7>" loc = g1 | iostandard =
lvcmos33 | pulldown;
net "ov7670 pclk" loc = j7 | iostandard =
lvcmos33;
net "ov7670_vsync" loc = j6 | iostandard =
lvcmos33;
net "ov7670 sioc" loc = f2 | iostandard =
lvcmos33;
```

10. Program Simulasi *Timing Diagram* Blok Antarmuka SCCB

File: tb_ov7670_sccb.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity tb_ov7670_sccb is
end tb_ov7670_sccb;

architecture behavior of tb_ov7670_sccb is
-- component declaration for the unit under
test (uut)
component ov7670_sccb
 port (
  clk_100mhz : in std_logic;
  reset : in std_logic;
  pwdn_trig : in std_logic;
  send : in std_logic;
  rw : in std_logic;
  addr : in std_logic_vector(7 downto 0);
  din : in std_logic_vector(7 downto 0);
  dout : out std_logic_vector(7 downto 0);
  sioc : out std_logic;
  siod : inout std_logic;
  busy : out std_logic;
  pwdn : out std_logic
 );
end component;

--inputs
signal clk_100mhz : std_logic := '0';
signal reset : std_logic := '1';
signal pwdn_trig : std_logic := '0';
signal send : std_logic := '0';
```

```
signal rw : std_logic := '0';
signal addr : std_logic_vector(7 downto 0)
:= (others => '0');
signal din : std_logic_vector(7 downto 0) :=
(others => '0');
--bidirs
signal siod : std_logic := 'z';
--outputs
signal dout : std_logic_vector(7 downto 0);
signal sioc : std_logic;
signal busy : std_logic;
signal pwdn : std_logic;
-- clock period definitions
constant clk_100mhz_period : time := 10 ns;
begin
 -- instantiate the unit under test (uut)
 uut: ov7670_sccb port map (
  clk_100mhz => clk_100mhz,
  reset => reset,
  pwdn_trig => pwdn_trig,
  send => send,
  rw => rw,
  addr => addr,
  din => din,
  dout => dout,
  sioc => sioc,
  siod => siod,
  busy => busy,
  pwdn => pwdn
 );

 -- clock process definitions
```

```
 clk_100mhz_process :process
 begin
  clk_100mhz <= '0';
  wait for clk_100mhz_period/2;
  clk_100mhz <= '1';
  wait for clk_100mhz_period/2;
 end process;

 -- stimulus process
 stim_proc: process
 begin
  -- hold reset state for 100 ns.
  wait for 100 ns;
  wait for clk_100mhz_period*10;
  reset <= '0';
  wait;
 end process;
```

```
 -- stimulus process
 stim_proc2: process
 begin
  wait for 100 ns;
  wait for clk_100mhz_period*10;
  send <= '1';
  rw <= '0';
  addr <= x"ab";
  din <= x"cd";
  wait for 100 ns;
  wait for clk_100mhz_period*10;
  send <= '0';
  addr <= x"00";
  din <= x"00";
  wait until busy = '0';
 end process;
end;
```

11. Program Simulasi *Timing Diagram* Blok VGA *Driver*

File: tb_vga_driver.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use work.work_package.all;

entity tb_vga_driver is
end tb_vga_driver;

architecture behavior of tb_vga_driver is
 -- component declaration for the unit under
test (uut)
 component vga_driver
 port (
  clk_100mhz : in std_logic;
  reset : in std_logic;
  next_index : in std_logic;
  next_index_dir : in std_logic;
  next_pixel : in std_logic;
  next_pixel_ori : in std_logic;
  next_pixel_dir : in std_logic;
  r : out std_logic_vector(2 downto 0);
  g : out std_logic_vector(2 downto 0);
  b : out std_logic_vector(2 downto 1);
  addr1 : out std_logic_vector(16 downto 0);
  din1 : in std_logic_vector(2 downto 0);
  addr2 : out std_logic_vector(16 downto 0);
  din2 : in std_logic;
  addr_py1 : out std_logic_vector(7 downto
0);
  din_py1 : in std_logic_vector(8 downto 0);
  ym1 : in std_logic_vector(7 downto 0);
  y0_1 : in std_logic_vector(7 downto 0);
  y1_1 : in std_logic_vector(7 downto 0);
  addr_px1 : out std_logic_vector(8 downto
0);
  din_px1 : in std_logic_vector(7 downto 0);
  xm1 : in std_logic_vector(8 downto 0);
  xmw1 : in std_logic_vector(8 downto 0);
  x0_1 : in std_logic_vector(8 downto 0);
  x1_1 : in std_logic_vector(8 downto 0);
  addr_px2 : out std_logic_vector(8 downto
0);
  din_px2 : in std_logic_vector(7 downto 0);
  addr_py2 : out std_logic_vector(11 downto
0);
  din_py2 : in std_logic_vector(8 downto 0);
  ym2 : in std_logic_vector(7 downto 0);
  bounds_addr : in std_logic_vector(3 downto
0);
  bounds_count : in std_logic_vector(3
downto 0);
  addr_b : out std_logic_vector(3 downto 0);
  din_bx0 : in std_logic_vector(8 downto 0);
  din_bx1 : in std_logic_vector(8 downto 0);
  din_by0 : in std_logic_vector(7 downto 0);
  din_by1 : in std_logic_vector(7 downto 0);
  addr_f : out std_logic_vector(9 downto 0);
  din_f : in std_logic_vector(15 downto 0);
  addr_c : out std_logic_vector(3 downto 0);
  din_c : in std_logic_vector(5 downto 0);
  addr_ci : out std_logic_vector(7 downto
0);
  din_ci : in std_logic_vector(15 downto 0);
  addr_d : out std_logic_vector(8 downto 0);
  din_d : in std_logic_vector(8 downto 0);
  state : in state_type;
  ce : in std_logic;
  capture : in std_logic;
  threshold : in std_logic_vector(9 downto
0);
  processing_time : in std_logic_vector(31
downto 0);
  capture_time : in std_logic_vector(31
downto 0);
  icr_btn : out std_logic;
  pixel_clock : in std_logic_vector(7 downto
0);
  hs : out std_logic;
  vs : out std_logic;
  active : out std_logic
 );
end component;

--inputs
signal clk_100mhz : std_logic := '0';
signal reset : std_logic := '1';
--outputs
signal r : std_logic_vector(2 downto 0);
signal g : std_logic_vector(2 downto 0);
signal b : std_logic_vector(2 downto 1);
signal hs : std_logic;
signal vs : std_logic;
signal active : std_logic;
-- clock period definitions
constant clk_100mhz_period : time := 10 ns;
begin
 -- instantiate the unit under test (uut)
 uut: vga_driver port map (
  clk_100mhz => clk_100mhz,
  reset => reset,
  next_index => '0',
  next_index_dir => '0',
  next_pixel => '0',
  next_pixel_ori => '0',
  next_pixel_dir => '0',
  din1 => (others => '0'),
  din2 => '0',
```

```
din_py1 => (others => '0'),
ym1 => (others => '0'),
y0_1 => (others => '0'),
y1_1 => (others => '0'),
din_px1 => (others => '0'),
xm1 => (others => '0'),
xmw1 => (others => '0'),
x0_1 => (others => '0'),
x1_1 => (others => '0'),
din_px2 => (others => '0'),
din_py2 => (others => '0'),
ym2 => (others => '0'),
bounds_addr => (others => '0'),
bounds_count => (others => '0'),
din_bx0 => (others => '0'),
din_bx1 => (others => '0'),
din_by0 => (others => '0'),
din_by1 => (others => '0'),
din_f => (others => '0'),
din_c => (others => '0'),
din_ci => (others => '0'),
din_d => (others => '0'),
state => idle,
ce => '0',
capture => '0',
threshold => (others => '0'),
processing_time => (others => '0'),
capture_time => (others => '0'),
pixel_clock => (others => '0'),
r => r,
g => g,
b => b,
hs => hs,
vs => vs,
active => active
);

-- clock process definitions
clk_100mhz_process :process
begin
 clk_100mhz <= '0';
 wait for clk_100mhz_period/2;
 clk_100mhz <= '1';
 wait for clk_100mhz_period/2;
end process;

-- stimulus process
stim_proc: process
begin
 -- hold reset state for 100 ns.
 wait for 100 ns;
 wait for clk_100mhz_period*10;
 reset <= '0';
 wait;
end process;
end;
```