Implementasi Discrete Cosine Transform Pada Field Programmable Gate Array

Yan Felix Monangin, Waru Djuriatno ST., MT., Mochammad Rif'an, ST., MT.
Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya
Jalan MT. Haryono 167, Malang 65145, Indonesia
e – mail : yanfelixmonangin@gmail.com

Abstrak

Pengurangan jumlah perkalian, merupakan suatu cara untuk menghasilkan komputasi cepat Discrete Cosine Transform (DCT). Algoritma 1-D DCT Loeffler merupakan modifikasi dari persamaan 1-D DCT klasik yang mampu meminimumkan penggunaan operasi perkalian dari 64 pengali menjadi 11 pengali. Implementasi DCT ke perangkat keras FPGA Xilinx Spartan 3E dilakukan sebagai upaya efektifitas dan untuk mempercepat proses komputasi. Implementasi dengan FPGA artinya kita merancang sebuah perangkat keras langsung dengan cara mengkonfigurasi unit yang ada di FPGA. Pengkonfigurasian unit yang ada di dalam FPGA dapat dilakukan melalui skematic maupun dengan menggunakan *Hardware Description Language* (Verilog atau VHDL). Sumber data untuk implementasi DCT menggunakan mikrokontroler dengan operasi interupsi dengan pemicu *clock* FPGA. Akurasi perhitungan implementasi DCT menunjukkan adanya *error* yang masih dapat ditoleransi terhadap perhitungan DCT dengan MATLAB. Jumlah *slice* yang digunakan untuk implementasi unit 2-D DCT pada penelitian ini sebesar 54 % dari total 4656 *slices.*
Kata kunci : *Discrete Cosine Transform,* Algoritma *Loeffler-Ligtenberg-Moschytz,* FPGA, VHDL, Xilinx ISE.

I PENDAHULUAN

1.1 Latar Belakang

Pengolahan citra dalam hal kompresi citra digital bertujuan untuk efisiensi proses penyampaian informasi. Data digital umumnya disimpan pada perangkat keras dan dapat dikirimkan dalam waktu yang singkat. Dalam proses perubahan data dalam bentuk sinyal analog ke sinyal digital dibutuhkan transformasi [1]. Pada dasarnya beberapa transformasi sudah banyak mengalami perkembangan pesat seperti *Discrete Fourier Transform* (DFT), *Fast Fourier Transform* (FFT), *Discrete Cosines Transform* (DCT). Transformasi DCT merupakan algoritma berbasis cosinus, untuk N masukan akan mengasilkan N keluaran, artinya jika diimplementasikan ke perangkat keras DCT untuk N masukan akan membutuhkan N memori sebagai media penyimpanannya, jika dibandingkan dengan transformasi Fourier yang diimplementasikan ke dalam perangkat keras maka untuk N masukan transformasi Fourier akan membutuhkan 2N memori. Perbandingan pemakaian perangkat keras antara transformasi Fourier dengan transformasi DCT menunjukkan bahwa DCT jauh lebih efisien [5,2].

FPGA adalah komponen elektronika dan semikonduktor yang mempunyai terdiri dari gerbang terprogram dan sambungan terprogram. Komponen gerbang terprogram yang dimiliki meliputi jenis gerbang logika AND, OR, XOR, NOT dan jenis fungsi matematis yanglebih kompleks seperti decoder, adder, subtractor, multiplier, dan lain lain. Blok-blok komponen di dalam FPGA bisa juga mengandung elemen memori (*register*) mulai dari flip-flop sampai pada RAM (*Random Access Memory*). FPGA memiliki beberapa keunggulan yaitu dalam hal kecepatan, kemudahan instalasi dan kemudahan untuk modifikasi bila terjadi kesalahan perancangan. Untuk merancang program ke dalam FPGA salah satunya menggunakan bahasa VHDL (*Very high speed integrated circuit Hardware Description Language*). VHDL adalah bahasa pemrograman untuk mendeskripsikan suatu perangkat keras yang digunakan dalam desain elektronis [2].

Melihat *performance* FPGA di atas, maka transformasi DCT dapat diimplementasikan pada perangkat FPGA (*Field Programmable Gate Array*) agar proses pengolahan citra dalam hal kompresi dapat lebih efisien..

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan diatas, maka rumusan masalah antara lain :

1. Bagaimana proses implementasi *Discrete Cosine Transform* (DCT) pada *Field Programmable Gate Array* (FPGA).
2. Bagaimana cara merancang desain pengali yang digunakan dalam proses komputasi untuk mempercepat proses komputasi.
3. Bagaimana akurasi dan kinerja FPGA saat diimplementasikan sebagai DCT.

1.3 Batasan Masalah

Beberapa hal yang menjadi batasan masalah dalam pembuatan program ini antara lain:

1. Pembahasan difokuskan pada transformasi *Discrete Cosine Transform* (DCT).
2. Input yang digunakan berupa matriks elemen 8x8.
3. FPGA yang digunakan adalah Xilinx Spartan 3E XC3S500E
4. Algoritma yang digunakan adalah algoritma Loeffler.
5. Pembuktian algoritma Loeffler menggunakan Matlab 7.0.4.
6. Sumber data masukan untuk system dihasilkan oleh program yang dikirim melalui mikrokontroller.
7. Matlab 7.0.4 digunakan sebagai referensi dalam membandingkan hasil DCT

1.4 Tujuan

Tujuan penyusunan tugas akhir (skripsi) ini adalah :

1. Merancang dan mengimplementasikan proses transformasi *Discrete Cosine Transform* (DCT) dengan menggunakan *Field Programmable array* (FPGA).
2. Menghasilkan suatu perangkat keras yang mampu melakukan proses transformasi *Discrete Cosine Transform* (DCT) dengan efisien.

II. TINJAUAN PUSTAKA

2.1 *Discrete Cosine Transfrom (DCT)*

Discrete Cosine Transform adalah sebuah teknik untuk mengubah sebuah sinyal ke dalam komponen frekuensi dasar. Discrete Cosine Transform merepresentasikan sebuah citra dari penjumlahan sinusoida dari magnitude dan frekuensi yang berubah-ubah. Sifat dari DCT adalah mengubah informasi citra yang signifkan dikonsentrasikan hanya pada beberapa koefisien DCT [2]

Secara teori DCT melakukan proses transformasi data dari kawasan ruang ke dalam kawasan frekuensi, sedangkan IDCT melakukan proses kebalikan dari DCT. Pada aplikasi kompresi video, operasi DCT dan IDCT bekerja pada 8 x 8 blok data yang merupakan nilai-nilai piksel dari gambar. Untuk 8 x 8 sampel data dari *x(m,n),* dua dimensi 2D-DCT dan 2D-IDCT dirumuskan pada persamaan (2.1) untuk $0 \le k,l \le 7$ dan persamaan (2.2) untuk $0 \le m,n \le 7$.

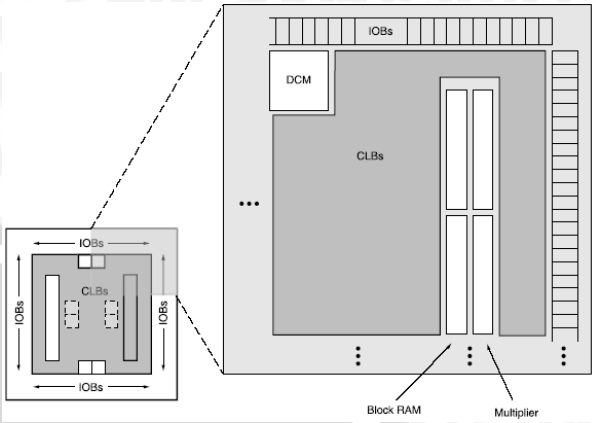
$$Y(k,l) = \frac{1}{4} a\,(k) a(l) \sum_{n=0}^{7} \sum_{m=0}^{7} x(m,n) \cdot \cos\left(\frac{(2m+1)\Pi k}{16}\right) \cos\left(\frac{(2n+1)\Pi l}{16}\right) \quad (2.1)$$

$$x(m,n) = \frac{1}{4} a\,(k) a(l) \sum_{k=0}^{7} \sum_{l=0}^{7} Y(k,l) \cdot \cos\left(\frac{(2m+1)\Pi k}{16}\right) \cos\left(\frac{(2n+1)\Pi l}{16}\right) \quad (2.2)$$

Dengan $\alpha(0) = \frac{1}{2}$ dan $\alpha(j) = 1$ untuk $j \neq 0$ [4].

2.2 *Field Programmable Gate Arrays* (FPGA)

Field Programmable Gate Arrays (FPGA) adalah komponen elektronika dan semikonduktor yang mepunyai komponen gerbang terprogram (*programmable logic*) dan sambungan terpogram. Komponen gerbang terprogram yang dimiliki meliputi jenis gerbang logika biasa (AND, OR, XOR, NOT) maupun jenis fungsi matematis dan kombinatorik yang lebih kompleks (*decoder. adder, subtractor, multiplier, dll*). Blok-blok komponen di dalam FPGA bisa juga mengandung elemen memori (*register*) mulai dari flip-flop sampai pada RAM (*Random Acces Memory*).



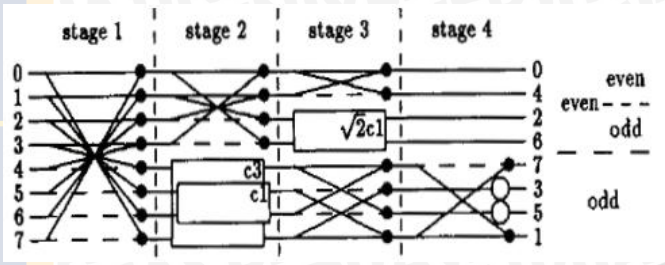
Gambar 2.2 Arsitektur FPGA Xilinx Spartan 3E [6].

Arsitektur FPGA Xilinx Spartan 3E seperti yang terlihat pada gambar 2.2 terdiri dari 5 bagian fungsional [6], yaitu :

1. *Configurable Logic Block* (CLB), setiap CLB mempunyai 4 *slice*, masing-masing *slice* memiliki 2 *Look-Up Tables* (LUT). LUT mengimplementasikan fungsi logika termasuk elemen penyimpanan (*flip-flop* atau *latch*). LUT dapat digunakan sebagai memori 16 x 1 atau sebagai *shift register* 16 bit, dan unit pengali (*multiplier*) tambahan serta fungsi aritmatika. Logika-logika pada desain secara otomatis dipetakan ke dalam *slice* pada CLB-CLB.
2. *Input/Output Block* (IOB), mengatur aliran data antara pin *input/output* dan logika internal rangkaian.
3. *Block RAM*, menyediakan penyimpanan data dalam bentuk blok.
4. *Multiplier Block*,sebagai blok pengali dengan 2 buah *input* biner bertanda 18 bit.
5. *Digital Clock Manager* (DCM) *Block*, mendistribusikan, menunda, menjamak, membagi, dan menggeser fase sinyal *clock*

2.3 Algoritma *Loeffler-Ligtenberg-Moschytz* (LLM).

Dalam implementasi DCT pada perangkat keras, proses komputasi perkalian dan penjumlahan akan membutuhkan daya yang relatif tinggi, terutama pada proses perkalian. Maka dibutuhkan pemilihan algoritma yang tepat agar perangkat keras dapat berjalan efektif dan efisien. Algoritma Loeffler adalah salah satu dari sekian banyak algoritma yang dapat diterapkan dalam implementasi DCT, dalam algoritma ini hanya diperlukan 11 perkalian dan 29 penjumlahan dalam 1D-DCT [3].



Gambar 2.3 Signal Flow Graph (SFG) algoritma Loeffler [3].

Symbol	Equation	Effort
I_0, I_1 → O_0, O_1	$O_0 = (I_0 + I_1)/2$ $O_1 = (I_0 - I_1)/2$	2 add
I_0, I_1 → kcn → O_0, O_1	$O_0 = I_0\left(\frac{1}{k}\cos\frac{n\pi}{2N}\right) - I_1\left(\frac{1}{k}\sin\frac{n\pi}{2N}\right)$ $O_1 = I_0\left(\frac{1}{k}\sin\frac{n\pi}{2N}\right) + I_1\left(\frac{1}{k}\cos\frac{n\pi}{2N}\right)$	3 mult. + 3 add
I — O	$O = I/\sqrt{2}$	1 mult.

Tabel 2.1 Fungsi pada algoritma Loeffler [3].

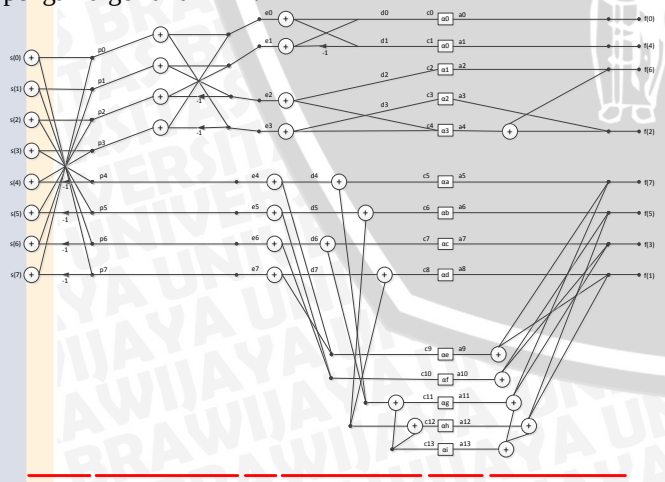
III. Metodologi

1. Studi Literatur
2. Perancangan dan Implementasi Sistem
3. Pengujian dan Pembahasan
4. Pengambilan Kesimpulan dan Saran

IV. PERANCANGAN DAN IMPLEMENTASI.

4.1 Perancangan Secara Umum

Perancangan sistem diawali dengan membagi 1D-DCT *flowgraph* algoritma LLM menjadi beberapa *step*/langkah dan melakukan penyekalaan terhadap koefisien pengali algoritma LLM.



Gambar 4.1. 1D-DCT *flowgraph* algoritma LLM [3].

Berdasarkan pada gambar 4.1 maka komputasi aritmatik dapat di bagi kedalam enam tahap (*step*) menjadi persamaan 4.1 sampai persamaan 4.13.

Tahap 1 :
s(0)=p0+p7; s(1)=p1+p6; s(2)=p2+p5; s(3)=p3+p4; (4.1)
s(4)=p3+(-p4); s(5)=p2+(-p5); s(6)=p1+(-p6); s(7)=p0+(-p7); (4.2)

Tahap 2 :
p0=e0+e3; p1=e1+e2; p2=e1+(-e2); (4.3)
p3=e0+(-e3); p4=e4; p5=e5; p6=e6; p7=e7; (4.4)

Tahap 3 :
e0=d0+d1; e1=d0+(-d1); e2=d2+c4; e3=d3+c4; (4.5)
e4=d4+c9; e5=d5+c10; e6=d6+c10; e7=d7+c9; (4.6)

Tahap 4 :
d0=c0; d1=c1; d2=c2; d3=c3; d4=c5+c11+c13; (4.7)
d5=c6+c12+c13; d6=c7+c11+c13; d7=c8+c12+c13; (4.8)

Tahap 5 :
c0=α0*a0; c1=α0*a1; c2=α1*a2; c3=α2*a3 (4.9)
c4=α3*a4; c5=αa*f(7); c6=αb*f(5); c7=αc*f(3); (4.10)
c8=αd*f(1); c9=αe*a5; c10=αf*a6; c11=αg*a7; (4.11)
c12=αh*a8; c13=αi*a9; (4.12)

Tahap 6 :
a0=f(0); a1=f(4); a2=f(6); a3=f(2); a4=f(6)+f(2); (4.13)
a5=f(7)+f(1);a6=f(5)+f(3);a7=f(7)+f(3);a8=f(1)+f(5);a9=a7+a8; (4.14)

Proses penyekalaan terhadap semua koefisien pengali algortima LLM dibutuhkan karena hasil dari proses 1D-DCT dua kali lipat dari hasil sebenarnya, maka untuk mendapatkan hasil keluaran tanpa melakukan proses pembagian pada keluaran adalah dengan cara melakukan penyekalaan pada seluruh koefisien pengali dengan faktor $\frac{1}{2}$, sehingga koefisien pengali hasil penyekalaan dapat dilihat pada tabel 4.1.

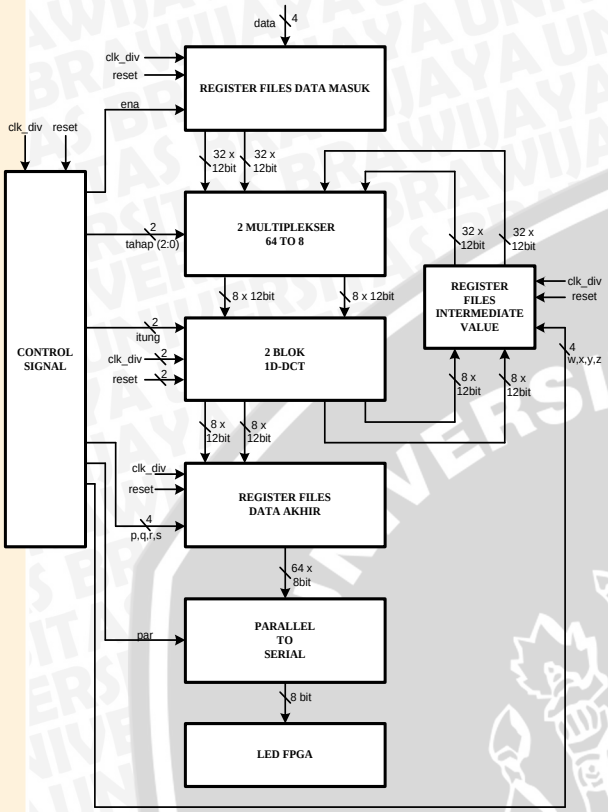
Tabel 4.1 Koefisien pengali algoritma LLM hasil penyekalaan

Konstanta	Amplitude	Hasil penyekalaan
a0	0,70711	0,3535
a1	-1,30656	-0,6532
a2	0,5412	0,2706
a3	0,38268	0,1913
aa	0,21116	0,1055
ab	1,45177	0,7258
ac	2,17273	1,0863
ad	1,06159	0,5308
ae	-0,63638	-0,3181
af	-1,81225	-0,9061
ag	-1,38704	-0,6935
ah	-0,2759	-0,1379
ai	0,83147	0,4157

4.2 Perancangan Implementasi 2D-DCT

Proses 2D-DCT dapat dilakukan dengan melakukan proses 1D-DCT pada bagian baris dari matriks 8x8 data masukan (64 data) kemudian menyimpan hasilnya pada *register*, kemudian dilakukan proses 1D-DCT pada bagian

kolom. Proses 2D-DCT dirancang menggunakan proses *sequential*, yang berarti proses dilakukan secara bertahap pada tiap modul atau unit yang membentuk sistem 2D-DCT.



Gambar 4.2 Diagram alir implementasi proses 2D-DCT

Sumber data dikirimkan lewat mikrokontroler setiap 4 bit, pengiriman data tersebut akan dikendalikan oleh *clock* yang dihasilkan FPGA sebagai pemicu fungsi *interrupt* yang tersedia pada mikrokontroler. Mikrokontroler menggunakan fungsi *interupt0* untuk mendeteksi *rising edge* dari *clock*, sehingga ketika terdapat perubahan logika dari '0' menjadi '1' maka *interupt0* akan aktif dan kemudian 4 bit data dikirimkan.

Prosesor 2D-DCT tersusun atas unit sumber data 4 bit dari mikrokontroler, *control signal*, *register files*, multiplekser dua unit 1D-DCT, unit *parallel to serial* dan unit penghasil detak (*clock generator*) seperti yang terlihat pada gambar 4.2. Data dikirim oleh mikrokontroler sebesar 4 bit sekali pengiriman yang kemudian ditampung dalam *register files* data masuk dan dibentuk mejadi 64 x 12bit data. Data akan dikelurakan dari *register files* data masuk dan selanjutnya akan dipilih oleh multiplekser data mana yang akan diproses oleh blok DCT pertama. Setelah proses komputasi DCT pertama selesai, data dikeluarkan dan ditampung oleh *register files intermediate value* yang selanjutnya akan dipilih kembali oleh multiplekser untuk diolah pada blok DCT kedua. Hasil komputasi kedua blok DCT akan ditampung oleh *register files* data akhir yang selanjutnya akan dikirim pada blok *parallel to serial* yang akan mengeluarkan data secara serial yang ditampilkan pada LED FPGA berupa data biner 8bit sesuai urutan indeks matriks.

Tabel 4.2 I/O *Port* sistem 2D-DCT

Nama I/O	I/O	Fungsi	Alamat Pin
clk	I	Sinyal *clock* FPGA	B8
rst	I	Untuk me-reset sistem	B18
data	I	Data masukan 4 bit dari mikro	L15,K12,L17,M15
clk_div	O	Sinyal *clock* sistem dan berfungsi juga untuk mengaktifkan interupsi mikrokontroler	M13
serial	O	Data keluaran 8 bit hasil akhir 2D-DCT	J14,J15,K15,K14, E17,P15,F4,R4

Tabel 4.2 menjelaskan I/O *port* dari perancangan sistem 2D-DCT, "clk" sebagai *input* yang berfungsi sebagai *clock* FPGA terletak pada pin B8, "rst" sebagai *input* yang berfungsi untuk me-*reset* sistem diatur pada pin B18, "data" sebagai *input* yang berfungsi sebagai jalan masuk 4bit data dari mikrokontroler diatur pada pin L15,K12,L17,M15, "clk_div" sebagai *output* yang berfungsi untuk mengaktifkan interupsi mikrokontroler, "serial" sebagai *output* yang berfungsi sebagai jalan keluar data hasil komputasi diatur pada pin J14,J15,K15,K14,E17,P15,F4,R4.

V. PENGUJIAN

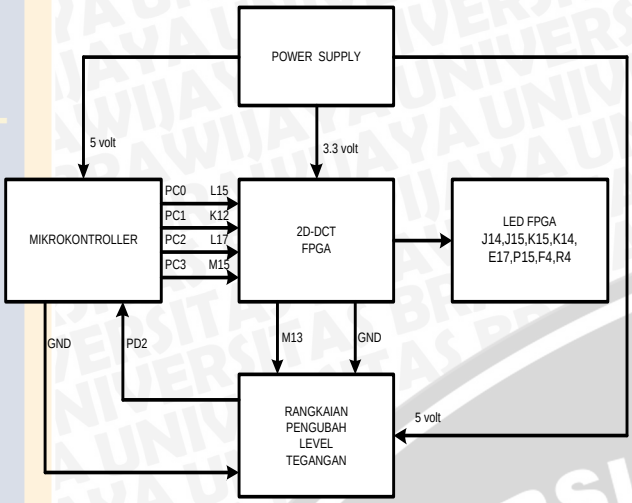
Pengujian akan membuktikan tingkat keberhasilan perancangan sistem serta akurasi dalam proses perhitungan. Pengujian dan pembahasan sistem meliputi beberapa aspek :

1. Akurasi proses perhitungan 2D-DCT meliputi akurasi perhitungan keseluruhan.
2. Unjuk kerja sistem pada FPGA Spartan 3E yaitu seberapa besar kapasitas yang digunakan dalam implementasi.

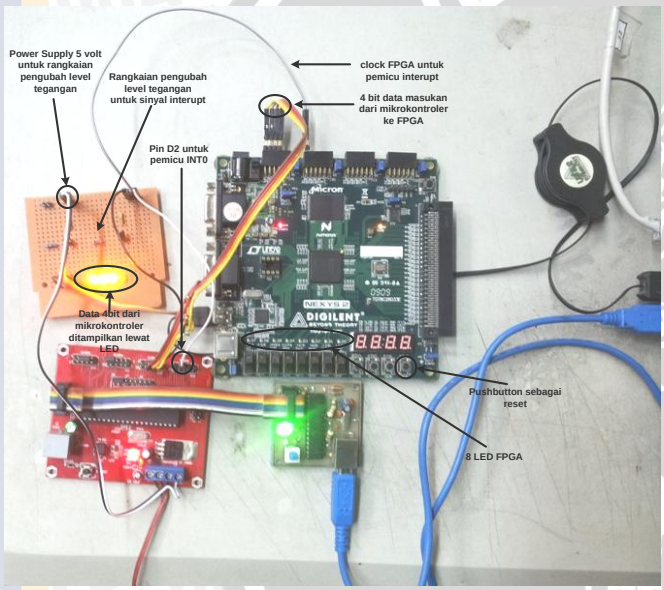
Akurasi algoritma LLM yang diterapkan pada sistem akan dibandingkan dengan hasil yang telah dihitung dengan program MATLAB. Unjuk kerja sistem akan diukur kapasitas sistem yang digunakan pada keeping FPGA (jumlah *slice* atau logika yang digunakan).

5.1 Pengujian Implementasi Sistem

Cara pengujian sistem 2D-DCT diperlihatkan pada gambar 5.1. Gambar 5.2 memperlihatkan sistem yang telah diimplementasikan ke dalam *hardware*.



Gambar 5.1 Cara pengujian implementasi 2D-DCT



Gambar 5.2 Cara pengujian implementasi 2D-DCT

5.2 Pengujian akurasi komputasi DCT

Dalam pengujian akurasi yang akan diamati adalah *error* mutlak, yaitu selisih kesalahan atau perbedaan antara nilai hasil komputasi FPGA dengan data masukan asli. *error* mutlak dirumuskan pada persamaan 5.1.

Error mutlak = |UE – UF| (5.1)
UE = hasil komputasi DCT dalam MATLAB
UF = hasil komputasi DCT dalam FPGA

Selain mengamati *error* mutlak, akan diamati *error* relatif yaitu persentase nilai *error* mutlak terhadap data piksel (data masukan asli) yang dapat dirumuskan pada persamaan 5.2.

$$\text{Error Relatif} = \frac{\text{error mutlak}}{\text{nilai data piksel}} \times 100\% \quad (5.2)$$

5.2.1 Pengujian Akurasi komputasi 2D-DCT

Rancangan sistem 2D-DCT dengan algortima LLM terdiri dari 64 masukan dan 64 keluaran. Masukan 2D-DCT dikirimkan oleh mikrokontroler per 4 bit sampai membentuk masukan 64x12 bit, sedangkan 64 keluaran ditampilkan secara berurutan dengan selang sekitar 1,342 detik melalui LED yang berupa data biner tak bertanda 8 bit.

Tabel 5.1 Percobaan Akurasi komputasi 2D-DCT algoritma LLM

Indeks Matriks	Data Piksel	Keluaran DCT		*Error* FPGA	
		MATLAB	FPGA(8bit)	Mutlak	Relatif (%)
0,0	110	935	931	4	3.636
0,1	110	-63	-60	3	2.727
0,2	118	18	16	2	1.695
0,3	118	-7	-7	0	0.000
0,4	121	7	7	0	0.000
0,5	126	13	12	1	0.794
0,6	131	-7	-7	0	0.000
0,7	131	0	0	0	0.000
1,0	108	74	70	4	3.704
1,1	111	-3	-3	0	0.000
1,2	125	-20	-18	2	1.600
1,3	122	-21	-18	3	2.459
1,4	120	-18	-18	0	0.000
1,5	125	-11	-10	1	0.800
1,6	134	8	8	0	0.000
1,7	135	5	4	1	0.741
2,0	106	-64	-62	2	1.887
2,1	119	3	3	0	0.000
2,2	129	5	2	3	2.326
2,3	127	15	16	1	0.787
2,4	125	10	8	2	1.600
2,5	127	9	6	3	2.362
2,6	138	1	1	0	0.000
2,7	144	-1	0	1	0.694
3,0	110	4	4	0	0.000
3,1	126	3	3	0	0.000
3,2	130	7	6	1	0.769
3,3	133	9	7	2	1.504
3,4	133	-3	-3	0	0.000
3,5	131	2	2	0	0.000
3,6	141	1	1	0	0.000
3,7	148	-1	2	1	0.676
4,0	115	3	3	0	0.000
4,1	116	1	1	0	0.000
4,2	119	-4	4	0	0.000
4,3	120	1	1	0	0.000
4,4	122	3	3	0	0.000
4,5	125	-3	2	1	0.800
4,6	137	0	0	0	0.000
4,7	139	1	1	0	0.000
5,0	115	8	8	0	0.000
5,1	106	-2	0	2	1.887
5,2	99	-1	1	0	0.000
5,3	110	0	0	0	0.000

Indeks Matriks	Data Piksel	Keluaran DCT		*Error* FPGA	
		MATLAB	FPGA(8bit)	Mutlak	Relatif (%)
5,4	107	0	0	0	0.000
5,5	116	-3	2	1	0.862
5,6	130	-3	3	0	0.000
5,7	127	-2	-2	0	0.000
6,0	110	-3	-3	0	0.000
6,1	91	0	0	0	0.000
6,2	82	1	0	1	1.220
6,3	101	0	0	0	0.000
6,4	99	0	0	0	0.000
6,5	104	0	0	0	0.000
6,6	120	-1	0	1	0.833
6,7	118	-1	0	1	0.847
7,0	103	-2	-1	1	0.971
7,1	76	-1	0	1	1.316
7,2	70	2	0	2	2.857
7,3	95	0	0	0	0.000
7,4	92	0	0	0	0.000
7,5	91	0	0	0	0.000
7,6	107	1	0	1	0.935
7,7	106	2	0	2	1.887
Rata - rata				0,797	0,706

Tabel 5.1 menginformasikan hasil komputasi 2D-DCT algoritma LLM setelah diimplementasikan pada FPGA dan kemudian hasilnya dibandingkan terhadap komputasi menggunakan *tool box* 2D-DCT MATLAB. Tabel 5.1 juga menginformasikan bahwa dalam percobaan tersebut percobaan memberikan nilai rata-rata *error* mutlak sebesar 0,797 dan nilai rata-rata *error* relatif sebesar 0,706 % untuk kisaran data masukan 70 sampai dengan 148 dari data pembanding.

5.3 Unjuk Kerja Sistem

Unjuk kerja implementasi sistem 2D-DCT dapat dilihat pada *report* implementasi. Kapasitas (*area*) merupakan aspek yang akan dianalisa pada sistem.

5.3.1 Unjuk kerja Implementasi 2D-DCT

Implementasi sistem 2D-DCT algoritma LLM pada FPGA Spartan 3E XC3S500E menghasilkan *report* implementasi yang terlihat pada tabel 5.2

Tabel 5.2 Penggunaan logika pada implementasi 2D-DCT

Jenis Pemakaian Logika	Kapasitas yang digunakan	Kapasitas yang tersedia	Persentasi Pemakaian
Jumlah *Slices*	2525	4656	54 %
Jumlah LUT	3532	9312	37%
Jumlah IOB	15	232	6%
Jumlah MULT18x18	14	20	70%
Jumlah GCLKs	2	24	8%

Penggunaan kapasitas logika dapat diminimalkan dengan meminimalkan perancangan sistem serta tata cara pemrograman VHDL. Dari tabel 5.2 dapat disimpulkan bahwa sistem 2D-DCT dapat diimplementasikan pada FPGA karena pemakaian kapasitas logikanya tidak melebihi kapasitas logika yang tersedia pada FPGA.

5.4 *Timing Summary*

Xilinx menyediakan *tools* untuk menampilkan *timing summary* dari perancangan sistem. *Timing summary* akan diperoleh pada saat proses sintesis, hasil perhitungan *software* Xilinx. Hasil *timing summary* merupakan hasil perhitungan secara simulasi sehingga tidak dapat menggambarkan *latency time*. Perancangan sistem 2D-DCT pada gambar 5.2 mengahsilkan *timing summary* sebagai berikut :

```
Timing Summary:
---------------
Speed Grade: -4

   Minimum period: 29.362ns (Maximum Frequency: 34.058MHz)
   Minimum input arrival time before clock: 4.679ns
   Maximum output required time after clock: 5.518ns
   Maximum combinational path delay: No path found

Timing Detail:
--------------
All values displayed in nanoseconds (ns)
```

VI. PENUTUP

6.1 Kesimpulan

Berdasarkan pengujian dan pembahasan implementasi sistem maka dapat diambil beberapa kesimpulan sebagai berikut :

1. Nilai kesalahan (*error*) hasil perhitungan 2D-DCT setelah diimplementasikan pada FPGA menggunakan algoritma LLM terjadi karena proses komputasi hanya melibatkan bilangan bulat baik dalam proses perkalian maupun penjumlahan serta pengurangan.
2. Implementasi 2D-DCT menggunakan 2525 *slices* atau 54 % dari kapasitas total *slices*, 3532 LUT atau 37 % dari kapasitas total LUT, 15 IOB atau 6 % dari kapasitas total IOB, 14 *embedded multiplier* atau 70 % dari kapasitas total *embedded multiplier* dan 2 GCLKs atau 8 % dari kapasitas total GCLKs FPGA XC3S500E FG320.
3. Total waktu tunda setelah 8 data 12 bit terkirim sampai data valid keluar dari untai 1D-DCT pada implementasi sebesar 4 siklus *clock*, jika satu siklus memiliki periode 1µs, maka total waktu tunda yang terjadi sebesar 4 µs.

6.2 Saran

Implementasi DCT pada FPGA ini membuka kemungkinan terhadap pengembangan lebih lanjut antara lain:

1. Selain menggunakan mikrokontroler sebagai sumber data masukan, modul RS-232 yang tersedia pada FPGA XC3S500E dapat dimanfaatkan sebagai sumber data masukan serial melalui PC.
2. Pada implementasi 2D-DCT unjuk kerja sistem dapat diperbaiki dengan memanfaatkan *block RAM* yang tersedia pada FPGA XC3S500E sebagai media penyimpanan data komputasi, karena penggunaan *register files* akan lebih banyak membutuhkan *slices flip-flop.*

DAFTAR PUSTAKA

[1] Atitallah A. B., Kadionik P., Ghozzi F., Nouel P., Masmoudi N., Marchegay P. "Optimization and implementation on FPGA of the DCT/IDCT algorithm" // IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '06), P. 928–931, 2006.

[2] Basri, I.Y. "Implementasi DCT pada Field Programmable Gate Array", Thesis S2 Jurusan Teknik Elektro. UGM : Yogyakarta, 2010.

[3] C. Loeffler and A. Lightenberg, "Practical fast 1-D DCT algorithms with 11 Multiplications," Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP '89), Scotland, pp. 988-991, May 1989.

[4] Narasimha M. J., Peterson A. M." On the computation of the discrete cosine transform" // IEEE Trans. Commun.,– No. 26(6). – P. 934–936, 1978.

[5] W.c Chen, C.h Smith and S.C. Fralick, "A fast Computational Algorithm for thr Discrete Cosine Transform,"IEEE Trans. On Communications, Vol. COM-25, No. 9, pp.1004-1009, Sept.1997.

[6] Xilinx (2013). *Spartan 3E FPGA Family Data Sheet.*From:http://www.xilinx.com/support/documentation/data_sheets/ds312.pdf.