PENGEMBANGAN PERANGKAT PENGHITUNG JITTER BERBASIS FPGA

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar Sarjana Sains dalam bidang Fisika

Oleh:

PRICYLIA VALENTINA

145090801111009

JURUSAN FISIKA
FAKULTAS MATEMTIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG
2019

LEMBAR PENGESAHAN SKRIPSI

PENGEMBANGAN PERANGKAT PENGHITUNG JITTER BERBASIS FPGA

Oleh:
PRICYLIA VALENTINA
145090801111009

Setelah dipertahankan didepan Majelis Penguji
Pada tanggal..... 06 MAR 2019
Dan dinyatakan memenuhi syarat untuk memperoleh gelar Sarjana Sains dalam bidang Fisika

Menyetujui,

Pembimbing I	Pembimbing II
Drs. Hari Arief Dharmawan, M.Eng., Ph.D.	**Dr.Eng. Agus Naba, S.Si.,MT**
NIP. 196909201994121001	**NIP. 197208061995121001**

Mengetahui,
Ketua Jurusan Fisika FMIPA UB

Prof.Dr. rer. nat. Muhammad Nurhuda
NIP. 196409101990021001

(Halaman ini sengaja dikosongkan)

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama : Pricylia Valentina
Nim : 145090801111009
Jurusan : Fisika
Penulis Tugas Akhir Berjudul :

PENGEMBANGAN PERANGKAT PENGHITUNG JITTER BERBASIS FPGA

Dengan ini menyatakan bahwa:

1. Tugas Akhir ini adalah benar-benar karya saya sendiri dan bukan hasil plagiat dari karya orang lain. Karya-karya yang tercantum dalam daftar pustaka Tugas Akhir ini semata-mata digunakan sebagai referensi.
2. Apabila kemudian hari diketahui bahwa isi Tugas Akhir saya merupakan hasil plagiat, maka saya bersedia menanggung akibat hukuman dari keadaan tersebut.

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, 22 Februari 2019
Yang menyatakan

Pricylia Valentina
145090801111009

(Halaman ini sengaja dikosongkan)

PENGEMBANGAN PERANGKAT PENGHITUNG JITTER BERBASIS FPGA

ABSTRAK

Pada penelitian ini dibahas pengembangan perangkat berbasis FPGA untuk penghitungan jitter dari sinyal digital yang memiliki level tegangan 0 volt hingga 5 volt. Metode yang dilakukan adalah mencacah *duty cycle* (*time high* dan *time low*) dari sinyal digital berfrekuensi 1 Khz hingga 10 Khz, 20 Khz hingga 100 Khz (dengan kelipatan 10) dan 100 Khz hingga 500 Khz (dengan kelipatan 100) untuk sinyal masukan yang bersumber dari Arduino uno. 1 Khz hingga 10 Khz, 20 Khz hingga 100 Khz (dengan kelipatan 10), 100 Khz hingga 1000 Khz (dengan kelipatan 100), dan 1 Mhz hingga 15 Mhz untuk sinyal digital masukan yang bersumber dari sinyal generator menggunakan sistem counter FPGA DE0-NANO dengan frekuensi clock 50 Mhz. Hasil cacahan counter FPGA dibandingkan dengan nilai cacahan pembandingnya untuk mengetahui tingkat kelayakan counter FPGA dalam melakukan cacahan sinyal. Nilai cacahan pembanding diperoleh dengan cara membagi periode *time high* atau *time low* dari real frekuensi sinyal yang dapat diketahui menggunakan oscilloscope (periode *time high* atau *time low* merupakan periode sinyal dibagi 2 hal ini karena *duty cycle* terdiri dari 50% *time high* dan 50% *time low* sehingga besar periode *time high* dan *time low* adalah sama) dengan periode *clock* counter FPGA. Hasil dari penelitian menunjukan bahwa nilai cacahan yang dihasilkan oleh FPGA memiliki hasil yang mendekati nilai cacahan pembandingnya. Metode ini juga terbukti dapat digunakan untuk mencacah sinyal digital hingga frekuensi 13 Mhz sehingga dapat dikatakan bahwa pengembangan counter FPGA layak untuk digunakan dalam mencacah sinyal digital untuk melakukan analisa nilai jitter.

Kata Kunci : *jitter, time high, time low, counter FPGA*

(Halaman ini sengaja dikosongkan)

THE DEVELOPMENT OF JITTER COUNTING DEVICE BASED ON FPGA

ABSTRACT

In this research, there will be explanations about the development of FPGA-based device for counting jitter of digital signal whose 0 volt until 5 volt voltage level. The method is counting *duty cycle* (*time high* and *time low*) from digital signal with the frequency of 1 kHz until 10 kHz, 20 kHz until 100 kHz (in multiples of 10), and 100 kHz until 500 kHz (in multiples of 100) for input signal that is sourced from Arduino uno; 1 kHz until 10 kHz, 20 kHz until 100 kHz (in multiples of 10), 100 kHz until 1000 kHz (in multiples of 100), and 1 MHz until 15 MHz for digital input signal which is sourced from signal generator that uses FPGA DE0-NANO counter system with the clock frequency of 50 MHz. The counting number of FPGA counter then compared with its comparative counting number to knowing the feasibility of FPGA counter in course of signal counting. The comparative counting number is obtained by dividing the period of *time high* or *time low* of real signal frequency which is able to known by using oscilloscope (the period of *time high* or *time low* is signal period that is divided by two because of *duty cycle* which is consisted of 50% *time high* and 50% *time low*, so that the period of *time high* and *time low* is the same) with the period of FPGA *clock* counter. The result of this research shows that the counting number which is obtained from FPGA approaches the comparative counting number. This method is also proven to counting the digital signal until the frequency of 13 MHz, so it is able to state that the development of FPGA counter is worth to used for counting digital signal for analyzing jitter value.

Keyword : *jitter, time high, time low, counter FPGA*

(Halaman ini sengaja dikosongkan)

KATA PENGANTAR

Puji syukur kehadirat Allah SWT, atas limpahan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi sesuai dengan waktu yang telah direncanakan. Shalawat dan salam kami haturkan kepada Nabi Muhammad SAW, beserta keluarga dan para sahabatnya. Penulisan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk memperoleh gelar Sarjana dalam bidang Sains Jurusan Fisika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Brawijaya. Penulis mengucapkan terimakasih kepada pihak yang telah mendukung dan membantu penelitian ini khususnya kepada:

1. Kedua Orang tua, Ayah dan Ibu, yang telah memberikan bantuan materil dan dukungan yang besar demi kelancaran pengerjaan penelitian ini
2. Bapak Drs. Hari Arief Dharmawan, M.Eng., Ph.D. selaku pembimbing pertama dan pembimbing akademik dari semester satu hingga saat ini, yang telah meluangkan waktu dan pikiran, serta arahan selama pembuatan alat dan penulisan skripsi ini.
3. Bapak Dr.Eng. Agus Naba, S.Si.,MT selaku pembimbing kedua karena telah memberikan arahan dalam penyusunan skripsi ini.
4. Seluruh Dosen, Staf dan Karyawan jurusan Fisika yang telah memberikan pendidikan dan bantuan selama di jurusan Fisika FMIPA UB.

Penulis menyadari bahwa penulisan ini masih terdapat kekurangan baik dalam penyusunan, bahasa dan penyajian penjelasannya. Oleh karena itu penulis mengharapkan saran dan kritik dari pembaca sehingga dapat memberikan perubahan ke arah yang lebih baik. Semoga penelitian ini dapat bermanfaat bagi para pembaca.

Malang, 22 Februari 2018

Penulis

(Halaman ini sengaja dikosongkan)

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Lebar pulsa pada *timing* digital menurut National Instruments (2016) idealnya memiliki lebar yang sama dalam proses komunikasi data. Namun, pada kenyataannya beberapa sinyal kotak (digital) memiliki lebar pulsa yang berbeda satu sama lain, perbedaan lebar pulsa inilah yang disebut sebagai jitter. Menurut Wulandari (2016) dampak jitter menjadi buruk atau mempengaruhi sistem ketika nilai jitter pada sinyal digital semakin besar. Besarnya jitter mengakibatkan terjadinya *delay* pada proses komunikasi data dan menyebabkan kualitas proses komunikasi data menjadi buruk. Pentingnya mengetahui jitter yang ada pada sinyal digital untuk memperkecil buruknya kualitas dari proses komunikasi data. Analisa jitter sendiri dapat dilakukan dengan metode *sampling counter* dimana lebar *time high* dan *time low* pulsa sinyal digital yang diukur, disampling dengan sinyal *timing* (*clock*) berfrekuensi lebih tinggi.

Dengan demikian dibutuhkan alat yang memiliki *clock* frekuensi tinggi agar diperoleh hasil cacahan dari lebar pulsa sinyal yang diukur. Salah satu perkembangan teknologi yang ada saat ini adalah FPGA (*Field Programmable Gate Array*), menurut Prasetio., dkk (2017) merupakan IC yang dapat diprogram sesuai kebutuhan pengguna dan dapat mengimplementasikan suatu rangkaian digital. FPGA memiliki *internal clock* atau sistem timing sebesar 50MHz. Dengan besarnya timing yang dimiliki oleh tersebut menjadikan FPGA sangat cocok untuk metode *sampling counter* dalam analisa perhitungan jitter yang terjadi pada sinyal kotak (digital) pada proses komunikasi data. Sehingga maksud dari penelitian ini adalah perancangan program counter pada FPGA untuk mengetahui nilai jitter yang terjadi dalam sinyal digital.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan, rumusan masalah dari penelitian ini adalah bagaimana perancangan program counter pada FPGA sehingga dapat mencacah *time high* dan *time low* sinyal digital pada proses komunikasi data agar diperoleh nilai jitter yang terjadi pada sinyal digital.

1.3 Batasan Masalah

Batasan masalah dari penelitian ini adalah

1. Sinyal yang digunakan merupakan sinyal kotak digital dengan level tegangan 0 volt hingga 5 volt.
2. Sinyal digital masukan yang digunakan pada penelitian kali ini berasal dari Arduino uno dan sinyal generator.
3. Frekuensi sinyal digital yang digunakan untuk pengambilan data adalah 1Khz hingga 10Khz, 20Khz hingga 100Khz (dengan kelipatan 10), dan 100Khz hingga 500Khz (dengan kelipatan 100) untuk sinyal input yang berasal dari Arduino Uno.
4. Frekuensi sinyal digital yang digunakan dari input sinyal generator adalah 1Khz hingga 10Khz, 20Khz hingga 100Khz (dengan kelipatan 10), 100Khz hingga 1000Khz (dengan kelipatan 100), dan 1Mhz hingga 15Mhz.
5. FPGA yang digunakan merupakan FPGA jenis Altera DE0-Nano cyclone IV EP4CE22F17C6N.
6. Menggunakan software Quartusprime Lite Edition untuk merancang program FPGA.
7. Menggunakan bahasa VHDL.

1.4 Tujuan

Tujuan dari penelitian ini adalah mengembangkan sistem counter pada FPGA sebagai pencacah *time high* dan *time low* sinyal digital untuk memeperoleh nilai jitter pada sinyal digital.

1.5 Manfaat

Manfaat dari penelitian ini adalah dapat mengetahui hasil pengukuran jitter dari sinyal digital, dimana nilai jitter tersebut salah satunya dapat digunakan untuk analisa kualitas proses komunikasi data.

(Halaman ini sengaja dikosongkan)

BAB II
TINJAUAN PUSTAKA

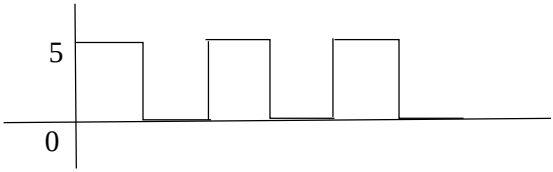
2.1 Sinyal

Sinyal adalah bentuk data yang mempunyai informasi dalam besaran fisis sesuai dengan perubahan dalam ruang, waktu, atau perubah-perubah bebas lainnya. Contohnya adalah sinyal suara, *electrocardiogram*, dan *electroencephalogram*. Dalam bentuk matematika, sinyal merupakan fungsi perubahan bebas dengan mempunyai satu perubah atau perubah lebih dari satu (Mustofa, 2018).

Saat ini pengolahan sinyal banyak dilakukan secara digital. Hal ini karena kelebihan-kelebihan yang dimilikinya, antara lain:

1. Untuk menyimpan hasil pengolahan, sinyal digital lebih mudah dibandingkan penyimpanan sinyal analog. Untuk media penyimpanan dapat digunakan elemen memori: *Flash memory*, CD/DVD *hard disk*. Untuk penyimpanan sinyal analog dapat digunakan pita tape *Inagnetik*.
2. Sinyal digital lebih kebal terhadap *noise*, karena bekerja dengan level tegangan logika "1" dan "0"
3. Lebih kebal terhadap perubahan temperatur.
4. Lebih mudah memprosesnya, secara teori tidak ada batasannya, tergantung dari kreativitas dan inovasi perancang (Tanudjaja, 2007).

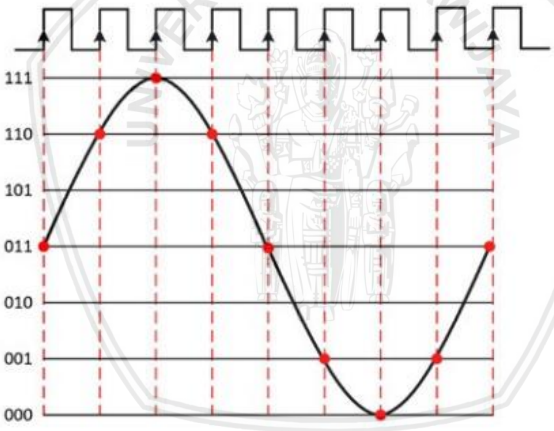
Sinyal digital dalam bentuk gelombang kotak atau *square waveform* umum digunakan pada rangkaian *mikro elektronik* untuk pengendaian waktu atau *timing control*. Hal tersebut terjadi karena memiliki bentuk gelombang yang simestris dengan durasi yang sama pada siklus setengah kotak dengan setengah kotak yang lainnya atau interval yang teratur. Bentuk gelombang kotak dapat diklasifikasikan dalam sinyal digital ketika level tegangan yang dimiliki oleh gelombang adalah 0 volt hingga 5 volt.



Gambar 2.1 Sinyal Digital Gelombang Kotak

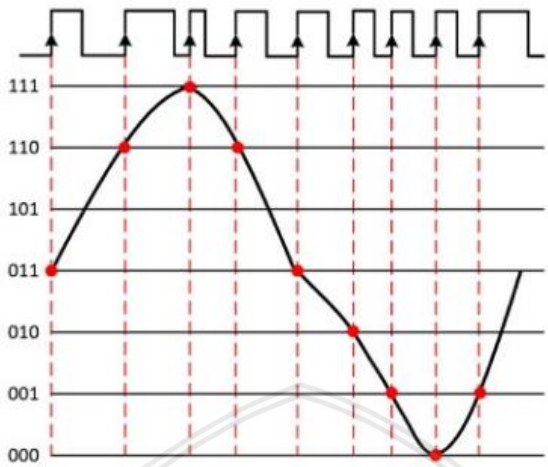
2.2 Jitter

Jitter adalah penyimpangan dari *Timing ideal* suatu *event* terhadap *Timing* yang sebenarnya dari suatu *event*. Bayangkan ketika mengirim gelombang sinus dengan bentuk data digital dan mencetaknya di kertas grafik. Setiap kotak sesuai dengan lebar pulsa *clock* karena garis-garis vertikal berjarak sama jauhnya. Karena sifat periodik dari *clock* yang sangan presisi maka hasil plot akan menghasilkan gelombang sinus yang bagus.



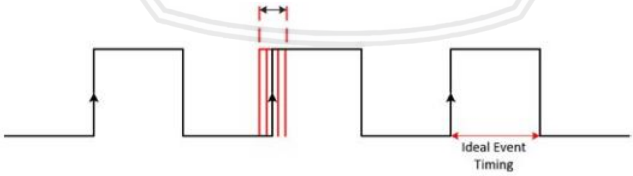
Gambar 2.2 Contoh *Clock* Presisi

Bayangkan bahwa garis-garis tersebut tidak sama jaraknya karena sinyal *clock* yang tidak periodik atau tidak presisi. Maka saat mengirimkan data tidak pada interval yang sama menyebabkan data yang terkirim akan terdistorsi.



Gambar 2.3 *Jitter* pada Sinyal

Pada Gambar 2.3, terlihat bahwa jarak antara transisi naik pada sinyal *clock* tidak merata (jarak panah hitam keatas), hal inilah yang disebut jitter di dalam *clock*. Pada gambar diatas sinyal *clock timing* memiliki banyak *jitter* sehingga menyebabkan interval yang tidak sama. Perbedaan ini menyebabkan distorsi kedalam bentuk gelombang yang direkam, ditransmisikan dan direproduksi ulang pada ploter. Dengan melihat satu denyut pulsa *clock* tiap satu *duty cycle* maka jitter adalah penyimpangan-penyimpangan pada tepi pulsa *clock* terhadap *Timing* Sinyal Idealnya sehingga menyebabkan lebar pulsa *clock* yang tidak sama.



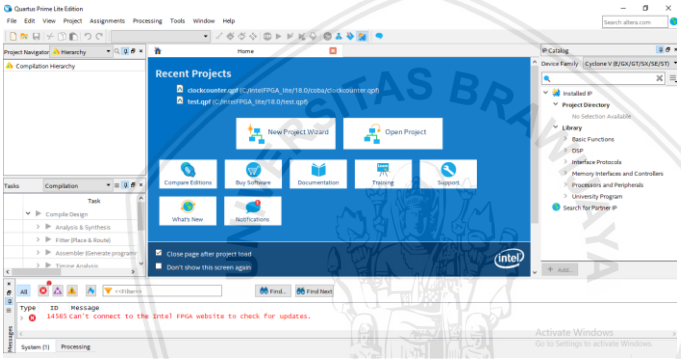
Gambar 2.4 Jitter pada Pulsa Tunggal

Jitter biasanya diukur pada *zero-crossing* dari sinyal referensi. Berasal dari *cross-talk*, *output switching* simultan, dan

sinyal interferensi lainnya yang terjadi secara teratur (National Instruments, 2016).

2.3 Quartus Prime Lite Edition

Quartus Prime Lite Edition merupakan salah satu jenis software dari quartus yang digunakan untuk membuat simulasi rangkaian logika secara digital pada perangkat FPGA dengan memanfaatkan bahasa deskripsi yaitu VHDL ataupun verilog. Quartus dibuat oleh perusahaan Altera. Melalui Software Quartus Prime Lite Edition hasil pengkodean dapat dilihat hasilnya secara fisik atau real melalui perangkat FPGA.



Gambar 2.5 Tampilan Awal Quartus Prime Lite Edition

2.4 VHDL

VHDL merupakan kepanjangan dari VHSIC *Hardware Description Language*, sedangkan VHSIC sendiri kepanjangan dari *Very High Speed Integrated Circuit*. VHDL, merupakan bahasa deskripsi perangkat keras formal untuk menentukan watak dan struktur rangkaian digital. Kode VHDL mempunyai kemiripan dengan bahasa pemrograman tingkat tinggi seperti C, C++, Pascal, dan sebagainya. Bahasa deskripsi perangkat keras Iain yang juga sering digunakan untuk konfigurasi perangkat keras adalah verilog. Kode VHDL muncul pada tahun 1980-an karena kebutuhan para perekayasa perangkat keras dalam bahasa standar untuk mendeskripsikan struktur dan fungsi *integrated*

circuit (IC), kemudian kode VHDL diterima sebagai salah satu bahasa standar yang paling penting untuk penentuan (*specifying*), pengujian (*verifying*), dan perancangan (*designing*) dalam dunia elektronika digital oleh *Institute of Electrical and Electronic Engineers* (IEEE) (Wibowo, 2015).

Sebagai sebuah bahasa pemrograman, VHDL memiliki metode dan syarat-syarat penulisan yang harus dipatuhi. Berikut ini adalah konvensi dasar dari VHDL:

1. VHDL adalah *case insensitive* artinya penulisan huruf besar dan kecil memiliki makna yang sama.
2. Penamaan dan pelabelan :
   a. Semua nama harus dimulai dengan huruf.
   b. Harus terdiri hanya karakter *alfanumerik* dan garis bawah.
   c. Tidak boleh terdiri dari dua garis bawah.
   d. Semua nama dan label dari *entity* harus unik.
3. Format bahasa bebas (memperbolehkan spasi)
4. Komen dimulai dengan “ ” (Prasetio dkk., 2017).

Deskripsi dari VHDL sendiri terdiri dari 3 struktur dasar, yang pertama adalah *library*. *Library* berisi semua operasi logika yang digunakan pada desain VHDL. Untuk membuat desain suatu VHDL dibutuhkan pendeklarasian *library* terlebih dahulu. Program VHDL memiliki *syntax* sebagi berikut:

library logical_names; //Contohnya, library freds_lib;

Ada 4 standar dasar *libray* yaitu:

1. library ieee; // membaca paket standart dari *library* ieee.
2. use ieee.ieee.std_logic_1164.all; //memasukkan semua bagian dari ieee std_logic *variable*.
3. use ieee.std_logic_arith.all; //memasukkan semua operasi aritmatika untuk standar *logic variable*.
4. use ieee.std_logic_unsigned.all; // memaksukan semua fungsi yang belum didesain untuk operasi aritmatika (Mazor dan Patricia, 1992).

Struktur yang kedua adalah *entity*, menurut Perry (2002) *entity* merupakan bagian utama dari desain VHDL yang mendeklarasikan *input* dan *output* dari desain program. *Entity* menjelaskan tentang *input output* dari program VHDL yang di buat dari penentuan port hingga mode dan tipe data.

Struktur ketiga adalah *architecture* yang merupakan deklarasi dari isi atau sistem utama yang bekerja pada program. Pada *architecture* akan di jelaskan fungsi kerja masing-masing jalur. Terdapat lebih dari satu *architecture* dalam satu *entity*. Penamaan dari *architecture* sendiri harus sama dengan penamaan yang ada pada *entity* dengan memberikan nama secara bebas didepan penamaan yang sama dengan *entity*. Untuk lebih jelasnya mengenai *library*, *entity*, dan *architecture* dapat di lihat pada contoh dari program VHDL dalam sistem *counter* di bawah ini.

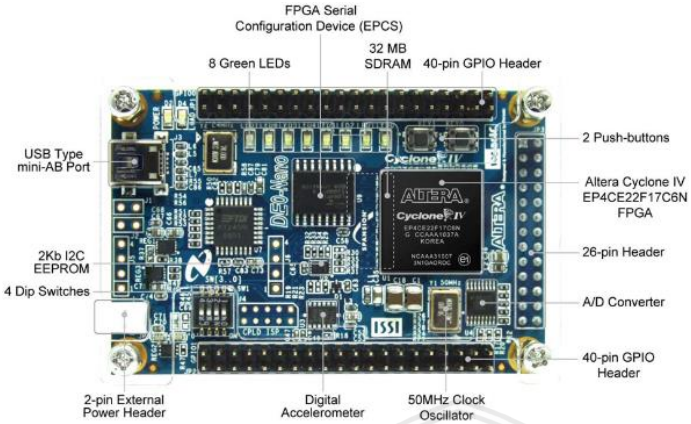
```
library ieee;
use ieee.ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity up_counter is
port (clk : in std_logic;
rst : in std_logic ;
Q : out std_logic_vector (3 downto 0));
end up_counter;

architecture Behavioral of up_counter is
signal tmp:std_logic_vector (3 downto 0);
begin
process (clk,rst)
begin
if (rst='1') then
tmp<="0000";
elseif(clk'event and cl='1') then
tmp<=+1;
end if;
end process;
Q <= tmp;
```

```
end Behaviorial;
```

Deklarasi pada bagian *library* bertujuan untuk kebutuhan *logic*. Deklarasi *entity* berupa deklarasi *input* (clk,rst) dan *output* (Q) yang berupa *vector* sepanjang 4 bit. Deklarasi *architecture* berisi kode utama. Saat *clock* tepi naik maka nilai *output* adalah nilai *output* sebelumnya ditambah 1. Jika rst bernilai 1, maka nilai *output* direset menjadi 0000 (Prasetio dkk., 2017).

2.5 FPGA

FPGA merupakan sebuah *Integrated Circuit* (lC) yang digunakan untuk mengimplementasikan suatu rangkaian digital. FPGA memiliki kemampuan untuk diprogram ulang sesuai desain rangkaian digital yang dirancang oleh pengguna. Keuntungan dari penggunakan FPGA yaitu dikonfigurasi langsung oleh *end user* dan juga tersedianya solusi yang mendukung *chip customized Verylarge Scale Integration* (VLSI). Saat ini terdapat beberapa perusahaan yang memproduksi FPGA di antaranya adalah Altera, Lattice, quicklogic, Actel dan Xilinx. Pada dasarnya, FPGA merupakan unit pemroses yang dapat disematkan di dalam sebuah sistem *embedded*. Fungsi yang ditawarkan juga tak jauh berbeda dengan mikrokontroller sejenis arduino atau mikrokomputer seperti raspberry pi. Hanya saja, perbedaan utama yang menjadi ciri FPGA adalah bahwa setiap pemrograman yang didownload ke dalam chip FPGA akan mempengaruhi konfigurasi gerbang didalam chip. Hal ini menjadi kekuatan FPGA dibandingkan dengan mikrokontroller maupun mikrokomputer karena FPGA dapat dikonfigurasi untuk menjalankan fungsi secara *paralel* pada waktu yang bersamaan. Dengan demikian, umumnya FPGA digunakan untuk keperluan aplikatif dimana ada keperluan untuk menjalankan komputasi secara *paralel* dan cepat yang jika dilakukan oleh mikrokontroller biasa akan memerlukan proses yang relatif lebih lama (Prasetio dkk., 2017).



Gambar 2.6 FPGA Altera DE0-Nano

Dengan fitur FPGA Altera DE0-Nano adalah sebagai berikut:

1. 22.320 LEs
2. 594-Kb *embedded* memori
3. 66 *embedded* 18x18 *multipliers*
4. 4 *general-purpose phase-locked loops* (PLLs)
5. 153 maksimum FPGA I/O pin
6. Status konfigurasi dan elemen *set-up*
   a. *Onboard* USB-BlasterTM sirkuit untuk pemrograman
   b. Altera konfigurasi perangkat *seri* - EPCS16
7. *Expansion header*
   a. Dua *header* 40-pin (GPIOs) menyediakan 72 I / O pin
   b. Dua pin 5-V *power*, dua pin kekuatan 3,3-V, dan empat pin *ground*
   c. Satu *header* 26-pin menyediakan 16 I / O pin digital dan 8 pin *input* analog untuk terhubung ke sensor analog
8. perangkat memori
   a. 32-MB SDRAM
   b. 2-Kb I2C EEPROM
9. penggunaan umum *input*/*output*
   a. 8 LED hijau
   b. 2 tombol *push debounced*
   c. 4 *dual in-line* paket (DIP) *switch*

10. G-sensor
    3-*axis accelerometer* dengan resolusi tinggi (13 bit)
11. Analog ke digital *converter*
    a. 8-*channel*, 12-bit *analog-to-digital converter*
    b. 50 KSP 200 KSP
12. *Clock* sistem
    Onboard 50-MHz *clock* osilator
13. Sumber Daya listrik
    a. USB *Type Port Mini*-AB (5 V)
    b. Dua DC 5-V pin *header* GPIO (5 V)
    c. 2-pin *header* daya eksternal (3,6 V - 5,7 V)
14. konektivitas
    a. Terhubung D5M. Terhubung dengan sensor CMOS 5 *megapixel* (D5M)
    b. Terhubung LTM. Terhubung dengan modul LCD layar sentuh (LTM)
    c. terhubung THDB-ADA. Terhubung dengan *analog-to-digital / digital-to-analog* daughtercard konversi (THDB-ADA)
15. USB Mini-B Kabel (Intel, 2018).

(Halaman ini sengaja dikosongkan)

BAB III
METODE PENELITIAN

3.1 Waktu dan Tempat Penelitian

Penelitian ini dilakukan mulai September 2018 hingga Desember 2018 bertempat di laboratorium Instrumentasi dan Pengukuran Gedung Biomol lantai 3 Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Brawijaya Malang.

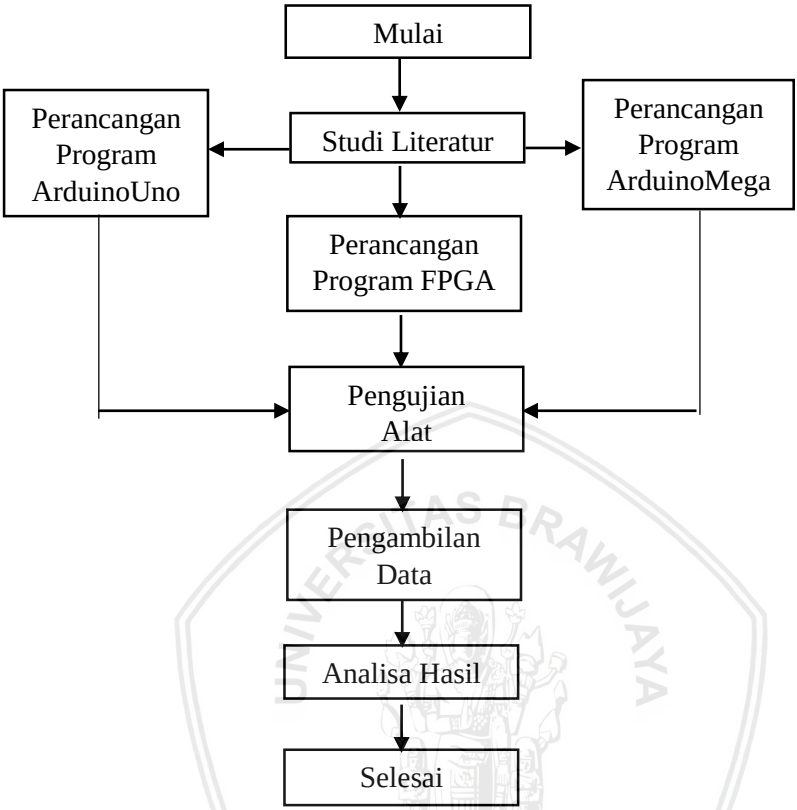
3.2 Peralatan dan Bahan

Pada penelitian kali ini alat dan bahan yang digunakan adalah:

1. FPGA Altera DE0-Nano
2. Quartus Prime Lite Edition
3. Sumber sinyal (Arduino uno dan sinyal generator)
4. Laptop
5. Arduino Mega 2560
6. LCD 2x16 I2C
7. Kabel Header
8. Modul *Push Button*
9. Oscilloscope

3.3 Prosedur Penelitian

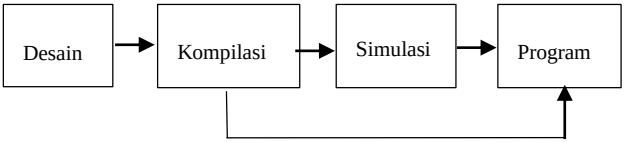
Penelitian kali ini memiliki lima tahapan utama, yaitu studi literatur, perancangan program, pengujian alat, pengambilan data, dan analisa hasil. Garis besar tahapan penelitian dapat digambarkan dengan diagram alir pada Gambar 3.1



Gambar 3.1 Diagram Alir Tahap Penelitian

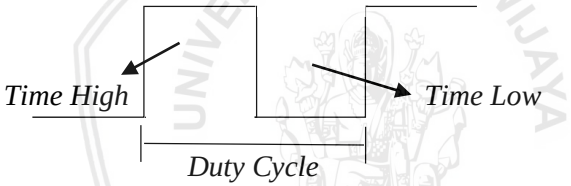
3.3.1 Perancangan Program FPGA

Setelah melakukan studi literatur mengenai materi penelitian yang dilakukan, langkah selanjutnya adalah perancangan program FPGA. Terdapat beberapa tahapan dalam perancangan program FPGA diantaranya desain, kompilasi, simulasi, dan program. Lebih jelasnya dapat dilihat dari alur prancangan pada Gambar 3.2



Gambar 3.2 Alur Perancangan FPGA

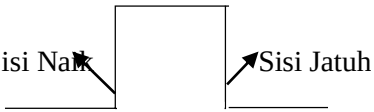
Sistem yang digunakan untuk mendesain FPGA sebagai penghitung lebar pulsa pada sinyal digital merupakan sistem counter. Pada sinyal digital lebar pulsa (*duty cycle*) terdiri dari *time high* dan *time low*. FPGA di rancang untuk mencacah seluruh *time high* dan *time low* yang ada dan dengan membandingkan hasil cacahan dari sebelum dan sesudahnya dapat diperoleh lebar pulsa terbesar dan terkecil dari *time high* dan *time low* sinyal digital. Dengan demikian dibutuhkan dua sistem counter pada FPGA untuk mecacah *time high* dan *time low* yang ada.



Gambar 3.3 *Clock Signal*

Logika program counter untuk mencacah *time high* adalah dimana counter mulai mencacah ketika FPGA mendeteksi adanya sisi naik pada *clock* sinyal input dan secara otomatis counter *high* berada pada keadaan *start* dan mulai mencacah. Counter berhenti mencacah atau berada pada posisi *stop* ketika FPGA mendeteksi adanya sisi jatuh pada *clock* sinyal input. Hasil cacahan counter pada *time high* disimpan dan dilakukan pengulangan cacahan untuk *time high* berikutnya. Sedangkan program counter FPGA untuk mencacah *time low* memiliki prinsip saat FPGA mendeteksi adanya sisi jatuh pada *clock* sinyal input maka counter *low* berada pada posisi *start* dan mulai mencacah. Saat FPGA mendeteksi sisi naik pada *clock* sinyal masukan maka counter *low* berhenti mencacah atau

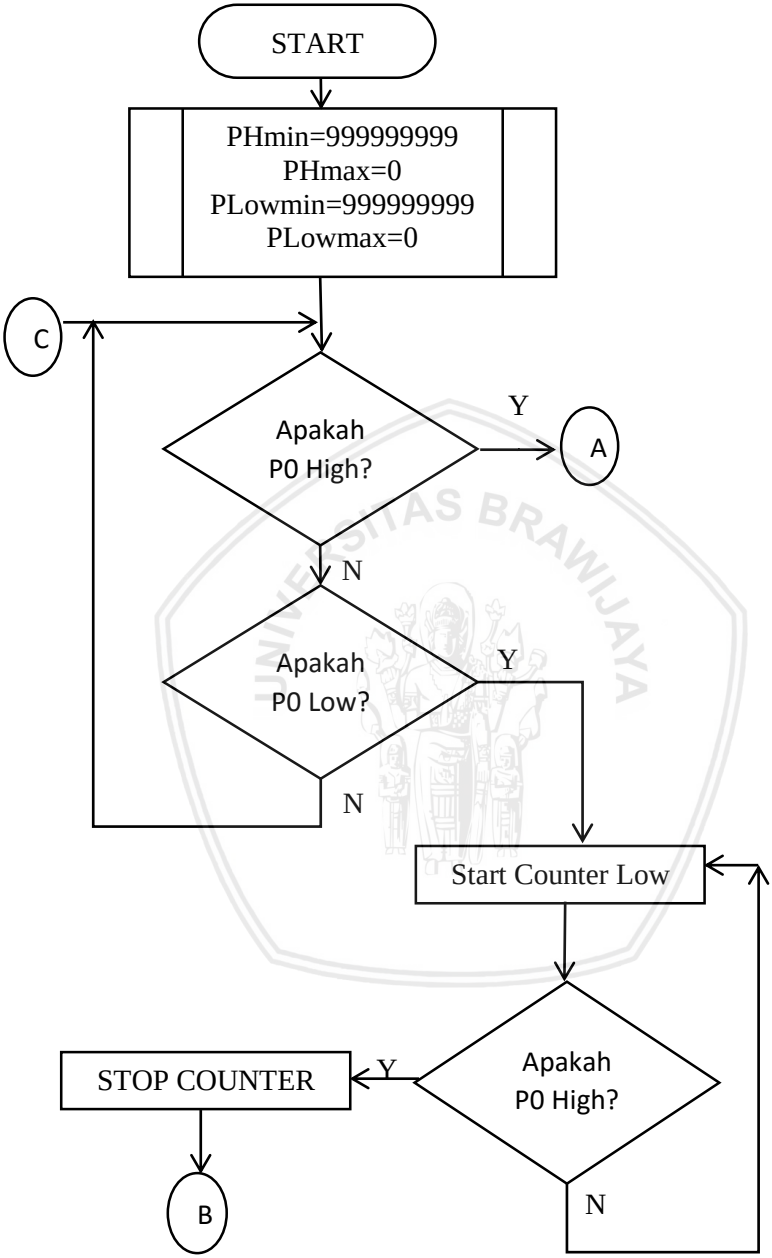
berada pada posisi *stop*. Hasil cacahan counter *time low* disimpan dan counter *low* melakukan pengulangan cacahan untuk *duty cycle* selanjutnya.

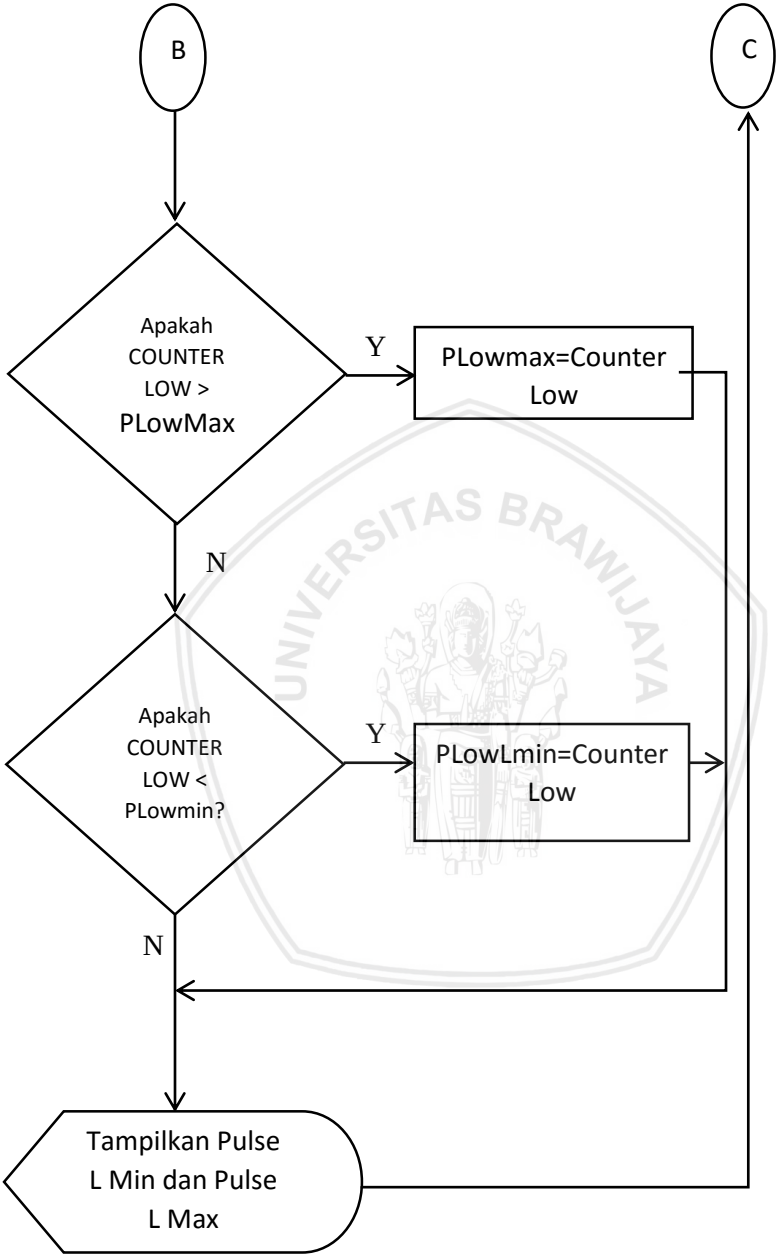


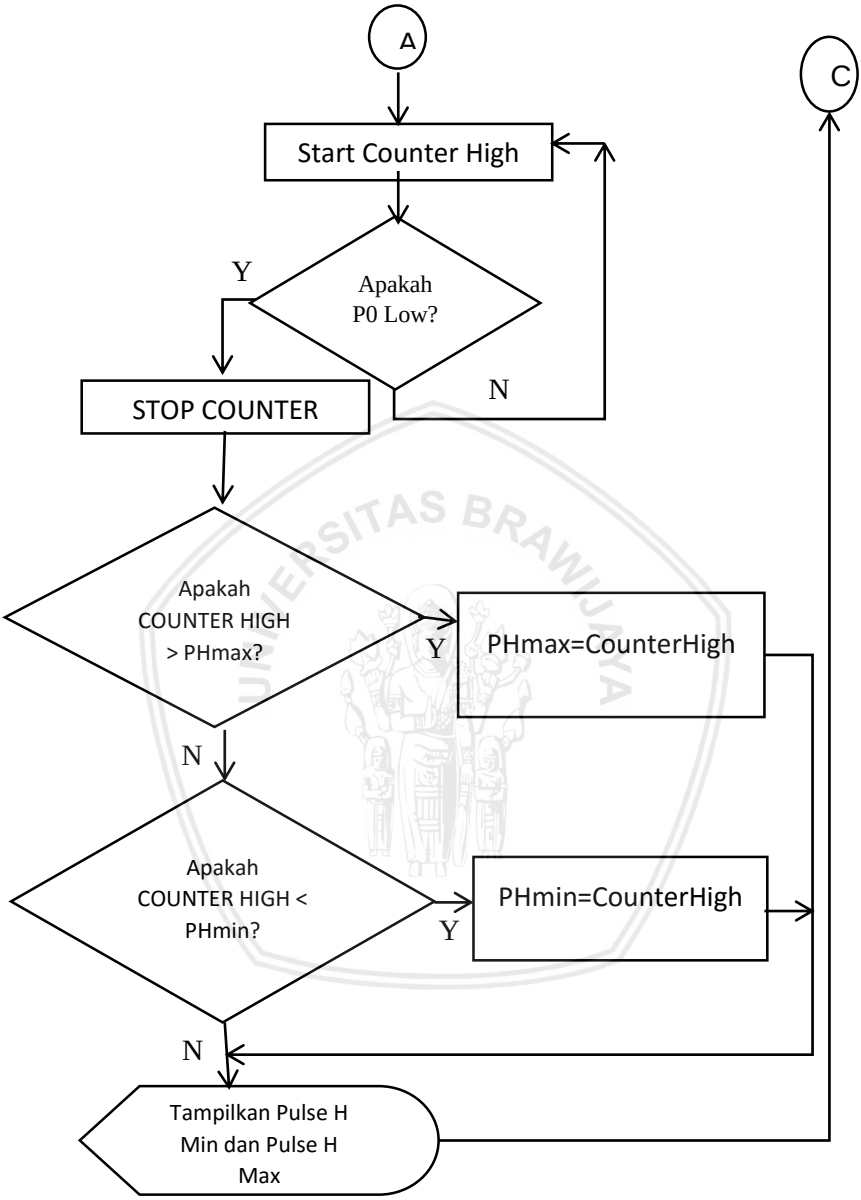
Gambar 3.4 Sisi Naik dan Sisi Jatuh *Time High Clock* Sinyal Input

Hasil cacahan dari counter *high* maupun counter *low* dibandingkan dengan hasil cacahan selanjutnya. Ketika data hasil cacahan *counter* berikutnya memiliki jumlah cacahan lebih banyak maka data yang tersimpan akan diubah dengan data cacahan yang baru dimana memiliki nilai cacahan lebih besar. Perlakuan tersebut akan terus berulang hingga *duty cycle* terahir dan data terahir yang tersimpan adalah data dengan nilai cacahan *time high* dan *time low* terbanyak.

Kemudian untuk lebar time high dan time low terkecilnya, data hasil cacahan di simpan dan ketika data hasil cacahan *counter* selanjutnya memiliki jumlah cacahan lebih sedikit maka data yang tersimpan diubah dengan data cacahan yang baru dimana memiliki nilai cacahan lebih kecil. Jika hasil cacahan lebih besar maka data akan diabaikan atau tidak tersimpan. Perlakuan tersebut akan terus berulang hingga *duty cycle* terahir dan data terahir yang tersimpan adalah data dengan nilai cacahan *time high* dan *time low* paling sedikit atau terkecil. Lebih jelasnya mengenai program FPGA yang didesain dalam penelitian ini dapat dilihat dalam *flowchart* pada Gambar 3.5.


START
PHmin=999999999
PHmax=0
PLowmin=999999999
PLowmax=0
C
Apakah
P0 High?
Y
A
N
Apakah
P0 Low?
Y
N
Start Counter Low
Apakah
P0 High?
Y
STOP COUNTER
B
N


B
C
Apakah COUNTER LOW > PLowMax
Y
PLowmax=Counter Low
N
Apakah COUNTER LOW < PLowmin?
Y
PLowLmin=Counter Low
N
Tampilkan Pulse L Min dan Pulse L Max



Gambar 3.5 *Flowchart* Program FPGA

3.3.2 Perancangan Pogram Arduino Mega 2560

Arduino Mega 2560 difungsikan sebagai penerima data dari FPGA dan penampil data ke LCD display. Arduino Mega dirancang untuk menerima data 16bit dari FPGA. Dengan logika program adalah FPGA mengirimkan data berupa logika 0 pada digital pin 9 Arduino Mega yang menandakan bahwa data yang dikirim adalah data *time low* terbesar yang siap pada port f dan port k. Dimana port f adalah LSB dan port k adalah MSB.

Karena Arduino Mega hanya mampu menerima data 8 bit untuk satu port maka data dipecah kedalam dua port. Sehingga untuk memperoleh data counter 16bit data MSB di geser ke kiri atau Shift Left (SHL) sebanyak 8 bit kemudian di OR kan dengan data LSB dengan demikian diperoleh data angka 16 bit. Lebih jelasnya dapat di liat pada contoh di bawah ini.

MSB = 00000000 00000011 // data MSB asli
MSB = MSB<< 8 // data MSB digeser ke kiri 8 bit

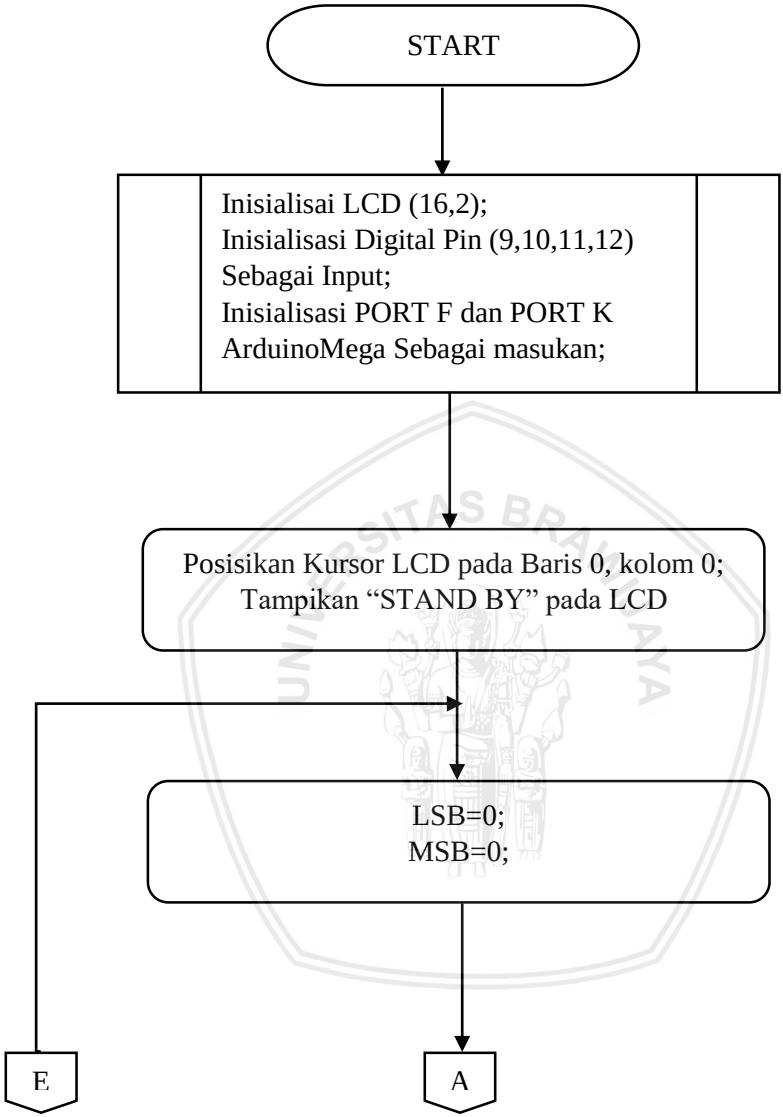
MSB = 00000011 00000000 // hasil data MSB
LSB = 00000000 00001111 //data LSB
_____ OR //MSB or LSB
HSL=00000011 000001111 //hasil MSB or LSB

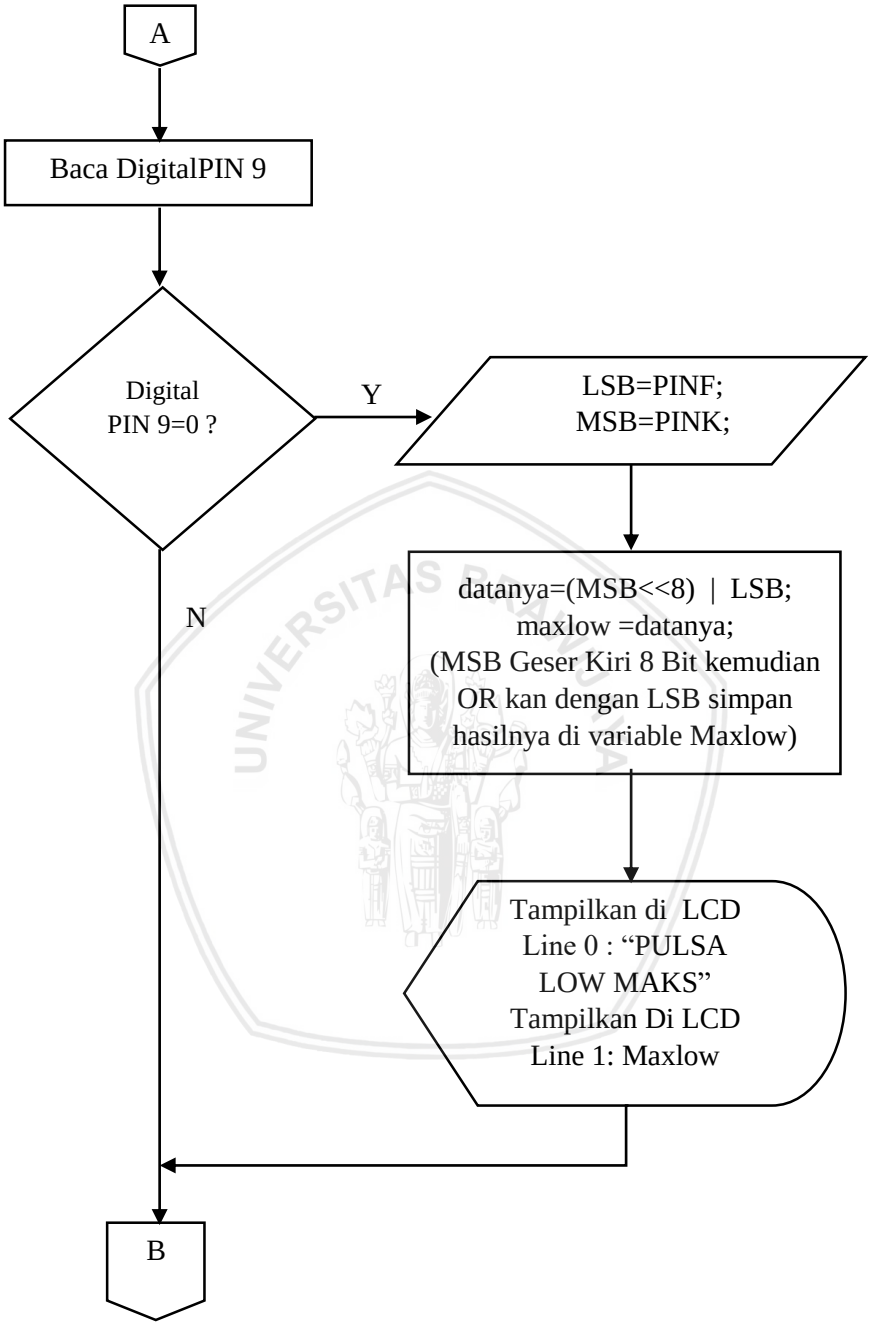
Dipilihnya Arduino Mega 2560 sebagai penampil data cacahan dari time high dan time low sinyal digital adalah Arduino Mega2560 mampu mengolah bilangan sampai 16bit dengan tipe data LONG. Hasil data yang di terima tersebut adalah 00000011000001111 dan dikonfersikan dalam desimal adalah 1551. Bilangan desimal inilah yang ditampilan ke LCD.

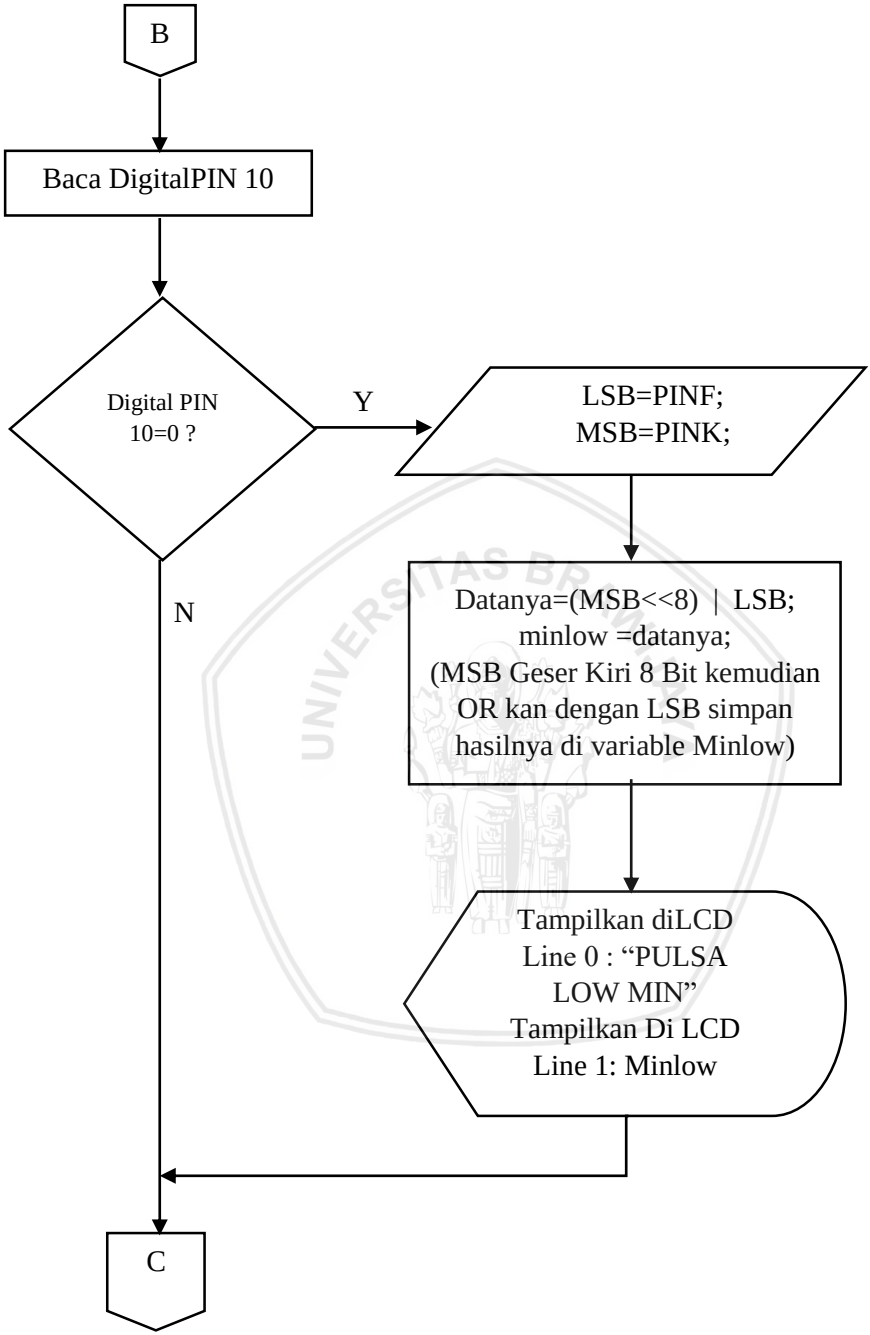
Untuk data cacahan pulsa yang lainya memiliki prinsip yang sama hanya berbeda pin masukan untuk tiap jenis data. FPGA mengirimkan data berupa Logic 0 pada digital pin 10

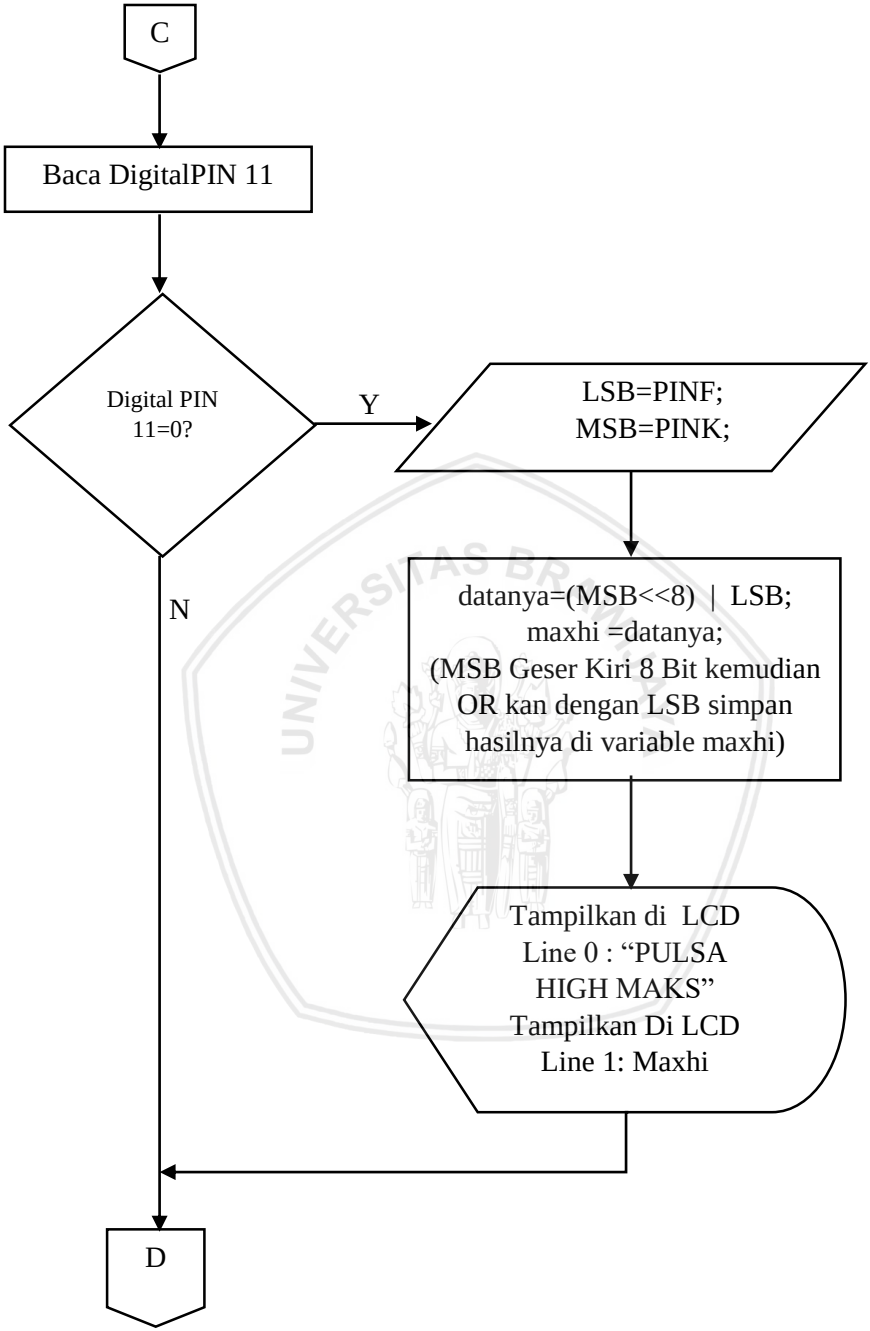
Arduino Mega yang menandakan data *time low* terkecil siap pada port f dan port k. Data cacahan time high terbesar dikirimkan FPGA pada digital pin 11 Arduino Mega dengan logika 0 dan untuk data *time high* terkecil dikirimkan oleh FPGA berupa logika 0 pada digital pin 12 Arduino Mega.

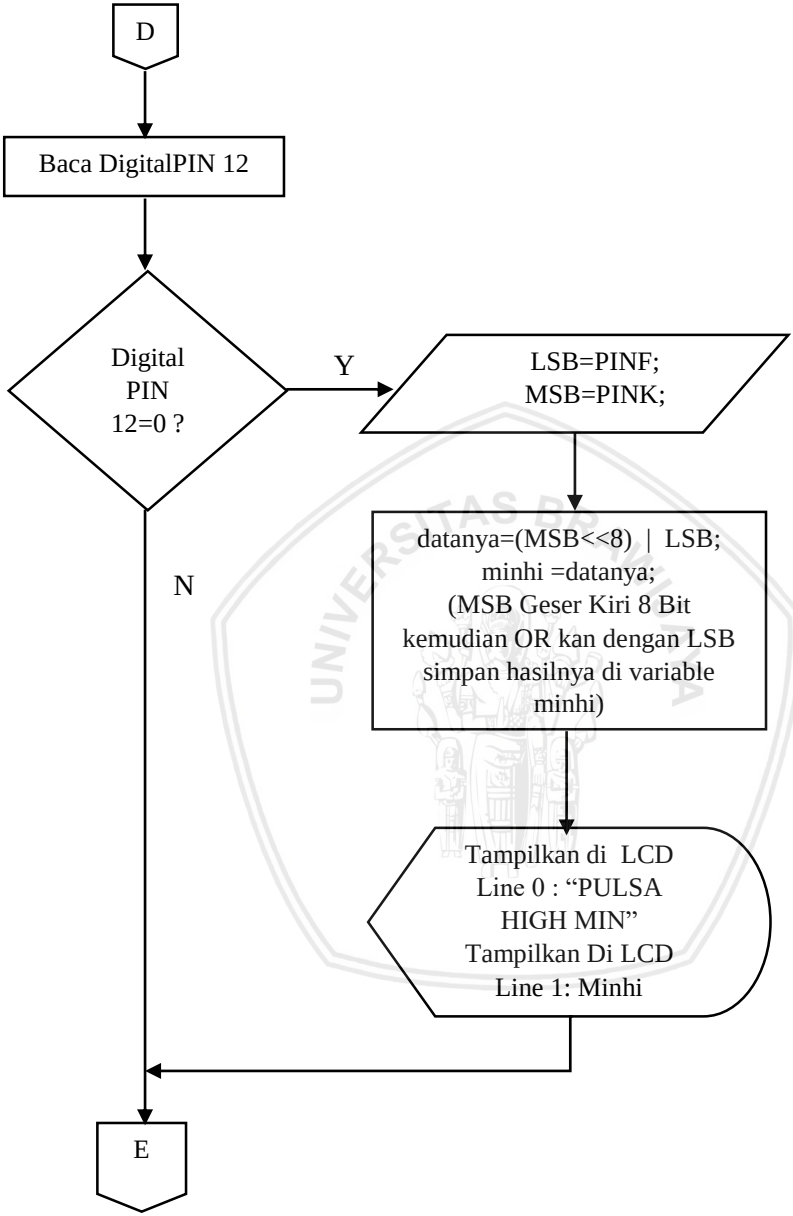
Lebih jelasnya mengenai program Arduino Mega 2560 sebagai penerima dan penampil data dari FPGA yang didesain dalam penelitian ini dapat dilihat dalam *flowchart* Gambar 3.6.


START
Inisialisai LCD (16,2);
Inisialisasi Digital Pin (9,10,11,12)
Sebagai Input;
Inisialisasi PORT F dan PORT K
ArduinoMega Sebagai masukan;
Posisikan Kursor LCD pada Baris 0, kolom 0;
Tampikan “STAND BY” pada LCD
LSB=0;
MSB=0;
E
A


A
Baca DigitalPIN 9
Digital
PIN 9=0 ?
Y
N
LSB=PINF;
MSB=PINK;
datanya=(MSB<<8) | LSB;
maxlow =datanya;
(MSB Geser Kiri 8 Bit kemudian
OR kan dengan LSB simpan
hasilnya di variable Maxlow)
Tampilkan di LCD
Line 0 : “PULSA
LOW MAKS”
Tampilkan Di LCD
Line 1: Maxlow
B


B
Baca DigitalPIN 10
Digital PIN
10=0 ?
Y
LSB=PINF;
MSB=PINK;
N
Datanya=(MSB<<8) | LSB;
minlow =datanya;
(MSB Geser Kiri 8 Bit kemudian
OR kan dengan LSB simpan
hasilnya di variable Minlow)
Tampilkan diLCD
Line 0 : “PULSA
LOW MIN”
Tampilkan Di LCD
Line 1: Minlow
C


C
Baca DigitalPIN 11
Digital PIN 11=0?
Y
N
LSB=PINF;
MSB=PINK;
datanya=(MSB<<8) | LSB;
maxhi =datanya;
(MSB Geser Kiri 8 Bit kemudian OR kan dengan LSB simpan hasilnya di variable maxhi)
Tampilkan di LCD Line 0 : "PULSA HIGH MAKS"
Tampilkan Di LCD Line 1: Maxhi
D



Gambar 3.6 *Flowchart* Program Arduino Mega 2560

3.3.3 Perancangan Program Arduino Uno

Arduino Uno difungsikan sebagai penghasil sinyal digital masukan. Membuat sinyal kotak dengan arduino dapat dilakukan dengan memanfaatkan perintah delay yang ada pada arduino. *Syntax* perintah delay pada arduino adalah delayMicroseconds(). Agar diperoleh sinyal digital dengan frekuensi 1Khz dari Arduino uno maka dengan *Duty Cycle* 100% (50% *time high* dan 50% *time low*) program akan dibuat untuk memberikan logika 1 pada keluaran Arduino yaitu digital pin 2 kemudian didelay selama 500 Microsecond untuk setengah *duty cycle* atau 50% *time high*. Kemudian digital PIN 2 kita set menjadi logika 0 di ikuti dengan delay yang sama yaitu 500 microsecond untuk setengah *duty cycle* selanjutnya atau 50% *time low*. Dngan demikian diperoleh sinyal digital dengan frekuensi 1Khz. Program akan diulang terus menerus hingga arduino uno bisa digunakan menjadi sebuah *Clock* Generator untuk masukan FPGA.

Diperolehnya nilai 500 microsecond untuk sinyal digital dengan frekuensi 1Khz pada penjelasan diatas adalah dengan rumus frekuensi

$$F = 1/T$$

dengan T adalah periode sinyal. Sehingga untuk sinyal digital dengan frekuensi 1 KHz dapat diperoleh.

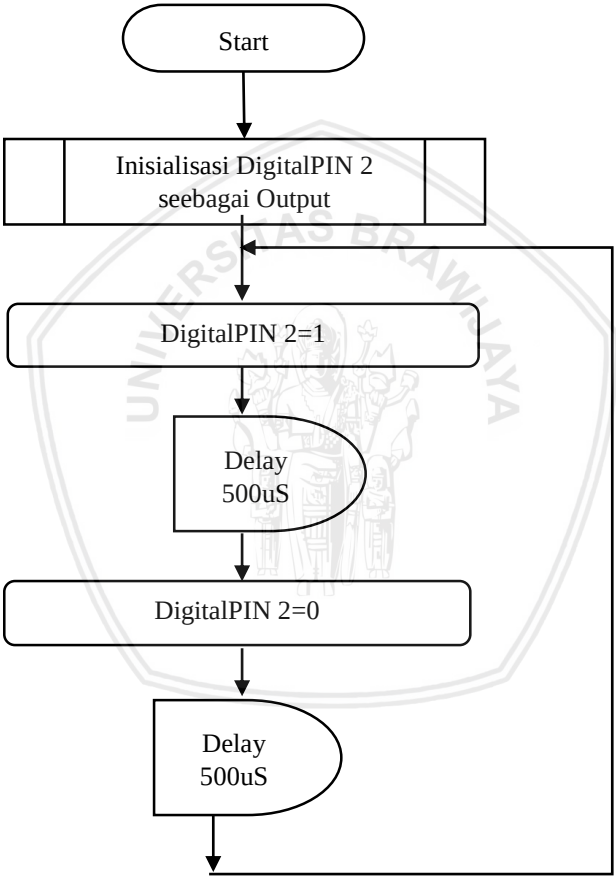
$$1\ KHz = 1000\ Hz$$
$$F = 1/\ T$$
$$T = 1/1000$$
$$= 0.0001S$$
$$= 1mS$$
$$= 1000\ uS$$

Nilai 1000 uS merupakan nilai dari *Duty Cycle* 100%. Karena *duty cycle* terdiri dari 50% *time high* dan 50% *time low* maka untuk memperoleh sinyal digital dengan perintah

delay periodenya dibagi 2 (*time high* 50% dan *time low* 50%).

T/2 = 1000/2= 500uS.

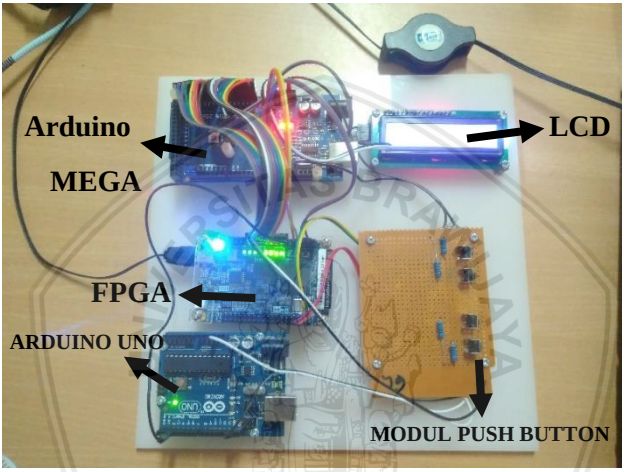
Dengan demikian digunakan 500uS untuk *delay time high* dan 500uS untuk *delay time low*. Lebih Jelasnya dapat dilihat pada *flowchart* Gambar 3.7.



Gambar 3.7 *Flowchart* Program Arduino Uno

3.3.4 Pengujian Alat

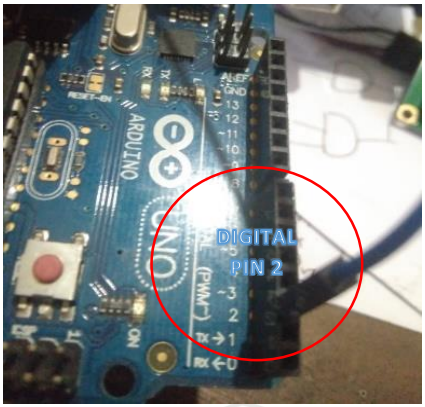
Pengujian alat dilakukan dengan cara menghubungkan FPGA pada sumber sinyal yang berupa sinyal generator dan Arduino uno (tidak dipasang secara bersamaan). Kemudian program VHDL diupload dari PC ke Board Altera. Jika program sudah berjalan dengan benar maka hasil cacahan lebar *time high* terbesar dan terkecil serta lebar *time low* terbesar dan terkecil dari sumber sinyal akan ditampilkan pada Display LCD.



Gambar 3.8 Rangkaian Alat dengan Sinyal Input berasal Dari Arduino Uno

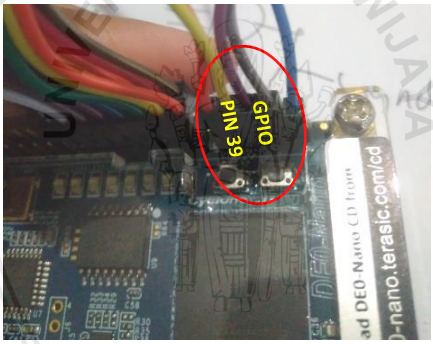
Pemasangan alat untuk pengambilan data pada penilitian kali ini memiliki 2 jenis. Pertama pemasangan untuk pengambilan data dengan sinyal masukan dari Arduino Uno. Kedua pemasangan alat dengan sinyal masukan berasal dari sinyal generator. Pada dasarnya seluruh rangkaian dipasang tetap dan tidak merubah pin apapun hanya mengganti kabel data masukan dari jenis board yang digunakan sebagai sinyal masukan.

Pada sinyal masukan yang berasal dari Arduino uno maka kabel data dipasang pada pin 2 arduino uno yang difungsikan sebagai sumber sinyal seperti Gambar 3.9.



Gambar 3.9 Kabel Data dipasang pada pin 2 Arduino Uno

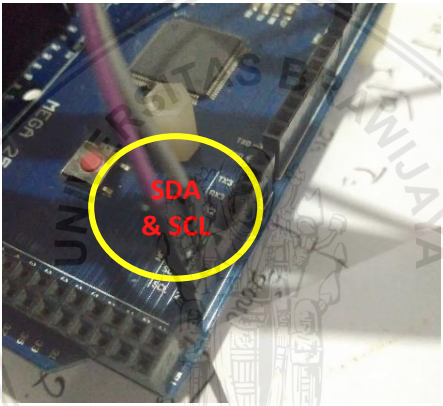
Digital pin 2 pada arduino uno dihubungkan ke GPIO FPGA pin nomor 39 (D12).



Gambar 3.10 Kabel Data Masukan Sinyal Pada FPGA

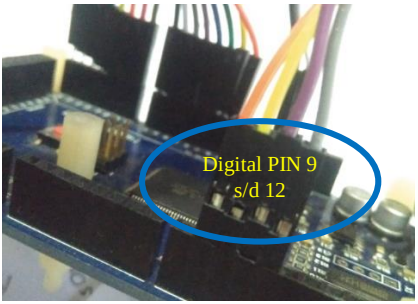
Pada board Arduino Mega 2560 kabel data LCD I2C dipasang pada PIN SDA, SCL dan pin power supply LCD diambil dari board Arduino Mega.

Gambar 3.11 Kabel Data pada LCD

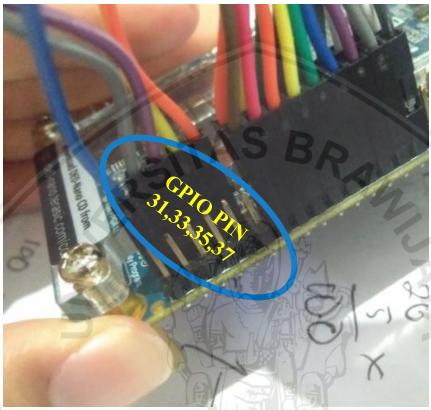


Gambar 3.12 Kabel Data SDA dan SCL pada Arduino Mega 2560

Kabel data untuk indikator jenis data yang dikirimkan oleh FPGA yaitu pada GPIO pin nomor 31, 33, 35, 37 (C9, E11, C11, A12) dihubungkan ke Arduino Mega 2560 pada pin 9, 10, 11, 12.



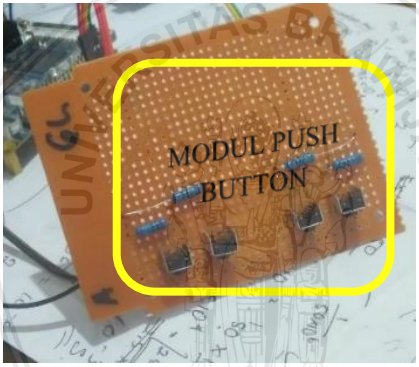
Gambar 3.13 Kabel Data Indikator pada Arduino Mega 2560



Gambar 3.14 Kabel Data Indikator pada GPIO FPGA

Pin data selektor dari modul push button yang berfungsi sebagai selektor pengiriman data dihubungkan ke GPIO pin 2, 4, 6 dan 8 pada FPGA.

Gambar 3.15 Kabel Data Selektor pada GPIO FPGA



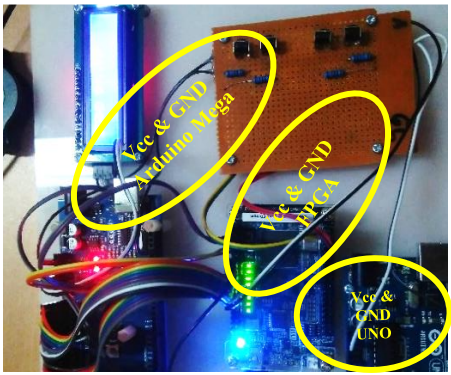
Gambar 3.16 Modul *Push Button* sebagai Selektor

Kemudian 16 bit data output dari FPGA yang berada pada GPIO pin 13 sampai dengan 28 dihubungkan ke port K dan port F Arduino Mega.

Gambar 3.17 Kabel Data untuk pengiriman 16 bit data hasil cacahan pada FPGA

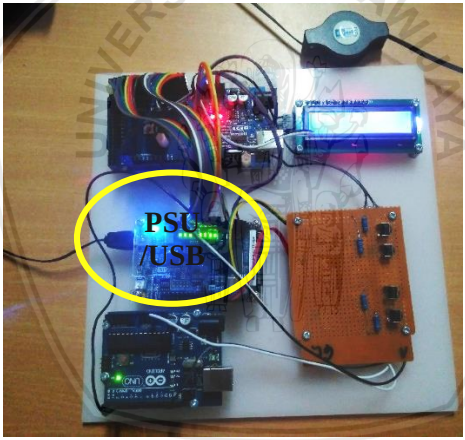
Gambar 3.18 Kabel Data untuk penerimaan 16 bit data hasil cacahan pada Arduino Mega 2560

Sedangkan pin VCC dan GND untuk semua board (FPGA, Arduino uno, dan Arduino Mega 2560) dihubungkan ke sumber tegangan yang dijumper pada modul tombol *push button*.



Gambar 3.19 Kabel Data VCC dan GND dari seluruh board

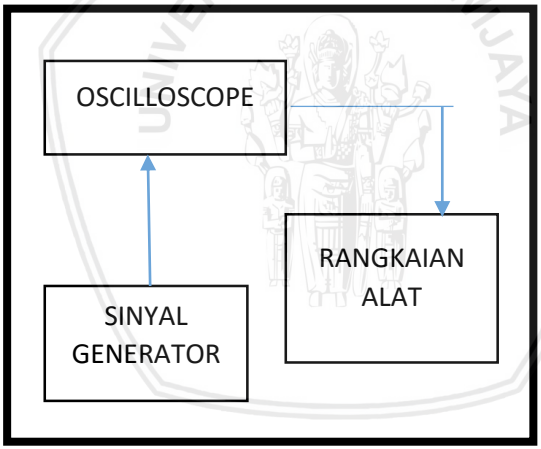
Setelah semua pin terkoneksi, maka power supply dari Board DE0-NANO dipasang melalui kabel USB.



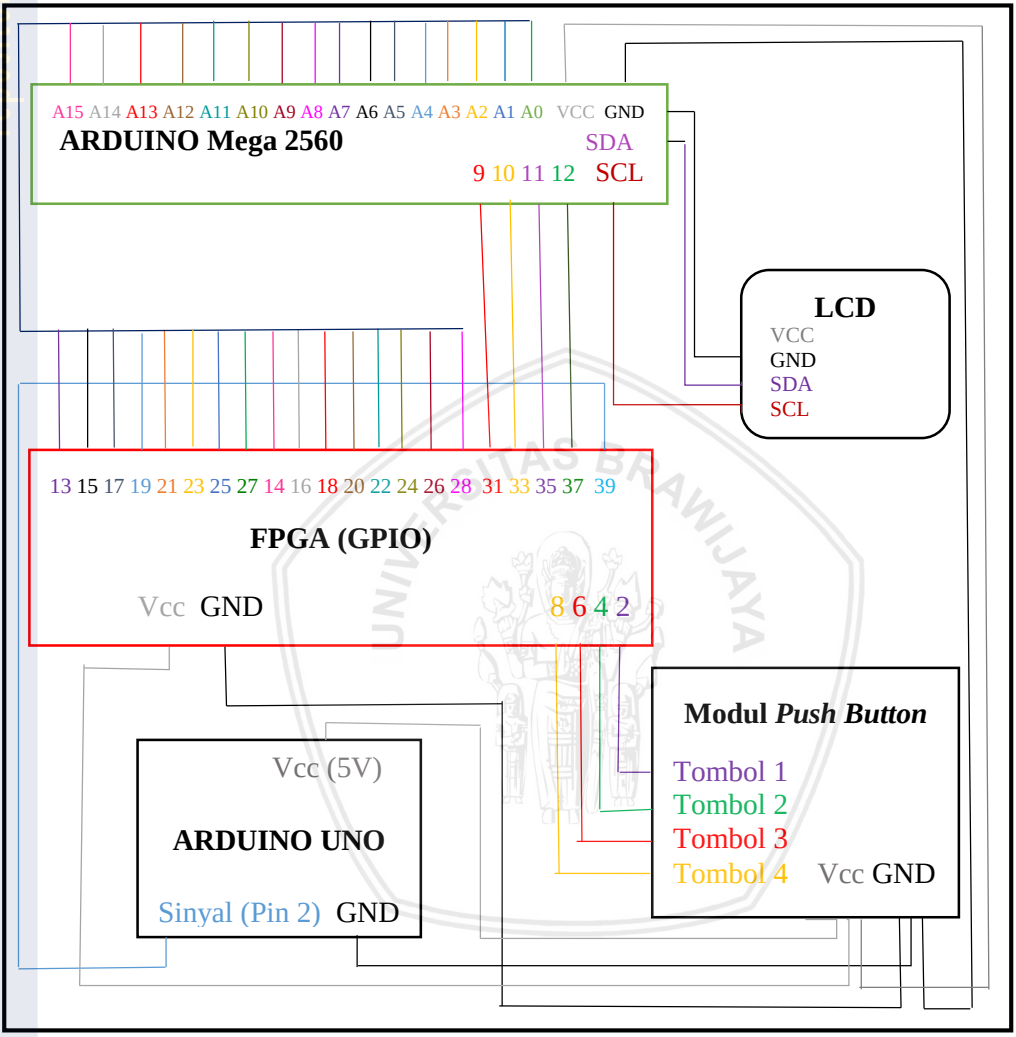
Gambar 3.20 Kabel USB FPGA untuk Power Supply seluruh rangkaian

Sedangkan pemasangan alat untuk masukan yang berasal dari sinyal generator maka kabel data sinyal input dan ground (sinyal generator) dihubungkan ke GPIO FPGA nomor 39 dan Ground dan kabel data input dari arduino dilepas (kabel masukan arduino diganti dengan kabel masukan sinyal generator).

Gambar 3.21 Rangkaian Alat dengan Sinyal Masukan berasal dari Sinyal Generator



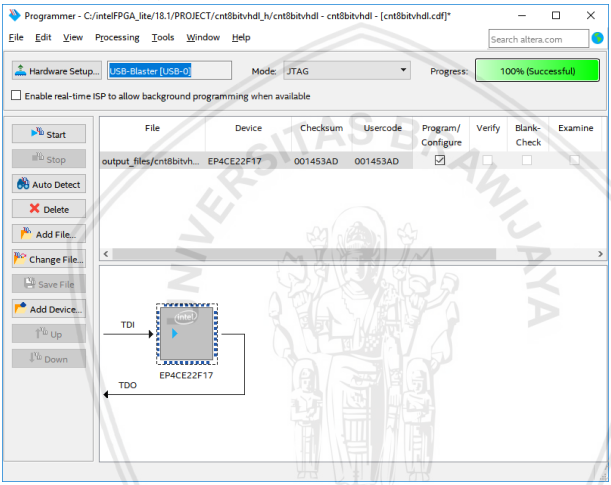
Gambar 3.22 Diagram Rangkaian Alat dengan Sinyal Input berasal dari Sinyal Generator



Gambar 3.23 Diagram Rangkaian Alat dengan Sinyal Input berasal dari Arduino Uno

3.3.5 Pengambilan Data

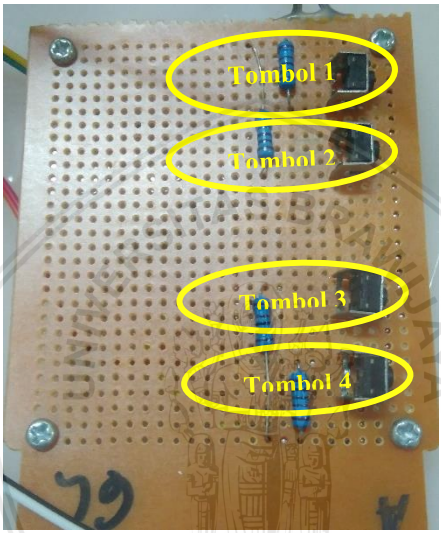
Proses pengambilan data dilakukan dari dua jenis sinyal masukan. Pertama sinyal masukan yang berasal dari Arduino uno dan yang kedua sinyal masukan dari sinyal generator. Pengambilan data dengan sinyal masukan berasal dari board Arduino uno dilakukan dengan menyalakan semua Board yang terkoneksi dengan DE0-NANO kemudian program VHDL yang telah decompile diupload ke Board DE0-NANO melalui Quartus Prime Lite Edition.



Gambar 3.24 Mengupload program VHDL ke board FPGA

Setelah program VHDL sukses 100% diupload ke Board DE0-Nano tekan tombol reset untuk memulai pengukuran baru. Arduino Uno akan otomatis mengirimkan sinyal masukan dan FPGA secara otomatis mencacah sinyal yang diterima. Untuk menampilkan hasil cacahan pada LCD dapat menekan tombol pada modul *push button*. Ada 4 tombol dimana tombol 1 untuk mengirimkan data hasil cacahan *time high* terbesar dari FPGA ke Arduino Mega 2560, tombol 2 untuk cacahan *time high* terkecil, tombol 3 untuk cacahan *time low* terbesar, dan tombol 4 untuk cacahan *time low*

terkecil. Pengambilan data dilakukan sebanyak dua kali pengulangan tiap satu frekuesi sinyal masukan. Dengan frekuensi sinyal digital yang digunakan untuk pengambilan data adalah 1Khz hingga 10Khz, 20Khz hingga 100Khz (dengan kelipatan 10), dan 100Khz hingga 500Khz (dengan kelipatan 100). Bila tombol pada modul push button ditekan maka secara otomatis hasil cacahan akan ditampilkan pada display LCD.



Gambar 3.25 Tombol Selektor Pengiriman Data Cacahan

Gambar 3.26 Tampilan Hasil Cacahan pada layar LCD

Untuk pengambilan data dengan frekuensi yang berbeda maka power supply utama (USB FPGA) dilepas agar saat

pengambilan data untuk frekuensi selanjutnya tidak terjadi eror atau kurang presisi dan rangkaian tidak rusak.

Sedangkan pengambilan data untuk sinyal masukan yang berasal dari sinyal generator pada dasarnya memiliki prinsip yang sama dengan pengambilan data dengan sinyal masukan yang berasal dari Arduino. Perbedaan yang ada dari kedua alat tersebut adalah frekuensi sinyal digital yang digunakan dari input sinyal generator 1Khz hingga 10Khz, 20Khz hingga 100Khz (dengan kelipatan 10), 100Khz hingga 1000Khz (dengan kelipatan 100), dan 1Mhz hingga 15Mhz.

BAB IV
HASIL DAN PEMBAHASAN

1.1 Implementasi Program FPGA

1.1.1 Program FPGA

Menghitung lebar *time high* dan *time low* sinyal digital dengan menggunakan FPGA dibutuhkan dua buah counter 16Bit dengan. Entity dari VHDL untuk program pencacah *time high* dan *time low* sinyal digital terdiri dari deklarasi input dan deklarasi output. Dimana untuk input terdiri dari CLK yaitu pin masukan dari Sistem Clock Generator berfungsi sebagai clock sampling yang dihubungkan ke pin R8 pada FPGA. CLK input ini memiliki sistem clock dengan frekuensi 50MHz. RST yaitu pin reset untuk mereset sistem counter. CE yaitu pin Clock Enable atau input sinyal dimana pin CE inilah yang digunakan sebagai input dari sinyal yang diukur. Dua CE (CE dan CE2) digunakan untuk melakukan proses cacahan, CE untuk *time low* dan CE2 untuk *time high*. Counter *time low* mulai bekerja saat CE berlogika 0 dan sebaliknya counter *time high* mulai bekerja saat CE berlogika 1. Kemudian t1 hingga t4 yang merupakan masukan pin tombol 1 sampai 4 dari modul push button untuk mengirim jenis data cacahan. Dimana t1 untuk *time high* terbesar, t2 untuk *time high* terkecil, t3 untuk *time low* terbesar, dan t4 untuk *time low* terkecil. Sedangkan untuk outputnya lMax sebagai data indikator untuk *time low* terbesar, lmin sebagai data indikator untuk *time low* terkecil, HMax sebagai data indikator untuk *time high* terbesar, Hmin sebagai data indikator untuk *time high* terkecil, dan oFA sebagai pin Output 16 Bit untuk hasil cacahan counter.

Architecture pada program counter ini dapat dijelaskan sebagai berikut. Pendeklarasian signal counter sebagai menyimpan hasil cacahan counter *time low*. Signal counterh sebagai menyimpan hasil cacahan counter *time high*. Signal data1 sebagai menyimpan hasil cacahan sementara dari *time low*. Signal data2 sebagai menyimpan hasil cacahan

sementara dari *time high*. Signal lowmax sebagai menyimpan hasil cacahan *time low* terbesar. Signal lowmin sebagai menyimpan hasil cacahan *time low* terkecil. Signal highmax sebagai menyimpan hasil cacahan *time high* terbesar. Signal highmin sebagai menyimpan hasil cacahan *time high* terkecil. Signal CLK1 sebagai clock pencacah *time low*. Signal CLK2 sebagai clock pencacah *time high*. Signal CE2 sebagai signal untuk masukan *time high* yang di ukur sedangkan untuk *time low* menggunakan signal CE. Signal CE ini dihubungkan dengan *Virtual Wire* ke CE2. Signal RST2 sebagai Signal untuk reset counter low. Signal RST1 sebagai signal untuk reset counter high. Kemudian menghubungkan signal CLK1 dengan CLK, CLK2 dengan CLK, CE2 dengan CE, RST1 dengan RST, dan RST2 dengan RST menggunakan Virtual Wire.

Proses untuk penghitungan *time low* adalah jika PIN RST1 berlogika 0 maka counter di reset pada keadaan awal yaitu 0 hasil perhitungan lowmax dan lowmin juga dikembalikan pada keadaan awal yaitu lowmax = 0000000000000000 dan lowmin = 1111111111111111. Jika CE=0 maka sinyal input berada pada time low sehingga counter mulai menghitung. Counter mulai menghitung saat CE mendeteksi adanya sisi jatuh dari Pin CLK1 yang terhubung pada entity CLK diamana adalah pin sistem clock 50MHz. Jika CE mendeteksi adanya sisi naik maka counter berhenti mencacah (panjang *time low* berakhir) dan hasil cacahan counter disimpan ke signal data1 sebagai buffer data sementara. Kemudian dilakukan pembandingan untuk diperoleh hasil *time low* terbesar dan terkecil. Jika signal lowmax lebih kecil dari data1 maka nilai dari data1 disimpan ke signal lowmax. Jika signal lowmin lebih besar dari data1 maka nilai dari data1 disimpan ke signal lowmin. Jika CE berogika 1 maka counter di reset kembali ke 0.

Proses untuk penghitungan time high adalah jika PIN RST2 berlogika 0 maka counter direset pada keadaan awal yaitu 0, hasil perhitungan highmax dan highmin juga

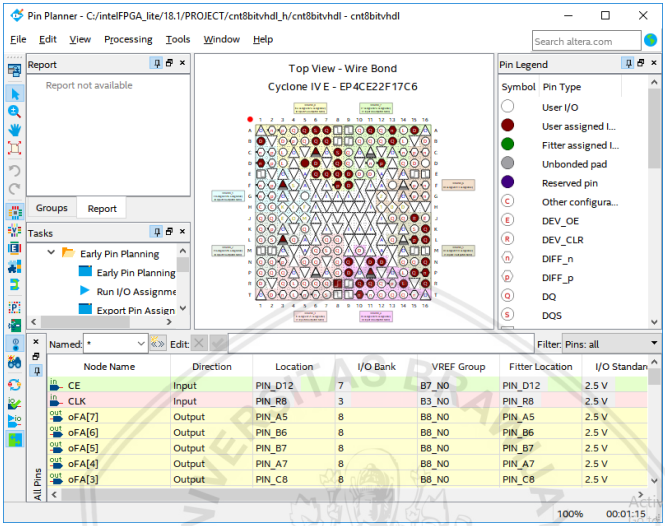
dikembalikan pada keadaan awal. Jika CE2=1 maka sinyal input berada pada *time high* sehingga counter mulai menghitung. Counter mulai menghitung saat CE2 mendeteksi sisi naik dari Pin CLK2 yang terhubung pada entity CLK. Jika CE2 mendeteksi sisi turun maka panjang *time high* telah selesai dan hasil cacahan counter di simpan ke signal data2 sebagai buffer data sementara. Jika signal highmax lebih kecil dari data2 maka nilai dari data2 di masukkan ke signal highmax. Jika signal highminn lebih besar dari data2 maka nilai dari data2 di masukkan ke signal highmin. Jika CE2 berlogika 0 maka counter di reset kembali ke 0

FPGA mengeluarkan data 16 bit biner dengan 4 jenis data yaitu *time high* terbesar, *time high* terkecil, *time low* terbesar, dan *time low* terkecil. Proses untuk pengiriman data 16 bit berserta jenis datanya adalah jika tombol 1 pada modul push button di tekan maka entity t1 berlogika 0 dan data highmax dikirim ke entity oFA yang dimaping ke GPIO 16Bit. Kemudian PIN hmax diset berlogika 0 yang menandakan data siap untuk dibaca Arduino Mega 2560. Jika tombol 2 ditekan entity t2 berlogika 0 maka data highmin dikirim ke entity oFA yang dimaping ke GPIO 16Bit. Kemudian PIN hmin diset berlogika 0 yang menandakan data siap untuk dibaca Arduino Mega 2560. Jika tombol 3 ditekan maka entity t3 berlogika 0 dan data lowmax dikirim ke entity oFA yang dimaping ke GPIO 16Bit. Kemudian PIN lmax diset berlogika 0 yang menandakan data siap di baca Arduino Mega 2560. Jika tombol 4 ditekan entity t4 berlogic 0 dan data lowmin dikirim ke entity oFA yang dimaping ke GPIO 16Bit. Kemudian PIN lmin diset berlogika 0 yang menandakan data siap untuk di baca Arduino Mega 2560.

1.1.2 Mapping PIN FPGA.

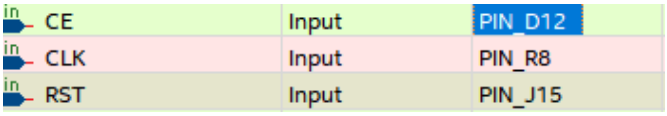
Setelah program selesai dibuat dan sukses decompile yang perlu dilakukan selanjutnya adalah menentukan PIN dari board DE0-NANO yang mana akan digunakan sebagai CLK, CE, Reset, oFA, t1, t2, t3, t4, lmin, lmax, hmin, dan

hmax. Pembuatan mapping PIN FPGA pada Quartus Prime Lite Edition dilakukan dengan memilih menu **Assigment -> PIN Planer** maka muncul jendela pin planer seperti gambar 4.1



Gambar 4.1 Menu Pin Planner pada Quartus Prime Lite Edition

Dalam mapping pin sendiri ketentuan untuk menetapkan posisi pin-pin yang digunakan adalah bebas kecuali untuk Vcc dan Ground yang telah ditetapkan secara permanen. Sehingga untuk pin yang lainnya dapat diatur sesuai posisi yang di inginkan. Untuk mengetahui posisi pin yang digunakan berada pada board FPGA bagian mana, maka di butuhkan denah GPIO dari FPGA DE0-NANO.



Gambar 4.2 Maping PIN CE, CLK dan RST

Pada gambar 4.2 ditetapkan bahwa PIN_D12 sebagai CE (Clock Enable) atau sinyal masukan yang akan diukur. CLK dihubungkan ke PIN_R8 dimana pin tersebut merupakan PIN

Sistem clock dari Altera FPGA. RST berada pada PIN_J15 yang merupakan tombol yang ada pada Board FPGA DE0-NANO.

Selanjutnya Mapping Pin untuk t1, t2, t3, t4 dan lmin, lmax, hmin, hmax sebagai data selektor dan data indikator untuk dibaca mikrokontroller Arduino Mega 2560.

Node Name	Direction	Location	I/O Bank
t1	Input	PIN_T15	4
t2	Input	PIN_F13	6
t3	Input	PIN_T12	4
t4	Input	PIN_T13	4

Node Name	Direction	Location	I/O Bank
HMax	Output	PIN_C11	7
Hmin	Output	PIN_A12	7
lMax	Output	PIN_C9	7
lmin	Output	PIN_E11	7

Gambar 4.3 Mapping Pin untuk t1, t2, t3, t4, lmin, lmax, hmin, dan hmax

Sedangkan Mapping Output untuk entity oFA yang merupakan data keluaran 16 bit ditunjukkan pada Gambar 4.4

Node Name	Direction	Location	I/O Bank
oFA[15]	Output	PIN_D5	8
oFA[14]	Output	PIN_A6	8
oFA[13]	Output	PIN_D6	8
oFA[12]	Output	PIN_C6	8
oFA[11]	Output	PIN_E6	8
oFA[10]	Output	PIN_D8	8
oFA[9]	Output	PIN_F8	8
oFA[8]	Output	PIN_E9	7
oFA[7]	Output	PIN_A5	8
oFA[6]	Output	PIN_B6	8
oFA[5]	Output	PIN_B7	8
oFA[4]	Output	PIN_A7	8
oFA[3]	Output	PIN_C8	8
oFA[2]	Output	PIN_E7	8
oFA[1]	Output	PIN_E8	8
oFA[0]	Output	PIN_F9	7

Gambar 4.4 Mapping Pin untuk output data (OFA)

1.2 Program Mikrokontroller Arduino UNO

Dalam membuat sebuah sinyal digital yang berupa pulsa dengan duty cycle 100% (time high 50% dan time low 50%) yang digunakan sebagai input signal FPGA, dapat digunakan Arduino UNO untuk menghasilkan sinyal dengan frekuensi 1Khz hingga 500Khz. Digital pin 2 diset sebagai Output dimana pin inilah sinyal pulsa akan dibuat

```
void loop() {
digitalWrite(2,HIGH); // Set PIN 2Time High (1)
delayMicroseconds(500); // Delay ½ Duty Cycle
digitalWrite(2,LOW); // Set PIN 2 Time Low(0)
delayMicroseconds(500); // Delay ½ Duty Cycle
    }
```

Nilai dari delayMicroseconds() diubah sesuai dengan frekuensi yang diinginkan. Nilai dari delay tersebut hanya dapat diset dalam bilangan bulat. Dengan demikian saat hasil perhitungan diperoleh nilai yang pecahan maka dibulatkan terlebih dahulu sebelum dimasukan pada program. Sehingga nilai terkecil yang dapat dimasukkan adalah delaymicrosecons(1) yang menandakan periode dari sinyal adalah 2 uS. Perlu diingat dalam bab 3 bahwa nilai delaymicroseconds dapat diketahui dengan membagi 2 periode dari sinyal (T/2). Saat periode dari sinyal adalah 2uS maka frekuensi sinyal tersebut adalah 500KHz yang secara langsung menandakan frekuensi terbesar sinyal yang dapat dihasilkan oleh Arduino uno adalah 500Khz.

Tabel 1.1 Daftar Delay untuk Arduino Uno

Frek (Hz)	Delay (us)
1000	500
2000	250
3000	166.6666667
4000	125
5000	100

6000	83.33333333
7000	71.42857143
8000	62.5
9000	55.55555556
10000	50
20000	25
30000	16.66666667
40000	12.5
50000	10
60000	8.333333333
70000	7.142857143
80000	6.25
90000	5.555555556
100000	5
200000	2.5
300000	1.666666667
400000	1.25
500000	1

Frek pada tabel 1.1 merupakan nilai frekunsi yang diharapkan dan delay adalah periode yang di masukan pada program delay.

1.3 Program Arduino Mega 2560

Data hasil cacahan oleh FPGA DE0-NANO dapat ditampilkan mengguakan mikrokontroller Arduino Mega 2560. Dengan penjelasan programnya adalah variable minlow, maxlow, minhi, maxhi didevinisikan dengan nilai awal 0. LCD 16 Karakter 2 Baris di aktifkan. Digital PIN 9, PIN 10, PIN 11, PIN12, PIN PORT F dan K diset sebagai Input. Pada LCD baris pertama ditampilakn tulisan STAND BY yang menjadi tanda bahwa LCD berada pada kondisi bersiap. Kemudian variable lsb1 dan msb1 didefinisikan dengan nilai awal 0.

Saat data cacahan dari FPGA di terima oleh arduinomega maka data 16 bit akan masuk pada port F dan K dan jenis datanya melalui pin 9 hingga 12 sesuai jenisnya. Data pada port F simpan ke variable lsb1 dan data pada port K pada msb1. Geser 8Bit

untuk data MSB1 kemudian di OR kan dengan lsb1 untuk menghasilkan data utuh 16bit dan dikonfersikan ke desimal. Data hasil kombinasi lsb dan MSB disimpan ke variable sesuai jenis datanya. Kemudian menampilkan tulisan jenis data yang ingin di tampilkan ke LCD pada baris pertama dan data cacahan dalam desimal ditampilkan pada baris ke 2

1.4 Hasil Cacahan Counter FPGA

Hasil cacahan oleh FPGA didibandingkan menggunakan nilai cacahan pembanding *time high* dan *time low* dari sinyal digital. Hal tersebut di lakukan untuk mengetahui kualitas counter FPGA yang telah diprogran baik dan memenuhi syarat untuk digunakan sebagai pencacah *time high* dan *time low* atau tidak yaitu tidak memiliki perbedaan yang sangat jauh hingga hampir setengah dari nilai cacahan pembandingnya. Cacahan pembanding dapat diketahui dengan cara frekuensi sinyal yang berasal dari sinyal generator atau arduino uno di cek pada oscilloscope kemuadian pada oscilloscope akan muncul dan terdeteksi frekuensi sebenarnya (real frekuensi) dari sinyal hasil generator atau Arduino uno. Kemudian nilai cacahan pembanding dapat diperoleh dari membagi periode dari *time high* atau *time low* (riode *time high* atau *time low* merupakan periode sinyal dibagi 2 hal ini karena *duty cycle* terdiri dari 50% *time high* dan 50% *time low* sehingga besar periode *time high* dan *time low* adalah sama) dari real frekuensi dengan periode *clock* counter. Lebih jelasnya dapat dilihat pada contoh di bawah ini

Sinyal digital yang berasal dari sinyal generator atau arduino uno dengan frekuensi 1 Khz memiliki real frekuensi 990.1 Hz setelah dilakukan pengecekan pada oscilloscope. Sehingga periode real freakuensinya sebesar 1/990.1 s atau 0.00101s (diperoleh dari rumus T = 1/F). 0.00101s merupakan periode untuk *duty cycle* secara utuh maka untuk mengetahui periode *time high* atau *time low* maka periode *duty cycle* tersebut dibagi 2. Dengan demikian diperoleh 0.0005s untuk periode dari *time high* atau *time low* sinyal digital dengan real frekuensi 990.1 Hz. Periode dari *time high* atau *time low* tersebut kemudian dibagi

dengan periode *clock* counter. Frekuensi *clock* counter dari FPGA adalah 50Mhz sehingga dapat diperoleh periodenya 2 x 10^{-8} s. Maka cacahan pembanding dari *time high* atau *time low* untuk frekuensi 1 Khz adalah

$$C = \frac{T_s}{T_c}$$

$$C = \frac{5 \times 10^{-4}}{2 \times 10^{-8}}$$

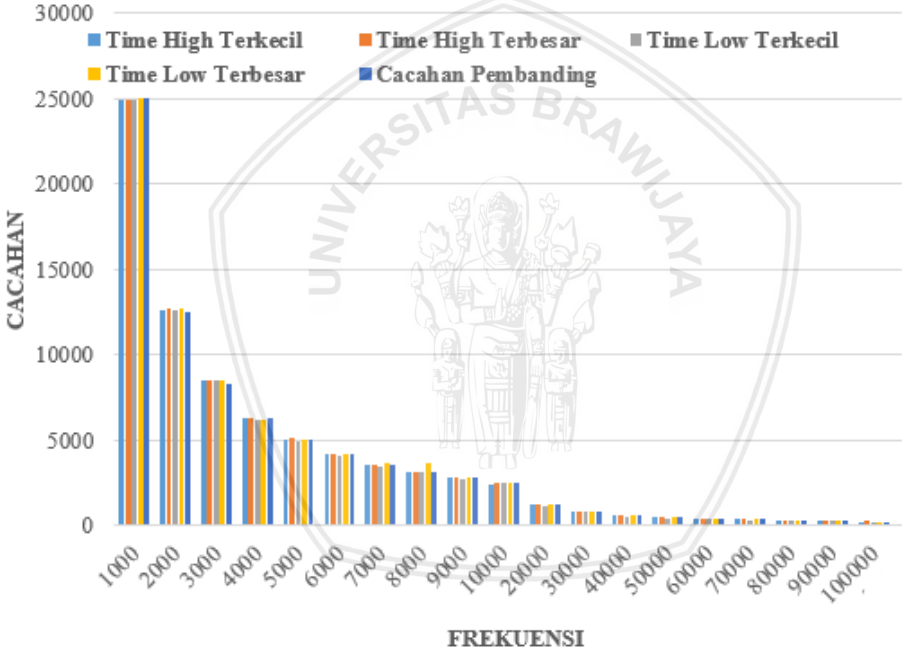
$$C = 25250$$

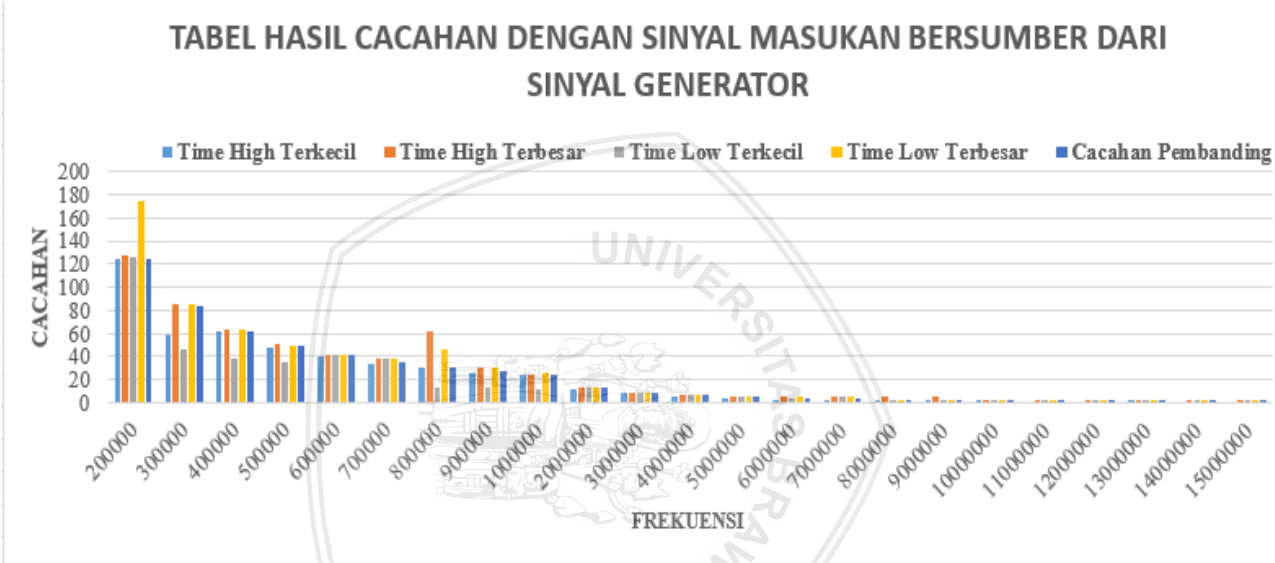
Dengan C adalah cacahan pembanding, T_s adalah periode *time high* atau *time low* dari real frekuensi sinyal digital, dan T_c adalah periode clock counter FPGA. Maka cacahan pembanding dari sinyal digital berfrekuensi 1 Khz adalah 25250 cacahan. Sinyal digital yang berasal dari sinyal generator maupun arduino uno masing-masing harus di cek dengan oscilloscope untuk mengetahui nilai cacahan pembandingnya. Sehingga untuk nilai cacahan pembanding sinyal generator dan arduino uno tidak lah sama (harus dicek dan dihitung masing-masing).

Dari grafik hasil yang diperoleh terlihat bahwa counter FPGA yang difungsikan sebagai pencacah *time high* dan *time low* dari sinyal digital dapat digunakan untuk menghitung lebar *time high* dan *time low* dari sinyal digital dengan baik. Hal tersebut dapat di buktikan dengan hasil cacahan FPGA yang mendekati nilai cacahan pembandingnya. Namun ada beberapa kelemahan yang terlihat dari hasil cacahan diatas. Saat FPGA mencacah sinyal dengan frekuensi tinggi yaitu 14 Mhz keatas, counter FPGA tidak dapat mencacah dengan baik. Hal tersebut terbukti pada hasil cacahan counter FPGA pada frekuensi 14 MHz dan 15 Mhz yang menghasilkan nilai cacahan time high terkecil sebesar nol (0) cacahan. Sementara pada perhitungannya untuk frekuensi 14 Mhz dan 15 Mhz memiliki nilai cacahan pembandingnya 1,6 dan 1,7 atau jika dibulatkan adalah 2 (untuk detail nilai cacahan dan cacahan pembandingnya dapat di lihat pada lampiran 7). Nilai

cacahan dari sinyal yang berfrekuensi 14 dan 15 MHz juga sama, hal ini membuktikan pula bahwa counter sudah berada pada batas cacahannya dimana untuk sinyal digital dengan frekuensi 14 Mhz keatas memiliki jumlah cacahan yang sama. Hal ini disebabkan karena tingginya frekuensi sinyal yang tidak dapat dijangkau oleh clock counter FPGA. Dengan demikian dapat dikatakan bahwa FPGA DE0-NANO dapat difungsikan sebagai pencacah *time high* dan *time low* sinyal digital hingga frekuensi 13Mhz.

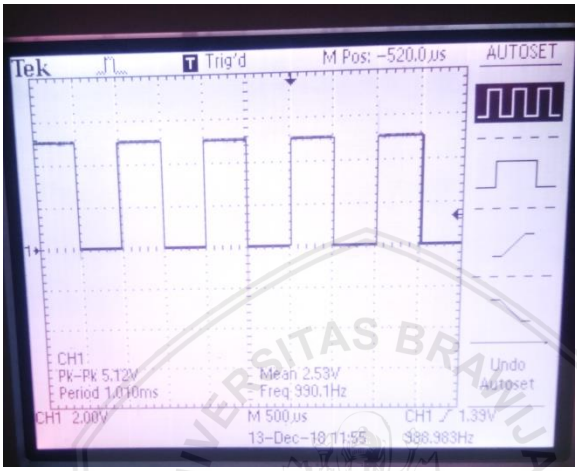
TABEL HASIL CACAHAN DENGAN SINYAL MASUKAN BERSUMBER DARI SINYAL GENERATOR



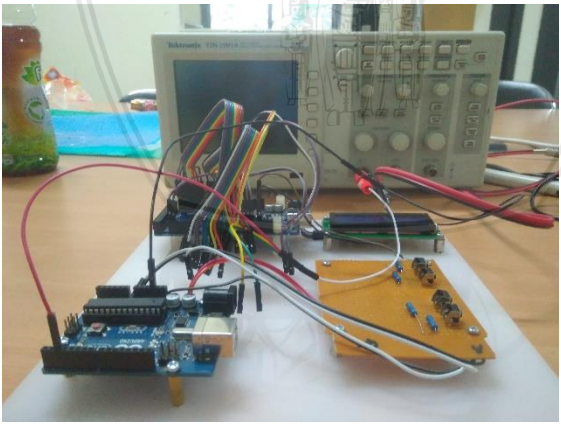


Gambar 4.5 Hasil Cacahan dengan Sumber Sinyal Berasal dari Sinyal Generator

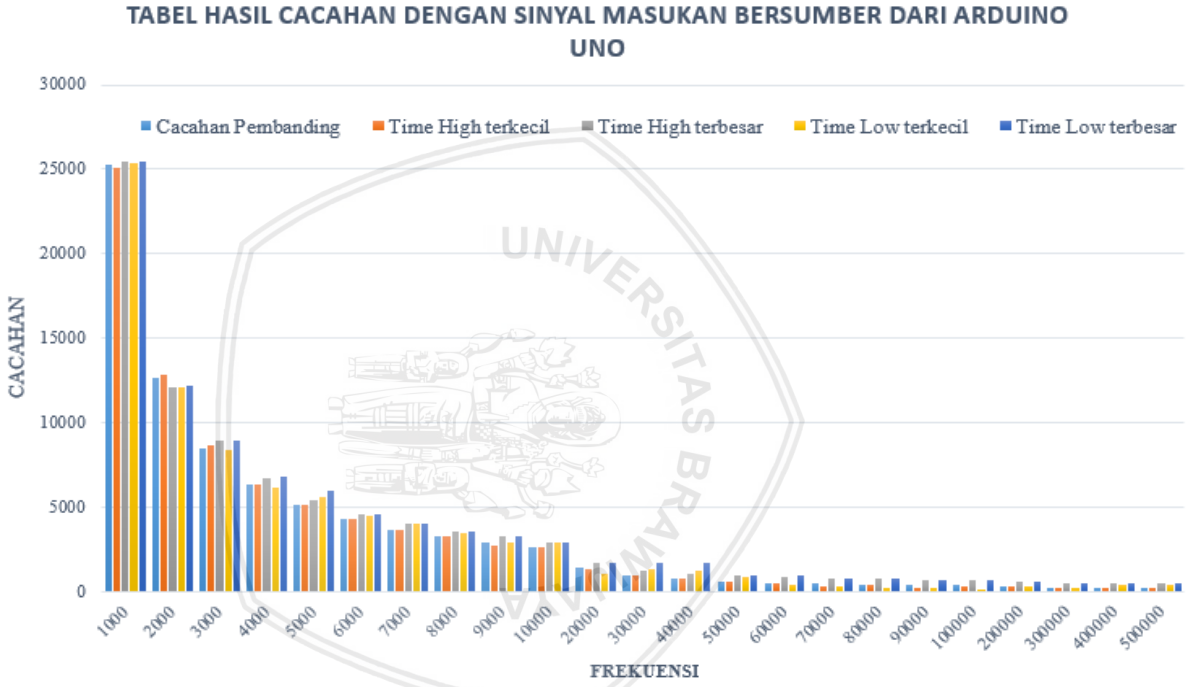
Sedangkan untuk sinyal masukan yang berasal dari arduino uno juga harus di cek dan dihitung cacahan pembandingnya untuk mengetahui kualitas dari counter FPGA yang telah di program.



Gambar 4.6 Tampilan Sinyal yang dihasilakan Arduino Uno pada Oscilloscope dengan frekuesi analisa 1Khz



Gambar 4.7 Pengujian Sinyal Keluaran Arduino Uno pada Oscilloscope



Gambar 4.8 Hasil Cacahan dengan Sumber Sinyal Berasal dari Arduino Uno

Dari Gambar 4.8 dapat dilihat bahwa hasil cacahan sinyal digital yang bersumber dari arduino uno oleh counter FPGA terhadap cacahan pembandingnya tidak memiliki perbedaan yang jauh (untuk nilai cacahannya lebih detail dapat di lihat pada lampiran 8). Hal tersebut juga membuktikan bahwa counter FPGA memiliki kelayakan dan kualitas yang baik untuk difungsikan sebagi pencacah *time high* dan *time low* sinyal digital sebagai analisa nilai jitter yang terjadi pada masing-masing sinyal digital yang ada.

1.5 Analisa Nilai Jitter

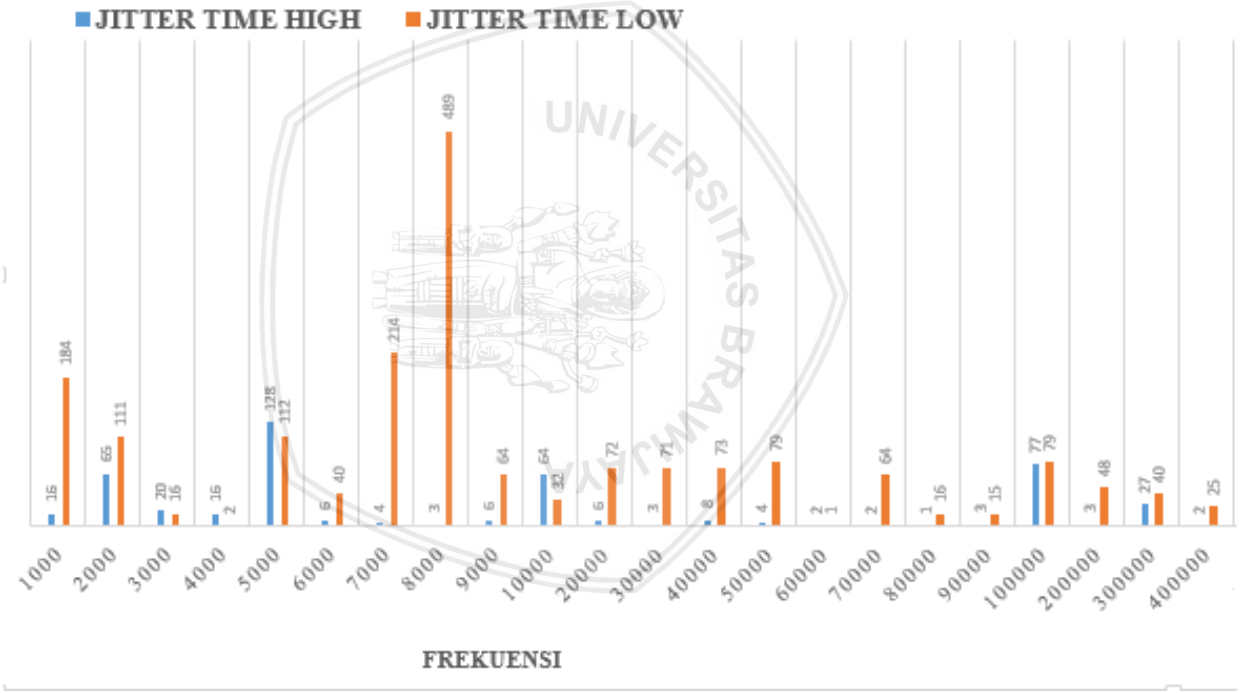
Hasil cacahan terbesar dan terkecil dari *time high* dan *time low* sinyal digital yang diperoleh dapat digunakan untuk analisa nilai jitter sinyal digital sesuai frekuensinya. Nilai dari jitter pada tiap frekuensinya dapat diperoleh dengan cara mengurangkan cacahan terbesar dengan cacahan terkecil sesuai jenis timenya.

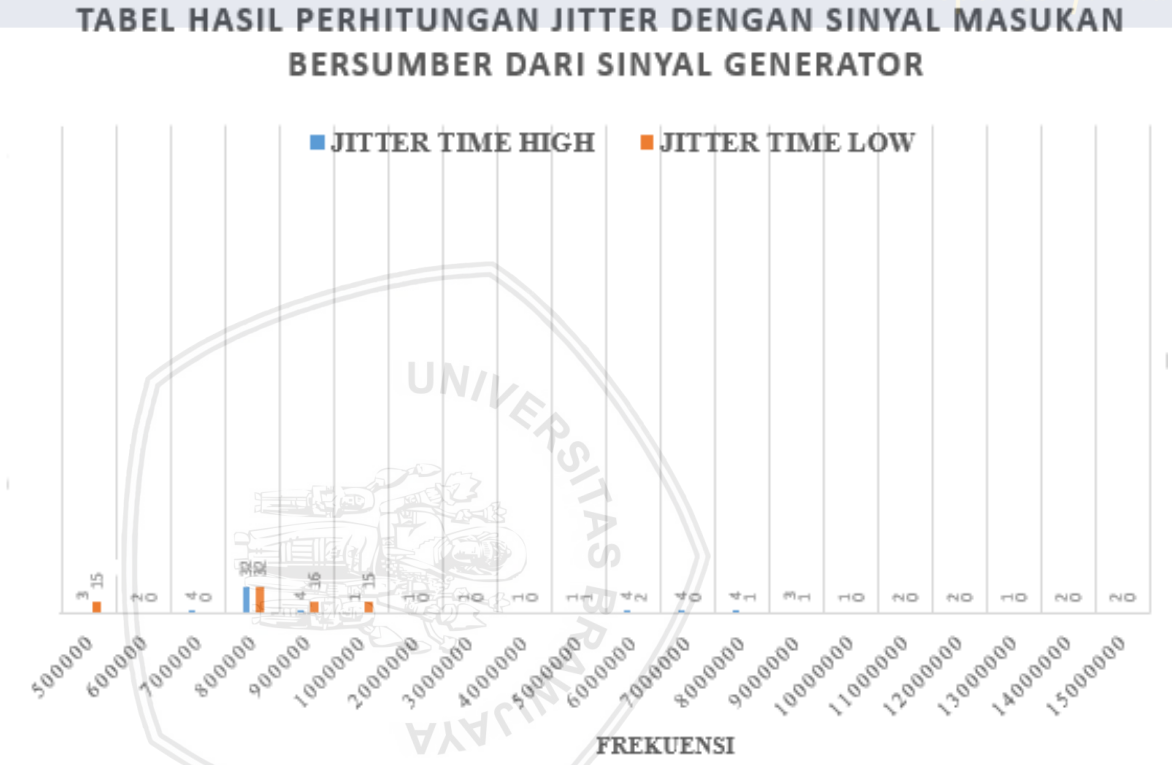
$$JH = nTHmax - nTHmin$$
$$JL = nTLmax - nTLmin$$

Dimana JH = Nilai jitter pada *time high*
JL = Nilai jitter pada *time low*
nTHmax = Nilai cacahan *time high* terbesar dari FPGA
nTHmin = Nilai cacahan *time high* terkecil dari FPGA
nTLmax = Nilai cacahan *time low* terbesar dari FPGA
nTLmin = Nilai cacahan *time low* terkecil dari FPGA

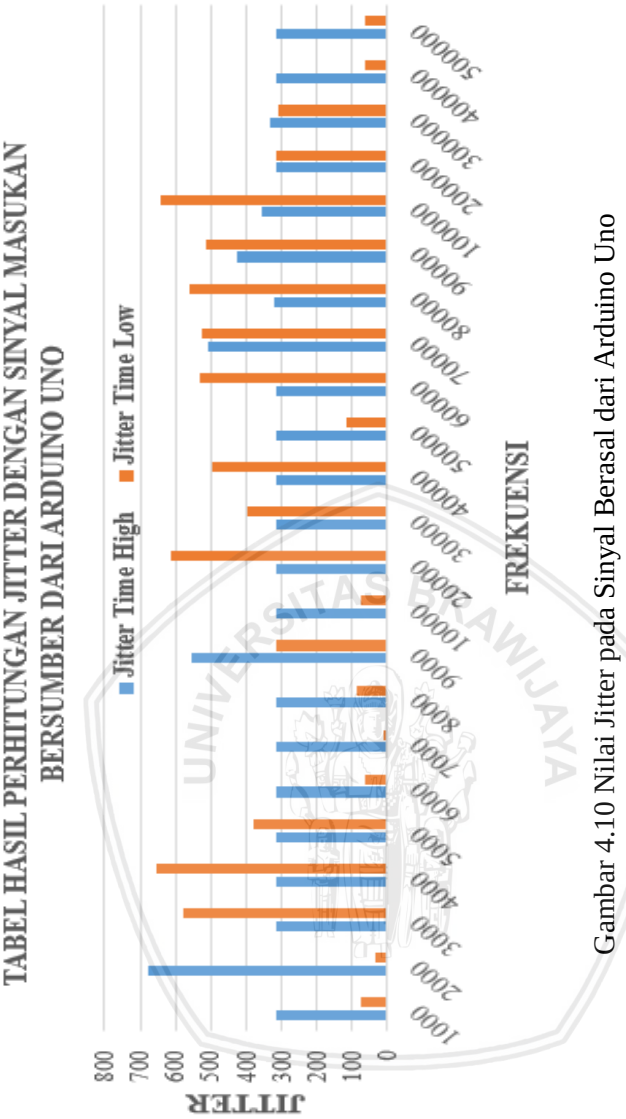
Sehingga diperoleh dua jenis jitter pada satu sinyal digital yaitu jitter pada *time high* dan jitter pada *time low*.

TABEL HASIL PERHITUNGAN JITTER DENGAN SINYAL MASUKAN BERSUMBER DARI SINYAL GENERATOR





Gambar 4.9 Nilai Jitter pada Sinyal Berasal dari Sinyal Generator



Gambar 4.10 Nilai Jitter pada Sinyal Berasal dari Arduino Uno

Dari kedua grafik hasil nilai jitter pada sinyal generator maupun Arduino uno diperoleh nilai jitter pada masing-masing frekuensinya. Dimana dari hasil diatas jitter yang ada pada sinyal masukan yang berasal dari sinyal generator cukup rendah. Sedangkan jitter yang ada pada sinyal digital yang bersumber dari Arduino uno cukup tinggi.

(Halaman ini sengaja dikosongkan)

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan diperoleh hasil bahwa sistem counter pada FPGA DE0-NANO dapat difungsikan sebagai pencacah *time high* dan *time low* dari sinyal digital hingga frekuensi 13Mhz. Kelayakan counter FPGA dalam mencacah sinyal digital dibuktikan dengan mendekatinya hasil cacahan oleh counter FPGA dengan nilai cacahan pembanding. Sehingga hasil dari cacahan counter FPGA dapat digunakan sebagai analisa nilai jitter pada sinyal digital.

5.2 Saran

Saran untuk penelitian selanjutnya adalah pengambilan data dengan sinyal masukan yang diproses memiliki range frekuensi rendah, misalnya sinyal dengan range frekuensi 1 Hz.

DAFTAR PUSTAKA

Abdurohman, Maman. 2013. *Perancangan Embedded Sistem Berbasis FPGA*. Graha Ilmu. Yogyakarta.

Intel. 2018. FPGAs and Programmable Devices.*Get Started Quickly with the $79 DE0-Nano Development Board*. Diakses Tanggal 28 Oktober 2018. https://www.intel.com/content/www/us/en/programmable/b/de0-nano-dev-board.html.

Mazor, Stanley dan Patricia Langstraat. 1992. *A Guide to VHDL*. Springer Science and Business Media. New York.

Mustofa, Ali. 2018. *Pengolahan Sinyal Digital*. UB Press. Malang.

Perry, Douglas L. 2002. *VHDL: Programming by Example*. McGraw-Hill Companies, Inc. New York.

Prasetio, Barlian Henryranu, Rizal Maulana, dan Dahnial Syauqy. 2017. *Desain Sistem Digital Menggunakan FPGA dan VHDL: Teori dan Aplikasi*. UB Press. Malang.

Tanudjaja, Harlianto. 2007. *Pengolahan Sinyal Digital & Sistem Pemrosesan Sinyal*. C.V Andi Offset. Yogyakarta.

Wibowo, Ferry Wahyu. 2015. *FPGA & VHDL Teori, Antarmuka dan Aplikasi*. Deepublish. Yogyakarta.

Wulandari, Rika. 2016. Analisis QoS (Quality of Service) Pada Jaringan Internet (Studi Kasus: UPT LOKA Uji Teknik Penambangan Jampang Kulon-LIPI). *Jurnal Teknik Informatika dan Sistem Informasi*. 2. 164.

Instruments, National. 2016. White Papers. *Digital Timing: Clock Signals, Jitter, Hystereisis, and Eye Diagram*. Diakses Tanggal 6-11-2018. http://www.ni.com/white-paper/3299/en/.

LAMPIRAN

1. PROGRAM ARDUINO MEGA 2560 (LCD)

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27
,2,1,0,4,5,6,7,3, POSITIVE);

long minlow=0;
long maxlow=0;
long minhi=0;
long maxhi=0;

void setup()
{
lcd.begin(16,2);

pinMode(9, INPUT);
pinMode(10, INPUT);
pinMode(11, INPUT);
pinMode(12, INPUT);

DDRF=B00000000;
DDRK=B00000000;
lcd.setCursor(0, 0);
lcd.print("STAND BY....");
}

void loop() {
long lsb1=0;
long msb1=0;
if (digitalRead(9)==LOW)
{
lsb1=PINF;
msb1=PINK;
```

```
long datanya=(msb1<<8) | lsb1;
maxlow =datanya;
lcd.setCursor(0, 0);
lcd.print("PULSA LOW MAKS ");
lcd.setCursor(0, 1);
lcd.print("      ");
lcd.setCursor(0, 1);
lcd.print(String(maxlow));
}
if (digitalRead(10)==LOW)
{
lsb1=PINF;
msb1=PINK;
long datanya2=(msb1<<8) | lsb1;
minlow =datanya2;
lcd.setCursor(0,0);
lcd.print("PULSA LOW MIN");
lcd.setCursor(0,1);
lcd.print("      ");
lcd.setCursor(0, 1);
lcd.print(String(minlow));
}
if (digitalRead(11)==LOW)
{
lsb1=PINF;
msb1=PINK;
long datanya2=(msb1<<8) | lsb1;
maxhi =datanya2;
lcd.setCursor(0, 0);
lcd.print("PULSA HIGH MAKS ");
lcd.setCursor(0, 1);
lcd.print("       ");
lcd.setCursor(0, 1);
lcd.print(String(maxhi));
}
if (digitalRead(12)==LOW)
```

```
{
lsb1=PINF;
msb1=PINK;
long datanya2=(msb1<<8) | lsb1;
minhi =datanya2;
lcd.setCursor(0, 0);
lcd.print("PULSA HIGH MIN  ");
lcd.setCursor(0, 1);
lcd.print("      ");
lcd.setCursor(0, 1);
lcd.print(String(minhi));
}
}
```

2. PROGRAM ARDUINO UNO (SINYAL MASUKAN)

```
void setup() {
pinMode(2,OUTPUT);
}
void loop() {
digitalWrite(2,HIGH);
delayMicroseconds(50);
delay(100);
digitalWrite(2,LOW);
delayMicroseconds(50);
delay(100);
}
```

3. PROGRAM FPGA DENGAN VHDL

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity cnt8bitvhdl is
Port (
CLK : in STD_LOGIC;
```

```
lMax : out STD_LOGIC;
lmin : out STD_LOGIC;
HMax : out STD_LOGIC;
Hmin : out STD_LOGIC;
t1 : in STD_LOGIC;
t2 : in STD_LOGIC;
t3 : in STD_LOGIC;
t4 : in STD_LOGIC;
RST : in  STD_LOGIC;
CE  : in  STD_LOGIC;
oFA  : out STD_LOGIC_VECTOR(15 downto 0)
);
end cnt8bitvhdl;

architecture c8bitku of cnt8bitvhdl is

signal counter : STD_LOGIC_VECTOR(15
downto 0) := (others => '0');
signal counterh : STD_LOGIC_VECTOR(15
downto 0) := (others => '0');
signal data1 : STD_LOGIC_VECTOR(15
downto 0) := (others => '0');
signal data2 : STD_LOGIC_VECTOR(15
downto 0) := (others => '0');

signal lowmax : STD_LOGIC_VECTOR(15
downto 0) := (others => '0');
signal lowmin : STD_LOGIC_VECTOR(15
downto 0) := "1111111111111111";
signal highmax : STD_LOGIC_VECTOR(15
downto 0) := (others => '0');
signal highmin : STD_LOGIC_VECTOR(15
downto 0) := "1111111111111111";

signal CLK1 :  STD_LOGIC;
signal CLK2 :  STD_LOGIC;
signal CE2 :  STD_LOGIC;
signal RST2 :  STD_LOGIC;
```

```
signal RST1 :  STD_LOGIC;

begin
CLK1<=CLK;
CLK2<=CLK;
CE2<=CE;
RST1<=RST;
RST2<=RST;

count_process: process(CLK1,CE,RST1)
begin
if RST1='0' then
counter <= "0000000000000000";
lowmax <= "0000000000000000";
lowmin <= "1111111111111111";
else
if CE='0' then
if falling_edge(CLK1) then
counter <= counter + 1;
end if;
end if;
if rising_edge(CE) then
data1<=counter;
if lowmax<data1 then lowmax <=data1 ;
end if;
if lowmin>data1 then lowmin <=data1 ;
end if;
end if;

if CE='1' then
counter <= "0000000000000000";
end if;
end if;
end process;

count_process2: process(CLK2,CE2,RST2)
begin
if RST2='0' then
```

```
counterh <= "0000000000000000";
highmax <= "0000000000000000";
highmin <= "1111111111111111";
else
if CE2='1' then
if  rising_edge(CLK2) then
counterh <= counterh + 1;
end if;
end if;
if falling_edge(CE2) then
data2<=counterh;
if highmax<data2 then highmax <=data2 ;
end if;
if highmin>data2 then highmin <=data2 ;
end if;
end if;
if CE2='0' then
counterh <= "0000000000000000";
end if;
end if;
end process;
display: process (t1,t2)
begin
if t1='0' Then oFA<=highmax; hmax<='0';
hmin<='1'; lmax<='1'; lmin<='1';
end if;
if t2='0' Then oFA<=highmin; hmax<='1';
hmin<='0'; lmax<='1'; lmin<='1';
end if;
if t3='0' Then oFA<=lowmax; hmax<='1';
hmin<='1';  lmax<='0'; lmin<='1';
end if;
if t4='0' Then oFA<=lowmin; hmax<='1';
hmin<='1'; lmax<='1'; lmin<='0';
end if;
end process;
end c8bitku;
```

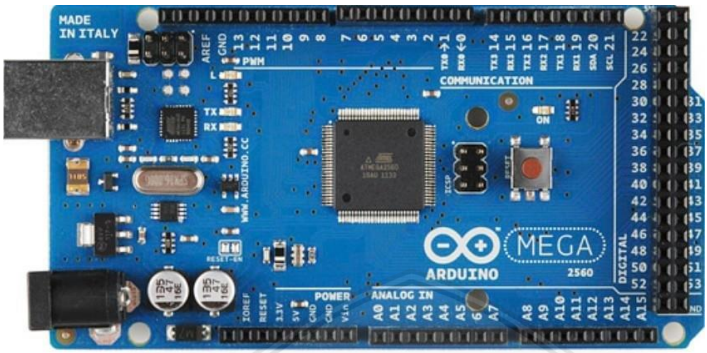
4. GPIO FPGA DE0-NANO

GPIO_00	GPIO_01	GPIO_03	GPIO_05	GPIO_07	GND	GPIO_09	GPIO_011	GPIO_013	GPIO_015	GPIO_017	GPIO_019	GPIO_021	GPIO_023	GND	GPIO_025	GPIO_027	GPIO_029	GPIO_031	GPIO_033
D3	C3	A3	B4	B5		D5	A6	D6	C6	E6	D8	F8	E9		D9	E10	B11	D11	B12
2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40
1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37	39
A8	B8	A2	B3	A4		A5	B6	B7	A7	C8	E7	E8	F9		C9	E11	C11	A12	D12
GPIO_0_IN0	GPIO_0_IN1	GPIO_02	GPIO_04	GPIO_06	VCC_5V	GPIO_08	GPIO_010	GPIO_012	GPIO_014	GPIO_016	GPIO_018	GPIO_020	GPIO_022	VCC_3V3	GPIO_024	GPIO_026	GPIO_028	GPIO_030	GPIO_032

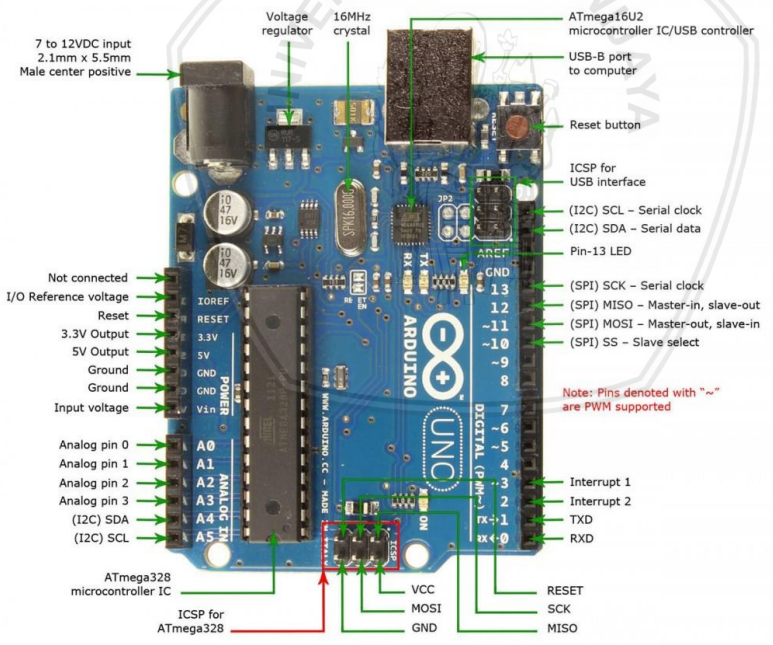
Pin 1

GPIO_132	GPIO_130	GPIO_128	GPIO_126	GPIO_124	VCC_3V3	GPIO_122	GPIO_120	GPIO_118	GPIO_116	GPIO_114	GPIO_112	GPIO_110	GPIO_18	VCC_5V	GPIO_16	GPIO_14	GPIO_12	GPIO_1_IN1	GPIO_1_IN0
J13	J16	M10	L14	N15		R14	P15	R16	L16	N9	N12	P11	T10		R12	R13	T14	R9	T9
39	37	35	33	31	29	27	25	23	21	19	17	15	13	11	9	7	5	3	1
40	38	36	34	32	30	28	26	24	22	20	18	16	14	12	10	8	6	4	2
J14	K15	L13	N14	P14		N16	P16	L15	K16	N11	P9	R10	R11		T11	T12	T13	T15	F13
GPIO_133	GPIO_131	GPIO_129	GPIO_127	GPIO_125	GND	GPIO_123	GPIO_121	GPIO_119	GPIO_117	GPIO_115	GPIO_113	GPIO_111	GPIO_19	GND	GPIO_17	GPIO_15	GPIO_13	GPIO_11	GPIO_10

5. ARDUINO MEGA 2560



6. ARDUINO UNO



7. TABEL HASIL CACAHAN DENGAN SINYAL MASUKAN BERSUMBER DARI SINYAL GENERATOR

Frekuensi (Hz)	Real Frekuensi (Hz)	TIME HIGH		TIME LOW		Cacahan Pembanding			
						TIME HIGH		TIME LOW	
		terkecil	terbesar	terkecil	terbesar	terkecil	terbesar	terkecil	terbesar
1000	1000	24944	24960	24870	25054	25000			
2000	2000	12616	12681	12623	12734	12500			
3000	3012	8520	8540	8526	8542	8300.1328			
4000	4000	6248	6264	6188	6190	6250			
5000	5000	4984	5112	4910	5022	5000			
6000	6023	4160	4166	4134	4174	4150.7554			
7000	7000	3578	3582	3502	3716	3571.4286			
8000	8000	3120	3123	3132	3621	3125			
9000	9000	2784	2790	2724	2788	2777.7778			
10000	10000	2440	2504	2478	2510	2500			
20000	20000	1248	1254	1190	1262	1250			
30000	29950	832	835	807	878	834.72454			
40000	40000	624	632	554	627	625			
50000	50000	498	502	422	501	500			
60000	60000	416	418	418	417	416.66667			
70000	70000	356	358	294	358	357.14286			
80000	79900	312	313	297	313	312.89111			
90000	90000	276	279	263	278	277.77778			
100000	100000	174	251	171	250	250			

200000	200000	124	127	126	174	125
300000	299900	59	86	46	86	83.36112
400000	400000	62	64	38	63	62.5
500000	500000	48	51	35	50	50
600000	598000	40	42	42	42	41.80602
700000	700000	34	38	38	38	35.714286
800000	801000	30	62	14	46	31.210986
900000	899000	26	30	14	30	27.808676
1000000	1000000	24	25	11	26	25
2000000	1900000	12	13	13	13	13.157895
3000000	3000000	8	9	9	9	8.3333333
4000000	3900000	6	7	7	7	6.4102564
5000000	5000000	4	5	5	6	5
6000000	5900000	2	6	4	6	4.2372881
7000000	6900000	2	6	6	6	3.6231884
8000000	8032000	2	6	2	3	3.1125498
9000000	8955000	2	5	2	3	2.7917365
10000000	10020000	2	3	3	3	2.49501
11000000	10990000	1	3	3	3	2.2747953
12000000	12020000	1	3	3	3	2.0798669
13000000	12990000	2	3	2	2	1.9245574
14000000	14080000	0	2	2	2	1.7755682
15000000	14960000	0	2	2	2	1.671123

8. TABEL HASIL CACAHAN DENGAN SINYAL MASUKAN BERSUMBER DARI ARDUINO UNO

Frek (Hz)	real frek (hz)	Cacahan Pembanding				Time High		Time Low	
		Time High		Time Low					
		Terkecil	terbesar	Terkecil	terbesar	terkecil	terbesar	terkecil	terbesar
1000	990.1	25249.975				25124	25438	25382	25454
2000	1984	12600.806				12800	12121	12126	12159
3000	2959	8448.800				8654	8968	8396	8973
4000	3937	6350.013				6360	6674	6158	6814
5000	4892	5110.384				5108	5423	5557	5933
6000	5866	4261.848				4258	4573	4515	4578
7000	6831	3659.786				3658	3974	3970	3978
8000	7669	3259.877				3258	3574	3490	3578
9000	8591	2910.022				2668	3222	2914	3230
10000	9580	2609.603				2608	2921	2855	2927
20000	18270	1368.363				1356	1671	1061	1676
30000	25880	965.997				956	1270	1276	1670
40000	32700	764.526				756	1070	1204	1702
50000	40750	613.497				606	920	814	926
60000	48750	512.821				506	820	426	954
70000	54050	462.535				262	770	258	782
80000	60650	412.201				406	728	166	726
90000	60650	412.201				245	670	174	686
100000	69090	361.847				315	670	46	686
200000	95740	261.124				256	570	262	575
300000	118600	210.793				189	520	212	520
400000	144480	173.034				168	482	430	494
500000	144480	173.034				168	482	430	494

9. TABEL HASIL PERHITUNGAN JITTER DENGAN SINYAL MASUKAN BERSUMBER DARI SINYAL GENERATOR

Frekuensi (Hz)	TIME HIGH		JITTER TIME HIGH	TIME LOW		JITTER TIME LOW
	terkecil	terbesar		terkecil	terbesar	
1000	24944	24960	16	24870	25054	184
2000	12616	12681	65	12623	12734	111
3000	8520	8540	20	8526	8542	16
4000	6248	6264	16	6188	6190	2
5000	4984	5112	128	4910	5022	112
6000	4160	4166	6	4134	4174	40
7000	3578	3582	4	3502	3716	214
8000	3120	3123	3	3132	3621	489
9000	2784	2790	6	2724	2788	64
10000	2440	2504	64	2478	2510	32
20000	1248	1254	6	1190	1262	72
30000	832	835	3	807	878	71
40000	624	632	8	554	627	73
50000	498	502	4	422	501	79
60000	416	418	2	417	418	1
70000	356	358	2	294	358	64
80000	312	313	1	297	313	16
90000	276	279	3	263	278	15
100000	174	251	77	171	250	79
200000	124	127	3	126	174	48
300000	59	86	27	46	86	40
400000	62	64	2	38	63	25

500000	48	51	3	35	50	15
600000	40	42	2	42	42	0
700000	34	38	4	38	38	0
800000	30	62	32	14	46	32
900000	26	30	4	14	30	16
1000000	24	25	1	11	26	15
2000000	12	13	1	13	13	0
3000000	8	9	1	9	9	0
4000000	6	7	1	7	7	0
5000000	4	5	1	5	6	1
6000000	2	6	4	4	6	2
7000000	2	6	4	6	6	0
8000000	2	6	4	2	3	1
9000000	2	5	3	2	3	1
10000000	2	3	1	3	3	0
11000000	1	3	2	3	3	0
12000000	1	3	2	3	3	0
13000000	2	3	1	2	2	0
14000000	0	2	2	2	2	0
15000000	0	2	2	2	2	0

10. TABEL HASIL PERHITUNGAN JITTER DENGAN SINYAL MASUKAN BERSUMBER DARI ARDUINO UNO

Frek (Hz)	Time High		Jitter Time High	Time Low		Jitter Time Low
	terkecil	terbesar		terkecil	terbesar	
1000	25124	25438	314	25382	25454	72
2000	12121	12800	679	12126	12159	33
3000	8654	8968	314	8396	8973	577
4000	6360	6674	314	6158	6814	656
5000	5108	5423	315	5557	5933	376
6000	4258	4573	315	4515	4578	63
7000	3658	3974	316	3970	3978	8
8000	3258	3574	316	3490	3578	88
9000	2668	3222	554	2914	3230	316
10000	2608	2921	313	2855	2927	72
20000	1356	1671	315	1061	1676	615
30000	956	1270	314	1276	1670	394
40000	756	1070	314	1204	1702	498
50000	606	920	314	814	926	112
60000	506	820	314	426	954	528
70000	262	770	508	258	782	524
80000	406	728	322	166	726	560
90000	245	670	425	174	686	512
100000	315	670	355	46	686	640
200000	256	570	314	262	575	313
300000	189	520	331	212	520	308
400000	168	482	314	430	494	64