UNIVERSITAS INDONESIA

RANCANG BANGUN DAN ANALISIS SISTEM PERINGATAN POLUSI UDARA PADA AREA PARKIR TERTUTUP MENGGUNAKAN FPGA XILINX SPARTAN 3E

SKRIPSI

VICKY DWI KURNIAWAN
0806459910

FAKULTAS TEKNIK UNIVERSITAS INDONESIA
DEPARTEMEN TEKNIK ELEKTRO
TEKNIK KOMPUTER
DEPOK
JUNI
2012

UNIVERSITAS INDONESIA

RANCANG BANGUN DAN ANALISIS SISTEM PERINGATAN POLUSI UDARA PADA AREA PARKIR TERTUTUP MENGGUNAKAN FPGA XILINX SPARTAN 3E

SKRIPSI

Diajukan sebagai salah satu syarat memperoleh gelar sarjana.

VICKY DWI KURNIAWAN
0806459910

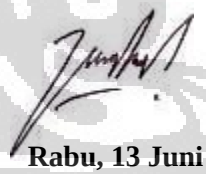
FAKULTAS TEKNIK UNIVERSITAS INDONESIA
DEPARTEMEN TEKNIK ELEKTRO
TEKNIK KOMPUTER
DEPOK
JUNI
2012

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri,

dan semua sumber baik yang dikutip maupun dirujuk

telah saya nyatakan dengan benar.

Nama	**:**	**Vicky Dwi Kurniawan**
NPM	**:**	**0806459910**
Tanda Tangan	**:**	
Tanggal	**:**	**Rabu, 13 Juni 2012**

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh

Nama	:	Vicky Dwi Kurniawan
NPM	:	0806459910
Program Studi	:	Teknik Komputer
Judul Skripsi	:	Rancang Bangun dan Analisis Sistem Peringatan Polusi Udara pada Area Parkir Tertutup Menggunakan FPGA Xilinx Spartan 3E

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang dilakukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Komputer, Fakultas Teknik, Universitas Indonesia.

DEWAN PENGUJI

Pembimbing : Prima Dewi Purnamasari S.T., M.Sc. (.....)

Penguji : Ir. A. Endang Sriningsih M.T., Si. (.....)

Penguji : Dr. Ir. Anak Agung Putri Ratna M.Eng. (.....)

Ditetapkan di : Depok
Tanggal : 22 Juni 2012

KATA PENGANTAR

Puji syukur saya panjatkan kehadirat Allah SWT, karena atas segala berkat dan rahmat-Nya, saya dapat menyelesaikan skripsi yang berjudul "Rancang Bangun dan Analisis Sistem Peringatan Polusi Udara pada Area Parkir Tertutup Menggunakan FPGA Xilinx Spartan 3E". Penulisan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Teknik Jurusan Teknik Komputer pada Fakultas Teknik Universitas Indonesia. Saya menyadari bahwa, tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada penyusunan skripsi ini, sangatlah sulit bagi saya untuk menyelesaikan skripsi ini. Oleh karena itu, saya mengucapkan terima kasih kepadar:

1. Prima Dewi Purnamasari S.T., M.Sc., selaku pembimbing telah menyediakan waktu, tenaga, dan pikiran untuk mengarahkan saya dalam penyusunan skripsi ini;
2. Orang tua dan keluarga yang telah memberikan bantuan dukungan moril dan materil serta do'a;
3. Rr. Wulan Apriliyanti, Evan G. Sumbayak, dan teman–teman dari Teknik Elektro angkatan 2008, yang tiada hentinya mendukung saya baik secara langsung maupun tidak langsung;
4. Rekan–rekan asisten dari Laboratorium Digital Departemen Teknik Elektro; dan
5. Sophia Firtiesa, SP. Yang telah banyak membantu saya dalam menyelesaikan skripsi ini.

Akhir kata, saya berharap Tuhan Yang Maha Esa berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu.

Depok, 13 Juni 2012

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama : Vicky Dwi Kurniawan
NPM : 0806459910
Program Studi : Teknik Komputer
Departemen : Teknik Elektro
Fakultas : Teknik
Jenis Karya : Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Non Ekslusif (Non-exclusive Royalty Free Right)** Atas karya ilmiah saya yang berjudul:

"Rancang Bangun dan Analisis Sistem Peringatan Polusi Udara pada Area Parkir Tertutup Menggunakan FPGA Xilinx Spartan 3E"

Dengan Hak Bebas Royalti Nonekskulif ini Universitas Indonesia Berhak menyimpan, mengalihmediakan/format-kan, mengelola dalam bentuk pangkalan data (database), merawat, dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.
Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : 13 Juni 2012
Yang menyatakan

(Vicky Dwi Kurniawan)

ABSTRAK

Nama : Vicky Dwi Kurniawan
Program Studi : Teknik Komputer
Judul : Rancang Bangun dan Analisis Sistem Peringatan Polusi Udara pada Area Parkir Tertutup Menggunakan FPGA Xilinx Spartan 3E

Area parkir tertutup merupakan salah satu tempat terjadinya polusi udara tertutup akibat emisi gas buang kendaraan bermotor yang tidak dapat keluar dari ruangan tertutup. Polusi udara tertutup jauh lebih berbahaya dibandingkan dengan polusi udara terbuka. Skripsi ini merancang, membuat prototipe, serta menganalisis sistem peringatan polusi udara menggunakan *Field Programmable Gate Array* (FPGA) Xilinx Spartan 3E. Peralatan yang digunakan sebagai sistem peringatan dalam prototipe berupa LED, *buzzer*, dan *fan*. Metode yang digunakan dalam sistem *embedded* ini mengikuti *Software Development Life Cycle* (SDLC). Bahasa yang digunakan adalah VHDL dengan software Xilinx ISE. Berdasarkan hasil uji coba, didapatkan hasil bahwa *timing diagram* antara simulasi *Register Transfer Level* (RTL) dan implementasi tidak jauh berbeda dengan selisih waktu 0.37%, sehingga untuk melihat *output* dan *response time* keseluruhan sistem dapat melalui simulasi RTL. Waktu yang dibutuhkan sistem untuk mengeluarkan CO lebih lama 60-71% dari perhitungan dikarenakan terdapat jeda waktu pembacaan kadar CO oleh sensor. Diperlukan sebanyak 1024 sampel data ADC pada FPGA Spartan 3E agar hasil pembacaan sensor stabil.

Kata kunci :
Area parkir tertutup, FPGA Xilinx Spartan 3E, sistem peringatan.

ABSTRACT

Name : Vicky Dwi Kurniawan
Study Program : Computer Engineering
Title : Development and Analysis of Air Pollution Warning System at Closed Parking Area Using FPGA Xilinx Spartan 3E

Closed parking area can deposit motor gas emission that could be harmful to humans. Indoor air pollution is more dangerous than the outdoor one. This thesis discusses the design, prototype making, and analyzes the embedded air pollution warning system using Field Programmable Gate Array (FPGA) Xilinx Spartan 3E. Other equipments use in this system are LED, buzzer, and fan. The method used in this research follows the Software Development Life Cycle (SDLC). The programming language used in configuring the FPGA Xilinx Spartan 3E is VDHL using Xilinx ISE Design Suite 13.2. Based on result, Register Transfer Level (RTL) simulation and implementation timing diagram does not have much different with a difference 0,37% so to see the output and overall system response time can be through RTL simulation. Time required to remove carbon monoxide from the dummy box is 60-71% longer than the calculation because there is a lag time of the reading levels of CO by the sensor. 1024 data ADC samples are needed in order to give a stable result from FPGA Spartan 3E

Key words:
Closed parking area, FPGA Xilinx Spartan 3E, warning system

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR SINGKATAN

FPGA	Field Programmable Gate Array
RTL	Register Transfer Level
SPI	Serial Peripheral Interface
PWM	Pulse Width Modulation
ADC	Analog to Digital Converter
UCF	User Constraints File
PCB	Printed Circuit Board
RAM	Random Access Memory
IOB	Input Output Block
CLB	Configurable Logic Block
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuits
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
FET	Field Effect Transistor
UML	Unified Modeling Language
CO	Carbon Monoksida
DSO	Digital Storage Oscilloscope

BAB I
PENDAHULUAN

1.1 Latar Belakang

Semakin meningkatnya jumlah penduduk menyebabkan kebutuhan akan lahan untuk parkir makin meningkat pula. Hal ini mendorong berkembangnya sistem perparkiran tertutup terutama di kota-kota besar. Banyak gedung-gedung bertingkat yang kini menggunakan sistem parkir di bawah tanah atau *basement* karena sistem perparkiran ini dapat mengoptimalkan penggunaan lahan yang terbatas untuk dijadikan area parkir. Jenis lahan parkir ini mempunyai kelemahan karena biasanya kurang memperhatikan aspek-aspek kesehatan lingkungan bagi para pengguna dan pekerja yang menggunakan area parkir tersebut. Polusi udara merupakan masalah utama yang harus dihadapi dalam sistem perparkiran ini.

Polusi udara *indoor* diakibatkan oleh sirkulasi udara yang buruk di tempat parkir tertutup sehingga menyebabkan tidak lancarnya pertukaran udara di dalam area parkir dengan udara bebas di luar. Jika hal ini dibiarkan, polusi udara akan semakin parah dan bahkan bisa membahayakan kesehatan orang yang menghirup udara tersebut. Polusi udara *indoor* lebih berbahaya karena relatif stabilnya polutan yang berada di dalamnya dan tidak adanya sinar matahari untuk mempercepat reaksi kimia atau mengencerkan konsentrasi polutan. Jenis polutan yang mungkin ada meliputi CO, SO_2, O3, NO_X, dan PM. Diantara jenis polutan tersebut, karbon monoksida (CO) adalah yang harus paling diwaspadai. Karbon monoksida (CO) adalah gas yang sangat beracun, tidak berwarna, tidak berbau, dan tidak berasa. Oleh karena itu CO dapat menyebabkan sakit kepala pada dosis rendah dan dapat mematikan pada dosis yang tinggi.

Buruknya efek yang ditimbulkan dari polusi udara akibat sistem perparkiran tertutup ini mendorong perlu adanya suatu sistem perparkiran tertutup yang dilengkapi dengan sistem pemantau polusi udara guna mengontrol dan mengatur polusi udara di ruangan, serta rambu peringatan akan bahaya berdiam terlalu lama di ruang parkir tertutup.

Berkembangnya teknologi komponen elektronika tidak lepas dari kebutuhan umat manusia yang terus bertambah untuk menyelesaikan suatu masalah. Teknologi *Field Programmable Gate Array* (FPGA) merupakan salah satu teknologi Integrated Circuit Very Large Scale Integration (IC VLSI) yang dapat diprogram sesuai kebutuhan.

Sistem peringatan kualitas udara berbasis FPGA merupakan tujuan yang ingin dicapai dari penelitian pada skripsi ini. Diharapkan aplikasi ini dapat menjadi awal dari kepedulian masyarakat dalam mengatasi masalah polusi udara *indoor* yang sangat berbahaya dibandingkan polusi udara *outdoor*.

1.2 Tujuan Penelitian

Tujuan dari penelitian yang dilakukan adalah sebagai berikut:

1. Merancang dan membuat prototipe sistem embedded peringatan polusi pada area parkir tertutup menggunakan FOGA
2. Menganalisis *response time* yang dibutuhkan *Analog to Digital Convereter* (ADC) dalam mengumpulkan data dari sensor yang disimpan pada *Random Access Memory* (RAM) sampai data siap dikeluarkan dana *response time* secara keseluruhan.
3. Menganalisis waktu yang dibutuhkan fan untuk mengeluarkan udara CO pada *dummy box* pada saat kecepatan perputaran kipas 50% (level 1) dan 100% (level 2).

1.3 Batasan Penelitian

FPGA yang digunakan dalam penelitian adalah Xilinx Spartan 3E. Alat pemberi peringatan yang digunakan terbatas pada LED, buzzer, dan fan. Pengujian prototipe sistem dilakukan menggunakan dummy box berukuran 82 liter.

1.4 Metodologi Penulisan

Metodologi penelitian yang digunakan dalam pembuatan sistem ini mengacu pada *System Development Life Cycle* (SDLC)[4], antara lain:

1. Inisiasi ide
2. Menentukan tema yang akan dikembangkan berkaitan dengan isu masyarakat saat ini.
3. *System requirement*
4. Mendefinisikan fitur-fitur yang diperlukan dalam sistem yang akan dibuat.
5. Desain
6. Merubah kebutuhan-kebutuhan yang sudah dirangkum pada tahapan sebelumnya sehingga menjadi suatu rancangan.
7. Pengembangan dan implementasi
8. Merubah desain menjadi bentuk sistem utuh.
9. Uji coba

1.5 Sistematika Penulisan

Agar penulisan skripsi ini dapat terarah dengan baik, maka penulisan skripsi ini dibagi atas beberapa bab, yaitu:

BAB I PENDAHULUAN

Bab ini menjelaskan tentang latar belakang, tujuan, metodologi penelitian, batasan masalah, dan sistematika penulisan

BAB II PENCEMARAN UDARA, FPGA XILINX SPARTAN 3E, DAN ALAT INPUT OUTPUT

Bab ini menjelaskan tentang definisi udara, pencemaran udara, *Fields Programmable Gate Arrays* beserta komponennya, dan alat *input output* yang digunakan sebagai sistem peringatan.

BAB III RANCANG BANGUN SISTEM PERINGATAN POLUSI UDARA BERBASIS FPGA

Bab ini menjelaskan tentang *user requirement*, desain sistem yang meliputi tahapan perancangan implementasi dan rencana uji coba berdasarkan metode penelitian yang telah dipaparkan sebelumnya.

BAB IV PENGAMATAN DAN ANALISIS SISTEM PERINGATAN POLUSI UDARA BERBASIS FPGA

Bab ini menjelaskan tentang pengujian dan analisis sistem peringatan polusi udara berbasis FPGA.

BAB V KESIMPULAN

Bab ini memaparkan kesimpulan yang diambil dari pembahasan sistem peringatan polusi udara berabasis FPGA.

BAB II
PENCEMARAN UDARA, FPGA XILINX SPARTAN 3E, DAN ALAT INPUT OUTPUT

2.1 Definisi Udara

Menurut kementrian lingkungan hidup, udara merupakan kumpulan atau campuran gas, yang terbanyak adalah nitrogen dan oksigen. Rata-rata persentase (per volume) gas dalam udara bersih dan kering mengandung 78% nitrogen, 20.8% oksigen, 0.9% argon, 0.03% karbon dioksida, dan 0.27% gas-gas lain[1].

2.2 Pencemaran Udara

Pencemaran udara atau sering disebut dengan istilah polusi udara diartikan sebagai adanya bahan-bahan atau zat-zat asing di dalam udara yang menyebabkan perubahan susunan atau komposisi udara dari keadaan normalnya[2].

Tabel 2.1 Perkiraan persentase komponen pencemar udara dari sumber pencemar transportasi di Indonesia[2]

Komponen Pencemaran	Persentase
CO	70,50%
NOx	8,89%
SOx	0,88%
HC	18,34%
Partikel	1,33%
Total	100%

Seperti pada Tabel 2.1 karbon monoksida (CO) merupakan komponen pencemaran udara paling tinggi. Oleh karena itu karbon monoksida dipilih sebagai sampel yang akan dipantau dan digunakan dalam sistem peringatan polusi udara CO di area parkir tertutup berbasis FPGA.

2.2.1 Karbon Monoksida

Gas karbon monoksida disebut juga gas CO adalah gas yang tidak berwarna, tidak berbau, dan tidak berasa[3]. Karbon monoksida merupakan hasil pembakaran yang tidak sempurna dari kendaraan bermotor, yang sangat berbahaya apabila dikomsumsi oleh tubuh secara berlebihan. Efek dari karbon monoksida dapat dilihat pada Tabel 2.2.

Tabel 2.2 Efek dari karbon monoksida[3]

Ppm CO	Lama Paparan	Gejala
35	8 jam	Taraf yang masih diperbolehkan di dalam lingkungan kerja dalam kurun waktu 8 jam kerja per hari.
200	2 – 3 jam	Sakit kepala ringan, rasa lelah, rasa mual, dan disorientasi.
400	1 – 2 jam	Sakit kepala berat, ancaman kematian setelah 3 jam.
800	45 menit	Pusing hebat, mual, kejang. Kemungkinan tidak sadarkan diri selama 2 jam. Kematian dalam 2-3 jam.
1600	20 menit	Sakit kepala hebat, mual, mengakibatkan kematian dalam 1 jam.
3200	5 – 10 menit	Sakit kepala hebat, mual, mengakibatkan kematian dalam 1 jam.
6400	1 – 2 menit	Sakit kepala hebat, mual, mengakibatkan kematian dalam 25-30 menit.
12.800	1 – 3 menit	Kematian.

2.3 Field Programmable Gate Array

FPGA (*Fields Programmable Gate Arrays*) merupakan salah satu teknologi IC VLSI, dimana FPGA sendiri adalah komponen elektronika dan semikonduktor yang mempunyai *progammable logic* dan *programmable interconnects* yang dapat dikonfigurasi oleh pengguna atau desainer yang

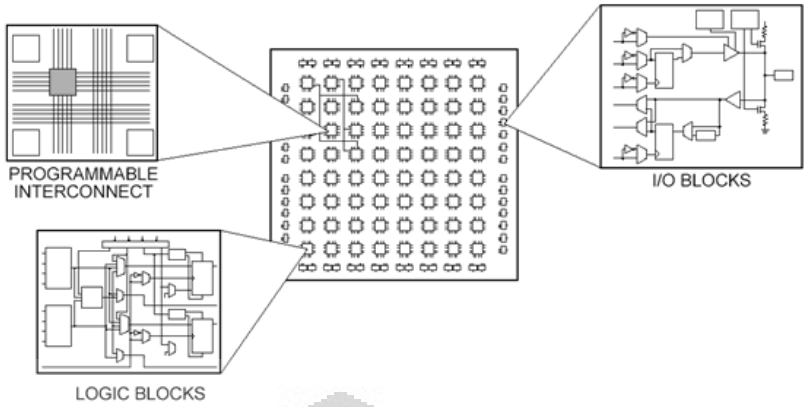
dihubungkan melalui kanal-kanal *routing*. FPGA berisi komponen *programmable logic* yang biasa disebut blok logika. Komponen *programmable logic* yang dimiliki oleh FGPA adalah gerbang logika dasar (OR, AND, NOT, XOR) maupun jenis fungsi matematis dan gerbang kombinasi yang lebih kompleks (*decoder*, *adder*, *subtractor*, dan lain-lain). Selain itu blok komponen FPGA juga bisa berisi element memori (*register*) dari *flip-flop* sampai dengan RAM[4].

FPGA merupakan IC tipe HDL (*Hardware Description Lenguage*) yang dimana pengguna dapat mendesain hardware dalam IC FPGA sesuai kebutuhan. FPGA diprogram dengan menggunakan *circuit diagram logic* atau *source code* untuk menentukan bagaimana cara kerja chip FPGA tersebut. Setiap chip FPGA terdiri dari puluhan hingga puluhan ribu sel logika. Secara umum, setiap sel logika menggabungkan beberapa *input* binary menjadi satu atau dua keluaran sesuai dengan fungsi logika yang ditetapkan.

Secara umum FPGA akan lebih lambat jika dibandingkan dengan jenis chip yang lain karena FPGA membutuhkan supply yang besar. FPGA bersifat *volatile*, yang berarti ketika sumber daya yang mensuplainya tesebut dicabut maka secara otomatis FPGA akan kehilangan fungsinya, jadi FPGA tidak mampu menyimpan program ketika *supply* tenaganya dicabut.

2.3.1 Arsitektur FPGA

Secara umum arsitektur bagian dalam dari IC FPGA yang dapat dilihat pada Gambar 2.1 terdiri atas tiga bagian yaitu *Input Output Block* (IOB), *Configurable Logic Block* (CLB), dan Interkoneksi (kanal-kanal *routing*)[5]. Sumber daya internal IC FPGA terdiri dari dua bagian diatas yaitu matriks CLB yang dikelilingi oleh pinggiran IOB.

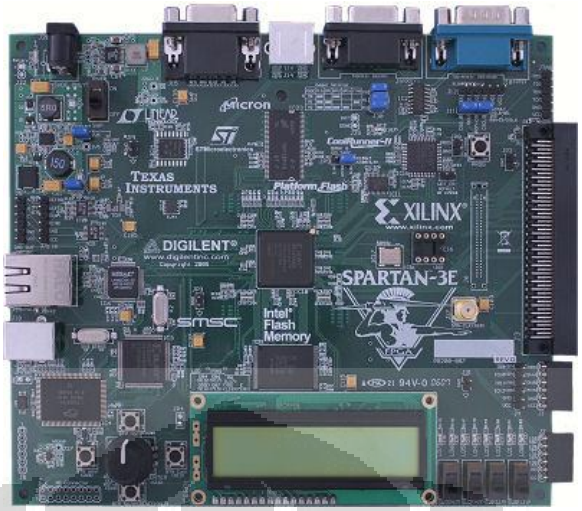


Gambar 2.1 Arsitektur FPGA [5]

- *Configuration Logic block*
  - Berfungsi untuk mengimplementasikan fungsi-fungsi logika yang akan dibuat.
- *Programmable Interconnect*
  - Berisi *wire segments* dan *programmable switches*
  - Menghubungkan antara CLB yang berbeda
- *Input/Output Block*
  - Sebagai *interface* antara *external package pin* dari *device* dan *internal user logic.*

2.3.2 FPGA SPARTAN 3E

Spartan 3E merupakan salah satu dari keluarga FPGA yang diproduksi oleh Xilinx yang bisa dilihat pada Gambar 2.2. Bahasa pemrograman yang digunakan untuk memprogram FPGA ini adalah VHDL (*Very high Hardware Description Language*) dengan bantuan *software development kit* dari Xilinx.

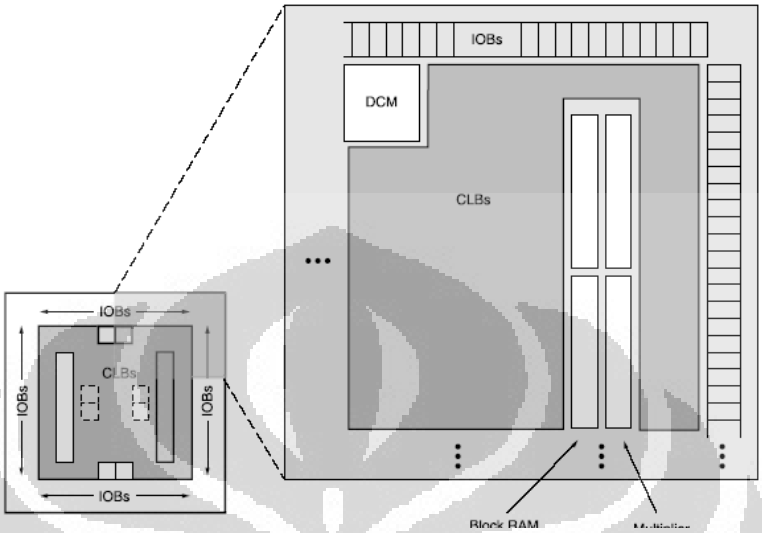


Gambar 2.2 FPGA Spartan 3E [6]

Secara umum arsitektur bagian dalam dari IC FPGA terdiri atas tiga bagian yaitu *Input Output Block* (IOB), *Configurable Logic Block* (CLB), dan Interkoneksi (kanal-kanal *routing*). Sedangkan arsitektur bagian dalam dari IC FPGA Spartan 3E dapat dilihat pada Gambar 2.3.

- *Configurable Logic Block*s (CLBs) memiliki *Look-Up Tables* (LUTs) yang fleksibel dimana dapat mengimplementasikan logika ditambah tempat penyimpanan yang digunakan sebagai flip-flop atau latches. CLBs melakukan berbagai fungsi logika dan juga penyimpan data.
- *Input/Output Blocks* (IOBs) mengontrol aliran data antara I/O pin dan logika internal. Setiap IOB mendukung aliran data dua arah ditambah operasi 3-state. Selain itu IOB juga mendukung berbagai standar sinyal, termasuk empat *high performance differential standards*.
- *Block* RAM menyediakan penyimpanan data dalam bentuk dual-port blok 18-Kbit.
- *Block Multiplier* menerima dua angka biner 18-bit sebagai *input* dan menghitungnya.

- *Digital Clock Manager* menyediakan kalibrasi, *delay*, *multiplying*, *dividing*, dan fase pergesaran sinyal clock.



Gambar 2.3 Arsitektur IC FPGA Spartan 3E[6]

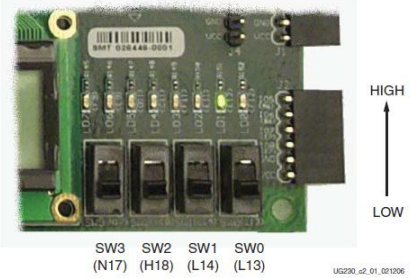
Semua bagian dari arsitektur ini terorganisir dengan baik seperti yang ditunjukkan pada Gambar 2.3. Sebuah cincin IOBs mengelilingi array CLBs. Setiap perangkat memiliki dua blok kolom RAM. Setiap kolom RAM terdiri dari beberapa blok 18-Kbit RAM. Setiap blok RAM dikaitkan dengan multiplier. DCMS diposisikan di tengah dengan dua di atas dan dua di bagian bawah perangkat[6].

2.3.3 Komponen Board Starter Kit Spartan-3E

Board yang digunakan dalam penelitian ini adalah Xilinx FPGA Spartan-3E (XC3S500E-4FG320C) dimana memiliki 232-pin I/O, 320-pin paket FBGA dan lebih dari 10.000 sel logika.

2.3.3.1 Slide Switches

Slide switches pada board FPGA Spartan 3E dapat dilihat pada Gambar 2.4. Konstrain slide switches didefinisikan pada file UCF meliputi koneksi pin I/O, resistor pull-down, dan tegangan TTL low-voltage yang dapat dilihat pada Gambar 2.5.



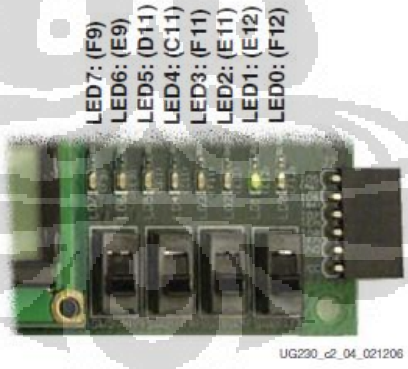
Gambar 2.4 4 *slide switch*[6]

```
NET "SW<0>" LOC = "L13" | IOSTANDARD = LVTTL | PULLUP ;
NET "SW<1>" LOC = "L14" | IOSTANDARD = LVTTL | PULLUP ;
NET "SW<2>" LOC = "H18" | IOSTANDARD = LVTTL | PULLUP ;
NET "SW<3>" LOC = "N17" | IOSTANDARD = LVTTL | PULLUP ;
```

Gambar 2.5 UCF *slide switch* [6]

2.3.3.2 LED

LED pada board FPGA Spartan 3E dapat dilihat pada Gambar 2.6. Konstrain 6-pin accessory header didefinisikan di file UCF meliputi koneksi I/O, slew-rate, dan arus keluaran yang dapat dilihat pada Gambar 2.7.



Gambar 2.6 8 LED [6]

```
NET "LED<7>" LOC = "F9" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<6>" LOC = "E9" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<5>" LOC = "D11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<4>" LOC = "C11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<3>" LOC = "F11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<2>" LOC = "E11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<1>" LOC = "E12" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<0>" LOC = "F12" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
```

Gambar 2.7 UCF LED [6]

2.3.3.3 Clock Sources

Spartan-3E Starter Kit board memiliki tiga sumber daya clock yang dapat dilihat pada Gambar 2.8, dimana setiap *input* clock terhubung dengan Digital Clock Manager.

1. 50 MHz *On-Board Oscillator*

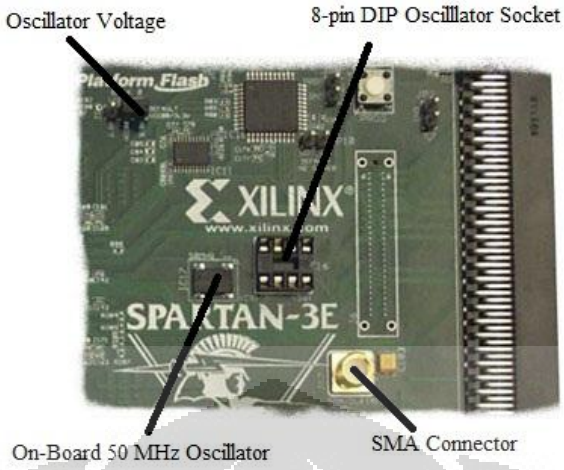
   Board ini memilki 50 MHz osilator dengan 40% - 60% *duty cycle*. Osilator ini mampu bekerja secara akurat sampai dengan ±2500 Hz atau ±50 ppm.

2. *Auxiliary Clock Oscillator Socket*

   Soket 8-pin ini digunakan untuk osilator clock tambahan yang memiliki 8-pin DIP footprint. Soket ini digunakan apabila membutuhkan frekuensi lain dari 50 MHz.

3. SMA *Clock Input or Output Connector*

   Digunakan untuk memberikan clock dari sumber eksternal, dimana sinyal *input* clock dihubungkan ke konektor SMA.



Gambar 2.8 Clock sources [6]

Untuk menggunakan salah satu atau semua sumber daya clock, maka diharuskan untuk mengkonfigurasi lokasi konstrain UCF yang dapat dilihat pada Gambar 2.9

```
NET "CLK_50MHZ" LOC = "C9" | IOSTANDARD = LVCMOS33 ;
NET "CLK_SMA" LOC = "A10" | IOSTANDARD = LVCMOS33 ;
NET "CLK_AUX" LOC = "B8" | IOSTANDARD = LVCMOS33 ;
```

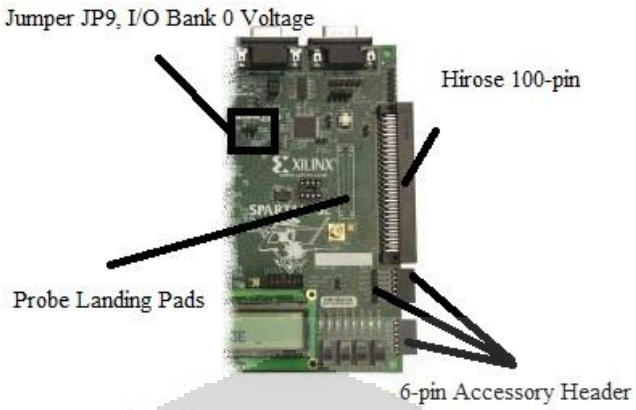
Gambar 2.9 UCF clock sources [6]

2.3.3.4 Konektor Ekspansi

Spartan-3E Starter Kit board menyediakan berbagai macam konektor ekpansi untuk fleksibilitas interface yang mudah untuk komponen tambahan dari luar board, dapat dilihat pada Gambar 2.10.

Secara default konektor ekspansi ini memberikan supply daya sebesar 3.3V dimana dapat dirubah menjadi 2.5V dengan mengubah jumper yang ada di port JP9. Konektor ekspansi yang ada pada board terdiri dari tiga jenis, yaitu:

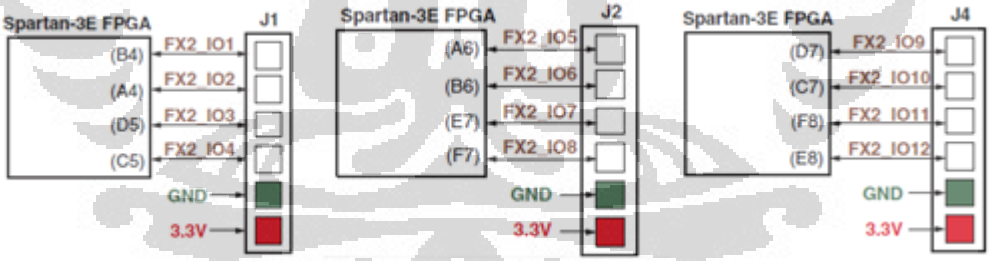
1. *Hirose 100-pin edge*
2. 3 buah 6-pin peripheral modul connections
3. *Probe landing pads*



Gambar 2.10 Konektor ekspansi [6]

Pada penilitian ini digunakan 6-pin *peripheral modul connections* untuk dihubungkan dengan sensor gas CO (MQ7) dan *relay board*.

6-Pin *Accessory Header* yang dapat dilihat pada Gambar 2.11 terdiri dari J1, J2, dan J4 header, dimana J1 dan J2 header adalah female 6-pin soket dan J4 adalah male 6-pin soket. 4-pin paling atas sebagain I/O dan 2-pin terakhir sebagai *ground* dan *power supply* sebesar 3.3V.



Gambar 2.11 Koneksi ke Accessory Header[6]

Konstrain 6-pin *accessory header* didefinisikan di file UCF meliputi koneksi I/O, slew-rate, dan arus keluaran yang dapat dilihat pada Gambar 2.12.

```
# ==== 6-pin header J1 ====
#NET "J1<0>" LOC = "B4" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J1<1>" LOC = "A4" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J1<2>" LOC = "D5" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J1<3>" LOC = "C5" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;
# ==== 6-pin header J2 ====
#NET "J2<0>" LOC = "A6" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J2<1>" LOC = "B6" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J2<2>" LOC = "E7" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J2<3>" LOC = "F7" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;
# ==== 6-pin header J4 ====
#NET "J4<0>" LOC = "D7" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J4<1>" LOC = "C7" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J4<2>" LOC = "F8" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J4<3>" LOC = "E8" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;
```

Gambar 2.12 UCF 6-pin accessory header [6]

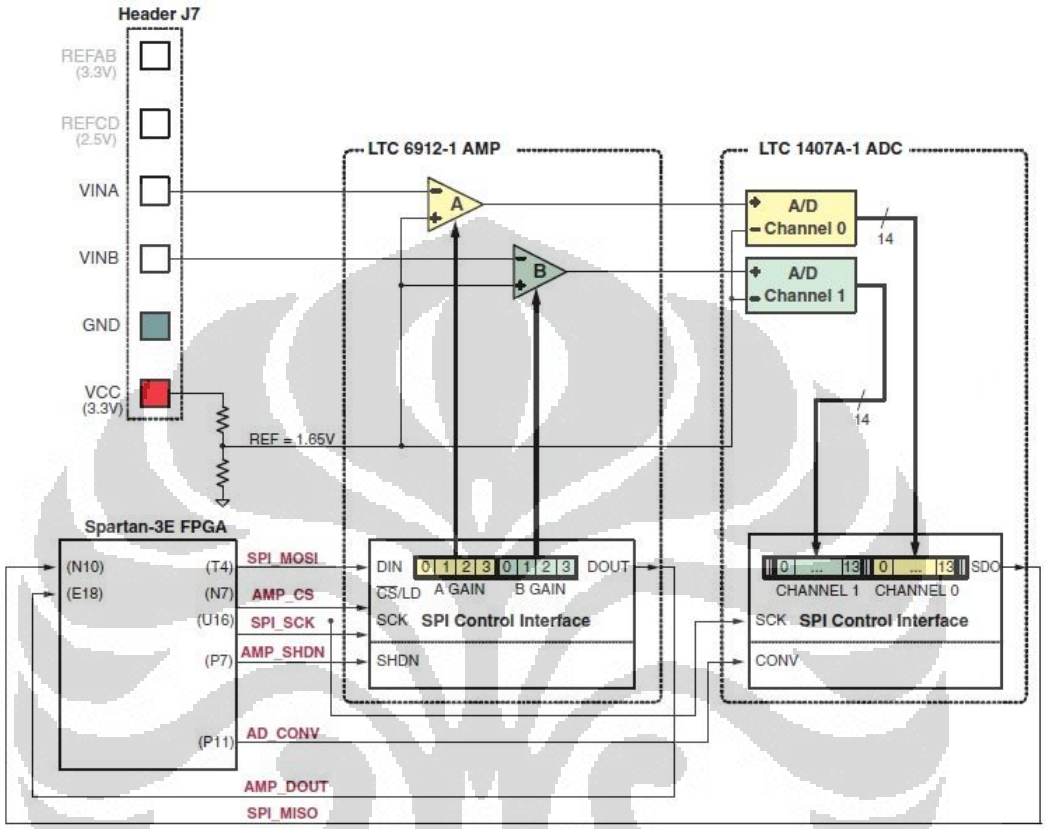
2.3.3.5 Analog Capture Circuit

Spartan 3E Starter Board mempunyai dua channel penangkap sinyal analog. Dimana dalam proses penangkapan sinyal analog ini FPGA akan melakukan *pre-amplifier* dan ADC pada sinyal analog tersebut sampai mendapatkan data dalam bentuk digital. Gambar 2.13 menggambarkan proses yang dilakukan oleh FPGA pada saat menangkap sinyal analog.

Pada saat sinyal masuk ke FPGA melalui channel 1 (VinA) atau 2 (VinB) maka sinyal akan melalui *pre-amplifier* yang akan memberikan penguatan negatif dengan mensetting gain sebesar 8 bit untuk channel 1 dan 2. Tabel 2.3 menunjukkan besar gain yang dapat dimasukkan ke *pre-amplifier* pada FPGA. Setelah itu sinyal akan diterjemahkan oleh ADC yang akan menjadi bentuk digital.

Untuk memprogram *pre-amplifier* dan ADC pada FPGA maka harus menggunakan SPI, dimana SPI atau Serial Peripheral Interface adalah salah satu cara untuk mentransfer data secara serial. Pada saat memprogram *pre-amplifier*, yang harus diperhatikan adalah pembuatan SPI timming yang dapat dilihat pada Gambar 2.14, setelah itu sinyal akan menuju ADC. Untuk menggunakan ADC juga dibutuhkan SPI. Pada Gambar 2.15 merupakan gambaran bagaimana ADC

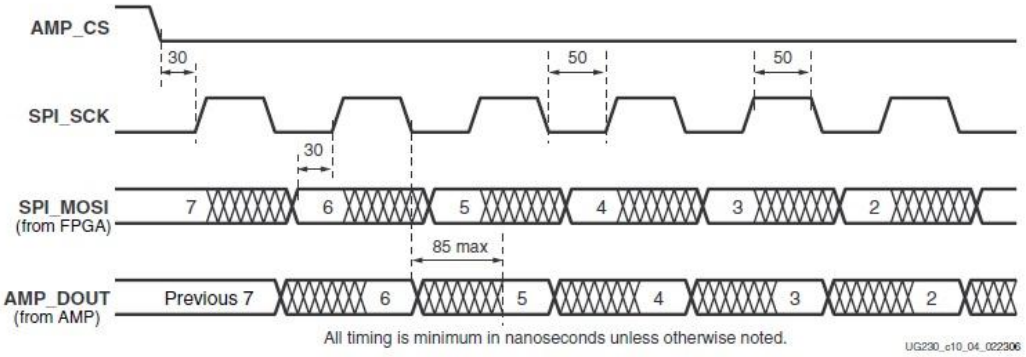
pada FPGA merubah sinyal analog menjadi sinyal digital. ADC akan melakukan sampel pada tahap pengambilan data pertama dan data akan keluar pada tahap kedua. Keluaran dari ADC berupa 14 bit 2's *complement*.



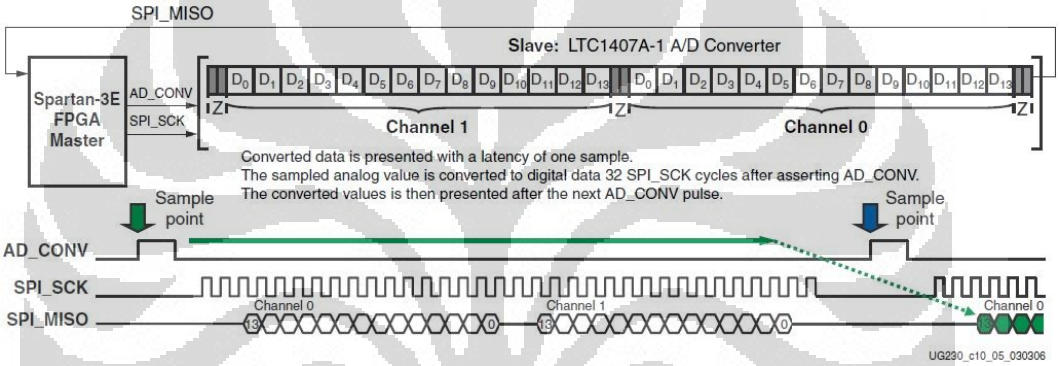
Gambar 2.13 Detail *Analog Capture Circuit* [6]

Tabel 2.3 Pengaturan Gain untk *Pre-Amplifier* [6]

Gain	A3	A2	A1	A0	Input Voltage Range	
	B3	B2	B1	B0	Minimum	Maksimum
0	0	0	0	0		
-1	0	0	0	1	0,4	2,9
-2	0	0	1	0	1,025	2,275
-5	0	0	1	1	1,4	1,9
-10	0	1	0	0	1,525	1,775
-20	0	1	0	1	1,5875	1,7125
-50	0	1	1	0	1,625	1,675
-100	0	1	1	1	1,6375	1,6625



Gambar 2.14 SPI Timming untuk berkomunikasi dengan Pre-Amplifier [6]



Gambar 2.15 Analog to Digital Conversion Interface [6]

2.3.3.6 High Description Language (HDL)

HDL (*Hardware Description Language*) adalah bahasa dari kelas bahasa komputer dan atau bahasa pemrograman yang berfungsi untuk mendeskripsikan sirkuit elektronik dan khususnya *digital logic*. HDL diperuntukan bagi bahasa pemrograman apa saja yang didesain untuk menyusun atau memodelkan dan mengimplementasikan blok-blok hardware. Karena bahasa HDL digunakan untuk memodelkan fungsionalitas dari sebuah sirkuit, bahasa-bahasa HDL memiliki kerangka dasar dari bahasa software yaitu fungsi for, *while*, *if* dan *case*[7].

2.3.3.7 VHDL

VHDL (VHSIC *Hardware Description Language*) adalah bahasa dari kelas bahasa komputer yang berfungsi untuk mendeskripsikan hardware

elektronika digital. VHDL merupakan standar IEEE/ANSI dan Departemen Pertahanan US. Pertama kali dipublikasikan oleh IEEE pada tahun 1987, dengan label IEEE Std 1076-1987. Bahasa ini telah mengalami modifikasi dan revisi, dengan versi terbaru berlabel IEEE Std 1076-1993[7].

Desain digital yang akan digunakan dalam VHDL akan digambarkan dengan menggunakan external view dengan satu atau beberapa internal view. External view merupakan interface dari rancangan sedangkan internal view menyatakan fungsi atau struktur dari rancangan yang akan dibuat, dimana satu rancangan mungkin memiliki satu atau lebih internal view. Selain itu desain digital dapat juga digambarkan dalam VHDL dengan menggunakan beberapa external view yang berbeda, dimana tiap external view dan salah satu darinya berkaitan dengan internal view, yang secara bersama-sama merupakan representasi tertentu dari device, disebut dengan entity design. Entitiy juga bisa diartikan sebagai tempat untuk menentukan variabel *input* dan *output*[7].

VHDL memungkinkan seseorang untuk menggambarkan sistem digital di struktural atau tingkat perilaku. Tingkat perilaku dapat dibagi lagi menjadi dua jenis gaya yaitu data flow dan algoritmik. Data flow merupakan representasi yang menggambarkan bagaimana data bergerak melalui sistem. Ini biasanya dilakukan dalam hal aliran data antara register (*register transfer leve*l). Model aliran data penggunaan bersamaan membuat pernyataan yang dijalankan secara paralel data segera setelah tiba dimasukan. Di sisi lain, pernyataan sekuensial dijalankan dalam urutan yang mereka ditetapkan. VHDL memungkinkan baik bersamaan dan sinyal berurutan tugas yang akan menentukan cara dimana mereka dieksekusi[7].

2.4 CO Gas Sensor (MQ-7)

CO gas sensor MQ-7 merupakan sensor buatan Hanwei China yang digunakan untuk mendeteksi gas CO yang merupakan hasil pembakaran tidak sempurna dari kendaraan bermotor. Sensor MQ-7 yang bisa dilihat pada Gambar 2.16 ini terbuat dari tabung mikro keramik AL_2O_3, lapisan tipis SnO_2, elektroda dan *heater* yang digabungkan dalam suatu lapisan kerak yang terbuat dari plastik dan *stainless steel*.

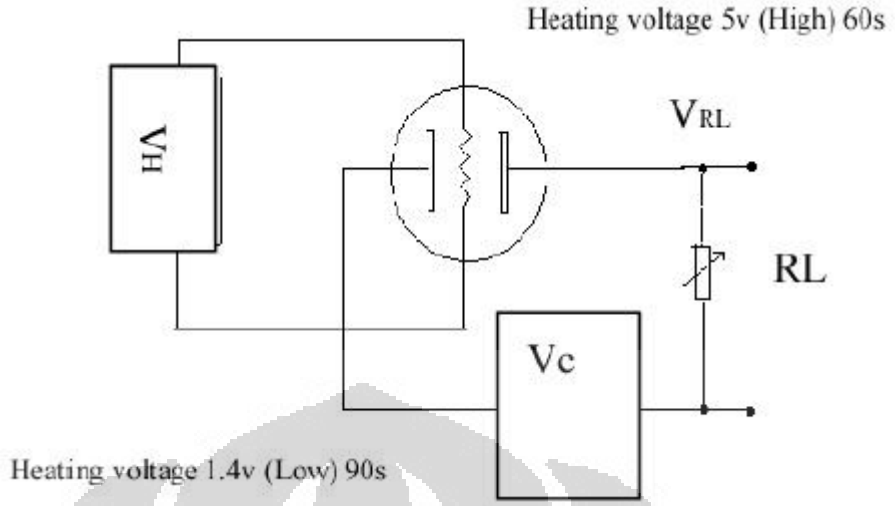
Gambar 2.16 CO Gas Sensor (MQ-7)[8]

Apabila sensor MQ-7 mendeteksi gas CO yang ada di udara, maka tegangan *output* pada sensor akan meningkat, sehingga konsentrasi gas akan menurun dan terjadi proses deoksidasi. Hal ini mengakibatkan penurunan resistansi sensor yang juga memiliki sebuah heater, yang berfungsi sebagai pembersih dari kontaminasi udara di dalam jangkauan sensor.

Output sensor memiliki karakteristik berupa tegangan *output* yang akan semakin besar sesuai dengan besarnya kadar ppm pada saat mendeteksi keberadaan gas CO. Pengukuran kadar ppm asap rokok diperoleh dari perbandingan antara resistansi sensor pada saat terdapat gas (Rs) dengan resistansi sensor pada saat udara bersih (Ro). Berikut rumus untuk mencari nilai ppm:

$$\frac{Rs}{Ro} = \frac{Vc - Vout}{Vout} \; x \; RL \qquad (2.1)$$

Rangkaian tambahan pada sensor MQ-7 ini menggunakan nilai RL sebesar 10KΩ, bisa dilihat pada Gambar 2.17. Dari persamaan 2.1, semakin banyak gas CO maka resistansi semakin menurun dan nilai Vout semakin membesar. Vc digunakan tegangan DC sebesar 5volt.



Gambar 2.17 Rangkaian sensor MQ-7[8]

Sensor gas CO MQ-7 memiliki dua buah masukan tegangan yaitu 5volt (high voltage) dan 1.4volt (low voltage), dimana dengan menggunakan dua masukan tegangan pada sensor gas CO MQ-7 menyebabkan sensor berosilasi selama beberapa selang waktu, yaitu 60 detik pada saat 5volt dan 90 detik pada saat 1.4volt.

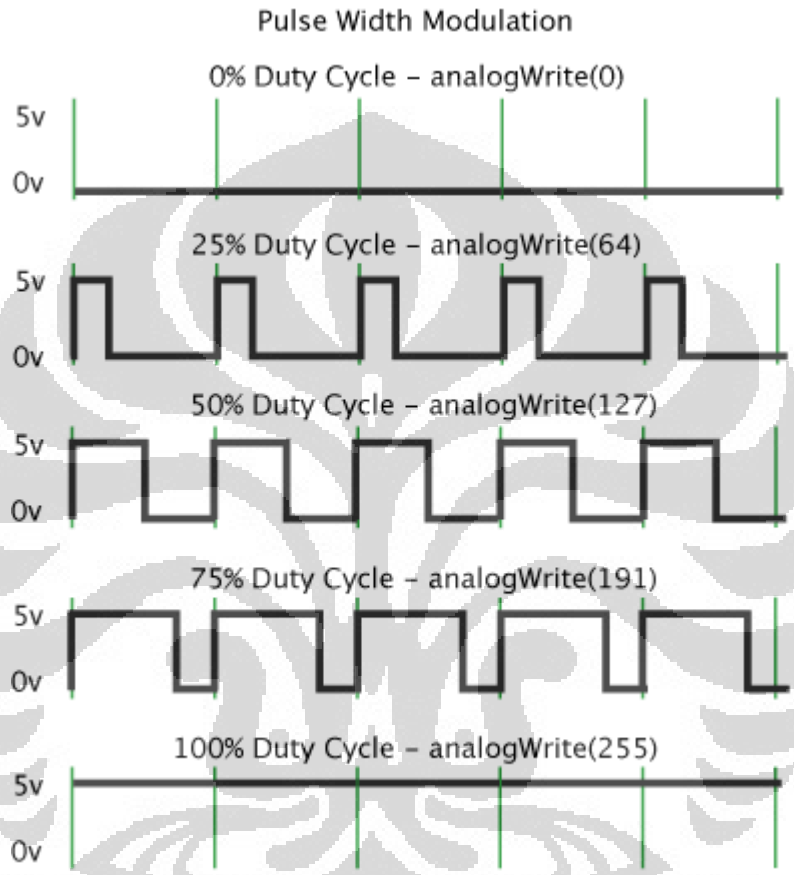
Dari persamaan 2.1 maka akan didapat rasio yang kemudian dengan menggunakan persamaan 2.2 nilai rasio ini diproses untuk mendapatkan nilai dari ppm sensor MQ-7.

$$\frac{0{,}9}{rasio} = \frac{konsentrasi}{100} \qquad (2.2)$$

2.5 Pulse Width Modulation

Pulse Width Modulation (PWM) merupakan salah satu cara untuk mengirimkan informasi analog dalam bentuk pulsa-pulsa tegangan atau pulsa-pulsa arus. Dengan modulasi, pembawa informasi terdiri dari pulsa-pulsa persegi yang berulang-ulang. Salah satu teknik modulasi pulsa yang digunakan adalah teknik modulasi durasi atau lebar dari waktu tunda positif ataupun tunda negatif pulsa-pulsa persegi tersebut. Metode tersebut dikenal dengan nama Pulse Width Modulation (PWM). Metode PWM dikenal juga dengan nama Pulse Duration Modulation (PDM) atau Pulse Length Modulation (PLM). Untuk membangkitkan

sinyal PMW, digunakan komparator untuk membandingkan dua buah masukan yaitu generator sinyal dan sinyal referensi. Hasil keluaran dari komparator adalah sinyal PWM yang berupa pulsa-pulsa persegi yang berulang-ulang. Durasi atau lebar pulsa dapat dimodulasi dengan cara mengubah sinyal referensi[9]. Gambar 2.18 merupakan gambaran dari bentuk sinyal PWM.



Gambar 2.18 Bentuk sinyal PWM [10]

Metode PWM digunakan untuk mengatur kecepatan motor, informasi yang dibawa oleh pulsa-pulsa persegi merupakan tegangan rata-rata. Besarnya tegangan rata-rata tersebut dapat diperoleh dari:

$$Vout = \frac{Vref \times duty\ cycle}{periode} \qquad (2.3)$$

Semakin lebar durasi waktu tunda positif pulsa dari sintal PWM yang dihasilkan, maka perputaran motor akan semakin cepat, demikian juga sebaliknya.

2.6 *Exhaust Fan*

Exhaust fan berfungsi untuk menghisap udara di dalam ruangan yang kemudian dibuang ke luar, dan pada saat bersamaan menarik udara segar di luar ke dalam ruangan. Berikut ini merupakan cara kerja *exhaust fan* [11]:

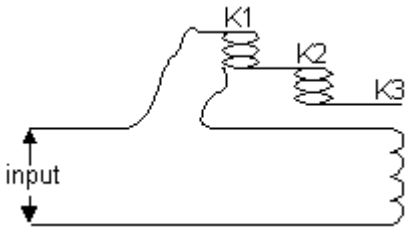
- Saat *exhaust fan* diaktifkan maka *exhaust fan* akan menghisap udara dari dalam ruangan dan membuangnya keluar ruangan.
- Udara yang dihisap dan terbuang adalah udara kotor yang sebelumnya berada dalam ruangan.
- Dengan terhisap dan terbuangnya udara kotor maka volume udara kotor di dalam ruangan akan berkurang
- Setiap kali udara terhisap keluar maka udara bersih dari luar ruangan akan masuk ke ruangan melalui lubang ventilasi, begitu seterusnya. Hal tersebut dimungkinkan karena saat udara terhisap ke luar maka tekanan udara total di dalam ruangan menjadi lebih kecil dari tekanan udara di luar ruangan, dengan demikian maka ruangan akan mendapatkan supply udara dari luar ruangan.
- Hal ini akan terus berulang selama *exhaust fan* dalam keadaan on.

2.7 *Buzzer*

Buzzer merupakan perangkat sinyal audio yang mempuyai fungsi menghasilkan suara seperti speaker, namun *buzzer* hanya mampu menghasilkan suara berfrekuensi tinggi, sedangkan speaker mampu menghasilkan suara berfrekuensi tinggi dan rendah.

Rangkaian dalam *buzzer* yang bisa dilihat pada Gambar 2.19 merupakan komponen yang berisikan lilitan dari tiga buah kawat yang berbentuk seperti switch. Apabila arus dialirkan, maka kumparan akan menghasilkan medan magnetik, sehingga menarik kawat (K3), dan memutuskan kawat (K2) dengan kawat (K1), tetapi jika arus dimatikan, maka kumparan akan kehilangan medan magnetnya sehingga kawat K3 akan terlepas dari kumparan, dan kawat K2

berhubungan dengan K1. *Buzzer* biasa dipakai pada alat-alat ringan yang membutuhkan daya kecil[12].



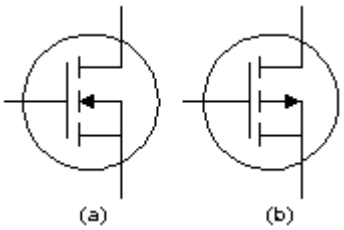
Gambar 2.20 Rangkaian dalam buzzer [12]

2.8 *Light Emitting Diode* (LED)

Light Emitting Diode (LED) adalah sebuah dioda yang dapat memancarkan cahaya, baik cahaya tampak maupun cahaya tak tampak ketika dinyalakan. Dengan memberikan forward bias pada LED maka akan terjadi rekombinasi antar pembawa muatan positif dengan pembawa muatan negatif pada daerah persambunganya. Rekombinasi ini akan menghasilkan energi, dimana pada dioda biasa energi ini keluar dalam bentuk panas sedangkan pada LED akan berbentuk cahaya. Pada umumnya LED mempunyai tegangan jatuh antara 1.5 volt sampai 3 volt dan membutuhkan arus 15 mA sampai 100 mA tergantung dari karakteristik masing-masing LED[13].

LED pada penelitian ini akan digunkan sebagai penanda kondisi ppm pada suatu ruangan. LED warna merah menandakan kondisi berbahaya (ppm>200), LED warna kuning menandakan kondisi sedang (35<ppm<200) dan LED warna hijau menandakan kondisi aman (ppm<35).

2.9 MOSFET



Gambar 2.21 Simbol MOSFET, (a) kanal-n (b) kanal-p [14]

MOSFET pada Gambar 2.21 adalah transistor berjenis FET, Field Effect Transistor jenis MOS. Fungsi transistor bermacam-macam dan bisa dianalogikan antara *collector* dan *emitter* nya merupakan resistor variable yang tergantung pada seberapa besar arus yang mengalir lewat *vase* nya. Untuk FET, *drain* dan *source* nya yang merupakan resistor variable dan tergantung pada tegangan *gate* nya. Transistor atau juga FET yang termasuk kelompok yang sama, bisa difungsikan untuk penguat arus atau tegangan elektronik, untuk fungsi sebagai saklar elektronik dan berbagai keperluan lain[14].

MOSFET pada penelitian ini digunakan untuk rangkaian PWM dimana berfungsi sebaga switch pembuat keluaran 1 dan 0. Lama keluarnya nilai 1 dan 0 ini berpengaruh terhadap *duty cycle* PWM yang akan menggerakan kecepatan perputaran kipas.

BAB III
RANCANG BANGUN SISTEM PERINGATAN POLUSI UDARA BERBASIS FPGA

Bab ini menerangkan rancang bangun sistem peringatan polusi udara berbasis FPGA secara bertahap berdasarkan metodologi *System Development Life Cycle* (SDLC) yang sudah dijelaskan pada bab sebelumnya. Tahapan perencanaan ini meliputi *user requirement*, desain, dan implementasi. Dari tahapan yang sudah dibahas sebelumya, *testing* dan analisis tidak termasuk dalam bab ini. Dalam mendokumentasikan setiap tahapan SDLC digunakan metode *Unified Modelling Language* (UML) yang merupakan diagram standar. Dengan UML, rancangan perangkat lunak dapat direpresentasikan ke dalam diagram-diagram yang memiliki fungsi masing-masing.

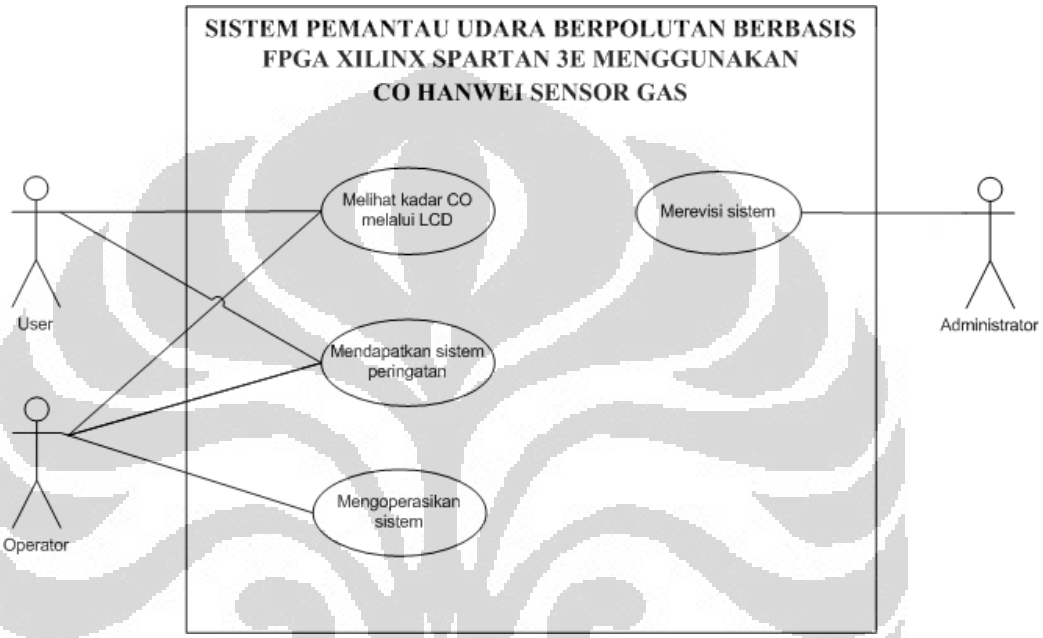
3.1 User Requirement

User requirement merupakan tahapan yang sangat penting dalam membangun sebuah sistem karena menyangkut fungsionalitas sistem yang akan dibangun, dimana *user requirement* adalah tahapan perumusan kebutuhan dari *user*. Dalam hal ini ada dua *actor* yang akan terlibat dalam sistem, yaitu *user* sebagai orang yang merasakan fasilitas dari sistem secara langsung dan operator sebagai orang yang mengoperasikan sistem.

Pada tahapan *user requirement, functional requirement* merupakan hasil dari sistem ini. *Functional requirement* merupakan suatu kebutuhan terhadap fungsionalitas sistem yang akan dibuat. Kebutuhan fungsional ini didapat langsung dari user yang akan merasakan secara langsung sistem ini. Di bawah ini adalah hasil dari *functional requirement* yang didapat dari tukar wawasan dengan beberapa *user*.

1. Operator dapat mengoperasikan sistem dengan mudah melalui Graphical User Interface yang ditampilkan di layar LCD.

1. *User* dan operator dapat melihat kadar CO secara *real time* melalui layar LCD secara interaktif. Dalam hal ini akan digunakan grafik dan warna.
2. *User* dan operator mendapatkan sistem peringatan berupa LED, *buzzer*, dan *exhaust fan*.
3. Administrator dapat melakukan perubahan pada sistem



Gambar 3.1 ***Use case diagram***

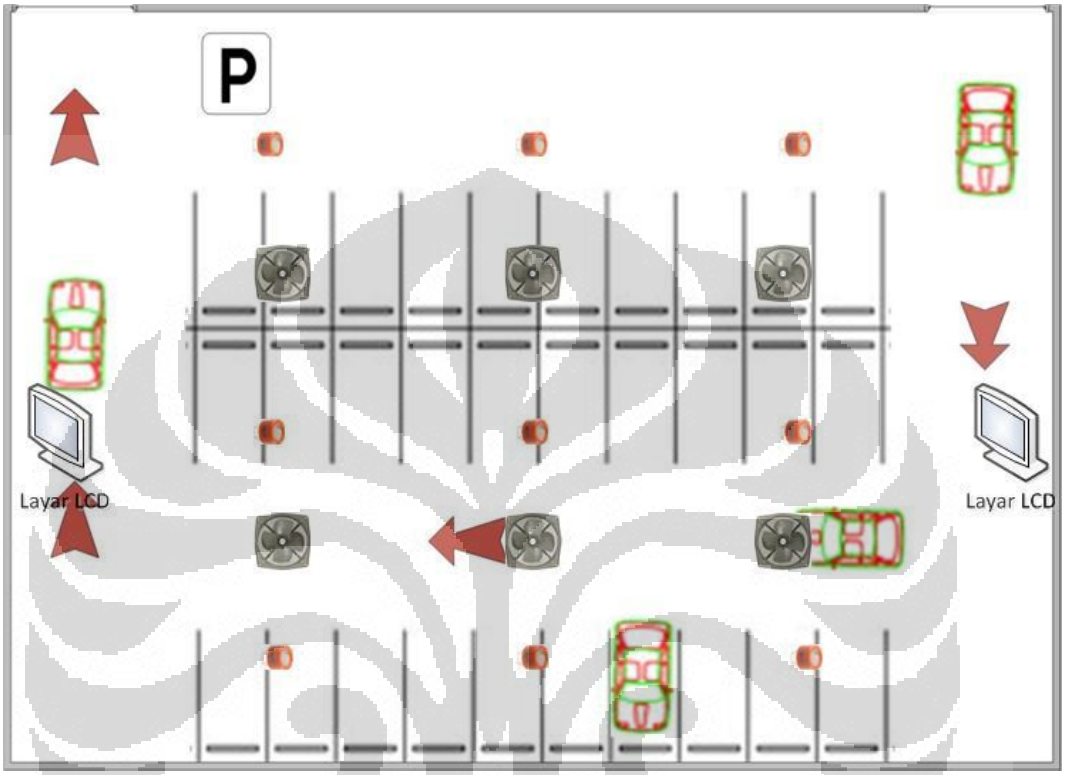
Functional requirement di atas merupakan fungsi-fungsi dasar yang harus ada pada sebuah sistem untuk memenuhi kebutuhan user yang akan menggunakannya secara langsung. Untuk menggambarkan fungsional sistem dapat digunakan *use case diagram* yang dapat dilihat pada Gambar 3.1.

3.2 Desain Sistem Peringatan Polusi Udara

Desain sistem peringatan polusi udara merupakan merupakan salah satu tahapan perancangan, dimana pada tahap ini hasil dari *user requirement* diubah menjadi sebuah model atau prototype. Desain ini mencakup desain perangkat keras, desain perangkat lunak, dan algoritma (*flowchart*, *pseudocode*, dan UML).

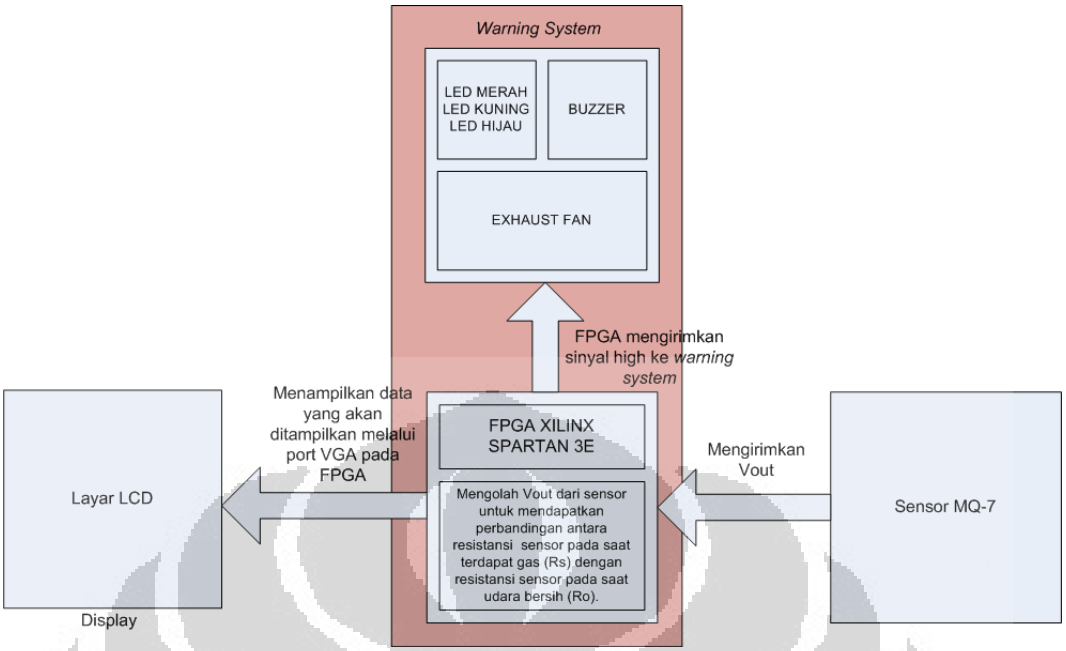
Layout dari rancang bangun sistem pada area parkir tertutup dapat dilihat pada Gambar 3.2. Pada kenyataannya rancang bangun sistem peringatan polusi

udara ini akan diimplementasikan pada sebuah *dummy box*. *Dummy box* ini digunakan sebagai simulasi pada area parkir tertutup. Sistem peringatan polusi udara dibagi menjadi tiga kondisi berdasarkan data dari sensor. Kondisi-kondisi tersebut akan dijelaskan pada sub bab berikutnya.



Gambar 3.2 *Layout* area parkir tertutup

Secara garis besar perancangan perangkat keras dari sistem ini akan menjelaskan tentang sistem peringatan dari polusi udara yang ditangkap oleh sensor MQ7 yang dikendalikan oleh FPGA. Perancangan perangkat keras untuk sistem peringatan dari sistem ini bisa dilihat pada Gambar 3.3 pada bagian yang di-*highlight* dengan warna merah.



Gambar 3.3 Blog diagram sistem peringatan polusi udara berbasis FPGA

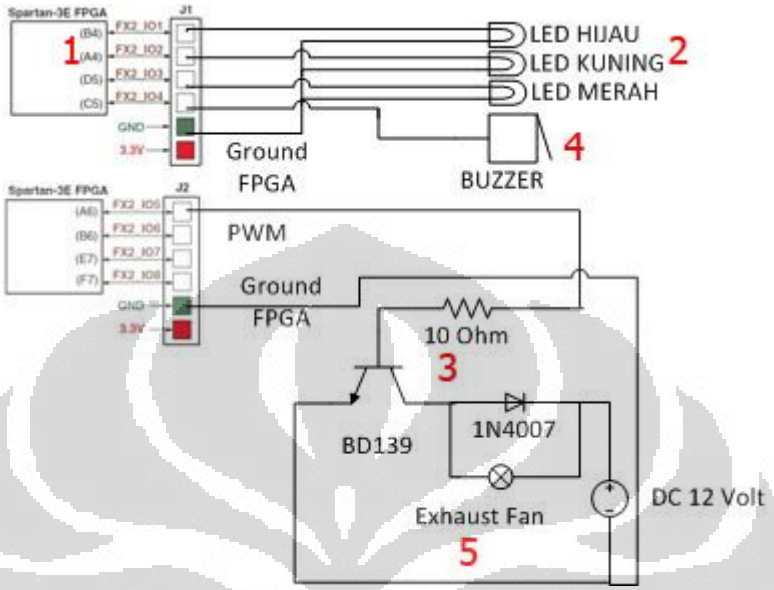
3.2.1 Desain Perangkat Keras

Komponen utama yang digunakan pada sistem peringatan ini, adalah:

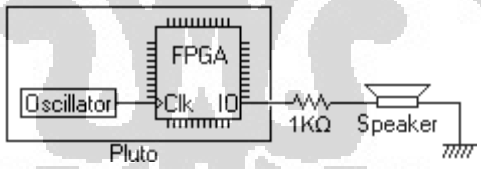
1. FPGA Xilinx Spartan 3E sebagai pengendali sistem peringatan
2. LED sebagai penanda kondisi kualitas udara
3. Rangkaian MOSFET yang digunakan sebagai switch untuk mengendalikan *exhaust fan*
4. *Buzzer* sebagai *output* suara yang menandakan keadaan berbahaya
5. *Exhaust fan* sebagai alat untuk melakukan pertukaran udara dari dalam ke luar.

Konektor ekspansi yang ada pada FPGA Xilinx Spartan 3E akan digunakan sebagai pengendali sistem peringatan pada penelitian ini. LED penanda kondisi kualitas udara akan di sambungkan dengan tiga pin header dan satu pin header akan dihubungkan dengan *buzzer* yang ada pada konektor ekspansi J1. *Output* dari PWM akan dikeluarkan pada satu pin header pada konektor ekspansi J2 yang kemudian dihubungkan dengan rangkaian MOSFET untuk

mengendalikan *exhaust fan*. *Buzzer* sebagai penanda kondisi bahaya dalam bentuk suara ini akan dihubungkan dengan satu1 pin header pada konektor ekspansi J2. Rangkaian dari sistem peringatan ditunjukkan pada Gambar 3.4.

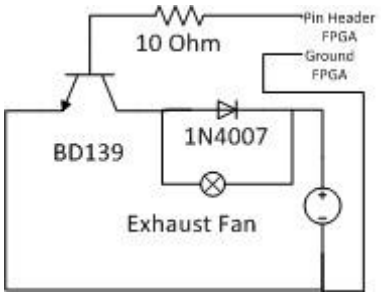


Gambar 3.4 Rangkaian *warning system*



Gambar 3.5 Rangkaian *buzzer*

Osilator menyediakan *frequency* yang tetap untuk FPGA yang dimana *frequency* ini akan dibagi oleh FPGA untuk menggerakkan IO. IO dari pin header konektor ekspansi yang terhubung dengan *buzzer* akan melalui resistor 1KΩ. Dengan mengubah frekuensi IO pada FPGA, *buzzer* akan menghasilkan suara yang berbeda-beda untuk sirine penanda keadaan berbahaya. Rangkaian dari *buzzer* yang akan digunakan pada sistem ini dapat dilihat pada Gambar 3.5.



Gambar 3.6 Rangkaian PWM

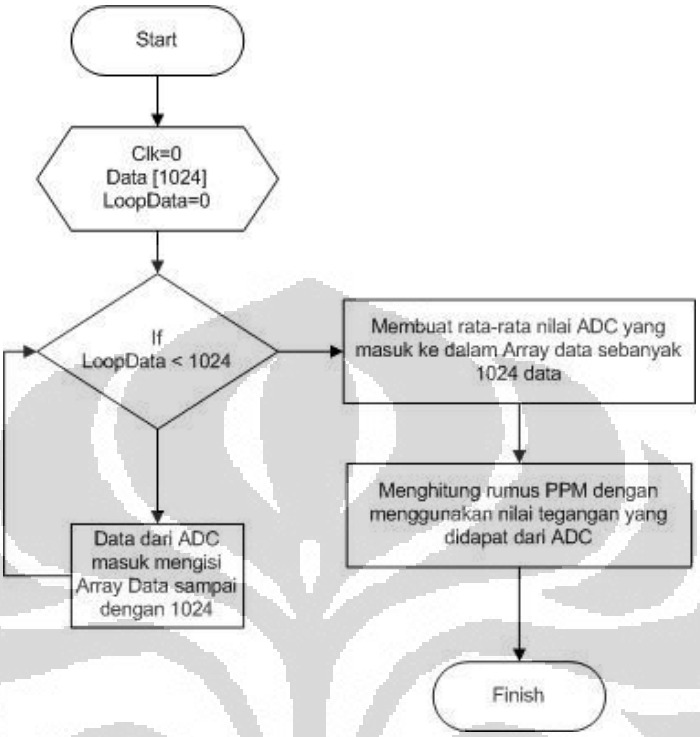
PWM akan mengendalikan besar tegangan yang akan masuk ke dalam *exhaust fan*. Konektor ekspansi pada FPGA akan mengeluarkan *output* PWM yang kemudian dihubungkan dengan resistor 10 ohm. Dengan menggunakan MOSFET sebagai switch maka tegangan yang akan diberikan kepada fan akan berbeda-beda berdasarkan pengaturan PWM. Rangkaian dari *exhaust fan* yang dikontrol dengan menggunakan PWM untuk menentukan level 1 dan level 2 dapat dilihat pada Gambar 3.6.

3.2.2 Desain Perangkat Lunak

Perancangan perangkat lunak yang paling penting pada sistem ini adalah pembacaan sinyal analog melalui rangkaian penangkap sinyal analog pada FPGA Spartan 3E karena harus melewati *pre-amplifier* dan ADC dengan menggunakan *serial peripheral interface* (SPI), dan pengaturan *exhaust fan* dengan menggunakan PWM. Dengan menggunakan SPI untuk menjalankan device *pre-amplifier* dan ADC, maka harus dibuat sebuah mekanisme untuk mematikan device lain yang juga menggunakan SPI pada saat memprogram *pre-amplifier* dan ADC. Seperti yang telah dibahas pada bab sebelumnya, memprogram *pre-amplifier* dan ADC membutuhkan SPI *timing* yang sesuai dengan datasheet dikarenakan device tidak akan berjalan dengan benar apabila tidak mengikuti karakteristik SPI *timing* yang ada pada datasheet.

Berdasarkan SPI *timing* pada Gambar 2.15, ADC akan melakukan sampel pada tahap pengambilan data pertama dan data akan keluar pada tahap kedua. Oleh karena itu, sampel nilai dari ADC akan diambil sebanyak 1024 kali agar data

yang didapat lebih stabil. Dengan begitu diagram alur atau *flowchart* dari sistem akan berubah seperti pada Gambar 3.7.



Gambar 3.7 Flowchart pengambilan data melalui ADC

Berdasarkan penjelasan sebelumnya, teknik modulasi PWM yang akan digunakan adalah teknik modulasi durasi, dimana hal ini digunakan untuk mengatur tegangan yang diberikan pada kipas sebagai pengatur kecepatan *fan*. Penundaan waktu positif dan negatif pada FPGA Spartan 3E akan dibuat dengan menggunakan sumber clock sebesar 50Mhz. Perbedaan tegangan yang diberikan akan diatur oleh *duty cycle* dengan lebar periode sebesar 1000.

3.2.2.1 Pengaturan pre-amplifier pada Rangkaian Penangkap Sinyal Analog FPGA Spartan 3E

Timer atau *counter* adalah fungsi pada FPGA yang paling sering digunakan untuk mengukur nilai yang berhubungan dengan waktu atau *clock*. Port yang digunakan untuk berkomunikasi dengan *pre-amplifier* adalah:

1. SPI_MOSI

   Serial data: *Master Output, Slave Input*. Berfungsi sebagai port yang mengirimkan setting gain ke *pre-amplifier* dalam bentuk 8-bit data

2. AMP_CS

   Digunakan sebagai port *enable* untuk mengaktifkan gain *amplifier*, Active-Low.

3. SPI_SCK

   Digunakan sebagai clock untuk SPI timing *pre-amplifier*

Untuk dapat berkomunikasi dengan *pre-amplifier* maka harus dibuat *timing* berdasarkan SPI *timing* yang ada pada Gambar 2.14, dimana *pre-amplifier* akan berjalan ketika port AMP_CS mengalami perubahan kondisi dari high ke low yang kemudian nilai *gain* dalam bentuk 8 bit akan masuk secara serial ke port SPI_MOSI dengan waktu 30ns sebelum SPI_SCK berubah kondisi dari low ke high. Setelah 8 bit gain ditransfer ke port SPI_MOSI, maka port AMP_CS harus dimatikan dengan merubah sinyal dari low ke high. Dengan mematikan port AMP_CS maka device lain yang menggunakan SPI dapat digunakan sebagai media transfer data.

3.2.2.2 Pengaturan ADC pada Rangkaian Penangkap Sinyal Analog FPGA Spartan 3E

Setelah melewati *pre-amplifier* maka sinyal analog akan masuk ke ADC. Untuk berkomunikasi dengan rangkaian ADC maka port yang dibutuhkan adalah:

1. SPI_MISO

   Serial data: *Master Input, Slave Output*. Berfungsi sebagai port yang akan mengeluarkan konversi data dari analog ke digital dalam bentuk binary sebesar 14-bit 2’s complement.

2. AD_CONV

   Digunakan sebagai port yang berfungsi untuk *shutdown* dan *reset*, *Active-High*.

3. SPI_SCK

   Digunakan sebagai clock untuk SPI timing ADC

Berdasarkan Gambar 2.15, ADC akan melakukan sampel satu kali selama siklus SPI_SCK (34 kali) dan data akan keluar pada siklus SPI_SCK berikutnya. Data yang keluar dari ADC berupa data digital dalam bentuk 14 bit 2's *complement*.

Pada sistem peringatan polusi udara ini, sampel nilai dari ADC akan diambil sebanyak 1024 kali agar data yang didapat lebih stabil dikarenakan karakteristik dari ADC yang melakukan sampel terlebih dahulu sebelum data keluar. Semakin banyak sampel yang diambil akan semakin stabil data yang dikeluarkan oleh ADC. Data dari ADC akan keluar melalui port SPI_MISO, dimana data ini akan masuk setelah waktu ke tiga dari SPI_SCK dan sebelumnya akan bernilai high *impedance*. Proses transfer data dari analog ke digital akan berlangsung secara serial ketika port AD_CONV bernilai low.

3.2.2.3 Pengaturan SPI untuk Berkomunikasi dengan Pre-Amplifier dan ADC pada FPGA Spartan 3E

Sinyal SPI pada board FPGA Spartan 3E digunakan oleh beberapa device yang lain. Hal terpenting pada saat berkomunikasi dengan AMP atau ADC adalah mematikan atau men-*disable* device yang lain untuk menghindari terjadinya tabrakan antar sinyal komunikasi dengan *device* yang lain. Tabel 3.1 menunjukkan sinyal dan nilai logic yang dibutuhkan untuk men-*disable* beberapa *device* yang menggunakan SPI sebagai komunikasi transfer data.

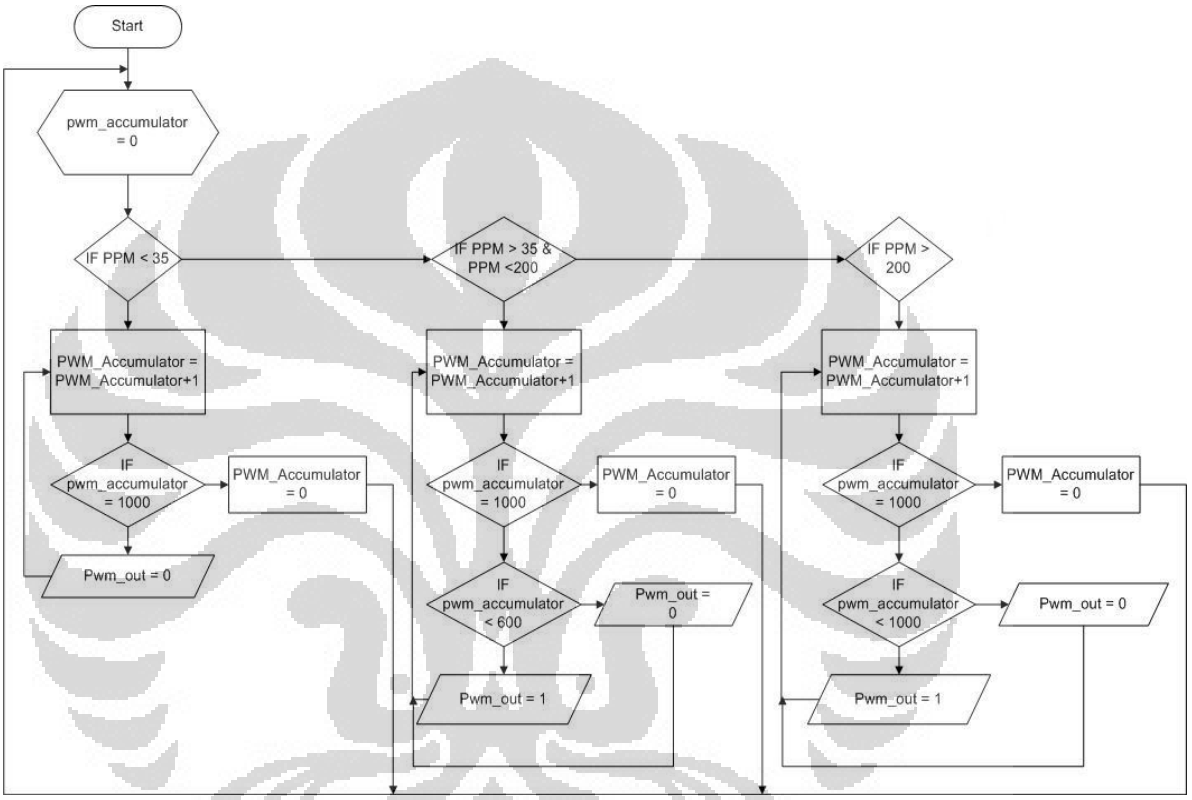
Tabel 3.1 Port SPI pada FPGA Spartan 3E

Signal	Disable Device	Disable Value
SPI_SS_B	SPI Serial Flash	1
AMP_CS	Programmable *Pre-amplifier*	1
DAC_CS	DAC	1
SF_CE0	StrataFlash Parallel Flash PROM	1
FPGA_INIT_B	Platform Flash PROM	1

3.2.2.4 Pengaturan PWM pada FPGA Spartan 3E

Teknik yang digunakan untuk mengatur modulasi pada sistem ini adalah teknik modulasi durasi, dimana teknik ini digunakan untuk mengatur tegangan

yang diberikan pada kipas sebagai pengatur kecepatan *fan*. Penundaan waktu positif dan negatif pada FPGA Spartan 3E akan dibuat dengan menggunakan sumber clock sebesar 50Mhz. Perbedaan tegangan yang diberikan akan diatur oleh *duty cycle* dengan lebar periode sebesar 1000. Gambar 3.8 menggambarkan diagram alur dari pengaturan PWM yang digunakan dalam mengatur kecepatan kipas pada sistem ini.



Gambar 3.8 Flowchart cara kerja PWM untuk mengatir kecepatan fan

Dari diagram alur pada Gambar 3.8 dapat dilihat bahwa pengaturan kecepatan level diatur dari kondisi ppm. Ketika ppm yang didapat lebih dari 35ppm, maka keluaran dari PWM akan memberikan *duty cycle* sekitar 60-65% ke Vref sebesar 12 volt sehingga Vin yang diberikan ke *fan* sebesar 7 volt, sedangkan ketika ppm yang didapat lebih dari 200ppm maka keluaran dari PWM akan memberikan *duty cycle* sebesar 100% ke Vref agar *fan* dapat bekerja dengan maksimal dengan Vin sebesar 12 volt. Pada saat ppm yang didapat pada kondisi aman maka keluaran dari PWM akan memberikan *duty cycle* sebesar 0% ke Vref agar Vin yang masuk ke *fan* sekitar 0 Volt.

3.2.3 Algoritma Sistem Peringatan Polusi Udara

Algoritma dari perancangan sistem peringatan polusi udara akan digambarkan dengan menggunakan diagram alir dan beberapa diagram UML untuk menjelaskan fungsi dan cara kerja dari sistem yang akan dibuat. Penjelasan secara garis besar dari sistem akan digambarkan melalui *flowchart* atau diagram alir dan *pseudocode*, sedangkan untuk penjelasan secara keseluruhan dari sistem akan digambarkan melalui diagram UML yaitu *sequence diagram* dan *state diagram*.

3.2.3.1 Flowchart dan Pseudocode Sistem

Desain dari sistem yang akan dibuat didasarkan pada *pseudocode* yang dapat dilihat pada Gambar 3.9 dan *flowchart* pada Gambar 3.10. Berikut ini adalah tiga kondisi yang terjadi pada saat sensor mencatat kondisi udara di sekitar area parkir:

1. Lampu hijau, kondisi dimana kualitas udara masih dalam keadaan aman (CO<35ppm).
2. Lampu kuning, kondisi dimana kualitas udara sudah masuk dalam keadaan kurang baik (35ppm<CO<200ppm) dan *exhaust fan* akan mengeluarkan udara di dalam area parkir dengan kecepatan putaran kipas tidak terlalu cepat atau level 1.
3. Lampu merah, kondisi dimana kualitas udara di area parkir sudah sangat berbahaya (CO>200ppm), *exhaust fan* akan mengeluarkan udara di dalam area parkir dengan kecepatan putaran kipas secara maksimal atau level 2 dan *buzzer* akan menyala untuk memperingati keadaan berbahaya.

Berdasarkan Gambar 3.9 dan Gambar 3.10, dibawah ini adalah proses yang terjadi pada saat sistem peringatan polusi udara mendapatkan nilai ppm dari FPGA:

1. Sensor MQ-7 menangkap karbon monoksida di udara yang kemudian mengirimkan Vout sensor ke FPGA melalui port ADC
2. Data yang di baca oleh ADC akan disampel sebanyak 1024 kali untuk mendapatkan data yang stabil

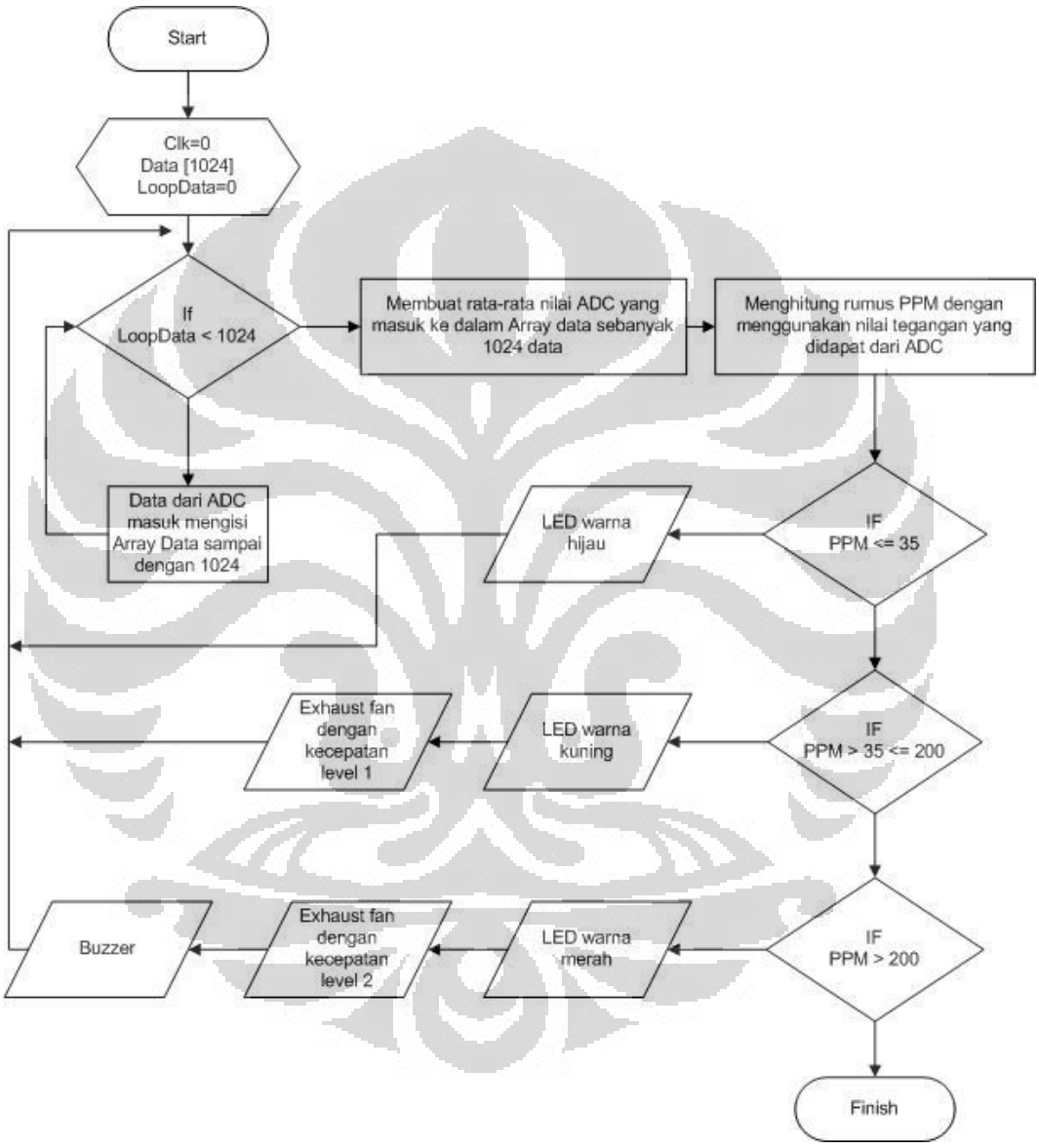
```
Start
port : clk, data[1024], ppm_sensor, loop_data, av_data,
pwm_fan1, pwm_fan2, PPM, rasio, LED_hijau, LED_kuning,
LED_merah, data_ready, buzzer
begin
if rising_edge (clk)
      when loop_data < 1024{
            data[loop_data] <= data dari ADC
            av_data = av_data + data[loop_data]
            loop_data = loop_data+1
            if loop_data = 1023 then
                  data_ready <= 1
            else
                  data_ready <= 0
            end if
      }
      if data_ready = 1 then
            av_data = av_data/1024
            rasio = 5 - av_data/av_data
            ppm_sensor = 100/rasio
            if (ppm_sensor < 25) then
                  LED_hijau <= 1
            elsif (ppm_sensor < 35 && ppm_sensor >=200) then
                  LED_kuning <= 1
                  pwm_fan1 <= 1
            elseif (ppm_sensor > 200)
                  LED_merah <= 1
                  pwm_fan2 <= 1
                  buzzer <= 1
            end if
      end if
end if
end
```

Gambar 3.9 *Pseudocode* sistem peringatan

3. Data sampel sebanyak 1024 akan dirata-rata yang kemudian akan dimasukkan dalam rumus menghitung ppm
4. Dengan menggunakan rumus yang ada pada bab 3 maka rasio dan konsentrasi CO dalam ppm akan didapat
5. Nilai ppm carbon ini akan di bandingkan dengan standar kualitas udara yang sudah dijelaskan di bab sebelumnya.
6. FPGA akan memberikan sinyal *output* melalui konektor ekspansi untuk menyalakan salah satu lampu perngatan yang menandakan kualitas udara yang didapat setelah dibandingkan dengan standar kualitas udara.
7. Pada saat nilai ppm yang didapat melewati standar kualitas udara yang baik, FPGA akan memberikan *output* PWM melalui slot konektor untuk

menyalakan *exhaust fan*, dan melaui konektor ekspansi akan menyalakan *buzzer*, dan lampu peringatan.

8. Pada saat nilai ppm yang didapat berada dalam standar kualitas udara yang baik, FPGA akan memberikan *output* PMW melalui konektor ekspansi untuk mematikan *exhaust fan*.

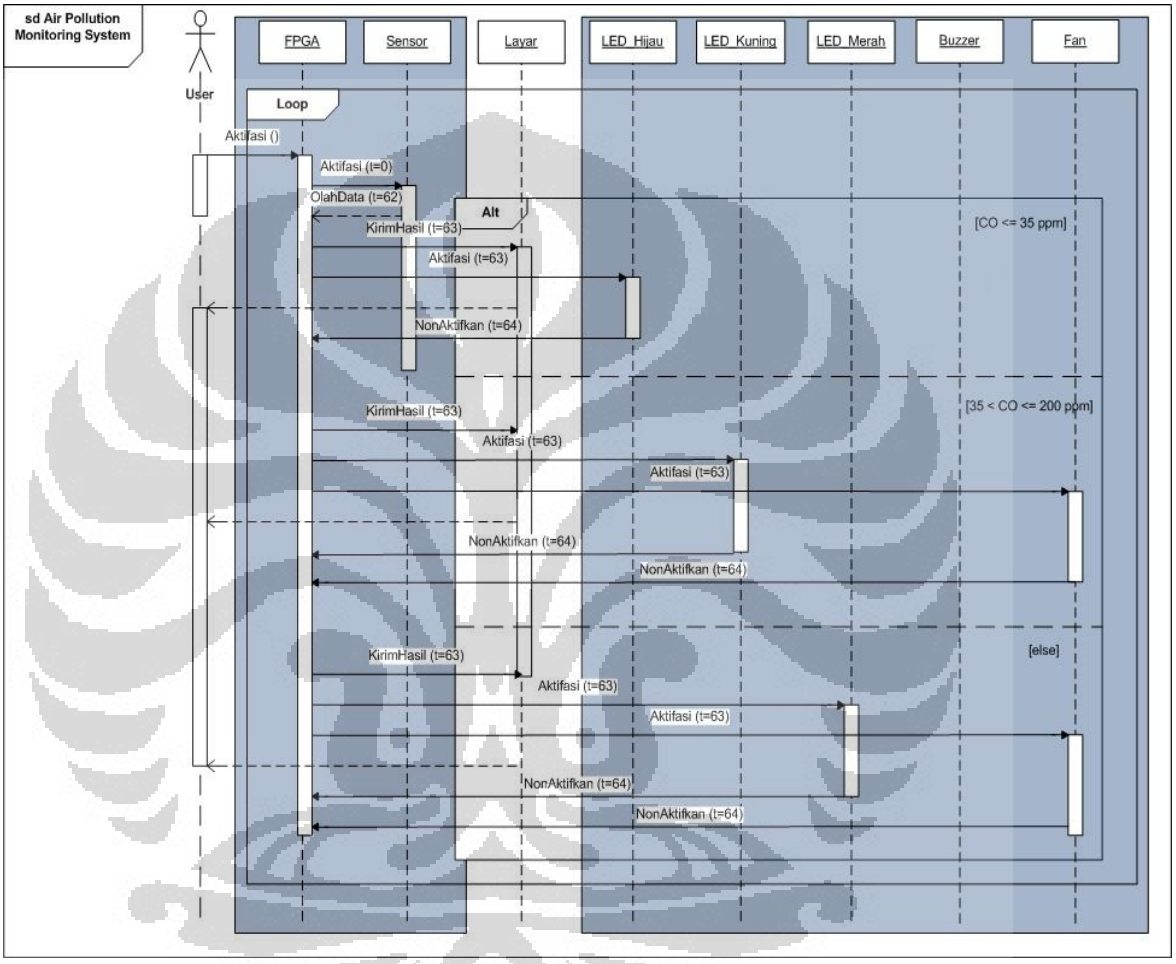


Gambar 3.10 Flowchart sistem peringatan

3.2.3.2 Unified Modelling Language

Penjelsan sistem secara keseluruhan akan dijelaskan melalui diagram UML yang digambarkan melalui *sequance diagram* yang dapat dilihat pada Gambar 3.11 pada bagian yang di-*highlight* dengan warna merah.

Program dimulai dengan *user* melakukan aktivitas pada sistem yang kemudian sensor MQ7 akan menjalakan *powercycle sensor*, yaitu melakukan kalibrasi selama 60 detik dengan masukan tegangan 5volt pada udara bersih dan 90 detik dengan masukan tegangan 1.4volt untuk melakukan pengecekan pada kondisi area parkir.



Gambar 3.11 *Sequance diagram* sistem peringatan

Setelah program pada FPGA menjalankan *powercycle sensor* dan menghitung besarnya Vout. Selanjutnya akan dikonversi menjadi nilai tegangan hambatan, dari perbandingan nilai tegangan hambatan akan diperoleh besarnya nilai ppm. Nilai ppm yang didapat akan dibandingkan dengan standar kondisi udara yang baik, sedang, dan membahayakan yang kemudian akan ditampilkan pada layar LCD, lampu penunjuk kondisi udara, dan *exhaust fan*.

Ketika nilai ppm dalam kondisi baik (CO<35ppm) maka FPGA akan memberikan sinyal *output* high ke LED untuk menyalakan lampu penunjuk

kondisi udara yang berwarna hijau, ketika nilai ppm dalam kondisi sedang (35ppm<CO<200ppm) maka FPGA akan menyalakan lampu berwarna kuning dan FPGA akan memberikan sinyal *output* PWM melalui slot ekspansi yang terhubung dengan rangkaian MOSFET yang kemudian akan menyalakan *exhaust fan* dengan kecepatan sedang dan ketika nilai ppm dalam kondisi membahayakan (CO>200ppm) maka lampu akan berwarna merah, *exhaust fan* akan berputar dengan kecepatan maksimal dan osilator FPGA memberikan *output* sebesar 3Hz untuk menyalakan *buzzer*.

3.3 Rencana Implementasi

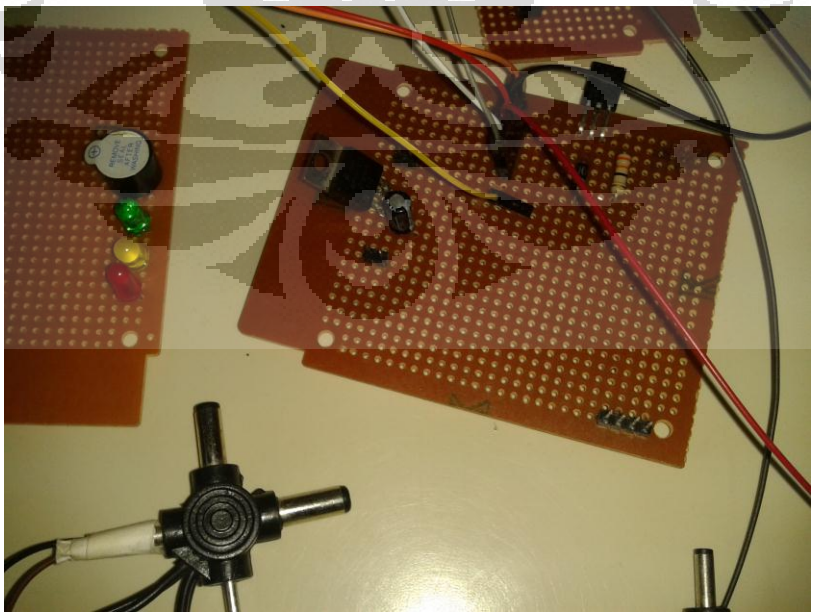
Pengujian sistem peringatan pulosi udara berbasis FPGA ini dilakukan dengan melakukan uji coba terhadap waktu yang dibutuhkan *exhaust fan* untuk mengeluarkan data CO pada *dummy box* dan *response time* yang dibutuhkan FPGA dalam melakukan proses pengolahan data sampai *output* keluar yang akan dibandingkan dengan simulasi RTL. Hasil uji coba ini akan menampilkan kemampuan kerja sistem apakah sistem peringatan berbasis FPGA ini cukup *reliable* untuk diimplementasikan pada area parkir tertutup.

BAB IV
UJI COBA DAN ANALISIS SISTEM PERINGATAN POLUSI UDARA BERBASIS FPGA

4.1 Implementasi Perangkat Keras

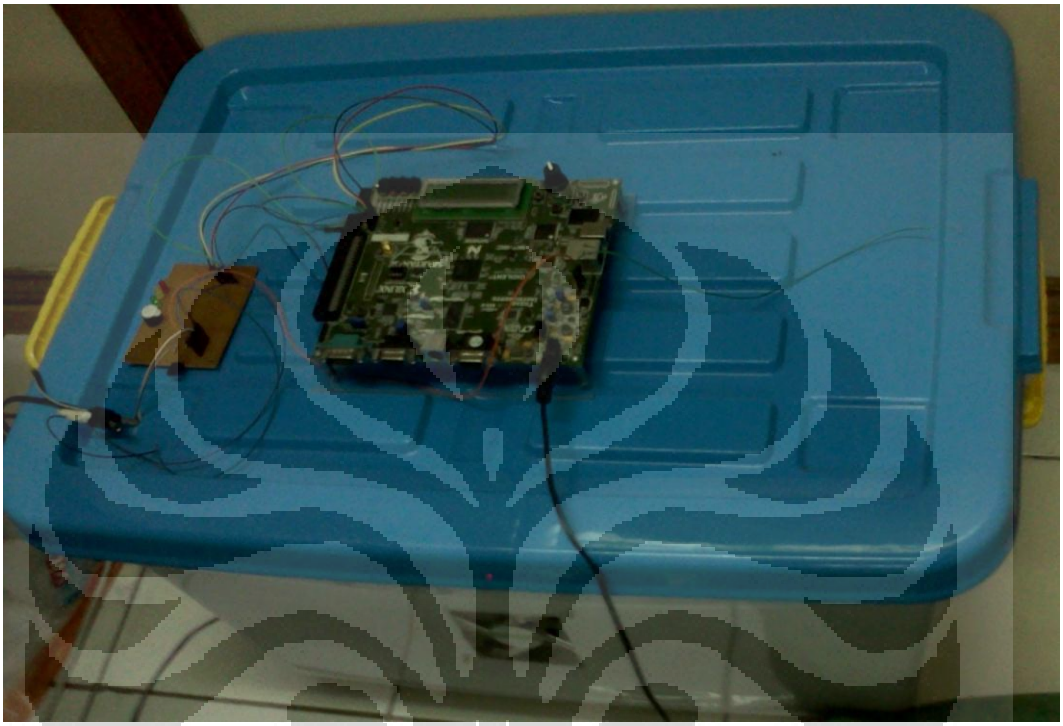
Perangkat keras dibuat menjadi dua buah modul yaitu modul pengendali *fan* dengan menggunakan MOSFET yang memberikan tegangan berbeda-beda berdasarkan *output* dari PWM dan modul penanda kondisi udara CO yang berisi LED dan *buzzer*. Port slot ekspansi pada FPGA Spartan 3E akan dihubungkan dengan dua rangkaian tersebut sebagai pembawa *output* dari program.

Rangkaian yang akan digunakan dapat dilihat pada Gambar 3.4. Salah satu pin pada port slot ekspansi pada FPGA Spartan 3E digunakan untuk menghubungkan FPGA dengan rangkaian MOSFET yang berfungsi sebagai switch pengendali kencepatan *fan*. Kecepatan putar dari *fan* diatur dengan menggunakan PWM untuk mengubah sinyal referensi sebesar 12 volt menjadi 7 volt dan 12 volt.



Gambar 4.1 Rangkaian *warning system* pada PCB

Pada Gambar 4.1, terlihat dua modul yang sudah dirangkai pada PCB bolong. Perbedaan terdapat pada salah satu modul berfungsi untuk menjalankan LED dan buzzer Satu modul yang lain adalah rangkaian PWM yang berfungsi untuk mengatur kecepatan fan.

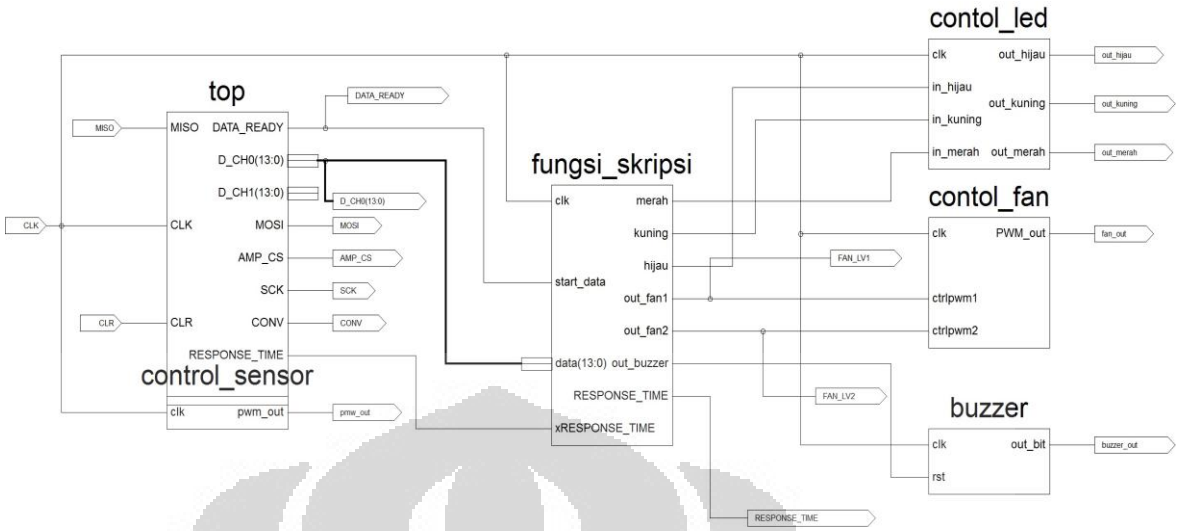


Gambar 4.2 Simulasi area parkir tertutup ada *dummy box*

Untuk simulasi area parkir tertutup digunakan dummy box sebagai tempat untuk menguji sistem peringatan polusi udara CO. Gambar 4.2 menggambarkan simulasi sistem peringatan polusi udara CO pada *dummy box* yang sudah terpasang fan sebagai pengganti *exhaust fan*, LED dan *buzzer*. *Fan* komputer digunakan sebagai pengganti *exhaust fan* pada pengujian sistem peringatan berbasis FPGA.

4.2 Implementasi Perangkat Lunak

Perancangan perangkat lunak pada sistem peringatan polusi udara CO pada area parkir tertutup ini dibuat menjadi beberapa modul pada Xilinx ISE WebPack 13.2 yang kemudian dibuat menjadi *schematic* untuk digabungkan menjadi satu. Gambar 4.3 menunjukkan rangkaian *schematic* pada Xilinx ISE WebPack 13.2 yang akan dimasukkan ke FPGA Spartan 3E.



Gambar 4.3 *Schematic* sistem peringatan polusi udara

4.3 Pengujian dan Analisa Sistem

Untuk memvalidasi rancangan sistem peringatan polusi udara berbasis FPGA yang telah dibuat, pada bab ini akan dilakukan pengujian dari rancangan yang telah dibuat. Beberapa parameter yang telah ditentukan pada perancangan sebelumnya juga akan diujicobakan agar mendapat hasil yang optimal. Pengujian yang dilakukan adalah pengujian *response time* FPGA pada pengambilan data sampel ADC, pengujian *response time fan* dengan kondisi bahaya dan sedang, pengujian *response time* sistem pada *dummy box* dan variasi tegangan fan pada sistem peringatan polusi udara.

4.3.1 Pengujian dan Analisa Response Time FPGA pada Pengambilan Data Sampel ADC dan Keseluruhan Sistem

Pengujian dilakukan dengan menghitung *response time* saat sistem menerima data sampai data siap dikeluarkan dan menghitung besar *delay* pada sistem saat akan menjalankan siklus program berikutnya. Kondisi yang digunakan adalah pada saat sistem melakukan pengambilan sampel data pada ADC dan keseluruhan sistem dari pengambilan sampel sampai data akhir siap untuk dikeluarkan ke *exhaust fan*, *buzzer*, led, dan layar CRT. Pengambilan data setiap kondisi dilakukan sebanyak 10 kali pada saat menghitung *response time* kerja

sistem dan menghitung besar *delay* saat akan menjalankan siklus program berikutnya dengan menggunkaan *oscilloscope dso*.

Pengujian dilakukan dengan cara menggunakan port ekspansi J4 pada FPGA Spartan 3E untuk mencatat *response time.* Kondisi yang digunakan adalah pada saat ADC melakukan sampel pertama kali sampai ADC mengeluarkan hasil perhitungan rata-rata dari 1024 sampel dan pada saat menjalankan keseluruhan sistem.

Pada saat ADC akan melakukan pengambilan data sampel maka port ekspansi J4 akan mengirimkan nilai 1 dan ketika data akhir pada ADC keluar dalam bentuk rata-rata maka port ekspansi J4 akan mengirimkan nilai 0. Port ekspansi J4 akan dihubungkan dengan *oscilloscope dso* untuk melihat frekuensi selama pengambilan sampel ADC. Dengan melihat frekuensi pada *oscilloscope dso* maka dapat dihitung *response time* dari FPGA dengan kondisi pengambilan data pada ADC, begitu juga dengan pengambilan data pada saat kondisi keseluruhan sistem. Hasil pengujian pada kondisi pengambilan sampel pada ADC dan hasil pengujian pada kondisi keseluruhan sistem dapat dilihat pada Tabel 4.1.

Tabel 4.1 Data pengujian dengan menggunkan *oscilloscope dso*

Sampel Data	Sampel ADC		Sistem Keseluruhan	
	Response Time (ns)	Delay (ns)	*Response Time* (ns)	Delay (ns)
1	147,000	20.20	149,000	20.20
2	148,000	20.40	150,000	20.40
3	148,000	19.80	150,000	20.00
4	147,000	20.00	150,000	19.60
5	148,000	20.00	150,000	20.00
6	148,000	20.00	148,000	20.00
7	148,000	20.20	148,000	19.80
8	148,000	20.20	148,000	20.20
9	149,000	20.00	148,000	20.00
10	149,000	20.00	150,000	20.00
Rata-rata	**148,000.00**	**20.08**	**149,100.00**	**20.02**

Sebagai data tambahan disertakan pula pengujian dengan menggunakan simulasi RTL pada program Xilinx ISE WebPack 13.2 yang dapat dilihat pada lampiran A.2. Hasil pengujian dari simulasi RTL didapat dari dua kondisi yaitu

pengambilan sampel data pada ADC dan keseluruhan sistem yang dapat dilihat pada Tabel 4.2.

Tabel 4.2 Data pengujian dengan menggunkan simulasi RTL

Kondisi	*Response Time* (ns)	Delay (ns)
Sampel ADC	148,300.00	20.00
Keseluruhan Sistem	148,300.00	20.00

Dengan membandingkan Tabel 4.1 dan Tabel 4.2 maka akan didapat perbedaan waktu antara simulasi dan implementasi. Perbedaan waktu yang didapat dapat dilihat pada Tabel 4.3.

Tabel 4.3 Perbedaan waktu antara implementasi dan simulasi RTL pada sistem

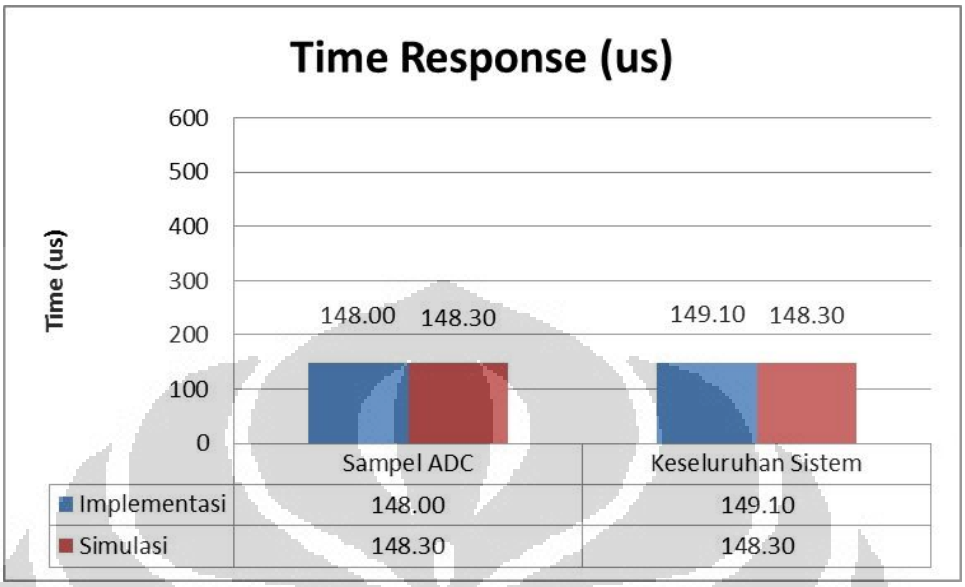
Kondisi	Implementasi		Simulasi		Perbedaan waktu	
	Response Time (ns)	Delay (ns)	*Response Time* (ns)	Delay (ns)	*Response Time* (ns)	Delay (ns)
Sampel ADC	148,000	20.08	148,300	20.00	**300**	**0.08**
Keseluruhan Sistem	149,100	20.02	148,300	20.00	**800**	**0.02**

Dari Tabel 4.3 dapat dilihat bahwa *response time* dan delay antara implementasi dan simulasi RTL tidak jauh berbeda dengan selisih 0,37% dan 0,25%. Sehingga untuk melihat *output*, *response time* dan kerja sistem secara keseluruhan dapat dilihat melalui simulasi RTL karena perbedaan yang terjadi antara implementasi dan simulasi RTL dapat dikatakan tidak ada.

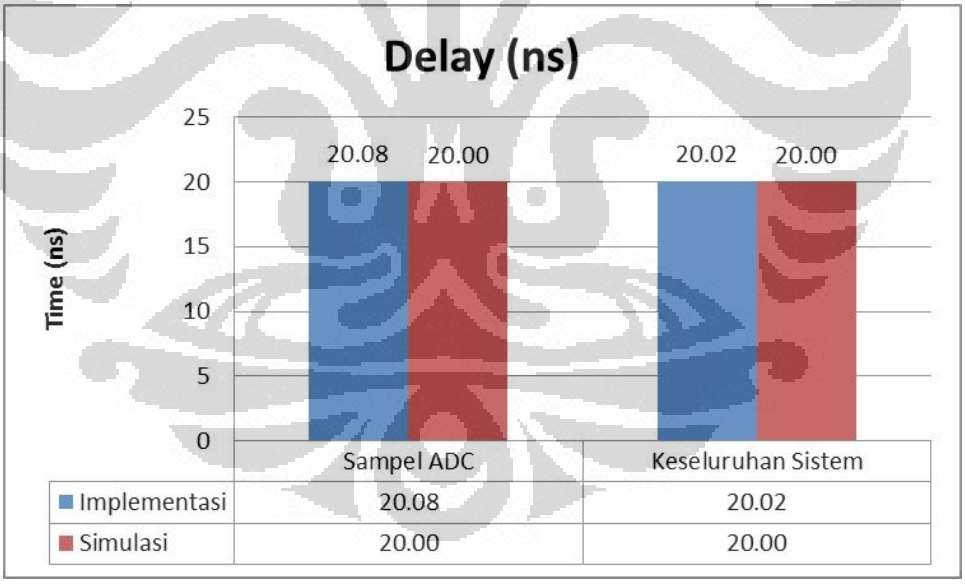
Berdasarkan data pengujian, timing diagram simulasi RTL dan implementasi dari sistem peringatan ini bisa dikatakan sama dengan selisih 0.37% untuk *response time* dan 0.25% untuk *delay*. Dari berbagai kondisi yang diujicobakan pada simulasi dan implementasi semua memiliki waktu yang tidak jauh berbeda. Dengan menggunakan rata-rata pada setiap kondisi yang bisa dilihat pada Gambar 4.4 dan Gambar 4.5, dimana perbedaan *response time* yang terjadi tidak begitu signifikan. Hal ini dapat terjadi karena beberapa faktor, antara lain:

- Tidak stabilnya frekuensi *clock* FPGA, sehingga memungkinkan terjadinya penyimpangan.
- Pengukuran waktu dilakukan secara manual yaitu dengan membaca frekuensi pada *oscilloscope dso* sehingga data yang diperoleh kurang

presisi, dimana cara pembacaan pada *oscilloscope dso* dapat dilihat pada Lampiran A.1.



Gambar 4.4 Grafik perbandingan response time implementasi dan simulasi RTL pada sampel ADC dan keseluruhan sistem



Gambar 4.5 Grafik perbandingan delay implementasi dan simulasi RTL pada sampel ADC dan keseluruhan sistem

Pada Gambar 4.4 dapat dilihat bahwa *response time* antara kondisi ADC dan keseluruhan sistem tidak memiliki perbedaan waktu, dimana diasumsikan kita menggunakan data dari simulasi RTL. Hal ini dikarenakan FPGA menerapkan

elemen programmable logic berjalan secara parallel, dimana FPGA akan menjalanakan setiap modul secara paralel.

4.3.2 Pengujian dan Analisa Response Time Fan pada Kondisi Bahaya dan Kondisi Sedang

Pengujian dilakukan dengan menghitung waktu yang dibutuhkan fan untuk mengeluarkan CO pada *dummy box* dengan menggunakan stopwatch. Kondisi yang digunakan adalah pada saat kadar CO di *dummy box* berbahaya (lebih dari 200ppm) dan sedang (diantara 35ppm sampai dengan 200ppm). Pengambilan data setiap kondisi dilakukan sebanyak 10 kali pada saat Vin fan 12 volt dengan jangakuan ppm CO sebesar 200-250 ppm (3.4 volt pada Vout sensor) dan 7 volt dengan jangakuan ppm CO sebesar 100-150 ppm (2.6 volt pada Vout sensor). Tegangan yang digunakan pada fan sebesar 7 volt dan 12 volt dengan kecepatan kipas sebesar 800 - 1500 rpm dan daya hisap udara sebesar 23,6 – 45.54 l/s. Data yang didapatkan akan dibandingkan dengan kondisi sebenarnya dengan perbandingan daya hisap kipas dan besar area parkir yang digunakan melalui perhitungan. Hasil pengujian pada kondisi bahaya dan sedang dapat dilihat pada Tabel 4.4.

Tabel 4.4 Data pengujian response time fan pada dummy box

No	Bahaya (detik)		Sedang (detik)	
	12 Volt	7 Volt	12 Volt	7 Volt
1	5.2	9.5	3.2	6.2
2	4.8	8.3	3.5	5.8
3	4.5	9.1	2.5	6.4
4	5.2	10.2	3.2	6.2
5	5.5	8.7	2.8	5.5
6	5.1	8.2	2.8	5.7
7	5.3	9.1	3.1	5.8
8	4.8	8.5	3.2	5.2
9	5.4	8.7	2.7	5.4
10	4.7	8.9	3.1	5.2
Rata - rata	**5.04**	**8.92**	**3.1**	**5.74**

Sebagai data tambahan disertakan juga perhitungan waktu yang dibutuhkan fan untuk mengeluarkan CO pada *dummy box* berdasarkan karakteristik fan pada Tabel 4.5. Kadar CO mempunyai volume 82 liter pada

perhitungan kondisi bahaya, sedangkan pada perhitungan kondisi sedang CO mempunyai volume 41 liter. Perhitungan ini mengabaikan waktu tunda yang dibutuhkan sensor dalam membaca kadar CO.

$$Waktu = \frac{V}{AF} \qquad (4.1)$$

Dimana,

V: volume ruangan

AF: air flow dari fan atau exhaust fan

Tabel 4.5 Karakteristik *fan aerocool shark* 14cm

	Power mode	Silence mod
Voltage (V)	12	7
Speed (RPM)	1500	800
Air Flow (l/s)	45.54	23.6

Dengan menggunakan persamaan 4.1 dan data *air flow fan* sebesar 45.54 l/s pada Tabel 4.6 dan volume gas di *dummy box* sebesar 82 liter dan 41 liter, maka didapatkan waktu yang dibutuhkan fan untuk mengeluarkan CO sebesar:

$$Waktu = \frac{82\ l}{45.54\ l/s} = 1.8\ second$$

$$Waktu = \frac{41\ l}{45.54\ l/s} = 0.9\ second$$

Sedangkan untuk air flow fan sebesar 23.6 l/s akan didapatkan waktu yang dibutuhkan fan untuk mengeluarkan CO sebesar:

$$Waktu = \frac{82\ l}{23.6\ l/s} = 3.5\ second$$

$$Waktu = \frac{41\ l}{23.6\ l/s} = 1.75\ second$$

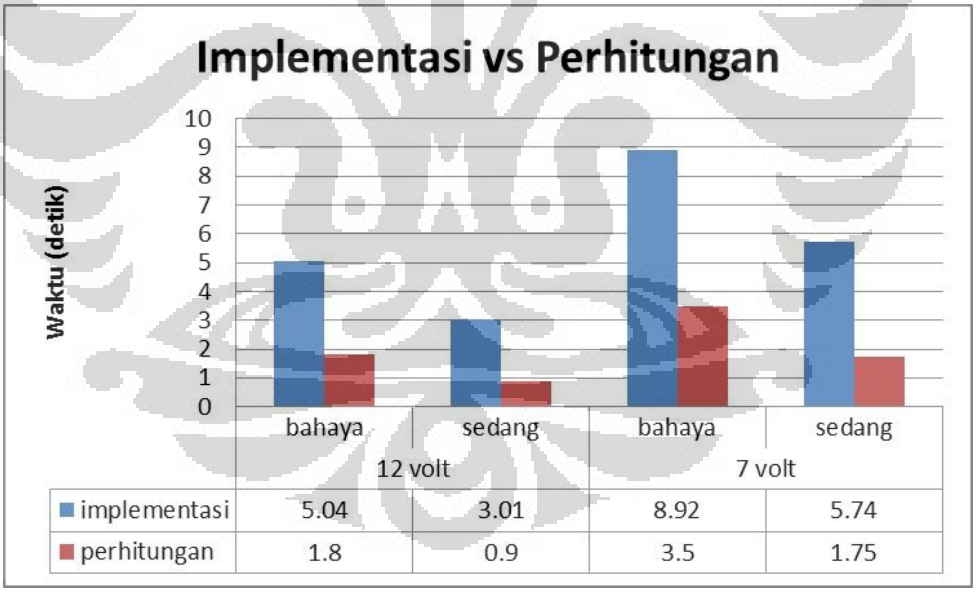
Dengan membandingkan Tabel 4.4 dan hasil dari perhitungan, maka akan didapat perbedaan waktu antara perhitungan dan implementasi. Perbedaan waktu yang didapat dapat dilihat pada Tabel 4.6.

Tabel 4.6 Perbedaan waktu antara implementasi dan perhitungan

Kondisi	Implementasi (detik)		Perhitungan (detik)		Perbedaan waktu (detik)	
	12 Volt	7 Volt	12 Volt	7 Volt	12 Volt	7 Volt
Bahaya	5.04	8.92	1.8	3.5	3.24	5.42
Sedang	3.1	5.74	0.9	1.75	2.2	3.99

Dari Tabel 4.7 dapat dilihat bahwa terdapat perbedaan waktu antara implementasi dan perhitungan cukup besar dengan selisih sebagai berikut:

1. Pada kondisi bahaya dengan *duty cycle fan* sebesar 100% dan 60%, didapat selisih 64.3% dan 60.7%
2. Pada kondisi sedang dengan *duty cycle fan* sebesar 100% dan 60%, didapat selisih 71% dan 69.5%



Gambar 4.6 Grafik perbandingan response time fan pada implementasi dan perhitungan

Berdasarkan data pengujian pada fan, perbedaan waktu yang dihasilkan cukuplah besar. Dari berbagai kondisi dan Vin fin yang diujicobakan semua dideteksi dengan baik, dengan waktu yang dihasilkan untuk mengeluarkan CO pada *dummy box* dapat dilihat pada Gambar 4.6. Perbedaan waktu implementasi dan perhitungan cukuplah besar dengan selisih 64.3% dan 60.7% pada kondisi

bahaya dan 71% dan 69.5% pada kondisi sedang. Hal ini dapat terjadi karena beberapa faktor, antara lain:

- Tidak stabilnya kadar CO yang ada pada dummy box pada saat pengambilan data
- Tidak diikutsertakannya waktu tunda sensor saat membaca kadar CO dalam perhitungan
- Adanya waktu tunda dari FPGA ke dalam rangkaian PWM yang menjalakan fan
- Pengukuran waktu dilakukan secara manual yaitu dengan stopwatch sehingga data yang diperoleh kurang presisi.

4.3.3 Analisis Perbandingan Sistem Peringatan pada Dummy Box dan Area Parkir Tertutup

Dari hasil pengujian simulasi sistem peringatan polusi udara pada *dummy box* dapat dilakukan perbandingan dengan keadaan nyata yaitu area parkir tertutup. Variabel pembanding untuk menghitung perkiraan waktu yang dibutuhkan sistem untuk mengeluarkan CO antara fan (aerocool shark 14cm) yang digunakan pada *dummy box* dan *exhaust fan* (sanhe DJF(a) series) ditunjukkan pada Tabel 4.7.

Tabel 4.7 Variabel pembanding antara fan (aerocool shark 14cm) dan *exhaust fan* (sanhe DJF(a) series)

	Fan	*Exhaust fan*
RPM	1500	1400
Air Flow (l/s)	45.54	1583.33
Volt	12 Volt	380 Volt

Berdasarkan data pengujian, waktu yang dibutuhkan fan untuk mengeluarkan CO pada *dummy box* pada kondisi bahaya (200 ppm) adalah 5.04 detik dengan volume sebesar 0.082 m^3. Pada area parkrir tertutup, diasumsikan volumenya sebesar 3000 m^3, air flow exhaust fan adalah 1583.33 l/s, dan exhaust yang digunakan sebanyak satu buah. Dengan menggunkan persamaan 4.2 akan didapatkan waktu yang dibutuhkan *exhaust fan* untuk mengeluarkan CO pada area parkir tertutup.

$$t_{exhaust\ fan} = \frac{V \times 5.04\ detik}{AF \times \sum exhaust\ fan} \quad (4.2)$$

Dimana,

∑exhaust fan = jumlah *exhaust fan*

$t_{exhaust\ fan}$ = waktu yang dibutuhkan *exhaust fan* untuk mengeluarkan CO

5.04 detik = waktu yang dibutuhkan fan pada percobaan untuk mengeluarkan CO

V = perbandingan volume area parkir tertutup dengan *dummy box*

$$V = \frac{Volume\ Area\ Parkir\ (m^3)}{0.082\ m^3}$$

$$V = \frac{3000\ m^3}{0.082\ m^3} = 36585.4$$

AF = perbandingan Air Flow *exhaust fan* dengan fan

$$AF = \frac{Air\ Flow\ Exhaust\ Fan\ (l/s)}{45.54\ l/s}$$

$$AF = \frac{1583.33\ l/s}{45.54\ l/s} = 34.77$$

$$Waktu_{exhaust\ fan} = \frac{36585.4}{34.77} \times 5.04\ detik = 5303.15\ detik = 1.47\ jam$$

Secara teoritis, waktu yang dibutuhkan satu buah *exhaust fan* untuk mengeluarkan CO pada area parkir tertutup pada kondisi bahaya (200 ppm) adalah selama 1.47 jam, namun pada kenyataanya proses yang terjadi bisa lebih cepat. Hal ini dapat terjadi karena beberapa faktor, antara lain:

- Penggunaan exhaust fan pada area parkir tertutup bisa lebih dari satu buah sehingga proses pengeluaran CO menjadi lebih cepat dibanding dengan menggunakan satu exhaust fan.

- Area parkir yang tidak tertutup secara rapat seperti *dummy box* dapat membantu mempercepat proses pembuangan CO karena terjadi pertukaran udara dari luar ke dalam.

Sehingga dengan menurunkan persamaan 4.2, akan didapat persamaan 4.3 untuk menghitung jumlah exhaust fan yang dibutuhkan pada saat mengeluarkan CO dengan kondisi bahaya (200 ppm) di area parkir tertutup.

$$\sum exhaust\ fan = \frac{V \times 5.04\ detik}{AF \times t_{exhasut\ fan}} \qquad (4.3)$$

BAB V
KESIMPULAN

1. Sistem peringatan polusi udara CO telah berhasil dibuat menggunakan FPGA Spartan 3E dan berhasil membaca sinyal analog yang dikirim sensor yang kemudian diolah dan dikeluarkan dalam bentuk LED, *buzzer* dan *fan*.
2. Untuk mendapatkan data yang stabil dari ADC pada FPGA Spartan 3E harus dilakukan 1024 kali sampel pengambilan data.
3. Dari pengujian dan analisis *response time* antara simulasi RTL dan implementasi, didapatkan hasil bahwa timing diagram antara simulasi RTL dan implementasi tidak berbeda jauh dengan selisih waktu 0.37%.
4. Dari pengukuran *response time fan*, didapatkan hasil bahwa waktu yang dibutuhkan sistem untuk mengeluarkan CO lebih lama 60 - 71% dari perhitungan dikarenakan terdapat jeda waktu pembacaan kadar CO terbaru oleh sensor.
5. Jumlah exhaust fan yang dibutuhkan untuk mengeluarkan CO pada kondisi bahaya (200 ppm) di area parkir tertutup adalah:

$$\sum exhaust\ fan = \frac{V \times 5.04\ detik}{AF \times t_{exhasut\ fan}}$$

DAFTAR REFERENSI

[1] Anonymous. (2011). Polusi Udara. [Online]. Available: http://www.menlh.go.id [Accessed: 27 Desember 2011]

[2] Wardhana, W.A. 1999. Dampak Pencemaran Lingkungan. Andi Offset. Yogyakarta.

[3] Hadayani, Dra Murti. (2012). Keracunan Karbon Monoksida. Available: http://www.pom.go.id/public/siker/desc/produk/RacunKarMon.pdf [Accessed: 27 Desember 2011]

[4] Anonymous. (2011). VLSI (FPGA , FPAA DAN VHDL). [Online]. Available: http://tsulaksana.wordpress.com/2011/06/29/vlsi-fpga-fpaa-dan-vhdl/ [Accessed: 27 Desember 2011]

[5] Anonymous. (2011). MC CDMA pada FPGA.[Online]. Available: http://ska22.wordpress.com/2011/02/14/mc-cdma-pada-fpga/ [Accessed: 27 Desember 2011]

[6] Spartan-3E Starter Kit Board User Guide, UG230 (v1.0) March 9, 2006.

[7] Modul Praktikum Teknik Digital. Universitas Indonesia., 2009.

[8] Hanwei Electronics Co. LTD. Technical Data MQ-7 Gas Sensor. [Online]. http://www.sparkfun.com/datasheets/Sensors/Biometric/MQ-7.pdf [Accessed: 27 Desember 2011]

[9] Aditya Febrianto. (2009). PWM. [Online]. Available: http://www.scribd.com/doc/23350279/pwm [Accessed: Mei 2012]

[10] Timothy Hirzel. (2012). PWM. [Online]. Available: http://arduino.cc/it/Tutorial/PWM [Accessed: Mei 2012]

[11] Anonymous. (2011). Exhaust fan. [Online]. Available: http://elektindo.com/cara-kerja-exhaust-fan.html [Accessed: 27 Desember 2011]

[12] Anonymous. (2011). Buzzer. [Online]. Available: http://opi.110mb.com/opihomepage/pendukung.htm [Accessed: 27 Desember 2011]

[13] Anonymous. (2011). Efek Fotolistrik. [Online]. Available: http://blog.uad.ac.id/sulisstyawan/2012/01/03/efek-fotolistrik/ [Accessed: Mei 2012]

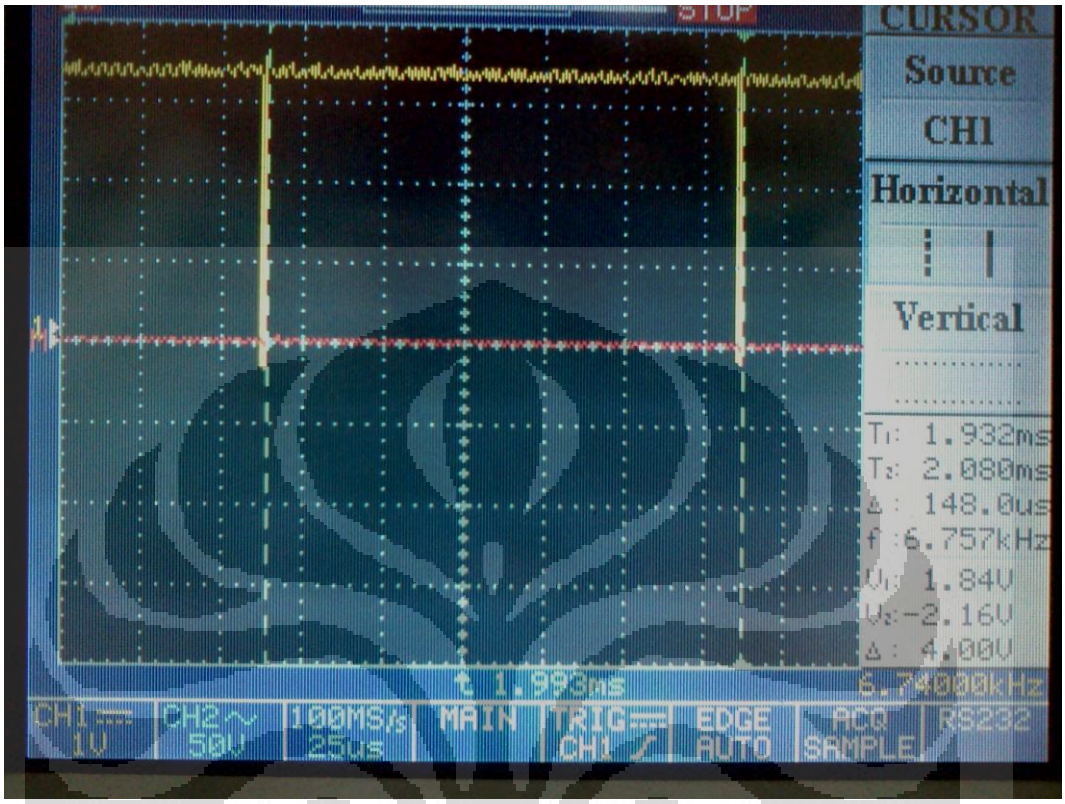
[14] Robby.C. (2011). Transistor FET. [Online]. Available: http://robby.c.staff.gunadarma.ac.id/Downloads/files/4798/Transistor+FET.doc [Accessed: Mei 2012]

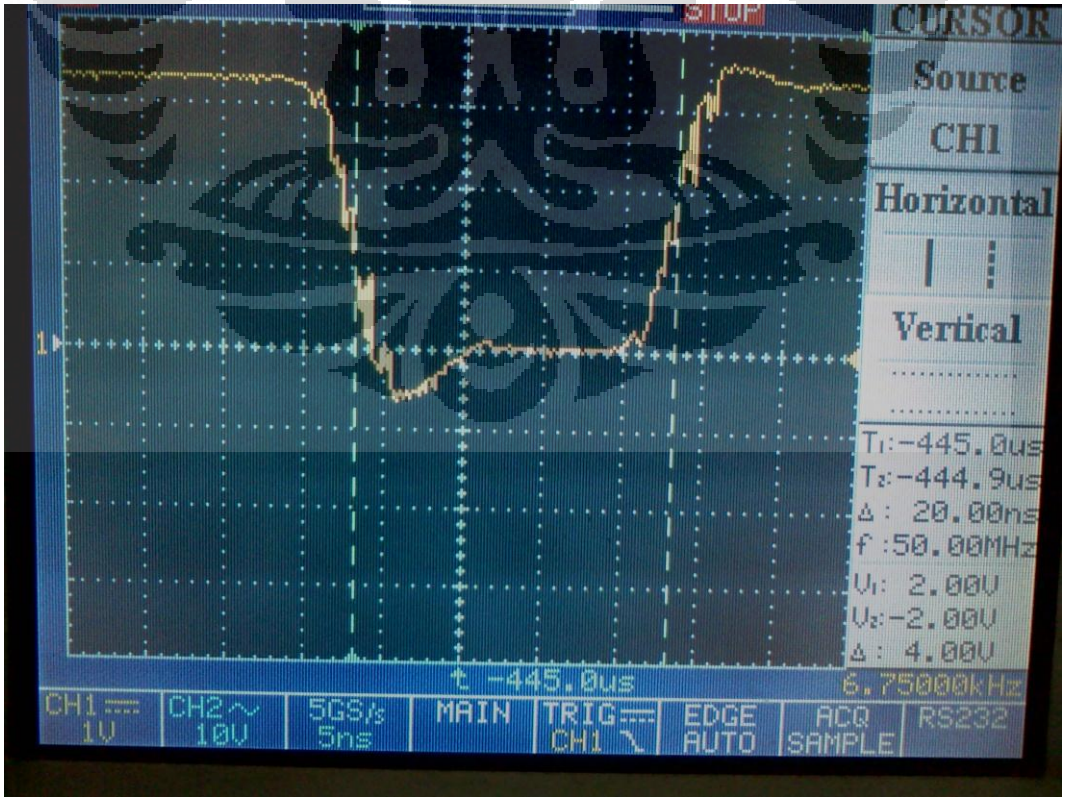
DAFTAR PUSTAKA

Peter J. Ashenden, *The VHDL Cookbook*, 1st ed. Adelaide, South Australia: Dept. Computer Science, University of Adelaide, 1990.

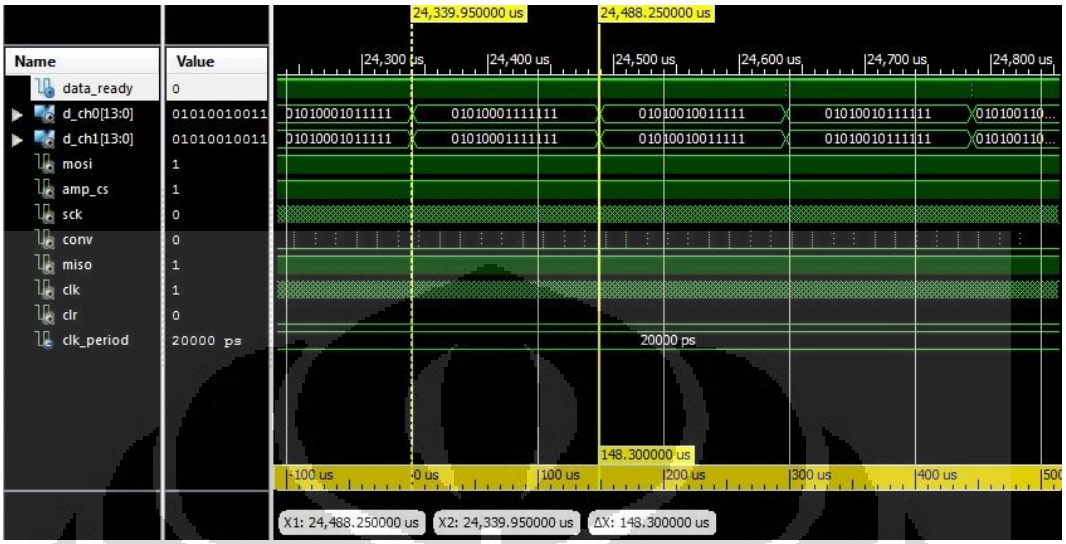
K. Chapman, Amplifier and A/D Converter Control for Spartan-3E Starter Kit, Xilinx Ltd, 2006.

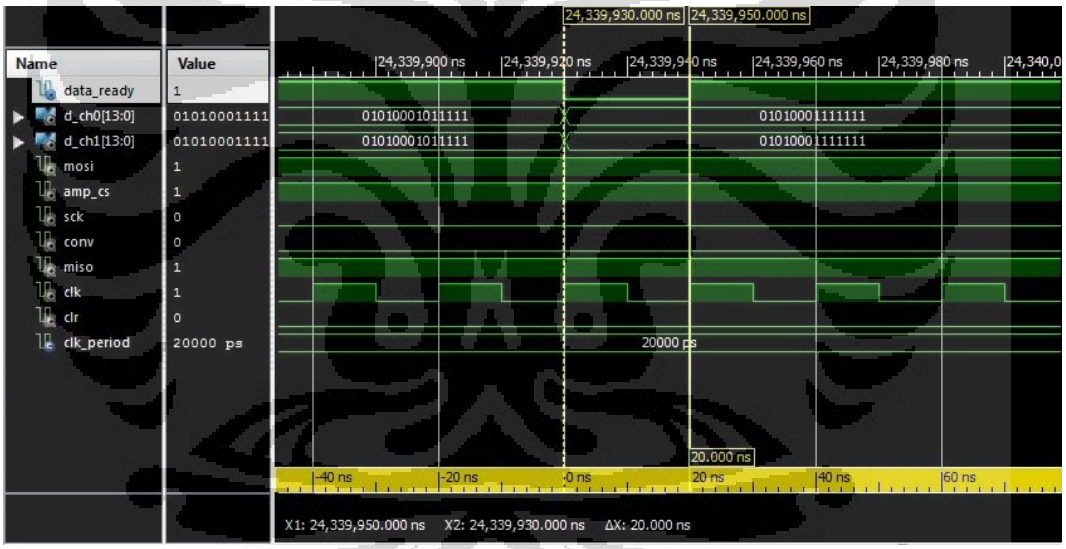
A.1 – Lampiran Gambar Pengambilan Data *Response Time* dan Delay pada Implementasi



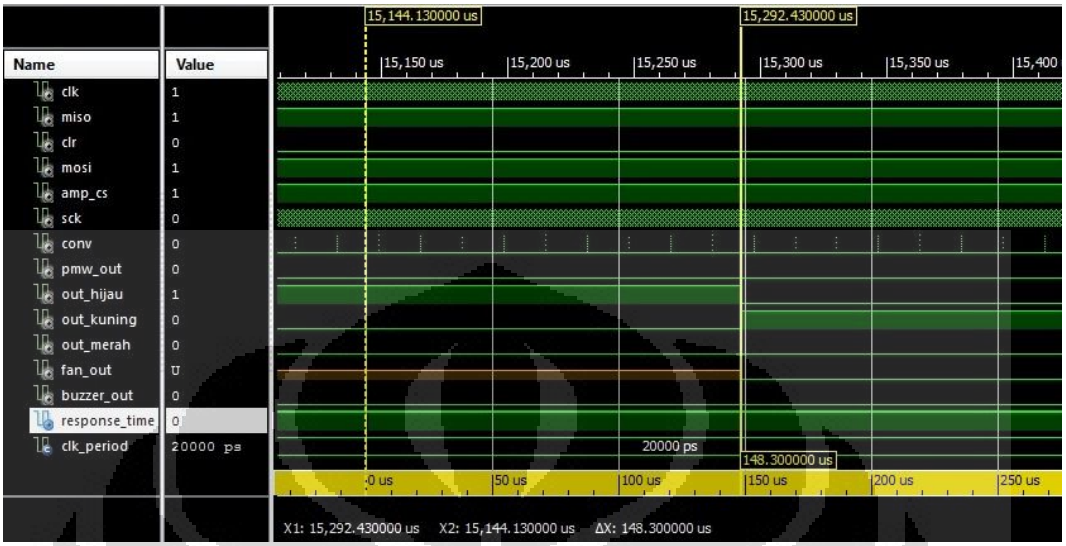


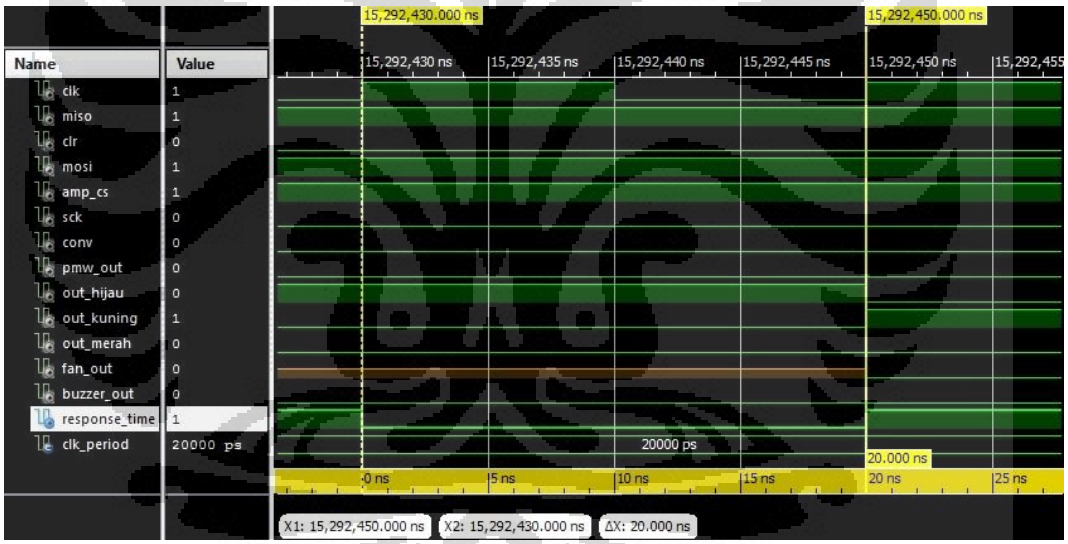
A.2 – Lampiran Gambar Pengambilan Data *Response Time* dan Delay pada Simulasi RTL





A.2 – Lampiran Gambar Pengambilan Data *Response Time* dan Delay pada Simulasi RTL (lanjutan)





A.3 – Lampiran Datasheet Fan Aerocool Shark 14cm

	Power mode	Silence mode
Dimensions	140 x 140 x 25 mm	140 x 140 x 25 mm
Rated Voltage	12V	7V
Starting Voltage	<=6.0V	<=6.0V
Power Consumption	4.23W	1.8W
Rated Current	0.36A	0.15A
Speed	1500 RPM+-10%	800 RPM+-10%
Air Flow	96.5CFM / 45.54 l/s	50CFM / 23.6l l/s
Air Pressure	1.069 mm-H2O	0.305 mm-H2O
Noise	29.6dBA	14.5 dBA
MTBF	100000 hours	100000 hours

A.4 – Lampuran Datasheet Exhaust Fan Shane DJF(a)series

	Shane DJF(a) - 620
Dimensions	620 x 620 x 400 mm
Speed	1400 RPM
Air Flow	5700 m^3/h
Rated Voltage	380V
Input Power	250W