DESAIN DAN IMPLEMENTASI LAMPU LALU LINTAS TERPADU OTOMATIS BERBASIS FPGA Xilinx Spartan II Xc2s50 – PQ208

TUGAS AKHIR

Diajukan Sebagai Syarat Untuk Memperoleh Gelar Sarjana Teknik
Di Jurusan Teknik Elektro Fakultas Teknologi Industri
Universitas Islam Indonesia Yogyakarta

Disusun oleh :

Nama : Rully Alfani

No. Mahasiswa : 02.524.099

JURUSAN TEKNIK ELEKTRO

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

YOGYAKARTA

2007

LEMBAR PENGESAHAN PEMBIMBING

DESAIN DAN IMPLEMENTASI LAMPU LALU LINTAS TERPADU OTOMATIS BERBASIS FPGA Xilinx Spartan II Xc2s50 – PQ208

TUGAS AKHIR

Oleh :

Nama	: **Rully** Alfani
No. Mahasiswa	: 02 524 099

Telah dikonsultasikan dan disetujui oleh pembimbing skripsi
Jurusan Teknik Elektro Universitas Islam Indonesia Yogyakarta

Menyetujui / Mengesahkan

Yogyakarta, April 2007

Pembimbing I,	Pembimbing II,
(Tito Yuwono, ST, MSc.)	(Yusuf Aziz Amrulloh, ST)

LEMBAR PENGESAHAN PENGUJI

DESAIN DAN IMPLEMENTASI LAMPU LALU LINTAS TERPADU OTOMATIS BERBASIS FPGA Xilinx Spartan II Xc2s50 – PQ208

TUGAS AKHIR

Oleh :

Nama : Rully Alfani

No. Mahasiswa : 02 524 099

Telah Dipertahankan Di Depan **Sidang Penguji** Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana Teknik Elektro, Fakultas Teknologi Industri Universitas Islam Indonesia

Tim Penguji,

Tito Yuwono, ST, MSc.
Ketua

Yusuf Aziz Amrulloh, ST
Anggota I

Ir. H. Suyamto
Anggota II

Mengetahui,
Ketua Jurusan Teknik Elektro
Fakultas Teknologi Industri
Universitas Islam Indonesia

(Tito Yuwono, ST, MSc.)

Halaman persembahan

Tugas akhir ini kupersembahkan untuk kedua orang tuaku yang telah memberikan seluruh apa yang dimilikinya demi masa depanku

Ayahanda Tercinta, atas bimbingan, ketauladanan, pengorbanan, kesabaran dan do'a ayahanda adalah motivator utama dalam hidupku.

Ibunda Tersayang, wujud kasih sayangmu, kesabaran dan ketabahan serta keselarasan hidup yang telah ibunda tunjukkan telah mendewasakanku.

Adik-adik ku tercinta (Rino & Risa) yang selalu membuat hidupku begitu bersemangat

Kak Adi Windro yang telah banyak membantu, memberikan ide, dukungan, dan do'a..

MOTTO

" Hai orang-orang yang beriman, jadikanlah sabar dan shalat sebagai penolong mu.

Sesungguhnya Alloh beserta orang-orang yang sabar"

(Q.S. Al Baqarah : 153)

" Sesungguhnya sesudah kesulitan ada kemudahan "

(Q.S. Al Insyiraah : 6)

" Hiduplah seperti pohon besar, tumbuh di tepi jalan. Berbuah lebat, dilempari batu dibalas dengan buah "

(Abu Bakar as Syibli)

"Berbuatlah sesuatu, walaupun sederhana. Untuk membuat orang terdekat mu bahagia dan merasa lebih baik."

(Me....)

KATA PENGANTAR

بسم الله الرحمن الرحيم

Assalamu'alaikum Wr.Wb.

Puji syukur penulis panjatkan kehadirat Allah SWT yang telah melimpahkan segala karunia-Nya, sehingga penulis dapat menyelesaikan Skripsi ini. Shalawat dan salam semoga tercurah pada junjungan Rasulullah Muhammad SAW beserta keluarga, dan pengikutnya.

Dengan dilaksanakannya penelitian dalam bentuk Skripsi ini penulis dapat belajar banyak tentang bahasa pemrograman VHDL, Xilinx dan FPGA khususnya Spartan-2 Xc2s50 PQ208. Melalui Skripsi ini pula penulis dapat menerapkan ilmu-ilmu elektronika digital yang didapat dibangku kuliah dengan kenyataan pada saat melakukan pendesainan IC untuk pengaturan lampu lalu lintas terpadu otomatis sehingga memberikan pengalaman yang sangat berharga.

Selama mengerjakan Tugas Akhir dan dalam penyusunan laporan, tidak lepas dari hambatan, namun berkat motivasi, informasi dan konsultasi dari berbagai pihak, semua masalah dapat diatasi. Untuk itu penyusun menyampaikan rasa hormat sebagai ungkapan terima kasih kepada:

1. Bapak Fathul Wahid, ST, MSc., selaku Dekan Fakultas Teknologi Industri Universitas Islam Indonesia.
2. Bapak Tito Yuwono, ST, MSc. Selaku ketua jurusan Teknik Elektro dan Dosen pembimbing tugas akhir yang telah banyak memberikan bimbingan dan dukungan moril dalam pelaksanaan tugas akhir ini.
3. Bapak Yusuf Azis Amrulloh, ST Selaku Dosen pembimbing tugas akhir yang telah membantu memberikan ide dan informasi yang diperlukan untuk penyusunan tugas akhir ini.
4. Papa dan Mama yang selalu memberikan do'a dan dukungan moril sehingga tugas akhir ini dapat selesai.
5. Saudara-saudaraku tercinta : Bang Ino dan Dek Ica yang juga senantiasa memberikan semangat dan doa.

6. Kak Adi yang selalu memberikan arahan, membantu memberikan ide dan dukungan dalam menyelesaikan tugas akhir ini.
7. Cocoh Robert Kristanto yang selalu memberikan dukungan dan do'a.
8. Daru dan didi yang telah banyak membantu dan memberikan ide.
9. Semua keluarga besar di Palembang dan di Lampung yang selalu memberikan do'a dan semangat.
10. Teman-teman angkatan 2002 : April, Dini, Daru, Didi, dan Juna serta semua rekan-rekan elektro UII pada umumnya yang telah membantu dalam penyusunan tugas akhir ini.
11. Mas Yogi, Mas Tri, dan Mas Heri yang telah banyak membantu dan membimbing, sehingga terselesaikannya tugas akhir ini.
12. Sahabat-sahabatku di Palembang (Ika H., Ika W., serta YURIPOD Gank) terima kasih atas support dan do'anya.
13. Teman-teman Gamers di Pekanbaru (Robert, Neka, Yose, Aaf, Nelson, Cucu).
14. Kos Al-Hikmah (mb ayu, mb eva, mb eka, mb ency, mb devi, mb oce, dek depi, dek tia, dek titi, dek muj2, dan dek nty) yang telah banyak membantu dan sebagai teman curhat.
15. Pihak-pihak yang tidak mungkin penulis sebut satu persatu

Semoga Allah SWT memberikan balasan limpahan rahmat dan karunia serta kelapangan hati atas segala kebaikan yang mereka berikan.

Penulis menyadari bahwa Tugas Akhir ini masih banyak terdapat kekurangannya, untuk itu sangat diharapkan saran dan kritik yang sekiranya dapat menambah pengetahuan serta lebih menyempurnakan Tugas Akhir ini. Semoga apa yang telah penulis ketengahkan ini dapat bermanfaat bagi kita semua.

Wasalamu'alaikum Wr.Wb

Yogyakarta, April 2007

Rully Alfani

ABSTRAKSI

FPGA (*Field Programmable Gate Array*) adalah suatu piranti yang berguna untuk merancang berbagai aplikasi dengan memanfaatkan gerbang logika dasar dan mengujinya sebelum diproduksi. Bahasa yang digunakan adalah VHDL (VHSIC *Hardware description language*); VHSIC singkatan dari *Very High Speed Integrated Circuit*.Tujuan dari skripsi ini adalah merancang suatu kalkulator sederrhana sebagai alat kalkulasi yang dapat digunakan sesuai dengan kebutuhan.

Sistem kerja lampu lalu lintas terpadu otomatis ini adalah ditampilkannya penampil waktu pada saat lampu merah, hijau, dan kuning menyala untuk setiap jalur pada empat persimpangan jalan.

Dari kesimpulan pengujian diketahui bahwa peralatan dapat berjalan dengan baik. Dilihat perpindahan lampu dari lampu merah ke hijau 25 detik, hijau ke kuning 20 detik, kuning ke merah 5 detik.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pesatnya perkembangan teknologi saat ini mengakibatkan meningkatnya pembangunan sarana dan prasarana di bidang transportasi, sehingga menyebabkan semakin banyaknya kendaraan berlalu lintas. Hal ini memicu tingkat kepadatan lalu lintas dan rawannya kecelakaan di persimpangan jalan. Untuk itu diperlukan pengaturan lalu lintas terpadu yang memadai terutama di persimpangan-persimpangan jalan dengan memperhatikan tingkat kepadatan di tiap ruas jalan supaya tidak menimbulkan kemacetan, yang dapat mengurangi kenyamanan berlalu lintas serta mengurangi kecelakan di persimpangan jalan.

Untuk mengurangi tingkat kemacetan dan mengurangi tingkat kecelakaan pada setiap ruas jalan, maka diperlukan suatu pengendalian lampu lalu lintas terpadu secara otomatis di persimpangan jalan. Pengendalian lampu lalu lintas terpadu otomatis ini dapat dilakukan dengan memakai sistem elektronik rangkaian terpadu dengan menggunakan FPGA *(Field Programmable Gate Arrays)* dan menampilkan pewaktu pada setiap lampu hijau, kuning, dan merah menyala dengan menggunakan *seven segment* 2 digit, khususnya dalam penelitian ini digunakan FPGA Xilinx Spartan II Xc2s50 – PQ208.

Dengan pemakaian FPGA ini maka pengaturan lampu lalu lintas dapat dilakukan berdasarkan pembagian waktu pada kondisi jalan dengan memperhatikan tingkat kepadatan lalu lintas pada tiap persimpangan. Pembagian

waktu pada kondisi jalan terutama untuk jalur padat akan diberikan waktu yang lebih lama dibandingkan jalur sepi. Dengan demikian diharapkan adanya kenyamanan bagi semua pengguna jalan dan mengurangi tingkat kemacetan serta mengurangi dampak kecelakaan.

Dari hasil pengamatan inilah akan dibuat suatu alat yang dapat mengurangi tingkat kemacetan dan kecelakaan pada persimpangan jalan dengan menggunakan FPGA Xilinx Spartan II Xc2s50 – PQ208 (Lampu Lalu Lintas Terpadu).

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan diatas, maka dapat diambil suatu rumusan masalah sebagai berikut ; "Bagaimana membuat suatu piranti FPGA Xilinx Spartan II Xc2s50 – PQ208 yang dapat dimanfaatkan sebagai pengaturan lampu lalu lintas terpadu secara otomatis".

1.3 Batasan Masalah

Agar permasalahan yang dibahas dalam laporan skripsi ini tidak menyimpang dari judul yang telah ditetapkan maka perlu ditetapkan pokok-pokok permasalahan yang akan dibahas, maka penulisan dibatasi pada masalah :

a. Penelitian difokuskan pada pembuatan Lampu Lalu Lintas Terpadu Otomatis baik *hardware* maupun *software*.
b. Penampil dari Lampu Lalu Lintas yang menggunakan *seven segment* 2 digit pada saat lampu hijau, lampu kuning, dan lampu merah menyala.

c. Pembuatan *software* dalam bahasa Asembler agar dapat mengatur sistem sehingga dapat bekerja sesuai fungsinya.

d. Pengujian sistem apakah bekerja sesuai dengan keinginan.

1.4 Tujuan Penelitian

Tujuan dari pembuatan alat ini adalah mempermudah pengguna jalan untuk mengetahui seberapa lama lagi lampu hijau,lampu kuning, dan lampu merah menyala, sehingga mengurangi tingkat kecelakaan dan kemacetan berlalu lintas pada persimpangan jalan.

1.5 Sistematika Penulisan

Untuk memudahkan sistematika penulisan, penulis membagi dalam beberapa bab pembahasan dengan urutan sebagai berikut:

BAB I PENDAHULUAN

Bab ini akan menguraikan latar belakang dan alasan pemilihan judul, tujuan penulisan, pembatasan masalah, metodologi, dan sistem penulisan.

BAB II LANDASAN TEORI

Bab ini berisi teori dan rumusan yang merupakan acuan perancangan dan pembuatan rangkaian yang meliputi penjelasan tentang gambaran alat, sifat, karakteristik dan kegunaan komponen-komponen pendukung.

BAB III PERANCANGAN SISTEM

Bab ini berisi mengenai perancangan perangkat keras dan perangkat lunak keseluruhan sistem.

BAB IV ANALISA DAN PEMBAHASAN

Bab ini menguraikan proses pengujian, pengukuran alat, analisa dan pengamatan dari alat alat yang dibuat.

BAB V PENUTUP

Bab ini berisi kesimpulan dan saran-saran sebagai masukkan untuk penunjang pembuatan alat tersebut.

DAFTAR PUSTAKA

LAMPIRAN

BAB II

LANDASAN TEORI

2.1. *Field Programmable Gate Array* (FPGA)

FPGA merupakan sebuah piranti digital yang dapat diprogram untuk merepresentasikan sistem logika yang telah dirancang. Teknologi *Integrated Circuit* (IC) FPGA diperkenalkan pada tahun 1985 oleh perusahaan semikonduktor Xilinx. FPGA adalah sebuah konsep teknologi IC yang dapat diprogram dan dihapus seperti halnya *Random Access Memory* (RAM). FPGA kemudian berkembang pesat, baik dari segi kepadatan gerbang, kecepatan dan disertai dengan penurunan harga jual.

Penemuan FPGA telah membuat peningkatan yang pesat akan pembuatan prototipe beberapa sistem digital. Salah satu produsen FPGA yang ada di pasaran adalah Xilinx, disamping produsen lainnya Actel dan Altera. Prinsip dasar dari pemrograman atau pengkonfigurasian FPGA Xilinx adalah pengubahan gambar rangkaian elektronik digital dari perangkat lunak Xilinx yang berupa file aliran bit (*bitstream*) dan di konfigurasi (di *download*) ke dalam IC FPGA Xilinx tersebut sehingga IC tersebut terkonfigurasi secara perangkat keras yang dirancang dalam perangkat lunak Xilinx. FPGA produk Xilinx sudah melewati beberapa generasi antara lain XC2000, XC3000 dan XC4000. Tiap generasi memiliki sifat dan gerbang logika, jumlah *Configurable Logic Blocks* (CLB) dan jumlah *Input/Output Blocks* (IOB).

Keuntungan-keuntungan yang dimiliki FPGA sehingga disukai oleh para penggunanya antara lain adalah :

a. FPGA selalu dapat diprogram kembali sehingga memudahkan modifikasi tanpa harus merubah keseluruhan sistem.
b. Sebuah rancangan secara otomatis dapat diubah dari level logika gerbang menjadi struktur *layout* dengan fasilitas yang dimilikinya, sehingga perubahan dapat dilakukan dengan mudah tanpa harus merubah rancangan awal.
c. Simulasi hasil desain dapat dilakukan pada keluaran gerbang yang terpakai dan pada karakteristik pewaktuan yang dimiliki oleh desain yang dibuat. Hal ini sangat menguntungkan ketika waktu juga menjadi faktor yang harus diperhatikan dalam desain yang dibuat.
d. IC FPGA keluaran terbaru mempunyai jumlah gerbang yang semakin banyak dengan fasilitas yang semakin lengkap.

2.1.1. FPGA Keluarga Xilinx Spartan II

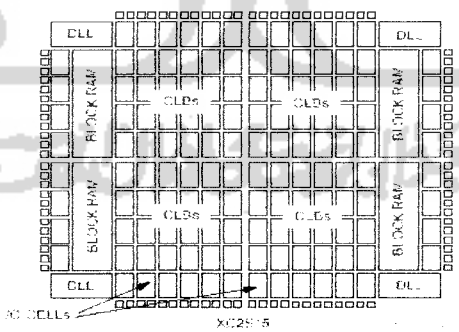
Spartan II merupakan slah satu keluarga FPGA yang dikeluarkan oleh Xilinx. Xilinx merupakan salah satu pabrik pembuat FPGA yang cukup terkenal. Keluarga Spartan II merupakan keluarga FPGA yang memiliki 15.000 sampai 20.000 gerbang. IC Xilinx ini dapat deprogram dan dihapus dengan waktu yang tidak terbatas. Keluarga Spartan ini dapat diprogram dengan mudah menggunakan Xilinx *Development System* ataupun dengan *Development System* yang lain yang dikembangkan oleh para pengguna.

Tabel 2.1 Data Keluarga Spartan II

Device	Logic Cells	System Gates (Logic dan RAM)	CLB Array (R x C)	Total CLBs	Maximum Available User I/O	Total Distributed RAM Bits	Total Block RAM Bits
XC2S15	432	15.000	8 x 12	96	86	6.144	16K
XC2S30	972	30.000	12 x 18	216	132	13.824	24K
XC2S50	1.728	50.000	16 x 24	384	176	24.576	32K
XC2S100	2.700	100.000	20 x 30	600	196	38.400	40K
XC2S150	3.888	150.000	24 x 36	864	260	55.296	48K
XC2S200	5.292	200.000	28 x 42	1.176	284	75.264	56K

2.1.1.1. Struktur Dasar Keluarga Spartan II

Suatu piranti FPGA terdiri atas CLB, unit input/output, *Delay-Locked Loops* (DLLs), unit RAM dan unit routing yang dapat diprogram secara otomatis penuh. Susunan dan letak masing-masing bagian tersebut dapat dilihat pada Gambar 2.1



Gambar 2.1 Blok diagram dasar keluarga Spartan II

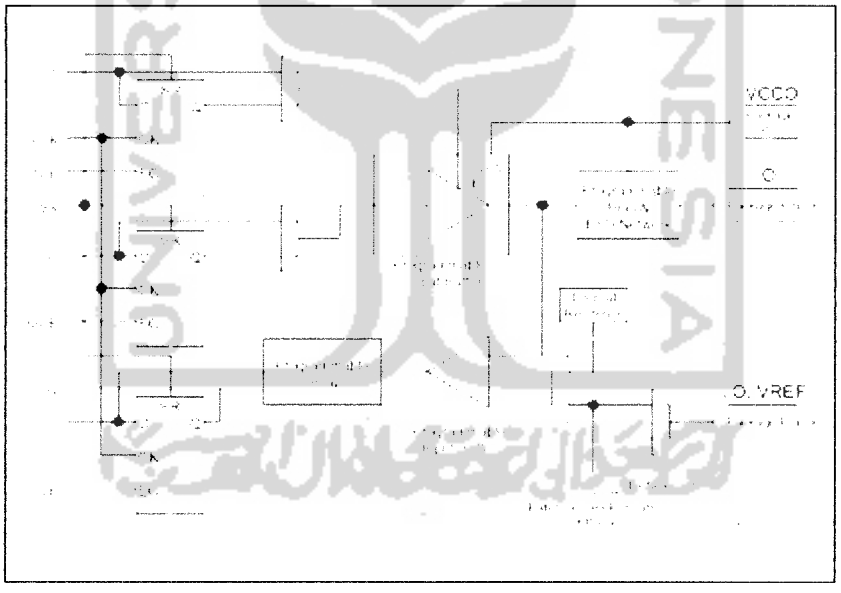
Struktur dasar CLB terdiri dari RAM dan fungsi logika dasar. DLL digunakan untuk mengatur *clock*, perkalian *clock* dan pembagian *clock*. Pengaturan *clock*

dapat dilakukan secara *eksternal* (*board level)* dan *internal (chip level).* Memorinya terdiri dari 4 K bit yang dapat dikonfigurasi dari 1 bit ke 16 bit. Sedangkan untuk pemilihan I/O mengikuti standar I/O untuk diimplementasikan ke dalam *chip* ke *chip*, *chip* ke memori dan *chip* ke *interface.*

2.1.1.2. Konfigurasi Blok FPGA Spartan II

Konfigurasi blok-blok yang termuat dalam FPGA Spartan II adalah :

a. *Input/Output Blocks* (IOB)



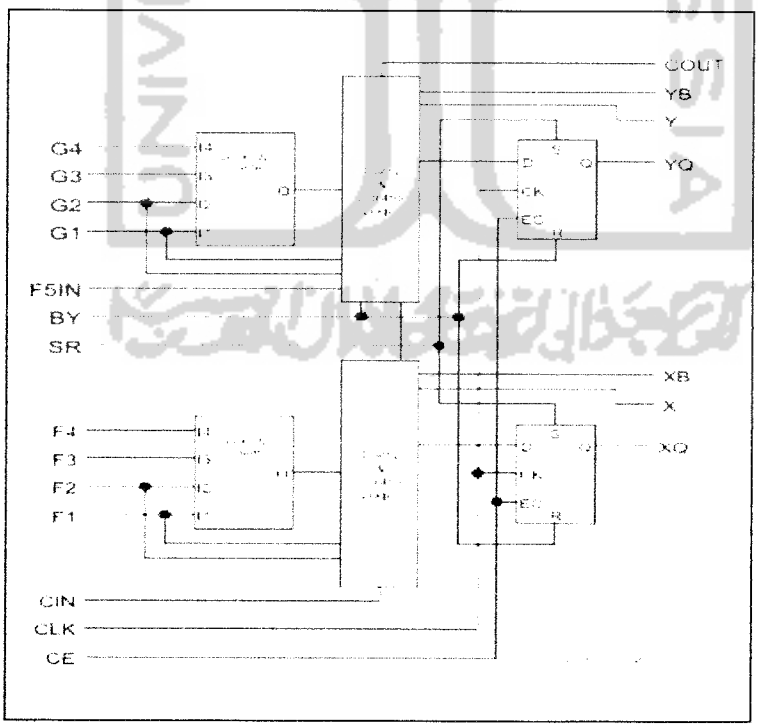
Gambar 2.2 Blok diagram I/O Spartan II

Input/Output Blocks merupakan bagian dari FPGA yang berfungsi menghubungkan FPGA dengan piranti lain yang terkoneksi. IOB keluarga Spartan

II mampu bekerja pada berbagai macam standar I/O seperti TTL, CMOS, dan PCI. Kemampuan untuk menyesuaikan dengan berbagai macam I/O didukung dengan kemampuan tiap *pad* I/O untuk ditambahi *pull-up* dan *pull-down resistor.*

Bagian *buffer* pada Spartan II IOB *input path* akan menghubungkan sinyal *input* yang masuk secara langsung dengan logika internal atau secara tidak langsung melalui *input flip-flop* optional. Sedangkan bagian *output path* termasuk *buffer* 3 keadaan akan men-*drive* sinyal output menuju *pad output*. Sama dengan sinyal *input*, sinyal *output* ini dapat dihubungkan dengan *pad output* melalui logika *internal* maupun melalui *output flip-flop optional.*

b. *Configurable Logic Blocks* (CLB)



Gambar 2.3 CLB pada Spartan III

CLB merupakan bagian dari FPGA yang berfungsi merubah logika-logika terprogram yang dimasukkan menjadi fungsi-fungsi yang dipahami oleh FPGA dan dapat bekerja sesuai dengan program yang diinginkan. CLB Spartan II terdiri atas *Logic Cell* (LC) sebagai bangunan utama. Sebuah LC terdiri atas 4 buah *input* yang akan membangkitkan fungsi logika yang diinginkan, *carry logic* dan elemen penyimpan. Keluaran setiap LC akan men-*drive* keluaran CLB dan masukan pada D *flip-flop*. Setiap CLB pada Spartan II terdiri atas 4 LC yang tersusun dalam 2 *slices* yang identik. Tiap CLB ini juga memuat logika yang akan mengkombinasi generator pembangkit fungsi logika untuk 4 sampai 6 input.

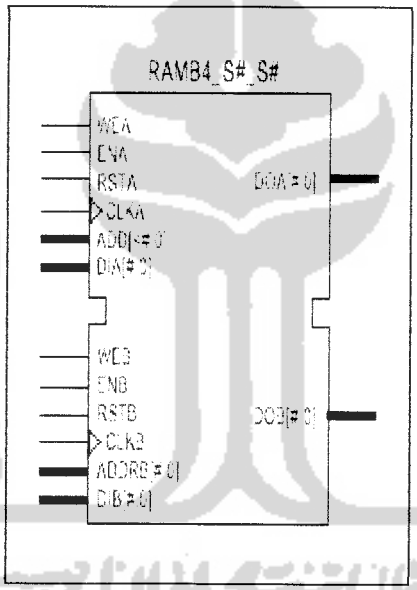
Generator pembangkit fungsi diimplementasikan dalam sebuah *look-up tables* (LUT) 4 *input*. LUT ini juga dapat membangkitkan sebuah RAM 16 x 1 sinkron serta membangkitkan fungsi *shift register* 16 bit. Fungsi RAM yang dibangkitkan generator ini akan melengkapi *block* RAM yang dimiliki oleh Spartan II sehingga mampu menghasilkan unit penyimpan data yang handal.

c. *Delay-Loocked Loops* (DLLs)

Masing-masing *clock* input secara menyeluruh dikelompokkan dengan penguatan, maksudnya adalah pemenuhan digital DLL dapat menghilangkan kemiringan antara titik *clock input* dan *pin clock internal* yang dilalui oleh *hardware*. Tiap-tiap DLL dapat digerakkan oleh 2 buah jaringan *clock* secara menyeluruh. Pemantauan DLL dapat dilakukan dari *clock* masukan dan aliran *clock* serta penentuan tunda *clock* secara otomatis. Penambahan tunda yang

diperkenalkan sebagaimana beberapa *clock* yang diraih secara pasti oleh sebuah *flip-flop* internal salah satunya setelah periode *clock* yang masing-masing melaju terhadap input. Sistem tertutup ini secara efektif akan menghilangkan aliran *clock* yang tertunda, dimana hal tersebut dikerjakan oleh pembawa sistem yang mana setiap c*lock* yang tersisa tepatnya dalam *flip-flop internal* akan diserempakkan dengan *edges clock.*

d. *Blocks Random Access Memory* (RAM)



Gambar 2.4 Blok RAM spartan-II

Konfigurasi FPGA spartan II sangat luas dan memiliki memori 4 K bit. Tiap-tiap blok RAM akan menempati CLB serta tiap-tiap blok dapat dikonfigurasi pada perbandingan diantara 4K x 1 dan 256 x 16 bit.

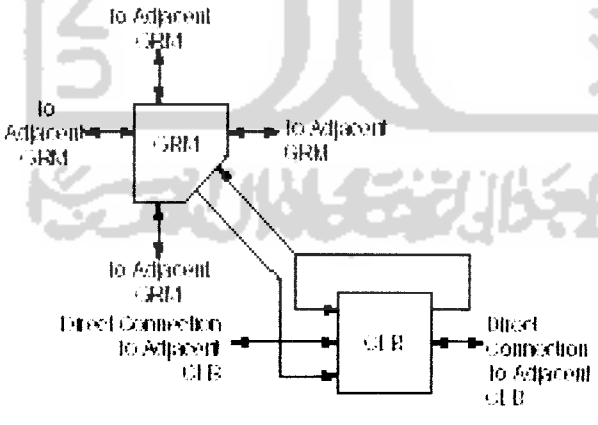
2.1.2. *Programmable Routing Matrix*

Programmable Routing Matrix merupakan cara sebuah FPGA melakukan *routing* menghubungkan CLB-CLB dan IOB-IOB yang digunakan dalam desain menjadi satu kesatuan sistem. *Routing* ini dilakukan secara otomatis penuh. Namun untuk keperluan tertentu, optimasi jalur yang paling pendek dapat dilakukan *routing* manual.

Dalam keluarga Spartan II ada beberapa macam *routing* yang bisa digunakan yaitu:

a. Local Routing

Local Routing digunakan untuk mengimplementasikan hubungan antara LUT, *flip-flop* dan *General Routing Matrix* (GRM), antara jalur umpan balik internal CLB dengan LUT lain pada CLB yang sama untuk koneksi *high speed*, serta antara jalur-jalur langsung yang bisa dibuat untuk memperkecil *delay*.



Gambar 2.5 Struktur *local routing*

b. General Purpose Routing

General Purpose Routing digunakan untuk melakukan koneksi vertikal dan horizonal antar kolom dan baris CLB yang digunakan.

c. I/O Routing

I/O Routing khusus digunakan untuk menghubungkan *array* pada CLB dengan IOB.

d. Dedicated Routing

Dedicated Routing digunakan untuk menguhubungkan beberapa CLB, IOB ataupun LUT yang memerlukan perlakuan khusus untuk memaksimalkan performa.

Gambar 2.6 Koneksi BUFT untuk *Dedicated Horizontal Bus Line*

e. Global Routing

Global Routing digunakan untuk menghubungkan *clock* dengan bagian yang membutuhkan serta sinyal-sinyal dengan *fan-out* yang tinggi ke bagian lain.

2.1.3. Mode Operasi

Keluarga Spartan II dapat dioperasikan dalam 4 mode yaitu:

a. *Slave Serial mode*

b. *Master Serial mode*

c. *Slave Parallel mode*

d. *Boundary Scan mode.*

Mode yang paling mudah digunakan adalah *Boudary Scan Mode* dimana tidak diperlukan koneksi-koneksi khusus, cukup menggunakan kabel paralel yang dikoneksikan menggunakan JTAG, maka desain dapat dengan mudah diimplementasikan ke dalam FPGA.

2.2. *Board* Pegasus

Modul Pegasus dikeluarkan oleh pengmbang ke 3, yaitu DILIGENT. Dengan IC FPGA utama yaitu, Xilinx Spartan II XC2S50, dengan perangkat lunak Xilinx. Perangkat pendukung yang terdapat pada modul Pegasus diantaranya :

a. 50k-gate Xilinx Spartan II FPGA, dengan 50k gerbang dasar dan 200MHzoperasi *maximum*.

b. XCF01S Xilinx Flash ROM.

c. Berbagai macam I/O, termasuk di dalamnya 8 LED, empat *seven-segment*, empat saklar *pushbutton* dan delapan saklar geser.

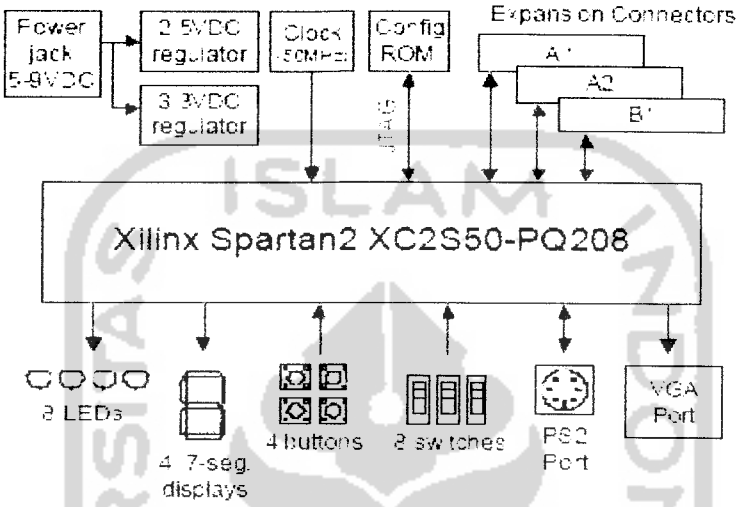
d. 50MHz *oscilator* dan satu pin untuk *oscilator* tambahan.

e. PS/2 dan VGA port.

f. *Pin* 96 I/O dibagi dalam 3 bagian/port, yaitu : A1, A2 dan B1 yang masing-masing terdiri dari 40 *pin*.

g. Seluruh I/O *pin* sudah dilengkapi pengaman.

h. *Port* pemrogram mode JTAG.



Gambar 2.7 Blok diagram Pegasus

Modul Pegasus sudah dapat dibilang sangat komplit karena hanya dengan menggunakan modul ini, sudah dapat men*download design* dan menjalankannya tanpa perlu menambah komponen lain, karena telah tersedia beberapa masukan dan keluaran.

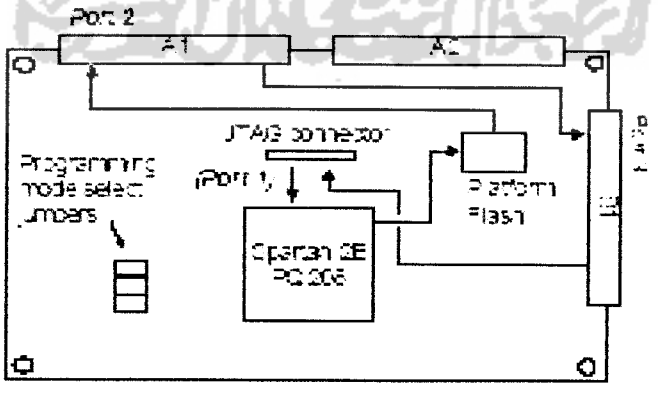
a. *Port* JTAG dan Pengkonfigurasi Modul

Pada modul Pegasus ini selain terdapat IC FPGA utama (Spartan II XC2S50) juga terdapat IC FPGA *secondary* yaitu Spartan II XCF01S yang berfungsi sebagai IC *Flash* ROM, sehingga apabila ada suatu modul yang dapat diprogram

(terdapat IC FPGA) yang terhubung ke modul Pegasus ini, maka dapat dikonfigurasikan melalui *port* JTAG (*port*1), dari modul utama.

Dengan menggunakan *port* JTAG dapat diketahui IC tipe, jenis dan dari keluarga FPGA yang ada pada modul utama maupun pada modul lain (tambahan) yang terhubung ke modul utama melalui *port* tambahan secara *automatic scan chain.*

Scanning dengan menggunakan JTAG sangat mudah karena perangkat lunak Xilinx melakukannya secara otomatis, Xilinx mencari melalui *port* JTAG dan membaca IC FPGA utama, kemudian apabila tidak ada modul lain yang terhubung pada *port* JTAG tambahan *port2* (A1) atau *port3* (B1) maka *buffer* pada modul Pegasus menghilangkan keberadaan *port* tambahan tersebut, sedangkan jika ada modul (memilik IC FPGA) maka *buffer* akan menyatakan bahwa ada modul yang terhubung. Saat *scanning port* JTAG membaca FPGA utama, ke flash ROM (XCF01S), lalu membaca modul yang terhubung melalui *port* tambahan tersebut, maka dapat dikonfigurasi secar bersamaan melalui 1 kabel utama, baik itu *port* USB, *Parallel* maupun *Ethernet*.



Gambar 2.8 Aliran scan JTAG pada Pegasus

b. Catu Daya

Modul Pegasus memerlukan 5Vdc catu daya, sedangkan untuk masukan/keluaran dapat menggunakan *power* dari luar 3Vdc-5Vdc. Catu daya ini juga dipergunakan untuk mengaktifkan 8 LED, 4 display *seven-segment*, sebagai masukan +5Vdc untuk saklar *pushbutton* dan saklar geser serta sumber *clock internal* dan *power port* tambahan.

c. Oscilator

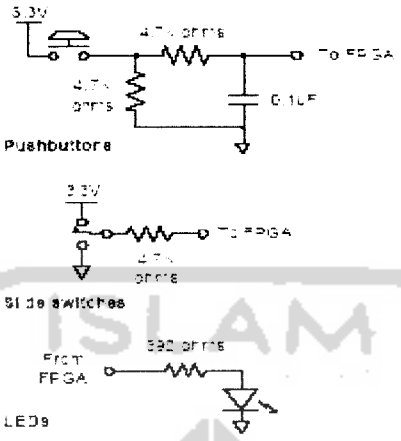
Modul Pegasus menyediakan 50MHz *oscilator* sebagai *clock* utama untuk aplikasi-aplikasi yang akan dikonfigurasi ke Spartan II XC2S50. *Oscilator* tersebut terhubung langsung ke Spartan II XC2S50 (*pin*77).

d. Saklar *Pushbutton*, saklar geser indikator LED

Empat saklar *pushbutton* dan 8 saklar geser disediakan sebagai masukan. Saklar *pushbutton* dalam keadaan normal menghasilkan 0Vdc (rendah), dan akan berubah menjadi 3-5Vdc (tinggi) ketika saklar *pushbutton* ditekan. Saklar geser menghasilkan rendah (0Vdc) atau tinggi (3Vdc-5Vdc) secara tetap tergantung dari posisinya. Saklar *pushbutton* sebagai masukan menggunakan rangkaian RC sebagai pengaman dan agar menghasilkan masukan yang stabil, sedangkan saklar geser hanya terhubung dengan resistor secara serial.

Delapan LED disediakan sebagai keluaran anoda LED terhubung ke *pin* keluaran melalui resistor 390 Ω, sedangkan katoda LED terhubung langsung ke

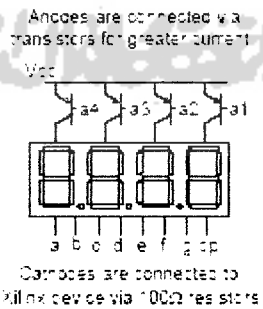
Ground. LED 9 digunakan sebagai *indikator power* FPGA dan LED ke 10 digunakan sebagai *indicator* status pemrograman JTAG.



Gambar 2.9 Rangkaian saklar *pushbutton*, saklar geser, LED

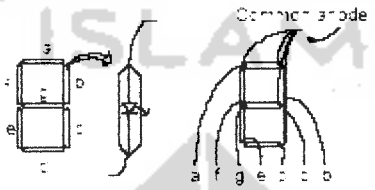
e. *Seven Segment*

Pada modul Pegasus terdapat 4 digit *common* anoda *seven-segment*. *Display seven-segment* dikonfigurasikan secara *multiplexer*, jadi hanya ada 7 masukan katoda untuk mengaktifkan 28 katoda pada 4 digit *seven-segment*. Empat digit selektor *enable* berfungsi sebagai pengatur *digit* pada *seven-segment.*



Gambar 2.10 *Common anode seven-segment* 4 *digit*

Ketujuh anoda dari 4 *seven-segment* saling tersambung ke dalam selektor "*common anode*", *display* ini memiliki 4 selektor yang di namakan AN0 – AN3, apabila ada sinyal atau pulsa yang mengaktifkan selektor ini maka *digit* dari selektor tersebut akan aktif. Sinyal/pulsa yang digunakan adalah sinyal/pulsa rendah (0Vdc). Katoda dari setiap *seven-segment* terhubung ke dalam 7 keluaran, dan diberi nama CA – CG, dan akan aktif apabila ada sinyal/pulsa rendah (0Vdc).



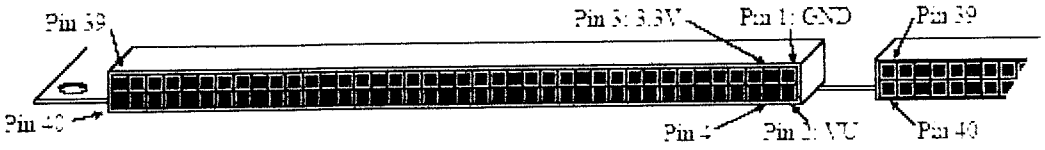
Gambar 2.11 *Common anode seven-segment* 1 *digit*

Dilihat dari diagram yang terlihat maka menghasilkan *display seven-segment multiplexer*, yang mana bila anoda atau selektor (AN0 – AN3) diaktifkan maka *digit* tersebut yang akan aktif dan mendapatkan sinyal/pulsa katoda (CA – CG). Jika dilakukan secara terus menerus dan bila benar maka ke-4 *digit* akan terlihat aktif /seolah-olah semua aktif secara bersamaan. Jika diberi sinyal rendah (0Vdc) secara terus menerus 1ms – 16ms, maka akan terlihat semua *digit* aktif. Dengan syarat, bersamaan dengan *digit enable* data untuk *digit* tersebut harus ada. *Refresh frekuency* berkisar 60Hz – 1KHz.

f. *Port* I/O Tambahan

Pada modul Pegasus terdapat 3 *port* tambahan (A1, A2, dan B3) dengan masing-masing *port* terdapat 40 *pin*. Masing-masing *port* memiliki GND pada *pin*

1, VU pada *pin* 2, dan 3Vdc-5Vdc pada *pin* 3. Untuk *pin* 4-35 merupakan *pin* sinyal I/O, dan *pin* 36-40 digunakan sebagai *port* JTAG tambahan, atau juga bisa digunakan sebagai *clock* tambahan.



Gambar 2.12 *Pin* penghubung tambahan

Tabel 2.2 Konfigurasi *pin* yang terdapat pada modul Pegasus

Pegasus Expansion Connector Pinout								
Connector B1			Connector A1			Connector A2		
Pin	Signal	B1	Pin	Signal	A1	Pin	Signal	A2
39	TDO	TDO	39	TDO	TDO	39	GND	GND
40	TD	TDI	40	TD	TDI	40	GND	GND
37	TMS	TMS	37	TMS	TMS	37	P-I	P-I
38	TCK	TCK	38	TCK	TCK	38	P-I	P-I
36	MB-INT	B0	36	MA-INT	A0	36	MAD-INT	A0
35	GND	GND	35	GND	GND	35	Not Used	P-I
34	MB-WAIT	B1	34	MA-WAIT	A1	34	MAD-WAIT	A1
33	MB-RST	B2	33	MA-RST	A2	33	MAD-RST	A2
32	MB-BOOTB	B3	32	MA-BOOTB	A3	32	MAD-BOOTB	A3
31	MB-WRIT	B4	31	MA-WRIT	A4	31	MAD-WRIT	A4
30	MB-DBT	B5	30	MA-DBT	A5	30	MAD-DBT	A5
29	MB-AUTB	B6	29	MA-AUTB	A6	29	MAD-AUTB	A6
28	MB-DB8	B7	28	MA-DB8	A7	28	MAD-DB8	A7
27	MB-DB9	B8	27	MA-DB9	A8	27	MAD-DB9	A8
26	MB-DB10	B9	26	MA-DB10	A9	26	MAD-DB10	A9
25	MB-DB11	B10	25	MA-DB11	A10	25	MAD-DB11	A10
24	MB-DB12	B11	24	MA-DB12	A11	24	MAD-DB12	A11
23	MB-DB13	B12	23	MA-DB13	A12	23	MAD-DB13	A12
22	MB-DB14	B13	22	MA-DB14	A13	22	MAD-DB14	A13
21	MB-DB15	B14	21	MA-DB15	A14	21	MAD-DB15	A14
20	P-IOCS16	B15	20	P-IOCS16	A15	20	P-IOCS16	A15
19	P-IOR	B16	19	P-IOR	A16	19	P-IOR	A16
18	P-IOW	B17	18	P-IOW	A17	18	P-IOW	A17
17	P-DB8	B18	17	DB8	A18	17	P-IO14	A18
16	P-DB7	B19	16	DB7	A19	16	P-IO15	A19
15	P-DB6	B20	15	DB6	A20	15	P-IO12	A20
14	P-DB5	B21	14	DB5	A21	14	P-IO13	A21
13	P-DB4	B22	13	DB4	A22	13	P-IO10	A22
12	P-ADR0	B23	12	ADR0	A23	12	P-IO11	A23
11	P-DB3	B24	11	DB3	A24	11	P-IO8	A24
10	P-ADR1	B25	10	ADR1	A25	10	P-IO9	A25
9	P-DB2	B26	9	DB2	A26	9	P-IO6	A26
8	P-ADR2	B27	8	ADR2	A27	8	P-IO7	A27
7	P-DB1	B28	7	DB1	A28	7	P-IO4	A28
6	P-ADR3	B29	6	ADR3	A29	6	P-IO5	A29
5	P-DB0	B30	5	DB0	A30	5	P-IO2	A30
4	P-ADR4	B31	4	ADR4	A31	4	P-IO3	A31
3	+5V	+5V	3	+5V	+5V	3	+5V	+5V
2	P-ADR5	B32	2	ADR5	A32	2	P-IO1	A32
1	GND	GND	1	GND	GND	1	GND	GND
2	+5V	+5V	2	+5V	+5V	2	+5V	+5V

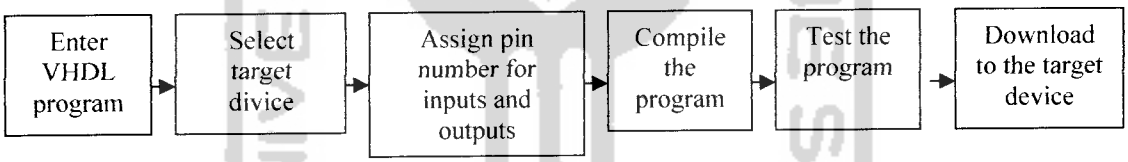
Tabel 2.3 *Port accessory*

Accessory Port Pinout					
Pin	Name	FPGA Pin	Pin	Name	FPGA Pin
1	ADC	P45	6	ADC3	P47
2	ADC1	P46	7	GND	-
3	ADC2	P61	8	Vcc	-

2.3. *VHSIC Hardware Description Language* (VHDL)

VHDL adalah sebuah bahasa dengan banyak konstruksi. VHDL dapat mendeskripsikan seluruh sistem sampai kepada gerbang dasar. Dalam VHDL dimungkinkan untuk membuat sebuah sistem dengan perilaku sama, tetapi menggunakan struktur yang berbeda.

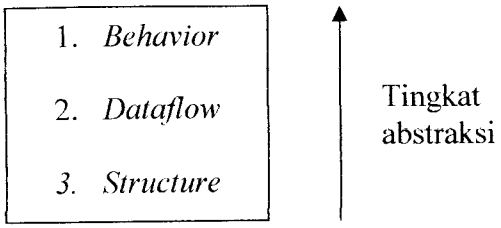
Langkah-langkah mengunakan *software* untuk memprogram FPGA dapat dilihat pada Gambar 2.13



Gambar 2.13 Diagram penggunaan software untuk memprogram PLD

Program VHDL dimasukan, dan perangkat tujuan ditentukan pula *pin* dari perangkat dibuat sebelum program di *compile*. Setelah program di *compile*, *software* menyediakan simulasi desain untuk menyajikan kinerja yang sepatutnya, desain siap untuk di *download* keperangkat yang diinginkan.

VHDL dapat digunakan untuk melakukan desain sebuah sistem sampai dengan implementasinya, dengan kemungkinan untuk mencampur beberapa level abstraksi bahasa.



Gambar 2.14 Level abstraksi VHDL

Beberapa level abstraksi bahasa yang digunakan dalam VHDL dimana level tertinggi memiliki tingkat abstraksi yang terbesar serta memiliki tingkat pemahaman yang paling mudah ketika dikomunikasikan dengan manusia. Sedangkan level terendah memiliki tingkat abstraksi yang paling dipahami oleh perangkat keras.

a. *Behavior*

Dimana dalam level ini sebuah sistem digital didefinisikan sebagai operasi yang dilakukan sistem terhadap waktu. Konsep waktu menjadi pembeda utama antara level ini dengan yang lain. Dalam level ini tunda waktu juga dapat dimasukkan ke dalam perhitungan proses. Level inilah yang paling mirip dengan bahasa pemrograman yang lazim digunakan pada komputer dan paling mudah untuk dipelajari oleh manusia.

b. *Dataflow*

Pada level ini sistem digital atau sebuah rangkaian digital dideskripsikan dengan cara bagaimana data berpindah-pindah dalam sistem yang kita

inginkan. Karena hampir semua piranti digital sekarang memakai register, maka level abstraksi *dataflow* mendeskripsikan bagaimana data berpindah-pindah diantara register yang ada dalam sistem.

c. *Structure*

Dalam level ini sistem digital atau rangkaian digital dideskripsikan dalam komponen-komponen penyusunnya dalam bentuk gerbang-gerbang dasar. Level ini membantu dalam membagi sebuah sistem yang besar menjadi bagian-bagian tertentu yang lebih kecil dan mudah diwujudkan. Semakin kompleks sebuah sistem maka semakin diperlukan struktur disain yang jelas agar kesalahan mudah dideteksi.

Tiga hal yang menjadi struktur dasar VHDL, yaitu :

1. *Library*

Library merupakan bagian penting dari program. Dengan mengetahui *library* yang sudah ada dalam *software* maka tidak perlu lagi membuat program dengan fungsi yang sama yang telah disediakan dalam *software*. Terdapat dua macam *library* bila dilihat dari sumber, yaitu :

a. *library* yang berasal dari program yang telah dibuat seperti berikut :

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

```
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

b. library yang berasal asli dari bawaan *software* seperti berikut :

std_logic_misc	VITAL_Primitives
std_logic_arith	std_logic_unsigned
std_logic_1164	std_logic_textio
WAVES_INTERFACE	std_logic_signed
WAVES_1164_UTILITY	Numeric_std
WAVES_1164_FRAMES	Numeric_bit
WAVES_1164_DECLAR...	MATH_REAL
VITAL_Timing	MATH_COMPLEX

2. Deklarasi Entitas

Entity menggambarkan suatu pemberi fungsi logika, atau bagian yang menginisialisasikan input dan output dan juga tipe data yang dikendalikan. Input dan output untuk fungsi logika biasanya atau sering disebut *port*. Struktur umum dan sintaks deklarasi entitas VHDL adalah seperti berikut:

```
entity <pengenal entitas> is
        port (deskripsi sinyal);
end entity <pengenal entitas>;
```

Baris pertama dalam deklarasi entitas berawal dengan kata *entity*, yang diikuti dengan pengenal yang ditetapkan sebagai nama entitas dan diakhiri kata *is*. Baris

kedua adalah pernyataan *port* yang mendefinisikan sinyal-sinyal input dan output, yang disebut *port*. Setiap *port* input dan output ditetapkan sebuah pengenal. Baris ketiga mendefinisikan akhir deklarasi entitas dengan memakai kata kunci *end entity* dan nama entitas. Perhatikan, harus ada titik koma setelah pernyataan *port* dan pernyataan *end*.

Pernyataan *port* merupakan sinyal input atau output. Pernyataan *port* harus menyebutkan pengenal *port*, arah *port*, dan tipe data *port* untuk setiap sinyal. Ada tiga arah yang dapat ditetapkan untuk sebuah *port* yakni : *in* untuk input, *out* untuk output, dan *in-out* untuk port yang dapat menjadi input atau output yang disebut *port* dua arah (*bidirectional*).

3. Arsitektur

Begitu *port* input dan output sudah didefinisikan dengan entitas, operasi fungsi logika harus dideskripsikan dengan *architecture*. Deklarasi arsitektur merupakan tempat menyebutkan operasi fungsi logika. Untuk setiap entitas harus ada arsitektur.

Setiap deklarasi arsitektur harus dikaitkan dengan nama serta sebuah entitas. Struktur dan sintaks dasar deklarasi arsitektur adalah sebagai berikut:

architecture <nama arsitektur> **of** <nama entitas> **is**

begin

Deskripsi fungsi logika ditulis di sini.

end architecture <nama arsitektur>;

Baris pertama dalam deklarasi arsitektur dimulai dengan kata kunci *architecture,* lalu diikuti dengan pengenal yang ditetapkan sebagai nama arsitektur itu. Kata kunci *of* dan nama entitas terkait mengikuti nama arsitektur; baris itu berakhir dengan kata kunci *is*. Perintah ini mengikatkan arsitektur ke entitas tertentu.

Baris kedua hanya merupakan kata kunci *begin*. Setelah begin adalah pernyataan-pernyataan VHDL yang diperlukan untuk menjelaskan fungsi logika. Jumlah baris kode dalam deskripsi fungsional ini bebas, tergantung pada tipe fungsi logika yang ingin dideskripsikan dan kompleksitasnya.

Baris terakhir mendefinisikan akhir deklarasi arsitektur dengan memakai kata kunci *end architecture* dan nama arsitektur tersebut. Titik koma harus dibubuhkan di akhir pernyataan ini.

Aturan dasar untuk menulis sebuah kode VHDL adalah sebagai berikut :

1. Dalam sebuah desain VHDL minimal terdapat satu pasang *entity / architecture.*
2. *Entity* mendeskripsikan unjuk kerja sistem jika dilihat dari perspektif input / output.
3. *Entity* tidak memberikan deskripsi kerja (*behavioral / structure / dataflow)* dari sistem yang dibuat. Deskripsi kerja termuat dalam *architecture.*
4. Sintak :

   Entity <nama_rangkaian> **is**

Port ([signal]<nama> : [<mode>]<indikasi_tipe>;

.....

[signal]<nama> : [<mode>]<indikasi_tipe>;

end <nama_rangkaian>;

Architecture <nama arsitektur> **of** <nama_rangkaian> **is**

begin

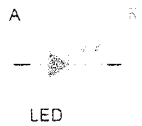
Deskripsi fungsi logika ditulis di sini.

end architecture <nama arsitektur>;

2.4. *Light Emitting Diode* (LED)

Dioda pemancar cahaya LED adalah dioda semikonduktor khusus yang dapat memancarkan cahaya apabila arus melaluinya. Apabila diberi tegangan maju, energi elektron yang mengalir melewati tahanan sambungan diubah langsung menjadi energi cahaya. Karena LED adalah dioda, maka arus hanya akan mengalir apabila LED dihubungkan dengan tegangan maju.

LED harus dioperasikan didalam ukuran kerja tegangan dan arus yang tertentu untuk mencegah kerusakan. Sebagian besar LED membutuhkan 1,5 sampai 2,2V untuk memberi tegangan maju dan dapat dilalui dengan aman arus sebesar 20 sampai 30mA. LED biasanya dihubungkan seri dengan tahanan yang membatasi tegangan dan arus pada nilai yang dikehendaki.



Gambar 2.15 Simbol LED

LED merupakan produk temuan lain setelah dioda. Strukturnya juga sama dengan dioda, tetapi belakangan ditemukan bahwa elektron yang menerjang sambungan P-N juga melepaskan energi berupa energi panas dan energi cahaya. LED dibuat agar lebih efisien jika mengeluarkan cahaya. Untuk mendapatkan emisi cahaya pada semi konduktor, doping yang dipakai adalah galium, arsenic dan phosporus.

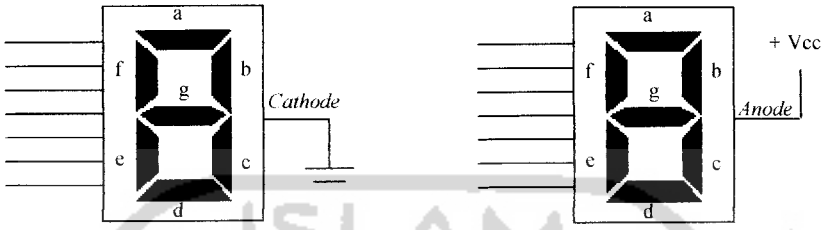
Pada saat ini warna-warna cahaya LED yang banyak adalah warna merah, kuning, hijau dan biru. Pada dasarnya semua warna bisa dihasilkan, namun akan menjadi sangat mahal dan tidak efisien. Dalam memilih LED selain warna, perlu diperhatikan tegangan kerja, arus maksimum dan disipasi daya-nya. Rumah (*casing*) LED dan bentuknya bermacam-macam, ada yang persegi empat, bulat dan lonjong

2.5. Seven-segment

Hasil pemrosesan sinyal dari suatu rangkaian digital merupakan sinyal digital dalam bentuk kode-kode biner. Jika hasil tersebut tetap disajikan dalam bentuk aslinya yakni kode biner, maka akan mengalami kesulitan di dalam membacanya karena tidak terbiasa menggunakan kode biner dalam kehidupan sehari-hari. Kebiasaan menggunakan sajian bilangan dalam bentuk desimal. Agar menjadi mudah dibaca, maka kode-kode biner tersebut perlu diubah tampilannya menggunakan tampilan desimal.

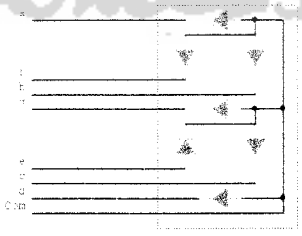
Sebagai tampilan digunakan LED (*Light Emitting Diode*) 7 ruas, sehingga dapat menampilkan angka mulai dari 0 hingga 9. Dengan menyusun dua buah 7-

segment maka dapat ditampilkan angka dua digit, sebagai pewaktu pada saat lampu hijau, lampu kuning, dan lampu merah menyala. Bentuk peraga *seven-segment* ditunjukkan pada Gambar 2.16 berikut ini.



Gambar 2.16 Peraga *seven segment* : (a) jenis *common cathode*, (b) jenis *common anode*

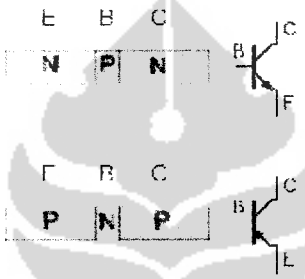
Seven-segment memiliki 2 jenis yaitu *common cathode* dan *common anode*. Pada jenis *common cathode* diperlukan sinyal tinggi untuk menyalakan setiap *segment*nya. Sedangkan jenis *common anode* memerlukan sinyal rendah untuk setiap *segment*nya. Pada skripsi ini digunakan *seven-segment common anode*. Rangkaian *seven-segment common anode* bersama dapat dilihat pada Gambar 2.17



Gambar 2.17 Rangkaian *seven-segment common anode*

2.6. Transistor

Transistor merupakan dioda dengan dua sambungan (*junction*). Sambungan itu membentuk transistor NPN maupun PNP. Ujung-ujung terminalnya berturut-turut disebut emitor, base dan kolektor. Base selalu berada di tengah, di antara emitor dan kolektor. Transistor ini disebut transistor *bipolar*, karena struktur dan prinsip kerjanya tergantung dari perpindahan elektron di kutub negatif mengisi kekurangan elektron (hole) di kutub positif. Adalah William Schockley pada tahun 1951 yang pertama kali menemukan transistor *bipolar*.

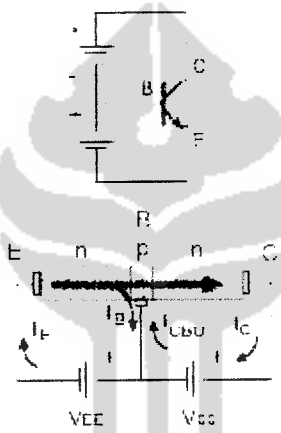


Gambar 2.18 Transistor NPN dan PNP

Transistor adalah komponen yang bekerja sebagai sakelar (switch *on/off*) dan juga sebagai penguat (*amplifier*). Transistor *bipolar* adalah inovasi yang mengantikan transistor tabung (*vacum tube*). Selain dimensi transistor *bipolar* yang relatif lebih kecil, disipasi dayanya juga lebih kecil sehingga dapat bekerja pada suhu yang lebih dingin. Dalam beberapa aplikasi, transistor tabung masih digunakan terutama pada aplikasi audio, untuk mendapatkan kualitas suara yang baik, namun konsumsi dayanya sangat besar. Sebab untuk dapat melepaskan elektron, teknik yang digunakan adalah pemanasan filamen seperti pada lampu pijar.

Bias Transistor

Transistor *bipolar* memiliki 2 *junction* yang dapat disamakan dengan penggabungan 2 buah dioda. Emiter-Base adalah satu junction dan Base-Kolektor *junction* lainnya. Seperti pada dioda, arus hanya akan mengalir hanya jika diberi bias positif, yaitu hanya jika tegangan pada material P lebih positif daripada material N (*forward bias*). Pada gambar ilustrasi transistor NPN berikut ini, *junction* base-emiter diberi bias positif sedangkan base-colector mendapat bias negatif (*reverse bias*).



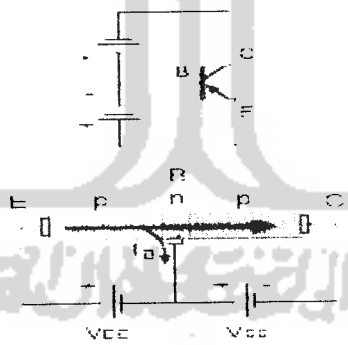
Gambar 2.19 Arus elektron transistor NPN

Karena base-emiter mendapat bias positif maka seperti pada dioda, elektron mengalir dari emiter menuju base. Kolektor pada rangkaian ini lebih positif sebab mendapat tegangan positif. Karena kolektor ini lebih positif, aliran elektron bergerak menuju kutub ini. Misalnya tidak ada kolektor, aliran elektron seluruhnya akan menuju base seperti pada dioda. Tetapi karena lebar base yang sangat tipis, hanya sebagian elektron yang dapat bergabung dengan hole yang ada pada base. Sebagian besar akan menembus lapisan base menuju kolektor. Inilah

alasannya mengapa jika dua dioda digabungkan tidak dapat menjadi sebuah transistor, karena persyaratannya adalah lebar base harus sangat tipis sehingga dapat diterjang oleh elektron.

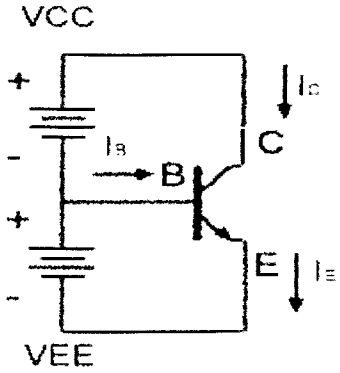
Jika misalnya tegangan base-emitor dibalik (*reverse bias*), maka tidak akan terjadi aliran elektron dari emitor menuju kolektor. Jika pelan-pelan 'keran' base diberi bias maju (*forward bias*), elektron mengalir menuju kolektor dan besarnya sebanding dengan besar arus bias base yang diberikan. Dengan kata lain, arus base mengatur banyaknya elektron yang mengalir dari emiter menuju kolektor. Ini yang dinamakan efek penguatan transistor, karena arus base yang kecil menghasilkan arus emiter-kolektor yang lebih besar.

Pada transistor PNP, fenomena yang sama dapat dijelaskan dengan memberikan bias seperti pada gambar berikut. Dalam hal ini yang disebut perpindahan arus adalah arus hole.



Gambar 2.20 Arus hole transistor PNP

Untuk memudahkan pembahasan prinsip bias transistor lebih lanjut, berikut adalah terminologi parameter transistor. Dalam hal ini arah arus adalah dari potensial yang lebih besar ke potensial yang lebih kecil.



Gambar 2.21 Arus potensial

I_C : arus kolektor

I_B : arus basis

I_E : arus emitor

I_{CBO} : arus basis-kolektor

VCC : tegangan pada kolektor

VCE : tegangan jepit kolektor-emitor

VEE : tegangan pada emitor

Gambar 2.22 Penampang transistor bipolar

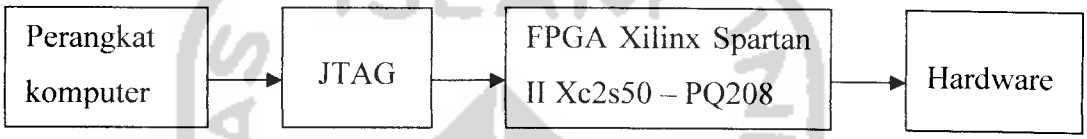
Dari satu bahan silikon (*monolitic*), emitor dibuat terlebih dahulu, kemudian base dengan doping yang berbeda dan terakhir adalah kolektor. Terkadang dibuat juga efek dioda pada terminal-terminalnya sehingga arus hanya akan terjadi pada arah yang dikehendaki.

BAB III

PERANCANGAN SISTEM

3.1. Pendahuluan

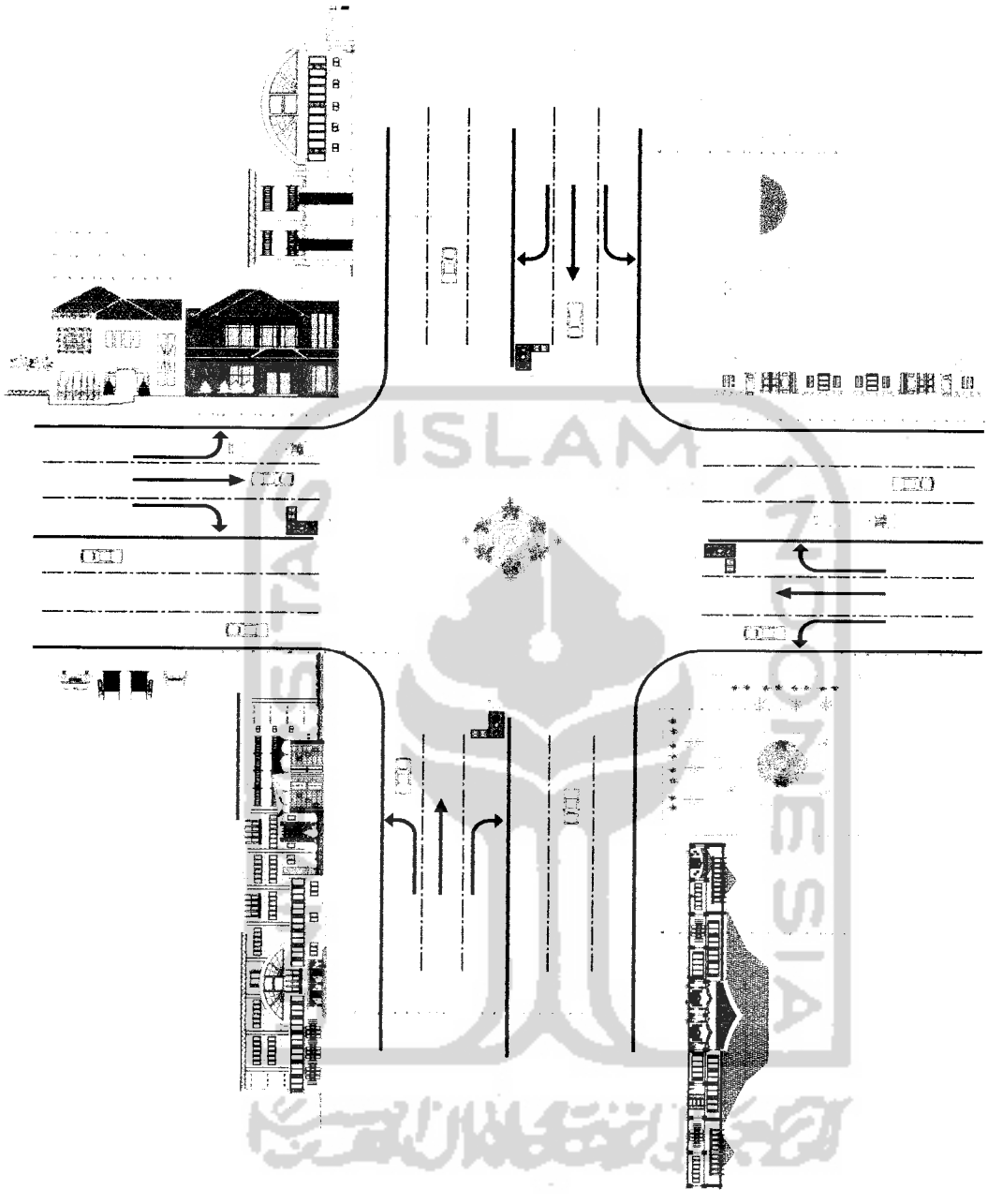
Perancangan sistem lampu lalu lintas terpadu otomatis berbasis FPGA xilinx spartan II Xc2s50 – PQ208 dapat dilihat pada blok diagram Gambar 3.1.



Gambar 3.1 Perancangan sistem lampu lalu lintas terpadu otomatis

Dalam proses perancangan dibutuhkan perangkat komputer sebagai media dalam menggunakan *software* xilinx dengan VHDL sebagai bahasa dalam pemrograman. JTAG merupakan *interface* yang digunakan saat *download* program ke FPGA xilinx Spartan II Xc2s50-PQ208. *Output* dari FPGA xilinx Spartan II Xc2s50-PQ208 dapat berupa piranti internal berupa LED dan *seven segment* yang sudah terdapat pada *board* Pegasus atau piranti eksternal seperti rangkaian lampu lalu lintas terpadu.

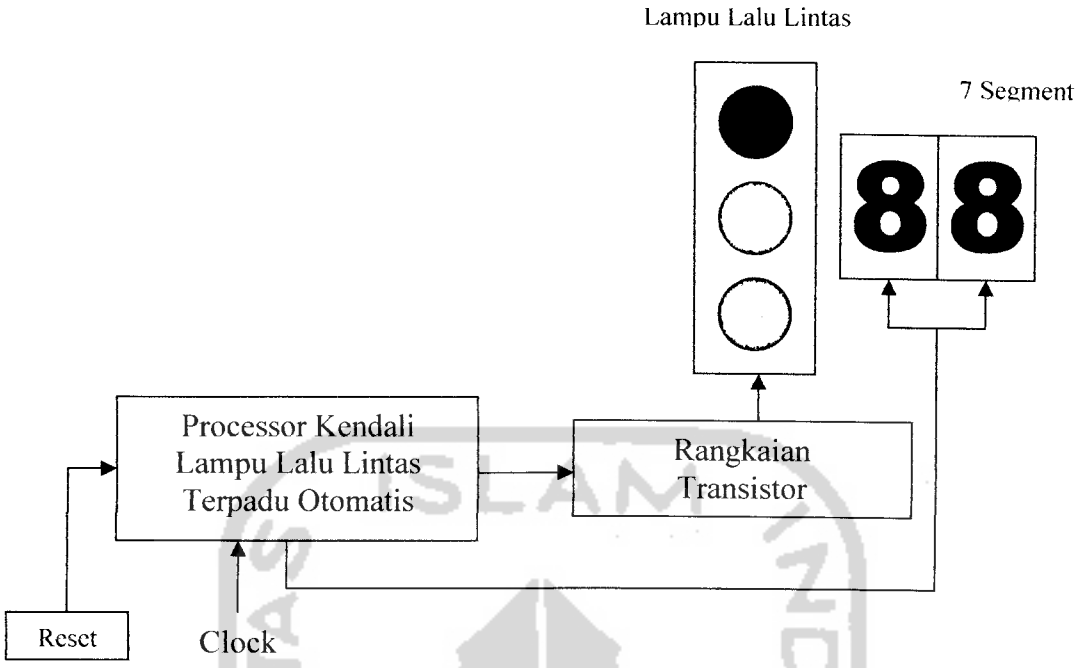
Lampu lalu lintas otomatis terpadu dirancang untuk penerapan lampu diikuti *display* pewaktu pada jalur yang berjumlah 4 seperti diilustrasikan pada Gambar 3.2.



Gambar 3.2 Perancangan lampu lalu lintas terpadu otomatis

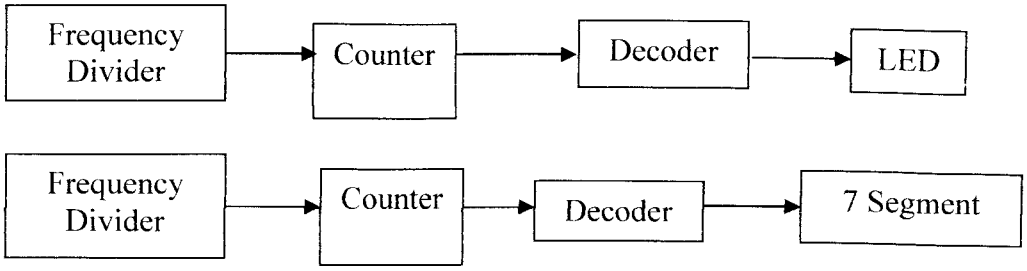
3.2. Perancangan Software Lampu Lalu Lintas Terpadu

Perancangan khusus lampu lalu lintas terpadu otomatis dapat dilihat pada blok diagram dari sistem lampu lalu lintas ditunjukkan oleh Gambar 3.3



Gambar 3.3 Blok diagram dari sistem

Blok diagram sistem perancangan *software* lampu lalu lintas otomatis terpadu terdiri dari empat blok yakni *frequency divider*, *counter*, *decoder* dan *output*. Perancangan ini menggunakan dua blok *frequency divider*, dua blok *counter*, dua blok *decoder* dan dua *output* pada tiap jalurnya. Blok *frequency divider* berfungsi sebagai pengatur keluaran *clock* utama dengan status keluaran yang berubah secara konstan sehingga menghasilkan gelombang kotak atau pulsa. Blok *counter* berfungsi untuk mencacah keadaan lampu dan *seven segment*. Blok *decoder* untuk mengubah kode-kode biner yang ada pada *input*-nya menjadi data asli pada *output*-nya. Keluarannya terdiri dari keadaan lampu dan penampil waktu.



Gambar 3.4 Blok diagram lampu lalu lintas otomatis terpadu

Prinsip kerja lampu lalu lintas dapat dilihat pada Tabel 3.1. Output masing-masing LED bersifat aktif *low* dimana LED akan ON saat diberi logika 0 dan akan OFF saat diberi logika 1.

Tabel 3.1. Logika sistem lampu lalu lintas.

Light Output											
Line 1			*Line 2*			*Line 3*			*Line 4*		
M	K	H	M	K	H	M	K	H	M	K	H
1	1	0	0	1	1	0	1	1	0	1	1
1	0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	1	0	1	1	0	1	1
0	1	1	1	1	0	0	1	1	0	1	1
0	1	1	1	0	1	0	1	1	0	1	1
0	1	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	1	1	1	0	0	1	1
0	1	1	0	1	1	1	0	1	0	1	1
0	1	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	1	0	1	1	1	1	0
0	1	1	0	1	1	0	1	1	1	0	1
0	1	1	0	1	1	0	1	1	0	1	1

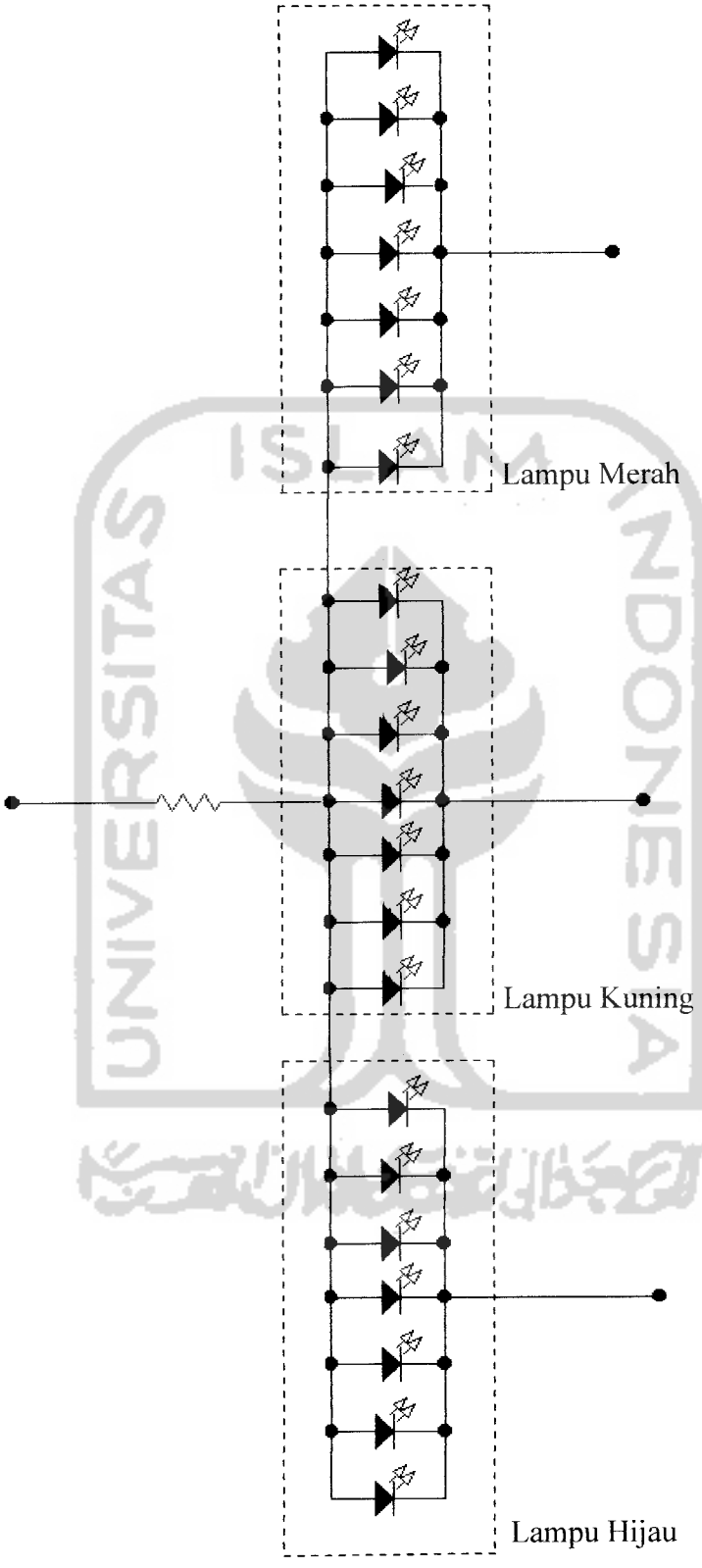
3.3. Perancangan *Hardware* Lampu Lalu Lintas Otomatis

Perancangan *hardware* lampu lalu lintas ini terdiri dari:

1. Rangkaian LED
2. Rangkaian Darlington
3. Rangkaian *Seven segment*

3.3.1 Rangkaian LED

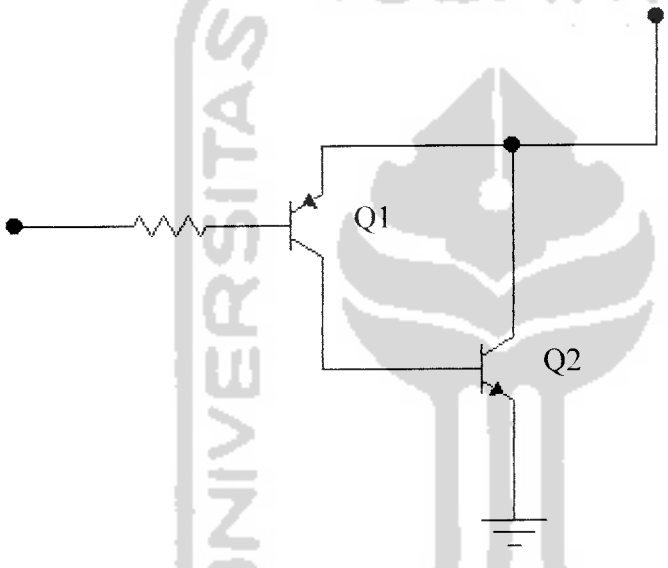
Pada rangkaian ini masing–masing jalur mempunyai 3 warna lampu yang berbeda, yaitu : lampu merah, lampu kuning, dan lampu hijau. Tiap warna lampu terdiri dari 7 LED yang dipasang secara *paralel* yang kemudian keluaran dari LED tersebut dihubungkan ke rangkaian transistor. Pada rangkaian LED ini diberi hambatan 100 ohm pada masing-masing warna lampu, yang berfungsi untuk membatasi arus yang keluar dari *power suplai.* Gambar rangkaian LED dapat dilihat pada Gambar 3.5.



Gambar 3.5 Rangkaian LED untuk 1 jalur

3.3.2 Rangkaian Darlington

Rangkaian transistor yang digunakan adalah rangkaian Darlington, dengan penggabungan transistor NPN dan PNP. Resistor yang digunakan pada rangkaian ini adalah 1 Kohm. Gambar rangkaian transistor dapat dilihat pada Gambar 3.6. Transistor 1 (Q_1) merupakan transistor jenis PNP BC559 dan transistor 2 (Q_2) merupakan transistor jenis NPN 2N2222.



Gambar 3.6 Rangkaian Darlington

Rangkaian Darlington adalah rangkaian antara dua transistor yang gain arus totalnya adalah perkalian gain arus sendiri-sendiri (lihat persamaan 3.1). Karena gain arus jauh lebih besar, rangkaian Darlington dapat mempunyai impedansi masukan yang sangat tinggi dan dapat menghasilkan arus keluaran yang sangat besar.

Jika penguatan $Q_1 = \beta_1$ dan $Q_2 = \beta_2$

Maka penguatan rangkaian adalah :

$I_{C2} \approx I_{E2}$

$I_{C2} = \beta_2 I_{B2}$

$I_{E2} = \beta_2 I_{B2}$

$I_{E1} = \beta_1 I_{B1}$

$= \beta_1 I_{E2}$

$= \beta_1 (\beta_1 I_{B2})$

$= (\beta_1 \beta_2) I_{B2}$

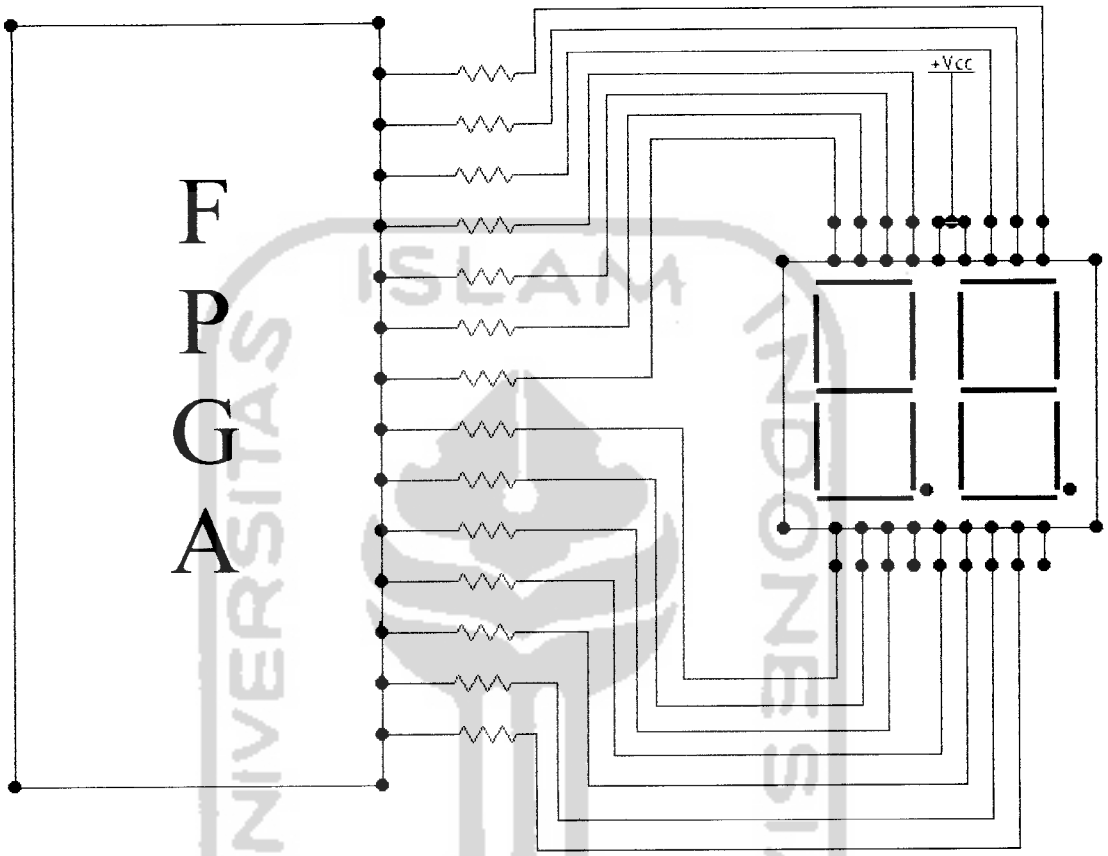
$\beta = \beta_1 \beta_2$ .....(3.1)

3.3.3 Rangkaian *Seven Segment*

Rangkaian ini digunakan untuk menampilkan waktu pada tiap keadaan lampu. *Seven segment* yang digunakan 2 digit, untuk menampilkan waktu sampai batas puluhan. Catu daya yang digunakan *seven segment* ini adalah catu dari modul FPGA sebesar 3.3 volt. Resistor yang digunakan sebesar 100 Ohm pada tiap *segment*-nya. Gambar rangkaian seven segment dapat dilihat pada Gambar 3.7.

Jenis *seven segment* yang digunakan adalah *common* anoda maka anoda *seven segment* disatukan dihubungkan ke V_{CC} FPGA kemudian masing-masing katodanya dihubungkan ke resistor 100Ω keluaran dari resistor dihubungkan ke port *input/output* FPGA. Prinsip kerjanya adalah karena jenis *seven segment* yang digunakan adalah *common* anoda (aktif *low*) maka saat diberi masukan 0 volt oleh

FPGA, *seven segment* akan menyala, sebaliknya jika diberi masukan 3.3 volt *seven segment* akan mati.



Gambar 3.7 Rangkaian *Seven segment*

BAB IV

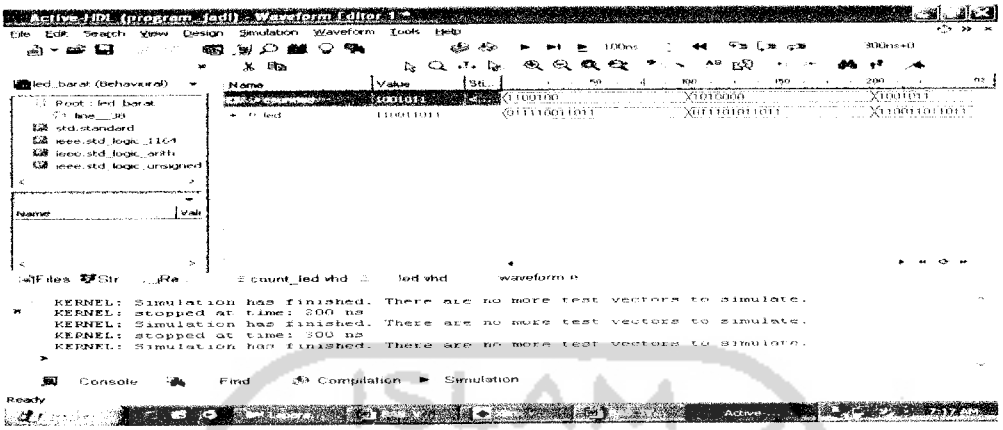
ANALISA DAN PEMBAHASAN

Pada bab ini akan dibahas mengenai keseluruhan pengujian sistem. Materi pengujian sistem meliputi dua bagian yakni dengan simulasi, yang bertujuan untuk mengetahui apakah program yang dibuat tidak terdapat *error* pada saat peng-*compile*-lan dan pengujian mekanik, yang bertujuan untuk menyelaraskan antara program dan alat yang telah dibuat. Simulasi dilakukan untuk meliahat proses yang terjadi saat dilakukan pemberian *input* melalui tombol *push button pada* modul Pegasus dengan menggunakan *software active-HDL.*

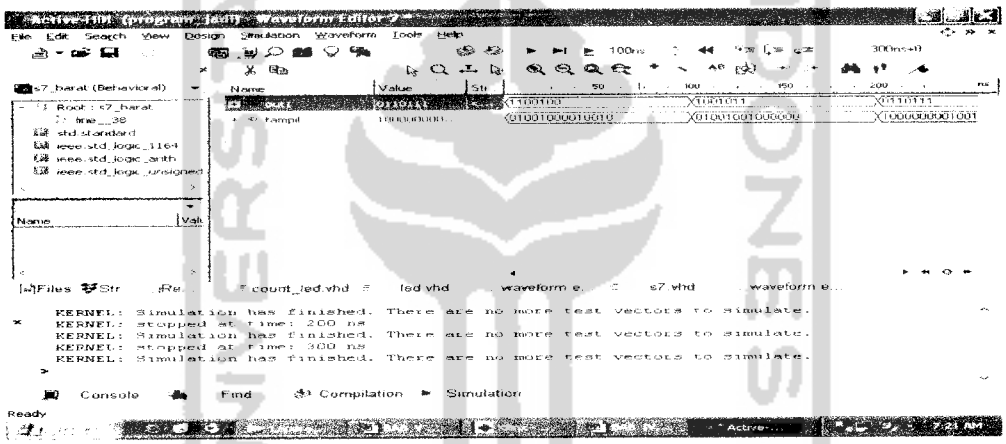
4.1 Hasil Simulasi Sistem

Setelah program untuk mengkonfigurasikan FPGA selesai dibuat, kemudian untuk memastikan kinerja dari program apakah sudah sesuai dengan yang diinginkan, maka ada baiknya program tersebut disimulasikan terlebih dahulu. Simulasi dilakukan perbagian dari setiap bagian pada sistem. Hasil dari simulasi adalah berupa *timing* diagram dan untuk melakukannya digunakan *software active-HDL.*

Ada 8 buah *output* dari keseluruhan sistem, dengan setiap jalurnya terdiri dari 2 buah *output*, yaitu : *timing* diagram *output* LED (dapat dilihat pada Gambar 4.1) dan *timing* diagram *output* penampil 2 *digit* (dapat dilihat pada Gambar 4.2).



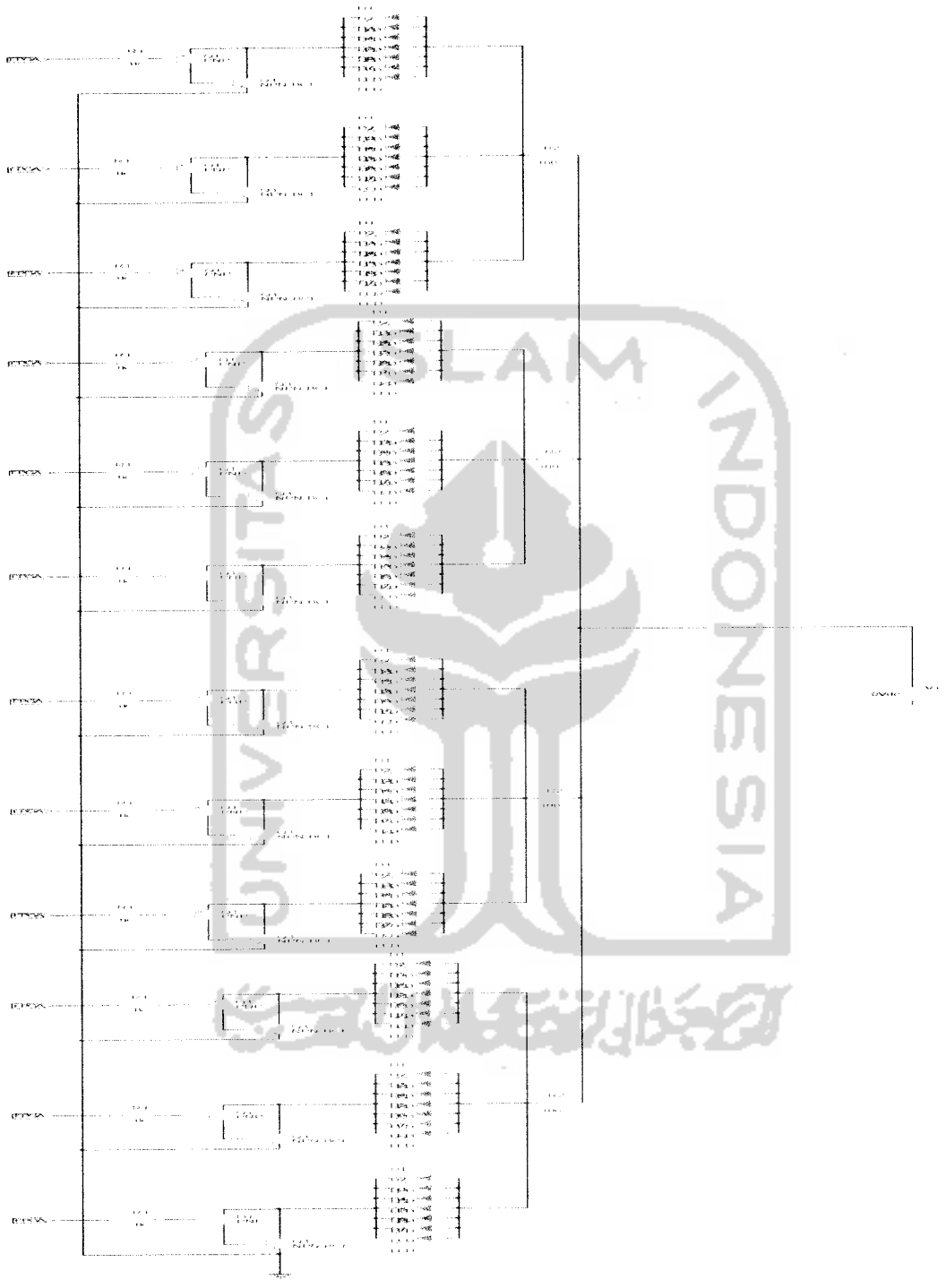
Gambar 4.1 *Timing* diagram *output* LED



Gambar 4.2 *Timing* diagram *output* penampil

4.1.1 LED

Timing diagram *output* LED mempunyai 12 bit keluaran, yang setiap bitnya menunjukkan keadaan tiap warna lampu untuk setiap jalur. Untuk setiap lampu hijau lama waktunya adalah 20 detik, lampu kuning 5 detik, dan lampu merah adalah penjumlahan dari kedua keadaan lampu tersebut. Untuk mencacah keadaan lampu dari empat jalur digunakan *counter*.



Gambar 4.3 Blok diagram rangkaian LED

Analisis Rangkaian LED

$V_{adaptor}$ = 6,12 volt

$$\frac{1}{Rp} = \frac{1}{R1} + \frac{1}{R2} + \frac{1}{R3} + \frac{1}{R4}$$

$$= \frac{1}{100} + \frac{1}{100} + \frac{1}{100} + \frac{1}{100}$$

$$= \frac{4}{100}$$

$$Rp = \frac{100}{4} = 25 \text{ ohm}$$

$$I = \frac{Vadaptor}{Rp} = \frac{6.12volt}{25ohm} = 0{,}2448 \text{ A} = 244{,}8 \text{ mA}$$

$$I_{tiap\ jalur} = \frac{Itiap\_jalur}{4} = \frac{244.8mA}{4} = 61{,}2 \text{ mA}$$

$$I_{tiap\ warna} = \frac{Itiap\_jalur}{3} = \frac{61.2mA}{3} = 20{,}4 \text{ mA}$$

Cara kerja Rangkaian LED

LED yang digunakan adalah jenis LED *common* anoda (aktif *low*). Catu daya yang digunakan adalah catu daya adaptor 6,12 Volt. Kutub positif dari catu daya diberi hambatan 100Ω pada tiap-tiap jalur. Kemudian keluaran dari tiap-tiap hambatan dihubungkan ke rangkaian LED, keluaran dari rangkaian LED dihubungkan ke rangkaian Darlington, keluaran dari rangkaian Darlington diberi

hambatan 100Ω kemudian dihubungkan ke *port input/output* FPGA. Prinsip kerjanya adalah LED akan menyala jika FPGA memberikan logika *low*, sebaliknya LED akan mati jika FPGA logika *high*.

4.1.2 Penampil

Timing diagram *output* penampil mempunyai 14 bit keluaran. Tiap keadaan bit menunjukkan keadaan penampil yang berbeda, yaitu mulai dari penampil angka 1-75. Untuk mencacah keadaan penampil digunakan *counter*.

4.2 Analisis Sistem

Sistem pewaktu dapat bekerja dengan sempurna jika telah di *synthesize* sebelum *listing* program akan di *compare* menjalankan *counter* untuk LED dan *counter* untuk penampil. Untuk mengoperasikannya alat ini dapat dilakukan dengan langkah-lngkah berikut ini :

1. Menekan tombol reset pada modul Pegasus yang telah dialamatkan, dengan alamat pin 59 pada program yang telah dibuat.
2. *Counter* akan mulai mencacah LED dan penampil waktu pada saat yang bersamaan.
3. Keadan awal saat saat tombol reset ditekan adalah :

- Jalur Selatan : LED hijau *ON* dan penampil pewaktu menunjukkan angka 20.
- Jalur Barat : LED merah *ON* dan penampil waktu menunjukkan angka 25.
- Jalur Utara : LED merah *ON* dan penampil waktu menunjukkan angka 50.
- Jalur Timur : LED merah *ON* dan penampil waktu menunjukkan angka 75.

4. Saat tombol reset dilepas *counter* untuk LED *dan counter* untuk penampil waktu akan mulai mencacah.
   - Pada jalur Selatan, LED hijau *ON* dan penampil pewaktu menunjukkan angka 20. *Counter* LED dan *counter* penampil waktu akan mulai mencacah mundur dari angka 20 sampai angka 01, kemudian LED kuning akan *ON* dan *counter* penampil waktu akan mencacah dari angka 05 sampai angka 01, setelah itu LED merah *ON* dan *counter* penampil waktu akan mencacah dari angka 75 sampai angka 01.
   - Pada jalur barat, LED merah *ON* dan penampil pewaktu menunjukkan angka 25. *Counter* LED dan *counter* penampil waktu akan mulai mencacah dari angka 25 sampai angka 01, kemudian LED hijau akan *ON* dan *counter* penampil waktu akan mencacah dari angka 20 sampai

angka 01, setelah itu LED kuning *ON* dan *counter* penampil waktu akan mencacah dari angka 05 sampai angka 01.

- Pada jalur utara, LED merah *ON* dan penampil pewaktu menunjukkan angka 50. *Counter* LED dan *counter* penampil waktu akan mulai mencacah mundur dari angka 50 sampai angka 01, kemudian LED hijau akan *ON* dan *counter* penampil waktu akan mencacah dari angka 20 sampai angka 01, setelah itu LED kuning *ON* dan *counter* penampil waktu akan mencacah dari angka 05 sampai angka 01.
- Pada jalur timur, LED merah *ON* dan penampil pewaktu menunjukkan angka 75. *Counter* LED dan *counter* penampil waktu akan mulai mencacah mundur dari angka 75 sampai angka 01, kemudian LED hijau akan *ON* dan *counter* penampil waktu akan mencacah dari angka 20 sampai angka 01, setelah itu LED kuning *ON* dan *counter* penampil waktu akan mencacah dari angka 05 sampai angka 01.

5. Tampilan 7-ruas 2 buah untuk menampilkan hasil lamanya keadaan waktu.

4.3 Pembahasan

Pada perancangan lampu lalu lintas terpadu otomatis kali ini ditampilkan pewaktu dua *digits* pada saat LED merah, LED kuning, dan LED hijau menyala pada masing-masing jalur. Sehingga para pengguna jalan mengetahui berapa lama lagi

mereka akan menunggu lampu hijau menyala dan berapa lama lagi lampu hijau akan mati dan berganti ke lampu kuning, dan berapa lama lagi lampu kuning akan mati yang kemudian akan berganti ke lampu merah. Hal ini dapat mengurangi tingkat kemacetan dan kecelakaan pada ruas jalan.

Pada perancangan kali ini, penulis membuat program *counter* untuk LED dan *counter* untuk *seven segment* untuk setiap jalurnya. Yang kemudian keadaan setiap cacahan *counter* akan ditampilkan lewat LED dan *seven segment*. Dengan *input counter* yang digunakan adalah 7 bit untuk mencacah lamanya keadaan lampu pada masing-masing jalur dan menampilkannya lewat *seven segment*. *Counter* yang digunakan adalah counter *step down*. Agar dapat mencacah mundur untuk 100 keadaan dalam 1 siklus perputaran.

BAB V

PENUTUP

5.1. Kesimpulan

Dari hasil pengujian, pengamatan dan analisa pada perancangan sistem, dapat diperoleh beberapa kesimpulan sebagai berikut :

1. Alat ini bekerja secara otomatis dengan menampilkan waktu pada setiap keadaan lampu.
2. Penggunaan *display* lebih dari satu digit dapat dilakukan dengan cara membagi frekuensi pada *clock*-nya.
3. Lama waktu lampu hijau menyala 20 detik, kuning 5 detik, dan merah 25 detik.
4. Gerbang logika yang mampu dilayani oleh FPGA Xilink spartan2 Xc2s50-PQ208 hanya 50 ribu gerbang logika dasar digital sehingga untuk perancangan ini sudah cukup hanya dengan menggunakan FPGA seri ini.
5. Tanda *WARNING* yang muncul saat program disintesis hanyalah merupakan informasi yang memberi tahu bahwa *syntax* ada yang tidak digunakan dan bukan suatu kesalahan.

5.2. Saran

1. Waktu perpindahan setiap LED masih sama. Untuk penelitian berikutnya dicoba untuk mengembangkan sistem lampu lalu lintas terpadu otomatis ini dengan pewaktuan masing-masing LED yang bervariasi.
2. Jika pengendalian dilakukan dengan FPGA hendaknya diperhitungkan banyaknya gerbang logika dasar digital yang mampu dilayani. Untuk perancangan lampu lalu lintas terpadu otomatis ini cukup dengan menggunakan Xilink Spartan2 Xc2s50-PQ208, karena seri ini sudah memiliki jumlah gerbang yang lebih dari cukup.

DAFTAR PUSTAKA

Floyd, Thomas L., *Digital Fundamentals with VHDL.* New Jersey: Prentice Hall, 2003.

Ibrahim, K. F., *Teknik Digital.* Jogjakarta: Andi Offset, 1996.

Malvino, Albert, Paul, *Elektronika Komputer Digital.* Edisi Kedua, Jakarta: Erlangga, 1996.

Parnel, Karen., Mehta, Nick., *Programmable Logic Design Quick Start Hand Book.* Xilinx, 2002.

Training Center Electrical Engineering. *Modul Pelatihan IC Design With Based FPGA Pegasus..* Jogjakarta: Teknik Elektro Universitas Islam Indonesia, 2006.

Widodo Budiharto, S.Si, M.Kom., &Sigit Firmansyah. *Elektronika Digital; dan Mikroprosesor.* Jogjakarta: Andi Offset, 2005.

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:    07:27:06 01/28/07
-- Design Name:
-- Module Name:    fd_led_barat - Behavioral
-- Project Name:
-- Target Device:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity fd_led_barat is
port (clk,rst: in STD_LOGIC;
                clk_out: out STD_LOGIC);
end fd_led_barat;

architecture Behavioral of fd_led_barat is
signal clk_int: std_logic;
constant data:std_logic_vector(24 downto 0):="10111110101111000001000000";
signal sig1: std_logic;
signal sig3: std_logic_vector(24 downto 0);

begin
  u1:process(clk,sig3,rst)
begin
```

```
        if rst='1' then
                sig3<=data;
        elsif clk='1' and clk'event then
                        if sig3="00000000000000000000000000" then
                        sig3<=data;
                else
                        sig3<=(sig3 - 1);
                end if;
         end if;
end process;

u2:process(sig3,clk,rst)
begin
        if rst='1' then
                sig1<='1';
        elsif clk='1' and clk'event then
                if sig3="00000000000000000000000000" then
                        sig1<='1';
                else
                        sig1<='0';
                end if;
        end if;
end process;

u3:process(sig1,clk,clk_int,rst)
begin
        if rst='1' then
                clk_int<='0';
        elsif rising_edge(sig1) then
                clk_int<=not(clk_int);
        else
                clk_int<=clk_int;
        end if;
                clk_out<=clk_int;

end process;

end Behavioral;
```

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:    07:30:53 01/28/07
-- Design Name:
-- Module Name:    fd_s7_barat - Behavioral
-- Project Name:
-- Target Device:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity fd_s7_barat is
port (clk,rst: in STD_LOGIC;
                clk_out: out STD_LOGIC);
end fd_s7_barat;

architecture Behavioral of fd_s7_barat is
signal clk_int: std_logic;
constant data:std_logic_vector(24 downto 0):="1011111010111100001000000";
signal sig1: std_logic;
signal sig3: std_logic_vector(24 downto 0);

begin
 u1:process(clk,sig3,rst)
begin
```

```
        if rst='1' then
                sig3<=data;
        elsif clk='1' and clk'event then
                        if sig3="00000000000000000000000000" then
                        sig3<=data;
                else
                        sig3<=(sig3 - 1);
                end if;
         end if;
end process;

u2:process(sig3,clk,rst)
begin
        if rst='1' then
                sig1<='1';
        elsif clk='1' and clk'event then
                if sig3="00000000000000000000000000" then
                        sig1<='1';
                else
                        sig1<='0';
                end if;
        end if;
end process;

u3:process(sig1,clk,clk_int,rst)
begin
        if rst='1' then
                clk_int<='0';
        elsif rising_edge(sig1) then
                clk_int<=not(clk_int);
        else
                clk_int<=clk_int;
        end if;
                clk_out<=clk_int;

end process;

end Behavioral;
```

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:    07:25:57 01/28/07
-- Design Name:
-- Module Name:    count_led_barat - Behavioral
-- Project Name:
-- Target Device:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity count_led_barat is
   Port ( clk : in std_logic;
          rst : in std_logic;
          cout:out std_logic_vector(6 downto 0));

end count_led_barat;

architecture behavioral of count_led_barat is
signal count:std_logic_vector(6 downto 0);
signal countled:std_logic_vector(6 downto 0);
constant dataled:std_logic_vector(6 downto 0):="1100100";

begin
          process(clk,rst)
```

```
        begin
                if rst='1' then
                        countled<=dataled;
                elsif clk='1' and clk'event then
                        if countled="0000001" then
                          countled<=dataled;
                        else
                                countled<=(countled - 1);
                        end if;
                end if;
        end process;
        cout<=countled;
end behavioral;
```

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:    07:30:04 01/28/07
-- Design Name:
-- Module Name:    count_s7_barat - Behavioral
-- Project Name:
-- Target Device:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity count_s7_barat is
   Port ( clk : in std_logic;
          rst : in std_logic;
          cout:out std_logic_vector(6 downto 0));

end count_s7_barat;

architecture rtl of count_s7_barat is
signal count:std_logic_vector(6 downto 0);
signal counts7HKM:std_logic_vector(6 downto 0);
constant datas7HKM:std_logic_vector(6 downto 0):="1100100";

begin
          process(clk,rst)
```

```
        begin
                if rst='1' then
                        counts7HKM<=datas7HKM;
                elsif clk='1' and clk'event then
                        if counts7HKM="0000001" then
                          counts7HKM<=datas7HKM;
                        else
                                counts7HKM<=(counts7HKM - 1);
                        end if;
                end if;
        end process;
        cout<=counts7HKM;
end rtl;
```

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:    07:28:02 01/28/07
-- Design Name:
-- Module Name:    led_barat - Behavioral
-- Project Name:
-- Target Device:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity led_barat is
   Port ( countled:in std_logic_vector(6 downto 0);
          led     :out std_logic_vector(11 downto 0));
end led_barat;

architecture Behavioral of led_barat is

begin
        process(countled)
        begin
                case countled is

                        when "0000001"=>led<="011011011101";--merah_barat--
                        when "0000010"=>led<="011011011101";--merah_barat--
```

```
when "0000011"=>led<="011011011101";--merah_barat--
when "0000100"=>led<="011011011101";--merah_barat--
when "0000101"=>led<="011011011101";--merah_barat--

when "0000110"=>led<="011011011110";--merah_barat--
when "0000111"=>led<="011011011110";--merah_barat--
when "0001000"=>led<="011011011110";--merah_barat--
when "0001001"=>led<="011011011110";--merah_barat--
when "0001010"=>led<="011011011110";--merah_barat--
when "0001011"=>led<="011011011110";--merah_barat--
when "0001100"=>led<="011011011110";--merah_barat--
when "0001101"=>led<="011011011110";--merah_barat--
when "0001110"=>led<="011011011110";--merah_barat--
when "0001111"=>led<="011011011110";--merah_barat--
when "0010000"=>led<="011011011110";--merah_barat--
when "0010001"=>led<="011011011110";--merah_barat--
when "0010010"=>led<="011011011110";--merah_barat--
when "0010011"=>led<="011011011110";--merah_barat--
when "0010100"=>led<="011011011110";--merah_barat--
when "0010101"=>led<="011011011110";--merah_barat--
when "0010110"=>led<="011011011110";--merah_barat--
when "0010111"=>led<="011011011110";--merah_barat--
when "0011000"=>led<="011011011110";--merah_barat--
when "0011001"=>led<="011011011110";--merah_barat--

when "0011010"=>led<="011011101011";--merah_barat--
when "0011011"=>led<="011011101011";--merah_barat--
when "0011100"=>led<="011011101011";--merah_barat--
when "0011101"=>led<="011011101011";--merah_barat--
when "0011110"=>led<="011011101011";--merah_barat--

when "0011111"=>led<="011011110011";--merah_barat--
when "0100000"=>led<="011011110011";--merah_barat--
when "0100001"=>led<="011011110011";--merah_barat--
when "0100010"=>led<="011011110011";--merah_barat--
when "0100011"=>led<="011011110011";--merah_barat--
when "0100100"=>led<="011011110011";--merah_barat--
when "0100101"=>led<="011011110011";--merah_barat--
when "0100110"=>led<="011011110011";--merah_barat--
when "0100111"=>led<="011011110011";--merah_barat--
when "0101000"=>led<="011011110011";--merah_barat--
when "0101001"=>led<="011011110011";--merah_barat--
```

```
when "0101010"=>led<="011011110011";--merah_barat--
when "0101011"=>led<="011011110011";--merah_barat--
when "0101100"=>led<="011011110011";--merah_barat--
when "0101101"=>led<="011011110011";--merah_barat--
when "0101110"=>led<="011011110011";--merah_barat--
when "0101111"=>led<="011011110011";--merah_barat--
when "0110000"=>led<="011011110011";--merah_barat--
when "0110001"=>led<="011011110011";--merah_barat--
when "0110010"=>led<="011011110011";--merah_barat--

when "0110011"=>led<="101011011011";--kuning_barat--
when "0110100"=>led<="101011011011";--kuning_barat--
when "0110101"=>led<="101011011011";--kuning_barat--
when "0110110"=>led<="101011011011";--kuning_barat--
when "0110111"=>led<="101011011011";--kuning_barat--

when "0111000"=>led<="110011011011";--hijau_barat--
when "0111001"=>led<="110011011011";--hijau_barat--
when "0111010"=>led<="110011011011";--hijau_barat--
when "0111011"=>led<="110011011011";--hijau_barat--
when "0111100"=>led<="110011011011";--hijau_barat--
when "0111101"=>led<="110011011011";--hijau_barat--
when "0111110"=>led<="110011011011";--hijau_barat--
when "0111111"=>led<="110011011011";--hijau_barat--
when "1000000"=>led<="110011011011";--hijau_barat--
when "1000001"=>led<="110011011011";--hijau_barat--
when "1000010"=>led<="110011011011";--hijau_barat--
when "1000011"=>led<="110011011011";--hijau_barat--
when "1000100"=>led<="110011011011";--hijau_barat--
when "1000101"=>led<="110011011011";--hijau_barat--
when "1000110"=>led<="110011011011";--hijau_barat--
when "1000111"=>led<="110011011011";--hijau_barat--
when "1001000"=>led<="110011011011";--hijau_barat--
when "1001001"=>led<="110011011011";--hijau_barat--
when "1001010"=>led<="110011011011";--hijau_barat--
when "1001011"=>led<="110011011011";--hijau_barat--

when "1001100"=>led<="011101011011";--merah_barat--
when "1001101"=>led<="011101011011";--merah_barat--
when "1001110"=>led<="011101011011";--merah_barat--
when "1001111"=>led<="011101011011";--merah_barat--
when "1010000"=>led<="011101011011";--merah_barat--
```

```
                when "1010001"=>led<="011110011011";--merah_barat--
                when "1010010"=>led<="011110011011";--merah_barat--
                when "1010011"=>led<="011110011011";--merah_barat--
                when "1010100"=>led<="011110011011";--merah_barat--
                when "1010101"=>led<="011110011011";--merah_barat--
                when "1010110"=>led<="011110011011";--merah_barat--
                when "1010111"=>led<="011110011011";--merah_barat--
                when "1011000"=>led<="011110011011";--merah_barat--
                when "1011001"=>led<="011110011011";--merah_barat--
                when "1011010"=>led<="011110011011";--merah_barat--
                when "1011011"=>led<="011110011011";--merah_barat--
                when "1011100"=>led<="011110011011";--merah_barat--
                when "1011101"=>led<="011110011011";--merah_barat--
                when "1011110"=>led<="011110011011";--merah_barat--
                when "1011111"=>led<="011110011011";--merah_barat--
                when "1100000"=>led<="011110011011";--merah_barat--
                when "1100001"=>led<="011110011011";--merah_barat--
                when "1100010"=>led<="011110011011";--merah_barat--
                when "1100011"=>led<="011110011011";--merah_barat--
                when "1100100"=>led<="011110011011";--merah_barat--
                when others=>led<="000000000000";
                end case;
        end process;

end Behavioral;
```

```
----------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:    07:31:36 01/28/07
-- Design Name:
-- Module Name:    s7_barat - Behavioral
-- Project Name:
-- Target Device:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity s7_barat is
   Port ( out1 : in std_logic_vector(6 downto 0);
          tampil : out std_logic_vector(13 downto 0));
end s7_barat;

architecture Behavioral of s7_barat is

begin
        process(out1)
        begin

                case out1 is

                                when "0000001"=>tampil<="01001000000010";--26--
```

```
when "0000010"=>tampil<="01001001111000";--27--
when "0000011"=>tampil<="01001000000000";--28--
when "0000100"=>tampil<="01001000010000";--29--
when "0000101"=>tampil<="01100001000000";--30--
when "0000110"=>tampil<="01100001111001";--31--
when "0000111"=>tampil<="01100000100100";--32--
when "0001000"=>tampil<="01100000110000";--33--
when "0001001"=>tampil<="01100000011001";--34--
when "0001010"=>tampil<="01100000010010";--35--
when "0001011"=>tampil<="01100000000010";--36--
when "0001100"=>tampil<="01100001111000";--37--
when "0001101"=>tampil<="01100000000000";--38--
when "0001110"=>tampil<="01100000010000";--39--
when "0001111"=>tampil<="00110011000000";--40--
when "0010000"=>tampil<="00110011111001";--41--
when "0010001"=>tampil<="00110010100100";--42--
when "0010010"=>tampil<="00110010110000";--43--
when "0010011"=>tampil<="00110010011001";--44--
when "0010100"=>tampil<="00110010010010";--45--
when "0010101"=>tampil<="00110010000010";--46--
when "0010110"=>tampil<="00110011111000";--47--
when "0010111"=>tampil<="00110010000000";--48--
when "0011000"=>tampil<="00110010010000";--49--
when "0011001"=>tampil<="00100101000000";--50--
when "0011010"=>tampil<="00100101111001";--51--
when "0011011"=>tampil<="00100100100100";--52--
when "0011100"=>tampil<="00100100110000";--53--
when "0011101"=>tampil<="00100100011001";--54--
when "0011110"=>tampil<="00100100010010";--55--
when "0011111"=>tampil<="00100100000010";--56--
when "0100000"=>tampil<="00100101111000";--57--
when "0100001"=>tampil<="00100100000000";--58--
when "0100010"=>tampil<="00100100010000";--59--
when "0100011"=>tampil<="00000101000000";--60--
when "0100100"=>tampil<="00000101111001";--61--
when "0100101"=>tampil<="00000100100100";--62--
when "0100110"=>tampil<="00000100110000";--63--
when "0100111"=>tampil<="00000100011001";--64--
when "0101000"=>tampil<="00000100010010";--65--
when "0101001"=>tampil<="00000100000010";--66--
when "0101010"=>tampil<="00000101111000";--67--
when "0101011"=>tampil<="00000100000000";--68--
when "0101100"=>tampil<="00000100010000";--69--
```

```
when "0101101"=>tampil<="111100001000000";--70--
when "0101110"=>tampil<="111100001111001";--71--
when "0101111"=>tampil<="111100000100100";--72--
when "0110000"=>tampil<="111100000110000";--73--
when "0110001"=>tampil<="111100000011001";--74--
when "0110010"=>tampil<="111100000010010";--75--

when "0110011"=>tampil<="100000001111001";--01--
when "0110100"=>tampil<="100000000100100";--02--
when "0110101"=>tampil<="100000000110000";--03--
when "0110110"=>tampil<="100000000011001";--04--
when "0110111"=>tampil<="100000000010010";--05--

when "0111000"=>tampil<="100000001111001";--01--
when "0111001"=>tampil<="100000000100100";--02--
when "0111010"=>tampil<="100000000110000";--03--
when "0111011"=>tampil<="100000000011001";--04--
when "0111100"=>tampil<="100000000010010";--05--
when "0111101"=>tampil<="100000000000010";--06--
when "0111110"=>tampil<="100000001111000";--07--
when "0111111"=>tampil<="100000000000000";--08--
when "1000000"=>tampil<="100000000010000";--09--
when "1000001"=>tampil<="111100011000000";--10--
when "1000010"=>tampil<="111100011111001";--11--
when "1000011"=>tampil<="111100010100100";--12--
when "1000100"=>tampil<="111100010110000";--13--
when "1000101"=>tampil<="111100010011001";--14--
when "1000110"=>tampil<="111100010010010";--15--
when "1000111"=>tampil<="111100010000010";--16--
when "1001000"=>tampil<="111100011111000";--17--
when "1001001"=>tampil<="111100010000000";--18--
when "1001010"=>tampil<="111100010010000";--19--
when "1001011"=>tampil<="010010010000000";--20--

when "1001100"=>tampil<="100000001111001";--01--
when "1001101"=>tampil<="100000000100100";--02--
when "1001110"=>tampil<="100000000110000";--03--
when "1001111"=>tampil<="100000000011001";--04--
when "1010000"=>tampil<="100000000010010";--05--
when "1010001"=>tampil<="100000000000010";--06--
when "1010010"=>tampil<="100000001111000";--07--
when "1010011"=>tampil<="100000000000000";--08--
when "1010100"=>tampil<="100000000010000";--09--
```

```
                when "1010101"=>tampil<="111100011000000";--10--
                when "1010110"=>tampil<="111100111111001";--11--
                when "1010111"=>tampil<="111100101001100";--12--
                when "1011000"=>tampil<="111100101100000";--13--
                when "1011001"=>tampil<="111100100011001";--14--
                when "1011010"=>tampil<="111100100100010";--15--
                when "1011011"=>tampil<="111100100000010";--16--
                when "1011100"=>tampil<="111100111111000";--17--
                when "1011101"=>tampil<="111100100000000";--18--
                when "1011110"=>tampil<="111100100100000";--19--
                when "1011111"=>tampil<="010010010000000";--20--
                when "1100000"=>tampil<="010010011111001";--21--
                when "1100001"=>tampil<="010010001001000";--22--
                when "1100010"=>tampil<="010010001100000";--23--
                when "1100011"=>tampil<="010010000110010";--24--
                when "1100100"=>tampil<="010010000100100";--25--
                when others=>tampil<="000000000000000";
        end case;
    end process;

end Behavioral;
```

```
--------------------------------------------------------------------------------
-- Copyright (c) 1995-2003 Xilinx, Inc.
-- All Right Reserved.
--------------------------------------------------------------------------------
--   ____  ____
--  /   /\/   /
-- /___/  \  /    Vendor: Xilinx
-- \   \   \/     Version : 7.1i
--  \   \         Application : sch2vhdl
--  /   /         Filename : GAB_ALL.vhf
-- /___/   /\     Timestamp : 02/05/2007 06:33:26
-- \   \  /  \
--  \___\/\___\
--
--Command: "H:/Xilinx ISE 71.i/bin/nt/sch2vhdl.exe" -intstyle ise -family spartan2 -
flat -suppress -w GAB_ALL.sch GAB_ALL.vhf
--Design Name: GAB_ALL
--Device: spartan2
--Purpose:
--    This vhdl netlist is translated from an ECS schematic. It can be
--    synthesis and simulted, but it should not be modified.
--

library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
-- synopsys translate_off
library UNISIM;
use UNISIM.Vcomponents.ALL;
-- synopsys translate_on

entity GAB_ALL is
  port ( clk          : in    std_logic;
         rst          : in    std_logic;
         lampu_barat  : out   std_logic_vector (11 downto 0);
         lampu_selatan : out   std_logic_vector (11 downto 0);
         lampu_timur  : out   std_logic_vector (11 downto 0);
         lampu_utara  : out   std_logic_vector (11 downto 0);
         tampil_barat  : out   std_logic_vector (13 downto 0);
         tampil_selatan : out   std_logic_vector (13 downto 0);
         tampil_timur  : out   std_logic_vector (13 downto 0);
         tampil_utara  : out   std_logic_vector (13 downto 0));
end GAB_ALL;
```

```
architecture BEHAVIORAL of GAB_ALL is
  component Gab_line_barat
    port ( rst         : in    std_logic;
           clk         : in    std_logic;
           lampu       : out   std_logic_vector (11 downto 0);
           tampil_barat : out   std_logic_vector (13 downto 0));
  end component;

  component Gab_line_selatan
    port ( rst          : in    std_logic;
           clk          : in    std_logic;
           lampu        : out   std_logic_vector (11 downto 0);
           tampil_selatan : out   std_logic_vector (13 downto 0));
  end component;

  component Gab_line_timur
    port ( rst         : in    std_logic;
           clk         : in    std_logic;
           lampu       : out   std_logic_vector (11 downto 0);
           tampil_timur : out   std_logic_vector (13 downto 0));
  end component;

  component Gab_line_utara
    port ( rst         : in    std_logic;
           clk         : in    std_logic;
           lampu       : out   std_logic_vector (11 downto 0);
           tampil_utara : out   std_logic_vector (13 downto 0));
  end component;

begin
  XLXI_1 : Gab_line_barat
    port map (clk=>clk,
              rst=>rst,
              lampu(11 downto 0)=>lampu_barat(11 downto 0),
              tampil_barat(13 downto 0)=>tampil_barat(13 downto 0));

  XLXI_2 : Gab_line_selatan
    port map (clk=>clk,
              rst=>rst,
              lampu(11 downto 0)=>lampu_selatan(11 downto 0),
              tampil_selatan(13 downto 0)=>tampil_selatan(13 downto 0));
```

```
  XLXI_3 : Gab_line_timur
    port map (clk=>clk,
              rst=>rst,
              lampu(11 downto 0)=>lampu_timur(11 downto 0),
              tampil_timur(13 downto 0)=>tampil_timur(13 downto 0));

  XLXI_4 : Gab_line_utara
    port map (clk=>clk,
              rst=>rst,
              lampu(11 downto 0)=>lampu_utara(11 downto 0),
              tampil_utara(13 downto 0)=>tampil_utara(13 downto 0));

end BEHAVIORAL;
```