UNIT PENJUMLAH, PENGURANG DAN PENGALI 8 BIT BERBASIS FPGA

Tugas Akhir

Diajukan Sebagai Salah Satu Syarat Untuk Menyelesaikan

Pendidikan Program Strata 1 Pada Jurusan Teknik Elektro Fakultas

Teknologi Industri Universitas Islam Indonesia

Disusun Oleh:

NAMA : Adi Windro

NO.MHS : 00 524 141

JURUSAN TEKNIK ELEKTRO

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

YOGYAKARTA

2007

LEMBAR PENGESAHAN DOSEN PEMBIMBING

UNIT PENJUMLAH, PENGURANG DAN PENGALI 8 BIT BERBASIS FPGA

TUGAS AKHIR

Oleh :

Nama : Adi Windro

No. Mahasiswa : 00524141

Yogyakarta, April 2007

Pembimbing I,

(Ir. Hj. Budi Astuti, MT)

Pembimbing II,

(Tito Yuwono, ST, MSc.)

LEMBAR PENGESAHAN DOSEN PENGUJI

UNIT PENJUMLAH, PENGURANG DAN PENGALI 8 BIT BERBASIS FPGA

TUGAS AKHIR

Oleh :

Nama : Adi Windro

No. Mahasiswa : 00524141

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

Tim Penguji,

Ir. Hj. Budi Astuti, MT
Ketua

Tito Yuwono, ST, Msc.
Anggota I

Dwi Ana Ratna Wati, ST
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.

Adikku tercinta yang selalu membuat hidupku begitu bersemangat

Penjaga hatiku yang selalu meredakan emosiku dan membuat hidupku lebih berwarna

Sahabat seperjuangan..Galih, semoga dapat menyelesaikan skripsinya juga

MOTTO

"Tiada seseorang muslim yang mendoakan saudaranya tanpa sepengetahuan saudaranya, kecuali malaikat berkata-kata: "Dan untuk kamu pula seperti itu.""

(HR. Muslim)

"Janganlah kamu sekali-kali meremehkan kebaikan, walaupun hanya dengan bermuka manis apabila kamu berjumpa dengan saudaramu."

(HR. Muslim)

Pandanglah orang yang lebih rendah daripadamu. Jangan memandang kepada orang yang lebih tinggi daripadamu. Karena yang demikian itu lebih baik, agar kamu tidak meremehkan nikmat karunia Allah yang telah dianugerahkan kepadamu.

(HR Bukhori dan Muslim)

KATA PENGANTAR

بسم الله الرحمن الرحيم

Assalamu'alaikum Wr. Wb.

Puji syukur kehadirat Allah SWT, atas nikmat iman, rahmat, hidayah dan pikiran yang diberikan. Sehingga penulis dapat menyelesaikan tugas akhir dengan judul "Penjumlahan, Pengurangan Dan Perkalian 8 Bit Berbasis FPGA". Tidak lupa shalawat serta salam selalu tercurah kepada Nabi Muhammad. SAW beserta keluarga dan para sahabatnya.

Adapun maksud dari penyusunan tugas akhir ini adalah untuk memenuhi kurikulum S-1 Jurusan Teknik Elektro, Fakultas Teknologi Industri, Universitas Islam Indonesia. Disamping itu juga untuk menambah pengetahuan atas disiplin ilmu yang telah dipelajari di bangku perkuliahan untuk diterapkan ke masyarakat.

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. Bpk Fathul Wahid, ST, M.Sc, selaku Dekan Fakultas Teknologi Industri Universitas Islam Indonesia.
2. Bpk Tito Yuwono, ST, M.Sc, selaku Kajur Teknik Elektro sekaligus sebagai dosen pembimbing 2 saya yang telah banyak sekali membantu saya.

3. Serta Ir. Hj. Budi Astuti, MT, sebagai dosen pembimbing 1 dan juga semua dosen Teknik Elektro UII yang telah memberikan ilmunya.
4. Mba' Umi yang banyak membantu.Mas Heri dan Mas Tri serta serta seluruh teman-teman Lab Digital dan Lab Kendali terima kasih atas bantuannya dan telah mau berbagi ilmu.
5. Ayahanda dan ibunda yang tidak henti-henti berdo'a dan memberikan semangat kepada saya.
6. Kakak dan adik serta saudara-saudara ku yang telah memberikan saran dan do'anya.
7. Mama Solo yang selalu memberi semangat, menghibur, terimakasih atas kebaikan mu.
8. Teman-teman satu kos dan seluruh pihak yang tidak dapat di sebutkan satu-persatu, yang telah memberikan dukungan dan doa.

Penulis menyadari bahwa Tugas Akhir ini masih terdapat kesalahan dan kekurangan. Oleh karena itu, kritik dan saran yang membangun akan senantiasa penulis terima dengan senang hati. Besar harapan laporan ini dapat bermanfaat kepada penulis pada khususnya dan pembaca pada umumnya, amin.

Wassalamu'alaikum Wr.Wb

Jogjakarta, April 2007

Adi Windro

DAFTAR ISI

Halaman Judul i
Lembar Pengesahan Pembimbing ii
Lembar Pengesahan Penguji iii
Halaman Persembahan iv
Halaman Motto v
Kata Pengantar vi
Abstraksi viii
Daftar Isi ix
Daftar Gambar xii
Daftar Tabel xiv

Bab I PENDAHULUAN

1.1 Latar Belakang 1
1.2 Maksud dan Tujuan 2
1.3 Rumusan Masalah 3
1.4 Batasan Masalah 3
1.5 Manfaat Penulisan Skripsi 3
1.6 Metodologi Penelitian 4
1.7 Sistematika Penulisan 5

Bab II LANDASAN TEORI

2.1 Sistem Bilangan 6
2.2.1 Sistem Bilangan Desimal dan Biner 6
2.2.2 Sistem Bilangan Heksadesimal 8
2.2 Teknik Pencacah 10
2.2.1 Pencacah Biner 4-Bit 11
2.2.2 Pencacah Tak Serempak 12
2.2.3 Pencacah Paralel 13
2.2.4 Pencacah BCD 14

Gambar 3.2 Blok diagram dari sistem 42
Gambar 3.3 Blok diagram Penjumlah dan Pengurang 8 bit 43
Gambar 3.4 Mensinkronkan input yang lebih lebar terhadap *clock* 43
Gambar 3.5 Mendapatkan mode pilihan dari 0 sampai 2 44
Gambar 3.6 Hasil 8-bit diubah menjadi format desimal 45
Gambar 3.7 *Flowchart* Penjumlah dan Pengurang 8 bit 47
Gambar 4.1 Skematik Blok Kalkulasi 50

DAFTAR TABEL

Tabel 2.1 Angka-angka Heksadesimal 8
Tabel 2.2 Data keluarga Spartan II 23
Tabel 2.3 Masukan katoda untuk digit desimal 35
Tabel 2.4 Konfigurasi pin yang terdapat pada modul pegasus 36
Tabel 2.5 *Port accessory* 37
Tabel 2.6 Pin FPGA XC2S50 37
Tabel 2.7 Konfigurasi pin yang terdapat pada modul pegasus 38
Tabel 2.8 Tabel keadaan sistem digital 39
Tabel 2.9 Data keluarga Spartan II 29
Tabel 2.10 Masukan katoda untuk digit desimal 41
Tabel 2.11 Konfigurasi pin yang terdapat pada modul pegasus 42
Tabel 2.12 *Port accessory* 42
Tabel 2.13 Pin FPGA XC2S50 43
Tabel 3.1 Dekoder pengubah Heksadesimal menjadi desimal 45

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 alat ini adalah bilangan A dan bilangan B yang merupakan masukan dari keypad eksternal yang dirancang khusus. Pilihan kalkulasi dilakukan dengan menekan salah satu tombol yang terdapat pada keypad. Operasi dalam alat ini meliputi operasi penjumlahan, pengurangan dan perkalian. Hasil dari operasi di dalam FPGA masih berupa bilangan heksadesimal. Untuk itu hasil kalkulasi akan dikodekan twerlebih dahulu dan ditampilkan dalam tampilan 7-ruas dalam format desimal.

Dengan dirancangnya alat ini diharapkan mahasiswa dapat Mengetahui proses aritmatika khususnya penjumlahan, pengurangan dan perkalian pada suatu mesin hitung (dasar perhitungan pada komputer), dapat mensimulasikan suatu unit digital yang dapat digunakan untuk menyelesaikan operasi aritmatik penjumlahan, pengurangan dan perkalian, dapat Memprogram suatu FPGA supaya dapat digunakan untuk menyelesaikan operasi aritmatik penjumlahan, pengurangan dan perkalian, dan dapat Membuat prototipe aplikasi FPGA yang berupa unit penjumlahan, pengurangan dan perkalian.

2.2.5 Tampilan Alfanumerik 16
2.3 Aritmatik Biner 17
2.3.1 Penambahan Biner 17
2.3.2 Pengurangan Biner 18
2.3.3 Perkalian Biner (*Binary Multiplication*) 19
2.4 Tombol Masukan 20
2.5 Penampil Tujuh Segmen 21
2.6 *Field Programmable Gate Array*(FPGA) 22
2.3.1 FPGA Keluarga Xilinx Spartan II 23
2.3.2 Struktur Dasar Keluarga Spartan II 24
2.3.3 *Input/Output blocks* (IOB) 24
2.3.4 *Configurable Logic Block* (CLB) 25
2.3.5 *Programmable Routing Matrix* 26
2.3.6 Pemrograman FPGA 28
2.7 Modul Pegasus 29
2.7.1 Port JTAG dan Mengkonfigurasi Modul 30
2.7.2 *Power Supply* 32
2.7.3 *Oscilator* 32
2.7.4 Saklar *Pushbuttons*, Saklar Geser, Indikator Led 32
2.7.5 *Seven-segment* 33
2.7.6 *Port* I/O Tambahan 35
2.8 Sistem Digital 39

Bab III PERANCANGAN SISTEM

3.1 Perancangan unit penjumlah dan pengurang 8 bit 40
3.1.1 *User*/ Pengguna 41
3.1.2 Pengendali (*Controller*) 41
3.1.3 *Display* 41
3.2 Perancangan *Software* 41
3.2.1 Blok Sinkronisasi Input 43
3.2.2 Blok pemilih Mode Kalkulasi 44

3.2.3 Blok Penjumlah dan Pengurang 8 bit 44
3.2.4 Blok Tampilan ke Seven Segment 45
3.2.5 Blok Sinkronisasi Dekoder 45
3.3 Program Utama 46

Bab IV ANALISA DAN PEMBAHASAN
4.1 Analisis Sistem 48
4.2 Pembahasan 49

Bab V PENUTUP
5.1 Kesimpulan 50
5.2 Saran 51

DAFTAR GAMBAR

Gambar 2.1 (a) Pencacah empat bit, (b) Bentuk gelombang 11
Gambar 2.2 (a) Diagram logika, (b) Bentuk gelombang 12
Gambar 2.3 Pencacah biner paralel mod-8 (a) Diagram logika
(b) Tabel kebenaran, (c) Bentuk gelombang 14
Gambar 2.4 Pencacah BCD 2421. (a) Diagram logika
(b) Tabel kebenaran, (c) Bentuk gelombang 16
Gambar 2.5 Sistem tampilan dasar 17
Gambar 2.6 Aturan untuk penambahan biner 17
Gambar 2.7 Aturan untuk pengurangan biner 18
Gambar 2.8 Persoalan pengurangan biner yang menunjukkan
suatu pinjaman 19
Gambar 2.9 Aturan-aturan untuk perkalian biner 19
Gambar 2.10 Persoalan perkalian biner sederhana 19
Gambar 2.11 Penampil tujuh segmen 21
Gambar 2.12 Tujuh segmen dalam digit desimal 22
Gambar 2.13 Rangkaian penampil tujuh segmen anoda bersama 22
Gambar 2.14 Diagram Blok Dasar Keluarga Spartan II 24
Gambar 2.15 Blok IOB Spartan II 25
Gambar 2.16 CLB pada Spartan II 26
Gambar 2.17 Struktur *Local Routing* 27
Gambar 2.18 Koneksi BUFT untuk *dedicated Horizontal Bus Line* 28
Gambar 2.19 Blok Diagram Pegasus 30
Gambar 2.20 Aliran scan JTAG pada Pegasus 31
Gambar 2.21 Rangkaian saklar *pushbutton*, saklar geser, LED 33
Gambar 2.22 *Common anode seven-segment* 4 *digit* 33
Gambar 2.23 *Common anode seven-segment* 1 *digit* 34
Gambar 2.24 Diagram waktu sinyal *seven-segment* 35
Gambar 2.25 *Pin* penghubung tambahan 36
Gambar 3.1 Blok diagram Penjumlah dan Pengurng 8 bit 40

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pada masa yang sudah bertambah maju ini, alat elektronis bisa dikatakan sudah merajai dunia. Manusia terkesan dibodohi atau keyakinan dirinya semakin berkurang, sebagai contoh adalah penggunaan mesin hitung atau yang lebih dikenal sebagai kalkulator. Dengan adanya kalkulator, orang yang dulunya sewaktu duduk di sekolah dasar bisa merasa yakin pada hasil pemikirannya berkaitan dengan operasi aritmatik sederhana sekarang menjadi kurang yakin.

Melihat kenyataan ini, mahasiswa Tehnik Elektro yang sering dianggap sebagai orang yang lebih mengerti akan elektronika, setidaknya mengetahui prinsip kerja alat elektronis yang dapat berfungsi sebagai operator aritmatika. Sebagai mahasiswa elektro, mengetahui prinsip kerja dari suatu alat elektronis yang berfungsi sebagai operator aritmatika, sehingga tidak hanya sekedar bisa menggunakan saja. Karena zaman sudah semakin modern maka komponen-komponen yang digunakan sudah bukan lagi berupa resistor, capasitor, diode dan transistor tetapi dapat diwakili oleh sebuah IC saja. Dengan demikian untuk mempelajari prinsip kerja dari operator aritmatik ini tidak perlu harus membuat rangkaian secara nyata tetapi cukup dengan memprogram sebuah IC. Hal ini tentu saja membuat perancangan elektronis menjadi lebih efisien, selain bentuknya bisa lebih kecil juga perakitannya bisa lebih rapi dengan bantuan pabrik

Begitu juga dengan perkembangan FPGA (*Field-Programable Gate Arrays*), walaupun sudah sejak lama berkembang di dunia, tapi di Indonesia masih sedikit sekali yang menggunakanya bahkan mengetahuinya. Di Negara-negara maju FPGA telah banyak digunakan, karena FPGA dapat mengimplementasikan rangkaian kombinasi maupun rangkaian skuensial, tanpa terhambat kekurangan register. Salah satu bahasa yang digunakan adalah VHDL (VHSIC *Hardware Description Language*) dan Schematic. Dalam FPGA kita bukannya membuat program, tapi lebih tepatnya adalah merekonfigurasi FPGA.

1.2 Maksud dan Tujuan

Adapun maksud dan tujuan dari skripsi ini selain sebagai salah satu syarat untuk memenuhi kurikulum S-1 Jurusan Teknik Elektro Fakultas Teknologi Industri Universitas Islam Indonesia juga untuk:

1. Mensimulasikan suatu unit digital yang dapat digunakan untuk menyclesaikan operasi aritmatik penjumlahan, pengurangan dan perkalian.
2. Memprogram suatu FPGA supaya dapat digunakan untuk menyelesaikan operasi aritmatik penjumlahan, pengurangan dan perkalian.
3. Membuat prototipe aplikasi FPGA yang berupa unit penjumlahan, pengurangan dan perkalian.

1.3 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan diatas, maka dapat di ambil suatu rumusan masalah sebagai berikut; "Bagaimana cara merancang unit penjumlahan, pengurangan dan perkalian 8 bit berbasis FPGA"

1.4 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. Data masukan maupun hasil berupa data integer hexadesimal(positif).
b. Data masukan untuk unit penjumlahan, pengurangan dan perkalian adalah data 8 bit(2 digit hexadesimal).
c. Jenis operasi bisa dilakukan ada 3 yaitu penjumlahan, pengurangan dan perkalian.

1.5 Manfaat Penulisan Skripsi

Adapun manfaat yang dapat diambil dalam melakukan penelitian ini adalah:

a. Mengetahui proses aritmatika khususnya penjumlahan, pengurangan dan perkalian pada suatu mesin hitung (dasar perhitungan pada komputer).

b. Dapat menggunakan FPGA untuk proses penjumlahan dan pengurangan.

1.6 Metodologi Penelitian

Dalam melaksanakan studi dan penulisan ini, maka diperlukan tahap-tahap yang perlu dilalui. Hal ini berguna agar diperoleh suatu hasil yang maksimal. Adapun metode yang digunakan agar didapat data-data yang sesuai dengan apa yang diharapkan adalah:

1. **Studi Literatur**

   Merupakan metode pengumpulan bahan-bahan yang diperoleh dari buku-buku literatur yang menyangkut bidang yang sedang dibahas serta membandingkan dan menerapkannya pada masalah yang akan dibahas tersebut. Dalam metode pengumpulan data secara literatur ini dilakukan dengan cara mengadakan suatu riset perpustakaan yang berdasarkan kepada sumber-sumber literatur yang diinginkan. Sumber-sumber literatur tersebut diperoleh dari perpustakaan-perpustakaan maupun sumber lain.

2. **Implementasi Alat**

   Merupakan metode pengumpulan data dengan cara pengamatan langsung pada suatu objek yang hendak diamati juga mencakup pengalaman yang diperoleh sendiri terhadap alat yang bersangkutan dan dalam penyusunan laporan ini mencakup perangkat keras (*hardware*) dan perangkat lunak (software) dari FPGA.

1.7 Sistematika Penulisan

Dalam penulisan, sistematika penulisanya terdiri dari lima bab yaitu :

BAB I PENDAHULUAN

Bab ini berisi tentang latar belakang masalah, perumusan masalah, batasan masalah, tujuan penulisan dan sistematika penulisan.

BAB II DASAR TEORI

Bab ini berisi tentang teori-teori yang digunakan dalam perancangan dan pembuatan peralatan.

BAB III PERANCANGAN SISTEM

Bab ini menjelaskan perancangan sistem, komponen yang digunakan serta penjelasannya dan desain perangkat kerasnya.

BAB IV ANALISA DAN PEMBAHASAN

Bab ini akan menjelaskan hasil dari pengujian alat yang dibuat dan akan dibandingkan dengan teori yang digunakan.

BAB V PENUTUP

Bab ini berisi kesimpulan-kesimpulan dari peralatan yang dibuat dan berisi saran-saran guna pengembangan dimasa yang akan datang.

BAB II

LANDASAN TEORI

2.1 Sistem Bilangan

Sistem bilangan tidak lain adalah merupakan suatu sandi bagi masing-masing kuantitas yang bersangkutan.Setelah menghafalkan sandi, yang bersangkutan dapat mencacah. Dalam hal ini mengarah kepada aritmatika maupun bentuk-bentuk matematika yang lebih tinggi.

Sistem bilangan yang paling umum dan banyak digunakan pada saat ini adalah sistem desimal yang menggunakan sepuluh lambang bilangan yaitu 0, 1, 2, 3, 4, 5, 6, 7, 8, dan 9. Penerapan bilangan yang ingin dinyatakan hanya menggunakan kombinasi dari kesepuluh angka tersebut. Selain sistem desimal, terdapat sistem bilangan lain yang umum digunakan dalam bidang teknik digital yaitu sistem biner, sistem oktal dan sistem heksadesimal yang masing-masing memiliki basis 2, basis 8 dan basis 16. Basis dari bilangan menunjukkan banyaknya lambang bilangan yang digunakan dan biasanya dinyatakan dalam subsrip, misalnya seperti 190, 1901, 100102, 4038, 1A6BF16

2.1.1 Sistem Bilangan Desimal dan Biner

Sistem bilangan desimal adalah sistem bilangan yang biasa digunakan dalam kehidupan sehari-hari yaitu sistem bilangan dengan basis 10 yang simbol bilangannya adalah 0, 1, 2, 3, 4, 5, 6, 7, 8, dan 9, penggunaan 0 sampai 9 tidak merupakan keharusan. Bagaimanapun juga, karena sistem bilangan hanyalah

suatu sandi dapat menggunakan berapapun banyaknya lambang yang kita inginkan. Sistem bilangan biner adalah suatu sandi yang hanya menggunakan dua lambang dasar, lambang yang lazim digunakan untuk sistem biner ini adalah 0 dan 1.

Suatu bilangan dapat dikonversikan kedalam bentuk sistem bilangan yang lainnya, hal ini paling mudah dapat dipahami dengan cara melihat secara langsung dari contoh-contoh sebagai berikut:

1. Konversi desimal ke biner

   $176 = \ldots\ldots\ldots_2$

   176 : 2 = 88 sisa 0 (LSB)

   88 : 2 = 44 sisa 0

   44 : 2 = 22 sisa 0

   22 : 2 = 11 sisa 0

   11 : 2 = 5 sisa 1

   5 : 2 = 2 sisa 1

   2 : 2 = 1 sisa 0

   1 : 2 = 0 sisa 1 (MSB)

   Jadi $176_{10} = 10110000_2$

2. Konversi biner ke desimal

   $10101101_2 = (1X2^7) + (1X2^5) + (1X2^3) + (1X2^2) + (1X2^0)$

   $= 128 + 32 + 8 + 4 + 1$

   $= 173$

   Jadi $10101101_2 = 173_{10}$

2.1.2 Sistem Bilangan Heksadesimal

Sistem bilangan heksadesimal mempunyai basis 16, walaupun dapat digunakan 16 buah lambang yang manapun, namun lazimnya digunakan 0 sampai 9 dan A sampai F, seperti terlihat pada Tabel 2.1. setelah mencapai 9 pada sistem heksadesimal pencacahan dilanjutkan dengan A, B, C, D, E, F.

Setelah kehabisan lambang dasar, dapat dibentuk kombinasi 2 angka, dengan mengambil angka kedua diikuti oleh angka pertama, kemudian angka kedua diikuti oleh angka kedua dan seterusnya.

Tabel 2.1 Angka-angka Heksadesimal

Desimal	Biner	Heksadesimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

1. Konversi heksadesimal ke desimal

Untuk mengubah bilangan heksadesimal ke ekivalen desimalnya adalah dengan cara menjumlahkan hasil kali antara angka-angka heksadesimal bobot-bobotnya. Salah satu contohnya adalah sebagai berikut:

$$F8E6 = F(16^3) + 8(16^2) + E(16^1) + 6(16^0)$$

$$= 15(16^3) + 8(16^2) + 14(16^1) + 6(16^0)$$

$$= 63718_{10}$$

2. Konversi desimal ke heksadesimal

Suatu cara untuk melaksanakan perubahan desimal ke heksadesimal adalah mengubah dari desimal ke biner dan kemudian ke heksadesimal. Jika dikehendaki perubahan secara langsung, dapat digunakan metode *hex-dabble* (heksa–plus sisa). Gagasannya adalah membagi secara berturut-turut dengan 16 sambil menuliskan sisa-sisanya.

Berikut diperlihatkan contoh cara pengerjaannya yaitu mengubah 2479 ke heksa desimal:

2479 : 16 = 154 sisa 15 = F (LSB)

154 : 16 = 9 sisa 10 = A

9 : 16 = 0 sisa 9 = 9 (MSB)

Jadi heksadesimal dari 2479 adalah 9AF

3. Konversi Heksadesimal ke Biner

Untuk mengubah bilangan heksadesimal ke bilangan biner adalah dengan mengubah masing-masing angka heksadesimal kedalam 4 bitnya dengan

menggunakan sandi pada Tabel 2.1. Sebagai contoh, berikut adalah perubahan 9AF ke biner:

9	A	F
1001	1010	1111

Jadi ekivalen biner dari 9AF adalah 1001 1010 1111

4. Konversi Biner ke Heksadesimal

Untuk mengubah dari biner ke heksadesimal juga dapat digunakan Tabel 2.8 sebagai contoh heksadesimal dari biner 1000 1100 adalah :

1000	1100
8	C

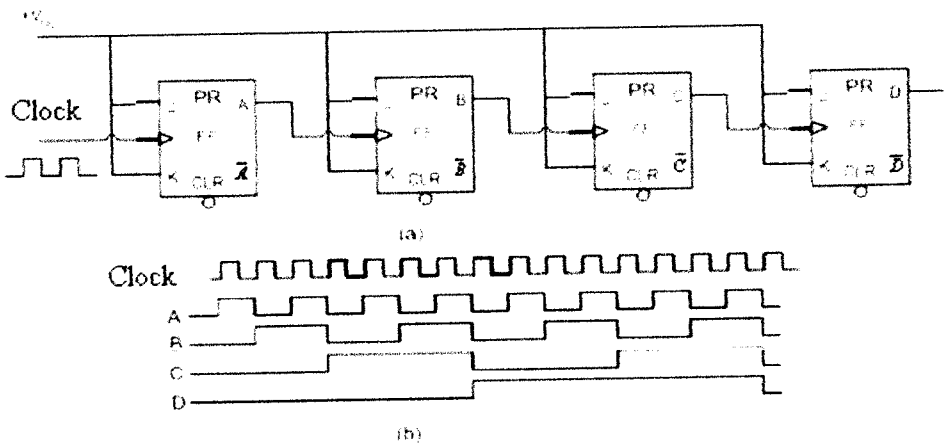
Jadi ekivalen heksadesimal dari biner 1000 1100 adalah 8C.

2.2 Teknik Pencacah

Pencacah merupakan salah satu sub sistem yang paling berguna dan paling banyak kemampuannya dalam suatu sistem digital. Pencacah yang di *drive* oleh sebuah *clock* dapat digunakan untuk mencacah banyaknya siklus *clock*. Karena pulsa *clock* terjadi pada selang waktu yang diketahui, pencacah dapat digunakan sebagai suatu instrumen untuk mengukur waktu (dan dengan demikian periode atau frekuensi).

2.2.1 Pencacah Biner 4-Bit

Flip-flop dapat dihubungkan untuk mendapatkan sebuah pencacah elektronik; suatu unit yang mencacah banyaknya picu masukan. Gambar 2.8 memperlihatkan empat buah flip-flop dalam gandengan. Suatu gelombang segi empat mendrive flip-flop A. Gelombang ini disebut *clock*.Keluaran flip-flop A mendrive flip-flop B, flip-flop B selanjutnya mendrive flip-flop C, yang selanjutnya mendrive flip-flop D. Semua masukan J dan K dihubungkan ke +Vcc. Ini berarti masing-masing flip-flop akan berubah keadaan *(toggle)* akibat peralihan negatif pada masukan *clock*-nya.

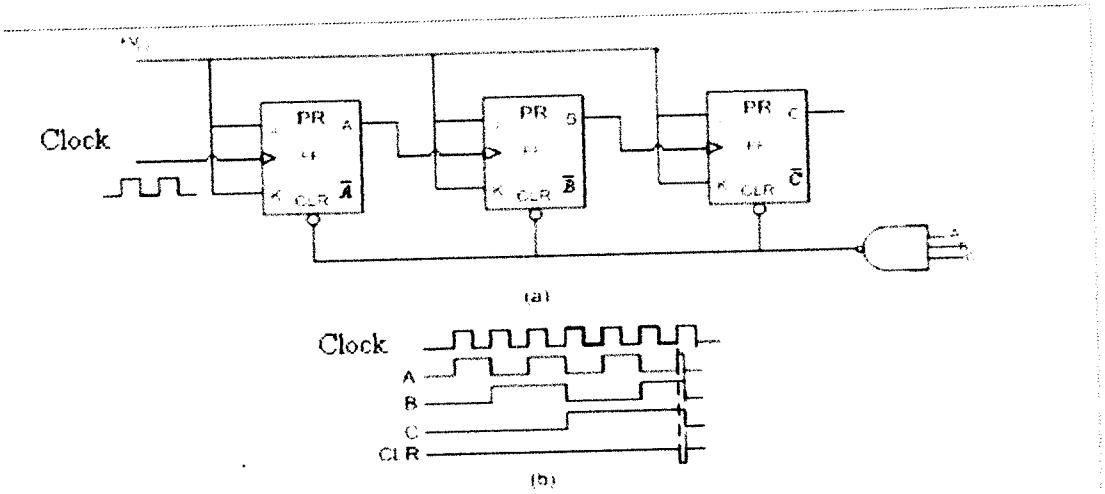


Gambar 2.1 (a) Pencacah empat bit; (b) Bentuk gelombang.

Bila keluaran suatu flip-flop lainnya, dinamakan *Ripple counter* atau pencacah tak serempak; flip-flop A harus berubah keadaan sebelum dapat memicu flip-flop B; B harus berubah sebelum dapat memicu C, dan seterusnya.

2.2.2 Pencacah Tak Serempak

Salah satu metode yang digunakan untuk membuat sebuah pencacah melompati cacahan-cacahan tertentu adalah dengan mencatu-balikkan suatu sinyal dari beberapa flip-flop tertentu ke flip-flop yang mendahuluinya. Diperhatikan pencacah yang terlihat pada Gambar 2.2, karena hanya sebuah cacahan yang harus dilompati, akan sangat memudahkan bila kita membuat pencacah melewati urutan dasarnya dan kemudian meresetnya satu cacahan lebih awal.



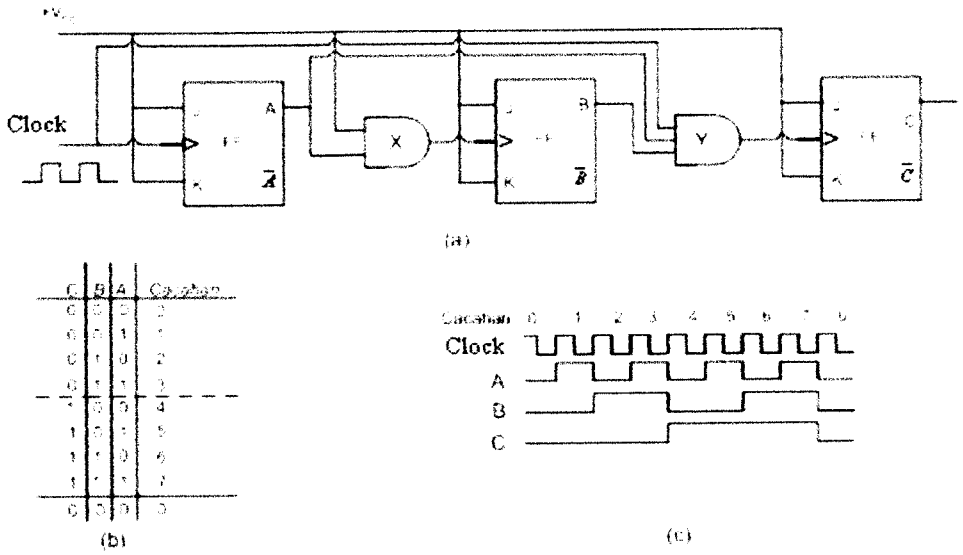
Gambar 2.2 (a) Pencaca Tak Serempak; (b) Bentuk gelombang

Jika A,B,dan C dihubungkan kesebuah gerbang NAND seperti terlihat pada Gambar 2.2 keluaran gerbang NAND akan menjadi rendah hanya pada saat A = 1, B = 1, dan C = ! (cacahan 7). Jika keluaran gerbang NAND ini dihubungkan ke CLR pada masing-masing flip-flop, semua flip-flop akan direset ke nol secara langsung ketika pencacah mencapai cacahan 7. Jenis pencacahan ini kadang-kadang disebut "pencacah permutasi" karena cacahan dasarnya telah dipermutasi (diubah).

2.2.3 Pencacah Paralel

Ripple Counter paling sederhana pembuatannya, namun terdapat suatu keterbatasan pada frekuensi kerja maksimumnya. Pada *ripple counter* waktu-waktu tunda ini bersifat aditif (ditambah), dan waktu penyelesaian keseluruhan bagi pencacah yang bersangkutan adalah sekitar waktu tunda dikalikan dengan banyaknya flip-flop seluruhnya. Keterbatasan kecepatan ini dapat ditanggulangi dengan penggunaan pencacah serempak atau paralel. Perbedaanya dalam hal ini adalah bahwa setiap flip-flop dipicu oleh *clock*. Maka, semuanya mengalami peralihan secara serempak.

Konstruksi suatu pencacah biner paralel diperlihatkan pada Gambar 2.3 bersama tabel kebenaran dan bentuk gelombangnya bagi urutan cacahan dasar. Flip-flop A pada gambar 2.3 berubah keadaan oleh setiap peralihan negatif pada masukan lonceng. Keluaran gerbang AND X menjadi tinggi bilamana *clock* adalah tinggi dan A adalah tinggi. Maka flip-flop B berubah keadaan pada setiap pulsa *clock* berikutnya. Keluaran gerbang AND Y menjadi tinggi setiap kali lonceng serta A dan B menjadi tinggi. Maka flip-flop C berubah keadaan oleh setiap pulsa *clock* keempat. Semuanya ini mewujudkan sebuah pencacah biner paralel mod-8



Gambar 2.3 Pencacah biner paralel mod-8 (a) Diagram logika; (b) Tabel kebenaran; (c) bentuk gelombang

Pencacah paralel yang telihat pada Gambar 2.3 dapat digunakan sebagai dasar untuk membangun pencacah dengan modul lain.

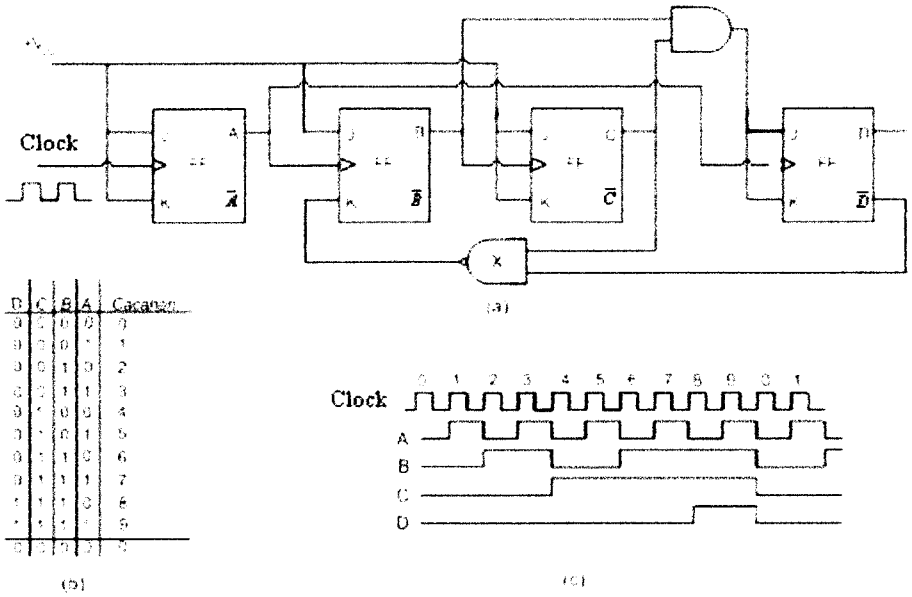
2.2.4 Pencacah BCD

Untuk membangun pencacah-pencacah yang bukan mencacah dalam sandi biner langsung. Suatu bentuk lain yang sangat bermanfaat adalah pencacah dekade yang mencacah dalam desimal bersandi biner termodifikasi 2421. Pencacah ini diperlihatkan pada Gambar 2.4 operasi pencacah ini dapat dijelaskan sebagai berikut:

1. A harus berubah keadaan setiap kali pulsa lonceng menurun, dan dengan demikian masukan *clock* ke *flip-flop* A haruslah *clock* itu sendiri.

2. B harus berubah keadaan setiap kali A menurun kecuali pada peralihan dari cacahan 7 ke cacahan 8. Keluaran gerbang NAND X adalah rendah bilamana C dan $\overline{D}$ keduanya tinggi. Hal ini mencegah B agar tidak menurun pada peralihan dari cacahan 7 ke cacahan 8, karena keluaran gerbang NAND X menahan masukan X ke flip-flop B tetap rendah sepanjang cacahan 4,5,6, dan 7. Dapat diperhatikan bahwa masukan J ke flip-flop B selalu TINGGI (benar), dan dengan demikian B dapat menjadi TINGGI pada peralihan dari cacahan 5 ke cacahan 6.
3. Masukan *clock* ke *flip-flop* C didrive oleh B, karena C harus berubah keadaan setiap kali B menurun.
4. D harus berubah keadaan setiap kali B dan C keduanya TINGGI dan A menurun. Jika masukan J dan K ke *flip-flop* D ditentukan oleh keluaran sebuah gerbang AND yang masukan-masukannya adalah B dan C, maka D dapat berubah keadaan bila B dan C keduanya TINGGI. Selanjutnya jika masukan lonceng ke *flip-flop* D didrive oleh A, D akan berubah keadaan pada peralihan dari 7 ke 8 dan dari 9 ke 0

Pencacah ini merupakan pencacah dekade, dan oleh sebab itu hanya memiliki 10 keadaan. Karena digunakan empat buah *flip-flop*, maka terdapat enam keadaan tak absah (diabaikan) yang harus dikaji. Pencacah BCD yang terlihat pada Gambar 2.4 mewakili salah satu metode untuk mewujudkan pencacah ini.

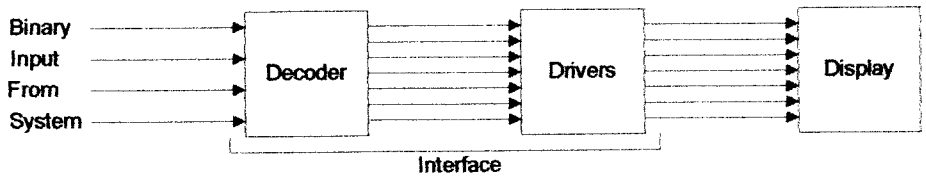


Gambar 2.4 Pencacah BCD 2421. (a) Diagram logika; (b) Tabel kebenaran; (c) Bentuk gelombang.

2.2.5 Tampilan Alfanumerik

Tampilan alfanumerik biasa untuk menampilkan pembacaan alat ukur digital atau data digital lain atau sistem *mikroposessor*. Sejumlah teknik bisa digunakan untuk menampilkan bilangan dan atau sejumlah alfabet dan sejumlah karakter - karakter lain seperti tanda tanya atau tanda seru. Jenis yang paling sering digunakan adalah tampilan tujuh segmen yang cocok untuk sistem desimal dan heksadesimal. Teknik lain adalah dengan menggunakan matriks titik yang mempunyai kualitas yang lebih baik dan cacah karakter yang lebih banyak. Sistem tampilan dasar tersaji pada Gambar 2.5 Keluaran biner dari sistem digital pertama kali harus diawa-sandikan kedalam bentuk digital yang cocok dengan jenis tampilan yang digunakan dan kemudian diumpankan ketingkat penggerak

sebelum ke piranti tampilan. Dengan demikian, untai antar muka terdiri dari dua komponen dasara: pengawa-sandi dan penggerak.



Gambar 2.5 Sistem tampilan dasar

2.3 Aritmatik Biner

2.3.1 Penambahan Biner

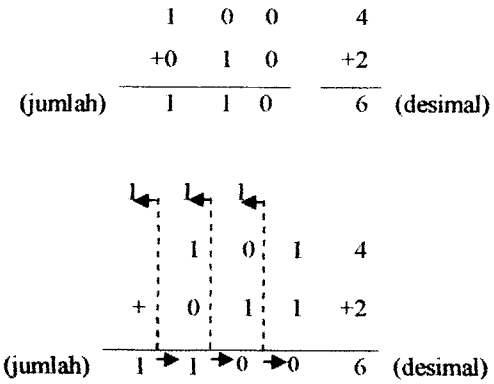
Penambahan bilangan biner merupakan suatu ugas yang sangat sederhana. Aturan – aturan (tabel-tabel penambahan) untuk penambahan biner yang menggunakan dua bit, dipaparkan pada gambar 2.6. Aturan 4 mengatakan bahwa, dalam biner 1+1 = 1 0 (desimal 2). Angka 1 dalam penambahan tersebut harus dibawa ke kolom berikutnya seperti dalam kolom penambahan desimal yang biasa.

					Jumlah	Bawaan keluar
Aturan 1	0	+	0	=	0	
Aturan 2	0	+	1	=	1	
Aturan 3	1	+	0	=	1	
Aturan 4	1	+	1	=	0	Dan bawaan 1 = 1 0

Simbol + berarti tambah

Gambar 2.6 Aturan untuk penambahn biner

Dua contoh persoalan penambahan biner diilustrasikan sebagai berikut :



2.3.2 Pengurangan Biner

Aturan-aturan untuk pengurangan biner dari dua bit diilutrasikan pada gambar 2.7. Angka sebelah atas dalam suatu persoalan pengurangan disebut *minumen* (yang dikurangi). Angka sebelah bawah disebut pengurang *(substrahend)*, dan jawabannya disebut selisih *(difference)*. Perhatikan kolom 1 dari bilangan biner pada gambar 2.8, angka 1 dikurangkan dari 0. angka 1 dipinjam dari kolom 2 biner, sehingga tinggal 0 pada kolom tersebut. Pengurang 1 dikurangkan dari minuen 10 (desimal 2). Pengurang ini menghasilkan selisih 1 pada kolom 1. kolom 2 biner menggunakan aturan 1 (0-0) dan sama dengan 0. maka, aturan 2 adalah 0 – 1 = 1 dengan pinjaman 1.

	Minuend		Subtrahend		Selisih	Pinjaman keluar
Aturan 1	0	-	0	=	0	
Aturan 2	0	-	1	=	1	Pinjam 1
Aturan 3	1	-	0	=	1	
Aturan 4	1	-	1	=	0	

Gambar 2.7 Aturan untuk pengurangan biner

		Biner	Desimal
	pinjam		
Minuend	10	10	2
Subtrahend	- 0	1	-1
Selisih	0	1	1

Gambar 2.8 Persoalan pengurangan biner yang menunjukkan suatu pinjaman

2.3.3 Perkalian Biner (*Binary Multiplication*)

Aturan untuk perkalian biner adalah sangat sederhana. Aturan – aturan ditunjukkan pada gambar 2.9. Perkalian biner hanyalah seperti perkalian desimal belaka. Gambar 2.10 mengilustrasikan suatu persoalan perkalaian biner yang sederhana.

0	0	1	1
x 0	x 1	x 0	x 1
0	0	0	1

Gambar 2.9 Aturan – Aturan untuk perkalian Biner

a	13	B	1101	Multiplikan
	x 10		x 1010	Pengali
	00		0000	Hasil Parsial ke 1
	13		1101	Hasil Parsial ke 2
	130		0000	Hasil Parsial ke 3
			1101	Hasil Parsial ke 4
			10000010	Hasil

Gambar 2.10 Persoalan perkalian biner sederhana

Persoalan perkalian desimal pada sebelah kiri gambar 2.10 menunjukkan metode tradisional pengerjaan operasi ini dengan tangan. Pada sebelah kanan gamabr 2.10, metode yang sama digunakan untuk bilangan biner. Biner 1101 (multiplikan, multiplicand) dikalikan dengan 1010 (pengali, multiplier). Prosedur tersebut ialah, mula –mula mengalikan bit 1 dari pengali (0) dengan multiplikan. Hasilnya berupa hasil parsial ke 1 yaitu 0000. kedua bit 2 (1) pada gambar 2.10 dikalikan dengan pengali. Hasilnya berupa hasil parsial ke 2, yaitu 1101. perlu diperhatikan bahwa hasil parsial ke 2 tersebut digeser satu posisi ke sebelah kiri dari hasil parsial ke 1. ketiga, bit 4 (0) dikalikan dengan pengali. Hasilnya adalah 0000, yang muncul sebagai hasil parsial ke 3. Keempat, bit 8 pada pengali, dikalikan dengan pengali, yang sama yaitu 1101. hasil ini dicatat pada posisi hasil parsial ke 4. Akhirnya, empat hasil parsial tersebut ditambahkan sehingga menghasilkan hasil akhir 10000010 desimal (130). Jika diubah kedalam bentuk heksadesimal maka memiliki hasil 82, yaitu:

$$82 = (8 \times 16^1) + (2 \times 16^0)$$

$$= 128 + 2$$

$$= 130$$

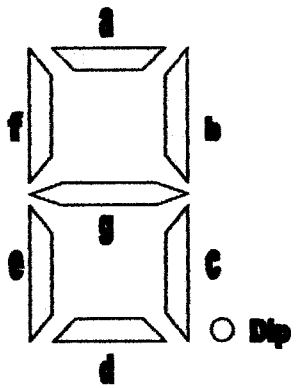
2.4 Tombol Masukan

Sebagai media untuk memasukkan data pada unit aritmatik ini digunakan tombol-tombol input yang keseluruhan berjumlah 16 tombol. 16 tombol tersebut terdiri atas 10 tombol untuk angka yaitu "0" s/d "9", 4 tombol untuk jenis operasi yaitu "+", "-", "x", 1 tombol "=" dan 1 tombol "clear". Semua tombol berupa

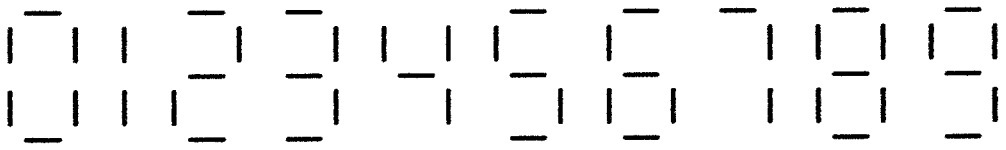
saklar push on tetapi bentuknya berbeda. Tombol bilangan berbentuk bulat sedangkan tombol operator, tombol "=" dan tombol "clear" berbentuk kotak. Tombol push on adalah tombol yang awalnya off, jika ditekan akan on, dan akan off kembali jika dilepas.

2.5 Penampil Tujuh Segmen

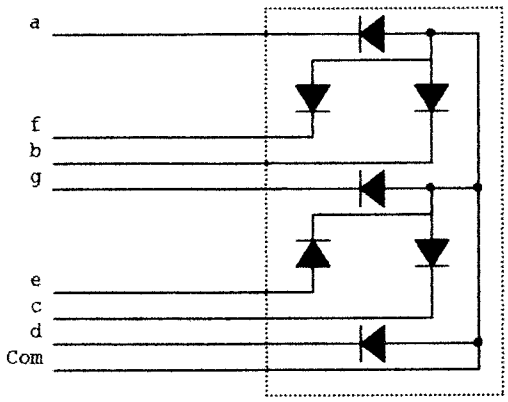
Penampil tujuh segmen ada 2 jenis, yaitu anoda bersama dan katoda bersama. Sebenarnya kedua jenis penampil tujuh segmen tersebut memiliki karakteristik yang hamper sama hanya berbeda penggabungan kaki-kaki led. Pada skripsi ini digunakan penampil tujuh segmen anoda bersama. Penampil tujuh segmen anoda bersama terdiri dari tujuh segmen terpisah yang diberi label a sampai g seperti yang disajikan pada gambar 2.11. tujuh segmen merupakan cacah segmen minimum yang diperlukan untuk menampilkan angka 0 sampai 9 seperti diilustrasikan pada gambar 2.12. rangkaian penampil tujuh segmen anoda bersama dapat dilihat pada gambar 2.13.



Gambar 2.11 Penampil tujuh segmen



Gambar 2.12 Tujuh segmen dalam digit desimal



Gambar 2.13 Rangkaian penampil tujuh segmen anoda bersama

2.6 *Field Programmable Gate Array* (FPGA)

Field-Programable Gate Array (FPGA) adalah salah satu piranti yang termasuk dalam kelompok *programable logic Devices.* Teknologi IC FPGA diperkenalkan pada tahun 1985 oleh perusahaan semikonduktor Xilinx. FPGA adalah sebuah konsep teknologi IC yang dapat diprogram dan dihapus seperti halnya 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 prototype beberapa system digital. Salah satu produsen yang ada dipasaran adalah Xilinx. Disamping produsen lainnya Actel dan Altera. Prinsip dasar dari

pemrograman/pengkonfigurasian FPGA Xilinx adalah pengubahan gambar rangkaian elektronika digital dari perangkat lunak Xilinx menjadi file aliran bit (bitstream) dan dikonfigurasi (didownload) 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 kemampuan yang berbeda. Perbedaan tersebut meliputi kecepatan, kapasitas gerbang logika.

2.6.1 FPGA keluarga Xilinx Spartan II

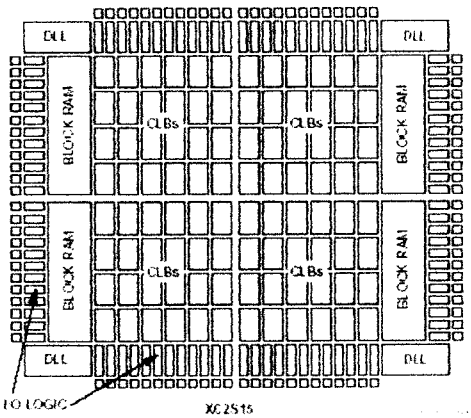
Spartan II merupakan salah satu keluarga FPGA yang dikeluarkan oleh Xilinx. Xilinx merupakan salah satu pabrik pembuat FPGA yang cukup terkenal. Keluarga Spartan II merupakan keluarga FPGA yang memiliki 15.000 sampai 200.000 gerbang. IC Xilinx ini dapat diprogram dan dihapus dengan waktu yang tidak terbatas serta murah harganya. Kelurga Spartan ini dapat diprogram dengan mudah menggunakan Xilinx *Development System* ataupun dengan *development system* lain yang dikembangkan oleh para pengguna.

Tabel 2.2 Data Keluarga Spartan II

Device	Logic Cells	System Gates (Logic and RAM)	CLB Array (R x C)	Total CLBs	Maximum Available User I/O[1]	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

2.6.2 Struktur Dasar Keluarga Spartan II

Suatu piranti FPGA terdiri atas *Configurable Logic Blocks* (CLB), unit input/output, empat buah *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 2.4.



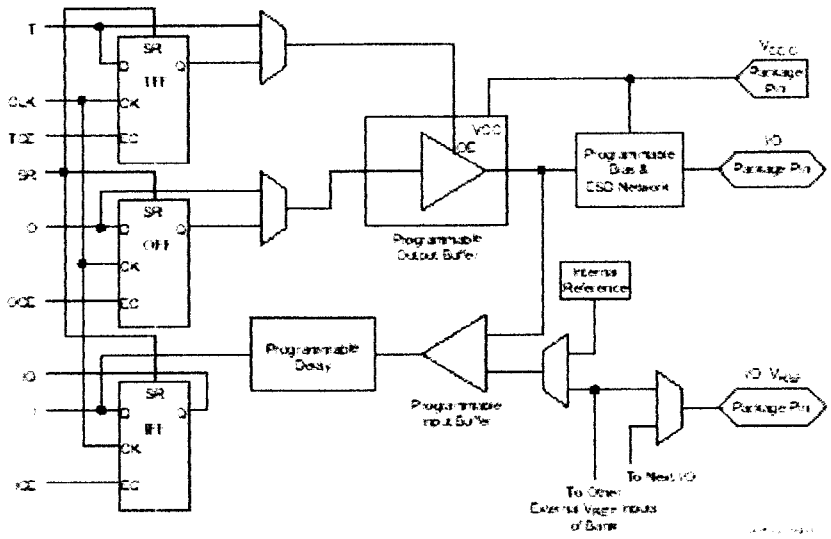
Gambar 2.14 Diagram Blok Dasar Keluarga Spartan II

2.6.3 *Input/Output Blocks* (IOB)

Input/Output blocks merupakan bagian dari FPGA yang berfungsi menghubungkan FPGA dengan piranti lain yang akan 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.

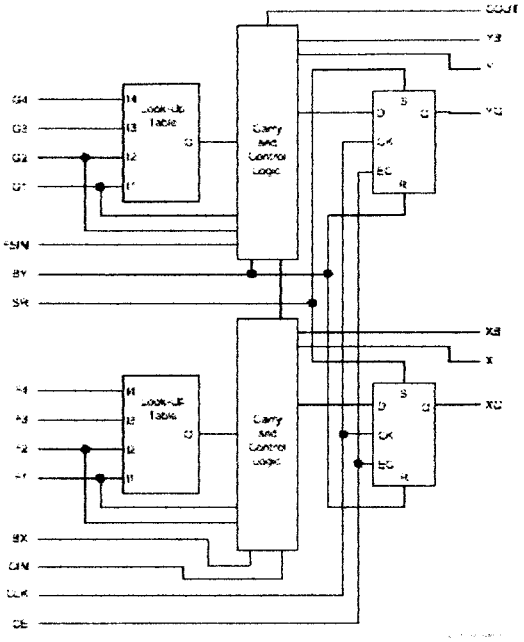


Gambar 2.15 Blok IOB Spartan II

2.6.4 *Configurable Logic Blocks* (CLB)

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 terditi 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 input pada D flip-flop. Setiap CLB pada Spartan II terdiri atas empat LC yang tersusun

dalam dua *slices* yang identik. Tiap CLB ini juga memuat logika yang akan mengkombinasi generator pembangkit fungsi logika untuk 4 sampai 6 input.



Gambar 2.16 CLB pada Spartan II

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.

2.6.5 *Programmable Routing Matrix*

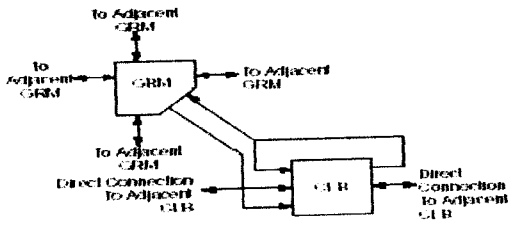
Programmable Routing Matrix merupakan cara sebuah FPGA melakukan *routing* menghubungkan CLB-CLB dan IOB-IOB yang digunakan dalam desain menjadi satu kesatuan sistem. *Routing* ini dilakukan secara otomatis penuh.

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:

1. *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 2.17 Struktur *Local Routing*

2. *General Purpose Routing*

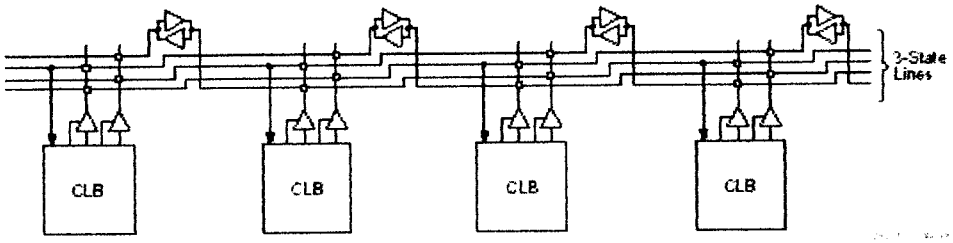
General Purpose Routing digunakan untuk melakukan koneksi vertikal dan horizonal antar kolom dan baris CLB yang digunakan.

3. *I/O Routing*

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

4. *Dedicated Routing*

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



Gambar 2.18 Koneksi BUFT untuk *dedicated Horizontal Bus Line*

5. *Global Routing*

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

2.6.6 Pemrograman FPGA

Keluarga Spartan II yang dikeluarkan oleh Xilinx dapat dengan mudah diprogram menggunakan *development system software* keluaran Xilinx yaitu Xilinx Foundation Series. Saat ini versi yang terbaru adalah versi 6.i3. *Software* memiliki kemampuan *place and route tools* (PAR) yang memungkinkan penggunanya menyusun IOB, LUT dan CLB menjadi sebuah kesatuan sistem.

FPGA dapat diprogram menggunakan metode *schematic* ataupun menggunakan *Hardware Description Language*, baik Verilog maupun VHDL. Masing-masing metode memiliki kelebihan dan kelemahannya sendiri.

Keluarga Spartan II dapat dioperasikan dalam 4 mode yaitu:

1. *Slave Serial mode*

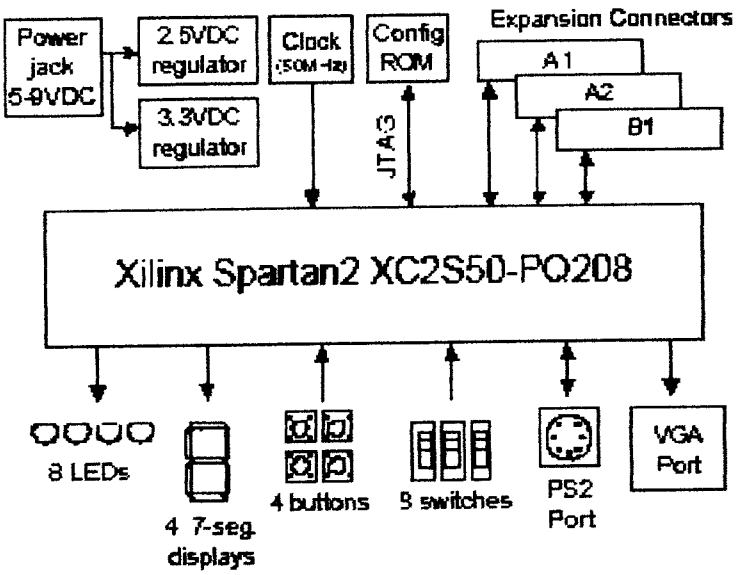
2. *Master Serial mode*
3. *Slave Parallel mode*
4. *Boundary Scan mode*

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.

2.7 Modul Pegasus

Modul pegasus dikeluarkan oleh pengembang ke 3, yaitu DIGILENT. Dengan IC FPGA utama yaitu, Xilinx Spartan2 XC2S50, dengan perangkat lunak Xilinx. Perangkat pendukung yang terdapat pada modul Pegasus diantaranya:

a. 50k-gate Xilinx Spartan2 FPGA dengan 50k gerbang dasar dan 200MHz operasi *maximum*.
b. XCF01S Xilinx Flash ROM.
c. Berbagai macam I/O, termasuk didalamnya delapan 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 2.19 Blok diagram Pegasus

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

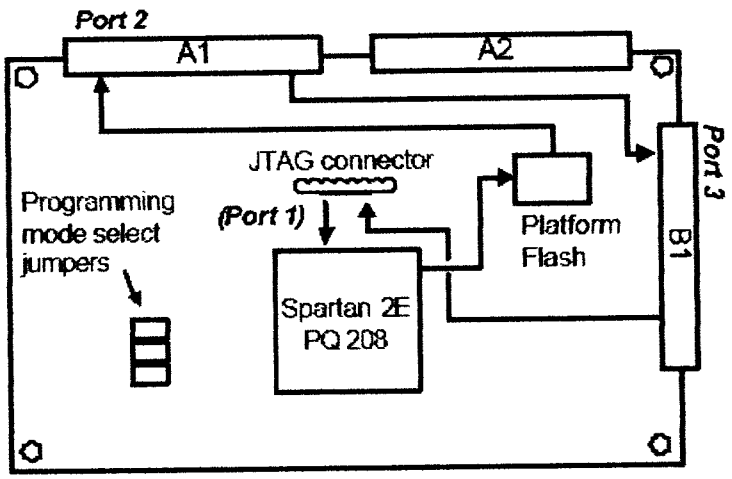
2.7.1 Port JTAG dan Mengkonfigurasi Modul

Pada modul Pegasus ini selain terdapat IC FPGA utama (Spartan2 XC2S50) juga terdapat IC FPGA *secondary* yaitu Spartan2 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*.

Scaning 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 *port*2 (A1) atau *port*3 (B1) maka *buffer* pada modul pegasus menghilangkan keberadaan *port* tambahan tersebut, sedangkan jika ada modul (memiliki 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 pada *port* JTAG tambahan, setelah terbaca IC FPGA *secondary* yang terhubung melalui *port* tambahan tersebut, maka dapat dikonfigurasi secara bersamaan melalui 1 kabel utama, baik itu *port* USB, Paralel maupun Ethernet.



Gambar 2.20 Aliran scan JTAG pada Pegasus

2.7.2 *Power Supply*

Modul Pegasus memerlukan 5Vdc *power supply*, sedangkan untuk masukan/keluaran dapat menggunakan *power* dari luar 3Vdc - 5Vdc. *power supply* 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.

2.7.3 *Oscilator*

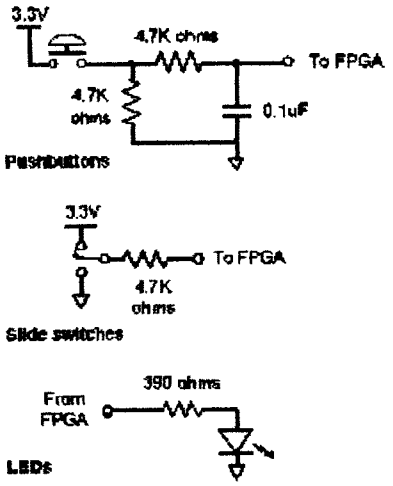
Modul Pegasus menyediakan 50MHz *Oscilator* sebagai clock utama untuk aplikasi-aplikasi yang akan dikonfigurasikan ke Spartan2 XC2S50. *Oscilator* tersebut terhubung langsung ke Spartan2 XC2S50 (*pin* 77).

2.7.4 Saklar *Pushbuttons*, Saklar Geser, Indikator Led

Empat saklar *pushbutton*, dan delapan 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 series.

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

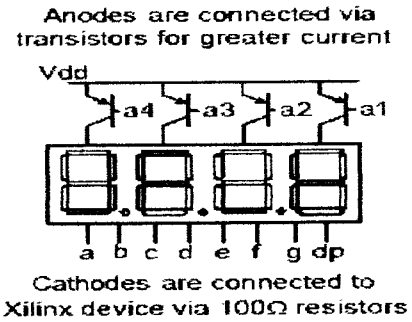
LED ke 9 digunakan sebagai *indikator power* FPGA, dan LED ke 10 digunakan sebagai *indikator* status pemrograman JTAG.



Gambar 2.21 Rangkaian saklar *pushbutton*, saklar geser, LED

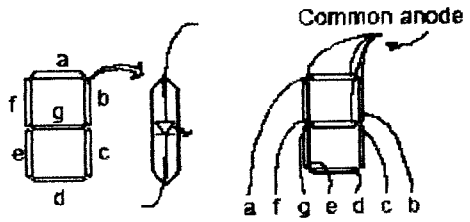
2.7.5 Seven-Segment

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*. 4 *digit* selektor *enable* berfungsi sebagai pengatur *digit* pada *seven-segment*.



Gambar 2.22 *Common anode seven-segment* 4 *digit*

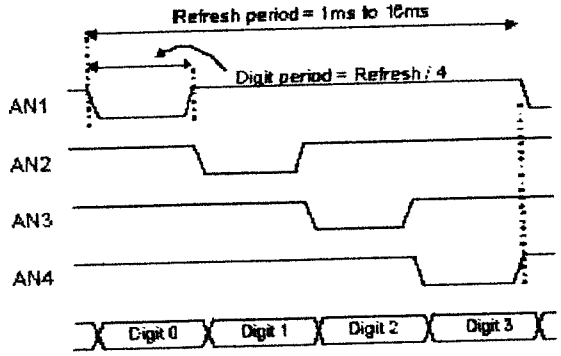
Ketujuh anoda dari 4 *seven-segment* saling tersambung kedalam selektor "*common anode*", *display* ini memiliki 4 selektor yang dinamakan AN0 - AN3, apabila ada sinyal/pulsa yang mengaktifkan selektor ini maka *digit* dari selektor tersebut akan *aktif*. Sinyal/pulsa yang digunakan adalah sinyal/pulsa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



Gambar 2.23 *Common anode seven-segment* 1 *digit*

Dilihat dari diagram yang telihat maka menghasilkan *display seven segment multiplexer*, yang mana bila anoda atau selektor (AN0 - AN3) di aktifkan maka *digit* tersebut yang akan *aktif* dan mendapatkan sinyal/pulsa katoda (CA - CG). Jika dilakukan secara terus-menerus dan bila benar maka keempat *digit* akan terlihat *aktif*/seolah-olah semua *aktif* secara bersamaan. Jika diberi sinyal rendah (0Vdc) secara terus menerus 1ms - 16 ms (selector), maka akan terlihat semua digit *aktif*. Dengan syarat, bersamaan dengan *digit enable* data untuk *digit* tersebut harus ada. *Refresh frekuensi* berkisar 60Hz sampai 1KHz. Sebagai contoh dengan 16ms (62.5Hz) sebagai *refresh time* setiap *digit* akan mendapat waktu *aktif* selama ¼ dari *refresh time* (dalam hal ini 4ms), selektor harus sesuai dengan data yang akan dimasukkan, dengan kata lain saat *digit* 1 *aktif* hanya boleh data

untuk *digit* 1 yang ada pada CA - CG. Begitu pula dengan *digit* 2, *digit* 3 dan *digit* 4. hal ini dapat dilihat pada gambar 2.24.



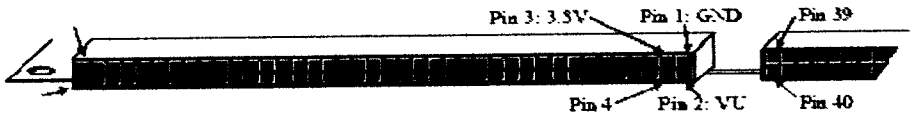
Gambar 2.24 Diagram waktu sinyal *seven-segment*

Tabel 2.3 Masukan katoda untuk *digit* desimal

Digit Shown	Cathode Signals						
	a	b	c	d	e	f	g
0	0	0	0	0	0	0	1
1	1	0	0	1	1	1	1
2	0	0	1	0	0	1	0
3	0	0	0	0	1	1	0
4	1	0	0	1	1	0	0
5	0	1	0	0	1	0	0
6	0	1	0	0	0	0	0
7	0	0	0	1	1	1	1
8	0	0	0	0	0	0	0
9	0	0	0	1	1	0	0

2.7.6 *Port* I/O Tambahan

Pada modul Pegasus terdapat 3 *port* tambahan (A1, A2, dan B1) 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 2.25 *Pin* penghubung tambahan

Tabel 2.4 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	GCK0	GCK0
40	TDI	TDI	40	TDI	TDI	40	GND	GND
37	TMS	TMS	37	TMS	TMS	37	n/c	n/c
38	TCK	TCK	38	TCK	TCK	38	n/c	n/c
35	MB1-INIT	90	35	MA1-INIT	189	35	MA2-INT	*138
36	GND	GND	36	GND	GND	36	Not used	n/c
33	MB1-WAIT	95	33	MA1-WAIT	192	33	MA2-WAIT	*140
34	M1-RST	94	34	M1-RST	191	34	MA2-RST	*139
31	MB1-DSTB	97	31	MA1-DSTB	194	31	MA2-DSTB	*142
32	MB1-WRIT	96	32	MA1-WRIT	193	32	MA2-WRIT	*141
29	MB1-DB7	99	29	MA1-DB7	199	29	MA2-DB7	*147
30	MB1-ASTB	98	30	MA1-ASTB	195	30	MA2-ASTB	*146
27	MB1-DB5	101	27	MA1-DB5	201	27	MA2-DB5	*149
28	MB1-DB6	100	28	MA1-DB6	200	28	MA2-DB6	*148
25	MB1-DB3	103	25	MA1-DB3	203	25	MA2-DB3	*151
26	MB1-DB4	102	26	MA1-DB4	202	26	MA2-DB4	*150
23	MB1-DB1	110	23	MA1-DB1	205	23	MA2-DB1	*160
24	MB1-DB2	109	24	MA1-DB2	204	24	MA2-DB2	*152
21	P-LSBCLK	112	21	LSBCLK	3	21	P-IO18	*162
22	MB1-DB0	111	22	MA1-DB0	206	22	MA2-DB0	*161
19	P1-DB7	114	19	DB7	5	19	P-IO16	*164
20	P-CSA	113	20	CSA	4	20	P-IO17	*163
17	P-DB6	119	17	DB6	7	17	P-IO14	*166
18	P-OE	115	18	OE	6	18	P-IO15	*165
15	P-DB5	121	15	DB5	9	15	P-IO12	*168
16	P-WE	120	16	WE	8	16	P-IO13	*167
13	P-DB4	123	13	DB4	14	13	P-IO10	*173
14	P-ADR5	122	14	ADR5	10	14	P-IO11	*172
11	P-DB3	126	11	DB3	16	11	P-IO8	*175
12	P-ADR4	125	12	ADR4	15	12	P-IO9	*174
9	P-DB2	129	9	DB2	18	9	P-IO6	*178
10	P-ADR3	127	10	ADR3	17	10	P-IO7	*176
7	P-DB1	133	7	DB1	21	7	P-IO4	*180
8	P-ADR2	132	8	ADR2	20	8	P-IO5	*179
5	P-DB0	135	5	DB0	23	5	P-IO2	*187
6	P-ADR1	134	6	ADR1	22	6	P-IO3	*181
3	VCCO	VCCO	3	VCCO	VCCO	3	VCCO	VCCO
4	P-ADR0	136	4	ADR0	24	4	P-IO1	*188
1	GND	GND	1	GND	GND	1	GND	GND
2	VU	VU	2	VU	VU	2	VU	VU

Tabel 2.5 *Port Accessory*

Accessory Port Pinout					
Pin	Name	FPGA Pin	Pin	Name	FPGA Pin
1	AC0	P49	4	AC3	P47
2	AC1	P48	5	GND	-
3	AC2	P81	6	Vdd	-

Tabel 2.6 Pin FPGA XC2S50

Pegasus FPGA Pin Assignments							
Pin	Function	Pin	Function	Pin	Function	Pin	Function
1	GND	53	VCCO	105	VCCO	157	TDO
2	TMS	54	MODE2	106	PROGRAM	158	GND
3	LLSBCLK	55	PB-IO14	107	INIT/IO	159	TDI
4	LCSA	56	PB-IO13	108	LMB1-DB3	160	LMA2-DB1
5	LDB7	57	BTN2	109	LMB1-DB2	161	LMA2-DB0
6	LOE	58	BTN1	110	LMB1-DB1	162	LPA-IO16
7	LDB6	59	BTN0	111	LMB1-DB0	163	LPA-IO17
8	LWE	60	AN0	112	LPB-LSBCLK	164	LPA-IO15
9	LDB5	61	CE	113	LPB-CSA	165	LPA-IO15
10	LADR5	62	CD	114	LPB-DB7	166	LPA-IO14
11	GND	63	DP	115	LPB-OE	167	LPA-IO13
12	VCCO	64	GND	116	GND	168	LPA-IO12
13	VCCINIT	65	VCCO	117	VCCO	169	GND
14	LDB4	66	VCCINIT	118	VCCINIT	170	VCCO
15	LADR4	67	CC	119	LPB-DB6	171	VCCINIT
16	LDB3	68	CG	120	LPB-WE	172	LPA-IO11
17	LADR3	69	AN1	121	LPB-DB5	173	LPA-IO10
18	LDB2	70	CB	122	LPB-ADR5	174	LPA-IO9
19	GND	71	AN2	123	LPB-DB4	175	LPA-IO8
20	LADR2	72	GND	124	GND	176	LPA-IO7
21	LDB1	73	CF	125	LPB-ADR4	177	GND
22	LADR1	74	CA	126	LPB-DB3	178	LPA-IO6
23	LDB0	75	AN3	127	LPB-ADR3	179	LPA-IO5
24	LADR0	76	VCCINIT	128	VCCINIT	180	LPA-IO4
25	GND	77	GCK1	129	LPB-DB2	181	LPA-IO3
26	VCCO	78	VCCO	130	VCCO	182	GCK2
27	VS	79	GND	131	GND	183	GND
28	VCCINT	80	GCK0	132	LPB-ADR2	184	VCCO
29	HS	81	SW7/AC2	133	LPB-DB1	185	GCK3
30	BLUE	82	SW6	134	LPB-ADR1	186	VCCINIT
31	GRN	83	SW5	135	LPB-DB0	187	LPA-IO2
32	GND	84	SW4	136	LPB-ADR0	188	LPA-IO1
33	RED	85	GND	137	GND	189	LMA1-INT
34	PS2C	86	SW3	138	LMA2-INT	190	GND
35	PS2D	87	SW2	139	LMA2-RESET	191	LMA1-RESET
36	LD7	88	SW1	140	LMA2-WAIT	192	LMA1-WAIT
37	LD6	89	SW0	141	LMA2-WRITE	193	LMA1-WRITE
38	VCCINIT	90	LMB1-INT	142	LMA2-DSTB	194	LMA1-DSTB
39	VCCO	91	VCCINIT	143	VCCINIT	195	LMA1-ASTB
40	MC1-DB4	92	GND	144	VCCO	196	VCCINIT
41	LD5	93	GND	145	GND	197	VCCO
42	LD4	94	LMB1-RESET	146	LMA2-ASTB	198	GND
43	LD3	95	LMB1-WAIT	147	LMA2-DB7	199	LMA1-DB7
44	LD2	96	LMB1-WRITE	148	LMA2-DB6	200	LMA1-DB6
45	LD1	97	LMB1-DSTB	149	LMA2-DB5	201	LMA1-DB5
46	LD0	98	LMB1-ASTB	150	LMA2-DB4	202	LMA1-DB4
47	AC3	99	LMB1-DB7	151	LMA2-DB3	203	LMA1-DB3
48	AC1	100	LMB1-DB6	152	LMA2-DB2	204	LMA1-DB2
49	ACD	101	LMB1-DB5	153	DIN/D0/IO	205	LMA1-DB1
50	MODE1	102	LMB1-DB4	154	BTN3	206	LMA1-DB0
51	GND	103	GND	155	CCLK	207	TCK
52	MODE0	104	DONE	156	VCCO	208	VCCO

Tabel 2.7 Konfigurasi Pin yang Terdapat pada *Board* Pegasus.

Pegasus Expansion Connector Pinout								
Conrector B1			Connector A1			Connector A2		
Pin	Signal	B1	Pin	Signal	A1	Pin	Signal	A2
39	TDO	TDO	39	TDO	TDO	39	GCK0	GCK0
40	TDI	TDI	40	TDI	TDI	40	GND	GND
37	TMS	TMS	37	TMS	TMS	37	n/c	n/c
38	TCK	TCK	38	TCK	TCK	38	n/c	n/c
35	MB1-INIT	90	35	MA1-INIT	188	35	MA2-INIT	139
36	GND	GND	36	GND	GND	36	Not used	n/c
33	MB1-WAIT	95	33	MA1-WAIT	192	33	MA2-WAIT	140
34	M1-RST	94	34	M1-RST	191	34	MA2-RST	138
31	MB1-DSTB	97	31	MA1-DSTB	194	31	MA2-DSTB	142
32	MB1-WRIT	96	32	MA1-WRIT	193	32	MA2-WRIT	141
29	MB1-DB7	99	29	MA1-DB7	198	29	MA2-DB7	147
30	MB1-ASTB	98	30	MA1-ASTB	195	30	MA2-ASTB	146
27	MB1-DB5	101	27	MA1-DB5	201	27	MA2-DB5	149
28	MB1-DB6	100	28	MA1-DB6	200	28	MA2-DB6	148
25	MB1-DB3	103	25	MA1-DB3	203	25	MA2-DB3	151
26	MB1-DB4	102	26	MA1-DB4	202	26	MA2-DB4	150
23	MB1-DB1	110	23	MA1-DB1	205	23	MA2-DB1	160
24	MB1-DB2	109	24	MA1-DB2	204	24	MA2-DB2	152
21	P-LSBCLK	112	21	LSBCLK	3	21	P-IO13	162
22	MB1-DB0	111	22	MA1-DB0	206	22	MA2-DB0	161
19	P1-DB7	114	19	DB7	5	19	P-IO18	164
20	P-CSA	113	20	CSA	4	20	P-IO17	163
17	P-DB6	116	17	DB6	7	17	P-IO14	166
18	P-OE	115	18	OE	6	18	P-IO15	165
15	P-DB5	121	15	DB5	8	15	P-IO12	169
16	P-WE	120	16	WE	9	16	P-IO16	167
13	P-DB4	123	13	DB4	14	13	P-IO10	173
14	P-ADR5	122	14	ADR5	10	14	P-IO11	172
11	P-DB3	126	11	DB3	16	11	P-IO9	175
12	P-ADR4	125	12	ADR4	15	12	P-IO8	174
9	P-DB2	128	9	DB2	18	9	P-IO6	178
10	P-ADR3	127	10	ADR3	17	10	P-IO7	176
7	P-DB1	133	7	DB1	21	7	P-IO4	180
8	P-ADR2	132	8	ADR2	20	8	P-IO5	179
5	P-DB0	135	5	DB0	23	5	P-IO2	187
6	P-ADR1	134	6	ADR1	22	6	P-IO3	181
3	VCCO	VCCO	3	VCCO	VCCO	3	VCCO	VCCO
4	P-ADR0	136	4	ADR0	24	4	P-IO1	189
1	GND	GND	1	GND	GND	1	GND	GND
2	VU	VU	2	VU	VU	2	VU	VU

2.7 Sistem Digital

Suatu isyarat digital merupakan suatu gelombang yang mempunyai peralihan mendadak antara dua nilai besaran atau dengan kata lain hanya mengenal dua macam keadaan. Dua keadaan ini tidak dikenal besarannya, tetapi dikenal dengan keadaan tinggi rendah. Dengan bilangan biasa, dua macam keadaan ini dapat diberi tanda 0 (nol) jika rendah, dan tanda 1 (satu) jika tinggi. Dalam isyarat-isyarat listrik tanda nol dapat diartikan tidak ada tegangan, dan isyarat satu dapat diartikan ada tegangan. Dalam hal ini dimungkinkan satu tingkat isyarat dapat bernilai 0 dan tingkat isyarat lain bernilai 1. keadaan 0 dan 1 inilah yang dinamakan nalar (LOGIC).

Tabel berikut ini menunjukkan dua macam keadaan yang dimaksud dalam sistem digital:

Tabel 2.8 Tabel keadaan sistem digital

NALAR	Keadaan suatu sistem				
1	ON	Tinggi	Ya	Naik	+5 Volt
0	OFF	Rendah	Tidak	Turun	0 Volt

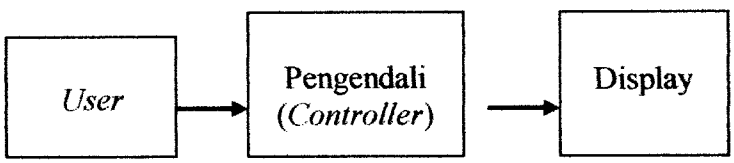
BAB III

PERANCANGAN SITEM

3.1 Perancangan Unit Penjumlah dan Pengurang 8 bit

FPGA (*Field Programable Gate Arrays*) adalah salah satu piranti yang termasuk dalam kelompok *programable logic Devices* (PLD). FPGA berbeda dari *general-purpose* mikroprosesor (misalnya intel) dalam hal fleksibilitas *logic*-nya. Mikroprosesor mempunyai *hardware* yang tetap. *Assembly programmer* memprogram suatu komputasi dengan keterbatasan pada tetapnya banyaknya register dan fungsi ALU (*aritmetich and logic unit*) dan pada banyaknya bit suatu register. Pada FPGA tidak terbatas akan pemakaian register-register dan dapat mengimplementasikan rangkaian kombinasi maupun rangkaian skuensial. Jadi intinya FPGA digunakan untuk direkonfigurasi, bukan diprogram.

Perancangan umum Penjumlah dan Pengurang 8 bit dapat dilihat pada blok diagram perancangan di bawah ini.



Gambar 3.1 Blok Diagram Penjumlah dan Pengurang 8 bit

Pada blok diagram di atas terbagi atas bagian-bagian yang masing-masing mempunyai fungsi berlainan tetapi saling berkaitan. Adapun fungsi dari bagian-bagian tersebut adalah:

3.1.1 *User*/pengguna

User/pengguna pada bagian ini dipakai sebagai input. Sebab pengguna akan merekonfigurasi FPGA menjadi fungsi – fungsi tertentu yang akan ditampilkan ke penampil.

3.1.2 Pengendali *(Controller)*

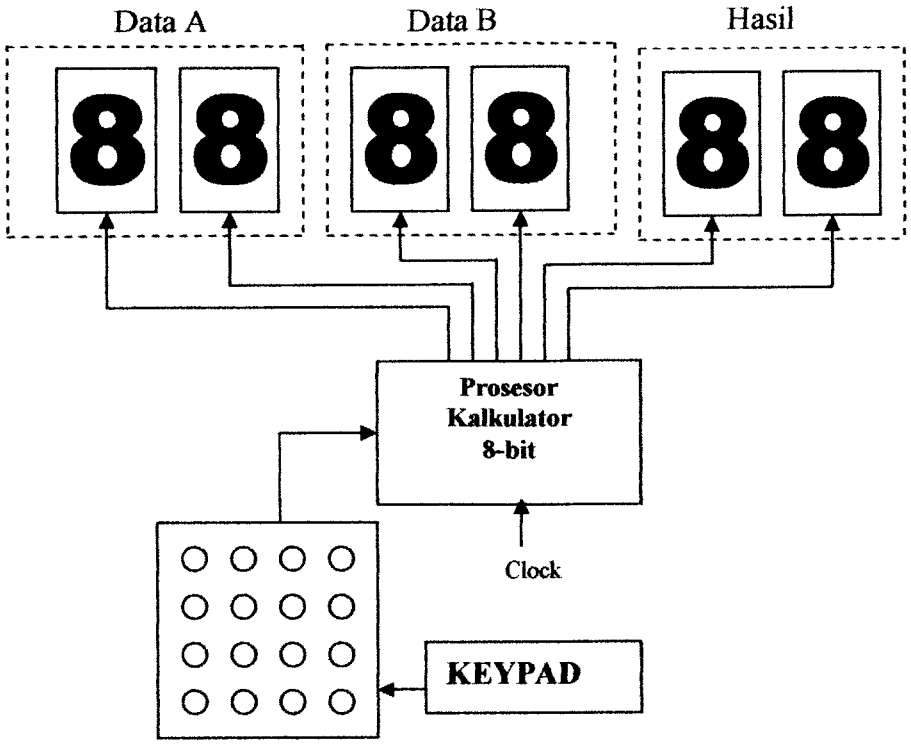
Pengendali yang kita gunakan adalah FPGA. Blok ini berfungsi sebagai *interface* (antarmuka) antara pengguna dengan penampil dan ALU (*aritmetich and logic unit*), karena dalam blok semua proses komputasi diproses dan di eksekusi. Ketika ada data yang dimasukkan oleh *user* maka FPGA akan memproses dan mengeksekusi data tersebut, kemudian data hasil eksekusi di*transfer* oleh FPGA ke *Penampil 7-Segmen.*

3.1.3 *Display*

Pada bagian ini digunakan penampil 7-Segmen sebagai output dari semua nilai masukan yang dikehendaki pengguna.

3.2 Perancangan *Software*

Perancangan khusus Kalkulatordapat dilihat pada Blok diagram dari sistem kalkulator ditunjukkan oleh gambar 3.2. Kalkulator ini menerima masukan dua bilangan masing-masing selebar 8-bit.



Gambar 3.2 Blok diagram dari sistem

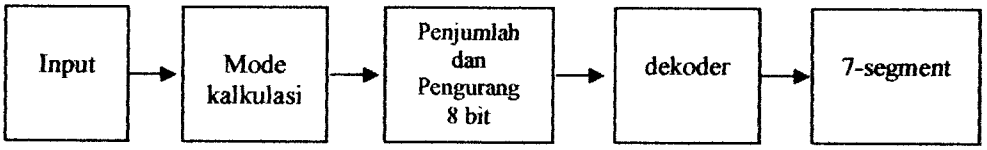
Untuk merealisasikan gambar 3.2, maka disain besar tersebut dapat dibuat empat buah blok disain yang lebih kecil yaitu:

1. Blok sinkronisasi input
2. Blok pemilih *mode* kalkulasi
3. Blok Penjumlah dan Pengurang 8 bit
4. Blok sinkronisasi dekoder
5. Blok tampilan ke *seven segment*

Empat blok diatas didisain menggunakan VHDL. Kemudian buat (*create*) menjadi skematik. Hubungkan keempat blok secara skematik (*top-level* disain *project* adalah skematik). Kemudian mendefinisikan koneksi pin-pin IC nya menggunakan

file *.ucf. Maka *project* siap disintesis, implementasi, dan men*download bit-stream* ke dalam divais.

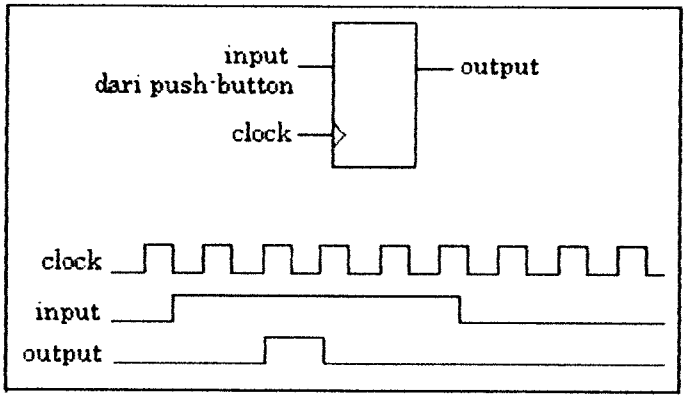
Berikut Blok diagram Penjumlah dan Pengurang 8 bit pada gambar 3.3 :



Gambar 3.3 Blok diagram Penjumlah dan Pengurang 8 bit

3.2.1 Blok Sinkronisasi *Input*

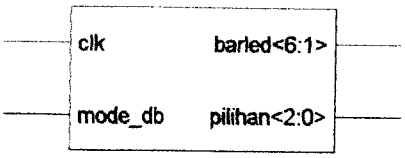
Semua blok didisain secara sinkron menggunakan sebuah sinyal *clock* yang sama, penekanan tombol push-button akan disinkronkan terhadap clock. Clock yang digunakan (*default*) mempunyai frekuensi 50MHz atau 20n detik sedangkan penekanan tombol *keypad* dapat menghasilkan pulsa yang lebih lebar sekitar beberapa milidetik. Ilustrasinya diperlihatkan oleh gambar 3.4. Blok tersebut bertugas menghasilkan satu pulsa selebar satu periode *clock*.



Gambar 3.4 Mensinkronkan input yang lebih lebar terhadap *clock*

3.2.2 Blok Pemilih mode kalkulasi

Bagian ini bertugas untuk menghasilkan hitungan dari 0 sampai 2. Hasil hitungan tersebut digunakan untuk menentukan mode pilihan kalkulasi .



Gambar 3.5 Mendapatkan mode pilihan dari 0 sampai 2

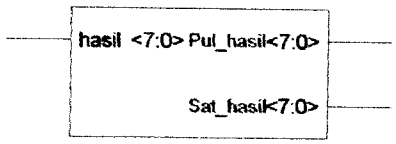
Mode kalkulasi dipilih dengan cara menekan salah satu tombol pada *keypad* yang telah ditentukan fungsinya. Bilangan A dan B masing-masing selebar 8-bit yang akan dimasukkan menggunakan *keypad*. Apabila tombol yang berfungsi sebagai tombol *enter* pada *keypad* ditekan, maka hasil kalkulasi akan ditampilkan pada dua buah seven segmen dalam format desimal.

3.2.3 Blok Penjumlah dan Pengurang 8 bit

Bagian ini adalah inti dari Penjumlah dan Pengurang 8 bit. Blok ini mendapat masukan dua buah bilangan A dan B masing-masing selebar 8-bit yang didapat dari *keypad*. Penekanan *enter* (sinyal ini didapat dari keluaran blok sinkronisasi *input*) akan menghasilkan keluaran hasil selebar 8-bit sesuai dengan sinyal pilihan yang didapat dari blok pemilih *mode* kalkulasi.

3.2.4 Blok Tampilan ke Seven Segment

Bagian ini akan menampilkan hasil keluaran dari blok kalkulasi selebar 8-bit pada dua buah *seven segment.*



Gambar 3.6 Hasil 8-bit diubah menjadi format desimal

3.2.5 Blok Sinkronisasi Dekoder

Bagian ini berfungsi mengubah bilangan heksadesimal menjadi desimal. Hasil keluaran akan menunjukkan angka -angka desimal. berikut daftar sinkronisasi dekoder yang mengubah keluaran menjadi desimal.

Tabel 3.1 Dekoder pengubah Heksadesimal menjadi desimal

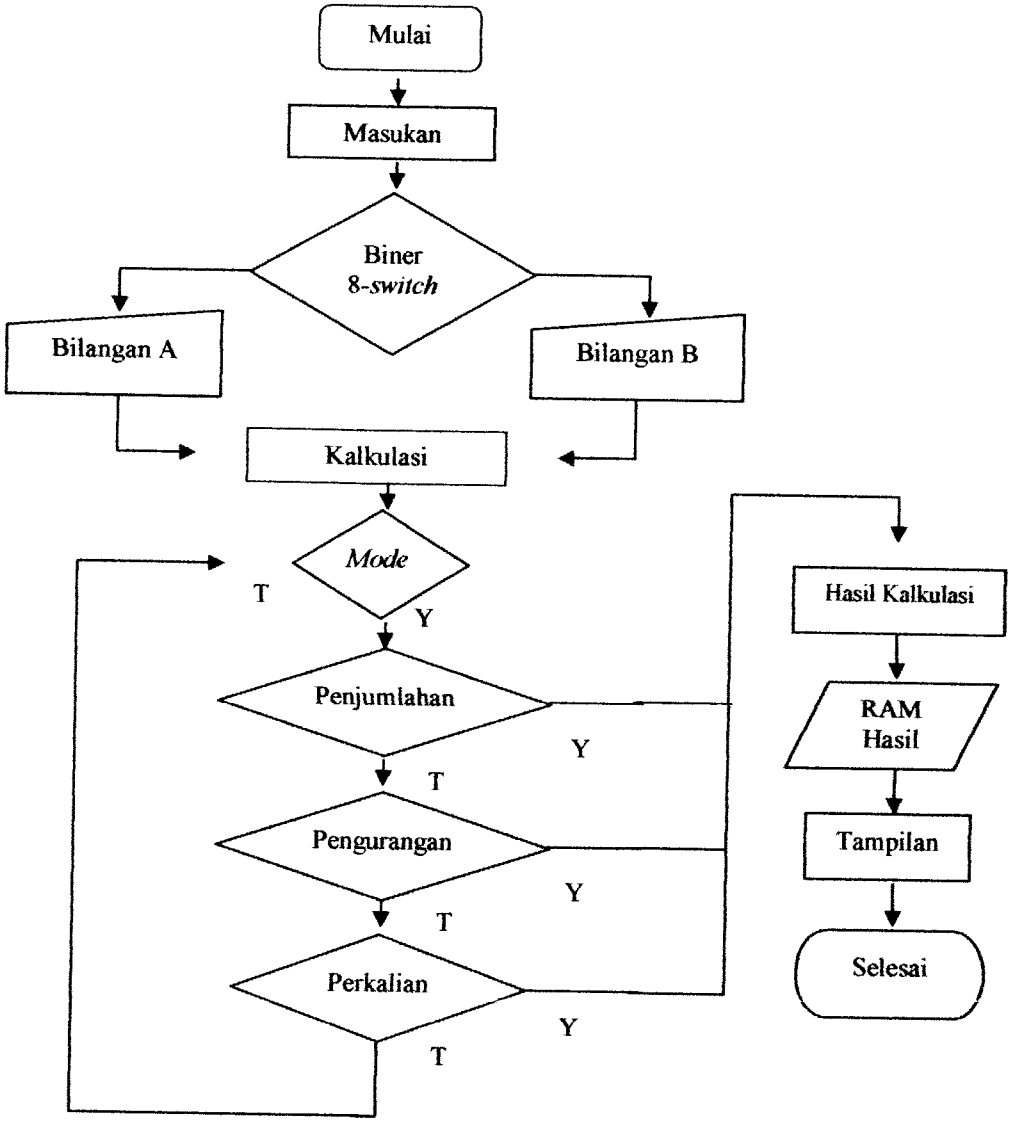
No.	Desimal	Heksadesimal
10	0001 0000	0000 1010
11	0001 0001	0000 1011
12	0001 0010	0000 1100
13	0001 0011	0000 1101
14	0001 0100	0000 1110
15	0001 0101	0000 1111
16	0001 0110	0001 0000
... dst		

3.3 Program Utama

Untuk menjalankan sistem, diperlukan perangkat lunak untuk mengendalikan perangkat keras. Perangkat lunak terdiri program utama yang didalamnya terdapat sub-sub program. Sus-sub program itu antaralain:

1. Logika isyarat masukan
2. Pengkode hexadesimal ke biner
3. Logika counter / pencacah
4. Pengontrol / kalkulasi
5. Mode operator
6. Logika keluaran

Untuk memudahkan dalam pembuatan program, diperlukan diagram alir sebagai kerangka dasar logika program. Diagram alir Penjumlah dan Pengurang 8 bit dapat dilihat pada Gambar 3.7 dibawah ini:



Gambar 3.7 *Flowchart* Penjumlah dan pengurang 8 bit

BAB IV

ANALISA DAN PEMBAHASAN

4.1 Analisis sistem

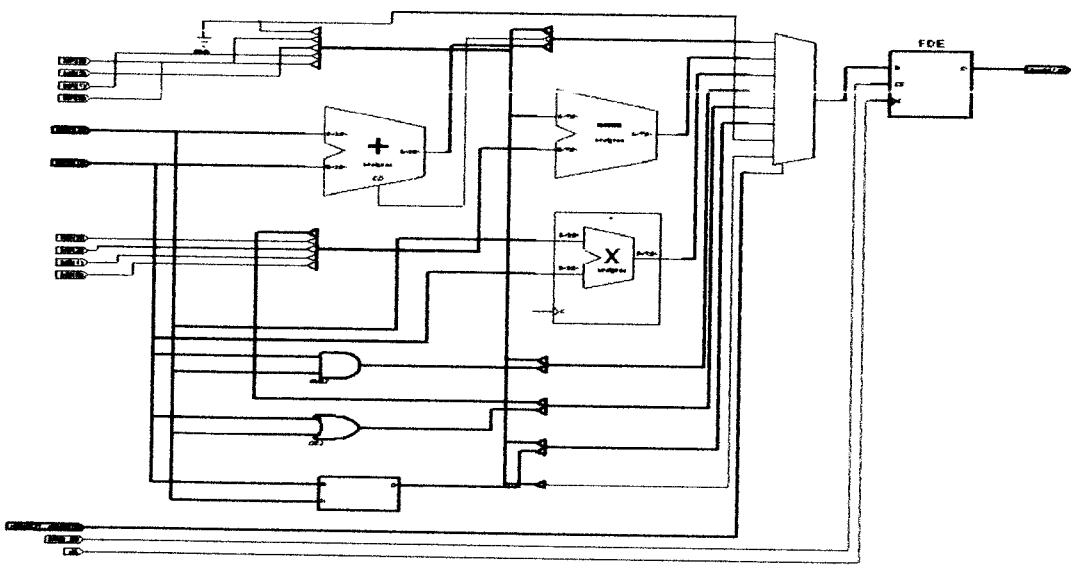
Sistem Kalkulasi dapat bekerja dengan baik jika telah di *synthesize* sebelum *listing* program akan di *generate* menjalankan kalkulasi data pada bilangan A dan bilangan B. Untuk mengoperasikannya alat ini dapat dilakukan dengan langkah-lngkah berikut ini :

1. Memberi masukan nilai pada bagian bilangan A dan bilangan B dengan cara menekan salah satu tombol pada *keypad* sesuai dengan nilai yang kita inginkan.
2. Memilih operasi aritmatika pada salah satu tombol pada *keypad*, jika memilih operasi penjumlah maka blok kalkulator akan mengeksekusi operasi penjumlahan dan seterusnya.
3. Hasil kalkulasi akan diproses pada blok kalkulator, kemudian menekan tombol *enter* atau hasil pada *keypad*.
4. Keluaran hasil berupa desimal diubah kedalam bentuk desimal melalui dekoder yang telah disusun. Lihat tabel 3.1 Dekoder pengubah Heksadesimal menjadi desimal
5. Tampilan 7-ruas menampilkan hasil kalkulasi. Berikut layout jika dilihat dari *software project navigator.*

4.2 Pembahasan

Pada perancangan kalkulator kali ini input tidak menggunakan bilangan negatif dan tidak lebih dari angka 99 serta output yang di hasilkan hanya sampai 99. Hasil maksimal dari kalkulasi pada perancangan ini adalah 99. Dalam perancangan ini, kalkulator tidak menggunakan operasi beruntun karena hasil dari kalkulasi langsung ditampilkan pada *seven-segment*. Sedangkan pada operasi beruntun hasil dari operasi sebelumnya dapat dikalkulasikan lagi sampai pada batas yang diijinkan.

Ketika input dimasukkan melalui kaypad kadang data yang ditampilkan tidak sesuai dengan data aslinya misalnya ketika kita memasukkan data 7, terkadang dat yang diterima tidak 7. hal ini disebabkan karena efek getaran dari tombol kaypad ketika kita memasukkan data sehingga data yang diterima tidak sesuai. Efek ini disebut dengan efek bouncing.



Gambar 4.1 Skematik Blok kalkulasi

BAB V

Penutup

5.1 Kesimpulan

Berdasarkan hasil perancangan serta pengujian sistem maka dapat disimpulkan beberapa hal yaitu :

1. Operasi yang digunakan dalam FPGA menggunakan operasi heksadesimal.
2. Operasi pembagian tidak dapat di sintesis karena pada bilangan heksadesimal tidak mengenal tanda koma.
3. 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.
4. Penggunaan display lebih dari 1 digit dapat dilakukan dengan cara membagi frekuensi pada *clock*nya.
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

Karena perancangan kalkulator ini dinilai masih banyak kekurangannya maka untuk pengembangan selanjutnya disarankan untuk memperhatikan beberapa hal berikut:

1. Kedepan hendaknya perancangan kalkulator dapat dilakukan dengan menambahkan *digit* masukan dan operasinya.
2. Jika pengendalian dilakukan dengan FPGA hendaknya diperhitungkan banyaknya gerbang logika dasar digital yang mampu dilayani. Untuk perancangan kalkulator ini cukup dengan menggunakan Xilink Spartan2 Xc2s50-PQ208, karena seri ini sudah memiliki jumlah gerbang yang lebih dari cukup.
3. Dengan kemampuan FPGA yang lebih besar antara *display* dengan sistem utama dapat menggunakan hanya dengan satu buah FPGA.

DAFTAR PUSTAKA

Aswir, I, 2004, *Implementasi Filter Wiener Pada Noise Canceling Berbasis FPGA*, Skripsi S-1, Teknik Elektro, Fakultas Teknologi Industri, Universitas Islam Indonesia, Jogjakarta.

Horn, D.T, *Teknik Merancang Rangkaian Dengan IC*, Jakarta, PT Elektronika Media Komputindo, 1989.

Lee, S.C, *Rangkaian Digital Dan Rangkaian Logika*, Jakarta, Penerbit Erlangga, 1994.

Malvino, A.P, *Elektronika Komputer Digital*, Edisi Kedua, Jakarta, Penerbit Erlangga, 1996.

Santosa, P.I, *Teknik Digital*, Jogjakrta, Penerbit Andi, 1996.

Supramono, *Statistika*, Jogjakarta, Penerbit Andi, 1996.

Stallings, W, *Komunikasi Data dan Komputer*, Jakarta, Salemba Teknika, 2001.

Tokheim, R.L, *Elektronika Digital*, Edisi Kedua, Jakarta, Penerbit Erlangga, 1995.

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:
-- Design Name:
-- Module Name:    Hasil_2 - 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;

entity Hasil_2 is
port(reset_all,enter,clk:in std_logic;
                key : in std_logic_vector(9 downto 0);
                operasi:in std_logic_vector(2 downto 0);
                an: inout std_logic_vector(3 downto 0);
                d : inout std_logic_vector(7 downto 0);
                am: inout std_logic_vector(1 downto 0);
                m : inout std_logic_vector(7 downto 0));
end Hasil_2;

architecture Behavioral of Hasil_2 is
signal triger:std_logic;
signal e,A_B,k:std_logic_vector(1 downto 0);
signal hitung:std_logic_vector(2 downto 0);
signal klok:std_logic_vector(17 downto 0);
signal segment,pulx,puly,satx,saty,sat_hasil,pul_hasil:std_logic_vector(7 downto
0);
signal angka,hasil,A,B,C,F,x,y:integer;
begin
process(clk,key)
begin
if clk='1' and clk'event then
```

```
        if reset_all='1' then
                angka<=0;segment<="11000000";
        else
                case key is
                        when "0000000001" =>
angka<=0;segment<="11000000";triger<='1';
                        when "0000000010" =>
angka<=1;segment<="11111001";triger<='1';
                  when "0000000100" =>
angka<=2;segment<="10100100";triger<='1';
                  when "0000001000" =>
angka<=3;segment<="10110000";triger<='1';
                        when "0000010000" =>
angka<=4;segment<="10011001";triger<='1';
                        when "0000100000" =>
angka<=5;segment<="10010010";triger<='1';
                        when "0001000000" =>
angka<=6;segment<="10000010";triger<='1';
                        when "0010000000" =>
angka<=7;segment<="11111000";triger<='1';
                        when "0100000000" =>
angka<=8;segment<="10000000";triger<='1';
                        when "1000000000" =>
angka<=9;segment<="10010000";triger<='1';
                        when others=>
angka<=angka;segment<=segment;triger<='0';
                end case;
        end if;
end if;
end process;

process(clk,operasi)
begin
if clk='1' and clk'event then
        if reset_all='1' then
                hitung<="000";
        else
        case operasi is
                when "001" => hitung<="001";
                when "010" => hitung<="010";
          when "100" => hitung<="100";
                when others=> hitung<=hitung;
        end case;
        end if;
end if;
end process;
```

```
process(triger)
begin
        if triger='1' and triger'event then
                if A_B<5 then
                        A_B<=A_B+1 ;
                else
                  A_B<=(others=>'0');
                end if;
        end if;
end process;

process (clk,reset_all,A_B)
begin
if clk='1' and clk'event then
        if reset_all='1' then

        A<=0;pulx<="11000000";B<=0;satx<="11000000";C<=0;puly<="110000
00";F<=0;saty<="11000000";
        else
        case A_B is
                when "00"
=>A<=angka;B<=B;C<=C;F<=F;pulx<=segment;satx<=satx;puly<=puly;saty<=s
aty;
                when "01"
=>A<=A;B<=angka;C<=C;F<=F;pulx<=pulx;satx<=segment;puly<=puly;saty<=
saty;x<=(A*10)+B;
                when "10"
=>A<=A;B<=B;C<=angka;F<=F;pulx<=pulx;satx<=satx;puly<=segment;saty<=s
aty;
                when "11"
=>A<=A;B<=B;C<=C;F<=angka;pulx<=pulx;satx<=satx;puly<=puly;saty<=seg
ment;y<=(C*10)+F;
           when others
=>A<=A;B<=B;C<=C;F<=F;x<=x;y<=y;pulx<=pulx;satx<=satx;puly<=puly;saty
<=saty;
        end case;
        end if;
end if;
end process;

process (clk,reset_all,enter,hitung)
begin
if clk='1' and clk'event then
        if reset_all='1' then
                hasil<=0;
```

```
        elsif enter='1' then
                case hitung is
                        when "001" => hasil<=x+y;
                        when "010" => hasil<=x-y;
                  when "100" => hasil<=x*y;
                        when others=> hasil<=hasil;
                end case;
        end if;
end if;
end process;

process(hasil)
begin
case hasil is
        when 0=>pul_hasil<="11111111";sat_hasil<="11000000";
        when 1=>pul_hasil<="11111111";sat_hasil<="11111001";
        when 2=>pul_hasil<="11111111";sat_hasil<="10100100";
        when 3=>pul_hasil<="11111111";sat_hasil<="10110000";
        when 4=>pul_hasil<="11111111";sat_hasil<="10011001";
        when 5=>pul_hasil<="11111111";sat_hasil<="10010010";
        when 6=>pul_hasil<="11111111";sat_hasil<="10000010";
        when 7=>pul_hasil<="11111111";sat_hasil<="11111000";
        when 8=>pul_hasil<="11111111";sat_hasil<="10000000";
        when 9=>pul_hasil<="11111111";sat_hasil<="10010000";
        when 10=>pul_hasil<="11111001";sat_hasil<="11000000";
        when 11=>pul_hasil<="11111001";sat_hasil<="11111001";
        when 12=>pul_hasil<="11111001";sat_hasil<="10100100";
        when 13=>pul_hasil<="11111001";sat_hasil<="10110000";
        when 14=>pul_hasil<="11111001";sat_hasil<="10011001";
        when 15=>pul_hasil<="11111001";sat_hasil<="10010010";
        when 16=>pul_hasil<="11111001";sat_hasil<="10000010";
        when 17=>pul_hasil<="11111001";sat_hasil<="11111000";
        when 18=>pul_hasil<="11111001";sat_hasil<="10000000";
        when 19=>pul_hasil<="11111001";sat_hasil<="10010000";
        when 20=>pul_hasil<="10100100";sat_hasil<="11000000";
        when 21=>pul_hasil<="10100100";sat_hasil<="11111001";
        when 22=>pul_hasil<="10100100";sat_hasil<="10100100";
        when 23=>pul_hasil<="10100100";sat_hasil<="10110000";
        when 24=>pul_hasil<="10100100";sat_hasil<="10011001";
        when 25=>pul_hasil<="10100100";sat_hasil<="10010010";
        when 26=>pul_hasil<="10100100";sat_hasil<="10000010";
        when 27=>pul_hasil<="10100100";sat_hasil<="11111000";
        when 28=>pul_hasil<="10100100";sat_hasil<="10000000";
        when 29=>pul_hasil<="10100100";sat_hasil<="10010000";
        when 30=>pul_hasil<="10110000";sat_hasil<="11000000";
        when 31=>pul_hasil<="10110000";sat_hasil<="11111001";
```

```
when 32=>pul_hasil<="10110000";sat_hasil<="10100100";
when 33=>pul_hasil<="10110000";sat_hasil<="10110000";
when 34=>pul_hasil<="10110000";sat_hasil<="10011001";
when 35=>pul_hasil<="10110000";sat_hasil<="10010010";
when 36=>pul_hasil<="10110000";sat_hasil<="10000010";
when 37=>pul_hasil<="10110000";sat_hasil<="11111000";
when 38=>pul_hasil<="10110000";sat_hasil<="10000000";
when 39=>pul_hasil<="10110000";sat_hasil<="10010000";
when 40=>pul_hasil<="10011001";sat_hasil<="11000000";
when 41=>pul_hasil<="10011001";sat_hasil<="11111001";
when 42=>pul_hasil<="10011001";sat_hasil<="10100100";
when 43=>pul_hasil<="10011001";sat_hasil<="10110000";
when 44=>pul_hasil<="10011001";sat_hasil<="10011001";
when 45=>pul_hasil<="10011001";sat_hasil<="10010010";
when 46=>pul_hasil<="10011001";sat_hasil<="10000010";
when 47=>pul_hasil<="10011001";sat_hasil<="11111000";
when 48=>pul_hasil<="10011001";sat_hasil<="10000000";
when 49=>pul_hasil<="10011001";sat_hasil<="10010000";
when 50=>pul_hasil<="10010010";sat_hasil<="11000000";
when 51=>pul_hasil<="10010010";sat_hasil<="11111001";
when 52=>pul_hasil<="10010010";sat_hasil<="10100100";
when 53=>pul_hasil<="10010010";sat_hasil<="10110000";
when 54=>pul_hasil<="10010010";sat_hasil<="10011001";
when 55=>pul_hasil<="10010010";sat_hasil<="10010010";
when 56=>pul_hasil<="10010010";sat_hasil<="10000010";
when 57=>pul_hasil<="10010010";sat_hasil<="11111000";
when 58=>pul_hasil<="10010010";sat_hasil<="10000000";
when 59=>pul_hasil<="10010010";sat_hasil<="10010000";
when 60=>pul_hasil<="10000010";sat_hasil<="11000000";
when 61=>pul_hasil<="10000010";sat_hasil<="11111001";
when 62=>pul_hasil<="10000010";sat_hasil<="10100100";
when 63=>pul_hasil<="10000010";sat_hasil<="10110000";
when 64=>pul_hasil<="10000010";sat_hasil<="10011001";
when 65=>pul_hasil<="10000010";sat_hasil<="10010010";
when 66=>pul_hasil<="10000010";sat_hasil<="10000010";
when 67=>pul_hasil<="10000010";sat_hasil<="11111000";
when 68=>pul_hasil<="10000010";sat_hasil<="10000000";
when 69=>pul_hasil<="10000010";sat_hasil<="10010000";
when 70=>pul_hasil<="11111000";sat_hasil<="11000000";
when 71=>pul_hasil<="11111000";sat_hasil<="11111001";
when 72=>pul_hasil<="11111000";sat_hasil<="10100100";
when 73=>pul_hasil<="11111000";sat_hasil<="10110000";
when 74=>pul_hasil<="11111000";sat_hasil<="10011001";
when 75=>pul_hasil<="11111000";sat_hasil<="10010010";
when 76=>pul_hasil<="11111000";sat_hasil<="10000010";
when 77=>pul_hasil<="11111000";sat_hasil<="11111000";
```

```
        when 78=>pul_hasil<="11111000";sat_hasil<="10000000";
        when 79=>pul_hasil<="11111000";sat_hasil<="10010000";
        when 80=>pul_hasil<="10000000";sat_hasil<="11000000";
        when 81=>pul_hasil<="10000000";sat_hasil<="11111001";
        when 82=>pul_hasil<="10000000";sat_hasil<="10100100";
        when 83=>pul_hasil<="10000000";sat_hasil<="10110000";
        when 84=>pul_hasil<="10000000";sat_hasil<="10011001";
        when 85=>pul_hasil<="10000000";sat_hasil<="10010010";
        when 86=>pul_hasil<="10000000";sat_hasil<="10000010";
        when 87=>pul_hasil<="10000000";sat_hasil<="11111000";
        when 88=>pul_hasil<="10000000";sat_hasil<="10000000";
        when 89=>pul_hasil<="10000000";sat_hasil<="10010000";
        when 90=>pul_hasil<="10010000";sat_hasil<="11000000";
        when 91=>pul_hasil<="10010000";sat_hasil<="11111001";
        when 92=>pul_hasil<="10010000";sat_hasil<="10100100";
        when 93=>pul_hasil<="10010000";sat_hasil<="10110000";
        when 94=>pul_hasil<="10010000";sat_hasil<="10011001";
        when 95=>pul_hasil<="10010000";sat_hasil<="10010010";
        when 96=>pul_hasil<="10010000";sat_hasil<="10000010";
        when 97=>pul_hasil<="10010000";sat_hasil<="11111000";
        when 98=>pul_hasil<="10010000";sat_hasil<="10000000";
        when 99=>pul_hasil<="10010000";sat_hasil<="10010000";
        when -1=>pul_hasil<="10111111";sat_hasil<="11111001";
        when -2=>pul_hasil<="10111111";sat_hasil<="10100100";
        when -3=>pul_hasil<="10111111";sat_hasil<="10110000";
        when -4=>pul_hasil<="10111111";sat_hasil<="10011001";
        when -5=>pul_hasil<="10111111";sat_hasil<="10010010";
        when -6=>pul_hasil<="10111111";sat_hasil<="10000010";
        when -7=>pul_hasil<="10111111";sat_hasil<="11111000";
        when -8=>pul_hasil<="10111111";sat_hasil<="10000000";
        when -9=>pul_hasil<="10111111";sat_hasil<="10010000";
        when others =>pul_hasil<=pul_hasil;sat_hasil<=sat_hasil;
end case;
end process;

process(clk)
begin
        if clk='1' and clk'event then
                if klok<199999 then
                        klok<=klok+1;
                else
                  klok<=(others=>'0');
                end if;
        end if;
end process;
```

```
process (k)
begin
        if klok(17)'event and klok(17)='1' then
                if k<3 then
                        k<=k+1;
                else
                        k<=(others=>'0');
                end if;
        end if;
        case k is
                when "00"=>an<="0111";d<=pulx;
                when "01"=>an<="1011";d<=satx;
                when "10"=>an<="1101";d<=puly;
                when "11"=>an<="1110";d<=saty;
                when others=>an<=an;d<=d;
        end case;
end process;

process (e)
begin
        if klok(17)'event and klok(17)='1' then
                if e<1 then
                        e<=e+1;
                else
                        e<=(others=>'0');
                end if;
        end if;
        case e is
                when "00"=>am<="01";m<=pul_hasil;
                when "01"=>am<="10";m<=sat_hasil;
                when others=>am<=am;m<=m;
        end case;
end process;

end Behavioral;
```