BAB III

PERANCANGAN SISTEM

3.1 Perancangan Perangkat Keras

Pada penelitian ini, sistem yang akan dibangun adalah perancangan "*running text*" pada LCD (*Liquid Crystal Display*) yang diimplementasikan pada FPGA. Secara keseluruhan sistem ini terdiri dari perangkat keras dan program. Gambar dibawah ini memperlihatkan perancangan sistem yang akan dibangun.

Gambar 3.1. Blok Diagram Perencanaan Sistem.

Masukan dari sistem diatas adalah berasal dari program yang kemudian akan diterjemahkan oleh komputer dan mengirimkan data bit yang dikeluarkan melalui kabel JTAG yang dihubungkan ke *board* FPGA yang telah terpasang rangkaian LCD yang ada di pin *external* dari *board* FPGA. Keluaran yang dihasilkan adalah karakter yang berbentuk teks "TE UII JOGJA" yang akan ditampilkan pada tampilan LCD. Untuk mengubah teks yang ada di tampilan , maka harus merubah kode yang ada di program, dengan mengganti kode ASCII.

3.1.1 FPGA (*Field Programmable Gate Array*)

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.

3.1.2. FPGA Keluarga Xilinx Spartan II

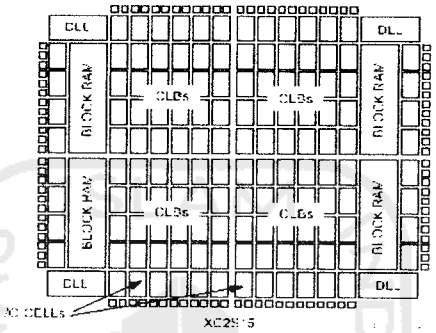
Spartan II merupakan salah satu keluarga FPGA yang dikeluarkan oleh Xilinx. Xilinx merupakan salah satu pabrik pembuat FPGA yang cukup terkenal. Keluarga Spartan II merupakan keluarga FPGA yang memiliki 15.000 sampai 20.000 gerbang. IC Xilinx ini dapat diprogram dan dihapus dengan waktu yang tidak terbatas.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

Tabel 3.1. Data Keluarga Spartan II

Device	Logic Cells	System Gates (Logic and 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

3.1.2.1 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 3.2.



Gambar 3.2. 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*.

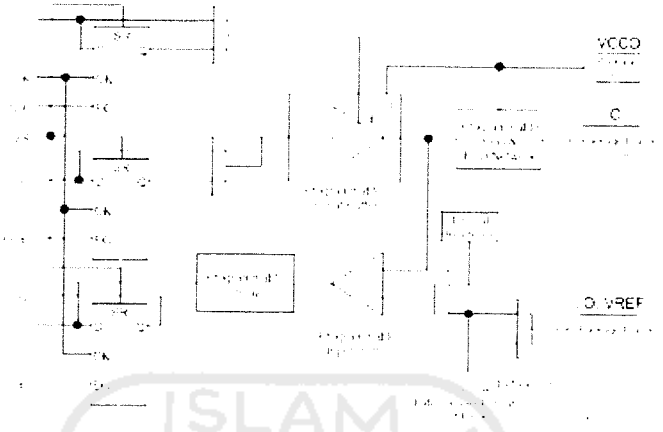
3.1.2.2 Konfigurasi Blok FPGA Spartan II

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

a. Input/Output Blocks (IOB)

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*



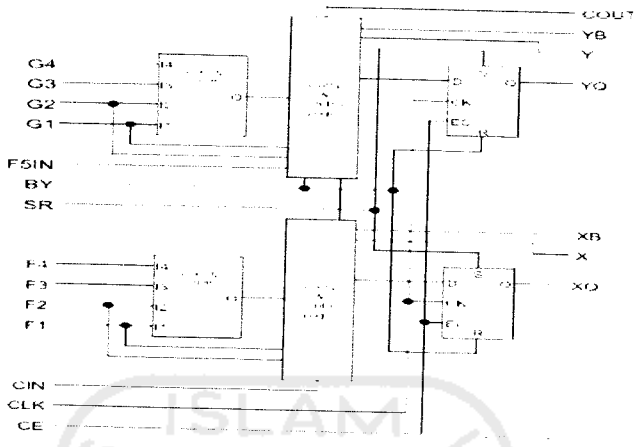
Gambar 3.3. Blok diagram I/O Spartan II

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)

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.



Gambar 3.4. CLB pada Spartan II

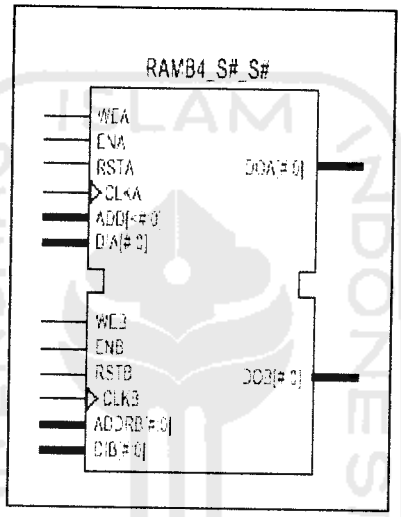
Generator pembangkit fungsi diimplementasikan dalam sebuah *look-up tables* (LUT) 4 *input*. LUT ini juga dapat membangkitkan sebuah RAM 16 x 1 sinkron serta membangkitkan fungsi *shift register* 16 bit. Fungsi RAM yang dibangkitkan generator ini akan melengkapi *block* RAM yang dimiliki oleh Spartan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

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 *clock* yang tersisa tepatnya dalam *flip-flop internal* akan diserempakkan dengan *edges clock.*

d. Blocks Random Access Memory (RAM)



Gambar 3.5. 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.

3.1.2.3 *Programmable Routing Matrix*

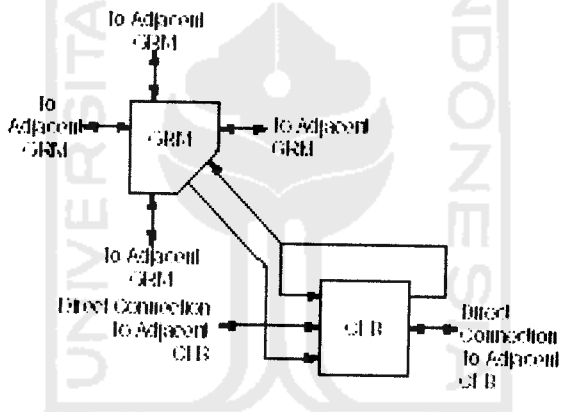
Programmable Routing Matrix merupakan cara sebuah FPGA melakukan *routing* menghubungkan CLB-CLB dan IOB-IOB yang digunakan dalam desain menjadi satu kesatuan sistem. *Routing* ini dilakukan secara otomatis penuh.

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 3.6. Struktur *local routing*

b. General Purpose Routing

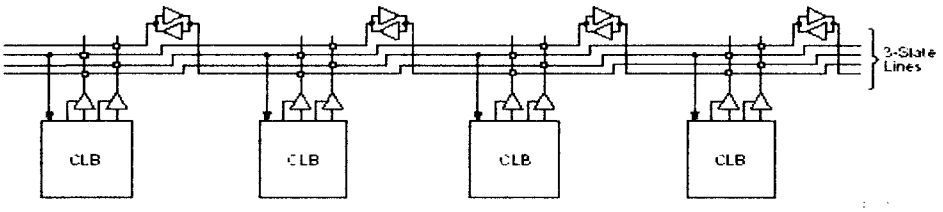
General Purpose Routing digunakan untuk melakukan koneksi vertikal dan horizonal antar kolom dan baris CLB yang digunakan.

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 performasinya.



Gambar 3.7. 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.

3.1.2.4 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.

3.1.2 *Board* Pegasus

Modul Pegasus dikeluarkan oleh pengembang 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. 50 k gerbang Xilinx Spartan II FPGA, dengan 50 k gerbang dasar dan 200 MHz operasi *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.

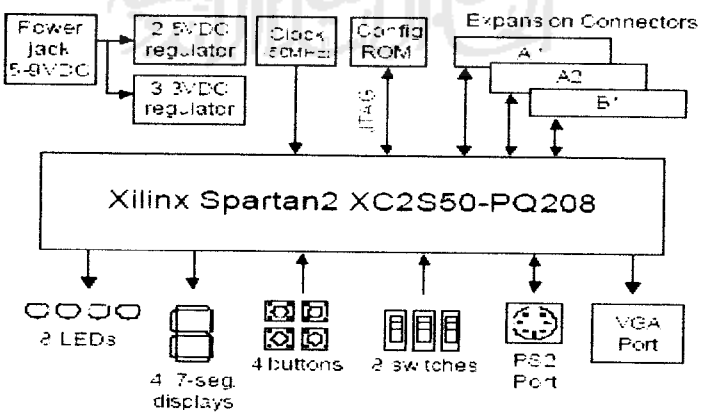
d. 50 MHz *oscilator* dan satu pin untuk *oscilator* tambahan.

e. PS/2 dan VGA port.

f. 96 *Pin* 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.



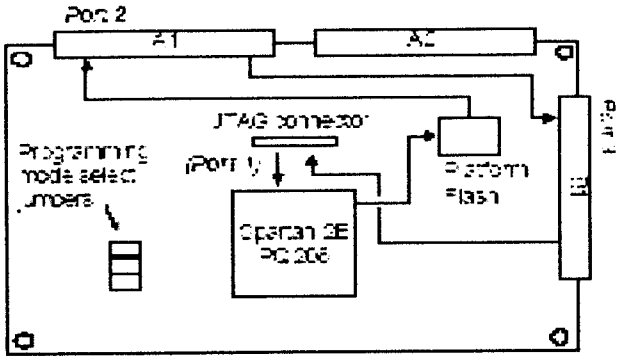
Gambar 3.8. Blok diagram Pegasus

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 secara bersamaan melalui 1 kabel utama, baik itu *port* USB, *Parallel* maupun *Ethernet.*



Gambar 3.9. Aliran scan JTAG pada Pegasus

b. Catu Daya

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

c. Oscilator

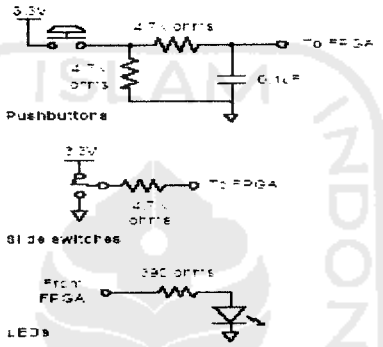
Modul Pegasus menyediakan 50 MHz *oscilator* sebagai *clock* utama untuk aplikasi-aplikasi yang akan dikonfigurasi ke Spartan II XC2S50. *Oscilator* tersebut terhubung langsung ke Spartan II XC2S50 (*pin*77).

d. Saklar *Pushbutton*, saklar geser indikator LED

Empat saklar *pushbutton* dan 8 saklar geser disediakan sebagai masukan. Saklar *pushbutton* dalam keadaan normal menghasilkan 0 Vdc (rendah), dan akan berubah menjadi 3 Vdc sampai dengan 5 Vdc (tinggi) ketika saklar *pushbutton* ditekan. Saklar geser menghasilkan rendah (0 Vdc) atau tinggi (3Vdc sampai dengann 5 Vdc) 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.

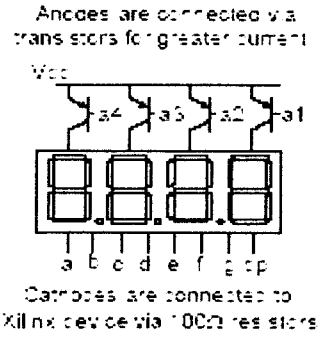
Delapan LED disediakan sebagai keluaran anoda LED terhubung ke *pin* keluaran melalui resistor 390 Ω, sedangkan katoda LED terhubung langsung ke Ground. LED 9 digunakan sebagai *indikator power* FPGA dan LED ke 10 digunakan sebagai *indicator* status pemrograman JTAG.



Gambar 3.10. Rangkaian saklar *pushbutton*, saklar geser, LED

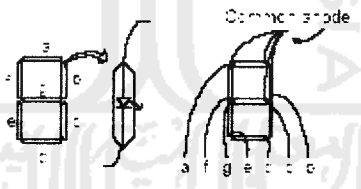
e. Seven Segment

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 3.11. *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 *Anode Not* atau yang disingkat dengan AN0 sampai 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 (0 Vdc). Katoda dari setiap *seven-segment* terhubung ke dalam 7 keluaran, dan diberi nama CA – CG, dan akan aktif apabila ada sinyal/pulsa rendah (0 Vdc).



Gambar 3.12. *Common anode seven-segment* 1 *digit.*

Dilihat dari diagram yang terlihat maka menghasilkan *display seven-segment multiplexer*, 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 atau seolah-olah semua aktif secara bersamaan. Jika diberi sinyal rendah (0 Vdc)

secara terus menerus 1 ms – 16 ms, maka akan terlihat semua *digit* aktif. Dengan syarat, bersamaan dengan *digit enable* data untuk *digit* tersebut harus ada. *Refresh frequency* berkisar 60 Hz sampai dengan 1 KHz.

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

Gambar 3.13. *Pin* penghubung tambahan

Tabel 3.2. *Port accessory*

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	LCE	58	BTN1	110	LMB1-DB1	162	LPA-IO18
7	LDB6	59	BTN0	111	LMB1-DB0	163	LPA-IO17
8	LWE	60	AND	112	LPB-LSBCLK	164	LPA-IO16
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	AC0	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

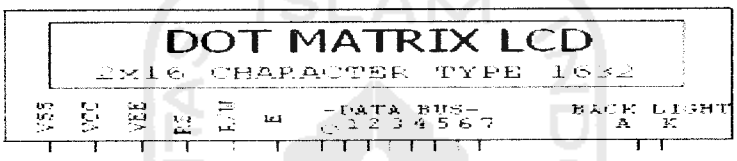
Tabel 3.3. *Pin* FPGA XC2S50

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-IO18
7	LDB6	59	BTN0	111	LMB1-DB0	163	LPA-IO17
8	LWE	60	AND	112	LPB-LSBCLK	164	LPA-IO16
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	VCCINT	65	VCCO	117	VCCO	169	GND
14	LDB4	66	VCCINT	118	VCCINT	170	VCCO
15	LADR4	67	CC	119	LPB-DB6	171	VCCINT
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	VCCINT	128	VCCINT	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	VCCINT
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	VCCINT	90	LMB1-INT	142	LMA2-DSTB	194	LMA1-DSTB
39	VCCO	91	VCCINT	143	VCCINT	195	LMA1-ASTB
40	MC1-DB4	92	GND	144	VCCO	196	VCCINT
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

3.1.3. LCD (*Liquid Crystal Display*)

LCD (*Liquid Crystal Display*) yang digunakan sebagai *prototype* dari sebuah informasi. Agar terhubung dengan FPGA, LCD dilengkapi dengan 8 bit data bus (DB0 - DB7) yang digunakan untuk menyalurkan data ASCII (*American*

Standard Code for Information Interchange) maupun perintah pengatur kerjanya. Modul LCD sendiri terdiri dari *display* dan *chipset*, dimana *chipset* ini sendiri sebenarnya merupakan mikrokontroler. *Chipset* ini berfungsi untuk mengatur tampilan informasi serta berfungsi mengatur komunikasi dengan FPGA yang memakai tampilan LCD. Sebelum merancang, harus kita ketahui dahulu susunan pin dari LCD tersebut. Adapun susunan pin serta bentuk dari standar LCD 16 pin beserta fungsi dari masingn-masing pin adalah seperti pada Gambar 3.2 berikut ini:



Gambar 3.14. LCD 2x16 Karakter

Tabel 3.4. Susunan LCD 2x16

NO	SIMBOL	LEVEL	FUNGSI
1	Vss	-	Power Supply 0 Volt (gnd)
2	Vcc	-	Power Supply 5 Volt ± 10%
3	Vcc	-	Kontras LCD
4	RS	I/O	1 = Data, 0 = Instruksi
5	RW	I/O	1 = Baca, 0 = Tulis
6	EN	1 ke 0	Penyerempak (Clock)
7	DB0	I/O	Bus Data
:	:	-	Bus Data
14	DB7	I/O	Bus Data
15	A	-	Back light 4-42 V, 50-200 mA
16	K	-	Back light 0 V (gnd)

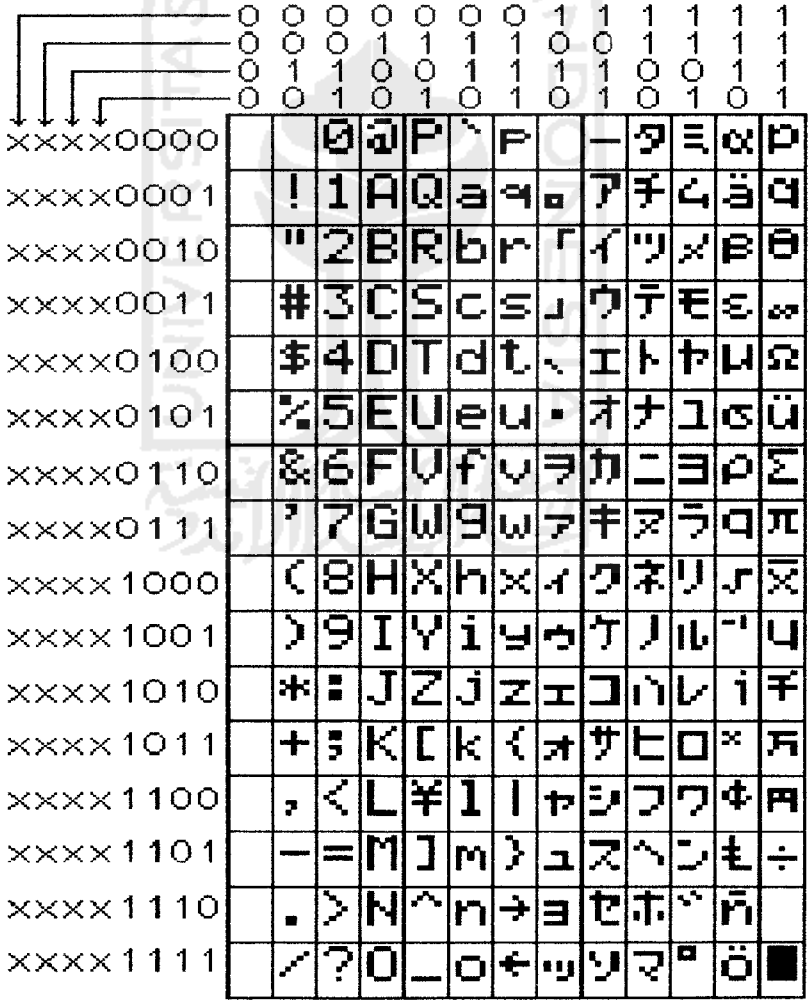
Untuk membuat suatu karakter tampilan LCD pada suatu posisi tertentu harus diketahui dahulu alamat dari LCD itu sendiri, misalnya diinginkan menulis

suatu kata dimulai dari baris kedua kolom pertama, berarti alamat yang digunakan pada LCD adalah *0C0h*, tanda *h* menunjukkan nilai tersebut dalam kode bilangan heksadesimal. Tabel 3.2 berikut merupakan peta alamat LCD degan spesifikasi 2x16 karakter.

Tabel 3.5. Peta alamat LCD 2x16

Peta alamat LCD 2 x 16															
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF

Tabel 3.6. Karakter LCD



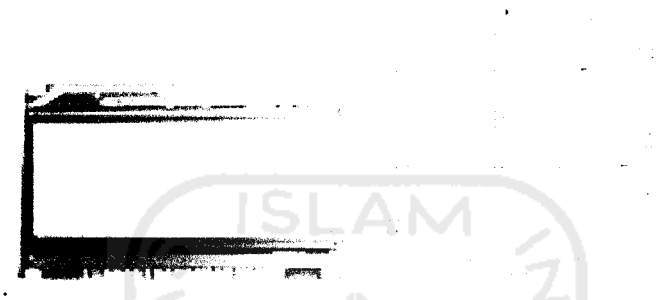
Tabel 3.7. Bilangan Desimal-Biner-Heksadesimal

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

Karakter LCD yang menggunakan kode ASCII memuat karakter Inggris dan karakter kana Jepang yang dipergunakan untuk membuat teks yang diinginkan. Setelah *display* diinisialisasi dan komunikasi ditetapkan, semua perintah dan pengiriman data pada karakter dengan 8 bit, menggunakan 2 kali operasi pengiriman data 4 bit. Tiap-tiap pengiriman data 8 bit harus dirubah menjadi pengiriman data 4 bit, jarak antara pengiriman pertama dan selanjutnya adalah 1 μs, data *upper nibble* dikirimkan pertama kali, diikuti data *lower nibble*. Pada operasi tulis data 8 bit harus ada jarak antar pengiriman 40 μs, sedangkan untuk *delay* harus dinaikkan menjadi 1.64 ms mengikuti perintah *Clear Display*.

LCD yang digunakan merupakan modul LCD dengan tampilan 2 x 16 baris dengan konsumsi daya yang rendah. LCD ini mempunyai CGROM (*Character Generator Read Onli Memory*), CGRAM (*Character Generator*

random Access Memory) dan DDRAM (*Display Data random Access Memory*), dan juga memiliki 3 *bit control* yaitu E yang merupakan *input clock*, RW sebagai *input* untuk memilih *read* atau *write* dan RS sebagai *register select*, juga memiliki 8 bit data yaitu DB0 sampai DB7 (Data Sheet, HD44780U, 2009:Page9)



Gambar 3.15. Konstruksi Dasar *LCD (Liquid Crystal Display)*

3.2. Perancangan Program

3.2.1 Xilinx (Xilinx *Foundation Series*)

Xilinx (Xilinx *Foundation Series*) adalah suatu perangkat lunak untuk merancang IC, yang nantinya hasil dari perancangan di-*download* ke *board* FPGA. Dengan menggunakan Xilinx, proses simulasi rangkaian yang telah dirancang dapat diketahui apakah benar ataukah masih ada yang mengandung kesalahan. Proses perancangan dengan menggunakan Xilinx sama seperti merancang suatu rangkaian logika secara manual, akan tetapi dengan menggunakan simulator Xilinx dapat meminimalisasi kesalahan pada proses perancangan. Untuk proses perancangan rangkaian digital, Xilinx mempunyai 3 cara, yaitu dengan menggunakan *state diagram*, HDL, dan *schematic*, pada proses

perancangan kita dapat menggunakan salah satunya atau menggabungkan ketiganya.

3.2.2 Verilog HDL

Verilog *Hardware Description Language* merupakan suatu bahasa standar yang digunakan untuk mendeskripsikan suatu rangkaian digital. Suatu kode HDL merupakan suatu representasi dari gambar rangkaian digital, dimana pada prakteknya, komputer akan lebih mudah membaca suatu kode bahasa dari pada gambar buatan manusia. Verilog HDL juga mempunyai kemampuan lebih dari sekedar menggambar rangkaian digital, karena Verilog HDL dapat melakukan pemodelan rangkaian digital menggunakan kode sekuensial yang bersifat behavioral. Bagian-bagian dari kode Verilog adalah:

a. *Module description* merupakan bagian awal dari kode verilog yang menyatakan nama modul yang dibuat beserta daftar nama pin input dan output dari module tersebut.

```
module      and_or_demo (A, B, C, R)
input       A, B, C;
output      R;
wire        A, B, C, R;
assign      R= (A | B) & (B | C);
endmodule
```

b. *Port Declaration*, dalam bagian ini, setiap nama signal *input* dan *output* dideklarasikan. Ada tiga macam deklarasi I/O, yaitu *input*, *output* dan *bidirectional* I/O.

c. *Signal declaration* merupakan bagian dimana sinyal-sinyal I/O maupun sinyal internal dideklarasikan. Setiap objek sinyal, baik pin *Input/Output*,

sinyal interkoneksi, maupun sinyal memori baru dideklarasikan dibagian ini. Ada dua macam tipe objek signal, yang pertama adalah tipe *reg* dan yang kedua adalah *wire*.

d. *Parameter Definition* digunakan untuk mendefinisikan konstanta-konstanta dalam beberapa bagian kode, sehingga dapat dihasilkan kode yang lebih solid, lebih mudah dibaca, dan fleksibel terhadap modifikasi. Parameter dapat digunakan sebagai nilai yang di *assign* ke sebuah sinyal, digunakan untuk membuat kode arsitektur rangkaian yang *scalable* (lebar bit suatu sinyal dapat diubah dengan mudah) baik dalam bentuk *combinational assinment* maupun *behavioral assinment.*

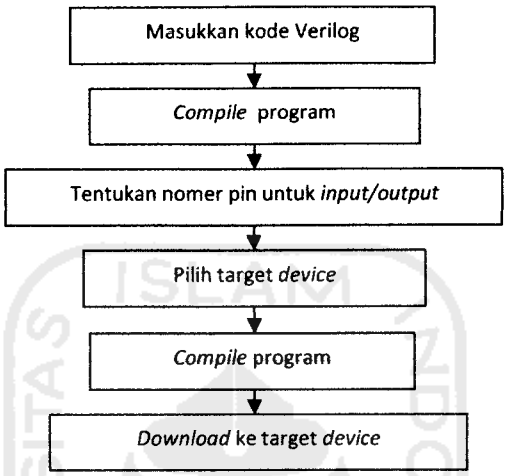
e. *Combinational assignment* merupakan *assignment* nilai dari sinyal dengan menggunakan suatu fungsi kombinasional tertentu.

f. *Behavioral assignment* dapat menghasilkan rangkaian kombinasional yang memiliki perilaku sesuai dengan urutan statement-statement *sequensial* pada deskripsi behavioral.

g. *Component Instantiation* adalah *syntax* utnuk memasukkan desain submodul kedalam modul utama dan menghubungkan sinyal internal dalam modul utama dengan pin I/O submodul tersebut.

Langkah-langkah menggunakan *software* untuk memprogram FPGA dapat dilihat pada Gambar 3.16. Program Verilog dimasukan, untuk mengetahui apakah program yang dibuat tersebut sudah benar maka program di *compile*. Jika sudah dipastikan tidak ada *error* pada program maka pin *input* dan *output* yang digunakan diisikan dengan nomor pin yang dapat dilihat pada Tabel 3.3.

pemilihan peralatan salah satunya dengan pengaturan *inteface* menggunakan JTAG. Setelah semua tahap dilakukan dan tidak terdapat *error* maka program di-*download* ke perangkat yang diinginkan. *Software* menyediakan simulasi desain yang menyajikan pendukung kinerja sistem yang ada.

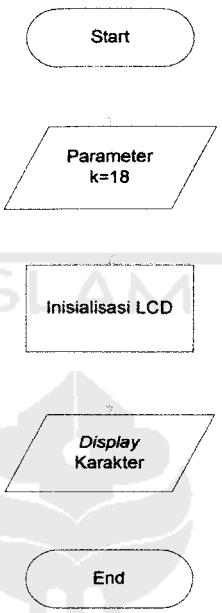


Gambar 3.16. Blok diagram alir penggunaan *software* untuk memprogram FPGA

Pada program, sistem dirancang sedemikian rupa, untuk merealisasikannya dibuat empat blok desain, yakni: *Clock, parameter,* inisialisasi LCD, dan *output* (*display*). Pada perancangan ini menggunakan satu blok parameter, yang berfungsi untuk mengatur keluaran *clock* utama dengan status keluaran yang berubah secara konstan sehingga menghasilkan gelombang kotak atau pulsa. Blok *counter* berfungsi sebagai untuk mencacah jumlah karakter dan kontrol LCD yang akan dikeluarkan pada *display*.

3.2.3. Program Utama

Untuk menjalankan sistem, diperlukan program untuk mengendalikan perangkat keras. Perancangan program sistem *"running text"* pada LCD yang diimplementasikan pada FPGA ditunjukkan pada *flowchart* dibawah ini.



Gambar 3.16. *Flowchart* program.

Perancangan kode program dibawah ini dipakai untuk menampilkan *running text* pada LCD.

```
module lcd (clk, sf_ce0, lcd_rs, lcd_rw, lcd_e, lcd_4, lcd_5,
lcd_6, lcd_7);
parameter    k = 18;
```

Kode diatas merupakan bagian awal program dibuat, yaitu dengan menentukan nama modul yang dibuat beserta nama pin masukan dan keluaran dari modul *running text* pada LCD. Sedangkan parameter berfungsi untuk mendefinisikan konstanta yang dipakai dalam *case*.

```
always @ (posedge clk) begin
count  <= count + 1;
```

Pada kode diatas, *clock* yang diperlukan untuk menjalankan program pada LCD, diberikan secara terus-menerus. Karena *clock* yang diberikan disini untuk men-*drive* pada nilai cacah positif.

```
case (count[k+7:k+2])
 0: lcd_code <= 6'h03;          // Inisialisasi Power-On
 1: lcd_code <= 6'h03;
 2: lcd_code <= 6'h03;
 3: lcd_code <= 6'h02;
 4: lcd_code <= 6'h02;          // function set
 5: lcd_code <= 6'h08;
 6: lcd_code <= 6'h00;          // entry mode set
 7: lcd_code <= 6'h06;
 8: lcd_code <= 6'h00;          // kontrol display on/off
 9: lcd_code <= 6'h0E;
10: lcd_code <= 6'h00;          // display clear
11: lcd_code <= 6'h01;
```

Inisialisasi kontrol LCD diberikan sesuai dengan data yang ada pada *datasheet* LCD yang digunakan. Mulai dari inisialisasi power-on yang diberikan dengan kode 0: lcd_code <= 6'h03 untuk *function set* diberikan kode 4: lcd_code <= 6'h02 untuk *upper nibble*, sedangkan *lower nibble* diberikan kode 5: lcd_code <= 6'h08; dengan data biner "0010 1000" untuk mengatur lebar data, nomer baris pada *display* dan *font* karakter. Untuk pengaturan *entry mode set* yang berfungsi untuk mengatur mode geser diberikan kode 6: lcd_code <= 6'h00; 7: lcd_code <= 6'h06; dengan data biner "0000 0110" yang mengatur perpindahan *running teks* dari kanan kekiri atau dari kiri ke kanan. *Display on/off* diberikan kode 8: lcd_code <= 6'h00; 9: lcd_code <= 6'h0E; dengan data biner "0000 1110". *Display clear* dengan kode biner "0000 0001" yang berguna untuk posisi awal *display* pada pojok kiri, waktu eksekusi dari *display clear*-nya dari 82μs – 1.64 ms.

```
12: lcd_code <= 6'h25;          // T
13: lcd_code <= 6'h24;
14: lcd_code <= 6'h24;          // E
15: lcd_code <= 6'h25;
16: lcd_code <= 6'h22;          //
17: lcd_code <= 6'h20;
18: lcd_code <= 6'h25;          // U
19: lcd_code <= 6'h25;
20: lcd_code <= 6'h24;          // I
21: lcd_code <= 6'h29;
22: lcd_code <= 6'h24;          // I
23: lcd_code <= 6'h29;
24: lcd_code <= 6'h22;          //
25: lcd_code <= 6'h20;
26: lcd_code <= 6'h24;          // J
27: lcd_code <= 6'h2A;
28: lcd_code <= 6'h24;          // O
29: lcd_code <= 6'h2F;
30: lcd_code <= 6'h24;          // G
31: lcd_code <= 6'h27;
32: lcd_code <= 6'h24;          // J
33: lcd_code <= 6'h2A;
34: lcd_code <= 6'h24;          // A
35: lcd_code <= 6'h21;
```

Kode diatas memberikan kode karakter yang berfungsi untuk menampilkan teks pada *display* yang berbentuk " TE UII JOGJA". Kode tersebut memakai kode ASCII yang dikirim secara 2 kali per 4 bit-nya. Untuk karakter A data binernya adalah "0100 0001" yang dirubah dalam bentuk heksadesimal menjadi 'h41 begitupun dengan karakter yang lain juga mempunyai kode yang berbeda-beda.