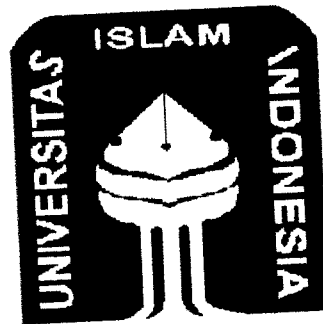


**PERANCANGAN VGA MONITOR SEDERHANA
MENGUNAKAN FPGA XILINX SPARTAN II XC2S50**

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 : ARDZIAN IBNU MUSTOFA

NO.MHS : 02 524 092

**JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
JOGJAKARTA**

2007

LEMBAR PENGESAHAN DOSEN PEMBIMBING

**PERANCANGAN VGA MONITOR SEDERHANA
MENGUNAKAN FPGA XILINX SPARTAN II XC2S50**

TUGAS AKHIR

Oleh :

Nama : ARDZIAN IBNU MUSTOFA

No. Mahasiswa : 02524092


Yogyakarta, November 2007

Pembimbing I,



(Tito Yuwono, ST. MSc)

Pembimbing II,



(Yusuf Aziz Amrullah, ST)

LEMBAR PENGESAHAN DOSEN PENGUJI

**PERANCANGAN VGA MONITOR SEDERHANA
MENGUNAKAN FPGA XILINX SPARTAN II XC2S50**

TUGAS AKHIR

Oleh :

Nama : ARDZIAN IBNU MUSTOFA

No. Mahasiswa : 02524092

Telah Dipertahankan Di Depan Sidang Penguji Sebagai Salah Satu Syarat Untuk
Memperoleh Gelar Sarjana Teknik Elektro, Fakultas Teknologi Industri
Universitas Islam Indonesia
Jogjakarta, 2 Januari 2008

Tim Penguji,

Tito Yuwono, ST. MSc

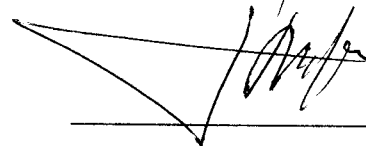
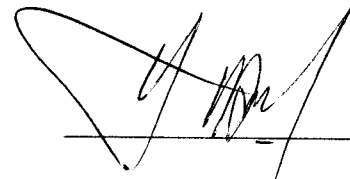
Ketua

Yusuf Aziz Amrullah, ST

Anggota I

Hendra Setiawan, ST.MT

Anggota II



Mengetahui,

Ketua Jurusan Teknik Elektro

Fakultas Teknologi Industri

Universitas Islam Indonesia



(Tito Yuwono, ST. MSc)

Halaman persembahkan

Tugas akhir ini kupersembahkan untuk

Ayahanda dan Ibunda Tersayang

yang telah memberikan seluruh apa yang dimilikinya demi masa depanku wujud kasih sayang, kesabaran dan ketabahan serta keselarasan hidup yang telah kalian tunjukkan telah mengajarku apa arti hidupku ini.

Kakak-kakakku

*Mbak Wafir, mbak Nurul, mbak Hany
terima kasih atas kasih sayang kalian karena kalian ada
maka aku ada*

Nisa(wanita)-ku

*tanpa semangat dan bantuan darimu takkan mengantarku
sejauh ini
tanpa hatimu takkan mengajarku apa makna cinta dan
mungkin aku masih tersesat
thanks for everythink*

Temen-temenku

*Seluruh mahasiswa jurusan Teknik Elektro UII
GILANSEE genk*

Terima kasih kepada semua teman-teman yang telah membantu dan mendukung saya dalam menyelesaikan tugas akhir ini. Semoga dengan adanya tugas akhir ini dapat bermanfaat bagi saya dan teman-teman.

Teman-teman satu kos

*Dan semua orang yang kusayang
dan yang menyayangiku, terima kasih
karena kalian mewarnai hidupku.*



MOTTO

“Sesungguhnya sesudah kesulitan itu ada kemudahan.”

(Q.S. Al Insyirah; 94:5)

“Allah akan mengangkat orang-orang yang beriman dari golonganmu dan juga orang-orang yang dikaruniai ilmu pengetahuan hingga beberapa derajat.”

(Q.S. Al Mujaadilah; 58:11)

“Sebaik-baik manusia ialah orang yang banyak manfaatnya (kebaikannya) kepada manusia lainnya.”

(H.R. Qadla'ie dari Jabir)

KATA PENGANTAR



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 “Perancangan VGA Monitor Menggunakan Xilinx Spartan2 Xc2s50-PQ208”. 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. Allah SWT, terima kasih atas segala karunia, petunjuk, kemudahan, dan perlindungan-Mu.
2. Kedua orang tuaku yang senantiasa memberikan dukungan moril, materi dan doa setiap saat.
3. Bpk Fathul Wahid, ST, M.Sc, selaku Dekan Fakultas Teknologi Industri Universitas Islam Indonesia.

4. Bpk Tito Yuwono, ST. M.Sc., selaku Kajar Teknik Elektro dan sebagai dosen pembimbing 1 serta Bpk Yusuf Aziz Amrulloh, ST., selaku Sekjur Teknik Elektro dan sebagai dosen pembimbing 2 dan juga semua dosen Teknik Elektro UII yang telah memberikan ilmunya.
5. Mas Tri, Mas Heri dan Mas Agung serta serta seluruh teman-teman Lab Digital dan Lab Kendali terima kasih atas bantuanya dan telah mau bertukar pikiran.
6. Mbak Umi dan seluruh staf pengajaran FTI yang telah banyak membantu dalam urusan administrasi.
7. Semua pihak yang telah membantu dalam perancangan dan penyusunan tugas akhir ini yang tidak dapat disebutkan satu persatu disini.

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, November 2007

Ardzian Ibnu Mustofa

ABSTRAK

Perkembangan teknologi komputerisasi saat ini berkembang dengan pesatnya. Dari *processor* sampai monitor selau berkembang dengan beriringan. 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 FPGA sebagai VGA (*video graphic accelerator*) untuk menampilkan karakter pada monitor CRT (*Cathode Ray Tube*). Maka dari itu suatu FPGA akan diaplikasikan sebagai *processor* sekaligus *vga card* seperti pada CPU. FPGA yang digunakan adalah jenis Xilinx Spartan II Xc2s50 – PQ208. Keypad digunakan sebagai masukan ke FPGA yang berupa *code ASCII*. Berdasarkan input yang diberikan tersebut FPGA akan memanggil data karakter yang berbentuk *array* yang membentuk suatu huruf dan ditampilkan pada monitor. Namun karena adanya keterbatasan maka sistem yang dibuatpun hanya mampu menampilkan beberapa karakter saja pada monitor.

DAFTAR ISI

Halaman Judul	i
Lembar Pengesahan Pembimbing.....	ii
Lembar Pengesahan Penguji.....	iii
Halaman Persembahan.....	iv
Halaman Motto	v
Kata Pengantar.....	vi
Abstrak.....	viii
Daftar Isi	ix
Daftar Gambar	xi
Daftar Tabel.....	xii
Bab I PENDAHULUAN	
1.1 Latar Belakang.....	1
1.2 Rumusan Masalah.....	2
1.3 Batasan Masalah	2
1.4 Tujuan Penelitian	3
1.5 Sitematika Penulisan.....	4
Bab II DASAR TEORI	
2.1 Monitor CRT.....	6
2.1.1 R, G, B	8
2.1.2 Hs dan Vs.....	9
2.1.3 VGA sistem <i>Timing</i>	9
2.2 FPGA.....	11
2.3 VHDL.....	12
2.4 <i>Board</i> Pegasus.....	16
2.4.1 Port JTAG dan Pengkonfigurasi Modul.....	17
2.4.2 <i>Power Supply</i>	19
2.4.3 <i>Oscillator</i>	19
2.4.4 Saklar <i>Pushbutton</i> , Saklar Geser, Indikator LED.....	19

2.4.5	<i>Seven Segment</i>	20
2.4.6	<i>Port I/O Tambahan</i>	22
Bab III PERANCANGAN SISTEM		
3.1	<i>Keypad</i>	27
3.2	<i>Pengendali (controller)</i>	28
3.3	<i>Port VGA</i>	28
3.4	<i>Display</i>	29
3.5	<i>Pembuatan software</i>	29
3.5.1	<i>Pembangkit clock 25MHz</i>	30
3.5.2	<i>Input</i>	30
3.5.3	<i>Character Map</i>	30
3.5.4	<i>VGA Controller</i>	31
3.6	<i>Simulasi</i>	32
3.7	<i>Test</i>	32
Bab IV ANALISA DAN PEMBAHASAN		
4.1	<i>Hasil Simulasi Sistem</i>	33
4.1.1	<i>Clock 25MHz</i>	33
4.1.2	<i>Keypad</i>	34
4.1.3	<i>Program Utama</i>	36
4.2	<i>Pembentuk Mode 640 x 480 dan Pewarnaan</i>	38
4.3	<i>Display Area</i>	39
Bab V PENUTUP		
5.1	<i>Kesimpulan</i>	41
5.2	<i>Saran</i>	41
DAFTAR PUSTAKA		42
LAMPIRAN		43

DAFTAR GAMBAR

Gambar 2.1	Bagian-bagian monitor.....	7
Gambar 2.2	<i>Timing of a line</i>	10
Gambar 2.3	<i>Timing of a frame</i>	11
Gambar 2.4	Papan sirkuit Pegasus.....	11
Gambar 2.5	Diagram penggunaan software untuk memprogram PLD	13
Gambar 2.6	Contoh skema rangkaian digital.....	15
Gambar 2.7	Blok diagram Pegasus.....	17
Gambar 2.8	Aliran scan JTAG pada Pegasus	18
Gambar 2.9	Rangkaian saklar <i>pushbutton</i> (a), saklar geser (b), LED (c).....	20
Gambar 2.10	<i>Common anode seven-segment</i> 4 digit.....	20
Gambar 2.11	<i>Common anode seven-segment</i> 1 digit.....	21
Gambar 2.12	Diagram waktu dari <i>selector AN</i>	21
Gambar 2.13	<i>Pin</i> penghubung tambahan.....	22
Gambar 3.1	Blok diagram perancangan sistem	26
Gambar 3.2	Rangkaian RC untuk input FPGA(a), <i>keypad</i> (b).....	27
Gambar 3.3	Konektor VGA.....	28
Gambar 3.4	Blok diagram desain program.....	29
Gambar 3.5	<i>Bit map</i> huruf A.....	30
Gambar 4.1	Hasil simulasi Pembangkit sinyal 25MHz.....	34
Gambar 4.2	Hasil simulasi dari <i>keypad</i>	35
Gambar 4.3	Hasil simulasi dari <i>output</i>	36

DAFTAR TABEL

Tabel 2.1 Tabel warna 1 bit	8
Tabel 2.2 Tabel sistem <i>timing</i> pada monitor.....	10
Tabel 2.3 <i>Port accessory</i>	22
Tabel 2.4 Konfigurasi <i>pin</i> pada modul Pegasus.....	23
Tabel 2.5 <i>Pin</i> FPGA XC2s50.....	24

BAB I

PENDAHULUAN

1.1 Latar Belakang

Field-Programmable Gate Arrays (FPGA) adalah salah satu piranti yang termasuk dalam kelompok *programmable logic devices*. FPGA sebagai suatu *platform* untuk implementasi sistem digital merupakan alternatif yang menarik dan berpotensi menjadi komponen utama masa depan. Bahasa pemrograman yang biasa digunakan adalah VHDL (*Very High Speed Integrated Circuit Hardware description Language*) yang merupakan bahasa level tinggi.

FPGA dapat diimplementasikan bermacam-macam salah satunya yaitu sebagai VGA suatu monitor. Meskipun Indonesia lemah di sisi industri komponen, paling tidak teknologi perancangan "silikon" dapat dikuasai. Oleh karena itu penelitian ini akan mengimplementasikan FPGA menjadi VGA monitor. Jika dalam perancangan-perancangan FPGA yang sebelumnya untuk menampilkan data menggunakan *seven-segment*, dengan terciptanya VGA yang menggunakan FPGA maka *seven-segment* dapat diganti dengan monitor. Dengan sistem ini maka suatu FPGA dapat digunakan sebagai pengganti CPU.

Dalam perancangan ini mengaplikasikan FPGA sebagai penampil karakter. Hal ini sangat berguna bagi dunia industri yang membutuhkan monitor sebagai sarana menampilkan informasi.

1.2 Rumusan Masalah

Dari latar belakang dan dasar pemikiran di atas, maka dapat diambil suatu rumusan masalah sebagai berikut:

1. Pada FPGA terdapat *port vga* untuk monitor. Jadi penelitian ini akan mencari tahu bagaimana merancang kode VHDL agar FPGA dapat digunakan sebagai VGA (*video graphic accelerator*).
2. Sebuah monitor dalam rangka menampilkan sebuah gambar membutuhkan beberapa sinyal input. Jadi penelitian kali ini akan menentukan sinyal *output* dari FPGA yang dibutuhkan monitor sehingga monitor dapat menampilkan data yang diberikan oleh user.

1.3 Batasan Masalah

Untuk menyederhanakan masalah, menghindari kerancuan dan agar tidak menyimpang dari apa yang akan diteliti maka dibuat batasan masalah sebagai berikut:

1. Dalam perencanaan pembuatan VGA dari FPGA ini menitik beratkan pada pembuatan perangkat lunak atau *software* dalam bahasa VHDL (*Very High Speed Integrated Circuit Hardware description Language*) dengan menggunakan XILINX atau Active-HDL.
2. Dalam FPGA ini akan diprogram agar dapat menampilkan data yang berupa karakter dari suatu keypad. Data yang diberikan akan ditampilkan pada sebuah monitor CRT.

1.4 Tujuan Penelitian

Adapun 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. Mengetahui lebih mendalam mengenai teori-teori dasar elektronika, sistem digital dan sekaligus dalam prakteknya.
2. Mempelajari, mengerti dan memahami FPGA serta prinsip kerja dalam aplikasinya.
3. Mempelajari dan mengerti perangkat lunak (*software*) yang digunakan dalam membuat program pada FPGA terutama bahasa VHDL.
4. Mengetahui prinsip kerja dari alat yang dibuat sehingga dapat digunakan sesuai dengan fungsinya.
5. Merancang kode VHDL agar FPGA Xilinx Spartan II Xc2s50 – PQ208 dapat difungsikan sebagai VGA *card* suatu monitor sehingga monitor yang langsung dihubungkan ke FPGA dapat menampilkan data yang diberikan.

1.5 Metodologi Penelitian

Dalam melaksanakan tugas akhir ini, maka diperlukan tahap-tahap yang perlu dilalui. Hal ini berguna agar diperoleh suatu hasil yang maksimal. Adapun metode yang digunakan adalah:

1. Studi literatur untuk mengumpulkan dan mempelajari bahan-bahan pustaka yang berhubungan dengan permasalahan yang dihadapi.

2. Perancangan sistem yang akan dibuat, meliputi perancangan *software* dan signal yang dibutuhkan monitor.
3. Pengujian sistem, meliputi pengujian dari keseluruhan sistem.

1.6 Sistematika Penulisan

Dalam sistematika penulisan tugas akhir ini diberikan uraian bab demi bab yang berurutan untuk mempermudah pembahasannya.

Pokok-pokok permasalahan dalam penulisan ini dibagi menjadi lima bab :

BAB I : PENDAHULUAN

Bab ini memuat tentang latar belakang masalah, Tujuan dan Manfaat Penelitian, Rumusan Masalah, Batasan Masalah, Manfaat Penulisan, Metodologi Penulisan, dan Sistematika Penulisan.

BAB II : LANDASAN TEORI

Bab ini memuat teori-teori yang berhubungan dengan FPGA dan monitor serta dasar teori yang berhubungan dengan teknik atau cara pembuatan VGA menggunakan FPGA.

BAB III : PERANCANGAN SISTEM

Bab ini menjelaskan metode-metode perancangan yang digunakan, cara mensimulasikan rancangan dan pengujian sistem yang telah dibuat, pembagian fungsi kerja dalam diagram blok serta batasan dan hambatan yang ditemui selama proses perancangan dan implementasi sistem.

BAB IV : PENGUJIAN, ANALISIS DAN PEMBAHASAN

Bab ini membahas tentang hasil pengujian dan analisis dari sistem yang dibuat dibandingkan dengan dasar teori sistem atau sistem yang lain yang dapat dijadikan sebagai pembandingan .

BAB V : PENUTUP

Bab ini memuat kesimpulan dan saran-saran dari proses perancangan guna pengembangan dimasa yang akan datang.

BAB II

DASAR TEORI

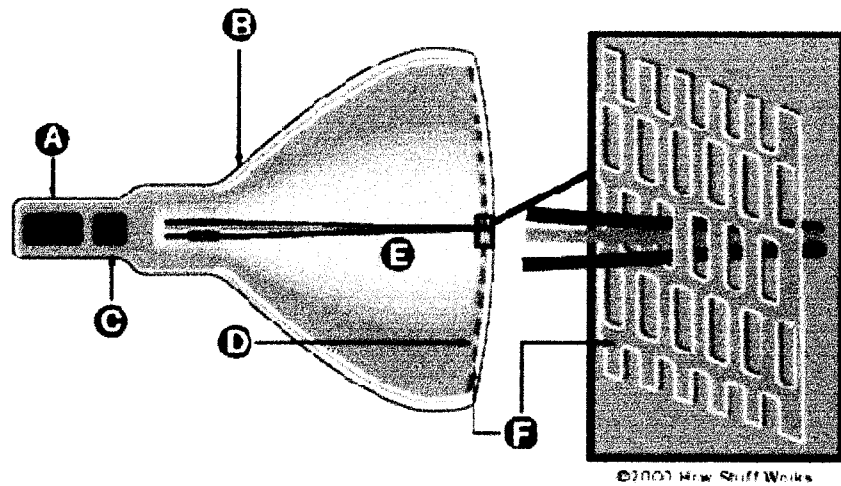
2.1 Monitor CRT (*Cathode RayTube*)

Prinsip kerja monitor konvensional, monitor CRT (*Cathode Ray Tube*), sama dengan prinsip kerja televisi yang berbasis CRT. Elektron ditembakkan dari belakang tabung gambar menuju bagian dalam tabung yang dilapis elemen yang terbuat dari bagian yang memiliki kemampuan untuk memancarkan cahaya. Sinar elektron tersebut melewati serangkaian magnet yang membelok-belokkan sinar menuju bagian-bagian tertentu dari tabung bagian dalam.

Begitu sinar tersebut sampai ke bagian kaca tabung TV atau monitor, maka akan menyinari lapisan tersebut, menyebabkan tempat-tempat tertentu untuk menyala secara temporer. Setiap tempat tertentu mewakili pixel tertentu. Dengan mengontrol tegangan dari sinar tersebut, terciptalah teknologi yang mampu mengatur pixel-pixel tersebut untuk berpendar dengan intensitas cahaya tertentu. Dari pixel-pixel tersebut, dapat dibentuklah gambar.

Teorinya, untuk membentuk sebuah gambar, sinar tadi menyapu sebuah garis horizontal dari kiri ke kanan, menyebabkan pixel-pixel tadi menyala dengan intensitas cahaya sesuai dengan tegangan yang telah diatur. Proses tersebut terjadi pada semua garis horizontal yang ada pada pixel layar, dan ketika telah sampai

ujung, sinar tersebut akan mati sementara untuk mengulang proses yang sama untuk menghasilkan gambar yang berbeda.



- | | |
|-----------------------------|---------------------------------|
| A Cathode | D Phosphor-coated screen |
| B Conductive coating | E Electron beams |
| C Anode | F Shadow mask |

Gambar 2.1 Bagian-bagian dari monitor CRT.

Sesaat sebelum elektron (*electron beams*) tersebut menyentuh *phosphor-coated screen*, mereka melalui sebuah *shadow mask* atau *aperture grille* yang terletak sepersekian inci di belakang layar, yang menyaring tembakan elektron tersebut agar mengenai fosfor yang tepat. Pada sebuah monitor CRT *shadow mask*, selembar metal yang memiliki lubang-lubang mengarahkan elektron yang ditembakkan pada lingkaran *phosphor-coated screen*. Pada monitor CRT *aperture grille* sinar diarahkan langsung melalui slot diantara kawat vertikal yang tipis. Pada kedua jenis monitor tersebut, ruang diantara lubang atau kawat tersebut (yang dikenal sebagai "dot pitch" pada jenis *shadow mask* dan "grille pitch" pada jenis *aperture grille*)

menentukan seberapa detail gambar yang dihasilkan oleh monitor: Secara garis besar, semakin kecil pitch, semakin presisi penempatan sinar tersebut, sehingga semakin jelas gambar yang ditampilkan.

Untuk menampilkan gambar suatu monitor memerlukan 5 signal input yaitu:

1. R (*red signal*).
2. G (*green signal*).
3. B (*blue signal*).
4. HS (*horizontal synchronization*).
5. VS (*vertical synchronization*).

2.1.1 R, G, B

Red, *Green* dan *Blue* adalah input yang dibutuhkan oleh monitor dimana pengiriman data dan penentuan warna adalah melalui R, G, B ini.

Tabel 2.1 Tabel warna 1 bit.

Color	Red	Green	Blue
Black	0	0	0
Blue	0	0	1
Green	0	1	0
Cyan	0	1	1
Red	1	0	0
Purple	1	0	1
Yellow	1	1	0
White	1	1	1

Tabel 2.1 adalah logika pembentuk warna yang akan kita gunakan untuk menentukan warna text yang akan kita tampilkan.

2.1.2 HS dan VS (*horizontal dan vertical synchronization*).

Sinyal dan *h_sync* dan *v_sync* dibutuhkan untuk mengaktifkan monitor saat *h_sync* dan *v_sync high* berarti monitor aktif sedangkan jika *h_sync* dan *v_sync* rendah monitor tidak aktif atau dalam keadaan *sleep mode* .

2.1.3 VGA Sistem Timing

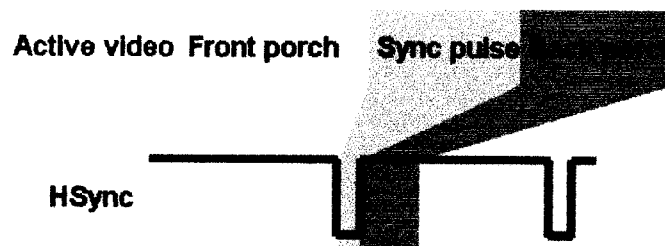
Dalam rangka untuk menampilkan gambar pada suatu monitor, gambar diletakkan pada suatu bingkai atau *frame*. Untuk itu dibutuhkan garis horisontal (pixel) dan garis vertikal (line) untuk membangun suatu *frame*. Untuk menentukan nilai-nilai dari garis horisontal dan garis vertikal disebut VGA sistem *timing*.

VGA sistem *timing* dibutuhkan untuk menentukan mode dan pengaturan *Vsync* dan *Hsync*-nya. Jadi pada sebuah mode akan mempunyai frekuensi serta nilai horisontal dan vertikal *synchronization* agar monitor dapat bekerja pada mode tersebut. Dibawah ini adalah tabel sistem timing dari monitor CRT dimana untuk setiap mode mempunyai *clock*, *active video*, *front porch*, *sync pulse*, *back porch* yang berbeda-beda. Nilai-nilai dibawah ini bukanlah harga mati hanya sistem *timing* standar monitor. Monitor akan dapat bekerja pada suatu mode asalkan nilai *Vsync* dan *Hsync*-nya sesuai.

Tabel 2.2 Tabel sistem *timing* pada monitor

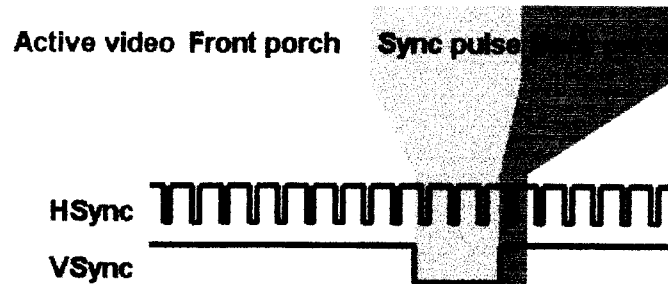
Format	Pixel Clock (MHz)	Horizontal (in Pixels)				Vertical (in Lines)			
		Active Video	Front Porch	Sync Pulse	Back Porch	Active Video	Front Porch	Sync Pulse	Back Porch
640x480, 60Hz	25.175	640	16	96	48	480	11	2	31
640x480, 72Hz	31.500	640	24	40	128	480	9	3	28
640x480, 75Hz	31.500	640	16	96	48	480	11	2	32
640x480, 85Hz	36.000	640	32	48	112	480	1	3	25
800x600, 56Hz	38.100	800	32	128	128	600	1	4	14
800x600, 60Hz	40.000	800	40	128	88	600	1	4	23
800x600, 72Hz	50.000	800	56	120	64	600	37	6	23
800x600, 75Hz	49.500	800	16	80	160	600	1	2	21
800x600, 85Hz	56.250	800	32	64	152	600	1	3	27
1024x768, 60Hz	65.000	1024	24	136	160	768	3	6	29
1024x768, 70Hz	75.000	1024	24	136	144	768	3	6	29
1024x768, 75Hz	78.750	1024	16	96	176	768	1	3	28
1024x768, 85Hz	94.500	1024	48	96	208	768	1	3	36

Sumber: Rick Ballantyne, Xilinx Inc.

**Gambar 2.2** *Timing of a line.*

Masing-Masing garis video mulai dengan suatu daerah *active video*, di mana nilai RGB adalah *output* untuk masing-masing pixel dalam suatu baris. Daerah aktif diikuti oleh suatu daerah yang kosong, di mana pixel hitam dipancarkan. Pada

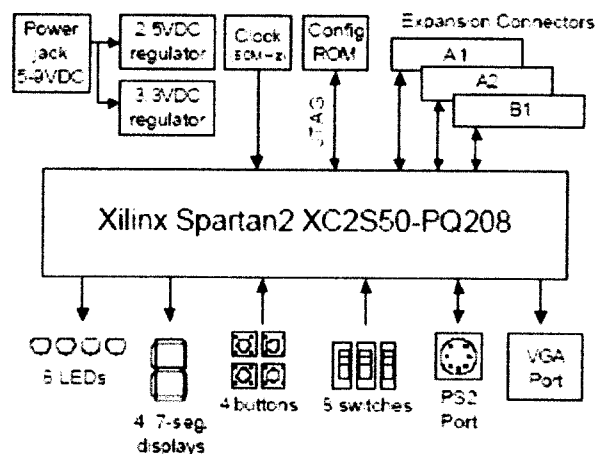
pertengahan interval yang kosong, suatu *sync pulse* horisontal dipancarkan. Di depan *sync pulse* terdapat *front porch* dan dibelakang *sync pulse* terdapat *back porch*.



Gambar 2.3 *Timing of a frame.*

Bingkai (*frame*) dibentuk dengan cara yang hampir sama seperti membentuk baris dari pixel-pixel. Permulaan suatu bingkai berisi semua baris yang akan ditampilkan pada layar, diikuti oleh *front porch*. Pada pertengahan interval Vsync yang kosong, suatu *sync pulse* vertikal dipancarkan. Dan dibelakang *sync pulse* terdapat *back porch*.

2.2 FPGA XC2S50



Gambar 2.4 Papan sirkuit Pegasus

FPGA (*Field Programmable Gate Arrays*) adalah salah satu piranti yang termasuk dalam kelompok PLD (*Programmable Logic Devices*). Media perancangan berkembang pesat semenjak ditemukannya PLD. Piranti PLD ini merupakan sebuah piranti yang dapat diprogram isinya dan dapat dihapus jika terjadi kesalahan. FPGA berbeda dari *general-purpose* mikroprosesor (misalnya intel) dalam hal fleksibilitas *logic*-nya. Karena pada FPGA tidak terbatas akan pemakaian register-register dan dapat mengimplementasikan rangkaian kombinasi maupun rangkaian sekuensial.

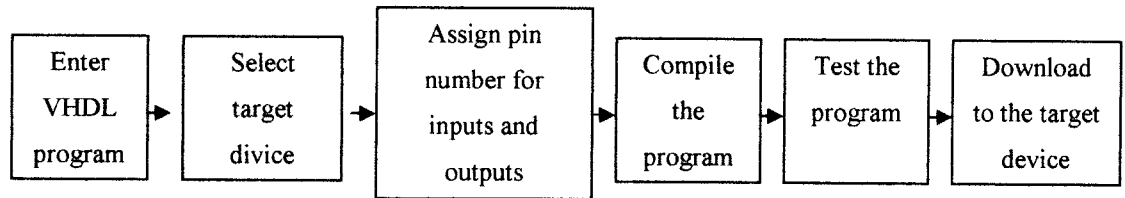
Keistimewaan piranti-piranti yang berbasis PLD adalah:

- a) Mudah diprogram sehingga memudahkan perubahan desain jika terdapat kesalahan.
- b) Bahasa pemrograman yang digunakan dapat lebih dari satu jenis, misalnya menggunakan *schematic diagram*, *state diagram* maupun *hardware description language* (HDL).
- c) Harga tergolong murah untuk pembuatan sebuah *prototype*.

2.3 VHDL

VHDL merupakan singkatan dari *Very High Speed Integrated Circuit Hardware Description Language*. VHDL merupakan bahasa level tinggi yang dapat digunakan untuk mendeskripsikan dan mengoptimalkan watak sebuah sistem digital. VHDL merupakan bahasa pemodelan yang telah memiliki standar internasional sehingga mudah digunakan dimana saja. Hal ini cukup penting karena jika bahasa yang digunakan tidak standar bisa dimungkinkan adanya kesalahan deskripsi perangkat kerasnya. VHDL adalah salah satu jenis *hardware description language*

(HDL) yang digunakan untuk memprogram PLD. Langkah-langkah menggunakan *software* untuk memprogram PLD dapat dilihat pada Gambar 2.5.



Gambar 2.5 Blok diagram penggunaan software untuk memprogram PLD.

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

VHDL dapat digunakan pada saat akan:

- a) Implementasi sistem digital dengan struktur “*top-down*”.
- b) Meningkatkan produktivitas hasil. Karena menggunakan bahasa level tinggi sehingga desain dapat dilakukan dalam waktu yang lebih singkat jika dibandingkan dengan metode *schematic diagram*.
- c) Membuat *prototype*.
- d) Merancang sistem yang berubah-ubah.
- e) Merancang sistem yang selalu berkembang I/O dan struktur internalnya.

VHDL dapat digunakan untuk melakukan desain sebuah sistem sampai dengan implementasinya, dengan kemungkinan untuk mencampur beberapa level abstraksi bahasa. Berikut ini adalah level abstraksi bahasa mulai dari yang tertinggi sampai yang rendah:

a) Level abstraksi *behavior*

Bahasa yang mudah dipahami oleh manusia harus dispesifikasikan lagi menjadi bahasa deskripsi yang dapat disimulasikan. Dalam level ini sebuah sistem digital didefinisikan sebagai operasi yang dilakukan sistem terhadap waktu. Konsep waktu menjadi pembeda utama antara level ini dengan level yang lain. Dalam level *behavior* tunda waktu juga dapat dimasukkan kedalam perhitungan proses. Level inilah yang paling mirip dengan bahasa pemrograman yang lazim digunakan pada komputer dan paling mudah untuk dipelajari oleh manusia.

b) Level abstraksi *dataflow*

Pada level ini sistem digital dideskripsikan dengan cara bagaimana data berpindah-pindah dalam sistem yang diinginkan. Karena hampir semua piranti digital sekarang menggunakan register, maka level abstraksi *dataflow* mendeskripsikan bagaimana data berpindah-pindah diantara register yang ada dalam sistem.

c) Level abstraksi struktur

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

VHDL memiliki sintak-sintak tertentu yang harus dipatuhi agar sistem yang diinginkan dapat terwujud dengan baik. Aturan dasar untuk menulis sebuah kode VHDL adalah sebagai berikut:

1. Dalam sebuah desain VHDL minimal terdapat satu pasang *entity* dan *architecture*.
2. Entity mendeskripsikan unjuk kerja sistem jika dilihat dari perspektif *input/output*. Entity tidak memuat deskripsi kerja (behaviour/dataflow/struktire) dari sistem yang dibuat.
3. Architecture berisi deskripsi kerja dari sistem yang dibuat.
4. Sintaknya:

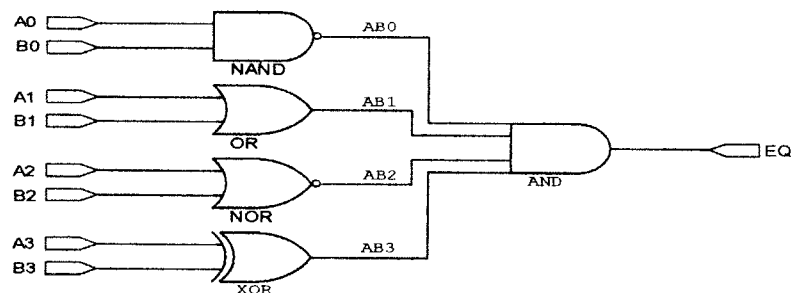
```

Entity <nama_rangkaian> is
port ([signal]<nama>:[<mode>]<indikasi_type>;
.....
[signal]<nama>:[<mode>]<indikasi_type>);
end <nama_rangkaian>;

architecture <nama_desain> of <nama_rangkaian> is
begin
<deskripsi_kerja_dari_sistem>;
end <nama_desain>;

```

Berikut ini adalah contoh dari code VHDL



Gambar 2.6 Contoh skema rangkaian digital.

Berdasarkan logika pada Gambar 2.6 maka dapat dibuat code VHDL sebagai berikut:

```

library ieee;
use ieee.std_logic_1164.all;
entity example is
    Port ( A0,A1,A2,A3,B0,B1,B2,B3 : in std_logic;
          EQ : out std_logic);
end example;
architecture Behavioral of example is
signal AB0,AB1,AB2,AB3:std_logic;
begin
    AB0<=A0 nand B0;
    AB1<=A1 or B1;
    AB2<=A2 nor B2;
    AB3<=A3 xor B3;
    EQ<=AB0 and AB1 and AB2 and AB3;
end Behavioral;

```

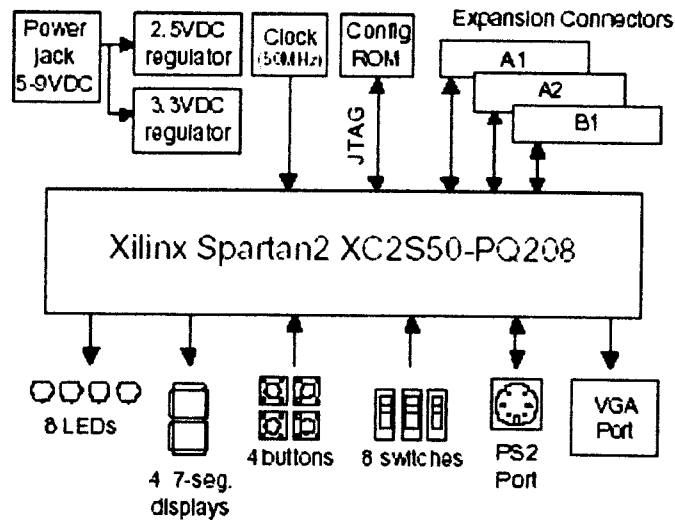
2.4 Board Pegasus

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 Spartan 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 *osilator* dan satu pin untuk *osilator* 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 pengamanan.

h. *Port* pemrograman mode JTAG



Gambar 2.7 Blok diagram Pegasus.

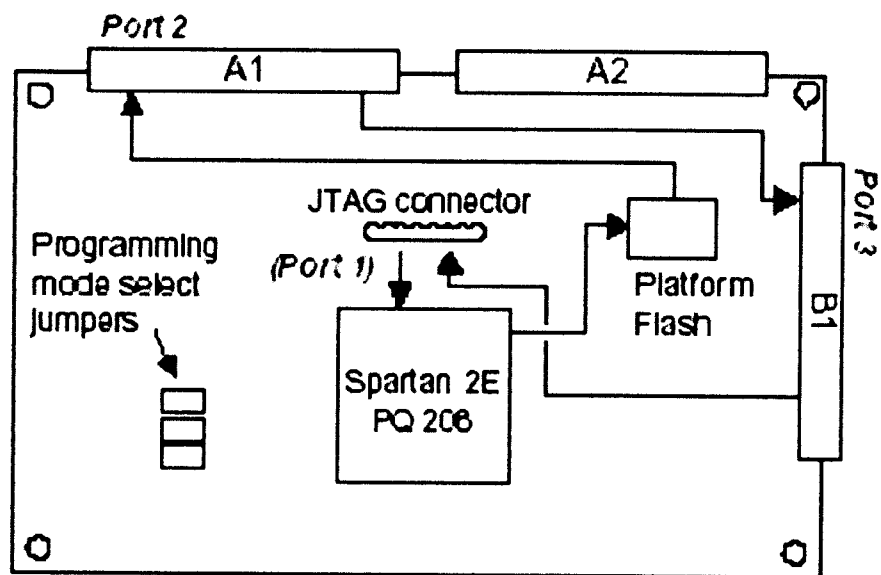
Modul Pegasus sudah dapat dibilang sangat komplit karena hanya dengan menggunakan modul ini, sudah dapat mendownload *design* dan menjalankannya tanpa perlu menambah komponen lain, karena telah tersedia beberapa masukan dan keluaran.

2.4.1 *Port* JTAG dan pengkonfigurasi Modul

Pada modul Pegasus ini 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 dikonfigurasi melalui *port* JTAG (*port* 1), dari modul utama. Dengan menggunakan *port* JTAG dapat diketahui tipe IC, 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 (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 tersebut, maka dapat dikonfigurasi secara bersamaan melalui 1 kabel utama, baik itu *port* USB, Paralell maupun Ethernet.



Gambar 2.8 Aliran scan JTAG pada Pegasus.

2.4.2 *Power supply*

Modul Pegasus memerlukan 5Vdc *power supply*, sedangkan untuk I/O dapat menggunakan *power* dari luar 3Vdc-5Vdc. *Power supply* ini juga digunakan untuk mengaktifkan 8 LED, 4 *display seven-segment*,, sedangkan masukan +5Vdc untuk saklar *pushbutton* dan saklar geser, serta sumber *clock internal* dan *power port* tambahan.

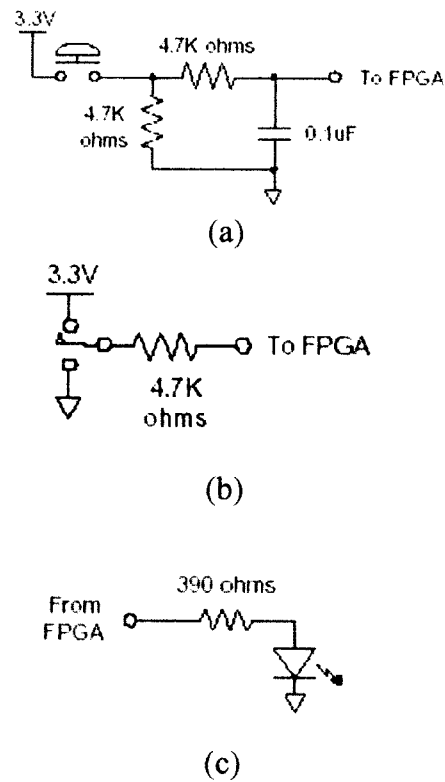
2.4.3 *Osilator*

Modul Pegasus menyediakan 50MHz *osilator* sebagai *clock* utama untuk aplikasi-aplikasi yang akan dikonfigurasi ke Spartan2 XC2S50. *osilator* tersebut terhubung langsung ke Spartan2 XC2S50 (pin77).

2.4.4 *Saklar Pushbutton, Saklar geser, Indikator LED*

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

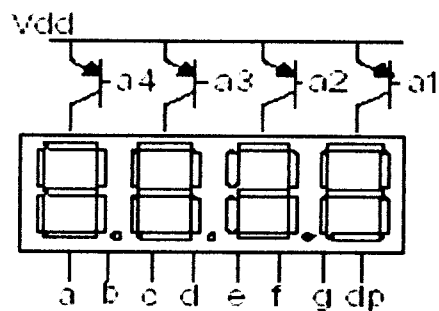
8 LED disediakan sebagai keluaran anoda LED terhubung ke *pin* keluaran melalui resistor 390 Ω , sedangkan katoda LED terhubung langsung ke *grouound*. LED ke 9 digunakan sebagai *indikator power* FPGA, dan LED ke 10 digunakan sebagai *indikator* status pemrograman JTAG.



Gambar 2.9 Rangkaian saklar *pushbutton* (a), Saklar geser (b), LED (c).

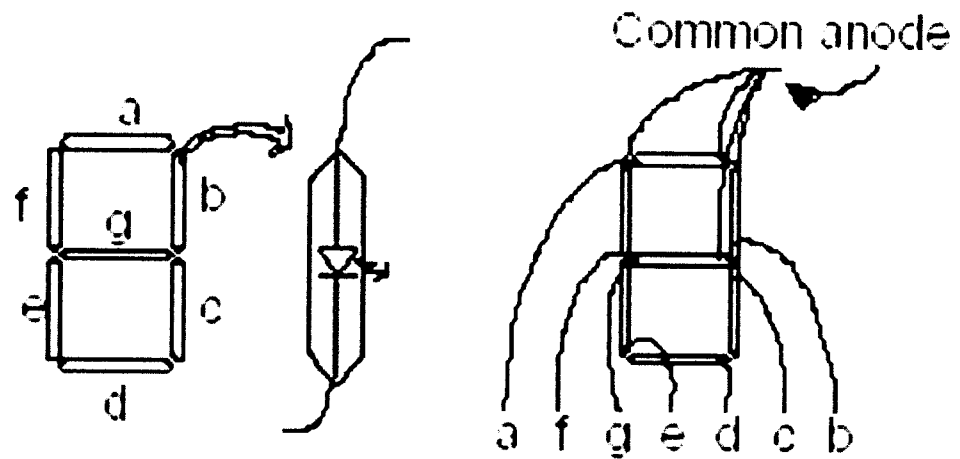
2.4.5 Seven-Segment

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

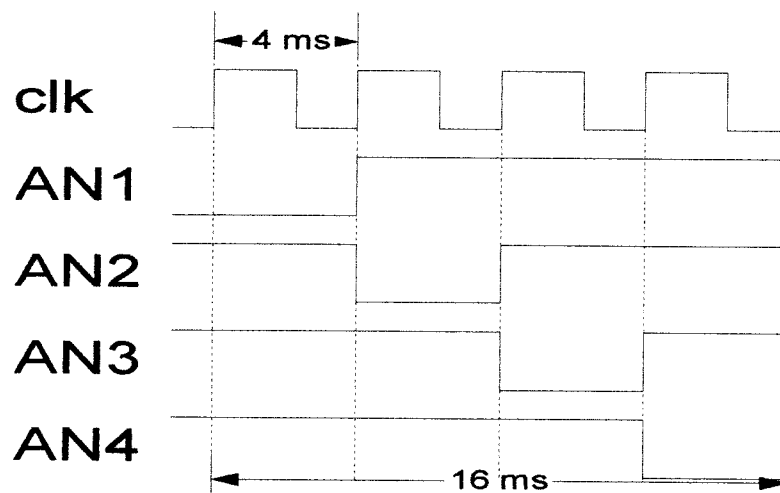


Gambar 2.10 *Seven-segment common anoda* 4 digit.

Ketujuh anoda dari 4 *seven-segment* saling tersambung kedalam *selector* “*common anode*”, *display* ini memiliki 4 selektor yang dinamakan AN1-AN4, apabila ada sinyal /pulsa yang mengaktifkan selektor ini maka *digit* dari *selector* 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.



Gambar 2.11 -segment common anoda 1 digit.

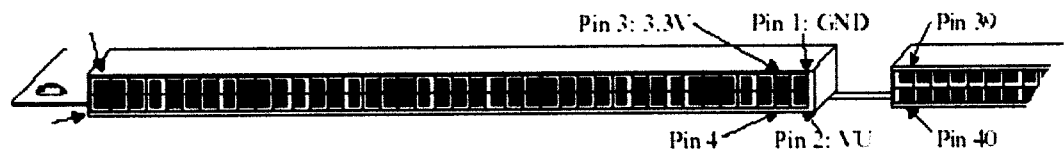


Gambar 2.12 Diagram waktu dari selektor AN.

Dilihat dari diagram yang terlihat maka menghasilkan *display seven-segment multiplexer*, yang mana bila anoda atau *seletor* (AN1-AN4) diaktifkan maka digit tersebut yang akan *aktif* dan mendapat sinyal/pulsa katoda (CA-CG). Jika dilakukan secara terus-menerus dan bila benar maka ke 4 *digit* akan terlihat seolah-olah semua aktif secara bersamaan.

2.4.6 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 dan *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 2.13 *Pin* penghubung tambahan.

Tabel 2.3 *Port accessory*

<i>Accessory Port Pin out</i>					
<i>Pin</i>	Name	FPGA <i>Pin</i>	<i>Pin</i>	Name	FPGA <i>Pin</i>
1	AC0	P49	4	AC3	P47
2	AC1	P48	5	GND	-
3	AC2	P81	6	Vdd	-

Setiap *pin* pada masing-masing I/O tersebut memiliki alamat sendiri-sendiri. Dan pengalamatan *pin* harus benar-benar sesuai dengan alamat *pin* yang ada jika tidak maka I/O tidak berfungsi pada *pin* yang digunakan. Alamat dari *pin* tersebut dapat dilihat pada Tabel 2.4.

Tabel 2.4 Konfigurasi *pin* yang terdapat pada modul Pegasus

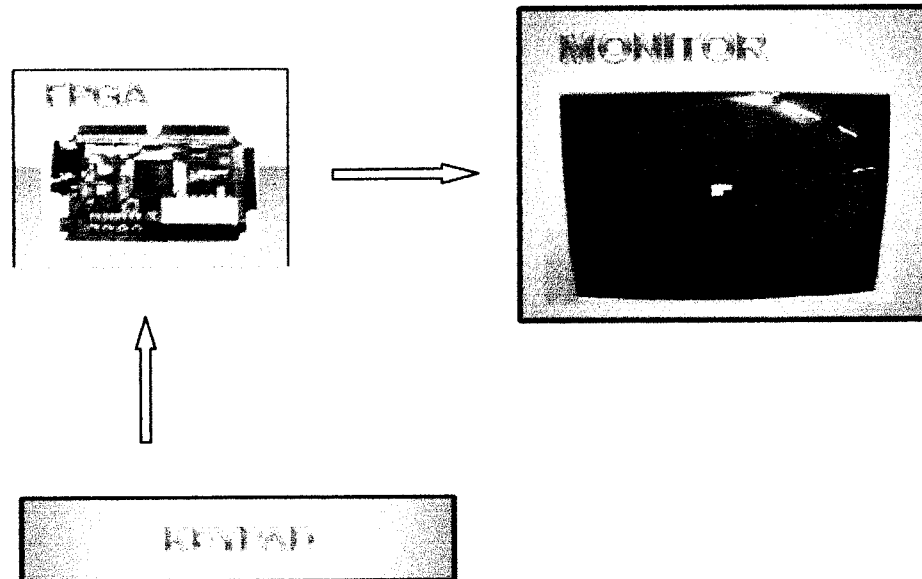
PEGASUS Expansion Conector Pinout								
Conector B1			Conector A1			Conector A2		
Pin	Signal	B1	Pin	Signal	A1	Pin	Signal	A2
1	GND	GND	1	GND	GND	1	GND	GND
2	VU	VU	2	VU	VU	2	VU	VU
3	VCCO	VCCO	3	VCCO	VCCO	3	VCCO	VCCO
4	P-ADR0	136	4	ADR0	24	4	P-IO1	188
5	P-DB0	135	5	DB0	23	5	P-IO2	187
6	P-ADR1	134	6	ADR1	22	6	P-IO3	181
7	P-DB1	133	7	DB1	21	7	P-IO4	180
8	P-ADR2	132	8	ADR2	20	8	P-IO5	179
9	P-DB2	129	9	DB2	18	9	P-IO6	178
10	P-ADR3	127	10	ADR3	17	10	P-IO7	176
11	P-DB3	126	11	DB3	16	11	P-IO8	175
12	P-ADR4	125	12	ADR4	15	12	P-IO9	174
13	P-DB4	123	13	DB4	14	13	P-IO10	173
14	P-ADR5	122	14	ADR5	10	14	P-IO11	172
15	P-DB5	121	15	DB5	9	15	P-IO12	168
16	P-WE	120	16	WE	8	16	P-IO13	167
17	P-DB6	119	17	DB6	7	17	P-IO14	166
18	P-OE	115	18	OE	6	18	P-IO15	165
19	P1-DB7	114	19	DB7	5	19	P-IO16	164
20	P-CSA	113	20	CSA	4	20	P-IO1	163
21	P-LSBCLK	112	21	MA1-DB0	3	21	P-IO18	162
22	MB1-DB0	111	22	MA1-DB0	206	22	MA2-DB0	161
23	MB1-DB1	110	23	MA1-DB1	205	23	MA2-DB1	160
24	MB1-DB2	109	24	MA1-DB2	204	24	MA2-DB2	152
25	MB1-DB3	108	25	MA1-DB3	203	25	MA2-DB3	151
26	MB1-DB4	10	26	MA1-DB4	202	26	MA2-DB4	150
27	MB1-DB5	101	27	MA1-DB5	201	27	MA2-DB5	149
28	MB1-DB6	100	28	MA1-DB6	200	28	MA2-DB6	148
29	MB1-DB7	99	29	MA1-DB7	199	29	MA2-DB7	147
30	MB1-ASTB	98	30	MA1-ASTB	195	30	MA2-ASTB	146
31	MB1-DSTB	97	31	MA1-DSTB	194	31	MA2-DSTB	142
32	MB1-WRIT	96	32	MA1-WRIT	193	32	MA2-WRIT	141
33	MB1-WAIT	95	33	MA1-WAIT	192	33	MA2-WAIT	140
34	M1-RST	94	34	M1-RST	191	34	MA2-RST	139
35	MB1-INIT	90	35	MA1-INIT	189	35	MA2-INIT	138
36	GND	GND	36	GND	GND	36	Not used	n/c
37	TMS	TMS	37	TMS	TMS	37	n/c	n/c
38	TCK	TCK	38	TCK	TCK	38	n/c	n/c
39	TDO	TDO	39	TDO	TDO	39	GCK0	GCK0
40	TDI	TDI	40	TDI	TDI	40	GND	GND

Tabel 2.5 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	GND
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	AN0	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

Pada modul Pegasus ada beberapa *pin* yang memiliki fungsi khusus dan *pin* tersebut juga dapat digunakan jika dibutuhkan. Dan tentu untuk menggunakannya harus dilakukan pemanggilan yang sesuai dengan alamat *pin* yang akan digunakan. Dan *pin* khusus tersebut memiliki alamat yang tertera sesuai pada Tabel 2.5.

Bab III
PERANCANGAN SISTEM



Gambar 3.1 Blok diagram perancangan sistem.

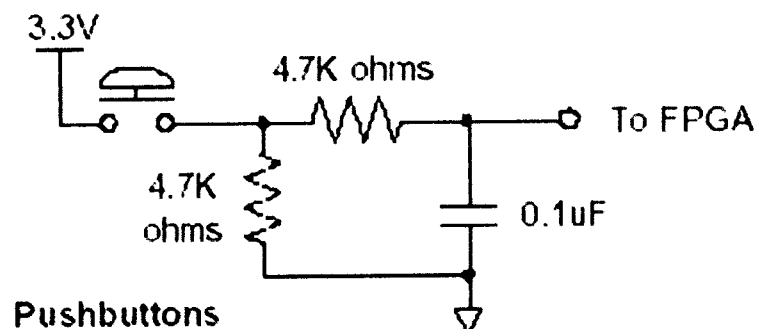
Gambar diatas adalah blok diagram yang menggambarkan sistem yang akan dirancang. Data yang diberikan melalui *keypad* akan ditampilkan dalam VGA monitor.

Fungsi masing - masing blok :

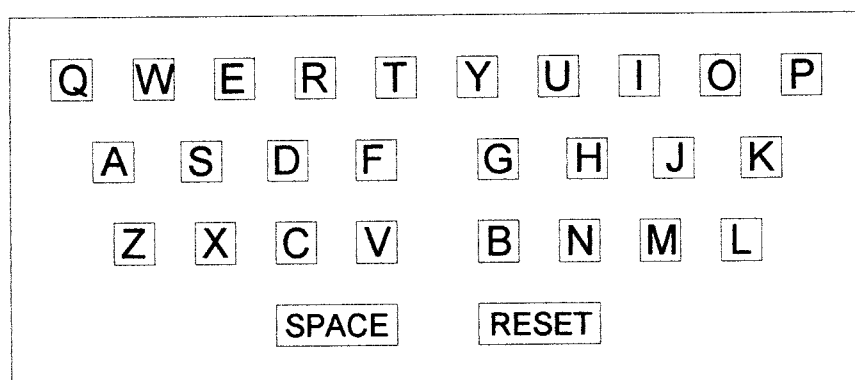
1. *Keypad* : sebagai input atau tempat memberikan data. Melalui *keypad* user dapat menentukan karakter apa yang akan ditampilkan pada monitor.
2. *FPGA board XC2S50* : akan mengolah data yang diberikan melalui *keypad* menjadi sinyal video.
3. Monitor akan menampilkan huruf yang telah diketik pada *keypad*.

3.1 Keypad

Keypad digunakan sebagai masukan, *port* yang digunakan sebagai masukan adalah *Port B1* pada modul pegasus dengan jumlah *pin* sebanyak 28 *pin*, dan satu *pin* untuk *ground* (*pin 1*) satu *pin* lagi untuk positif 3,3 Vdc (*pin 2*). Masing-masing *pin* mewakili 1 masukan. Setiap masukan berupa *pushbutton* harus melalui rangkaian RC terlebih dahulu, agar didapatkan masukan yang stabil dari *keypad*, sehingga FPGA dapat mengeksekusi perintah dengan benar, hal dikarenakan FPGA merupakan sistem digital (*high* dan *low*) dan dengan rangkaian RC maka masukan yang dihasilkan pasti berupa *low* (0) atau *high* (1).



(a)



(b)

Gambar 3.2 Rangkaian RC untuk *input* FPGA (a), *Keypad* (b).

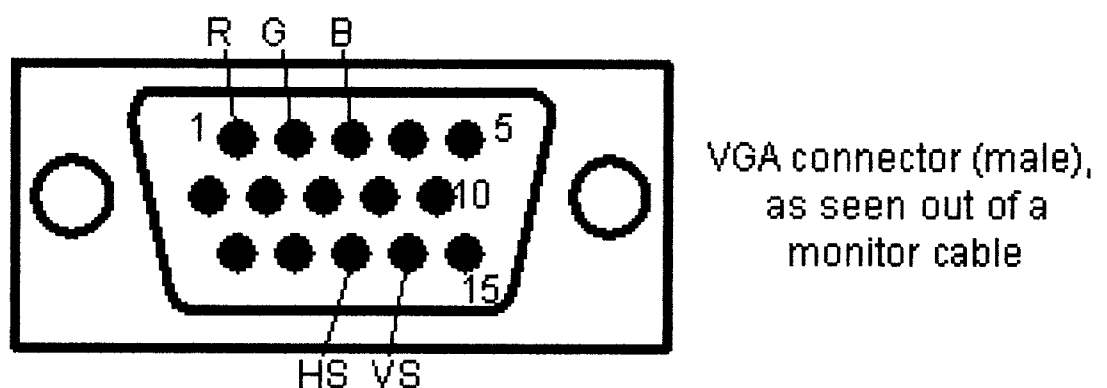
Masing-masing tombol pada *keypad* terbagi beberapa fungsi sebagai berikut:

1. Tombol A-Z : sebagai masukan karakter yang akan ditampilkan pada monitor.
2. Tombol *space* : sebagai tombol spasi.
3. Tombol reset : sebagai tombol reset keseluruhan, sehingga monitor akan menjadi *blank*.

3.2 Pengendali (*Controller*)

Pengendali yang digunakan adalah FPGA XC2S50. Blok ini berfungsi sebagai otak atau CPU, karena dalam blok semua proses komputasi diproses dan di eksekusi. Ketika ada data yang dimasukkan oleh *user* melalui *keypad* maka FPGA akan memproses dan mengolah data tersebut menjadi sinyal video, kemudian sinyal video tersebut akan ditampilkan ke monitor.

3.3 Port VGA



Gambar 3.3 konektor VGA.

Gambar 3.3 diatas adalah *port* VGA DB 15 yang terdapat pada FPGA sebagai penghubung antara monitor dan FPGA. Pin 1 atau *Red* pada FPGA mempunyai

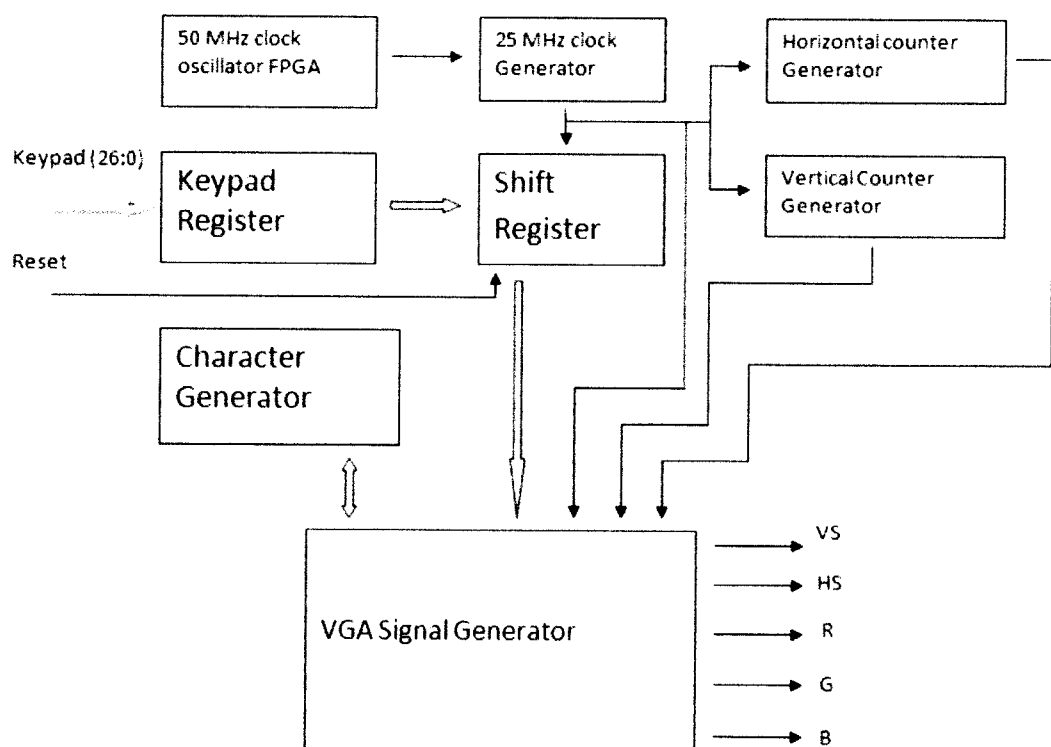
alamat pin p33, untuk G atau *Green* p31, untuk B atau *Blue* p30 sedangkan untuk Hs p29 dan Vs p27.

3.4 Display

Pada bagian ini digunakan monitor CRT sebagai penampil karakter yang telah diketik oleh *user*. Mode yang akan dipakai adalah 640 x 480 dengan frekuensi 25MHz.

3.5 Pembuatan *Software* (perangkat lunak)

Perancangan *software* dibuat dalam bahasa VHDL dengan menggunakan Xilinx.



Gambar 3.4 Blok Diagram Desain Program.

3.5.1 Pembangkit *clock* 25MHz

Pada bagian ini akan dibuat suatu proses dalam program untuk membangkitkan *clock* dengan frekuensi 25MHz karena untuk mode monitor 640 x 480 membutuhkan *clock* frekuensi 25Mhz sedangkan *clock* pada FPGA XC2s50 adalah 50MHz.

3.5.2 Input

Input yang berupa keypad mempunyai beberapa fungsi, yaitu untuk mengeluarkan kode ASCII dari A sampai Z dan space yang akan digunakan untuk memanggil data yang telah dibuat dari *character map* serta fungsi yang kedua adalah sebagai trigger yang nanti dipakai dalam pembentukan *case digit* dalam program.

3.5.3 *Character Map*

Pada bagian ini akan dibentuk data untuk menampilkan huruf A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z dan space. Data dibuat dalam bentuk *array* sebagai contoh untuk huruf A dengan ukuran 12 x 8 pixel adalah sebagai berikut :

ASCII : 65	0 to 7	7 downto 0
	08 00001000	00010000 10
	1C 00011100	00111000 38
	36 00110110	01101100 6C
	63 01100011	11000110 C6
	63 01100011	11000110 C6
	7F 01111111	11111110 FE
	63 01100011	11000110 C6
	63 01100011	11000110 C6
	63 01100011	11000110 C6
	63 01100011	11000110 C6
	00 00000000	00000000 00
	00 00000000	00000000 00

Gambar 3.5 *Bitmap* huruf A.

Dari gambar *Bitmap* huruf A diatas maka dibuat data *array*-nya menjadi :

```
(( '0','0','0','1','0','0','0','0'),
  ( '0','0','1','1','1','0','0','0'),
  ( '0','1','1','0','1','1','0','0'),
  ( '1','1','0','0','0','1','1','0'),
  ( '1','1','0','0','0','1','1','0'),
  ( '1','1','1','1','1','1','1','0'),
  ( '1','1','0','0','0','1','1','0'),
  ( '1','1','0','0','0','1','1','0'),
  ( '1','1','0','0','0','1','1','0'),
  ( '1','1','0','0','0','1','1','0'),
  ( '0','0','0','0','0','0','0','0'),
  ( '0','0','0','0','0','0','0','0'));
```

Data diatas ini adalah yang nantinya akan dikeluarkan melalui R G B dimana untuk bit '0' adalah pixel yang mati sedangkan bit '1' adalah pixel yang hidup sehingga akan membentuk huruf A .

3.5.4 *VGA Controller*

Dalam bagian ini berisi kode VHDL yang membentuk *output* 5 sinyal video yaitu R, G, B, Vs dan Hs. R, G, B, atau singkatan dari *Red*, *Green*, *Blue* adalah sinyal video untuk mengeluarkan data dan membentuk karakter serta menentukan warna dari karakter melalui sinyal ini.

Sedangkan untuk membuat mode horisontal 640 dan vertikal 480 menggunakan *horisontal counter* dan *vertical counter* untuk mencacahnya. Untuk mode 640 x 480 membutuhkan *horisontal counter* sepanjang 800 dan *vertical counter* sepanjang 524. Angka ini didapat dari panjang horisontal *active video* 640 ditambah *front porch* 16 *sync pulse* 96 dan *back porch* 48 sedangkan untuk lebar vertikal *active video* 480 ditambah *front porch* 11 *sync pulse* 2 dan *back porch* 31.

3.6 Simulasi

Setelah semua program dibuat terlebih dahulu disimulasikan dengan menggunakan Active-HDL untuk melihat apakah program yang telah dibuat menghasilkan *output* atau tidak.

3.7 Test

Setelah simulasi dilakukan maka langkah selanjutnya program yang dibuat dalam Xilinx di-*download* pada FPGA dengan menggunakan JTAG *cable*. Metode yang paling mudah adalah dengan *Boundary-Scan Mode*.

BAB IV

ANALISA DAN PEMBAHASAN

Pada bab ini akan dibahas mengenai keseluruhan pengujian sistem. Materi pengujian sistem meliputi dua bagian yakni dengan simulasi dan pengujian pada monitor. Simulasi dilakukan untuk melihat proses yang terjadi saat dilakukan pemberian input melalui *keypad* serta *output* dari sinyal R, G, B, Vs dan Hs.

4.1 Hasil Simulasi Sistem

Setelah program untuk mengkonfigurasi FPGA selesai dibuat, kemudian untuk memastikan kinerja dari program apakah sudah sesuai dengan yang diinginkan maka ada baiknya program tersebut disimulasikan terlebih dahulu. Simulasi dilakukan perbagian dari setiap bagian pada sistem menggunakan Active-HDL. Hasil dari simulasi adalah berupa *waveform* atau *timing* diagram.

4.1.1 Clock 25Mhz

Untuk membuat *clock* 25MHz dapat dilihat pada penggalan *source code* VHDL berikut ini :

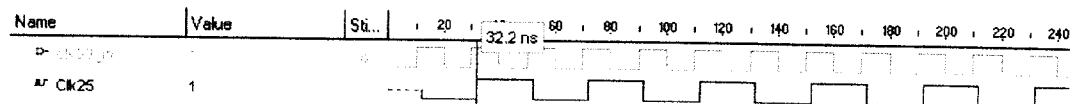
```
process (clk50_in)
begin
    if clk50_in'event and clk50_in='1' then
        if (Clk25 = '0')then
```

```

        Clk25 <= '1' after 2 ns;
    else
        Clk25 <= '0' after 2 ns;
    end if;
end if;
end process;

```

Dari program pembangkit sinyal 25Mhz diatas dapat dilihat hasil simulasinya sebagai berikut :



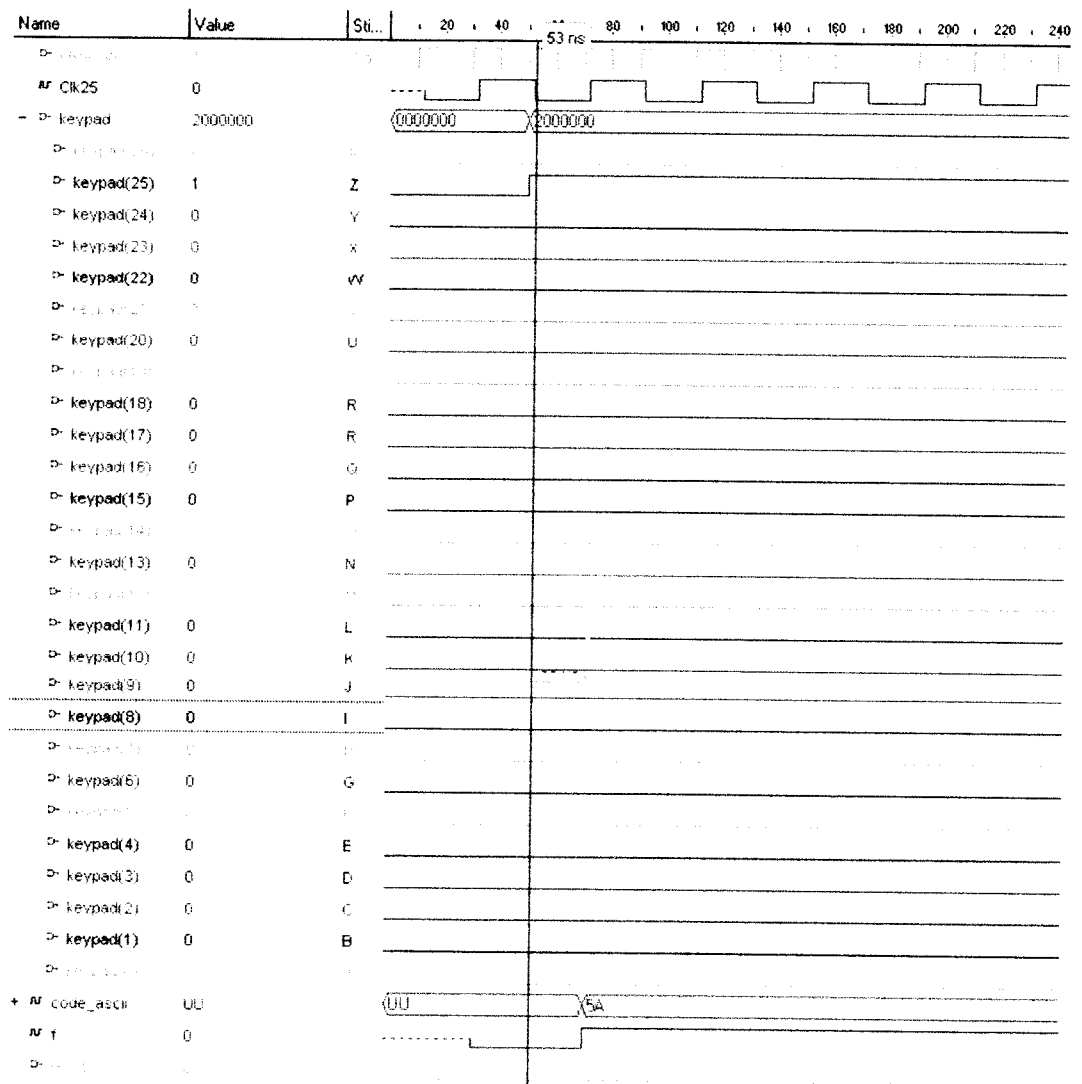
Gambar 4.1 Hasil simulasi pembangkit sinyal 25Mhz.

Setelah 10ns *clock* 50Mhz (clk50_in) yang merupakan *clock internal* pada FPGA mulai berdetak sepanjang 20 ns. Untuk menghasilkan 25Mhz atau *clock* sepanjang 40ns maka buat saja 2 ns setelah clk50_in naik bernilai 1 atau *event*, clk25 bernilai 1, dan setelah *clock* clk50_in mencapai 1 interval atau 20 ns maka 2 ns setelah *clock* clk50_in naik clk25 dibuat bernilai 0 maka akan mendapatkan *clock* 25Mhz dengan panjang 1 interval clk25 adalah 40ns. Nilai 2 ns diatas hanya sebagai *delay* saja jika tidak menginginkan *delay* maka “*after 2 ns*” pada program dapat dihapus saja.

4.1.2 Keypad

Pada *keypad* terdapat 27 tombol masukan untuk huruf yakni tombol 0 sampai dengan 26 yang pada Gambar 4.1 ditunjukkan *keypad* (0) sampai dengan *keypad* (25) untuk huruf A sampai Z dan *keypad* (26) sebagai tombol reset.

Pada simulasi ini akan mencoba untuk memberi masukan pada sistem yakni huruf Z yang terlihat sebagai berikut :

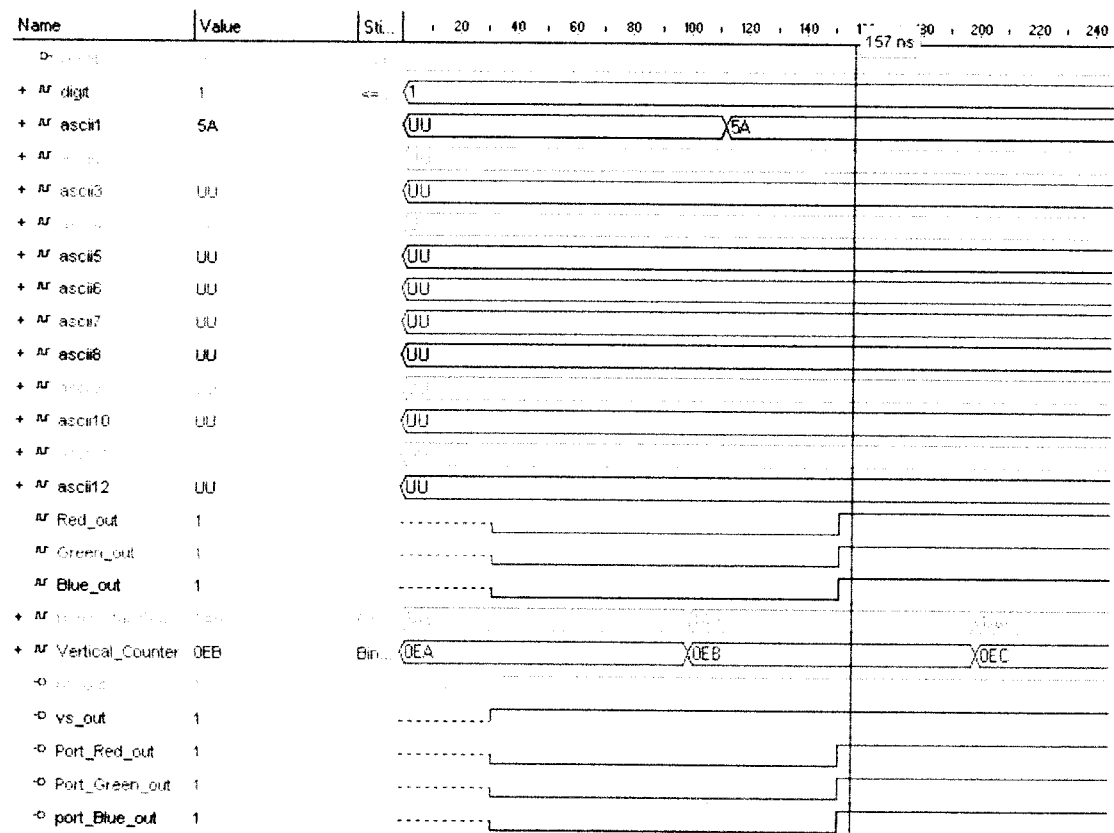


Gambar 4.2 Hasil simulasi dari *keypad*.

Dari hasil simulasi diatas dapat dilihat jika salah satu tombol ditekan maka sinyal `code_ascii` akan memiliki suatu nilai dan sinyal `trigger (f)` akan berlogika 1. Masukkan `keypad` pada simulasi diatas kita beri stimulator `hotkey`, jadi pada hasil simulasi diatas setelah 50 ns, ditekan huruf Z pada `keyboard` sehingga `keypad(25)` yang memiliki stimulator Z nilainya berubah menjadi 1. `Case keypad` pada program dibuat 27 bit agar pengalaman `pin` pada Xilinx terdeteksi 27 keypad.

4.1.3 Program Utama

Pada bagian ini dapat dilihat hasil simulasi lanjutan dari dari `clock` dan `keypad` diatas. Hasil dari simulasinya adalah sebagai berikut :



Gambar 4.3 Hasil simulasi dari output.

Gambar diatas adalah potongan dari hasil simulasi *clock* dan *keypad*. Sinyal *ascii1* mempunyai keluaran sama dengan *code_ascii* yang terlihat pada simulasi *keypad*, karena keluaran *code_ascii* yang pertama kali dikeluarkan akan digunakan sebagai keluaran sinyal *ascii1* dan seterusnya sampai *code_ascii* yang kesepuluh mengisi *ascii10*.

Setelah *ascii1* mempunyai nilai maka sinyal *Red_out*, *Green_out*, *Blue_out* akan bernilai sehingga *Port_Red_out*, *Port_Green_out*, *Port_Blue_out* akan bernilai 1 yang nantinya *Port_Red_out*, *Port_Green_out*, *Port_Blue_out* akan dihubungkan ke monitor.

Untuk simulasi ini sistemnya akan berbeda dengan sistem sebenarnya karena pada simulasi *Port_Red_out*, *Port_Green_out*, *Port_Blue_out* akan bernilai 0 jika nilai digit, *horisontal_counter* dan *vertical_counter* tidak disinkronkan. Pada simulasi diatas digit diberi nilai 1 sehingga *horisontal_counter* yang diberi stimulator *counter* diawali dengan nilai 424 dan *vertical_counter* 234 karena untuk digit 1 atau *display area* untuk karakter pertama diawali dengan nilai tersebut. Jadi untuk melakukan simulasi tiap masukan dilakukan sendiri-sendiri karena setiap masukan dan *display area* dimana masukan tersebut akan diletakkan harus diatur nilai awal *horisontal_counter* dan *vertical_counter* dari *display area* yang diinginkan.

Untuk sinyal *Hs* akan bernilai satu setelah *horisontal counter* mencapai nilai 93 yaitu dari *sync pulse* 92 ditambah 1 dan *Vs* juga akan bernilai satu setelah *vertical counter* mencapai 3 yaitu dari *sync pulse* 2 ditambah satu. *Source code* VHDL-nya terlihat sebagai berikut :

```

if (Horizontal_Counter > "0000000000" )
    and (Horizontal_Counter < "0001100001" ) -- 96+1
        then
            hs_out <= '0';
        else
            hs_out <= '1';
    end if;

if (Vertical_Counter > "0000000000" )
    and (Vertical_Counter < "0000000011" ) -- 2+1
        then
            vs_out <= '0';
        else
            vs_out <= '1';
    end if;

```

Beberapa detik setelah program di-*download* pada FPGA maka monitor akan aktif.

4.2 Pembentuk mode 640 x 480 dan pewarnaan

Dalam pembentukan mode 640 x 480 dibuat *horizontal counter* yang mempunyai panjang 640 dan *vertical counter* yang mempunyai lebar 480. Berikut adalah penggalan *source code* VHDL-nya :

```

if Clk25'event and Clk25 = '1' then

```

```

if (Horizontal_Counter >= "0010010000" ) -- 144
and (Horizontal_Counter < "1100010000" ) -- 784
and (Vertical_Counter >= "0000 100001" ) -- 33
and (Vertical_Counter < " 1000000001" ) -- 513

    then

        Red_out <= '0';
        Green_out <= '0';
        Blue_out <='0';

```

Setelah Clk25 aktif maka langkah selanjutnya menentukan *horisontal counter* sepanjang 640 yang dimulai dari 144. Angka ini didapat dari *sync pulse* 96 dan *back porch* 48 sehingga setelah *sync pulse* dan *back porch* mencapai 144 maka akan didapat *pixel* atau kolom yang aktif sampai 784 dan setelah itu terdapat *front porch* 16 jadi untuk mode 640 beserta *frame*-nya dalam satu baris mempunyai total *pixel* atau kolom sebanyak 800. Sedangkan untuk *vertical counter* dimulai dengan nilai 33, angka ini didapat dari *back porch* 31 dan *sync pulse* 2 maka akan didapat baris yang aktif sampai 513 jadi jumlah total *line* atau baris dalam 1 kolom adalah 513 ditambah *front porch* 11 adalah 524.

Setelah pembentukan mode 640 x 480 maka selanjutnya menentukan warna dari text yang nanti akan ditampilkan. Disini berlaku *active low* jadi warna putih yang berlogika R '1', G '1' B '1' maka kita balik menjadi '0'.

4.3 Display Area

Untuk menentukan *display area* dimana akan meletakkan suatu karakter atau menentukan letak digit pada monitor sebagai contoh dapat dilihat dalam penggalan *source code* VHDL berikut ini :

```
if (Horizontal_Counter >= "0110101000" ) -- 424
    and (Horizontal_Counter <= "0110101111") -- 431

    and (Vertical_Counter >= "0011101010") -- 234
    and (Vertical_Counter <= "0011101010") then -- 245
```

maka karakter akan diletakkan pada kolom antara 424 sampai 431 dan antara baris 234 sampai 245.

BAB V

PENUTUP

5.1 Kesimpulan

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

1. Untuk menampilkan suatu gambar monitor membutuhkan sinyal yaitu R, G, B, VS dan HS.
2. Data yang akan ditampilkan pada monitor berbentuk *array* yang terdiri dari kolom (*pixel*) dan Baris (*line*).
3. Gerbang logika yang mampu dilayani oleh FPGA Xilinx spartan2 Xc2s50-PQ208 hanya 50 ribu gerbang logika dasar digital dan memorinya juga terbatas maka dengan metode ini hanya dapat menampilkan beberapa digit atau beberapa karakter saja pada monitor.

5.2 Saran

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

1. Jika pengendalian dilakukan dengan FPGA hendaknya dapat membuat bahasa yang lebih baik sehingga tidak memakan memori atau memakai gerbang logika yang melebihi kapasitas dari FPGA sehingga dapat dapat menampilkan lebih banyak karakter pada monitor.

2. Dengan tersedianya *port* PS2 maka untuk kedepan dapat membuat kode VHDL sebagai *interface* antara FPGA dengan *keyboard* sehingga memasukkan datanya dapat menggunakan *keyboard* .

DAFTAR PUSTAKA

- Adinandra, Sisdarmanto, ST, 2004, Bahan Kuliah Perancangan Sistem Elektronika, Yogyakarta.
- Anonim, 2006. *Pong Game* available at <http://www.fpga4fun.com>.
- Anonim, 2005. *Course 31002 - Digital Electronics* available at <http://server.oersted.dtu.dk>
- Boult, D. Vanden, 2004. *VGA Generator for the XSA Boards* Available at <http://www.xess.com>.
- Carmack, Carmen & Jeff, Tyson, 1998 – 2007. *How Computer Monitors Work* available at <http://computer.howstuffworks.com>
- Pulman, 2005. *Digilent Pegasus Board Reference Manual* Available at <http://www.digilentinc.com>.
- Stessen, H. Jeroen, 2002. *VGA timing information* available at <http://www.ePanorama.net>.
- Xilinx inc, Ise 7 Tutorial, data sheet available at <http://www.Xilinx.com>

Lampiran 1

Listing program

```
-----  
library IEEE;  
use IEEE.STD_LOGIC_1164.ALL;  
use IEEE.STD_LOGIC_ARITH.ALL;  
use IEEE.STD_LOGIC_UNSIGNED.ALL;  
  
---- Uncomment the following library declaration if instantiating  
---- any Xilinx primitives in this code.  
--library UNISIM;  
--use UNISIM.VComponents.all;  
  
entity vga is  
    Port ( clk50_in,reset : in std_logic;  
          keypad : in std_logic_vector (26 downto 0);  
          Port_Red_out : out std_logic;  
          Port_Green_out : out std_logic;  
          port_Blue_out : out std_logic;  
          hs_out : out std_logic;  
          vs_out : out std_logic);  
  
end vga;  
  
architecture Behavioral of vga is  
    signal Clk25,f,Red_out,Green_out,Blue_out : std_logic;  
    signal code_ascii,ascii1,ascii2,ascii3,ascii4,ascii5,ascii6,ascii7,  
    ascii8,ascii9,ascii10,ascii11,ascii12 : std_logic_vector (7 downto 0);  
    signal Horizontal_Counter : std_logic_vector (9 downto 0);  
    signal Vertical_Counter : std_logic_vector (9 downto 0);  
    signal digit : std_logic_vector (3 downto 0);  
  
begin  
    Port_Red_out <= Red_out;  
    Port_Green_out <= Green_out;  
    port_Blue_out <= Blue_out;  
  
    --Generate 25Mhz Clock  
    process (clk50_in)  
    begin  
        if clk50_in'event and clk50_in='1' then  
            if (Clk25 = '0')then  
                Clk25 <= '1' after 2 ns;  
            else  
                Clk25 <= '0' after 2 ns;  
            end if;  
        end if;  
    end process;  
end process;
```



```

process (reset,keypad,code_ascii,Clk25)
begin
    if reset='1' then
        code_ascii <="00000000";
    elsif Clk25'event and Clk25 = '1' then
        case keypad is
            when "00000000000000000000000001" => code_ascii <=
"01000001";f<='1'; ----A--65
            when "00000000000000000000000010" => code_ascii <=
"01000010";f<='1'; ----B--66
            when "000000000000000000000000100" => code_ascii <=
"01000011";f<='1'; ----C--67
            when "0000000000000000000000001000" => code_ascii <=
"01000100";f<='1'; ----D--68
            when "00000000000000000000000010000" => code_ascii <=
"01000101";f<='1'; ----E--69
            when "000000000000000000000000100000" => code_ascii <=
"01000110";f<='1'; ----F--70
            when "0000000000000000000000001000000" => code_ascii <=
"01000111";f<='1'; ----G--71
            when "00000000000000000000000010000000" => code_ascii <=
"01001000";f<='1'; ----H--72
            when "000000000000000000000000100000000" => code_ascii <=
"01001001";f<='1'; ----I--73
            when "0000000000000000000000001000000000" => code_ascii <=
"01001010";f<='1'; ----J--74
            when "00000000000000000000000010000000000" => code_ascii <=
"01001011";f<='1'; ----K--75
            when "000000000000000000000000100000000000" => code_ascii <=
"01001100";f<='1'; ----L--76
            when "0000000000000000000000001000000000000" => code_ascii <=
"01001101";f<='1'; ----M--77
            when "00000000000000000000000010000000000000" => code_ascii <=
"01001110";f<='1'; ----N--78
            when "000000000000000000000000100000000000000" => code_ascii <=
"01001111";f<='1'; ----O--79
            when "00000000000010000000000000000000" => code_ascii <=
"01010000";f<='1'; ----P--80
            when "00000000000100000000000000000000" => code_ascii <=
"01010001";f<='1'; ----Q--81
            when "00000000010000000000000000000000" => code_ascii <=
"01010010";f<='1'; ----R--82
            when "00000000100000000000000000000000" => code_ascii <=
"01010011";f<='1'; ----S--83
            when "00000001000000000000000000000000" => code_ascii <=
"01010100";f<='1'; ----T--84
            when "00000010000000000000000000000000" => code_ascii <=
"01010101";f<='1'; ----U--85
            when "00000100000000000000000000000000" => code_ascii <=
"01010110";f<='1'; ----V--86
            when "00001000000000000000000000000000" => code_ascii <=
"01010111";f<='1'; ----W--87
            when "00010000000000000000000000000000" => code_ascii <=
"01011000";f<='1'; ----X--88

```

```

        when "001000000000000000000000" => code_ascii <=
"01011001";f<='1'; ----Y--89
        when "010000000000000000000000" => code_ascii <=
"01011010";f<='1'; ----Z--90
        when "100000000000000000000000" => code_ascii <=
"11111111";f<='1'; --SPACE--
        when others => code_ascii <= code_ascii; f<='0';
    end case;
end if;
end process;

```

```

process (f,digit,reset)
begin
    if reset = '1' then
        digit<="0000";
    elsif f='1' and f'event then
        if digit<11 then
            digit <= digit + 1;
        elsif digit=11 then
            digit <= digit + 0;
        else
            digit <=(others=>'0');
        end if;
    end if;
end process;

```

```

process(reset,ascii1,ascii2,ascii3,ascii4,ascii5,ascii6,Clk25)
begin
    if reset='1' then
        ascii1<="00000000";ascii2<="00000000";ascii3<="00000000";ascii4<="00000000";ascii5<
="00000000";ascii6<="00000000";ascii7<="00000000";ascii8<="00000000";ascii9<="00000000";as
cii10<="00000000";

        elsif Clk25='1' and Clk25'event then
            case digit is
                when "0001"=>ascii1<=code_ascii; ascii2<=ascii2; ascii3<=ascii3; ascii4<=ascii4;
ascii5<=ascii5; ascii6<=ascii6; ascii7<=ascii7; ascii8<=ascii8; ascii9<=ascii9; ascii10<=ascii10;
                when "0010"=>ascii1<=ascii1; ascii2<=code_ascii; ascii3<=ascii3; ascii4<=ascii4;
ascii5<=ascii5; ascii6<=ascii6; ascii7<=ascii7; ascii8<=ascii8; ascii9<=ascii9; ascii10<=ascii10;
                when "0011"=>ascii1<=ascii1; ascii2<=ascii2; ascii3<=code_ascii; ascii4<=ascii4;
ascii5<=ascii5; ascii6<=ascii6; ascii7<=ascii7; ascii8<=ascii8; ascii9<=ascii9; ascii10<=ascii10;
                when "0100"=>ascii1<=ascii1; ascii2<=ascii2; ascii3<=ascii3; ascii4<=code_ascii;
ascii5<=ascii5; ascii6<=ascii6; ascii7<=ascii7; ascii8<=ascii8; ascii9<=ascii9; ascii10<=ascii10;
                when "0101"=>ascii1<=ascii1; ascii2<=ascii2; ascii3<=ascii3; ascii4<=ascii4;
ascii5<=code_ascii; ascii6<=ascii6; ascii7<=ascii7; ascii8<=ascii8; ascii9<=ascii9; ascii10<=ascii10;
                when "0110"=>ascii1<=ascii1; ascii2<=ascii2; ascii3<=ascii3; ascii4<=ascii4;
ascii5<=ascii5; ascii6<=code_ascii; ascii7<=ascii7; ascii8<=ascii8; ascii9<=ascii9; ascii10<=ascii10;
                when "0111"=>ascii1<=ascii1; ascii2<=ascii2; ascii3<=ascii3; ascii4<=ascii4;
ascii5<=ascii5; ascii6<=ascii6; ascii7<=code_ascii; ascii8<=ascii8; ascii9<=ascii9; ascii10<=ascii10;
                when "1000"=>ascii1<=ascii1; ascii2<=ascii2; ascii3<=ascii3; ascii4<=ascii4;
ascii5<=ascii5; ascii6<=ascii6; ascii7<=ascii7; ascii8<=code_ascii; ascii9<=ascii9; ascii10<=ascii10;
                when "1001"=>ascii1<=ascii1; ascii2<=ascii2; ascii3<=ascii3; ascii4<=ascii4;
ascii5<=ascii5; ascii6<=ascii6; ascii7<=ascii7; ascii8<=ascii8; ascii9<=code_ascii; ascii10<=ascii10;
            end case;
        end if;
    end process;

```

```

        when "1010"=>ascii1<=ascii1; ascii2<=ascii2; ascii3<=ascii3; ascii4<=ascii4;
        ascii5<=ascii5; ascii6<=ascii6; ascii7<=ascii7; ascii8<=ascii8; ascii9<=ascii9; ascii10<=code_ascii;
        when others =>ascii1<=ascii1; ascii2<=ascii2; ascii3<=ascii3; ascii4<=ascii4;
        ascii5<=ascii5; ascii6<=ascii6; ascii7<=ascii7; ascii8<=ascii8; ascii9<=ascii9;
        ascii10<=ascii10;
    end case;
end if;
end process;
process (Clk25)

```

```

TYPE huruf is ARRAY(0 to 11, 0 to 7) OF std_logic;
CONSTANT char_A : huruf :=
    (( '0','0','0','1','0','0','0','0'),
      ('0','0','1','1','1','0','0','0'),
      ('0','1','1','0','1','1','0','0'),
      ('1','1','0','0','0','1','1','0'),
      ('1','1','0','0','0','1','1','0'),
      ('1','1','1','1','1','1','1','0'),
      ('1','1','0','0','0','1','1','0'),
      ('1','1','0','0','0','1','1','0'),
      ('1','1','0','0','0','1','1','0'),
      ('1','1','0','0','0','1','1','0'),
      ('1','1','0','0','0','1','1','0'),
      ('0','0','0','0','0','0','0','0'),
      ('0','0','0','0','0','0','0','0'));

```

```

CONSTANT char_B : huruf :=
    (('1','1','1','1','1','1','0','0'),
      ('0','1','1','0','0','1','1','0'),
      ('0','1','1','0','0','1','1','0'),
      ('0','1','1','0','0','1','1','0'),
      ('0','1','1','1','1','1','0','0'),
      ('0','1','1','0','0','1','1','0'),
      ('0','1','1','0','0','1','1','0'),
      ('0','1','1','0','0','1','1','0'),
      ('0','1','1','0','0','1','1','0'),
      ('0','1','1','0','0','1','1','0'),
      ('1','1','1','1','1','1','0','0'),
      ('0','0','0','0','0','0','0','0'),
      ('0','0','0','0','0','0','0','0'));

```

```

CONSTANT char_C : huruf :=
    (('0','0','1','1','1','1','0','0'),
      ('0','1','1','0','0','1','1','0'),
      ('1','1','0','0','0','0','0','0'),
      ('1','1','0','0','0','0','0','0'),
      ('1','1','0','0','0','0','0','0'),
      ('1','1','0','0','0','0','0','0'),
      ('1','1','0','0','0','0','0','0'),
      ('1','1','0','0','0','0','0','0'),
      ('1','1','0','0','0','0','0','0'),
      ('0','1','1','0','0','1','1','0'),
      ('0','0','1','1','1','1','0','0'),
      ('0','0','0','0','0','0','0','0'),
      ('0','0','0','0','0','0','0','0'));

```

```

CONSTANT char_D : huruf :=
    (('1','1','1','1','1','1','0','0'),
      ('0','1','1','0','1','1','0','0'),
      ('0','1','1','0','0','1','1','0'),
      ('0','1','1','0','0','1','1','0'),
      ('0','1','1','0','0','1','1','0'));

```

```
( '0','1','1','0','0','1','1','0'),  
( '0','1','1','0','0','1','1','0'),  
( '0','1','1','0','0','1','1','0'),  
( '0','1','1','0','1','1','0','0'),  
( '1','1','1','1','1','0','0','0'),  
( '0','0','0','0','0','0','0','0'),  
( '0','0','0','0','0','0','0','0'));
```

```
CONSTANT char_E : huruf := (( '1','1','1','1','1','1','1','0'),  
( '0','1','1','0','0','1','1','0'),  
( '0','1','1','0','0','0','1','0'),  
( '0','1','1','0','1','0','0','0'),  
( '0','1','1','1','1','0','0','0'),  
( '0','1','1','0','1','0','0','0'),  
( '0','1','1','0','0','0','0','0'),  
( '0','1','1','0','0','0','1','0'),  
( '0','1','1','0','0','1','1','0'),  
( '1','1','1','1','1','1','1','0'),  
( '0','0','0','0','0','0','0','0'),  
( '0','0','0','0','0','0','0','0'));
```

```
CONSTANT char_F : huruf := (( '1','1','1','1','1','1','1','0'),  
( '0','1','1','0','0','1','1','0'),  
( '0','1','1','0','0','0','1','0'),  
( '0','1','1','0','1','0','0','0'),  
( '0','1','1','1','1','0','0','0'),  
( '0','1','1','0','1','0','0','0'),  
( '0','1','1','0','0','0','0','0'),  
( '0','1','1','0','0','0','0','0'),  
( '0','1','1','0','0','0','0','0'),  
( '1','1','1','1','0','0','0','0'),  
( '0','0','0','0','0','0','0','0'),  
( '0','0','0','0','0','0','0','0'));
```

```
CONSTANT char_G : huruf := (( '0','0','1','1','1','1','0','0'),  
( '0','1','1','0','0','1','1','0'),  
( '1','1','0','0','0','0','1','0'),  
( '1','1','0','0','0','0','0','0'),  
( '1','1','0','0','0','0','0','0'),  
( '1','1','0','1','1','1','1','0'),  
( '1','1','0','0','0','1','1','0'),  
( '1','1','0','0','0','1','1','0'),  
( '0','1','1','0','0','1','1','0'),  
( '0','0','1','1','1','0','1','0'),  
( '0','0','0','0','0','0','0','0'),  
( '0','0','0','0','0','0','0','0'));
```

```
CONSTANT char_H : huruf := (( '1','1','0','0','0','1','1','0'),  
( '1','1','0','0','0','1','1','0'),  
( '1','1','0','0','0','1','1','0'),  
( '1','1','0','0','0','1','1','0'),  
( '1','1','1','1','1','1','1','0'),  
( '1','1','0','0','0','1','1','0'),  
( '1','1','0','0','0','1','1','0'),
```

('1','1','0','0','0','1','1','0'),
('1','1','0','0','0','1','1','0'),
('1','1','0','0','0','1','1','0'),
('0','0','0','0','0','0','0','0'),
('0','0','0','0','0','0','0','0'));

CONSTANT char_I : huruf : = (('0','0','1','1','1','1','0','0'),
('0','0','0','1','1','0','0','0'),
('0','0','0','1','1','0','0','0'),
('0','0','0','1','1','0','0','0'),
('0','0','0','1','1','0','0','0'),
('0','0','0','1','1','0','0','0'),
('0','0','0','1','1','0','0','0'),
('0','0','0','1','1','0','0','0'),
('0','0','0','1','1','0','0','0'),
('0','0','1','1','1','1','0','0'),
('0','0','0','0','0','0','0','0'),
('0','0','0','0','0','0','0','0'));

CONSTANT char_J : huruf : = (('0','0','0','1','1','1','1','0'),
('0','0','0','0','1','1','0','0'),
('0','0','0','0','1','1','0','0'),
('0','0','0','0','1','1','0','0'),
('0','0','0','0','1','1','0','0'),
('0','0','0','0','1','1','0','0'),
('1','1','0','0','1','1','0','0'),
('1','1','0','0','1','1','0','0'),
('1','1','0','0','1','1','0','0'),
('0','1','1','1','1','0','0','0'),
('0','0','0','0','0','0','0','0'),
('0','0','0','0','0','0','0','0'));

CONSTANT char_K : huruf : = (('1','1','1','0','0','1','1','0'),
('0','1','1','0','0','1','1','0'),
('0','1','1','0','0','1','1','0'),
('0','1','1','0','1','1','0','0'),
('0','1','1','1','1','0','0','0'),
('0','1','1','1','1','0','0','0'),
('0','1','1','0','1','1','0','0'),
('0','1','1','0','0','1','1','0'),
('0','1','1','0','0','1','1','0'),
('1','1','1','0','0','1','1','0'),
('0','0','0','0','0','0','0','0'),
('0','0','0','0','0','0','0','0'));

CONSTANT char_L : huruf : = (('1','1','1','1','0','0','0','0'),
('0','1','1','0','0','0','0','0'),
('0','1','1','0','0','0','0','0'),
('0','1','1','0','0','0','0','0'),
('0','1','1','0','0','0','0','0'),
('0','1','1','0','0','0','0','0'),
('0','1','1','0','0','0','0','0'),
('0','1','1','0','0','0','1','0'),
('0','1','1','0','0','1','1','0'),
('1','1','1','1','1','1','1','0'));

('0','0','0','0','0','0','0','0'),
('0','0','0','0','0','0','0','0'));

CONSTANT char_M : huruf := (('1','1','0','0','0','0','1','1'),
('1','1','1','0','0','1','1','1'),
('1','1','1','1','1','1','1','1'),
('1','1','1','1','1','1','1','1'),
('1','1','0','1','1','0','1','1'),
('1','1','0','0','0','0','1','1'),
('1','1','0','0','0','0','1','1'),
('1','1','0','0','0','0','1','1'),
('1','1','0','0','0','0','1','1'),
('1','1','0','0','0','0','1','1'),
('0','0','0','0','0','0','0','0'),
('0','0','0','0','0','0','0','0'));

CONSTANT char_N : huruf := (('1','1','0','0','0','1','1','0'),
('1','1','1','0','0','1','1','0'),
('1','1','1','1','0','1','1','0'),
('1','1','1','1','1','1','1','0'),
('1','1','0','1','1','1','1','0'),
('1','1','0','0','1','1','1','0'),
('1','1','0','0','0','1','1','0'),
('1','1','0','0','0','1','1','0'),
('1','1','0','0','0','1','1','0'),
('1','1','0','0','0','1','1','0'),
('1','1','0','0','0','1','1','0'),
('0','0','0','0','0','0','0','0'),
('0','0','0','0','0','0','0','0'));

CONSTANT char_O : huruf := (('0','1','1','1','1','1','0','0'),
('1','1','0','0','0','1','1','0'),
('1','1','0','0','0','1','1','0'),
('1','1','0','0','0','1','1','0'),
('1','1','0','0','0','1','1','0'),
('1','1','0','0','0','1','1','0'),
('1','1','0','0','0','1','1','0'),
('1','1','0','0','0','1','1','0'),
('1','1','0','0','0','1','1','0'),
('0','1','1','1','1','1','0','0'),
('0','0','0','0','0','0','0','0'),
('0','0','0','0','0','0','0','0'));

CONSTANT char_P : huruf := (('1','1','1','1','1','1','0','0'),
('0','1','1','0','0','1','1','0'),
('0','1','1','0','0','1','1','0'),
('0','1','1','0','0','1','1','0'),
('0','1','1','1','1','1','0','0'),
('0','1','1','0','0','0','0','0'),
('0','1','1','0','0','0','0','0'),
('0','1','1','0','0','0','0','0'),
('0','1','1','0','0','0','0','0'),
('0','1','1','0','0','0','0','0'),
('1','1','1','1','0','0','0','0'),
('0','0','0','0','0','0','0','0'),
('0','0','0','0','0','0','0','0'));

```

CONSTANT char_Q : huruf :=
(( '0','1','1','1','1','1','0','0'),
( '1','1','0','0','0','1','1','0'),
( '1','1','0','0','0','1','1','0'),
( '1','1','0','0','0','1','1','0'),
( '1','1','0','0','0','1','1','0'),
( '1','1','0','0','0','1','1','0'),
( '1','1','0','0','0','1','1','0'),
( '1','1','0','1','0','1','1','0'),
( '1','1','0','1','1','1','1','0'),
( '0','1','1','1','1','1','0','0'),
( '0','0','0','0','1','1','0','0'),
( '0','0','0','0','1','1','1','0'));

```

```

CONSTANT char_R : huruf :=
(( '1','1','1','1','1','1','0','0'),
( '0','1','1','0','0','1','1','0'),
( '0','1','1','0','0','1','1','0'),
( '0','1','1','0','0','1','1','0'),
( '0','1','1','1','1','1','0','0'),
( '0','1','1','0','1','1','0','0'),
( '0','1','1','0','0','1','1','0'),
( '0','1','1','0','0','1','1','0'),
( '0','1','1','0','0','1','1','0'),
( '0','1','1','0','0','1','1','0'),
( '1','1','1','0','0','1','1','0'),
( '0','0','0','0','0','0','0','0'),
( '0','0','0','0','0','0','0','0'));

```

```

CONSTANT char_S : huruf :=
(( '0','1','1','1','1','1','0','0'),
( '1','1','0','0','0','1','1','0'),
( '1','1','0','0','0','1','1','0'),
( '0','1','1','0','0','0','0','0'),
( '0','0','1','1','1','0','0','0'),
( '0','0','0','0','1','1','0','0'),
( '0','0','0','0','0','1','1','0'),
( '1','1','0','0','0','1','1','0'),
( '1','1','0','0','0','1','1','0'),
( '0','1','1','1','1','1','0','0'),
( '0','0','0','0','0','0','0','0'),
( '0','0','0','0','0','0','0','0'));

```

```

CONSTANT char_T : huruf :=
(( '1','1','1','1','1','1','1','1'),
( '1','1','0','1','1','0','1','1'),
( '1','0','0','1','1','0','0','1'),
( '0','0','0','1','1','0','0','0'),
( '0','0','0','1','1','0','0','0'),
( '0','0','0','1','1','0','0','0'),
( '0','0','0','1','1','0','0','0'),
( '0','0','0','1','1','0','0','0'),
( '0','0','0','1','1','0','0','0'),
( '0','0','1','1','1','1','0','0'),
( '0','0','0','0','0','0','0','0'),
( '0','0','0','0','0','0','0','0'));

```

```

CONSTANT char_U : huruf :=
(( '1','1','0','0','0','1','1','0'),
( '1','1','0','0','0','1','1','0'),

```

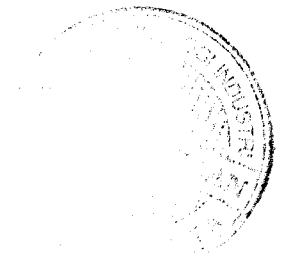
('1','1','0','0','0','1','1','0'),
('1','1','0','0','0','1','1','0'),
('1','1','0','0','0','1','1','0'),
('1','1','0','0','0','1','1','0'),
('1','1','0','0','0','1','1','0'),
('1','1','0','0','0','1','1','0'),
('1','1','0','0','0','1','1','0'),
('0','1','1','1','1','0','0'),
('0','0','0','0','0','0','0'),
('0','0','0','0','0','0','0'));

CONSTANT char_V : huruf : = (('1','1','0','0','0','1','1','0'),
('1','1','0','0','0','1','1','0'),
('1','1','0','0','0','1','1','0'),
('1','1','0','0','0','1','1','0'),
('1','1','0','0','0','1','1','0'),
('1','1','0','0','0','1','1','0'),
('1','1','0','0','0','1','1','0'),
('1','1','0','0','0','1','1','0'),
('0','1','1','0','1','1','0','0'),
('0','0','0','1','0','0','0','0'),
('0','0','0','0','0','0','0','0'),
('0','0','0','0','0','0','0','0'));

CONSTANT char_W : huruf : = (('1','1','0','0','0','0','1','1'),
('1','1','0','0','0','0','1','1'),
('1','1','0','0','0','0','1','1'),
('1','1','0','0','0','0','1','1'),
('1','1','0','0','0','0','1','1'),
('1','1','0','1','1','0','1','1'),
('1','1','0','1','1','0','1','1'),
('1','1','1','1','1','1','1'),
('0','1','1','0','0','1','1','0'),
('0','1','1','0','0','1','1','0'),
('0','0','0','0','0','0','0','0'),
('0','0','0','0','0','0','0','0'));

CONSTANT char_X : huruf : = (('1','1','0','0','0','0','1','1'),
('1','1','0','0','0','0','1','1'),
('0','1','1','0','0','1','1','0'),
('0','0','1','1','1','1','0','0'),
('0','0','0','1','1','0','0','0'),
('0','0','0','1','1','0','0','0'),
('0','0','1','1','1','1','0','0'),
('0','1','1','0','0','1','1','0'),
('1','1','0','0','0','0','1','1'),
('1','1','0','0','0','0','1','1'),
('0','0','0','0','0','0','0','0'),
('0','0','0','0','0','0','0','0'));

CONSTANT char_Y : huruf : = (('1','1','0','0','0','0','1','1'),
('1','1','0','0','0','0','1','1'),
('1','1','0','0','0','0','1','1'),
('0','1','1','1','1','1','0'),
('0','0','1','1','1','1','0','0'),




```

( '0','0','0','1','1','0','0','0'),
( '0','0','0','1','1','0','0','0'),
( '0','0','0','1','1','0','0','0'),
( '0','0','0','1','1','0','0','0'),
( '0','0','1','1','1','1','0','0'),
( '0','0','0','0','0','0','0','0'),
( '0','0','0','0','0','0','0','0'));

CONSTANT char_Z : huruf : =
(( '1','1','1','1','1','1','1','1'),
( '1','1','0','0','0','0','1','1'),
( '1','0','0','0','0','1','1','0'),
( '0','0','0','0','1','1','0','0'),
( '0','0','0','1','1','0','0','0'),
( '0','0','1','1','0','0','0','0'),
( '0','1','1','0','0','0','0','0'),
( '1','1','1','0','0','0','0','1'),
( '1','1','0','0','0','0','1','1'),
( '1','1','1','1','1','1','1','1'),
( '0','0','0','0','0','0','0','0'),
( '0','0','0','0','0','0','0','0'));

CONSTANT char_Reset : huruf : =
(( '0','0','0','0','0','0','0','0'),
( '0','0','0','0','0','0','0','0'),
( '0','0','0','0','0','0','0','0'),
( '0','0','0','0','0','0','0','0'),
( '0','0','0','0','0','0','0','0'),
( '0','0','0','0','0','0','0','0'),
( '0','0','0','0','0','0','0','0'),
( '0','0','0','0','0','0','0','0'),
( '0','0','0','0','0','0','0','0'),
( '0','0','0','0','0','0','0','0'),
( '0','0','0','0','0','0','0','0'),
( '0','0','0','0','0','0','0','0'),
( '0','0','0','0','0','0','0','0'),
( '0','0','0','0','0','0','0','0'));

variable Line:integer:=0;
variable Pixel:integer:=0;

begin

if Clk25'event and Clk25 = '1' then
  if (Horizontal_Counter >= "0010010000" ) -- 144
  and (Horizontal_Counter < "1100010000" ) -- 784
  and (Vertical_Counter >= "0000100001" ) -- 33
  and (Vertical_Counter < "1000000001" ) -- 513
  then
    Red_out <= '0'; ---- Buat text putih
    Green_out <= '0'; ---- Buat text putih
    Blue_out <= '0'; ---- Buat text putih
  if (Horizontal_Counter >= "0110101000" )--424
  and (Horizontal_Counter <= "0110101111")-- 431
  and (Vertical_Counter >= "0011101010" ) --234
  and (Vertical_Counter <= "0011110101" ) then -- 245
    if (Pixel <= 7) and (line <=11) then
      if (ascii1 <= "00000000") then
        Red_out <= char_Reset(Line, Pixel);

```

```

Green_out <= char_Reset(Line, Pixel);
Blue_out <= char_Reset(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii1 <= "01000001")then
Red_out <= char_A(Line, Pixel);
Green_out <= char_A(Line, Pixel);
Blue_out <= char_A(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii1 <= "01000010") then
Red_out <= char_B(Line, Pixel);
Green_out <= char_B(Line, Pixel);
Blue_out <= char_B(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii1 <= "01000011") then
Red_out <= char_C(Line, Pixel);
Green_out <= char_C(Line, Pixel);
Blue_out <= char_C(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii1 <= "01000100") then
Red_out <= char_D(Line, Pixel);
Green_out <= char_D(Line, Pixel);
Blue_out <= char_D(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii1 <= "01000101") then
Red_out <= char_E(Line, Pixel);
Green_out <= char_E(Line, Pixel);
Blue_out <= char_E(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii1 <= "01000110") then
Red_out <= char_F(Line, Pixel);
Green_out <= char_F(Line, Pixel);
Blue_out <= char_F(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii1 <= "01000111") then
Red_out <= char_G(Line, Pixel);
Green_out <= char_G(Line, Pixel);
Blue_out <= char_G(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii1 <= "01001000") then
Red_out <= char_H(Line, Pixel);
Green_out <= char_H(Line, Pixel);
Blue_out <= char_H(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii1 <= "01001001") then
Red_out <= char_I(Line, Pixel);
Green_out <= char_I(Line, Pixel);
Blue_out <= char_I(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii1 <= "01001010") then
Red_out <= char_J(Line, Pixel);
Green_out <= char_J(Line, Pixel);
Blue_out <= char_J(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii1 <= "01001011") then
Red_out <= char_K(Line, Pixel);

```

```

        Green_out <= char_K(Line, Pixel);
        Blue_out <= char_K(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii1 <= "01001100") then
        Red_out <= char_L(Line, Pixel);
        Green_out <= char_L(Line, Pixel);
        Blue_out <= char_L(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii1 <= "01001101") then
        Red_out <= char_M(Line, Pixel);
        Green_out <= char_M(Line, Pixel);
        Blue_out <= char_M(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii1 <= "01001110") then
        Red_out <= char_N(Line, Pixel);
        Green_out <= char_N(Line, Pixel);
        Blue_out <= char_N(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii1 <= "01001111") then
        Red_out <= char_O(Line, Pixel);
        Green_out <= char_O(Line, Pixel);
        Blue_out <= char_O(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii1 <= "01010000") then
        Red_out <= char_P(Line, Pixel);
        Green_out <= char_P(Line, Pixel);
        Blue_out <= char_P(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii1 <= "01010001") then
        Red_out <= char_Q(Line, Pixel);
        Green_out <= char_Q(Line, Pixel);
        Blue_out <= char_Q(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii1 <= "01010010") then
        Red_out <= char_R(Line, Pixel);
        Green_out <= char_R(Line, Pixel);
        Blue_out <= char_R(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii1 <= "01010011") then
        Red_out <= char_S(Line, Pixel);
        Green_out <= char_S(Line, Pixel);
        Blue_out <= char_S(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii1 <= "01010100") then
        Red_out <= char_T(Line, Pixel);
        Green_out <= char_T(Line, Pixel);
        Blue_out <= char_T(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii1 <= "01010101") then
        Red_out <= char_U(Line, Pixel);
        Green_out <= char_U(Line, Pixel);
        Blue_out <= char_U(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii1 <= "01010110") then
        Red_out <= char_V(Line, Pixel);

```

```

        Green_out <= char_V(Line, Pixel);
        Blue_out <= char_V(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii1 <= "01010111") then
        Red_out <= char_W(Line, Pixel);
        Green_out <= char_W(Line, Pixel);
        Blue_out <= char_W(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii1 <= "01011000") then
        Red_out <= char_X(Line, Pixel);
        Green_out <= char_X(Line, Pixel);
        Blue_out <= char_X(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii1 <= "01011001") then
        Red_out <= char_Y(Line, Pixel);
        Green_out <= char_Y(Line, Pixel);
        Blue_out <= char_Y(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii1 <= "01011010") then
        Red_out <= char_Z(Line, Pixel);
        Green_out <= char_Z(Line, Pixel);
        Blue_out <= char_Z(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii1 <= "11111111") then
        Red_out <= char_Reset(Line, Pixel);
        Green_out <= char_Reset(Line, Pixel);
        Blue_out <= char_Reset(Line, Pixel);
        Pixel:= Pixel+1;
    end if;
end if;
----digit 2
elsif (Horizontal_Counter >= "0110110000")--432
and (Horizontal_Counter <= "0110110111")-- 439
and (Vertical_Counter >= "0011101010") --234
and (Vertical_Counter <= "0011110101") then -- 245
    if (Pixel <= 15) and (line <=11) then
        if (ascii2 <= "00000000") then
            Red_out <= char_Reset(Line, Pixel);
            Green_out <= char_Reset(Line, Pixel);
            Blue_out <= char_Reset(Line, Pixel);
            Pixel:= Pixel+1;
        elsif (ascii2 <= "01000001")then
            Red_out <= char_A(Line, Pixel);
            Green_out <= char_A(Line, Pixel);
            Blue_out <= char_A(Line, Pixel);
            Pixel:= Pixel+1;
        elsif (ascii2 <= "01000010") then
            Red_out <= char_B(Line, Pixel);
            Green_out <= char_B(Line, Pixel);
            Blue_out <= char_B(Line, Pixel);
            Pixel:= Pixel+1;
        elsif (ascii2 <= "01000011") then
            Red_out <= char_C(Line, Pixel);
            Green_out <= char_C(Line, Pixel);
            Blue_out <= char_C(Line, Pixel);
        end if;
    end if;
end if;

```

```

Pixel:= Pixel+1;
elsif (ascii2 <= "01000100") then
  Red_out <= char_D(Line, Pixel);
  Green_out <= char_D(Line, Pixel);
  Blue_out <= char_D(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii2 <= "01000101") then
  Red_out <= char_E(Line, Pixel);
  Green_out <= char_E(Line, Pixel);
  Blue_out <= char_E(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii2 <= "01000110") then
  Red_out <= char_F(Line, Pixel);
  Green_out <= char_F(Line, Pixel);
  Blue_out <= char_F(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii2 <= "01000111") then
  Red_out <= char_G(Line, Pixel);
  Green_out <= char_G(Line, Pixel);
  Blue_out <= char_G(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii2 <= "01001000") then
  Red_out <= char_H(Line, Pixel);
  Green_out <= char_H(Line, Pixel);
  Blue_out <= char_H(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii2 <= "01001001") then
  Red_out <= char_I(Line, Pixel);
  Green_out <= char_I(Line, Pixel);
  Blue_out <= char_I(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii2 <= "01001010") then
  Red_out <= char_J(Line, Pixel);
  Green_out <= char_J(Line, Pixel);
  Blue_out <= char_J(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii2 <= "01001011") then
  Red_out <= char_K(Line, Pixel);
  Green_out <= char_K(Line, Pixel);
  Blue_out <= char_K(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii2 <= "01001100") then
  Red_out <= char_L(Line, Pixel);
  Green_out <= char_L(Line, Pixel);
  Blue_out <= char_L(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii2 <= "01001101") then
  Red_out <= char_M(Line, Pixel);
  Green_out <= char_M(Line, Pixel);
  Blue_out <= char_M(Line, Pixel);
  Pixel:= Pixel+1;
elsif(ascii2 <= "01001110") then
  Red_out <= char_N(Line, Pixel);
  Green_out <= char_N(Line, Pixel);
  Blue_out <= char_N(Line, Pixel);

```

```

Pixel:= Pixel+1;
elsif(ascii2 <= "01001111") then
    Red_out <= char_O(Line, Pixel);
    Green_out <= char_O(Line, Pixel);
    Blue_out <= char_O(Line, Pixel);
    Pixel:= Pixel+1;
elsif(ascii2 <= "01010000") then
    Red_out <= char_P(Line, Pixel);
    Green_out <= char_P(Line, Pixel);
    Blue_out <= char_P(Line, Pixel);
    Pixel:= Pixel+1;
elsif (ascii2 <= "01010001") then
    Red_out <= char_Q(Line, Pixel);
    Green_out <= char_Q(Line, Pixel);
    Blue_out <= char_Q(Line, Pixel);
    Pixel:= Pixel+1;
elsif (ascii2 <= "01010010") then
    Red_out <= char_R(Line, Pixel);
    Green_out <= char_R(Line, Pixel);
    Blue_out <= char_R(Line, Pixel);
    Pixel:= Pixel+1;
elsif (ascii2 <= "01010011") then
    Red_out <= char_S(Line, Pixel);
    Green_out <= char_S(Line, Pixel);
    Blue_out <= char_S(Line, Pixel);
    Pixel:= Pixel+1;
elsif (ascii2 <= "01010100") then
    Red_out <= char_T(Line, Pixel);
    Green_out <= char_T(Line, Pixel);
    Blue_out <= char_T(Line, Pixel);
    Pixel:= Pixel+1;
elsif (ascii2 <= "01010101") then
    Red_out <= char_U(Line, Pixel);
    Green_out <= char_U(Line, Pixel);
    Blue_out <= char_U(Line, Pixel);
    Pixel:= Pixel+1;
elsif (ascii2 <= "01010110") then
    Red_out <= char_V(Line, Pixel);
    Green_out <= char_V(Line, Pixel);
    Blue_out <= char_V(Line, Pixel);
    Pixel:= Pixel+1;
elsif (ascii2 <= "01010111") then
    Red_out <= char_W(Line, Pixel);
    Green_out <= char_W(Line, Pixel);
    Blue_out <= char_W(Line, Pixel);
    Pixel:= Pixel+1;
elsif (ascii2 <= "01011000") then
    Red_out <= char_X(Line, Pixel);
    Green_out <= char_X(Line, Pixel);
    Blue_out <= char_X(Line, Pixel);
    Pixel:= Pixel+1;
elsif (ascii2 <= "01011001") then
    Red_out <= char_Y(Line, Pixel);
    Green_out <= char_Y(Line, Pixel);
    Blue_out <= char_Y(Line, Pixel);

```

```

        Pixel:= Pixel+1;
    elsif (ascii2 <= "01011010") then
        Red_out <= char_Z(Line, Pixel);
        Green_out <= char_Z(Line, Pixel);
        Blue_out <= char_Z(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii2 <= "11111111") then
        Red_out <= char_Reset(Line, Pixel);
        Green_out <= char_Reset(Line, Pixel);
        Blue_out <= char_Reset(Line, Pixel);
        Pixel:= Pixel+1;
    end if;
end if;
----digit3
elsif (Horizontal_Counter >= "0110111000" )--440
and (Horizontal_Counter <= "0110111111")--      447
and (Vertical_Counter >= "0011101010") --234
and (Vertical_Counter <= "0011110101") then -- 245
    if (Pixel <= 23) and (line <=11) then
        if (ascii3 <= "00000000") then
            Red_out <= char_Reset(Line, Pixel);
            Green_out <= char_Reset(Line, Pixel);
            Blue_out <= char_Reset(Line, Pixel);
            Pixel:= Pixel+1;
        elsif (ascii3 <= "01000001")then
            Red_out <= char_A(Line, Pixel);
            Green_out <= char_A(Line, Pixel);
            Blue_out <= char_A(Line, Pixel);
            Pixel:= Pixel+1;
        elsif (ascii3 <= "01000010") then
            Red_out <= char_B(Line, Pixel);
            Green_out <= char_B(Line, Pixel);
            Blue_out <= char_B(Line, Pixel);
            Pixel:= Pixel+1;
        elsif (ascii3 <= "01000011") then
            Red_out <= char_C(Line, Pixel);
            Green_out <= char_C(Line, Pixel);
            Blue_out <= char_C(Line, Pixel);
            Pixel:= Pixel+1;
        elsif (ascii3 <= "01000100") then
            Red_out <= char_D(Line, Pixel);
            Green_out <= char_D(Line, Pixel);
            Blue_out <= char_D(Line, Pixel);
            Pixel:= Pixel+1;
        elsif(ascii3 <= "01000101") then
            Red_out <= char_E(Line, Pixel);
            Green_out <= char_E(Line, Pixel);
            Blue_out <= char_E(Line, Pixel);
            Pixel:= Pixel+1;
        elsif(ascii3 <= "01000110") then
            Red_out <= char_F(Line, Pixel);
            Green_out <= char_F(Line, Pixel);
            Blue_out <= char_F(Line, Pixel);
            Pixel:= Pixel+1;
        elsif
            (ascii3 <= "01000111") then

```

```

Red_out <= char_G(Line, Pixel);
Green_out <= char_G(Line, Pixel);
Blue_out <= char_G(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii3 <= "01001000") then
Red_out <= char_H(Line, Pixel);
Green_out <= char_H(Line, Pixel);
Blue_out <= char_H(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii3 <= "01001001") then
Red_out <= char_I(Line, Pixel);
Green_out <= char_I(Line, Pixel);
Blue_out <= char_I(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii3 <= "01001010") then
Red_out <= char_J(Line, Pixel);
Green_out <= char_J(Line, Pixel);
Blue_out <= char_J(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii3 <= "01001011") then
Red_out <= char_K(Line, Pixel);
Green_out <= char_K(Line, Pixel);
Blue_out <= char_K(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii3 <= "01001100") then
Red_out <= char_L(Line, Pixel);
Green_out <= char_L(Line, Pixel);
Blue_out <= char_L(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii3 <= "01001101") then
Red_out <= char_M(Line, Pixel);
Green_out <= char_M(Line, Pixel);
Blue_out <= char_M(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii3 <= "01001110") then
Red_out <= char_N(Line, Pixel);
Green_out <= char_N(Line, Pixel);
Blue_out <= char_N(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii3 <= "01001111") then
Red_out <= char_O(Line, Pixel);
Green_out <= char_O(Line, Pixel);
Blue_out <= char_O(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii3 <= "01010000") then
Red_out <= char_P(Line, Pixel);
Green_out <= char_P(Line, Pixel);
Blue_out <= char_P(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii3 <= "01010001") then
Red_out <= char_Q(Line, Pixel);
Green_out <= char_Q(Line, Pixel);
Blue_out <= char_Q(Line, Pixel);
Pixel:= Pixel+1;

```



```

elseif (ascii3 <= "01010010") then
    Red_out <= char_R(Line, Pixel);
    Green_out <= char_R(Line, Pixel);
    Blue_out <= char_R(Line, Pixel);
    Pixel:= Pixel+1;
elseif (ascii3 <= "01010011") then
    Red_out <= char_S(Line, Pixel);
    Green_out <= char_S(Line, Pixel);
    Blue_out <= char_S(Line, Pixel);
    Pixel:= Pixel+1;
elseif (ascii3 <= "01010100") then
    Red_out <= char_T(Line, Pixel);
    Green_out <= char_T(Line, Pixel);
    Blue_out <= char_T(Line, Pixel);
    Pixel:= Pixel+1;
elseif (ascii3 <= "01010101") then
    Red_out <= char_U(Line, Pixel);
    Green_out <= char_U(Line, Pixel);
    Blue_out <= char_U(Line, Pixel);
    Pixel:= Pixel+1;
elseif (ascii3 <= "01010110") then
    Red_out <= char_V(Line, Pixel);
    Green_out <= char_V(Line, Pixel);
    Blue_out <= char_V(Line, Pixel);
    Pixel:= Pixel+1;
elseif (ascii3 <= "01010111") then
    Red_out <= char_W(Line, Pixel);
    Green_out <= char_W(Line, Pixel);
    Blue_out <= char_W(Line, Pixel);
    Pixel:= Pixel+1;
elseif (ascii3 <= "01011000") then
    Red_out <= char_X(Line, Pixel);
    Green_out <= char_X(Line, Pixel);
    Blue_out <= char_X(Line, Pixel);
    Pixel:= Pixel+1;
elseif (ascii3 <= "01011001") then
    Red_out <= char_Y(Line, Pixel);
    Green_out <= char_Y(Line, Pixel);
    Blue_out <= char_Y(Line, Pixel);
    Pixel:= Pixel+1;
elseif (ascii3 <= "01011010") then
    Red_out <= char_Z(Line, Pixel);
    Green_out <= char_Z(Line, Pixel);
    Blue_out <= char_Z(Line, Pixel);
    Pixel:= Pixel+1;
elseif (ascii3 <= "11111111") then
    Red_out <= char_Reset(Line, Pixel);
    Green_out <= char_Reset(Line, Pixel);
    Blue_out <= char_Reset(Line, Pixel);
    Pixel:= Pixel+1;
end if;
end if;
----digit 4
elseif (Horizontal_Counter >= "0111000000")--448
and (Horizontal_Counter <= "0111000111")--

```

```

and (Vertical_Counter >= "0011101010") --234
and (Vertical_Counter <= "0011110101") then -- 245
  if (Pixel <= 31) and (line <=11) then
    if (ascii4 <= "00000000") then
      Red_out <= char_Reset(Line, Pixel);
      Green_out <= char_Reset(Line, Pixel);
      Blue_out <= char_Reset(Line, Pixel);
      Pixel:= Pixel+1;
    elsif (ascii4 <= "01000001")then
      Red_out <= char_A(Line, Pixel);
      Green_out <= char_A(Line, Pixel);
      Blue_out <= char_A(Line, Pixel);
      Pixel:= Pixel+1;
    elsif (ascii4 <= "01000010") then
      Red_out <= char_B(Line, Pixel);
      Green_out <= char_B(Line, Pixel);
      Blue_out <= char_B(Line, Pixel);
      Pixel:= Pixel+1;
    elsif (ascii4 <= "01000011") then
      Red_out <= char_C(Line, Pixel);
      Green_out <= char_C(Line, Pixel);
      Blue_out <= char_C(Line, Pixel);
      Pixel:= Pixel+1;
    elsif (ascii4 <= "01000100") then
      Red_out <= char_D(Line, Pixel);
      Green_out <= char_D(Line, Pixel);
      Blue_out <= char_D(Line, Pixel);
      Pixel:= Pixel+1;
    elsif (ascii4 <= "01000101") then
      Red_out <= char_E(Line, Pixel);
      Green_out <= char_E(Line, Pixel);
      Blue_out <= char_E(Line, Pixel);
      Pixel:= Pixel+1;
    elsif (ascii4 <= "01000110") then
      Red_out <= char_F(Line, Pixel);
      Green_out <= char_F(Line, Pixel);
      Blue_out <= char_F(Line, Pixel);
      Pixel:= Pixel+1;
    elsif (ascii4 <= "01000111") then
      Red_out <= char_G(Line, Pixel);
      Green_out <= char_G(Line, Pixel);
      Blue_out <= char_G(Line, Pixel);
      Pixel:= Pixel+1;
    elsif (ascii4 <= "01001000") then
      Red_out <= char_H(Line, Pixel);
      Green_out <= char_H(Line, Pixel);
      Blue_out <= char_H(Line, Pixel);
      Pixel:= Pixel+1;
    elsif (ascii4 <= "01001001") then
      Red_out <= char_I(Line, Pixel);
      Green_out <= char_I(Line, Pixel);
      Blue_out <= char_I(Line, Pixel);
      Pixel:= Pixel+1;
    elsif (ascii4 <= "01001010") then
      Red_out <= char_J(Line, Pixel);

```

```

        Green_out <= char_J(Line, Pixel);
        Blue_out <= char_J(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii4 <= "01001011") then
        Red_out <= char_K(Line, Pixel);
        Green_out <= char_K(Line, Pixel);
        Blue_out <= char_K(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii4 <= "01001100") then
        Red_out <= char_L(Line, Pixel);
        Green_out <= char_L(Line, Pixel);
        Blue_out <= char_L(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii4 <= "01001101") then
        Red_out <= char_M(Line, Pixel);
        Green_out <= char_M(Line, Pixel);
        Blue_out <= char_M(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii4 <= "01001110") then
        Red_out <= char_N(Line, Pixel);
        Green_out <= char_N(Line, Pixel);
        Blue_out <= char_N(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii4 <= "01001111") then
        Red_out <= char_O(Line, Pixel);
        Green_out <= char_O(Line, Pixel);
        Blue_out <= char_O(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii4 <= "01010000") then
        Red_out <= char_P(Line, Pixel);
        Green_out <= char_P(Line, Pixel);
        Blue_out <= char_P(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii4 <= "01010001") then
        Red_out <= char_Q(Line, Pixel);
        Green_out <= char_Q(Line, Pixel);
        Blue_out <= char_Q(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii4 <= "01010010") then
        Red_out <= char_R(Line, Pixel);
        Green_out <= char_R(Line, Pixel);
        Blue_out <= char_R(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii4 <= "01010011") then
        Red_out <= char_S(Line, Pixel);
        Green_out <= char_S(Line, Pixel);
        Blue_out <= char_S(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii4 <= "01010100") then
        Red_out <= char_T(Line, Pixel);
        Green_out <= char_T(Line, Pixel);
        Blue_out <= char_T(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii4 <= "01010101") then
        Red_out <= char_U(Line, Pixel);

```

```

        Green_out <= char_U(Line, Pixel);
        Blue_out <= char_Ū(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii4 <= "01010110") then
        Red_out <= char_V(Line, Pixel);
        Green_out <= char_V(Line, Pixel);
        Blue_out <= char_V(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii4 <= "01010111") then
        Red_out <= char_W(Line, Pixel);
        Green_out <= char_W(Line, Pixel);
        Blue_out <= char_W̄(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii4 <= "01011000") then
        Red_out <= char_X(Line, Pixel);
        Green_out <= char_X(Line, Pixel);
        Blue_out <= char_X(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii4 <= "01011001") then
        Red_out <= char_Y(Line, Pixel);
        Green_out <= char_Y(Line, Pixel);
        Blue_out <= char_Ȳ(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii4 <= "01011010") then
        Red_out <= char_Z(Line, Pixel);
        Green_out <= char_Z(Line, Pixel);
        Blue_out <= char_Z̄(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii4 <= "11111111") then
        Red_out <= char_Reset(Line, Pixel);
        Green_out <= char_Reset(Line, Pixel);
        Blue_out <= char_Reset(Line, Pixel);
        Pixel:= Pixel+1;
    end if;
end if;
----digit 5
elsif (Horizontal_Counter >= "0111001000")--456
and (Horizontal_Counter <= "0111001111")-- 463
and (Vertical_Counter >= "0011101010") --234
and (Vertical_Counter <= "0011110101") then -- 245
    if (Pixel <= 39) and (line <=11) then
        if (ascii5 <= "00000000") then
            Red_out <= char_Reset(Line, Pixel);
            Green_out <= char_Reset(Line, Pixel);
            Blue_out <= char_Reset(Line, Pixel);
            Pixel:= Pixel+1;
        elsif (ascii5 <= "01000001")then
            Red_out <= char_A(Line, Pixel);
            Green_out <= char_A(Line, Pixel);
            Blue_out <= char_A(Line, Pixel);
            Pixel:= Pixel+1;
        elsif (ascii5 <= "01000010") then
            Red_out <= char_B(Line, Pixel);
            Green_out <= char_B(Line, Pixel);
            Blue_out <= char_B(Line, Pixel);

```

```

Pixel:= Pixel+1;
elsif (ascii5 <= "01000011") then
  Red_out <= char_C(Line, Pixel);
  Green_out <= char_C(Line, Pixel);
  Blue_out <= char_C(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii5 <= "01000100") then
  Red_out <= char_D(Line, Pixel);
  Green_out <= char_D(Line, Pixel);
  Blue_out <= char_D(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii5 <= "01000101") then
  Red_out <= char_E(Line, Pixel);
  Green_out <= char_E(Line, Pixel);
  Blue_out <= char_E(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii5 <= "01000110") then
  Red_out <= char_F(Line, Pixel);
  Green_out <= char_F(Line, Pixel);
  Blue_out <= char_F(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii5 <= "01000111") then
  Red_out <= char_G(Line, Pixel);
  Green_out <= char_G(Line, Pixel);
  Blue_out <= char_G(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii5 <= "01001000") then
  Red_out <= char_H(Line, Pixel);
  Green_out <= char_H(Line, Pixel);
  Blue_out <= char_H(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii5 <= "01001001") then
  Red_out <= char_I(Line, Pixel);
  Green_out <= char_I(Line, Pixel);
  Blue_out <= char_I(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii5 <= "01001010") then
  Red_out <= char_J(Line, Pixel);
  Green_out <= char_J(Line, Pixel);
  Blue_out <= char_J(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii5 <= "01001011") then
  Red_out <= char_K(Line, Pixel);
  Green_out <= char_K(Line, Pixel);
  Blue_out <= char_K(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii5 <= "01001100") then
  Red_out <= char_L(Line, Pixel);
  Green_out <= char_L(Line, Pixel);
  Blue_out <= char_L(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii5 <= "01001101") then
  Red_out <= char_M(Line, Pixel);
  Green_out <= char_M(Line, Pixel);
  Blue_out <= char_M(Line, Pixel);

```

```

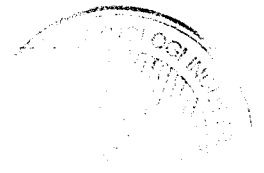
Pixel:= Pixel+1;
elsif (ascii5 <= "01001110") then
  Red_out <= char_N(Line, Pixel);
  Green_out <= char_N(Line, Pixel);
  Blue_out <= char_N(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii5 <= "01001111") then
  Red_out <= char_O(Line, Pixel);
  Green_out <= char_O(Line, Pixel);
  Blue_out <= char_O(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii5 <= "01010000") then
  Red_out <= char_P(Line, Pixel);
  Green_out <= char_P(Line, Pixel);
  Blue_out <= char_P(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii5 <= "01010001") then
  Red_out <= char_Q(Line, Pixel);
  Green_out <= char_Q(Line, Pixel);
  Blue_out <= char_Q(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii5 <= "01010010") then
  Red_out <= char_R(Line, Pixel);
  Green_out <= char_R(Line, Pixel);
  Blue_out <= char_R(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii5 <= "01010011") then
  Red_out <= char_S(Line, Pixel);
  Green_out <= char_S(Line, Pixel);
  Blue_out <= char_S(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii5 <= "01010100") then
  Red_out <= char_T(Line, Pixel);
  Green_out <= char_T(Line, Pixel);
  Blue_out <= char_T(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii5 <= "01010101") then
  Red_out <= char_U(Line, Pixel);
  Green_out <= char_U(Line, Pixel);
  Blue_out <= char_U(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii5 <= "01010110") then
  Red_out <= char_V(Line, Pixel);
  Green_out <= char_V(Line, Pixel);
  Blue_out <= char_V(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii5 <= "01010111") then
  Red_out <= char_W(Line, Pixel);
  Green_out <= char_W(Line, Pixel);
  Blue_out <= char_W(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii5 <= "01011000") then
  Red_out <= char_X(Line, Pixel);
  Green_out <= char_X(Line, Pixel);
  Blue_out <= char_X(Line, Pixel);

```

```

Pixel:= Pixel+1;
elsif (ascii5 <= "01011001") then
  Red_out <= char_Y(Line, Pixel);
  Green_out <= char_Y(Line, Pixel);
  Blue_out <= char_Y(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii5 <= "01011010") then
  Red_out <= char_Z(Line, Pixel);
  Green_out <= char_Z(Line, Pixel);
  Blue_out <= char_Z(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii5 <= "11111111") then
  Red_out <= char_Reset(Line, Pixel);
  Green_out <= char_Reset(Line, Pixel);
  Blue_out <= char_Reset(Line, Pixel);
  Pixel:= Pixel+1;
end if;
end if;
----digit 6
elsif (Horizontal_Counter >= "0111010000")--464
and (Horizontal_Counter <= "0111010111")-- 471
and (Vertical_Counter >= "0011101010") --234
and (Vertical_Counter <= "0011110101") then -- 245
  if (Pixel <= 47) and (line <=11) then
    if (ascii6 <= "00000000") then
      Red_out <= char_Reset(Line, Pixel);
      Green_out <= char_Reset(Line, Pixel);
      Blue_out <= char_Reset(Line, Pixel);
      Pixel:= Pixel+1;
    elsif (ascii6 <= "01000001")then
      Red_out <= char_A(Line, Pixel);
      Green_out <= char_A(Line, Pixel);
      Blue_out <= char_A(Line, Pixel);
      Pixel:= Pixel+1;
    elsif (ascii6 <= "01000010") then
      Red_out <= char_B(Line, Pixel);
      Green_out <= char_B(Line, Pixel);
      Blue_out <= char_B(Line, Pixel);
      Pixel:= Pixel+1;
    elsif (ascii6 <= "01000011") then
      Red_out <= char_C(Line, Pixel);
      Green_out <= char_C(Line, Pixel);
      Blue_out <= char_C(Line, Pixel);
      Pixel:= Pixel+1;
    elsif (ascii6 <= "01000100") then
      Red_out <= char_D(Line, Pixel);
      Green_out <= char_D(Line, Pixel);
      Blue_out <= char_D(Line, Pixel);
      Pixel:= Pixel+1;
    elsif(ascii6 <= "01000101") then
      Red_out <= char_E(Line, Pixel);
      Green_out <= char_E(Line, Pixel);
      Blue_out <= char_E(Line, Pixel);
      Pixel:= Pixel+1;
    elsif (ascii6 <= "01000110") then

```



```

Red_out <= char_F(Line, Pixel);
Green_out <= char_F(Line, Pixel);
Blue_out <= char_F(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii6 <= "01000111") then
Red_out <= char_G(Line, Pixel);
Green_out <= char_G(Line, Pixel);
Blue_out <= char_G(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii6 <= "01001000") then
Red_out <= char_H(Line, Pixel);
Green_out <= char_H(Line, Pixel);
Blue_out <= char_H(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii6 <= "01001001") then
Red_out <= char_I(Line, Pixel);
Green_out <= char_I(Line, Pixel);
Blue_out <= char_I(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii6 <= "01001010") then
Red_out <= char_J(Line, Pixel);
Green_out <= char_J(Line, Pixel);
Blue_out <= char_J(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii6 <= "01001011") then
Red_out <= char_K(Line, Pixel);
Green_out <= char_K(Line, Pixel);
Blue_out <= char_K(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii6 <= "01001100") then
Red_out <= char_L(Line, Pixel);
Green_out <= char_L(Line, Pixel);
Blue_out <= char_L(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii6 <= "01001101") then
Red_out <= char_M(Line, Pixel);
Green_out <= char_M(Line, Pixel);
Blue_out <= char_M(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii6 <= "01001110") then
Red_out <= char_N(Line, Pixel);
Green_out <= char_N(Line, Pixel);
Blue_out <= char_N(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii6 <= "01001111") then
Red_out <= char_O(Line, Pixel);
Green_out <= char_O(Line, Pixel);
Blue_out <= char_O(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii6 <= "01010000") then
Red_out <= char_P(Line, Pixel);
Green_out <= char_P(Line, Pixel);
Blue_out <= char_P(Line, Pixel);
Pixel:= Pixel+1;

```



```

elsif (ascii6 <= "01010001") then
    Red_out <= char_Q(Line, Pixel);
    Green_out <= char_Q(Line, Pixel);
    Blue_out <= char_Q(Line, Pixel);
    Pixel:= Pixel+1;
elsif (ascii6 <= "01010010") then
    Red_out <= char_R(Line, Pixel);
    Green_out <= char_R(Line, Pixel);
    Blue_out <= char_R(Line, Pixel);
    Pixel:= Pixel+1;
elsif (ascii6 <= "01010011") then
    Red_out <= char_S(Line, Pixel);
    Green_out <= char_S(Line, Pixel);
    Blue_out <= char_S(Line, Pixel);
    Pixel:= Pixel+1;
elsif (ascii6 <= "01010100") then
    Red_out <= char_T(Line, Pixel);
    Green_out <= char_T(Line, Pixel);
    Blue_out <= char_T(Line, Pixel);
    Pixel:= Pixel+1;
elsif (ascii6 <= "01010101") then
    Red_out <= char_U(Line, Pixel);
    Green_out <= char_U(Line, Pixel);
    Blue_out <= char_U(Line, Pixel);
    Pixel:= Pixel+1;
elsif (ascii6 <= "01010110") then
    Red_out <= char_V(Line, Pixel);
    Green_out <= char_V(Line, Pixel);
    Blue_out <= char_V(Line, Pixel);
    Pixel:= Pixel+1;
elsif (ascii6 <= "01010111") then
    Red_out <= char_W(Line, Pixel);
    Green_out <= char_W(Line, Pixel);
    Blue_out <= char_W(Line, Pixel);
    Pixel:= Pixel+1;
elsif (ascii6 <= "01011000") then
    Red_out <= char_X(Line, Pixel);
    Green_out <= char_X(Line, Pixel);
    Blue_out <= char_X(Line, Pixel);
    Pixel:= Pixel+1;
elsif (ascii6 <= "01011001") then
    Red_out <= char_Y(Line, Pixel);
    Green_out <= char_Y(Line, Pixel);
    Blue_out <= char_Y(Line, Pixel);
    Pixel:= Pixel+1;
elsif (ascii6 <= "01011010") then
    Red_out <= char_Z(Line, Pixel);
    Green_out <= char_Z(Line, Pixel);
    Blue_out <= char_Z(Line, Pixel);
    Pixel:= Pixel+1;
elsif (ascii6 <= "11111111") then
    Red_out <= char_Reset(Line, Pixel);
    Green_out <= char_Reset(Line, Pixel);
    Blue_out <= char_Reset(Line, Pixel);

```

```

Pixel:= Pixel+1;
end if;
end if;
----- digit7
elsif (Horizontal_Counter >= "0111011000" )--472
and (Horizontal_Counter <= "0111011111")-- 479
and (Vertical_Counter >= "0011101010") --234
and (Vertical_Counter <= "0011110101") then -- 245
if (Pixel <= 55) then
if (ascii7 <= "00000000") then
Red_out <= char_Reset(Line, Pixel);
Green_out <= char_Reset(Line, Pixel);
Blue_out <= char_Reset(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii7 <= "01000001")then
Red_out <= char_A(Line, Pixel);
Green_out <= char_A(Line, Pixel);
Blue_out <= char_A(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii7 <= "01000010") then
Red_out <= char_B(Line, Pixel);
Green_out <= char_B(Line, Pixel);
Blue_out <= char_B(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii7 <= "01000011") then
Red_out <= char_C(Line, Pixel);
Green_out <= char_C(Line, Pixel);
Blue_out <= char_C(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii7 <= "01000100") then
Red_out <= char_D(Line, Pixel);
Green_out <= char_D(Line, Pixel);
Blue_out <= char_D(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii7 <= "01000101") then
Red_out <= char_E(Line, Pixel);
Green_out <= char_E(Line, Pixel);
Blue_out <= char_E(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii7 <= "01000110") then
Red_out <= char_F(Line, Pixel);
Green_out <= char_F(Line, Pixel);
Blue_out <= char_F(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii7 <= "01000111") then
Red_out <= char_G(Line, Pixel);
Green_out <= char_G(Line, Pixel);
Blue_out <= char_G(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii7 <= "01001000") then
Red_out <= char_H(Line, Pixel);
Green_out <= char_H(Line, Pixel);
Blue_out <= char_H(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii7 <= "01001001") then

```

```

Red_out <= char_I(Line, Pixel);
Green_out <= char_I(Line, Pixel);
Blue_out <= char_I(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii7 <= "01001010") then
Red_out <= char_J(Line, Pixel);
Green_out <= char_J(Line, Pixel);
Blue_out <= char_J(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii7 <= "01001011") then
Red_out <= char_K(Line, Pixel);
Green_out <= char_K(Line, Pixel);
Blue_out <= char_K(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii7 <= "01001100") then
Red_out <= char_L(Line, Pixel);
Green_out <= char_L(Line, Pixel);
Blue_out <= char_L(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii7 <= "01001101") then
Red_out <= char_M(Line, Pixel);
Green_out <= char_M(Line, Pixel);
Blue_out <= char_M(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii7 <= "01001110") then
Red_out <= char_N(Line, Pixel);
Green_out <= char_N(Line, Pixel);
Blue_out <= char_N(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii7 <= "01001111") then
Red_out <= char_O(Line, Pixel);
Green_out <= char_O(Line, Pixel);
Blue_out <= char_O(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii7 <= "01010000") then
Red_out <= char_P(Line, Pixel);
Green_out <= char_P(Line, Pixel);
Blue_out <= char_P(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii7 <= "01010001") then
Red_out <= char_Q(Line, Pixel);
Green_out <= char_Q(Line, Pixel);
Blue_out <= char_Q(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii7 <= "01010010") then
Red_out <= char_R(Line, Pixel);
Green_out <= char_R(Line, Pixel);
Blue_out <= char_R(Line, Pixel);
Pixel:= Pixel+1;
elsif (ascii7 <= "01010011") then
Red_out <= char_S(Line, Pixel);
Green_out <= char_S(Line, Pixel);
Blue_out <= char_S(Line, Pixel);
Pixel:= Pixel+1;

```

```

    elsif (ascii7 <= "01010100") then
        Red_out <= char_T(Line, Pixel);
        Green_out <= char_T(Line, Pixel);
        Blue_out <= char_T(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii7 <= "01010101") then
        Red_out <= char_U(Line, Pixel);
        Green_out <= char_U(Line, Pixel);
        Blue_out <= char_U(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii7 <= "01010110") then
        Red_out <= char_V(Line, Pixel);
        Green_out <= char_V(Line, Pixel);
        Blue_out <= char_V(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii7 <= "01010111") then
        Red_out <= char_W(Line, Pixel);
        Green_out <= char_W(Line, Pixel);
        Blue_out <= char_W(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii7 <= "01011000") then
        Red_out <= char_X(Line, Pixel);
        Green_out <= char_X(Line, Pixel);
        Blue_out <= char_X(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii7 <= "01011001") then
        Red_out <= char_Y(Line, Pixel);
        Green_out <= char_Y(Line, Pixel);
        Blue_out <= char_Y(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii7 <= "01011010") then
        Red_out <= char_Z(Line, Pixel);
        Green_out <= char_Z(Line, Pixel);
        Blue_out <= char_Z(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii7 <= "11111111")
        Red_out <= char_Reset(Line, Pixel);
        Green_out <= char_Reset(Line, Pixel);
        Blue_out <= char_Reset(Line, Pixel);
        Pixel:= Pixel+1;
    end if;
end if;
----digit 8
elsif (Horizontal_Counter >= "0111100000")--480
    and (Horizontal_Counter <= "0111100111")-- 487
    and (Vertical_Counter >= "0011101010") --234
    and (Vertical_Counter <= "0011110101") then -- 245
    if (Pixel <= 63) then
        if (ascii8 <= "00000000") then
            Red_out <= char_Reset(Line, Pixel);
            Green_out <= char_Reset(Line, Pixel);
            Blue_out <= char_Reset(Line, Pixel);
            Pixel:= Pixel+1;
        elsif (ascii8 <= "01000001")
            Red_out <= char_A(Line, Pixel);

```

```

        Green_out <= char_A(Line, Pixel);
        Blue_out <= char_A(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii8 <= "01000010") then
        Red_out <= char_B(Line, Pixel);
        Green_out <= char_B(Line, Pixel);
        Blue_out <= char_B(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii8 <= "01000011") then
        Red_out <= char_C(Line, Pixel);
        Green_out <= char_C(Line, Pixel);
        Blue_out <= char_C(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii8 <= "01000100") then
        Red_out <= char_D(Line, Pixel);
        Green_out <= char_D(Line, Pixel);
        Blue_out <= char_D(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii8 <= "01000101") then
        Red_out <= char_E(Line, Pixel);
        Green_out <= char_E(Line, Pixel);
        Blue_out <= char_E(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii8 <= "01000110") then
        Red_out <= char_F(Line, Pixel);
        Green_out <= char_F(Line, Pixel);
        Blue_out <= char_F(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii8 <= "01000111") then
        Red_out <= char_G(Line, Pixel);
        Green_out <= char_G(Line, Pixel);
        Blue_out <= char_G(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii8 <= "01001000") then
        Red_out <= char_H(Line, Pixel);
        Green_out <= char_H(Line, Pixel);
        Blue_out <= char_H(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii8 <= "01001001") then
        Red_out <= char_I(Line, Pixel);
        Green_out <= char_I(Line, Pixel);
        Blue_out <= char_I(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii8 <= "01001010") then
        Red_out <= char_J(Line, Pixel);
        Green_out <= char_J(Line, Pixel);
        Blue_out <= char_J(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii8 <= "01001011") then
        Red_out <= char_K(Line, Pixel);
        Green_out <= char_K(Line, Pixel);
        Blue_out <= char_K(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii8 <= "01001100") then
        Red_out <= char_L(Line, Pixel);

```

```

        Green_out <= char_L(Line, Pixel);
        Blue_out <= char_L(Line, Pixel);
        Pixel:= Pixel+1;
elseif (ascii8 <= "01001101") then
        Red_out <= char_M(Line, Pixel);
        Green_out <= char_M(Line, Pixel);
        Blue_out <= char_M(Line, Pixel);
        Pixel:= Pixel+1;
elseif (ascii8 <= "01001110") then
        Red_out <= char_N(Line, Pixel);
        Green_out <= char_N(Line, Pixel);
        Blue_out <= char_N(Line, Pixel);
        Pixel:= Pixel+1;
elseif (ascii8 <= "01001111") then
        Red_out <= char_O(Line, Pixel);
        Green_out <= char_O(Line, Pixel);
        Blue_out <= char_O(Line, Pixel);
        Pixel:= Pixel+1;
elseif (ascii8 <= "01010000") then
        Red_out <= char_P(Line, Pixel);
        Green_out <= char_P(Line, Pixel);
        Blue_out <= char_P(Line, Pixel);
        Pixel:= Pixel+1;
elseif (ascii8 <= "01010001") then
        Red_out <= char_Q(Line, Pixel);
        Green_out <= char_Q(Line, Pixel);
        Blue_out <= char_Q(Line, Pixel);
        Pixel:= Pixel+1;
elseif (ascii8 <= "01010010") then
        Red_out <= char_R(Line, Pixel);
        Green_out <= char_R(Line, Pixel);
        Blue_out <= char_R(Line, Pixel);
        Pixel:= Pixel+1;
elseif (ascii8 <= "01010011") then
        Red_out <= char_S(Line, Pixel);
        Green_out <= char_S(Line, Pixel);
        Blue_out <= char_S(Line, Pixel);
        Pixel:= Pixel+1;
elseif (ascii8 <= "01010100") then
        Red_out <= char_T(Line, Pixel);
        Green_out <= char_T(Line, Pixel);
        Blue_out <= char_T(Line, Pixel);
        Pixel:= Pixel+1;
elseif (ascii8 <= "01010101") then
        Red_out <= char_U(Line, Pixel);
        Green_out <= char_U(Line, Pixel);
        Blue_out <= char_U(Line, Pixel);
        Pixel:= Pixel+1;
elseif (ascii8 <= "01010110") then
        Red_out <= char_V(Line, Pixel);
        Green_out <= char_V(Line, Pixel);
        Blue_out <= char_V(Line, Pixel);
        Pixel:= Pixel+1;
elseif (ascii8 <= "01010111") then
        Red_out <= char_W(Line, Pixel);

```

```

        Green_out <= char_W(Line, Pixel);
        Blue_out <= char_W(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii8 <= "01011000") then
        Red_out <= char_X(Line, Pixel);
        Green_out <= char_X(Line, Pixel);
        Blue_out <= char_X(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii8 <= "01011001") then
        Red_out <= char_Y(Line, Pixel);
        Green_out <= char_Y(Line, Pixel);
        Blue_out <= char_Y(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii8 <= "01011010") then
        Red_out <= char_Z(Line, Pixel);
        Green_out <= char_Z(Line, Pixel);
        Blue_out <= char_Z(Line, Pixel);
        Pixel:= Pixel+1;
    elsif (ascii8 <= "11111111") then
        Red_out <= char_Reset(Line, Pixel);
        Green_out <= char_Reset(Line, Pixel);
        Blue_out <= char_Reset(Line, Pixel);
        Pixel:= Pixel+1;
    end if;
end if;
----digit 9
elsif (Horizontal_Counter >= "0111101000")--488
    and (Horizontal_Counter <= "0111101111")-- 495
    and (Vertical_Counter >= "0011101010") --234
    and (Vertical_Counter <= "0011110101") then -- 245
    if (Pixel <= 71) then
        if (ascii9 <= "00000000") then
            Red_out <= char_Reset(Line, Pixel);
            Green_out <= char_Reset(Line, Pixel);
            Blue_out <= char_Reset(Line, Pixel);
            Pixel:= Pixel+1;
        elsif (ascii9 <= "01000001")then
            Red_out <= char_A(Line, Pixel);
            Green_out <= char_A(Line, Pixel);
            Blue_out <= char_A(Line, Pixel);
            Pixel:= Pixel+1;
        elsif (ascii9 <= "01000010") then
            Red_out <= char_B(Line, Pixel);
            Green_out <= char_B(Line, Pixel);
            Blue_out <= char_B(Line, Pixel);
            Pixel:= Pixel+1;
        elsif (ascii9 <= "01000011") then
            Red_out <= char_C(Line, Pixel);
            Green_out <= char_C(Line, Pixel);
            Blue_out <= char_C(Line, Pixel);
            Pixel:= Pixel+1;
        elsif (ascii9 <= "01000100") then
            Red_out <= char_D(Line, Pixel);
            Green_out <= char_D(Line, Pixel);
            Blue_out <= char_D(Line, Pixel);

```

```

Pixel:= Pixel+1;
elsif (ascii9 <= "01000101") then
  Red_out <= char_E(Line, Pixel);
  Green_out <= char_E(Line, Pixel);
  Blue_out <= char_E(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii9 <= "01000110") then
  Red_out <= char_F(Line, Pixel);
  Green_out <= char_F(Line, Pixel);
  Blue_out <= char_F(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii9 <= "01000111") then
  Red_out <= char_G(Line, Pixel);
  Green_out <= char_G(Line, Pixel);
  Blue_out <= char_G(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii9 <= "01001000") then
  Red_out <= char_H(Line, Pixel);
  Green_out <= char_H(Line, Pixel);
  Blue_out <= char_H(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii9 <= "01001001") then
  Red_out <= char_I(Line, Pixel);
  Green_out <= char_I(Line, Pixel);
  Blue_out <= char_I(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii9 <= "01001010") then
  Red_out <= char_J(Line, Pixel);
  Green_out <= char_J(Line, Pixel);
  Blue_out <= char_J(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii9 <= "01001011") then
  Red_out <= char_K(Line, Pixel);
  Green_out <= char_K(Line, Pixel);
  Blue_out <= char_K(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii9 <= "01001100") then
  Red_out <= char_L(Line, Pixel);
  Green_out <= char_L(Line, Pixel);
  Blue_out <= char_L(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii9 <= "01001101") then
  Red_out <= char_M(Line, Pixel);
  Green_out <= char_M(Line, Pixel);
  Blue_out <= char_M(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii9 <= "01001110") then
  Red_out <= char_N(Line, Pixel);
  Green_out <= char_N(Line, Pixel);
  Blue_out <= char_N(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii9 <= "01001111") then
  Red_out <= char_O(Line, Pixel);
  Green_out <= char_O(Line, Pixel);
  Blue_out <= char_O(Line, Pixel);

```



```

Pixel:= Pixel+1;
elsif (ascii9 <= "01010000") then
  Red_out <= char_P(Line, Pixel);
  Green_out <= char_P(Line, Pixel);
  Blue_out <= char_P(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii9 <= "01010001") then
  Red_out <= char_Q(Line, Pixel);
  Green_out <= char_Q(Line, Pixel);
  Blue_out <= char_Q(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii9 <= "01010010") then
  Red_out <= char_R(Line, Pixel);
  Green_out <= char_R(Line, Pixel);
  Blue_out <= char_R(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii9 <= "01010011") then
  Red_out <= char_S(Line, Pixel);
  Green_out <= char_S(Line, Pixel);
  Blue_out <= char_S(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii9 <= "01010100") then
  Red_out <= char_T(Line, Pixel);
  Green_out <= char_T(Line, Pixel);
  Blue_out <= char_T(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii9 <= "01010101") then
  Red_out <= char_U(Line, Pixel);
  Green_out <= char_U(Line, Pixel);
  Blue_out <= char_U(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii9 <= "01010110") then
  Red_out <= char_V(Line, Pixel);
  Green_out <= char_V(Line, Pixel);
  Blue_out <= char_V(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii9 <= "01010111") then
  Red_out <= char_W(Line, Pixel);
  Green_out <= char_W(Line, Pixel);
  Blue_out <= char_W(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii9 <= "01011000") then
  Red_out <= char_X(Line, Pixel);
  Green_out <= char_X(Line, Pixel);
  Blue_out <= char_X(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii9 <= "01011001") then
  Red_out <= char_Y(Line, Pixel);
  Green_out <= char_Y(Line, Pixel);
  Blue_out <= char_Y(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii9 <= "01011010") then
  Red_out <= char_Z(Line, Pixel);
  Green_out <= char_Z(Line, Pixel);
  Blue_out <= char_Z(Line, Pixel);

```

```

Pixel:= Pixel+1;
elsif (ascii9 <= "11111111") then
  Red_out <= char_Reset(Line, Pixel);
  Green_out <= char_Reset(Line, Pixel);
  Blue_out <= char_Reset(Line, Pixel);
  Pixel:= Pixel+1;
end if;
end if;

```

----digit 10

```

elsif (Horizontal_Counter >= "0111110000")--496
  and (Horizontal_Counter <= "0111110111")-- 503
  and (Vertical_Counter >= "0011101010") --234
  and (Vertical_Counter <= "0011101011") then -- 245
  if (Pixel <= 82) then
    if (ascii10 <= "00000000") then
      Red_out <= char_Reset(Line, Pixel);
      Green_out <= char_Reset(Line, Pixel);
      Blue_out <= char_Reset(Line, Pixel);
      Pixel:= Pixel+1;
    elsif (ascii10 <= "01000001")then
      Red_out <= char_A(Line, Pixel);
      Green_out <= char_A(Line, Pixel);
      Blue_out <= char_A(Line, Pixel);
      Pixel:= Pixel+1;
    elsif (ascii10 <= "01000010") then
      Red_out <= char_B(Line, Pixel);
      Green_out <= char_B(Line, Pixel);
      Blue_out <= char_B(Line, Pixel);
      Pixel:= Pixel+1;
    elsif (ascii10 <= "01000011") then
      Red_out <= char_C(Line, Pixel);
      Green_out <= char_C(Line, Pixel);
      Blue_out <= char_C(Line, Pixel);
      Pixel:= Pixel+1;
    elsif (ascii10 <= "01000100") then
      Red_out <= char_D(Line, Pixel);
      Green_out <= char_D(Line, Pixel);
      Blue_out <= char_D(Line, Pixel);
      Pixel:= Pixel+1;
    elsif (ascii10 <= "01000101") then
      Red_out <= char_E(Line, Pixel);
      Green_out <= char_E(Line, Pixel);
      Blue_out <= char_E(Line, Pixel);
      Pixel:= Pixel+1;
    elsif (ascii10 <= "01000110") then
      Red_out <= char_F(Line, Pixel);
      Green_out <= char_F(Line, Pixel);
      Blue_out <= char_F(Line, Pixel);
      Pixel:= Pixel+1;
    elsif (ascii10 <= "01000111") then
      Red_out <= char_G(Line, Pixel);
      Green_out <= char_G(Line, Pixel);
      Blue_out <= char_G(Line, Pixel);
      Pixel:= Pixel+1;
    end if;
  end if;
end if;

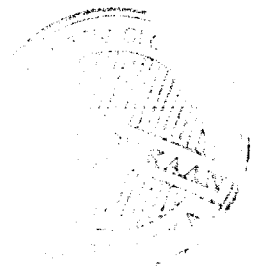
```

```

elsif (ascii10 <= "01001000") then
  Red_out <= char_H(Line, Pixel);
  Green_out <= char_H(Line, Pixel);
  Blue_out <= char_H(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii10 <= "01001001") then
  Red_out <= char_I(Line, Pixel);
  Green_out <= char_I(Line, Pixel);
  Blue_out <= char_I(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii10 <= "01001010") then
  Red_out <= char_J(Line, Pixel);
  Green_out <= char_J(Line, Pixel);
  Blue_out <= char_J(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii10 <= "01001011") then
  Red_out <= char_K(Line, Pixel);
  Green_out <= char_K(Line, Pixel);
  Blue_out <= char_K(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii10 <= "01001100") then
  Red_out <= char_L(Line, Pixel);
  Green_out <= char_L(Line, Pixel);
  Blue_out <= char_L(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii10 <= "01001101") then
  Red_out <= char_M(Line, Pixel);
  Green_out <= char_M(Line, Pixel);
  Blue_out <= char_M(Line, Pixel);
  Pixel:= Pixel+1;

elsif (ascii10 <= "01001110") then
  Red_out <= char_N(Line, Pixel);
  Green_out <= char_N(Line, Pixel);
  Blue_out <= char_N(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii10 <= "01001111") then
  Red_out <= char_O(Line, Pixel);
  Green_out <= char_O(Line, Pixel);
  Blue_out <= char_O(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii10 <= "01010000") then
  Red_out <= char_P(Line, Pixel);
  Green_out <= char_P(Line, Pixel);
  Blue_out <= char_P(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii10 <= "01010001") then
  Red_out <= char_Q(Line, Pixel);
  Green_out <= char_Q(Line, Pixel);
  Blue_out <= char_Q(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii10 <= "01010010") then
  Red_out <= char_R(Line, Pixel);
  Green_out <= char_R(Line, Pixel);
  Blue_out <= char_R(Line, Pixel);

```



```

Pixel:= Pixel+1;
elsif (ascii10 <= "01010011") then
  Red_out <= char_S(Line, Pixel);
  Green_out <= char_S(Line, Pixel);
  Blue_out <= char_S(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii10 <= "01010100") then
  Red_out <= char_T(Line, Pixel);
  Green_out <= char_T(Line, Pixel);
  Blue_out <= char_T(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii10 <= "01010101") then
  Red_out <= char_U(Line, Pixel);
  Green_out <= char_U(Line, Pixel);
  Blue_out <= char_U(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii10 <= "01010110") then
  Red_out <= char_V(Line, Pixel);
  Green_out <= char_V(Line, Pixel);
  Blue_out <= char_V(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii10 <= "01010111") then
  Red_out <= char_W(Line, Pixel);
  Green_out <= char_W(Line, Pixel);
  Blue_out <= char_W(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii10 <= "01011000") then
  Red_out <= char_X(Line, Pixel);
  Green_out <= char_X(Line, Pixel);
  Blue_out <= char_X(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii10 <= "01011001") then
  Red_out <= char_Y(Line, Pixel);
  Green_out <= char_Y(Line, Pixel);
  Blue_out <= char_Y(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii10 <= "01011010") then
  Red_out <= char_Z(Line, Pixel);
  Green_out <= char_Z(Line, Pixel);
  Blue_out <= char_Z(Line, Pixel);
  Pixel:= Pixel+1;
elsif (ascii10 <= "00100000") then
  Red_out <= char_Reset(Line, Pixel);
  Green_out <= char_Reset(Line, Pixel);
  Blue_out <= char_Reset(Line, Pixel);
  Pixel:= Pixel+1;
end if;
end if;

```

```

elsif(Pixel >= 83) then -- Semua yang ga aktif hitam
  Red_out <= '0';
  Green_out <= '0';

```

```

                                Blue_out <= '0';

end if;
end if;

if (Horizontal_Counter > "0000000000" )
    and (Horizontal_Counter < "0001100001" ) -- 96+1
    then
                                hs_out <= '0';
        else
            hs_out <= '1';
    end if;

if (Vertical_Counter > "0000000000" )
    and (Vertical_Counter < "0000000011" ) -- 2+1
    then
        vs_out <= '0';
    else
        vs_out <= '1';
    end if;

Horizontal_Counter <= Horizontal_Counter+"0000000001";
if (Horizontal_Counter="1100100000") then ---800
    Vertical_Counter <= Vertical_Counter+"0000000001";
    Horizontal_Counter <= "0000000000";
    Pixel:= 0;
if (Vertical_Counter >= "0011101010") ----Line pertama text
and (Vertical_Counter <= "0100000100") then ----Line paling bawah dari text 260
    if (Line <= 11) then
        Line:= Line+1;
    elsif (Line >= 12) then
        Line:= 0;
    end if;
end if;
end if;
if (Vertical_Counter="1000001001") then ----521
    Vertical_Counter <= "0000000000";
    Line:= 0;
end if;
end if;
end process;
end Behavioral;

```

LAMPIRAN 2

Gambar RTL Schematic

