LAPORAN LATIHAN ILMIAH TAHAP AKHIR II

RUN LENGTH ENCODING IN IMAGE COMPRESSION

Dihasilkan Oleh :
Yusniza binti Yaakub
WEK 000186

Di bawah Penyeliaan :
En. Noorzaily Mohamed Nor
Fakulti Sains Komputer Dan Teknologi Maklumat
Universiti Malaya

Moderator :
En. Mohd Yamani Idna Idris
Fakulti Sains Komputer Dan Teknologi Maklumat
Universiti Malaya

ABSTRAK

Dokumen ini adalah mengenai projek Latihan Ilmiah Tahap Akhir 1.Projek yang dijalankan bertajuk Pengekodan Run-Length dalam Pemampatan Imej.Ia adalah berkaitan dengan pengimplimentasian algoritma Run-Length dalam perkakasan yang mana kod bagi perkakasan ini di tulis menggunakan bahasa pengaturcaraan VHDL. Pengekodan Run-Length ini berfungsi dengan mencari dan mengkodkan piksel yang bersebelahan yang mempunyai nilai kecerahan atau grey-level yang sama. Input imej yang dimasukkan adalah bersaiz 8 piksel x 8 piksel . Jenis perwakilan imej digital yang digunakan adalah imej penskalaan kelabu atau grey scale. Imej ini menggunakan perwakilan 8 bit bagi satu piksel.

PENGHARGAAN

Alhamdulillah , syukur kepada Ilahi,kerana dengan limpah kurnia dan keizinan-Nya maka saya boleh menyiapkan Projek Latihan Ilmiah Tahap ini.Di sini saya ingin mengambil peluang untuk mengucapkan setinggi penghargaan kepada individu-individu yang telah membantu bagi menyiapkan projek ini. Tanpa bantuan mereka ,mungkin projek ini tidak dapat disiapkan dengan sempurna.

Penghargaan dan penghormatan kepada penyelia saya ,En Noorzaily Mohamed Nor ,kerana telah memberi garis panduan dan tunjuk ajar yang berguna bagi menyiapakan Projek Ilmiah Tahap Akhir ini. Terima kasih juga ditujukan kepada moderator , En Mohd Yamani Idna Idris di atas pandangan bernas yang diberikan. Terima kasih saya tujukan kepada En Zaidi Razak di atas bantuan yang diberikan.

Tidak lupa kepada ibu bapa dan ahli keluarga yang telah memberi sokongan ,dorongan doa sepanjang perjuangan saya. Penghargaan juga ditujukan kepada rakan-rakan seperjuangan yang banyak membantu mengeluarkan idea sehingga projek ini dapat disiapkan dengan jayanya.

Sekian, terima kasih.

Yusniza binti Yaakub,

Fakulti Sains Komputer dan Teknologi Maklumat,

Universiti Malaya.

SENARAI KANDUNGAN **MUKA SURAT**

SENARAI RAJAH **MUKA SURAT**

SENARAI JADUAL **MUKA SURAT**

BAB 1
PENGENALAN PROJEK

1.1 PENGENALAN

Teknologi pemampatan imej (image compression) merupakan element penting bagi memenuhi permintaan yang tinggi terhadap peralatan dan aplikasi multimedia.Sistem pemprosesan imej yang terkini menggunakan beberapa teknologi pemampatan dengan kehendak semulajadi iaitu pelaksanaan yang berkelajuan tinggi.Dengan pengimplementasian algoritma pemadatan ke dalam perkakasan,peningkatan yang terhasil dalam kelajuan pemprosesan membolehkan pemadatan data digunakan dalam pelbagai bidang. Terdapat beberapa jenis algoritma untuk pemampatan imej iaitu Pengekodan Run-Length, Pengekodan Huffman, dan pengkodan LZW. Bagaimanapun Latihan Ilmiah ini lebih menumpukan kepada algoritma Run-Length yang diimplementasikan dalam perkakasan.Pengekodan Run-Length akan ditulis dalam kod VHDI. (Very High Description Language) yang digunakan untuk menerangkan sesuatu perkakasan digital.

1.2 DEFINASI MASALAH

1) Menghadapi kesukaran untuk menulis kod program bagi Run-Length Encoding dalam VHDI. .Algoritma ini mungkin lebih mudah ditulis di dalam kod untuk perisian seperti C dan C++.

1.3 SKOP PROJEK

1) Imej yang diproses adalah imej grayscale bersaiz 8 piksel x 8 piksel.Imej ini menggunakan perwakilan 8 bit bagi satu piksel.

2) Imej yang terhasil adalah imej yang telah melalui proses pemampatan dan proses pemampatan ini ditumpukan kepada imej statik.

1.4 OBJEKTIF PROJEK

1) Untuk menjana IP (Intellectual Property) bagi Pengekodan Run-Length dalam bentuk kod VHDL.
2) Mengimplimentasikan algoritma Pengekodan run-Length ke dalam perkakasan bagi pelaksanaan yang lebih pantas.
3) Untuk memahami pengimplimentasian menggunakan bahasa pengaturcaraan VHDL dalam menulis kod program untuk perkakasan.

1.5 KEKANGAN

1) 'Error' atau kesalahan yang berlaku semasa proses simulasi sukar untuk diperbetulkan.

1.6 KEPENTINGAN PROJEK

1) Memenuhi keperluan bagi penghasilan computer bersaiz kecil
2) Menyokong proses pemampatan yang mudah .Dengan adanya perkakasan ini, proses pemampatan boleh dilakukan dalam persekitaran real-time dan menyumbang kepada pelaksanaan yang lebih pantas.

3) Menambah kebolehan peranti.Fungsi peranti akan lebih bertambah dengan penghasilan perkakasan ini.Ini bermakna fungsi yang dimainkan oleh perkakasan semakin bertambah dan seterusnya menambah kebolehpercayaan peranti.

1.7 PENJADUALAN

Penjadualan adalah perlu bagi mamastikan projek dapat disiapkan dalam masa yang ditetapkan.Untuk memastikan perkara ini ,satu jadual ringkas yang menerangkan tempoh masa dan aktiviti-aktiviti sepanjang proses pembangunan projek dibuat.Jadual ini meliputi keseluruhan aktiviti bermula dengan kajian sehingga ke peringkat dokumentasi.

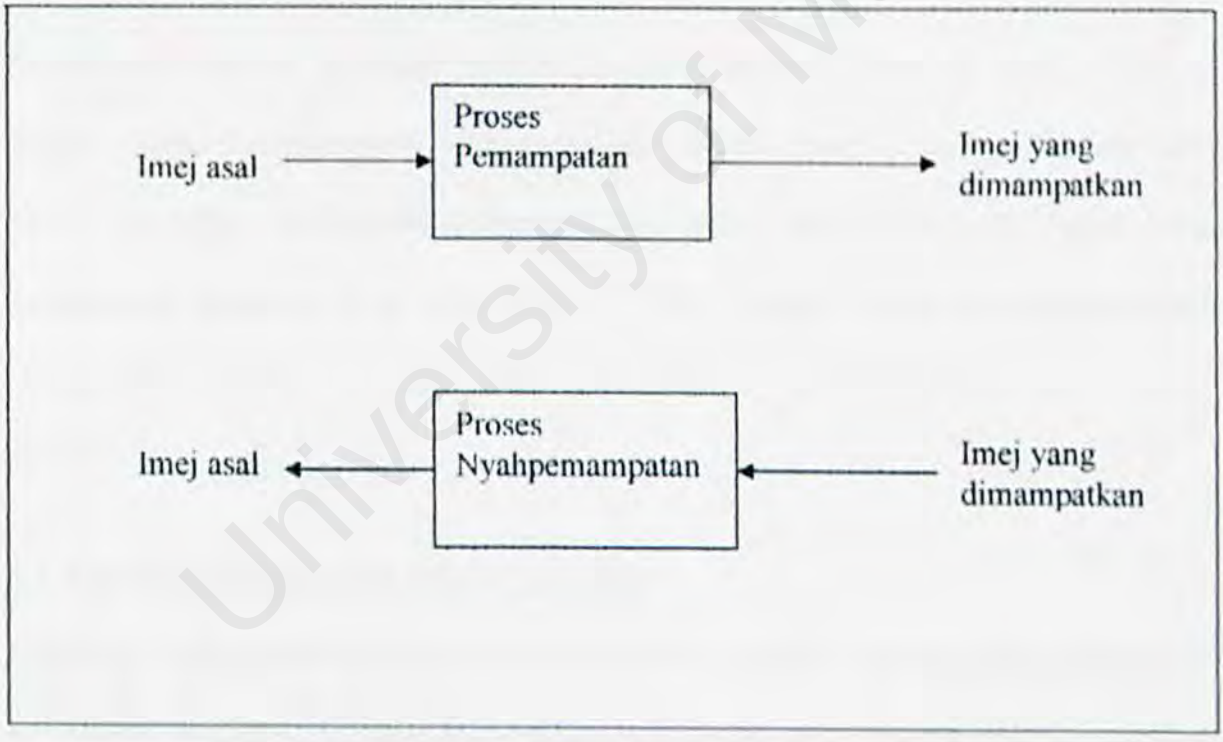
FASA DAN AKTIVITI	TEMPOH FASA DAN AKTIVITI								
	Jun 2003	Julai 2003	Ogos 2003	Sept 2003	Okt 2003	Nov 2003	Dis 2003	Jan 2003	Feb 2003
KAJIAN PERINGKAT AWAL									
Mempelajari perisian VHDL yang akan digunakan									
Mengkaji konsep pemampatan data secara teori									
Mengkaji konsep pemprosesan imej digital									
Rekabentuk									
Pembentukan kod program dalam VHDL									
Pengujian									
Dokumentasian									

Rajah 1 : Penjadualan Projek

BAB 2
KAJIAN LITERASI

2.1 PEMAMPATAN IMEJ

Pemadatan imej merupakan satu proses mengurangkan saiz bagi sesuatu fail data imej asal kepada saiz yang lebih kecil dan imej asal ini boleh diperolehi semula melalui teknik nyah pemampatan.Ia juga boleh dikatakan sebagai satu proses dalam satu aliran bit untuk menghasilkan satu maklumat yang dikehendaki dalam bilangan bit yang lebih kecil.Matlamat utama dalam pemampatan adalah untuk mewakilkan sesuatu imej dalam bilangan bit sekecil yang boleh , atau dengan lebih tepat lagi , untuk menjimatkan ruang simpanan dan kapasiti saluran transmisi dalam rangkaian. Terdapat dua jenis teknik pemampatan data atau imej iaitu "lossless" dan "lossy". Rajah 2 menunjukkan proses umum dalam pemampatan dan nyahpemampatan.



Rajah 2.1 : Proses pemampatan dan nyahpemampatan

2.2 TEKNIK PEMAMPATAN 'LOSSLESS'

Pemampatan data secara 'lossless' digunakan apabila data yang melalui proses penyahmampatan sama dengan keadaan sebelum pemampatan.Fail teks disimpan dengan menggunakan teknik 'lossless', memandangkan satu karektor akan mencacatkan teks yang hendak dimampatkan secara umumnya juga data-data lain seperti imej,grafik,audio,dan video juga menggunakan teknik ini. Antara algoritma pemampatan yang menggunakan teknik pemampatan ini adalah algoritma Pengekodan Run-Length, Pengekodan Huffman dan Pengekodan LZW.

2.3 TEKNIK PEMAMPATAN 'LOSSINESS'

Pemampatan cara ini berfungsi dengan tanggapan bahawa data tidak perlu disimpan dengan sempurna.Kebanyakan maklumat secara mudah boleh dibuang daripada imej ,video dan audio ; apabila dinyahmampat data masih boleh mempunyai kualiti.Ratio pemampatan dengan cara ini lebih besar daripada lossless. Contoh pemampatan yang menggunakan teknik ini adalah 'Transform coding' (DCT/Wavelets) dan 'Vector quantisation'.

2.4 KEPENTINGAN PEMAMPATAN IMEJ

Teknologi pemampatan imej merupakan satu elemen penting bagi memenuhi permintaan yang tinggi terhadap peralatan dan aplikasi multimedia. Pengurangan atau pemampatan saiz fail ini diperlukan untuk disesuaikan dengan keperluan lebar jalur bagi sesuatu sistem transmisi, sama juga dengan keperluan storan bagi sesuatu pangkalan data

komputer. Sebagai contoh satu imej bersaiz 512x512, imej 8 bit memerlukan 2,097,152 bit untuk storan. Sekiranya ia perlu dihantar melalui rangkaian internet,ia mungkin mengambil masa beberapa minit untuk penghantaran itu berlaku. Sebagai contoh : Untuk penghantaran satu imej berwarna bersaiz 512x512,24-bit (8 bit/piksel/warna) melalui modem pada 28.8 kbaud(kilobit/saat) mengambil masa selama 3.6 minit. Bayangkan sekiranya beberapa imej perlu dihantar serentak,ia tentunya memerlukan masa yang lebih panjang. Oleh sebab itu proses pemampatan data diperlukan - salah satunya adalah untuk mengurangkan masa transmisi. Sistem pemprosesan imej yang terkini menggunakan beberapa teknologi pemampatan disesuaikan dengan kehendak semulajadi iaitu memerlukan pelaksanaan yang berkelajuan tinggi.Terdapat beberapa algoritma pemampatan bagi data imej seperti pengekodan Run-Length, pengkodan Huffman , Lemple Ziv Welch dan sebagainya. Manakala piawai yang biasa dalam proses pemadatan imej ialah :

- JPEG (pemadatan imej tetap)
- MPEG-1 (pemadatan imej bergerak)
- MPEG-2 (pemadatan imej bergerak bagi aplikasi TV digital)
- H.261 dan H.263 (pemadatan imej bergerak bagi persidangan melalui video dan videotelefoni).

Dengan pengimplementasian algoritma pemampatan ini ke dalam perkakasan,peningkatan yang terhasil dalam kelajuan pemprosesan membolehkan pemampatan data digunakan dalam pelbagai bidang.

2.5 PERWAKILAN IMEJ DIGITAL

Sedia maklum bahawa system visual manusia menerima imej input sebagai himpunan ruang cahaya yang dipanggil imej optikal.Imej optikal diwakilkan dengan informasi video dalam bentuk isyarat elektrik analog untuk menjana imej digital, *I(r,c)*.Imej digital, *I(r,c)* diwakilkan sebagai data tatasusunan 2 dimensi di mana setiap nilai piksel sama dengan kecerahan bagi imej pada sesuatu titik (r,c).Terdapat beberapa jenis imej iaitu imej binary,imej gray-scale,dan imej berwarna .

2.5.1 IMEJ PERDUAAN (BINARY)

Imej binari merupakan imej yang paling ringkas yang diambil dari dua nilai,biasanya hitam dan putih atau '0' dan '1'.Imej binari dirujuk sebagai imej 1 bit/piksel kerana ia hanya mengambil 1 digit binari untuk mewakilkan satu piksel.Imej ini digunakan dalam aplikasi computer untuk bentuk asas,dan garis luaran.Imej binary biasanya dicipta dari imej gray-scale melalui operasi *threshold*,dimana setiap piksel di atas nilai threshold akan bertukar menjadi putih('1') dan piksel yang berada bawah nilai threshold akan bertukar menjadi hitam ('0').

2.5.2 IMEJ GRAY SCALE

Imej Gray-scale dirujuk sebagai monokrom,atau imej satu warna.Ia hanya mempunyai maklumat bagi kecerahan,bukan maklumat bagi warna.Bilangan bit yang digunakan dalam setiap piksel menentukan bilangan bagi tahap kecerahan berbeza yang boleh diperolehi.Imej biasa mengandungi data 8 bit/piksel,yang mana membenarkan 256 (0-255)tahap kecerahan (gray).

2.5.3 IMEJ BERWARNA

Imej berwarna dimodelkan sebagai data imej 3 band monokrom,yang mana setiap band data mewakili warna yang berbeza.Maklumat sebenar yang disimpan dalam data imej digital adalah maklumat kecerahan bagi setiap band spectral.Apabila imej dipaparkan pada skrin,maklumat kecerahan yang sama dipaparkan pada skrin oleh elemen gambar yang kuasa pancaran cahaya sama dengan sesuatu warna tersebut.Biasanya imej berwarna diwakilkan sebagai Merah (red,R) Hijau(green,G) dan biru(Blue,B) atau imej RGB.Menggunakan piawai monokrom 8-bit sebagai model,imej berwarna yang sama mempunyai 24-bit per piksel-8-bit untuk setiap warna (red,green,blue).

2.6 PENGEKODAN RUN-LENGTH

Run-Length Encoding merupakan satu algorimta pemampatan data yang paling ringkas tetapi berguna bagi set data yang mengandungi jujukan yang berturutan. Sebagai contoh, teks file yang mempunyai ruang tab mungkin boleh dimampatkan dengan baik menggunakan algoritma ini.RLE berfungsi dengan mencari bilangan jujukan yang berulang bagi sesuatu karektor di dalam aliran input dan digantikan dengan kod 3 bait. Kod tersebut terdiri daripada karektor flag , bilangan bait ,dan karektor yang berulang.Sebagai contoh , turutan ``AAAAAABBBBCCCCC" mungkin boleh diwakilkan sebagai ``*A6*B4*C5" menggunakan algoritma RLE. Didapati 6 bait telah dapat dijimatkan . RLE juga boleh digunakan dalam pemampatan imej dan ia digunakan sebagai salah satu langkah pemampatan dalam piawai pemampatan imej JPEG.

2.7 PENGEKODAN RUN-LENGTH DALAM PEMAMPATAN IMEJ

Format bagi pemampatan imej yang sedia ada seperti piawai (.gif dan JPG) memberikan prestasi yang cukup baik dalam kebanyakan keadaan. Algoritma Pengkodan Run-Length ini akan memberi hsil yang baik. Program ini akan membuang piksel –piksel yang dikelilingi oleh piksel yang mempunyai warna yang sama.

- Bermula dengan piksel yang paling kiri pada sebelah atas, Pengecaman imej bermula dari sebelah kiri ke kanan , dan program bergerak ke baris seterusnya.

- Baca piksel yang berada pada atas kiri, dan ia dioutputkan.Sekiranya terdapat sekiranya piksel yang berikutnya mempunyai nilai yang berbeza , maka perlu bergerak ke piksel yang seterusnya, jika tidak, outputkan kunci dan panjang jujukan bagi nilai yang sama.

- Sekiranya panjang jujukan lebih besar daripada sepatutnya , piksel itu dioutputkan semula diikuti dengan baki jujukan yang belum dikodkan. Proses ini tidak akan kodkan nilai piksel yang mempunyai hanya satu jujukan.

- Bagi imej yang mempunyai dua tone warna hitam dan putih (imej binary) ,teknik yang sama juga digunakan.

- Bagaimanapun, pengecaman piksel mungkin juga sesuai menggunakan teknik lajur-ke-lajur selain baris-ke-baris (contohnya bagi imej yang mempunyai jalur)

- Prestasi adalah tidak sama pada mana-mana pembesaran bagi imej asal : Pengekodan adalah baik bagi panjang jujukan yang lebih daripada tiga.

2.8 KEBAIKAN PENGEKODAN RUN-LENGTH DALAM PERKAKASAN

1) RLE mempunyai algoritma kod pemampatan yang mudah difahami dan mempunyai konsep yang ringkas.

2) Senang diimplementasikan dalam aturcara

- algoritma RLE biasanya digunakan dalam perisian dan sekarang ia cuba diimplementasikan dalam perkakasan di mana kod bagi perkakasan ditulis dalam VHDL.

3) Imej yang telah dikodkan boleh dibentukkan ke imej asal melalui proses nyah-mampat tanpa mencacatkan struktur imej tersebut kerana RLE menggunakan teknik pemampatan "lossless".

4) RLE sesuai untuk digunakan dalam pemampatan imej statik dan bergerak

5) Pelaksanaan yang lebih pantas dengan pengimplimentasian dalam perkakasan tanpa program ditulis ke dalam kod perisian.

BAB 3
VHDL

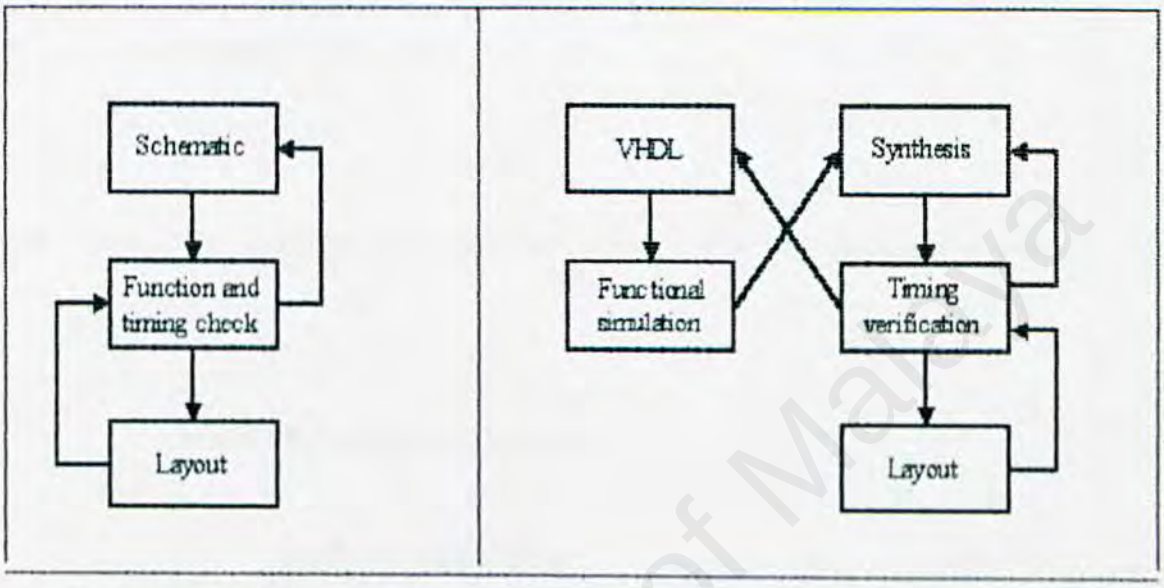
3.1 PENGENALAN KEPADA VHDL

VHDL adalah ringkasan bagi VHSIC Hardware Description Language.VHSIC pula adalah ringkasan bagi Very High Speed Integrated Circuits). VHDL merupakan satu bahasa yang boleh menerangkan sesuatu perkakasan dan boleh digunakan untuk memodelkan sistem digital. Ia boleh menerangkan kelakuan dan struktur bagi rekabentuk perkakasan digital seperti ASIC , FPGA dan litar digital lain. Simulasi dan sistesis merupakan dua alatan yang berfungsi dalam bahasa VHDL. VHDL tidak menghadkan pengguna menggunakan satu jenis penerangan tetapi ia membenarkan rekabentuk diterangkan dalam pelbagai metodologi seperti top-down , bottom up atau middle out. Rekabentuk tahap tinggi yang baik memerlukan bahasa ,alatan dan metodologi yang sesuai.

3.2 METODOLOGI REKABENTUK DALAM VHDL

Rajah 3.1(a) menunjukkan bagaimana pendekatan rekabentuk awal yang mana prosesnya bermula dengan melukis satu skematik , dan kemudian diikuti dengan simulasi fungsi dan masa berdasarkan kepada skematik tersebut. Semua kesalahan dalam rekabentuk akan diperbetulkan semula dengan memperbaharui skematik tersebut.Untuk perbandingan , pendekatan rekabentuk menggunakan VHDL ditunjukkan dalam rajah 3.1(b) yang mana rekabentuk diterangkan menggunakan VHDL dan pengesahan dibuat dengan menggunakan simulasi VHDL. Ini memberikan satu kaedah yang lebih cepat berbanding proses simulasi berdasarkan rajah 3.1(a). Dengan mengubahsuai kod VHDL

,perekabentuk boleh mencipta rekabentuk yang lebih cepat dan cekap dan boleh menerokai pelbagai senibina rekabentuk.Apabila fungsi berpadanan dengan keperluan sistem, kod boleh digunakan untuk menjana skematik terakhir.



Rajah 3.1 (a) Pendekatan rekabentuk Skematik tradisional

Rajah 3.1 (b) pendekatan rekabentuk berasaskan VHDL

3.3 BENTUK PENERANGAN

VHDL membolehkan penerangan tentang sesuatu rekabentuk digital dilakukan dalam pelbagai peringkat berasingan seperti algoritma, dan tahap get logic. Dalam rekabentuk atas-bawah, perekabentuk mewakilkan system pada peringkat tinggi dan menukarkannya kepada proses rekabentuk berstruktur melalui sistesis. Untuk menerangkan sesuatu entity,

VHDL menyediakan 5 jenis bentuk asas yang dipanggil unit rekabentuk. Unit rekabentuk tersebut adalah

- Pengisytiharan entity
- Architecture
- Pengisytiharan konfigurasi
- Pengisytiharan Pakej
- Badan Pakej

Entiti dimodelkan menggunakan pengisytiharan entity dan sekurang-kurangnya satu badan architecture.

ENTITY ComponentName **IS**

Input and output ports

Physical and other parameters

END ComponentName;

ARCHITECTURE identifier **OF** ComponentName **IS** declarations

BEGIN

specification of model in term of its inputs and influenced by physical and other parameters

END identifier;

3.3.1 ENTITI

Entiti menerangkan unit rekabentuk asas dan membina pengisytiharan 'port' yang bersambung kepada system .Ia mewakili keseluruhan system .

Format bagi entity

ENTITY entity_name is

Port (port list);

end entity_name;

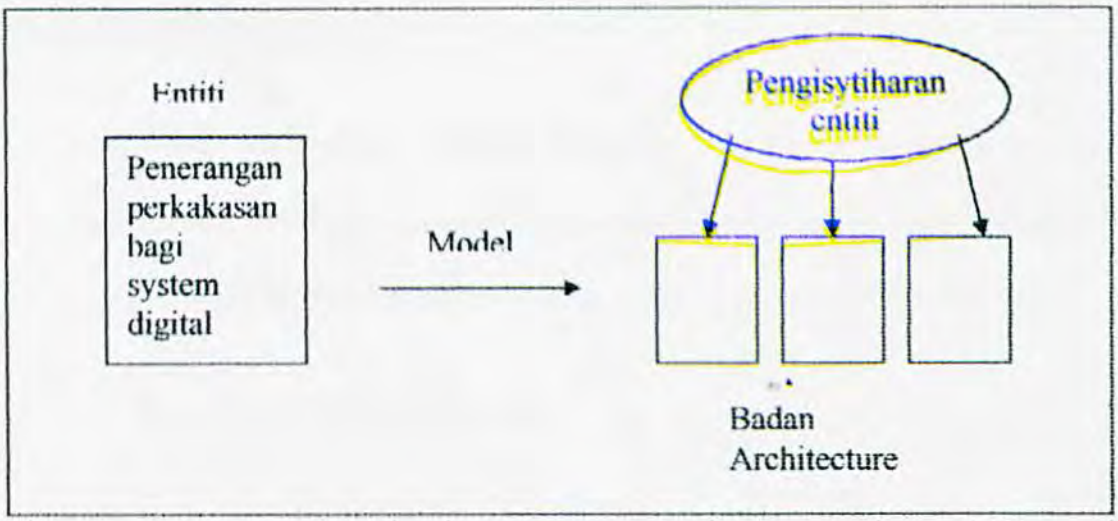
'Port' berfungsi untuk menyediakan saluran untuk komunikasi antara entity dengan persekitarannya.Setiap 'port' mempunyai nama ,mod dan jenis.Format bagi 'port' adalah

port (port name : mode type_indication[bus])

Pengisytiharan 'bus' dibuat jika 'port' disambung dari satu keluaran dari modul atau sistem lain.

3.3.2 ARCHITECTURE

Badan Architecture mengandungi penerangan dalaman bagi entitiy :sebagai contoh kumpulan komponen yang bersambung yang mewakili struktur bagi entity, kumpulan pernyataan yang berturutan atau serentak mewakili kelakuan entity.Setiap perwakilan boleh diklasifikasikan dalam badan architecture yang berlainan atau boleh digabungkan sekali di dalam satu badan architecture. Rajah 4 menunjukkan satu entiti dan model.



Rajah 3.2 : Satu entiti dan satu model

3.3.3 PENGISYTIHARAN KONFIGURASI

Pengisytiharan konfigurasi digunakan untuk mencipta konfigurasi bagi entiti. Ia menentukan gabungan beberapa badan architecture menjadi satu badan architecture sahaja yang mungkin disatukan dengan entity.Pengisytiharan konfigurasi juga menetukan gabungan komponen yang digunakan dalam badan *architecture* tertentu dengan entiti lain.

CONFIGURATION ConfName **OF** ComponentName **IS**

Bindings of Entities and Architecture

Specifying paramenters of a design

END CONFIGURATION :

3.3.4 PENGISTIHARAN PAKEJ

Pengisytiharan Pakej menerangkan tentang kumpulan pengisytiharan yang berkaitan seperti pengisytiharan jenis (type) ,pengisytiharan sub-jenis (subtype) ,dan pengistiharan sub-program yang mana boleh dikongsi melalui dua atau lebih unit rekabentuk.

PACKAGE PackageName IS

Componenr Declaration

Sub-program declarations

Type definitions

END PackageName ;

PACKAGE BODY PackageName IS

Sub-programs.

END PackageName ;

3.3.5 BADAN PAKEJ

Badan pakej mengandungi penerangan bagi sub-program yang diisytiharkan dalam pengisytiharan pakej.

3.3.6 LIBRARY

Library terdiri daripada himpunan pakej dan komponen untuk digunakan dalam rekabentuk lain (mempunyai kebolehan untuk digunakan semula.

```
LIBRARY LibName ;

USE LibName.SubPackage.Scope

LIBRARY IEEE ;

USE IEEE.std_logic_1164. ALL ;

LIBRARY WORK:

USE WORK.util_package.int2bit;
```

3.4 SIMULASI

Setelah dimodelkan penerangan rekabentuk yang dijanakan oleh VHDL mestilah dilarikan melalui simulasi VHDL bagi mengesahkan fungsi bagi sesuatu system.Untuk mensimulasikan rekabentuk ,perekabentuk mesti membekalkan alat simulasi bersama dengan set yang dipanggil stimuli. Sistem akan bertindakbalas dengan stimuli ini dan kemudian akan system akan dianalisa samada ia berfungsi dengan betul atau tidak. Simulasi boleh digunakan di dalam mana-mana peringkat dalam proses rekabentuk.Pada

peringkat yang lebih tinggi, proses simulasi akan berlaku dengan pantas ,tetapi tidak memberikan maklumat lanjut tentang fungsi sesuatu litar dan pemasaan.Pada tahap rendah bagi proses rekabentuk , simulasi akan mengambil masa yang lebih lama tetapi ia menyediakan maklumat terperinci tentang fungsi dan pemasaan bagi sesuatu litar. VHDL membenarkan rekabentuk pada pelbagai peringkat yang mana sesetengah modul diterangkan pada peringkat yang lebih tinggi dan sesetengah modul pula diterangkan pada peringkat rendah.Kebaikannya adalah perekabentuk boleh fokus kepada rekabentuk modul pemasaan kritikal. Simulasi pada peringkat tinggi perlu digunakan dengan efektif untuk mengesan kerosakan atau kesalahan bagi rekabentuk pada peringkat awal dan mengurangkan keperluan bagi simulasi yang lebih terperinci.

3.5 SINTESIS

Sistesis adalah terjemahan bagi penerangan sesuatu rekabentuk bermula dari satu peringkat pengabstrakan sehingga ke peringkat pengabstrakan seterusnya.Ia mungkin penterjemahan kelakuan (behaviour) ke kelakuan atau kelakuan kepada struktur.Proses terjemahan ini adalah sama dengan proses kompilasi bagi bahasa pengaturcaraan bertahap tinggi seperti C kepada kod penghimpunan (assembly code). Input bagi alatan sistesis ini termasuklah penerangan HDL, pemasaan, bahagian yang ingin disisntesis dan perpustakaan teknologi (technology library). Output pula adalah netlist yang dioptima, prestasi yang telah ditafsirkan dan bahagian yang telah disintesis. Dibawah adalah teknik asas bagi sistesis.

i. Sintesis Kelakuan (Behaviour synthesis)

Sintesis kelakuan menterjemah penerangan algoritma seperti –C kepada penerangan RTL. Rekabentuk pada peringkat RTL termasuklah (a) laluan data (data path) (b) memori dan (c) unit kawalan. Rajah 3.3 (a) menerangkan proses sintesis kelakuan yang juga dikenali sebagai sintesis senibina (architecture) atau sintesis peringkat tinggi. Tugas utama adalah mensintesis laluan data,kawalan dan mamori.

ii. Sintesis RTL

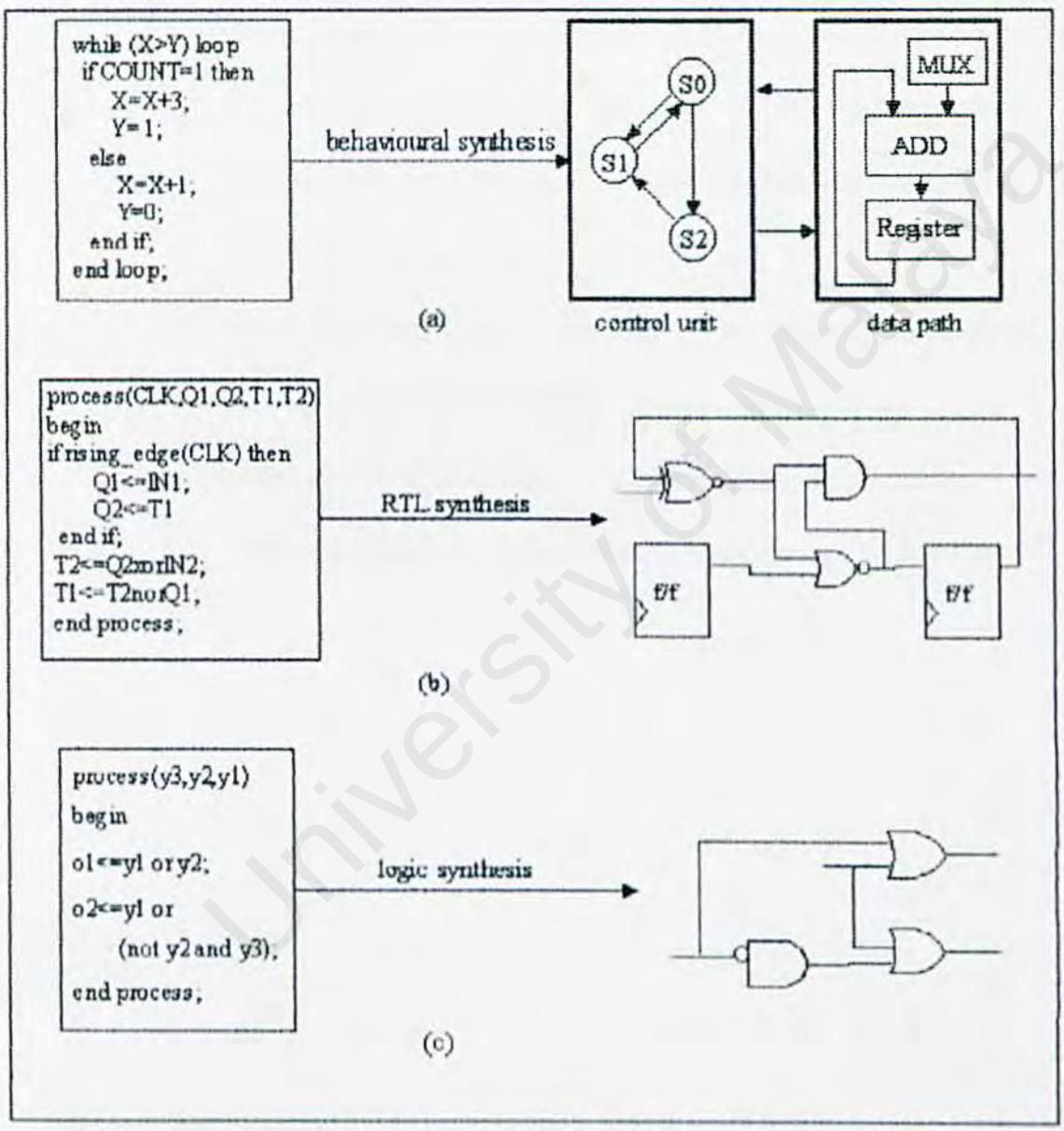
Sintesis RTL menjana struktur netlist dari set fungsi pemindahan pendaftar (register transfer function). Sempadan keadaan (kelakuan clock- ke – clock) diterangkan pada peringkat ini. Operasi pemindahan pendaftar boleh diterangkan sebagai msin keadaan terhad (finite state machine) ,dan ia termasuklah meminimakan keadaan , pengkodan keadaan ,meminimakan logic dan pemetaan teknologi (Rajah 3.3 (b)).

iii. Sistesis logic

Sistesis logic menterjemah ungkapan Boolean kepada logic gabungan.Pengoptimaan bagi sintesis logic dibahagikan kepada dua fasa :

1) Alatan front end yang mengurangkna ketidakbergantungan litar dalam teknologi perpustakaan.

2) Alatan back end yang memetakan struktur netlist kepada penyambungan sel perpustakaan.Rajah 3.3 (c).



Rajah 3.3 : Teknik Simulasi

3.6 KEBOLEHAN VHDL

VHDL dicipta untuk memenuhi beberapa keperluan.

- VHDL boleh menghuraikan struktur bagi sesuatu rekabentuk : pemecahan kepada sub-rekabentuk dan interconnections.
- VHDL menggunakan bentuk pengaturcaraan yang biasa untuk menerangkan fungsi sesuatu rekabentuk.
- VHDL membenarkan sesuatu rekabentuk itu disimulasi sebelum peringkat pengaturcaraan perkakasan dilakukan ke atas sesuatu rekabentuk.
- VHDL boleh digunakan dalam pelbagai metodologi rekabentuk.
- VHDL tidak bergantung kepada teknologi.
- VHDL boleh menerangkan rekabentuk bagi pelbagai perkakasan digital.
- VHDL memudahkan komunikasi antara bahasa piawai.
- VHDL memberi pengurusan rekabentuk yang lebih baik.
- VHDL membenarkan penggunaan pelbagai bahasa rekabentuk.

BAB4
REKABENTUK YANG DICADANGKAN

4.1 PENGKODAN RUN_LENGTH

Pengekodan Run- Length merupakan kaedah pemadatan imej yang berfungsi dengan mengira jumlah piksel yang bersebelahan yang menpunyai nilai grey-level yang sama.Jumlah ini dikenali sebagai run length kemudian dikodkan dan disimpan.

Secara asasnya RLE digunakan untuk imej perduaan tetapi ia juga boleh digunakan imej kompleks yang telah diproses oleh thresholding yang digunakan untuk mengurangkan jumlah grey level kepada dua.Terdapat dua cara untuk mengimplementasi RLE, dan langkah pertama adalah menentukan parameter yang dikehendaki. RLE boleh digunakan secara melintang iaitu kiraan dilakukan sepanjang baris atau secara menegak di mana kiraan dilakukan sepanjang lajur.

Bagi RLE yang digunakan secara melintang , bilangan bit yang digunakan untuk dikodkan bergantung kepada jumlah piksel yang terdapat pada baris tersebut.

Contoh adalah bagi imej perwakilan 8 bit bagi satu piksel :

11110000	11110000	11110000	11110000	11110000	11110000	11110000	11110000
10000000	10000000	10000000	10000000	10000000	10110000	10110000	10110000
10000000	10000000	10000000	10000000	10000000	10110000	10110000	10110000
00000000	00000000	00000000	10000000	10000000	10000000	00000000	00000000
00001010	00001010	00001010	00000000	00000000	00000000	00000000	00000000
00001010	00001010	00001010	10000000	10000000	01110000	01110000	10000000
00001010	00001010	00001010	00001111	00001111	00001111	00000000	00000000
00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000

Rajah 4.1: Imej perwakilan 8 bit.

Imej di dalam rajah adalah bersaiz 8 bait x 8 bait yang mana jumlah keseluruhan bit yang digunakan adalah 512 bit. Selepas proses pengekodan RLE secara melintang, keputusan adalah seperti berikut, nilai yang dipaparkan dalam 8 bit adalah kecerahan (gray level) manakala nilai dalam kurungan merupakan bilangan gray level yang berjujukan, nilai ini juga akan diwakilkan dalam bentuk 8 bit :

Baris pertama : 11110000(8)

Baris kedua : 10000000(5) 10110000(3)

Baris ketiga : 10000000(5) 10110000(3)

Baris keempat : 00000000(3) 10000000(3) 00000000(3)

Baris kelima : 00001010(3) 00000000(5)

Baris keenam : 00001010(3) 10000000(2) 01110000(2) 10000000(1)

Baris ketujuh : 00001010(3) 00001111(3) 00000000(2)

Baris kelapan : 00000000(8)

Algoritma pemampatan RLE digunakan ke atas imej tersebut , bilangan bit yang terhasil dan digunakan untuk mewakili imej tersebut adalah 288 bit sahaja dan ini dapat menjimatkan ruang storan.

Proses nyah – pemadatan memerlukan bilangan piksel dalam baris tersebut dan jenis teknik pengekodan yang digunakan samada secara melintang atau menegak.Proses nyah-pemampatan akan menghasilkan imej asal seperti dalam rajah 6.

RLE paling biasa digunakan untuk memadatkan imej hitam putih atau imej berwarna berindeks 8 bit yang berkemungkinan mempunyai panjang larian (run length).Biasanya RLE tidak digunakan untuk imej berwarna tinggi (high color image)

seperti fotograf yang mana biasanya setiap piksel mungkin berbeza dengan piksel sebelumnya .

Tiga imej berikut dalam rajah 7 menunjukkan imej yang sama tetapi dalam bentuk berbeza, imej pertama mengandungi run dalam setiap baris dan proses pemadatan boleh dilakukan dengan baik.Imej kedua merupakan imej yang sama dengan imej pertama tetapi telah diputar 90 darjah ,tiada run dalam setiap baris dan ini memberikan pemadatan yang tidak baik dan saiz fail yang lebih besar. Jadi imej pertama sesuai untuk dikodkan menggunakan R1.F. secara malintang manakala bagi imej kedua, Pengekodan R1.F. secara menegak mungkin lebih sesuai.Imej ketiga dapat memberi hasil terbaik dalam proses pemadatan kerana keseluruhan imej mempunyai nilai tetap.

Saiz asal: 10000 bait

Saiz selepas pemadatan:

5713 bait

Ratio: 1.75

Saiz asal: 10000 bait

Saiz selepas pemadatan:

10100 bait

Ratio: 0.99

Saiz asal: 10000 bait

Saiz selepas pemadatan:

200 bait

Ratio: 50

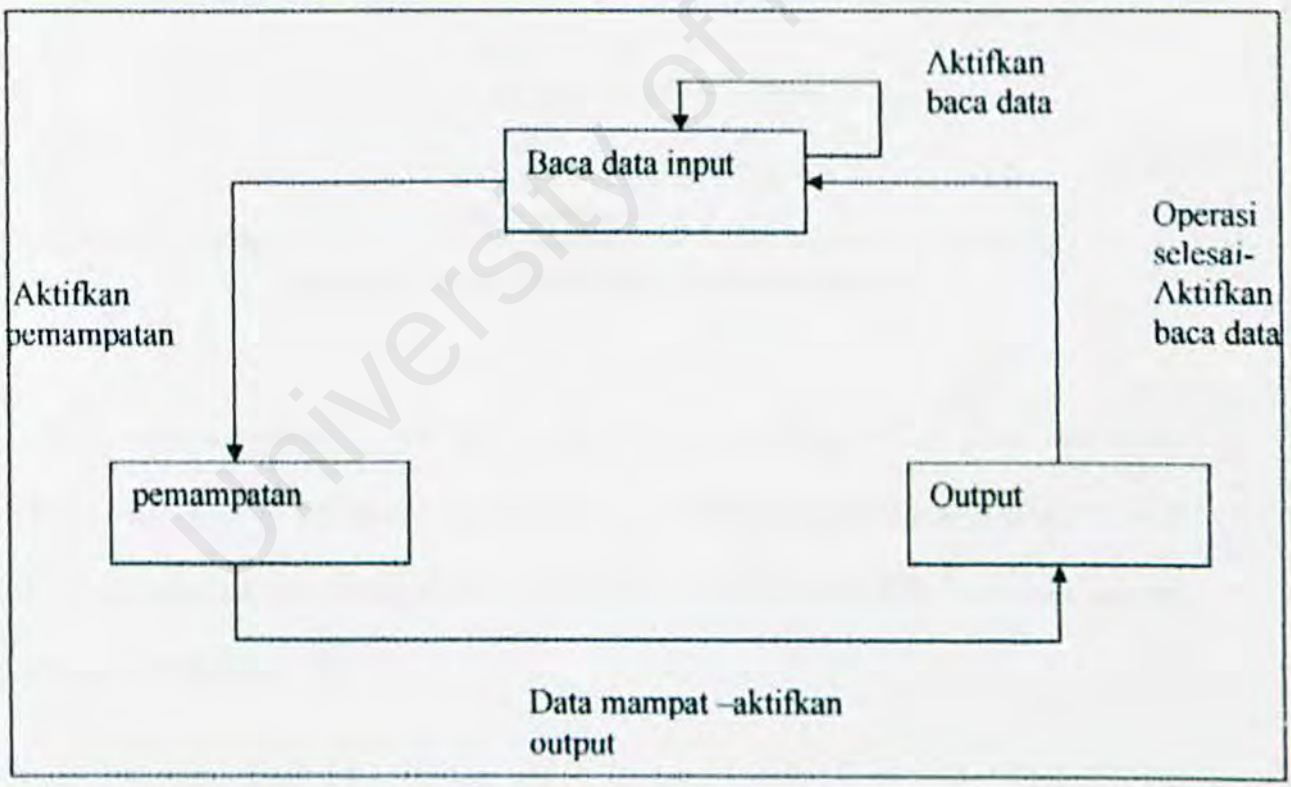
Rajah 4.2 : Contoh bagi imej gray-scale

4.2 REKABENTUK PERKAKASAN YANG DICADANGKAN

Setelah analisis dan kajian dibuat, rekabentuk perkakasan yang akan dihasilkan adalah terdiri daripada dua modul iaitu modul untuk pemampatan dan satu lagi modul untuk nyah-pemampatan. Ini menunjukkan bahawa proses pengekodan dan penyahkodan Run – Length diterangkan secara berasingan.

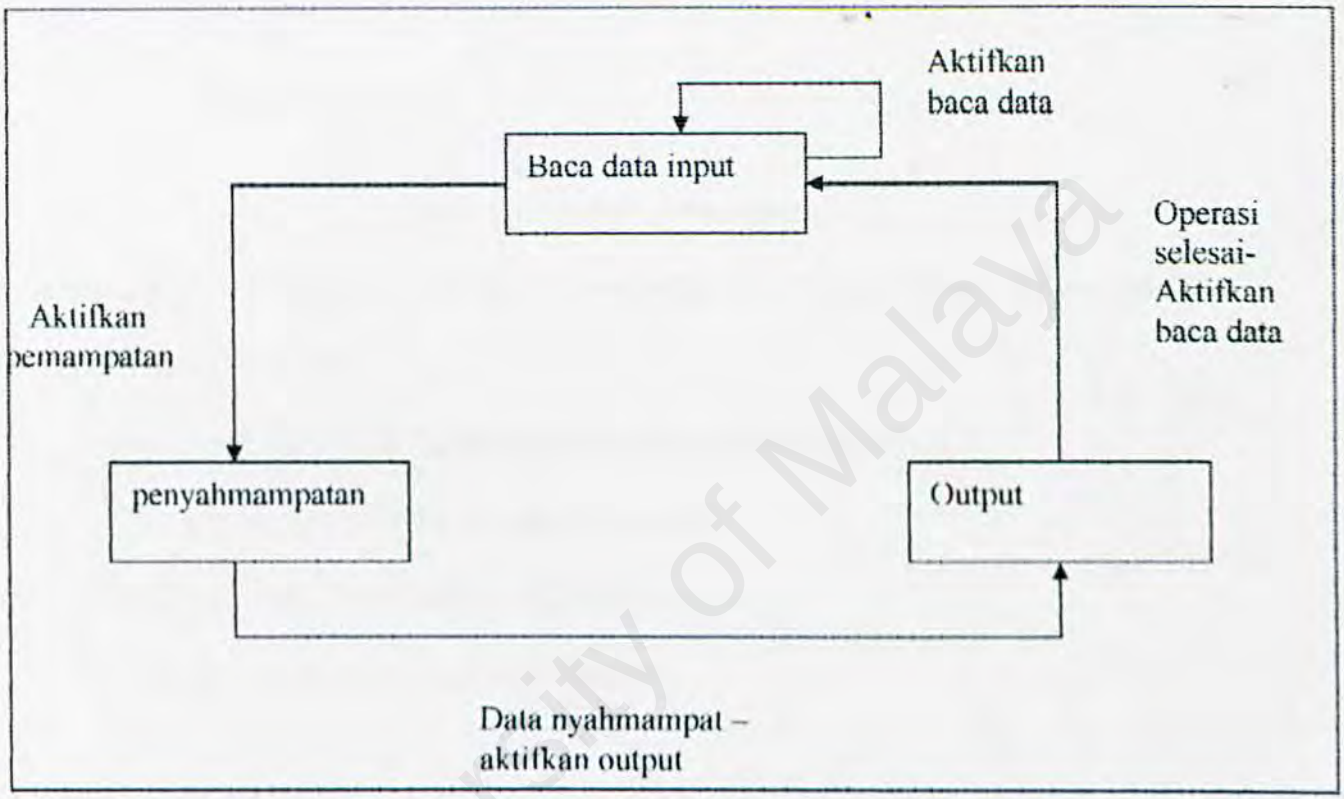
4.2.1 Rajah status rekabentuk perkakasan

Rajah status dicipta bagi memudahkan pemahaman terhadap operasi perkakasan semasa fasa pengkodan dan pengujian.Setiap modul diteliti operasinya daripada modul utama hingga modul lain.Rajah 8 menunjukkan rajah status bagi pemampatan manakala Rajah 9 menunjukkan rajah status bagi nyah-pemampatan.



Rajah 4.3 : Gambarajah status untuk pemampatan

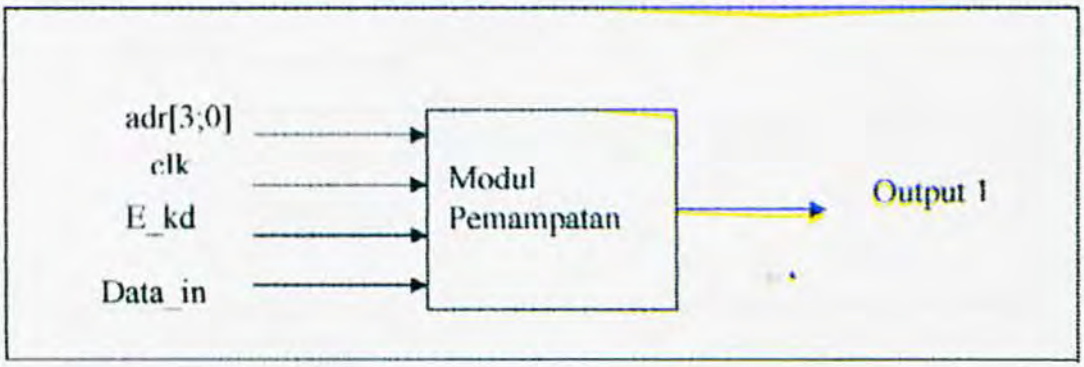
Setelah modul kawalan hantar isyarat e_kd ,modul pemampat akan bersedia untuk memampatkan data yang masuk.Arahan baca data akan diaktifkan dan setelah itu,jujukan input akan dibaca dan dimampatkan. Kemudian hasil mampat akan keluar.



Rajah 4.4 : Gambarajah status nyahmampatan

Setelah menerima arahan e_nkd dari modul kawalan pusat, arahan baca data akan diaktifkan dan jujukan data yang telah dibaca akan bersedia untuk dinyahmampat.Setelah proses nyahmampat selesai,output akan dihasilkan dan baca data kan diaktifkan semula untuk data berikutnya.

4.2.2 Modul pemampatan



Rajah 4.5 : Modul pemampatan

Modul pemampatan adalah modul yang menjalankan proses pemampatan .Berikut adalah port input dan port output.

Port input : adr [3: 0] : mengawal jujukan pengulangan semula.

Clk : menyediakan modul dengan jam sistem.

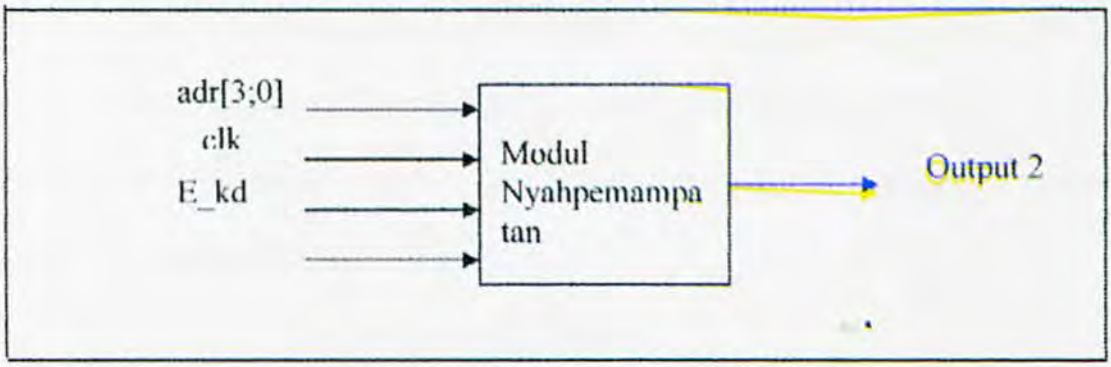
E_kd : mengaktifkan modul pemampatan.

Data_in : jujukan data yang ingin dimampat.

Port output

Data_out : jujukan data yang telah dimampat.

4.2.3 Modul penyahmampatan



Rajah 4.6: Modul nyah pemampatan

Modul nyahpemampatan adalah modul yang menjalankan proses nyah mampatan .Berikut adalah port input dan port output.

Port input : adr [3: 0] : mengawal jujukan pengulangan semula.

Clk : menyediakan modul dengan jam sistem.

E_kd : mengaktifkan modul nyah pemampatan.

Data_in : jujukan data yang ingin dinyahmampat.

Port output

Data_out : jujukan data yang telah dinyahmampat.

4.3 Gambarajah blok bagi pengekodan dan penyahkodan Run-Length.

Rajah 12 dan 13 menunjukkan gambarajah blok bagi pengkodan dan penyahkodan Run-Length. Gambarajah ini menerangkan proses yang berlaku di dalam modul bagi pemampatan dan pemampatan. Rekabentuk bagi Pengekodan Run-Length mengandungi komponen digital berikut :

- DATA REGISTERS (Pendaftar data imej)
- THRESHOLD REGISTER (pendaftar threshold)
- DOWN COUNTER (Pembilang)
- MULTIPLEXER (Multiplexer)
- DEMULTIPLEXER

Penyataan input/output dan fungsi bagi setiap komponen adalah seperti berikut.

DATA REGISTERS:

REGISTER B: Merupakan REGISTER 8 bit yang mengambil datain 8 bit input pada rising edge bagi clock dan apabila signal LDx tinggi.Output B bagi register juga 8 bit. Apabila LDx ditentukan nilai dimasukkan pada clock berikutnya

REGISTER C: Merupakan register 8 bit yang mengambil 8 bit input B pada pada rising edge clock dan bila signal LDy tinggi.Output bagi C bagi register juga 8 bit. Apabila LDy ditentukan nilai dimasukkan pada clock berikutnya

COUNTER REGISTER : Merupakan register 8 bit yang mengambil 8 bit dari counter pada rising edge clock. Output juga adalah 8 bit.

THRESHOLD REGISTER: Biasanya digunakan untuk selak nilai threshold.Bila LDT ditentukan nilai bagi input dimasukkan ke clock berikutnya.

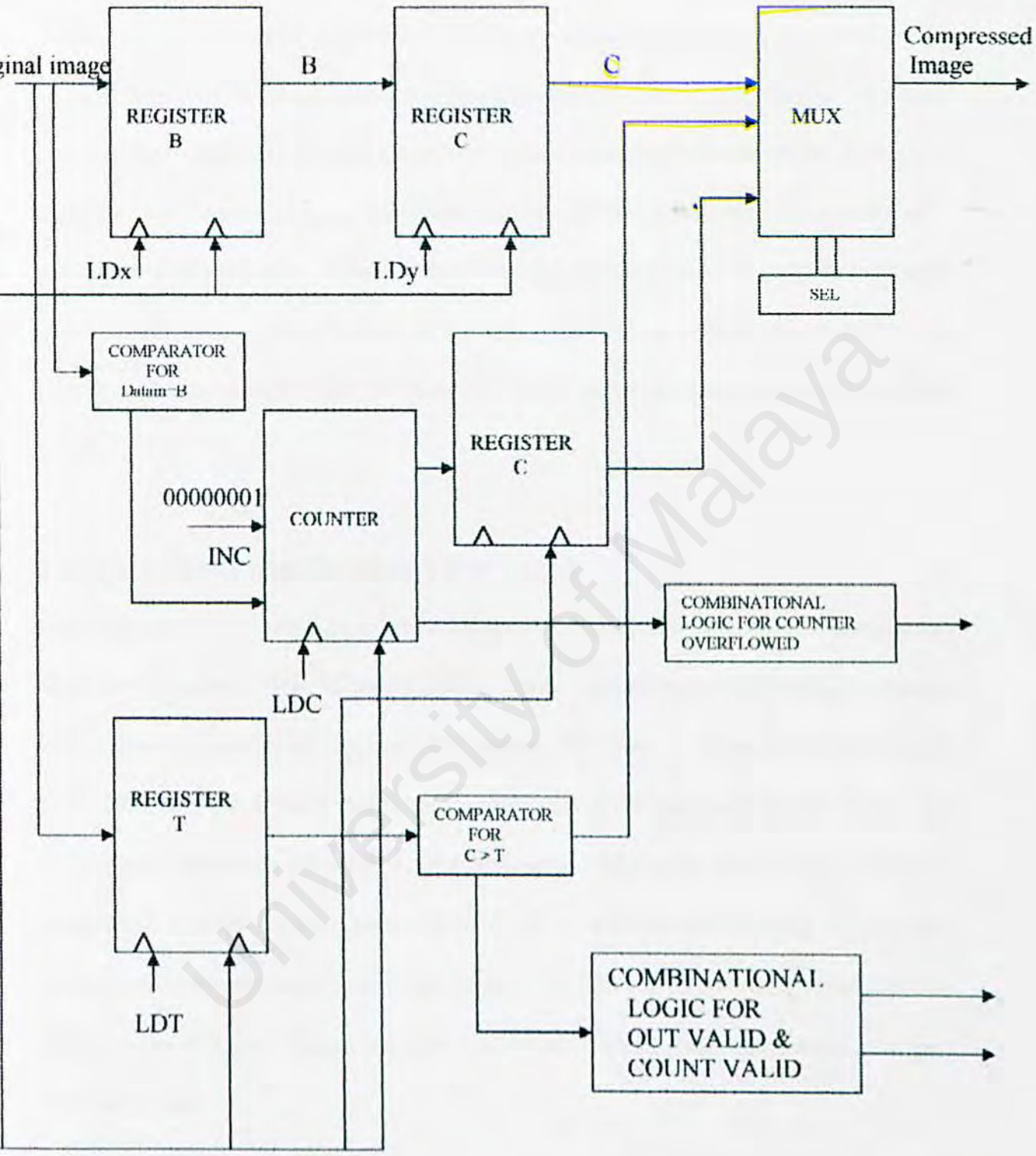
COUNTER: Pembilang up-down 8 bit.Pembilang ditentukan semula dengan memasukkan nilai 1 setiap kali signal Cntrst ditentukan.nilai bagi pembilang meningkat dan menurun bergantung kepada signal INC dan DEC.

MULTIPLEXER: Merupakan multiplekser 3 input dan satu output yang memilih baris yang sesuai bergantung kepada nilai yang diwakilkan pada signal yang dipilih.Multiplexer mempunyai input ,iaitu daripada 2 register dan output daripada pembilang.

4.2.5 Signal dalam rekabentuk

- **LDx** Datain dimasukkan ke register B pada clock berikutnya.
- **LDy** Nilai B dimasukkan ke dalam register kedua hanya sekiranya LDy adalah 1
- **LDT** nilai T dimasukkan ke register threshold hanya sekiranya LDT adalah 1.
- **INC** Peningkatan nilai pembilang .
- **SEL** Pilih signal ke multiplekser. Pilih data sekiranya "00" , "01" , "10"
- **T** nilai threshold 6 bit.
- **T1** nilai Threshold ditukar ke 8 bit.
- **B** output daripada register pertama.

C Current State Signal digunakan untuk kod logic keadaan sebelum dan selepas.



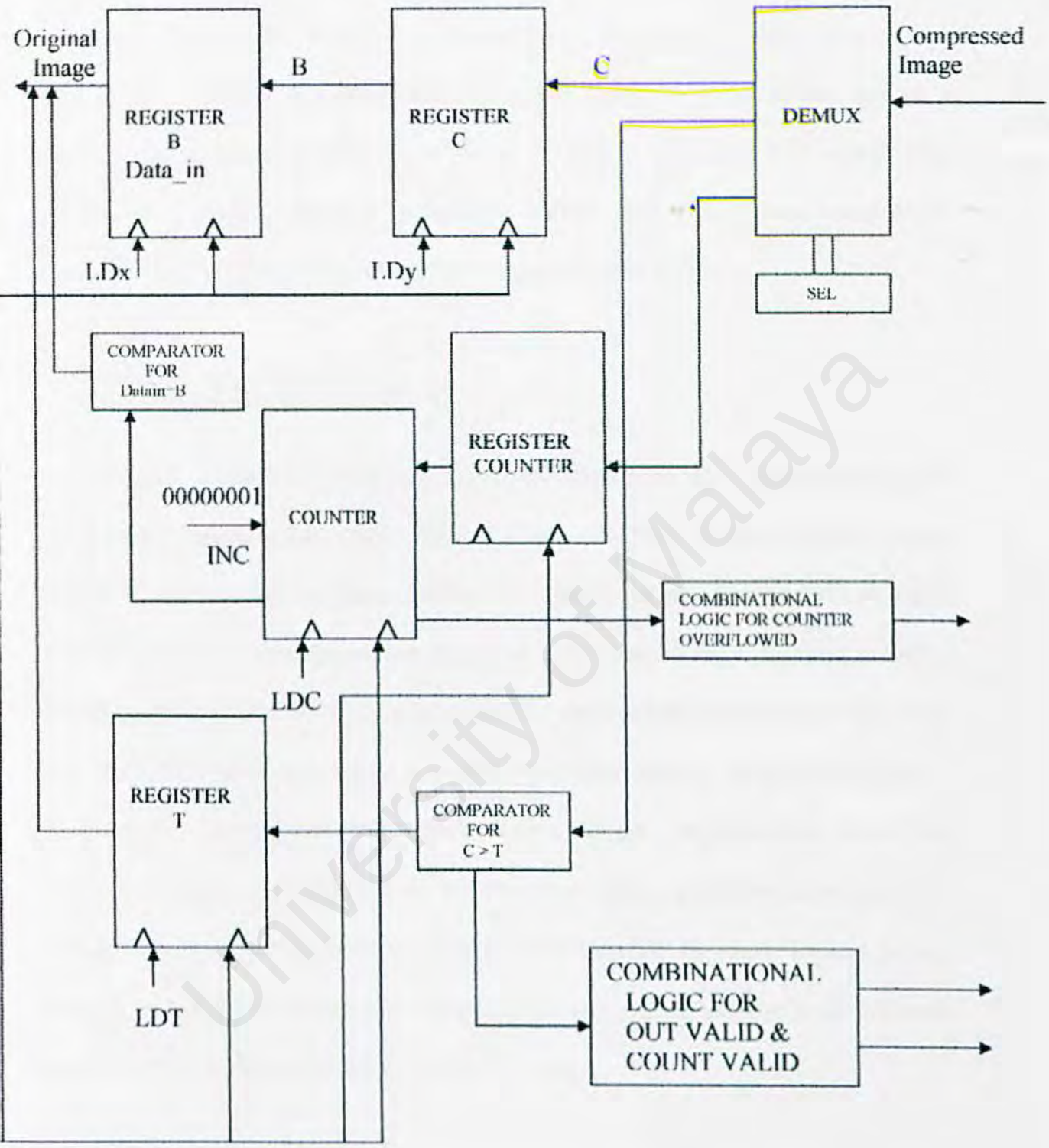
Rajah 4.7 : Gambarajah Blok bagi pengekodan Run-Length

4.3.1 Penerangan proses pengekodan Run-Length

Input ,Original Image adalah dalam bentuk 8 bit bagi mewakili satu piksel manakala output, Compressed image juga adalah 8 bit yang mewakili bilangan jujukan piksel yang bersebelahan yang telah dikodkan menggunakan teknik Run-Length. Output ini adalah bentuk image yang telah dimampatkan.Sekiranya jujukan bagi sesuatu piksel itu berlaku melebihi nilai 'threshold' yang ditetapkan, turutan ini akan digantikan dengan satu nilai piksel dan kemudian akan diikuti dengan bilangan jujukan piksel,C yang memepunyai nilai gray-level yang sama. Sekiranya bilangan jujukan bagi sesuatu piksel itu berlaku kurang atau sama dengan nilai threshold,T , maka outputnya sama dengan piksel yang masuk.

4.3.2 Penerangan Proses Pengekodan Run-Length

Piksel pertama akan masuk ke dalam Register B.Satu lagi nilai threshold ,T dimasukkan ke dalam Register 'I'.Apabila piksel kedua masuk , piksel tersebut dinyatakan sebagai data_in manakala piksel pertama tadi dimasukkan ke Register B. Kemudian dua piksel ini akan dibandingkan samada mempunyai nilai gray-level yang sama atau tidak oleh comparator . Sekiranya nilai gray-level adalah sama , nilai pada counter akan meningkat sebanyak 1. Operasi ini akan berulang sehingga nilai piksel terbaru yang diinput tidak mempunyai nilai gray-level yang sama dengan sebelumnya. Output yang terhasil adalah nilai grey-level diikuti dengan bilangan jujukan bagi piksel yang mempunyai nilai grey level yang sama.



Rajah 4.8 : Gambarajah blok bagi penyahkodan Run-Length

4.3.3 Penerangan proses penyahkodan Run-Length

Gambarajah blok bagi penyahkodan Run-Length merupakan proses pembalikan bagi pengkodan Run-Length. Komponen digital yang digunakan adalah sama ,tetapi multiplexer ditukarkan kepada demultiplekxer.Imej input yang masuk kedalam blok diagram adalah imej yang telah dimampatkan dan output yang dihasilkan adalah imej asal.Imej asal yang dikeluarkan mempunyai bentuk yang sama seperti sebelum ia diproses kerana Run-Length melakukan teknik pemampatan 'lossless'.

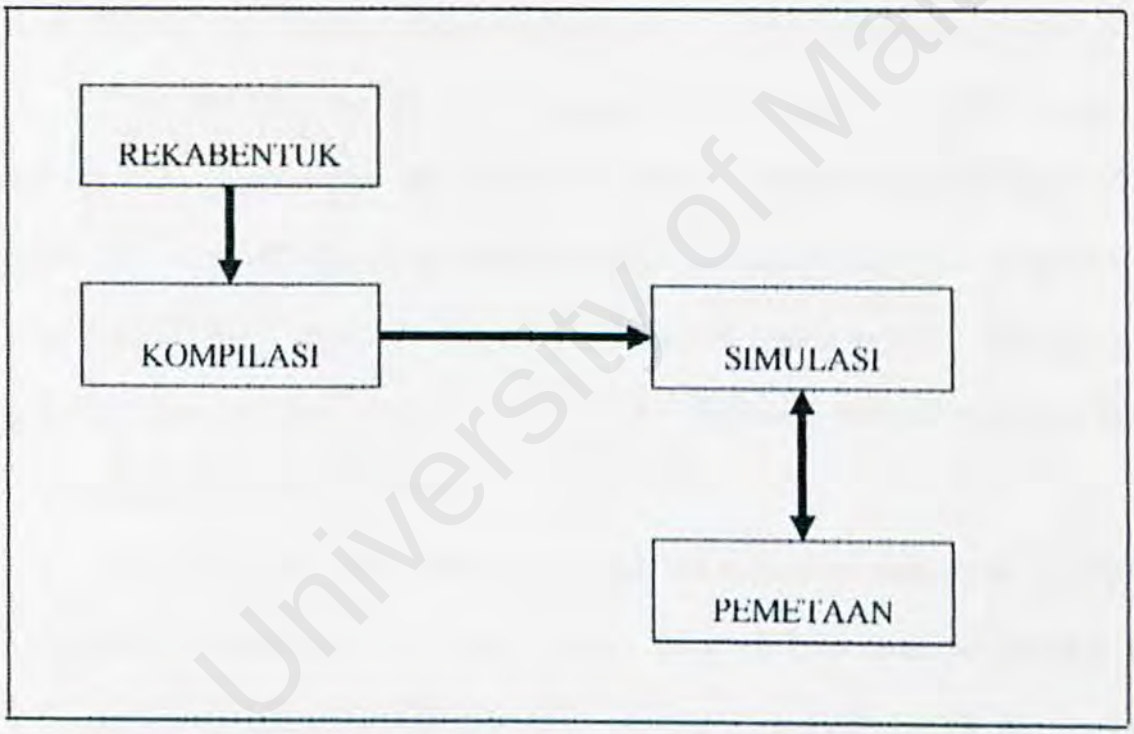
4.4 TEXT INPUT OUTPUT (Text I/O)

Semasa proses pengujian , teknik yang digunakan untuk input data adalah teknik TEXT I/O. Text I/O membolehkan VHDL membuka satu atau lebih fail data, membaca baris-baris dalam fail tersebut dan menghuraikan baris tersebut untuk membentuk elemen data tertentu seperti elemen dalam jujukan atau rekod.Untuk menyokong kegunaan file,VHDL mempunyai konsep file data type, ia termasuklah piawai ,fungsi built-in untuk buka, baca dan tulis ke file data type. Pakej text I/O , yang mengandungi perpustakaan piawai, dikembangkan kepada jenis file built-in dengan dengan mengformatkan fungsi.Test bench membaca baris daripada fail dan menggunakan data yang terdapat pada setiap baris sebagai vector ujian untuk stimulasi.Test bench membaca teks fail secara dinamik semasa simulasi, jadi test bench tidak perlu dikompil semula apabila stimulus ujian ditambah atau diubahsuai.Ini bak untuk rekabentuk yang besar.

BAB 5
IMPLEMENTASI

5.1 PENGENALAN

Selepas proses rekabentuk dipenuhi,fasa implementasi ke atas pemampatan imej menggunakan teknik pengekodan run-length boleh dimulakan. Rajah 5.1 menunjukkan proses rekabentuk yang diringkaskan yang mengandungi kedua-dua proses kompilasi dan simulasi untuk satu atau lebih rekabentuk pengaturcaraan logic. Kunci bagi memahami proses ini dan memahami kelebihan menggunakan bahasa pengaturcaraan VHDL adalah dengan mengingati kepentingan pembangunan pengujian(test development). Pembangunan pengujian boleh dilakukan selagi keperluan umum bagi system adalah diketahui.



Rajah 5.1 : Langkah – langkah dalam mengimplimentasi modul VHDL

VHDL (sepanjang dengan entity-entiti, contohnya seperti gambarajah blok dan rajah berskima) adalah digunakan untuk merekabentuk .Selepas itu, rekabentuk menggunakan satu editor teks (atau melalui satu alat rekabentuk di mana VHDL dihasilkan daripada paparan bergambar aras tinggi).Kod sumber VHDL mesti boleh dihantar secara terus kepada alat sintesis untuk diimplementasi ke dalam peranti yang dispesifikkan .Kemudian input mengalami proses simulasi iaitu membenarkan bahawa kefungsian sesuatu modul itu disahkan.

Pada bahagian pembangunan ujian,fail skrip atau VHDL 'testbench' dihasilkan bagi melakukan latihan litar untuk sahkan kefungsiannya dan spesifikasi pemasaan yang telah ditetapkan.Fail skrip ini boleh boleh dimuatkan menggunakan editor teks ,atau boleh dijanakan daripada maklumat ujian stimulus seperti gelombang bergambar.

Fasa ini juga menunjukkan implementasi ke atas kod VHDL yang telah dibangunkan menggunakan perisian PEAK FPGA SYNTHESIS EDITION Version 5.20c. Kod yang telah dibangunkan akan dikompil untuk mengesan ralat .Sekiranya tiada ralat dikesan ,maka proses yang berikut nya adalah proses simulasi. Fasa ini adalah berkenaan pemetaan kod kepada gambarajah blok.Sekiranya terdapat ralat pada fasa ini ,kod ini akan dibangunkan semula.

Proses simulasi satu pendekatan saling tak bersandar bagi setiap modul telah dijalankan dimana setiap entity yang wujud disimulasikan untuk mengetahui tahap ketepatan pemprosesan yang dijalankan berbanding dengan kitaran pengaturcaraan sendiri.Proses ini dilaksanakan setelah melakukan satu ujian pengaturcaraan (testbench) ke atas kod VHDL yang diaturcara bagi melihat output yang diperolehi dan setiap keputusan akan dipaparkan.

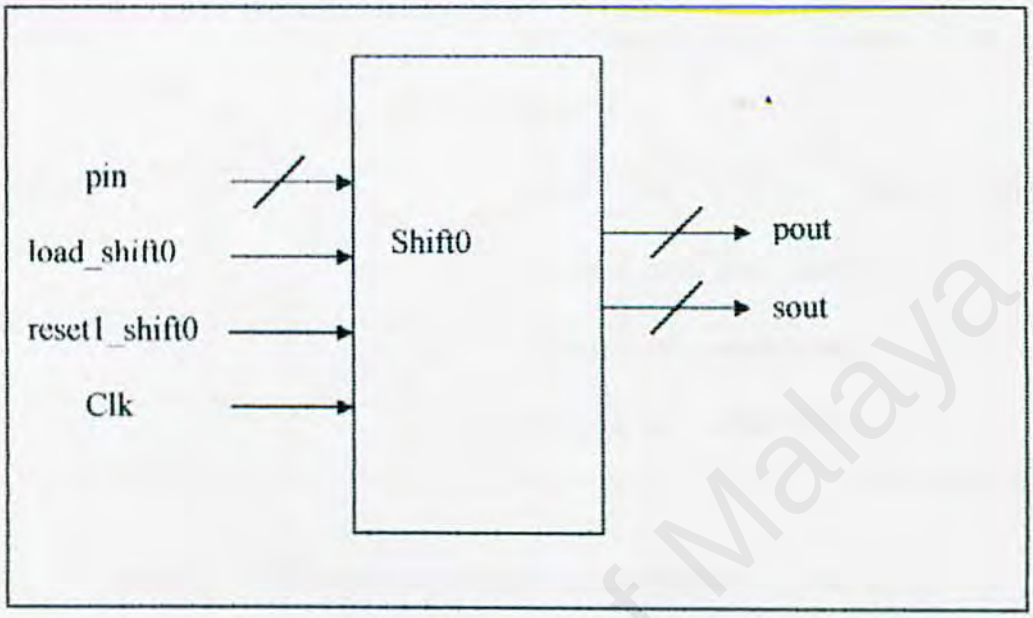
5.2 IMPLEMENTASI SETIAP MODUL

Dalam proses pembangunan pengekodan menggunakan teknik run-length,terdapat beberapa modul telah direkabentuk bagi memastikan ketepatan kefungsiannya.Modul-modul tersebut adalah

- Modul shift0.vhd
- Modul shift1.vhd
- Modul compare.vhd.
- Modul counter.vhd
- Modul mux.vhd
- Modul countreg.vhd
- Modul control.vhd
- Modul rle.vhd (modul top level).

Penerangan tentang setiap modul akan diterangkan di bawah.

5.2.1 MODUL SHIFT0



Rajah 5.2 : Gambarajah blok bagi Shift0

Dalam modul ini ,data yang masuk melalui PIN adalah bersaiz 8 bit yang mana bit-bit tersebut akan dikeluarkan dalam bentuk jujukan 8 bit melalui POUT dan SOUT. Keluaran POUT ini akan memasuki COMPARE LOGIC untuk dibandingkan manakala keluaran SOUT akan memasuki register kedua iaitu shift1. Fungsi bagi setiap pin diterangkan pada jadual penerangan pin pada jadual ()

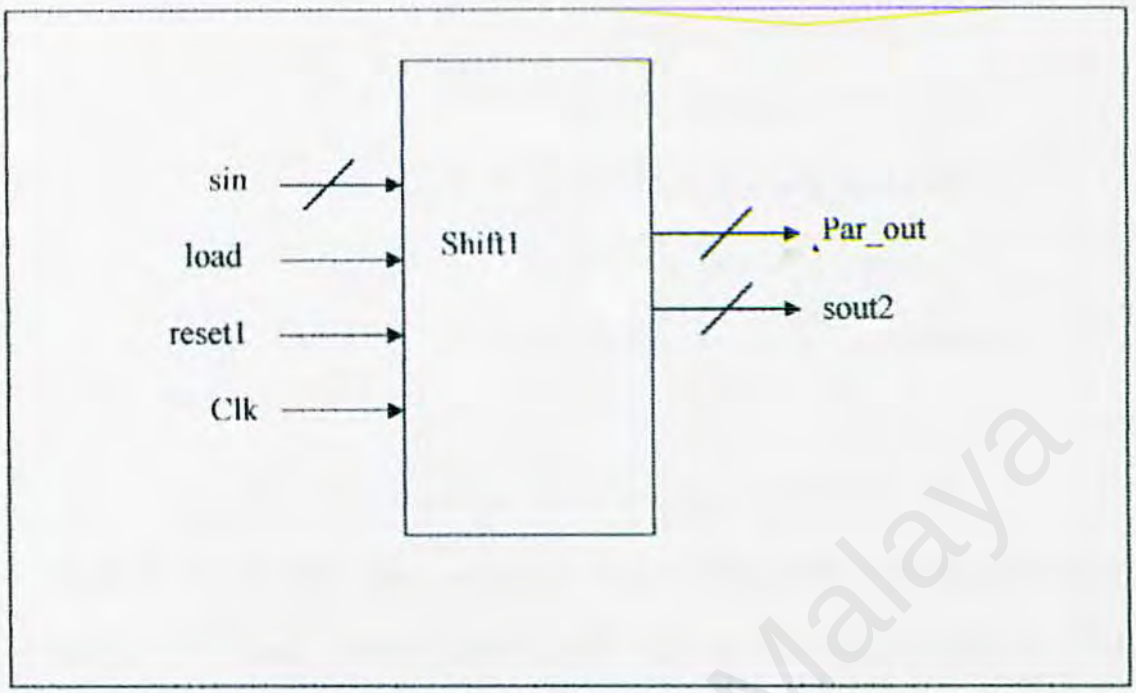
PIN	PENERANGAN
CLK	Kawalan pemasaan
PIN	Masukan bersaiz 8 bit
Reset1_shift0	Input dari keluaran control. Reset modul pada '1'
Load_shift0	Input dari keluaran control.Keluarkan keluaran pada load_shift0 '0'
Pout	Keluaran ke modul compare logic
Sout	Keluaran ke modul shift1

Jadual 5.1 : Penerangan pin input dan output bagi modul shift0

Kod aturcara di bawah merupakan architecture bagi modul shift0 berfungsi seperti yang dinyatakan di atas.Kod sumber keseluruhan bagi modul shift0 dimuatkan dalam appendik :

```
architecture structural of shift0 is
    begin
        process (clk, reset1_shift0)
            begin
                if (reset1_shift0 = '1') then -- dalam keadaan 'high state'
                pout <= "00000000";
                sout <= "00000000";
                elsif (clk'event and clk ='1') then -- dinilai jika nilai reset bukan 1
                                                    -- untuk kesan 'clockedges'
                    if (load_shift0 = '0') then
                        pout <= pin;
                        sout <= pin;
                end if;
```

5.2.2 MODUL SHIFT1



Rajah 5.3 : Gambarajah blok bagi modul shift1

Modul ini merupakan register kedua didalam rekabentuk untuk run-length. Ia akan mengambil SIN sebagai masukan yang bersaiz 8 bit dari keluaran SOUT pada shift0. Ia akan mengeluarkan 2 keluaran bersaiz 8 bit iaitu Par_out dan Sout2. Keluaran Par_out akan ke modul multiplexer manakala keluaran Sout2 akan ke COMPARE LOGIC.

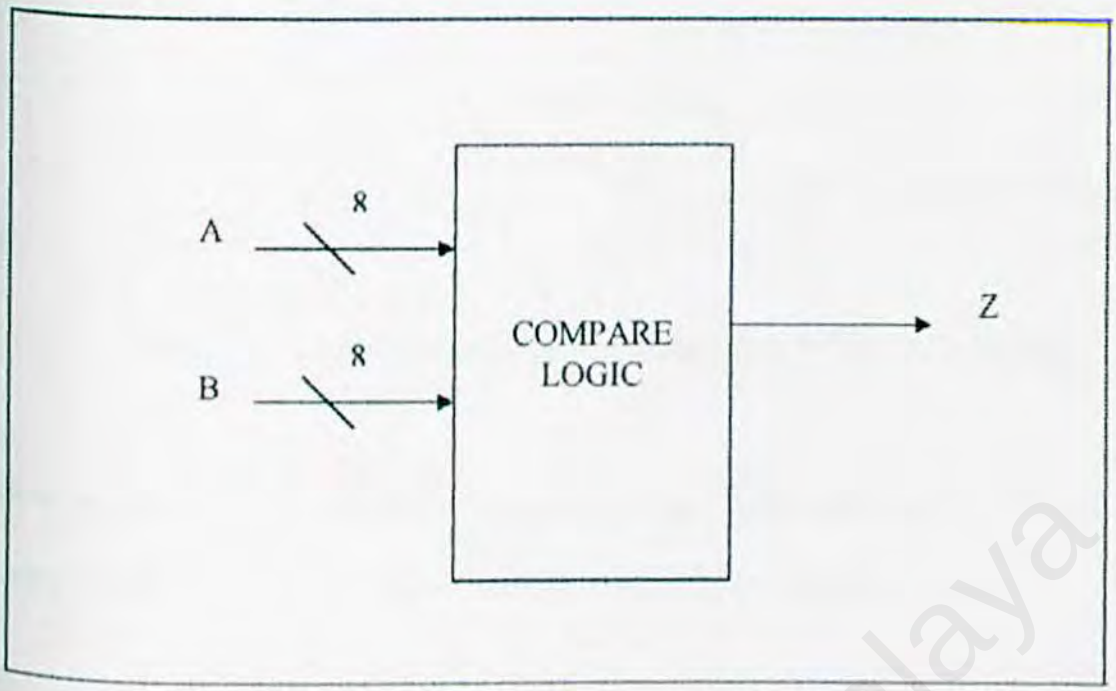
PIN	PENERANGAN
CLK	Kawalan pemasaan
SIN	Masukan bersaiz 8 bit dari shift0
Reset1	Reset modul pada '1'
Load	Keluarkan keluaran pada load '0'
Par_out	Keluaran ke modul multiplexer
Sout2	Keluaran ke modul compare logic

Jadual 5.2 : Penerangan pin input dan output bagi modul shift1

Kod aturcara di bawah merupakan architecture bagi modul shift1 berfungsi seperti yang dinyatakan di atas. Kod sumber keseluruhan bagi modul shift1 dimuatkan dalam appendik :

```
architecture structural of shift1 is
    begin
        process (clk, reset1)
            begin
                if (reset1 = '1') then -- dalam keadaan 'high state'
                    par_out <= "00000000";
                    sout2 <= "00000000";
                    elsif (clk'event and clk ='1') then -- dinilai jika nilai reset bukan 1
                                                        -- untuk kesan 'clockedges'
                        if (load = '0') then
                            par_out <= sin;
                            sout2 <= sin;
                    end if;
```

5.2.3 MODUL COMPARE LOGIC



Rajah 5.4 : Gambarajah blok bagi modul compare logic

Modul Compare Logic ini digunakan untuk membandingkan output yang keluar daripada dua register iaitu shift0 dan shift1. Modul ini akan mengambil 2 input 8 bit iaitu A daripada keluaran shift0 (POUT) dan B daripada shift1(Par out) untuk dibandingkan. Manakala output Z adalah keluaran yang menentukan samada kedua-dua input A dan B adalah sama atau tidak.Sekiranya A dan B mempunyai nilai yang sama, Z akan menghasilkan keluaran '1' manakala sekiranya berlainan keluaran '0' akan dikeluarkan.

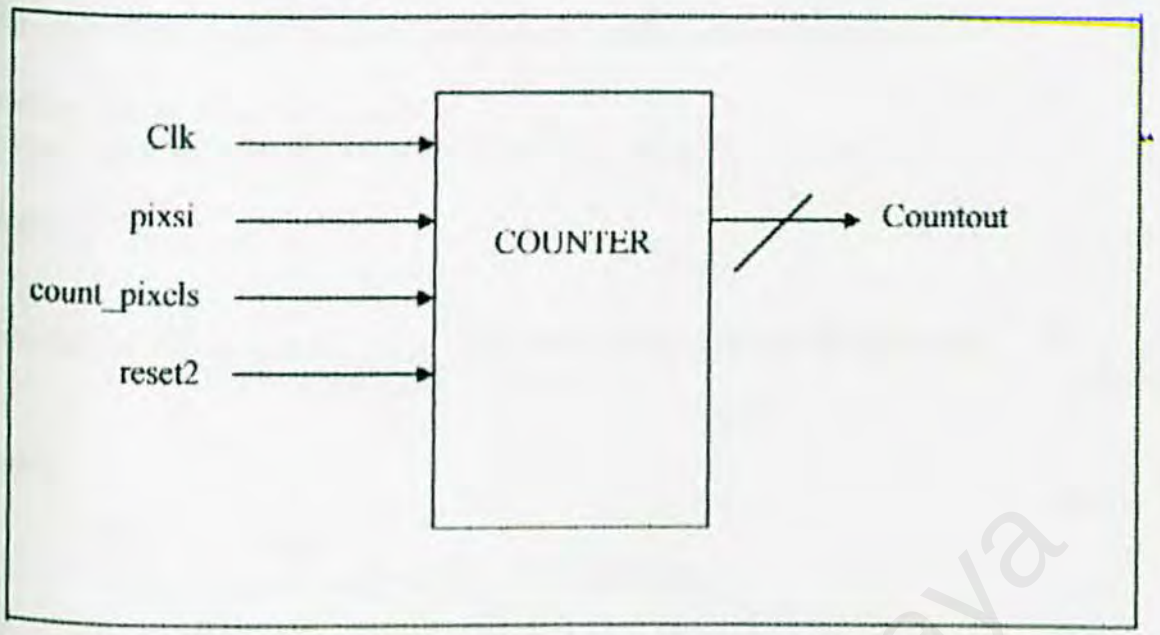
PIN	PENERANGAN
A	Masukan 8 bit dari shift0
B	Masukan 8 bit dari shift1
Z	Isyarat keluaran '1' sekiranya sama dan '0' Jika berlainan.

Jadual 5.3 : Penerangan pin input dan output bagi modul compare logic

Kod aturcara di bawah merupakan architecture bagi modul multiplexer berfungsi seperti yang dinyatakan di atas.Kod suber keseluruhan diletakkan dalam appendik :

```
architecture structural of compare is
        begin
            z <= '1' when (A = B) else '0';
end structural;
```

5.2.4 MODUL COUNTER



Rajah 5.5 : Gambarajah blok bagi modul counter

Modul counter ini berfungsi untuk mengira jujukan piksel yang mempunyai nilai yang sama. Masukan reset2 , pixsi, dan count pixels adalah keluaran dari modul controller yang mana gabungan input ini akan menghasilkan fungsi seperti berikut :

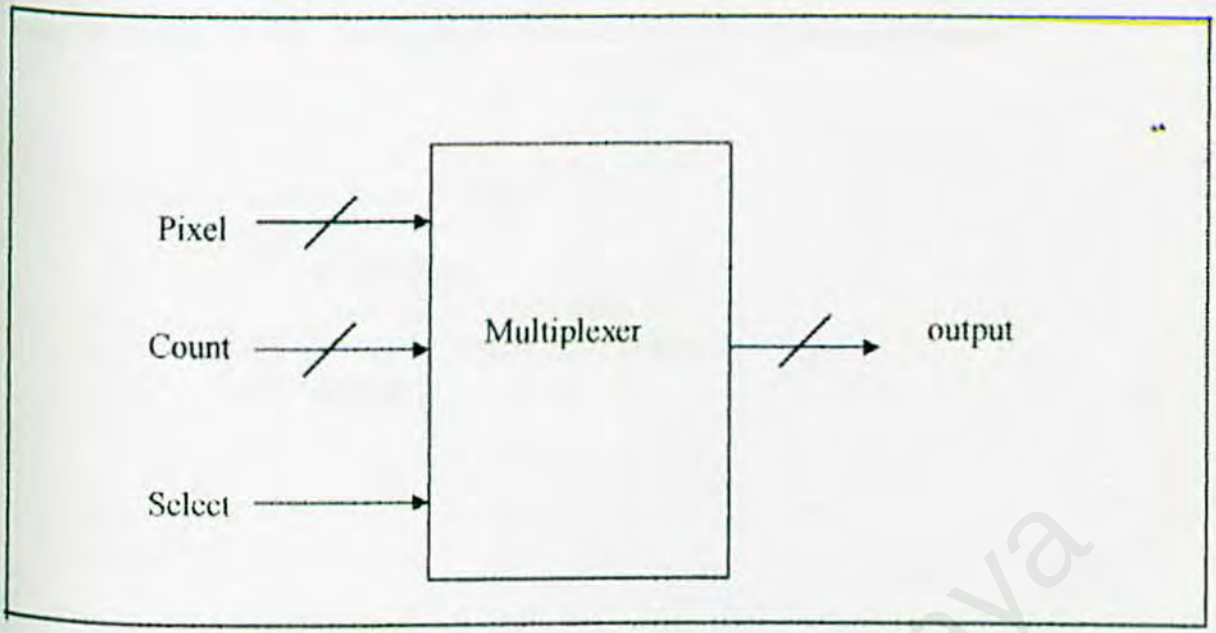
PIN	PENERANGAN
CLK	Kawalan pemasaan
Reset2	Counter akan direset semula kepada 00000000 bila reset2 '0'
Pixsi, count_pixels = 1,0	Piksel akan diset kepada 00000001
Pixsi, count_pixels = 1,1	Counter akan bertambah dengan 1
Pixsi, count_pixels = 0,0 Pixsi, count_pixels = 0,1	Counter memegang nilai sebelumnya Nilai counter akan dikeluarkan
countout	Keluaran

Jadual 5.4 : Penerangan pin bagi modul counter.

Kod aturcara di bawah merupakan architecture bagi modul counter berfungsi seperti yang dinyatakan di atas.Kod sumber keseruhan dimuatkan dalam appendik:

```
architecture structural of counter is
signal Temp_Counter_Value_Pixels : std_logic_vector(7 downto 0);

begin


counter: process (RESET2,CLK)   -- Counter for the number of pixels that
                                -- have been received.

begin

      if RESET2 = '0' then
       Temp_Counter_Value_Pixels <= "00000000";

      elsif CLK'event and CLK = '1' then
       if (PIXSI = '1')
       then
         if Count_Pixels = '0' then
           Temp_Counter_Value_Pixels <= "00000001";
         else
           Temp_Counter_Value_Pixels <= Temp_Counter_Value_Pixels +"00000001";
         end if;
       else
         if Count  Pixels = '0' then
           Temp_Counter_Value_Pixels <= temp_counter_value_pixels;
         end if;
       end if;
      end if;
  end process;
      Countout <= Temp_Counter_Value_Pixels;

end structural;
```

5.2.5 MODUL MULTIPLEXER



Rajah 5.6 : Gambarajah blok bagi modul multiplexer

Modul multiplexer ini digunakan untuk memegang input untuk sementara waktu. Output akan dikeluarkan apabila menerima arahan 'select' daripada modul control. Multiplekser yang digunakan akan mengambil dua input yang bersaiz 8 bit iaitu 'pixel' dan 'count' dan akan mengeluarkan 'output' bersaiz 8 bit iaitu samada daripada 'pixel' atau 'count' bergantung kepada isyarat input 'select'.

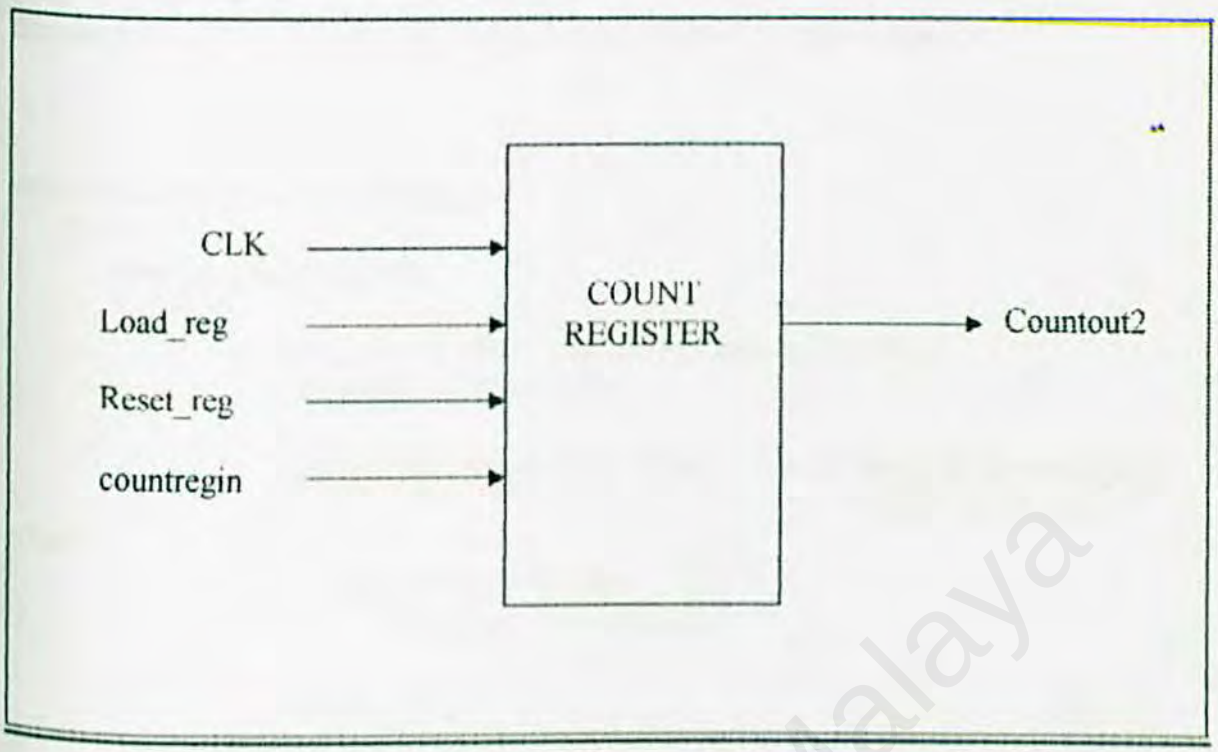
Pin	Penerangan
Pixel	Masukan yang terhasil dari keluaran modul shift1
Count	Masukan yang terhasil daripada modul count
Select	'0' untuk keluarkan 'pixel' '1' untuk keluarkan 'count'
Output	Keluaran pixel atau count berdasarkan isyarat dari 'select'

Jadual 5.5 : Penerangan input output bagi modul multiplexer

Kod aturcara di bawah merupakan architecture bagi modul multiplexer berfungsi seperti yang dinyatakan di atas.Kod sumber keseruhan diletakkan dalam appendik:

```
architecture structural of mux is
        begin
            with sel select
                output <= pixel when '0',
                          count when others ;
        end structural ;
```

5.2.6 MODUL COUNT REGISTER



Rajah 5.7 : Gambarah blok bagi modul count register

Modul ini digunakan untuk menyimpan nilai 'countout' yang dikeluarkan daripada modul counter. Jadual menunjukkan penerangan pin bagi modul count register.

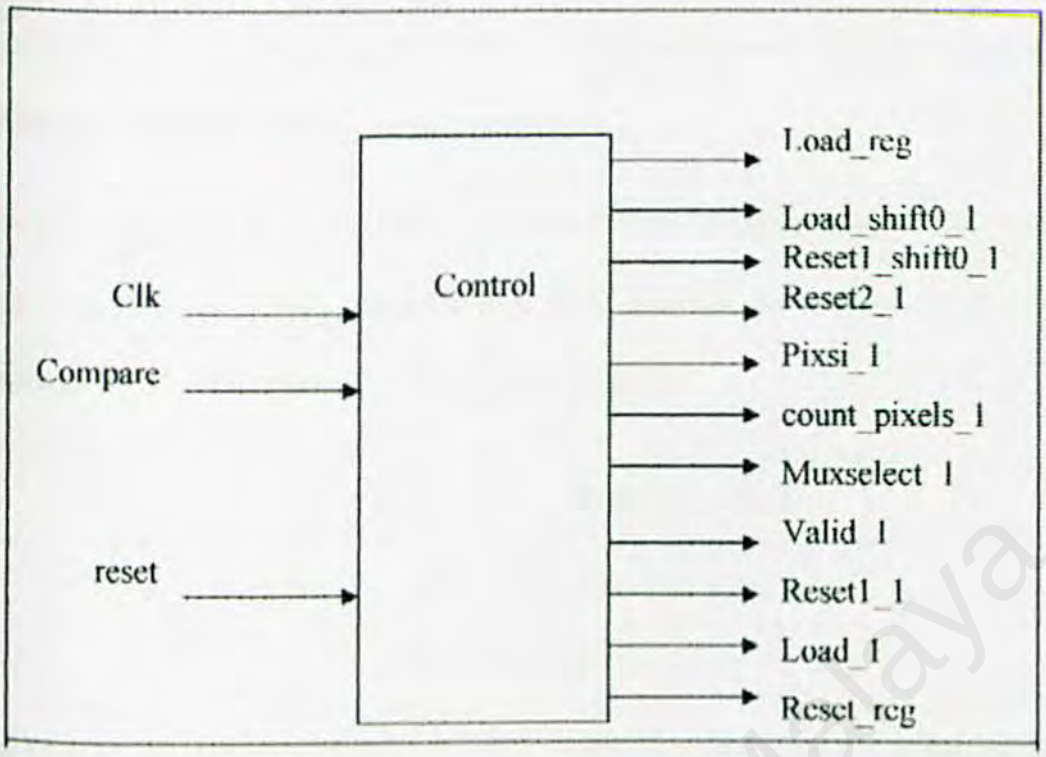
PIN	PENERANGAN
Clk	Kawalan pemasaan
Load_reg	Masukan daripada modul kawalan. Keluarkan ouput pada Load '0'
Reset_reg	Masukan daripada modul kawalan. Reset modul pada '1'
Countout	Keluaran ke multiplexer

Jadual 5.6: Penerangan input dan output bagi modul count register

Kod aturcara di bawah menunjukkan architecture yang dapat berfungsi seperti yang dikatakan di atas. Kod aturcara keseluruhan diletakkan di dalam appendik:

```
architecture structural of countreg is
    begin
        process (clk, reset_reg)
            begin
                if (reset_reg = '1') then -- dalam keadaan 'high state'
                    countout2 <= "00000000";

                elsif (clk'event and clk ='1') then -- dinilai jika nilai reset bukan 1
                                                    -- untuk kesan 'clock
edges'
                        if (load_reg = '0') then
                            countout2 <= countregin;

                    end if;
            end if;
        end process;
    end structural;
```

5.2.7 MODUL CONTROL



Rajah 5.8 : Gambarajah blok bagi control

Modul control merupakan modul yang digunakan untuk mengawal modul-modul lain yang ada dengan mengeluarkan isyarat kawalan kepada modul-modul tersebut.Modul kawalan ini dihasilkan berdasarkan kepada gambarajah keadaan di bawah :

State 1 : 8 bit pertama input dimuatkan ke dalam register pertama. Counter disetkan pada 1 dan tiada keluaran sistem dijanakan.

State 2 : Counter direset pada 1 dan register masih berada pada keadaan asal.tiada sebarang output dihasilkan.

State 3 : Jujukan 8 bit kedua dimuatkan kedalam register pertama dan kandungannya ditukar ke register kedua. Keuda-dua output dibandingkan .Sekiranya (compare=1),keadaan berubah ke keadaan 3, dan sekiranya (compare=0) ,keadaan akan berubah ke keadaan 4.

State 4 : Kandungan bagi register kedua, diikuti kandungan pada counter ditukar keluar ke sistem. Semasa 8 clock cycle pertama, register bertukar ke nilai yang baru untuk dibandingkan.Kaunter direset semula ke 1.

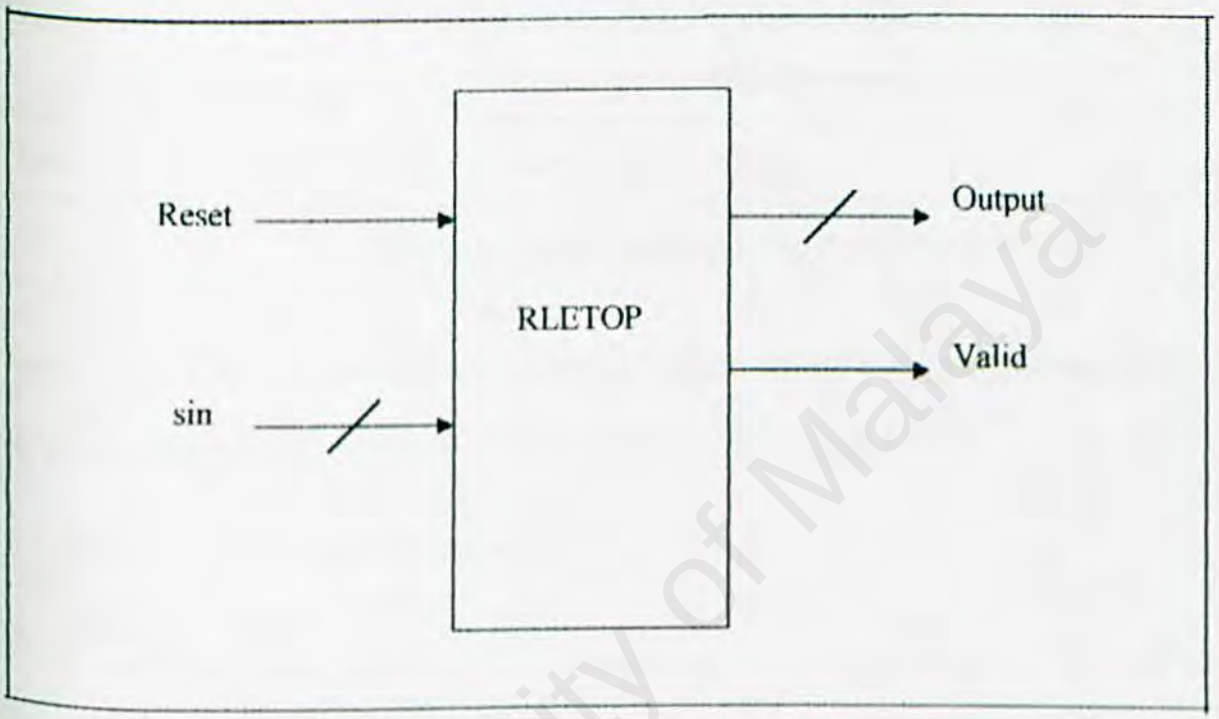
State 5 : Counter direset dan nilai di dalam 2 register dibandingkan. Sekiranya nilai bagi dua regiater adalah sama,keadaan akan berubah ke keadaan 3,dan sekiranya nilainya berlainan,keadaan akan berubah ke keadaan 5.

PIN	PENERANGAN
Clk	Kawalan pemasaan
Reset	Reset sistem
Compare	Masukan dari modul compare logic
Pixsi_1	Keluaran ke modul Counter
Count_pixels_1	Keluaran ke modul Counter
Reset2_1	Keluaran ke modul counter
Reset1_shift0_1	Keluaran ke modul shift0
Load_shift0_1	Keluaran ke modul shift1
Muxselect_1	Keluaran ke modul multiplexer
Valid_1	Keluaran sistem
Reset1_1	Keluaran ke modul shift1
Load_1	Keluaran ke modul shift1
Load_reg	Keluaran ke modul count register
Reset_reg	Keluaran ke modul count register

Jadual 5.7 : penerangan input dan output bagi modul control

Kod sumber bagi modul controller ini akan diletakkan di dalam appendik.

5.2.8 MODUL RLE (TOP LEVEL)



Rajah 5.9: Gambarajah blok bagi modul top level.

Modul RLE merupakan modul utama iaitu gabungan dari modul-modul yang ada iaitu modul shift0, modul shift1, modul compare, modul control, modul counter dan modul multiplexer. Rajah 5.9 merupakan gambarajah blok bagi rle manakala rajah 5.10 pula menunjukkan gambarajah keseluruhan bagi algoritma run-length ini.

PIN	PENERANGAN
Sin	Masukan 8 bit dari imej
Reset	Reset sistem
Output	Keluaran dari multiplexser (pixel atau count) Bergantung kepada input select
Valid	Sistem sah atau tidak.

Jadual 5.8 : Penerangan input dan output bagi modul rle (top level)

Kod sumber di bawah merupakan kod sumber bagi rle dn berfungsi sebagaimana yang dinyatakan di atas yang mana ia adalah berdasarkan kepada rajah 5.10 :

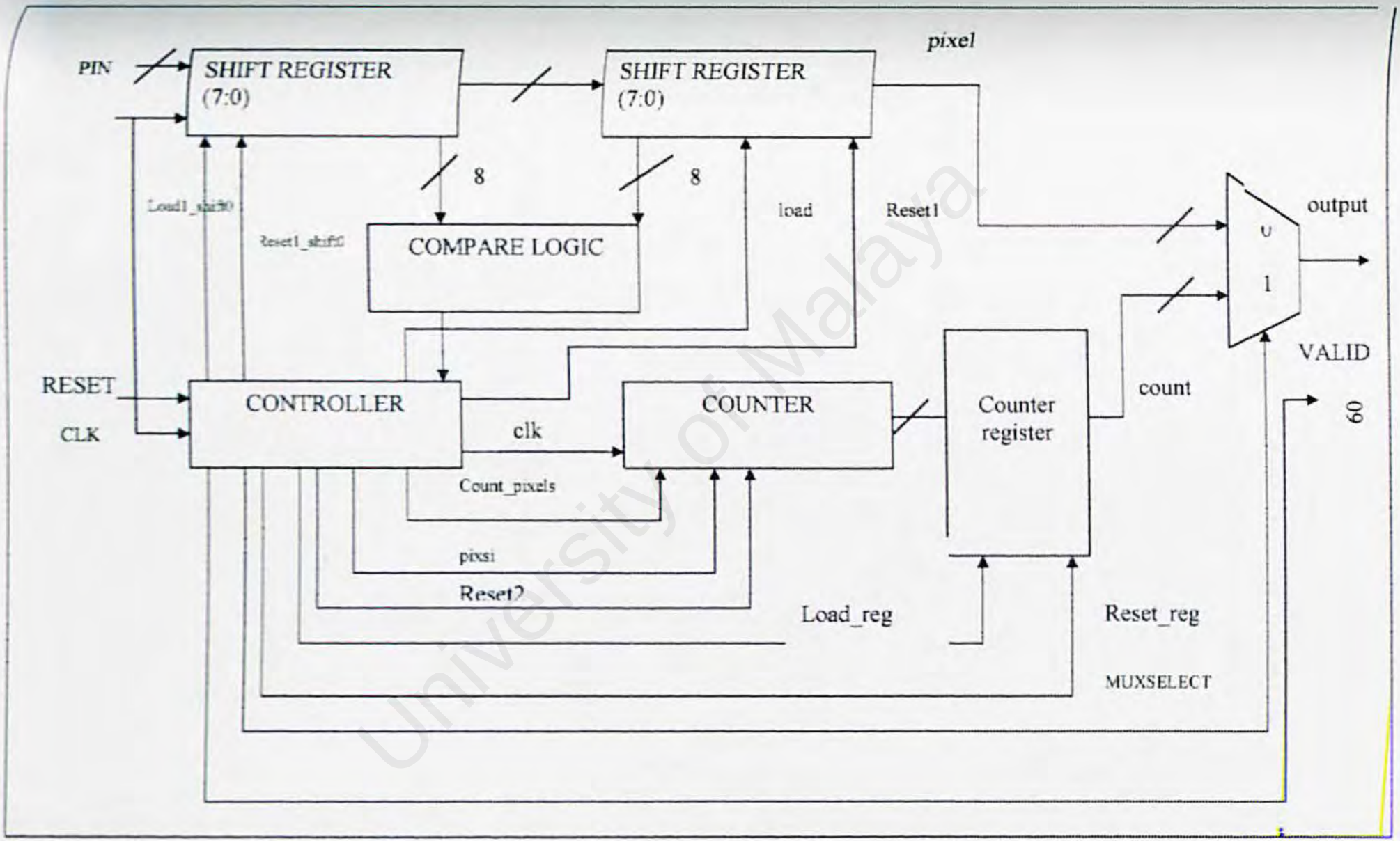
```
architecture structural of rletoplevel is

component shift0
    port ( pin : in std_logic_vector(7 downto 0);
           load_shift0,reset1_shift0,clk : in std_logic;
               sout,pout : out std_logic_vector(7 downto 0 ));

end component ;

component shift1
    port (clk       :   in  std_logic;
        sin :   in  std_logic_vector (7 downto 0);
        reset1  :   in  std_logic;
        load    :   in  std_logic;
        par_out :   out std_logic_vector (7 downto 0);
        sout2   :   out std_logic_vector (7 downto 0));
end component ;

component compare
    port ( A, B: in std_logic_vector(7 downto 0);
        z: out std_logic);

end component ;
```

```
component counter
    port ( clk,reset2,count_pixels,pixsi : in std_logic;
        countout : out std_logic_vector(7 downto 0) );
end component  ;


component countreg

    port ( clk : in std_logic;
            countregin : in std_logic_vector (7 downto 0);
            reset_reg : in std_logic;
            load_reg : in std_logic ;
            countout2 : out std_logic_vector(7 downto 0));
end component ;


component control
    port ( clk : in std_logic;
        reset : in std_logic;
        compare : in std_logic;
        count_pixels_1 : out std_logic;
        pixsi_1 : out std_logic;
        reset2_1 : out std_logic;
        muxselect_1 : out std_logic;
        valid_1  : out std_logic;
        reset1_1 : out std_logic;
        load_1 : out std_logic;
        reset1_shift0_1 : out std_logic;
        load_shift0_1 : out std_logic;
        reset_reg_1 : out std_logic;
        load_reg_1 : out std_logic);
end component ;

component mux
    port ( pixel:   in std_logic_vector(7 downto 0) ;
    count:   in std_logic_vector(7 downto 0) ;
    sel:     in std_logic ;
    output:  out std_logic_vector(7 downto 0) );
end component ;


    signal comp0,comp1,between_register, count1,sout2_1: std_logic_vector(7
downto 0);
    signal compout :std_logic;
    signal load_shift0_s,reset2_s,count_pixels_s,pixsi_s,muxselect_s :std_logic;
    signal reset1_shift0_s,reset1_s,load_s : std_logic;
```

```
    signal reset_reg_s, load_reg_s : std_logic;
    signal countreg1 :std_logic_vector (7 downto 0);

begin

U1 : shift0
        port map (
                        clk => clk,
                        pin => pin,
                        load_shift0 => load_shift0_s,
                        reset1_shift0 => reset1_shift0_s,
                        sout     => between_register,
                        pout     => comp0);
U2 : shift1
        port map (
                        clk => clk,
                        sin => between_register,
                        load => load_s,
                        reset1 => reset1_s,
                        Par_out => comp1,
                        sout2 => sout2_1);
U3 : compare
        port map (
                        A => comp0,
                        B => comp1,
                        Z => compout);
U4 : counter
        port map (
                        clk => clk,
                        reset2 => reset2_s,
                        count_pixels => count_pixels_s,
                        pixsi => pixsi_s,
                        countout => countreg1);


U5 : countreg
        port map (
                        clk => clk,
                        reset_reg => reset_reg_s,
                        load_reg => load_reg_s,
                        countregin => countreg1,
                        countout2 => count1);

U6 : mux
        port map (
                        pixel => sout2_1,
```

```
                count => count1,
                sel  => muxselect_s,
                output => output);

U7 : control
        port map (  clk => clk,
                reset => reset,
                compare => compout,
                pixsi_1 => pixsi_s,
                count_pixels_1 => count_pixels_s,
                reset2_1      => reset2_s,
                reset1_shift0_1 => reset1_shift0_s,
                muxselect_1 => muxselect_s,
                valid_1 => valid,
                reset1_1 => reset1_s,
                load_1 => load_s,
                load_shift0_1 => load_shift0_s,
                reset_reg_1 => reset_reg_s,
                load_reg_1 => load_reg_s);
end structural;
```

Gambarajah (Rajah 5.10) menunjukkan gabungan modul-modul yang telah digabungkan untuk menghasilkan satu rekabentuk perkakasan untuk pemampatan imej menggunakan teknik pengkodan run-length.Didapati bahawa register pertama iaitu modul shift0 akan menerima input dalam bentuk 8 bit (imej gray scale) . Dua output yang dikeluarkan iaitu pout dan sout masing-masing akan memasuki compare logic dan register kedua iaitu modul shift1. Kemudian modul shift1 akan melakukan tugas sama sebagaimana shift0 yang mana sout daripada shift0 akan diambil sebagai input dan output yang dikeluarkan adalah par_out dan sout2.Par_out ini akan memasuki compare logic manakala sout2 akan dijadikan input bagi modul multiplexer. Compare logic akan mengambil pout dan par_out sebagai input dan akan membandingkan kedua-dua input ini. Isyarat samada kedua-dua input adalah sama atau berlainan akan memasuki modul control dan modul ini akan mengeluarkan tiga isyarat iaitu ,reset2, pixsi dan counter_pixels ke modul counter, dan counter akan berfungsi berdasarkan kepada kedua-dua isyarat ini yang telah diterangkan pada (Jadual 5.4). Output daripada counter iaitu count akan dikeluarkan ke count register yang manaia akan menyimpan nilai count itu sehingga mendapat isyarat dari modul controller untuk mengeluarkan nilai itu ke multiplexer. Multiplexer pula akan memegang sementara kedua-dua input sout2 dan count dan akan mengeluarkan output berdasarkan isyarat select yang dikeluarkan dari modul control.



Rajah 5.9 : Gambarajah gabungan blok bagi pengekodan run-length

BAB 6
PENGUJIAN

5.3 TEKNIK PENGEKODAN

Di dalam membangunkan sebuah aturcara ataupun pengekodan yang baik,penulis telah merujuk kepada beberapa contoh kod sumber yang sedia ada.Apa yang penting bagi saya adalah bagaimana sesuatu penamaan itu diberikan terutamanya bagi setiap port-port masukan dan keluaran serta susunan bagi setiap proses yang berlaku.Contohnya adalah seperti sintaks IF THEN ELSE,CASE, WHEN dan lain-lain.

Bagi penamaan bagi setiap entity modul, saya memilih untuk menamakan setiap entiti dengan nama yang hampir sama dengan modul seperti 'control' bagi modul kawalan,counter,shiftregister,register,compare logic, dan multiplexer.

Dari segi penamaan bagi setiap port masukan dan keluaran juga saya memilih untuk memberikan skrip yang mudah untuk difahami.Ini memudahkan saya untuk membuat sebarang rujukan bagi mengelakkan sebarang kekeliruan daripada berlaku.

Saya juga menitikberatkan tentang susunan bagi setiap proses yang berlaku di dalam sesuatu modul yang dibangunkan terutamanya tentang susunan bagi sintaks yang berbeza.

Ianya bertujuan bagi memudahkan penulis atau pengguna lain memahami setiap proses yang berlaku.Disamping itu tanda'- -' diletakkan bagi menjelaskan operasi yang berlaku.

5.4 PEMBANGUNAN SISTEM

Pembangunan sistem merupakan elemen yang amat penting di dalam persekitaran pembangunan perisian.Ia melibatkan satu tugas yang memainkan peranan penting di dalam menjana kebolehberkesanan aturcara tersebut atau juga dikenali sebagai

pengekodan sistem.Di dalam membangunkan aturcara ini,pengekodan VHDL telah digunakan di mana ia juga digunakan sebagai satu bahasa pengaturcaraan berstruktur. Dalam membangunkan satu aplikasi pengisian satu tugas dilaksanakan di dalam projek untuk mengguna semua fail-fail yang berbeza yang akan menghasilkan satu aplikasi apabila digabungkan.Kesemua modul-mudul yang digabungkan ini disambungkan untuk menjadi satu aplikasi yang lengkap.

5.5 PENGUJIAN SISTEM

Proses pengujian mungkin merupakan satu bahagian yang kurang difahami di dalam projek pembangunan perisian dan kefungsian di dalam perisian.Selain daripada itu, pengujian juga satu proses latihan penggunaan dan penilaian sistem dengan manual yang telah disediakan oleh pembanguan sisten tersebut.Proses ini adalah bertujuan untuk mengesahkan dan memenuhi keperluan pengguna serta untuk mengenalpasti perbezaannya di antara keputusan yang dijangka dengan keputusan yang sebenar. Beberapa strategi pengujian digunakan dalam menguji aturcara rle ini iaitu pengujian modul,pengujian integrasi dan pengujian sistem top-level.

5.5.1 PENGUJIAN MODUL

Proses pengujian modul yang dijalankan ini adalah untuk mengenalpasti dan mengesan adakah kesemua komponen dan fungsian di dalam sistem ini berfungsi dengan betul dan berkesan melalui jenis input yang dijangkakan daripada kajian rekabentuk komponen-komponen tersebut.Langkah pertama yang perlu dilaksanakan adalah merekabentuk satu algonitima untuk fungsian tersebut dan menulis satu sumber

pengekodan dengan menggunakan algotima tersebut.Selepas itu, semak semula keseluruhan sumber pengekodan itu untuk memastikan algontimanya berfungsi dengan berkesan serta membandingkan kod-kod tersebut dengan gambarajah blok dan rekabentuknya supaya semua perkara-perkara berkaitan telah dilaksanakan.Langkah seterusnya adalah dengan melarikan dan memkompil sumber pengekodan tersebut dengan menggunakan perisian PEAK PPGA untuk memaparkan keputusan dalam bentuk simulasi di mana akan dipaparkan dalam bentuk gelombang (wareform) dan menghapuskan kesilapan sintaks yang berlaku sekiranya ada.Akhir sekali, kes-kes ujian dibangunkan untuk menunjukkan bahawa input telah ditukar dengan teliti kepada output yang benar-benar dikehendaki.

5.5.2PENGUJIAN INTEGRASI MODUL

Setelah kesemua komponen telah diuji sepenuhnya proses pengujian unlt dan berfungsi dengan betul memenuhi objektif yang dinyatakan.Kesemua komponen ini digabungkan untuk menjadi satu sistem yang lengkap dan dapat dilaksanakan dengan berkesan.Proses pengujian integrasi ini sebenarnya boleh dikatakan sebagai satu proses untuk menguji dan menetukan samada semua modul digabungkan mampu untuk berfungsi seperti yang dinyatakan di dalam fasa rekabentuk sistem.Pengujian adalah bertujuan untuk berfungsi dengan tepat.Sekiranya kesilapan ditemui,ia perlu diperbaiki dengan segera supaya objektif penggunaan sistem ini dapat dicapai dan dilaksanakan.

5.5.3PENGUJIAN KESELURUHAN SISTEM

Proses dan fasa pengujian sistem ini merupakan satu langkah pengujian yang terakhir dilaksanakan.Ia dilaksanakan bertujuan unutk memastikan aturcara yang dibangunkan ini dapat berjalan seperti yang dikehendaki bagi menghasilkan sebuah Aturcara ini diuji sepenuhnya samada memenuhi objektif kecekapan pelaksanaan spesifik di dalam pengujian pelaksanaan.

6.1 PENGENALAN

Di dalam bab ini saya akan menerangkan tentang bagaimana sesuatu modul-modul itu bila dibangunkan serta kaedah dan langkah-langkah yang diambil untuk menguji keberkesanan sesuatu modul seperti yang dikehendaki.

Pengujian merupakan langkah yang diambil bagi memastikan sesuatu modul itu dapat dilaksanakan mengikut arahan-arahan yang telah ditetapkan oleh saya sendiri.

Dengan melakukan pengujian ke atas modul yang telah dibangunkan, saya dapat mengesan sebarang ralat yang berlaku pada modul iaitu ke atas kod-kod yang telah ditulis.

Selain daripada mengesan ralat yang berlaku, teknik pengujian dapat memastikan tidak berlakunya sebarang kesalahan pada kod-kod yang telah ditulis dan memastikan ianya dapat beroperasi seperti yang dikehendaki.

6.2 PEMBANGUNAN MODUL

Bagi memastikan sesuatu modul itu dapat berfungsi dengan lancar, saya telah memilih untuk menghasilkan sesuatu modul yang besar kepada modul-modul yang kecil bagi memudahkan pengesanan ke atas ralat yang berlaku.Langkah ini diambil bertujuan untuk mempercepatkan pengesanan ke atas ralat yang berlaku ke atas kod-kod yang telah ditulis.Sebagai contoh saya telah membahagikan satu modul unit kawalan(oc) kepada beberapa blok mengikut input yang dimasukan buat modul-modul kecil.

6.3 PEMBANGUNAN TEST BENCH

Modul test bench dibangunkan secara berasingan dengan modul utama. Test bench bertujuan untuk menguji kebolehlaksanaan sesuatu modul yang telah dibangunkan. Dengan membangunkan testbench, saya dapat mengetahui sesuatu modul telah berfungsi seperti yang dikehendaki apabila ianya disimulasikan kemudiannya.Bagi aturan di atas contohnya testbench yang dihasilkan adalah seperti berikut.

6.4 PENGUJIAN DENGAN PERISIAN PEAK FPGA

Perisian PEAK PPGA telah digunakan untuk membangunkan projek ini.Perisian ini menyediakan servis yang mudah bagi kegunaan pengguna.Saya akan menerangkan tentang kemudahan ataupun cirri-ciri yang disediakan dalam membangunkan modul.Tiga menu utama yang penting dalam pengujian menggunakan perisian ini adalah penghompil (compile), pautan (link) dan simulasi (simulate).

6.5 PENGKOMPIL (COMPILER)

Perisian PEAK PPGA menyediakan khidmat penghompil untuk memudahkan saya mengetahui jika terdapat ralat pada modul yang ditulis oleh saya.Perisian ini akan mengeluarkan satu dialog yang memberitahu kepada pengguna tentang kesalahan yang terdapat pada aturcara yang telah ditulis.Antara kesalahan yang sering berlaku adalah kesalahan dari segi sintaks penamaan port yang tidak sama panjang bit yang berbeza, seperti yang telah diistiharkan pada entity (type mismata) dan yang *

Terdapat juga ralat yang kompleks yang mampu dikesan oleh perisian ini.

6.6 PAUTAN (LINK) DAN PEMETAAN PORT (PORT MAP)

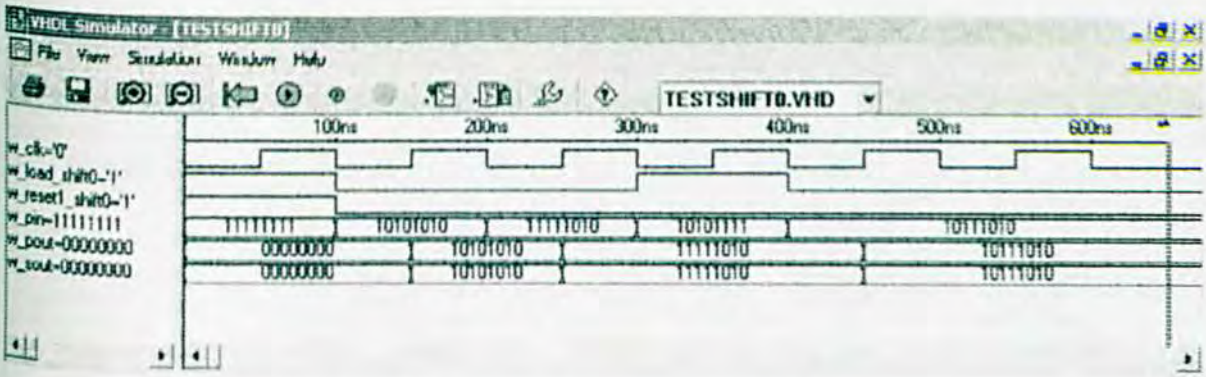
Link digunakan untuk membolehkan sesuatu modul itu diintegrasi dengan modul lain.Sebelum itu ia juga berfungsi untuk hubungkan modul dengan testbench.

Port map merupakan satu kaedah di mana setiap modul yang dibangunkan tadi disatukan dengan menyediakan satu lagi modul port map yang tersendiri.Ianya melibatkan kesemua modul dan top level.

6.7 SIMULASI

Merupakan antara yang paling penting dalam pastikan sesuatu modul itu berfungsi seperti yang dikehendaki.Simulasi dapat menunjukkan aliran input,output atau signal yang diistiharkan melalui bentuk gelombang atau 'woreform'.Daripada simulasi itu, kita boleh memriksa input yang dimasuksudkan dan juga output yang terhasil adalah seperti yang dijangkakan.

6.7.1 Simulasi bagi modul shift0



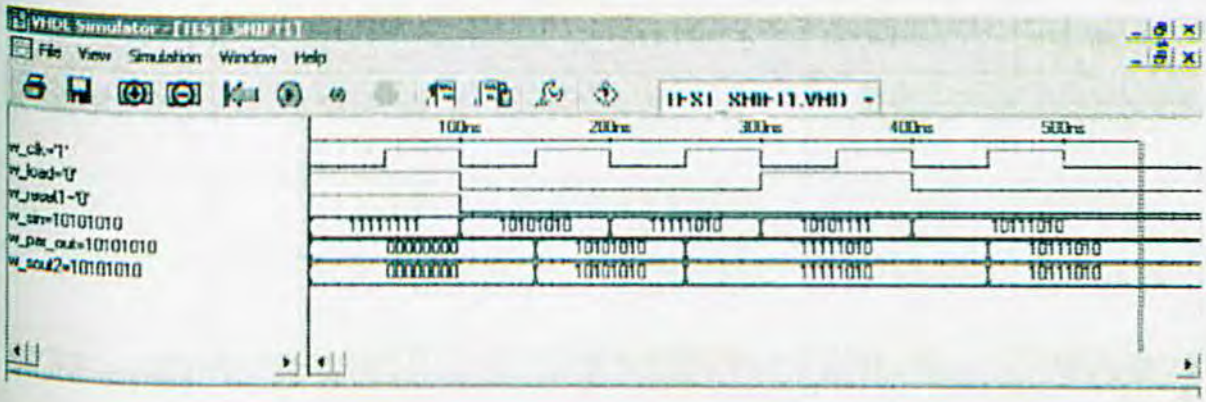
Rajah 6.1 : Simulasi bagi modul shift0

Rajah simulasi (Rajah 6.1) di atas merujuk kepada modul shif0.Rajah simulasi di atas menunjukkan input dan output yang digunakan.Nilai input adalah merujuk kepada 'test bench' yang dijangka oleh saya.Ini adalah kerana kesemua nilai input adalah daripada modul kawalan.Saya menunjukkan bahawa modul ini dapat digunakan dengan betul. Sila rujuk jadual di bawah (Jadual 6.1) bagi membandingkan input dan output yang terdapat pada simulasi dan rujuk kepada aturcara modul yang terdapat dalam appendiks.

INPUT				OUTPUT	
CLK	RESET1_SHIFT0	LOAD_SHIFT0	PIN	POUT	SOUT
0	1	1	11111111	00000000	00000000
1	1	1	11111111	00000000	00000000
0	0	0	10101010	10101010	10101010
1	0	0	11111010	11111010	11111010
0	0	0	11111010	11111010	11111010

Jadual 6.1 : pernyataan input dan output bagi modul shift0.

6.7.2 Simulasi bagi modul shift1



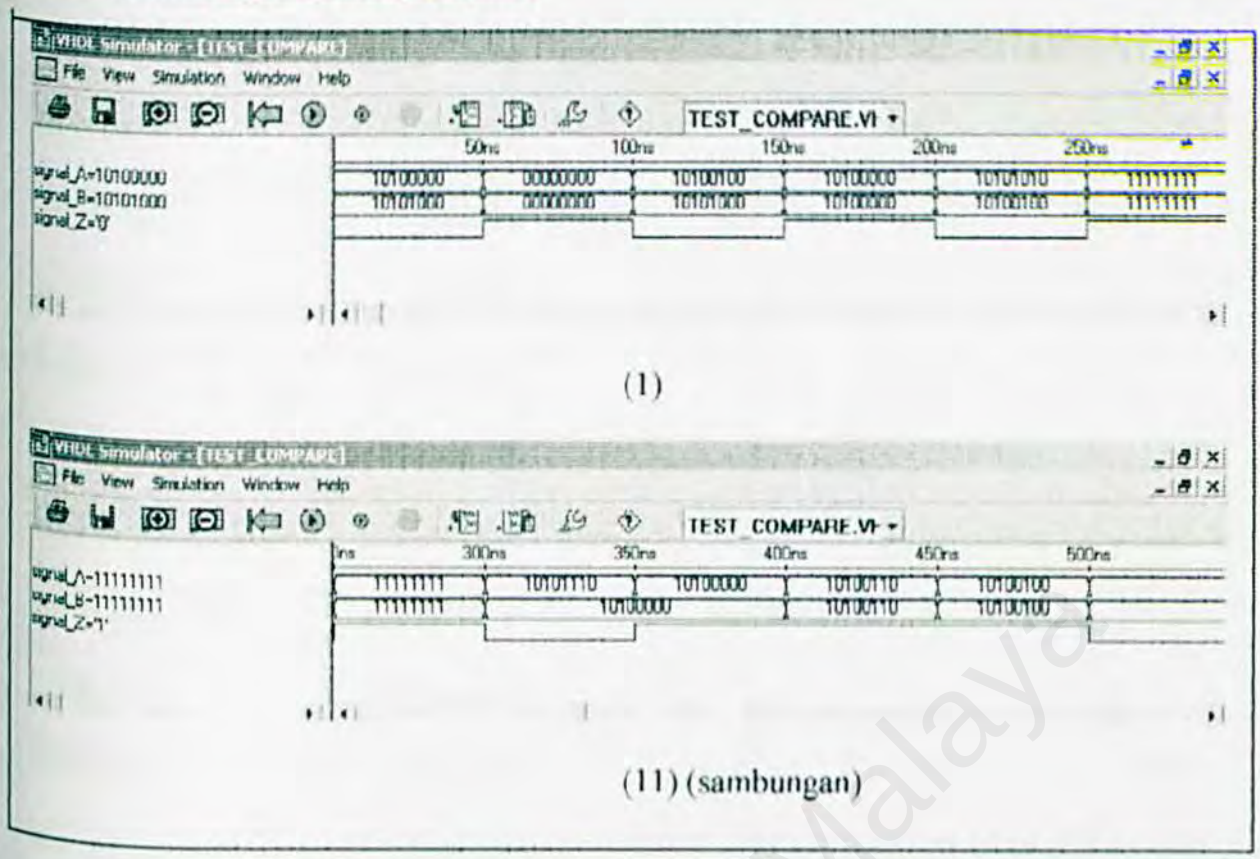
Rajah 6.2 : Simulasi bagi modul shift1

Rajah simulasi (Rajah 6.2) di atas merujuk kepada modul shift1.Rajah simulasi di atas menunjukkan input dan output yang digunakan.Nilai input adalah merujuk kepada 'test bench' yang dijangka oleh saya.Ini adalah kerana kesemua nilai input adalah daripada modul kawalan.Saya menunjukkan bahawa modul ini dapat digunakan dengan betul. Sila rujuk jadual di bawah (Jadual 6.2) bagi membandingkan input dan output yang terdapat pada simulasi dan rujuk kepada aturcara modul yang terdapat dalam appendiks.

INPUT				OUTPUT	
CLK	RESET1	LOAD	SIN	PAR OUT	SOUT2
0	1	1	11111111	00000000	00000000
1	1	1	11111111	00000000	00000000
0	0	0	10101010	10101010	10101010
1	0	0	11111010	11111010	11111010
0	0	0	11111010	11111010	11111010
1	0	1	10101111	11111010	11111010
0	0	0	10111010	10111010	10111010

Jadual 6.2 : pernyataan input dan output bagi modul shift1.

6.7.3 Simulasi bagi modul Compare.



(1)

(11) (sambungan)

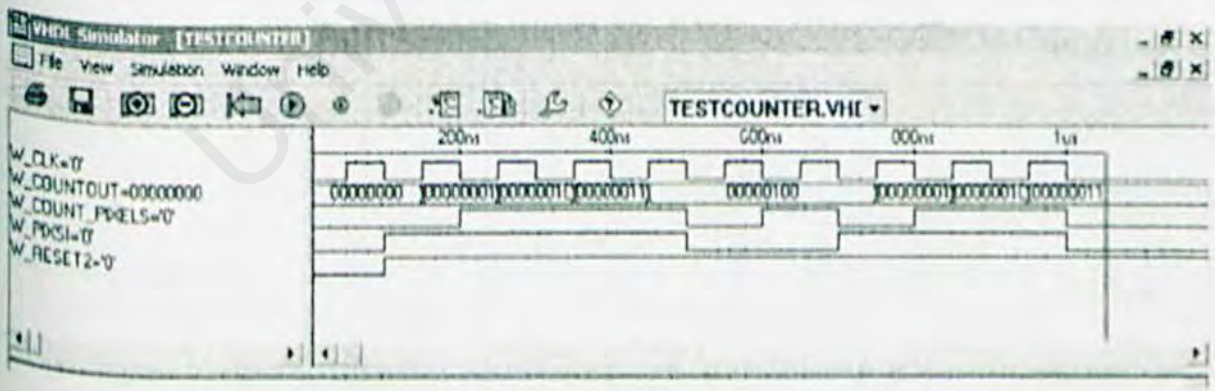
Rajah 6.3 : Simulasi bagi modul compare

Rajah simulasi di atas (Rajah 6.3) merujuk kepada modul compare.Rajah simulasi di atas menunjukkan input dan output yang digunakan.Nilai input adalah merujuk kepada 'test bench' yang dijangka oleh saya.Ini adalah kerana kesemua nilai input adalah daripada modul kawalan.Saya menunjukkan bahawa modul ini dapat digunakan dengan betul. Sila rujuk jadual di bawah (Jadual 6.3) bagi membandingkan input dan output yang terdapat pada simulasi dan rujuk kepada aturcara modul yang terdapat dalam appendiks.

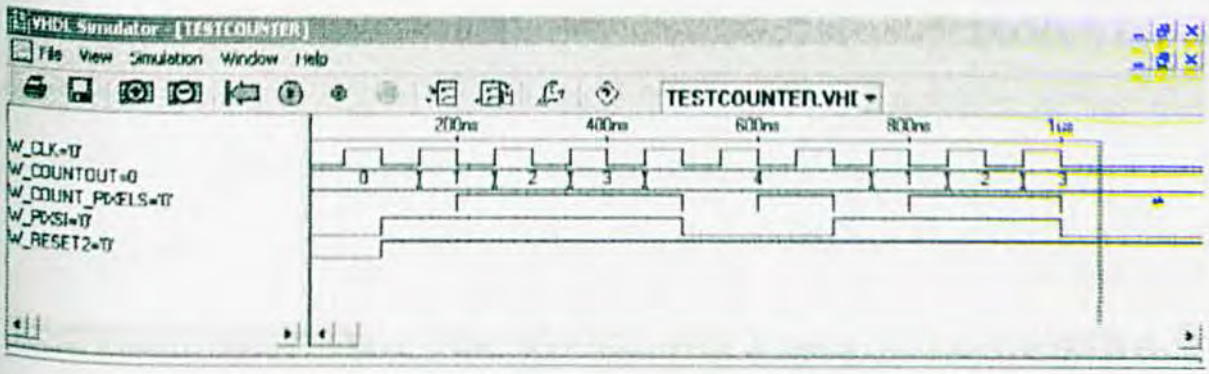
INPUT		OUTPUT
A	B	Z
10100000	10101000	0
00000000	00000000	1
10100100	10101000	0
10100000	10100000	1
10101010	10100100	0
11111111	11111111	1
10101110	10100000	0
10100000	10100000	1
10100100	10100100	1
10101100	10101000	0

Jadual 6.3 : Pernyataan input dan output bagi modul compare

6.7.4 Simulasi bagi modul counter



Rajah 6.4 (a) : Simulasi counter dalam binari



Rajah 6.4 (b) : Simulasi counter dalam desimal.

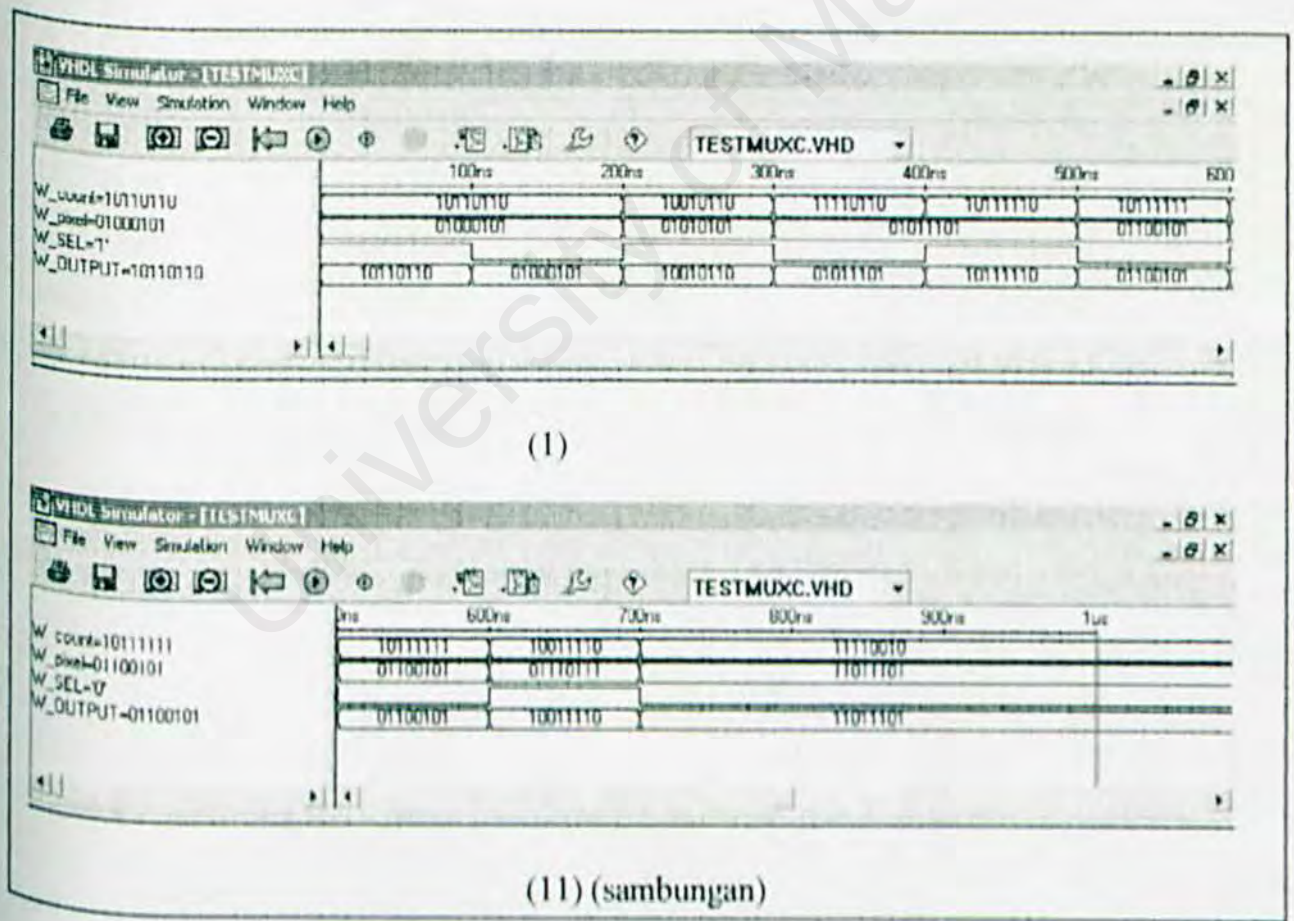
Rajah simulasi di atas (Rajah 6.4(a) dan 6.4(b)) merujuk kepada modul counter. Rajah simulasi di atas menunjukkan input dan output yang digunakan. Nilai input adalah merujuk kepada 'test bench' yang dijangka oleh saya.Ini adalah kerana kesemua nilai input adalah daripada modul kawalan.Saya menunjukkan bahawa modul ini dapat digunakan dengan betul. Sila rujuk jadual di bawah (Jadual 6.4) bagi membandingkan input dan output yang terdapat pada simulasi dan rujuk kepada aturcara modul yang terdapat dalam appendiks.

INPUT			OUTPUT	
Counter_pixels	Pixsi	Reset2	Countout (desimal)	Countout (binary)
0	0	0	0	00000000
0	1	1	1	00000001
1	1	1	2	00000010
1	1	1	3	00000011
1	1	1	4	00000100
0	0	1	4	00000100
1	0	1	4	00000100

0	1	1	1	00000001
1	1	1	2	00000010
1	1	1	3	00000011
0	0	1	3	00000011
1	1	1	3	00000011
1	0	1	3	00000011

Jadual 6.4 : penerangan input dan output bagi modul counter

6.7.5 Simulasi bagi modul multiplexer



(1)

(11) (sambungan)

Rajah 6.5 : Simulasi bagi modul multiplexer

Rajah simulasi di atas (Rajah 6.5) merujuk kepada modul multiplexer.Rajah simulasi di atas menunjukkan input dan output yang digunakan.Nilai input adalah merujuk kepada 'test bench' yang dijangka oleh saya.Ini adalah kerana kesemua nilai input adalah daripada modul kawalan.Saya menunjukkan bahawa modul ini dapat digunakan dengan betul. Sila rujuk jadual di bawah (Jadual 6.5) bagi membandingkan input dan output yang terdapat pada simulasi dan rujuk kepada aturcara modul yang terdapat dalam appendiks.

Input			output
Count	Pixel	Sel	output
10110110	01000101	1	10110110
10110110	01000101	0	01000101
10010110	01010101	1	10010110
11110110	01011101	0	01011101
10111110	01011101	1	10111110
10011111	01100101	0	01100101
10011110	01110111	1	10011110
11110010	11011101	0	11011101

Jadual 6.5 : Pernyataan input dan output bagi multiplexer

6.7.6 Simulasi modul countregister

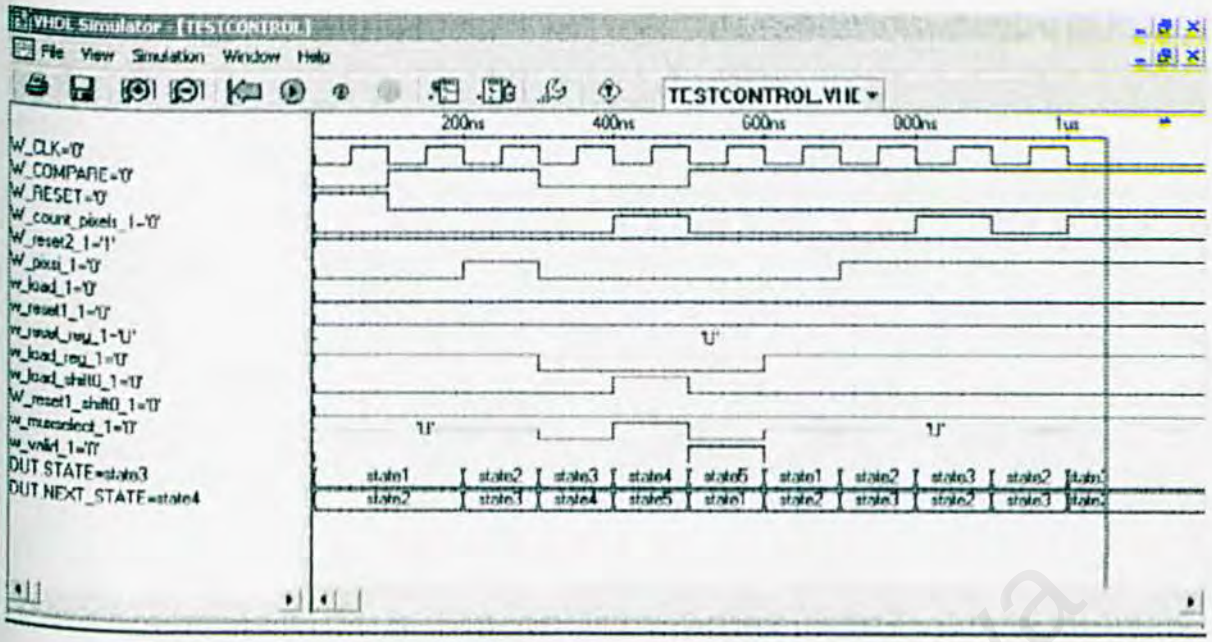
Rajah 6.6 : Simulasi bagi modul count register

Rajah simulasi di atas (Rajah 6.6) merujuk kepada modul modul count register.Rajah simulasi di atas menunjukkan input dan output yang digunakan.Nilai input adalah merujuk kepada 'test bench' yang dijangka oleh saya.Ini adalah kerana kesemua nilai input adalah daripada modul kawalan.Saya menunjukkan bahawa modul ini dapat digunakan dengan betul. Sila rujuk jadual di bawah (Jadual 6.6) bagi membandingkan input dan output yang terdapat pada simulasi dan rujuk kepada aturcara modul yang terdapat dalam appendiks.

INPUT				OUTPUT
Clk	Countregin	Load reg	Reset reg	Countout2
0	11111111	1	1	00000000
1	11011011	0	0	11011011
0	11100111	0	0	11100111

Jadual 6.6 : Pernyataan input dan output bagi modul count register.

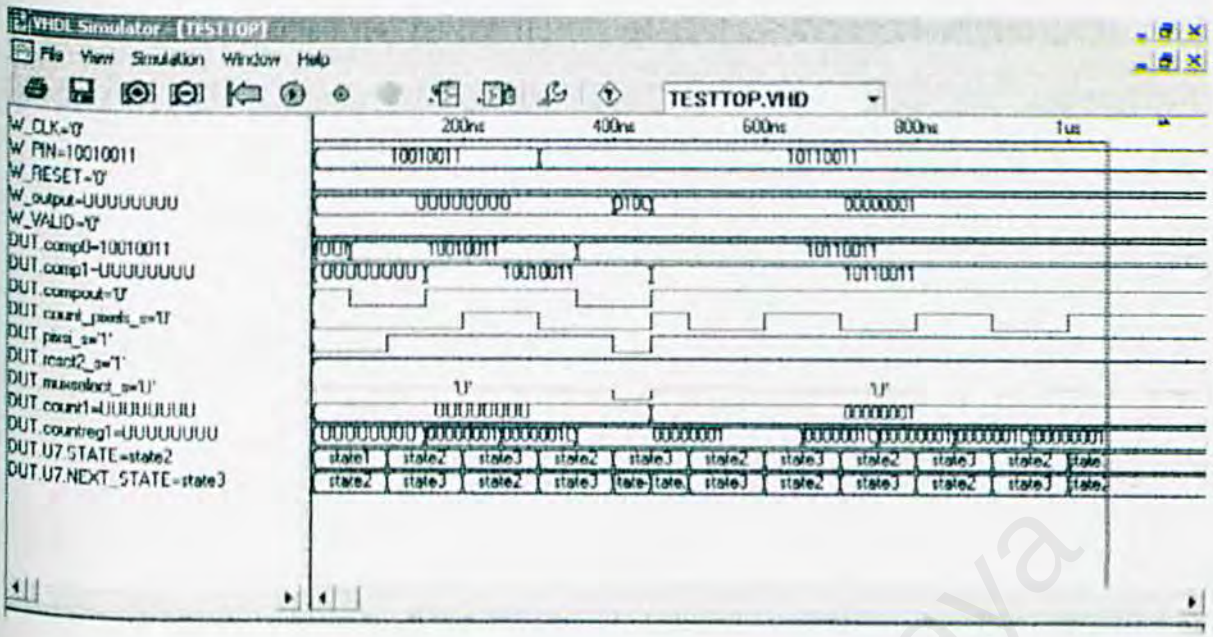
6.7.7 Simulasi bagi control



Rajah 6.7 : Simulasi bagi modul control

Rajah simulasi di atas (Rajah 6.7) merujuk kepada modul control.Rajah simulasi di atas menunjukkan input dan output yang digunakan.Nilai input adalah merujuk kepada 'test bench' yang dijangka oleh saya.Ini adalah kerana kesemua nilai input adalah berdasarkan kepada nilai compare yang dimasukkan dari modul compare logic. Rujuk kepada aturcara modul yang terdapat dalam appendiks.

6.7.8 Simulasi bg rletoplevel



Rajah 6.8 : Simulasi bagi modul rle (top level)

Rajah 6.8 menunjukkan simulasi bagi modul top level bagi rekabentuk perkakasan untuk pemampatan imej menggunakan algoritma run-length. Nilai PIN merupakan nilai masukan manakala nilai "OUTPUT" adalah untuk keluaran system. Walaupun modul-modul ini berjaya digabungkan, tetapi keluaran yang di hasilkan tidak menepati dengan apa yang sepatutnya. Walaubagaimanapun saya akan terangkan disini OUTPUT yang sepatutnya dikeluarkan berdasarkan kepada nilai PIN yang dimasukkan di dalam test bench :

Nilai PIN : 10010011 10010011 10010011 10110011 10110011

Nilai OUTPUT yang sepatutnya : 10010011(00000011) 10110011(00000010)

BAB 7
PERBINCANGAN DAN KESIMPULAN

7.1 PENGENALAN

Secara keseluruhannya, saya telah berjaya mencapai tujuan utama untuk kajian kali ini iaitu bagi menghasilkan sebuah rekabentuk untuk pemampatan imej menggunakan teknik run-length. Walaubagaimanapun, disebabkan kekangan yang dihadapi semasa membangunkan perisian ini, terdapat beberapa komponen yang tidak dapat dihasilkan semasa membangunkan rekabentuk ini. Dalam bab ini saya akan membincangkan tentang beberapa perkara yang dihadapi oleh saya semasa membangunkan projek ini.Antaranya adalah perubahan-perubahan yang dilakukan terhadap rekabentuk, masalah-masalah yang dihadapi beserta dengan penyelesaiannya, cadangan di masa hadapan, dan juga kesimpulan terhadap projek yang dijalankan sepanjang semester dua ini.

7.2 PERUBAHAN TERHADAP REKABENTUK

Sekiranya diperhatikan pada rekabentuk perkakasan yang dipaparkan dalam bab 4 iaitu bab rekabentuk yang dicadangkan, didapati beberapa perubahan telah dilakukan bagi memenuhi skop projek. Antara perubahan yang dilakukan adalah :

7.2.1 Rekabentuk terbaru tidak menggunakan register untuk threshold

Register untuk threshold tidak digunakan kerana penggunaan threshold di dalam proses pemampatan imej akan menyebabkan run-length ini menjadi teknik pemampatan 'lossiness' sedangkan run-length merupakan satu teknik pemampatan secara 'lossless'.

7.2.2Penambahan satu modul controller.

Modul controller atau pengawal ini merupakan modul yang perlu ada pada setiap rekabentuk untuk perkakasan. Ia dihasilkan daripada mesin keadaan (state machine) dan digunakan untuk mengawal modul-modul yang ada iaitu modul multiplexer, modul compare, modul shift0, shift1 dan modul counter.

7.3 MASALAH PEMBANGUNAN YANG DIHADAPI

Di dalam membangunkan projek pengkodan run-length dalam pemampatan imej ini, tentunya saya tidak dapat lari daripada menghadapi masalah yang sukar untuk diatasi.Langkah penyelesaian perlu dirangka dengan segera supaya masalah-masalah ini boleh dipertimbangkan pada masa yang akan datang dalam membangunkan projek ini.Antara masalah-masalah yang dihadapi oleh saya ialah :

7.3.1 Pemahaman tentang run-length

Saya mempunyai pengetahuan yang terhad tentang proses rekabentuk untuk perkakasan terutamanya untuk algoritma run –length ini walaupun ia merupakan satu algoritma yang ringkas dan mudah difahami.

Penyelesaian

Saya banyak melakukan proses pencarian di internet berkenaan algoritma run-length dan pemampatan imej itu sendiri. Jadi sedikit sebanyak pengetahuan saya berkenaan

algoritma ini dapat dipertingkatkan bagi membolehkan saya melakukan proses seterusnya dalam merekabentuk perkakasan untuk run-length ini.

7..3.2 Pengetahuan tentang VIIDL

Pengetahuan saya tentang VHDL adalah sangat terhad walaupun pernah mengikuti kursus VHDL sebelum ini. Jadi saya terpaksa merujuk kepada banyak sumber bagi memahami teknik dan cara untuk menghasilkan sebuah aturcara yang baik dalam VHDL.Saya juga berhadapan dengan masalah untuk memahami sebarang kesilapan yang berlaku semasa pengujian dijalankan, terutamanya ketika proses mengkompil aturcara yang telah siap untuk diuji.Ini berlaku terutamanya untuk kesilapan yang tidak melibatkan sintaks dan jarang dijumpai oleh saya seperti "aborting compile" ,amaran-amaran semasa mengkompil dan lain-lain lagi.

Penyelesaian

Saya melakukan kajian bagi setiap kod-kod yang diperolehi daripada internet bagi memahirkan diri dengan VHDL. Saya juga mempelajari bagaimana untuk menghasilkan sebuah teknik pengekodan yang baik melalui rujukan yang dibuat melalui internet dan juga buku. Hasilnya walaupun masih kurang mahir tentang penggunaan bahasa pengaturcaraan VHDI. ,sekurang-kurangnya pengetahuan saya tentang VHDI. adalah lebih baik dari sebelumya. Saya juga turut mendapatkan mendapatkan bantuan daripada penyelia projek dan rakan-rakan yang terlibat dalam menjalalani projek yang berkaitan.

7.3.3 Perisian PEAK FPGA

Perisian PEAK FPGA merupakan sesuatu yang baru bagi saya. Sebelum ini semasa mengikuti kursus VHDL. perisian yang digunakan adalah perisian Xillinx. Jadi saya terpaksa membiasakan diri dengan perisian PEAK FPGA terlebih dahulu sambil menghasilkan aturcara yang dikehendaki berkaitan dengan algortima run-length.

Penyelesaian

Saya kerap menggunakan perisian PEAK FPGA ini untuk membiakan diri dengan fungsi-fungsi yang tedapat dalam perisian ini. Saya juga kerap mengadakan perbincangan dengan rakan-rakan untuk mengatasi masalah ini.

7.3.4 Pengintegrasian modul

Ini merupakan masalah utama yang terpaksa dihadapi oleh saya iaitu untuk menggabungkan modul-modul seperti control, shift0 , shift1 ,compare , counter, dan mux ke dalam modul top-level.

Penyelesaian

Saya melakukan kaedah cuba jaya untuk menggabungkan modul-modul ini sambil melakukan perbincangan dengan rakan-rakan yang terlibat.

7.3.5 Had Masa

Untuk menghasilkan satu perkakasan untuk pemampatan imej ,pastinya memakan masa yang panjang. Jadi dengan masa yang terhad sepanjang semester kedua ini , saya

hanya sempat menghasilkan modul untuk pemampatan sahaja. Ini kerana dalam masa lebih kurang 3 bulan untuk semester 2 ini, saya harus mengkaji tentang penggunaan bahasa pengaturcaraan VHDL , mamahirkan diri dengan penggunaan perisian PEAK FPGA di samping terpaksa menyiapkan tugasan – tugasan bagi kursus yang diambil pada semester ini.

Penyelesaian

Saya harus peka terhadap pembahagian masa bagi membangunkan projek ini. Pembahagian masa yang lebih efisien harus dibentuk untuk membolehkan saya memberi sepenuh perhatian dan tumpuan terhadap pembangunan projek. Pembahagian masa adalah penting untuk membolehkan saya menyiapkan projek ini pada masa yang ditetapkan.

7.3.6 Sumber rujukan

Rujukan merupakan satu sumber yang penting di dalam membuat kajian dan membangunkan projek ini. Tapi saya mempunyai masalah bahan rujukan yang kurang .Contohnya perpustakaan sendiri kurang menyediakan buku berkenaan VHDL dan berkenaan pemampatan imej sendiri. Buku rujukan yang disediakan juga sudah lama dan disediakan dengan jumlah yang kecil dan terhad.

Peyelesaian

Memandangkan sumber rujukan dari buku adalah berkurangan, saya telah mengambil inisiatif untuk memperbanyakkan pencarian di internet sebagai satu sumber rujukan yang

terpenting. Di samping itu , pihak perpustakaan juga diminta agar meyediakan lebih banyak rujukan berkaitan computer terutamanya buku-buku berkaitan VHDL. Buku-buku adisi terkini juga diharap dapat disediakan oleh pihak perpustakaan.

7.3.7 Perkakasan dan perisian.

Masalah yang dihadapi adalah berkaitan dengan perkakasan yang dimiliki oleh saya tidak menepati spesifikasi yang diperlukan oleh PEAK FPGA yang memerlukan computer berkuasa tinggi bagi membolehkan ia dilarikan dengan lancar. Penggunaan computer adalah terhad di makmal sahaja iaitu pada waktu pagi dan petang. Jadi ia sidikit sebanyak mengganngu usaha saya dalam mambangunkan projek ini.

Penyelesaian

Saya telah membuat draf bagi setiap modul yang ingin dibangunkan . Ini bagi membolehkan saya menggunakan kemudahan di makmal dengan semaksima mungkin untuk menyiapkan modul-modul tersebut.

7.4 CADANGAN MASA HADAPAN

Sekiranya projek ini ingin diteruskan pada masa hadapan, saya dapati terdapat cirri- cirri yang boleh ditambah dan diperbaiki untuk membolehkan rekabentuk perkakasan ini menepati apa yang dirancangkan. Ciri-ciri tersebut adalah seperti berikut :

7.4.1 Penggunaan TEXT I/O

Penggunaan TEXT I/O juga merupakan satu cadangan yang patut digunakan pada peringkat awal. TEXT I/O adalah satu pakej yang digunakan untuk membaca input yang dibina dari satu fail teks. Jadi ,input tidak dimasukkan sendiri didalam testbench sebaliknya dibaca dari fail teks tadi. Jadi ia seolah-olah melakukan satu pemampatan terhadap bit yang terdapat pada imej sebenar.

7.4.2 Penambahan modul untuk penyahmampat.

Dalam menghasilkan projek pemampatan imej ini, saya hanya sempat menyiapkan modul untuk pemampatan sahaja . jadi jika projek ini dibangunkan semula, saya cadangkan agar modul penyahmampatan ini dapat ditambah bagi mempertingkatkan lagi kebolehpercayaan terhadap peranti.

7.4.3 Penambahan modul pengesanan ralat

Oleh kerana perkakasan yang direkabentuk ini tidak menyokong proses pengesanan ralat, jadi saya berharap modul pengesanan ralat ditambah pada modul yang sedia ada. Ini adalah untuk menyokong proses penghantaran data dari satu lokasi ke satu lokasi yang lain (melalui rangkaian).

Seperti yang diketahui, penghantaran data antara dua lokasi berlainan akan menghasilkan ralat bit. Walaupun pemampatan telah dilakukan , data yang telah termampat tidak terelak dari bencana ralat. Oleh itu ,sekiranya ia dinyahmampat ,ia akan menghasilkan data yang berlainan daripada data sebelum dimampat. Oleh itu dengan adanya modul pengesanan ralat, ralat akan dapat dikesan dan masalah kehilangan data akan dapat diatasi.

7.4.4 Menyiapkan modul top-level bagi RLE (mengkaji balik modul controller)

Sekiranya dilihat pada Rajah 6.7 didapati bahawa output yang dikeluarkan adalah berlainan daripada yang diharapkan. Setelah dibuat kajian yeng teliti , iaitu memeriksa kod sumber bagi rletoplevel.vhd, didapati bahawa tiada kesalahan berlaku pada kod tersebut. Jadi, saya mengganggap bahawa kesalahan telah berlaku pada kod sumber control.vhd. Oleh kerana kesuntukan masa , kerana proses pengujian bagi modul top level adalah paling akhir dilakukan, saya tidak berkesempatan untuk mengkaji state machine bagi algoritma run-length bagi mengesan kesalahan-kesalahan yang berlaku Oleh sebab itu saya cadangkan sekiranya projek ini diteruskan, kod sumber bagi modul controller ini dapat diperbaiki dan dikaji semula . Apa yang perlu dikaji adalah mesin keadaan (state

machine) iaitu proses-proses yang patut dilalui oleh algoritma ini bagi menghasilkan satu modul pemampatan bagi run-length dapat berfungsi seperti yang dikehendaki.

7.5 KESIMPULAN.

Alhamdulillah, walaupun projek ini tidak dapat disiapkan mengikut perancangan, namun ia tetap mencapai objektifnya, walaupun saya hanya merekabentuk modul pemampatan untuk pengekodan run-length, namun idea pembangunannya boleh dikembangkan lagi untuk menghasilkan modul myahpemampatan.

Pembangunan kod sumber bagi perkakasan merupakan suatu perkara yang kompleks berbanding pembangunan kod sumber untuk perisian. Segala perancangan yang pada awalnya agak mudah,namun setelah melalui proses pembangunannya, ternyata ia amat sukar bagi saya tidak berlatarbelakangkan tentang pembangunan perkakasan.

Walaupun kod sumber yang dihasilkan telah betul, dan dapat menghasilkan keluaran yang sepatutnya , namun litar dalam perkakasan boleh memberi kesan yang sebaliknya kepada data output yang dihasilkan.

Secara keseluruhannya, projek ini telah memberikan saya satu ilmu yang baru serta pengalaman yang amat berguna dalam merekabentuk satu perkakasan. Dalam projek ini,say telah melalui kaedah merekabentuk seperti dalam fasa pembangunan , pengekodan dan pengujian. Selain itu saya juga dapat mempelajari tentang cirri-ciri dan kelakuan perkakasan ,iaitu sesuatu yang tidak pernah saya ketahui sebelum ini.

APPENDIK A
KOD SUMBER BAGI SETIAP MODUL

APPENDIK A Kod sumber bagi setiap modul

Kod sumber bagi modul shift0.vhd

```
library ieee;
use ieee.std_logic_1164.all;

entity shift0 is -- 8 bit register file
    port(   clk      :   in  std_logic;
            pin :    in  std  logic  vector (7 downto 0);
            reset1_shift0    :   in  std_logic;
            load_shift0 :    in  std_logic;
            pout     :   out std_logic_vector (7 downto 0);
            sout     :   out std_logic_vector (7 downto 0));
end shift0;

architecture structural of shift0 is
    begin
        process (clk, reset1_shift0)
            begin
                if (reset1_shift0 = '1') then -- dalam keadaan 'high state'
                    pout <= "00000000";
                    sout <= "00000000";
                    elsif (clk'event and clk ='1') then -- dinilai jika nilai reset bukan 1
                                                        -- untuk kesan 'clock
edges'
                        if (load_shift0 = '0') then
                            pout <= pin;
                            sout <= pin;
                    end if;
            end if;
        end process;
    end structural;
```

Kod sumber bagi modul shift1.vhd

```
library ieee;
use ieee.std_logic_1164.all;

entity shift1 is -- 8 bit register file
    port(   clk      :   in  std_logic;
            sin      :   in  std_logic_vector (7 downto 0);
            reset1   :   in  std_logic;
            load     :   in  std_logic;
            par_out  :   out std_logic_vector (7 downto 0);
            sout2    :   out std_logic_vector (7 downto 0));
end shift1;

architecture structural of shift1 is
    begin
        process (clk, reset1)
            begin
                if (reset1 = '1') then -- dalam keadaan 'high state'
                    par_out <= "00000000";
                    sout2 <= "00000000";
                    elsif (clk'event and clk ='1') then -- dinilai jika nilai reset bukan 1
                                                        -- untuk kesan 'clock
edges'
                        if (load = '0') then
                            par_out <= sin;
                            sout2 <= sin;
                    end if;
            end if;
        end process;
    end structural;
```

APPENDIK A Kod sumber bagi setiap modul

Kod sumber bagi modul compare.vhd

```
library ieee;                -- Load the ieee 1164 library
use ieee.std_logic_1164.all; -- Make the ieee package 'visible'

-- Define the top-level interface for the circuit...

entity compare is
     port(A, B: in std_logic_vector(7 downto 0);
                z: out std_logic);
end compare;

-- Now define the contents of the entity...

architecture structural of compare is
     begin

     z <= '1' when (A = B) else '0';    -- Concurrent assignment

end structural;


-- Now define the three possible configurations...

configuration equality of compare is
  for structural
  end for;
end configuration equality;
```

APPENDIK A Kod sumber bagi setiap modul

Kod sumber bagi modul counter.vhd

```
Library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity counter is


 port (
  CLK              : in  std_logic;
  Count_Pixels     : in  std_logic;
  PIXSI            : in  std_logic;
  RESET2           : in  std_logic;
  Countout : out std_logic_vector(7 downto 0));

end counter;
architecture structural of counter is
signal Temp_Counter_Value_Pixels : std_logic_vector(7 downto 0);

begin


counter: process (RESET2,CLK)  -- Counter for the number of pixels that
                               -- have been received.

begin

      if RESET2 = '0' then
       Temp_Counter_Value_Pixels <= "00000000";

            elsif CLK'event and CLK = '1' then
               if (PIXSI = '1') then
                     if Count_Pixels = '0' then
                     Temp_Counter_Value_Pixels <= "00000001";
                  else
                  Temp_Counter_Value_Pixels <= Temp_Counter_Value_Pixels +
"00000001";
```

```
                end if;
            else
                if Count_Pixels = '0' then
                Temp_Counter_Value_Pixels <= temp_counter_value_pixels;
        end if;
       end if;
      end if;
  end process;
      Countout <= Temp_Counter_Value_Pixels;

end structural;
```

APPENDIK A Kod sumber bagi setiap modul

Kod sumber bagi modul multiplexer

```
library ieee ;
use   ieee.std_logic_1164.all ;

entity mux is
   port (
            pixel:   in std_logic_vector(7 downto 0) ;
        count:   in std_logic_vector(7 downto 0) ;
        sel:     in std_logic ;
        output:  out std_logic_vector(7 downto 0)
   ) ;
end mux ;

architecture structural of mux is
begin
  with sel select
      output <= pixel when '0',
                count when others ;
end structural ;
```

APPENDIK A Kod sumber bagi setiap modul

Kod sumber bagi modul control.vhd

```
library IEEE;

use IEEE.std_logic_1164.all;

entity control is

  port( clk : in std_logic;
                    reset : in std_logic;
                    compare : in std_logic;
                    muxselect_1 : out std_logic;
                    valid_1 : out std_logic;
                    load_shift0_1 : out std_logic;
                    reset1_shift0_1 : out std_logic;
                    pixsi_1 : out std_logic;
                    count_pixels_1 : out std_logic;
                    reset2_1 : out std_logic;
                    reset1_1 : out std_logic;
                    load_1 : out std_logic;
                    reset_reg_1 : out std_logic;
                    load_reg_1 : out std_logic);


end control;

architecture SYN_USE_DEFA_ARCH_NAME of control is

  -- Declare an enum type for the state
  type STATE_TYPE is (state1,state2,state3,state4,state5,state6);


  -- Declare state variables
  signal STATE : STATE_TYPE;
  signal NEXT_STATE : STATE_TYPE;

  -- Set the state vector attribute
  attribute STATE_VECTOR : STRING;
  attribute STATE_VECTOR of SYN_USE_DEFA_ARCH_NAME : architecture is
"STATE";
```

```
begin
  -- This process sets the next state on the clock edge.
  SET_STATE: process ( clk, reset )
  begin
    if ( reset = '1' ) then
      STATE <= state1;
    elsif ( clk'event and clk = '0' ) then
      STATE <= NEXT_STATE;
    end if;
  end process SET_STATE;

  -- This process determines the next state and output values
  -- based on the current state and input values.
  SET_NEXT_STATE: process (STATE, compare)
    variable NEXT_STATE_VALUE : STATE_TYPE;
  begin

    -- Set defaults for the next state and all outputs.
        pixsi_1 <= 'X';
        count_pixels_1 <= 'X';
        reset2_1 <= 'X';
        muxselect_1 <= 'X';
        valid_1 <= 'X';
                load_shift0_1 <= 'X';
                reset1_shift0_1 <= 'X';
                load_1 <= 'X';
                reset1_1 <= 'X';
                reset_reg_1 <= 'X';
                load_reg_1 <= 'X';

    NEXT_STATE_VALUE := state1;
    case STATE is
      when state1 =>
                        pixsi_1 <= '0';
                count_pixels_1 <= '0';
                reset2_1 <= '1';
                muxselect_1 <= 'U';
                valid_1 <= '0';
                        load_shift0_1 <= '0';
                        reset1_shift0_1 <= '0';
                        load_1 <= '0';
                        reset1_1 <= '0';
                        reset_reg_1 <= '0';
                        load_reg_1 <= '0';

                        NEXT_STATE_VALUE := state2;
```

```
when state2 =>
                pixsi_1 <= '1';--reset pada satu
        count_pixels_1 <= '0';
                reset2_1 <= '1';
        muxselect_1 <= 'U';
        valid_1 <= '0';
                load_shift0_1 <= '0';
                reset1_shift0_1 <= '0';
                load_1 <= '0';
                reset1_1 <= '0';
                reset_reg_1 <= '0';
                load_reg_1 <= '0';

                NEXT_STATE_VALUE := state3;

when state3 =>
                pixsi_1 <= '0';--
        count_pixels_1 <= '0';
                reset2_1 <= '1';
        muxselect_1 <= 'U';
        valid_1 <= '0';
                load_shift0_1 <= '0';--
                reset1_shift0_1 <= '0';
                load_1 <= '0';--
                reset1_1 <= '0';
                reset_reg_1 <= '0';
                load_reg_1 <= '0';


  if ((compare) = '1' ) then
                        pixsi_1 <= '1';
                count_pixels_1 <= '1';
                        reset2_1 <= '1';
                muxselect_1 <= 'U';
                valid_1 <= '0';
                        load_shift0_1 <= '0';
                        reset1_shift0_1 <= '0';
                        load_1 <= '0';
                        reset1_1 <= '0';
                        reset_reg_1 <= '0';
                        load_reg_1 <= '0';

    NEXT_STATE_VALUE := state2;
  end if;
  if ((compare) = '0' ) then
```

```
                pixsi_1 <= '0';
            count_pixels_1 <= '1';--1
                reset2_1 <= '1';
            muxselect_1 <= '0';
            valid_1 <= '0';
                load_shift0_1 <= '0';--1
                reset1_shift0_1 <= '0';
                load_1 <= '0';--
                reset1_1 <= '0';
                reset_reg_1 <= '0';
                load_reg_1 <= '0';


    NEXT_STATE_VALUE := state4;
  end if;

when state4 =>
        pixsi_1 <= '0';
            count_pixels_1 <= '1';--
                reset2_1 <= '1';
            muxselect_1 <= '1';
            valid_1 <= '0';
                load_shift0_1 <= '1';--
                reset1_shift0_1 <= '0';
                load_1 <= '0';
                reset1_1 <= '0';
                reset_reg_1 <= '0';
                load_reg_1 <= '0';

        NEXT_STATE_VALUE := state5;

when state5 =>
        pixsi_1 <= '0';
            count_pixels_1 <= '0';--
                reset2_1 <= '1';
            muxselect_1 <= '0';
            valid_1 <= '1';
                load_shift0_1 <= '0';
                reset1_shift0_1 <= '0';
                load_1 <= '0';
                reset1_1 <= '0';
                reset_reg_1 <= '0';
                load_reg_1 <= '0';
```

```
        NEXT_STATE_VALUE := state1;

    end case;
    NEXT_STATE <= NEXT_STATE_VALUE;
  end process SET_NEXT_STATE;

end SYN_USE_DEFA_ARCH_NAME;
```

APPENDIK A Kod sumber bagi setiap modul

Kod sumber bagi modul countreg.vhd

```
library ieee;
use ieee.std_logic_1164.all;

entity countreg is -- 8 bit register file
        port(   clk             :       in      std_logic;
                countregin      :       in      std_logic_vector (7 downto 0);
                reset_reg       :       in      std_logic;
                load_reg        :       in      std_logic;
                countout2       :       out     std_logic_vector (7 downto 0));

end countreg;

architecture structural of countreg is
        begin
                process (clk, reset_reg)
                        begin
                                if (reset_reg = '1') then -- dalam keadaan 'high state'
                                        countout2 <= "00000000";

                                elsif (clk'event and clk ='1') then -- dinilai jika nilai
reset bukan 1

                        -- untuk kesan 'clock edges'
                                                if (load_reg = '0') then
                                                        countout2 <= countregin;

                                        end if;
                        end if;
                end process;
        end structural;
```

APPENDIK A Kod sumber bagi setiap modul

Kod sumber bagi modul rletoplevel.vhd

```
library ieee;
use ieee.std_logic_1164.all;

entity rletoplevel is
        port (clk : in std_logic;
                pin : in std_logic_vector(7 downto 0);
                reset : in std_logic;
                output : out std_logic_vector(7 downto 0);
                valid : out std_logic
        );
end rletoplevel;

architecture structural of rletoplevel is

        component shift0
                port ( pin : in std_logic_vector(7 downto 0);
                                load_shift0,reset1_shift0,clk : in std_logic;
                                        sout,pout : out std_logic_vector(7 downto 0 ));

        end component ;

        component shift1
                port (clk               :       in      std_logic;
                        sin     :       in      std_logic_vector (7 downto 0);
                        reset1  :       in      std_logic;
                        load    :       in      std_logic;
                        par_out         :       out     std_logic_vector (7 downto 0);
                        sout2   :       out std_logic_vector (7 downto 0));
        end component ;

        component compare
                port ( A, B: in std_logic_vector(7 downto 0);
                z: out std_logic);

        end component ;

        component counter
                port ( clk,reset2,count_pixels,pixsi : in std_logic;
                countout : out std_logic_vector(7 downto 0) );
```

```
end component  ;


component countreg

        port ( clk : in std_logic;
                        countregin : in std_logic_vector (7 downto 0);
                        reset_reg : in std_logic;
                        load_reg : in std_logic ;
                        countout2 : out std_logic_vector(7 downto 0));
end component ;


component control
        port ( clk : in std_logic;
                reset : in std_logic;
                compare : in std_logic;
                count_pixels_1 : out std_logic;
                pixsi_1 : out  std_logic;
                reset2_1 : out std_logic;
                muxselect_1 : out std_logic;
                valid_1   : out std_logic;
                reset1_1 : out std_logic;
                load_1 : out std_logic;
                reset1_shift0_1 : out std_logic;
                load_shift0_1 : out std_logic;
                reset_reg_1 : out std_logic;
                load_reg_1 : out std_logic);
end component ;

component mux
        port ( pixel:   in std_logic_vector(7 downto 0) ;
count:   in std_logic_vector(7 downto 0) ;
sel:     in std_logic ;
output:  out std_logic_vector(7 downto 0) );
end component ;


        signal comp0,comp1,between_register, count1,sout2_1:
std_logic_vector(7 downto 0);
        signal compout :std_logic;
        signal
load_shift0_s,reset2_s,count_pixels_s,pixsi_s,muxselect_s :std_logic;
        signal reset1_shift0_s,reset1_s,load_s : std_logic;
        signal reset_reg_s, load_reg_s : std_logic;
        signal countreg1 :std_logic_vector (7 downto 0);
```

```
begin

U1 : shift0
        port map (
                clk => clk,
                pin => pin,
                load_shift0 => load_shift0_s,
                reset1_shift0 => reset1_shift0_s,
                sout    => between_register,
                pout    => comp0);
U2      : shift1
        port map (
                clk => clk,
                sin => between_register,
                load => load_s,
                reset1 => reset1_s,
                Par_out => comp1,
                sout2 => sout2_1);
U3      : compare
        port map (
                A => comp0,
                B => comp1,
                Z => compout);
U4      : counter
        port map (
                clk => clk,
                reset2 => reset2_s,
                count_pixels => count_pixels_s,
                pixsi => pixsi_s,
                countout => countreg1);

U5  : countreg
        port map (
                clk => clk,
                reset_reg => reset_reg_s,
                load_reg => load_reg_s,
                countregin => countreg1,
                countout2 => count1);

U6 : mux
        port map (
                pixel => sout2_1,
                count => count1,
                sel     => muxselect_s,
```

```
                                output => output);

U7 : control
        port map (      clk => clk,
                        reset => reset,
                        compare => compout,
                        pixsi_1 => pixsi_s,
                        count_pixels_1 => count_pixels_s,
                        reset2_1        => reset2_s,
                        reset1_shift0_1 => reset1_shift0_s,
                        muxselect_1 => muxselect_s,
                        valid_1 => valid,
                        reset1_1 => reset1_s,
                        load_1 => load_s,
                        load_shift0_1 => load_shift0_s,
                        reset_reg_1 => reset_reg_s,
                        load_reg_1 => load_reg_s);

end structural;
```

APPENDIK B
TEST BENCH

APPENDIK B Test bench

TESTSHIFT0.VHD

```
library ieee;
use ieee.std_logic_1164.all;
-- use ieee.numeric_std.all;     -- Note: uncomment this if you use
                                 -- IEEE standard signed or unsigned types.
-- use ieee.std_logic_arith.all; -- Note: uncomment/modify this if you use
                                 -- Synopsys signed or unsigned types.
use std.textio.all;

entity TESTBNCH is
end TESTBNCH;

architecture stimulus of TESTBNCH is
component SHIFT0 is
   port (
      clk: in std_logic;
      pin: in std_logic_vector(7 downto 0);
      load_shift0: in std_logic;
      reset1_shift0: in std_logic;
      pout: out std_logic_vector(7 downto 0);
      sout: out std_logic_vector(7 downto 0)
   );

end component;
constant PERIOD: time := 100 ns;



        signal w_clk          : std_logic :='0';
        signal w_reset1_shift0        : std_logic;
        signal w_pin          : std_logic_vector (7 downto 0);
        signal w_load_shift0          : std_logic;
        signal w_pout    : std_logic_vector (7 downto 0);
        signal w_sout         : std_logic_vector (7 downto 0);

signal done: boolean := false;

begin
   DUT: SHIFT0 port map (
        Clk             => w_clk,
                Reset1_shift0       => w_Reset1_shift0,
```

```
            pin         => w_pin,
                load_shift0     -> w_load_shift0,
                pout        => w_pout,
                sout        => w_sout);

w_clk <= not w_clk after period/2;


STIMULUS1: process
  begin

        w_reset1_shift0 <= '1'; --reset dalam keadaan 'high state' iaitu dalam keadaan
awal
            w_pin <= "11111111";
            w_load_shift0 <= '1';

     wait for period;

            w_reset1_shift0 <= '0';
            w_pin <= "10101010";
            w_load_shift0 <= '0';-- data dimuatkan ke dalam register apabila nilai load
adalah 'low'
                        -- selain daripada nilai load='0'; output adalah 00000000

        wait for period;

            w_reset1_shift0 <= '0';
            w_pin <= "11111010";
            w_load_shift0 <= '0';-- data dimuatkan ke dalam register apabila nilai load
adalah 'low'
                        -- selain daripada nilai load='0'; output adalah 00000000

        wait for period;

            w_reset1_shift0 <= '0';
            w_pin <= "10101111";
            w_load_shift0 <= '1';-- data dimuatkan ke dalam register apabila nilai load
adalah 'low'
                        -- selain daripada nilai load='0'; output adalah 00000000

        wait for period;

            w_reset1_shift0 <= '0';
            w_pin <= "10111010";
            w_load_shift0 <= '0';-- data dimuatkan ke dalam register apabila nilai load
adalah 'low'
```

```
                                -- selain daripada nilai load='0'; output adalah 00000000

        wait for period;


        wait;
        end process STIMULUS1;

end stimulus;

configuration CFG_TESTBNCH of TESTBNCH is
    for stimulus
    end for;
end CFG_TESTBNCH;
```

APPENDIK B Test bench

TESTSHIFT1.VHD

```
-------------------------------------------------------
-- Auto-generated test bench template for: SHIFT1
--
-- Note: comments beginning with WIZ should be left intact.
--       Other comments are informational only.
--
-- PeakVHDL software version: 5.20c
--
library ieee;
use ieee.std_logic_1164.all;
-- use ieee.numeric_std.all;     -- Note: uncomment this if you use
                                 -- IEEE standard signed or unsigned types.
-- use ieee.std_logic_arith.all; -- Note: uncomment/modify this if you use
                                 -- Synopsys signed or unsigned types.
use std.textio.all;
use work.SHIFT1;

entity TESTBNCH is
end TESTBNCH;

architecture stimulus of TESTBNCH is
component SHIFT1 is
   port (
      clk: in std_logic;
      sin: in std_logic_vector(7 downto 0);
      load: in std_logic;
      reset1: in std_logic;
      par_out: out std_logic_vector(7 downto 0);
      sout2: out std_logic_vector(7 downto 0)
   );

end component;
constant PERIOD: time := 100 ns;



        signal w_clk          : std_logic :='0';
        signal w_reset1       : std_logic;
        signal w_sin          : std_logic_vector (7 downto 0);
        signal w_load         : std_logic;
```

```
        signal w_par_out    : std_logic_vector (7 downto 0);
        signal w_sout2      : std_logic_vector (7 downto 0);

signal done: boolean := false;

begin
   DUT: SHIFT1 port map (
        Clk             => w_clk,
                Reset1      => w_Reset1,
                sin         => w_sin,
                    load        => w_load,
                    par_out     => w_par_out,
                    sout2       => w_sout2);

w_clk <= not w_clk after period/2;


STIMULUS1: process
   begin

        w_reset1 <= '1'; --reset dalam keadaan 'high state' iaitu dalam keadaan awal
            w_sin <= "11111111";
            w_load <= '1';

      wait for period;

            w_reset1 <= '0';
            w_sin <= "10101010";
            w_load <= '0';-- data dimuatkan ke dalam register apabila nilai load adalah
'low'
                        -- selain daripada nilai load='0'; output adalah 00000000

        wait for period;

            w_reset1 <= '0';
            w_sin <= "11111010";
            w_load <= '0';-- data dimuatkan ke dalam register apabila nilai load adalah
'low'
                        -- selain daripada nilai load='0'; output adalah 00000000

        wait for period;

            w_reset1 <= '0';
            w_sin <= "10101111";
            w_load <= '1';-- data dimuatkan ke dalam register apabila nilai load adalah
'low'
```

```
                    -- selain daripada nilai load='0'; output adalah 00000000

        wait for period;

            w_reset1 <= '0';
            w_sin <= "10111010";
            w_load <= '0';-- data dimuatkan ke dalam register apabila nilai load adalah
'low'
                    -- selain daripada nilai load='0'; output adalah 00000000

        wait for period;



        wait;
        end process STIMULUS1;

end stimulus;

configuration CFG_TESTBNCH1 of TESTBNCH1 is
    for stimulus
    end for;
end CFG_TESTBNCH;
```

APPENDIK B Test bench

TESTCOMPARE.VHD

```
library IEEE;
use IEEE.std_logic_1164.all;

entity TB_COMPARE is
end TB_COMPARE;

architecture BEH of TB_COMPARE is

  component COMPARE
    port(A  : in std_logic_vector ( 7 downto 0 );
         B  : in std_logic_vector ( 7 downto 0 );
         Z  : out std_logic );

  end component;


  constant PERIOD : time := 50 ns;

  signal signal_A  : std_logic_vector ( 7 downto 0 );
  signal signal_B  : std_logic_vector ( 7 downto 0 );
  signal signal_Z  : std_logic ;

begin

  DUT : COMPARE
    port map(A  => signal_A,
             B  => signal_B,
             Z  -> signal_Z);



  STIMULI : process
  begin
    signal_A  <= "10100000";
    signal_B  <= "10101000";

    wait for PERIOD;

     signal_A  <= "00000000";
    signal_B  <= "00000000";
```

```
wait for PERIOD;

  signal_A <= "10100100";
signal_B <= "10101000";

wait for PERIOD;

  signal_A <= "10100000";
signal_B <= "10100000";

wait for PERIOD;

  signal_A <= "10101010";
signal_B <= "10100100";

wait for PERIOD;

  signal_A <= "11111111";
signal_B <= "11111111";

wait for PERIOD;

  signal_A <= "10101110";
signal_B <= "10100000";

wait for PERIOD;

  signal_A <= "10100000";
signal_B <= "10100000";

wait for PERIOD;

  signal_A <= "10100110";
signal_B <= "10100110";

wait for PERIOD;

  signal_A <= "10100100";
signal_B <= "10100100";

wait for PERIOD;

  signal_A <= "10101100";
signal_B <= "10101000";
```

```
    wait for PERIOD;


    wait;
  end process STIMULI;

end BEH;
```

APPENDIK B Test bench

TESTCOUNTER.VHD

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;

entity TB_COUNTER is

end TB_COUNTER;

architecture structural of TB_COUNTER is

  component COUNTER


    port(    CLK              : in std_logic ;
        COUNT_PIXELS          : in std_logic ;
        PIXSI                 : in std_logic ;
        RESET2                : in std_logic ;
            COUNTOUT          : out std_logic_vector ( 7 downto 0 ));


  end component;


  constant PERIOD : time := 100 ns;

  signal W_CLK                : std_logic  := '0';
  signal W_COUNTOUT           : std_logic_vector ( 7 downto 0 );
  signal W_COUNT_PIXELS       : std_logic ;
  signal W_PIXSI              : std_logic ;
  signal W_RESET2             : std_logic ;


begin

  DUT : COUNTER
```

```
  port map(CLK              => W_CLK,
        COUNTOUT   -> W_COUNTOUT,
        COUNT_PIXELS        => W_COUNT_PIXELS,
        PIXSI             => W_PIXSI,
        RESET2              -> W_RESET2);


W_CLK <= not W_CLK after PERIOD/2;

STIMULI : process
begin
  W_COUNT_PIXELS        <= '0';
  W_PIXSI             <= '0';
  W_RESET2              <= '0';


  wait for PERIOD;

    W_COUNT_PIXELS        <= '0';
  W_PIXSI             <= '1';
  W_RESET2              <= '1';

    wait for period;

    W_COUNT_PIXELS        <= '1';
  W_PIXSI             <= '1';
  W_RESET2              <= '1';


  wait for PERIOD;

    W_COUNT_PIXELS        <= '1';
  W_PIXSI             <= '1';
  W_RESET2              <= '1';


  wait for PERIOD;

    W_COUNT_PIXELS        <= '1';
  W_PIXSI             <= '1';
  W_RESET2              <= '1';


  wait for PERIOD;

    W_COUNT_PIXELS        <= '0';
```

```
W_PIXSI             <= '0';
W_RESET2            <= '1';

wait for PERIOD;

  W_COUNT_PIXELS        <= '1';
W_PIXSI             <= '0';
W_RESET2            <= '1';

wait for PERIOD;

  W_COUNT_PIXELS        <= '0';
W_PIXSI             <= '1';
W_RESET2            <= '1';

wait for PERIOD;

  W_COUNT_PIXELS        <= '1';
W_PIXSI             <= '1';
W_RESET2            <= '1';

wait for PERIOD;

  W_COUNT_PIXELS        <= '1';
W_PIXSI             <= '1';
W_RESET2            <= '1';

  wait for period;

  W_COUNT_PIXELS        <= '0';
W_PIXSI             <= '0';
W_RESET2            <= '1';

  wait for period;

  W_COUNT_PIXELS        <= '1';
W_PIXSI             <= '1';
W_RESET2            <= '1';

wait for PERIOD;
```

```
    W_COUNT_PIXELS      <= '1';
    W_PIXSI             <= '0';
    W_RESET2            <= '1';

    wait for PERIOD;

    wait;
  end process STIMULI;

end structural;
```

APPENDIK B Test bench

TESTMUXC.VHD

```
library IEEE;
use IEEE.std_logic_1164.all;

entity TB_MUX is
end TB_MUX;

architecture BEH of TB_MUX is

  component MUX
    port(pixel    : in std_logic_vector ( 7 DOWNTO 0 );
         count    : in std_logic_vector ( 7 DOWNTO 0 );
         SEL      : in std_logic ;
         OUTPUT   : out std_logic_vector ( 7 DOWNTO 0 ) );

  end component;


  constant PERIOD : time := 100 ns;

  signal W_pixel    : std_logic_vector ( 7 downto 0 );
  signal W_count    : std_logic_vector ( 7 downto 0 );
  signal W_SEL      : std_logic ;
  signal W_OUTPUT   : std_logic_vector ( 7 downto 0 ) ;

begin

  DUT : MUX
    port map(pixel      => W_pixel,
             count      => W_count,
             SEL        => W_SEL,
             OUTPUT     => W_OUTPUT);


  STIMULI : process
  begin
    W_pixel    <= "01000101";
    W_count    <= "10110110";
    W_SEL      <= '1';

    wait for PERIOD;
```

```
    wait;
  end process STIMULI;

end BEH;

configuration CFG_TB_MUX of TB_MUX is
  for BEH
  end for;
end CFG_TB_MUX;
```

APPENDIK B TESTBENCH

TESTCONTROL.VHD

```
library IEEE;
use IEEE.std_logic_1164.all;

entity TB_CONTROL is
end TB_CONTROL;

architecture BEH of TB_CONTROL is

  component CONTROL
    port(CLK          : in std_logic ;
         RESET        : in std_logic ;
         COMPARE      : in std_logic ;
         muxselect_1 : out std_logic;
                       valid_1 : out std_logic;
                       load_shift0_1 : out std_logic;
                       reset1_shift0_1 : out std_logic;
                       pixsi_1 : out std_logic;
                       count_pixels_1 : out std_logic;
                       reset2_1 : out std_logic;
                       reset1_1 : out std_logic;
                       load_1 : out std_logic;
                       load_reg_1 : out std_logic;
                       reset_reg_1 : out std_logic);


  end component;

  constant PERIOD : time := 100 ns;

  signal W_CLK        : std_logic := '0';
  signal W_RESET      : std_logic ;
  signal W_COMPARE    : std_logic ;
  signal w_muxselect_1 : std_logic;
  signal w_valid_1 : std_logic;
  signal w_load_shift0_1 : std_logic;
  signal W_reset1_shift0_1 : std_logic;
  signal W_pixsi_1    : std_logic;
  signal W_count_pixels_1 : std_logic;
  signal W_reset2_1 : std_logic;
  signal w_reset1_1 : std_logic;
```

```
signal w_load_1 : std_logic;
signal w_load_reg_1 : std_logic;
signal w_reset_reg_1 : std_logic;


begin

DUT : CONTROL
  port map(CLK        -> W_CLK,
        RESET      => W_RESET,
        COMPARE    => W_COMPARE,
              muxselect_1 -> w_muxselect_1,
                             valid_1   => w_valid_1,
                             load_shift0_1 => w_load_shift0_1,
                             reset1_shift0_1 => w_reset1_shift0_1,
                             pixsi_1 => w_pixsi_1,
                             count_pixels_1 -> w_count_pixels_1,
                             reset2_1 => w_reset2_1,
                             reset1_1 => w_reset1_1,
                             load_1 -> w_load_1,
                             load_reg_1 => w_load_reg_1,
                             reset_reg_1 => w_load_reg_1);



W_CLK <= not W_CLK after PERIOD/2;

STIMULI : process
begin
  W_RESET     <= '1';
  W_COMPARE   <= '0';


  wait for PERIOD;

       W_RESET     <= '0';
  W_COMPARE   <= '1';


  wait for PERIOD;

       W_RESET     <= '0';
  W_COMPARE   <= '1';
```

```
    wait for PERIOD;

        W_RESET     <= '0';
    W_COMPARE   <= '0';

    wait for PERIOD;

    W_RESET     <= '0';
    W_COMPARE   <= '0';
    wait for PERIOD;

        W_RESET     <= '0';
    W_COMPARE   <= '1';

    wait for PERIOD;

        W_RESET     <= '0';
    W_COMPARE   <= '1';

    wait for PERIOD;



    wait;
  end process STIMULI;

end BEH;

configuration CFG_TB_CONTROL of TB_CONTROL is
  for BEH
  end for;
end CFG_TB_CONTROL;
```

APPENDIK B Test bench

TESTCOUNTREG.VHD

```
library ieee;
use ieee.std_logic_1164.all;
-- use ieee.numeric_std.all;     -- Note: uncomment this if you use
                                 -- IEEE standard signed or unsigned types.
-- use ieee.std_logic_arith.all; -- Note: uncomment/modify this if you use
                                 -- Synopsys signed or unsigned types.
use std.textio.all;

entity TESTBNCH is
end TESTBNCH;

architecture stimulus of TESTBNCH is
component countreg is
   port (
      clk: in std_logic;
      countregin: in std_logic_vector(7 downto 0);
      load_reg: in std_logic;
      reset_reg: in std_logic;
      countout2: out std_logic_vector(7 downto 0)
   );

end component;
constant PERIOD: time := 100 ns;


                signal w_clk            : std_logic :='0';
                signal w_reset_reg              : std_logic;
                signal w_countregin :  std_logic_vector (7 downto 0);
                signal w_load_reg               : std_logic;
                signal w_countout2    : std_logic_vector (7 downto 0);

signal done: boolean := false;

begin
   DUT: countreg port map (
        Clk                     =>      w_clk,
                                Reset_reg               =>      w_Reset_reg,
                                countregin                      =>      w_countregin,
                                        load_reg                =>      w_load_reg,
```

```
                countout2          =>      w_countout2);

w_clk <= not w_clk after period/2;


STIMULUS1: process
  begin

                w_reset_reg <= '1';
                w_countregin <= "11111111";
                w_load_reg <= '1';

                wait for period;

                w_reset_reg <= '0';
                w_countregin <= "11011011";
                w_load_reg <= '0';

                wait for period;

                w_countregin <= "11100111";
                w_load_reg <= '0';

                wait for period;



        wait;
        end process STIMULUS1;

end stimulus;

configuration CFG_TESTBNCH of TESTBNCH is
        for stimulus
        end for;
end CFG_TESTBNCH;
```

APPENDIK B Test bench

```
--------------------------------------------------------------------------------
TESTRLETOPLEVEL.VHD
--------------------------------------------------------------------------------

library IEEE;
use IEEE.std_logic_1164.all;

entity TB_RLETOPLEVEL is
end TB_RLETOPLEVEL;

architecture structural of TB_RLETOPLEVEL is

  component RLETOPLEVEL
    port(CLK : in std_logic;
                    PIN    : in std_logic_vector(7 downto 0) ;
         RESET  : in std_logic ;
         output   : out std_logic_vector(7 downto 0) ;
         VALID  : out std_logic
               ) ;


  end component;


  constant PERIOD : time := 100 ns;
       signal W_CLK : std_logic := '0';
  signal W_PIN    : std_logic_vector(7 downto 0);
  signal W_RESET  : std_logic ;
  signal W_output   : std_logic_vector(7 downto 0) ;
  signal W_VALID  : std_logic ;


begin

  DUT : RLETOPLEVEL
    port map(clk => w_clk,
                          PIN    => W_PIN,
          RESET  => W_RESET,
          output   => W_output,
          VALID  => W_VALID);

               w_clk <= not w_clk after period/2;
```

```
STIMULI : process
begin
        W_PIN    <= "10010011";
        W_RESET  <= '0';

        wait for PERIOD;

                W_PIN    <= "10010011";
        W_RESET  <= '0';

        wait for PERIOD;

                W_PIN    <= "10010011";
         --W_RESET  <= '0';

        wait for PERIOD;

                W_PIN    <= "10110011";
    --  W_RESET  <= '0';

        wait for PERIOD;

                W_PIN    <= "10110011";
        --  W_RESET  <= '0';

        wait for PERIOD;


        --      W_PIN    <= "01101000";
    --  W_RESET  <= '0';

    --  wait for PERIOD;


     wait;
  end process STIMULI;

end structural;

configuration CFG_TB_RLETOPLEVEL of TB_RLETOPLEVEL is
  for structural
  end for;
end CFG_TB_RLETOPLEVEL;
```