WXES 3182

Mekanisma Pegujian ALU menggunakan Teknik Boundary Scan
(*ALU Testing Mechanism Using Boundary Scan Techniqe*)

Oleh:
Ahmad Zaki Zamani bin Abdul Aziz
WEK 010350

Penyelia: En Mohd Yamani Idna Idris
Moderator: Dr. Phang Keat Keong

ABSTRAK

Mekanisma Pengujian ALU Menggunakan Teknik Boundary Scan merupakan projek yang dibangunkan untuk menguji litar digital, litar bersepadu dan juga cip. Matalamat pembangunan projek ini adalah untuk menghasilkan satu mekanisma yang dapat menentukan samada sesebuah cip tersebut berada dalam keadaan baik atau sebaliknya. Projek ini terbahagi kepada dua bahagian iaitu pembinaan ALU dan juga pembangunan Boundary Scan.

ALU merupakan satu unit atau komponen litar digital atau litar bersepadu yang akan dibangunkan menggunakan bahasa pengaturcaraan VHDL atau *Very High Speed Integration Circuit (VHSIC) Hardware Description Language*. Ia akan melaksanakan operasi aritmetik dan logik bagi data 8-bit

Boundary Scan pula adalah platform yang akan menguji litar ALU samada ia mengalami ralat atau pun tidak. Ia juga dibangunkan menggunakan bahasa pengaturcaraan VHDL atau *Very High Speed Integration Circuit (VHSIC) Hardware Description Language*. Projek ini akan menghasilkan satu sintesis yang berbentuk litar logik.

ALU akan dibangunkan oleh penulis iaitu Ahmad Zaki Zamani manakala Boundary Scan akan dibangunkan oleh saudara Wan Muhammad Idzuan. Didalam laporan setebal 105 muka surat ini ia akan menyentuh semua aspek pembangunan ALU dari awal hingga keakhir.

ABSTRACT

ALU testing mechanism using Boundary Scan technique is a project which is develop to be the tester for testing the digital circuit, integrated circuit and computer chip. The goal of this project is to develop a mechanism and as well building an ALU chip so that we will able to test and determine whether the chips are in a good shape and condition. This project contains two parts; one is to develop an ALU and second is to develop the boundary scan chip.

ALU is a unit or a component for a digital circuit or a integrated circuit which is develop using VHDL or *Very High Speed Integration Circuit (VHSIC) Hardware Description Language*. It will be executing arithmetic and logic operations for 8-bits data.

Boundary Scan on the other hand is a platform that will test the ALU chip whether is functioning or not. It is also will be develop using VHDL or *Very High Speed Integration Circuit (VHSIC) Hardware Description Language*. This project will display or produce a synthesis of a logic circuit.

The writer which is Ahmad Zaki Zamani will be developing the ALU and Wan Muhammad Idzuan will be developing the Boundary Scan. In this 105 pages report it will cover all of the aspect of the ALU development from the start till the end of it.

PENGHARGAAN

Alhamdulillah, syukur kepada allah swt yang telah mengizinkan saya untuk menyiapkan laporan Latihan Ilmiah 2 (WXES 3182) yang bertajuk ***Mekanisma Pengujian ALU Menggunakan Teknik Boundary Scan.*** Sesungguhnya tanpa kekuatan darinya maka sukarlah projek dan juga laporan ini dapat dihasilkan.

Pertama sekali saya ingin mengucapkan jutaan terima kasih kepada penyelia latihan ilmiah ini iaitu Encik Yamani Idna Idris dan juga Dr. Phang Keat Keong selaku moderator projek ini yang telah banyak bantuan serta tunjuk ajar kepada saya dalam melaksanakan latihan ilmiah ini. Sesungguhnya, bantuan dan tunjuk ajar mereka telah banyak membantu saya dalam melaksanakan latihan ilmiah ini.

Saya menggunakan kesempatan ini juga untuk mengucapkan ribuan terima kasih kepada rakan seperjuangan saya iaitu saudara Wan Muhammad Idzuan yang telah sama-sama berusaha malaksanakan latihan ilmiah ini tanpa mengira masa dan ketika dalam menjayakan pembangunan ***Mekanisma Pengujian ALU Menggunakan Teknik Boundary Scan.***

Seterusnya, saya ingin mengucapkan berbanyak terima kasih kepada semua rakan-rakan seperjuangan yang lain terutamanya Nor Wany Affinda, Fadzlinnisa', Nurul Akmal, Nazlin Irham, Khairul Anwar, Mohd Yohan, Aidill, Zarina, Rajanorazrina, Siti, Nurul Hani, Nurul Ain, Marina dan Ahmad Ridhuan yang telah sama-sama melalui latihan

ilmiah ini dan semua yang telah terlibat secara langsung atau tidak langsung dalam melaksanakan ***Mekanisma Pengujian ALU Menggunakan Teknik Boundary Scan*** ini.

Ucapan terima kasih yang tak terhingga juga saya tujukan buat insan tersayang iaitu Shahirah Zainudi yang telah memberi sokongan kepada saya dari awal hingga keakhir latihan ilmiah ini. Segala sokongan dan bantuan yang telah diberikan olehmu akan dikenang buat selama-lamanya.

Akhir sekali saya ingin mengucapkan ribuan terima kasih terima kasih yang teristimewa saya kepada kedua ibu bapa saya iaitu Abdul Aziz Johan dan Rodziahtun Adawiah yang telah mendoakan saya sepanjang masa dan juga telah memberikan segala sokongan yang saya perlukan baik dari segi moral dan material sepanjang saya melaksanakan latihan ilmiah ini. Tanpa bantuan mereka latihan ilmiah in mungkin agak sukar untuk dilaksanakan.

Sebagai penutup, saya ingin mengucapkan jutaan terima kasih kepada semua pihak yang telah terlibat dalam ***Mekanisma Pengujian ALU Menggunakan Teknik Boundary Scan*** ini samada secara langsung atau tidak langsung. Sekian terima kasih.

AHMAD ZAKI ZAMANI ABDUL AZIZ
WEK010350
Jabatan Sistem Dan Rangkaian
Fakulti Sains Komputer Dan Teknologi Maklumat
Universiti Malaya.

ISI KANDUNGAN

Senarai Rajah

Senarai Jadual

BAB 1
PENDAHULUAN

BAB 1

PENDAHULUAN

1.0 PENGENALAN.

Perkembangan teknologi masa kini telah menyaksikan kepesatan penggunaan litar bersepadu. Pada masa kini terdapat banyak syarikat pengeluar cip-cip yang menggunakan teknologi litar bersepadu yang mampu melaksanakan arahan dan fungsi yang banyak. Namun begitu ia tidak terlepas daripada cabaran bagi memastikan cip yang dihasilkan itu bebas daripada ralat dan melaksanakan fungsinya seperti yang telah direkabentuk.

Untuk tujuan itu, cip-cip tersebut perlulah diuji dari peringkat awal lagi bagi memastikan ia dapat berfungsi dengan baik. Adalah diketahui bahawa, semakin kompleks sesebuah litar tersebut dibina maka pengujian keatas cip tersebut akan menjadi sukar dan kos untuk mengujinya juga akan meningkat.

Untuk melakukan pengujian terhadap cip, satu kaedah iaitu boundary scan akan digunakan bagi menguji litar tersebut. Dengan adanya kaedah ini ia akan dapat menentukan samaada sesebuah litar tersebut mengalami kegagalan fungsi atau tidak.

Untuk permulaan projek ini satu unit aritmetik logik 8 bit (ALU-Arithmetic Logic Unit) akan digunakan sebagai modul ujian bagi kaedah pengujian Boundary Scan ini.

1.1 SKOP PROJEK

Didalam projek ini pembangunan akan meliputi pembinaan ALU 8 bit yang mana ia akan melaksanakan operasi aritmetik dan operasi logik. Jadual satu menujukkan operasi aritmetik dan logik yang boleh dikendalikan oleh ALU 8 bit yang dibangunkan.

OPERASI ARITMETIK	OPERASI LOGIK
UMPUKAN	EKSLUSIF-ATAU
PENGURANGAN	DAN
PENAMBAHAN	ATAU
PENAMBAHAN BERSAMA PELENGKAP SATUAN BAGI OPERAN KEDUA	TAK (PELENGKAP SATUAN)
PENAMBAHAN BERSAMA NILAI SAMBUTAN 1	
PENOLAKAN	
PENGURANGAN	

Jadual 1.1: Operasi ALU

Seterusnya, pembangunan sistem pengujian Boundary scan juga akan dilaksanakan dalam projek ini dimana ia akan menjadi nadi utama projek ini kerana ia akan bertanggungjawab dalam melaksanakan pengujian keatas litar ALU. Komponen penting bagi Boundaray scan ini adalah terdiri daripada

- Test Access Port (TAP)
- Pengawal TAP (TAP Controller)
- Daftar Arahan (Instruction Register)
- Daftar Data Ujian (Test Data Register)

Projek ini akan dilaksanakan hingga kepada pembinaan sebuah prototaip ALU dan BOUNDARY SCAN dimana bahasa pengaturcaraan yang akan digunakan untuk

pembangunan adalah VHSIC Hardware Description Language. Ianya akan dilaksanakan dalam bentuk model dan akan disimulasikan.

Untuk kesimpulannya pembangunan projek ini terdiri kepada beberapa sub proses:

- Pembangunan ALU
- Pembangunan JTAG Boundary Scan
- Intergrasi sistem
- Pengujian Sistem

1.2 OBJEKTIF PROJEK

Matlamat utama projek ini adalah untuk menghasilkan sebuah litar pengujian cip menggunakan teknik Boundary Scan dimana ia akan digunakan untuk menguji ALU 8 bit dengan menggunakan bahasa pengaturcaraan VHDL semasa membangunkan model projek ini.

Antara objektif utama projek yang telah ditetapkan adalah:

- Merekabentuk satu mekanisma untuk menguji unit aritmetik menggunakan kaedah *boundary scan*..
- Merekabentuk dan membangunkan satu model ALU menggunakan bahasa pengaturcaan VHDL.

- Mendapatkan gambaran sebenar pengujian yang dijalankan semasa pengujian dijalankan melalui keputusan yang dikeluarkan.
- Membuat perbandingan pengujian antara *boundry scan* dengan BIST

Selain itu, projek ini dapat dijadikan sebagai perintis kepada pembangunan projek-projek yang lebih dimasa hadapan.

1.3 PENJADUALAN PROJEK

Penjadualan pembangunan Boundary Scan untuk ALU amat penting bagi memastikan semua fasa pembangunan projek dilaksanakan dalam jangkamasa yang telah ditetapkan dan mengikut fasa pembangunan projek. Kerja-kerja perlaksanaan untuk Projek ini akan mengambil masa hampir 8 bulan untuk disiapkan sepenuhnya. Bermula dengan penyiasatan awal hingga kepada dokumenatsi projek. Antara fasa kerja yang terlibat adalah:-

1. Penyiasatan awal
2. Analisis keperluan
3. Rekabentuk
4. Pembangunan modul
5. Pengujian intergrasi
6. Dokumentasi

FASA	PERKARA	Julai 2004	Ogos 2004	Sept 2004	Okt 2004	Nov 2004	Dis 2004	Jan 2005	Feb 2005
1	Penyiasatan awal								
2	Analisis keperluan								
3	Rekabentuk modul								
4	Pembangunan modul								
5	Pengujian intergrasi								
6	Dokumentasi								

Jadual 1.2: Pembangunan Projek

FASA 1

Dalam fasa ini, penyiasatan dilakukan keatas sistem yang ingin dibangunkan. Segala maklumat mengenai JTAG Boundary Scan dan ALU akan dikumpulkan. Maklumat di pada pembacaan buku dan juga jurnal dan artikel daripada sumber yang sahih dari internet akan dikumpulkan bagi mendapatkan gambaran awal untuk membantu dalam pembangunan projek. Seterusnya maklumat ini akan di bentangkan kepada penyelia projek iaitu Encik Yamani untuk mendapatkan nasihat dan panduan daripada beliau.

FASA 2

Setelah maklumat telah diperolehi, fasa berikutnya pula melibatkan analisis terhadap sistem yang akan dilakukan. segala keperluan akan dikenalpasti dalam pembangunan projek ini dan perbincangan akan diadakan dengan penyelia projek untuk sebarang pengubahsuaian.

FASA 3

Semasa dalam fasa ini reka bentuk sistem yang akan dibangunkan akan dilakukan ia adalah tindakan lanjutan daripada fasa sebelumnya. Rekabentuk yang telah dikenalpasti akan dibawa kepada penyelia projek untuk pemeriksaan dan sebarang perubahan akan dilakukan dengan panduan daripada penyelia.

FASA 4&5

Pembangunan seterusnya pula akan melibatkan dua fasa iaitu fasa 4 dan 5. Ia dilakukan bersama kerana ia mempunyai kaitan rapat antara satu sama lain. Fasa pengujian diperlukan dalah kerana perlaksanaan sistem memerlukan proses sokongan ini bagi memastikan sistem yang dibangunkan akan beroperasi dengan baik dan bebas daripada ralat.

FASA 6

Setelah fasa pengujian dan pembangunan telah dilaksanakan dengan jayanya maka proses dokumentasi haruslah dilaksanakan bagi menyediakan sistem dengan dokumen sokongan bagi memudahkan pengendaliannya kelak dan juga untuk tujuan peningkatupayaan sistem dimasa hadapan.

1.4 KEKANGAN & MASALAH

Merekabentuk mekanisme pengujian ALU menggunakan teknik boundary scan akan menimbulkan beberapa masalah yang berkaitan dengan pengujian. Antaranya adalah:

- ALU tidak dapat melakukan pengoperasian pembahagian nombor integer
- ALU tidak dapat melakukan pengoperasian *floating point*
- Boundary scan tidak dapat melakukan pengujian untuk senibina dalaman sesuatu cip dengan lebih baik berbanding teknik *Built-In Self-Test* (BIST).
- Pembinaan litar yang perlu memenuhi kriteria pengujian
- Memulakan proses rekabentuk dari awal untuk memastikan rekabentuk yang dihasilkan dapat menepati objektif pengujian.

1.5 RINGKASAN BAB

Laporan ini mengandungi lima bab dimana iakan membincangkan setiap prose dalam pembangunan sistem bagi LAtihan Ilmiah tahap Akhir (WXES 3181)

Bab 1 merupakan pendahuluan untuk keseluruhan laporan ini. Didalamnya terdapat penerangan mengenai objektif, skop projek mekanisma pengujian ALU menggunakan teknik Boundary Scan ini. Fasa-fasa pembangunan yang terlibat dalam pembangunan projek ini juga dimuatkan didalam bab ini.

Bab 2 pula mebentangkan laporan hasil kajian dan penyelidikan yang telak dilakukan bagi membangunkan projek ini. Didalam bab ini topik mengenai pengujian, ALU, RCA, CLA dan Boundary Scan akan dihuraikan.

Bab 3 pula membincangkan metodologi yang akan terlibat dalam proses pembangunan dan pembahagian tugas untuk membangunkan projek Mekanisma pengujian ALU menggunakan teknik Boundary Scan ini juga akan dijelaskan didalam bab ini

Bab 4 pula mengariskan analisis tehadap sistem yang akan dibagunkan. Ia meliputi segala keperluan sistem.

Bab 5 pula menerangkan mengenai reka bentuk sistem yanga akan dibangunkan. Ia akan merangkumi ALU dan juga Boundary Scan.

1.6 KESIMPULAN

Sebagai kesimpulan awal, projek ini merupakan satu langkah pengaplikasian teknik untu mereka satu sistem yang dapat menguji sesebuah cip dimana dalam kontek projek ini sebuah ALU. Ianya adalah satu perintis jalan bagi melaksanakan projek yang lebih kompleks pada masa akan datang menggunakan teknik Boundary Scan ini.

BAB 2
KAJIAN LITERASI

BAB 2

KAJIAN LITERASI

2.0 PENGENALAN

Kajian literasi untuk pembinaan **ALU** menggunakan perlu dirancang dengan teliti dan dilakukan dengan sempurna. Ini adalah untuk memastikan matlamat pembangunan dan objektifnya akan dapat dicapai. Kajian dan pengumpulan maklumat mengenai konsep, teknik, perisian dan alatan yang diperlukan juga perlu dilakukan. Setelah semua maklumat tersebut diperolehi, proses menganalisa pula akan dilakukan. Kajian literasi dimulakan dengan menyenaraikan semua sumber maklumat yang mempunyai kaitan dengan kajian yang sedang dijalankan . Sumber yang telah dikenalpasti ialah:-

- Artikel-artikel yang berkaitan dengan **ALU**.
- Kaedah pengujian yang pernah dilakukan.
- Projek ilmiah pelajar terdahulu.

Selain itu kajian literasi juga tertumpu pada skop pembangunan ALU, teknik pengujian yang sesuai dengan ALU dan komponen-komponen yang diperlukan untuk mengaplikasikan pembangunan mekanisma pengujian alu menggunakan teknik boundary scan. Tumpuan utama diberikan kepada komponen asas pembangunan ALU iaitu:

- Unit Aritmetik & Logik (ALU).
- Penambah (*Adder*)

2.1 FAKTOR PENDORONG REKABENTUK BOLEH UJI

Faktor pengujian litar bersepadu bergantung kepada rekabentuk penghasilan litar itu sendiri kerana apabila sesuatu litar itu semakin kompleks, maka pengujian yang akan dilakukan adalah lebih sukar. Ia adalah kerana rekabentuk litar yang kompleks mempunyai bilangan get yang lebih banyak dan bilangan pinnya juga menjadi semakin bertambah. Oleh sebab itu masa perlaksanaan yang diambil bagi sesuatu pengujian adalah semakin tinggi. Faktor ini juga diambil kira dalam penentuan kaedah rekabentuk pengujian yang akan dilakukan.

Rekabentuk pengujian merupakan cara untuk menguji litar pada tahap-tahap tertentu. Pengujian yang baik haruslah mempunyai nilai liputan kegagalan yang rendah bagi setiap pengujian yang dilakukan. Liputan kegagalan adalah merujuk kepada nisbah kegagalan yang dikesan kepada jumlah bilangan kegagalan yang mungkin. Pada kebiasaannya, kaedah pengujian yang baik menghasilkan liputan kegagalan melebihi 95%. Pengiraan liputan kawalan adalah diwakili seperti persamaan di bawah ini.

$$\text{Liputan kegagalan} = \frac{\text{Bilangan kegagalan yang di kesan}}{\text{Bilangan kegagalan yang mungkin}} \mathbf{X}\ 100\%$$

2.3 PRINSIP & MATLAMAT REKABENTUK BOLEH UJI

Tujuan utama rekabentuk boleh uji adalah untuk meningkatkan kualiti dan kebolehpercayaan keluaran sesuatu produk itu sendiri sambil mengurangkan kos penyelenggaraan. Sesebuah rekabentuk pengujian yang baik haruslah menepati ciri-ciri yang telah ditetapkan. Antaranya adalah:

- Pengendalian ujian yang mudah
- Masa ujian yang rendah
- Kebolehan mengadakan pengujian pada kelajuan yang berbeza
- Tokokan liputan kegagalan (pengesanan dan penempatan)
- Corak ujian yang mudah dijana
- Penilai ujian yang boleh dipercayai, mudah dan kos asas (*overhead)* perkakasan yang mudah.

Terdapat dua parameter utama dalam ujian rekabentuk iaitu kebolehcerapan dan kebolehkawalan. Kebolehcerapan bermaksud kebolehan untuk melihat atau memacu sambutan litar-litar dalaman pada keluaran utama. Untuk kebolehkawalan pula ia bermaksud kebolehan meletakkan suatu litar dalaman kepada suatu keadaan yang diketahui. Contohnya, kebolehan untuk mengadakan ujian pada litar tertentu.

2.4 PENGUJIAN PERKAKASAN DAN REKABENTUK BOLEH UJI

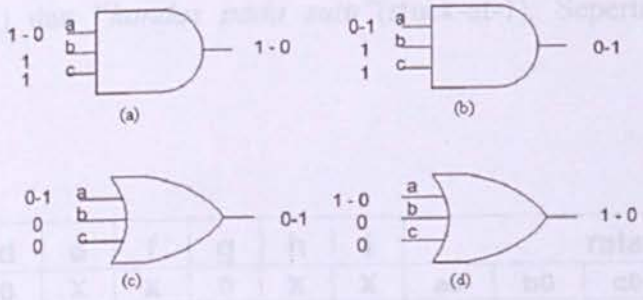
Sebagai permulaan, perlu dinyatakan bahawa sesebuah cip yang dihasilkan mestilah mampu melaksanakan segala fungsi yang telah direkabentuk. Namun begitu cabaran utama adalah jaminan kualiti dan ketepatan hasil yang diharapkan adalah ukuran sebenar bagi cip tersebut bagi menjamin kualiti, prestasi, kebolehpercayaan dan kos pengeluaran adalah optimum, Bagi syarikat pengeluar cip, faktor inilah yang mendorong kepada mereka untuk menghasilkan sistem pengujian ini bagi tujuan pengawalan kualiti. Sebuah cip yang mempu melaksanakan pelbagai fungsi dan tugas, cip tersebut memerlukan bilangan get logik yang mencecah ratusan. Maka pengujian yang dilakukan adalah kompleks dan sukar.

Kewujudan sebarang ralat didalam sebuah cip atau litar bersepadu bukanlah satu perkara yang mustahil. Sebarang kemungkinan boleh diambil kira, ralat berkemungkinan berlaku semasa proses pembuatan.

2.4.1 PENGUJIAN LITAR LOGIK GABUNGAN

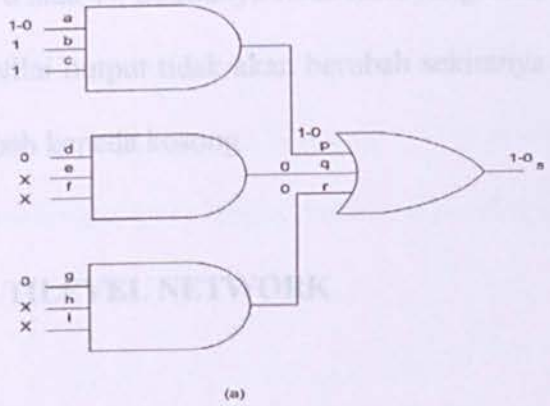
Dalam membincangkan mengenai ralat, dua jenis ralat yang biasa berlaku adalah litar pintas atau litar terbuka. Bagi ralat litar pintas ianya dipengaruhi kemana litar tesebut telah dipintaskan sekira ia dipintaskan pada bumi maka ia kan berada pada keadaan *"kandas pada kosong"*(stuck-at-0). Sekiranya ia terpintas pada sumber kuasa positif maka ia adalah *"kandas pada satu"*(stuck-at-1). Untuk lebih memahami keadan ini,

pertimbangkan pengujian terhadap get logik DAN yang mempunyai tiga input iaitu A,B dan C seperti rajah 2.1. Pada keadaan normal, sekiranya semua input bernilai 1 maka output adalah 1. Jika salah satu atau lebih dari satu input yang diuji mempunyai ralat *"kandas pada kosong"*(stuck-at-0), maka output akan memberikan nilai kosong. Cara yang sama juga boleh digunakan bagi menguji samada get logik DAN mengalani ralat *"kandas pada satu"*(stuck-at-1) dimana pada keadaan normal semua nilai input akan di berikan dengan nilai satu dan sekiranya nilai salah satu input ditukar kepada kosong maka nilai output akan berubah menjadi kosong. Nilai output tidak berubah sekiranya input yang diuji mempunyai ralat *"kandas pada satu"*(stuck-at-1). Gambarajah berikut akan menerangkan bagaimana konsep pengujian *"kandas pada kosong"*(stuck-at-0) dan *kandas pada satu"*(stuck-at-1).



Rajah 2.1 : Pengujian Get Satu Aras

Berdasarkan rajah diatas, pengujian diatas hanya melibatkan satu aras get sahaja, untuk lebih memahami konsep pengujian mengikut aras ini, pertimbangkan satu sambungan get yang membentuk litar dibawah.



Rajah 2.2 : Pengujian Get Dua Aras

Daripada rajah 2.2, kita menganggap bahawa input *p, q* dan *r* pada get atau tidak boleh di capai. Satu cara untuk menguji litar ini adalah *brute force* iaitu dengan melakukan semua $2^9 = 512$ set ujian dan memerhatikan output yang dikeluarkan. Namun cara yang lebih efisyen untuk menguji litar ini adalah menggunakan konsep *"kandas pada kosong"*(stuck-at-0) dan *"kandas pada satu"*(stuck-at-1). Seperti yang di tunjukkan dalam jadual 2.1.

a	b	C	d	e	f	g	h	i	ralat teruji					
1	1	1	0	X	X	0	X	X	a0	b0	c0	p0		
0	X	X	1	1	1	0	X	X	d0	e0	f0	q0		
0	X	X	0	X	X	1	1	1	g0	h0	i0	r0		
0	1	1	0	1	1	0	1	1	a1	d1	g1	p1	q1	r1
1	0	1	1	0	1	1	0	1	b1	e1	h1	p1	q1	r1
1	1	0	1	1	0	1	1	0	c1	f1	i1	p1	q1	r1

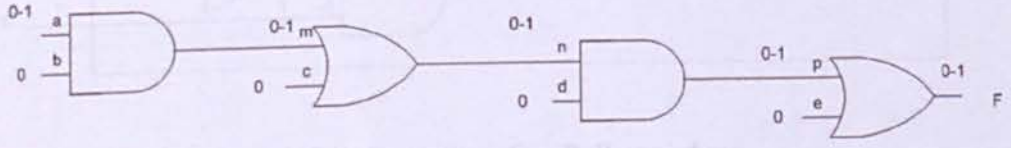
Jadual 2.1: Vektor Ujian Get

Untuk menguji pin input a,b dan c untuk ralat *"kandas pada kosong"*(stuck-at-0) apa yang perllu dilakukan adalah pin *a,b* dan **c** diberikan nilai satu manakala pin-pin *d* dan *g* diberikan nilai kosong manakala *e,f,h* dan *i* dalah daialam keadaan tidak peduli(*don't*

care- nilai adalah samada 0 atau 1). Sekiranya tiada ralat yang berlaku maka output akan menghasilkan nilai satu. Nilai output tidak akan berubah sekiranya ralat berlaku apabila nila mana-mana input diubah kepada kosong.

2.4.2 PENGUJIAN MULTILEVEL NETWORK

Bagi pengujian *multi level*, ianya adalah satu pengujian yang agak sukar daripada litar dua aras. Sekiranya ujian akan dilakukan penggunaan kaedah *brute force* sudah tidak lagi praktikal. Oleh itu satu set ujian haruslah dipilih bagi melaksanakan ujian yang lebih teratur.

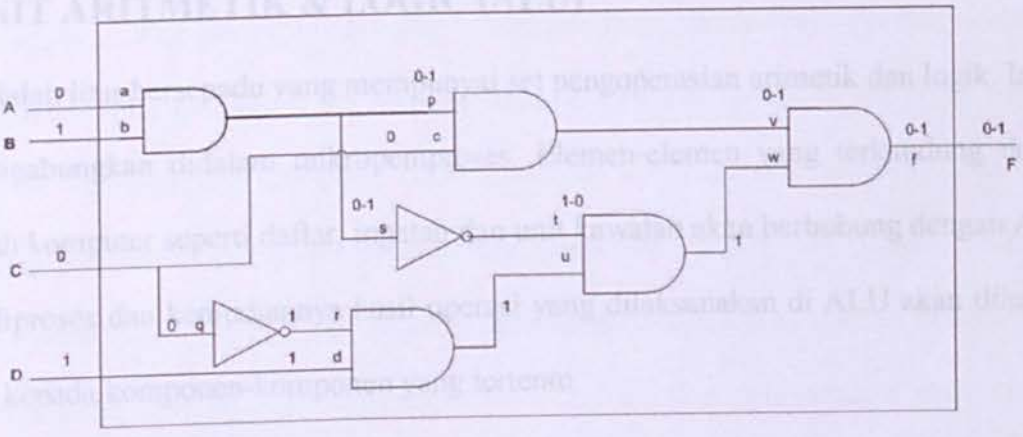


Rajah 2.3: Pengujian Litar Berangkaian

Daripada rajah 2.3 kita dapat lihat bagaimana susunan dilakukan bagi menguji satu laluan input bemula dari a dan b sehingga F. Sebagai contoh sekiranya kita ingin menguji n untuk ralat *kandas pada satu"*(stuck-at-1).nilai input n mestilah bernilai kosong. Untuk itu nilai nilai berikut haruslah diberi iaitu $c = 0$, $a = 0$ dan $b = 1$. untuk menjadikan ralat *kandas pada satu"*(stuck-at-1) pada output F kita mestilah memberi nilai $d = 1$ dan $e =$

0, dengan menggunakan set input ini, sebarang ralat pada input *a,m,n* atau *p* akan menyebabkan output yang salah. Seterusnya sekiranya memberikan input *a* nilai 1 ralat *kandas pada kosong"*(stuck-at-0) pada pin *a,m,n* atau *p* akan dapat dikesan. Kaedah *path sensitization* ini membolehkan kita menguji beberapa pin dengan menggunakan satu set input rangkaian get.

Menggunakan cara ini kita dapat meminimumkan vektor ujian bagi proses pengujian sesebuah cip. Sebagai contoh , kita lihat rajah 2.4 yang berikut.



Rajah 2.4: Pengujian Cip Pelbagai Aras

Daripada gambarajah tersebut, dengan andaian menganggap bahawa input dan output didalam kotak tidak dapat dicapai dan hanya input pada kawasan luaran kotak iaitu A, B, C dan D sahaja yang boleh di berikan nilai. Sekiranya kita ingin menguji input p samada ralat *"kandas pada satu"*(stuck-at-1), segala input pada A, B, C dan D mestilah mengikut nilai yang ditunjukkan didalam rajah 2.4. Dengan menggunakan set ujian ABCD = 0101 maka kita dapat menguji laluan input A-*a*-*p*-*v*-*f*-F. Jadual 2.2 merupakan lima vektor ujian yang akan dapta menguji keseluruhan litar diatas.

A	B	C	D	a	b	p	c	q	r	d	s	t	u	v	w	ralat teruji									
0	1	0	1	0	1	0	0	0	1	1	0	1	1	0	1	a 1	p 1	c 1	v 1	f 1					
1	1	0	1	1	1	1	0	0	1	1	1	0	1	1	1	a 0	b 0	p 0	q 1	r 0	d 0	u 0	v 0	w 0	f 0
1	0	1	1	1	0	0	1	1	0	1	0	1	0	1	1	b 1	c 0	s 1	t 0	v 0	w 0	f0			
1	1	0	0	1	1	1	0	0	1	0	1	0	0	1	0	a 0	b 0	d 1	s 0	t 1	u 1	w 1	f1		
1	1	1	1	1	1	1	1	1	0	1	1	0	0	1	0	a 0	b 0	q 0	r 1	s 0	t1	u 1	w 1	f1	

Jadual 2.2: Vektor Ujian Get

2.5 UNIT ARITMETIK & LOGIC (ALU)

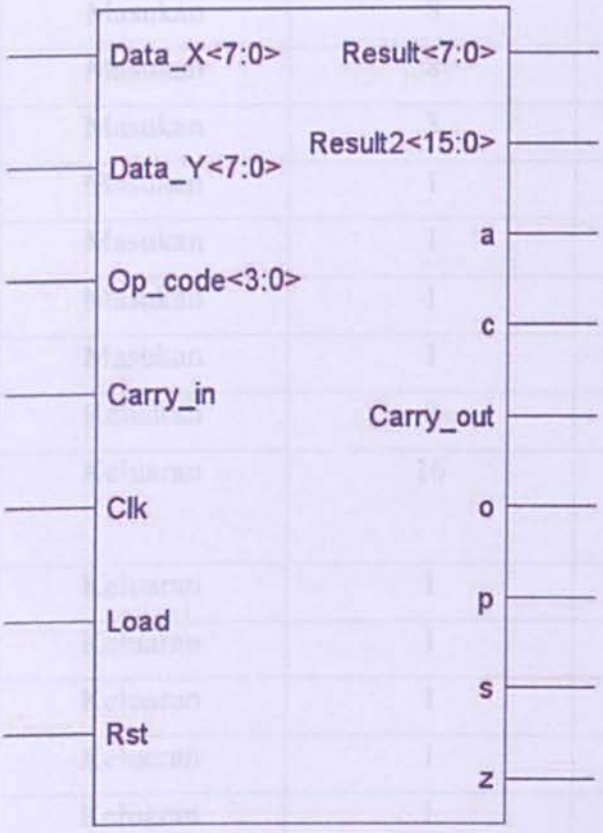
ALU adalah litar bersepadu yang mempunyai set pengoperasian arimetik dan logik. Ianya biasa digabungkan didalam mikropemproses. Elemen-elemen yang terkandung dalam sesebuah komputer seperti daftar, ingatan dan unit kawalan akan berhubung dengan ALU untuk diproses dan kemudiannya hasil operasi yang dilaksanakan di ALU akan dihantar semula kepada komponen-komponen yang tertentu.

Fungsi ALU di dalam sesebuah komputer adalah sebagai satu komponen elektronik yang melaksanakan set-set arahan logik Boolean (0 dan 1) dalam bentuk digital. Rajah 2.5 menunjukkan gambarajah blok bagi senibina asas ALU. Data-data masukan yang diberikan kepada ALU akan dihantar kepada pendaftar. Manakala hasil pengiraan yang dikeluarkan oleh ALU juga akan tersimpan di dalam daftar. Daftar bertindak sebagai memori simpanan sementara yang terletak di antara ALU dan komponen yang berikutnya untuk menghubungkan kedua-dua komponen tersebut dengan memberikankan isyarat laluan bagi data tersebut. ALU juga mengadungi set bendera yang berfungsi untuk

mengeluarkan hasil bagi operasi yang dilakukan. Unit kawalan pula berfungsi untuk menyediakan isyarat bagi mengawal operasi dan pergerakan data keluar masuk melalui ALU.

Rajah 2.5: Gambarajah BlokALU 8-bit

ALU merupakan litar gabungan atau litar bersepadu yang melaksanakan pelbagai operasi Boolean dan aritmetik. Rajah 2.6 menunjukkan gambarajah kotak hitam bagi sebuah **ALU 8-bit**. Ianya terdiri daripada tujuh masukan dan juga dua keluaran. Masukan tersebut adalah terdiri daripada masukan data input iaitu pin A dan B. selain itu, pin bawaan masuk iaitu *Carry in* adalah yang menerima data daripada operasi sebelumnya. Seterusnya pula input bagi kod operan dimana kod ini akan menentukan jenis operasi yang akan dilakukan oleh ALU ini. Bagi keluaran pula terdapat keluaran keputusan atau *result*. Satu lagi keluaran yang terdapat ialah keluaran bawaan keluar dimana ia akan menyimpan nilai bawaan kehadapan bagi data yang telah dioperasikan oleh **ALU 8-bit** sebelum tadi dimana ia akan membawa output nilai tertinggi daripada operasi **ALU 8-bit** tersebut. Keluaran berikut pula adalah bendera diaman ia akan menberikan nilai yang akan menunjukkan status keadaan akumulator selepas operasi.



Rajah 2.6: Gambarajah Kotak Hitam ALU 8-bit

Sebuah **ALU** adalah komponen teras bagi semua unit pemprosesan pusat **(CPU**- central processing units**)**. Kemampuan yang paling utama yang boleh dilakukan oleh **ALU 8-bit** ini adalah operasi penambahan (addition), penolakan (subtraction) dan pendaraban (multiplication) asas serta operasi logik berasaskan bit seperti DAN ATAU dan Ekskulif ATAU. Secara amnya sesebuah ALU yang biasa tidak menjalankan operasi pembahagian integer ataupun operasi nombor perpuluhan. Untuk melaksanakan operasi tersebut ia memerlukan komponen berasingan yang melakukan operasi tersebut seperti pembahagi atau unit nombor perpuluhan (floating point unit -FPU). Selain komponen tambahan, penggunaan program kodmikro juga dapat melaksanakan operasi tersebut.

Nama	Arah	Bit	Penerangan
Data_X	Masukan	8	Operan A
Data_Y	Masukan	8	Operan B
Op_code	Masukan	3	Opkod
Carry_in	Masukan	1	Nilai sebelum
Clk	Masukan	1	Kegunaan shifter
Load	Masukan	1	Kegunaan shifter
Rst	Masukan	1	Kegunaan shifter
Result	Keluaran	8	Hasil
Result2	Keluaran	16	Hasil Pendarab sahaja
a	Keluaran	1	Bendera
c	Keluaran	1	Bendera
o	Keluaran	1	Bendera
p	Keluaran	1	Bendera
s	Keluaran	1	Bendera
z	Keluaran	1	Bendera
Carry_out	Keluaran	1	Bawaan keluaran

Jadual 2.3: Penerangan masukan dan keluaran ALU 8-bit

Secara pengoperasian sesebuah ALU akan mengambil data yang diinputkan dan kemudiannya peroseskan dilaksanakan keatas data tersebut mengikut arahan yang diberikan dalam bentuk kod *operand* diberikan oleh unit kawalan. Kod ini akan menyatakan operasi apakah yang akan dilaksanakan terhadap input data tersebut. Seterusnya hasil operasi iaitu Result, Result2, a, c, o, p, s, z dan Carry_out akan dikeluarkan dan dihantar kepada unit yang menggunakan data tersebut. Jadual 2.4 menerangkan kod operan yang digunakan dalam ALU 8-bit.

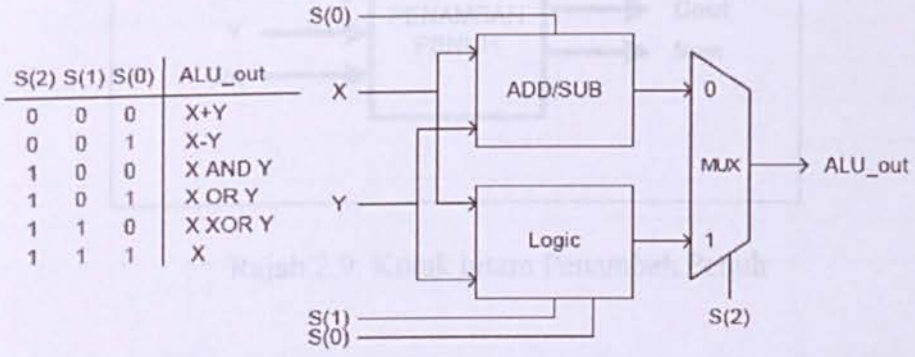
Opkod	Nama Operasi	Penerangan
0000	NOP	Tiada operasi
0001	ADDER(RCA)	Penambahan menggunakan RCA
0010	ADDER(CLA)	Penambahan menggunakan CLA
0011	SUBTRACTOR(RCA)	Pengurangan menggunakan RCA
0100	SUBTRACTOR(CLA)	Pengurangan menggunakan CLA
0101	AND	Operasi DAN
0110	OR	Operasi ATAU
0111	XOR	Operasi Eksklusif ATAU
1000	SHIFT(LEFT)	Anjak ke kiri
1001	SHIFT(RIGHT)	Anjak ke kanan
1010	ROTATE(LEFT)	Putar ke kiri
1011	ROTATE(RIGHT)	Putar ke kanan
1100	INCREMENT(A)	Penambahan 1 bagi A
1101	INCREMENT(B)	Penambahan 1 bagi B
1110	DECREMENT(A)	Pengurangan 1 bagi A
1111	DECREMENT(B)	Pengurangan 1 bagi B

Jadual 2.4: Kod Operan ALU 8-bit

2.5.1 REKA BENTUK ALU

Seperti yang telah dinyatakan ALU boleh melakukan operasi aritmetik dan logik. Daripada gambarajah dan jadual dibawah, ia boleh menerangkan mengenai operasi yang dapat dilakukna oleh ALU tersebut. Daripada gambarajah 2.7, dapat dilihat bahawa terdapat dua input data iaitu X dan Y. Selain dari itu terdapat tiga lagi input yang akan menentukan mod operasi bagi ALU ini. Input tersebut adalah S(0), S(1) dan S(2). Jika diperhatikan dalam jadual tersebut apabila data dimasukkan kedalam input X dan Y, ia memerlukan nilai dari S(0), S(1) dan S(2). Ia akan menentukan jenis operasi. Sekiranya nila masukan tersebut iaitu S(0) S(1)S2)= 000 maka operasi yang akan dilaksanakan adalah operasi penambahan. Nilai pada S(2) akan mengarahkan multiplexer untuk mengeluarkan nilai dari cip add/sub. Secara amnya rekabentukk ALU terbahagi kepada

tiga peringkat dimana ia terdiri daripada unit aritmetik, logik dan gabungan kedua-duanya menggunakan pemultipleks.

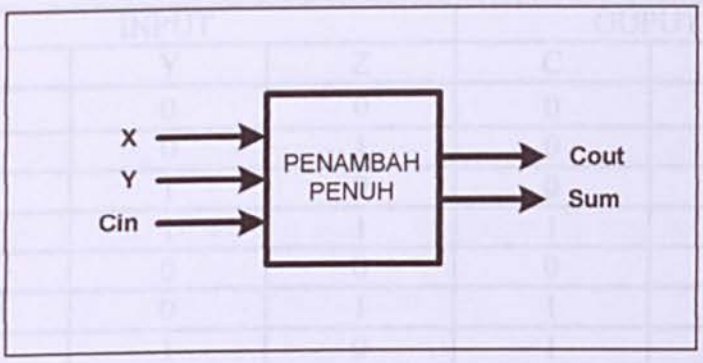


Rajah 2.7: Gambarajah Blok ALU

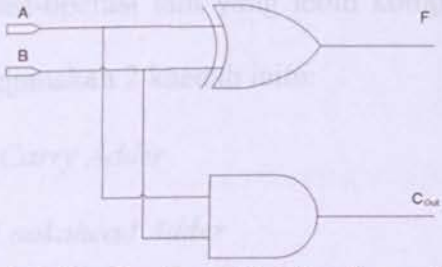
5+3 = 8				
CARRY	1	1	1	
A	0	1	0	1
B	0	0	1	1
RESULT	(0)1	(1)0	(1)0	(1)0

Rajah 2.8: Contoh operasi ALU

2.5.2 PENAMBAH / ADDER

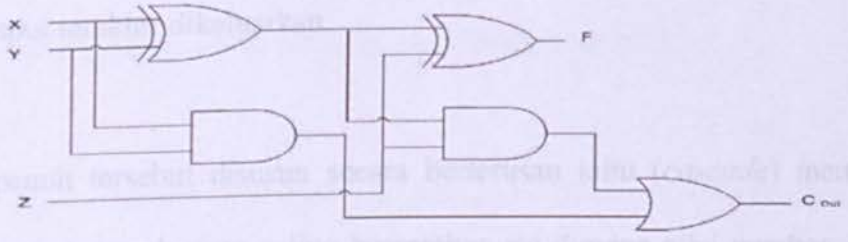


Rajah 2.9: Kotak hitam Penambah Penuh



Rajah 2.10: Gambarajah Litar Penambah Separuh

Litar penamabah merupakan asas bagi pembinaan bagi pembinaan litar arimetik yang akan melaksanakan operasi aritmetik keatas nombor binari atau nombor desimal dalam bentuk binari. Litar ini mampu melaksanakan operasi penambahan. Hasil yang diperolehi daripada daripada jadual kebenaran adalah manipulasi terhadap dua penambahan separuh.



Rajah 2.11: Gambarajah Litar Penambah Penuh

INPUT			OUPUT	
X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

Jadual 2.5: Jadual Kebenaran Penambah Penuh

Untuk melakukan operasi-operasi lain yang lebih kompleks, litar penambah penuh ini akan digabungkan menggunakan 2 kaedah iaitu:

- *Ripple Carry Adder*
- *Carry Lookahead Adder*

2.5.3 PENAMBAH / ADDER

Ripple Carry Adder merupakan penambah selari bagi sebuah litar digital. Ia kan melaksanakan operasi aritmetik dua atau lebih nombor binari menggunakan babungan beberapa penambah penuh. Pengoperasiannya adalah bergantung kepada nilai bit terbesar sebelumnya atau Carry in dan hasil ini akan melalui jujukan litar penambah penuh, sehingga output terakhir dikeluarkan.

Penambah penuh tersebut disusun secara berterusan iaitu (*cascade*) membentuk satu barisan penambah penuh yang saling bersambungan dimana nilai pembawa (Carry out) terdahulu dimasukkan kepada penambah penuh sterusnya sehingga operasi selesai.

Bilangan penambah penuh yang diperlukan adalah bergantung kepada bilangan bit yang akan dioperasikan oleh penambah selari tersebut. Sekiranya sejumlah n bit dioperasikan, maka sejumlah n bit jugalah penambah penuh diperlukan. Sebagai contoh, penambahan nombor 4-bit: 1111 + 0001 seperti ditujukkan dibawah:

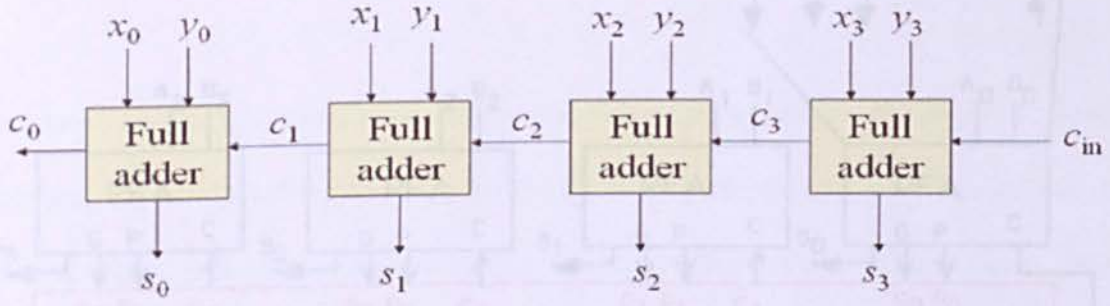
CARRY	1	1	1	1	
		1	1	1	1
	+	0	0	0	1
	1	0	0	0	0

Rajah 2.12: Penambahan nombor 4-bit untuk menerangkan konsep Ripple Carry Adder

Dalam kes ini, penambahan ($1+1 = 10_2$) pada bit terkanan menyebakan bawaan (*Carry*) mempunyai nilai 1. Bit seterusnya akan menghasilkan bawaan (*Carry*) yang benilai satu juga sehingga nilai bawaan akhir (*Carry*) dikeluarkan. Keadaan ini memerlukan isyarat melalui atau bergerak ke semua penambah. Ini akan mengakibatkan lengahan (*delay*). Secara keseluruhannya, nilai jumlah hasil dan nilai bawaan akan hanya dapat diperolehi selepas 2(N-1) + 1 lengahan get dimana N ialah bilangan bit. Jumlah nilai bawaan teakhir pula akan diperolehi selepas 2N lengahan get. Lengahan juga akan bertambah disebabkan oleh lengahan akibat sambungan antara get tersebut.

Kelemahan dalam Ripple Carry Adder ini adalah ia akan menjadi perlahan sekiranya ia memerlukan penambahan bit yang banyak. Contohnya untuk pengiraan 32-bit, lengahan yang akan terhasil adalah 63 ns (2(32-1)+1= 63). Secara keseluruhannya Ripple Carry

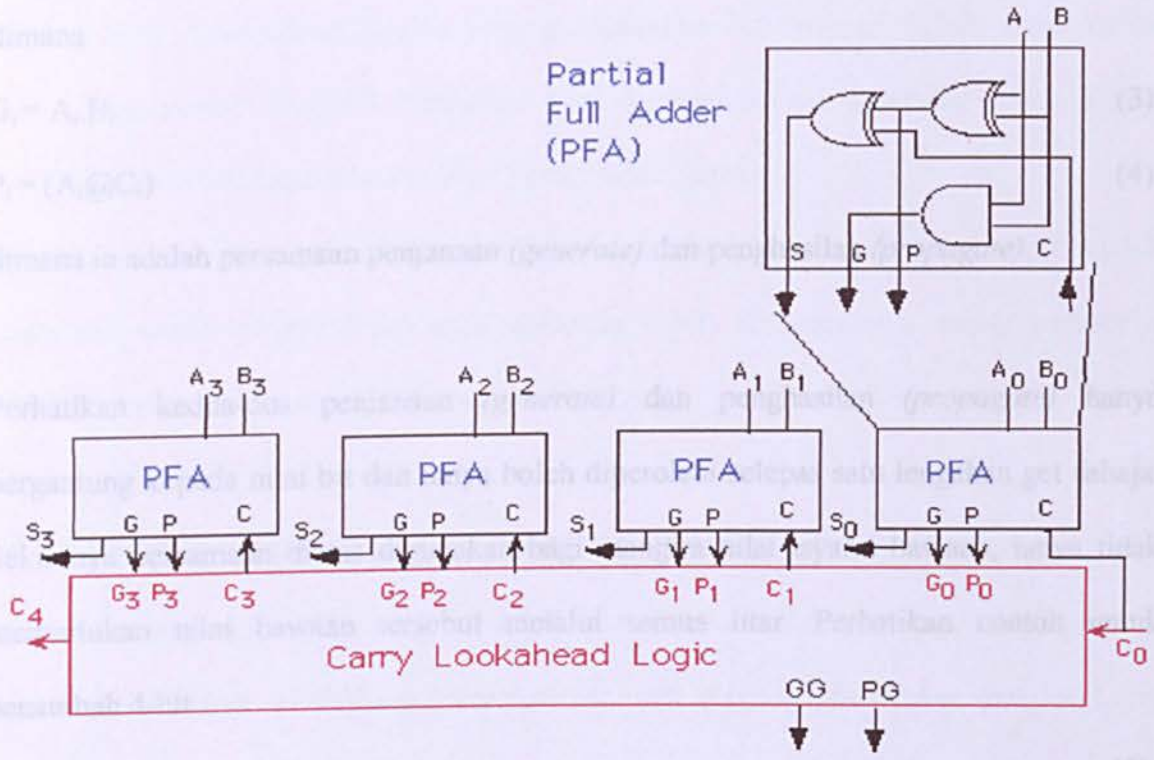
Adder dianggap perlahan kerana penghasilan lengahan semasa pengoperasian. Ia juga menghadkan kadar frekuensi *data stream* yang diperoses, walaupun implimentasinya adalah mudah daripada jenis penambah lain yang akan dibincangkan seterusnya.



Rajah 2.13: Rekabentuk Ripple Carry Adder

2.5.4 CARRY LOOKAHEAD ADDER

Carry Lookahead Adder(CLA) adalah teknik yang dapat menyelesaikan masalah lengahan dalam operasi aritmetik. Objektif utamanya adalah untuk melakukan operasi yang lebih pantas. *Carry Lookahead Adder(CLA)* menggunakan *partial full adder* bagi melaksanakan operasi penambahan. Dengan menggunakanya masa pemprosesan kan menjadi lebih cepat tanpa nilai lengahan yang besar. Namun begitu litar yang digunakan akan menjadi lebih kompleks. Disamping itu, *Carry Lookahead Adder(CLA)* hanya dapat melakukan penambahan 4-bit sahaja. Bagi penambahan melebihi 4-bit, beberapa set 4-bit *Carry Lookahead Adder(CLA)* perlu digunakan.



Rajah 2.14: Rekabentuk 4-bit *Carry Lookahead Adder(CLA)*

Seperti yang kita tahu *Carry Lookahead Adder(CLA)* mengatasi masalah lengahan dengan mengira nilai isyarat bawaan kehadapan terlebih dahulu berdasarkan isyarat input. Ianya berdasarkan keatas dua andaian iaitu

1. apabila kedua-dua bit A_i dan B_i adalah bernilai 1, atau;
2. nilai bit salah satu input adalah satu dan nilai bawaan masuk(litar sebelumnya) bernilai 1.

Secara persamaan boleh kita tuliskan sebagai,

$$C_{out} = C_{i+1} = A_i.Bi+(A_i@B_i).C_i \quad (1)$$

Simbol @ merupakan symbol bagi operasi Eksklusif Atau. Seterusnya persamaan boleh ditulis sebagai

$$C_{i+1} = G_i+P_i.C_i \quad (2)$$

dimana

$G_i = A_i.B_i$ (3)

$P_i = (A_i@C_i)$ (4)

dimana ia adalah persamaan penjanaan *(generate)* dan penghasilan *(propagate)*.

Perhatikan kedua-dua penjanaan *(generate)* dan penghasilan *(propagate)* hanya bergantung kepada nilai bit dan ianya boleh diperolehi selepas satu lengahan get sahaja. Sekiranya persamaan diatas digunakan bagi mengira nilai isyarat bawaan, ianya tidak memerlukan nilai bawaan tersebut melalui semua litar. Perhatikan contoh untuk penambah 4-bit

$C_1 = G_0 + P_0.C_0$ (5)

$C_2 = G_1 + P_1.C_1 = G_1 + P_1.G_0 + P_1.P_0.C_0$ (6)

$C_3 = G_2 + P_2.G_1 + P_2.P_1.G_0 + P_2.P_1P_0.C_0$ (7)

$C_4 = G_3 + P_3.G_2 + P_3.P_2.G_1 + P_3.P_2.P_1.G_0 + P_3.P_2.P_1.P_0.G_0$ (8)

Perhatikan nilai bit bawaan keluar, Ci+1 pada niali keempat boleh didapati selepas tiga lengahan. Nilai keluaran jumlah pula boleh dikira boleh diperolehi dengan persamaan berikut.

$S_i = A_i@B_i@C_i = P_i@C_i$ (9)

Nilai jumlah boleh didapati selepas empat lengahan get. *Carry Lookahead Adder (CLA)* oleh dibahagikan kepada dua modul iaitu.

1. *Partial Full Adder (PFA)* yang menjana S_i, P_i dan G_i seperti yang ditunjukkan oleh persamaan 3,4 dan 9.

2. *Carry Look-ahead Logic* yang menjanakan bit bawaan keluar berdasarkan persamaan 5 hingga 8. Penambah 4-bit boleh dibina menggunakan 4 *Partial Full Adder (PFA)* dan sebuah *Carry Look-ahead Logic.*

Kelemahan *Carry Lookahead Adder (CLA)* ini adalah *Carry Look-ahead Logic* adalah kompleks untuk penambahan lebih daripada 4-bit. Kebiasaannya untuk pengiraan melebihi 4-bit ia kan menggunakan modul berhirarki.

2.5.5 RIPPLE CARRY ADDER VS CARRY LOOKAHEAD ADDER

Secara keseluruhannya Ripple Carry Adder dan Carry Lookahead Adder melaksanakan fungsi penambahan. Walaubagaimanapun, terdapat beberapa perbezaan yang terdapat diantara kedua-duanya

RCA	CLA
Operasi penambahan bergantung kepada hasil operasi sebelumnya	Operasi penambahan tidak bergantung kepada hasil sebelumnya
Implementasi yang mudah	Implementasi lebih kompleks. Menggunakan partial Full Adder
Masa pemprosesan bergantung kepada jumlah penambah penuh akibat lengahan yang terhasil	Tiada lengahan tetapi hanya boleh beroperasi secara optima dalam bentuk 4-bit

Jadual 2.6: RCA Vs CLA

2.6 BENDERA

Bendera didalam ALU merujuk kepada sebuah daftar bersaiz 16 bit. Ia berfungsi sebagai penunjuk kepada status akumulator sesebuah ALU setelah operasi dilaksanakan.

Daripada 16-bit tersebut, hanya 6-bit sahaja yang digunakan untuk menunjukkan keadaan ALU.

Bendera-bendera yang terdapat didalam ALU adalah:

1. bendera sifar (zero flag)
2. bendera tanda (sign flag)
3. bendera pariti (parity flag)
4. bendera bawa (carry flag)
5. bendera bawa bantu (auxiliary flag)
6. bendera limpahan (overflow flag)

Bendera Sifar akan disetkan ke nilai 1 apabila semua bit hasil adalah o. ia digunakan bagi bagi menentukan sama ada nilai hasil adalah sifar atau tidak selepas operasi aritmetik.

00000000 ZF = 1

01110101 ZF = 0

Bagi **bendera tanda**, apabila operasi aritmetik selesai dilaksanakan, status bit tanda akan di umpukkan kepadanya iaitu nilai kosong sekiranya ia bernilai positif dan satu bagi nilai negatif.

10001001 SF = 1

01001000 SF = 0

Bendera pariti pula merujuk kepada jumlah bit satu pada nilai hasil operasi yang dikeluarkan. Sekiranya nilai bit satu pada hasi adalah genap maka bendera akan disetkan kepada niali satu. Sekiranya nilai bit satu hasil adalah ganjil, maka nilai bendera akan disetkan kepadan nilai kosong. Contoh berikut menerangkan penggunaan bendera pariti.

Nombor	Bilangan 1	Bendera Pariti
00001001	2	PF = 1
00111000	3	PF = 0

Bendera bawa pula, merujuk kepada nilai bit paling kiri hasil operasi. Sekiranya ia bernilai 1, maka bendera akan diset kepada nilai satu, sekiranya nilai bawaan adalah kosong, maka nilai bendera akan diset pada nilai kosong.

AX = 11111111

BX = 00000001

100000000 CF = 1

Bendera bawa bantu pula merujuk kepada nilai bawaan bit rendah kepada bit tinggi. Dengan maksud lain , jika berlaku nilai bawaan pada bit ketiga kepada bit keempat semasa operasi aritmetik, maka nilai bendera bawa Bantu akan di setkan kepad nilai satu dan sebalik nya sekiranya niali bawaan adalah kosong.

CARRY	1	1	1	1	1	0	0	
AX		1	0	1	1	0	1	1	0
BX		0	1	0	1	1	1	0	1
1		0	0	1	1	0	0	1	1

AF =1

Akhir sekali adalah **bendera limpahan,** ianya akan diset kepada hasil operasi yang dilaksanakan melebihi julat yang dibenar.contohnya dalam penomboran lapan bit bertanda mempunyai niali tertinggi 127 dan terendah –128. sekiranya nilai hasil melebini nilai tersebut, maka niali bendera limpah akan di set kepada nilai 1. contoh berikut akan menerangkannya dengan lebih jelas.

CARRY	0	1	1	1	1	1	1	
AX	1	0	1	1	0	1	1	1
BX	1	0	0	1	1	1	0	1
	10	1	0	1	0	1	0	0

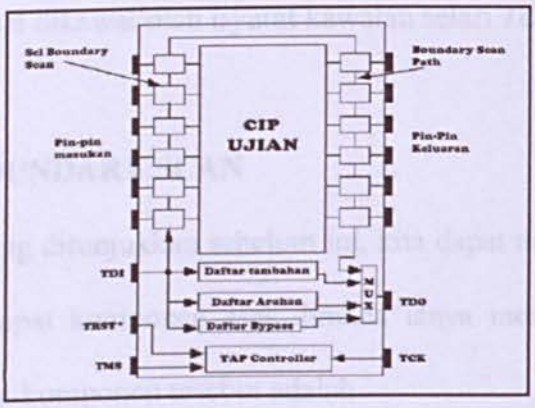
OF = 1

2.7 BOUNDARY SCAN (PENGENALAN)

Metodologi pengujian *boundary scan* adalah kaedah yang digunakan untuk menguji hubungan saling sambungan pada papan litar yang dilaksanakan pada peringkat litar bersepadu (IC). Ketidakupayaan untuk menguji papan litar yang terlalu kompleks menggunakan penguji litar tradisional dan membetulkan *bed of nail* menjadi masalah pada awal tahun 80-an. Merujuk kepada kekangan ruang fizikal dan kehilangan pada capaian fizikal untuk membaiki komponen, kos untuk membaikinya telah meningkat secara dramatik sementara keupayaan untuk membaikinya juga telah menurun pada masa yang sama.

Pengujian *boundary scan* juga boleh menggabungkan beberapa IC secara bersama pada papan PC supaya ianya dapat diuji dengan hanya menggunakan beberapa pin sahaja pada penyambung pinggir papan PC. Malah pengujian menggunakan *boundary scan* dapat mengurangakan kos pembangunan dan penghasilan. Disebabkan kos yang murah dan kebolehan capaian *boundary scan* pada peringkat IC, kegunaannya adalah melebihi kemampuan aplikasi litar ujian tradisional pada rekabentuk produk dan perkhidmatan. Kelebihan yang dikenalpasti daripada *boundary scan* adalah:

- Menjimatkan masa pengujian
- Meliputi julat ujian yang tinggi
- Meningkatkan keupayaan diagnostik
- Kos peralatan kapital yang rendah



Rajah 2.15: Rekabentuk Asas Boundary Scan

Daripada rajah 2.15, kita dapat melihat sepintas lalu bagai mana *boundary scan* melaksanakan fungsinya sebagai penguji litar digital. Dapat diperhatikan bahawa Sel

boundary scan disetkan kepada daftar anjak masukan selari dan keluaran selari. Operasi masukan selari, juga dikenali sebagai operasi *capture*, menyebabkan nilai isyarat pada pin masukan unit yang akan diumpukkankan kedalam sel masukan dan nilai isyarat akan melepasi daripada teras logik kepada pin keluaran unit yang akan diumpukkankan kepada sel keluaran.

Operasi *unload* selari yang dikenali sebagai operasi kemaskini (*update),* menyebabkan nilai isyarat yang sedia ada di dalam sel pengesan keluaran akan dihantar melepasi pin keluaran unit. Nilai isyarat sedia ada di dalam masukan sel pengesan akan dihantar ke dalam teras logik.

Data juga boleh dianjakkan kedalam daftar anjakan secara sesiri iaitu dengan dimulakan daripada pin masukan bernama *Test Data In* (TDI) dan diakhiri pada pin keluaran bernama *Test Data Out* (TDO). Pemasa ujian (TCK) dimasukkan melalui pin masukan yang lain dan mod operasi dikawal oleh isyarat kawalan selari *Test Mode Select* (TMS).

2.7.1 KOMPONEN BOUNDARY SCAN

Daripada gambarajah yang ditunjukkan sebelum ini, kita dapat melihat bahawa *boundary scan* ini mempunyai empat komponen asas dimana ianya mestilah ada dalam sistem pengujian ini. Komponen-komponen tesebut adalah

- Test Access Port (TAPs)
- TAP Controller
- Daftar Arahan (Instruction Register)
- Daftar Data Ujian (Test Data Register)

Kesemua komponen ini tidak akan dibincangkan dengan lebih lanjut didalam bab ini kerana ia akan disentuh didalam bab 5 iaitu rekabentuk sistem.

2.8 KESIMPULAN

Didalam bab ini, topik utama yang dibincangkan dalah mengenai kepentingan ujian dan jeniss ralat dan ujian yang akan dilakukan. Selain itu, perbincangan mengenai ALU, penambah, Ripple Carry Adder, Carry Lookahead Adder, bendera dan Boundary scan telah dibentangkan didalam bab ini. Semua ini akan menjadi panduan dalam mereka bentuk dan membangunkan sistem ini kelak.

BAB 3
METODOLOGI

BAB 3

METODOLOGI

3.0 PENGENALAN

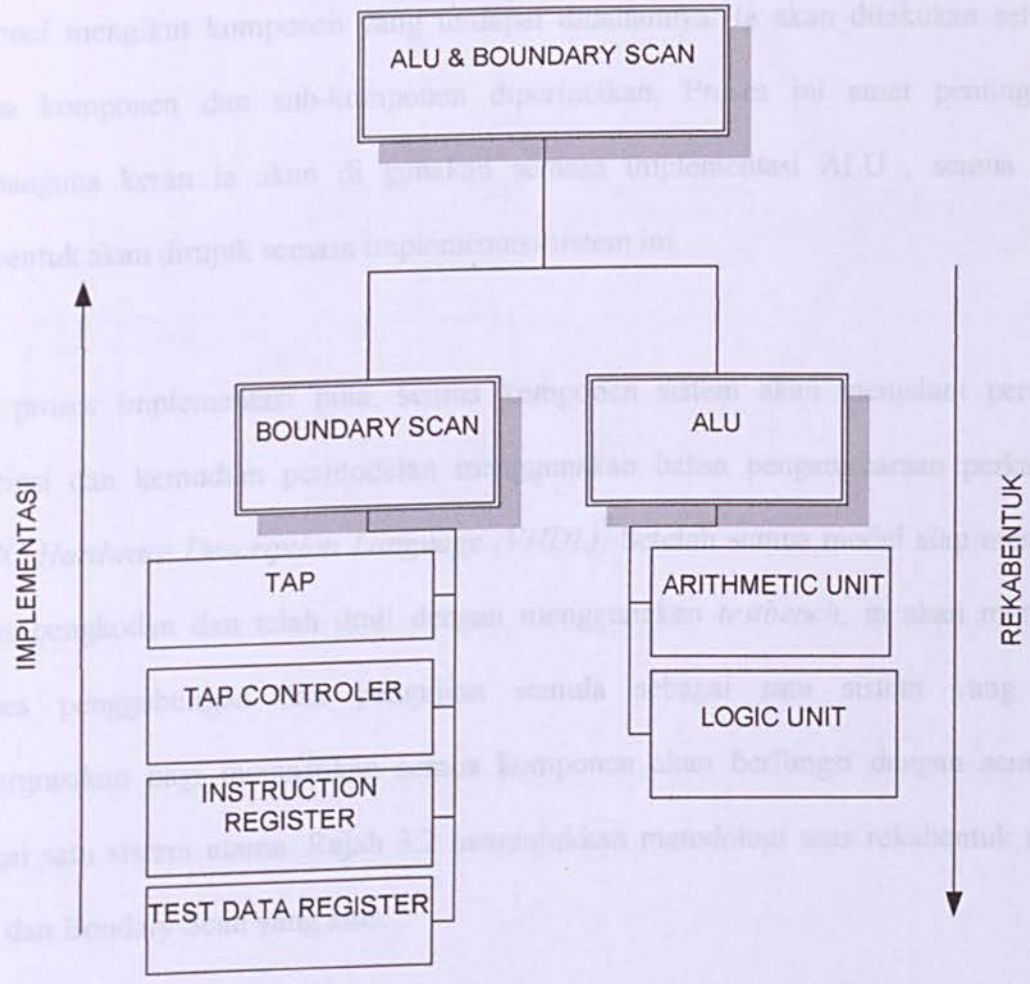
Projek ini mempunyai had masa yang agak singkat, maka pembangunan projek perlu dilaksanakan dengan teratur. Pembangunan harus di laksanakan dengan mengambil masa yang minima dan akan menghasilkan keputusan yang optimum. Bagi memastikan projek dibangunkan berlandaskan spesifikasi yang telah ditetapkan kawalan perstasi juga harus dilaksanakan dari semasa ke semasa.

Bagi projek ini, dua komponen besar yang harus diberi perhatian adalah ALU dan juga Boundary Scan. Pembangunan dua komponen ini haruslah berjalan mengikut perancangan, siap mengikut jadual, memenuhi segala spesifikasi yang telah ditetapkan. Disamp[ing itu pendekatan pembangunan dan juga implementasi adalah dua perkara yang saling bersandaran antara satu sama lain bagi memastikan pembangunan akan berjalan dengan lancar dan lebih sistematik.

3.1 PENDEKATAN PEMBANGUNAN

Pendekatan yang akan digunakan membangunkan sistem ini adalah mengunakan dua pendekatan iaitu pendekatan dari atas ke bawah bagi proses rekabentuk, manakala bagi

permodulan pula, pendekatan dari bawah keatas akan dilakukan bagi menyiapkan projek tersebut sekaligus untuk tujuan simulasi.



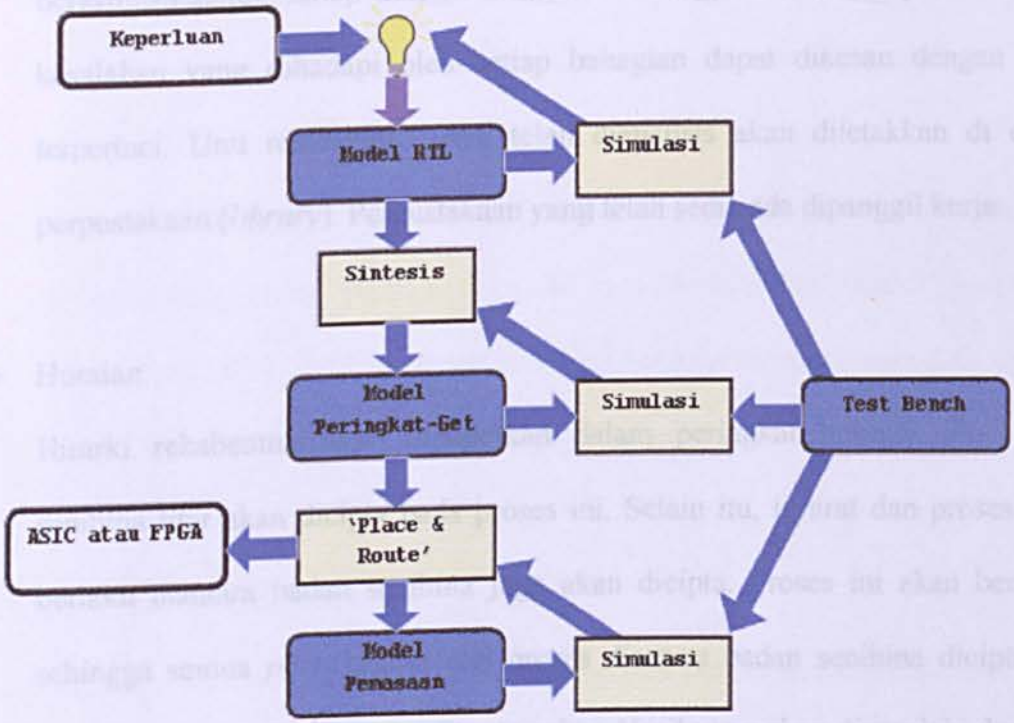
Rajah 3.1: Pendekatan pembagunan Mekanisma Pengujian ALU Menggunakan Teknik Boundary Scan

Rajah 3.1 menunjukkan pendekatan yang akan digunakan dalam membangunkan mekanisma pengujian ALU menggunakan teknik boundary scan ini. Seperti yang telah dinyatakan sebelum ini.dua pendekatan yang akan digunakan dalm membangunkan projek ini. Pada awalnya, rajah kotak hitam sesebuah ALU akan dihasilkan. Ia

menyatakan berapa banyak masukan dan keluaran yang akan terlibat dalam membangunkan ALU tersebut seperti C_{in}, Flag, dan S0.

Selepas semua item telah dikenalpasti dan ditentukan, ALU ini akan dipecah serta diperinci mengikut komponen yang terdapat didalamnya. Ia akan dilakukan sehingga semua komponen dan sub-komponen diperincikan. Proses ini amat penting bagi pembanguna keran ia akan di gunakan semasa implementasi ALU , semua aspek rekabentuk akan dirujuk semasa implementasi sistem ini.

Bagi proses implementasi pula, semua komponen sistem akan menjalani peringkat deskripsi dan kemudian permodelan menggunakan bahsa pengaturcaraan perkakasan *VHSIC Hardware Description Language (VHDL)*. Setelah semua model siap menjalani proses pengkodan dan telah diuji dengan menggunakan *testbench*, ia akan menjalani peruses penggabungan dan pengujian semula sebagai satu sistem yang telah diintergrasikan bagi memastikan semua komponen akan berfungsi dengan sempurna sebagai satu sistem utama. Rajah 3.2 menunjukkan metodologi asas rekabentuk modul ALU dan Boudary Scan yang asas.



Rajah 3.2: Metodologi Rekabentuk Asas Pembangunan ALU dan Boundary Scan

Dalam proses perlaksanaan metodologi tersebut ,. 4 proses utama yang dilaksanakan dalam proses perlaksanaan metodologi adalah seperti berikut:

- Analisis

  Proses analisis adalah proses yang melibatkan pemeriksaan terhadap kesalahan sintaks dan kesalahan semantik. Kesalahan sintaks boleh berlaku apabila tatabahasa bagi sesuatu pengaturcaraan adalah tidak menepati kriteria yang telah ditetapkan. Manakala kesalahan semantik pula boleh berlaku apabila sesuatu permodelan itu tidak diwakilkan dengan persamaan yang betul. Dalam proses analisis ini, setiap unit rekabentuk akan diuji secara berasingan seperti pengisytiharan entiti dan badan senibina (*architecture body*). Analisis secara

berasingan untuk setiap unit rekabentuk adalah suatu cara yang baik kerana kesalahan yang dihadapi oleh setiap bahagian dapat dikesan dengan lebih terperinci. Unit rekabentuk yang telah dianalisis akan diletakkan di dalam perpustakaan (*library*). Perpustakaan yang telah sedia ada dipanggil kerja.

- Huraian

  Hirarki rekabentuk akan dipaparkan dalam peringkat huraian. *Port* pada senibina litar akan dicipta pada proses ini. Selain itu, isyarat dan proses yang berlaku diantara badan senibina juga akan dicipta. Proses ini akan berulang sehingga semua *port*, isyarat dan proses diantara badan senibina dicipta dan memenuhi kehendak senibina litar tersebut. Hasil yang akan diperolehi daripada peringkat huraian ini adalah satu himpunan isyarat rangkaian dan pemprosessan.

- Simulasi

  Pada peringkat ini, simulasi dilakukan terhadap proses-proses yang terdapat di dalam model huraian. Proses-proses tersebut adalah sensitif terhadap acara (*events)* pada isyarat masukan. Ia hanya ditetapkan pada kenyataan tunggu. Ia akan meneruskan dan menjadualkan nilai baru pada isyarat keluaran. Pada peringkat ini, penjadualan perlaksanan berlaku. Acara pada isyarat sekiranya nilai baru adalah berbeza dengan nilai yang lama. Pada permulaan fasa, setiap isyarat diberikan nilai awalan. Masa simulasi disetkan kepada '0'. Untuk setiap proses yang aktif, perlaksanaan yang aktif akan berlangsung sehingga menerima

isyarat tunggu dimana ianya akan diberhentikan. Perlaksanaan biasanya melibatkan penjadualan transaksi pada isyarat untuk masa selanjutnya.

- Sintesis

  Sintesis adalah proses yang menterjemahkan rekabentuk *register-transfer-level* (RTL) kepada *netlist* peringkat get. Ia juga menghadkan bentuk pengkodan untuk model RTL. Selain itu sintesis ini juga adalah bebas peralatan dimana sintesis boleh dilakukan dengan sebarang peralatan.

3.2 KAEDAH IMPLEMETASI

Untuk melaksanakan projek ini, aplikasi yang akan digunakan semasa implementasi ialah *VHSIC Hardware Description Language (VHDL).* Ia akan digunakan bagi memodelkan deskripsi pelakuan ALU. Setelah itu, sebuah prototaip ALU akan diperolehi dengan menterjemahkannya kepad litar get logik.

3.2.1 VHDL

Very High Speed Integrated Circuit (VHSIC) Hardware Description Language (VHDL) merupakan bahasa yang digunakan dalam menghasilkan deskripsi sistem elektronik digital. Mula digunakan di Amerika Syarikat pada tahun 1980 berikutan perkembangan yang pesat dalam pembinaan litar bersepadu serta kehendak semasa yang mahukan satu bahasa yang piawai dalam mendiskripsikan struktur dan fungsi sesebuah litar bersepadu. Pemilihan VHDL dibuat atas sebab kelebihan yang terdapat padanya. Antara kelebihan yang terdapat pada VHDL adalah:

- VHDL adalah bahasa pengaturcaraan yang memenuhi piawaian IEEE (IEEE 1076 1993) untuk litar digital.
- Membolehkan permodelan litar dari tahap get asas sehingga tahap sistem penuh.
- Menukarkan kekompleksan litar elektronik kepada rekabentuk sistem untuk sintesis litar ataupun simulasi sistem.
- Menyediakan mekanisma untuk rekabentuk digital dan penggunaan semula dokumentasi rekabentuk.
- Digunakan secara meluas dalam industri.
- Meminimumkan kos dan masa rekabentuk.
- Dapat menghasilkan rekabentuk yang lebih baik.
- Boleh digunakan untuk dengan teknologi terbaru.
- Kepelbagaian methodologi rekabentuk.
- Pengurusan rekabentuk yang lebih baik.
- Penggunaan peralatan secara bebas.

3.3 PEMBAHAGIAN TUGAS MEMBANGUNKAN MEKANISMA PENGUJIAN ALU MENGGUNAKAN TEKNIK BOUNDARY SCAN

Pembangunan projek Mekanisma Pengujian ALU Menggunakan Teknik Boundary Scan ini akan dibangunkan bersama dengan kerjasama Wan Muhammad Idzuan (WEK010389). Ia dliakukan kerana faktor skop yang luas serta jangkamasa pembangunan projek yangsingkat memnyebabkan ia terpaksa dilakukan oleh dua orang. Pembanguna projek dilakukan dengan serentak dimana ia akan dibahagikan kepada dua bahagian iaitu:

- Pembangunan ALU 8-bit
- Pembangunan Boundary Scan

Untuk pembanguna ALU 8-bit, ianya akan dilaksanakan oleh penulis. Pembangunan Boundary Scan pula akan dilaksanakan oleh Wan Muhammad Idzuan.

3.4 KESIMPULAN

Perkara yang paling penting dalam melaksanakan projek ini adalah mengenalpasti pendekatan yang akan digunakan dalam membangunkan projek ini. Kaedah implementasi yang akan digunakan adalah dengan menggunakan bahsa pengaturcaraan perkakasan iaitu VHDL. Secara keseluruhannya ia akan dibangunkan serentak iaitu pembangunan ALU dan pembangunan Boundary Scan dengan kerjasama Wan Muhammad Idzuan.

BAB 4
ANALISIS SISTEM

BAB 4

ANALISIS SISTEM

4.0 PENGENALAN

Analisa terhadap sistem merupakan proses mengenalpasti segala keperluan untuk membangunkan sistem termasuk keperluan fungsian dan bukan fungsian serta perisian dan perkakasan sistem tersebut. Selain itu analisis juga akan meningkatkan pemahaman tentang sistem yang akan dibina dimana aspek kebolehan dan kekangan yang dihadapi oleh sistem dapat diurus dengan baik.

4.1 OBJEKTIF

Tujuan analisis sistem dilaksanakan adalah:

- Mengenalpasti keperluan fungsian sistem
- Mengenalpasti keperluan bukan fungsian sistem
- Mengenalpasti keperluan perisian dan perkakasan

4.2 PROSES ANALISIS

Bagi mendapatkan maklumat yang diperlukan untuk membangunkan sistem mekanisme pengujian ALU menggunakan teknik boundary scan ini, sumber maklumat yang diperlolehi hasil dari pembacaan buku dan juga carian jurnal di internet adalah mejadi asas pembangunan sistem. Segala maklumat awal mengenai ALU dan juga Boundary Scan adalah diperolehi daripada sumber ini. Setelah maklumat mengenai diperolehi analisa mengenai fungsi yang akan dijalankan akan diketengahkan. Seterusnya keperluan fungsian dan bukan fungsian akan dikenalpasti dan

KEPERLUAN FUNGSIAN

Keperluan fungsian merupakan perkara atau syarat yang perlu dipenuhi dlam membangunkan mekanisme pengujian ALU menggunakan teknik Boundary Scan ini. Ia diperlukan oleh sistem bagi melaksanakan tugas yang telah digariskan seperti yang telah ditetapkan untuk pembangunan. Sistem pengujian yang dijalankan haruslah memenuhi beberapa criteria yang telah ditetapkan. Antara keperluan berkenaan yang diklasifikasikan sebagai keperluan fungsian adalah:

- Mempunyai set komponen asas iaitu unit logic dan aritmetik
- Mampu menerima data input 8-bit
- Mempunyai gambaran yang jeles berkaitar litar ALU tersebut

- Mampu melaksanakan operasi aritmetik menggunakan Carry Lookahead Adder (CLA).

4.3 KEPERLUAN BUKAN FUNGSIAN

Bagi keperluan bukan fungsian pula, ia merujuk kepada cirri dan syat yang tidak nerkaitan secara langsung terhadap sistem tetapi mempunyai kepentingan terhadap sistem serta pembangunannya. Antara keperluan bukan fungsian bagi ALU yang akan dibangunkan ini terdiri daripada:

- Masa pemprosesan

  Masa perlaksanaan yang singkat bagi sistem aritmetik dan logik ini memproses data bagi memastikan pengoptimuman kegunaan masa tercapai.

- Ketepatan

  ALU ini perlulah memiliki ketepatan data yang konsisten semasa melaksanakan operasi serta memiliki ralat data yang sifar untuk menjadi platform ujian kepada boundary scan.

4.4 KEPERLUAN PERISIAN

Pada masa kini, terdapat banyak perisian bagi pembangunan sistem yang berteraskan litar digital di pasaran. Antara perisian pengaturcaraan litar digital yang banyak digunakan di dalam industri adalah ialah perisian yang dibangunkan oleh Xilinx & Altera

Bahasa pengaturcaraan yang sesuai digunakan bagi merekabentuk, menganalisa, simulasi dan sintesis sistem iaitu VHDL atau *Very High Speed Integrated Circuit (VHSIC) Hardware Description Language*. Bahasa pengaturcaraan digital ini digunakan secara meluas untuk rekabentuk, pembuatan dan dokumentasi bagi litar digital dari IC kepada sistem lengkap.

Terdapat juga bahasa pengaturcaraan yang lain untuk menganalisis dan simulasi litar seperti Verilog tetapi setelah dibuat perbandingan diantara VHDL dan Verilog, didapati VHDL lebih sesuai digunakan dalam perlaksanaan projek ini kerana beberapa sebab. Antaranya adalah seperti berikut:

- Kod yang dihasilkan adalah fleksibel boleh diguna semula.
- Dapat menyokong rekabentuk yang besar.
- Penggunaan sehingga pada senibina peringkat tinggi.
- Bahasa pengaturcaraan yang didefinasikan oleh pengguna.

4.4.1 MENGENAI VHDL

VHDL menjadi piawai bagi IEEE 1076 pada tahun 1987. Menjalani pengemaskinian semula pada tahun 1993 dan kini dikenali sebagai piawai IEEE 1076 1993. Penggunaan VHDL mempunyai beberapa kelebihan dalam rekabentuk litar terutamanya litar digital. Antaranya adalah seperti berikut:

- **Piawai** – Memenuhi piawaian yang ditetapkan oleh IEEE 1076 1993. Sama seperti piawaian yang lain, taraf piawaian tersebut telah mengurangkan masalah kekeliruan dan menjadikan antaramuka diantara perkakasan syarikat dan produk

lebih mudah. Mana-mana hasil pembangunan yang menepati piawaian berupaya kekal lebih lama dan kurang risiko untuk diabaikan bergantung kepada ketidaksesuaiannya dengan yang lain.

- **Sokongan industri** – Ianya lebih berkuasa dan lebih efisisen, maka ianya mendapat sokongan daripada pihak industri terutamanya daripada industri permbuatan yang melibatkan elektronik.

- **Mudah alih** – kod yang sama boleh disimulasikan dan digunakan dalam pelbagai peralatan rekabentuk pada peringkat yang berbeza untuk proses rekabentuk. Ia mengurangkan kebergantungan oada sesuatu set perkakasan rekabentuk dimana kebolehannya adalah terhad dan tidak mempunyai daya saing pada pasaran akan datang. Piawaian juga memudahkan penukaran data rekabentuk berbanding dengan rekabentuk pangkalan data yang mempunyai hakmilik perkakasan rekabentuk.

- **Kebolehan permodelan** – VHDL dibangunkan untuk membolehkan permodelan bagi semua tahap rekabentuk, daripada kotak elektronik sehingga kepada transistor. VHDL mampu memenuhi kelakuan pembangunan dan rutin matematik yang menerangkan permodelan yang rumit.

- **Bolehguna semula** – Sesetengah rekabentuk biasa boleh diterangkan, ditentusahkan dan diubahsuai di dalam VHDL untuk kegunaan pada masa akan dating, ianya lebih menjimatkan masa dan bergantung kepada masalah.

- **Teknologi bebas** – Pengesahan pada rekabentuk yang diterangkan terus oleh VHDL ini menjadikan ia bebas teknologi. Ini membolehkan perekabentuk bebas untuk meneruskan rekabentuk tanpa perlu menunggu kepada teknologi yang akan dipilih untuk digunakan.

- **Dokumetasi** – VHDL adalah rekabentuk bahasa himpunan yang membolehkan pendokumentasian diletakkan pada suatu tempat dengan melakukan proses terbina dalam pada kod. Kombinasi komen dan kod telah menetapkan apa yang patut direkabentuk dan mengurangkan kekaburan diantara spesifikasi dan perlaksanaan.

- **Metodologi rekabentuk yang baru** – Dengan menggunakan VHDL dan sintesis, ia dapat mereka metodologi yang dapat meningkatkan produktiviti rekabentuk, memendekkan kitaran rekabentuk dan mengurangkan kos.

4.5 KEPERLUAN PERKAKASAN

Untuk menghasilkan rekabentuk yang baik, keperluan perkakasan yang minimum diperlukan untuk melaksanakan proses rekabentuk. Ini kerana, bagi sesetengah perisian, sekiranya keperluan perkakasan itu tidak memenuhi keperluan minimum yang ditetapkan, perisian berkenaan tidak akan dapat dilaksanakan.

Bagi perkakasan yang hanya menyokong keperluan minimum bagi perisian berkenaan, ianya boleh melaksanakan program itu tetapi dengan tahap kepuasan pengguna yang rendah. Spesifikasi minimun yang dicadangkan adalah:

Komponen	Penerangan
Sistem Pengoperasian	Windows 2000 atau lebih tinggi.(windows XP-dicadangkan)
Pemproses	Pentium 3 atau lebih tinggi (Pentium 4-dicadangkan)
RAM	Minimum 128 MB (256MB-dicadangkan)
Ruang simpanan (cakera keras)	3GB.
Aplikasi	Xilinx
Monitor	VGA atau SVGA.
Peranti input	Tetikus, papan kekunci.

Jadual 4.1: Keperluan perkakasan dan perisian

4.6 KEKANGAN & MASALAH

Merekabentuk mekanisme pengujian ALU menggunakan teknik boundary scan akan menimbulkan beberapa masalah yang berkaitan dengan pengujian. Antaranya adalah:

- ALU tidak dapat melakukan pengoperasian pembahagian nombor integer
- ALU tidak dapat melakukan pengoperasian *floating point*

- Boundary scan tidak dapat melakukan pengujian untuk senibina dalaman sesuatu cip dengan lebih baik berbanding teknik *Built-In Self-Test* (BIST).
- Pembinaan litar yang perlu memenuhi kriteria pengujian
- Memulakan proses rekabentuk dari awal untuk memastikan rekabentuk yang dihasilkan dapat menepati objektif pengujian.

4.7 KESIMPULAN

Analisis kepada sistem yang akan dibangunkan adalah penting bagi menjamin ia akan menepati segala keperluan dan kehendak projek. Untuk mengelakkan sebarang maslah pada mas hadapan, semua perkasa yang perlu dianalisis perlulah dijelaskan secara baik bagi memudahkan sebarang rujukan untuk tujuan mengenal pasti keupayaan sistem yang dikehendaki. Adalah menjadi satu kemestian ianya didokumentasikan secara teratur untuk tujuan panduan.

BAB 5
REKABENTUK SISTEM

BAB 5

REKABENTUK SISTEM

5.0 PENGENALAN

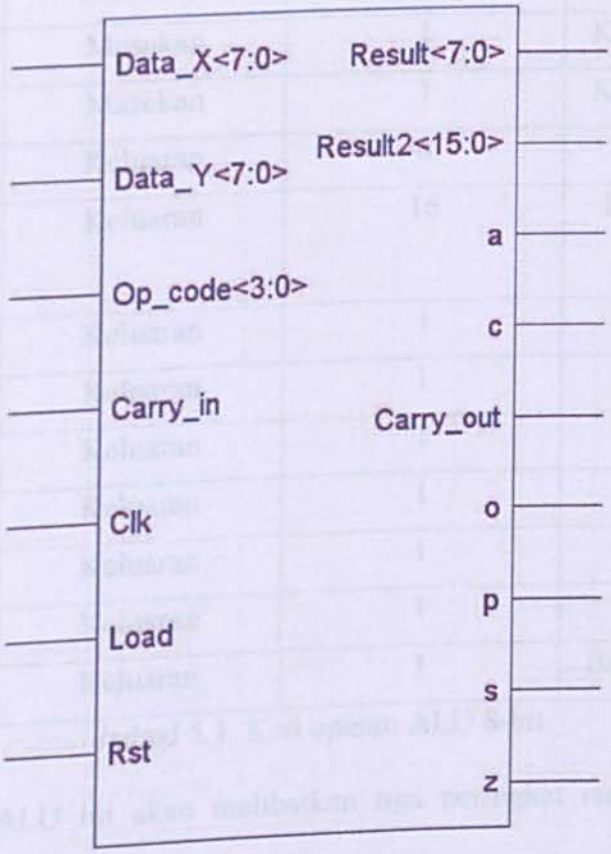
Sistem yang akan dibangunkan adalah sebuah unit aritmetik dan logik ALU yang dapat melaksanakan operasi aritmetik dan juga logik bagi satu susunan data 8-bit. Sistem yang dibangunkan akan dapat melaksanakan operasi aritmrtik dan logik yang telah dispesifikasikan. Ianya akan ditengangkan lagi kemudian. Disamping itu juga sistem pengujian terhadap ALU yang di bangunkan iaitu boundary scan haruslah mempunyai tahap keboleharapan (*reliability*) yang tinggi supaya ianya dapat menguji data ujian secara tepat bagi menghasilkan keputusan ujian yang lebih jitu.

Pembangunan sistem ini adalah hasil daripada intergrasi pembangunan 8 bit ALU dan pembangunan *boundary scan* yang memenuhi spesifikasi pengujian untuk ALU.

5.1 PEMBANGUNAN ALU 8-BIT

ALU merupakan nadi utama bagi setiap mikropemproses yangbberfungsi sebagai pusat bagi perlaksanaan semua operasi aritmetik dan logik bagi ALU tersebut. Arahan yang dilaksanakan oleh ALU adalah berbentuk digital iaitu (0, 1).

Pembangunan ALU 8-bit adalah merangkumi kepada masukan data dan juga keluaran hasil operasi. Masukan data X dan Y akan menerima 8 bit data manakala keluaran akan disalurkan kepada F.



Rajah 5.1: Gambarajah Kotak Hitam ALU 8-bit

Selain daripada masukan Data_X dan Data_Y, terdapat tiga empat masukan lain dimana masukan tersebut merupakan kod operan yang akan menentukan operasi yang akan dilakukan keatas nilai yang dimasukkan pada masukan Data_X dan Data_Y. masukan kod operan ini adalah berbentuk nombor binari 4 bit.Sila rujuk rajah 5.1.

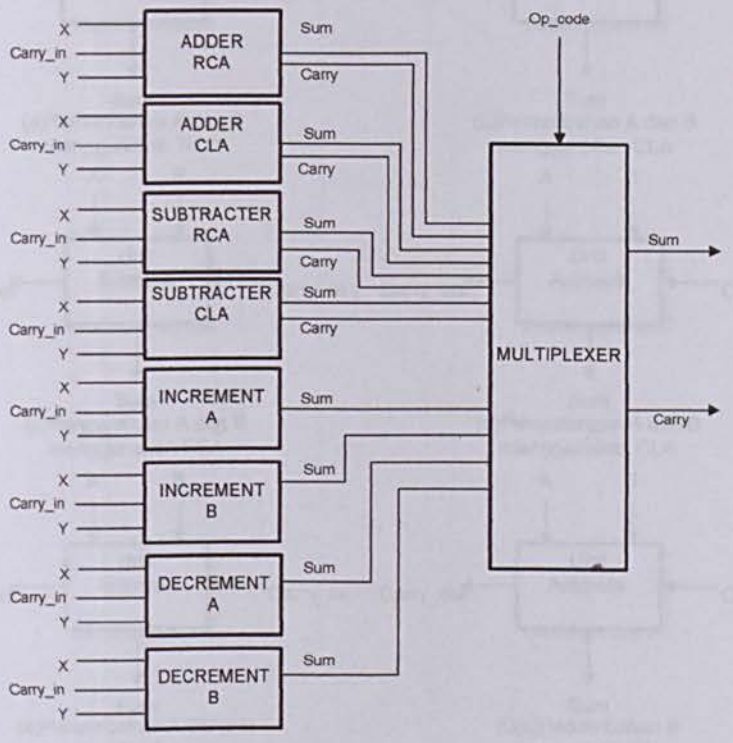
Nama	Arah	Bit	Penerangan
Data_X	Masukan	8	Operan A
Data_Y	Masukan	8	Operan B
Op_code	Masukan	3	Opkod
Carry_in	Masukan	1	Nilai sebelum
Clk	Masukan	1	Kegunaan shifter
Load	Masukan	1	Kegunaan shifter
Rst	Masukan	1	Kegunaan shifter
Result	Keluaran	8	Hasil
Result2	Keluaran	16	Hasil Pendarab sahaja
a	Keluaran	1	Bendera
c	Keluaran	1	Bendera
o	Keluaran	1	Bendera
p	Keluaran	1	Bendera
s	Keluaran	1	Bendera
z	Keluaran	1	Bendera
Carry_out	Keluaran	1	Bawaan keluaran

Jadual 5.1: Kod operan ALU 8-bit

Proses rekabentuk ALU ini akan melibatkan tiga peringkat iaitu pembangunan unit aritmetik. Seterusnya rekabentuk unit logik pula akan dilaksanakan. Peringkat akhir sekali adalah proses penggabungan kedua-dua unit tersebut untuk menghasilkan satu ALU yang lengkap didalam satu litar.

5.1.1 REKABENTUK UNIT ARITMETIK

Seperti yang telah dinyatakan didalam bab dua tardahulu, komponen asas dalam unit aritmetik adalah penambah. Unit pembinaan ini kita akan menggunakan *Carry Lookahead Adder(CLA)* dan *Ripple Carry Adder(RCA)*. Operasi yang berbeza dapat dilakukan oleh CLA dan RCA ini bergantung kepada pengawalan data. Rajah 5.2 menunjukkan rekabentuk unit aritmetik yang dimaksudkan

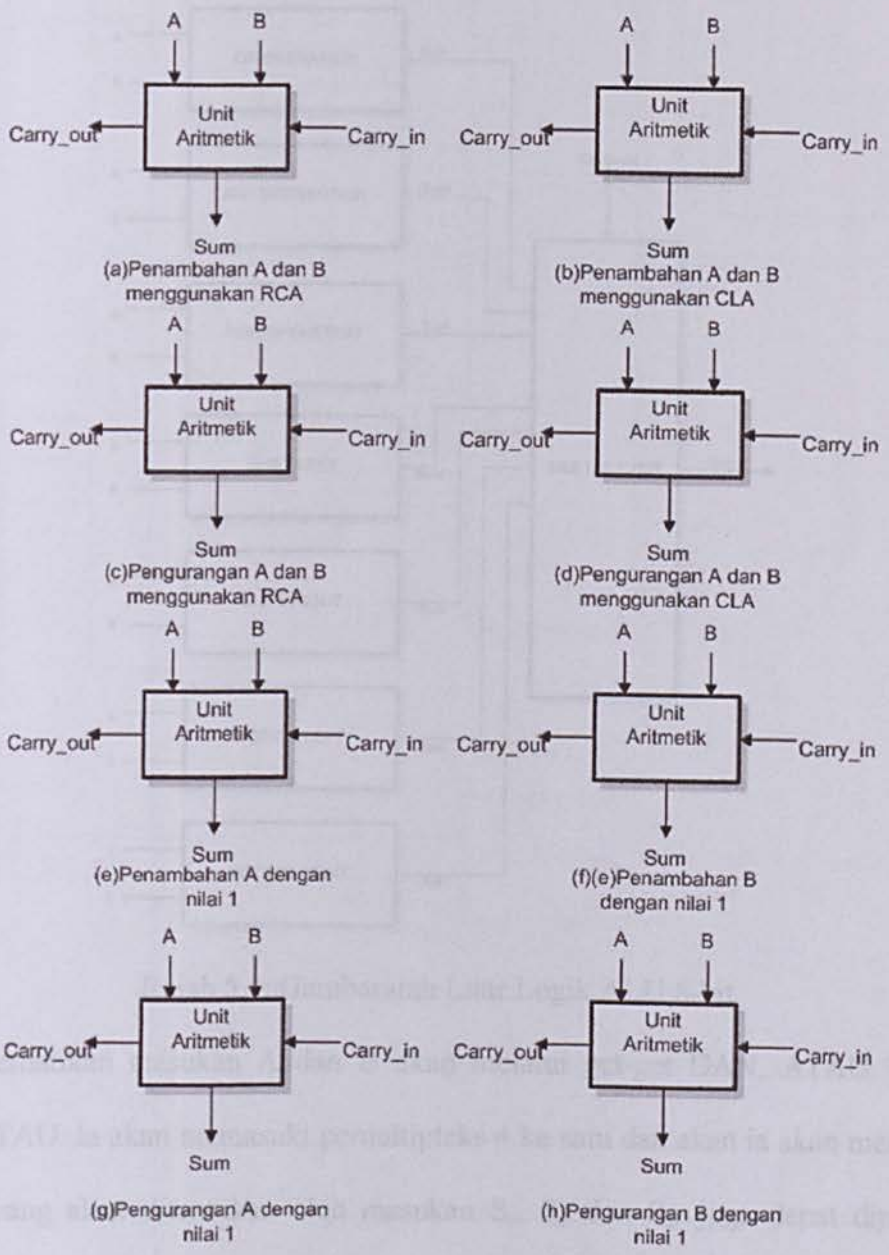


Rajah 5.2: Gambarajah Blok Litar Aritmetik

Bagi gambarajah 5.2, ia mempunyai dua masukan data iaitu A Dan B, keluaran pula akan disalurkan kepada F. Pembawa sambutan (Carry in) pula akan membawa input kelokasi nilai bit terendah (least significant bit) pada CLA. Nilai keluaran pula akan dikeluarkan oleh operasi aritmetik F = X + Y + C pada keluaran C_{out}. Semua masukan data dan juga

keluaran data adalah nombor binari 8-bit dimana tanda '+' adalah untuk mewakili operasi yang akan dilaksanakan.

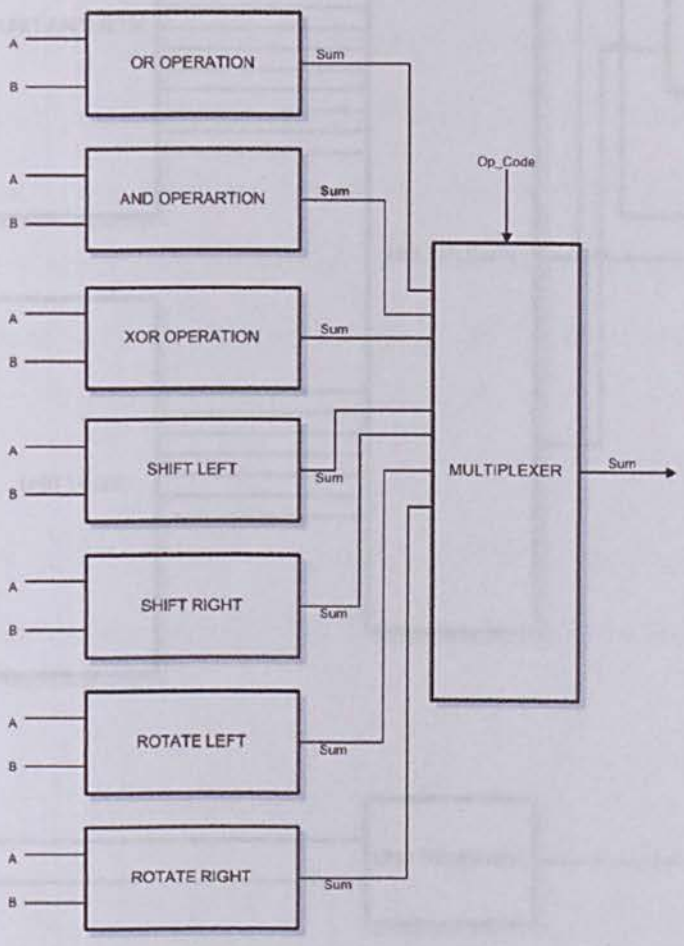
Operasi aritmetik yang dilaksanakan adalah bergantung kepada nilai masukan-masukan Op_code. Rajah 5.3 akan menerangkan operasi yang dilakukan oleh unit ini.



Rajah 5.3: Operasi-Operasi aritmetik ALU 8-bit

5.1.2 REKABENTUK UNIT LOGIK

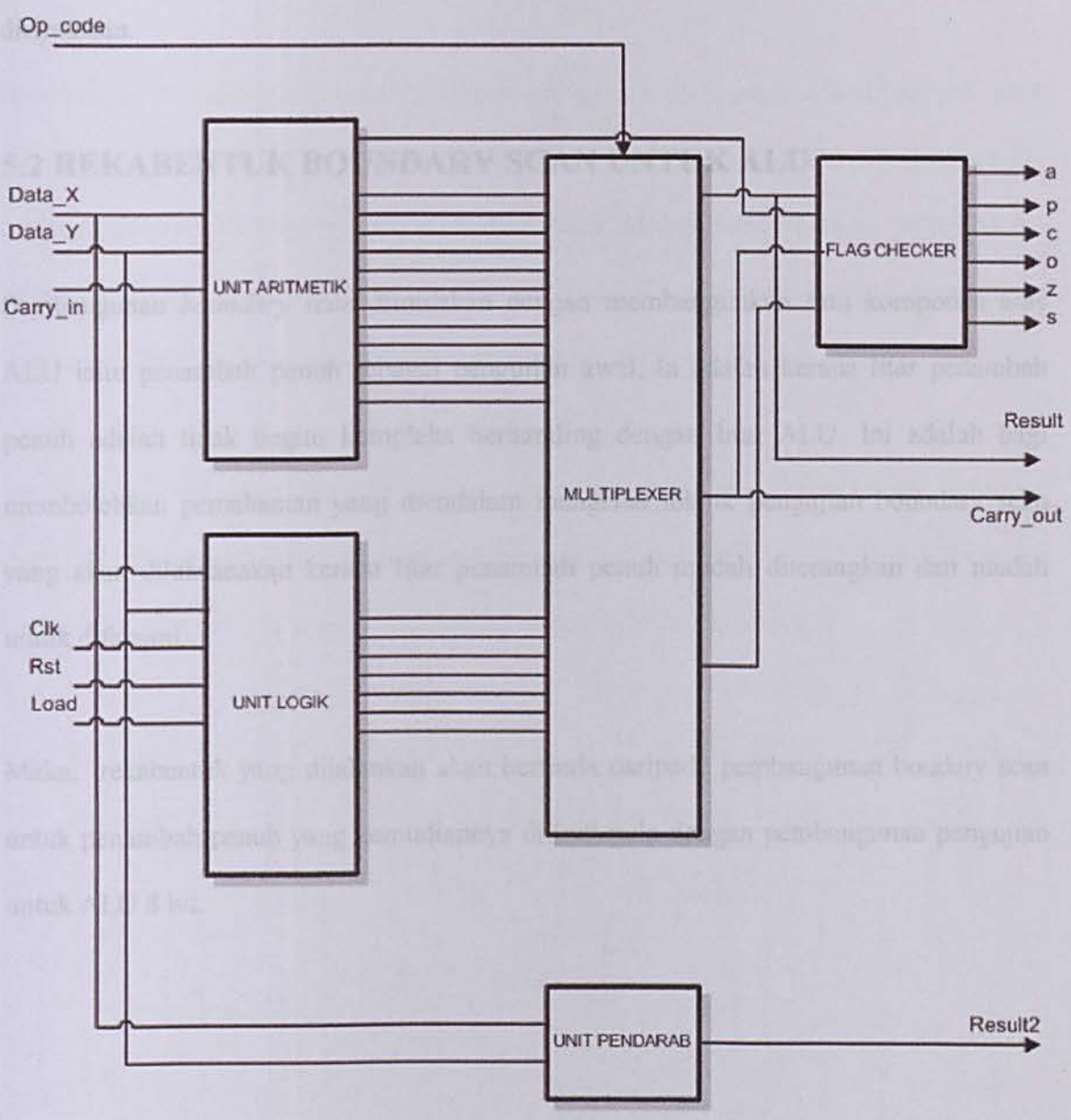
Ianya akan beroperasi apabila nilai S_3 bernilai 1. Ianya akan melaksanakan empat jenis operasi DAN, TAK, ekslusif-ATAU dan ATAU. Masukan S_0, S_1 dan S_2 akan menentukan operasi yang akan dilaksanakan. Rajah 5.4 menujukkan rekabentuk unit logik tersebut.



Rajah 5.4: Gambarajah Litar Logik ALU 8-bit

Dapat diperhatikan masukan A dan B akan melalui get-get DAN, ATAU TAK dan ekslusif-ATAU. Ia akan memasuki pemultipleks 4 ke satu dan akan ia akan memaparkan opewrasi yang akan ditentukan oleh masukan S_0, S_1 dan S_2. juga dapat diperhatikan bahawa pembawa sambutan (Carry in) dan keluaran C_{out} tidak diperlukan dalam unit ini.

5.1.3 REKABENTUK ALU 8-BIT



Rajah 5.5: Rekabentuk ALU 8-bit

Apabila semua sub unit ALU 8-bit selesai dibina, tibalah masa untuk menggabungkan nya. Daripada rajah 5.5 dapat dilihat rekabentuk lengkap sebuah ALU 8-bit. Kedua-dua keluaran unit aritmetik dan juga logik akan disambungkan ke pemultipleks 2 ke satu dan hasil akan ditentukan oleh .Op_code

Rekabentuk ini mampu melakukan operasi aritmetik dan logik seperti yang telah dinyatakan.

5.2 REKABENTUK BOUNDARY SCAN UNTUK ALU

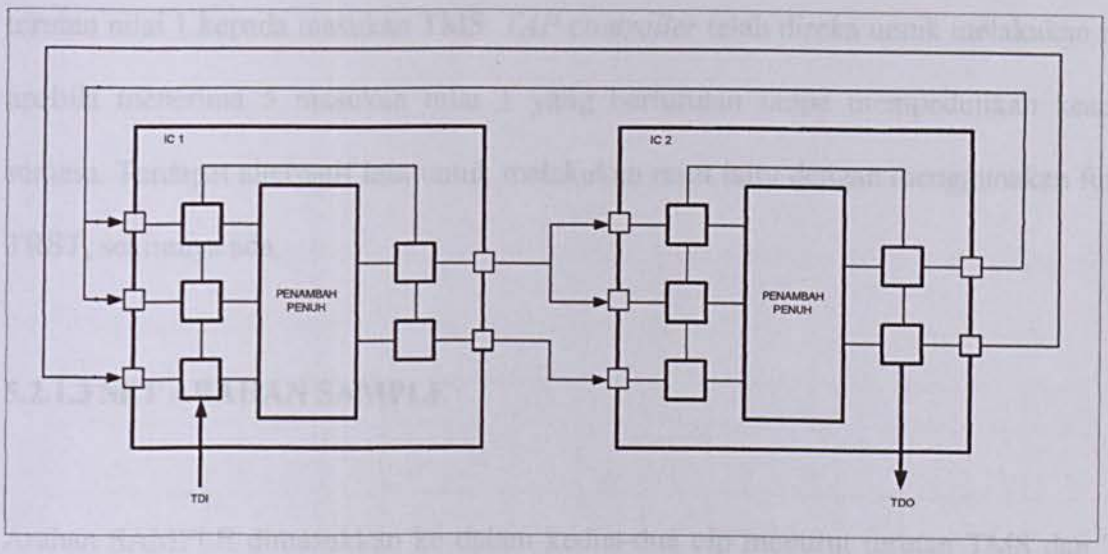
Pembangunan *boundary scan* dimulakan dengan membangunkan satu komponen asas ALU iaitu penambah penuh sebagai pengujian awal, Ia adalah kerana litar penambah penuh adalah tidak begitu kompleks berbanding dengan litar ALU. Ini adalah bagi membolehkan pemahaman yang mendalam mengenai teknik pengujian boundary scan yang akan dilaksanakan kerana litar penambah penuh mudah diterangkan dan mudah untuk difahami.

Maka, rekabentuk yang dijalankan akan bermula daripada pembangunan boudary scan untuk penambah penuh yang kemudiannya diikuti pula dengan pembangunan pengujian untuk ALU 8 bit.

5.2.1 BOUNDARY SCAN UNTUK PENAMBAH PENUH

Rekabentuk *boundary scan* untuk pengujian penambah penuh adalah seperti yang ditunjukkan dalam rajah 5.6. Dua buah penambah penuh yang akan diuji salinghubung di antara keduanya. Pengujian yang akan dijalankan adalah untuk menguji salinghubung kedua cip tersebut iaitu samaada ianya berfungsi dengan baik ataupun mempunyai masalah seperti litar terbuka (*open circuit*) dan litar pintas (*short circuit*).

Terdapat tiga masukan dan dua keluaran pada setiap cip penambah penuh. Keluaran daripada cip pertama akan disuapmasuk kepada masukan cip kedua. ini adalah untuk memastikan cip pertama mempunyai salinghubung dengan cip kedua.



Rajah 5.6: Litar pengujian 2 penambah penuh

5.2.1.1 NILAI AWALAN PENGUJIAN

Pengujian yang dilakukan akan menggunakan arahan EXTEST dan SAMPLE/PRELOAD sahaja. Nilai pendaftar arahan yang diandaikan pada setiap cip adalah dalam nilai 3-bit. Dalam pengujian ini, arahan SAMPLE diwakilkan dengan nilai 100 dan arahan EXTEST diwakilkan dengan nilai 000. Data ujian akan dimasukkan secara selari melalui TDI kepada pendaftar *boundary scan* (BSR) dan akan dianjak keluar melalui TDO.

5.2.1.2 RESET KEADAAN TAP

Keadaan TAP akan direset kepada keadaan *Test-Logic-Reset* dengan memasukkan turutan nilai 1 kepada masukan TMS. *TAP controller* telah direka untuk melakukan reset apabila menerima 5 masukan nilai 1 yang berturutan tanpa mempedulikan keadaan semasa. Terdapat alternatif lain untuk melakukan reset iaitu dengan menggunakan fungsi TRST, sekiranya ada.

5.2.1.3 SET ARAHAN SAMPLE

Arahan SAMPLE dimasukkan ke dalam kedua-dua cip menurut turutan TMS dan TDI seperti jadual 5.2. Turutan masukan 01100 pada masukan TMS membolehkan *TAP controller* berada pada keadaan *Shift-IR*. Pada keadaan ini, arahan SAMPLE (100) akan dianjak masuk ke dalam pendaftar arahan pada setiap IC. Pada keadaan *Update-IR*,

arahan dimasukkan ke dalam pendaftar pengkod arahan (*instruction decode*). Kemudiannya, *TAP controller* akan kembali kepada keadaan *Select-DR*.

State:	0	1	2	9	10	11	11	11	11	11	11	12	15	2
TMS:	0	1	1	0	0	0	0	0	0	0	1	1	1	
TDI:	-	-	-	-	-	1	0	0	1	0	0	-	-	

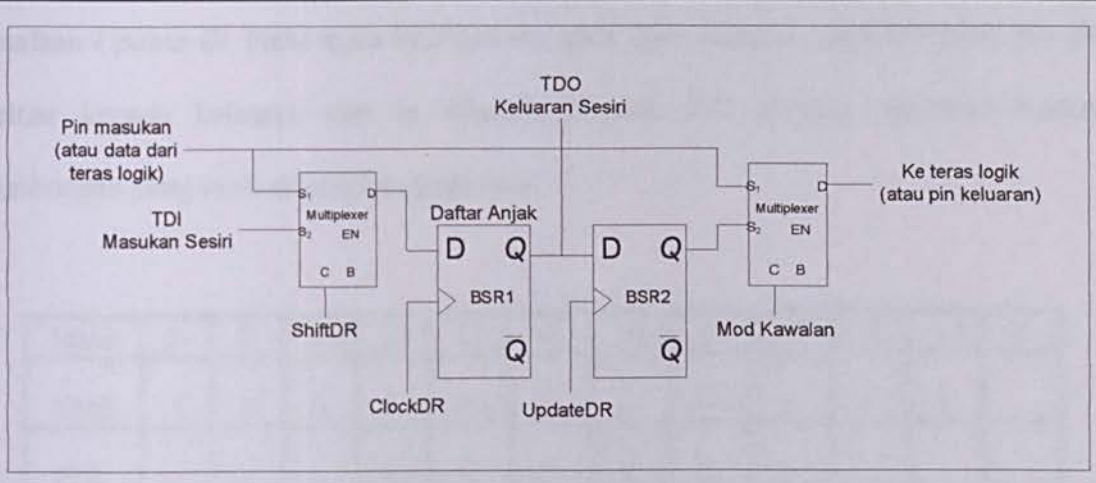
Jadual 5.2: Set arahan SAMPLE/RELOAD

5.2.1.4 MASUKAN DATA UJIAN

Ujian menggunakan contoh data bernilai 01101. Data ujian ini seterusnya dianjakkan ke dalam IC menggunakan turutan TMS dan TDI seperti di dalam jadual 5.3. Data ujian akan dianjakkan ke dalam BSR1 dan pada keadaan *Shift-DR*. Kemudian, pada keadaan *Update-DR*, data ujian tersebut akan dipindahkan ke dalam BSR2. Gambaran BSR1 dan BSR2 di dalam sel *boundary scan* boleh digambarkan di dalam rajah 5.7. BSR1 mewakilkan daftar anjak. Sementara itu, BSR2 pula akan menerima masukan secara selari daripada BSR1 apabila isyarat *update* diterima.

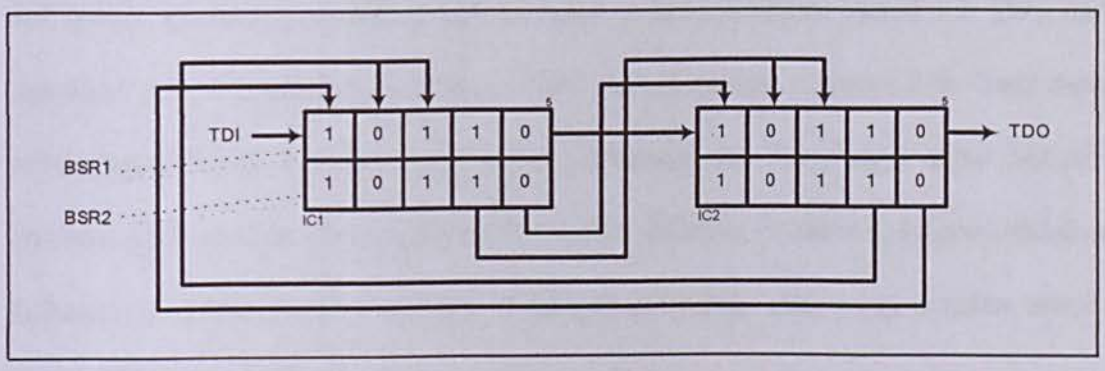
State:	2	3	4	4	4	4	4	4	4	4	4	4	5	8	2
TMS:	0	0	0	0	0	0	0	0	0	0	0	1	1	1	
TDI:	-	-	0	1	1	0	1	0	1	1	0	1	-	-	

Jadual 5.3: Set masukan data ujian



Rajah 5.7: Sel *boundary scan*

Gambaran kemasukan data ke dalam BSR1 dan BSR2 boleh digambarkan melalui rajah 5.8. Nilai masukan data ujian akan dianjakkan keseluruhan BSR.



Rajah 5.8: Masukan data ujian pada *boundary scan*

5.2.1.5 SET ARAHAN EXTEST

Arahan EXTEST akan diimbas kepada kedua-dua IC menggunakan turutan TMS seperti di dalam jadual 5.4. arahan EXTEST (000) akan diimbas masuk ke dalam pendaftar arahan pada keadaan *Shift-IR* dan dimasukkan ke dalam pendaftar pengkod arahan pada

keadaan *Update-IR*. Pada masa keadaan ini, data ujian yang dimasukkan pada IC1 akan keluar kepada keluaran dan ia dihantar kepada IC2 melalui masukan menerusi sambungan yang telah ditetapkan pada litar.

State:	2	9	10	11	11	11	11	11	11	12	15	2
TMS:	1	0	0	0	0	0	0	0	1	1	1	
TDI:	-	-	-	0	0	0	0	0	0	-	-	

Jadual 5.4: Set arahan EXTEST

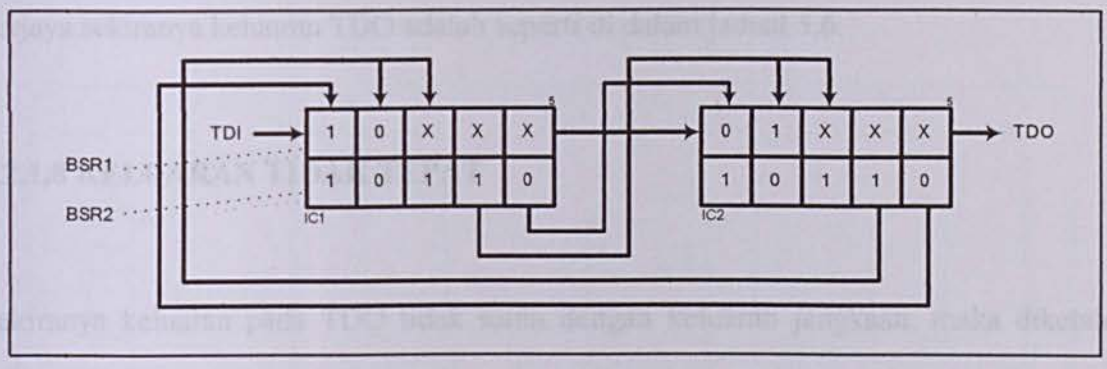
5.2.1.6 KELUARAN TDO

Keputusan daripada masukan IC akan dikesan. Data ini diimbas keluar kepada TDO dan set kedua data ujian diimbas masuk menurut turutan di dalam jadual 5.5. Data daripada masukan pin dimasukkan ke dalam BSR1 pada keadaan *Capture-DR*. Pada masa ini, sekiranya terdapat ralat dan masalah yang dihadapi oleh litar, ianya dapat dikesan pada set data ujian pertama. Sekiranya tiada masalah dikesan, keluaran jangkaan adalah seperti keluaran di dalam jadual 5.5. Nilai X adalah mewakili data yang diimbas tetapi tidak relevan kepada pegujian.

State:	2	3	4	4	4	4	4	4	4	4	4	4	5	8	2
TMS:	0	0	0	0	0	0	0	0	0	0	0	1	1	1	
TDI:	-	-	1	1	0	1	0	1	1	0	1	0	-	-	
TDO:	-	-	x	x	x	0	1	x	x	x	1	0	-	-	

Jadual 5.5: Set keluaran TDO set pertama

Keputusan pengujian dianjakan keluar daripada BSR1 pada keadaan *Shift-DR* dan data baru dianjak masuk. Data baru tersebut dimasukkan ke dalam BSR2 pada keadaan *Update-DR*. Gambaran keputusan ujian digambarkan seperti rajah 5.9.



Rajah 5.9: Keluaran pada TDO

5.2.1.7 KELUARAN JANGKAAN (*EXPECTED OUTPUT*)

Data keputusan yang terhasil akan diterima daripada masukan IC2. Data ini diimbas keluar kepda TDO dan nilai semua 0 diimbas masuk menurut turutan seperti jadual 5.6.

State:	2	3	4	4	4	4	4	4	4	4	4	4	5	8	2
TMS:	0	0	0	0	0	0	0	0	0	0	0	1	1	1	
TDI:	-	-	0	0	0	0	0	0	0	0	0	0	-	-	
TDO:	-	-	x	x	x	0	1	x	X	x	1	0	-	-	

Jadual 5.6: Set keluaran TDO keseluruhan

Data daripada input masukan akan dimasukkan ke dalam BSR1 pada keadaan *Capture-DR*. Kemudian ianya dianjak keluar pada keadaan *Update-DR*. *TAP controller* kemudianya akan kembali kepada keadaan *Test-Logic-Reset* dan operasi normal pada IC boleh dijalankan seperti biasa. Ujian salinghubung diantara dua cip tersebut dianggap berjaya sekiranya keluaran TDO adalah seperti di dalam jadual 5.6.

5.2.1.8 KELUARAN TIDAK TEPAT

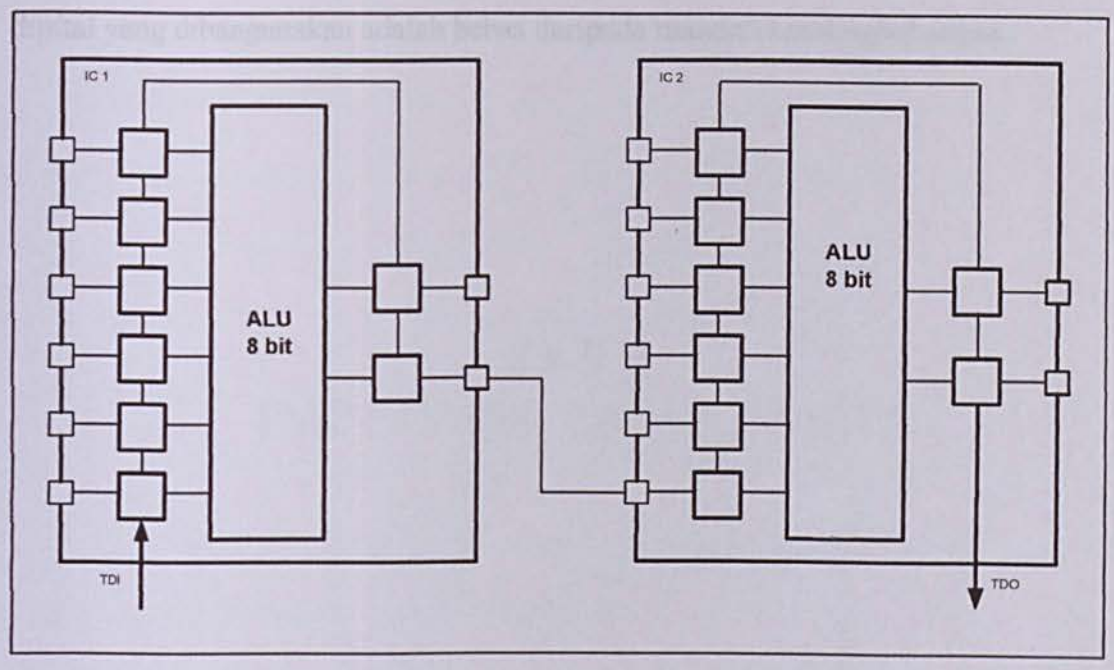
Sekiranya keluaran pada TDO tidak sama dengan keluaran jangkaan, maka diketahui bahawa terdapat masalah pada salinghubung antara dua cip berkenaan.

State:	2	3	4	4	4	4	4	4	4	4	4	4	5	8	2
TMS:	0	0	0	0	0	0	0	0	0	0	0	1	1	1	
TDI:	-	-	0	0	0	0	0	0	0	0	0	0	-	-	
TDO:	-	-	x	x	x	0	**0**	x	x	x	1	0	-	-	
TDO_jangkaan:	-	-	x	x	x	0	**1**	x	x	x	1	0	-	-	

Jadual 5.7: Set keluaran beserta ralat

5.3 BOUNDARY SCAN UNTUK ALU 8 BIT

Bagi rekabentuk *boundary scan* untuk ALU 8 bit pula, ia mempunyai rekabentuk yang serupa dengan penambah penuh dari segi konsepnya. Perbezaan antara kedua-duanya hanyalah pada bilangan pin-pin masukan dan keluaran beserta nilai data ujian yang dilakukan *boundary scan* untuk ALU 8 bit. Rujuk rajah 5.9



Rajah 5.10: Rekabentuk Boundary Scan untuk ALU 8-bit

5.4 KESIMPULAN

Daripada pengujian-pengujian yang dilaksanakan, didapati kaedah pengujian menggunakan *boundary scan* dapat menguji masalah salinghubung antara cip-cip. Pengujian tidak hanya terhad kepada pengujian antara 2 cip sahaja tetapi ianya meliputi pengujian untuk cip dalam julat yang besar. Sekaligus ianya dapat menjimatkan masa dan

meningkatkan kadar kebolehpercayaan terhadap pengujian melalui teknik *boundary scan* ini.

Pengujian terhadap ALU 8 bit menunjukkan 2 keadaan yang berkemungkinan boleh berlaku iaitu pada keadaan tiada sebarang masalah untuk salinghubung dan mempunyai masalah salinghubung. Apabila masalah ini dapat dikesan, ianya dapat memastikan litar digital yang dibangunakan adalah bebas daripada masalah kesalinghubungan.

BAB 6
IMPLEMENTASI SISTEM

BAB 6

IMPLEMENTASI SISTEM

6.0 PENGENALAN

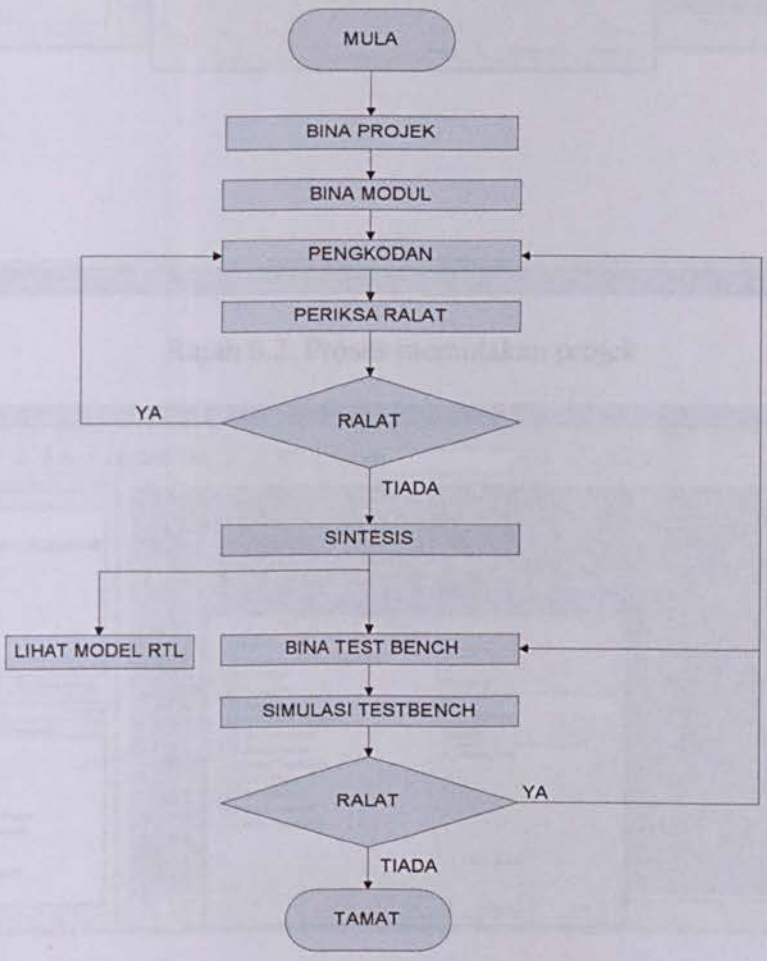
Setalah menjalani proses analisis dan reka bentuk sistem, maka tiba masa untuk sistem direalisasikan. Pembangunan dimulakan dsengan menghasilkan prototaip atau modul modul secara berasingan menggunakan aturcara VHSIC *Hardware Description Langguage* (VHDL). Ianya menggunakan platform xilinx dan kemudiannya akan melakukan proses pemerisaan sintaks dan kemudian akan melakukan proses sistesis dimana sintaks pengaturcaraan tersebut akan diterjemahkan kepada rekabentuk dan susunan get-get logik. Kesemua ini akan dilaksanakan selepas ekabentuk sistem dihasilkan mengikut susunatur yang dirancang.

Seterusnya, didalam fasa ini juga segala langkah dan strategi pembangunan akan dibincangkan bagi memastikan kelancaran pembangunan sistem. Dalam fasa ini, aktiviti yang paling terpenting adalah pengaturcaraan kerana ia adalah proses yang akan merealisasikan semua rekabentuk yang telah direka.

Dalam peringkat ini, segalanya adalah bergantung kepada penagturcaraan. Ianya memerlukan ketekunan dan kesabaran. Kesilapan dan sebarang ralat adalah akan mengakibatkan lengahan dalam pembangunan sistem.

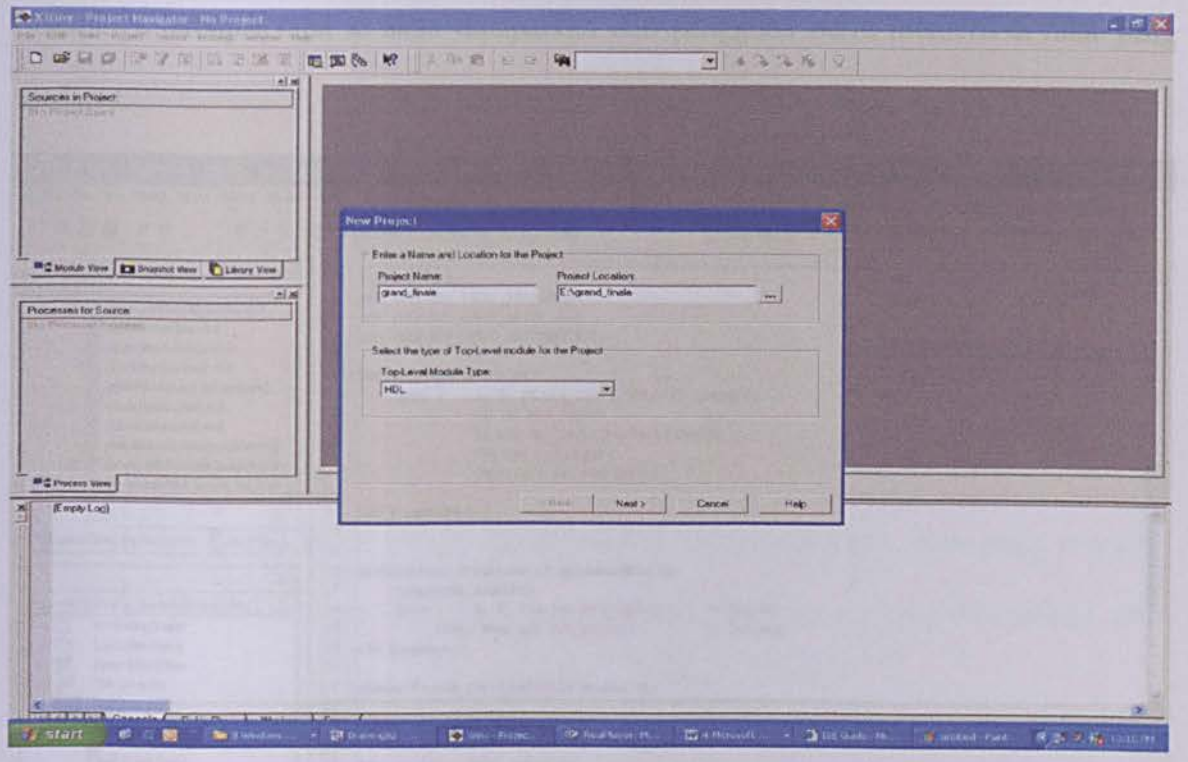
6.1 XILINX

Bagi membangunkan ALU dan Boudary scan ini, perisian yang diplih untuk membangunkannya adalah XILINX. Perkara yang pertama perlu dilakukan oleh pengguna xilinx adalah mencipta projek(create project) dimana satu *folder* atau ruang kerja akan diwujudkan dimana semua fail modul akan disimpan didalam ruang tersebut. Rajah 6.1 menunjukkna carta alir bagaimana sesebuah modul dihasilkan.

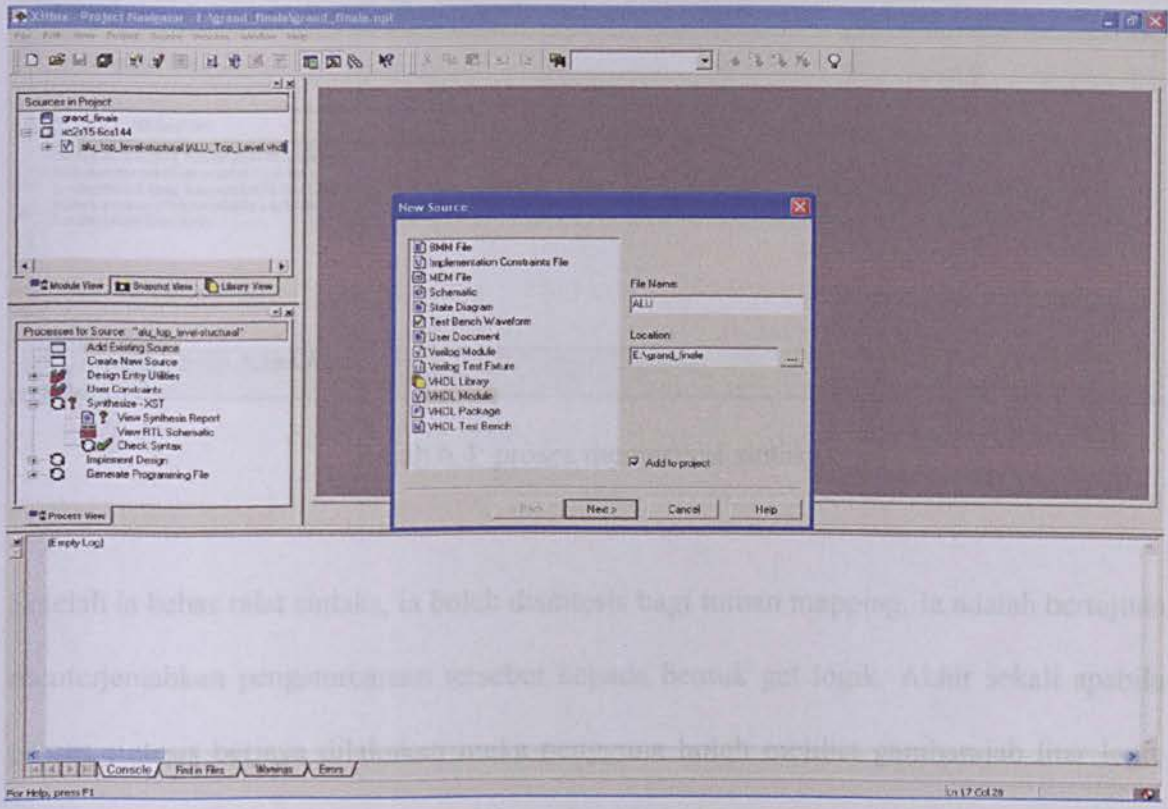


Rajah 6.1 Prosedur peggunaan Xilinx

Langkah seterusnya pula adalah untuk menulis kod atau aturcara ALU yang akan dibangunkan. Kod-kod tesebut akan disimpan dalam bentuk fail vhdl. Setelah kod selasai dimasukkan ia akan diperiksa sintaksnya bagi memastikan ianya bebas dari ralat.

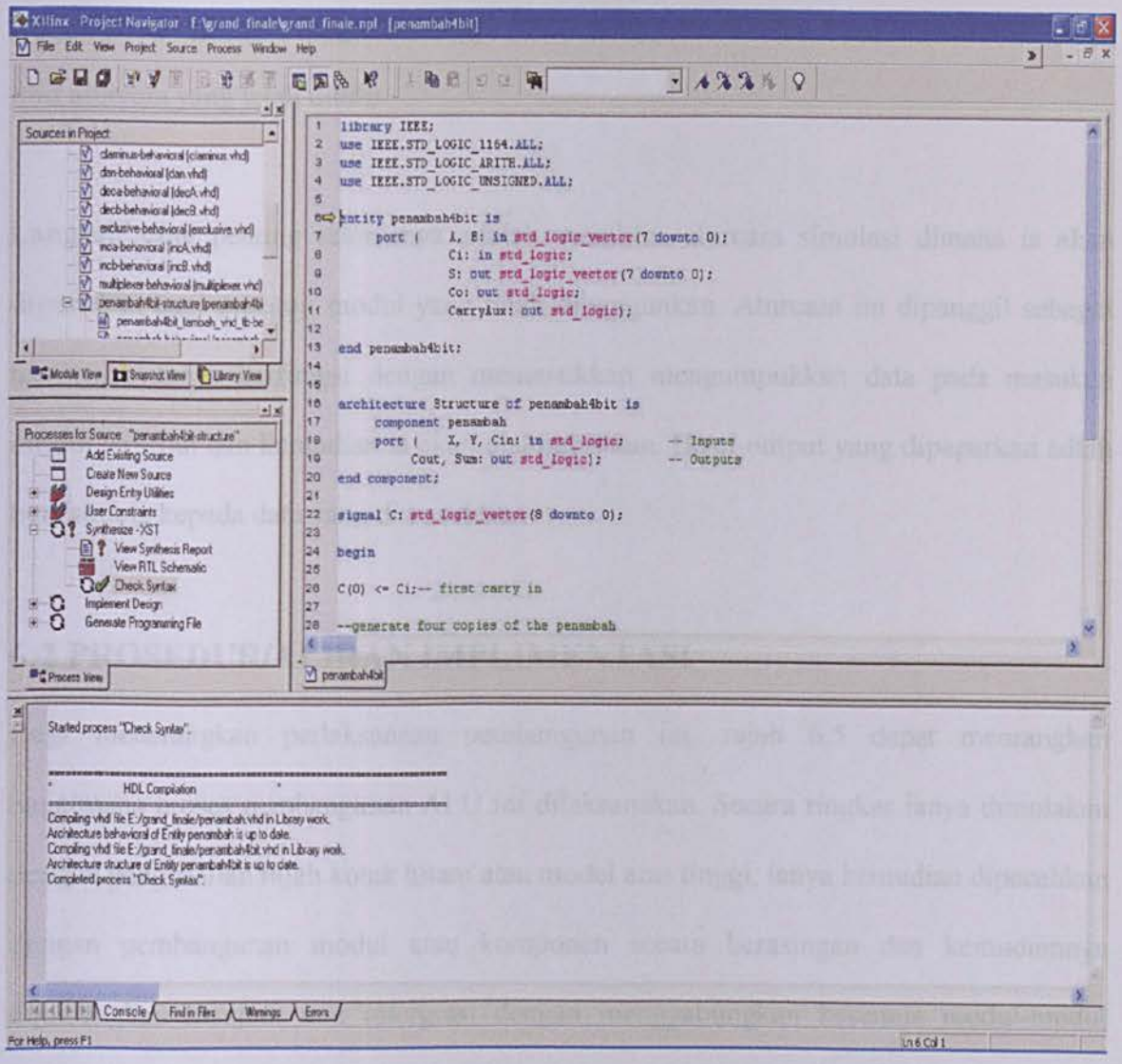


Rajah 6.2: Proses memulakan projek



Rajah 6.3: Proses memulakan modul

Sekiranya terdapat ralat ia akan dipaparkan dan pengguna perlu memeriksa ralat yang terdapat.



Rajah 6.4: proses memeriksa sintaks

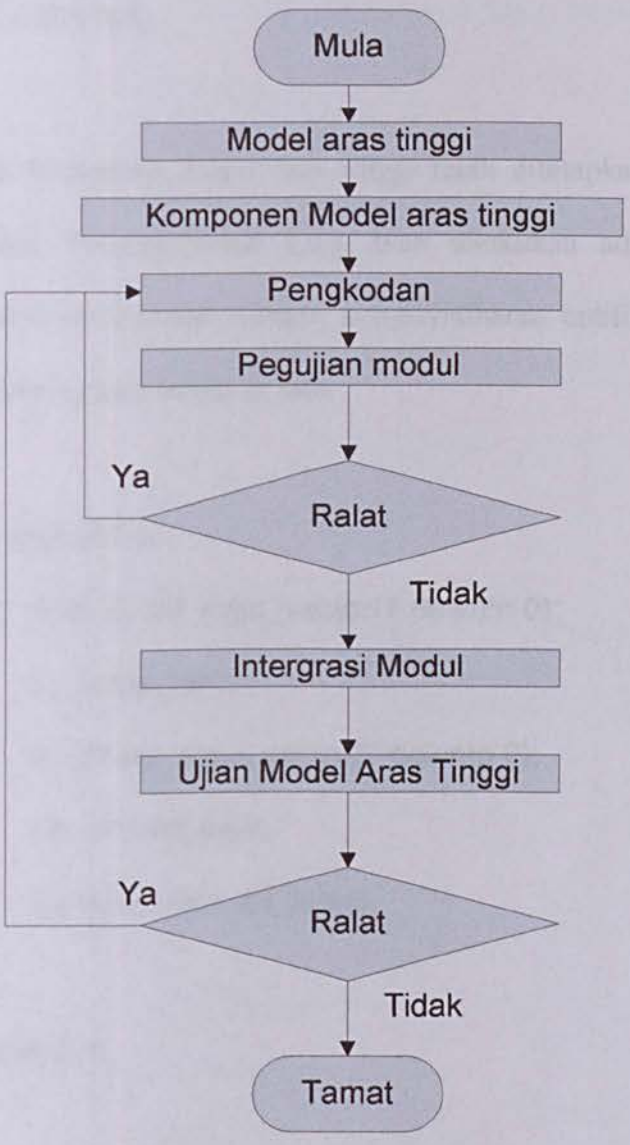
Setelah ia bebas ralat sintaks, ia boleh disintesis bagi tujuan mapping. Ia adalah bertujuan menterjemahkan pengaturcaraan tersebut kepada bentuk get logik. Akhir sekali apabila proses sintesis berjaya dilakukan maka pengguna boleh melihat gambarajah litar logic yang telah disintesis tadi dengan melaksanakan fungsi *view RTL schematic.*

Fungsi melihat gambarajah litar bukanlah satu perkara yang wajib untuk dilakukan. Ia hanyalah sebagai satu fungsi tambahan untuk pengguna malihat hasil implimentasi kod-kod aturcara yang telah ditulis.

Langkah yang penting seterusnya adalah membina aturcara simulasi dimana ia akan digunakan bagi menguji modul yang telah dibangunkan. Aturcara ini dipanggil sebagai *testbench*. Ianya berfungsi dengan memasukkan mengumpukkan data pada masukan modul tersebut dan kemudian ia akan disimulasikan. Hasil output yang dipaparkan adlah bergantung kepada data yang dimasukkan..

6.2 PROSEDUR/ALIRAN IMPLIMENTASI

Bagi menerangkan perlaksanaan pembangunan ini, rajah 6.5 dapat menrangkan bagaimana proses pembangunan ALU ini dilaksanakan. Secara ringkas ianya dimulakan dengan penghasilan rajah kotak hitam atau model aras tinggi, ianya kemudian dipecahkan dengan pembangunan modul atau komponen secara berasingan dan kemudiannya digabungkan menjadi unit intergrasi dengan menggabungkan kesemua modul-modul yang telah dibangunkan. Sudah semestinya pengujian secara konsisten bermula dari modul hingga kepada moel aras tinggi harus diuji.



Rajah 6.5: Aliran Pembangunan ALU

6.3 PNGATURCARAAN

Apabila reka bnetuk komponen model aras tinggi telah ditetapkan, maka bermulalah proses pengaturcaraan. Pengaturcaraan yang akan dilakukan adalah pengaturcaraan VHDL. Pengaturcaraan ini bermula dengan pengisystiharan entiti. Contoh berikutnya adalah bagaimana entiti sesuatu modul di tulis

```
entity penambah8bit is
        port (  A, B: in std_logic_vector(7 downto 0);
                Ci: in std_logic;
                S: out std_logic_vector(7 downto 0);
                Co: out std_logic;
                CarryAux: out std_logic);

end penambah8bit;
```

daripada contoh diatas dapat dilihat bagaimana entiti diisytiharkan, ianya mengandungi pengisytiharan port ataupun pin masukan dan juga keluaran. Contoh diatas menunjukkan pengisytiharan *penambah8bit.*

Seterusnya fungsi modul dan juga cara ia beroperasi akan di tulis. Ia melibatkan pengsitiharan *architecture*, secara amnya ia kana mengenakan operasi keatas nilai yang dimasukan. Ianya begantung samada secara selari atau berjujukan. Penjanaan isyarat

anata modul pula akan menggunakan analogi signal bagi mengadaptasikan konsep pendawaian(*wiring*) dianatara komponen.

Dalam pembangunan ini *architecture* akan digunakan bagi mendiskripiskan *behavior* dan *strucuture* dalam modul-modul yang akan dibangunkan. Contoh dibawah wakan menerangkan bagaimana ia di lakukan.

```
architecture Structure of penambah4bit is

    component penambah
            port (  X, Y, Cin: in std_logic;        -- Inputs
                    Cout, Sum: out std_logic);      -- Outputs
    end component;

    signal C: std_logic_vector(8 downto 0)

begin

C(0) <= Ci;-- first carry in

--generate four copies of the penambah

FullAdd4: for i in 0 to 7 generate

begin
FAx: penambah port map (A(i), B(i), C(i), C(i+1), S(i));
end generate FullAdd4;

Co <= C(8);
CarryAux <= C(3);
end Structure;
```

selain dari itu komponen juga menjadi asas dalam pembanguan ALU ini dimana secara keseluruhan terdapat 18 komponen didalam ALU ini.

Contoh pengaturcaraan CLA:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY CLA IS
   PORT
      (
      x_in     : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
      y_in     : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
      carry_in : IN  STD_LOGIC;
      sum      : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
      carry_out: OUT STD_LOGIC
      );
END CLA;
ARCHITECTURE behavioral OF CLA IS
SIGNAL   h_sum          :  STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL   carry_generate   :  STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL   carry_propagate  :  STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL   carry_in_internal :  STD_LOGIC_VECTOR(7 DOWNTO 1);
BEGIN
   h_sum <= x_in XOR y_in;
   carry_generate <= x_in AND y_in;
   carry_propagate <= x_in OR y_in;
   PROCESS (carry_generate,carry_propagate,carry_in_internal)
   BEGIN
   carry_in_internal(1) <= carry_generate(0) OR (carry_propagate(0) AND
carry_in);
      inst: FOR i IN 1 TO 6 LOOP
         carry_in_internal(i+1)      <=      carry_generate(i)      OR
(carry_propagate(i) AND carry_in_internal(i));
         END LOOP;
   carry_out  <=  carry_generate(7)  OR  (carry_propagate(7)  AND
carry_in_internal(7));
   END PROCESS;
   sum(0) <= h_sum(0) XOR carry_in;
   sum(7 DOWNTO 1) <= h_sum(7 DOWNTO 1) XOR carry_in_internal(7
DOWNTO 1);
END behavioral;
```

6.4 PENGUJIAN

Untuk memastikan agar ALU dan komponen-komponennya berfungsi dengan betul dan menepati kehendak pembangunan, maka pegujian modul-modul tersebut telah dilakukan secara berterusan daripada awal hingga ke akhir pembangunan.

6.5 KESIMPULAN

Dalam membangun ALU dan Boundary Scan ini, teknik yang digunakan untuk membangunkannya adalah dengan pembangunan modul secara berasingan dan kemudian ianya akan digabungkan untuk menjadi satu unit induk seperti yang dikehendaki.

BAB 7
PENGUJIAN

BAB 7

PENGUJIAN

7.0 PENGENALAN

Pengujian sistem adalah proses yang paling penting dalam pembangunan sistem kerana ia bertujuan untuk mementusahkan bahawa sistem yang dibangunkan tidak akan menagalami kecacatan opersi dan juga menepati spesifkasi pembangunan sistem. Proses pengujian yang dilakukan akan dapat mengesan ralat yang berlaku tehadap sistem atau pun kegagalan yang terdapat pada perisaian yang digunakan untuk membangunkan sistem. Kegagalan perisian yang dimaksudka adalah merujuk kepada sejauh mana sistem mampu berfungsi dengan baik dan tepat semasa pengujian dilakukan

7.1 RASIONAL PELAKSANAAN PENGUJIAN

- Memastikan sistem bebas ralat sebelum memasuki pasaran
- Memastikan sistem menepati piawaian.

Umumnya pembangunan ALU dan Boundary Scan ini melibatkan dua pengujiuan iaitu pengujian modul dan pengujian Model aras tinggi.

7.2 PENGUJIAN MODUL

Modul merupakan komponen kepada model aras tinggi dan ianya boleh beroperasi sendiri tanpa bergantung kepada komponen lain. Contohnya penambah 8bit dan juga CLA.

Proses pengujian ini melibatkan simulasi terhadap modul dengan maukan data yang diumpukkan didalam testbench. Testbench adalah satu set aturcara dimana ia mempunyai maklumat mengenai modul tesebut dan mempunyai bahagian definisi pengguna terhadap data ujian yang aan dimasukkan. Data yang dimasukkan merupakan data atau isyarat digital. Hasildari simulasi ini akan dibandingkan dengan output jangkaan yang dilakukan secara manual. Ini dapat memastikan ketepatan keluaran. Adalah diharapkan agar ralat yang terdapat didalam modul ini tidak akan mempengaruhi integriti keseluruhan cip yang dibina. Aturcara dibawah merupakan contoh testbench.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY penambah4bit_tambah_vhd_tb IS
END penambah4bit_tambah_vhd_tb;

ARCHITECTURE behavior OF penambah4bit_tambah_vhd_tb IS

    COMPONENT penambah4bit
    PORT(
        A : IN std_logic_vector(7 downto 0);
        B : IN std_logic_vector(7 downto 0);
        Ci : IN std_logic;
        S : OUT std_logic_vector(7 downto 0);
        Co : OUT std_logic;
        CarryAux : OUT std_logic
```

```
        );
    END COMPONENT;

    SIGNAL A :  std_logic_vector(7 downto 0);
    SIGNAL B :  std_logic_vector(7 downto 0);
    SIGNAL Ci :  std_logic;
    SIGNAL S :  std_logic_vector(7 downto 0);
    SIGNAL Co :  std_logic;
    SIGNAL CarryAux :  std_logic;

BEGIN

    uut: penambah4bit PORT MAP(
            A => A,
            B => B,
            Ci => Ci,
            S => S,
            Co => Co,
            CarryAux => CarryAux
    );


-- *** Test Bench - User Defined Section ***
  tb : PROCESS
  BEGIN
  A<="00001111";
  B<="00111100";
  Ci<='0';
    wait; -- will wait forever
  END PROCESS;
-- *** End Test Bench - User Defined Section ***

END;
```

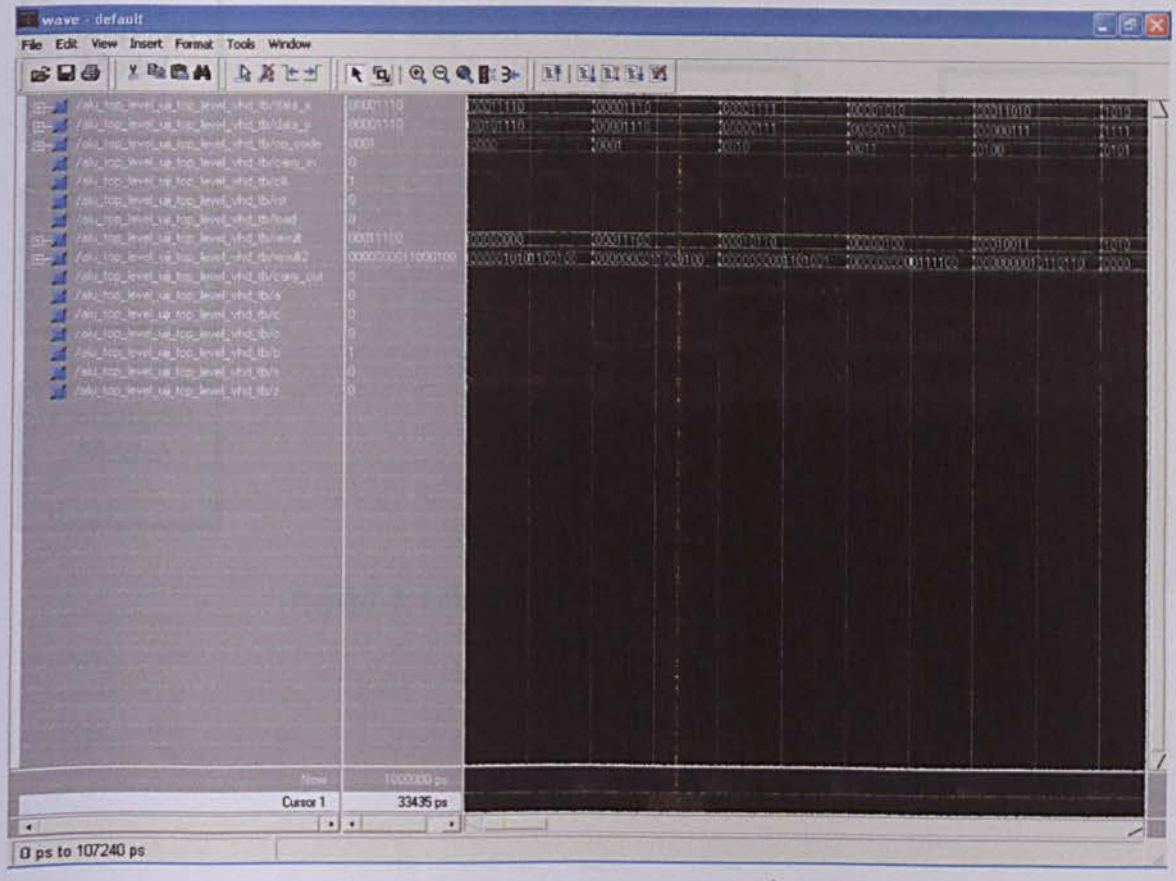
7.3 PENGUJIAN MODEL ARAS TINGGI

Ini merupakan fasa terakhir pengujian sistem dimana kesemua modulyang telah diintergrasikan akan diuji sebagi satu sistem induk.. sama seperti pengujian modul hasil aoutput simulasi akan dibandingkan dengan nilai kiraan secara manual. Aturcara dibawah merupakan aturcara testbench bagi pengujian model aras tinggi.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY alu_top_level_uji_top_level_vhd_tb IS
END alu_top_level_uji_top_level_vhd_tb;

ARCHITECTURE behavior OF alu_top_level_uji_top_level_vhd_tb IS

    COMPONENT alu_top_level
    PORT(
        Data_X : IN std_logic_vector(7 downto 0);
        Data_Y : IN std_logic_vector(7 downto 0);
        Op_code : IN std_logic_vector(3 downto 0);
        Carry_in : IN std_logic;
        Clk : IN std_logic;
        Rst : IN std_logic;
        Load : IN std_logic;
        Result : OUT std_logic_vector(7 downto 0);
        Result2 : OUT std_logic_vector(15 downto 0);
        Carry_out : OUT std_logic;
        a : OUT std_logic;
        c : OUT std_logic;
        o : OUT std_logic;
        p : OUT std_logic;
        s : OUT std_logic;
        z : OUT std_logic
        );
    END COMPONENT;

    SIGNAL Data_X :  std_logic_vector(7 downto 0);
    SIGNAL Data_Y :  std_logic_vector(7 downto 0);
    SIGNAL Op_code :  std_logic_vector(3 downto 0);
    SIGNAL Carry_in :  std_logic;
    SIGNAL Clk :  std_logic;
    SIGNAL Rst :  std_logic;
```

```
        SIGNAL Load :  std_logic;
        SIGNAL Result :  std_logic_vector(7 downto 0);
        SIGNAL Result2 :  std_logic_vector(15 downto 0);
        SIGNAL Carry_out :  std_logic;
        SIGNAL a :  std_logic;
        SIGNAL c :  std_logic;
        SIGNAL o :  std_logic;
        SIGNAL p :  std_logic;
        SIGNAL s :  std_logic;
        SIGNAL z :  std_logic;

BEGIN

        uut: alu_top_level PORT MAP(
                Data_X => Data_X,
                Data_Y => Data_Y,
                Op_code => Op_code,
                Carry_in => Carry_in,
                Clk => Clk,
                Rst => Rst,
                Load => Load,
                Result => Result,
                Result2 => Result2,
                Carry_out => Carry_out,
                a => a,
                c => c,
                o => o,
                p => p,
                s => s,
                z => z
        );

  CLOCK: process
   begin
      Clk <= '1';
      wait for 10 ns / 2;
      Clk <= '0';
      wait for 10 ns / 2;
   end process;
-- *** Test Bench - User Defined Section ***
  tb : PROCESS
  BEGIN
                Data_X <="00011110";
                Data_Y <="00101110";
                Op_code <="0000";
                Carry_in <='0';
```

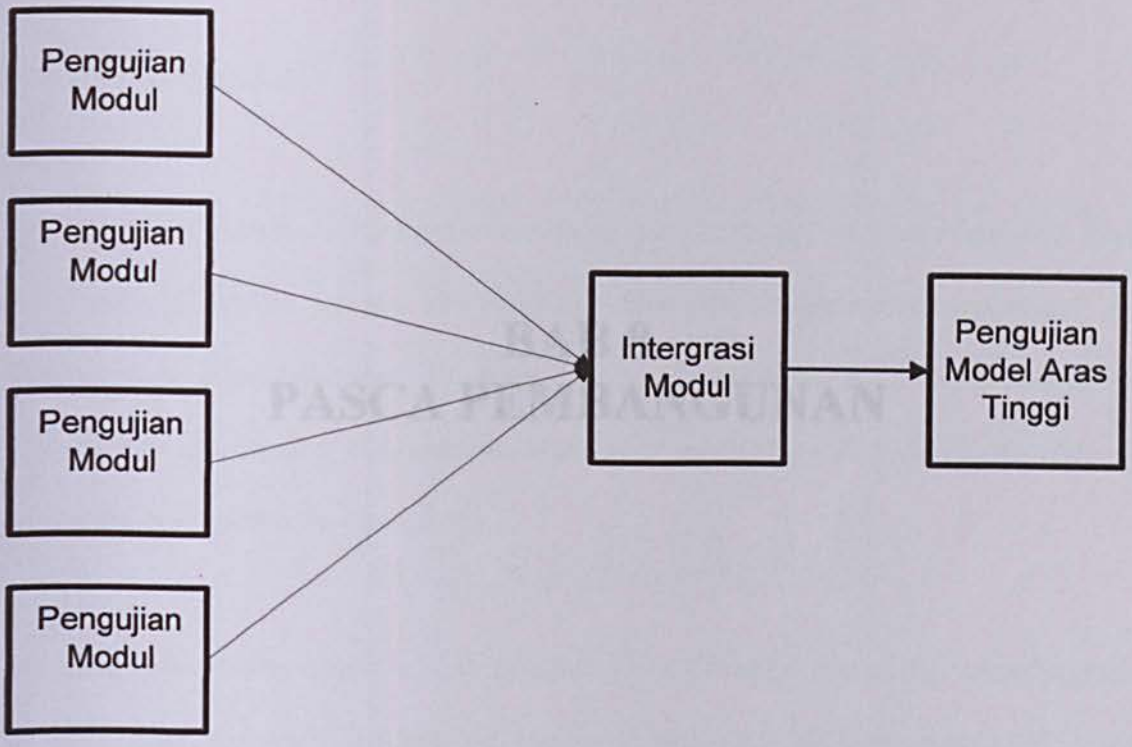
```
            Rst <='0';
            Load <='0';
            wait for 20 ns;

            Data_X <="00001110";
            Data_Y <="00001110";
            Op_code <="0001";
            Carry_in <='0';
            Rst <='0';
            Load <='0';
            wait for 20 ns;
            wait; -- will wait forever
  END PROCESS;
END;
```



Rajah 7.1: Graf isyarat keluaran testbench.

7.4 KESIMPULAN

Secara keseluruhan, dua peringkat pengujian telah membuktikan bahawa ALU yang dibangunkan adalah bebas ralat dan juga mencapai objektif sistem yang dirancang pembangunannya pada peringkat wal. Rajah 7.2 merujuk kepada gambaran bagaimana ujian dilakukan terhadap ALU.



Rajah7.2: Ujian yang dilakukan terhadap ALU.

BAB 8
PASCA PEMBANGUNAN

BAB 8

PASCA PEMBANGUNAN

8.0 PENGENALAN

Setelah semua aktiviti pembangunan dan pengujian telah selesai, maka penilaian keatas ALU perlu dilakukan semula. Ia bertujuan untuk menilai semula relevan projek yang dibangunkan serta beberapa isu yang boleh dibangkitkan bagi menambahbaikkan keseluruhan sistem yang dibangunkan. Ianya amat penting untuk tujuan kesesuaian dan perkembangan pada masa akan datang.

Isu yang akan dibincangkan adalah masalah yang dihadapi sepanjang membangunkan ALU ini, seterusnya pula perbincangan mengenai kelebihan dan kelemahan ALU ini akan dibincangkan , akhir sekali cadangan-cadangan yang mungkin dapat diketengahkan bagi meningkatkan lagi keupayaan dan juga kualiti ALU ini pada masa akan datang.

8.1 MASALAH YANG DIHADAPI

Antara masalah yang dihadapi semasa pembangunan sistem adalah.

- Kesukaran mencari contoh ALU 8 bit
- Pemilihan perisian pembangunan, samada mamilih Peak FPGA atau Xilinx.
- Kesukaran mendapatkan lesen pengguanaan Model Sim bagi tujuan simulasi projek
- Perisan yang digunakan (Xilinx) tidak menyokong beberapa fungsi, sematik dan juga frasa didalam bahasa pengaturcaraan VHDL
- Ralat logik yang berlaku dalam sistem mengakibatakan pembinaan sesetengah modul akan dibuat berulangkali dan juga pangujian secara sepsifik akan dilakukan.
- Kurang pengalaman dalam membangunkan projek.

8.2 KELEBIHAN DAN KELEMAHAN ALU

Seperti yang telah diketahui ALU 8-bit ini mempumyai 15 operasi yang seperti yang telah dinyatakan. Berikut adalah beberapa kelebihan dan kelemahan ALU

8.2.1 Kelebihan

- Rekabentuk ALU amat mudah difahamai dengan melihat rajah skematik RTL
- Modul dapat diubahsuai dengan mudah
- Kos yang minimum untuk membangunkannya
- Pembangunan yang bebas ralat hasil penggunaan bahasa pengaturcaraan VHDL
- Penggunaan CLA atau *Carry Lookahead Adeer* dalam unit arimetik ini dapat mengurangkan lengahan dalam litar.
- Semua modul dapat melaksanakan operasi dengan serentak
- Penggunaan Xilinx amat mudah dan menyeronokkan
- Fugsi *View RTL Schematic* amat memudahkan kerja pembangunan
- Fungsi *synthesis* menjadikan projek ini lebih praktikal.

8.2.2Kekurangan

- Fungsi pendaraban tidak dapat dihasilkan bersama-sama dengan ALU ini kerana masalah teknikal iaitu dari segi bit (hasil) yang keluar. Jesteru, satu set modul berasingan terpaksa di bina bagi tujuan pendaraban sahaja.
- Rekabentuk yang sedikit kompleks menghalang penggabungan secara keseluruhan dengan *boudary scan*
- Fungsi pembahagi tidak dapat laksanakan langsung akibat masalah *library* yang tidak menyokong fungsi *package*.

8.3 CADANGAN

Pada masa akan datang, adalah diharapkan ALU ini akan diperbaiki dan juga di tingkatkan keupayaannya. Antara cadangan yang boleh dibuat adalah

- Penambahan beberapa fungsi yang masih belum diletakana didalam ALU tersebut
- Mengoptimumkan litar yang telah dibina dengan cara mengoptimumkan sintaks pengaturcaraan
- Menyediakan lebih banyak penyelidikan untuk menghasilkan cip yang lebih baik berdasarkan ALU yang dibina sekarang.
- Pembangunan yang berterusan sehingga kepad pembinaan cip agar ia akan dapat diuji dengan sebenar.

8.4 KESIMPULAN

Setelah lapan bulan lamanya projek nin dilakukan akhirnya ia telah samapai ke penghujungnya. Selama itu jua pelbagai cabaran dan juga kekangan telah dihadapi, namun ia semua bukan sebagai pematah semangat bagi meneruskan projek ini. dalam menjayaka projek ini ketabahan, ketekunan dan kesabaran adlah tonggak kepada kejayaan besar dalam projek ini. saya amat berbangga dengan projek yang telah buat ini dan saya amat berharap agar ia akan menjadi satu pencetus kepada penghasilan ALU yang lebih baiik pada masa hadapan.

Akhir sekali saya amat bersyukur kerana semuanya telah berakhir dengan jayanya. Segala penat lelah saya semasa menuntut ilmu di Universiti Malaya ini akhirnya membolehkan saya menjalani latihan ilmiah ini dengan jayanya

RUMUSAN

RUMUSAN

Pada masa sekarang, penggunaan litar bersepadu atau cip bukanlah perkara asing bagi kita. Jika dilihat, kebanyakan barangan elektronik yang terdapat didalam kehidupan seharian kita daripada mesin pengira hinggalah kepada komputer, ianya mempunyai cip yang dapat melaksanakan tugas yang telah diprogramkan. Maka dengan itu, keperluan untuk memastikan bahawa ianya adalah berada dalam keaadaan yang baik dan sempurna amatlah penting. Oleh kerana kompleksnya binaan litar bersepadu ini disebabkan oleh saiz litar, ia telah menyaksikan kesan terhadap pertambahan kos dalam pengujian. Jesteru, penghasilan mekanisma pengujian ALU menggunakan teknik boundary scan ini adalah satu platform untuk menghasilkan produk pengujian yang efisyen dan efektif.

Pembangunan ALU adalah aspek penting dalam projek ini. ianya adalah komponen atau bahan pengujian yang kan dilaksanakan nanti. Tidak lupa juga dengan boundary scan, bagi memastikan ALU adalah bebas ralat ia akan menjadi alat untuk mengukur kebolehpercayaan sesuatu ALU yang dibangunkan.

Setelah menjalani WXES 3181 dengan jayanya maka, didalam WXES 3182 segala yang telah dinyatakan didalam WXES 3181 akan direlisasikan.sesungguhnya ia bukanlah satu kerja yamg mudah. Malah ianya mempunyai lebih banyak cabaran dari yang sebelumnya. Namun akhirnya ia dapat juga diselesaikan

Secara keseluruhannya, projek ini telah menjadi lebih berkeyakinan untuk mempraktikkan segala pelajaran yang telah dipelajari semasa tiga tahun menuntut dalam bidang sains komputer FSKTM Universiti Malaya.

RUJUKAN

RUJUKAN

1. Charles H. Roth. Jr, (1996). *Digital Systems Design Using VHDL*. PWS Publishing Company.
2. Peter J. Ashenden, (1996). *The Designer's Guide To VHDL*, Morgan Kaufman Publisher Inc.
3. R. Omid. *Test Access Port and Boundary-Scan Architecture*. UPII Seminar. 1-6.
4. J, Bhasker. (1996). *A VHDL Primer*. Bell Laboratories, Lucent Technologies, Allentown, PA.
5. Peter J. Ashenden. (July, 1990). *The VHDL Cookbook*. Dept. Computer Science University of Adelaide South Australia
6. Mazlena Salleh, Hazinah Kutty Mammi, Nor Afida Ithnin, Mazura Mat Din. (2000) *Organisasi Komputer Dan Bahasa Himpunan*. Universiti Teknologi Malaysia, Malaysia
7. William Stallings, (2003) *Computer Organization Architecture*. Pearson Education, Upper Saddle River N.J.
8. http://www.xilinx.com
9. http://www.asset-intertech.com
10. http://www.semiconductorfabtech.com
11. http://members.aol.com/SGalaxyPub/useful_links_vhdl.htm
12. http://www.cs.ucr.edu/content/esd/labs/tutorial/
13. http://users.senet.com.au/~dwsmith/vhdl.htm
14. http://www.cs.ucr.edu/content/esd/labs/tutorial/

15. http://www.altera.com/support/examples/vhdl/vhdl.html

16. http://www.cs.umbc.edu/help/VHDL/samples/samples.html

17. Zainalabedin Navabi. (1998) VHDL: Analysis and Modeling of Digital Systems. Second Edition International Edition. Northeastern University, University of Tehran. McGraw-Hill

LAMPIRAN 1

MANUAL PENGGUNA

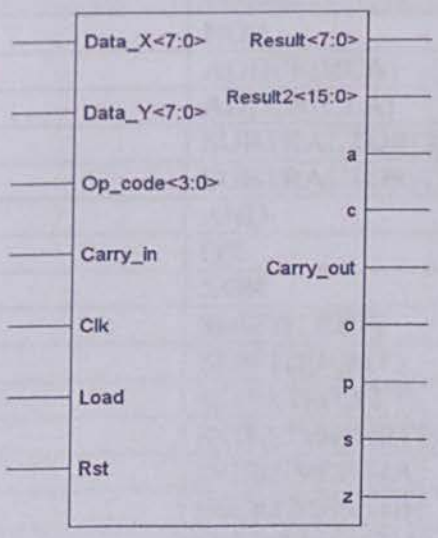
KANDUNGAN

1. PENGENALAN

ALU adalah litar bersepadu yang mempunyai set pengoperasian arimetik dan logik. Ianya biasa digabungkan didalam mikropemproses. Elemen-elemen yang terkandung dalam sesebuah komputer seperti daftar, ingatan dan unit kawalan akan berhubung dengan ALU untuk diproses dan kemudiannya hasil operasi yang dilaksanakan di ALU akan dihantar semula kepada komponen-komponen yang tertentu.

Fungsi ALU di dalam sesebuah komputer adalah sebagai satu komponen elektronik yang melaksanakan set-set arahan logik Boolean (0 dan 1) dalam bentuk digital.. ALU juga mengadungi set bendera yang berfungsi untuk mengeluarkan hasil bagi operasi yang dilakukan. Unit kawalan pula berfungsi untuk menyediakan isyarat bagi mengawal operasi dan pergerakan data keluar masuk melalui ALU.

2. ALU TOP LEVEL



]

Rajah 1: ALU Top Level

Penerangan pin

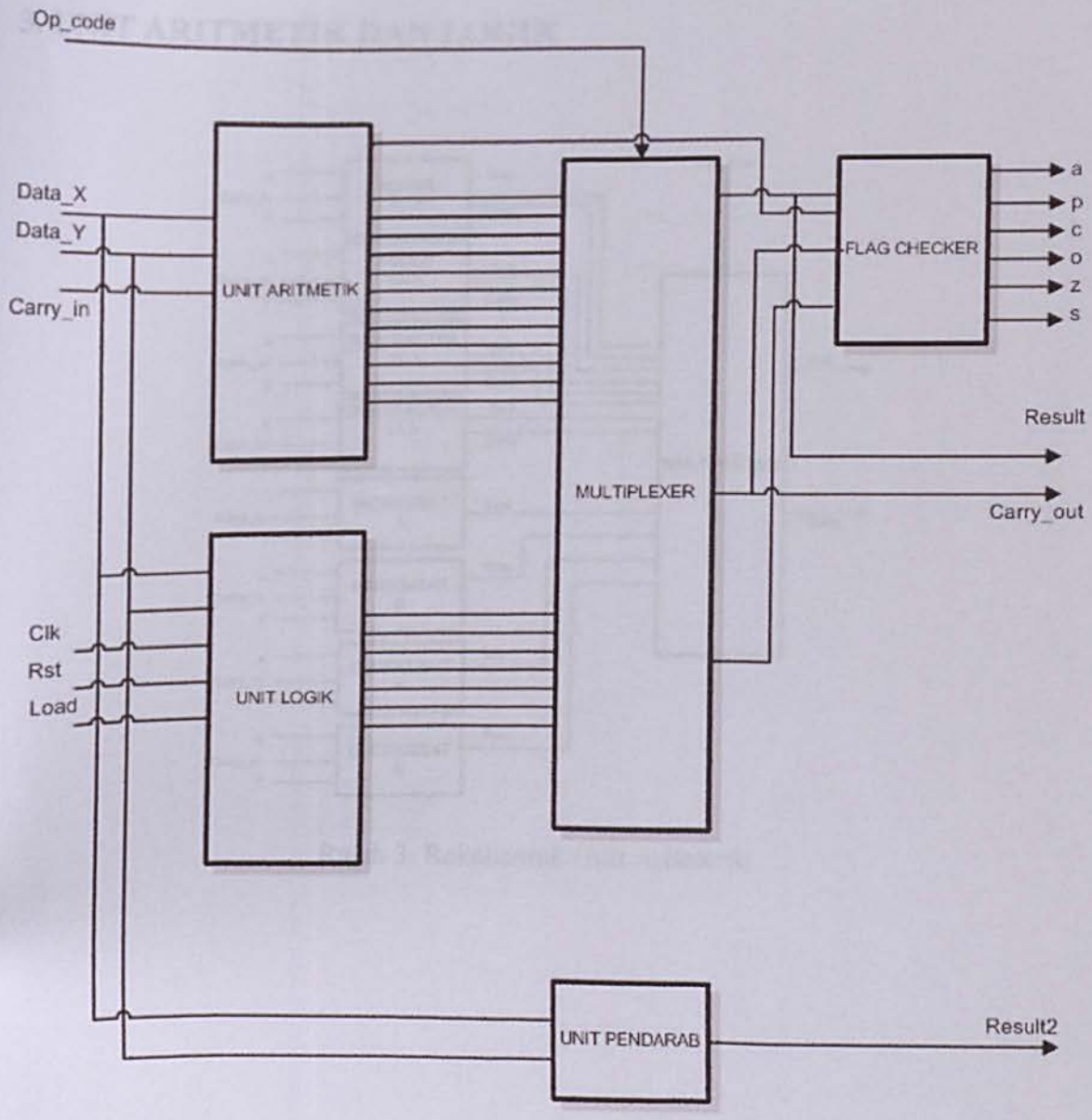
Jenis	Fungsi
Data_X & Data_Y	Masukan data 8 bit
Carry_in	Pembawa masukan
Op_code	Penentu operasi
Clk, Rst, Load	Pengawalan Shifter dan rotator
Result	Keluaran
Result2	Keluaran untuk pendarab
Carry_out	Pembawa keluaran
a,c ,o,p,s,z	Bendera

Jadual 1. Penerangan Pin

Operasi ALU

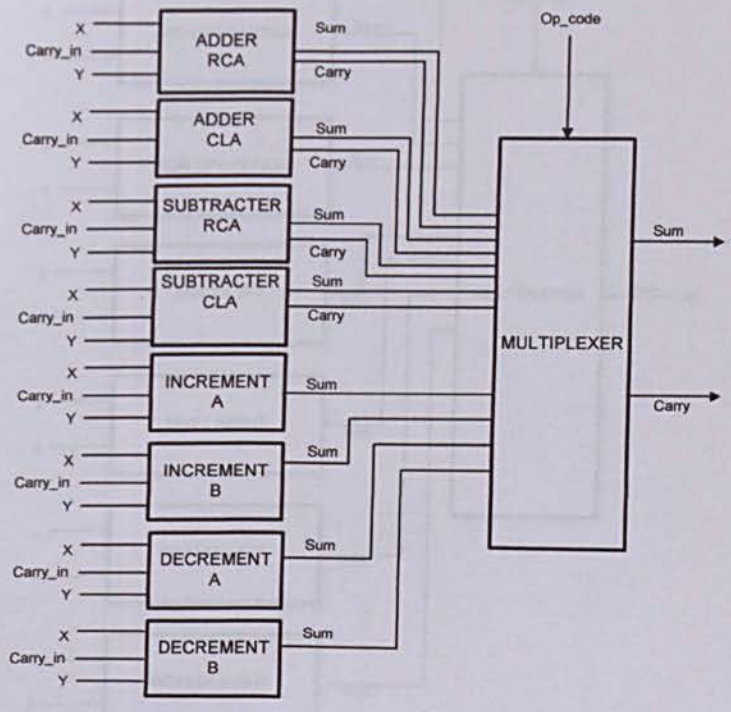
Nilai Bit	Operasi
0000	NOP
0001	ADDER(RCA)
0010	ADDER(CLA)
0011	SUBTRACTOR(RCA)
0100	SUBTRACTOR(CLA)
0101	AND
0110	OR
0111	XOR
1000	SHIFT(LEFT)
1001	SHIFT(RIGHT)
1010	ROTATE(LEFT)
1011	ROTATE(RIGHT)
1100	INCREMENT(A)
1101	INCREMENT(B)
1110	DECREMENT(A)
1111	DECREMENT(B)

Jadual 2: Operasi ALU

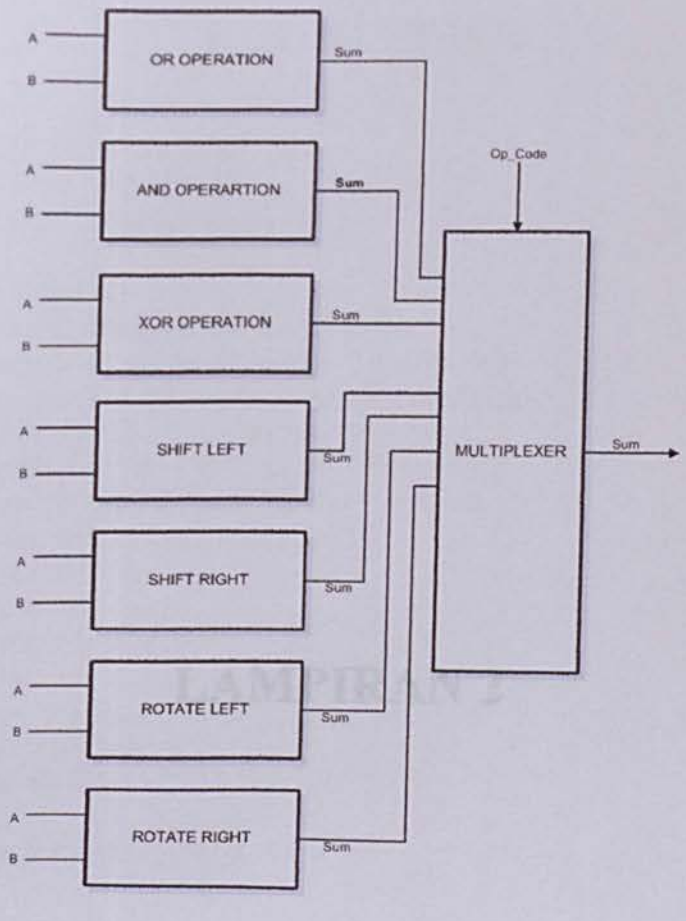


Rajah 2: Rekabentuk ALU

3. UNIT ARITMETIK DAN LOGIK



Rajah 3. Rekabentuk Unit Aritmetik



Rajah 4. Rekabentuk Unit Logik

LAMPIRAN 2

KOD SUMBER

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

entity ALU_Top_Level is
port(   Data_X: in std_logic_vector(7 downto 0);
        Data_Y: in std_logic_vector(7 downto 0);
        Op_code: in std_logic_vector(3 downto 0);
        Carry_in, Clk, Rst, Load: in std_logic;
        Result: out std_logic_vector(7 downto 0);
        Result2: out std_logic_vector(15 downto 0);
        Carry_out: out std_logic;
        a: out std_logic;
        c: out std_logic;
        o: out std_logic;
        p: out std_logic;
        s: out std_logic;
        z: out std_logic);
end ALU_Top_Level;

architecture Stuctural of ALU_Top_Level is


-- ***component list***

-- 01 adder
component penambah4bit
port(   A, B: in std_logic_vector(7 downto 0);
        Ci: in std_logic;
        S: out std_logic_vector(7 downto 0);
        CarryAux: out std_logic;
        Co: out std_logic);
end component;

-- 02 adder CLA
component CLA
port(   x_in    : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
        y_in    : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
        carry_in : IN  STD_LOGIC;
        sum     : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
        carry_out : OUT STD_LOGIC);
end component;
```

```
-- 03 subtracter CLA
component claminus
port(   x_in     : IN   STD_LOGIC_VECTOR(7 DOWNTO 0);
        y_in     : IN   STD_LOGIC_VECTOR(7 DOWNTO 0);
        carry_in : IN   STD_LOGIC;
        sum      : OUT  STD_LOGIC_VECTOR(7 DOWNTO 0);
        carry_out :  OUT  STD_LOGIC);
end component;

-- 04 subtracter
component penolak4bit
port(   A, B: in std_logic_vector(7 downto 0);
                Ci: in std_logic;
                S: out std_logic_vector(7 downto 0);
                Co: out std_logic);
end component;

-- 05 and
component dan
port(   X, Y : in std_logic_vector(7 downto 0);
                Sum : out std_logic_vector(7 downto 0));
end component;

-- 06 or
component atau
port(   X, Y : in std_logic_vector(7 downto 0);
                Sum : out std_logic_vector(7 downto 0));
end component;

-- 07 XOR
component exclusive
port    ( X, Y : in std_logic_vector(7 downto 0);
          Sum : out std_logic_vector(7 downto 0));
end component;

-- 08 Shifter left
component shifter_left
port(   clk, reset, load  : in std_logic;

        Data  : in std_logic_vector(7 downto 0);
                Q : out std_logic_vector(7 downto 0));
end component;

-- 09 Shifter right
component shifter_right
```

```
port(   clk, reset, load  : in std_logic;

        Data  : in std_logic_vector(7 downto 0);
                Q : out std_logic_vector(7 downto 0));
end component;

-- 10 Rotate Left
component rotator_left
port(   clk, reset, load  : in std_logic;

        Data  : in std_logic_vector(7 downto 0);
                Q : out std_logic_vector(7 downto 0));
end component;

-- 11 Rotate Right
component rotator_right
port(   clk, reset, load  : in std_logic;

        Data  : in std_logic_vector(7 downto 0);
                Q : out std_logic_vector(7 downto 0));
end component;

-- 12 Increment A
component     incA
port ( a : in std_logic_vector(7 downto 0);
        c : out std_logic_vector(7 downto 0));
end component;

-- 13 Increment B
component     incB
port ( b : in std_logic_vector(7 downto 0);
        c : out std_logic_vector(7 downto 0));
end component;

-- 14 Decrement A
component     decA
port ( a : in std_logic_vector(7 downto 0);
        c : out std_logic_vector(7 downto 0));
end component;

-- 15 Decrement B
component     decB
port ( B : in std_logic_vector(7 downto 0);
        c : out std_logic_vector(7 downto 0));
end component;
```

```
-- 16 multiplier
component    signed_mult
port(   a: IN STD_LOGIC_VECTOR (7 DOWNTO 0);
        b: IN STD_LOGIC_VECTOR (7 DOWNTO 0);
    --  Enn: in std_logic;
        sum: OUT STD_LOGIC_VECTOR (15 DOWNTO 0));
end component;

-- 17 multiplexer
component multiplexer
port(   sumcla: in std_logic_vector(7 downto 0);
        coutcla: in std_logic;
        sumrca: in std_logic_vector(7 downto 0);
        coutrca: in std_logic;
        sumclaminus: in std_logic_vector(7 downto 0);
        coutclaminus: in std_logic;
        sumsub: in std_logic_vector(7 downto 0);
        coutsub: in std_logic;
        sumor: in std_logic_vector(7 downto 0);
        sumand: in std_logic_vector(7 downto 0);
        sumex: in std_logic_vector(7 downto 0);
        sumshiftleft: in std_logic_vector(7 downto 0);
        sumshiftright: in std_logic_vector(7 downto 0);
        sumrotleft: in std_logic_vector(7 downto 0);
        sumrotright: in std_logic_vector(7 downto 0);
        sumincA: in std_logic_vector(7 downto 0);
        sumincB: in std_logic_vector(7 downto 0);
        sumdecA: in std_logic_vector(7 downto 0);
        sumdecB: in std_logic_vector(7 downto 0);
        sel: in std_logic_vector(3 downto 0);
        En: out std_logic;
        R: out std_logic_vector(7 downto 0);
        C: out std_logic);
end component;

-- 18 flag checker
component    bendera
port(realsum: in std_logic_vector(7 downto 0);
     Enab: in std_logic;
     carry: in std_logic;
     carryauxalary: in std_logic;
     zerof: out std_logic;
     signf: out std_logic;
     overf: out std_logic;
     parf: out std_logic;
     carryf: out std_logic;
```

```
        auxf: out std_logic);
end component;


-- ***component signals***
--01
signal rcasum : std_logic_vector(7 downto 0);
signal rcacout : std_logic;
signal cau : std_logic;

--02
signal clasum : std_logic_vector(7 downto 0);
signal clacout : std_logic;

--03
signal subsum : std_logic_vector(7 downto 0);
signal subcout : std_logic;

--04
signal claminussum : std_logic_vector(7 downto 0);
signal claminuscout : std_logic;

--05
signal andsum : std_logic_vector(7 downto 0);

--06
signal orsum : std_logic_vector(7 downto 0);

--07
signal exsum : std_logic_vector(7 downto 0);

--08
signal shiftsumleft : std_logic_vector(7 downto 0);

--09
signal shiftsumright : std_logic_vector(7 downto 0);

--10
signal rotsumleft : std_logic_vector(7 downto 0);

--11
signal rotsumright : std_logic_vector(7 downto 0);

--12
signal incsumA : std_logic_vector(7 downto 0);
```

```
--13
signal incsumB : std_logic_vector(7 downto 0);

--14
signal decsumA : std_logic_vector(7 downto 0);

--15
signal decsumB : std_logic_vector(7 downto 0);

--16
signal mulsum : std_logic_vector(15 downto 0);

--17
signal res: std_logic_vector(7 downto 0);
signal realcout : std_logic;
signal enable : std_logic;

--18
signal ze : std_logic;
signal si : std_logic;
signal ov : std_logic;
signal pa : std_logic;
signal ca : std_logic;
signal au : std_logic;


begin


-- ***component port mapping***

--01
PlusRCA : penambah4bit
          port
map(A=>Data_X,B=>Data_Y,Ci=>Carry_in,CarryAux=>cau,S=>rcasum,Co=>rcacout);

--02
PlusCLA : CLA
          port
map(x_in=>Data_X,y_in=>Data_Y,carry_in=>Carry_in,sum=>clasum,carry_out=>claco
ut);

--03
MinusCLA : claminus
```

```
        port
map(x_in=>Data_X,y_in=>Data_Y,carry_in=>Carry_in,sum=>claminussum,carry_out=
>claminuscout);

--04
Minus : penolak4bit
        port map(A=>Data_X,B=>Data_Y,Ci=>Carry_in,S=>subsum,Co=>subcout);

--05
And_op :  dan
        port map(X=>Data_X,Y=>Data_Y,sum=>andsum);

--06
Or_op :  atau
        port map(X=>Data_X,Y=>Data_Y,sum=>orsum);

--07
Ex : exclusive
        port map(X=>Data_X,Y=>Data_Y,sum=>exsum);

--08
ShiftL : shifter_left
        port map(Data=>Data_X,clk=>Clk,reset=>Rst,load=>Load,Q=>shiftsumleft);

--09
ShiftR : shifter_right
        port map(Data=>Data_X,clk=>Clk,reset=>Rst,load=>Load,Q=>shiftsumright);

--10
RotateL : rotator_left
        port map(Data=>Data_X,clk=>Clk,reset=>Rst,load=>Load,Q=>rotsumleft);

--11
RotateR : rotator_right
        port map(Data=>Data_X,clk=>Clk,reset=>Rst,load=>Load,Q=>rotsumright);

--12
IncrementA : incA
        port map(a => Data_X, c => incsumA);

--13
IncrementB : incB
        port map(b => Data_Y, c => incsumB);

--14
DecrementA : decA
```

```
        port map(a => Data_X, c => decsumA);

--15
DecrementB : decB
        port map(b => Data_Y, c => decsumB);

--16
Multiplier: signed_mult
        port map(a=>Data_X,b=>Data_Y,sum=>mulsum);

--17
Mul_general    : multiplexer
      port map(sumrca=>rcasum,coutrca=>rcacout,sumcla=>clasum,
               coutcla=>clacout,sumsub=>subsum,coutsub=>subcout,
                        sumclaminus=>claminussum,coutclaminus=>claminuscout,
                        sumand=>andsum,sumor=>orsum,sumex=>exsum,
                        sumshiftleft=>shiftsumleft,sumshiftright=>shiftsumright,
                        sumrotleft=>rotsumleft,sumrotright=>rotsumright,
                        sumincA=>incsumA,sumincB=>incsumB,sumdecA=>decsumA,
                        sumdecB=>decsumB,sel=>Op_code,R=>res,C=>realcout,
                        En=>enable);

--18
Flag_checker : bendera
        port map(realsum=>res,Enab=>enable,carry=>realcout,
                          carryauxalary=>cau,zerof=>ze,signf=>si,overf=>ov,
                          parf=>pa,carryf=>ca,auxf=>au);


-- top level output

Result <= res;
Result2 <= mulsum;
Carry_out <= realcout;
a<=au;
c<=ca;
o<=ov;
p<=pa;
s<=si;
z<=ze;
end Stuctural;
```