LAPORAN LATIHAN ILMIAH TAHUN AKHIR

Rekabentuk Perkakasan Khas ECC
(Reed-Solomon)

oleh

NOORIZAN MUHAMAD MURSID
WEK 98197

di bawah bimbingan

EN. ZAIDI RAZAK

Laporan Latihan Ilmiah ini diserahkan kepada

Fakulti Sains Komputer dan Teknologi Maklumat
Universiti Malaya
2000 / 2001

bagi memenuhi syarat penganugerahan
Ijazah Sarjana Muda Sains Komputer

ABSTRAK

Dokumen ini disiapkan bagi menyediakan laporan tentang perjalanan projek perkakasan khas ECC yang dijalankan. Bahagian pertama dokumen ini terdapat pengenalan tentang projek. Bahagian kedua pula menyatakan tentang kajian yang telah dilakukan bagi memilih kod-kod pembetulan yang ingin dibangunkan. Seterusnya pada bahagian tiga, dinyatakan serba sedikit tentang bahasa yang digunakan dalam pembangunan projek iaitu VHDL. Bahagian keempat pula menceritakan tentang senibina-senibina dan rekabentuk-rekabentuk perkakasan khas. Dalam bahagian lima, dokumen menceritakan tentang fasa pembangunan yang telah dijalankan. Bahagian enam dan tujuh pula menceritakan tentang perbincangan dan kesimpulan daripada projek yang telah dijalankan.

PENGHARGAAN

Syukur alhamdulillah di atas limpah kurnia-Nya saya dapat menyiapkan projek ini dengan jayanya walaupun pada peringkat awal banyak masalah yang saya hadapi.

Ucapan terima kasih saya buat penyelia Encik Zaidi Razak yang telah banyak memberi tunjuk ajar dan bantuan sepanjang saya membangunkan projek. Juga jutaan terima kasih buat tutor Encik Yamani Idna Idris yang tidak jemu membantu dan memberi bimbingan kepada saya dan rakan. Terima kasih juga buat Cik Nor Aniza sebagai moderator saya.

Buat keluarga yang saya sayangi, terima kasih kerana telah dengan sepenuh hati memberikan dorongan dan sokongan di atas usaha-usaha saya selama ini. Buat rakan-rakan seperjuangan saya, Terima kasih atas budi kalian semua teruama buat Teput, Deana, Ciko, Farah, Zai, Awin, Farazila dan juga buat Rul.

Sekalung budi buat sekalian yang terlibat dalam projek saya ini tidak kira sama ada secara langsung ataupun tidak langsung.

KANDUNGAN

SENARAI RAJAH

SENARAI JADUAL

1.1 PENGENALAN

Kod pembetulan ralat adalah kod yang berupaya untuk mengesan kedudukan ralat dalam jujukan data yang diterima dan akan membetulkan ralat tersebut berdasarkan kedudukan yang telah dikesan. Pembetulan ralat diperlukan bagi memastikan ketepatan dan keutuhan data malah dalam sesetengah keadaan ia digunakan bagi mencipta persekitaran toleransi silap dimana komponen yang gagal berfungsi tidak menyebabkan kehilangan data atau fungsi bagi komponen tersebut tidak terjejas.

Sehingga kini, didapati bahawa tiada sistem penghantaran atau penyimpanan data secara elektronik yang cukup sempurna. Setiap sistem yang ada pasti melakukan ralat pada kadar yang tertentu bergantung kepada faktor-faktor yang tertentu. Didapati bahawa kadar ralat adalah berkadar langsung dengan kadar penghantaran dan kadar keupayaan penyimpanan data.

Jika kita melihat kepada kadar ralat dalam konteks media, sestengah sistem elektronik mungkin mempunyai kadar ralat yang lebih tinggi daripada suatu sistem yang lain.

Sebagai contoh:

Cakera optik mempunyai kadar ralat yang lebih tinggi berbanding dengan cakera magnetik. Pita magnetik mempunyai kadar ralat yang lebih tinggi jika ingin dibandingkan pula dengan cakera magnetik manakala kabel komunikasi gentian optik dan ingatan semikonduktor mempunyai kadar ralat yang lebih rendah.

Salah satu cara untuk mengukur kadar ralat bit adalah dengan membahagikan jumlah bit ralat dengan jumlah bit data yang dihantar.

Sebagai contoh:

Dalam cakera magnetik kita akan lihat kepada data yang dihantar. Secara purata terdapat 1 bit ralat bagi setiap billion bit data yang dihantar. Jika kita lihat kepada suatu cakera optik, kadar ralatnya adalah lebih tinggi iaitu kira-kira 1 bit ralat bagi setiap penghantaran 100, 000 bit data.

Tanpa pengesanan dan pembetulan ralat, kebanyakan peranti storan akan hilang kebolehpercayaannya. Peranti yang kadar ralatnya terjamin akan lebih banyak digunakan bagi penyimpanan data-data sensitif untuk sesetengah aplikasi. Satu lagi cara untuk mengukur kadar bit ralat adalah dengan berdasarkan kepada bilangan ralat bagi unit masa yang tertentu.

Sebagai contoh:

Menggunakan cakera magnetik dengan kadar purata penghantaran kira-kira satu juta bit sesaat, ralat yang terhasil adalah sekitar satu bit ralat setiap seribu saat. Jika kita membuat penghantaran purata satu billion bit sesaat ralat yang berlaku adalah kira-kira satu bit ralat bagi setiap saat. Akhir-akhir ini, beberapa pemacu cakera boleh membuat penghantaran sekitar 40 milion bit sesaat dan dengan kadar 1 bit ralat setiap 25 saat.

Perkakasan khas yang ingin dibangunkan dalam projek ini harus berupaya untuk menjana Kod Pembetulan Ralat. Pembangunan perkakasan ini melibatkan logik digital. Data yang diterima ataupun data yang dihasilkan adalah dalam bentuk perwakilan binari iaitu sama ada "0" atau "1".

Penggunaan Kod Pembetulan Ralat dalam perkakasan dapat menghasilkan satu keadaan yang dapat menyokong peningkatan kadar penghantaran data dan ruang storan bagi peranti yang bersaiz kecil.

1.2 TUJUAN

Projek ini dijalankan untuk membuat analisa, merekabentuk dan membangunkan suatu perkakasan khas bagi menjalankan operasi Kod Pembetulan Ralat. Dalam kod tersebut bit-bit ulangan ditambahkan kepada jujukan data yang asal dengan cara supaya penerima data tersebut dapat mengesan ralat dan membetulkannya untuk memperolehi data yang sebenar. Sistem seperti ini amat penting bagi sesetengah aplikasi seperti penyimpanan data yang tidak boleh membuat penghantaran semula. Masalah yang perlu dikendalikan disini adalah bagaimana ingin menambahkan bit-bit ulangan tersebut supaya ia boleh mengesan dan membetulkan ralat semaksimum yang mungkin.

Perlaksanaan kod pembetulan ralat yang dicadangkan perlulah menyokong komunikasi data dengan kadar data yang tinggi. Ia mestilah suatu perlaksaan yang pantas tetapi

berkesan. Bahasa pengaturcaraan yang digunakan adalah VHDL dengan menggunakan peralatan XILINX.

1.3 OBJEKTIF PROJEK

Untuk menghasilkan perkakasan yang berupaya menjanakan Kod Pembetulan Ralat, perkakasan tersebut perlulah berupaya untuk melakukan 4 perkara asas dibawah:

- Menerima input

  Perkakasan boleh menerima maklumat dari luar dalam bentuk digital iaitu perwakilan binari "0" atau "1".

- Memproses maklumat

  Jujukan data yang diterima masuk akan diproses oleh perkakasan. Pemprosesan melibatkan penjanaan Kod Pembetulan Ralat. Ia terbahagi kepada 2 iaitu pengkodan dan penyahkodan data. Jujukan data akan di tambahkan atau dibuang bit-bit ulangan pada pengiraan yang tertentu.

- Menghasilkan output

  Daripada jujukan asal data yang diterima masuk kedalam perkakasan, suatu output sama ada jujukan data sebenar ataupun jujukan data yang telah dikodkan akan dihasilkan oleh perkakasan.

- Menyimpan maklumat

Sebagai suatu keperluan, perkakasan yang ingin dibangunkan perlu berupaya memegang data yang dimasukkan. Ia perlu mempunyai mekanisme untuk menyimpan, mencapai dan memulangkan hasil pemprosesan semula kepada tempat data disimpan.

1.4 SKOP PROJEK

Bagi menghadkan kekompleksan penyelesaian masalah, projek pembangunan perkakasan ECC ini akan hanya tertumpu kepada komunikasi setempat yang hanya melibatkan penghantaran dan penerimaan data di dalam sesebuah komputer dan juga peranti disekitarnya.

Projek akan dibangunkan hanya dalam skop selepas data diterima dalam bentuk digital. Ia tidak melibatkan penukaran data analog ke dalam bentuk digital ataupun bentuk digital ke analog. Jujukan data yang ingin dikodkan atau dinyahkodkan akan dianggap telah selamat diterima oleh penerima dalam bentuk digital iaitu perwakilan binari "0" dan "1".

1.5 KEPENTINGAN PROJEK

Dalam 20 tahun kebelakangan ini, kita telah lihat pembangunan pesat dalam industri perkomputeran. Kelajuan pemprosesan telah meningkat manakala tumpuan lebih kepada penghasilan komputer yang mempunyai saiz yang lebih kecil. Saiz aturcara komputer juga telah menjadi semakin besar dan lebih kompleks. Ini telah membawa kepada

keperluan untuk meningkatkan saiz ingatan RAM dan dengan ruang terhad pada papan litar utama kita juga memerlukan RAM yang lebih kecil. Dalam memenuhi kedua-dua keperluan tersebut, lebar talian yang digunakan dalam membina RAM telah menurun sehingga ke dua mikro. Walaubagaimanapun pengurangan lebar talian ini telah secara tidak langsung meningkatkan ralat rawak semasa pergerakan data.

Keadaan semulajadi dalam semua peranti elektronik adalah kadar penghantaran data akan semakin meningkat mengikut peredaran masa. Ini bermakna ralat akan berlaku dengan lebih kerap kerana pengeluar peranti tersebut ingin memuatkan lebih bit ke dalam ruang yang lebih kecil. Apabila kadar kelajuan dan kapasiti semakin meningkat, ini bermakna pembetulan ralat akan semakin diperlukan.

1.6 PERKAKASAN DAN PERISIAN

Projek dibangunkan menggunakan komputer peribadi. Platform yang digunakan untuk pembangunan adalah Windows 98 dan Windows 2000. Kapasiti muatan yang digunakan adalah sekitar 4.0 hingga 20.4 GB dengan pemprosesan Intel Pentium.

Perisian utama yang akan digunakan dalam pembangunan projek ini adalah Xilinx® dimana ia menggunakan bahasa pengaturcaraan Very high speed integrated circuit Hardware Description Language (VHDL). Xilinx akan digunakan dalam fasa pengkodan dan ujian. Kod-kod aturcara yang membentuk modul-modul untuk fungsi perkakasan adalah dibangunkan menggunakan VHDL di atas platform Xilinx.

Selain itu dalam proses pembangunan projek beberapa perisian sampingan yang lain akan turut digunakan. Senarai perisian dan aplikasinya yang digunakan adalah seperti berikut:

- Microsoft Word 2000

  Dokumentasi dan carta alir

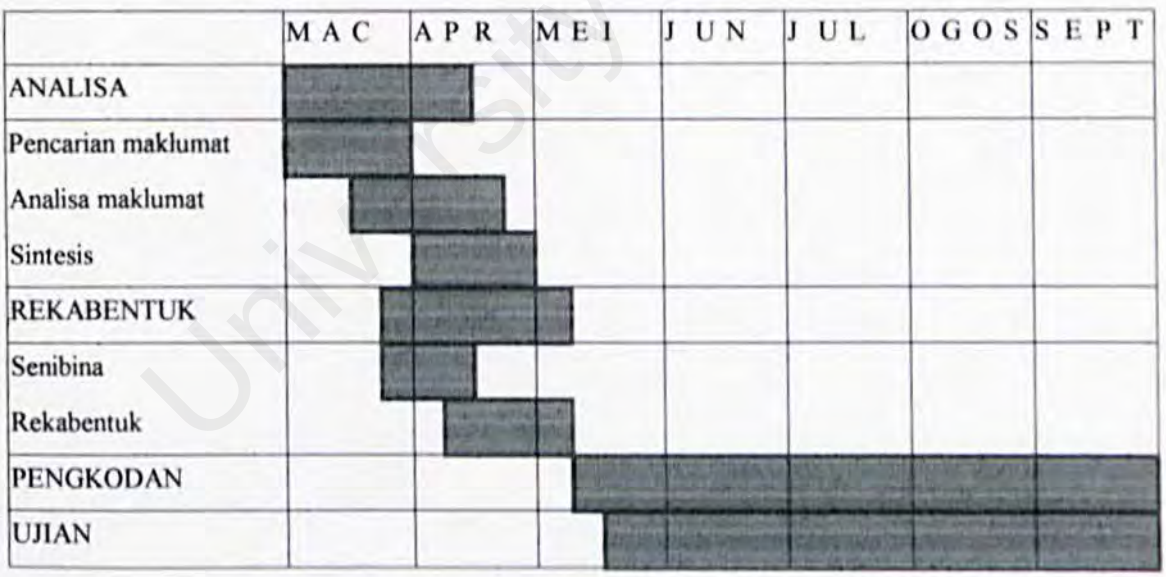
- Microsoft Excel

  Membina jadual dan carta Gantt

- Microsoft Power Point

  Kertas cadangan dan persembahan

1.7 PENJADUALAN PROJEK



Jadual 1.1 Jadual perancangan pembangunan projek

1.8 RINGKASAN

Kod Pembetulan Ralat adalah kod yang digunakan bagi mengesan kedudukan dan membetulkan ralat dalam jujukan data. Objektif perlaksanaan projek ini adalah untuk membina suatu perkakasan khas yang dapat menjanakan Kod Pembetulan Ralat. Bahasa yang digunakan adalah VHDL yang dilaksanakan menggunakan Xilinx. Kod pembetulan ralat merupakan kod yang digunakan dalam pengendalian ruang storan ataupun kadar komunikasi data supaya ralat pada data yang diterima ataupun yang ingin disimpan dapat dikesan dan dibetulkan. Penjadualan tugasan-tugasan dalam projek pembangunan ini melibatkan fasa-fasa analisa, rekabentuk, pengkodan dan ujian.

2.1 PENGENALAN

Dalam sistem elektronik digital, maklumat diwakili dalam bentuk kod-kod binari iaitu sama ada '0' atau '1'. Apabila maklumat binari dihantar dari satu sumber ke satu destinasi, akan sentiasa wujud kemungkinan untuk kesilapan berlaku di mana suatu kod data '1' di tafsirkan sebagai '0' atau sebaliknya kod data '0' sebagai '1'. Ini mungkin disebabkan oleh kesan media penghantaran, hingar elektronik, kegagalan komponen tertentu, sambungan yang lemah, pelusuhan yang disebabkan oleh usia dan banyak lagi faktor-faktor lain.

Apabila suatu bit disalahtafsirkan, diketahui bahawa suatu ralat bit telah berlaku. Mekanisme pembetulan ralat bukan sahaja mengesan ralat yang wujud malah membetulkan ralat yang terdapat dalam data yang dihantar. Terdapat dua kaedah yang boleh digunakan dalam pembetulan ralat iaitu:

- Apabila ralat dikesan, penerima boleh meminta supaya penghantar membuat penghantaran semula.
- Penerima boleh menggunakan kod pembetulan ralat yang akan secara automatik membetulkan ralat yang dikesan.

Secara teori, adalah tidak mustahil untuk membetulkan sebarang ralat kod binari secara automatik kerana ia hanya melibatkan bit "1" dan "0" sahaja. Walaubagaimanapun, penggunaan kod pembetulan ralat adalah rumit dan memerlukan banyak bit ulangan. Bilangan bit yang diperlukan bagi membetulkan ralat berbilang bit ataupun ralat letusan

adalah sangat tinggi sehinggakan pada sesetengah keadaan adalah tidak efisien untuk melaksanakannya. Oleh sebab itu, kod pembetulan ralat kebanyakkan kod ini terhad hanya kepada ralat-ralat satu, dua atau tiga bit.

Poses pembetulan ralat ini boleh dilakukan sama ada dengan menggunakan perisian ataupun perkakasan bergantung kepada kelajuan yang dikehendaki. Bagi kadar data yang tinggi, pembetulan ralat mesti dilakukan menggunakan perkakasan khas kerana penggunaan perisian adalah lebih perlahan.

Dalam kebanyakkan kes seperti pemacu cakera magnetic atau ingatan semikonduktor, ralat mesti dikesan dan dibetulkan pada kadar sama seperti kadar pembacaan data dalam cakera. Ini biasanya memerlukan perlaksaan pembetulan ralat dilakukan dalam logic digital iaitu perkakasan. Dalam sesetengah kes seperti beberapa sistem telekomunikasi di mana kadar data adalah lebih rendah, perlaksaan perisian bagi pembetulan ralat adalah sudah mencukupi.

2.2 SEJARAH RINGKAS

Sekitar 1947-1948, perkara-perkara dalam teori maklumat telah dicipta oleh Claude Shannon. Teori Matematik bagi Komunikasi oleh Shannon adalah satu-satunya cara untuk mendapatkan lebih kapasiti storan dalam sesuatu peranti storan ataupun bagi memperolehi penghantaran terpantas menerusi suatu saluran komunikasi dengan menggunakan sistem pembetulan ralat yang hebat.

Pada sekitar masa yang sama, Richard Hamming telah berjaya menemui dan melaksanakan suatu kod pembetulan ralat bit tunggal. Pada tahun 1960, penyelidik, termasuk Irving Reed dan Gustave Solomon, telah menemui cara untuk membangunkan kod pembetulan ralat yang boleh membuat pembetulan untuk suatu bilangan rawak bit-bit atau bait-bait di mana bait bermaksud suatu kumpulan yang terdiri daripada w bit. Walaubagaimanapun, pada masa tersebut cara untuk menyahkodkan kod ini masih belum diketahui.

Buku panduan pertama tentang kod pembetulan ralat ditulis pada 1961 oleh W. Wesley Peterson. Pada tahun 1968, Elwyn Berlekamp dan James Massey telah menemui algoritma yang diperlukan untuk membina pengkod bagi menghasilkan kod pembetulan ralat berbilang. Algoritma tersebut kemudiannya dikenali sebagai algoritma Berlekamp-Massey untuk menyelesaikan persamaan nyahkod kunci.

Dalam 30 tahun kebelakangan ini, para penyelidik telah menemui bahawa algoritma Berlekamp-Massey adalah suatu variasi daripada satu algoritma purba yang dikenalpasti di Egypt sekitar 300 tahun sebelum Maihi. Algoritma tersebut dihasilkan oleh seseorang yang dikenali sebagai Euclid. Ia dipanggil sebagai algoritma terbitan Euclid yang digunakan bagi mencari pembahagi sepunya terbesar bagi 2 polimonial.

Pada hari ini, pelbagai variasi daripada algoritma-algoritma Berlekamp-Massey dan Euclid wujud bagi menyelesaikan persamaan penyahkodan kunci.

2.3 KONSEP ASAS SISTEM PEMBETULAN RALAT

Secara asasnya, setiap suatu kumpulan bit-bit di dalam suatu komputer dirujuk sebagai "word". Setiap bit adalah salah satu daripada 2 karakter yang wujud iaitu sama ada "1" ataupun "0". Dilihat bahawa:

- 8 bit = 1 bait
- 4 bait = 1 "word"
- bit = "1" atau "0"

Sistem pembetulan ralat menambahkan bit-bit tambahan kepada data komputer. Bit-bit penambahan ini dikenali sebagai bit ulangan. Bit ulangan ini menambahkan struktur tertentu kepada setiap data. Jika struktur tersebut diubahsuai oleh ralat, perubahan tersebut boleh dikenalpasti dan dibetulkan.

Bagi tujuan pemahaman, kita boleh melihat kepada perkataan-perkataan dalam bahasa Inggeris. Dalam Bahasa Inggeris, terdapat satu analogi di mana dinyatakan bahawa tidak semua kombinasi-kombinasi huruf-huruf dapat membentuk perkataan yang yang memberi makna. Perkataan yang sebenar adalah perkataan-perkataan yang tersenarai di " dalam kamus bahasa.

Ralat yang terjadi semasa penghantaran atau penyimpanan perkataan Inggeris itu tadi boleh dikenalpasti dengan melihat sama ada perkataan yang diterima terdapat dalam kamus ataupun sebaliknya. Jika perkataan tersebut tiada dalam kamus, ralat dibetulkan

dengan mengambil perkataan daripada kamusl yang paling hampir dengan perkataan yang salah itu tadi. Sistem pembetulan ralat beroperasi mengikut kaedah yang lebih kurang sama. Dalam membuat pembetulan, oleh kerana data yang ada dalam bentuk perwakilan binari, maka ralat yang dikesan akan ditukar kepada perwakilan binari yang sebaliknya.

Pengkod dan penyahkod kod pembetulan ralat beroperasi dalam bentuk bit ataupun bait. Dalam bentuk matematik, setiap bit atau bait adalah suatu elemen dalam suatu ruang terhad yang merupakan suatu sistem algebra abstrak dengan suatu bilangan elemen terhad bersama takrifan dua operasi dan operasi sebaliknya. Operasi-operasi tersebut biasanya adalah operasi pendaraban dan penambahan manakala operasi sebaliknya adalah operasi pembahagian dan penolakkan.

Ruang terhad juga dipanggil sebagai ruang Galois (disebut Gal Wah) sempena nama ahli matematik Peranchis yang menemuinya iaitu Evariste Galois (1811-1832). Teori ruang terhad adalah cabang besar dalam bidang matematik. Ia agak mudah dilaksanakan dalam litar logik digital atau dalam aturcara komputer.

2.4 PEMBETULAN RALAT BIT TUNGGAL

Ralat bit tunggal boleh dikesan dengan penambahan suatu bit ulangan (bit pariti) kepada unit data (VRC). Suatu bit penambahan tunggal boleh mengesan ralat bit tunggal pada mana-mana jujukan bit kerana ia perlu menimbangkan hanya dua keadaan iaitu ada ralat

ataupun tiada ralat. Satu bit mempunyai dua keadaan iaitu 0 atau 1. Kedua-dua keadaan ini adalah memadai pada tahap pengesanan ini.

Bagi pembetulan ralat disamping pengesanannya, dua keadaan di atas adalah tidak mencukupi. Suatu ralat wujud apabila penerima membaca bit 1 sebagai 0 ataupun sebaliknya. Untuk membetulkan ralat sebegini, penerima hanya perlu menterbalikkan nilai bit songsang tersebut. Walaubagaimanapun, untuk membuat pembetulan tersebut, penerima perlu mengenalpasti kedudukan bit ralat tersebut. Oleh itu, kunci kepada pembetulan ralat adalah lebih kepada mengesan kedudukan bit atau bit-bit yang ralat dan bukan kepada membetulkan ralat songsang tersebut.

Sebagai contoh:

Pengesanan ralat dalam suatu karakter kod ASCII. Kos pengesanan ralat perlu mengambilkira 8 kebarangkalian berikut:

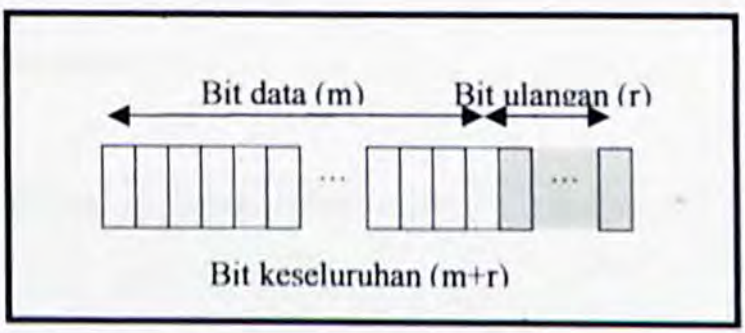
- Tiada ralat dalam jujukan 7 bit tersebut
- Ralat pada bit pertama
- Ralat pada bit kedua
- Ralat pada bit ketiga
- Ralat pada bit keempat
- Ralat pada bit kelima
- Ralat pada bit keenam
- Ralat pada bit ketujuh

Untuk itu, ia memerlukan banyak bit ulangan yang dapat menunjukan kesemua lapan keadaan tersebut. Dalam keadaan sebegini, dapat diperhatikan bahawa jujukan 3 bit dari 000 hingga 111 dapat menunjukkan kedudukan ralat bagi kelapan-lapan kebarangkalian tersebut.

Namun kita juga perlu memikirkan sama ada bit ulangan yang diberikan mengandungi ralat atau tidak. Penggunaan 3 bit ulangan ini juga terhad hanya untuk kod ASCII yang terdiri daripada 7 bit. Bilangan bit ulangan yang lebih banyak diperlukan untuk mengenalpasti ralat bagi bilangan jujukan data yang lebih banyak daripada itu untuk mewakili kesemua kebarangkalian kedudukan ralat yang mungkin.

2.4.1 BIT ULANGAN

Bagi menentukan bilangan bit ulangan (r) yang diperlukan dalam membetulkan ralat bagi jujukan bit data yang dihantar (m), kita perlu mengenalpasti hubungan antara m dan r tersebut. **Rajah 2.1** menunjukkan data m bit dengan r bit ulangan ditambahkan padanya.



Rajah 2.1 Bit data dan bit ulangan

Panjang keseluruhan bagi kod ini adalah m+r. Sekiranya panjang unit data yang dihantar adalah m+r, penerima perlu menentukan m+r+1 kedudukan ralat yang mungkin bagi data tersebut. Jadi, kesemua keadaan ini perlu dinyatakan didalam bit r di mana bit r ini boleh menyatakan 2^r keadaan yang berlainan bagi setiap kedudukan ralat yang mungkin. Untuk itu, nilai 2^r perlulah lebih besar atau sama dengan m+r+1. Ini boleh ditunjukkan seperti dalam persamaan berikut:

$$\mathbf{2^r > m+r+1}$$

Sebagai contoh:

Jika nilai m adalah 7, nilai r terkecil yang memenuhi persamaan hubungan m dan r adalah:

$$\mathbf{2^4 > 7+4+1}$$

dimana nilai r adalah 4.

Jadual 2.1 menunjukkan beberapa nilai m yang mungkin dan nilai r yang bersepadanan dengannya.

Bilangan bit data (m)	Bilangan bit ulangan (r)	Jumlah bit (m+r)
1	2	3
2	3	5

3	3	6
4	3	7
5	4	9
6	4	10
7	4	11

Jadual 2.1 Hubungan antara bit data dan bit ulangan

2.5 PEMBETULAN RALAT BIT BERBILANG DAN RALAT LETUSAN

Bit-bit ulangan yang dikira untuk bit data boleh juga digunakan untuk membetulkan ralat bit berbilang. Walaubagaimanapun, bilangan bit ulangan yang diperlukan untuk pembetulan ralat bit berbilang adalah jauh lebih banyak berbanding bit ulangan yang diperlukan untuk pembetulan ralat bit tunggal.

Sebagai contoh:

Untuk membuat pembetulan 2 bit ralat, kita perlu mengambil perhatian bahawa mana-mana 2 kombinasi bit pada bit data adalah ralat. Untuk membuat pembetulan 3 bit ralat, kita perlu mengambil perhatian bahawa mana-mana 3 kombinasi bit pada bit data adalah ralat dan seterusnya.

Untuk membetulkan ralat bit berbilang, konsep asas dalam pembetulan ralat bit tunggal perlu diubahsuai supaya lebih efisien dan berkesan.

Ini sama seperti kaedah membetulkan ralat letusan kecuali ralat letusan berada secara berturutan dalam suatu jujukan data. Jadi apabila kedudukan bit ralat pertama dalam data telah diketahui, langkah seterusnya adalah bagi mengenalpasti jumlah bit yang ralat supaya pembetulan bagi bit-bit ralat tersebut boleh dilakukan.

2.6 APLIKASI PEMBETULAN RALAT

Pembetulan ralat pada masa kini digunakan untuk membuat peranti dengan kapasiti storan yang tinggi seperti cakera magnetik, cakera optik dan pita lebih dipercayai untuk digunakan. Pembetulan ralat membolehkan rakaman berkapasiti lebih tinggi dilakukan pada peranti storan tersebut.

Sebagai contoh:

Dalam cakera magnetik, pemacu yang lebih murah dan boleh percaya dengan kapasiti sekitar 2 hingga 20 gigabait hanya boleh dicapai dengan mengaplikasikan mekanisme pembetulan ralat yang tertentu.

Pembetulan ralat boleh digunakan untuk mengurangkan kos pengeluaran yang seterusnya mengurangkan harga peranti di samping meningkatkan kadar kebolehpercayaan modul ingatan DRAM (SIMMS) kerana komponen DRAM gred rendah boleh digunakan.

Pembetulan ralat juga boleh digunakan bagi melanjutkan usia bateri bagi komputer mudah alih dan lain-lain peranti elektronik yang menggunakan kuasa bateri dengan membenarkan pengurangan drastik pada kadar **refresh** dalam DRAM.

Dalam sistem rangkaian komunikasi seperti *Asynchronous Transfer Mode* (ATM), pembetulan ralat digunakan untuk membangunkan semula paket data yang hilang tanpa memerlukan penghantaran semula dilakukan oleh penghantar. Ini seterusnya boleh meningkatkan persembahan bagi sistem penghantaran data tersebut kerana bagi sistem rangkaian jarak jauh, penghantaran semula data yang kerap boleh menyebabkan persembahan rangkaian menurun.

Dalam Tatasusunan Ulangan bagi Peranti Murah (RAID), pembetulan ralat boleh digunakan bagi mencipta suatu toleransi silap pada kadar tertentu yang boleh diterima. Peranti berbilang boleh gagal tanpa memberi kesan kepada kehilangan data atau persembahan. Pembetulan ralat juga boleh digunakan untuk menyediakan sistem pengkabelan toleransi silap.

2.7 JENIS-JENIS KOD PEMBETULAN RALAT

Secara amnya, kod pembetulan ralat terbahagi kepada dua jenis iaitu:

- kod pepohon
- kod blok

Dalam setiap kes di atas, rentetan bit-bit akan dibahagikan kepada blok-blok data dan setiap blok data akan dikodkan kepada satu blok data yang lebih panjang. Kod pepohon beroperasi pada blok data dengan saiz yang lebih kecil berbanding dengan kod blok.

Tidak seperti kod blok, pengkodan setiap blok data dalam kod pepohon bergantung kepada status pengkod dan juga data yang ingin dikodkan. Penyahkodan pula dilakukan dengan melaksanakan beberapa jenis algoritma di dalam sesuatu pemproses.

Dengan menggunakan kod blok, setiap blok data yang ingin dikodkan adalah lebih panjang dan setiap blok yang ingin dikod dan dinyahkodkan adalah tidak bergantung kepada blok-blok data yang lain. Kod blok boleh diklasifikasikan dalam pelbagai cara.

Sebagai contoh:
Kita boleh meletakkan semua kod yang beroperasi dalam bit kepada satu kategori kod binari dan kod-kod lain kepada kategori kod bukan binari.

Terdapat juga ciri-ciri lain pada kod yang boleh digunakan untuk pengkelasan. Antaranya ialah sama ada sesuatu kod itu berkitar, diringkaskan, beralgebra, bersistematik, sempurna dan sebagainya. Antara kod-kod yang popular adalah kod Hamming, kod Ulangan dan kod Reed-Solomon. Selain itu ada juga kod-kod lain yang diaplikasi pada sistem-sistem yang terdahulu seperti kod Reed-Muller dan kod Golay.

2.7.1 KOD HAMMING

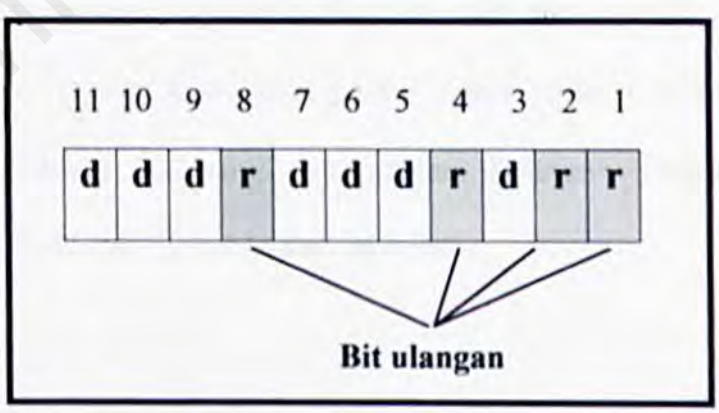
Suatu teknik untuk menentukan kedudukan mana yang terdapat ralat dengan memanipulasikan bit-bit ulangan telah dibangunkan oleh R.W. Hamming. Teknik ini yang kemudiannya dikenali sebagai Kod Hamming.

MENEMPATKAN BIT ULANGAN

Kod Hammning boleh digunakan pada apa-apa panjang data dengan menggunakan hubungan bit data dan bit ulangan.

Sebagai contoh:

Suatu kod ASCII 7 bit memerlukan 4 bit ulangan yang boleh ditambahkan pada hujung data yang ingin dihantar ataupun diantara data-data tersebut. Dalam **rajah 2.2,** bit-bit ini diletakkan pada kedudukan kuasa 2 iaitu 2^1, 2^2, 2^4 dan 2^8. Bit-bit ulangan ini dirujuk sebagai r_1, r_2, r_4 dan r_8.



Rajah 2.2 Penempatan bit ulangan dalam kod Hamming

Dalam kod Hamming, setiap bit r adalah bit VRC untuk satu kombinasi bit data. Kombinasi yang digunakan untuk mengira setiap nilai r dalam jujukan 7 bit data m adalah seperti berikut:

r_1 : bit 1,3,5,7,9,11

r_2 : bit 2,3,6,7,10,11

r_4 : bit 4,5,6,7

r_8 : bit 8,9,10,11

Setiap bit data berkemungkinan termasuk dalam lebih daripada satu pengiraan VRC. Dalam jujukan di atas, setiap bit data telah sekurang-kurangnya dimasukkan dalam dua set pengiraan manakala bit r hanya dimasukkan dalam satu sahaja.

Strategi pemilihan bit data untuk pengiraan VRC adalah:

Dengan melihat kepada persembahan binari bagi kedudukan setiap bit. Nilai r_1 dikira menggunakan semua kedudukan bit yang persembahan binarinya memasukkan 1 pada kedudukan paling kanan. Nilai r_2 dikira menggunakan semua kedudukan bit yang persembahan binarinya memasukkan 1 pada kedudukan kedua paling kanan dan seterusnya.

MENGIRA NILAI r

Rajah 2.3 menunjukkan suatu perlaksanaan kod Hamming untuk suatu karakter kod ASCII.

Langkah pertama:

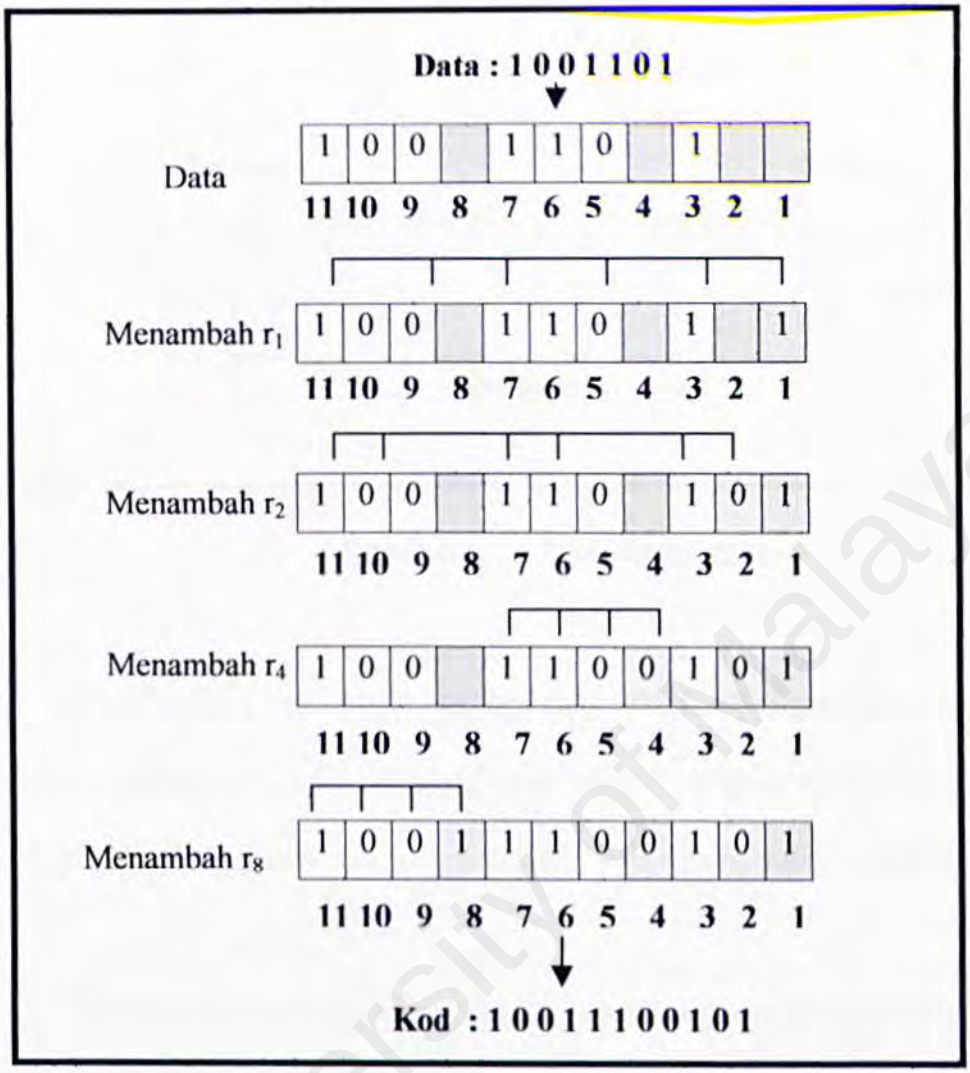
Setiap bit asal data akan diperuntukkan kedudukan yang berpadanan dalam jujukan 11 unit bit data yang akan dihantar.

Langkah kedua:

Mengira pariti genap untuk kombinasi bit yang berlainan. Nilai pariti bagi setiap kombinasi adalah nilai r yang sepadan.

Sebagai contoh:

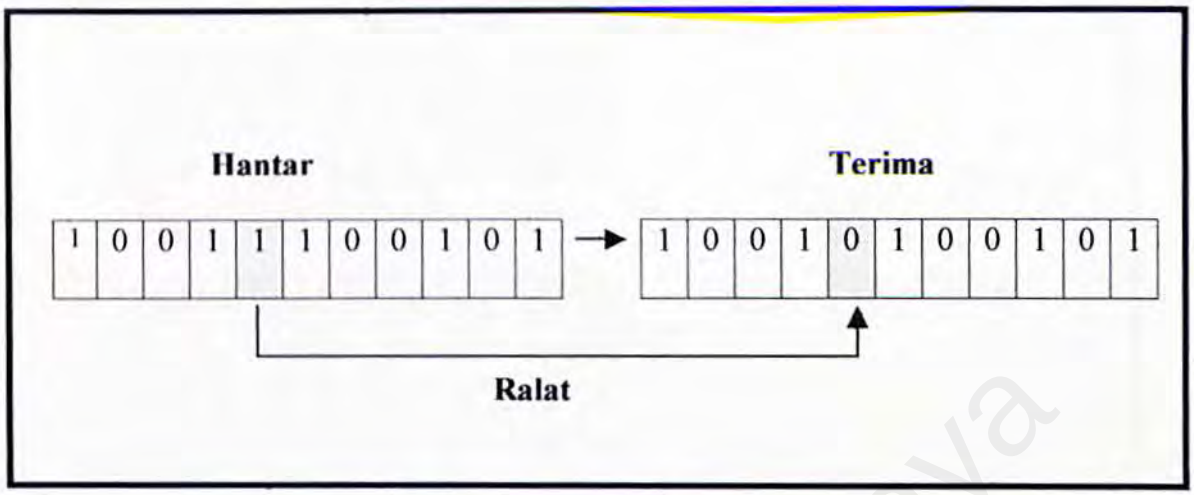
Nilai r_1 dikira untuk menyediakan pariti genap bagi kombinasi bit-bit 3, 5, 7, 9 dan 11 manakala nilai r_2 dikira untuk menyediakan pariti genap bagi kombinasi bit-bit 3, 6, 7, 10 dan 11 dan seterusnya. Kod 11 bit akhir akan seterusnya dihantar menerusi talian penghantaran.



Rajah 2.3 Contoh Pengiraan bit ulangan

PENGESANAN DAN PEMBETULAN RALAT

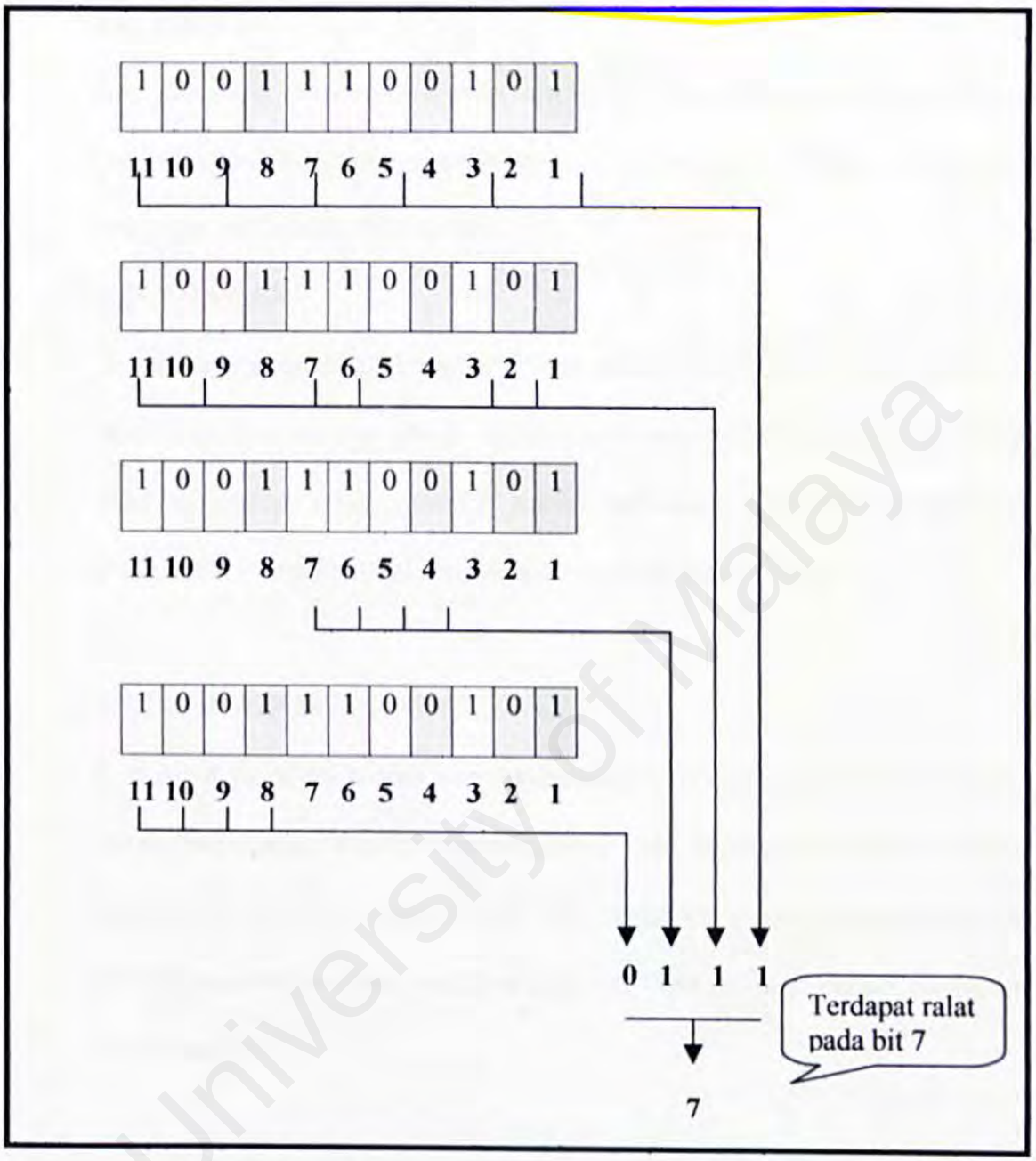
Seperti dalam **rajah 2.4**, pada masa penghantaran dibuat, bit ke 7 telah tertukar daripada binari "1" ke "0".



Rajah 2.4 Ralat bit tunggal

Penerima data akan menerima data yang dihantar dan mengira semula nilai VRC menggunakan set-set bit yang sama seperti penghantar dan bit pariti (r) yang berkaitan dengan setiap set. Ini dapat dilihat seperti dalam **rajah 2.5**.

Penerima akan menggabungkan nilai pariti yang baru kepada satu nombor binari mengikit turutan kedudukan r iaitu r_8 , r_4 , r_2 dan r_1. Dalam contoh yang telah diberikan terlebih dahulu, didapati bahawa nombor binari yang diperolehi adalah 0111 iaitu nilai 7 dalam decimal. Apabila bit yang ralat telah dikenalpasti, penerima boleh dengan mudah menterbalikkan nilai tersebut dan ralat berjaya dibetulkan bagi memperoleh data yang sebenar.



Rajah 2.5 Pengesanan ralat menggunakan kod Hamming

KELEBIHAN

Kod Hamming telah menyediakan satu kaedah penyelesaian yang praktikal. Kod Hamming boleh digunakan pada apa-apa panjang data dengan menggunakan hubungan bit data dan bit ulangan.

KELEMAHAN

Kod Hamming lebih praktikal sekiranya ralat dalam jujukan data adalah kecil iaitu 1 bit ralat sahaja. Untuk jumlah ralat yang lebih banyak, bit-bit ulangan yang diperlukan untuk mewakili semua kedudukan ralat yang mungkin akan lebih besar. Ini secara tidak langsung menambahkan *overhead.*

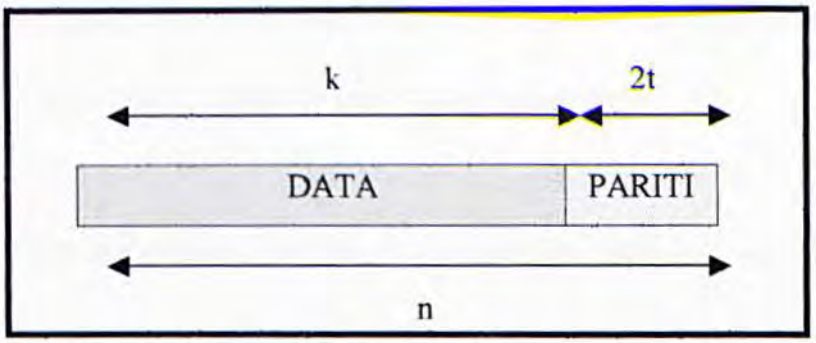
2.7.2 KOD REED-SOLOMON

Kod Reed Solomon adalah kod pembetulan ralat dari jenis kod blok linear. Ia dinyatakan sebagai RS(n,k) dengan simbol s bit. Pernyataan tersebut bermakna, pengkod mengambil k simbol data bagi setiap bit s dan menambahkan pariti simbol untuk menjanakan kata kod simbol n. Oleh itu, terdapat n-k simbol pariti bagi setiap bit s.

Suatu penyahkod Reed-Solomon boleh membetulkan sehingga t symbol yang mengandungi ralat dalam satu-satu kata kod di mana

$$2t = n-k$$

Rajah 2.7 menunjukkan struktur asas bagi suatu kod Reed-Solomon.



Rajah 2.7 Struktur asas kata kod Reed Solomon

Kod Reed-Solomon dengan struktur seperti di atas dikenali sebagai kod sistematik kerana data asal tidak mengalami perubahan sementara pariti simbol menokok.

CONTOH PERLAKSANAAN

Sebagai contoh diambil suatu pernyataan kod Reed-Solomon RS(255,233) dengan simbol 8 bit. Ini bermakna kata kod tersebut mengandungi 255 bait kata kod di mana 233 bait adalah data dan 32 adalah pariti. Untuk kod ini:

$$\mathbf{n = 255}$$

$$\mathbf{k = 223}$$

$$\mathbf{s = 8}$$

dan

$$\mathbf{2t = 32}$$

$$\mathbf{t = 16}$$

Ini bermakna penyahkod boleh membetulkan sebarang 16 ralat simbol dalam kata kod tersebut. Pembetulan ralat sehingga 16 bait pada mana-mana lokasi dalam kata kod tersebut akan dilakukan secara automatik.

Dengan diberi saiz simbol s, panjang maksimum bagi kata kod (n) adalah:

$$\mathbf{n = 2s - 1}$$

Sebagai contoh:

Panjang maksimum bagi kod dengan simbol 8 bit (s = 8) adalah 255 bait. Kod Reed-Solomon boleh diringkaskan dengan meletakkan sifar bilangan symbol data pada pengkod, menghantar data tersebut dan memuatkannya di dalam penyahkod.

Sebagai contoh:

Kod RS(255,223) di atas boleh diringkaskan kepada RS(200,168) jika pengkod mengambil satu blok 168 bait data dan menambahkan 55 bait sifar untuk mencipta suatu kata kod RS(255,223). Ia kemudiannya menghantar hanya 168 bait data dan 32 bait pariti.

Kuasa pemprosesan yang diperlukan untuk mengkod atau menyahkodkan kod Reed-Solomon adalah berkait dengan bilangan simbol pariti bagi setiap kata kod.

Jika nilai t adalah besar, ini bermakna lebih banyak ralat yang boleh dibetulkan tetapi memerlukan lebih kuasa pemprosesan berbanding dengan nilai kecil t.

RALAT SIMBOL

Ralat simbol terjadi apabila 1 bit dalam suatu simbol silap ataupun apabila kesemua bit dalam suatu simbol adalah salah.

Sebagai contoh:

RS(255,223) boleh membuat pembetulan 16 ralat simbol. Dalam kes yang paling buruk, mungkin ada 16 bit ralat yang terjadi dimana setiapnya dalam suatu simbol berasingan supaya penyahkod membetulkan 16 bit ralat. Dalam kes yang paling baik, ralat 16 keseluruhan bait terjadi supaya penyahkod membetulkan ralat 16 x 8 bit.

Kod Reed-Solomon khusus sesuai untuk membetulkan ralat letusan dimana suatu siri bit dalam kata kod yang diterima mengalami ralat.

PENYAHKODAN

Prosedur penyahkodan algebra Reed-Solomon berupaya membetulkan ralat dan pemadaman. Suatu pemadaman berlaku apabila kedudukan ralat telah diketahui. Suatu penyahkodan boleh membetulkan sehingga t ralat ataupun 2t pemadaman.

Apabila suatu kata kod dinyahkodan, terdapat 3 kemungkinan yang mungkin berlaku iaitu:

- Sekiranya $2s + r < 2t$ iaitu s ralat dan r pemadaman, maka kata kod yang asal akan berjaya ditemui.
- Jika sebaliknya, penyahkodan akan mengesan yang mesej asal gagal ditemui dan menyatakan keadaan tersebut.
- Ataupun, penyahkod akan mengabaikan pengkodan dan menemui kata kod yang salah tanpa sebarang pemberitahuan kepada penerima.

Kebarangkalian bagi setiap ketiga-tiga kemungkinan di atas bergantung kod Reed-Solomon yang tertentu dan juga kepada bilangan pengagihan ralat.

KELEBIHAN

Kod Reed-Solomon adalah kod yang meluas perlaksaannya kerana dianggap sebagai kod yang sempurna dalam konteks bit ulangannya. Dalam kod ini, bit ulangan yang ditambahkan oleh pengkod adalah minimum pada mana-mana paras dalam proses pembetulan ralat. Oleh itu, tiada bit yang dibazirkan.

Kod Reed-Solomon adalah lebih mudah untuk dinyahkodkan berbanding dengan kebanyakan kod-kod bukan binari yang lain. Ia membenarkan pembetulan pemadaman. Pembetulan pemadaman boleh dilihat seolah-olah memadamkan suatu huruf daripada satu perkataan dengan menggunakan pemadam pensel.

Pemadaman huruf tersebut dilakukan tanpa perlu menjejaskan huruf-huruf lain dalam perkataan tersebut. Huruf yang berada di kedudukan pemadaman adalah tidak diketahui tetapi kedudukan pemadaman perlu dipastikan.

Kod Reed-Solomon boleh membetulkan sehingga dua kali ganda pemadaman mengikut jumlah ralat apabila pengkod tidak mempunyai apa-apa maklumat tentang ralat tersebut. Dengan menggunakan kod Reed-Solomon, kita boleh mempunyai lebih kuasa untuk membuat pembetulan bagi setiap usaha dan kos yang dikeluarkan kerana kita berupaya untuk membetulkan ralat pelbagai pada kedudukanrawak.

2.7.3 KOD ULANGAN

Konsep asal bagi kod ulangan adalah dengan mengulang setiap bit dalam jujukan bit data mengikut bilangan yang tertentu.

CONTOH PERLAKSANAAN

Jadual 2.2 menunjukkan keadaan bit sekiranya dihantar menggunakan kod ulangan 3 kali iaitu dipanggil sebagai kod R3.

Jujukan Sumber (s)	Jujukan yang dihantar (t)
0	000
1	111

Jadual 2.2 Kod Ulangan R3

Dengan menggunakan kod ulangan R3, jika kita menghantar mesej sumber menerusi suatu saluran simetri binari dengan paras hingar 0.1, kita boleh menerangkan saluran tersebut sebagai menambah vektor hingar n kepada vektor yang ingin dihantar.

Sebagai contoh:

Dalam modulo 2, di mana 1 % 1 = 0.

Jadual 2.3 menunjukkan perwakilan-perwakilan bagi s, t, n dan r dimana r adalah merupakan vector yang akan diterima.

S	0	0	1	0	1	1	0
T	000	000	111	000	111	111	000
N	000	001	000	000	101	000	000
R	000	001	111	000	010	111	000

Jadual 2.3 Jujukan data dan simbol untuk R3

MENYAHKOD VEKTOR YANG DITERIMA

Apabila kata kod telah diterima pada sebelah penerima, penyahkod akan melihat kepada jujukan data iaitu vektor yang dihantar dalam bentuk 3 bit sekali pada satu-satu masa. Penyahkod akan mengambil kira perwakilan bit majoriti dalam ketiga-tiga bit tersebut. Ia akan membuat penterjemahan berdasarkan penilaiannya ke atas ketiga-tiga bit pada satu-satu masa untuk mendapatkan jujukan data yang asal. **Jadual 2.4** menunjukkan algoritma yang diguna untuk menterjemah jujukan data yang diterima apabila kod R3 digunakan.

Jujukan yang diterima (r)	Jujukan yang telah dinyahkod (s)
000	000
001	000
010	000
011	111
100	100
101	111
110	111
111	111

Jadual 2.4 Algoritma penyahkodan untuk R3

KELEBIHAN

Apabila kita menggunakan kod ulangan untuk membuat pembetulan ralat semasa penghantaran data, kita akan melihat bahawa kebarangkalian ralat menurun

sehingga kira-kira 3%. Ini bermakna bahawa kadar ralat telah menurun kepada 1/3 sahaja.

KELEMAHAN

Walaubagaimanapun, dalam kod ulangan tidak semua ralat yang wujud dapat dibetulkan. Sekiranya terdapat 2 ralat pada satu-satu blok, ini akan menyebabkan penyahkod memperoleh data yang silap. Seperti dalam **jadual 2.5**, sekiranya ralat berlaku pada hanya satu bit pada satu-satu blok, ralat tersebut akan dapat dibetulkan. Ini dapat dilihat pada blok kedua yang mengalami hanya satu bit ralat. Tetapi sekiranya suatu blok mempunyai dua bit ralat, penyahkod akan melakukan kesilapan pada penterjemahan data. Ini dapat dilihat pada blok kelima yang mengalami dua bit ralat.

s	0	0	1	0	1	1	0
t	000	000	111	000	111	111	000
n	000	001	000	000	101	000	000
r	000	001	111	000	010	111	000
s	0	0	1	0	0	1	0

Jadual 2.5 Data yang telah dikodkan dalam R3

2.8 KOD PEMBETULAN RALAT YANG INGIN DIJANA

Selepas meneliti dan menganalisan maklumat yang ada, satu keputusan telah dibuat untuk menjanakan kod pembetulan ralat berjenis blok. Ini adalah kerana ia dapat

memproses blok data dengan saiz yang lebih besar daripada kod pepohon di samping pengkodan tidak dipengaruhi oleh status penyahkod dan jujukan data yang ingin dikodkan.

Oleh itu, daripada beberapa jenis kod pembetulan ralat yang telah dinyatakan di atas didapati bahawa kod Reed-Solomon sesuai digunakan setelah mempertimbangkan kelebihan ciri-ciri kod tersebut seperti yang telah dibincangkan.

Kod ini banyak digunakan bagi membuat pembetulan ralat dalam pelbagai sistem. Beberapa aplikasi kod ini adalah termasuk:

- Peranti storan termasuk pita, cakera padat, DVD dan kod bar
- Komunikasi tanpa wayar atau komunikasi bergerak termasuk telefon bimbit dan sambungan gelombang mikro
- Komunikasi satelit
- Televisyen digital atau DVB
- Modem berkuasa tinggi seperti ADSL dan xDSL

2.9 RINGKASAN

Mekanisme pembetulan ralat adalah suatu mekanisme pengesanan ralat yang bukan sahaja mengesan ralat yang wujud malah membetulkan ralat yang terdapat dalam data yang telah melalui proses penghantaran. Bagi kadar data yang tinggi, pembetulan ralat mesti dilakukan menggunakan logic digital iaitu perkakasan khas kerana penggunaan

perisian adalah lebih perlahan. Dalam membuat pembetulan, oleh kerana data yang ada dalam bentuk perwakilan binari, maka ralat yang dikesan akan ditukar kepada perwakilan binari yang sebaliknya. Terdapat tiga jenis pembetulan ralat yang ada iaitu pembetulan :

- Ralat bit tunggal
- Ralat bit berbilang dan
- Ralat letusan

Konsep asas pembetulan ralat boleh diaplikasi dan diubahsuai bagi ketiga-tiga jenis pembetulan ralat tersebut dengan mengambilkira hubungan di antara mesej asal dan bit ulangan yang ditambahkan ke dalam mesej asal untuk membentuk suatu mekanisme pembetulan ralat. Secara amnya, bit-bit ulangan mestilah boleh mencakupi semua kombinasi kedudukan ralat yang mungkin.

Pembetulan ralat penting kerana ia dapat menambahkan kebolehpercayaan terhadap peranti storan disamping menyokong komunikasi berkadar tinggi.

Kod pembetulan ralat adalah kod yang digunakan bagi membuat pembetulan ralat. Ia terbahagi kepada 2 jenis iaitu kod pepohon dan kod blok. Antara contoh-contoh kod pembetulan ralat yang boleh diaplikasikan adalah kod Hamming, Kod Reed-Solomon dan kod Ulangan.

Kod Hamming adalah suatu teknik untuk menentukan kedudukan mana yang terdapat ralat dengan memanipulasikan bit-bit ulangan yang telah dibangunkan oleh R.W. Hamming. Kod Reed Solomon adalah kod pembetulan ralat dari jenis kod blok linear yang dinyatakan sebagai RS(n,k) dengan simbol s bit. Suatu penyahkod Reed-Solomon boleh membetulkan sehingga t symbol yang mengandungi ralat dalam satu-satu kata kod dimana n-k=2t. Konsep asal bagi kod ulangan adalah dengan mengulang setiap bit dalam bit data mengikut bilangan yang tertentu.

Projek akan menggunakan kod pembetulan ralat dari jenis blok dimana ia mesti sesuai untuk pembetulan ralat berbilang ataupun ralat letusan. Pendekatan yang akan digunakan adalah pendekatan logik digital iaitu pembangunan perkakasan khas kerana ingin meyokong komunikasi pantas. Setelah analisa dan kajian dijalankan didapati beberapa ciri menarik dalam kod Reed-Solomon sesuai untuk digunakan dalam pembangunan projek.

3.1 PENGENALAN

VHDL adalah ringkasan untuk **V**ery high speed integrated circuit **H**ardware **D**escription **L**anguage. Ia digunakan untuk tujuan simulasi, permodelan, pengujian, sintesis, rekabentuk dan dokumentasi sesuatu sistem digital. Bahasa ini menyediakan suatu format mudah dan padat untuk persembahan berhieraki bagi penerangan fungsian dan pendawaian sistem digital.

3.2 UNIT REKABENTUK VHDL

VHDL terbahagi kepada 3 komponen asas iaitu:

- Entiti
- Senibina
- Konfigurasi

Untuk penggunaan yang lebih kompleks, VHDL telah menambahkan dua komponen tambahan iaitu:

- Pakej
- Badan pakej

3.2.1 ENTITI

Entiti adalah merupakan unit rekabentuk yang asas. Ia membina pengisytiharan **port** yang berantaramuka dengan sistem. Entiti mungkin adalah perwakilan untuk keseluruhan system, papan litar, cip, sel ataupun get.

Format untuk pengisytiharan entiti adalah seperti berikut:

```
entity nama_entiti is
        port (senarai_port);
end nama_entity
```

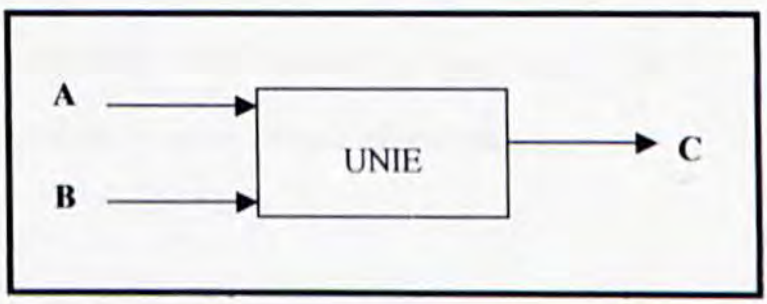
Fungsi port adalah untuk menyediakan saluran komunikasi antara satu entiti dengan persekitarannya. Setiap port yang ditakrifkan perlu mempunyai nama, mod operasi dan jenisnya.

Format untuk pengisytiharan port adalah seperti berikut:

```
port (nama_port :  mod jenis[bas]);
```

Pengisytiharan bas dibuat sekiranya port bersambung dengan lebih daripada satu output dari modul ataupun sistem lain.

Sebagai contoh:



Rajah 3.1 Rajah Blok untuk UNIE

Kodnya:

```
entity UNIE is
        port (A : in bit_vector (15 downto 0);
              B : in bit;
              C : out bit_vector (15 downto 0);
             );
end UNIE;
```

Berikut adalah jenis-jenis mod yang boleh digunakan dalam pernyataan port:

- IN → input atau masukan kepada modul
- OUT → output atau keluaran daripada modul
- INOUT → isyarat dua hala
- BUFFER → satu pendaftar bersambung kepada satu output

3.2.2 SENIBINA

Senibina adalah merupakan unit rekabentuk sekunder. Ia mewakili struktur sebenar yang membina suatu rekabentuk yang telah diberikan. Senibina menerangkan kelakuan fungsian sesuatu rekabentuk.

Suatu entiti rekabentuk tunggal mungkin mempunyai beberapa senibina di mana setiap senibina yang ditakrifkan mewakili kelakuan berlainan rekabentuk tersebut.

Format untuk pengisytiharan senibina adalah seprti berikut:

```
architecture nama_senibina of nama_entiti is
begin
        if B = '1' then
                C<=A;
        end if;
end  nama_senibina;
```

Sebagai contoh:

```
architecture LAKU_UNIE of UNIE is
        begin
        if B = '1' then
                C<=A;
        end if;
end LAKU_UNIE;
```

Nama port entiti dalam senibina menyediakan hubungan anatar pengisytiharan dengan senibina tersebut.

3.2.3 KONFIGURASI

Entiti dan senibina boleh digabungkan bagi membentuk suatu unit rekabentuk tunggal.

Format untuk pengisytiharan konfigurasi adalah seprti berikut:

```
configuration nama_konfigurasi of nama_entiti is
    for nama_senibina;
    end for;
end nama_konfigurasi;
```

Sebagai contoh:

```
configuration KONFIG_UNIE of UNIE is
    for LAKU_UNIE;
    end for;
end KONFIG_UNIE;
```

Arahan di atas akan mengikat entiti UNIE kepada senibinanya.

3.2.4 PAKEJ DAN BADAN PAKEJ

Pakej adalah merupakan suatu unit perpustakaan yang terdiri daripada pengisytiharan yang boleh digunakan dalam unit rekabentuk yang lain. Unit pakej terbahagi kepada 2 bahagian iaitu:

Pengisytiharan

Badan

Pengisytiharan suatu pakej mungkin terdiri daripada nilai tetap, jenis, fungsi dan pengisytiharan fungsi dan peosedur. Format ini adalah sama seperti pengisytiharan suatu entiti. Kelakuan fungsi dan subaturcara sebenar pula dimasukkan kedalam badan pakej.

Format untuk pengisytiharan pakej dan badan pakej adalah seperti berikut:

```
package nama_pakej is
        ... pengisytiharan
end nama_pakej;

pakage body nama_pakej is
        ... fungsi atau subaturcara sebenar
end nama_pakej;
```

Sebagai contoh:

```
package JENIS_UNIE is
begin
        subtype bit_16 is bit_vector (15 downto 0);
        type bit_16_array is array (integer range < >) of bit_16;
        function resolve_bit_16 (driver : in bit_16_array) return bit_16;
        subtype bus_bit_16 is resolve_bit_16 bit_16;
end JENIS_UNIE;

pakage body JENIS_UNIE is
        function resolve_bit_16 (driver : in bit_16_array) return bit_16 is
        variable result : bit_16;
        begin
                result :=X "0000";
                for i in driver'range loop
                        result := result or driver (i);
                end loop;
        return result;
        end resolve_bit_16;
end JENIS_UNIE;
```

3.3 SIMULASI VHDL

Simulasi adalah proses mengenalpasti rekabentuk litar VHDL. Suatu unit pengujian perlu dicipta sebelum proses simulasi dilakukan. Setiap modul yang telah direkabentuk mesti mempunya satu unit pengujian bagi menguji operasinya.

Format untuk pengisytiharan unit pengujian adalah seperti berikut:

```
entity nama_unit_ujian is
        ....
end nama_unit_ujian;

architecture nama_senibina of nama_unit_ujian is
        component nama_entiti (Senarai port);
        end component;
... pengisytiharan isyarat
begin
        ... kenyataan
end nama_senibina;
```

Sebagai contoh:

```
entity UJI_UNIE is
```

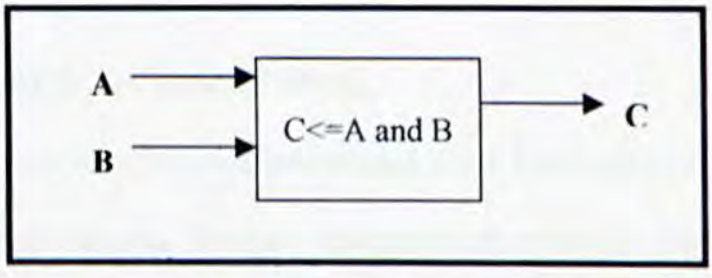
```
end UJI_UNIE;

architecture STIMULASI of UJI_UNIE is
        component UNIE
                port (A : in bit_vector (15 downto 0);
                      B : in bit;
                      C : out bit_vector (15 downto 0);
                      );
        end component;
begin
        process
        begin
                A<= "0000000000000111";
                B<= '1';
                wait for period;

                A<= "0000000100000111";
                B<= '0';
                wait for period;
        end process;
```

```
end STIMULASI;

configuration KFG_UJI_UNIE of UJI_UNIE
    for STIMULASI
    end for;
end CFG_UJI_UNIE;
```

3.4 PENERANGAN KELAKUAN

Penerangan kelakuan memodelkan perkakasan dengan menerangkan apa atau bagaimana sistem berkelakuan. Ia adalah sangat berguna bagi suatu sistem litar yang kecil dan ringkas. Walaubagaimanapun, penerangan kelakuan bagi litar yang lebih besar dan kompleks adalah sukar walaupun ianya adalah tidak mustahil untuk dilakukan. Untuk litar yang besar, penerangan kelakuan boleh menyebabkan pengurangan tahap kebolehgunaan semula kod dan membuatkan proses pembangunan kod menjadi lebih susah.

Contoh suatu penerangan kelakuan adalah seperti yang ditunjukkan dalam rajah model di bawah.

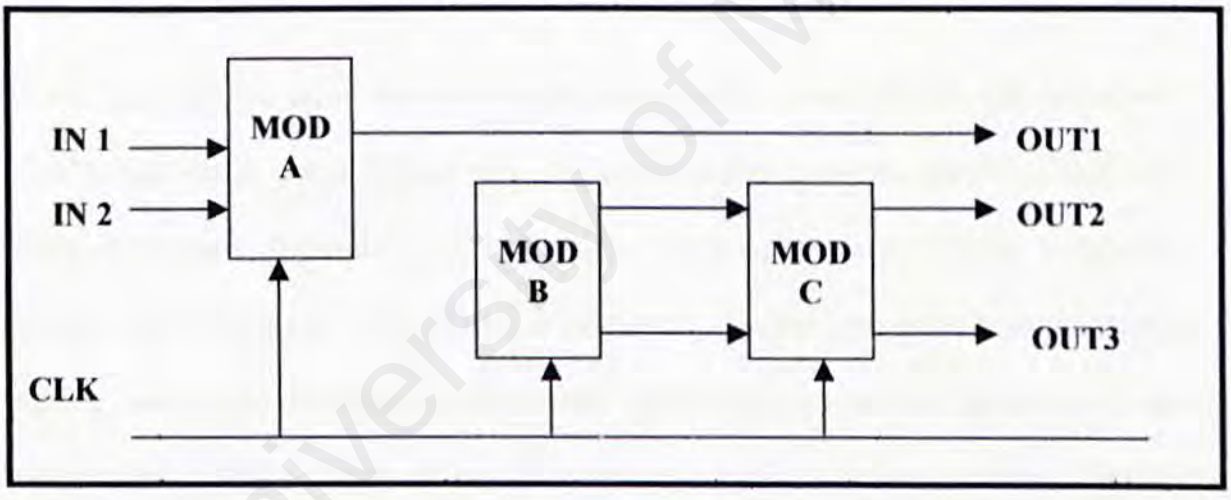


Rajah 3.2 Model Kelakuan untuk Pembanding

3.5 PENERANGAN STRUKTUR

Penerangan struktur menerangkan entiti dalam konteks struktur yang tertakrif. Kita boleh mengambil suatu entiti kecil yang ditakrifkan menggunakan penerangan kelakuan untuk menghubungkan mereka. Penerangan struktur adalah suatu tugas yang besar dan perlu dipecahkan kepada tugasan yang lebih kecil bagi memenuhi keperluan pembangunan kod atau bagi meningkatkan kepantasan perlaksanaan.

Contoh suatu penerangan struktu adalah seperti yang ditunjukkan dalam rajah model di bawah.



Rajah 3.3 Model Struktur

3.6 KEBAIKAN MENGGUNAKAN VHDL

VHDL digunakan bagi menghasilkan perkakasan untuk kod pembetulan ralat ini adalah kerana ciri-ciri kebaikannya. Dengan menggunakan VHDL, pengeluar cip boleh

menyediakan perekabentuk dengan penerangan VHDL bagi komponen mereka bagi tujuan simulasi.

Bahasa VHDL yang akan digunakan adalah bahasa yang tidak bergantung kepada teknologi perlaksanaan. Oleh itu, peranti yang serupa dalam fungsinya dilaksanakan dengan pantas menggunakan teknologi yang berlainan. Secara tidak langsung ini dapat mengurangkan keperluan untuk merekabentuk semula sepenuhnya suatu system apabila kemajuan teknologi memerlukan perlaksanaan yang baru. Penerangan VHDL boleh digunakan selagi fungsi litar tidak berubah.

VHDL juga adalah suatu bahasa pengaturcaraan yang senang dibaca dan berpiawai. Oleh kerana VHDL adalah bahasa teks yang menerangkan kelakuan, adalah lebih mudah untuk menentukan kelakuan system daripada sesuatu penerangan VHDL berbanding dengan suatu schematic. Oleh kerana ia berpiawai, dengan pengetahuan tentang suatu bahasa yang lain, seseorang pengaturcara boleh dengan mudah memahami dan menggunakan VHDL. Pengaturcara akan dengan mudah memahami kelakuan system yang direkabentuk menggunakan pelbagai alatan rekabentuk.

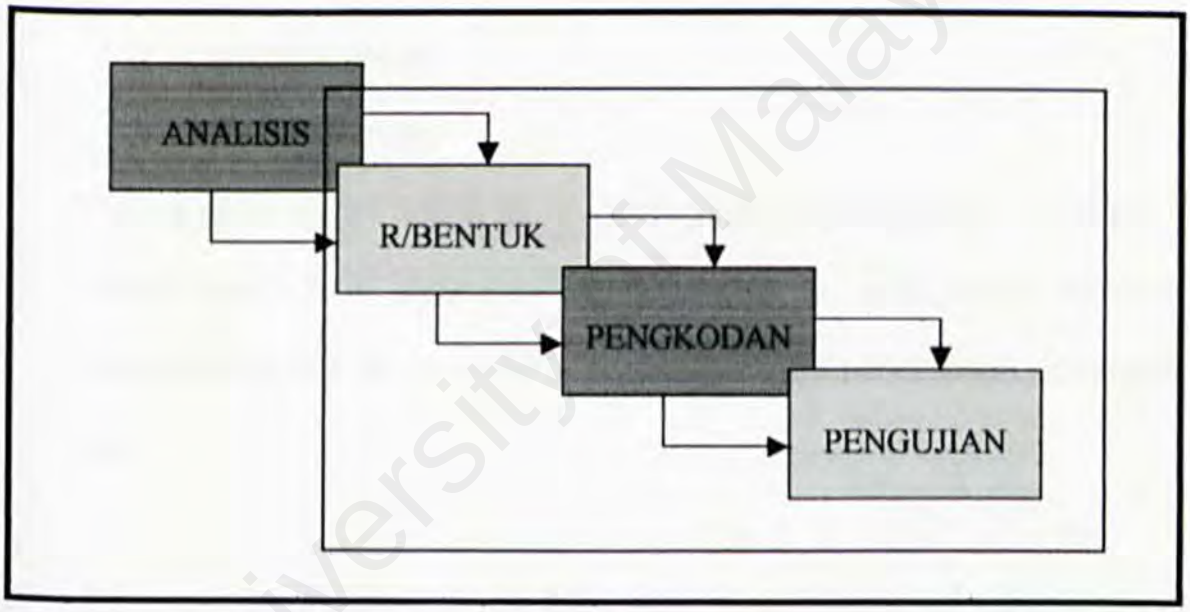
Antara lain-lain kebaikan VHDL adalah kod-kod yang terhasil dalam VHDL adalah boleh guna semula. Sama seperti lain-lain program komputer, blok kod yang ditulis oleh suatu perekabentuk boleh diguna semula oleh perekabentuk yang lain. Ini akan dapat menjimatkan masa dan kos yang diperlukan untuk memenuhi keperluan semasa.

3.7 RINGKASAN

VHDL adalah bahasa pengaturcaraan khas yang digunakan untuk membangunkan suatu logic digital dalam penghasilan perkakasan. 3 komponen asas dalam VHDL adalah entity, senibina dan konfigurasi. Ini diikuti oleh 2 lagi komponen iaitu pakej dan badan pakej. Menggunakan VHDL kita dapat menyatakan masukkan-masukkan input yang perlu diproses untuk menhasilkan keluaran yang dikehendaki. Penerangan kelakuan memodelkan perkakasan dengan menerangkan apa atau bagaimana sistem berkelakuan. Penerangan struktur menerangkan entiti dalam konteks struktur yang tertakrif. VHDL adalah bahasa pengaturcaraan yang sesuai untuk pembangunann perkakasan kerana ia tidak bergantung kepada teknologi perlaksanaan, senang dibaca, berpiawai dan kod yang dijana menggunakannya boleh digunakan semula untuk projek yang lain.

4.1 PENGENALAN

Seperti yang telah dinyatakan dalam Bab 1, pembangunan perkakasan khas ini terbahagi kepada 4 fasa utama iaitu fasa analisis, rekabentuk, pembangunan kod dan pengujian. Fasa-fasa rekabentuk, pembangunan kod dan pengujian dianggap sebagai suatu fasa tunggal dan setiap fasa dihubungkan secara dua hala. **Rajah 4.1** menunjukkan fasa-fasa yang telah dinyatakan.



Rajah 4.1 Fasa pembangunan projek

Jadual bagi fasa-fasa tersebut telah diterangkan dalam bahagian penjadualan Bab 1.

4.1.1 FASA ANALISIS

Fasa analisis melibatkan analisa kepada perkara-perkara berikut:

- Jenis pendekatan pembangunan
- Jenis Kod Pembetulan Ralat yang ingin dijanakan
- Spesifikasi kod yang telah dipilih
- Spesifikasi Senibina dan Rekabentuk dan
- Risiko-risiko yang mungkin berlaku

Jenis pendekatan pembangunan yang boleh dilakukan terbahagi kepada dua iaitu:

- Pendekatan digital
- Pendekatan analog

Selepas perbandingan bagi kedua-dua jenis pendekatan telah dibuat, pendekatan secara digital akan digunakan memandangkan ia lebih pantas daripada pendekatan analog dan sesuai untuk pembangunan dan perlaksanaan perkakasan ini.

Untuk memilih Kod Pembetulan Ralat yang sesuai, 3 jenis kod yang berlainan telah dianalisa dengan menyenaraikan kebaikan dan kelemahan setiapnya untuk dibuat perbandingan. Kod-kod tersebut adalah:

- Kod Hamming
- Kod Ulangan dan
- Kod Reed Solomon

Penerangan bagi ketiga-tiga jenis kod tersebut dapat dilihat dalam Bab 2.

Memandangkan kod Reed-Solomon dapat memenuhi keperluan terhadap pembetulan ralat bagi kedudukan bit ralat letusan yang rawak, maka ia akan di gunakan sebagai kod yang akan dibangunkan untuk perkakasan khas yang ingin dibina.

Analisis-analisis lain yang dibuat iaitu spesifikasi kod RS, senibina, rekabentuk dan risiko akan diterangkan dalam bahagian-bahagian yang seterusnya.

4.1.2 FASA REKABENTUK

Selepas semua analisis dan sintesis kepada semua maklumat yang diperolehi telah dibuat, beberapa cadangan rekabentuk dikemukakan. Setiap rekabentuk pengkod dan penyahkod ini di analisis sekali lagi bagi memilih rekabentuk yang paling baik dan sesuai untuk digunakan dalam pembangunan projek.

4.1.3 FASA PENGKODAN

Selepas fasa analisis dan rekabentuk selesai, pembangunan projek akan seterusnya menumpu kepada pembangunan kod-kod aturcara VHDL menggunakan Xilinx. Aturcara dibuat berdasarkan kepada rekabentuk yang telah disiapkan dalam fasa analisis dan rekabentuk. Walaubagaimanapun, rekabentuk asal boleh diubahsuai bagi penyesuaian kepada masalah-masalah lain yang mungkin wujud kemuadiannya.

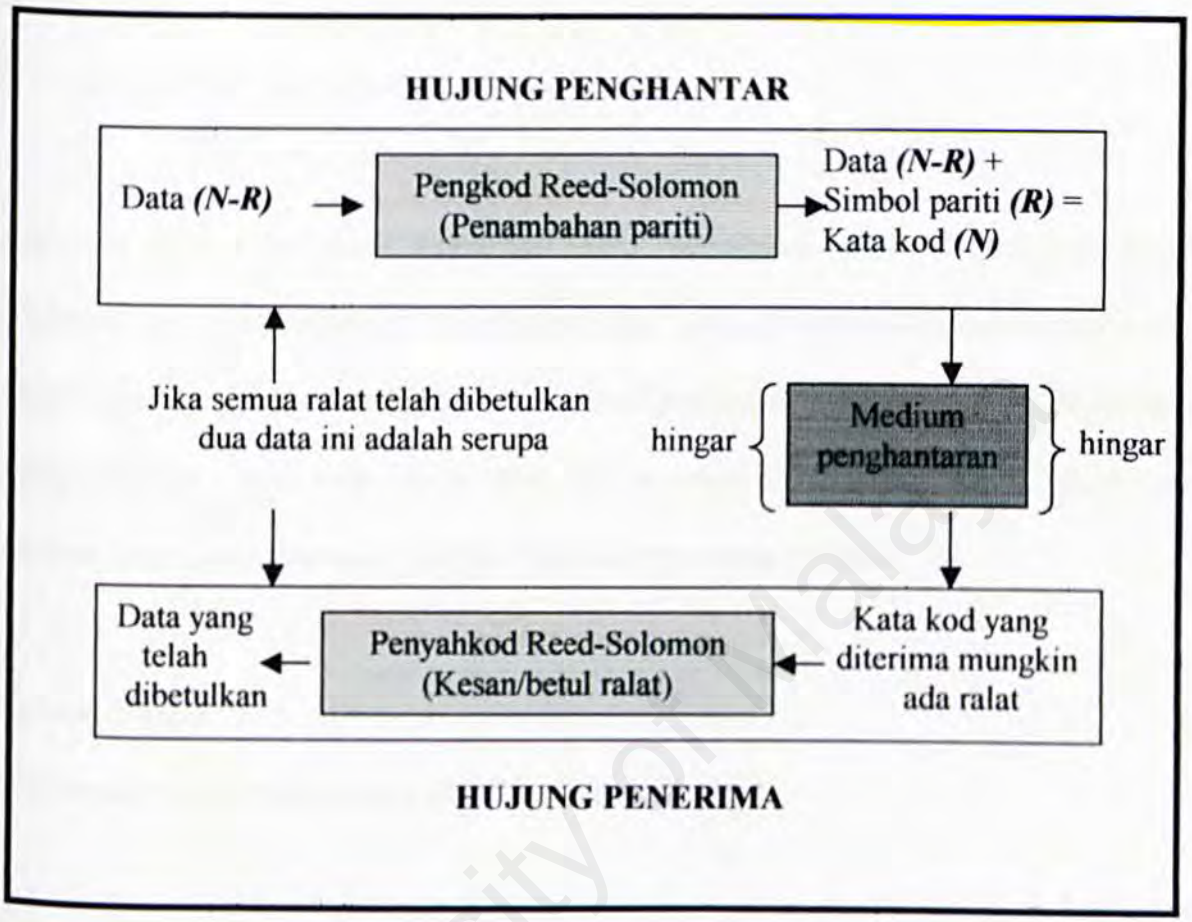
4.1.4 FASA PENGUJIAN

Dijangkakan bahawa fasa pengkodan dan fasa ujian akan dilaksanakan secara serentak dimana setiap modul yang dibina akan diuji bagi memastikan setiap kesilapan dikesan lebih awal bagi mengelakkan ralat berpropagasi.

Setiap modul rekabentuk yang telah dikodkan akan diuji sama ada ia boleh menjalankan fungsi seperti yang telah dinyatakan dalam rekabentuknya dan menghasilkan output seperti yang dikehendaki.

4.2 ARAS SISTEM REED-SOLOMON

Rajah blok aras sistem bagi perlaksanaan pengkod dan penyahkod RS ditunjukkan dalam **rajah 4.2.**



Rajah 4.2 Rajah blok aras sistem RS

Data asal yang merupakan suatu blok yang mengandungi N–R simbol dilarikan menerusi suatu pengkod RS dimana simbol pariti R akan ditambahkan kepada data asal tersebut bagi membentuk suatu kata kod dengan panjang N.

Pengkodan RS melibatkan penjanaan simbol-simbol pariti daripada data yang asal. Proses penjanaan ini berasaskan kepada aritmetik ruang finit dimana elemen-elemen dalam ruang adalah nilai-nilai dari 0 hingga 2m + 1 dimana m adalah jumlah bit per

simbol. Oleh itu, bagi suatu lebar bit 3, ruang tersebut mengandungi nilai-nilai 0,1,2,3,4,5,6 dan 7 yang tidak semestinya berturutan.

Beberapa definisi lain turut diperlukan untuk memahami pembolehubah yang perlu diketahui bagi menjanakan kod RS yang tertentu. Yang pertama adalah polinomial ruang yang digunakan untuk menentukan turutan elemen-elemen dalam ruang finit. Polinomial ruang yang sah adalah suatu fungsi lebar bit yang akan dioperasikan. Lebar bit tertentu memberikan nilai polinomial tertentu yang sah bagi ruang tersebut.

Sebagai contoh:

Polinomial ruang yang sah bagi m = 3 adalah 11 dan 13.

Biasanya, polinomial ruang yang mesti digunakan adalah merupakan spesifikasi yang telah ditentukan. Dalam pengkod RS yang tertentu, nilai ini disimpan dalam cip itu sendiri tetapi ada juga pengkod yang membenarkan sebarang nilai polinomial yang munasabah digunakan.

Akhir sekali, untuk menjana suatu perlaksanaan RS yang tertentu, kita perlu mengetahui asas permulaan polinomial penjana. Ia adalah nilai yang selalunya ditentukan menerusi spesifikasi. Sebarang nilai boleh digunakan sebagai asas polinomial ini.

Satu lagi ciri aras sistem bagi pengkodan RS adalah sama ada perlaksanaannya adalah sistematik ataupun tak sistematik. Perlaksanaan sistematik menghasilkan suatu kata kod yang mengandungi aliran data masukan asal yang tidak berubah. Manakala perlaksanaan tak sistematik pula aliran data masuk berubah-ubah semasa proses pengkodan berlangsung. Kebanyakkan spesifikasi melibatkan perlaksanaan kod RS yang sistematik.

4.3 SPESIFIKASI KOD

Di bawah dinyatakan ciri-ciri kod Reed-Solomon yang ingin dibangunkan untuk perkakasan khas pembetulan ralat:

1. Menggunakan kod RS sistematik
2. Penjanaan RS(16,13) iaitu 13 bait data dan 3 bait pariti
3. Saiz simbol adalah 8 bit
4. Nilai polinomial ruang adalah
5. Nilai asas permulaan polinonial penjana adalah

Rajah 4.3 menunjukkan struktur am bagi RS(255,249) yang ingin dijanakan oleh perkakasan khas.

Rajah 4.3 Struktur kod RS(16,13)

di mana

$2t = 3$ bait

$t = 1.5$ bait

$= 12$ bit

4.4 SENIBINA KOD REED-SOLOMON

Ciri utama teori ruang finit yang digunakan dalam kod RS adalah operasi-operasi aritmetik ke atas elemen-elemen dalam ruang tersebut akan sentiasa menghasilkan suatu hasil dalam ruang itu. Pengkodan dan penyahkodan Reed-Solomon perlu menjalankan operasi-operasi tersebut. Operasi-operasi ini memerlukan fungsi perkakasan atau perisian yang khas untuk dilaksanakan.

Kata kod Reed-Solomon dijanakan menggunakan suatu polynomial khas. Bentuk am polinomial penjana adalah:

$$g(x) = (a - x^j)\,(a - x^{j+1})\ldots\,(a - x^{j+2t})$$

dan kata kod dijana menggunakan:

$$c(x) = g(x) \cdot i(x)$$

dimana

$g(x)$ adalah polinomial penjana

$i(x)$ adalah blok maklumat

$c(x)$ adalah kata kod yang sah

a dirujuk sebagai suatu elemen primitif bagi ruang tersebut

Sebagai contoh:

Penjana untuk RS(255,249)

$$g(x) = (a - x^i)(a - x^2)(a - x^3)(a - x^4)(a - x^5)$$

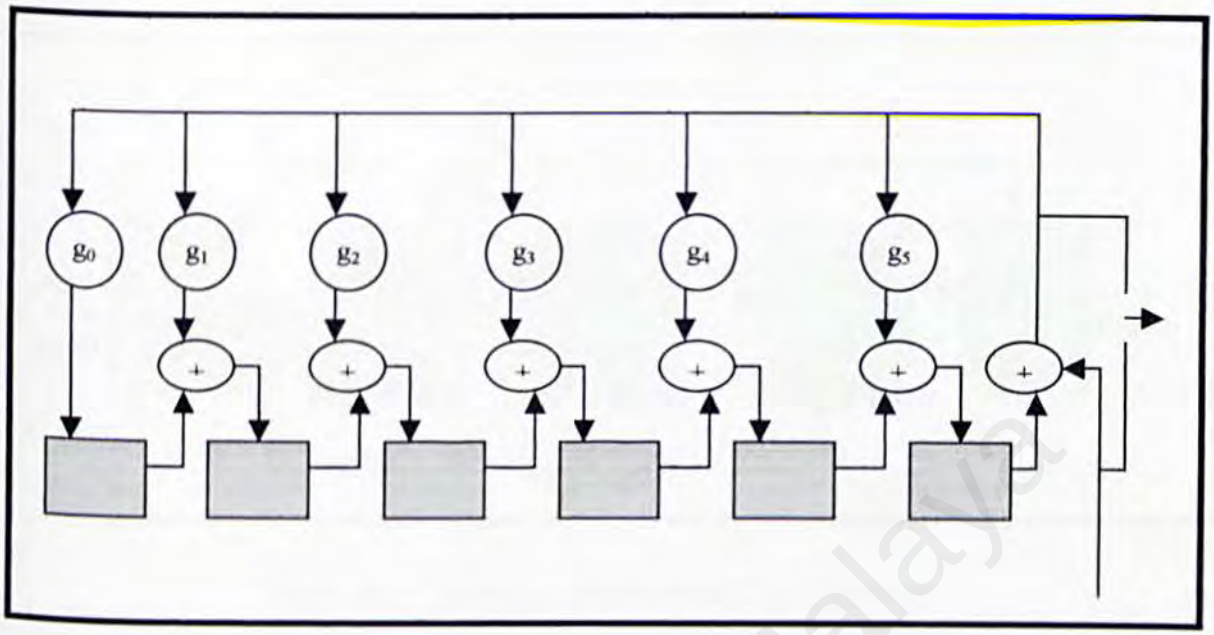
$$= x^6 + g_5 x^5 + g_4 x^4 + g_3 x^3 + g_2 x^2 + g_1 x^1 + g_0$$

4.4.1 SENIBINA PENGKOD

Simbol pariti 2t dalam suatu kata kod Reed-Solomon yang sistematik adalah seperti berikut:

$$p(x) = i(x) \cdot x^{n-k} \bmod g(x)$$

Rajah 4.4 menunjukkan senibina bagi suatu pengkod RS(255,249) yang sistematik.

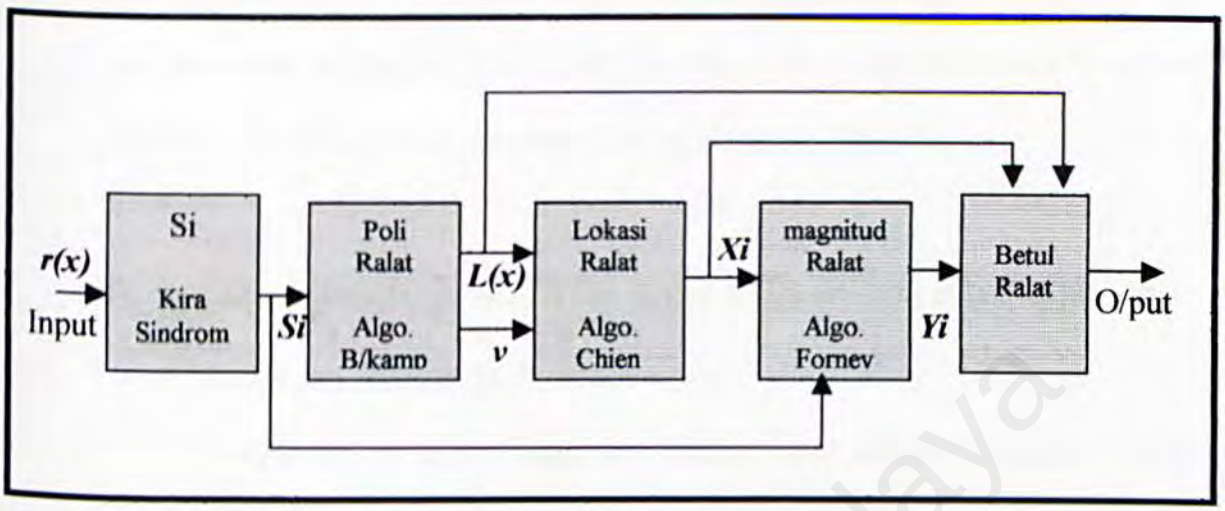


Rajah 4.4 Senibina pengkodan Reed-Solomon

Setiap 6 daftar memegang satu simbol 8 bit. Pengoperasi aritmetik melakukan penambahan atau pendaraban ruang finit dalam satu simbol lengkap.

4.2.2 SENIBINA PENYAHKOD

Rajah 4.5 menunjukkan senibina am bagi suatu penyahkod kod Reed-Solomon.



Rajah 4.5 Senibina penyahkodan Reed-Solomon

di mana

r(x)	kata kod yang diterima
Si	sindrom-sindrom
L(x)	polinomial penentu lokasi ralat
Xi	lokasi ralat
Yi	magnitud ralat
c(x)	kata kod yang ditemui
v	bilangan ralat

Kata kod ***r(x)*** yang diterima adalah kata kod asal ***c(x)*** yang bercampur dengan ralat.

$$r(x) = c(x) + e(x)$$

Suatu penyahkod Reed-Solomon berupaya mengenalpasti kedudukan dan magnitud ralat sehingga t ralat (atau 2t pemadaman) dan seterusnya membuat pembetulan terhadap ralat tersebut ataupun membuat pemadaman.

Berikut diterangkan fungsi bagi setiap modul dalam senibina dalam **rajah 4.5.**

- Pengiraan Sindrom (Kira Sindrom)

  Pengiraan di dalam fungsi ini adalah sama seperti pengiraan pariti. Suatu kata kod Reed-Solomon mempunyai 2t sindrom yang bergantung hanya kepada ralat dan bukan kepada kata kod yang dihantar. Sindrom tersebut boleh dikira dengan penukargantian asas 2t bagi polinomial penjana g(x) kepada r(x).

- Mencari Lokasi Ralat Simbol

  Ini melibatkan penyelesaian persamaan serentak dengan nilai t tidak diketahui. Terdapat beberapa algoritma pantas yang boleh menyelesaikan persamaan ini. Algoritma-algoritma tersebut menggunakan struktur khas matriks kod Reed-Solomon dan dengan berkesan boleh mengurangkan kos pengiraan yang diperlukan. Secara amnya, ini melibatkan dua langkah dalam fungsi yang berikutnya.

- Mencari suatu polinomial penentu lokasi ralat (Poli Ralat)

  Ini boleh dilakukan menggunakan algoritma Berlekamp-Massey ataupun algoritma Euclid. Algoritma Euclid adalah lebih banyak digunakan dalam aktiviti sebenar kerana ia mudah untuk dilaksanakan. Walaubagaimanapun, penggunaan algoritma Berlekamp-Massey menghasilkan perlaksanaan perkakasan atau perisian yang lebih efisien.

- Mencari asas bagi polinomial tersebut (Lokasi Ralat)

  Daripada polinomial yang didapati daripada fungsi Poli Ralat, lokasi ralat dapat dikira menggunakan algoritma pencarian Chien.

- Mencari nilai ralat simbol (magnitud Ralat)

  Sekali lagi, ini melibatkan penyelesaian persamaan serentak dengan nilai t yang tidak diketahui. Selalunya algoritma Forney akan digunakan untuk pengiraan yang lebih pantas.

- Membetulkan ralat (Betul Ralat)

  Dengan kedudukan dan magnitud ralat yang telah diperolehi, ralat akan dbetulkan. Output adalah data yang telah dibetulkan.

4.5 ANALISIS REKABENTUK

Daripada spesifikasi kod RS yang telah ditetapkan seperti yang di atas dan dengan merujuk kepada senibina pengkod dan penyahkod RS, maka beberapa rekabentuk telah dicipta bagi memenuhi ciri-ciri spesifikasi tersebut. Rekabentuk-rekabentuk yang dicadangkan dicipta berdasarkan kepada pendekatan-pendekatan yang berbeza walaupun ia bakal memberikan hasil yang sama.

4.5.1 CADANGAN REKABENTUK PENGKOD

Bagi rekabentuk pengkod, didapati ada 2 pendekatan yang boleh digunakan untuk menghasilkan rekabentuk yang berupaya menjanakan kod sistematik RS(16,13) dengan 8 bit simbol.

CADANGAN SATU

Pengkod terdiri daripada 4 modul yang dibina berdasarkan kepada persamaan-persamaan kod RS iaitu:

- $p(x) = i(x) \cdot x^{n-k} \bmod g(x)$
- $g(x) = x_3 + g_2x_2 + g_1x_1 + g_0$
- $c(x) = i(x) \cdot g(x)$

dimana

$p(x)$ adalah simbol pariti

$g(x)$ adalah polinomial penjana

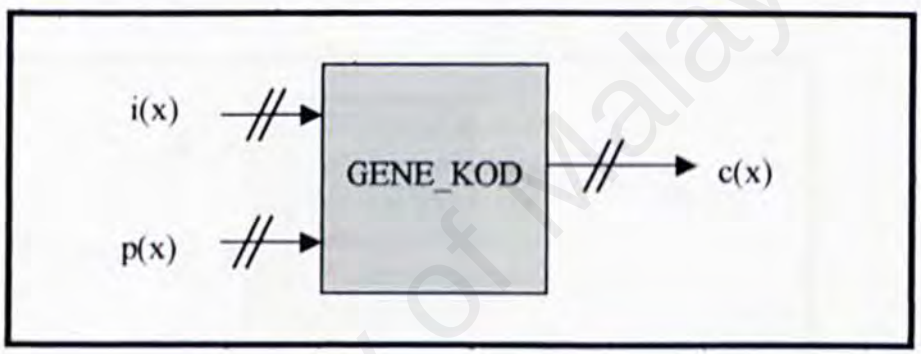
$c(x)$ adalah kata kod RS

$i(x)$ adalah blok maklumat

Modul-modul tersebut adalah seperti berikut:

- Modul Gene_Kod

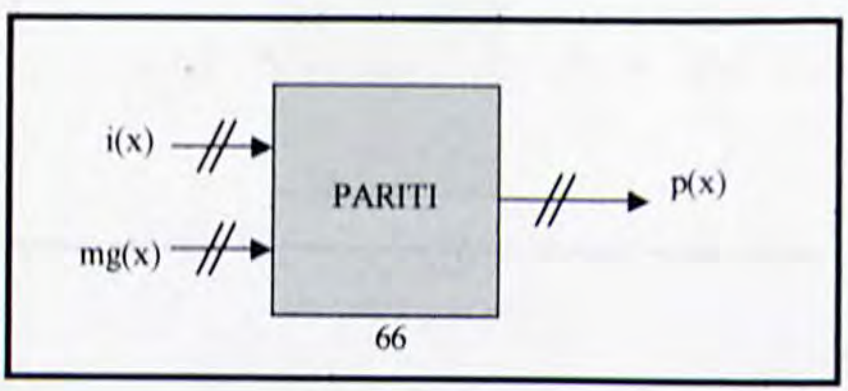
  Rajah 4.6 menunjukkan struktur ringkas modul ini yang akan menjanakan kod RS daripada blok maklumat dan simbol pariti yang di masukkan.



Rajah 4.6 Modul Gene_Kod

- Modul Pariti

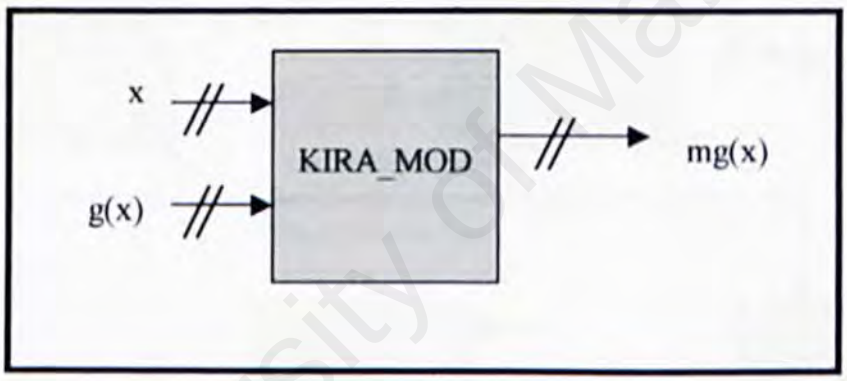
  Rajah 4.7 adalah struktur asas bagi modul pariti yang akan berupaya untuk menjanakan simbol-simbol pariti daripada blok maklumat dan nilai x^{n-k} ***mod g(x)*** yang dikira dalam modul yang seterusnya.



Rajah 4.7 Modul Pariti

- Modul Kira_Mod

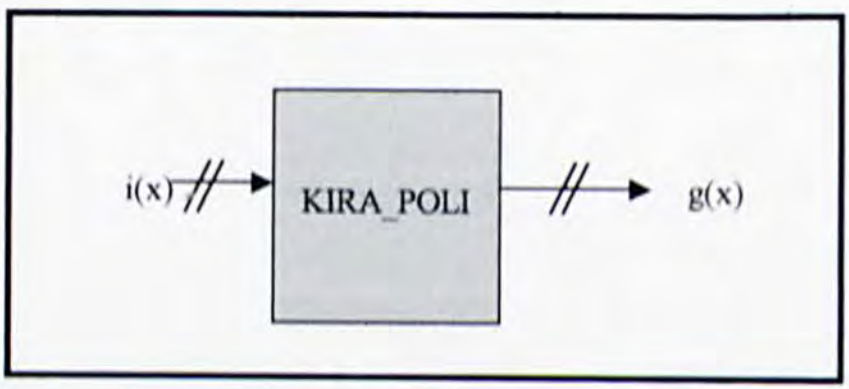
  Modul yang digunakan untuk membuat pengiraan bagi nilai x^{n-k} ***mod g(x)*** untuk melengkapkan persamaan pariti.



Rajah 4.8 Modul Kira_mod

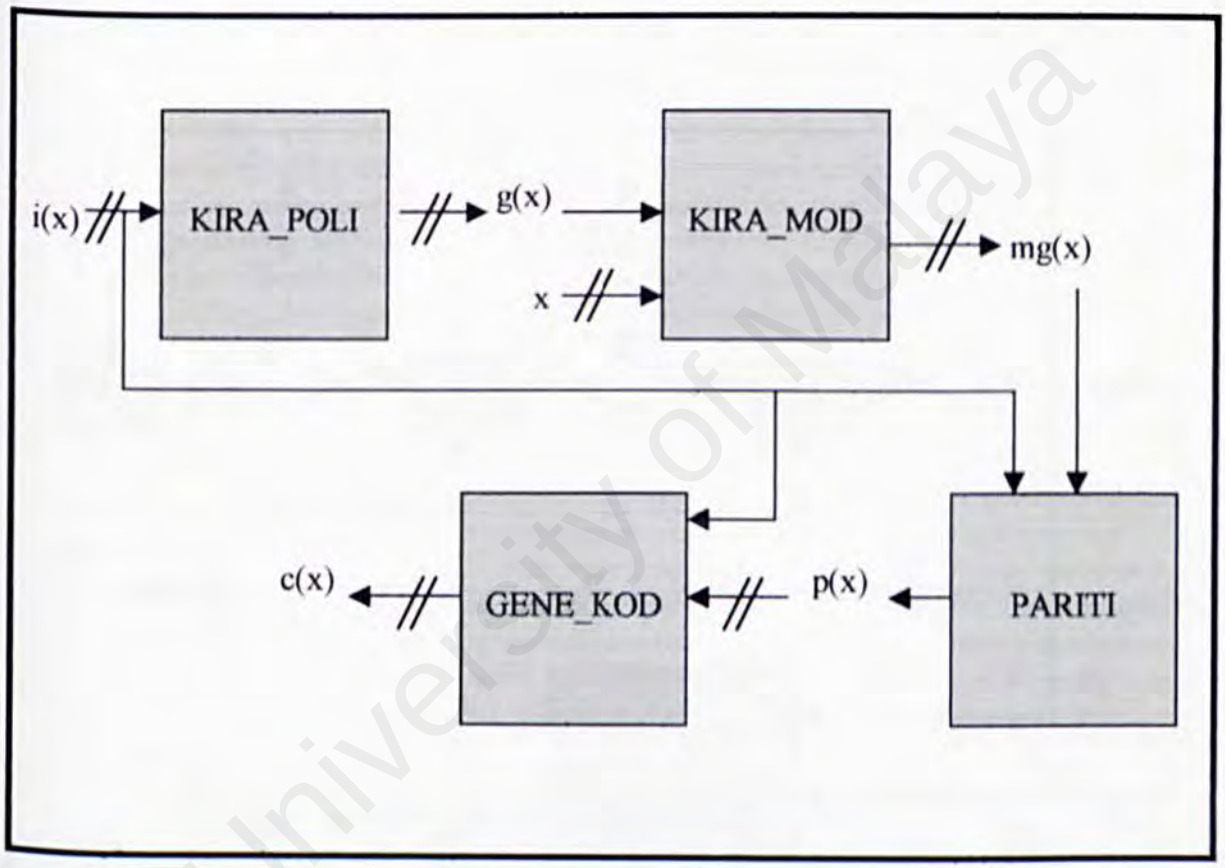
- Modul Kira_Poli

  Mengira nilai polinomial penjana bagi mendapatkan simbol pariti.



Rajah 4.9 Modul Kira_Poli

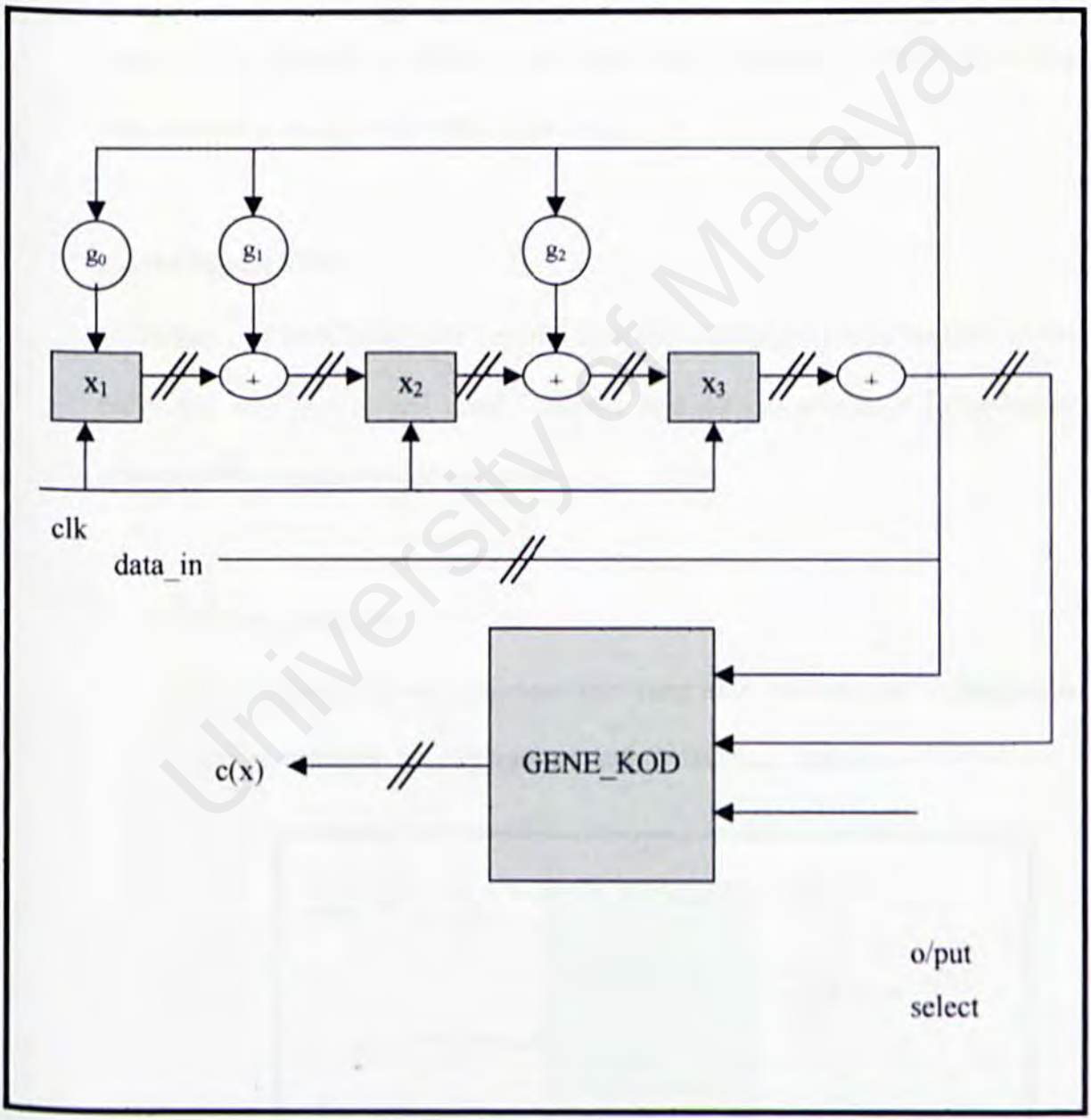
Empat modul di atas akan digabungkan untuk membentuk satu modul pengkodan seperti dalam **rajah 4.10.**



Rajah 4.10 Rekabentuk keseluruhan cadangan satu

CADANGAN DUA

Rajah 4.11 menunjukkan cadangan kedua rekabentuk pengkodan.



Rajah 4.11 Rekabentuk Pengkod menggunakan 3 daftar

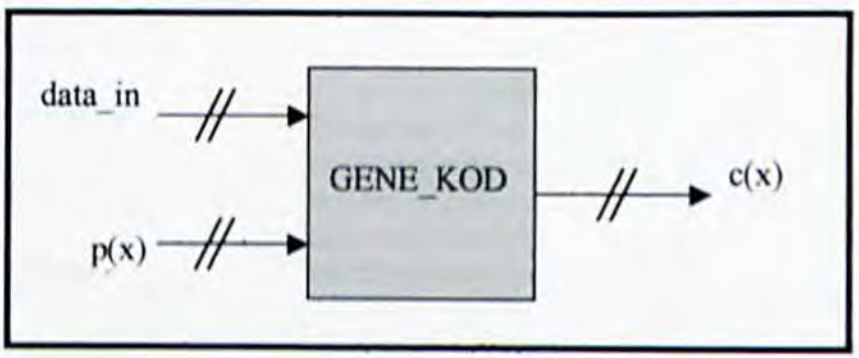
Cadangan kedua melibatkan rekabentuk pengkod yang terdiri daripada 1 modul dan 3 daftar. Data input dimasukkan terus ke dalam fungsi sementara dihantar kepada penjana simbol pariti. Ini adalah sifat bagi kod sistematik. 3 ruang finit membuat penambahan dan pendaraban dalam setiap daftar yang mengandungi satu simbol pariti setelah keseluruhan aliran data dmasukkan. Pada masa ini, *output select* dihantar ke daftar simbol pariti dan simbol itu dihantar ke hujung maklumat asal untuk membentuk kata kod.

CADANGAN TIGA

Cadangan tiga terdiri daripada 2 modul iaitu satu modul yang menjanakan simbol pariti dan satu lagi modul untuk menjana kod dengan membuat penambahan simbol pariti kepada jujukan data asal.

- Modul Gene_Kod

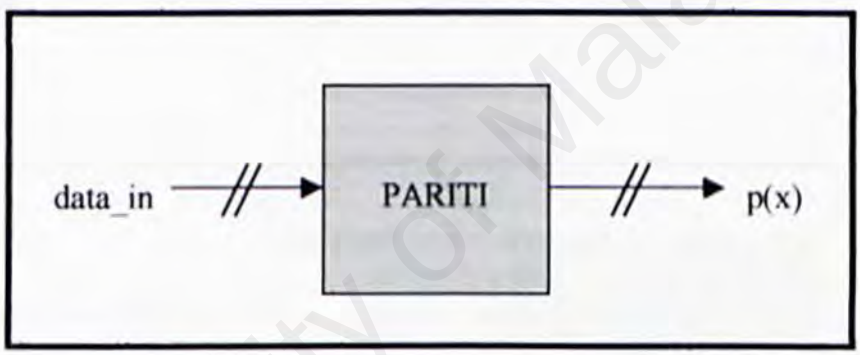
  Modul utama dalam penjanaan kod yang akan meletakkan simbol pariti p(x) ke bahagian hujung jujukan simbol data yang diterima.



Rajah 4.12 Modul Gene_Kod

- Modul Pariti

  Modul yang akan memproses blok maklumat yang masuk kepada suatu polinomial penjana yang akan seterusnya menghasilkan suatu simbol pariti untuk digunakan sebagai input modul Gene_Kod. **Rajah 4.13** menunjukkan struktur asas bagi modul penjanaan simbol pariti ini.



Rajah 4.13 Modul Pariti

4.5.2 REKABENTUK PENYAHKOD

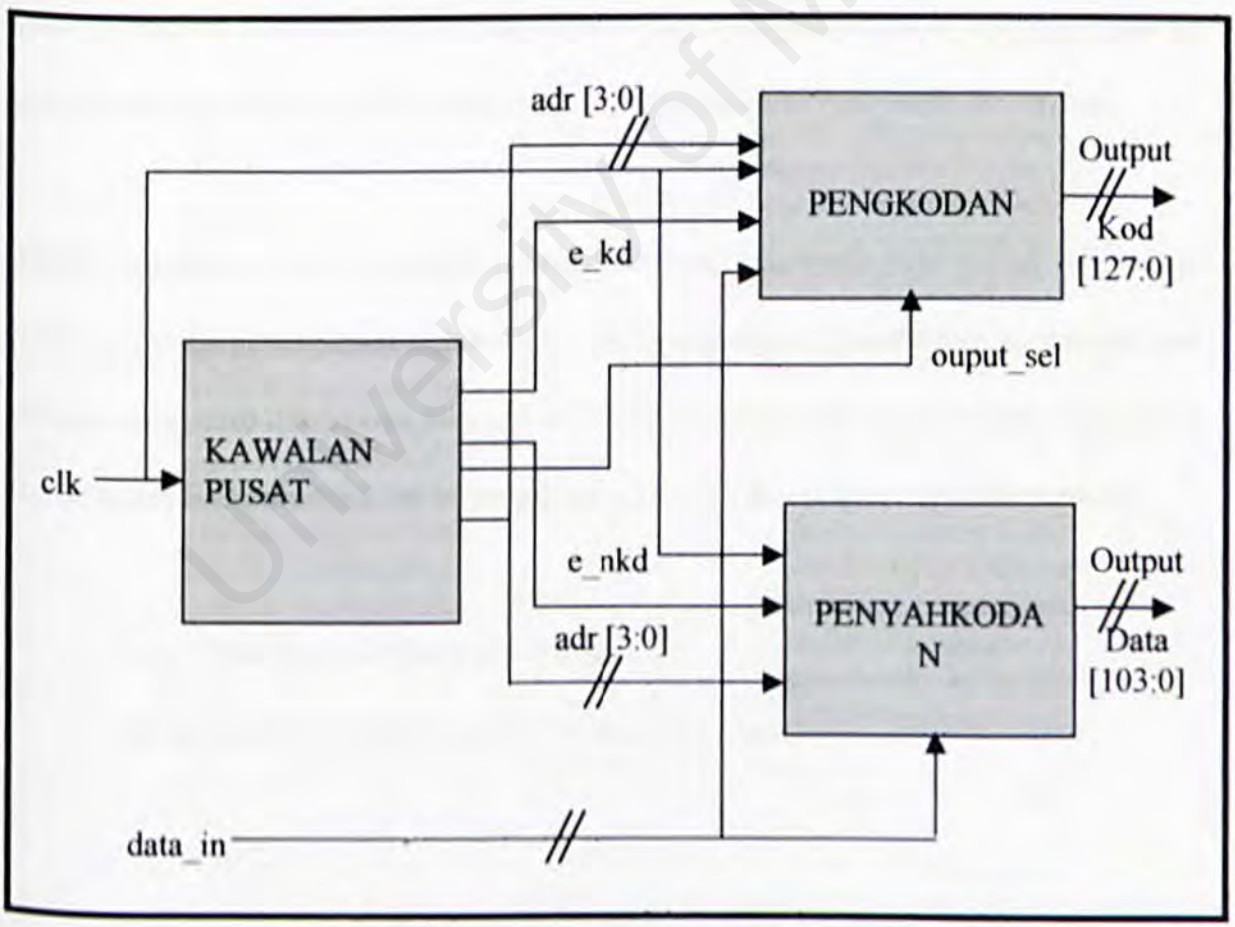
Rekabentuk penyahkod bagi RS(16,13) terdiri daripada 5 modul yang saling berkaitan iaitu:

- Modul Sindrom
- Modul Poli
- Modul Lokasi
- Modul Magnitud
- Modul Betul

Apabila suatu kata kod di hantar kepada penyahkod,ia akan melalui proses-proses pengiraan tertentu menerusi setiap modul.

4.6 REKABENTUK AKHIR PERKAKASAN KHAS

Setelah membuat penilaian ke atas persamaan-persamaan dan algoritma-algoritma RS(16,13), suatu keputusan telah dibuat untuk membangunkan perkakasan khas dengan berpandukan kepada rekabentuk-rekabentuk seperti dalam **rajah 4.14.**



Rajah 4.14 Rekabentuk keseluruhan bagi perkakasan khas

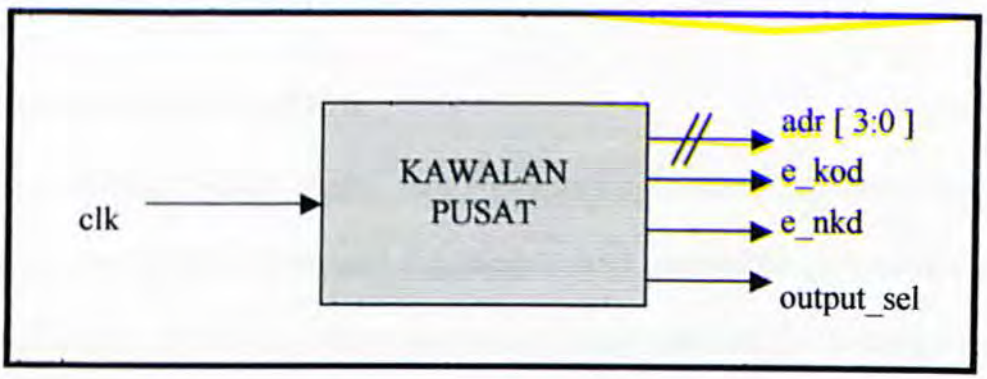
Rekabentuk di atas adalah terdiri daripada gabungan dua modul yang dapat menjana kod RS daripada jujukan data yang diterima dan menyahkodkan kata kod yang masuk. Untuk mengawal kedua-dua modul tersebut, maka satu unit kawalan diperlukan bagi memastikan kedua-dua fungsi bagi perkakasan ini dapat disinkronikan.

Apabila data memasuki perkakasan, satu isyarat dihantar bagi menentukan mod operasi yang perlu dijalankan oleh perkakasan iaitu sama ada mod pengkodan ataupun mod penyahkodan. Sekiranya data yang masuk ingin dikodkan, maka ia akan terus dihantar ke modul pengkodan untuk menjalani proses seterusnya sehinggalah suatu kata kod RS dapat dijanakan. Sebaliknya, jika data tersebut ingin dinyahkodkan ia akan di alirkan ke modul penyahkodan hingga menghasilkan jujukan data asal yang telah dibetulkan.

Modul pengkodan dan penyahkodan bukanlan terdiri daripada satu modul tunggal. Ia terdiri daripada gabungan submodul-submodul yang dapat menjalankan operasi-operasi tertentu yang telah ditetapkan. Ciri-ciri dan peranan setiap tiga modul di atas diterangkan dalam bahagian berikutnya. Ini termasuklah port input dan output bagi setiap modul.

4.6.1 MODUL KAWALAN PUSAT

Rajah 4.15 menunjukkan struktur modul kawalan.



Rajah 4.15 Modul Kawalan Pusat

Modul kawalan pusat adalah modul yang mengawal modul-modul lain dalam rekabentuk. Ia menyediakan alamat bagi modul-modul lain, mod pengkod atau penyahkod dan juga pemilihan output untuk modul pengkodan.

Port input:

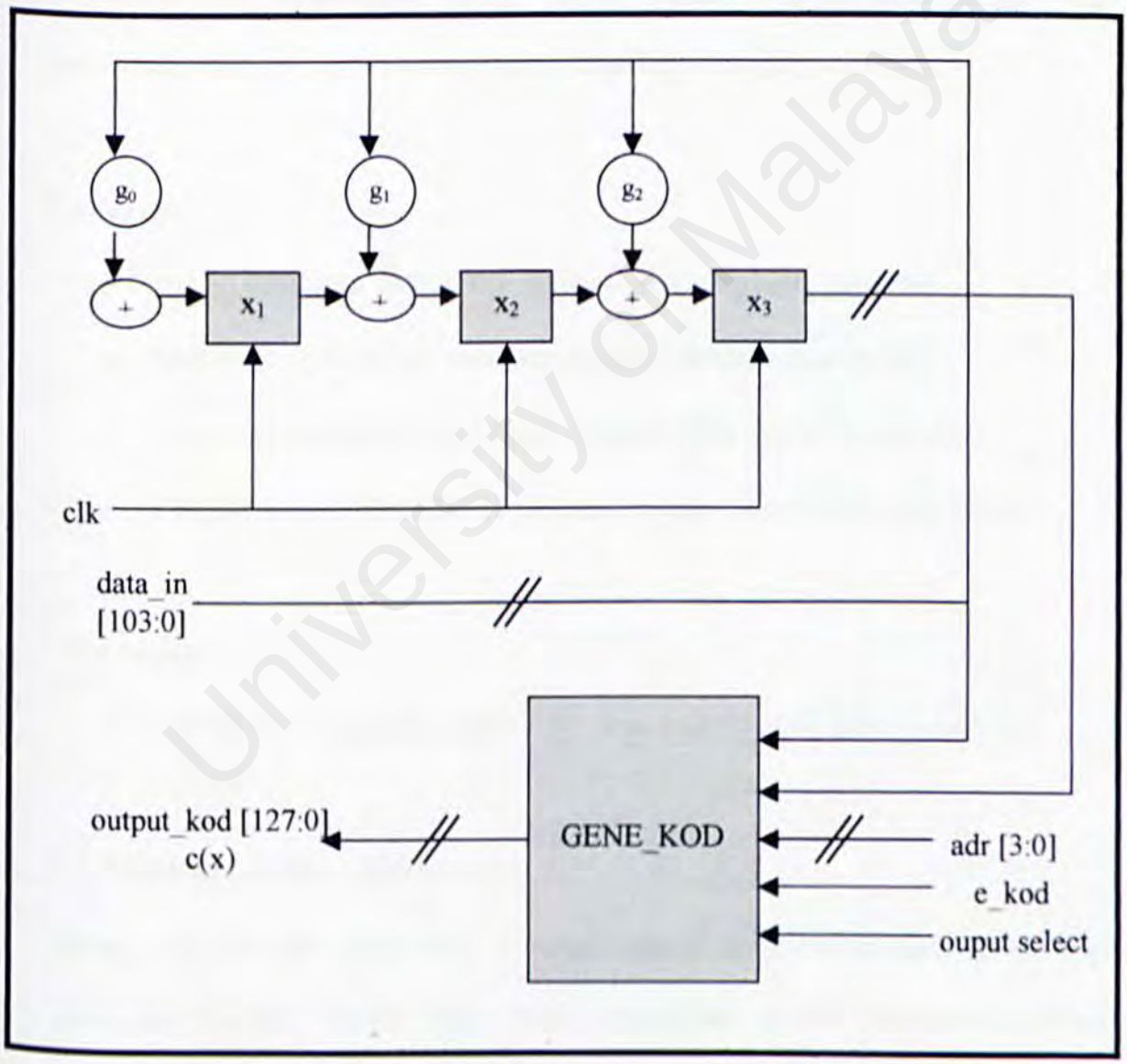
- 1 bit jam (clk) : menyediakan modul dengan jam sistem

Port output:

- 4 bit alamat (adr [3:0]) : memperuntukkan alamat untuk modul-modul lain
- 1 bit isyarat mod pengkodan (e_kod) : menentukan mod pengkodan
- 1 bit isyarat mod penyahkodan (e_nkd) : menentukan mod penyahkodan
- 1 bit isyarat output (output_sel) : memanggil kod simbol tertentu untuk pengkodan

4.6.2 MODUL PENGKODAN

Modul pengkodan adalah modul yang menjalankan proses pengkodan jujukan data ke dalam bentuk kata kod RS. **Rajah 4.16** menunjukkan keseluruhan rekabentuk penyahkod yang terdiri daripada 1 modul utama dan 6 daftar sebagai storan sementara nilai pariti.



Rajah 4.16 Rekabentuk Modul Pengkodan

Pembolehubah g_0 hingga g_2 adalah hasil daripada pendaraban yang dilakukan mengikut teori ruang finit manakala simbol pariti yang diperolehi daripada proses pendaraban ini di buat proses penambahan ruang finit untuk menghasilkan suatu simbol pariti akhir yang akan digunakan dalam proses pengkodan bagi memprolehi suatu kata kod RS. Simbol-simbol pariti yang diperolehi daripada setiap operasi pendaraban dan penambahan ruang finit ini akan disimpan dalam daftar yang merupakan ruang storan sementara yang dilabelkan sebagai x_1, x_2 dan x_3.

Port input:

- 104 bit data (data_in[103:0]) : jujukan data yang ingin dikodkan
- 4 bit alamat (adr[3:0]) : mengawal jujukan pengulangan semula
- 1 bit isyarat pengaktifan (e_kod) : mengaktifkan modul pengkodan
- 1 bit isyarat (output_select) : merujuk kepada simbol pariti yang diingini

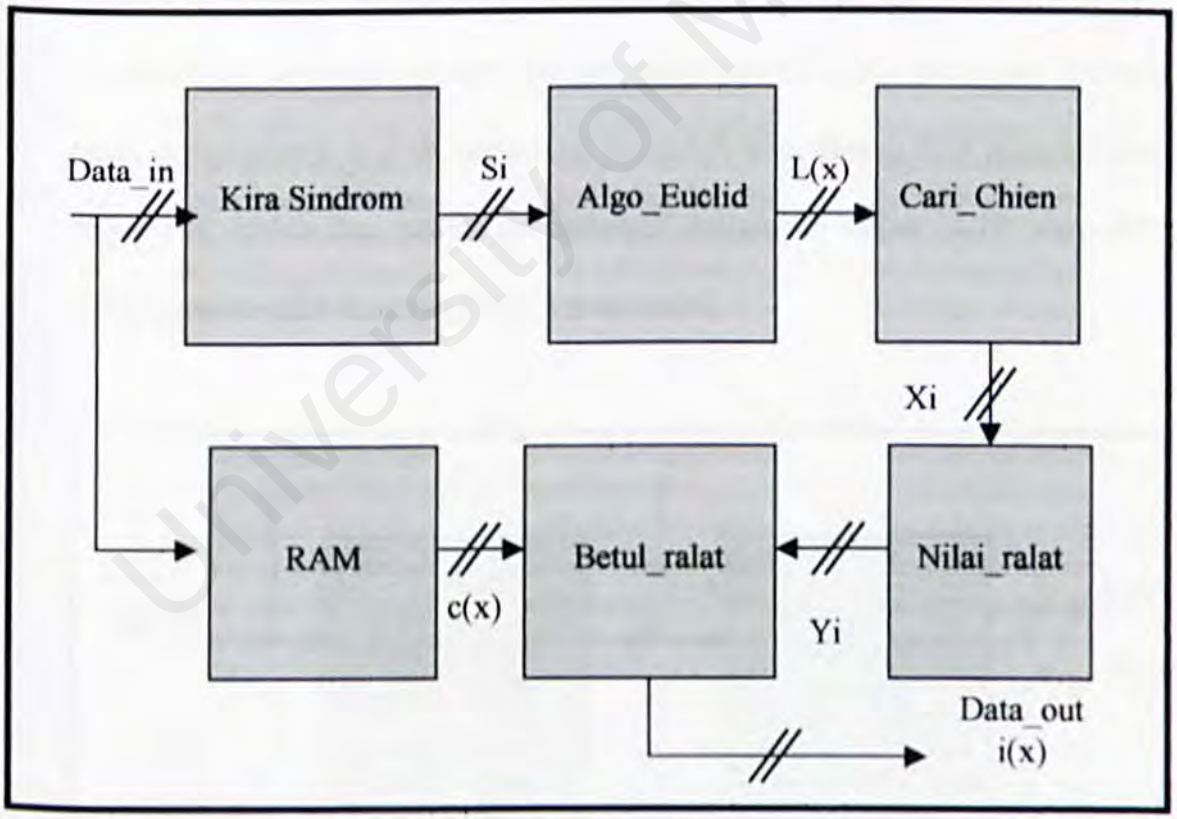
Port output:

- 128 bit output (output_kod[127:0]) : kata kod RS yang telah dijanakan

4.6.3 MODUL PENYAHKODAN

Proses penyahkodan melibatkan 5 fungsi utama yang menjalankan pengiraan-pengiraan tertentu seperti yang telah diterangkan dalam bahagian senibina penyahkodan. Oleh itu, fungsi-fungsi tersebut akan dijadikan sebagai satu modul bagi setiapnya. Selain itu, bagi tujuan menguruskan lengah suatu modul yang

bertindak sebagai ruang ingatan utama *Random Access Memory* (RAM) diwujudkan bagi menyimpan kata kod yang diterima sebagai input sehinggalah nilai-nilai yang perlu siap dikira sebelum kod tersebut dicapai semula daripada ingatan tadi untuk meneruskan proses penyahkodan sehingga kepada pembetulan ralat. Untuk modul penyahkodan, dua output dijangkakan iaitu kod yang telah dibetulkan dan bendera yang menandakan sama ada ralat berjaya dibetulkan ataupun tidak. **Rajah 4.17** menunjukkan rekabentuk bagi modul penyahkodan berdasarkan kepada rajah aliran data dalam analisis rekabentuk.

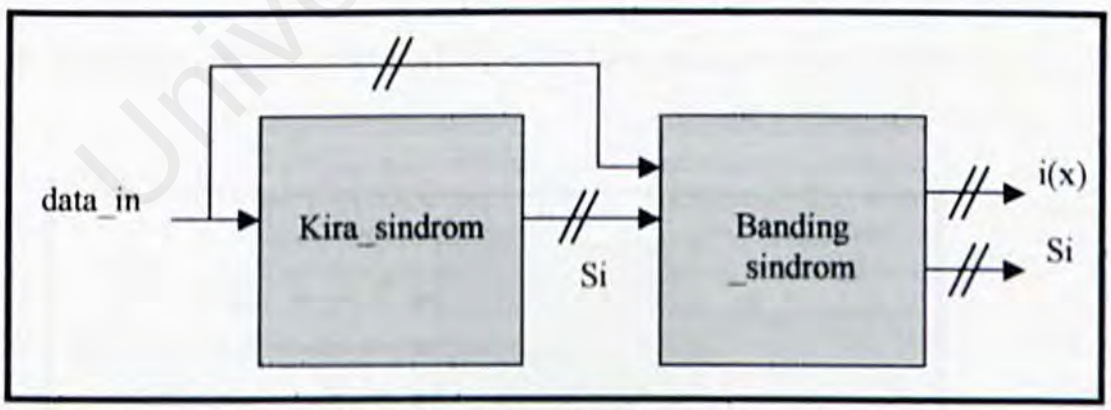


Rajah 4.17 Rekabentuk bagi modul Penyahkodan

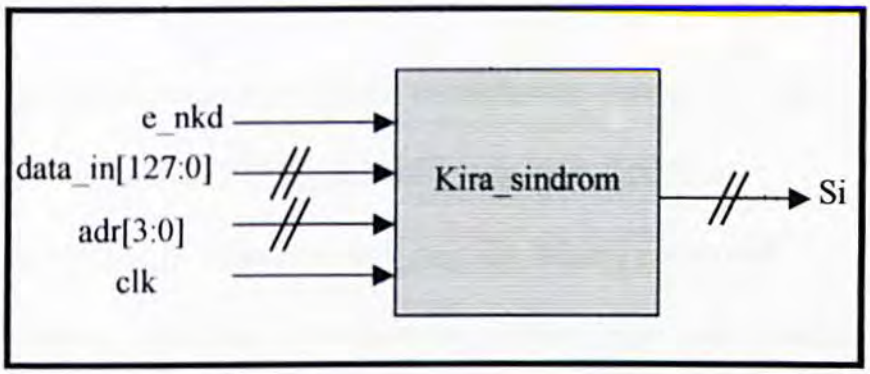
Selain itu, rekabentuk penyahkod juga turut memasukkan 3 ingatan SRAM1, SRAM2 dan SRAM3 bagi tujuan penyimpanan nilai Si, L(x) dan Xi yang akan dicapai oleh modul-modul lain yang memerlukannya. Ini bermakna modul-modul yang ingin menggunakan nilai-nilai tersebut sebagai input perlu membuat capaian daripada SRAM1, SRAM2 dan SRAM3.

SUBMODUL KIRA SINDROM

Submodul Kira Sindrom adalah terbahagi kepada dua iaitu Kira_sindrom dan Banding_sindrom. Tujuan diwujudkan dua submodul ini adalah bagi membuat pengecualian daripada operasi lain sekiranya jujukan kata kod yang diterima tidak mengandungi ralat di mana nilai Si adalah sifar. **Rajah 4.18** menunjukkan hubungan kedua-dua submodul tersebut manakala **rajah 4.19** dan **4.20** menunjukkan rajah terperinci bagi setiap modul.



Rajah 4.18 Submodul Kira Sindrom



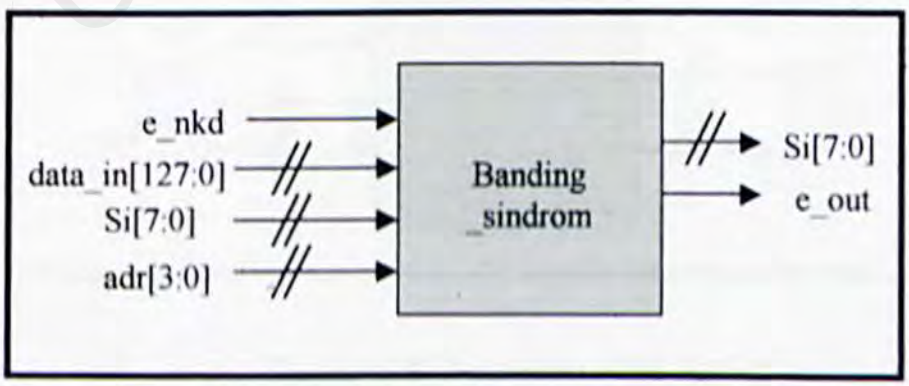
Rajah 4.19 Submodul Kira_sindrom

Port input:

- 1 bit isyarat penyahkodan (e_nkd) : mengaktifkan modul penyahkodan
- 128 bit data (data_in[127:0]) : kata kod yang ingin diperiksa
- 4 bit alamat (adr[3:0]) : alamat bagi blok data yang disimpan dalam RAM
- 1 bit jam (clk) : memberi jam sistem kepada modul

Port output:

- 8 bit data (Si[7:0]) : memberi bilangan ralat yang ada pada kata kod



Rajah 4.20 Submodul Banding_sindrom

Port input:

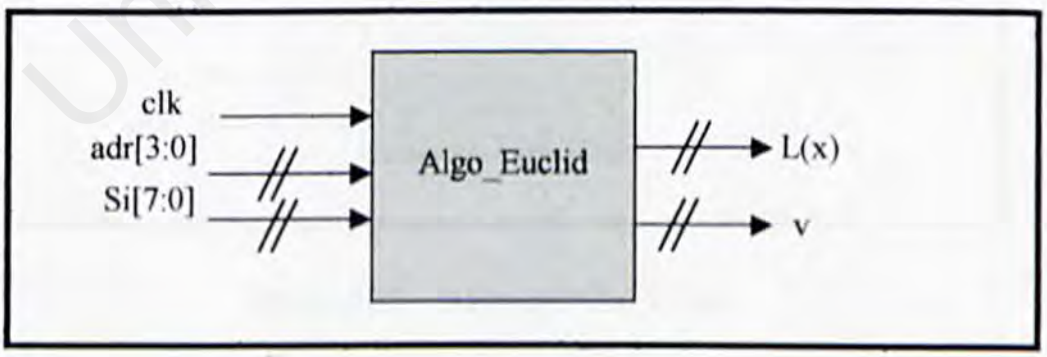
- 1 bit isyarat penyahkodan (e_nkd) : mengaktifkan modul penyahkodan
- 128 bit data (data_in[127:0] : kata kod yang ingin diperiksa
- 8 bit data (Si[7:0]) : Bilangan ralat yang ada dalam jujukan kod
- 4 bit alamat (adr[3:0]) : memberikan alamat bagi blok maklumat yang disimpan dalam RAM

Port output:

- 1 bit kawalan (e_out) : bertujuan untuk mengaktifkan output bagi pengecualian proses

SUBMODUL ALGO_EUCLID

Rajah 4.21 menunjukan struktur bagi submodul Algo_Euclid yang berfungsi untuk mengira polinomial bagi menentukan lokasi ralat dan bilangan ralat.



Rajah 4.21 Submodul Algo_Euclid

Port input:

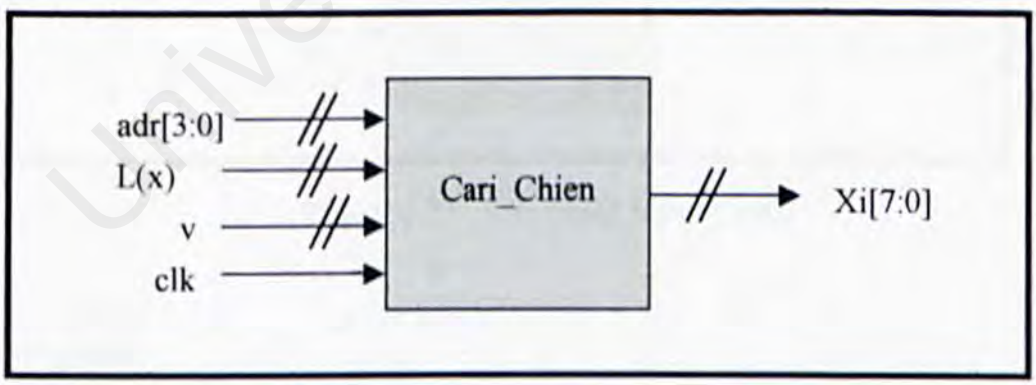
- 1 bit jam (clk) : menyediakan jam sistem kepada modul
- 4 bit alamat (adr[3:0]) : alamat untuk nilai Si yang disimpan dalam SRAM1
- 8 bit data (Si[7:0]) : nilai Si yang dibaca dari SRAM1

Port output:

- L(x) : Polinomial yang digunakan bagi menentukan kedudukan ralat
- 8 bit data (v[7:0]) : memberi nilai bagi bilangan ralat

SUBMODUL CARI_CHIEN

Rajah 4.22 menunjukkan struktur bagi submodul Cari_Chien yang beroperasi bagi menentukan lokasi dan kedudukan ralat yang ada pada jujukan kod.



Rajah 4.22 Submodul Cari_Chien

Port input:

- 4 bit alamat (adr[3:0]) : alamat untuk mencapai L(x) daripada SRAM2

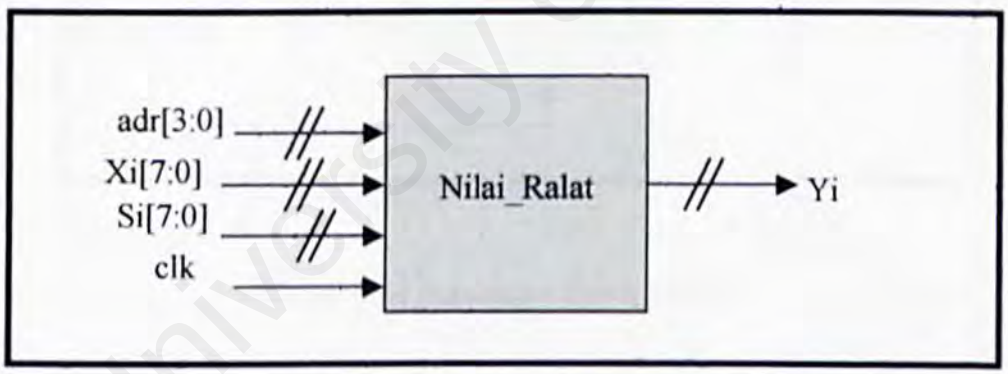
- v : jumlah ralat
- L(x) : Polinomial kedudukan ralat yang dicapai daripada SRAM2
- 1 bit jam (clk) : menyediakan modul dengan jam sistem

Port output:

- 8 bit data (Xi[7:0]) : Lokasi ralat

SUBMODUL NILAI_RALAT

Rajah 4.23 menunjukkan submodul Nilai_ralat yang beroperasi bagi megira nilai ralat bagi setiap kedudukan yang telah dikenalpasti.



Rajah 4.23 Submodul Nilai_Ralat

Port input:

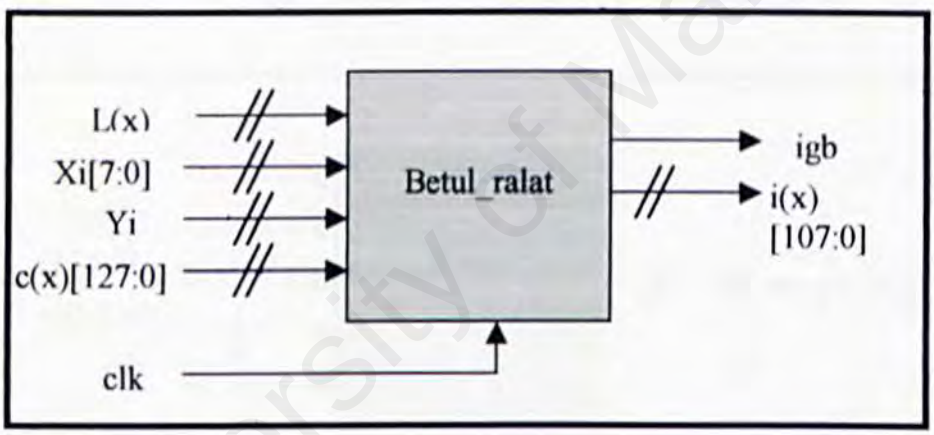
- 4 bit alamat (adr[3:0]) : alamat Si yang disimpan di SRAM1
- 8 bit data (Si[7:0]) : nilai Si yang dibaca dari SRAM1
- 8 bit data (Xi[7:0]) : Lokasi ralat pada jujukan kod
- 1 bit jam (clk) : menyediakan modul dengan jam sistem

Port output:

- Yi : nilai ralat

SUBMODUL BETUL_RALAT

Rajah 4.24 menunjukkan submodul Betul_ralat yang berfungsi untuk membuat penambahan nilai ralat pada lokasi bit dalam jujukan kod yang diterima bagi menghasilkan jujukan data yang sebenar.



Rajah 4.24 Submodul Betul_Ralat

Port input:

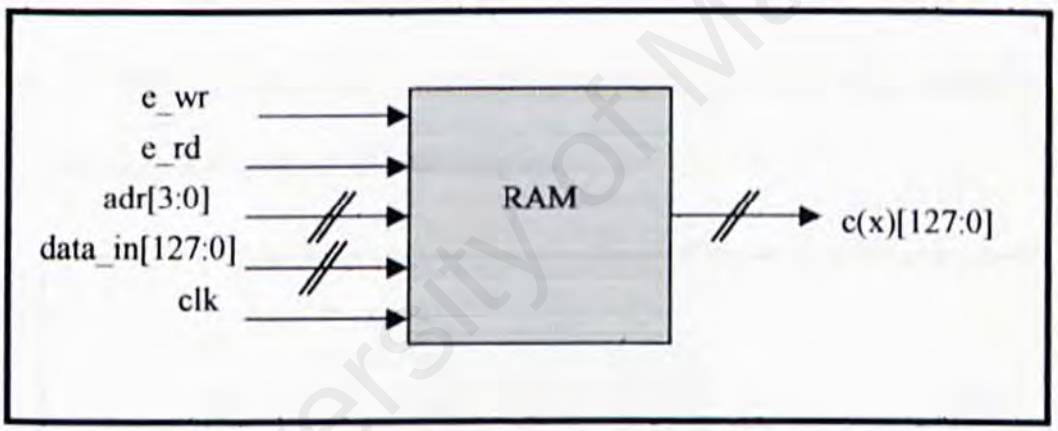
- L(x) : polinomial yang menentukan lokasi ralat
- 8 bit data (Xi[7:0]) : Nilai lokasi ralat
- Yi : Nilai ralat
- 128 bit data (c(x)[127:0]) : kata kod yang ingin dilaksanakan
- 1 bit jam (clk) : menyediakan modul dengan jam sistem

Port output:

- i(x)[107:0] : data output iaitu blok jujukan data asal tanpa pariti
- 1 bit isyarat (igb) : isyarat berjaya atau gagal untuk menentukan bendera

SUBMODUL RAM

Submodul RAM dalam **rajah 4.25** berfungsi untuk menyimpan jujukan kod data yang hendak dikodkan sementara pengiraan-pengiraan oleh modul lain siap dibuat.



Rajah 4.25 Submodul RAM

Port input:

- 1 bit isyarat (e_wr) : memberi mod tulis kedalam ruang ingatan RAM
- 1 bit isyarat (e_rd) : memberi mod baca kedalam ruang ingatan RAM
- 4 bit alamat (adr[3:0]) : alamat untuk diperuntukkan atau mencapai data_in
- 128 bit data (data_in[127:0]) : kata kod input bagi RAM
- 1 bit jam (clk) : menyediakan modul dengan jam sistem

Port output:

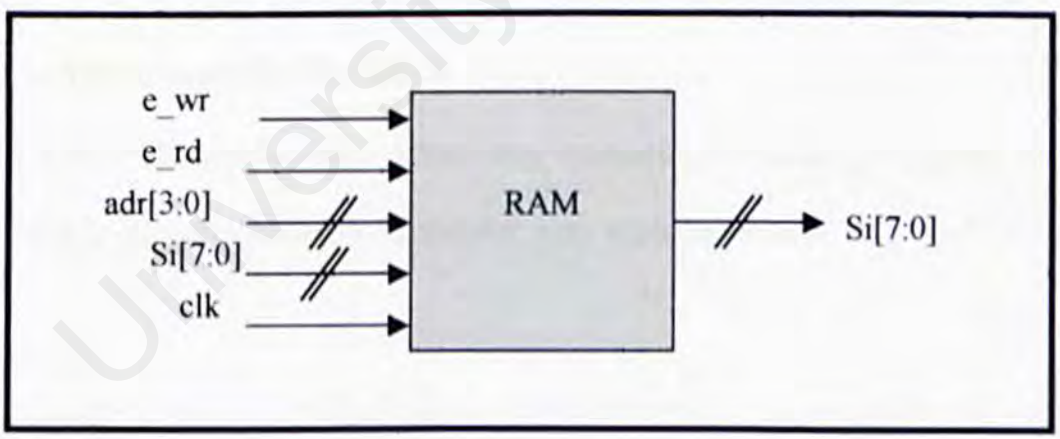
- 108 bit data (i(x)[107:0]) : kata kod yang dibaca daripada RAM

SUBMODUL SRAM1, SRAM2 DAN SRAM3

Pada dasarnya ketiga-tiga modul storan ini berfungsi dengan cara yang sama kecuali pada jenis data yang ingin disimpan iaitu:

- SRAM1 : Nilai Si
- SRAM2 : Polinomial L(x)
- SRAM3 : Lokasi Xi

Rajah 4.26 menunjukkan struktur bagi submodul SRAM1 yang asasnya boleh diguna untuk submodul SRAM yang lain.



Rajah 4.26 Submodul SRAM1

Port input:

- 1 bit isyarat (e_wr) : memberi mod tulis kedalam ruang ingatan RAM
- 1 bit isyarat (e_rd) : memberi mod baca kedalam ruang ingatan RAM

- 4 bit alamat (adr[3:0]) : alamat untuk diperuntukkan atau mencapai data_in
- 8 bit data (Si[7:0]) : nilai sindrom input bagi SRAM1
- 1 bit jam (clk) : menyediakan modul dengan jam sistem

Port output:

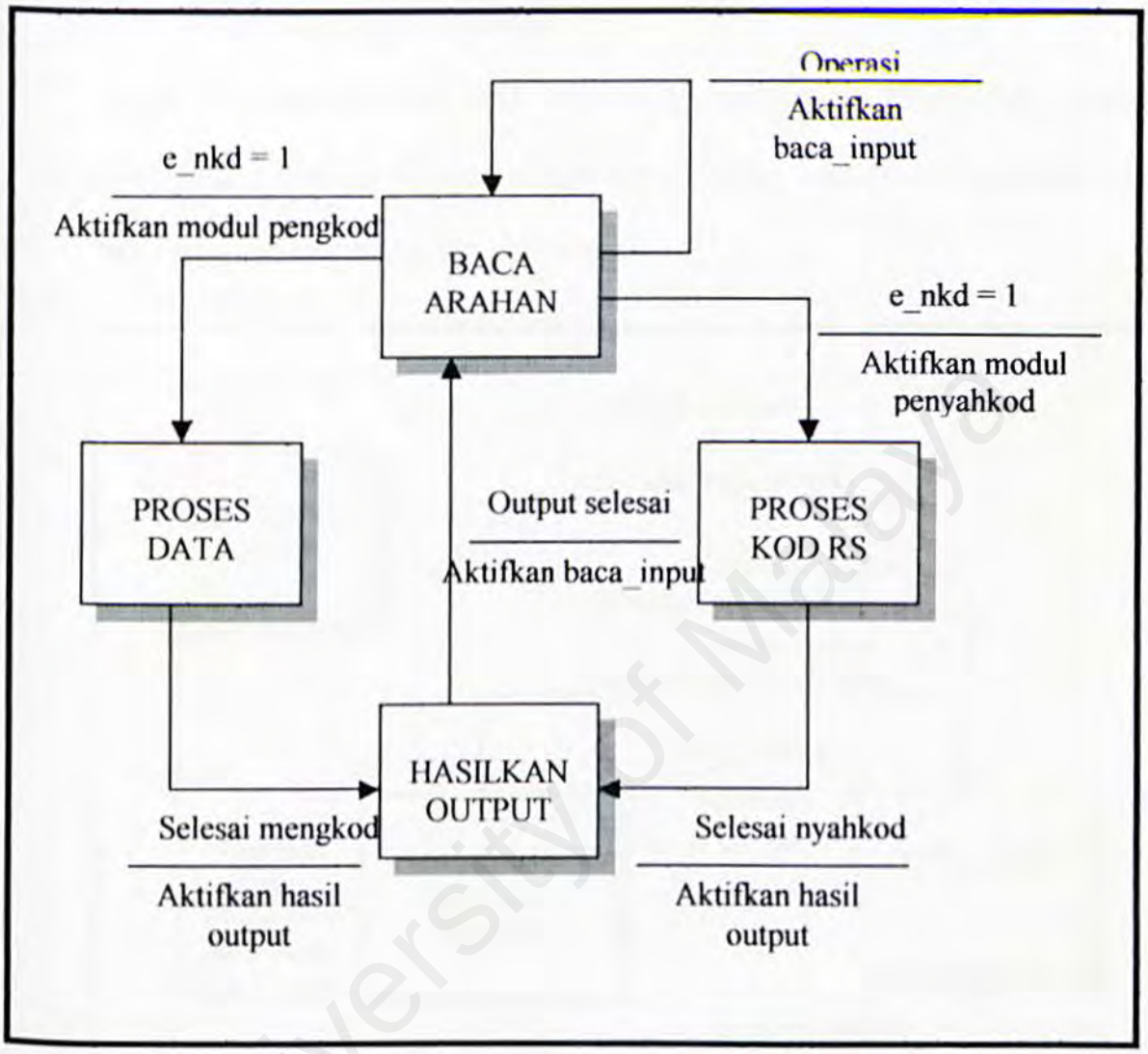
- 8 bit data (Si[7:0]) : nilai Si yang dibaca daripada SRAM1

4.7 RAJAH STATUS REKABENTUK PERKAKASAN

Rajah status dicipta bagi memudahkan pemahaman operasi-operasi perkakasan semasa fasa pengkodan dan pengujian nanti. Setiap modul diteliti operasinya daripada modul utama hinggalah kepada submodul-submodul yang lain.

4.7.1 OPERASI KESELURUHAN

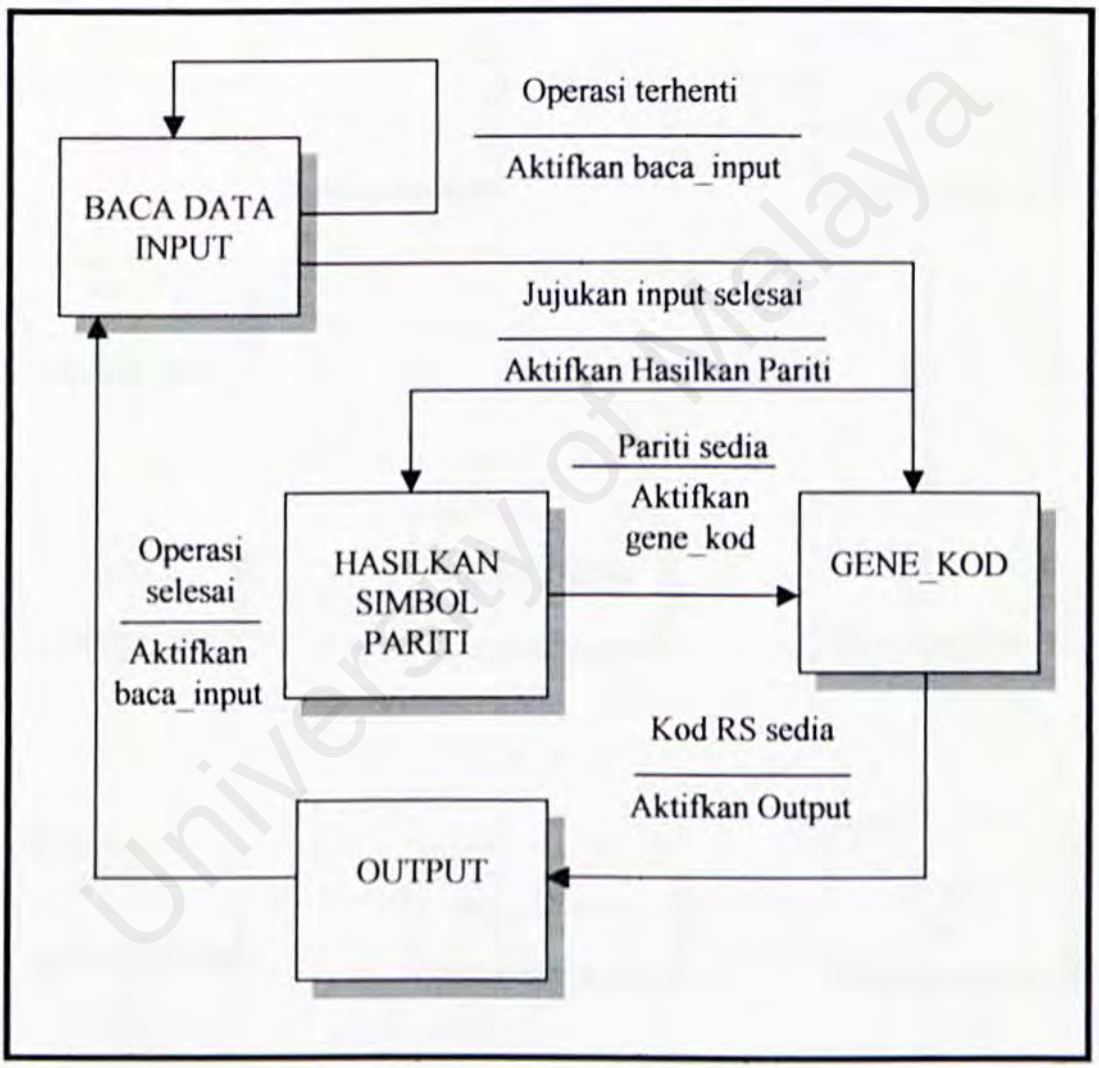
Rajah 4.27 menunjukkan rajah status bagi keseluruhan operasi perkakasan ECC berpandukan kepada spesifikasi rekabentuk yang telah dinyatakan.



Rajah 4.27 Rajah status keseluruhan perkakasan ECC

4.7.3 PENGHASILAN KOD RS

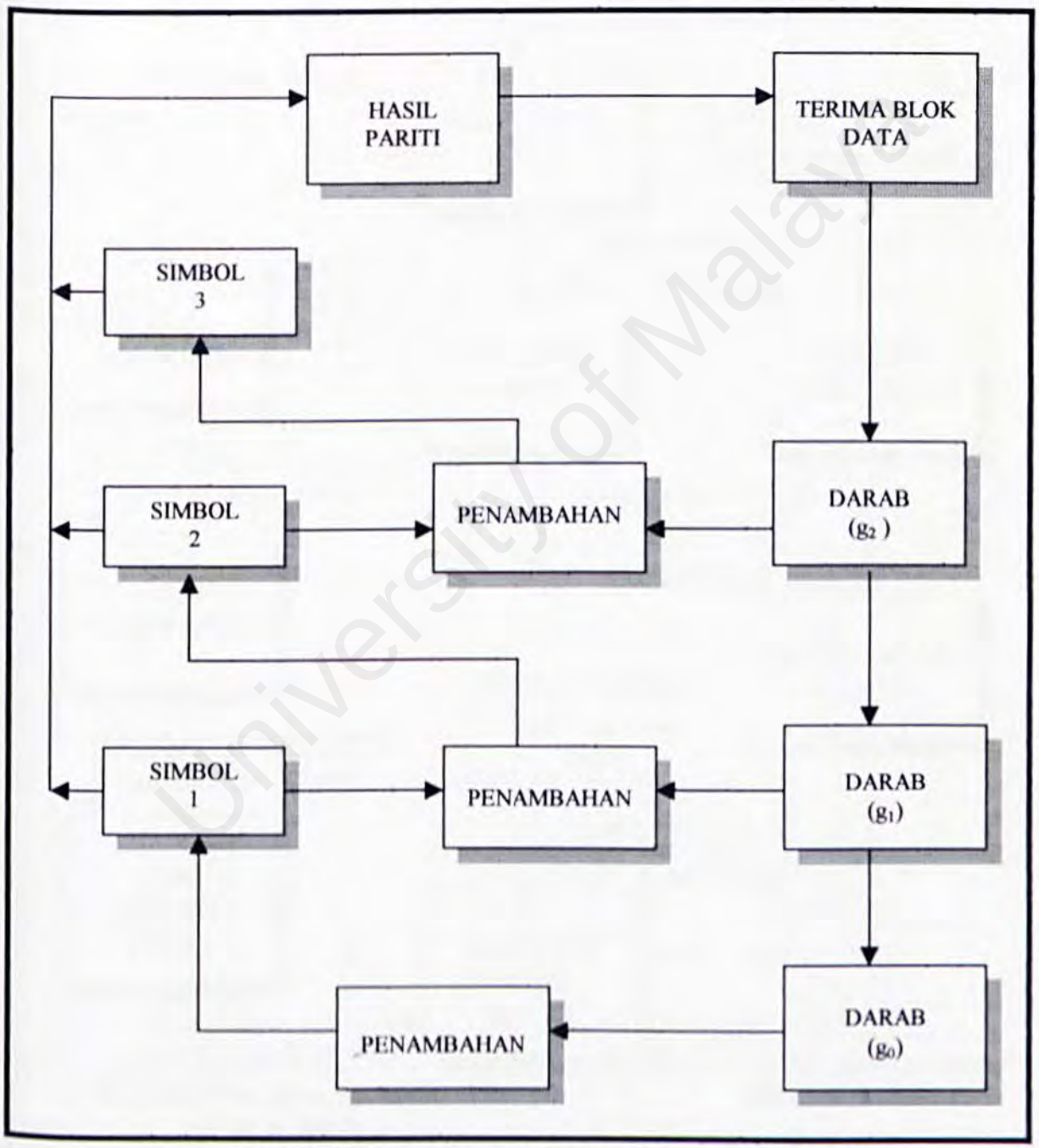
Rajah 4.28 menunjukkan rajah status bagi pengkod RS berdasarkan kepada rekabentuk terdahulu. Operasi pengkodan terbahagi kepada dua bahagian iaitu penjanaan kod dan penjanaan simbol pariti.



Rajah 4.28 Rajah Status Pengkod RS

4.7.4 PENGHASILAN SIMBOL PARITI

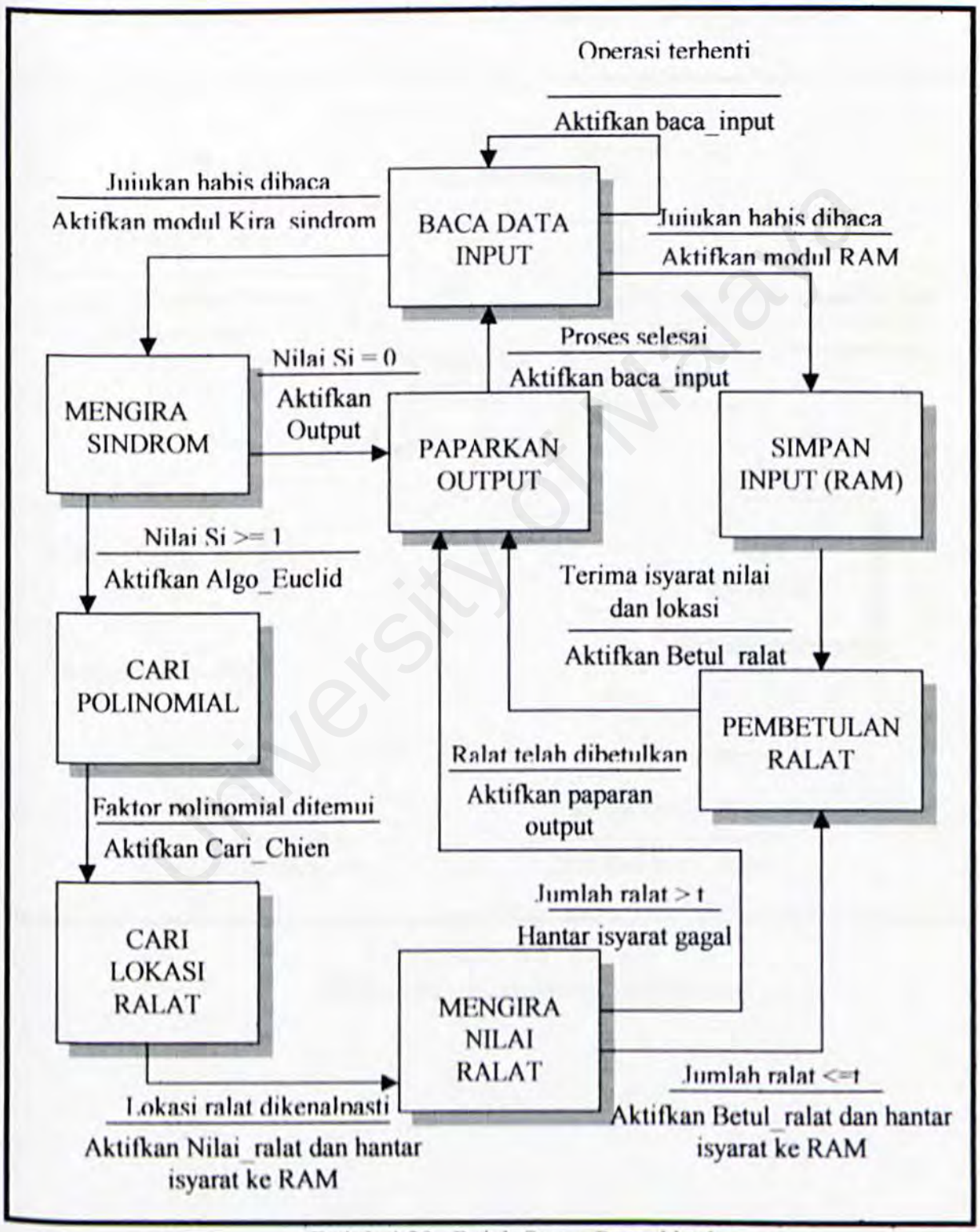
Rajah 4.29 menunjukkan rajah status penjanaan simbol pariti bagi penyahkod RS.



Rajah 4.29 Rajah status bagi penjanaan simbol pariti

4.7.5 PENYAHKODAN KATA KOD RS

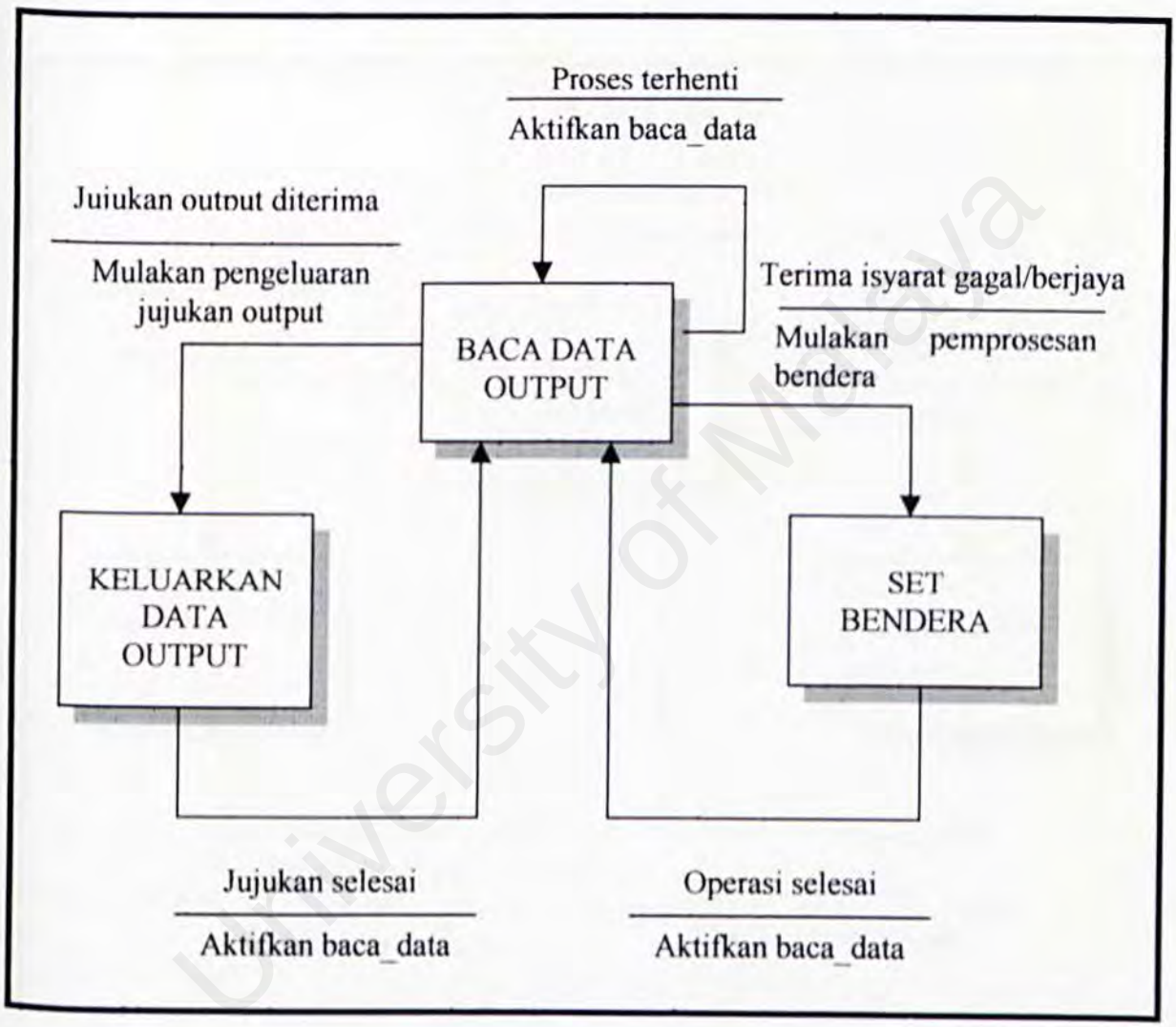
Rajah 4.30 menunjukkan rajah status bagi struktur penyahkod.



Rajah 4.30 Rajah Status Penyahkodan

4.7.6 PENGHASILAN OUTPUT

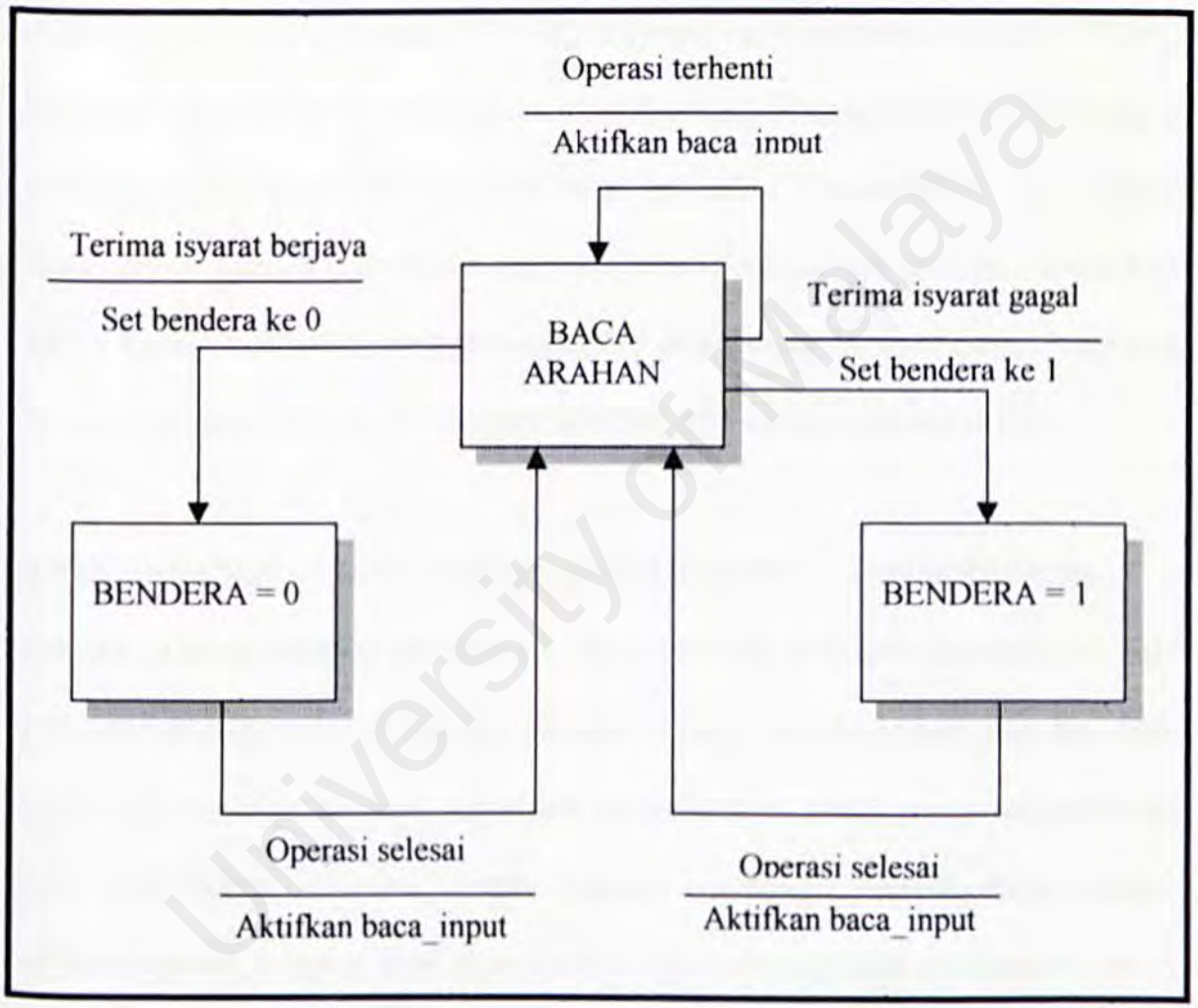
Rajah 4.31 menunjukkan rajah status bagi penghasilan output oleh penyahkodan yang terbahagi kepada 2 bahagian iaitu output data dan output bendera.



Rajah 4.31 Rajah Status Unit Output

4.7.7 PENENTUAN ISYARAT BENDERA

Rajah 4.32 menunjukkan rajah status bagi penentuan isyarat bendera yang dihasilkan oleh penyahkod.



Rajah 4.32 Rajah Status Set Bendera

4.8 HASIL YANG DIJANGKAKAN

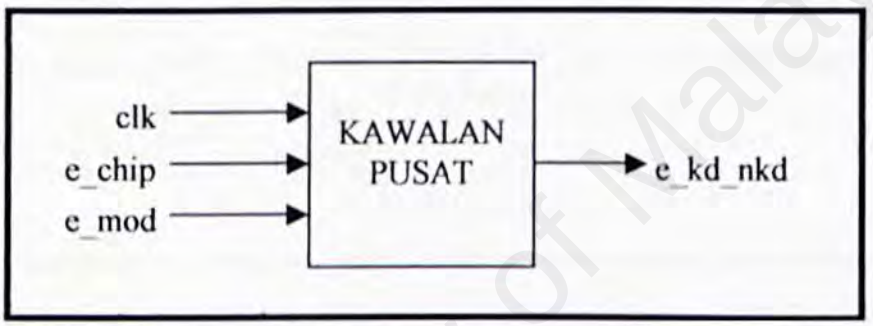
Secara amnya, suatu kod RS boleh membetulkan sehingga R/2 ralat simbol di mana R adalah merupakan bilangan simbol pariti dalam kata kod tersebut. Oleh kerana secara amnya kod RS diterangkan sebagai RS(N,N-R), maka bilangan ralat yang boleh dibetulkan oleh kod ini adalah [N-(N-R)]/2. Untuk itu, dijangkakan kod RS(255,249) yang akan digunakan akan berupaya membetulkan ralat sehingga [16-(16-13)]/2 iaitu 6 bilangan ralat. Keupayaan kawalan ralat ini boleh diperhebatkan lagi dengan menggunakan kaedah pemadaman ralat iaitu satu teknik yang membantu menentukan lokasi sesuatu ralat tanpa menggunakan salah satu daripada simbol pariti. Suatu kod RS yang menggunakan teknik ini boleh membetulkan sehingga R jumlah ralat.

4.9 KEBARANGKALIAN KESILAPAN DALAM KOD REED-SOLOMON

Penyahkod boleh membuat kesilapan dalam pembetulan ralat sekiranya bilangan ralat mencapai sehingga separuh daripada bilangan karakter Reed-Solomon yang ditambah. Dalam kebanyakkan keadaan, algoritma Reed-Solomon boleh mengesan kehadiran ralat berlebihan ini menerusi panapis dalaman. Walaubagaimanapun, dalam kes-kes terpencil, mesej mungkin ditafsirkan kepada kata kod yang berlainan daripada mesej asal. Ini yang dikatakan sebagai pembetulan ralat yang silap.

5.1 MODUL KAWALAN UTAMA

Memberi kawalan ke atas keseluruhan cip ECC. Akan diaktifkan apabila menerima isyarat pengaktifan. Menerima satu isyarat dan akan menentukan sama ada ingin melaksanakan mod pengkodan ataupun mod penyahkodan. Sekiranya isyarat yang dihasilkan oleh modul ini adalah '1' maka mod pengkodan akan diaktifkan manakala jika isyarat '0' yang dihasilkan, maka mod penyahkodan yang akan diaktifkan.



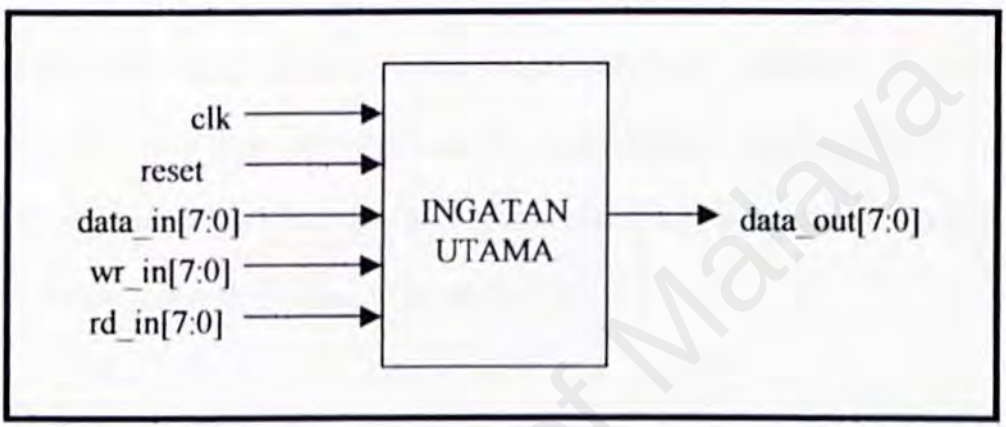
Kod VHDL bagi modul kawalan pusat ini dapat dilihat pada lampiran I.

Hasil simulasi bagi kawalan pusat adalah seperti berikut:

5.2 MODUL INGATAN UTAMA

Berfungsi sebagai satu unit ingatan yang akan menyimpan jujukan data yang sedang diproses sehingga pemprosessan data tersebut selesai dibuat. Tujuan adanya modul ini adalah supaya data yang sedang diproses boleh dicapai pada bila-bila masa semasa pemprosesan data sedang dilakukan. Ini juga dapat mengelakkan kehilangan data yang mungkin berlaku disebabkan masa pemprosesan yang mungkin mengalami lengah.

Dalam rekabentuk asal, ingatan sepatutnya berada pada setiap entiti pemprosesan bagi menyimpan setiap data yang telah diproses tetapi dalam fasa pembangunan idea tersebut telah diubahsuai dengan hanya menggunakan satu ingatan utama yang akan menyimpan keseluruhan data yang sedang diproses.



Kod VHDL bagi modul ingatan utama ini dapat dilihat dalam **Lampiran II.**

Hasil simulasi bagi ingatan utama adalah seperti berikut:

5.3 MODUL PENGKODAN

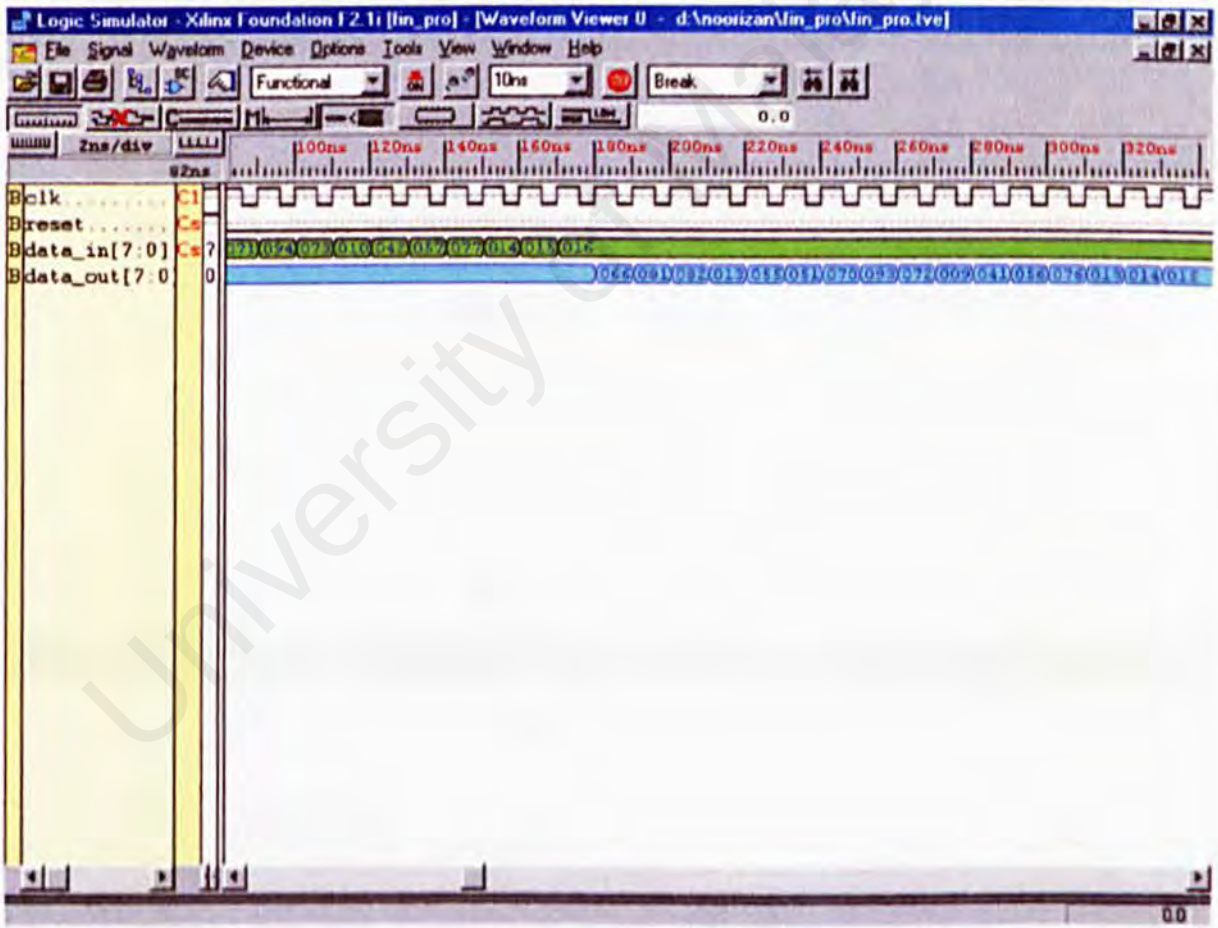
Oleh kerana pembangunan kod bagi perkakasan yang ingin dibina memerlukan ruang ingatan yang besar bagi mempercepatkan proses sintesis kod manakala kemudahan sedia ada tidak dapat mengatasi masalah tersebut maka, rekabentuk di dalam bab 4 terpaksa diubahsuai bagi membolehkan kod di sintesis dengan lebih laju semasa proses pembangunan perkakasan sedang dijalankan. Modul pengkodan asal yang terdiri daripada 3 entiti pemprosesan simbol pariti dan satu entiti final telah dipecahkan menjadi entiti-entiti dengan saiz yang lebih kecil supaya proses sintesis kod dapat

dipercepatkan. Walaubagaimanapun, sekiranya kemudahan sedia ada dapat dipertingkatkan, dipercayai, entiti-entiti yang menjanakan simbol-simbol pariti dapat disatukan menjadi satu entiti pemprosesan yang lebih besar. Rajah bagi rekabentuk yang dibangunkan dapat dilihat pada **Lampiran III.**

Seperti yang telah dapat dilihat, modul pengkodan terdiri daripada 11 entiti yang dijanakan secara berasingan dan akhirnya akan digabungkan bagi menghasilkan jujukan data 19 simbol yang terdiri daripada 16 simbol data dan 3 simbol pariti. Hasil simulasi bagi kesemua entiti-entiti ini adalah seperti berikut:

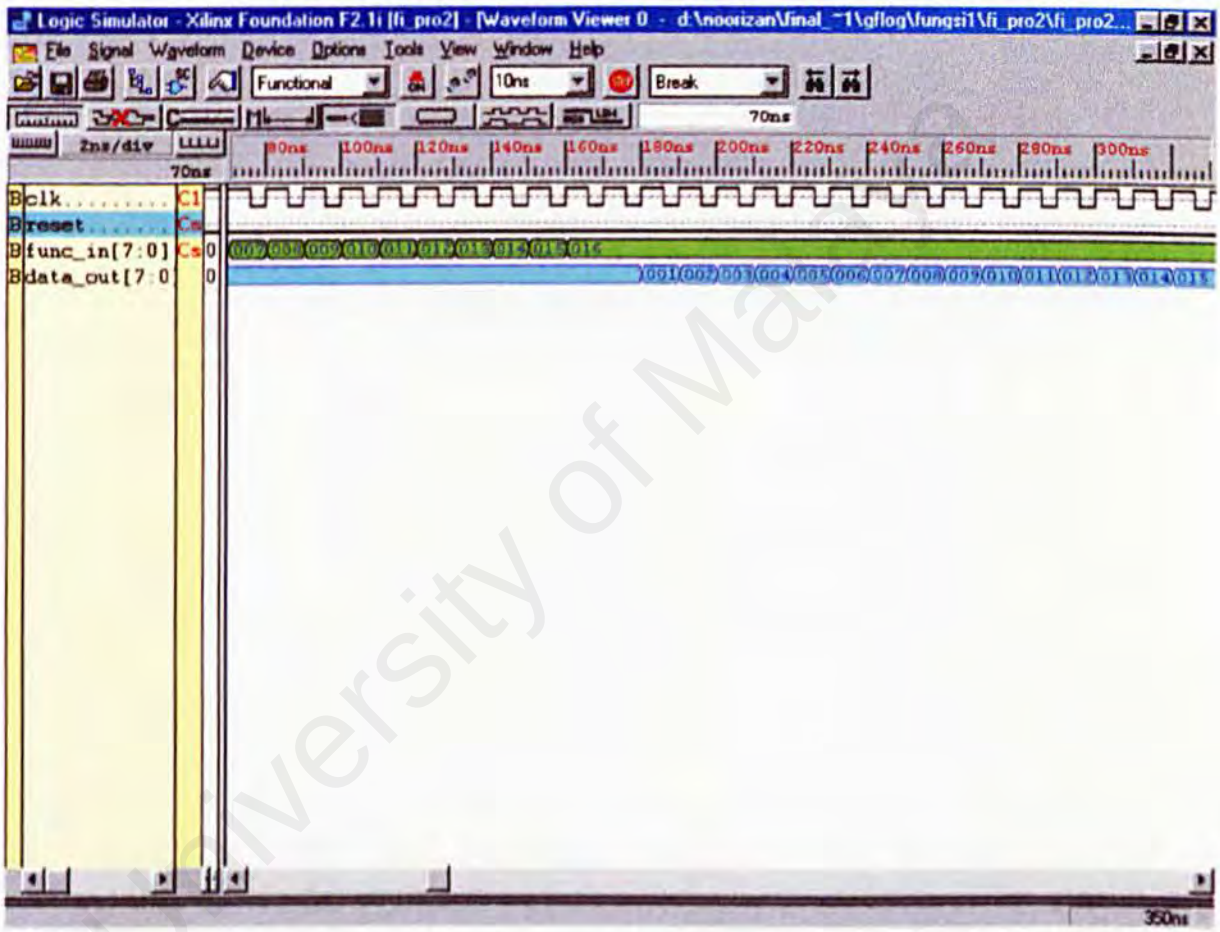
5.3.1 ENTITI GFLOG_DATA

Entiti yang menghasilkan nilai gflog bagi data yang diterima kerana pendaraban bagi nombor-nombor binari adalah perlu mengikut konsep ruang finit dimana suatu jadual akan dirujuk sebelum nombor didarabkabkan dan hasil darab akan rujuk semula bagi mendapatkan nilai gfilog. Nilai gflog bagi data yang diperoleh akan dihantar ke entiti darab1 dan entiti darab2.



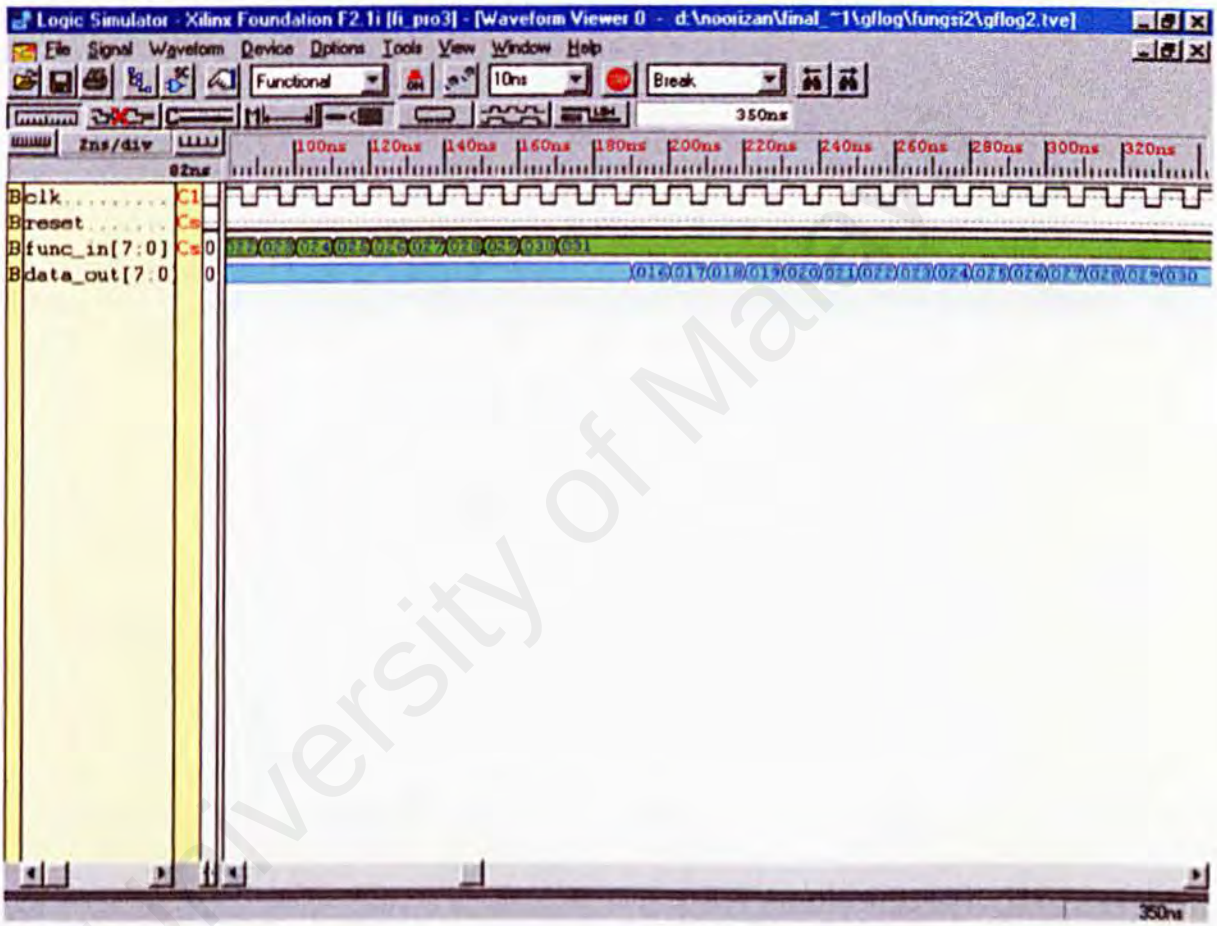
5.3.2 ENTITI GFLOG_FUNC1

Entiti ini bagi mendapatkan nilai gflog untuk jujukan fungsi pertama. Nilai yang diperoleh akan dihantar ke entiti darab1.



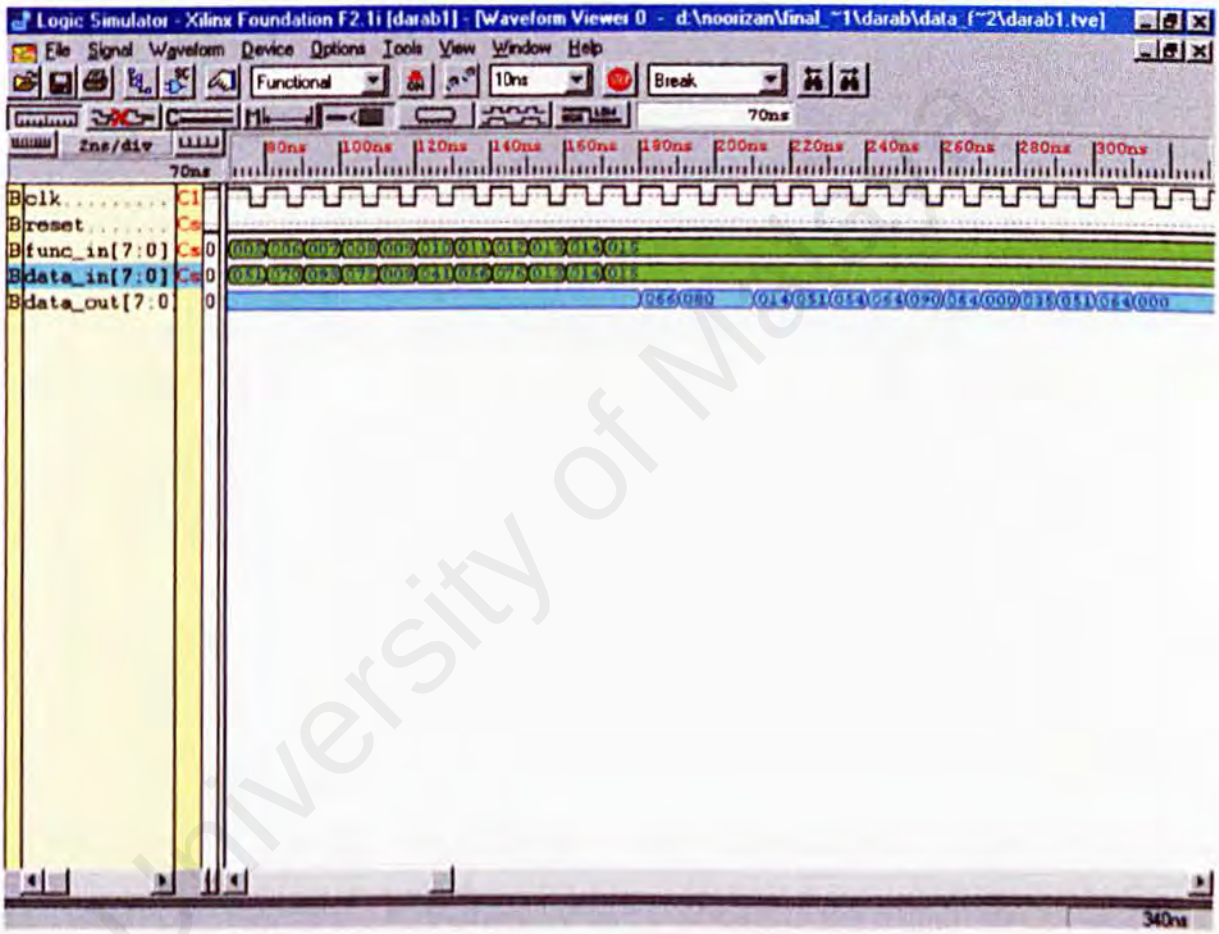
5.3.3 ENTITI GFLOG_FUNC2

Entiti ini adalah untuk mendapatkan nilai gflog nagi jujukan fungsi kedua. Nilai yang diperoleh akan dihantar ke entiti darab2.



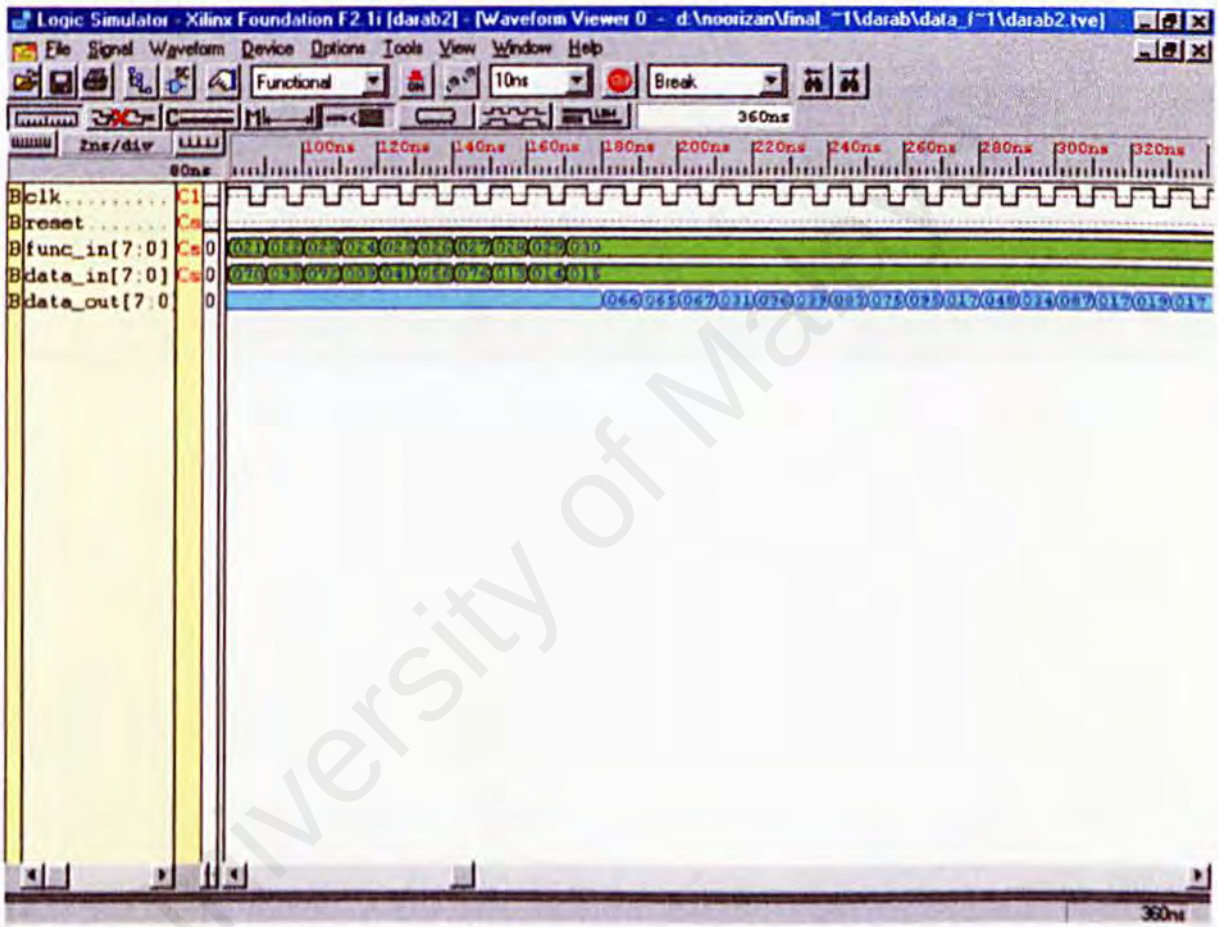
5.3.4 ENTITI DARAB1

Entiti yang membuat pendaraban nilai gflog data dengan gflog fungsi 1. Nilai yang diperoleh akan dihantar ke entiti gfilog1.



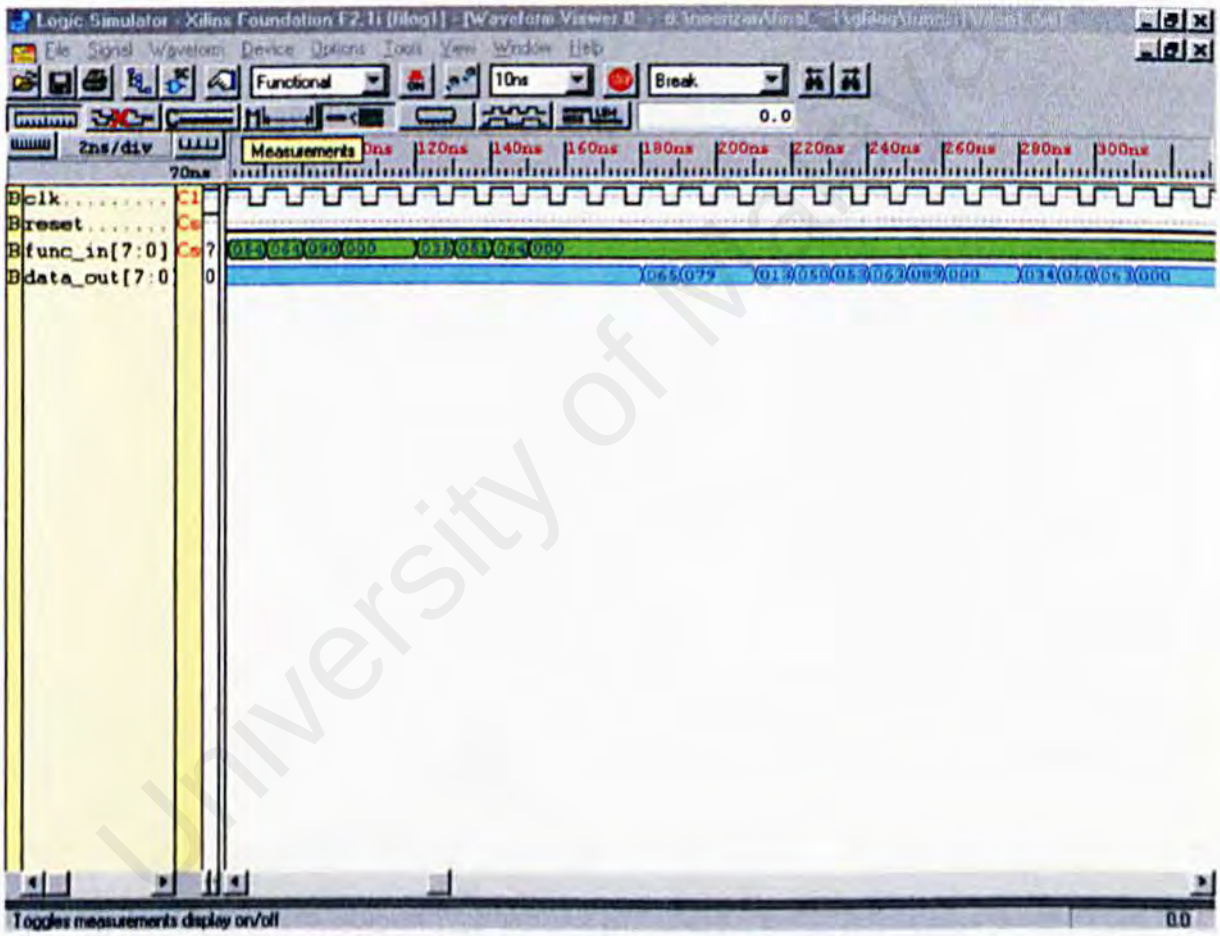
5.3.5 ENTITI DARAB2

Entiti yang membuat pendaraban nilai gflog data dengan gflog fungsi 2. Nilai yang diperoleh akan dihantar ke entiti gfilog2.



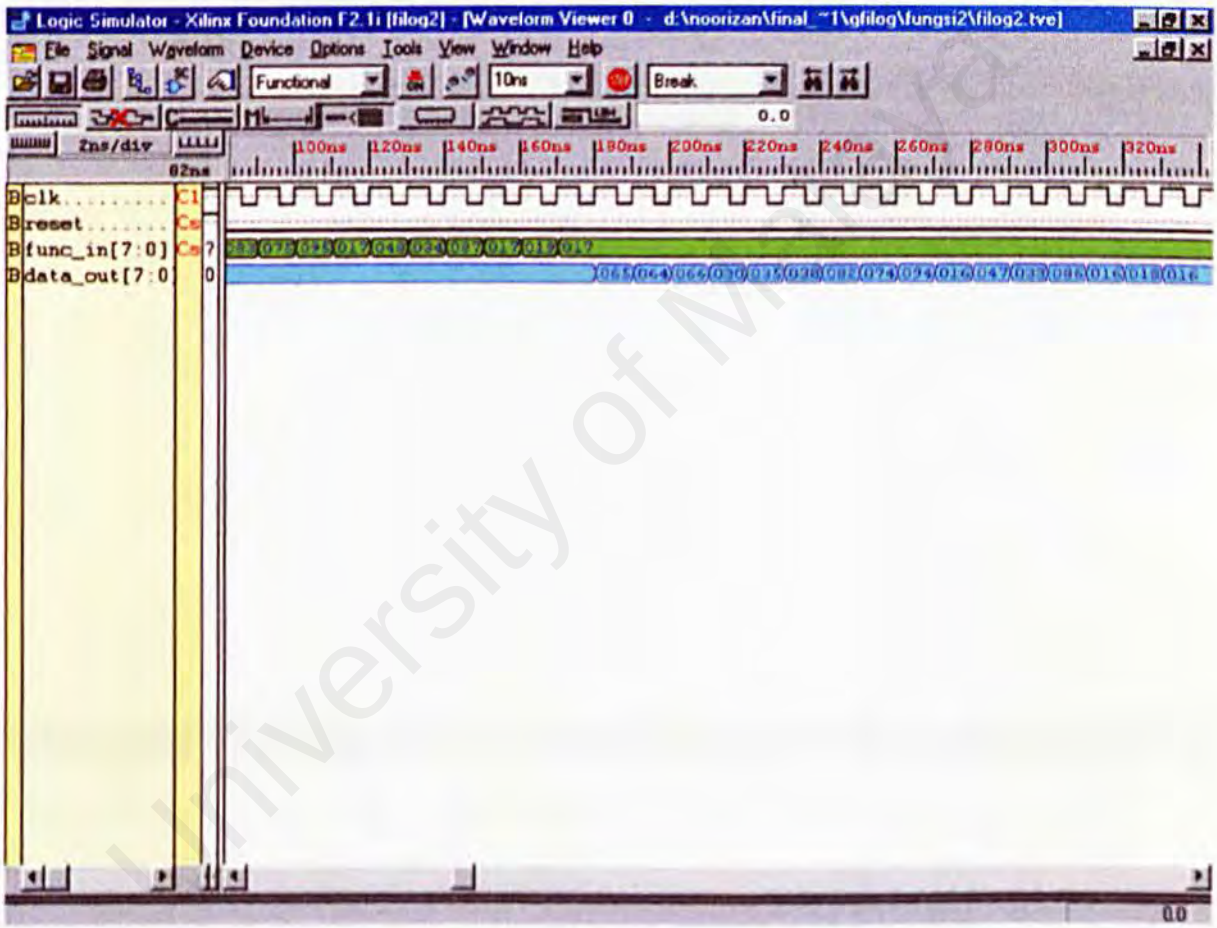
5.3.6 ENTITI GFILOG1

Entiti ini bagi mendapatkan nilai gfilog untuk jujukan fungsi1 yang telah didarabkan dengan nilai gflog data. Nilai yang diperoleh akan dihantar ke entiti pariti2.



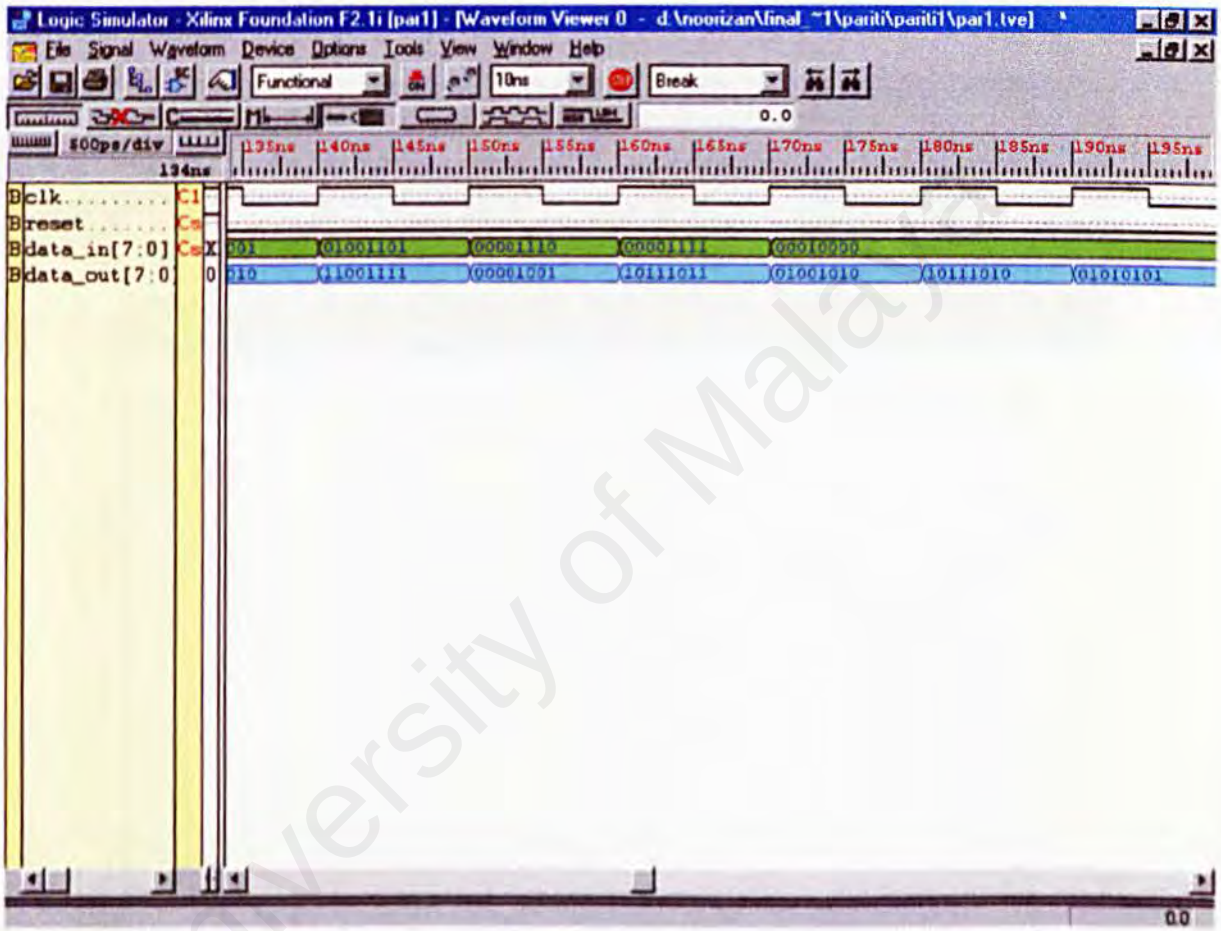
5.3.7 ENTITI GFILOG2

Entiti ini bagi mendapatkan nilai gfilog untuk jujukan fungsi2 yang telah didarabkan dengan nilai gflog data. Nilai yang diperoleh akan dihantar ke entiti pariti3.



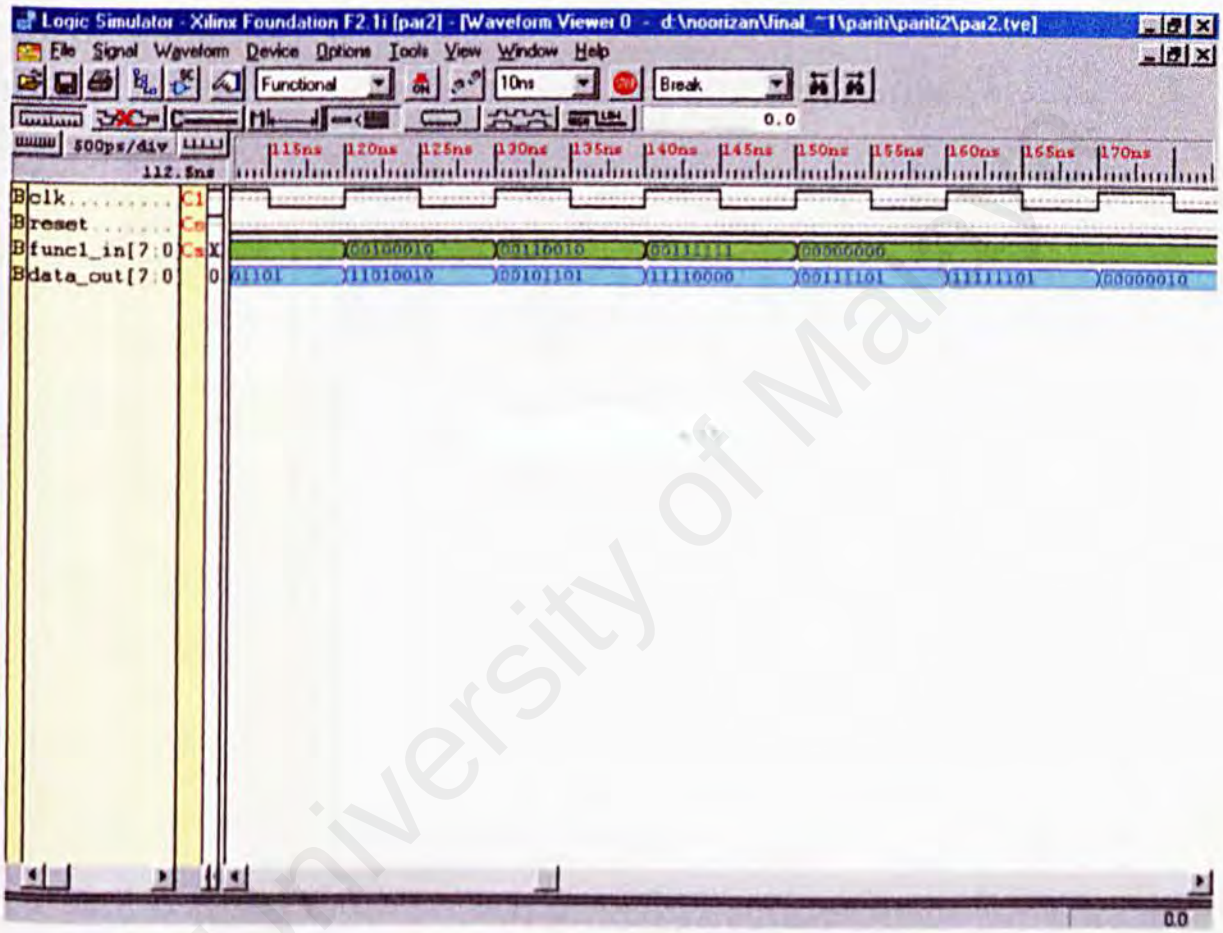
5.3.8 ENTITI PARITI 1

Mendarabkan nilai-nilai data bagi memperoleh satu nilai yang akan iaitu nilai bagi simbol pariti pertama. Nilai ini akan dihantar ke entiti gabung.



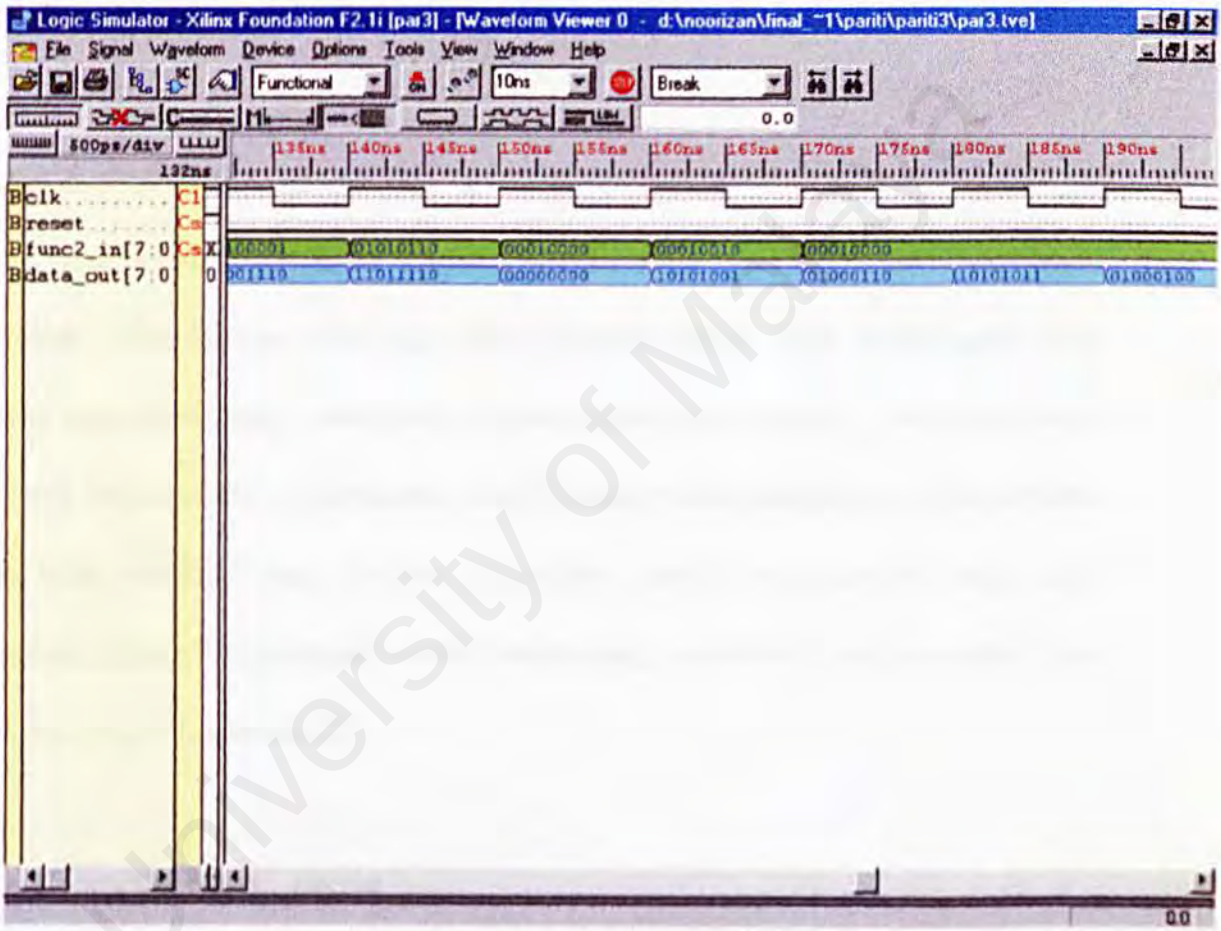
5.3.9 ENTITI PARITI 2

Mendarabkan nilai-nilai gfilog fungsi 1 bagi memperoleh satu nilai yang akan iaitu nilai bagi simbol pariti kedua. Nilai ini akan dihantar ke entiti gabung.



5.3.10 ENTITI PARITI 3

Mendarabkan nilai-nilai gfilog fungsi 2 bagi memperoleh satu nilai yang akan iaitu nilai bagi simbol pariti ketiga. Nilai ini akan dihantar ke entiti gabung.



5.3.11 ENTITI GABUNG

Entiti ini akan menerima 3 simbol pariti dan jujukan 16 simbol data. Masukan-masukan ini akan digabungkan bagi membentuk jujukan data 19 simbol sebagai data output mod pengkodan yang akan dihantar ke penerima jika suatu komunikasi berlaku.

5.4 MODUL PENYAHKODAN

Masalah yang sama juga telah berlaku untuk modul penyahkodan yang ingin dibangunkan. Oleh kerana data yang ingin diproses adalah besar maka masa yang diperlukan bagi memproses setiap kod aturcara adalah amat panjang. Bagi mengatasi masalah ini, maka modul penyahkodan telah dibahagi-bahagikan supaya pemprosesan menjadi lebih singkat. Bagi melihat rekabentuk modul penyahkodan yang telah dibangunkan, sila ke **Lampiran IV.** Hasil simulasi bagi entiti-entiti yang tersenarai juga dapat dilihat dalam **Lampiran IV.**

5.4.1 ENTITI SEL_16SIM

Entiti Sel_16Sim adalah entiti yang akan memilih 16 simbol data terakhir daripada 19 jujukan data yang diterima untuk dihantar ke entiti Kira_Par supaya 16 data ini dapat dikira paritinya.

5.4.2 ENTITI KIRA_PAR

Entiti ini sepatutnya akan menerima 16 simbol data dan seterusnya akan mengira pariti bagi data-data tersebut. Entiti kira par adalah terdiri daripada 10 entiti yang akan menghasilkan 3 simbol pariti bagi data yang akan dihantar ke entiti kesan_sindrom.

5.4.3 ENTITI KESAN_SINDROM

Entiti ini akan menerima 3 simbol pariti yang telah dikira untuk dibandingkan dengan pariti-pariti yang diterima nbersama jujukan data. Jika hasil sindrom adalah isyarat '0', ini bermakna jujukan data yang diterima bebas dari ralat.

5.4.4 ENTITI LOKASI

Entiti ini sepatutnya membuat pengiraan bagi mengesan lokasi simbol yang ralat. Ia sepatutnya menyatakan 3 lokasi. Walaupun hanya ada 1 atau 2 simbol yang hilang, ia akan menganggap lokasi perlu sentiasa tiga.

5.4.5 ENTITI SEL_DATA

Entiti sel_data berfungsi untuk membuat pemilihan 16 data yang akan digunakan untuk membuat pengiraan bagi data yang hilang.

5.4.6 ENTITI KIRA_DATA

Entiti ini akan membuat pengiraan bagi mendapatkan semula 3 data yang hilang daripada 16 data yang telah dipilih.

5.4.7 ENTITI GABUNG_DATA

Setelah 3 data yang hilang berjaya diperolehi, 3 data tersebut akan dimasukkan ke dalam jujukan data yang diterima bagi mendapatkan jujukan data yang sebenar.

5.5 PENGUJIAN

Untuk menguji sama ada perkakasan yang dibangunkan benar-benar dapat berfungsi seperti yang dikehendaki, saya telah menjalankan 5 set ujian yang mengambil kira kesemua kedudukan data yang mungkin iaitu data paling awal, data paling akhir, data yang hampir dengan data awal, data yang hampir dengan data akhir dan data yang berada di pertengahan. Daripada set-set data yang telah dipilih, ia akan digunakan sebagai data masukan pada entiti-entiti diatas dan kesemua data tersebut dapat dihasilkan seperti yang dikehendaki.

6.1 MASALAH-MASALAH YANG WUJUD

Walaupun suatu perancangan yang rapi telah dibuat semasa fasa-fasa sebelum perkakasan sebenar dibangunkan, beberapa masalah telah dikesan semasa perkakasan ingin dibangunkan. Walaubagaimanapun beberapa masalah telah dapat diatasi dengan menggunakan pelbagai pendekatan termasuk membuat sedikit pengubahsuaian ke atas perancangan terdahulu.

6.1.1 MASA PERLAKSAAN KOD YANG PANJANG

Semasa dalam fasa rekabentuk, suatu rekabentuk telah di reka untuk dibangunkan dalam fasa pembangunan. Malangnya, apabila program dibangunkan mengikut rekabentuk asal program tersebut tidak dapat dilaksanakan kerana data masukan yang banyak telah menyebabkan entiti-entiti dalam rekabentuk asal memerlukan masa yang terlalu lama untuk dilaksanakan. Oleh kerana data masukkan perlu dimasukkan ke dalam tatasusunan di dalam program, kadangkala gegelung yang diperlukan untuk pemprosesan data masukan ini melakukan gegelung infinit semasa program hendak dilaksanakan. Gegelung inilah yang telah menyebabkan perlaksanaan program mengambil masa yang lama untuk disintesiskan.

6.1.2 INGATAN KECIL DAN KUASA PEMPROSES

Semasa mula membangunkan perkakasan, disedari bahawa perlaksaan VHDL memerlukan saiz RAM yang besar selain pemproses yang berkuasa. Oleh kerana

kemudahan komputer yang disediakan hanya mempunyai RAM 128 dan pemproses Intel Pentium II, ini telah menyebabkan program yang mengikut rekabentuk asal tidak dapat dilaksanakan kerana sekiranya rekabentuk asal dibangunkan setiap program bakal mempunyai bilangan gegelung yang lebih besar dan ini akan menyebabkan komputer tidak dapat melaksanakan arahan dengan betul. Masalah ini juga telah menyebabkan pemetaan entiti-entiti tidak dapat dilakukan kerana pemetaan akan memerlukan lebih banyak ruang ingatan.

6.1.3 MASALAH PERISIAN XILINX

Perisian Xilinx yang disediakan oleh pihak fakulti hanya boleh digunakan di makmal VLSI. Oleh itu, semua kerja-kerja terpaksa dilakukan hanya apabila makmal tersebut dibuka. Dengan masa penggunaan yang terhad, masa yang diperuntukkan untuk membangunkan perkakasan adalah sangat singkat. Oleh itu, terdapat beberapa masalah lain yang melibatkan teori-teori algoritma gagal diselesaikan.

6.1.4 KEKANGAN BAHAN RUJUKAN

Kod Reed-Solomon yang digunakan dalam perlaksaan pembetulan ralat adalah merupakan kod pembetulan yang jarang digunakan dalam bidang akademik. Oleh sebab itu, saya mempunyai banyak kesulitan dalam mendapatkan rujukan dan bantuan berkenaan dengan teori-teori algoritma yang melibatkan kod ini. Kebanyakkan rujukan yang tersedia untuk kod ini adalah dalam bentuk kertas

kerja-kertas kerja di dalam internet tetapi suatu masalah yang ada ialah, kesemua kertas kerja tersebut tidak menerangkan secara lengkap bagaimana lokasi sesuatu ralat dalam jujukan data yang diterima dapat dikesan.Kertas kerja-kertas kerja tersebut juga lebih memberi tumpuan kepada persamaan-persamaan matematik yang tidak dapat dilaksanakan ke atas data-data binari.

6.1.5 PENJANAAN NILAI DATA DALAM PERKAKASAN

Dalam membuat pengiraan pariti dan pengiraan untuk memperoleh data asal semula jika data hilang semasa penghantaran suatu jujukan nilai-nilai fungsi yang sentiasa tetap setiap kali pengiraan dilakukan perlu dijanakan tetapi penjanaan suatu nilai data di dalam perkakasan sering bermasalah disebabkan oleh kesesuaian litar, masa umpukan dan sebagainya. Oleh itu, nilai fungsi ini gagal dihasilkan di dalam litar.

6.2 PENYELESAIAN MASALAH

Setelah meneliti semua masalah yang timbul, terdapat beberapa masalah yang telah berjaya saya selesaikan.

6.2.1 MENGGUNAKAN PERNYATAAN IF

Setelah meneliti masalah masa perlaksanaan yang terlalu lama, selain daripada memecahkan modul kepada entiti-entiti yang lebih kecil, saya juga telah cuba menggantikan pernyataan gegelung kepada pernyataan if bersyarat. Walaupun

pernyataan if boleh menyebabkan masalah kehilangan data kerana ia menggunakan flip-flop di mana terdapat kemungkinan data yang telah diproses akan disuap kembali ke dalam litar untuk pemprosesan data selanjutnya namun saya telah dapat mengelak dari mengalami masalah tersebut.

6.2.2 PEMECAHAN PROGRAM-PROGRAM BESAR

Seperti yang telah dinyatakan dalam bahagian 6.1, pembangunan program yang besar akan menyebabkan masa pemprosesan menjadi terlalu panjang. Bagi mengatasi masalah ini dan juga masalah keperluan ingatan RAM dan pemprosesan berkuasa tinggi, rekabentuk asal telah dibuat sedikit pengubahsuaian supaya program yang lebih kecil dan ringkas dapat dibangunkan. Ini adalah supaya masa perlaksanaan dapat dipercepatkan dan sekiranya terdapat ralat atau hasilan yang salah bagi program-program ini, ia akan lebih mudah untuk dikesan.

Pada mulanya saya ingin menggabungkan program-program apabila kesemuanya telah diuji tetapi masalah yang sama telah terjadi iaitu masa perlaksanaan yang lama.

6.2.3 MENGGUNAKAN KONSEP LIBRARY

Bagi membolehkan pemprosesan lebih licin dan menjimatkan ruang ingatan, beberapa fungsi yang kerap digunakan atau yang digunakan berulangkali dalam

suatu program telah disimpan didalam library. Untuk itu saya telah mengisytiharkan suatu library yang diberi nama ijanlib.

6.2.4 ANALISA TERHADAP PERSAMAAN MATEMATIK

Oleh kerana tiada rujukan yang menerangkan secara lengkap algoritma Reed-Solomon maka saya telah membuat suatu teori yang dapat mengaitkan data pariti dengan jujukan data asal serta suatu nilai fungsi tetap yang dapat digunakan dalam membuat pengiraan terhadap suatu jujukan data digital. Ini sukar dilakukan kerana kebanyakkan kertas kerja-kertas kerja dan jurnal-jurnal yang diperolehi tidak memberi penerangan secara mendalam tentang algoritma Reed-Solomon. Jurnal-jurnal yang memperkatakan tentang Reed-Solomon hanya lebih banyak tertumpu kepada persamaan-persamaan polinomial. Oleh kerana saya tidak mempunyai latarbelakang matematik yang baik maka sukar bagi saya untuk memahami teori-teori yang dikemukakan.

6.2.5 MENGGUNAKAN PERISIAN PEMACU/ANTARAMUKA

Oleh kerana nilai fungsi gagal dihasilkan di dalam program atau litar, maka saya telah membuat satu andaian bahawa nilai tetap ini akan dimasukkan oleh antaramuka pada setiap kali cip saya diaktifkan. Ini boleh dilakukan kerana nilai yang igin saya masukkan adalah nilai-nilai yang tetap.

6.3 CIRI YANG BOLEH DIBAIKI

Pada masa yang akan datang, sekiranya projek ini ingin diteruskan, saya mendapati ada 2 ciri pertambahan yang boleh dibaiki bagi perkakasan khas saya. Ciri-ciri tersebut saya nyatakan seperti di bawah.

6.3.1 MENGENALPASTI LOKASI RALAT

Dalam pembangunan perkakasan, saya telah membuat anggapan bahawa simbol yang ralat telah diketahui tanpa perlu membuat pencarian lokasi kerana saya telah gagal menemui formula bagi mengaitkan pariti data dengan jujukan data dan lokasi ralat. Semua artikel yang ada cuma memberi formula pengesanan ralat yang melihat kepada setiap bit dalam data. Saya tidak dapat menggunakan formula tersebut kerana jumlah bit data masukan saya adalah besar dan ini akan menyebabkan proses pengiraan yang terlalu panjang sekiranya saya ingin menggunakan formula tersebut. Saya telah menggunakan pendekatan pemecahan data kepada simbol-simbol yang mana pengesanan lokasi sepatutnya dilakukan dengan mengambil kira bukan pada setiap bit dalam jujukan data sebaliknya hanya kepada setiap simbol dalam jujukan data yang mana setiap simbol adalah bersaiz 8 bit.

6.3.2 MASUKAN BAGI FUNGSI & FUNGSI SONGSANG

Untuk masa yang akan datang, nilai bagi fungsi dan fungsi songsang boleh dihasilkan di dalam litar. Pada masa ini, saya tidak dapat mencari nilai-nilai bagi

matriks fungsi songsang yang perlu saya masukkan ke dalam entiti saya bagi membuat pengiraan terhadap data yang hilang. Sekiranya hubungan di antara nilai fungsi asal dan fungsi songsang dapat dicari, saya percaya entiti yang telah saya bangunkan boleh digunakan bagi membuat pengiraan bagi memperoleh data yang hilang.

7.1 KESIMPULAN

Perkakasan khas yang dibangunkan telah dapat membuat pengiraan bagi menghasilkan simbol-simbol pariti bagi jujukan data yang diterima bagi mod pengkodan. Ia juga telah dapat memproses jujukan data bagi mendapatkan nilai bagi data yang hilang. Tetapi satu kelemahan yang ada ialah operasi mendapatkan lokasi ralat dan juga nilai-nilai bagi fungsi songsangan.

Dua ciri ini boleh dibangunkan sekiranya projek ingin diteruskan pada masa yang akan datang. Selain itu, sekiranya kemudahan pada masa akan datang adalah lebih baik iaitu dari segi peralatan lebih berkuasa, perkakasan yang dibangunkan boleh di petakan untuk membentuk modul-modul yang sebenar. Dengan membuat pemetaan modul-modul ini, operasi yang dijalankan oleh perkakasan dapat dilihat dengan lebih jelas.

Daripada projek ini, saya dapati, untuk menghasilkan rekabentuk perkakasan kita perlu melihat permasalahan yang kita ada secara teliti dan masalah tersebut perlu kita selesaikan secara berperingkat, langkah demi langkah. Bagi setiap masalah matematik seperti A=A + B, kita bukan hanya perlu dengan mudah menjalankan operasi penambahan dua nilai pembolehubah tetapi kita perlu juga memikirkan data-data *sum* dan *carry* yang perlu ada bagi operasi penambahan tersebut.

Ini hanya suatu masalah kecil, bagaimana pula sekiranya kita ingin melaksanakan operasi pembezaan atau pengamiran di dalam litar kita. Untuk itu, kita perlu

memecahkan masalah matematik tersebut dan memikirkan cara bagaimana ingin mendapatkan hasil bagi setiap langkah yang telah kita pecah-pecahkan. Penyelesaian masalah di dalam pembangunan suatu perkakasan perlu kita jalankan secara berperingkat.

Pembangunan kod sumber bagi perkakasan adalah lebih kompleks daripada perisian kerana kita perlu memikirkan situasi-situasi yang mungkin berlaku kerana ia melibatkan liar-litar elektronik. Walaupun kod sumber yang telah kita bangunan mungkin telah betul dan dapat menghasilkan keluaran seperti yang sepatutnya tetapi kadangkala ia tidak berlaku seperti itu. Ini berlaku kerana kadangkala keadaan litar dalam perkakasan boleh memberi kesan kepada data output yang dihasilkan.

Secara keseluruhannya, projek pembangunan perkakasan khas ECC (Reed-Solomon) ini telah memberi satu pengalaman berguna kepada saya. Saya telah berpeluang untuk mempelajari selok belok dalam merekabentuk perkakasan yang tidak saya pelajari dalam kursus Sains Komputer Sistem & Rangkaian yang sedang saya ambil. Sekurang-kurangnya saya telah dapat mengaplikasikan pengetahuan pengaturcaraan dan elektronik yang saya pelajari sebelum ini.

Algoritma lengkap bagi pengkod dan penyahkod RS

Algoritma pengkod RS

Bait pariti R perlu di gabungkan dengan mesej K bait bagi membentuk suatu kata kod N= K+R bait. Sutu jujukan bait a0, a1, ..., am boleh diwakilkan sama ada dalam bentuk suatu vektor a = (a0, a1, ... ,am) ataupun suatu polinomial a(x) = a0xm-1 + a1xm-2 + ... + am-1x +am, dimana a0 dan am merupakan bait pertama dan terakhir. Polinomial pariti P(x) dan kata kod C(x) diperolehi daripada persamaan (1) dan (2) seperti di bawah:

$$P(x) = M(x)\, xR \bmod G(x) \quad \text{..... (1)}$$

$$C(x) = M(x)\, xR + P(x) \quad \text{..... (2)}$$

Di mana

$M(x) = m0xK\text{-}1 + m1xK\text{-}2 + \ldots + mK\text{-}2x + mK\text{-}1$ adalah polinomial mesej

$G(x) = P\ i=0..\ R\text{-}1(x + a\ i)$ adalah polinomial penjana

$P(x) = p0xR\text{-}1 + p1xR\text{-}2 + \ldots + pR\text{-}2x + pR\text{-}1$ adalah polinomial pariti

$C(x) = c0xN\text{-}1 + c1xN\text{-}2 + \ldots + cN\text{-}2x + cN\text{-}1$

$= m0xN\text{-}1 + m1xN\text{-}2 + \ldots + mK\text{-}1xN\text{-}K + p0xR\text{-}1 + p1xR\text{-}2 + \ldots + pR\text{-}1$ adalah polinomial kata kod

$P(x)$ adalah merupakan baki apabila $M(x)\ x\ R$ dibahagikan dengan $G(x)$.

Perhatikan bahawa aritmetik tersebut dilakukan dalam bentuk GF(256). Suatu bait data $(d7, d6, \ldots, d1, d0)$ adalah dikenalpstikan dengan elemen ruang Galois $d7a\ 7 + d6a\ 6 +$

... + d1a + d0, dimana *d0* merupakan bit berkeutamaan rendah *(LSB)* dan *d7* adalah bit berkeutamaan tinggi *(MSB).*

Algoritma Penyahkod RS

Algoritma penyahkodan adalah terdiri daripada 5 langkah seperti dibawah.

1. Pengiraan Sindrom

Sindrom, Sj diisytiharkan sebagai:

$$Sj = r(a\ j)$$
$$= c(a\ j) + e(a\ j)$$
$$= e(a\ j), j$$
$$= 0,1,\ldots,2t - 1$$

Persamaan terakhir berlaku kerana sindrom bagi suatu kata kod adalah sifar dan 2t mewakili bilangan simbol pariti. Sindrom 2t boleh dikira daripada polinomial kata yang diterima $r(x) = r0xn-1 + r1xn-2 + \ldots + rn-2x + rn-1$, dimana:

$$Sj = r(a\ j)$$
$$= r0(a\ j)n-1 + r1(a\ j\ )n-2 + \ldots + rn-2a\ j + rn-1$$
$$j = 0,1,\ldots,2t - 1$$

Persamaan tersebut boleh dinilaikan dengan lebih efisien menggunakan peraturan Horner. Dengan menganggap polinomial ralat $e(x) = e0xn-1 + e1xn-2 + \ldots + en-2x + en-1$ mempunyai n ralat (n pekali bukan sifar), sindrom 2t boleh ditulis sebagai:

$$S0 = ei1 + ei2 + \ldots + ein$$
$$S1 = ei1X1 + ei2X2 + \ldots + ein\ Xn$$

...

$$S2t\text{-}1 = ei1X12t\text{-}1 + ei2X22t\text{-}1 + \ldots + ein\ Xn\ 2t\text{-}1$$

Dimana $Xj\ ^{o}\ a\ ij$ dipanggil sebagai penentu kedudukan ralat yang menyatakan lokasi ralat. Persamaan-persamaan sindrom tersebut mempunyai suatu penyelesaian Xjs tak bernilai apabila $n \pounds t$.

2. Membangunkan polinomial penentu lokasi ralat

Menggunakan polinomial penentu lokasi ralat dan identiti Newton , persamaan-persamaan sindrom akan ditafsirkan kepada suatu set baru persamaan yang lebih mudah untuk diselesaikan. Polinomial penentu lokasi ralat diisytiharkan sebagai:

$$L(x) = P\ j=1 \ldots n\ (1\text{-}Xjx) = L\ n\ xn + L\ n\text{-}1xn\text{-}1 + \ldots + L\ 1x + L\ 0$$

Dengan pekali L adalah daripada GF(q). Algoritma Berlekamp-Massey atau Euclid boleh digunakan bagi mencari pekali tersebut.

3. Mencari Lokasi Ralat

Terbalikkan bagi asas of L (x) menentukan penentu lokasi ralat-ralat, Xjs. Oleh kerana suatu ruang finit mempunyai suatu bilangan finit elemen, asas bagi polinomial penentu lokasi ralat boleh ditemui dengan penukargantian setiap elemen kepada polinomial penentu lokasi ralat. Prosedur ini dikenali sebagai Pencarian Chien. Sekiranya asas tersebut adalah tidak jelas (asas berulang) atau tidak berada pada ruang Galois yang

betul (GF(q) yang sama dalam langkah 2), operasi perlu mengisytiharkan bahawa kegagalan penyahkodan berlaku dan berhenti menerusi langkah 4 dan 5.

4. Mencari Nilai Ralat

Penukargantian penentu lokasi Xjs semula kepada persamaan-persamaan sindrom 2t bagi menyelesaikan nilai ralat ***eijs.*** Ini biasa dilakukan dengan algoritma Forney.

5. Membetulkan Ralat

Menambahkan ralat, e, kepada kata kod yang diterima, r bagi mencari semula kata kod yang dihantar, *c.*

betul (GF(q) yang sama dalam langkah 2), operasi perlu mengisytiharkan bahawa kegagalan penyahkodan berlaku dan berhenti menerusi langkah 4 dan 5.

4. Mencari Nilai Ralat

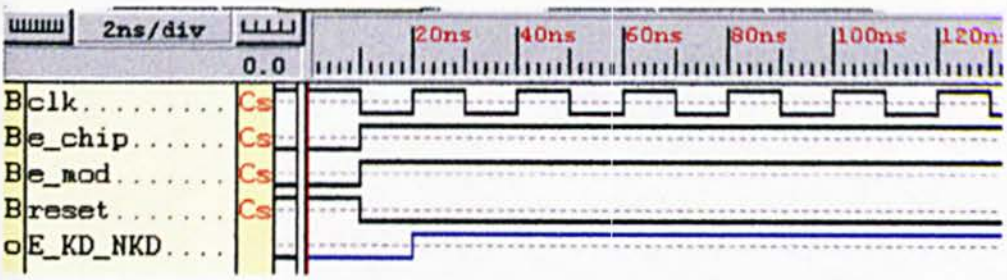
Penukargantian penentu lokasi Xjs semula kepada persamaan-persamaan sindrom 2t bagi menyelesaikan nilai ralat ***eijs.*** Ini biasa dilakukan dengan algoritma Forney.

5. Membetulkan Ralat

Menambahkan ralat, e, kepada kata kod yang diterima, r bagi mencari semula kata kod yang dihantar, *c.*

```
--*****************************************************************--
--*                                                               *--
--*   DESCRIPTION: Reed Solomon coder & decoder                   *--
--*                                                               *--
--*   AUTHOR: Noorizan Muhamad Mursid                             *--
--*                                                               *--
--*****************************************************************--
--unit kawalan
--mod cip
--tentukan baca/tulis dalam daf

LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;

ENTITY controller IS
  PORT (
    clk          : IN STD_LOGIC;
    e_chip  : IN STD_LOGIC;        --masukkan yg memastikan cip sedia
    e_mod   : IN STD_LOGIC;        --masukkan yg beri pilihan encode/decode
    reset   : IN STD_LOGIC;

    e_kd_nkd       : OUT STD_LOGIC;       --isyarat e_kd/e_nkd utk mod cip
    );
END controller;

ARCHITECTURE controller_arch OF controller IS

SIGNAL s_code_ncode    : std_logic;

BEGIN

chip_mod : PROCESS (e_chip,e_mod)

BEGIN

IF (reset='1') THEN

    e_kd_nkd<='0';

ELSIF (clk'event AND clk='1') THEN          --syarat untuk pengkodan dan nyahkod
    IF (e_chip='1') AND (e_mod='1') THEN
        e_kd_nkd<='1';

    ELSIF (e_chip='1') AND (e_mod='0') THEN

        e_kd_nkd<='0';

    END IF;
END IF;

END PROCESS chip_mod;
END controller_arch;
```

Simulasi entiti controller

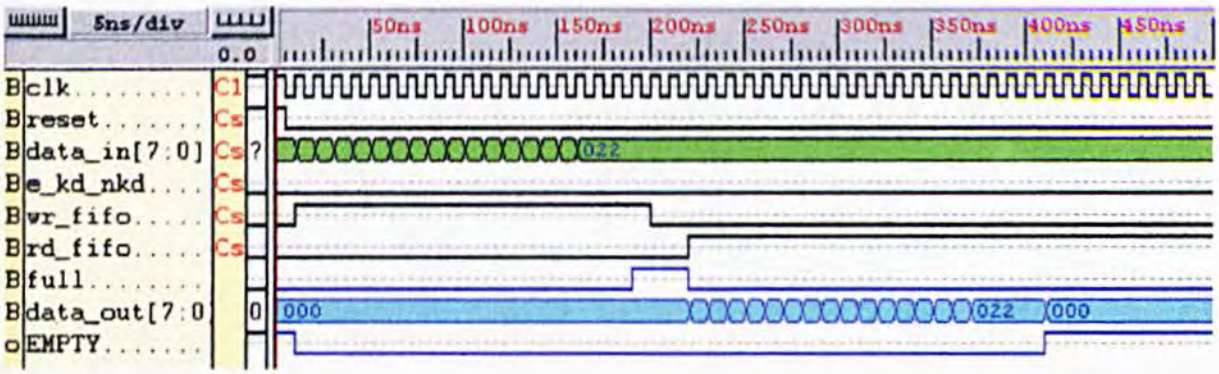


```
--*************************************************************--
--*                                                           *--
--*   DESCRIPTION: Reed Solomon coder & decoder               *--
--*                                                           *--
--*   AUTHOR: Noorizan Muhamad Mursid                         *--
--*                                                           *--
--*************************************************************--

--ingatan utama yg akan terima masuk data input & keluarkan data jika diminta

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY fifo IS

   PORT (
      clk      : IN  std_logic;
      reset    : IN  std_logic;
      data_in  : IN  std_logic_vector (7 DOWNTO 0);
      e_kd_nkd : IN      std_logic;                    -- tentukan jum data 16/19 sim
bol
      wr_fifo        : IN  std_logic;                  -- tulis ke daftar
      rd_fifo  : IN  std_logic;                  -- baca dr daftar

      full     : OUT std_logic;                  -- menunjukkan daftar penuh
      data_out    : OUT std_logic_vector (7 DOWNTO 0);
      empty    : OUT std_logic                   -- menunjukkan daftar kosong

      );

END ENTITY fifo;

ARCHITECTURE one OF fifo IS
   SUBTYPE ptr_range IS natural RANGE 0 TO 18;
   type a IS ARRAY (ptr_range) OF std_logic_vector (7 DOWNTO 0);

   SIGNAL  s_full_flag   : std_logic;
   SIGNAL  s_empty_flag  : std_logic;

   SIGNAL  s_reg_file    : a;                        -- simpan kandungan fifo
   SIGNAL  s_out_ptr_sig : ptr_range;

   SIGNAL  s_in_ptr      : ptr_range;
   SIGNAL  s_out_ptr     : ptr_range;
   SIGNAL  s_in_ptr1     : ptr_range;                -- jumlah masukan & keluaran
   SIGNAL  s_out_ptr1    : ptr_range;

   -- menentukan jumlah bit yg perlu
   CONSTANT c_16bytes  : std_logic:='1';
   CONSTANT c_19bytes  : std_logic:='0';
```

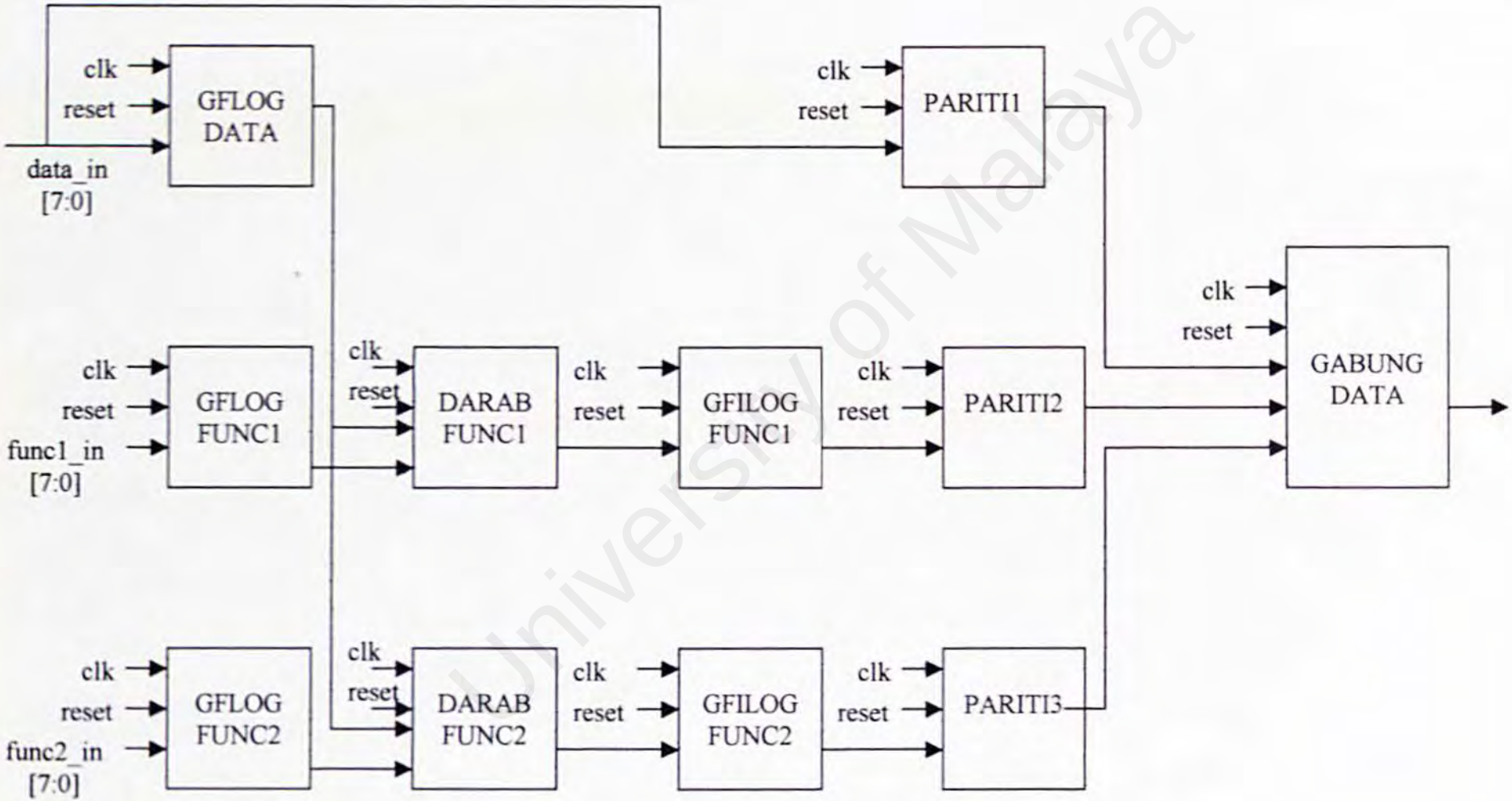
```
BEGIN
-- isyarat daftar penuh/kosong dihantar ke port
  full  <= s_full_flag;
  empty  <= s_empty_flag;

daftar: PROCESS (reset, clk, s_out_ptr, s_reg_file)

  BEGIN
    IF (reset = '1') THEN
      s_in_ptr   <= 0;
      s_out_ptr  <= 0;
      s_in_ptr1  <= 0;
      s_out_ptr1 <= 0;

      s_full_flag <= '0';
      s_empty_flag <= '1';

      data_out<= (OTHERS=>'1');

      FOR z_loop IN 0 TO (18) LOOP
      s_reg_file(z_loop)<=(OTHERS=>'1');
      END LOOP;

    ELSIF (clk'event AND clk='1') THEN

    -- menulis ke dalam daftar
      IF (wr_fifo = '1') AND (s_full_flag = '0') THEN

        s_reg_file(s_in_ptr) <= data_in;
        s_empty_flag <= '0';


        s_in_ptr  <= s_in_ptr+1;
        IF(   (s_in_ptr=15  AND e_kd_nkd=c_16bytes)
          OR (s_in_ptr=18  AND e_kd_nkd=c_19bytes))
        THEN
          s_full_flag <= '1';
          s_in_ptr    <= 0;
        ELSE
          s_full_flag <= '0';
        END IF;

      END IF;


    -- membaca data dari daftar
      IF (rd_fifo = '1') AND (s_empty_flag = '0') THEN
        s_full_flag <= '0';

        s_out_ptr1 <= s_out_ptr;
        s_out_ptr  <= s_out_ptr+1;
```

```
        IF(   (s_out_ptr1=15  AND e_kd_nkd=c_16bytes)
          OR (s_out_ptr1=18  AND e_kd_nkd=c_19bytes))
        THEN
         s_empty_flag <= '1';
         s_out_ptr1   <= 0;
         s_out_ptr    <= 0;
        ELSE
         s_empty_flag <= '0';
        END IF;
        data_out <= s_reg_file((s_out_ptr));

      END IF;

    END IF;

  END PROCESS daftar;

END ARCHITECTURE one;
```

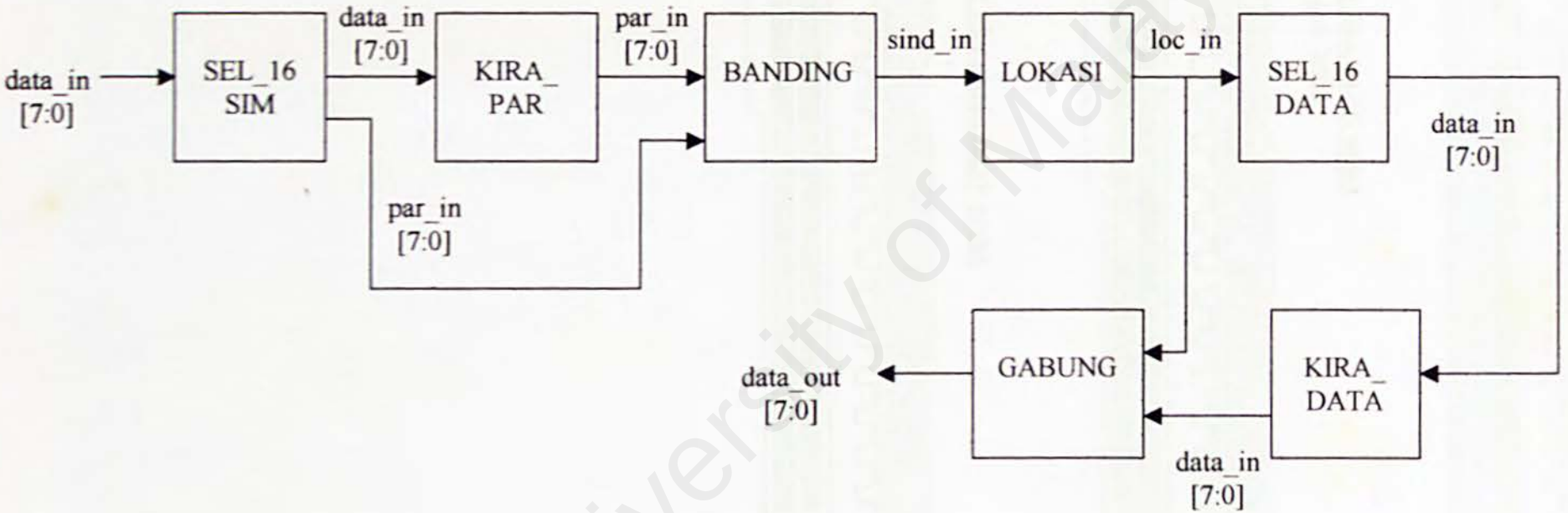
Simulasi entiti main_mem



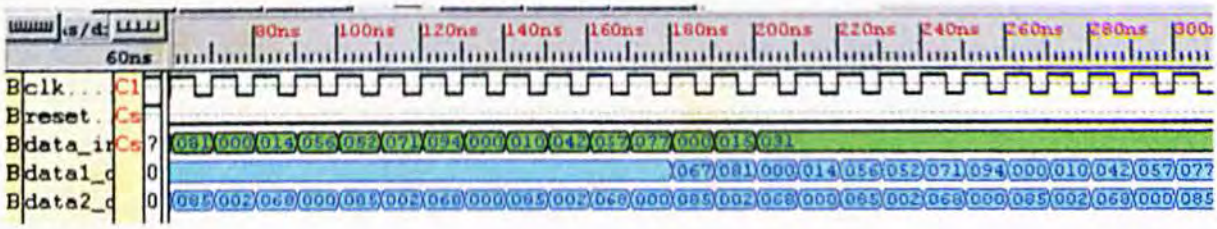
REKABENTUK UNIT PENGKODAN



REKABENTUK UNIT PENYAHKODAN

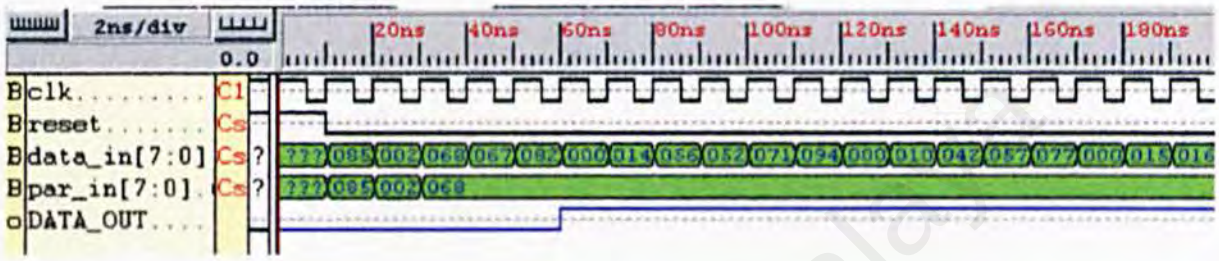


Simulasi entiti sel_16simb



Simulasi entiti banding jika pariti sama

-isyarat '1' kerana timing & latch



Simulasi entiti banding jika pariti tidak sama