DESIGN OF AN 8 BIT RECEIVER BASED ON FPGA.

Oleh

Haidzir Bin Mohammad.

Disertasi ini dikemukakan kepada
UNIVERSITI SAINS MALAYSIA

Sebagai memenuhi sebahagian daripada syarat keperluan
untuk ijazah dengan kepujian

SARJANA MUDA KEJURUTERAAN (KEJURUTERAAN ELEKTRONIK)

Pusat Pengajian Kejuruteraan
Elektrik dan Elektronik
Universiti Sains Malaysia **Mei2006**

ABSTRAK.

Projek ini bertujuan untuk membina dan merekabentuk bahagian penerima didalam UART (*Universal Asynchronous Receive Transmit*) berasaskan FPGA (*Field Programmable Gate Array*). Terdapat dua bahagian utama yang terlibat dalam membangunkan penerima ini iaitu bahagian perisian dan perkakasan. Bahagian perisian adalah yang terpenting dimana ia melibatkan pembangunan keseluruhan penerima menggunakan perisian Xilinx Foundation 2.1i berasaskan kod VHDL (*Hardware Description Languange*) melalui kaedah model perilaku (*Behavioral Model*). Penerima yang dibina akan berfungsi sama seperti penerima didalam UART dimana ia mampu untuk menerima 8 bit data secara siri dan menukarkannya kepada 8 bit data secara selari untuk diproses oleh komputer. Bahagian perkakasan pula melibatkan sambungan litar antara papan demo Xilinx XC4010PC84 dengan komputer melalui litar RS232 dan perkakasan yang lain. Papanuji demo Xilinx XC4010PC84 ini akan berfungsi sebagai penerima dan menunjukkan keluaran data yang dihantar oleh komputer dalam bentuk kod ASCII dan dipamerkan oleh LED dalam bentuk binari.

ABSTRACT.

This project is developed in purpose to design and build a receiver in UART (Universal Asynchronous Receive Transmit) based on FPGA (Field Programmable Gate Array). There are two main parts associated in developing this receiver including software development and hardware installation. The most important part in this project is, in software development section where involving to develop the whole receiver application using Xilinx Foundation 2.1i software based on VHDL (Hardware Description Language) code via Behavioral Modelling. This receiver will have functionality as a receiver in UART to accept 8 bit data in serial and converted to 8 bit data in parallel for communication in computer system. For hardware installation part, it involve a whole connection circuitry with Xilinx XC4010PC84 demo board with computer via connection RS232 cable and the other hardware connection. The Xilinx XC4010PC84 demo board will act like a receiver and display the data that computer send in ASCII character and display it by LED in binary.

PENGHARGAAN

Assalamualaikum w.b.t. dan salam sejahterah.

Alhamdulillah saya panjatkan kesyukuran kehadrat Illahi kerana dengan limpah kurniaNya dan kekuasaanNya dapat saya menyempurnakan disertasi ini untuk projek tahun akhir saya ini.

Di kesempatan ini saya ingin mengucapkan ribuan terima kasih kepada semua yang terlibat dalam menjayakan projek saya ini, terutamanya kepada penyelia saya Puan Zaini Bte. Abd. Halim yang telah banyak memberikan bantuan serta tunjuk ajar dalam menjayakan projek ini. Tidak lupa juga kepada staf-staf akademik, pentadbiran dan teknikal PPK Elektrik dan Elektronik yang banyak memberi bantuan seperti menyediakan komponen, penggunaan alatan dan sebagainya dalam menjayakan projek ini.

Sekalung penghargaan tidak terhingga diucapkan kepada kedua ibu bapa saya serta keluarga tercinta dan semua sahabat kerana telah banyak memberi dorongan, galakan dan pendapat kepada saya untuk menjayakan projek ini.

Akhir kata sekali lagi jutaan terima kasih saya ucapkan kepada semua dalam menjayakan projek ini. Sekian, terima kasih.

HAIDZIR BIN MOHAMMAD.

KANDUNGAN

		Muka Surat
ABSTRAK		ii
ABSTRACT		iii
PENGHARGAAN		iv
JADUAL KANDUNGAN		v
SENARAI RAJAH DAN JADUAL		viii
BAB 1 PENGENALAN		
1.1	Latar Belakang Projek	1
1.2	Operasi UART Secara Keseluruhan	3
1.3	Objektif Projek	4
1.4	Skop Projek	5
1.5	Ringkasan Perlaksanaan Projek	6
	a. Peringkat Pertama	6
	b. Peringkat Kedua	6
	c. Peringkat Ketiga	7
1.6	Penutup	7
BAB 2 KAJIAN ILMIAH DAN TEORI		
2.1	Pendahuluan	8
2.2	UART	11
2.3	Bit Pemula, Bit Henti Dan Pariti Bit	13
2.4	Pengenalan Kepada Xilinx Foundation 2.1i	17
BAB 3 REKA BENTUK DAN KAEDAH PELAKSANAAN		
3.1	Pengenalan	19
3.2	Litar Asas Skematik	19
3.3	Perisian Xilinx Foundation	23

3.31 Membina Folder Simpanan 23
3.32 Menggunakan HDL Editor 24
3.33 Melakukan Sintesis 28
3.34 Melakukan Simulasi 29
3.35 Melakukan Pelaksanaan 32
3.36 Kaedah Verifikasi 34
3.37 Analisa Masa 35
3.38 Programming 35
3.4 Membina Litar Kadar Baud Dan Max 232 38
3.5 Menguji Penerima 39

BAB 4 KEPUTUSAN DAN ANALISA

4.1 Pengenalan 42
4.2 Keputusan Reka bentuk 42
4.3 Keputusan Verifikasi 43
4.4 Hasil Sambungan Litar 44

BAB 5 PENUTUP

5.1 Kesimpulan 45
5.2 Masalah Yang Dihadapi 46
5.51 Memahami Aplikasi Dan Penggunaan Perisian 46
5.52 Kesalahan Dari Kod VHDL 46
5.53 Kesilapan Sambungan 46
5.3 Cadangan Memajukan Projek 47

RUJUKAN

LAMPIRAN

A. VHDL Kod Bagi Penerima

B. Skematik Bagi Papan Demo Xilinx XC4010PC84

C. Helaian Data

D. Laporan Keseluruhan Xilinx FPGA

SENARAI RAJAH DAN JADUAL

MukaSurat

Rajah 1.1 : Aplikasi UART didalam penghantaran data. 3

Rajah 1.2 : Bagi satu paket data yang terdiri dari 7 data bit dan 1 pariti bit bagi bit 1 adalah genap. 4

Rajah 1.3 : Bagi satu paket data yang terdiri dari 7 data bit dan 1 pariti bit bagi bit 1 adalah ganjil. 5

Rajah 2.1 : Perhubungan data secara selari (parallel). 10

Rajah 2.2 : Perhubungan data secara siri (serial). 10

Rajah 2.3 : Perhubungan tanpa medium atau tanpa wayar (mengunakan gelombang) 10

Rajah 2.4 :Litar Bersepadu UART- 8251 berasaskan Intel (PD8251A/AF) 12

Rajah 2.5 : Sistem dikatakan didalam keadaan idle. 13

Rajah 2.6 : Bit 0 yang pertama dihantar selapas mod idle, bit pemula diterima. 14

Rajah 2.7 : 7 bit seterusnya dikatakan sebagai data bit dan bit kelapan sebagai bit pariti. 15

Rajah 2.8 : Jadual sebahagian dari kod ASCII 16

Rajah 2.9 : Paparan antaramuka utama Xilinx Foundation 2.1i. 18

Rajah 3.1 : Blok-blok didalam UART. 20

Rajah 3.2 : Gambarajah skematik penerima. 21

Rajah 3.3 : Permulaan Xilinx Foundation. 23

Rajah 3.4 : Masukan nama Folder projek. 24

Rajah 3.5 : Bahagian HDL Editor dipilih. 25
Rajah 3.6 : Antaramuka permulaan VHDL kod. 26
Rajah 3.7 : Kod VHDL yang selesai dimasukan. 27
Rajah 3.8 : Konfigurasi yang digunakan untuk projek ini. 29
Rajah 3.9 : Tetingkap Simulator. 30
Rajah 3.10 : Memilih isyarat yang ingin disimulasikan. 31
Rajah 3.11 : Hasil simulator untuk fungsi penerima (Receiver). 32
Rajah 3.12 : Rajah simulasi untuk Implementation. 33
Rajah 3.13 : Gambarajah simulasi fungsi masa. 34
Rajah 3.14 : Paparan PROM File Formatter. 36
Rajah 3.15 : Konfigurasi untuk menyimpan data TEKHEX 37
Rajah 3.16 : Rajah skematik litar kadar baud. 38
Rajah 3.17 : Litar RS232. 39
Rajah 3.18 : Papan demo Xilinx XC4010PC84 dan sambungannya. 41

BAB 1

PENGENALAN

1.1 Latarbelakang Projek.

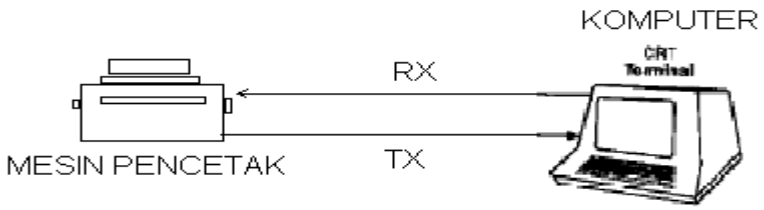
Pada kebiasaannya kaedah yang biasa digunakan dalam pembangunan suatu litar bersepadu (*IC:Integrated Circuit*) didalam industri perkilangan elektronik adalah menggunakan kaedah fabrikasi silikon menggunakan kaedah VLSI (*Very Large Scale Integrated*). Melalui kaedah ini fungsi dan aplikasi litar bersepadu yang hendak dibangunkan dikaji dan diteliti kemudian litar-litar serta get-get logik yang perlu digunakan dijana menggunakan perisian tertentu seperti Tanner Tool atau Cadence. Melalui perisian ini juga hasil pembangunan litar bersepadu yang dikehendaki dibina sepenuhnya kemudian diuji menggunakan simulator Tanner Tool atau Cadence. Sekiranya hasil simulator bagi litar bersepadu yang dibangunkan sesuai dengan objektif dan matlamat, litar bersepadu ini akan dibina menggunakan kaedah fabrikasi silikon. Perlu diketahui walaupun ujian simulator yang dijana oleh perisian Tanner Tool ataupun Cadence memenuhi objektif dan matlamat penghasilan litar bersepadu yang dibangunkan, kadangkala selepas proses fabrikasi silikon dan wafer silikon telah siap dibina, litar bersepadu yang dibina tidak dapat beroperasi seperti apa yang dikehendaki didalam objektif dan matlamat penghasilan litar bersepadu. Melalui kaedah ini juga

sukar untuk membina model prototaip kerana selepas sahaja proses fabrikasi silikon ia akan menghasilkan beribu atau beratus litar bersepadu diatas sekeping wafer bergantung kepada saiz litar bersepadu yang direka dan proses penambahbaikan litar bersepadu yang dibagunkan sukar untuk dilakukan. Setiap kali penambahbaikan dilakukan ianya seperti membuat kembali litar bersepadu sedia ada dari awal proses. Ini akan memakan masa yang lama dan membabitkan penggunaan kos yang tinggi kerana setiap kali proses fabrikasi silikon dilakukan ia memerlukan perbelanjaan berpuluh-puluh ribu ringgit.

Salah satu kaedah yang lebih baik adalah menggunakan kaedah FPGA (*Field Programmable Gate Array*). Kaedah ini sebenarnya menggunakan konfigurasi get-get logik untuk menghasilkan aplikasi suatu litar bersepadu. Pada masa lalu kaedah ini tidak sesuai digunakan kerana ia mempunyai jumlah get-get logik yang terhad dimana get-get logik ini boleh dikonfigurasikan untuk kesesuaian litar bersepadu yang dibina. Tetapi pada masa kini ia telah mempunyai berjuta-juta get-get logik yang boleh dikonfigurasikan untuk kesesuaian pembangunan litar bersepadu dan sesuai dengan perkembangan litar bersepadu yang lebih kecil tetapi mempunyai banyak aplikasi.(*Spartan3 Generation* : Xilinx) Melalui kaedah ini juga ia boleh menghasilkan prototaip litar bersepadu yang dibangunkan dan proses penambahbaikan mudah dilakukan serta menjimatkan kos.

1.2 Operasi UART secara keseluruhan.

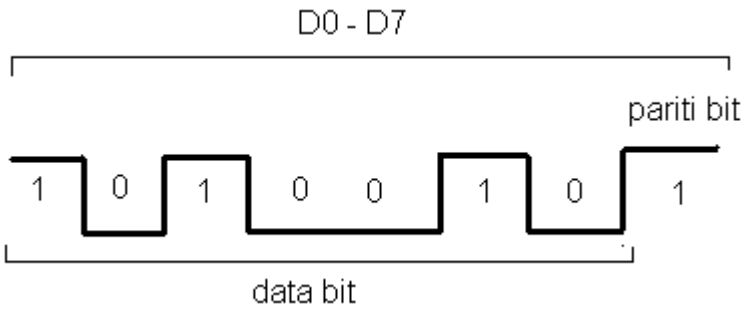
Secara ringkasnya, UART berfungsi untuk perhubungan dua hala yang terdiri daripada penghantar (*Transmitter*-TX) dan penerima (*Receiver*-RX) melalui perhubungan separa duplex (*half duplex*). Ini bermaksud pada satu masa hanya satu laluan data dibenarkan iaitu samaada talian penghantar ataupun talian penerima. Ini dapat dilakukan dengan kaedah "*hardware hand-shaking*". Secara amnya UART terdiri dari 3 bahagian utama iaitu blok penghantar, blok penerima dan penjana baudrate (*baudrate generator*). Blok penghantar akan berfungsi sebagai penghantar data dari alatan komputer ke komputer atau sebaliknya manakala penerima berfungsi menerima data dari komputer ke alatan komputer atau sebaliknya. Kedua-dua perhubungan ini melibatkan penghantaran data secara siri. Data yang diterima akan digunakan sebagai masukan bagi perhubungan yang dikehendaki seperti perhubungan data diantara komputer dan pencetak atau lain-lain lagi.(Larry Hughes,1992)



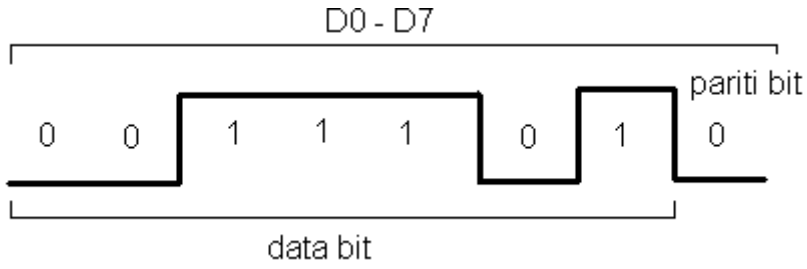
Rajah 1.1 : Aplikasi UART didalam penghantaran data.

1.3 Objektif Projek.

Projek yang dijalankan ini bertujuan untuk membina dan mereka bentuk 8-bit penerima untuk UART (*Universal Asynchronous Receive Transmit*) berasaskan FPGA (*Field Programmable Gate Array*). Penerima ini akan dibina menggunakan perisian Xilinx Foundation 2.1i dengan ciri-ciri yang dikehendaki. Penerima ini akan berfungsi untuk menerima data secara siri dari komputer ke alatan komputer atau sebaliknya. Data yang diterima adalah dalam bentuk 8 bit secara siri termasuk pariti bit. Dalam sistem komputer kebiasaannya pariti bit digunakan untuk mengesan ralat data yang diterima. Untuk UART biasanya bit yang ke lapan adalah untuk pariti bit. Sekiranya jumlah bit '1' adalah dalam bilangan genap, pariti bitnya akan '0' dan sekiranya jumlah bit '1' adalah ganjil, pariti bitnya adalah '1'. Contoh diberikan seperti dalam Rajah 1.2 dan Rajah 1.3 :



Rajah 1.2 : Bagi satu paket data yang terdiri dari 7 data bit dan 1 pariti bit bagi bit 1 adalah genap.



Rajah 1.3 : Bagi satu paket data yang terdiri dari 7 data bit dan 1 pariti bit bagi bit 1 adalah ganjil.

Bagi satu paket data ia terdiri dari 7 bit data kerana komputer pada kebiasaannya menghantar data dalam bentuk kod ASCII. Kod ASCII menggunakan 7 bit data untuk membentuk 2^7 = 128 simbol-simbol bagi perhubungan data komputer. Pada keseluruhannya penerima yang dibina dapat menerima data dalam bentuk 8 bit data secara siri dan menukarkannya kepada 8 bit data secara selari. Apabila projek ini dapat disiapkan dengan dengan baik, ia akan berfungsi sepenuhnya sebagai penerima seperti didalam UART dimana ia dapat mengesan ralat data seterusnya menerima data dengan tepat tanpa ralat.

1.4 Skop Projek.

Penerima akan dibina berasaskan kod VHDL menggunakan kaedah model perilaku.(Volnei A. Pedroni-2004) Penerima ini akan menerima data-data bit yang dihantar oleh komputer dalam bentuk data binari dimana bit ‘1’ LED akan menyala dan bit ‘0’ LED tidak menyala.

1.5 Ringkasan Pelaksanaan Projek.

Secara umumnya ada dua bahagian yang berbeza dalam projek ini yang perlu dilaksanakan dalam memastikan projek ini dapat berjalan dengan lancar dan penerima dapat berfungsi dengan baik iaitu bahagian perisian yang melibatkan penggunaan perisian Xilinx Foundation 2.1i dan bahagian perkakasan (*hardware*) untuk penggunaan papan demo Xilinx (*Demo Board*).

a. **Peringkat pertama** :

   Memahami tajuk projek dan mempelajari penggunaan perisian Xilinx Foundation 2.1i serta memahami aplikasi penerima bagi UART untuk dibangunkan menggunakan FPGA berasaskan Xilinx Foundation 2.1i. Menentukan komponen-komponen yang perlu digunakan untuk keperluan projek.

b. **Peringkat Kedua :**

   Membangunkan komponen penerima menggunakan VHDL kod didalam Xilinx Foundation 2.1i serta menentukan ciri-ciri yang perlu ada untuk penerima berfungsi sama seperti penerima didalam UART. Selepas VHDL kod yang ditulis boleh digunakan, simulator akan dilakukan keatas kod tersebut. Sekiranya tiada kesalahan yang berlaku, perisian Xilinx Foundation 2.1i akan menghasilkan data dalam format TEKHEX untuk ditulis (*write/burn*) keatas litar bersepadu

UVEPROM. Seterusnya papan demo Xilinx dapat digunakan untuk menunjukkan keluaran penerima yang dibangunkan.

c. <u>Peringkat Ketiga</u> :

Membina litar penuh yang terdiri daripada papan demo Xilinx yang bertindak sebagai penerima, sambungan litar RS232 dan komponen yang lain seperti LED, penjana kadar baud dan lain-lain.

1.6 Penutup.

Penetapan objektif projek amat penting bagi memastikan projek yang dilaksanakan dapat berfungsi dengan baik dan berfungsi seperti yang dikehendaki. Ini juga dapat memberi gambaran yang menyeluruh serta pemahaman keatas projek yang dilakukan seterusnya memudahkan dalam menyiapkan projek. Disamping itu, pengumpulan data serta analisis maklumat daripada sumber-sumber rujukan amat diperlukan bagi memastikan kesempurnaan projek.

BAB 2

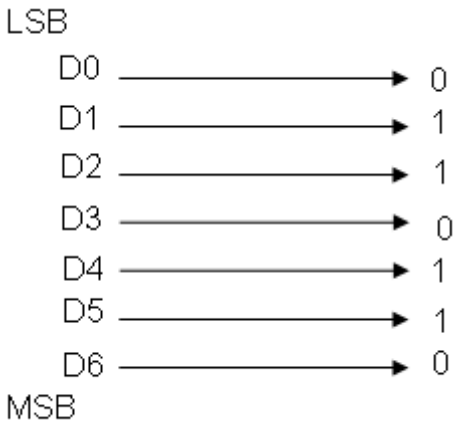
KAJIAN ILMIAH DAN TEORI

2.1 Pendahuluan.

Perhubungan data dapat diketegorikan kepada dua kaedah asas iaitu menggunakan medium atau tanpa medium. Medium bermaksud kita menggunakan suatu benda atau alatan untuk penghantaran data seperti menggunakan wayar, optik fiber dan pelbagai lagi. Tanpa medium pula kaedah yang biasa digunakan adalah menggunakan gelombang. Walaupun kaedah ini berbeza, namun satu persamaan yang ada untuk penghantaran data pada masa kini ialah penghantaran data menggunakan data binari iaitu data yang terdiri dari bit 1 dan bit 0.(Ifiok Otung, 2001)

Untuk penghantaran data menggunakan medium, dua cara penghantaran yang selalu digunakan iaitu secara siri atau selari. Pada masa lalu kaedah selari dikatakan kaedah penghantaran data yang pantas dimana satu data dapat dihantar pada satu masa sahaja seperti didalam Rajah 2.1. Namun demikian kaedah ini memerlukan kos yang tinggi sekiranya jarak penghantaran data adalah jauh diantara penerima dan penghantar. Ini kerana sekiranya data yang dihantar terdiri dari 7 bit, wayar yang perlu disambung kepada penerima adalah terdiri dari 7 laluan wayar dan sekiranya jarak yang dihantar adalah jauh dari sumber data ini akan melibatkan kos yang tinggi hanya untuk saluran data. Perhubungan data jenis selari ini hanya sesuai untuk jarak yang dekat sahaja dan tidak sesuai untuk jarak perhubungan data yang jauh. Perhubungan data jenis siri pula terdiri dari satu saluran data atau menggunakan hanya satu medium atau wayar seperti didalam Rajah 2.2. Dulu perhubungan jenis ini dikatakan hubungan data yang perlahan

dimana data dihantar sebagai bit per bit. Ini bermaksud sekiranya 7 bit data perlu dihantar ia akan menghantar bit yang pertama dahulu biasanya sebagai LSB (*Least Significant Bit*) kemudian bit yang kedua dan seterusnya sehingga bit yang ke tujuh sebagai MSB (*Most Significant Bit*). Ini akan memakan masa yang lama tetapi menjimatkan kos dari segi penggunaan medium seperti wayar. Oleh kerana kaedah siri ini lebih menjimatkan, pakar-pakar elektronik telah memajukan sistem ini sehingga ia mampu menghantar data dengan masa yang lebih singkat. Oleh itu pada masa kini perhubungan data menggunakan kaedah siri ini telah menjadi satu perhubungan data yang pantas dan menjimatkan kos serta banyak digunakan didalam sistem elektronik.Bagi perhubungan data tanpa medium, kaedah ini lebih rumit untuk dibangunkan. Ini kerana litar-litar yang dibina perlu sensitif dan kompleks untuk mengesan perubahan gelombang untuk mengesan bit 1 dan 0. Bagi projek ini perhubungan data yang akan dibincangkan adalah dalam bentuk siri. (Larry Hughes, 1992)



Rajah 2.1 : Perhubungan data secara selari (*parallel*).

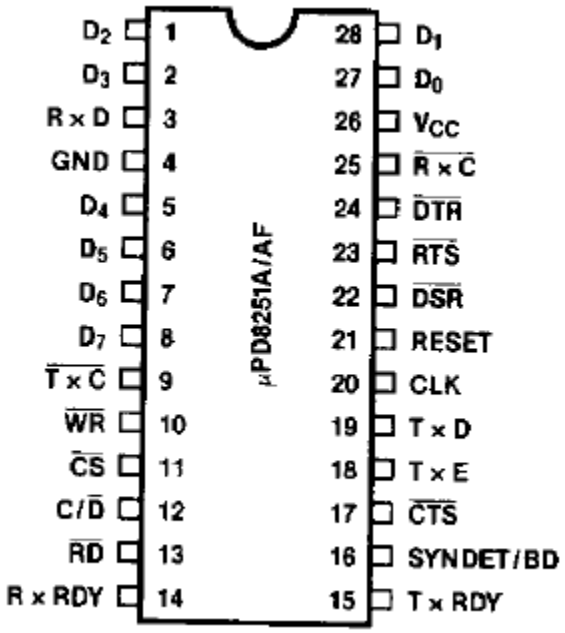
Rajah 2.2 : Perhubungan data secara siri (*serial*).

Rajah 2.3 : Perhubungan tanpa medium atau tanpa wayar (mengunakan gelombang)

2.2 UART.

Pada permulaan era penghantaran data dalam bentuk bit, satu sistem yang sama seperti UART digunakan dimana ia menggunakan komutator mekanikal yang berputar. Ini diaplikasikan untuk penghantaran 5 bit kod boudot menggunakan tele-mesintaip mekanikal (*mechanical tele-typewriter*) kemudian kaedah ini digantikan dengan penggunaan kod morse. Selepas itu penggunaan kod ASCII diperkenalkan yang menggunakan kod berasaskan 7 bit. Apabila syarikat IBM, pengasas penciptaan komputer pada awal tahun 1960 menggunakan 8 bit untuk simbol data dalam komputer, ia menjadi satu permulaan untuk menyimpan kod ASCII dalam bentuk 8 bit data. Gordon Bell adalah orang pertama yang membina chip UART untuk PDP-1 (*Programmed Data Processor-1*) komputer pertama yang dicipta. Pada awal tahun 1980 UART yang dibangunkan oleh National Semiconductor adalah 8250. Kemudian UART yang lebih baik dicipta yang mengandungi pemampan (*buffer*) diatas chip. Ini akan membolehkan chip UART manghantar data dengan kelajuan tinggi tanpa kehilangan data. Sebagai contoh chip National Semiconductor 16550 mengandungi 16 bait FIFO (*First In First Out*). Bergantung kepada pengeluar, pelbagai kaedah digunakan untuk mengenalpasti chip-chip yang beroperasi sama seperti UART. Bagi Intel, ia menamakan chip yang sama fungsi seperti UART sebagai 8251 "*Programmable Communication Interface*" manakala Motorola pula menggunakan nama "*Serial Communication Interface*" (SCI). Walaupun

nama-nama yang digunakan berbeza, tetapi fungsinya adalah sama iaitu digunakan untuk perhubungan data sesiri (*serial interface*). (Wikipedia Encyclopedia)



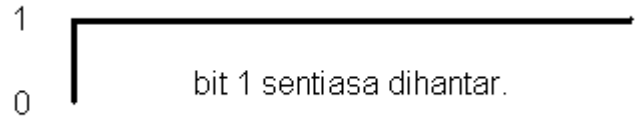
Rajah 2.4 :Litar Bersepadu UART- 8251 berasaskan Intel (PD8251A/AF)

UART (*Universal Asynchronous Receiver/Transmitter*) adalah satu litar bersepadu Litarbersepadu Skala Besar (LSI) yang mengandungi semua program aturcara (*software*) yang diperlukan untuk mengawal port sesiri (*serial port*) bagi komputer. UART adalah satu litar bersepadu elektronik yang berfungsi untuk meghantar (*Transmitter*) data dan menerima (*Receiver*) data melalui port sesiri. Ia akan menukar bait (*byte*) kepada bit sesiri untuk penghantaran data dan sebaliknya. Ia juga berfungsi untuk mengenal pasti bit pemula (*start bit*) dan bit henti (*stop bit*) untuk setiap paket data yang

diterima. Sekiranya penghantaran data bermula, UART kemudiannya akan membina satu bait data yang terdiri dari 8 bit berdasarkan talian penerima. Sekiranya pariti digunakan didalam penghantaran data, ia akan memeriksa bit pariti untuk memeriksa ralat data dan sekiranya pariti bit yang diterima adalah betul, data akan dihantar ke komputer atau sebaliknya untuk diproses. (USRobotics)

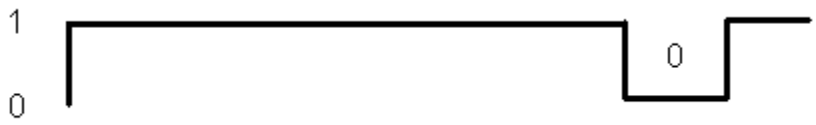
2.3 Bit Pemula (*Start Bit*), Bit Henti (*Stop Bit*) Dan Pariti Bit.

Bagi satu paket data yang diterima oleh talian penerima ia selalunya terdiri dari 10 bit dimana satu bait bersamaan dengan 8 bit adalah untuk data dalam bentuk kod ASCII dan dua bit lagi adalah untuk bit pemula dan bit akhir. Sekira talian adalah dalam mode *idle* samaada untuk talian penerima (*Receiver*) atau talian penghantar (*Transmitter*), bit 1 akan sentiasa dihantar seperti didalam Rajah 2.5. (Larry Hughes, 1992)



Rajah 2.5 : Sistem dikatakan didalam keadaan idle.

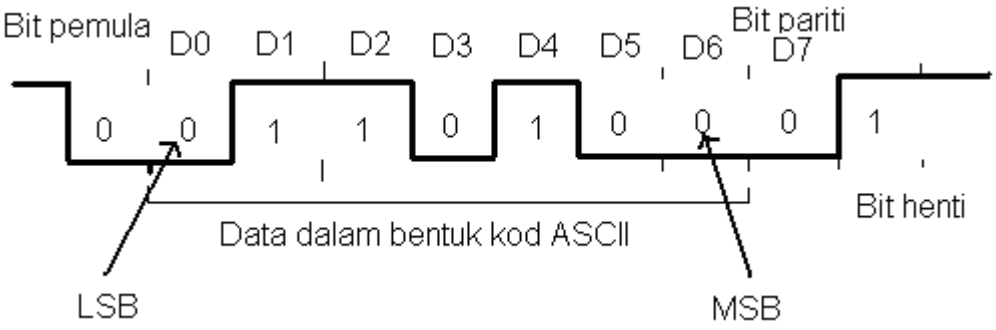
Sekiranya ada data yang dihantar oleh komputer melalui talian penerima (*Receiver*) bit 0 akan dihantar sebagai bit penanda untuk bit pemula atau dikenali sebagai *active low* seperti didalam Rajah 2.6 :



Rajah 2.6 : Bit 0 yang pertama dihantar selapas mod idle, bit pemula diterima.

Sekiranya UART mengesan bit 0 ini, 8 bit yang seterusnya akan ditandakan sebagai data bit dan bit yang kesepuluh ditanda sebagai bit henti. Bit henti biasanya ditanda sebagai bit 1. Ini penting bagi memastikan penerima (*Receiver*) UART dapat memastikan data yang diterima adalah sah. Sekiranya bit yang kesepuluh bukan bit 1, UART akan menghantar isyarat tidak sah kepada komputer kemudian komputer perlu menghantar kembali data sebelumnya bagi memastikan data yang diterima adalah sah. Walaubagaimanapun kadangkala komputer akan menghantar data mengikut nilai bit pemula dan bit henti yang betul tetapi 8 bit data yang diterima kadangkala adalah data

yang salah. Sebagai contoh bit pemula adalah bit 0 dan bit kesepuluh seterusnya adalah bit 1 tetapi 8 bit diantara bit pemula dan bit henti mempunyai bit-bit data yang berlainan dari kod ASCII. Ini dapat dielakkan dengan menggunakan bit pariti. Seperti yang diketahui, kod ASCII sebenarnya terdiri dari kod 7 bit data. Tetapi didalam sistem komputer kod ASCII dihantar menggunakan 8 bit lebar data. Ini kerana bit yang kelapan biasanya adalah untuk pariti bit. Pariti bit berfungsi untuk memastikan nilai bit 1 dalam kod ASCII adalah genap atau ganjil. Sekiranya nilai bit 1 adalah genap bit kelapan bagi bit data adalah bit 0 dan sekiranya nilai bit 1 adalah ganjil bit kelapan bagi bit data adalah 1. Ini telah ditunjukan seperti didalam bab pengenalan. Walau bagaimanapun bagi 8 bit data yang dihantar data adalah mengikut susunan dari *Least Significant Bit* (LSB) ke *Most Significant Bit* (MSB) iaitu selapas bit pemula dihantar kemudian diikuti LSB seterusnya bit yang ke tujuh adalah MSB dan kelapan adalah bit pariti seperti ditunjukkan didalam Rajah 2.7 :.



Rajah 2.7 : 7 bit seterusnya dikatakan sebagai data bit dan bit kelapan sebagai bit pariti.

Seperti didalam gambarajah diatas, data yang diterima oleh penerima UART adalah data yang sah kerana ia mematuhi bit pemula, bit henti dan bit pariti. Data yang akan diambil oleh UART adalah dari D0 hingga D7 dan akan ditafsirkan mengikut kod ASCII dari jadual kod ASCII mengikut data dari D0 hingga D6 = 01101000 iaitu 0001011 (dalam binari) = 11 (dalam desimal) bersamaan dengan simbol " VT ". Dibawah diberikan jadual kod ASCII untuk data 0001011. Jadual penuh untuk kod ASCII dilampirkan didalam Lampiran. (Zoran Salcic, Asim Smailagic-2000)

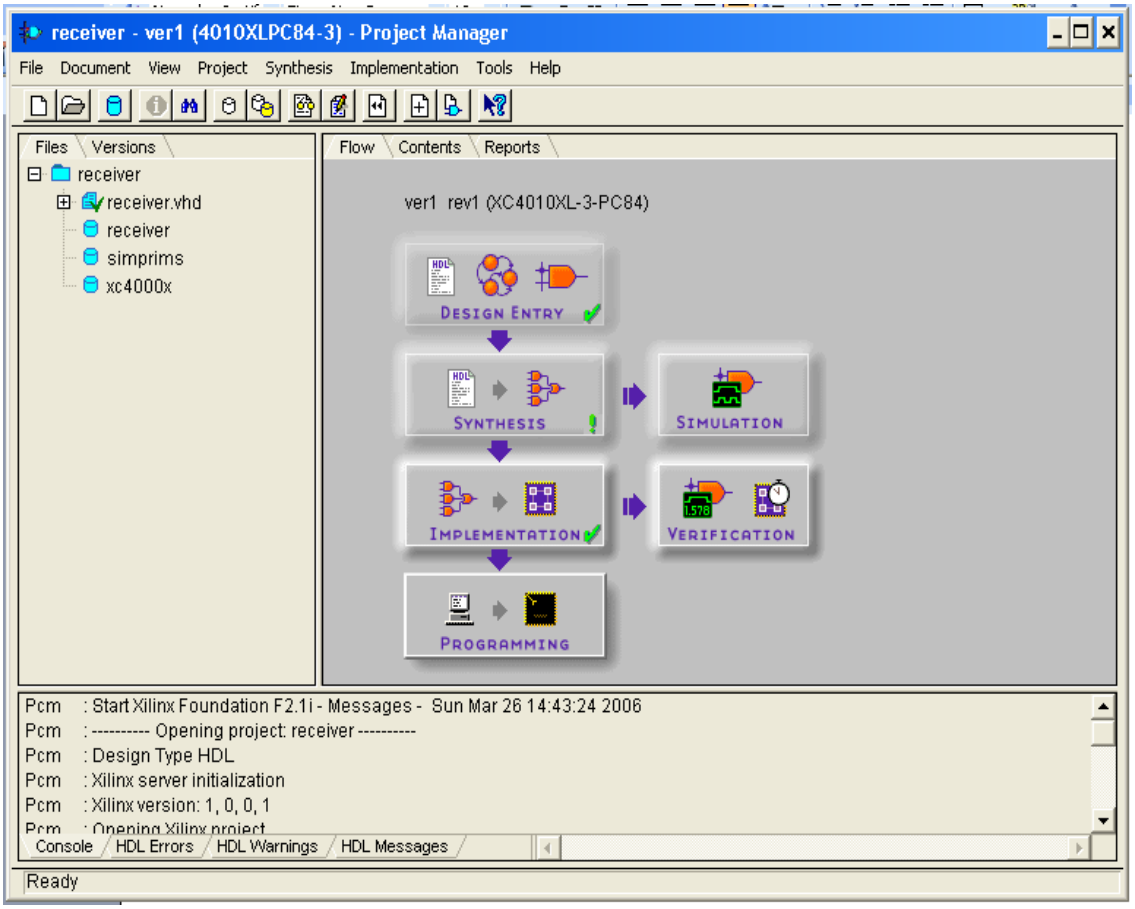
ASCII Table

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	NUL	44	2C	,	88	58	X
1	1	SOH	45	2D	-	89	59	Y
2	2	STX	46	2E	.	90	5A	Z
3	3	ETX	47	2F	/	91	5B	[
4	4	EOT	48	30	0	92	5C	\
5	5	ENQ	49	31	1	93	5D	]
6	6	ACK	50	32	2	94	5E	^
7	7	BEL	51	33	3	95	5F	
8	8	BS	52	34	4	96	60	‘
9	9	HT	53	35	5	97	61	a
10	A	LF	54	36	6	98	62	b
11	B	VT	55	37	7	99	63	c
12	C	FF	56	38	8	100	64	d
13	D	CR	57	39	9	101	65	e
14	E	SO	58	3A	:	102	66	f
15	F	SI	59	3B	;	103	67	g
16	10	DLE	60	3C	<	104	68	h
17	11	DC1	61	3D	=	105	69	i
18	12	DC2	62	3E	>	106	6A	j
19	13	DC3	63	3F	?	107	6B	k

Rajah 2.8 : Jadual sebahagian dari kod ASCII

2.4 Pengenalan Kepada Xilinx Foundation 2.1i.

Xilinx Foundation 2.1i adalah merupakan satu perisian yang digunakan untuk membangunkan litar bersepadu berasaskan FPGA. Dengan menggunakan perisian ini, litar bersepadu yang hendak dibangunkan boleh dibina melalui salah satu dari 3 kaedah iaitu melalui skematik get logik, menggunakan arahan HDL atau menggunakan *Finite State Machine* (FSM) – (James W.Stewart, Chao-Ying Wang-2001). Kadangkala gabungan antara kaedah-kaedah ini juga boleh digunakan. Paparan antaramuka Xilinx Foundation 2.1i adalah seperti didalam Rajah 2.40 (Xilinx Foundation 2.1i : Build 3.1.162 – Project Manager : Build 5.00.33). *Project Manager* adalah antaramuka utama dimana ia akan menggabungkan semua tugas-tugas yang dilakukan bermula dari *Design Entry* → *Synthesis* → *Simulation* → *Implementation* → *Verification* → *Programming*. Perisian ini dapat membina projek yang berasaskan get logik dan juga untuk membangunkan litar bersepadu berasaskan FPGA



Rajah 2.9 : Paparan antaramuka utama Xilinx Foundation 2.1i.

BAB 3

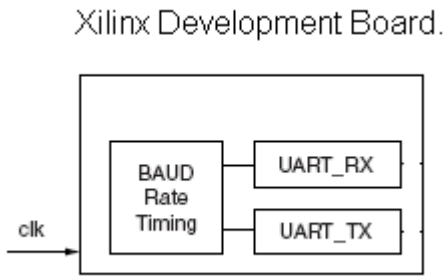
REKABENTUK DAN KAEDAH PERLAKSANAAN

3.1 Pengenalan.

Seperti yang telah diterangkan didalam Bab 1 dan Bab 2, bahagian ini pula akan menerangkan dengan lebih lanjut bagaimana kaedah mereka bentuk penerima UART menggunakan perisian Xilinx Foundation 2.1i. Bahagian ini akan menerangkan dengan terperinci tentang merekabentuk penerima dari awal proses sehingga penerima dapat berfungsi dengan sepenuhnya. Perlu diketahui bahawa kaedah yang digunakan dalam melaksanakan projek ini adalah menggunakan VHDL (*Hardware Description Language*) kod melalui kaedah model perilaku (*Behavioral Modelling*).

3.2 Litar Asas Skematik UART.

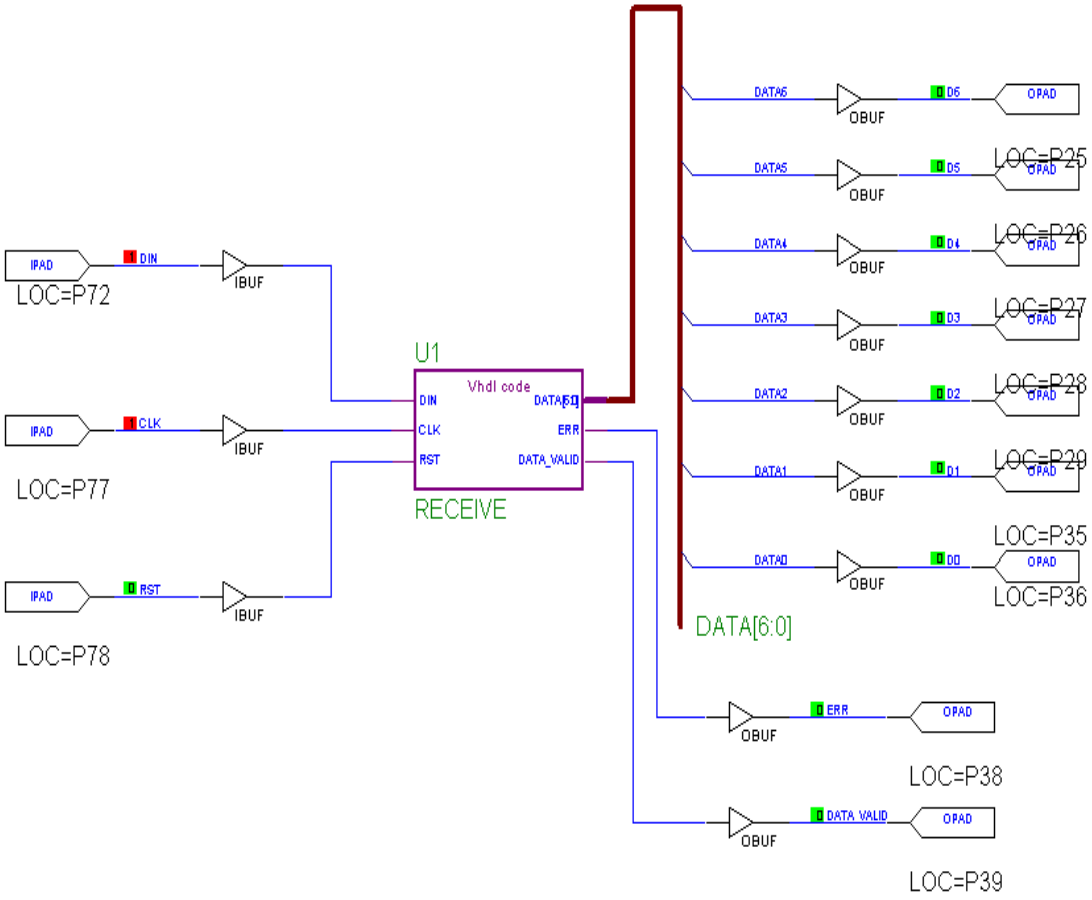
Litar asas bagi UART adalah seperti didalam Rajah 3.1 yang terdiri dari blok Penghantar (*Transmitter-Tx*), Penerima (*Receiver-Rx*) dan Penjana Kadarbaud (*Baudrate Generator*). Gabungan ketiga-tiga blok ini akan menghasilkan satu litar bersepadu UART yang lengkap.



Rajah 3.1 : Blok-blok didalam UART.

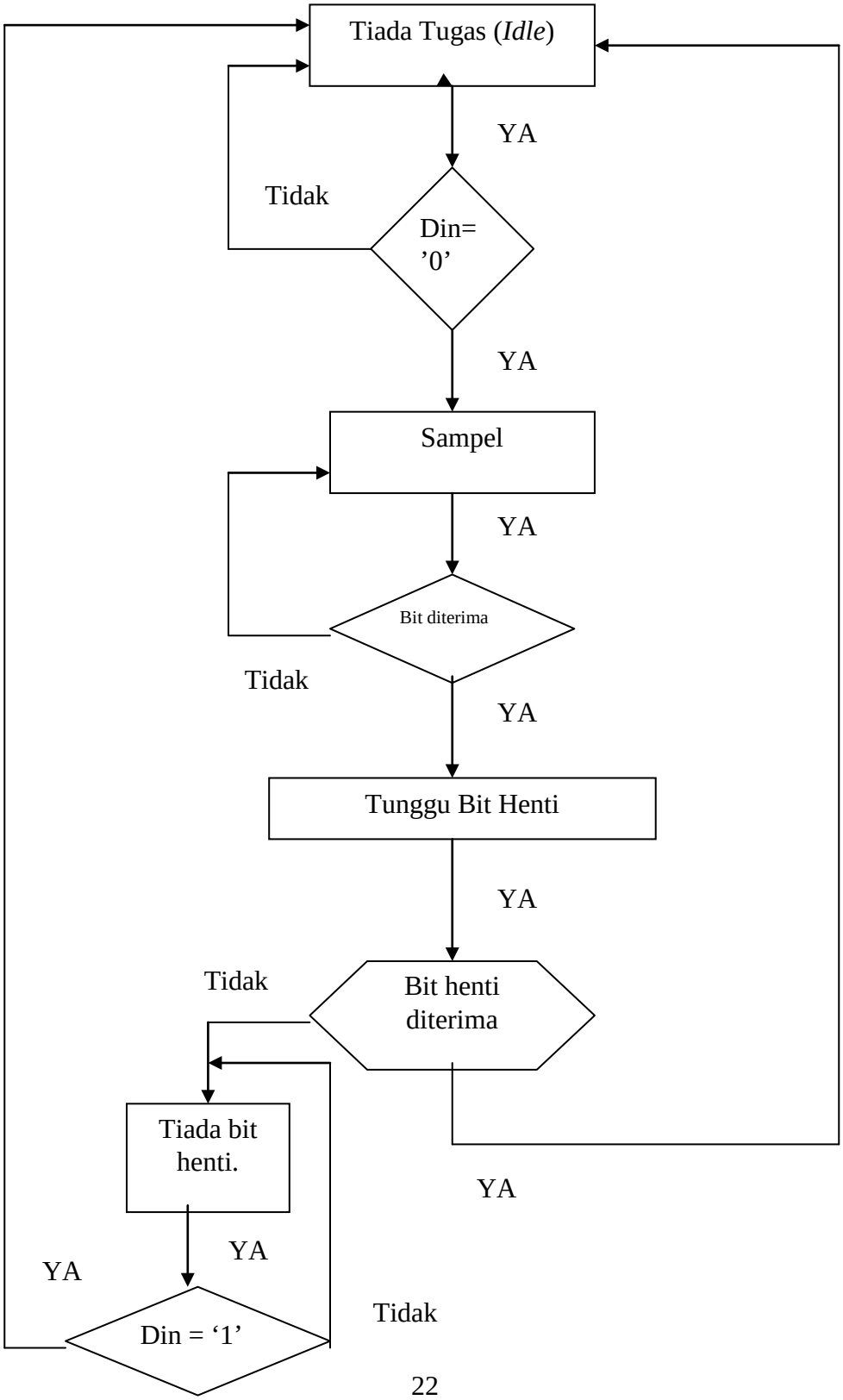
Bagi projek ini hanya blok penerima (*Receiver-Rx*) sahaja yang akan dibina menggunakan Xilinx Foundation 2.1i.

Daripada carta alir yang dihasilkan, kod VHDL melalui kaedah model perilaku dapat dihasilkan. Kaedah model perilaku dipilih kerana penerima yang dibangunkan menggunakan kod VHDL yang mempunyai ciri-ciri komplek dan kaedah ini dapat memenuhi semua aplikasi penerima yang dikehendaki. Penerima ini akan terdiri dari tiga masukan iaitu Din (masukan data), CLK (masukan kadar baud), RST (set semula) dan sembilan keluaran iaitu data-data keluara dari D0-D6, ERR (isyarat kesalahan), DATA_VALID (isyarat data sah). Kadar baud komputer dan kadar baud penerima perlu sama bagi memastikan data yang dihantar dapat dibaca dan ditafsirkan oleh penerima. (Muhammad Ali Mazidi, Janice Gillispie Mazidi.-2000) Gambarajah skematik ditunjukkan seperti didalam Rajah 3.2. Makro *receive* akan dibina menggunakan kod VHDL seperti didalam lampiran.



Rajah 3.2 : Gambarajah skematik penerima.

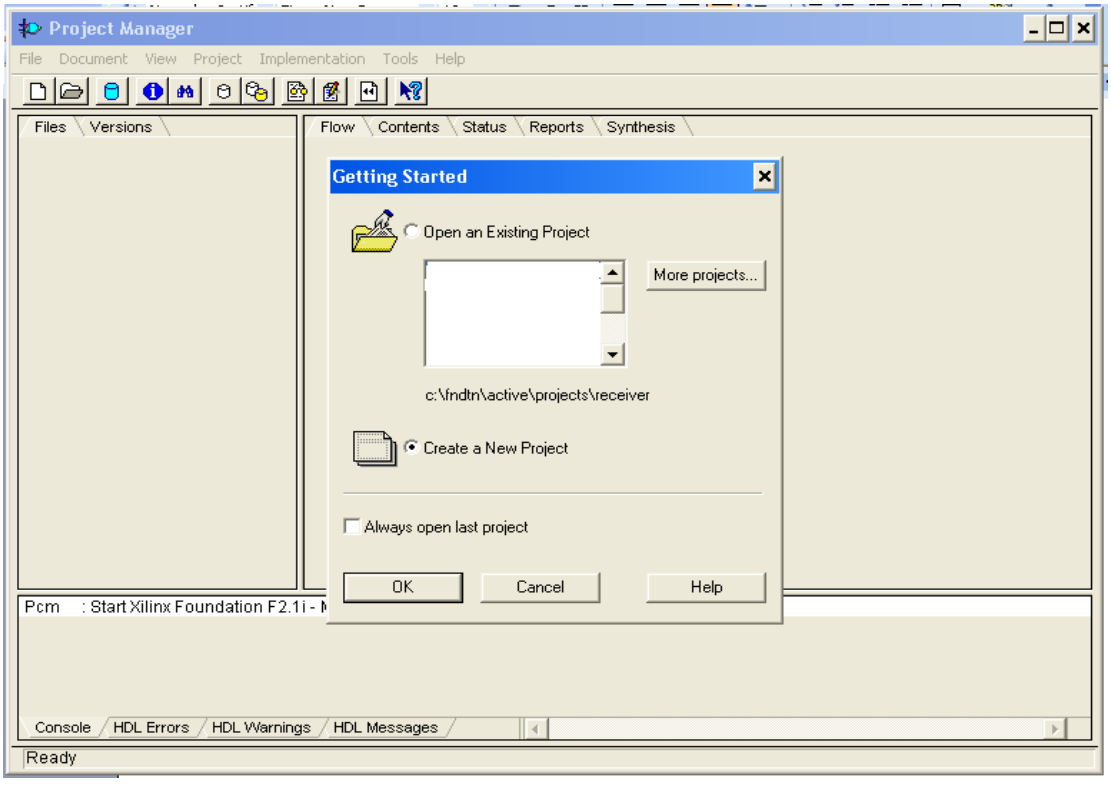
CARTA ALIR PENERIMA



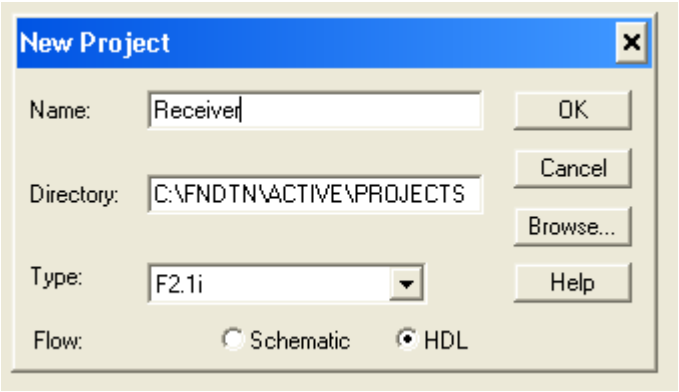
3.3 Perisian Xilinx Foundation 2.1i.

3.31 Membina Folder Simpanan Fail.

Untuk memulakan perisian Xilinx Foundation 2.1i, ikon *Project Manager* di pilih didalam *Desktop*. Paparan antaramuka seperti Rajah 3.3 akan dipaparkan. Selepas itu klik *Creat A New Project* → klik *OK* → masukan nama *Folder* untuk menyimpan data sebagai contoh nama *Folder Receiver* → klik pada HDL → klik *OK*, contoh diberikan seperti Rajah 3.4.



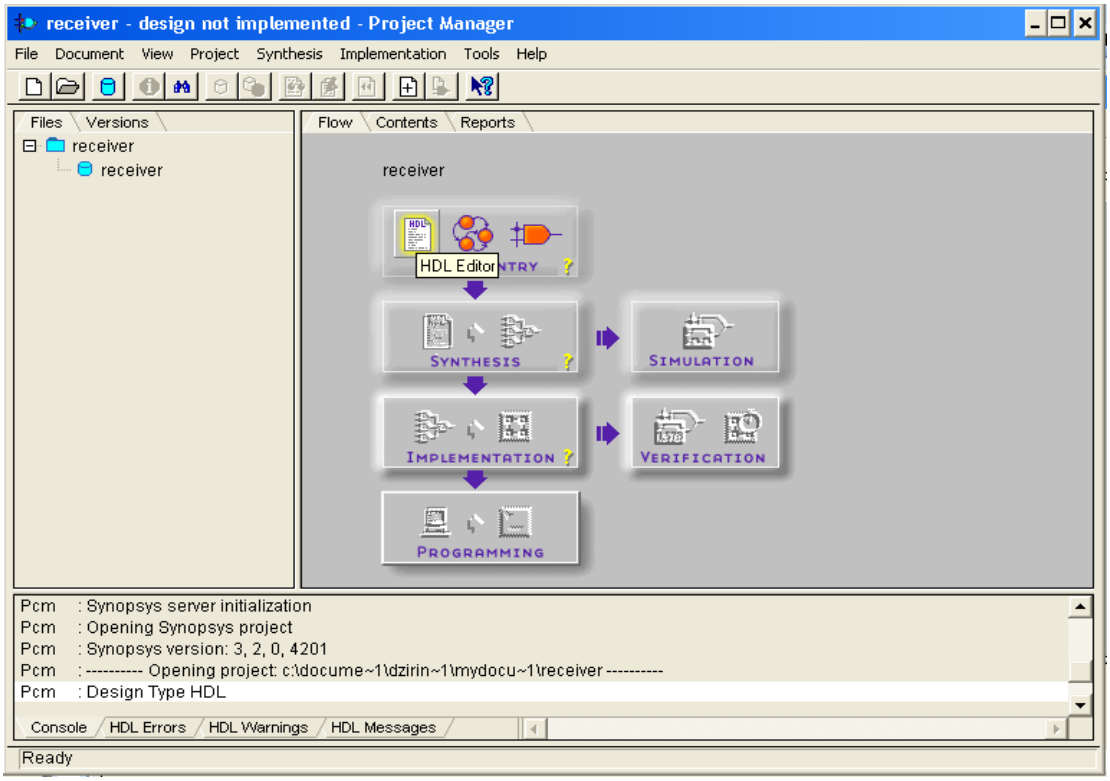
Rajah 3.3 : Permulaan Xilinx Foundation.



Rajah 3.4 : Masukan nama Folder projek.

3.32 Menggunakan HDL Editor.

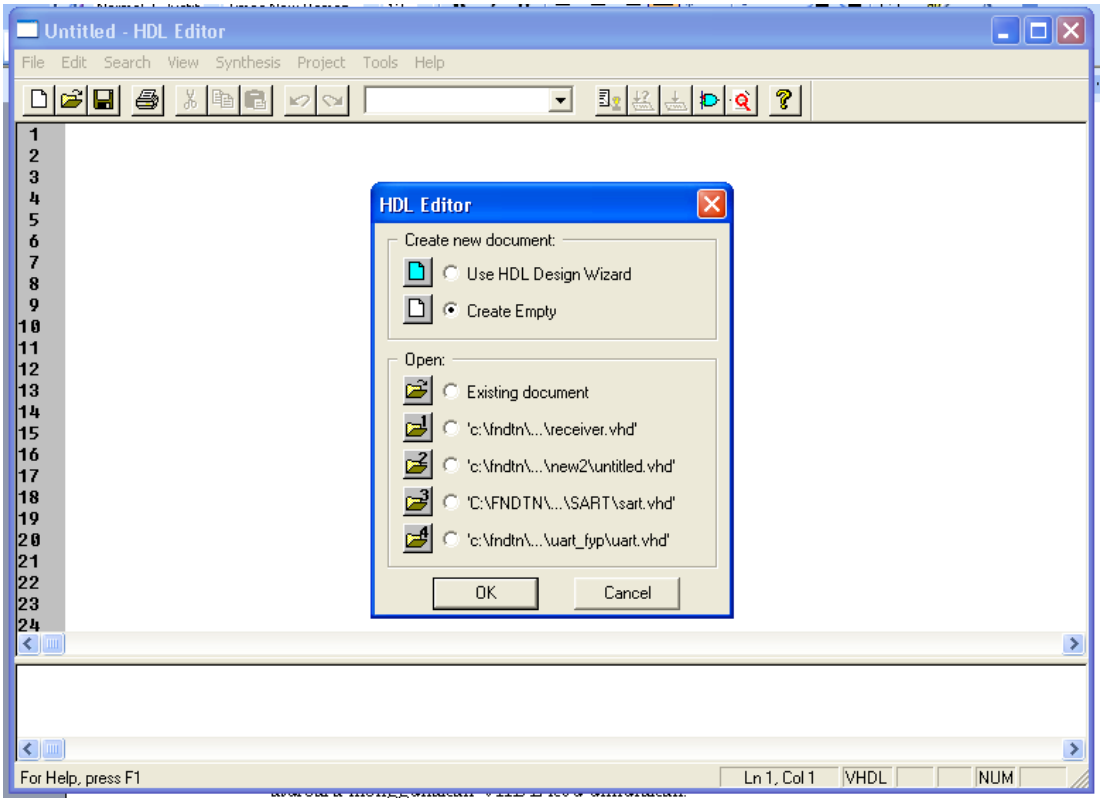
Paparan *Project Manager* seperti Rajah 2.9 didalam Bab 2 akan dipaparkan. Apabila antaramuka ini telah dipaparkan, ikon *Design Entry* bahagian *HDL Editor* dipilih. Paparan seperti Rajah 3.5 akan dipaparkan.



Rajah 3.5 : Bahagian HDL Editor dipilih.

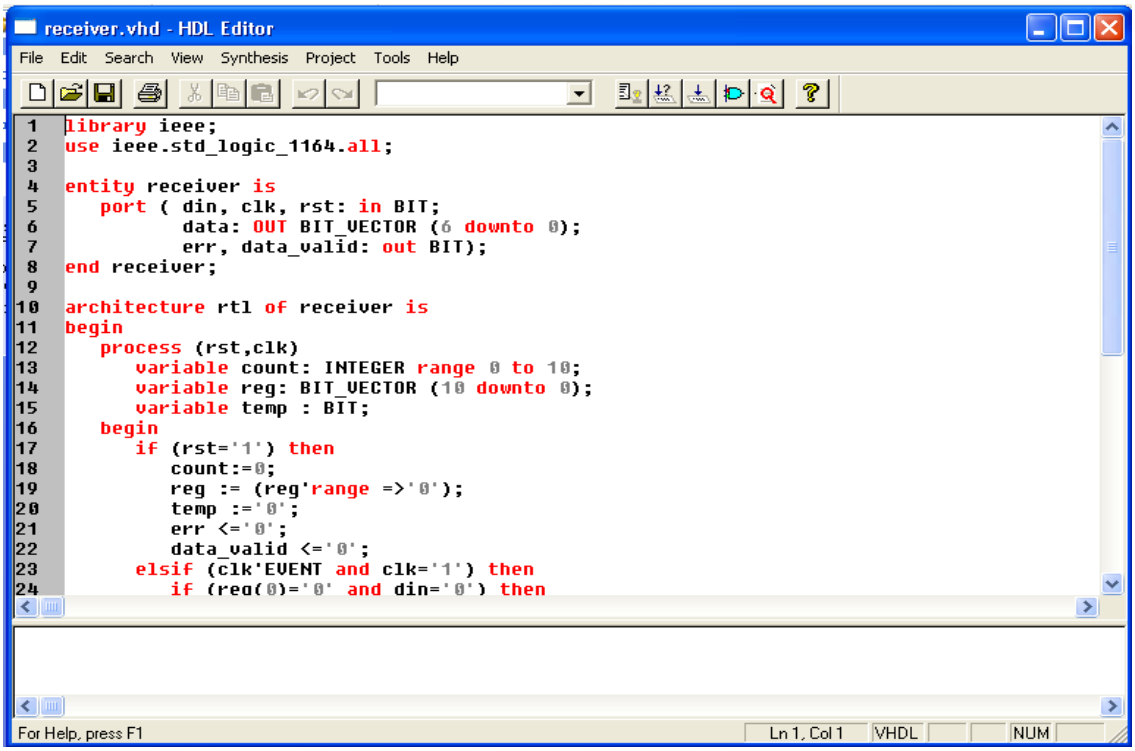
Selepas sahaja ikon HDL Editor di pilih, antaramuka seperti Rajah 3.6 untuk memasukkan aturcara menggunakan VHDL kod dimulakan.

HDL *Editor* → *Create Empty*



Rajah 3.6 : Antaramuka permulaan VHDL kod.

Selepas itu kod VHDL yang telah disiapkan dimasukan ke dalam ruang penulisan kod. Apabila kod yang diperlukan selesai dimasukan, Rajah 3.7 akan dihasilkan.



Rajah 3.7 : Kod VHDL yang selesai dimasukan.

Kod ini boleh diperiksa samaada *Syntax* yang digunakan mempunyai kesilapan atau tidak dengan mengikut kaedah ini :

Klik *Synthesis* → *Check Syntax*

Sekiranya tiada kesalahan yang dilakukan, masej “tiada kesalahan” akan dipaparkan dan sekiranya perkara sebaliknya berlaku, kesalahan yang ada hendaklah dibetulkan sehingga tiada kesalahan dikesan. Laporan untuk HDL kod yang dijana dapat dilihat dengan menggunakan :

Klik *Synthesis* → klik *View Report*.

Dengan kaedah ini juga kesalahan yang dilakukan dalam manghasilkan kod VHDL dapat diketahui. Selepas memastikan tiada kesalahan dilakukan, seterusnya makro perlu dijana dengan menggunakan arahan :

Klik ikon *Project* → *Create Macro.*

Selepas arahan ini dilakukan secara automatik, kod VHDL yang dijana akan dimasukkan ke *Project Manager* untuk diproses dalam *Synthesis* pula.

3.33 Melakukan Synthesis Ke Atas Makro.

Paparan antaramuka HDL *Editor* boleh di kecilkan seterusnya antaramuka *Project Manager* diklik. Selepas itu, proses *Synthesis* dilakukan keatas macro yang telah dijana, menggunakan arahan :

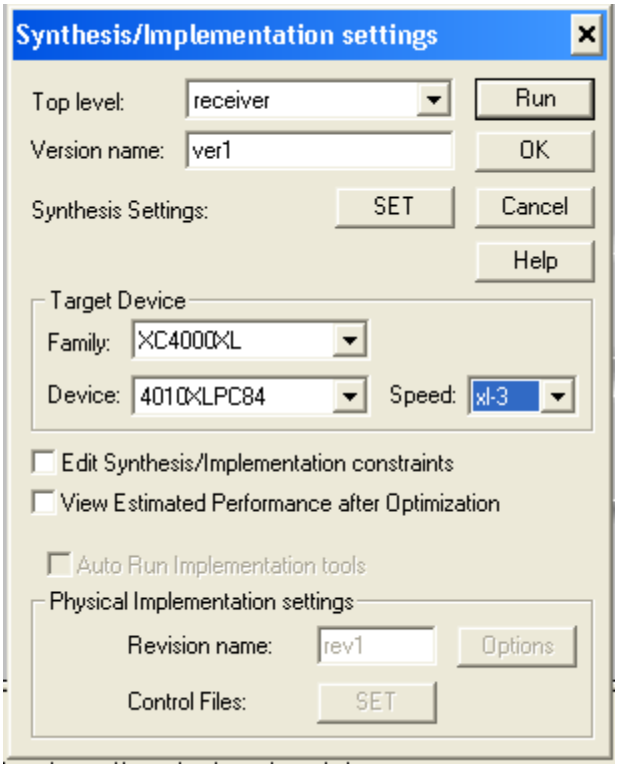
Klik ikon *Synthesis*

Selepas ikon ini diklik, paparan seperti Rajah 3.8 akan diperolehi dan perlu disetkan kepada set yang diterima oleh chip Xilinx *Development Board* yang akan digunakan. Bagi projek ini konfigurasi yang digunakan ialah :

Family : XC4000XL

Device : 4010XLPC84

Speed : xl-3



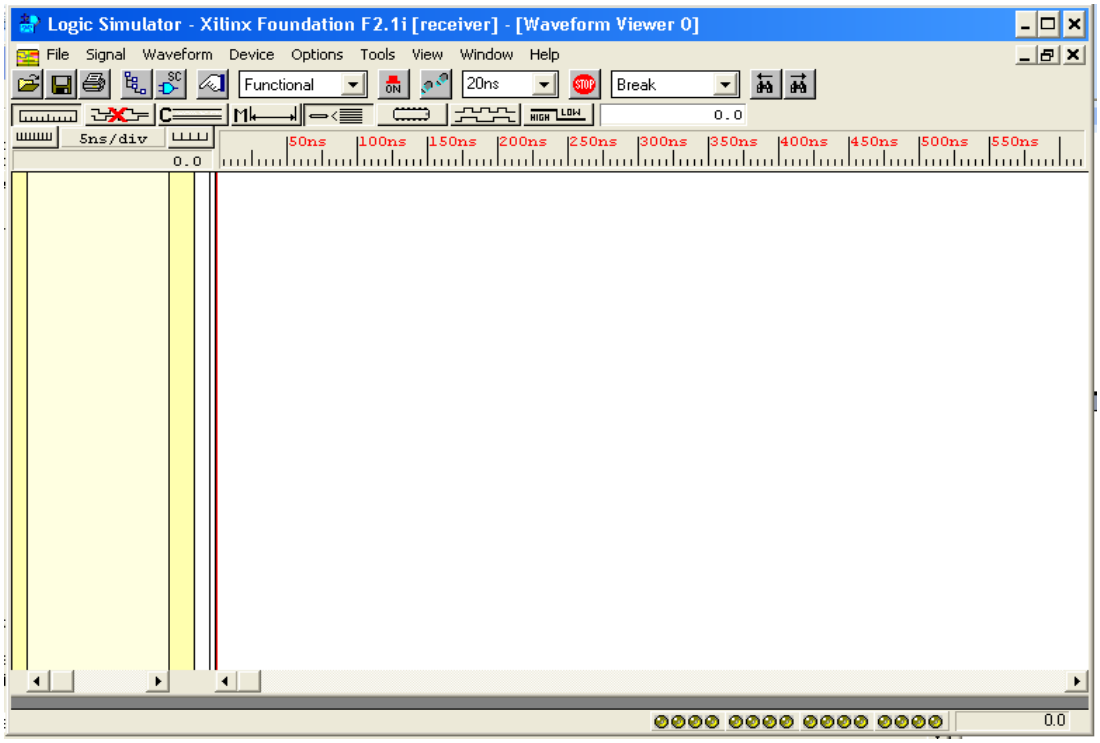
Rajah 3.8 : Konfigurasi yang digunakan untuk projek ini.

Selepas sahaja konfigurasi ini dipilih butang *Run* di klik. *Macro* yang dijana akan di sintesiskan.

3.34 Melakukan *Simulation.*

Untuk mengetahui macro yang dihasilkan adalah berfungsi atau tidak, simulasi boleh dilakukan menggunakan *Simulation* dengan mengklik butang 

. Apabila butang ini di pilih, secara automatik tetingkap simulator akan dipaparkan. Rajah 3.9 menunjukan tetingkap simulator.

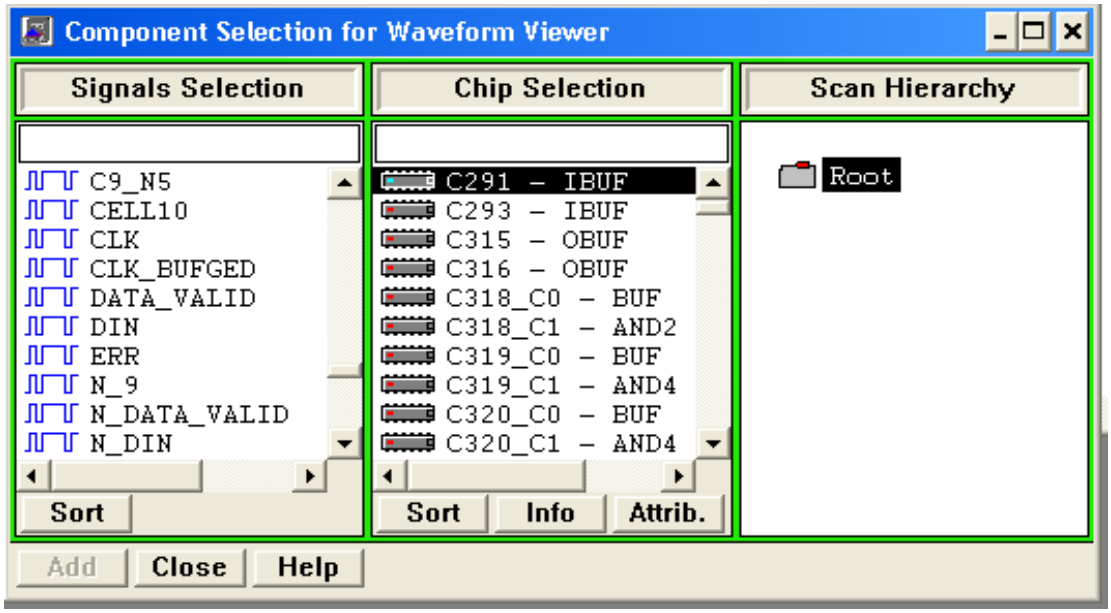


Rajah 3.9 : Tetingkap Simulator.

Apabila tetingkap simulator ini telah dipaparkan, isyarat masukan dan keluaran yang telah diisytiharkan didalam kod VHDL perlu dipaparkan. Ini dapat dilakukan dengan mengikut arahan ini :

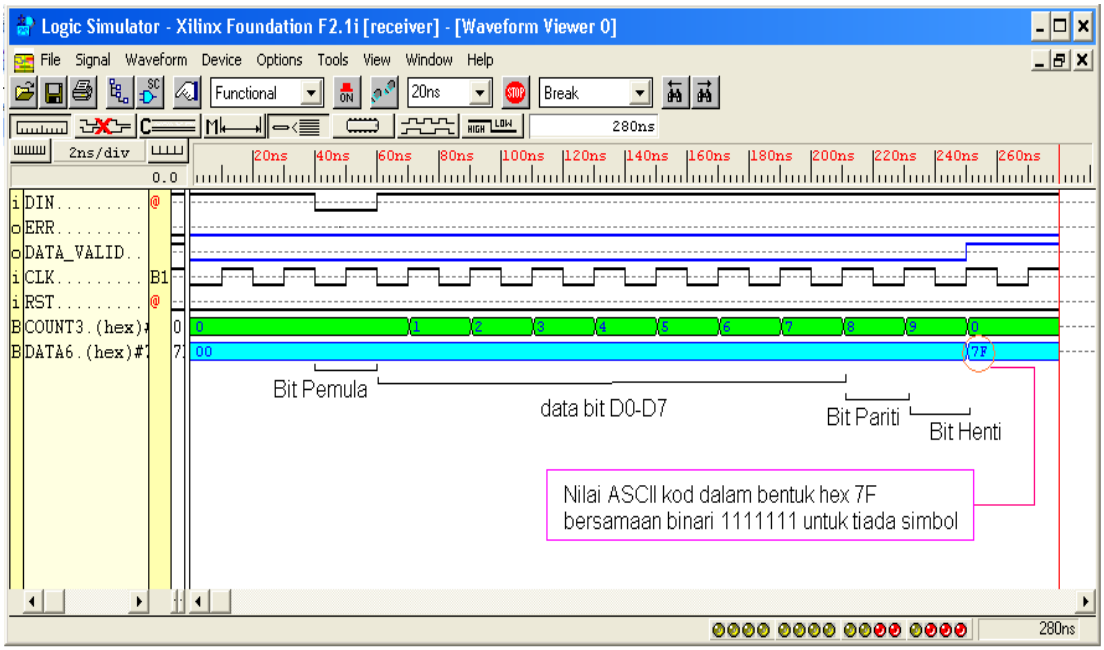
Klik butang *Signal* → *Add Signal*

Paparan seperti Rajah 3.10 akan dipaparkan seterusnya isyarat Count, Data, Clk, Data Valid, Din, Rst dan Err dipilih.



Rajah 3.10 : Memilih isyarat yang ingin disimulasikan.

Setelah memasukan ciri-ciri isyarat yang sepatutnya simulasi dilakukan dengan memastikan parameter yang sepatutnya telah disetkan seperti *Simulation Step* dan sebagainya. Sekiranya semua ini telah disetkan dengan betul, simulasi boleh dilakukan dan hasilnya akan diperolehi seperti Rajah 3.11.



Rajah 3.11 : Hasil simulator untuk fungsi penerima (Receiver).

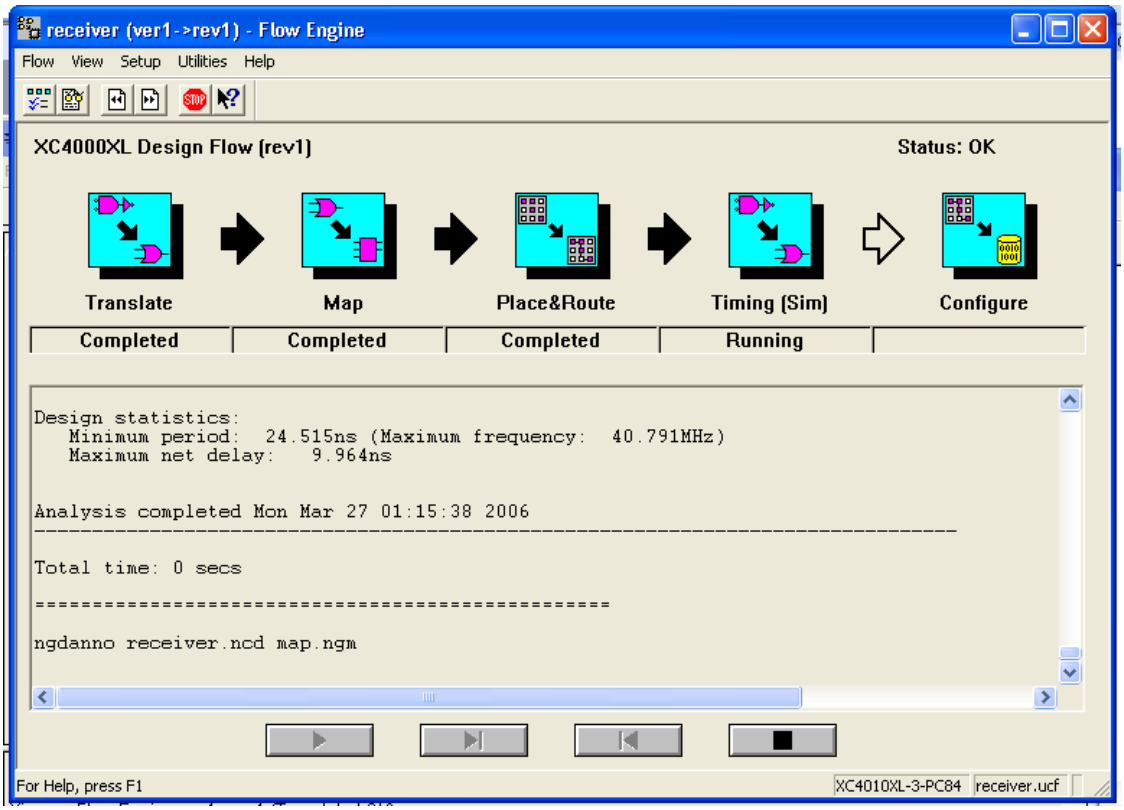
Sekiranya hasil simulasi ini mengikut spesifikasi keluaran yang dikehendaki, ini menunjukan VHDL kod yang dibina boleh diterima dan langkah seterusnya dilakukan.

3.35 Melakukan Perlaksanaan.

Selepas kod VHDL yang dimasukan telah disintesis, seterusnya implimentasi perlu dilakukan keatas makro. Kaedah ini adalah untuk perisian Xilinx Foundation 2.1i melakukan *Net List* serta *Route* iaitu melakukan laluan litar berasaskan get-get logik untuk dilaksanakan dari kod VHDL yang dibina. Ini dapat dilakukan dengan membuka tetingkap *Project Manager* seterusnya klik pada *Implementation*. Seterusnya tetingkap

sperti Rajah 3.12 akan dipaparkan. Proses ini akan melibatkan masa lebihkurang 30 saat untuk diproses. Hasil dari perlaksanaan ini juga boleh disimulasi untuk memeriksa makro yang telah dijana dalam bentuk get logik. Proses simulasi ini kali ini akan melibatkan fungsi masa. Simulasi ini adalah lebih penting kerana ia akan menunjukan masa lengah dalam makro yang dijana. Kaedah untuk perlaksanaan adalah :

Klik *Implementation* → Klik *Run*



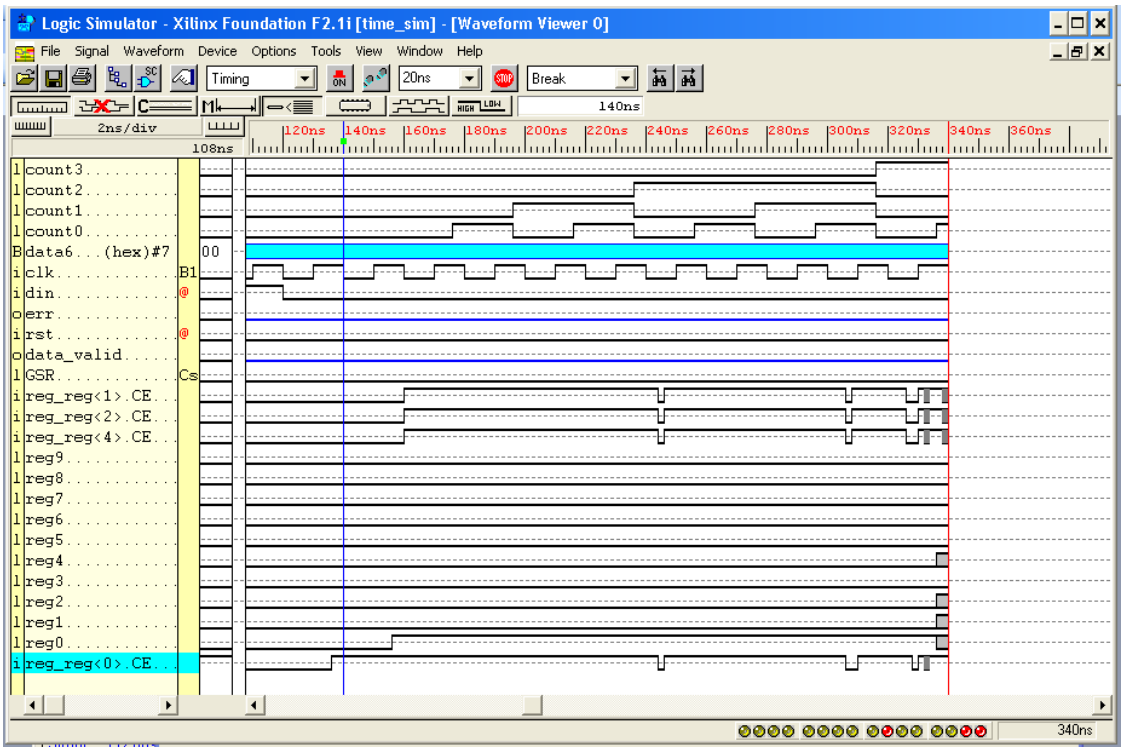
Rajah 3.12 : Rajah simulasi untuk Implementation.

3.36 Kaedah Verifikasi.

Verifikasi ini adalah penting untuk menguji litar yang dibina bagi memastikan tiada kesalahan yang berlaku. Proses ini penting untuk menguji litar dari segi masa kerana ini akan menunjukan masa lengah sekiranya ada. Untuk verifikasi, arahan ini perlu dilakukan :

Klik pada *Verification – Timing Simulation* → Masukan *Signal*

Rajah 3.13 menunjukan simulasi masa yang dilakukan.



Rajah 3.13 : Gambarajah simulasi fungsi masa.

3.37 Analisa Masa.

Bahagian ini akan menunjukkan seluruh laporan simulasi yang dibuat serta analisa lengah masa yang ada dalam litar yang dibuat. Sekiranya ada lengah masa yang tinggi, ini perlu di baiki kerana ia akan mengakibatkan litar atau chip yang dibina akan mengalami masalah apabila digunakan untuk mengujinya diatas Xilinx Development Board. Bahagian ini perlu diteliti dengan sebaiknya supaya chip yang direkabentuk dapat beroperasi dengan sebaik mungkin. Laporan ini adalah dalam bentuk teks dokumen . Untuk memperolehi butiran laporan ini klik pada ikon *Timing Analyzer* .

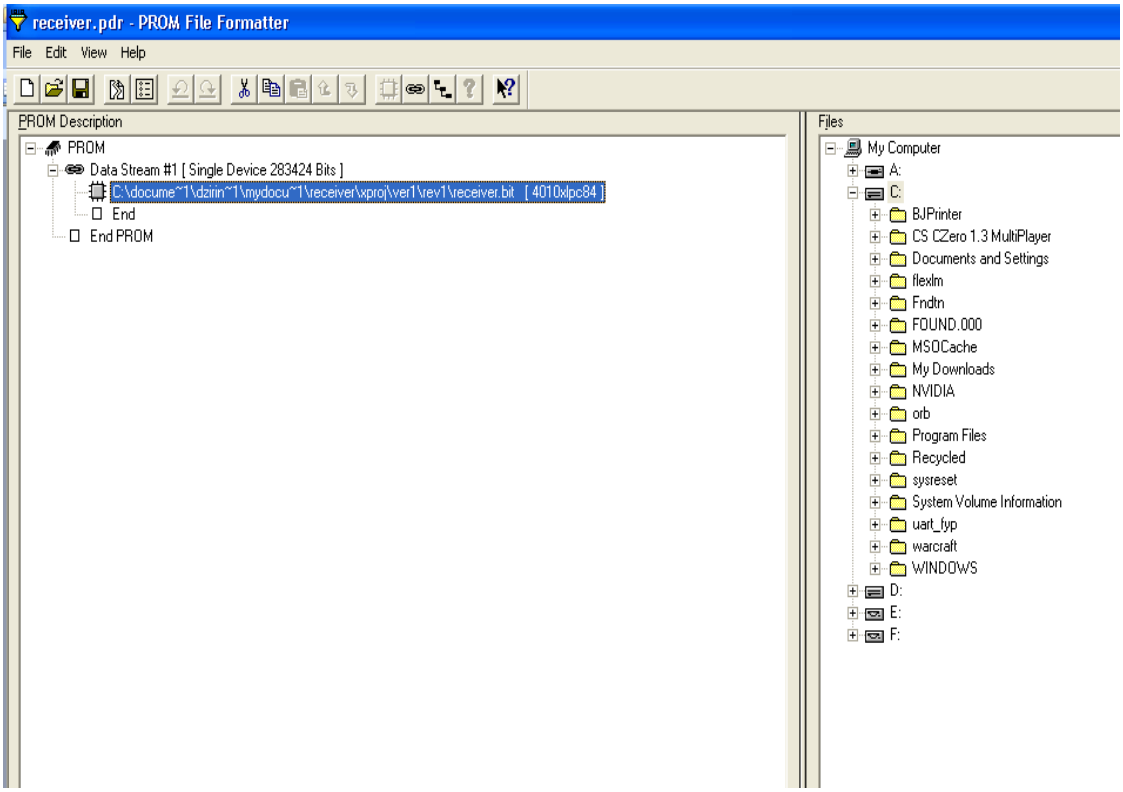
3.38 Programming.

Proses yang terakhir sekali adalah untuk memprogramkan semua makro yang telah dijana oleh Xilinx Foundation 2.1i kedalam EPROM melalui kaedah "*burn data*" kedalam EPROM menggunakan perisian Chip Max Data Loader melalui EPROM "*burner*". Proses ini hanya boleh dilakukan sekiranya semua simulasi-simulasi yang dilakukan menghasilkan keluaran yang disahkan oleh perisian Xilinx Foundation 2.1i. Untuk projek ini EPROM yang digunakan adalah UVEPROM ST M27C512. UVEPROM perlu dipastikan tiada data didalamnya. Untuk memadam data didalam

UVEPROM ST M27C512, chip ini perlu didedahkan kepada cahaya UV selama 10 – 15 minit. Arahan yang perlu dilakukan meghasilkan data yang boleh disimpan didalam UVEPROM ialah :

Klik ikon *Programming* → pilih PROM *File Formatter* → klik *OK*.

Tetingkap PROM *File Formatter* akan dipaparkan seperti Rajah 3.14

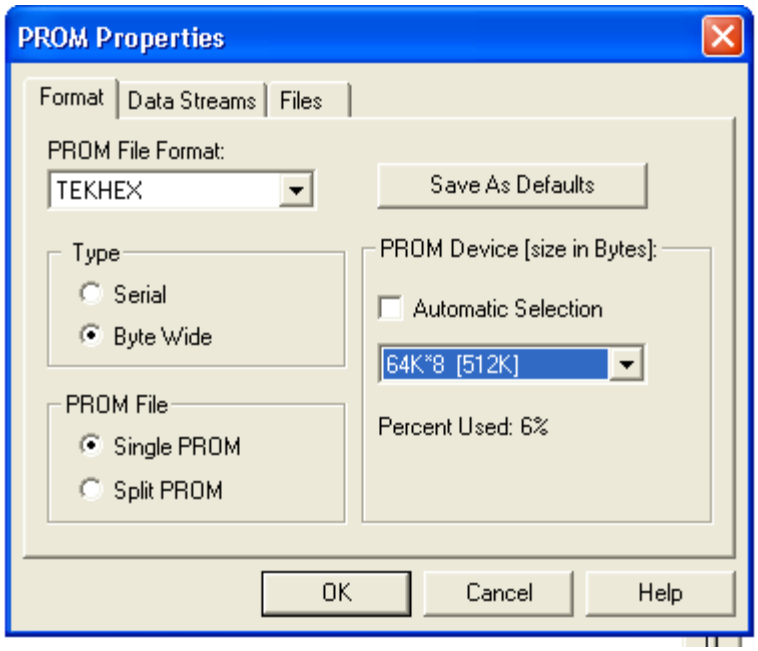


Rajah 3.14 : Paparan PROM File Formatter.

Seterusnya,

klik *File* → PROM *Properties* → tukar PROM *File Format* kepada TEKHEX → Pilih *Byte Wide* → Pilih *Single* PROM → Pilih PROM *Device* 64k*8(512) → Klik *OK*.

Didalam PROM *Device* 8k*8(64k) dipilih. Selepas itu, fail yang telah dihasilkan disimpan seterusnya fail yang berakhir dengan **.tekhex** sahaja akan disimpan didalam UVEPROM menggunakan perisian Chip Max Data Loader. Rajah 3.15 menunjukan konfigurasi yang perlu disetkan untuk menyimpan data dalam format TEKHEX.

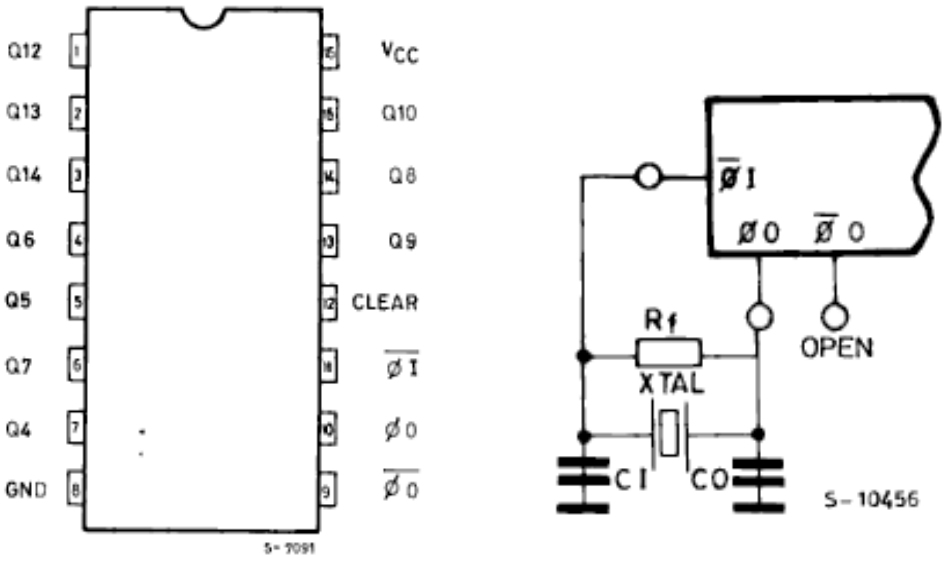


Rajah 3.15 : Konfigurasi untuk menyimpan data TEKHEX.

Akhir sekali UVEPROM yang telah dimuat turun dengan data TEKHEX boleh dimuatkan kedalam papan demo Xilinx dan sedia untuk diuji.

3.4 Membina Litar Kadar Baud Dan Max 232.

Litar kadar baud luaran dibina menggunakan litar bersepadu SN74HC4060N. Litar dibina seperti didalam Rajah 3.16.



Rf = 100k Ohm

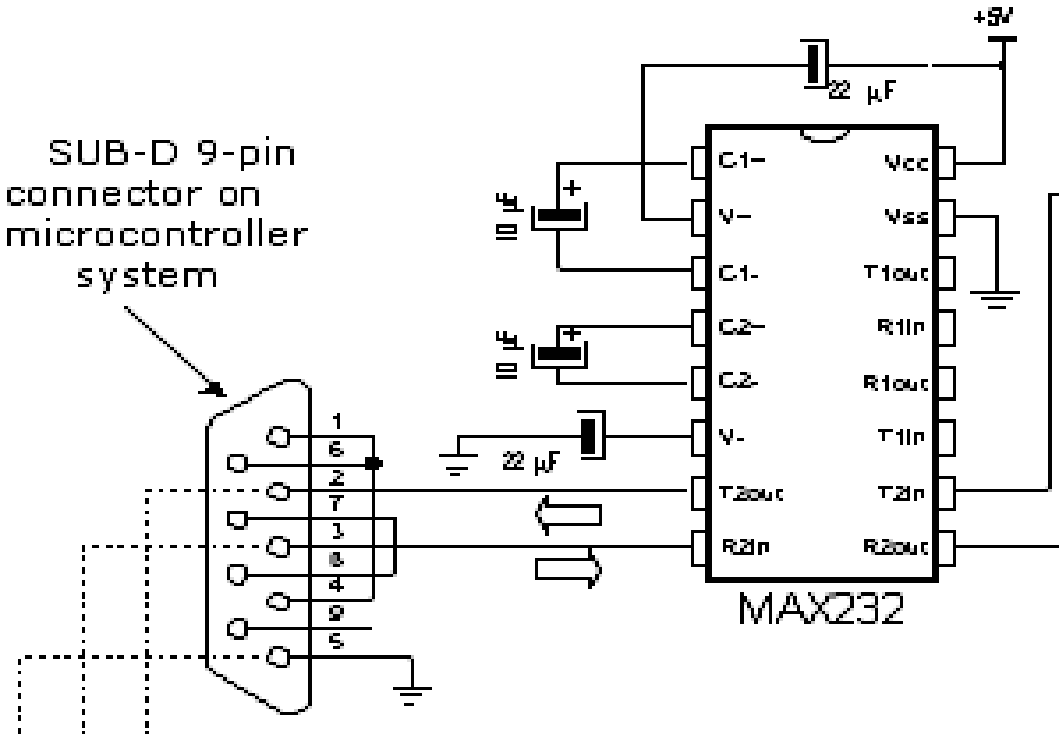
C1 = 22p F

C2 = 100p F

XTAL = 2.5 MHz.

Rajah 3.16 : Rajah skematik litar kadar baud.

Litar RS232 juga dibina untuk menghasilkan sambungan RS232. Litar seperti Rajah 3.17 dibina.



Rajah 3.17 : Litar RS232.

3.5 Menguji Penerima (*Receiver*) Menggunakan Papan Demo Xilinx.

Setelah data dimasukkan kedalam UVEPROM, ia akan dimuatkan kedalam soket EPROM diatas papan demo Xilinx (*Xilinx Development Board*). Port-port yang telah disytiharkan didalam perisian Xilinx Foaundation 2.1i dikenal pasti untuk disambungkan dengan wayar. Port yang telah disytihar ditunjukan didalam *Pad Report* seperti dibawah :

Input file: map.ncd
Output file: receiver.ncd

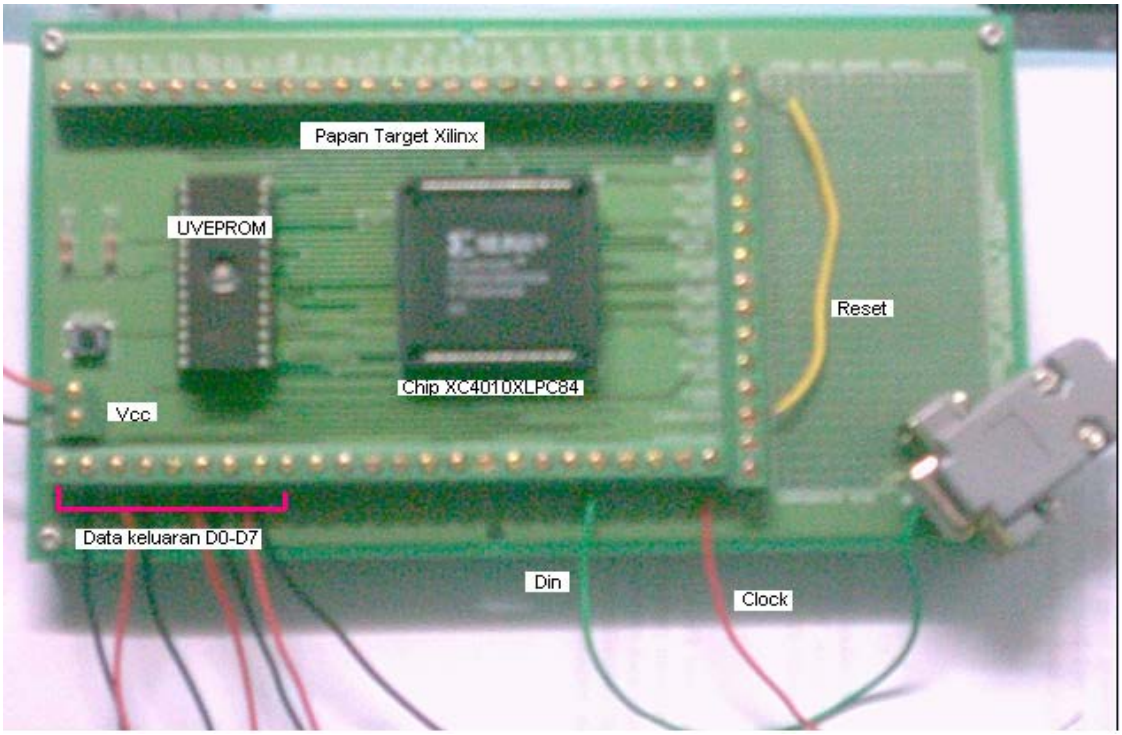
Part type: xc4010xl
Speed grade: -09
Package: pc84

Mon Mar 27 02:45:54 2006

Pinout by Pin Name:

Pin Name	Direction	Pin Number
clk	INPUT	P13
data<0>	OUTPUT	P82
data<1>	OUTPUT	P81
data<2>	OUTPUT	P5
data<3>	OUTPUT	P3
data<4>	OUTPUT	P4
data<5>	OUTPUT	P83
data<6>	OUTPUT	P84
data_valid	OUTPUT	P15
din	INPUT	P7
err	OUTPUT	P18
rst	INPUT	P6

Port-port ini akan digunakan didalam sambungan diantara papan target Xilinx ke komputer. Sambungan dilakukan seperti didalam Rajah 3.18



Rajah 3.18 : Papan demo Xilinx XC4010PC84 dan sambungannya.

Setelah semua makro dijana, penerima yang dibina berasaskan FPGA akan diuji menggunakan papan demo Xilinx XC4010PC84. Sambungan semua litar akan dilakukan seperti didalam Rajah 3.18 dimana papan demo Xilinx akan disambungkan dengan litar kadar baud, litar RS232, :LED dan seterusnya dihubungkan dengan komputer melalui kabel RS232.

BAB 4

KEPUTUSAN DAN ANALISIS

4.1 Pengenalan.

Bahagian ini adalah bahagian yang penting dalam penghasilan projek ini. Setiap keputusan yang diperolehi menunjukkan keupayaan rekabentuk penerima (*Receiver*) yang direka. Daripada projek yang dijalankan, semua bahagian akan digabungkan dari papan demo Xilinx. Pengulangan ujian dan ubahan rekabentuk perlu dilakukan bagi memastikan komponen penerima (*Receiver*) yang dibina boleh beroperasi dengan sepenuhnya.

4.2 Keputusan Untuk Rekabentuk (*Design Entry*) Dan Sintesis (*Synthesis*).

Setelah kod HDL dijana ia kemudiannya diperiksa melalui *Check Syntax*. Hasil dari pemeriksaan ini didapati tiada kesalahan berlaku dari segi penggunaan arahan-arahan HDL. Ini bermaksud kod HDL yang dibina boleh diterima dan makro boleh dijana.

Setelah makro telah dijana, sintesis dilakukan keatas makro. Sintesis ini juga berjaya dilakukan tanpa ada kesilapan. Simulasi dilakukan untuk bahagian fungsi dari makro yang dibina. Hasilnya penerima yang direka berasaskan kod HDL dapat berfungsi dengan baik bagi setiap data yang dimasukkan menggunakan simulator samaada untuk data yang sah, tidak sah dan keadaan tidak dibenarkan. Ini menunjukan penerima yang dibina mungkin dapat berfungsi dengan baik.

4.3 Keputusan Verifikasi Dari Perlaksanaan.

Selepas proses *Implementation* dilakukan, proses verifikasi pula dilaksanakan. Walaupun hasil dari proses implimentasi berjaya dilaksanakan tanpa ada kesilapan namun pada peringkat verifikasi untuk analisa masa (*Timing Analyzer*) didapati ada kesilapan berlaku. Kesilapan ini berlaku kerana lengah masa yang terhasil dari proses analisa masa yang dilakukan. Lengah masa ini berlaku disebabkan lengah yang terhasil dari get-get logik yang dijana. Walaubagaimanapun kelemahan lengah masa ini dapat diatasi dengan merendahkan frekuensi jam (Clk). Nilai jam ini sebenarnya adalah untuk konfigurasi kadarbaud supaya penerima yang direka mempunyai kadarbaud yang sama dengan komputer. Sekiranya kadarbaud ini bersamaan dengan kadarbaud komputer, maka perhubungan data dapat dilakukan dengan baik. Kadarbaud juga boleh disetkan bagi komputer. Kadarbaud yang boleh disetkan didalam komputer adalah 110, 150, 300, 600, 1200, 4800, 9600,19200 dan banyak lagi.

Dari projek ini didapati frekuensi maksimum yang boleh diterima oleh CLK penerima ialah 46.038 MHz. Konfigurasi blok logic (CLB) bagi projek ini adalah 28 daripada 400 CLB iaitu hanya 7% CLB digunakan. Laporan keseluruhan projek boleh dilihat didalam lampiran.

4.4 Hasil Sambungan Litar.

Setelah sambungan litar siap dibina didapati apabila ada data yang dihantar maka penerima akan menerima data yang dihantar dan LED akan menyala. Kadangkala berlaku masalah dimana apabila data dihantar tetapi penerima tidak dapat menafsirkan data yang diterima seterusnya mengeluarkan ralat dimana LED tidak menyala. Masalah ini telah dikaji dengan teliti dan kemungkinan yang berlaku adalah kadarbaud yang dijana oleh penerima tidak sama dengan komputer. Ini mungkin berlaku kerana penggunaan penjana kadarbaud luaran. Tetapi masalah ini akan cuba diatasi dan diperbaiki supaya penerima dapat berfungsi dengan sebaiknya.

BAB 5

PENUTUP

5.1 Kesimpulan.

Secara umumnya projek yang dijalankan ini adalah bertujuan untuk merekabentuk dan membina bahagian penerima UART berasaskan FPGA menggunakan perisian Xilinx Foundation 2.1i. Penerima ini akan berfungsi sama seperti penerima yang ada didalam UART (8251). Ia akan berupaya menerima data dari komputer secara siri seterusnya mentafsirkan data tersebut didalam keadaan selari untuk alatan komputer menerima data tersebut.

Kaedah yang boleh digunakan bagi merekabentuk penerima terdiri dari 3 kaedah iaitu menggunakan rekabentuk skematik, gambarajah keadaan (FSM) dan penggunaan kod HDL. Tetapi bagi projek ini kaedah yang digunakan adalah menggunakan HDL kod. Kod HDL digunakan kerana ia mudah untuk diadaptasikan dan diubahsuai mengikut keperluan. Penerima ini dapat menerima data untuk 8 bit data secara siri dan menukarkannya kepada data dalam bentuk selari. Bagi projek ini mikropengawal digunakan untuk menjadi antaramuka bagi penerima dan LED. Ianya berfungsi untuk

menunjukan bahawa data telah diterima atau tidak melalui LED menyala atau tidak menyala.

5.2 Masalah Yang Dihadapi.

Sepanjang tempoh perlaksanaan projek ini terdapat beberapa masalah yang dihadapi antaranya ialah :

a. Memahami aplikasi dan penggunaan perisian.

Masalah ini hanya berlaku pada peringkat awal projek dijalankan. Buku-buku panduan berkenaan dengan perisian Xilinx Foundation dirujuk untuk menyelesaikan masalah ini. Penyelia juga menjadi sumber penting dalam memahami aplikasi perisian. Buku-buku rujukan yang berkaitan dengan Xilinx Foundation pula tidak banyak disediakan oleh pihak perpustakaan. Oleh itu sumber rujukan yang ada sangat terhad.

b. Kesalahan dari kod HDL.

Perkara ini juga selalu berlaku dimana penggunaan arahan yang salah ataupun tidak mengikut piawaian struktur kod HDL. Kesalahan ini selalu berlaku kerana kurang pemahaman dan pengamalan dalam penggunaan kod HDL. Ini dapat dielakan dengan merujuk buku-buku berkaitan kod VHDL dan penggunaannya.

c. Kesilapan dalam menyambung terminal wayar.

Pada peringkat untuk menyambung wayar ke komponen, kesilapan seharusnya tidak boleh berlaku. Ini kerana sekiranya terminal wayar yang disambung adalah tidak betul ia akan menyebabkan kesilapan dari segi penerimaan data dan juga boleh merosakan komponen elektronik. Perkara ini perlu dielakkan dan patut diberi perhatian supaya masalah kesilapan menyambung wayar tidak berlaku.

5.3 Cadangan Memajukan Projek.

Bagi memajukan projek ini, cadangan-cadangan berikut perlu diambilkira supaya projek ini dapat diperbaiki dan penerima ini boleh beroperasi dengan baik. Dalam projek ini, penggunaan penjana kadar baud dalaman yang dijana oleh FPGA adalah lebih baik serta kadar baud boleh ditentukan mengikut kadar baud yang dikehendaki melalui pin luaran. Oleh itu Clock masukan yang digunakan hanya akan memastikan penjana kadar baud menghasilkan kadar baud yang sesuai. Kesilapan sambungan litar kadar baud luaran mungkin menyebabkan penerima tidak dapat beroperasi dengan baik.

Kod VHDL yang dibangunkan juga perlu menitikberatkan setiap keadaan yang sah dan tidak sah. Setiap litar sampingan juga perlu diperiksa dengan teliti supaya litar lengkap dapat beroperasi dengan sepenuhnya.