UNIVERSITAS INDONESIA

PERANCANGAN *VLSI* 0.25µm DENGAN DESAIN *HYBRID VHDL* BERBASIS *FPGA XILINX SPARTAN* 3 UNTUK *CPU OCEAN BOTTOM UNIT TSUNAMI EARLY WARNING SYSTEM*

TESIS

RIYANTO
1006788870

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK ELEKTRO
DEPOK
JUNI 2012

UNIVERSITAS INDONESIA

PERANCANGAN *VLSI* 0.25µm DENGAN DESAIN *HYBRID VHDL* BERBASIS *FPGA XILINX SPARTAN* 3 UNTUK *CPU OCEAN BOTTOM UNIT TSUNAMI EARLY WARNING SYSTEM*

TESIS

Diajukan sebagai salah satu syarat untuk memperoleh gelar Magister Teknik

RIYANTO
1006788870

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK ELEKTRO
KEKHUSUSAN CAD VLSI
JUNI 2012

HALAMAN PERNYATAAN ORISINALITAS

Tesis ini adalah hasil karya saya sendiri,
dan semua sumber baik yang dikutip maupun yang dirujuk
telah saya nyatakan dengan benar

Nama : Riyanto
NPM : 1006788870
Tanda Tangan :

Tanggal : 21 Juni 2012

HALAMAN PENGESAHAN

Tesis ini diajukan oleh :

Nama	:	Riyanto
NPM	:	1006788870
Program Studi	:	Teknik Elektro
Judul Tesis	:	Perancangan *VLSI* 0.25 µm dengan desain *Hybrid VHDL* berbasis *FPGA Xilinx Spartan* 3 untuk *CPU Ocean Bottom Unit Tsunami Early Warning System*

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Magister Teknik pada Program Studi Teknik Elektro, Fakultas Teknik, Universitas Indonesia.

DEWAN PENGUJI

Pembimbing	**: Prof. Dr. Ir. Harry Sudibyo, DEA**	( )
Penguji	**: Ir. Purnomo Sidi P. M.Sc., Ph.D.**	( )
Penguji	**: Dr. Ir. Retno Wigajatri P. MS**	( )
Penguji	**: Dr. Ir. Agus Santoso Tamsir, MT**	( )

Ditetapkan di : Depok

Tanggal : 21 Juni 2012

UCAPAN TERIMA KASIH

Penulis mengucapkan terimakasih yang sebesar-besarnya kepada semua pihak yang telah berperan dalam menyelesaikan Tesis ini. Untuk itu, penulis mengucapkan terimakasih yang sebesar-besarnya kepada :

1. Bapak Prof. Dr. Ir. Bambang Sugiarto, M.Eng. selaku Dekan Fakultas Teknik Universitas Indonesia.
2. Bapak Dr. Ir. M. Asvial, M.Eng selaku Ketua Departemen Teknik Elektro Fakultas Teknik Universitas Indonesia.
3. Bapak Prof. Dr. Ir Harry Sudibyo, DEA, selaku dosen pembimbing pertama, yang telah memberikan koreksi, bimbingan dan pengarahan.
4. Dewan penguji tesis Ir. Purnomo Sidi P. M.Sc., Ph.D., Dr. Ir. Retno Wigajatri P. MS, Dr. Ir. Agus Santoso Tamsir, MT.
5. Segenap staf pengajar Jurusan Teknik Elektro yang telah memberikan ilmu dan pengetahuan pada penulis.
6. Segenap staf tata usaha dan karyawan Jurusan Teknik Elektro.
7. Istriku Tri Wahyuningsih dan kedua anakku Dhiyaulhaq Syifa Riyanti dan Furqon Ghazi Fikriyanto, bapak dan ibu tersayang, adik, kakak dan keponakanku, serta seluruh saudara dan keluargaku yang selalu memberikan dorongan serta bantuan baik material atau spiritualnya.
8. Teman-teman seangkatan, terutama Bapak Didik, Bapak Nanang, Bapak Wahyu, Bapak Haris, Bapak Khoirul, Bapak Rudi, Ibu Ima atas bantuan dan dorongannya.
9. Semua pihak yang telah membantu terselesaikannya tugas akhir ini.

Akhir kata penulis berharap semoga laporan ini dapat bermanfaat bagi semua pihak.

Depok, 21 Juni 2012

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan dibawah ini :

Nama	: Riyanto
NPM	: 1006788870
Program Studi	: CAD VLSI
Departemen	: Teknik Elektro
Fakultas	: Teknik
Jenis Karya	: Tesis

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif** ***(Non-exclusive Royalty Free Right)*** atas karya ilmiah saya yang berjudul :

PERANCANGAN *VLSI* 0.25 µm DENGAN DESAIN *HYBRID VHDL* BERBASIS *FPGA XILINX SPARTAN* 3 UNTUK *CPU OCEAN BOTTOM UNIT TSUNAMI EARLY WARNING SYSTEM*

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (database), merawat, dan mempublikasikan tesis saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai hak cipta.

Demikianlah pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok

Pada tanggal : 21 Juni 2012

ABSTRAK

Nama : Riyanto
Program Studi : CAD VLSI
Departemen : Teknik Elektro

Judul : Perancangan VLSI 0.25 μm dengan desain hybrid VHDL berbasis FPGA Xilinx Spartan 3 untuk CPU Ocean Bottom Unit Tsunami Early Warning System

Perancangan VLSI dengan *hybrid* VHDL merupakan metode desain untuk menghasilkan *Sistem On Chip* yang berbasis FPGA Xilinx Spartan 3. Sistem yang di desain adalah arsitektur CPU yang terdapat di *Ocean Bottom Unit* (OBU) *Tsunami Early Warning System*. Proses desain di implementasikan pada FPGA *board* Xilinx Spartan 3.

Perancangan VLSI CPU OBU dengan metode *hybrid* VHDL di lakukan dengan urutan proses desain yaitu membuat kode VHDL untuk menyimpan data pengukuran dan mengolah dengan algoritma mofjeld, Mengubah kode VHDL menjadi RTL, Mengubah RTL menjadi schematic dan kode verilog, Mengubah verilog menjadi CMOS layout, Menggunakan kode VHDL sebagai *configure device* pada XC3S200, *genetrate* PROM file pada XCF02S.

Hasil rancangan adalah VLSI 0,25 μm pada CPU OBU dengan jumlah gerbang logika yang digunakan sebanyak 699 buah dan 347 buah flipflop. Sedangkan dalam teknologi VLSI kapasitas adalah 10k -1M. Dengan metode *hybrid* VHDL jumlah gate pada desain CPU OBU masih dapat ditingkatkan dengan cara meningkatkan memori simpan sebanyak mungkin.

Kata Kunci : VLSI, VHDL, CMOS layout, FPGA, Xilinx Spartan, TEWS

ABSTRACT

Name : Riyanto
Study Program : CAD VLSI
Department : Electrical Engineering

Title : VLSI design 0.25 μm of a hybrid design with VHDL Xilinx Spartan 3 FPGA-based for Ocean Bottom Unit CPU Tsunami Early Warning System

VLSI design with a hybrid VHDL is a design methods to produce a System On Chip based on CMOS layout. The designed system is CPU architecture located on Ocean Bottom Unit Tsunami Early Warning System. The design process implemented on Xilinx Spartan 3 FPGA board.

Design of VLSI OBU CPU with a hybrid VHDL method is done by order of the design process is to make VHDL code for storing and processing the measurement data with the algorithm mofjeld, Changing the VHDL code into RTL, Changing RTL into schematic and verilog file, Changing verilog code into CMOS layout, Using the VHDL code as configure devices on the XC3S200, generating PROM files on XCF02S Xilinx Spartan.

The design results is VLSI 0,25 μm in CPU OBU with 699 logic gates and 347 flip-flops. While in VLSI technology the capacity is 10k-1M. With a hybrid method the gate of CPU OBU can be increased by increasing the memory as much as possible.

Key words: VLSI, VHDL, CMOS layout, FPGA, Xilinx Spartan, TEWS

DAFTAR ISI

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

DAFTAR LAMPIRAN

DAFTAR SINGKATAN

TEWS	: Tsunami Early Warning System
BPR	: Bottom Pressure Recorder
CMOS	: Complementary Metal Oxide Semiconductor
FPGA	: Field Programmable Gate Array
DSCH	: Design a Schematic diagram
RTL	: Register Transfer Language
VHDL	: VHSIC Hardware Description Language
ISE	: Integrated Software Environment
CLB	: Configurable Logic Block
LUT	: Look-Up Tables
ROM	: Read Only Memory
RAM	: Read Acess Memory
ALU	: Aritmatic Logic Unit
CPU	: Control Processing Unit
OBU	: Ocean Bottom Unit
VLSI	: Very Large Scale Integrated
DART	: Deep-ocean Assessment and Reporting of Tsunami

DAFTAR ISTILAH/SIMBOL

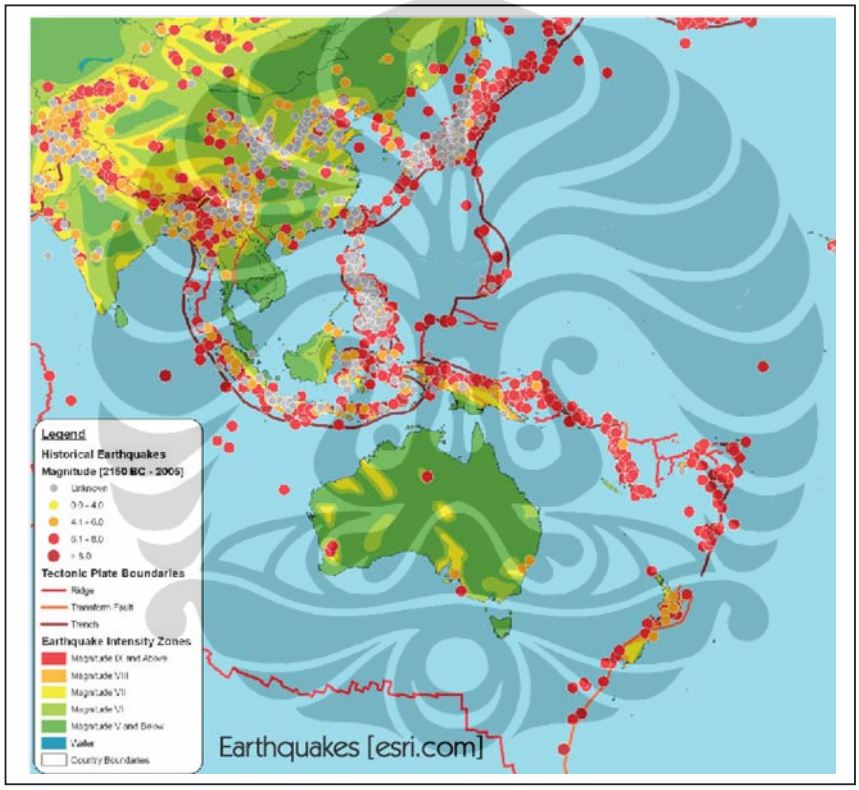
φ	:	Beda fase
t	:	Waktu pergerakan gelombang
d	:	Jarak antar dua sensor
VR	:	Gelombang Rayleigh
λ	:	Panjang gelombang
π	:	Pi Radian
w(i)	:	Koeffisien dari hukum Newton II
H_p	:	Prediksi yang di perbaharui setiap interval sample
*	:	Rata-rata 10 menit
$\bar{\zeta}$	:	Takanan rata-rata
t_i	:	Waktu aktual di ekspresikan tiap menit
$w_{jh}^{(1)}\ dan\ w_{bh}^{(1)}$	:	*adaptive weights* yang berhubungan dengan unit input dan bias unit tersembunyi
$w_h^{(2)}\ dan\ w_b^{(2)}$	:	ini berhubungan dengan unit tersembunyi dan bias pada unit output.
g(.) dan g̃ (.)	:	merepresentasikan unit tersembunyi dan fungsi aktivasi unit output.
I_{ds}	:	Arus dari Drain ke Current
V_{gs}	:	Tegangan dari Gate ke Source
V_t	:	Tegangan ambang batas transistor
β	:	Faktor penguatan transistor MOS
μ	:	Mobilitas efektif elektron permukaan dalam channel
ε	:	Permitivitas isolator pada gerbang (gate)
t_{ox}	:	Ketebalan isolator pada gerbang (gate)
W	:	Lebar dari channel
L	:	Panjang dari channel
t_p	:	Delay propagasi
i	:	Arus pengisian pengosongan pada C_L

v	:	Tegangan di kapasitor, v_1 dan v_2 adalah awal dan akhir tegangan
C_L	:	Kapasitor beban
t_{pHL}	:	Delay propagasi untuk transisi rendah ke tinggi
R_{eqp}	:	Resistansi transistor PMOS

BAB I
PENDAHULUAN

1.1 Latar Belakang

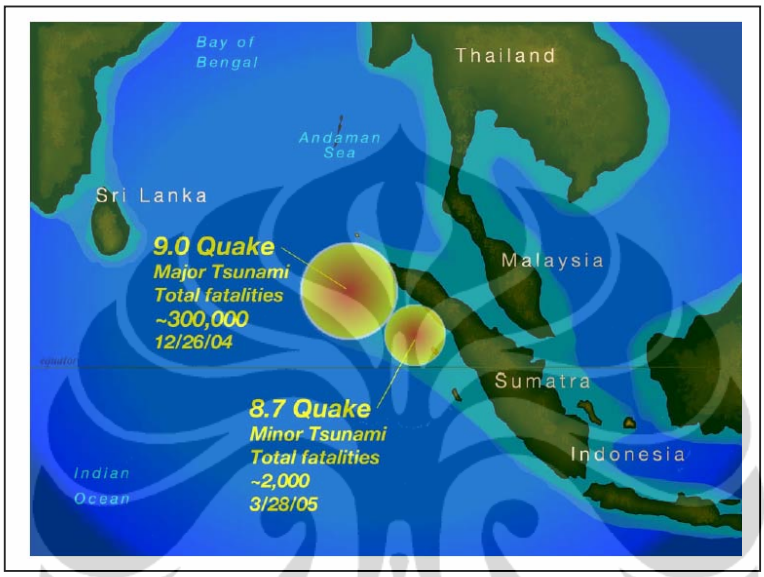
Negara Indonesia adalah negara yang berada pada kawasan yang rawan bencana, terutama bencana tsunami. *Tsunami* yang terjadi di Indonesia lebih banyak dipicu oleh gempa. Dari data geologi menunjukkan kekuatan gempa, pertemuan lempeng dan intensitas gempa di Indonesia dapat dipetakan seperti pada Gambar 1.1.



Gambar 1.1 Kekuatan gempa, pertemuan lempeng, intensitas gempa di Indonesia

Salah satu pengalaman bencana besar adalah terjadinya *tsunami* di Aceh pada 26 Desember 2004, gempa terjadi pada waktu 7:58:53 WIB. Tsunami Aceh terjadi oleh karena gerakan pergeseran retakan lempeng yang disebabkan oleh gempa dengan pusat gempa terletak pada bujur 3.316° N 95.854° E, posisi kurang lebih 160 km sebelah barat Aceh sedalam 10 kilometer, dengan kekuatan gempa 9,3 menurut skala *Richter*. Korban karena bencana tsunami ini adalah 230.000 orang tewas di 8 negara. Jumlah korban jiwa yang disebabkan oleh *tsunami* seharus nya bisa ditekan seminimal mungkin dengan bantuan teknologi, yaitu

dengan membuat system yang dapat memberikan informasi prediksi dini akan terjadinya tsunami. Pada dasarnya bencana alam tsunami merupakan siklus, dan potensi nya dapat diprediksi dengan pendekatan teknologi. *Tsunami Early Warning System* (TEWS) dibangun dan dirancang untuk tujuan mendapatkan informasi prediksi cepat pada saat akan terjadi tsunami.



Gambar 1.2 Lokasi dua gempa di samudra hindia penyebab *tsunami* Aceh

Ada persyaratan internasional dalam desain sistem deteksi dini *tsunami*[4]. Dalam persyaratan internasional ini pengembangan dan produksi sistem deteksi dini *tsunami* terdiri dari empat bagian utama yaitu *OBU, BUOY,* Satelite dan Data center. Pada saat ini pengembangan dan pembuatan system *CPU OBU tsunami early warning system* menggunakan *PC based*. Dengan mempertimbangkan efisiensi khususnya pada bagian *CPU OBU* maka pada penelitian ini akan dilakukan rancang bangun *VLSI* dengan desain *hybrid VHDL* berbasis *FPGA Xilinx Spartan* 3 untuk *CPU OBU Tsunami Early Warning System* dengan menggunakan algoritma *DART*.

Penelitian ini menggunakan metode experimental untuk meneliti sistem kerja CPU OBU dan kemudian dilakukan metode perancangan sistem deteksi dini *tsunami* pada *level lagic gate* dengan menggunakan *hybrid* desain *VHDL* pada *Xilinx Spartan* untuk implementasi system dan berikutnya adalah mengubah system yang diimplementasikan pada *Xilinx Spartan* tersebut ke dalam *system on chip* pada level *CMOS layout*.

1.2 Perumusan Masalah

Dengan meriview kembali system *OBU* yang berbasis *PC* saat ini maka penelitian ini dapat dirumuskan sebagai berikut:

Bagaimanakah membuat dan mengembangkan *CPU OBU Tsunami Early Warning System* dengan desain *hybrid VHDL* berbasis *FPGA Xilinx Spartan* 3 dan menuangkan dalam desain VLSI teknologi 0.25 um sehingga diperoleh sistem yang lebih efisien. Maksud dari efisien dari perumusan masalah ini adalah jika penggunaan sumber daya dapat dilakukan secara minimum dengan menghasilkan hasil yang optimum berarti cara ini disebut efisien. efisien dapat dievaluasi dengan membandingkan antara besarnya masukan dan besarnya keluaran yang diterima.

1.3 Batasan Masalah

Agar penelitian yang dilakukan lebih terarah, maka masalah-masalah yang dibahas dalam penelitian ini hanya difokuskan pada proses perancangan *VLS*I dengan desain hybrid *VHDL* berbasis *FPGA Xilinx Spartan* 3 untuk *CPU OBU* dengan algoritma *DART*.

1.4 Maksud dan Tujuan

Berdasarkan permasalahan penelitian yang akan diteliti, maka penelitian ini bertujuan untuk mengembangkan *CPU OBU Tsunami Early Warning System* dengan desain *hybrid VHDL* berbasis *FPGA Xilinx Spartan* 3 sehingga dapat membuktikan sistem kerja *CPU OBU tsunami early warning system* menggunakan dengan metode baru yaitu berjalan dalam *Xilinx Spartan 3* dan dituangkan dalam desain *VLSI* teknologi 0.25 um.

1.5 Metode Penulisan

Penelitian ini dilakukan dengan dua metode :

1.5.1 Mencoba memahami proses terjadinya *tsunami* dan algoritma deteksi *tsunami* yang sudah teruji dan dipakai pada semua sistem deteksi *tsunami* saat ini.

1.5.2 Menganalisa model arsitektur *CPU OBU* dan menuangkan dalam desain blok diagram, *state* diagram, *schematic* diagram, *VHDL*, *verilog* dan *CMOS layout*.

1.6 Sistematika Penulisan

Agar mudah difahami penelitian ini ditulis secara sistematis dan bertahap yaitu :

BAB I : Pendahuluan

Menjelaskan secara umum tentang *event* pemicu terjadinya *tsunami*, *sistem* deteksi tsunami untuk persyaratan internasional, produksi dan pengembanganya, testing dan fabrikasi, evaluasi tiap bagian.

BAB II : Tinjauan Teori

Menjelaskan data dan teori terkait dengan perancangan sistem *OBU tsunami early warning sistem.*pada tinjauan teori adalah bersifat referensi yang diambil dari beberapa sumber seperti jurnal internasional, nasional dan buku cetak matakuliah yang berkaitan. Referensi ini dibahas sebagai acuan pendalaman materi perancangan.

BAB III : Metode Penelitian

Adalah metode yang di gunakan dalam proses perancangan CPU Ocean Bottom Unit Tsunami Early Warning System VLSI 0.25 μm dengan desain Hybrid VHDL.

BAB IV : Perancangan CPU Ocean Bottom Unit Tsunami Early Warning System VLSI 0.25 μm dengan desain Hybrid VHDL

Mengulas proses perancangan system dengan mengacu persyaratan dasar internasional dengan metode *VHDL hybrid* design.

BAB V : Integrasi dan Demo Sistem

Membahas tentang integrasi, testing dengan pemodelan dan simulasi.

BAB VI : Kesimpulan dan Saran

Sebagai kesimpulan, saran dan hal lain yang perlu disampaikan.

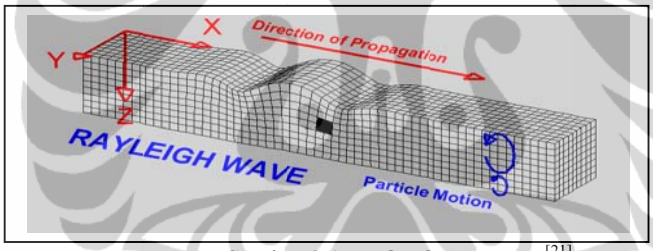
BAB II
TINJAUAN TEORI

2.1 Teori Gelombang Tsunami

Demonstrasi gelombang gempa (*Seismic Waves*)[21] digunakan untuk menggambarkan berbagai jenis perambatan gelombang melalui bahan elastis. gelombang seismik yang merambat secara mekanis dalam media dibedakan menjadi dua macam yaitu gelombang badan dan gelombang permukaan. Gelombang Rayleigh merupakan jenis gelombang permukaan.

Ciri *Rayleigh wave* adalah :

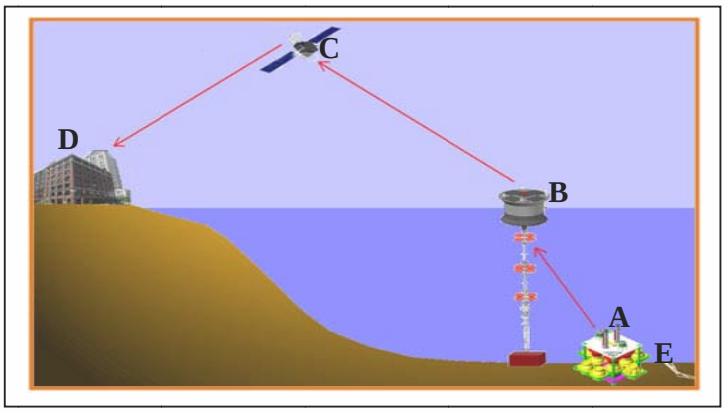
- ✓ gerakan eliptik retrograde/ "ground roll" (tanah memutar ke belakang tapi secara umum gelombangnya merambat ke depan—analog dengan gelombang laut)
- ✓ Sedikit lebih cepat dari Love Wave (90% dari kecepatan S-wave)



Gambar 2.1 *Rayleigh (R) wave* [21]

Gelombang Rayleigh identik dengan gelombang pada saat terjadi gelombang tsunami, sehingga gelombang ini digunakan sebagai acuan dalam membuat algoritma deteksi tsunami [5].

2.2 *Existing Tsunami Early Warning System*



Gambar 2.2 Desain *Tsunami Early Warning System*

TEWS (*Tsunami Early Warning System*) adalah sebuah sistem yang dirancang untuk mendeteksi *tsunami* memberikan informasi lebih awal pada penduduk di sekitar bencana *tsunami* agar segera melakukan evakuasi sehingga mencegah jatuhnya korban. Deteksi *tsunami* didasarkan pada perubahan tekanan air laut dalam dengan menggunakan algoritma [5] sehingga didapatkan informasi pesan bahwa terjadi peristiwa *tsunami*. Data tekanan air laut dalam di peroleh dari sensor *BPR* (*Bottom Pressure Recorder*) yang terpasang di *OBU*. *CPU OBU* mengolah data dari BPR dengan algoritma DART untuk mendeteksi tsunami terus menerus, sehingga pada saaat tsunami terdeteksi maka pesan segera dikirim ke surface Buoy dengan perantara modem acoustic. Kemudian *unit surface buoy* akan mengkonstruksi format pesan tsunami dan kemudian dikirim ke *RDS* melalui komunikasi satelit. Setelah itu dari *RDS* informasi tsunami disebarluaskan ke pada pihak yang berkepentingan.

Sistem ini umumnya terdiri dari dua bagian penting yaitu jaringan sensor untuk mendeteksi tsunami serta infrastruktur jaringan komunikasi. Komponen utama TEWS (seperti pada Gambar 2.2) adalah :

A. *OBU (Ocean Bottom Unit)* : bagian yang berada di dasar air laut dimana BPR sebagai tsunamometer berada untuk mengukur sea level.

B. *BUOY* : bagian yang berada di permukaan air laut di mana *Embedded* sistem sebagai pengolah data berada untuk mendapatkan informasi dini tsunami.

C. *SATELIT* : sebagai media komunikasi cepat dari BUOY yang berada di tengah laut dan kirim ke RDS (*Read Down Station*).

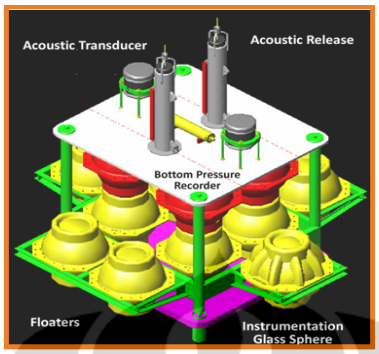
D. *RDS* : Data center , yaitu tempat dimana untuk membaca seluruh pesan informasi tsunami atau juga biasa mengirim sinyal control.

E. RETAKAN : Lokasi pertemuan lempeng/patahan yang berada di bawah laut yang berpotensi memicu terjadinya gempa dan *tsunami*.

2.2.1 Jaringan Sensor OBU (*Ocean Bottom Unit*)

OBU (*Ocen Bottom Unit*) merupakan rangka baja yang digunakan untuk menyusun komponen-komponen sensor. Komponen-komponen yang tersusun pada OBU terdiri dari BPR (*Bottom Pressure Recorder*), *Instrumentation Glass*

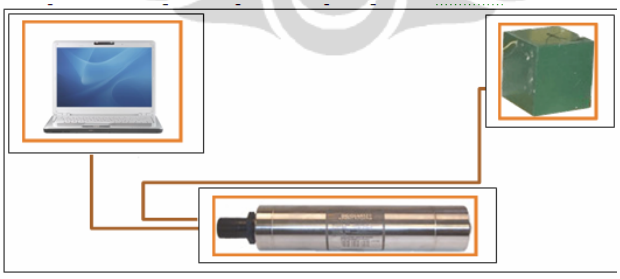
Sphere, *Acoustic transducer*, *Acoustic releaser*, *floater* (pelampung), dan pemberat. Bentuk OBU adalah seperti terlihat pada gambar berikut :



Gambar 2.3 *Ocean Bottom Unit (OBU)* [24]

2.2.1.1 BPR (*Bottom Pressure Recorder*)

BPR menggunakan kristal *piezo* (Paroscientific Model Digiquartz 410K) sebagai sensor tekanan, yang memiliki resolusi setara dengan 1 mm dari perubahan permukaan laut[19]. Tekanan diukur dan diambil tiap rentang waktu 15 detik. Sensor tekanan statik air laut merepresentasikan ketinggian air laut hingga kepermukaan tanpa menghiraukan percikan-percikan gelombang yang disebabkan oleh gerakan kapal, angin dan lain sebagainya. Pada sistem TEWS digunakan sensor 8CB7000-I dari Paroscientific yang mampu bertahan hingga kedalaman 7000 m. Sensor ini dilengkapi dengan prosesor sendiri yang mengolah data mentah menjadi informasi yang dapat diakses dengan menggunakan serial port. Untuk keperluan testing BPR dapat dilihat pada gambar berikut :



Gambar 2.4 Testing Bottom Pressure Recorder (BPR) [19].

2.2.1.2 CPU OBU

OBU (Gambar 2.3) terdapat Vitrovex 17" yaitu bola kaca dan penutup plastik yang sangat keras, di bagian ini berisi semua instrumen komponen. Seluruh Instrumen ini terlindungi dengan bola kaca Glassphere dengan tekanan maksimum

6000 dbar. Batterai kering dan CPU dimasukkan kedalam sebuah Instrument Housing yang terbuat dari Glass dengan spesifikasi yang khusus, yaitu kedap dan tahan terhadap tekanan sampai dengan 600 bar. Glass Instrument Housing ini dilengkapi dengan connector khusus untuk mengeluarkan power untuk Bottom Pressure Recorder (BPR) dan connector khusus untuk komunikasi antara CPU dan Akustik Modem SR-100

Gambar 2.5 Glass Instrument Housing

System *OBU* bekerja barbasis pada embedded *PC*. *CPU* yang dipakai pada *Ocean Bottom Unit (OBU)* untuk *Tsunami Early Warning System* adalah menggunakan *Single Board Computer ZeusArcom*. Pada saat pembelian *CPU Card ViperArcom* yg bukan *DevelopmentKit version*, *CPUCard* belum dilengkapi dengan sistem operasi. *CPUCard* hanya dilengkapi dengan *bootloader Redboot* sehingga perlu melakukan instalasi sistem operasi AEL dalam *flash disk Zeus*.

Gambar 2.6 *Control Processing Unit (CPU) pada OBU*

2.2.2 *Deep-ocean Assessment and Reporting of Tsunamis (DART) Algorithm*

Deteksi otomatis dari kejadian *tsunami* dilakukan seperti yang digunakan pada *Deep-ocean Assessment and Reporting of Tsunamis* (DART). DART adalah sebuah sistem monitoring untuk mendeteksi tsunami, yang dikembangkan oleh Pacific Marine Environmental Laboratory (PMEL). Sistem ini menggunakan algoritma yang dikembangkan oleh Mofjeld (1997)[5] dan telah dipatenkan dalam [10; US Patent 11]., sehingga algoritma ini selanjutnya disebut sebagai algoritma DART. Algoritma DART menggunakan perubahan tekanan di dasar laut untuk memprediksi terjadinya tsunami. Konsep dasar deteksi *tsunami* ini adalah dengan cara membandingkan tekanan aktual dasar laut yang diukur setiap 15 detik dengan

tekanan prediksi dasar laut berdasarkan ekstrapolasi polinomial kubik (suku tiga). Tekanan prediksi bawah laut dapat dinotasikan sebagai notasi sigma berikut

$$H_p(t') = \sum_{i=0}^{3} w(i)H^*(t-idt) \qquad (2.1)$$

Dimana :

t' : waktu yang sebenarnya dinyatakan dalam menit.

w(i) : koefisien yang diperoleh dari ekstrapolasi maju dengan cara Newton

H_p : tekanan prediksi yang diperbarui setiap interval sampel (setiap 15 s).

***** : tanda bintang yang menunjukkan rata-rata 10 menit dan dt = 1 jam

Rumus (2.1) dapat direpresentasikan seperti pada Gambar 2.7. polinomial kubik ini diambil dari 4 buah nilai rata-rata data 10 menit-an yang meliputi data 10 menit paling baru dan 3 buah data 10 menit sebelumnya. Polinomial ini diperbarui untuk setiap pengukuran 15 detik. Prediksi waktu **t'** diatur pada 5,25 menit, yang merupakan setengah dari waktu 10 menit Interval ditambah dengan 15 detik (0,25 menit) interval sampling untuk pengukuran. Koefisien w yang diperoleh adalah :

w(0) = 1.16818457031250

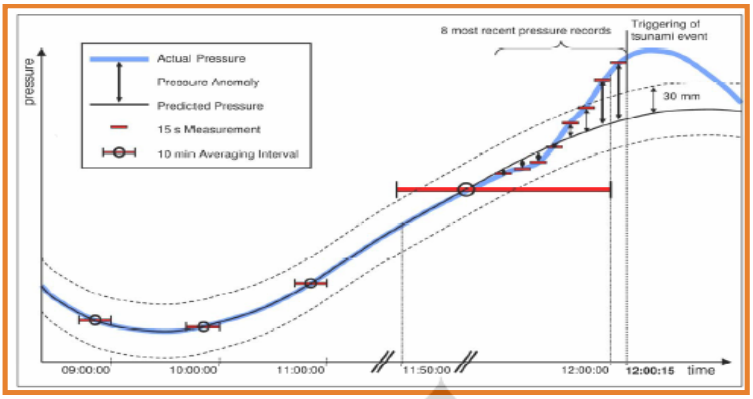
w(1) = -0.28197558593750

w(2) = 0.14689746093750

w(3) = -0.03310644531250

Amplitudo dalam algoritma DART dihitung dengan mengurangkan tekanan prediksi dasar laut dari tekanan aktual dasar laut untuk mendapatkan sampel sinyal, dimana tekanan prediksi cocok dengan pasang surut dan fluktuasi frekuensi yang lebih rendah. tekanan prediksi diperbarui setiap 15 detik, yang merupakan periode sampling dari DART.

Berdasarkan pengamatan terakhir yang dilakukan, ambang batas yang wajar untuk Pasifik Utara adalah 3 cm (atau 30 mm). *Tsunami* terdeteksi jika perbedaan antara tekanan aktual dan tekanan predikasi melebihi ambang batas yang ditentukan besarnya.



Gambar 2.7 Sketsa yang menggambarkan algoritma DART[2]

Di laut dalam, gelombang permukaan pendek (gelombang angin) tidak memiliki pengaruh pada tekanan dasar laut, hanya gelombang gravitasi yang panjang seperti pasang surut dan tsunami, dengan gelombang-panjang dari ratusan kilometer ini yang mempengaruhi tekanan dasar. Dalam kasus pasang surut normal, prediksi tekanan (H_p) sangat cocok dengan tekanan aktual karena perubahan dari selisih keduanya pada rentang waktu beberapa jam. Sebaliknya, gelombang tsunami memiliki rentang waktu hanya beberapa menit, sehingga menghasilkan anomali yang lebih besar antara tekanan prediksi dan tekanan aktual dasar laut.

Sebuah kejadian *tsunami* dipicu ketika dua sampling 15 detik terbaru dalam pengamatan melebihi ambang batas tekanan *anomaly* 30 mm (Gambar 2.7). Dalam hal ini, kriteria ambang batas lonjakan yang dilewati (tidak ditunjukkan) tujuan ambang batas adalah untuk proses validasi menghindari kriteria palsu. Untuk ini, pembacaan 15 detik untuk tekanan kedua dari terakhir tidak harus lebih dari 100 mm karena setelah itu seharusnya perubahan tekanan kembali seperti semula dan waktu yang dibutuhkan juga kembali seperti semula yaitu 45 detik dari terakhir.

Segera setelah deteksi dari suatu peristiwa *tsunami*, maka pesan *tsunami* yang dihasilkan ditransmisikan ke surface buoy. Pesan ini berisi 8 pembacaan tekanan aktual (terbaru) yang terdiri dari pembacaan 2 menit terakhir, dan anomali tekanan yang sesuai (tekanan aktual dikurangi dengan tekanan diprediksi), ada dua pengukuran terakhir yang lebih besar dari ambang 30 mm. kemudian isi pesan selanjutnya adalah timestamp, ID pesan alarm yang dimulai dengan "1", dan polinomial terganggu terakhir yang memicu peristiwa tsunami semua

ditransmisikan dengan mengutamakan pesan tsunami. Format pesan ini menjamin bahwa waktu yang tepat mulai terjadinya peristiwa *tsunami*, dan bentuk yang tepat dari anomali gelombang yang diperoleh

Yang terpenting adalah dapat memberikan informasi pesan bahaya potensi tsunami dengan akurasi tertinggi. Bahkan pada saat peristiwa pesan pertama tsunami hilang (misalnya akibat kesalahan komunikasi satelit), awal waktu mulai terjadi peristiwa dan bentuk gelombang yang sedang berlangsung dapat direkonstruksi menggunakan ID pesan, tekanan aktual dan polinomial terganggu tersimpan dalam *buffer* dan kemudian pesan ditransmisikan kembali dalam interval 2-menit.

2.3 Pengembangan desain teknologi CMOS

2.3.1 Persamaan Desain CMOS

Seperti yang dinyatakan pada *principles of CMOS VLSI Design* [12], transistor MOS memiliki tiga daerah operasi :

1. Daerah cut-off
2. Daerah linear
3. Daerah saturasi

Berikut persamaan ideal (first order) [Cobb70] [Sah64] yang menggambarkan perilaku perangkat nMOS di ketiga daerah opersi tersebut :

$$I_{ds} = \begin{cases} 0; & \text{➢ Daerah cut-off} \quad V_{gs} - V_t \le 0 \quad \text{(a)} \\ \beta\left[(V_{gs} - V_t)V_{ds} - \frac{V^2_{ds}}{2}\right]; & \text{➢ Daerah linear} \quad 0 < V_{ds} < V_{gs} - V_t \quad \text{(b)} \\ \frac{\beta}{2}(V_{gs} - V_t)^2; & \text{➢ Daerah saturasi} \quad 0 < V_{gs} - V_t < V_{ds} \quad \text{(c)} \end{cases} \tag{2.2}$$

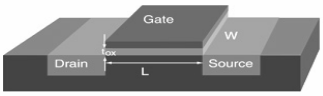
Dimana :

I_{ds} = Arus dari Drain ke Source

V_{gs} = Tegangan dari Gate ke Source

V_t = Tegangan ambang batas transistor

β = Faktor penguatan transistor MOS



Gambar 2.8 Transistor CMOS Layout 3 D

β tergantung pada parameter-parameter proses dan geometri perangkat , β dinyatakan dengan :

$$\beta = \frac{\mu\varepsilon}{t_{ox}}\left(\frac{W}{L}\right) \qquad (2.3)$$

Dimana :

μ = Mobilitas efektif elektron permukaan dalam channel

ε = Permitivitas isolator pada gerbang (gate)

t_{ox} = Ketebalan isolator pada gerbang (gate)

W = Lebar dari channel

L = Panjang dari channel

Faktor penguatan β tergantung pada factor proses $\frac{\mu\varepsilon}{t_{ox}}$, yaitu semua persyaratan proses yang memperhitungkan faktor-faktor seperti kepadatan doping dan ketebalan gerbang oksida, dan penguatan β tergantung pada geometry $\left(\frac{W}{L}\right)$, yaitu tergantung dengan layout CMOS sebenarnya. Seperti terlihat pada gambar :

Nilai-nilai tetapan yang dipakai adalah:

μ_n = 500 cm ^{2} / V-sec

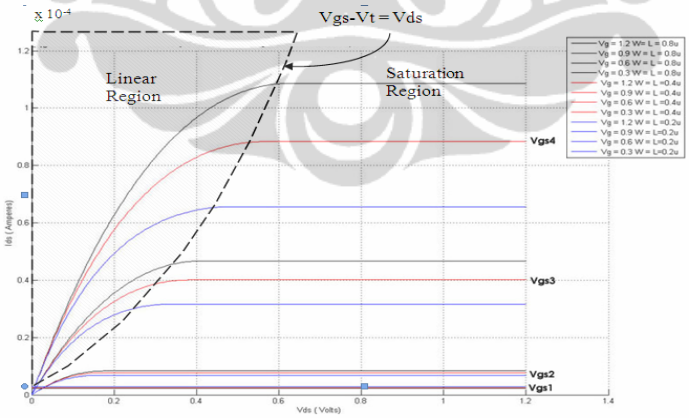
ε = $4\,\varepsilon_o$ = 4 x 8.85 x 10^{-14} F/cm

t_{ox} = 500 Å

Dengan memasukan parameter tetapan maka didapatkan β pada n-device sebagai berikut :

$$\beta = \frac{\mu\varepsilon}{t_{ox}}\left(\frac{W}{L}\right) = \frac{500 \times 4 \times 8.85 \times 10^{-14}}{5 \times 10^{-14}}\frac{W}{L} = 35\frac{W}{L}\ \mu A/V^2 \qquad (2.4)$$

Kondisi cut-off didiskripsikan seperti pada persamaan 2.2a juga disebut sebagai kondisi subthreshold, dimana Ids naik secara eksponensial terhadap V_{ds} dan V_{gs}.



Gambar 2.9 Grafik Ids terhadap Vds MOS dengan model tegangan saturasi

Kondisi cut-off merupakan batas antara daerah linier dan saturasi sesuai persamaan $V_{ds} = V_{gs}$-Vt (dimana $V_{ds} \approx 0$). Meskipun nilai Ids sangat kecil ($I_{ds} \approx 0$) , nilai batas I_{ds} dapat mempengaruhi kinerja sirkuit penyimpanan dinamis seperti sel memori.

Kondisi linear didiskripsikan seperti pada persamaan 2.2b, resistansi keluaran pada kondisi liner dapat diperoleh dengan penurunan persamaan 2.2b dengan memperhatikan Vds, yang mengakibatkan konduktansi output seperti berikut :

$$\lim_{V_{ds}\to 0} \frac{dI}{dV_{ds}} \approx \beta \left(V_{gs} - V_t\right) \quad (2.5)$$

Dengan mengatur ulang persamaan resisteance R_c dapat didekati dengan persamaan :

$$R_c(linear) = \frac{1}{\beta(V_{gs}-V_t)} \quad (2.6)$$

Persamaan ini menunjukan bahwa yang mengatur resistansi output saat kondisi linear adalah tegangan V_{gs}.

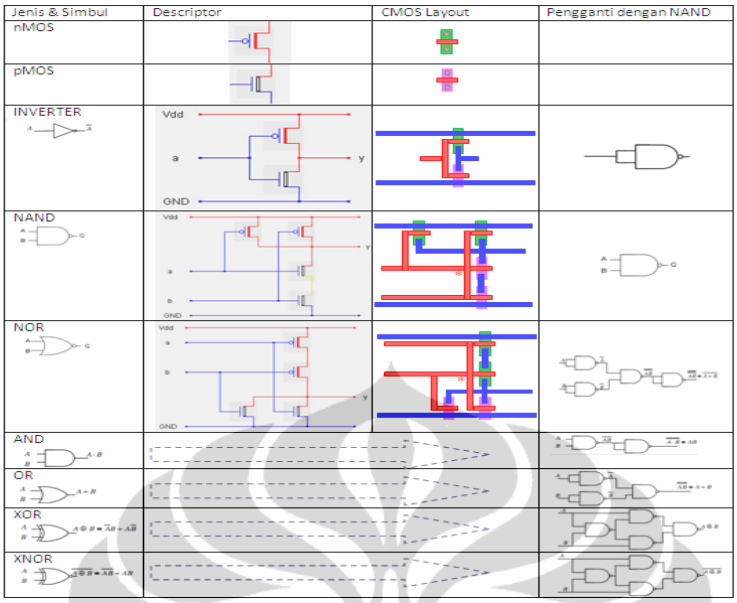
Kondisi saturasi didiskripsikan seperti pada persamaan 2.2c, pendekatan Ids pada persamaan ini diasumsikan bahwa arus pada channel atau saluran terjadi saturasi (konstan) dan tidak tergantung dari V_{ds} yang diterapkan. Pada prakteknya, I_{ds} yang menyebabkan saturasi sedikit meningkat dengan meningkatnya V_{ds}

2.3.2 CMOS Inverter

2.3.2.1 Inverter sebagai inti dari semua desain digital

Inverter adalah merupakan inti dari semua desain digital. Analisis inverter dapat diperdalam untuk menjelaskan perilaku gerbang yang lebih kompleks seperti NAND, NOR, atau XOR, yang pada gilirannya membentuk blok bangunan modul seperti pengganda dan prosesor.

Inverter dibentuk dari dua CMOS transistor yaitu PMOS dan NMOS. Secara teknik penggunaan inverter sebagai komponen switch (saklar on-off) dalam system digital memiliki kelebihan lebih robust di banding menggunakan karakteristik switch satu transistor. Parameter digital menunjukkan lebih kuat dengan inverter. Jadi Seluruh rancang bangun rangkain digital dalam level CMOS layout di arahkan selalu menggunakan inverter. berikut table desain CMOS



Tabel 2.1 Simbul, descriptor, CMOS layout dan NAND rangkaian pengganti (*tidak terhubung)

2.3.2.2 Analisa delay propagasi

Salah satu cara untuk menghitung delay propagasi inverter adalah dengan mengintegrasikan muatan kapasitor dan debit saat ini. Berikut adalah ekspresi persamaannya :

$$t_p = \int_{v1}^{v2} \frac{C_L(v)}{i(v)}\, dv \qquad (2.7)$$

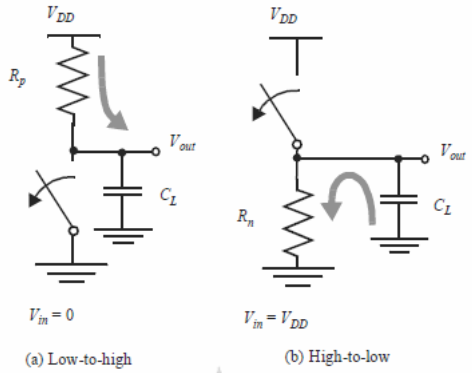
t_p = Delay propagasi

i = Arus pengisian pengosongan pada C_L

v = Tegangan di kapasitor, v_1 dan v_2 adalah awal dan akhir tegangan

C_L = Kapasitor beban

Pada kenyataanya perhitungan pada persamaan ini tidak biasa dilacak, karena baik C_L dan $i(v)$ adalah fungsi nonlinier. Mari kita lihat model switch-disederhanakan dari inverter seperti pada Gambar 2.10 untuk memperoleh pendekatan yang wajar dari delay propagasi yang memadai untuk analisis manual



Gambar 2.10 Perilaku model switch dinamis dari inverter CMOS statis.

Tegangan keluaran tergantung pada resistansi dan kapasitor beban, dimana arus kapasitor beban di arahkan dengan mengganti baik oleh elemen linier konstan dengan nilai rata-rata selama interval waktu. Ekspresi rata-rata resistansi dari transistor MOS dapat diturunkan sebagai berikut :

$$t_{pHL} = \ln(2)\, R_{eqn} C_L = 0.69\, R_{eqn} C_L \qquad (2.8)$$

Jadi kita bisa mendapatkan delay propagasi untuk transisi rendah ke tinggi, seperti berikut :

$t_{pHL} = 0.69\, R_{eqn} C_L$

R_{eqp} adalah setara resistensi dari transistor PMOS dengan interval waktu tertentu. Analisis ini mengasumsikan bahwa beban kapasitansi setara dan identik untuk kedua transisi *low to high* dan *high to low*. Delay propagasi keseluruhan inverter didefinisikan sebagai rata-rata dari dua nilai, seperti berikut:

$$t_p = \frac{t_{pHL} + t_{pLH}}{2} = 0.69\, C_L \left(\frac{R_{eqn} + R_{eqp}}{2}\right) \qquad (2.9)$$

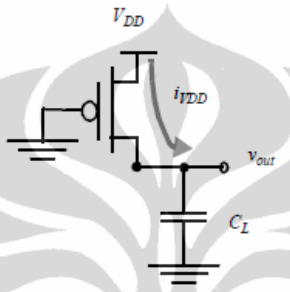
Persamaan ini yang sangat sering digunakan untuk mengidentifikasi propagasi delay pada saat perubahan masukan dari *low to high* dan *high to low*. Persamaan ini berlaku dengan membuat resistansi pada NMOS dan PMOS kira-kira sama.

2.3.2.3 Analisa power konsumsi

Setiap kali kapasitor CL mendapat pembebanan dari transistor PMOS, tegangan naik dari 0 sampai Vdd, dan sejumlah energi yang ditarik dari catu daya. Sebagian dari energi ini didisipasikan dalam perangkat PMOS, sementara sisanya

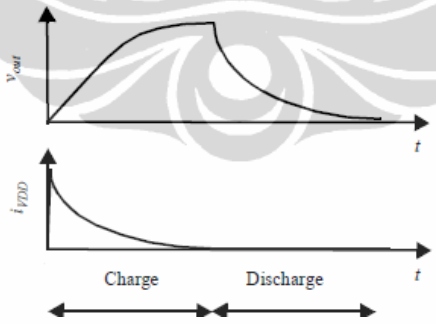
disimpan pada kapasitor beban. Selama transisi tinggi ke rendah, energi pada kapasitor ini dibuang, dan energi didisipasikan dalam transistor NMOS.

Konsumsi energi pada inverter dapat di turunkan. Pertama kita mempertimbangkan transisi *low to high*. Asumsi awalnya, bahwa setiap gelombang input pada perangkat NMOS dan PMOS tidak pernah merespon sinyal secara sama, atau berbeda antara NMOS dan PMOS. Oleh karena itu, berlaku rangkaian ekuivalen seperti gambar berikut



Gambar 2.11 Rangkaian ekivalen selama transisi rendah ke tinggi (*low to high*).

Nilai energi E $\boldsymbol{v}_{\mathrm{DD}}$, diambil dari pasokan selama transisi, seperti pada energi E C, yang tersimpan pada kapasitor diakhir transisi, jadi dapat diturunkan dengan mengintegrasikan daya sesaat selama periode transisi. Gelombang Vout (t) dan iV_{DD} (t) digambarkan gambar berikut



Gambar 2.12 Tegangan keluaran dan supply arus selama (dis) charge pada C_L

$$E_{V_{DD}} = \int_0^\infty i_{V_{DD}}(t) V_{DD} dt = V_{DD} \int_0^\infty C_L \frac{dv_{out}}{dt} dt = C_L V_{DD} \int_0^{V_{DD}} dv_{out} = C_L V_{DD}^2 \quad (2.10)$$

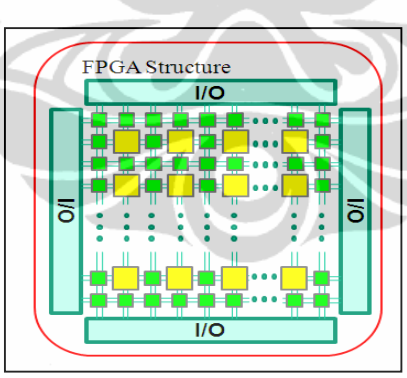
$$E_C = \int_0^\infty i_{V_{DD}}(t) v_{out} dt = \int_0^\infty C_L \frac{dv_{out}}{dt} v_{out} dt = C_L \int_0^{V_{DD}} v_{out} dv_{out} = \frac{C_L V_{DD}^2}{2} \quad (2.11)$$

Persamaan di atas berlaku pada saat transisi dari *low to high*. Dengan mengamati selama transisi rendah ke tinggi, C_L di beri muatan melalui $C_L V_{DD}$. Dengan kata lain bahawa $C_L V_{DD}^2$ $(= Q \; x \; V_{DD})$. Energi yang tersimpan pada kapasitor sama dengan $\frac{C_L V_{DD}^2}{2}$. Ini berarti bahwa hanya setengah dari energi yang disediakan oleh sumber daya disimpan di C_L. Separuh lainnya telah disebarkan transistor PMOS. Kemudian pada saat fase berikutnya yaitu perubahan hightо-low, energy pada kapasitor akan keluar, dan energi didisipasikan dalam perangkat NMOS. Dan hal ini, tidak ada ketergantungan pada ukuran perangkat. Kesimpulannya bahwa setiap siklus switching (perubahan dari H ke L dan dari L ke H) jumlah energi tetap sama yaitu $C_L V_{DD}^2$

2.4 *Hybrid* Design

2.4.1 FPGA (*Field Programmable Gate Array*)

Field Programmable Gate Array adalah IC digital yang digunakan untuk mengimplementasikan rangkaian digital. FPGA merupakan sebuah IC digital yang bersifat *Programmable*, yang artinya user dapat memakai IC digital secara berulang-ulang untuk menyesuaikan program apa yang akan ingin di *download* kedalam FPGA. Struktur FPGA dapat dilihat seperti pada gambaer berikut :



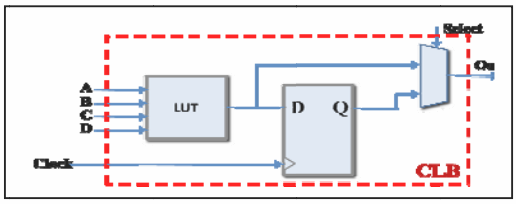
Gambar 2.13 Struktur FPGA

Seperti terlihat pada gambar di atas bahwa FPGA terdiri dari beberapa bagian yaitu :

1. *Configure Logic Blocks* (CLB), bagian ini yang akan memproses segala bentuk rangkaian logika yang dibuat oleh user/pemakai.

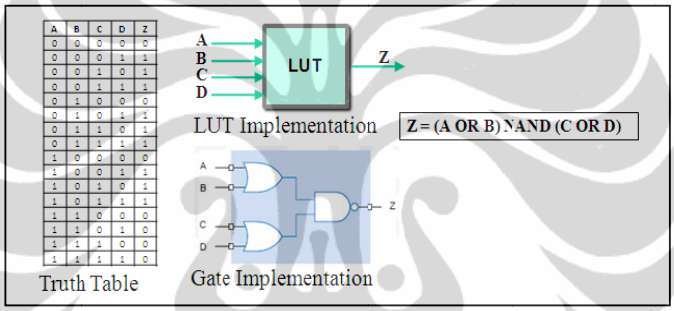
2. I/O, sebagai *interface* antara *external pin* dari *device* dan *internal user logic*

3. *Programmable Interconnect*, bagian ini menghubungkan antara CLB satu dengan CLB lainnya.



Gambar 2.14 Isi setiap CLB

Arsitektur *CLB* berisi *LUT, DFF,* dan *Multipexer*. *LUT (Loock Up Table)* digunakan untuk implementasi rangkaian digital kombinasional. LUT dengan n Input dapat digunakan untuk implementasi beberapa fungsi rangkaian digital kombinasional dengan n input. *LUT* di program dengan truth-table seperti terlihat pada gambar berikut:



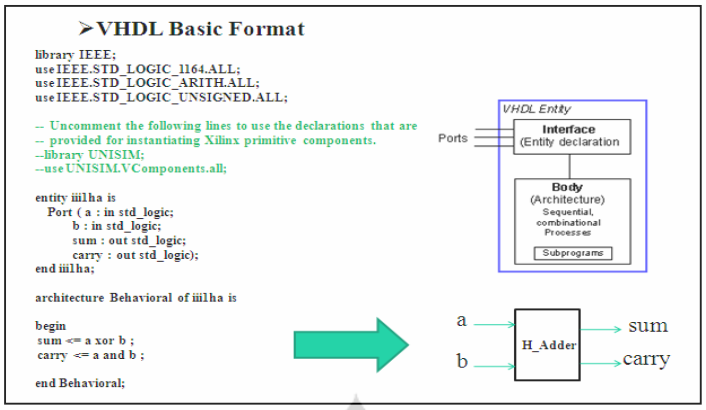
Gambar 2.15 Implementasi rangkain digital dengan truth table

FPGA berkembang pada tahun 1980-an dan baru dikembangkan pada tahun 1984 oleh perusahaan Xilinx yang berbasis di San Jose CA. Terdapat 5 perusahaan besar yang memproduksikan FPGA diantaranya Xilinx, Altera, Lattice, Actel, Quicklogic. Hanya 2 perusahaan yang memiliki nama tingkat tinggi yang memproduksikan FPGA yaitu Xilinx dan Altera, perusahaan Xilinx terkenal dengan software miliknya yang bernama ISE WebPack, dan perusahaan Altera terkenal dengan software bernama Quartus II Web Edition.

2.4.2 *Hybrid* Model

2.4.2.1 *Coding Style*

Coding style adalah gaya desain rangkaian digital dengan cara memprogram melalui bahasa pemprograman VHDL (*VHSIC Hardware Description Language*), contoh format dasar pemprograman bahasa VHDL seperti berikut :



Gambar 2.16 Format dasar pemprograman VHDL

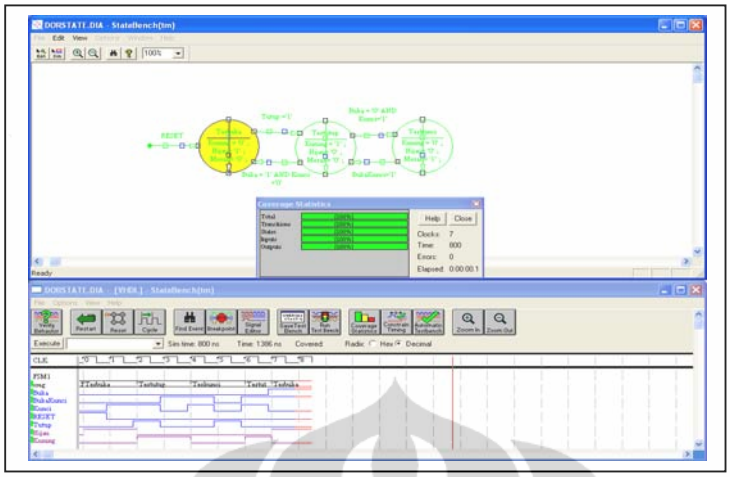
Pada Gambar 2.16 di sebelah kiri terlihat desain *half adder* dengan menggunakan bahasa VHDL, sedangkan di sebelah kanan adalah simbol arsitektur dan *entity dari half adder*.

2.4.2.2 *State diagram style*

State diagram adalah gaya desain rangkaian logika dengan cara membuat *state diagram*, yaitu menyusun beberapa kondisi keadaan dimana perubahan kondisi keadaan satu ke kondisi keadaan lain di pengaruhi oleh *event* tertentu, contoh nya adalah kita biasa membuat rangkaian logika dari mekanisme buka tutup pintu, mekanisme nya sederhana yaitu:

1. *State* A = Kondisi pintu terbuka (event yang berpengaruh adalah pintu di tutup, yang lain diabaikan). Jika pintu ditutup maka *State* akan berubah menjadi *State* B(pintu tertutup).
2. *State* B = Kondisi pintu tertutup (*event* yang berpengaruh ada 2 yaitu pintu dibuka atau pintu dikunci, selain itu diabaikan). Jika pintu dibuka maka *state* akan berubah menjadi *state* A(pintu terbuka), sedangkan jika pintu dikunci maka state akan berubah menjadi *state* C(pintu terkunci).
3. *State* C = Kondisi pintu terkunci (*event* yang berpengaruh adalah pintu dibuka kuncinya, selain itu diabaikan).

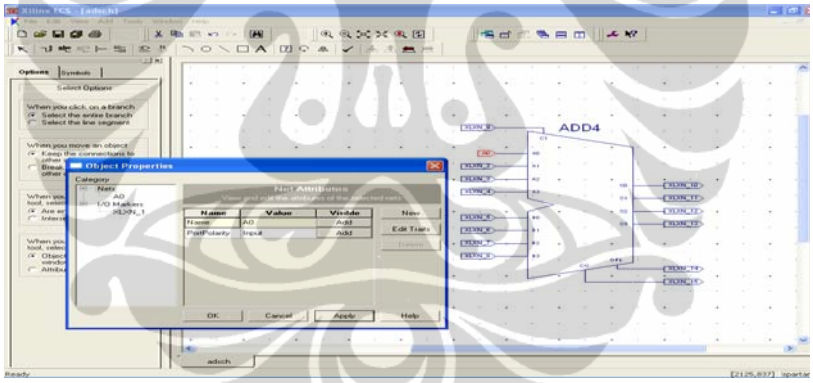
Jadi mekanisme pintu mempunya 3 *state* yaitu, pintu terbuka, pintu tertutup dan pintu terkunci. Berikut cantoh disain *state* diagram dengan program ISE :



Gambar 2.17 Desain *state* diagram dengan ISE 6.3i

2.4.2.3 *Schematic diagram style*

Schematic style adalah gaya desain rangkaian logika dengan cara membuat *schematic* diagram, yaitu menyusun rangkaian digital dengan simbul rangkaian. Berikut adalah contoh membuat rangkain dengan *schematic* berupa adder 4 bit:

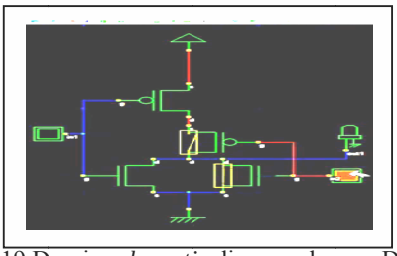


Gambar 2.18 Desain *schematic* diagram dengan ISE

2.4.3 Metode desain CMOS layout dengan RTL

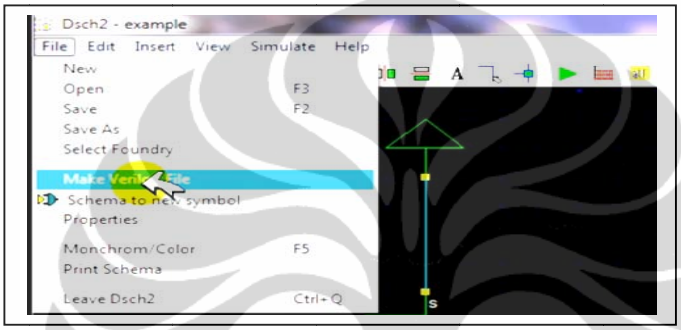
2.4.3.1 Membuat file verilog dari *schematic*

File verilog adalah berupa kode seperti *netlist* yang menentukan sirkuitri rangkaian dan penggunaan komponennya, kode verilog dapat diperoleh dari *schematic* rangkaian yang dibuat dengan menggunakan DSCH2. Dengan menggunakan DSCH2 ini schematic rangkaian dapat kita buat dan kita simulasikan, contoh desain *schematic* dan membuat file verilognya adalah seperti berikut :



Gambar 2.19 Desain *schematic* diagram dengan DSCH2

Dari Gambar 2.19 ditunjukan sebuah *schematic* rangkain logika yang disimulasi dengan program DSCH2. Jika rangkaian *schematic* yang kita buat sudah benar dan hasil simulasi sesuai dengan harapan maka, rangkaian *schematic* ini dapat kita buat file verilognya sebagai berikut :

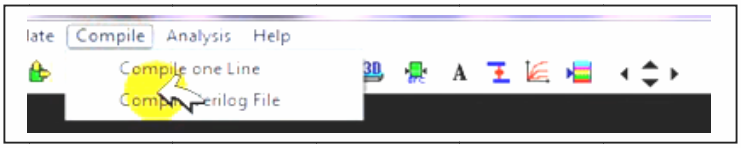


Gambar 2.20a Membuat verilog dari schematic diagram dengan DSCH2

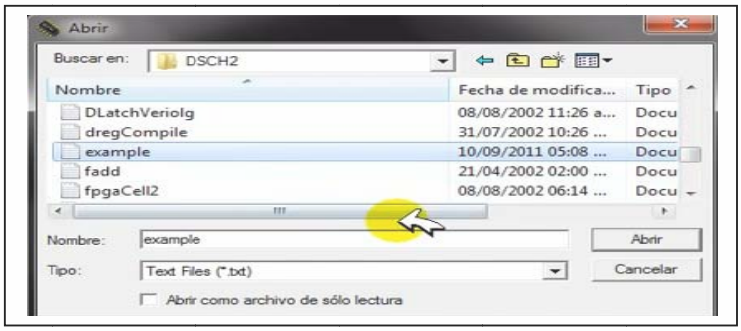
Gambar 2.20b Hasil kode verilog yang diperoleh

2.4.3.2 CMOS *Layout*

Cara mengubah file verilog menjadi CMOS *layout* adalah dengan menggunakan program microwind. Program microwind adalah program yang digunakan untuk mendesain CMOS *layout* secara langsung atau dengan mengkompile file verilog dari *schematic* rangkaian. Contoh kompilasi file verilog menjadi CMOS layout sebagai berikut :



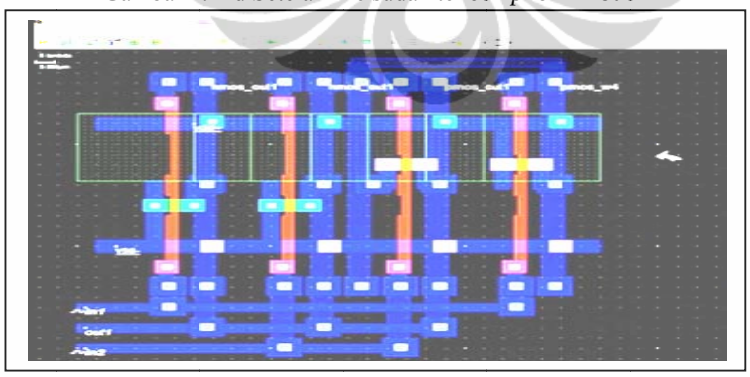
Gambar 2.21a Pada program microwind pilih menu *compile* verilog file



Gambar 2.21b Setelah muncul file folder pilih file yang mau di *compile*

Gambar 2.21c Setelah itu akan muncul kode verilog, klik *compile*

Gambar 2.21d Setelah file sudah ter *compile* kilk *back*



Gambar 2.21e Hasil akhir file verilog yang sudah menjadi CMOS *layout*

BAB III
METODE PENELITIAN

Dalam metode penelitian ini dilakukan perancangan dan pengujian terhadap prinsip kerja *CPU OBU Tsunami Early Warning System* berbasis pada Xilinx Spartan 3 dan sekaligus mengubah system tersebut menjadi CMOS layout teknologi *VLSI* 0.25 μm. Pada uji coba yang akan dilakukan, dibuat program simulasi pengukuran kolong laut yaitu Bottom Pressure Recorder (BPR) yang memberikan informasi ketinggian air laut. Tahapan penelitian secara garis besar akan meliputi langkah-langkah
sebagai berikut:

1. Mencari dan mengumpulkan referensi
2. Menentukan flowchart dan blok diagram sistem
3. Membuat kode VHDL sesuai flowchart dan blok diagram
4. Mengubah kode VHDL menjadi RTL
5. Mengubah RTL menjadi schematic dan kode verilog per blok
6. Mengubah verilog menjadi CMOS layout teknologi VLSI 0.25 μm
7. Menanam sistem ke dalam Xilinx Spartan 3

8 Demo sistem yang tertanam pada Xilinx Spartan 3 dengam menggunakan program simulasi

3.1 Mencari dan mengumpulkan referensi

Proses mengumpulkan referensi diharapkan mendapatkan informasi penting yang berkaitan dengan gambaran terhadap prinsip kerja *CPU OBU TEWS* dan juga informasi penting yang berkaitan dengan metode proses perancangan *VLSI* 0.25 μm dengan *hybrid VHDL* sehingga dapat menghasilkan dua output yang diharapkan yaitu desain *CPU OBU TEWS* dalam bentuk *CMOS layout* teknologi *VLSI 0,25 μm* dan system *CPU OBU TEWS* yang ditanam di Xilinx Spartan 3 sehingga dapat dilakukan pengujian terhadap prinsip kerja dari *CPU OBU TEWS*

3.2 Menentukan *flowchart* dan blok diagram sistem

Penentuan *flowchart* dan blok diagram sistem adalah hal yang paling utama dalam proses perancangan. *Flowchart* dan blok diagram digunakan sebagai

acuan atau target dari hasil perancangan sistem. *Flowchart* dan blok diagram dapat membantu untuk memberikan gambaran alur kerja dari sistem. *Flowchart* dan blok diagram adalah gambaran dariprinsip kerja sistem yang akan di rancang.

3.3 Membuat kode *VHDL* sesuai *flowchart* dan blok diagram

Setelah *flowchart* dan blok diagram sudah dibuat dengan benar maka langkah selanjutnya adalah melakukan *coding,* yaitu membuat kode *VHDL* sesuai dengan *flowchart* dan blok diagram dari sistem. Alur dan prinsip kerja *flowchart* dan blok diagram dari sistem akan dirubah ke bahasa mesin yaitu *VHSIC (Hardware Description Language)*. Pemilihan menggunakan desain *hybrid VHDL* karena memberikan banyak keuntungan dalam proses desain yang paling utama yaitu *VHDL* dapat ditanam pada *Xilinx Spartan* 3 dengan proses impact, selain itu *VHDL* dapat dirubah ke desain *RTL* sehingga diperoleh schematic dari kode *VHDL* tersebut. Dengan menggunakan *RTL* dapat diproses lebih lanjut sehingga diperoleh desain *CMOS layout*.

3.4 Mengubah kode VHDL menjadi RTL

VHDL perlu dirubah ke bentuk *RTL* terlebih dahulu untuk mendapatkan schematic dari sistem yang dibuat. *Register Transfer Language (RTL)* merupakan fitur yang ada pada program *ISE* yang digunakan untuk mengubah kode *VHDL* menjadi schematic.

3.5 Mengubah RTL menjadi schematic dan kode verilog per blok

Setelah diperoleh RTL maka langkah berikutnya adalah mengubah RTL tersebut menjadi kode verilog dengan menggunakan program DSCH2. Kode verilog adalah kode yang terdiri dari netlist dan library komponen dengan format tertentu, sehingga dengan kode verilog yang terstruktur ini dapat di gunakan untuk membentuk desain CMOS layout pada proses berikutnya.

3.6 Mengubah verilog menjadi CMOS layout teknologi VLSI 0.25 µm

Program microwind adalah merupakan program yang dapat mengubah kode verilog menjadi CMOS layout. Dengan menggunakan program microwind maka dapat dilakukan desain teknologi VLSI dalam ukuran λ yang diinginkan. Proses desain VLSI 0.25 µm dapat diatur dengan cara memilih template yang sudah ada. 0.25 µm merupakan ukuran λ yang menunjukan lebar dari CMOS

layout. Penentuan lebar dari desain CMOS adalah dengan cara pemilihan template Cmos025.rul yang tersimpan di folder dan dapat dipanggil sebagai template desain.

3.7 Menanam sistem ke dalam Xilinx Spartan 3

Untuk mengetahui alur kerja dari kode VHDL maka, kode VHDL perlu ditanam pada Xilinx Spartan 3 dengan memanfaatkan fitur impact atau EPROM impact. Untuk media interfacing antara program ISE di komputer dengan Xilinx Spartan 3 digunakan kabel Jtag. Kabel Jtag merupakan media yang di gunakan untuk upload program VHDL ke Xilinx Spartan 3.

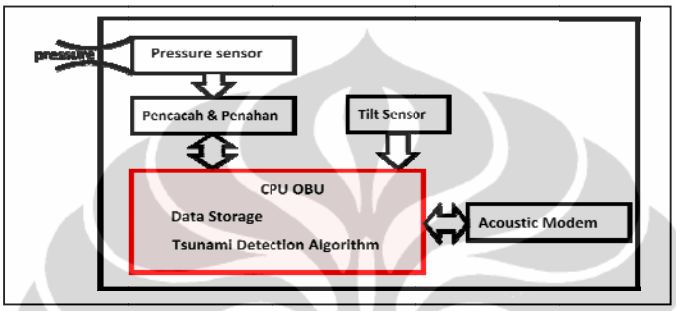
3.8 Demo sistem yang tertanam pada Xilinx Spartan 3 dengam menggunakan program simulasi

Setelah sistem telah tertanam pada Xilinx Spartan 3 maka langkah selanjutnya adalah menguji sistem apakah sesuai dengan kinerja flowchart dan blok diagram yang di inginkan. Untuk menguji kinerja sistem yang tertanam pada Xilinx Spartan 3 diperlukan program bantu yaitu program simulasi yang dapat memberikan input pada sistem sehingga dari input tersebut akan diketahui respon dari sistem apakah sudah berjalan dengan semestinya.

BAB IV
PERANCANGAN CPU OCEAN BOTTOM UNIT TSUNAMI EARLY WARNING SYSTEM VLSI 0.25 µm DENGAN DESAIN HYBRID VHDL

4.1 Tsunameter CPU OBU

4.1.1 Blok Diagram Tsunameter



Gambar 4.1 Blok Diagram *Tsunameter*

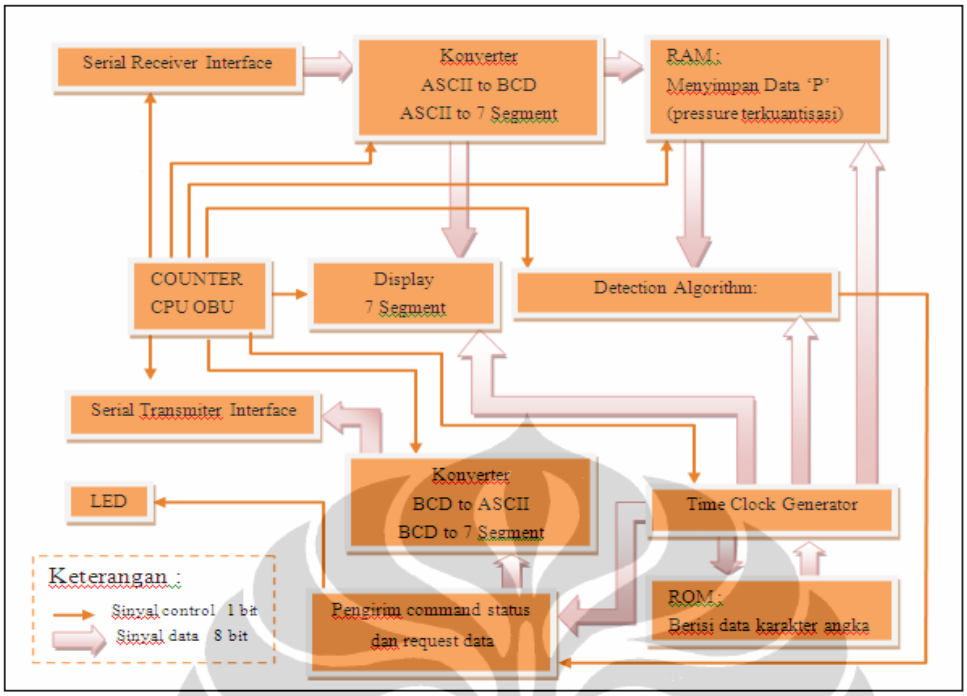
CPU OBU merupakan bagaian dari tsunameter yang berfungsi sebagai pengolah data yang terdiri dari dua bagian utama yaitu :

1. *Data Storage*
2. *Tsunami detection Algorithm*

Data storage digunakan untuk menyimpan data, *Data storage* ini di *CPU OBU* terdiri dari *ROM* dan *RAM*. *ROM* digunakan untuk menyimpan karakter sedangkan *RAM* di gunakan untuk menyimpan data perhitungan dalam algoritma. Sedangkan *tsunami detection algorithm* digunakan algoritma *mofjeld*[5], algoritma ini di *CPU OBU* adalah mencari variable 'PP' (*Prediction Pressure*).

4.1.2 Blok Diagram *CPU OBU*

Tujuan utama dari penelitian ini adalah merancang *CPU OBU* dengan desain *hybrid VHDL* berbasis *FPGA Xilinx Spartan* 3, Untuk lebih jelas memahami modul *CPU OBU* yang akan dirancang dapat dijelaskan dengan gambar blok diagram *CPU OBU* berikut:

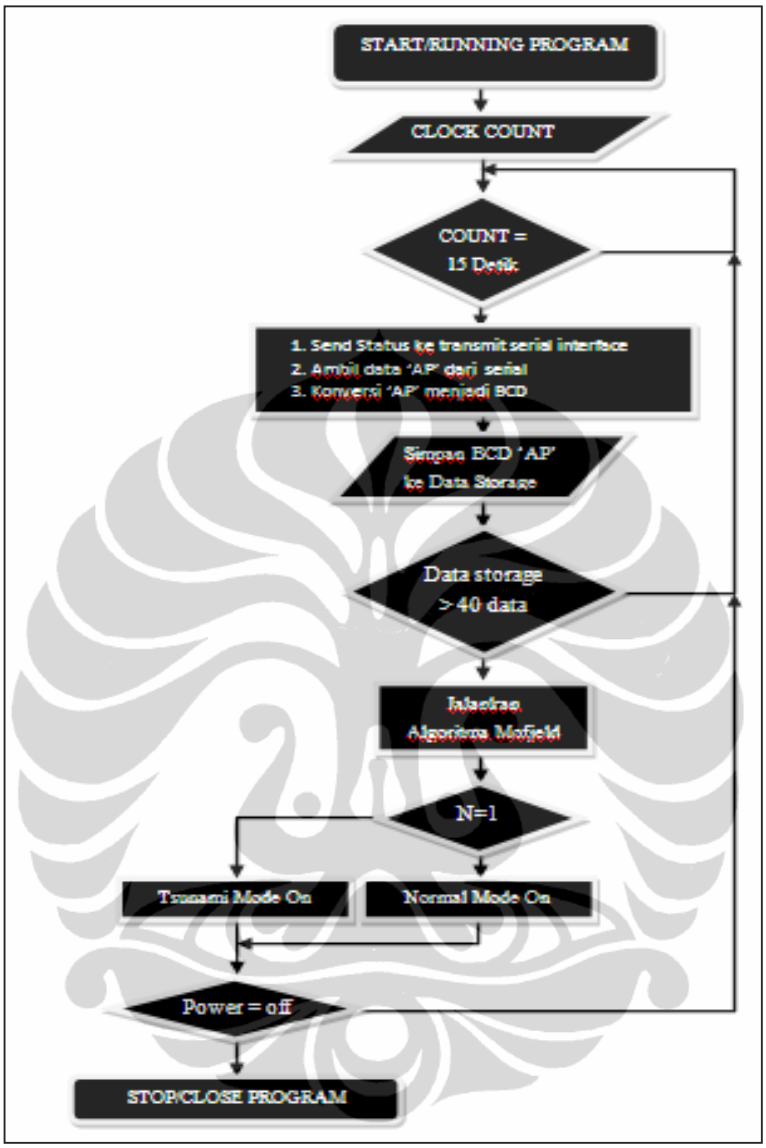


Gambar 4.2 Blok diagram CPU OBU

Secara garis besar, bagian-bagian dari CPU OBU yang akan didisain seperti terlihat pada Gambar 4.2, pada gambar ini ditunjukkan bagian-bagian terpisah dalam suatu blok yang terangkai menjadi satu kesatuan sistem *CPU OBU*. Bagian utama *CPU OBU* ini antara lain adalah :

1. *Serial Receiver interface*
2. Konverter ASCII to BCD dan ASCII *to* 7 *segment*
3. RAM penyimpan data 'P' (*pressure* terkuantisasi)
4. Counter CPU OBU (terdiri dari CTR, CTR1, CTR2, ADD dll)
5. *Display* 7 *segment*
6. *Detection Algorithm*
7. *Serial Transmitter*
8. Konverter BCD to ASCII dan BCD *to* 7 *Segment*
9. *Time Clock Generator*
10. Pengirim *Command* status dan *Command request* data
11. *ROM* berisi data karakter angka

4.1.3 Flow Chart Diagram



Gambar 4.3 *Flowchart* diagram *CPU OBU*

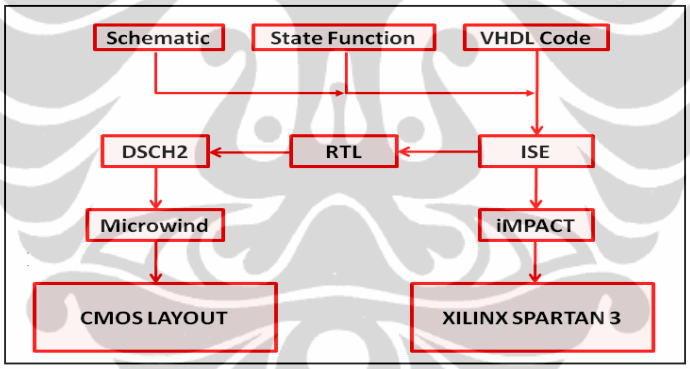
Prinsip kerja dari *CPU OBU* dapat dijelaskan sesuai *flowchart* diagram seperti pada Gambar 4.3, yaitu:

Saat program pertama kali dijalankan maka *counter time* dari CPU OBU menjalankan *time clock generator*. *Time clock generator* menghitung selama 10 menit setelah sepuluh menit *time clock generator* kembali *reset* mengitung dari 0 lagi. Jika *COUNT* < 15 detik maka *No Activity* tidak melakukan apapun, setelah *COUNT* = 15 detik maka *event* yang dilakukan adalah *send* status ke *transmit*

serial interface (jika sudah lebih dari 10 menit), ambil data AP dari receiver serial interface, konversi AP menjadi BCD. Setelah itu simpan data BCD dari AP ke *data storage*. Periksa apakah *data storage* sudah mencapai 40 data jika belum ulang kembali mengambil dan menyimpan data, jika *data storage* mencapai 40 maka jalankan algorithma Mofjeld. Algorithma Mofjeld *learning* data 10 menit sebelumnya untuk mendapatkan variable PP (*prediction pressure)* dan kemudian PP dibandingkan dengan AP (*actual pressure*), jika perbandingan hasil perbandingan kurang dari nilai REF (*refferensi*) maka N =1, T=0 dan jika lebih maka N =0, T =1. Jika N=1 maka sistem masuk ke *Normal mode* sedangkan jika T =1 sistem akan masuk ke *Tsunami mode*. Selama *power* dari sistem masih *on* maka aliran sistem seperti diatas akan di ulang terus menerus, dan jika *power* dari sistem *off* maka sistem berhenti.

4.2 Desain *Hybrid VHDL*

4.2.1 Blok Diagram Proses Desain *Hybrid*



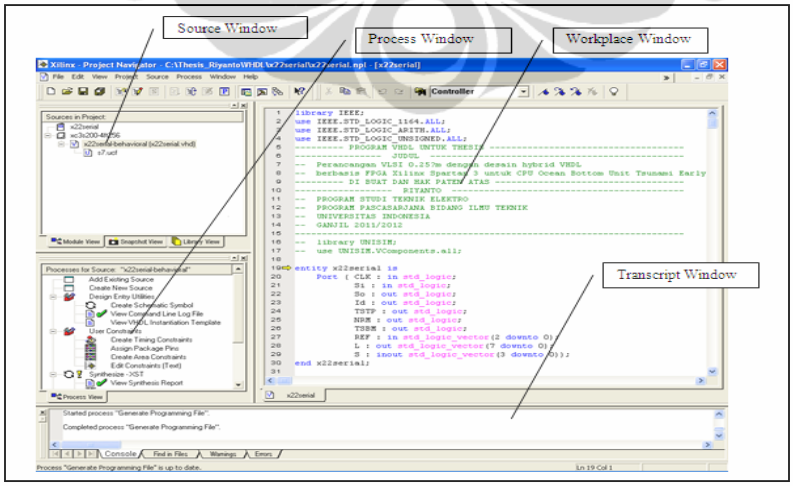
Gambar 4.4 Proses *Design Hybrid* VHDL

Tujuan dari proses *design hybrid* VHDL *configure device* (iMPACT) adalah menanam sistem CPU OBU kedalam *chip* XC3S200 *board* Xilinx Spartan 3, selain itu hasil yang ingin dicapai adalah CMOS *layout* dari sistem CPU OBU. Untuk melakukan proses design *hybrid* VHDL diperlukan tiga program utama yaitu : *Integrated Software Environment* (ISE 6.3i), DSCH2 dan Microwind. Dengan menggunakan ISE 6.3i kita dapat mendesain sistem digital dengan beberapa cara yaitu *Schematic, State function,* dan VHDL *code*. Kita bisa menggunakan salah satu cara atau dapat menggunakan lebih dari satu cara atau gabungan (*hybrid design*). Proses *hybrid design* adalah sebagai berikut :

1. Desain VHDL *code* sesuai dengan arsitektur sistem digital yang diinginkan dengan menggunakan ISE 6.3i.
2. Buat file ucf untuk konfigurasi *entity*.
3. Pada proses *window* klik RTL untuk mendapatkan *top level schematic*, kemudian klik *top level schematic* untuk mendapatkan *schematic* rangkaian.
4. Desain kembali *schematic* rangkaian dengan DSCH2 sesuai dengan RTL dari yang dibuat.
5. Setelah rangkain *schematic* di DSCH2 selesai kemudian klik file *make* verilog file.
6. Setelah mendapatkan file verilog rangkain sistem maka dengan microwind lakukan *compile* verilog file dengan cara klik *compile* kemudian *compile* verilog file. Maka setelah itu akan diperoleh CMOS *layout*. (catatan *select foundry* cmos025.rul untuk mendesain VLSI 0,25 um).
7. Kemudian lakukan proses lain dengan mengklik iMPACT pada proses window ISE 6.3i, untuk menanam sistem dirancang ke dalam *chip* XC3S200 yang ada di Xilinx Spartan4.

4.2.2 *VHDL Code*

Program ISE 6.3i adalah *software* bawaan dari Xilinx Spartan 3 yang digunakan untuk *editing* dan *uploading* desain *hybrid* VHDL. ISE dapat digunakan desainer dengan spektrum penuh, transisi desain ASIC dari CPLD ke FPGA. ISE dapat memberikan informasi atau iktisar tentang proses desain secara *progressif*. Berikut adalah gambar yang menjelaskan bagian dari tampilan ISE 6.3i



Gambar 4.5 Window ISE 6.3i

Berikut cara dan proses desain CPU OBU dengan VHDL *code* :

4.2.2.1 *Entity*

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
--------- PROGRAM VHDL UNTUK THESIS ------------------------------------
----------------  JUDUL  -----------------------------------------------
--  Perancangan VLSI 0.25?m dengan desain hybrid VHDL
--  berbasis FPGA Xilinx Spartan 3 untuk CPU Ocean Bottom Unit Tsunami Early Warnin
--------- DI BUAT DAN HAK PATEN ATAS -----------------------------------
------------------  RIYANTO  -----------------------------------------
--  PROGRAM STUDI TEKNIK ELEKTRO
--  PROGRAM PASCASARJANA BIDANG ILMU TEKNIK
--  UNIVERSITAS INDONESIA
--  GANJIL 2011/2012
----------------------------------------------------------------------
--  library UNISIM;
--  use UNISIM.VComponents.all;

entity x22serial is
    Port ( CLK : in std_logic;
           Si : in std_logic;
           So : out std_logic;
           Id : out std_logic;
           TSTP : out std_logic;
           NRM : out std_logic;
           TSBM : out std_logic;
           REF : in std_logic_vector(2 downto 0);
           L : out std_logic_vector(7 downto 0);
           S : inout std_logic_vector(3 downto 0));
end x22serial;
```

Gambar 4.6 *Entity* CPU OBU

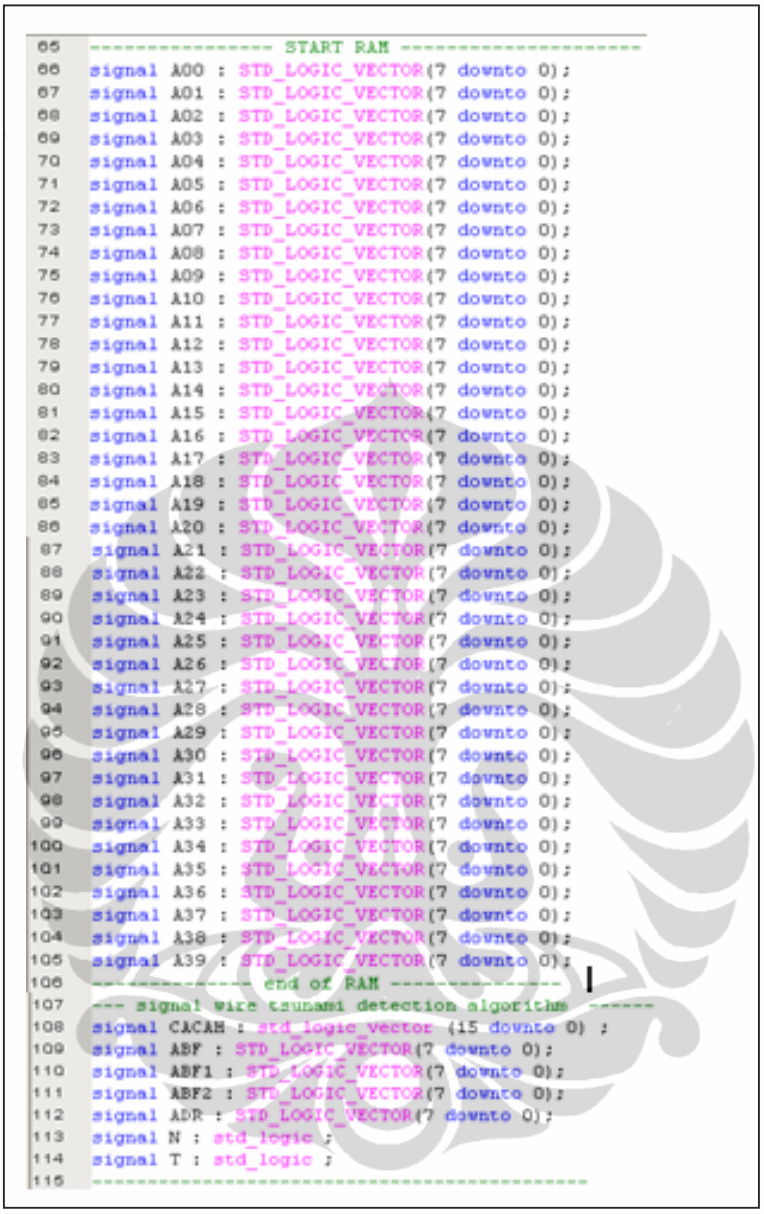
Entity adalah *port* yang digunakan sebagai *input* atau *output* data atau *signal*. *Port* dapat dideklarasikan sebagai *single* (1 bit) atau lebih dari satu bit misalnya 1 byte (8 bit).

4.2.2.2 *Signal*

Signal dapat diartikan sebagai *wire* yang dapat menjadi penghubung antar blok modul di dalam CPU OBU berikut ini adalah deklarasi signal untuk system CPU OBU dalam VHDL.

```
architecture Behavioral of x22serial is
-- signal wire yang di butuhkan
signal tik : std_logic_vector (27 downto 0) ;
signal CTR : std_logic_vector (15 downto 0) ;
signal CTRT : std_logic_vector (15 downto 0) ;
signal B : std_logic_vector (7 downto 0) ;
signal D : std_logic_vector (7 downto 0) ;
signal E : std_logic_vector (2 downto 0) ;
signal F : std_logic_vector (7 downto 0) ;
signal G : std_logic_vector (7 downto 0) ;
signal H : std_logic_vector (7 downto 0) ;
signal C : std_logic_vector (7 downto 0) ;
signal STP : std_logic_vector (3 downto 0) ;
signal Sift : std_logic_vector (3 downto 0) ;
signal STR : std_logic ;
signal STR1 : std_logic ;
signal CTR1 : STD_LOGIC_VECTOR(28 downto 0);
signal CTR2 : STD_LOGIC_VECTOR(28 downto 0);
signal angka0 : STD_LOGIC_VECTOR(7 downto 0);
signal angka1 : STD_LOGIC_VECTOR(7 downto 0);
signal angka2 : STD_LOGIC_VECTOR(7 downto 0);
signal angka3 : STD_LOGIC_VECTOR(7 downto 0);
signal angka4 : STD_LOGIC_VECTOR(7 downto 0);
signal angka5 : STD_LOGIC_VECTOR(7 downto 0);
signal angka6 : STD_LOGIC_VECTOR(7 downto 0);
signal angka7 : STD_LOGIC_VECTOR(7 downto 0);
signal angka8 : STD_LOGIC_VECTOR(7 downto 0);
signal angka9 : STD_LOGIC_VECTOR(7 downto 0);
signal det0 : STD_LOGIC_VECTOR(7 downto 0);
signal det1 : STD_LOGIC_VECTOR(2 downto 0);
signal det2 : STD_LOGIC_VECTOR(7 downto 0);
signal men0 : STD_LOGIC_VECTOR(3 downto 0);
signal men1 : STD_LOGIC_VECTOR(7 downto 0);
```

Gambar 4.7a *Signal CPU OBU* yang dibutuhkan

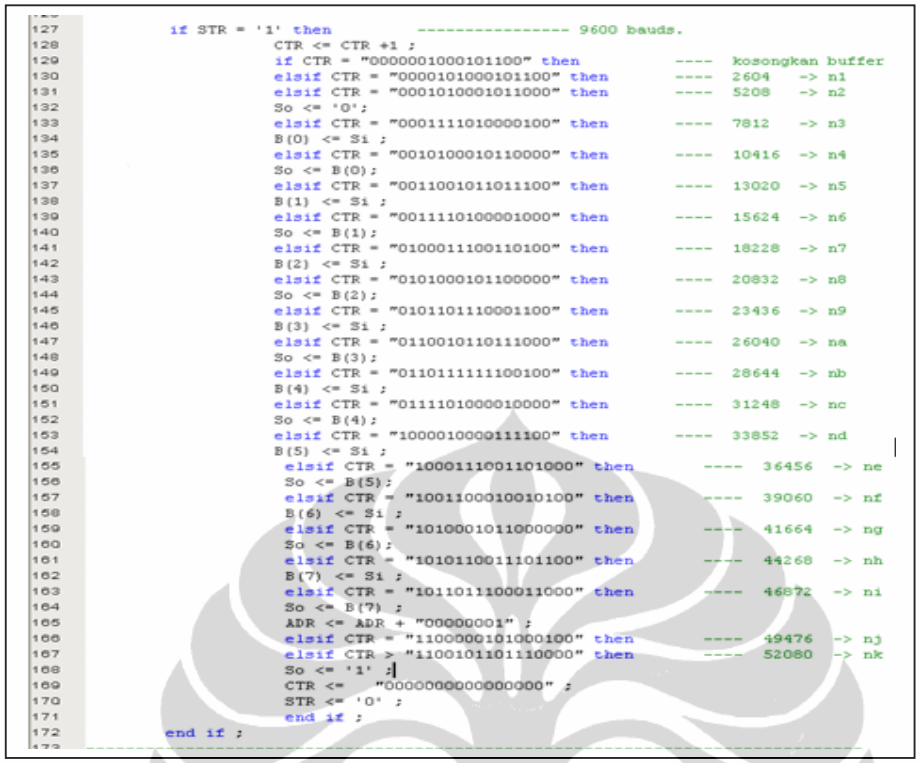


```
65  ---------------- START RAM --------------------
66  signal A00 : STD_LOGIC_VECTOR(7 downto 0);
67  signal A01 : STD_LOGIC_VECTOR(7 downto 0);
68  signal A02 : STD_LOGIC_VECTOR(7 downto 0);
69  signal A03 : STD_LOGIC_VECTOR(7 downto 0);
70  signal A04 : STD_LOGIC_VECTOR(7 downto 0);
71  signal A05 : STD_LOGIC_VECTOR(7 downto 0);
72  signal A06 : STD_LOGIC_VECTOR(7 downto 0);
73  signal A07 : STD_LOGIC_VECTOR(7 downto 0);
74  signal A08 : STD_LOGIC_VECTOR(7 downto 0);
75  signal A09 : STD_LOGIC_VECTOR(7 downto 0);
76  signal A10 : STD_LOGIC_VECTOR(7 downto 0);
77  signal A11 : STD_LOGIC_VECTOR(7 downto 0);
78  signal A12 : STD_LOGIC_VECTOR(7 downto 0);
79  signal A13 : STD_LOGIC_VECTOR(7 downto 0);
80  signal A14 : STD_LOGIC_VECTOR(7 downto 0);
81  signal A15 : STD_LOGIC_VECTOR(7 downto 0);
82  signal A16 : STD_LOGIC_VECTOR(7 downto 0);
83  signal A17 : STD_LOGIC_VECTOR(7 downto 0);
84  signal A18 : STD_LOGIC_VECTOR(7 downto 0);
85  signal A19 : STD_LOGIC_VECTOR(7 downto 0);
86  signal A20 : STD_LOGIC_VECTOR(7 downto 0);
87  signal A21 : STD_LOGIC_VECTOR(7 downto 0);
88  signal A22 : STD_LOGIC_VECTOR(7 downto 0);
89  signal A23 : STD_LOGIC_VECTOR(7 downto 0);
90  signal A24 : STD_LOGIC_VECTOR(7 downto 0);
91  signal A25 : STD_LOGIC_VECTOR(7 downto 0);
92  signal A26 : STD_LOGIC_VECTOR(7 downto 0);
93  signal A27 : STD_LOGIC_VECTOR(7 downto 0);
94  signal A28 : STD_LOGIC_VECTOR(7 downto 0);
95  signal A29 : STD_LOGIC_VECTOR(7 downto 0);
96  signal A30 : STD_LOGIC_VECTOR(7 downto 0);
97  signal A31 : STD_LOGIC_VECTOR(7 downto 0);
98  signal A32 : STD_LOGIC_VECTOR(7 downto 0);
99  signal A33 : STD_LOGIC_VECTOR(7 downto 0);
100 signal A34 : STD_LOGIC_VECTOR(7 downto 0);
101 signal A35 : STD_LOGIC_VECTOR(7 downto 0);
102 signal A36 : STD_LOGIC_VECTOR(7 downto 0);
103 signal A37 : STD_LOGIC_VECTOR(7 downto 0);
104 signal A38 : STD_LOGIC_VECTOR(7 downto 0);
105 signal A39 : STD_LOGIC_VECTOR(7 downto 0);
106 ---------------- end of RAM ---------------
107 --- signal wire tsunami detection algorithm ------
108 signal CACAH : std_logic_vector (15 downto 0) ;
109 signal ABF : STD_LOGIC_VECTOR(7 downto 0);
110 signal ABF1 : STD_LOGIC_VECTOR(7 downto 0);
111 signal ABF2 : STD_LOGIC_VECTOR(7 downto 0);
112 signal ADR : STD_LOGIC_VECTOR(7 downto 0);
113 signal N : std_logic ;
114 signal T : std_logic ;
115 -------------------------------------------
```

Gambar 4.7b *Signal CPU OBU* untuk *RAM* dan keperluan algoritma

4.2.2.3 *Serial Receiver Interface*

Serial Receiver Interface pada desain *VHDL* dibuat dengan *bautrate* 9600, dengan format ASCII 8 bit.



```
127         if STR = '1' then          ---------------- 9600 bauds.
128                 CTR <= CTR +1 ;
129                 if CTR = "0000001000101100" then          ----  kosongkan buffer
130                 elsif CTR = "0000101000101100" then       ----  2604   -> n1
131                 elsif CTR = "0000101000101100" then       ----  5208   -> n2
132                 So <= '0';
133                 elsif CTR = "0001111010000100" then       ----  7812   -> n3
134                 B(0) <= Si ;
135                 elsif CTR = "0010100010110000" then       ----  10416  -> n4
136                 So <= B(0);
137                 elsif CTR = "0011001011011100" then       ----  13020  -> n5
138                 B(1) <= Si ;
139                 elsif CTR = "0011110100001000" then       ----  15624  -> n6
140                 So <= B(1);
141                 elsif CTR = "0100011100110100" then       ----  18228  -> n7
142                 B(2) <= Si ;
143                 elsif CTR = "0101000101100000" then       ----  20832  -> n8
144                 So <= B(2);
145                 elsif CTR = "0101101110001100" then       ----  23436  -> n9
146                 B(3) <= Si ;
147                 elsif CTR = "0110010110111000" then       ----  26040  -> na
148                 So <= B(3);
149                 elsif CTR = "0110111111100100" then       ----  28644  -> nb
150                 B(4) <= Si ;
151                 elsif CTR = "0111101000010000" then       ----  31248  -> nc
152                 So <= B(4);
153                 elsif CTR = "1000010000111100" then       ----  33852  -> nd
154                 B(5) <= Si ;
155                  elsif CTR = "1000111001101000" then        ----  36456  -> ne
156                  So <= B(5);
157                  elsif CTR = "1001100010010100" then        ----  39060  -> nf
158                  B(6) <= Si ;
159                  elsif CTR = "1010001011000000" then        ----  41664  -> ng
160                  So <= B(6);
161                  elsif CTR = "1010110011101100" then        ----  44268  -> nh
162                  B(7) <= Si ;
163                  elsif CTR = "1011011100011000" then        ----  46872  -> ni
164                  So <= B(7) ;
165                  ADR <= ADR + "00000001" ;
166                  elsif CTR = "1100000101000100" then        ----  49476  -> nj
167                  elsif CTR > "1100101101110000" then        ----  52080  -> nk
168                  So <= '1' ;
169                  CTR <=   "0000000000000000" ;
170                  STR <= '0' ;
171                  end if ;
172         end if ;
```

Gambar 4.8 *Serial Receiver Interface*

Dari kode *VHDL Serial Receiver Interface* dapat dijelaskan bahwa *CLK* adalah merupakan *Cristal* 50MHz yang ada di alamat (T9, GCLK0) dari *board* Xiling Spartan 3 ini berarti bahwa kita harus membagi frekuensi *cristal* sesuai dengan seper empat dari *bautrate* atau 9600 /4 = 2400. Jadi daerah cuplik bit effektif antara 0 sampai 4800 atau *typecal* di 2400. Pencupilkan berikutnya daerah effektif pencuplikan antara 4800 sampai 9600 *typecal* di 7200 dan seterusnya untuk pencuplikan bit berikutnya. Ternyata hasil dari *experiment* didapatkan daerah cuplik data sebagai berikut :

Data Biner yang di cuplik	**Bit Counter Cuplik (CTR 15 downto 0)**
B(0) <= Si ;	**0001111010000100**
B(1) <= Si ;	**0011001011011100**
B(2) <= Si ;	**0100011100110100**
B(3) <= Si ;	**0101101110001100**
B(4) <= Si ;	**0110111111100100**
B(5) <= Si ;	**1000010000111100**
B(6) <= Si ;	**1001100010010100**
B(7) <= Si ;	**1010110011101100**

Tabel 4.1 Pewaktu pencuplikan data ASCII serial *interface*

4.2.2.4 *Converter*

Fungsi dari *converter* adalah untuk mengubah format data, dalam sistem *CPU OBU* diperlukan pengubahan format data yaitu dari *ASCII* ke *BCD*, *ASCII* ke *Seven Segment*, atau dari *BCD* ke *ASCII, BCD* ke *Seven Segment*. Pengubahan data dalam sistem diperlukan untuk menampilkan angka karakter ke *display seven segment* atau untuk dilakukan operasi matematik dalam perhitungan algoritma dan untuk komunikasi serial agar sesuai standar *ASCII*. Berikut adalah *Converter* dalam *VHDL code* yang sudah dibuat untuk mengubah format data.

```
-- ASCII to Character --
if B = "00110001" then        -- 1
D <= "11111001" ;
ABF1 <= "00000001" ;
end if ;
if B = "00110010" then        -- 2
D <= "00100100" ;
ABF1 <= "00000010" ;
end if ;
if B = "00110011" then        -- 3
D <= "00110000" ;
ABF1 <= "00000011" ;
end if ;
if B = "00110100" then        -- 4
D <= "00011001" ;
ABF1 <= "00000100" ;
end if ;
if B = "00110101" then        -- 5
D <= "00010010" ;
ABF1 <= "00000101" ;
end if ;
if B = "00110110" then        -- 6
D <= "00000010" ;
ABF1 <= "00000110" ;
end if ;
if B = "00110111" then        -- 7
D <= "11111000" ;
ABF1 <= "00000111" ;
end if ;
if B = "00111000" then        -- 8
D <= "00000000" ;
ABF1 <= "00001000" ;
end if ;
if B = "00111001" then        -- 9
D <= "00010000" ;
ABF1 <= "00001001" ;
end if ;
if B = "00110000" then        -- 0
D <= "01000000" ;
ABF1 <= "00000000" ;
end if ;
-------------------------------------
```

(a)

```
-- BCD to ASCII  --
if ABF = "00000001" then          -- 1
ABF2 <= "00110001" ;
end if ;
if ABF = "00000010" then          -- 2
ABF2 <= "00110010" ;
end if ;
if ABF = "00000011" then          -- 3
ABF2 <= "00110011" ;
end if ;
if ABF = "00000100" then          -- 4
ABF2 <= "00110100" ;
end if ;
if ABF = "00000101" then          -- 5
ABF2 <= "00110101" ;
end if ;
if ABF = "00000110" then          -- 6
ABF2 <= "00110110" ;
end if ;
if ABF = "00000111" then          -- 7
ABF2 <= "00110111" ;
end if ;
if ABF = "00001000" then          -- 8
ABF2 <= "00111000" ;
end if ;
if ABF = "00001001" then          -- 9
ABF2 <= "00111001" ;
end if ;
if ABF = "00001010" then          -- 0
ABF2 <= "00110000" ;
end if ;
-----------------------------------------
```

(b)

Gambar 4.9 (a) Konverter dari ASCII ke BCD dan *seven segment*
(b) Konverter dari BCD ke ASCII

4.2.2.5 Mengirim *Command Request Data* dan Status

Command request data adalah karakter *ASCII* yang dikirim dari *CPU OBU* ke *BPR* untuk meminta *Actual Pressure data*. *Command Status* adalah

karakter *ASCII* yang dikirim dari *CPU OBU* ke *BUOY* sebagai *signal trigger* apabila terjadi *tsunami* (untuk keperluan penelitian *command status* dikirim ke *computer* atau program simulasi untuk informasi). Gambar 4.10 adalah kode *VHDL command request data* dan *status*. Dari kode ini dapat dijelaskan bahwa det2 dan det0 masing masing mewakili karakter angka dimana det2 adalah karakter detik digit kedua, sedangkan det0 adalah karakter detik digit pertama. Sedangkan *signal* 'C' adalah karakter 'P' atau 'T' dalam format *ASCII* yang dikirim ke serial *transmitter interface*. P menunjukan bahwa status dalam kondisi normal sedangkan T menunjukkan bahwa status dalam kondisi *tsunami*. Baik 'P' ataupun 'T' adalah sama-sama *command* untuk *request data*.

```
248   if N = '1' then
249   -------    Mengirim Status Normal  ------------------
250   -------    Mengirim Request Data per Lima Belas Detik  ------------------
251   if (det2 = "01111001"  and  det0= "00010010" ) then              --- detik 15
252   C <= "01010000" ;
253   STR1 <= '1'  ;
254   elsif ( det2 = "00110000"  and  det0= "01000000" ) then          --- detik 30
255   C <= "01010000" ;
256   STR1 <= '1'  ;
257   elsif ( det2 = "00011001"  and det0 = "00010010" ) then          --- detik 45
258   C <= "01010000" ;
259   STR1 <= '1'  ;
260   elsif ( det2 = "01000000"  and det0 = "01000000" ) then          --- detik "00"
261   C <= "01010000" ;
262   STR1 <= '1'  ;
263   end if ;
264   end if ;
265
266   if T = '1' then
267   -------    Mengirim Status Tsunami  ------------------
268   if (det2 = "01111001"  and  det0= "00010010" ) then        --- detik 15
269   C <= "01010100" ;
270   STR1 <= '1'  ;
271   elsif (det2 = "00110000"  and  det0= "01000000" ) then     --- detik 30
272   C <= "01010100" ;
273   STR1 <= '1'  ;
274   elsif (det2 = "00011001"  and det0 = "00010010" ) then     --- detik 45
275   C <= "01010100" ;
276   STR1 <= '1'  ;
277   elsif (det2 = "01000000"  and det0 = "01000000") then      --- detik "00"
278   C <= "01010100" ;
279   STR1 <= '1'  ;
280   end if ;
281   end if ;
```

Gambar 4.10 Pengirim *command status* dan *request data*

4.2.2.6 *Serial Transmitter Interface*

Serial Transmitter Interface seperti halnya *Serial Receiver Interface* hanya fungsi nya merupakan kebalikannya.

```
284  if STR1 = '1' then
285  CTRT <= CTRT +1 ;
286  end if ;
287                     if CTRT = "0000001000101100" then         ----  kosongkan buffer
288                     So <= '0' ;
289                     elsif CTRT = "0000101000101100" then      ----  2604   -> n1
290                     elsif CTRT = "0001010001011000" then      ----  5208   -> n2
291                     elsif CTRT = "0001111010000100" then      ----  7812   -> n3
292                     So <= C(0);
293                     elsif CTRT = "0010100010110000" then      ----  10416  -> n4
294                     elsif CTRT = "0011001011011100" then      ----  13020  -> n5
295                     So <= C(1);
296                     elsif CTRT = "0011110100001000" then      ----  15624  -> n6
297                     elsif CTRT = "0100011100110100" then      ----  18228  -> n7
298                     So <= C(2);
299                     elsif CTRT = "0101000101100000" then      ----  20832  -> n8
300                     elsif CTRT = "0101101110001100" then      ----  23436  -> n9
301                     So <= C(3);
302                     elsif CTRT = "0110010110111000" then      ----  26040  -> na
303                     elsif CTRT = "0110111111100100" then      ----  28644  -> nb
304                     So <= C(4);
305                     elsif CTRT = "0111101000010000" then      ----  31248  -> nc
306                     elsif CTRT = "1000010000111100" then      ----  33852  -> nd
307                     So <= C(5);
308                     elsif CTRT = "1000111001101000" then      ----  36456  -> ne
309                     elsif CTRT = "1001100010010100" then      ----  39060  -> nf
310                     So <= C(6);
311                     elsif CTRT = "1010001011000000" then      ----  41664  -> ng
312                     elsif CTRT = "1010110011101100" then      ----  44268  -> nh
313                     So <= C(7);
314                     elsif CTRT = "1011011100011000" then      ----  46872  -> ni
315                     elsif CTRT = "1100000101000100" then      ----  49476  -> nj
316                     So <= '1';
317                     elsif CTRT > "1100101101110000" then      ----  52080  -> nk
318                     STP <= STP +1 ;
319                     end if ;
```

Gambar 4.11 *Serial Transmitter Interface*

4.2.2.7 *Counter* dan *Sifter Display Seven Segment*

Counter adalah penghitung akumulasi naik, *counter* berguna untuk membagi frekuensi *cristal* sesuai dengan *counter clock generator* yang diinginkan. *Sifter display seven segment* berfungsi untuk tujuan menampilkan data ke nyala LED *seven segment*. Gambar 4.12 berikut adalah kode VHDL *counter* dan *sifter display seven segment.*

```
320  ------------------------- delay transmitter ------------------------------------
321  if STP = "1111" then
322   CTRT <= "0000000000000000" ;
323   STR1 <= '0' ;
324  end if ;
325  ----------------- COUNTER CTR1 dan CTR2 untuk Memory dan Aalgorithm -----------------
326        CTR1 <= CTR1 + "00000000000000000000000000000001" ;
327        if (CTR1 > "11101110011010110010100000000") then   -- counter reaches 2^13
328        CTR1 <="00000000000000000000000000000000";
329        end if;
330        -------------  Mengatur kecepatan geser SEVEN SEGMEN
331        CTR2 <= CTR2 + "00000000000000000000000000000001" ;
332        if (CTR2 > "00000000000000000010000000011111") then   -- counter reaches 2^13
333        CTR2 <="00000000000000000000000000000000";
334        end if;
335  -------  Sifter display 7 segment --
336   if (CTR2 >"00000000000000000000000000000" and CTR2<"00000000000000000000100000") then
337          if (S(0)='0') then
338            S(0) <= '1';
339            L <= det2;              -- Digit 2  Detik Bit Kedua
340            S(1) <= '0';
341          elsif (S(1)='0') then
342            S(1) <= '1';
343            L <= men1;              -- Digit 3  Menit Bit Pertama
344            S(2) <= '0';
345          elsif (S(2)='0') then
346            S(2) <= '1';
347            L <= G;                 -- MSB  Data serial
348            S(3) <= '0';
349          elsif (S(3)='0') then
350            S(3) <= '1';
351            L <= det0;              -- Digit 1  Detik Bit Pertama
352            S(0) <= '0';
353          end if;
354        end if;
355  ------------------------------------------------------
```

Gambar 4.12 *Counter* dan *sifter seven segment*

4.2.2.8 Tulis dan Baca *RAM* untuk Algoritma Deteksi *Tsunami*

Program *VHDL* untuk tulis dan baca *RAM* untuk algoritma deteksi *tsunami* seperti terlihat pada Gambar 4.13a dan 4.13b menunjukkan bahwa *ADR* merupakan *counter* untuk menyimpan data ABF1 (*actual pressure*) ke alamat *RAM* yaitu Axx. Proses penyimpanan menunggu *sifter seven segment* S(2) adalah 0 artinya penyimpanan data yaitu pada saat *update* data *actual pressure*. Sedangkan pada bagian bawah program di tunjukkan persamaan rata-rata selama 10 menit ditunjukan oleh *variable* CACAH.

```
355  ------------------------------------
356  if (ADR > "00100111") then
357  ADR <= "00000000" ;
358  end if ;
359  ---------- Tulis RAM ( READ ACCESS M
360  if S(2) = '0' then
361       if ADR ="00000001" then
362       AO1 <= ABF1 ;
363       elsif ADR ="00000010" then
364       AO2 <= ABF1 ;
365       elsif ADR ="00000011" then
366       AO3 <= ABF1 ;
367       elsif ADR ="00000100" then
368       AO4 <= ABF1 ;
369       elsif ADR ="00000101" then
370       AO5 <= ABF1 ;
371       elsif ADR ="00000110" then
372       AO6 <= ABF1 ;
373       elsif ADR ="00000111" then
374       AO7 <= ABF1 ;
375       elsif ADR ="00001000" then
376       AO8 <= ABF1 ;
377       elsif ADR ="00001001" then
378       AO9 <= ABF1 ;
379       elsif ADR ="00001010" then
380       A10 <= ABF1 ;
381       elsif ADR ="00001011" then
382       A11 <= ABF1 ;
383       elsif ADR ="00001100" then
384       A12 <= ABF1 ;
385       elsif ADR ="00001101" then
386       A13 <= ABF1 ;
387       elsif ADR ="00001110" then
388       A14 <= ABF1 ;
389       elsif ADR ="00001111" then
```

```
390       A15 <= ABF1 ;
391       elsif ADR ="00010000" then
392       A16 <= ABF1 ;
393       elsif ADR ="00010001" then
394       A17 <= ABF1 ;
395       elsif ADR ="00010010" then
396       A18 <= ABF1 ;
397       elsif ADR ="00010011" then
398       A19 <= ABF1 ;
399       elsif ADR ="00010100" then
400       A20 <= ABF1 ;
401       elsif ADR ="00010101" then
402       A21 <= ABF1 ;
403       elsif ADR ="00010110" then
404       A22 <= ABF1 ;
405       elsif ADR ="00010111" then
406       A23 <= ABF1 ;
407       elsif ADR ="00011000" then
408       A24 <= ABF1 ;
409       elsif ADR ="00011001" then
410       A25 <= ABF1 ;
411       elsif ADR ="00011010" then
412       A26 <= ABF1 ;
413       elsif ADR ="00011011" then
414       A27 <= ABF1 ;
415       elsif ADR ="00011100" then
416       A28 <= ABF1 ;
417       elsif ADR ="00011101" then
418       A29 <= ABF1 ;
419       elsif ADR ="00011110" then
420       A30 <= ABF1 ;
421       elsif ADR ="00011111" then
```

Gambar 4.13a Tulis *RAM* 30 dari 40 data

```
419       elsif ADR ="00011110" then
420       A30 <= ABF1 ;
421       elsif ADR ="00011111" then
422       A31 <= ABF1 ;
423       elsif ADR ="00100000" then
424       A32 <= ABF1 ;
425       elsif ADR ="00100001" then
426       A33 <= ABF1 ;
427       elsif ADR ="00100010" then
428       A34 <= ABF1 ;
429       elsif ADR ="00100011" then
430       A35 <= ABF1 ;
431       elsif ADR ="00100100" then
432       A36 <= ABF1 ;
433       elsif ADR ="00100101" then
434       A37 <= ABF1 ;
435       elsif ADR ="00100110" then
436       A38 <= ABF1 ;
437       elsif ADR ="00100111" then
438       A39 <= ABF1 ;
439       end if;
440   ------------------------------------------------------------------------
441   --------------  Baca RAM dan Tsunami Detection Algorithm   ---------------------
442   if A39 > "00000000" then
443   Id <= '1' ;
444   CACAH <= "00" & ((A01 + A02 + A03 + A04 + A05 + A06 + A07 + A08 + A09 + A10 + A11 + A12
445   A39 <= "00000000" ;
446   end if ;
447   if A39 < "00000001" then
448   Id <= '0' ;
449   end if ;
```

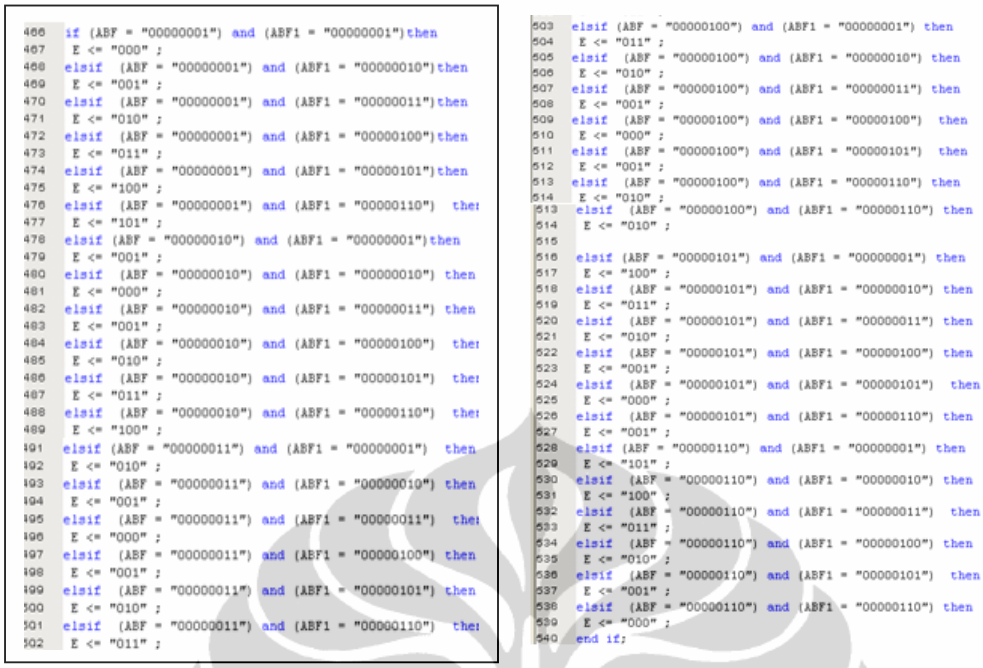
Gambar 4.13b Tulis *RAM* 10 dari 40 data dan baca *RAM* untuk algoritma

4.2.2.9 Algoritma Deteksi Tsunami

```
451   if (CACAH >= "0010001001100000") and (CACAH < "0010100010100000") then
452   ABF <= "00000110" ;
453   elsif (CACAH >= "0001110000100000") and (CACAH < "0010001001100000") then
454   ABF <= "00000101" ;
455   elsif (CACAH >= "0001010111100000") and (CACAH < "0001110000100000") then
456   ABF <= "00000100" ;
457   elsif (CACAH >= "0000111110100000") and (CACAH < "0001010111100000") then
458   ABF <= "00000011" ;
459   elsif (CACAH >= "0000100101100000") and (CACAH < "0000111110100000") then
460   ABF <= "00000010" ;
461   elsif (CACAH >  "0000000000000000") and (CACAH < "0000100101100000") then
462   ABF <= "00000001" ;
463   end if ;
```

Gambar 4.14 *Prediction Pressure* hasil dari algoritma Mofjeld

Variabel CACAH adalah jumlah 40 data selama 10 menit di kali 40, *variable* CACAH dieksekusi pada saat *RAM address* A39 tidak sama dengan 0, atau dengan kata lain A39 telah terisi data ini berarti pada saat RAM penuh. Setelah nilai CACAH didapat maka nilai cacah di kalikan dengan 40 (dalam data biner), setelah itu hasil kali CACAH dengan 40 dibatasi untuk mendapatkan ABF. ABF inilah yang merupakan *prediction pressure* .



```
if (ABF = "00000001") and (ABF1 = "00000001")then
 E <= "000" ;
elsif  (ABF = "00000001") and (ABF1 = "00000010")then
 E <= "001" ;
elsif  (ABF = "00000001") and (ABF1 = "00000011")then
 E <= "010" ;
elsif  (ABF = "00000001") and (ABF1 = "00000100")then
 E <= "011" ;
elsif  (ABF = "00000001") and (ABF1 = "00000101")then
 E <= "100" ;
elsif  (ABF = "00000001") and (ABF1 = "00000110")  then
 E <= "101" ;
elsif (ABF = "00000010") and (ABF1 = "00000001")then
 E <= "001" ;
elsif  (ABF = "00000010") and (ABF1 = "00000010") then
 E <= "000" ;
elsif  (ABF = "00000010") and (ABF1 = "00000011") then
 E <= "001" ;
elsif  (ABF = "00000010") and (ABF1 = "00000100")  then
 E <= "010" ;
elsif  (ABF = "00000010") and (ABF1 = "00000101")  then
 E <= "011" ;
elsif  (ABF = "00000010") and (ABF1 = "00000110")  then
 E <= "100" ;
elsif (ABF = "00000011") and (ABF1 = "00000001")  then
 E <= "010" ;
elsif  (ABF = "00000011") and (ABF1 = "00000010") then
 E <= "001" ;
elsif  (ABF = "00000011") and (ABF1 = "00000011")  then
 E <= "000" ;
elsif  (ABF = "00000011") and (ABF1 = "00000100") then
 E <= "001" ;
elsif  (ABF = "00000011") and (ABF1 = "00000101") then
 E <= "010" ;
elsif  (ABF = "00000011") and (ABF1 = "00000110")  then
 E <= "011" ;
```

```
elsif (ABF = "00000100") and (ABF1 = "00000001") then
 E <= "011" ;
elsif  (ABF = "00000100") and (ABF1 = "00000010") then
 E <= "010" ;
elsif  (ABF = "00000100") and (ABF1 = "00000011") then
 E <= "001" ;
elsif  (ABF = "00000100") and (ABF1 = "00000100")  then
 E <= "000" ;
elsif  (ABF = "00000100") and (ABF1 = "00000101")  then
 E <= "001" ;
elsif  (ABF = "00000100") and (ABF1 = "00000110") then
 E <= "010" ;
elsif  (ABF = "00000100") and (ABF1 = "00000110") then
 E <= "010" ;

elsif (ABF = "00000101") and (ABF1 = "00000001") then
 E <= "100" ;
elsif  (ABF = "00000101") and (ABF1 = "00000010") then
 E <= "011" ;
elsif  (ABF = "00000101") and (ABF1 = "00000011") then
 E <= "010" ;
elsif  (ABF = "00000101") and (ABF1 = "00000100") then
 E <= "001" ;
elsif  (ABF = "00000101") and (ABF1 = "00000101")  then
 E <= "000" ;
elsif  (ABF = "00000101") and (ABF1 = "00000110") then
 E <= "001" ;
elsif (ABF = "00000110") and (ABF1 = "00000001") then
 E <= "101" ;
elsif  (ABF = "00000110") and (ABF1 = "00000010") then
 E <= "100" ;
elsif  (ABF = "00000110") and (ABF1 = "00000011")  then
 E <= "011" ;
elsif  (ABF = "00000110") and (ABF1 = "00000100") then
 E <= "010" ;
elsif  (ABF = "00000110") and (ABF1 = "00000101")  then
 E <= "001" ;
elsif  (ABF = "00000110") and (ABF1 = "00000110") then
 E <= "000" ;
end if;
```

Gambar 4.15 Perbandingan *Prediction Pressure* dengan *Actual Pressure*

Pada Gambar 4.15 *variable* ABF (*Prediction pressure*) dengan *variabel* ABF1 (*Actual Pressure*) dibandingkan sehingga didapatkan *variabel* E, kemudian *variabel* E ini dibatasi oleh nilai REF. jika E di bawah nilai REF maka status *Normal mode*, sedangkan jika nilai E di atas nilai REF maka status *Tsunami mode*

```
if E < REF then
 TSBM <= '0' ;
 NRM  <= '1' ;
 TSTP <= '0' ;
 T <= '0' ;
 N <= '1' ;
end if;

if E > REF then
 TSTP <= '1' ;
 NRM  <= '0' ;
 TSBM <= '1' ;
 T <= '1' ;
 N <= '0' ;
end if ;
end if;

H <= ABF1 ;
```

Gambar 4.16 REF sebagai batas deteksi Tsunami

Perubahan status di tandai oleh perubahan T dan N, jika T = 1 adalah menunjukkan kondisi *tsunami mode*, sedangkan N = 1 adalah kondisi *normal mode*.

4.2.2.10 *Converter* tambahan untuk keperluan sistem

Seperti dijelaskan pada sub bab 4.2.2.d tentang *converter*, *converter* disini seperti *converter* sebelumnya, penambahan convereter ini diperlukan untuk keperluan sistem.

```
562   -- BCD to ASCII  --
563   if H = "00000001" then        -- 1
564   F <= "00110001" ;
565   end if ;
566   if H = "00000010" then        -- 2
567   F <= "00110010" ;
568   end if ;
569   if H = "00000011" then        -- 3
570   F <= "00110011" ;
571   end if ;
572   if H = "00000100" then        -- 4
573   F <= "00110100" ;
574   end if ;
575   if H = "00000101" then        -- 5
576   F <= "00110101" ;
577   end if ;
578   if H = "00000110" then        -- 6
579   F <= "00110110" ;
580   end if ;
581   if H = "00000111" then        -- 7
582   F <= "00110111" ;
583   end if ;
584   if H = "00001000" then        -- 8
585   F <= "00111000" ;
586   end if ;
587   if H = "00001001" then        -- 9
588   F <= "00111001" ;
589   end if ;
590   if H = "00001010" then        -- 0
591   F <= "00110000" ;
592   end if ;
```

```
594   -- ASCII to Character --
595   if F = "00110001" then        -- 1
596   G <= "11111001" ;
597   end if ;
598   if F = "00110010" then        -- 2
599   G <= "00100100" ;
600   end if ;
601   if F = "00110011" then        -- 3
602   G <= "00110000" ;
603   end if ;
604   if F = "00110100" then        -- 4
605   G <= "00011001" ;
606   end if ;
607   if F = "00110101" then        -- 5
608   G <= "00010010" ;
609   end if ;
610   if F = "00110110" then        -- 6
611   G <= "00000010" ;
612   end if ;
613   if F = "00110111" then        -- 7
614   G <= "11111000" ;
615   end if ;
616   if F = "00111000" then        -- 8
617   G <= "00000000" ;
618   end if ;
619   if F = "00111001" then        -- 9
620   G <= "00010000" ;
621   end if ;
622   if F = "00110000" then        -- 0
623   G <= "01000000" ;
624   end if ;
```

Gambar 4.17 *Converter* tambahan untuk keperluan sistem

4.2.2.11 *Time Clock Generator*

Time Clock Generator adalah bagian yang membangkitkan *counter clock* antara 0 sampai 10 menit. Gambar 4.18 menunjukan pembangkit *counter* detik pertama, Gambar 4.19 menunjukkan pembangkit *counter* detik digit kedua dan Gambar 4.20 menunjukan pembangkit *counter* menit digit ke tiga

```
626   ------------ TIME / CLOCK, pengatur pewaktuan 15 detik dan 10 menit dan Baca ROM --
627    ---  detik digit pertama
628    if  (CTR1="00010111110101111000010000000" )      then     --1
629                det0 <= angka1    ;
630    elsif  (CTR1="00101111101011110000100000000"  ) then     --2
631                det0 <= angka2    ;
632    elsif  (CTR1="01000111100001101000110000000"  ) then     --3
633                det0 <= angka3    ;
634    elsif  (CTR1="01011111010111100001000000000"  ) then     --4
635                det0 <= angka4    ;
636    elsif  (CTR1="01110111001101011001010000000"  ) then     --5
637                det0 <= angka5    ;
638    elsif  (CTR1="10001111000011010001100000000"  ) then     --6
639                det0 <= angka6    ;
640    elsif  (CTR1="10100110111001001001110000000" )  then     --7
641                det0 <= angka7    ;
642    elsif  (CTR1="10111110101111000010000000000"  ) then     --8
643                det0 <= angka8    ;
644    elsif  (CTR1="11010110100100111010010000000"  ) then    --9
645                det0 <= angka9    ;
646    elsif  (CTR1="11101110011010110010100000000" )  then     --0
647                det0 <= angka0    ;
648                det1 <= det1 +1 ;
649    end if   ;
```

Gambar 4.18 *Counter* detik digit pertama

```
651   ---  detik digit kedua
652       if    det1 = "000"  then
653       det2 <= angka0 ;
654       elsif det1 = "001"  then
655       det2 <= angka1 ;
656       elsif det1 = "010"  then
657       det2 <= angka2 ;
658       elsif det1 = "011"  then
659       det2 <= angka3 ;
660       elsif det1 = "100"  then
661       det2 <= angka4 ;
662       elsif det1 = "101"  then
663       det2 <= angka5 ;
664       elsif det1 = "110"  then
665       det1 <= "000" ;
666       men0 <= men0 +1 ;
667       end if ;
```

Gambar 4.19 *Counter* detik digit kedua

```
668   --- menit digit pertama
669       if  men0 = "0000"  then
670       men1 <= angka0 ;
671       elsif men0 = "0001"  then
672       men1 <= angka1 ;
673       elsif men0 = "0010"  then
674       men1 <= angka2 ;
675            elsif men0 = "0011"  then
676       men1 <= angka3 ;
677       elsif men0 = "0100"  then
678       men1 <= angka4 ;
679            elsif men0 = "0101"  then
680       men1 <= angka5 ;
681       elsif men0 = "0110"  then
682       men1 <= angka6 ;
683            elsif men0 = "0111"  then
684       men1 <= angka7 ;
685       elsif men0 = "1000"  then
686       men1 <= angka8 ;
687            elsif men0 = "1001"  then
688       men1 <= angka9 ;
689       elsif men0 = "1010"  then
690       men0 <= "0000" ;
691       end if ;
692
693    end if; -- CLK'event and CLK = '1'
694    End Process;
```

Gambar 4.20 *Counter* menit digit ke tiga

4.2.2.12 *ROM*

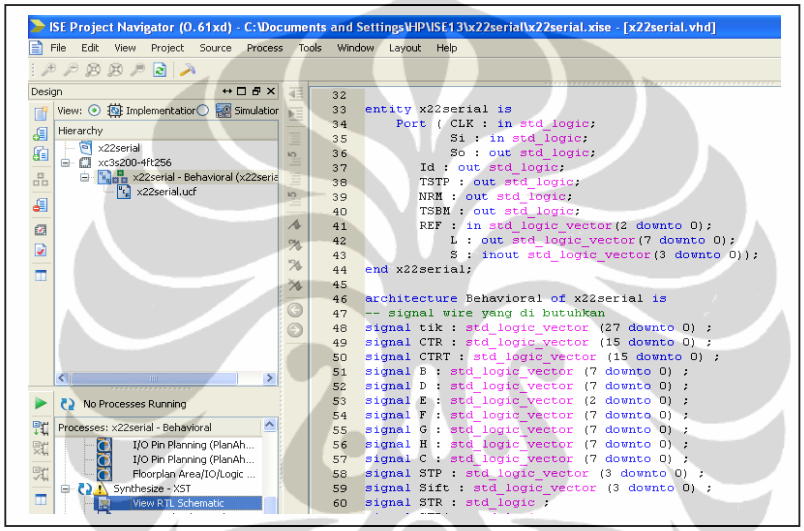
ROM merupakan memori yang hanya bisa dibaca, pada kode *VHDL ROM* digunakan sebagai penyimpan karakter angka yang dapat dipanggil untuk keperluan *time clock generator*. Kode *VHDL ROM* karekter angka untuk format *seven segment* ditunjukan pada Gambar 4.21

```
696   ---  ROM ( Read Only Memory ) Karakter Number
697       angka1 <= "01111001" ;
698       angka2 <= "00100100" ;
699       angka3 <= "00110000" ;
700       angka4 <= "00011001" ;
701       angka5 <= "00010010" ;
702       angka6 <= "00000010" ;
703       angka7 <= "01111000" ;
704       angka8 <= "00000000" ;
705       angka9 <= "00010000" ;
706       angka0 <= "01000000" ;
707   end Behavioral;
```

Gambar 4.21 *ROM* karakter angka format *seven segment*

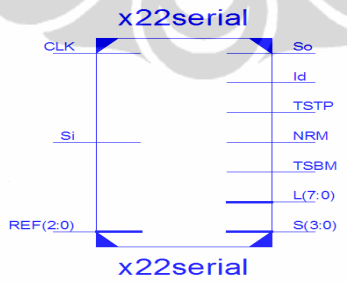
4.2.3 *RTL*

Register Transfer Language (*RTL*) merupakan fitur pada *ISE* 6.3i yang digunakan untuk mengubah *VHDL code* menjadi *schematic diagram*. Dengan menggunakan *RTL* memungkinkan kita mendapatkan *schematic diagram* dari kode *VHDL* dari sistem *CPU OBU* yang kita buat di *sub* bab sebelumnya. Cara nya adalah dengan mengklik *RTL* pada proses *window* seperti terlihat pada gambar berikut ini:



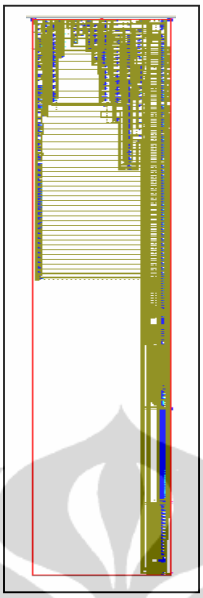
Gambar 4.22 *View RTL Schematic*

Setelah beberapa saat akan muncul window *schematic* dan *RTL top level* berupa *simbul box system* dengan port i/o sebagai *entity* seperti berikut.



Gambar 4.23 *Top level schematic CPU OBU*

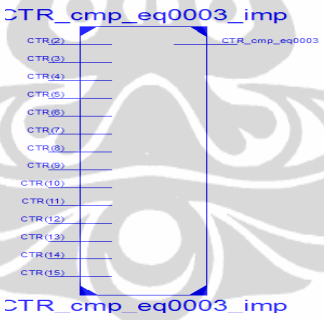
Top *level schematic* dapat di*generate* ke *level* di bawahnya yaitu rangkaian *schematic* yang masih merupakan blok modul seperti terlihat pada Gambar 4.24 berikut ini.



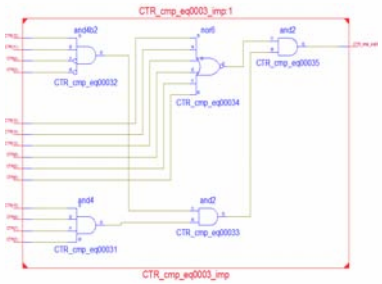
Gambar 4.24 RTL *Schematic* dari *system* CPU OBU

Dari setiap blok modul masih dapat di *generate* lagi ke rangkaian *schematic* seperti pembahasan beberapa bagian blok modul dari CPU OBU sebagai berikut (untuk selengkapnya bisa di lihat di dalam lampiran).

4.2.3.1 *CTR*

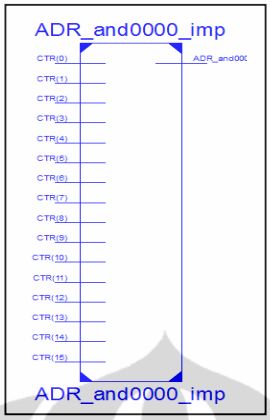


Gambar 4.25 Blok modul *CTR*

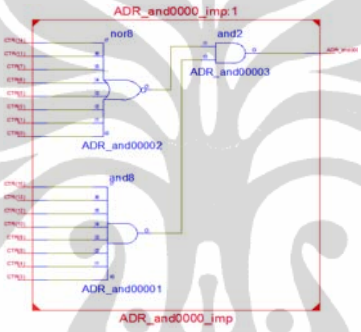


Gambar 4.26 *Schematic* dari modul *CTR*

4.2.3.2 *ADR*

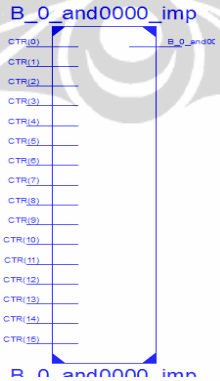


Gambar 4.27 Blok modul *ADR*

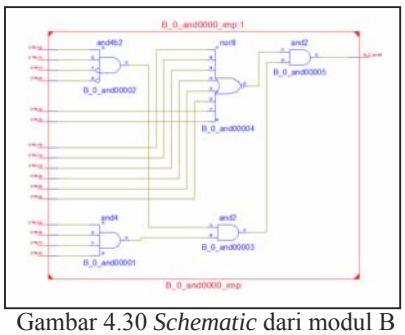


Gambar 4.28 *Schematic* dari modul *ADR*

4.2.3.3 B

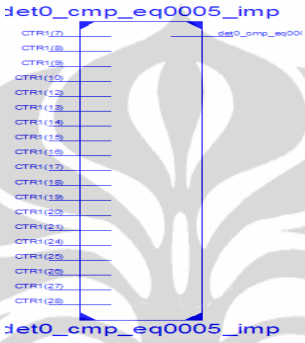


Gambar 4.29 Blok modul B

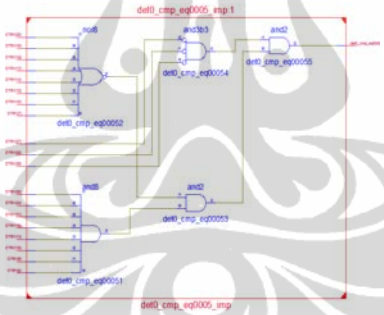


Gambar 4.30 *Schematic* dari modul B

4.2.3.4 det0



Gambar 4.31 Blok modul det0

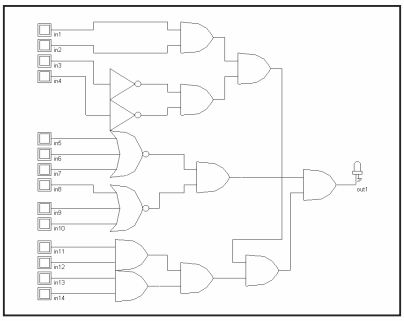


Gambar 4.32 *Schematic* dari modul det0

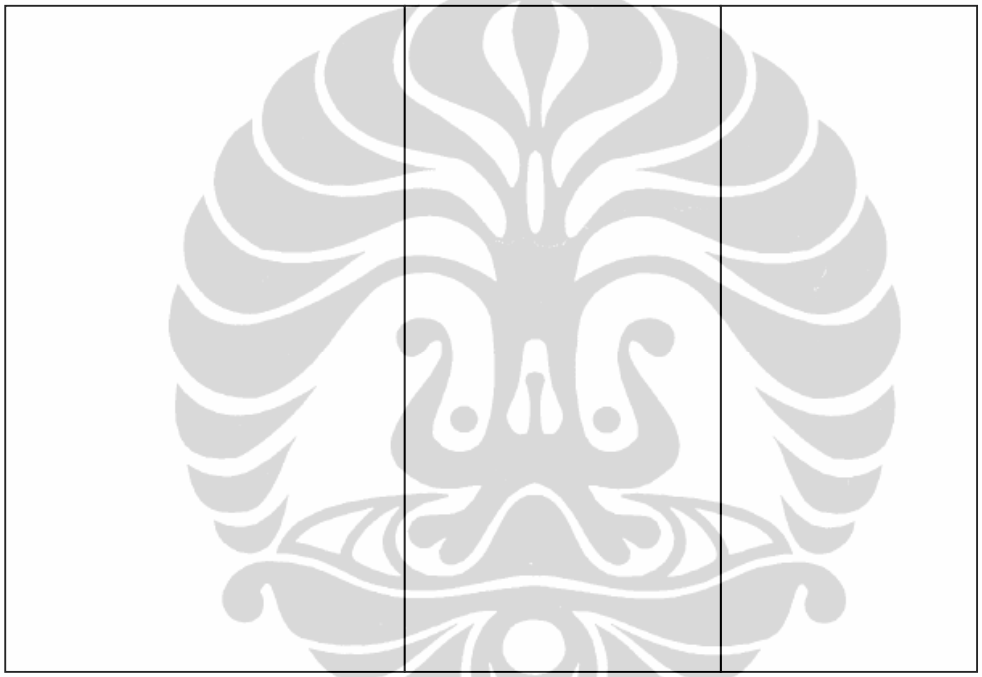
4.2.4 *Make verilog file* dari *schematic* dengan *DSCH2*

Membuat file verilog dari *schematic* rangkaian dapat dilakukan dengan menggunakan program *DSCH2*. Langkah membuat *verilog file* sederhana yaitu pertama-tama kita desain dahulu *schematic* yang ingin kita peroleh file verilognya. Kemudian setelah *schematic* yang kita buat selesai klik file *make* verilog file. Berikut ini proses memperoleh file verilog dari beberapa bagian blok modul yang ada di *CPU OBU*, untuk selengkapnya dapat dilihat di lampiran.

4.2.4.1 *Make verilog file schematic CTR*

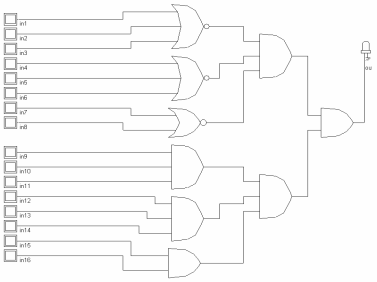


Gambar 4.33 *Schematic CTR* dengan menggunakan *DSCH2*



Gambar 4.34 *Make verilog CTR* dengan menggunakan *DSCH2*

4.2.4.2 *Make verilog file schematic ADR*

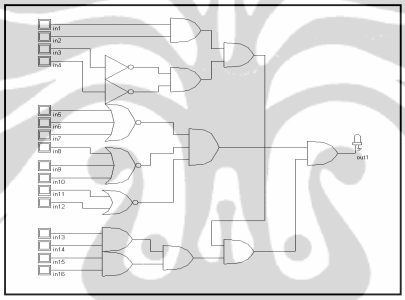


Gambar 4.35 *Schematic* ADR dengan menggunakan DSCH2

```
// DSCH 2.7f
// 6/5/2012 8:20:11 PM
//
C:\Thesis_Riyanto\Perbaikan1\RTL\images\
ADR_and0000_imp_1.sch
module ADR_and0000_imp_1(
in8,in16,in7,in15,in14,in13,in6,in5,
 in4,in1,in3,in2,in12,in11,in10,in9,
 out1);
 input in8,in16,in7,in15,in14,in13,in6,in5;
 input in4,in1,in3,in2,in12,in11,in10,in9;
 output out1;
 and #(16) and(w4,in16,in15);
 and #(16) and(w8,w6,w7,w4);
 and #(16) and(out1,w8,w22);
 and #(16) and(w6,in9,in10,in11);
and #(16) and(w7,in12,in13,in14);
 nor #(16) or(w23,in7,in8);
 nor #(13) or(w24,in1,in2,in3);
nor #(13) or(w25,in4,in5,in6);
and#(16)
and(w22,w24,w25,w23);
endmodule
// Simulation parameters in
Verilog Format
always
#1000 in8=~in8;
#2000 in16=~in16;
#4000 in7=~in7;
#8000 in15=~in15;
#16000 in14=~in14;
#32000 in13=~in13;
#64000 in6=~in6;
#128000 in5=~in5;
#256000 in4=~in4;
#512000 in11=~in11;
#512000 in1=~in1;
#1024000 in3=~in3;
#2048000 in2=~in2;
#4096000 in12=~in12;
#8192000 in11=~in11;
#16384000 in10=~in10;
#32768000 in9=~in9;
// Simulation parameters
// in8 CLK 10 10
// in16 CLK 20 20
// in7 CLK 40 40
// in15 CLK 80 80
// in14 CLK 160 160
// in13 CLK 320 320
// in6 CLK 640 640
// in5 CLK 1280 1280
// in4 CLK 2560 2560
// in1 CLK 5120 5120
// in3 CLK 10240 10240
// in2 CLK 20480 20480
// in12 CLK 40960 40960
// in11 CLK 81920 81920
// in10 CLK 163840 163840
// in9 CLK 327680 327680
```

Gambar 4.36 *Make verilog* ADR dengan menggunakan DSCH2

4.2.4.3 *Make verilog file schematic* B

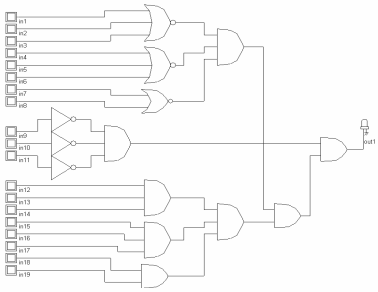


Gambar 4.37 *Schematic* B dengan menggunakan DSCH2

```
// DSCH 2.7f
// 6/6/2012 9:28:09 AM
//
C:\Thesis_Riyanto\Perbaikan1\RTL\images\
B_0_and0000_imp_1.sch
module B_0_and0000_imp_1(
in4,in13,in14,in15,in1,in3,in2,in16,
 in5,in6,in7,in8,in12,in11,in10,in9,
 out1);
 input in4,in13,in14,in15,in1,in3,in2,in16;
 input in5,in6,in7,in8,in12,in11,in10,in9;
 output out1;
 not #(10) inv(w2,in4);
 not #(10) inv(w4,in3);
 and #(16) and(w7,in16,in15);
 and #(16) and(w11,in2,in1);
 nor #(13) or(w15,in5,in6,in7);
 and #(16) and(w17,w2,w4);
 and #(16) and(w18,in14,in13);
and #(16) and(w19,w7,w18);
and #(16) and(out1,w20,w21);
and #(16) and(w20,w19,w23);
 nor #(13) or(w27,in8,in9,in10);
 nor #(16) or(w30,in11,in12);
 and #(16)
and(w21,w15,w27,w30);
 and #(16) and(w23,w17,w11);
endmodule
// Simulation parameters in
Verilog Format
always
#1000 in4=~in4;
#2000 in13=~in13;
#4000 in14=~in14;
#8000 in15=~in15;
#16000 in1=~in1;
#32000 in3=~in3;
#64000 in2=~in2;
#128000 in16=~in16;
#256000 in5=~in5;
#512000 in6=~in6;
#1024000 in7=~in7;
#2048000 in8=~in8;
#4096000 in12=~in12;
#8192000 in11=~in11;
#16384000 in10=~in10;
#32768000 in9=~in9;
// Simulation parameters
// in4 CLK 10 10
// in13 CLK 20 20
// in14 CLK 40 40
// in15 CLK 80 80
// in1 CLK 160 160
// in3 CLK 320 320
// in2 CLK 640 640
// in16 CLK 1280 1280
// in5 CLK 2560 2560
// in6 CLK 5120 5120
// in7 CLK 10240 10240
// in8 CLK 20480 20480
// in12 CLK 40960 40960
// in11 CLK 81920 81920
// in10 CLK 163840 163840
// in9 CLK 327680 327680
```

Gambar 4.38 *Make verilog* B dengan menggunakan DSCH2

4.2.4.4 *Make verilog file schematic* det0



Gambar 4.39 *Schematic* det0 dengan menggunakan DSCH2

```
// DSCH 2.7f
// 6/6/2012 10:25:07 PM
//
C:\Thesis_Riyanto\Perbaikan1\RTL\images\
det0_cmp_eq0005_imp_1.sch
module det0_cmp_eq0005_imp_1(
in8,in19,in18,in7,in9,in10,in11,in17,
 in6,in5,in4,in1,in3,in2,in16,in15,
 in14,in13,in12,out1);
input in8,in19,in18,in7,in9,in10,in11,in17;
input in6,in5,in4,in1,in3,in2,in16,in15;
input in14,in13,in12;
output out1;
not =(10) inv(w18,in11);
not =(10) inv(w19,in10);
and =(16) and(out1,w20,w21);
not =(10) inv(w22,in9);
nor =(16) or(w24,in7,in8);
nor =(13) or(w25,in1,in2,in3);
nor =(13) or(w26,in4,in5,in6);
and =(16) and(w27,w25,w26,w24);
and =(16) and(w28,in15,in16,in17);
and =(16) and(w31,in12,in13,in14);
and =(16) and(w33,w31,w28,w32);
and =(16) and(w32,in19,in18);
and =(16) and(w20,w33,w27);
 and                =(16)
and(w21,w22,w19,w18);
endmodule
//    Simulation    parameters    in
Verilog Format
always
=1000 in8=~in8;
=2000 in19=~in19;
=4000 in18=~in18;
=8000 in7=~in7;
=16000 in9=~in9;
=32000 in10=~in10;
=64000 in11=~in11;
=128000 in17=~in17;
=256000 in6=~in6;
=512000 in5=~in5;
=1024000 in4=~in4;
=2048000 in1=~in1;
=4096000 in3=~in3;
=8192000 in2=~in2;
=16384000 in16=~in16;
=32768000 in15=~in15;
=65536000 in14=~in14;
=131072000 in13=~in13;
=262144000 in12=~in12;
// Simulation parameters
// in8 CLK 10 10
// in19 CLK 20 20
// in18 CLK 40 40
// in7 CLK 80 80
// in9 CLK 160 160
// in10 CLK 320 320
// in11 CLK 640 640
// in17 CLK 1280 1280
// in6 CLK 2560 2560
// in5 CLK 5120 5120
// in4 CLK 10240 10240
// in1 CLK 20480 20480
// in3 CLK 40960 40960
// in2 CLK 81920 81920
// in16 CLK 163840 163840
// in15 CLK 327680 327680
// in14 CLK 655360 655360
//    in13    CLK    1310720
1310720
//    in12    CLK    2621440
2621440
```

Gambar 4.40 *Make verilog* det0 dengan menggunakan DSCH2

4.2.5 *Compile verilog file* dengan *Microwind*

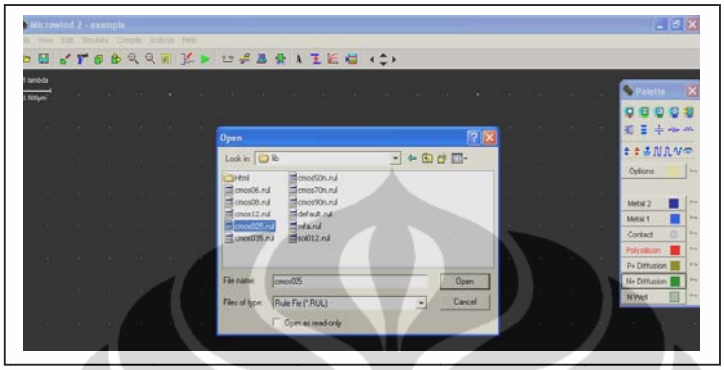
Untuk mendapatkan *CMOS layout* yang perlu dilakukan adalah meng-*compile verilog file* yang sudah dibuat dengan menggunakan program *Microwind*. Untuk menentukan ukuran desain teknologi *VLSI*, di dalam program *Microwind* menyediakan *template rule VLSI* dengan ukuran dalam *micro* teknologi. *Tamplate rul* disetting atau dipanggil pertama kali dengan cara klik file *select foundry* pada program *Microwind*, kemudian pilih *file rul* yang sudah tersedia. Berikut tabel yang menjelaskan ukuran dari *rul* tersebut.

Lithography	*Year*	*Metal layers*	*VDD supply (V)*	*Oxide (nm)*	*Threshold voltage (V)*	*Input/output pads*	*Microwind2 rule file*
1.2µm	1986	2	5.0	25	0.8	250	Cmos12.rul
0.7µm	1988	2	5.0	20	0.7	350	Cmos08.rul
0.5µm	1992	3	3.3	12	0.6	600	Cmos06.rul
0.35µm	1994	5	3.3	7	0.5	800	Cmos035.rul
0.25µm	1996	6	2.5	6	0.45	1000	Cmos025.rul

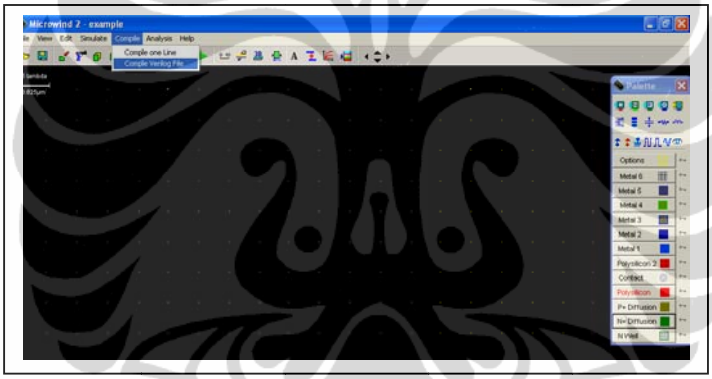
Tabel 4.2 *Rule template* pada *microwind*

Dibawah ini proses memperoleh perancangan *VLSI* 0.25µm dari beberapa bagian blok modul yang ada di *CPU OBU*, sedangkan untuk selengkapnya dapat dilihat di lampiran

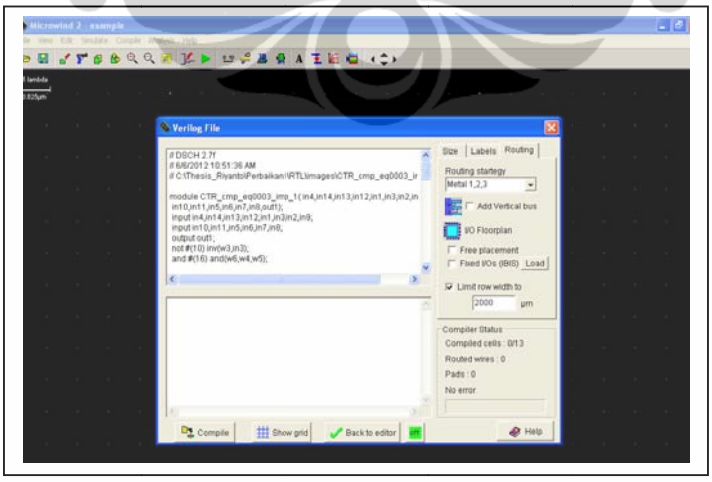
4.2.5.1 Perancangan *VLSI* 0.25µm untuk rangkaian CTR



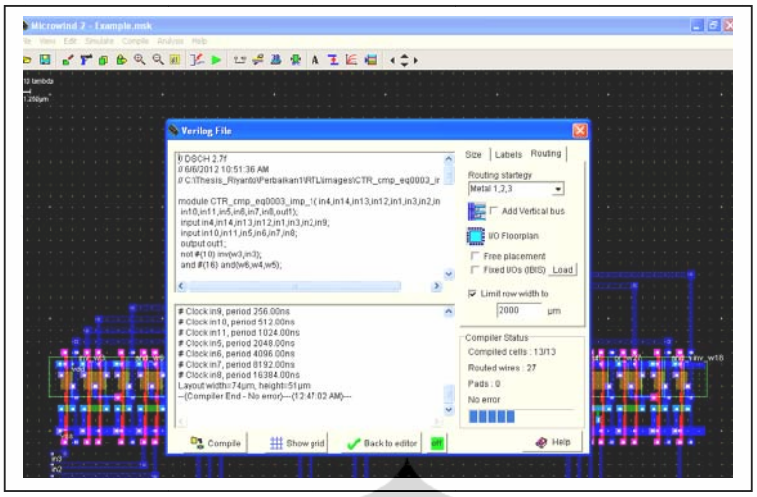
Gambar 4.41 *Select Foundry CMOS*025.rul



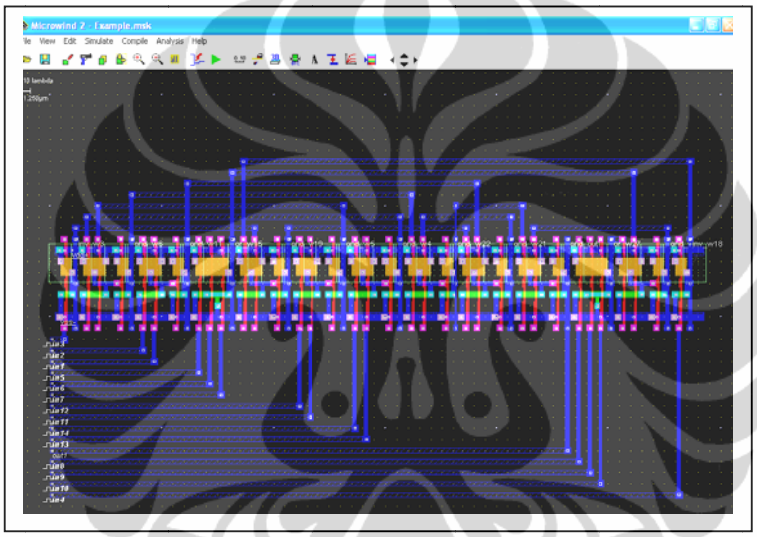
Gambar 4.42 *Compile Verilog file*



Gambar 4.43 Pilih *file verilog* CTR_cmp_eq0003_imp_1.txt

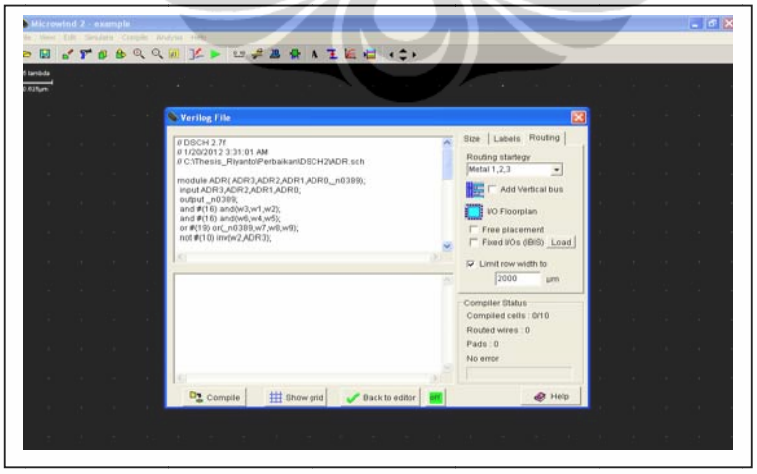


Gambar 4.44 Klik *Compile* CTR_cmp_eq0003_imp_1.txt

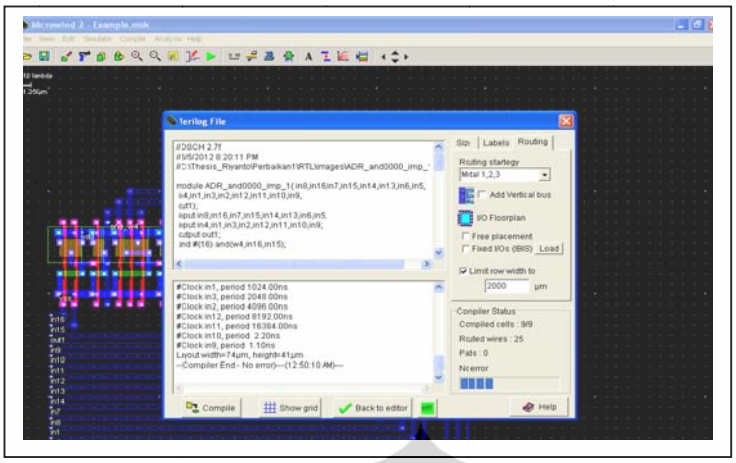


Gambar 4.45 Hasil akhir *CMOS layout* dari CTR

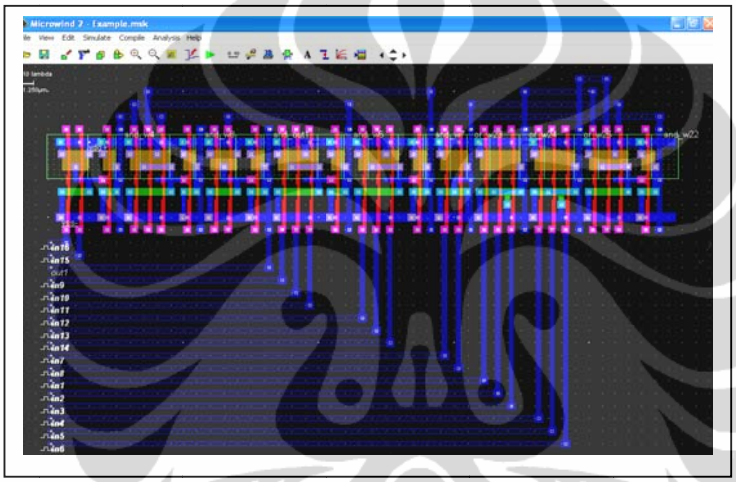
4.2.5.2 Perancangan *VLSI* 0.25µm untuk rangkaian ADR



Gambar 4.46 Pilih *file verilog* ADR_and0000_imp_1.txt

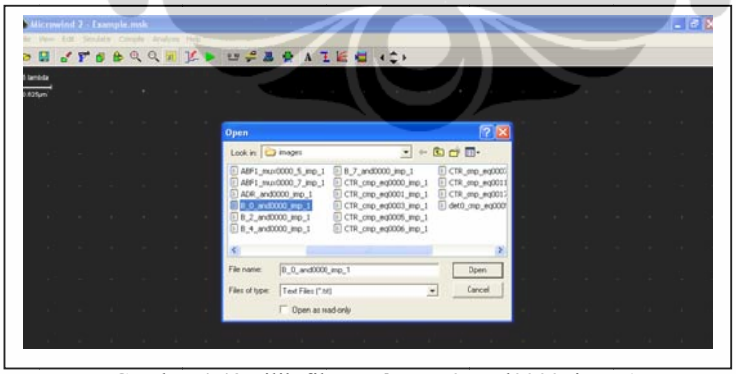


Gambar 4.47 Klik *Compile* ADR_and0000_imp_1.txt

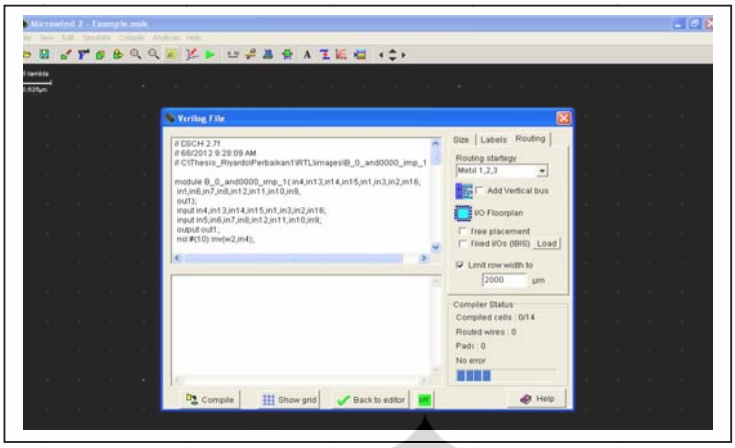


Gambar 4.48 Hasil akhir *CMOS layout* dari ADR

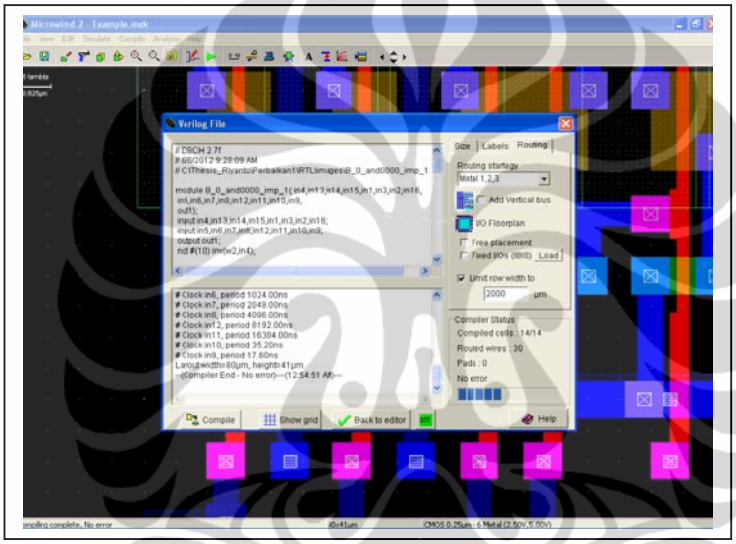
4.2.5.3 Perancangan *VLSI* 0.25µm untuk rangkaian B



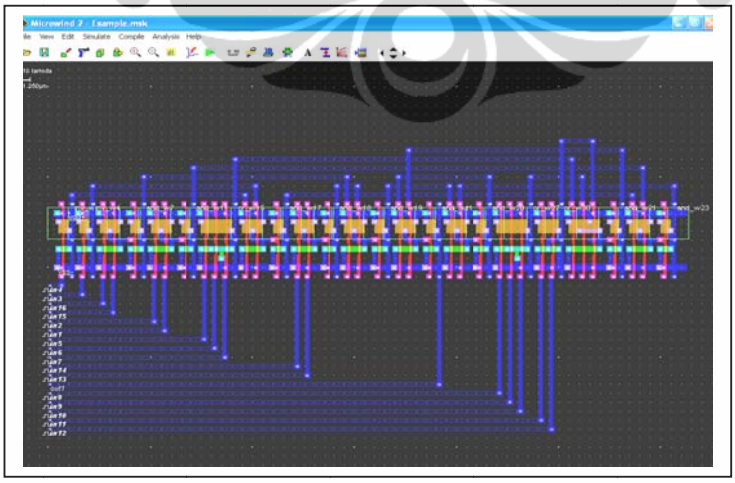
Gambar 4.49 Pilih file *verilog* B_0_and0000_imp_1.txt



Gambar 4.50 Window setting konversi verilog ke CMOS layout

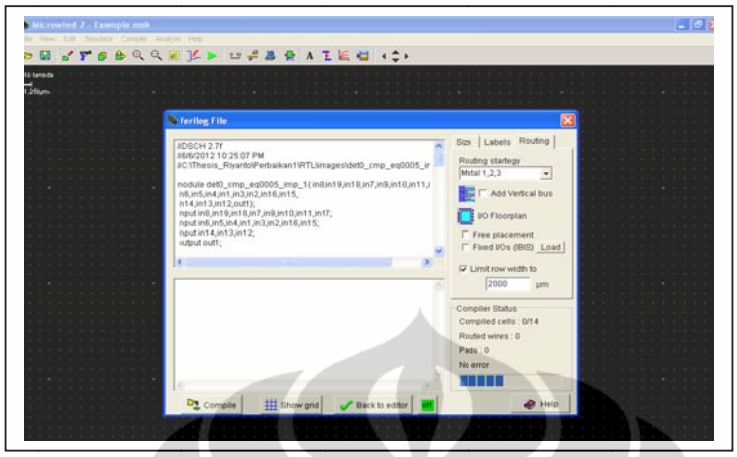


Gambar 4.51 Klik *Compile* B_0_and0000_imp_1.txt

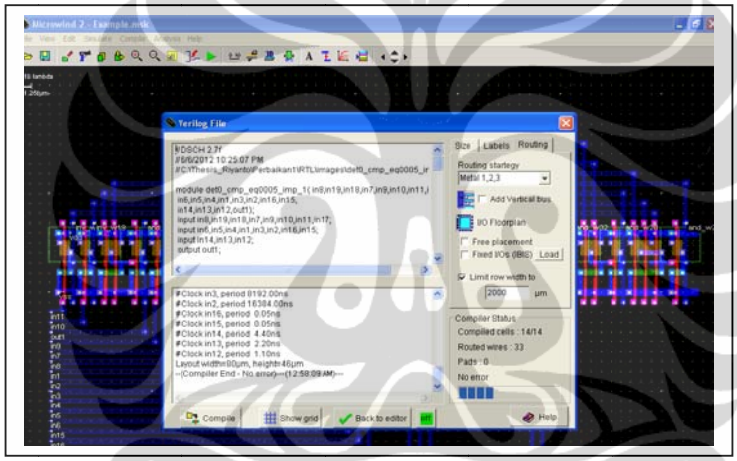


Gambar 4.52 Hasil akhir *CMOS layout* dari B

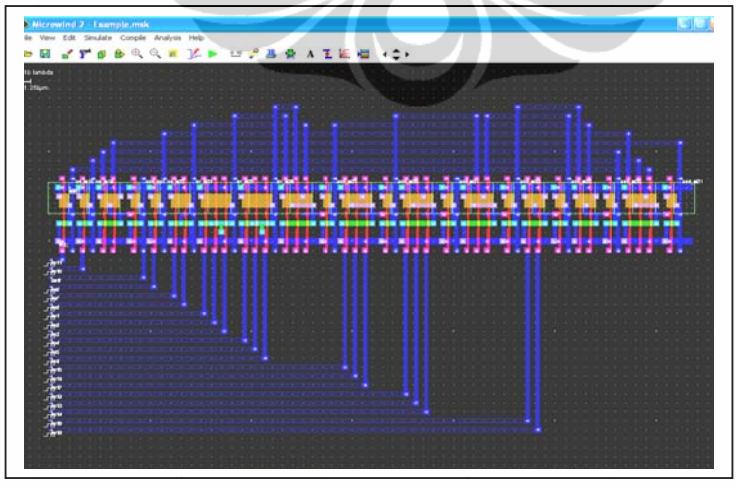
4.2.5.4 Perancangan *VLSI* 0.25µm untuk rangkaian det0



Gambar 4.53 Pilih *file verilog* det0_cmp_eq0005_imp_1.txt



Gambar 4.54 Klik *Compile* det0_cmp_eq0005_imp_1.txt

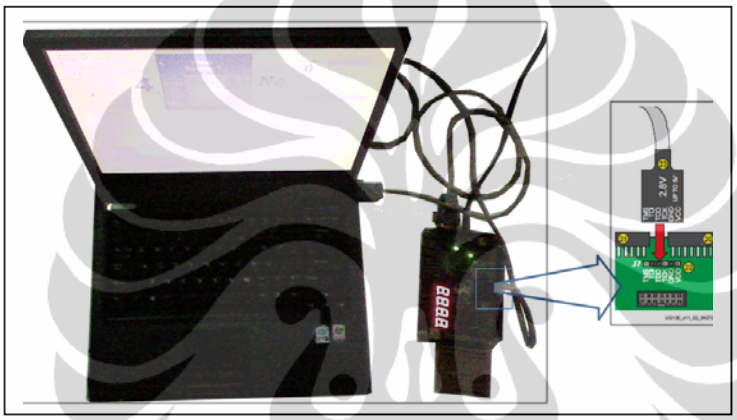


Gambar 4.55 Hasil akhir *CMOS layout* det0_cmp_eq0005_imp_1.txt

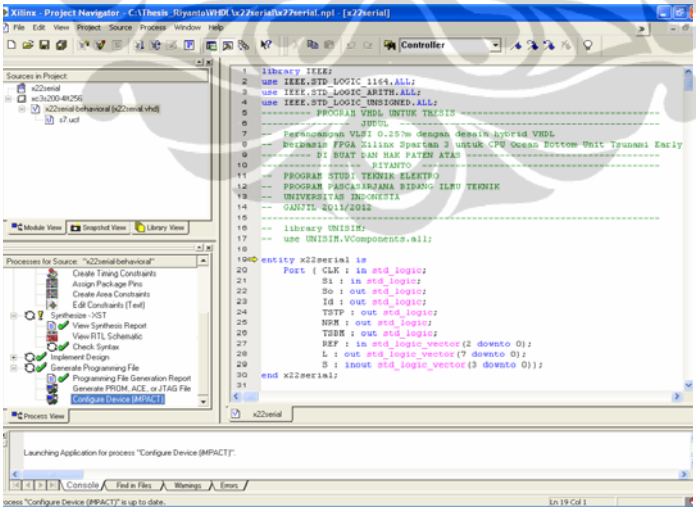
4.3 Sistem tertanam pada Xilinx Spartan 3

4.3.1 *Configure Device (iMPACT)*

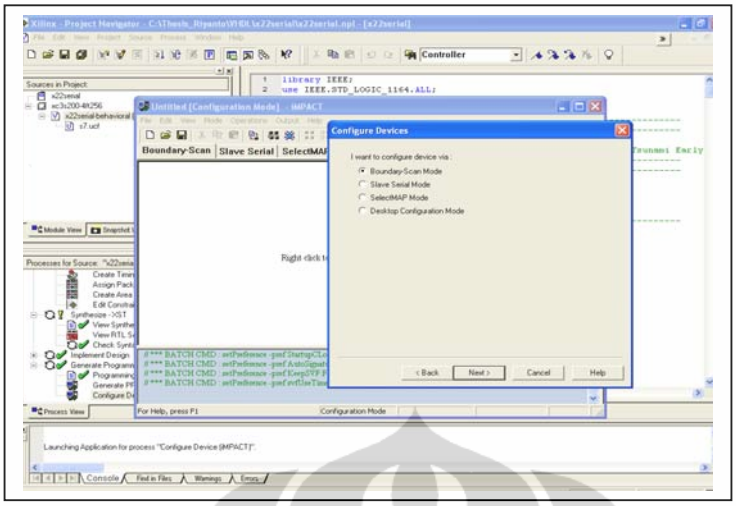
Configure Device (iMPACT) merupakan menu fitur dari ISE 6.3i yang berada pada proses *window*. *Menu* (iMPACT) digunakan untuk *upload* program *VHDL CPU OBU* yang sudah dibuat sebelumnya diupload kedalam *chip* XC3S200 yang berada pada *board Xilinx Spartan* 4. Langkah *Configure Device (iMPACT)* cukup panjang dan perlu ketelitian proses langkah demi langkahnya. Berikut proses *Configure Device* (iMPACT). Pertama rangkai pengkabelan *programming Xilinx Spartan* 3 seperti terlihat pada Gambar 4.56.



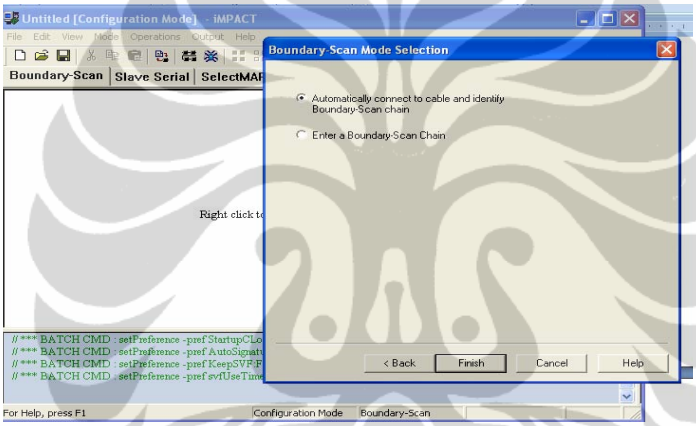
Gambar 4.56 Pengkabelan *Programming Xilinx Spartan* 3



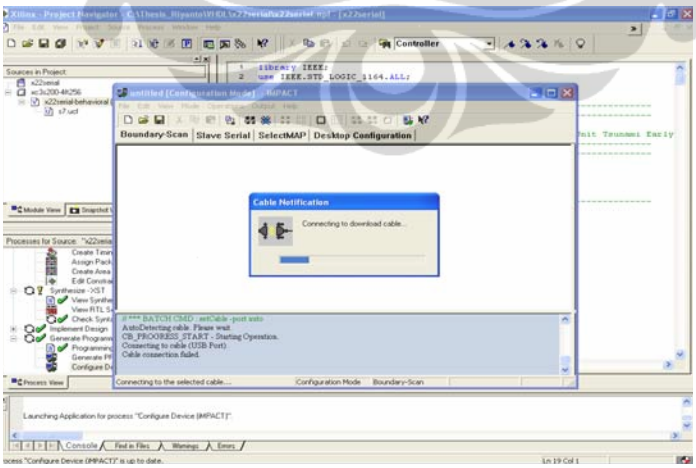
Gambar 4.57 Klik *Configure Device* (iMPACT) pada proses *window*



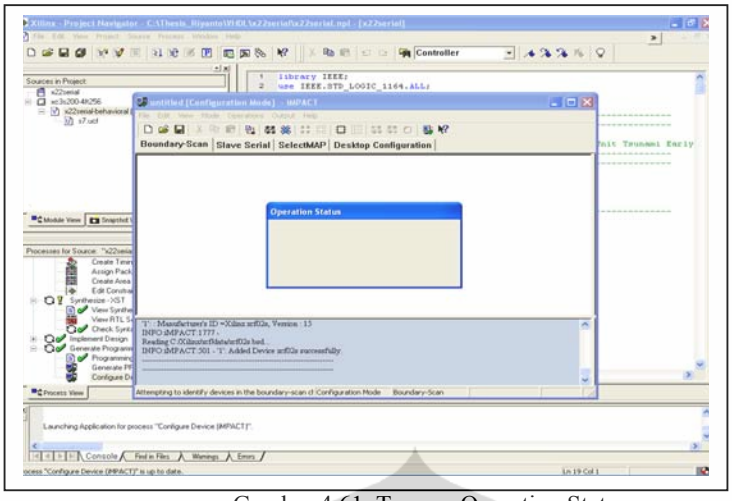
Gambar 4.58 Pilih *Boundary-Scan Mode* kemudian klik *next*



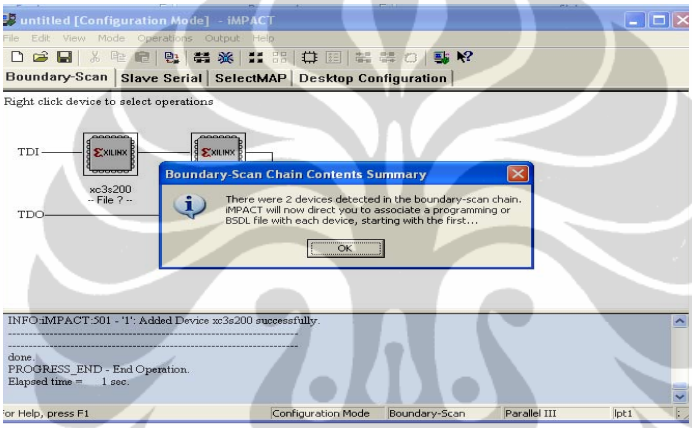
Gambar 4.59 Pilih *Automatically* kemudian klik *Finish*



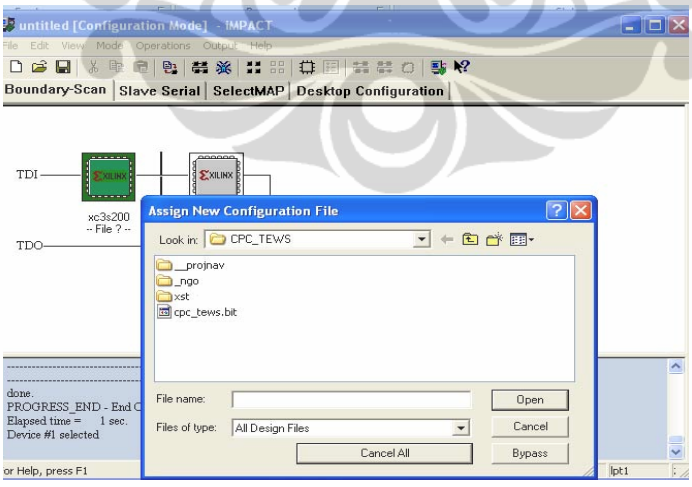
Gambar 4.60 Tunggu proses *connecting*



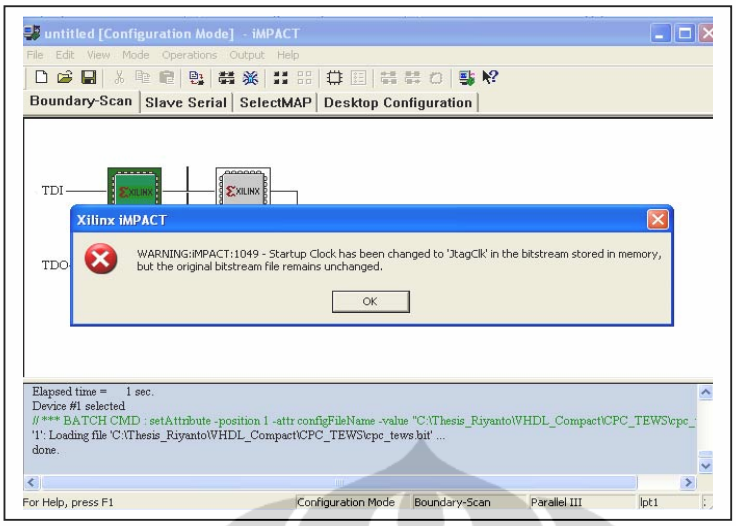
Gambar 4.61 Tunggu Operation Status



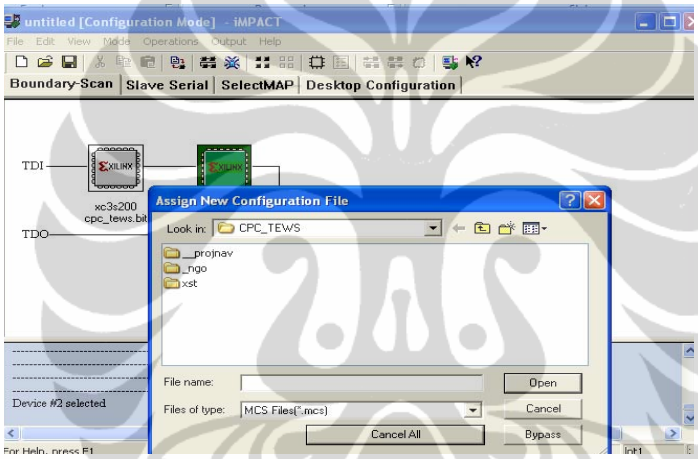
Gambar 4.62 Setelah muncul *window Boundary Scan Chain*, klik OK



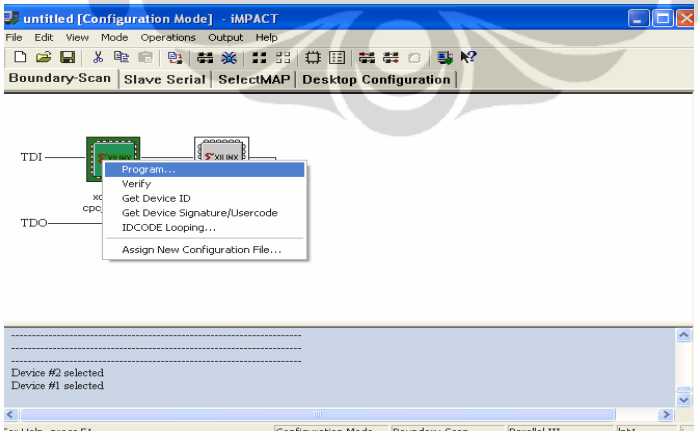
Gambar 4.63 Setelah muncul *window Configuration file*, pilih *file*, klik *open*



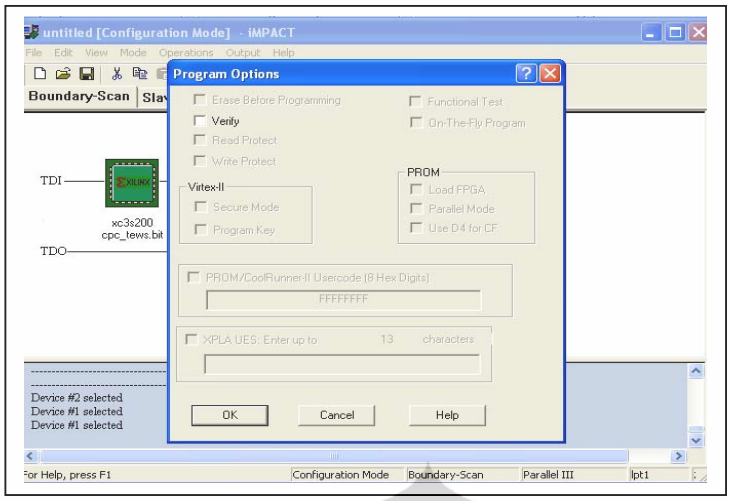
Gambar 4.64 Setelah muncul *window Xilinx iMPACT*, klik OK



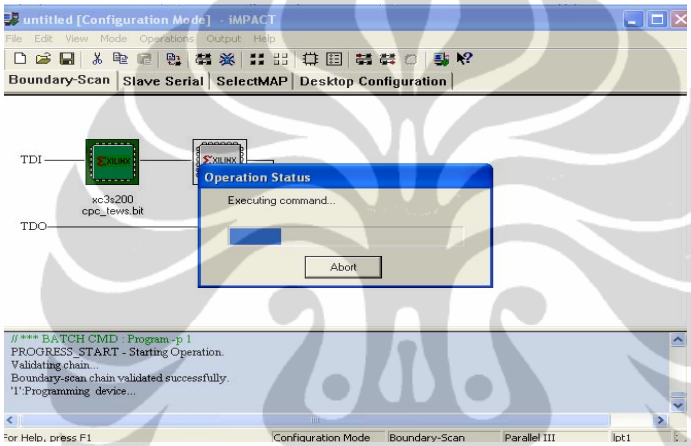
Gambar 4.65 Setelah muncul *window Configuration file*, klik *Bypass*



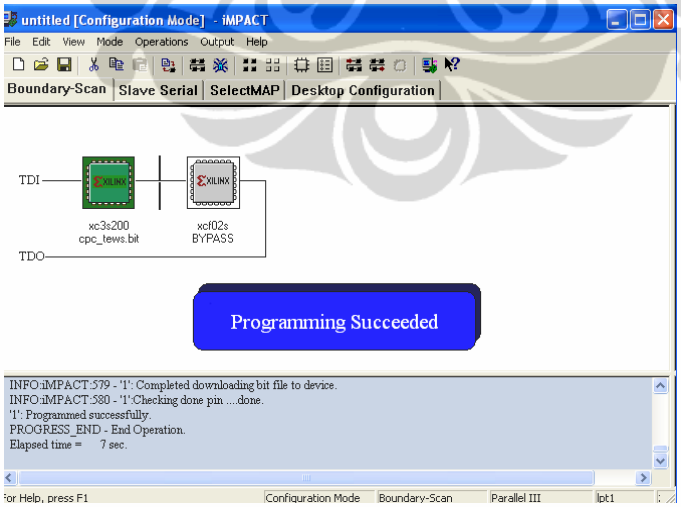
Gambar 4.66 Klik kanan XC3S200, klik Program



Gambar 4.67 Setelah muncul program option klik OK



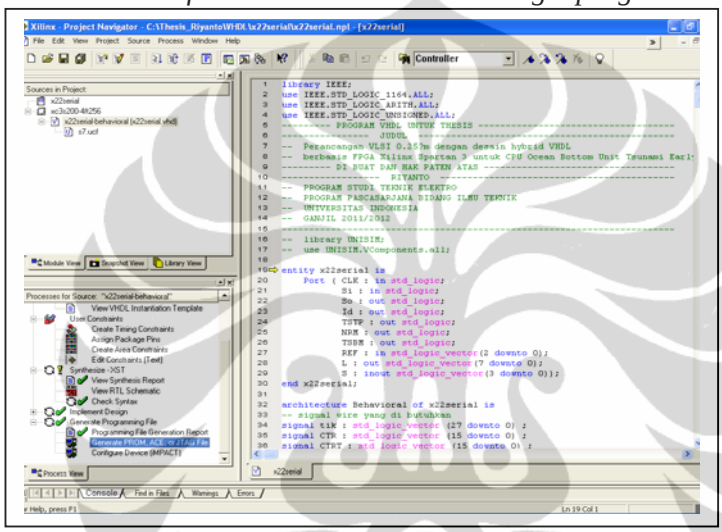
Gambar 4.68 Tunggu proses *programming*



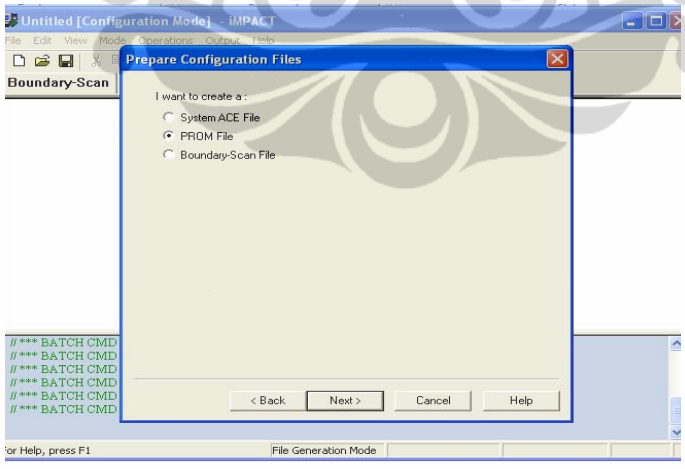
Gambar 4.69 *Programming iMPACT Succeeded*

4.3.2 *Generate PROM File*

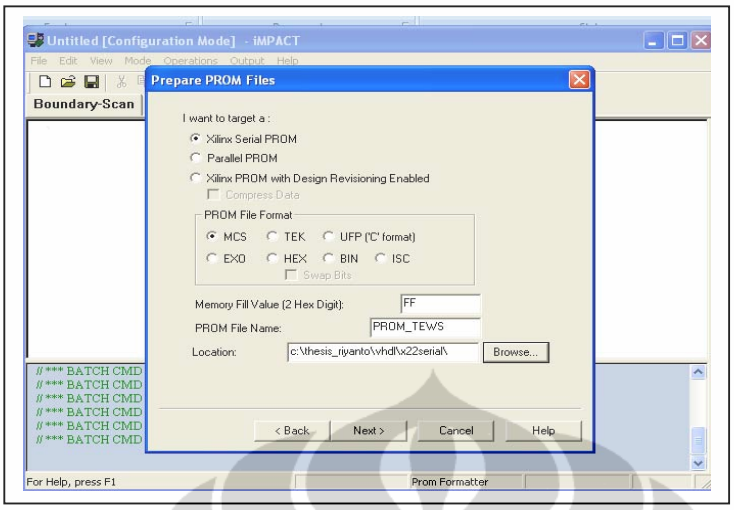
Proses *Configure Device* (iMPACT) seperti sub bab 4.3 adalah melakukan *upload* program *VHDL* dalam *chip* XC3S200, dimana *chip* ini menyimpan program bersifat sementara, jika *power supplay* dimatikan maka program yang sudah kita *upload* akan hilang. Untuk itu diperlukan *PROM target programming* yaitu XCF02S. Metode ini dilakukan dengan cara *generate PROM* file yang merupakan salah satu fitur *ISE* 6.3i yang ada di proses window. Berikut langkah proses *generate PROM file CPU OBU* ke *PROM target programming* XCF02S.



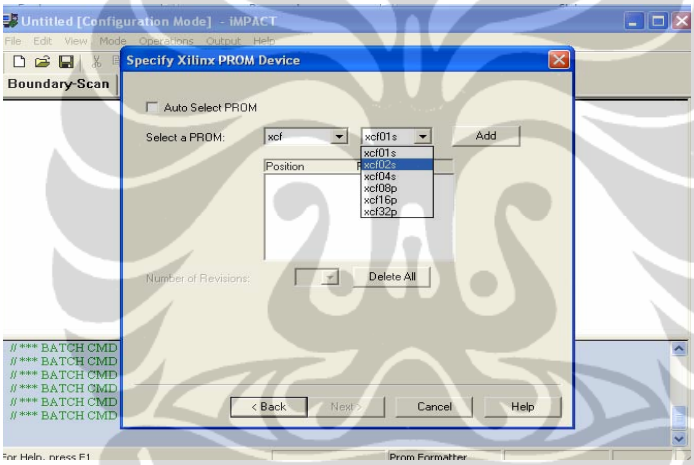
Gambar 4.70 *Generate PROM*



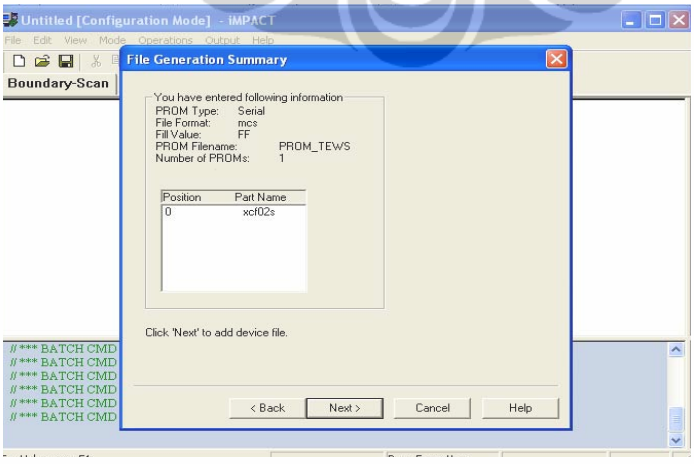
Gambar 4.71 *Prepare Configuration*, klik *next*



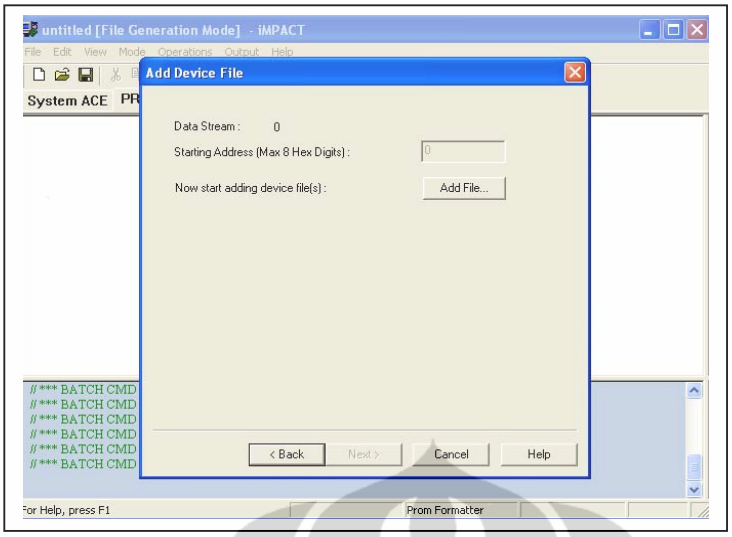
Gambar 4.72 *Prepare PROM*, ganti *PROM file name*, klik *next*



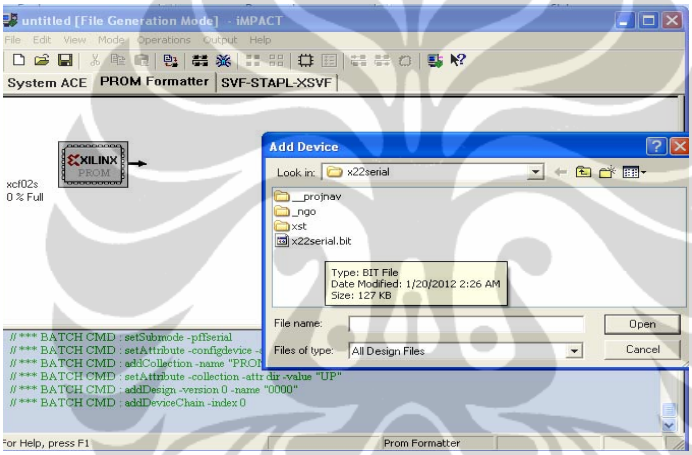
Gambar 4.73 *Specify Xilinx, Select PROM xcf, xcf02s,* klik *add, next*



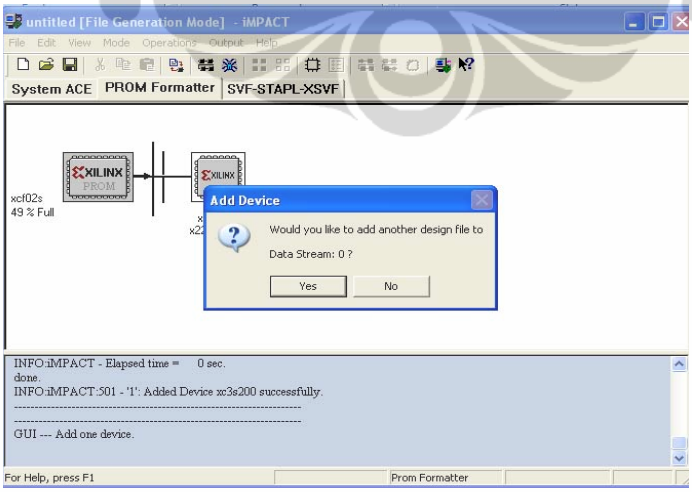
Gambar 4.74 *File generation*, klik *next*



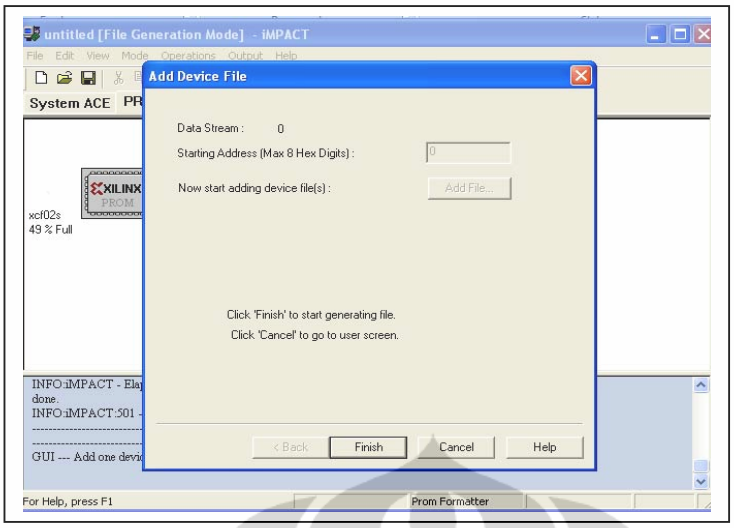
Gambar 4.75 *Add device*, *add file*



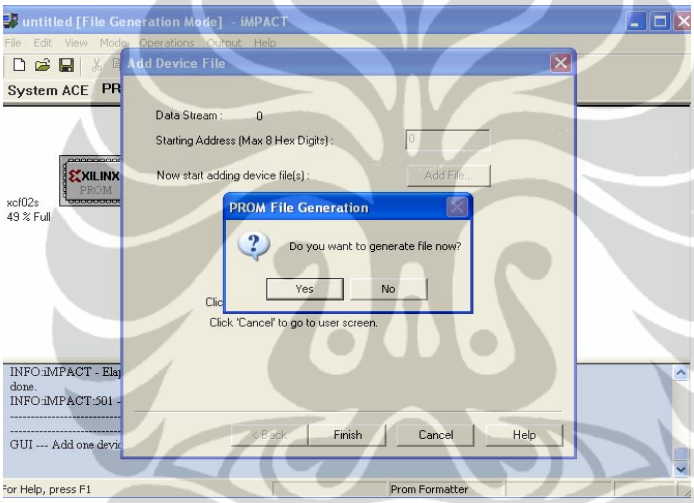
Gambar 4.76 *Add device*, pilih *file* *.*bit* kemudian *open*



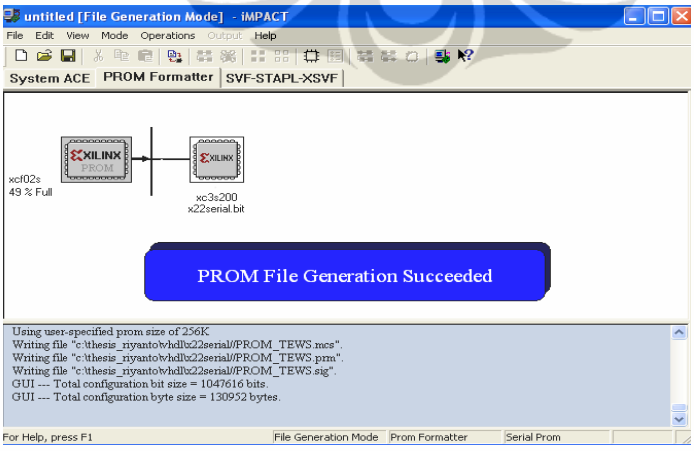
Gambar 4.77 *Add device*, klik *no*



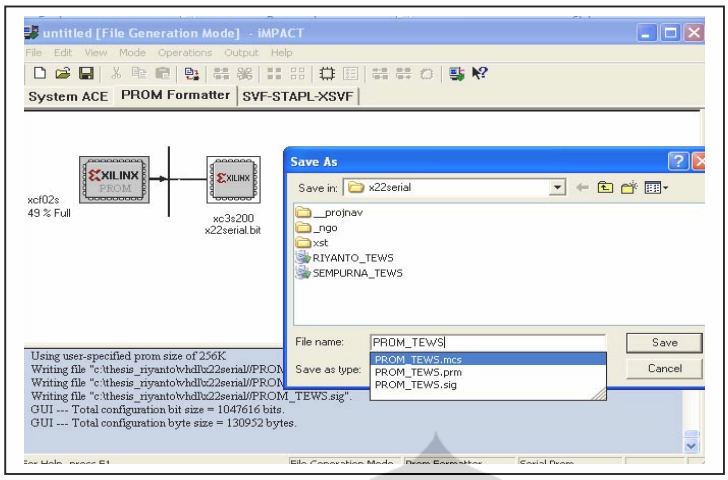
Gambar 4.78 *Add device*, klik *Finish*



Gambar 4.79 *PROM file*, *Generate now*, klik *yes*



Gambar 4.80 *PROM file File Generation Succeeded*



Gambar 4.81 *Save As* kemudian *close*

Langkah pembuatan *file PROM* sudah selesai untuk *upload* ke *PROM* maka ikuti langkah sub bab 4.3.1 *configure device (iMPACT)* persis sama, yang membedakan adalah pada Gambar 4.65 Setelah muncul *window Configuration file*, jangan klik *Bypass*, tetapi ambil file *.MCS sesuai file name *PROM* yang sudah dibuat dengan prosedur sebelumnya di atas. Setelah itu lakukan *programming PROM* , setelah *programming succeeded* maka program sudah tersimpan dalam *PROM Xilinx Spartan* 3.

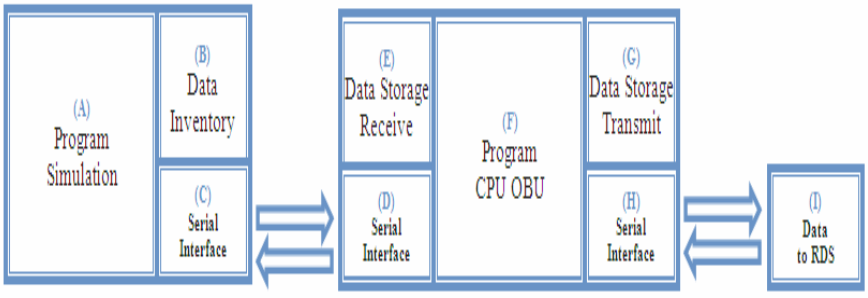
BAB V
INTEGRASI DAN DEMO SISTEM

5.1 Integrasi system

Integrasi system dilakukan dengan cara menghubungkan program simulasi yang di buat di dalam *PC* dengan program *VHDL* yang sudah dibuat di dalam *Xilinx Spartan* 3. Agar program simulasi dan program *CPU OBU* saling berhubungan secara umpan balik diperlukan kabel serial, kabel serial ini menjadi media komunikasi data melalui kedua serial *interface*. Gambar 5.1 berikut merupakan *integrasi system demo program* simulasi dan CPU OBU.

Gambar 5.1 Integrasi system demo CPU OBU

Untuk memperjelas gambaran integrasi system dapat dijelaskan melalui blok diagram komunikasi program simulasi dengan *CPU OBU* seperti terlihat pada Gambar 5.2.

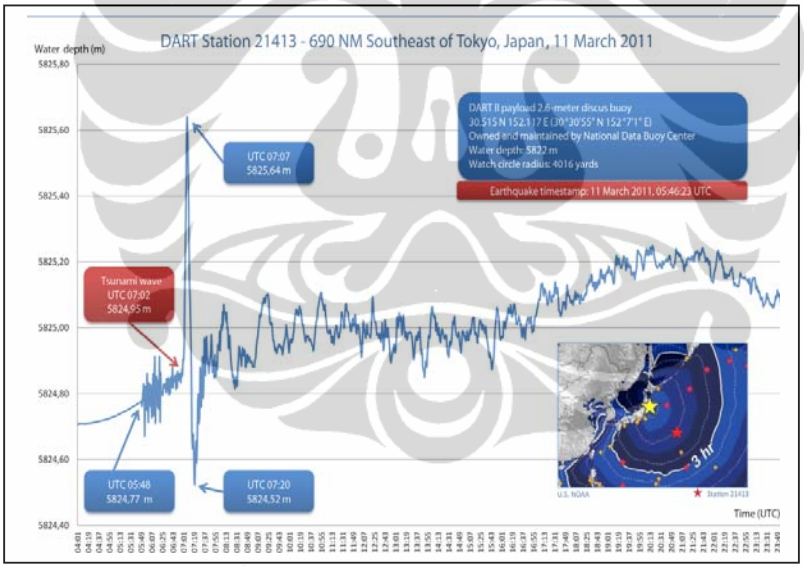


Gambar 5.2 Blok diagram komunikasi program simulasi dengan CPU OBU

Blok diagram komunikasi program simulasi dengan CPU OBU dibagi menjadi sembilan bagian utama yaitu :

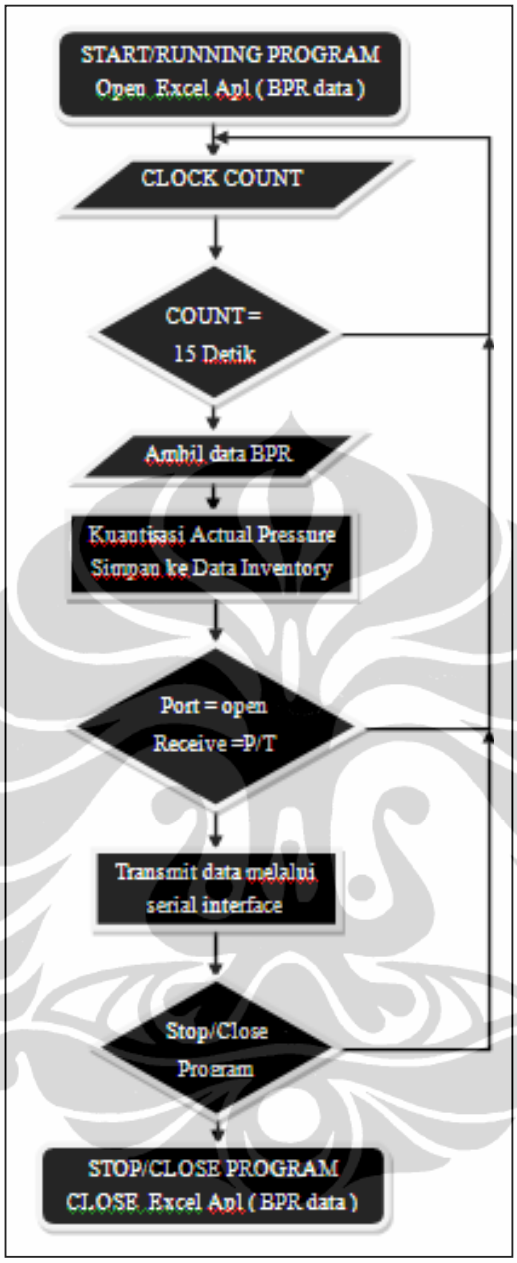
A. *Program simulation*
B. *Data Inventory*
C. *Serial Interface Transceiver (Computer)*
D. *Serial Interface Receiver (Xilinx Spartan)*
E. *Data Storage Receiver*
F. *Program CPU OBU*
G. *Data storage Transmitter*
H. *Serial Interface transmitter (Xilinx Spartan)*

Program simulasi merupakan program yang menggambarkan kondisi fenomena tekananan (*pressure*) atau kedalaman (*depth*) air laut. Program simulasi ini menggunakan data *BPR* yang sebenarnya, yang diambil pada kondisi momen saat terjadinya *tsunami* di Sendai Jepang tanggal 11-3-2011. Berikut grafik saat terjadi *tsunami* di Sendai Jepang.



Gambar 5.3 Grafik kedalaman air laut dalam meter terhadap waktu (id 21413)

Untuk mensimulasikan pemodelan kondisi seperti grafik Gambar 5.3 maka di perlukan program simulasi yang dibuat dalam computer. Program simulasi ini bekerja sesuai dengan diagram alir seperti terlihat pada Gambar 5.4.



Gambar 5.4 Flowchart diagram program simulasi fenomena kondisi laut

Cara kerja program simulasi ini pertama-tama pada saat program *start/running* maka program akan memanggil aplikasi *excel* dan membuka file *book1.xls*, dimana didalam *book1.xls* ini sudah terdapat data BPR sesuai format data standar. Data BPR ini disimpan secara urut dari mulai A3 sampai A4802. Data *BPR* ini merupakan representasi grafik kedalaman air laut dalam meter terhadap waktu (id 21413) seperti pada Gambar 5.4. setelah *excel* terbuka maka

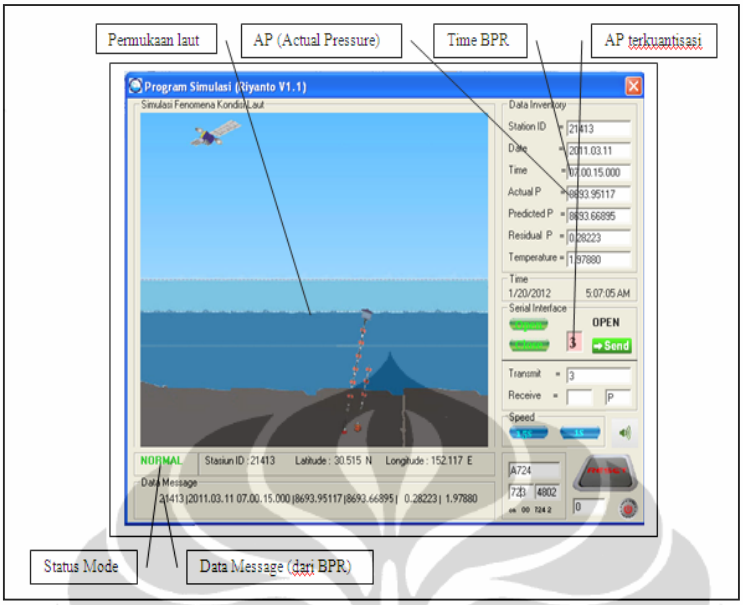
CLOCK *COUNT* aktif menghitung akumulasi naik. Jika *COUNT* < 15 menit *No Event,* setelah *COUNT* = 15 menit maka ambil data BPR yang terdapat di dokumen excel secara berurutan. Setelah itu *extract* data menjadi pecahan data-data informasi kemudian simpan pada data *inventory*, kemudian kuantisasi *actual pressure* menjadi data biner 8 bit seperti tabel 5.1 berikut.

Kuantisasi	Biner 8 bit	Range
1	00000001	5825.50 sampai 5825.70
2	00000010	5825.70 sampai 5825.90
3	00000011	5825.90 sampai 5825.10
4	00000100	5825.10 sampai 5825.30
5	00000101	5825.30 sampai 5825.50
6	00000110	5825.50 sampai 5825.70

Tabel 5.1 Kuantisasi actual pressure menjadi BCD 8 bit.

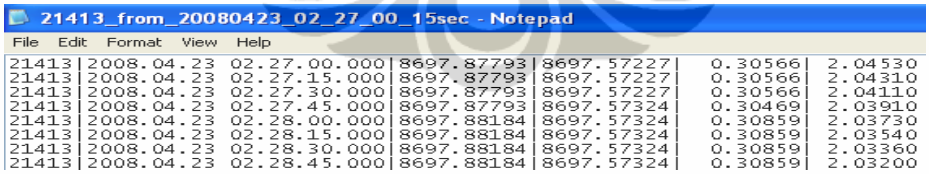
Setelah data *actual pressure* terkuantisasi masukkan data ini ke *AP* terkuantisasi. Apabila *port open* dan *serial receive interface* mendapat *command* 'P' atau 'T' dari *CPU OBU* maka *AP* akan dikirim ke *CPU OBU*. Program akan berjalan terus menerus, hanya akan berhenti jika program *stop/close*.

Program simulasi yang dibuat, terlihat seperti pada Gambar 5.5. bagian-bagian program simulasi ini antara lain gambar permukaan laut yang dapat berubah sesuai dengan perubahan data *BPR*, *Actual Pressure* (*AP*) data utama yang digunakan sebagai data deteksi *tsunami*, time *BPR* adalah waktu data *BPR* sebenarnya saat diambil, *Actual Pressure* terkuantisasi, *status mode*, dan *data message* (*BPR*). Selain itu ada beberapa fitur tambahan yang berguna untuk mempermudah penggunaan program simulasi seperti *button Open* dan *Close* di gunakan untuk membuka atau menutup *port serial* secara manual, *button speed* 15s dan 1s untuk merubah kecepatan cuplik data (15 detik atau 1 detik), *button reset* untuk mengembalikan semua nilai pada nilai awal. Dan masih ada fitur tambahan lain.

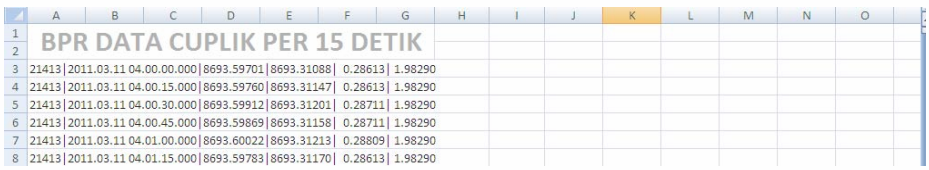


Gambar 5.5 Program simulasi pemodelan fenomena air laut saat tsunami

Data message (dari *BPR*) adalah diambil sesuai *speed* pencuplikan jika 15s berarti pengambilan data *message* dari dokumen excel dengan periode waktu 15 detik. *Data message* merupakan data pengukuran *BPR* yang sebenarnya data ini didapat dari sumber website resmi NOAA[19], bentuk data *BPR* dalam dokumen notepad, isi data sangat besar karena periode pengambilan data 15 detik selama 2 tahun. Agar dapat digunakan oleh program simulasi maka data dalam dokumen notepad dipindah dulu ke dalam *excel* dengan cara *copy paste* biasa disimpan dengan nama file *book1.xls*.

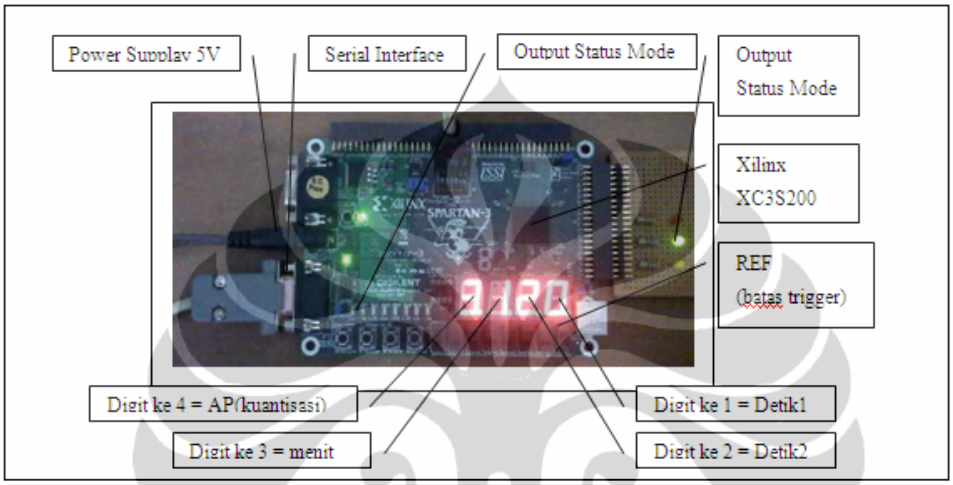


Gambar 5.6 Data *BPR* dalam bentuk dokumen notepad



Gambar 5.7 Data BPR dalam bentuk dokumen *excel*

Dalam integrasi system program *CPU OBU* akan merespon data program simulasi yang diumpan, program *CPU OBU* akan memantau data terus menerus dengan algoritma deteksi *tsunami* yang telah ditanam diprogram *CPU OBU* dan akan memberikan *feedback* sesuai dengan kerja algoritma tersebut. program *CPU OBU* telah dibahas pada *sub bab* 4.1. pada sub bab ini dijelaskan bagian-bagian dari Xilinx Spartan 3 sebagai *CPU OBU*, penjelasannya seperti pada Gambar 5.8.



Gambar 5.8 *Xilinx Spartan* 3 sebagai *CPU OBU*

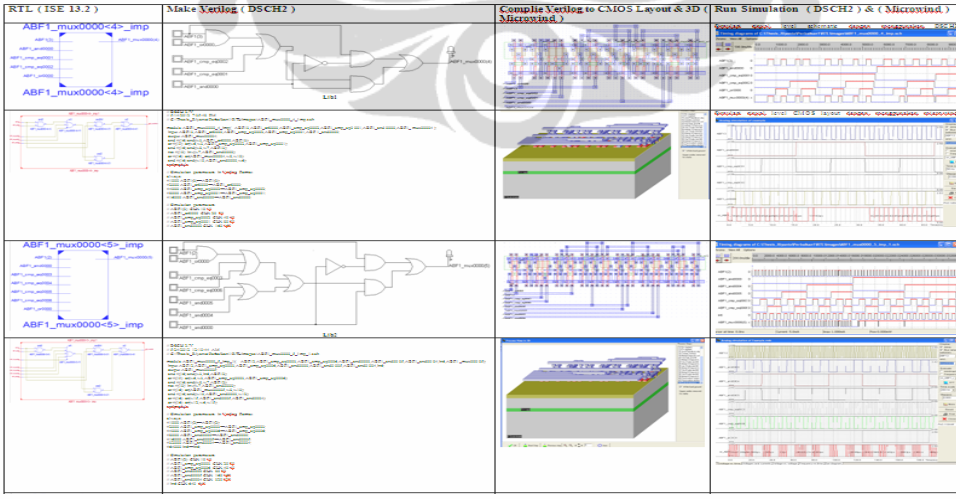
Dengan program simulasi data pengukuran BPR sebagai input dan *CPU OBU* yang bekerja seperti *Flowchart* diagram Gambar 5.4 maka respons prediction pressure (PP) yang diperoleh dapat dibandingkan dengan actual pressure (AP) seperti grafik hasil analisa data pada Gambar 5.9 berikut

Gambar 5.9 Analisa data respon dari CPU OBU

RTL hasil dari kode VHDL CPU OBU diperoleh data jumlah logic yang digunakan adalah 699 menggunakan flipflop sebanyak 347, hal ini sangat besar untuk dibuat dokumen yang dapat dibaca dan difahami. Hasil pencetakan agar dokumen dapat dibaca dengan jelas adalah 10 halaman dengan ukuran A1, berikut ini 1 lembar dari 10 halaman.

Gambar 5.10 Print screen RTL *Schematic* CPU OBU dalam dokumen A1

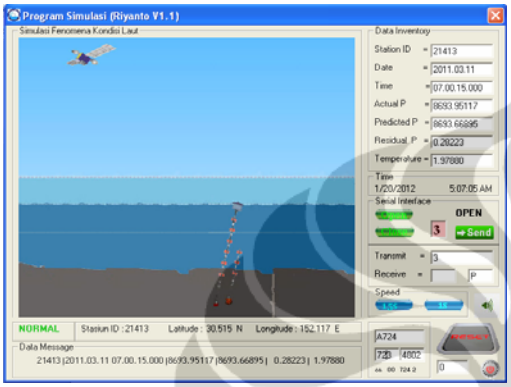
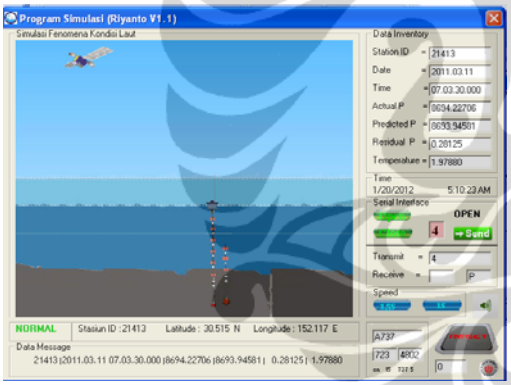
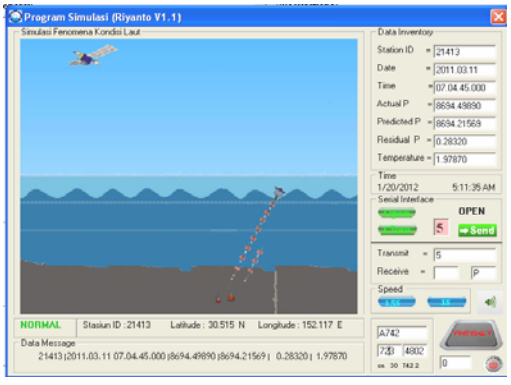
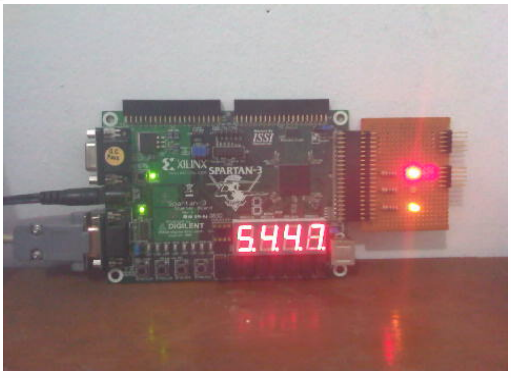
Hasil schematic yang salah satunya seperti terlihat pada Gambar 5.10 diinserting perblok dan akan diperoleh schematic pada level gate dan pada lavel gate ini yang akan dirubah peroleh kode verilog dengan menggunakan program DSCH2, dan berikutnya setelah mendapatkan kode verilog, kode ini dapat dicompile menjadi CMOS layout dengan menggunakan program microwind. Jadi metode desain bisa disimpulkan seperti tabel 5.2 berikut



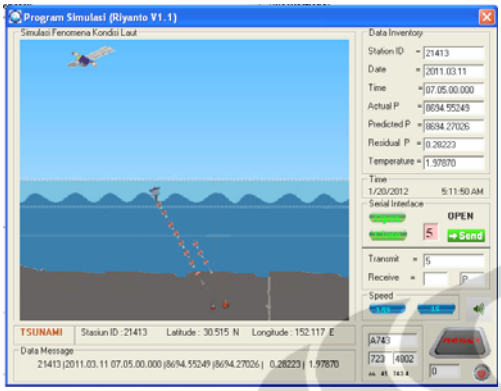
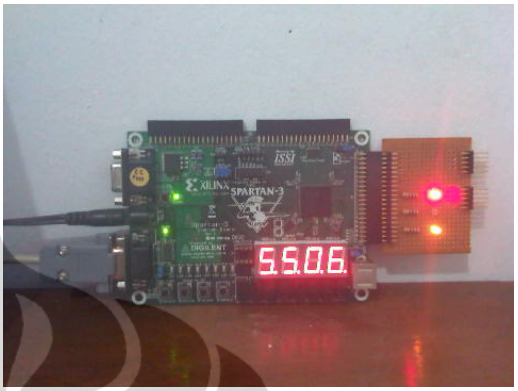
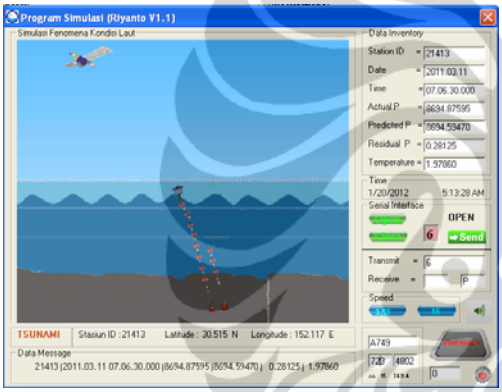
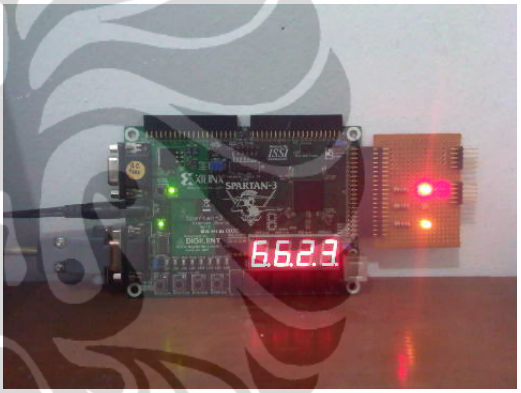
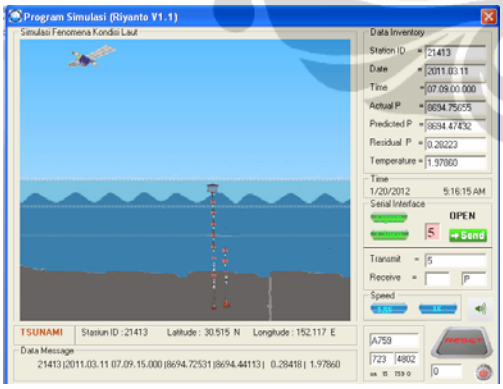
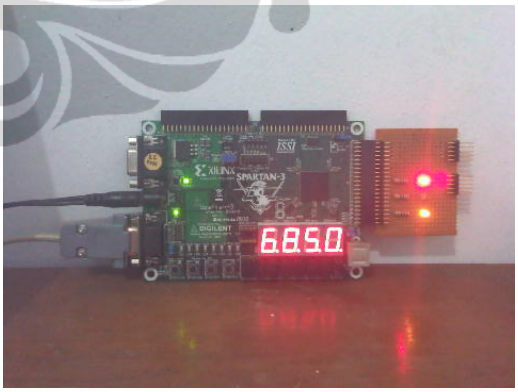
Tabel 5.2 Print screen Tabel design VLSI 0.25 µm CPU OBU dalam dokumen A1

Tabel 5.2 diatas baru dua blok dari seluruh blok schematic rangkaian yang ada, untuk lebih lengkapnya tabel design VLSI 0.25 µm CPU OBU dalam dokumen A1 tersimpan dalam e-file tersendiri dan tidak memungkinkan ditampilkan dalam laporan dikarenakan file yang besar yaitu dokumen dengan ukuran A1 sebanyak kurang lebih 100 halaman

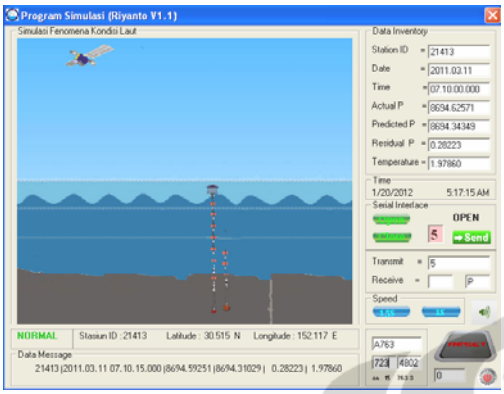
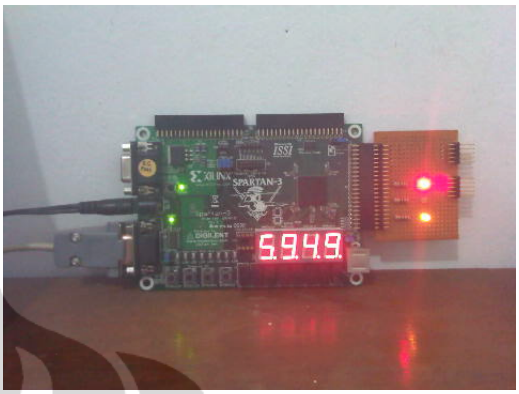
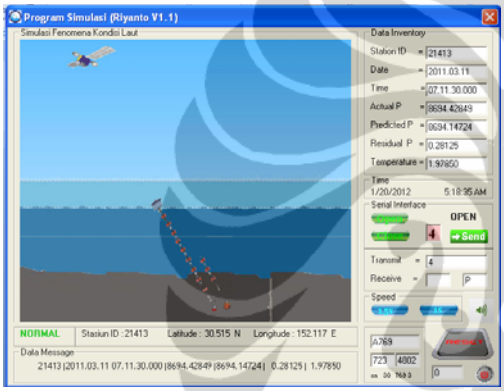
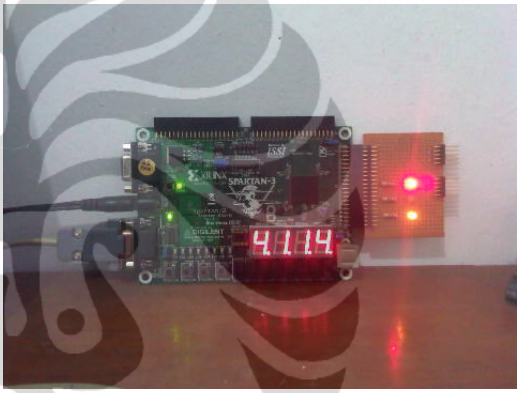
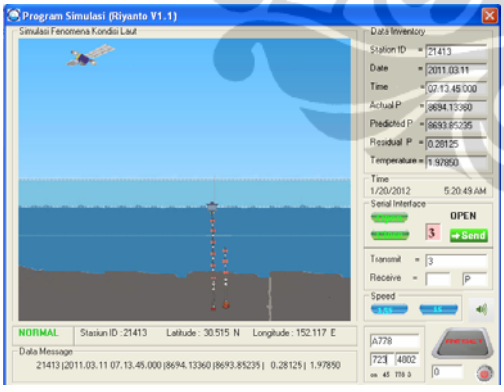
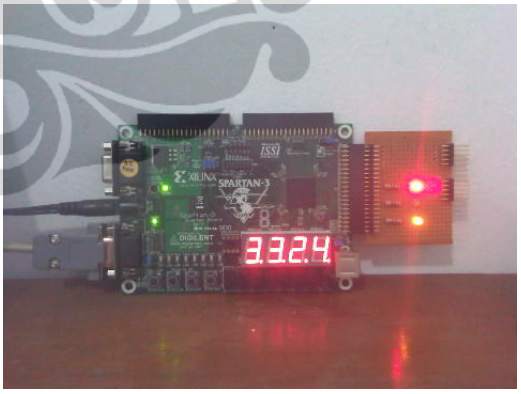
5.2 Demo Sistem

Program Simulasi, Data Inventory, Data Serial	Respons Program CPU OBU Xilinx Spartan 3
Waktu Pencacahan : 5:07:05 AM	Baris Data BPR (dari excel) = A724
	
Waktu Pencacahan : 5:10:23 AM	Baris Data BPR (dari excel) = A737
	
Waktu Pencacahan : 5:11:35 AM	Baris Data BPR (dari excel) = A742
	

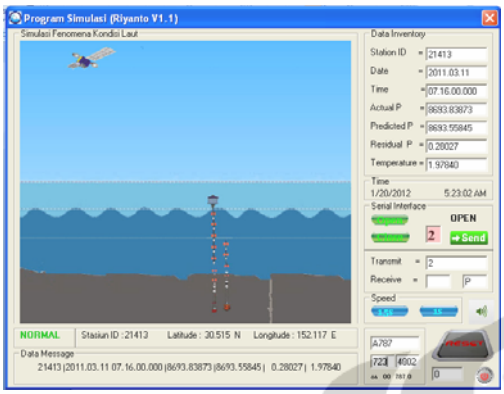
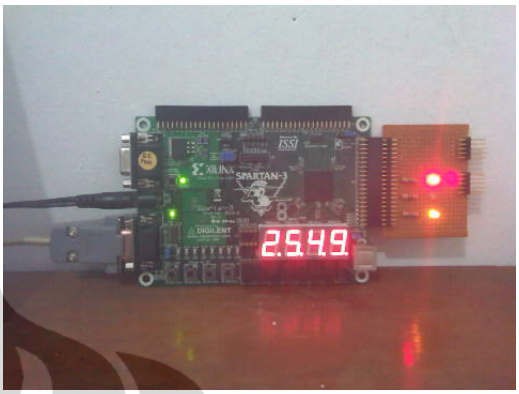
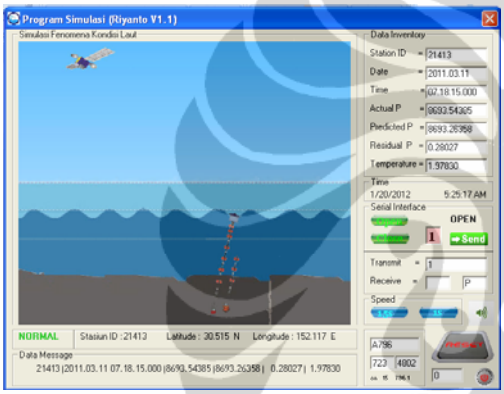
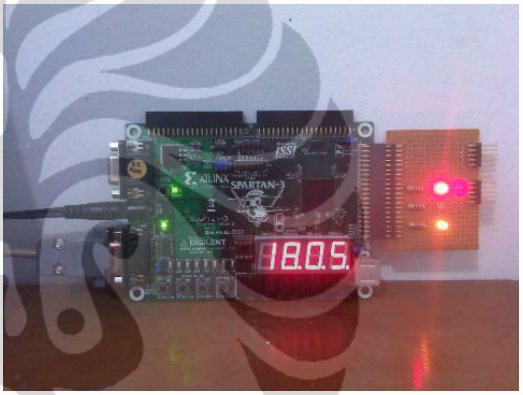
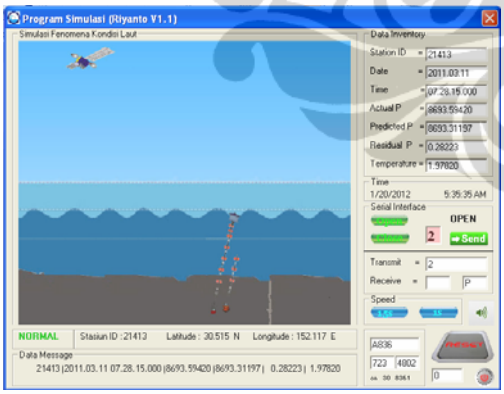
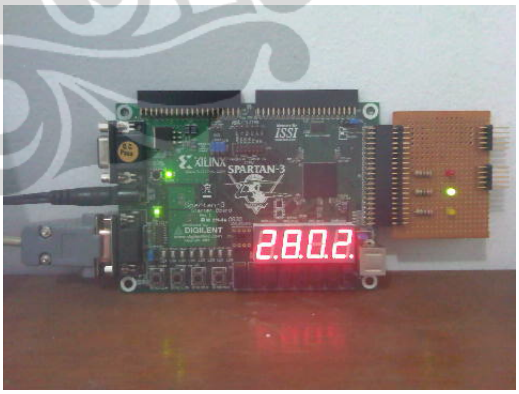
Tabel 5.3 *Experiment demo 1*

Program Simulasi, Data Inventory, Data Serial	Respons Program CPU OBU Xilinx Spartan 3
Waktu dari Data Inventory : 07:05:00:000	Baris Data BPR (dari excel) = A743
	
Waktu dari Data Inventory : 07:06:30:000	Baris Data BPR (dari excel) = A749
	
Waktu dari Data Inventory : 07:09:00:000	Baris Data BPR (dari excel) = A759
	

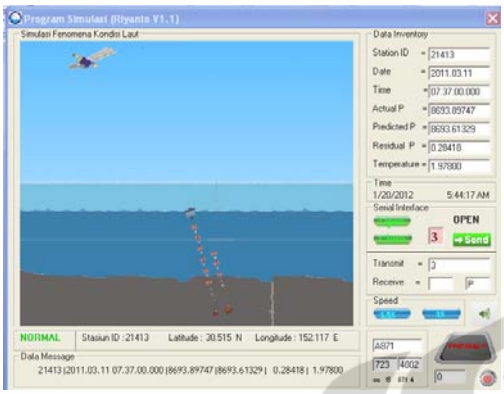
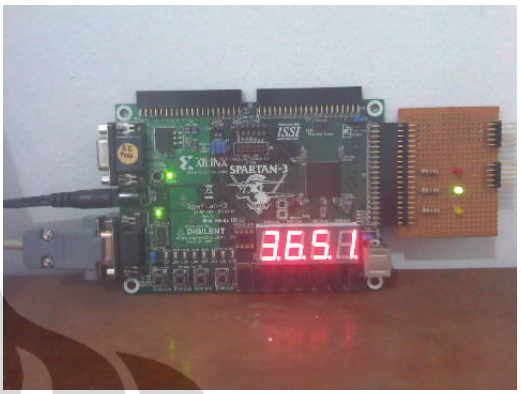
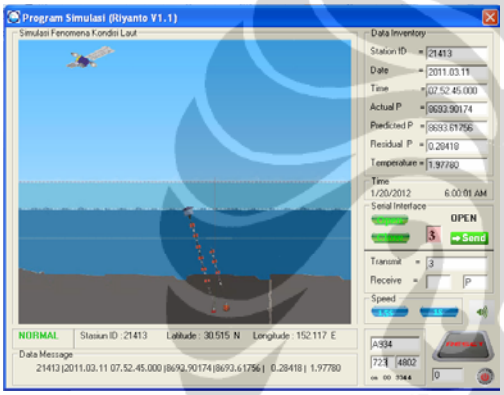
Tabel 5.4 *Experiment demo 2*

Program Simulasi, Data Inventory, Data Serial	Respons Program CPU OBU Xilinx Spartan 3
Waktu dari Data Inventory : 07:10:00:000 	Baris Data BPR (dari excel) = A763 
Waktu dari Data Inventory : 07:11:30:000 	Baris Data BPR (dari excel) = A769 
Waktu dari Data Inventory : 07:13:45:000 	Baris Data BPR (dari excel) = A778 

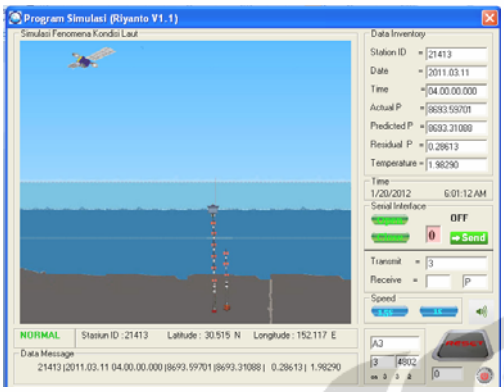
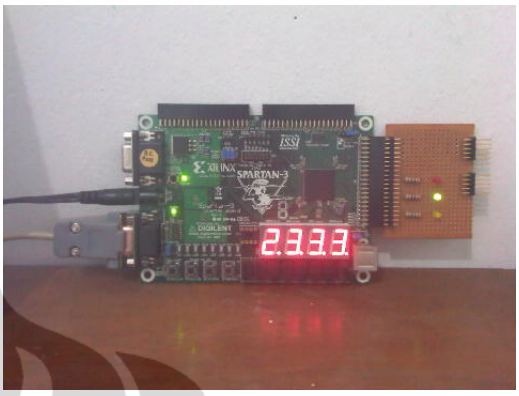
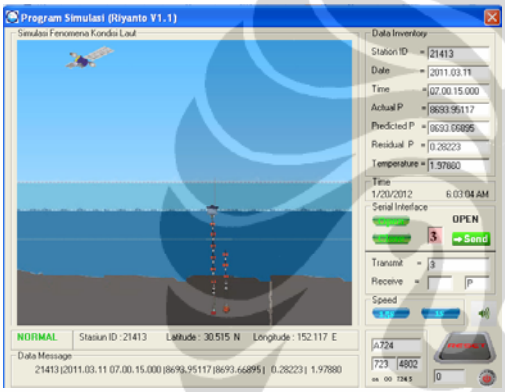
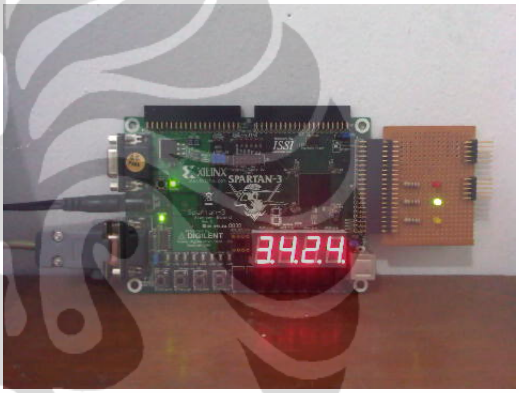
Tabel 5.5 *Experiment demo 3*

Program Simulasi, Data Inventory, Data Serial	Respons Program CPU OBU Xilinx Spartan 3
Waktu dari Data Inventory : 07:10:00:000 	Baris Data BPR (dari excel) = A787 
Waktu dari Data Inventory : 07:11:30:000 	Baris Data BPR (dari excel) = A796 
Waktu dari Data Inventory : 07:13:45:000 	Baris Data BPR (dari excel) = A836 

Tabel 5.6 *Experiment demo 4*

Program Simulasi, Data Inventory, Data Serial	Respons Program CPU OBU Xilinx Spartan 3
Waktu dari Data Inventory : 07:10:00:000 	Baris Data BPR (dari excel) = A871 
Waktu dari Data Inventory : 07:11:30:000 	Baris Data BPR (dari excel) = A934 

Tabel 5.7 *Experiment demo 5*

Program Simulasi, Data Inventory, Data Serial	Respons Program CPU OBU Xilinx Spartan 3
Waktu dari Data Inventory : 07:10:00:000 	Baris Data BPR (dari excel) = A3 
Waktu dari Data Inventory : 07:11:30:000 	Waktu Pencacahan : A724 

Tabel 5.8 *Experiment demo 6*

BAB VI
KESIMPULAN DAN SARAN

6.1 KESIMPULAN

Hasil analisa data respon dari CPU OBU diperoleh grafik prediction pressure dan dibandingkan dengan actual. Pada saat terjadi anomaly dari perbandingan grafik ada selisih 30 mm.

Hasil perancangan VLSI 0,25 μm pada CPU OBU diperoleh data jumlah logic yang digunakan adalah 699 menggunakan flipflop sebanyak 347. Sedangkan dalam teknologi VLSI kapasitas adalah 10k -1M, dengan metode hybrid VHDL desain CPU OBU dapat ditingkatkan penambahan jumlah gate dengan cara meningkatkan memori simpan sebanyak mungkin.

File schematic yang dihasilkan oleh RTL dari CPU OBU sangat besar sehingga desain VLSI 0,25 μm untuk menghasilkan CMOS layout disertakan tersendiri dalam tabel desain VLSI 0,25 μm untuk CPU OBU dalam bentuk e-file dikarenakan tidak memungkinkan untuk ditampilkan.

.

6.2 SARAN

Dengan waktu yang sangat sempit dalam melakukan proses perancangan system yang sangat kompleks dengan mendapatkan hasil yang sempurna sangat perlu diapresiasi.

Tidak ada gading yang tak retak, dari semua proses perancangan yang sudah dilalui tentu terdapat celah dan kekurangan, untuk itu perlu adanya dukungan untuk memperbaiki bersama.

Sebaiknya dalam perancangan yang demikian komplek diperlukan pembentukan team desain, tidak hanya satu orang desainer saja.

DAFTAR ACUAN

[1] Christian Meinig, Scott E. Stalin, Alex I. Nakamura, Frank Gonzalez and Hugh B. Milburn, "Technology Developments in Real-Time Tsunami Measuring, Monitoring and Forecasting", NOAA, Pacific Marine Environmental Laboratory (PMEL) 2007.

[2] A. Macrander (1), V. Gouretski (1,*), O. Boebel (1),(1) Alfred-Wegener-Institute fur Polar- und Meeresforschung, Bussestr. 24, D-27570 Bremerhaven, Germany (*) now at Institut fur Meereskunde, Bundesstr. 53, D-20146 Hamburg, Germany "PACT – a Bottom Pressure Based, Compact Deep-Ocean Tsunameter with Acoustic Surface Coupling", IEEE, 2009.

[3] R. A. Lawson, Science Aplications International Corporation, 4065 Hancock Street San Diego, CA 92110 USA, "Tsunami Detection System for International Requirements", MTS, 2007.

[4] George Georgiou(1), Andrew M Clark(2), George Zodiatis(1), Dan Hayes(1), Dimitris Glekas(3); (1) Cyprus Oceanography Centre, University of Cyprus, Nicosia, CYPRUS ; (2) CSnet International, Inc, Stuart, Florida, USA ; (3) CSnet (CYPRUS) Ltd, Limassol, CYPRUS, "Design of Prototype Tsunami Warning and Early Response system for Cyprus – TWERC" IEEE,2010.

[5] H.O.Mofjeld harold.mofjeld@noaa.gov ,"Tsunami detection algorithm " , http://www.pmel.noaa.gov/tsunami/tda_documentation.html

[6] Peter Frederikssen, Group 42: Yang Liu, Ru Liu, Jingjing Liu, Lei Hong, Shenyuan Wang, Hui Zhao, Kangming Wu, "Pacific tsunami warning system is creditable or not", 4th Semester, Spring 2007

[7] C. Mathew Cherian(1) , Nivethitha Jayaraj (1), Ganesh Vaidyanathan S.(3) ; (1) Department of Electronics and Communication Engineering Sri Venkateswara College of Engineering Sriperumbudur, Chennai, India 602105, (2) Assistant professor(1)"Artificially Intelligent Tsunami Early Warning System," IEEE, 2010.

[8] M.C. Eble And F.I.Gonzalez ; Pacific Marine Environmental Laboratory, National Oceanic and Administration, Seattle, Washington"Deep-Ocen Bottom Pressure Measurements in the Northeast Pacific"Journal of Atmospheric and Oceanic Technology Volume8, 1990.

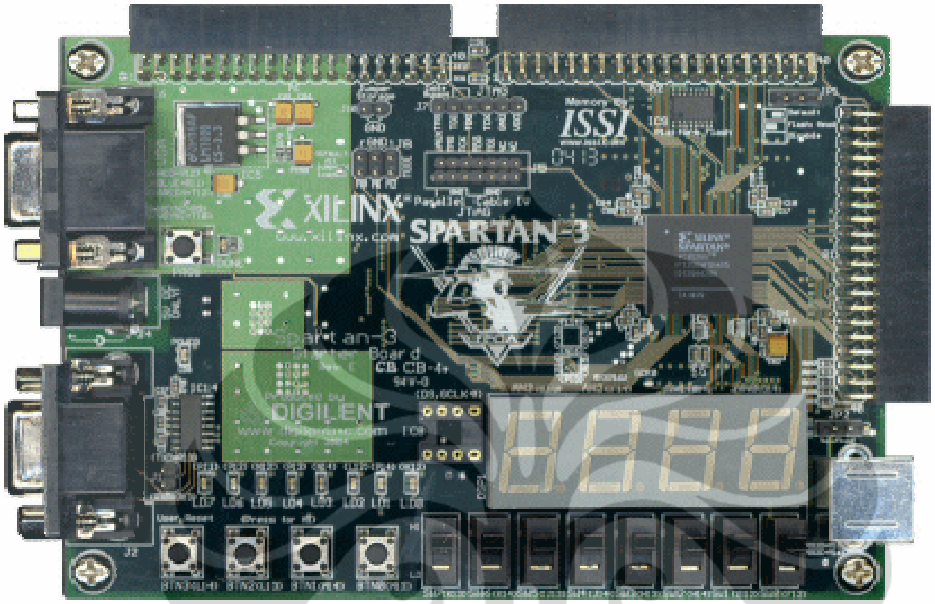
[9] Yuchiro Fujii[1], Kenji Satake[2], Shin-ichi Sakai[2], Masanao Shinohara[2] and Toshihiko Kanazawa[2]; [1] International Institute of Seismology and Earthquake Engineering (IISEE),Building Research Institute (BRI) 1-3 Tachihara, Tsukuba, Ibaraki 305-0802, Japan ; [2] Earthquake Research Institute (ERI), University of Tokyo 1-1-1 Yayoi, Bhunkyo-ku, Tokyo 113-0032, Japan. " Tsunami Source of the 2011 off the pacific coast of Tohoku, Japan Earthquake" EPS 2011

[10] Duong Tran, Kyung Ki Kim, Yong-Bin Kim ; Department of Electrical and Computer Engineering Northeastern University, Boston, MA, 02115, USA" Power Estimation in Digital CMOS VLSI Chips" IEEE, 2005.

[11] Matthias Passlack, Manfred Uhle, And Horst Elschner , "Analysis of Propagation Delays in High-Speed VLSI Circuits Using a Distributed Line Model" IEEE, 1990.
[12] Neil Weste, Kamran Eshraghian, "Principles of CMOS VLSI Design A System Perspective," AT&T Bell Laboratories, Incorporated, 1985 .
[13] William Stallings, "Data & Computer Communications, 6th Edition," Prentice-Hall, Inc, 2000.
[14] Pong P. Chu, " FPGA Prototyping by VHDL Examples: Xilinx Spartan-3 Version", Wiley-Interscience, 2008 .
[15] Pong P. Chu, "Embedded SoPC Design with Nios II Processor and VHDL Examples", Wiley 2011.
[16] Pong P. Chu, "FPGA Prototyping By Verilog Examples: Xilinx Spartan-3 Version", Wiley-Interscience 2008.
[17] *User's Manul ,"Digiquartz Broadband Intelligent Instruments with Dual RS-232 and RS-485 Interfaces"* Paroscientific. Inc. Digiquartz Pressure Instrumentation, 2010.
[18] Installation Manual "SAILOR TT-3026D/S/M Maritime mini-c" *SAILOR Thrane & Thrane,2006*.
[19] Tsunami Data and Information, "Bottom Pressure Recorder (BPR) Data available for download", NOAA (National Geophysical Data Center). http://www.ngdc.noaa.gov/nndc/struts/results?&t=102597&s=1&d=1
[20] Marie C.Eble and Schott E.Stalin, "Description of Real-time DART System Messages", U.S Nasional Oceanic & Atmospheric Administration Pacific Marine Environmental Laboratory Engineering Development Division 7600 Sand Point Way Seattle, WA 98115. http://www.nctr.pmel.noaa.gov/Dart/Pdf/dartMsgManual3.01.pdf
[21] Wikipedia, the free encyclopedia, " Seismic wave", , This page was last modified on 20 December 2011 at 06:26. http://en.wikipedia.org/wiki/Seismic_wave
[22] Sri Atmaja P. Rosyidi, ST. [1], M.Sc.Eng, Ph.D. [2], P.Eng. [3], [1] Staf Pengajar Jurusan Teknik Sipil , Fakultas Teknik, Universitas Muhammadiyah Yogyakarta, Jalan Lingkar Selatan, Yogyakarta 55183, Email: atmaja_sri@umy.ac.id. [2] Postdoctoral Research Associate, Geohazards and geoenvironments Research Group and Geotechnical and Geoenvironmental Engineering Research Group, Universiti Kebangsaan Malaysia (UKM) 43600 Bangi, Selangor Malaysia.[3] Senior Principal Research Fellow, Center for Regional Energy Management (CREM) Universitas Muhammadiyah Yogyakarta, Energy Conservation in Built Environment., " penggunaan metode analisis gelombang seismik permukaan (spectral analysis of surface wave) untuk pengembangan teknik evaluasi tanpa rusak (ndt) perkerasan lentur dan kaku di indonesia", , Stadium General Sekolah Tinggi Teknik Dumai, Provinsi Riau Dumai, Sabtu 26 Juni 2010.
[23] Gian Mario Beltrami, Marcello Di Risio and Paolo De Girolamo DISAT-LIAM - Universit`a di L'Aquila Italy, "Algorithms for Automatic, Real-Time Tsunami Detection in Sea Level Measurements", The Tsunami Threat - Research and Technology
[24] Ina Buoy BPPT, "InaBuoy LapTPSA 28des 2009" Laporan presentasi 2009

DAFTAR LAMPIRAN

Lampiran 1 Xilinx Spartan 3



Gambar Lampiran 1 Pengkabelan Programming Xilinx Spartan 3

The Spartan-3 Starter Kit board has three 40-pin expansion connectors labeled A1, A2, and B1. The A1 and A2 connectors, indicated as (21) and (20), respectively, in Figure 1-2, are on the top edge of the board. Connector A1 is on the top left, and A2 is on the top right. The B1 connector, indicated as (19) in Figure 1-2, is along the right edge of the board.

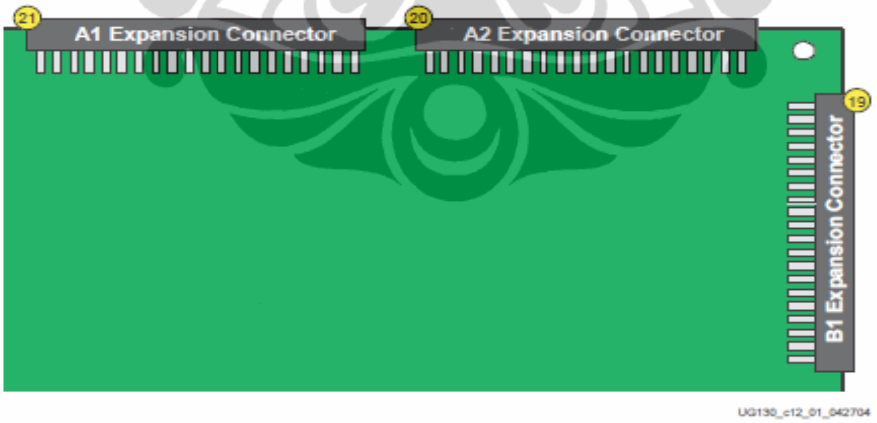


Figure 13-1: **Spartan-3 Starter Kit Board Expansion Connectors**

Table 13-1 summarizes the capabilities of each expansion port. Port A1 supports a maximum of 32 user I/O pins, while the other ports provide up to 34 user I/O pins. Some pins are shared with other functions on the board, which may reduce the effective I/O count for specific applications. For example, pins on the A1 port are shared with the SRAM address signals, with the SRAM OE# and WE# control signals, and with the eight least-significant data signals to SRAM IC10 only.

Gambar Lampiran 1 General Purpose Input Output

A1 Connector Pinout

Schematic Name	FPGA Pin	Connector		FPGA Pin	Schematic Name
GND		1	2		VU (+5V)
V_{CCO} (+3.3V)	V_{CCO} (all banks)	3	4	(N8)	ADR0
DB0	(N7) SRAM IC10 IO0	5	6	(L5) SRAM A0	ADR1
DB1	(T8) SRAM IC10 IO1	7	8	(N3) SRAM A1	ADR2
DB2	(R6) SRAM IC10 IO2	9	10	(M4) SRAM A2	ADR3
DB3	(T5) SRAM IC10 IO3	11	12	(M3) SRAM A3	ADR4
DB4	(R5) SRAM IC10 IO4	13	14	(L4) SRAM A4	ADR5
DB5	(C2) SRAM IC10 IO5	15	16	(G3) SRAM WE#	WE
DB6	(C1) SRAM IC10 IO6	17	18	(K4) SRAM OE#	OE
DB7	(B1) SRAM IC10 IO7	19	20	(P9) FPGA DOUT/BUSY	CSA
LSBCLK	(M7)	21	22	(M10)	MA1-DB0
MA1-DB1	(F3) SRAM A6	23	24	(G4) SRAM A5	MA1-DB2
MA1-DB3	(E3) SRAM A8	25	26	(F4) SRAM A7	MA1-DB4
MA1-DB5	(G5) SRAM A10	27	28	(E4) SRAM A9	MA1-DB6
MA1-DB7	(H4) SRAM A12	29	30	(H3) SRAM A11	MA1-ASTB
MA1-DSTB	(J3) SRAM A14	31	32	(J4) SRAM A13	MA1-WRITE
MA1-WAIT	(K5) SRAM A16	33	34	(K3) SRAM A15	MA1-RESET
MA1-INT	(L3) SRAM A17	35	36	JTAG Isolation	JTAG Isolation
TMS	(C13) FPGA JTAG TMS	37	38	(C14) FPGA JTAG TCK	TCK
TDO-ROM	Platform Flash JTAG TDO	39	40	Header J7, pin 3	TDO-A

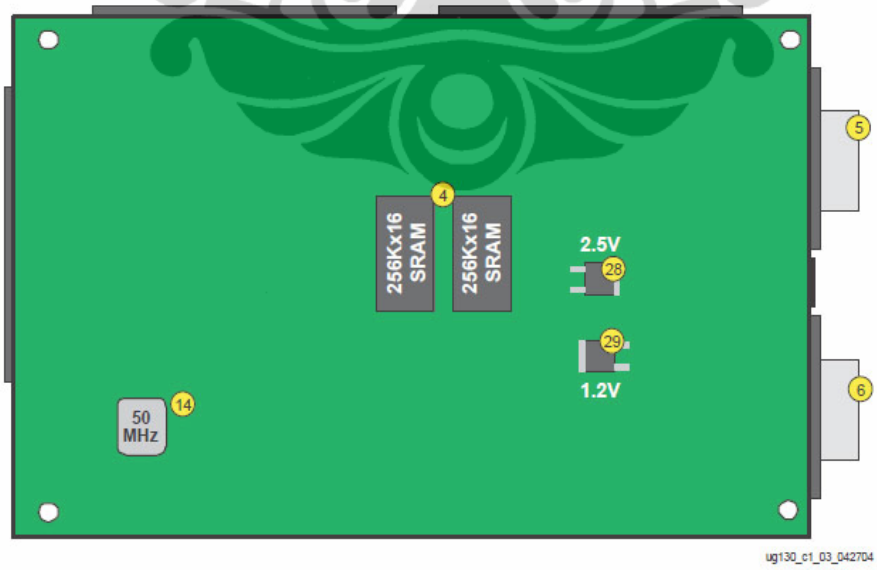
A2 Connector Pinout

Schematic Name	FPGA Pin	Connector		FPGA Pin	Schematic Name
GND		1	2		VU (+5V)
V_{CCO} (+3.3V)	V_{CCO} (all banks)	3	4	(E6)	PA-IO1
PA-IO2	(D5)	5	6	(C5)	PA-IO3
PA-IO4	(D6)	7	8	(C6)	PA-IO5
PA-IO6	(E7)	9	10	(C7)	PA-IO7
PA-IO8	(D7)	11	12	(C8)	PA-IO9
PA-IO10	(D8)	13	14	(C9)	PA-IO11
PA-IO12	(D10)	15	16	(A3)	PA-IO13
PA-IO14	(B4)	17	18	(A4)	PA-IO15
PA-IO16	(B5)	19	20	(A5)	PA-IO17
PA-IO18	(B6)	21	22	(B7)	MA2-DB0
MA2-DB1	(A7)	23	24	(B8)	MA2-DB2
MA2-DB3	(A8)	25	26	(A9)	MA2-DB4
MA2-DB5	(B10)	27	28	(A10)	MA2-DB6
MA2-DB7	(B11)	29	30	(B12)	MA2-ASTB
MA2-DSTB	(A12)	31	32	(B13)	MA2-WRITE
MA2-WAIT	(A13)	33	34	(B14)	MA2-RESET
MA2-INT/GCK4	(D9) Oscillator socket	35	36	(B3) FPGA PROG_B	PROG-B
DONE	(R14) FPGA DONE	37	38	(N9) FPGA INIT_B	INIT
CCLK	(T15) FPGA CCLK Connects to (A14) via 390Ω resistor	39	40	(M11)	DIN

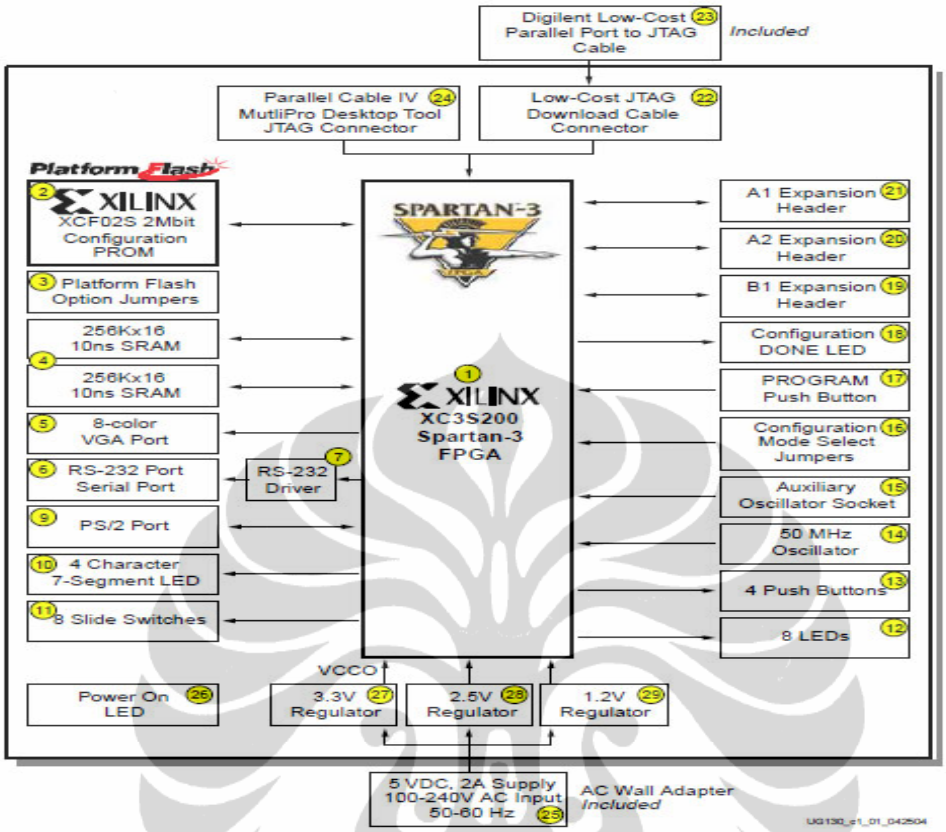
B1 Connector Pinout

Schematic Name	FPGA Pin	Connector		FPGA Pin	Schematic Name
GND		1	2		VU (+5V)
V_{CCO} (+3.3V)	V_{CCO} (all banks)	3	4	(C10)	PB-ADR0
PB-DB0	(T3) FPGA RD_WR_B config	5	6	(E10)	PB-ADR1
PB-DB1	(N11) FPGA D1 config	7	8	(C11)	PB-ADR2
PB-DB2	(P10) FPGA D2 config	9	10	(D11)	PB-ADR3
PB-DB3	(R10) FPGA D3 config	11	12	(C12)	PB-ADR4
PB-DB4	(T7) FPGA D4 config	13	14	(D12)	PB-ADR5
PB-DB5	(R7) FPGA D5 config	15	16	(E11)	PB-WE
PB-DB6	(N6) FPGA D6 config	17	18	(B16)	PB-OE
PB-DB7	(M6) FPGA D7 config	19	20	(R3) FPGA CS_B config	PB-CS
PB-CLK	(C15)	21	22	(C16)	MB1-DB0
MB1-DB1	(D15)	23	24	(D16)	MB1-DB2
MB1-DB3	(E15)	25	26	(E16)	MB1-DB4
MB1-DB5	(F15)	27	28	(G15)	MB1-DB6
MB1-DB7	(G16)	29	30	(H15)	MB1-ASTB
MB1-DSTB	(H16)	31	32	(J16)	MB1-WRITE
MB1-WAIT	(K16)	33	34	(K15)	MB1-RESET
MB1-INT	(L15)	35	36	(B3) FPGA PROG_B	PROG-B
DONE	(R14) FPGA DONE	37	38	(N9) FPGA INIT_B	INIT
CCLK	(T15) FPGA CCLK Connects to (A14) via 390Ω resistor	39	40	(M11)	DIN

Bottom Layer Xilinx Spartan 3



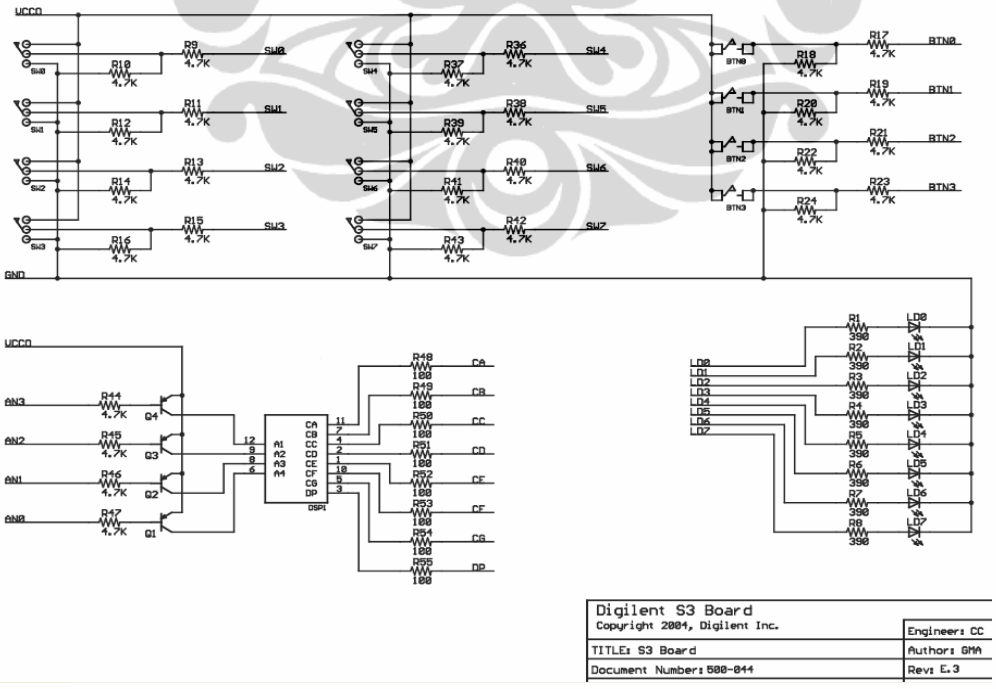
Feature Xilinx Spartan 3



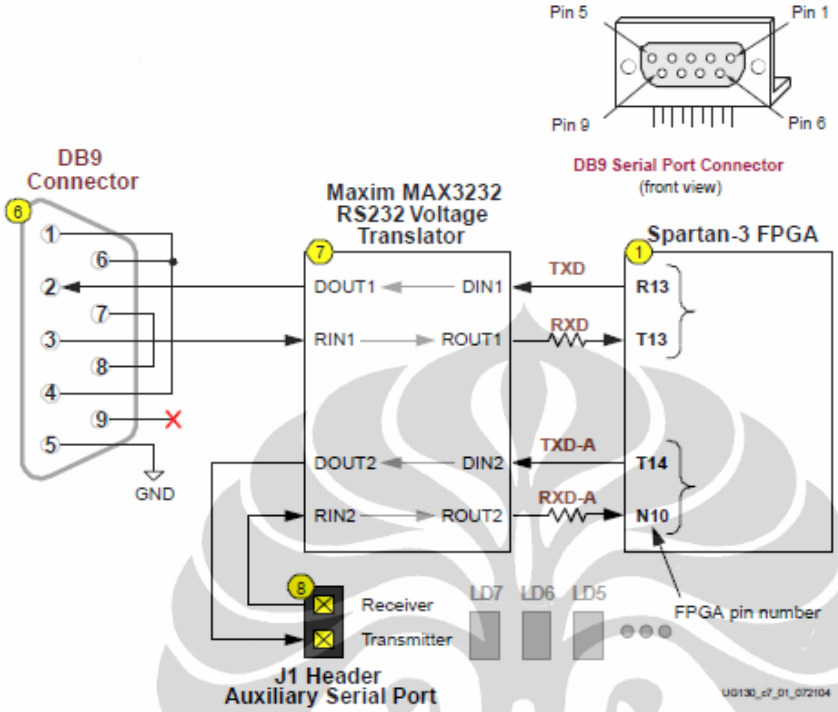
- **200,000-gate Xilinx Spartan-3 XC3S200 FPGA in a 256-ball thin Ball Grid Array package (XC3S200FT256)**
  - 4,320 logic cell equivalents
  - Twelve 18K-bit block RAMs (216K bits)
  - Twelve 18x18 hardware multipliers
  - Four Digital Clock Managers (DCMs)
  - Up to 173 user-defined I/O signals
- **2Mbit Xilinx XCF02S Platform Flash, in-system programmable configuration PROM**
  - 1Mbit non-volatile data or application code storage available after FPGA configuration
  - Jumper options allow FPGA application to read PROM data or FPGA configuration from other sources
- **1M-byte of Fast Asynchronous SRAM (bottom side of board, see Figure 1-3)**
  - Two 256Kx16 ISSI IS61LV25616AL-10T 10 ns SRAMs
  - Configurable memory architecture
    - Single 256Kx32 SRAM array, ideal for MicroBlaze code images
    - Two independent 256Kx16 SRAM arrays
  - Individual chip select per device
  - Individual byte enables
- **3-bit, 8-color VGA display port**
- **9-pin RS-232 Serial Port**
  - DB9 9-pin female connector (DCE connector)
  - RS-232 transceiver/level translator
  - Uses straight-through serial cable to connect to computer or workstation serial port
  - Second RS-232 transmit and receive channel available on board test points

- PS/2-style mouse/keyboard port (9)
- Four-character, seven-segment LED display (10)
- Eight slide switches (11)
- Eight individual LED outputs (12)
- Four momentary-contact push button switches (13)
- 50 MHz crystal oscillator clock source (bottom side of board, see Figure 1-3) (14)
- Socket for an auxiliary crystal oscillator clock source (15)
- FPGA configuration mode selected via jumper settings (16)
- Push button switch to force FPGA reconfiguration (FPGA configuration happens automatically at power-on) (17)
- LED indicates when FPGA is successfully configured (18) (19)
- Three 40-pin expansion connection ports to extend and enhance the Spartan-3 Starter Kit Board (20)
  - See www.xilinx.com/s3boards for compatible expansion cards (21)
  - Compatible with Digilent, Inc. peripheral boards https://digilent.us/Sales/boards.cfm#Peripheral
  - FPGA serial configuration interface signals available on the A2 and B1 connectors
    - PROG_B, DONE, INIT_B, CCLK, DONE
- JTAG (22) port for low-cost download cable (23)
- Digilent JTAG download/debugging cable connects to PC parallel port (23) (24)
- JTAG download/debug port compatible with the Xilinx Parallel Cable IV and MultiPRO Desktop Too
- AC power adapter input for included international unregulated +5V power supply (25)
- Power-on indicator LED (26)
- On-board 3.3V (27), 2.5V (28), and 1.2V (29) regulators

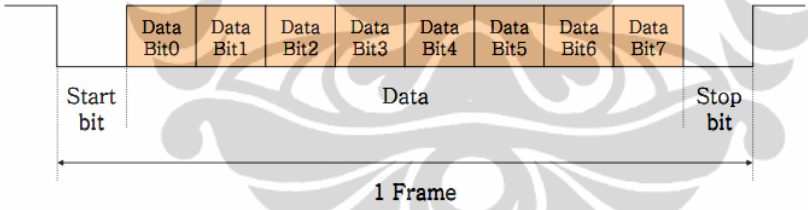
Slide Switches, Push Buttons, LEDs, and Four-Character 7-Segment Display



RS-232 Serial Port



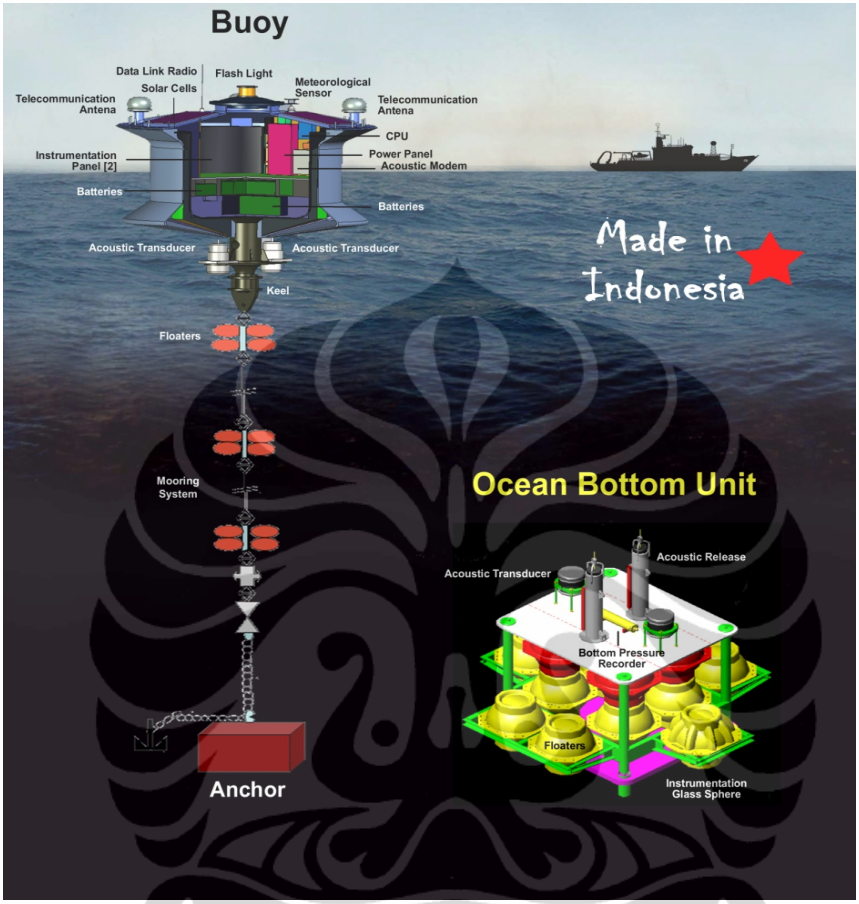
We use 1 start bit, 1 stop bit, 8 bits data, and no parity.



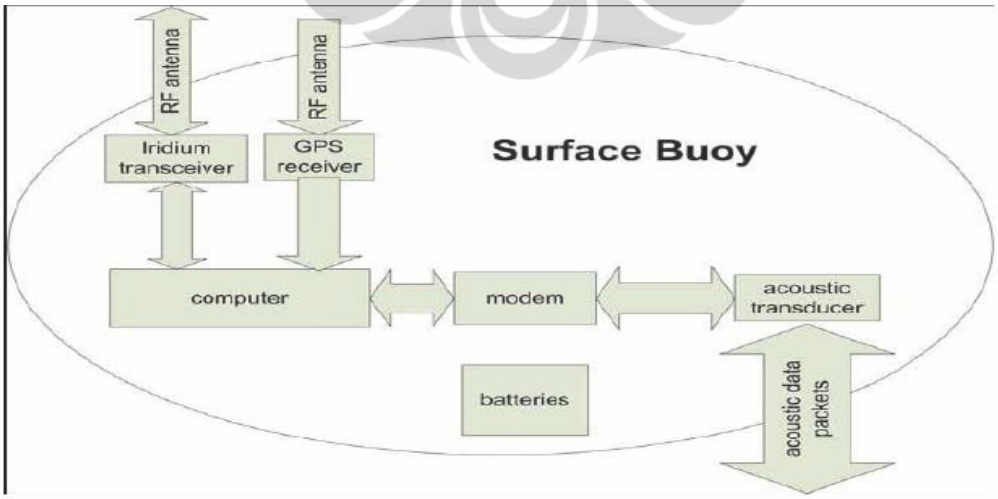
Bit Length: 1/Speed
Ex) 9600 BPS: 1/9600 = about 104.167 [us]

Frame Error should be less than 1.87%.

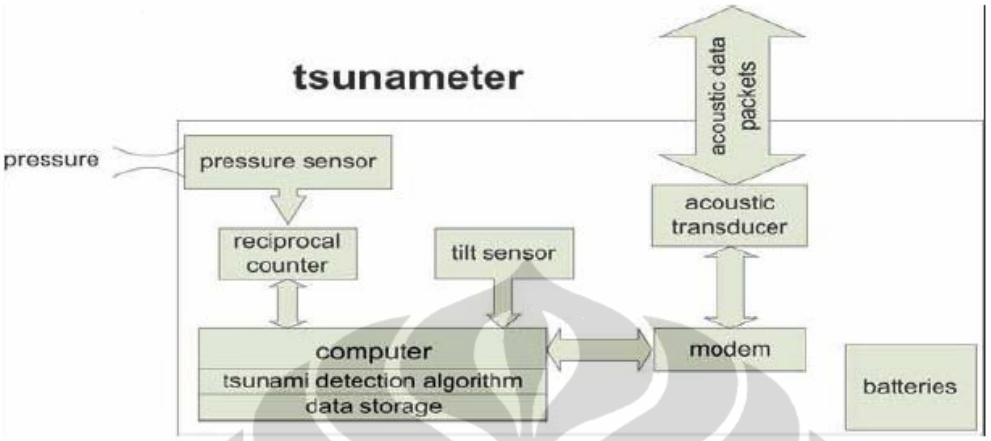
Lampiran 2 TEWS



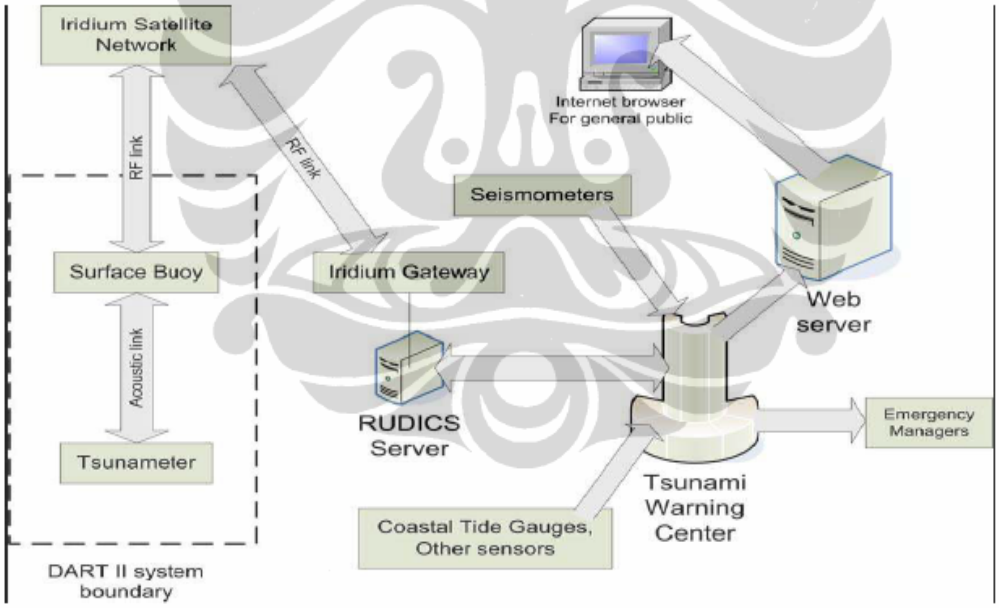
Blok Diagram Surface Buoy



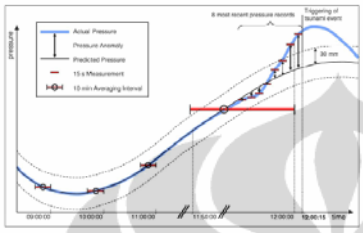
Blok Diagram Tsunameter



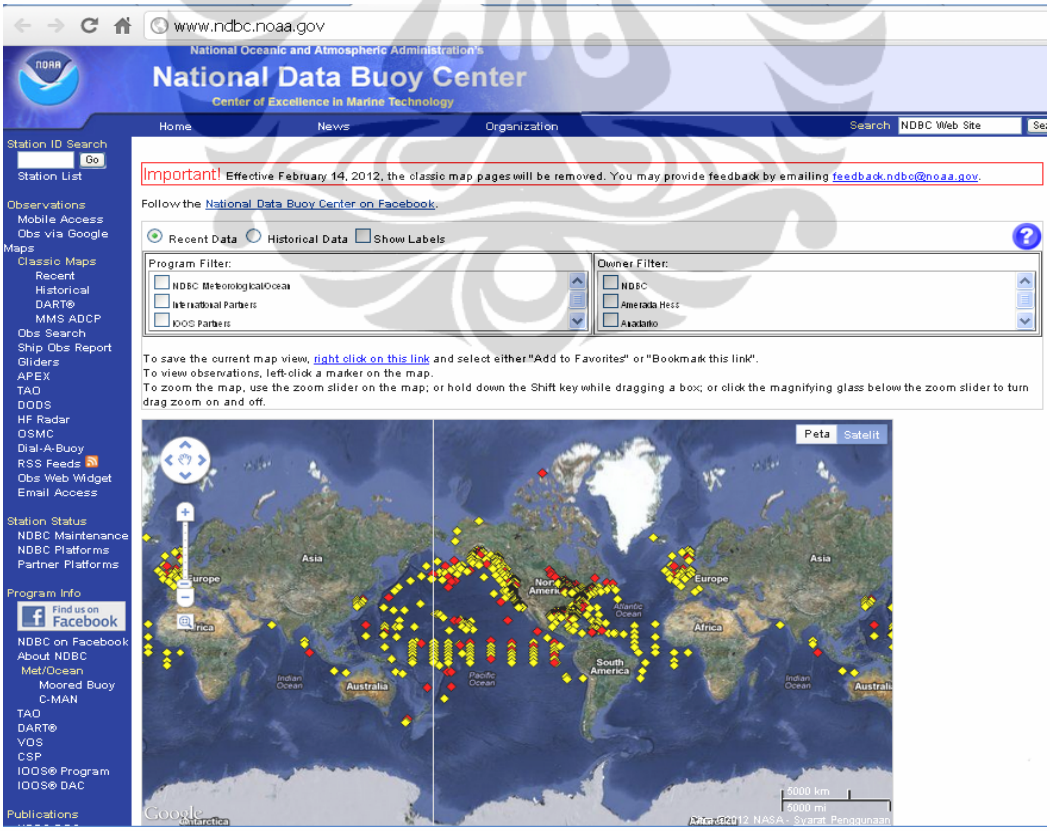
Blok Diagram TEWS



TEWS Algorithm

Algoritma	Formula & Graph	Konstanta	Description
1. Mofjeld (1997) Patent [10; US Patent 11]. Source :	**H (t')= w(i)H*(t-idt)** **p** 	Koefisien W: **w(0) = 1.16818457031250** **w(1) = -0.28197558593750** **w(2) = 0.14689746093750** **w(3) = -0.03310644531250**	**t'** adalah waktu yang sebenarnya dinyatakan dalam menit. **w(i)** adalah koefisien berasal dari hukum Newton (II) untuk ekstrapolasi maju. **Hp** prediksi yang diperbarui setiap interval sampel (yaitu setiap 15 s). * adalah tanda bintang yang menunjukkan rata-rata 10 menit dan dt = 1 jam
2. Beltrami (2008)		Koefisien W: **w(0) =2.4432451059353566** **w(1) =- 2.7008348451980630** **w(2) = 1.6554065948122720** **w(3) =- 0.3978168555495660**	

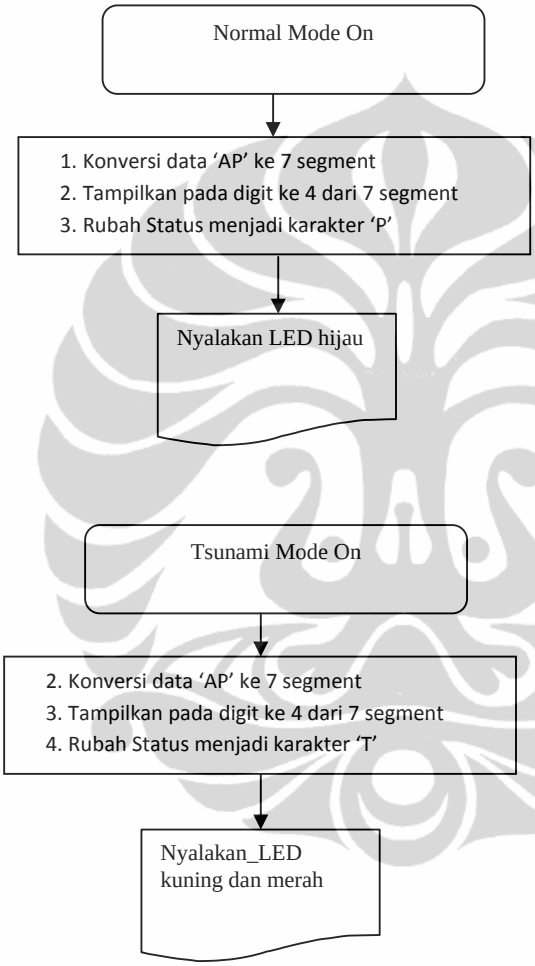
DART Monitoring & BPR data Resource



Lampiran 3 Desain Sistem CPU OBU

Flow chard tambahan CPU OBU

2 Mode CPU OBU



Jawaban E-mail, Tanya Jawab Ke DART

Dear Riyanto,
I asked around and it looks like the fields are:
Station ID
Year
Month
Day
Hour
Min
Sec
Milli-second
Raw Pressure [PSIA]
Predicted Pressure [PSIA]
Residual Pressure [PSIA]
Temperature [degrees C]
**NOTE: PSIA is converted to meters using the approximation .670 m/PSIA

Also attached is the DART messages description manual that Scott Stalin and Marie Eble put together. Scott and Marie are available to answer specific questions after you review the document.
The information applies only to U.S. developed systems and
not to those developed internally by Korea, India, and others.
I hope this is what you needed. Jean

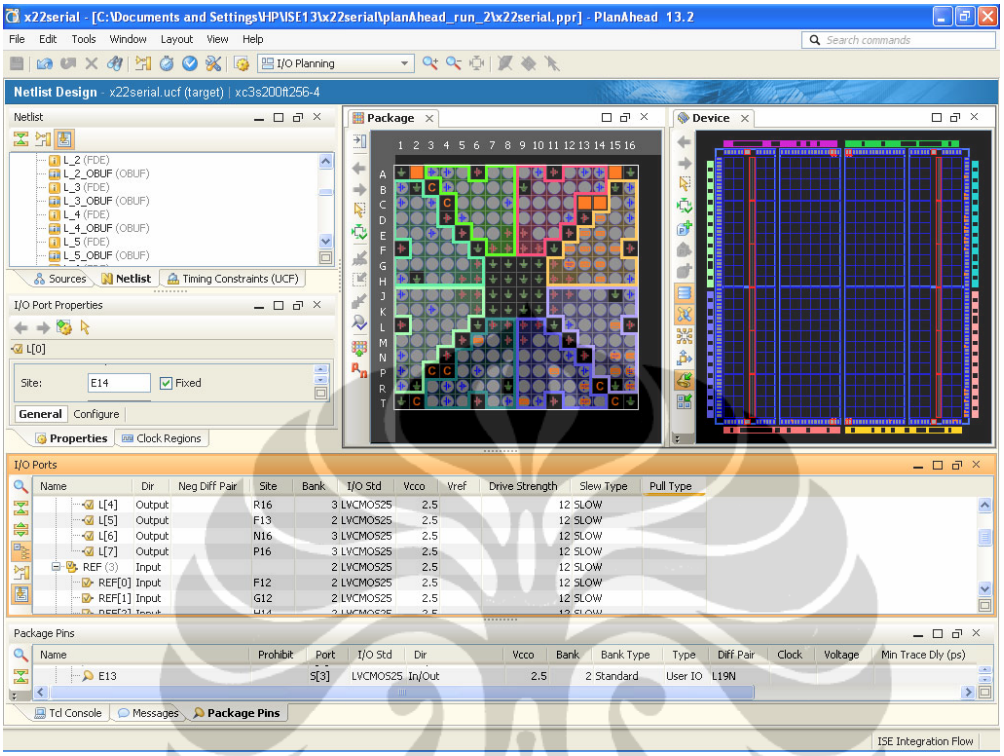
Jean Newman Tel: 206-526-6531
NOAA Center for Tsunami Research
NOAA/PMEL/OERD2 - UW/JISAO FAX: 206-526-6485
7600 Sand Point Way NE, Bldg. 3 jean.c.newman@noaa.gov
Seattle, WA 98115-6349 jcn@u.washington.edu
. URL: http://nctr.pmel.noaa.gov/

ALASAN MENGGUNAKAN CMOS BASED VLSI 0.25um

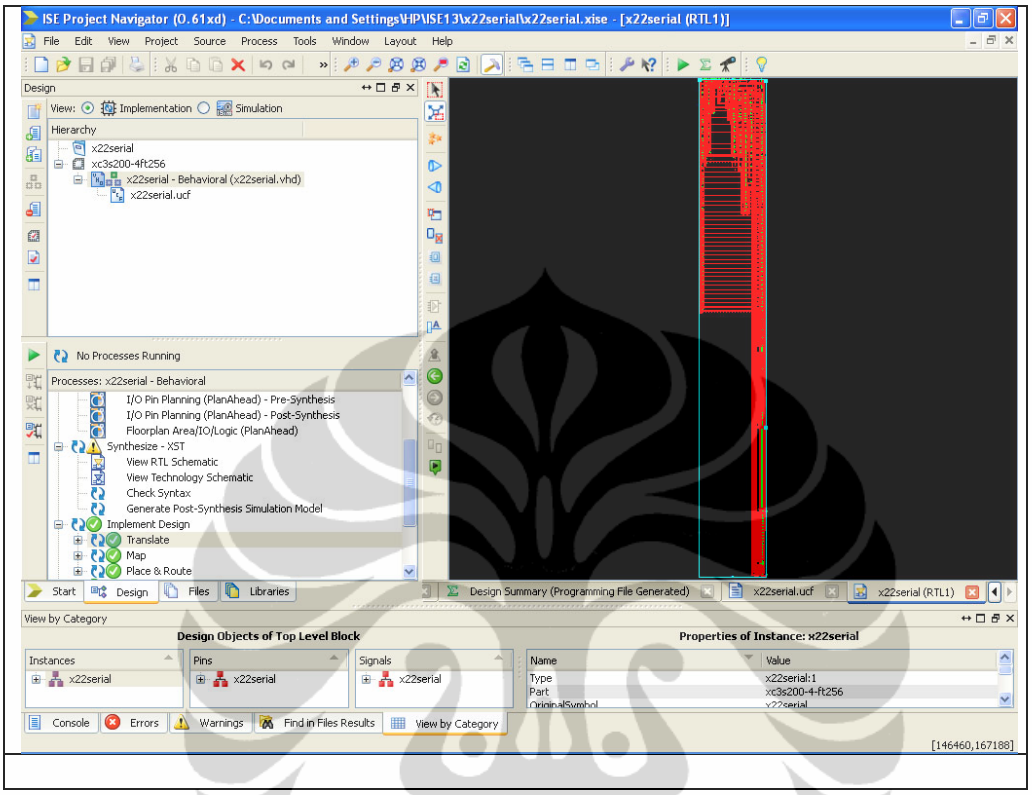
VLSI 0,25um

PC Based	CMOS Based
Multi Purpose	Single Purpose
High Energy Consumption	Low Energy Consumption
Speed Delay Process	High Speed Process
Need Operating System	Operating System is State Function
Embedded PC	Implementable to Embedded

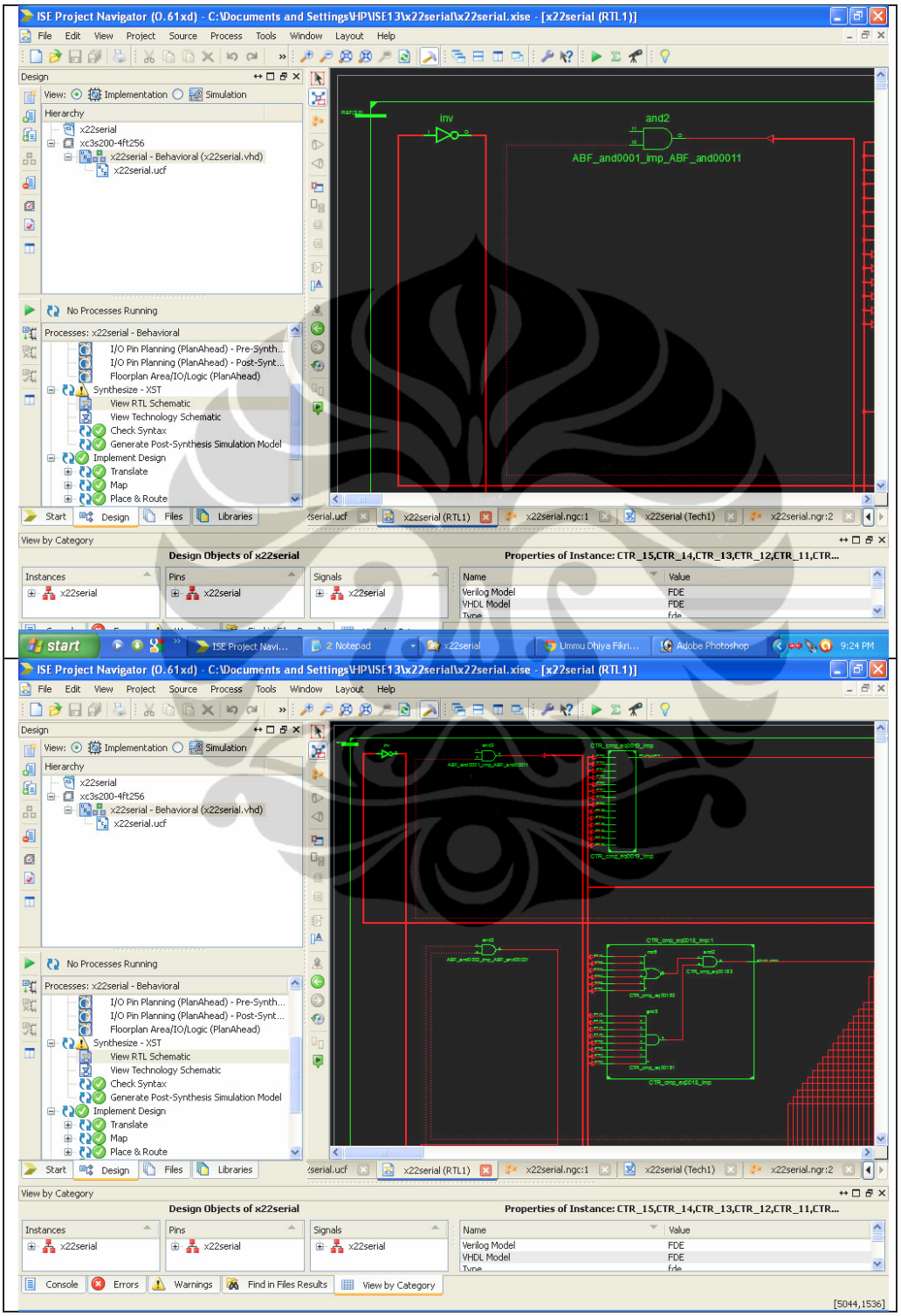
NETLIST DESIGN & ENTITY CPU OBU (dengan ISE 13.2)

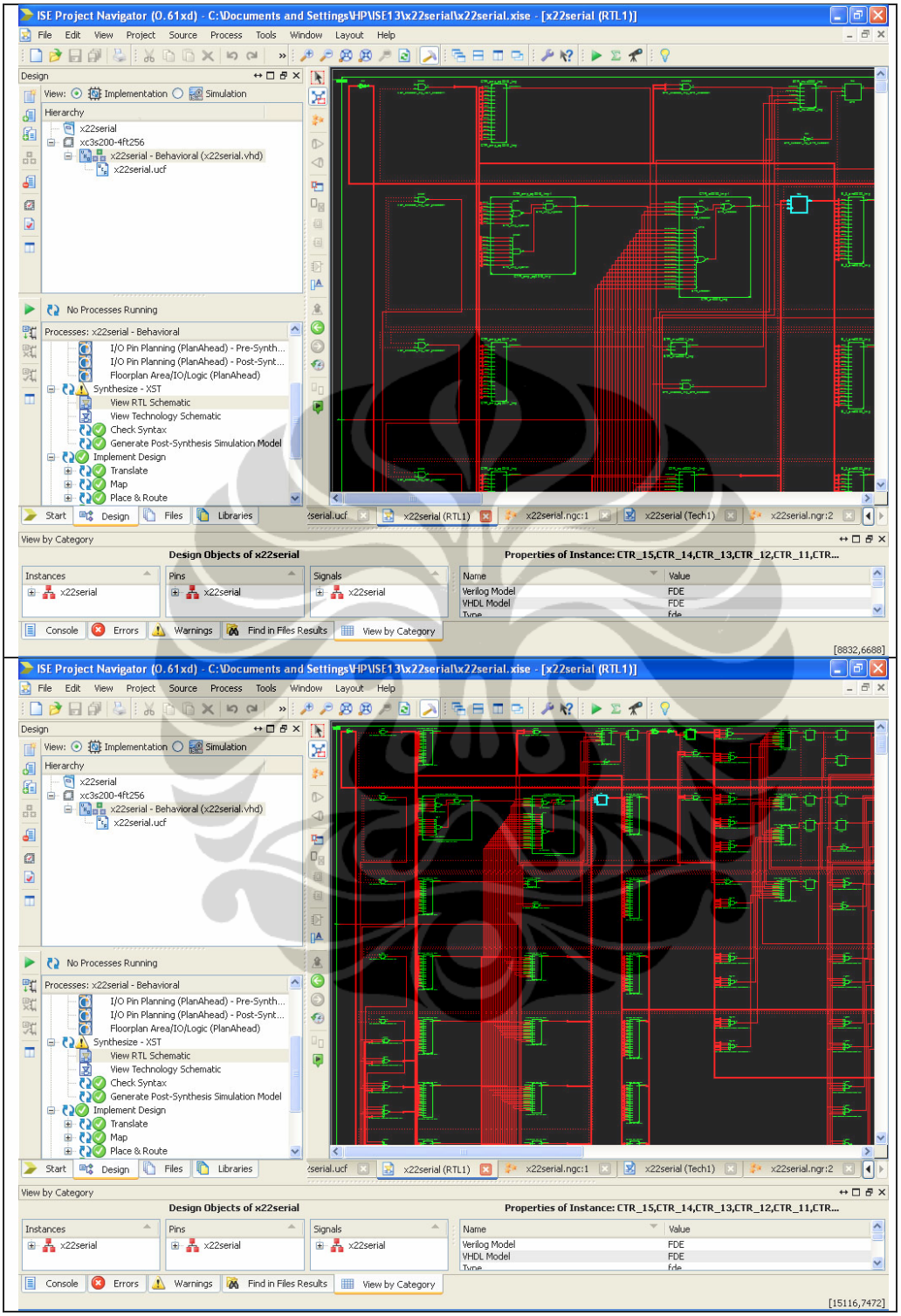


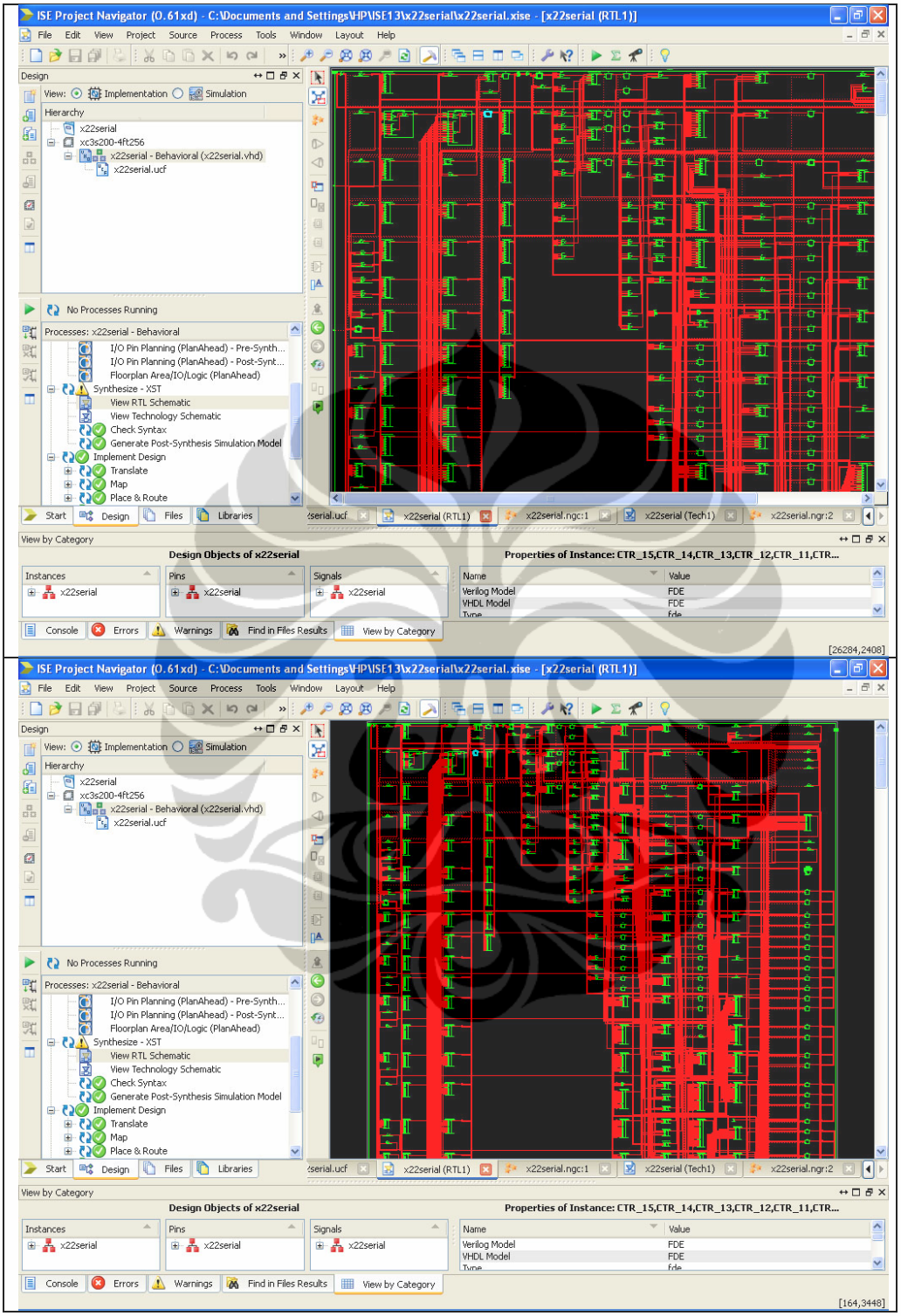
RTL SCHEMATIC CPU OBU (dengan ISE 13.2) COMPLATE

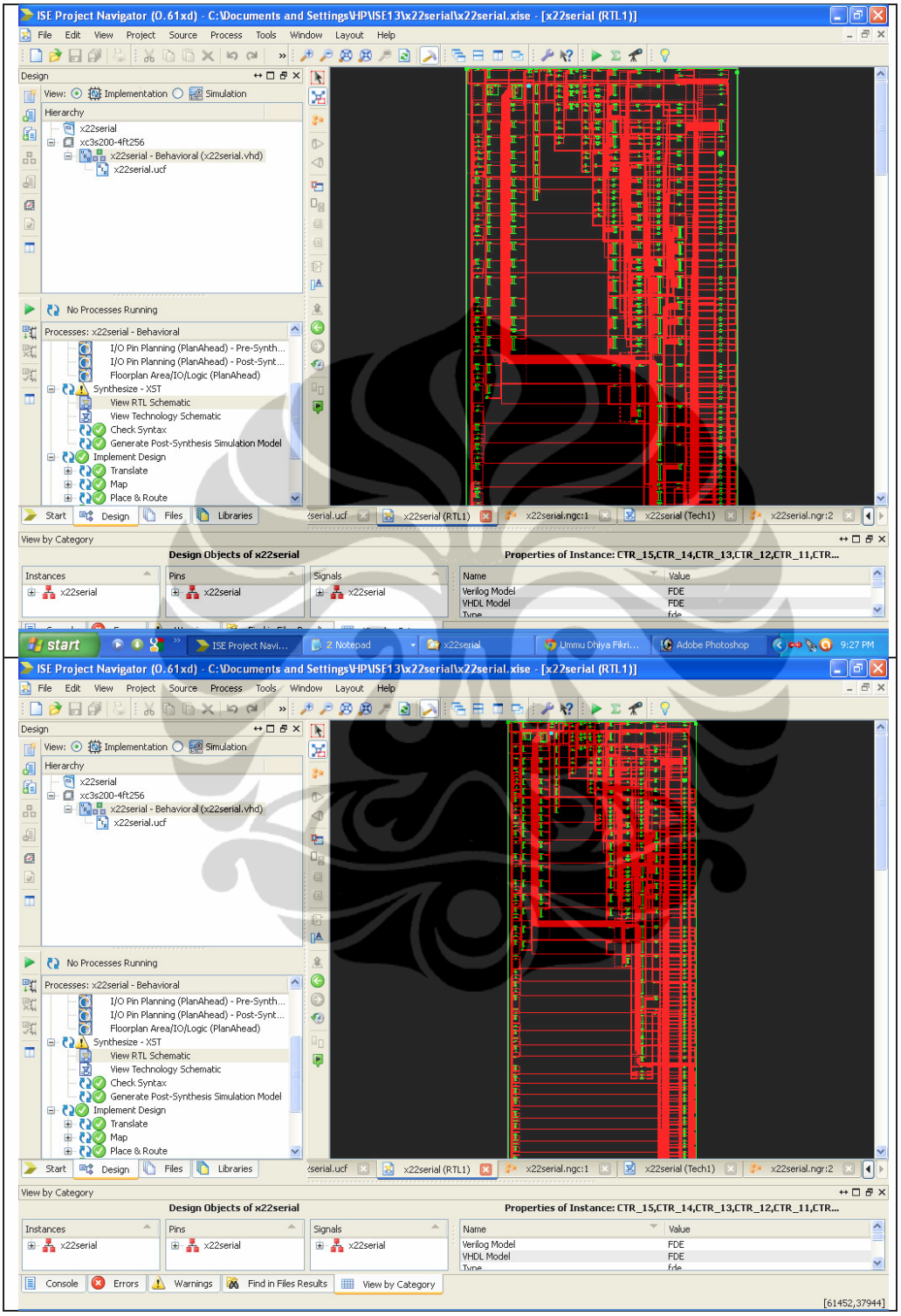


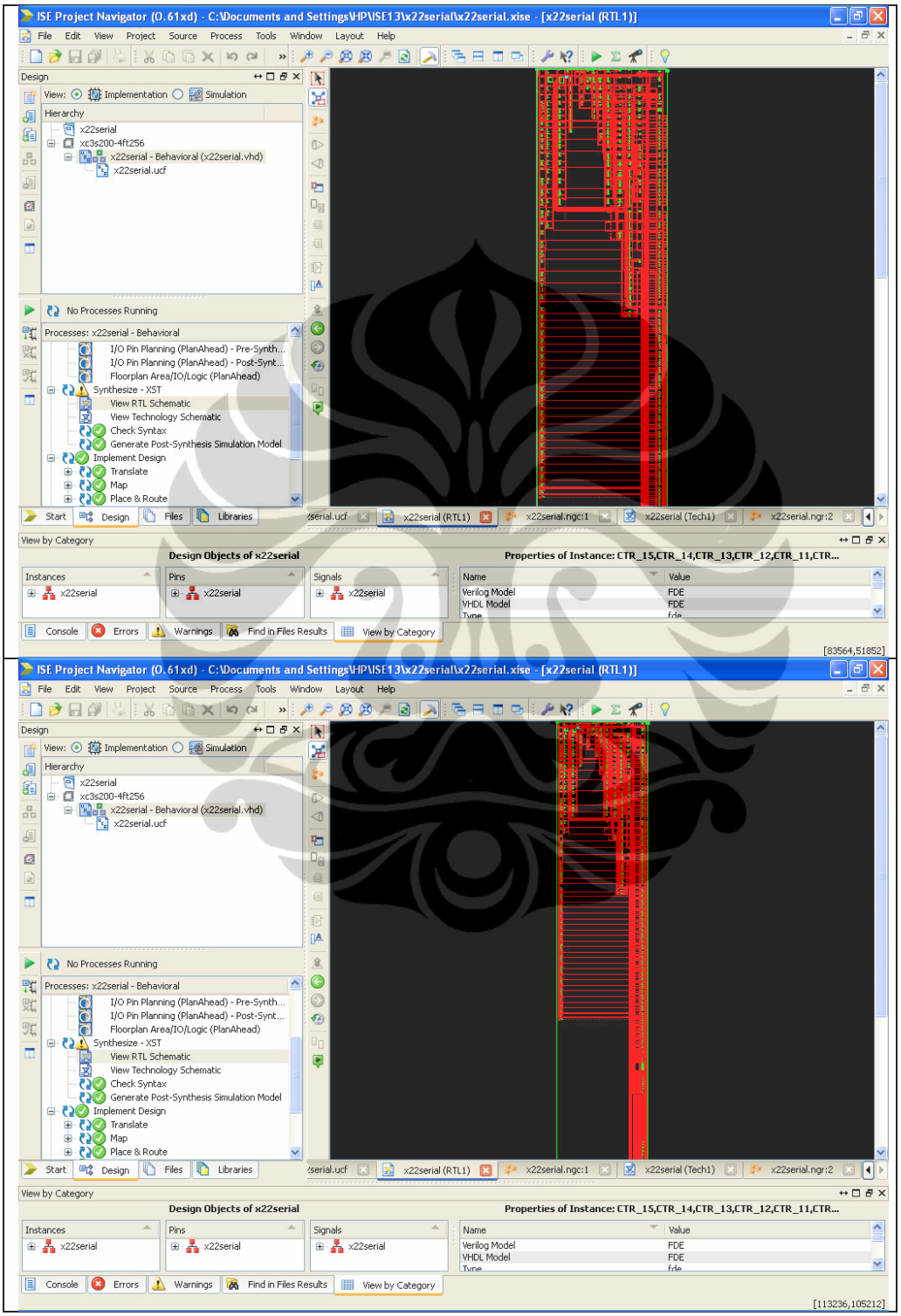
ZOOMING . . .


ISE Project Navigator (O.61xd) - C:\Documents and Settings\HP\ISE13\x22serial\x22serial.xise - [x22serial (RTL1)]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
x22serial
xc3s200-4ft256
x22serial - Behavioral (x22serial.vhd)
x22serial.ucf
No Processes Running
Processes: x22serial - Behavioral
I/O Pin Planning (PlanAhead) - Pre-Synth...
I/O Pin Planning (PlanAhead) - Post-Synt...
Floorplan Area/IO/Logic (PlanAhead)
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation Model
Implement Design
Translate
Map
Place & Route
Start Design Files Libraries
inv
and2
ABF_and0001_lmp_ABF_and00011
x22serial (RTL1)
x22serial.ngc:1
x22serial (Tech1)
x22serial.ngr:2
View by Category
Design Objects of x22serial
Properties of Instance: CTR_15,CTR_14,CTR_13,CTR_12,CTR_11,CTR...
Instances
Pins
Signals
x22serial
Name
Value
Verilog Model
VHDL Model
FDE
start
ISE Project Navi...
2 Notepad
x22serial
Ummu Dhiya Fikri...
Adobe Photoshop
9:24 PM
Console
Errors
Warnings
Find in Files Results
View by Category
[5044,1536]


ISE Project Navigator (O.61xd) - C:\Documents and Settings\HP\ISE13\x22serial\x22serial.xise - [x22serial (RTL1)]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
x22serial
xc3s200-4ft256
x22serial - Behavioral (x22serial.vhd)
x22serial.ucf
No Processes Running
Processes: x22serial - Behavioral
I/O Pin Planning (PlanAhead) - Pre-Synth...
I/O Pin Planning (PlanAhead) - Post-Synt...
Floorplan Area/IO/Logic (PlanAhead)
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation Model
Implement Design
Translate
Map
Place & Route
Start Design Files Libraries
x22serial (RTL1)
x22serial.ngc:1
x22serial (Tech1)
x22serial.ngr:2
View by Category
Design Objects of x22serial
Properties of Instance: CTR_15,CTR_14,CTR_13,CTR_12,CTR_11,CTR...
Instances
Pins
Signals
Name
Value
Verilog Model
VHDL Model
FDE
Console Errors Warnings Find in Files Results View by Category
[8832,6688]
ISE Project Navigator (O.61xd) - C:\Documents and Settings\HP\ISE13\x22serial\x22serial.xise - [x22serial (RTL1)]
[15116,7472]


ISE Project Navigator (O.61xd) - C:\Documents and Settings\HP\ISE13\x22serial\x22serial.xise - [x22serial (RTL1)]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
x22serial
xc3s200-4ft256
x22serial - Behavioral (x22serial.vhd)
x22serial.ucf
No Processes Running
Processes: x22serial - Behavioral
I/O Pin Planning (PlanAhead) - Pre-Synth...
I/O Pin Planning (PlanAhead) - Post-Synt...
Floorplan Area/IO/Logic (PlanAhead)
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation Model
Implement Design
Translate
Map
Place & Route
Start Design Files Libraries
x22serial (RTL1)
x22serial.ngc:1
x22serial (Tech1)
x22serial.ngr:2
View by Category
Design Objects of x22serial
Properties of Instance: CTR_15,CTR_14,CTR_13,CTR_12,CTR_11,CTR...
Instances
Pins
Signals
Name
Value
Verilog Model
VHDL Model
FDE
Console Errors Warnings Find in Files Results View by Category
[26284,2408]
ISE Project Navigator (O.61xd) - C:\Documents and Settings\HP\ISE13\x22serial\x22serial.xise - [x22serial (RTL1)]
[164,3448]


ISE Project Navigator (O.61xd) - C:\Documents and Settings\HP\ISE13\x22serial\x22serial.xise - [x22serial (RTL1)]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
x22serial
xc3s200-4ft256
x22serial - Behavioral (x22serial.vhd)
x22serial.ucf
No Processes Running
Processes: x22serial - Behavioral
I/O Pin Planning (PlanAhead) - Pre-Synth...
I/O Pin Planning (PlanAhead) - Post-Synt...
Floorplan Area/IO/Logic (PlanAhead)
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation Model
Implement Design
Translate
Map
Place & Route
Start Design Files Libraries
x22serial.ucf x22serial (RTL1) x22serial.ngc:1 x22serial (Tech1) x22serial.ngr:2
View by Category
Design Objects of x22serial
Properties of Instance: CTR_15,CTR_14,CTR_13,CTR_12,CTR_11,CTR...
Instances
Pins
Signals
x22serial
Name
Value
Verilog Model
VHDL Model
FDE
start
ISE Project Navi...
2 Notepad
x22serial
Ummu Dhiya Fikri...
Adobe Photoshop
9:27 PM
Console Errors Warnings Find in Files Results View by Category
[61452,37944]


ISE Project Navigator (O.61xd) - C:\Documents and Settings\HP\ISE13\x22serial\x22serial.xise - [x22serial (RTL1)]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
x22serial
xc3s200-4ft256
x22serial - Behavioral (x22serial.vhd)
x22serial.ucf
No Processes Running
Processes: x22serial - Behavioral
I/O Pin Planning (PlanAhead) - Pre-Synth...
I/O Pin Planning (PlanAhead) - Post-Synt...
Floorplan Area/IO/Logic (PlanAhead)
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation Model
Implement Design
Translate
Map
Place & Route
Start Design Files Libraries
x22serial.ucf x22serial (RTL1) x22serial.ngc:1 x22serial (Tech1) x22serial.ngr:2
View by Category
Design Objects of x22serial
Properties of Instance: CTR_15,CTR_14,CTR_13,CTR_12,CTR_11,CTR...
Instances
Pins
Signals
x22serial
Name
Value
Verilog Model
VHDL Model
FDE
Console Errors Warnings Find in Files Results View by Category
[83564,51852]
ISE Project Navigator (O.61xd) - C:\Documents and Settings\HP\ISE13\x22serial\x22serial.xise - [x22serial (RTL1)]
[113236,105212]