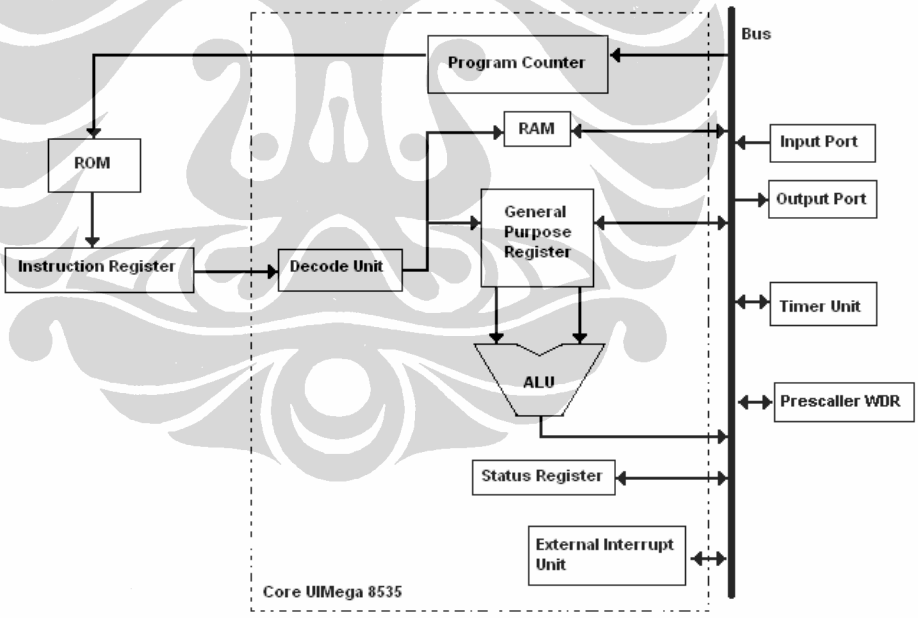
BAB III
PERANCANGAN UIMEGA 8535

3.1 ARSITEKTUR UIMEGA 8535

Arsitektur UIMega 8535 secara umum diperlihatkan pada Gambar 3.1. UIMega 8535 terdiri dari lima modul utama, yaitu modul ROM, modul *instruction register*, modul *core* UIMega 8535, modul *timer interrupt*, dan modul *prescallerWDR*. Modul *core* UIMega 8535 terdiri dari unit *decode*, unit RAM, unit *General Purpose Register*, *Arithmatic and Logic Unit* (ALU), unit *status register*, unit *program counter*, *stack*, dan unit *External Interrupt*. Masing-masing unit dibuat terpisah secara program dan dihubungkan dengan sinyal-sinyal (*wire*). *Bus* pada Gambar 3.1 merupakan kumpulan *wire* yang menghubungkan antar unit dan saling terpisah antara satu dengan yang lain.



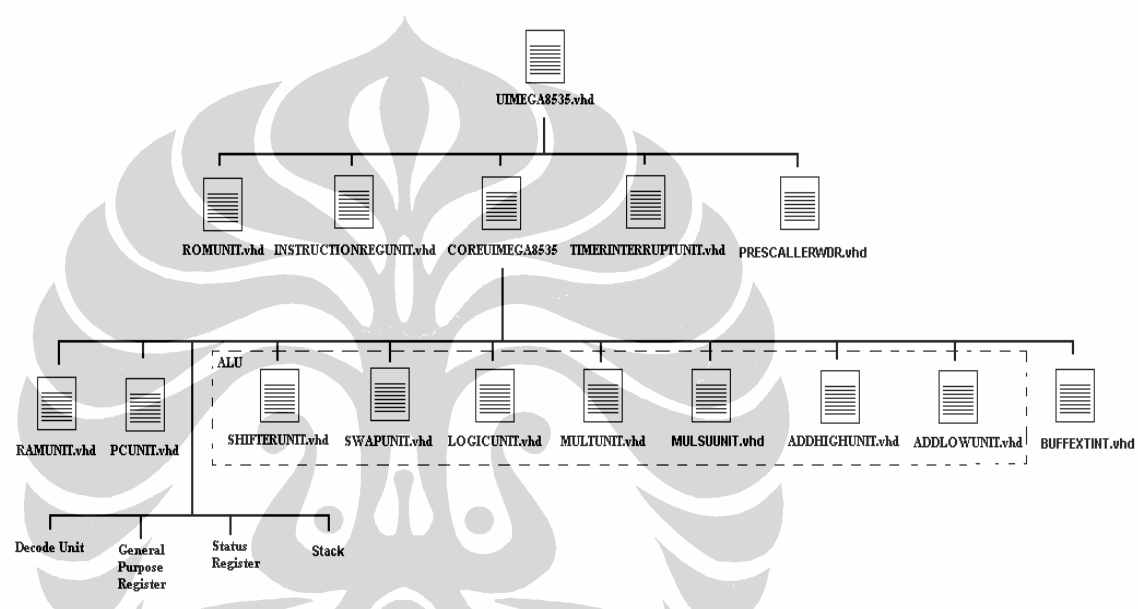
Gambar 3.1 Arsitektur UIMega 8535

3.2 MODUL-MODUL UIMEGA 8535

Hubungan antar modul-modul yang membentuk UIMega 8535 diperlihatkan pada gambar di Lampiran 1.

Hubungan antar *file-file* VHDL diperlihatkan pada Gambar 3.2. Pada tingkat tertinggi adalah *UIMEGA8535.vhd*, dimana *file* ini yang menghubungkan semua *file-file* modul, seperti :

1. *ROMUNIT.vhd*
2. *INSTRUCTIONREGUNIT.vhd*
3. *COREMEGA8535.vhd*
4. *TIMERINTERRUPTUNIT.vhd*
5. *PRESCALLERWDR.vhd*



Gambar 3.2 Hubungan antar *file-file* VHDL

File COREMEGA8535.vhd menghubungkan beberapa *file* unit, seperti :

1. *RAMUNIT.vhd*
2. ALU yang terbentuk dari gabungan *file SHIFTERUNIT.vhd, SWAPUNIT.vhd, LOGICUNIT.vhd, MULTUNIT.vhd, MULSUUNIT.vhd, ADDHIGHUNIT.vhd,* dan *ADDLOWUNIT.vhd*
3. *PCUNIT.vhd*
4. *BUFFEXTINT.vhd*

Beberapa unit terintegrasi langsung di dalam *file COREMEGA8535.vhd*, unit tersebut adalah *General Purpose Register, status register, stack,* dan *decode unit*.

3.2.1 Modul ROM

Modul ROM akan menyimpan program yang akan dijalankan. Program ini disimpan dalam bentuk ekstensi *.hex*. Modul ini mempunyai masukan *pin_pc* yang akan menunjukkan alamat instruksi yang tersimpan di modul ROM dan keluaran *pin_instrom* yang berisi instruksi yang disimpan. Instruksi yang disimpan mempunyai lebar 16-*bit*. Simbol modul ROM diperlihatkan pada Gambar 3.3. Jenis *port*, fungsi, dan konektivitas unit ROM diperlihatkan pada Tabel 3.1. *Source code* modul ini disimpan dalam *file ROMUnit.vhd* pada Lampiran 5.

Gambar 3.3 Modul ROM

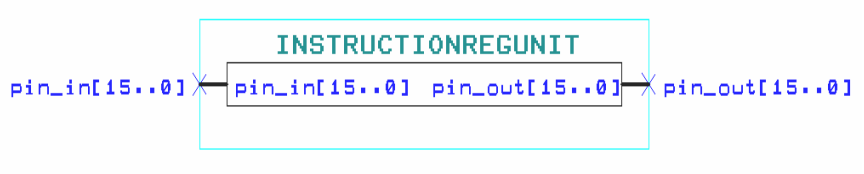
Tabel 3.1 Jenis, Fungsi, dan Konektivitas *Port* Modul ROM

No	Nama *Port*	Jenis *Port*	Fungsi	Terhubung Ke
1	*pin_pc*	masukan	berisi alamat di mana instruksi disimpan di ROM	modul *core* UIMega 8535
2	*pin_datain*	masukan	berisi data yang akan disimpan ke ROM	modul *core* UIMega 8535
3	*pin_werom*	masukan	sinyal penulisan ke ROM	modul *core* UIMega 8535
4	*pin_instrom*	keluaran	berisi instruksi yang dipanggil	modul *instruction register*

3.2.2 Modul *Instruction Register*

Modul *instruction register* berfungsi untuk menyimpan instruksi yang akan dieksekusi oleh modul *core UIMega 8535*. Modul ini juga akan melakukan pembalikan *nibble* dari instruksi. Hal ini ditujukan untuk mempermudah pengkodean dari instruksi oleh *decode unit* di dalam modul *core UIMega 8535*. Acuan pengkodean dapat dilihat pada Lampiran 2. Simbol dari modul *instruction register* diperlihatkan pada Gambar 3.4. Jenis *port*, fungsi, dan konektivitas unit

instruction register diperlihatkan pada Tabel 3.2. *Source code* modul ini disimpan dalam *file InstructionRegUnit.vhd* pada Lampiran 5.



Gambar 3.4 Modul *instruction register*

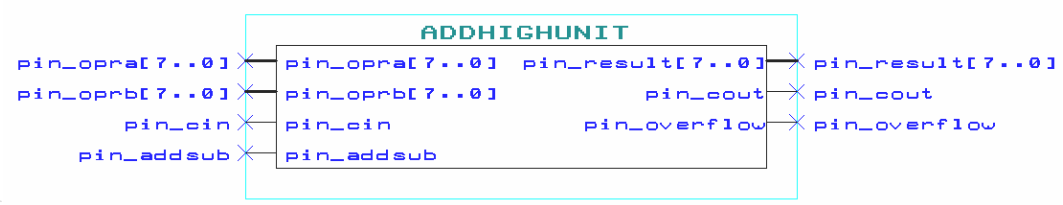
Tabel 3.2 Jenis, Fungsi, dan Konektivitas *Port* Modul *Instruction Register*

No	Nama *Port*	Jenis *Port*	Fungsi	Terhubung Ke
1	*pin_in*	masukan	berisi instruksi yang dipanggil dari ROM	modul ROM
2	*pin_out*	keluaran	berisi instruksi yang dipanggil dan telah dibalik secara *nibble*	modul *core UIMega 8535*

3.2.3 Modul *Core UIMega 8535*

Modul *core UIMega 8535* terdiri dari beberapa unit yaitu unit *decode*, ALU, *General Purpose Register*, RAM, *program counter*, *status register*, *external interrupt*, dan *stack*. Unit *decode* berfungsi untuk melakukan pengkodean terhadap instruksi yang berasal dari ROM untuk kemudian menentukan apa yang selanjutnya harus dilakukan. Unit ini terintegrasi langsung di dalam modul *core* UIMega 8535. Unit ALU terdiri dari beberapa sub-unit, yaitu *adder-subracter*, *multiplier*, *logic*, *swapunit*, dan *shifter*.

Sub-unit *adder-subracter* akan menangani operasi penjumlahan dan pengurangan. Sub-unit ini dibagi menjadi dua buah yaitu sub-unit yang akan menangani pola instruksi 16-*bit* dan 32-*bit* namun masukan dan keluaran dari sub-unit ini sama. Simbol sub-unit ini diperlihatkan pada Gambar 3.5. Jenis *port* dan fungsi sub-unit *adder-subracter* diperlihatkan pada Tabel 3.3. *Source code* modul ini disimpan dalam *file AddLowUnit.vhd* dan *AddHighUnit.vhd* pada Lampiran 5.

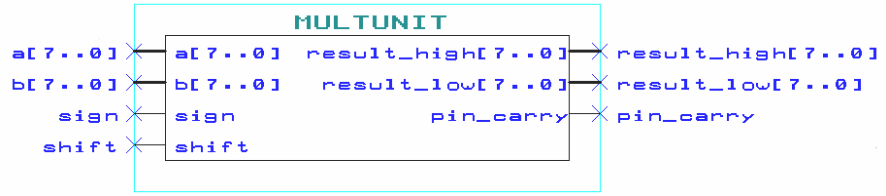


Gambar 3.5 Sub-unit *adder-subracter*

Tabel 3.3 Jenis dan Fungsi *Port* Sub-Unit *Adder-Subracter*

No	Nama *Port*	Jenis *Port*	Fungsi
1	*pin_opra*	masukan	*operand* 1
2	*pin_oprb*	masukan	*operand* 2
3	*pin_cin*	masukan	*carry in*
4	*pin_addsub*	masukan	menentukan operasi penjumlahan (logika ’1’) atau pengurangan (logika ’0’)
5	*pin_result*	keluaran	hasil operasi
6	*pin_cout*	keluaran	*carry out*
7	*pin_overflow*	keluaran	*overflow out*

Sub-unit *multiplier* menangani operasi perkalian. Sub-unit ini terbagi dua, yaitu sub-unit *multunit* yang menangani operasi perkalian bertanda dan sub-unit *mulsuunit* yang menangani operasi perkalian tidak bertanda. Simbol sub-unit *multunit* diperlihatkan pada Gambar 3.6. Jenis *port* dan fungsi sub-unit *multiplier* diperlihatkan pada Tabel 3.4. *Source code* modul ini disimpan dalam *file MultUnit.vhd* pada Lampiran 5.

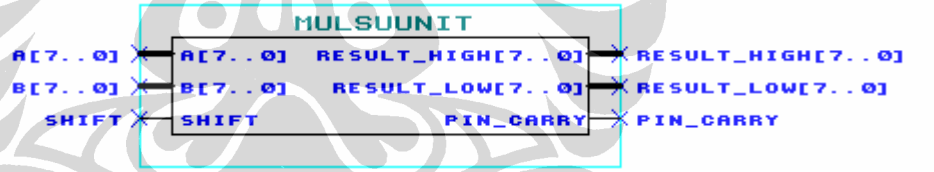


Gambar 3.6 Sub-unit *multunit*

Tabel 3.4 Jenis dan Fungsi *Port* Sub-Unit *Multunit*

No	Nama *Port*	Jenis *Port*	Fungsi
1	*a*	masukan	*operand* 1 (8-*bit*)
2	*b*	masukan	*operand* 2 (8-*bit*)
3	*sign*	masukan	menentukan operasi bertanda (logika ‘1’) atau tidak (logika ‘0’)
4	*shift*	masukan	menentukan operasi membutuhkan pergeseran (logika ’1’) atau tidak (logika ’0’).
5	*result_high*	keluaran	hasil operasi (*bit* ke15 sampai ke 8)
6	*result_low*	keluaran	hasil operasi (*bit* ke 7 sampai ke 0)
7	*pin_carry*	keluaran	*carry out*

Simbol sub-unit *mulsunit* diperlihatkan pada Gambar 3.7. Jenis *port* dan fungsi sub-unit *mulsuunit* diperlihatkan pada Tabel 3.5. *Source code* modul ini disimpan dalam *file MulsuUnit.vhd* pada Lampiran 5.

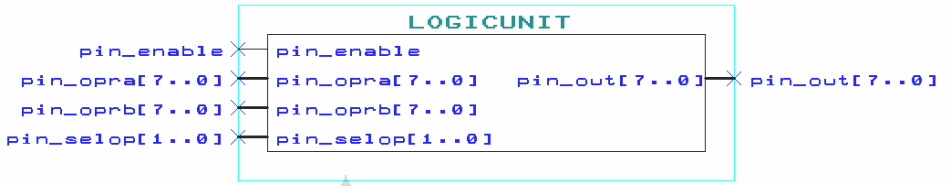


Gambar 3.7 Sub-unit *mulsuunit*

Tabel 3.5 Jenis dan Fungsi *Port* Sub-Unit *Mulsuunit*

No	Nama *Port*	Jenis *Port*	Fungsi
1	*a*	masukan	*operand* 1 (8-*bit*)
2	*b*	masukan	*operand* 2 (8-*bit*)
3	*shift*	masukan	menentukan operasi membutuhkan pergeseran (logika ’1’) atau tidak (logika ’0’).
4	*result_high*	keluaran	hasil operasi (*bit* ke15 sampai ke 8)
5	*result_low*	keluaran	hasil operasi (*bit* ke 7 sampai ke 0)
6	*pin_carry*	keluaran	*carry out*

Sub-unit *logic* akan menangani operasi logika seperti *and, or, ex-or,* dan komplemen. Simbol sub-unit ini diperlihatkan pada Gambar 3.8. Jenis instruksi yang ditangani berdasarkan masukan *pin_selop* diperlihatkan pada Tabel 3.6. Jenis *port* dan fungsi sub-unit *logic* diperlihatkan pada Tabel 3.7. *Source code* modul ini disimpan dalam *file LogicUnit.vhd* pada Lampiran 5.



Gambar 3.8 Sub-unit *logic*

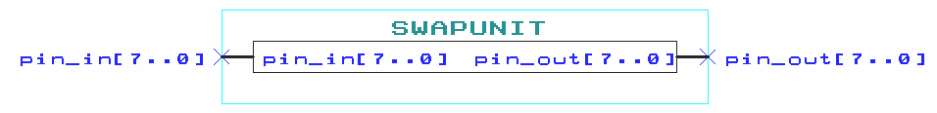
Tabel 3.6 Jenis Operasi Berdasar Masukan *Pin_selop*

Masukan pin_selop	Jenis instruksi
0	AND, ANDI
1	OR, ORI
2	EOR
3	COM

Tabel 3.7 Jenis dan Fungsi *Port* Sub-Unit *Logic*

No	Nama *Port*	Jenis *Port*	Fungsi
1	*pin_enable*	masukan	untuk memfungsikan sub-unit *logic*
2	*pin_opra*	masukan	*operand* 1
3	*pin_oprb*	masukan	*operand* 2
4	*pin_selop*	masukan	untuk menentukan operasi yang dilakukan
5	*pin_out*	keluaran	hasil operasi

Sub-unit *swapunit* khusus menangani instruksi SWAP yang akan membalik *nibble* dari masukan yang diberikan. Simbol sub-unit ini diperlihatkan pada Gambar 3.9. Jenis *port* dan fungsi sub-unit *swapunit* diperlihatkan pada Tabel 3.8. *Source code* modul ini disimpan dalam *file SwapUnit.vhd* pada Lampiran 5.

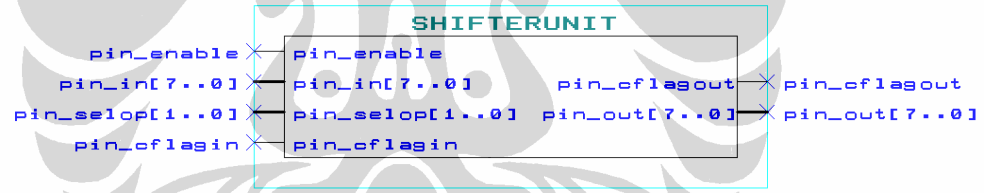


Gambar 3.9 Sub-unit *swapunit*

Tabel 3.8 Jenis dan Fungsi *Port* Sub-Unit *Swapunit*

No	Nama *Port*	Jenis *Port*	Fungsi
1	*pin_in*	masukan	*operand*
2	*pin_out*	keluaran	hasil operasi

Sub-unit ***shifter*** khusus menangani instruksi pergeseran *bit*. Simbol sub-unit ini diperlihatkan pada Gambar 3.10. Jenis instruksi yang ditangani berdasar masukan *pin_selop* diperlihatkan pada Tabel 3.9. Jenis *port* dan fungsi sub-unit *shifter* diperlihatkan pada Tabel 3.10. *Source code* modul ini disimpan dalam *file ShifterUnit.vhd* pada Lampiran 5.



Gambar 3.10 Sub-unit ***shifter***

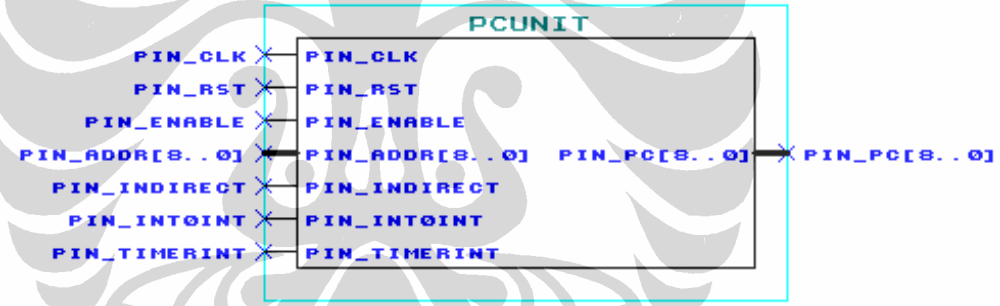
Tabel 3.9 Jenis Operasi Berdasar Masukan *Pin_selop*

Masukan pin_selop	Jenis instruksi
0	LSR
1	ROR
2	ASR

Tabel 3.10 Jenis dan Fungsi *Port* Sub-Unit *Shifter*

No	Nama *Port*	Jenis *Port*	Fungsi
1	*pin_enable*	masukan	untuk memfungsikan sub-unit *shifer*
2	*pin_in*	masukan	*operand*
3	*pin_selop*	masukan	untuk menentukan operasi yang dilakukan
4	*pin_cflagin*	masukan	*flag carry in*
5	*pin_out*	keluaran	hasil operasi
6	*pin_cfagout*		*flag carry out*

Unit *program counter* berfungsi untuk menunjuk alamat instruksi yang tersimpan di unit ROM. Unit ini mampu melakukan pengalamatan sebesar 2^9 *bit* lokasi atau 512 lokasi modul ROM. Simbol unit *program counter* diperlihatkan pada Gambar 3.11. *Source code* modul ini disimpan dalam *file PCUnit.vhd* pada Lampiran 5.



Gambar 3.11 Unit *Program Counter*

Unit ini memiliki masukan *pin_clk, pin_rst, pin_enable, pin_indirect, pin_addr, pin_int0int,* dan *pin_timerint.* Unit ini hanya akan bekerja bila *pin_enable* diberi masukan logika '1'. *Pin_indirect* akan menunjukkan pada unit *program counter* untuk mengambil alamat baru dari pin_*addr*. Masukan *pin_int0int* dan *pin_timerint* akan mengindikasikan adanya *interrupt* sehingga *program counter* perlu melaksanakan rutin *interrupt*. *Pin_int0int* akan mengindikasikan unit *program counter* bahwa terjadi *external interrupt* sedangkan *pin_timerint* mengindikasikan unit *program counter* bahwa terjadi *timer interrupt*. Bilamana terjadi *interrupt* maka alamat eksekusi program yang sedang berlangsung akan disimpan ke dalam *stack* sehingga setelah dilakukan

pemrosesan rutin *interrupt*, eksekusi program dapat dilakukan pada alamat program sebelum dilakukan *interrupt*.

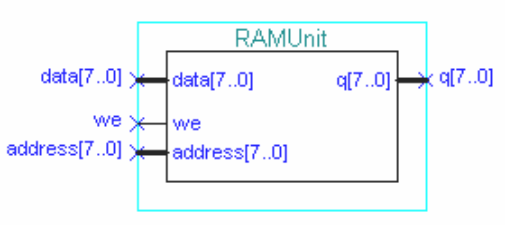
Keluaran *pin_pc* akan menunjukkan alamat instruksi yang akan diambil dari modul ROM. Jenis *port*, fungsi, dan konektivitas unit *program counter* diperlihatkan pada Tabel 3.11.

Tabel 3.11 Jenis, Fungsi, dan Konektivitas *Port Program Counter*

No	Nama *Port*	Jenis *Port*	Fungsi
1	*pin_clk*	masukan	sistem *clock*
2	*pin_rst*	masukan	sistem *reset*
3	*pin_enable*	masukan	untuk memfungsikan unit *program counter*
4	*pin_indirect*	masukan	untuk mengindikasikan *program counter* untuk mengambil alamat dari *pin_addr*
5	*pin_addr*	masukan	alamat percabangan
6	*pin_int0int*	masukan	mengindikasikan terjadi *external interrupt*
7	*pin_timerint*	masukan	mengindikasikan terjadi *timer interrupt*
8	*pin_pc*	keluaran	menunjukkan alamat instruksi di ROM

Unit RAM berfungsi sebagai *data space* atau tempat penyimpanan data bagi mikrokontroler. Unit ini memiliki masukan *data, we,* dan *address.* Keluaran dari unit ini adalah *q*. Masukan dari *address* akan mengalamatkan RAM sehingga isi dari RAM akan langsung dikeluarkan pada keluaran *q.* Proses penulisan dilakukan dengan mengaktifkan masukan *we* dan pemberian data yang yang hendak ditulis pada masukan *data*.

Simbol unit ini diperlihatkan pada Gambar 3.12. Jenis *port* dan fungsi unit RAM diperlihatkan pada Tabel 3.12. *Source code* modul ini disimpan dalam *file RAMUnit.vhd* pada Lampiran 5.



Gambar 3.12 Unit RAM

Tabel 3.12 Jenis, Fungsi, dan Konektivitas *Port* RAM

No	Nama *Port*	Jenis *Port*	Fungsi
1	*address*	masukan	alamat RAM yang hendak dituju
2	*data*	masukan	masukan data yang hendak ditulis pada alamat tertentu dari RAM
3	*we*	masukan	untuk mengaktifkan penulisan pada RAM
4	*q*	keluaran	mengeluarkan isi RAM berdasarkan alamat yang diberikan di masukan *address*

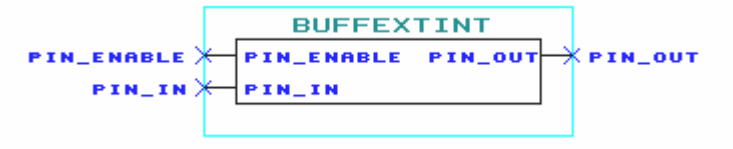
Unit *buffextint* berfungsi untuk mendeteksi *interrupt* dari luar. Unit ini memiliki masukan *pin_enable* dan *pin_in*. Keluaran dari unit ini adalah *pin_out*. Unit ini hanya akan bekerja bila masukan *pin_enable* diaktifkan dari modul *core UIMega 8535*. Masukan *pin_in* dihubungkan ke masukan pin ke-7 dari port A mikrokontroler. Bila unit ini diaktifkan, maka saat terjadi logika *high* pada pin ke-7 dari port A mikrokontroler akan dideteksi sebagai *interrupt*.

Perancangan ini hanya memakai register *General Interrupt Control Register* (GICR) untuk pengaktifan unit ini dengan alamat $3B. Perancangan ini hanya memakai *bit Interrupt Request 0*, yaitu *bit* ke-6 (INT0). Eksternal *interrupt* akan aktif bilamana bernilai logika ’1’ dan nilai *flag* I pada *status register* juga harus berlogika ’1’. Susunan register ini dapat dilihat pada Gambar 3.13.

Bit	7	6	5	4	3	2	1	0	
	INT1	INT0	INT2	–	–	–	IVSEL	IVCE	GICR
Read/Write	R/W	R/W	R/W	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Gambar 3.13 *General Interrupt Control Register* (GICR) [5]

Simbol unit ini diperlihatkan pada Gambar 3.14. Jenis *port* dan fungsi unit *buffextint* diperlihatkan pada Tabel 3.13. *Source code* modul ini disimpan dalam *file buffExtInt.vhd* pada Lampiran 5.

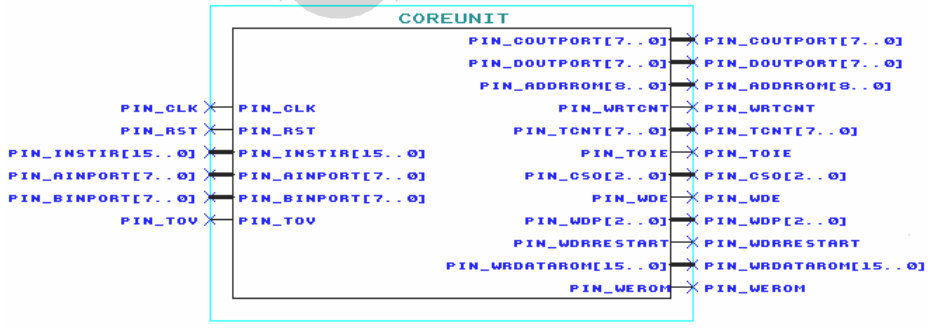


Gambar 3.14 Unit *buffextint*

Tabel 3.13 Jenis, Fungsi, dan Konektivitas *Port Buffextint*

No	Nama *Port*	Jenis *Port*	Fungsi
1	*pin_enable*	masukan	untuk mengaktifkan unit
2	*pin_in*	masukan	untuk mendeteksi kondisi *high* dari pin ke-7 port A
3	*pin_out*	keluaran	untuk mensinyalkan terjadi *external interrupt*

Secara keseluruhan simbol dari modul *core* UIMega 8535 diperlihatkan pada Gambar 3.15. Jenis *port*, fungsi, dan konektivitas modul *core* UIMega 8535 diperlihatkan pada Tabel 3.14. *Source code* modul ini disimpan dalam *file CoreUnit.vhd* pada Lampiran 5.



Gambar 3.15 Modul *core UIMega 8535*

Tabel 3.14 Jenis, Fungsi, dan Konektivitas *Port* Modul *Core* UIMega 8535

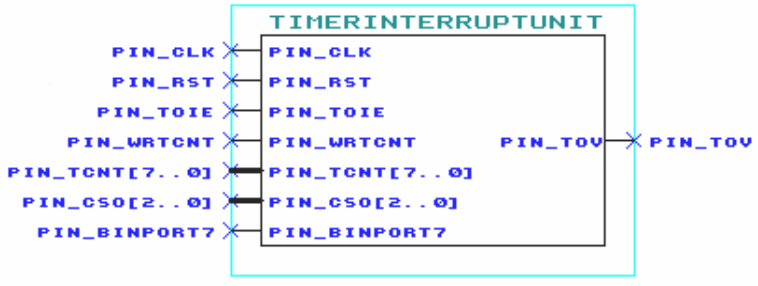
No	Nama *Port*	Jenis *Port*	Fungsi	Terhubung Ke
1	*pin_clk*	masukan	sistem *clock*	sistem *clock* utama
2	*pin_rst*	masukan	sistem *reset*	sistem *reset* utama
3	*pin_instir*	masukan	menerima instruksi dari *instruction register*	modul *instruction register*
4	*pin_Ainport*	masukan	menerima masukan dari *port A*	*port A*
5	*pin_Binport*	masukan	menerima masukan dari *port B*	*port B*
6	*pin_tov*	masukan	menerima sinyal *overflow* dari modul *timer interrupt*	modul *timer interrupt*
7	*pin_Coutport*	keluaran	keluaran ke port C	*port C*
8	*pin_Doutport*	keluaran	keluaran ke port D	*port D*
9	*pin_addrom*	keluaran	keluaran alamat ROM	modul ROM
10	*pin_wrtcnt*	keluaran	sinyal *enable* untuk penulisan nilai awal *timer*	modul *timer interrupt*
11	*pin_tcnt*	keluaran	keluaran nilai awal *timer*	modul *timer interrupt*
12	*pin_toie*	keluaran	sinyal *enable* untuk modul *timer interrupt*	modul *timer interrupt*
13	*pin_cso*	keluaran	keluaran nilai *prescaller* untuk modul *timer interrupt*	modul *timer interrupt*

Tabel 3.14 Jenis, Fungsi, dan Konektivitas *Port* Modul *Core* UIMega 8535 (sambungan)

No	Nama *Port*	Jenis *Port*	Fungsi	Terhubung Ke
14	*pin_wde*	keluaran	sinyal *enable* untuk modul *prescaller watchdog timer*	modul *prescallerWDR*
15	*pin_wdp*	keluaran	keluaran nilai *prescaller* untuk modul *prescallerWDR*	modul *prescallerWDR*
16	*pin_wdrrestart*	keluaran	keluaran sinyal *restart* untuk modul *prescallerWDR*	modul *prescallerWDR*
17	*pin_wrdatarom*	keluaran	berisi data yang akan ditulis ke ROM	modul ROM
18	*pin_werom*	keluaran	sinyal penulisan ke ROM	modul ROM

3.2.4 Modul *Timer Interrupt*

Modul *timer interrupt* berfungsi sebagai *timer* yang dapat digunakan sebagai sumber *interrupt*. Simbol modul diperlihatkan pada Gambar 3.16. Jenis *port*, fungsi, dan konektivitas modul *timer interrupt* diperlihatkan pada Tabel 3.15. *Source code* modul ini disimpan dalam *file TimerInterruptUnit.vhd* pada Lampiran 5.



Gambar 3.16 Modul *timer interrupt*

Tabel 3.15 Jenis, Fungsi, dan Konektivitas *Port* Modul *Timer Interrupt*

No	Nama *Port*	Jenis *Port*	Fungsi	Terhubung Ke
1	*pin_clk*	masukan	sebagai sumber sinyal *clock*	sistem *clock* utama
2	*pin_rst*	masukan	sebagai sumber sinyal *reset*	sistem *reset* utama
3	*pin_toie*	masukan	untuk mengaktifkan unit	modul *core UIMega 8535*
4	*pin_wrtcnt*	masukan	untuk mengaktifkan penulisan nilai awal *timer*	modul *core UIMega 8535*
5	*pin_tcnt*	masukan	masukan nilai awal *timer*	modul *core UIMega 8535*
6	*pin_cso*	masukan	untuk memiilih *clock source*	modul *core UIMega 8535*
7	*pin_Binport7*	masukan	*external source*	*port* B pin 7
8	*pin_tov*	keluaran	sebagai pensinyalan bahwa telah terjadi *overflow*	modul *core UIMega 8535*

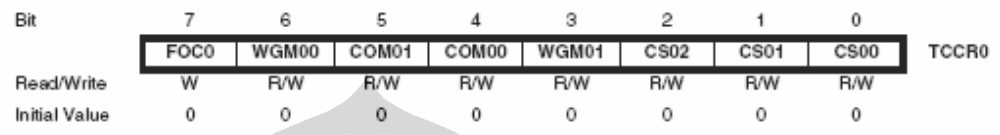
Perancangan ini menggunakan beberapa *register* untuk menjalankan fungsi dari *timer*. *Register-register* tersebut adalah sebagai berikut:

1. *Timer/Counter Interrupt Mask Register* (TIMSK). Register ini memiliki alamat $39. Susunan *bit* pada *register* ini diperlihatkan pada Gambar 3.17. *Bit* ke-0 yaitu *Timer0/Counter0 Compare Match Interrupt Enable* (TOIE0) harus bernilai logika ’1’ dan nilai *flag* I pada *status register* juga harus berlogika ’1’ untuk mengaktifkan *timer interrupt*. *Bit-bit* lain tidak dipakai dalam perancangan ini.

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Gambar 3.17 *Timer/Counter Interrupt Mask Register* (TIMSK) [5]

2. 8-*bit Timer0/Counter0 Register Description* (TCCR0). Register ini memiliki alamat $33. Susunan *bit* pada *register* ini diperlihatkan pada Gambar 3.18. Perancangan ini menggunakan *bit Clock Select*, yaitu *bit* ke-2 (CSO2), ke-1 (CSO1), dan ke-0 (CSO0). Kombinasi dari *bit-bit* ini akan menentukan sumber sinyal *clock* untuk *timer* dan diperlihatkan pada Tabel 3.16.

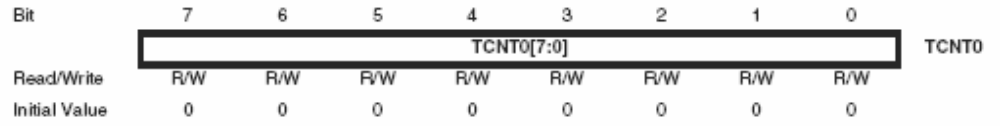


Gambar 3.18 8-*bit Timer/Counter Register Description* [5]

Tabel 3.16 Sumber *Clock Timer* Berdasarkan TCCR0

CSO2	**CSO1**	**CSO0**	**Sumber *clock timer***
0	0	0	*timer* tidak berfungsi
0	0	1	sistem *clock* utama
0	1	0	sistem *clock* utama / 8
0	1	1	sistem *clock* utama / 64
1	0	0	sistem *clock* utama /256
1	0	1	sistem *clock* utama / 1024
1	1	0	*port* eksternal, tepian sinyal turun
1	1	1	*port* eksternal, tepian sinyal naik

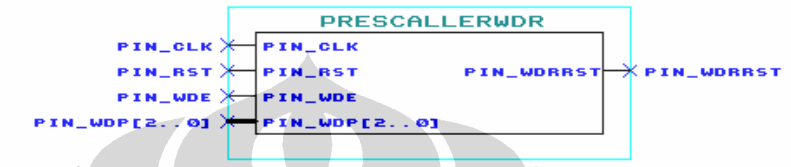
3. *Timer0/Counter0 Register* (TCNT0). Register ini memiliki alamat $32. Isi dari *register* ini dipakai untuk perhitungan *timer* ataupun *counter*. *Register* ini diperlihatkan pada Gambar 3.19.



Gambar 3.19 *Timer0/Counter0 Register* (TCNT0) [5]

3.2.5 Modul *PrescallerWDR*

Modul *prescallerWDR* berfungsi untuk mengatur pewaktuan dari *watchdog timer*. Simbol modul ini dapat dilihat pada Gambar 3.20. Jenis *port*, fungsi, dan konektivitas modul *prescallerWDR* diperlihatkan pada Tabel 3.17. Kombinasi masukan *pin_wdp* menentukan pewaktuan *watchdog timer*. Kombinasi masukan ini dapat dilihat pada Tabel 3.18. *Source code* modul ini disimpan dalam *file prescallerWDR.vhd* pada Lampiran 5.



Gambar 3.20 Modul *prescallerWDR*

Tabel 3.17 Jenis *Port*, Fungsi, Dan Konektivitas Modul *PrescallerWDR*

No	Nama *Port*	Jenis *Port*	Fungsi	Terhubung Ke
1	*pin_clk*	masukan	sebagai sumber *clock*	sistem *clock* utama
2	*pin_rst*	masukan	sebagai sumber *reset*	sistem *reset* utama
3	*pin_wde*	masukan	sistem *enable*	modul *core UIMega 8535*
3	*pin_wdp*	masukan	*prescaller timer*	modul *core UIMega 8535*
4	*pin_wdrrst*	keluaran	sistem reset sistem	semua modul

Tabel 3.18 Sumber *Clock Watchdog Timer* Berdasarkan *Pin_wdp*

Masukan *pin_wdp*	Sumber *Clock*
000	sistem *clock* utama/16
001	sistem *clock* utama/32
010	sistem *clock* utama/64
011	sistem *clock* utama/128
100	sistem *clock* utama/256
101	sistem *clock* utama/512
110	sistem *clock* utama/1024
111	sistem *clock* utama/2048

3.3 REGISTER UIMEGA 8535

UIMega 8535 memiliki beberapa register pendukung untuk kerja dari mikrokontroler. Register-register tersebut adalah *general purpose register* dan I/O register. *General purpose register* UIMega 8535 terdiri dari 16 buah register dan diberi nomor register 0 sampai dengan 15. Register 10 sampai dengan register 15 adalah register khusus yang juga berfungsi sebagai *pointer* dan identik dengan register 26 hingga 31 pada ATMega 8535. Register 10 adalah register X *low byte*, register 11 adalah register X *high byte*, register 12 adalah register Y *low byte*, register 13 adalah register Y *high byte*, register 14 adalah register Z *low byte*, dan register 15 adalah register Z *high byte*. Register X, Y, dan Z digunakan untuk pengalamatan tidak langsung ke RAM.

Pengalamatan ke register hanya membutuhkan 4 *bit* karena hanya berjumlah 16 buah register pada *general purpose register*, oleh karena itu *bit* ke-5 dari register di kode mesin instruksi tidak digunakan.

I/O register dari UIMega 8535 digunakan untuk mengontrol kerja dari beberapa fungsi mikrokontroler, seperti modul *timer interrupt*, external *interrupt*, *watchdog timer*, dan juga untuk penyimpanan *flag* status hasil operasi. Register yang termasuk dalam I/O register dapat dilihat pada Tabel 3.19.

Tabel 3.19 I/O Register UIMega 8535

Nama	Alamat	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SREG	$3F	I	T	H	S	V	N	Z	C
GICR	$3B	-	INT0	-	-	-	-	-	-
TIMSK	$39	-	-	-	-	-	-	-	TOIE
TCCR0	$33	-	-	-	-	-	CS02	CS01	CS00
TCNT0	$32	Timer0 (8 bit)							
WDTCR	$21	-	-	-	-	WDE	WDP2	WDP1	WDP0
PORT A	$1B	A7	A6	A5	A4	A3	A2	A1	A0
PORT B	$18	B7	B6	B5	B4	B3	B2	B1	B0
PORT C	$15	C7	C6	C5	C4	C3	C2	C1	C0
PORT D	$12	D7	D6	D5	D4	D3	D2	D1	D0

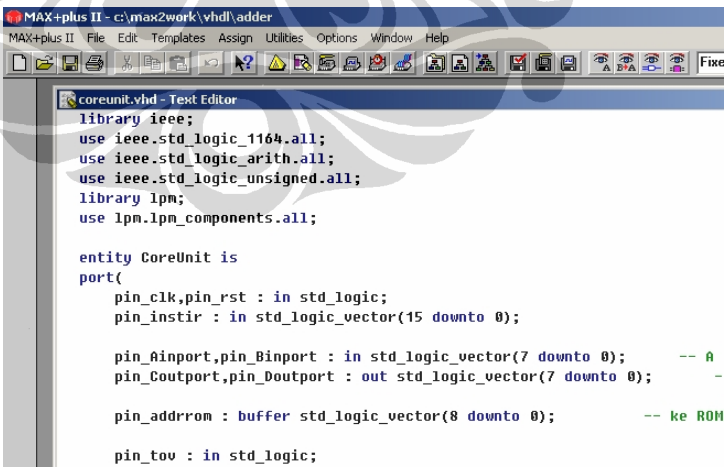
3.4 TAHAPAN PERANCANGAN DAN IMPLEMENTASI DENGAN APLIKASI MAX+PLUS II

Aplikasi MAX+plus II dari Altera memiliki kemampuan untuk proses pendesainan sistem logika yang lengkap, dimulai dari proses pendeskripsian hingga pemrograman *hardware*. Perancangan hingga implementasi dengan menggunakan aplikasi MAX+plus II memiliki beberapa tahapan, yaitu *design entry, project processing, error detection & location, project verification*, dan *device programming*.

3.4.1 *Design Entry*

Pendeskripsian sistem logika yang akan dirancang dilakukan pada tahapan *design entry*. MAX+plus II menyediakan beberapa *editor* untuk mendeskripsikan sistem yang akan dirancang, yaitu *text, waveform, floorplan, graphic* dan *symbol editor*. Pendesain dapat memilih jenis *editor* sesuai dengan keinginan dan kebutuhannya.

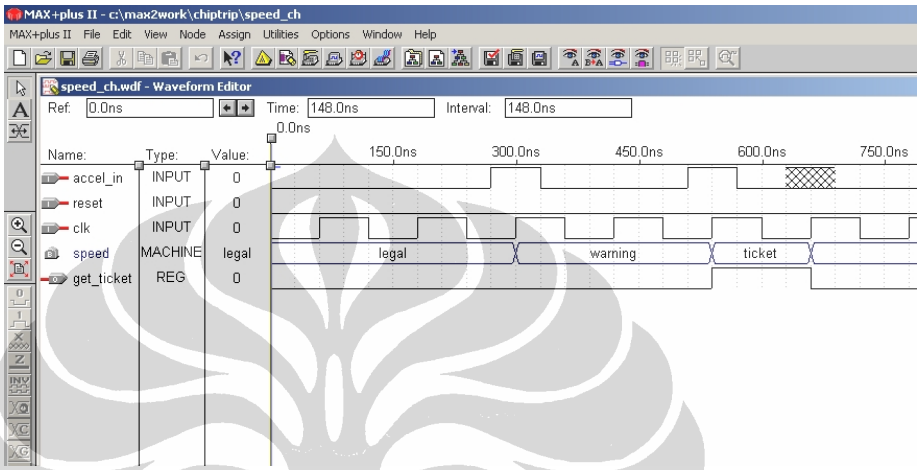
Text editor merupakan *tool* bagi pendesain untuk membuat suatu desain berbasiskan teks/tulisan. MAX+plus II mendukung beberapa bahasa HDL, seperti AHDL, VHDL, dan Verilog HDL. Tampilan *text editor* diperlihatkan pada Gambar 3.21.



Gambar 3.21 Tampilan *text editor*

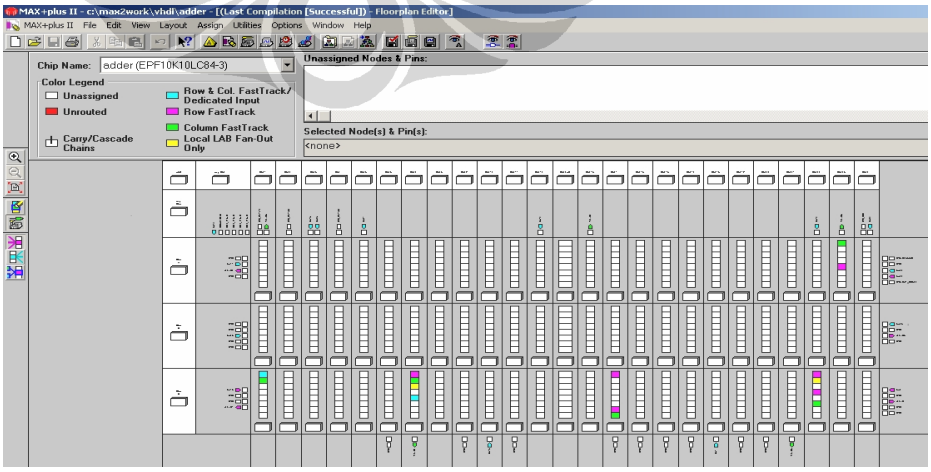
Waveform editor dapat digunakan sekaligus sebagai *editor* maupun sebagai *simulator*. Pendesain dapat menggunakan *waveform editor* untuk

mendefinisikan masukan dalam format logika ataupun *state machine* dan keluaran dalam format *combinatorial, registered,* dan *state machine*. *Waveform editor* dapat melakukan simulasi pewaktuan dengan memberikan masukan yang ada sebuah vektor tes atau kondisi logika tertentu dan diamati keluarannya. Tampilan *waveform editor* dapat dilihat pada Gambar 3.22.



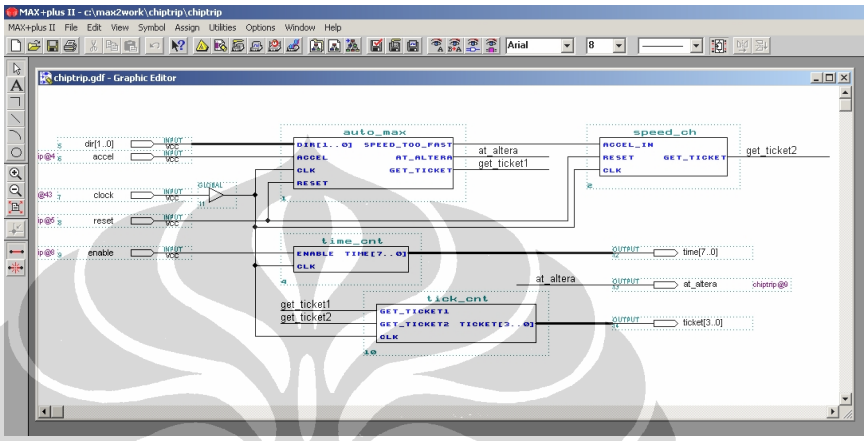
Gambar 3.22 Tampilan *waveform editor*

Floorplan editor menampilkan penggunaan *resources* dari divais target secara fisik. *Floorplan editor* juga menampilkan hasil dari pengalokasian beberapa divais target oleh *compiler*. Tampilan *floorplan editor* diperlihatkan pada Gambar 3.23.



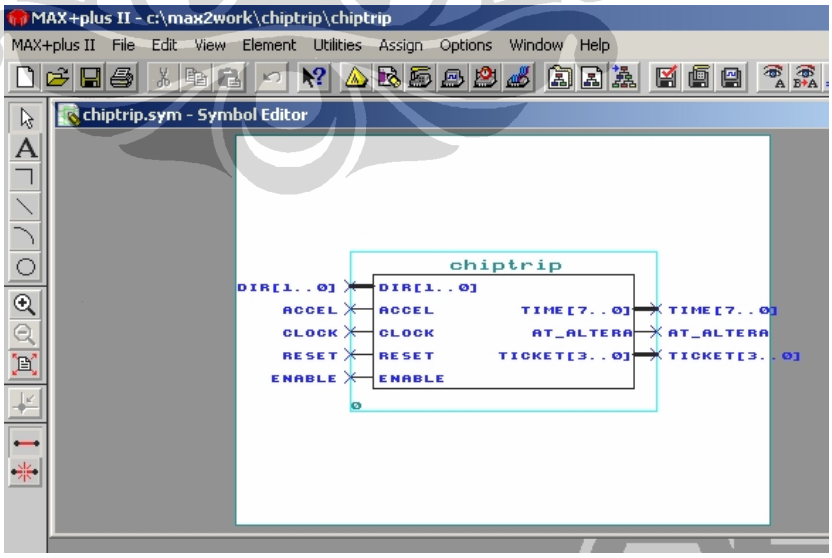
Gambar 3.23 Tampilan *floorplan editor*

Graphic editor memungkinkan pendesain untuk membuat suatu desain yang sederhana maupun kompleks secara cepat. *Graphic editor* dapat menggunakan simbol-simbol bawaan dari MAX+plus II maupun dari simbol yang dibuat oleh pendesain. Tampilan dari *graphic editor* ini dapat dilihat pada Gambar 3.24.



Gambaran 3.24 Tampilan *graphic editor*

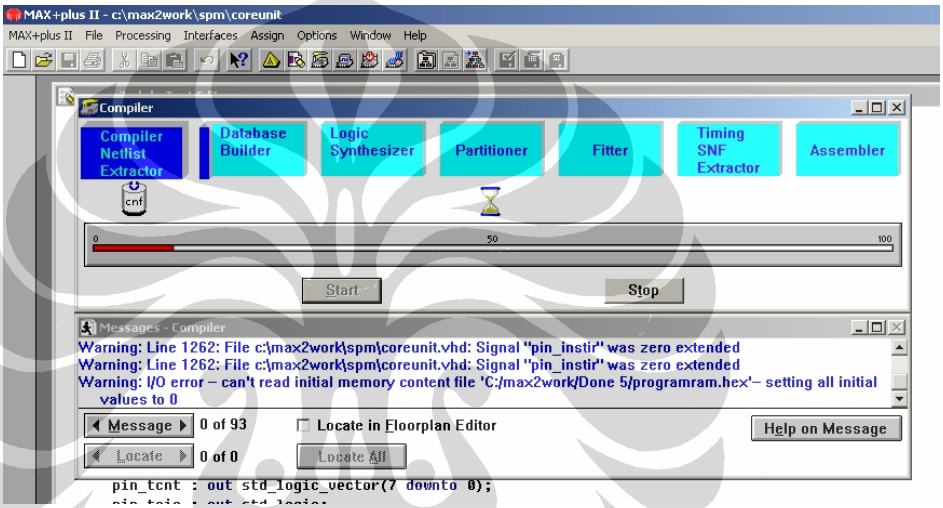
Symbol editor memiliki kemampuan untuk mengedit simbol-simbol yang ada maupun simbol yang dibuat oleh pendesain. Tampilan dari *symbol editor* diperlihatkan pada Gambar 3.25.



Gambar 3.25 Tampilan *symbol editor*

3.4.2 *Project Processing*

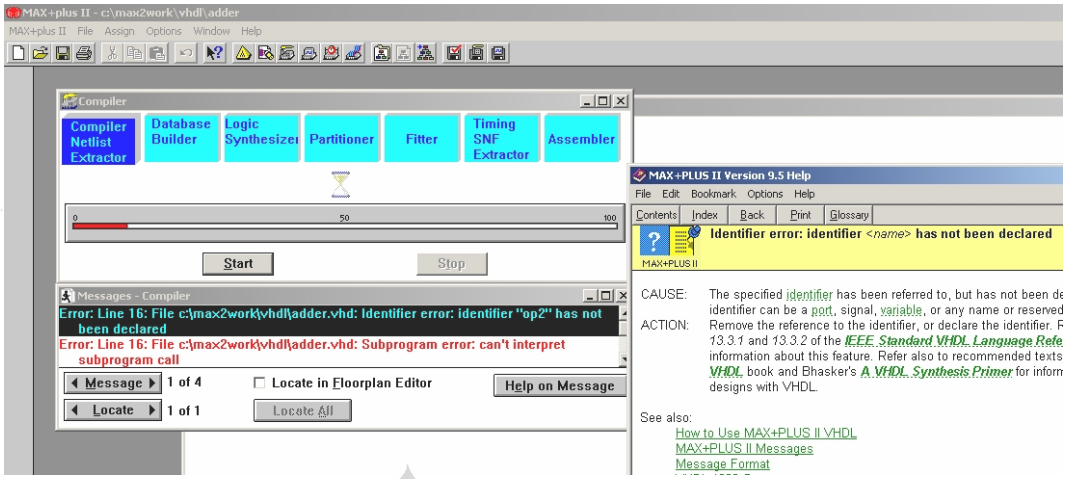
Tahapan *project processing* dilakukan oleh *compiler* aplikasi MAX+plus II. *Compiler* aplikasi MAX+plus II terdiri dari serangkaian modul yang akan melakukan pengecekkan terhadap *error* yang mungkin terjadi, sintesis rangkaian logika, penempatan hasil kompilasi ke sebuah atau beberapa divais Altera, dan sekaligus menghasilkan *file* yang nantinya akan dipergunakan untuk simulasi, analisa waktu, dan pemrograman divais. Gambar 3.26 memperlihatkan proses kompilasi sedang dilakukan.



Gambar 3.26 Tampilan proses kompilasi

3.4.3 *Error Detection & Location*

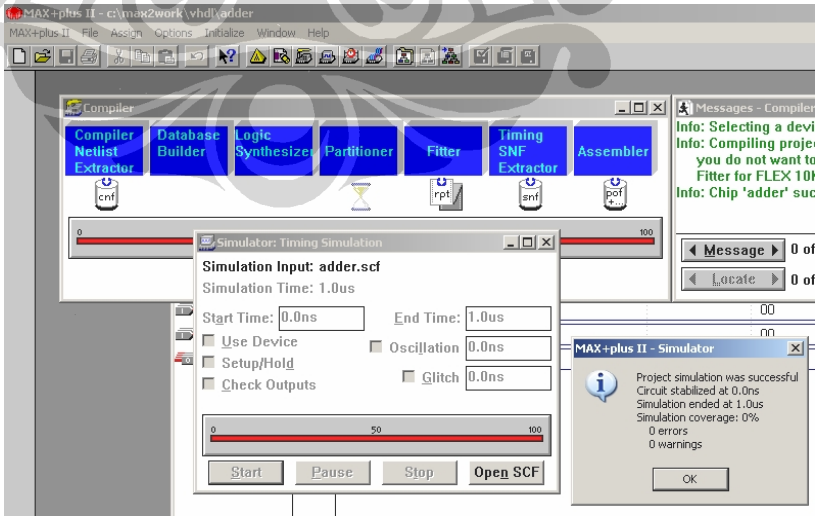
Tahap *error detection & location* merupakan tahap selanjutnya setelah kompilasi dilakukan. Pada tahapan ini dapat diketahui apakah rancangan memiliki *error* atau tidak, bila rancangan memiliki *error* maka aplikasi MAX+plus II dapat menunjukkan di mana *error* itu terjadi pada bagian *message processor*. Bagian *message processor* juga dapat memberikan saran untuk mengatasi *error* yang terjadi. Gambar 3.27 memperlihatkan tampilan bilamana terjadi *error* dan saran dari *message processor* untuk mengatasi *error* tersebut.



Gambar 3.27 Tampilan bila terjadi *error*

3.4.4 *Project Verification*

Tahapan *project verification* dilakukan untuk mengecek apakah desain yang telah dihasilkan sesuai dengan apa yang diinginkan. Tahapan ini menggunakan MAX+plus II Simulator dan *waveform editor* untuk mensimulasikan hasil desain. Verifikasi dilakukan dengan memberikan vektor tes atau kondisi logika tertentu kepada masukan dan diamati keluaran dari desain. Tampilan MAX+plus II Simulator dapat dilihat pada Gambar 3.28.

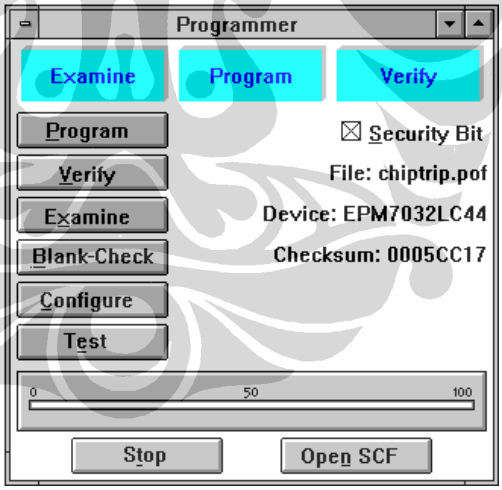


Gambar 3.28 Tampilan MAX+plus II Simulator

3.4.5 *Device Programming*

Tahapan ini akan melakukan *download* data konfigurasi dari hasil rancangan untuk memprogram divais target. Tahapan ini membutuhkan kabel data khusus dari Altera yang dihubungkan ke *development board* tempat divais target berada. Kabel-kabel data ini yaitu diantaranya adalah *BitBlaster, ByteBlaster,* dan *ByteBlasterMV*. Perbedaan jenis kabel *download* ini terletak pada jenis kabel dan tipe divais target yang didukung.

Kabel *BitBlaster* digunakan untuk memprogram atau mengkonfigurasi divais target Altera tipe MAX9000, MAX7000S, MAX7000A, FLEX10K, FLEX8000, dan FLEX6000, melalui *serial port* RS-232 standar. Kabel *ByteBlaster* mendukung semua divais target yang didukung oleh kabel *BitBlaster,* ditambah divais target MAX9000A. Kabel *ByteBlasterMV* mendukung semua divais target yang didukung oleh kabel *ByteBlaster*, ditambah divais target FLEX10KV, FLEX10KA, dan FLEX10KB. Tampilan MAX+plus II *Hardware Programmer* diperlihatkan pada Gambar 3.29.



Gambar 3.29 Tampilan MAX+plus II *Hardware Programmer*

Tesis ini hanya mencapai tahapan *project verification* dan tidak sampai tahapan *device programming*. Simulasi dan verifikasi dilakukan hingga pada simulasi pewaktuan yang terdapat pada *waveform editor*.