DESAIN PROCESSOR-16bit SEDERHANA MENGGUNAKAN PERANGKAT LUNAK QUARTUS II V.2

Oleh:
Arie S. M. Lumenta
Jurusan Teknik Elektro Fakultas Teknik UNSRAT

Abstrak

Pada tulisan ini akan dipaparkan desain processor sederhana menggunakan VHDL. Disebut processor sederhana karena processor tersebut hanya terdiri dari beberapa set instruksi yang telah ditentukan. Inti dari tulisan ini adalah ingin menunjukkan bahwa mendesain suatu prosessor itu relatif mudah, apalagi dengan alat bantu VHDL untuk melakukan simulasi dan verifikasi. Pada tulisan ini digunakan perangkat lunak Quartus II versi 2 untuk melakukan simulasi dan implementasi serta Synopsis untuk melakukan sintesis. Dengan menggunakan perangkat lunak yang ada terbukti bahwa desain arsitektur dan set instruksi yang ada dapat disimulasikan, disintesis dan diimplementasikan.

Kata kunci: *prosessor sederhana, asm, vhdl*

1. Pendahuluan

Processor atau dikenal juga dengan central processing unit adalah suatu perangkat pengolah data dalam suatu sistem komputer. Pada prinsipnya suatu prosessor terdiri dari ALU (Aritmethic Logic Unit) dan register-register. Sebagai pengontrol operasinya sendiri dilakukan oleh sebuah control unit.

Pada tulisan ini akan didesain suatu processor sederhana dengan beberapa set instruksi menggunakan pendekatan top-down. Sebagai alat bantu digunakan CAD tool yakni Quartus II versi 2 untuk mendeskripsikan hardware. Dalam mendeskripsikan hardware digunakan VHDL (VHSIC Hardware Description Language).

Set instruksi yang akan diimplementasikan adalah seperti pada tabel 1

Tabel 1. Set instruksi simple processor

Instruksi	Deskripsi	Operasi (C Style Coding)
load	Load to rs	rs = extend(immediate)
mov	Move rs to rd	rd = rs
add	Add	rd = rs + extend(immediate)
subi	Sub immediate	rd = rs - extend(immediate)
sw	Store word	MEM[rs + extend(immediate)] = rd

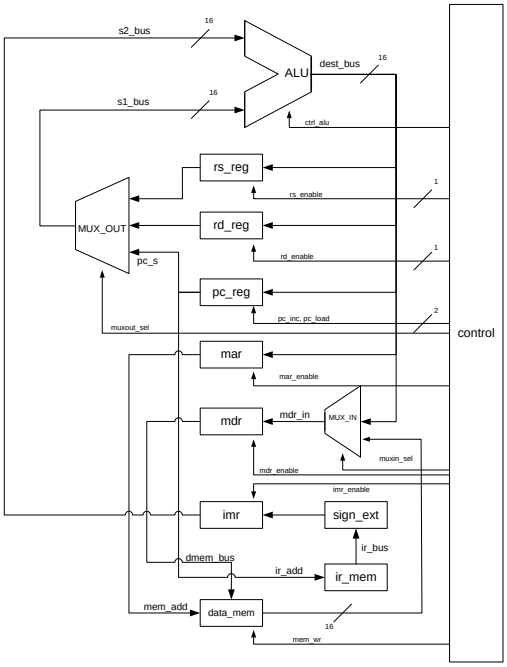
Sedangkan format instruksinya adalah sebagai berikut, sesuai tabel 2. Opcode instruksi adalah bit 8 sampai bit 15 (8 bit). Sisanya bit 7 sampai bit 0 (8 bit) adalah immediate/unused

Tabel 2. Format instruksi simple processor

Instruksi	Bits	
	15 8	7 0
load	00000001	immediate
mov	10000000	unused
add	00000100	unused
subi	00000011	immediate
sw	00011001	immediate

2. Diagram Arsitektur Processor Sederhana dan Penjelasan Fungsional

Disain arsitektur simple processor adalah seperti pada gambar 1 dibawah ini;



Gambar 1. Arsitektur processor sederhana

Arsitektur processor sederhana diatas dikelompokkan menjadi 3 bagian besar yaitu:

1. Unit processor
2. Unit memory dan
3. Unit controller.

Komponen-komponen di unit processor adalah:

1. Alu
2. Register-register (rs_reg, rd_reg, pc_reg, mar, mdr dan imr)
3. Multiplexer.
4. Memory (ir_mem dan data_mem).

Alu memilki 2 operand dan operasi-operasi yang diolah alu adalah :

1. add (rd = rs + immediate)
2. subi (rd = rs - extend)
3. sw (rd = mem[rs + extend(imm)]

Penjelasan lengkap masing masing bagian adalah sebagai berikut:

1 alu.

Sinyal input : s1_bus(16bit), s2_bus (16bit), ctrl_alu(3 bit)

Sinyal output : dest_bus (16 bit)

Tabel 4. Deklarasi Alu

input s1_bus	input s2_bus	ctrl_alu	dest_bus
Don't care	Don't care	"001"	s1_bus
Don't care	Don't care	"010"	s2_bus
Don't care	Don't care	"101"	dest_bus = s1_bus + s2_bus
Don't care	Don't care	"110"	dest_bus = s1_bus - s2_bus

2. register (rs reg dan rd reg)

Sinyal input : d_in (16 bit), reset, enable (1 bit), clk

Sinyal output : q_out (16 bit)

Tabel 5. Deklarasi register

d_in	reset	clk	enable	q_out
Don't care	'1'	Don't care	Don't care	"0000000000000000"
Don't care	'0'	Rising	'1'	d_in
Don't care	'0'	Rising	Don't care	q_out

3. pc reg

Sinyal input : pc_reg_in(16bit),clk, reset,enable(1bit), pc_reg_load (1bit)

Sinyal output : pc_reg_out (16 bit)

Table 6. Deklarasi pc reg

pc_reg_in	clk	rst	enable	pc_reg_load	pc_reg_out
Don't care	Don't care	'1'	Don't care	Don't care	"0000000000000000"
Don't care	Rising	'0'	'1'	Don't care	pc_reg_out + 1
Don't care	Rising	'0'	Don't care	'1'	pc_reg_in

4. multiplekser Out
Sinyal input : rs_reg_out(16bit), rd_reg_out(16bit), pc_reg_out(16bit), mux_out_sel (2 bit)
Sinyal output : s1_bus(16bit).

Tabel 7. Deklarasi MuxOut

rs_reg_ out	rd_reg_ out	pc_reg_ out	mux_out _sel	s1_bus
Don't care	Don't care	Don't care	"00"	rs_reg_ out
Don't care	Don't care	Don't care	"01"	rd_reg_ out
Don't care	Don't care	Don't care	"10"	pc_reg_ out

5. multiplekser in
Sinyal input : dest_bus(16bit), q_mem_out(16bit), muxin_sel(1bit)
Sinyal output : mdr_in (16 bit)

Tabel 8. Deklarasi MuxIn

dest_bus	q_mem_out	muxin_sel	mdr_in
Don't care	Don't care	'0'	dest_bus
Don't care	Don't care	'1'	q_mem_out

6. ir mem
Sinyal input : address (8 bit)
Sinyal output : q (16 bit)

Tabel 9. Deklarasi ir mem

address	q
"00000000"	"0000000100000001"
"00000001"	"1000000000000000"
"00000010"	"0000010000000111"
"00000011"	"0000001100000010"
"00000100"	"0001100100000011"
Don't care	"1111111111111111"

7. data mem
Sinyal input : address(8bit),we(1bit), data (16 bit)
Sinyal output : q (16 bit)

Tabel 10. Deklarasi data mem

address	we	data	q
Don't care	Don't care	Don't care	"0000000000000000"
Don't care	'1'	Don't care	mdr_out

8. mar
Sinyal input : mar_in(16bit), mar_enable(1bit), mar_reset, clk
Sinyal output : mar_out (16 bit)

Tabel 11. Deklarasi mar

mar_in	mar_ enable	mar_ reset	clk	mar_out
Don't care	Don't care	'1'	Don't care	"0000000000000000"
Don't care	'1'	'0'	rising	mar_in

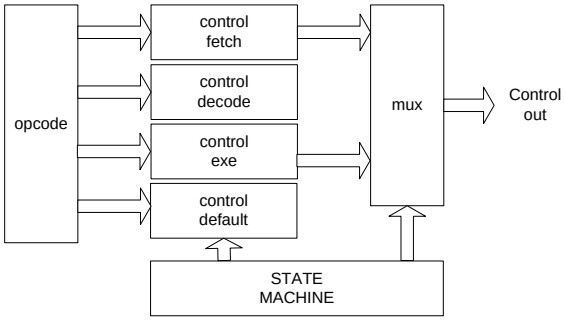
9. mdr
Sinyal input : mdr_in(16bit), mdr_enable(1bit), mdr_reset, clk
Sinyal output : mdr_out (16 bit)

Tabel 12. Deklarasi mar

mdr_in	mdr_ enable	mdr_ reset	clk	mdr_out
Don't care	Don't care	'1'	Don't care	"0000000000000000"
Don't care	'1'	'0'	rising	mar_in

3. Control Unit

Control unit yang ada pada processor sederhana yang didesain, dibagi menjadi 4 kelompok yaitu: control fetch, control decode, control exe dan control default. Model control ini digambarkan dalam gambar dibawah ini.



Gambar 2. Blok control

Control default pada blok control diatas hanya merupakan fungsi dari state machine yang mendefinisikan urut-urutan proses yaitu:

fetch → decode →exe

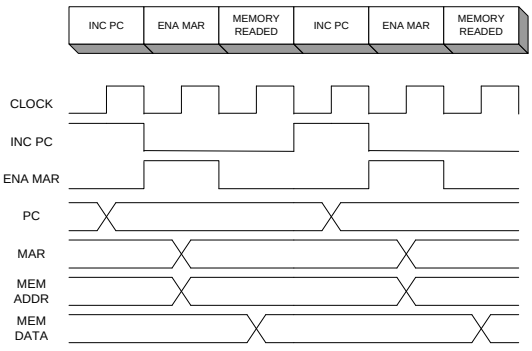
urut-urutan tersebut adalah urut-urutan yang standar dalam siklus instruksi suatu processor.

4. State Machine

Hal-hal yang menentukan desain state machine:

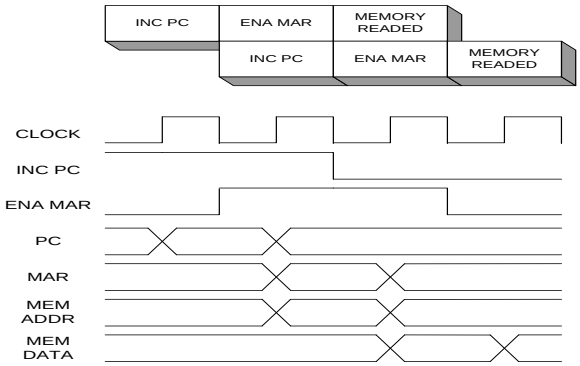
1. Latency 1 clock dari PC
2. Latency 1 clock dari MAR
3. Latency 1 clock dari MEMORY

Dapat disimpulkan untuk membaca satu alamat dari memory dibutuhkan 3 clock. Sementara memory mempunyai lebar data 8 bit sehingga sebuah instruksi lengkap disimpan dalam 2 alamat memory. Fetch sebuah instruksi membutuhkan 6 clock.



Gambar 3. Siklus Fetch 6 clock

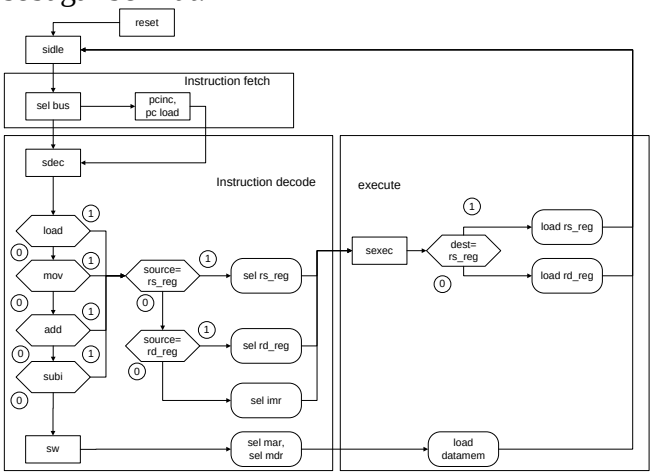
Dari gambar 3, siklus fetch dapat dioptimasi menjadi 4 clock seperti pada gambar 4 :



Gambar 4. Siklus Fetch 4 clock

5. Bagan ASM

Bagan ASM dari set instruksi processor sederhana yang didesain digambarkan sebagai berikut:



Gambar 5. Bagan ASM set instruksi simple processor

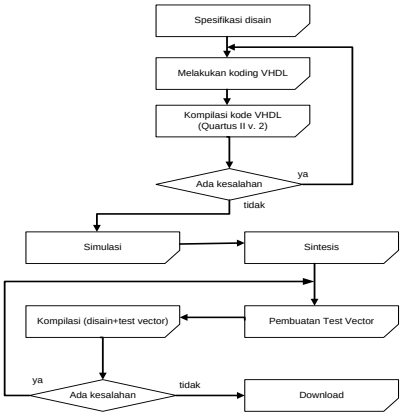
Set instruksi simple processor dibagi menjadi 3 bagian yaitu:

1. Fetch.
2. Decode dan
3. Execute.

Urutan langkah ini sesuai dengan disain kontrol unit diatas.

6. Metode

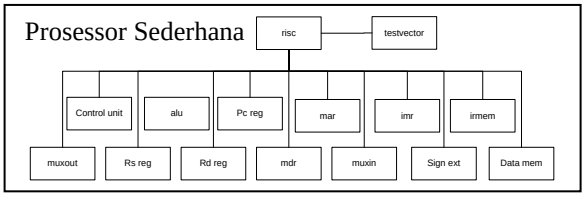
Metode yang dilakukan dalam pembuatan tulisan ini adalah sebagai berikut:



Gambar 6. Diagram alir metode

7. Struktur File .VHD

Struktur file .VHD processor sederhana yang didesain adalah sebagai berikut:

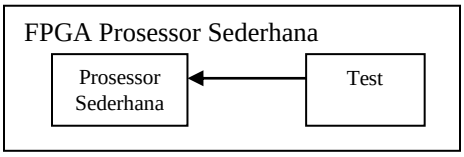


Gambar 7. struktur file .VHD

Top level pada disain processor sederhana adalah risc.vhd, sedangkan komponen-komponen lainnya berada dibawah top level. File top level inilah yang menghubungkan semua komponen yang ada pada processor sederhana.
File testvector.vhd adalah sebagai test untuk menguji processor setelah di download atau diimplementasikan. File ini diperlukan untuk keperluan pengujian processor setelah proses download.

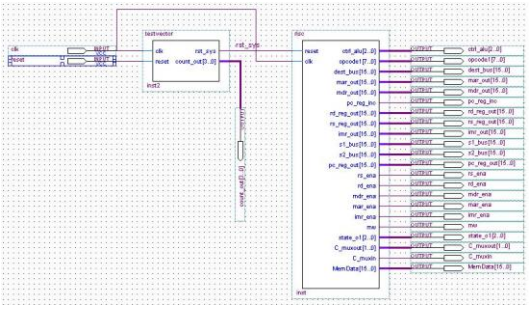
8. Sistem Keseluruhan dan Hasil Sintesis

Sistem keseluruhan adalah sebagai berikut:



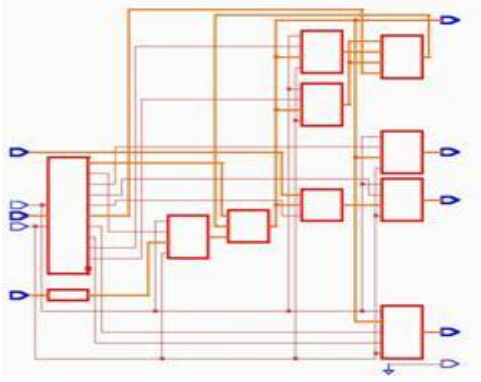
Gambar 8. Sistem Processor Sederhana

Dengan menggunakan Quartus II versi 2 sistem keseluruhan terlihat seperti gambar 9 dibawah ini.



Gambar 9. Sistem keseluruhan

Sedangkan hasil sintesi terhadap koding VHDL prosessor sederhana menggunakan CAD Tool Synopsis adalah sebagai berikut seperti tampak pada gambar 10 berikut.

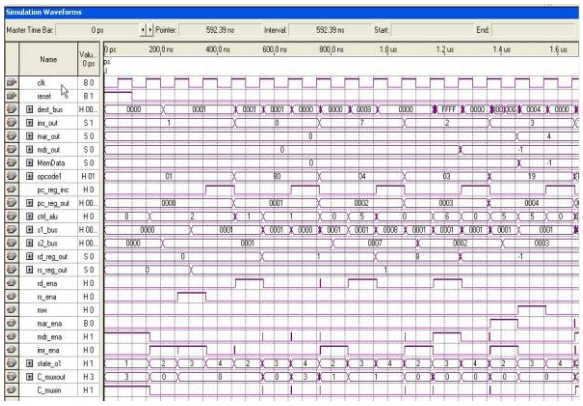


Gambar 10. Hasil sintesis prosessor sederhana

Pada gambar 10 hasil sintesis tersebut diatas ada satu pin yang tidak terhubung ke rangkaian. Pin tersebut adalah pin yang menghubungkan prosessor dengan memori. Pin tersebut lepas dari rangkaian, karena rangkaian memori tidak ikut disintesis.

9. Simulasi Sebelum dan Setelah Download (Implementasi)

Hasil simulasi keseluruhan sistem sederhana sebelum download atau implementasi adalah sebagai berikut:



Gambar 11. Simulasi Prosessor Sederhana

Hasil simulasi tersebut menunjukkan bahwa prosessor menjalankan set instruksi sesuai dengan spesifikasi set instruksi yang diberikan. Untuk memudahkan analisa dikeluarkan sinyal state (state_o1) yang menunjukkan posisi state pada saat melakukan operasi. State tersebut adalah sebagai berikut seperti ditunjukkan pada tabel 4 berikut ini.

Tabel 13. Status state

state	status	Jumlah clock
1	idle	-
2	fetch	1
3	decode	1
4	execute	1

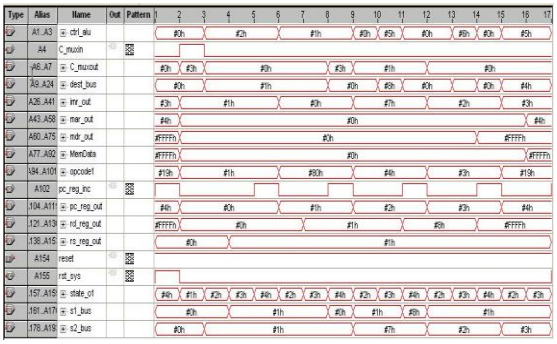
Posisi *idle* hanya pada saat pertama saja. Selanjutnya setelah posisi *execute* posisi state kembali pada state 2 yakni *fetch*. Setiap perpindahan state dibutuhkan 1 clock.

Sedangkan untuk masing-masing instruksi jumlah clock yang dibutuhkan adalah sebagai berikut seperti pada tabel 5 dibawah ini.

Tabel 14. Jumlah clock tiap instruksi

Instruksi	Jumlah clock
load	2
mov	1
add	3
subi	3
sw	2

Sedangkan hasil simulasi setelah proses download (implementasi) ke dalam device EP20K400EB652 adalah sebagai berikut:

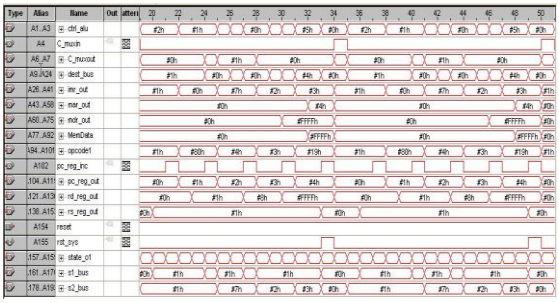


Gambar 12. Simulasi setelah proses download.

Hasil simulasi dengan menggunakan quartus sebelum dan setelah di download menunjukkan hasil yang sama. Untuk lebih jelasnya dapat dilihat pada gambar 13 berikut. Setelah 5 instruksi selesai dilaksanakan dan sistem te*reset*, maka proses akan kembali pada instruksi pertama lagi, begitu setrerusnya yakni:

inst1→inst2→inst3→inst4→inst5→inst12→inst22→inst32→inst42→inst51→…

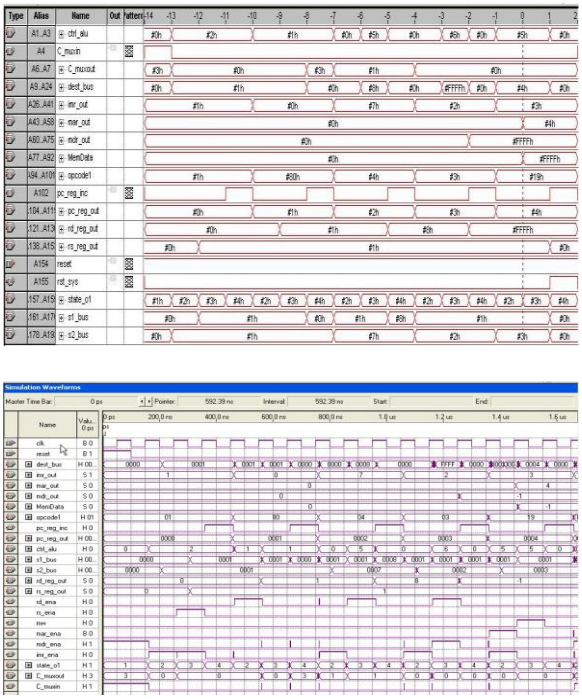
Proses tersebut terlihat pada gambar 13 berikut.



Gambar 13. Keadaan opcode setelah reset sistem

Isi register-register yakni rs register dan rd register serta pc register tetap sama dengan siklus instruksi sebelumnya. Begitu juga dengan isi datamem dan mdr tidak berubah.

Hasil perbandingan yang lebih jelas antara simulasi sebelum dan setelah proses download (implementasi) dapat dilihat pada gambar 14 dibawah.



Gambar 14. Perbandingan hasil simulasi

10. Kesimpulan dan Rekomendasi

Proses implementasi processor sederhana ke board memiliki hasil simulasinya yang sama dengan simulasi sebelum dimplementasikan dengan logic cell usage adalah 327 Hal ini menunjukkan bahwa hasil rancangan sesuai spesifikasi yang ditetapkan dapat disimulasikan, disintesis dan diimplementasikan.

Hasil ini menunjukkan bahwa sebuah prosessor sederhana dapat didesain dengan relatif mudah sesuai dengan yang diinginkan bedasarkan kebutuhan dengan cara merancang set instruksi yang diinginkan. Hal ini membuka jalan untuk perancangan prosessor yang lebih rumit.

Daftar Pustaka.

[1] Green, D. (1986). Modern Logic Design. Addison Wesley.

[2] http://debian.paume.itb.ac.id/~pse.

[3] Chang, K. C. (1999). Digital Systems Design with VHDL and Synthesis: An Integrated Approach. IEEE Computer Society Press.

[4] _____ (1997). Digital Design and Modeling with VHDL and Synthesis. IEEE Computer Society Press.

[5] Mano, M. (1997). Computer System Architecture. Prentice Hall.

[6] _____ (2003). Digital System. Prentice Hall.