Innehållsförteckning

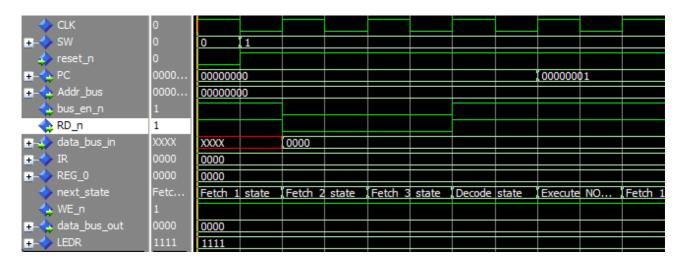
1 Sammanfattning	1
2 Tidsanalys	
2.1 Cykler 1-14	1
	4
2.3 Cykler 20 – 24	5
5	7
	8
	9
	10
	11

1 Sammanfattning

Verifiering med Modelsim har gjorts av enkel CPU. 50 instruktionscykel har stegats igenom, och värdet i CPU:ns register, adress och data bussar, samt lysdioderna status har analyserats.

2 Tidsanalys

2.1 Cykler 1-14



Lasse Karagiannis, Simulering med Modelsim Uppgift8 2016-11-05

CLK_PERIOD	1 to 10	11	12	13	14
SW	0	1	1	1	1
reset_n	0	1	1	1	1
PC	0b00000000	0b00000000	0b0000000	0b00000000	0b00000001
Addr_bus	0b00000000	0b00000000	0b0000000	0b00000000	0b00000000
bus_en_n	1	0	0	1	1
RD_n	1	0	0	1	1
data_bus_in	0xUUUU	0x0000	0x0000	0x0000	0x0000
IR	0x0000	0x0000	0x0000	0x0000	0x0000
REG_0	0x0000	0x0000	0x0000	0x0000	0x0000
next_state	Fetch_1_state	Fetch_2_state	Fetch_3_state	Decode	Execute_NOP
WE_n	1	1	1	1	1
data_bus_out	0x0000	0x0000	0x0000	0x0000	0x0000
LEDR	0b1111	0b01111	0b1111	0b1111	0b1111

Fetch_1_state ligger som next_state fram till att reset_n går går hög, samtliga register är nollställda under reset-fasen, vi har nämligen kod (klistrar in en del av den)

```
process(reset n, clk 50)
begin
if reset n = '0' then
 PC_reg <= X"00"; -- after reset, PC is zero (i.e address 0)
       IR <= X"0000"; -- NOP, no operation
       Addr bus <= X"00"; -- initialize registers
       CPU REG 0 <= (others => '0'); -- initialize registers
       next state <= Fetch 1 state;</pre>
       bus en n \le 1';
       data bus out <= "000000000000000"; -- initialize registers
  WE n \le '1';
  RD n \le '1';
       CPU state <= "00";
elsif(rising edge(clk 50)) then
Lysdioderna lyser p.g.a. koden i OUT LED
process(reset n, clk 50)
              begin
              if reset n = '0' then
                      LEDG <= "1111"; -- LED-signals go high during reset
              elsif(rising_edge(clk_50)) then
```

Instruktionsläsningen kräver 2 cykler, där IR har laddats med instruktionskoden för NOP 0x0000 och är giltlig under cykel 14.

2. 2 Cykler 15-19

CLK SW 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1						
Teset	◆ CLK	1				
PC	∳ SW	1 1				
PC		1				
Addr_bus 00000 1	- T	0000 0000000	1			100000010
Dus_en_n	-4 Addr bus			1		
CLK_PERIOD	👍 bus_en_n	1				
CLK_PERIOD	♣ RD_n	1				
Reg_0	7.5	0000 0000		100A		
Next State Fetch State	⊢ ♦ IR				100A	
WE_n 1 0000 0000 1111 1111 1111 1111 CLK_PERIOD 15 16 17 18 19 SW 1 1 1 1 1 1 reset_n 1 <td></td> <td>0000</td> <td></td> <td></td> <td></td> <td></td>		0000				
WE_n 1 0000 0000 1111 1111 1111 1111 CLK_PERIOD 15 16 17 18 19 SW 1 1 1 1 1 1 reset_n 1 <td>next_state</td> <td>Fetc Fetch</td> <td>1 state Fetch 2</td> <td>state (Fetch 3 state</td> <td>Decode state</td> <td>Execute LOA</td>	next_state	Fetc Fetch	1 state Fetch 2	state (Fetch 3 state	Decode state	Execute LOA
CLK_PERIOD 15 16 17 18 19 SW 1 1 1 1 1 1 1 1 reset_n 1 1 1 1 1 1 1 1 PC 0b00000001 0b00000001 0b00000001 0b00000001 0b00000001 Addr_bus 0b0000000 0b00000001 0b00000001 0b00000001 0b00000001 bus_en_n 1 0 0 0 1 1 RD_n 1 0 0 0 1 1 RD_n 1 0 0 0 1 1 data_bus_in 0x0000 0x0000 0x100A 0x100A 0x100A IR 0x0000 0x0000 0x0000 0x0000 0x0000 REG_0 0x0000 0x0000 0x0000 0x0000 0x0000	♦ WE_n	1				
CLK_PERIOD 15 16 17 18 19 SW 1	- data_bus_out	0000				
SW 1 1 1 1 1 1 reset_n 1 1 1 1 1 1 1 PC 0b00000001 0b000000001 0b000000000 0b0000000 0b0000000 0b00000000 0b000000000 0b00000000 0b000000000 <td></td> <td>1111 1111</td> <td></td> <td></td> <td></td> <td></td>		1111 1111				
SW 1 1 1 1 1 1 reset_n 1 1 1 1 1 1 1 PC 0b00000001 0b000000001 0b000000000 0b0000000 0b0000000 0b00000000 0b000000000 0b00000000 0b000000000 <td></td> <td></td> <td></td> <td></td> <td></td> <td></td>						
SW 1 1 1 1 1 1 reset_n 1 1 1 1 1 1 1 PC 0b00000001 0b000000001 0b000000000 0b0000000 0b0000000 0b00000000 0b000000000 0b00000000 0b000000000 <td></td> <td></td> <td></td> <td></td> <td></td> <td></td>						
SW 1 1 1 1 1 1 reset_n 1 1 1 1 1 1 1 PC 0b00000001 0b000000001 0b000000000 0b0000000 0b0000000 0b00000000 0b000000000 0b00000000 0b000000000 <td>CLK PERIOD</td> <td>1!</td> <td>5 16</td> <td>17</td> <td>18</td> <td>19</td>	CLK PERIOD	1!	5 16	17	18	19
reset_n 1 </td <td>_</td> <td></td> <td>1 1</td> <td> 1</td> <td>1</td> <td>1</td>	_		1 1	1	1	1
PC 0b00000001 0b00000001 0b00000001 0b00000001 0b00000001 Addr_bus 0b00000000 0b00000001 0b00000001 0b00000001 0b00000001 bus_en_n 1 0 0 1 1 RD_n 1 0 0 1 1 data_bus_in 0x0000 0x0000 0x100A 0x100A 0x100A IR 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 REG_0 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000			1 1	1	1	1
Addr_bus 0b0000000 0b0000001 0b0000001 0b0000001 0b00000001 0b00000001 bus_en_n 1 0 0 1 1 RD_n 1 0 0 1 1 data_bus_in 0x0000 0x0000 0x100A 0x100A 0x100A IR 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 REG_0 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000	_	0b0000000	1 0b00000001	0b00000001	0b00000001	0b00000010
bus_en_n 1 0 0 1 1 RD_n 1 0 0 1 1 data_bus_in 0x0000 0x0000 0x100A 0x100A 0x100A IR 0x0000 0x0000 0x0000 0x100A 0x100A REG_0 0x0000 0x0000 0x0000 0x0000 0x0000						
RD_n 1 0 0 1 1 data_bus_in 0x0000 0x0000 0x100A 0x100A 0x100A IR 0x0000 0x0000 0x0000 0x100A 0x100A REG_0 0x0000 0x0000 0x0000 0x0000 0x0000	-					1
data_bus_in 0x0000 0x0000 0x100A 0x100A 0x100A IR 0x0000 0x0000 0x0000 0x100A 0x100A REG_0 0x0000 0x0000 0x0000 0x0000 0x0000			1 0	0	1	1
IR 0x0000 0x0000 0x0000 0x100A 0x100A REG_0 0x0000 0x0000 0x0000 0x0000	-	0x0000	0x0000	0x100A	0x100A	0x100A
REG_0 0x0000 0x0000 0x0000 0x0000 0x0000						
-		0x000	J UXUUUU			
	IR					
WE n 1 1 1 1 1 1	IR REG_0	0x0000	0x0000	0x0000	0x0000	0x0000
data_bus_out 0x0000 0x0000 0x0000 0x0000 0x0000	IR REG_0	0x0000	0x0000	0x0000	0x0000	0x0000
LEDR 0b1111 0b1111 0b1111 0b1111	IR REG_0 next_state WE_n	0x0000 Fetch_	0x0000 1 Fetch_2 1 1	0x0000 Fetch_3 1	0x0000 Decode	0x0000 Execute_LOAD 1

Här hämtas och exekveras intruktionen LOAD_R0 #A med op-kod 0x100A på adressposition 1 i kod-minnet. PC inkrementerades redan under exekveringsfasen av NOP, under cykel 14. Addr_bus kan inte adressera kod-minnet under decode-fasen av NOP, eftersom samma adressbuss används för både kod och data.

En hel cykel används (16) till att låta RD_n och bus_en_n bli stabil, och koppla PC till adressbussen.

Nästa cykel (17) liger kod-data på kod-bussen, för att läsas in till IR cykeln därpå. Man borde kunna slå ihop cykel 16 och 17, dvs. att RD_n och bus_en_n bara får vara låga under en cykel och IR läses samtidigt som kod ligger stabil på kodbussen utan något mellansteg.

2.3 Cykler 20 - 24

◆ CLK	1										
⊢ ♦ SW	1	1									
reset_n	1										
⊢ ∢ PC	0000	00000	0010							0000001	1
⊢ ⇔ Addr_bus	0000	00000	0001	000000	10						
💠 bus_en_n	1										
♠ RD_n	1										
⊢ 4 > data_bus_in	0000	100A				2010					
⊢ ♦ IR	0000	100A						2010			
⊢ → REG_0	0000	000A									
next_state	Fetc	Fetch	1 s	Fetch 2	state	Fetch 3	state	Decode	state	Execute	STO
♦ WE_n	1										
⊢ ⇔ data_bus_out	0000	0000									
⊢ ♦ LEDR	1111	1111									

CLK_PERIOD	20	21	22	23	24
SW	1	1	1	1	1
reset_n	1	1	1	1	1
PC	0b00000010	0b00000010	0b0000010	0b00000010	0b00000011
Addr_bus	0b0000001	0b00000010	0b0000010	0b00000010	0b00000010
bus_en_n	1	0	0	1	1
RD_n	1	0	0	1	1
data_bus_in	0x100A	0x100A	0x2010	0x2010	0x2010
IR	0x100A	0x100A	0x100A	0x2010	0x2010
REG_0	0x000A	0x000A	0x000A	0x000A	0x000A
next_state	Fetch_1	Fetch_2	Fetch_3	Decode:x	ecute_STORE
WE_n	1	1	1	1	1
data_bus_out	0x0000	0x0000	0x0000	0x0000	0x0000
LEDR	0b1111	0b1111	0b1111	0b1111	0b1111

Under cykel 20 sker exekvering av LOAD instruktionen där R0 laddas med argumentet 0xA. PC inkrementerades under decode-fasen av LOAD_R0 #A. Nästa cykel (21) är Fetch_1, och då är PC kopplad till adressbusesen, och då bus_en_n och RD_n går från låg till hög laddas IR med op-koden 0x2010, vars mnemonic är STORE_R0 #10. Under decode (cykel 24) inkrementeras PC återigen.

2.4 Cykler 25-29

A OV											ŀ
◆ CLK	1										
∓ -♦ SW	1	1									
reset_n	1										
II −◆ PC		000000									
- Addr_bus	0000	000100	00					0000001	1		
💠 bus_en_n	1										
♣ RD_n	1										
- → data_bus_in	0000	2010		0000						1001	
 → IR	0000	2010									
 → REG_0	0000	000A									
next_state	Fetc	STORE	1 st	STORE	2 state	Fetch 1	state	Fetch 2	state	Fetch 3	state
♦ WE_n	1										
	0000	000A									
■→ LEDR	1111	1111				1010					
V											
CLK PERIOD		25		2	26		27		28		29
SW		1		_	1		1		1		1
reset n		1			1		1		1		1
PC	0b00	000011	0b	0000001	∣1	0b000	000011	0b0000	00011	0b000	000011
Addr_bus		010000		0001000			10000	0b0000			000011
bus_en_n	0000	1	00000100000		0	1		05000	0	05000	0
RD_n		1		1		1		0		0	
data_bus_in	0x2010			0x000)O	()x0000	0:	k0000	C	0x1001
IR		0x2010 0x2010		0x201)x2010		k2010		0x2010
REG 0		0x2010 0x000A		0x000			x000A		(000A)x000A
next_state		ORE_1		STORE_			etch_1		tch_2		etch_3
WE n	31		•	JIONE_	<u>_</u>	Г	1	ге	1	Г	اد_انا_ع 1
					U				- 1		- 1
-	,	ו ^ חחח אר		0,,000	۸	^		٥.	۸ ۵۵۵۸	^	×000 A
data_bus_out LEDR		0x000A 0b1111		0x000 0b111			x000A 0b1010		000A 01010)x000A)b1010

Under Execute_STORE (cykel 25) kopplas IR:s argument 0x10 till Adressbussen och Reg_0 läggs ut på data_bussen. Under cykel 26 görs data giltlig genom att WE_n går låg. Senast cykel 27 lyser dioderna enligt bitmönstret 0b1010 (0xA).

PC har varit inkrementerad sedan flera klockcykler tidigare. Under Fetch_1 (cykel 28) koppas PC ut till Adressbssen och bus_en_n och RD_n går då återigen samtidigt låga. IR kommer laddas under Fetch_3 (cykel 30, se nästa avsnitt).

2.5 Cykler 30 - 34

◆ CLK	1									
→ SW										
✓ reset_n										
II - △ PC	0000	011	0000010	00	+		_		_	
	0000		333332				000001	00		
bus_en_n	1									
	1	_								
→ data_bus_in	0000 1001	+-			+				2010	
→ IR	0000 1001									
REG_0	0000 000A				0001					
next_state		de state	Execute	LO	Fetch 1	state	Fetch 2	state	Fetch 3	state
♦ WE_n	1									
- data_bus_out	0000 000A									
II - → LEDR	1111 1010									
OLIV DEDIOD	,	_	0.4			00		00		0.4
CLK_PERIOD	3	0	31			32		33		34
SW		1				1		1		1
reset_n	01-000000	1	-00000400	l 、	05000	1	050000	1	050000	1
PC	0b0000001		0b00000100		0b00000100		0b0000		0b0000	
Addr_bus	0b0000001	1 0	0b00000011		0b00000011		0b0000	00100	0b0000	01000
bus_en_n		1				1		0		0
RD_n	0.400	1	0×1001		0	1 41001	0.	1001	^	x2010
data_bus_in IR	0x100		0x1001 0x1001			x1001 x1001		1001		
	0x 100									x1001
REG_0			0x000A cute_LOAE			x0001 etch 1		:0001 :ch 2		x0001
next_state WE n	Decoc	E ⊏XEC 1	ute_LOAL	, I	ге	1	rei	.UI_Z 1	ге	etch_3
data_bus_out	0x000	^	0x000A		0	ا 000A	∩∨	000A	0	и к000А
LEDR	0x000		0b1010			b1010		1010		b1010
LLDN	0010	U	יו טו עט	,	U	01010	UL	1010	U	01010

Op-koden som IR håller är 0x1001 vars mnemonic är LOAD_R0 #1. Under decode (cykel 31) inkrementeras PC. Under Execute_LOAD (cykel 32) laddas Reg_0 med OP-kodens argument. Under Fetch_1 (cykel 33) kopplas PC till adressbussen bus_en_n och RD_n går då låga. Nästa instruktion som är resultatet av adresseringen av kod-minnet läggs ut på kod-bussen under cykel 34, och läses in till IR under cykel 36.

2.6 Cykler 35-39

◆ CLK	1				
-🔷 SW	1 1				
reset_n	1				
-🔷 PC	0000	0100 00000101			
-📤 Addr_bus	0000	0100	00010000		
📤 bus_en_n	1				
♠ RD_n	1				
data_bus_in	0000 2010			10000	
-🍑 IR	0000 2010				
- REG_0	0000 0001				
next_state		de s Execute ST	. ISTORE 1 st	ate ISTORE 2 sta	te Fetch 1 state
- WE_n	1				
-🚣 data_bus_out	0000 000A		0001		
	1111 1010				0001
*					
CLK_PERIOD	35	36	37	38	39
SW	1	1	1	1	1
reset_n	1	1	1	1	1
PC	0b00000100	0b00000101	0b00000101	0b00000101	0b00000101
Addr_bus	0b00000100	0b00000100	0b00010000	0b00010000	0b00010000
bus_en_n	1	1	1	0	1
RD_n	1	1	1	1	1
data_bus_in	0x2010	0x2010	0x2010	0x0000	0x0000
IR	0x2010	0x2010	0x2010	0x2010	0x2010
REG 0	0x0001	0x0001	0x0001	0x0001	0x0001
next_state	Decode	Execute_STORE	STORE 1	STORE 2	Fetch 1
WE n	1	_ 1	1	_0	_ 1
_ data_bus_out	0x000A	0x000A	0x0001	0x0001	0x0001
LEDR	0b1010	0b1010		0b1010	0b0001

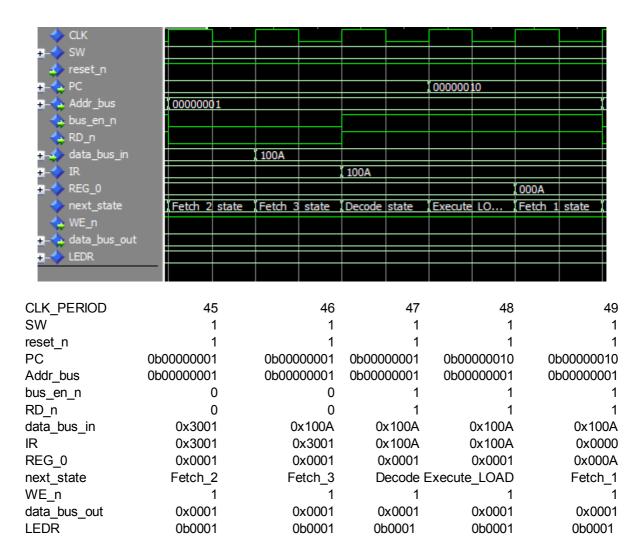
Inläsningen till IR innebär att op-koden 0x2010, vilket är menomic för STORE_R0 #10. Under deode inkrementeras PC (cykel 36), och Execute_STORE görs under cykel 37. Då kopplas IR:s argument till adressbussen och REG_0 läggs ut på databussen. WE_n går låg under följande cykel (38) och senast cykel 39 lyser lysdioderna i enlighet med vad som lades ut på databussen dvs. 0b0001.

2.7 Cykler 40 - 44

CLK SW reset_n PC Addr_bus bus_en_n Ablanta_bus_in REG_0 next_state WE_n LEDR	1		X 3001	(00000110 e (Execute JMP	(00000001), (Fetch 1 state)
CLK_PERIOD SW reset_n PC Addr_bus bus_en_n RD_n data_bus_in	40 1 1 0b00000101 0b00000101 0 0 0x0000	41 1 0b00000101 0b00000101 0 0 0x3001	42 1 1 0b00000101 0b00000101 1 1 0x3001	43 1 1 0b00000110 0b00000101 1 1 0x3001	44 1 0b00000001 0b00000101 1 1 0x3001
IR REG_0 next_state WE_n data_bus_out LEDR	0x2010 0x0001 Fetch_2 1 0x0001 0b0001	0x2010 0x0001 Fetch_3 1 0x0001 0b0001	0x3001 0x0001 Decode 1 0x0001 0b0001	0x3001 0x0001 Execute_JMP 1 0x0001	0x3001 0x0001 Fetch_1 1 0x0001 0b0001

Under Fetch_1 (cykel 40) kopplas PC till adressbussen och bus_en_n och RD_n går båda låga. Nästa instruktion, dvs. den på adress adresserad av PC finns på kodbussen under cykel 41, och inläsning till IR sker under nästa cykel (42). Op koden är 0x3001 vars mnemonic är JMP #1. Under decode inkrementeras PC (cykel 43), men under Execute_JMP (cykel 44) laddas PC med instruktionens argument dvs. IR:s argument-del kopplas till PC.

2.8 Cykler 45-49



Under Fetch_1 (cykel 45) kopplas PC till Adressbussen och bus_en_n och RD_n går båda låga samtidigt. Instrutkionen som läses in till IR (cykel 47) är från adress 1 i kodminnet LOAD_R0 #A med OP-kod 0x100A, som exekverats förut.

Under decode (cykel 48) inkrementeras PC, och under Execute_LOAD (cykel 49) har Reg_0 laddats med argument delen av IR.