Laboration 4

Controller

I denna laboration skall Controllern, d.v.s. styrenheten, för CPU:n skapas. Dessutom skall en enklare tillståndsmaskin i form av en 16-Counter konstrueras och laddas ned på FPGA.

1. Förberedelseuppgifter

Förberedelseuppgifter ska redovisas när laborationen börjar!

1. Konstruera en 16-Counter som räknar varje förändring av en insignal STEP. Countern skall triggas av FPGA-kortets interna snabba klocksignal(CLK), men bara räkna upp om STEP har förändrats. Counterns utvärden skall visas på en sjusegmentdisplay.
2. Skapa en modell av Controllern enligt specifikationen i Bilaga. Använd följande VHDL-skelett.

```
ENTITY Controller IS
    PORT(  adr        : OUT address_bus;          -- unsigned
           data       : IN program_word;          -- unsigned
           rw_RWM     : OUT std_logic;            -- read on high
           RWM_en     : OUT std_logic;            -- active low
           ROM_en     : OUT std_logic;            -- active low
           clk        : IN std_logic;
           reset      : IN std_logic;             -- active high
           rw_reg     : OUT std_logic;            -- read on high
           sel_op_1   : OUT unsigned (1 downto 0);
           sel_op_0   : OUT unsigned (1 downto 0);
           sel_in     : OUT unsigned (1 downto 0);
           sel_mux    : OUT unsigned (1 downto 0);
           alu_op     : OUT unsigned (2 downto 0);
           alu_en     : OUT std_logic;            -- active high
           z_flag     : IN std_logic;             -- active high
           n_flag     : IN std_logic;             -- active high
           o_flag     : IN std_logic;             -- active high
           out_en     : OUT std_logic;            -- active high
           data_imm   : OUT data_word);           -- signed
    END ENTITY Controller;
```

Figur 1. VHDL-skelett för Controller

3. För att öka läsbarheten i VHDL-koden och underlätta avläsningen av simuleringsresultat samt inte minst möjliggöra en latchfri implementation i en FPGA skall programräknare (PC) och instruktionsregister (IR) definieras samt uppdateras synkront.

   Därutöver kan läsbarheten ökas ytterligare genom att använda ALIAS-begreppet i VHDL.

2. Arbetsgång

1. Simulera Countern tills ni är säkra på att den fungerar enligt specifikationen.
2. Syntetisera samt ladda ner denna Countern på FPGA och redovisa en fungerande konstruktion.
3. Simulera Controllern tills ni är säkra på att den fungerar enligt specifikationen.
4. Prova att Controllern går att syntetisera mot komponenten (EP20K200EFC484-2X). Observera eventuella varningar och studera strukturerna i RTL-view. Speciell vikt skall därvid läggas på att undvika oavsiktliga latchar i kretsen.

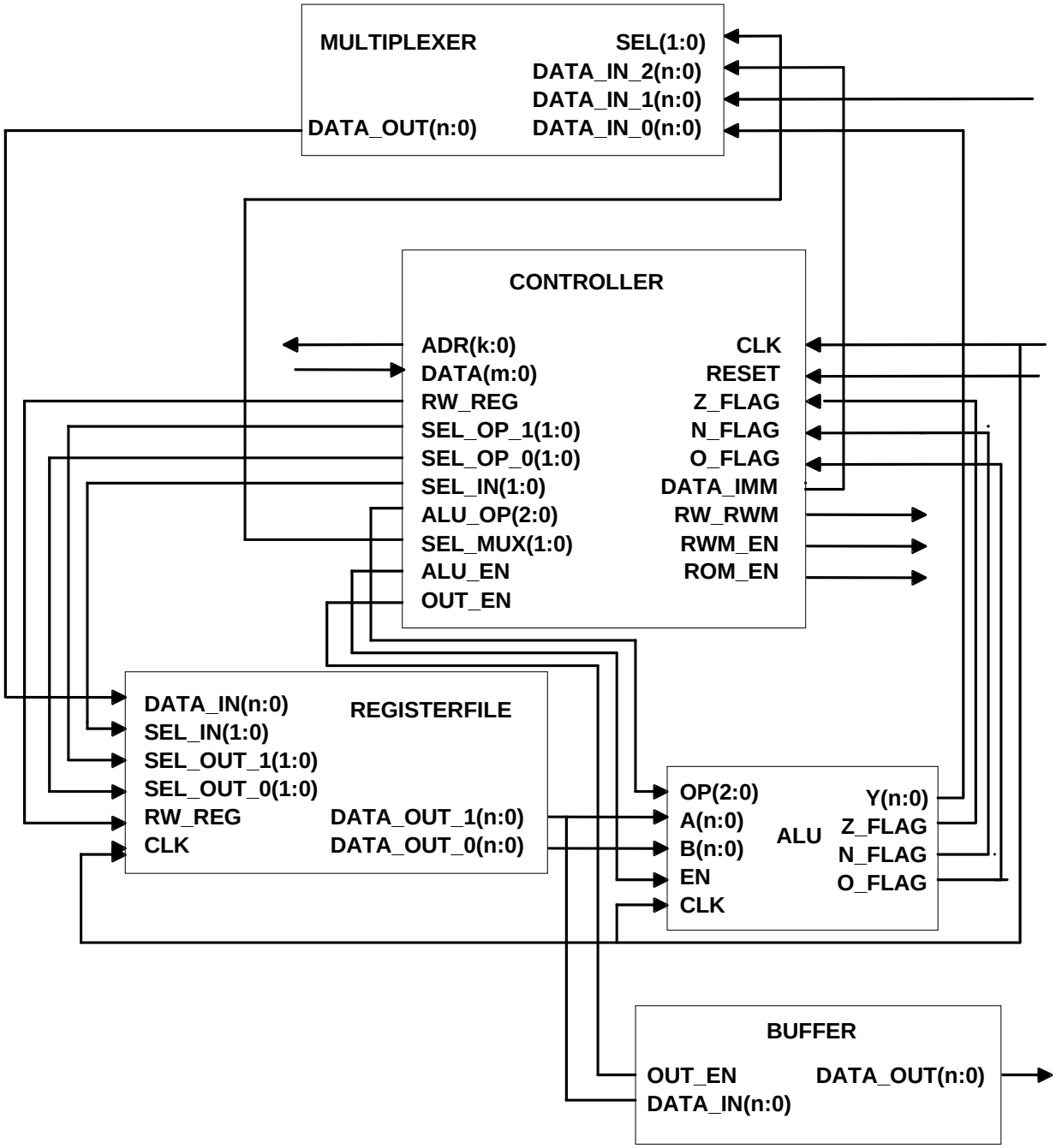
3. Redovisning

För en godkänd laboration ska följande visas:

- en väl dokumenterad Counter-modell som uppfyller funktionskraven och fungerar på FPGA.
- en väl dokumenterad Controller-modell som uppfyller funktionskraven enligt Bilaga.

Bilaga

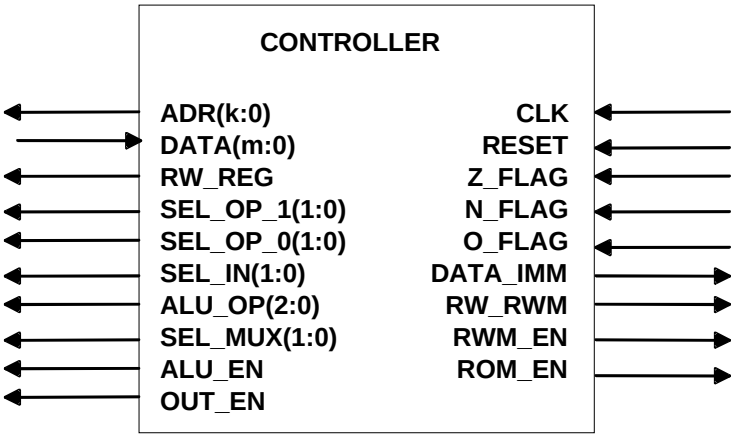
Mikroprocessorn Electrum består av flera delsystem som är beskrivna i de följande avsnitten. Mikroprocessorns uppbyggnad visas i Figur 2



Figur 2. Mikroprocessorn Electrums uppbyggnad

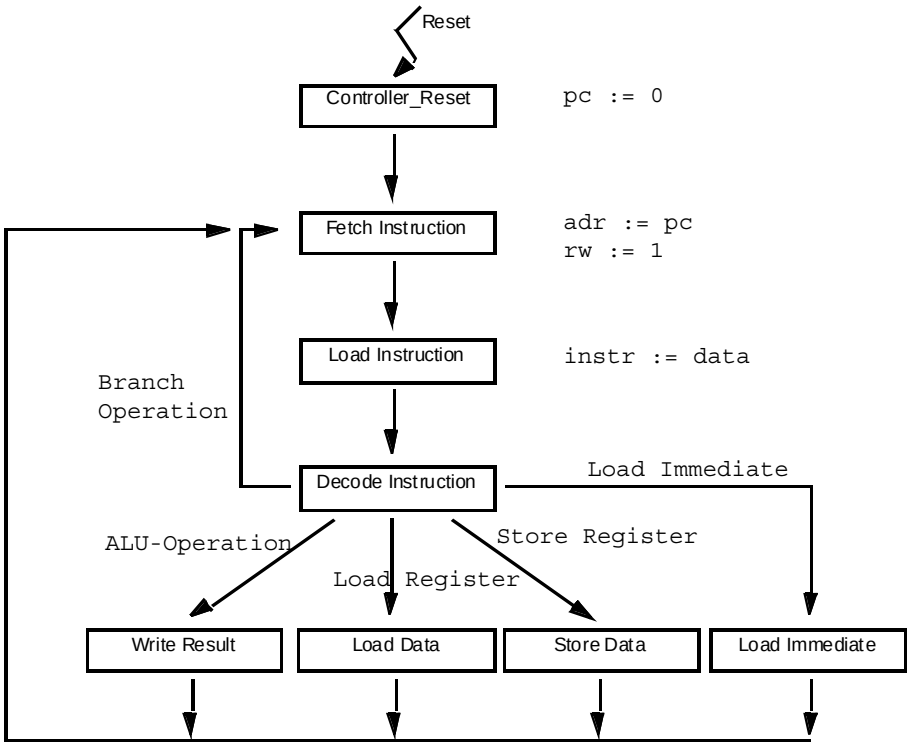
Controller

Controllern har som uppgift att koordinera de olika delsystemen och minneshanteringen. Controllern inkluderar en programräknare och ett instruktionsregister. Controllerns blockdiagram finns i Figur 3.



Figur 3. Blockdiagram Controller

Controllern ska implementeras som en tillståndsmaskin som kan tänkas arbeta enligt följande.



Figur 4. Tillståndmaskin Controller

Controllern ska implementera följande instruktioner enligt Tabell 1.

Instruktion	I_9	I_8	I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	Förklaring
ADD r1, r2, r3	0	0	0	0	r1		r2		r3		r3 = r1 + r2 pc = pc + 1
SUB r1, r2, r3	0	0	0	1	r1		r2		r3		r3 = r1 - r2 pc = pc + 1
AND r1, r2, r3	0	0	1	0	r1		r2		r3		r3 = r1 AND r2 pc = pc + 1
OR r1, r2, r3	0	0	1	1	r1		r2		r3		r3 = r1 OR r2 pc = pc + 1
XOR r1, r2, r3	0	1	0	0	r1		r2		r3		r3 = r1 XOR r2 pc = pc + 1
NOT r1, r3	0	1	0	1	r1		0	0	r3		r3 = NOT r1 pc = pc + 1
MOV r1,r3	0	1	1	0	r1		0	0	r3		r3 = r1 pc = pc + 1
LDR r1, mem	1	0	0	0	r1		mem				r1 = <mem> pc = pc + 1
STR r1, mem	1	0	0	1	r1		mem				<mem> = r1 pc = pc + 1
LDI r1, $d_3d_2d_1d_0$	1	0	1	0	r1		d_3	d_2	d_1	d_0	r1 = $d_3d_2d_1d_0$ pc =pc+1
NOP	1	0	1	1	0	0	0	0	0	0	pc = pc + 1
BRZ mem	1	1	0	0	0	0	mem				z = 1 -> pc = mem z = 0 -> pc = pc + 1
BRN mem	1	1	0	1	0	0	mem				n = 1 -> pc = mem n = 0 -> pc = pc + 1
BRO mem	1	1	1	0	0	0	mem				c = 1 -> pc = mem c = 0 -> pc = pc + 1
BRA mem	1	1	1	1	0	0	mem				pc = mem

Tabell 1. Instruktionskoder. Obs! r1, r2, r3 är adresser till något av registren R0-R3