Laborator 11

Proiectarea în VHDL a unui microprocesor pe 8-biți – partea a IV-a

1. **Elaborarea programului de testare a microprocesorului**
2. **Descriere memorie RAM utilizând utilitarul IP Core**
3. **Conectare microprocesor cu memorie RAM şi elaborare fişier UCF**
4. **Simulare, implementare şi testare în FPGA**

1. Elaborarea programului de testare a microprocesorului

Pentru verificarea proiectului este necesar un program de testare. Acest program face adunarea unei liste de numere şi este prezentat în tabelul T.1. Se poate observa că programul începe la adresa 02 din memoria RAM, conform celor expuse în cursul 10.

Tabelul T.1 Programul de testare a microprocesorului

Adresa	Opcod	Operand	Mnemonică	Comentariu
0002	40	00	load #0	; R1<- adresa de start a şirului
0004	01		store R1 ;	de numere din zona de DATE
0005	40	10	load #10 ; R2<- numărul de valori din şir	
0007	02		store R2	
0008	40	00	load #0	;R3<- 0 , şterge registru de însumare (*sum*)
000A		03	store R3	
		loop:		; etichetă
000B	29		load (R1)	; ACC<- următorul nr. din listă şterge
000C	A0		clear_c	; rezultatul anterior
000D	8B		add R3	; adună conținutul reg. *sum* cu ACC
000E	03		store R3 ;stochează rezultatul înapoi în reg. *sum*	
000F	40	01	load #1	; incrementează R1 astfel încât să indice
0011	A0		clear_c	; următorul nr. din listă
0012	89		add R1	
0013	01		store R1	
0014	40	FF	load #FF; decrementează R2 pt. că tocmai s-a	
0016	A0		clear_c	; adăugat încă un număr
0017	8A		add R2	
0018	02		store R2	
0019	40	FF	load #FF; acum, testează R2 dacă este zero	
001B	92		test R2	; (ex. pt. a vedea dacă şirul de nr. este gata)
001C	C8	20	jz *afişează*	; ieşi din buclă (loop) dacă numărarea
001E	D0	0B	jump *loop*	; a luat sfârşit dacă nu continuă procesarea
		afişează:	; etichetă	
0020	23		load R3	; ACC <- stochează suma în reg. R3
0021	68		out	;afişează biții sumei folosind segm. LED
		halt:		; etichetă
0022	D0	22	jump *halt*	; execută această instr. şi stai în buclă

Opcodurile instrucțiunilor vor fi scrise în format hexazecimal, după cum este prezentat în tabelul T2.

Tabelul T.2 Conținutul memorie RAM

- zona de program de la adresa 0002
0002 - 40 00 01 40 10 02 40 00 03 29 A0 8B 03 40 01 A0 89 01 40 FF A0 8A 02 40 FF 92 C8 20 D0 0B 23 68 D0 22
- zona de date de la adresa 512 (0200 în hexa)
0200 - 01 23 45 67 89 AB CD EF FE DC BA 98 76 54 32 10

În prima parte a tabelului sunt cuprinse opcodurile instrucțiunilor şi operanzii programului, aceste valori hexazecimale vor fi încărcate în zona de PROGRAM a memoriei RAM, începând cu adresa 02.

Cele 16 valori hexazecimale din ultima linie vor fi încărcate în zona de DATE a memoriei începând cu adresa 0x0200, aceste valori vor forma şirul de caractere cu care va opera programul.

2. Descrierea memorie RAM utilizând IP Core

Atât pentru simularea funcțională cât şi pentru testarea in FPGA microprocesorul are nevoie de o memorie RAM. Doar pentru simularea funcțională această memorie ar putea fi descrisă ca fişier text I/O. Însă acest tip de descriere nu va fi util la implementare. Astfel că se va face o descriere a memoriei RAM utilizând utilitarul *Core Generator* din pachetul Xilinx ISE. Cu ajutorul acestui utilitar se va construi o memorie RAM cu capacitate de 3x256 = 768 octeți, acoperind astfel toate cele 3 zone de memorie: de program de stivă şi de date. Memoria astfel proiectată se va baza pe blocurile specializate de memorie RAM aflate în structura circuitului FPGA.

În cadrul proiectului final se adaugă un fişier sursă nou, *add new source -> IP (Core &...)*, se stabileşte un nume, ex. *RAM_768.*

- Se alege tipul de memorie ce se doreşte a fi generat, vezi figura 1, apoi *Next -> Finish;*
- Se stabileşte capacitatea memoriei, 768x8, vezi figura 2;
- Se trece cu *next* peste paginile 2 şi 3;
- În pagina 4 se stabileşte conținutul memorie, încărcându-se un fişier de configurare ce conține infrmațiile din tabelul T2, vezi figura 3;
- Se apasă butonul *Generate,* după care la proiect se va ataşa un fişier cu numele stabilit de noi şi extensia XCO.

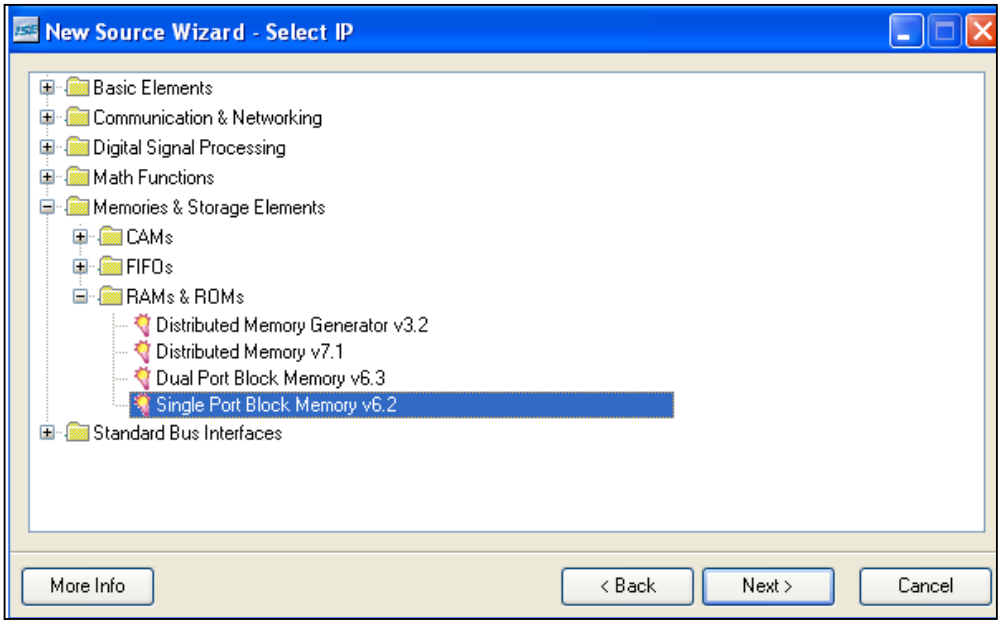

Figura 1

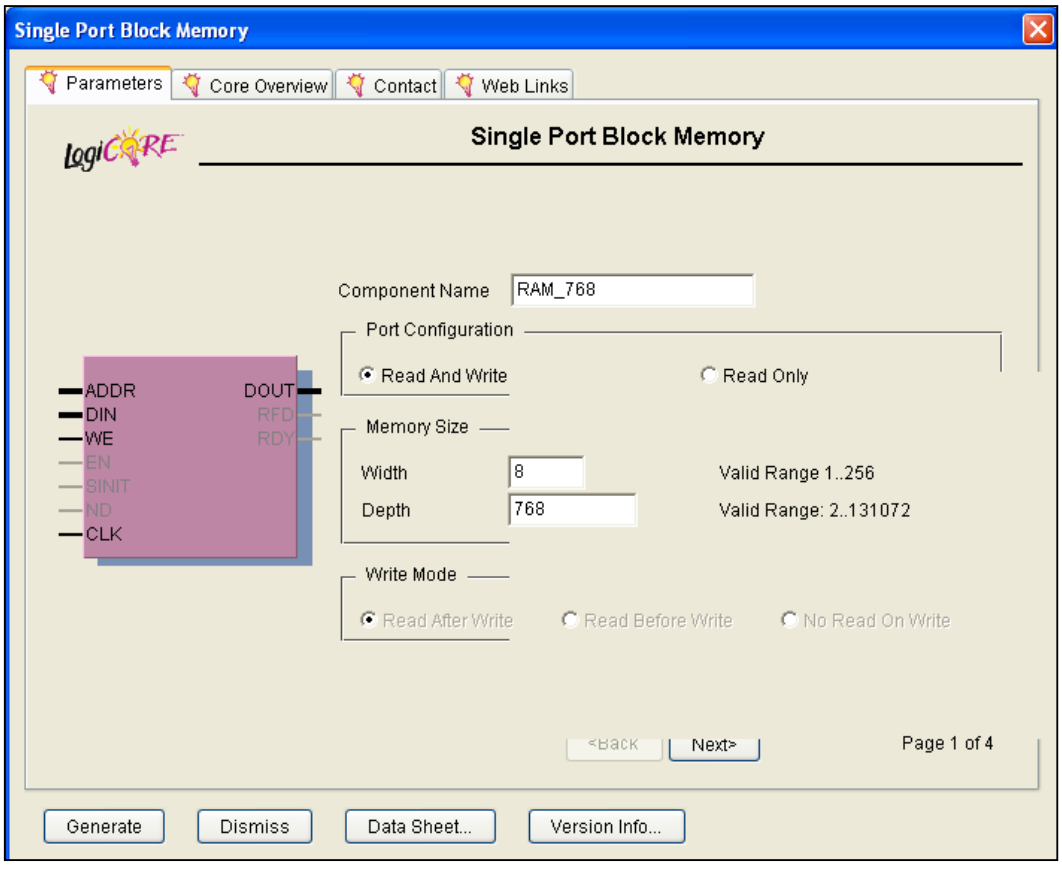

Figura 2

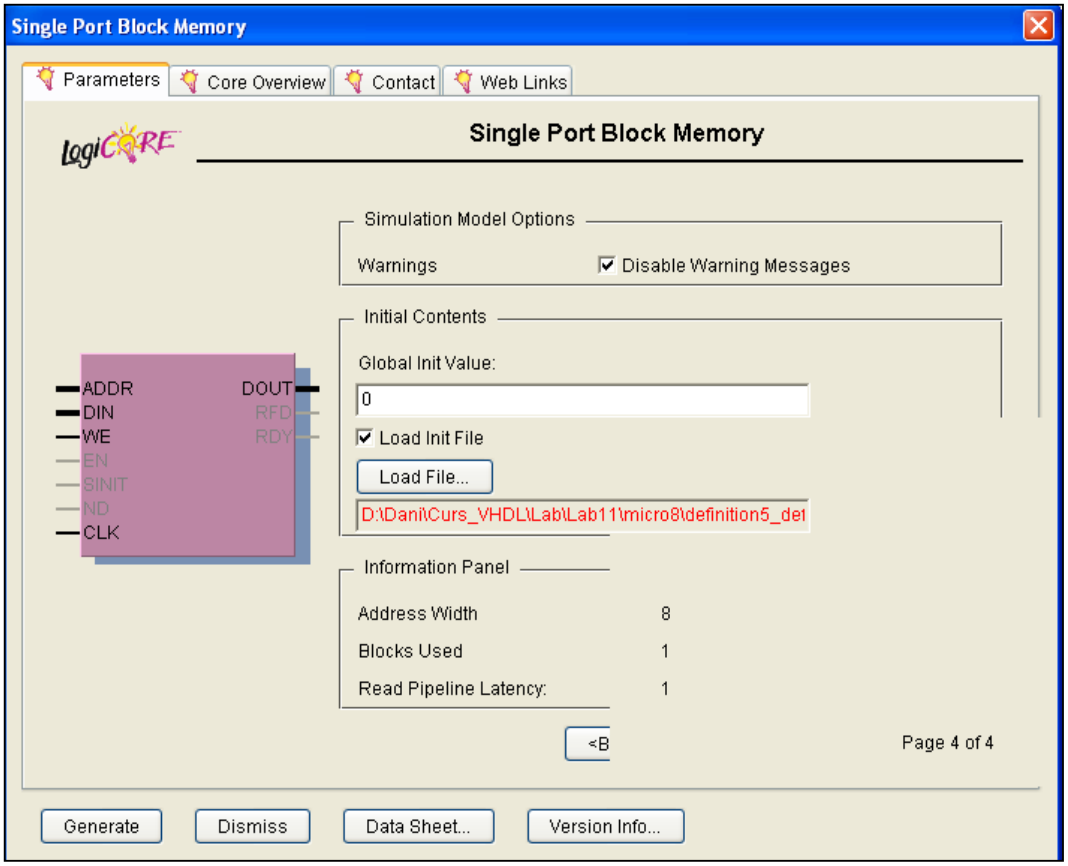

Figura 3

3. Conectare microprocesor cu memorie RAM şi elaborare fişier UCF

După ce au fost asamblate toate blocurile microprocesorului se va genera un simbol în schematic pentru micropocesor.

Memoria generată ca şi core IP se va regăsii şi în foaia de schematic unde va fi conectată la microprocesor., conform figurii 4.

În tabelul T.3 este prezentată structura fişierului UCF. Se poate observa că portul de intrare este conectat la comutatoare, portul de ieşire la LED-uri, observați semnalul de validare LEDG conectat la VCC, figura 4. Intrarile Reset şi INT (întreruperea) sunt conectate la butoanele *BTN0* şi *BTN1* de pe placa DIO 4.

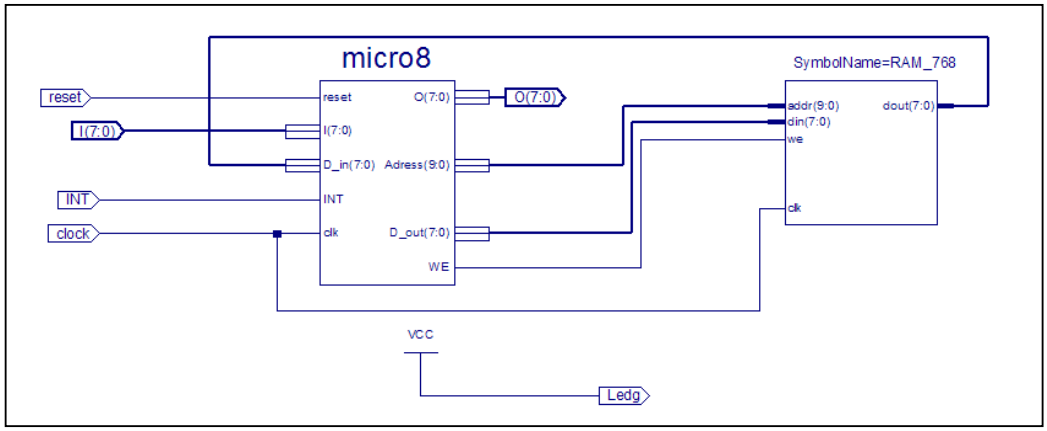

Figura 4

Tabelul T.3 Conţinutul fişierului UCF

```
NET "clock"  LOC = "P182"  ;
NET "I<0>"  LOC = "P23"  ;#sw0-sw7
NET "I<1>"  LOC = "P21"  ;
NET "I<2>"  LOC = "P18"  ;
NET "I<3>"  LOC = "P16"  ;
NET "I<4>"  LOC = "P11"  ;
NET "I<5>"  LOC = "P9"  ;
NET "I<6>"  LOC = "P7"  ;
NET "I<7>"  LOC = "P5"  ;
NET "INT"  LOC = "p206"  ;#Btn1
NET "O<0>"  LOC = "P111"  ;#Led0-7
NET "O<1>"  LOC = "P109"  ;
NET "O<2>"  LOC = "P102"  ;
NET "O<3>"  LOC = "P100"  ;
NET "O<4>"  LOC = "P98"  ;
NET "O<5>"  LOC = "P96"  ;
NET "O<6>"  LOC = "P94"  ;
NET "O<7>"  LOC = "P89"  ;
NET "reset"  LOC = "P3"  ;#Btn0
NET "Ledg"  LOC = "P45"  ;
```

4. Simulare, implementare şi testare în FPGA

În continuare se va sintetiza întregul proiect şi se vor citii fişierele raport, urmărindu-se resurse FPGA consumate, viteza de lucru, etc. Fişierul *.bit* care a rezultat în urma sintezei va fi uploadat în placă.

În timpul rulării programul nu va afişa valori intermediare pe afişajul 7-segmente. Afişarea unei valori va avea loc doar când programul va fi rulat complet. Rezultatul însumării şirului de valori este 0x7F8 (sistem hexazecimal) şi va fi păstrat în ACC. Ultima comandă a programului va trimite acest rezultat trunchiat (doar F8) la portul de ieşire sub forma „11111000”.

Teme

- Se va extinde setul de instrucțiuni, cu instrucțiuni de incrementare, decrementare, înmulțire, împărțire şi eventual comparare.
- Se va dezvolta un program pentru tratarea întreruperilor. Indicație: la adresa 00 din zona de PROGRAM se va încărca o instrucțiune de salt urmată de adresa de la care va începe propriu-zis ISR-ul.
- Să va dezvolta un program asamblor simplu care să traducă instrucțiunile microprocesorului reprezentate ca şi mnemonici, în opcoduri (format hexazecimal).