1

Proiectarea dispozitivelor aritmetice în virgulă mobilă

1. Prezentare teoretică

În cadrul acestui laborator se vor prezenta algoritmii hardware pentru implementarea operaţiilor de adunare, scădere, înmulţire şi împărţire a numerelor în virgulă mobilă utilizând conceptele benzii de asamblare.

Banda de asamblare

Banda de asamblare este o tehnică de descompunere a proceselor secvenţiale în suboperaţii, fiecare suboperaţie putând fi executată într-un segment special dedicat şi în paralel cu alte suboperaţii. Rezultatul obţinut în cadrul fiecărui segment este transferat următorului segment din banda de asamblare. La rezultatul final se ajunge atunci când datele au trecut prin toate segmentele benzii de asamblare.

Suprapunerea calculelor este posibilă datorită asocierii unui registru fiecărui segment al benzii de asamblare. Registrele au rolul de a izola segmentele astfel încât fiecare segment al benzii de asamblare poate opera pe date distincte simultan cu operarea altor segmente.

Exemplu Se doreşte aflarea rezultatul expresiei: $A_i \cdot B_i + C_i$ pentru $i = \overline{1 \ldots 7}$.

Soluţie Fiecare operaţie poate fi implementată în câte un segment al benzii de asamblare. Fiecare segment conţine unul sau două registre precum şi un circuit combinaţional aşa cum se poate observa în figura 1. Registrele R_1 ... R_5 primesc date noi la fiecare apariţie a frontului de ceas. Circuitul de înmulţire precum şi sumatorul din figura 1 sunt circuite combinaţionale.

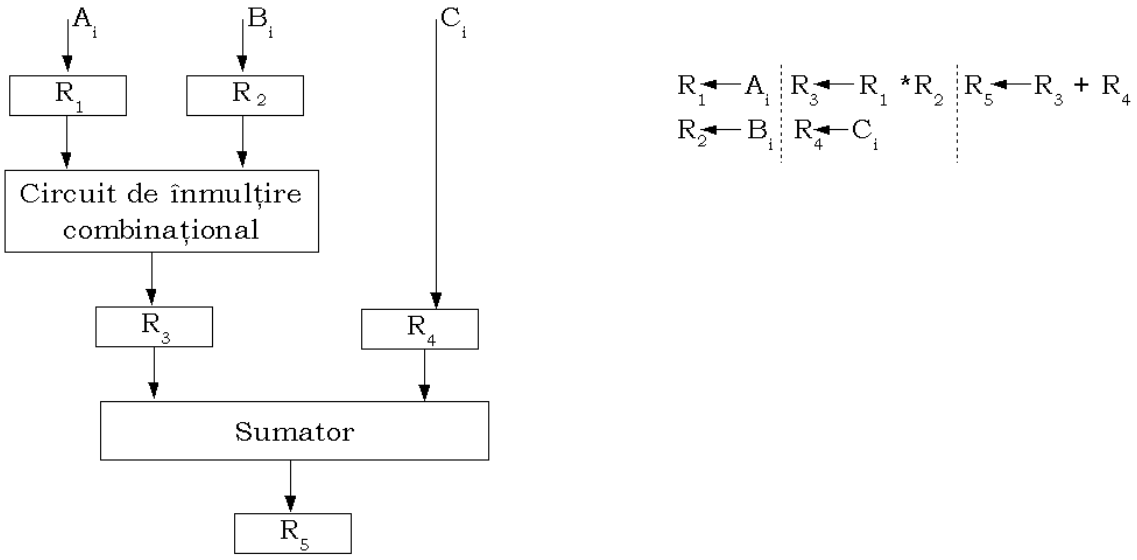

Figura 1: Calculul expresiei $A_i \cdot B_i + C_i$ folosind banda de asamblare.

Conţinutul registrelor benzii de asamblare este prezentat în tabelul 1.

Ceas	Segmentul 1		Segmentul 2		Segmentul 3
	R_1	**R_2**	**R_3**	**R_4**	**R_5**
1	A_1	B_1			
2	A_2	B_2	$A_1 \cdot B_1$	C_1	
3	A_3	B_3	$A_2 \cdot B_2$	C_2	$A_1 \cdot B_1 + C_1$
4	A,	B_4	$A_3 \cdot B_3$	C_3	$A_2 \cdot B_2 + C_2$
5	A_5	B_5	$A_4 \cdot B_4$	C_4	$A_3 \cdot B_3 + C_3$
6	A_6	B_6	$A_5 \cdot B_5$	C_5	$A_4 \cdot B_4 + C_4$
7	A_7	B_7	$A_6 \cdot B_6$	C_6	$A_5 \cdot B_5 + C_5$
8			$A_7 \cdot B_7$	C_7	$A_6 \cdot B_6 + C_6$
9					$A_7 \cdot B_7 + C_7$

Tabelul 1: Conţinutul registrelor benzii de asamblare.

Reprezentarea numerelor în virgulă mobilă

Reprezentarea numerelor în virgulă mobilă face obiectul standardului IEEE 754. Conform acestui standard reprezentarea lor este alcătuită din două părţi. Prima parte reprezintă un număr cu semn denumit *mantisă.* Partea a doua specifică poziţia punctului zecimal şi se numeşte *exponent.*

Reprezentarea numerelor în virgulă mobilă poate fi făcută în precizie simplă sau dublă.

Folosirea preciziei simple impune o reprezentare a numărului utilizând 32 de biţi. Primul bit (bitul cel mai semnificativ) este bitul de semn. Dacă valoarea acestui bit este 0, numărul este pozitiv, altfel numărul este negativ. Următorii 8 biţi sunt alocaţi pentru valoarea exponentului, iar ultimii 23 de biţi reprezintă mantisa. Gama de reprezenatre în cazul preciziei simple este: $-1.8 \times 10^{-38} \div 3.40 \times 10^{38}$.

În cazul în care se foloseşte precizia dublă, numărul biţilor utilizaţi pentru reprezentarea numărului se dublează devenind 64. Primul bit (bitul cel mai semnificativ) este bitul de semn. Exponentul este reprezentat pe următorii 11 biţi, iar mantisa sau fracţia pe 52 de biţi.

Reprezentarea generală a unui număr în virgulă mobilă, folosind precizia simplă, este dată în ecuaţia 6.1.

$$(-1)^{s} \cdot (1 + MANTISA) \cdot 2^{Exponent - 127} \qquad (6.1)$$

Valoarea variabilei *Exponent* din cadrul ecuaţiei 6.1 este *127* pentru a se evita valori negative ale exponentului. Pentru precizia dublă, această valoare se modifică devenind 1023.

Adunarea şi scăderea în virgulă mobilă

Resursele hardware necesare implementării acestor operaţii sunt prezentate în figura 2.

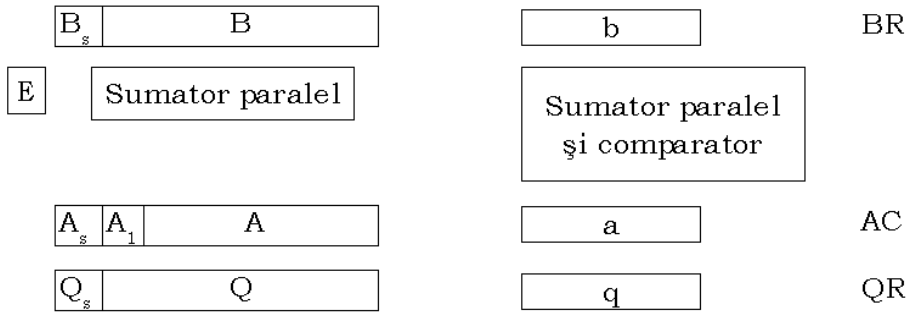

Figura 2: Resursele hardware necesare adunării/scăderii a două numere în virgulă mobilă.

Algoritmul necesită parcurgerea următoarelor etape:

1. Se verifică dacă unul dintre operanzi este zero sau nu;
2. Se aliniază mantisele;
3. Se adună sau se scad mantisele;

4. Se normalizează rezultatul obţinut.

Algoritmul de adunare/scădere a două numere în virgulă mobilă utilizând tehnica bandă de asamblare poate fi observat în figura 3.

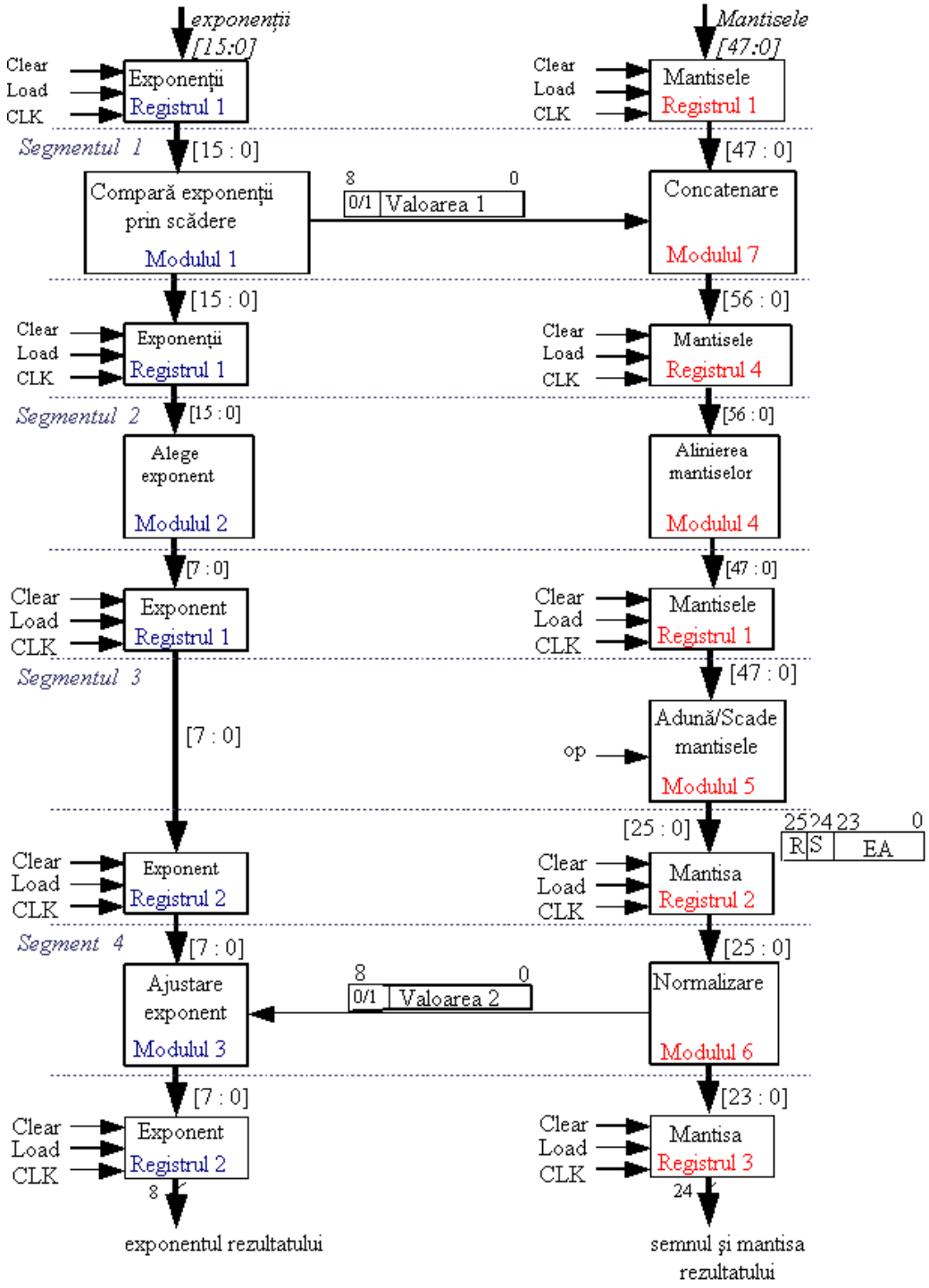

Figura 3: Adunarea şi scăderea a două numere reprezentate în virgulă mobilă.

În cadrul segmentului 1 al benzii de asamblare, exponenţii sunt comparaţi prin scădere.

Exponentul mai mare este ales ca exponent al rezultatului. Rezultatul diferenţei indică de câte ori mantisa asociată cu exponentul mai mic trebuie deplasată către dreapta. Astfel se realizează alinierea mantiselor.

Variabila *Valoarea1* memorează rezultatul comparării. Dacă bitul cel mai semnificativ al acestei variabile este 0, mantisa primului număr trebuie deplasată dreapta. Pentru a memora numărul de deplasări care trebuie efectuate, au fost alocaţi 8 biţi *(Valoarea1[7 : 0]*).

Dacă *Valoarea1[8]* este l, atunci mantisa asociată numărului 2 trebuie deplasată dreapta de un număr de ori egal cu valoarea lui *Valoarea1[7 : 0]*. Pentru implementarea sumatorului care apare în cadrul benzii de asamblare se recomandă utilizarea unui sumator cu transport anticipat (CARRY LOOK AHEAD).

În cadrul segmentului 3 al benzii de asamblare, cele două mantise sunt adunate sau scăzute în funcţie de valoarea semnalului *OP*. Dacă *OP = 0* atunci trebuie realizată operaţia de adunare şi, în consecinţă, cele două mantise trebuie adunate. Dacă *OP = 1,* operaţia de scădere trebuie realizată şi, în consecinţă, cele două mantise trebuie scăzute.

Rezultatul pasului de adunare/scădere a mantiselor se reprezintă pe 26 de biţi. Bitul cel mai semnificativ, bitul 25, este utilizat pentru a specifica dacă mantisa este egală sau nu cu zero. Restul de biţi este folosit pentru a reprezenta registrele *E* şi *A* concatenate.

Rezultatul este normalizat în cadrul segmentului 4 al benzii de asamblare. Când apare o depăşire superioară, mantisa-rezultat este deplasată dreapta şi exponentul este incrementat cu o unitate. Când apare o depăşire inferioară, numărul de zerouri din cadrul mantisei (poziţiile cele mai semnificative) determină numărul de deplasări stânga al mantisei, număr care trebuie scăzut din exponent. Memorarea acestui număr se face prin intermediul variabilei *Valoarea2*.

Dacă bitul cel mai semnificativ al variabilei *Valoarea1* este 0, atunci exponentul trebuie deplasat spre stânga, altfel exponentul trebuie să fie deplasat spre dreapta. Numărul de deplasări este indicat de către *Valoarea2[7 :* 0].

Rezultatele simulării acestui algoritm folosind mediul de dezvoltare Xilinx pot fi observate în figura 4.

Înmulţirea în virgulă mobilă

Operaţia de înmulţire în virgulă mobilă presupune realizarea următorilor paşi:

1. Se verifică dacă unul dintre numere este zero sau nu;

2. Se adună exponenţii;

3. Se înmulţesc mantisele;

4. Se normalizează produsul.

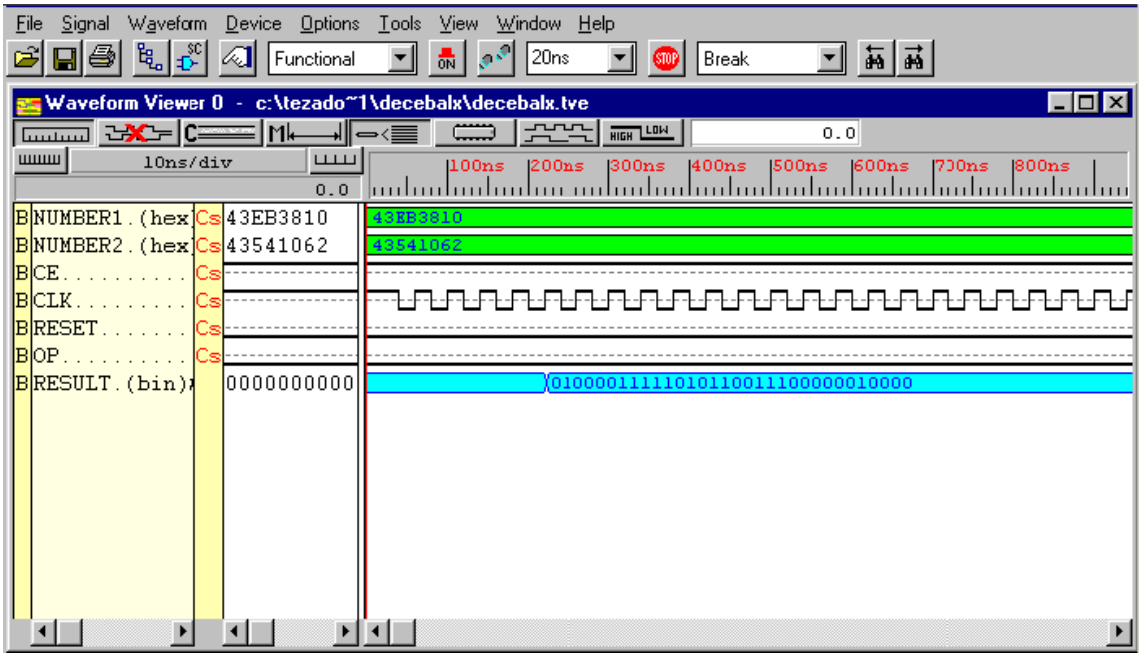

Figura 4: Rezultatele simulării algoritmului de adunare/scădere.

Implementarea în bandă de asamblare al acestui algoritm poate fi observată în figura 5.

Înmulţirea a două numere binare se realizează cu ajutorul unui circuit combinaţional (denumit matrice de multiplicare) care formează toţi biţii produsului odată. Aceasta este cea mai rapidă metodă deoarece timpul necesar efectuării operaţiei este egal cu timpul de propagare al semnalului luând în calcul ruta cea mai dezavantajoasă.

Circuitul *matrice de multiplicare* necesită un număr mare de porţi, implementarea lui este neeconomică, acesta fiind principalul motiv pentru care acest circuit nu a avut o răpândire foarte mare până la apariţia circuitelor integrate. Proiectarea unui asemenea circuit ale cărui intrări ar avea dimensiunile de j respectiv k biţi va necesita $j \times k$ porţi *ŞI* şi sumatoare de $(j-1)\cdot k$ biţi pentru a obţine un produs de *(j + k)* biţi.

Un exemplu de circuit matrice de multiplicare este prezentat în figura 6.

Trebuie remarcat că exponentul rezultat corect este obţinut prin scăderea valorii 127 din exponentul sumă.

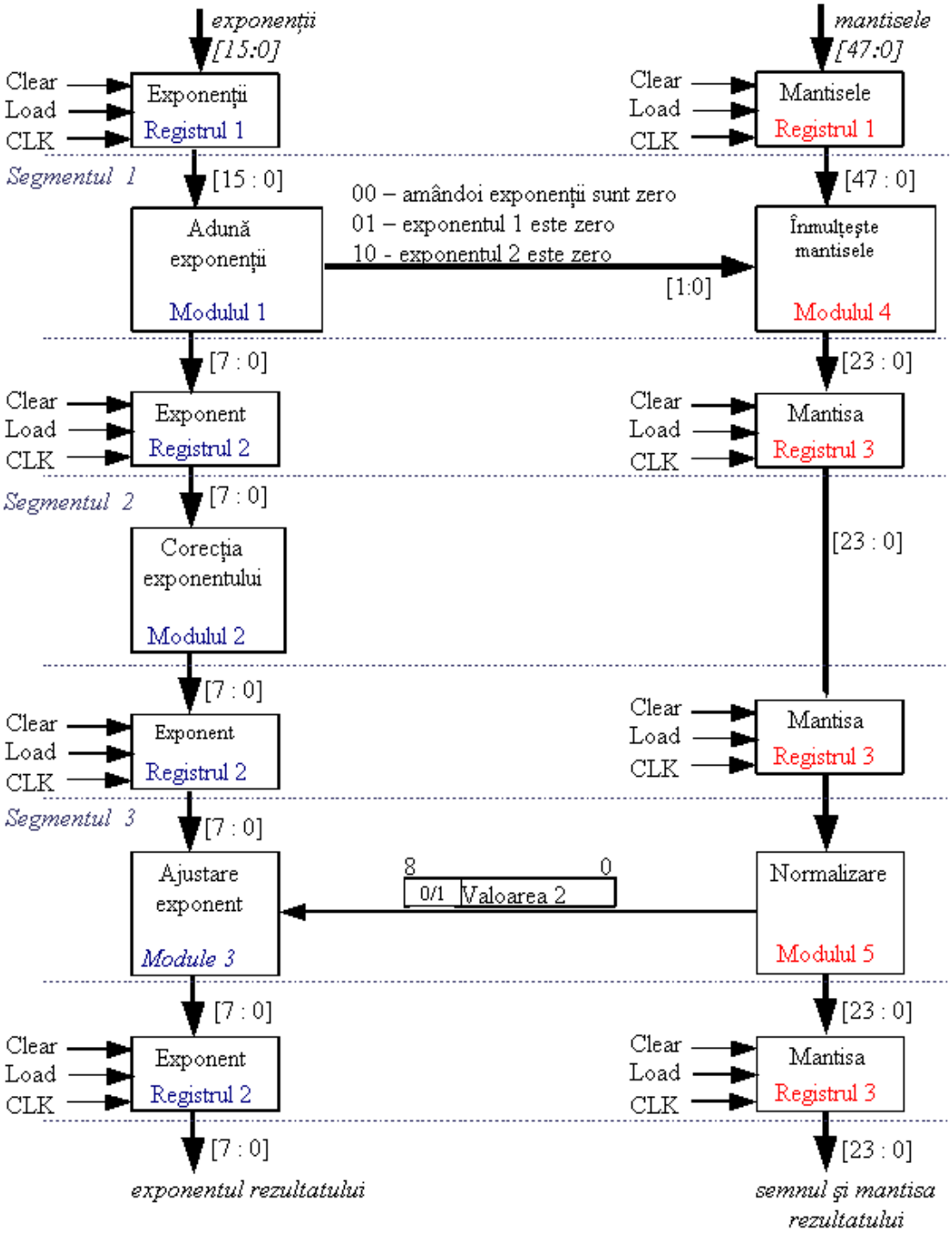

Figura 5: Algoritmul de înmulţire a două numere în virgulă mobilă.

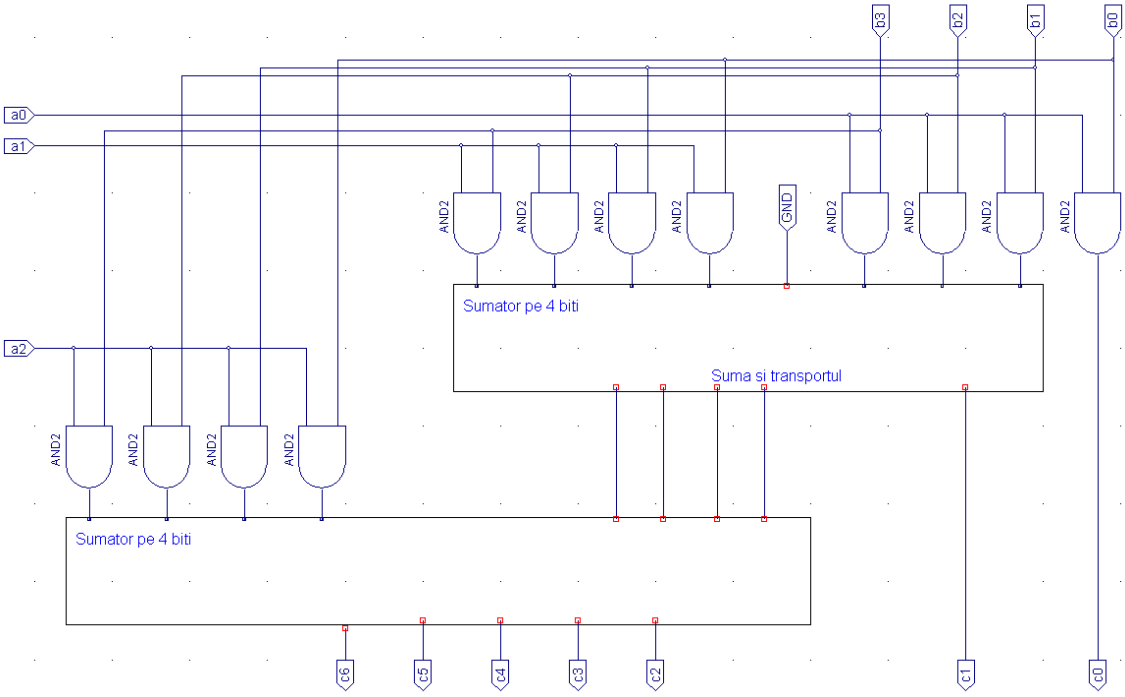

Figura 6: Circuitul matrice de multiplicare pentru *k = 4* şi *j = 3*.

Împărţirea în virgulă mobilă

Algoritmul de împărţire a două numere reprezentate în virgulă mobilă necesită parcurgerea următorilor paşi:

1. Se verifică dacă unul dintre cei doi operanzi este zero sau nu;
2. Se iniţializează registrele şi se evaluează semnul;
3. Se aliniază deîmpărţitul;
4. Se scad exponenţii;
5. Se împart mantisele.

Algoritmul în bandă de asamblare al acestei operaţii este prezentat în figura 7

Împărţirea a două numere în virgulă mobilă normalizate va produce întotdeauna un cât normalizat. Din acest motiv, spre deosebire de celelalte operaţii în virgulă mobilă, rezultatul operaţiei de împărţire nu necesită normalizare.

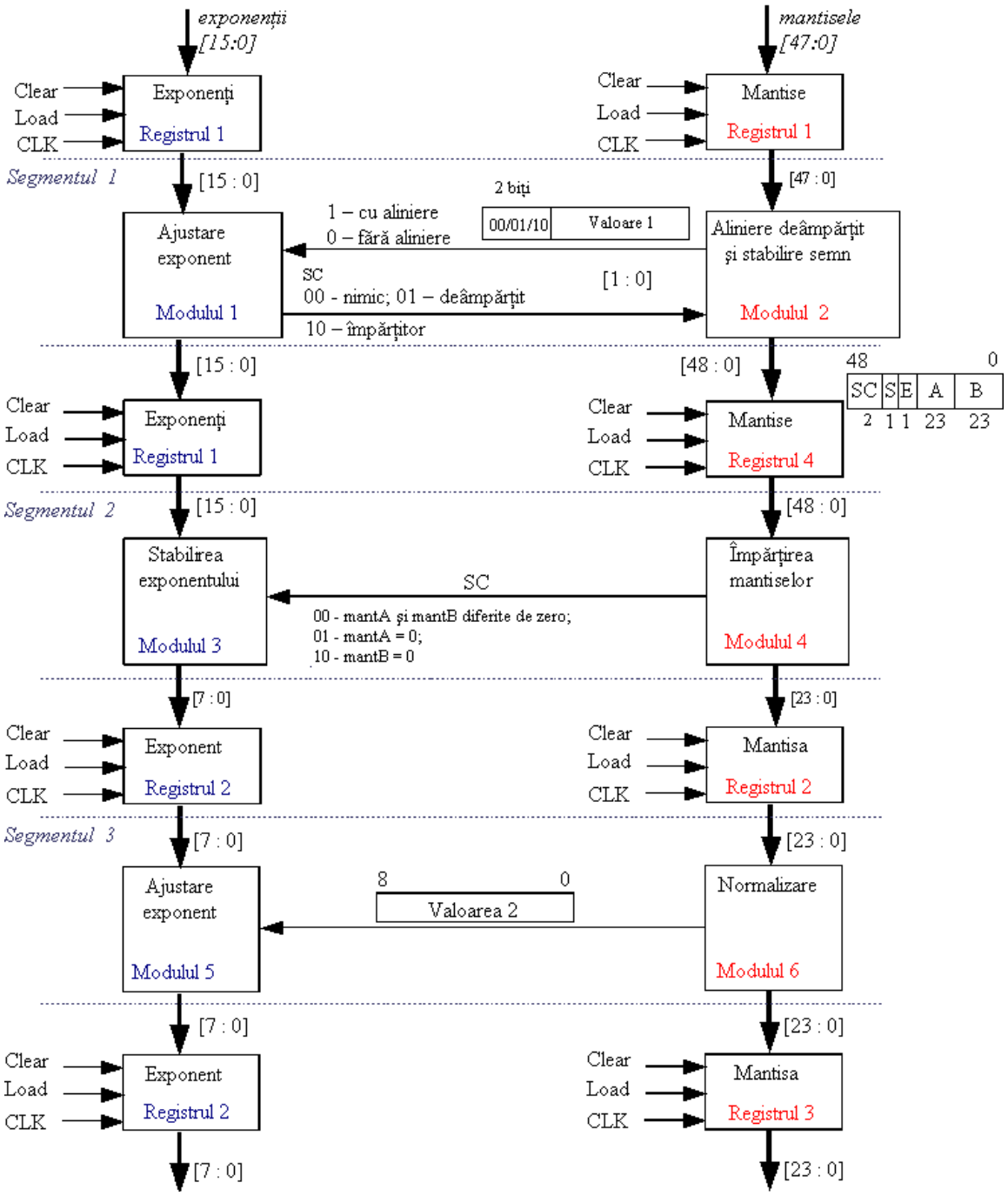

Figura 7: Algoritmul de împărţire a două numere în virgulă mobilă

2. Desfăşurarea lucrării

1. Se va proiecta în Verilog sau VHDL utilizând Xilinx WebPACK ISE 10.1 algoritmul de adunare/scădere a două numere reprezentate în virgulă mobilă, prezentat în figura 3.

2. Se va verifica funcţionarea algoritmului utilizând simulatorul ModelSim.

3. Se va utiliza circuitul Xilinx Spartan 3 XC3S400 (package FT256; speed -5) pentru a verifica timpii reali de funcţionare a circuitului proiectat.

3. Probleme propuse

1. Să se proiecteze şi simuleze utilizând XilinxWebPACK 10.1 şi limbajul Verilog un circuit combinaţional care să realizeze operaţia de înmulţire dintre două numere binare. Se va utiliza schema prezentată în figura 6;

2. Să se proiecteze şi simuleze utilizând Xilinx WebPACK 10.1 şi limbajul Verilog un circuit care să realizeze operaţia de înmulţire în virgulă mobilă precizie simplă prezentate în figura 5;

3. Să se proiecteze şi simuleze utilizând Xilinx WebPACK 10.1 şi limbajul Verilog un circuit care să realizeze operaţia de împărţire în virgulă mobilă precizie simplă prezentate în figura 7.