Lucrarea 10

Sinteza Dispozitivelor de Înmulțire în Radix Superior

Lucrarea urmăreşte o abordare practică, din punctul de vedere al designerului hardware, a problematicii sintezei dispozitivelor hardware de înmulțire în radix superior. Se pleacă de la considerațiile de natură algoritmică, ce cuprind inclusiv implicații la nivelul operațiilor elementare, cum sunt modificarea operanzilor şi a procedurii de shiftare. Apoi, se studiază modul în care o platformă hardware fixată poate fi folosită pentru a implementa algoritmul dat. Rezultă ordinograma (sau descrierea într-un Hardware Description Language) şi, în final, se trece la sinteza dispozitivului de control. Problemele propuse şi indicațiile aferente urmăresc consolidarea metodologiei de abordare a sintezei dispozitivelor aritmetice.

1 Algoritmul Booth Radix-4

În cele ce urmează, ne propunem sinteza unui dispozitiv hardware care să implementeze algoritmul Booth radix-4 pentru operanzi pe 8 biți. Vom folosi ca structură fixată, platforma hardware de tip Hayes.

În acest scop, se impune mai întâi o analiză de natură algoritmică a problemei date. Din start, pentru a lucra în baza 4, trebuie ca algoritmul nostru să producă 2 biți ai produsului parțial la fiecare iterație. Prin urmare, ideea este de a contopi 2 paşi ai algoritmului Booth simplu într-o singură iterație Booth radix-4. Astfel, vor fi investigați nu 2 ci 3 biți ai

înmulțitorului (două perechi adiacente de biți ai înmulțitorului), iar shiftarea aplicată produsului parțial, care – la Booth radix-2 – a fost shift dreapta cu 1 poziție, va fi în cazul de față o shiftare la dreapta de 2 poziții.

Concret, dacă algoritmul Booth Radix-2 are operațiile dictate iterației specificate prin Tabela 10.1, atunci la Booth Radix-4 specificarea iterației se face după cum se arată în Tabela 10.2; observația suplimentară este că fiecare iterație se termină printr-o shiftare la dreapta a produsului parțial, shiftare care este pe 1 simplă pentru radix-2 şi dublă pentru radix-4.

$Q[i]$	$Q[i-1]$	*Operation*
0	0	0
0	1	$+M$
1	0	$-M$
1	1	0

Tabela 10.1 Caracterizarea iterației algoritmului Booth radix-2.

$Q[i+1]$	$Q[i]$	$Q[i-1]$	*Operation*
0	0	0	0
0	0	1	$+M$
0	1	0	$+M$
0	1	1	$+2M$
1	0	0	$-2M$
1	0	1	$-M$
1	1	0	$-M$
1	1	1	0

Tabela 10.2 Caracterizarea iterației algoritmului Booth radix-4.

Elementele Tabelei 10.2 au fos obținute prin inspectarea perechilor ($Q[i+1]$, $Q[i]$) şi ($Q[i]$, $Q[i-1]$) şi contopirea operațiilor corespunzatoare perechilor, din Tabela 10.1, cu observația că prima pereche are o pondere dublă față de prima (2^{i+1} față de 2^i).

Pentru a arăta în mod concret cum funcționează noul algoritm (Booth radix-4) am luat un exemplu la nivel de bit al desfăşurării iterațiilor aplicate regiştrilor ce conțin operanzii.

Count	A	Q	Q[–1]	M
00	0 0 0 0 0 0 0 0 0 0	1 1 0 1 0 0 **1 1**	**0**	0 1 0 0 0 1 0 1
+	1 1 0 1 1 1 0 1 1		└	⟶ – M
	1 1 0 1 1 1 0 1 1			
	1 1 1 1 0 1 1 1 0	1 1 1 1 0 1 **0 0**	**1**	
01 +	0 0 1 0 0 0 1 0 1		└	⟶ + M
	0 0 0 1 1 0 0 1 1			
	0 0 0 0 0 1 1 0 0	1 1 1 1 1 1 **0 1**	**0**	
10 +	0 0 1 0 0 0 1 0 1		└	⟶ + M
	0 0 1 0 1 0 0 0 1			
	0 0 0 0 1 0 1 0 0	0 1 1 1 1 1 **1 1**	**0**	⟶ – M
11 +	1 1 0 1 1 1 0 1 1			
	1 1 1 0 0 1 1 1 1			
	1 1 1 1 1 0 0 1 1	1 1 0 1 1 1 1 1	**1**	

-3105_{10}

Figura 10.1 Un exemplu de acţiune al algoritmului Booth radix-4 pe 2 operanzi (-45 stocat în registrul Q şi + 69 stocat în registrul M):

În Figura 10.1 sunt 4 paşi ai algoritmului, pentru operanzii pe 8 biţi. Primul pas înseamnă scăderea lui *M* din produsul parţial (iniţial '0' – stocat în acumulator) urmată de un dublu shift dreapta prin regiştrii *A* şi *Q*. Faptul că se scade *M* se datorează faptului că în Tabela 10.2 avem –*M* în dreptul intrării *Q*[*i*+1, *i*, *i*+1]. }n implementarea cu regiştrii fixaţi – corespunzătoare platformei hardware Hayes, *i*=0.

2 Structura Hardware Folosită

Structura hardware pe care rulează algoritmul este bazată pe platforma hardware Hayes, care nu suferă prea multe modificări. În primul rând registrul counter va fi pe doar 2 biţi, întrucât cei patru paşi ai algoritmului sunt etichetaţi cu numere de la 0 la 3. Faţă de implementarea aferentă algoritmului Booth radix-2, aici unitatea de control trebuie să mai aibă un input, şi anume semnalul *Q*[1]. În plus, regiştrii *A* şi *Q* trebuie să fie proiectaţi astfel încât, pe un singur impuls de clock să realizeze un shift dublu (aritmetic) la dreapta.

Semnalele de control vor acționa și asupra unui multiplexor care să selecteze operandul care se adună sau scade din produsul parțial (*M* sau 2*M*), după cum dictează unitatea de control.

Figura 10.2 prezintă structura hardware folosită pentru implementarea algoritmului. În timp ce Figura 10.3 conține ordinograma corespunzătoare algoritmului care rulează pe structura din Figura 10.2.

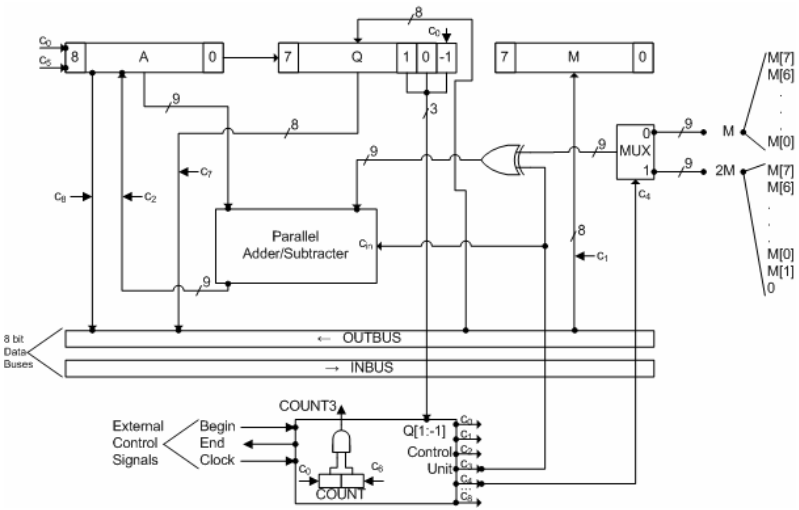

Figura 10.2 Structura hardware folosită la implementarea algoritmului Booth Radix-4.

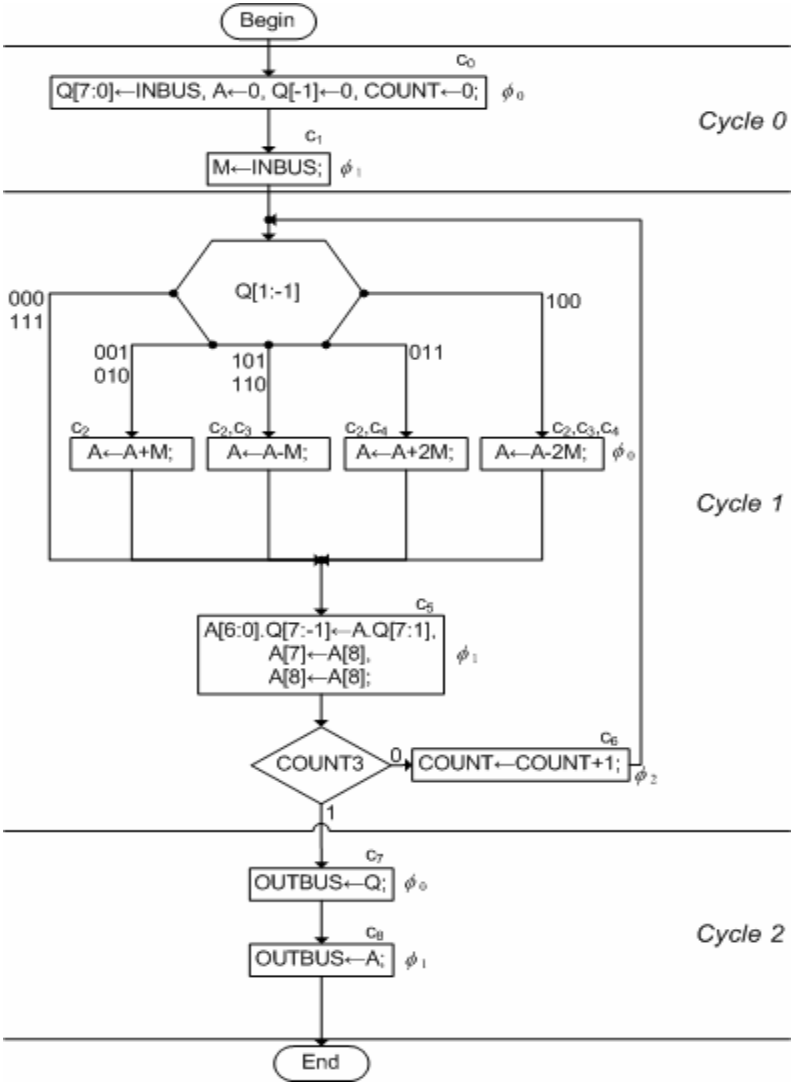

Figura 10.3 Ordinograma folosită la implementarea algoritmului Booth Radix-4.

3 Aplicații

Problema 10.1 Să se producă o soluție în radix-4 a versiunii algoritmului Booth modificat prezentate în Lucrarea 8.

Extinderea efectului ecuației $A'(n-1) = F' \oplus M(n-1)$ în vederea crearii unei implementări radix-4 a algoritmului Booth modificat, în versiunea din Lucrarea 8, revendică investigarea într-un singur pas a biților x_{i+2}, x_{i+1}, x_i ai înmulțitorului şi a flagului F, pentru a obține flagurile F', F'', şi x^o. x^o indică natura operației dictate de pasul curent al algoritmului, în conformitate cu Tabela 10.3.

x_{i+2}	x_{i+1}	x_i	F	x_i''	F'	x_{i+1}''	F''	x^o	Operație revendicată
0	0	0	0	0	0	0	0	0	Nici una
0	0	0	1	1	0	0	0	1	+M
0	0	1	0	1	0	0	0	1	+ M
0	0	1	1	0	1	1	0	2	+ 2·M
0	1	0	0	0	0	1	0	2	+ 2·M
0	1	0	1	$\overline{1}$	1	0	1	$\overline{1}$	-M
0	1	1	0	$\overline{1}$	1	0	1	$\overline{1}$	- M
0	1	1	1	0	1	0	1	0	Nici una
1	0	0	0	0	0	0	0	0	Nici una
1	0	0	1	1	0	0	0	1	+ M
1	0	1	0	1	0	0	0	1	+ M
1	0	1	1	0	1	$\overline{1}$	1	$\overline{2}$	- 2·M
1	1	0	0	0	0	$\overline{1}$	1	$\overline{2}$	-2·M
1	1	0	1	$\overline{1}$	1	0	1	$\overline{1}$	- M
1	1	1	0	$\overline{1}$	1	0	1	$\overline{1}$	- M
1	1	1	1	0	1	0	1	0	Nici una

Tabela 10.3 Operațiile revendicate şi flagurile pentru algoritmul Booth modificat radix-4, în versiunea de la Lucrarea 8 (M este registrul care stochează deînmulțitul).

Astfel, pentru algoritmul Booth modificat radix-4, atunci cînd se aplică tehnica "*running over zeros*", un pas constă dintr-o operație aritmetică

(adunare sau scădere) în conformitate cu Tabela 10.3 şi un rightshift dublu (i.e. pe două poziții binare). La efectuarea operației de dublu rightshift, cele mai semnificative două poziții ale produsului parțial (reprezentate pe n+1 biți, cu pozițiile 0 şi n fiind cel mai puțin semnificativ, respectiv cel mai semnificativ bit), $A'(n)$ şi $A'(n\text{-}1)$, vor avea valori în conformitate cu Ecuația 10.1.

$$\begin{aligned} A'(n) &= F''\oplus M(n-1) \\ A'(n-1) &= F'\oplus M(n-1) \end{aligned} \qquad (10.1)$$

Ştiind că F este inițial 0 vom avea valorile flagurilor:

$$\begin{aligned} F' &= F\cdot(x_{i+1}+x_i)+x_{i+1}\cdot x_i \\ F'' &= F'\cdot(x_{i+2}+x_{i+1})+x_{i+2}\cdot x_{i+1} \end{aligned} \qquad (10.2)$$

Prin urmare, valoarea biților post-shiftare $A'(n)$ şi $A'(n\text{-}1)$ este total independentă de natura produselor parțiale.

Problema 10.2 (propusă) Să se demonstreze faptul că algoritmul Booth modificat radix-2 este doar un caz particular al algoritmului Booth simplu radix-4.

Problema 10.3 (propusă) Pornind de la problematica prezentată în prezenta lucrare să se implementeze, în aceiaşi manieră, un înmulțitor Booth radix-16 pentru operanzi pe 16 biți**.** Se va folosi tot o platformă hardware de tip Hayes.