CIRCUITE ARITMETICE COMBINAȚIONALE

În această lucrare de laborator se descriu diferite tipuri de sumatoare și circuite combinaționale de înmulțire. Sunt prezentate următoarele tipuri de sumatoare: sumatorul elementar, sumatorul cu anticiparea transportului, sumatorul cu salvarea transportului și sumatorul zecimal. Dintre circuitele combinaționale de înmulțire, este prezentat circuitul de înmulțire matriceală și arborele Wallace. În lucrarea de laborator se descrie și modul de implementare pe o placă de dezvoltare a unui sumator cu anticiparea transportului și a unui circuit de înmulțire matriceală.

1. Sumatoare

1.1. Sumatorul elementar

Un sumator elementar adună trei intrări de câte un bit. Două din intrări sunt biții care trebuie adunați (notați cu x_i, y_i), iar cealaltă intrare este transportul de la bitul din poziția mai puțin semnificativă (notat cu T_i) Ieșirile sunt bitul sumă (S_i) și transportul către bitul din poziția mai semnificativă (T_{i+1}).

Ieșirile sumatorului elementar au următoarele expresii booleene:

$$S_i = x_i \oplus y_i \oplus T_i \tag{1}$$

$$T_{i+1} = x_i y_i + (x_i + y_i)\, T_i \tag{2}$$

unde:

$$x_i \oplus y_i = x_i \overline{y}_i + \overline{x}_i y_i \tag{3}$$

Sumatorul elementar este unul din blocurile de bază ale sumatoarelor mai complexe. Un caz particular de sumator elementar este *semisumatorul elementar*, care adună două intrări de câte un bit (nu există intrare de transport) și generează un bit sumă și un bit de transport.

După cum sumatoarele elementare se utilizează pentru realizarea sumatoarelor, semiscăzătoarele elementare și scăzătoarele elementare se utilizează pentru realizarea scăzătoarelor. Un *semiscăzător elementar* scade două intrări de câte un bit și generează un bit diferență și un bit de împrumut. Dacă bitul descăzut este mai mic decât bitul scăzător, bitul de împrumut va fi setat la 1; în caz contrar, bitul de împrumut va fi setat la 0. Un *scăzător elementar* are trei intrări și două ieșiri. Două din intrări sunt cei doi biți care trebuie scăzuți (descăzut și scăzător), iar cealaltă intrare este împrumutul de la bitul din poziția mai puțin semnificativă. Ieșirile sunt bitul diferență și împrumutul către bitul din poziția mai semnificativă.

1.2. Sumatorul cu anticiparea transportului

Sumatorul cu anticiparea transportului crește viteza operației de adunare prin reducerea timpului necesar pentru formarea semnalelor de transport. Intrarea de transport necesară pentru un etaj este generată în mod direct, utilizând semnale de la toate etajele precedente, în loc de a se aștepta propagarea lentă a transporturilor de la un etaj la altul.

Figura 1 prezintă schema bloc a unui sumator cu anticiparea transportului pentru adunarea a două numere de câte 4 biți. În această figură, blocurile de transport generează intrările de transport pentru sumatoarele elementare. Intrările fiecărui bloc de transport sunt doar numerele care trebuie adunate și intrarea de transport inițială (T_0).

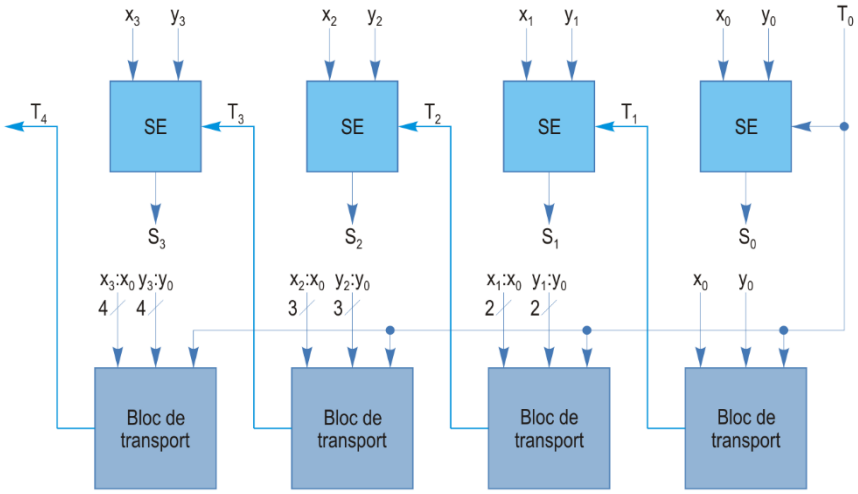

Figura 1. Schema bloc a unui sumator de 4 biți cu anticiparea transportului.

Expresiile booleene ale fiecărui bloc de transport pot fi definite pe baza expresiei transportului de ieșire al unui sumator elementar:

$$T_{i+1} = x_i y_i + (x_i + y_i)\, T_i \tag{4}$$

De exemplu, T_1 și T_2 pot fi generate astfel:

$$T_1 = x_0 y_0 + (x_0 + y_0)\, T_0 \tag{5}$$
$$T_2 = x_1 y_1 + (x_1 + y_1)\, T_1 = x_1 y_1 + (x_1 + y_1)\, [x_0 y_0 + (x_0 + y_0)\, T_0] \tag{6}$$

Pentru simplificarea expresiei fiecărui transport T_i, se utilizează de obicei două funcții, notate cu g și p. Funcția g se referă la *generarea* transportului, iar funcția p se referă la *propagarea* intrării de transport la ieșirea de transport. Aceste funcții sunt definite astfel:

$$g_i = x_i\, y_i \tag{7}$$
$$p_i = x_i + y_i \tag{8}$$

Denumirea de *generare* a transportului provine din faptul că etajul i generează un transport egal cu '1' (T_{i+1} = '1') independent de valoarea T_i dacă atât x_i cât și y_i este '1' (dacă x_i y_i = '1'). Etajul i propagă transportul T_i, deci T_{i+1} va fi '1' ca răspuns la T_i = '1', dacă x_i sau y_i este '1' (dacă $x_i + y_i$ = '1').

Astfel, expresia (1) poate fi scrisă în funcție de g_i și p_i ca:

$$T_{i+1} = g_i + p_i\, T_i \tag{9}$$

Utilizând aceste notații, semnalele de transport pentru sumatorul cu anticiparea transportului de 4 biți se pot exprima astfel:

$$T_1 = g_0 + p_0\, T_0 \tag{10}$$
$$T_2 = g_1 + p_1\, g_0 + p_1\, p_0\, T_0 \tag{11}$$
$$T_3 = g_2 + p_2\, g_1 + p_2\, p_1\, g_0 + p_2\, p_1\, p_0\, T_0 \tag{12}$$
$$T_4 = g_3 + p_3\, g_2 + p_3\, p_2\, g_1 + p_3\, p_2\, p_1\, g_0 + p_3\, p_2\, p_1\, p_0\, T_0 \tag{13}$$

Figura 2 prezintă schema bloc a sumatorului cu anticiparea transportului realizat în acest fel. Fiecare sumator de un bit produce semnalele de propagare transport pe bit p și generare transport pe bit g în locul ieșirilor de transport. Generatorul de transport anticipat convertește cele patru seturi de semnale p, g în intrările de transport necesare pentru cele patru sumatoare. Fiecare sumator de un bit produce, de asemenea, bitul corespunzător al sumei, în același mod ca și un sumator elementar.

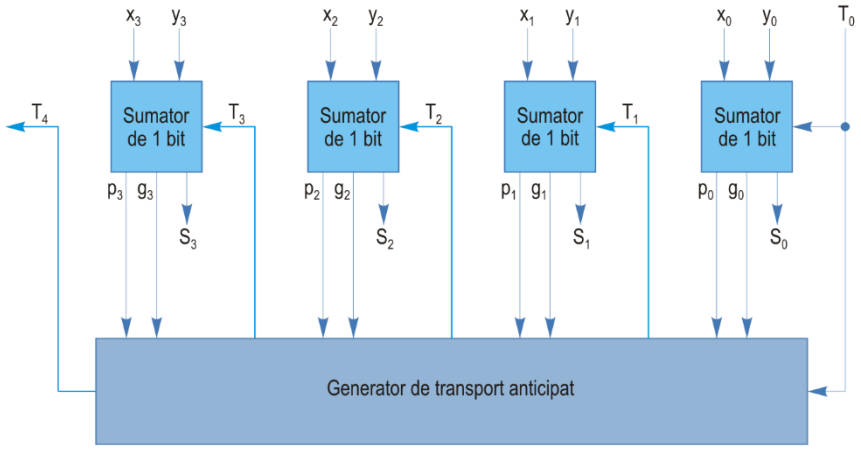

Figura 2. Schema bloc modificată a unui sumator de 4 biți cu anticiparea transportului care utilizează un generator de transport anticipat.

Pentru a reduce complexitatea circuitelor necesare generării semnalelor de transport, se limitează numărul de intrări ale porților și numărul de porți alimentate de acestea la o anumită valoare în funcție de tehnologie. Aceasta necesită adăugarea unor nivele logice suplimentare la circuitul cu anticiparea transportului. De exemplu, dacă în cazul precedent se limitează numărul de intrări la 4, va fi necesară o întârziere mai mare pentru generarea transportului T_4. Pentru a realiza limitarea amintită, se definesc două noi funcții, pentru *generarea transportului pe grup* $G_{i,k}$ și *propagarea transportului pe grup* $P_{i,k}$ pentru blocul corespunzător biților i până la k. De exemplu:

$$G_{0,3} = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 \quad (14)$$

$$P_{0,3} = p_3 p_2 p_1 p_0 \quad (15)$$

Cu aceasta se obține:

$$T_4 = G_{0,3} + P_{0,3} T_0 \quad (16)$$

Această ecuație are o formă similară cu ecuația (10) și ea poate fi obținută prin înlocuirea semnalelor de generare transport pe bit și propagare transport pe bit cu semnalele corespunzătoare de generare transport pe grup și propagare transport pe grup.

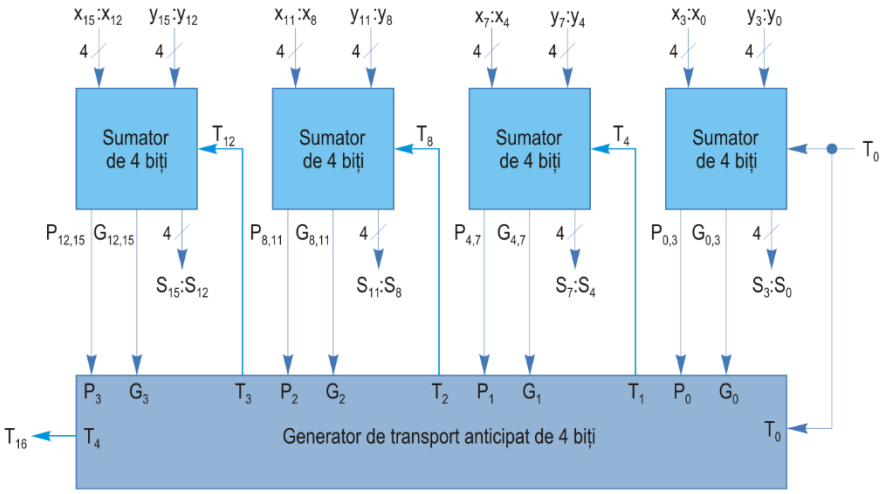

Figura 3. Schema bloc a unui sumator de 16 biți format din sumatoare de 4 biți conectate prin semnale de transport generate anticipat.

Sumatorul cu anticiparea transportului de 4 biți poate fi extins la un sumator de dimensiune mai mare. Sumatoarele de un bit se pot înlocui cu sumatoare pentru grupuri de 4

biți, după cum se arată în figura 3. Fiecare etaj al sumatorului produce semnalele de propagare transport pe grup și generare transport pe grup P și G, iar un generator de transport anticipat convertește cele patru seturi de semnale P, G în intrările de transport necesare pentru cele patru etaje.

În figura 3, T_8 are o ecuație care este similară cu ecuația (11):

$$T_8 = G_{4,7} + P_{4,7} G_{0,3} + P_{4,7} P_{0,3} T_0 \quad (17)$$

unde:

$$G_{4,7} = g_7 + p_7 g_6 + p_7 p_6 g_5 + p_7 p_6 p_5 g_4 \quad (18)$$
$$P_{4,7} = p_7 p_6 p_5 p_4 \quad (19)$$

Ecuațiile pentru semnalele T_{12} și T_{16} sunt similare cu ecuațiile (12), respectiv (13).

Sumatorul cu anticiparea transportului este mai rapid decât sumatorul cu transport succesiv, dar necesită un spațiu suplimentar în cadrul circuitului integrat datorită complexității mai ridicate.

1.3. Sumatorul cu salvarea transportului

Adunarea cu salvarea transportului este o altă tehnică ce poate fi utilizată pentru creșterea vitezei operației de adunare atunci când trebuie adunate mai mult de două numere. În timp ce tehnica de adunare cu anticiparea transportului reduce timpul necesar pentru generarea semnalelor de transport, adunarea cu salvarea transportului reduce timpul de propagare al semnalelor de transport. Un sumator cu salvarea transportului (SST) de n biți este o simplă colecție de n sumatoare elementare independente. Intrările sale sunt reprezentate de trei numere de câte n biți care trebuie adunate. Fiecare sumator elementar generează câte un bit sumă și un bit de transport, care formează un cuvânt sumă S de n biți, respectiv un cuvânt de transport T de n biți.

Spre deosebire de sumatorul cu propagarea succesivă a transportului, într-un sumator cu salvarea transportului fiecare sumator elementar funcționează independent unul de celălalt, iar semnalele de transport nu sunt propagate între sumatoarele elementare. Astfel, viteza de adunare este substanțial mai ridicată, deoarece toți biții sumă și toți biții de transport pot fi generați în paralel. Pentru obținerea rezultatului final, suma și transportul obținut trebuie adunate utilizând un sumator obișnuit, numit și *sumator cu propagarea transportului* (SPT). Acest termen indică un sumator care nu este un sumator cu salvarea transportului, și utilizează propagarea succesivă a transportului, anticiparea transportului, sau o altă metodă.

Exemplificăm funcționarea unui sumator cu salvarea transportului pentru adunarea a patru numere X, Y, Z și W. Sumatorul cu salvarea transportului generează mai întâi o sumă a primelor trei numere și un transport salvat. Presupunând că X = 5 (0101), Y = 3 (0011), Z = 4 (0100) și W = 1 (0001), suma și transportul salvat vor fi:

```
X                       0101
Y                       0011
Z                       0100
Suma                    0010
Transportul salvat      1010
```

În următoarea etapă, se adună suma, al patrulea număr (W) și transportul salvat, generând o nouă sumă și un nou transport salvat:

```
Suma                    0010
W                       0001
Transportul salvat      1010
Noua sumă               1001
Noul transport salvat   0100
```

În ultima etapă, se utilizează un sumator cu anticiparea transportului pentru a aduna noua sumă și noul transport salvat; rezultatul este 1101 (13).

```
Noua sumă               1001
Noul transport salvat   0100
Rezultat                1101
```

Pe baza acestor etape, se poate realiza un sumator cu operanzi multipli. Un asemenea sumator este utilizat de multe ori la circuitele de înmulțire pentru acumularea produselor parțiale. De exemplu, figura 4 prezintă o schemă bloc pentru adunarea a patru numere. Acest circuit cuprinde două sumatoare cu salvarea transportului, fiecare fiind format dintr-o serie de sumatoare elementare. În ultima etapă, se utilizează un sumator cu anticiparea transportului pentru obținerea sumei finale.

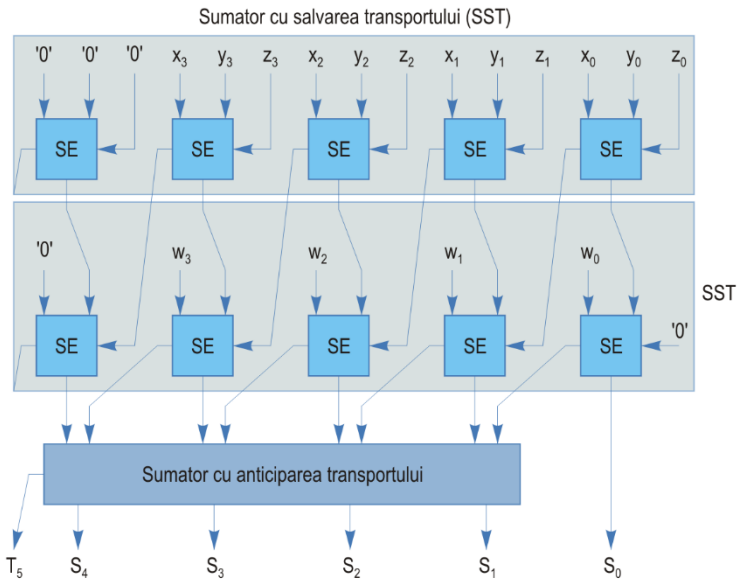

Figura 4. Schema bloc a unui sumator pentru adunarea a 4 numere de câte 4 biți.

Figura 5 ilustrează modul în care se poate utiliza un singur sumator cu salvarea transportului (un nivel) pentru adunarea unui set de numere. Mai întâi, se aplică trei numere la intrările *X*, *Y* și *Z*. Suma și transportul generate în urma adunării acestor numere sunt aplicate din nou la intrările *X* și *Y*, și apoi sunt adunate cu al patrulea număr, care se aplică la intrarea *Z*. Acest proces continuă până când se adună toate numerele.

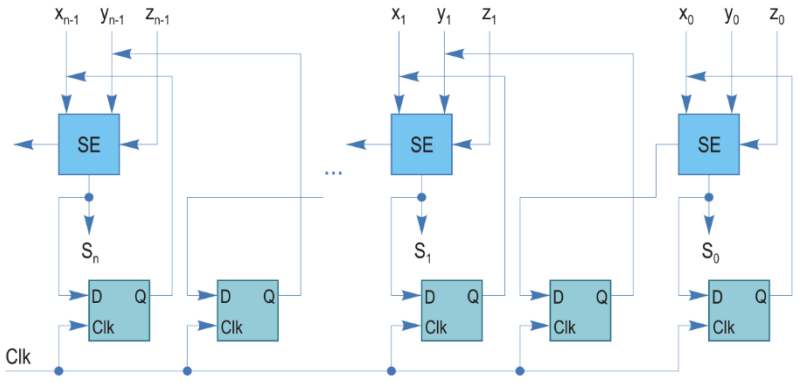

Figura 5. Schema bloc a unui sumator de *n* biți cu salvarea transportului pentru adunarea unui set de numere.

1.4. Sumatorul zecimal

Există cazuri în care numerele sunt reprezentate în zecimal, în codul BCD (*Binary Coded Decimal*). În asemenea cazuri, în locul conversiei numerelor în binar și utilizarea unui sumator binar pentru adunarea lor, este mai eficient să se utilizeze un sumator zecimal. Un sumator zecimal adună două cifre BCD în paralel și generează o sumă în cod BCD. Deoarece o cifră BCD are valori între 0 și 9, ori de câte ori suma depășește valoarea 9 sau se generează un transport către cifra următoare, rezultatul este corectat prin adunarea valorii 6 la rezultat. De exemplu, considerăm următoarea adunare:

```
245
474
6B9          suma intermediară
060
719          suma zecimală
```

Suma cifrelor 4 și 7 este mai mare decât 9; în acest caz, cifra sumă intermediară este corectată prin adunarea la aceasta a valorii 6.

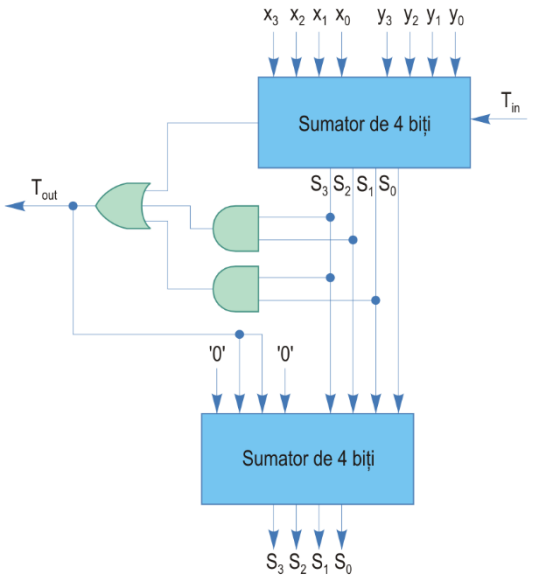

Figura 6. Schema bloc a unui sumator BCD.

Figura 6 prezintă un sumator zecimal bazat pe două sumatoare de 4 biți. Pe lângă cele două sumatoare de 4 biți, sumatorul mai cuprinde un număr de porți pentru corecția cifrelor sumă intermediare care sunt egale sau mai mari decât 10 (1010_2). Corecția este realizată și atunci când există un transport în urma adunării celor două cifre zecimale. Transportul de intrare T_{in} și transportul de ieșire T_{out} pot fi utilizate pentru realizarea unui sumator pentru numere de mai multe cifre.

2. Circuite combinaționale de înmulțire

2.1. Înmulțirea matriceală

Comparativ cu circuitele secvențiale de înmulțire, circuitele de înmulțire combinațională conțin o logică suplimentară care permite calcularea produsului într-un pas. Aceste circuite sunt formate din matrice de elemente combinaționale simple, fiecare din acestea implementând o operație de adunare și deplasare pentru un bit sau un număr redus de biți.

Presupunem că trebuie înmulțite două numere binare $X = x_{n-1}\ldots x_1x_0$ și $Y = y_{n-1}\ldots y_1y_0$. Pentru simplitate, presupunem că X și Y sunt numere întregi fără semn. Produsul P se poate exprima sub forma:

$$P = X * \left(\sum_{i=0}^{n-1} 2^i * y_i \right) \tag{20}$$

Această ecuație poate fi scrisă sub forma:

$$P = \sum_{i=0}^{n-1} 2^i * \left(\sum_{j=0}^{n-1} x_i * y_j * 2^j \right) \tag{21}$$

Fiecare din cei n^2 termeni produs de un bit x_i*y_j se poate calcula cu ajutorul unei porți ȘI cu două intrări, deoarece produsul aritmetic și cel logic coincid pentru un bit. Astfel, o matrice de $n \times n$ porți ȘI cu două intrări poate calcula toți termenii x_i*y_j simultan. Acești termeni sunt însumați cu ajutorul unei matrice de $n(n\text{-}1)$ sumatoare elementare, dintre care un număr de n sumatoare elementare pot fi înlocuite prin semisumatoare elementare. Circuitul rezultat este similar cu un sumator bidimensional cu transport succesiv. Deplasările implicate de factorii 2^i și 2^j în ecuația (21) sunt implementate prin deplasarea spațială a sumatoarelor pe direcția x și y.

Considerăm înmulțirea a două numere de câte 4 biți, $X = x_3x_2x_1x_0$ și $Y = y_3y_2y_1y_0$. Înmulțirea se efectuează după cum urmează:

				x_3	x_2	x_1	x_0 *
				y_3	y_2	y_1	y_0
0	0	0	0	x_3*y_0	x_2*y_0	x_1*y_0	x_0*y_0
0	0	0	x_3*y_1	x_2*y_1	x_1*y_1	x_0*y_1	0
0	0	x_3*y_2	x_2*y_2	x_1*y_2	x_0*y_2	0	0
0	x_3*y_3	x_2*y_3	x_1*y_3	x_0*y_3	0	0	0
P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0

Biții produsului final sunt următorii:

$$
\begin{aligned}
P_0 &= x_0*y_0 \\
P_1 &= x_1*y_0 + x_0*y_1 \\
P_2 &= x_2*y_0 + x_1*y_1 + x_0*y_2 \\
P_3 &= x_3*y_0 + x_2*y_1 + x_1*y_2 + x_0*y_3 \\
P_4 &= x_3*y_1 + x_2*y_2 + x_1*y_3 \\
P_5 &= x_3*y_2 + x_2*y_3 \\
P_6 &= x_3*y_3
\end{aligned}
\qquad (22)
$$

Figura 7 ilustrează o schemă posibilă pentru generarea și adunarea produselor parțiale. Dreptunghiurile notate cu x_iy_j reprezintă termeni produs generați cu porți ȘI. Semnalele de transport din fiecare rând de sumatoare elementare sunt conectate ca și la un sumator cu propagarea succesivă a transportului. Primul rând de sumatoare elementare adună primele două produse parțiale. Următoarele rânduri de sumatoare elementare adună câte un produs parțial la suma rezultată.

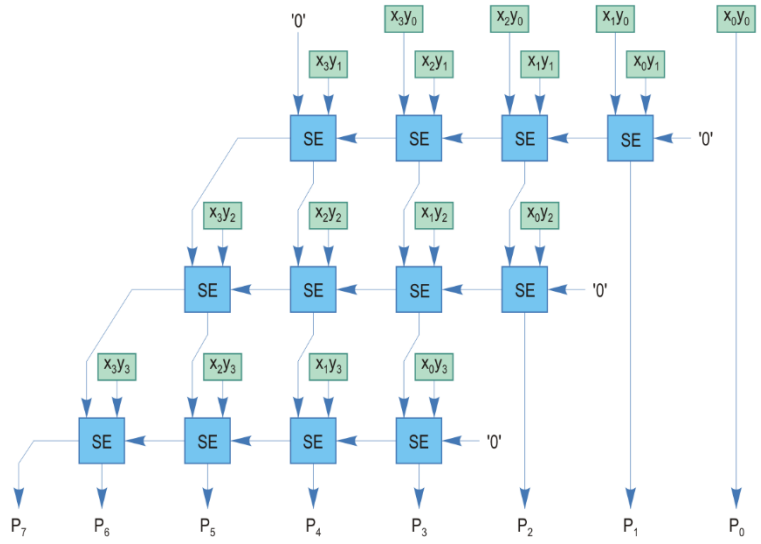

Figura 7. Matrice de sumatoare elementare pentru înmulțirea a două numere fără semn de câte 4 biți.

Funcția de adunare și funcția ȘI logic necesară circuitului de înmulțire matricială pot fi combinate într-o singură celulă *M*, după cum se ilustrează în figura 8.

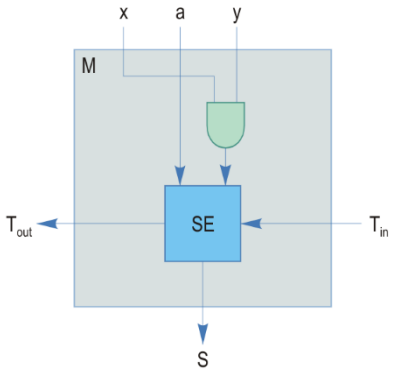

Figura 8. Celulă pentru înmulțirea matriceală a numerelor fără semn.

Această celulă implementează următoarea expresie aritmetică:

$$T_{out}S = a \textbf{ plus } xy \textbf{ plus } T_{in} \tag{23}$$

La intrarea *a* se conectează un bit al produsului parțial din linia precedentă de celule. Un circuit de înmulțire pentru $n \times n$ biți poate fi realizat prin utilizarea a n^2 celule de acest tip, deși unele celule de la periferia matricei vor avea intrările setate la 0. Timpul de execuție a operației de înmulțire este determinat de timpul de propagare al transportului pentru cazul cel mai defavorabil. Ignorând diferențele dintre celulele interne și cele periferice, acest timp este $(2n - 1)t_p$, unde t_p este timpul de propagare al celulei de bază.

Pentru creșterea vitezei operației de înmulțire, circuitele de înmulțire matricială pot fi realizate și prin utilizarea sumatoarelor cu salvarea transportului (SST). Viteza este crescută deoarece timpul necesar adunării produselor parțiale este mai redus, propagarea transportului între sumatoarele elementare din același rând fiind eliminată. Propagarea transportului este amânată până la ultimul etaj al circuitului.

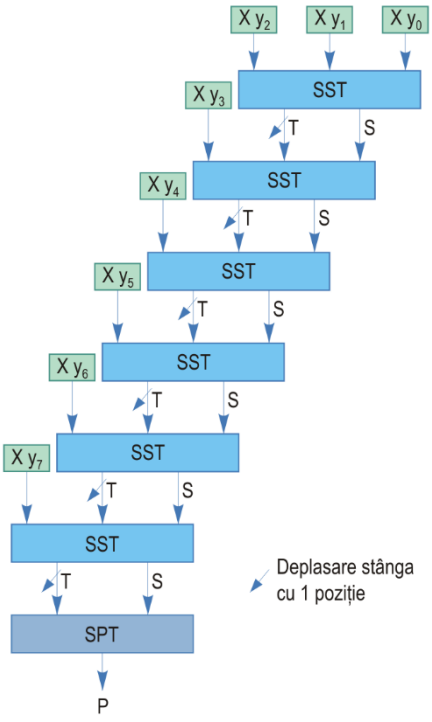

Figura 9. Circuit de înmulțire matriceală utilizând sumatoare cu salvarea transportului.

Figura 9 prezintă schema bloc a unui circuit de înmulțire matriceală pentru înmulțirea a două numere de câte 8 biți utilizând sumatoare cu salvarea transportului. Dreptunghiurile notate cu $X\,y_0$, $X\,y_1$, ..., $X\,y_7$ reprezintă produse parțiale de 8 biți generate cu porți ȘI. Această structură necesită un număr de șase sumatoare cu salvarea transportului și un sumator cu propagarea transportului (SPT). Ieșirea sumă S și ieșirea de transport T a unui sumator cu salvarea transportului sunt aplicate la intrările următorului sumator cu salvarea transportului. Conexiunile de transport sunt deplasate la stânga cu o poziție pentru a corespunde propagării normale a transportului. Această structură poate fi implementată în mod eficient într-un circuit VLSI datorită structurii sale regulate.

Circuitul de înmulțire combinațional cu structura din figura 9 este practic pentru valori moderate ale lui n. Pentru valori mari ale lui n, numărul necesar de sumatoare cu salvarea transportului poate fi excesiv. Tehnica de adunare cu salvarea transportului poate fi utilizată totuși dacă sumatorul este partiționat în k segmente de câte m biți fiecare. Sunt generate doar m produse parțiale, care sunt adunate utilizând sumatoare cu salvarea transportului. Procesul este repetat de k ori, sumele rezultate fiind acumulate.

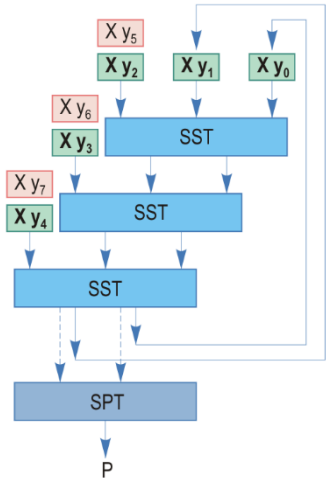

Figura 10. Circuit de înmulțire matriceală cu două treceri.

Figura 10 ilustrează o structură cu două treceri. La prima trecere se efectuează înmulțirea celor 5 biți mai puțin semnificativi (intrările utilizate la prima trecere sunt indicate prin caractere aldine). Rezultatul obținut după prima trecere este transferat apoi la intrarea circuitului pentru a fi combinat cu următoarele trei produse parțiale. Rezultatul final este calculat apoi cu un sumator cu propagarea transportului.

Spre deosebire de circuitele de înmulțire matriceală anterioare, care sunt complet combinaționale, circuitul de înmulțire matriceală cu două treceri necesită, în mod normal, un semnal de ceas. Totuși, dacă matricea din figura 10 are o dimensiune suficient de mare astfel încât nici un semnal nu se poate propaga de la intrare la ieșire în timpul necesar primului sumator pentru stabilizarea ieșirilor sale, este posibilă evitarea utilizării unui semnal de ceas.

2.2. Arborele Wallace

În modul cel mai simplu, înmulțirea a două numere de câte n biți se poate efectua prin adunarea a n produse parțiale, în modul ilustrat în secțiunile precedente. Circuitele de înmulțire prezentate până acum execută înmulțirea într-un timp $O(n)$. Acest timp poate fi redus la $O(\log n)$ prin utilizarea unui arbore. Arborele cel mai simplu ar combina perechi de produse parțiale $X\,y_0$ ... $X\,y_{n-1}$, reducând numărul produselor parțiale de la n la $n/2$. Aceste produse parțiale ar fi adunate apoi două câte două, obținându-se suma finală după un număr de $\log_2 n$ etape. Acest arbore binar simplu nu poate fi implementat însă utilizând sumatoare elementare, care generează două ieșiri din trei intrări, și nu o ieșire din două intrări.

C. S. Wallace a arătat că produsele parțiale pot fi adunate într-un mod rapid și economic utilizând nivele multiple de sumatoare cu salvarea transportului, organizate într-o

structură numită *arbore Wallace*. Presupunând că toate produsele parțiale sunt generate simultan, în primul nivel al arborelui numerele sunt grupate câte trei și se utilizează câte un sumator cu salvarea transportului pentru adunarea numerelor din fiecare grup. Astfel, problema adunării a n produse parțiale se reduce la problema adunării a $2n/3$ produse parțiale. În al doilea nivel, cele $2n/3$ produse parțiale rezultate sunt grupate din nou câte trei și adunate prin sumatoare cu salvarea transportului. Acest proces continuă până când rămân numai două numere de adunat. Pentru adunarea acestora se utilizează un sumator cu propagarea transportului (de multe ori, un sumator cu anticiparea transportului). Deoarece fiecare nivel reduce numărul termenilor care trebuie adunați cu un factor de 1,5, înmulțirea poate fi terminată într-un timp proporțional cu $\log_{1,5} n$.

Figura 11 prezintă schema bloc pentru înmulțirea a două numere de câte 8 biți utilizând un arbore Wallace.

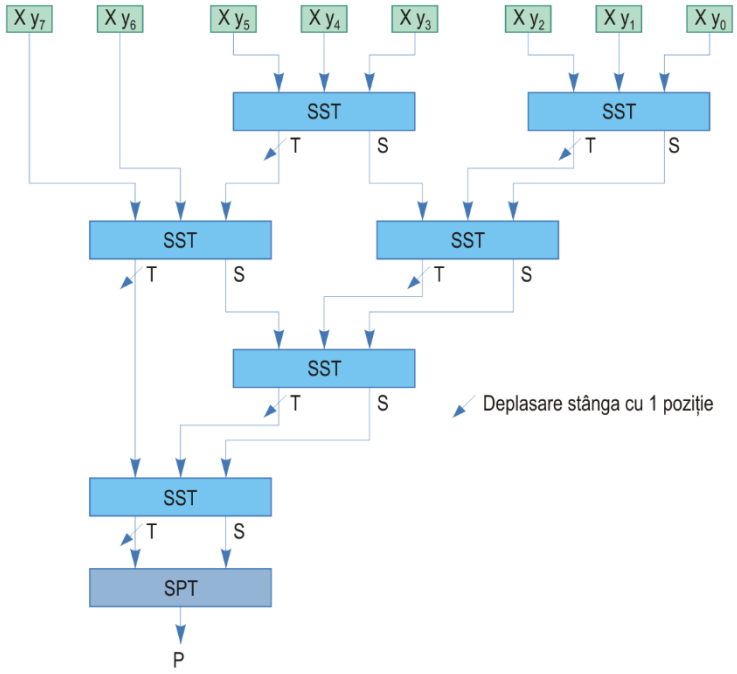

Figura 11. Schema bloc a unui circuit de înmulțire pentru numere de 8 biți utilizând un arbore Wallace.

De multe ori, metoda arborelui Wallace este combinată cu alte metode, ca de exemplu cu metoda Booth, pentru a se obține circuite rapide de înmulțire. Metoda Booth este utilizată pentru a genera produsele parțiale, iar apoi este utilizat un arbore Wallace pentru adunarea lor. De exemplu, presupunem înmulțirea a două numere de câte 8 biți, $X = x_7 \ldots x_1x_0$ și $Y = y_7 \ldots y_1y_0$. Figura 12 prezintă o structură posibilă pentru înmulțirea acestor numere.

În primul nivel al arborelui, se generează produsele parțiale prin utilizarea metodei Booth, în fiecare etapă fiind testați doi biți. În cazul unor numere de 8 biți, biții testați pentru generarea fiecărui produs parțial sunt: $y_0 0$, y_1y_0, y_2y_1, y_3y_2, y_4y_3, y_5y_4, y_6y_5, y_7y_6. Produsele parțiale sunt însumate apoi printr-un set de sumatoare cu salvarea transportului aranjate sub forma unui arbore Wallace.

Pentru reducerea circuitelor necesare în structura precedentă, în special pentru numere mari, de multe ori produsul final se obține prin mai multe treceri prin arborele Wallace. Figura 13 prezintă o asemenea structură, care necesită două treceri. În prima trecere se adună cele patru produse parțiale care rezultă din cei 4 biți mai puțin semnificativi ai înmulțitorului. În a doua trecere, suma parțială rezultată și transportul se aplică la intrarea unui sumator din primul nivel al arborelui, pentru a fi adunate cu următoarele patru produse parțiale.

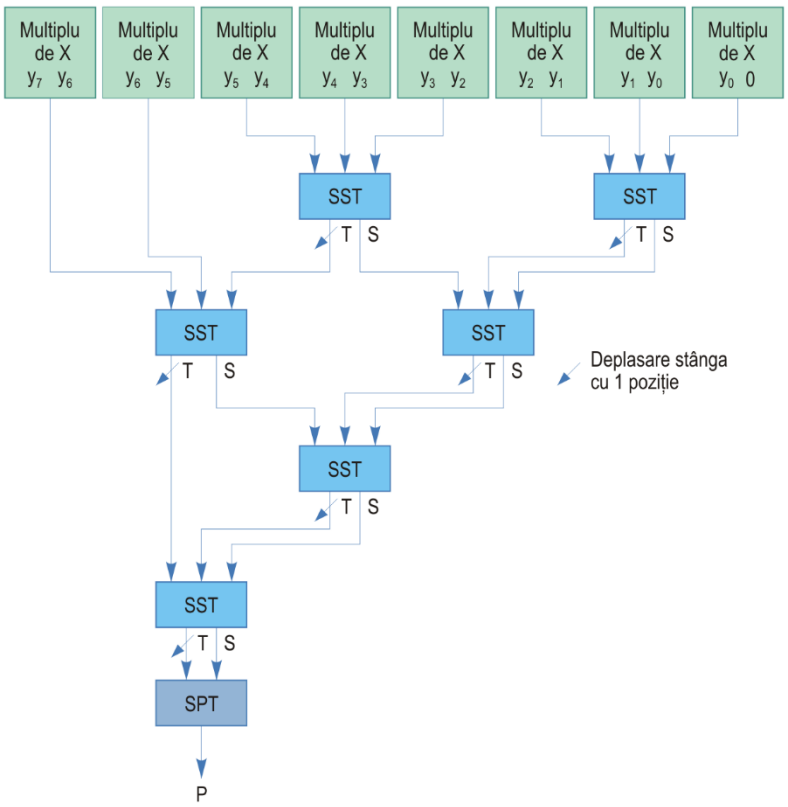

Figura 12. Circuit de înmulțire care combină metoda Booth cu un arbore Wallace.

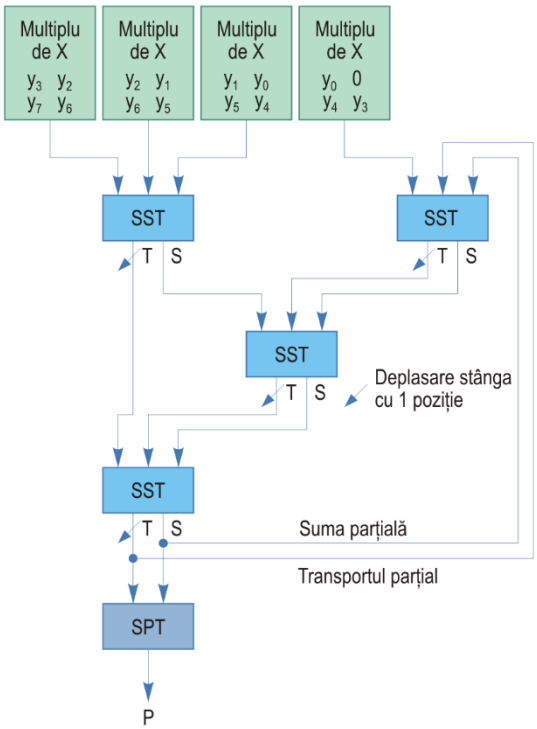

Figura 13. Circuit de înmulțire cu două treceri care combină metoda Booth cu un arbore Wallace.

3. Aplicații

3.1. Răspundeți la următoarele întrebări:

a. Care este principiul sumatorului cu anticiparea transportului?

b. Scrieți ecuațiile ieșirilor *P* și *G* ale unui sumator de 2 biți utilizat pentru sumatorul de 8 biți cu anticiparea transportului pe grupe de 2 biți din figura 14. Mai întâi, scrieți ecuațiile funcțiilor de generare a transportului pe bit pentru biții 0 și 1 (g_0, g_1) și a funcțiilor de propagare a transportului pe bit pentru biții 0 și 1 (p_0, p_1). Utilizând aceste funcții, scrieți apoi ecuația funcției de generare a transportului pe grupul de biți 0, 1 ($G_{0,1}$) și a funcției de propagare a transportului pe grupul de biți 0, 1 ($P_{0,1}$).

c. Scrieți ecuațiile semnalelor de transport T_2, T_4, T_6 și T_8 pentru sumatorul de 8 biți cu anticiparea transportului pe grupe de 2 biți din figura 14. Utilizați funcțiile de generare și de propagare a transportului pentru grupele de câte 2 biți.

d. Care este principiul arborelui Wallace?

3.2. Descrieți în limbajul VHDL sumatorul de 8 biți cu anticiparea transportului pe grupe de 2 biți din figura 14. Creați un modul VHDL pentru un sumator de 2 biți cu anticiparea transportului. Porturile de intrare ale acestui modul sunt *X*(1:0), *Y*(1:0) și *Tin*, iar porturile de ieșire sunt *S*(1:0), *P* și *G*. Creați apoi un modul VHDL pentru sumatorul de 8 biți, cu porturile de intrare *X*(7:0), *Y*(7:0), *Tin* și porturile de ieșire *S*(7:0), *Tout*. În acest modul, instanțiați direct entitatea sumatorului de 2 biți și scrieți ecuațiile semnalelor de transport T_2, T_4, T_6 și T_8 care sunt ieșiri ale generatorului de transport anticipat de 4 biți.

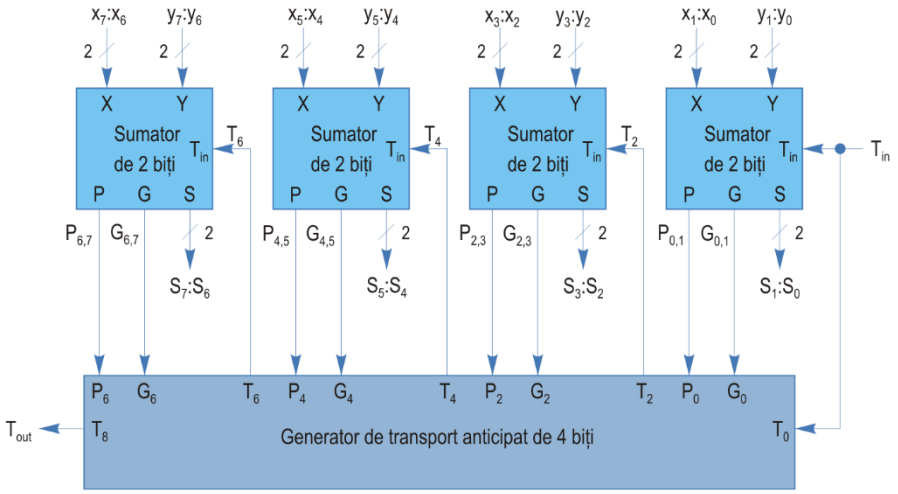

Figura 14. Schema bloc a unui sumator de 8 biți cu anticiparea transportului pe grupe de 2 biți.

Creați un fișier al bancului de test pentru sumatorul de 8 biți. Generați mai întâi vectori de intrare cu valori apropiate de 0 (de exemplu, cu valori între 0 și 2), apoi cu valori apropiate de cele maxime (de exemplu, cu valori între 253 și 255) și cu valori oarecare între valorile minime și maxime. De fiecare dată, verificați dacă ieșirile generate sunt corecte. Simulați funcționarea sumatorului cu simulatorul Vivado.

Observație

- Pentru generarea vectorilor de intrare ai sumatorului, se poate utiliza funcția de conversie `CONV_STD_LOGIC_VECTOR`, disponibilă în pachetul `STD_LOGIC_ARITH` din biblioteca `IEEE`. Această funcție convertește o valoare întreagă specificată de primul parametru al funcției într-o valoare de tip `STD_LOGIC_VECTOR`, pe un număr de biți specificat de al doilea parametru al funcției. De exemplu, funcția se poate utiliza pentru conversia indecșilor din buclele `for` .. `loop` în vectori de test.

3.3. Implementați pe placa de dezvoltare Nexys 4 DDR sumatorul de 8 biți cu anticiparea transportului creat pentru aplicația 3.2. Perifericele plăcii vor fi utilizate în felul următor:

- Primul operand *X* va fi introdus de la primele opt comutatoare SW15..SW8 (din partea stângă).
- Al doilea operand *Y* va fi introdus de la celelalte opt comutatoare SW7..SW0 (din partea dreaptă).
- Semnalul de resetare (necesar doar pentru multiplexorul afișajului cu șapte segmente) va fi generat prin butonul BTND.
- Starea comutatoarelor va fi afișată pe primele patru cifre (din partea stângă) ale afișajului cu șapte segmente.
- Pe a cincea cifră a afișajului cu șapte segmente se va afișa 0, iar pe a șasea cifră se va afișa transportul de ieșire T_{out} (0 sau 1).
- Suma va fi afișată pe ultimele două cifre (din partea dreaptă) ale afișajului cu șapte segmente.

Parcurgeți următoarele etape pentru implementarea sumatorului:

1. Adăugați la proiect fișierul sursă al multiplexorului pentru afișajul cu șapte segmente, disponibil pe pagina laboratorului în arhiva displ7seg.zip.
2. Creați un fișier sursă VHDL pentru modulul principal al sumatorului. Porturile de intrare ale acestui modul vor fi *Clk*, *Rst*, *X*(7:0) și *Y*(7:0) (porturile *Clk* și *Rst* vor fi necesare doar pentru multiplexorul afișajului cu șapte segmente). Porturile de ieșire ale modulului principal vor fi *An*(7:0) și *Seg*(7:0), necesare afișajului cu șapte segmente.
3. În fișierul sursă al modulului principal, instanțiați entitatea sumatorului de 8 biți. Aplicați valoarea logică '0' la intrarea de transport T_{in} a sumatorului și declarați semnalele care trebuie conectate la porturile sale de ieșire. Declarați un semnal *Data* care se va conecta la portul cu același nume al multiplexorului pentru afișajul cu șapte segmente. Asignați la acest semnal valorile care trebuie afișate, în modul specificat anterior. Instanțiați apoi entitatea multiplexorului cu șapte segmente `displ7seg`.
4. Adăugați la proiect fișierul de constrângeri Nexys4DDR_Master.xdc care a fost utilizat în proiectul ceasului de timp real și deschideți pentru editare acest fișier. Ștergeți caracterele # de la începutul liniilor corespunzătoare comutatoarelor SW[0] .. SW[15] și modificați numele porturilor pentru ca acestea să corespundă cu intrările *Y* și *X* ale sumatorului. Comentați liniile corespunzătoare butoanelor, cu excepția butonului BTND (la care a fost conectat portul *Down* al ceasului de timp real), și modificați numele portului din *Down* în *Rst*.
5. Realizați elaborarea proiectului și corectați erorile, dacă există.
6. Setați opțiunile pentru sinteza și implementarea proiectului. Pentru sinteză, selectați strategia de rulare *Flow_RuntimeOptimized*, iar pentru implementare selectați strategia de rulare *Flow_Quick*.
7. Realizați sinteza și implementarea proiectului, după care generați fișierul cu șirul de biți pentru configurarea circuitului FPGA.
8. Conectați o placă de dezvoltare Nexys4 DDR la un port USB al calculatorului și verificați funcționarea sumatorului.

3.4. Modificați descrierea sumatorului de 8 biți cu anticiparea transportului pe grupe de 2 biți creat pentru aplicația 3.2 astfel încât să utilizați instrucțiunea `for generate` pentru instanțierea sumatorului de 2 biți și pentru generarea semnalelor de transport.

Observație

- Recapitulați instrucțiunea `for generate` din secțiunea 6.1 a documentului *Proiectarea structurală în limbajul VHDL*, disponibil pe pagina laboratorului.

3.5. Descrieți în limbajul VHDL sumatorul de 16 biți cu anticiparea transportului pe grupe de 4 biți din figura 3. Simulați funcționarea sumatorului cu simulatorul Vivado.

3.6. Descrieți în limbajul VHDL un circuit de înmulțire matriceală pentru două numere de câte 8 biți fără semn. Creați mai întâi un modul VHDL pentru un sumator elementar. Creați apoi modulul VHDL pentru circuitul de înmulțire, cu porturile de intrare *X*(7:0), *Y*(7:0) și portul de ieșire *P*(15:0). Utilizați semnalele notate ca în figura 15.

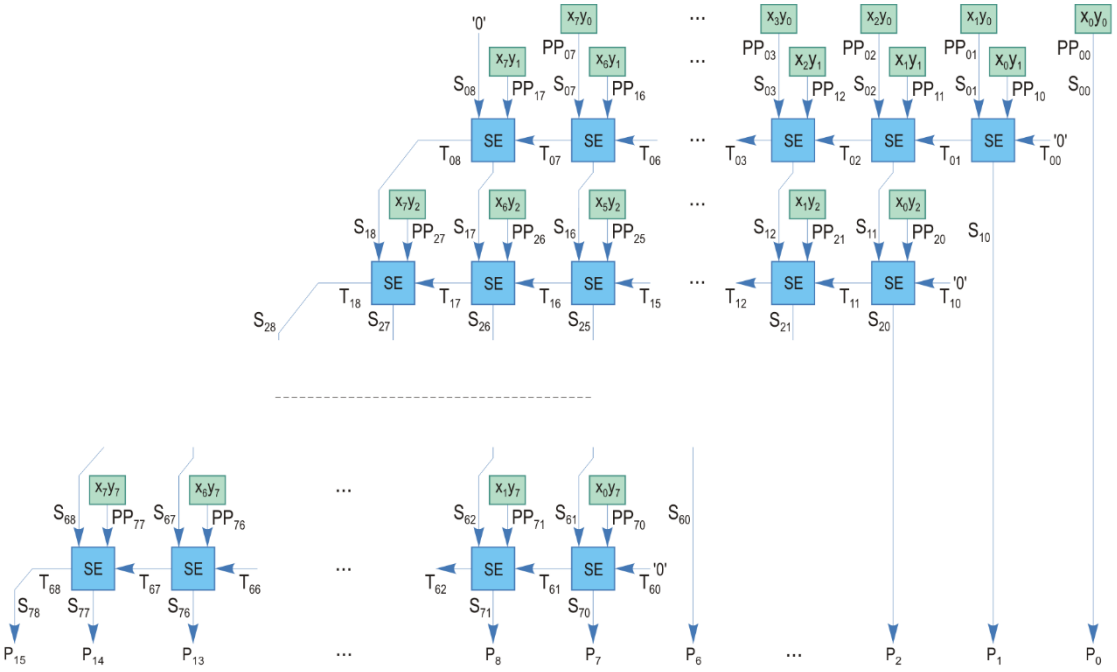

Figura 15. Denumirea semnalelor utilizate de circuitul de înmulțire matriceală pentru numere de câte 8 biți.

Definiți semnalele *PP*, *S* și *T* ca tablouri de vectori. Utilizați instrucțiunea `for generate` pentru generarea produselor parțiale, pentru inițializarea semnalelor de la marginile matricei și pentru generarea matricei de sumatoare elementare. Creați un fișier al bancului de test pentru circuitul de înmulțire. Generați vectori de intrare cu valori apropiate de 0, vectori cu valori apropiate de cele maxime și vectori cu valori oarecare între valorile minime și maxime, verificând de fiecare dată dacă ieșirile generate sunt corecte. Simulați funcționarea circuitului de înmulțire cu simulatorul Vivado.

3.7. Implementați pe placa de dezvoltare Nexys 4 DDR circuitul de înmulțire matriceală de 8 biți creat pentru aplicația 3.6. Perifericele plăcii vor fi utilizate în felul următor:

- Deînmulțitul *X* va fi introdus de la primele opt comutatoare SW15..SW8 (din partea stângă).
- Înmulțitorul *Y* va fi introdus de la celelalte opt comutatoare SW7..SW0 (din partea dreaptă).
- Semnalul de resetare (necesar doar pentru multiplexorul afișajului cu șapte segmente) va fi generat prin butonul BTND.
- Starea comutatoarelor va fi afișată pe primele patru cifre (din partea stângă) ale afișajului cu șapte segmente.

- Produsul va fi afișat pe ultimele patru cifre (din partea dreaptă) ale afișajului cu șapte segmente.

Parcurgeți următoarele etape pentru implementarea circuitului de înmulțire:

1. Adăugați la proiect fișierul sursă al multiplexorului pentru afișajul cu șapte segmente, disponibil pe pagina laboratorului în arhiva displ7seg.zip.
2. Creați un fișier sursă VHDL pentru modulul principal al circuitului de înmulțire. Porturile de intrare ale acestui modul vor fi *Clk*, *Rst*, *X*(7:0) și *Y*(7:0) (porturile *Clk* și *Rst* vor fi necesare doar pentru multiplexorul afișajului cu șapte segmente). Porturile de ieșire ale modulului principal vor fi *An*(7:0) și *Seg*(7:0), necesare afișajului cu șapte segmente.
3. În fișierul sursă al modulului principal, instanțiați entitatea circuitului de înmulțire de 8 biți. Declarați un semnal *Data* care se va conecta la portul cu același nume al multiplexorului pentru afișajul cu șapte segmente. Asignați la acest semnal valorile care trebuie afișate, în modul specificat anterior. Instanțiați apoi entitatea multiplexorului cu șapte segmente `displ7seg`.
4. Adăugați la proiect fișierul de constrângeri Nexys4DDR_Master.xdc care a fost utilizat în proiectul sumatorului de 8 biți cu anticiparea transportului, creat pentru aplicația 3.3.
5. Realizați elaborarea proiectului și corectați erorile, dacă există.
6. Setați opțiunile pentru sinteza și implementarea proiectului. Pentru sinteză, selectați strategia de rulare *Flow_RuntimeOptimized*, iar pentru implementare selectați strategia de rulare *Flow_Quick*.
7. Realizați sinteza și implementarea proiectului, după care generați fișierul cu șirul de biți pentru configurarea circuitului FPGA.
8. Conectați o placă de dezvoltare Nexys4 DDR la un port USB al calculatorului și verificați funcționarea circuitului de înmulțire.

3.8. Modificați descrierea circuitului de înmulțire matriceală creat pentru aplicația 3.6 pentru a utiliza tehnica de adunare cu salvarea transportului la adunarea produselor parțiale. În ultimul etaj al circuitului, utilizați un sumator cu propagarea succesivă a transportului. Creați un fișier al bancului de test și simulați funcționarea circuitului de înmulțire cu simulatorul Vivado.

3.9. Descrieți în limbajul VHDL un circuit de înmulțire matriceală pentru două numere de câte 8 biți fără semn. Utilizați celula de înmulțire din figura 8, fără alte componente sau porți logice. Pentru însumarea produselor parțiale utilizați tehnica de adunare cu propagarea succesivă a transportului. Creați un fișier al bancului de test și simulați funcționarea circuitului de înmulțire cu simulatorul Vivado.

3.10. Descrieți în limbajul VHDL sumatorul BCD din figura 6. Utilizați acest sumator pentru a realiza un sumator zecimal pentru două numere de câte 4 cifre. Creați un fișier al bancului de test și simulați funcționarea sumatorului zecimal cu simulatorul Vivado.