Lucrarea nr.1
Prelucrarea paralelă la nivel de bit.
Sumatorul cu transport progresiv și anticipat

1. Obiective

Proiectarea unor sisteme digitale de calcul ce implementează paralelismul la nivel de bit.

2. Resurse

- hardware: calculator PC
- software: compilatorul/simulatorul Icarus Verilog

3. Desfășurarea lucrării

Operația de adunare este cea mai frecventă operație executată de către unitatea centrală de procesare. Sumatorul este circuitul fizic ce realizează această operație pentru un întreg cuvânt.

3.1 Sumatorul cu transport progresiv

Are avantajul simplității în realizare/implementare, însă propagarea transportului prin cele N sumatoare de 1 bit produce întârzieri variabile în funcție de valorile efective ale operanzilor (figura 1).

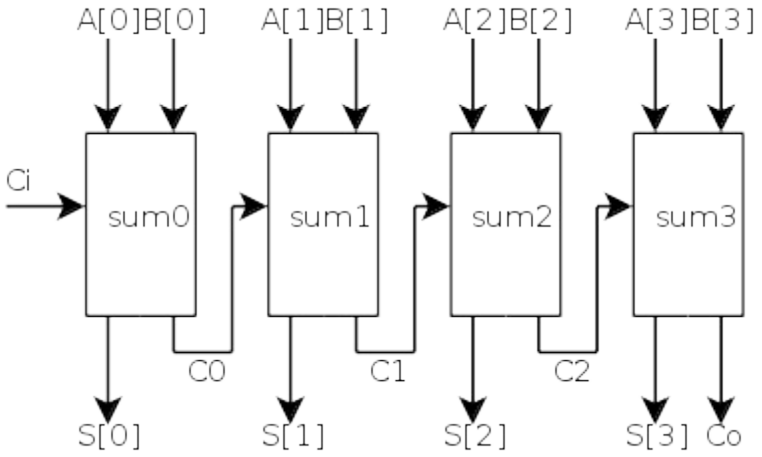

Fig.1. Sumatorul pe 4 biți cu transport progresiv

Implementarea în limbajul Verilog a sumatorului este prezentată în continuare (este utilizată implementarea la nivel RTL folosind operatorii logici disponibili). Pentru a modela cât mai aproape de realitate circuitul de tip sumator, se introduce o întârziere de 10 unități de timp în propagarea transportului și 5 unități de timp la calculul sumei pentru un sumator pe 1 bit.

Conținutul fișierului “sum4bit_ripple.v” este prezentat în continuare:

```
/*****************************************************************
    Modul Verilog pentru sumatorul cu transport progresiv
    pe 4 biti

*****************************************************************/
`timescale 10ns/1ns //scara de timp: 1unitate=10ns

/*****************************************************************
  Modul pentru sumatorul pe 1 bit
  intrare:
    - A=operand
    - B=operand
    - Ci=transport la intrare
  iesire:
    - S=suma
    - Co=transport la iesire
*****************************************************************/
module sum1bit(S,Co,A,B,Ci);
    input A,B,Ci;   //intrari
    output S,Co;    //iesiri

    assign #5       S=A^B^Ci;           //calcul suma
    assign #10      Co=Ci&(A^B)|(A&B);  //calcul transport
endmodule
/*****************************************************************/


/*****************************************************************
  Modul pentru sumatorul pe 4 bit
  intrare:
    - A=operand 4 biti
    - B=operand 4 biti
    - Ci=transport la intrare
  iesire:
    - S=suma 4 biti
    - Co=transport la iesire
*****************************************************************/
module sum4bit(S,Co,A,B,Ci);
    input[3:0] A,B; //intrari
    input Ci;       //intrari
    output[3:0]S;   //iesiri
    output Co;      //iesiri
    wire C0,C1,C2;  //conexiuni interne

    sum1bit sum0(S[0],C0,A[0],B[0],Ci); //sumatorul pt rangul 0
    sum1bit sum1(S[1],C1,A[1],B[1],C0); //sumatorul pt rangul 1
    sum1bit sum2(S[2],C2,A[2],B[2],C1); //sumatorul pt rangul 2
    sum1bit sum3(S[3],Co,A[3],B[3],C2); //sumatorul pt rangul 3
endmodule
/*****************************************************************/

/*****************************************************************
  Modul de simaulre pentru sumatorul pe 4 bit cu transport progresiv
*****************************************************************/
module simulation;
    reg[3:0] A,B;   //operanzii A,B
    reg Ci;         //transport la intrare
    wire[3:0]S;     //suma S
    wire Co;        //transport la iesire

    sum4bit sum(S,Co,A,B,Ci);    //instantierea sumatorului
    initial
```

```
      begin
          $dumpfile("sum4bit_ripple.vcd");    //fisier pentru stocarea formelor de unda
          $dumpvars();                        //directiva pentru exportul tuturor var.
          Ci=0;A=4'b0010;B=4'b0100;           //valori initiale de simulare: S=2+4+0
          #100 Ci=1;A=4'b1111;B=4'b0000;      //S=15+0+1
          #100 Ci=0;A=4'b1001;B=4'b0011;      //S=9+3+0
          #100 $finish;                       //terminare fortata a simularii
      end
    initial
      //directiva de afisare a variabilelor implicate si a timpului de simulare
      $monitor($time," %b %b <= %b + %b + %b",Co,S,A,B,Ci);
endmodule
/*************************************************************/
```

Compilarea fișierului se realizează cu comanda:

```
>iverilog -o num4bit_ripple num4bit_ripple.v
```

, iar simularea:

```
>vvp num4bit_ripple
```

În urma simulării, în modul text este afișat următorul rezultat:

```
  0 x xxxx <= 0010 + 0100 + 0
  5 x xxx0 <= 0010 + 0100 + 0
 10 0 xxx0 <= 0010 + 0100 + 0
 15 0 xx10 <= 0010 + 0100 + 0
 25 0 x110 <= 0010 + 0100 + 0
 35 0 0110 <= 0010 + 0100 + 0
100 0 0110 <= 1111 + 0000 + 1
105 0 1110 <= 1111 + 0000 + 1
115 0 1100 <= 1111 + 0000 + 1
125 0 1000 <= 1111 + 0000 + 1
135 0 0000 <= 1111 + 0000 + 1
140 1 0000 <= 1111 + 0000 + 1
200 1 0000 <= 1001 + 0011 + 0
205 1 0100 <= 1001 + 0011 + 0
215 1 1100 <= 1001 + 0011 + 0
220 0 1100 <= 1001 + 0011 + 0
```

Se observă ca rezultatul inițial: S=2+4+0=6 (0 0110) a fost obținut după 35 unități de timp, al doilea rezultat: S=15+0+1 (1 0000) după 40 unități de timp, iar S=9+3+0 (0 1100) după 20 de unități de timp.

Fișierul ce conține formele de undă asociate este prezentat în figura 2.

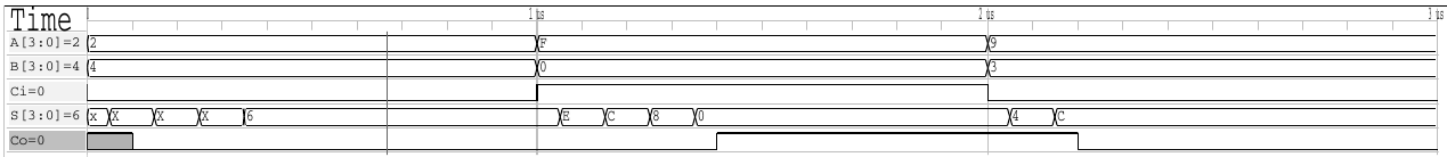

Fig.2. Forme de undă pentru sumatorul cu transport progresiv

3.1 Sumatorul cu transport anticipat (*carry look ahead adder*)

Pentru a înlătura întârzierea introdusă de propagarea transportului de la un sumator la următorul, se adaugă un bloc logic suplimentar ce permite calculul transportului în avans.

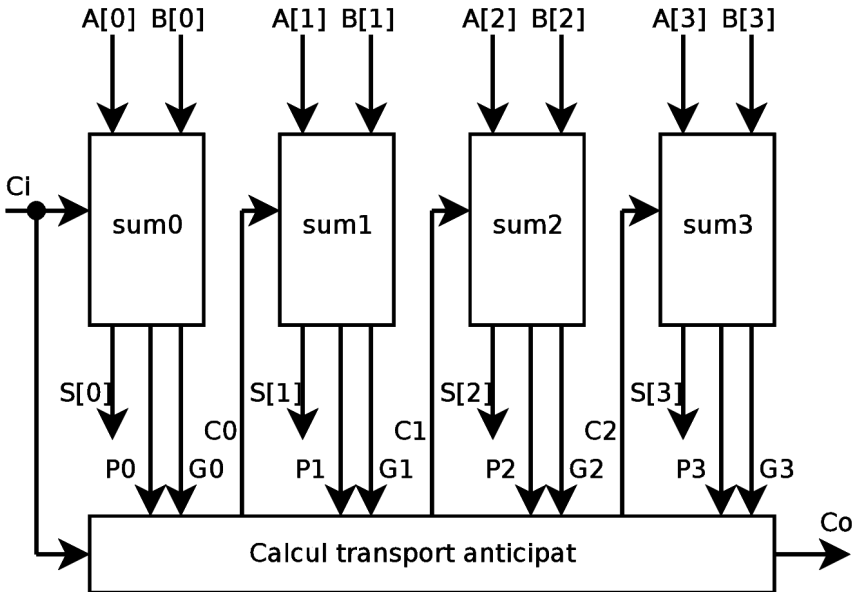

Fig.3. Schema bloc a sumatorului cu transport anticipat

Realizarea acestui tip de sumator implică utilizarea unor termeni suplimentari, cu scopul de a deduce valoarea bitului de transport în avand față de propagarea în cascadă.

Astfel, fiecare bloc de însumare calculează suplimentar față de suma S, doi termeni: G și P. Semnificația acestora este: G = generare, P = propagare.

G ia valoarea 1 dacă este constituită condiția de generare, adică cei doi operanzi A și B au valori 1, deci:

$$G=A\cdot B$$

P ia valoarea 1 dacă este îndeplinită condiția pentru propagarea transportului, adică în eventualitatea în care avem transport anterior, este îndeajuns ca cel puțin unul din operanzi să fie 1 ca să se transmită mai departe transportul, deci:

$$P=A+B$$

Pentru sumatorul pe 4 biți se pot scrie relațiile:

$C0=G0+P0\cdot Ci$
$C1=G1+P1\cdot C0$
$C2=G2+P2\cdot C1$
$Co=G3+P3\cdot C2$

Înlocuind relația pentru C0 în C1, C1 în C2 și C2 în C3, obținem:

$C0=G0+P0\cdot Ci$
$C1=G1+P1\cdot(G0+P0\cdot Ci)=G1+G0\cdot P1+P0\cdot P1\cdot Ci$
$C2=G2+P2\cdot(G1+P1\cdot(G0+P0\cdot Ci))=G1+G1\cdot P2+G0\cdot P1\cdot P2+P0\cdot P1\cdot P2\cdot Ci$
$Co=G3+P3\cdot(G2+P2\cdot(G1+P1\cdot(G0+P0\cdot Ci)))=G3+G2\cdot P3+G1\cdot P2\cdot P3+G0\cdot P1\cdot P2\cdot P3+P0\cdot P1\cdot P2\cdot P3\cdot Ci$

Diferența în timpul de calcul realizată practic este dependentă și de timpul de propagare efectiv prin porțile logice utilizate, însă în mod uzual, complexitatea unei porți de tip SAU/ȘI chiar cu 5 intrări nu crește timpul de calcul față de implementarea cu transport progresiv.

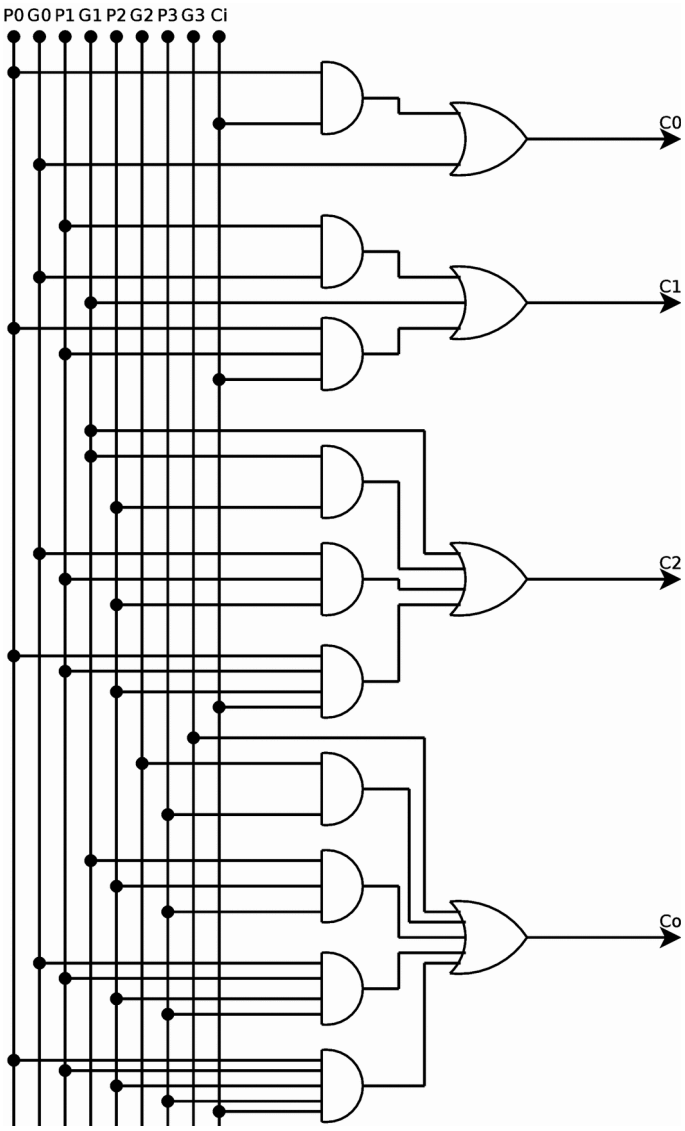

Fig.4. Implementarea cu porți logice SI, SAU a sumatorului
cu transport anticipat

Din figura 4 se observă că deși numărul de intrări variază, ajungând la maxim 5, sunt necesare doar două niveluri de porți logice.

Implementarea în limbajul Verilog cu utilizarea operatorilor logici este prezentată în listingul următor:

```
/*****************************************************************
    Modul pentru implementarea si simularea sumatorului cu
    transport anticipat pe 4 biti
*****************************************************************/
`timescale 10ns/1ns

/*****************************************************************
  Modul pentru sumatorul pe 1 bit
  intrare:
    - A=operand
    - B=operand
    - Ci=transport la intrare
  iesire:
    - S=suma
    - Co=transport la iesire
*****************************************************************/
module sum1bit(S,A,B,P,G,Ci);
    input A,B,Ci;
    output S,Co,P,G;

    assign #5       S=A^B^Ci;
    assign #2P=A|B;
    assign #2G=A&B;
endmodule
/*****************************************************************/

/*****************************************************************
  Modul pentru calculul transportului anticipat
  intrare:
    - P[4]=iesirile de propagare ale sumatoarelor
    - G[4]=iesirile de generare ale sumatoarelor
    - Ci=transport la intrare
  iesire:
    - C0=transport de la rangul 0
    - C1=transport de la rangul 1
    - C2=transport de la rangul 2
    - Co=transport la iesire (de la rangul 3)
*****************************************************************/
module lookahead(C0,C1,C2,Co,P,G,Ci);
    input[3:0] P,G;
    input Ci;
    output C0,C1,C2,Co;

    assign #6 C0=G[0]|(P[0]&Ci);
    assign #6 C1=G[1]|(G[0]&P[1])|(C0&P[0]&P[1]);
    assign #6 C2=G[2]|(G[1]&P[2])|(G[0]&P[1]&P[2])|(C0&P[0]&P[1]&P[2]);
    assign #6 Co=G[3]|(G[2]&P[3])|(G[1]&P[2]&P[3])|(G[0]&P[1]&P[2]&P[3])|
(C0&P[0]&P[1]&P[2]&P[3]);
endmodule
/*****************************************************************/

/*****************************************************************
  Modul pentru sumatorul pe 4 bit
  intrare:
    - A=operand 4 biti
    - B=operand 4 biti
    - Ci=transport la intrare
  iesire:
    - S=suma 4 biti
    - Co=transport la iesire
*****************************************************************/
```

```
module sum4bit(S,Co,A,B,Ci);
    input[3:0] A,B;
    input Ci;
    output[3:0]S;
    output Co;
    wire C0,C1,C2;
    wire[3:0]P,G;

    sum1bit sum0(S[0],A[0],B[0],P[0],G[0],Ci);
    sum1bit sum1(S[1],A[1],B[1],P[1],G[1],C0);
    sum1bit sum2(S[2],A[2],B[2],P[2],G[2],C1);
    sum1bit sum3(S[3],A[3],B[3],P[3],G[3],C2);

    lookahead lk(C0,C1,C2,Co,P,G,Ci);
endmodule

/*************************************************************
  Modul de simaulre pentru sumatorul pe 4 bit cu transport anticipat
*************************************************************/
module simulation;
    reg[3:0] A,B;
    reg Ci;
    wire[3:0]S;
    wire Co;

    sum4bit sum(S,Co,A,B,Ci);
    initial
      begin
          $dumpfile("sum4bit_fast.vcd");
          $dumpvars();
          Ci=0;A=4'b0111;B=4'b0111;
          #100 Ci=1;A=4'b1111;B=4'b0000;
          #100 Ci=0;A=4'b1001;B=4'b1011;
          #100 $finish;
      end
    initial
      $monitor($time," %b %b <= %b %b %b",Co,S,A,B,Ci);
endmodule
/*************************************************************/
```

Rezultatul simulării, afișat în mod grafic este prezentat mai jos, iar în figura 5 diagramele de semnal corespunzătoare scenariului de simulare.

```
  0 x xxxx <= 0111 0111 0
  5 x xxx0 <= 0111 0111 0
  8 0 xxx0 <= 0111 0111 0
 13 0 1110 <= 0111 0111 0
100 0 1110 <= 1111 0000 1
105 0 0000 <= 1111 0000 1
108 1 0000 <= 1111 0000 1
200 1 0000 <= 1001 1011 0
205 1 1100 <= 1001 1011 0
213 1 0100 <= 1001 1011 0
```

Se observă că timpul de calcul este redus semnificativ, astfel prima operație de adunare se încheie după 13 unități de timp (0-13), a doua după 8 unități de timp (100-108), iar ultima după 13 unități de timp (200-213).

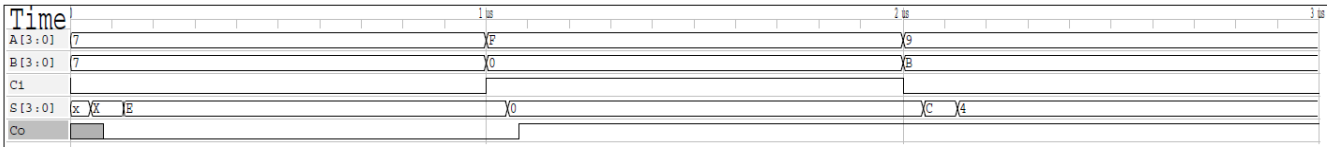

Fig.5. Forme de undă asociate sumatorului cu transport anticipat

4. Întrebări și exerciții

1. Implementați sumatorul cu transport anticipat în limbajul Verilog la nivel de porți logice, ținând cont de următorii parametri:

Poartă logică	Întârziere front pozitiv	Întârziere front negativ
AND	25	20
XOR	16	12
OR	20	20