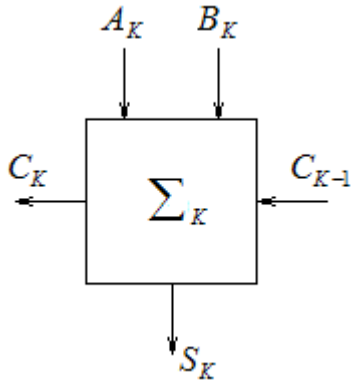
Modelarea/simularea sumatorului binar de rang (sumatorul complet – *full adder*)

În şedinţa de laborator se va realiza proiectul modelării în Verilog al unui sumator binar complet, adică al unui sumator de rang (un *full-adder*).

Proiectul va fi supus apoi simulării cu aplicaţia ModelSim.

A). Schema logică bloc a sumatorului binar de rang



B). Tabelul de adevăr

A_K	B_K	C_{K-1}	C_K	S_K
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

C). Funcţiile de ieşire S_K şi C_K deduse din tabelul de adevăr, sunt:

$$\begin{aligned} Sk &= \overline{A}_K \cdot \overline{B}_K \cdot C_{K-1} + \overline{A}_K \cdot B_K \cdot \overline{C}_{K-1} + A_K \cdot \overline{B}_K \cdot \overline{C}_{K-1} + A_K \cdot B_K \cdot C_{K-1} = \\ &= \left(\overline{A}_K \cdot \overline{B}_K + A_K \cdot B_K\right) \cdot C_{K-1} + \left(\overline{A}_K \cdot B_K + A_K \cdot \overline{B}_K\right) \cdot \overline{C}_{K-1} = \\ &= \overline{\left(A_K \oplus B_K\right)} \cdot C_{K-1} + \left(A_K \oplus B_K\right) \cdot \overline{C}_{K-1} = \\ &= A_K \oplus B_K \oplus C_{K-1} \end{aligned}$$

$$\begin{aligned} C_k &= \overline{A}_K \cdot B_K \cdot C_{K-1} + A_K \cdot \overline{B}_K \cdot C_{K-1} + A_K \cdot B_K \cdot \overline{C}_{K-1} + A_K \cdot B_K \cdot C_{K-1} = \\ &= A_K \cdot B_K + A_K \cdot C_{K-1} + B_K \cdot C_{K-1} \end{aligned}$$

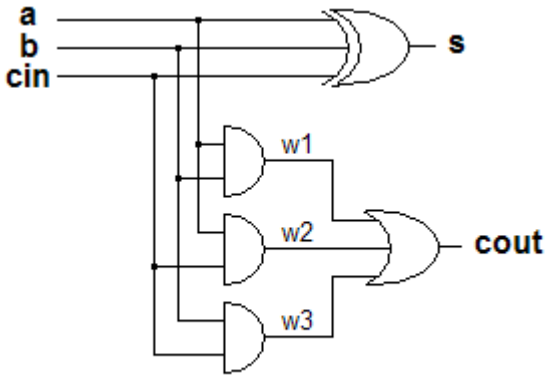
D). Schema logică structurală a sumatorului de rang

Schema de mai jos implementează cele două ecuaţii ale funcţiilor logice ale sumei şi bitului de transport deduse mai sus.

Pentru implementarea funcţiei bitului de transport (notat ***cout***), s-au utilizat trei porţi AND cu 2 intrări şi o poartă OR cu trei intrări iar pentru implemetarea funcţiei bitului sumă (notat ***s***) s-a utilizat o poartă XOR cu 3 intrări. În ambele situaţii s-a recurs, din motive de simplificare a schemei, la utilizarea de porţi OR şi XOR cu trei intrări, ştiut fiind că aceste porţi sunt fabricate doar cu două intrări; altfel ar fi fost necesare câte două porţi OR şi respectiv XOR.

În schemă, conexiunile interioare dintre porţile AND şi poarta OR s-au notat cu etichetele w_1, w_2 *şi* w_3. Aceste conexiuni vor fi reprezentate în modulul de modelare Verilog al sumatorului prin variabile de tipul **wire.**

Ecuaţiile celor două funcţii scrise cu operatori în limbaj Verilog, potrivit schemei de mai jos, sunt:
assign $s = a \,\hat{}\, b \,\hat{}\, cin$ şi *assign* $cout = (a \,\&\, b) \,|\, (a \,\&\, cin) \,|\, (b \,\&\, cin)$, sau altă variantă pentru cout
assign $w1 = (a \,\&\, b)$, *assign* $w2 = (a \,\&\, cin)$, *assign* $w3 = (b \,\&\, cin)$, *assign* $cout = w1 \,|\, w2 \,|\, w3$



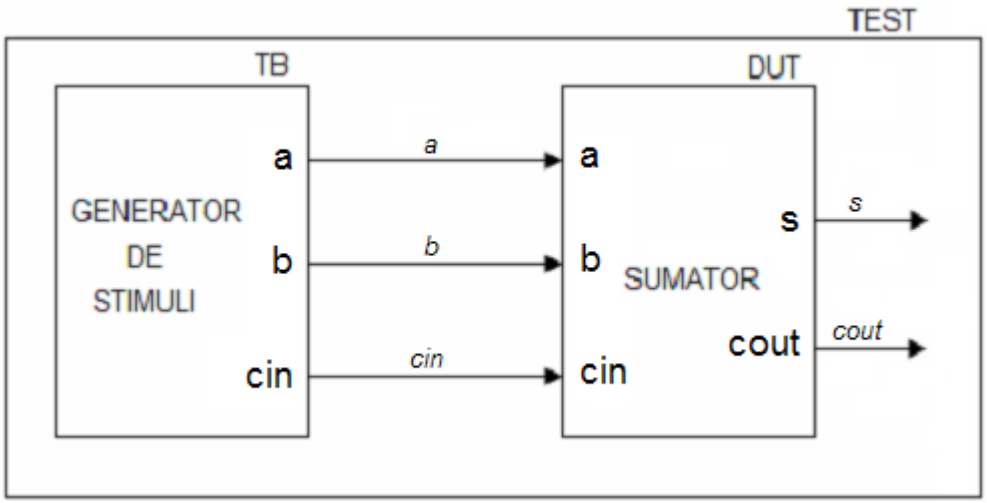
Schema logică a sumatorului

E). Schema bloc a mediului de simulare

Pentru simularea sumatorului de rang implementat prin schema logică de mai sus, se va utiliza schema bloc a modelului TEST de mai jos. Acest bloc este alcătuit, la rândul lui, din blocul TB (Test Bench) al generatorului de stimuli şi din blocul DUT (Device Under Test) al sumatorului.

Cele trei blocuri sunt modelate, fiecare, de modulele a căror cod Verilog este prezentat la punctul F: modulul ***sumator*** ce modelează sumatorul, modulul ***test_bench_sumator*** ce modelează generatorul de stimuli şi modulul ***test_sumator*** ce modelează blocul ansamblu de testare.

Schema blocului TEST evidenţiază şi firele de conexiune pentru transmiterea semnalelor generate de blocul TB spre intrările blocului DUT. Blocul TEST are deci rolul de modelare a conexiunilor dintre primele două blocuri şi totodată de evidenţiere a semnalelor funcţiilor de ieşire ***s*** şi ***cout*** ale sumatorului.



Schema bloc a modelului de simulare

F). Descrierea în cod Verilog a celor trei blocuri ale schemei de simulare

- **blocul DUT (Device Under Test)** este descris în modulul botezat **sumator** de mai jos. Acest modul se va salva într-un fişier botezat ***sumator.v*** (cu acelaşi nume, dar cu extensia "v"). Modul reprezintă descrierea comportamentală a sumatorului binar de rang, potrivit schemei logice date mai sus.

```
1   module sumator(a, b, cin, s, cout); // nume modul si listare porturi
2
3   input a, b, cin; // declarare porturi de intrare
4   output s, cout; // declarare a porturilor de iesire
5   wire w1, w2, w3; //fire legaturi interne (iesiri porti AND-intrari poarta OR)
6   wire w4;         //fir legatura nelegat, introdus pentru testare valoare
7
8   assign w1 = (a & b);
9   assign w2 = (a & cin);
10  assign w3 = (b & cin);
11
12  assign cout = (w1 | w2 | w3); //atribuire continua bitului de transport cout
13  assign s = (a ^ b ^ cin);     // atribuire continua bitului suma s
14
15  endmodule // incheiere modul
```

Notă: pentru obţinerea bitului sumă **s**-a admis, pentru simplificare, existenţa unei porţi XOR cu 3 intrări. In realitate, porţile XOR sunt prevăzute cu doar 2 intrări de date. Dacă ar fi fost utilizate 2 porţi XOR cu 2 intrări, expresia ar fi devenit $s = (a \hat{} b) \hat{} c_{in}$ şi deci ar fi necesitat evidenţierea a încă unui fir de conexiune internă, anume cel purtător al semnalului rezultat din operaţia (a^b).

- **blocul TB (Test Bench)** este descris în modulul denumit **test_bench_sumator**. Acest modul se va salva într-un fişier denumit ***test_bench_sumator.v*.** Modulul conţine decrierea funcţionării generatorului de stimuli, generator care trimite toată gama de combinaţii de valori binare pentru testare la intrările ***a, b*** şi ***cin*** ale modulului **sumator**. Modulul conţine un ceas (*clock*) care cadenţează seriile de stimuli expediate sumatorului şi conţine, de asemenea, un contor care contorizează numărul semnalelor de tact, oprind generarea de stimuli în momentul aplicării ultimei combinaţii de valori de 3 biti (111) sumatorului.

```
1  module test_bench_sumator (a, b, cin); //numele modulului si listare porturi
2
3   parameter per = 5; // declararea constantei per drept parametru,atribuita cu valoarea 5
4
5   output a, b, cin;  // declarare porturi de iesire din generatorul de stimuli
6
7   reg [2:0] counter; // declararea tipului si dimensiunii variabilei interne counter
8   reg clk;           // declararea tipului variabilei clk pentru semnalul de ceas
9
10  initial          //instructiune procedurala;blocul begin-end se executa o singura data
11    begin          // inceputul blocului de atribuiri ale lui initial
12     clk = 0;      // se initializeaza ceasul
13     counter = 0;  // se initializeaza contorul
14    end            // sfarsitul blocului de atribuiri ale lui initial
15
16  always @(posedge clk) // instruct. proced. always, sensibila la fronturile pozitive de ceas
17    if (counter == {3{1'b1}}) //daca counter devine 111,se trece la executia task-ului $stop
18      $stop;  // se opreste procedura de simulare si comanda trece interactiv user-ului
19    else                    // altfel, se trece la
20      counter = counter + 1; // incrementarea cu 1 a variabilei counter
21
22  assign a   = counter[2]; //asignere a cu valoarea bitului de rang 2 al lui counter
23  assign b   = counter[1]; //asignere b cu valoarea bitului de rang 1 al lui counter
24  assign cin = counter[0]; //asignere cin cu valoare bitului de rang 0 al lui counter
25
26  always
27   #per clk = !clk; // comutare semnal de ceas,la intervale de 5 unitati timp
28
29 endmodule   // incheiere corp modul
```

- **blocul TEST** este descris în modulul denumit t**est_sumator**, care se va salva într-un fişier denumit ***test_sumator.v*.** Acest modul realizează instanţierea într-un singur proiect a celor două module descrise mai sus şi descrierea conexiunilor necesare transmiterii semnalelor între cele două instanţe TB şi DUT. De asemenea, defineşte porturile ***.s*** şi ***.cout*** ale blocului TEST destinate vizualizării funcţiilor de ieşire (sumă şi transport).

```
1  module test_sumator();// numele modulului. Nu exista lista  de porturi.
2
3  parameter per = 5; // asignere a parametrului per cu 5 unitati de timp
4
5  //wire a, b, cin, s, cout; //nu e necesara declararea tipului de date.
6
7  test_bench_sumator #(per) TB   // citarea instantei TB a modulului test_bench_sumator
8      (.a(a),    // notatia se citeste: .a = portul a, (a) = firul a,ale modulului
9       .b(b),    // notatia se citeste: .b = portul b, (b) = firul b,ale modulului
10      .cin(cin) // notatia se citeste: .cin = portul cin, (cin) = firul cin,ale modulului
11     );
12
13 sumator DUT       // citarea instantei DUT a modulului sumator
14     (.a(a),        // notatia se citeste: .a = portul a, (a) = firul a,ale modulului
15      .b(b),        // notatia se citeste: .b = portul b, (b) = firul b,ale modulului
16      .cin(cin),    // notatia se citeste: .cin = portul cin, (cin) = firul cin,ale modulului
17      .s(s),        // notatia se citeste: .s = portul s, (s) = firul s,ale modulului
18      .cout(cout)   // notatia se citeste: .cout = portul cout, (cout) = firul cout,ale modulului
19     );
20 endmodule
```

Diagrama ***wave*** a simulării sumatorului de rang modelat şi lista ***list*** cu evoluţia în timp a valorilor semnalelor la comanda semnalului de ceas sunt prezentate în imaginile de mai jos:

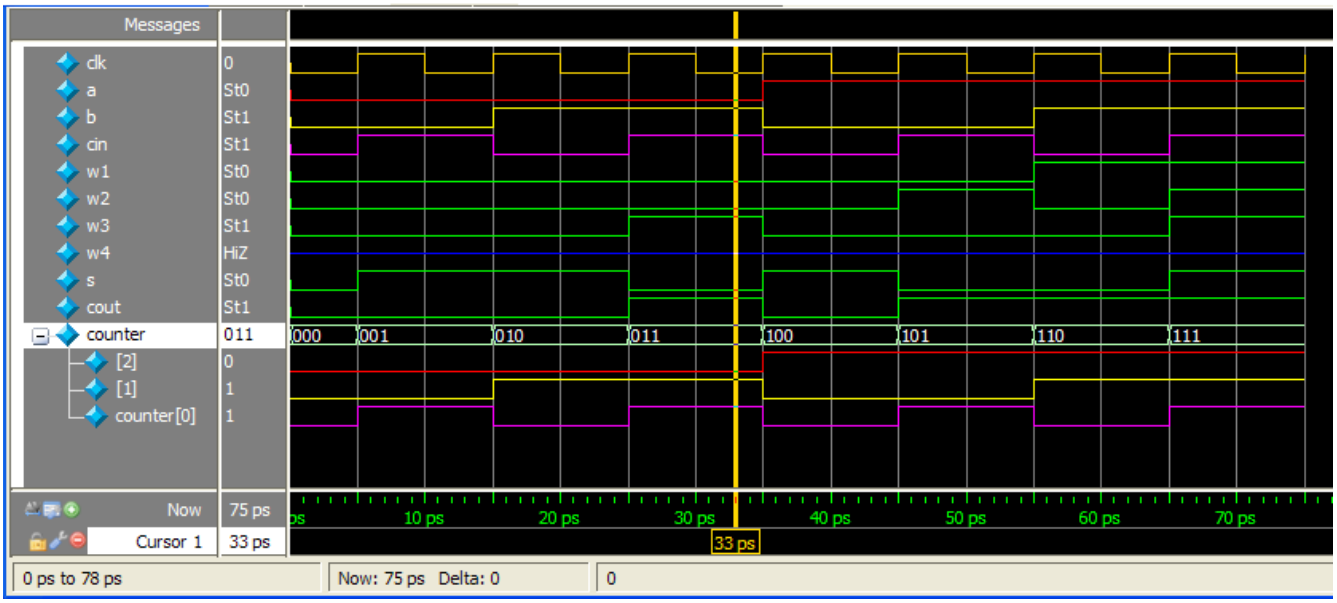


Diagrama de simulare ***wave*** a semnalelor de intrare şi de ieşire ale sumatorului de rang, modelat

ps	delta	a	b	cin	s	cout	w1	w2	w3	w4	clk	counter
0	+0	StX	StX	StX	StX	StX	StX	StX	StX	HiZ	0	000
0	+1	St0	St0	St0	StX	StX	StX	StX	StX	HiZ	0	000
0	+2	St0	St0	St0	St0	StX	St0	St0	St0	HiZ	0	000
0	+3	St0	St0	St0	St0	St0	St0	St0	St0	HiZ	0	000
5	+0	St0	St0	St0	St0	St0	St0	St0	St0	HiZ	1	001
5	+1	St0	St0	St1	St0	St0	St0	St0	St0	HiZ	1	001
5	+2	St0	St0	St1	St1	St0	St0	St0	St0	HiZ	1	001
10	+0	St0	St0	St1	St1	St0	St0	St0	St0	HiZ	0	001
15	+0	St0	St0	St1	St1	St0	St0	St0	St0	HiZ	1	010
15	+1	St0	St1	St0	St1	St0	St0	St0	St0	HiZ	1	010
20	+0	St0	St1	St0	St1	St0	St0	St0	St0	HiZ	0	010
25	+0	St0	St1	St0	St1	St0	St0	St0	St0	HiZ	1	011
25	+1	St0	St1	St1	St1	St0	St0	St0	St0	HiZ	1	011
25	+2	St0	St1	St1	St0	St0	St0	St0	St1	HiZ	1	011
25	+3	St0	St1	St1	St0	St1	St0	St0	St1	HiZ	1	011
30	+0	St0	St1	St1	St0	St1	St0	St0	St1	HiZ	0	011
35	+0	St0	St1	St1	St0	St1	St0	St0	St1	HiZ	1	100
35	+1	St1	St0	St0	St0	St1	St0	St0	St1	HiZ	1	100
35	+2	St1	St0	St0	St1	St1	St0	St0	St0	HiZ	1	100
35	+3	St1	St0	St0	St1	St0	St0	St0	St0	HiZ	1	100
40	+0	St1	St0	St0	St1	St0	St0	St0	St0	HiZ	0	100
45	+0	St1	St0	St0	St1	St0	St0	St0	St0	HiZ	1	101
45	+1	St1	St0	St1	St1	St0	St0	St0	St0	HiZ	1	101
45	+2	St1	St0	St1	St0	St0	St0	St1	St0	HiZ	1	101
45	+3	St1	St0	St1	St0	St1	St0	St1	St0	HiZ	1	101
50	+0	St1	St0	St1	St0	St1	St0	St1	St0	HiZ	0	101
55	+0	St1	St0	St1	St0	St1	St0	St1	St0	HiZ	1	110
55	+1	St1	St1	St0	St0	St1	St0	St1	St0	HiZ	1	110
55	+2	St1	St1	St0	St0	St1	St1	St0	St0	HiZ	1	110
60	+0	St1	St1	St0	St0	St1	St1	St0	St0	HiZ	0	110
65	+0	St1	St1	St0	St0	St1	St1	St0	St0	HiZ	1	111
65	+1	St1	St1	St1	St0	St1	St1	St0	St0	HiZ	1	111
65	+2	St1	St1	St1	St1	St1	St1	St1	St1	HiZ	1	111
70	+0	St1	St1	St1	St1	St1	St1	St1	St1	HiZ	0	111
75	+0	St1	St1	St1	St1	St1	St1	St1	St1	HiZ	1	111

Lista ***list*** de simulare a sumatorului de rang modelat, cu evoluţia în timp a valorilor semnalelor de intrare şi de ieşire

Întocmit,
Coordonator lucr. lab. CLP,
ing. Hurubeanu Ştefan Valeriu