

Restoring Division

PROIECT 1 OC:

MEMBRII ECHIPEI:

SPATACEAN OANA

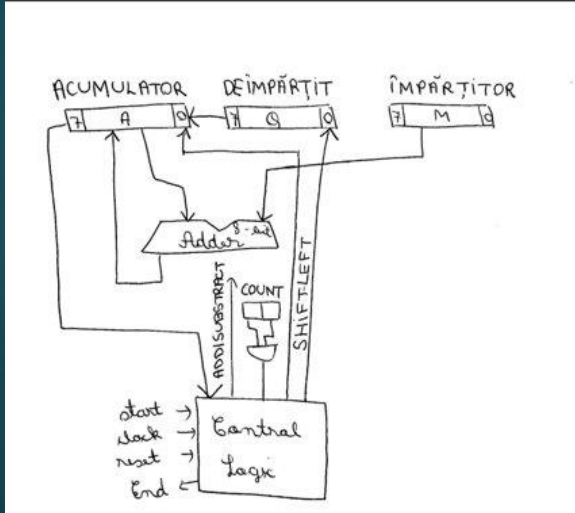
STOICHESCU IULIA

GRUPA: 2.C.06.1

Introducere

- ▶ Restoring Division este un algoritm de împărțire între două numere binare, deimpartit și impartitor. În acest algoritm, împărțirea este realizată prin substrații succesive, asemănătoare împărțirii cu rest.

Design



Algoritmul de împărțire prin restaurare funcționează în modul următor:

- ▶ Se așteaptă semnalul de start.
- ▶ Se încarcă deimpartitul în Q și se încarcă 0 în registrul A.
- ▶ Se execută operația de împărțire a lui A cu M până când toți biții sunt procesați.
- ▶ Dacă semnalul de reset este activat, toate valorile semnalelor de ieșire și de starea curentă sunt resetate.
- ▶ Se utilizează o serie de registre pentru a stoca valorile anterioare și a le utiliza la următoarea operație.

Implementare

▶ Algoritm pe scurt:

- ▶ $A[7:0], Q[7:0], M[7:0]$
- ▶ SUBTRACT: $A.Q[7:1] = A.Q;$
- ▶ $A = A - M$
- ▶ if($A[7] == 0$) than $Q[0] = 1$
- ▶ if($A[7] == 1$) than $A = A + M, Q[0] = 0;$
- ▶ if(count == 0) than goto OUTPUT
- ▶ else goto SUBTRACT

▶ Flowchart:

▶ **START**

▶ $A = 0$

▶ $M = \text{deimpartit}$

▶ $Q = \text{impartitor}$

▶ **SUBTRACT**

▶ Shifteaza stanga $AQ;$

▶ $A = A - M;$

▶ Verifica semnul lui A : daca e 0, atunci $Q[0] = 1;$

▶ daca e 1, atunci $Q[0] = 0$ si $A = A + M,$

▶ adica e restaurat; (restored)

▶ Decrementeaza count;

▶ Daca count e 0, atunci mergi la **STOP**;

▶ Daca nu, atunci revino la SUBTRACT si repeta-l.

Rezolvarea algoritmică exemplificată cu o problemă din laborator:

count	A	Q	M
0	0 0001 0110 0 0010 1101+ 1 0111 1001 = 1 1010 0110+ 0 1000 0111 = 0 0010 1101	1000 1011 0001 011_ 0001 0110	0 1000 0111
1	0 0 101 1010+ 1 0111 1001 = 1 1101 0011+ 0 1000 0111 = 0 0101 1010	0010 110_ 0010 1100	
2	0 1011 0100+ 1 011 1001 = 0 0010 1101	0101 100_ 0101 1001	
3	0 0101 1010+ 1 0111 1011 = 1 1100 1011+ 0 1000 0111 = 0 0101 1010	1011 001_ 1011 0010	
4	0 1011 0101+ 1 0111 1001 = 0 0010 1110	0110 010_ 0110 0101	
5	0 0101 1100+ 1 0111 1001 = 1 1101 0101+ 0 1000 0111 = 0 0101 1100	1100 101_ 1100 1010	
6	0 1011 1001+ 1 0111 0010 = 0 0011 0010	1001 010_ 1001 0101	
7	0 0110 0101+ 1 0111 1001 = 1 1101 1110+ 0 1000 0111 = 0 0110 0101	001 0101_ 0010 1010	

5771:135 Restul= 0 0110 0101= 101 Catul= 0010 1010=42

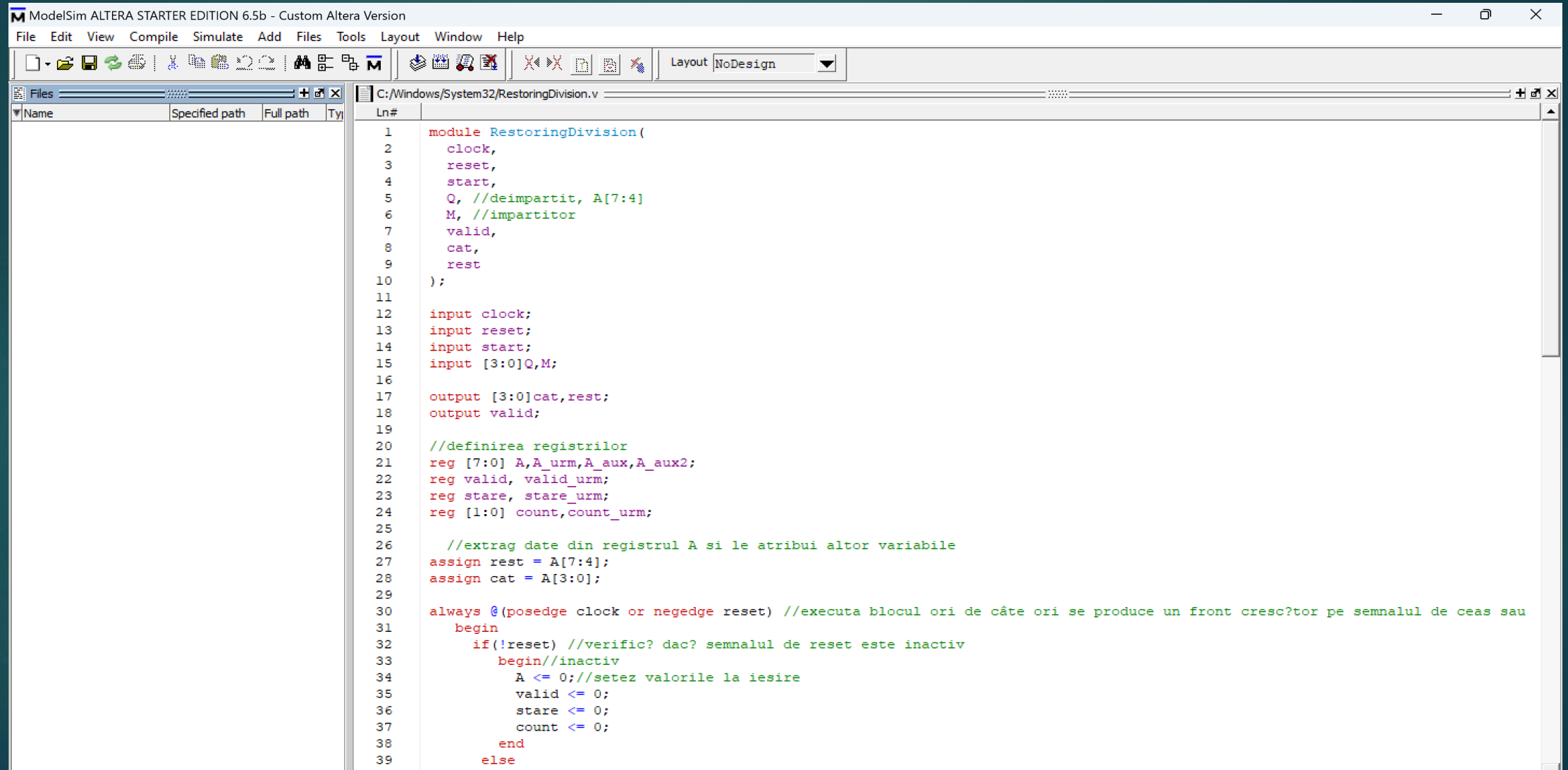
5771= 0001 0110 1000 1011

135= 0000 0000 1000 0111

M= 0 1000 0111

-M= 1 0111 1001

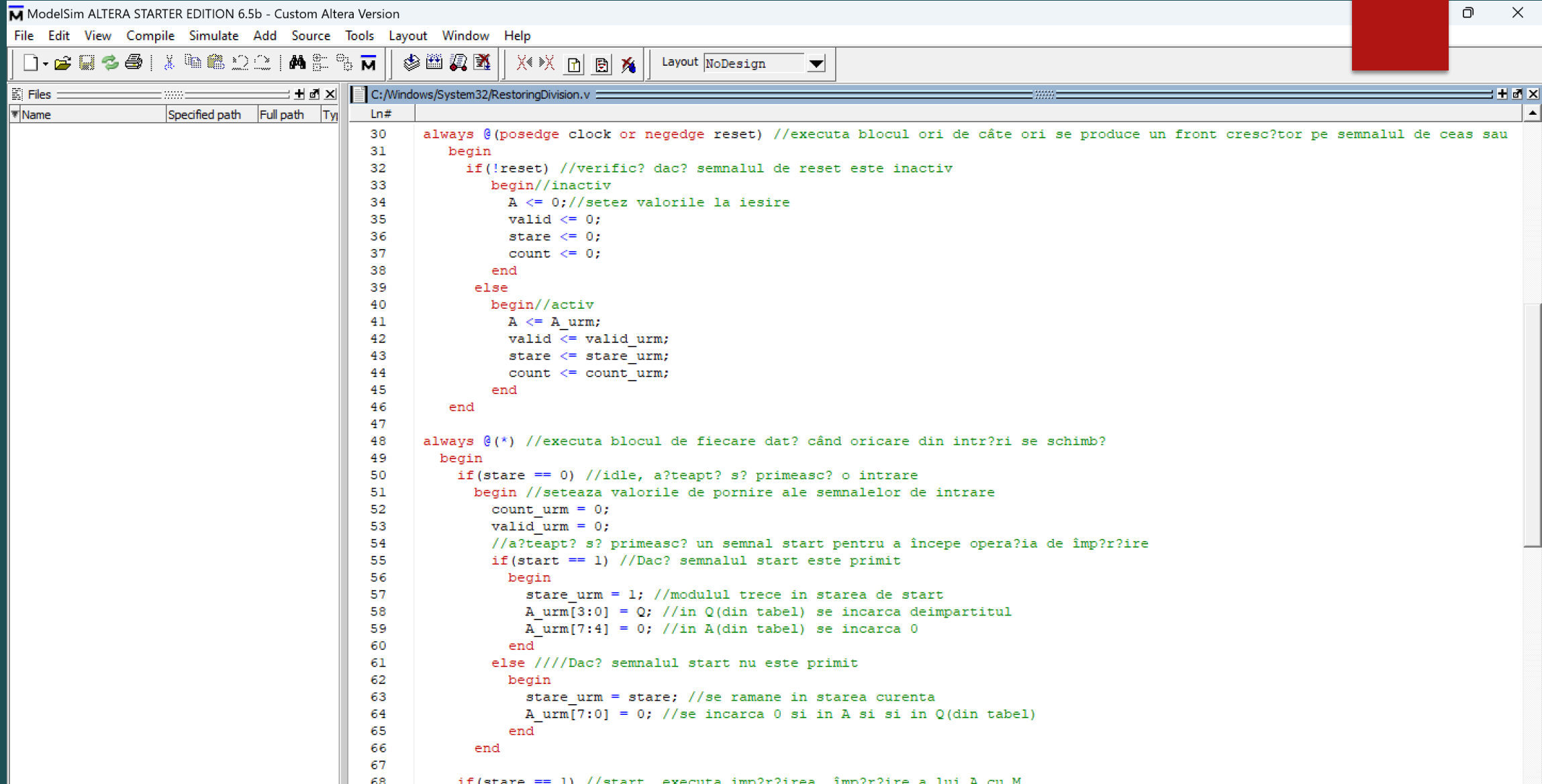
Cod Verilog:

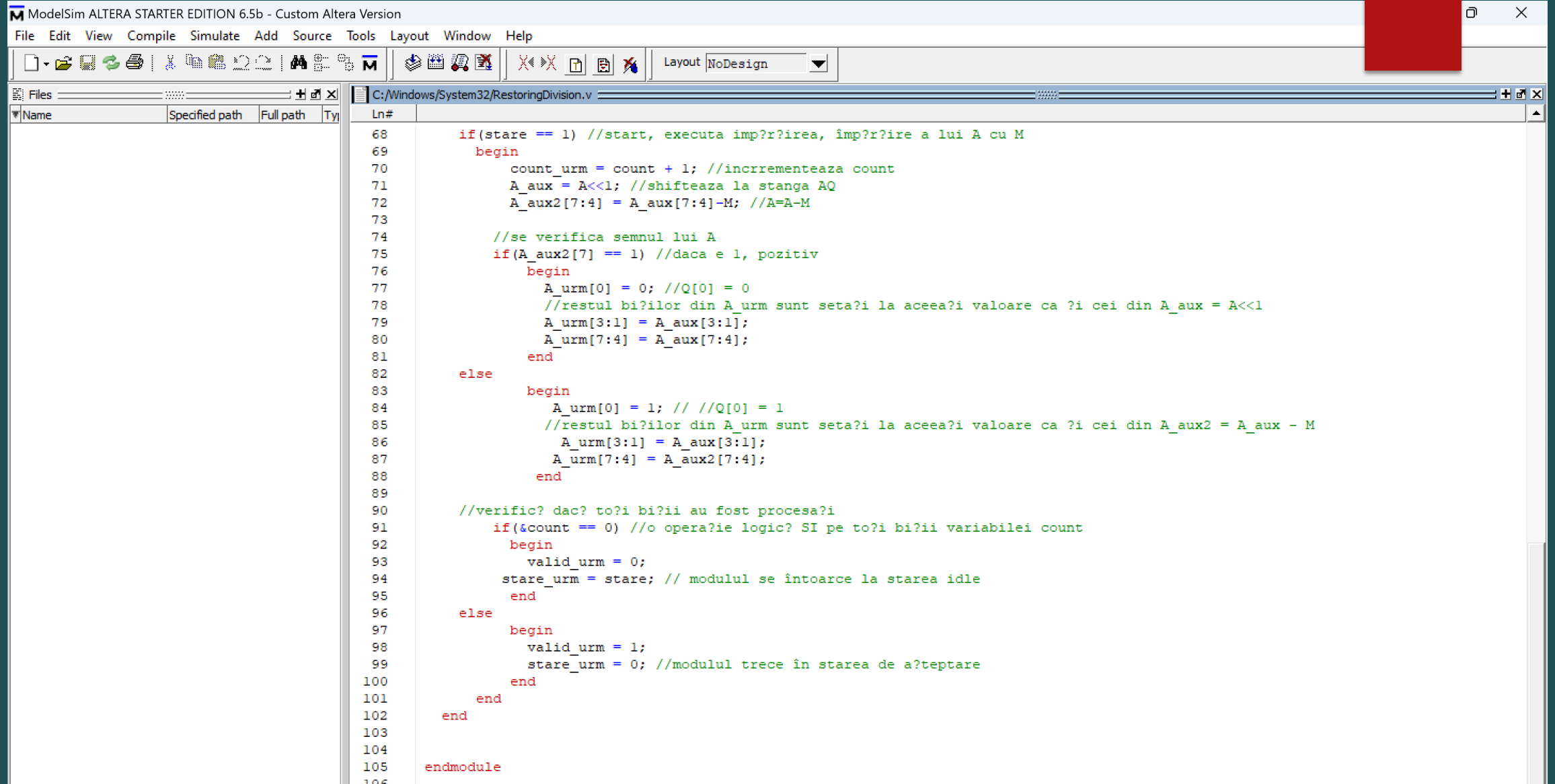


```
ModelSim ALTERA STARTER EDITION 6.5b - Custom Altera Version
File Edit View Compile Simulate Add Files Tools Layout Window Help

C:/Windows/System32/RestoringDivision.v

Ln#
1  module RestoringDivision(
2      clock,
3      reset,
4      start,
5      Q, //deimpartit, A[7:4]
6      M, //impartitor
7      valid,
8      cat,
9      rest
10 );
11
12 input clock;
13 input reset;
14 input start;
15 input [3:0]Q,M;
16
17 output [3:0]cat,rest;
18 output valid;
19
20 //definirea registrilor
21 reg [7:0] A,A_urm,A_aux,A_aux2;
22 reg valid, valid_urm;
23 reg stare, stare_urm;
24 reg [1:0] count,count_urm;
25
26 //extrag date din registrul A si le atribui altor variabile
27 assign rest = A[7:4];
28 assign cat = A[3:0];
29
30 always @(posedge clock or negedge reset) //executa blocul ori de câte ori se produce un front crescator pe semnalul de ceas sau
31 begin
32     if(!reset) //verific? dac? semnalul de reset este inactiv
33     begin//inactiv
34         A <= 0;//setez valorile la iesire
35         valid <= 0;
36         stare <= 0;
37         count <= 0;
38     end
39     else
```





Testare

- ▶ Pentru a testa algoritmul Restoring Division, putem simula modulul într-un simulator Verilog. În această simulare, putem aplica semnale de testare pentru deimpartit și impartitor și putem verifica semnalele de ieșire pentru a confirma că modulul funcționează așa cum este prevăzut. De asemenea, putem verifica rezultatele împărțirii pentru diferite combinații de deimpartit și impartitor pentru a asigura că algoritmul produce întotdeauna un rezultat corect.

Testbench:

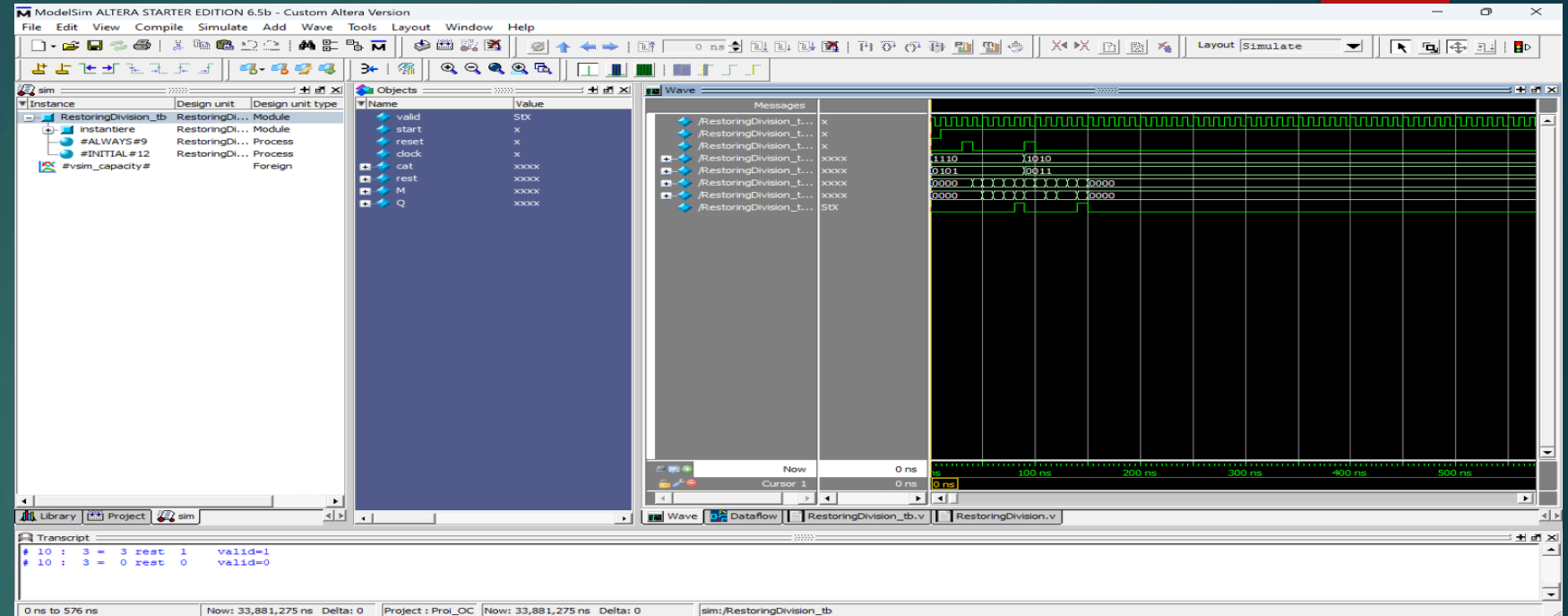
```
ModelSim ALTERA STARTER EDITION 6.5b - Custom Altera Version
File Edit View Compile Simulate Add Source Tools Layout Window Help

C:/Windows/System32/RestoringDivision_tb.v

Ln#
1 module RestoringDivision_tb;
2
3 reg clock,reset,start;
4 reg [3:0]Q,M;
5
6 wire [3:0]cat,rest;//pentru a obține ieșirile cat și rest
7 wire valid; //semnalul valid care indică dacă rezultatul este disponibil
8
9 always #5 clock = ~clock; //"clock" e alternativ 0 și 1 la fiecare perioadă a semnalului de ceas
10
11
12 initial
13 begin
14     $dumpfile("dump.vcd");
15     $dumpvars(1);
16
17     $monitor("%d : %d = %d rest %d    valid=%d",Q,M,cat,rest,valid);
18
19     Q=14;
20     M=5;
21     //așteaptă până când semnalul valid este activat
22     clock=1;
23     reset=0;
24     start=0;
25
26     #10
27     reset = 1;
28
29     #20
30     start = 1;
31
32     #10
33     start = 0;
34
35     @valid
36
37     #10
38     $display("\n");
39
40     //așteaptă
```

```
Ln#
26      #10
27      reset = 1;
28
29      #20
30      start = 1;
31
32      #10
33      start = 0;
34
35      @valid
36
37      #10
38      $display("\n");
39
40      //reia testul
41      Q=10;
42      M=3;
43
44      start = 1;
45
46      #10
47      start = 0;
48
49      end
50
51      RestoringDivision instantiere(
52          .clock(clock),
53          .reset(reset),
54          .start(start),
55          .Q(Q),
56          .M(M),
57          .valid(valid),
58          .cat(cat),
59          .rest(rest)
60      );
```

Simulare



- ▶ În timpul simulării, putem observa valorile semnalelor la fiecare perioadă a semnalului de ceas. Putem analiza semnalele și putem verifica dacă rezultatele sunt cele așteptate. În cazul în care avem semnale de ieșire neașteptate, putem utiliza informațiile obținute din simulare pentru a identifica și corecta problemele din designul nostru.

Concluzie

- ▶ În concluzie, implementarea algoritmului Restoring Division în Verilog este un proces complex, dar bine definit, care implică mai mulți pași. De la proiectarea arhitecturii hardware și a modulului, la scrierea testbencherului și simularea în Verilog, toate acestea sunt necesare pentru a valida funcționarea corectă a algoritmului. Testarea este crucială pentru a verifica dacă modulul Restoring Division generează rezultatele corecte pentru toate cazurile de test.

Bibliografie

- ▶ - <https://www.geeksforgeeks.org/restoring-division-algorithm-unsigned-integer/>
- ▶ <https://atozmath.com/example/RestoringDivision.aspx?he=e>
- ▶ <https://www.gyaanibuddy.com/assignments/assignment-detail/restoring-method-of-division/>
- ▶ <https://www.javatpoint.com/restoring-division-algorithm-for-unsigned-integer>