Seminar Logică Digitală săptămâna 12

Ștefan-Alexandru Jura

17 Mai 2024

1 Calea de control şi calea de date.

În cadrul seminarului precedent, s-a construit design-ul unui înmuţitor, operaţia de înmulţire fiind realizată prin adunări repetate. Implementarea hardware a înmulţitorului s-a realizat prin intermediul căii de control şi a căii de date, punctându-se care este legătura dintre cele două şi cum funcţionează, ce presupune fiecare dintre ele (a se revedea principiile explicate în cadrul seminarului trecut!). În continuare, se vor aplica aceleaşi principii în vederea consolidării cunoştinţelor, pentru o altă problemă.

2 Enunţ.

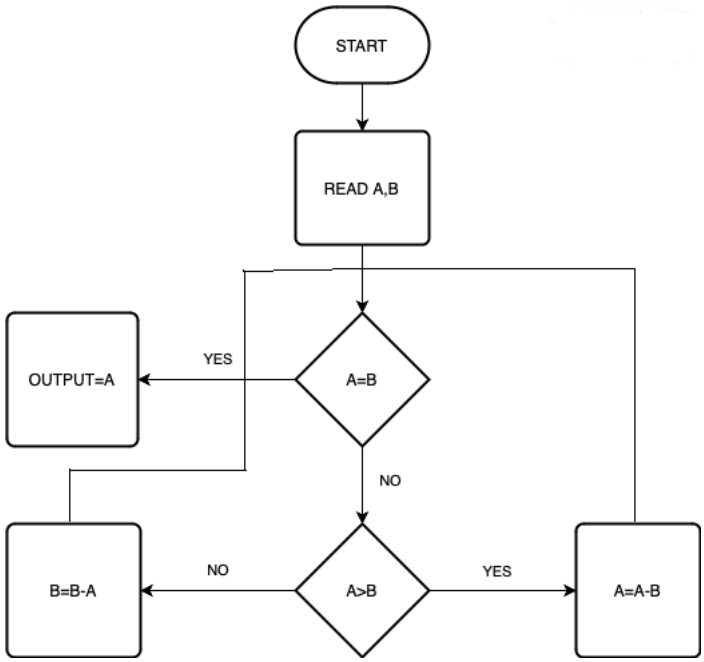
Se cere implementarea unităţii de control şi a căii de date pentru un circuit ce calculează cel mai mare divizor comun (CMMDC) a 2 numere în semn mărime.
Soluţie: Se va folosi o variantă a algoritmului lui Euclid pentru determinarea CMMDC, folosind scăderi repetate. Dacă $a < b$, atunci $b = b - a$, altfel $a = a - b$. Dacă $a = b$, atunci CMMDC va fi oricare dintre valorile celor 2 operanzi. Pseudocodul aferent algoritmului este

prezentat mai jos:

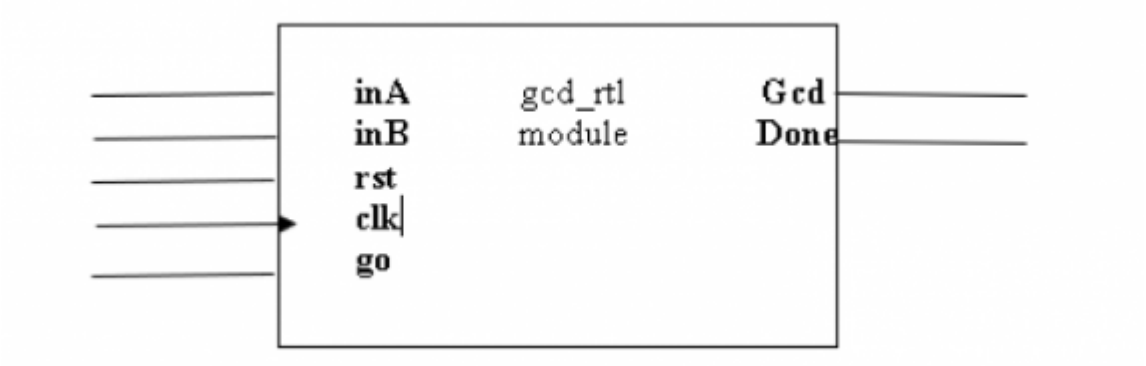
Algorithm 1 Calculul CMMDC al numerelor a și b prin scăderi repetate

```
1: procedure CMMDC(a, b)
2:     while a ≠ b do
3:         if a > b then
4:             a ← a − b
5:         else
6:             b ← b − a
7:         end if
8:     end while
9:     return a
10: end procedure
```

Ordinograma aferentă algoritmului este:

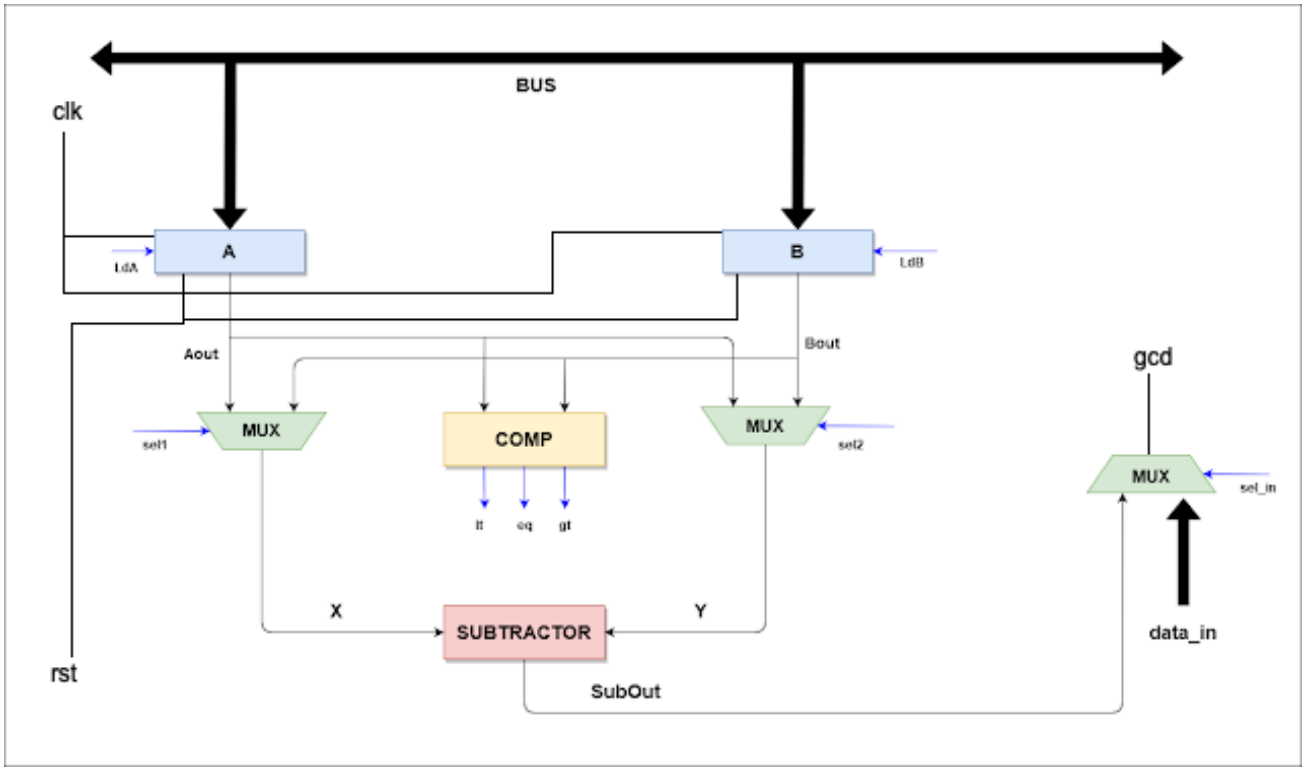


Blocul aferent circuitului digital proiectat este:



3 Implementarea căii de date.

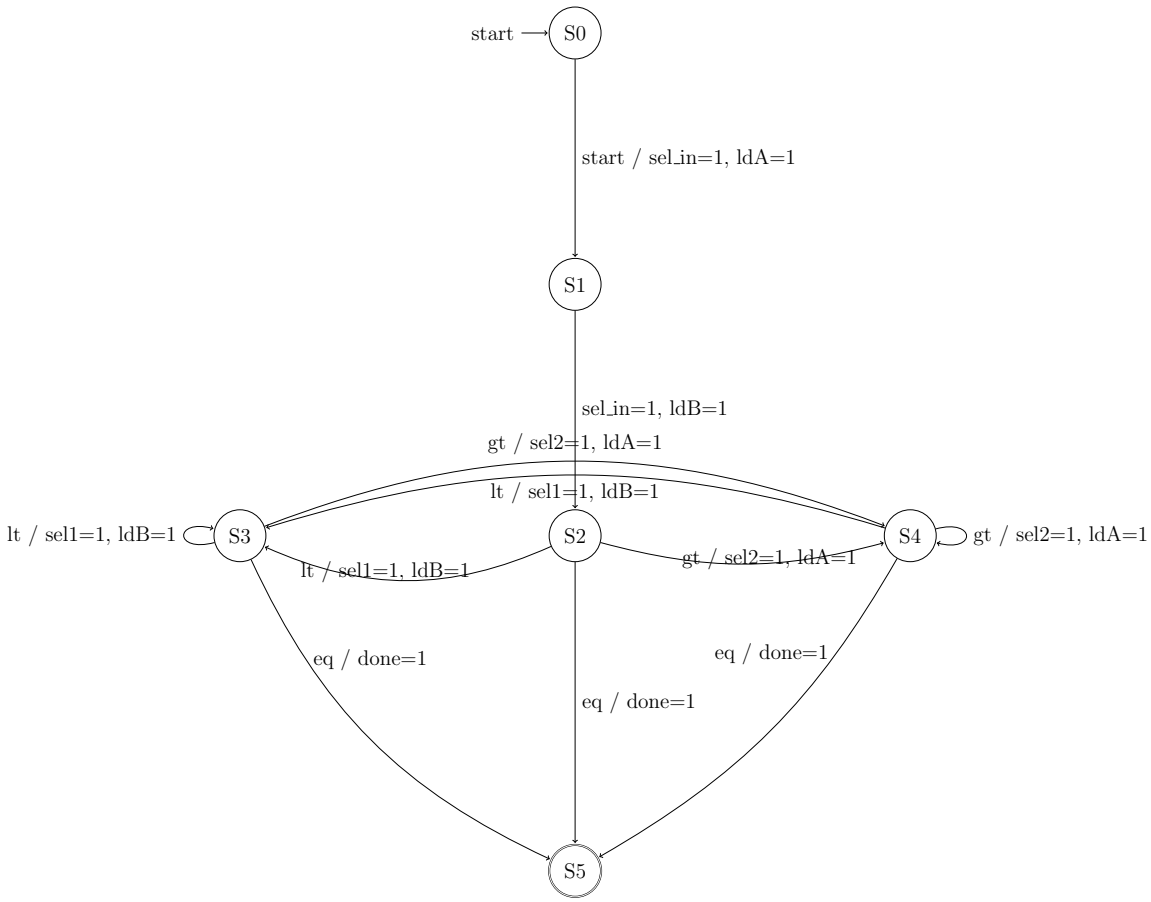
Din diagrama din figura precedentă, identificăm elementele necesare în calea de date. În primul rând, trebuie să citim cele două numere, A și B, pentru care vom avea nevoie de două registre. Apoi, este necesar un circuit de scădere, deoarece vom efectua operații de tipul A-B și B-A. Astfel, intrările trebuie să treacă printr-un modul multiplexor care selectează fie A-B, fie B-A. În final, pentru a realiza comparația, avem nevoie de un circuit comparator. Calea de date este prezentată mai jos:



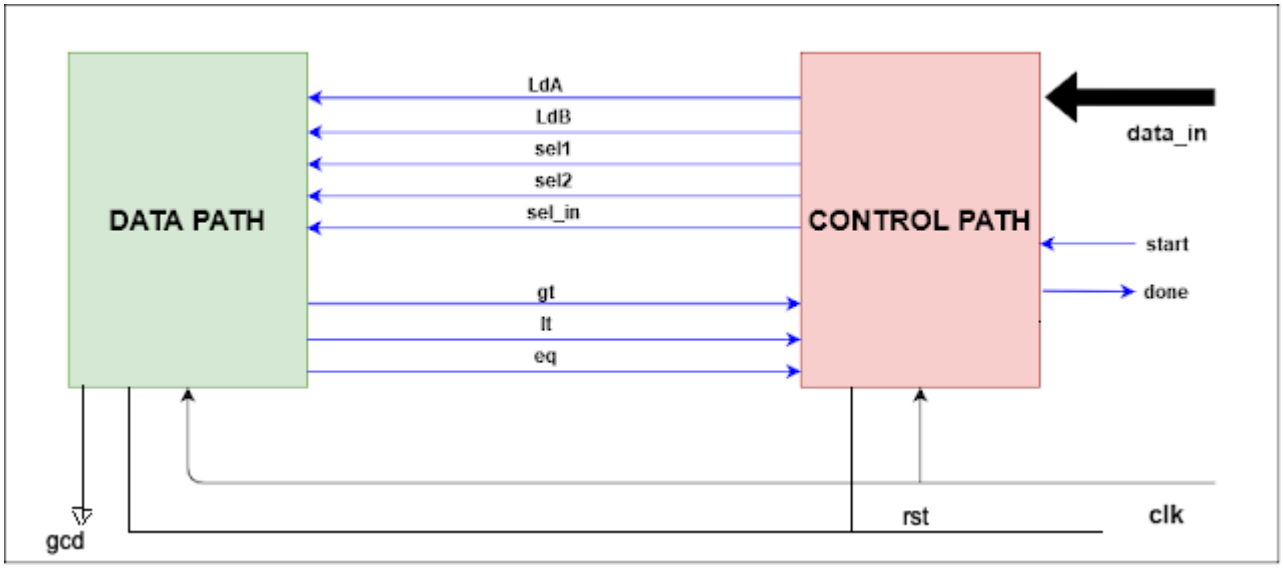
Pentru a implementa scăzătorul, se va utiliza un scăzător format din FSC (Full Substracter Cells), similar cu sumatorul de tip RCA cu FAC (Full Adder Cells). Aici nu mai aveam carry in şi carry out, ci borrow in şi borrow out (la scădere ne "împrumutăm", analogie cu scăderea din zecimal). Circuitul "Substractor" din calea de date este format din n module FSC, unde n este numărul de biţi al operanzilor. În cazul căii de date, se observă că are drept semnale de ieşire care vor merge spre calea de control gt (greater than), lt (less than) şi eq (equal). Aceste semnale transmit căii de control ce operaţie să realizeze (a-b sau b-a).

4 Implementarea căii de control.

Calea de control este ilustrată prin intermediul unui FSM Mealy cu 5 stări, reprezentat mai jos:



Tranziţia dintre stări este aleasă în funcţie de semnale. Practic, aşa se decide care operaţie se realizează (a-b daca a gt b sau b-a dacă a lt b; dacă a=b, atunci se activează semnalul eq venit din calea de date care marchează finalul algoritmului). Relaţia dintre calea de date şi calea de control este evidenţiată mai jos:



Rolul fiecărui semnal de control furnizat de calea de control este:
ldA - semnal pentru încărcarea primului număr în registru;
ldB - semnal pentru încărcarea celui de-al doilea număr în alt registru; sel1 şi sel2 - intrări de selecţie pentru primul multiplexor (cel din stânga), respectiv cel de-al doilea (din dreapta) din calea de date, asigurând selecţia ordinii operanzilor (dacă se face a-b sau b-a);
sel_in, care dacă este 0, încarcă rezultatul din scăzător; done - semnifică finalul algoritmului.
De asemenea, calea de control are ca intrare semnalul de clock (care apare ca input şi în calea de date), semnalul start care asigură începerea algoritmului, şi semnalele gt, lt şi eq care stabilesc ordinea operanzilor şi ce operaţie se realizează.
Codul Verilog aferent sintezei unităţii de control este:

```
1  `timescale 1ns / 1ps
2
3  module controller(
4      input clk,
5      input rst,
```

```
6       input lt,
7       input gt,
8       input eq,
9       input start,
10      output reg ldA,
11      output reg ldB,
12      output reg sel1,
13      output reg sel2,
14      output reg sel_in,
15      output reg done
16      );
17
18      reg [2:0] state, state_nxt;
19  parameter S0 = 3'b000, S1 = 3'b001, S2 = 3'b010, S3 = 3'b011, S4 = 3'b1
20
21  /************** State Transitions ******************/
22
23  always @(posedge clk or posedge rst)
24  begin
25      if (rst)
26          state <= S0;
27      else
28          state <= state_nxt;
29  end
30
31  /********** Generation of Load Signals ***************/
32
33  always@(*)
34  begin
35      // Default values to avoid latches
36      state_nxt = state;
37      {sel_in, ldA, ldB, done, sel1, sel2} = 6'b000000;
38
39      case(state)
40          S0: begin
41                  sel_in = 1;
42                  ldA = 1;
43                  if(start) state_nxt = S1;
```

```
44              end
45          S1: begin
46                  sel_in = 1;
47                  ldB = 1;
48                  state_nxt = S2;
49              end
50          S2: begin
51                  if(eq) begin
52                      done = 1;
53                      state_nxt = S5;
54                  end else if(lt) begin
55                      sel1 = 1;
56                      ldB = 1;
57                      state_nxt = S3;
58                  end else if(gt) begin
59                      sel2 = 1;
60                      ldA = 1;
61                      state_nxt = S4;
62                  end
63              end
64          S3: begin
65                  if(eq) begin
66                      done = 1;
67                      state_nxt = S5;
68                  end else if(lt) begin
69                      sel1 = 1;
70                      ldB = 1;
71                      state_nxt = S3;
72                  end else if(gt) begin
73                      sel2 = 1;
74                      ldA = 1;
75                      state_nxt = S4;
76                  end
77              end
78          S4: begin
79                  if(eq) begin
80                      done = 1;
81                      state_nxt = S5;
```

```
                end else if(lt) begin
                    sel1 = 1;
                    ldB = 1;
                    state_nxt = S3;
                end else if(gt) begin
                    sel2 = 1;
                    ldA = 1;
                    state_nxt = S4;
                end
            end
        S5: begin
                done = 1;
                state_nxt = S5;
            end
        default: state_nxt = S0;
    endcase
end

endmodule
```

Codul aferent căii de date este:

```
`timescale 1ns / 1ps

module GCD_datapath(
    input ldA,
    input ldB,
    input sel1,
    input sel2,
    input sel_in,
    input clk,
    input rst,
    input [15:0] data_in,
    output gt,
    output lt,
    output eq
    );

```

```
17 wire [15:0] Aout, Bout, X, Y, Bus, SubOut;
18
19 PIPO A (Aout, Bus, ldA, clk, rst);
20 PIPO B (Bout, Bus, ldB, clk, rst);
21 MUX MUX_in1 (X, Aout, Bout, sel1);
22 MUX MUX_in2 (Y, Aout, Bout, sel2);
23 MUX MUX_load (Bus, SubOut, data_in, sel_in);
24 SUB SB (SubOut, X, Y);
25 COMPARE COMP (lt, gt, eq, Aout, Bout);
26
27 endmodule
28
29
30 module PIPO(data_out, data_in, load, clk, rst);
31
32 input [15:0] data_in;
33 input load, clk, rst;
34 output reg [15:0] data_out;
35
36 always @(posedge clk or posedge rst)
37     if (rst)
38         data_out <= 16'b0;
39     else if (load)
40         data_out <= data_in;
41
42 endmodule
43
44
45 module MUX (out, in0, in1, sel);
46
47 input [15:0] in0, in1;
48 input sel;
49 output [15:0] out;
50
51 assign out = sel ? in1 : in0;
52
53 endmodule
54
```

```
55
56 module COMPARE (lt, gt, eq, data1, data2);
57
58 input [15:0] data1, data2;
59 output lt, gt, eq;
60
61 assign lt = data1 < data2;
62 assign gt = data1 > data2;
63 assign eq = data1 == data2;
64
65 endmodule
66
67 module SUB (out, in1, in2);
68
69 input [15:0] in1, in2;
70 output reg [15:0] out;
71
72 always @(*)
73     out = in1 - in2;
74
75 endmodule
```

iar testbench-ul (cu cazurile 0 şi 1, limita maximă admisă de un întreg fără semn pe 16 biţi şi 2 cazuri între limitele admise) este:

```
1     `timescale 1ns / 1ps
2
3 module GCD_testbench;
4
5 reg [15:0] data_in;
6 reg clk, rst, start;
7 wire done;
8 wire ldA, ldB, sel1, sel2, sel_in;
9 wire gt, lt, eq;
10 wire [15:0] Aout, Bout;
11
12 GCD_datapath DP (
13     .ldA(ldA),
```

```
14      .ldB(ldB),
15      .sel1(sel1),
16      .sel2(sel2),
17      .sel_in(sel_in),
18      .clk(clk),
19      .rst(rst),
20      .data_in(data_in),
21      .gt(gt),
22      .lt(lt),
23      .eq(eq),
24      .Aout(Aout),
25      .Bout(Bout)
26  );
27
28  controller CON (
29      .clk(clk),
30      .rst(rst),
31      .lt(lt),
32      .gt(gt),
33      .eq(eq),
34      .start(start),
35      .ldA(ldA),
36      .ldB(ldB),
37      .sel1(sel1),
38      .sel2(sel2),
39      .sel_in(sel_in),
40      .done(done)
41  );
42
43  always #5 clk = ~clk;
44
45  initial begin
46      clk = 1'b0;
47      rst = 1'b1;
48      start = 1'b0;
49      data_in = 16'b0;
50
51
```

```
52      #10 rst = 1'b0;
53      #3 start = 1'b1;
54
55
56      #12 data_in = 143;
57      #10 data_in = 78;
58
59
60      wait (done);
61
62      #20 start = 1'b0;
63      #10 rst = 1'b1;
64      #10 rst = 1'b0;
65      #3 start = 1'b1;
66      #12 data_in = 48;
67      #10 data_in = 18;
68
69      wait (done);
70
71      #20 start = 1'b0;
72      #10 rst = 1'b1;
73      #10 rst = 1'b0;
74      #3 start = 1'b1;
75      #12 data_in = 101;
76      #10 data_in = 10;
77
78      wait (done);
79
80      #20 start = 1'b0;
81      #10 rst = 1'b1;
82      #10 rst = 1'b0;
83      #3 start = 1'b1;
84      #12 data_in = 0;
85      #10 data_in = 1;
86
87      wait (done);
88
89      #20 start = 1'b0;
```

```
90      #10 rst = 1'b1;
91      #10 rst = 1'b0;
92      #3 start = 1'b1;
93      #12 data_in = 65535;
94      #10 data_in = 65535;
95
96      #1000 $finish;
97  end
98
99  initial begin
100     $monitor($time, " Aout = %d, Bout = %d, done = %b", Aout, Bout, don
101     $dumpfile("gcd.vcd");
102     $dumpvars(0, GCD_testbench);
103 end
104
105 endmodule
```

5 Temă.

1) Se cer:
a) implementarea căii de control (pe foaie) cu bistabile D;
b) diagrama de timp pentru calea de control (tot pe foaie);
2) Să se proiecteze (pe hârtie) un circuit secvenţial sincron care să realizeze apariţia stărilor prime din intervalul 0-15 şi cea a stărilor pare cuprinse între 0 şi 8.