

HDL

Hardware Description Languages

## Tema 2: Algoritm de inmultire cu deplasare stinga

Anul de studiu: 3

Nume si prenume student: Bentia Ioan Daniel

Grupa: 4981

Profesor : Gaspar Zoltan

Anul universitar: 2010-2011

**Algoritm de inmultire cu deplasare stinga**

- reseteaza P (2\*n biti)

- incarca deinmultitul in A (n biti)

- incarca inmultitorul in B (n biti)

- repeta de n ori

- deplaseaza cu o pozitie stinga P (LSB P = 0)

- daca MSB A = 1 atunci

- P <= P + B

- deplaseaza cu o pozitie stinga A (LSB A = X)

- P contine produs

233 x 195 = 45435

233 = 1110 1001

195 = 1100 0011

Reg. P Reg. A

----------- ---------

0000 0000 0000 0000 *1*110 1001 - initial

0000 0000 0000 0000

1100 0011

-------------------

0000 0000 1100 0011

0000 0001 1000 0110 *1*101 001X - iteratia 1

0000 0001 1000 0110

1100 0011

-------------------

0000 0010 0100 1001

0000 0100 1001 0010 *1*010 01XX - iteratia 2

0000 0100 1001 0010

1100 0011

-------------------

0000 0101 0101 0101

0000 1010 1010 1010 *0*100 1XXX - iteratia 3

0001 0101 0101 0100 *1*001 XXXX - iteratia 4

0001 0101 0101 0100

1100 0011

-------------------

0001 0110 0001 0111

0010 1100 0010 1110 *0*01X XXXX - iteratia 5

0101 1000 0101 1100 *0*1XX XXXX - iteratia 6

1011 0000 1011 1000 *1*XXX XXXX - iteratia 7

1011 0000 1011 1000

1100 0011

-------------------

**1011 0001 0111 1011**

**-------------------**

**Produs = 11x4096 + 1x256 + 7x16 + 11 = 45435**

**CALE DE CONTROL**

module cale\_de\_control (start,reset,clk,load, busy, ready);

input start;

input reset;

input clk;

input ready;

output load;

output busy;

reg load;

reg busy;

always @(posedge clk or posedge reset)

if (reset)

load <= 'b0; else

if (start)

load <= 'b1;

else load <= 0;

always @(posedge clk or posedge reset)

if (reset)

busy <= 'b0; else

if (ready)

busy <= 'b0; else

busy <= 'b1;

endmodule

**CALE DE DATE**

module cale\_de\_date(load,clk,reset,busy,OpA,OpB,mul,ready);

parameter width = 8;

//input Ctrl;

output [{2\*width-1}:0] mul;

output ready;

input clk;

input reset;

input load;

input busy;

input [width-1:0] OpA;

input [width-1:0] OpB;

reg [{2\*width-1}:0] P;

reg [width-1:0] A;

reg [width-1:0] B;

reg readyR;

reg [{2\*width-1}:0] Control;

reg [1:0] wait1;

wire[{2\*width-1}:0] mul;

wire ready;

initial begin

readyR <= 'b0;

P <= 'b0;

wait1 <= 'b0;

end

always @ (posedge clk or reset)

begin

if (reset) begin

A <= 'b0;

readyR <= 1'b0;

Control <= 'b0; end

else

if(load)

A <= OpA;

if(wait1 == 1)

if(((Control+1)<={2\*width})&&busy)begin

A <= A<<1;

A[0] = 1'bx; end

if(busy)

if(Control < {2\*width})

Control <= Control+1;

else begin

Control <= 'b0;

readyR <= 1;

end

wait1 <= wait1+1;

if (wait1 == 2) wait1 <= 'b0;

end

always @ (posedge clk or reset)

begin

if (reset)

B <= 'b0;

else

if(load)

B <= OpB;

end

always @ (posedge clk or reset)

begin

if (reset)

P <= 'b0;

else

if(wait1 == 0)

if(busy) begin

P <= P<<1;

P[0] <= 0; end

if (wait1 == 1)

if(A[width-1]== 1)

P <= P+B;

else

P <= P;

end

assign mul = P;

assign ready = readyR;

endmodule

**GENERATOR**

module tb2(clk,reset,start,OpA,OpB);

parameter width = 8;

output clk;

output start;

output reset;

output[width-1:0] OpA;

output[width-1:0] OpB;

reg clk;

reg reset;

reg start;

reg [width-1:0] OpA;

reg [width-1:0] OpB;

initial begin

OpA <='b11101001;

OpB <='b11000011;

clk <= 'b0;

reset <= 'b0;

#5 reset <= 1;

#40 reset <= 0;

#10 start <= 1;

#30 start <= 0;

end

always #5 clk = ~clk;

endmodule

**INMULTIRE TEST**

module cale\_de\_date\_test();

parameter x = 8;

wire loadMUL;

wire resetMUL;

wire busyMUL;

wire clkMUL;

wire [{2\*x-1}:0] mulMUL;

wire [x-1:0] AMUL;

wire [x-1:0] BMUL;

wire readyMUL;

wire startNET;

cale\_de\_date #(x) DUT\_d ( .load(loadMUL),

.reset(resetMUL),

.mul(mulMUL),

.busy(busyMUL),

.clk(clkMUL),

.OpA(AMUL),

.OpB(BMUL),

.ready(readyMUL));

tb2 #(x) TB( .clk(clkMUL),

.reset(resetMUL),

.OpA(AMUL),

.OpB(BMUL),

.start(startNET));

cale\_de\_control #(x) DUT\_c (.load(loadMUL),

.reset(resetMUL),

.clk(clkMUL),

.busy(busyMUL),

.start(startNET),

.ready(readyMUL));

endmodule