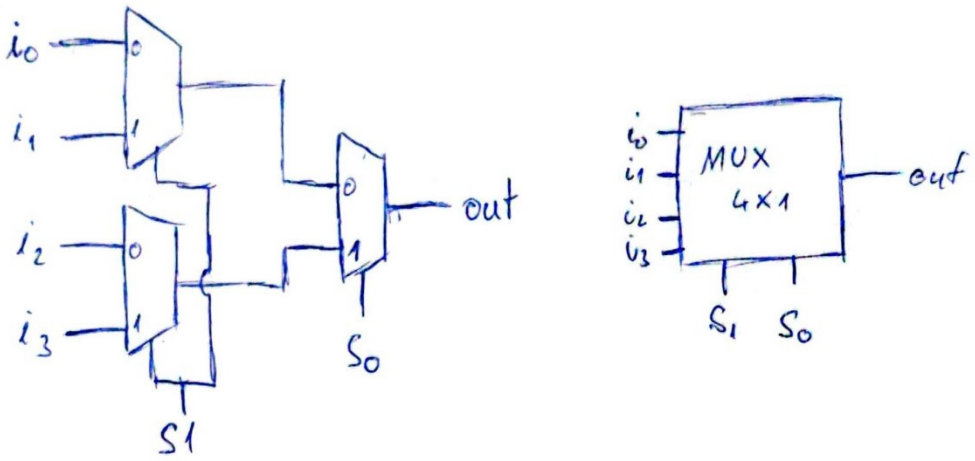
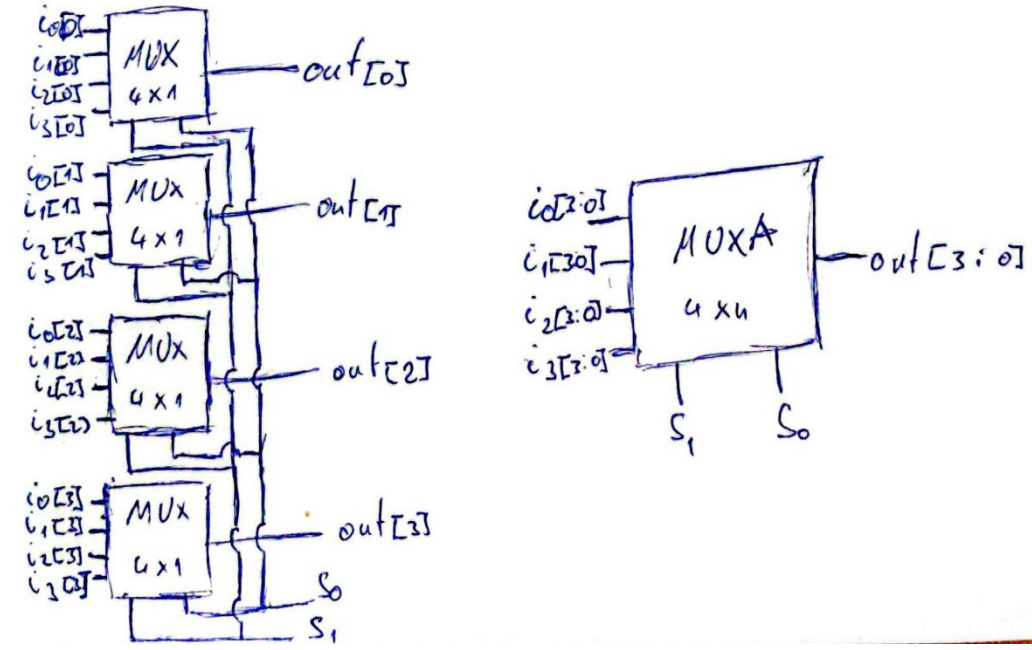
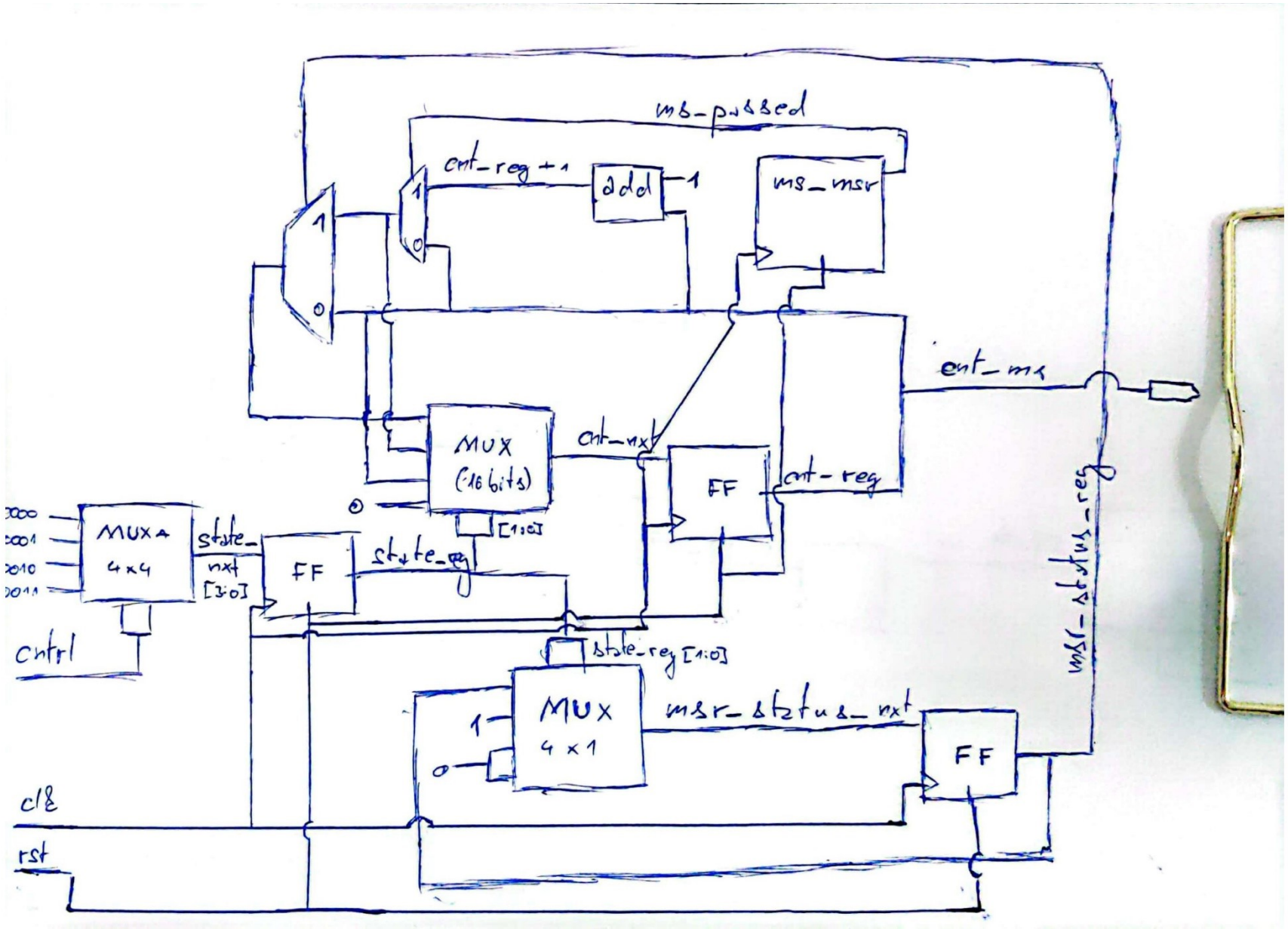
Adaugarea de opcode-uri

MUX 4x1



MUXA 4x4




ms_passed
cnt_reg + 1
add
1
ms_msr
MUX
(16 bits)
cnt_nxt
FF
cnt_reg
cnt_ms
MUX 4x4
state_nxt [3:0]
FF
state_reg
[1:0]
cntrl
state_reg [1:0]
MUX
4 x 1
msr_status_nxt
FF
msr_status_reg
clk
rst

*MUXA 4x4 ar fi de fapt o implementare posibilă pentru “case”. În acest caz, primul bit este ignorat la ieșire.

Module:

```
module ms_cnt(
 input                clk, rst,
 input        [2:0]   cntrl,
 output       [15:0]  cnt_ms
);

 reg    [15:0]  cnt_reg, cnt_nxt;
 reg    [2:0]   state_reg, state_nxt;
 reg                msr_status_reg, msr_status_nxt;
 wire               ms_passed;
 wire   [15:0]  cnt_clk;

 localparam S0 = 3'b000;
 localparam S1 = 3'b001;
 localparam S2 = 3'b010;
 localparam S3 = 3'b011;

 ms_msr ms_msr_1(
  .clk(clk),
  .rst(rst),
  .ms_passed(ms_passed),
  .cnt_clk(cnt_clk)
 );
 always @(*) begin
  case(state_reg)
   S0:  begin
```

```
   if(msr_status_reg & ms_passed)
     cnt_nxt = cnt_reg + 1;
   else
     cnt_nxt = cnt_reg;
  end

  S1:   begin
   msr_status_nxt = 1;
   if(ms_passed)
     cnt_nxt = cnt_reg + 1;
  end

  S2:   begin
   msr_status_nxt = 0;
   cnt_nxt = cnt_reg;
  end

  S3:   begin
   msr_status_nxt = 0;
   cnt_nxt = 0;
  end
endcase

case(cntrl)
  S0:   begin
   state_nxt = S0;
  end

  S1:   begin
```

```
      state_nxt = S1;
    end

    S2:   begin
      state_nxt = S2;
    end

    S3:   begin
      state_nxt = S3;
    end
  endcase

end

 always @(negedge clk or posedge rst) begin
  if(rst == 1) begin
    cnt_reg <= 0;
    msr_status_reg <= 0;
    state_reg <= 0;
  end
  else begin
    cnt_reg <= cnt_nxt;
    msr_status_reg <= msr_status_nxt;
    state_reg <= state_nxt;
  end
 end
 assign cnt_ms = cnt_reg;
endmodule
```

Testbench:

```
//correct functionality for a timescale of 1ns/<precision>
module test;

 initial begin
  $dumpfile("dump.vcd");
  $dumpvars(1, test);
 end

 reg                    clk, rst;
 reg    [2:0]    cntrl;

 wire   [15:0]   cnt_ms;

 parameter period      =      200;          //200ns periods correspond to 5 MHz
 parameter nr_p_s      =      5000;   //nr of periods in 1s

 ms_cnt ms_cnt_1(
  .clk(clk),
  .rst(rst),
  .cnt_ms(cnt_ms),
  .cntrl(cntrl)
 );

 initial begin
  rst = 1;
  clk = 0;
  #10 rst = 0;
  cntrl = 3'b001;
 end
```

```
 initial begin
  repeat(80000)
    #(period/2) clk = ~clk;
 end

 initial begin
  #(2 * nr_p_s * period) cntrl = 3'b010;
  #(1 * nr_p_s * period) cntrl = 3'b001;
  #(1 * nr_p_s * period) cntrl = 3'b000;
  #(1 * nr_p_s * period) cntrl = 3'b011;
  #(1 * nr_p_s * period) cntrl = 3'b001;
 end
endmodule
```

Functionality:

See waveform