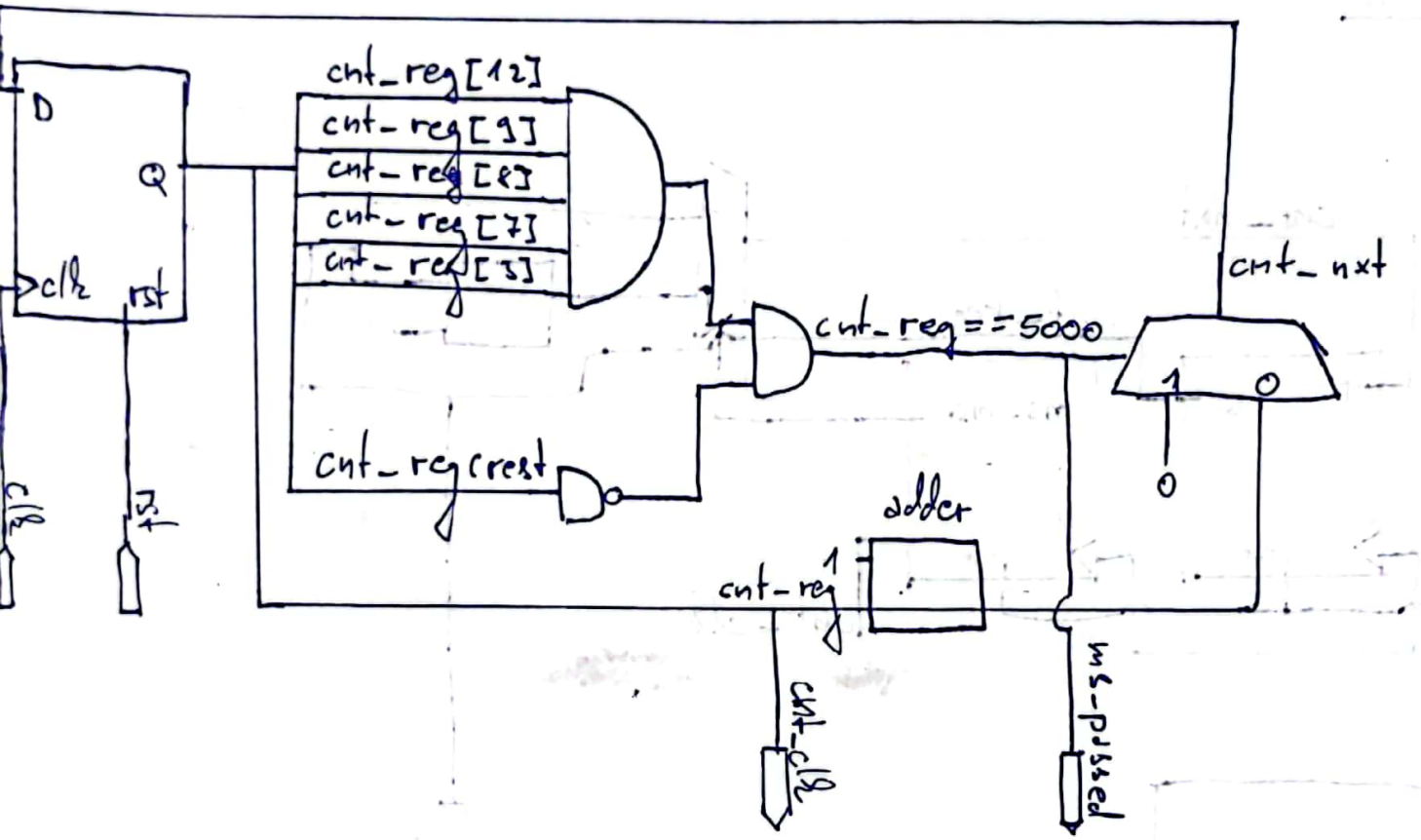
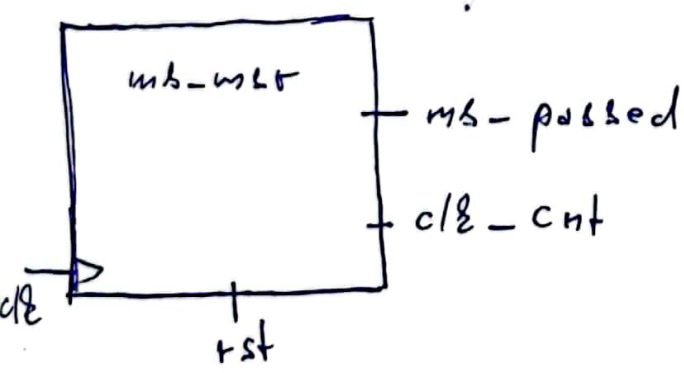
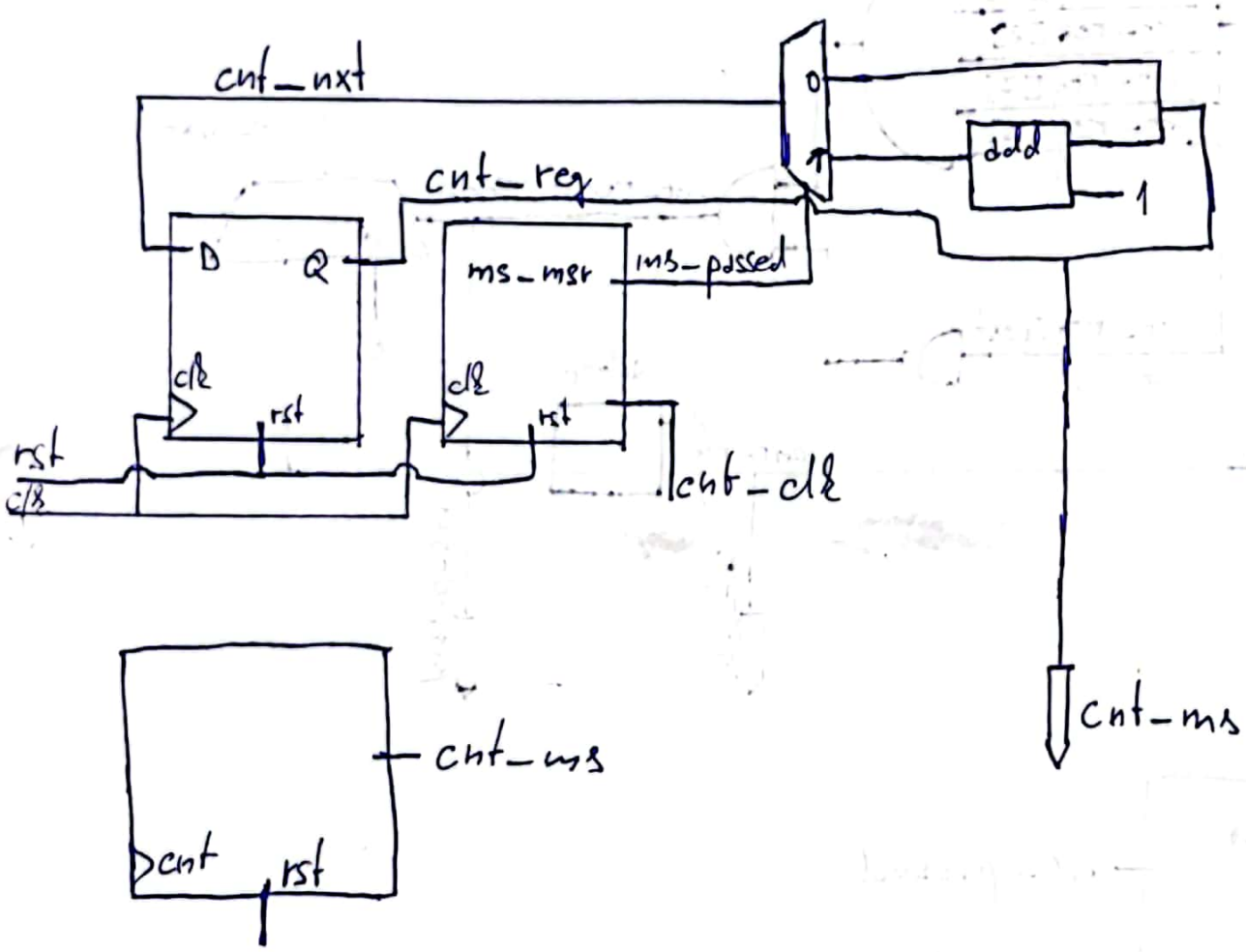
module ms_msr





ms_cnt


cnt_nxt
0
1
add
1
cnt_reg
D
Q
ms_msr
ms_passed
clk
rst
clk
rst
rst
clk
cnt_clk
cnt_ms

cnt_ms
cnt
rst


From:
0ns
To:
4,500,000ns
Get Signals
Radix
100%
0
500,000
1,000,000
1,500,000
2,000,000
2,500,000
3,000,000
3,500,000
4,000,000
clk
cnt_ms[15:0]
0000
0001
0002
0003
0004
rst

▾ Languages & Libraries

Testbench + Design

SystemVerilog/Verilog

UVM / OVM

None

Other Libraries

None
OVL 2.8.1
SVUnit 2.11

- [ ] Enable TL-Verilog
- [ ] Enable Easier UVM
- [ ] Enable VUnit

▾ Tools & Simulators

Synopsys VCS 2020.03

Compile Options

-timescale=1ns/1ns +vcs+flush+all

Run Options

Run Options

- [ ] Use run.do Tcl file
- [ ] Use run.bash shell script
- [x] Open EPWave after run
- [ ] Download files after run

▸ Examples

▾ Community

Collaborate

Forum

SV/Verilog Testbench

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

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

  reg clk, rst;
  wire [15:0] cnt_ms;
  parameter period = 200; //200ns periods correspond to 5 MHz

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

  initial begin
    rst = 1;
    clk = 0;
    #10 rst = 0;
  end

  initial begin
    repeat(45000)
      #(period/2) clk = ~clk;
  end
endmodule
```

```verilog
module ms_msr(
  input clk, rst,
  output ms_passed,
  output [15:0] cnt_clk
);
  reg [15:0] cnt_reg, cnt_nxt;

  always @(*) begin
    cnt_nxt = cnt_reg;

    if(cnt_reg == 5000) begin
      cnt_nxt = 0;
    end
    else begin
      cnt_nxt = cnt_reg + 1;
    end
  end

  always @(negedge clk or posedge rst) begin
    if(rst == 1) begin
      cnt_reg <= 0;
    end
    else begin
      cnt_reg = cnt_nxt;
    end
  end

  assign ms_passed = (cnt_reg == 5000);
  assign cnt_clk = cnt_reg;
endmodule

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

  reg [15:0] cnt_reg, cnt_nxt;
  wire ms_passed;
  wire [15:0] cnt_clk;

  ms_msr ms_msr_1(
    .clk(clk),
    .rst(rst),
    .ms_passed(ms_passed),
    .cnt_clk(cnt_clk)
  );

  always @(*) begin
    cnt_nxt = cnt_reg;

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

  always @(negedge clk or posedge rst) begin
    if(rst == 1) begin
      cnt_reg <= 0;
    end
    else begin
      cnt_reg = cnt_nxt;
    end
  end

  assign cnt_ms = cnt_reg;
endmodule
```