Computer Architecture

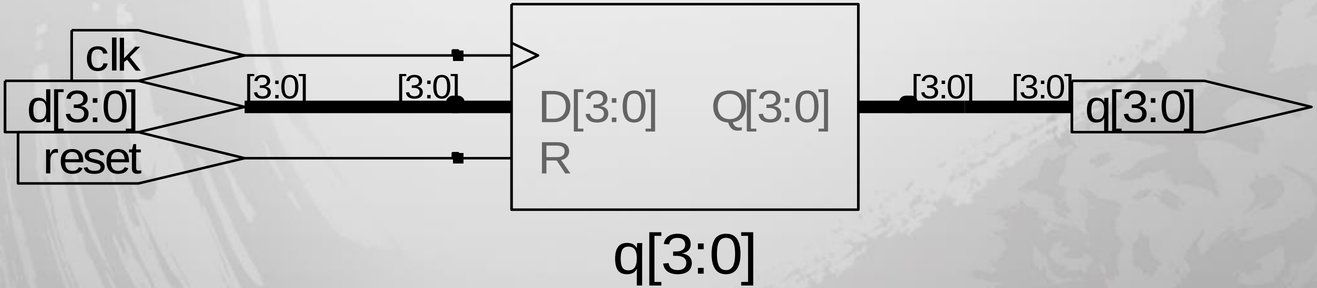
Lab 5: Verilog Basics 2

Getting Started

- Open a web browser and go to https://www.edaplayground.com/
- Create an account with your university email

Resettable D Flip-Flop

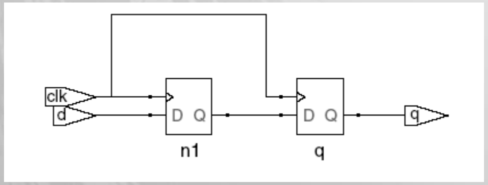
```
module flopr(input  logic       clk,
             input  logic       reset,
             input  logic [3:0] d,
             output logic [3:0] q);

  // synchronous reset
  always_ff @(posedge clk)
    if (reset) q <= 4'b0;
    else       q <= d;

endmodule
```



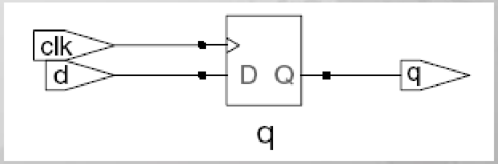
Blocking vs. Nonblocking Assignment

- <= is nonblocking assignment
  - Occurs simultaneously with others
- = is blocking assignment
  - Occurs in order it appears in file

```
// Good synchronizer using
// nonblocking assignments
module syncgood(input  logic clk,
                input  logic d,
                output logic q);
  logic n1;
  always_ff @(posedge clk)
    begin
      n1 <= d;   // nonblocking
      q  <= n1; // nonblocking
    end
endmodule
```



```
// Bad synchronizer using
// blocking assignments
module syncbad(input logic  clk,
               input  logic d,
               output logic q);
  logic n1;
  always_ff @(posedge clk)
    begin
      n1  = d;   // blocking
      q   = n1; // blocking
    end
endmodule
```



Rules for Signal Assignment

- **Synchronous sequential logic**: use `always_ff @(posedge clk)` and nonblocking assignments (<=)

```
always_ff @ (posedge clk)
    q <= d; // nonblocking
```

- **Simple combinational logic**: use continuous assignments (`assign…`)

```
assign y = a & b;
```

- **More complicated combinational logic**: use `always_comb` and blocking assignments (=)

- Assign a signal in **only one** `always` statement or continuous assignment statement.

Lab Assignment

- Write a SystemVerilog code to implement a register file
- It has 32 32-bit registers and the register 0 should be always 0
- It has two read ports and one write port
- It has the following inputs and outputs
  - iClk (1 bit): clock (rising edge)
  - iReset (1 bit): reset signal
  - iRaddr1 (5 bits): the register number of read port 1
  - oRdata1 (32 bits): the output of read port 1
  - iRaddr2 (5 bits): the register number of read port 2
  - oRdata2 (32 bits): the output of read port 2
  - iWaddr (5 bits): the register number of the write port
  - iWdata (32 bits): the input data for the write port
  - iWe (1 bit): the write enable signal
- Use the testbench in the following slide
- Save and submit the link of your design to the Blackboard.
  - Click 🔗 in the bottom window to copy the URL of your design.

Testbench

```
module testbench_regfile();
  logic clk;
  logic reset;
  logic [4:0] raddr1, raddr2, waddr;
  logic we;
  logic [31:0] wdata;
  logic [31:0] rdata1, rdata2;
  logic result;
  logic [31:0] regs[31:0];
  integer i;

  regfile dut(
    .iClk              (clk),
    .iReset            (reset),
    .iRaddr1(raddr1),
    .iRaddr2(raddr2),
    .iWaddr            (waddr),
    .iWe               (we),
    .iWdata            (wdata),
    .oRdata1(rdata1),
    .oRdata2(rdata2)
  );

  always     // no sensitivity list, so it always executes
    begin
      clk = 1; #5; clk = 0; #5;
    end

  initial begin
    result = 1;
    $dumpfile("dump.vcd"); $dumpvars;
    reset = 0; #21;
    raddr1 = 0; raddr2 = 0; waddr = 0; we = 0; wdata = 0;
    for(i=0; i<32; i=i+1) regs[i]=0;
    reset = 1; #10;
    reset = 0; #10;
    for(i=0; i<32; i=i+1) begin
      waddr = i; we = 1; wdata = $random; regs[i] = wdata; #10;
    end
    waddr = 0; we = 0; wdata = 0;
    for(i=0; i<32; i=i+1) begin
      raddr1 = i; raddr2 = i; #1
      if(i==0) begin
        if(rdata1 != 0 | rdata2 != 0) begin
          $display("Read data from r0 failed");
          result = 0;
        end
      end else begin
        if(rdata1 != regs[i]) begin
          $display("Read data from port 1 at address %d failed %x %x", i, rdata1, regs[i]);
          result = 0;
        end
        if(rdata2 != regs[i]) begin
          $display("Read data from port 2 at address %d failed", i);
          result = 0;
        end
      end
      #9;
    end
    if(result)         $display("SUCCESS!");
    else                                      $display("FAILURE!");
    #10; $stop;
  end
endmodule
```