

NMK20603

Computer Architecture

Codes so far:

✓ alu_4b

✓ dff, dreg

Describing Sequential Logic (cont.)

Homework?

⇒ regblock4

Components:

- ✓ dreg (x4)
 - ✓ zbuff (dreg req.)
 - ✓ decode24 (x2)
 - ✓ logic and (x8)
 - OR and_4b (x2)
- ⇒ we got it all!

Describe:

regblock4

⇒ iclk, wenb, renb

⇒ wadd, radd, idat, odat

```

module regblock4 (iclk,wenb,renb,wadd,radd,idat,odat);
parameter BITS=4;
input iclk,wenb,renb;
input[1:0] wadd,radd;
input[BITS-1:0] idat;
output[BITS-1:0] odat;
wire[BITS-1:0] odat;
wire[3:0] wdec,rdec; // output of decode24
wire[3:0] wtmp,rtmp; // output of and-gates
dreg reg0 (iclk,wtmp[0],rtmp[0],idat,odat);
dreg reg1 (iclk,wtmp[1],rtmp[1],idat,odat);
dreg reg2 (iclk,wtmp[2],rtmp[2],idat,odat);
dreg reg3 (iclk,wtmp[3],rtmp[3],idat,odat);
decode24 decW (wadd,wdec);
decode24 decR (radd,rdec);
assign wtmp = wdec & {4{wenb}};
assign rtmp = rdec & {4{renb}};
endmodule

```

Note:

✓ regblock4 NOT scalable

⇒ generate (keyword) statements

⇒ NOT covered (avoid confusion)

How to test?

⇒ move data!

mov dst,src

✓ read from src

✓ write to dst

⇒ separate cycle!

Practice:

⇒ task>reg_move

```
reg renb, wenb;
reg[1:0] rsel, wsel;
reg[BITS-1:0] ddat;
wire[BITS-1:0] mdat;
task reg_move ; // call on -ve edge
    input[1:0] wadd, radd; // dst,src
    begin
        // read cycle
        rsel = radd; renb = 1'b1;
        #(CLKP);
        ddat = mdat;
        // write cycle
        wsel = wadd; wenb = 1'b1;
        #(CLKP);
        wenb = 1'b0; renb = 1'b0;
    end
endtask
```

Practice:

⇒ task>reg_hack

```
task reg_hack; // call on -ve edge
    input[1:0] hadd;
    input[BITS-1:0] hdat;
    begin
        // write cycle
        wsel = hadd; wenb = 1'b1;
        ddat = hdat;
        #(CLKP);
        wenb = 1'b0;
    end
endtask
```

Practice:

⇒ 'monitor' code?

```
always @(posedge dclk) begin
    if (renb===1'b1) begin
        $display("[%05g] R%g (%b) read value %h@(%b)",
            $time,rsel,rsel,mdat,mdat);
    end
    if (wenb===1'b1) begin
        $display("[%05g] R%g (%b) write value %h@(%b)",
            $time,wsel,wsel,ddat,ddat);
    end
end
end
```



```
task reg_showall;
  begin
    $write("[%05g] Register Contents => { ", $time);
    $write("R0:%b ", dut.reg0.tdat);
    $write("R1:%b ", dut.reg1.tdat);
    $write("R2:%b ", dut.reg2.tdat);
    $write("R3:%b }\n", dut.reg3.tdat);
  end
endtask
```

Describe:

regblock4_tb

```
module regblock4_tb ();
parameter BITS=4;
// clock generator
// task:reg_move
// task:reg_hack
// output monitor
// task:reg_showall
// the rest...
initial begin
    dclk = 1'b1;
    wenb = 1'b0; renb = 1'b0;
    // start test
    reg_showall();
    //...
```

```
//...
#(CLKP/2) reg_hack(0,4'ha);
#(CLKP) reg_hack(1,4'h5);
#(CLKP*2) reg_hack(2,4'h9);
#(CLKP) reg_hack(3,4'hd);
reg_showall();
#(CLKP) reg_move(2,3); // mov r2,r3
#(CLKP) reg_move(1,0); // mov r1,r0
$stop;
end
defparam dut.BITS = BITS;
regblock4 dut (dclk,wenb,renb,wsel,rsel,ddat,mdat);
endmodule
```

'Merge' ALU and Registers?

Load/Store

Architecture

⇒ ALU operands from regs

⇒ ALU results to regs

Dual Output Port
for regblock4

⇒ 'Upgrade'!

✓ tb as well...

Lab Project

✓ groups: min=2 max=4

⇒ details coming soon...