NMK20603

Computer Architecture

MY1

Codes so far:

- ✔ alu_4b
- ✔ regblock4

MY1

Processor Core

⇒ what else?

Control Logic!

⇒ sequencing

Describing (Finite) State Machine

State Machine

✔ mathematical model

✔ abstract, finite states

✔ states mutual exclusive

MY1

Components:

✔ current state (seq.)

✔ next state logic (comb.)

✔ output logic (comb.)

MY1

How to 'keep' current state?

Still dff!

- different need

⇒ enb/preset/clear

MY1

Describe:

⇒ dffepc

✔ iclk,ienb

✔ ipre,iclr

✔ idat,odat

MY1

```
module dffepc (iclk,ienb,ipre,iclr,idat,odat);
input iclk,ienb,ipre,iclr;
input idat;
output odat;
reg odat;
// synchronous pre/clr
always @(posedge iclk) begin
  if (ipre===1'b1) begin
    odat = 1'b1;
  end
  else if (iclr===1'b1) begin
    odat = 1'b0;
  end
  else if (ienb===1'b1) begin
    odat = idat;
  end
end
endmodule
```

Practice:

dffepc_tb

MY1

State/Sequence:

⇒ get code

⇒ read reg

⇒ process

⇒ write reg

State Encoding

- ✔ binary
- ✔ gray
- ✔ 1-{hot,cold}

1-hot Encoding

⇒ Ring Counter!

✔ simpler output logic

Note: 'Covers' next state logic!

Describe:

ringctr4

✔ iclk,irst,ienb,ostt

MY1

```
module ringctr4 (iclk,irst,ienb,ostt);
input iclk,irst,ienb;
output[3:0] ostt;
wire[3:0] ostt, qout;
dffepc s0 (iclk,ienb,irst,1'b0,qout[3],qout[0]);
dffepc s1 (iclk,ienb,1'b0,irst,qout[0],qout[1]);
dffepc s2 (iclk,ienb,1'b0,irst,qout[1],qout[2]);
dffepc s3 (iclk,ienb,1'b0,irst,qout[2],qout[3]);
assign ostt = qout;
endmodule
```

Testbench?

MY1

Practice:

⇒ task>state_reset

```
// task to reset system
reg drst;
task state_reset ;
  input integer clen;
  integer iter;
  begin
  $write("[%05g] Reset state machine ",$time);
  $display("for %g clock cycles",clen);
  drst = 1'b1;
  for (iter=0;iter<clen;iter=iter+1) begin
    #(CLKP);
  end
  drst = 1'b0;
  end
endtask
```

Practice:

⇒ task>state_cycle

```
// task to runs single machine cycle
reg denb;
wire[3:0] mstt;
task state_cycle ;
  reg[3:0] curr;
  begin
    $display("[%05g] Running a cycle",$time);
    curr = mstt; denb = 1'b1;
    while (mstt==curr) #(CLKP);
    while (mstt!=curr) #(CLKP);
    denb = 1'b0;
    $display("[%05g] Completed cycle",$time);
  end
endtask
```

MY1

Practice:

⇒ 'monitor' code?

```
always @(mstt) begin // check changes in state
  case (mstt)
    4'b0001: $display("[%05g] STATE0",$time);
    4'b0010: $display("[%05g] STATE1",$time);
    4'b0100: $display("[%05g] STATE2",$time);
    4'b1000: $display("[%05g] STATE3",$time);
    default:
      $display("[%05g] STATEX (%04b)",$time,mstt);
  endcase
end
```

MY1

Describe:

ringctr4_tb

MY1

```
module ringctr4_tb ();
// clock generator
// task:state_reset
// task:state_cycle
// output monitor
// the rest...
initial begin
  // reset code
  dclk = 1'b1; denb = 1'b0; drst = 1'b0;
  $display("[%05g] CurrentState:%b",$time,mstt);
  #(CLKP/2) state_reset(2);
  #(CLKP);
  #(CLKP) state_cycle();
  #(CLKP);
  #(CLKP) state_cycle();
  #(CLKP);
  $finish;
end
ringctr4 dut (dclk,drst,denb,mstt);
endmodule
```

Alternative ringctr4 code?

✔ behavioral!

```
module ringctr4bhv (iclk,irst,ienb,ostt);
input iclk,irst,ienb;
output[3:0] ostt;
reg[3:0] ostt;
always @(posedge iclk) begin
  if (irst==1'b1) begin
    ostt = 4'b0001;
  end
  else if (ienb==1'b1) begin
    case (ostt)
      4'b0001: ostt = 4'b0010;
      4'b0010: ostt = 4'b0100;
      4'b0100: ostt = 4'b1000;
      4'b1000: ostt = 4'b0001;
    endcase
  end
end
endmodule
```

Homework:

⇒ ringctr4bhv_tb