Course: CS 223 Digital Design

Lab: 05

Name: Yahya Mahmoud Ahmed Elnouby Mohamed

Section: 02

ID: 21801332

Date: 10/12/2019

Trainer: No. 19

# Timer Design

```
`timescale 1ns / 1ps

module t20ms(input logic reset, clock, output reg clk2);

//where timer should stop counting
localparam limit = 20000000;
//register to save the value of timer count up
reg[24:0] countUp;

always @ (posedge clock or posedge reset)
begin
   if (reset)
    begin
      countUp = 0;
      clk2  = clk2;
    end

    else if (countUp < limit)
    begin
      countUp++;
      clk2 = clk2;
    end

    else if (countUp >= limit)
    begin
      countUp = 0;
      clk2 = ~clk2;
    end
```

# Counter Design

```
`timescale 1ns / 1ps

module counterUp(input logic reset, clock, limitP, output reg finalCount );

//new clock to be used after time slowing
  wire clk2 ;
//where timer should stop counting
   integer limit = limitP;
//register to save the value of timer count up
   reg[24:0] countUp;

t20ms t20ms (reset, clock,clk2);

always @ (posedge reset or posedge clk2)
begin
   if (reset)
    begin
      finalCount  = 0;
      countUp = 0;
    end

    else if(countUp < limit)
    begin
      countUp++;
      finalCount = finalCount;
    end

    else if(countUp >= limit)
    begin
      countUp = 0;
      finalCount = ~finalCount;
    end

 end
endmodule
```

## Slave Register

```
`timescale 1ns / 1ps

module slaveReg(input logic reset, clock, output reg [7:0] countUp );

//will be used as for the timeup flag of the counter
wire finalCount;

//counter for slave speed 500 ms
counterUp count (reset, clock,50, finalCount);

always @ (posedge finalCount or posedge reset)
begin
   if (reset)
        begin
          countUp = 0;
        end
    else
       begin
         countUp++;
       end

 end
endmodule
```

## Master Register

```
`timescale 1ns / 1ps

module masterReg(input logic reset, clock, output reg [7:0] countUp );

//will be used as for the timeup flag of the counter
wire finalCount;

//counter for master speed 1000ms
counterUp count  (reset, clock, 25,finalCount);

always @ (posedge finalCount or posedge reset)
   begin
      if (reset)
          begin
            countUp = 0;
        end
      else
        begin
         countUp++;
        end
   end
endmodule
```

## Master Slave Design

```
`timescale 1ns / 1ps

module masterSlave(input logic reset, clock, master, slave, enableMaster, enableSlave , output logic masterOut,
slaveOut, masterCout, slaveCout, output [6:0]seg,output dp, output [3:0] an);

reg[7:0] hexaDisplay;    //register to store the count for the seven-segment display

//slave register, clear, timeup flag
reg clearSlave;              //to reset slave count to zero
wire[7:0] slaveRegOut;       //store the count value of slave
wire finalCountSlave;    //timeup flag of slave

//master register, clear, timeup flag
reg clearMaster;         //to reset master count to zero
wire finalCountMaster;  //timeup flag of master
wire [7:0] masterRegOut;    //store the count value of master

//slave counter and slave register to save the value of the slave counter independently
counterUp slaveCounter (clearSlave , clock ,50, finalCountSlave);
slaveReg slaveReg1(clearSlave,clock,slaveRegOut);

//master counter and master register to save the value of the master counter independently
counterUp masterCounter (clearMaster,clock,25,finalCountMaster);
masterReg masterReg1(clearMaster,clock,masterRegOut);

//Output States when certain inputs are changed
always @(posedge clock or posedge reset)
begin

if( (!master && !slave && !enableMaster && !enableSlave) || (reset))
begin
masterOut =0; slaveOut = 0; masterCout =0;
slaveCout = 0; hexaDisplay = 0;
clearMaster = 1; clearSlave = 1;
end

else if(enableMaster || master)
begin
clearSlave = 1; masterOut = 1;
clearMaster= 0;  slaveOut =0; slaveCout = 0;
masterCout = finalCountMaster;
hexaDisplay = masterRegOut;

end

else if(!master && slave &&!enableMaster || !master && !enableMaster && enableSlave)
begin
clearMaster = 1; slaveOut = 1;
clearSlave = 0; masterOut= 0; masterCout = 0;
slaveCout = finalCountSlave;
hexaDisplay = slaveRegOut;
end
end
Seven_Seg sev(clock, hexaDisplay[3:0], hexaDisplay[7:4],0,0 ,seg, dp ,an);
endmodule
```

## TestBench of The Master Slave Design

```
module masterSlave_tb();

reg clock, reset, master, slave, enableMaster, enableSlave;
wire masterOut, slaveOut, masterCout, slaveCout, seg, dp, an;

masterSlave uut (.reset(reset),.clock(clock), .master(master),.slave(slave),
.enableMaster(enableMaster), .enableSlave(enableSlave));
initial begin

reset <=1; #1000;

reset<= 0; enableMaster <= 1; #1000;

reset <= 0; master <= 0; slave <= 0; enableMaster <=0; enableSlave <= 0; #1000;

reset<= 0; master <=1; #1000;

reset<= 0; master = 0; enableMaster =0; slave =1; #1000;

reset<= 0; master = 0; enableMaster =0; enableSlave =1; #1000;


end
  always
    begin
      clock = 0; #10000;
      clock = 1; #10000;
```