# Lab 10 – Worksheet

| Name:Syed Asghar Abbas Zaidi | ID:07201 | Section:T6 |
| --- | --- | --- |

## Task a. h\_counter

*Provide appropriately commented code for your h\_count design module*

| `timescale 1ns / 1ps  module Lab10TaskA(  input clk,  output [9:0] h\_count,  output trig\_v  );  reg [9:0] h\_count;  reg trig\_v;  initial trig\_v = 0;  initial h\_count = 0;  always @ (posedge clk)  begin  if (h\_count < 799)  begin  h\_count <= h\_count + 1;  trig\_v <= 0;  end  else  begin  trig\_v <= 1;  h\_count <= 0;  end  end  endmodule |
| --- |

Modify the testbench provided in Figure 10. 9(b) to verify the functionality of your designed h\_counter module.

*Attach your testbench here.*

| `timescale 1ns / 1ps  module h\_counter();  reg clk;  wire [9:0] h\_count;  wire trig\_v;  Lab10TaskA c1(.clk(clk),.h\_count(h\_count),.trig\_v(trig\_v));  initial  clk = 1'b0;  always  #5 clk = ~clk;    endmodule |
| --- |

*Attach screenshot of waveform here*

|  |
| --- |

### Concept check:

Assuming that a time unit, in above mentioned testbench, is defined in nano seconds; what is the frequency at which the designed h\_counter module would operate?

|  |
| --- |

## Task b. v\_counter

*Provide appropriately commented code for your v\_counter design module*

| `timescale 1ns / 1ps  module V\_Counter(  input clk,  input enable\_v,  output [9:0] v\_count  );    reg [9:0] v\_count;  //reg trig\_v;  initial v\_count = 0;  always @ (posedge clk)  begin  if (v\_count <= 524)  begin @ (posedge enable\_v)  //begin  v\_count <= v\_count + 1;  end  else  //begin  begin @ (posedge enable\_v)  v\_count <= 0;  end  end  endmodule |
| --- |

Modify the testbench provided in Figure 10. 9(b) to verify the functionality of your designed v\_counter module.

*Attach your testbench here.*

| `timescale 1ns / 1ps  module Simulation();  reg clk;  reg enable\_v;  wire [9:0] v\_count;  V\_Counter c1(.clk(clk),.enable\_v(enable\_v),.v\_count(v\_count));  initial  begin  clk = 1'b0;  enable\_v = 1'b0;    // #2000 enable\_v = 1'b1;  end    always #5 clk <= ~clk;  always #5 enable\_v <= ~enable\_v;  endmodule |
| --- |

*Attach screenshot of waveform here*

|  |
| --- |

## Task c: Clock divider

*Show working to calculate div\_value*

| {(100x10^6)/[(2)x(25x10^6)]}-1 = 1 |
| --- |

*Provide appropriately commented code for your clk\_div design module*

| module clk\_div(  input clk,  output clk\_d  );  parameter div\_value = 1;  reg clk\_d;  reg count;  initial  begin  clk\_d = 0;  count = 0;  end  always @(posedge clk)  begin  if (count == div\_value)  count <= 0;  else  count <= count + 1;  end  always @(posedge clk)  begin  if (count == div\_value)  clk\_d <= ~ clk\_d;  end  endmodule |
| --- |

Modify the testbench provided in Figure 10. 9(b) to verify the functionality of your designed clk\_div module.

*Attach your testbench here.*

| `timescale 1ns / 1ps  module Simulation();  reg clk;  wire clk\_d;  clk\_div c1(clk, clk\_d);  initial  begin  clk = 1'b0;    end    always #5 clk <= ~clk;  endmodule |
| --- |

*Attach screenshot of waveform here*

|  |
| --- |

## Exercise 1

*Provide appropriately commented code for your top design module*

| `timescale 1ns / 1ps  module TopLevelModuleLab10(  input clk,  output [9:0] h\_count,  output [9:0] v\_count  );  wire trig\_v;  wire clk\_d;  clk\_div M1(clk, clk\_d);  H\_counter M2(clk\_d, h\_count, trig\_v);  V\_Counter M3(clk\_d, trig\_v, v\_count);  endmodule  module clk\_div(  input clk,  output clk\_d  );  parameter div\_value = 1;  reg clk\_d;  reg count;  initial  begin  clk\_d = 0;  count = 0;  end  always @(posedge clk)  begin  if (count == div\_value)  count <= 0;  else  count <= count + 1;  end  always @(posedge clk)  begin  if (count == div\_value)  clk\_d <= ~ clk\_d;  end  endmodule  module V\_Counter(  input clk,  input enable\_v,  output [9:0] v\_count  );    reg [9:0] v\_count;  //reg trig\_v;  initial v\_count = 0;  always @ (posedge clk)  begin  if (v\_count <= 524)  begin @ (posedge enable\_v)  //begin  v\_count <= v\_count + 1;  end  else  //begin  begin @ (posedge enable\_v)  v\_count <= 0;  end  end  endmodule  module H\_counter(  input clk,  output [9:0] h\_count,  output trig\_v  );  reg [9:0] h\_count;  reg trig\_v;  initial trig\_v = 0;  initial h\_count = 0;  always @ (posedge clk)  begin  if (h\_count < 799)  begin  h\_count <= h\_count + 1;  trig\_v <= 0;  end  else  begin  trig\_v <= 1;  h\_count <= 0;  end  end  endmodule |
| --- |

Modify the testbench provided in Figure 10. 9(b) to verify the functionality of your designed top module.

*Attach your testbench here.*

| `timescale 1ns / 1ps  module Simulation();  reg clk;  wire [9:0] h\_count;  wire [9:0] v\_count;  TopLevelModuleLab10 c1(clk, h\_count,v\_count);  initial  begin  clk = 1'b0;  end    always #5 clk <= ~clk;  endmodule |
| --- |

*Attach screenshot of waveform here*

| when more zoomed out: |
| --- |

## Exercise 2

*Provide appropriately commented code for your top-level design module*

| **DESIGN**  `timescale 1ns / 1ps  module Simultaneous\_ID\_Displayer(  input [3:0] A,  input [3:0] B,  input [3:0] C,  input [3:0] D,  input clk,  input En,  output [6:0] Y,  output EnA,  output EnB,  output EnC,  output EnD  );  wire [3:0] O ;  wire [1:0] S;  //wire clk\_d;  clk\_Divider Divider(clk, clk\_d);  Counter Count(clk\_d, S);  mux\_4\_1 Mux(A,B,C,D,S,O);  demux\_1\_4 DeMux(En,S,EnA,EnB,EnC,EnD);  SevenSegment SS (O,Y);  endmodule  module mux\_4\_1(  input [3:0] A,  input [3:0] B,  input [3:0] C,  input [3:0] D,  input [1:0] S,  output [3:0] O  );  assign O[3:0] = S[1] ? (S[0] ? D : C) : ( S[0] ? B : A );  endmodule  module demux\_1\_4(  input En,  input [1:0] S,  output EnA,  output EnB,  output EnC,  output EnD  );  assign EnA = En ? (1'b1) : (S[1] ? (S[0] ? 1'b1 : 1'b1) : ( S[0] ? 1'b1 : 1'b0 ));  assign EnB = En ? (1'b1) : (S[1] ? (S[0] ? 1'b1 : 1'b1) : ( S[0] ? 1'b0 : 1'b1 ));  assign EnC = En ? (1'b1) : (S[1] ? (S[0] ? 1'b1 : 1'b0) : ( S[0] ? 1'b1 : 1'b1 ));  assign EnD = En ? (1'b1) : (S[1] ? (S[0] ? 1'b0 : 1'b1) : ( S[0] ? 1'b1 : 1'b1 ));  endmodule  module SevenSegment(  input [3:0] D,  output [6:0] S);    assign S[0] = ((~D[3])&(~D[2])&(~D[1])&D[0]) | ((~D[3])&(D[2])&(~D[1])&(~D[0])) | ((D[3])&(~D[2])&(D[1])&D[0]) | ((D[3])&(D[2])&(D[1])&D[0]);  assign S[1] = ((D[2])&(D[1])&(~D[0])) | ((D[3])&(D[1])&(D[0])) | ((D[3])&(D[2])&(~D[0])) | ((~D[3])&(D[2])&(~D[1])&D[0]);  assign S[2] = ((D[3])&(D[2])&(~D[0])) | ((D[3])&(D[2])&(D[1])) | ((~D[3])&(~D[2])&(D[1])&(~D[0]));  assign S[3] = ((D[2])&(D[1])&D[0]) | ((~D[3])&(~D[2])&(~D[1])&(D[0])) | ((~D[3])&(D[2])&(~D[1])&(~D[0])) | ((D[3])&(~D[2])&(D[1])&(~D[0]));  assign S[4] = ((~D[3])&D[0]) | ((~D[2])&(~D[1])&(D[0])) | ((~D[3])&(D[2])&(~D[1]));  assign S[5] = ((~D[3])&(~D[2])&D[0]) | ((~D[3])&(~D[2])&(D[1])) | ((~D[3])&(D[1])&D[0]) | ((D[3])&(D[2])&(~D[1])&D[0]);  assign S[6] = ((~D[3])&(~D[2])&(~D[1])) | ((~D[3])&(D[2])&(D[1])&(D[0])) | ((D[3])&(D[2])&(~D[1])&(~D[0]));  endmodule  module Counter(  input clk,  output [1:0] count  );  reg [1:0] count;  initial count = 0;  always @ (posedge clk)  begin  if (count < 3)  begin  count <= count + 1;  end  else  begin  count <= 0;  end  end  endmodule  module clk\_Divider(  input clk,  output clk\_d  );  parameter div\_value = 10000;  reg clk\_d;  reg [13:0]count;  initial  begin  clk\_d = 0;  count = 0;  end  always @(posedge clk)  begin  if (count == div\_value)  count <= 0;  else  count <= count + 1;  end  always @(posedge clk)  begin  if (count == div\_value)  clk\_d <= ~ clk\_d;  end  endmodule  **TESTBENCH**  `timescale 1ns / 1ps  module topLevelModuleSimulation();  reg [3:0] A;  reg [3:0] B;  reg [3:0] C;  reg [3:0] D;  reg clk;  reg En;  wire [6:0] Y;  wire EnA;  wire EnB;  wire EnC;  wire EnD;  Simultaneous\_ID\_Displayer module\_u\_test (A,B,C,D,clk,En,Y,EnA,EnB,EnC,EnD);  //clk\_div module\_u\_test (A,B,C,D,clk,En,Y,EnA,EnB,EnC,EnD);  initial  begin  En = 1'b0;  A = 3'b111;  B = 3'b010;  C = 3'b000;  D = 3'b001;  clk = 1'b0;  end    //initial clk = 1'b0;  always #5 clk <= ~clk;  endmodule  **SIMULATION**    **CONSTRAINT FILE**  set\_property IOSTANDARD LVCMOS33 [get\_ports {A[3]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {A[2]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {A[1]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {A[0]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {B[3]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {B[2]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {B[1]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {B[0]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {C[3]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {C[2]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {C[1]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {C[0]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {D[3]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {D[2]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {D[1]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {D[0]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {Y[6]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {Y[5]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {Y[4]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {Y[3]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {Y[2]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {Y[1]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports {Y[0]}]  set\_property IOSTANDARD LVCMOS33 [get\_ports clk]  set\_property IOSTANDARD LVCMOS33 [get\_ports En]  set\_property IOSTANDARD LVCMOS33 [get\_ports EnA]  set\_property IOSTANDARD LVCMOS33 [get\_ports EnB]  set\_property IOSTANDARD LVCMOS33 [get\_ports EnC]  set\_property IOSTANDARD LVCMOS33 [get\_ports EnD]  set\_property PACKAGE\_PIN W5 [get\_ports clk]  set\_property PACKAGE\_PIN R2 [get\_ports {A[3]}]  set\_property PACKAGE\_PIN T1 [get\_ports {A[2]}]  set\_property PACKAGE\_PIN U1 [get\_ports {A[1]}]  set\_property PACKAGE\_PIN W2 [get\_ports {A[0]}]  set\_property PACKAGE\_PIN R3 [get\_ports {B[3]}]  set\_property PACKAGE\_PIN T2 [get\_ports {B[2]}]  set\_property PACKAGE\_PIN T3 [get\_ports {B[1]}]  set\_property PACKAGE\_PIN V2 [get\_ports {B[0]}]  set\_property PACKAGE\_PIN W13 [get\_ports {C[3]}]  set\_property PACKAGE\_PIN W14 [get\_ports {C[2]}]  set\_property PACKAGE\_PIN V15 [get\_ports {C[1]}]  set\_property PACKAGE\_PIN W15 [get\_ports {C[0]}]  set\_property PACKAGE\_PIN W17 [get\_ports {D[3]}]  set\_property PACKAGE\_PIN W16 [get\_ports {D[2]}]  set\_property PACKAGE\_PIN V16 [get\_ports {D[1]}]  set\_property PACKAGE\_PIN V17 [get\_ports {D[0]}]  set\_property PACKAGE\_PIN W7 [get\_ports {Y[0]}]  set\_property PACKAGE\_PIN W6 [get\_ports {Y[1]}]  set\_property PACKAGE\_PIN U8 [get\_ports {Y[2]}]  set\_property PACKAGE\_PIN V8 [get\_ports {Y[3]}]  set\_property PACKAGE\_PIN U5 [get\_ports {Y[4]}]  set\_property PACKAGE\_PIN V5 [get\_ports {Y[5]}]  set\_property PACKAGE\_PIN U7 [get\_ports {Y[6]}]  set\_property PACKAGE\_PIN U2 [get\_ports EnA]  set\_property PACKAGE\_PIN U4 [get\_ports EnB]  set\_property PACKAGE\_PIN V4 [get\_ports EnC]  set\_property PACKAGE\_PIN W4 [get\_ports EnD]  set\_property PACKAGE\_PIN U18 [get\_ports En] |
| --- |

*Attach picture of FPGA here*

|  |
| --- |

## 

## Assessment Rubrics

**Marks Distribution:**

|  |  | **LR2**  **Code** | **LR5**  **Results** | **LR7**  **Viva** |
| --- | --- | --- | --- | --- |
| **In-lab** | **Task a** | 10 points | 5 points | 10 points |
| **Task b** | 10 points | 10 points |
| **Task c** | - | 5 points |
| **Exercise 1** |  | 20 points | 10 points |
| **Exercise 2** |  | 10 points | 10 points |  |
| **Total** |  |  |  |  |

**Marks Obtained:**

|  |  | **LR2**  **Code** | **LR5**  **Results** | **LR7**  **Viva** |
| --- | --- | --- | --- | --- |
| **In-lab** | **Task a** |  |  |  |
| **Task b** |  |  |
| **Task c** | - |  |
| **Exercise 1** |  |  |  |
| **Exercise 2** |  |  |  |  |
| **Obtained** |  |  |  |  |