UNIVERSITY OF NEVADA LAS VEGAS. DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING LABORATORIES.

Class:	СРІ	E300L	Semester:	Fall 2021							
Points		Document author:									
		Author's email:	batuj1@unlv.r	nevada.edu							
		Document topic: Postlab 8									
Instructor's	s com	iments:									

1. Introduction / Theory of Operation

In this lab, I will continue my learning about the general datapath's implementation for a given algorithm. I will also learn more about timing simulations by setting up timing constraints and timing information as well as modeling memories.

- a. In regards to time quest analysis **slack** is the range of where the timing requirement is met or not. When the timing requirement is met, this is called positive slack; hence, the negative slack is when the timing requirement is not met. Realistic slack values are met when all of the clocks in the design are constrained with real values.
- b. It is important to consider **timing analysis** in a digital circuit design because incorrect format and outputs may result in different timing cycles. In addition, the timing analysis allows the user to easily make a circuit design more feasible and efficient on how fast the outputs can be processed through the inputs. Timing analysis, specifically, will determine the amount of time it takes between the inputs and outputs to be initialized. By having the timing requirement not met and negative slack, the circuit will not be maximized and efficient as there may be a slight delay between outputs after the clock is on the positive edge.

2. Prelab

https://docs.google.com/document/d/1PKcaACI4mB_VmDI2OBT4IPvLjs1yjqKz/edit?usp=shari ng&ouid=102808507017671072128&rtpof=true&sd=true This is the link to my prelab8.

3. Results of Experiments

EXPERIMENT 1

```
`timescale 1ns / 100ps
module GCD (startGDP, rstGDP, clkGDP, n1GDP, n2GDP, complete, GCDoutDP);
      input startGDP, clkGDP, rstGDP;
      input [7:0] n1GDP, n2GDP;
      output complete;
      output [7:0] GCDoutDP;
      wire WE, RAE, RBE, OE, IE, neq, gt;
      wire [1:0] WA, RAA, RBA, SH;
      wire [2:0] ALU;
      CU control (~startGDP, clkGDP, ~rstGDP, neq, gt, IE, WE, WA, RAE, RAA, RBE,
RBA, ALU, SH, OE);
      DP datapath (n1GDP, n2GDP, clkGDP, IE, WE, WA, RAE, RAA, RBE, RBA, ALU,
SH, OE, neq, gt, GCDoutDP);
      assign complete = OE;
endmodule
module CU (start, clk, rst, neq, gt, IE, WE, WA, RAE, RAA, RBE, RBA, ALU, SH, OE);
      input start, clk, rst;
      output IE, WE, RAE, RBE, OE;
      output [1:0] WA, RAA, RBA, SH;
      output [2:0] ALU;
      input wire neq, gt;
      reg [2:0] state;
      reg [2:0] nextstate;
      parameter S0 = 3'b000;
      parameter S1 = 3'b001;
      parameter S2 = 3'b010;
      parameter S3 = 3'b011;
      parameter S4 = 3'b100;
      parameter S5 = 3'b101;
      parameter S6 = 3'b110;
      initial
             state = S0;
```

```
// State register
 always @ (posedge clk)
 begin
         state <= nextstate;</pre>
 end
 // Next State Logic
 always @ (*)
         case (state)
S0: if (start)
                                           // output a = 0
                                   nextstate = S1;
                           else
                                   nextstate = S0:
                  S1: nextstate = S2;
                                                            // input a
 S2: nextstate = S3;
                                           // input b
                                           // a > b
 S3: if (neq && gt)
                                   nextstate = S4;
         else if (neq && !gt)
                                   // b < a
                                   nextstate = S5;
                          else
                                                            // neither
                                   nextstate = S6;
                  S4: nextstate = S3;
                  S5: nextstate = S3;
                  S6: if (rst)
                                   nextstate = S0;
                           else
                                   nextstate = S6;
                  default: nextstate = S0;
         endcase
 // Output Logic
 assign IE = (state == S1) \parallel (state == S2);
 assign WE = (state == S1) \parallel (state == S2) \parallel (state == S4) \parallel (state == S5);
 assign WA[1] = (state == S2) \parallel (state == S5);
 assign WA[0] = (state == S1) \parallel (state == S4);
 assign RAE = (state == S3) \parallel (state == S4) \parallel (state == S5) \parallel (state == S6);
 assign RAA[1] = state == S5;
 assign RAA[0] = (state == S3) \parallel (state == S4) \parallel (state == S6);
 assign RBE = (state == S3) \parallel (state == S4) \parallel (state == S5);
 assign RBA[1] = (state == S3) \parallel (state == S4);
 assign RBA[0] = state == S5;
```

```
assign ALU[2] = (state == S3) \parallel (state == S4) \parallel (state == S5);
       assign ALU[1] = 0;
       assign ALU[0] = (state == S3) \parallel (state == S4) \parallel (state == S5);
       assign SH[1] = 0;
       assign SH[0] = 0;
       assign OE = state == S6;
endmodule
//Data Path
module DP (n1In, n2In, clk, IE, WE, WA, RAE, RAA, RBE, RBA, ALU, SH, OE, neg, gt,
out);
       input clk, IE, WE, RAE, RBE, OE;
       input [1:0] WA, RAA, RBA, SH;
       input [2:0] ALU;
       input [7:0] n1In, n2In;
       output neq, gt;
       output wire [7:0] out;
       reg [7:0] rfIn1, rfIn2;
       wire [7:0] rfA, rfB, aluOut, shOut, n1, n2;
       initial begin
               rfIn1 = 0;
       rfIn2 = 0:
  end
       always (a) (*)
   begin
               rfIn1 = n1;
     rfIn2 = n2;
   end
       gdpMux1 mux1 (shOut, n1In, IE, n1);
       gdpMux2 mux2 (shOut, n2In, IE, n2);
       gdpRF RF (clk, WE, RAE, RBE, RAA, RBA, WA, rfIn1, rfIn2, rfA, rfB);
       gdpALU theALU (rfA, rfB, ALU, aluOut);
       gdpShift SHIFT (aluOut, SH, shOut);
       gdpBuffer buff (shOut, OE, out);
       assign neq = (|aluOut[7:0]);
       assign gt = (\sim aluOut[7]) \&\& (|aluOut[6:0]);
endmodule
```

```
// 2-to-1 mux
module gdpMux1 (a, b, sel, out);
       input [7:0] a, b;
       input sel;
       output reg [7:0] out;
       always @(*) begin
              if(sel == 0)
                      out = a;
          else
                      out = b;
       end
endmodule
// 2-to-1 mux
module gdpMux2 (a, b, sel, out);
       input [7:0] a, b;
       input sel;
       output reg [7:0] out;
       always @(*) begin
     if(sel == 0)
                      out = a;
              else
                      out = b;
       end
endmodule
// register file
module gdpRF(clk, WE, RAE, RBE, RAA, RBA, WA, inD1, inD2, ReadA, ReadB);
       input clk, WE, RAE, RBE;
       input [1:0] RAA, RBA, WA;
  input [7:0] inD1, inD2;
       output [7:0] ReadA, ReadB;
  reg [7:0] regF1 [0:3];
  reg [7:0] regF2 [0:3];
       // Write when WE asserted
       always @(posedge clk)
   begin
              if (WE == 1)
                      begin
                             regF1[WA] \le inD1;
```

```
regF2[WA] \le inD2;
                       end
   end
       //reading to Port A and B, combinational
 assign ReadA = (RAE)? regF1 [RAA]:0;
 assign ReadB = (RBE)? regF2 [RBA]:0;
endmodule
// arithmetic logic unit
module gdpALU (a,b,sel, out);
  input [7:0] a,b;
  input [2:0] sel;
  output reg [7:0] out;
  always (a) (*)
  begin
     case (sel)
       3'b000: out = a;
       3'b001: out = a & b;
       3'b010: out = a | b;
       3'b011: out = !a;
       3'b100: out = a + b;
       3'b101: out = a - b;
       3'b110: out = a + 1;
       3'b111: out = a - 1;
     endcase
  end
endmodule
// Shifter
module gdpShift (a,sh,out);
  input [7:0] a;
  input [1:0] sh;
  output reg [7:0] out;
  always (a) (*)
  begin
     case(sh)
       3'b00: out = a;
       3'b01: out = a \ll 1;
       3'b10: out = a >> 1;
       3'b11: out = \{ a[6], a[5], a[4], a[3], a[2], a[1], a[0], a[7] \} ;
     endcase
```

```
end
endmodule
// Buffer
module gdpBuffer (a, buff, out);
    input [7:0] a;
    input buff;
    output reg [7:0] out;
    always @(*)
    if(buff == 1)
        out = a;
    else
        out = 8'bzzzz_zzzz;
endmodule
```

This is the general datapath for the GCD algorithm.

```
`timescale 1ns / 100ps
module GDPTB;
       reg startTB, rstTB, clkTB;
       reg [7:0] n1TB, n2TB;
       wire displayGCD;
       wire [7:0] GCDTest;
       // Test Vectors for desired output
       reg [7:0] inTest1 [1:5];
       reg [7:0] inTest2 [1:5];
       reg [7:0] outExpect [1:5];
       integer i;
       initial
              begin
                     // initialize inputs to test and their expected outputs
                   = 8'b00000000;
       inTest1[1]
       inTest2[1] = 8'b0000000;
              outExpect[1] = 8'b0000000;
       inTest1[2]
                   = 8'b00010110;
                   = 8'b00010110;
       inTest2[2]
```

```
outExpect[2] = 8'b00010110;
       inTest1[3] = 8'b00000010;
       inTest2[3] = 8'b00000100;
       outExpect[3] = 8'b00000010;
       inTest1[4] = 8'b10101010;
       inTest2[4] = 8'b01010101;
       outExpect[4] = 8'b01010101;
       inTest1[5] = 8'b11111111;
       inTest2[5] = 8'b00001010;
       outExpect[5] = 8'b00000101;
                      startTB = 1;
                      rstTB = 1;
                      clkTB = 0;
                      // generate clock
                      forever
                             #2 \text{ clkTB} = \sim \text{clkTB};
              end
       always
              begin
                      for (i = 1; i \le 5; i = i + 1) // test for the 5 inputs
                             begin
                                     n1TB \le inTest1[i];
              n2TB \le inTest2[i];
                                     startTB = 0;
                                     #40 \text{ rstTB} = 1;
                                     // Waiting for code to finish then displayResults when
done
                                     while (displayGCD != 1)
                                            #5 begin end
           $write ("InputA: %b InputB: %b Expected: n =%d Calculated: n =%d: ",
                 n1TB, n2TB, outExpect[i], GCDTest);
                                     if (GCDTest == outExpect[i])
                                            $display("Correct!");
                                     else
                                            $display("Incorrect!");
                                     rstTB = 0;
```

startTB = 1;	
end	
\$stop;	
end	
<pre>//Instantiate GDP GCD mainGDP (startTB, rstTB, clkTB, n1TB, n2TB, displayGCD, GCDTest); endmodule</pre>	

This is the testbench for the general datapath for the GCD algorithm.

			C1:48			Sim:14950
н		<u>▼</u> 313 ▼ 55 ▼	REF			(14902)
Name		Value	u ndara.	1000 12000 13000 14000 5000 6000 17000 18000 19000 10000 11000	0 1 1 1	12000 13000 14000
- Group1						
- 🛙 cikT	тв	St0				
🖭 🛙 i[31:	:0]	1	1 (2)	3	4	5
- 🛙 star	rtTB	St0				
- 🛙 rstTi	гв	St1				
⊕ 🛛 n1T	TB[7:0]	0	0 22	2 1	70	255
. ∎ n2T	TB[7:0]	0	0 22	4	85	10
🖶 🛙 inTe	est1[1:5][7:0]	{ 0, 22, 2, 170, }		{ 0, 22, 2, 170, }		
⊕- 🛙 inTe	est2[1:5][7:0]	{ 0, 22, 4, 85, }		{ 0, 22, 4, 85, }		
🕒 🛙 outB	Expect[1:5][7:0]	{ 0, 22, 2, 85, }		(0, 22, 2, 85,)		
	:DTest[7:0]	z	-(□)-↓-↓		€ _()	
	playGCD	Sto			ΙΓ	
New Grou	up					
1						

These are my VCS waveforms for the general datapath for the GCD algorithm. The clock is a yellow bar because there are several instantiations of clock within the full waveform view. The yellow line for the GCDTest represents "Z" as "Z" would only be outputted if there is no GCD.

Type: 라그 도 Severity: 라그 도 Code: All 도 이 이 이 이 이 이 이 이 이 이 이 이 이 이 이 이 이 이	
Chronologic VCS simulator copyright 1991-2017	
Contains Synopsys proprietary information.	
Compiler version N-2017.12-SP2-14; Runtime version N-2017.12-SP2-14; Oct 20 22:36 2021	-
VCD+ Writer N-2017.12-SP2-14 Copyright (c) 1991-2017 by Synopsys Inc.	
The file '/home/batuj1/Fall2021/lab8/inter.vpd' was opened successfully.	
InputA: 00000000 InputB: 00000000 Expected: n = 0 Calculated: n = 0:	
Correct!	
InputA: 00010110 InputB: 00010110 Expected: n = 22 Calculated: n = 22:	
Correct!	
InputA: 00000010 InputB: 00000100 Expected: n = 2 Calculated: n = 2:	
Correct!	
InputA: 10101010 InputB: 01010101 Expected: n = 85 Calculated: n = 85:	
Correct!	
InputA: 11111111 InputB: 00001010 Expected: n = 5 Calculated: n = 5:	
L'orrect!	

This is my VCS console for the general datapath for the GCD algorithm.

<u>https://drive.google.com/file/d/1CBf2H-Cdq6xv_JaxTidzjIVeSvzgOlaM/view?usp=sharing</u> This is the link to my video delivery of the GCD general datapath implemented onto the DE2 board.

EXPERIMENT 2



This is the timing analysis of Experiment 1 before setting up the constraints.

																			Jearen artera con	<u></u>
Report # #	Setup: dkGDP																			•
TimeQuest Timing Analyzer Summary	Command Infe	Commence	of Daths																	
🖉 Setup: clkGDP	Command Info	5 Summary	oi Fauls																	
	Slack	From N	ode		To Node	Launch Clo	ck Latch Clock	Relationship	Clock Skew	Data Delay										
	1 3.511	CU:control[st	ate.S4	DP:datapa	th gdpRF:RF re	gF1~12 dkGDP	dkGDP	10.000	0.000	6.525										
	2 3.516	CU:control st	ate.S6	DP:datapa	th gdpRF:RF re	egF1~12 dkGDP	dkGDP	10.000	0.000	6.520										
	3 3.528	CU:control st	ate.53	DP:datapa	th gdpRF:RF re	egF1~12 dkGDP	dkGDP	10.000	0.000	6.508										
	4 3.556	CU:controljst	ate.53	DP:datapa	th Igopke: Refe	SGP1~12 OKGUP	dKGDP	10.000	0.000	6.480										
	5 3.621	Clucontrolist	ate 64	DP:datapa DP:datapa	th Jodon Experies	SGF1~13 OKGDP	dkGDP	10.000	0.000	6.410										
	7 3.622	Clicontrolist	ate S6	DPidatapa	th ladeRE-RE in	oF1+12 dkGDP	dkopp	10.000	0.000	6.410										
Tasks # # *	8 3.627	Clicontrollet	ate S6	DP:datana	th lodoDE-DE la	eF1w12 dkGDP	dkopp	10.000	0.000	6 409										
🖋 🎯 Open Project 🔨	9 3.629	CU:controlist	ate.S4	DP:datana	th lodoRE:RE Ire	apE1v12 dkGDP	dkGDP	10.000	0.000	6.407										
🗁 Netlist Setup	10 3.638	CU:controllst	ate.S4	DP:datapa	th lodoRE:RE In	nE1~13 dkGDP	dkGDP	10.000	0.000	6.398										
🗸 🕨 Create Timing Netlist																				
Read SDC File																				
🗸 🕨 Update Timing Netlist																				
Reset Design	Path #1: Setu	ip slack is 3.5	11								Path #1: Setup slack	c is 3.511								
Set Operating Conditions	Path Summary	/ Statistics	Data	Path Wa	veform						Path Summary Stat	atistics	Data Path W	aveform						
Preports Reports	Data Arrival	Path																		
V 🗁 Slack	To	otal Ir	nor	RF Typ	e Fanout	Location	Ele	ement		^										
Report Setup Summary	1 0.00	0 0.00	0				launch edge time													
Report Hold Summary	2 ¥ 2.66	9 2.66	9				clock path				Launch Clock Launch	h								
Report Recovery Summary	1 (0.0 0.0	00				source latency				Edunch Clock	1								
Report Removal Summary	2 (0.00 0.0	00		1	PIN_N2	dkGDP								10.0 ns					
Report Minimum Pulse Width Summa	3 (0.999 0.9	99 RR	CELL	1	IOC_X0_Y18_N0	clkGDP combout				Setup Relationship								-	
Report Max Skew Summary	4	1.117 0.1	18 RR	IC	1	CLKCTRL_G2	clkGDP~clkctrl incl	<[0]				_								
Y 🗁 Datasheet	5	1.117 0.0	00 RR	CELL	55	CLKCTRL_G2	clkGDP~clkctrl out	dk			Latch Clock							Latch	n	
Report Fmax Summary	6	2.132 1.0	15 RR	IC	1	LCFF_X29_Y9_N11	control state.S4 d	k												
Report Datasheet	7	2.669 0.5	37 RR	CELL	1	LCFF_X29_Y9_N11	CU:control state.S	4			Data Arrival							X		
Device Specific	3 \$ 9.19	4 6.52	5	-			data path					-								
Report TCCS	1	2.919 0.2	50	uico	1	LCFF_X29_Y9_N11	CU:control state.5				Clock Delau		2.669 ns							
Report RSKM	2	2.919 0.0	CO 00	LELL IC	14	LCFF_X29_19_N11	control/state.54/n	egout tee						1						
Report DDR	,	5.461 0.5	02 KA	10	1	FCCCMP_X30_13_1430	Controllicerefollog	uaa	_	•	Data Dalau					6.525 ns				
Report Metastability	Data Require	d Path									Data Delay									
Diagnostic	То	tal Ir	nar	RF Typ	e Fanout	Location	Ele	ment											3 511 m	
Report Clocks	1 10.0	00 10.0	00				latch edge time				Slack							+	01012 110	
Report Clock Transfers	2 ¥ 12.6	69 2.66	9				clock path											_		
Report Unconstrained Paths	1	10.000 0.0	00				source latency				Data Required									Х
Report SDC	2	10.000 0.0	00		1	PIN_N2	dkGDP											-		
Report Ignored Constraints	3	10.999 0.9	99 RR	CELL	1	IOC_X0_Y18_N0	clkGDP combout				Clock Delay								2.669 ns	s
Check Timing	4	11.117 0.1	18 RR	IC	1	CLKCTRL_G2	dkGDP~dkctrl ind	(0]												1
Report Partitions	5	11.117 0.0	00 RR	CELL	55	CLKCTRL_G2	ckGDP~ckctrl out	dk												-0.036
Y 🗁 Custom Reports	6	12.132 1.0	15 RR	IC	1	LCFF_X29_Y9_N5	datapath RF regF	1~12 dk			ursu									1
Report Timing	7	12.669 0.5	37 RR	CELL	1	LCFF_X29_Y9_N5	DP:datapath gdpR	F:RF regF1~12			Tine (ns)	0.0	1.25 2	.5 3.1	75 5.0	6.25	7.5	8.75 1	10.0 11.25	12.5
Percet Minimum Dulce Width	3 12.7	0.03	6	UTSU	1	LCFF X29 Y9 N5	DP:datapathlodpR	E:REirenE1v12												

This is the timing analysis of Experiment 1 after setting up the constraints.

EXPERIMENT 3

```
module sngPrtRAM (WE, CLK, Inp, Addr, opd);
input WE, CLK;
input [3:0] Addr;
input [7:0] Inp;
output wire [7:0] opd;
//16x8 ram
reg [15:0] RAM [0:7];
always@(posedge CLK)
begin
if(WE == 1'b1)
RAM[Addr] <= Inp;
end
assign opd = RAM[Addr];
endmodule
```

This is my Verilog code for the single port RAM.



This is my RTL View for the single port RAM.

<u>https://drive.google.com/file/d/1BAPfnyEVVPpajUcIp9QMVUPvnrNUuBL6/view?usp=sharing</u> This is the link for my video delivery of the single port RAM implementation onto the DE2 board.

4. Answers to questions <u>Question 1:</u>

```
input RE, CLK
input Addr
output read
reg RAM
always @ (posedge CLK)
    if (RE == 1'b1)
        read <= RAM[Addr]
    else
        read <= 0</pre>
```

Question 2:

Memory initialization in Quartus II is actually created through a Memory Initialization File (.mif).

Purposes:

- Specifies initial content of a memory block
 - Specifies the initial values for each address
 - Used during project compilation or simulation
- Serves as input file for memory initialization in compiler and simulator
- Can be used in Hexadecimal file to provide memory initialization data
- Contains initial values for each address in memory
- Specifies memory depth and width values
 - Specifies data radixes in different representations (binary, hexadecimal, decimal, etc.)
- Overall purpose of initializing memory is to ensure proper range of memory is to be used within a program to prevent any overflow or underflow when coding
 - Overflow or underflow of memory will result into a program to crash as not enough memory is allocated for any functions to process

5. Conclusions & Summary

Experiment 1 and the last prelab question were the hardest parts in regards to the lab. The main issue that I kept having was using two inputs for the general datapath because we have been using only one input for almost all of the labs. In addition, I had a hard time implementing the ALUout to determine if a is equal to b. To solve these issues, I kept rewatching the lab video over and over to try to fully understand what was meant to be achieved from my problems. Eventually, I was able to figure out my Verilog code and register file implementations with the two different inputs. I am very familiar with the general data path after this lab, and I am getting a much better understanding of how timing and delay can affect the overall performance of Verilog code. In addition, this lab definitely assisted my understanding of the importance of RAM, and it's implementation in Verilog code.