Class:	**CPE300L Digital Systems Architecture and Design 1001**		Semester:	**Fall 2024**
Points		Document author:	**Narek Kalikian**	
		Author's email:	**kalikn1@unlv.nevada.edu**	
		Document topic:	**Postlab 5**	
Instructor's comments:				

1. Introduction / Theory of Operation

In this lab, we continued working with the general datapath and its components like the control unit, register file, ALU, etc. This time, we were tasked with implementing a greatest common divisor algorithm using the GDP. We also programmed a single-port behavioral RAM module and implemented these on our DE2-115 FPGA board.

2. Description of Experiments

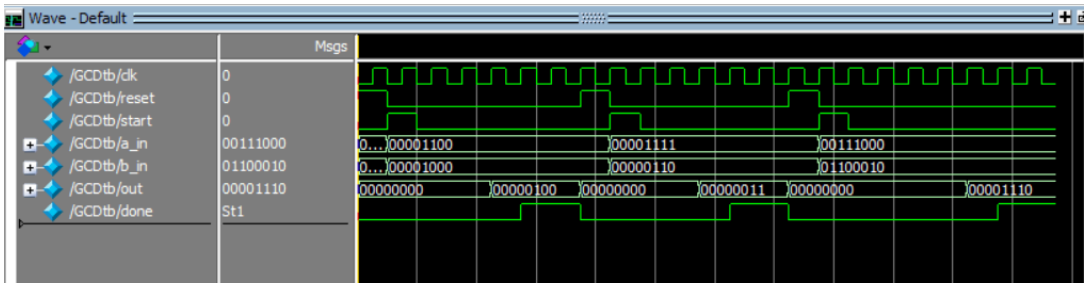
Experiment 1 - Verilog Code:

```
72  module GCD(
73      input wire clk,
74      input wire reset,
75      input wire start,
76      input wire [15:0] a_in,
77      input wire [15:0] b_in,
78      output reg [15:0] out,
79      output reg done
80  );
81
82      reg [15:0] a, b;
83      reg [15:0] temp;
84      reg [2:0] state;
85
86      parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b10;
87
88      always @(posedge clk or posedge reset) begin
89          if (reset) begin
90              a <= 0;
91              b <= 0;
92              out <= 0;
93              done <= 0;
94              state <= S0;
95          end else begin
96              case (state)
97                  S0: begin
98                      if (start) begin
99                          a <= a_in;
100                         b <= b_in;
101                         done <= 0;
102                         state <= S1;
103                     end
104                 end
105
106                 S1: begin
107                     if (b == 0) begin
108                         out <= a;
109                         state <= S2;
110                     end else begin
111                         temp = b;
112                         b = a % b;
113                         a = temp;
114                     end
115                 end
116
117                 S2: begin
118                     done <= 1;
119                 end
120             endcase
121         end
122     end
123 endmodule
```

Experiment 1 - Testbench Code:

```
module GCDtb();
    reg clk;
    reg reset;
    reg start;
    reg [15:0] a_in;
    reg [15:0] b_in;
    wire [15:0] out;
    wire done;

    gcd uut(
        .clk(clk),
        .reset(reset),
        .start(start),
        .a_in(a_in),
        .b_in(b_in),
        .out(out),
        .done(done)
    );

    always #5 clk = ~clk;

    initial begin
        clk = 0;
        reset = 1;
        start = 0;
        a_in = 0;
        b_in = 0;
        #10;

        reset = 0;
        a_in = 12;
        b_in = 8;
        start = 1;
        #10;
        start = 0;

        wait(done);
        $display("GCD of 12 and 8 is: %d", out);

        #20;
        reset = 1;
        #10;
        reset = 0;
        a_in = 15;
        b_in = 6;
        start = 1;
        #10;
        start = 0;

        wait(done);
        $display("GCD of 15 and 6 is: %d", out);

           #20;
           reset = 1;
           #10;
           reset = 0;
           a_in = 56;
           b_in = 98;
           start = 1;
           #10;
           start = 0;

           wait(done);
           $display("GCD of 56 and 98 is: %d", out);

           #20;
           $stop;
    end
endmodule
```

Experiment 1 - Waveform Simulation and Console:



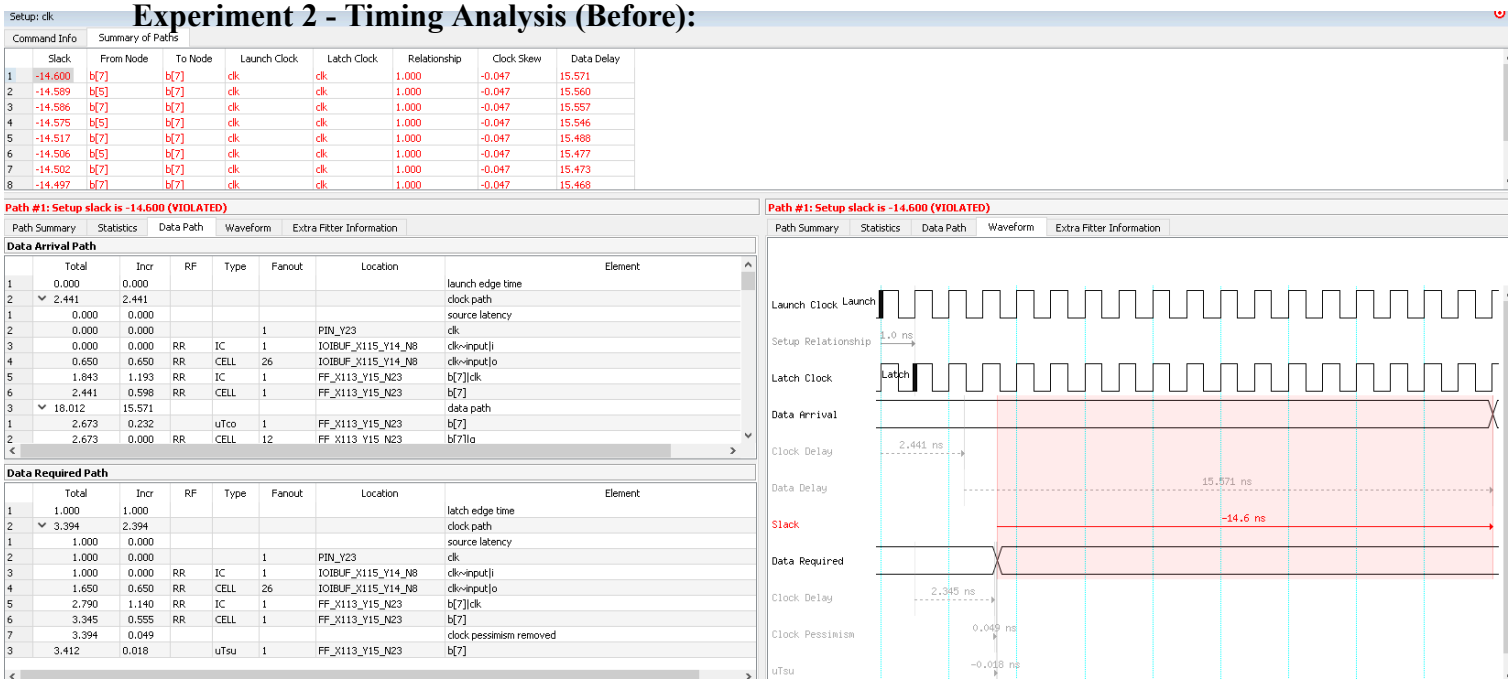

```
VSIM 27> run -all
# GCD of 12 and 8 is:   4
# GCD of 15 and 6 is:   3
# GCD of 56 and 98 is:  14
# ** Note: $stop    : C:/altera/13.0spl/GCD.v(73)
#    Time: 380 ns  Iteration: 0  Instance: /GCDtb
```

Experiment 1 - Video Delivery: Shared via Google Drive

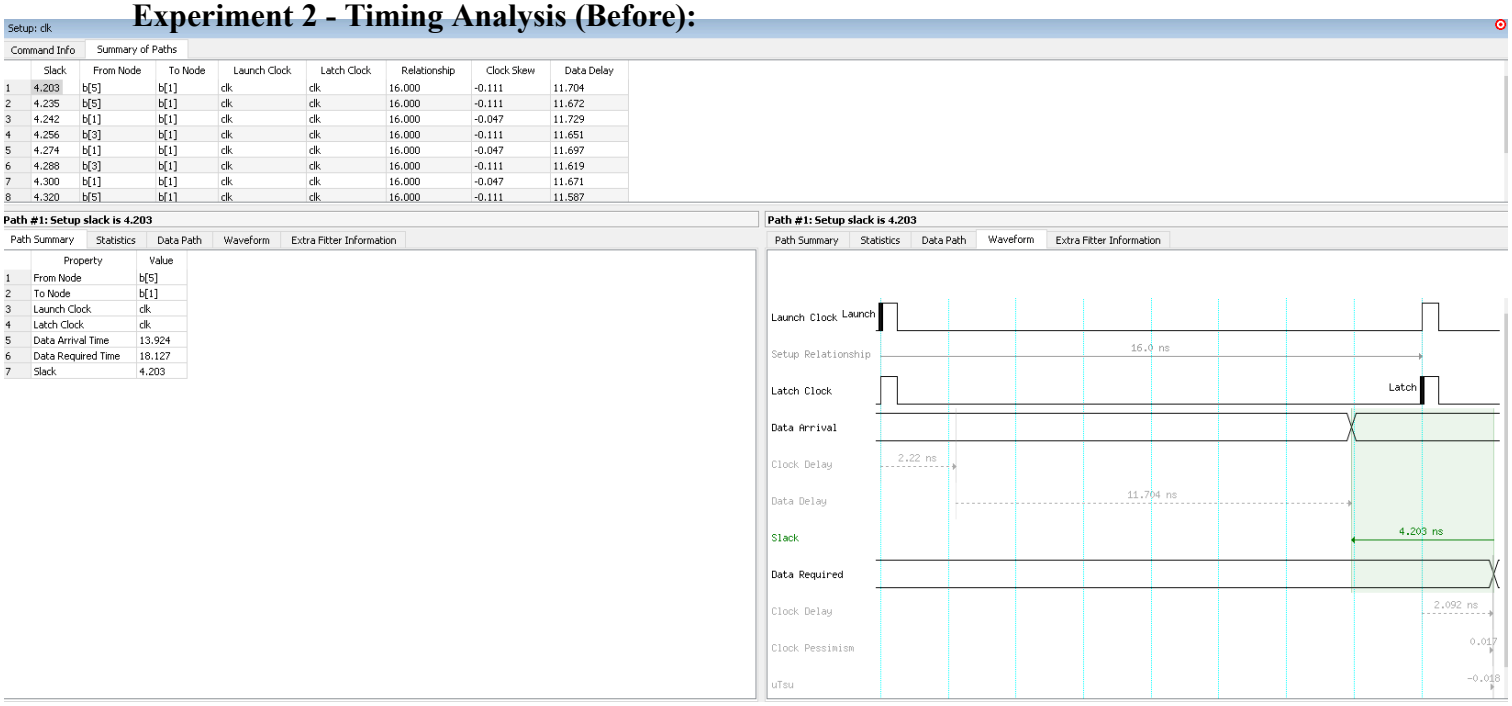
Experiment 1 - Video Explanation: a is set to 12 (001100) and b is set to 8 (001000). The resulting output, the GCD of those two input values, is shown on the seven-segment displays as 4 (000100).

We were unable to implement the GCD algorithm using the general datapath after trying numerous times, including with the TA, so we decided to create the GCD logic separately in order to complete the rest of the steps for experiment 1.

Experiment 2 - Timing Analysis (Before):



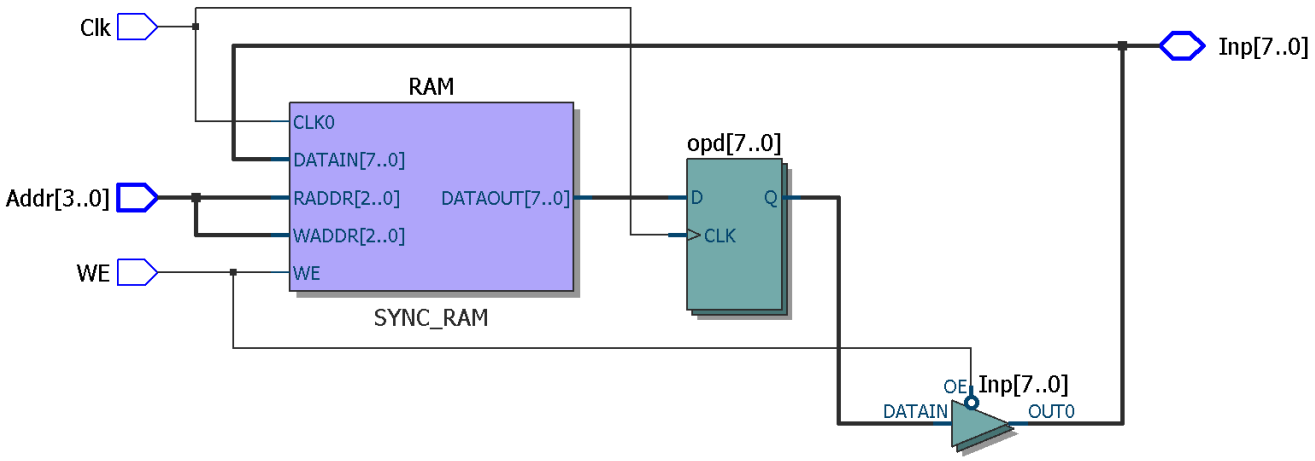
Experiment 2 - Timing Analysis (Before):



Experiment 3 - Verilog Code:

```
1   module RAM(WE, Clk, Addr, Inp);
2   input WE, Clk;
3   input [3:0] Addr; //4 address bits = depth of 16 memory locations
4   inout [7:0] Inp; //width of 8
5   reg [7:0] opd;
6   reg [15:0] RAM [0:7]; //16x8 RAM block
7
8   assign Inp = WE ? 16'hZZZZ: opd;
9   always@(posedge Clk) begin
10  if(WE == 1'b1)
11  RAM[Addr] <= Inp;
12  opd <= RAM[Addr];
13  end
14  endmodule
```

Experiment 3 - RTL View:



3. Questions and Answers

1. ROM pseudo code:

```
module ROM
(input [width] address,
output [width] data);
always @ (*) begin
case (address)
address 0: data = 1;
address 1: data = 2;
address 2: data = 3;
…
default: data = default value (0);
endcase
end
endmodule
```

2. Purposes of Memory Initialization: It's important to initialize memory in Quartus is important for setting specific predefined values within a memory block.

4. Conclusions

Clock cycles, timing analysis, and memory are key components of computer architecture. Learning these topics is important for understanding the fundamentals of how computers work.