Christopher Badolato

Designing with D-Flip flops:

Shift Register and Sequence Counter

EEE 3342 LAB 0014

12/02/2018

**Objective:**

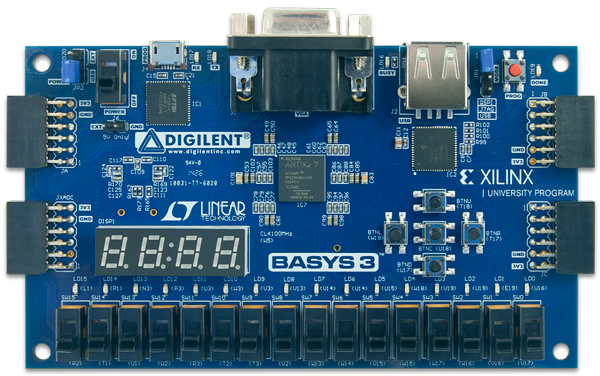
* Design a 4 bit parallel load register used as a right shift register and left shift register using sequential D flip-flips.
* As well as a sequence counter

Apparatus List:

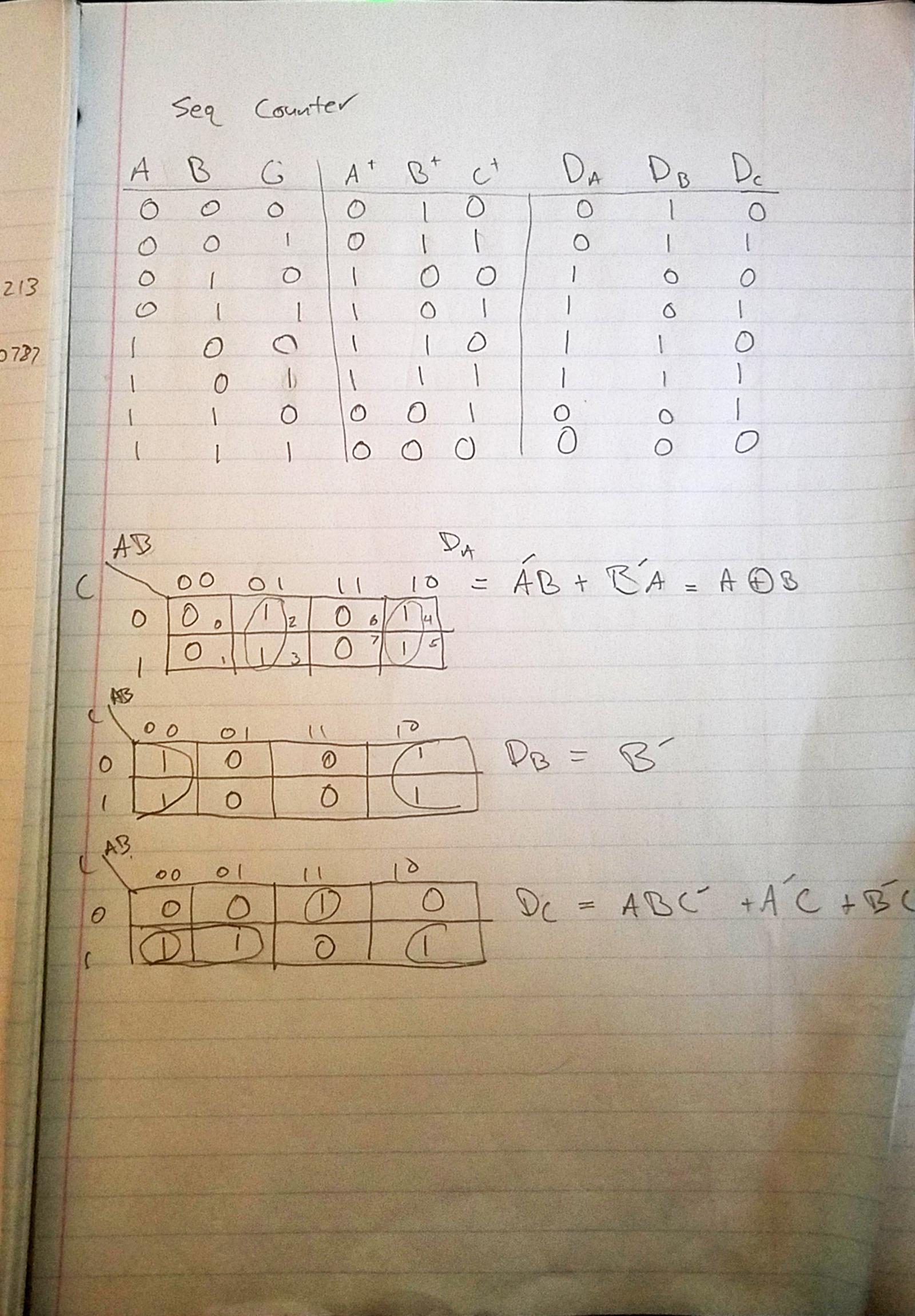
Xilinix’s FPGA VIVADO HLX edition

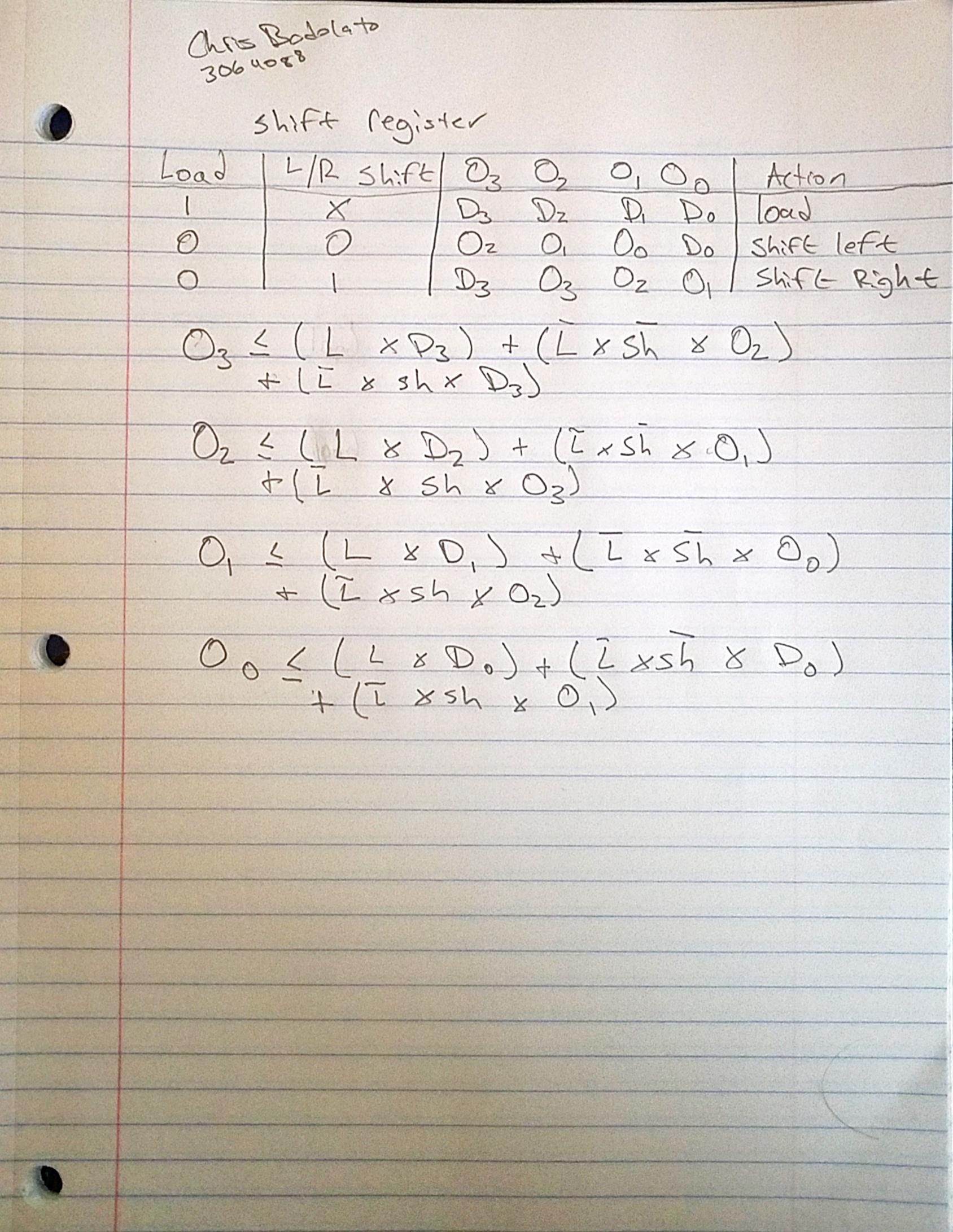
VIVDAO 2017.4

Basby Atrix-7



Procedure and/or Design Methodology:





Block Diagram:

Shift register

D

0

D

1

D

2

D

3

Clock

Load –

Shift

SHIFT

REGISTER

O

0

O

1

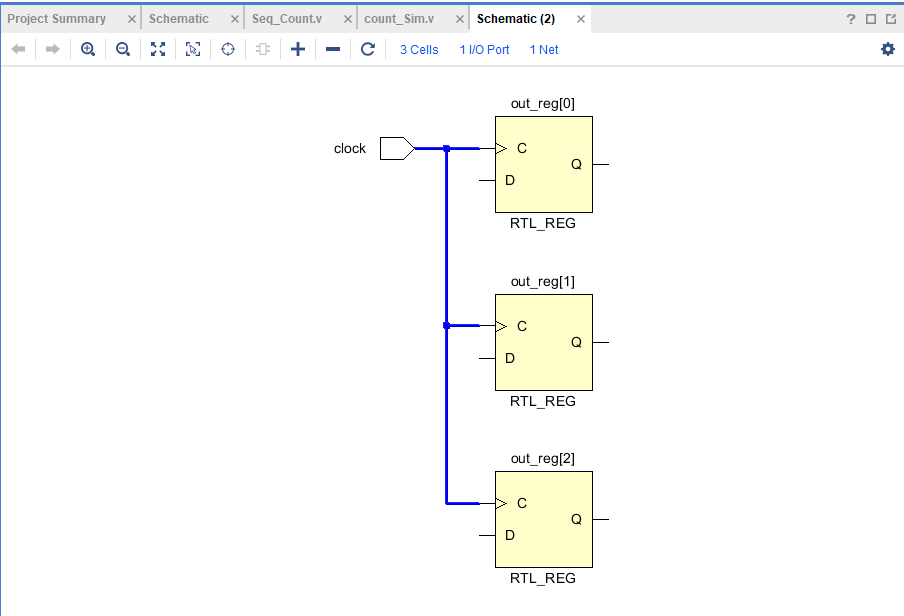
O

2

O

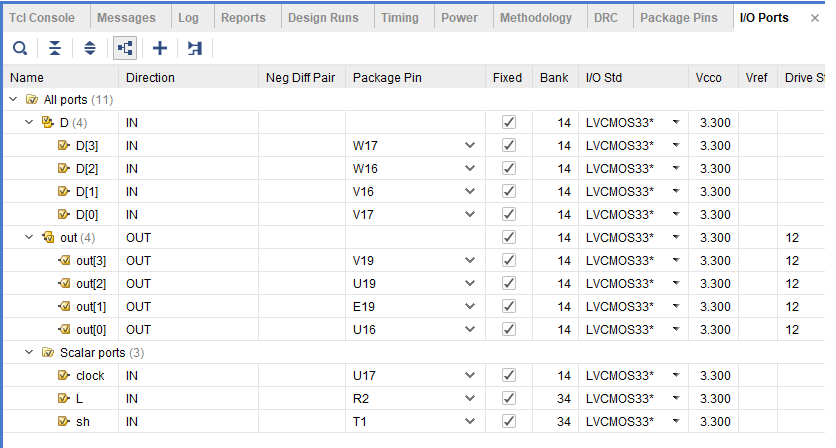
3

**Counter**

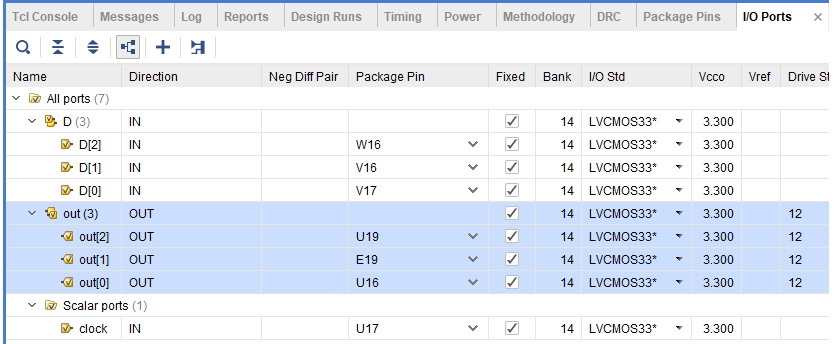


Design Specification:

Shift register



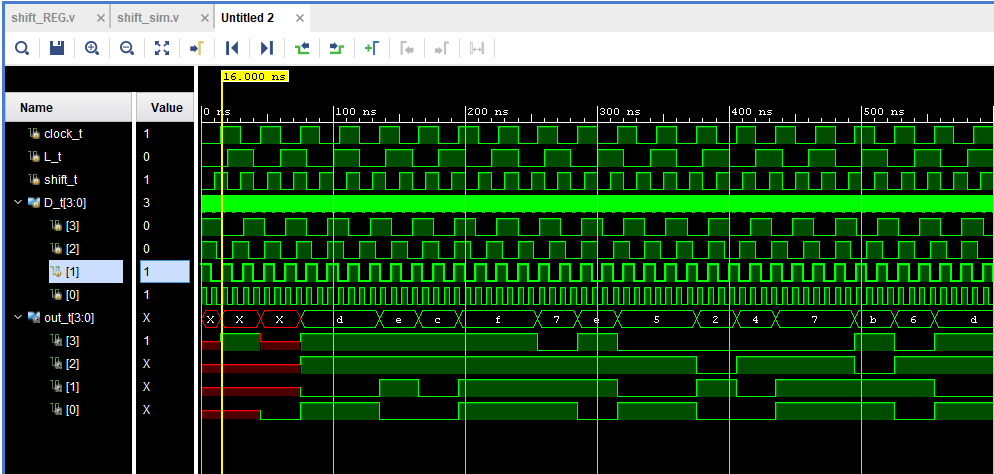
**Counter**



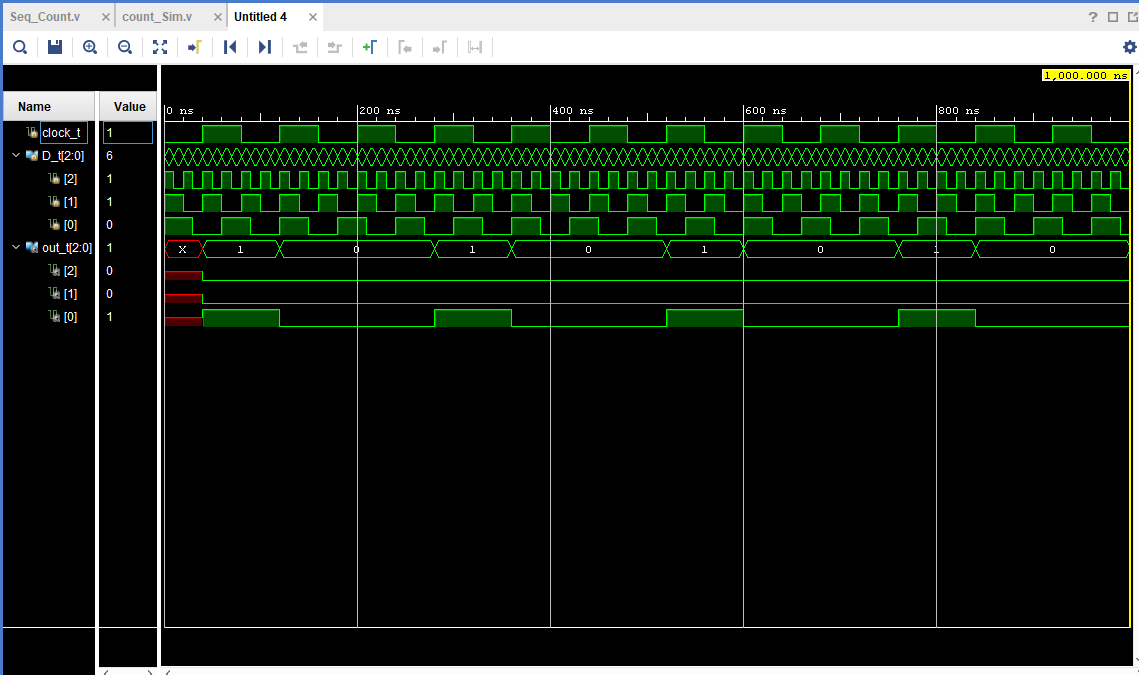
|  |
| --- |
| **Shift register**  shift\_REG.v |
| `timescale 1ns / 1ps  module shift\_REG(  input L,  input clock,  input sh,  input wire [3:0] D,  output reg [3:0] out  );    always@(posedge clock)  begin  out[3] <= (L & D[3]) | (~L & ~sh & out[2]) | (~L & sh & D[3]);  out[2] <= (L & D[2]) | (~L & ~sh & out[1]) | (~L & sh & out[3]);  out[1] <= (L & D[1]) | (~L & ~sh & out[0]) | (~L & sh & out[2]);  out[0] <= (L & D[0]) | (~L & ~sh & D[0]) | (~L & sh & out[1]);  end  endmodule |
| Shift\_Sim.v |
| `timescale 1ns / 1ps  module shift\_sim( );  reg clock\_t;  reg L\_t;  reg shift\_t;  reg [3:0] D\_t;  wire [3:0] out\_t;  shift\_REG UUT (  .clock(clock\_t),  .L(L\_t),  .sh(shift\_t),  .D(D\_t),  .out(out\_t)  );    initial begin  // Initialize clock, L and S inputs to 0  // Initialize D inputs to 1  clock\_t = 1'b0;  L\_t = 1'b0;  shift\_t = 1'b0;  D\_t[0] = 1'b1;  D\_t[1] = 1'b1;  D\_t[2] = 1'b1;  D\_t[3] = 1'b1;  end  always #15 clock\_t = ~clock\_t; //invert the clock\_t input using this period  always #20 L\_t = ~L\_t; //invert the L\_t input using this period  always #10 shift\_t = ~shift\_t ; //invert the S\_t input using this period  always #4 D\_t[0] = ~D\_t[0]; //invert the D\_t[0] input using this period  always #8 D\_t[1] = ~D\_t[1]; //invert the D\_t[1] input using this period  always #12 D\_t[2] = ~D\_t[2]; //invert the D\_t[2] input using this period  always #16 D\_t[3] = ~D\_t[3]; //invert the D\_t[3] input using this period    endmodule |
| **Sequence counter**  Seq\_Count.v |
| module Seq\_Count(  input clock,  input wire [2:0] D,  output reg [2:0] out  );  always@(posedge clock)  begin  out[2] <= D[2] ^ D[1];  out[1] <= ~D[1];  out[0] <= (D[2] & D[1] & ~D[0]) | (~D[2] & D[0]) | (~D[1] & D[0]);  end  endmodule |
| seq\_sim.v |
| `timescale 1ns / 1ps  module count\_Sim(  );    reg clock\_t;  reg [2:0] D\_t;  wire [2:0] out\_t;    Seq\_Count UUT (  .clock(clock\_t),  .D(D\_t),  .out(out\_t)  );    initial begin  // Initialize clock, L and S inputs to 0  // Initialize D inputs to 1  clock\_t = 1'b0;  D\_t[0] = 1'b1;  D\_t[1] = 1'b1;  D\_t[2] = 1'b1;    end  always #40 clock\_t = ~clock\_t; //invert the clock\_t input using this period  always #30 D\_t[0] = ~D\_t[0]; //invert the D\_t[0] input using this period  always #20 D\_t[1] = ~D\_t[1]; //invert the D\_t[1] input using this period  always #10 D\_t[2] = ~D\_t[2]; //invert the D\_t[2] input using this period  endmodule |

Resulting Wave Forms:

**Shift register**

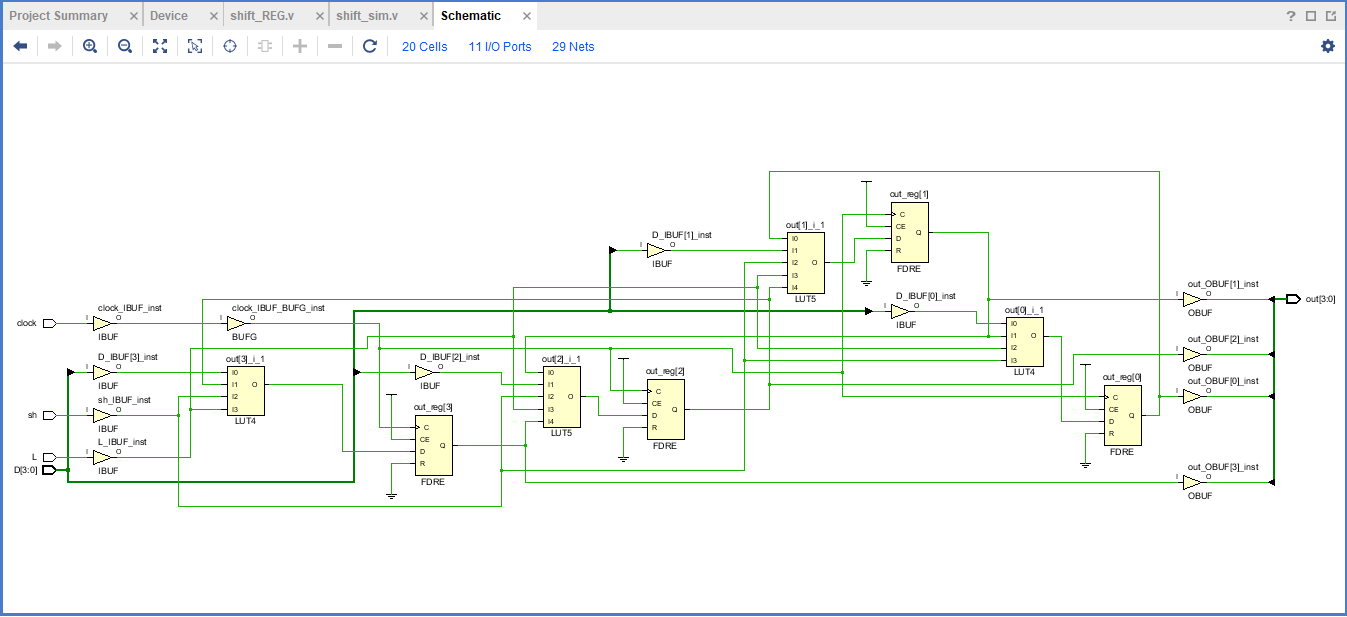
****

**Sequence counter**

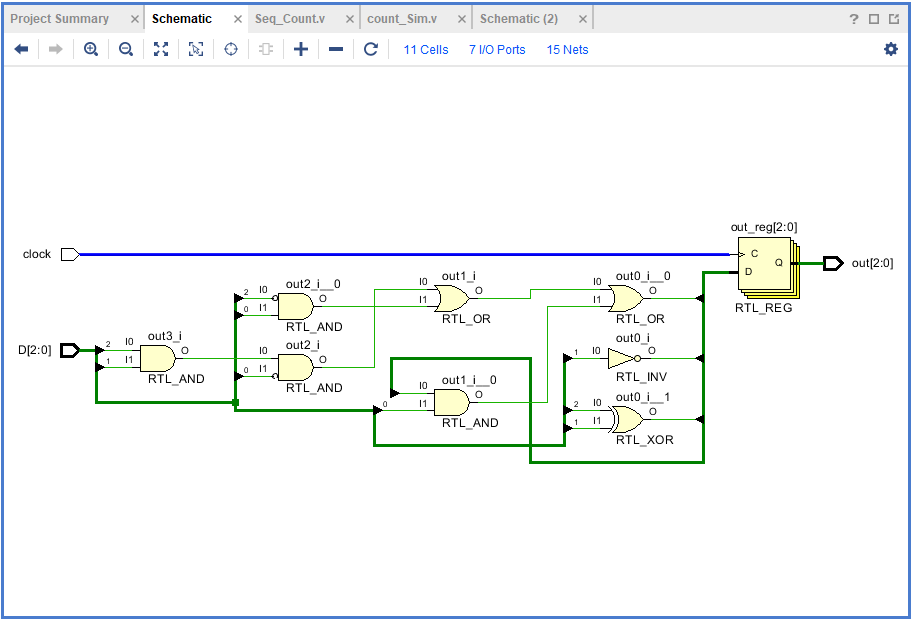
****

Detailed Schematic Diagram:

Shift register



Counter



Test Plan:

Shift register

To test the shift register first we must load a value on to the flip flops with load (T1), then as we click the clock the values will shift depending on the shift and load switch values. If we change switches for our shift (R2) we can shift left or right. This will shift our binary value down the line, “shifting” the LEDs accordingly.

Counter

To test the counter as we enter values onto the board with our switches following the truth table above, the corresponding output will show. 0 -> 2 -> 4 -> 6 1-> 3 -> 5->7 so, if we enter binary 0 into our inputs, our output LED will display 010. We can follow the truth table above in sequence.

Results:

The results of the experiment are the corresponding shift register, as well as a counter that counts by 2’s. in binary

Conclusion:

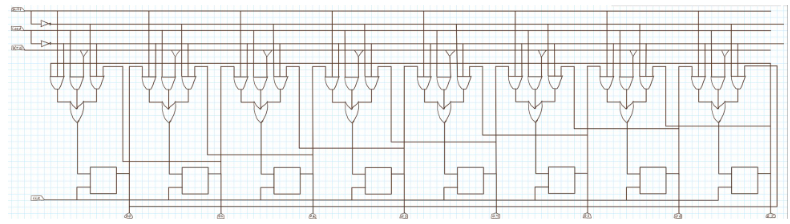
**Questions:**

(To be incorporated within the Conclusion section of your lab report.)

1. What happens if there is noise on the clock input for both part 1 and part 2?

Noise will produce the incorrect outputs due to too many clock changes.

1. How can the shift register of part 1 be expanded to 8 bits? Draw a schematic of this shift register.



1. What happens to the shift register of part 1 if during the left shift operation O3 is tied to D0?

The last bit will go to the first bit, resulting in a left shift

1. What happens to the shift register of part 1 if during the right shift operation O0 is tied to D3?

The first bit will go to the last bit, resulting in a right shift.

1. What problems may arise due to the placement of “don't cares” in the next states of states that are not desired?

The hardware may implement the wrong output, instead of the desired state.

1. How would one avoid the problems that the “don't cares” might produce?

Intentionally design of the circuit paying attention to where they are and leave them out on purpose.

1. Discuss the issues of using switches as clock inputs to edge triggered devices.

Potential for the clock to be triggered twice. Switches produce edge transitions when they switch from on an off causing extra shifts when they are flipped.