378 Lab #1

Farnam Adelkhani

Engineering 378 digital systems design lab #2 -- Prelab

Pre-lab assignment: Draw a circuit diagram to model the functionality of the arithmetic circuit shown below. You may draw the circuit on the level of gates, multiplexers, adders etc.

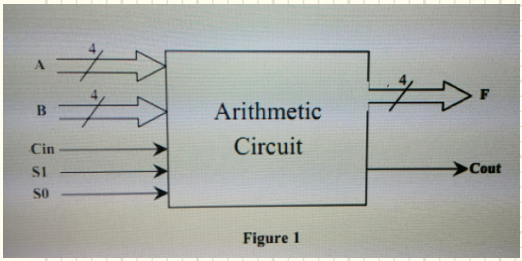
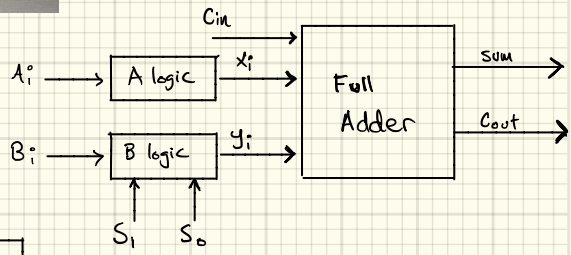
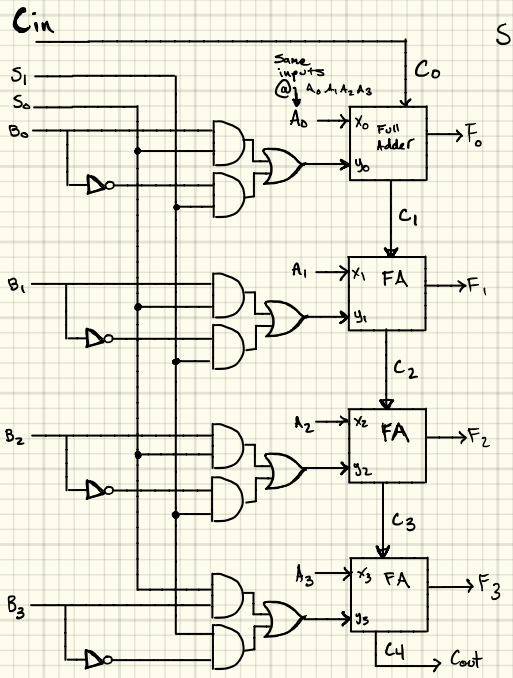


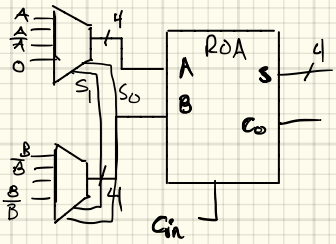
Figure 1

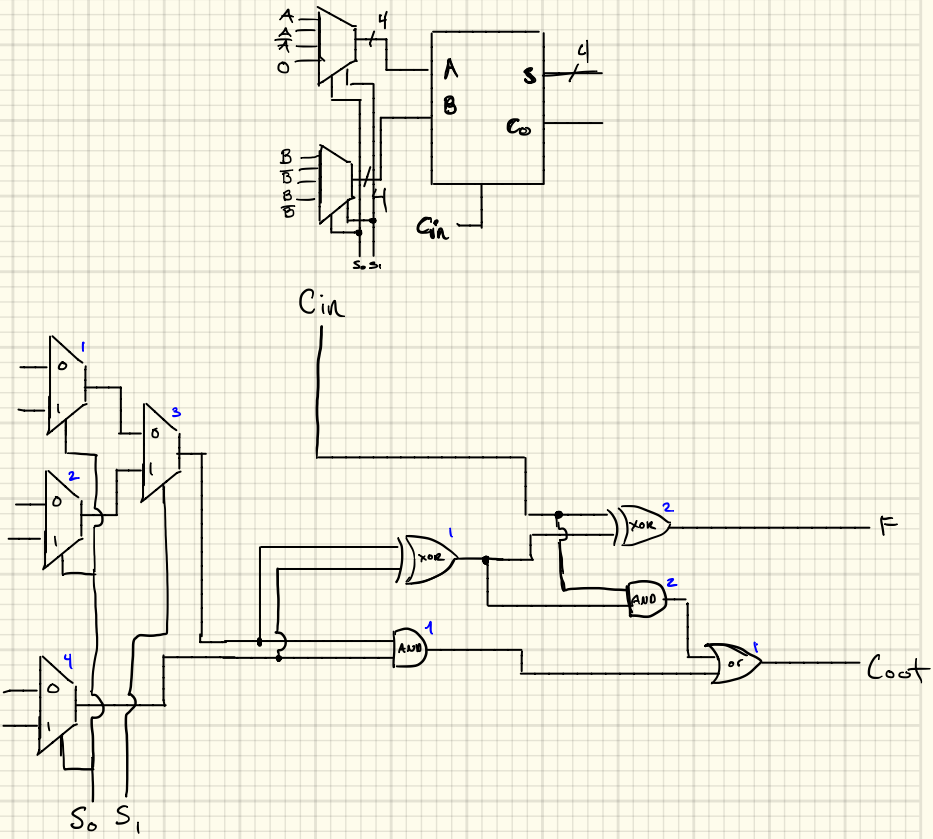
S1	S0	Cin=0	Cin=1
0	0	$F = A + B$ (Addition w/o carry)	$F = A + B + 1$ (Addition with carry)
0	1	$F = A + \overline{B}$	$F = A + \overline{B} + 1$ (Substract: A-B)
1	0	$F = \overline{A} + B$	$F = \overline{A} + B + 1$ (Substract: B-A)
1	1	$F = \overline{B}$ (1's Complement of B)	$F = \overline{B} + 1$ (2's Complement of B)

Table 1

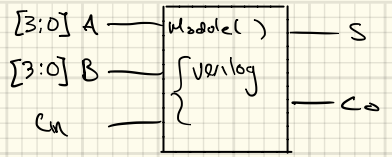







A
A
A
A
0
4
B
B
B
B
4
A
B
S
4
Co
Cin
S0 S1
Cin
1
2
3
4
0
1
XOR
AND
OR
F
Cout
S0
S1

Hint



concatination operator

always @ (*)

{A, B} =

use if or case statement

Part II.

```
module acc8bit (out, clk, in);
output [7:0] out;
input [7:0] in;
input clk;
reg [7:0] out;
initial
out = 8'b0;
always @(posedge clk)
begin
out <= out + in;
end
endmodule
```

ACCUMULATOR take a look at BENCH

```
module acctest;
reg clk;
reg [7:0] in;
wire [7:0] out;
acc8bit a1(out, clk, in);
initial
begin
clk = 1;
in = 8'b00000001;
end
always #10 clk = ~clk;
endmodule
```

Help

Verilog2.v*

```
1  module fullAdder(A, B, Cin, Cout, F);
2  input A, B, Cin;
3  output Cout, F;
4
5  assign #10 Sum = A ^ B ^ Cin;
6  assign #10 Cout = (A && B) || (A && Cin) || (A && Cin);
7
8  endmodule
9
10
11
12 module arithmetic(A, B, Sel, Cin, Cout, F);
13 input [3,0]A, B;
14 input Sel, Cin;
15 output Cout;
16 output [3,0] F;
17
18 wire [3: 1] C;
19
20 always @(Sel or A or A or A' or 0)
21    case Sel
22       2'b00 : H = A;
23       2'b01 : H = A;
24       2'b02 : H = A';
25       2'b03 : H = 0;
26    endcase
27
28 always @(Sel or B or B')
29    case Sel
30       2'b00 : G = B;
31       2'b01 : G = B';
32       2'b02 : G = B;
33       2'b03 : G = B';
34    endcase
35
36 fullAdder FA0 (H, G, Cin, C[1], F[0]);
37 fullAdder FA1 (A[1], b[1], C[1], C[2], F[1]);
38 fullAdder FA2 (A[2], b[2], C[2], C[3], F[2]);
39 fullAdder FA3 (A[3], b[3], C[3], Cout, F[3]);
40
41 endmodule
```

Critical Warning | Error | Suppressed | Flag

0%

EN

2:51

9/8/2

acer

San Francisco State University

Electrical Engineering

ENGR 378 Digital systems design

Lab 3. Latches, Flip-flops and Sequential Circuits

Objectives

• To learn Verilog design of basic sequential circuit components: latches and flip-flops.

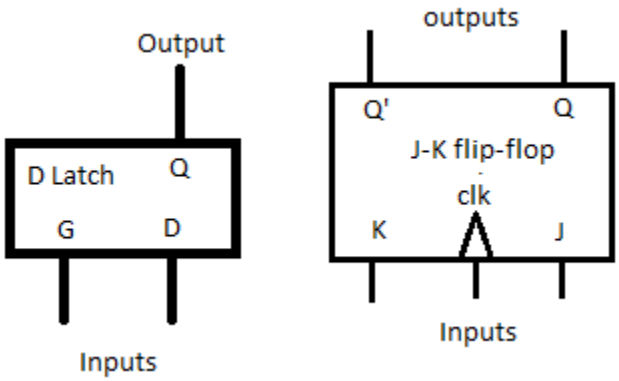
• To learn Verilog design of complete sequential circuits

• To design and experiment simulation testbench modules for sequential circuits using Verilog.

Prelab

(Every group member should do the prelab)

1. Write the truth tables for the following D latch and J-K flip-flops shown below and characterize their output signals in terms of their input signals in words.



2. Using a maximum of two JK flip-flops and no other registers, design a 2-bit counter with an active high reset signal. You can use any method such as circuit diagrams, logic tables, K-maps, etc. for this part as you see fit. Just be sure you are able to translate your design to Verilog code for lab. Also include an active high signal to reset the counter. The counter should run from 0 to 3 and then back to 0.

```
module D_latch_behavior
always @ ( D or enable)
begin
    Q <= D;
    Qbar <= ~D;
end
end module
```

Tasks

1. D Latch design and simulation.

a) Write a Verilog module for the D latch.

b) Write a Verilog testbench module for the D latch and perform the simulation.

c) Verify it works by comparing the waveforms to the truth table from prelab.

```
Module jk(Q, Q', j, k, c);
output Q, Q';
input  j, k, c;
reg    Q, Q';
```

```
initial begin Q = 1'b0;
    Q' = 1'b1; end
always @ (posedge c)
.....
```

2. J-K flip-flop design and simulation.

a) Write a Verilog module for the J-K flip-flop.

b) Write a Verilog testbench module for the J-K flip-flop and perform the simulation.

c) Verify it works by comparing the waveforms to the truth table from prelab.

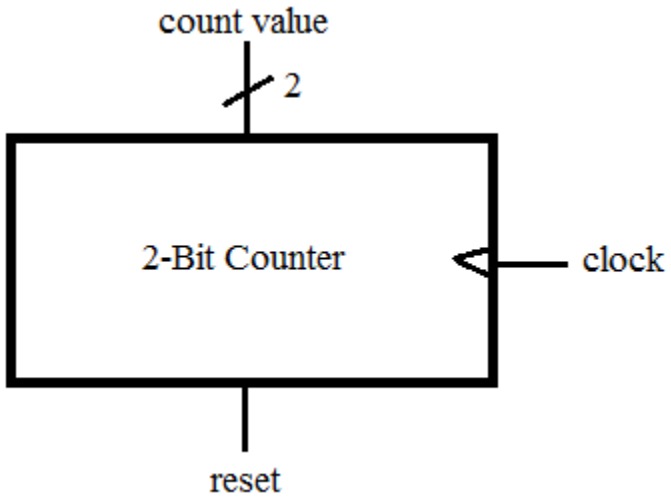
3. 2-bit counter design and simulation.

a) Write a Verilog module for the 2 bit counter (with active high reset signal) using two JK flip-flops.

b) Write a Verilog testbench module for the cou Verilog nter and perform the simulation.

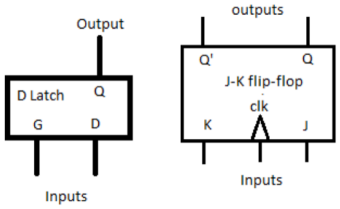
c) Verify the counter works by checking the 2-bit counter counts up from 0 to 3 and back to 0. Also make sure the reset signal resets the count to 0 when set high.

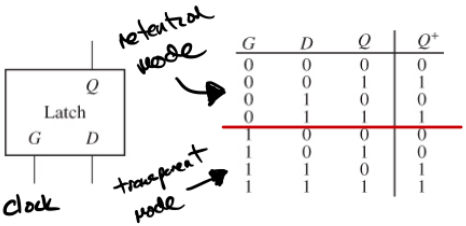
d) Show the counter waveforms to the lab instructor to have it signed off.

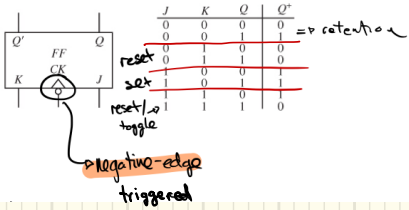


Prelab

1. Write the truth tables for the following D latch and J-K flip-flops shown below and characterize their output signals in terms of their input signals in words.







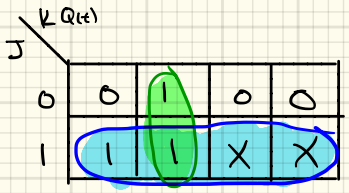
No 0... ➡ positive level sensitive latch

JK flip flop
- when J = 0 & K = 0 ... Q is in retention
- when J = 0 & K = 1 ... reset
- when J = 1 & K = 0 ... set
- when J = 1 & K = 1 ... toggle the circuit.

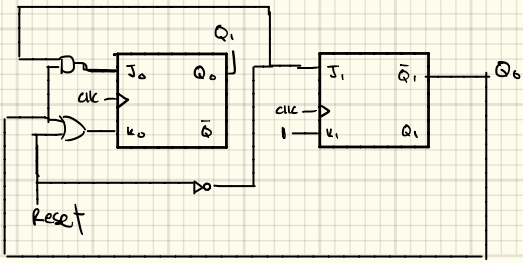
D latch
- when G = 0 & D = 0/1 ... retention mode
- when G = 1 & D = 0/1 ... transparent mode

2. Using a maximum of two JK flip-flops and no other registers, design a 2-bit counter with an active high reset signal. You can use any method such as circuit diagrams, logic tables, K-maps, etc. for this part as you see fit. Just be sure you are able to translate your design to Verilog code for lab. Also include an active high signal to reset the counter. The counter should run from 0 to 3 and then back to 0.

J	K	Q(t)	Q(t+1)	
0	0	0	0	
0	0	1	1	
0	1	0	0	
0	1	1	0	
1	0	0	1	
1	0	1	1	
1	1	0	X	} toggle
1	1	1	X	}



$Q_0, Q_1 = 1$ reset



1. D Latch design and simulation.

a) Write a Verilog module for the D latch.

b) Write a Verilog testbench module for the D latch and perform the simulation.

c) Verify it works by comparing the waveforms to the truth table from prelab.

San Francisco State University

Electrical Engineering

ENGR 378 Digital Systems Design

Lab 4. Downloading to FPGA

Objectives

• To learn how to prepare your Verilog design for implementation on actual hardware, an FPGA.

• To learn how to download our program to a commercial FPGA, and run it.

Prelab

In regards to task 2 of this lab, consider the following diagram as one way to organize the design for the Multi-code Converter Module.

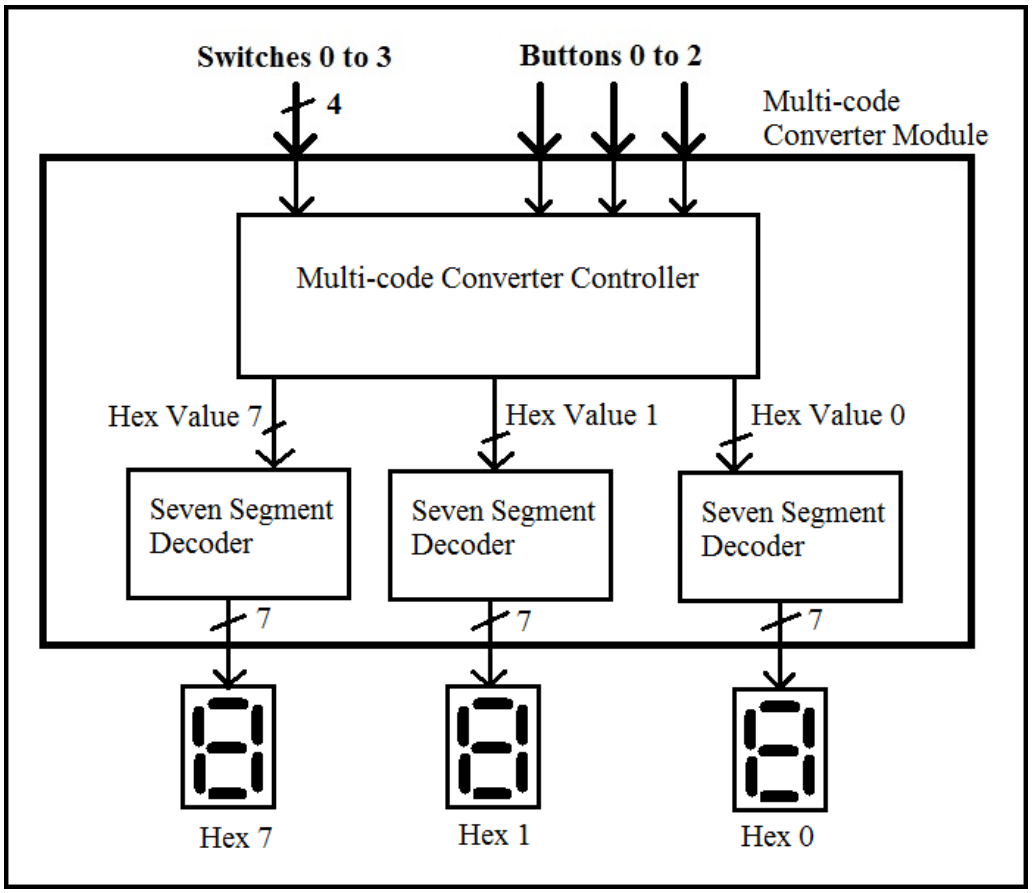


Figure 1: Multi-code Converter Module

For the prelab do **one** of the following and show to TA for prelab sign-off. Regardless of which choice you choose for the prelab, make sure you have an idea of how to write VHDL code to model it:

Choose one of the following to do for the prelab:

A**) Draw the circuit diagram** for the Multi-code Converter Controller in the Multi-Code Converter Module. Simplify where possible.

B) **Write pseudo-code** for the Multi-code Converter Controller in the Multi-Code Converter Module. Show what values are output from the Multi-code Converter Controller for the cases when button 0, button 1, and button 2 are pushed.

C) **In words**, write out how you would facilitate signals to the outputs of the multi-code converter controller given the cases of when button 0, button 1, and button 2 are pressed.

For the Seven segment decoder, write up a truth table for all possible inputs of a 4 bit seven segment decoder and its respective seven bit output that drives each seven segment display. Also notice that each seven segment LED in the display requires an active low signal. You can read the DE2_115 User Manual on how the seven segment displays are used in the "Using the 7-segment Displays" section on page 36.

Tasks

1. The first task will be to download a sample project to your FPGA in the DE2-115 Board. Open the Quartus_Intro_Tutorial_To_Verilog.pdf file from lab 1 and follow the instructions starting from page 29 "Programming and Configuring the FPGA Device" to the end of the tutorial. Resume your work from lab 1 to download the XOR gate design to the FPGA board.

Have the lab instructor sign off for the XOR gate after it is downloaded successfully to the FPGA (signoff #1).

2. Design, simulate and download your own complete application: Multi-code Converter.

a) The program will convert a four bit code into one of the following:

i. A 1-digit Hex number: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

ii. A 1-digit decimal number (BCD): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, (Display 9 for value 10+)

iii. A 2 digits decimal number: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16

b) The 4 bit code will be entered through the 4 switches available in the board (SW0 to SW3).

c) The type of conversion will be selected by pressing one of three push buttons in the board (BTN0 to BTN2).

d) The result of the conversion should be displayed in the two **rightmost** 7-segment displays (no leading zeros).

e) The **leftmost** 7-segment display in the board must show the current type of conversion: “h” for hex, “b” for BCD, “d” for decimal.

Have the TA sign off for the Multi-code Converter after it is downloaded successfully to the FPGA (signoff #2).

Notes

• Each team should borrow (one member of the team will sign for it) the FPGA test board, including power supply and download cable, from the stock room. The team is responsible for this material and should handle it carefully in order to avoid damage. It will be returned at the end of the semester.

B) **Write pseudo-code** for the Multi-code Converter Controller in the Multi-Code Converter Module. Show what values are output from the Multi-code Converter Controller for the cases when button 0, button 1, and button 2 are pushed.

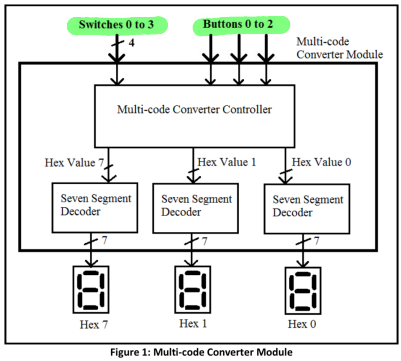


Figure 1: Multi-code Converter Module

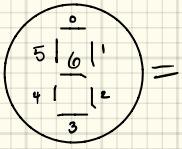
```
Module MultiConverter (Digi2, Digi1, Digi0...
                       swtch, bttn);

input [3:0] swtch;
input [2:0] bttn;

output [6:0] Digi2, Digi1, Digi0;
reg [6:0] Digi2, Digi1, Digi0;

always @ (swtch or bttn) begin;

  if (bttn == 2) begin;   % 1st case
    Digi2 = 7'b "
    case (swtch)
      0: begin Digi1 = 7'b " , Digi2= 7'b " ; end
      .  .     .       .       .
      .  .     .       .       .
      .  .     .       .       .
    endcase
  end
  else if (bttn == ?) begin  // other cases
    .  .
    .  .
    .  .
```



=

Farnam Adelkhani
Engr 378
Pre-lab