Lab 4: Sequential Logic Design (Simple (almost practical) Vending machine)

Starts: starting October 1, 2018 – attend your scheduled lab session
Demo: at the end of your lab session during the week of Oct 8 - 12.
Points (100 points): Pre-lab (15 points) and Demo (85 points)
Code submission: Submit all your .v files on D2L under "Lab4" by 11.59 pm on Oct 12, 2018.

References:

- Lecture Notes (on D2L): FSM 1 (in Unit 2 folder) and verilog_lecture2_seq (in Unit: Verilog folder)

Objectives:

- Design and implement **Button Synchronizer FSM**
- Design and implement **Simple Vending machine**
- Practice writing testbench for sequential circuits
- Use Structural Verilog style to connect multiple modules to implement a digital circuit.

Pre-lab (Group):

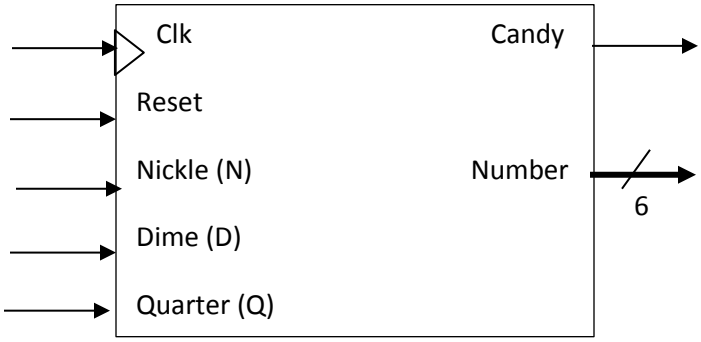
A) (5 pts) Draw a **state diagram of Button Synchronizer** (Page 4).

B) (8 pts) Draw a **state diagram of Simple Vending Machine** (See below (page 1 and 2) – read it carefully before drawing the state diagram)

C) (2 pts) Show TA the DivVal value in ClkDiv module such that ClkOut has a frequency of 5000 Hz.

Lab Overview:

The purpose of the lab is to implement a simple (almost practical) Vending machine on Nexys 4 DDR board.



You are to design the **electronic vending machine running with 5000-Hz clock** which sells a ***candy*** for **30 cents**.

- The machine has a clock (Clk) signal and Reset (Rst) button such that when Reset = 1, FSM goes back to an initial state.
- It accepts N (Nickel = 5 cents), D (Dime = 10 cents) and Q (Quarter = 25 cents)
- It keeps track of the amount of money that the customer enters using the *6-bit output, Number*.

- It releases the candy by setting Candy = 1 and also returns the change through *Number* if the customer enters more than 30 cents.
  For example,

  if the customer enters exactly 30 cents, your machine should go to the state that set Candy to 1 and *Number* to 0 (no change return)

  if the customer enters 35 cents, your machine should go to the state that set Candy to 1 and *Number* to 5 (5c is returned since the candy cost only 30c).

Note:

1) **For this lab, once the machine reaches the state(s) that release the candy, make your machine stay at that state** (this is the almost-practical part since we want to be able to see the outputs, we make the machine stay at that state (state(s) that Candy = 1) until the reset button is pushed. Once the reset button is pushed, the machine goes back to the initial state and waits for the next purchase).
   In practice, the machine will release the candy, return the change (if a customer enters more than 35 cents), and then go back to the initial state waiting for the next purchase by a customer.

2) In practice, the coin receptor is a mechanical device and thus very slow compared to an electronic circuit. There will very likely be an arbitrary long time between an insertion of a coin => at each state, it is possible to have all inputs N, D, Q = 0 ➔ when drawing the state diagram, when all inputs = 0, make it stay at the same state.

3) In practice, the coin receptor can accept only one coin at a time (there is a mechanism that prevents a customer to enter two coins at once), it is not possible to have N = D =1 or N = Q =1, or D = Q =1, or N = D = Q = 1.
   However, on our Nexys4 DDR board, it can happen. For these cases, when drawing a state diagram, make your state diagram stay at the same state (this is another almost-practical part since your machine will take the money if more than one coins inserted at once).

To make the simple Vending machine work on the Nexys 4 DDR board, we have to use more components as shown in the top-level design below. See Action Items for lab demo on page 5 and 6.

This **top-level design (.v file)** has

5 inputs: **Clk** (connected to pin E3 for 100 MHz signal), **N**, **D**, **Q**, and **Reset** (a push button will be used) and

3 outputs: Candy (LED will be used), 7-bit out7 and 8-bit en_out [out7 and en_out are used in TwodigitDisplay)

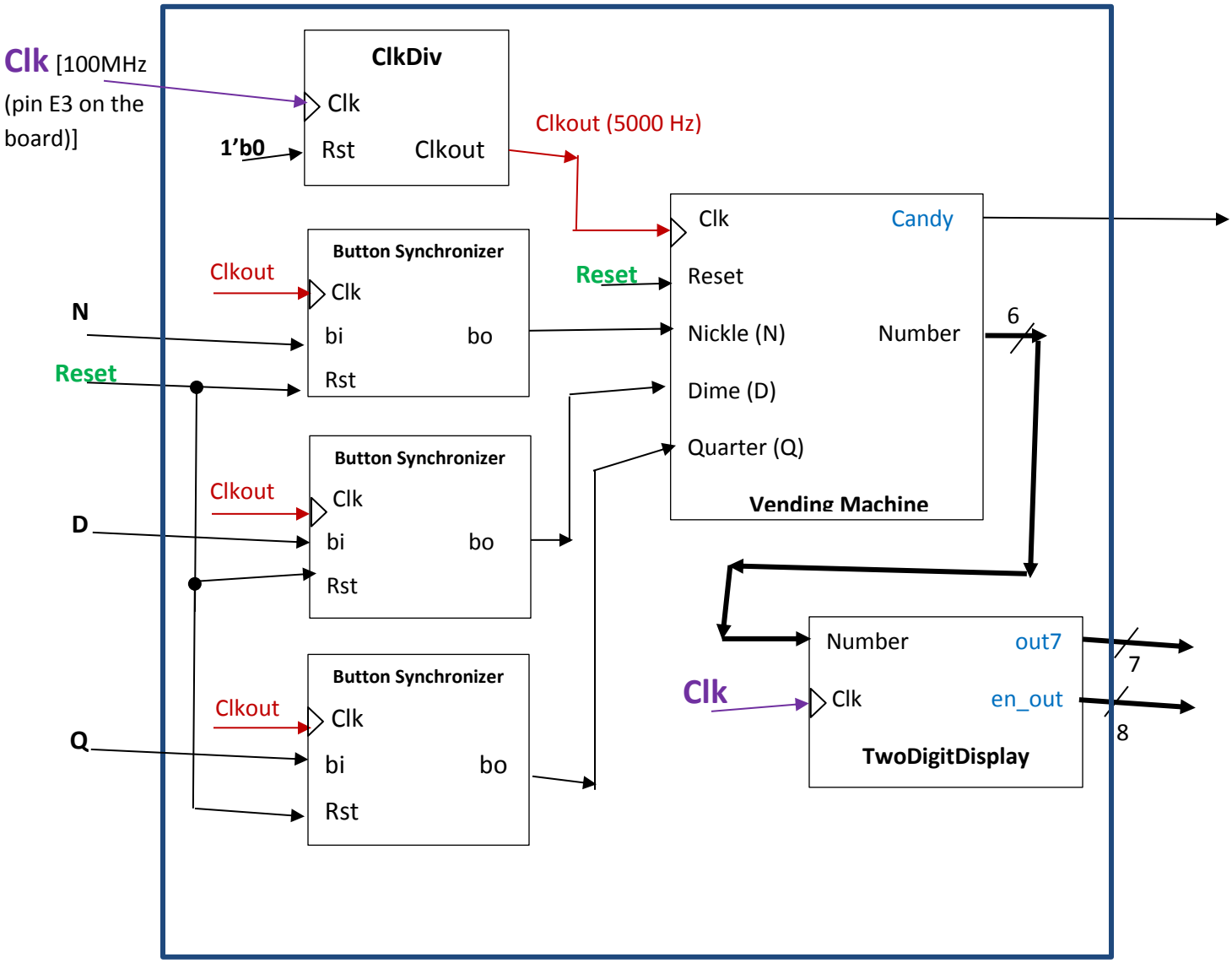


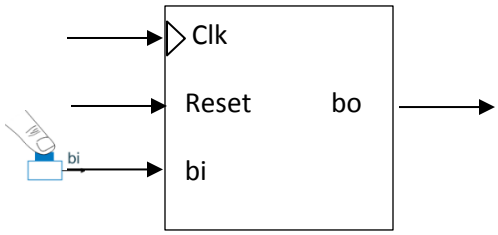
Figure 1: Top-level design

1) Button Synchronizer

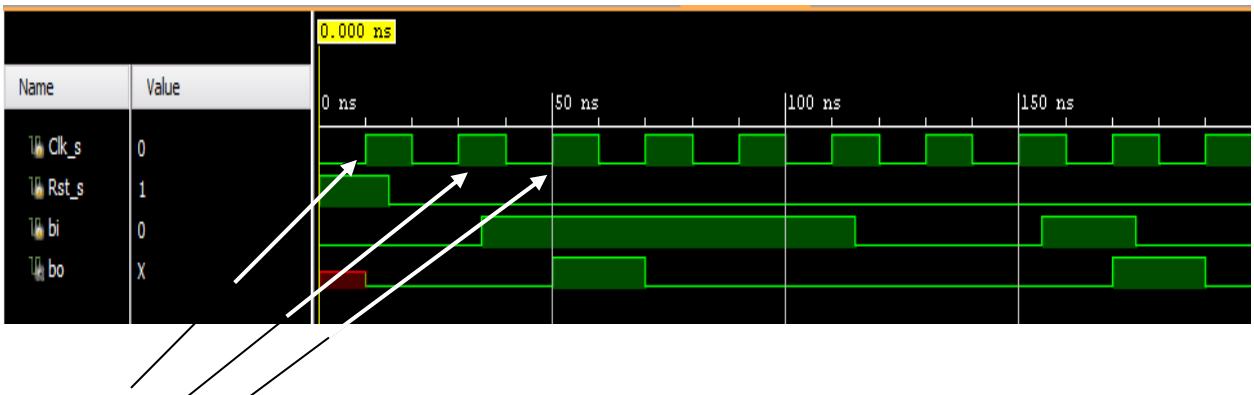
This sequential circuit, Button Synchronizer, converts button press to a **single cycle** duration, ***regardless of length of time that the button is actually pressed*** as shown in the waveform.

Input: bi (1-bit), Reset and Clk

Output: bo (1-bit)



Use the following waveform in your testbench (pre-lab)



@ 1st rising edge of the clock (Clk_s), Rst_s = 1, the FSM is at the initial state
@ 2nd rising edge of Clk_s, Rst_s = 0 and bi = 0, b0 = 0
@ 3rd rising edge of Clk_s, bi = 1, now b0 = 1.
@ 4th rising edge of Clk_s, even though bi = 1, b0 is now back to 0. This is what it means by the statement "it converts a button press to a **single cycle** duration, ***regardless of length of time that the button is actually pressed"***

2) ClkDiv: It has two inputs: Clk and Rst, and one output: ClkOut. The code is given on D2L (the zipped folder lab 3). **MODIFY the DivVal value such that ClkOut has a frequency of 5000 Hz** (instead of 1Hz given in the code).

Action Items for lab demo

Step 0) Show TA your pre-lab

Step 1) Write **Verilog code** and **perform Post-synthesis functional simulation** for the **Button Synchronizer** (Use the input waveform on page 3 for your testbench). When working, show TA the post-synthesis waveform.

Step 2) Write **Verilog code** and a **testbench** for your simple Vending machine and **perform Post-synthesis functional simulation.**

Your testbench should cover at least 5 cases. Below are suggestions of cases that you can use to create a testbench.

- No coin entered at any state in your state diagram
- More than 1 coins are entered at any state in your state diagram. Note: see Note 3) on page 2) how it should behave.
- A sum of exact 30 cents is entered (**again, one coin entered at a time or at the rising edge of the clock**). In this case, your machine should set Candy to 1 and Number to 0 (since no change returned). It can be
  - 3 Dimes (3 * 10 cents)
  - 6 Nickels (6 * 5 cents)
  - 2 Dimes and 2 Nickels
  - 1 Quarter and 1 Nickel
- A sum of 35 cents is entered. For 35 cents, your machine should set Candy to 1 and Number to 5 (5 cents are returned – the candy cost 30 cents)
  - 1 Quarter and then 1 Dime
  - 5 Nickels and then 1 Dime
- A sum of 40 cents is entered. For 40 cents, your machine should set Candy to 1 and Number to 10 (10 cents are returned – the candy cost 30 cents)
  - 3 Nickels and then a Quarter
- A sum of 45 cents is entered.
- A sum of 50 cents (2 Quarters) is entered. For this case, your machine should set Candy to 1 and Number to 20 (20 cents are returned – the candy cost 30 cents)

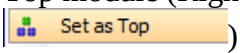
Show the waveform of Post-Synthesis functional simulation to TA to get credits

Step 3) Get the following .v files and put them in the .srcs folder of this lab4 project

- ClkDiv.v file (on D2L in the zipped folder lab 3)
- TwoDigitDisplay.v file and sevensegment.v file from your lab 2
- the .v file of Button Synchronizer (your pre-lab part A)

Use Add Sources to add the above .v files into your lab4 Vivado project.

Create a new .v file to implement the top-level design. You may have to set this new .v file as Top module (Right-click on the file name under Design Sources and choose "Set as Top"



)

Write ***Structural*** Verilog code to connect the following components together as shown in Figure 1 (page 2)

- ClkDiv
- Button Synchronizer
- Your simple Vending machine
- TwoDigitDisplay and seven-segment decoder (from lab 2)

Step 4) Create .xdc file to assign the pins to your top-level design circuit
From Figure 1, there are **5 inputs**: **Clk** (connected to pin E3 for 100 MHz signal), **N**, **D**, **Q**, and **Reset** and **3 outputs**: Candy (LED will be used), 7-bit out7 and 8-bit en_out [out7 and en_out are used in TwodigitDisplay.v file]

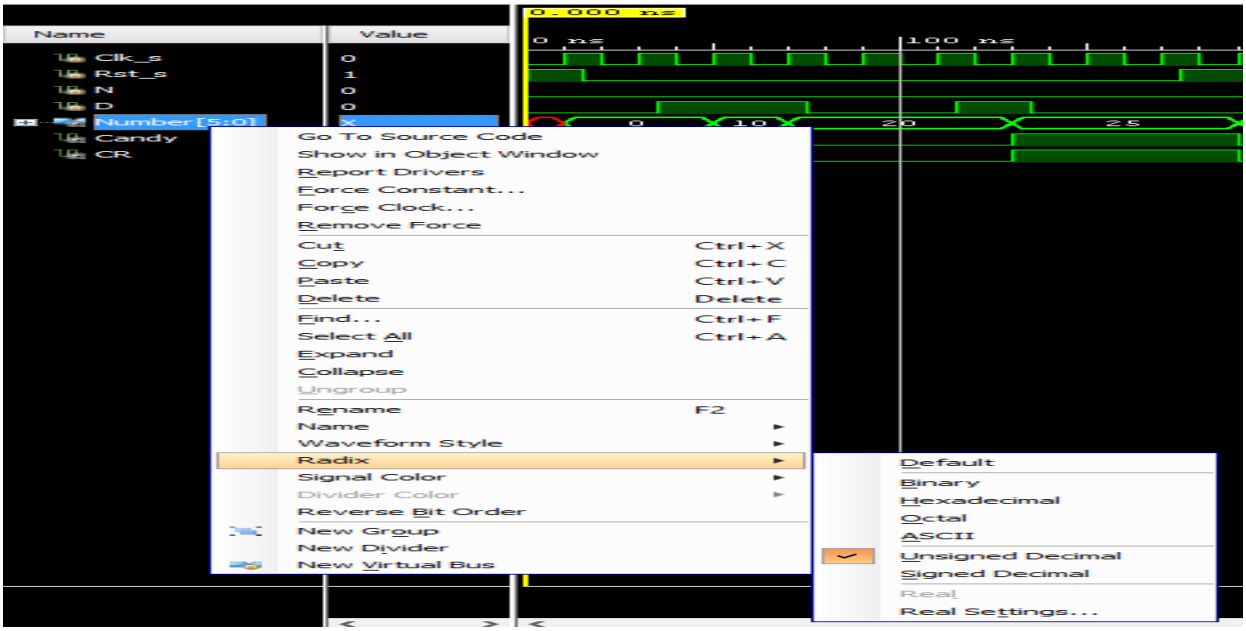
Step 5) Generate Bitstream and **download to the board**. Test that your Vending machine works and demo to TA.
Push the reset button to start over after your machine gets to the state that releases the candy.

Grading Criteria

Step 0) Pre-lab	15 pts
Step 1) Verilog and testbench for Button Synchronizer	8 pts
Step 2) Verilog code and Testbench for Vending Machine (at least 5 cases) and correct waveform from Post synthesis functional simulation	22 pts
Step 3) Structural code for Top-level design	25 pts
Step 4) Correct .xdc file for top-level circuit	15 pts
Step 5) Circuit working on the Nexys 4 DDR board	15 pts

Note (Vivado): To change the base number (Radix) for the vector signal

Right-click on the vector signal, choose *Radix*, then *choose the base-number that you want to use*. In this lab, we will use unsigned decimal since Number is used to represent number of cents that a user enters.


Name
Value
Clk_s
Rst_s
N
D
Number [5:0]
Candy
CR
0
1
0
0
x
0 ns
100 ns
0
10
20
25
Go To Source Code
Show in Object Window
Report Drivers
Force Constant...
Force Clock...
Remove Force
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Delete Delete
Find... Ctrl+F
Select All Ctrl+A
Expand
Collapse
Ungroup
Rename F2
Name
Waveform Style
Radix
Signal Color
Divider Color
Reverse Bit Order
New Group
New Divider
New Virtual Bus
Default
Binary
Hexadecimal
Octal
ASCII
Unsigned Decimal
Signed Decimal
Real
Real Settings...