

Lab 5 - Design, Implement and Test a State Machine

Please pay close attention to the red text in the problem assignment section.

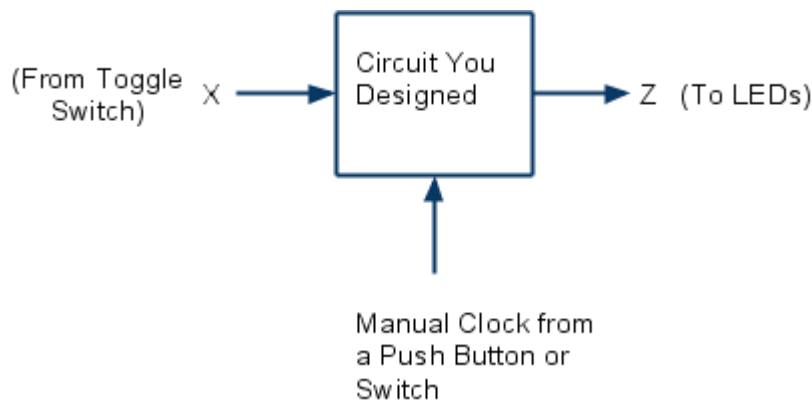
Author

Chengqing Hu, Youngtaek Kim, Mukund K M, Muhammad Faisal Iqbal

Executive Summary

Lab 5 consists of two parts:

- **Part 1** Design a state machine (more specifically, a Mealy sequential circuit of the form shown below), implement its output and next-state logic using only inverters, 2/3/4-input NAND/NOR gates and D flip-flops, and simulate your design in SimUaid.
- **Part 2** Express your design and testbench from Part 1 in VHDL and download to the Xilinx FPGA board.



Problem Assignment

The sequential circuit you are to design and implement is defined by the table below, modulo the changes specified below the table.

Problem	16.01	16.02	16.03	16.04	16.05	16.06	16.07	16.08	16.09	16.10	16.11	16.12	16.13	16.14
1 st Letter of Last	A,O	B,P	C,Q	D,R	E,S	F,T	G,U	H,V,I,W		J,X	K,Y	L,Z	M	N

Name														
------	--	--	--	--	--	--	--	--	--	--	--	--	--	--

The changes specified for each Design Problem are as follows,

We have made these changes to counter the use of old solutions. If you turn in a solution that solves the unmodified problem, and you cannot explain how you came up with the solution, we will treat your solution as copied and immediately refer the matter to Student Judicial Services. (As per the SJS website, an "F" in the course is a common penalty for plagiarism on a major paper.)

16.1: $Z = 1$ for any input sequence ending in 1101 or 011; determine the output sequence for each of the following input sequences:

- (1) 1 1 0 0 1 0 1 1 0 1 0 1 0 1 1 1 0 1 1 0 1 1 0 1
- (2) 0 0 1 1 0 0 1 1 0 1 0 1 0 1 1 0 1 0 1 0 1 1 0 1

16.2: $Z = 1$ for any input sequence ending in 0010 or 100; determine the output sequence for each of the following input sequences:

- (1) 0 0 1 1 0 1 0 0 1 0 1 0 1 0 0 0 1 0 0 1 0 0 1 0
- (2) 1 1 0 0 1 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0

16.3: Same as the book, except that the output should be the **complement** of the BCD code (i.e., output 1111 instead of 0000 on input 0011, output 1110 on input 0100, etc.).

16.4: Design a sequential circuit which adds **two (instead of "six" in the text)** to a binary number in the range 0000 through 1001. (The gate limitation specified in the book remains unchanged.)

16.5: $Z = 1$ for any input sequence ending in 1001 or 010; determine the output sequence for each of the following input sequences:

- (1) 1 1 0 0 1 0 0 0 0 1 1 0 1 0 1 1
- (2) 0 1 0 1 1 1 0 0 0 0 1 0 0 1 1 1

16.6: $Z = 1$ for any input sequence ending in 1010 provided that the sequence 001 has never occurred; determine the output sequence for the following input sequences:

- (1) 1 0 1 0 1 1 1 0 1 0 0 1
- (2) 0 1 0 1 0 1 0 0 1 0 1 0

16.7: $Z = 1$ if the total number of 0's received is even (consider zero 0's to be an even number of 0's) and the sequence 11 has occurred at least once; determine the output sequence for each of the following input sequences:

- (1) 1 0 0 1 1 0 1 0 1 1
- (2) 0 1 0 0 0 0 1 1 0 0 0 1

16.8: $Z = 1$ for any input sequence ending in 1100 or 001; determine the output sequence for each of the following input sequences:

- (1) 1 1 1 0 1 1 1 0 0 1 0 1
- (2) 0 0 0 1 1 0 1 1 1 0 0 1

16.9: The output Z is determined by two rules. The initial output from the circuit is $Z = 0$. Thereafter, the output Z will equal the *preceding* value of X (rule 1) until the input sequence **110 (instead of "001" in the text)** occurs. Starting with the next input after 110, the output Z will equal to the *complement* of the *present* value of X (rule 2) until the sequence **011 (instead of "100" in the text)** occurs. Starting with the next input after 011, the circuit

output is again determined by rule 1, etc. **The minimum number of states is 5 (instead of 6 in the text)**. Note that overlapping 110 and 011 sequences may occur; determine the output sequence for each of the following input sequences:

(1) 0 1 1 0 1 1 0 1 1 1 0 0

(2) 1 0 0 1 1 1 1 0 0 1 0 0

16.10: Same as the book, except that you are to output the complement of the BCD code, i.e., output 1111 if the digit is 0, 1110 if the digit is 1, etc.

16.11: Design a Mealy sequential circuit which adds **four (instead of "five" in the text)** to a binary number in the range 0000 through 1010. (The gate limitation specified in the book remains unchanged.)

16.12: No change.

16.13: $Z = 1$ for any input sequence ending in 0101, provided that the sequence 110 has occurred at least once; determine the output sequence for the following input sequences:

(1) 0 1 1 0 1 1 0 0 1 0 1 0

(2) 0 1 0 1 1 1 0 1 0 1 0 1

16.14: $Z = 1$ whenever the total number of 1's in the sequence is odd, provided that the sequence 10 has occurred at least once; determine the output sequence for the following input sequences:

(1) 0 1 1 1 0 0 1 0 1 1 0

(2) 1 1 1 1 0 1 0 1 1 1 0

Detailed Instructions

Lab 5.1

This part consist of the following three steps:

Step 1: Study Unit 15 in the textbook and answer the [preparatory questions](#).

- Review the example illustrated in Fig 13.16 of Unit 13 and then answer the preparatory questions. **Turn in your answered questions sheet that you've downloaded from the link above. Remember- if you skip this step, its very likely that you will flunk the demo.**

Step 2: Design a sequential circuit using gates and D flip-flops.

1. Derive a **state graph** and **state table** for the assigned problem. Reduce the table to a minimum number of states.
2. Check your state table using the **LogicAid state table checker** (The "solution state table" used for checking can be found at "Course Documents (Lab) " in the Laboratory Course module in your **Blackboard System**). When LogicAid verifies your state table, print the table and the verification window using the *Print All* Command.
3. Make a **state assignment**. State assignment is the process of choosing which combinations of flip-flop states to use for states in the finite state machine. Trying all possible state assignments is not practical, as

there are far too many possibilities to consider. The following (heuristic) guidelines will tend to place 1s together on the next-state maps and thus simplify the logic. (Section 15.8, Page 490 of the textbook further develops the concept of state assignment.)

- States which have the same next state for a given input should be given adjacent assignments.
- States which are the next states of the same state should be given adjacent assignments.
- States which have the same output for a given input should be given adjacent assignments.

The reset state must be assigned 000. Below is an example of optimal state assignment.

- We wish to find the optimal state assignment for the following state table
- With the straight binary order for state assignment we get the following equations for D-Flip Flops

```

*****
D(Q1) = X1'Q2 Q3' + X1 Q2 Q3 + X1'Q1 Q3      Input Cost =
12  Gate Cost = 4
D(Q2) = X1 Q3' + Q1 Q3' + X1'Q1'Q3 + X1 Q2' Input Cost = 13      Gate Cost = 5
D(Q3) = Q2 Q3 + X1'Q1'Q2' + X1'Q2'Q3'      Input Cost =
11  Gate Cost = 4
Z1 = X1'Q1      Input Cost = 2      Gate Cost = 1
***Total Input Cost = 38***
***Total Gate Cost = 14***

```

```

*****
*****

```

- Now if we follow the three rules guidelines listed above, the following groups of states should be adjacent
 1. (1,3,4) (2,5) (0,1,2,4,5)
 2. (1,2) (2,3) (2,4) (3,5)
 3. (0,1,2,3) (4,5)
- After state assignment our transition table looks like this:

And the equations and gate counts for this new state assignment are as follows:

```

*****

```

```

D(Q1) = X1'Q2 Q3 + X1 Q1'Q2 Q3'      Input Cost = 9      Gate Cost = 3
D(Q2) = X1 + Q3 + Q2      Input Cost = 3      Gate Cost = 1
D(Q3) = Q2'Q3' + X1      Input Cost = 4      Gate Cost = 2
Z1 = X1'Q1      Input Cost = 2      Gate Cost = 1
***Total Input Cost = 18***
***Total Gate Cost = 7***

```

```

*****

```

By following the guidelines, we are able to reduce both input cost and gate cost by **50%!**

After making an appropriate state assignment, **Derive the equations** for the next state in terms of the present states and input X by hand, using K-maps.

4. Use the method given on Page 15 of the *LogicAid User's Guide* to enter your state assignment in LogicAid to derive the equations for the above assignment and **check with the equations** derived by hand. Repeat the same for binary order state assignment and compare the gate cost. Print out both the state assignments and their corresponding equations in Logic Aid.
5. **Implement** the circuit with the equations derived from your assignment (not the binary assignment) using D flip-flops and gates in SimuAid.

Step 3: Test your design by simulating it on a computer.

- **Manual Test** - Connect switches to X, CLOCK, RESET. Determine the **transition table** for your circuit experimentally by using the input switches and observing the flip-flop states and circuit output in SimUaid. Follow Steps 1, 2, 3 and 4 in Section 16.7 of the textbook. **Make sure you fully understand how to do it as it is important for your demo. Complete the input and the output sequences (X and Z) below which are to be included in the final paperwork you need to turn in for Lab 5.1.**

(1) X = _____
Z = _____
(2) X = _____
Z = _____

The input sequence for which you need to test can be seen from the problem definition in your textbook, except for 16.3, 16.4, 16.10, 16.11 and 16.12, for which we have given the test sequence below.

- **Automatic simulation** - For this part, you need a 'clock' and 'input signal' for your CLK and X respectively. (Read Section 2.3 of the *SimUAid User's Guide*). For the same input sequence as before simulate the design automatically and obtain the timing diagram. Use a clock period of 100ns and inputs changing at 25ns, 125ns, 225ns ...and so on. Attach probes to the signals you want to observe (CLK, X, Z is good enough. You may add Q1, Q2, Q3). When you have the timing waveform, change the scale to 20ns/div (Select Options > Scale > 20ns/div when the timing window is active in SimUAid.) **Print the waveforms in landscape mode** (Select Options > Print Setup, select "Landscape", "OK", then print normally), **and clearly mark the times to read the Z output and the output values. Also mark any false outputs on the wave form. Include this in your lab report.**
- **SimuAid State checker** - You may check your design using SimUAid checker. Those checker files (e.g., 16-1.chk) can be downloaded from "Course Documents(Lab)" in Blackboard System. Remove your switches for X and RESET. Go to Parts -> Checker and then select the correct checker file based on your problem number from the location where you saved the checker files. Use the same 'clock' for your checker clk pin as used in automatic simulation. Connect 'X' pin of the checker to all the wires that were previously connected to X. Connect the 'R' pin to all the RESET wires. Connect 'Z' pin to your output Z. Then go to the Simulate menu and click on Go. If it works, it will say "PASS". **The checker should pass in order for you to get the full credit.**

For the problems whose input test sequences are not included in the textbook, here are the sequences we want you to test.

16.3: (1) 1100 0010 1010 0110

- (2) 1110 0001 1001 0101
- 16.4: (1) 0000 1000 0100 1100
(2) 0010 1010 0110 1110
- 16.10: (1) 0000 0010 1010 0110 1110
(2) 0001 1001 0101 1101 1111
- 16.11: (1) 0000 1000 0100 1100 0010
(2) 1010 0110 1110 0001 1001 0101
- 16.12: (1) 0000 1000 0100 1100 0010
(2) 1010 0110 1110 0001 1001 0101

Checklist for Lab 5.1 submission: 150 points possible

1. [Coversheet for Lab 5.1](#)
 2. Your answered [Preparatory Questions](#) sheet [Question a => 8 points, others => 2 points each]
 3. State graph and table, K-maps, and equations (by hand)
 4. Printout of state table and verification window in Logic Aid (*Print All* command in LogicAid)
 - 5-1. A state assignment (by hand)
 - 5-2. Printout of two different state assignments (binary order and another one you arrived at) in LogicAid
 6. Transition table determined from the SimUaid simulation (by hand)
 7. Output sequences for Z under two different input sequences
 8. Printout of waveforms in landscape mode (scaled to 20ns/div)
 9. Printout of SimUAid circuit (with minimum no. of gates)
- (Before submission, you should save/remember all your Lab 5.1 work for Lab 6.2)

Demo Example

Problem Statement

Design a Mealy sequential circuit which investigates an input sequence X and will produce an output of Z = 1 for any input sequence ending in 101. After completing your design, starting in the proper initial state (reset state), determine the output sequence for the following input sequence:

X = 0 0 0 1 1 0 1 1 0 0 1 0 1 0 1 0 0

Solution

1. State Machine Design

First, let's determine how many states we need to use.

Initially, we will start the circuit in a reset state designated S0. If a 0 input is received, the circuit can stay in S0 because the input sequence we are looking for does not start with 0.

However, if a 1 is received, the circuit must go to a new state (call it S1) to "remember" that the first input in the desired sequence has been received.

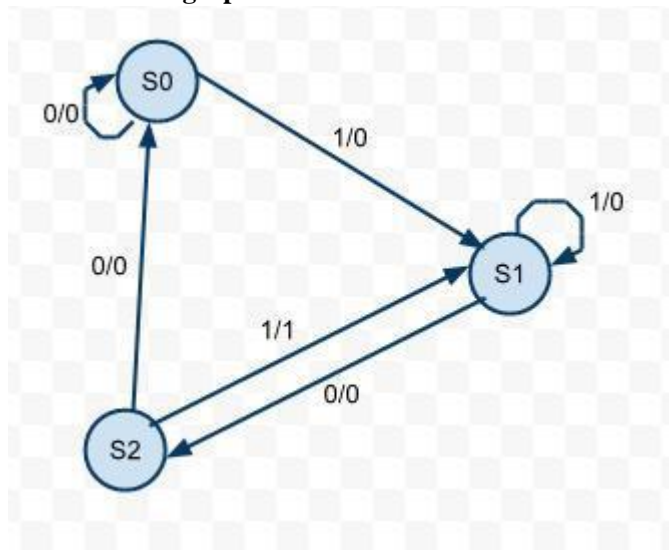
When in state S1, if we receive a 0, the circuit must change to a new state (call it S2) to remember that the first two inputs of the desired sequence (10) have been received; if we receive a 1, the circuit will stay in state S1.

If a 1 is received in state S2, the desired input sequence (101) is complete and the output should be 1. Since the circuit is not supposed to reset when an output occurs, we cannot go back to S0 after receiving a 1 in state S2. Instead, because the last 1 in a sequence can also be the first 1 in a new sequence, we can return to S1.

However, if a 0 input occurs in state S2, we have received two 0's in row and must reset the circuit to state S0 because 00 is not part of the desired input sequence, and going to one of the other states could lead to an incorrect output.

Finally, we've got three states for this state machine: S0, S1 and S2. Because the state machine has three states (less than 4 states), we know that only 2 flip-flops are enough (since $2^2 = 4 > 3$).

Here's the **state graph**:



2. Entering State Table in LogicAid

Next, use LogicAid to input your hand-written state table:

INPUT STATE TABLE FORMAT

Number of Next State Columns:

Number of Input Variables:

Number of Output Variables:

Length of State Names:

Input and Output Variable Names:

☐ Use Default Names (Do Not Enter)

☒ Enter Names ☐ Use Current Names

Machine Type:

☒ Mealy Machine ☐ Moore Machine

Column Headings:

☒ Use Default Straight Binary Order

☐ Enter Heading ☐ Use Current Heading

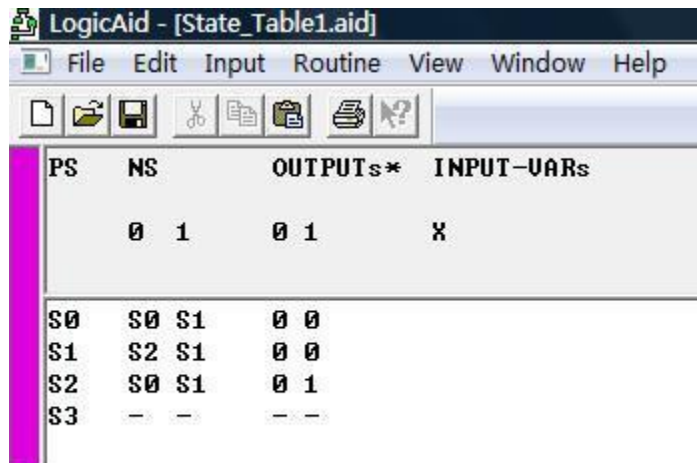
INPUT VARIABLE NAMES:

OUTPUT FUNCTION NAMES:

Select Range

☒ 1-16 ☐ 17-32

Here's the **state table** specified in LogicAid:



LogicAid - [State_Table1.aid]

PS	NS	OUTPUTs*	INPUT-VARs
	0 1	0 1	X
S0	S0 S1	0 0	
S1	S2 S1	0 0	
S2	S0 S1	0 1	
S3	- -	- -	

Then perform the LogicAid state table checker, and get the verification:

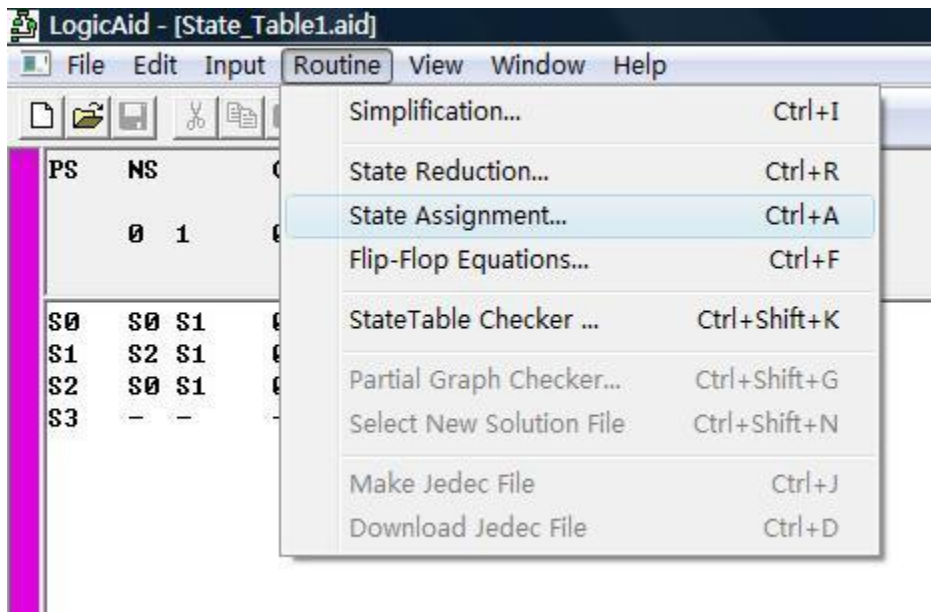


3. State Assignment

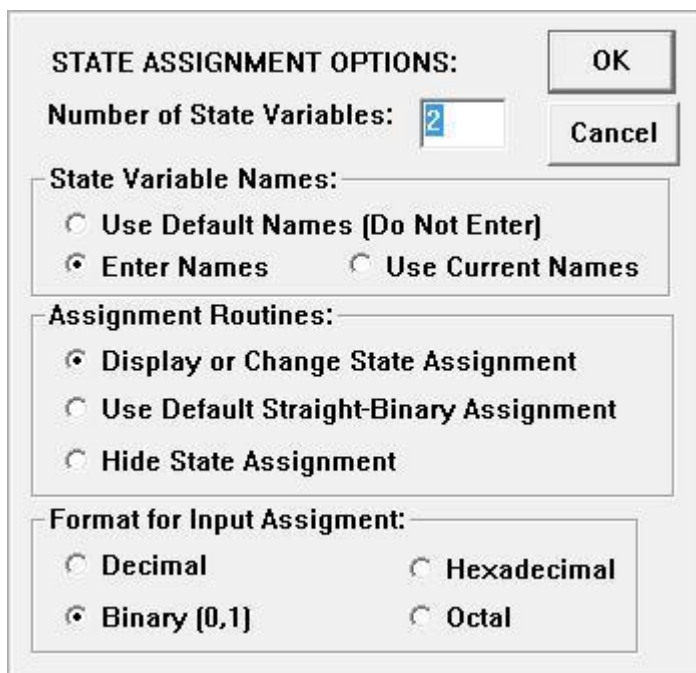
After verification, we perform a state assignment - specifically we select s0 to be encoded as flip-flop state 00, s1 to be 01, and s2 to be 10. The following is the **transition table** after the state assignment.

AB	A+B+		Z	
	X=0	X=1	X=0	X=1
00	00	01	0	0
01	10	01	0	0
10	00	01	0	1

Meanwhile, make the state assignment using LogicAid following the steps below:



Specify it:



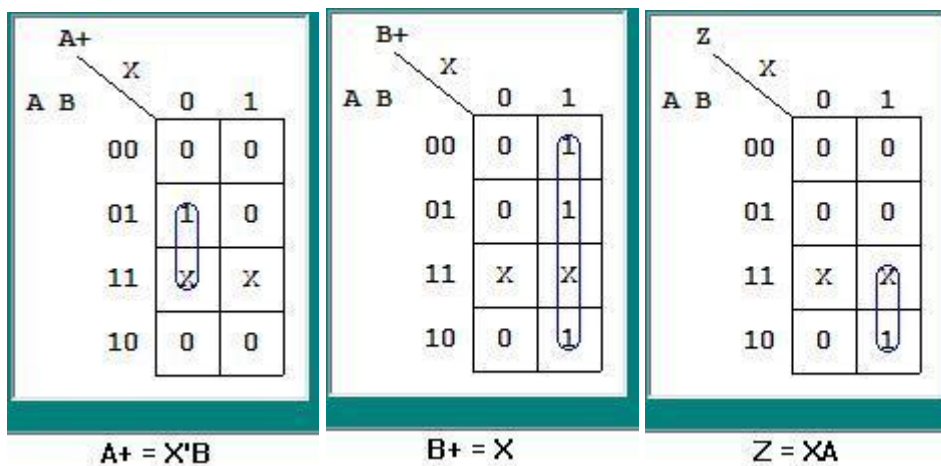
Here you've got the assigned state table as follows,

LogicAid - [State_Table1.aid]

ASSIGN	PS	NS	OUTPUTs*	INPUT-VARs
		0 1	0 1	X
00	S0	S0 S1	0 0	
01	S1	S2 S1	0 0	
10	S2	S0 S1	0 1	
11	S3	- -	- -	

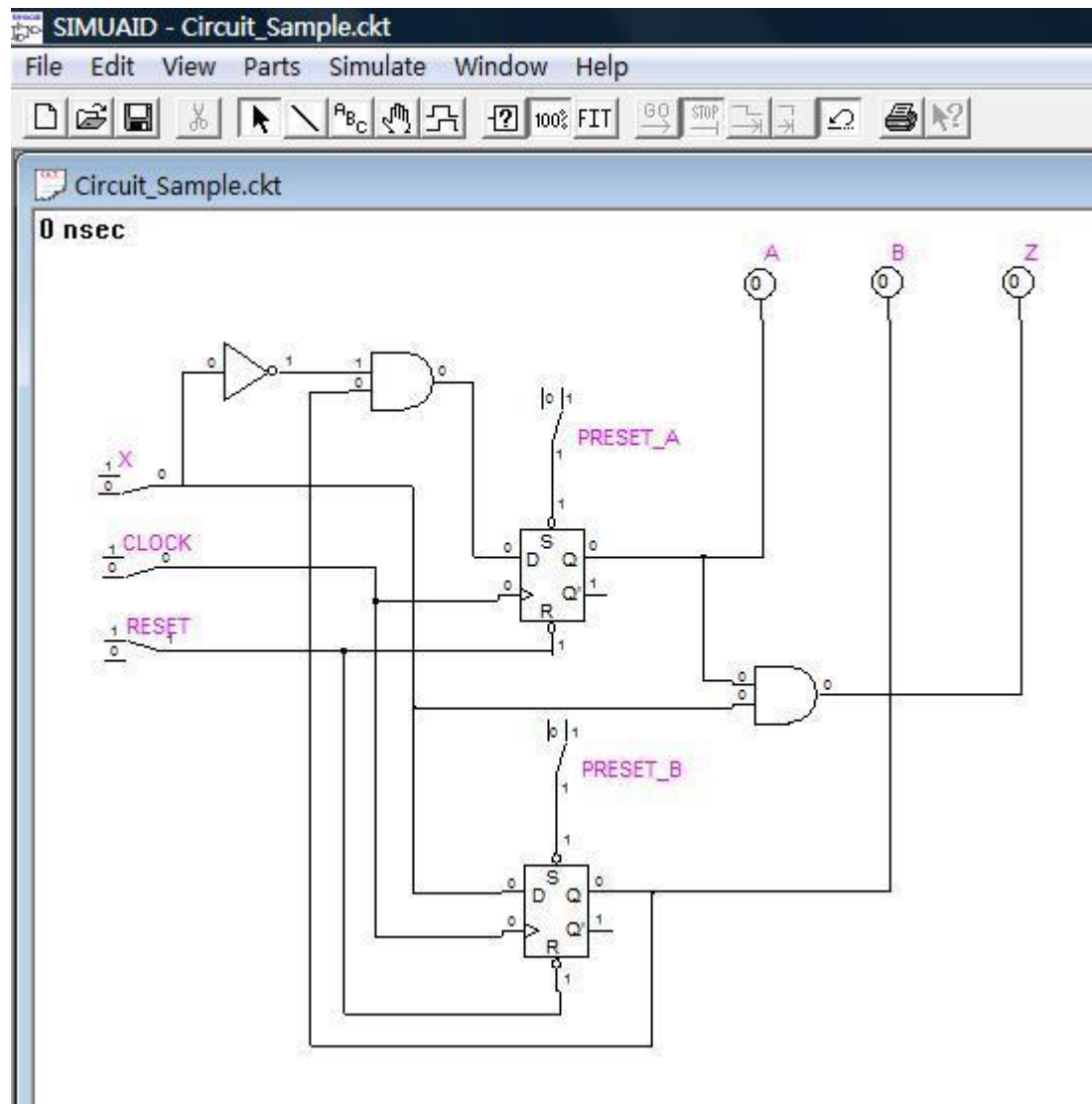
4. Derive the logic equations

From the transition table, we plot the next-state maps for the flip-flops and the map for the output function Z and derive the equations for A^+ , B^+ and Z:



5. Draw the circuit in SimUaid

Draw the schematic of the circuit in SimUaid, and test its functionality:



6. Generate the VHDL code for the design in SimUaid (Lab 5.2)

Here is the VHDL code for the above schematic generated by SimUaid:

```

1
2 -- This file has been automatically generated by SimUaid.
3
4 library ieee;
5 use IEEE.STD_LOGIC_1164.ALL;
6 use IEEE.STD_LOGIC_UNSIGNED.ALL;
7 use IEEE.STD_LOGIC_ARITH.ALL;
8
9 library SimUaid_synthesis;
10 use SimUaid_synthesis.SimUaid_synthesis_pack.all;
11
12 entity Circuit_Sample is
13 port(PRESET_A, PRESET_B, X, CLOCK, RESET: in STD_LOGIC;
14
15     A, B, Z: out STD_LOGIC
16     );
17 end Circuit_Sample;
18
19 architecture Structure of Circuit_Sample is
20     signal Vnet_0, Vnet_1, Vnet_2, Vnet_3, Vnet_4, Vnet_5, Vnet_6: STD_LOGIC;
21 begin
22     VHDL_Device_0: Dflipflop port map (CLOCK, Vnet_3, PRESET_A, RESET, Vnet_4, Vnet_5);
23     VHDL_Device_1: Dflipflop port map (CLOCK, X, PRESET_B, RESET, Vnet_2, Vnet_6);
24     VHDL_Device_2: inverter port map (X, Vnet_1);
25     VHDL_Device_3: and2 port map (Vnet_1, Vnet_2, Vnet_3);
26     VHDL_Device_4: and2 port map (Vnet_4, X, Z);
27     B <= Vnet_2;
28     A <= Vnet_4;
29 end Structure;

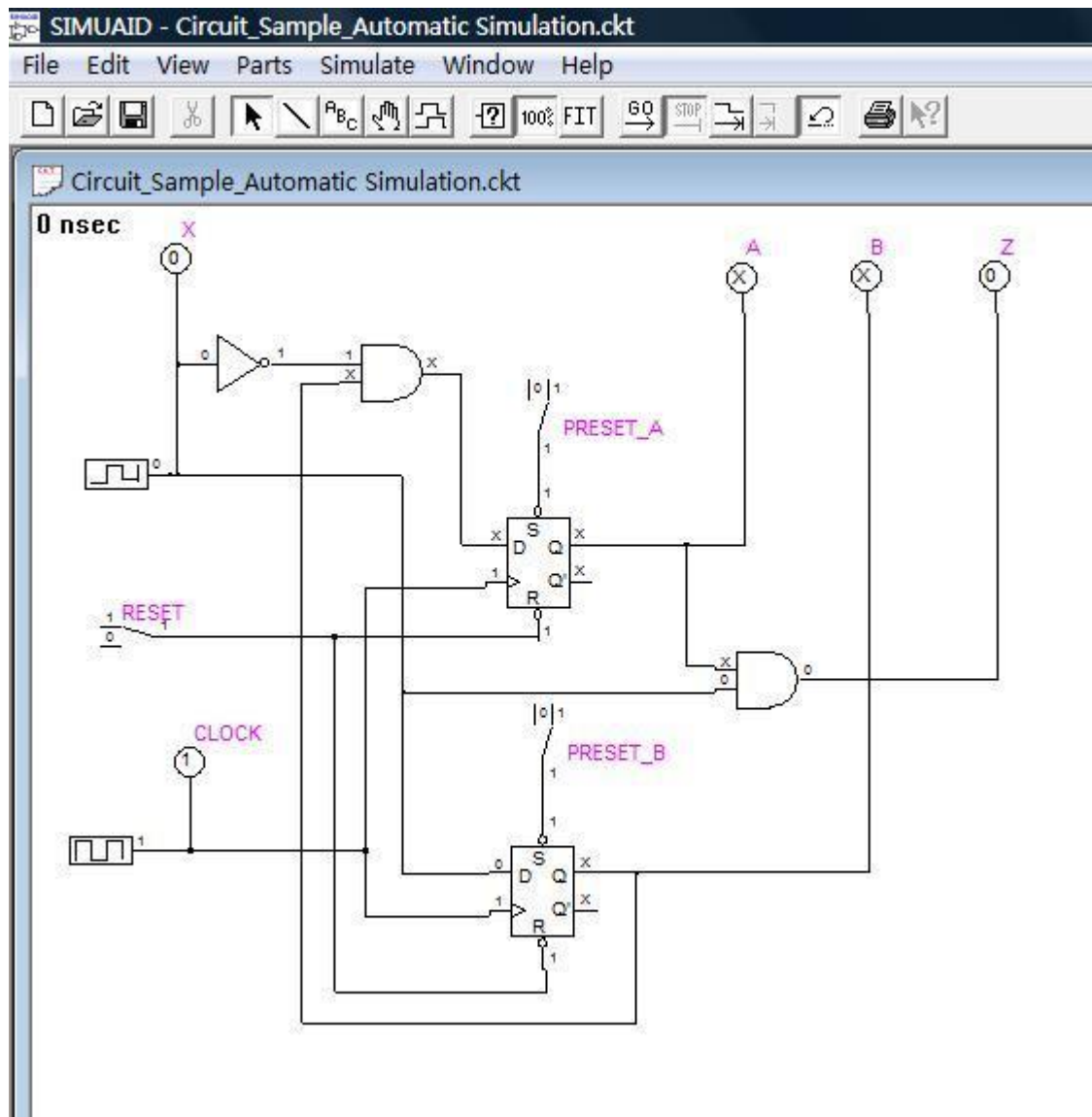
```


7. Test the design in SimUaid

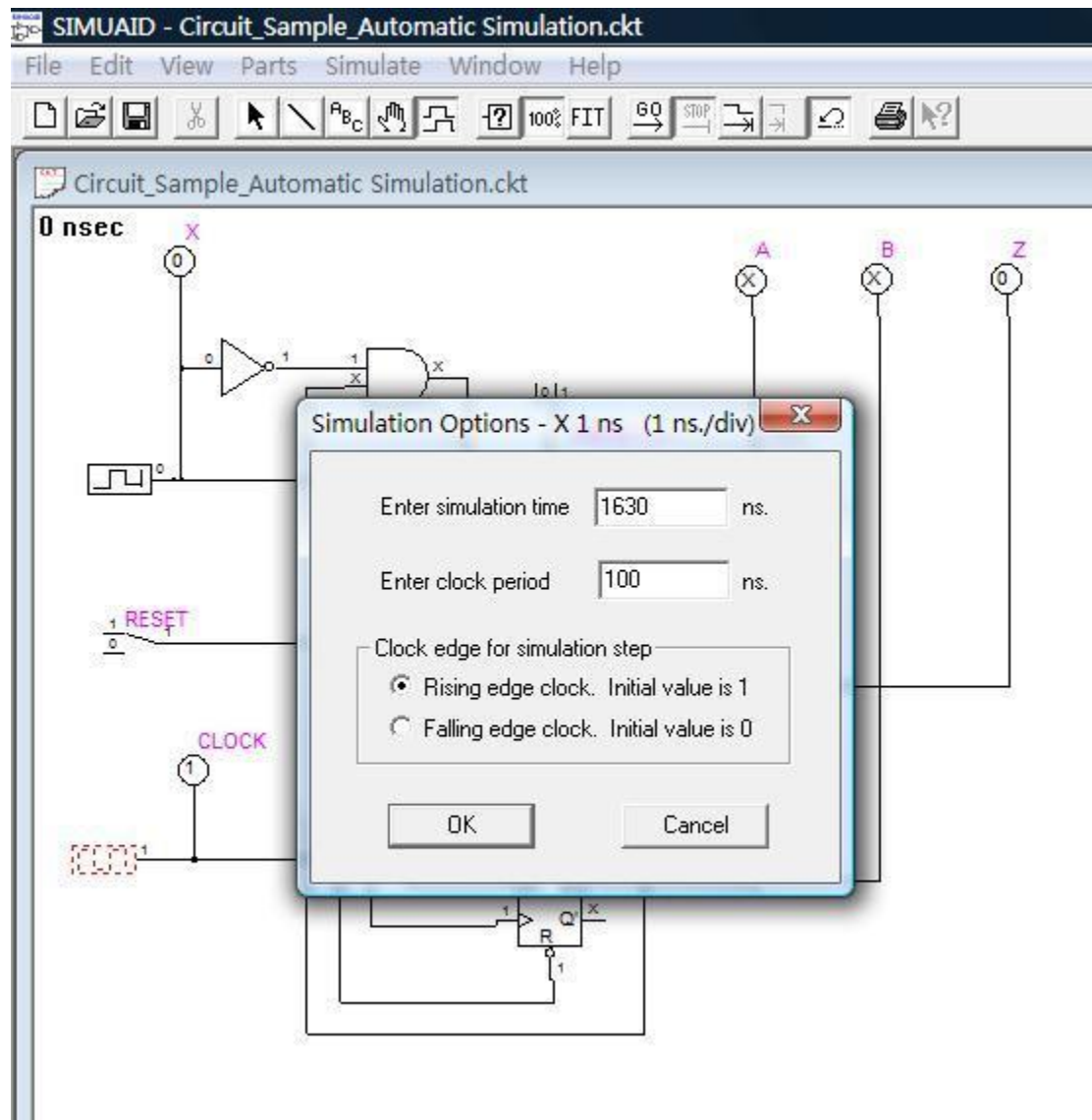
From here, note down the **transition table** determined experimentally in SimUaid, and see if it matches the one from our design. Then **manually** test the simulated design using the required input sequence for X. In this case, the relationship between the required input sequence, X and the corresponding output sequence, Z is as follows,

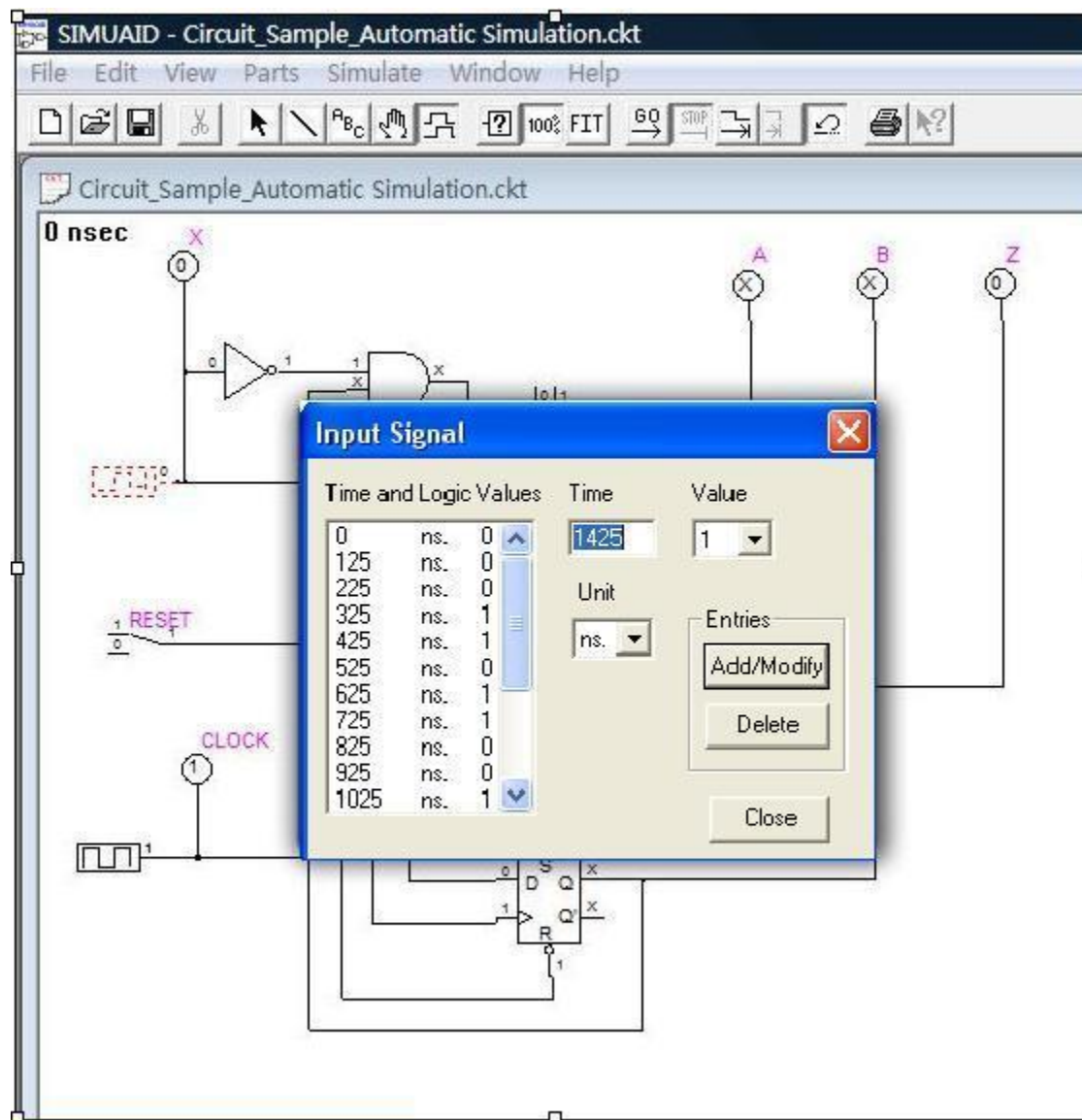
X=00011011001010100
Z=00000010000010100

Then display the simulator timing waveforms for CLOCK, X, Z, and the flip-flops outputs. In SimUaid, we use **Automatic** simulation, which is usually performed for sequential circuits, especially for circuits involving flip-flops. To automatically simulate our logic circuit, we will need to insert a Clock and an Input Signal in our circuit design. The Input Signal will go in place of the switches of the input to our logic circuit: You can pick the input generator and the clock generator from the parts menu.

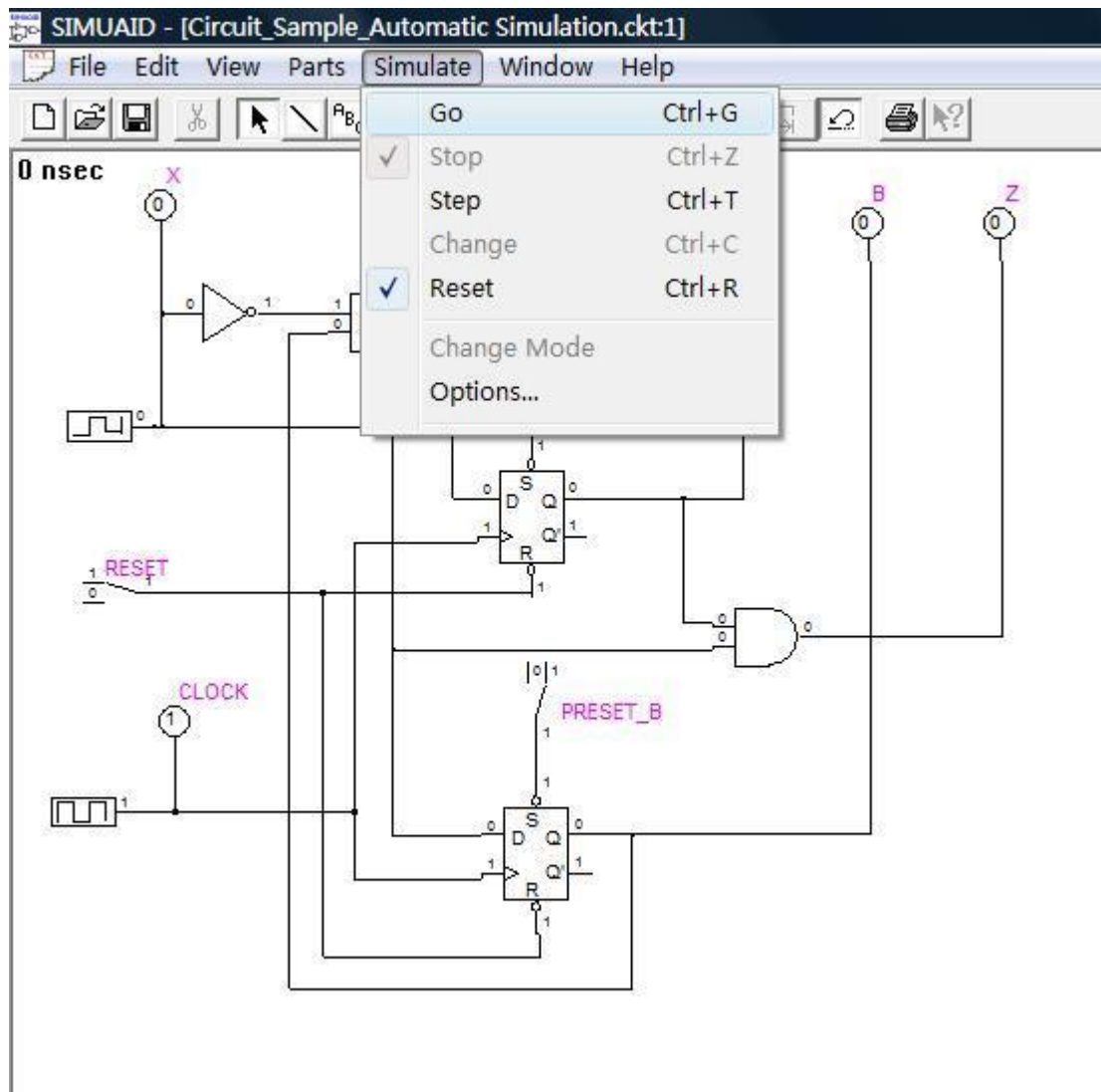


Configure the Clock and Input Signal "X" by clicking on the  icon on the tool bar and then clicking on the Clock or the Input Signal "X". Clicking on the clock will bring up the Simulation Options dialog box. Use a 100ns clock period and change the X input sequence (using the same input sequence that we've used in the manual test) 25ns after the rising edge of the clock (i.e., at 125ns, 225ns, 325ns, etc.) However, the first X input value should be given at 0ns or 25ns (actually, any time point is possible before the first rising clock edge at 100ns).

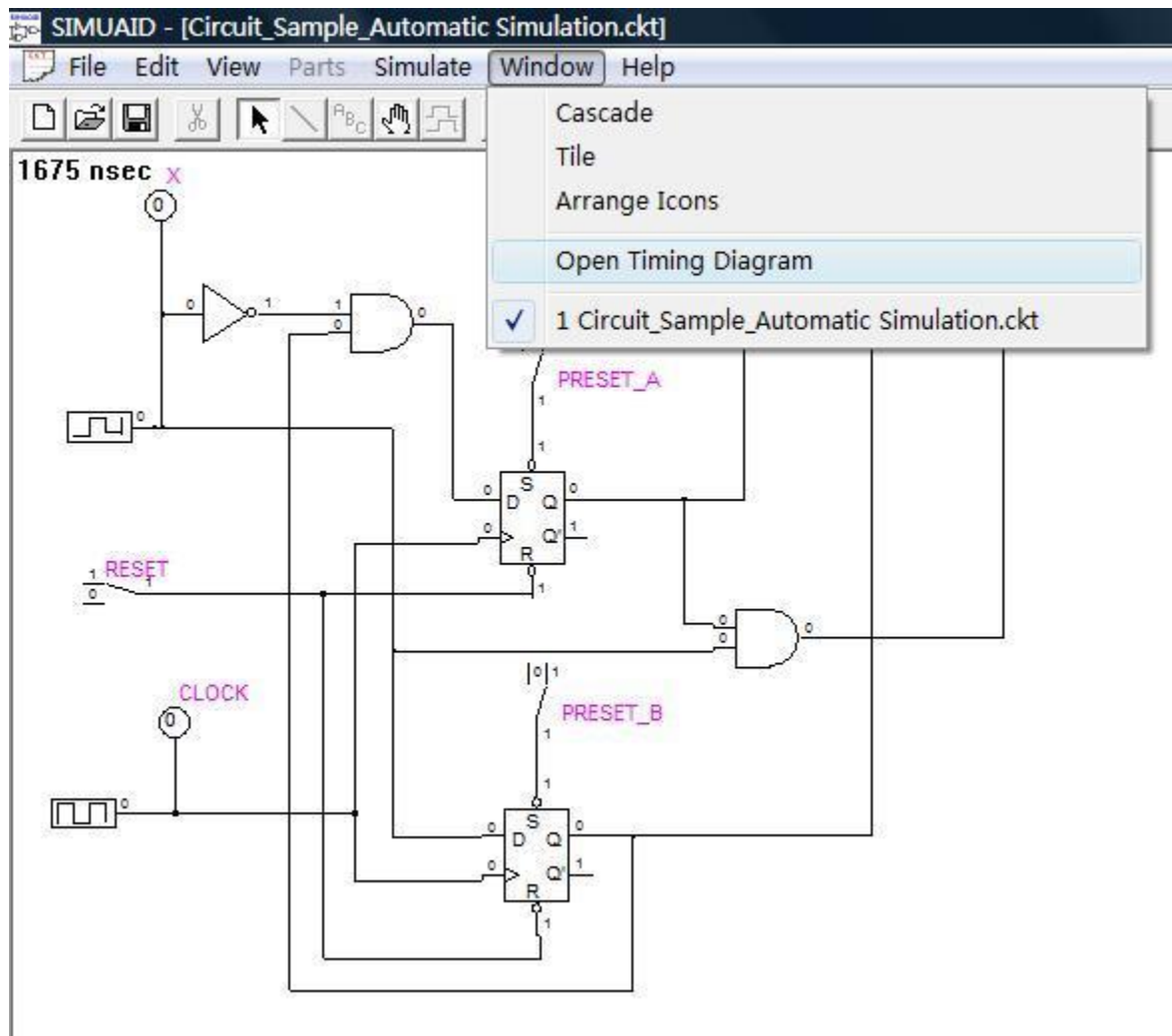




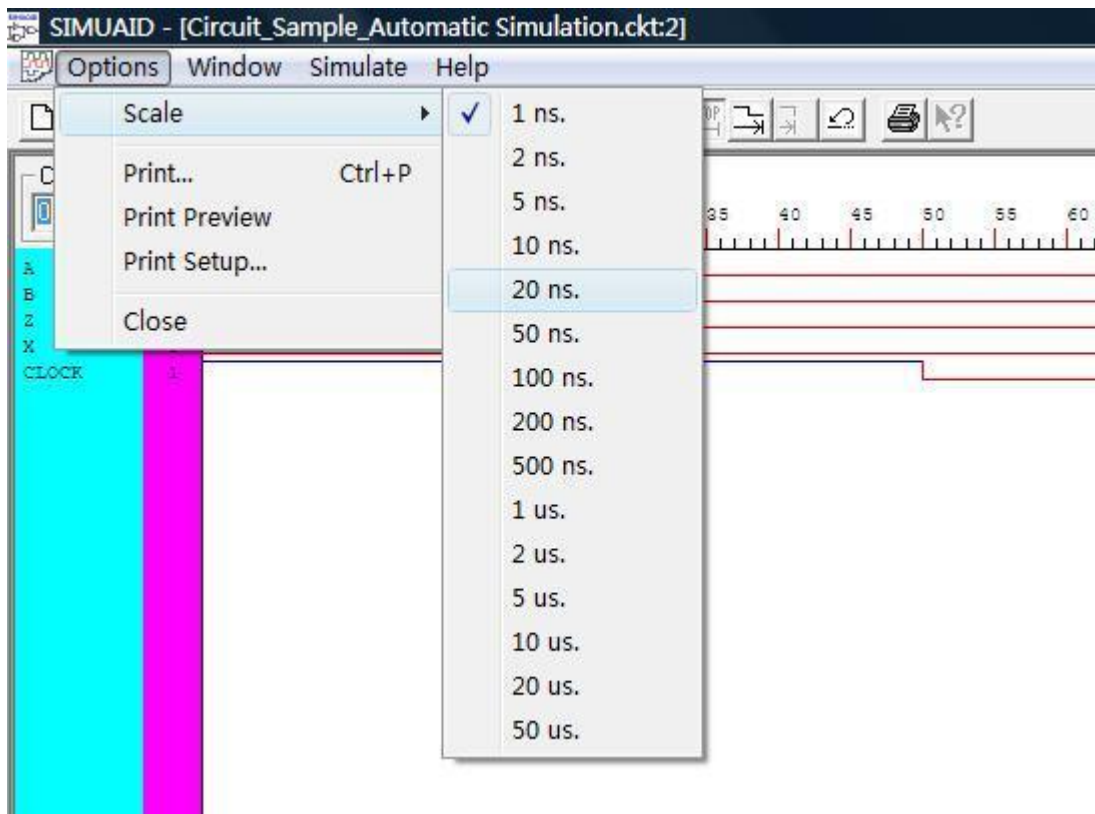
After we have configured the settings, let's run the simulation!



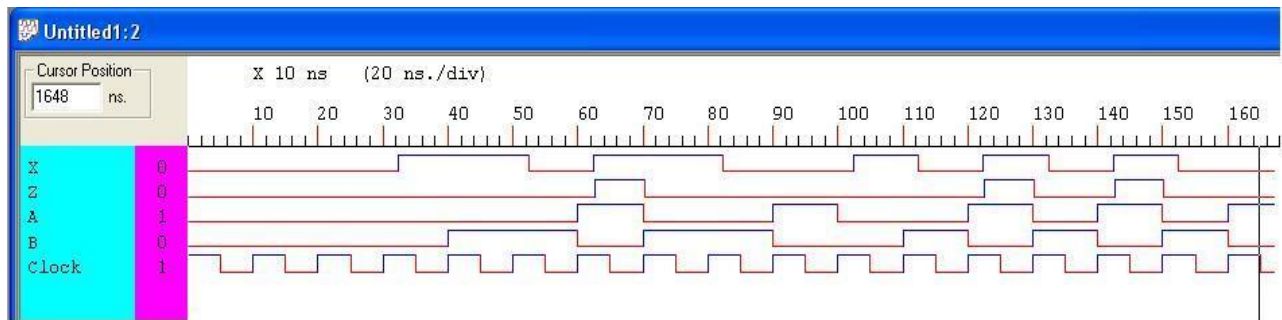
Now the simulation is completed, let's open the timing waveform window:



Now we have the timing waveform window, so let's change the scale to 20ns/div: (Select Options > Scale > 20ns/div when the timing window is active in SimUaid.)



Then we get the automatic simulation result:



Now print the waveforms in landscape mode (Select Options > Print Setup, select "Landscape", "OK", then print normally), and clearly mark the times to read the Z output and the output values (0's and 1's). Also mark any false outputs on the waveform.

```
*****
*****
*****
*****
```

Lab 5.2

Overview:-

After the manual and automatic simulations in Lab 5.1, we can now convert the schematic into VHDL file using SimUaid, and implement our design onto the Xilinx FPGA board. This section can be divided into two parts:

1. Manually test the design on the board. (The steps are the same as that is in Lab 3)
2. Automatically test our design on the board. (Basically the steps stay the same, except for the slight difference caused by the testbench and "pin.ucf" file, whose information is entirely included in **Detailed Instructions for Part Two.**)

Detailed Instructions:-

First, make sure your circuit has *Probes* connected to the output Z, and to the flip-flop outputs (Q1, Q2, and Q3). Delete all other *Probes*. Delete all *Switches* connected to the flip-flop *Presets*, and Connect each *Preset* to a

"+V" part. Make sure that you have a single *Switch* connected to all the *Reset* inputs. **You must label your *Switches* and *Probes* as shown below:**

- | | |
|--------------------------|-------------------|
| input | X |
| reset | rst |
| clock | clk |
| flip-flop outputs | Q1, Q2, Q3 |
| output | Z |

- Convert your design (the SimUaid file) into a VHDL file using SimUaid. After converting your circuit to VHDL, open the VHDL file and change the name of the entity to "SM" if the name is not SM. Also modify the architecture declaration to match the name "SM". The port declaration should look like the following; the order of the signals do not matter:

```
entity SM
  Port (rst, clk, X: in std_logic;
        Q1, Q2, Q3, Z: out std_logic);
end SM;
```

- Follow the instructions in Lab 3 Part 1 to add this vhd file onto a new project.
- You will add a VHDL testbench file("testbench_modified.vhd") (**the file can be downloaded from Blackboard Course documents section**) to your Xilinx project for this part (Please refer to Part D of the Course Reader for the detailed information). This file is required for carrying out automatic simulation.
- You need to add the file "pins.ucf" (the file can be downloaded from [Pins File for Unit 16/17 Lab](#)) to your project, in which the partial pin assignment is stated. Furthermore, You need to assign the following pins to the ports that are unassigned in this pins.ucf file namely,

clock_ext	D9
reset_ext	G12
x_ext	F12

Note: **clock_ext** is the clock you use to manually test the state machine. It is the (red color) button located on the smaller board connected on top of the Xilinx board. **reset_ext** is *SW1* and **x_ext** is *SW0*. These switches are for you to use while testing the state machine. **Your current state** will show up on *LD2* through *LD0*, and **your output** shows up on *LD3*.

- Following the instructions in Lab Part 1/Part D of the Course Reader, implement your code on a Xilinx lab board. Record the output sequences determined experimentally from your circuit under the required input test sequences. **(Make sure to include the following report in your paperwork)**

(1) X = _____
Z = _____
(2) X = _____
Z = _____

For the Automatic testing part, you will make use of the following switches

- (1) BTN2(L13) is the reset button.
- (2) SW7(K13), SW6(K14), SW5(J13), SW4(J14) which you use to specify the problem number who are assigned to. K13 is the Most Significant Bit. For example, if you were assigned 16.2, then your switches would have the value K13 (0), K14 (0), J13 (1) and J14 (0).
- (3) BTN3(L14) which is the **start** button.

These definitions can be seen in the pins.ucf file that you just added to the project. As soon as you press the **start** button, automatic testing takes place. The display of 5555 in your seven segment LED shows a pass.

- Turn in the paperwork and demo your design on the board to the TA. The TAs would be checking for both the automatic and manual testing on the Board.

* **Checklist** for Lab 5.2 submission:

- [Coversheet for Lab 5.2](#)
- Printout of your design code only (**Not** the testbench_modified.vhd)
- The report of the output sequences determined experimentally from the board with their corresponding input sequences

FAQ

Q. What is the *course reader*?

In the past we had a hardcopy of the course reader, which included tutorial on logic design, instructions on using the lab software, etc. Most of this material has been moved to softcopy and is associated with specific labs. Remaining general instructions on DirectVHDL and the FPGA board can be found [here](#).

There may still be references to the physical course reader in some instructions; you can disregard these, and just the the softcopy linked above. (The reader may be referred to as the course manual, supplementary reading in some places.)

Q: Where can I find the information in detail about Flip-Flops?

A: Please refer to p.334-p.336 (11.8 Flip-Flops with Additional Inputs) in the textbook.

Q: I am sure my state table is correct but the checker says "Incorrect". Why?

A: The checker files in the lab computers are for old problems. Since we have modified the problems slightly (to counter plagiarism), they won't work. **We have posted all the new files for test and check on Blackboard System.**

Q: I am not able to implement the state machine using **required no. of gates**.

A: You must have not done a good state assignment. Please refer to Unit 15, page 490, specifically state assignment guidelines 1-3.

Q: What do I need to do during demo?

A: For Lab 5.1, we check the following during demo-

i) We put an auto-checker in your circuit to make sure it's working

ii) We ask you to change the input(s) and clock manually and show the output. If you have answered the preparatory questions, you should know the proper sequence of changing the input and observing the output.

Remember, there is huge penalty for not being able to do it as this shows lack of your basic understanding of sequential circuits.

Q: What if I am confused about the timing diagrams of Mealy sequential circuits (e.g., when and how output is supposed to change based on the change of inputs and the present state, etc)?

A: Please refer to P.407 in Unit 13 of the textbook, where the detailed related information is provided. **Note that this is nontrivial when you are gonna check your timing diagram for Lab 5, and we TAs do care about whether or not you understand it. The demo is not only about the correctness of the output sequence, but also about the timing diagram.**

Q: I am not able to design a Mealy sequential circuit detecting multiple designated sequence (e.g., the output Z should be 1 if the input sequence ends in either 010 or 1001, and Z should be 0 otherwise).

A: Please read through Section 14.2 in Unit 14 on P.435 of the textbook, and Example 2 in Section 14.3 in Unit 14 on P.440 of the textbook.

Q: Common mistake(s)

A: There is one common mistake: not being able to figure out where (in the timing diagram) to place the first bit of the input sequence. (This problem even and naturally exists in Lab 6.) If you read the demo example (including the illustrations) carefully, you'll find that the first bit of input sequence should be put before the first rising edge of the clock - to be consistent, typically 25ns (0ns is of course fine, for instance), while the following testing bits should be placed 25ns after each rising edge of the clock.

Also refer to [General FAQs](#) for submission guidelines.