ISTANBUL TECHNICAL UNIVERSITY

Digital System Design Applications

Experiment VI
FINITE STATE MACHINES

Preliminary

Students should cover sequential circuits and state reduction methods.

Objectives

- Learn design and implementation of finite state machines.
- Learn difference between Mealy and Moore machines.
- Learn state reduction and encoding.

Requirements

Students are expected to be able to

- have knowledge about what sequential circuits are
- know reduction methods

Experiment Report Checklist

Each student is going to prepare his/her own report. Reports should include:

1. **State Reduction**
   - Explain state reduction in sequential circuit.
   - Explain why state reduction is not possible for Fig-1.
2. **State Encoding**
   - Explain how to perform efficient state encoding.
   - Make state encoding for the experiment and explain with details.
3. **Reduction of Combinatorial Parts**
   - Make reduction for input functions of each D flip-flops (Q2, Q1, Q0) and output function of z by using Quine-McCluskety or KarnaughMap.Reduction result with arbitrary state.
   - Think that you will implement combinatorial parts with LUT4. In this case, can you make a different reduction to use the least number of LUT4? Explain in detail.
4. **Creation FSM1 project**
   - Screen shots of verilog code, testbench code, RTL schematic, technology schematic, device layout
   - Simulations(behavioral- post-implementation functional) Screen shots of wave form.
   - Show that what faulty outputs are if any. Under which circumstances the circuit produces these outputs? Explain in the report.
   - Explain prevention method of faulty output.
   - Adding D flip-flop to the z output, how does it effect on circuit?
   - After adding D flip-flop test your circuit again and add screen shots of simulation wave, explain situations of faulty outputs.
   - Delete D flip-flop that you added to the z output in the previous section. Start your circuit from arbitrary states.Is the circuit switch its state to used states or is it stuck in arbitrary states? Explain in your report.
5. **Creation FSM2 project**
   - Add all results which is wanted for FSM1 section to FSM2 section.
6. **Designing of Circuit that detects ’101’ and ’100’**
   - Design a circuit with 1-bit x input and 1-bit z output. This circuit detects ’101’ and’100’ sequences. If one of these sequences comes twice in succession, the output will be 1 and the circuit stays in a state. Design this circuit with always and case. Examine report in Vivado and find that which FSM coding technique is used for the design. Try to design with least number of LUT and D-FF. Try to minimize delays. The best design will be determined and awarded with extra points.

- **Projects and reports are to be done INDIVIDUALLY. High amounts of points will be deducted from similar works.**
- **Reports must be written in a proper manner. Divide your text to sections and subsections if needed, label your figures and connect your sections with proper explanations of your works. Reports filled with imprecisely placed tables and figures, with no verbal explanations in workflow, will not fare well.**
- **Check homeworks section in Ninova for submission dates. There will be two different submissions for project archive folder and a PDF report file.**

CIRCUIT THAT DETECTS FOUR CONSECUTIVE 1 OR 0

Description of Problem

You will design a circuit that has 1-bit x input and 1-bit z output. D-type flip-flops on the FPGA will be used and all flip-flops will be triggered on the rising edge of the clock signal (clk). The circuit will sample the x input at each active clock edge. If four consecutive 1 or 0 come from x input, the output z will be 1. In all other cases, z output will be 0. The state machine to be designed should be a Mealy Machine. Therefore, the output of the circuit will be 1 simultaneously when four consecutive 1 or 0 come from the input. Remember that if the machine was Moore, the output will be 1 after one clock cycle. In Fig. 1, state diagram of this problem is given. Use the state table without reduction given in Fig. 2.

State Encoding

1. Explain in your report how to perform efficient state encoding. done.
2. Make state encoding for the experiment. Show it in your report.
3. If you do not suggest any encoding, use the table given in Fig. 3. The state table to be obtained when the coding in Fig. 3 is used is shown in Fig. 4.

Reduction of Combinatorial Parts

1. Make reduction for input functions of each D flip-flops (Q2, Q1, Q0) and output function of z by using Quine-McCluskety or KarnaughMap.
2. Show your reduction results in the report. Do not forget arbitrary states.
3. Think that you will implement combinatorial parts with LUT4. In this case, can you make a different reduction to use the least number of LUT4? Explain in detail in your report.

Verilog Encoding

1. Create new source as FSM1.v in your project. Give a same name (FSM1) to your module.
2. Combinatorial parts of the circuit will be implemented with assign.
3. always@(posedge clk) will be used for flipflops.
4. Make synthesize and implementation. Show your results in your report. Add schematics and layout to the report.

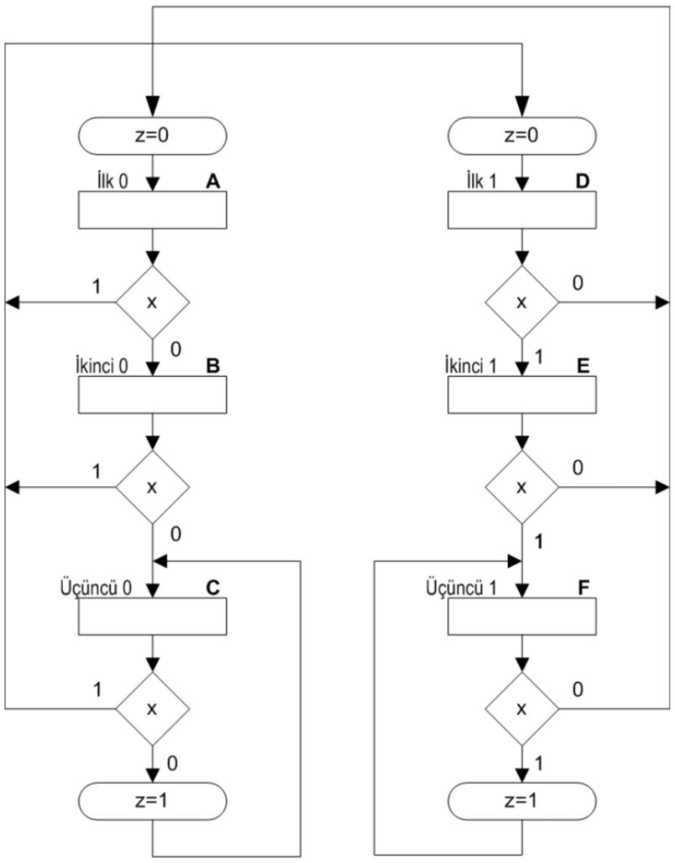
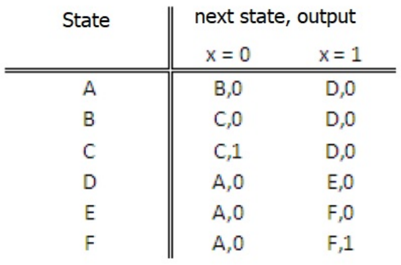
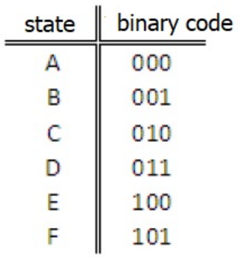


Figure 1: Algorithmic State Diagram



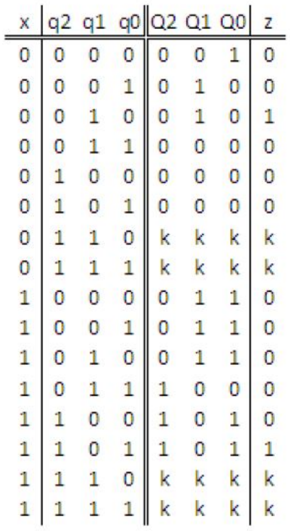
State	next state, output	
	x = 0	x = 1
A	B,0	D,0
B	C,0	D,0
C	C,1	D,0
D	A,0	E,0
E	A,0	F,0
F	A,0	F,1

Figure 2: State Table without any reduction



state	binary code
A	000
B	001
C	010
D	011
E	100
F	101

Figure 3: State Encoding Example



x	q2	q1	q0	Q2	Q1	Q0	z
0	0	0	0	0	0	1	0
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	0	k	k	k	k
0	1	1	1	k	k	k	k
1	0	0	0	0	1	1	0
1	0	0	1	0	1	1	0
1	0	1	0	0	1	1	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	k	k	k	k
1	1	1	1	k	k	k	k

Figure 4: State Table after Encoding

5. x input comes from switch, clk input comes from button. Connect z output to the LED.

Simulation of the Circuit

Write a testbench with given information below:

1. Use as an input: 0100110001110000111100000111111000000111111
2. This input will be used from left to right and sampled at the each clock cycle.
3. Make behavioral and post-ımplementation-functional simulation. Give results and waveform in your report.

Analysis of Faulty Outputs

1. Determine if your circuit is producing faulty 0 or faulty 1.
2. Show that what faulty outputs are if any. Under which circumstances the circuit produces these outputs? Explain in the report.
3. For preventing these faulty outputs, what are the methods? Explain in your report.

Machine Type Changing

1. Add D flipflop to the output z of the circuit. Is the circuit still Mealy Machine? Explain in your report.
2. Test your circuit with the test code you write before.
3. Are faulty outputs still there? Explain in your report.

Stucking in Undesirable States

1. Delete D flipflop that you added to the z output in the previous section. Start your circuit from arbitrary states (i.e you do not use this state).
2. Is the circuit switch its state to used states or is it stuck in arbitrary states? Explain in your report.

DESIGN WITH DIVIDED STATE DIAGRAMS

1. Show that, state diagram in Fig. 5 has same functionality with Fig. 1.
2. Create new source, FSM2.v. Give same name to your module and design the circuit again by applying previous commands for FSM1. Add all results to the your report.

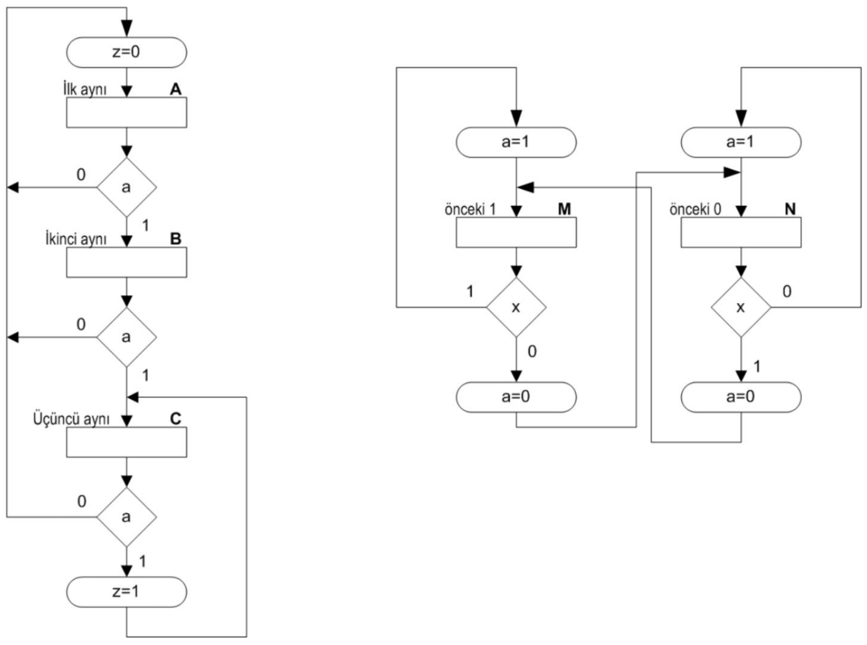


Figure 5: Divided Algorithmic State Table

References:

1. Nexys4DDR Reference Manual
2. Artix-7 Libraries Guide for HDL Designs
3. Constraints Guide
4. Brown&Vrasenic, "Fundementals of Digital Logic with Verilog Design", McGraw-Hill, 1st edition. pp445-521