

## **Developing Lab Exercises for Logic Circuit Design using FPGAs**

### **Mr. Baha Bachnak, The Pennsylvania State University**

Baha Bachnak is an undergraduate student majoring in Electrical Engineering at The Pennsylvania State University-Harrisburg. He is also a Schreyer Honors Scholar and has research interests in applications of Field-Programmable Gate Arrays (FPGAs).

### **Dr. Nashwa Elaraby, Pennsylvania State University, Harrisburg, The Capital College**

Dr. Elaraby is a faculty at Penn State Harrisburg. She received her PhD degree in Electrical and Computer Engineering from Temple University in 2014. She received her B.Sc and M.Sc. in Electrical Engineering from Alexandria University, Egypt.

Her research interests include digital logic design using Field Programmable Gate Arrays for massively parallel data computations, electronic circuit design, and neuronal data processing for Brain Machine Interface applications.

# Developing Lab Exercises for Logic Circuit Design using FPGAs

Baha Bachnak, Nashwa Elaraby

*Department of Electrical Engineering, The Pennsylvania State University-Harrisburg*

## Abstract

An introductory course to digital logic design is a requirement for most electrical and computer engineering (ECE) programs. It is also one of the first technical courses that ECE students are exposed to. Field Programmable Gate Arrays (FPGAs) is a versatile and adaptable technology with many applications ranging from medical image processing to cryptography. By combining an FPGA course and a digital logic design course, students can learn the basics and be introduced to new implementation tools and platforms at the same time. This paper describes a number of academic approaches to incorporate FPGA design in digital design courses and presents a number of laboratory experiments and tutorials that pave the path for designing a sophomore-level four semester credit hour (SCH) course. The results of a survey conducted to gauge student interest of such a course are included as well.

## Introduction

Digital logic design is a topic required by most ECE programs, based on curricula recommendations of IEEE and ABET. In such courses, students are taught how to design digital circuits to produce a set of output signals from a set of inputs through use of logic gates, registers, multiplexers, and other fundamental components. Students are also taught fundamentals such as Boolean algebra, sequential logic, finite state machines, and combinational logic. These skills are required to learn more complex concepts, for example, computer architecture, networking, and electronic systems. Thus, an early exposure to digital logic is necessary for engineering students.

As technology advances, universities and colleges struggle to prepare students for the tools they will use in industry. They also struggle to increase the amount of knowledge their students learn while maintaining an academic plan of four years without increasing the amount of required semester credit hours in order to graduate [1]. Digital logic design is a necessary topic for electrical engineers and computer engineers and FPGA's are an evolving technology with a diverse range of applications. By combining both these important topics into one course, students will be exposed to a great amount of important and useful information without the need to increase the number of credits required to earn their degree.

There exists a fair amount of literature pertaining to the design of a digital logic course incorporating field-programmable gate arrays (FPGA's) and hardware description languages (HDL's). As Carroll describes, FPGA's and solderless breadboards are used as a supplemental learning tool for lectures [1]. The tool does not only reemphasize concepts learned in class, but also exposes students to the physical application of these concepts, as well as satisfy ABET student outcomes. Carroll also outlines a list of topics, which are supplemented by appropriate

laboratory experiment modules. The Altera Quartus II design software was used extensively in simulating designs pertaining to each module, allowing students to modify and verify their designs relatively easily. Each module was then implemented using either a solderless breadboard, for simpler circuits, or an Altera Cyclone II FPGA board, for more complex modules. A center focus of Carrol's course is the "semester long project involving the design, implementation, and documentation of the computer processing unit (CPU) for a basic four-bit digital computer called TRIS (Tiny Reduced Instruction Set Computer)." Including such a project provided students with a clear goal in mind for the course, and set the importance for a clear road map of the course, list of topics, and nature of laboratory experiments [1].

Carroll's approach involving a semester long project also utilizes a similar method to teaching as the "flipped course" method, which is discussed by Yelamarthi and Drake [2]. The flipped course format employs interactive classroom learning activities and encourages open ended solutions and problem solving, in addition to technology based independent learning activities outside the classroom. Research indicates that such an approach to teaching results in students becoming more aware of their learning, making course content, and improving their learning. Research also indicates that in flipped engineering courses, especially upper-division courses, instructors are "able to cover more course content and implement active, problem-based learning activities without sacrificing course content [2]."

By providing his students with take home solderless breadboards, DE1 FPGA's, and a frequent amount of pre-laboratory experiments in preparation to the actual laboratory, Carrol and his team implemented some aspects of a flipped course into his digital logic course. According to results of surveyed students at the end of the course, all 17 students strongly agreed that the take-home parts and DE1 FPGA board were helpful. The entire surveyed group also agreed that the lab previews were beneficial and that they "helped tie fundamentals from lectures to implementations in labs [1]."

According to Pang, the use of Altera Quartus and Verilog gives students the ability to not only modify and verify their circuit designs before implementation on real hardware, but also provides students with an active learning tool [3]. The ability to improve design productivity and accuracy by using modern FPGA technology, allows students to save hardware cost, reduce design time, and actively participate in the laboratory design work [4]. The use of Altera Quartus and Verilog HDL also provides students with a deeper understanding of the functionality of different digital logic circuit blocks.

This paper describes a second-year level, four semester credit hour digital logic course implementing the use of FPGA's at the Pennsylvania State University, Capital College. The course is required for electrical engineering majors and computer engineering majors, as well as a number of minor programs.

This paper will discuss laboratory experiments pertaining to a digital logic course, designed in order to incorporate the concepts and methods of teaching stated above, a course that will teach digital logic topics and introduce FPGA's and Verilog.

## **Course Description and Structure**

The course will consist of an introduction to digital logic aided by the use of FPGAs and Verilog Hardware Description Language (HDL). Lectures and classroom time will focus on the digital logic aspect of the course while laboratory experiments will reinforce concepts learned in the classroom through applications based on FPGAs. In this paper, two laboratory experiments are described: one experiment introducing students to the functionality of a full adder circuit and another experiment introducing students to the functionality of a Finite State Machine (FSM). Both of these laboratory experiments have a physical (breadboard circuit synthesis) and coding (Modelsim functional simulation and implementation on an FPGA) aspect to them. Additionally, both experiments have the potential to be easily modified and additional activities for each of them have been included in the handouts. Possible solutions to the laboratory experiments have also been included for instructor reference. Please see the Appendix A and B for the laboratory handouts.

## **Evaluation Tools and Results**

Evaluation of the laboratory experiments designed will be conducted in the future. This is expected to be done through a survey of students enrolled in the course, or students who will experience a simplified laboratory experiment while enrolled in a standard introduction to digital logic class. Evaluation may also be done in the form of a final project, incorporating concepts learned throughout the entire course into one project or design, allowing the instructor to gauge students' understanding of the coursework and coding processes.

The interest among students in such a course, however, has been evaluated. This was done through a survey of 43 current students who had gone through or are currently enrolled in a traditional digital logic course. Of these 43 students, 4 were currently enrolled in the course and 40 had taken it at The Pennsylvania State University Harrisburg campus. A copy of this survey can be found in the Appendix C section at the end of this paper.

Students were first provided with the following information regarding an FPGA:

*A Field Programmable Gate Array (FPGA) is an integrated circuit that consists of configurable logic blocks that can be assigned specific operations and connected together using a hardware description language. FPGAs provide the high speed and parallel processing provided by ASICs along with the flexibility given by software. FPGAs can be reconfigured while in field operation and have many applications ranging from medical image processing to cryptography.*

When asked about wanting to learn more about FPGAs, 72.1% of respondents expressed a positive interest, while only 18.7% expressed that they are not interested.

1. Select from the following the expression that best describes you:

43 responses



Figure 1: Survey Response

Students were then provided with a brief description of Verilog HDL:

*Verilog HDL is a hardware description language (HDL) most commonly used in the design and simulation of digital circuits.*

Similarly, to the previous prompt, a majority of respondents expressed positive interest in learning about Verilog HDL (76.8%), while only 18.6% expressed no interest.

2. Select from the following the expression that best describes you:

43 responses

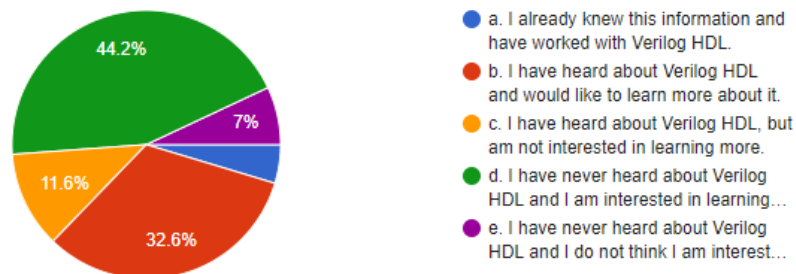


Figure 2: Survey Response

Surveyed individuals were then asked questions regarding laboratory experiments and the benefits of simulations in such experiments. The vast majority (83.2%) of students surveyed agreed that simulating laboratory experiments prior to performing them would be beneficial. By simulating experiments and digital logic circuits using Altera Quartus and an FPGA, students are able to visualize the behavior of the circuit and the intended results before assembly. This method also allows to students to think critically to which component is needed, the role of each component used, and how to translate the behavior of the circuit into code.

### 3. Simulating laboratory experiments prior to performing them would be beneficial

43 responses

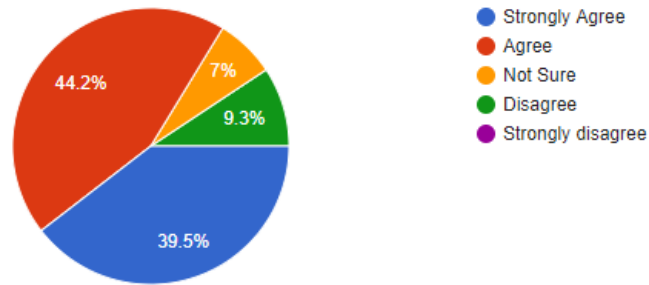


Figure 3: Survey Response

When asked about the benefits of simulating circuits using PSPICE, students were not as enthusiastic, with only 69.8% responding positively. After speaking with students, it was determined that this decrease in enthusiasm is because of the lack of independency and instructiveness of such simulations; using PSPICE, students all follow the same steps to reach their results. There is no independency or feeling of self-fulfillment when students simulate a circuit in PSPICE. Using Altera Quartus and Verilog HDL, however, students will need to understand the behavior of the circuit and translate this information into Verilog HDL code. They are then able to simulate the design, similarly to PSPICE, as well as physically interact with the FPGA board and visualize the onboard LEDs react to the different input values.

### 4. PSpice simulations of logic design circuits would be beneficial in learning about them.

43 responses

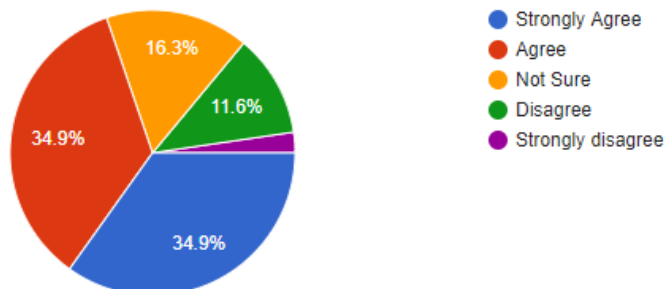


Figure 4: Survey Response

Students did agree on the potential benefit of utilizing Verilog HDL simulations and FPGA prototyping in learning more about digital logic circuits. Although 69.8% of respondents agreed that using these two methods would be beneficial, many respondents who answered “Not Sure” were unsure about the ease of use of these tools and whether they would be efficient in the classroom. These respondents, however, did voice that there is a large potential for positive learning through this combination.

**5. A combination of FPGA implementation and Multisim functional simulation of the Verilog code would be beneficial in simulating digital logic circuits and learning more about them.**

43 responses

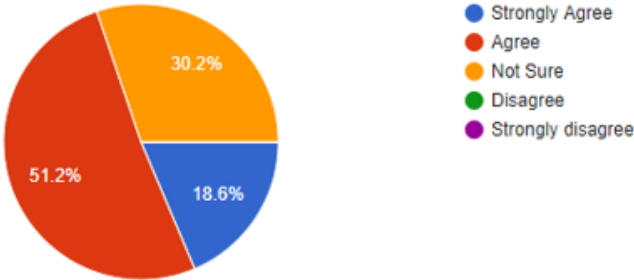


Figure 5: Survey Response

When gauged on the interest of an online course, the large majority of students (90.7%) expressed that they would like to experience both hardware and software-based learning. After speaking with respondents, it was clear that many saw the importance of both hands-on experience as well as coding. Many students expressed that the ability to design, simulate, prototype, and physically assemble a circuit is necessary in becoming an electrical engineer and is seen as crucial to employers.

## 6. Would you like to replace physical labs by simulations only, or both are important?

43 responses

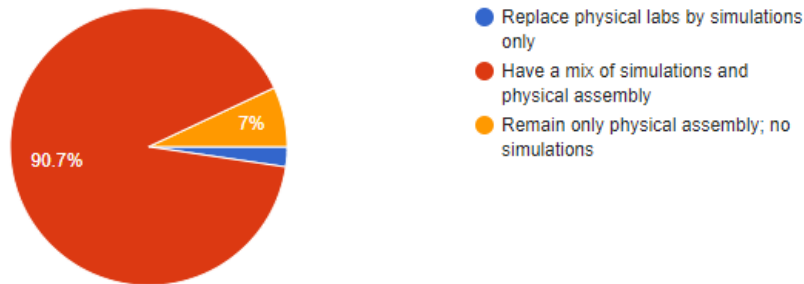


Figure 6: Survey Response

### Conclusion and Future Plans

In conclusion, this research project provided me with a greater understanding of FPGA's and Verilog. Through my time investigating the FPGA's use in aiding a course in digital logic, I have been able to learn aspects that go into play when designing a course and laboratory experiments.

Future work would include designing additional laboratory experiments, lectures, and evaluation of the course. The coding and computer simulation aspect of the course opens up possibilities for application as an online course, which I have experienced as I learned most of fundamentals of the FPGA and Verilog through guided research and independent study.

We have concluded that in order to make such a course successful/effective, a number of practices should be implemented into the course. As mentioned in the introduction, a flipped classroom format engages students and make the course more effective. By implementing online modules and practice coding assignments, such as pre-laboratory exercises, the benefits of a flipped classroom can be achieved. As a student, my experience has also taught me that practicing code frequently and continuously helps students in mastering the skill.

This course has a wide range for potential, as FPGA's present a continually growing field with a wide scope of applications. The course also combines the necessary introduction to digital logic class, making a credit efficient option to students curious about learning more about the Verilog computer language or FPGA's.

### Acknowledgements

The authors would like to thanks the Pennsylvania State University Multi-Campus Research Experience for Undergraduates and Dr. Jack Sampson (Assistant Professor of Computer Science and Engineering, The Pennsylvania State University) for their support throughout this project.



## References

- [1] B. Carroll, S. Gieser, and D. Levine, “A hierarchical project-based introduction to digital logic design course,” presented at the 121st ASEE Annual Conference and Exposition: 360 Degrees of Engineering Education, Indianapolis, IN, United states, 2014.
- [2] K. Yelamarthi and E. Drake, “A Flipped First-Year Digital Circuits Course for Engineering and Technology Students,” *IEEE Transactions on Education*, vol. 58, no. 3, pp. 179–186, Aug. 2015.
- [3] J. Pang, “Active Learning in the Introduction to Digital Logic Design Laboratory Course,” p. 10, 2015.
- [4] D. Bañeres, R. Clarisó, J. Jorba, and M. Serra, “Experiences in Digital Circuit Design Courses: A Self-Study Platform for Learning Support,” *IEEE Transactions on Learning Technologies*, vol. 7, no. 4, pp. 360–374, Oct. 2014.

## Appendix A

### Binary Full Adders and Subtractors

Objective: The binary half adder and binary half subtractor perform operations on single digits. To perform such operations on numbers with two or more digits, however, the full adder or full subtractor is needed. This laboratory experiment will provide participants with the understanding of how to construct circuits capable of these common operations.

Based on an existing Penn State Course

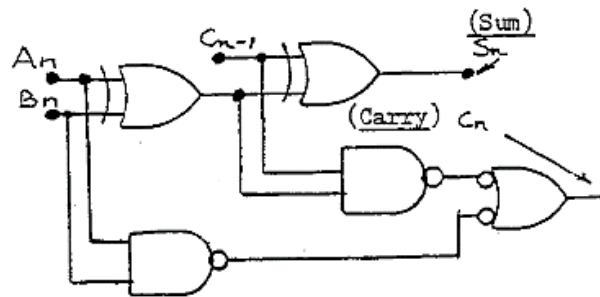


Figure 1: Full Adder Circuit

#### Laboratory Experiment Preparation:

##### Part I:

Create a Binary Full Adder Truth Table with columns identifying sum and carry values for the function  $A + B$ . Figure 1 shows a full adder using three NANDS, two X-ORs, and an OR.

#### Methods and Procedure:

##### Part I:

Using Altera Quartus create a Verilog HDL project named “BinaryFullAdder.v” and simulate the Binary Full Adder using Boolean algebra. Upload the file to an FPGA and simulate the behavior of the circuit physically on the FPGA board using the onboard switches and LED’s. Use the prepared truth tables to verify your results.

##### Part II:

Using a breadboard, construct the circuit shown in Figure 1 (Full Adder), remembering to connect terminals  $C_n$  and  $C_{n-1}$ . Include LED’s to indicate a “1” (high) when on and switches to control inputs. Verify the functioning of your circuit by testing all eight possible input combinations of  $A_n$ ,  $B_n$ , and  $C_{n-1}$ .

Conclusion and Discussion:

Explain what you learned about the functionality and limitations of the adder, including its carry delays. Discuss ways these limitations may be minimized.

Possible Solution

Laboratory Experiment Preparation:

$C_{n-1}$	$A_n$	$B_n$	$C_n$	$S_n$
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Table 1: Full Adder Truth Table

Calculations:

LS00 (NAND Gate):  $V = .8 \text{ V}$  (Maximum value) and  $I = .4 \text{ mA}$  (Maximum value)

$$\rightarrow \text{So } R = .8 \text{ V} / .4 \text{ mA} = 2 \text{ K}\Omega;$$

Use a  $1 \text{ K}\Omega$

LS86 (NOR Gate):  $V = .8 \text{ V}$  (Maximum value) and  $I = 1.2 \text{ mA}$  (Maximum value)

$$\rightarrow \text{So } R = .8 \text{ V} / 1.2 \text{ mA} = 667 \Omega;$$

Use a  $330 \Omega$

Pull Down Method: Resistor =  $(V_{CC} - V_{IH}) / 2(I_{IH} - \# \text{ of gates}) = 1 \text{ K}\Omega$

## Altera Quartus:

```

module Lab1B (
    SW,
    LEDR
);
    input [9:0] SW;
    output [9:0] LEDR;

    wire [3:0] A, B;
    wire [4:0] M;
    wire Cin;

    assign A = SW [3:0];
    assign B = SW [7:4];
    assign Cin = SW [9];

    assign LEDR [4:0] = M;
    assign LEDR [8:5] = 4'b0000;

    wire C0, C1, C2, C3;

    OneBitAdder ONE (A[0], B[0], Cin, M[0], C0);
    OneBitAdder TWO (A[1], B[1], C0, M[1], C1);
    OneBitAdder THREE (A[2], B[2], C1, M[2], C2);
    OneBitAdder FOUR (A[3], B[3], C2, M[3], M[4]);

endmodule

module OneBitAdder (
    X,
    Y,
    CarryIn,
    Sum,
    Carryout
);
    // Module that processes a single one bit adding function and outputs a Sum value and Carry-out value
    input X, Y, CarryIn;
    output Sum, Carryout;

    assign carryout = (X & Y) | (X & CarryIn) | (Y & CarryIn); // Boolean expressions for the Carryout and Sum values in Sum-of-Product form
    assign sum = (~X & ~Y & carryIn) | (~X & Y & ~carryIn) | (X & Y & carryIn) | (X & ~Y & ~carryIn);

endmodule

```

Figure 2: Verilog HDL Code for a Four Bit Adder

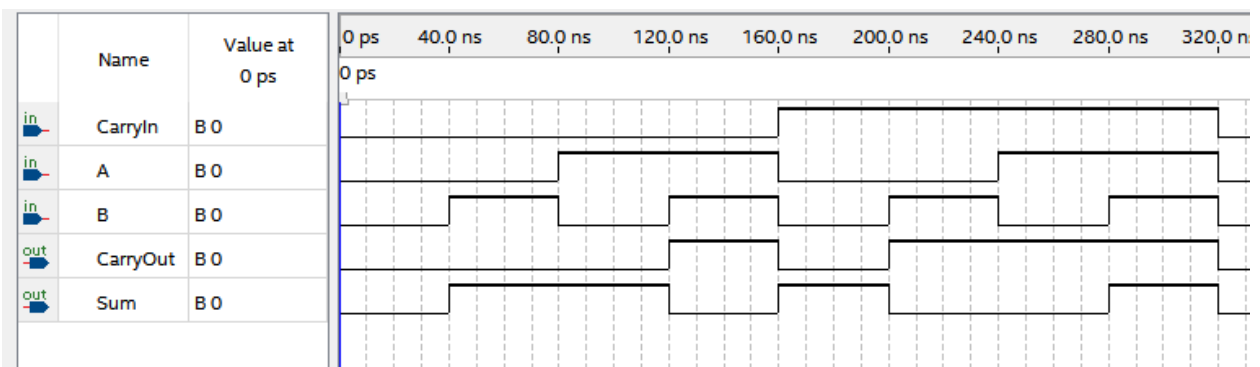


Figure 3: Waveform Timing Diagram displaying results of the Verilog HDL code for an individual One Bit Adder and confirming the prepared truth tables

## Appendix B

### Finite State Machine Design Laboratory

Objective: This experiment introduces the student to the use of a State Diagram in the logical design of sequential circuits and the process of designing and simulating a Finite State Machine using clocked signals on an FPGA board.

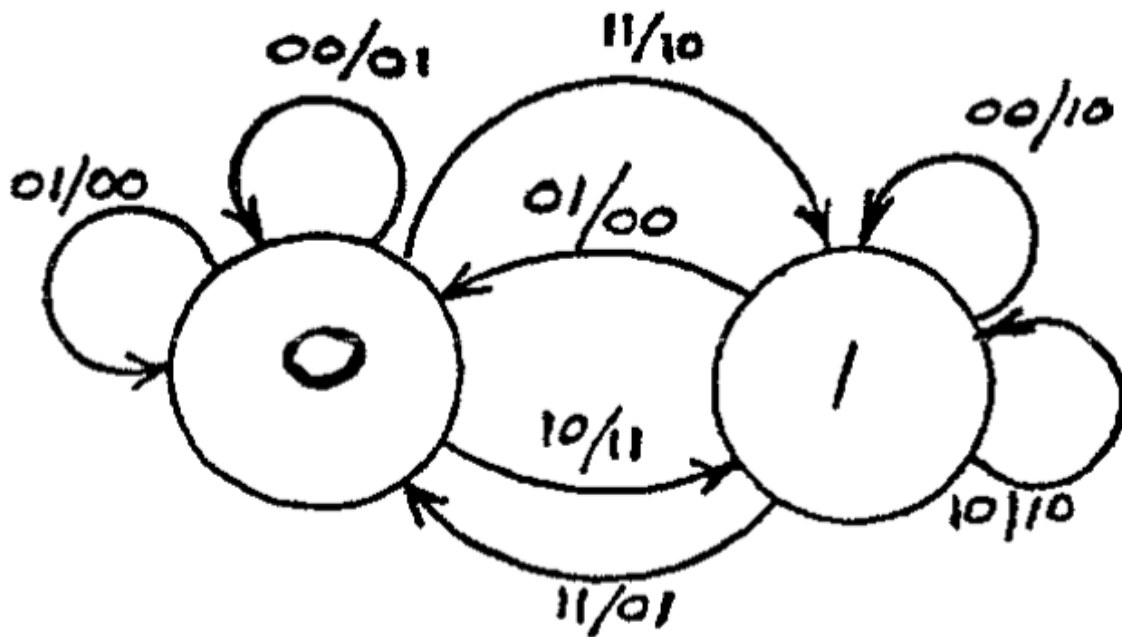


Figure 1: State Diagram with two states: Q = 0 and Q = 1

#### Laboratory Experiment Preparation:

Using the state diagram provided in Figure 1, construct the transition table for a JK flip flop and design logic that us necessary to derive the two outputs  $Y_1$  and  $Y_0$  .

#### Methods and Procedure:

##### Part I:

Create a Verilog HDL file named "FiniteStateMachine" and simulate process shown in Figure 1 on an FPGA board.

Part II:

Construct the circuit designed in the laboratory preparation using LED's to display the two outputs  $Y_1$  and  $Y_0$  .

Conclusion and Discussion:

How would the experiment differ if a Moore state machine was designed? How would other flip flops be incorporated in place of the JK flip flop?

## Possible Solution

### Laboratory Experiment Preparation:

$Q^+, Y_1, Y_0$

Q (Present State)	$X_1, X_2 = 0, 0$	0, 1	1, 1	1, 0
0	0, 0, 1	0, 0, 0	1, 1, 0	1, 1, 1
1	1, 1, 0	0, 0, 0	0, 0, 1	1, 1, 0

Table 1: Transition Table of the state diagram showing the next state and two outputs corresponding with each combination of present state and input values

	Q	Q <sup>+</sup>	J	K
→	0	0	0	–
	0	1	1	–
	1	0	–	1
	1	1	–	0

Table 2: Transition Table for a JK Flip Flop

$J = X_1$

	$X$			
	0	0	1	1
Q	–	–	–	–

Table 3: Karnaugh Map for the J port of the JK Flip Flop

$K = X_0$

	–	–	–	–
Q	0	1	1	0


Table 4: Karnaugh Map for the K port of the JK Flip Flop

	0	0	1	1
Q	1	0	0	1

$$Y_1 = X_1\bar{Q} + \bar{X}_0Q$$

Table 5: Karnaugh Map for output  $Y_1$

	1	0	0	1
Q	0	0	1	0


  
 X

$$Y_0 = \bar{X}_0\bar{Q} + X_0X_1Q$$

Table 6: Karnaugh Map for output  $Y_0$



## Altera Quartus:

```
1 module FiniteStateMachine (
2     SW,
3     LEDR,
4     clk
5 );
6
7 input [9:0] SW;           // Input: switch
8 output [9:0] LEDR;       // Output: LED
9 input clk;              // The clock signal is an input
10
11 parameter s0= 1'b0,     // Create states 0 and 1
12            s1= 1'b1;
13
14 wire x0, x1, reset, clk; // Create wires for output values
15 reg z0, z1, light;      // Create registers to hold values for output values
16 reg state, next_state; // Create placeholders for the current state and next state
17
18 assign x0 = SW[0];      // Assign wires to their respective switch or LED
19 assign x1 = SW[1];
20 assign reset = SW[9];
21 assign PIN_AF14 = clk; // Set clock to the onboard 50KHz clock signal or SW [9]
22 assign LEDR [0] = z0;
23 assign LEDR [1] = z1;
24 assign LEDR [9] = light;
25 assign LEDR [7:2] = 6'b000000; // Tie unused LED's low
26
27 always @(posedge clk) // Start process when there is a change in the clock signal
28     if (reset == 0)
29         state = s0;
30     else
31         state = next_state;
32
33 always @(state, x0, x1) // Start when change in state or x0 or x1
34     begin
35         case(state) // Run for each transition and output respective values
36             s0:
37                 if (x0 == 0 & x1 == 0) begin next_state = s0 ; z0 = 0 ; z1 = 1 ; light = 0; end
38                 else if (x0 == 0 & x1 == 1) begin next_state = s0 ; z0 = 0 ; z1 = 0 ; light = 0; end
39                 else if (x0 == 1 & x1 == 1) begin next_state = s1 ; z0 = 1 ; z1 = 0 ; light = 0; end
40                 else if (x0 == 1 & x1 == 0) begin next_state = s1 ; z0 = 1 ; z1 = 1 ; light = 0; end
41
42             s1:
43                 if (x0 == 0 & x1 == 0) begin next_state = s1 ; z0 = 1 ; z1 = 0 ; light = 1; end
44                 else if (x0 == 0 & x1 == 1) begin next_state = s0 ; z0 = 0 ; z1 = 0 ; light = 1; end
45                 else if (x0 == 1 & x1 == 1) begin next_state = s0 ; z0 = 0 ; z1 = 1 ; light = 1; end
46         endcase
47     end
48 endmodule
49
50
```

**Appendix C****CMPEN 270/271/275**  
Research Survey

Are you currently enrolled in CMPEN 270/275 (Lab component): Yes No

Campus or Institution where 270/275 was taken (Lab component): \_\_\_\_\_

Semester during which 270/275 was taken (Ex: Fall 17) (Lab component): \_\_\_\_\_

---

---

A Field Programmable Gate Array (FPGA) is an integrated circuit that consists of configurable logic blocks that can be assigned specific operations and connected together using a hardware description language. FPGAs provide the high speed and parallel processing provided by ASICs along with the flexibility given by software. FPGAs can be reconfigured while in field operation and have many applications ranging from medical image processing to cryptography.

Please complete the following questions:

1. Select from the following the expression that best describes you:
  - a. I already knew this information and have worked on FPGA.
  - b. I have heard about FPGA and would like to learn more about it.
  - c. I have heard about FPGA, but am not interested in learning more.
  - d. I have never heard about FPGA and I am interested in learning more.
  - e. I have never heard about FPGA and I do not think I am interested in this field.

Verilog HDL is a hardware description language (HDL) most commonly used in the design and simulation of digital circuits.

2. I know how to code in Verilog HDL:
  - a. I already knew this information and have worked with Verilog HDL.
  - b. I have heard about Verilog HDL and would like to learn more about it.
  - c. I have heard about Verilog HDL, but am not interested in learning more.
  - d. I have never heard about Verilog HDL and I am interested in learning more.
  - e. I have never heard about Verilog HDL and I do not think I am interested in this field.

3. Simulating laboratory experiments prior to performing them would be beneficial  
\_\_Strongly agree    \_\_Agree    \_\_Not sure    \_\_Disagree    \_\_Strongly disagree
4. PSpice simulations of logic design circuits would be beneficial in learning about them.  
\_\_Strongly agree    \_\_Agree    \_\_Not sure    \_\_Disagree    \_\_Strongly disagree
5. A combination of FPGA and Verilog simulation would be beneficial in simulating digital logic circuits and learning more about them.  
\_\_Strongly agree    \_\_Agree    \_\_Not sure    \_\_Disagree    \_\_Strongly disagree
6. Would you like to replace physical labs by simulations only, or both are important?
- a. Replace physical labs by simulations only
  - b. Have a mix of simulations and physical assembly
  - c. Remain only physical assembly; no simulations

Please use the following space to write any additional information you would like to share with us regarding this course, FPGAs, or Verilog HDL: