

# EXPERIMENT

# 6

# TEAM PROJECT

VERSION U06

In this experiment, you will perform a system design and test from start to finish. After completing this experiment, you should be able to:

- 1) Specify a simple digital system,
- 2) Separate the system into one or more datapaths and controls,
- 3) Design the datapath(s) for the system,
- 4) Design the control(s) for the system,
- 5) Integrate the datapath(s) and control(s) into an overall system,
- 6) Devise a thorough test for the system,
- 7) Implement the system using an FPGA and additional components, and
- 8) Test and debug the system..

## 6-1 INTRODUCTION

This experiment is to consist of the design, implementation, and test of a system of your lab team's own design. It is very important that your project can be completed in the design time and laboratory time available.

**You are required to have your initial project idea informally reviewed by your instructor before you invest design time in it to make sure that it is appropriate as a project.**

## PROJECT SUGGESTIONS

If you have no idea what to do for such a project, the following are some suggestions. Note that the complexity of most of these is only average or below, which will impact your grade, as noted later in this document.

### 4-BIT SERIAL ADDER

Must include two shift registers, one for the first operand and the result and the other for the second operand. Operand loading must be provided. After operands are loaded, addition must be automatic -- it may be manually clocked, but only a sufficient number of clock pulses are to be provided to complete an operation; they are not to be manually counted. Also, the carry is not to be reset manually before or after each addition, but only at power-up.

### 4-BIT SERIAL ALU

Must include two shift registers, one for the first operand and the result and the other for the second operand. The ALU must perform eight distinct operations including both 2's complement addition and subtraction. Operand loading must be provided. The ALU may be manually clocked and controlled.

### 4-BIT MULTIPLIER

Must be sequential using the shift and add principle with 4-bit operands and 8-bit result. Operand loading must be provided. Operation should be automatic after the operand entry is completed. May be manually clocked, but only

the required number of multiply cycles is to be provided; these cycles are not to be manually counted. When the multiply is completed, the result should be held no matter how many clocks are applied. It should only be destroyed if new operands are loaded or a new multiply is started.

#### **4-BIT DIVIDER**

Must use shift and “add” principle with 4-bit divisor and 8-bit dividend. Operand loading must be provided. Clocking and counting of division steps may be manual, but control such as the decision whether to subtract the divisor must be automated.

#### **VARIABLE-SPEED CHASER LEDS**

The bar LEDs should light in sequence from left to right or right to left at rates such that they appear to move. Four different rates should be digitally controlled by setting switches. The direction should be changeable by a switch. The clock frequency into the circuit is to be fixed.

#### **ASYNCHRONOUS COMBINATION LOCK**

This combination lock has a minimum sequence of four two-bit input symbol as the combination and appears to the user *as if it is an asynchronous circuit*. Actually, it is a *synchronous* circuit with a fast clock and synchronization of the user inputs. For a given input combination, the circuit goes to a state and cycles there until the input changes to a new symbol; thus, the combination cannot contain consecutive appearances of the same symbol. The lock is locked by using an asynchronous RESET.

#### **THUNDERBIRD TAILLIGHTS**

Use four bar LEDs for each tail-light. Two on each side should light up for night use (Switch), all four on each side should light up for brake use (Pushbutton), and all four should “chase” outward (see above) on respective left or right side for left or right turn signals (Switches). If the brakes are activated at the same time as a turn signal, the brake pattern and the turn signal chase should alternately be displayed. For emergency use (Switch), all eight LEDs should flash at a suitable rate.

#### **LED PING-PONG**

This is a two-paddle game in which the ping-pong ball is modeled by sequential illumination of the LEDs in two directions. The paddles are pushbuttons which must be pressed at just the right time to return the ball. Doing this well is one of the more difficult projects.

#### **CRAPS FOR A SINGLE PLAYER**

The random rolling of two die (1 through 6) is to be displayed on two seven-segment LED displays. In addition, a sequential circuit which implements the game rules and scores the game (WIN, LOSE, CONTINUE) automatically is to be provided. To obtain the game rules, enter the phrase Rules for Craps into a search engine such as google ([www.google.com](http://www.google.com)).

## DESIGN REQUIREMENTS/HINTS

1. Use the prior experiments as a guide in using the Quartus tools.
2. Import the pin assignments as done before to specify the pin locations.

**ASSIGNMENT OF PIN LOCATIONS:** Because of all the fixed connections to the FPGA pins on the board, it is vital that pins always be assigned locations to prevent damage to the board. The location for a pin can be selected on the following basis:

If a pin is an **OUTPUT** of the FPGA, assign it to:

- a) an FPGA pin that is connected to the input of a component essential for your testing, such as the 10 bits of the bar LED display, the 7 segments of the four 7-segment LED digit displays, or
- b) to a pin that **is not** attached to the **output** of any component or **is** attached to the output of a component known to have its outputs in the high-impedance state (such as RB0-7 when in Analog Output mode).

If the pin is an **INPUT** to the FPGA, assign it to:

- a) an FPGA pin that is connected to the output of a component essential for your system operation or testing, such as the pushbuttons, the DIP switches, or
- b) to a pin that **is not** attached to the **output** of any component or **is** attached to the output of a component known to have its outputs in the high-impedance state by fixing one of its input values (such as the RB0-7 when in Analog Output mode).

3. Be sure to remove or clear all entries from all schematic sheets that contain unused or partially designed circuitry since this will be included in your netlist and may get “connected by naming” to produce a very wrong design.
4. Synchronize all asynchronous inputs to your circuit as necessary based on circuit function. For example, an ENTER input needs to be synchronized, but assuming the DATA on switches has been set up before ENTER is pushed, the DATA bits need not be synchronized since they will not be changing at the time of entry.

## USING ANALOG INPUT AND/OR OUTPUT

The FPGA board has the additional capability to take analog signals and convert them to digital data, or to create an analog output from a digital value. These operations all use the eight FPGA pins designated as RB0-7. The two-position DIP switch S2 controls the operation of the analog input/output modes of the FPGA board. The available modes of operation are:

S2-2	S2-1							
OFF	OFF	<p><b>Analog Output (8-bit)</b></p> <p>FPGA pins RB0-7 should be configured as outputs. The value present on RB0-7 will be converted to an analog signal (available at the Analog Output BNC connector). The relationship of the digital value to the analog output voltage in general is</p> $V_{OUT} = (RB7:0 / 256) \times 2V$ <p>For example,</p> <table style="margin-left: 40px;"> <tr> <td>RB7:0 = 0000 0000<sub>2</sub> = 0<sub>10</sub></td> <td>→ Analog Output = 0.0 V</td> </tr> <tr> <td>RB7:0 = 1000 0000<sub>2</sub> = 128<sub>10</sub></td> <td>→ Analog Output = 1.0V</td> </tr> <tr> <td>RB7:0 = 1111 1111<sub>2</sub> = 255<sub>10</sub></td> <td>→ Analog Output ≈ 2.0V</td> </tr> </table> <p>Note that the analog output is updated at a 2kHz rate, so you should not change the value on RB0-7 any faster than that.</p>	RB7:0 = 0000 0000 <sub>2</sub> = 0 <sub>10</sub>	→ Analog Output = 0.0 V	RB7:0 = 1000 0000 <sub>2</sub> = 128 <sub>10</sub>	→ Analog Output = 1.0V	RB7:0 = 1111 1111 <sub>2</sub> = 255 <sub>10</sub>	→ Analog Output ≈ 2.0V
RB7:0 = 0000 0000 <sub>2</sub> = 0 <sub>10</sub>	→ Analog Output = 0.0 V							
RB7:0 = 1000 0000 <sub>2</sub> = 128 <sub>10</sub>	→ Analog Output = 1.0V							
RB7:0 = 1111 1111 <sub>2</sub> = 255 <sub>10</sub>	→ Analog Output ≈ 2.0V							

S2-2	S2-1	
OFF	ON	<p><b>1kHz Count</b></p> <p>FPGA pins RB0-7 should be configured as inputs. RB7:0 will be an 8-bit count incrementing at 1kHz, so <math>f_{RB7} = 1\text{kHz}</math>, <math>f_{RB6} = 500\text{Hz}</math>, and <math>f_{RB0} \approx 4\text{Hz}</math>.</p>
ON	OFF	<p><b>Analog Input (8-bit)</b></p> <p>FPGA pins RB0-7 should be configured as inputs. RB7:0 will be an 8-bit value related to the voltage at the Analog Input BNC connector as</p> $RB7:0 = (V_{IN} / 3.3V) * 256$ <p>Of course, the value at RB7:0 will be limited to the decimal range 0-255. For example,</p> $VIN = 0V \quad \rightarrow RB7:0 = 0000\ 0000_2 = 0_{10}$ $VIN = 1.65V \quad \rightarrow RB7:0 = 1000\ 0000_2 = 128_{10}$ $VIN \geq 3.3V \quad \rightarrow RB7:0 = 1111\ 1111_2 = 255_{10}$ <p>Note that the analog input is sampled at a 2kHz rate, so the analog input signal should have a bandwidth less than 1kHz to avoid aliasing.</p> <p><b>IMPORTANT:</b> The input signal applied to the Analog Input BNC connector <b>MUST</b> be limited to the range 0V – 3.3V.</p>
ON	ON	<p><b>Simultaneous 4-bit Analog Input/Output</b></p> <p>FPGA pins RB0-3 should be configured as outputs. The value present on RB0-3 will be converted to an analog signal (available at the Analog Output BNC connector). The relationship of the analog output digital input in general is</p> $(RB3:0 / 16) \times 2V$ <p>For example,</p> $RB3:0 = 0000b = 0_{10} \quad \rightarrow \text{Analog Output} = 0.0V$ $RB3:0 = 1000b = 8_{10} \quad \rightarrow \text{Analog Output} = 1.0V$ $RB3:0 = 1111b = 15_{10} \quad \rightarrow \text{Analog Output} = 1.9V$ <p>FPGA pins RB4-7 should be configured as inputs. RB7:4 will be an 4-bit value related to the voltage at the Analog Input BNC connector as</p> $RB7:4 = (V_{IN} / 3.3V) * 16$ <p>Of course, the value at RB7:4 will be limited to the decimal range 0-15. For example,</p> $VIN = 0V \quad \rightarrow RB7:4 = 0000_2 = 0_{10}$ $VIN = 1.65V \quad \rightarrow RB7:4 = 1000_2 = 8_{10}$ $VIN \geq 3.09V \quad \rightarrow RB7:4 = 1111_2 = 15_{10}$ <p>Note that the analog input is sampled at a 2kHz rate, so the analog input signal should have a bandwidth less than 1kHz to avoid aliasing. The analog output is updated at a 2kHz rate, so you should not change the value on RB0-7 any faster than that.</p> <p><b>IMPORTANT:</b> The input signal applied to the Analog Input BNC connector <b>MUST</b> be limited to the range 0V – 3.3V.</p>

## 6-2 PRELAB

- 1) Write a clear, complete specification for the function of your project system. You may use formal structures such as truth tables, state diagrams, ASMs, or register transfer language statements as necessary.
- 2) Partition the design and allocate parts of the detailed design to team members,
- 3) Perform the design and enter the schematics.
- 4) Devise a test plan for your design that will thoroughly test its function.
- 5) Functionally simulate the design and debug it.
- 6) Carry the design through implementation, looking at the assignments to be sure that the pins are in fact assigned as you expected.
- 7) Perform a timing simulation if possible. (This is to insure that implementation is producing a working system.)

**Submit the following single prelab for the team:**

- a. Your system specification,
- b. All schematics for your system,
- c. Your test plan,
- d. All functional simulation results, and
- e. A listing of the contributions of each team member to the project.

## 6-3 LABORATORY WORK

- 1) Attach the necessary instrumentation to demonstrate your system.
- 2) Power up the board and download your design file.
- 3) Apply your test plan. If your circuit does not work correctly, fix it.

**CHECKPOINT: Demonstrate your final working project to your instructor.**

## 6-4 GRADING CRITERIA AND POSTLAB REPORT

Your project will be graded based primarily on three criteria – functionality, complexity, and implementation (how the logic is actually constructed). Your demonstration should show that your design handles the expected input cases and performs correctly. Your instructor may also apply unexpected (but reasonable) inputs in order to test it further. Both team members can also expect to be asked to explain how parts of the circuit work.

A fully functional project of ‘normal’ complexity that is implemented in a reasonably efficient manner can be expected to receive a grade of 80%. Deductions will be made for less than full functionality, or an inability of either team member to fully explain how the design works. Project of higher complexity will receive a correspondingly higher score for full functionality, while a very simple yet fully functional project will receive a lower score. Similarly, a poor (inefficient) implementation of the logic will result in a deduction, while a very good implementation will result in an increased score. If your team has a project that is unusually complex or very well implemented, be sure to use the demonstration as an opportunity to ‘educate’ your instructor on the finer points of your design!

**No postlab report is required, but you must complete and turn in the assessment form on the next page.**



**ECE 351 – Confidential Team Assessment**

Name: \_\_\_\_\_

Rank as a percentage the relative contributions of your team's members to the success of your project. (The total must be 100%.) Also, identify each person's major responsibilities and write any comments you may have in the spaces below each name to explain your assessment.

Team Member

Percent of Effort

**you** \_\_\_\_\_

\_\_\_\_\_

---

---

---

---

---

---

---

---

\_\_\_\_\_

\_\_\_\_\_

---

---

---

---

---

---

---

---