**California State University, Fresno   
Lyles College of Engineering   
Electrical and Computer Engineering Department**   
**TECHNICAL REPORT**   
   
Assignment: Number 5  
Experiment Title: HW/SW Co-Design of an Embedded System on FPGA  
Course Title: ECE 178 (Embedded Systems)   
Instructor: Dr. Reza Raeisi   
   
 Prepared by: Anthony Herrick   
 Student ID #300144976  
Date Submitted: 10/09/2022

**INSTRUCTOR SECTION**   
   
Comments: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   
 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   
 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   
 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   
 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   
 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   
 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   
   
Final Grade: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TABLE OF CONTENTS

[Objective 4](#_Toc116242898)

[Hardware Requirements 4](#_Toc116242899)

[Software Requirements 4](#_Toc116242900)

[Background 4](#_Toc116242901)

[Project Overview 7](#_Toc116242902)

[Project Procedure 8](#_Toc116242903)

[Conclusion 30](#_Toc116242904)

[Appendix A (Part 1 Code) 31](#_Toc116242905)

[Appendix B (Part 2 Code) 33](#_Toc116242906)

[Appendix C (Part 3 Code) 36](#_Toc116242907)

[Appendix D (Part 4 Code) 37](#_Toc116242908)

[Appendix E (Part 5 Code) 40](#_Toc116242909)

LIST OF FIGURES

[1. Soft Core Embedded System Design 5](#_Toc116242871)

[2. 7-Segment Display Address Spaces 6](#_Toc116242872)

[3. 7-Segment Display Breakdown 7](#_Toc116242873)

[4. Register Values Before Call 8](#_Toc116242874)

[5. After Entering the Divide Subroutine 9](#_Toc116242875)

[6. Values After the First CONT loop 9](#_Toc116242876)

[7. Part 1 Final Values 10](#_Toc116242877)

[8. Value of Switches being loaded into r4 11](#_Toc116242878)

[9. Loading the Byte Sized Value for the 7 Segment 11](#_Toc116242879)

[10. Board After Value is Loaded into 7-Segment 12](#_Toc116242880)

[11. Squared Value on the LEDs 13](#_Toc116242881)

[12. After SRLI 14](#_Toc116242882)

[13. After And 15](#_Toc116242883)

[14. After r10 is iterated 16](#_Toc116242884)

[15. Starting Values Entering the Fourth Loop 17](#_Toc116242885)

[16. Fourth Loop After srli and and 18](#_Toc116242886)

[17. End of Part Three Program Registers 19](#_Toc116242887)

[18. Part 3 Test Case Results 20](#_Toc116242888)

[19. Registers before entering Ones Loop 21](#_Toc116242889)

[20. Registers after exiting Ones Subroutine 22](#_Toc116242890)

[21. Before Entering Ones subroutine Second Time 23](#_Toc116242891)

[22. After Exiting Ones Subroutine 24](#_Toc116242892)

[23. Final Values Of Program 25](#_Toc116242893)

[24. 7 Segment Part 4 26](#_Toc116242894)

[25. Starting values after Initialization 27](#_Toc116242895)

[26. Registers after Returning from Count 28](#_Toc116242896)

[27. Part 5 LEDs 29](#_Toc116242897)

# Objective

Upon the completion of this lab, one will have co-designed an embedded soft-core system and application development in an FPGA design environment. We will be using Intel’s EDA design tool Qsys, in conjunction with the Quartus software, and Intel FPGA monitor.

# Hardware Requirements

* Computer with Intel FPGA Monitor program 16.1
* Computer with Quartus Prime 16.1.
* DE2-115 Board
* A-B USB Cable

# Software Requirements

* Intel FPGA monitor program 16.1 or greater
* Quartus Prime with Qsys, version 16.1.

# Background

For this lab, we will be using the soft-core embedded system that was designed in the previous lab. The block diagram for this embedded system can be seen in the figure below.

Diagram

Description automatically generated

## Soft Core Embedded System Design

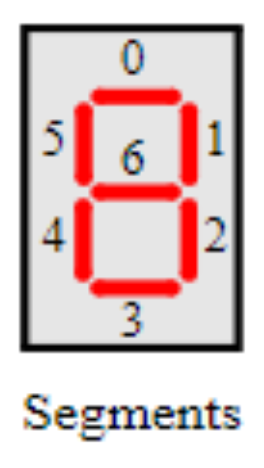
In order to write to a 7-segment display, we have two address spaces, one is for the four most significant bits and the second for the four least significant bits. This is because the DE2-115 board has eight 7-Segment displays. From the design in the last lab, the address spaces for these were, 0x2020 and 0x2030. The typical address spaces for the DE2-115 board can be seen below.

Diagram

Description automatically generated

## 7-Segment Display Address Spaces

Now, from this, we also need to know how to manipulate the 7-Segments in the way that we want. So, being that the display is an active low device, in order to light up a certain display we must pass a 1 into the bit that we want to light up. Now, the segments are controlled individually, so, passing a 7 bit value of 0b1111000 will turn on the bits corresponding to bit zero, one, and two. How these bits correspond to the actual 7 segment display can be seen below.



## 7-Segment Display Breakdown

With this knowledge we can begin this project.

# Project Overview

For this lab, we will be using the soft-core embedded system that was developed in the previous project in order to perform a variety of tasks on the DE2-115 board. The first of these tasks are converting a binary number to its decimal equivalent. Then, we will be creating a loop that will read the DIP switches, output the value on the 7-segment display, create a delay, and display the square on the LEDs. Next, we will be using a program to find the longest string of ones in a given word value. Then using this to work for a list of word values. And lastly creating a counter that works in the bits 10-15 on a given value.

.

# Project Procedure

Graphical user interface, text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

# Conclusion

This lab accomplished its goal of using the softcore embedded system formed in the last exercise, and Intel’s FPGA design enviroment in order to complete certain tasks. We performed binary to decimal conversions, displayed an input value from the switches on the 7-Segment Display, and the square of that value on the LEDs, counted the longest string of ones from a list, and created a counter in certain memory bits. This further bolstered the knowledge of PIO devices and how they work in a NIOS II coding enviroment.

# Appendix A (Part 1 Code)

/\* Program that converts a binary number to decimal, you can modify the program \*/

.text

.global \_start

\_start:

movia r4, N

addi r8, r4, 4 # r8 points to storage location

ldw r4, (r4) #r4 loads N

call DIVIDE #parameter for DIVIDE is in r4

stb r3, 1(r8)

stb r2, (r8)

END: br END

#Subroutine can be modified if needed\*/

# Subroutine to perform the integer division r4 / 10 \*/

# Returns: quotient in r3, and remainder in r2 \*/

DIVIDE:

mov r2, r4 # will be the remainder

movi r5, 10 #divisor

movi r3, 0 #r3 will be the quotient

CONT:

blt r2, r5, DIV\_END

sub r2, r2, r5

addi r3, r3, 1

br CONT

DIV\_END:

ret

N: .word 9876 #the decimal number to be converted

Digits: .space 4 #space for storage location

.end

# Appendix B (Part 2 Code)

.data

LUT:

.byte 0b11000000#0

.byte 0b11111001#1

.byte 0b10100100#2

.byte 0b10110000#3

.byte 0b10011001#4

.byte 0b10010010#5

.byte 0b10000010#6

.byte 0b11111000#7

.byte 0b10000000#8

.byte 0b10010000#9

.text

.equ RLEDs, 0x2040

.equ SWITCHES, 0x2010

.equ GLEDs, 0x2030

.equ HexLSB, 0x2020

.equ HexMSB, 0x2000

.global \_start

\_start:

movia r2, RLEDs #LED Add

movia r3, SWITCHES #Switch Add

#movia r5, GLEDs #Green LEDs

movia r6, HexLSB #LSB Hex Bits

movia r5, 0x500 #LUT

movia r13, 0x1

br LOOP

DELAY:

movia r11, 0x0 #Initialize R11, our iterator

movia r10, 0x04C4B40 #what we have to iterate to

Delay2:

bgt r11, r10, LOOP2 #Compares iteration 1 instructino

add r11 ,r11, r13 #1 #Adds one to the iteration, 2 instructions

movia r12, 0x0 # 3 instructions (Filler code)

add r12, r12, r12 # 4 instructions (FIller Code)

br Delay2

LOOP:

ldbio r4, (r3) #Switch State into R4

add r4, r4, r5

ldbio r7, (r4)

stbio r7, (r6)

br DELAY

LOOP2:

ldbio r4, (r3)

mul r4, r4, r4

stwio r4, (r2)

br LOOP

.end

# Appendix C (Part 3 Code)

.text

.global \_start

\_start:

ldw r9, TEST\_NUM(r0)

mov r10, r0

LOOP: beq r9, r0, END

srli r11, r9, 0x01

and r9, r9, r11

addi r10, r10, 0x01

br LOOP

END:

br END

TEST\_NUM:

.word 0x3fabedef

.end

# Appendix D (Part 4 Code)

.text

.global \_start

\_start:

movia r9, TEST\_NUM

mov r10, r0

movia r6, 0

movia r2, 0

main:

ldw r4, (r9)

bgt r6, r2, skip

mov r6, r2

skip:

movia r2, 0

beq r4, r0, end

call ONES

addi r9, r9, 4

br main

ONES:

beq r4, r0, ENDONES

srli r11, r4, 0x01

and r4, r4, r11

addi r2, r2, 0x01

br ONES

ENDONES:

ret

end:

movia r1, 0x10000020

movia r7, LUT

add r7, r7, r6

ldb r6, (r7)

stb r6, (r1)

ENDLOOP:

br ENDLOOP

TEST\_NUM:

.word 0x3fabedef

.word 0xFF12384F

.word 0x99FCAB94

.word 0x10219341

.word 0xFCABEFCA

.word 0x12345678

.word 0xFFF00000

.word 0xF1231112

.word 0x00FEABCA

.word 0xF1234F87

.word 0x00000000

LUT:

.byte 0b11000000#0

.byte 0b11111001#1

.byte 0b10100100#2

.byte 0b10110000#3

.byte 0b10011001#4

.byte 0b10010010#5

.byte 0b10000010#6

.byte 0b11111000#7

.byte 0b10000000#8

.byte 0b10010000#9

.byte 0b10001000#A

.byte 0b10000011#b

.byte 0b11000110#C

.byte 0b10100001#d

.byte 0b10000110#E

.byte 0b10001110#F

.end

# Appendix E (Part 5 Code)

.text

.equ RLEDs, 0x2040

.global \_start

\_start:

movia r3, 0b1111110000000000

movia r6, 0b11111111111111110000001111111111

movia r4, RLEDs

movia r1, 0x3fabedef

movia r2, 0b0001111110000000000

Loop:

and r2, r2, r1

call count

and r1, r1, r6

or r1, r1, r2

br Loop

count:

beq r2, r3, reset

addi r2, r2, 0x400

stw r2, (r4)

ret

reset:

movia r2, 0x0

stw r2, (r4)

ret

END:

br END

.end