Lab 5

David Shmailov, Aviram Lachmani

June 2021

Contents

1 Introduction

In this task we completed a MIPS architecture we were provided, created a I/O envelope that encapsulates the MIPS core with IO peripherals together creating a MIPS microprocessor.

2 Top level diagram

The top level in our design is called "System" and it envelopes the MIPS core with I/O blocks which creates a complete basic MIPS based microprocessor. Before we could properly measure the maximal frequency this design is able to work, we had to make sure all inputs and outputs are buffered by registers.

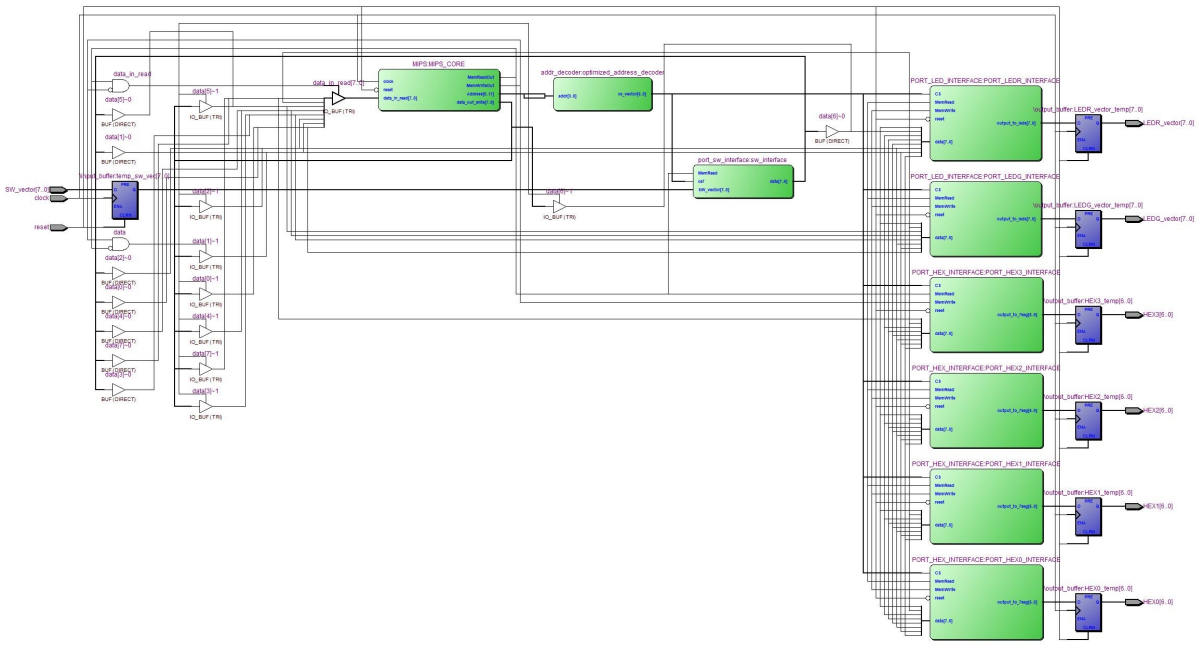

Figure 1: Top RTL view

3 MIPS core diagram

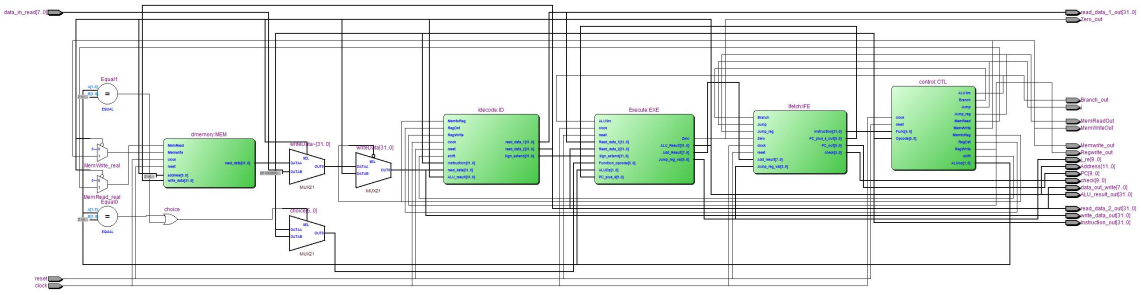

Figure 2: MIPS core RTL view

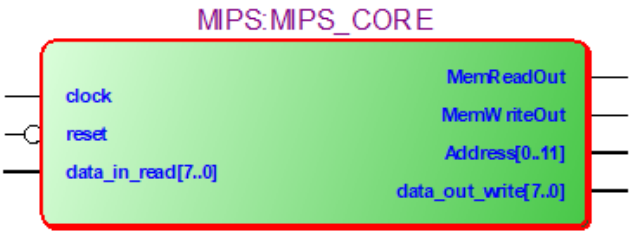

Figure 3: Graphical Description

Name	Direction	Size	Functionality
clock	Input	1 bit	system clock
reset	Input	1 bit	global positive edge reset
data_in_read	Input	8 bit	data to read from I/O
MemReadOut	Output	1 bit	control line for I/O
MemWriteOut	Output	1 bit	control line for I/O
Address	Output	12 bit	address to access I/O
data_out_write	Output	8 bit	data to write to I/O

Table 1: Port table

Description:

This is the MIPS core. It is based on the MIPS single cycle architecture without pipelines. The core consists of five stages: Ifetch, Idecode,Dmemory,Execute,Control. the instruction memory is inside Ifetch. Execute represents the main Data Path and Control for Control signals. since there is only single cycle for each instruction, the control is fairly simple and is a pure combinational logic circuit. Ifetch main purpose is to fetch the next command from the program memory, either PC +4 or a branch depending on the case. Execute consists of the main ALU and other units to carry out different operations. Idecode consists of the core's register file and the necessary logic to access the registers for read/write operations according to the instruction format. dmemory consists of purely data memory and not instructions. the main output of the MIPS is actually the dmemory, but in our case after compilation it is implemented as a LUT inside the MIPS it self, since we do not deal with a real memory component, we can assume memory read/write delays of 1 cycle. the other output of the core is to the peripherals I/O accessible via memory addressing.

4 LED interface diagram

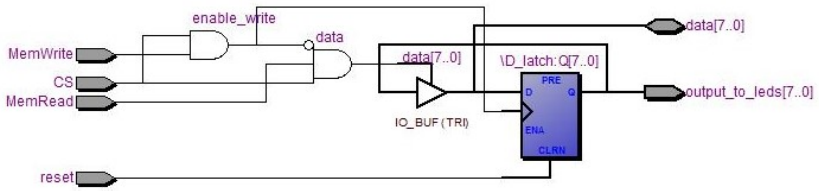

Figure 4: LED Interface RTL view

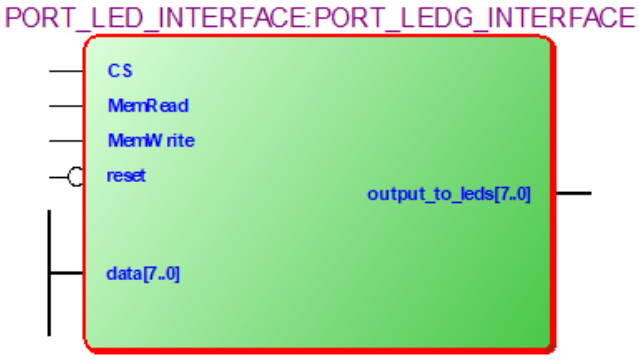

Figure 5: Graphical Description

Name	Direction	Size	Functionality
CS#	Input	1 bit	chip select
MemRead	Input	1 bit	control line
MemWrite	Input	1 bit	control line
reset	Input	1 bit	global positive edge reset
data	Input/Output	8 bit	data to write/read
output to LEDs	Output	8 bit	goes to the physical LEDs

Table 2: Port table

Description:

This is a port interface that enables the top System design to connect to the LEDs on the FPGA kit. the interface uses a latch for continues output to the LEDs and enables both reading from the latch and writing to the latch via bi-directional data port. Uses the chip select control line together with memRead or memWrite to determine whether to use data as input or output. If both memRead and memWrite are on the interface will fail.

5 HEX interface diagram

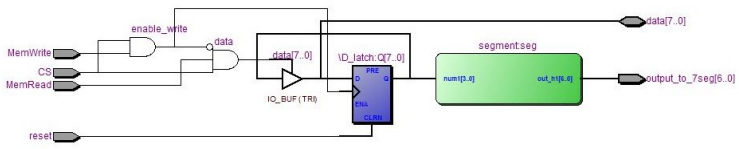

Figure 6: HEX Interface RTL view

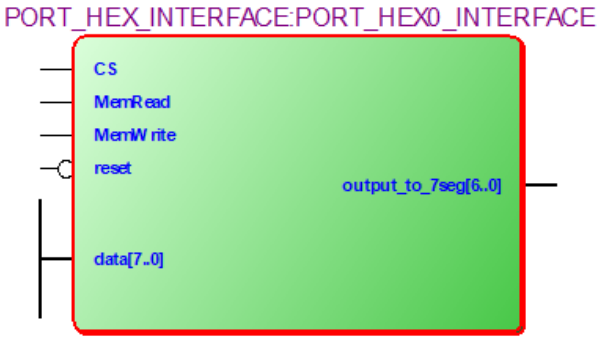

Figure 7: Graphical Description

Name	Direction	Size	Functionality
CS#	Input	1 bit	chip select
MemRead	Input	1 bit	control line
MemWrite	Input	1 bit	control line
reset	Input	1 bit	global positive edge reset
data	Input/Output	8 bit	data to write/read
output to 7seg	Output	8 bit	goes to the physical 7 segments

Table 3: Port table

Description:

This is a port interface that enables the top System design to connect to the 7 segment in a HEX format on the FPGA kit. the interface uses a latch for continues output to the 7 segment and enables both reading from the latch and writing to the latch via bi-directional data port. Uses the chip select control line together with memRead or memWrite to determine whether to use data as input or output. If both memRead and memWrite are on the interface will fail. the interface only outputs the 4 LSB bits of data, in a HEX decoding to 7 segment. the 4 MSB bits are discarded after the latch.

6 switch interface diagram

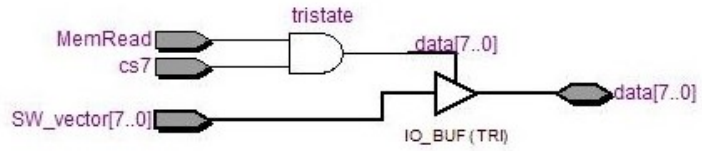

Figure 8: Switch Interface RTL view

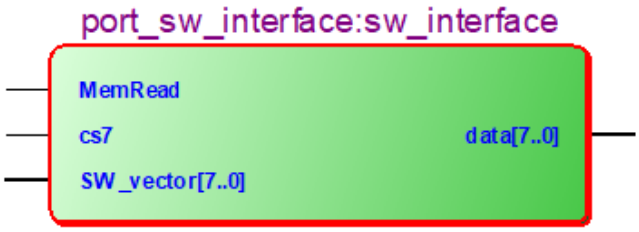

Figure 9: Graphical Description

Name	Direction	Size	Functionality
CS7	Input	1 bit	chip select
MemRead	Input	1 bit	control line
SW_vector	Input	8 bit	connects to physical switches
data	Input/Output	8 bit	outputs switch to read

Table 4: Port table

Description:

This is a port interface that enables the top System design to connect to the 8 toggle switches on the FPGA kit. the interface uses a data input/output bi-directional but actually functions as output only. the reason is for easier compatibility with the System envelope. Uses the chip select control line together with memRead in a tri-state for the data output.

7 address decoder diagram

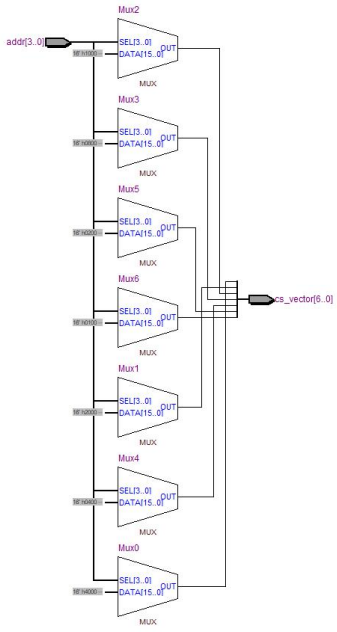

Figure 10: address decoder RTL view

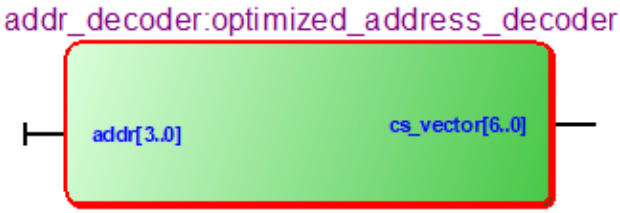

Figure 11: Graphical Description

Name	Direction	Size	Functionality
addr	Input	4 bit	address[11,4,3,2]
cs_vector	Output	7 bit	chip select vector for I/O

Table 5: Port table

Description:

This is a simple address decoder that receives the 12th bit and 3 others to determine whether there is read or write requests from the I/O and to which one. decodes the address to a chip select vector of length 7 that connects to all the I/O blocks. the decoding is done via the table provided in the assignment.

8 blocks logic usage

Block	Logic Cells	Logic Registers	LUT-Only LCs	Register-Only LCs	LUT/Register LCs
System (Total)	3888 (31)	1907 (8)	1981 (23)	614 (0)	1293 (12)
MIPS_CORE	2761 (7)	1000 (0)	1761 (7)	143 (0)	857 (0)
addr_decoder	0	0	0	0	0
HEX0_INTERFACE	18 (11)	8 (8)	10 (3)	0 (0)	8 (8)
HEX1_INTERFACE	18 (12)	8 (8)	10 (4)	0 (0)	8 (7)
HEX2_INTERFACE	18 (11)	8 (8)	10 (3)	0 (0)	8 (8)
HEX3_INTERFACE	18 (11)	8 (8)	10 (3)	0 (0)	8 (8)
LEDG_INTERFACE	11 (11)	8 (8)	3 (3)	0 (0)	8 (8)
LEDR_INTERFACE	13 (13)	8 (8)	5 (5)	0 (0)	8 (8)
sw_interface	11 (11)	0 (0)	2 (2)	0 (0)	9 (9)

Table 6: Block logic usage

9 Critical Path

The critical path in the design is in coherence with what we learned in our lectures. it is in the load word instructions. the maximal clock frequency is 19.79 MHz. The minimum path is the reset line, if we consider the LEDS to be an output of microprocessor, than the reset is the shortest input to output path. if we had more time we could optimize our design to use less logic gates with the help of minimization techniques, and use the pipeline architecture and that way reduce significantly the critical path.

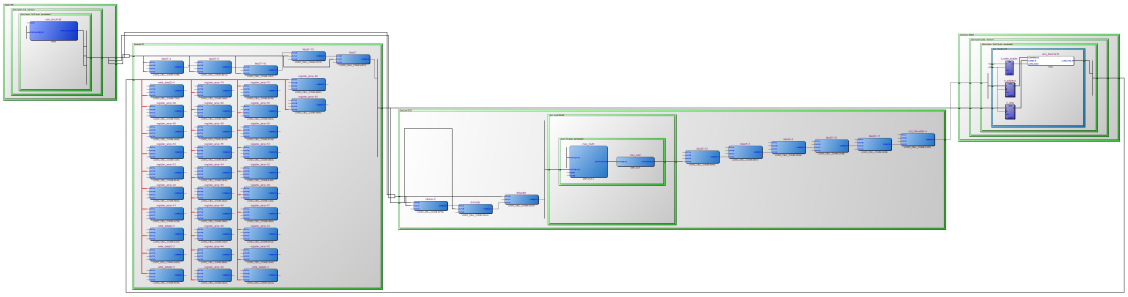

Figure 12: Critical path from the map viewer

10 Verification

We conducted several tests and used signal tap to verify them all. the following is the verification of the assembly code from self test 1. the code in this subtract mode subtracts 2 from 0 modulu 256 (or modulu 16 on the HEX outputs).

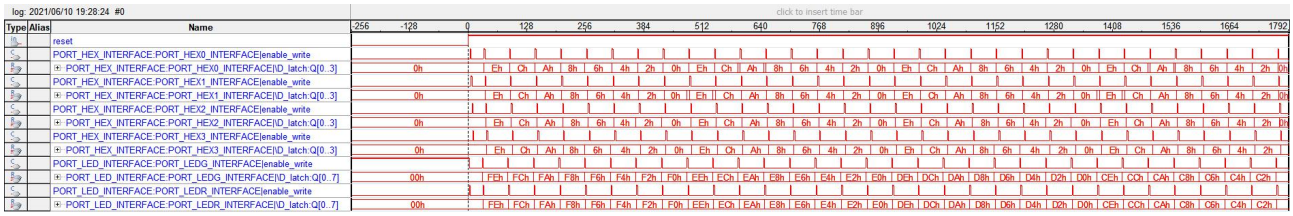

Figure 13: test 1 subtract mode signal tap verification

11 Conclusions

In this assignment we were required to complete the MIPS core and design the peripherals envelope to create a working minimal instruction microprocessor. we got a deeper understanding of microprocessors and the idea of memory mapped I/O. We learned how to verify during design process and we had to go each instruction and verify it is implemented in the MIPS core we were provided and if not, implement it our selves. we had the freedom to choose how to implement the support for said instructions and we had to change some of the implementation already provided.