Lab 5

Digital Design and Computer Architecture: Full Adder

Fall 2019

Paul Kim

Digital System Design

Reference and Resources:

- https://www.youtube.com/watch?v=isObSaq3Eg0&list=PLZax60ptwWrM9Suuj9t4ijIOE8J_Blirl&index=6&t=25s
- https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug_nios2_custom_instruction.pdf
- https://m.blog.naver.com/PostView.nhn?blogId=no-_-&logNo=220766580401&proxyReferer=https%3A%2F%2Fwww.google.com%2F
- https://blog.naver.com/no-_-/220766580401
- https://blog.csdn.net/yinghuanhuan/article/details/80079083
- https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=163&No=529&PartNo=5
- https://www.renesas.com/us/en/support/technical-resources/engineer-school/mcu-01-basic-structure-operation.html
- http://hardwarebee.com/adding-a-cpu-to-your-fpga-design-tutorial/

Lab Goal: Designing System on chip (SOC). Learning that one can easily design his own soft CPU or processing unit inside built in hardware that can handle any task as long as you could design the code

Lab Requirement:

- Design a calculator that can intake two inputs (0~9) and output the calculation result onto 2 seven-segment LED display
- The 2 inputs calculation must be done in the MCU not FPGA
- Use switch on the FPGA to write the data onto register of the MCU and able to read from the register of the MCU using C/C++ code
- Use LED display to show the result

My Design:

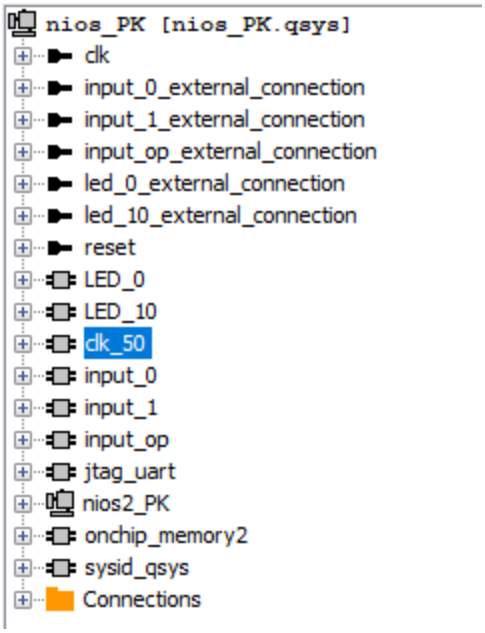
- I've used 2 four-bits inputs because the DE2-115 board has only 18 switches. I needed at least 20 switches for 2 inputs because each input is range from 0 to 9. So I've used sw[0]-sw[3] for first input and sw[4]-sw[7] for second input
- The alu operation input has 4 combinations that can be implemented in 2bits input. 00 for addition. 01 for subtraction. 10 for multiplication. 11 for division. sw[8]-sw[9] were used.
- The result LED was used to HEX[0] for first digit and HEX[1] for second digit. Each HEX required 7 bits. Therefore the total output bits were 14bits long. At beginning, I've used one 14 bits long LED output. But due to my design implementation, I had to divided to two 7 bits long LED output
- Reset button was key[0] that can be considered as equal sign. Whenever reset button is triggered, my C program inside the MCU which does the calculation, resets itself and goes back to main function. Therefore, triggering the reset button recalculates new values of the two inputs
- The default clock frequency for nios ii processor which connects all the component of the MCU (PIO, on chip memory, etc) was 50MHz. After some research the manual had signal name Clock_50 which was exact fit for 50MHz clock input. The FPGA Pin no. was 'PIN_Y2'
- All reading and writing and calculation and LED 7 segment decoding were done inside the MCU (c language coding).

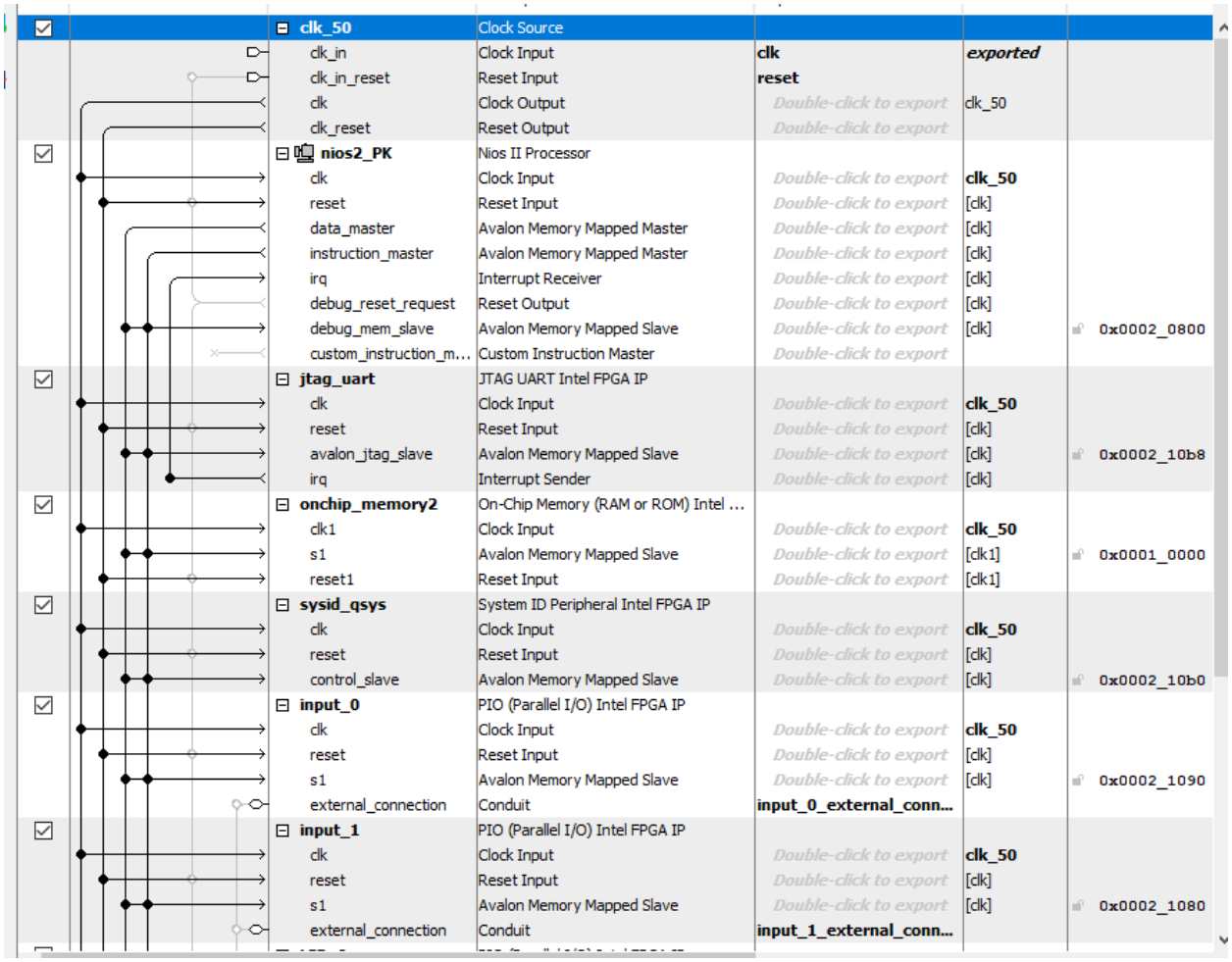
My mcu implementation:

Using Quartus 18.1 Lite software:

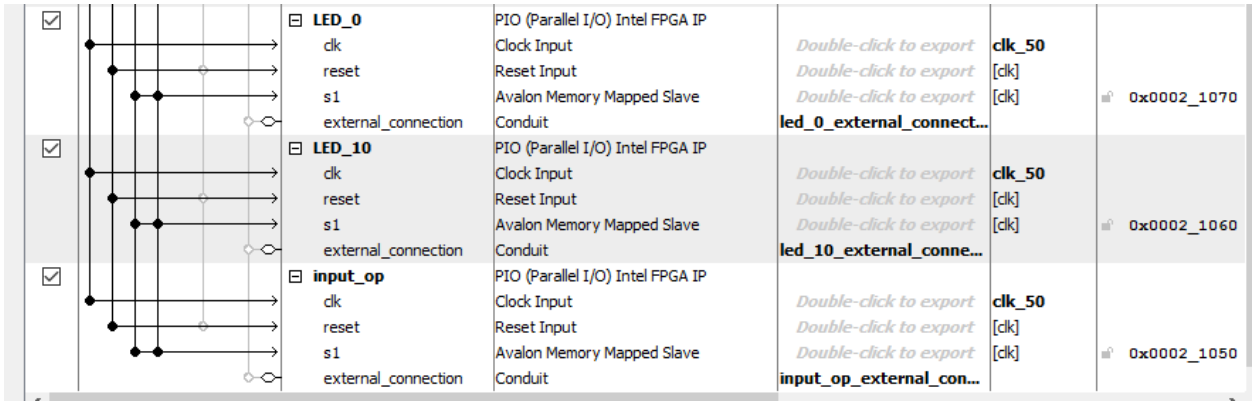
After the creation of new project, I've used platform designer (qsys) to add mcu called 'nios2_PK'.

Inside the mcu:





☑	**clk_50**	Clock Source			
	clk_in	Clock Input	**clk**	***exported***	
	clk_in_reset	Reset Input	**reset**		
	clk	Clock Output	*Double-click to export*	clk_50	
	clk_reset	Reset Output	*Double-click to export*		
☑	**nios2_PK**	Nios II Processor			
	clk	Clock Input	*Double-click to export*	**clk_50**	
	reset	Reset Input	*Double-click to export*	[clk]	
	data_master	Avalon Memory Mapped Master	*Double-click to export*	[clk]	
	instruction_master	Avalon Memory Mapped Master	*Double-click to export*	[clk]	
	irq	Interrupt Receiver	*Double-click to export*	[clk]	
	debug_reset_request	Reset Output	*Double-click to export*	[clk]	
	debug_mem_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0002_0800
	custom_instruction_m...	Custom Instruction Master	*Double-click to export*		
☑	**jtag_uart**	JTAG UART Intel FPGA IP			
	clk	Clock Input	*Double-click to export*	**clk_50**	
	reset	Reset Input	*Double-click to export*	[clk]	
	avalon_jtag_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0002_10b8
	irq	Interrupt Sender	*Double-click to export*	[clk]	
☑	**onchip_memory2**	On-Chip Memory (RAM or ROM) Intel ...			
	clk1	Clock Input	*Double-click to export*	**clk_50**	
	s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk1]	0x0001_0000
	reset1	Reset Input	*Double-click to export*	[clk1]	
☑	**sysid_qsys**	System ID Peripheral Intel FPGA IP			
	clk	Clock Input	*Double-click to export*	**clk_50**	
	reset	Reset Input	*Double-click to export*	[clk]	
	control_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0002_10b0
☑	**input_0**	PIO (Parallel I/O) Intel FPGA IP			
	clk	Clock Input	*Double-click to export*	**clk_50**	
	reset	Reset Input	*Double-click to export*	[clk]	
	s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0002_1090
	external_connection	Conduit	**input_0_external_conn...**		
☑	**input_1**	PIO (Parallel I/O) Intel FPGA IP			
	clk	Clock Input	*Double-click to export*	**clk_50**	
	reset	Reset Input	*Double-click to export*	[clk]	
	s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0002_1080
	external_connection	Conduit	**input_1_external_conn...**		



☑	**LED_0**	PIO (Parallel I/O) Intel FPGA IP			
	clk	Clock Input	*Double-click to export*	**clk_50**	
	reset	Reset Input	*Double-click to export*	[clk]	
	s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0002_1070
	external_connection	Conduit	**led_0_external_connect...**		
☑	**LED_10**	PIO (Parallel I/O) Intel FPGA IP			
	clk	Clock Input	*Double-click to export*	**clk_50**	
	reset	Reset Input	*Double-click to export*	[clk]	
	s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0002_1060
	external_connection	Conduit	**led_10_external_conne...**		
☑	**input_op**	PIO (Parallel I/O) Intel FPGA IP			
	clk	Clock Input	*Double-click to export*	**clk_50**	
	reset	Reset Input	*Double-click to export*	[clk]	
	s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0002_1050
	external_connection	Conduit	**input_op_external_con...**		

After the nios ii creation, the software gave me verilog code that maps all the components of the mcu:

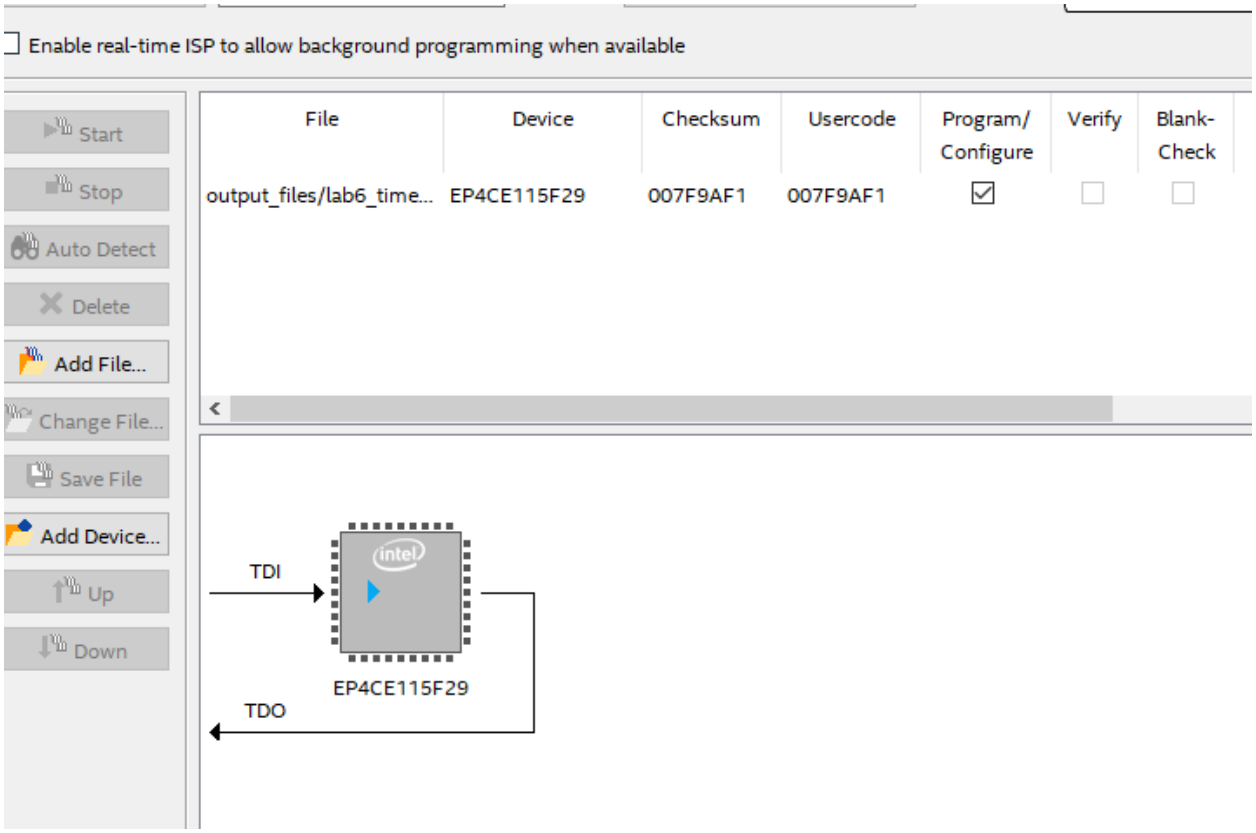
That I can use it to connect to my top entity module:

```
module niosii_top
(output logic [6:0] LEDone, LEDten, //output LED first digit, second digit
input logic [3:0] input0, input1, //two 4 bits input numbers that only accepts 0~9 decimal digit //second 4bit number that that only accepts 0~9 decimal dig
input logic [1:0] inputop, //2 bit operation: 00 add | 01 subtract| 10 multiply | 11 divide
input logic clk50, reset);

    nios_PK u0 (
        .clk_clk                       (clk50),                          //                    clk.clk
        .reset_reset_n                 (reset),                          //                  reset.reset_n
        .input_0_external_connection_export  (input0),  //  input_0_external_connection.export
        .input_1_external_connection_export  (input1),  //  input_1_external_connection.export
        .led_0_external_connection_export    (LEDone),    //    led_0_external_connection.export
        .led_10_external_connection_export   (LEDten),   //   led_10_external_connection.export
        .input_op_external_connection_export (inputop)  // input_op_external_connection.export
    );

endmodule
```

Pin planner for the FPGA:

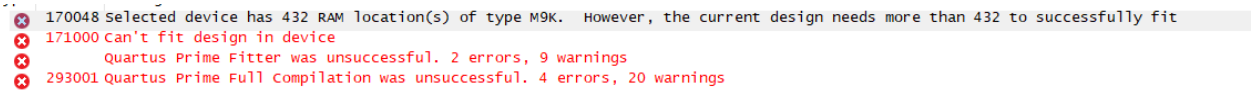
LEDone[6]	Output	PIN_H22	6	B6_N0	PIN_H22	2.5 V
LEDone[5]	Output	PIN_J22	6	B6_N0	PIN_J22	2.5 V
LEDone[4]	Output	PIN_L25	6	B6_N1	PIN_L25	2.5 V
LEDone[3]	Output	PIN_L26	6	B6_N1	PIN_L26	2.5 V
LEDone[2]	Output	PIN_E17	7	B7_N2	PIN_E17	2.5 V
LEDone[1]	Output	PIN_F22	7	B7_N0	PIN_F22	2.5 V
LEDone[0]	Output	PIN_G18	7	B7_N2	PIN_G18	2.5 V
LEDten[6]	Output	PIN_U24	5	B5_N0	PIN_U24	2.5 V
LEDten[5]	Output	PIN_U23	5	B5_N1	PIN_U23	2.5 V
LEDten[4]	Output	PIN_W25	5	B5_N1	PIN_W25	2.5 V
LEDten[3]	Output	PIN_W22	5	B5_N0	PIN_W22	2.5 V
LEDten[2]	Output	PIN_W21	5	B5_N1	PIN_W21	2.5 V
LEDten[1]	Output	PIN_Y22	5	B5_N0	PIN_Y22	2.5 V
LEDten[0]	Output	PIN_M24	6	B6_N2	PIN_M24	2.5 V
altera_reserved_tck	Input				PIN_P5	2.5 V (default)
altera_reserved_tdi	Input				PIN_P7	2.5 V (default)
altera_reserved_tdo	Output				PIN_P6	2.5 V (default)
altera_reserved_tms	Input				PIN_P8	2.5 V (default)
clk50	Input	PIN_Y2	2	B2_N0	PIN_Y2	2.5 V
input0[3]	Input	PIN_AD27	5	B5_N2	PIN_AD27	2.5 V
input0[2]	Input	PIN_AC27	5	B5_N2	PIN_AC27	2.5 V
input0[1]	Input	PIN_AC28	5	B5_N2	PIN_AC28	2.5 V
input0[0]	Input	PIN_AB28	5	B5_N1	PIN_AB28	2.5 V
input1[3]	Input	PIN_AB26	5	B5_N1	PIN_AB26	2.5 V
input1[2]	Input	PIN_AD26	5	B5_N2	PIN_AD26	2.5 V
input1[1]	Input	PIN_AC26	5	B5_N2	PIN_AC26	2.5 V
input1[0]	Input	PIN_AB27	5	B5_N1	PIN_AB27	2.5 V
inputop[1]	Input	PIN_AB25	5	B5_N1	PIN_AB25	2.5 V
inputop[0]	Input	PIN_AC25	5	B5_N2	PIN_AC25	2.5 V
reset	Input	PIN_M23	6	B6_N2	PIN_M23	2.5 V
<<new node>>						

The FPGA board must be programmed and be running before I've started to implement my code for the mcu:



Difficulty for implementing MCU:

I had a lot of difficulty because it was my first time using qsys and even making a SOC system. I've also wasted of a lot of time for misunderstanding error message such as:

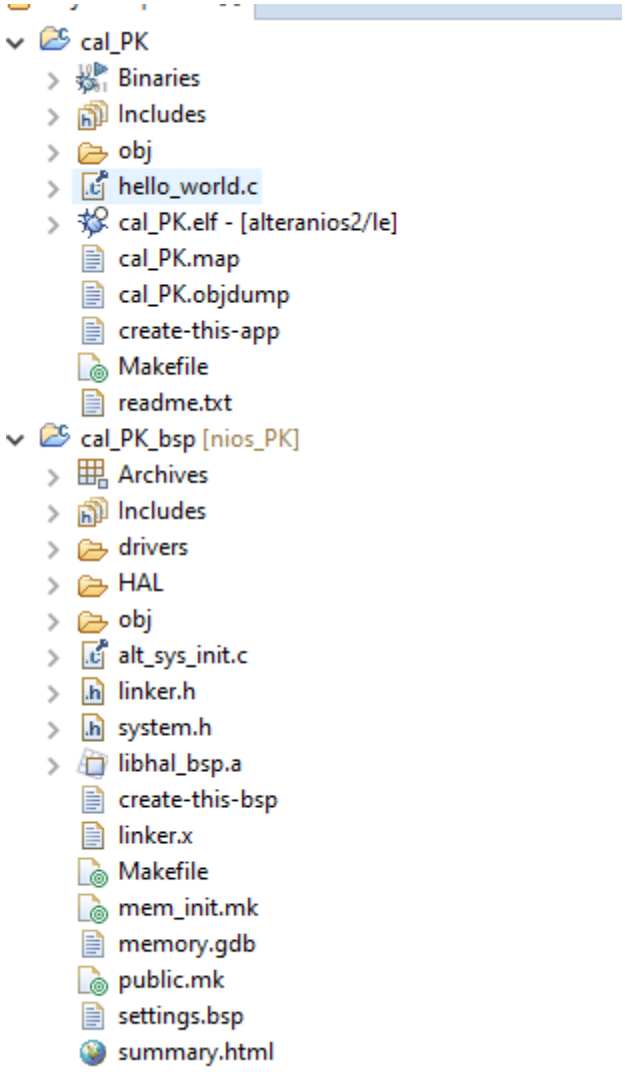


I've thought this message was telling me that my mcu on chip total memory was too small. But it was saying that the chip total memory was too big instead. And whenever I've kept increasing the memory, the qsys compilation took a long time I've tested upto 2^26 bytes (67108864 bytes) which took me more than 2 hours so I've later quit and found that the total memory should be smaller instead.

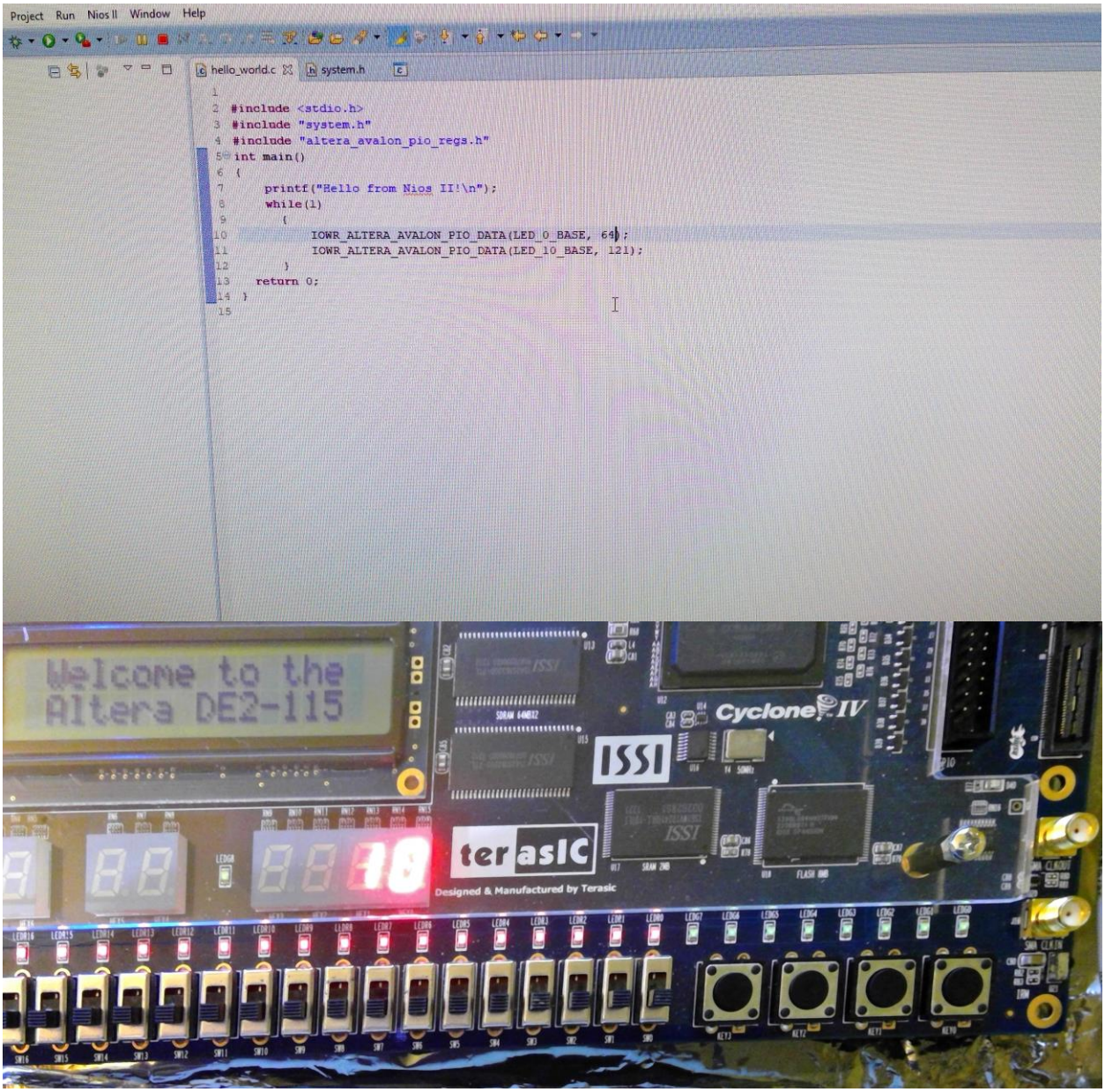
There was other software difficulties as well.

C code implementation:

I've used the Hello world template that was given by the eclipsed which automatically setup my workspace that had all the libraries that I needed for my c code:



From there, I've tried to learn how to read and write mcu's register by implementing and testing every decimal numbers:


Project Run Nios II Window Help
hello_world.c
system.h
#include <stdio.h>
#include "system.h"
#include "altera_avalon_pio_regs.h"
int main()
{
printf("Hello from Nios II!\n");
while(1)
{
IOWR_ALTERA_AVALON_PIO_DATA(LED_0_BASE, 64);
IOWR_ALTERA_AVALON_PIO_DATA(LED_10_BASE, 121);
}
return 0;
}
Welcome to the
Altera DE2-115
Cyclone IV
ISSI
terasIC
Designed & Manufactured by Terasic

Nios II Window Help

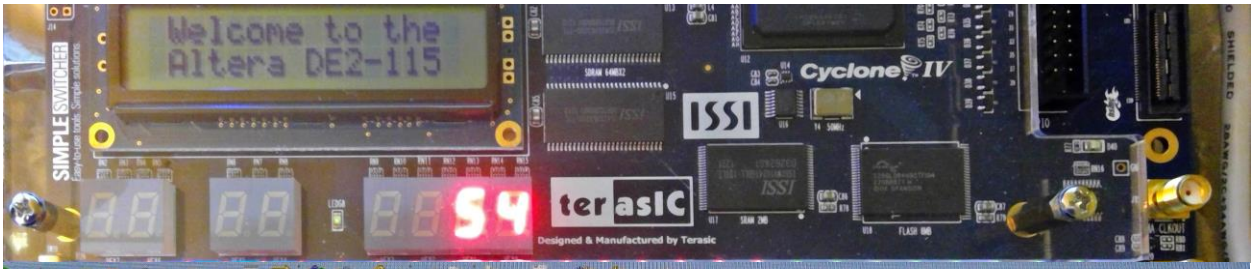
hello_world.c system.h

```
1
2  #include <stdio.h>
3  #include "system.h"
4  #include "altera_avalon_pio_regs.h"
5  int main()
6  {
7      printf("Hello from Nios II!\n");
8      while(1)
9        {
10           IOWR_ALTERA_AVALON_PIO_DATA(LED_0_BASE, 36);
11           IOWR_ALTERA_AVALON_PIO_DATA(LED_10_BASE, 48);
12       }
13    return 0;
14 }
15
```

Window Help

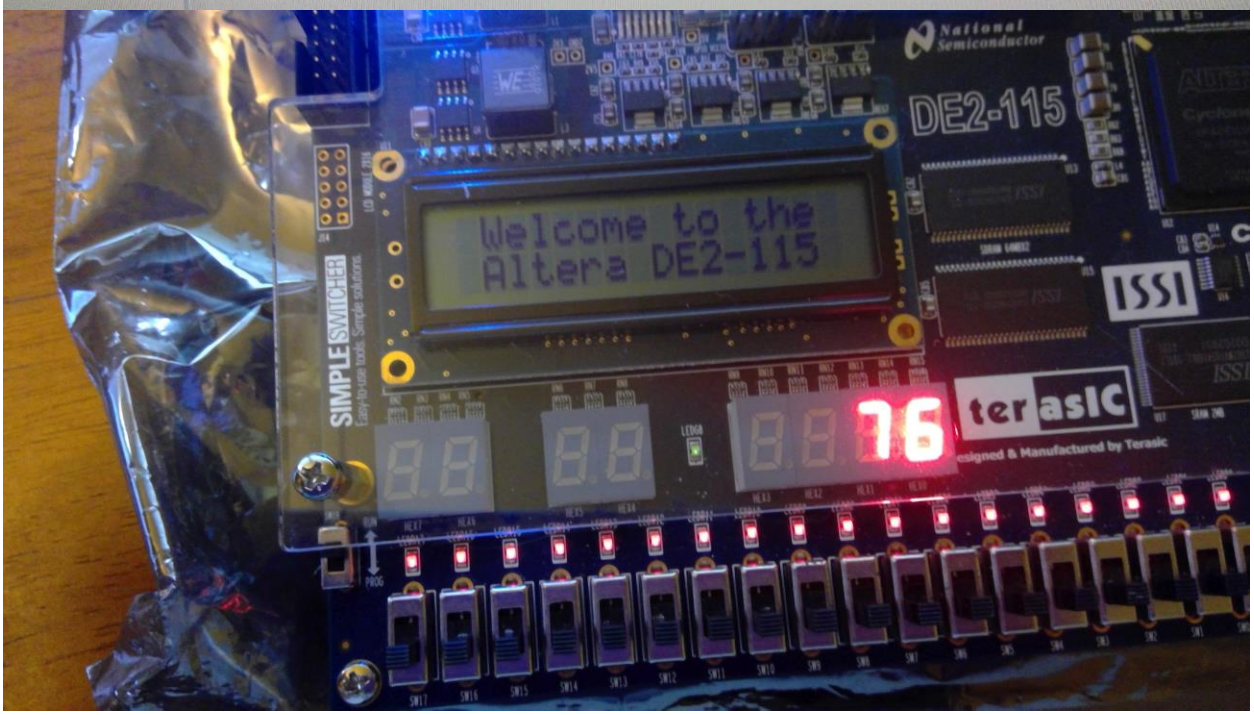
hello_world.c system.h

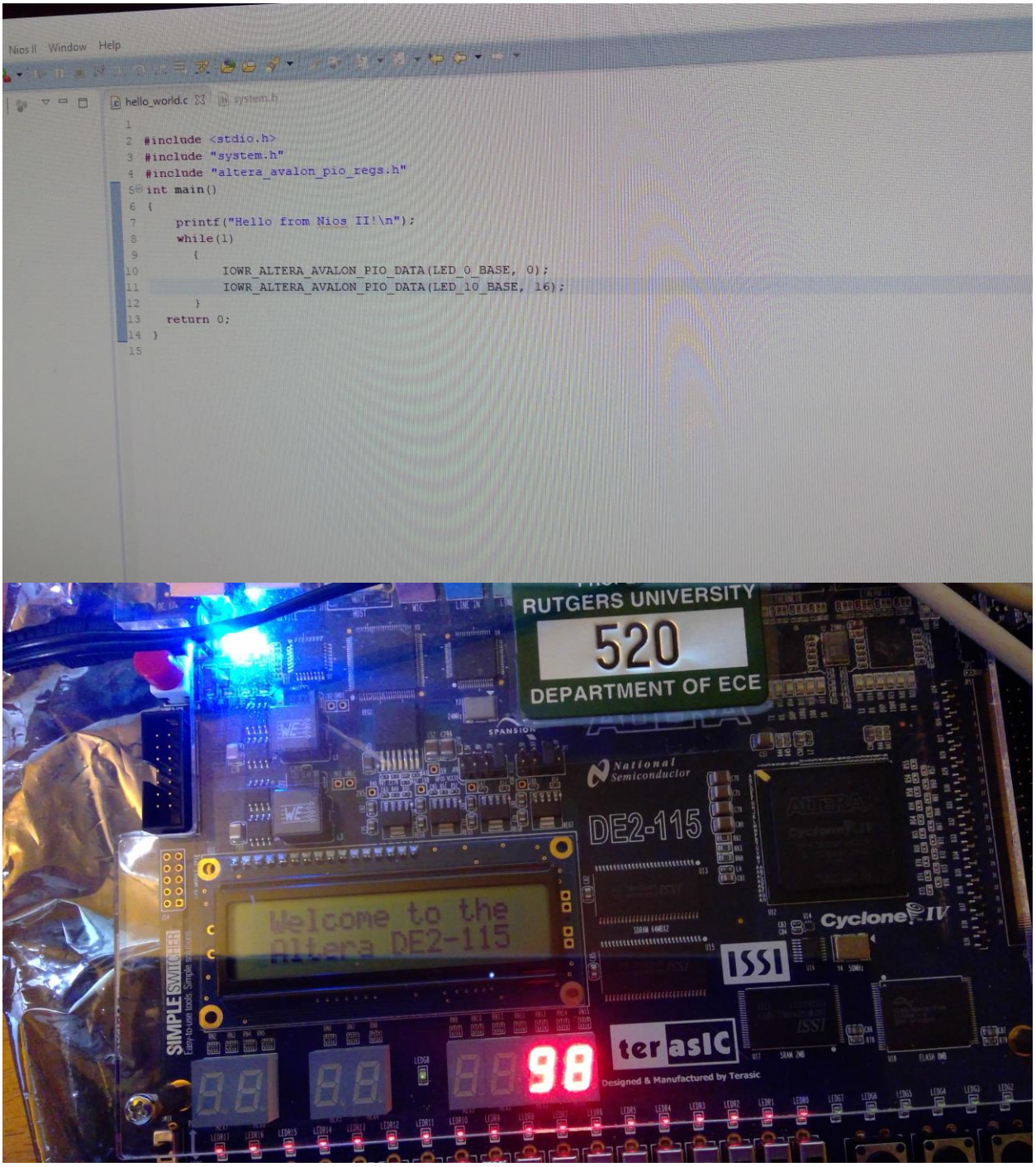
```
1
2  #include <stdio.h>
3  #include "system.h"
4  #include "altera_avalon_pio_regs.h"
5  int main()
6  {
7      printf("Hello from Nios II!\n");
8      while(1)
9        {
10           IOWR_ALTERA_AVALON_PIO_DATA(LED_0_BASE, 25);
11           IOWR_ALTERA_AVALON_PIO_DATA(LED_10_BASE, 18);
12       }
13    return 0;
14 }
15
```

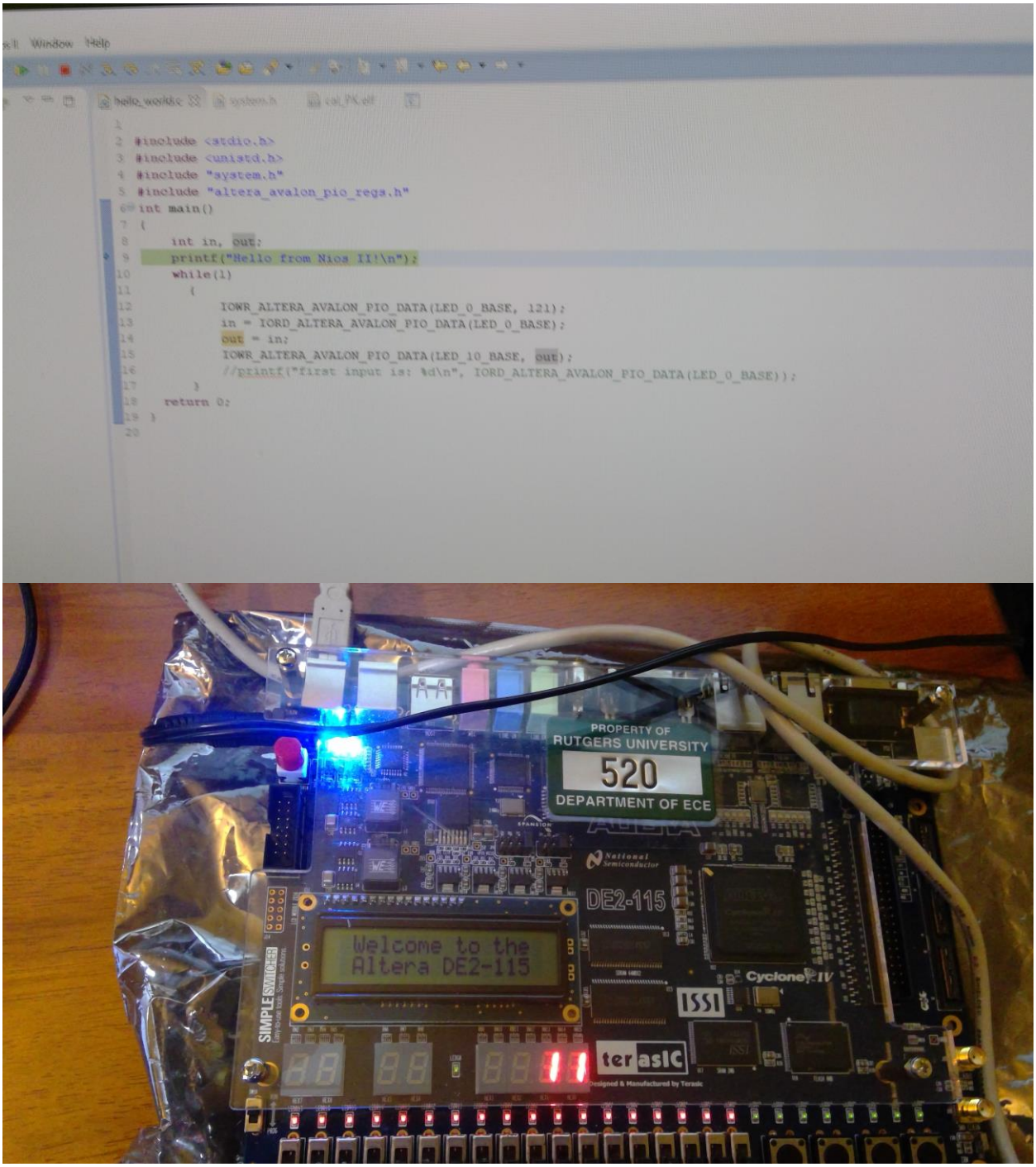


hello_world.c system.h

```
1
2  #include <stdio.h>
3  #include "system.h"
4  #include "altera_avalon_pio_regs.h"
5  int main()
6  {
7      printf("Hello from Nios II!\n");
8      while(1)
9        {
10           IOWR_ALTERA_AVALON_PIO_DATA(LED_0_BASE, 2);
11           IOWR_ALTERA_AVALON_PIO_DATA(LED_10_BASE, 120);
12       }
13   return 0;
14 }
15
```




Nios II Window Help
hello_world.c
system.h
#include <stdio.h>
#include "system.h"
#include "altera_avalon_pio_regs.h"
int main()
{
printf("Hello from Nios II!\n");
while(1)
{
IOWR_ALTERA_AVALON_PIO_DATA(LED_0_BASE, 0);
IOWR_ALTERA_AVALON_PIO_DATA(LED_10_BASE, 16);
}
return 0;
}
RUTGERS UNIVERSITY
520
DEPARTMENT OF ECE
DE2-115
Welcome to the
altera DE2-115
98
terasic
Designed & Manufactured by Terasic
Cyclone IV
ISSI



Once I've figured register read and write implementation, adding the calculation algorithm was easy.

Hello_world.c:

hello_world.c

```c

#include <stdio.h>
#include <unistd.h>
#include "system.h"
#include "altera_avalon_pio_regs.h"
int main()
{
    IOWR_ALTERA_AVALON_PIO_DATA(LED_0_BASE, 64);
    IOWR_ALTERA_AVALON_PIO_DATA(LED_10_BASE, 64);
    int in0, in1, inalu; //inputs
    int out0, out10; //outputs
    int result, result0, result10; //calculation result
    printf("Hello from Nios II!\n");
    //retriving input0, input1, inputop
    in0 = IORD_ALTERA_AVALON_PIO_DATA(INPUT_0_BASE);
    //any input greater than 9 is 9
    if (in0 > 9) {
        in0 = 9;
    }
    else {}
    in1 = IORD_ALTERA_AVALON_PIO_DATA(INPUT_1_BASE);
    //any input greater than 9 is 9
    if (in1 > 9) {
        in1 = 9;
    }
    else {}
    inalu = IORD ALTERA AVALON PIO DATA(INPUT OP BASE);

   //calculation
   switch (inalu)
      {
          case 0: //add
              result = in0 + in1;
              break;
          case 1: //subtract
              result = in0 - in1;
              break;
          case 2: //multiply
              result = in0 * in1;
              break;
          case 3: //divide
              result = in0 / in1;
              break;
          default:
              break;
      }

   result0 = result % 10; //result one decimal integer
   result10 = result / 10; //result ten decimal integer

```

```
50
51      //decoding one decimal digit of the calculation into seven segment integer
52      switch (result0)
53                  {
54                      case 0:
55                          out0 = 64;
56                          break;
57                      case 1:
58                          out0 = 121;
59                          break;
60                      case 2:
61                          out0 = 36;
62                          break;
63                      case 3:
64                          out0 = 48;
65                          break;
66                      case 4:
67                          out0 = 25;
68                          break;
69                      case 5:
70                          out0 = 18;
71                          break;
72                      case 6:
73                          out0 = 2;
74                          break;
75                      case 7:
76                          out0 = 120;
77                          break;
78                      case 8:
79                          out0 = 0;
80                          break;
81                   // anything greater than and equal to 9 is 9 seven segment decoded integ
82                      default:
83                          out0 = 16;
84                  }
```

```
    switch (result10)
                {
                    case 0:
                        out10 = 64;
                        break;
                    case 1:
                        out10 = 121;
                        break;
                    case 2:
                        out10 = 36;
                        break;
                    case 3:
                        out10 = 48;
                        break;
                    case 4:
                        out10 = 25;
                        break;
                    case 5:
                        out10 = 18;
                        break;
                    case 6:
                        out10 = 2;
                        break;
                    case 7:
                        out10 = 120;
                        break;
                    case 8:
                        out10 = 0;
                        break;
                 // anything greater than and equal to 9 is 9 seven segment decoded integ
                    default:
                        out10 = 16;
                }

        IOWR_ALTERA_AVALON_PIO_DATA(LED_0_BASE, out0);
        IOWR_ALTERA_AVALON_PIO_DATA(LED_10_BASE, out10);

        return 0;
}
```

Few key marks of my codes:

- Any input greater than 9 was changed to 9
- LED seven segment decoded by using:

```
4'b0000: LEDPin=7'b1000000; // 0 -> 64
4'b0001: LEDPin=7'b1111001; // 1 -> 121
4'b0010: LEDPin=7'b0100100; // 2 -> 36
4'b0011: LEDPin=7'b0110000; // 3 -> 48
4'b0100: LEDPin=7'b0011001; // 4 -> 25
4'b0101: LEDPin=7'b0010010; // 5 -> 18
4'b0110: LEDPin=7'b0000010; // 6 -> 2
4'b0111: LEDPin=7'b1111000; // 7 -> 120
4'b1000: LEDPin=7'b0000000; // 8 -> 0
4'b1001: LEDPin=7'b0010000; // 9 -> 16;
endcase
```

My system was tested and verified my system by my generous TA Xianglong Feng at 12/12/2019