Homework 1 - Basic Nios Setup

Zackary McClamma*†

September 23, 2019

1 Introduction

This homework assignment involved configuring the NIOS II processor in QuartusPrime, adding peripherals(7 Segment Display and Buttons), and creating software to manipulate the peripherals. It being the first homework assignment the main goals were to familiarize myself with the hardware/software development process and the IDE. I was able to successfully configure the processor and create software to accomplish the tasks outlined in the assignment which were to fist display 0-9 on one of the 7 segment displays and, second to poll the state of one of the buttons on the board.

2 System Description

The system consists of three interfaces (Button, 7 Segment Display, and USB Blaster) and the CPU. The USB blaster is used to upload all of the development done in Quartus (VHDL) and Eclipse (C code) to the FPGA. The FPGA uses the VHDL code to generate the processor, memory, and also connect all the peripherals defined in the VHDL code. The C code is loaded into memory where it is read by the processor and the code is executed. This process is defined in 1 below. In the figure the blue arrows indicate VHDL code, the red arrows indicate C code, and the green arrows indicate that both VHDL and C are being transmitted.

*University of Dayton

†Dept. of Electrical and Computer Engineering, University of Dayton, 300 College Park, Dayton, OH 45469-0226, U.S.A. E-mail: mcclammaz1@udayton.edu

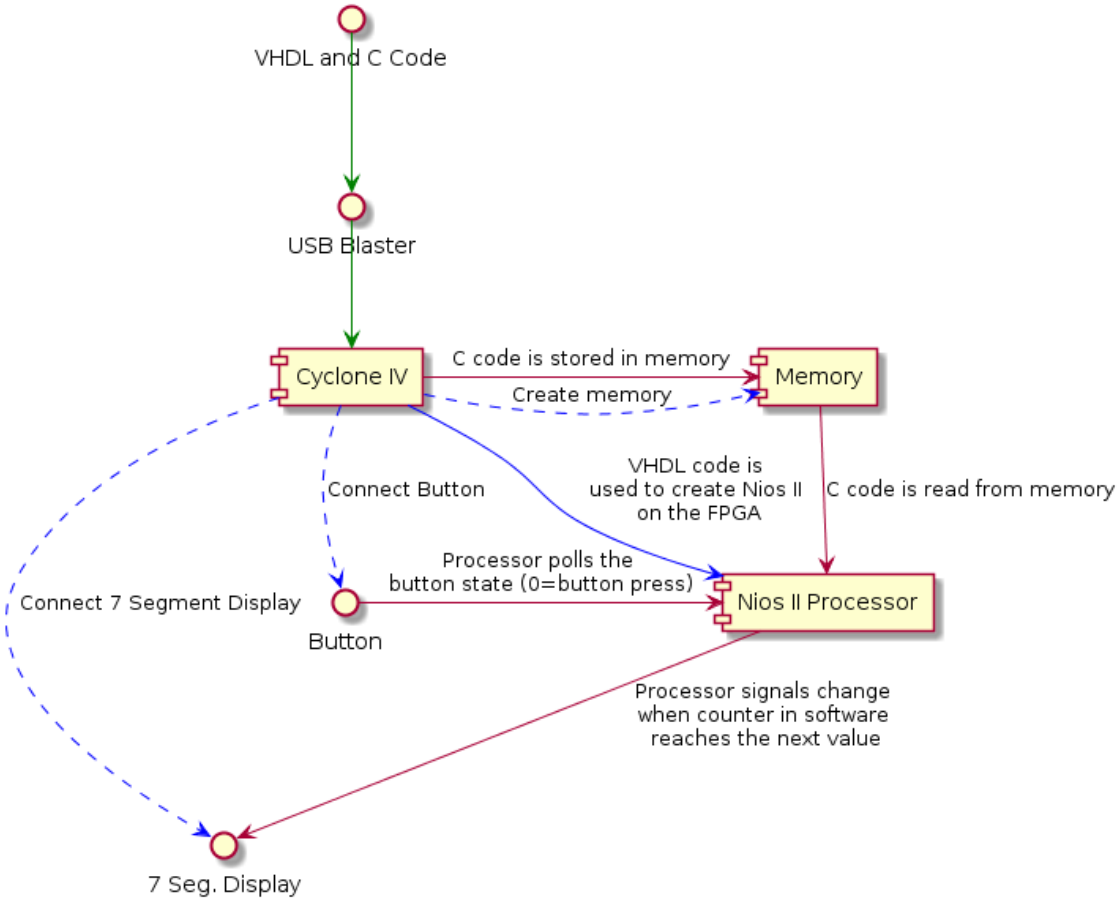


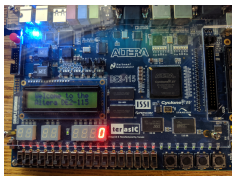
Figure (1) Embedded System Block Diagram

3 Theory of Operation

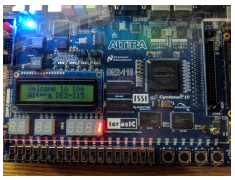
This system has one input (Button) and a set of outputs (7 Segment Display). The button uses a simple polling operation, the software runs in a loop and every iteration checks the status of the button. If the button has not been depressed "Button=1" is displayed in the console, if the button is depressed the console will display "Button=0". In regards to the 7 segment display there are 7 individual LEDs within the display, each one has a corresponding memory address associated with it. IF there is a 1 written to a given address the corresponding LED will be off, if a 0 is written to the address the corresponding LED is turned on. In the software there is an endless loop with a counter that counts from 0 to 9. Every time the counter is incremented the display changes to show the value the counter is at i.e. if the counter is at 3 the seven segment display will show '3'. This is accomplished by writing 0 to the memory addresses that turn the segments in the display on, for example if the counter were to read 8 it would write a 0 to all of the addresses for the display.

4 Results

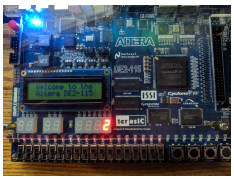
4.1 7 Segment Display



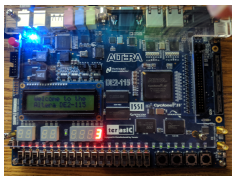
(a) Display 0



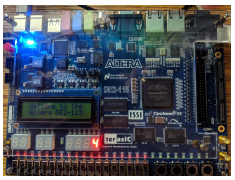
(b) Display 1



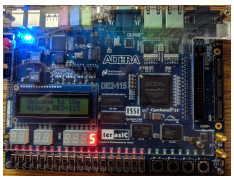
(c) Display 2



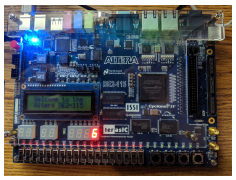
(d) Display 3



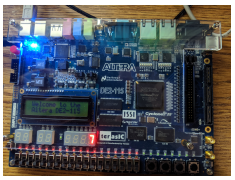
(e) Display 4



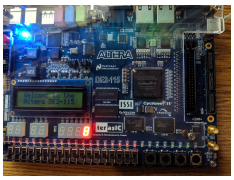
(f) Display 5



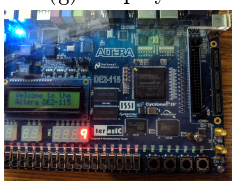
(g) Display 6



(h) Display 7



(i) Display 8



(j) Display 9

4.2 Button Polling

Figure (3) Button Polling Results

A VHDL Code

```
-- ECE532 HW1
-- File: homework1.vhd
-- Author: Zackary McClamma
-- Date: 23-Sep-2019

library ieee;
  use ieee.std_logic_1164.all;

entity homework1 is
  port
  (
    i_clk      : in     std_logic;
```

```
    i_rst_n    : in     std_logic;
         gpio0 : out std_logic;
         gpio1 : out std_logic;
         gpio2 : out std_logic;
         gpio3 : out std_logic;
         gpio4 : out std_logic;
         gpio5 : out std_logic;
         gpio6 : out std_logic;
         i_button : in std_logic;
         o_led : out std_logic

  );
end homework1;

architecture sch of homework1 is

  component homework1_cpu is
  port (
    clk_clk        : in std_logic;
    reset_reset_n : in std_logic;
         gpio0_export : out std_logic;
         gpio1_export : out std_logic;
         gpio2_export : out std_logic;
         gpio3_export : out std_logic;
         gpio4_export : out std_logic;
         gpio5_export : out std_logic;
         gpio6_export : out std_logic;
         button_export : in std_logic;
         led_export              : out std_logic
  );
  end component homework1_cpu;

  signal w_led   : std_logic;

begin

        o_led <= w_led;

  u0 : component homework1_cpu
       port map
  (
    clk_clk        => i_clk,
    reset_reset_n => i_rst_n,
         gpio0_export => gpio0,
         gpio1_export => gpio1,
         gpio2_export => gpio2,
         gpio3_export => gpio3,
         gpio4_export => gpio4,
         gpio5_export => gpio5,
         gpio6_export => gpio6,
```

```
        button_export => i_button,
        led_export            => w_led
  );

end sch;
```

B C Code

B.1 Headers

```
/*
        -- ECE 532 Homework 1
        -- File: counter.h
        -- Date: 23 SEP 2018
        -- Author: Zackary McClamma

*/

#ifndef COUNTER_H_
#define COUNTER_H_

#endif /* COUNTER_H_ */

#define GPIO_0_BASE 0x30000
#define GPIO_1_BASE 0x40000
#define GPIO_2_BASE 0x50000
#define GPIO_3_BASE 0x60000
#define GPIO_4_BASE 0x70000
#define GPIO_5_BASE 0x80000
#define GPIO_6_BASE 0x90000
#define BUTTON_BASE 0xB0000
#define DELAY 500000

typedef struct gr{
        unsigned int data;
        unsigned int dir;
        unsigned int intmask;
        unsigned int edge;
        unsigned int outset;
        unsigned int outclear;
} gpio_regs;

void displayNumber(int number);
void delay(unsigned int n);
unsigned int get_button(void);
```

B.2 Source

```
/*
```

```
        -- ECE 532 Homework 1
        -- File: main.c
        -- Date: 23 SEP 2018
        -- Author: Zackary McClamma

*/

#include <stdio.h>
#include "counter.h"

int main()
{
  printf("Hello from Nios II!\n");
  while (1)
  {
          for(int i = 0; i < 10; i++)
          {
                  displayNumber(i);
                  printf("Button = %d\n", get_button());
                  delay(DELAY);

          }
  }

  return 0;
}

void displayNumber(int num)
{
        int *gpio0 = GPIO_0_BASE;
        int *gpio1 = GPIO_1_BASE;
        int *gpio2 = GPIO_2_BASE;
        int *gpio3 = GPIO_3_BASE;
        int *gpio4 = GPIO_4_BASE;
        int *gpio5 = GPIO_5_BASE;
        int *gpio6 = GPIO_6_BASE;

        *gpio0 = 0x01;
        *gpio1 = 0x01;
        *gpio2 = 0x01;
        *gpio3 = 0x01;
        *gpio4 = 0x01;
        *gpio5 = 0x01;
        *gpio6 = 0x01;

        switch(num)
        {
        case 0:
                *gpio0 = 0x00;
                *gpio1 = 0x00;
```

```
            *gpio2 = 0x00;
            *gpio3 = 0x00;
            *gpio4 = 0x00;
            *gpio5 = 0x00;
            return;
    case 1:
            *gpio1 = 0x00;
            *gpio2 = 0x00;
            return;
    case 2:
            *gpio0 = 0x00;
            *gpio1 = 0x00;
            *gpio3 = 0x00;
            *gpio4 = 0x00;
            *gpio6 = 0x00;
            return;
    case 3:
            *gpio0 = 0x00;
            *gpio1 = 0x00;
            *gpio2 = 0x00;
            *gpio3 = 0x00;
            *gpio6 = 0x00;
            return;
    case 4:
            *gpio1 = 0x00;
            *gpio2 = 0x00;
            *gpio5 = 0x00;
            *gpio6 = 0x00;
            return;
    case 5:
            *gpio0 = 0x00;
            *gpio2 = 0x00;
            *gpio3 = 0x00;
            *gpio5 = 0x00;
            *gpio6 = 0x00;
            return;
    case 6:
            *gpio0 = 0x00;
            *gpio2 = 0x00;
            *gpio3 = 0x00;
            *gpio4 = 0x00;
            *gpio5 = 0x00;
            *gpio6 = 0x00;
            return;
    case 7:
            *gpio0 = 0x00;
            *gpio1 = 0x00;
            *gpio2 = 0x00;
            return;
    case 8:
```

```
                *gpio0 = 0x00;
                *gpio1 = 0x00;
                *gpio2 = 0x00;
                *gpio3 = 0x00;
                *gpio4 = 0x00;
                *gpio5 = 0x00;
                *gpio6 = 0x00;
                return;
        case 9:
                *gpio0 = 0x00;
                *gpio1 = 0x00;
                *gpio2 = 0x00;
                *gpio5 = 0x00;
                *gpio6 = 0x00;
                return;
        }
}

void delay (unsigned int n){
        for(unsigned int i=0;i<n;i++);
}

unsigned int get_button(void){
        gpio_regs *gr = BUTTON_BASE;

        return(gr->data);
}
```