Homework 3 - Frame Based Scheduling

Zackary McClamma*†

October 14, 2019

1 Introduction

In this homework assignment consisted of adding all of the available 7-segment displays (8 total), a switch for each display, and a timer. It also included creating software that uses a timer interrupt to trigger an ISR. The ISR uses a frame based schedule to poll the state of the switches and determine if the value on the 7-segment display will be incremented or decremented. This assignment also required the calculation of the load for each individual task and the total load on the processor.

2 System Design

This system uses a total of 16 peripherals, one clock, and of course the Nios II processor. There are a total of 8 seven segment displays and 8 switches. Each seven segment display is 'assigned' a switch and the state of the switch will determine what the display should output. Every 250ms a timer interrupt is triggered and the ISR executes a frame, the ISR cycles through and increments the frame count every time the interrupt is triggered, when the last frame is triggered the ISR resets the frame count and starts from the first frame (in this scenario there are 4 frames total). The switches have two possible states (on/off i.e. 1 or 0), every interrupt a set of the switches is polled and if the stat of the switch is 1 then the seven segment display is incremented, if it is zero then the display is decremented.

3 Theory of Operation

Below is the block diagram for this assignment it is fairly similar to the previous two assignments with the main difference being in this case that the interrupt is synchronous and is based of of the timer. As can be seen from 1 the timer triggers an interrupt which causes the processor to check the state of the switches (which switches are polled depends on which frame is being executed). Dependent on the state of the switch the corresponding display will be incremented (switch state is 1) or decremented (switch state is 0).

*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

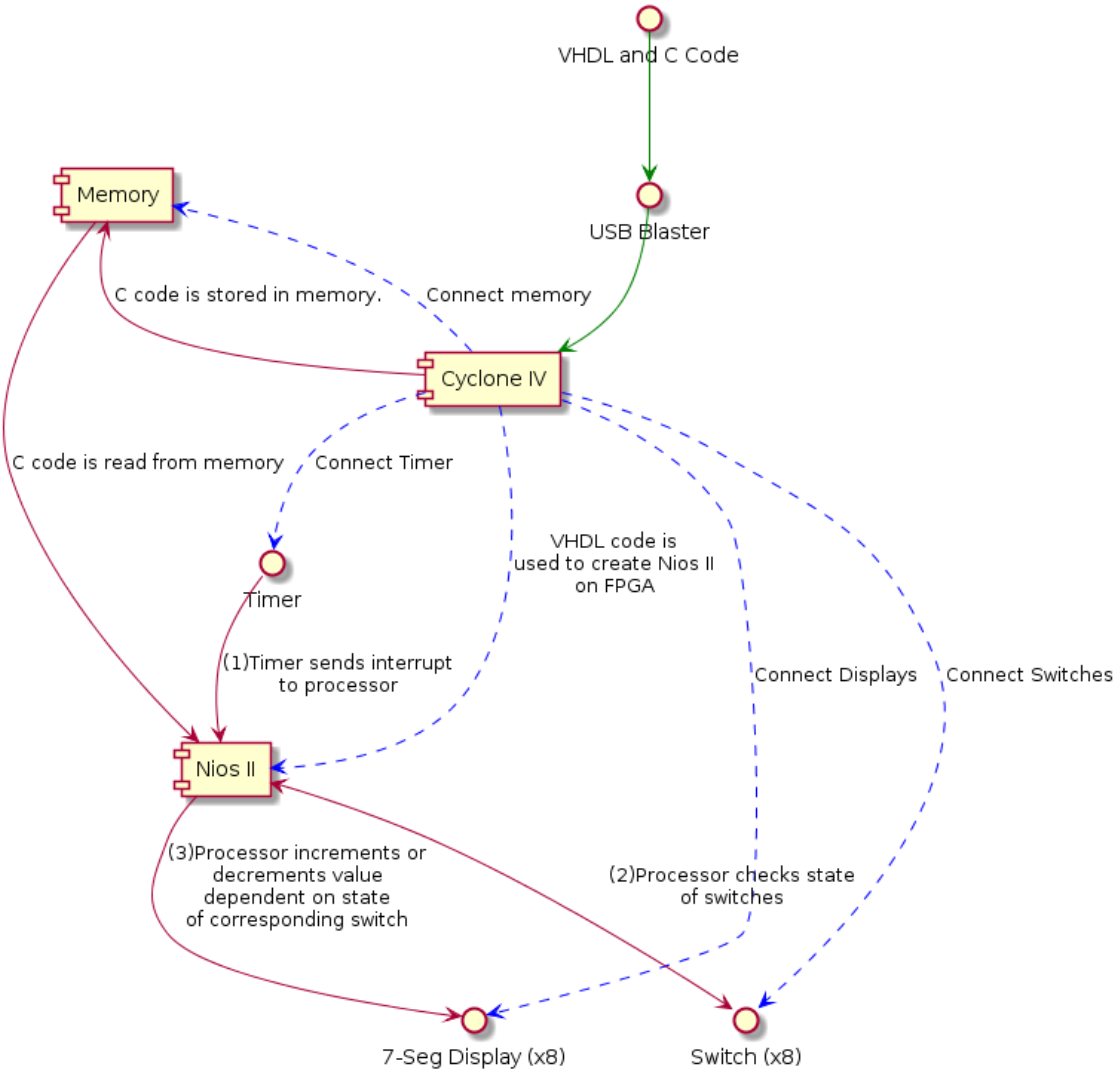

Figure (1) Embedded System Block Diagram

3.1 Tables

Below are a couple of tables outlining the structure of the frames in this project, 1 is the breakdown of each individual frame and 2 is the frame schedule. As you can see from 1 the system load for this program is below 2% this is to be expected as this is a simple program that only consists of a timer interrupt that triggers the reading of a memory address and an increment or decrement of another address.

Table (1) Task Information

Task Number	Rate (Hz)	Duration(ms)	Period(ms)	Load
0	4	1	250	.004
1	4	1	250	.004
2	2	1	500	.002
3	2	1	500	.002
4	2	.5	500	.002
5	2	.5	500	.002
6	1	.5	1000	.001
7	1	.5	1000	.001
System Load				1.8%

Table (2) Frame Schedule

	Frames			
	1	2	3	4
Tasks	0	0	0	0
	1	1	1	1
	2	4	2	4
	3	5	3	5
	6	7		

4 Results

The results of this project are difficult to capture in writing or with images, so I have included a video of the functioning project with the submission of this document. The end result functioned as I expected (as outlined in the assignment description).

5 Conclusion

This assignment was beneficial in understanding how to use timer interrupts with the Nios II processor. This was also the first assignment where I worked with vectored peripherals and I think that it made my code much more condensed and easier to work with. The only issues I have with my final product is the way that I execute the ISR, I was trying to find a way to return to main and run the frames but I had issues with the ISR triggering again before the code could execute. I attempted to disable the ISR in the meantime but to no avail. To circumvent this i put the frame execution code in the ISR and those frames call external functions to check the switches and change the displays. My goal was to use less code in the ISR because I know that when schedules get more complex having too much code in an ISR can cause issues, but in this case executing code in the ISR shouldn't cause any issues, as this is a simple program and is not doing any complex operations.

A
VHDL Code

```
-- ECE532 HW3
-- Zackary McClamma
-- 14-Oct-2019

library ieee;
  use ieee.std_logic_1164.all;

entity homework3 is
  port
  (
                        i_clk            : in  std_logic
:= 'X'; -- clk
                        i_reset_n        : in  std_logic
:= 'X'; -- reset_n
                        disp1       : out std_logic_vector(6 downto 0);
                        disp2       : out std_logic_vector(6 downto 0);
                        disp3      : out std_logic_vector(6 downto 0);
                        disp5       : out std_logic_vector(6 downto 0);
                        disp4       : out std_logic_vector(6 downto 0);
                        disp6       : out std_logic_vector(6 downto 0);
                        disp7       : out std_logic_vector(6 downto 0);
                        disp8       : out std_logic_vector(6 downto 0);
                        switch_pio  : in std_logic_vector(7 downto 0)


  );
end homework3;

architecture sch of homework3 is

  component hw3_cpu is
        port (
            clk_clk           : in  std_logic
:= 'X'; -- clk
            reset_reset_n     : in  std_logic
:= 'X'; -- reset_n
            disp1_export      : out std_logic_vector(6 downto 0);
-- export
            disp2_export      : out std_logic_vector(6 downto 0);
-- export
            disp3_export      : out std_logic_vector(6 downto 0);
-- export
            disp5_export      : out std_logic_vector(6 downto 0);
-- export
            disp4_export      : out std_logic_vector(6 downto 0);
-- export
```

```
            disp6_export       : out std_logic_vector(6 downto 0);
-- export
                                 disp7_export       : out std_logic_vector(6 down
-- export
                                 disp8_export       : out std_logic_vector(6 down
            switch_pio_export : in std_logic_vector(7 downto 0)
-- export
        );
  end component hw3_cpu;

begin

    u0 : component hw3_cpu
        port map (
            clk_clk            => i_clk,            --
clk.clk
            reset_reset_n      => i_reset_n,     --       reset.reset_n
            disp1_export       => disp1,       --       disp1.export
            disp2_export       => disp2,       --       disp2.export
            disp3_export       => disp3,       --       disp3.export
            disp5_export       => disp5,       --       disp5.export
            disp4_export       => disp4,       --       disp4.export
            disp6_export       => disp6,       --       disp6.export
                                  disp7_export       => disp7,
--       disp4.export
            disp8_export       => disp8,       --       disp6.export
            switch_pio_export => switch_pio  -- switch_pio.export
        );
end sch;
```

B C Code

B.1 Headers

```
/*
 * Name: Zackary McClamma
 * Course: ECE 532
 * Assignment: Homework 3
 * Date: 14 OCT 2019
 * File: hw3.h
 *
 * */

#ifndef HW3_H_
#define HW3_H_

#include <stdio.h>
#include <unistd.h>
```

```
#include <sys/alt_irq.h>
#include <sys/alt_timestamp.h>
#include <time.h>
#include "system.h"

#define _250_MS_TICKS 11500000

#define TIMER_IRQ 2

typedef struct str_timer_regs{
        unsigned int status;
        unsigned int control;
        unsigned int periodl;
        unsigned int periodh;
        unsigned int snapl;
        unsigned int snaph;
}timer_regs;

int frame = 0;
int display_values[] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8, 0x00, 0
int display_counter[] = {0,0,0,0,0,0,0,0};
int* display[] = {DISPLAY1_BASE, DISPLAY2_BASE, DISPLAY3_BASE,
                                  DISPLAY4_BASE, DISPLAY5_BASE, DISPLAY6_BASE,
                                  DISPLAY7_BASE, DISPLAY8_BASE};

void increment(int idx);
void decrement(int idx);
void timer_isr(void);
int checkSwitch(int swtch);


#endif /* HW3_H_ */
```

B.2 Source

```
/*
 * Name: Zackary McClamma
 * Course: ECE 532
 * Assignment: Homework 3
 * Date: 14 OCT 2019
 * File: main.c
 *
 * */

#include "hw3.h"

int main()
{
  for (int i = 0; i < 8; i++)
  {
```

```
            *display[i] = display_values[0];
    }

    printf("Hello from Nios II!\n");
    timer_regs *timerPtr = TIMER_0_BASE;

    printf("Registering interrupt...\n");
    alt_ic_isr_register(TIMER_0_IRQ_INTERRUPT_CONTROLLER_ID, TIMER_0_IRQ, (void*)
    alt_ic_irq_enable(TIMER_0_IRQ_INTERRUPT_CONTROLLER_ID, TIMER_0_IRQ);

    printf("Setting timeout period...\n");
    timerPtr->periodl = (_250_MS_TICKS & 0x0000FFFF);
    timerPtr->periodh = (_250_MS_TICKS >> 16);

    printf("Enabling interrupt...\n");
    timerPtr->control = 0x07;

    while(1)
    {
    }
    return 0;
}

void increment(int idx)
{
        display_counter[idx]++;
        if(display_counter[idx] > 9)
                display_counter[idx] = 0;
        *display[idx] = display_values[display_counter[idx]];
}

void decrement (int idx)
{
        display_counter[idx]--;
        if(display_counter[idx] < 0)
                display_counter[idx] = 9;
        *display[idx] = display_values[display_counter[idx]];
}

int checkSwitch(int swtch)
{
        clock_t start = clock();
        clock_t end;
        int elapsed_msec;
        int* base = SWITCHES_BASE;

        //Check switch state
        if(*base & (1 << (swtch)))
        {
                increment(swtch);
```

```
        }
        else
        {
                decrement(swtch);
        }

        //Ensure task runs for the full duration
        end = clock();
        elapsed_msec = (end - start)/CLOCKS_PER_SEC;
        switch(swtch)
        {
        case 0:
                if(elapsed_msec < 1000)
                {
                        usleep(1000 - elapsed_msec);
                }
                return(1);
        case 1:
                if(elapsed_msec < 1000)
                        usleep(1000 - elapsed_msec);
                return(1);
        case 2:
                if(elapsed_msec < 1000)
                        usleep(1000 - elapsed_msec);
                return(1);
        case 3:
                if(elapsed_msec < 1000)
                        usleep(1000 - elapsed_msec);
                return(1);
        case 4:
                if(elapsed_msec < 500)
                        usleep(500 - elapsed_msec);
                return(1);
        case 5:
                if(elapsed_msec < 500)
                        usleep(500 - elapsed_msec);
                return(1);
        case 6:
                if(elapsed_msec < 500)
                        usleep(500 - elapsed_msec);
                return(1);
        case 7:
                if(elapsed_msec < 500)
                        usleep(500 - elapsed_msec);
                return(1);
        }
        return(1);
}

void timer_isr(void)
```

```
{
        timer_regs *timerPtr = TIMER_0_BASE;

          switch(frame)
          {
                  case 0:
                          checkSwitch(0);
                          checkSwitch(1);
                          checkSwitch(2);
                          checkSwitch(3);
                          checkSwitch(6);
                          break;
                  case 1:
                          checkSwitch(0);
                          checkSwitch(1);
                          checkSwitch(4);
                          checkSwitch(5);
                          checkSwitch(7);
                          break;
                  case 2:
                          checkSwitch(0);
                          checkSwitch(1);
                          checkSwitch(2);
                          checkSwitch(3);
                          break;
                  case 3:
                          checkSwitch(0);
                          checkSwitch(1);
                          checkSwitch(4);
                          checkSwitch(5);
                          break;
          }
          frame++;
          if(frame == 4)
                  frame = 0;

        timerPtr->status = 0x01;

}
```