Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет им. Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ – Информатика, искусственный интеллект и системы управления
КАФЕДРА – Информационные системы и телекоммуникации

Отчет к лабораторной работе №1

по курсу "Программное обеспечение встроенных систем"

направления 09.04.02

"Создание простой системы в Platform Designer в пакете Quartus Prime"

Выполнил:
студент группы ИУ3-11М
Щесняк С.С.

Проверил:
Федоров С.В.

Москва 2022

Оглавление

1. Цель работы

Изучить средства создания систем на кристалле на основе процессора Nios II фирмы Intel в средстве Platform Designer. Освоить методику создания и конфигурации систем на кристалле. Реализовать и отладить программное обеспечение системы на кристалле. Освоить методику отладки проекта с применением программных и аппаратных средств. Получить навыки работы с документацией производителя.

2. Задание

1. Модифицируйте исходную программу в соответствии с индивидуальным заданием:
    a. Нажатие кнопки KEY3 должно зажигать все светодиоды, KEY0 – гасить все светодиоды, нажатие центральных кнопок KEY2 и KEY1 должно гасить светодиоды по очереди слева и справа по одному на каждое нажатие.
2. Реализуйте обработку нажатия кнопок и формирование значений на светодиодах по прерыванию. Используйте справочные материалы в каталоге лабораторной работы. Пример реализации обработчика прерывания от кнопок и его регистрации приведен в [1], раздел 8 “Exception Handling”.

3. Ход работы

1. Листинг кода

```
#include <stdio.h>
#include <unistd.h>
#include "altera_avalon_pio_regs.h"
#include "system.h"
#define NONE_PRESSED 0xF // Value read from button PIO when no buttons
pressed
#define DEBOUNCE 30000 // Time in microseconds to wait for switch
debounce
#define KEY0_PRESSED 0xE
#define KEY1_PRESSED 0xD
#define KEY2_PRESSED 0xB
#define KEY3_PRESSED 0x7

int main(void)
{
```

```
    int buttons; // Use to hold button pressed value
    int led; //Use to hold current led array state
    int prev_led; //Use to hold previous led array state
    int left=1; //Use to switch ends of led array
    int mask; //Use to single out diode
    while (1)
    {
        buttons = IORD_ALTERA_AVALON_PIO_DATA(BUTTONS_BASE); // read
buttons via pio
        if (buttons != NONE_PRESSED){ //if a button is pressed
            switch (buttons) {
                //shut off every led
                case(KEY0_PRESSED): {
                        led = 0x00;
                        break;
                }
                /*shut off a led on the side of the led array
                 * that is given by the current value of left*/
                case(KEY1_PRESSED):case(KEY2_PRESSED): {
                    prev_led=led; //remember current led array state
                    /*set mask to single out a diode on the end of
array
                     * that is given by the current value of left*/
                    if(left){ mask = 0x80; }
                    else{ mask = 0x01; }
                    /*attempt to turn off a diode until one at the
given
                     * end of the led array is turned off*/
                    while (led == prev_led){
                        led = led & ~mask;
                        if (left) { mask = mask>>1; }
                        else { mask = mask<<1; }
                    }
                    //switch end of the led array
                    if (left) left=0;
                    else left=1;
                    break;
                }
                //turn on every led
                case(KEY3_PRESSED):{
                    led = 0xFF;
                    break;
                }
            }
            IOWR_ALTERA_AVALON_PIO_DATA(GREEN_LED_BASE, led); // write
new value to pio
        }
    }
} // end
```

2. Листинг кода

```
#include <stdio.h>
#include <unistd.h>
#include "altera_avalon_pio_regs.h"
#include "system.h"

#define NONE_PRESSED 0xF  // Value read from button PIO when no buttons
pressed
#define DEBOUNCE 30000  // Time in microseconds to wait for switch
debounce

#define KEY0_PRESSED 1
```

```
#define KEY1_PRESSED 2
#define KEY2_PRESSED 4
#define KEY3_PRESSED 8

volatile int edge_capture; //Use to store the code of the button
pressed

/* Handle button interrupt */
#ifdef ALT_ENHANCED_INTERRUPT_API_PRESENT
static void handle_button_interrupts(void* context)
#else
static void handle_button_interrupts(void* context, alt_u32 id)
#endif
{
   /* Cast context to edge_capture's type. It is important that this
   be declared volatile to avoid unwanted compiler optimization. */
   volatile int* edge_capture_ptr = (volatile int*) context;
   /*
   * Read the edge capture register on the button PIO.
   * Store value.
   */
   *edge_capture_ptr =
   IORD_ALTERA_AVALON_PIO_EDGE_CAP(BUTTONS_BASE);
   /* Write to the edge capture register to reset it. */
   IOWR_ALTERA_AVALON_PIO_EDGE_CAP(BUTTONS_BASE, 0);
   /* Read the PIO to delay ISR exit. This is done to prevent a
   spurious interrupt in systems with high processor -> pio
   latency and fast interrupts. */
   IORD_ALTERA_AVALON_PIO_EDGE_CAP(BUTTONS_BASE);
}

/* Initialize the button_pio. */
static void init_button_pio()
{
   /* Recast the edge_capture pointer to match the
   alt_irq_register() function prototype. */
   void* edge_capture_ptr = (void*) &edge_capture;
   /* Enable all 4 button interrupts. */
   IOWR_ALTERA_AVALON_PIO_IRQ_MASK(BUTTONS_BASE, 0xf);
   /* Reset the edge capture register. */
   IOWR_ALTERA_AVALON_PIO_EDGE_CAP(BUTTONS_BASE, 0x0);
   /* Register the ISR. */
   #ifdef ALT_ENHANCED_INTERRUPT_API_PRESENT
   alt_ic_isr_register(BUTTONS_IRQ_INTERRUPT_CONTROLLER_ID,
   BUTTONS_IRQ,
   handle_button_interrupts,
   edge_capture_ptr, 0x0);
   #else
   alt_irq_register( BUTTONS_IRQ,
   edge_capture_ptr,
   handle_button_interrupts );
   #endif
}

int main(void)
{
    int buttons; // Use to hold button pressed value
    int led; //Use to hold current led array state
    int prev_led; //Use to hold previous led array state
    int left=1; //Use to switch ends of led array
    int mask; //Use to single out diode

   init_button_pio(); //call the function that sets up and enables
```

```
interrupts from buttons
   IOWR_ALTERA_AVALON_PIO_DATA(GREEN_LED_BASE,led); // write initial
value to pio

   while (1)
   {
      if (edge_capture)  // if button pressed
      {
         switch (edge_capture) {
                //shut off every led
                case(KEY0_PRESSED): {
                        led = 0x00;
                        break;
                }
                /*shut off a led on the side of the led array
                 * that is given by the current value of left*/
                case(KEY1_PRESSED):case(KEY2_PRESSED): {
                    prev_led=led; //remember current led array state
                    /*set mask to single out a diode on the end of
array
                     * that is given by the current value of left*/
                    if(left){ mask = 0x80; }
                    else{ mask = 0x01; }
                    /*attempt to turn off a diode until one at the
given
                     * end of the led array is turned off*/
                    while (led == prev_led){
                        led = led & ~mask;
                        if (left) { mask = mask>>1; }
                        else { mask = mask<<1; }
                    }
                    //switch end of the led array
                    if (left) left=0;
                    else left=1;
                    break;
                }
                //turn on every led
                case(KEY3_PRESSED):{
                    led = 0xFF;
                    break;
                }
            }

         edge_capture = 0; //reset pressed button
         IOWR_ALTERA_AVALON_PIO_DATA(GREEN_LED_BASE,led);  // write new
value to pio

         /* Switch debounce routine
            Wait for small delay after intial press for debouncing
            Wait for release of key
            Wait for small delay after release for debouncing */

         usleep (DEBOUNCE);
         while (buttons != NONE_PRESSED)  // wait for button release
            buttons = IORD_ALTERA_AVALON_PIO_DATA(BUTTONS_BASE);   //
update
         usleep (DEBOUNCE);
      }
   }
}  // end
```

4. Вывод

В результате выполнения лабораторной работы были изучены средства создания систем-на-кристалле на основе процессора NiosII фирмы Intel в средстве Platform Designer, освоены методики создания и конфигурации систем-на-кристалле, реализованы и отлажены программы для системы-на-кристалле Altera DE2-115, освоены методики отладки проекта с применением программных и аппаратных средств, получены практические навыки работы с документацией производителя.