Phần 2 – Nios II

Lab 2: Điều khiển xuất nhập dữ liệu

Mục đích:

Tìm hiểu cách tạo hệ thống Nios II với Parallel Input/Output (PIO) – sử dụng SWs/KEYs/LEDs trên Kit DE2 để nhập xuất dữ liệu.

Phần 1

Tạo một Nios system dùng cho việc nhập và xuất dữ liệu theo yêu cầu sau:

- Nios II/s processor
- On-chip memory RAM có size là 32 Kbytes, width là 32 bits
- Một PIO input 8 bits (đặt tên là: new_number, tương ứng với SW [7:0])
- Một PIO output 8 bits (đặt tên là: green_LEDs, tương ứng với LEDG [7:0])
- Một PIO output 16 bits (đặt tên là: red_LEDs, tương ứng với LEDR [15:0])

Chú ý: SOPC Builder sẽ tự động đặt tên 3 PIO vừa tạo ra là pio_0, pio_1, pio_2. Ta nên đổi tên cho dễ hiểu và gần với ý nghĩa của chúng, như new_number, green_LEDs và red_LEDs.

Phần 2

Viết chương trình **cộng tích lũy** một số 8 bits được nhập vào bằng 8 SWs trên DE2. Dùng 8 đèn LEDG để biểu diễn số tương ứng với số trên các SW nhập vào, dùng 16 LEDR để biểu diễn kết quả tổng tính được. Chương trình như sau:

Hình 1. Based address của các thành phần

```
.include "nios_macros.s"
.equ NEW_NUMBER,   0x11000
.equ GREEN_LEDS,   0x11010
.equ RED_LEDS ,    0x11020
.text
```

Lưu ý kiểm tra: giá trị của new_number, green_leds, red_leds tại hai nơi này (trong cột Base của SOPC Builder và assembly code) phải giống nhau

```
.global _start
_start:
                    add         r17, r0, r0
                    movia   r8,     NEW_NUMBER
                    movia   r9,     GREEN_LEDS
                    movia   r10,    RED_LEDS
    MAIN_LOOP:
                    ldwio   r16, 0(r8)
                    stwio   r16, 0(r9)
                    add     r17, r17, r16
                    stwio   r17, 0(r10)
                   br       MAIN_LOOP
.end
```

Yêu cầu: Sinh viên dùng Altera Monitor để nạp chương trình trên vào Nios system đã tạo trong phần 1 và kiểm tra chương trình đã chạy đúng chưa, ghi lại kết quả thí nghiệm.

Lưu ý: *Phần này chương trình phải chạy từng bước. Nếu chạy liên tục thì hiện tượng gì sẽ xảy ra, giải thích tại sao.*

Phần 3

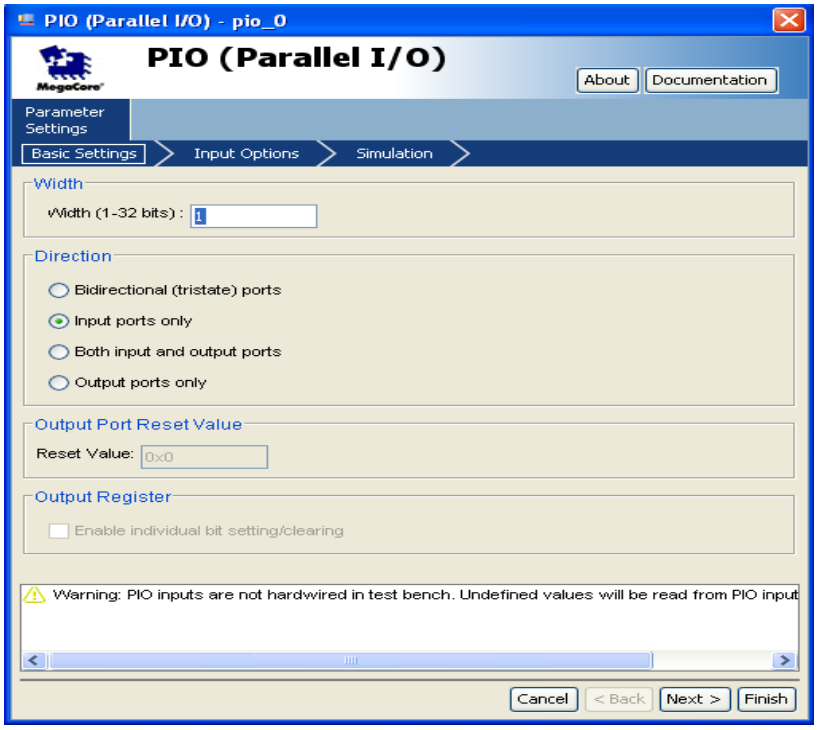
Mục đích của phần này là tạo một cờ hiệu để báo cho nios processor biết khi nào giá trị của input sẽ được nhận vào. KEY[1] trên Kit DE2 được sử dụng làm cờ này.

Chương trình tính tổng tích lũy lúc này sẽ được chạy liên tục (thay vì từng bước như phần 2), khi KEY[1] được nhấn và thả, thì tổng tích lũy sẽ cộng thêm giá trị input từ SW[7:0]; còn nếu không thì dù có thay đổi giá trị của input từ SW[7:0], giá trị này cũng sẽ không tác động vào chương trình, giá trị tổng tích lũy sẽ không thay đổi.

Thêm vào đó, giá trị tổng tích lũy sẽ được hiển thị vừa trên LEDR vừa **trên LED-7-đoạn**.

Các bước thực hiện như sau:

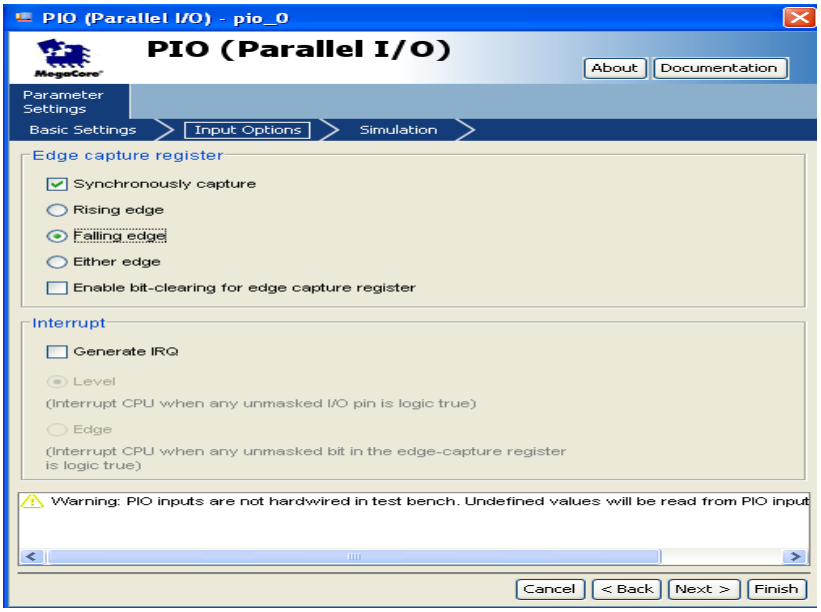
Bước 1. Từ phần 2, thoát khỏi Altera Monitor Program, quay về SOPC Builder, thêm vào hệ thống một input PIO.



Hình 1. Thiết lập thông số cho PIO

PIO mới thêm vào này có các thông số như sau:

- Tại tab "Basic Setting", các thông số thiết lập giống hình 1.
  - Width: 1bits
  - Direction: Input ports only
- Tại tab "Input Options", các thông số thiết lập giống như hình 2.
  - Chọn "Synchronously capture", "Falling Edge" rồi bấm "Finish"



Hình 2. Thiết lập thông số đồng bộ cho cổng nhập (Input Port)

Bước 2. Đổi tên PIO này thành "flag", lưu ý "flag" phải được kết nối (dấu đen như khoanh tròn trên hình 3), chọn clk

Hình 3. Đổi tên PIO mới trên cửa sổ SOPC Builder

Bước 3. Generate hệ thống nios

Bước 4. Quay về QuartusII, chỉnh sửa lại file .v như bảng bên dưới; sau đó biên dịch hệ thống và nạp xuống kit

Bước 5. **Viết lại file .s để chương trình hoạt động sao cho tổng tích lũy chỉ được cộng thêm mỗi khi KEY[1] được nhấn và thả.**

Chú ý:

Để viết được code assembly, sinh viên cần lưu ý:

- ***Vùng nhớ dành cho KEY[1] là vùng nào?***
- ***Khi KEY[1] được nhấn, vùng nhớ sẽ có giá trị bao nhiêu và ngược lại khi KEY[1] được thả ra, vùng nhớ sẽ có giá trị bao nhiêu.***

```
Module nios_system_lab2 (CLOCK_50,KEY,SW,LEDR,LEDG,HEX0,HEX1,HEX2,HEX3);
    // Inputs
    input           CLOCK_50;
    input   [3:0]   KEY;
    input   [17:0]  SW;
    // Outputs
    output  [17:0]  LEDR;
    output  [8:0]   LEDG;
    output  [6:0]   HEX0;
    output  [6:0]   HEX1;
    output  [6:0]   HEX2;
    output  [6:0]   HEX3;
    wire    [15:0]  SUM;
    assign LEDR[15:0] = SUM;
    nios_system the_nios_system (
      .clk (CLOCK_50),
      .reset_n (KEY[0]),
      .in_port_to_the_new_number(SW[7:0]),
      .in_port_to_the_flag (KEY[1]),
      .out_port_from_the_green_LEDs (LEDG[7:0]),
      .out_port_from_the_red_LEDs (SUM)
    );
    Hexadecimal_To_Seven_Segment Digit0 (
                .hex_number         (SUM[3:0]),
              .seven_seg_display   (HEX0)
    );
    Hexadecimal_To_Seven_Segment Digit1 (
                .hex_number         (SUM[7:4]),
                .seven_seg_display (HEX1)
    );
    Hexadecimal_To_Seven_Segment Digit2 (
              .hex_number         (SUM[11:8]),
              .seven_seg_display (HEX2)
    );
    Hexadecimal_To_Seven_Segment Digit3 (
            .hex_number         (SUM[15:12]),
            .seven_seg_display  (HEX3)
    );
endmodule
```

```
module Hexadecimal_To_Seven_Segment (hex_number, seven_seg_display);
    // Inputs
    input           [3:0]   hex_number;
    // Outputs
    output          [6:0]   seven_seg_display;
    assign seven_seg_display =
            ({7{(hex_number == 4'h0)}} & 7'b1000000) |
            ({7{(hex_number == 4'h1)}} & 7'b1111001) |
            ({7{(hex_number == 4'h2)}} & 7'b0100100) |
            ({7{(hex_number == 4'h3)}} & 7'b0110000) |
            ({7{(hex_number == 4'h4)}} & 7'b0011001) |
            ({7{(hex_number == 4'h5)}} & 7'b0010010) |
            ({7{(hex_number == 4'h6)}} & 7'b0000010) |
            ({7{(hex_number == 4'h7)}} & 7'b1111000) |
            ({7{(hex_number == 4'h8)}} & 7'b0000000) |
            ({7{(hex_number == 4'h9)}} & 7'b0010000) |
            ({7{(hex_number == 4'hA)}} & 7'b0001000) |
            ({7{(hex_number == 4'hB)}} & 7'b0000011) |
            ({7{(hex_number == 4'hC)}} & 7'b1000110) |
            ({7{(hex_number == 4'hD)}} & 7'b0100001) |
            ({7{(hex_number == 4'hE)}} & 7'b0000110) |
            ({7{(hex_number == 4'hF)}} & 7'b0001110);
    endmodule
```

Yêu cầu:

- Sinh viên hiểu và giải thích ý nghĩa của file .v
- Sinh viên phải viết được code assembly để điều kiển chương trình và giải thích ý nghĩa từng đoạn code thực hiện chức năng gì.