BÀI 2. GIAO TIẾP I/O ĐƠN GIẢN

1. Mục tiêu

Thông qua bài thực hành này, sinh viên sẽ hiểu rõ:

- Cấu trúc bộ PIO (Parallel Input/Output) của Intel (các thanh ghi, chức năng …).
- Cách xây dựng hệ thống phần cứng với PIO dùng Platform Designer để thực hiện chức năng I/O đơn giản.
- Cách truy xuất tới khối phần cứng PIO đã thiết kế trên chương trình C thông qua Nios II - Eclipse.

2. Phần lý thuyết

2.1. Giới thiệu

- Khối PIO (được cung cấp sẵn bởi Intel) có chức năng thực hiện các giao tiếp IO, thu nhận dữ liệu từ (switches, nút nhấn, …) hoặc xuất dữ liệu ra ngoài (leds đỏ, leds xanh, …).
- Module PIO có thể được cấu hình để hoạt động ở một trong bốn chế độ sau:
  - Input: Thu nhận dữ liệu từ các tín hiệu bên ngoài.
  - Output: Xuất dữ liệu ra các tín hiệu bên ngoài
  - Bidir: Vừa thu nhận dữ liệu, vừa xuất dữ liệu (không đồng thời)
  - InOut: Vừa thu nhận dữ liệu, vừa xuất dữ liệu (đồng thời).
- CPU Nios sẽ điều khiển khối PIO thông qua cách đọc hoặc ghi lên các thanh ghi của PIO thông qua giao tiếp chuẩn Avalon Bus.
- Độ dài các thanh ghi n từ 1 đến 32 bits và hướng của dữ liệu được cấu hình bởi người dùng thông qua công cụ Platform Designer.

2.2. Thanh ghi của PIO

- PIO bao gồm 4 thanh ghi chính được mô tả ở bảng 1.

Offset	Register		R/W	Description
0	data	read access	R	Data value currently on PIO inputs
		write access	W	New value to drive on PIO outputs
1	direction		R/W	Individual direction control for each I/O port. A value of 0 sets the direction to input; 1 sets the direction to output.
2	interruptmask		R/W	IRQ enable/disable for each input port. Setting a bit to 1 enables interrupts for the corresponding port.
3	edgecapture		R/W	Edge detection for each input port.
4	outset		W	Specifies which bit of the output port to set. Outset value is not stored into a physical register

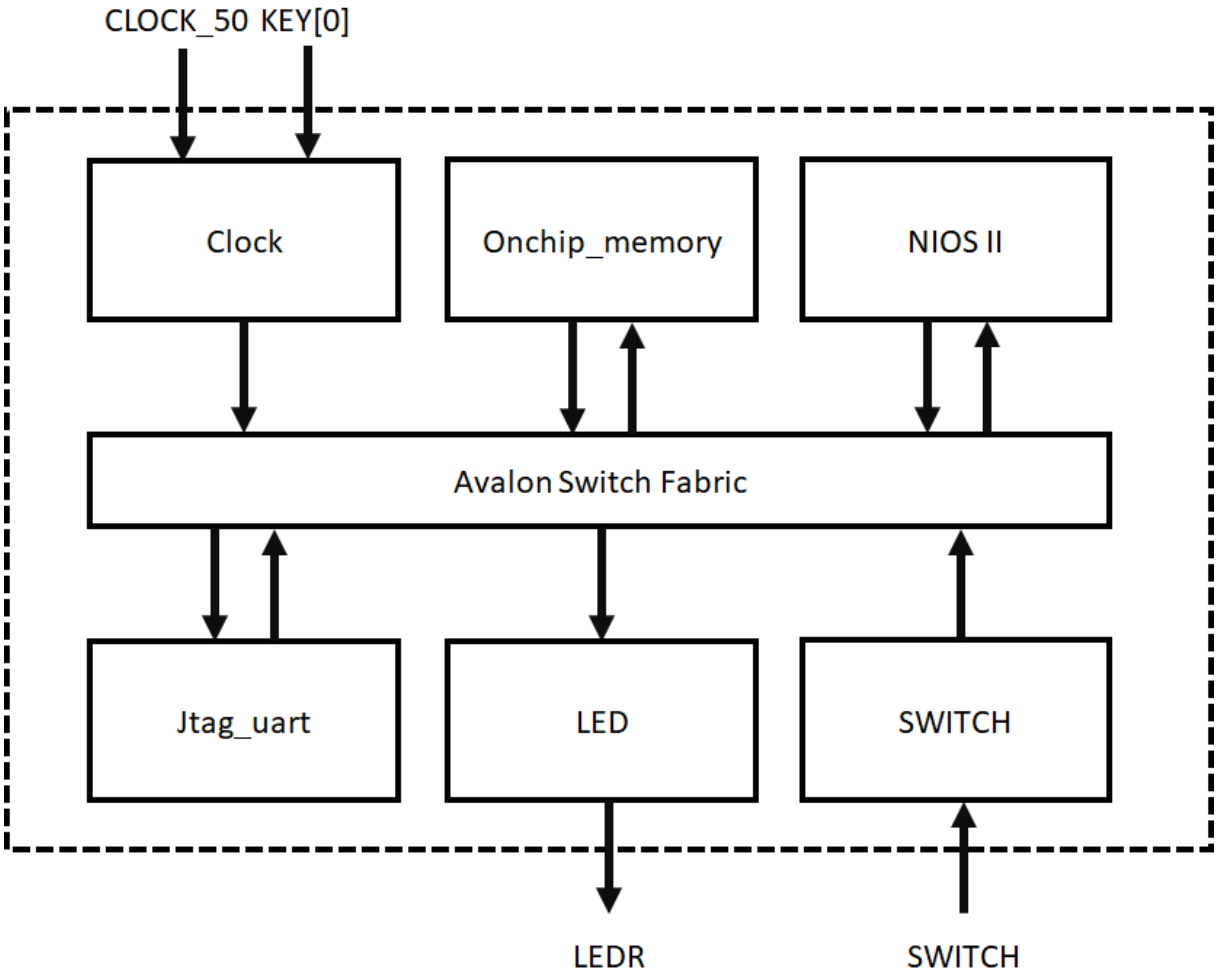
			in the IP core. Hence it's value is not reserve for future use.
5	outclear	W	Specifies which output bit to clear. Outclear value is not stored into a physical register in the IP core. Hence it's value is not reserve for future use.

- Chức năng của mỗi thanh ghi:
  - **Data Register**: thanh ghi chứa dữ liệu thu vào (chế độ Input) hoặc xuất ra (chế độ Output).
  - **Direction Register**: thanh ghi hướng của dữ liệu (chế độ Bidir).
  - **Interrupt-mask Register**: thanh ghi cho phép ngắt (chế độ Input).
  - **Edge-capture Register**: thanh ghi báo hiệu có cạnh lên/cạnh xuống của tín hiệu ngõ vào (chế độ Input).
- Mỗi thanh ghi của PIO có thể được truy xuất như một vị trí nhớ trong bộ nhớ. Địa chỉ để truy xuất các thanh ghi sẽ là tổng của Base Address và Offset.
  - ✓ Base Address: địa chỉ của khối PIO. Base Address được gán tự động thông qua Platform Designer.
  - ✓ Offset: Vị trí tương đối của các thanh ghi so với thanh ghi đầu tiên (Data Register). Địa chỉ của thanh ghi Direction, Interrupt-mask và Edge-capture sẽ có offset lần lượt là 4, 8, và 12 (trong trường hợp thanh ghi PIO là 32 bits hay 4 bytes).
  - ✓ Địa chỉ cuối cùng của từng thanh ghi sẽ theo công thức: Address = Base Address + Offset.
  - ✓ Ví dụ: PIO 32 bit được SoPC Builder gán Base Address là 0x11000 → địa chỉ thanh ghi Data là 0x11000 (offset = 0), của thanh ghi Direction là 0x11004 (offset = 4), của Interrupt-mask là 0x11008 (offset = 8) và thanh ghi Edge-capture sẽ có địa chỉ 0x1100c (offset = 12).

3. Phần thực hành

3.1. Tổng quan hệ thống

- Trong giao diện phần mềm Platform Designer, ta xây dựng phần cứng như hình 1.



Hình 1. Tổng quan hệ thống.

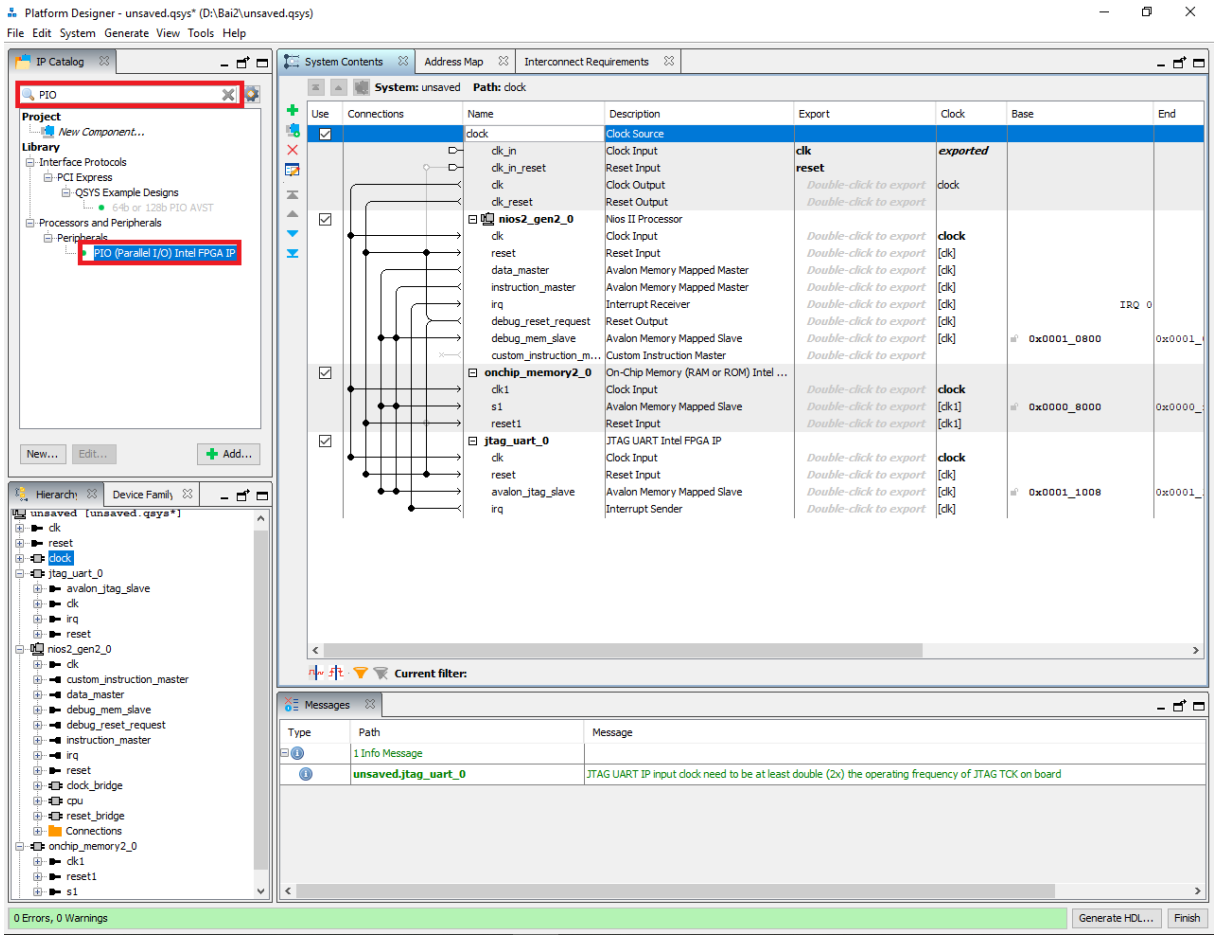
3.2. Xây dựng phần cứng

3.2.1. Tạo project

- Tạo project theo hướng dẫn ở bài 1. Trong đó, tạo project Quartus tên là "lab2". Lưu ý đường dẫn thư mục project không được có khoảng trắng. Chọn Family là Cyclone IV E, device là EP4CE115F29C7.

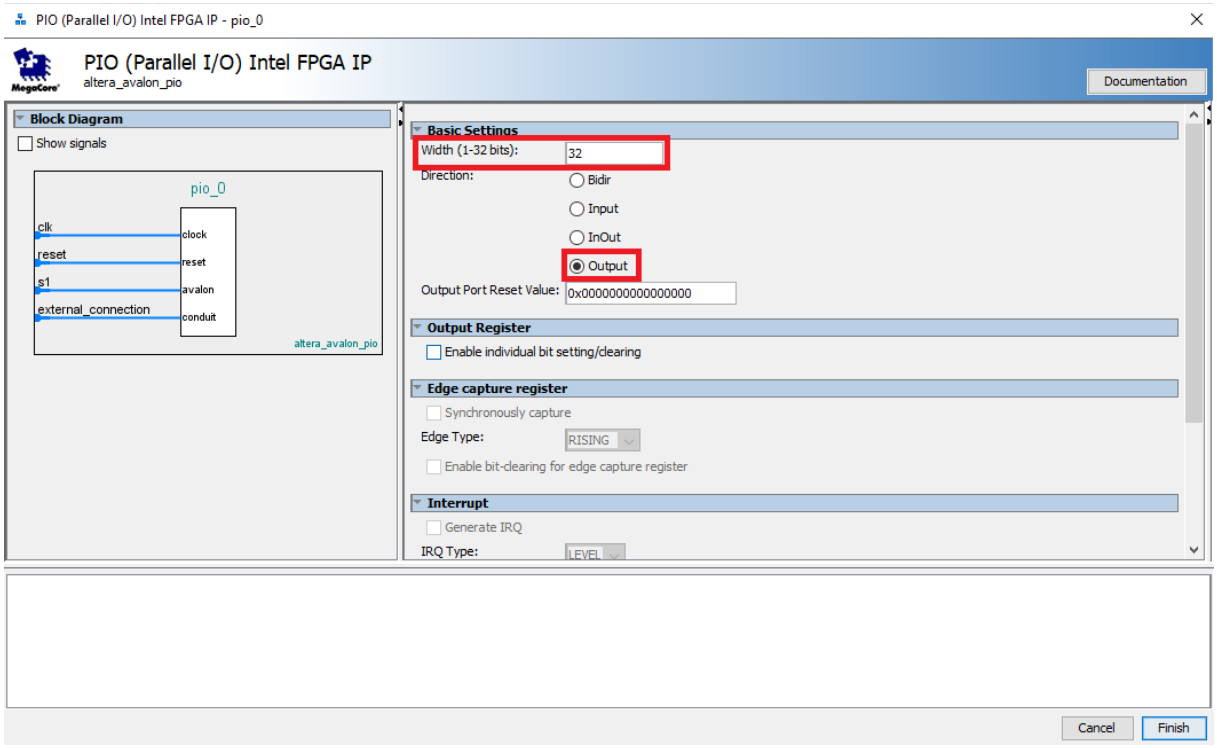
3.2.2. Xây dựng hệ thống phần cứng

- Xây dựng phần cứng theo quy trình ở bài 1. Trong bày này chỉ trình bày ngắn gọn quy trình tạo phần cứng trên Platform Designer, tạo file top-level và tổng hợp phần cứng.
- Trên hệ thống phần cứng như bài 1, thêm mô đun PIO bằng cách gõ PIO vào ô tìm kiếm và nhấp đúp chuột trái để thêm mô đun vào hệ thống, hình 2.



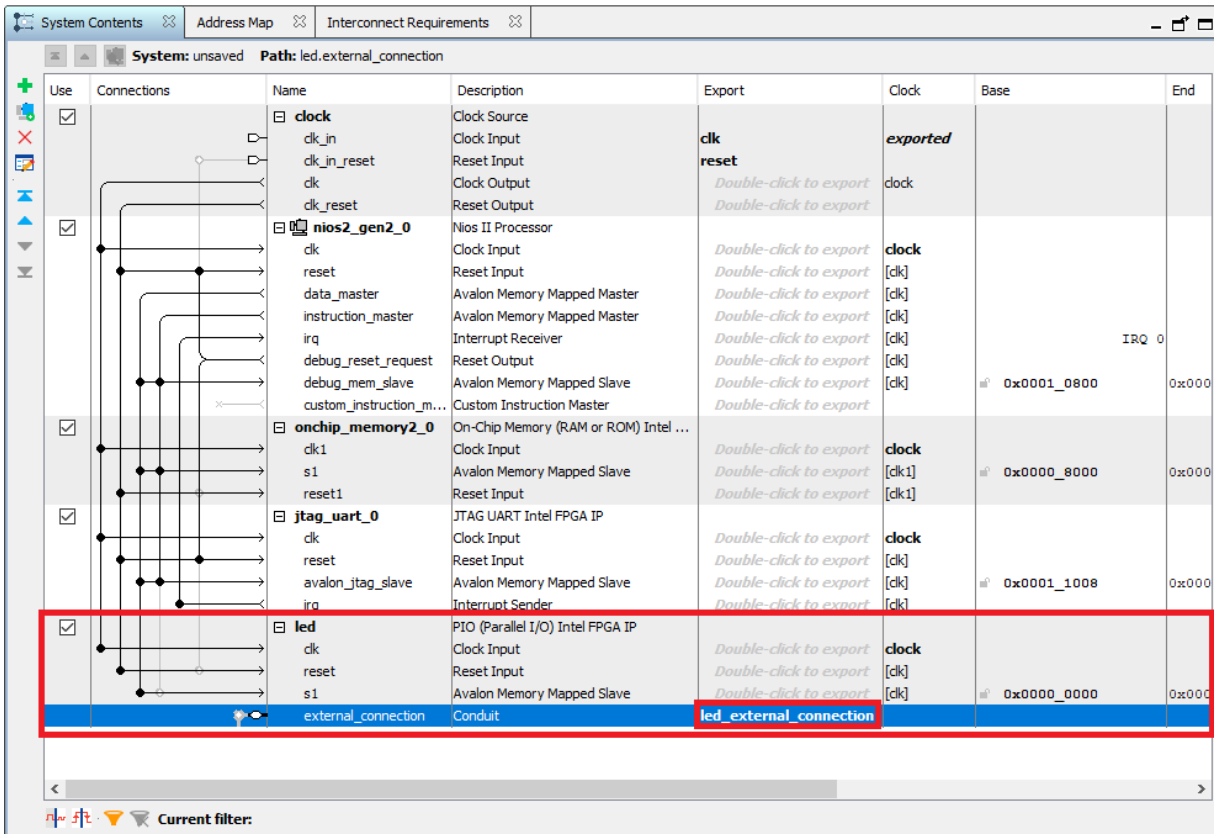
Hình 2. Thêm mô đun PIO.

- Khi thêm mô đun PIO, sẽ xuất hiện hợp thoại cấu hình PIO, cấu hình là độ rộng dữ liệu là 32 bits, hướng là output, hình 3.



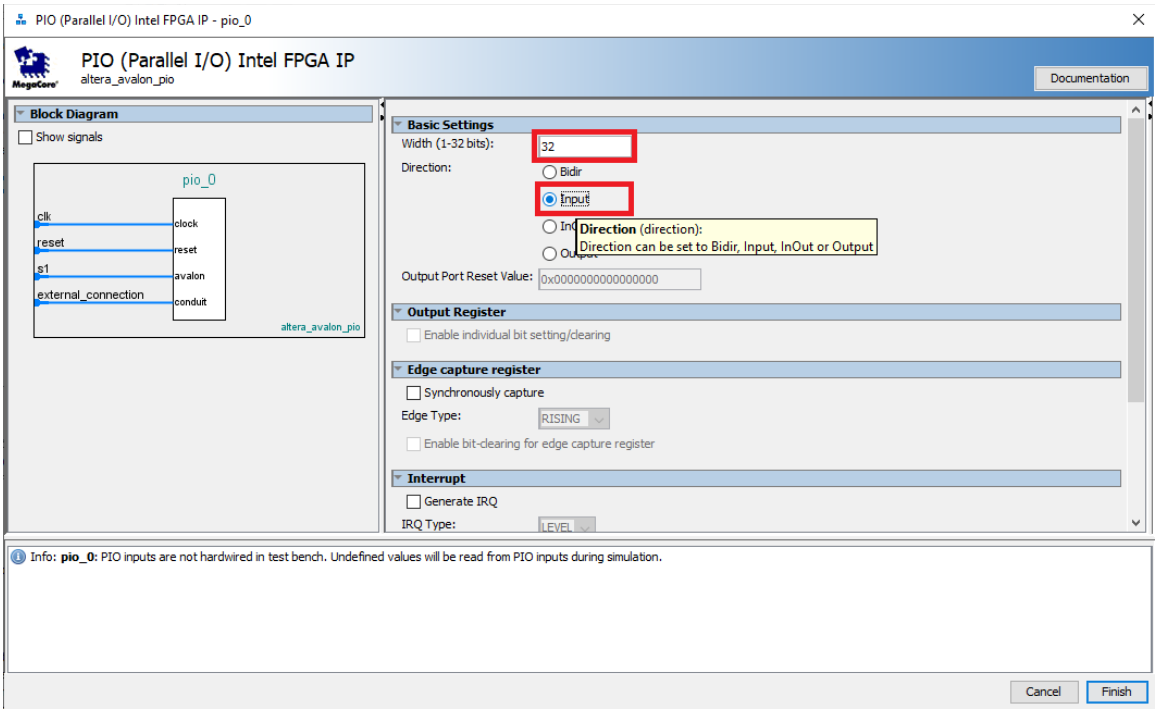
Hình 3. Cấu hình mô đun PIO.

- Sau khi thêm, đổi tên pio_0 thành **led** và kết nối vào hệ thống như hình 4. Lưu ý ở cột Export, ta nhấp đúp chuột trái để xuất tín hiệu ra ngoài.

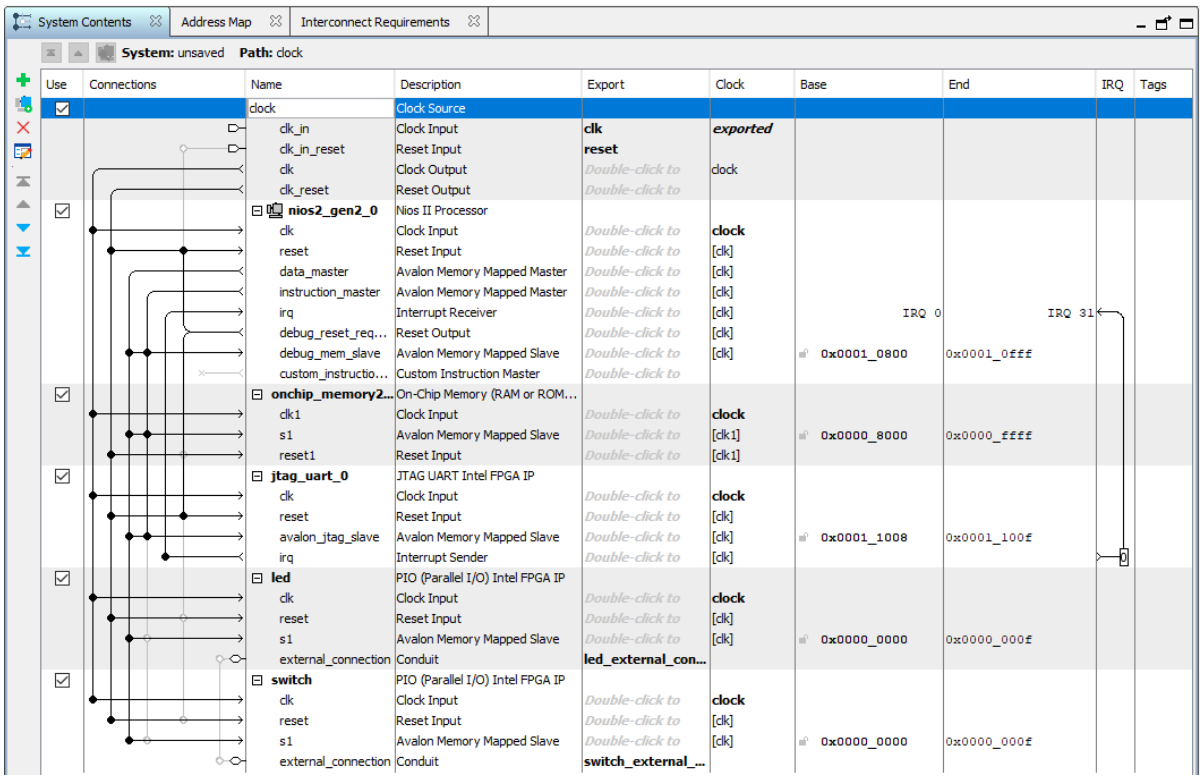


Hình 4. Kết quả thêm mô đun PIO-led.

- Tương tự, thêm một mô đun PIO nữa với độ rộng là 32 bits, hướng là input và đặt tên là **switch**. Hình 5 thể hiện việc cấu hình và hình 6 là kết quả của việc thêm mô đun vào hệ thống.

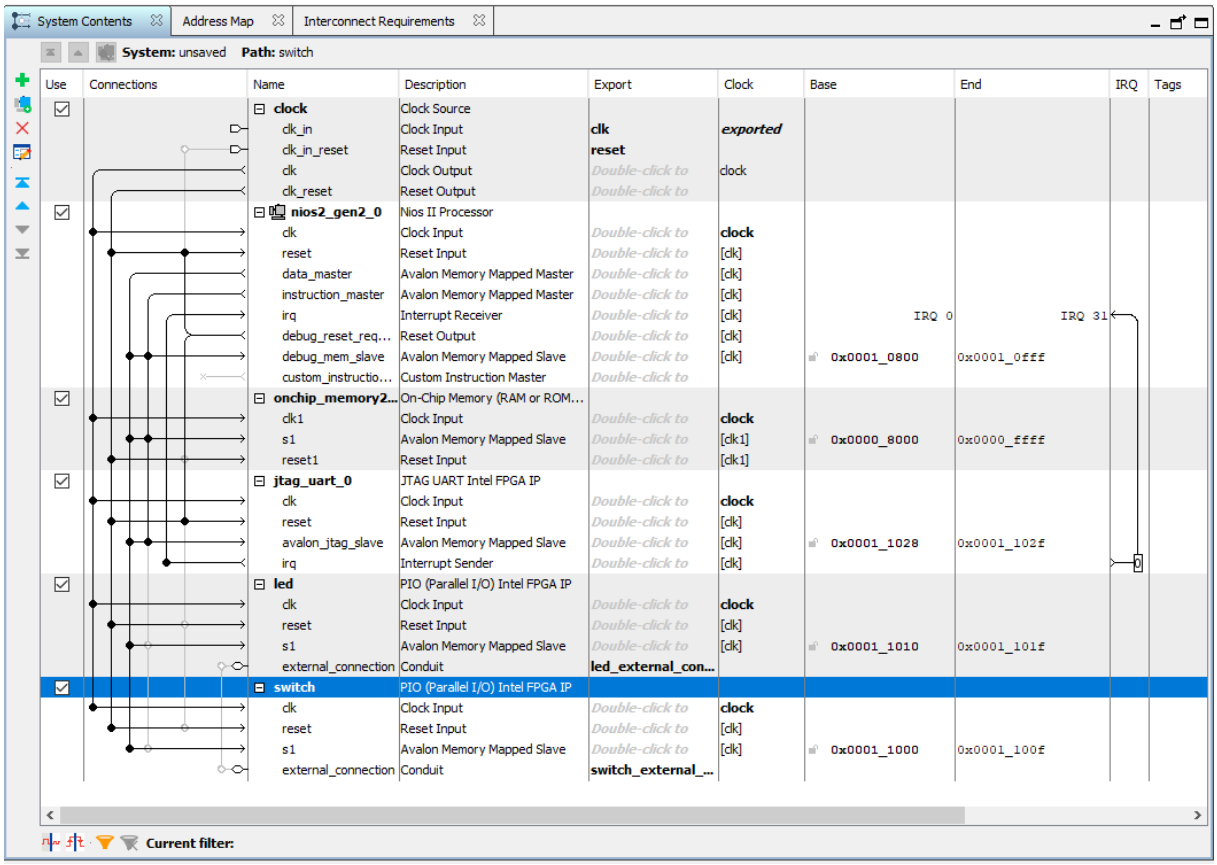


Hình 5. Cấu hình khối PIO-switch.



Hình 6. Kết quả toàn hệ thống.

- Tương tự bài 1, tiến hành gán địa chỉ cho các thành phần trong hệ thống: **System → Assign Base Addresses**. Kết quả được thể hiện trong hình 7.



Hình 7. Hệ thống với các địa chỉ Base.

- Ta để ý ở cột **Base,** đó là địa chỉ bắt đầu của các mô đun như đã đề cặp ở phần lý thuyết.
- Cuối cùng lưu lại và đặt tên hệ thống là **system** và generate hệ thống.

3.2.3. Tích hợp hệ thống.

- Tiến hành thêm file **system.qip** vào project.
- Tiến hành thêm file top-level là **Bai2.v** có nội dung như đoạn code bên dưới.

```
module Bai2
    (
    input               CLOCK_50,
    input       [0:0]   KEY,
    input       [15:0]  SW,
    output      [15:0]  LEDR
    );

    system nios_system(
            .clk_clk                                    (CLOCK_50),
            .reset_reset_n                              (KEY[0]),
            .switch_external_connection_export          ({16'd0,SW}),
            .led_external_connection_export             ({16'd0,LEDR}
)
            );
endmodule
```

- Tiến hành gán pin cho hệ thống bằng **Assignments → Import Assignments** với file pin **DE2_115_pin_assignments.cvs.**
- Tiến hành biên dịch và nạp phần cứng xuống board DE2-115.

3.3. Xây dựng phần mềm

- Thực hiện tạo project trên Eclipse tương tự bài 1, đặt tên là Bai2.
- Tạo file **source.c** có nội dung như đoạn code bên dưới, tiến hành build và nạp project phần mềm xuống board, thay đổi switch và quan sát kết quả.

```
#include <io.h>
#include <system.h>

void main() {
	int temp;
	while (1) {
		temp = IORD(SWITCH_BASE, 0);
		IOWR(LED_BASE, 0, temp);
	}
}
```

- Trong đó, **SWITCH_BASE** và **LED_BASE** là địa chỉ base của 2 mô đun **led** và **switch** trong hệ thống phần cứng như được thể hiện trong hình 7. Theo đó, **LED_BASE** là 0x11010 và SWITCH_BASE là 0x11000. Hình 8 và hình 9 thể hiện các địa chỉ được define trong file ***system.h.***

```
#define ALT_MODULE_CLASS_led altera_avalon_pio
#define LED_BASE 0x11010
#define LED_BIT_CLEARING_EDGE_REGISTER 0
#define LED_BIT_MODIFYING_OUTPUT_REGISTER 0
#define LED_CAPTURE 0
#define LED_DATA_WIDTH 32
#define LED_DO_TEST_BENCH_WIRING 0
#define LED_DRIVEN_SIM_VALUE 0
#define LED_EDGE_TYPE "NONE"
#define LED_FREQ 50000000
#define LED_HAS_IN 0
#define LED_HAS_OUT 1
#define LED_HAS_TRI 0
#define LED_IRQ -1
#define LED_IRQ_INTERRUPT_CONTROLLER_ID -1
#define LED_IRQ_TYPE "NONE"
#define LED_NAME "/dev/led"
#define LED_RESET_VALUE 0
#define LED_SPAN 16
#define LED_TYPE "altera_avalon_pio"
```

Hình 8. Địa chỉ của led.

```
#define ALT_MODULE_CLASS_switch altera_avalon_pio
#define SWITCH_BASE 0x11000
#define SWITCH_BIT_CLEARING_EDGE_REGISTER 0
#define SWITCH_BIT_MODIFYING_OUTPUT_REGISTER 0
#define SWITCH_CAPTURE 0
#define SWITCH_DATA_WIDTH 32
#define SWITCH_DO_TEST_BENCH_WIRING 0
#define SWITCH_DRIVEN_SIM_VALUE 0
#define SWITCH_EDGE_TYPE "NONE"
#define SWITCH_FREQ 50000000
#define SWITCH_HAS_IN 1
#define SWITCH_HAS_OUT 0
#define SWITCH_HAS_TRI 0
#define SWITCH_IRQ -1
#define SWITCH_IRQ_INTERRUPT_CONTROLLER_ID -1
#define SWITCH_IRQ_TYPE "NONE"
#define SWITCH_NAME "/dev/switch"
#define SWITCH_RESET_VALUE 0
#define SWITCH_SPAN 16
#define SWITCH_TYPE "altera_avalon_pio"
```

Hình 9. Địa chỉ của switch.

BÀI TẬP CHUẨN BỊ Ở NHÀ

Bài 1. Cho biết con trỏ là gì ? Cách truy xuất bộ nhớ với địa chỉ thông qua con trỏ được thực hiện như thế nào ? Ví dụ : ghi giá trị 0xabcd vào địa chỉ : 0x0800 sử dụng con trỏ ?

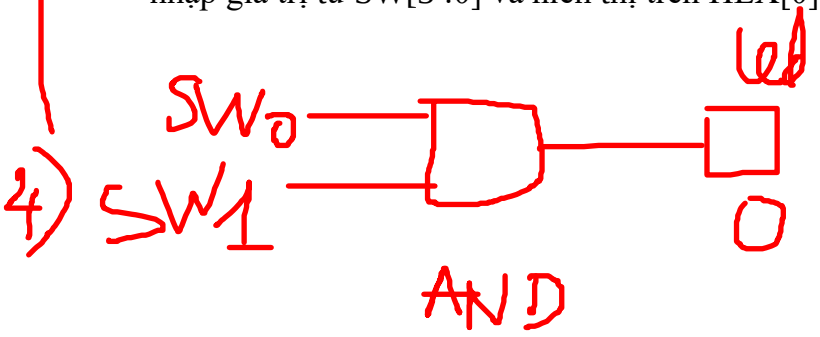
Bài 2. Giải thích các thanh ghi của bộ PIO của Intel ?

BÁO CÁO THỰC HÀNH

Bài 1. Giải thích cú pháp IORD và IOWR ?

Thay đổi code C trong phần 3.3 để truy xuất bộ led và switch thông qua con trỏ ?

Bài 2. Xây dựng lại phần cứng và phần mềm để tiến hành giải mã LED 7 đoạn được nhập giá trị từ SW[3 :0] và hiển thị trên HEX[0] bằng C code.



TÀI LIỆU THAM KHẢO

[1] DE2_115_User_Manual.

[2] Embedded Peripherals IP User Guide.

[3] Nios II Processor Reference.

[4] Avalon Interface Specification.