BÀI 3. GIAO TIẾP IO TỰ THIẾT KẾ

1. Mục đích

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

- Cách thêm mô đun tự thiết kế vào thư viện trong Platform Designer.
- Cách xây dựng và kết nối các giao tiếp IO (xây dựng bằng ngôn ngữ Verilog) vào hệ thống SoPC sẵn có.
- Cách truy xuất tới các giao tiếp IO từ công cụ Nios II.

2. Phần lý thuyết

2.1. Tạp project trên Quartus Prime

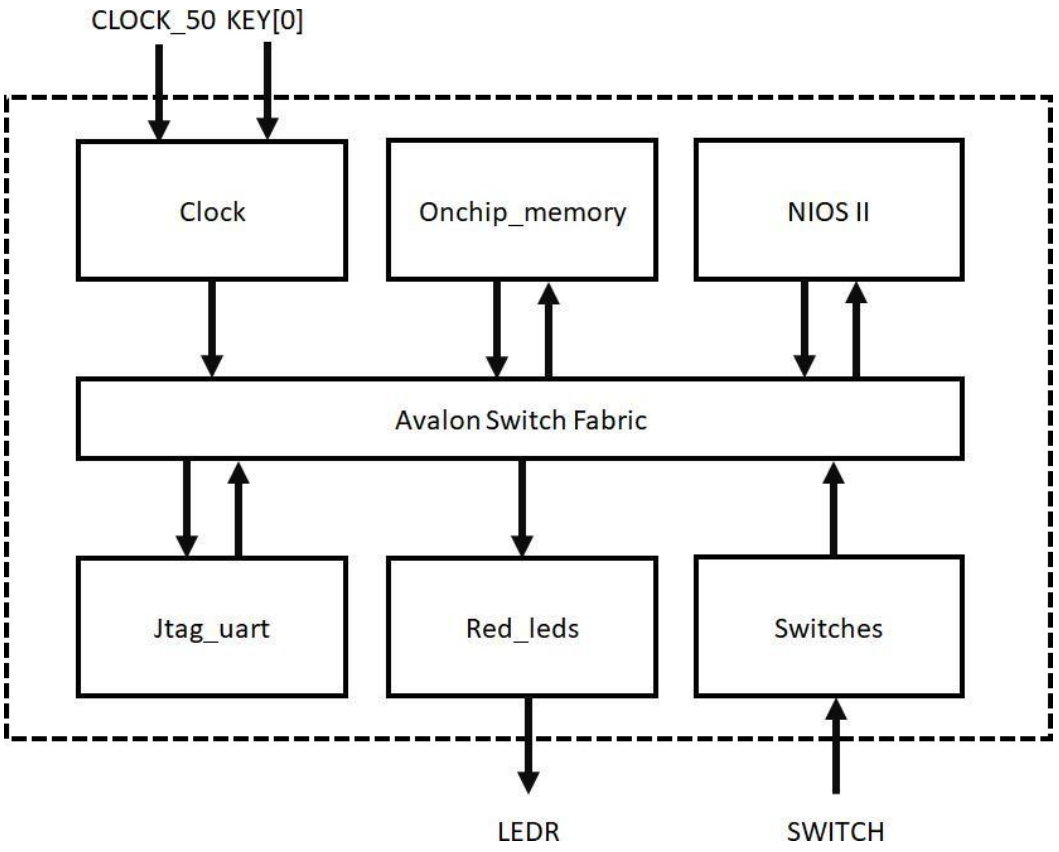
- Tạo project Quartus tên là "**Bai3**". Lưu ý đường dẫn thư mục project không được có khoảng trắng.
- Chọn Family là **Cyclone IV E**, device là **EP4CE115F29C7** nếu dùng board **DE2-115**.
- Chọn Family là **Cyclone V**, device là ***5CSXFC6D6F31C6*** nếu dùng board **DE10-Standard**.
- Trong project, tạo file **switches.v** và **red_leds.v** với nội dung được mô tả như 2 đoạn code bên dưới.

```
module switches
(
    input iClk,
    input iReset_n,
    input  iChip_select_n,
    input iRead_n,
    input [31:0] iSwitches_data,
    output reg [31:0] oSwitches_reg
);
    always@(posedge iClk, negedge iReset_n)
    begin
        if(~iReset_n)
        begin
            oSwitches_reg <= 32'd0;
        end
    else
    begin
        if(~iChip_select_n & ~iRead_n)
        begin
            oSwitches_reg <= iSwitches_data;
        end
    end
end endmodule
```

```
module red_leds
(
    input iClk,
    input  iReset_n,
    input  iChip_select_n,
    input iWrite_n,
    input [31:0] iRed_leds_data,
    output reg [31:0] oRed_leds
);
    always@(posedge iClk, negedge iReset_n)
    begin
        if(~iReset_n)
        begin
            oRed_leds <= 32'd0;
        end
        else
        begin
            if(~iChip_select_n & ~iWrite_n)
            begin
                oRed_leds <= iRed_leds_data;
            end
        end
    end
endmodule
```

2.2. Tổng quan hệ thống phần cứng

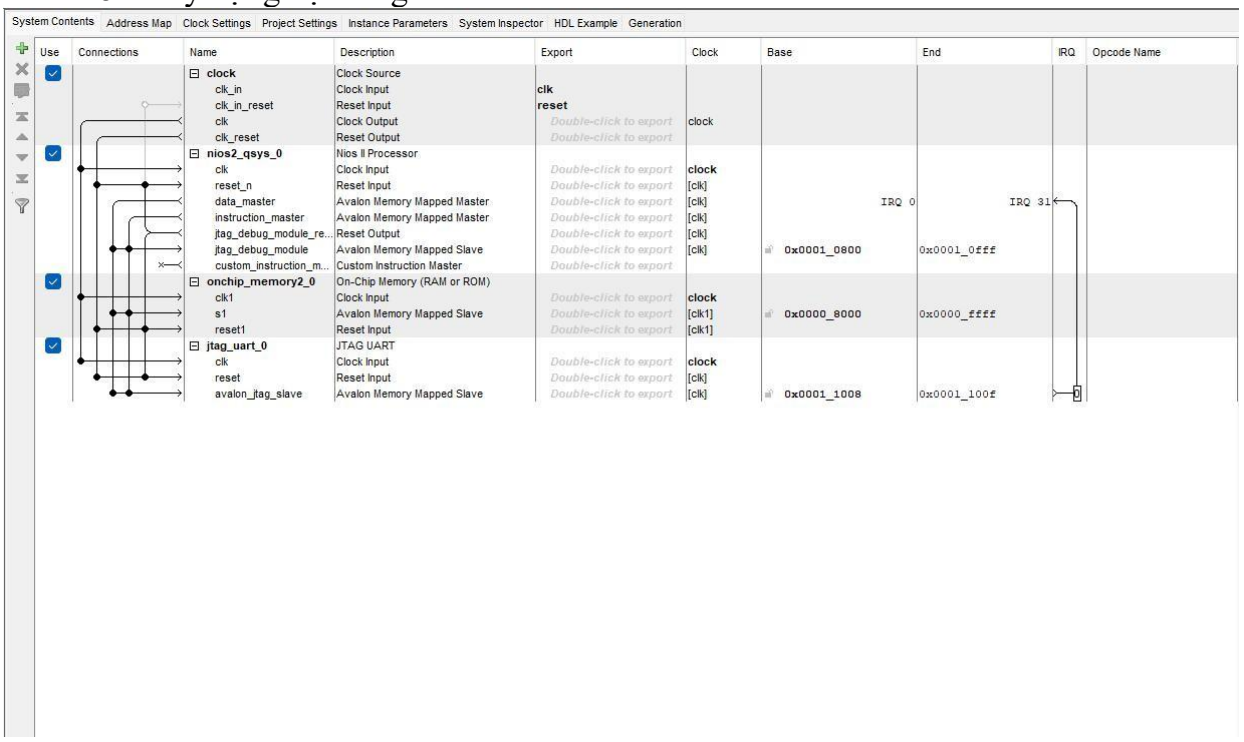
Hệ thống phần cứng được thể hiện trong hình 1 bên dưới. Trong đó mô đun Red_leds và switches là 2 mô đun giao tiếp IO tự thiết kế bằng code verilog như 2 đoạn code ở trên.



Hình 1. Tổng quan hệ thống.

2.3. Tạo hệ thống trên Qsys

- Xây dựng hệ thống như hình 2 bên dưới.

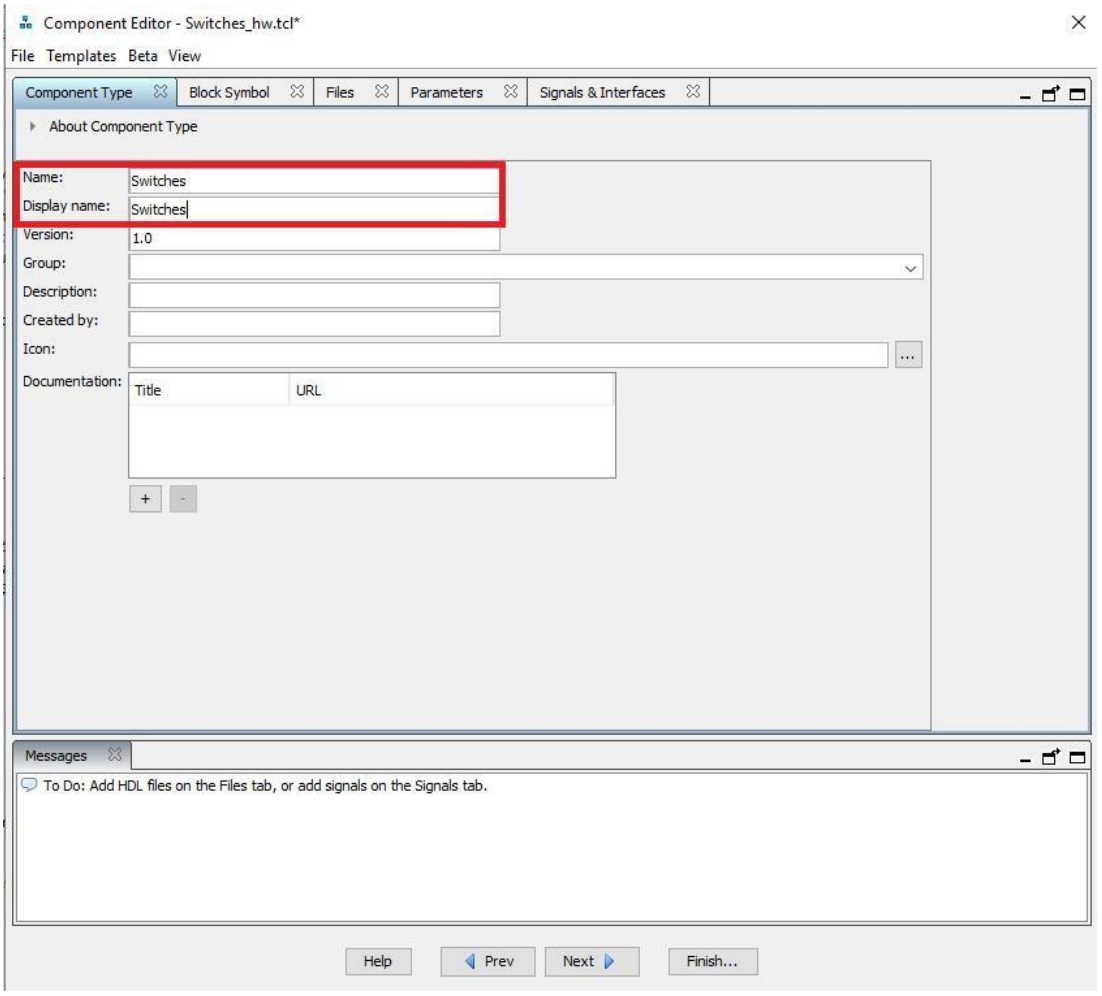


Hình 2. Hệ thống chuẩn bị ban đầu.

- Tiếp theo, tiến hành thêm mô đun switches đã được định nghĩa ở file switches.v. Tại tab thư viện – IP Catalog, ta nhấp đúp và ô New Component như hình 3.

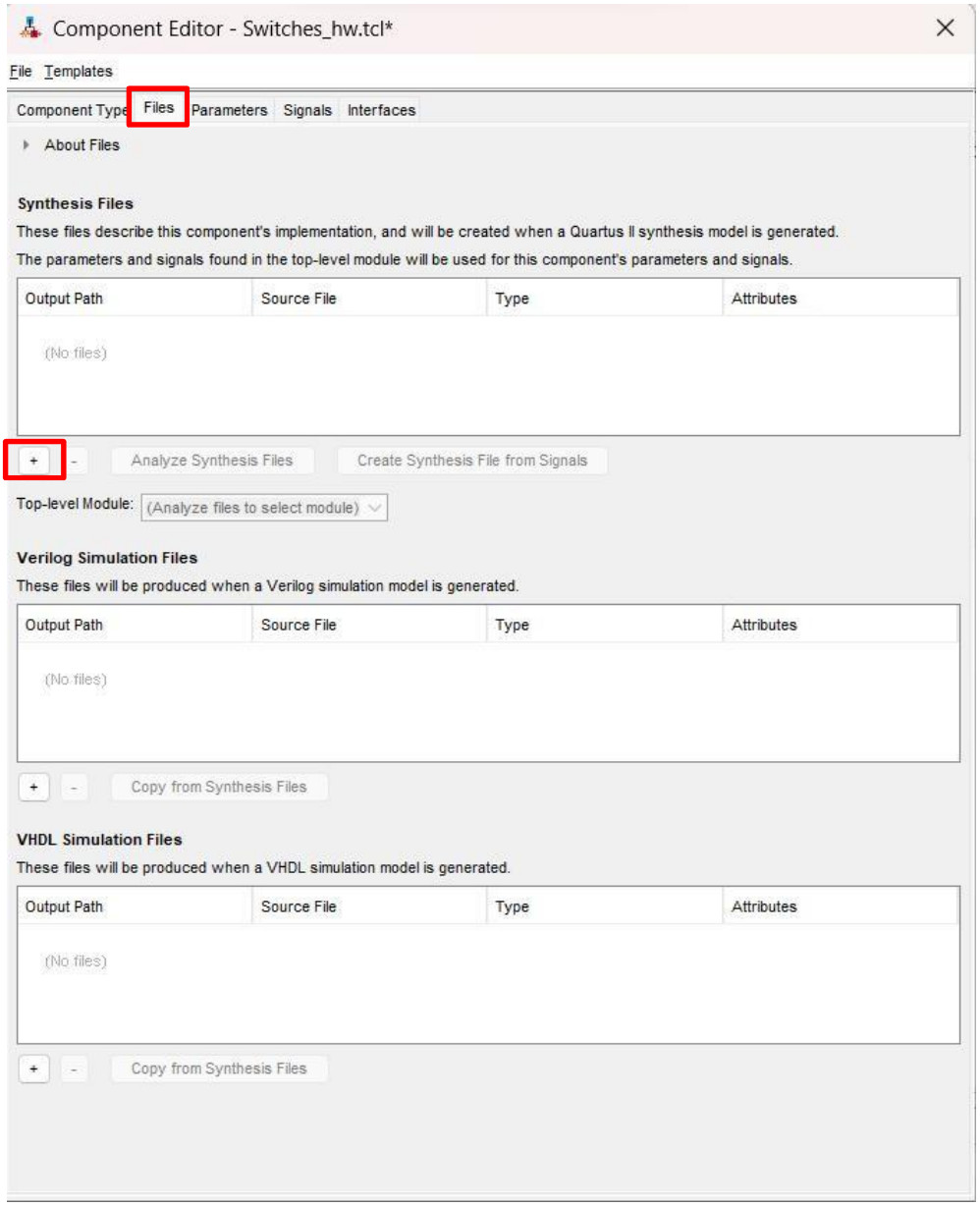
Hình 3. Tab Component Library.

- Sau đó, cửa sổ **Component Editor** sẽ hiện ra, tiếp tục đặt tên là **Switches** và Display name là **Switches** như hình 4.



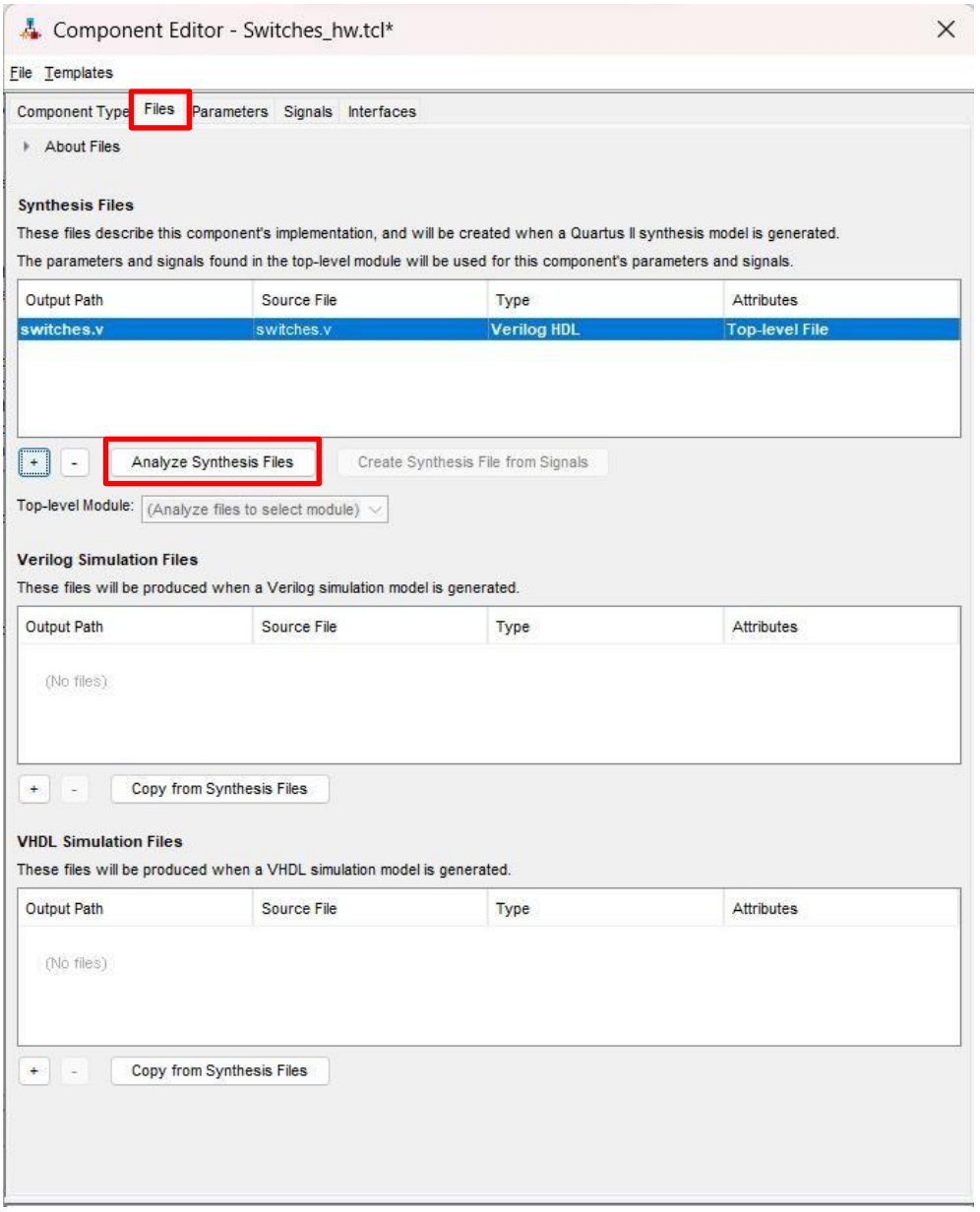
Hình 4. Tab Component Type.

- Tiếp theo, chuyển sang tab Files và chọn Add File như hình 5.



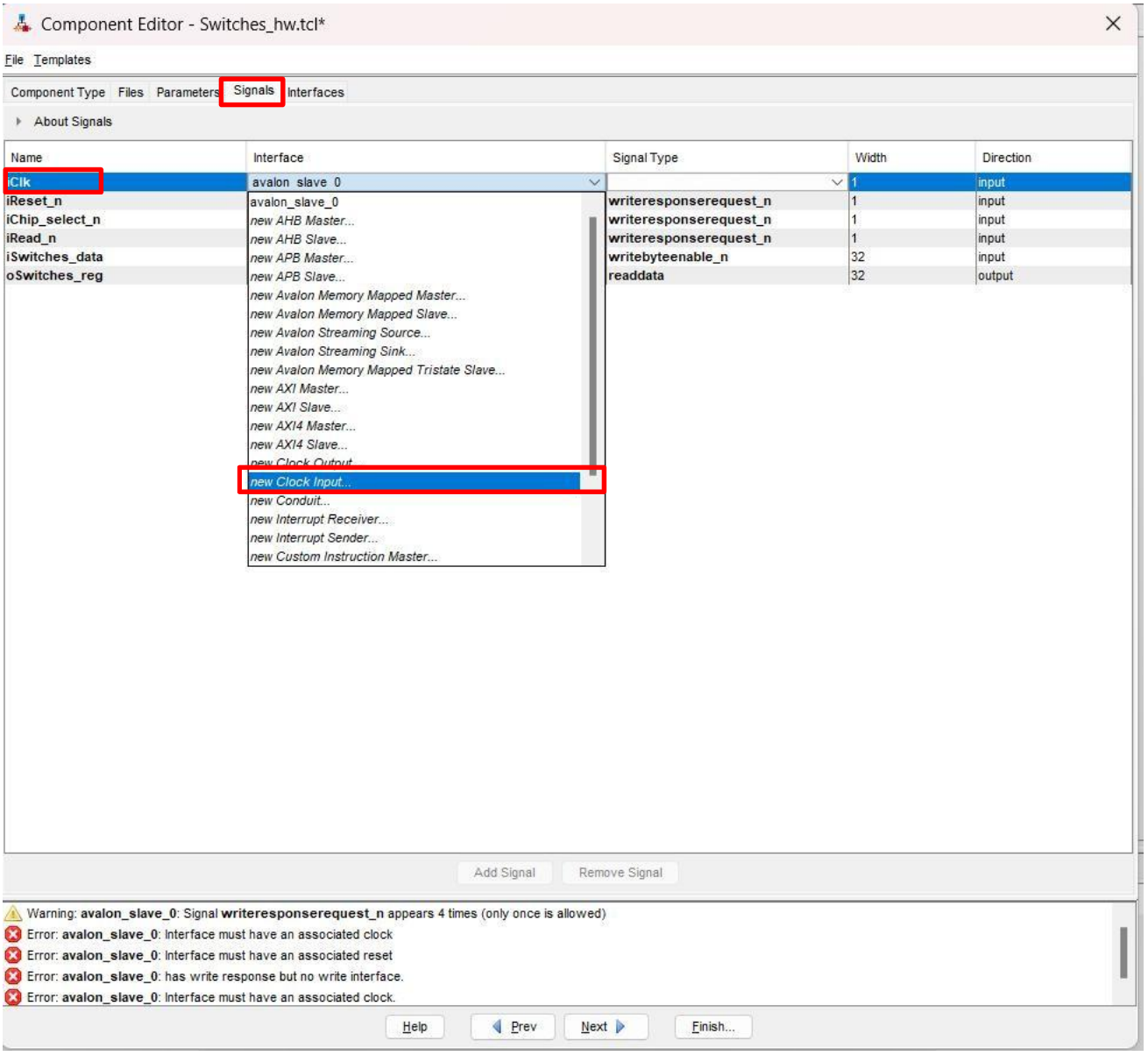
Hình 5. Tab Files.

- Sau khi thêm file hoàn tất chọn tiếp Analyze Synthesis Files như hình 6.



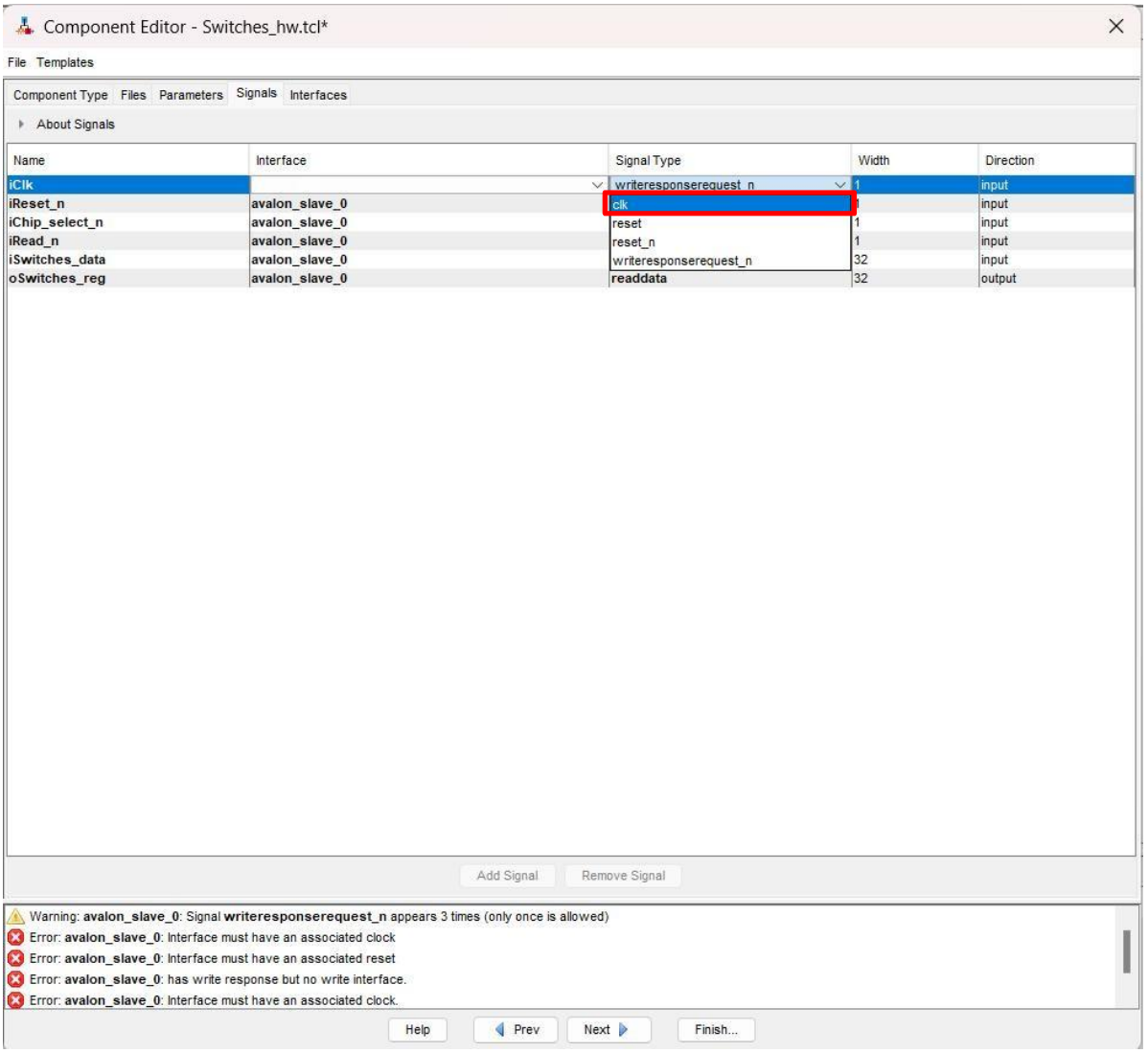
Hình 6. Thêm file và phân tích file đã thêm.

- Sau đó, chọn **Signals** để hiển thị tab **Signals**. Tiến hành cấu hình cho các tín hiệu, đầu tiên sẽ cấu hình cho tín hiệu **iClk**. Cấu hình cho tín hiệu **iClk** với interface là new Clock Input… , hình 7.



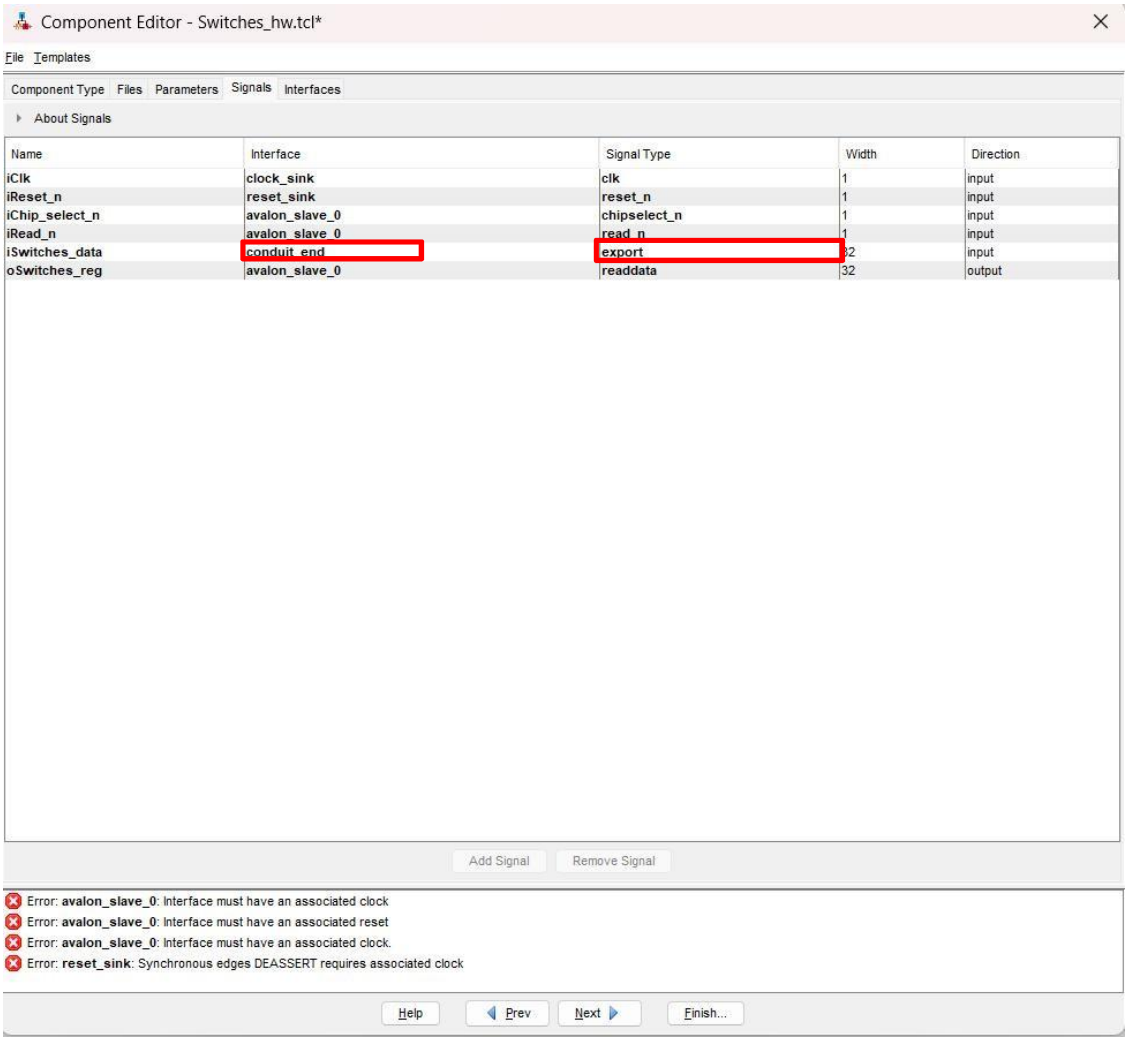
Hình 7. Tab signals.

- Tiếp tục với loại tín hiệu ở **Signal Type**, chọn là **clk**, hình 8.



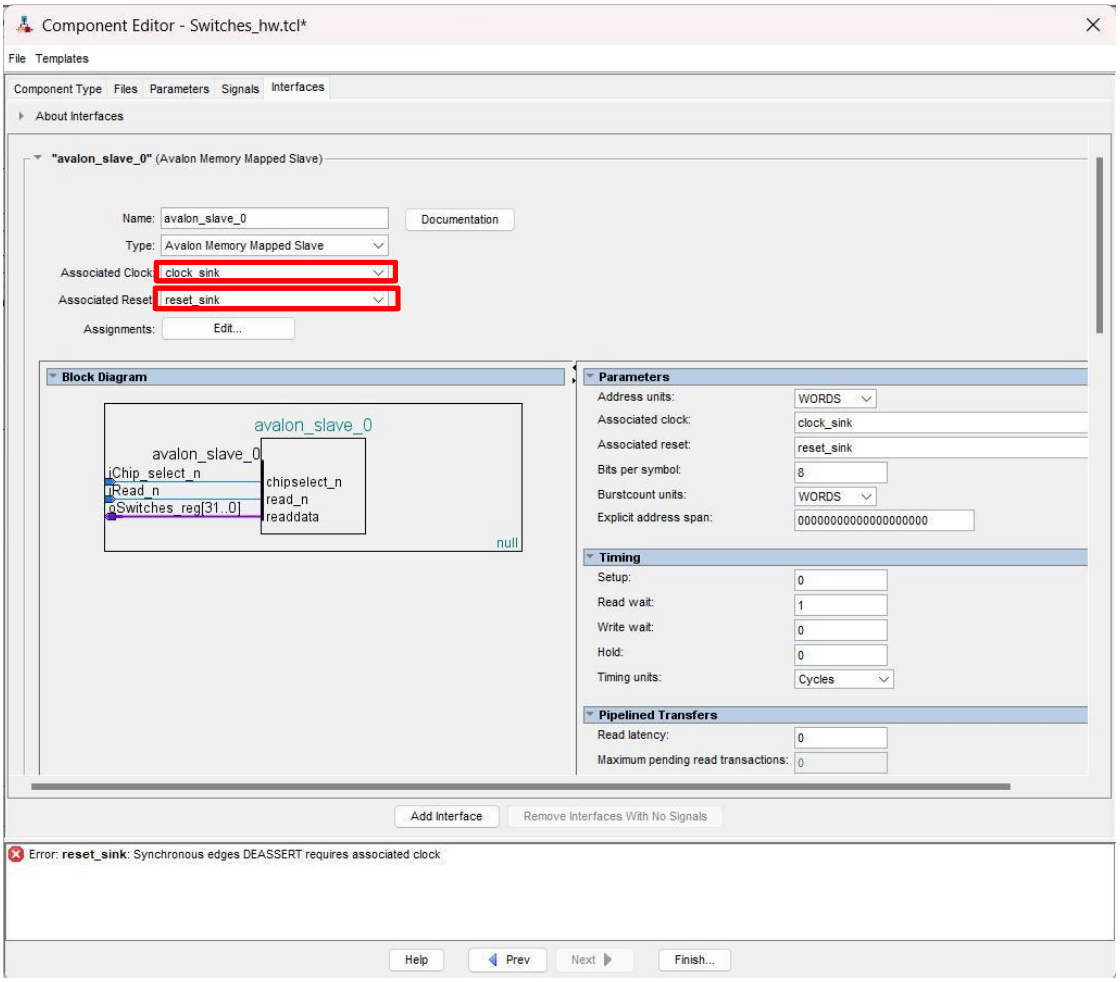
Hình 8. Cấu hình tín hiệu cho tín hiệu iClk.

- Tương tự, tiến hành cấu hình cho các tín hiệu còn lại như hình 9. Lưu ý, tín hiệu **iSwitches_data** là tín hiệu được thiết lập Interface là **conduit_end** và Signal Type gõ là **export**.



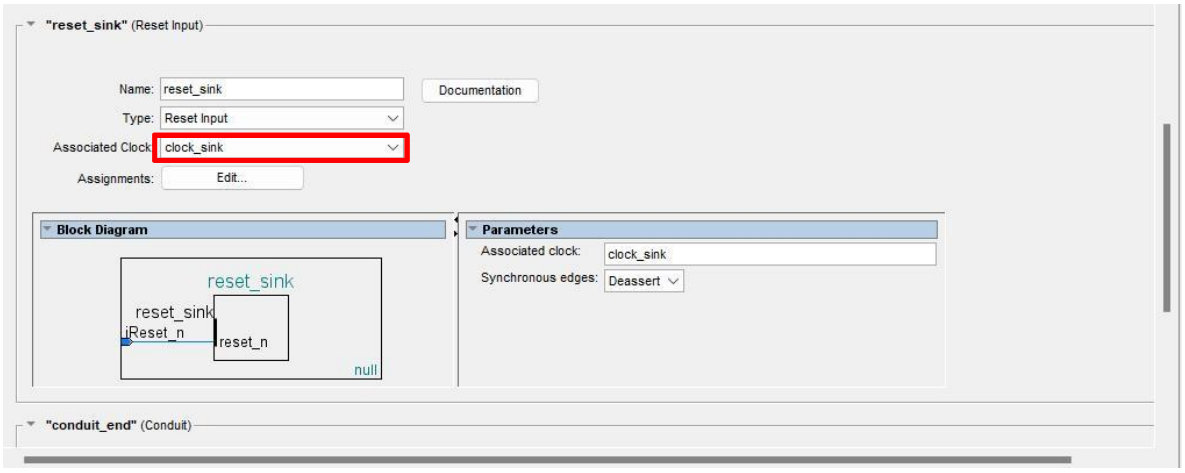
Hình 9. Cấu hình tín hiệu iSwitches_data.

- Cuối cùng chuyển sang tab **Signals & Interfaces**, tiến hành cấu hình tín hiệu Clock và Reset cho các interface. Hình 10 đại diện cho việc cấu hình tín hiệu **Clock** và **Reset** cho interface **avalon_slave_0**.

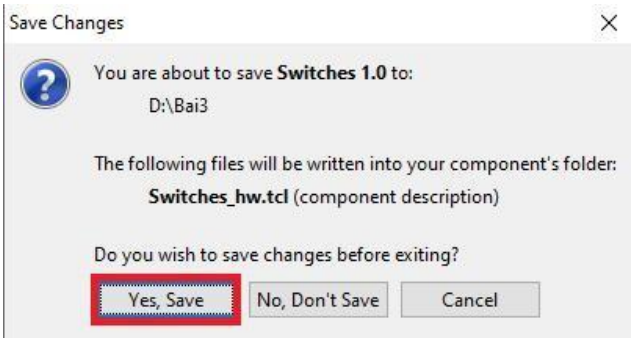


Hình 10. Tab Signals & Interfaces.

- Kéo xuống dưới tiến hành cấu hình cho reset_sink

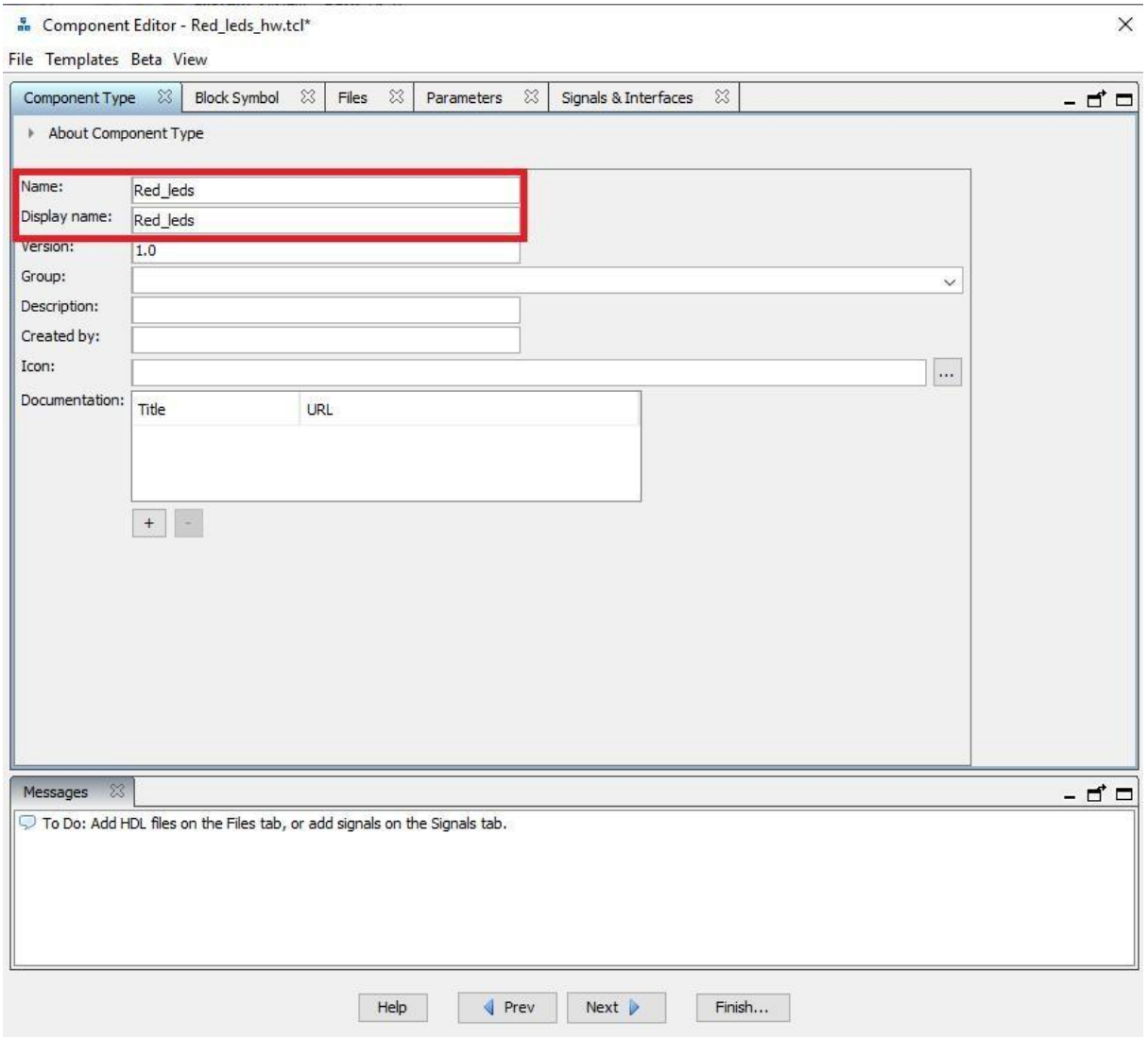


- Khi không còn lỗi, nhấn **Finish** và lưu lại để kết thúc.

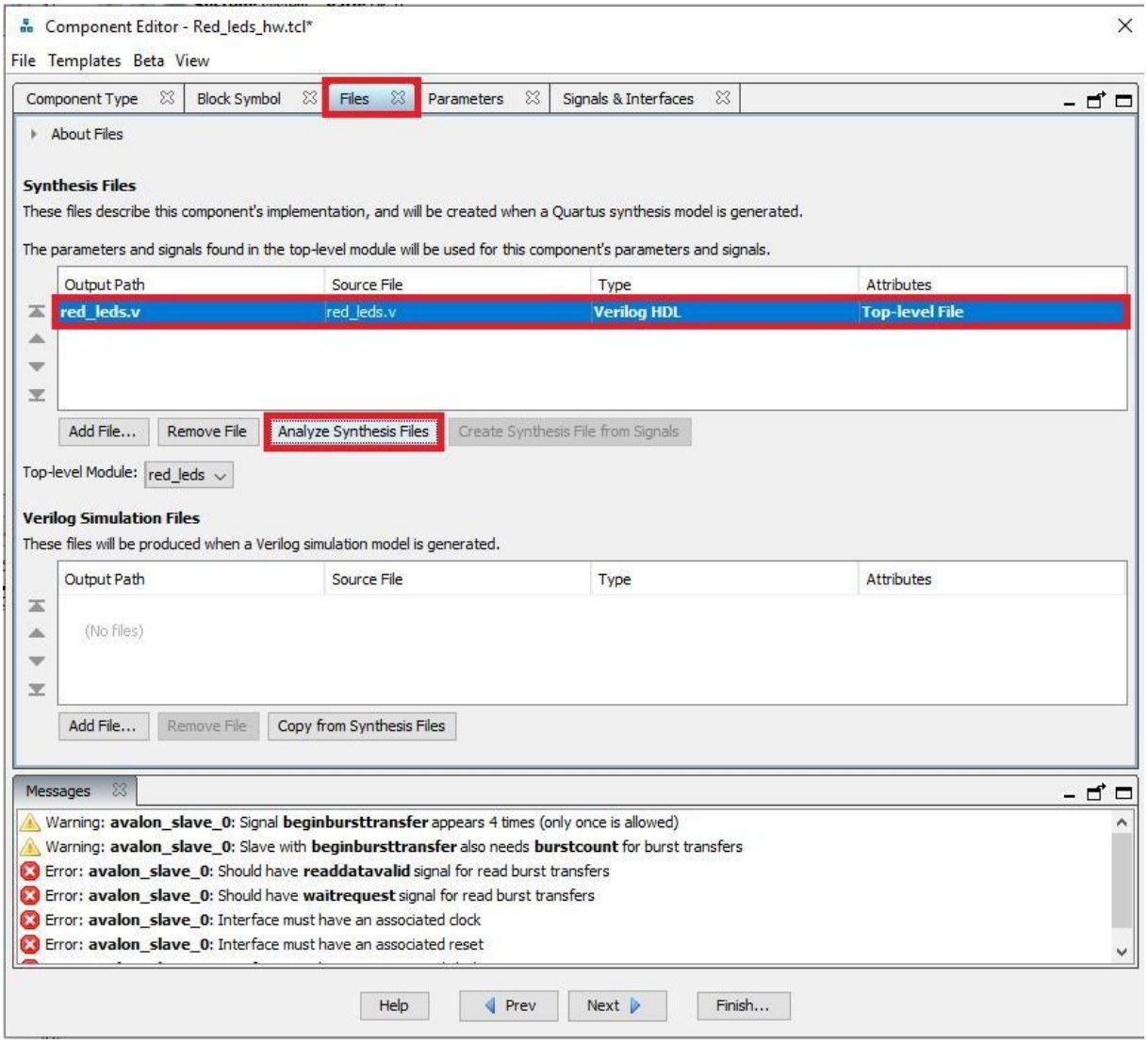


Hình 11. Lưu lại mô đun vừa thêm.

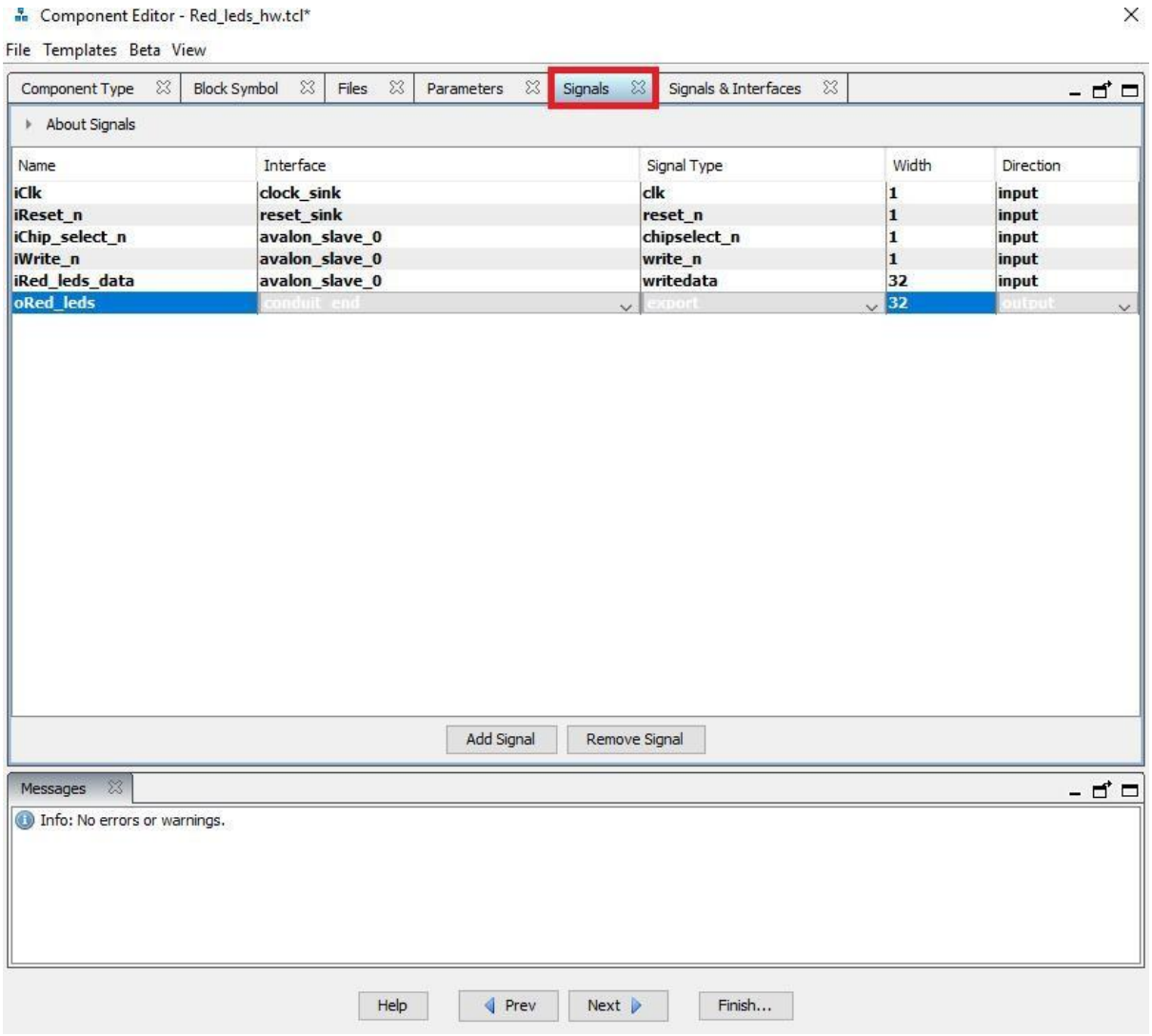
- Tương tự, ta thêm mô đun red_leds đã được định nghĩa ở file **red_leds.v**. Các bước thực hiện được thể hiện ở các hình bên dưới bên dưới.



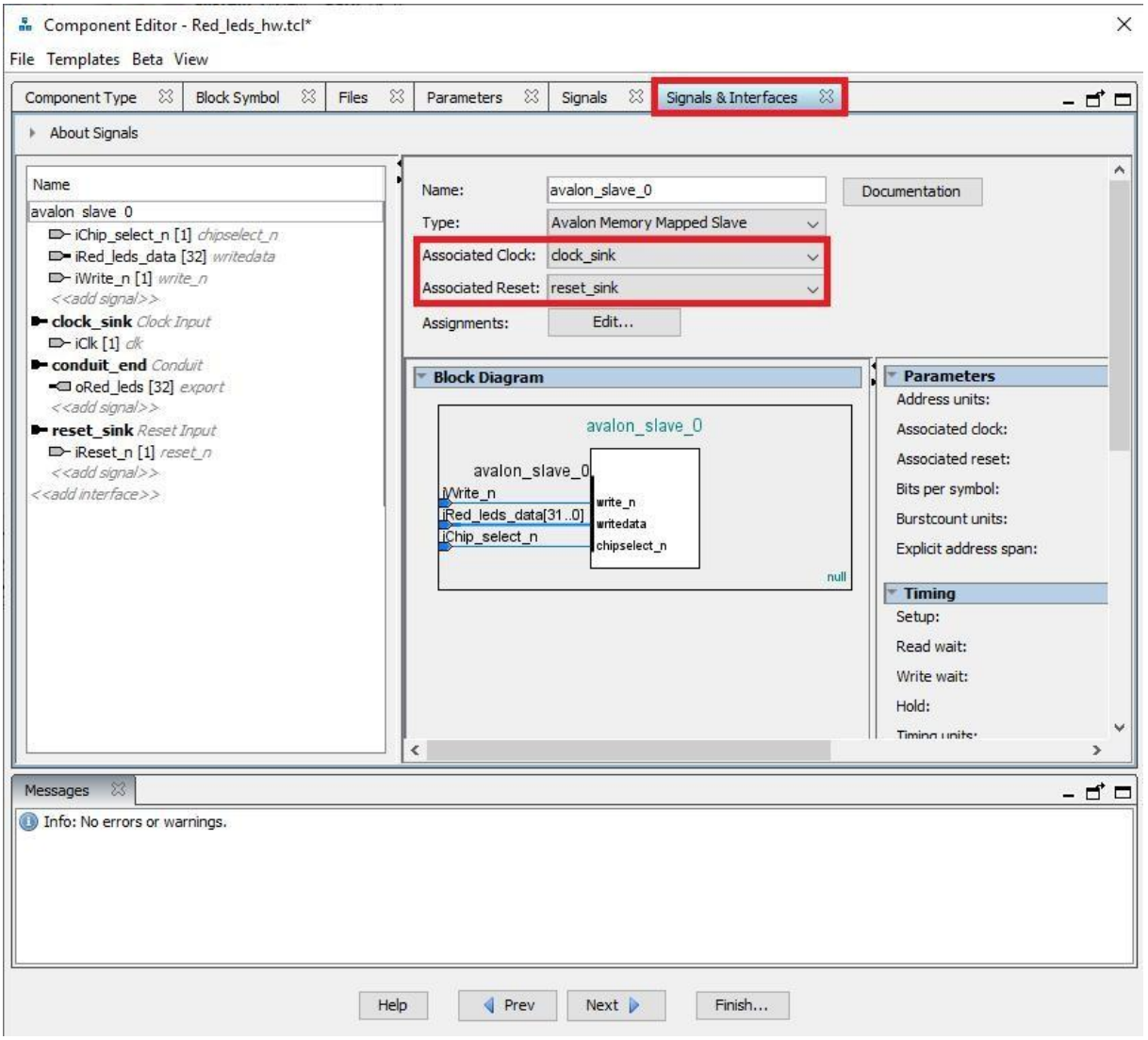
Hình 12. Tab Component Type.



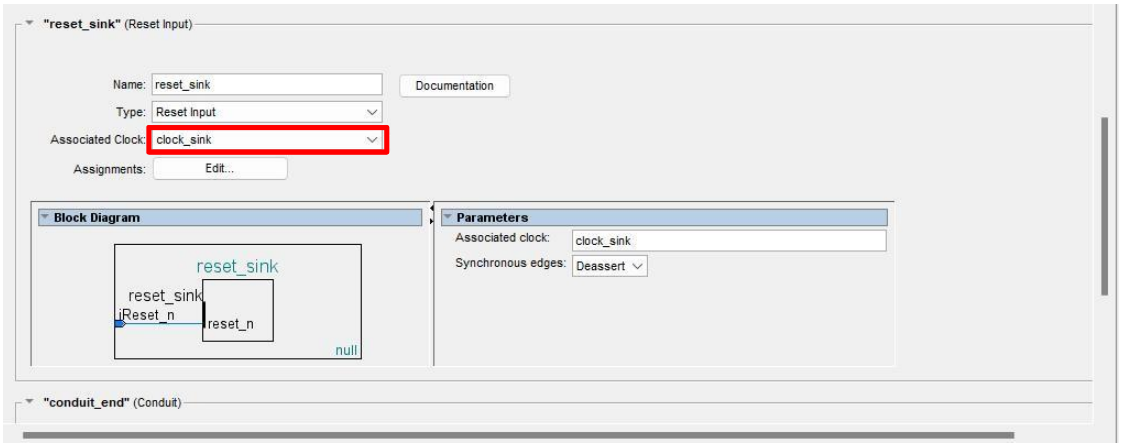
Hình 13. Thêm file và phân tích ở tab Files.



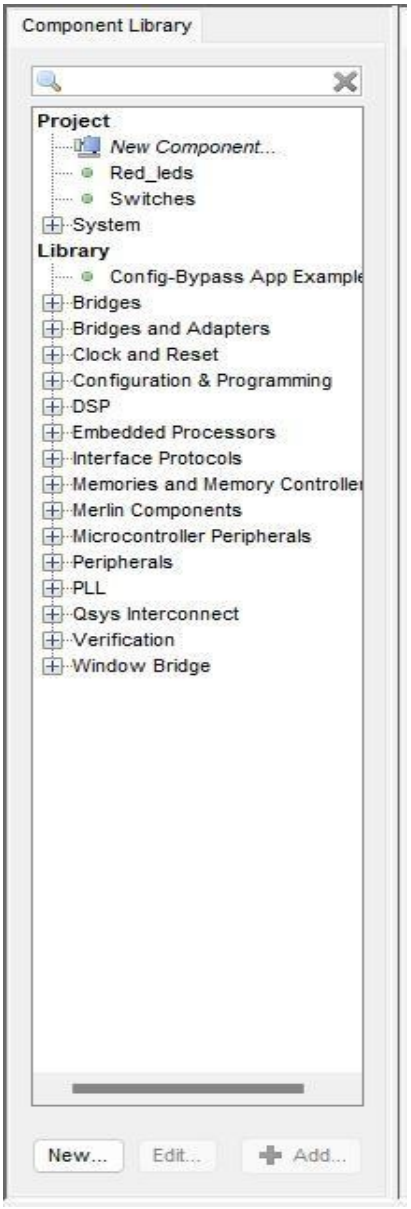
Hình 14. Cấu hình ở tab Signals.



Hình 15. Cấu hình tín hiệu Clock và Reset ở tab Signals & Interfaces.

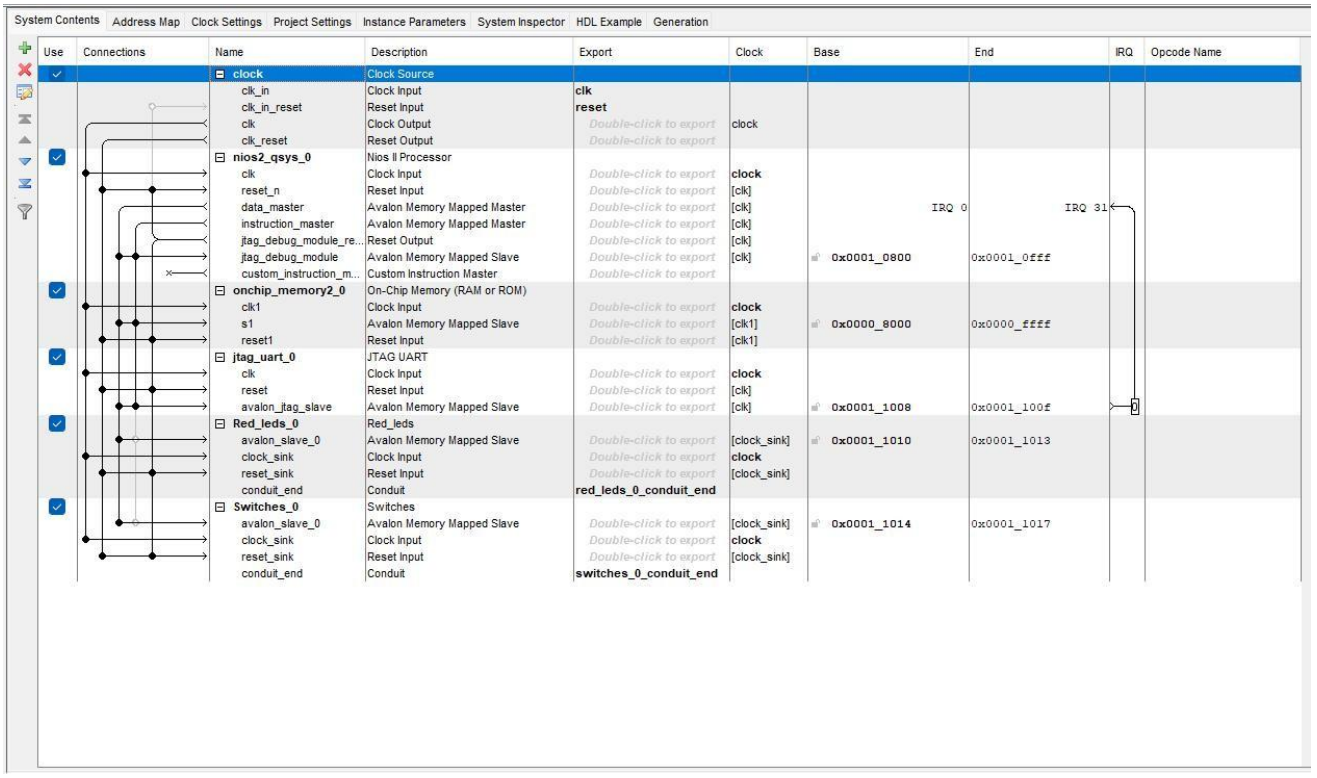


- Sau khi thêm 2 mô đun hoàn tất, có kết quả ở tab thư viện như hình 16.



Hình 16. Hai mô đun vừa thêm xuất hiện ở tab IP Catalog.

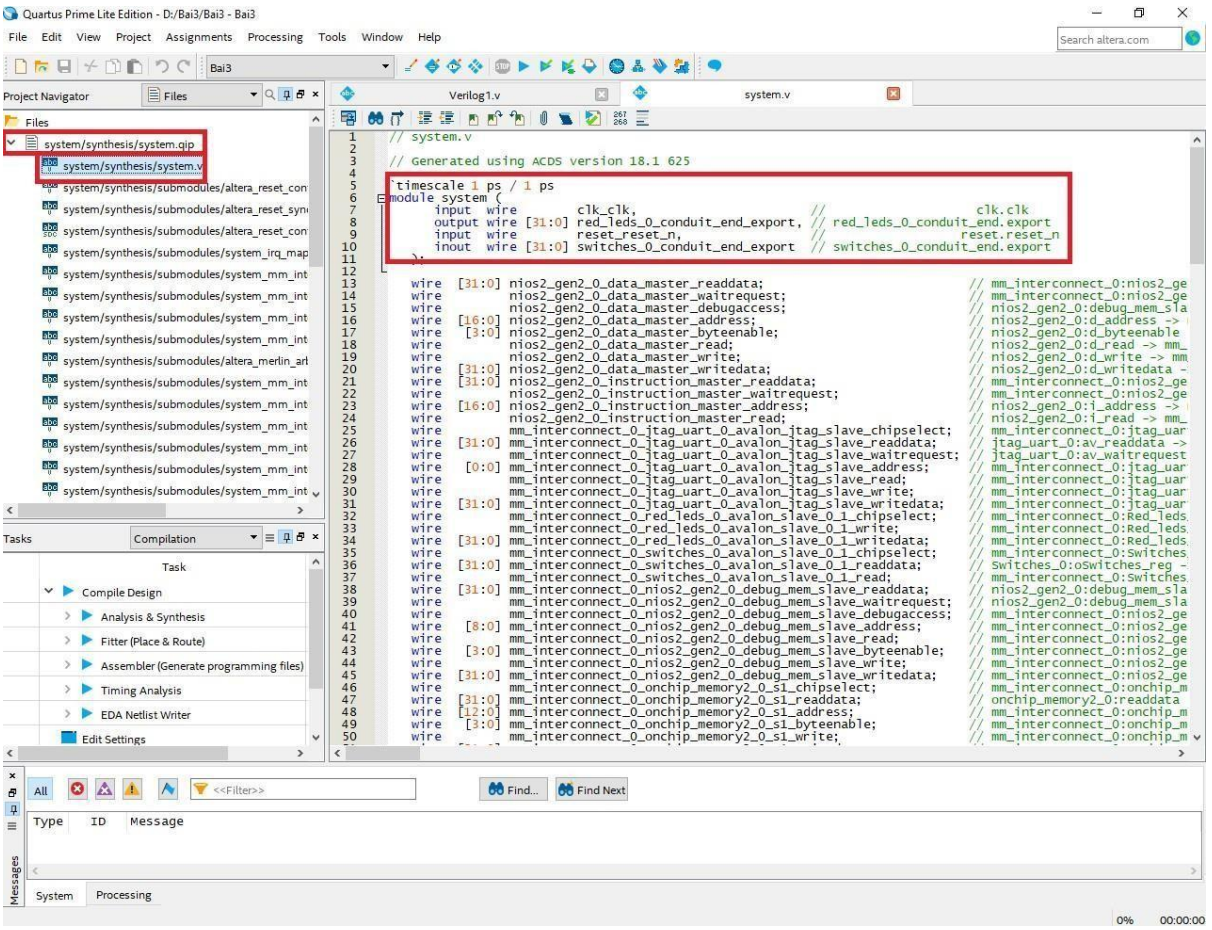
- Cuối cùng, tiến hành kết nối hệ thống như hình 17, gán địa chỉ, lưu lại và generate hệ thống với tên là system.



Hình 17. Hệ thống hoàn chỉnh.

2.4. Tích hợp hệ thống.

- Thêm file **system.qip** vào project. Mở file **system.v** để xem các tín hiệu của mô đun **system**, hình 20.



Hình 18. Thêm file system.qip và tín hiệu của mô đun system.

- Tạo file top-level là **Bai3.v** như đoạn code bên dưới. (chú ý đúng với tín hiệu file **system.v**, nếu có sự khác nhau, sinh viên sửa các tín hiệu trong file **Bai3.v** tương ứng với file **system.v**).

```
module
    Bai3(inp                CLOCK_50,
    ut input        [0:0]   KEY,
    input           [15:0]  SW,
    output          [15:0]  LEDR
);
system  Nios_system(
            .clk_clk                            (CLOCK_50),
            .reset_reset_n                      (KEY[0]),
            .red_leds_0_conduit_end_export      (LEDR),
            .switches_0_conduit_end_export      (SW)
            );
```

```
endmodule
```

- Gán chân pin cho board: **Assignements → Import Assignements**.
- Biên dịch phần cứng và nạp xuống board.

2.5. Lập trình phần mềm

- Tạo và đặt tên project là "**Bai3**".
- Thêm file "**source.c**" vào project "**Bai3". File "source.c**" có nội dung như đoạn code bên dưới. Build project và dowload phần mềm xuống board. Đổi trạng thái của Switch trên board và quan sát LED.

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

void main(){
	short temp;
	while (1){
		temp = IORD(SWITCHES_0_BASE, 0);
		IOWR(RED_LEDS_0_BASE, 0 , temp);
	}
}
```

BÀI TẬP CHUẨN BỊ Ở NHÀ

Bài 1. Dựa vào tài liệu tham khảo [4], vẽ dạng sóng thể hiện quá trình hoạt động của 2 file switches.v và red_leds.v

Bài 2. Viết code verilog mô tả mô đun thực hiện giải mã ra led 7 đoạn tương ứng với dữ liệu từ SW[3:0].

BÁO CÁO THỰC HÀNH

Bài 1. Đổi code ở file source.c để truy xuất 2 mô đun Switches và Red_leds từ Nios II thông qua con trỏ.

Bài 2. Thực hiện hệ thống SoC có nhiệm vụ giải mã led 7 đoạn tương ứng với dữ liệu từ SW[3:0] như code đã chuẩn bị ở bài chuẩn bị ở nhà.

TÀI LIỆU THAM KHẢO

[1] DE2_115_User_Manual.

[2] Embedded Peripherals IP User Guide.

[3] Nios II Processor Reference.

[4] Avalon Interface Specification.

BÀI 4. THIẾT KẾ MÔ ĐUN CHỨC NĂNG VÀ KHẢO SÁT BẰNG SIGNAL TAB

1. Mục tiêu.

Thông qua bài thực hành này, sinh viên sẽ:

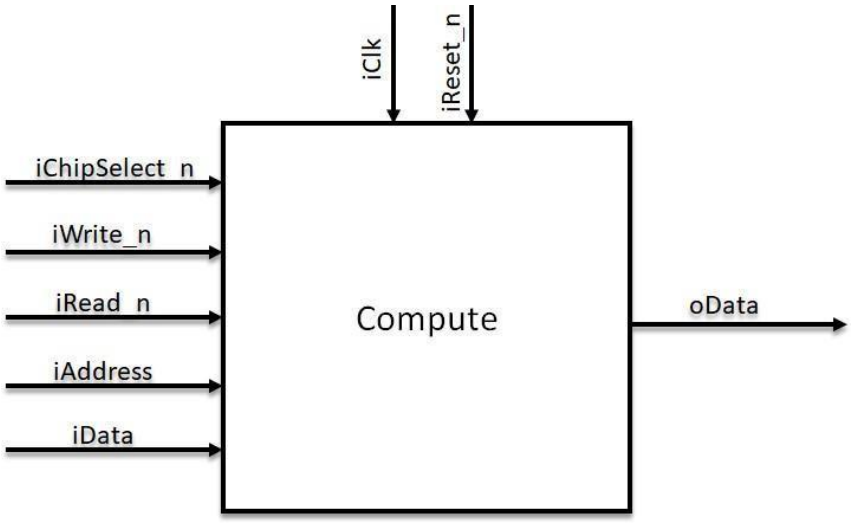
- Có khả năng thiết kế mô đun verilog đơn giản theo yêu cầu.
- Truy xuất mô đun vừa thiết kế thông qua Nios II.
- Biết cách phân tích tín hiệu logic và debug dùng công cụ Signal Tab.

2. Phần lý thuyết.

2.1. Tổng quan mô đun thiết kế

Tiến hành thiết kế mô đun Compute thực hiện tính toán như sau: = ( ) = + với a, b, x là các số nguyên dương.

Các tín hiệu của mô đun được mô tả như hình 1 và bảng 1 bên dưới.



Hình 1. Tín hiệu vào ra của mô đun Compute.

Bang 1. Bang mô ta tín hiệu của mô đun Compute.

STT	Tên tín hiệu	Độ rộng (bits)	Hướng	Mô tả
1	iClk	1	Input	Cấp xung clock cho mô đun hoạt động.
2	iReset_n	1	Input	Cấp tín hiệu reset mức thấp cho mô đun.

3	iChipSelect_n	1	Input	Nếu iChipSelect_n = 0, thì mô đun được phép hoạt động. Ngược lại, iChipSelect_n = 1, mô đun không được hoạt động.
4	iWrite_n	1	Input	Nếu iWrite_n = 0, mô được cho phép dữ liệu vào. Ngược lại, oWrite_n = 1, mô đun bỏ qua tín hiệu đưa vào.
5	iRead_n	1	Input	Nếu iRead_n = 0, mô đun cho phép đọc dữ liệu ra ngoài. Ngược lại, iRead_n = 0, mô đun không cho phép đọc dữ liệu ra ngoài.
6	iAddress	2	Input	Địa chỉ cần cho việc đọc hoặc ghi.
7	iData	32	Input	Dữ liệu ghi vào.
8	oData	32	Output	Dữ liệu đọc ra ngoài.

Các thanh ghi trong mô đun Compute:

Bang 2. Bang mô ta các thanh ghi của mô đun Compute.

Offset	Tên thanh ghi	Đọc/Ghi	Mô tả	
			31 - 4	3 - 0
0	a	Đọc/Ghi	Không sử dụng	Giá trị a
1	b	Đọc/Ghi	Không sự dụng	Giá trị b
2	x	Đọc/Ghi	Không sử dụng	Giá trị
3	y	Đọc	Không sử dụng	Kết quả y = ax + b

2.2. Code verilog mô tả mô đun Compute

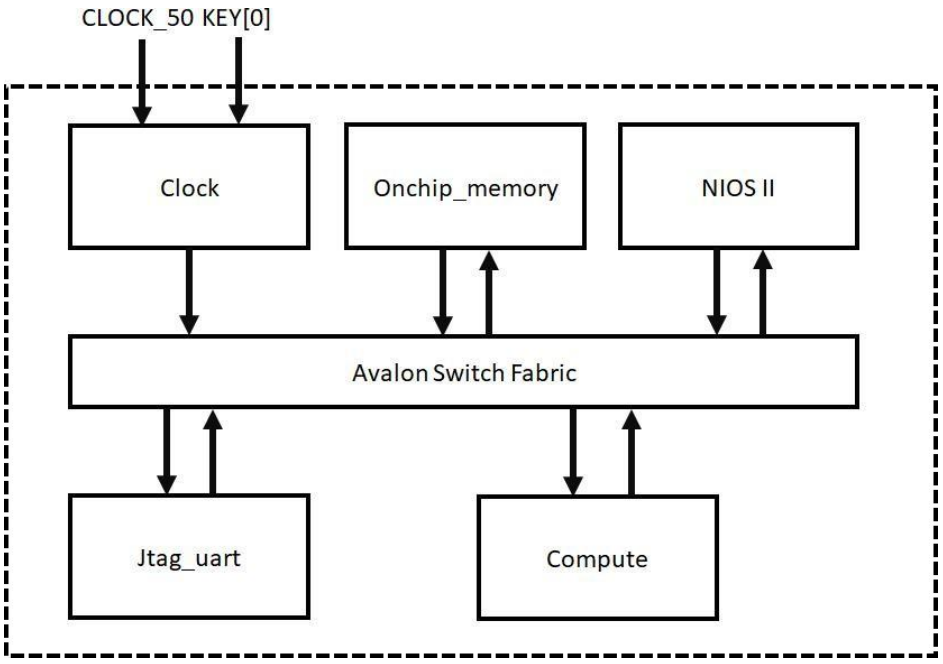
```
module
    Compute( input
    iClk, input
    iReset_n,
    input  iChipSelect_n,
    input iWrite_n,
    input iRead_n,
    input [1:0] iAddress,
    input [31:0] iData,
    output reg [31:0] oData
);
reg [31:0]a , b, x;

    always@(posedge iClk, negedge iReset_n)
    begin
        if(~iReset_n)
        begin
            oData <= 32'd0;
            a <= 32'd0;
            b <= 32'd0;
            x <= 32'd0;
        end
        else
```

```
        begin
        if( ~iChipSelect_n & ~iWrite_n)
        begin
        case (iAddress)
            2'd0: a[3:0] <= iData[3:0];
            2'd1: b[3:0] <= iData[3:0];
            2'd2: x[3:0] <= iData[3:0];
        endcase
        end
        if (~iChipSelect_n & ~iRead_n)
        begin
        case (iAddress)
            2'd0: oData <= a;
            2'd1: oData <= b;
            2'd2: oData <= x;
            2'd3: oData <= a*x + b;
        endcase
        end
        end
        end
        endmodule
```

3. Phần thực hành

3.1. Tổng quan hệ thống

- Hệ thống được thiết kế như hình 2 bên dưới bên dưới:



Hình 2. Tổng quan hệ thống.

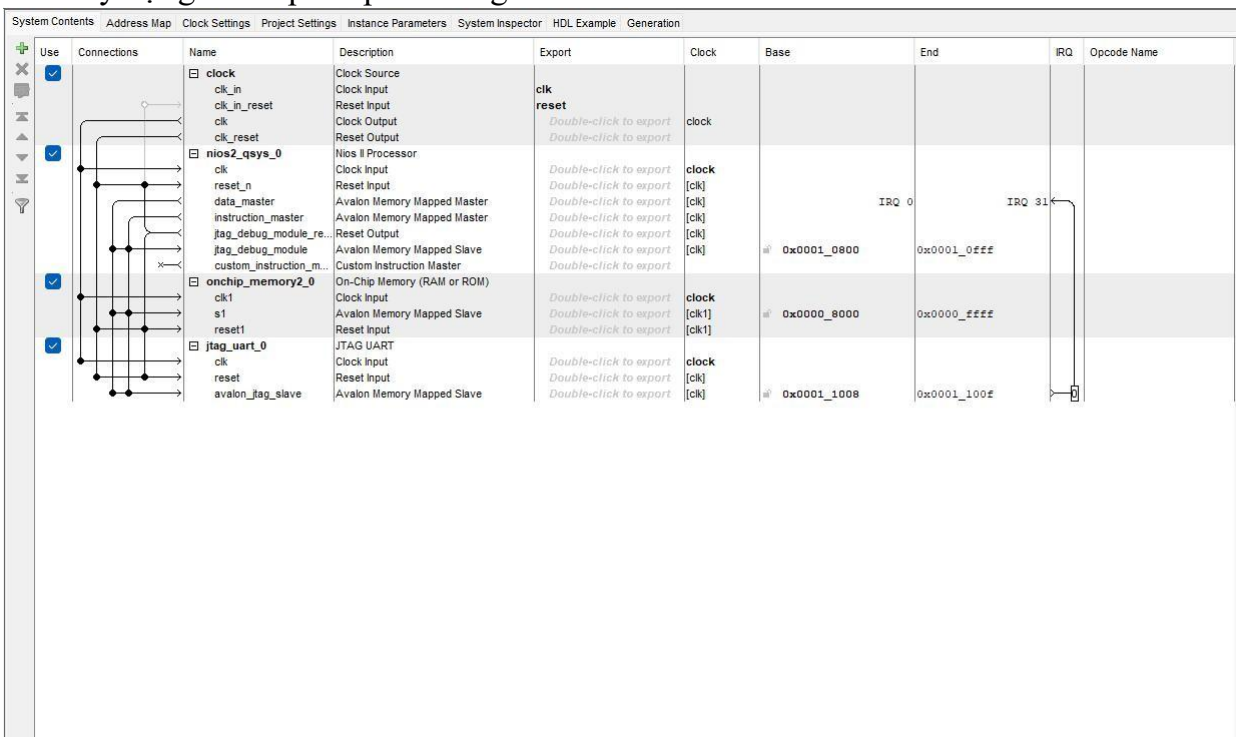
3.2. Tạo project trên Quartus Prime

- Tạo project Quartus tên là "**Bai4**". Lưu ý đường dẫn thư mục project không được có khoảng trắng.

- Chọn Family là **Cyclone II**, device là **EP2C35F672C6** nếu dùng board **DE2**

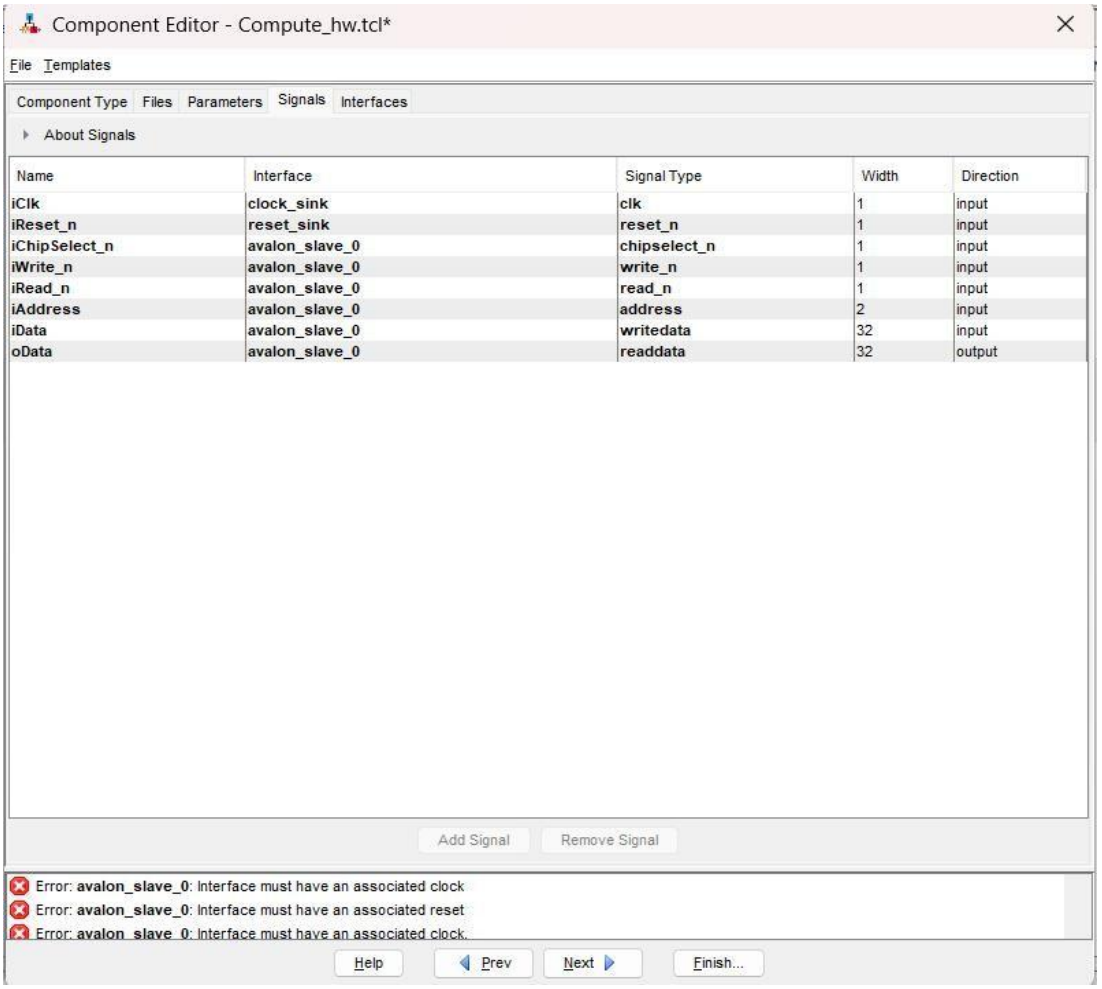
3.3. Xây dựng phần cứng trên Platform Designer

- Xây dựng thành phần phần cứng như hình 3.



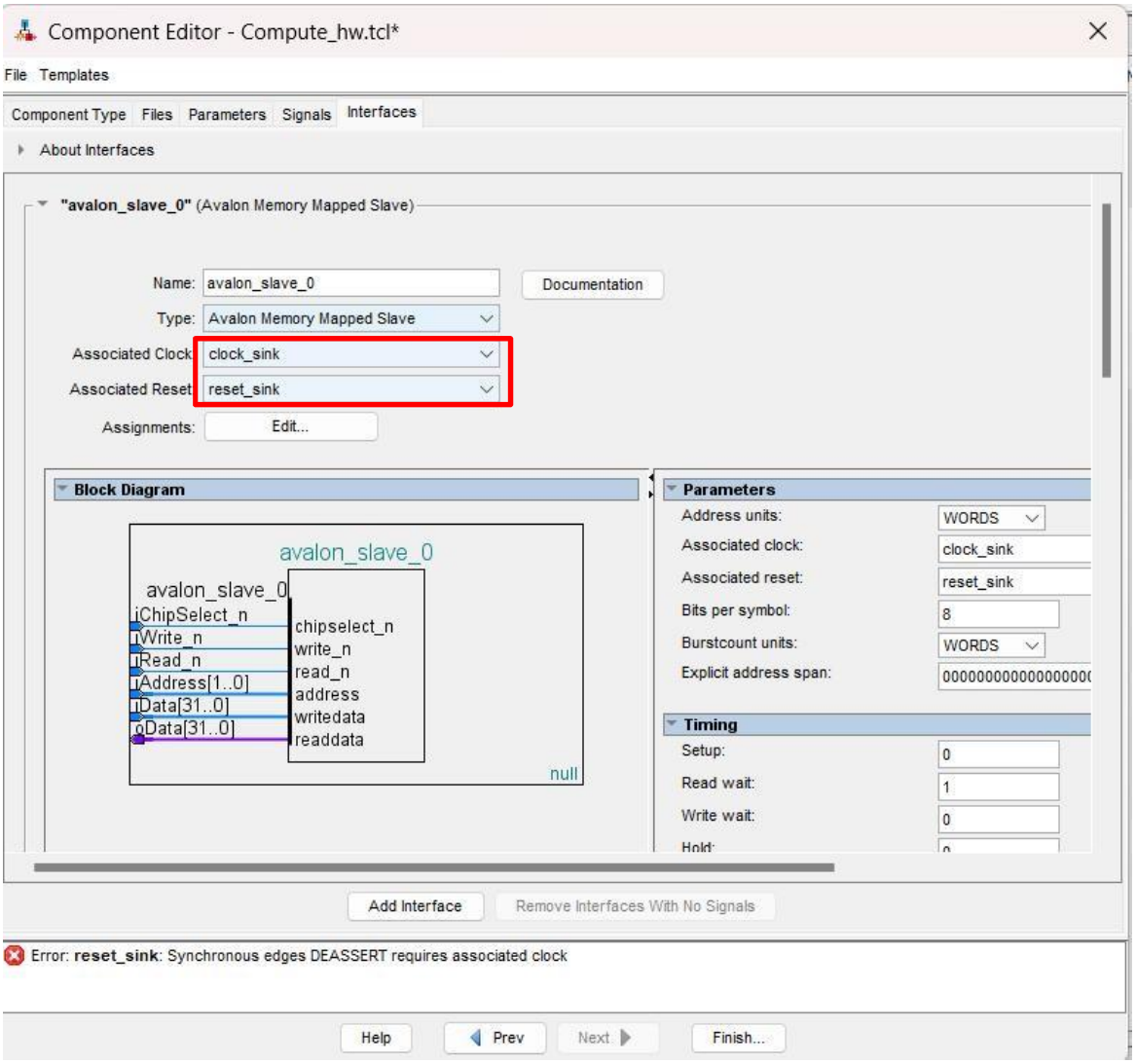
Hình 3. Hệ thống chuẩn bị ban đầu.

- Thêm mô đun **Compute** và cấu hình tín hiệu như hình 4.

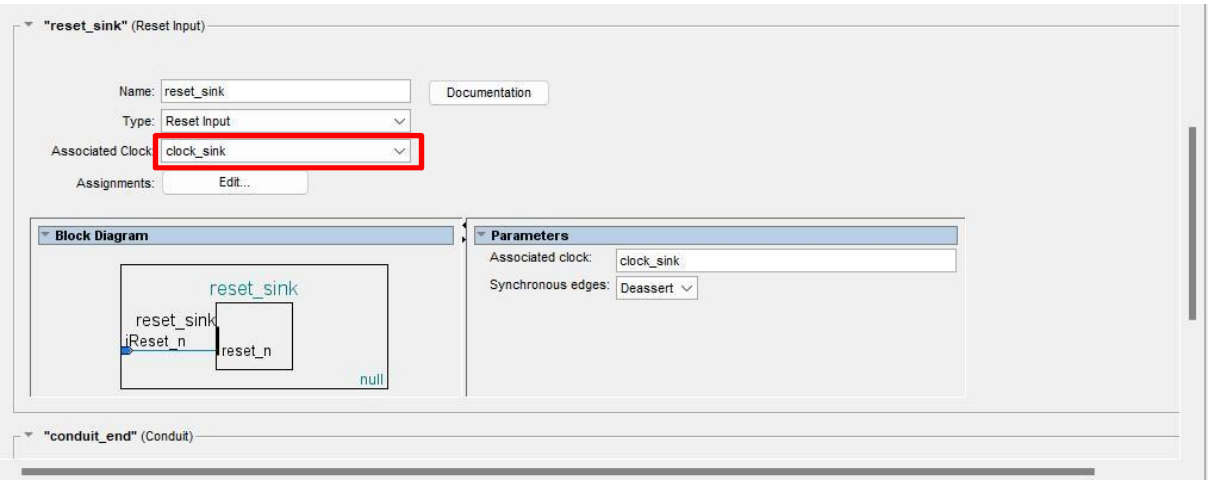


Hình 4. Cấu hình các tín hiệu của mô đun Compute.

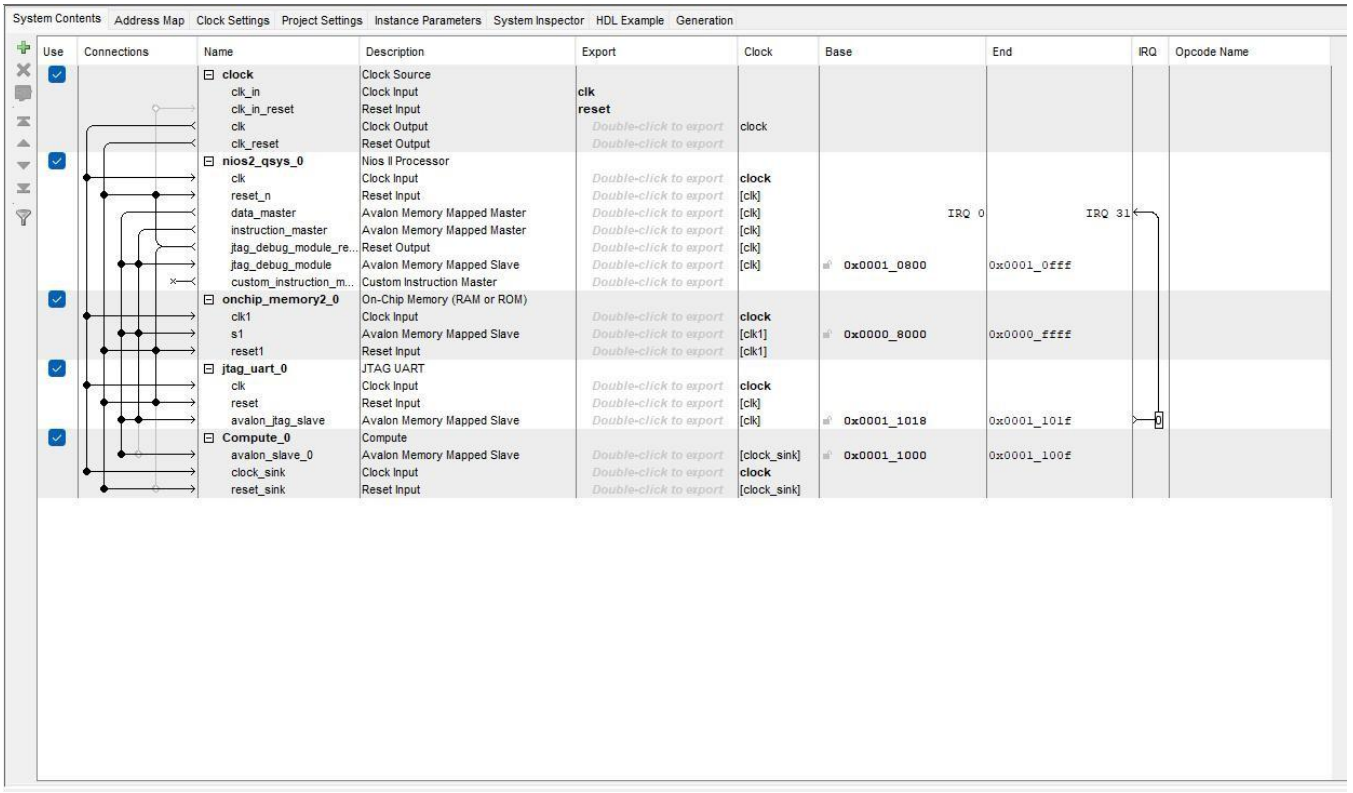
- Chuyển sang tab **Signals & Interfaces**, cấu hình tín hiệu **clock** và **reset** cho interface **avalon_slave_0**, và tín hiệu clock cho interface **reset_sink**, hình 5.



Hinh 5. Chọn clock và reset cho interface avalon_slave_0.



- Tiếp đến chọn **Finish** và lưu lại mô đun.
- Thêm mô đun vào hệ thống và kết nối tín hiệu, hình 6.



Hình 6. Hệ thống hoàn chỉnh.

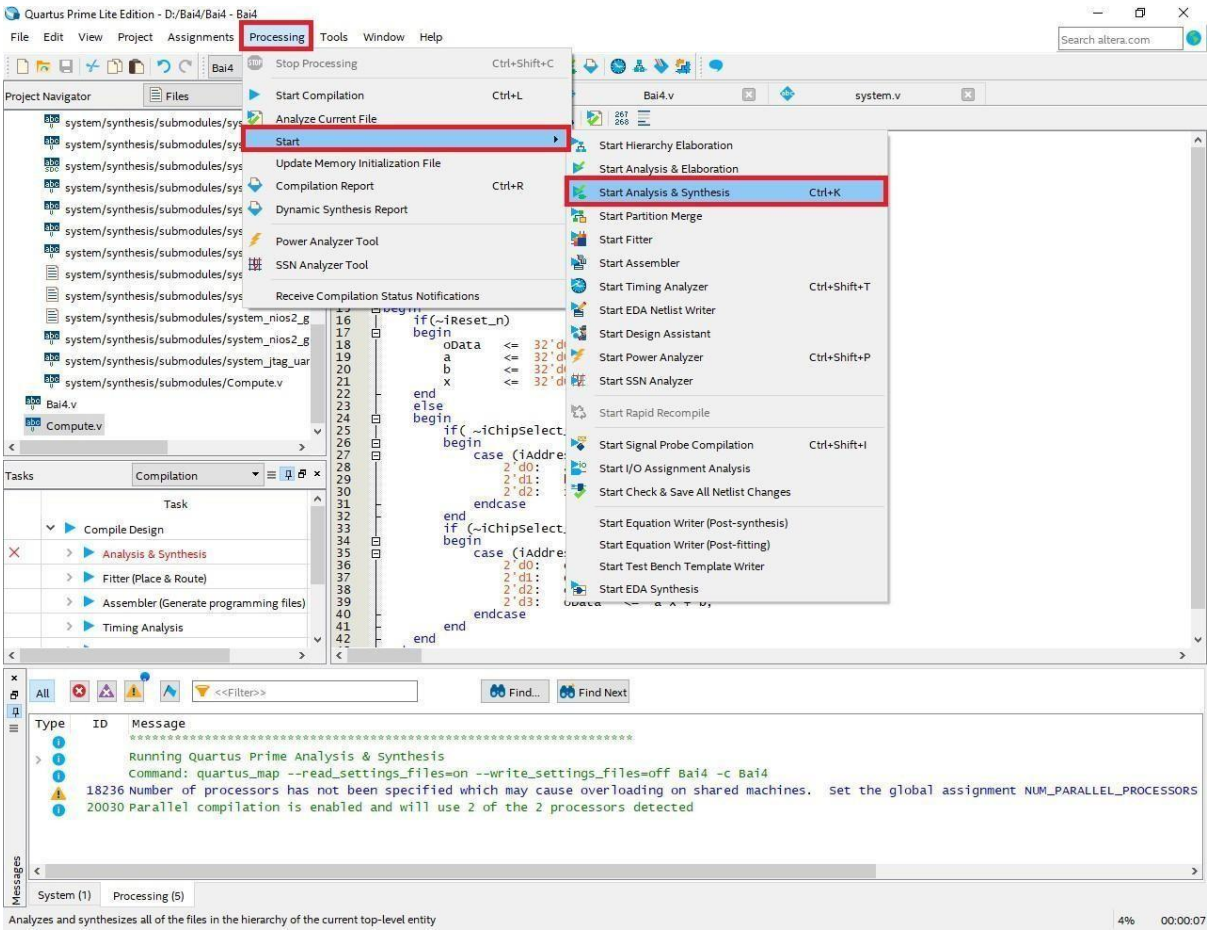
- Gán lại địa chỉ: **System → Assign Base Addresses**.
- Lưu hệ thống dưới tên là **system** và **generate** hệ thống.

3.4. Tích hợp phần cứng

- Thêm file **system.qip.**
- Tạo file top- level là **Bai4.v** được mô tả như đoạn code bên dưới để tổng hợp hệ thống.

```
module Bai4(
        input                   CLOCK_50,
        input           [0:0]   KEY
        );
    system Nios_system (
                .clk_clk                (CLOCK_50),
                .reset_reset_n          (KEY[0])
                );
Endmodule
```

- Gán chân cho thiết bị: **Assignments → Import Assignments.**
- Tiếp theo tiến hành phân tích và tổng hợp phần cứng bằng cách chọn **Processing → Start → Start Analysis & Synthesiss** (Ctrl + K), hình 7.

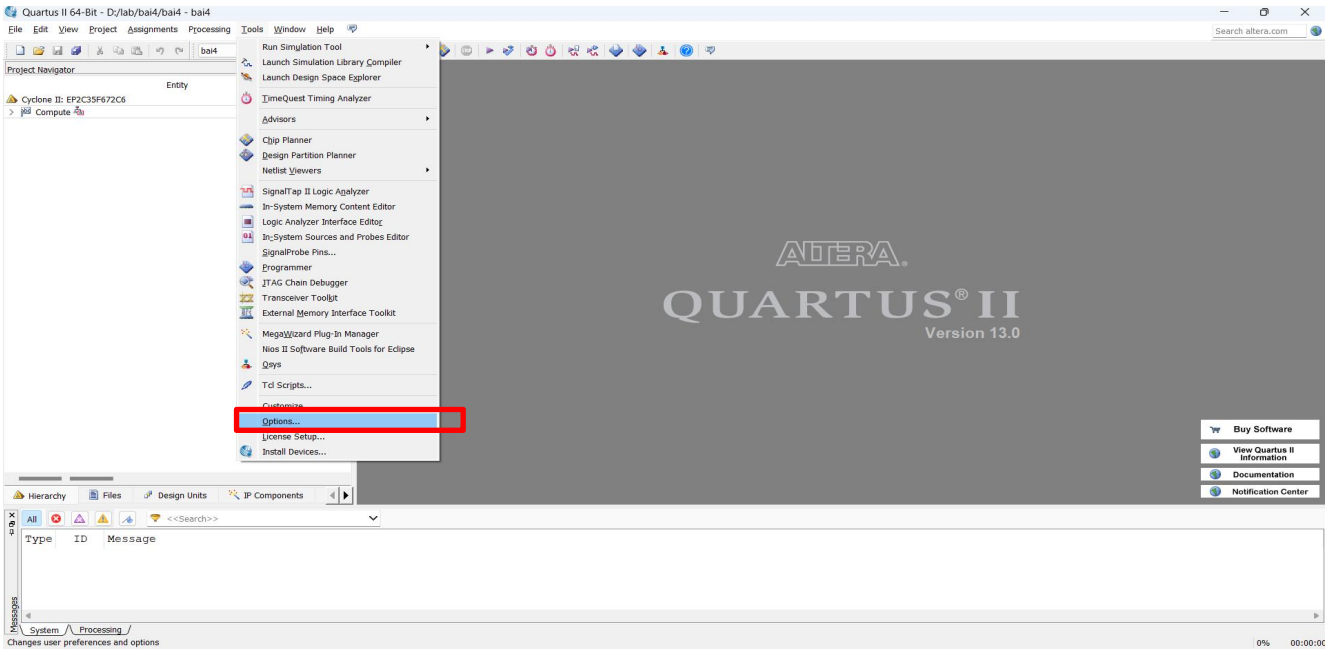


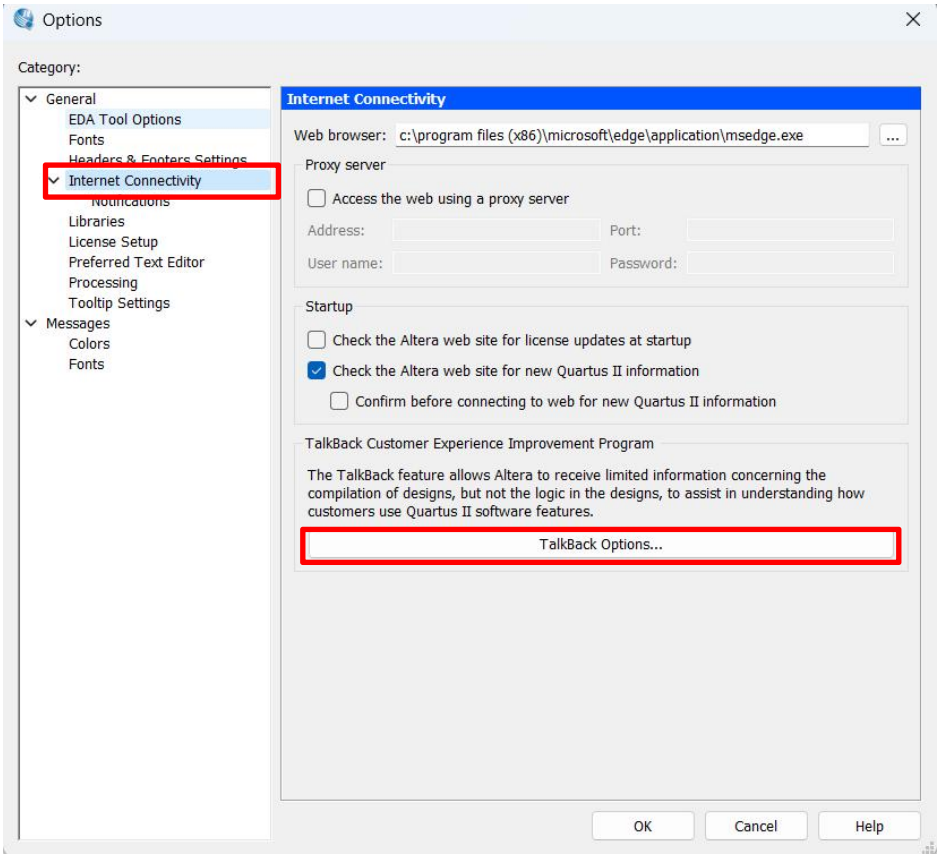
Hình 7. Tiến hành phân tích và tổng hợp hệ thống.

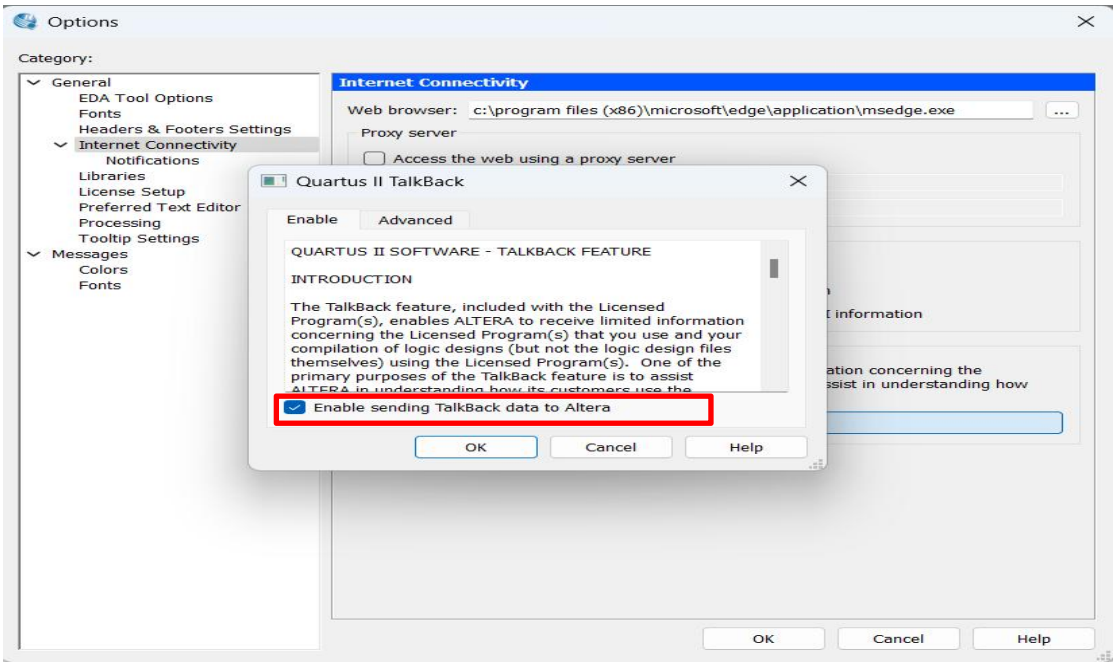
3.5. Cấu hình trên Signal Tab

Khi sử dụng Signal Tap (trước đây là SignalTap II) hoặc Signal Probe với Quartus® Prime Lite Edition hoặc Quarus® II Web Edition, bạn có thể sử dụng nó bằng cách bật chức năng TalkBack.

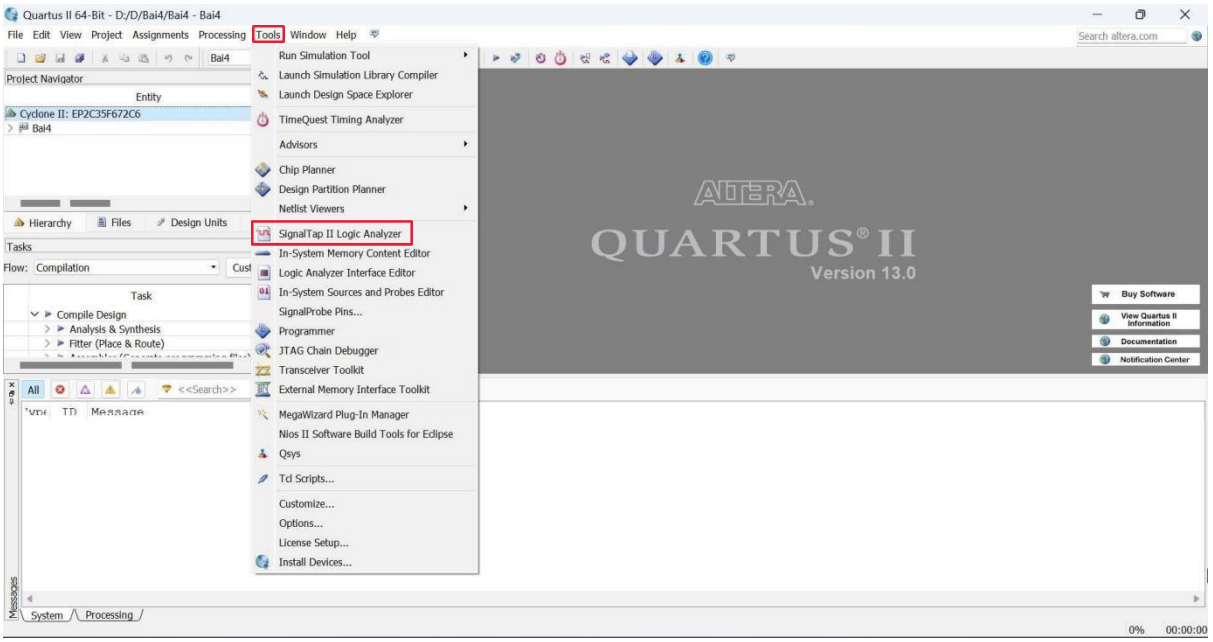
- Select Options from the Tools menu
- Select Internet Connectivity in the Options window
- Click TalkBack Options
- Enable Turn on the Quartus II software TalkBack eature in the TalkBack Options window


Quartus II 64-Bit - D:/lab/bai4/bai4 - bai4
File Edit View Project Assignments Processing Tools Window Help
Run Simulation Tool
Launch Simulation Library Compiler
Launch Design Space Explorer
TimeQuest Timing Analyzer
Advisors
Chip Planner
Design Partition Planner
Netlist Viewers
SignalTap II Logic Analyzer
In-System Memory Content Editor
Logic Analyzer Interface Editor
In-System Sources and Probes Editor
SignalProbe Pins...
Programmer
JTAG Chain Debugger
Transceiver Toolkit
External Memory Interface Toolkit
MegaWizard Plug-In Manager
Nios II Software Build Tools for Eclipse
Qsys
Tcl Scripts...
Options...
License Setup...
Install Devices...
ALTERA
QUARTUS® II
Version 13.0
Buy Software
View Quartus II Information
Documentation
Notification Center
Type ID Message


Options
Category:
General
EDA Tool Options
Fonts
Internet Connectivity
Libraries
License Setup
Preferred Text Editor
Processing
Tooltip Settings
Messages
Colors
Fonts
Internet Connectivity
Web browser: c:\program files (x86)\microsoft\edge\application\msedge.exe
Proxy server
Access the web using a proxy server
Address:
Port:
User name:
Password:
Startup
Check the Altera web site for license updates at startup
Check the Altera web site for new Quartus II information
Confirm before connecting to web for new Quartus II information
TalkBack Customer Experience Improvement Program
The TalkBack feature allows Altera to receive limited information concerning the compilation of designs, but not the logic in the designs, to assist in understanding how customers use Quartus II software features.
TalkBack Options...
OK
Cancel
Help

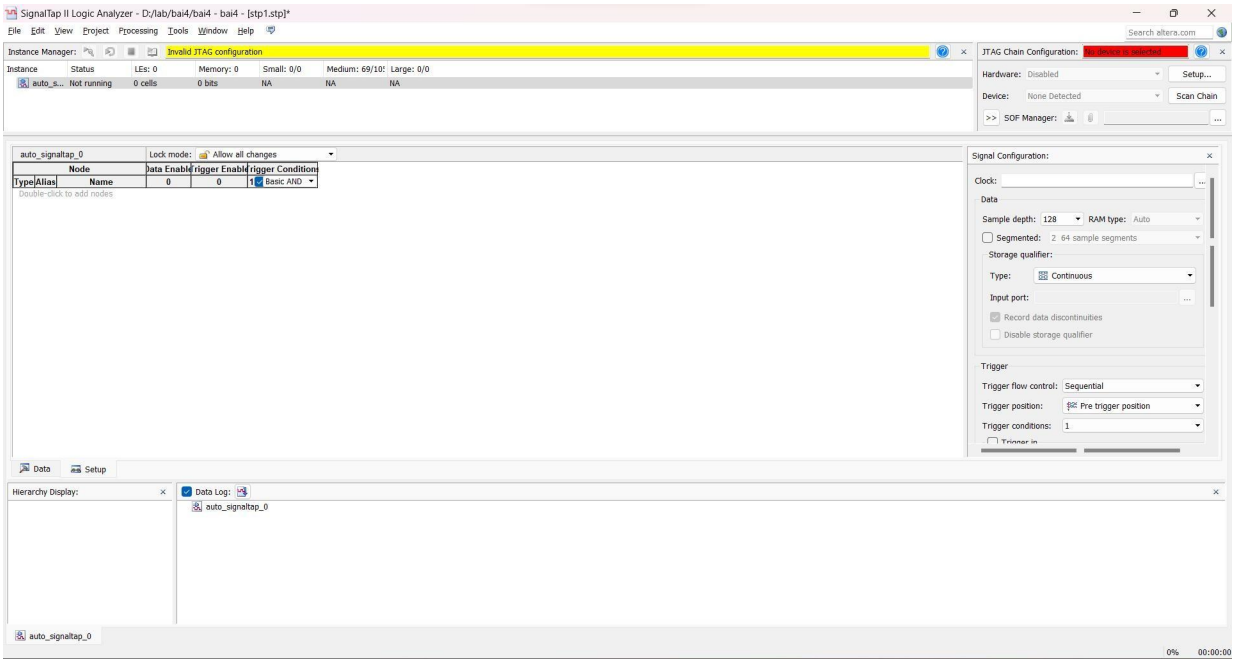

Options
Category:
General
EDA Tool Options
Fonts
Headers & Footers Settings
Internet Connectivity
Notifications
Libraries
License Setup
Preferred Text Editor
Processing
Tooltip Settings
Messages
Colors
Fonts
Internet Connectivity
Web browser: c:\program files (x86)\microsoft\edge\application\msedge.exe
Proxy server
Access the web using a proxy server
Quartus II TalkBack
Enable
Advanced
QUARTUS II SOFTWARE - TALKBACK FEATURE
INTRODUCTION
The TalkBack feature, included with the Licensed Program(s), enables ALTERA to receive limited information concerning the Licensed Program(s) that you use and your compilation of logic designs (but not the logic design files themselves) using the Licensed Program(s). One of the primary purposes of the TalkBack feature is to assist
Enable sending TalkBack data to Altera
OK
Cancel
Help
OK
Cancel
Help

Mở Signal Tab bằng cách chọn **Tools →Signal Tab II Logic Analyzer**, hình 8.



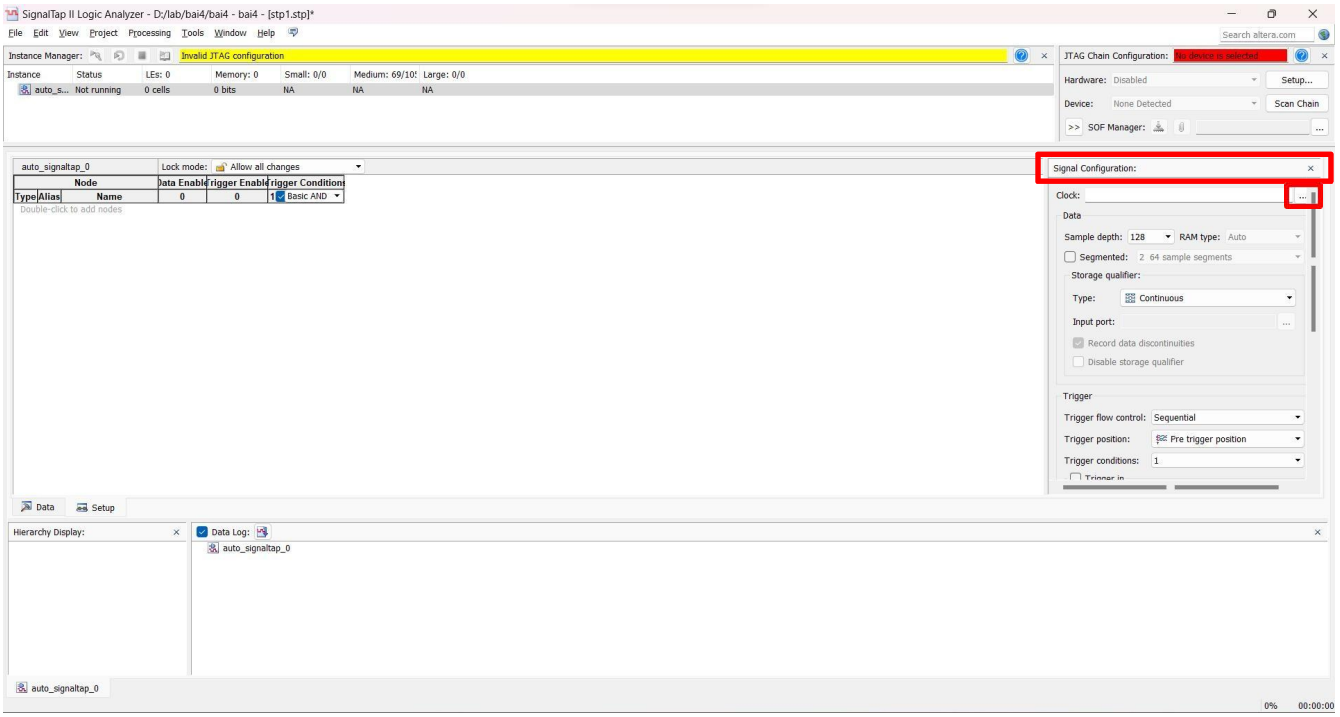
Hình 8. Mở phần mềm Signal Tab.

- Hình 9 thể hiện giao diện của phần mềm Signal Tab



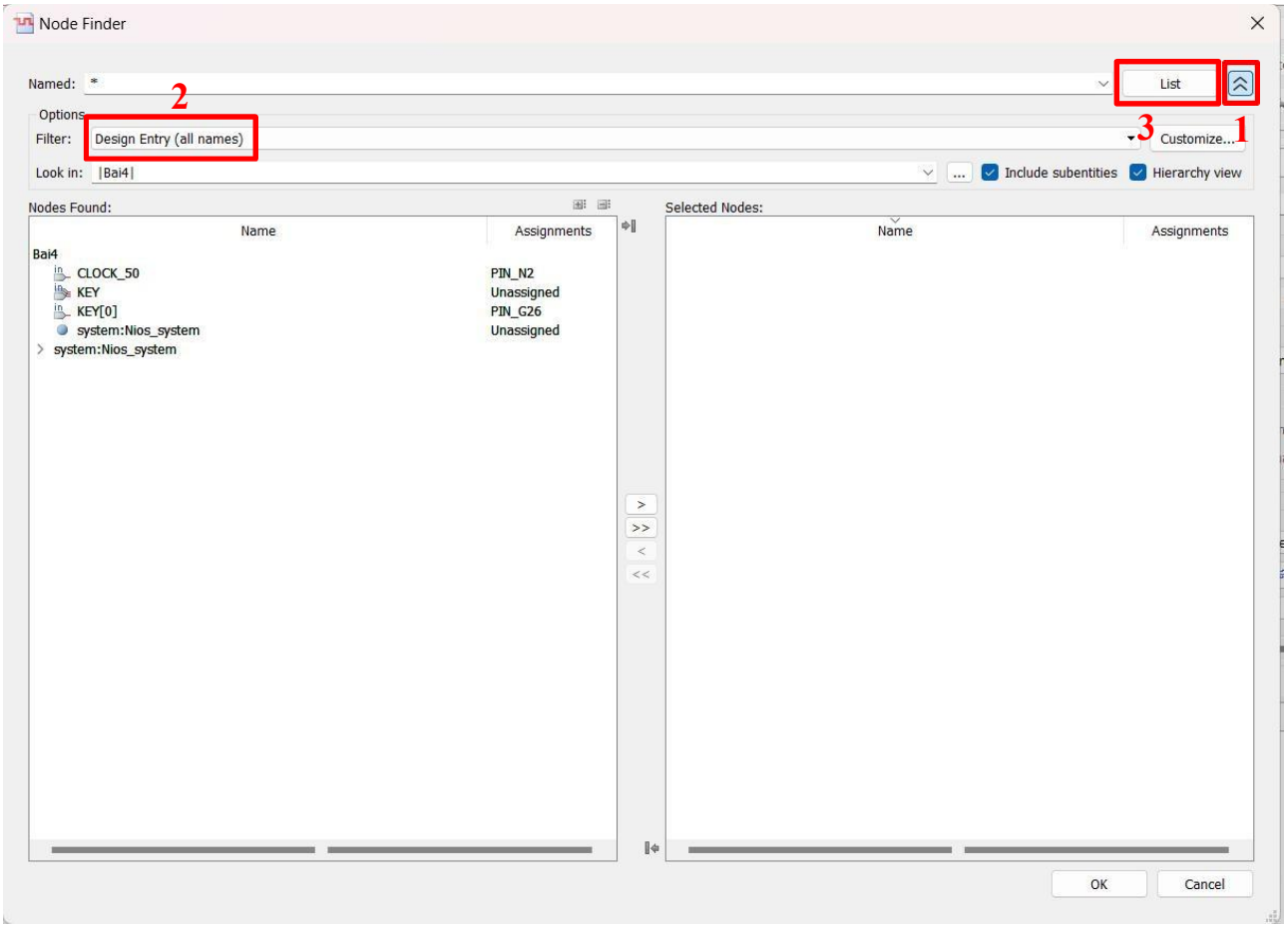
Hinh 9. Giao diện phần mềm Signal Tab.

- Đầu tiên, tiến hành cấu hình **clock** để hoạt động, chọn như hình 10. Ở tab Signal Configuration, chọn nút ba chấm.



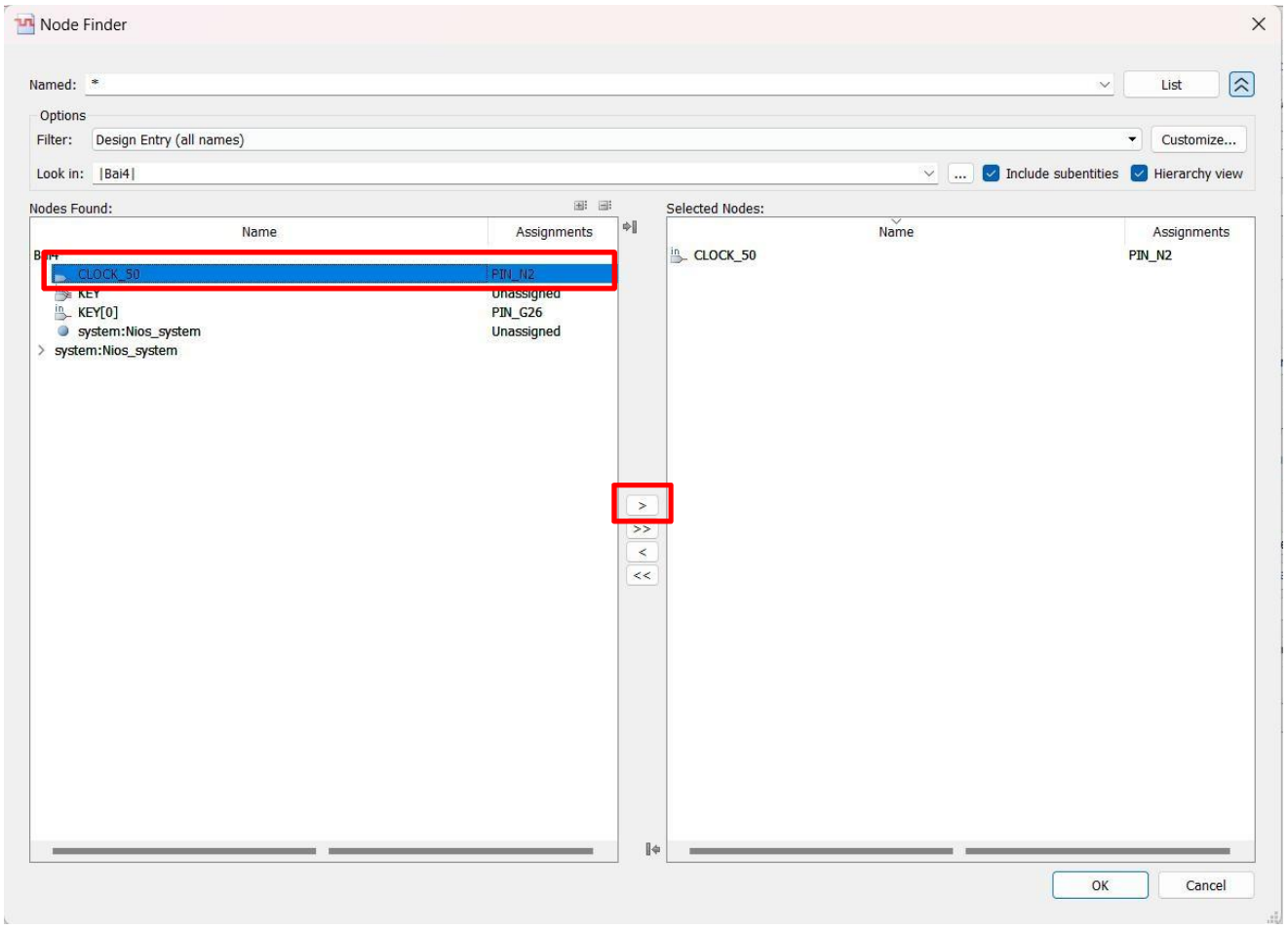
Hình 10. Cấu hình clock cho Signal Tab.

- Ở cửa sổ **Node Finder**, lần lượt chọn theo thứ tự được đánh số như hình 11.



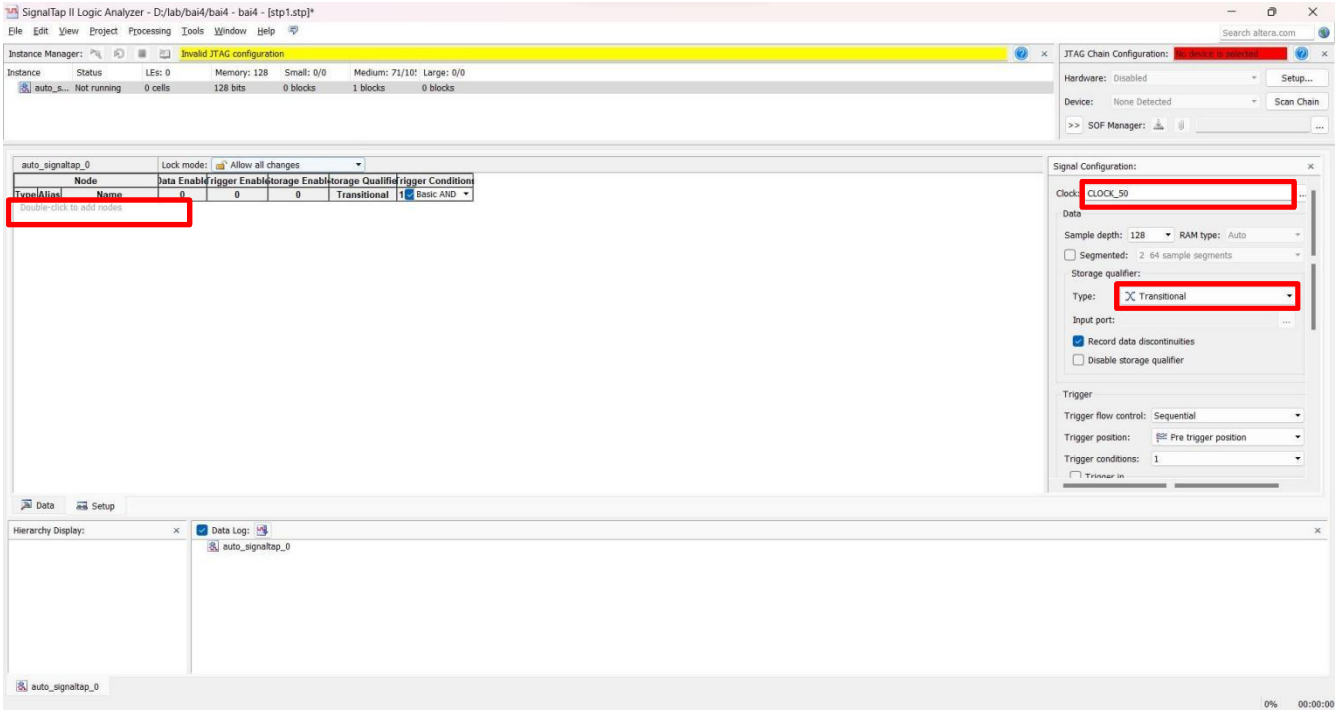
Hình 11. Cấu hình ở cửa sổ Node Finder.

- Khi thực hiện xong, ở tab Nodes found sẽ xuất hiện các note có thể thêm, chọn **CLOCK_50** và thêm theo thứ như hình 12. Kết quả và tín hiệu **CLOCK_50** ở tab **Selected Nodes**. Cuối cùng chọn OK để kết thúc.



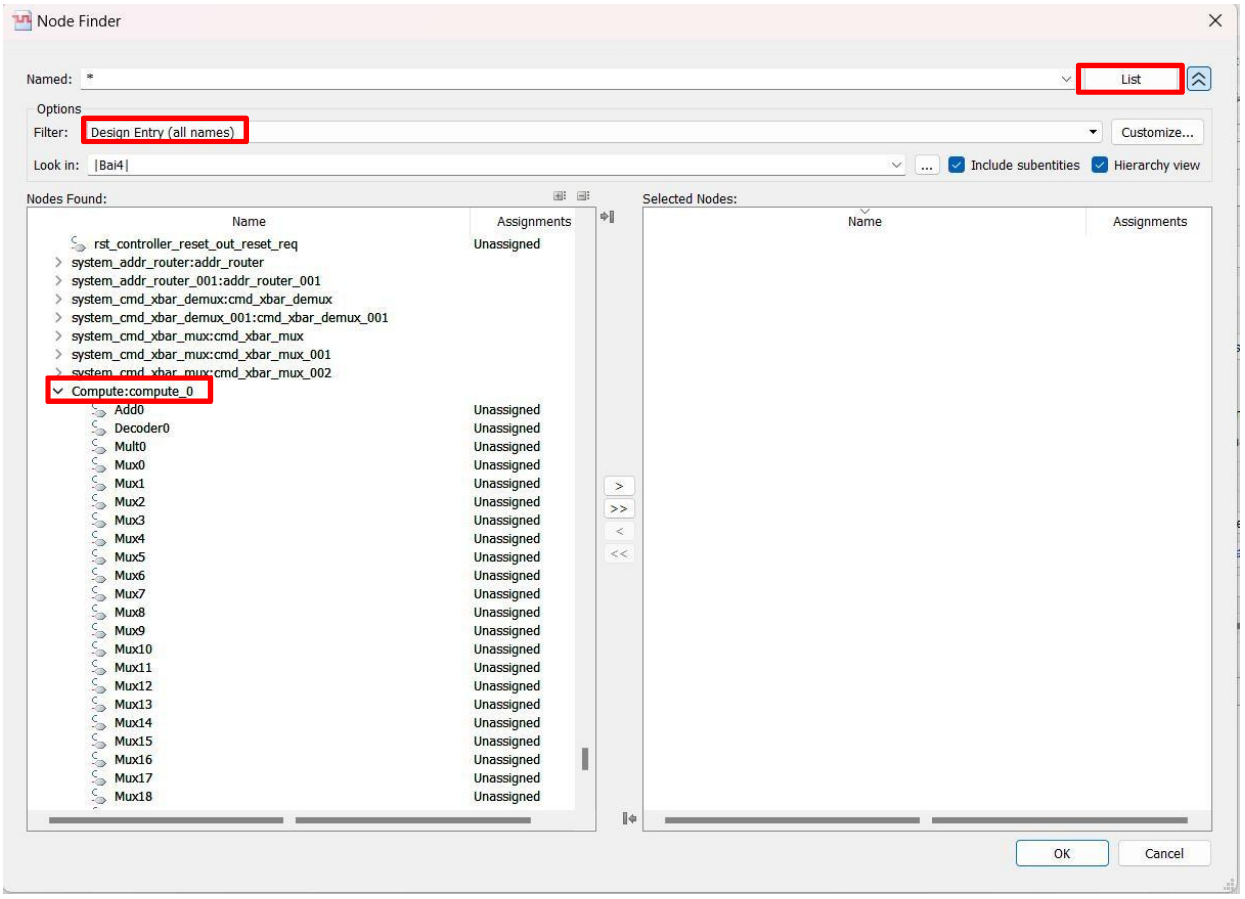
Hình 12. Thêm tín hiệu CLOCK_50.

- Tiếp tục cấu hình chất lượng lưu trữ là **Transitional** như hình 13. Hình 13 còn thể hiện kết quả của việc thêm **CLOCK_50** làm clock hoạt động (khung màu đỏ phía trên bên phải hình 13) và nơi để thêm tín hiệu cần xem (khung màu đỏ phím trên bên trái hình 13).

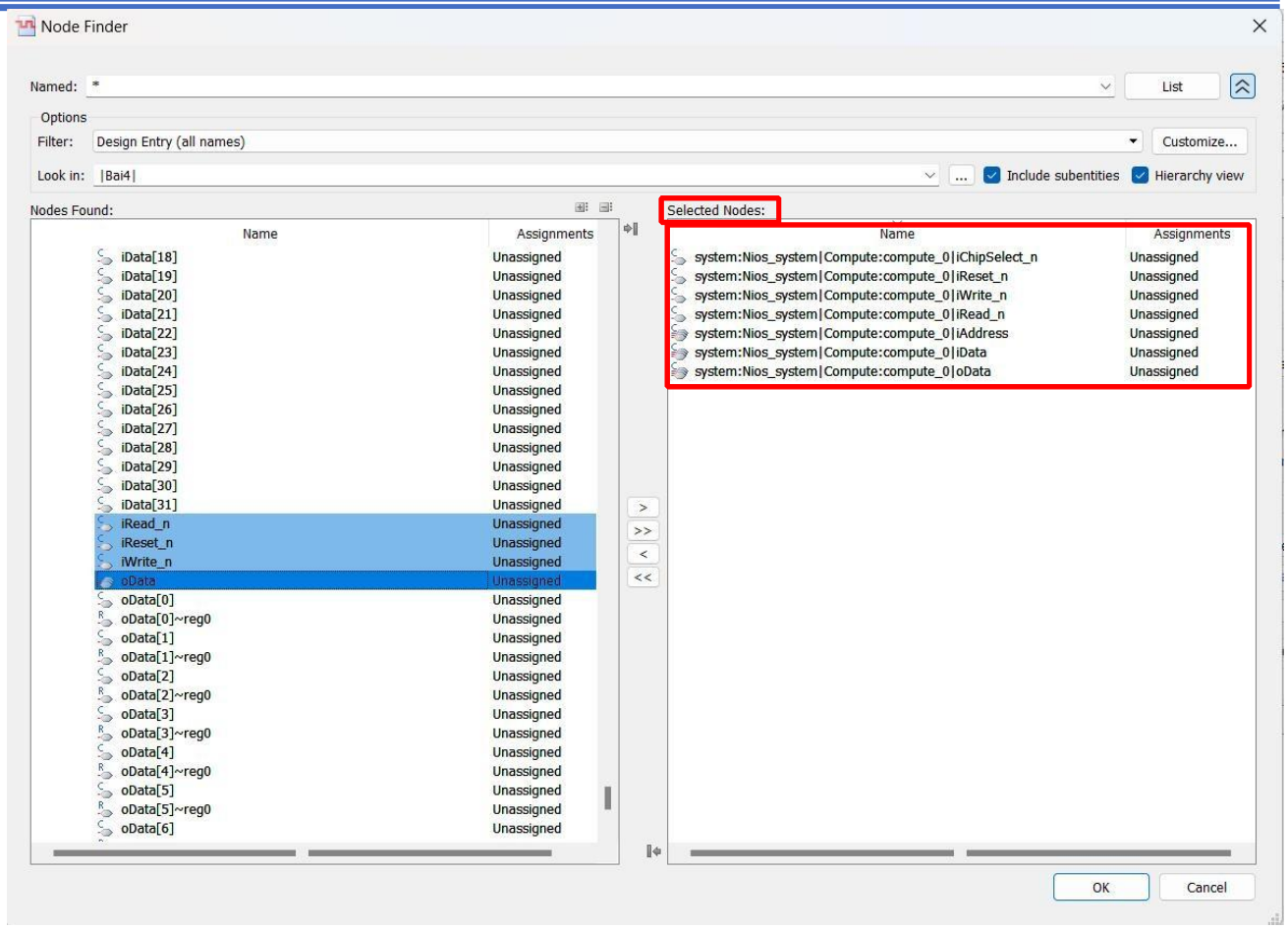


Hinh 13. Cấu hinh chất lượng lưu trữ là Transitional.

- Để thêm tín hiệu cần xem, nhấn đúp chuột vào vị trí khùng màu đỏ bên trái phía trên của hình 13, sau đó chọn như hình 15. Ở cửa sổ **Node Finder**, ta chọn **Filter** là **Design Entry (all names)** và chọn **list**. Khi các note hiện ra, mở system: **Nios_system**, tìm mô đun **Compute**, hình 14, rồi chọn tín hiệu thêm vào như hình 15.



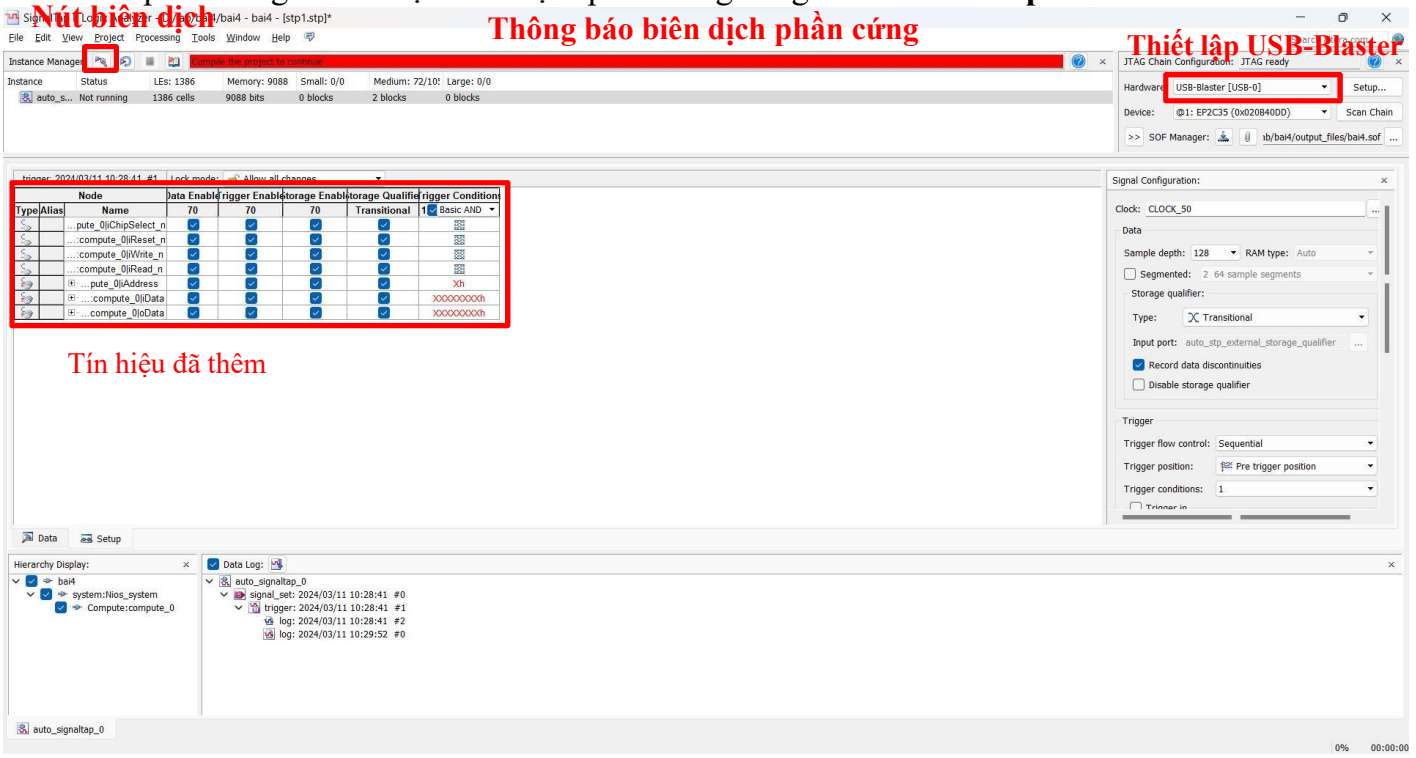
Hình 14. Mô đun Compute trong Nios_system.



Hình 15. Kết qua thêm các tín hiệu.

- Khi chọn xong tín hiệu như hình 15, chọn **OK**.

- Tiến hành cấu hình lại **USB Blaster** như hình 16 và khi có thông báo biên dịch phần cứng thì ta chọn biên dịch phần cứng bằng nút **Start Compilion**.

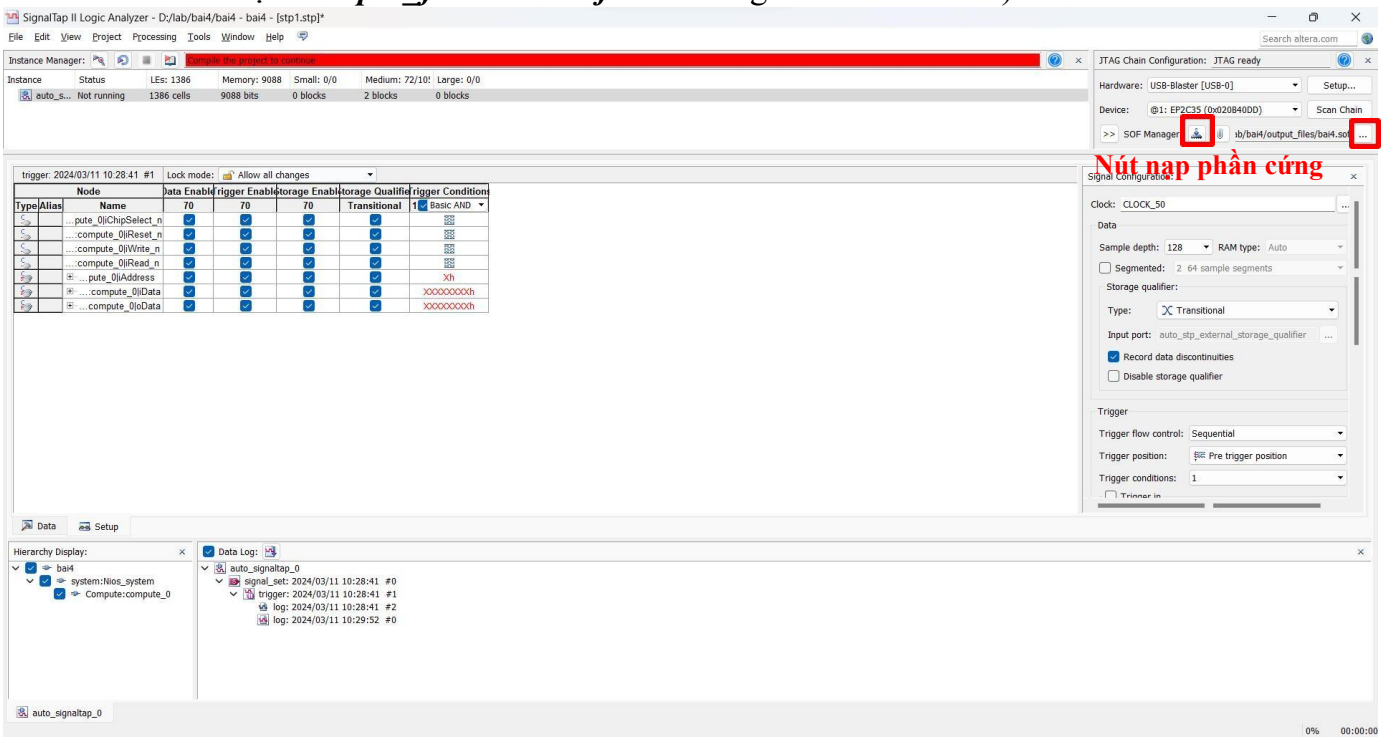


Hình 16. Kết qua thêm tín hiệu và thiết lập lại USB-Blaster.

- Khi được hỏi lưu file signal tab thì chọn yes và đặt tên là **stp1.stp**. Tiếp đến chọn **Yes** như hình 17.

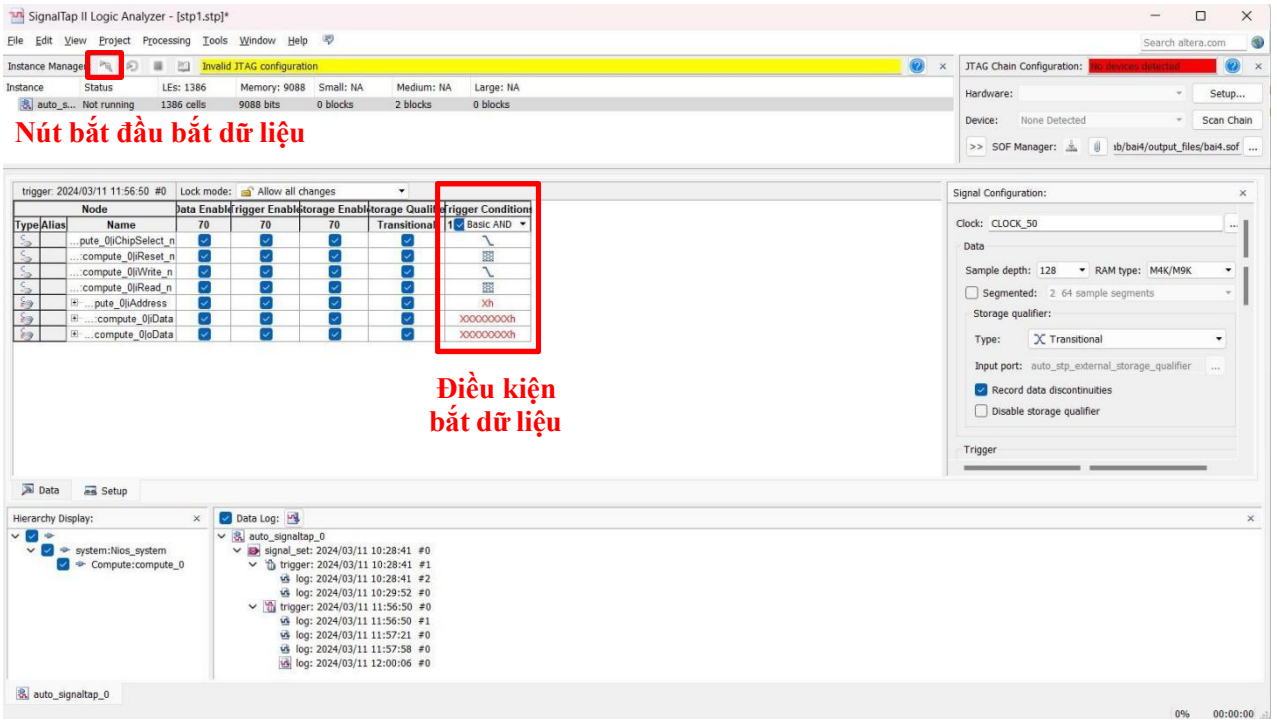
Hình 17. Cho phép Signal Tab hoạt động.

- Sau khi biên dịch xong sẽ có thông báo nạp phần cứng xuống board và nạp phần cứng xuống như hình 18. (Khi nút nạp phần cứng bị mờ thì phải dẫn file **Bai4.sof** ở thư mục: ***output_files*****Bai4.sof*** vào đường dẫn như hình 18).



Hình 18. Nạp phần cứng xuống board.

- Khi nạp thành công, chọn nút như hình 19 để bắt đầu bắt dữ liệu.



Hình 19. Cấu hình điều kiện bắt dữ liệu và bắt đầu quá trình hoạt động bắt dữ liệu.

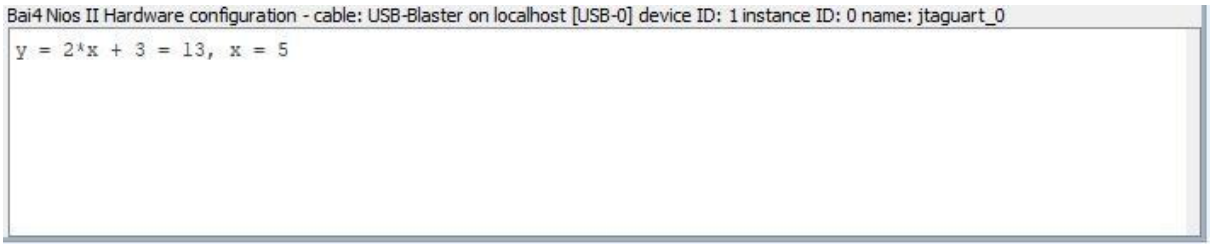
3.6. Xây dựng phần mềm

- Tạo và đặt tên project là "**Bai4**".
- Thêm file "**source.c**" vào project "**Bai4". File "source.c**" có nội dung như đoạn code bên dưới.

```
#include "stdio.h"
#include "io.h"
#include "system.h"
void main(){
	int a, b, x, y;
	//y = 2x + 3 , x = 5
	a = 2 & 0xf;
	b = 3 & 0xf;
	x = 5 & 0xf;
	IOWR(COMPUTE_0_BASE, 0, a);
	IOWR(COMPUTE_0_BASE, 1, b);
	IOWR(COMPUTE_0_BASE, 2, x);
	y = IORD(COMPUTE_0_BASE, 3);
	printf("y = %d*x + %d = %d, x = %d\n", a, b, y, x );
```

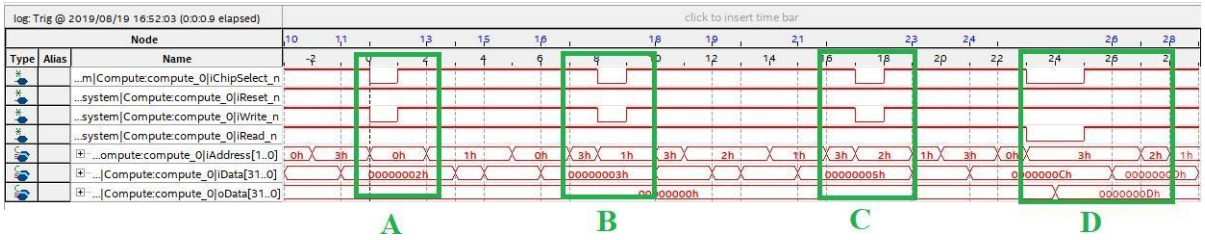
```
}
```

- Build project và dowload phần mềm xuống board.



Hình 20. Kết qua trên console.

- Quan sát kết quả trên Signal Tab, hình 22.



Hình 21. Kết qua trên Signal Tab.

- Kết quả trong hình 22 thể hiện 4 quá trình A, B, C, D. Trong đó, A, B, C tương ứng với 3 quá trình ghi dữ liệu của thanh ghi: a, b, x và quá trình D là quá trình đọc dữ liệu từ thanh ghi y.

BÀI TẬP CHUẨN BỊ Ở NHÀ

Bài 1. Dựa vào kết quả chạy trên Signal Tab (hình 22), giải thích sự tương qua giữa code C và code verilog của mô đun Compute trong việc đọc ghi dữ liệu.

BÁO CÁO THỰC HÀNH

Bài 1. Thực hiện lại hệ thống như bài hướng dẫn thực hành, chỉnh sửa code C để truy xuất mô đun Compute từ Nios II thông qua con trỏ.

Bài 2. Xây dựng hệ thống SoC như bài hướng dẫn thực hành, nhưng các giá trị a, b, x được nhập từ SW và kết quả xuất ra LEDG như sau:

  - a = SW[3:0].
  - b = SW[7:4].
  - c = SW[11:8].
  - LEDG[7:0] = y.

- Bắt dữ liệu để hiện thị trên Signal Tab.

TÀI LIỆU THAM KHẢO

[1] DE2_115_User_Manual.

[2] Embedded Peripherals IP User Guide.

[3] Nios II Processor Reference.

[4] Avalon Interface Specification.

[5] SignalTap II with Verilog Design.