BÀI 5. THIẾT KẾ HỆ THỐNG VỚI BỘ NHỚ ĐƠN GIẢN

1. Mục tiêu

Thông qua bài thực hành này, sinh viên sẽ biết cách:

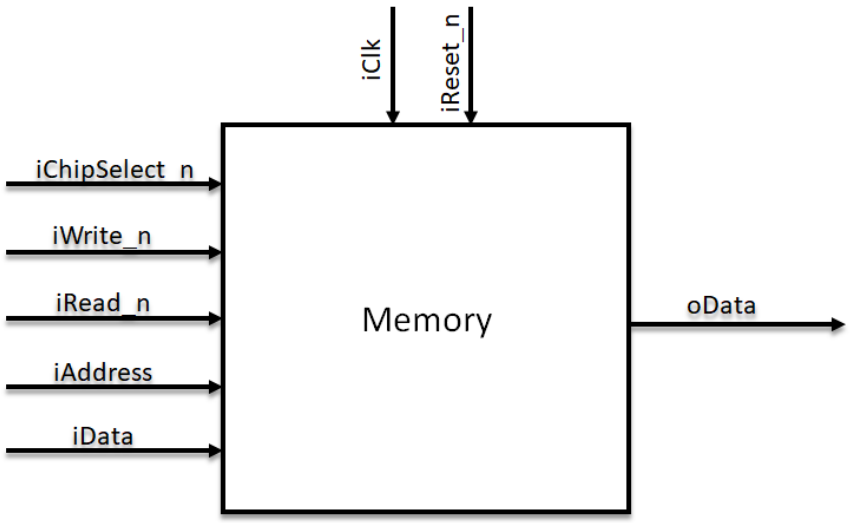
- Thiết kế một bộ nhớ đơn giản bằng code Verilog.
- Biết cách thêm component tự thiết kế vào hệ thống.
- Biết cách khảo sát bằng công cụ Signal Tab.

2. Phần lý thuyết

2.1. Tổng quan mô đun thiết kế

Tiến hành thiết kế mô đun **Memory** có dung lượng 16 x 32 bits: có 16 thanh ghi, mỗi thành ghi có độ rộng 32 bits.

Các tín hiệu của bộ nhớ được mô tả ở hình 1 và bảng 1.



Hình 1. Tín hiệu của mô đun Memory.

Bảng 1. Bảng mô tả tín hiệu của mô đun Memory.

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	4	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.

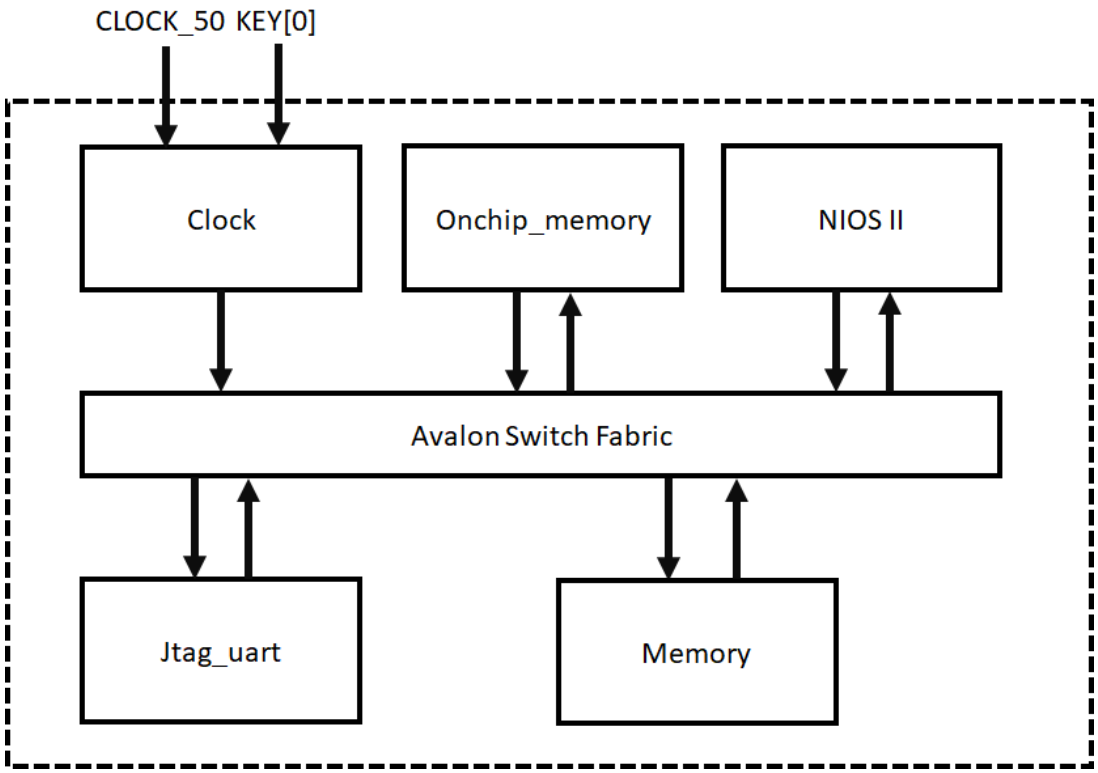
2.2. Code verilog mô tả mô đun Memory

```
module Memory
 #(
     parameter DATA_WIDTH       = 32,
     parameter ADDRESS_WIDTH    = 4
 )
 (
    input                                iClk,
    input                                iReset_n,
    input                                iChipSelect_n,
    input                                iRead_n,
    input                                iWrite_n,
    input       [ADDRESS_WIDTH - 1 : 0] iAddress,
    input       [DATA_WIDTH - 1 : 0]    iData,
    output      [DATA_WIDTH - 1 : 0]    oData
 );

 reg [DATA_WIDTH - 1 : 0] mem [2**ADDRESS_WIDTH - 1 : 0];
 reg [ADDRESS_WIDTH - 1 : 0] address_reg;
 always@(posedge iClk)
 begin
     if (~iChipSelect_n & ~iWrite_n)
     begin
         mem[iAddress] <= iData;
     end
     if(~iChipSelect_n & ~iRead_n)
     begin
         address_reg <= iAddress;
     end
 end

 assign oData = mem[address_reg];
 endmodule
```

3. Phần thực hành

3.1. Tổng quan hệ thống

Hệ thống được thiết kế như hình 2 bên dưới:



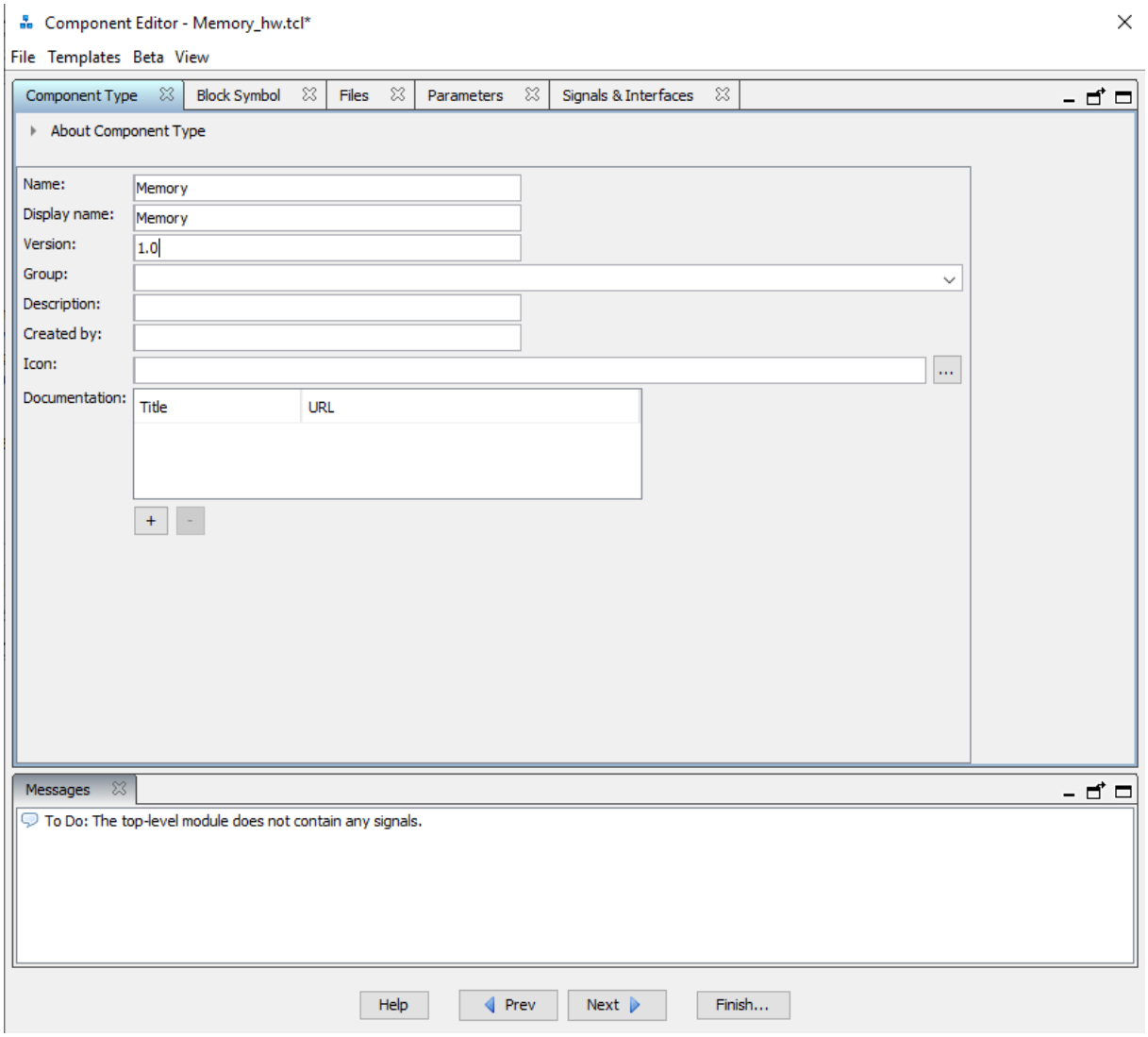
Hình 2. Tổng quan hệ thống phần cứng.

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 board DE2-115, chọn Family là **Cyclone IV E**, device là **EP4CE115F29C7**.

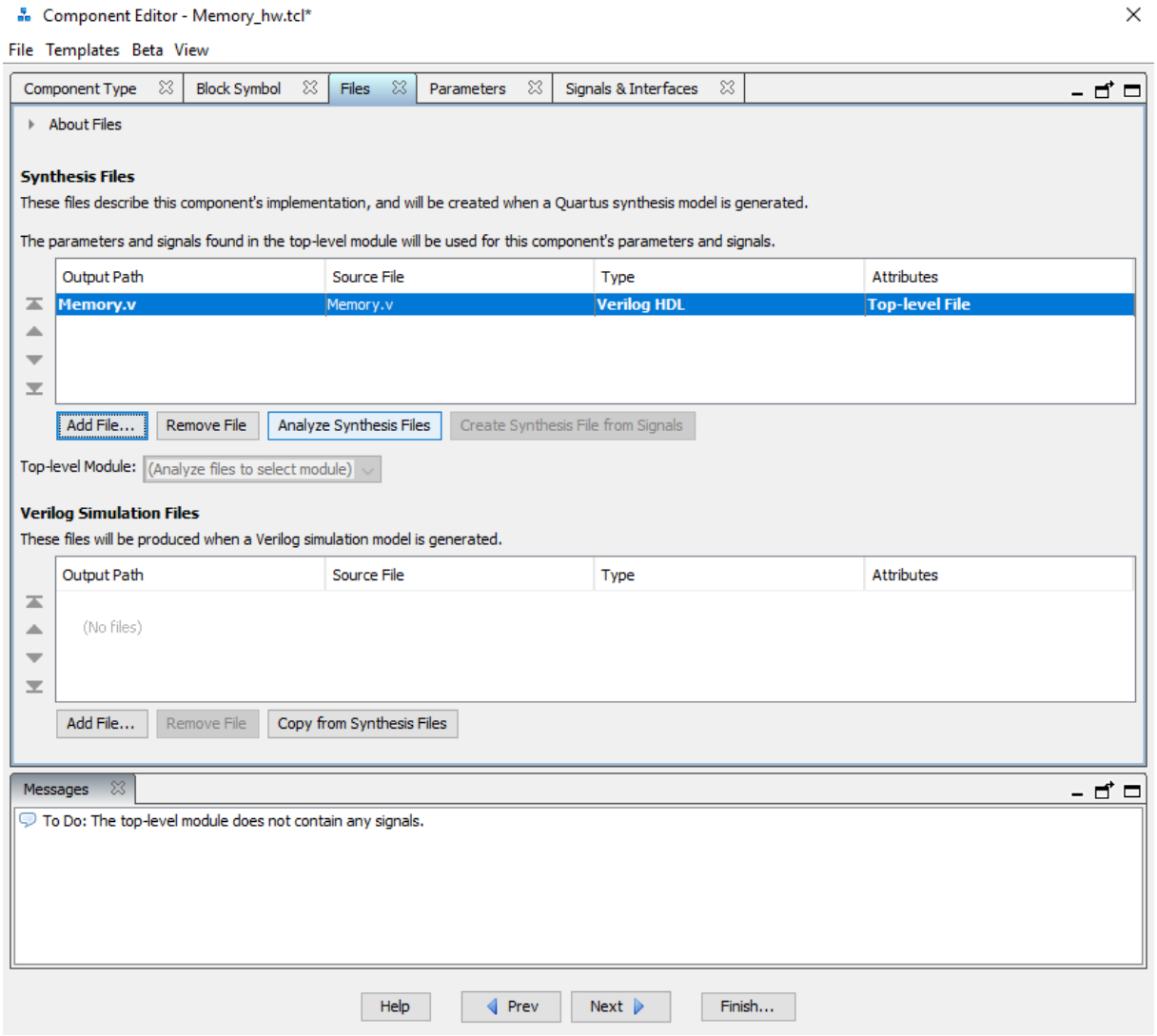
3.3. Xây dựng phần cứng trên Platform Designer

- Tiến hành thêm mô đun Memory vào thư viện như các hình bên dưới. Đầu tiên đặt mô đun là **Memory**, hình 3.



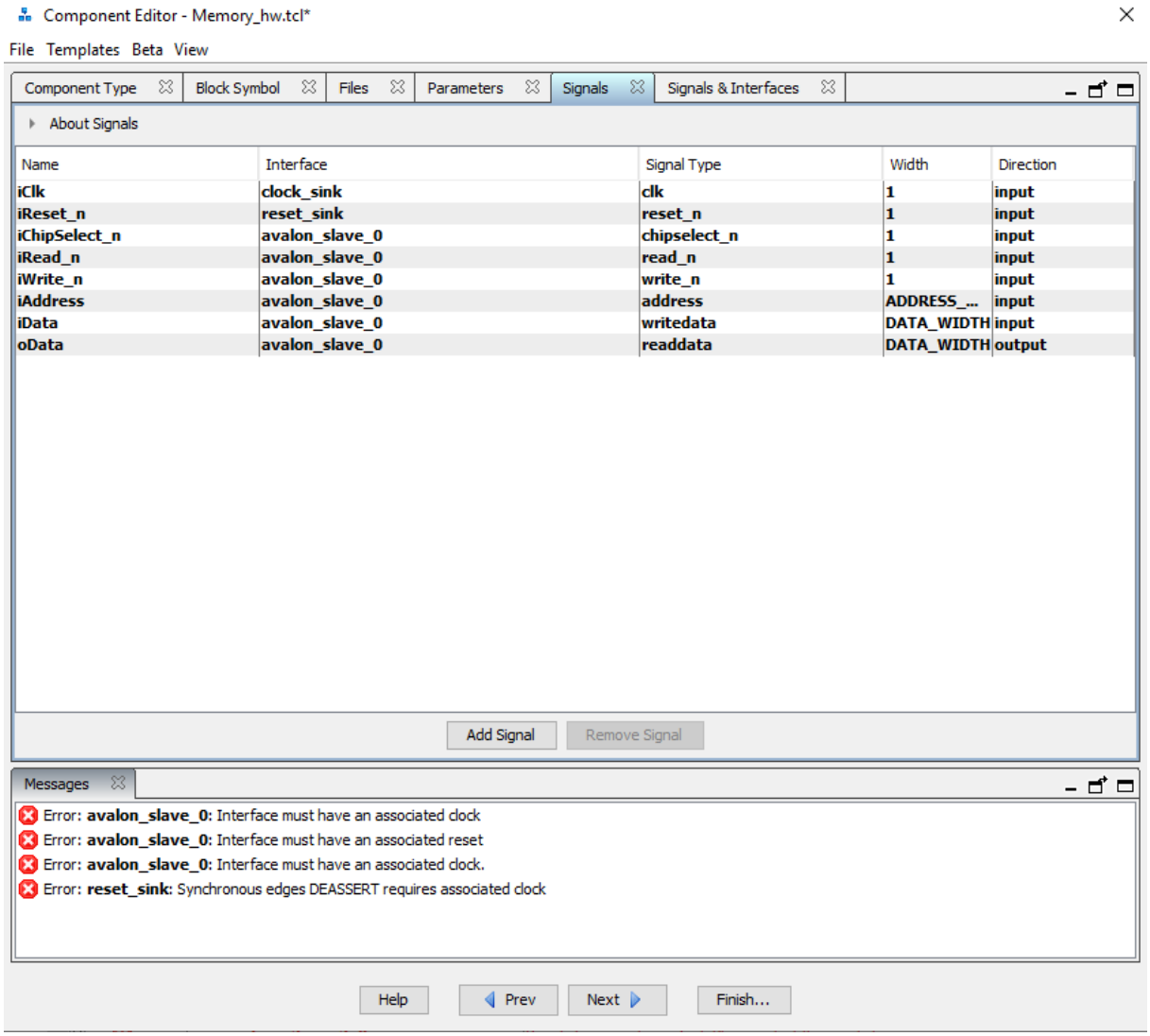
Hình 3. Đặt tên mô đun là Memory.

- Tiếp theo chuyển qua tab **Files**, thêm file **Memory.v** rồi tiến hành phân tích bằng **Analyze Synthesis Files**, hình 4.



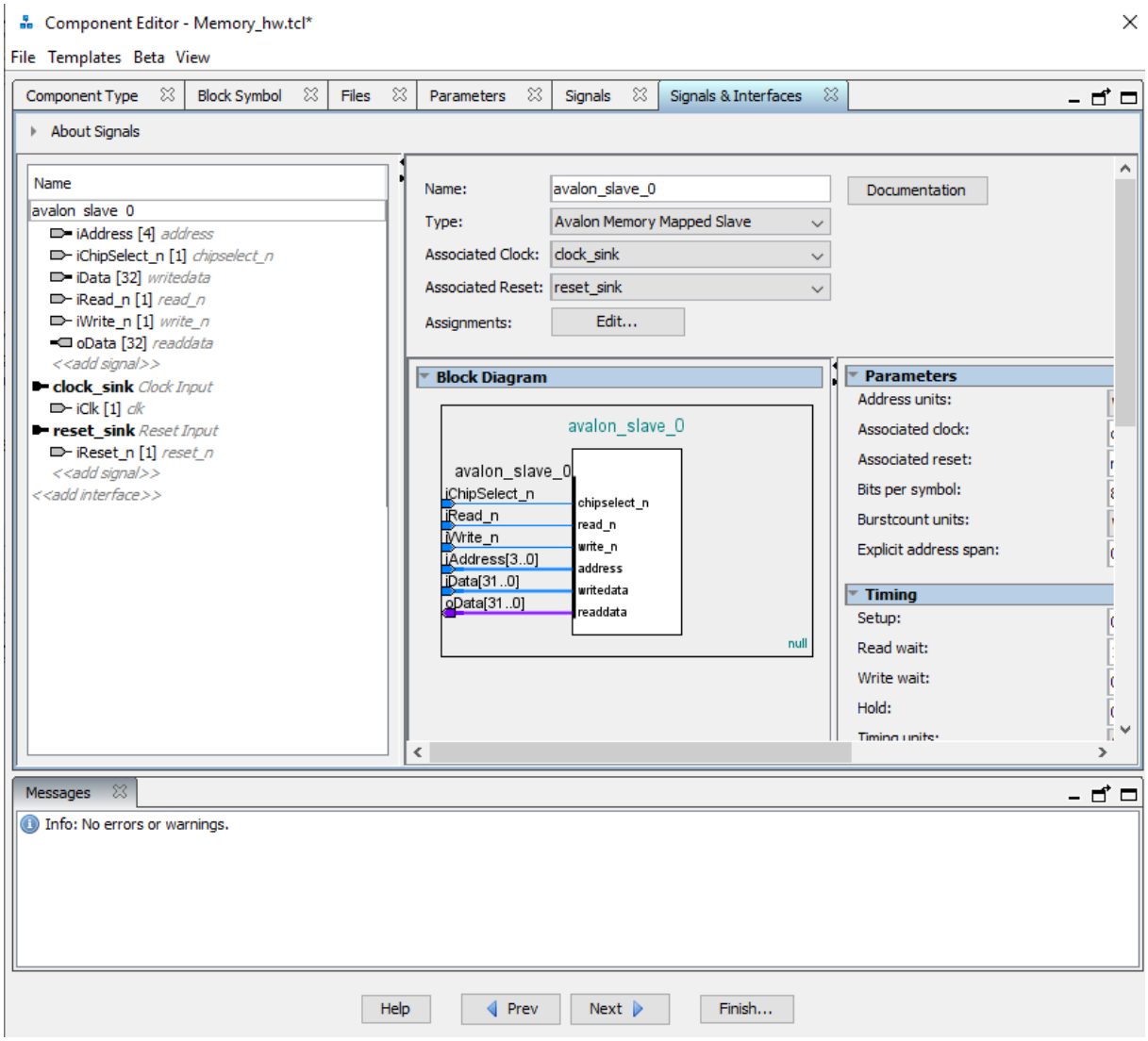
Hình 4. Thêm file và phân tích.

- Tiếp tục chuyên qua tab **Signals** và cấu hình như hình 5.



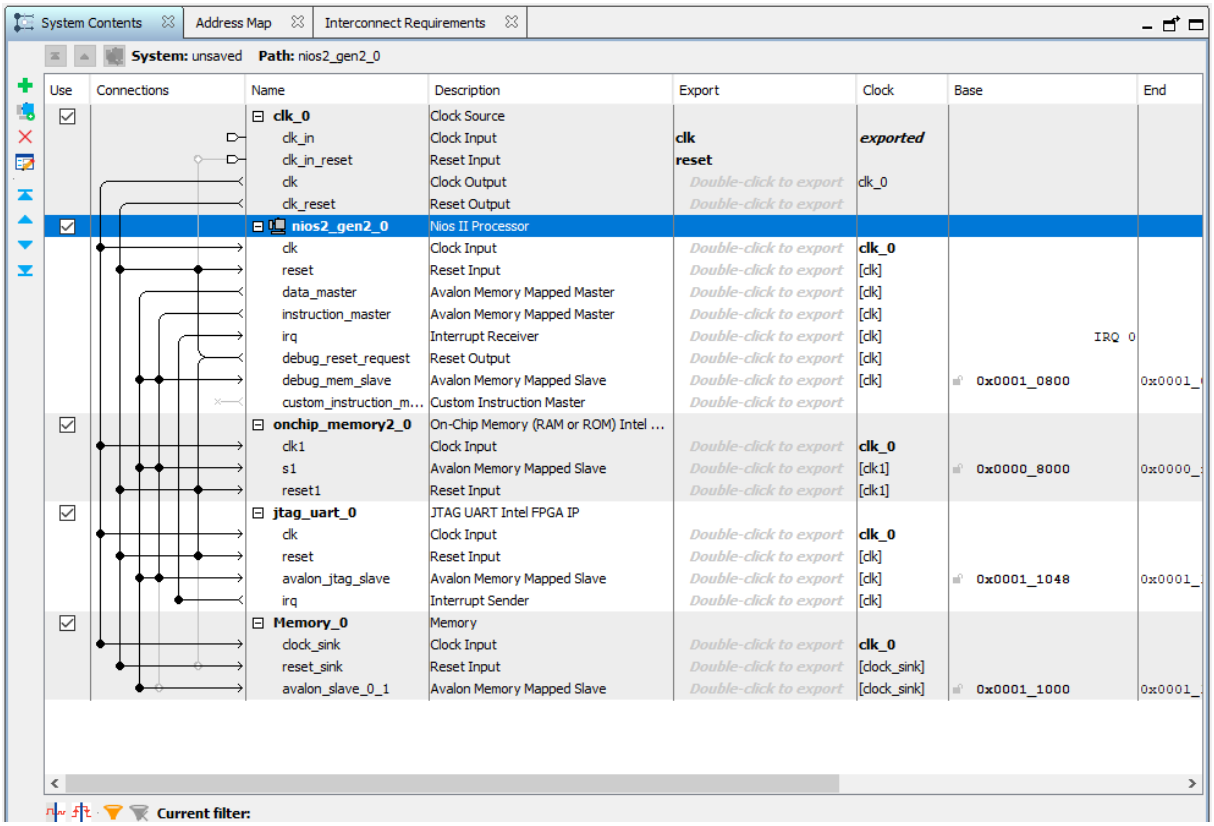
Hình 5. Cấu hình các tín hiệu.

- Cuối cùng cấu hình tín hiệu **clock** và **reset** cho các interface cần thiết, hình 6.



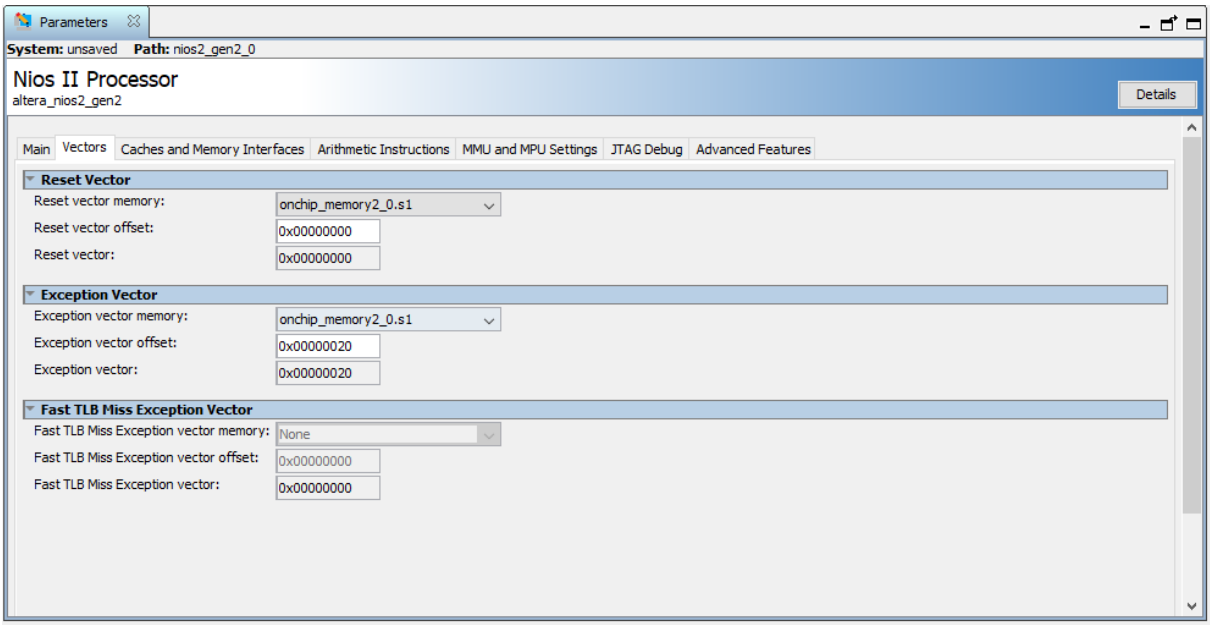
Hình 6. Cấu hình clock và reset cho các interface.

- Cuối cùng, lưu lại mô đun.
- Tạo ra hệ thống và kết nối hoàn chỉnh như hình 7 bên dưới.



Hình 7. Hệ thống hoàn chỉnh.

- Lưu ý, chọn vector cho Nios II là onchip memory như hình 8.



Hình 8. Cấu hình các vector của Nios II.

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

3.4. Tích hợp hệ thống

- Thêm file **system.qip.**
- Tạo file top-level là **Bai5.v** được mô tả như đoạn code bên dưới để tổng hợp hệ thống.

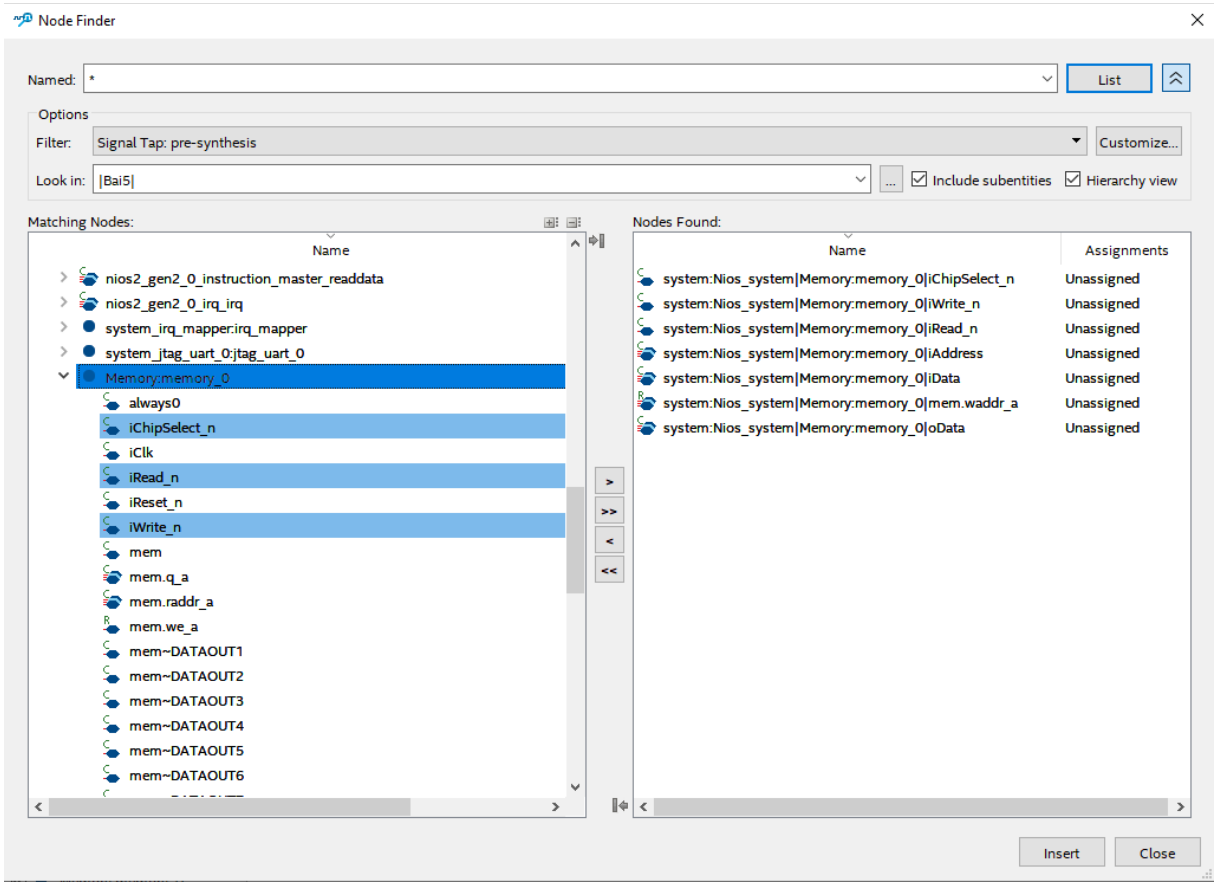
```
module Bai5(
        input               CLOCK_50,
        input       [0:0]   KEY
        );
    system Nios_system (
            .clk_clk            (CLOCK_50),
            .reset_reset_n      (KEY[0])
            );
Endmodule
```

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

3.5. Cấu hình Signal Tab

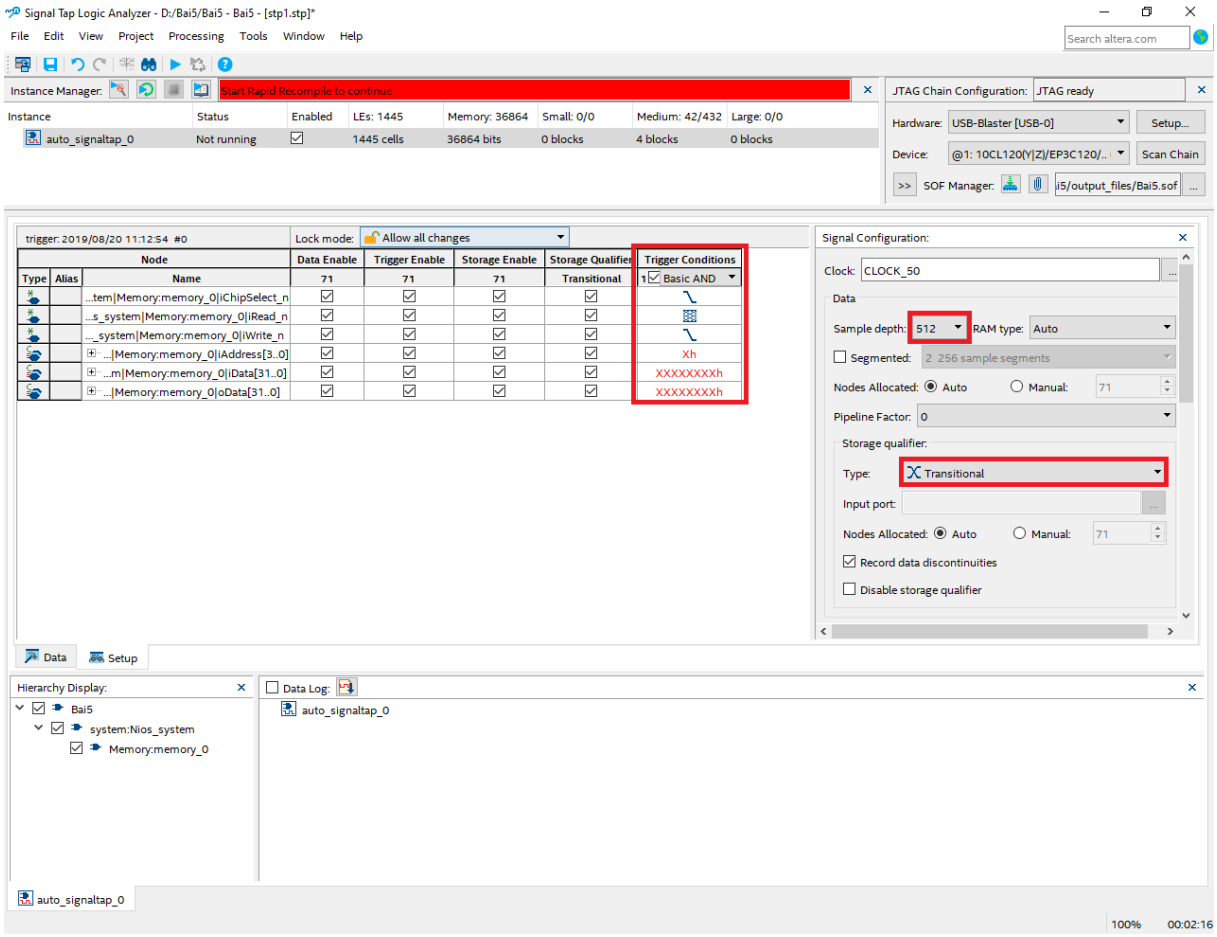
- Sau khi Phân tích và tổng hợp mạch, mở Signal Tab bằng cách chọn **Tools → Signal Tab Logic Analyzer.**
- Đầu tiên, tiến hành cấu hình **clock** để hoạt động là **CLOCK_50** (Bài 4).

  Thêm các tín hiệu cần lưu như hình 9.



Hình 9. Thêm các tín hiệu trên Signal Tab.

- Cấu hình điều kiện bắt tín hiệu như hình 10.



Hình 10. Cấu hình điều kiện bắt tín hiệu.

- Tiến hành biên dịch và nạp phần cứng xuống board.

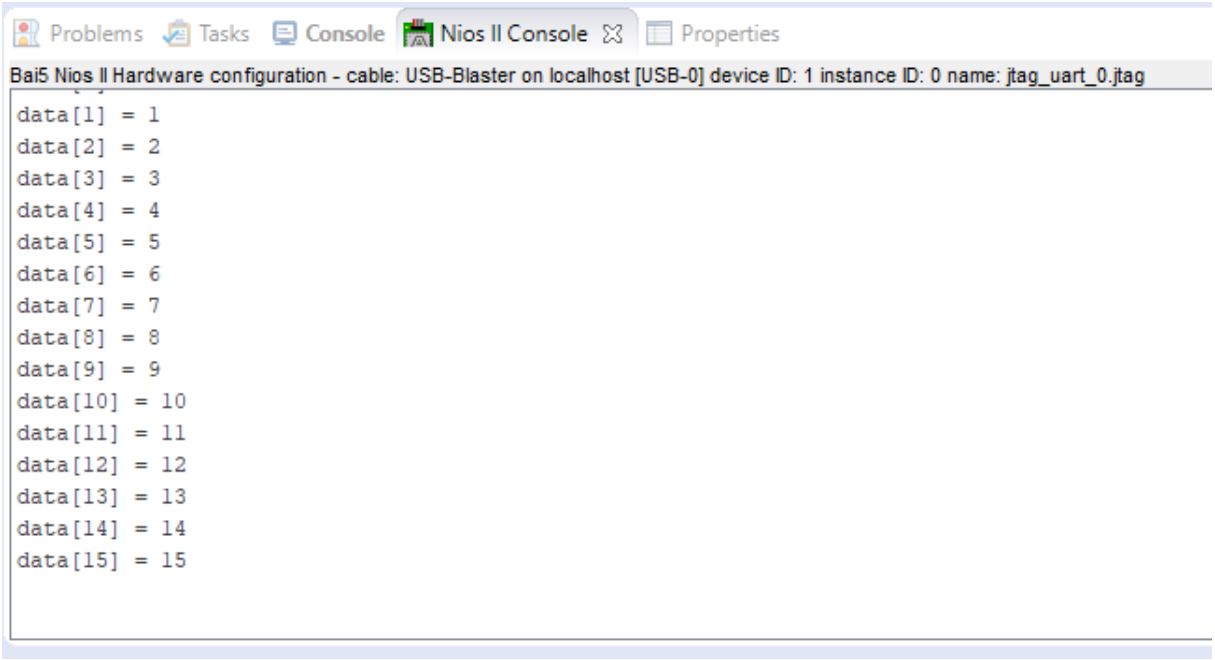
3.6. Xây dựng phần mềm

- Tạo và đặt tên project là "**Bai5**".
- Thêm file "**source.c**" vào project "**Bai5". File "source.c**" có nội dung như đoạn code bên dưới.

```
#include <stdio.h>
#include "io.h"
#include "system.h"
void main() {
	int data;
	char i;
	for (i = 0; i < 16; i++) {
		IOWR(MEMORY_0_BASE, i, i);
	}
	for (i = 0; i < 16; i++) {
		data = IORD(MEMORY_0_BASE, i);
		printf("data[%d] = %d\n", i, data);
	}
```

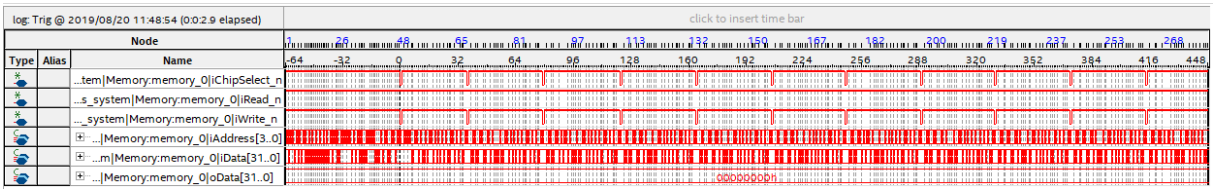
```
}
```

- Build project và dowload phần mềm xuống board.



Hình 11. Kết quả trên console.

- Quan sát kết quả trên Signal Tab, hình 11.



Hình 12. Kết quả trên Signal Tab.

BÀI TẬP CHUẨN BỊ Ở NHÀ

Bài 1. Dựa vào [1] và [4], bộ nhớ SRAM được kết nối với chip Cyclone IV như hình bên dưới:

Hãy viết code verilog mô tả mô đun SRAM_Controler mô tả hoạt động tương tác với avalon bus và với chip SRAM trên board DE2-115.

BÁO CÁO THỰC HÀNH

Bài 1. Thực hiện hệ thống như bài hướng dẫn thực hành, sửa code C để truy xuất mô đun Memory thông qua con trỏ.

Bài 2. Tiến hành xây dựng hệ thống SoC tương tác với mô đun SRAM_Controller như đã 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.

[5] SignalTap II with Verilog Design.