THỰC HÀNH SOC

Bài 4 – GIAO TIẾP BỘ NHỚ TỰ THIẾT KẾ

1. MỤC ĐÍCH

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

o Thiết kế một bộ nhớ đơn giản bằng code Verilog.

o Cách xây dựng hệ thống phần cứng bằng Qsys để giao tiếp với bộ nhớ.

o Cách xây dựng chương trình C trên công cụ Nios II – EDS để giao tiếp với bộ nhớ.

2. NỘI DUNG

2.1. Hệ thống phần cứng

2.1.1. Tạo project Quartus

Tạo project Quartus tên là "lab4". Lưu ý đường dẫn thư mục project không được có khoảng trắng.

Nếu sử dụng board DE2-115, chọn Family là Cyclone IV E, device là EP4CE115F29C8.

Nếu sử dụng board DE2, chọn Family là Cyclone II, device là EP2C35F672C6.

Các tín hiệu cơ bản của một bộ nhớ đơn giản gồm có:

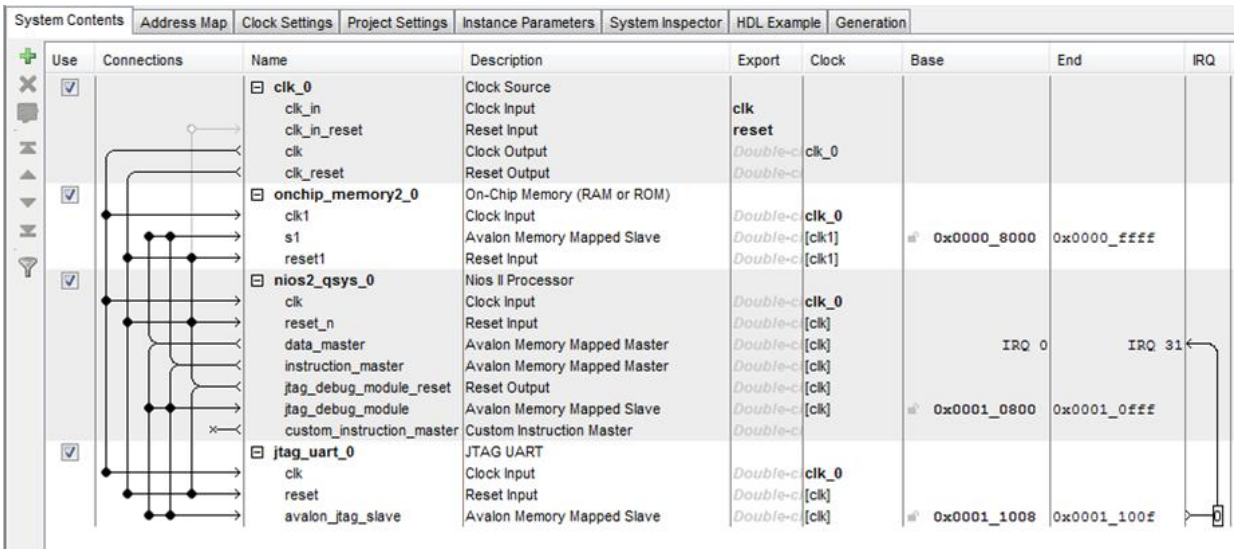
STT	*Tên tín hiệu*	*Vào/Ra*	*Mô tả*
1	iClk	Input	Cấp xung clock cho bộ nhớ hoạt động
2	iRead_n	Input	Cho phép đọc giá trị từ các ô nhớ trong bộ nhớ
3	iWrite_n	Input	Cho phép ghi giá trị vào các ô nhớ trong bộ nhớ
4	iAddress	Input	Chọn địa chỉ của ô nhớ cần đọc/ghi.
5	iData	Input	Dữ liệu cần ghi vào bộ nhớ
6	oData	Output	Dữ liệu đọc ra từ bộ nhớ

Tạo file “memory.v” có nội dung như bên dưới và thêm vào project.

```
module memory
#(parameter DATA_WIDTH=32, parameter ADDR_WIDTH=4)
(
    // Avalon Slave interface
    input iClk,iReset_n,
    input iRead_n, iWrite_n,
    input [(ADDR_WIDTH-1):0] iAddress,
    input [(DATA_WIDTH-1):0] iData,
    output [(DATA_WIDTH-1):0] oData
);

    // Declare the memory variable
    reg [DATA_WIDTH-1:0] mem[2**ADDR_WIDTH-1:0];

    // Variable to hold the registered read address
    reg [ADDR_WIDTH-1:0] addr_reg;

    always @ (posedge iClk) begin

        // Write
        if (!iWrite_n) begin
            mem[iAddress] <= iData;
        end

        //read
        if (!iRead_n) begin
            addr_reg <= iAddress;
        end

    end

    assign oData = mem[addr_reg];

endmodule
```

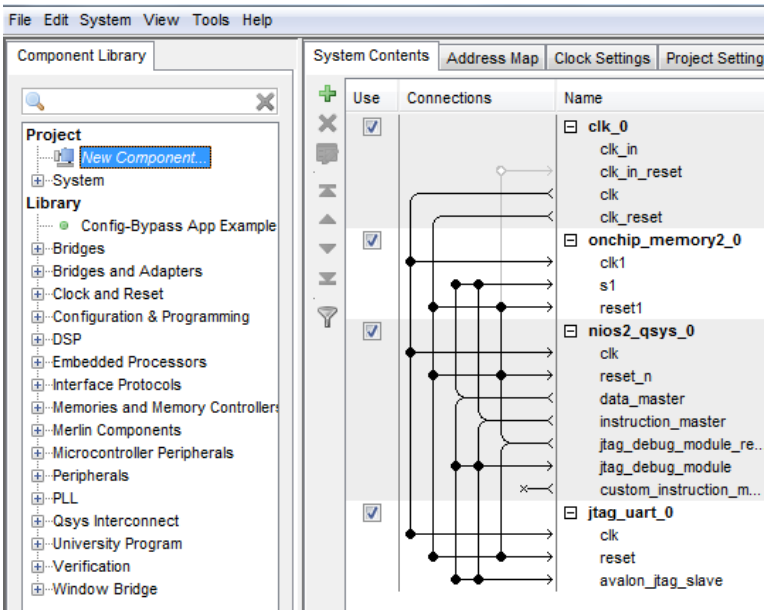
2.1.2. Xây dựng hệ thống Qsys

Xây dựng hệ thống phần cứng như hình bên dưới.

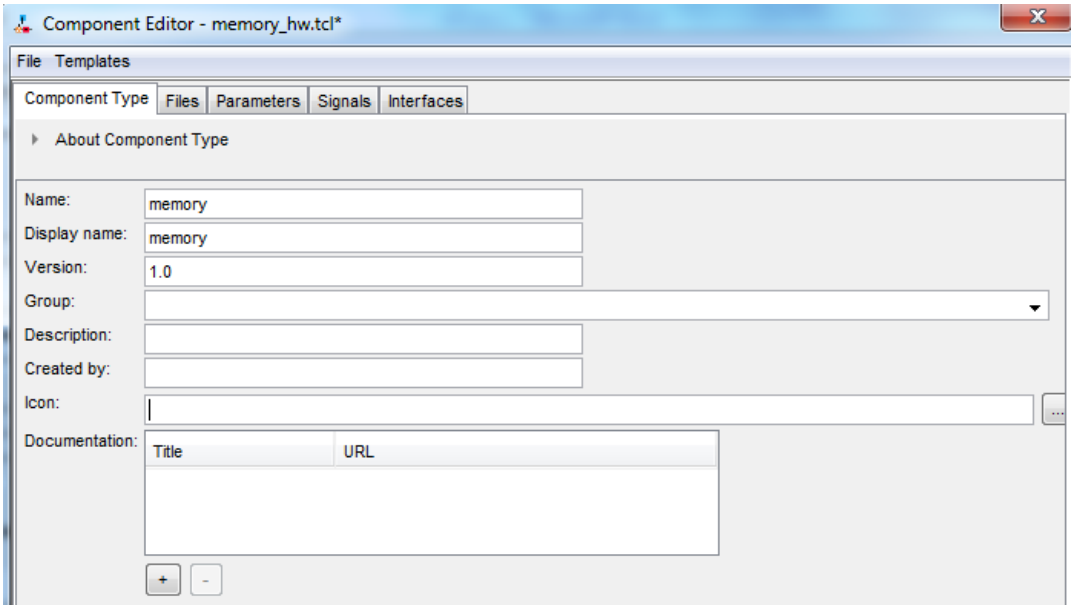


Tiếp theo, chúng ta sẽ thêm module “memory.v” đã thiết kế ở mục 2.1.1 vào thư viện của Qsys.

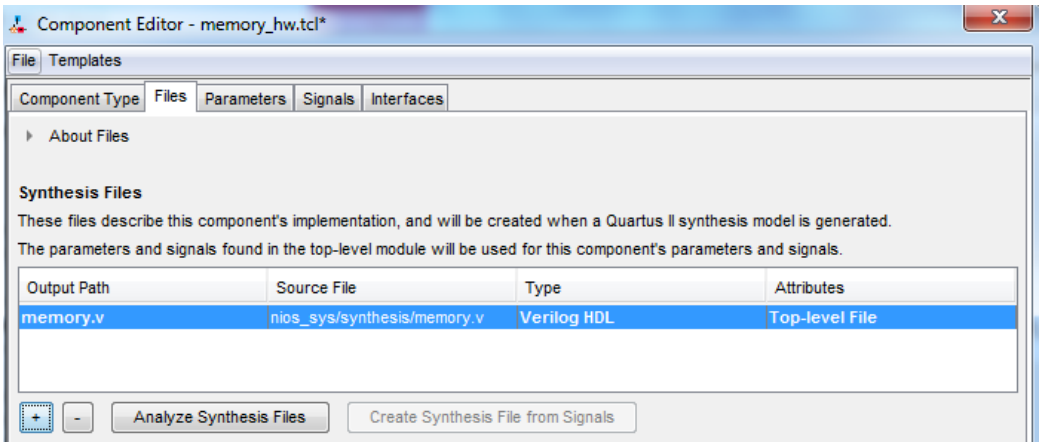
Double click vào “New component” ở cửa sổ thư viện.



Đặt tên cho module ở tab “Component Type” như bên dưới.

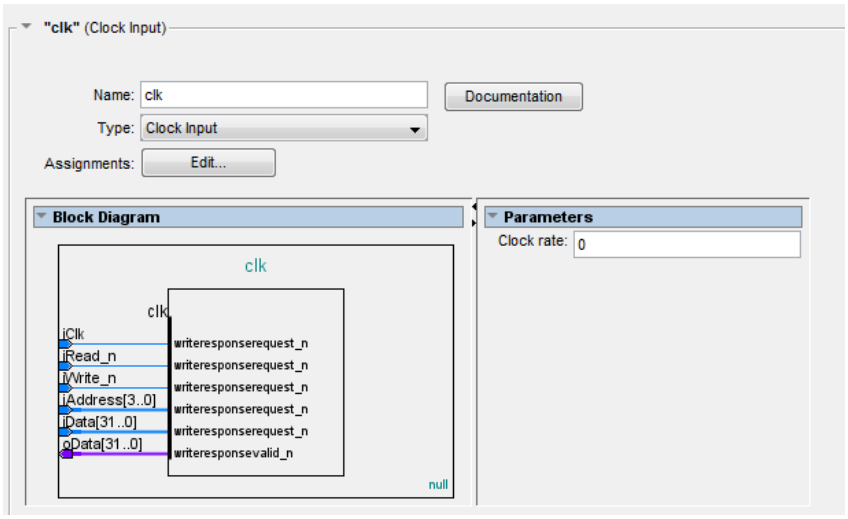


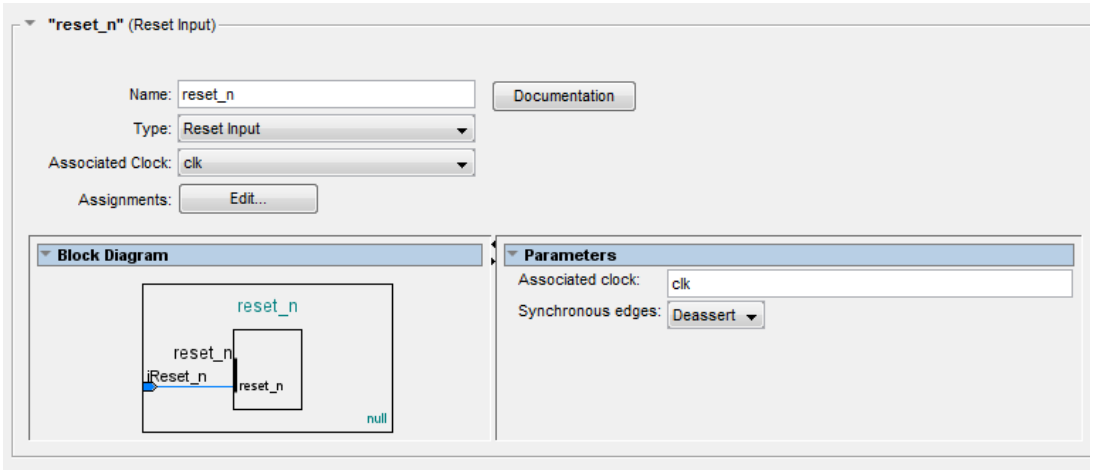
Chuyển sang tab “Files”, thêm file “memory.v” đã thiết kế ở mục 2.1.1.



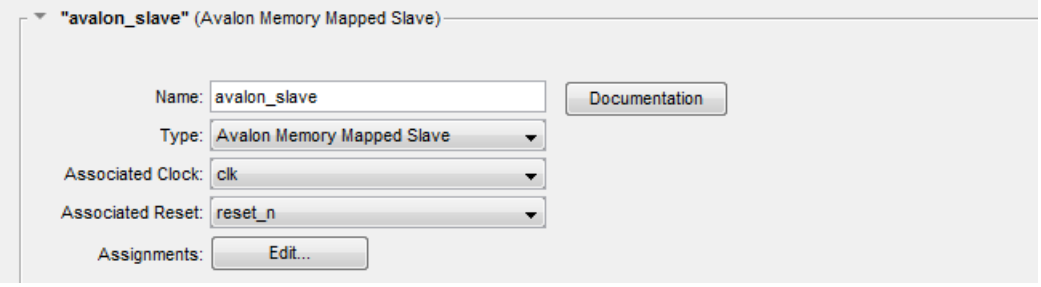
Click vào “Analyze Synthesis Files”.

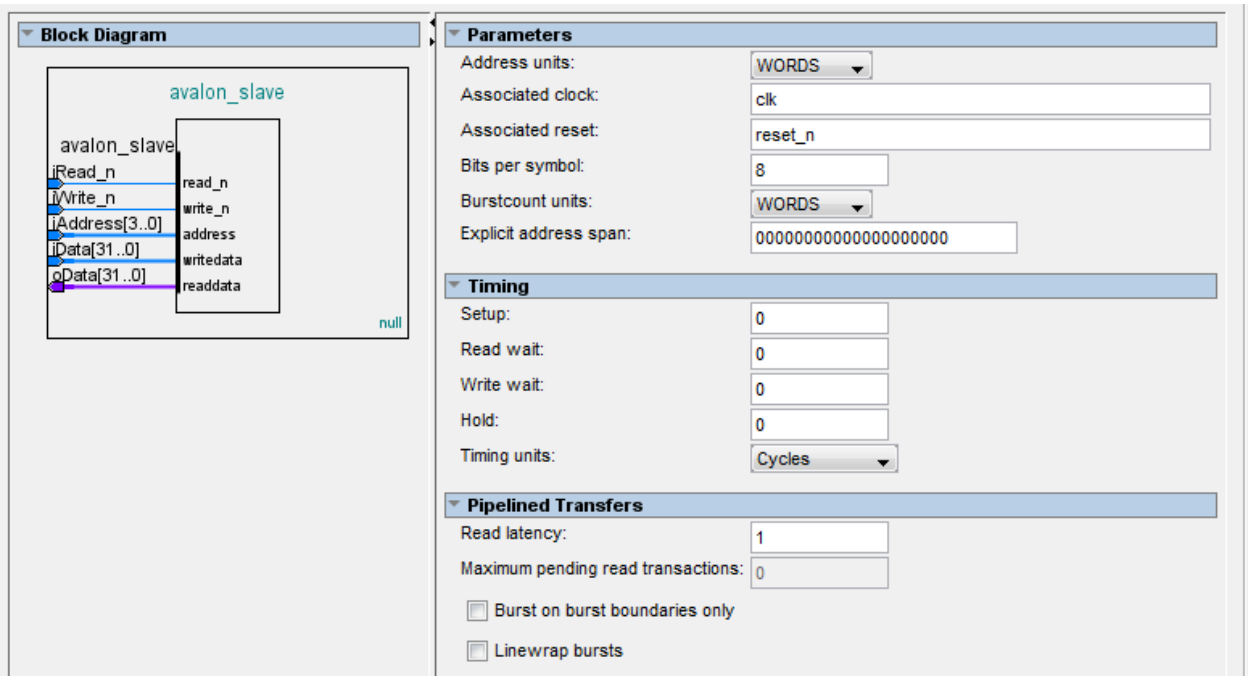
Chuyển qua tab “Interfaces”, thiết lập interface “clk” và “reset_n” như bên dưới.





Thêm interface “avalon_slave”. Thiết lập interface “avalon_slave” như bên dưới.

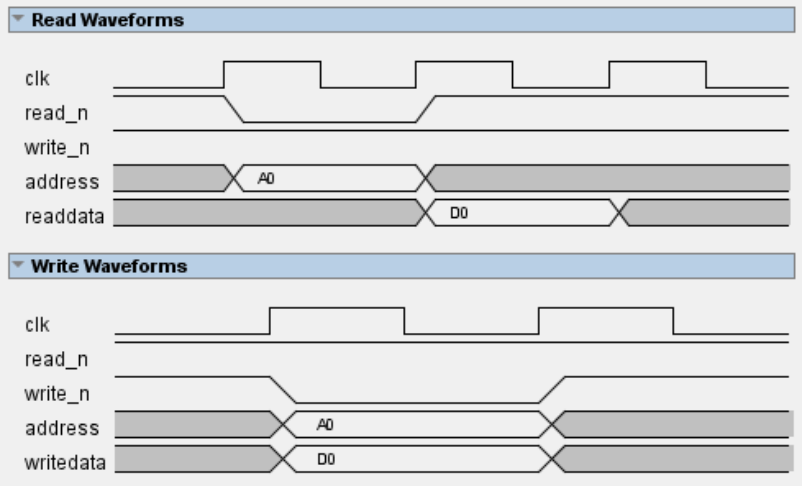




Chuyển qua tab “Signals”, gán các tín hiệu vào các interface tương ứng.

Name	Interface	Signal Type	Width	Direction
iClk	**clk**	**clk**	1	input
iRead_n	**avalon_slave**	**read_n**	1	input
iWrite_n	**avalon_slave**	**write_n**	1	input
iAddress	**avalon_slave**	**address**	ADDR_WIDTH	input
iData	**avalon_slave**	**writedata**	DATA_WIDTH	input
oData	**avalon_slave**	**readdata**	DATA_WIDTH	output
iReset_n	**reset_n**	**reset_n**	1	input

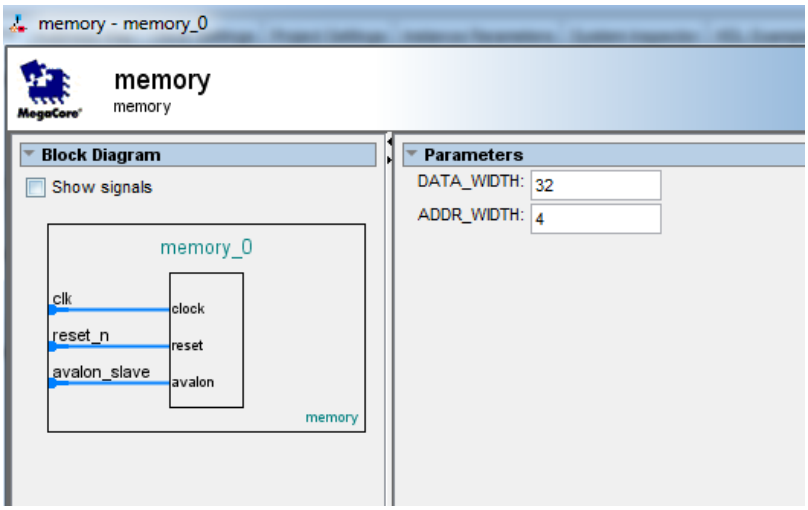
Lúc này, không còn lỗi. Quay trở lại tab “Interface”, xem dạng sóng đọc/ghi dữ liệu của interface “avalon_slave”.



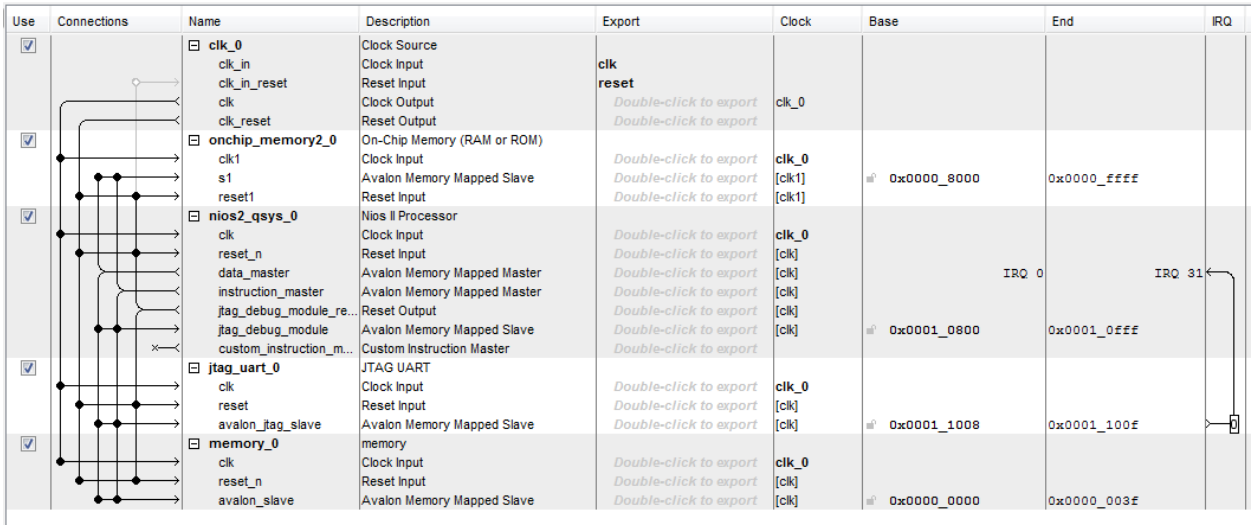
Dạng sóng đọc/ghi dữ liệu này phù hợp với module “memory.v” đã thiết kế.

Click “Finish”.

Trong cửa sổ thư viện, chọn module “memory” và click “Add” để thêm module này vào hệ thống Qsys. Thiết lập các thông số như bên dưới. Click “Finish”.



Kết nối các tín hiệu của module “memory_0” mới thêm vào hệ thống.



Gán địa chỉ cho các module (System > Assign Base Addresses).

Hệ thống phần cứng đã hoàn thành, không còn thông báo lỗi. Save lại hệ thống với tên “nios_sys”.

Chuyển sang tab “Generation”, click “Generate”.

2.1.3. Tích hợp hệ thống Qsys vào project Quatus

Thực hiện tương tự như bài thực hành trước.

Tạo file top module, đặt tên là “lab4.v” với nội dung như sau.

```
1   module lab4(CLOCK_50,KEY);
2
3   input CLOCK_50;
4   input [0:0] KEY;
5
6   nios_sys u0 (
7       .clk_clk       (CLOCK_50),
8       .reset_reset_n (KEY[0])
9    );
10
11
12  endmodule
```

Build project Quartus và download hệ thống phần cứng xuống board.

2.2. Lập trình phần mềm

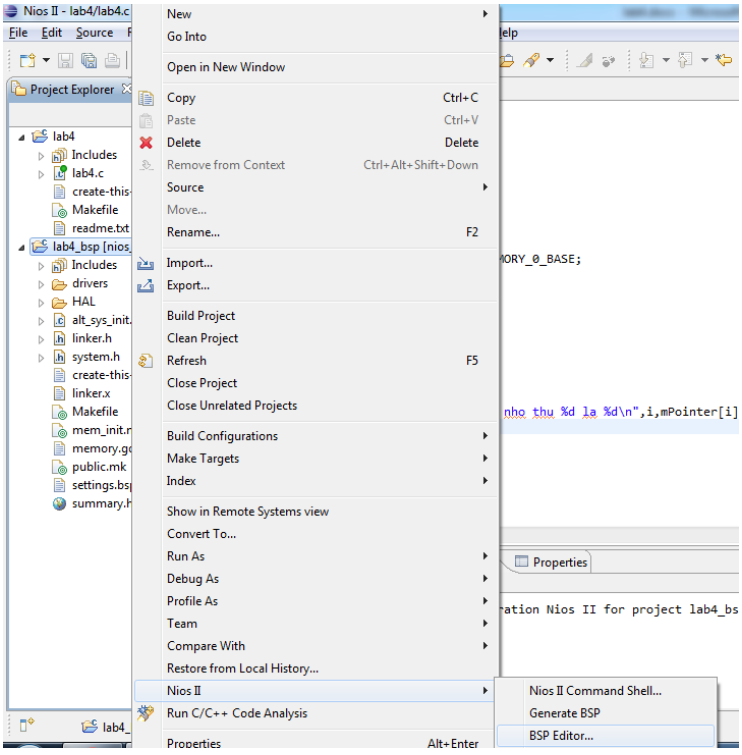
Tạo và đặt tên project là “lab4”.

Thêm file “lab4.c” vào project “lab4”. File “lab4.c” có nội dung như bên dưới.

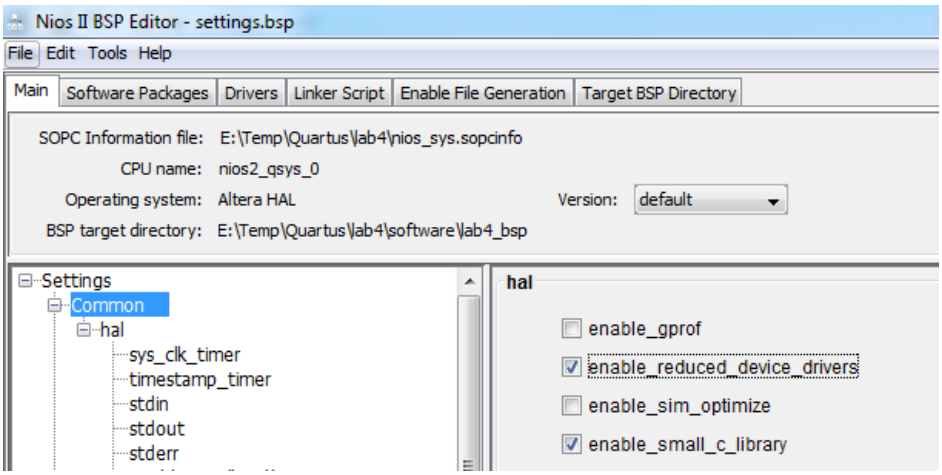
```
#include "system.h"
#include "stdio.h"

int main(void){

    int* mPointer = (int*) MEMORY_0_BASE;
    int i;

    // Ghi du lieu vao bo nho
    for(i=0;i<16;i++){
        mPointer[i] = i;
    }

    // Doc du lieu tu bo nho
    for(i=0;i<16;i++){
        printf("Du lieu tai o nho thu %d la %d\n",i,mPointer[i]);
    }

    while(1);
}
```

Dùng hàm “printf” sẽ in dữ liệu ra cửa sổ console. Khi sử dụng hàm này thì khối lượng thư viện đi kèm là khá lớn, sẽ vượt quá bộ nhớ 32KB của hệ thống. Thực hiện các thiết lập sau để giảm dung lượng thư viện.

Click chuột phải vào project “lab4_bsp”. Chọn “NiosII > BSP Editor”.

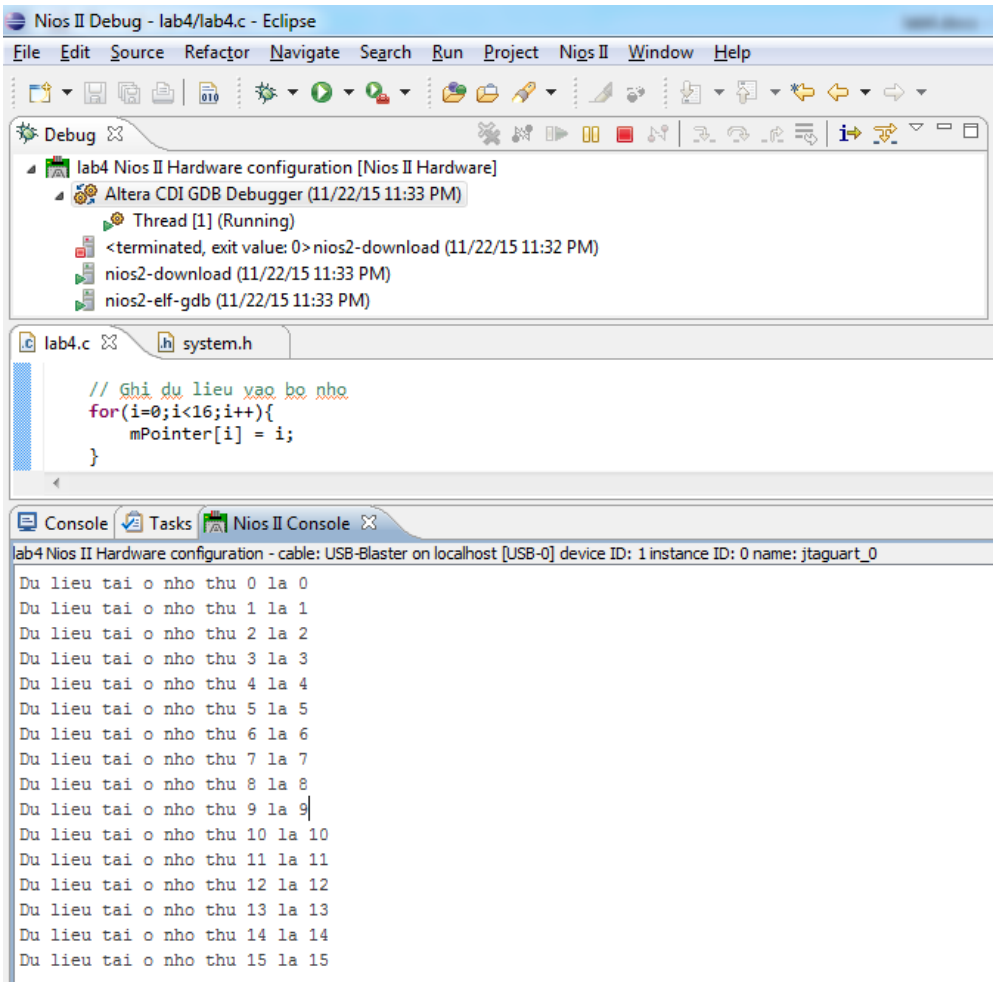


Tại tab “Main”, chọn “enable_reduced_device_drivers” và “enable_small_c_library”.



Click “Generate”, click “Finish”.

Build và download phần mềm xuống board. Kiểm tra giá trị in ra màn hình console.



BÁO CÁO THỰC HÀNH

Bài 4 – GIAO TIẾP BỘ NHỚ TỰ THIẾT KẾ

Sinh viên:

……...……

Lớp: …… Nhóm: ……..……..……..

Bài 1:

Lần lượt thay đổi khai báo con trỏ trong chương trình ở trang 8

int* mPointer = (int*) MEMORY_0_BASE;

thành các khai báo sau:

a/ char* mPointer = (char*) MEMORY_0_BASE;

b/ short* mPointer = (char*) MEMORY_0_BASE;

Chạy chương trình và giải thích kết quả in trên cửa sổ console.