Lab 6: VGA

I. Mục tiêu:

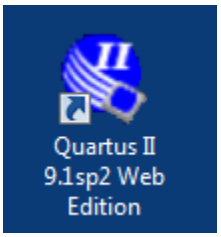
Thiết kế hệ thống với Nios II Processor thực hiện công việc sau:

Vẽ một hộp màu xanh lên màn hình hiển thị VGA. Đồng thời viết hàng chữ DHBK-HCM vào trong hộp màu xanh này

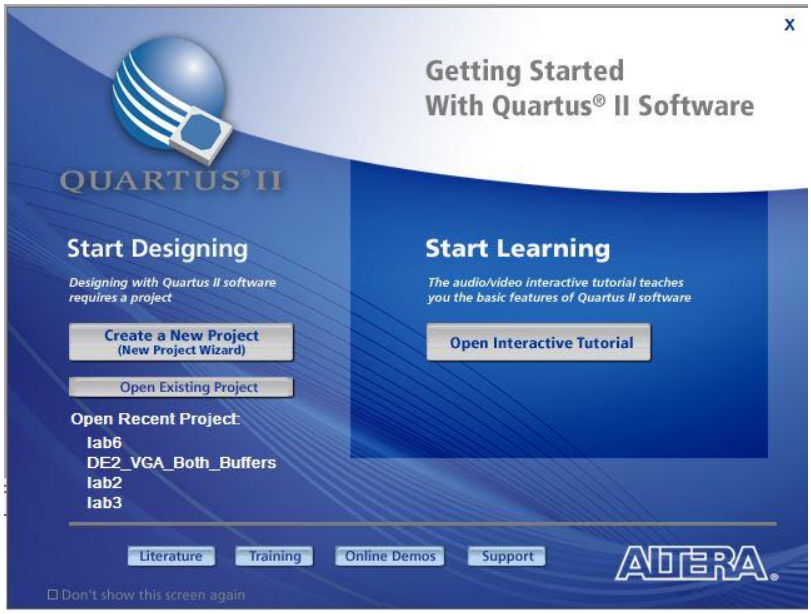
II. Tạo New Project Quartus II:

Thực hiện theo thứ tự các bước sau:

1. Tạo 1 file mới New folder với tên **lab6**
2. Mở Quartus II.



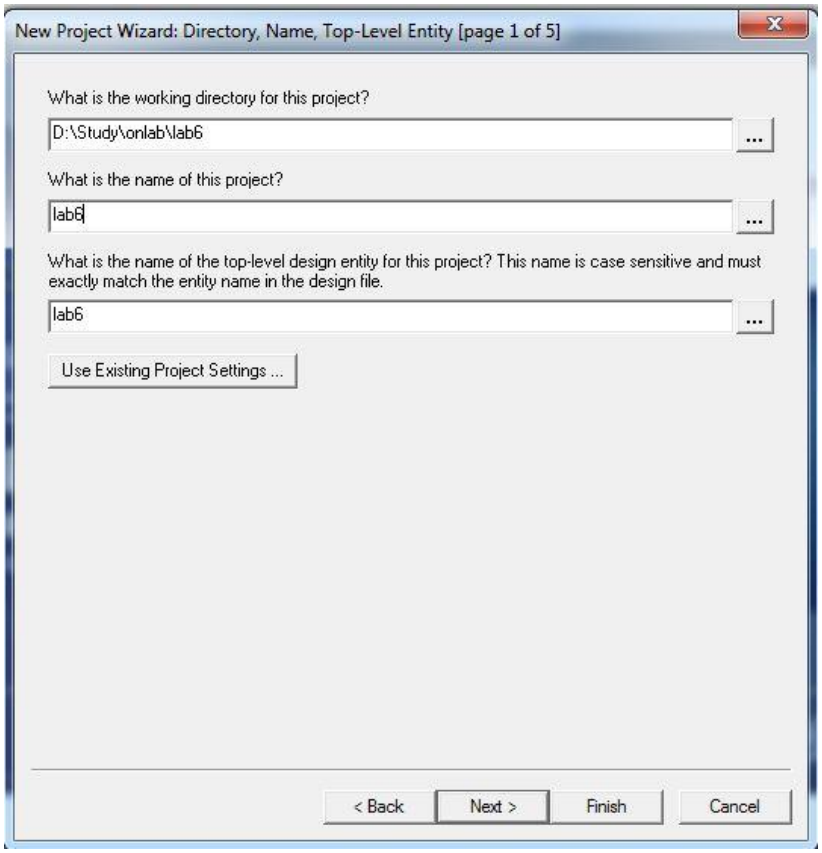
3. Trên Quartus II menu bar chọn File -> New Project Wizard.



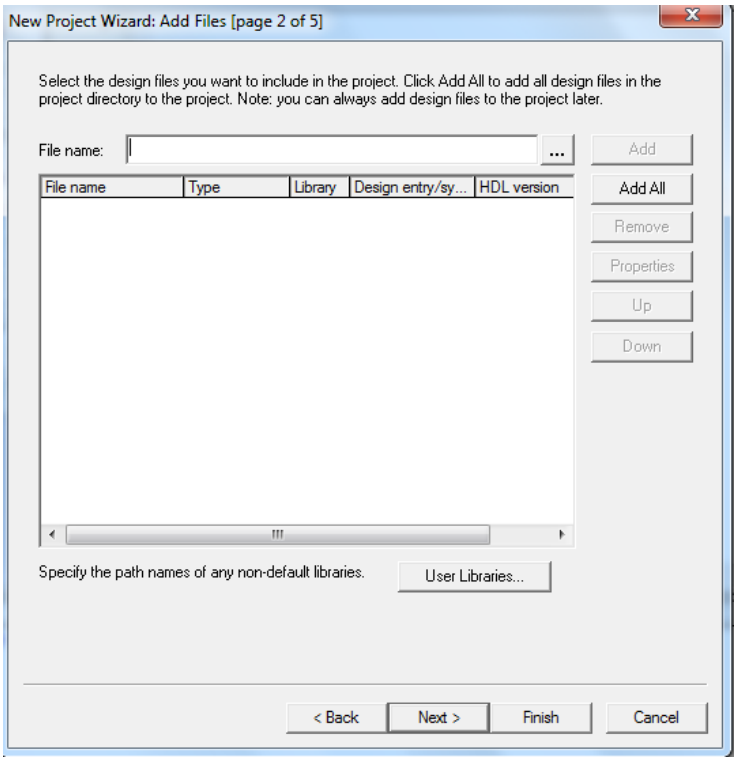
4. Trong khung thứ nhất chọn đường dẫn vào thư mục vừa tạo mang tên **lab6.**

Tên project phải trùng với tên thư mục là **lab6.**

Click Next



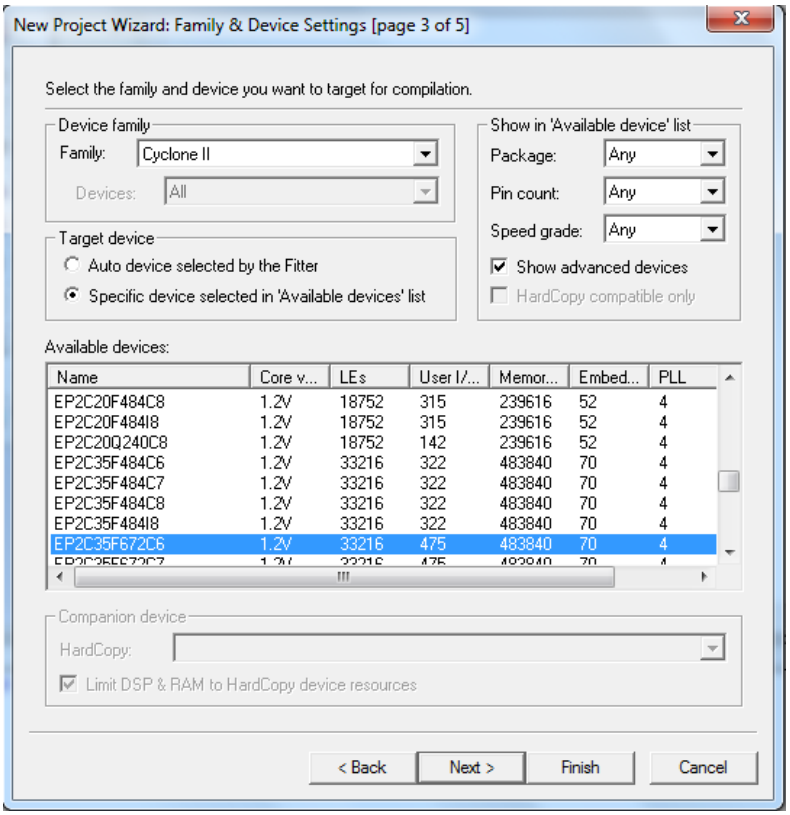
5. Click Next



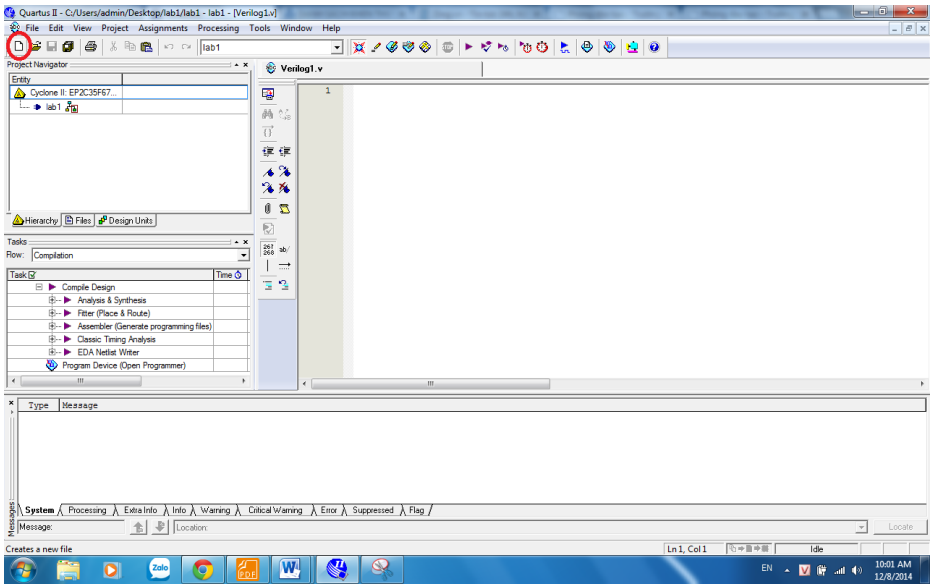
6. Chọn Cyclone II.

Available devices: **Chọn EP2C35F672C6.**

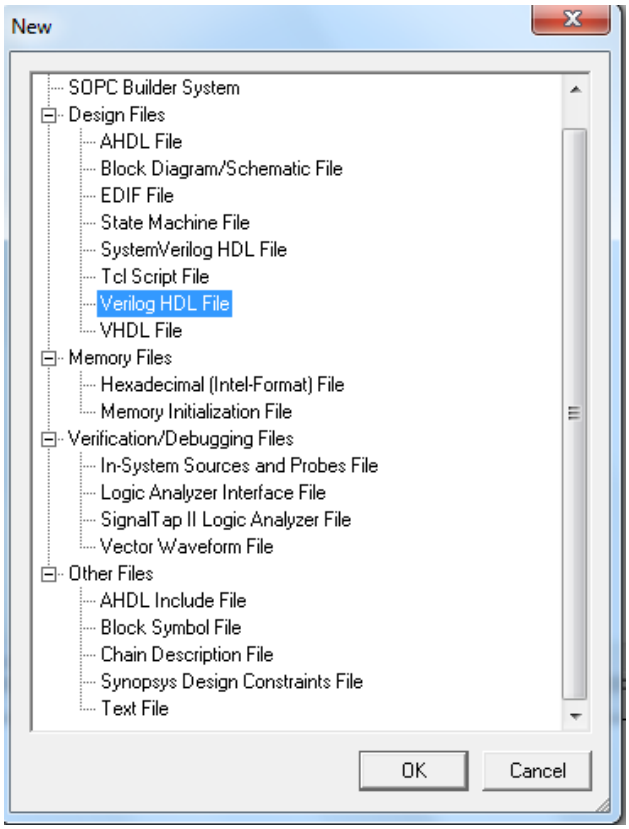
Click Next



7. Click Finish.
8. Click New

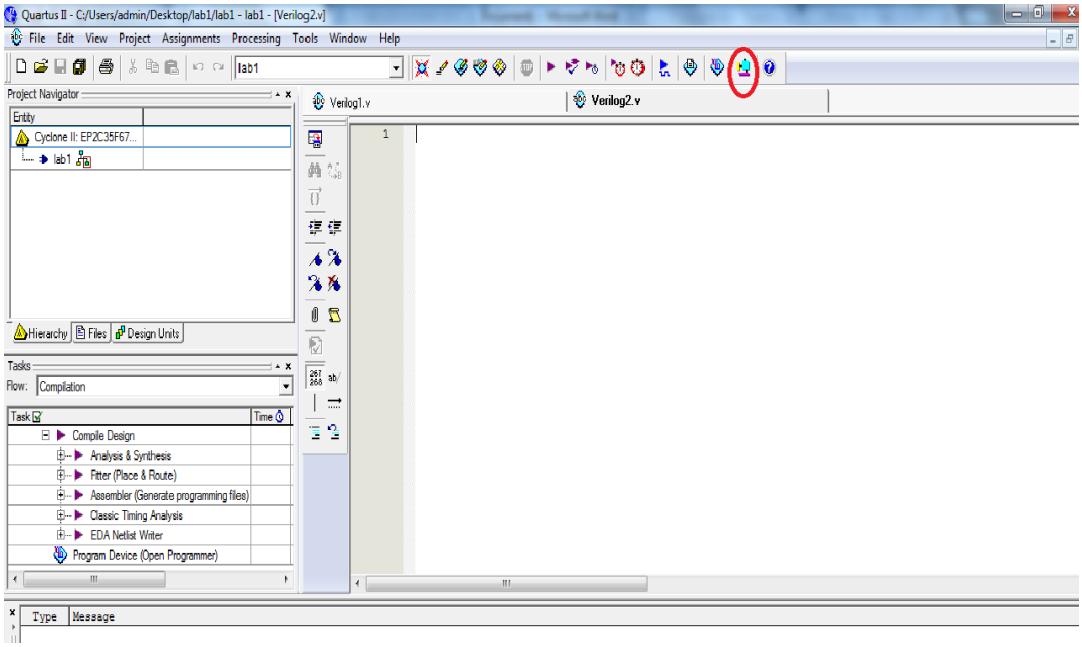


9. Chọn **Verilog HDL File -> click OK**

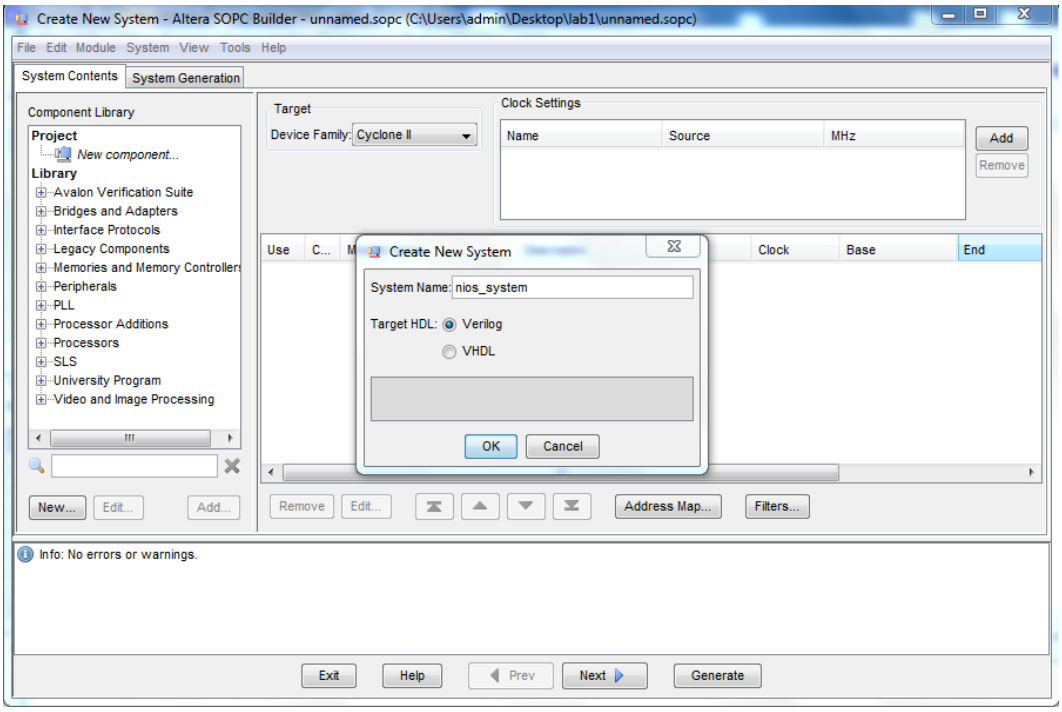


III. TẠO SOPC:

1. Click **SOPC Builder** để tạo file SOPC.



2. System name: **nios_system** -> Click **OK.**
   Target HDL: **Verilog**
   Sau đó chọn : **OK**



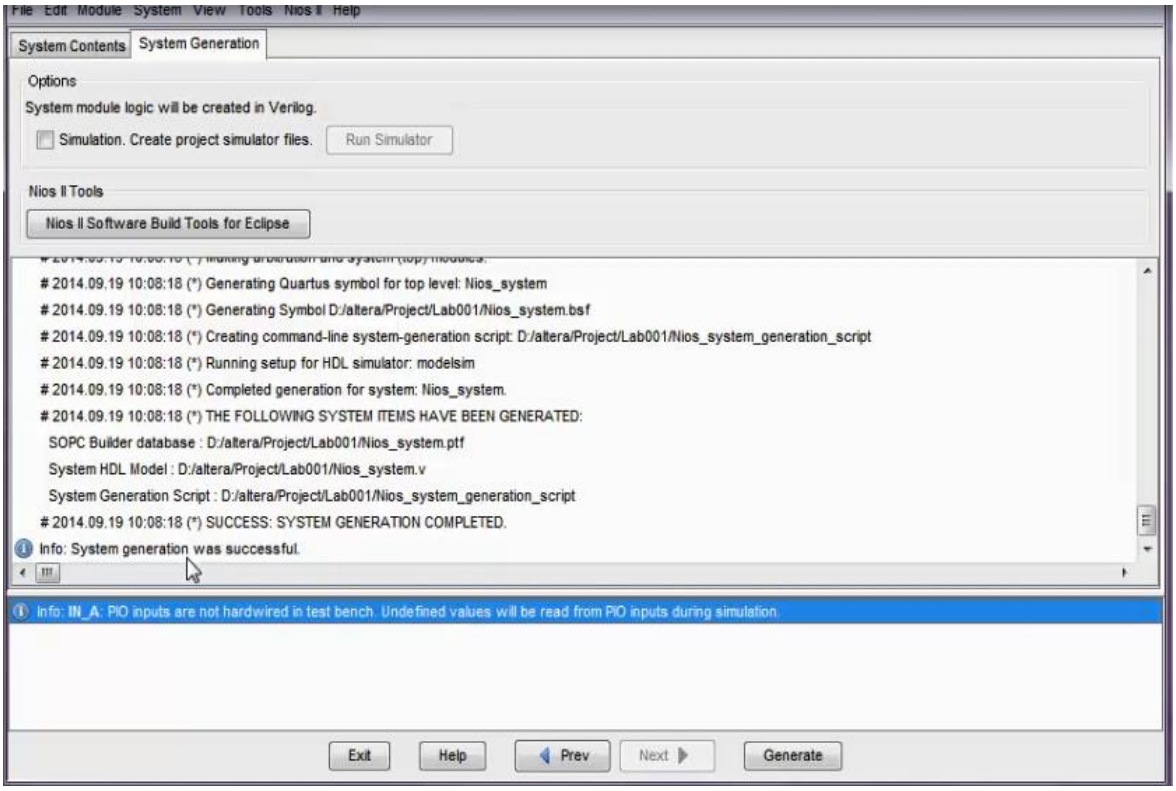
3. Building **SoPC:**
   - **Processors -> Nios II Processor.**
   - **Memories and Memory Control -> On-Chip Memory (RAM or ROM):** Chọn RAM, memory size 16 Kbytes.
   - **University Program -> Clocks Signals for DE-Series Board Periherals:** Chọn Video.
   - **University Program -> Audio & Video -> Video -> Character Buffer for VGA Display:** Video-Out Device chọn On-board VGA DAC, chọn enable Transparency
   - **University Program -> Memory -> SRAM/SSRAM Controller:** Chọn DE2, Use a pixel buffer for video out.
   - **University Program -> Audio & Video -> Video ->Pixel Buffer DMA Controller:** Chọn Width 320, Height 240, Color Space 16-bit RGB.
   - **University Program -> Audio & Video -> Video ->RGB Resampler:** Chọn Incoming Format 16-bit RGB, Outgoing Format 30-bit RGB.
   - **University Program -> Audio & Video -> Video -> Scaler:**
     Width Scaling Factor: 2, Height Scaling Factor: 2, Width: 320, Height: 240, Color Bits: 10, Color Planes 3.
   - **University Program -> Audio & Video -> Video -> Alpha Blender:** mode: Simple.

- **University Program -> Audio & Video -> Video-> Dual-Clock FIFO:** Color Bits: 10, Color Planes: 3.
- **University Program -> Audio & Video -> Video->VGA Controller:** Chọn DE2, Video Out Device VGA Connector.

  ⇨ Đây là kết quả.

Use	C...	Module Name	Description	Clock	Base	End
☑		⊞ CPU	Nios II Processor	sys_clk	0x0008a800	0x0008afff
☑		⊞ Onchip_Memory	On-Chip Memory (RAM or ROM)	sys_clk	0x00084000	0x00087fff
☑		⊞ Clock_Signals	Clocks Signals for DE-Series Board Pe...	clk_0	0x0008b018	0x0008b019
☑		⊟ Char_Buffer_with_D...	Character Buffer for VGA Display			
	→	avalon_char_control_...	Avalon Memory Mapped Slave	sys_clk	0x0008b010	0x0008b017
	→	avalon_char_buffer_s...	Avalon Memory Mapped Slave		0x00088000	0x00089fff
		avalon_char_source	Avalon Streaming Source			
☑		⊞ Pixel_Buffer	SRAM/SSRAM Controller	sys_clk	0x00000000	0x0007ffff
☑		⊞ Pixel_Buffer_DMA	Pixel Buffer DMA Controller	sys_clk	0x0008b000	0x0008b00f
☑		⊞ Pixel_RGB_Resampler	RGB Resampler	sys_clk		
☑		⊞ Pixel_Scaler	Scaler	sys_clk		
☑	→	⊞ Alpha_Blender	Alpha Blender	sys_clk		
☑		⊞ Dual_Clock_FIFO	Dual-Clock FIFO	*multiple*		
☑		⊞ VGA_Controller	VGA Controller	vga_clk		

9. Chọn **Generate.** Nếu **system generation was successful**, save lại và tắt SOPC builder.



IV. Verilog Code:

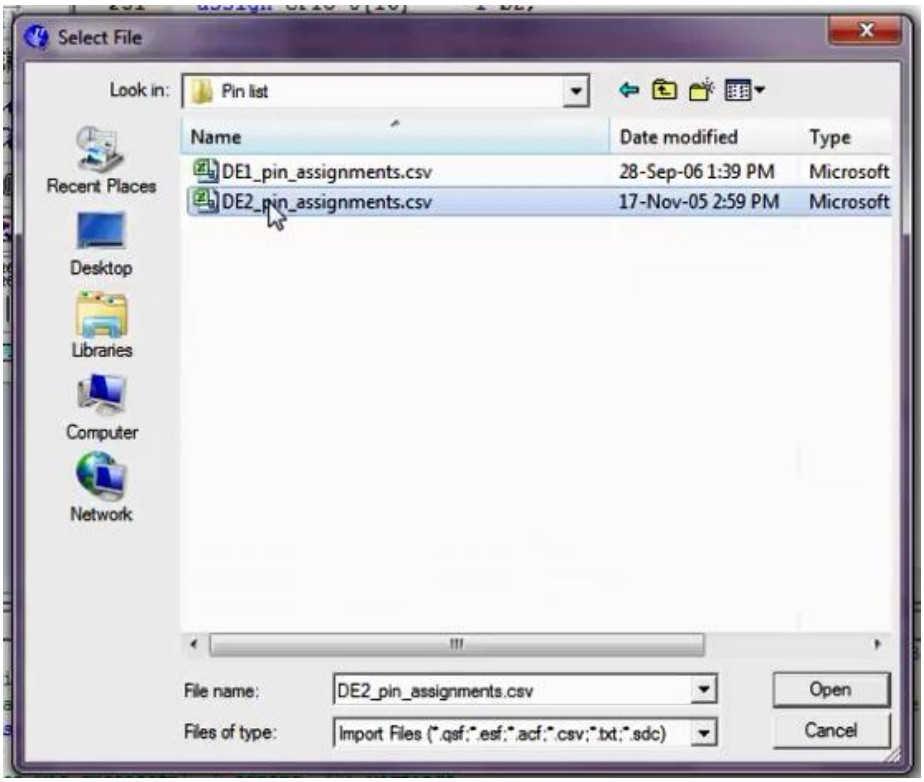
```
module  lab6(
        // Inputs
        CLOCK_50,
        KEY,

        // Bidirectionals
        //  Memory (SRAM)
        SRAM_DQ,

        // Outputs
        //      Memory (SRAM)
        SRAM_ADDR,

        SRAM_CE_N,
        SRAM_WE_N,
        SRAM_OE_N,
        SRAM_UB_N,
        SRAM_LB_N,

        //  VGA
        VGA_CLK,
        VGA_HS,
        VGA_VS,
        VGA_BLANK,
        VGA_SYNC,
        VGA_R,
        VGA_G,
        VGA_B
);

/*****************************************************************************
 *                    Parameter Declarations                    *
 *****************************************************************************/


/*****************************************************************************
 *                     Port Declarations                     *
 *****************************************************************************/
// Inputs
input                           CLOCK_50;
input           [3:0]     KEY;

// Bidirectionals
```

```
//      Memory (SRAM)
inout           [15:0]  SRAM_DQ;

// Outputs
//      Memory (SRAM)
output          [17:0]  SRAM_ADDR;

output                  SRAM_CE_N;
output                  SRAM_WE_N;
output                  SRAM_OE_N;
output                  SRAM_UB_N;
output                  SRAM_LB_N;

//  VGA
output                  VGA_CLK;
output                  VGA_HS;
output                  VGA_VS;
output                  VGA_BLANK;
output                  VGA_SYNC;
output          [ 9: 0] VGA_R;
output          [ 9: 0] VGA_G;
output          [ 9: 0] VGA_B;

/*****************************************************************************
 *          Internal Wires and Registers Declarations           *
 *****************************************************************************/
// Internal Wires

// Internal Registers

// State Machine Registers

/*****************************************************************************
 *                Finite State Machine(s)                  *
 *****************************************************************************/


/*****************************************************************************
 *                  Sequential Logic                    *
 *****************************************************************************/


/*****************************************************************************
 *                  Combinational Logic                 *
 *****************************************************************************/


/*****************************************************************************
```

```
*                    Internal Modules                    *
*****************************************************************************/

        nios_system Nios_II (
        // 1) global signals:
        .clk_0                                                          (CLOCK_50),
        .reset_n                                                        (KEY[0]),
        .sys_clk                                                ),
        .vga_clk                                                ),

        // the_Pixel_Buffer
        .SRAM_DQ_to_and_from_the_Pixel_Buffer       (SRAM_DQ),
        .SRAM_ADDR_from_the_Pixel_Buffer            (SRAM_ADDR),
        .SRAM_LB_N_from_the_Pixel_Buffer            (SRAM_LB_N),
        .SRAM_UB_N_from_the_Pixel_Buffer            (SRAM_UB_N),
        .SRAM_CE_N_from_the_Pixel_Buffer            (SRAM_CE_N),
        .SRAM_OE_N_from_the_Pixel_Buffer            (SRAM_OE_N),
        .SRAM_WE_N_from_the_Pixel_Buffer            (SRAM_WE_N),

        // the_vga_controller
        .VGA_CLK_from_the_VGA_Controller            (VGA_CLK),
        .VGA_HS_from_the_VGA_Controller                     (VGA_HS),
        .VGA_VS_from_the_VGA_Controller                     (VGA_VS),
        .VGA_BLANK_from_the_VGA_Controller                  (VGA_BLANK),
        .VGA_SYNC_from_the_VGA_Controller           (VGA_SYNC),
        .VGA_R_from_the_VGA_Controller                      (VGA_R),
        .VGA_G_from_the_VGA_Controller                      (VGA_G),
        .VGA_B_from_the_VGA_Controller                      (VGA_B)
);

endmodule
```

1.Save lại vào thư mục project của mình.

2.Vào Assignments → Import Assignments →Chọn file DE2_pin_assignments.csv →Open

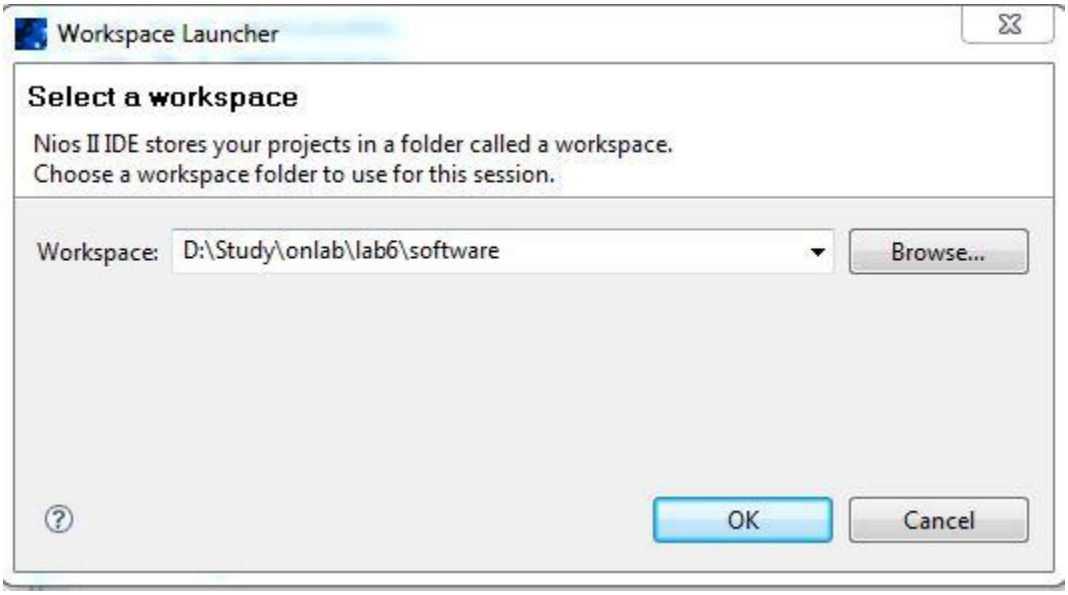


3.Start compile.

V. C code trên NIOS II 9.1 IDE

1. Mở Nios II 9.1 IDE

2. **File-> Switch Workspace**



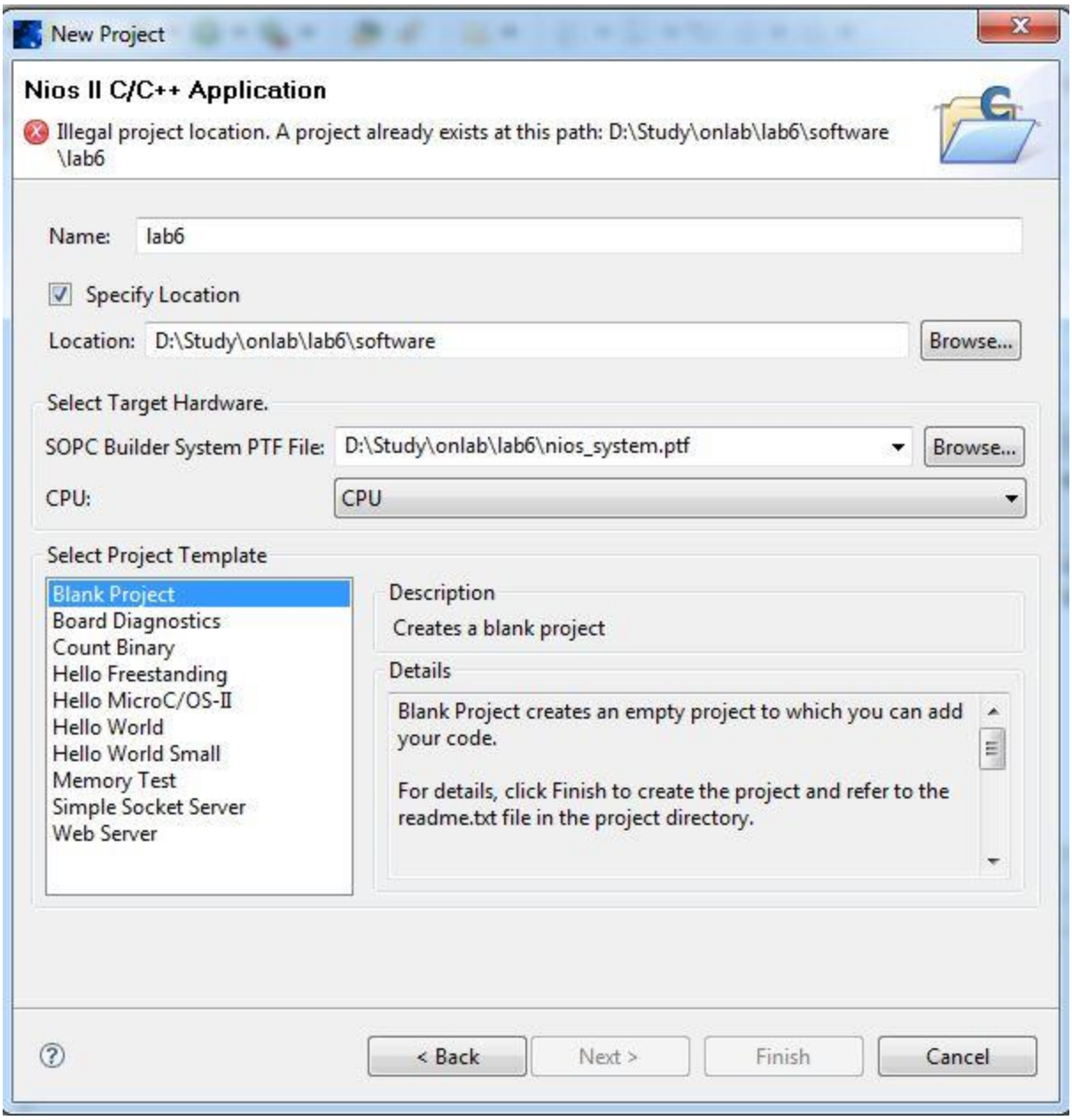
3. Chọn **File → New → Nios II C/C++ Application**

4. Đặt tên cho project.

   Chọn **Blank Project.**

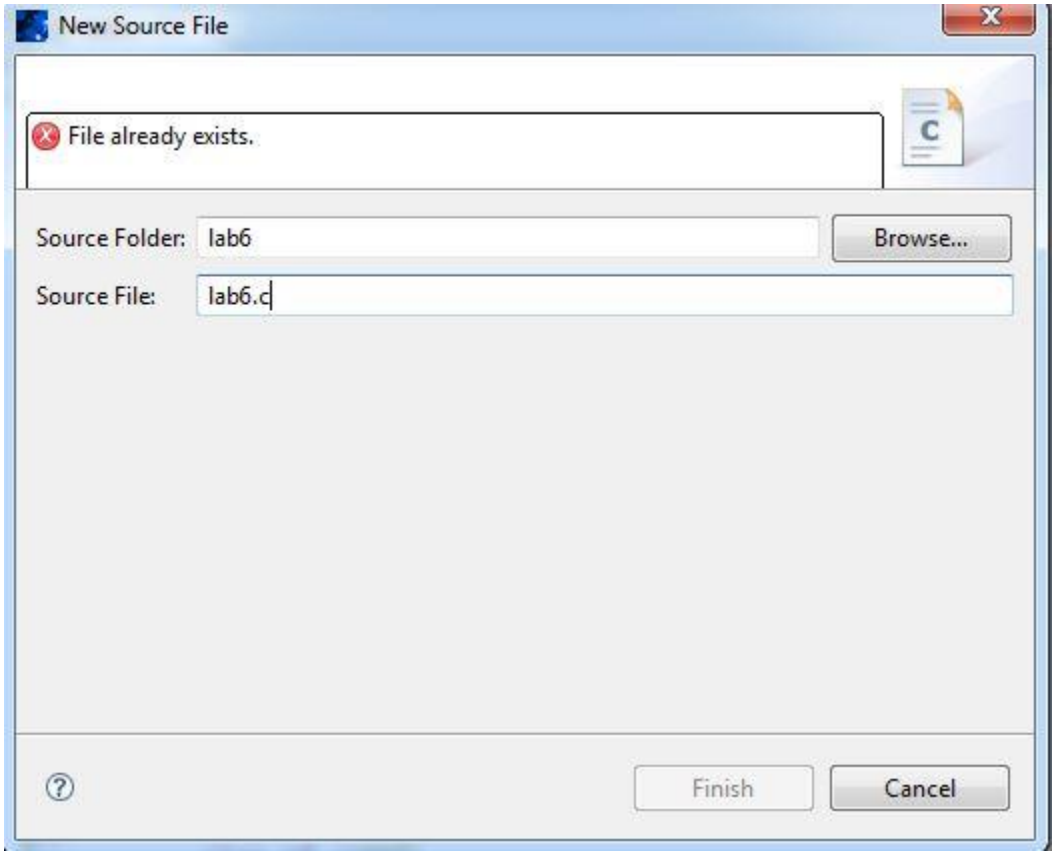
   Chọn đường dẫn để đến file **nios_system.ptf** (vừa tạo được ở các bước trên) ở mục **SOPC Builder System PTF File**

   Sau đó chọn **Finish.**



5. Click chuột phải vào thư mục **lab6_syslib[nios_system] -> Build Project**

6. Click chuột phải vào thư mục **lab6 → New → Source File.** Đặt tên source file giống với tên project mình đặt



7. Lập trình code C:

```
void VGA_text (int, int, char *);
void VGA_box (int, int, int, int, short);
int main(void)
{
   /* Declare volatile pointer to pixel DMA controller (volatile means that IO load
      and store instructions will be used to access these pointer locations,
      instead of regular memory loads and stores) */
   volatile int * Pixel_DMA_controller = (int *) 0x00008B000;  // DMA controller base
address

   int delay = 0;  // synchronize with the screen drawing

   /* these variables are used for a blue box  on the VGA screen */
   int ALT_x1; int ALT_x2; int ALT_y;
   int ALT_inc_x; int ALT_inc_y;
   int blue_x1; int blue_y1; int blue_x2; int blue_y2;
   int screen_x; int screen_y; int char_buffer_x; int char_buffer_y;
```

```
    short color;

    /* create messages to be displayed on the VGA display */
    char text_top_VGA[20] = "DHBK_HCMUT\0";
    char text_bottom_VGA[20] = "Lab6-VGA\0";


    /* the following variables give the size of the pixel buffer */
    screen_x = 319; screen_y = 239;
    color = 0x1863;     // a dark grey color
    VGA_box (0, 0, screen_x, screen_y, color);  // fill the screen with grey
    // draw a medium-blue box around the above text, based on the character buffer
coordinates
    blue_x1 = 28; blue_x2 = 52; blue_y1 = 26; blue_y2 = 34;
    // character coords * 4 since characters are 4 x 4 pixel buffer coords (8 x 8 VGA
coords)
    color = 0x187F;     // a medium blue color
    VGA_box (blue_x1 * 4, blue_y1 * 4, blue_x2 * 4, blue_y2 * 4, color);
    /* output text message in the middle of the VGA monitor */
    VGA_text (blue_x1 + 5, blue_y1 + 3, text_top_VGA);
    VGA_text (blue_x1 + 5, blue_y1 + 4, text_bottom_VGA);

}

/*****************************************************************************
*****************
 * Subroutine to send a string of text to the VGA monitor
*****************************************************************************
****************/
void VGA_text(int x, int y, char * text_ptr)
{
    int offset;
    volatile char * character_buffer = (char *) 0x00088000; // VGA character buffer

    /* assume that the text string fits on one line */
    offset = (y << 7) + x;
    while ( *(text_ptr) )
    {
        *(character_buffer + offset) = *(text_ptr); // write to the character buffer
        ++text_ptr;
```

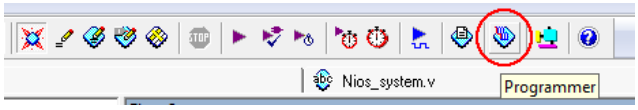
```
        ++offset;
    }
}


void VGA_box(int x1, int y1, int x2, int y2, short pixel_color)
{
    int offset, row, col;
    volatile short * pixel_buffer = (short *) 0x00000000;   // VGA pixel buffer

    /* assume that the box coordinates are valid */
    for (row = y1; row <= y2; row++)
    {
        col = x1;
        while (col <= x2)
        {
            offset = (row << 9) + col;
            *(pixel_buffer + offset) = pixel_color; // compute halfword address, set pixel
            ++col;
        }
    }

}
```

8. Save lại và Click chuột phải vào **lab6 -> Build Project**

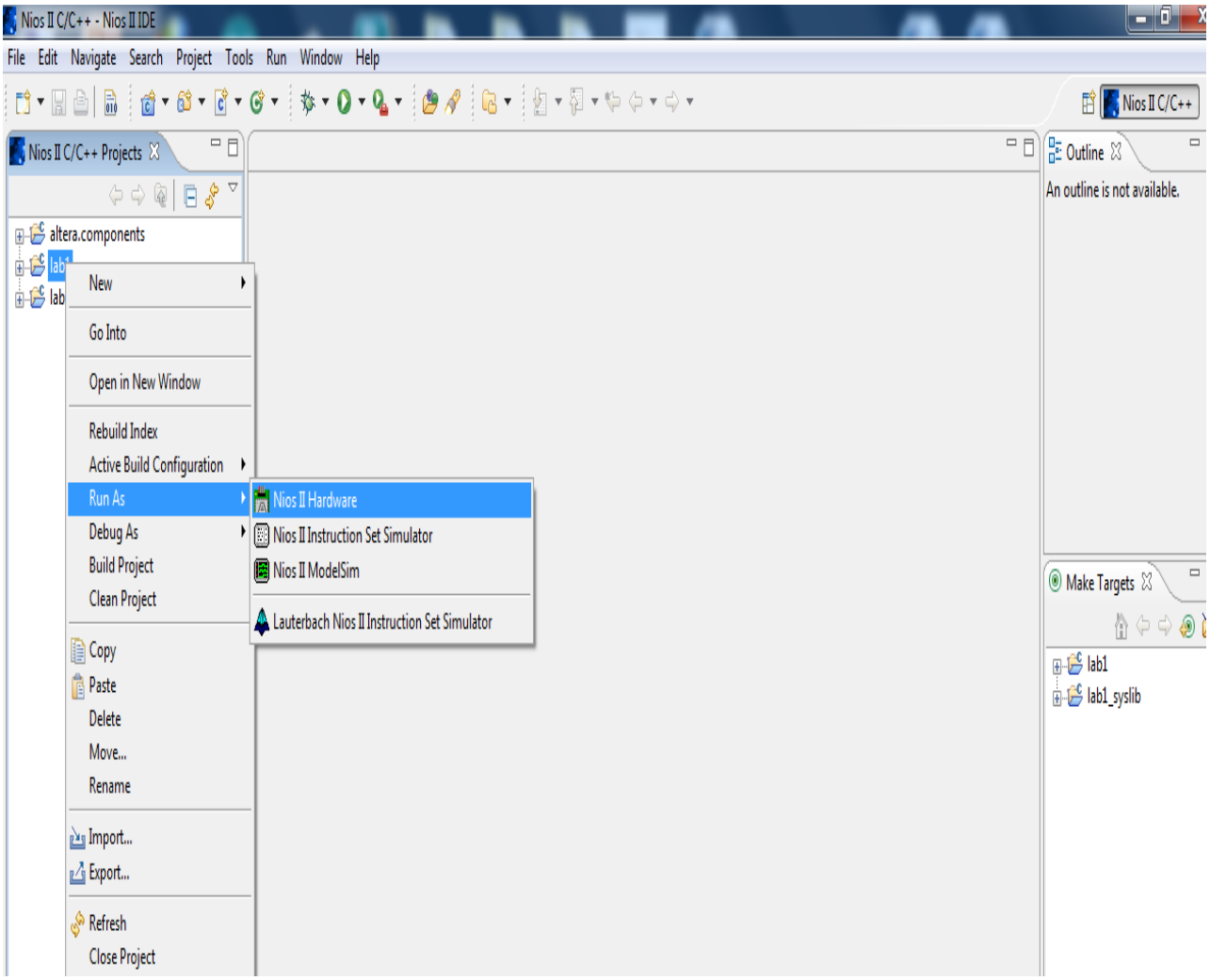
VI. Run Hardware on DE2 board:

1. USB Blaster:

- In window Quartus II, click **Programmer** in taskbar



2. Run


Nios II C/C++ - Nios II IDE
File Edit Navigate Search Project Tools Run Window Help
Nios II C/C++
Nios II C/C++ Projects
altera.components
New
Go Into
Open in New Window
Rebuild Index
Active Build Configuration
Run As
Debug As
Build Project
Clean Project
Copy
Paste
Delete
Move...
Rename
Import...
Export...
Refresh
Close Project
Nios II Hardware
Nios II Instruction Set Simulator
Nios II ModelSim
Lauterbach Nios II Instruction Set Simulator
Outline
An outline is not available.
Make Targets
lab1
lab1_syslib