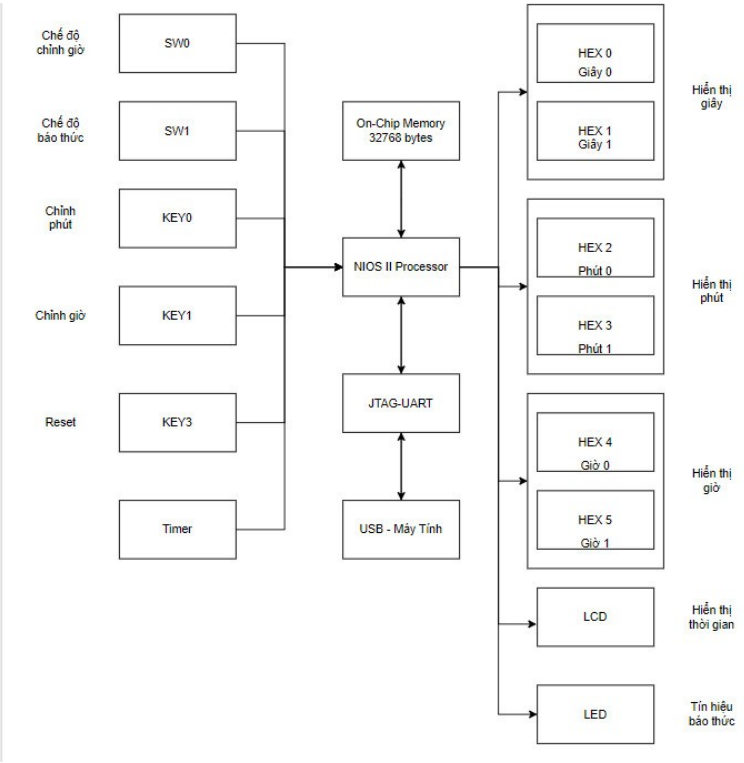
BÁO CÁO THỰC HÀNH "THIẾT KẾ SoC"

Tên bài thực hành: Thiết kế Đồng Hồ.....

I. Mô tả tóm tắt nội dung bài thực hành

Thiết kế Đồng Hồ có các chức năng sau:
- Chạy giờ, phút, giây lên LCD và HEX
- Có thể chỉnh thời gian giờ và phút
- Có thể Reset thời gian về thời gian mà đã đặt
- Có thể cài thời gian báo thức thông qua LEDG 5s.

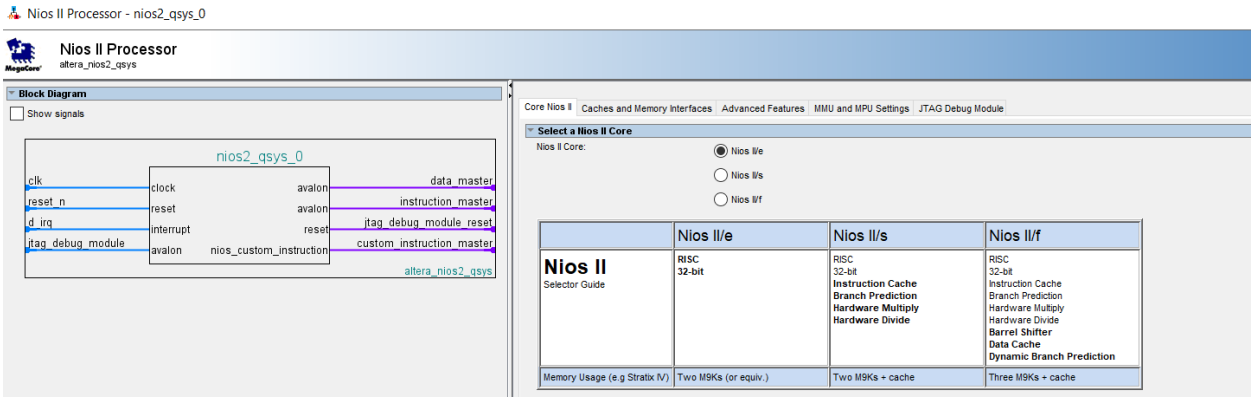


II. Tiến trình thực hiện bài thực hành

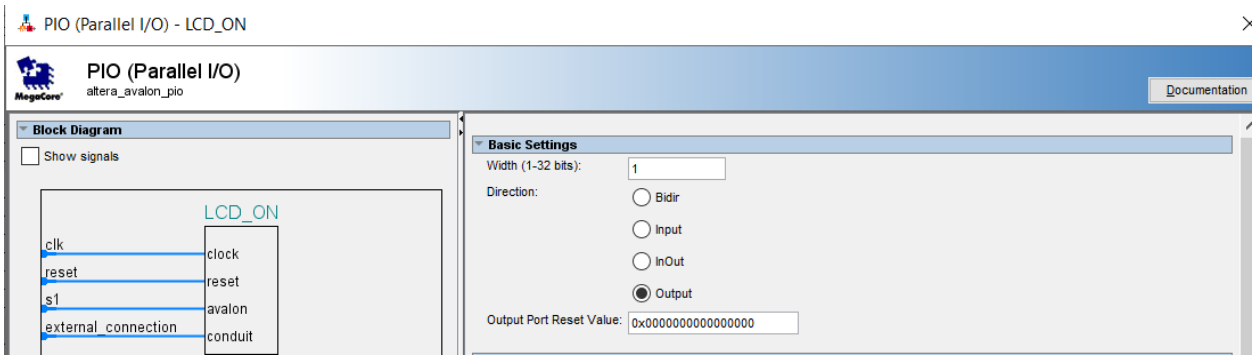
Phần 1: Tạo Hardware

<u>Bước 1:</u> Tạo File Quartus II với tên DongHo

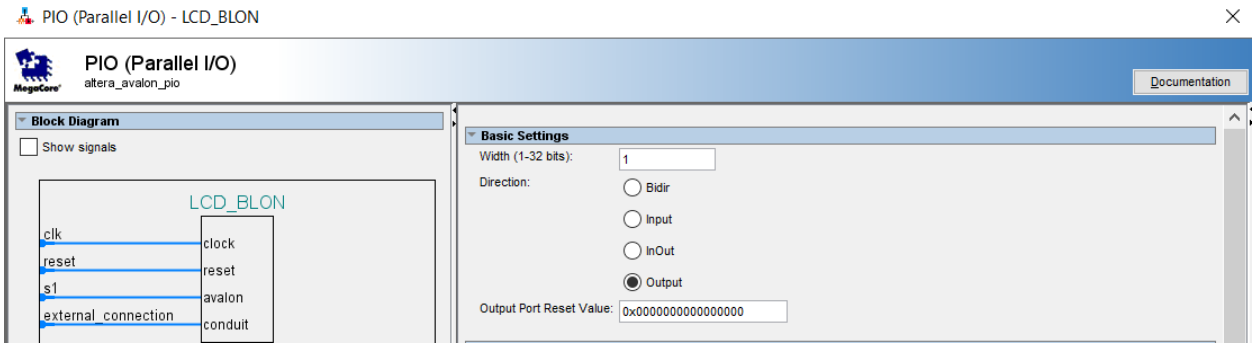
<u>Bước 2:</u> Tạo Qsys với tên file là system
Cần chỉnh thông số như sau:



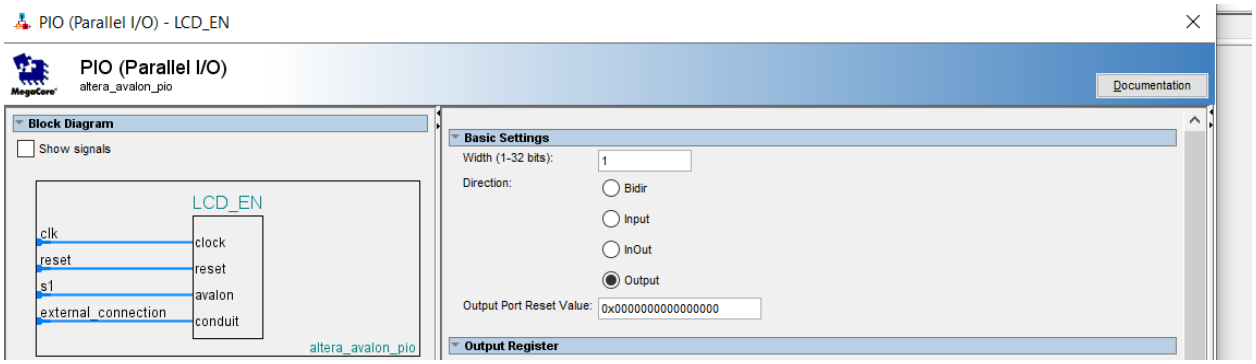
- *LCD ON*



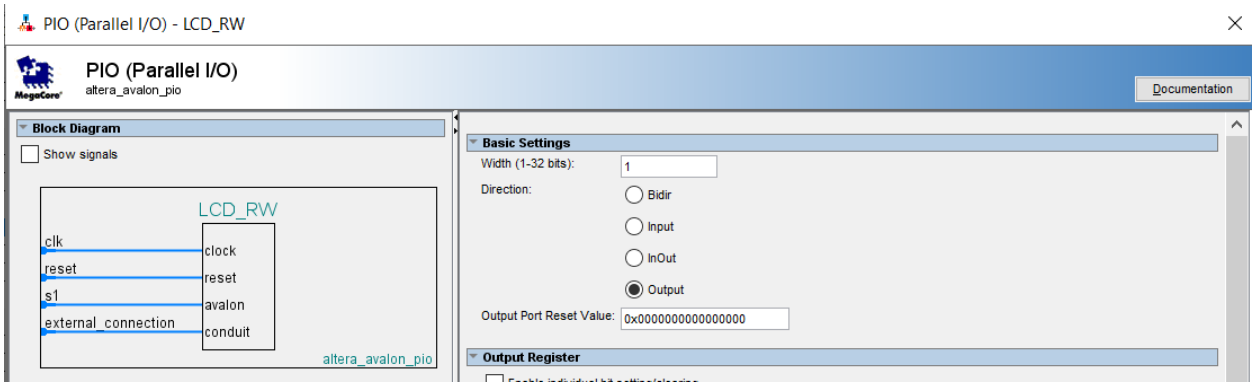
- *LCD BLON*



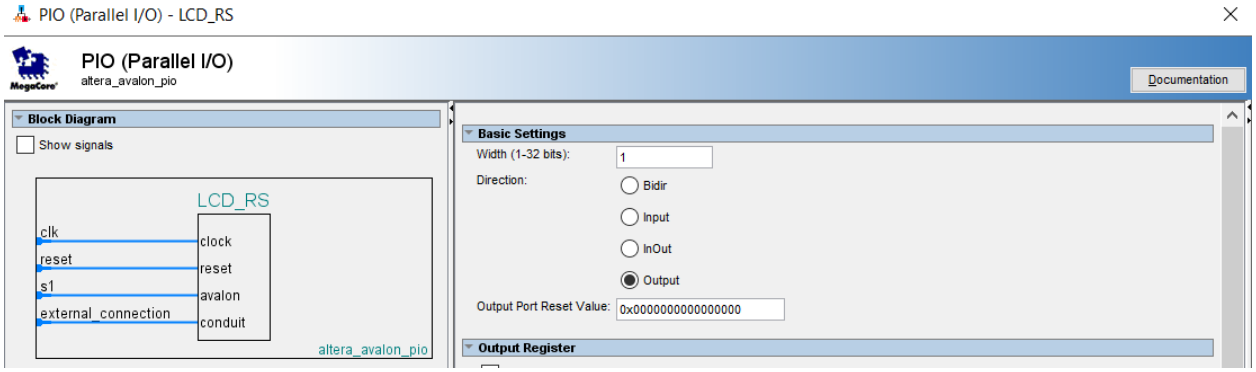
- *LCD EN*



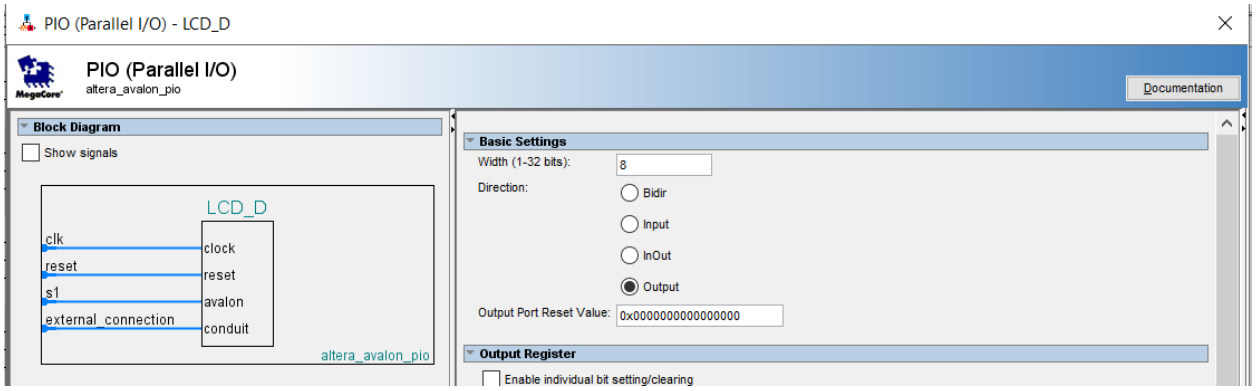
- *LCD RW*



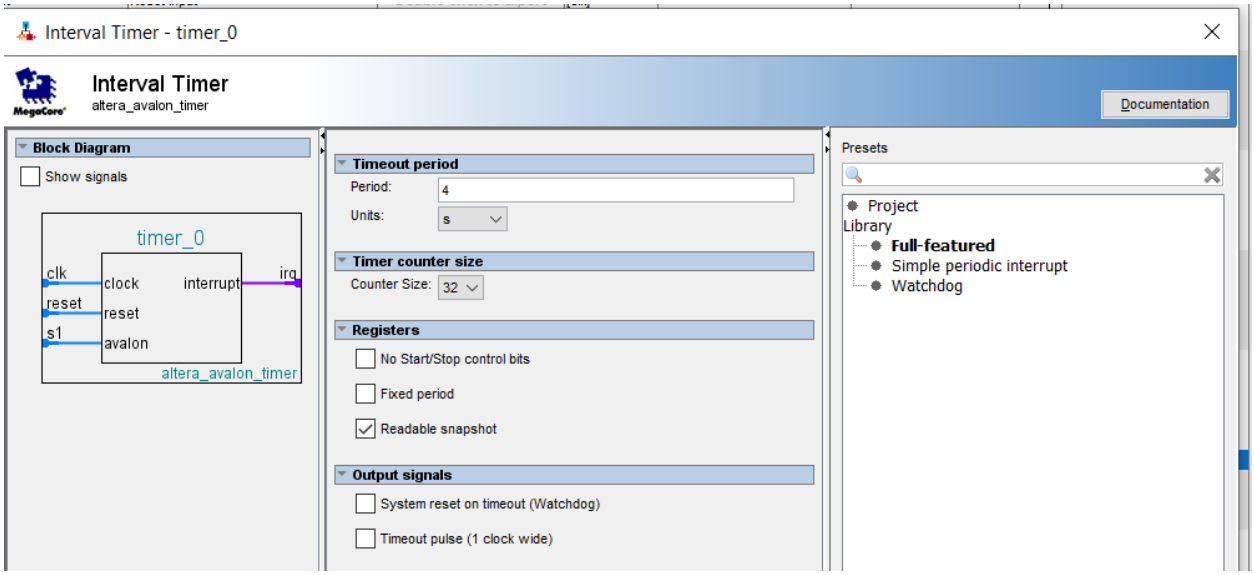
- *LCD RS*



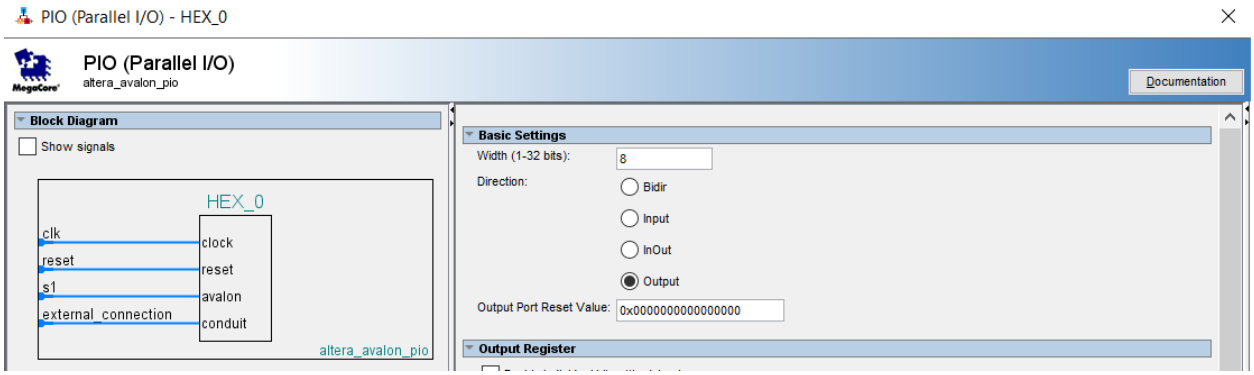
- *LCD D*



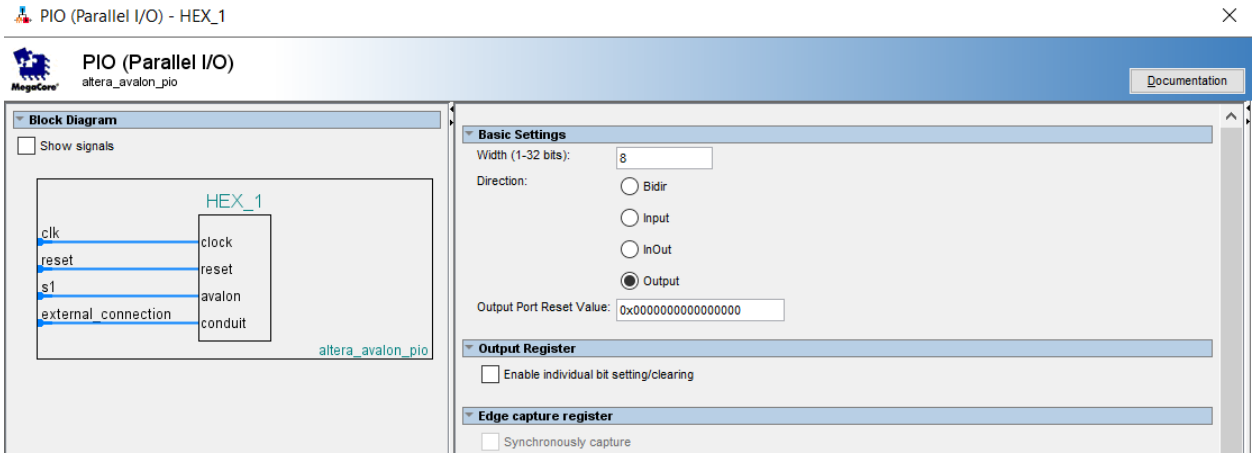
- *Timer 0*



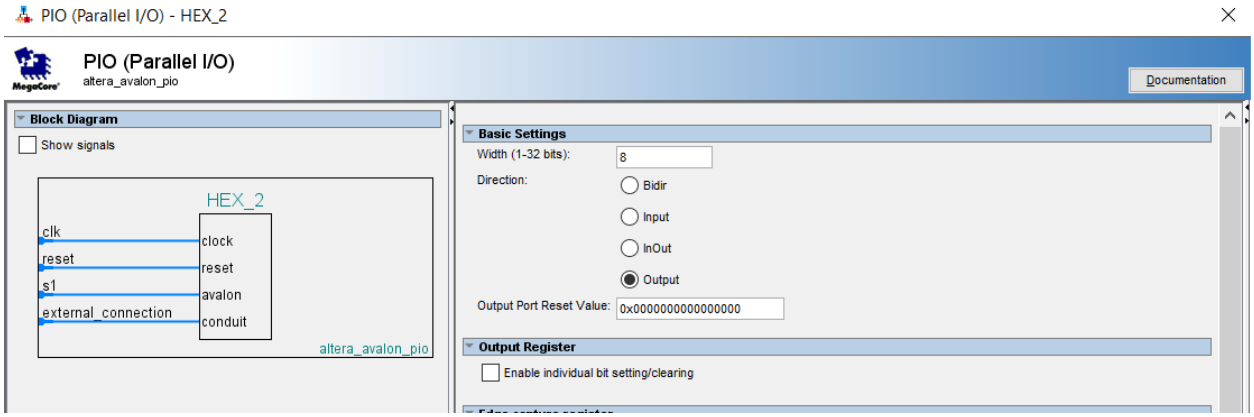
- *Hex 0*



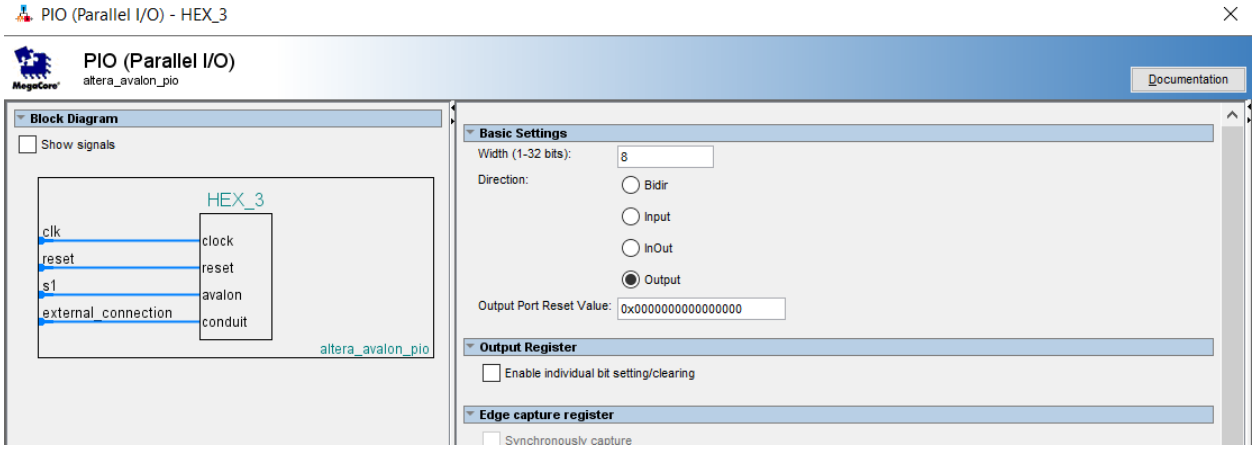
- *Hex 1*



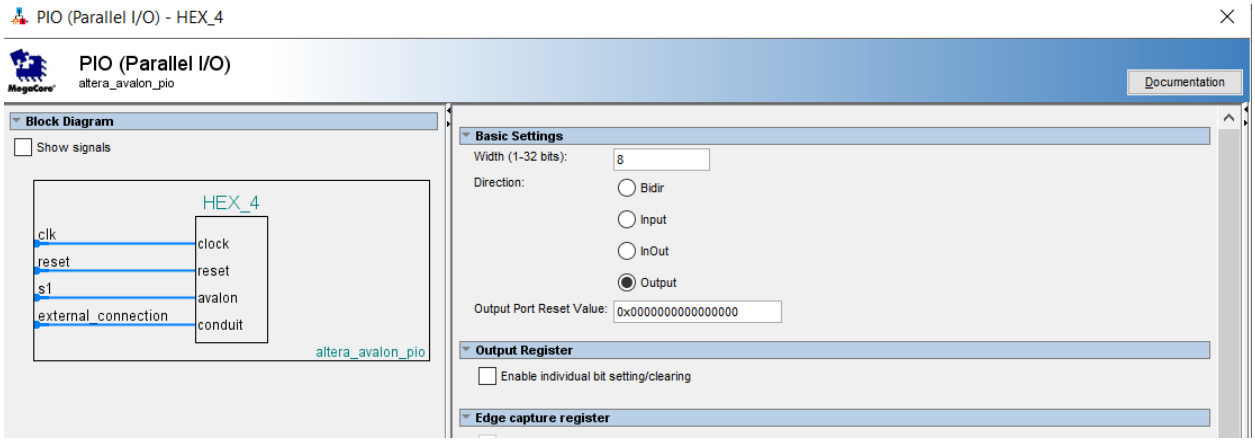
- *Hex 2*



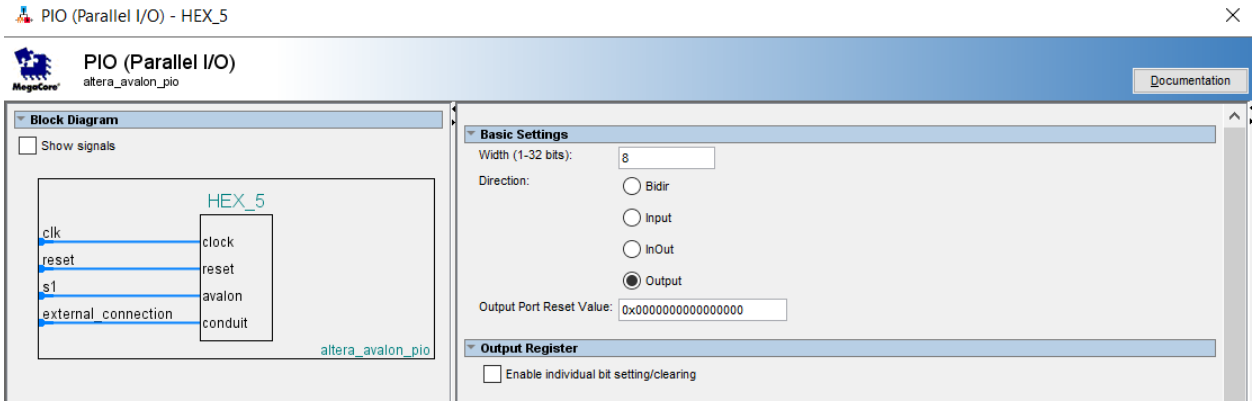
- *Hex 3*



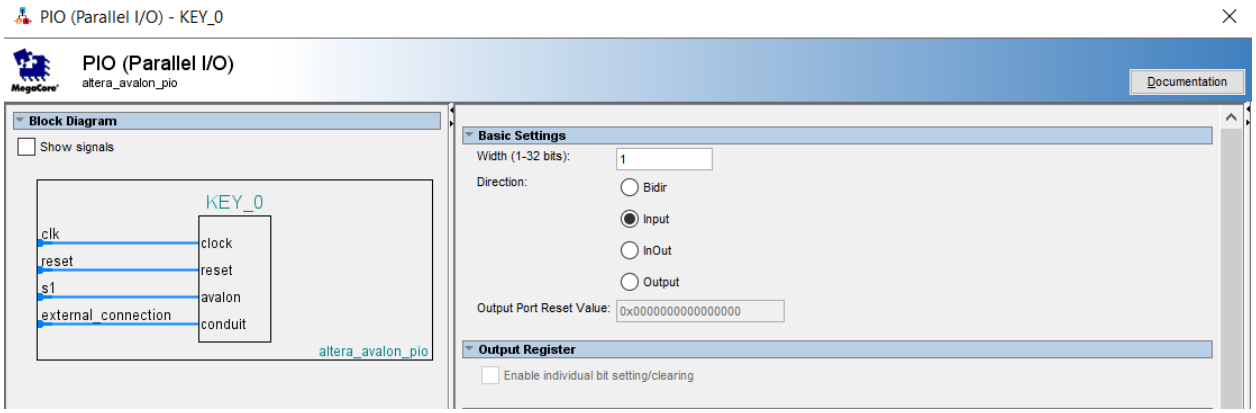
- *Hex 4*



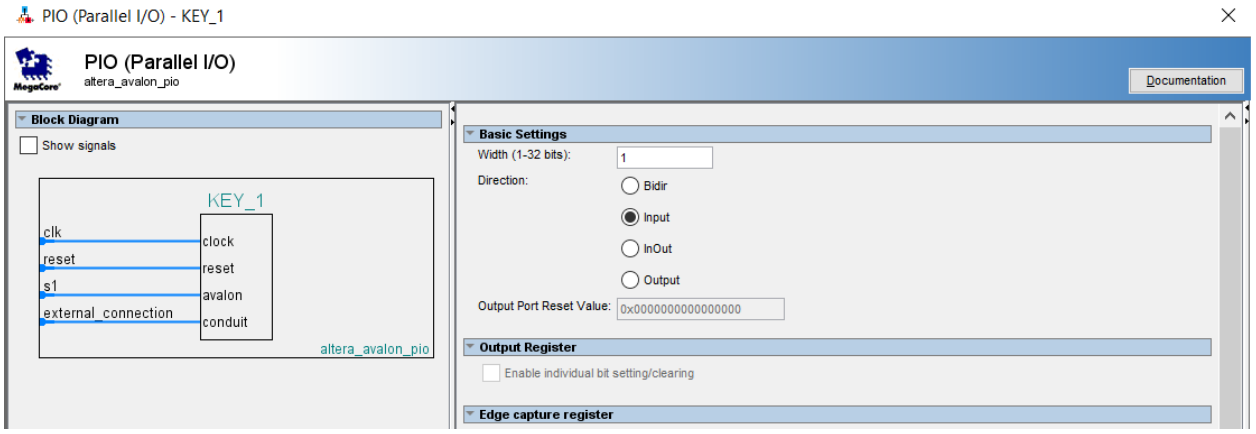
- *Hex 5*



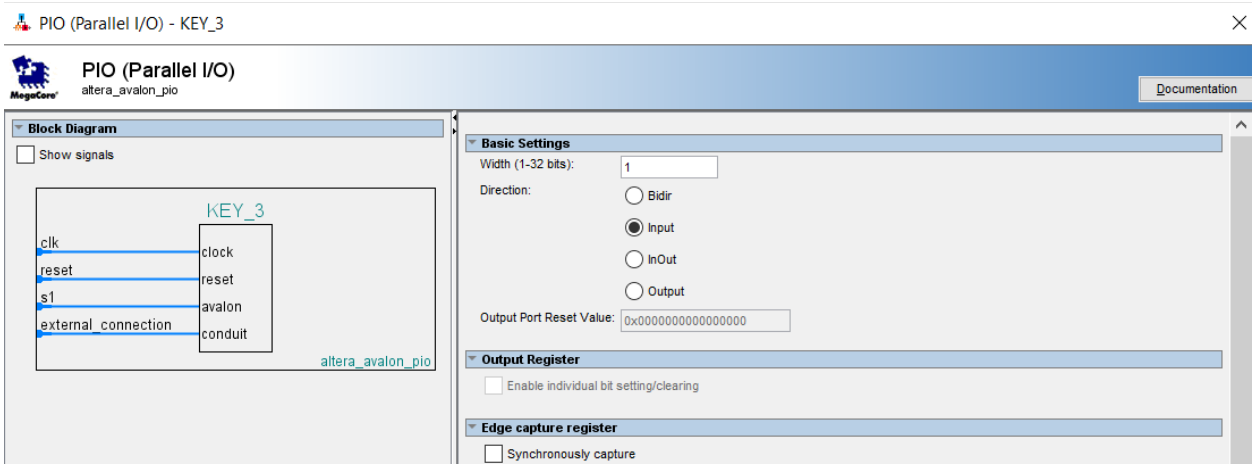
- *Key 0*



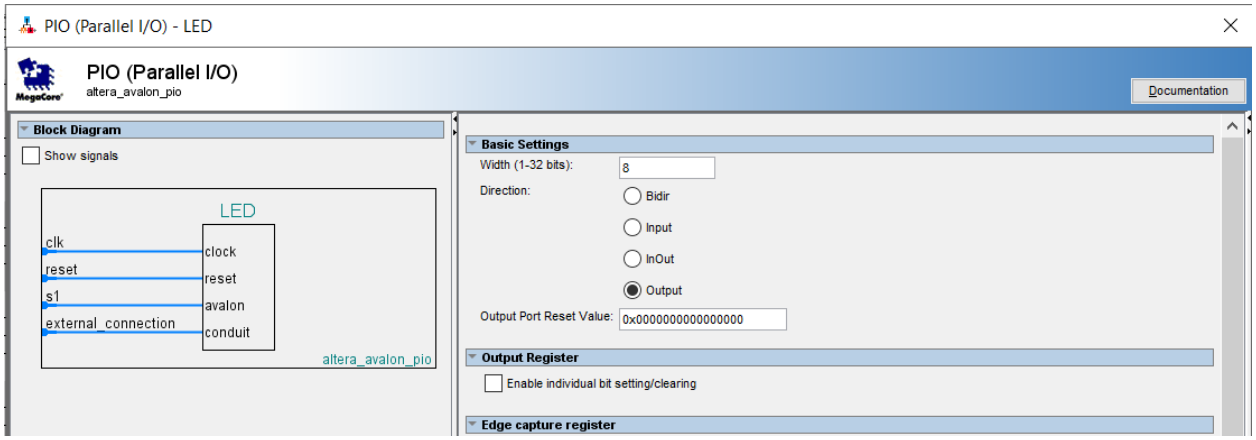
- *Key 1*



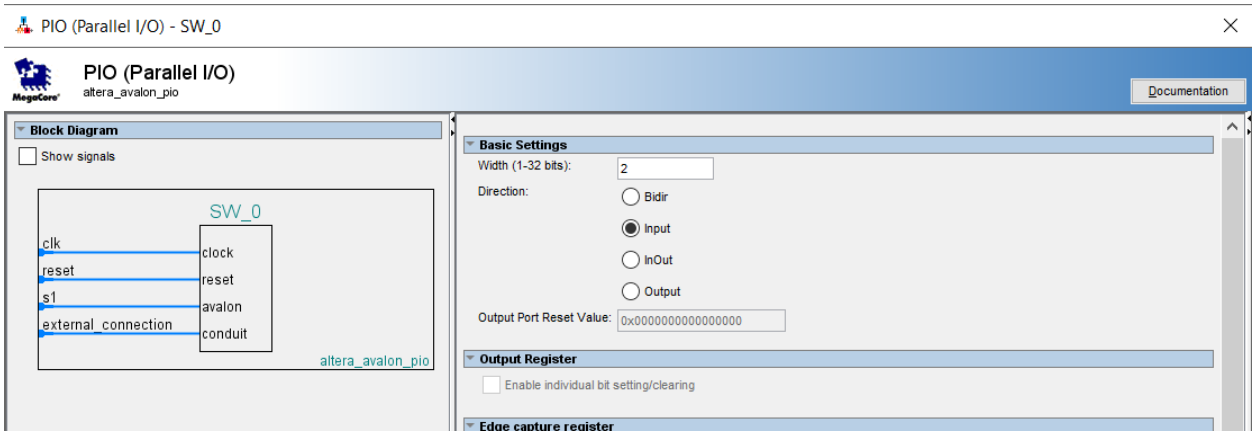
- *Key 3*


PIO (Parallel I/O) - KEY_3
PIO (Parallel I/O)
altera_avalon_pio
Documentation
Block Diagram
Show signals
KEY_3
clk
reset
s1
external_connection
clock
reset
avalon
conduit
altera_avalon_pio
Basic Settings
Width (1-32 bits):
1
Direction:
Bidir
Input
InOut
Output
Output Port Reset Value:
0x0000000000000000
Output Register
Enable individual bit setting/clearing
Edge capture register
Synchronously capture

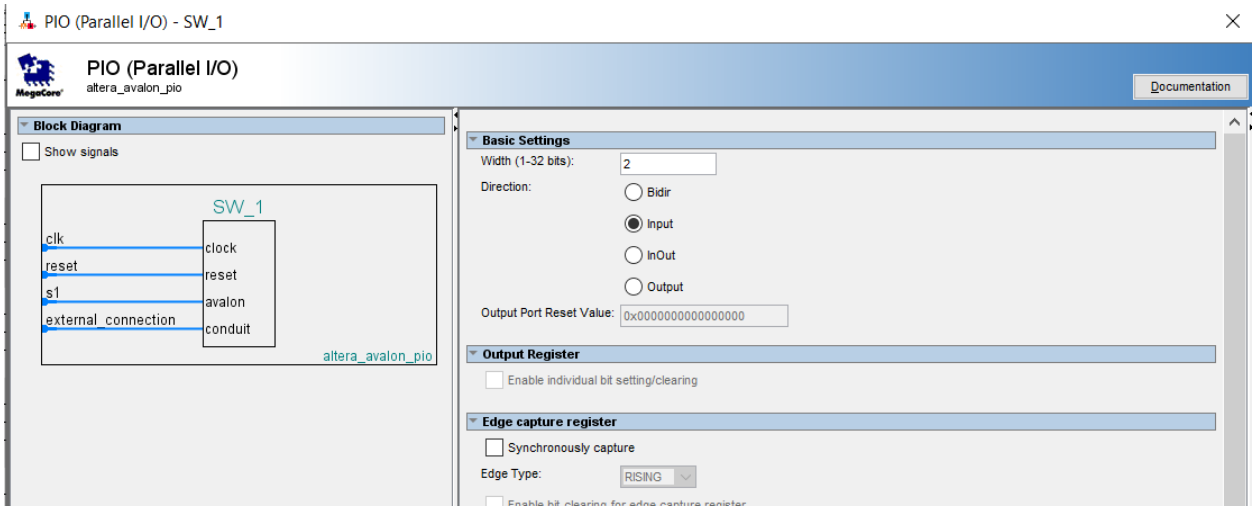
- *LED*


PIO (Parallel I/O) - LED
PIO (Parallel I/O)
altera_avalon_pio
Documentation
Block Diagram
Show signals
LED
clk
reset
s1
external_connection
clock
reset
avalon
conduit
altera_avalon_pio
Basic Settings
Width (1-32 bits):
8
Direction:
Bidir
Input
InOut
Output
Output Port Reset Value:
0x0000000000000000
Output Register
Enable individual bit setting/clearing
Edge capture register

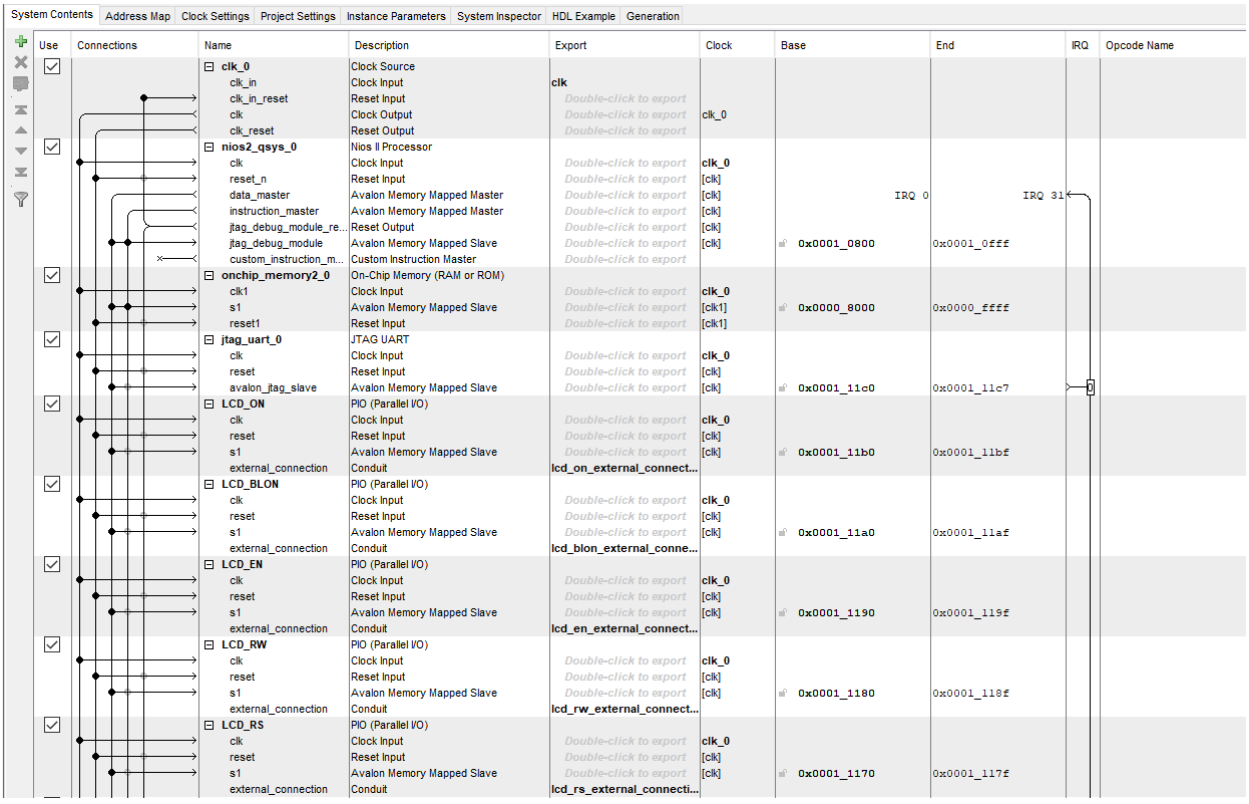
- *SW 0*

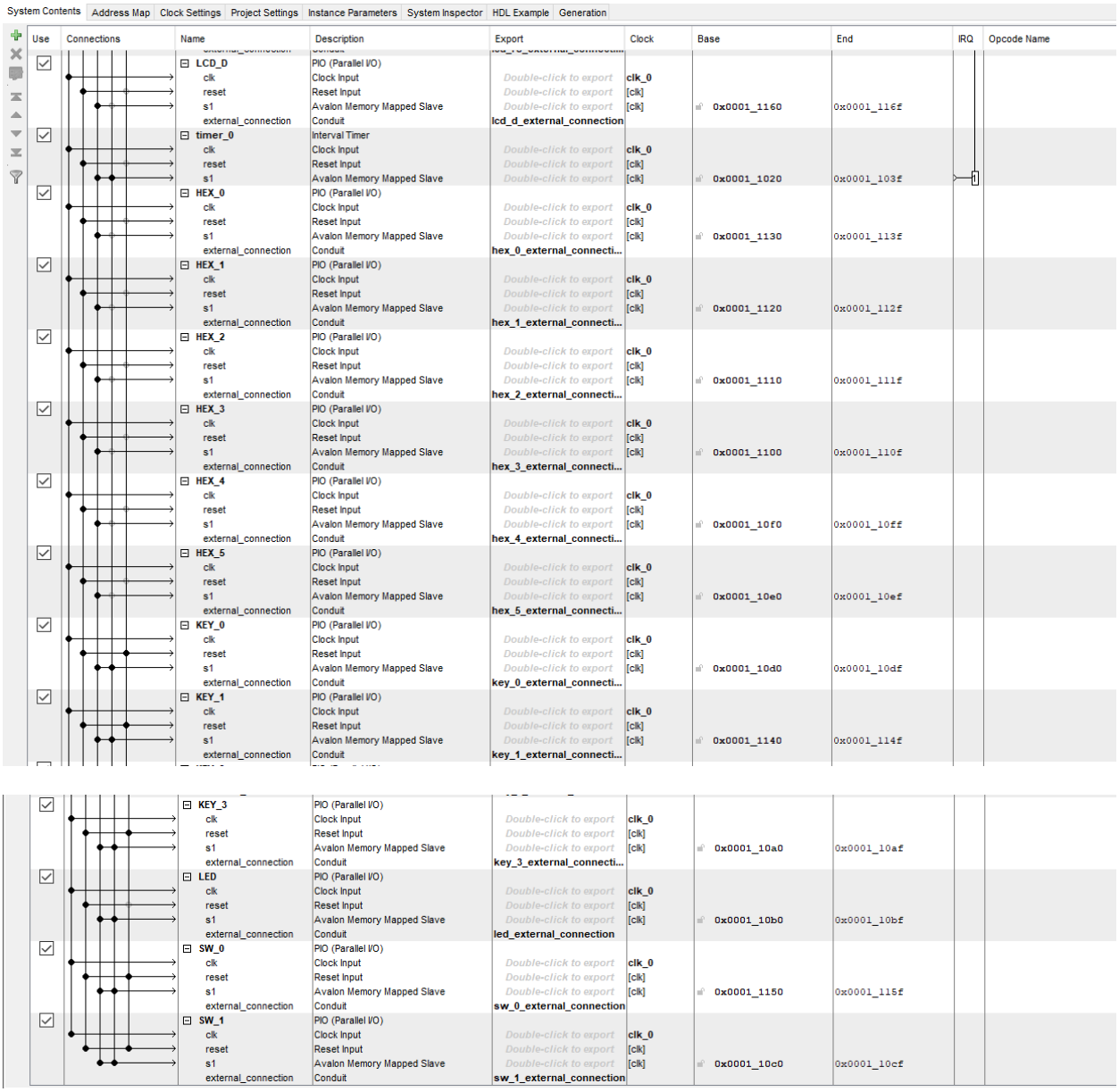

PIO (Parallel I/O) - SW_0
PIO (Parallel I/O)
altera_avalon_pio
Documentation
Block Diagram
Show signals
SW_0
clk
reset
s1
external_connection
clock
reset
avalon
conduit
altera_avalon_pio
Basic Settings
Width (1-32 bits):
2
Direction:
Bidir
Input
InOut
Output
Output Port Reset Value:
0x0000000000000000
Output Register
Enable individual bit setting/clearing
Edge capture register

- *SW 1*

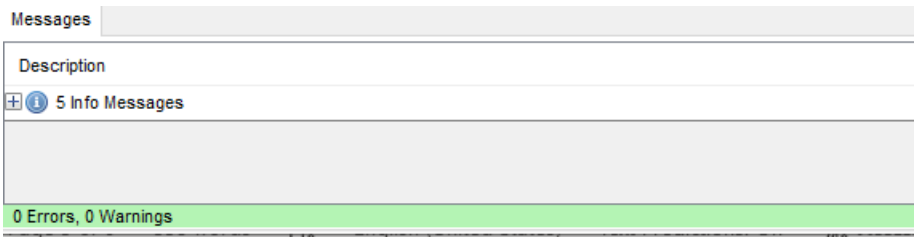


Sau khi thực hiện có Qsys sau:

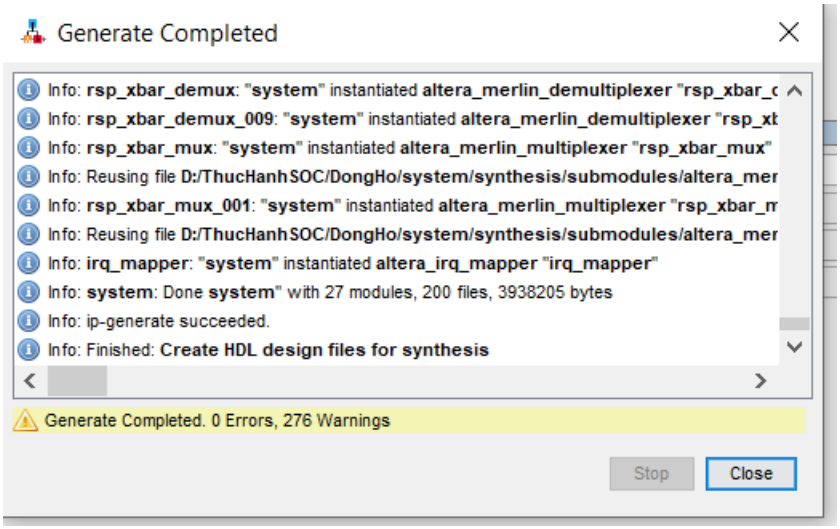




Thiết kế đúng sẽ hiển thị nội dung sau:



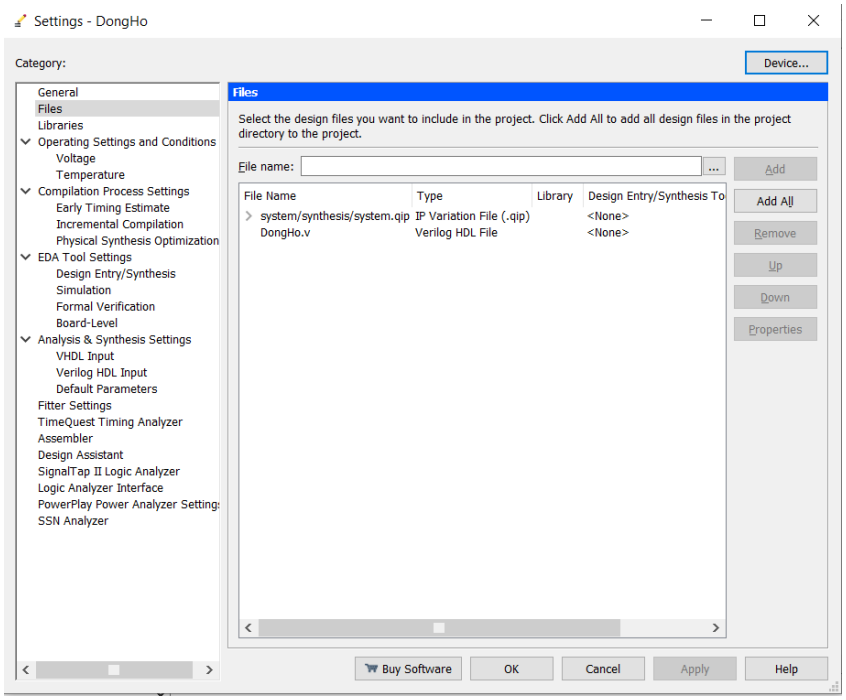
Sau đó qua Generation và nhấn Generate để chạy:



Sau khi chạy xong thì nhấn Close và trở lại Quartus để thực hiện tiếp.

<u>Bước 3:</u> Ta vào Quartus nhấn vào Project + Add/Remove Files in Project….

Tiếp theo tìm kiếm file .qip để add vào như hình bên dưới:



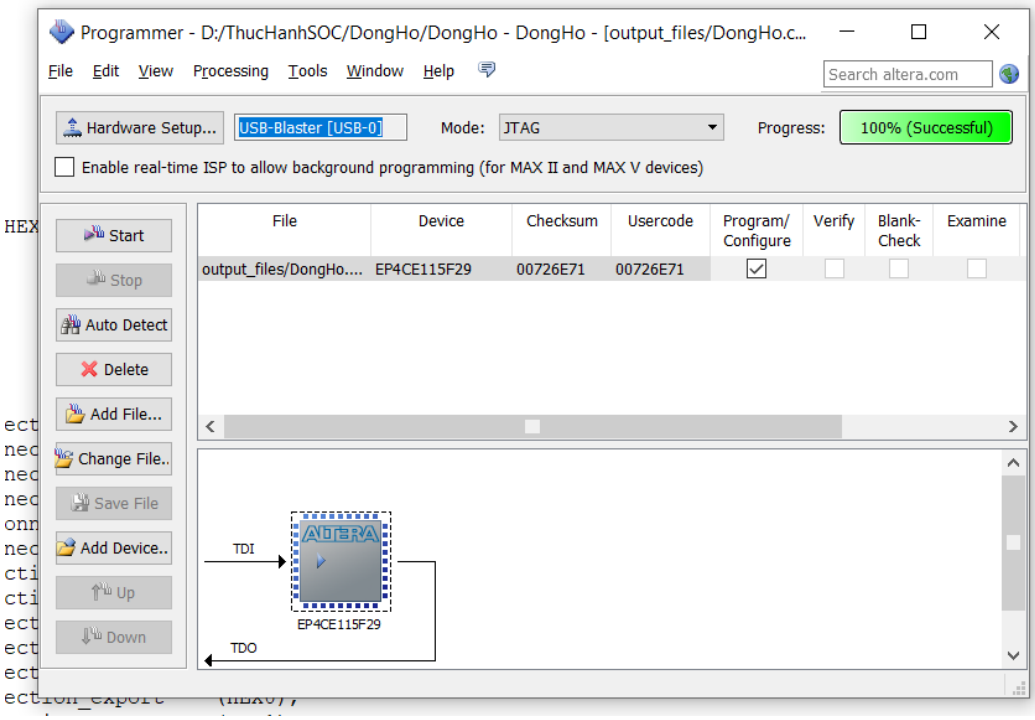
Tạo file .v có tên là DongHo.v

```
module DongHo(
    input CLOCK_50,
    input    [3:0] KEY,
    input    [1:0] SW,
    output   [7:0] LCD_DATA,
    output         LCD_EN,
    output         LCD_RS,
    output         LCD_RW,
    output         LCD_BLON,
    output         LCD_ON,
    output   [6:0] HEX0,HEX1,HEX2,HEX3,HEX4,HEX5,HEX6,HEX7,
    output   [7:0] LEDG
);
    assign HEX6 = 7'b1111111;
    assign HEX7 = 7'b1111111;

system u0 (
        .clk_clk                              (CLOCK_50),
        .lcd_d_external_connection_export     (LCD_DATA),
        .lcd_rs_external_connection_export    (LCD_RS),
        .lcd_rw_external_connection_export    (LCD_RW),
        .lcd_en_external_connection_export    (LCD_EN),
        .lcd_blon_external_connection_export  (LCD_BLON),
        .lcd_on_external_connection_export    (LCD_ON),
        .sw_0_external_connection_export      (SW[0]),
        .sw_1_external_connection_export      (SW[1]),
        .key_0_external_connection_export     (KEY[0]),
        .key_1_external_connection_export     (KEY[1]),
        .key_3_external_connection_export     (KEY[3]),
        .hex_0_external_connection_export     (HEX0),
        .hex_1_external_connection_export     (HEX1),
        .hex_2_external_connection_export     (HEX2),
        .hex_3_external_connection_export     (HEX3),
        .hex_4_external_connection_export     (HEX4),
        .hex_5_external_connection_export     (HEX5),
        .led_external_connection_export       (LEDG)

    );
endmodule
```

Chạy chương trình và nạp vào kit

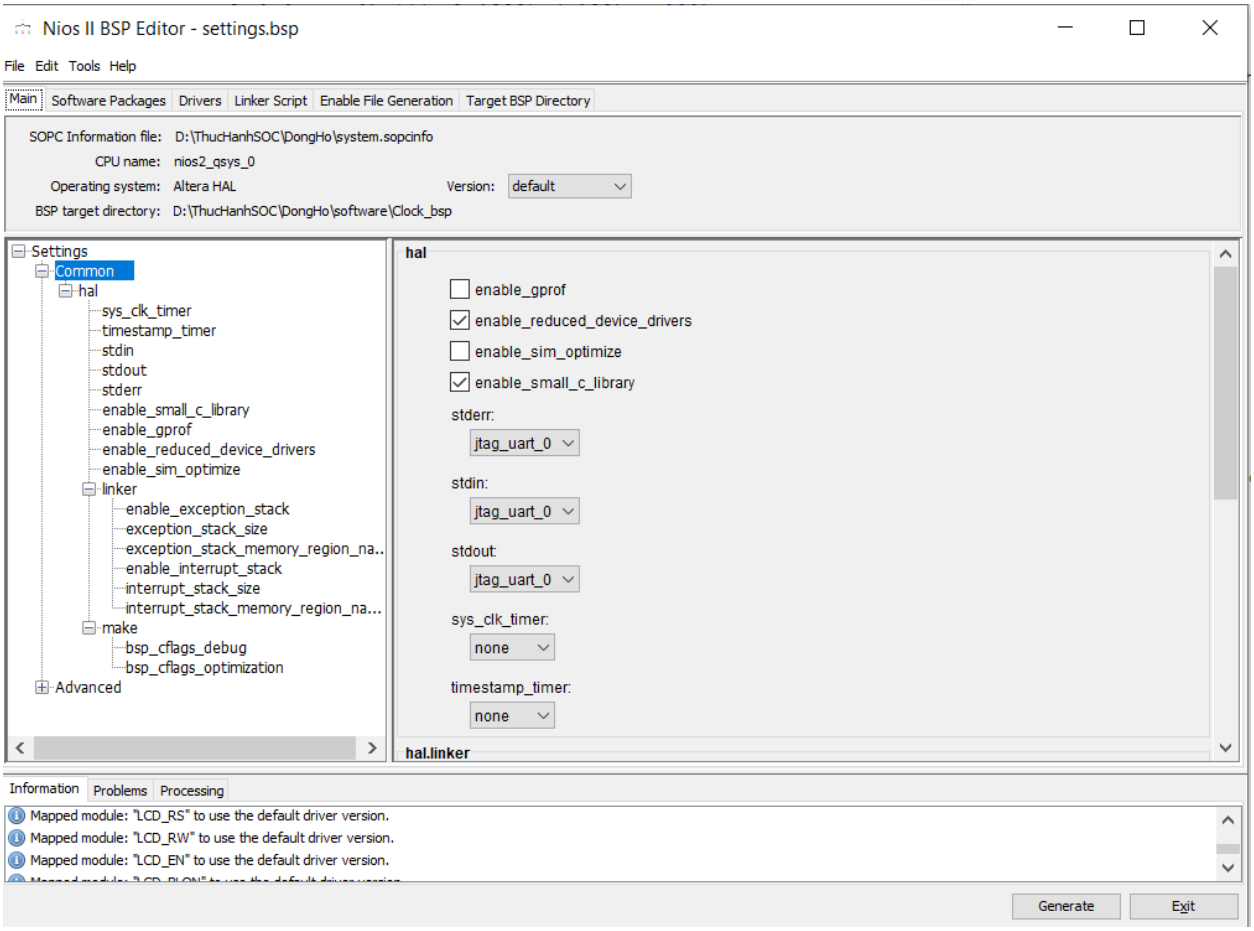


Phần 2: Viết code firmware (code C trong Eclipse)

Bước 1: Mở phần mềm eclipse

Tạo Project với đường link bài làm
Tiếp theo tạo file .c có tên là Clock.c

Bước 2: Chuột phải vào .bsp đưa chuột đến Nios II + BSP Editor… để điều chỉnh như hình dưới đây:

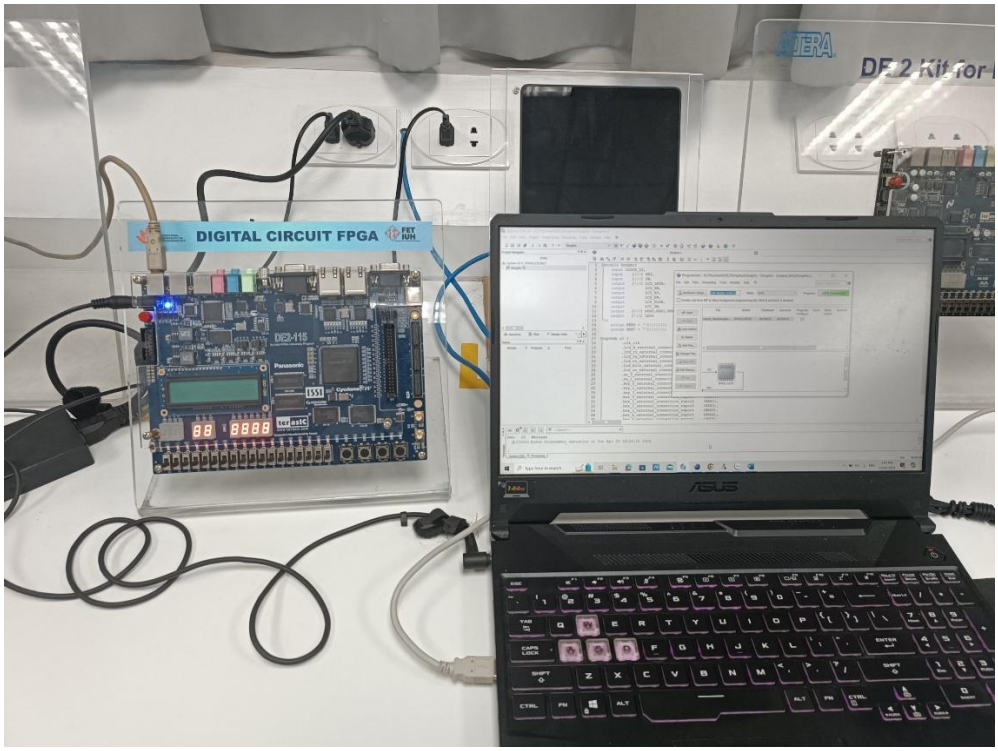


Bước 3: Chạy code và nạp chương trình vào Kit

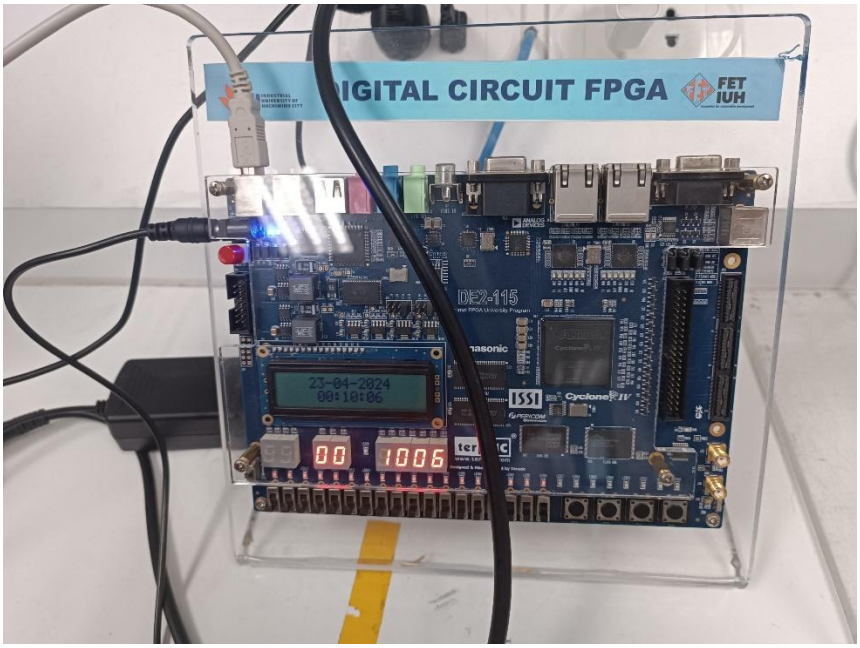
Phần 3: Kiểm thử trên kit DE2

Bước 1: Chuẩn bị Kit, Dây cắm, Laptop

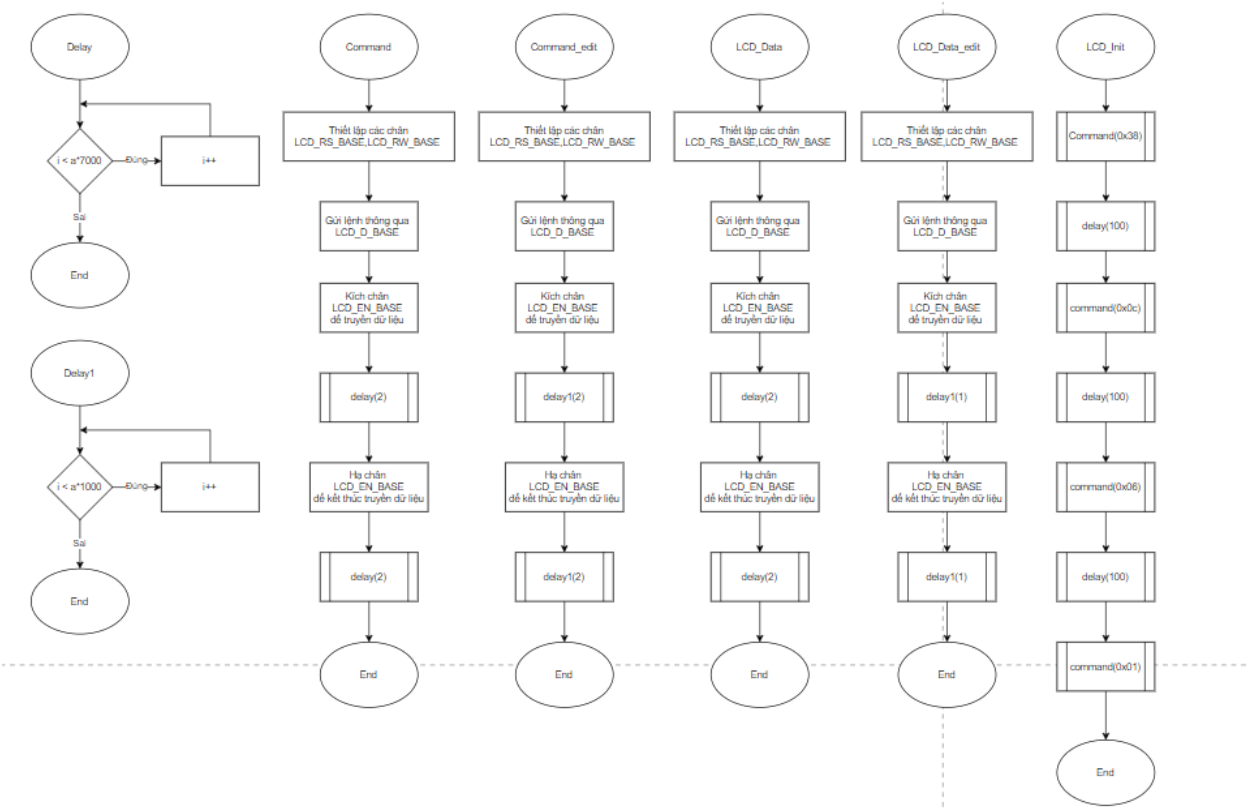
Bước 2: Kết nối vào kit và nạp chạy kit

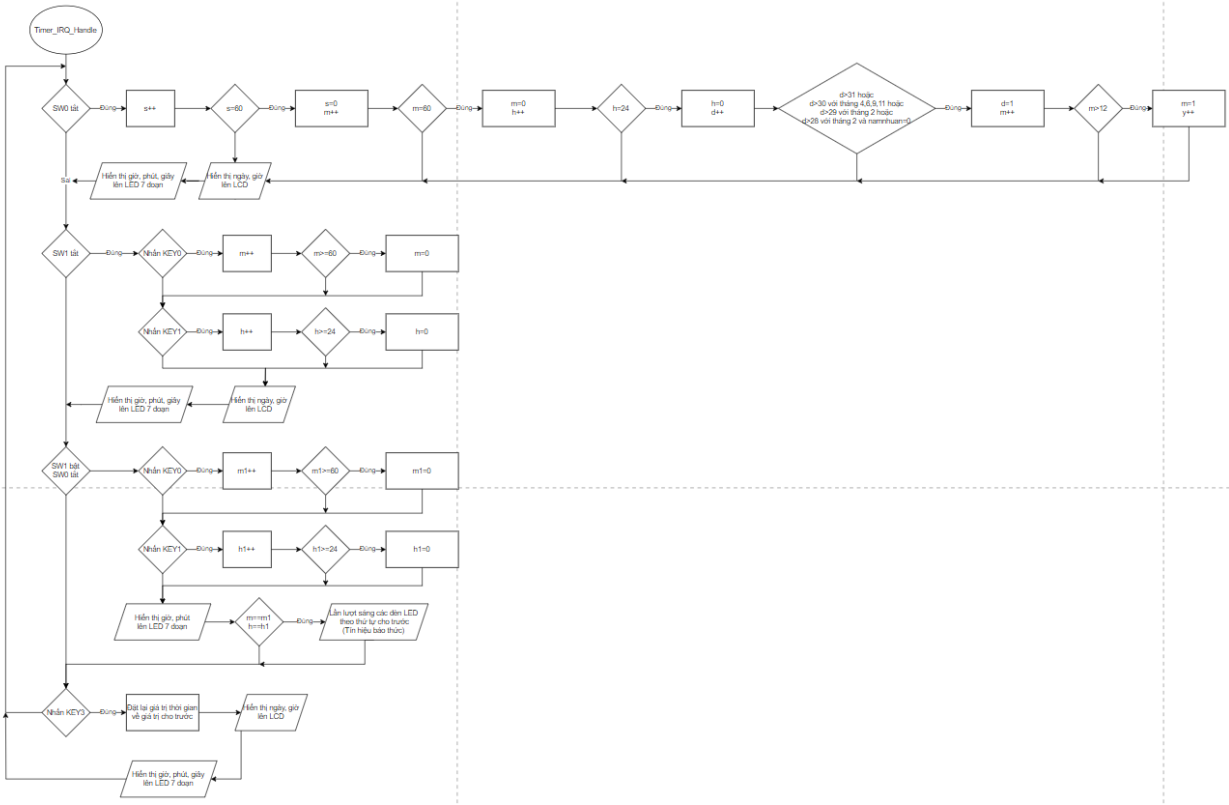


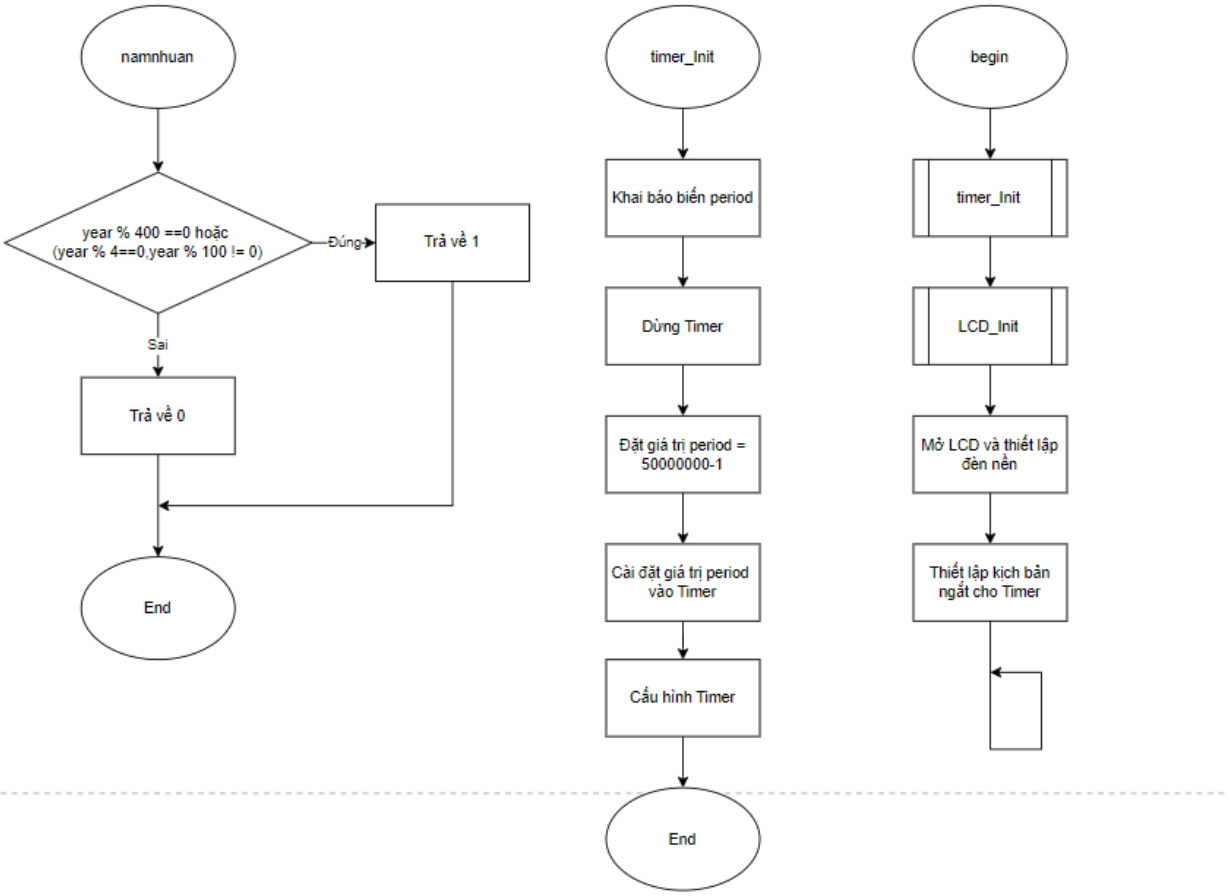
<u>Bước 3:</u> Nạp code Clipse xuống kit để xem kết quả:



III. Lưu đồ giải thuật


Delay
i < a*7000
Đúng
i++
Sai
End
Delay1
i < a*1000
Đúng
i++
Sai
End
Command
Thiết lập các chân LCD_RS_BASE,LCD_RW_BASE
Gửi lệnh thông qua LCD_D_BASE
Kích chân LCD_EN_BASE để truyền dữ liệu
delay(2)
Hạ chân LCD_EN_BASE để kết thúc truyền dữ liệu
delay(2)
End
Command_edit
Thiết lập các chân LCD_RS_BASE,LCD_RW_BASE
Gửi lệnh thông qua LCD_D_BASE
Kích chân LCD_EN_BASE để truyền dữ liệu
delay1(2)
Hạ chân LCD_EN_BASE để kết thúc truyền dữ liệu
delay1(2)
End
LCD_Data
Thiết lập các chân LCD_RS_BASE,LCD_RW_BASE
Gửi lệnh thông qua LCD_D_BASE
Kích chân LCD_EN_BASE để truyền dữ liệu
delay(2)
Hạ chân LCD_EN_BASE để kết thúc truyền dữ liệu
delay(2)
End
LCD_Data_edit
Thiết lập các chân LCD_RS_BASE,LCD_RW_BASE
Gửi lệnh thông qua LCD_D_BASE
Kích chân LCD_EN_BASE để truyền dữ liệu
delay1(1)
Hạ chân LCD_EN_BASE để kết thúc truyền dữ liệu
delay1(1)
End
LCD_Init
Command(0x38)
delay(100)
command(0x0c)
delay(100)
command(0x06)
delay(100)
command(0x01)
End


Timer_IRQ_Handle
SW0 tắt
Đúng
s++
s=60
s=0
m++
m=60
m=0
h++
h=24
h=0
d++
d=31 hoặc
d=30 với tháng 4,6,9,11 hoặc
d=29 với tháng 2 hoặc
d=28 với tháng 2 và namnhuan=0
d=1
m++
m=12
m=1
y++
Sai
Hiển thị giờ, phút, giây lên LED 7 đoạn
Hiển thị ngày, giờ lên LCD
SW1 tắt
Nhấn KEY0
m++
m>=60
m=0
Nhấn KEY1
h++
h>=24
h=0
SW1 bật
SW0 tắt
m1++
m1>=60
m1=0
h1++
h1>=24
h1=0
Hiển thị giờ, phút lên LED 7 đoạn
m=m1
h=h1
Lần lượt sáng các đèn LED theo thứ tự cho trước (Tín hiệu báo thức)
Nhấn KEY3
Đặt lại giá trị thời gian về giá trị cho trước



IV. Giải thích code Firmware

Code eclipse

```
#include <stdio.h>
#include "io.h"
#include "system.h"
#include "altera_avalon_timer_regs.h"
#include "sys/alt_irq.h"
#include "sys/alt_stdio.h"
#include "altera_avalon_pio_regs.h"
int s=0,h=0,m=0,d=23,mt=4,y=2024,m1=0,h1=0,c=0,t=0;
const unsigned char decoder[10] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8, 0x80, 0x90};
void delay(int a)
{
        volatile int i = 0;
        while(i < a*7000)
        {
                i++;
   }
}
void delay1(int a)
{
        volatile int i = 0;
        while(i < a*1000)
```

```
    {
                i++;
    }
}

void command(data)
{
        IOWR_ALTERA_AVALON_PIO_DATA(LCD_RS_BASE, 0x00);
        IOWR_ALTERA_AVALON_PIO_DATA(LCD_RW_BASE, 0x00);
        IOWR_ALTERA_AVALON_PIO_DATA(LCD_D_BASE, data&0xFF);
        IOWR_ALTERA_AVALON_PIO_DATA(LCD_EN_BASE, 0x01);
        delay(2);
        IOWR_ALTERA_AVALON_PIO_DATA(LCD_EN_BASE, 0x00);
        delay(2);
}
void command_edit(data)
{
        IOWR_ALTERA_AVALON_PIO_DATA(LCD_RS_BASE, 0x00);
        IOWR_ALTERA_AVALON_PIO_DATA(LCD_RW_BASE, 0x00);
        IOWR_ALTERA_AVALON_PIO_DATA(LCD_D_BASE, data&0xFF);
        IOWR_ALTERA_AVALON_PIO_DATA(LCD_EN_BASE, 0x01);
        delay1(2);
        IOWR_ALTERA_AVALON_PIO_DATA(LCD_EN_BASE, 0x00);
        delay1(2);
}
void lcd_data(char data)
{
        IOWR_ALTERA_AVALON_PIO_DATA(LCD_RS_BASE, 0x01);
        IOWR_ALTERA_AVALON_PIO_DATA(LCD_RW_BASE, 0x00);
        IOWR_ALTERA_AVALON_PIO_DATA(LCD_D_BASE, data&0xFF);
        IOWR_ALTERA_AVALON_PIO_DATA(LCD_EN_BASE, 0x01);
        delay(1);
        IOWR_ALTERA_AVALON_PIO_DATA(LCD_EN_BASE, 0x00);
        delay(1);
}
void lcd_data_edit(char data)
{
        IOWR_ALTERA_AVALON_PIO_DATA(LCD_RS_BASE, 0x01);
        IOWR_ALTERA_AVALON_PIO_DATA(LCD_RW_BASE, 0x00);
        IOWR_ALTERA_AVALON_PIO_DATA(LCD_D_BASE, data&0xFF);
        IOWR_ALTERA_AVALON_PIO_DATA(LCD_EN_BASE, 0x01);
        delay1(1);
        IOWR_ALTERA_AVALON_PIO_DATA(LCD_EN_BASE, 0x00);
        delay1(1);
}
void lcd_init()
{
        command(0x38);
        delay(10);
        command(0x0c);
        delay(10);
        command(0x06);
        delay(10);
        command(0x01);
}
int namnhuan(int year)
{
        if ((year % 400 == 0) || ((year % 4 == 0) && (year % 100 != 0)))
                return 1;
        else
                return 0;
}
void Timer_IRQ_Handle(void* isr_context)
{
        //const unsigned int decoder[10] = {0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90};
```

```
	while(1)
	{
		if(IORD(SW_0_BASE, 0) == 0)
		{
			s++;
			if (s == 60)
			{
				s = 0;
				m++;
				if (m == 60)
				{
					m = 0;
					h++;
					if (h == 24)
					{
						h = 0;
						d++;
						if (d > 31 || (d > 30 && (mt == 4 || mt == 6 || mt == 9 || mt == 11))
|| (d > 29 && mt == 2) || (d > 28 && mt == 2 && !namnhuan(y))) {
							d = 1;
							mt++;
							if (mt > 12)
							{
								mt = 1;
								y++;
							}
						}
					}
				}
			}
			command(0x02); // Clear display
			lcd_data((d / 10) + '0');
			lcd_data((d % 10) + '0');
			lcd_data('-');
			lcd_data((mt / 10) + '0');
			lcd_data((mt % 10) + '0');
			lcd_data('-');
			lcd_data((y / 1000) + '0');
			lcd_data(((y % 1000) / 100) + '0');
			lcd_data(((y % 100) / 10) + '0');
			lcd_data((y % 10) + '0');
			command(0xc0);
			lcd_data((h / 10) + '0');
			lcd_data((h % 10) + '0');
			lcd_data(':');
			lcd_data((m / 10) + '0');
			lcd_data((m % 10) + '0');
			lcd_data(':');
			lcd_data((s / 10) + '0');
			lcd_data((s % 10) + '0');
			IOWR(LED_BASE, 0, 0x00);
			IOWR(HEX_0_BASE, 0, decoder[s % 10]);
			IOWR(HEX_1_BASE, 0, decoder[s / 10]);
			IOWR(HEX_2_BASE, 0, decoder[m % 10]);
			IOWR(HEX_3_BASE, 0, decoder[m / 10]);
			IOWR(HEX_4_BASE, 0, decoder[h % 10]);
			IOWR(HEX_5_BASE, 0, decoder[h / 10]);


		}
		else
		{
			if(IORD(SW_1_BASE, 0) == 0)
			{
				if(IORD(KEY_0_BASE, 0)==0)
```

```
                {
                        m++;
                        if(m>=60)
                        {
                                m=0;
                        }
                }
                if(IORD(KEY_1_BASE, 0)==0)
                {
                        h++;
                        if(h>=24)
                        {
                                h=0;
                        }
                }
                command_edit(0x02); // Clear display
                lcd_data_edit((d / 10) + '0');
                lcd_data_edit((d % 10) + '0');
                lcd_data_edit('-');
                lcd_data_edit((mt / 10) + '0');
                lcd_data_edit((mt % 10) + '0');
                lcd_data_edit('-');
                lcd_data_edit((y / 1000) + '0');
                lcd_data_edit(((y % 1000) / 100) + '0');
                lcd_data_edit(((y % 100) / 10) + '0');
                lcd_data_edit((y % 10) + '0');
                command_edit(0xc0);
                lcd_data_edit((h / 10) + '0');
                lcd_data_edit((h % 10) + '0');
                lcd_data_edit(':');
                lcd_data_edit((m / 10) + '0');
                lcd_data_edit((m % 10) + '0');
                lcd_data_edit(':');
                lcd_data_edit((s / 10) + '0');
                lcd_data_edit((s % 10) + '0');
                IOWR(LED_BASE, 0, 0x00);
                IOWR(HEX_0_BASE, 0, decoder[s % 10]);
                IOWR(HEX_1_BASE, 0, decoder[s / 10]);
                IOWR(HEX_2_BASE, 0, decoder[m % 10]);
                IOWR(HEX_3_BASE, 0, decoder[m / 10]);
                IOWR(HEX_4_BASE, 0, decoder[h % 10]);
                IOWR(HEX_5_BASE, 0, decoder[h / 10]);

        }


}
if((IORD(SW_1_BASE, 0) == 1)&&(IORD(SW_0_BASE, 0) == 0))
{
        if(IORD(KEY_0_BASE, 0)==0)
        {
                m1++;
                if(m1==60)
                {
                        m1=0;
                }
        }
        if(IORD(KEY_1_BASE, 0)==0)
        {
                h1++;
                if(h1==24)
                {
                        h1=0;
                }
        }
```

```
                IOWR(HEX_0_BASE, 0, 0xFF);
                IOWR(HEX_1_BASE, 0, 0xFF);
                IOWR(HEX_2_BASE, 0, decoder[m1 % 10]);
                IOWR(HEX_3_BASE, 0, decoder[m1 / 10]);
                IOWR(HEX_4_BASE, 0, decoder[h1 % 10]);
                IOWR(HEX_5_BASE, 0, decoder[h1 / 10]);

                if (m1==m && h1==h)
                {
                        if(t==0)
                        {
                                c=5;
                                t=1;
                        }
                        switch (c)
                        {
                                case 1:
                                        IOWR(LED_BASE, 0, 0x18);
                                        break;
                                case 2:
                                        IOWR(LED_BASE, 0, 0x24);
                                        break;
                                case 3:
                                        IOWR(LED_BASE, 0, 0x42);
                                        break;
                                case 4:
                                        IOWR(LED_BASE, 0, 0x81);
                                        break;
                                case 5:
                                        IOWR(LED_BASE, 0, 0xFF);
                                        break;
                                default:
                                        break;
                        }
                        c--;
                }
                else t=0;
        }
        if(IORD(KEY_3_BASE, 0) == 0){
        s=20,h=14,m=50,d=23,mt=4,y=2024;
        command_edit(0x02); // Clear display
        lcd_data_edit((d / 10) + '0');
        lcd_data_edit((d % 10) + '0');
        lcd_data_edit('-');
        lcd_data_edit((mt / 10) + '0');
        lcd_data_edit((mt % 10) + '0');
        lcd_data_edit('-');
        lcd_data_edit((y / 1000) + '0');
        lcd_data_edit(((y % 1000) / 100) + '0');
        lcd_data_edit(((y % 100) / 10) + '0');
        lcd_data_edit((y % 10) + '0');
        command_edit(0xc0);
        lcd_data_edit((h / 10) + '0');
        lcd_data_edit((h % 10) + '0');
        lcd_data_edit(':');
        lcd_data_edit((m / 10) + '0');
        lcd_data_edit((m % 10) + '0');
        lcd_data_edit(':');
        lcd_data_edit((s / 10) + '0');
        lcd_data_edit((s % 10) + '0');
        IOWR(LED_BASE, 0, 0x00);
        IOWR(HEX_0_BASE, 0, decoder[s % 10]);
        IOWR(HEX_1_BASE, 0, decoder[s / 10]);
        IOWR(HEX_2_BASE, 0, decoder[m % 10]);
        IOWR(HEX_3_BASE, 0, decoder[m / 10]);
```

```
                    IOWR(HEX_4_BASE, 0, decoder[h % 10]);
                    IOWR(HEX_5_BASE, 0, decoder[h / 10]);

                    }
            }
            IOWR_ALTERA_AVALON_TIMER_STATUS(TIMER_0_BASE,
ALTERA_AVALON_TIMER_STATUS_TO_MSK);
}

void timer_Init()
{
    unsigned int period = 0;
    IOWR_ALTERA_AVALON_TIMER_CONTROL(TIMER_0_BASE,
ALTERA_AVALON_TIMER_CONTROL_STOP_MSK);
    period = 50000000 - 1;
    IOWR_ALTERA_AVALON_TIMER_PERIODL(TIMER_0_BASE, period);
    IOWR_ALTERA_AVALON_TIMER_PERIODL(TIMER_0_BASE, (period >> 16));
    IOWR_ALTERA_AVALON_TIMER_CONTROL(TIMER_0_BASE,
ALTERA_AVALON_TIMER_CONTROL_CONT_MSK |
    ALTERA_AVALON_TIMER_CONTROL_ITO_MSK | ALTERA_AVALON_TIMER_CONTROL_START_MSK);
}

int main()
{
    timer_Init();
    lcd_init();
    IOWR_ALTERA_AVALON_PIO_DATA(LCD_ON_BASE, 0x01);
    IOWR_ALTERA_AVALON_PIO_DATA(LCD_BLON_BASE, 0x01);
    alt_ic_isr_register(0, TIMER_0_IRQ, Timer_IRQ_Handle, (void*)0, (void*)0);
    while(1);
    return 0;
}
```

V. Báo cáo lỗi trong thực hành

Không có lỗi

VI. Kết luận bài thực hành

Chạy đúng với yêu cầu đề ra