ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA ĐIỆN – ĐIỆN TỬ

BỘ MÔN ĐIỆN TỬ

----------o0o----------

BÁO CÁO
THỰC TẬP TỐT NGHIỆP

GVHD: ThS. Võ Thị Thu Hồng

SVTH: Hoàng Minh Chiến

MSSV: 2112932

TP. HỒ CHÍ MINH, THÁNG 8 NĂM 2024

LỜI CẢM ƠN

Lời đầu tiên, em xin chân thành cảm ơn ThS. Võ Thị Thu Hồng, giảng viên bộ môn Điện tử vì đã giúp em có được kinh nghiệm và cách nhìn nhận thực tế về công việc sau khi hoàn thành thực tập. Cô đã tin tưởng và động viên em trong suốt quá trình thực hiện Thực tập tốt nghiệp. Sự nhiệt tình và quan tâm của cô là động lực giúp em vượt qua những giai đoạn khó khăn, tưởng chừng không thể theo kịp tiến độ của Thực tập tốt nghiệp. Những kiến thức và chia sẻ từ cô là tài sản quý báu mà em có được trên quãng đường đại học của mình.

Em cũng xin bày tỏ lòng biết ơn sâu sắc đến Trường Đại học Bách Khoa – ĐHQG Thành phố Hồ Chí Minh đã hỗ trợ và tạo điều kiện thuận lợi về phương tiện và vật chất cho việc nghiên cứu các dự án. Cảm ơn toàn thể quý Thầy, Cô khoa Điện – Điện tử đã tận tình truyền đạt cho em những kiến thức và kinh nghiệm hết sức quý báu trong suốt khoảng thời gian em theo học tại Khoa, Trường.

Dù bản thân đã thực hiện bằng tất cả sự nỗ lực để hoàn thành Thực tập tốt nghiệp này, song do kiến thức chuyên ngành còn nhiều hạn chế nên thiếu sót trong quá trình thực tập là điều không thể tránh khỏi. Em rất mong nhận được sự góp ý chân thành của quý Thầy, Cô để có thể hoàn thiện Thực tập tốt nghiệp một cách tốt hơn.

Trân trọng.

Tp. Hồ Chí Minh, ngày 23 tháng 08 năm 2024.

Hoàng Minh Chiến

MỤC LỤC

DANH SÁCH HÌNH MINH HỌA

DANH SÁCH BẢNG SỐ LIỆU

GIỚI THIỆU

1.1 Giới thiệu về công ty

1.2 Nhiệm vụ được giao thực tập

Nhiệm vụ và công việc được thực hiện trong Thực tập tốt nghiệp do giáo viên hướng dẫn giao, bao gồm các nội dung như sau:

Nội dung 1: Tìm hiểu, làm quen với Kit Altera DE2 và ngôn ngữ mô tả phần cứng Verilog HDL.

Nội dung 2: Viết chương trình giao tiếp với LED 7 đoạn và chương trình chia tần số bằng ngôn ngữ Verilog HDL.

Nội dung 3: Thiết kế mạch đồng hồ số đơn giản dùng ngôn ngữ Verilog HDL.

Nội dung 4: Mở rộng từ *Nội dung 3*: Mở rộng thiết kế đã được thực hiện từ *Nội dung 3*, thiết kế mạch định thì, từ đó thiết kế đồng hồ số điều chỉnh được thời gian bằng nút nhấn bằng ngôn ngữ Verilog HDL.

Nội dung 5: Viết chương trình giao tiếp với LCD trên Kit Altera DE2 bằng ngôn ngữ Verilog HDL.

1.3 Thời gian và lịch trình thực tập

Thời gian thực hiện Thực tập tốt nghiệp được mô tả theo lịch trình như sau:

- Thời gian thực tập: 2 tháng (từ ngày 15/06/2024 đến ngày 20/08/2024)
- Lịch trình thực tập:
  - **Tuần 1, 2:** Thực hiện Nội dung 1.
  - **Tuần 3 :** Thực hiện Nội dung 2.
  - **Tuần 4, 5:** Thực hiện Nội dung 3.
  - **Tuần 6, 7:** Thực hiện Nội dung 4.
  - **Tuần 8 :** Thực hiện Nội dung 5.

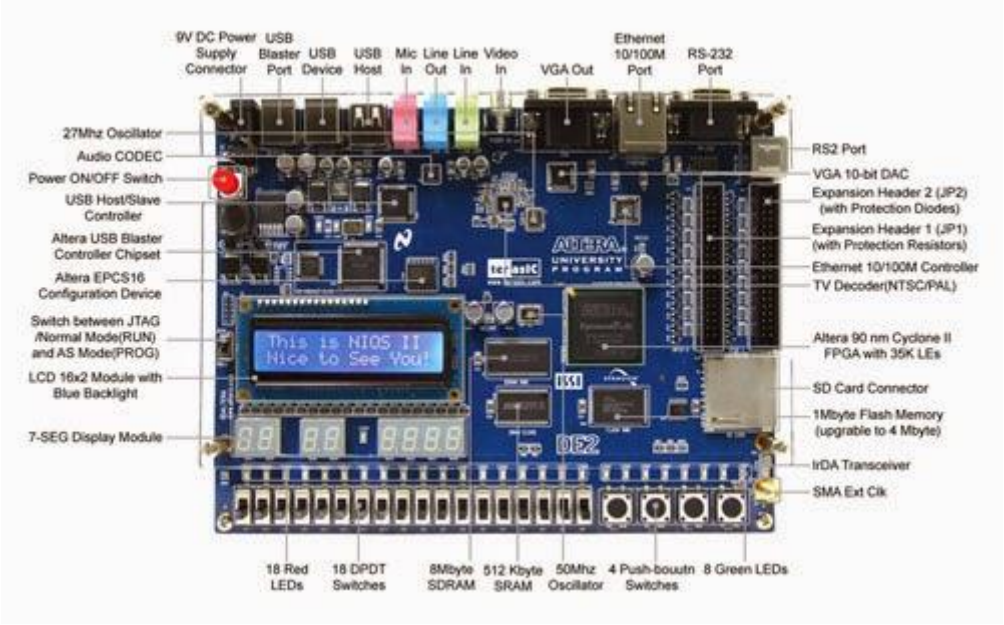
2. NỘI DUNG THỰC TẬP

Dưới đây là chi tiết các công việc thực tập trong quá trình thực hiện Thực tập tốt nghiệp:

2.1 Nội dung 1: Tìm hiểu, làm quen với Kit Altera DE2 và ngôn ngữ mô tả phần cứng Verilog HDL

2.1.1 Tìm hiểu và làm quen với Kit Altera DE2

Board Altera DE2 như Hình 2.1 được thiết kế với nhiều tính năng đa dạng dựa trên số lượng lớn các ngoại vi nhằm phục vụ cho các nghiên cứu khác nhau (ví dụ như: nghiên cứu và phát triển về các lĩnh vực luận lý logic (digital logic), tổ chức máy tính (computer organization) và FPGA). Trên Kit này, một FPGA CHIP học Cyclone II được tích hợp, và các ngõ vào/ra (I/O) của Chip được kết nối với tất cả các CHIP khác trên Kit Altera DE2 (như CHIP TV Decoder, Ethernet 10/100M Controller, SRAM,…) nhằm giúp cho người dùng có thể thay đổi các ứng dụng hay cấu hình nhằm mong muốn hướng đến một ứng dụng cụ thể.



Hình 2. 1 Kit Altera DE2

Board Altera DE2 cung cấp khá nhiều tính năng hỗ trợ cho việc nghiên cứu và phát triển, dưới đây là thông tin chi tiết của một board Altera DE2:

- FPGA:

  - Vi mạch FPGA Altera Cyclone II 2C35.

  - Vi mạch Altera Serial Configuration – EPCS16.

- Các thiết bị xuất nhập:

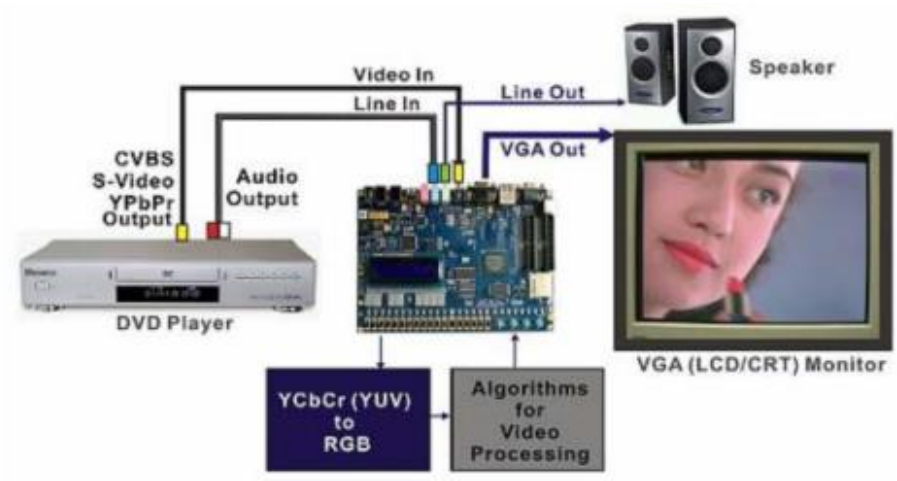
  - USB Blaster cho lập trình và điều khiển API của người dùng, hỗ trợ cả 2 chế độ lập trình JTAG và AS.

  - Bộ điều khiển Cổng 10/100 Ethernet.

  - Cổng VGA-out.

- Bộ giải mã TV và cổng nối TV-in.
- Bộ điều khiển USB Host/Slave với cổng USB kiểu A và kiểu B.
- Cổng nối PS/2 chuột/ bàn phím.
- Bộ giải mã/ mã hóa âm thanh 24-bit chất lượng đĩa quang với jack cắm line-in, line-out và microphone.
- 2 Header mở rộng 40-pin với lớp bảo vệ diode.
- Cổng giao tiếp RS-232 và cổng nối tiếp 9-pin.
- Cổng giao tiếp hồng ngoại.

- Bộ nhớ:
  - SRAM 512-Kbyte.
  - SDRAM 8-Mbyte.
  - Bộ nhớ Flash 4-Mbyte (1 số mạch là 1-Mbyte).
  - Khe SD card.
- Switch, các đèn LED, LCD, xung clock:
  - 4 nút nhấn, 18 nút gạt.
  - 18 LED đỏ, 9 LED xanh, 8 LED 7 đoạn.
  - LCD 16x2.
  - Bộ dao động với tần số 50-MHz và 27-MHz.
- Một số điểm lưu ý trong quá trình sử dụng Kit Altera DE2 được mô tả như sau:
  - Chỉ sử dụng nguồn 12 V DC.
  - Chốt sử dụng cho lập trình (RUN/PROG Switch for JTAG/AS Mdes) nên được thiết lập ở chế độ “RUN” trước khi sử dụng cho các thực nghiệm.
  - Chỉ dùng cổng USB Blaster (Sát bên cổng nguồn) để kết nối máy tính và lập trình.
- Một vài ứng dụng của Board Altera DE2:
  - Ứng dụng làm TV box.

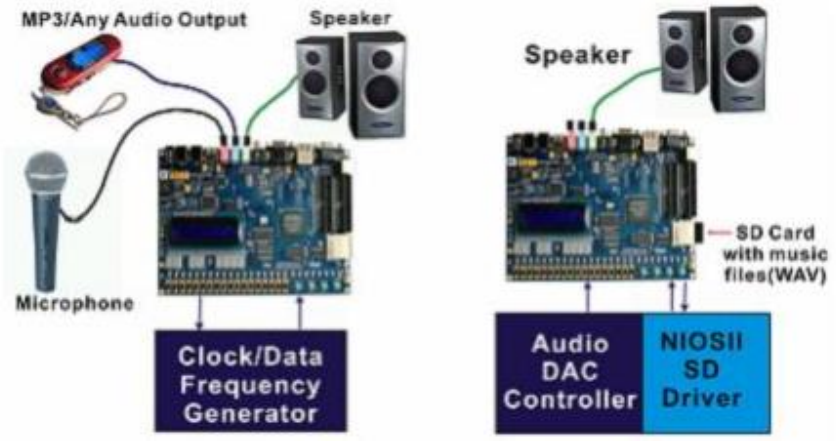


Hình 2. 2 TV box

- Chương trình vẽ bằng chuột USB (paintbrush)

Hình 2. 3 Chương trình vẽ (paintbrush)

- Mát hát Karaoke và máy nghe nhạc SD.



Hình 2. 4 Máy hát Karaoke và máy nghe nhạc từ card SD

2.1.2 Tìm hiểu cơ bản về ngôn ngữ mô tả phần cứng Verilog HDL

- Sơ lược về Verilog HDL:

- Ngôn ngữ mô tả phần cứng – Hardware Description Language (HDL).

- Phát triển từ năm 1984.

- Theo chuẩn 1364 IEEE: 12/2005.

- Giới hạn cơ bản của Verilog HDL: Chỉ đặc tả hệ thống số.
- Các mức mô tả Verilog HDL:
  - Behavioral level (Mức độ hành vi).
  - RTL level (Mức độ RTL).
  - Gate level (Mức **độ** cổng).
  - Layout (VLSI).
- Quy định sử dụng:
  - Có thể dùng {[A-Z], [a-z], [0-9], _, $,…}
  - Không bắt buộc đầu $ hay [0-9]

    myidentifier, m_y_identifier, _myidentifier$,…
  - Phân biệt chữ hoa và thường

    myid ≠ Myid ,…
- Comments – Trích dẫn:

```
//      Dòng chú thích
 /*
        Chú thích nhiều dòng
 */
```

- Các giá trị Verilog HDL:
  - 0 mức logic thấp hay điều kiện sai (false)
  - 1 mức logic cao hay điều kiên đúng (true)
  - x mức logic khồn xác định (unknown level)
  - z mức logic tổng trở cao (hi-Z)
- Số trong Verilog HDL:
  - Có thể chèn "_" để dễ đọc

    12'b 000_111_010_110

    12b 000111010110

    12'o 07_24
  - Mở rộng BIT (MSB: Most Significant Bit)

    MSB = 0, x or z ⇒ mở rộng chính nó

    Vi dụ: 4'b x1 = 4'b xx_x1

    MSB = 1 ⇒ mở rộng 0

4’b 1x = 4’b 00_1x

- Nếu hệ số không có, xem như hệ số 10

15 = <size>’d 15

- Đường nối (nets):
  - Xem như các dây phần cứng lái bởi logic.
  - Bằng Z khi không nối.
  - Các loại khác nhau của nets:

wire

wand (wire-AND)

wor (wire-OR)

tri (tri-state)

- Các thanh ghi – Registers:
  - Là các biến lưu giá trị
  - Không có thật trong phần cứng thật nhưng …
  - … Phần cứng thật có thể bổ sung các thanh ghi.
  - Chỉ có 1 loại: reg

```
reg A, C; // Khai báo
// Phép gán luôn được thực thi bên trong một procedure
A = 1;
C = A; // C sẽ có giá trị logic 1
A = 0; // C vẫn là 1
C = 0; // Bây giờ C là 0
```

  - Giá trị thanh ghi được ghi mới hiện hữu.
- Vectors:
  - Các Vector Bus

```
wire [3:0] busA;
reg [1:4] busB;
reg [1:0] busC;
```

  - Số bên trái có trọng số lớn nhất (MSB – Maximum Significant Bit).
  - Quản lý theo từng bit.

```
busC[1] = busA[2];
busC[0] = busA[1];
```

  - Gán vector theo vị trí.

```
busB[1] = busA[3];
busB[2] = busA[2];
```

```
busB[3] = busA[1];
busB[4] = busA[0];
```

- Loại dữ liệu và số nguyên:

  - Khai báo

```
integer I, k;
real r;
```

  - Dùng như thanh ghi (bên trong procedures)

```
i = 1; // Gán bên trong procedures
r = 2.9;
k = r; // k được làm tròn 3
```

  - Các số nguyên không có giá trị ban đầu

  - Các số thực ban đầu là 0.0

- Loại dữ liệu thời gian:

  - Loại dữ liệu đặc biệt để đo thời gian mô phỏng

  - Khai báo

```
time my_time;
```

  - Dùng bên trong procedure

```
my_time = $time; // Lấy giá trị hiện hành
```

  - Thời gian chạy mô phỏng không phải thời gian thực

- Mảng – Arrays:

  - Cú pháp:

```
integer count [1:5]; // 5 số nguyên
reg vả [-15:16]; // 32 thanh ghi 1-bit
reg [7:0] mem [0:1023]; // 1024 thanh ghi 8-bit
```

  - Truy xuất các phần tử của mảng

```
mem [10] = 8'b 1010_1010;
reg [7:0] temp;
…
temp = mem [10];
var[6] = temp[2];
```

  - Giới hạn: Không thể truy xuất một phần mảng hoặc toàn mảng một lúc

```
var [2:9] = ???; // SAI
```

var = ???; // SAI

- Không có mảng đa chiều

reg var [1:10] [1:100]; // SAI

- Mảng không làm việc với loại dữ liệu thực

real r [1:10]; // SAI

- Chuỗi – String:

- Bổ sung các thanh ghi:

reg [8*13:1] string_val; // Có thể giữ tối đa 13 ký tự

…

string_val = “Hello World”;

string_val = “hello”; // Các Byte trọng số cao được điền 0

string_val = “I am overflowed”; // “I” không gán được

- Escaped chars:

\n	dòng mới
\t	khoảng TAB
%%	%
\\	\
\””	Khoảng trắng

- Toán tử logic:

&& → AND logic

|| → OR logic

! → NOT logic

- Toán tử trả về 1 trong các giá trị: 0, 1 or x

- Kết quả là một giá trị: 0, 1 or x

A = 6;

B = 0;

C = x;

A && B → 1 && 0 → 0

A || !B → 1 || 1 → 1

C || B → x || 0 → x

- Toán tử trên bit – bitwise:

& → bitwise AND

| → bitwise OR

~ → bitwise NOT

^ → bitwise XOR

~^ hoặc ^~ → bitwise XNOR

- Các toán tử rút gọn:

& → AND

| → OR

^ → XOR

~& → NAND

~| → NOR

~^ or ^~ → XNOR

- Một phép toán nhiều bit → Kết quả là 1 bit

```
a = 4'b1001;
…
c = |a; // c = 1|0|0|1 = 1
```

- Toán tử dịch:

>> → dịch phải

<< → dịch trái

- Kết quả có cùng kích cỡ với phép toán ban đầu, 0 sẽ được chèn vào

```
a = 4'b1010;
…
d = a >> 2; // d = 0010
c = a << 1; // c = 0100
```

- Toán tử gộp:

- {op1, op2,…} → gộp op1, op2,… thành một số

- Phép toán phải định kích cỡ.

```
reg a;
reg [2:0] b, c;
…
a = 1'b 1;
b = 3'b 010;
c = 3'b 101;
catx = {a, b, c}; // catx = 1_010_101
caty = {b, 2'b11, a}; // caty = 010_11_1
catz = {b, 1};  // SAI
```

- Gộp đa tầng

catr = {4{a}, b, 2{c}}; // catr = 1111_010_101101

- Toán tử quan hệ:

> → lớn hơn

< → nhỏ hơn

>= → lớn hơn hoặc bằng

<= → bé hơn hoặc bằng

- Kết quả chỉ là một bit: 0, 1 or x

1 > 0 → 1

4'b1x1 <= 0 → x

10 < z → x

- Toán tử bằng:

== → bằng logic

!= → không bằng logic

=== → bằng trường hợp

!== → không bằng trường hợp

4'b 1z0x == 4'b 1z0x → x

4'b 1z0x != 4'b 1z0x → x

4'b 1z0x === 4'b 1z0x → 1

4'b 1z0x !== 4'b 1z0x → 0

- Toán tử điều kiện:

- Cú pháp:

assign output = (condition_expresion) ? true_expr : false_expr

- Như MUX 2-1

assign Y = (sel == 1) ? A : B;

assign Y = (sel) ? A : B;

- Toán tử toán học

+ (Cộng), - (Trừ), * (Nhân), / (Chia), % (Phần dư), ** (Lũy thừa)

- Nếu bất kỳ toán hạn nào là x, kết quả sẽ là x

- Các thanh ghi âm:

Thanh ghi có thể gán với giá trị âm nhưng được xử lý như không dấu

reg [15:0] regA;

…

regA = -4'd12; // được lưu như $2^{16} - 12 = 65524$

c = regA/3; // c có giá trị 21861

- Các thanh ghi âm:

Có thể gán các giá trị âm

Việc xử lý khác nhau dực trên tài liệu cơ bản (base specification) hoặc không (no base speccification)

reg [15:0] regA;

integer intA;

…

intA = -12/3; // evaluates to -4 (no base spec)

intA = -'d12/3; // evaluates to 1431655761 (base spec)

- Mô hình hành vi – Procedure:
  - Procedures = phần code được thực thi một cách tuần tự
  - Diễn tả Procedural = diễn tả bên trong một procedure (chúng thực thi tuần tự)
  - Kiểu diễn tả khác của MUX 2-1:

```
begin
    if(sel == 0)
        Y = B;
    else
        Y = A;
end
```

  - Module có thể chứa bất cứ số lượng procedure
  - Procedures thực thi song song (liên hệ với procedure khác) và …
  - … Có thể diễn tả trong hai loại khối:

initial → thực thi chỉ một lần

always → thực thi mãi (đến khi mô phỏng hoàn thành)

- Khối "initial":
  - Bắt đầu thực thi ở thời gian 0 và hoàn thành khi diễn tả cuối cùng được thực thi

```
module nothing;
    initial begin
```

```
                #20;
                $display("I'm first"); // Sẽ hiển thị ở Thời gian mô phỏng = 20
                end
        initial begin
                #50;
                $display("Really?"); // Sẽ hiển thị ở Thời gian mô phỏng = 50
                end
    endmodule
```

- Khối "always":

  - Bắt đầu thực thi ở thời gian 0 và kết thúc khi thời gian mô phỏng hoàn thành.

- Sự kiện – Events:

  - @ // Sensitivity list (list biến nhạy)

```
    always @(signal1 or signal2 or …) begin
        …
    end
// Thực thi khi bất cứ tín hiệu bất kì thay đổi
    always @(posedge clk) begin // Positive edge (cạnh lên)
        …
    end
// Thực thi khi bất cứ tín hiệu clk nào thay đổi từ 0 lên 1
    always @(negedge clk) begin // Negative edge (cạnh xuống)
        …
    end
// Thực thi khi bất cứ tín hiệu clk nào thay đổi từ 1 xuống 0
```

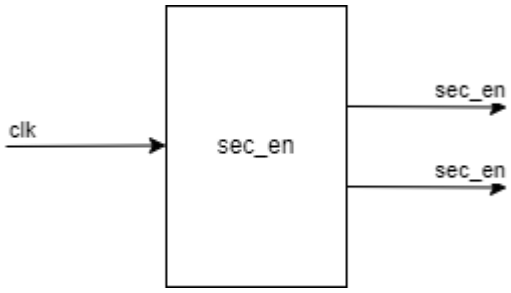
  - wait (expr)

```
    always @(…) begin
        wait (ctrl)
        #10 cnt = cnt + 1;
        #10 cnt = cnt + 2;
        end
```

2.2 Nội dung 2: Viết chương trình giao tiếp LED 7 đoạn và chương trình chia tần số bằng ngôn ngữ Verilog HDL

2.2.1 Chương trình chia tần số

- Sơ đồ khối của mạch chia tần số:



Hình 2.5 Sơ đồ khối của mạch chia tần số

Đối với Kit Altera DE2 thì tần số hoạt động mặc định của nó là 50 MHz, tùy vào mục đích sử dụng mà người lập trình sẽ chia tần số 50 MHz xuống những tần số nhỏ hơn, như trong mạch chia tần số được trình bày ở trên thì mạch này thực hiện chia ra 2 tần số là 1 Hz, 5 Hz.

Để thuận tiện cho việc tính toán chia tần số, ta áp dụng công thức xác định thời điểm chuyển trạng thái của tần số cần tìm từ tần số cần chia ban đầu, như sau:

$$i = \frac{Tần\ số\ ban\ đầu}{2\ \times Tần\ số\ cần\ thu\ được\ sau\ khi\ chia}$$

Trong đó:

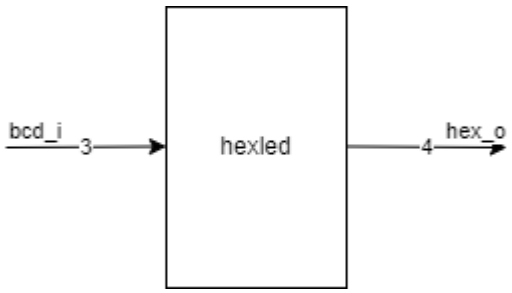
i: Số chu kì mà Tần số ban đầu nhận được cho đến khi Tần số mong muốn nhận được sau khi chia tần chuyển trạng thái.

Ví dụ: Cần chia tần số có giá trị là 50 MHz xuống còn 1 Hz thì cần phải qua 25000000 chu kì của tần số 50 MHz thì tần số 1 Hz mới chuyển trạng thái. Hay có thể hiểu để có được một chu kỳ của tần số 1 Hz thì vẫn thực hiện 50000000 chu kì của tần số 50 MHz.

2.2.2 Chương trình giao tiếp LED 7 đoạn

2.2.2.1 Sử dụng Switch để điều khiển hiển thị LED 7 đoạn dùng ngôn ngữ Verilog HDL

- Sơ đồ khối của mạch giải mã LED 7 đoạn:



Hình 2. 6 Sơ đồ khối của mạch giải mã LED 7 đoạn

Mạch giải mã LED 7 đoạn có ngõ vào là số nhị phân 4 bit, được đặt tên là "bcd_i" và ngõ ra là số nhị phân 7 bit, được đặt tên là "hex_o". Mạch giải mã LED 7 đoạn sẽ xác định giá trị của 4 bit bit ngõ vào bcd, từ đó đưa ra giá trị 7 bit ngõ ra hex tương ứng theo bảng trạng thái được nêu bên dưới, sau đó đưa ra LED 7 đoạn để hiển thị.

- Bảng trạng thái hoạt động của LED 7 đoạn loại Anode chung:

LED 7 đoạn trên Kit Altera DE2 được cấu hình theo kiểu Anode chung, có bảng trạng thái như sau:

Số	**Ngõ vào số nhị phân**				**Ngõ ra là mã 7 đoạn**						
	so_gm				**segments**						
	(3)	(2)	(1)	(0)	g	f	e	d	c	b	a
0	0	0	0	0	1	0	0	0	0	0	0
1	0	0	0	1	1	1	1	1	0	0	1
2	0	0	1	0	0	1	0	0	1	0	0
3	0	0	1	1	0	1	1	0	0	0	0
4	0	1	0	0	0	0	1	1	0	0	1
5	0	1	0	1	0	0	1	0	0	1	0
6	0	1	1	0	0	0	0	0	0	1	0
7	0	1	1	1	1	1	1	1	0	0	0
8	1	0	0	0	0	0	0	0	0	0	0
9	1	0	0	1	0	0	1	0	0	0	0
10	1	0	1	0	0	0	0	1	0	0	0
11	1	0	1	1	0	0	0	0	0	1	1
12	1	1	0	0	1	0	0	0	1	1	0
13	1	1	0	1	0	1	0	0	0	0	1
14	1	1	1	0	0	0	0	0	1	1	0
15	1	1	1	1	0	0	0	1	1	1	0

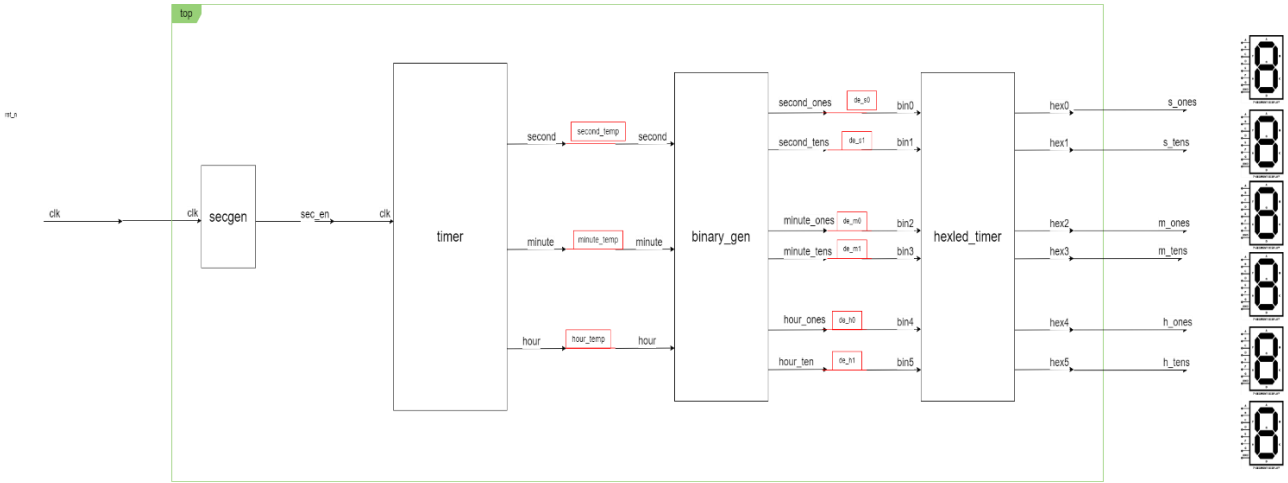
Bảng 2. 1 Bảng trạng thái hoạt động của LED 7 đoạn loại Anode chung

Từ bảng trạng thái ở trên, ta sẽ có: ví dụ, khi ngõ vào bcd_i có giá trị là 0100 thì qua bộ giải mã LED 7 đoạn sẽ cho ngõ ra hex_o có giá trị là 001_1001, đưa ra LED 7 đoạn để hiển thị số 4. Hay khi ngõ vào bcd_i có giá trị 1101 thì qua bộ giải mã LED 7 đoạn sẽ cho ngõ ra hex_o có giá trị là 010_0001, đưa ra LED 7 đoạn để hiển thị…

2.3 Nội dung 3: Thiết kế mạch đồng hồ đơn giản dùng ngôn ngữ Verilog HDL

2.3.1 Thiết kế mạch đồng hồ số đơn giản

- Yêu cầu đề ra: Đồng hồ bấm giờ phút giây sẽ đếm lên khi có tín hiệu cho phép (watch counter).
- Sơ đồ khối của mạch đồng hồ số đơn giản:

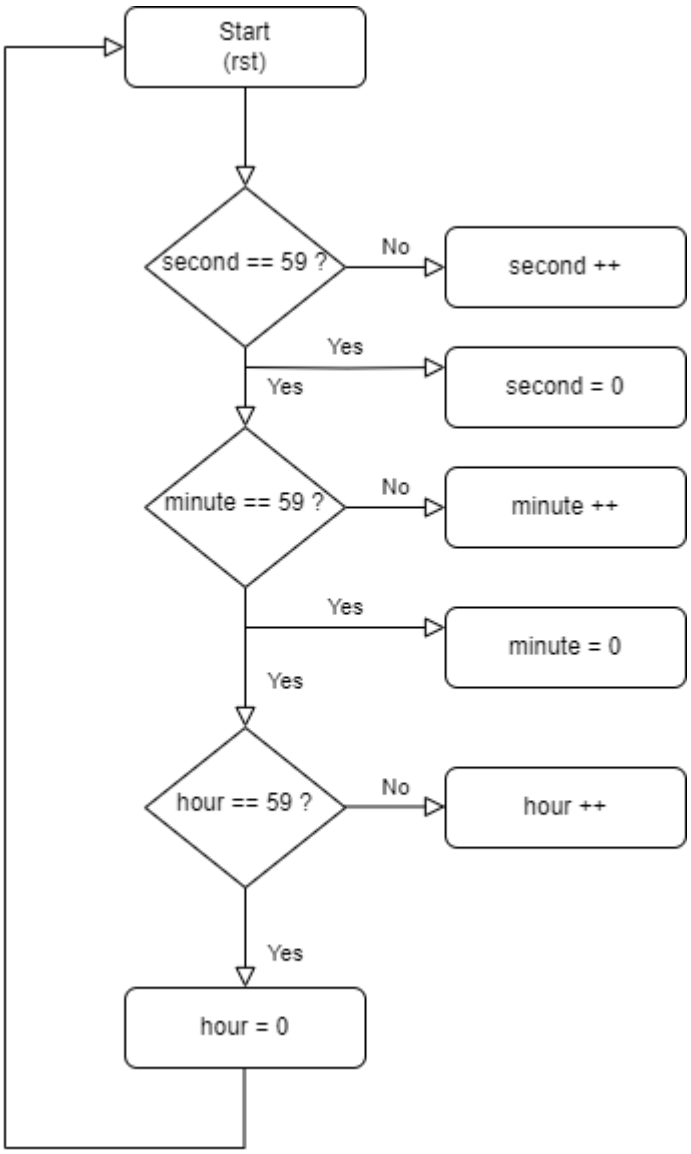


Hình 2.7 Sơ đồ khối của mạch đồng hồ đơn giản

Đối với mạch đồng hồ đơn giản này sẽ sử dụng lại các khối đã được trình bày ở các phần trên như khối secgen để tạo ra tần số hoạt động 1 Hz, khối hexled_timer từ việc kết hợp các khối hexled lại với nhau để hiển thị ra LED 7 đoạn giờ, phút, giây tương ứng.

Trong mạch này, có khối timer và binary_gen là mới. Riêng khối binary_gen là khối được tạo ra từ việc kết hợp nhiều khối bin2bcd lại với nhau, có chức năng chuyển đổi số nhị phân sang số BCD làm ngõ vào cho khối giải mã LED 7 đoạn xử lý và đưa kết quả ra hiển thị lên LED 7 đoạn tương ứng.

- Lưu đồ hoạt động của khối định thì (timer):



Hình 2.8 Lưu đồ hoạt động của khối định thì

Trong khối timer, ngõ vào có một tín hiệu clock với tần số hoạt động được sử dụng là 1 Hz nhận từ khối secgen đưa vào và một tín hiệu reset tích cực mức thấp có chức năng reset hệ thống về giá trị không. Khi tín hiệu reset = 1, lúc này hệ thống thoát khỏi trạng thái reset và bắt đầu đếm thời gian. Do hệ thống được thiết kế hoạt động ở tấn số là 1 Hz nên cứ sao mỗi 1s thì timer sẽ được động đếm tăng thêm một.

Khi timer đếm đến giá trị giờ là 59, giá trị phút là 59, giá trị giây là 59 thì trạng thái kế tiếp của giờ, phút và giây đều quay về giá trị 0, tiếp tục đếm.

Khi timer đếm đến giá trị phút là 59, giá trị giây bằng 59 thì trạng thái kế tiếp của phút, giây sẽ được quay về giá trị 0 và giá trị giờ sẽ được tăng thêm 1, tiếp tục đếm.

Khi timer đếm đến giá trị giây là 59 thì trạng thái kế tiếp của giây sẽ được quay về giá trị 0 và giá trị của phút sẽ được tăng thêm 1, tiếp tục đếm.

Khi timer đếm mà giá trị của giây chưa đến giá trị 59 thì giá trị của giây sẽ tiếp tục đếm, sau mỗi 1s giá trị của giây sẽ được tăng thêm 1.

Khối timer sẽ tiếp được đếm cho đến khi có tín hiệu rst_n = 0 thì nó sẽ ngừng đếm và các các giá trị giờ, phút, giây được reset về giá trị 0.

Do các giá trị của giờ, phút và giây có giá trị tối đa là 59 nên cần 6 bit nhị phân để biểu diễn, các bit nhị phân này chính là ngõ ra của khối timer, đồng thời, đây cũng chính là ngõ vào của khối chuyển đổi từ số nhị phân sang số BCD.

- Sơ đồ khối của khối giải mã số nhị phân sang số BCD:

Hình 2.9 Sơ đồ khối của khối giải mã từ số nhị phân sang số BCD

Trong khối binary_gen có ba khối bin2bcd, mỗi khối này biểu diễn cho một đối tượng giờ, phút hoặc giây. Khi nhận dữ liệu từ ngõ ra của khối timer gồm 6 bit nhị phân, khối giải mã số nhị phân sang số BCD sẽ xử lý và đưa ra hai ngõ ra là units có giá trị là 4 bit nhị phân và tens cũng có giá trị là 4 bit nhị phân cho mỗi đối tượng giờ, phút hoặc giây. Ngõ ra units đại diện cho hàng đơn vị và ngõ ra tens đại diện cho hàng chục của mỗi tượng, ví dụ như chục giờ, đơn vị giờ; chục phút, đơn vị phút và chục giây, đơn vị giây. Sau khi xử lý xong các giá trị này sẽ được đưa đến khối seven_seg_top để tiếp tục xử lý ở bước tiếp theo.

- Nguyên lý chuyển đổi số nhị phân sang số BCD:

Nguyên lý chuyển đổi số nhị phân n bit sang số BCD bằng cách dịch và cộng với 3 (Shift and add 3) như sau:

Bước 1: Sắp xếp số BCD và số nhị phân thành một dãy, số BCD nằm trước số nhị phân.

Bước 2: Dịch số BCD và số nhị phân được sắp xếp ở *Bước 1* sang trái một bit.

Bước 3: Sau mỗi lần dịch nếu bất kỳ số BCD nào lớn hơn hoặc bằng 5 thì cộng thêm với 3.

Tiếp tục thực hiện tiếp *Bước 2* và ở lần dịch thứ n thì không cần thực hiện *Bước 3* nữa và kết quả đó chính là số BCD.

Đối với mạch đồng hồ số đơn giản này thì giá trị lớn nhất trong các giá trị giờ, phút, giây lớn nhất có thể đạt được là 59 nên thực hiện tính toán chuyển đổi số nhị phân có giá trị là 59 sang số BCD, việc tính toán như vậy sẽ giúp cho việc lập trình trở nên dễ dàng vì nó đúng cho tất cả các trường hợp khác.

Tính toán: Thực hiện chuyển đổi số nhị phân 6 bit là 11_1011 B sang số BCD thì thực hiện 6 lần dịch như bảng sau:

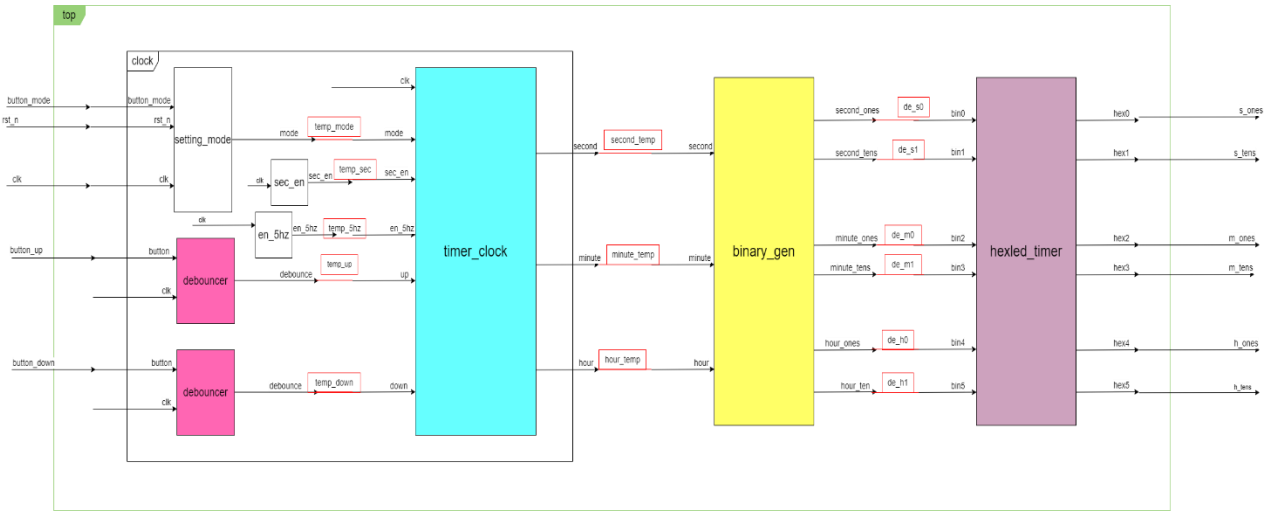
Operation	**Tens**				**Units**				**Binary**					
Start									1	1	1	0	1	1
Shift 1								1	1	1	0	1	1	
Shift 2							1	1	1	0	1	1		
Shift 3						1	1	1	0	1	1			
Add 3					1	0	1	0	0	1	1			
Shift 4				1	0	1	0	0	1	1				
Shift 5			1	0	1	0	0	1	1					
Add 3			1	0	1	1	0	0	1					
Shift 6		1	0	1	1	0	0	1						
BCD		1	0	1	1	0	0	1						

Bảng 2. 2 Bảng chuyển đổi số nhị phân 6 bit 11_1011 B sang số BCD

Để đơn giản hóa việc quan sát bảng chuyển đổi số nhị phân 6 bit sang số BCD, một số ô không quan trọng sẽ không được điền giá trị, các ô này mặc định được hiểu mang giá trị 0.

2.4 Nội dung 4: Mở rộng từ *Nội dung 3:* Mở rộng thiết kế đã thực hiện từ *Nội dung 3*, thiết kế mạch định thì, từ đó thiết kế đồng hồ điều chỉnh thời gian bằng nút nhấn bằng ngôn ngữ Verilog HDL

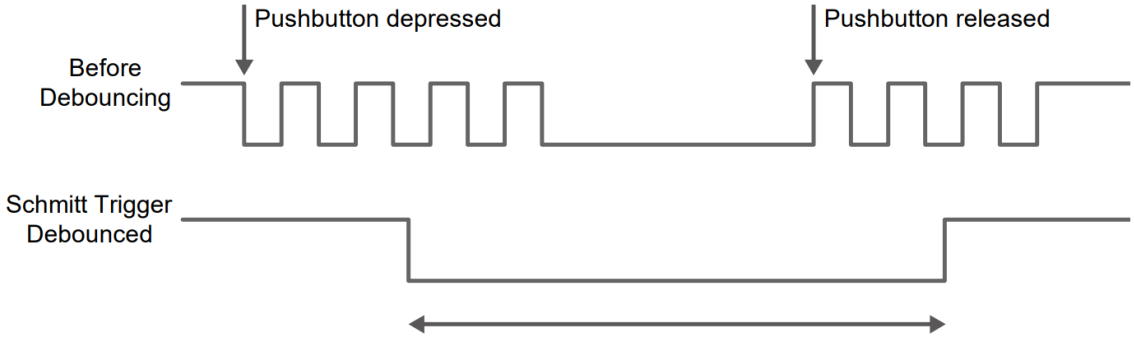
- Sơ đồ khối của mạch đồng hồ số điều chỉnh được thời gian bằng nút nhấn:



Hình 2.10 Sơ đồ khối của mạch đồng hồ số điều chỉnh được thời gian bằng nút nhấn

Đối với mạch đồng hồ số điều chỉnh thời gian bằng nút nhấn nên ta cần phải thực hiện chương trình chống dội cho nút nhấn.

- Dạng sóng chống dội cho nút nhấn:



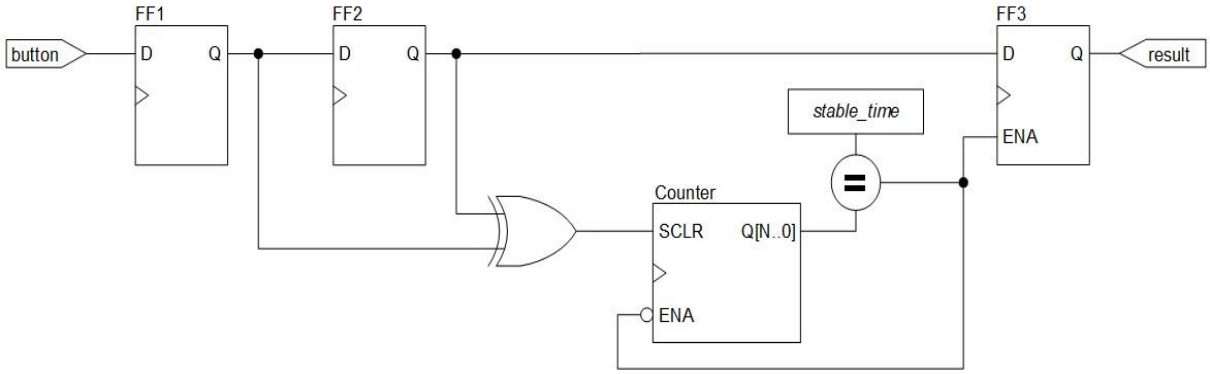
Hình 2.11 Dạng sóng chống dội cho nút nhấn

Từ Hình 2.17 mô tả dạng sóng chống dội cho nút nhấn, với clk có tần số hoạt động là 50 MHz và chu kỳ 20 ns.

Tín hiệu vào bị dội, sẽ đưa qua mạch chống dội.

Tín hiệu ra ổn định – thời gian ngõ ra ở mức logic “1” bằng thời gian nhấn phím. Ví dụ: nhấn giữ ổn định đúng 1s thì độ rộng xung đúng bằng 1 s, nhấn giữ 10 s thì độ rộng đúng bằng 10 s,…

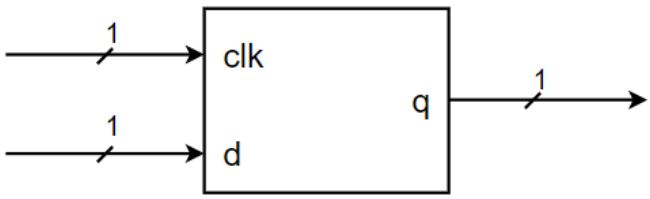
- Mạch chống dội cho nút nhấn:



Hình 2.12 Mạch chống dội cho nút nhấn

Vì thời gian ngõ ra bằng đúng thời gian nhấn phím nên cần phải có thêm một mạch làm hẹp xung để giải quyết vấn đề dù cho nhấn và giữ bao nhiêu lâu đi nữa thì mạch đếm chỉ tăng đúng 1 giá trị. Muốn tăng lần 2 thì phải nhả phím rồi nhấn lần 2.

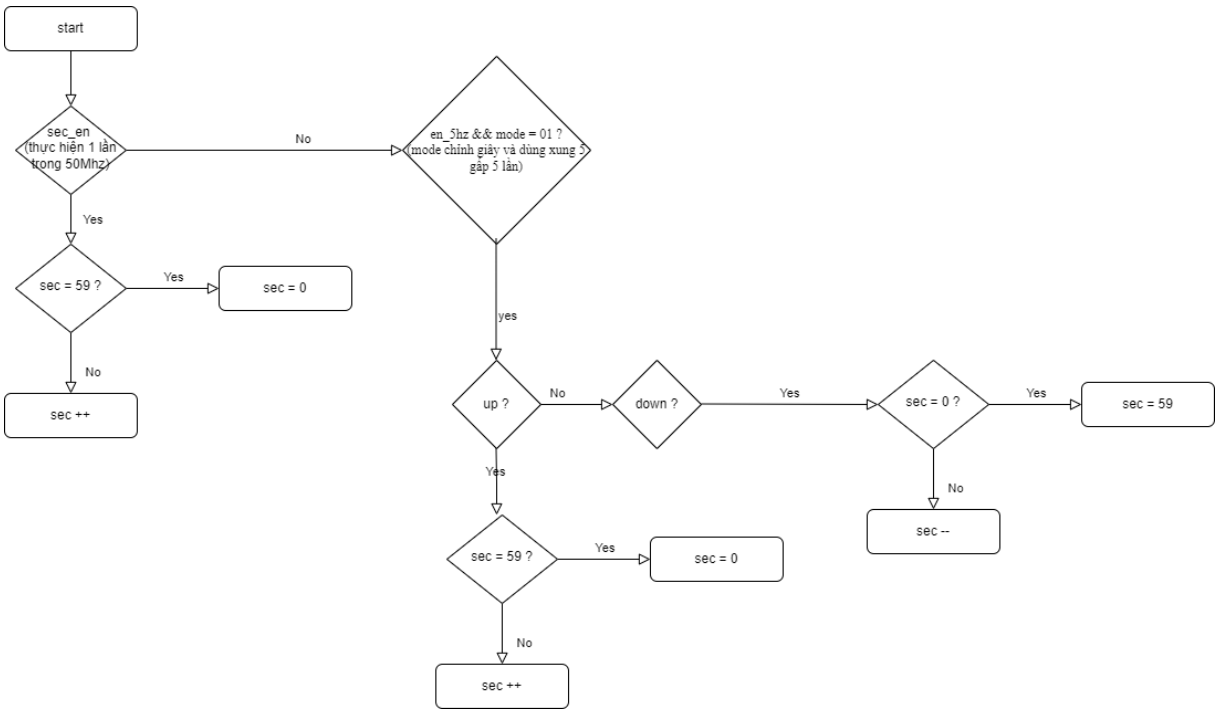
- Sơ đồ khối của khối làm hẹp xung:



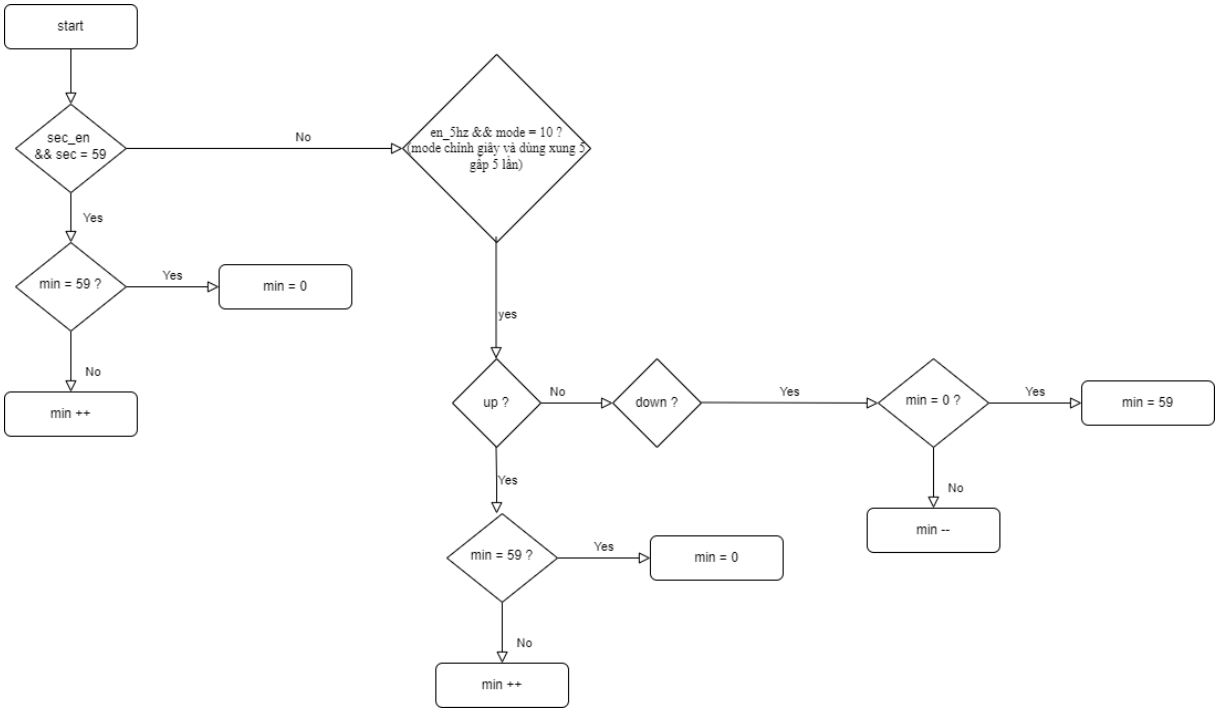
Hình 2.13 Sơ đồ khối của khối làm hẹp xung

Mạch làm hẹp xung có ngõ vào là xung clk và d, ngõ ra là q. Trong đó, hai ngõ vào là clk và d được kết nối đến ngõ vào của một flip – flop D, ngõ ra q có giá trị là (~ (ngõ ra của D flip – flop) & d). Tín hiệu ngõ vào d có độ rộng tùy ý nhưng ngõ ra có độ rộng đúng bằng xung 1 chu kì của xung clk.

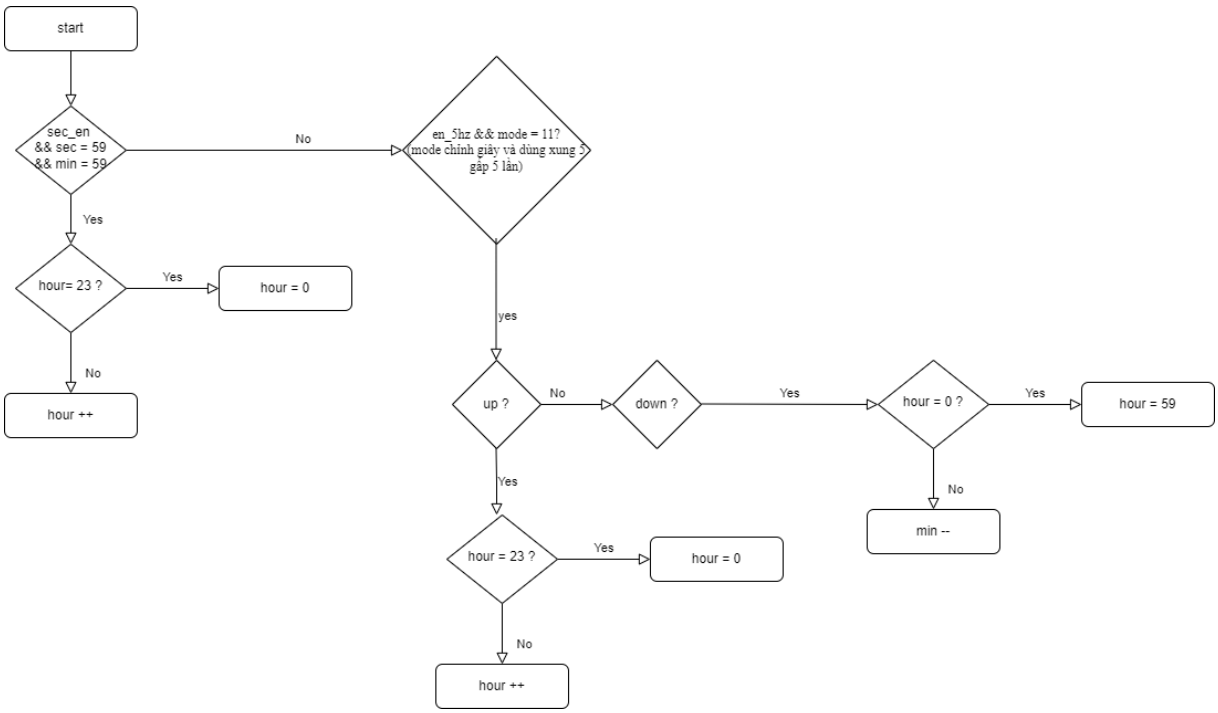
- Lưu đồ hoạt động của khối định thì:



Hình 2.14 Lưu đồ hoạt động của đơn vị giây



Hình 2.15 Lưu đồ hoạt động của đơn vị phút

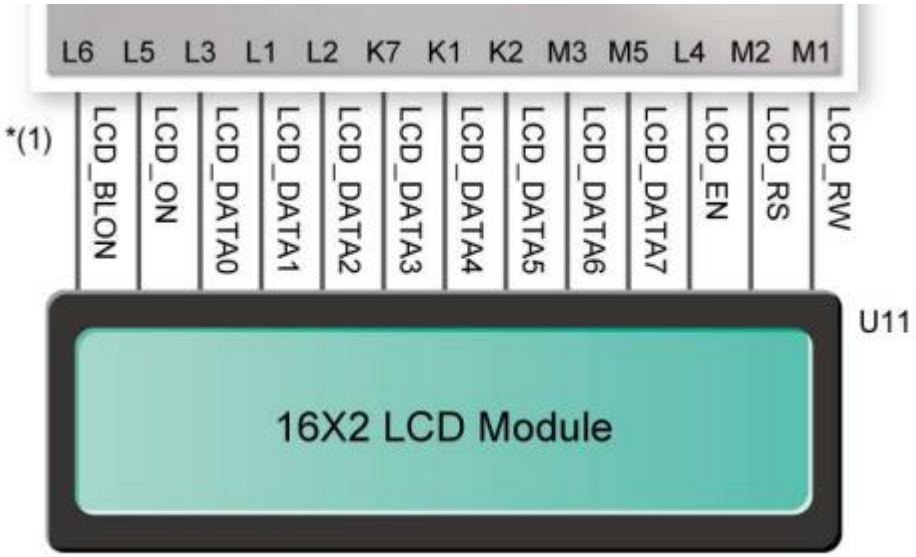


Hình 2.16 Lưu đồ hoạt động của đơn vị giờ

2.5 Nội dung 5: Viết chương trình giao tiếp LCD trên Kit Altera DE2 bằng ngôn ngữ Verilog HDL

- Giao tiếp với LCD 16x2 trên Kit Altera DE2:

Sau khi tham khảo datasheet LCD 16x2, đa phần sử dụng chip điều khiển HD44780 với sơ đồ chân như sau:



Hình 2.17 LCD 16x2

LCD yêu cầu gửi các mã HEX code dưới dạng các command, trong đó bao gồm mã HEX code khởi động và mã HEX code để thực hiện ghi/xóa nội dung.

Hình 2.18 Mã ký tự và dạng hiển thị tương ứng

- Các lệnh cho module LCD 16x2

Module LCD 16x2 có một bộ hướng dẫn lệnh đặt trước. Mỗi lệnh sẽ làm cho module thực hiện một nhiệm vị cụ thể. Các lệnh thường được sử dụng và chức năng của chúng được đưa ra bảng dưới đây:

Lệnh	Chức năng
0F	LCD bật, con trỏ bật, con trỏ nhấp nháy bật
01	Xóa toàn màn hình
02	Quay về màn hình chính
04	Giảm con trỏ
06	Tăng con trỏ
0E	Màn hình bật, con trỏ nhấp nháy tắt
80	Bắt con trỏ trở về vị trí đầu tiên của hàng 1
C0	Bắt con trỏ trở về vị trí đầu tiên của hàng 2

38	Sử dụng 2 hàng và ma trận 5x7
83	Con trỏ hàng 1 vị trí 3
3C	Kích hoạt dòng 2
08	Tắt màn hình hiển thị và con trỏ
C1	Nhảy đến dòng 2 vị trí 1
OC	Bật màn hình hiển thị, tắt con trỏ
C2	Nhảy đến hàng 2, vị trí 2

Bảng 2. 3 Các lệnh và chức năng của chúng trên LCD 16x2

- Khởi tạo LCD:

Các bước phải được thực hiện để khởi tạo màn hình LCD được đưa ra dưới đây và các bước này là phổ biến cho hầu hết các ứng dụng.

Bước 1: Gửi 38H đến dòng dữ liệu 8 bit để khởi tạo.

Bước 2: Gửi 0FH để bật LCD, con trỏ BẬT và con trỏ nhấp nháy ON.

Bước 3: Gửi 06H để tăng vị trí con trỏ.

Bước 4: Gửi 01H để xóa màn hình và trả về con trỏ.

- Đưa dữ liệu vào LCD:

Các bước để gửi dữ liệu đến module LCD được đưa ra dưới đây. Module LCD có các chân RS, R/W và E. Chính trạng thái logic của các chân này làm cho module xác định xem đầu vào dữ liệu đã cho là lệnh hay dữ liệu được hiển thị.

Đặt R/W mức thấp.

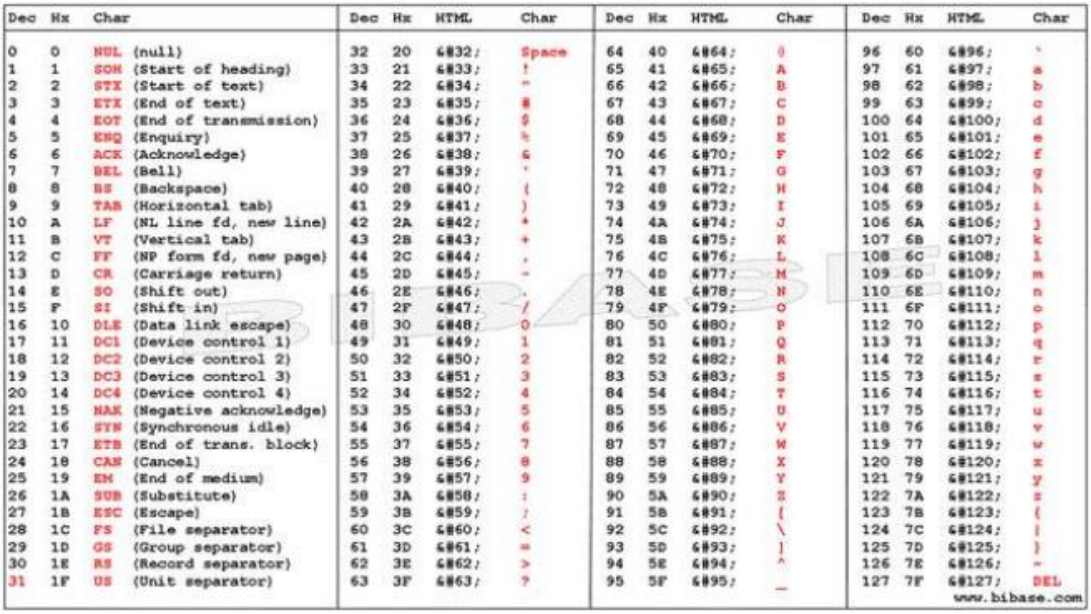
Đặt RS = 0 nếu byte dữ liệu là lệnh và tạo RS = 1 nếu byte dữ liệu là dữ liệu sẽ được hiển thị.

Đặt byte dữ liệu trên thanh ghi dữ liệu.

Xung E từ cao xuống thấp.

Lặp lại các bước trên để gửi dữ liệu khác.

Dữ liệu chữ cái và chữ số, ký hiệu được tham khảo từ bảng mã ASCII.



Dec	Hx	Char	Dec	Hx	HTML	Char	Dec	Hx	HTML	Char	Dec	Hx	HTML	Char
0	0	NUL (null)	32	20		Space	64	40	@	@	96	60	`	`
1	1	SOH (Start of heading)	33	21	!	!	65	41	A	A	97	61	a	a
2	2	STX (Start of text)	34	22	"	"	66	42	B	B	98	62	b	b
3	3	ETX (End of text)	35	23	#	#	67	43	C	C	99	63	c	c
4	4	EOT (End of transmission)	36	24	$	$	68	44	D	D	100	64	d	d
5	5	ENQ (Enquiry)	37	25	%	%	69	45	E	E	101	65	e	e
6	6	ACK (Acknowledge)	38	26	&	&	70	46	F	F	102	66	f	f
7	7	BEL (Bell)	39	27	'	'	71	47	G	G	103	67	g	g
8	8	BS (Backspace)	40	28	(	(	72	48	H	H	104	68	h	h
9	9	TAB (Horizontal tab)	41	29	)	)	73	49	I	I	105	69	i	i
10	A	LF (NL line fd, new line)	42	2A	*	*	74	4A	J	J	106	6A	j	j
11	B	VT (Vertical tab)	43	2B	+	+	75	4B	K	K	107	6B	k	k
12	C	FF (NP form fd, new page)	44	2C	,	,	76	4C	L	L	108	6C	l	l
13	D	CR (Carriage return)	45	2D	-	-	77	4D	M	M	109	6D	m	m
14	E	SO (Shift out)	46	2E	.	.	78	4E	N	N	110	6E	n	n
15	F	SI (Shift in)	47	2F	/	/	79	4F	O	O	111	6F	o	o
16	10	DLE (Data link escape)	48	30	0	0	80	50	P	P	112	70	p	p
17	11	DC1 (Device control 1)	49	31	1	1	81	51	Q	Q	113	71	q	q
18	12	DC2 (Device control 2)	50	32	2	2	82	52	R	R	114	72	r	r
19	13	DC3 (Device control 3)	51	33	3	3	83	53	S	S	115	73	s	s
20	14	DC4 (Device control 4)	52	34	4	4	84	54	T	T	116	74	t	t
21	15	NAK (Negative acknowledge)	53	35	5	5	85	55	U	U	117	75	u	u
22	16	SYN (Synchronous idle)	54	36	6	6	86	56	V	V	118	76	v	v
23	17	ETB (End of trans. block)	55	37	7	7	87	57	W	W	119	77	w	w
24	18	CAN (Cancel)	56	38	8	8	88	58	X	X	120	78	x	x
25	19	EM (End of medium)	57	39	9	9	89	59	Y	Y	121	79	y	y
26	1A	SUB (Substitute)	58	3A	:	:	90	5A	Z	Z	122	7A	z	z
27	1B	ESC (Escape)	59	3B	;	;	91	5B	[	[	123	7B	{	{
28	1C	FS (File separator)	60	3C	<	<	92	5C	\	\	124	7C			|
29	1D	GS (Group separator)	61	3D	=	=	93	5D	]	]	125	7D	}	}
30	1E	RS (Record separator)	62	3E	>	>	94	5E	^	^	126	7E	~	~
31	1F	US (Unit separator)	63	3F	?	?	95	5F	_	_	127	7F	&#127;	DEL

www.bibase.com

Hình 2.19 Bảng mã ASCII

3. TỔNG KẾT CÔNG VIỆC THỰC TẬP

3.1 Kết quả công việc thực tập

Kết quả thu được sau khi hoàn thành các công việc đã được giao tương đối tốt, đồng thời, đảm bảo tiến độ cho từng công việc. Tuy nhiên, vì đây là một lĩnh vực mới, một môi trường hoàn toàn mới nên không thể tránh khỏi những khó khăn và thiếu sót trong quá trình thực tập. Một số chương trình vẫn còn lỗi và chưa đáp ứng hoàn toàn những yêu cầu được đề ra ban đầu.

3.2 Kinh nghiệm học được sau khi thực tập

Trải qua quá trình Thực tập tốt nghiệp giúp em tiếp cận với một lĩnh vực hoàn toàn mới, cũng đầy khó khăn đó là lập trình với FPGA. Những qua đó, em được tìm hiểu, làm quen và thực hành với ngôn ngữ mô tả phần cứng Verilog HDL trên phần cứng là Kit Altera DE2. Từ đó, em có thể tự xây dựng, thiết kế một số mạch FPGA cơ bản, sau đó viết chương trình và mô phỏng trên phần cứng thực tế. Đó chính là những trải nghiệm vô cùng hữu ích và quý báu cho bản thân em.

4. TÀI LIỆU THAM KHẢO

[1] Tống Văn On, “Thiết kế mạch số với VHDL & Verilog”, Nhà xuất bản Lao động Xã Hội, 2007.

[2] Altera Corp., “SDRAM Controller for Altera’s DE2/ DE1 boards”, www.altera.com

[3] Nguyễn Như Anh, "Kỹ thuật số 1", Nhà xuất bản Đại học Quốc gia TP. Hồ Chí Minh, 2015.

[4] Altera Corp., "DE2 Development and Education Board – User Manual", www.altera.com

[5] M. Morris Mano, Charles R. Kime, Tom Martin, "Logic and Computer Design Fundamentals", 2015.

5. PHỤ LỤC