**ĐẠI HỌC QUỐC GIA THÀNH PHỒ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA ĐIỆN TỬ - VIỄN THÔNG**

**BỘ MÔN MÁY TÍNH – HỆ THỐNG NHÚNG**

---🙞🕮🙜---



**BÁO CÁO ĐỒ ÁN MÔN HỌC**

***Môn học:*** *Thiết kế SoC*

Giảng viên hướng dẫn: TS. Huỳnh Hữu Thuận

Lớp: 20Nhung1

Nhóm sinh viên thực hiện:

Lê Hoàng Việt Quốc 20200323

Phạm Vĩnh Phú 20200308

Thành phố Hồ Chí Minh, ngày 20 tháng 01 năm 2024

MỤC LỤC

[**ĐỀ BÀI** 2](#_Toc156728987)

[*Thiết kế phần cứng* 4](#_Toc156728988)

[Module Slave 4](#_Toc156728989)

[Mô tả tín hiệu của module 4](#_Toc156728990)

[Các thanh ghi trong module 5](#_Toc156728991)

[Dạng sóng đọc ghi 6](#_Toc156728992)

[Tổng quan hệ thống 6](#_Toc156728993)

[Xây dựng phần cứng trên Platform design 7](#_Toc156728994)

[System schematic 8](#_Toc156728995)

[*Tích hợp hệ thống SoC* 9](#_Toc156728996)

[Code verilog mô tả module Slave 9](#_Toc156728997)

[Code verilog mô tả module chính 11](#_Toc156728998)

[*Flow code C* 13](#_Toc156728999)

[*Kết quả* 19](#_Toc156729000)

# **ĐỀ BÀI:**

Thiết kế một hệ thống SoPC với Slave dùng DE-10 Standard thực hiện công việc sau:

1. Điều khiển 6 đèn 7 đoạn theo yêu cầu:

* 2 đèn hiển thị phút
* 2 đèn hiển thị giây
* 2 đèn hiển thị tỷ số trận đầu bóng đá

1. Hai button điều khiển

* Reset các LED
* Stop đồng hồ (giây, phút)

1. 8 switch hiển thị tỷ số, 4 switch cho mỗi 1 đèn 7 đoạn.

# *Thiết kế phần cứng:*

## Module Slave:

A diagram of a person

Description automatically generated

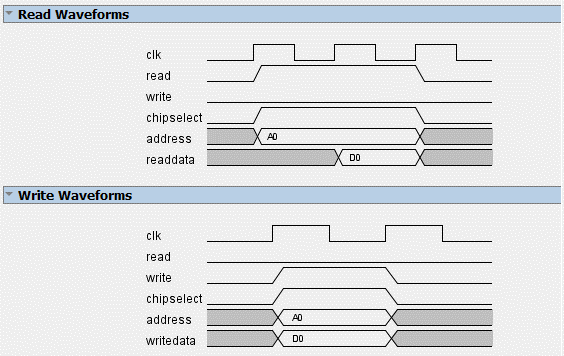
## Mô tả tín hiệu của module:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Tên tín hiệu | Độ rộng (bits) | Hướng | Mô tả |
| 1 | iClk | 1 | input | Cấp xung cho module hoạt động |
| 2 | iReset\_n | 1 | input | Cấp tín hiệu reset mức thấp cho module. |
| 3 | iWrite | 1 | input | Nếu write = 0, module được cho phép dữ liệu vào. Ngược lại, write = 1, module bỏ qua tín hiệu đưa vào. |
| 4 | iRead | 1 | input | Nếu read = 0, module cho phép đọc dữ liệu ra ngoài. Ngược lại, read = 1, module không cho phép đọc dữ liệu ra ngoài. |
| 5 | iChipSelect | 1 | input | Nếu cs = 0, module được phép hoạt động. Ngược lại, cs = 1, module không được hoạt động |
| 6 | iAddr | 2 | 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. |

## Các thanh ghi trong module:

|  |  |  |  |
| --- | --- | --- | --- |
| Offset | Tên thanh ghi | Đọc/Ghi | Mô tả |
| [31:0] |
| 0 | minute | Đọc/Ghi | Giá trị số phút của trận đấu |
| 1 | second | Đọc/Ghi | Giá trị số giây của trận đấu |
| 2 | point | Đọc/Ghi | Giá trị tỉ số của trận đấu |

## Dạng sóng đọc ghi:

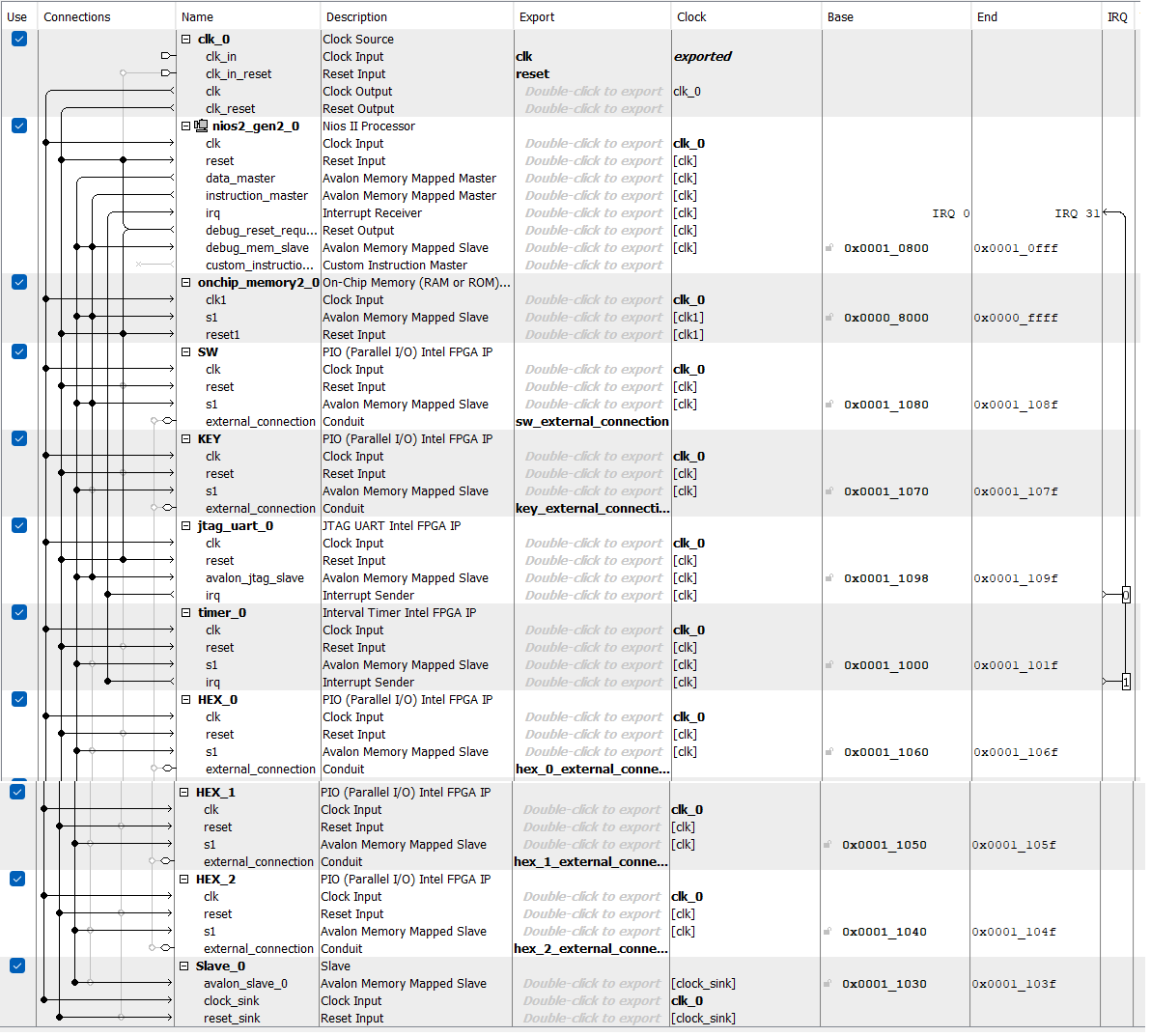


## Tổng quan hệ thống:

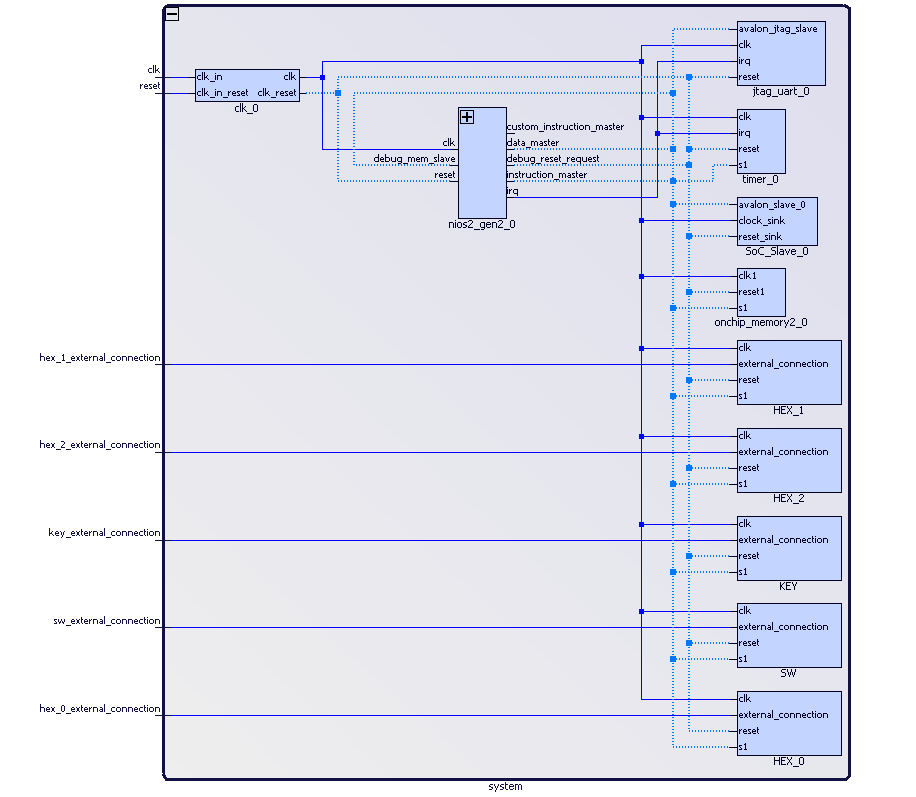
A diagram of a computer system

Description automatically generated

## Xây dựng phần cứng trên Platform design, thêm module Slave, các module PIO vào hệ thống, cấu hình tín hiệu clock, reset, kết nối tín hiệu. Hệ thống hoàn chỉnh như sau:



## System schematic



# *Tích hợp hệ thống SoC:*

## Code verilog mô tả module Slave:

module SoC\_Slave(

input iClk,

input iReset\_n,

input iWrite,

input iChipSelect,

input iRead,

input [1:0] iAddr,

input [31:0] iData,

output reg [31:0] oData

);

reg [31:0] minute, second, point;

always @(posedge iClk or negedge iReset\_n) begin

if(~iReset\_n) begin

oData <= 32'd0;

minute <= 32'd0;

second <= 32'd0;

point <= 32'd0;

end

else begin

if(iChipSelect & iWrite) begin

case (iAddr)

2'd0: minute[15:0] <= iData[15:0];

2'd1: second[15:0] <= iData[15:0];

2'd2: point[15:0] <= iData[15:0];

endcase

end

else if (iChipSelect & iRead) begin

case (iAddr)

2'd0: oData <= minute;

2'd1: oData <= second;

2'd2: oData <= point;

endcase

end

end

end

endmodule

Module Fir\_slave là một thành phần của hệ thống.

Trong đó:

**Cổng vào (Input Ports):**

* + iClk: clock.
  + iReset\_n: tín hiệu reset.
  + iWrite: tín hiệu ghi.
  + iRead: tín hiệu đọc.
  + iChipSelect: tín hiệu chọn chip.
  + iAddr: địa chỉ cần truy cập.
  + iData: đầu vào dữ liệu, là một bus 32-bit, chứa dữ liệu cần được ghi vào các thanh ghi khi **iWrite** được kích hoạt.

**Cổng ra (Output Ports):**

* oData: đầu ra dữ liệu, là một bus 32-bit, chứa dữ liệu được đọc từ thanh ghi tương ứng khi **iRead** được kích hoạt. Giá trị của **oData** có thể được truy cập từ các thành phần khác trong hệ thống.

**Biến và dây kết nối:**

* + minute: biến lưu giá trị phút.
  + second: biến lưu giá trị giây.
  + point: biến lưu giá trị tỉ số trận đấu.

Khối **always** xảy ra khi ở sườn lên xung clock hoặc ở sườn xuống iReset\_n.

* Nếu nút iReset\_n được nhấn inputs, outputs được reset về 0.
* Nếu nút reset\_n không được nhấn:
* Nếu trạng thái iChipSelect và iWrite được chọn, xét giá trị của address:

Nếu iAddr = 2’d0 thì gán giá trị của iData (input) vào biến minute.

Nếu iAddr = 2’d1 thì gán giá trị của iData (input) vào biến second.

Nếu iAddr = 2’d2 thì gán giá trị của iData (input) vào biến point.

* Nếu trạng thái iChipSelect và iRead được chọn, xét giá trị của address:

Nếu iAddr = 2’d0 thì gán giá trị oData = minute.

Nếu iAddr = 2’d1 thì gán giá trị oData = second.

Nếu iAddr = 2’d2 thì gán giá trị oData = point.

## Code verilog mô tả module chính:

module SoC\_lt(

input CLOCK\_50,

input [1:0] KEY,

input [9:0] SW,

output [0:6] HEX0, HEX1, HEX2, HEX3, HEX4, HEX5

);

system u\_sys(

.clk\_clk(CLOCK\_50),

.key\_external\_connection\_export(KEY[1]),

.reset\_reset\_n(KEY[0]),

.sw\_external\_connection\_export(SW),

.hex\_0\_external\_connection\_export({HEX1, HEX0}),

.hex\_1\_external\_connection\_export({HEX3, HEX2}),

.hex\_2\_external\_connection\_export({HEX5, HEX4})

);

endmodule

Trong đó:

**Cổng vào (Input Ports):**

* + CLOCK\_50: đầu vào xung đồng hồ 50 MHz.
  + KEY[1:0]: đầu vào 2-bit, là nút nhấn hoặc công tắc, KEY[0] dùng để reset, KEY[1] dùng để dừng/tiếp tục đếm giờ.
  + SW[9:0]: Đầu vào 10-bit, là SWITCH điều khiển tỉ số.

**Cổng ra (Output Ports):**

* + HEX0, HEX1, HEX2, HEX3, HEX4, HEX5: dữ liệu đầu ra mỗi đầu ra là một bus 7-bit, kết nối với hiển thị 7 đoạn.

**Module u\_sys:**

* **.clk\_clk(CLOCK\_50)**: kết nối đến đầu vào xung đồng hồ **CLOCK\_50**.
* **.key\_external\_connection\_export(KEY[1])**: kết nối đến bit thứ 2 của **KEY**.
* **.reset\_reset\_n(KEY[0])**: kết nối đến bit thứ 1 của **KEY**.
* **.sw\_external\_connection\_export(SW)**: kết nối đến cổng đầu vào **SW**.
* **.hex\_0\_external\_connection\_export({HEX1, HEX0})**: kết nối đến đầu ra **HEX0** và **HEX1**.
* **.hex\_1\_external\_connection\_export({HEX3, HEX2})**: kết nối đến đầu ra **HEX2** và **HEX3**.
* **.hex\_2\_external\_connection\_export({HEX5, HEX4})**: kết nối đến đầu ra **HEX4** và **HEX5**.

# *Flow code C:*

**#include** <stdio.h>

**#include** "io.h"

**#include** "system.h"

**#include** "altera\_avalon\_timer\_regs.h"

**#include** "sys/alt\_irq.h"

**volatile** **int**\* sw = (**int**\*) SW\_BASE;

**unsigned** **int** diemso = 0;

**unsigned** **int** doi1 = 0;

**unsigned** **int** doi2 = 0;

**unsigned** **int** phut = 45;

**unsigned** **int** giay = 0;

**volatile** **int**\* button = (**int**\*) KEY\_BASE;

**volatile** **int**\* hex0 = (**int**\*) HEX\_0\_BASE;

**volatile** **int**\* hex1 = (**int**\*) HEX\_1\_BASE;

**volatile** **int**\* hex2 = (**int**\*) HEX\_2\_BASE;

**void** **dem**(){

**volatile** **int**\* slave = (**int**\*) SLAVE\_0\_BASE;

**unsigned** **int** hex\_decode[10] = {1, 79, 18, 6, 76, 36, 32, 15, 0, 4};

**int** giay\_h, phut\_h;

**if**(giay == 60){

giay = 0;

phut++;

}

**if**(phut == 90){

IOWR\_ALTERA\_AVALON\_TIMER\_CONTROL(TIMER\_0\_BASE, ALTERA\_AVALON\_TIMER\_CONTROL\_CONT\_MSK | ALTERA\_AVALON\_TIMER\_CONTROL\_ITO\_MSK | ALTERA\_AVALON\_TIMER\_CONTROL\_STOP\_MSK);

}

\*(slave + 0) = hex\_decode[giay/10] << 7 | hex\_decode[giay%10];

giay\_h = \*(slave + 0);

\*(hex0) = giay\_h;

\*(slave + 1) = hex\_decode[phut/10] << 7 | hex\_decode[phut%10];

phut\_h = \*(slave + 1);

\*(hex1) = phut\_h;

}

**void** **tiso**(){

**volatile** **int**\* slave = (**int**\*) SLAVE\_0\_BASE;

**unsigned** **int** hex\_decode[10] = {1, 79, 18, 6, 76, 36, 32, 15, 0, 4};

**int** y;

diemso = \*(sw);

doi2 = (diemso & 240) >> 4;

doi1 = diemso & 15;

\*(slave + 2) = hex\_decode[doi2] << 7 | hex\_decode[doi1];

y = \*(slave + 2);

\*(hex2) = y;

}

**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\_PERIODH(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);

}

**void** **Timer\_IRQ\_Handler**(**void**\* isr\_context){

giay++;

IOWR\_ALTERA\_AVALON\_TIMER\_STATUS(TIMER\_0\_BASE, ALTERA\_AVALON\_TIMER\_STATUS\_TO\_MSK);

tiso();

}

**void** **xu\_ly\_ngat**(){

**unsigned** **int** temp = 1;

**if**(\*(button) == 0){

**while**(\*(button) == 0);

temp = ~temp & 1;

**if**(temp == 0){

IOWR\_ALTERA\_AVALON\_TIMER\_CONTROL(TIMER\_0\_BASE, ALTERA\_AVALON\_TIMER\_CONTROL\_CONT\_MSK | ALTERA\_AVALON\_TIMER\_CONTROL\_ITO\_MSK |ALTERA\_AVALON\_TIMER\_CONTROL\_STOP\_MSK);

}

**else**

{

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**(){

alt\_ic\_isr\_register(0, TIMER\_0\_IRQ, Timer\_IRQ\_Handler, (**void**\*)0,(**void**\*)0);

timer\_Init();

**while**(1){

xu\_ly\_ngat();

dem();

};

**return** 0;

}

# *Kết quả:*

Hệ thống hoạt động đúng theo yêu cầu: 2 đèn HEX1, HEX0 hiển thị tỉ số trận đấu, 2 đèn HEX3, HEX2 hiển thị số phút, 2 đền HEX5, HEX4 hiển thị số giây. Nút KEY[0] dùng để reset, KEY[1] dùng để tạm dừng/tiếp tục đếm giờ. SWITCH[7:0] dùng để điều khiển tỉ số trận đấu.

−−−🙜🕮🙞−−−