TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN ĐIỆN TỬ - VIỄN THÔNG

ESRC LAB

Đề tài:

SNAKE GAME trên Kit DE1

Nhóm sinh viên:

Group 1 – K53:

Vũ Quang Trọng

Đỗ Sơn Tùng

Hà Nội, 8/2011

1. Giới thiệu

1.1.Đề tài

Sau khi hoàn thành các bài lab thực hành trên Kit DE1 của Altera, chúng em tiếp tục phát triển kỹ năng thiết kế và vận dụng vào thực tế, đó là triển khai một hệ thống hoàn chỉnh trên Kit DE1 với đế tài:

“ Sử dụng Kit DE1 của hãng Altera để tạo trò chơi Snake cho một người chơi với giao diện đồ họa, giao tiếp với người chơi qua bàn phím PS2 và màn hình VGA”.

1.2. Các thành viên và phân công công việc

Picture go here	Picture go here
Vũ Quang Trọng (trưởng nhóm) 0973.750.337 vuquangtrong@gmail.com	Đỗ Sơn Tùng 0168.9.929.537 tungmontaint@gmail.com
Lập sơ đồ tổng thể đề tài. Khối logic trạng thái hệ thống. Khối điều khiển Rắn. Khối hiển thị Đồ họa. Khối hiển thị Văn bản. Và các khối phụ cần thiết khác.	Tìm hiểu kết nối PS2. Tìm hiểu và điều khiển VGA. Tìm hiểu IC AudioCodec và điều khiển khối Âm thanh.

1.3.Yêu cầu của đề tài

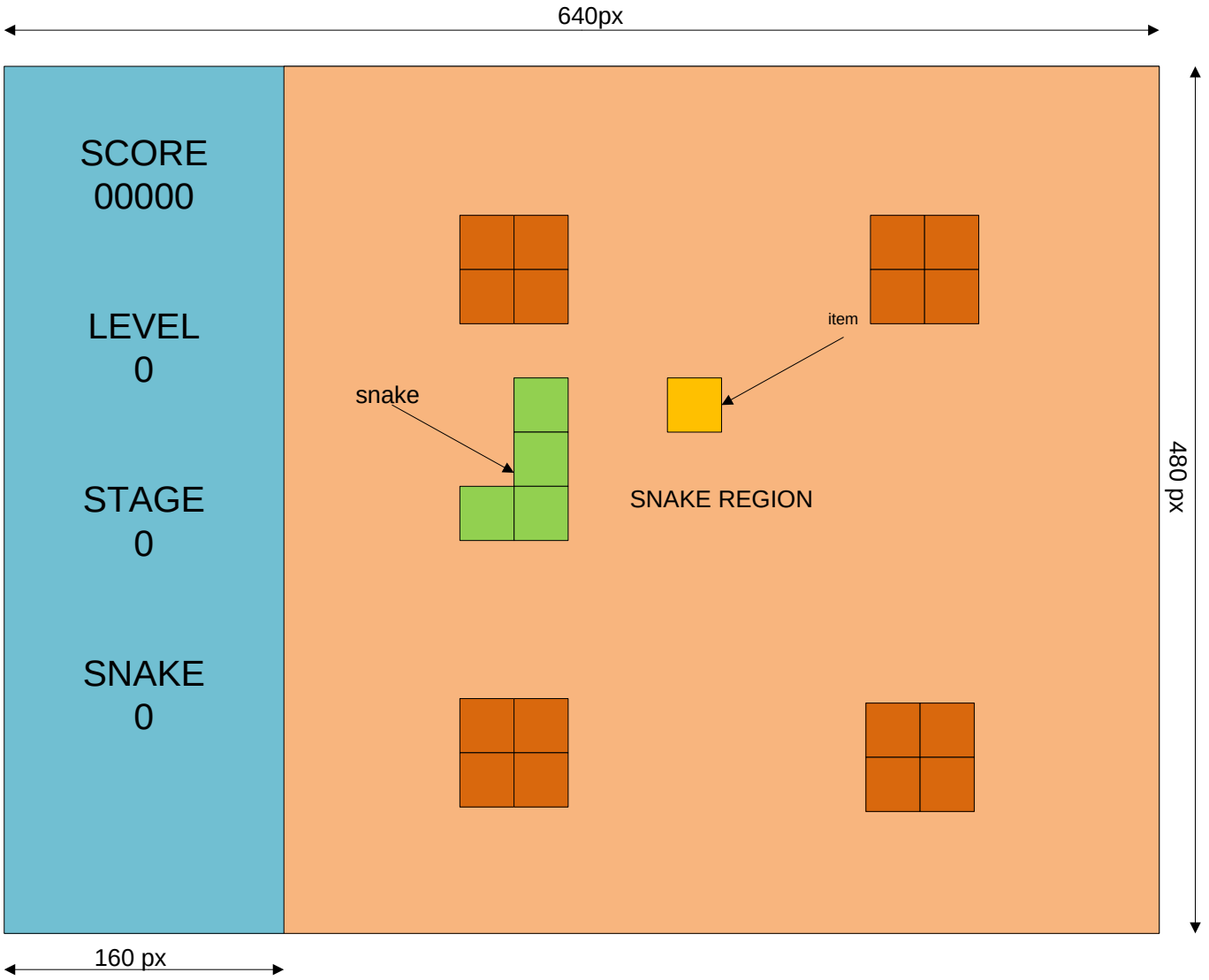
1.3.1. Yêu cầu chức năng

- Phần cứng
  - Game chạy hoàn toàn trên Kit DE1
  - Nhận điều khiển từ bàn phím PS2

  - Hiển thị trên màn hình VGA độ phân giải 640 x 480
  - Âm thanh phát qua loa 2.0 bằng IC audio codec WM8731 có sắn trên Kit DE1
- Phần mềm
  - Sử dụng Quartus II 9.1 Sp2
  - Ngôn ngữ VHDL thuần
  - Game có đủ các yếu tố để trở thành một game hoàn chỉnh:
    - Người chơi điều khiển rắn bằng 4 nút chỉ hướng trên bàn phím.
    - Các phím chức năng như PAUSE, SELECT đê người chơi thao tác với MENU trong game.
    - Rắn dài ra khi ăn mồi, hoặc có các tương tác khác tùy loại mồi vừa ăn.
    - Răn sẽ chết nếu như rắn đâm vào tường hoặc chính đuôi nó, lúc này bạn sẽ có một rắn mới đề chơi lại.
    - Điều kiện để chơi lại khi và chỉ khi số lượt chơi lại của bạn lớn hơn 0, ban đầu, số lượt chơi lại sẽ là 3, có thể tăng nếu ăn loại mồi LIVE_UP, và giảm khi rắn chết. Nếu số lượt chơi lại của bạn là 0, trò chơi sẽ kết thúc.
    - Các mức độ khó khác nhau, đó là tốc độ của rắn sẽ tăng qua một số lần ăn mồi nhất định, tiếp đó là màn chơi có các chướng ngại vật khác nhau.
    - Tính điểm cho người chơi, điểm được tính dựa vào level hiện tại và loại mồi mà rắn ăn được.
    - Lưu tên người chơi khi điểm người chơi đạt được một số điểm đứng trong top 5

1.3.2. Yêu cầu phi chức năng

- Thời gian đáp ứng phím gõ nhỏ nhằm giúp người chơi chuyển hướng khi điều khiển dễ dàng, tốc độ bấm khoảng 4 lần / giây => thời gian đáp ứng 250ms
- Hiển thị trên màn hình VGA 640x480 với 8 màu cơ bản, sử dụng 3 bit cho một màu.
- Tần số quét màn hình đủ lớn để đảm bảo việc hiển thị hình ảnh mượt mà, ta lấy 60Hz là tần số quét.
- Luật điều khiển được áp dụng như sau:
    - Rắn không thể quay đầu ngược lại hướng đang di chuyển, tức là nếu đang tiến lên thì khi nhấn nút lùi sẽ không có tác dụng, tương tự khi răng đang sang trái, phải, hoặc đi xuống.
- Khung màn hình của người chơi:



Khung màn chơi được chia thành 2 khung chính:

- Khung hiển thị thông tin của người chơi:
  - Bao gồm Điểm, Level, Stage và số rắn còn lại của người chơi.
  - Kích thước 160x480.
  - Text được hiển thị với cỡ 32x16.
- Khung hiển thị phần chơi:
  - Kích thước 480x480

- Rắn, mồi , và tường được xây dựng từ các khối vuông 16x16 ghép lại.
- Phần này có thể để hiển thị thông báo khi cần.

1.4. Giới thiệu phần cứng

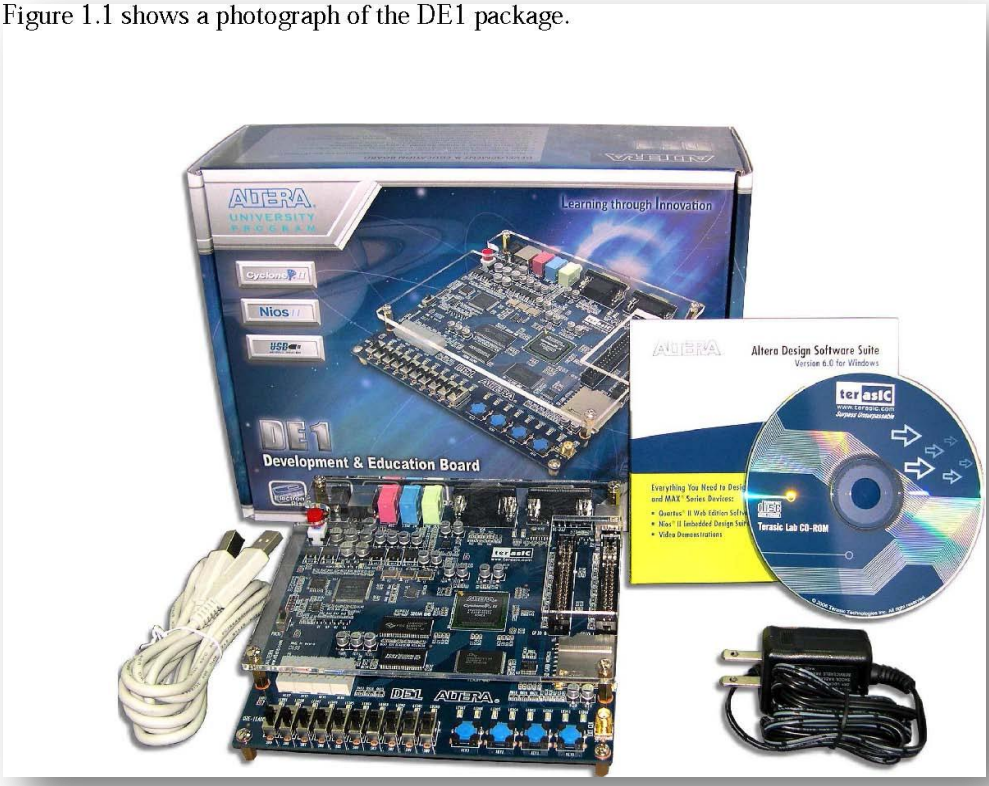
1.4.1. Kit DE1

1.4.1.1. Giới thiệu KIT DE1

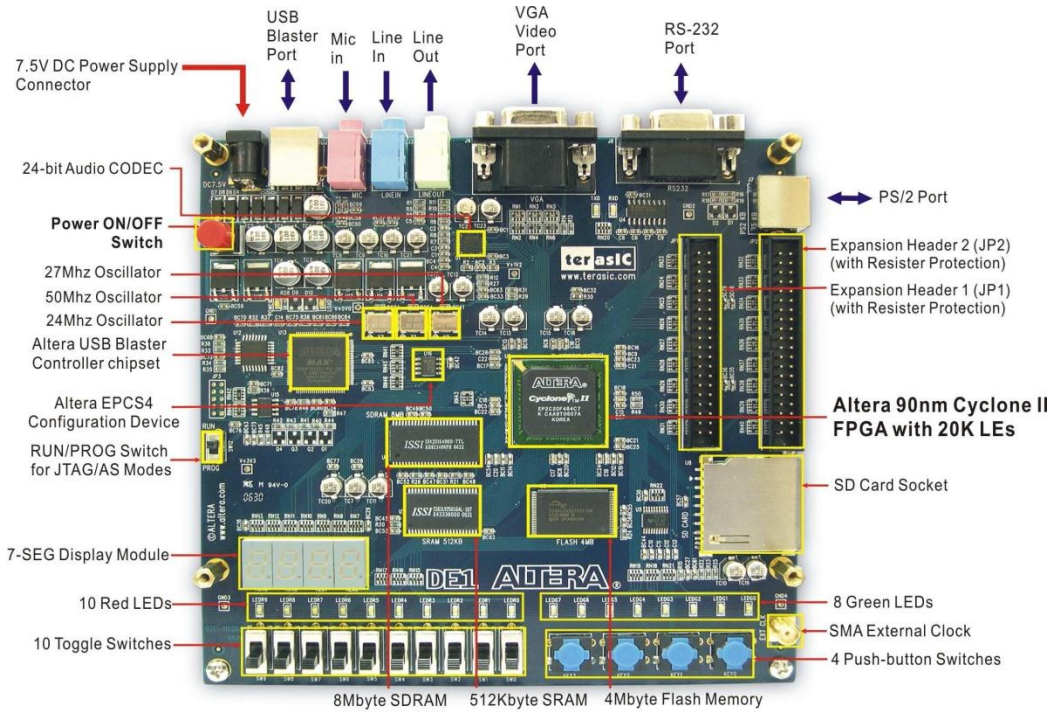
KIT DE1 là một sản phẩm của hãng Altera.Mục đích của nhà phát triển khi tạo ra KIT DE1 là cung cấp một công cụ lý tưởng để phục vụ cho các thiết kế tiên tiến trong một số lĩnh vực như đa phương tiện, lưu trữ, mạng…

Để sử dụng được KIT DE1, chúng ta cần kết nối với một máy tính chạy phần mềm Microsoft Windows.

Figure 1.1 shows a photograph of the DE1 package.



1.4.1.2. Các thành phần trên KIT DE1



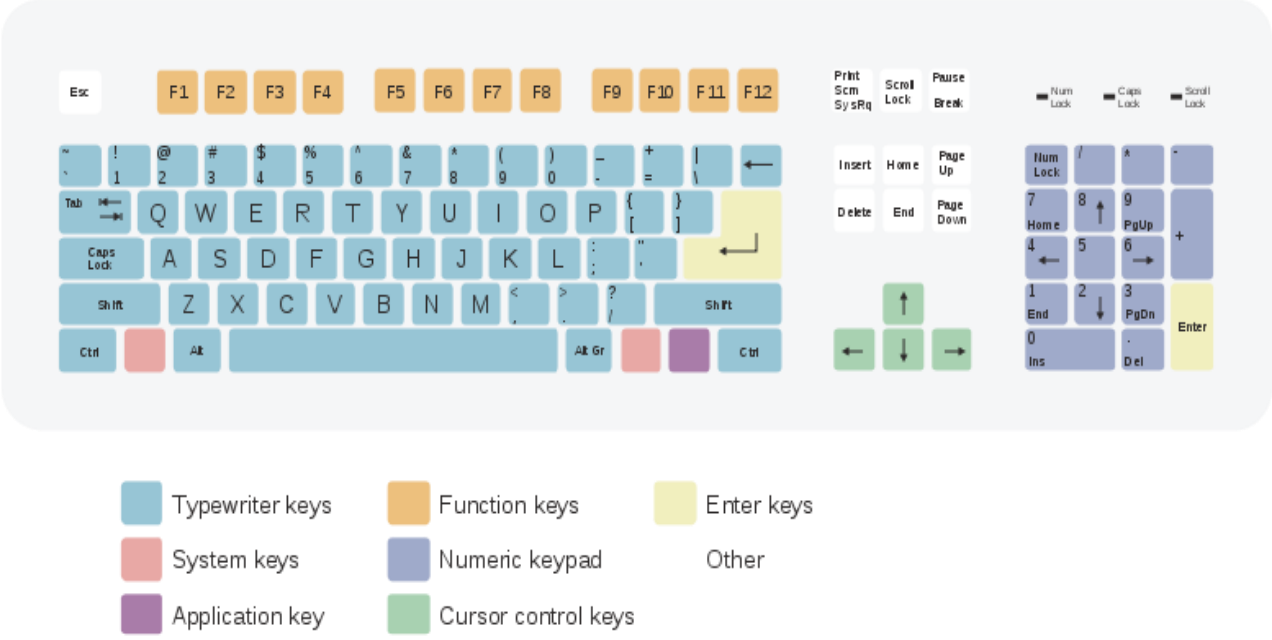
- Altera Cyclone® II 2C20 FPGA device
- Altera Serial Configuration device – EPCS4
- USB Blaster (on board) for programming and user API control; both JTAG and Active Serial

(AS) programming modes are supported

- 512-Kbyte SRAM
- 8-Mbyte SDRAM
- 4-Mbyte Flash memory
- SD Card socket
- 4 pushbutton switches
- 10 toggle switches
- 10 red user LEDs
- 8 reen user LEDs

- 50-MHz oscillator, 27-MHz oscillator and 24-MHz oscillator for clock sources
- 24-bit CD-quality audio CODEC with line-in, line-out, and microphone-in jacks
- VGA DAC (4-bit resistor network) with VGA-out connector
- RS-232 transceiver and 9-pin connector
- PS/2 mouse/keyboard connector
- Two 40-pin Expansion Headers with resistor protec
- Powered by either a 7.5V DC adapter or a USB cable

1.4.2. Keyboard PS2

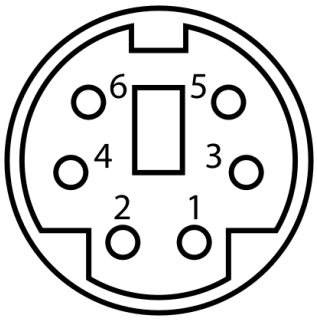


Với phạm vi của chúng tôi thiết kế project này, chỉ quan tâm đến các nút điều khuyển hường di chuyển : lên, xuống, trái, phải. Hai phím chức năng tạm dừng và lựa chọn : Esc, Enter.Có thể phát triển thêm các nút khác do yêu cầu từng đề tài.

Chi tiết về cách nhận phím và truyền dữ liệu từ bàn phím sẽ được liệt kê ở phần chi tiết hệ thống.

Sơ đồ chân PS2

Sử dụng chuẩn kết nối PS2 để kết nối keyboard với KIT DE1



Pin 1 +DATA Data

Pin 2 Not connected Not connected*

Pin 3 GND Gr

Pin 4 Vcc +5 V DC at 275 mA

Pin 5 +CLK

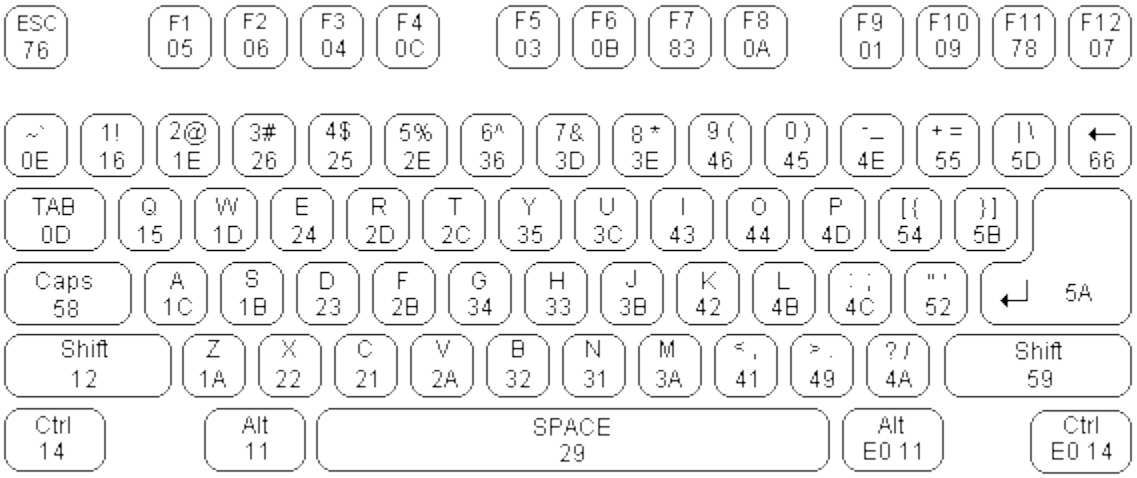
Pin 6 Not connected Not connected**

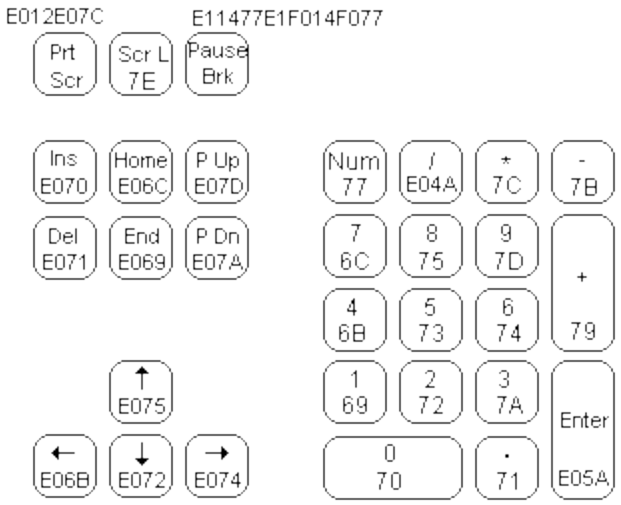
Ngoài chuẩn PS2 còn có một số chuẩn kết nối thông dụng khác là USB và không dây(wireless). Phạm vi đề tài project chỉ thực hiện với kết nối PS2 nên chúng ta chỉ giới thiệu và không tìm hiểu về 2 chuẩn kia.

Giao diện PS2 là một loại đầu nối 6-pin MINI DIN.

2.1.2.3. Scan code

Một bàn phím bao gồm một ma trận các phím và một vi xử lý nhúng để kiểm tra những hoạt động của phím và gửi *scan code* phù hợp.





Hoạt động của bàn phím :

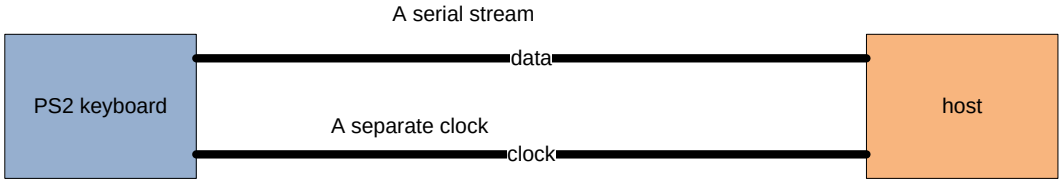
- Khi 1 phím được ấn thì *the make code* của phím được truyền đi.

- Khi 1 phím được giữ liên tục,trạng thái được biết như là *typematic thì the make code* được truyền đi liên tục với một tốc độ nhất định. Ở chế độ mặc định, một PS2 keyboard truyền *make –code* vào khoảng 100ms sau khi 1 phím đã được giữ trong khoảng 0.5s.

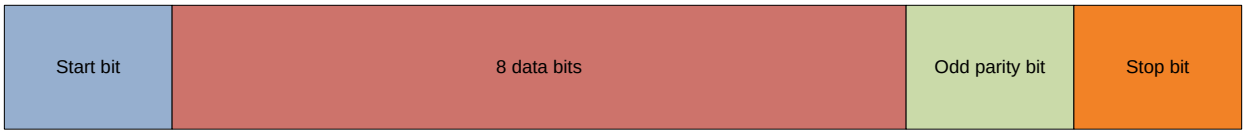
- Khi 1 phím được nhả ra thì *the break code(0xF0)* được truyền đi sau đó là *make code* của phím để nhận biết phím nào vừa được nhả.

Cách truyền nhận dữ liệu

Một thiết bị PS2 (bàn phím) kết nối với KIT DE1 và trao đổi dữ liệu thông qua 2 đường là data và clock



Đường data gồm 11 bit



Đường clock được mang trong một tín hiệu clock riêng biệt.

Dữ liệu sẽ được truyền đi khi xung clock có sự thay đổi và đang tích cực mức thấp(falling-edge).

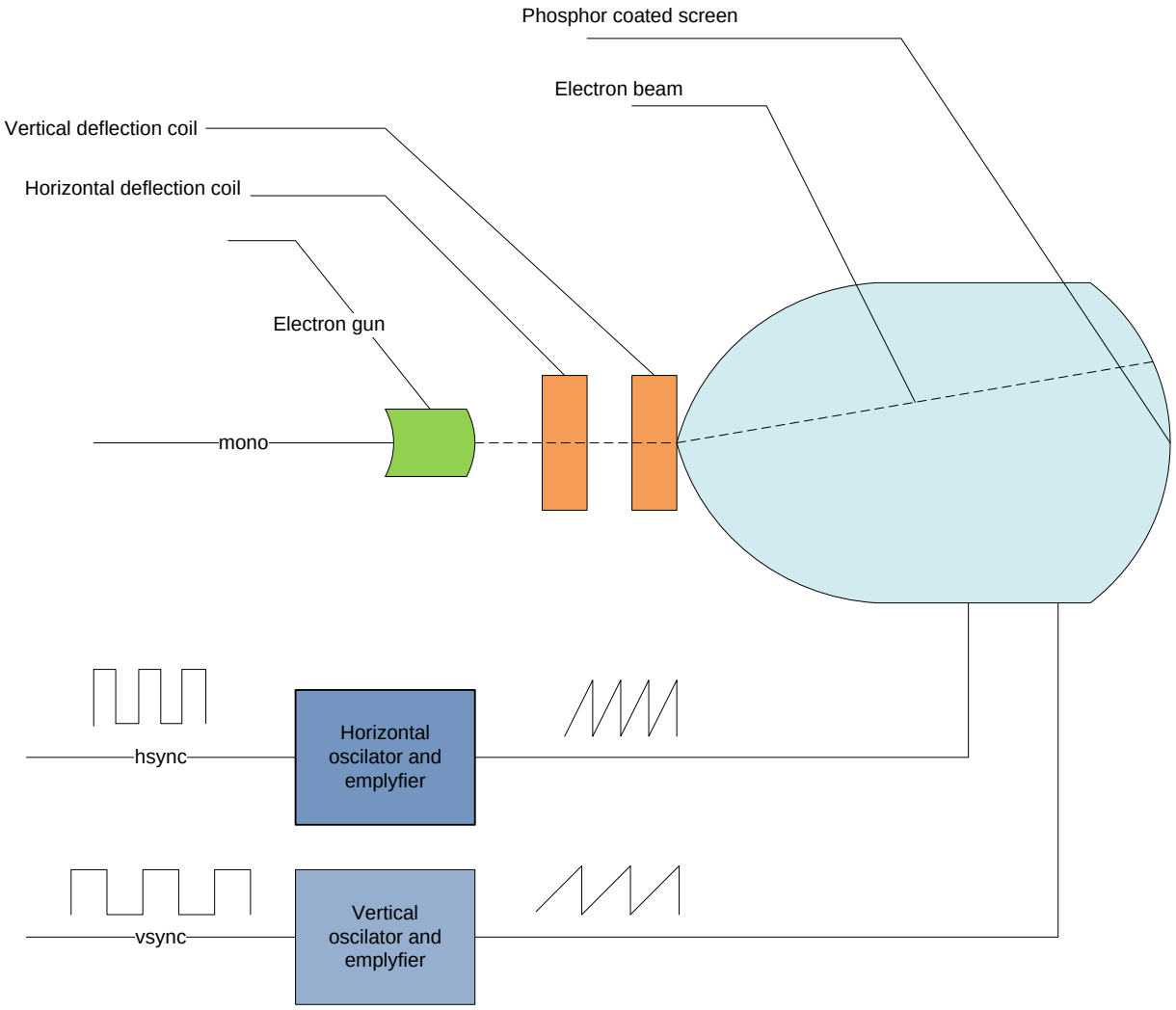
1.4.3. Monitor VGA

Giới thiệu

VGA(Video Graphics Arrays: mảng đồ họa video) được giới thiểu bởi IBM PCs được hỗ trợ bởi phần cứng đồ họa PC và màn hình.Chúng ta sẽ thiết kế một giao diện gồm 8 màu cơ bản với độ phân giải 640x480 cho màn hình CRT.

Cơ chế hoạt động cơ bản của một CRT

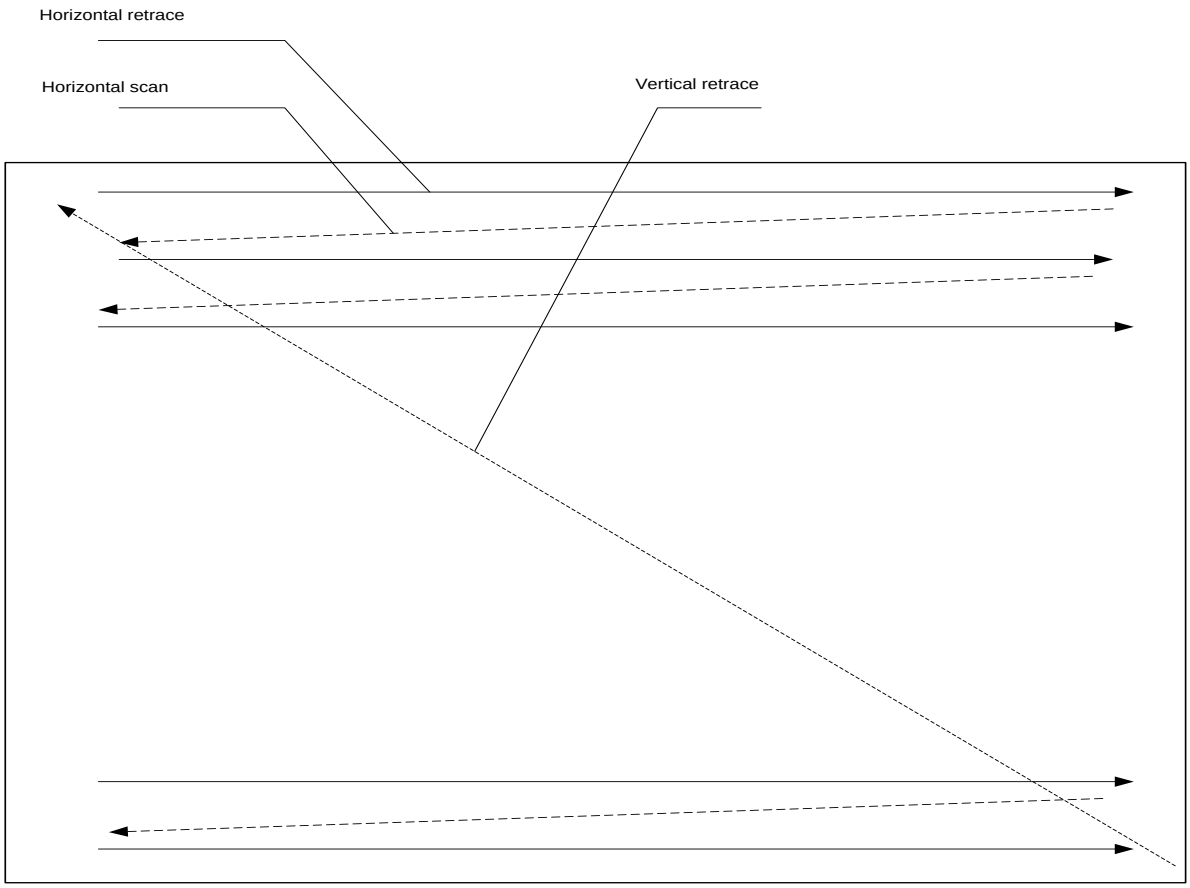
Sơ đồ khối:



Hình 1

- Cường độ tia electron và độ sáng của các điểm được quyết định bởi mức điện thế tín hiệu video đầu vào,mono.Tín hiệu mono là tín hiệu tương tự có mức điện thế thay đổi giữa 0 và 0,7.
- The vertical deflection coil và horizontal deflection coil điều khiển hành trình của dòng electron và quyết định nơi mà electron đậu trên màn hình.Với các màn hình ngày nay,trùm electron được điều khiển từ trái sang phải từ trên xuống dưới.

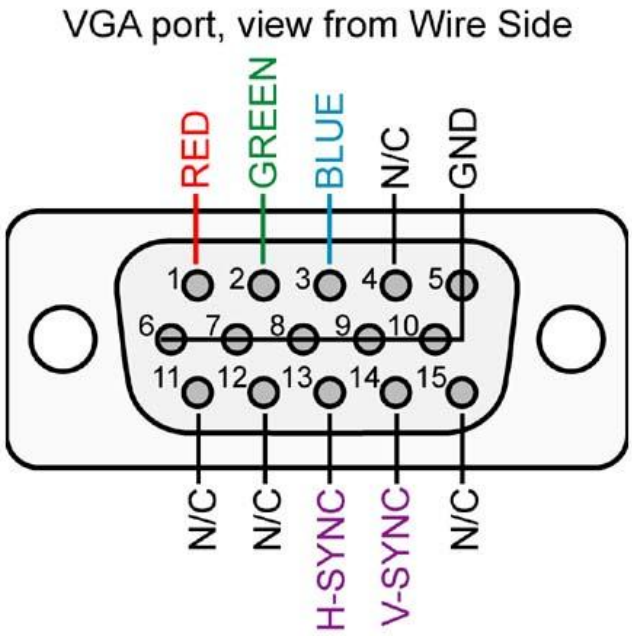
Cách quét VGA



Khi điện áp được đưa vào the honrizontal deflection coil và tăng một cách đều đặn thì trùm electron sẽ di chuyển từ góc trái sang góc phải. Sau khi chạm tới góc phải, trùm tia sẽ nhanh chóng quay trở lại góc trái khi điện áp về 0V (hsync). Cho đến khi trùm electron trạm tới đáy màn hình thì điện áp sẽ được đưa vào the vertical deflection coil, trùm tia sẽ được đưa trở lại đỉnh màn hình (vsync) và tiếp tục quá trình như trong hình

Tín hiệu hsync dùng để quét màn hình theo hàng và tín hiệu vsync dùng để quét toàn bộ màn hình với tần số 25MHz pixel rate(25 triệu điểm ảnh được thực hiện trong 1s) để có thể tạo ra màn hình VGA độ phân giải 640x480 .

Các chân VGA



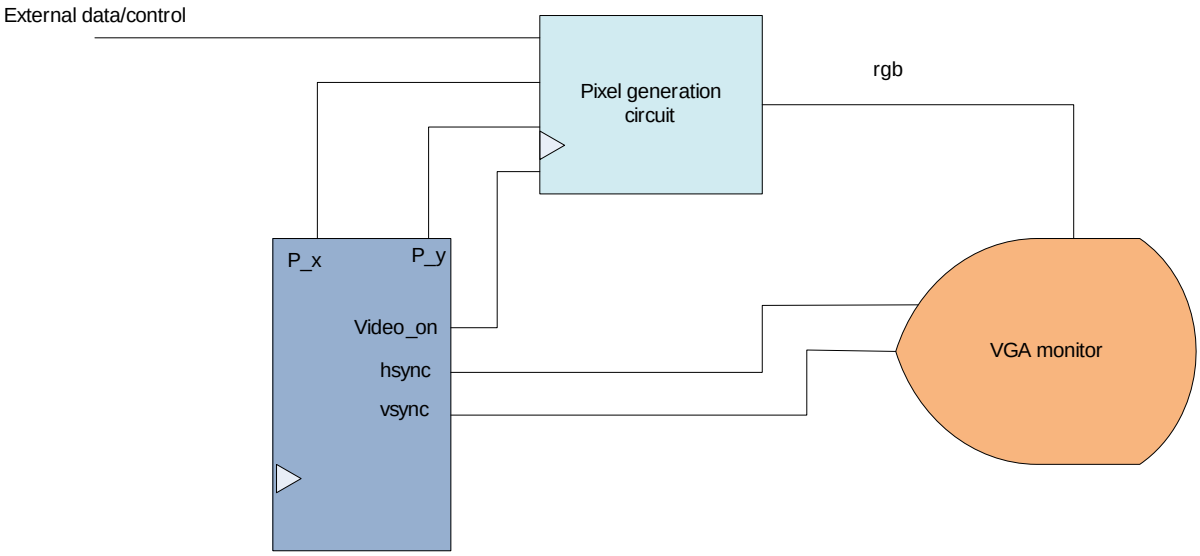
Cổng VGA bao gồm 5 tín hiệu hoạt động: hai tín hiệu hsync và vsync, ba tín hiệu video là đỏ, xanh biển, xanh lá cây được kết nối vào 15 chân

Red	Green	Blue	Resulting colour
0	0	0	Black
0	0	1	Blue
0	1	0	green
0	1	1	Cyan
1	0	0	Red
1	0	1	Magenta
1	1	0	Yellow
1	1	1	White

Cách truyền nhận dữ liệu

Để có thể truyền nhận dữ liệu và hiển thị lên màn hình, chúng tôi đã thiết kế một mạch vga_sync gồm bộ đếm và các tín hiệu đồng bộ.Mạch này có 2 tín

hiệu hsync và vsync được nối trực tiếp đến màn hình,chúng dùng để điều khuyển sự quét ngang và dọc màn hình.Hai tín hiệu này được giải mã bởi một bộ đếm có sẵn trong mạch và đầu ra của 2 tín hiệu này là pixel_x, pixel_y.Hai tín hiệu đầu ra này chỉ ra quan hệ giữa vị trí quét và vị trí hiện tại của điểm ảnh.Mạch có một tín hiệu video_on để điều khuyển tắt hay bật sự hiển thị.



Khối điều khiển VGA

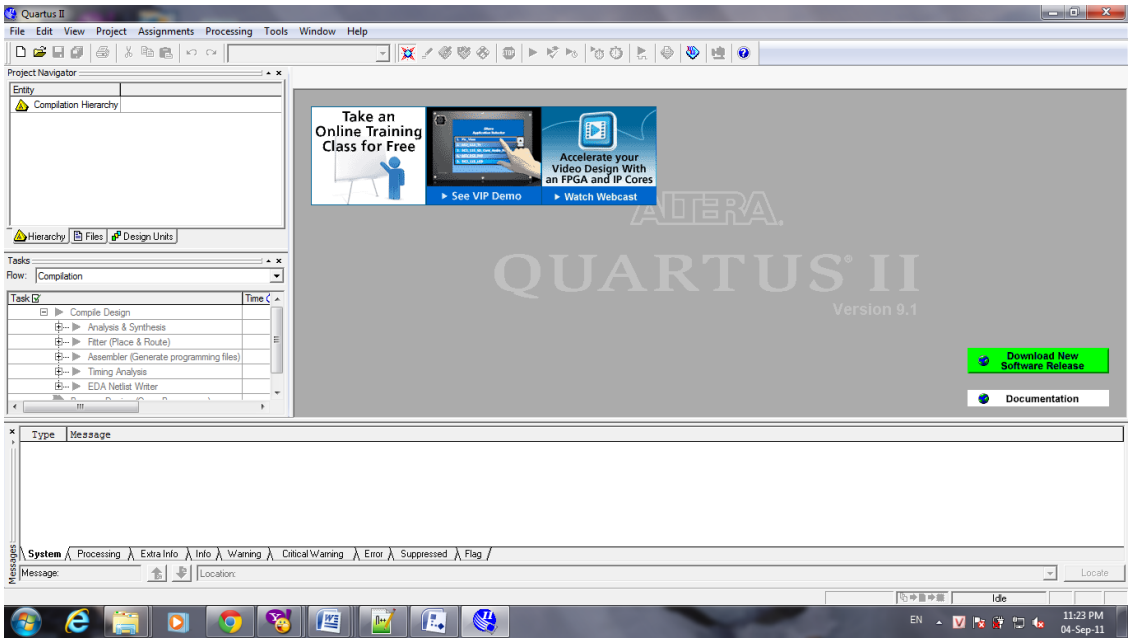
Một mạch để tạo ra 3 tín hiệu video được gọi chung là tín hiệu rgb(red green blue) có đầu vào là pixel_x và pixel_y, video_on. Giá trị của một màu được hiện thị trên màn hình phụ thuộc vào vị trí điểm ảnh hiện tại(pixel_x và pixel_y) và tín hiệu data và điều khuyển bên ngoài

1.5.Giới thiệu phần mềm

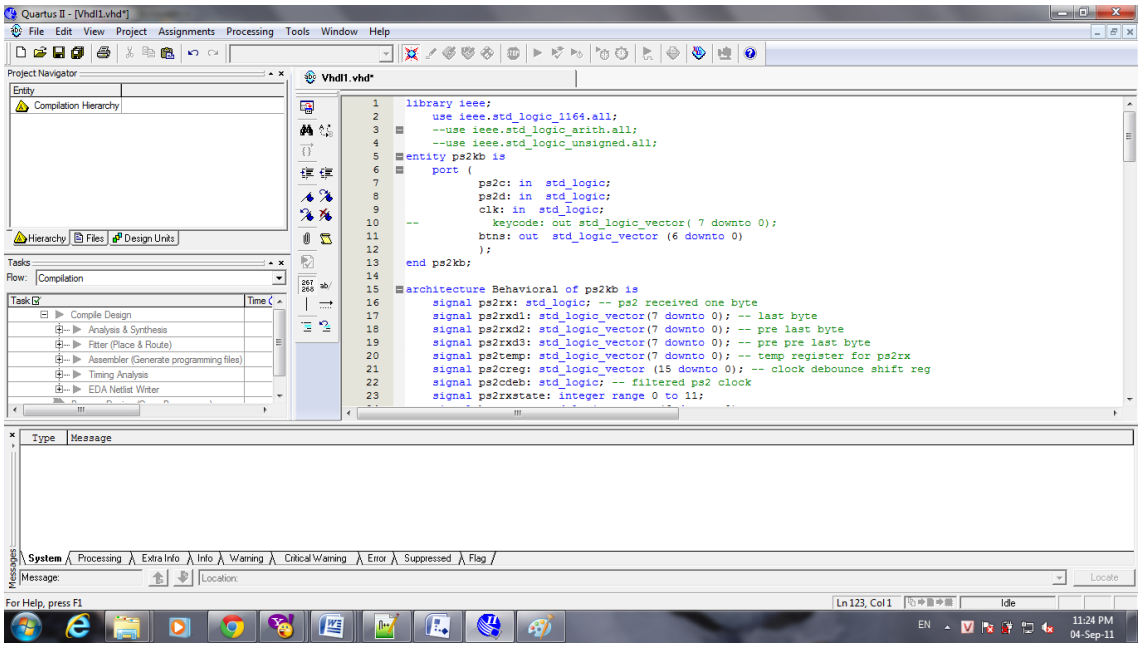
1.5.1. Quartus II

Giao diện chính

Quartus 9.1 Web Edition dùng để lập trình và nạp lên KIT DE1



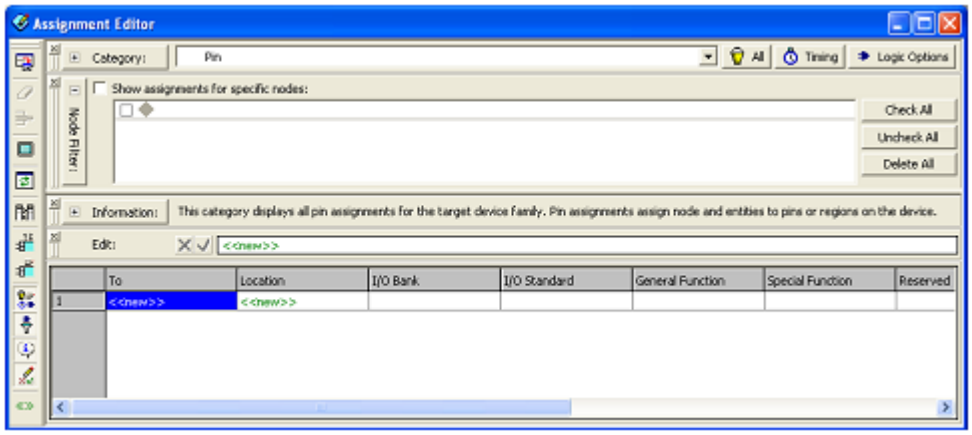
2.2.1.2 Trình soạn thảo



Gán chân

Bước 1: Nhấp chọn Assignments > Assignment Editor. Trong Category chọn Pin. Nhấp đôi vào <<new>>. Nhấp chọn tín hiệu từ danh sách trải xuống làm chân được gán. Tiếp đến nhấp đôi vào hộp bên phải hộp dành

cho tín hiệu cần gán (cột Location). Chọn chân PIN_XYZ từ danh mục hoặc có thể gõ trực tiếp vào hộp Location.



Hình 6. Cửa sổ Assignment Editor dùng để gán chân

Bước 2: Tương tự như trên, thực hiện gán chân cho các đầu vào khác

Bước 3: Sau khi hoàn thành việc gán chân, nhấp chọn File > Save. Đóng cửa sổ Assignment Editor, nhấp chọn Yes và biên dịch lại mạch.

Chú ý : Nên chọn tên các tín hiệu trùng với trong bảng DE1_pin_assigment.csv thì khi gán chân ta chỉ cần vào assigment > import assigment,trong đường dẫn ta trỏ đến file DE1_pin_assigment.csv rồi ấn OK và làm theo bước 3 là xong.Không mất thời gian gán chân bằng tay.

Biên dịch

Khi đã viết code xong cho một chương trình nào đó bạn cần biên dịch để tạo ra những file dùng để nạp lên KIT DE1

Bước 1: Nhấp chọn mục Processing > Start Complication. Biên dịch thành công (hay không thành công) sẽ được thông báo trên hộp thoại bung ra sau khi quá trình biên dịch kết thúc. Xác nhận bằng cách nhấp nút OK.

Bước 2: Khi biên dịch hoàn thành, một báo cáo biên dịch được đưa ra. Cửa sổ này cũng có thể mở ra bất kỳ lúc nào bằng cách nhấp chọn Processing > Complication Report. Trong báo cáo này bao gồm một số danh mục ở bên

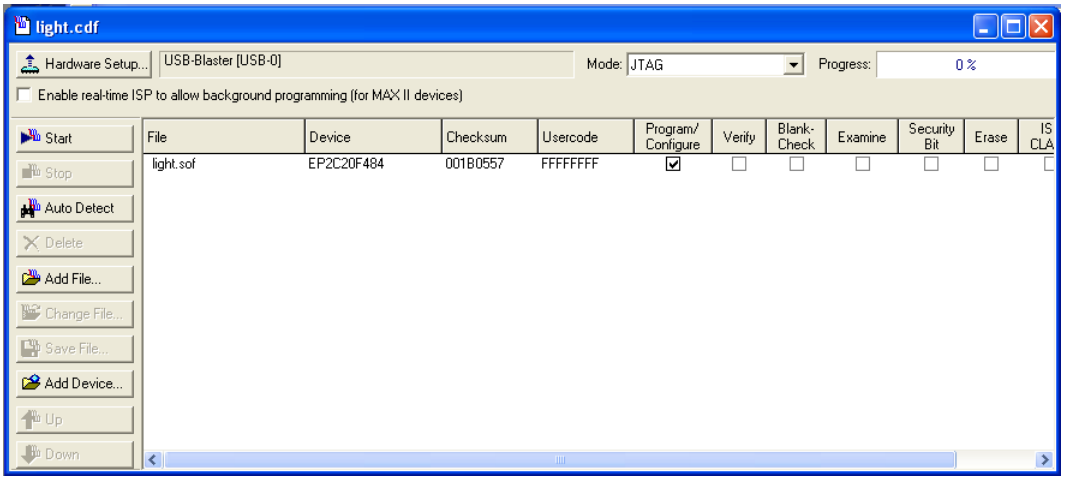
trái cửa sổ, nhấp vào các danh mục này để thấy thông tin chi tiết của danh mục này hiện lên ở bên phải cửa sổ.

Bước 3: Sửa các lỗi

Chọn mục Analysis & Synthesis > Messages để hiển thị thông báo lỗi. Nhấp đôi vào thông báo lỗi đầu tiên, dòng lệnh lỗi sẽ được đánh dấu trên trình soạn thảo văn bản, sửa lại cho đúng rồi biên dịch lại dự án.

Nạp lên KIT

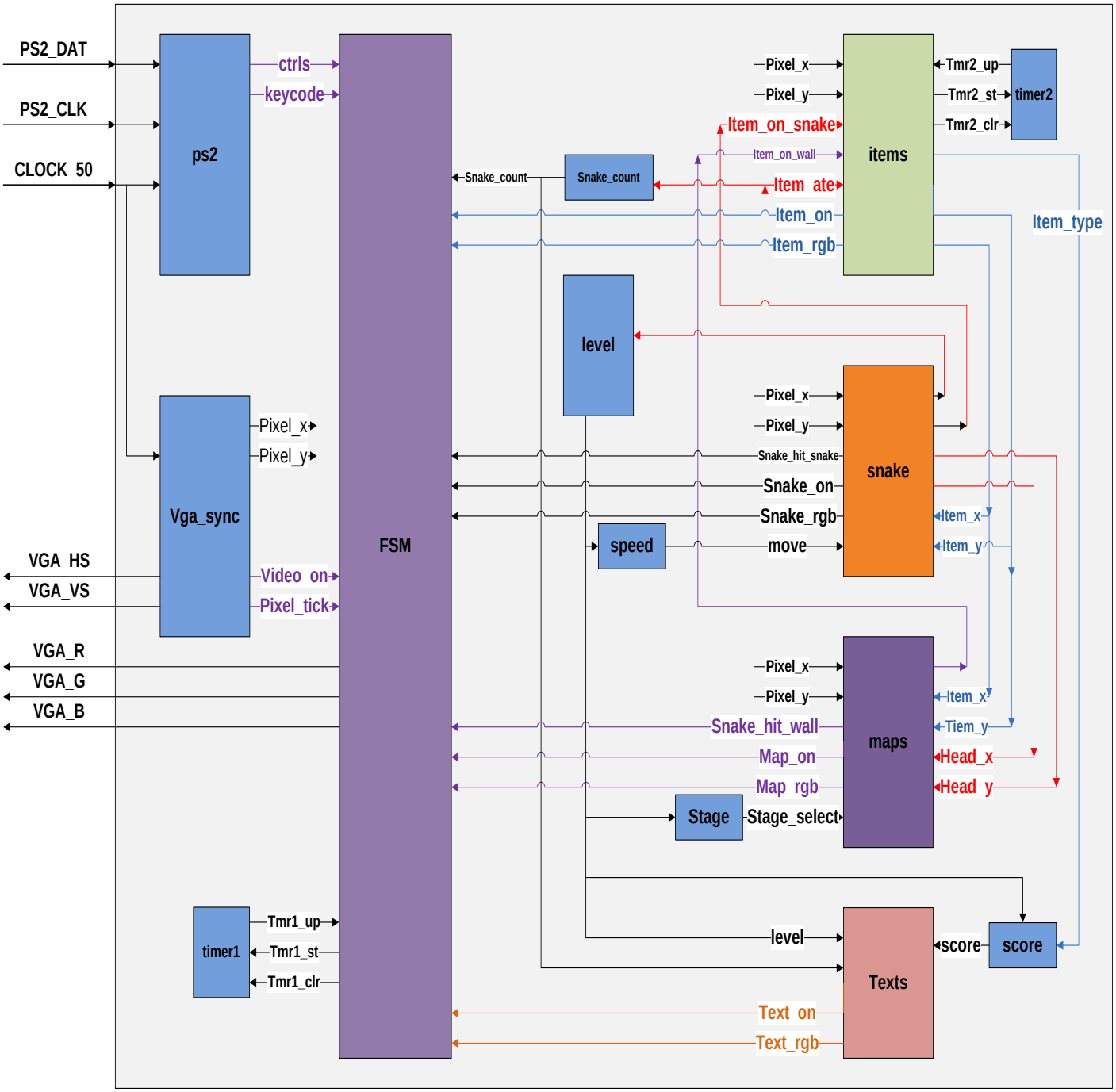
Bước 1: Gạt chuyển mạch RUN/PROG sang vị trí RUN. Nhấp chọn Tools > Programmer để có cửa sổ như trong hình 11. Đánh dấu vào tùy chọn Program/Configue để cho phép nạp tệp cấu hình xxxxxx.sof.



Bước 2: Nhấp nút Start bên trái cửa sổ để nạp tệp cấu hình này xuống FPGA. Sau khi nạp thành công xuống FPGA, hãy kiểm tra mạch điện này thực hiện trên FPGA có chạy đúng theo chức năng mong muốn hay không.

1.6. Sơ đồ khối hệ thống

1.6.1. Tổng quát hệ thống



1.6.2. Các khối và chức năng chính.

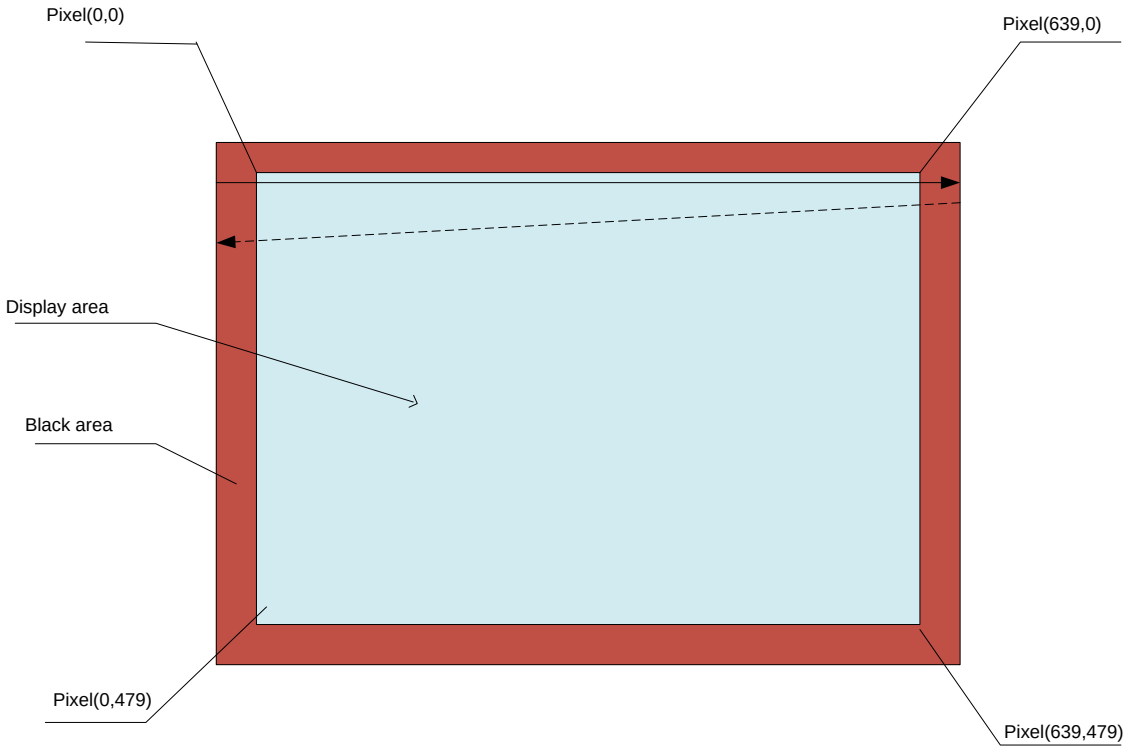
Khối	Chức năng
PS2	Nhận tín hiệu từ bàn phím, gửi tín hiệu điều khiển.
VGA_SYNC	Điều khiển việc hiển thị trên VGA
ITEMS	Tạo mồi cho rắn.
SNAKE	Khối điều khiển hành vi của rắn.
MAPS	Tạo các chướng ngại vật khác nhau trong màn chơi.
TEXTS	Hiển thị thông tin cho người dùng.
FSM	Khối điều khiển hệ thống, liên kết các khối khác.
TIMER	Bộ đếm thời gian, tạo độ trễ.
SPEED	Bộ chỉnh tốc độ di chuyển của rắn.
LEVEL	Bộ tính toán cấp bậc người chơi.

2. Thiết kế chi tiết

2.1.Khối VGA

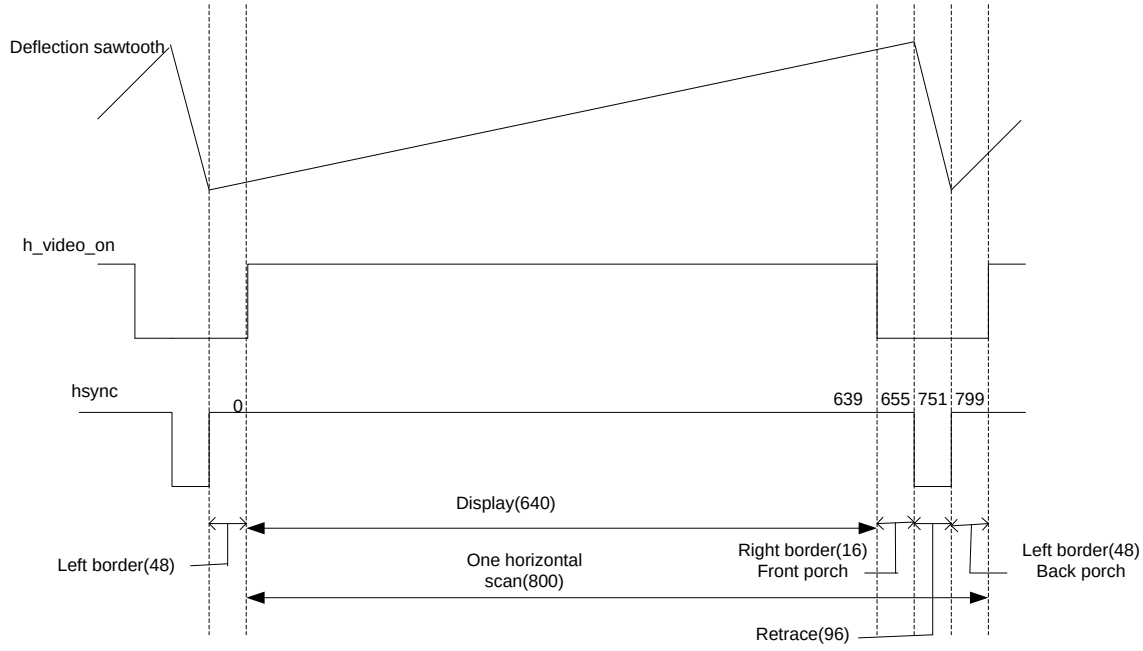
Giới thiệu về màn hình CRT 640x480

640x480 là số điểm ảnh và đường có thể hiện thị được trên màn hình.Mỗi đường ngang gồm 640 điểm ảnh, 480 là số đường ngang cần quét nhưng thực tế là mỗi đường ngang gồm 800 điểm ảnh và só đường ngang cần quét là 525.Số điểm ảnh và đường không được hiển thị được gọi là *black border*(biên đen).Tần số hoạt động là 25MHz.



1.1.1. Trục ngang đồng bộ

Trục ngang có 800 điểm ảnh được chia làm 4 vùng



- Dislay: vùng mà các điểm ảnh thực được hiển thị có độ dài 640px.

- Retrace: vùng mà trùm electron quay trở lại góc trái,tín hiệu video nên được tắt,có độ dài 96px.
- Right border(biên phải): vùng tạo thành biên phải của vùng hiển thị được gọi là front porch(cổng trước),tín hiệu nên được tắt,có độ dài 16px
- Left border(biên trái): vùng tạo thành biên trái của vùng hiển thị được gọi là back porch(cổng sau),tín hiệu nên được tắt,có độ dài 48px.

Đoạn code sau xác định các đại lượng cho trục ngang:

```
-- VGA 640-by-480 sync parameters
  constant HD: integer:=640; --horizontal display area
  constant HF: integer:=16 ; --h. front porch
  constant HB: integer:=48 ; --h. back porch
  constant HR: integer:=96 ; --h. retrace
```

Độ dài của left and right border có thể thay đổi giữa các màn hình khác nhau. Tín hiệu hsync cần có thêm bộ đếm 800 điểm ảnh và một mạch giải mã.Khi bắt đầu việc hiển thị thì bộ đếm cũng bắt đầu đếm và tín hiệu ra hợp thành tín hiệu pixel_x.Tín hiệu hsync ở mức thấp khi tín hiệu ra của bộ đếm nằm trong khoảng 656 va 751.

Chúng ta sử dùng tín hiệu video_on để điều chỉnh hiển thị/không hiển thị khi bộ đếm có giá trị nhỏ hơn 640

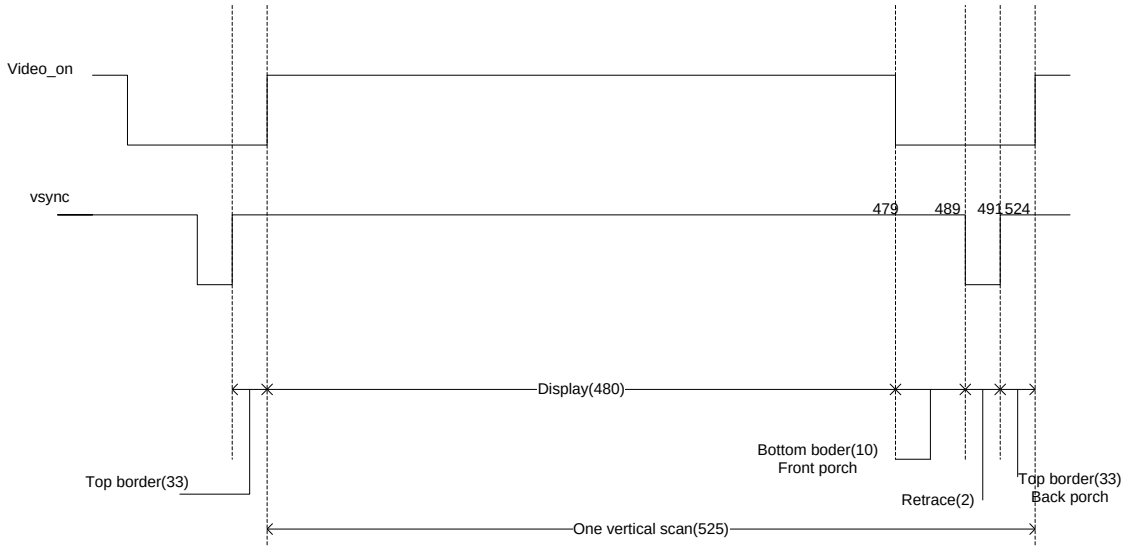
Màn hình CRT nên được để đen ở biên trái và phải và trong suốt quá trình retrace.

1.1.2 Trục dọc đồng bộ

Trong suốt quá trình quét dọc, chùm electron di chuyển đều đặn từ đầu tới cuối cùng màn hình,sau đó lại quay trờ lại đầu màn hình.Sự giống nhau này yêu cầu một khoảng thời gian để làm tươi màn hình.Cấu trúc của vsync tương tự với hsync.

Một chu kỳ của tín hiệu vsync là 525 đường và được chia làm 4 khu vực giống như hsync.

Các đặc điểm của từng khu vực cũng tương tự như hsync



Tín hiệu vsync cần 1 bộ đếm 525 dòng và một mạch giải mã.Bắt đầu đếm khi bắt đầu khu vực hiện thị.tín hiệu đầu ra của bộ đếm là pixel_y.Tín hiệu vsync ở mức thấp khi bộ đếm dòng ở dòng 490 hoặc 491.

Cũng như hsync,chúng ta sử dụng video_on để hiển thị/không hiển thị khi bộ đếm có giá trị nhỏ hơn 480.

1.1.3 Cách tính thời gian của tín hiệu VGA đồng bộ

Ở phạm vi project này chúng ta sự dụng tần số là 25MHz.Sự lựa chọn này được quyết định bởi 3 đại lượng:

- p: số điểm ảnh trên một đường quét ngang. p = 800 pixels/line
- l: tổng số đường trong màn hình. l = 525 lines/screen
- s: số khung ảnh trên một giây. s = 60 screens/second

Chọn s = 60 ở đây là vì mắt người hoạt động tốt ở khung hình này và chống được sự nhấp nháy.

Vậy pixel rate = p*l*s = 25M (pixel/second)

1.1.4 Hoàn tất VGA Graphic

Ở trên chúng ta đã thiết kế 2 bộ đếm.Vấn đề thiết kế ở đây là KIT DE1 chỉ hỗ trợ tần số 50MHz mà yêu cầu là 25MHz.Vì vậy do yêu cầu thiết kế chúng ta tạo ra 1 bộ 25MHz cho phép đánh dấu để tạm dừng hoặc cho phép việc đếm.Tín hiệu p_tick là một tín hiệu ra thực hiện công việc này và phối hợp với sự hoạt động của mạch the pixel generation.

Sử dụng 2 tiến hiệu h_end và v_end để kiểm tra việc hoàn thành quét ngang và dọc.

Ngoài ra để tránh khỏi tình trạng nhiễu chúng ta cần sử dụng thêm những bộ đệm được chèn thêm vào các tín hiệu hsync và vsync.

Đoạn code sau dùng để tạo ra bộ mod-2 counter dùng để dánh dấu thời điểm:

```
-- mod-2 circuit to generate 25 MHz enable tick
  mod2_next <= not mod2_reg;
  -- 25 MHz pixel tick
  pixel_tick <= '1' when mod2_reg='1' else '0';
```

Đoạn code sau dùng để nhận diện việc hoàn thành việc quét ngang:

```
h_end <=  -- end of horizontal counter
    '1' when h_count_reg=(HD+HF+HB+HR-1) else --799  '0'; //trừ đi 1 vì chúng ta đếm từ 0//
```

Đoạn code sau dùng để loại trừ nhiễu:

```
-- horizontal and vertical sync, buffered to avoid glitch
  h_sync_next <= '1' when (h_count_reg>=(HD+HF))          --656
                 and (h_count_reg<=(HD+HF+HR-1)) else –751
```

'0';

2.2.Khối PS2

Mạch trên bao gồm một bộ nhận tín hiệu trong đó có chèn thêm một khối chống nhiễu tín hiệu, một khối FSM để điều khuyển truyền nhận scan code, khối FIFO để ghi dữ liệu và truyền ra ngoài theo cơ chế vào trước ra trước(first in first out).

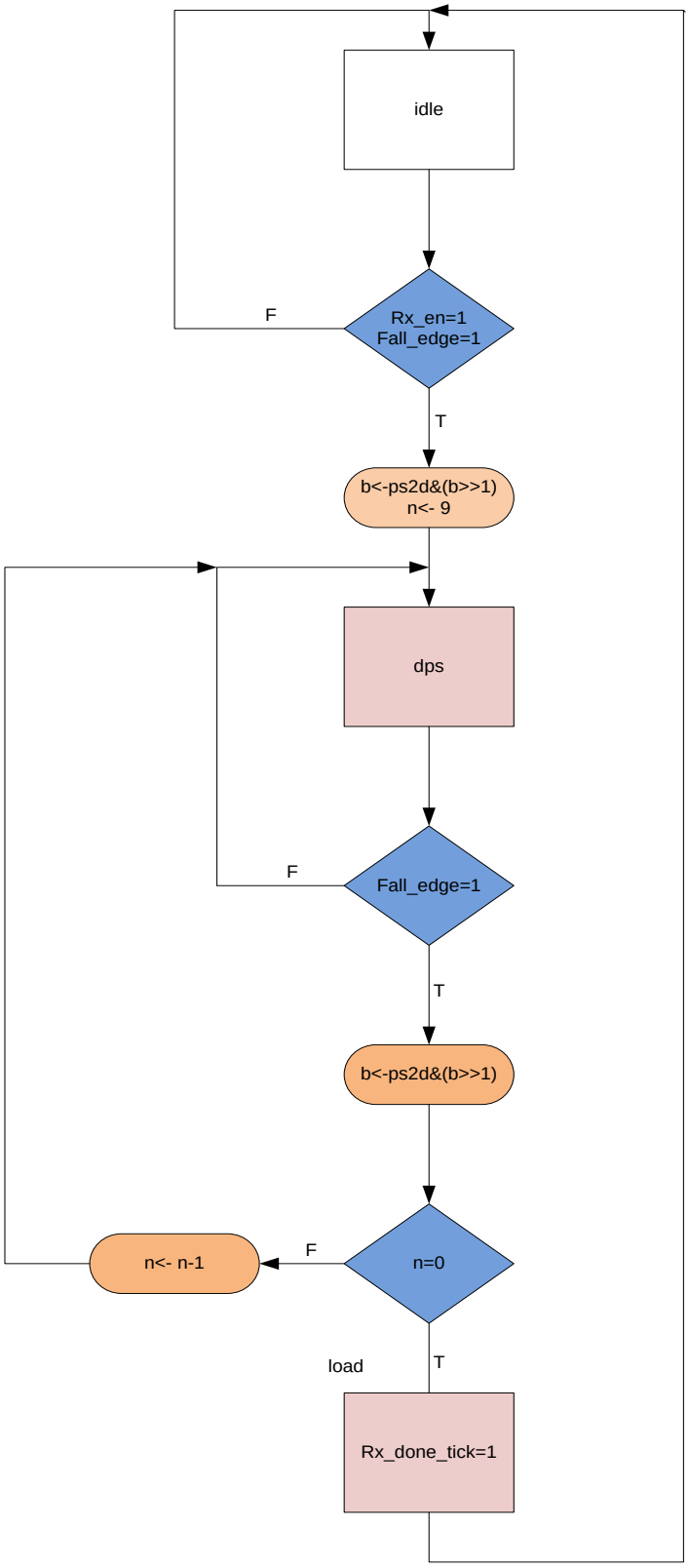
2.1 Khối nhận dữ liệu (PS2_rx)

Đoạn code sau dùng để chống nhiễu cho tín hiệu :

ps2cdeb <= '0' when ps2creg = "1111111111111111" else //trả về 0 khi tất cả các bit là 1

'1' when ps2creg = "0000000000000000" else //trả về 1 khi tất cả các bit là 0;

Để có thể hiểu cơ chế nhận dữ liệu của PS2 ta xét lưu đồ sau:



ASMD chart of PS2 port receiver

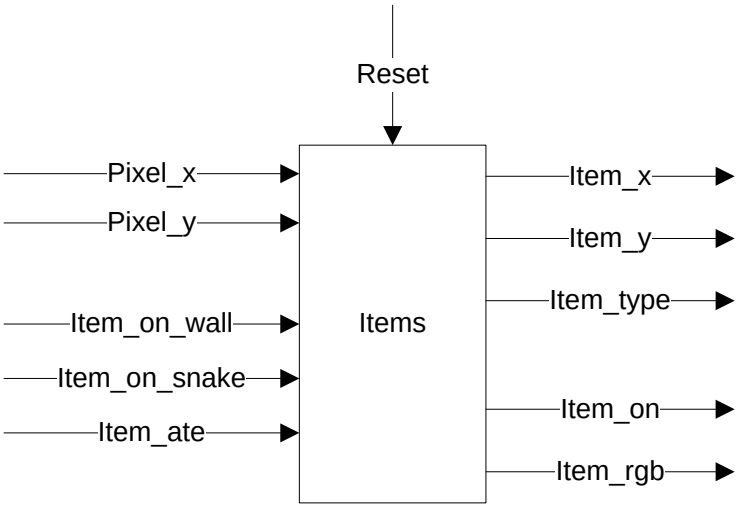
Khi có tín hiệu cho phép nhận rx_en và tín hiệu tích cực sườn xuống thì trạng thái sẽ dịch đến start bit và di chuyển đến trạng thái dps.

Kể từ khi dữ liệu được nhận trong một khối đã được kiểm tra, chúng ta sẽ dịch lại 10bit ở trong một trang thái riêng hơn là sử dụng các trạng thái riêng biệt như là data, parity, stop.

Sau đó mạch sẽ chuyển tới trạng thái load trong đó có một chu kỳ tín hiệu clock thêm được phân phối để hoàn thành quá trình dịch đến bit stop, và tín hiệu rx_done_tick được chèn vào sau 1 chu kỳ để thông bao khi đã nhận xong dữ liệu.

2.3.Khối ITEMS

2.3.1. Mô tả chung



Hình 2. Khối ITEMS

Khối ITEMS có nhiệm vụ sinh ra mồi cho rắn, với vị trí mồi thỏa mã các điều kiện:

- Mồi mới được tạo ra nếu mồi bị ăn, hoặc ngay khi vừa tạo ra đã trùng mới một đốt rắn nào đó, hoặc trung vào vật cản.

- Mồi được tạo ra tại vị trí ngẫu nhiên, không phụ thuộc vào vị trí mồi trước đó.
- Mồi mới sinh ra phải nằm trong vùng di chuyển được của rắn.
- Mồi mới được ra có vị trí không trùng bất kỳ đốt rắn nào, nếu nó nắm trên rắn, phải tạo lại mồi mới.
- Mồi mới sinh ra không được trùng vật cản nào, nếu không, phải tạo lại mồi mới.
- Các loại mồi đặc biệt sẽ biến mất sau một khoảng thời gian nhất định.

Loại	Hình dạng	Thời gian biến mất	Điểm	Tính năng phụ
1	Quả táo		1	
2	Sao	5s	1	Giảm tốc độ
3	Trái tim	5s	1	Tăng lượt chơi lại
--	--	--	--	--

Các đầu vào, đầu ra được khai báo như sau:

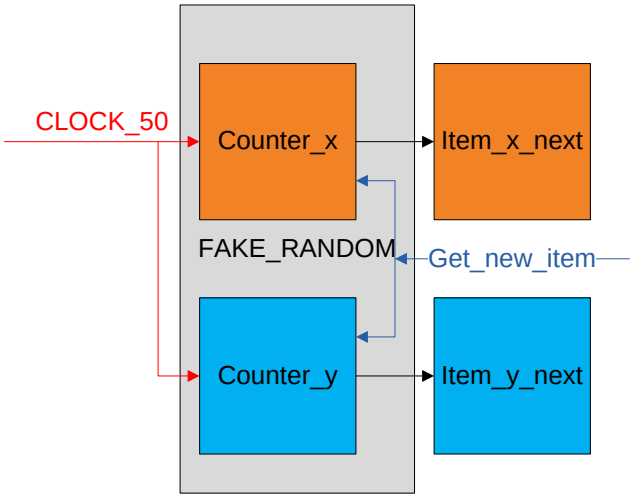
```
entity items is
port(
-- main clock
clk: in std_logic;
reset: in std_logic;
-- tọa độ điểm ảnh đang được quét
pixel_x: in std_logic_vector (9 downto 0);
pixel_y: in std_logic_vector (9 downto 0);
-- cấp độ hiện tại để xuất hiện mồi khác nhau phù hợp cấp độ
```

```
        level: in integer range 1 to 7;
    item_x: out unsigned(9 downto 0);
    item_y: out unsigned(9 downto 0);
    pause: in std_logic;
    item_on_wall: in std_logic;
    item_on_snake: in std_logic;
    item_ate: in std_logic;
    timer2_reset: out std_logic;
    timer2_start: out std_logic;
    timer2_up: in std_logic;
    item_type: out integer range 1 to 7:=1;
    item_on: out std_logic;
        item_rgb: out std_logic_vector (2 downto 0)
        );
end items;
```

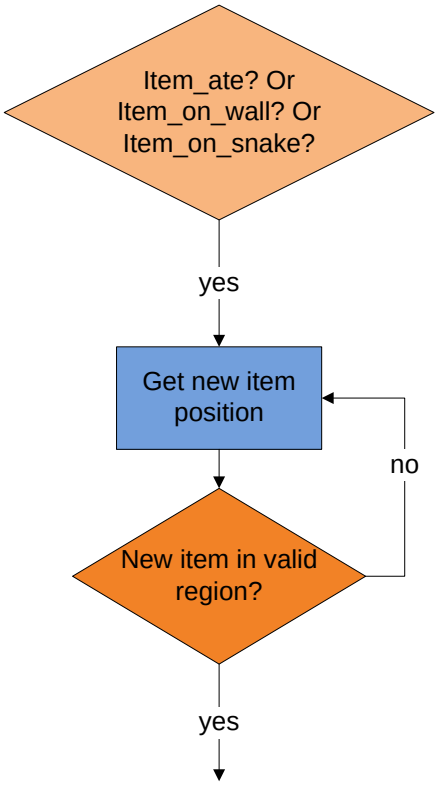
Tạo rom cho Items:

Chúng ta mô tả ROM qua mảng các bit, cụ thể như sau:

Tạo số ngẫu nhiên:



We use two counters as random number generator. with 50Mhz clock, and unknown time when item ate, value of counter seem to be random.



Item position must be checked untill we have a valid value

Để tạo số ngẫu nhiên ta dùng các bộ đếm quay vòng và giá trị bộ đếm được lấy ra bất cứ lúc nào ta muốn, để đảm bảo tính ngẫu nhiên thì thời gian quay vòng bộ đếm nhỏ hơn nhiều so với thời gian trung bình lấy giá trị từ bộ đếm

Ta sử dụng bộ đếm có giá trị nhỏ, nhưng lặp lại với tần số nhanh, đó là clock 50Mhz.

Bộ đếm 1 được sử dụng cho khoảng giá trị tọa độ x theo phương ngang của mồi

Bộ đếm 2 sử dụng để tạo ra giá trị tọa đọ y theo phương thẳng đứng của mồi

Kết hợp 2 giá trị này ta sẽ có được tọa độ của mồi mới được sinh ra, và có vẻ gần như là ngẫu nhiêu.

Code :

2.4.Khối SNAKE

2.4.1. Sơ đồ

2.4.2. Code

2.5.Khối MAPS

2.6.Khối TEXTS

2.6.1. Font

```
-- ROM with synchonous read (inferring Block RAM)
--  character ROM
--   - 8-by-16 (8-by-2^4) font
--   - 128 (2^7) characters
--   - ROM size: 512-by-8 (2^11-by-8) bits
--              16K bits: 1 BRAM

library ieee;
    use ieee.std_logic_1164.all;
    use ieee.numeric_std.all;

entity font_rom is
  port(
     clk: in std_logic;
     addr: in std_logic_vector(10 downto 0);
```

```
    data: out std_logic_vector(0 to 7)
  );
end font_rom;

architecture arch of font_rom is
  constant ADDR_WIDTH: integer:=11;
  constant DATA_WIDTH: integer:=8;
  signal addr_reg: std_logic_vector(ADDR_WIDTH-1 downto 0);
  type rom_type is array (0 to 2**ADDR_WIDTH-1)
      of std_logic_vector(0 to DATA_WIDTH-1);
  -- ROM definition
  constant ROM: rom_type:=(   -- 2^11-by-8
  "00000000", -- 0
  "00000000", -- 1
  "00000000", -- 2
  "00000000", -- 3
  "00000000", -- 4
  "00000000", -- 5
  "00000000", -- 6
  "00000000", -- 7
  "00000000", -- 8
  "00000000", -- 9
  "00000000", -- a
  "00000000", -- b
  "00000000", -- c
  "00000000", -- d
```

```
    "00000000", -- e
    "00000000", -- f
    -- code x01
    "00000000", -- 0
    "00000000", -- 1
    "01111110", -- 2  ******
    "10000001", -- 3 *      *
    "10100101", -- 4 * *  * *
    "10000001", -- 5 *      *
    "10000001", -- 6 *      *
    "10111101", -- 7 * **** *
    "10011001", -- 8 *  **  *
    "10000001", -- 9 *      *
    "10000001", -- a *      *
    "01111110", -- b  ******
    "00000000", -- c
    "00000000", -- d
    "00000000", -- e
    "00000000", -- f
   …
  );
begin
    -- addr register to infer block RAM
    process (clk)
    begin
      if (clk'event and clk = '1') then
```

```
        addr_reg <= addr;
      end if;
    end process;
    data <= ROM(to_integer(unsigned(addr_reg)));
  end arch;
```

2.7.Khối FSM

2.7.1. Sơ đồ

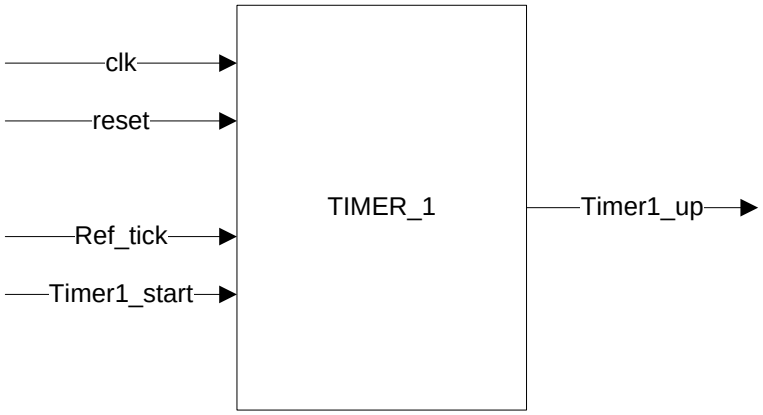
2.7.2. Code

2.8.Khối phụ

2.8.1. TIMER_1

Đây là khối đếm thời gian 2s nhằm tạo độ trễ giữa các trạng thái của máy trạng thái FSM, hữu ích khi muốn thông báo điều gì đó trên màn hình. Ở đây là các thông báo GAME OVER, STAGE CLEAR, LEVEL UP…

Mô hình:



Khai báo các đầu vào, ra:

```
entity timer is
    port(
        -- Main-clock input at 50MHz
        clk:                in std_logic;
```

```
        -- Synchronous reset
        reset:          in std_logic;
        -- 60Hz-clock input is rate of monitor
        refr_tick:      in std_logic;
        -- start signal
        timer_start:    in std_logic;
        -- output
        timer_up:       out std_logic
    );
end timer;
```

Chúng ta sẽ dùng bộ đếm để làm nhiện vụ tính thời gian, vì vậy chúng ta lưu giá trị đếm vào một thanh ghi, có thể đếm tăng, đếm giảm tùy ý nhưng cách đếm giảm có lợi hơn khi so sánh giá trị của thanh ghi với giá trị 0.

Ở đây chúng ta cần đếm thời gian khoảng 2 giây, với đầu vào là một tín hiệu refresh của màn hình, vào khoảng 60Hz nên 2 giây sẽ tương ứng với giá trị của thanh ghi là 120.

Ta chọn thanh ghi có 7 bit, tức giá trị max của nó là 127, tức là khoảng thời gian trễ là 127/60 = 2.1 giây, sai số vẫn chấp nhận được.

Code:

```
architecture arch of timer is
    -- max value in register is 127, max counter value is about 2 seconds
    signal timer_reg, timer_next: unsigned(6 downto 0);
begin
```

Phần code tiếp theo xử lý cách đếm thời gian, trước tiên là xử lý thanh ghi, việc dùng clock 50Mhz làm clock chính nhằm tăng tính đồng bộ cho hệ thống

Chúng ta triển khai bộ đếm dạng máy trạng thái nhưng không có bất cứ trạng thái nào được tạo ram chỉ có điều thanh ghi giá trị chia làm hai thanh ghi : timer_reg chỉ giá trị bộ đếm hiện tại, và timer_next chỉ giá trị bộ đếm tiếp sau đó nếu có clock xảy ra.

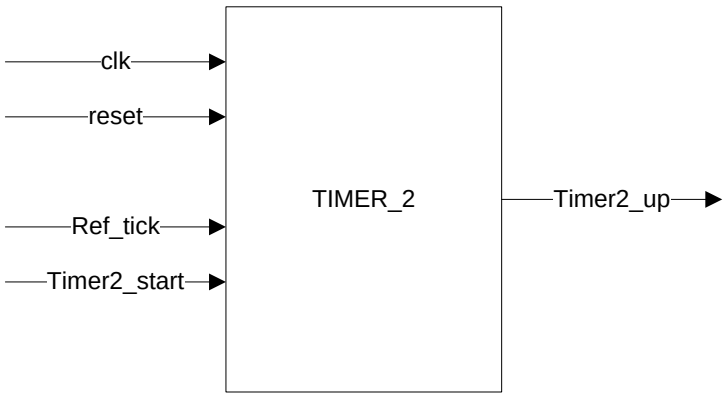
Code:

```
-- registers
process (clk, reset)
begin
	if (clk'event and clk='1') then
		if reset = ‘1’ then
			timer_reg <= (others=>'1');
		else
			timer_reg <= timer_next;
		end if;
	end if;
end process;
-- next-state logic
process(timer_start,timer_reg,refr_tick)
begin
	if (timer_start='1') then
		timer_next <= (others=>'1');
	elsif refr_tick='1' and timer_reg/=0 then
		timer_next <= timer_reg - 1;
```

```
            else
                timer_next <= timer_reg;
            end if;
        end process;
        -- output
        timer_up <='1' when timer_reg=0 else '0';
end arch;
```

2.8.2. TIMER_2

Tương tự TIMER_1, TIMER_2 cũng là bộ đếm, nhưng lại được sử dụng vào mục đích khác, đó là thời gian hiển thị của một các loại mồi đặc biệt có trong trò chơi, chẳng hạn LIVE_UP, SPEED_DOWN … Các loại mồi này hiển thị trong một khoảng thời gian nhất định rồi biến mất nếu trong thời gian đó mà rắn không kịp ăn mồi.



Giống như TIMER_1, chúng ta đếm thời gian qua thanh ghi với nhịp clock 60Hz thông qua giá trị của thanh ghi. Khác nhau duy nhất đó là đọ dài thanh ghi tức là giá trị max mà thanh ghi lưu trữ.

Ở đây thanh ghi sẽ là:

```
        signal timer2_reg, timer2_next: unsigned(9 downto 0);
```

có 10 bit, với giá trị lưu max = 511, như vậy độ trễ lớn nhất là 511 / 60 = 9 giây.

Việc xử lý TIMER_2 giống hệt như TIMER_1 nên sẽ không nói ở đây nữa.

2.8.3. LEVEL

Khối Level theo dõi số lượng mồi đã ăn được của rắn, sao cho khi rắn ăn một lượng mồi nhất định, cấp bậc của người chơi sẽ được tăng lên, kèm theo đó là tốc độ di chuyển rắn tăng lên nhằm tăng mức độ khó cho màn chơi.

Sau khi người chơi bắt đầu từ cấp độ 1 và đạt đến cấp độ 5, màn chơi cũng sẽ thay đổi, chúng ta cần kiểm soát khi nào màn chơi thay đổi, lúc này cần thêm một đoạn code dành cho việc xem xét các cấp độ đã đạt được.

Các chân tín hiệu cần thiết được mô tả như sau:

```
entity level is
  port(
    -- main clock
     clk: in std_logic;
      restart: in std_logic_vector(1 downto 0);
    -- mồi đã bị ăn
      item_ate: in std_logic;
    -- tín hiệu thông báo tăng cấp độ
      level_up: out std_logic;
```

```
    -- tín hiệu báo đã qua màn chơi.
      stage_clear: out std_logic;
    -- màn chơi hiện tại của người chơi.
      stage_select_in: in integer range 1 to 7;
      stage_select_out: out integer range 1 to 7;
    -- thông báo cấp độ hiện tại của người chơi.
      level: out integer range 1 to 7
  );
end level;
```

Khối code kiểm soát cấp độ người chơi dựa vào số mồi đã ăn.

Số mồi cần ăn để lên cấp độ cao hơn là 5 lần cấp độ hiện tại, ví dụ từ cấp độ 2, muốn lên cấp độ 3 cần ăn 2*5=10 mồi.

```
                elsif item_ate='1' then
                        if counter_next_level = 5*level_tmp then
                                level_tmp := level_tmp + 1;
```

code này sẽ kiểm soát màn chơi nếu người chơi đã đạt cấp độ 6, người chơi sẽ được nâng lên màn chơi khó hơn

```
                                if level_tmp=6 then
                                        if stage_sel_tmp<2 then
                                        stage_sel_tmp:= stage_sel_tmp+1;
                                        end if;
                                        stage_clear_tmp:= '1';
```

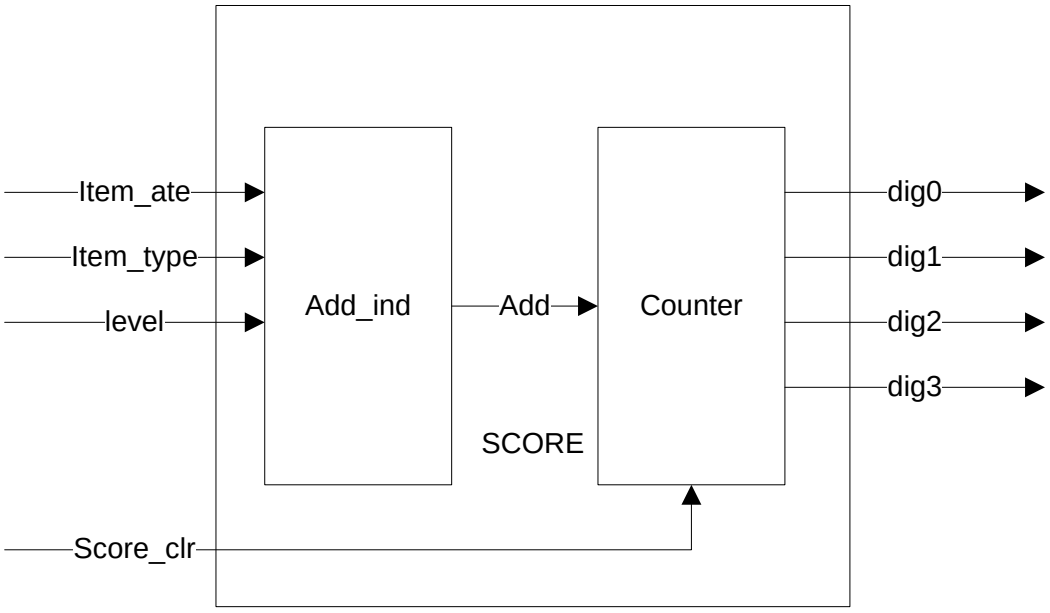
đồng thời cấp bậc trong màn chơi mới sẽ được bắt đầu lại từ cấp 1.

```
                                        level_tmp:=1;
                                end if;
```

```
                    level_up_tmp := '1';
                    counter_next_level := 0;
                else
                counter_next_level := counter_next_level + 1;
                end if;
            end if;
            level <= level_tmp;
            level_up <= level_up_tmp;
            stage_clear <= stage_clear_tmp;
            stage_select_out <= stage_sel_tmp;
        end if;
    end process;
end arch;
```

2.8.4. SCORES

2.8.4.1. Sơ đồ



- Khối này có nhiệm vụ tính điểm cho người chơi, và số điểm nhận được sẽ tùy thuộc vào loại mồi mà rắn ăn, level hiện tại của rắn(tốc độ càng nhanh, level càng cao, càng nhận được hiều điểm).
- Khối nhận tín hiệu từ khối ITEMS gồm Item_ate, và Item_type, sau khi khối Add_ind tính toán lượng điểm mà ngưii chơi được cộng thêm, nó sẽ tạo ra xung Add để khối Counter tính điểm. Ví dụ khi Item_ate = 1, Item_type=3, Level = 2 thì số điểm nhân được là 3 + 2 = 5, lúc này khối Add_Ind sẽ phát ra 5 tín hiệu Add cho bộ Counter .
- Score_lcr nhắm reset số điểm về 0000 khi màn chơi mới được bắt đầu

2.8.4.2. Counter_9999

Như đã nói ở trên, khối COUNTER_9999 nhận tín hiệu cộng qua chân d_inc, mỗi lần tín hiệu này từ mức thấp lên mức cao, giá trị của bộ đếm sẽ tăng lên 1.

Các chân của khối được mô tả như sau:

```
entity counter9999 is
    port(
            -- main clock
            clk: in std_logic;
            -- reset
            reset: in std_logic;
            -- addition indicate
            d_inc: in std_logic;
            -- đầu ra là các số riêng biệt, thuận lợi cho việc đưa vào bộ
TEXTS để biến số thành chữ số, phục vụ việc hiển thị lên màn hình.
```

```
            dig0: out std_logic_vector (3 downto 0);
            dig1: out std_logic_vector (3 downto 0);
            dig2: out std_logic_vector (3 downto 0);
            dig3: out std_logic_vector (3 downto 0)
    );
end counter9999;
------------------------------------------------------------
architecture arch of counter9999 is
    -- các thanh ghi lưu giá trị hiện tại và tiếp theo của 4 biến đếm.
    signal dig0_reg, dig1_reg,dig2_reg,dig3_reg: unsigned(3 downto 0);
    signal dig0_next, dig1_next,dig2_next,dig3_next:   unsigned(3 downto
0);
begin
-- registers
    process (clk, reset)
    begin
            if (clk'event and clk='1') then
                    if reset='1' then
                            dig0_reg <= (others=>'0');
                            dig1_reg <= (others=>'0');
                            dig2_reg <= (others=>'0');
                            dig3_reg <= (others=>'0');
                    else
                            dig0_reg <= dig0_next;
                            dig1_reg <= dig1_next;
                            dig2_reg <= dig2_next;
```

```
                dig3_reg <= dig3_next;
            end if;
        end if;
    end process;
    -- next-state logic for the decimal counter
    process(reset, d_inc, dig0_reg, dig1_reg, dig2_reg, dig3_reg)
    begin
        if reset='1' then
            dig0_next <= (others=>'0');
            dig1_next <= (others=>'0');
            dig2_next <= (others=>'0');
            dig3_next <= (others=>'0');
        else
            dig0_next <= dig0_reg;
            dig1_next <= dig1_reg;
            dig2_next <= dig2_reg;
            dig3_next <= dig3_reg;
        end if;

        if (d_inc='1') then
            if dig0_reg=9 then
                dig0_next <= (others=>'0');
                if dig1_reg=9 then
                    dig1_next <= (others=>'0');
                    if dig2_reg=9 then
                        dig2_next <= (others=>'0');
```

```
                        if dig3_reg=9 then
                            dig3_next <= (others=>'0');
                        else
                            dig3_next <= dig3_reg + 1;
                        end if;
                    else
                        dig2_next <= dig2_reg + 1;
                    end if;
                else
                    dig1_next <= dig1_reg + 1;
                end if;
            else
                dig0_next <= dig0_reg + 1;
            end if;
        end if;
    end process;
    -- output
    dig0 <= std_logic_vector(dig0_reg);
    dig1 <= std_logic_vector(dig1_reg);
    dig2 <= std_logic_vector(dig2_reg);
    dig3 <= std_logic_vector(dig3_reg);
end arch;
```

2.8.4.3. Scores unit

Khối này mang tính chất là khối điều khiển khối con counter999 và giao tiếp với các khối cấp cao hơn.

Khai báo các chân tín hiệu như sau:

```
library ieee;
    use ieee.std_logic_1164.all;
entity score is
    port(
        -- main clock
        clk: in std_logic;
        -- tín hiệu xóa điểm về 0000
        score_clr: in std_logic;
        -- báo mồi đã bị ăn
        item_ate: in std_logic;
        -- loại mồi
        item_type: in integer range 1 to 7;
        -- cấp độ hiện tại của người chơi
        level: in integer range 1 to 7;
        -- đưa các chữ số ra cho bộ TEXTS
        dig0: out std_logic_vector (3 downto 0);
        dig1: out std_logic_vector (3 downto 0);
        dig2: out std_logic_vector (3 downto 0);
        dig3: out std_logic_vector (3 downto 0)
        );
end score;
```

chúng ta khai báo một component cho khối counter9999 và thực hiện nối tín hiệu cho khối.

```
    counter_u0: entity work.counter9999
    port map(…);
```

Phần quan trọng ở đây là cách tạo nhịp đếm tương ứng với đô điểm sẽ được cộng cho người chơi, nhịp đếm này sẽ đưa vào counter9999 thông qua biến counter và tín hiệu d_inc.

Chúng ta xử lý qua một process như sau:

```
process(clk, level, item_ate, item_type)
        -- biến lưu điều kiện
        variable add: std_logic;
        -- lưu giá trị cần cộng
        variable counter : integer range 0 to 63;
  begin
            if clk'event and clk='1' then
            -- nếu mồi bị ăn
                if item_ate='1' then
            -- xuất tín hiệu công cho bộ đếm
                    add:='1';
                end if;
                -- nếu có tín hiệu cho bộ đếm
                if add='1' then
                    -- tính toán lượng cộng thêm
                    if counter = level + item_type then
                        -- nếu cộng đủ thì nhả biến báo
cộng
                        add:= '0';
                        -- đưa counter về 0
                        counter:=0;
                    else
```

```
                    -- nếu không, đếm tiếp
                    counter:=counter+1;
                end if;
            end if;
            -- và đưa ra tín hiệu công cho counter9999
            d_inc <= add;
        end if;
    end process;
end arch;
```

2.8.5. SPEEDS

2.8.6. SNAKE_COUNTER