Hanoi University of Science and Technology

INSTITUTE OF ELECTRONIC - TELECOMMUNICATION

ESRC LAB

Topic:

SNAKE GAME on DE1 Kit

Groups of students:

Group 1 - K53:

Vu Quang Trong

Do Son Tung

Hanoi 8/2011

first. Introduce

1.1. Topic

After completion of the lab practice on Altera DE1 Kit, we

continue to develop their skills and apply into the design practice, which is deployed

a complete system on DE1 Kit with the theme:

"Using Altera DE1 Kit to create a Snake game

players with a graphical interface, communicate with players via keyboard

PS2 and VGA monitor. "

1.2. Members and assign jobs

Picture go here	Picture go here
Vu Quang Trong (leader)	Do Son Tung
0973.750.337	0168.9.929.537
vuquangtrong@gmail.com	tungmontaint@gmail.com
Mapping the overall theme. Logical block system status. Solid control block. Graphic display blocks. Display text blocks. And blocks other necessary accessories.	Learn connect PS2. Learn and VGA driver. Learn AudioCodec and control IC Sound volume.

1.3. Requirements of topic

1.3.1. require function

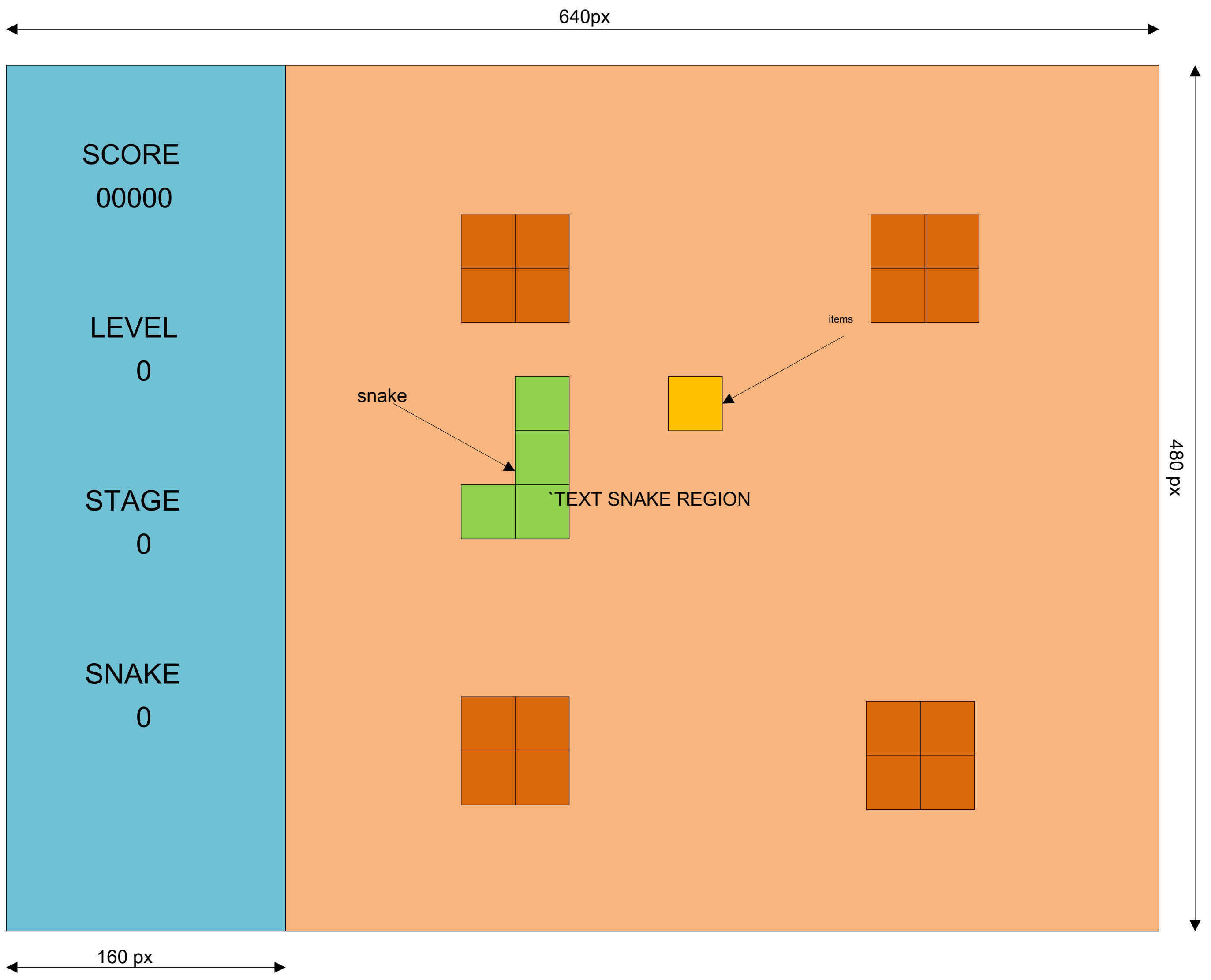
- Hardware
  - o The game runs entirely on Kit DE1
  - o Get control from the keyboard PS2

  - o Display on VGA screen resolution of 640 x 480
  - o Sound through a loudspeaker 2.0 with WM8731 audio codec IC has cassava on DE1 Kit

- Software
  - o Quartus II 9.1 SP2 using
  - o Pure VHDL language
  - o The game has all the elements to become a complete game:
    - The player controls Snake with 4 directional buttons on keyboard.
    - The keys function as PAUSE, SELECT dike people MENU play in game manipulation.
    - Long snake out when feeding, or have other interactions depending baits taste.
    - Commandments will die if the snake hit the walls or the tail it, at which point you will have a new solid replay threads.
    - Conditions to play again if and only if the number of plays of you greater than 0, the original, the plays will be 3, can if eating baits LIVE_UP increase, and decrease when the snake died. If the number of plays your back is 0, the game will end.
    - The different levels of difficulty, which is the speed of the snake will increase through a certain number of times Predator, followed by the game screen there various obstacles.
    - Scoring for the game, the score is calculated on the level current and baits that snakes eat.
    - Save name players when players reach a point the score in the top 5

1.3.2. Non-functional requirements

- Response time less keystrokes to help players navigation the easy controls, speed press about 4 times / sec => time 250ms response
- Displayed on the screen VGA 640x480 with 8 basic colors, use 3 bits for color.
- Frequency The screen is large enough to ensure the smooth display that we take is 60Hz scanning frequency.
- Control laws are applied as follows:
    - o Snakes can not turn back in the direction you travel, ie as if going forward, then the button will not work backwards, similar when teeth are left, right, or down.
- Frame of players:



Frame game screen is divided into two main frames:

- Information display frame of the player:
  - o Including Score, Level, Stage and number of players remaining solid.
  - o 160x480 size.
  - o Text is displayed with size 32x16.
- Frame display part to play:
  - o Size 480x480

o Snakes, prey, and the wall was built from blocks of 16x16 square match.

o This section can be to display a message when needed.

1.4. Recommended hardware

1.4.1. DE1 kit

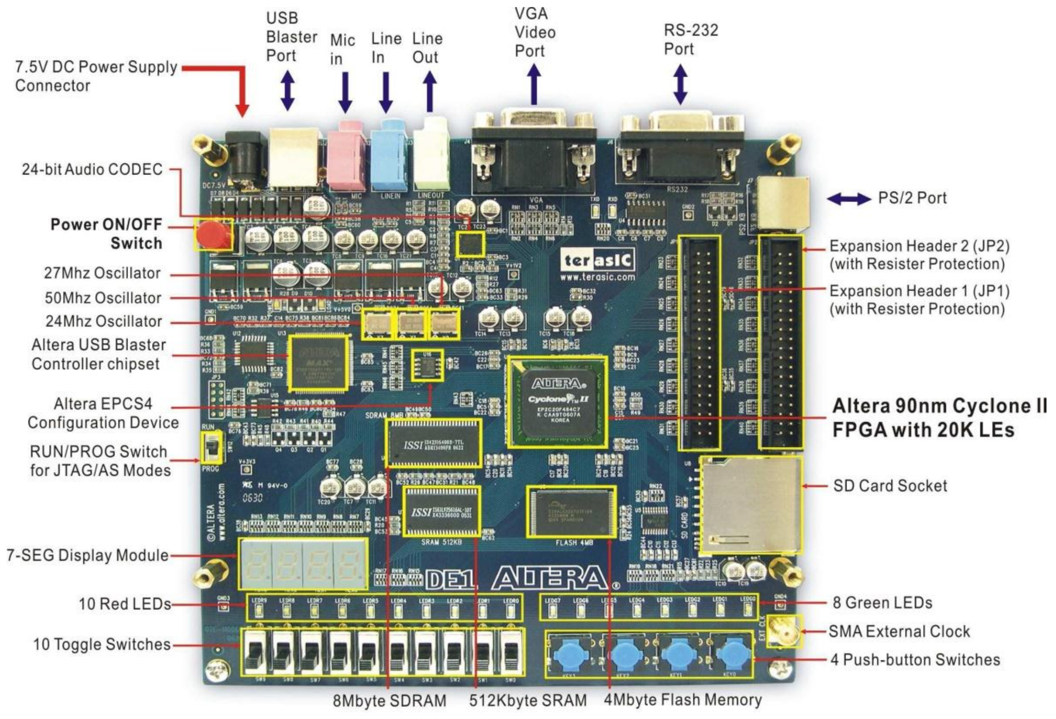
1.4.1.1. Introducing KIT DE1

KIT DE1 is a genuine product of the developer's landing Altera.Muc KIT DE1 when creating a tool that offers the ideal service for advanced design in some areas such as multimedia, storage, network ...

KIT DE1 to use, we need to connect to a computer running Microsoft Windows software.

Figure 1.1 shows a photograph of the DE1 package.

1.4.1.2. KIT components on 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 user LEDs red
- 8 user LEDs reen

- 50-MHz oscillator, 27-MHz and 24-MHz oscillator for clock oscillator

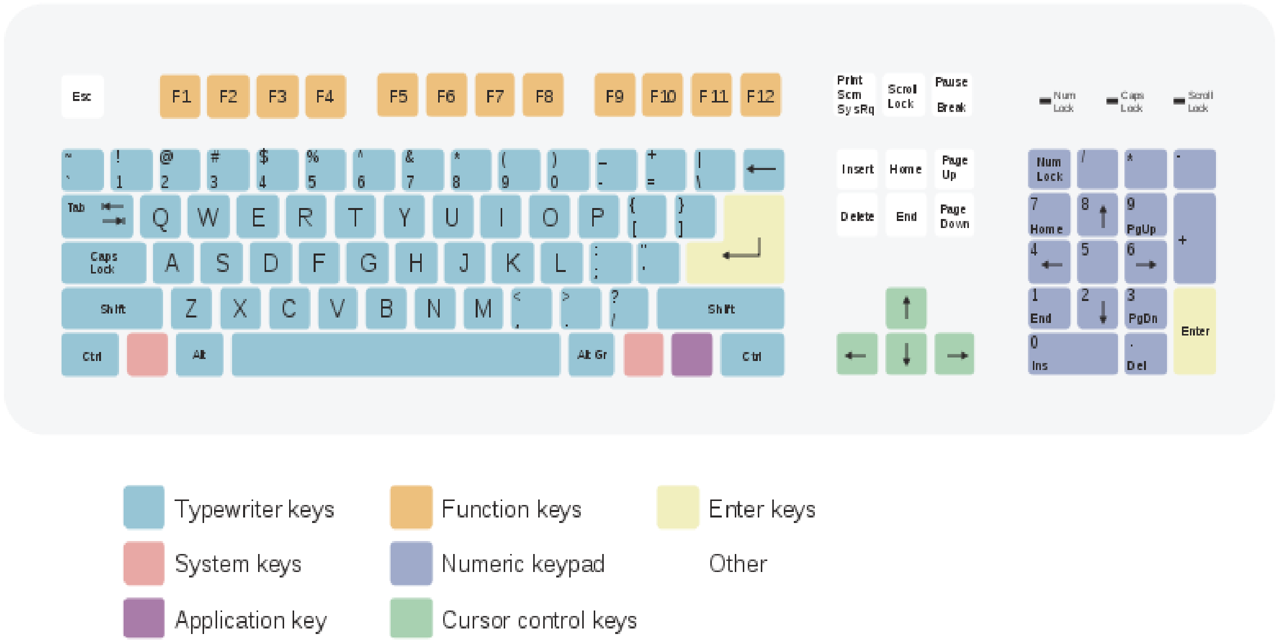
sources

- CD-quality 24-bit 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 hoặc a 7.5V DC adapter or a USB cable

1.4.2. PS2 Keyboard



With our range designed this project, only interested in the buttons

Moving dog thing: up, down, left, right. Two function keys

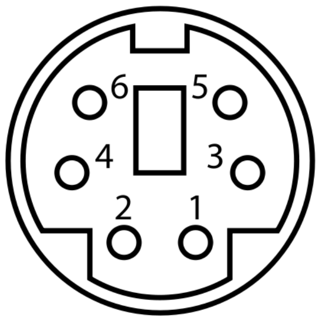
pause and options: Esc, Enter.Co can develop more other nodes

requirements for each subject.

Details on how to receive and transmit data key from the keyboard will be listed in section details the system.

Pinout PS2

Use standard PS2 connector to connect a keyboard with KIT DE1



Pin 1 + DATA Data

Pin 2 Not connected Not connected *

GND pin 3 Gr

Vcc +5 V DC battery 4 at 275 mA

Pin 5 + CLK

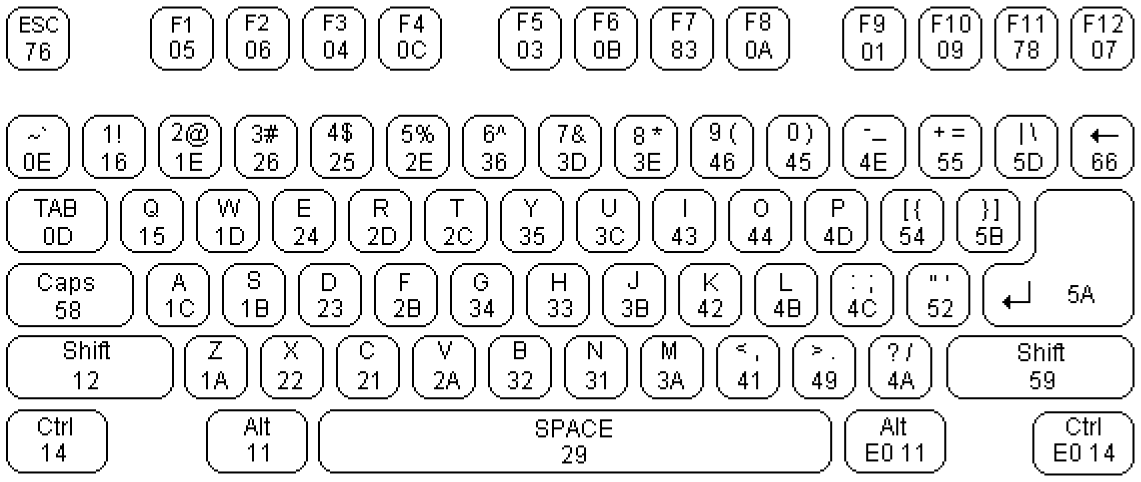
Pin 6 Not connected Not connected **

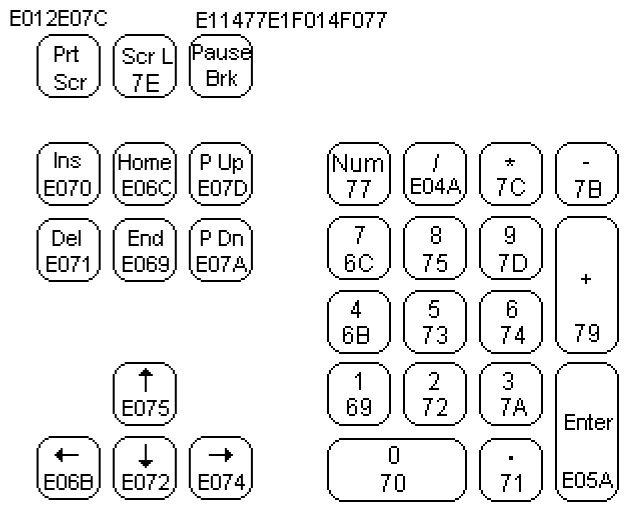
Nonstandard PS2 has some other common connection standard is USB and wireless (wireless). Scope of project topics connected only with the PS2 Should we introduce and learn about the other two standards.

PS2 interface connector is a 6-pin MINI DIN.

2.1.2.3. scan code

A keyboard includes a key matrix and an embedded microprocessor to check the operation of the keys and send *scan code* fit.





Operation of the keyboard:

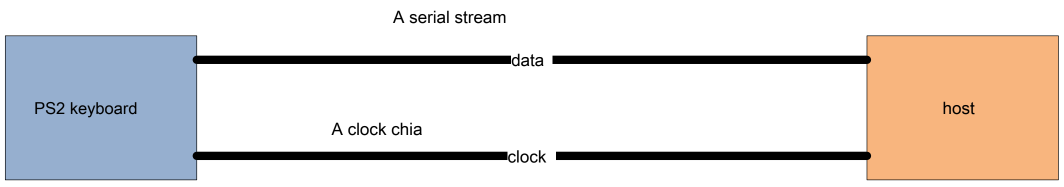
- When a key is pressed, the first *the make code* the keys are transmitted.

- When one key is kept constant, the state is known as *typematic then the make code* is transmitted continuously with a certain speed. In default mode making, one PS2 keyboard communications *make -code* about 100ms after 1 keys were kept for about 0.5s.

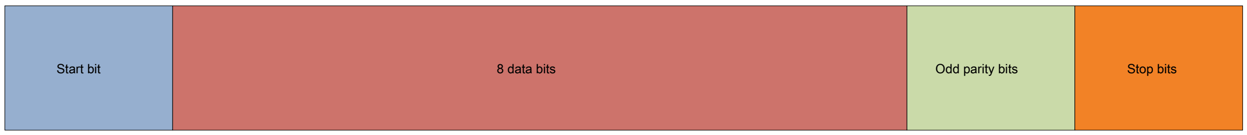
- When the key is released, the first *the break code (0xf0)* transmissions later *make code* of keys to identify which key has been released.

Way data transfer

A device PS2 (keyboard) connected to KIT DE1 and exchange data through the data path and clock 2



Road data includes 11 bits



Street clock was brought in a separate clock signal.

Data will be transmitted when the clock change and is actively low (falling-edge).

1.4.3. VGA Monitor

Introduce

VGA (Video Graphics Arrays: video graphics array) was introduced by IBM PCs powered by the PC graphics hardware and display design will hinh.Chung an interface consists of 8 basic colors with 640x480 screen CRT.

Basic mechanism of action of a CRT

Block diagram:

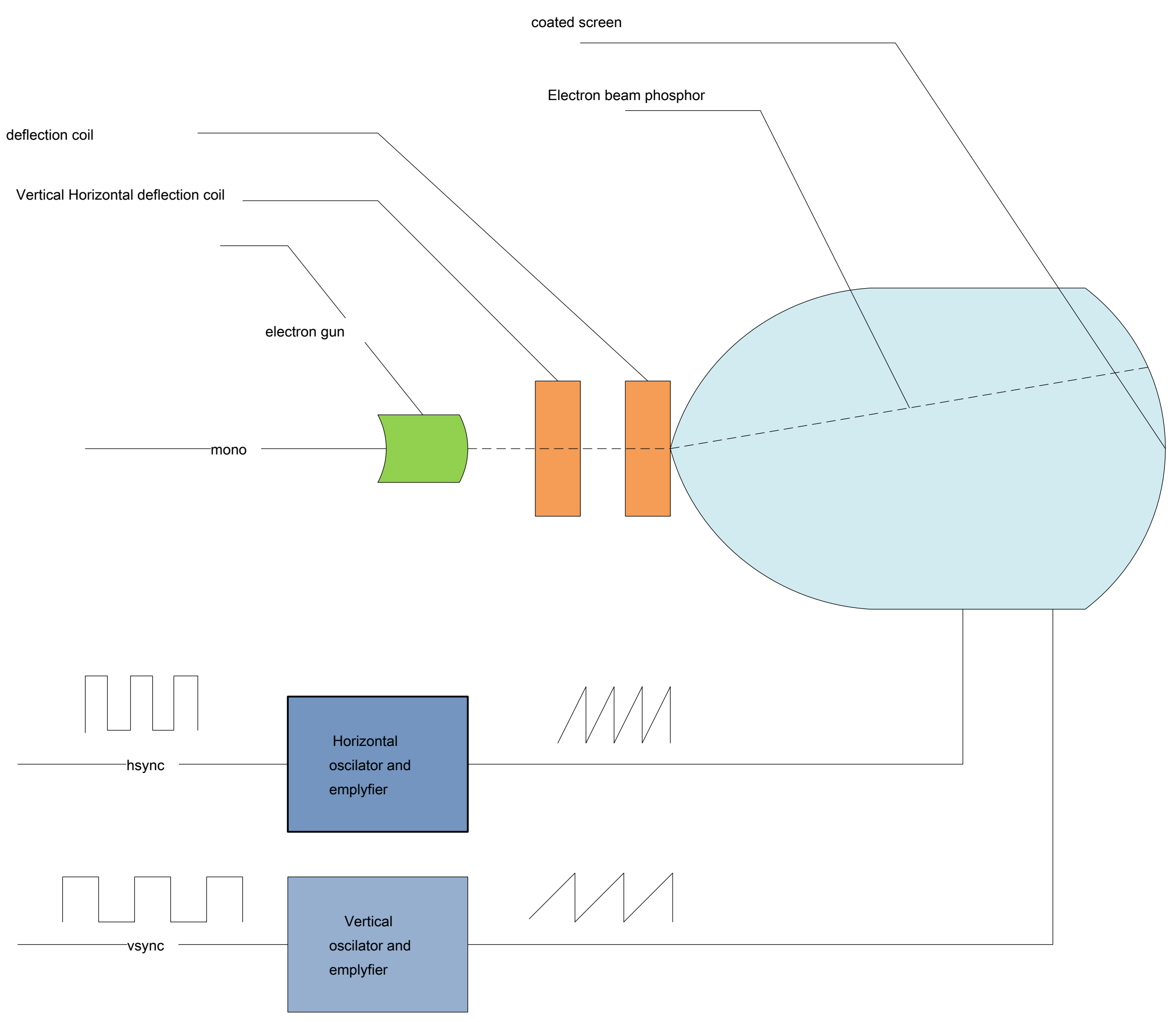
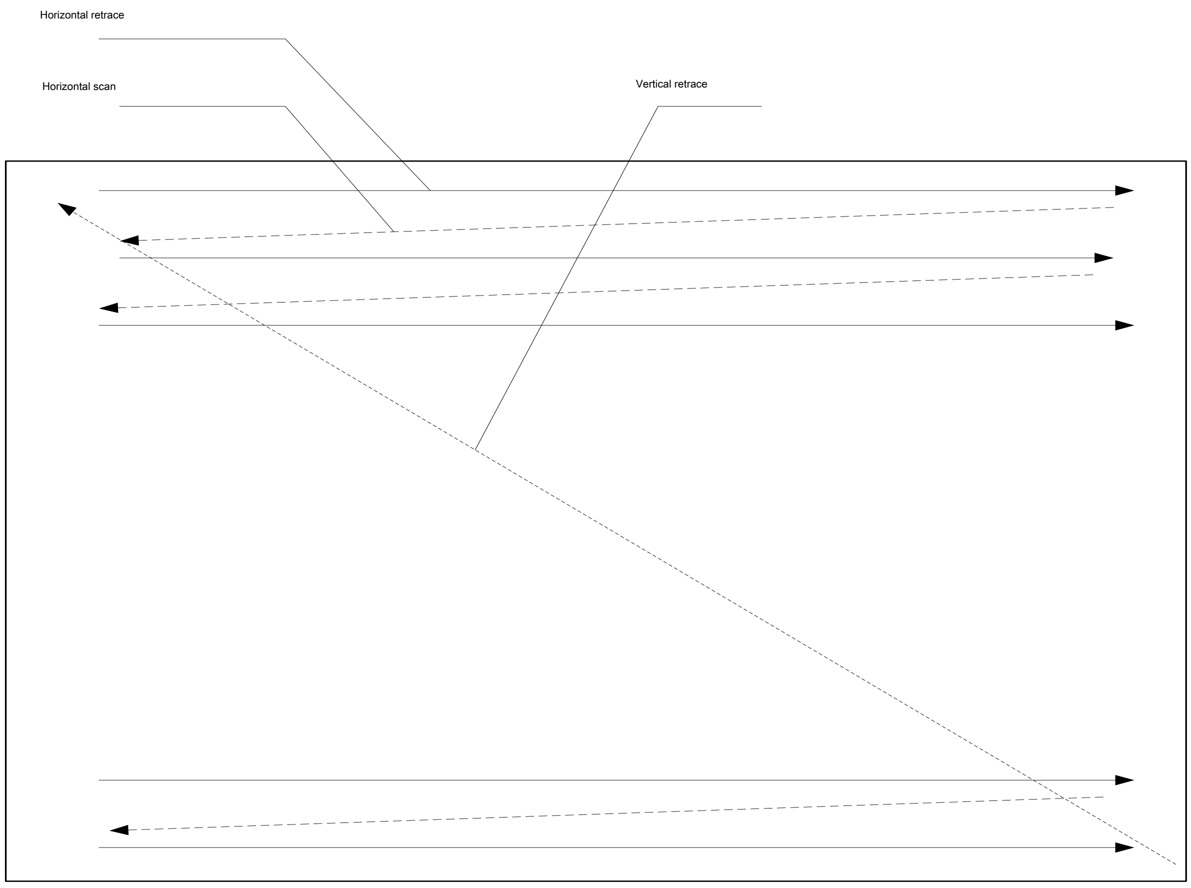

coated screen
Electron beam phosphor
deflection coil
Vertical Horizontal deflection coil
electron gun
mono
hsync
Horizontal oscilator and emplyfier
vsync
Vertical oscilator and emplyfier

Figure 1

- Electron beam intensity and brightness of the spot is determined by the level voltage input video signal, the signal is mono signal mono.Tin similar voltage levels between 0 and 0.7 changes.

- The vertical deflection coil and a horizontal deflection coil cruise control the flow of electrons and electron bean decide where on screen hinh.Voi screens today, boss electrons are driven from left to right from up to down.

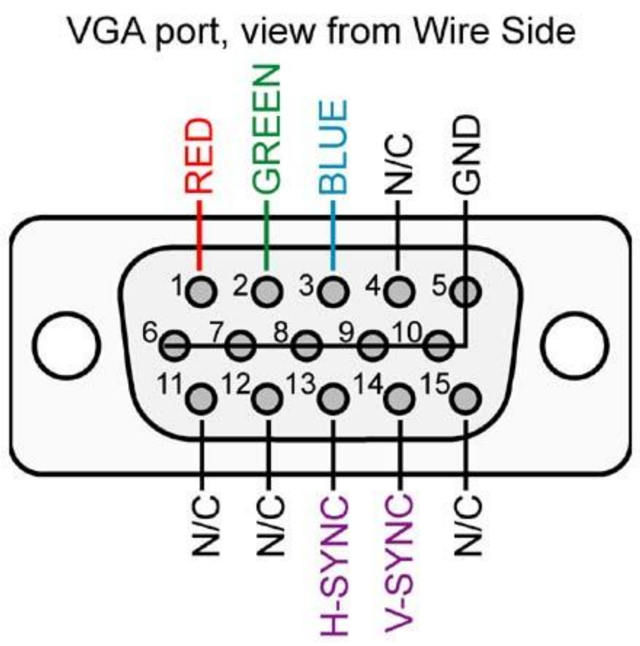
Scanning VGA



When voltage is applied to the deflection coil honrizontal and increase a steadily, the boss electrons will move from the left corner to the right corner. After touch to the right corner, rays will quickly cover back left corner when the voltage about 0V (hsync). Until boss electrons to the bottom station monitors the voltage will be put into the vertical deflection coil, hooded rays will be taken back to the top screen (vsync) and continue the process as shown in Figure

Signal used to scan the screen hsync row and vsync signal used to scan the entire screen with pixel rate 25MHz frequency (25 million pixels implementation of 1s) to create screen VGA 640x480 resolution.

VGA pins



VGA signal including 5 activities: two hsync and vsync signal, three credits

video signal is red, blue, green is connected to 15 feet

Red	Green	Blue	quả color
0	0	0	Black
0	0	first	Blue
0	first	0	green
0	first	first	Cyan
first	0	0	Red
first	0	first	Magenta
first	first	0	Yellow
first	first	first	White

Way data transfer

To be able to receive data transmitted and displayed on the screen, we have designed

vga_sync circuit includes a timer and the clock signal with 2 credit bo.Mach hsync and vsync signal is connected directly to the screen, we used to

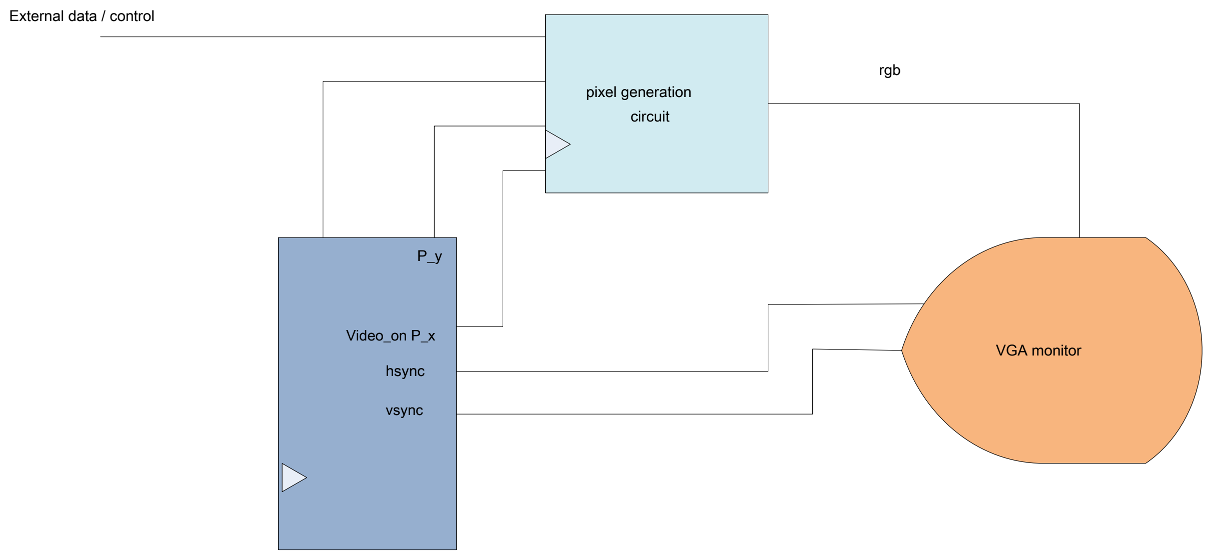
Dog the screen horizontal and vertical scanning signals hinh.Hai decoded by

a counter available in the circuit and output signal 2 is pixel_x,

pixel_y.Hai output signal indicating the relationship between the scan position and the current position

at the point of a signal anh.Mach video_on to dog off or on

the display.



VGA controller block

A circuit to generate the video signal 3 is referred to as RGB (red green

blue) input is pixel_x and pixel_y, video_on. The value of a color is

displayed on the screen depends on the current pixel position (pixel_x and

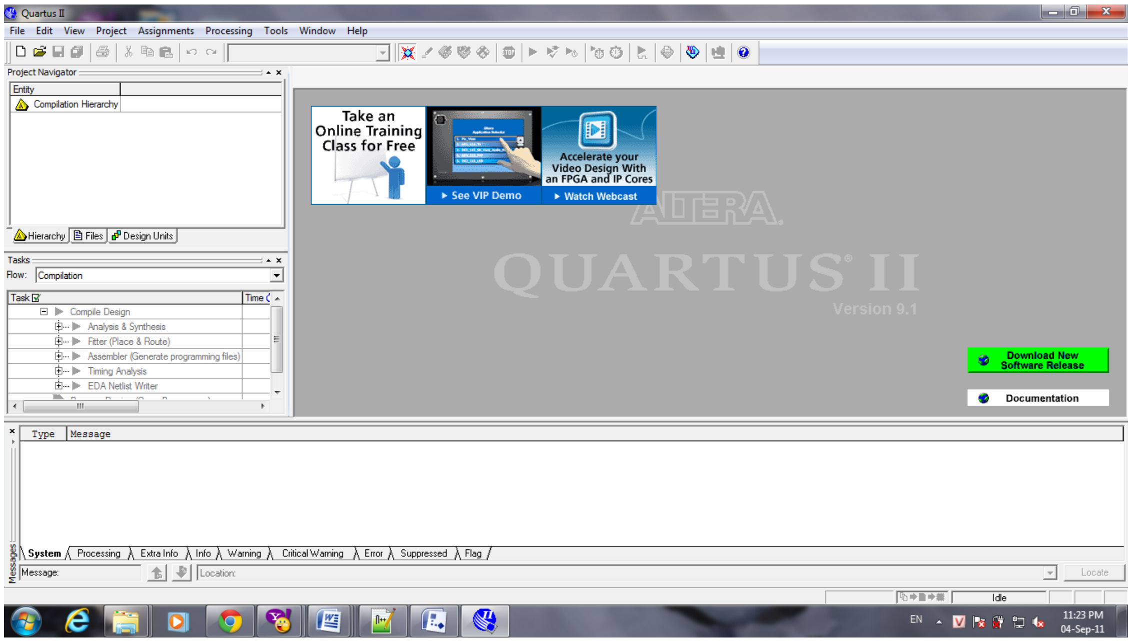
pixel_y) and data signals and dog outside

1.5. Introducing software

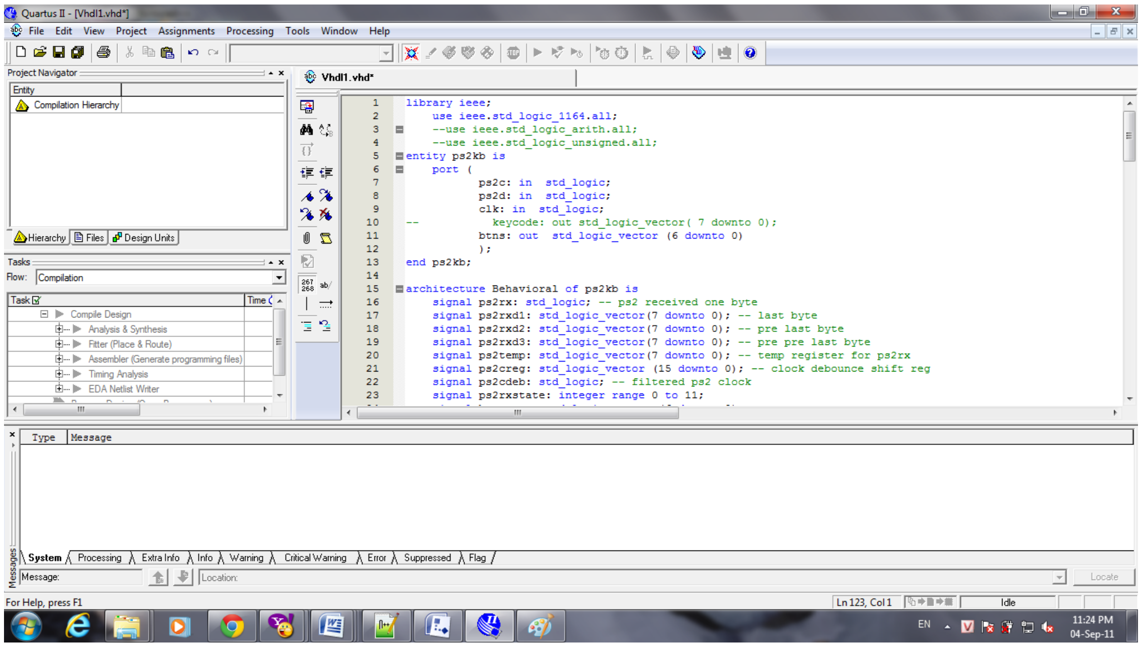
1.5.1. Quartus II

Main interface

Quartus Web Edition 9.1 for programming and loaded onto KIT DE1


Quartus II
File Edit View Project Assignments Processing Tools Window Help
Project Navigator
Entity
Compilation Hierarchy
Hierarchy Files Design Units
Tasks
Flow: Compilation
Task Time
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programming files)
Timing Analysis
EDA Netlist Writer
Take an Online Training Class for Free
See VIP Demo
Accelerate your Video Design With an FPGA and IP Cores
Watch Webcast
ALTERA
QUARTUS II
Version 9.1
Download New Software Release
Documentation
Type Message
System Processing Extra Info Info Warning Critical Warning Error Suppressed Flag
Message: Location: Locate
Idle
11:23 PM 04-Sep-11

2.2.1.2 Editor



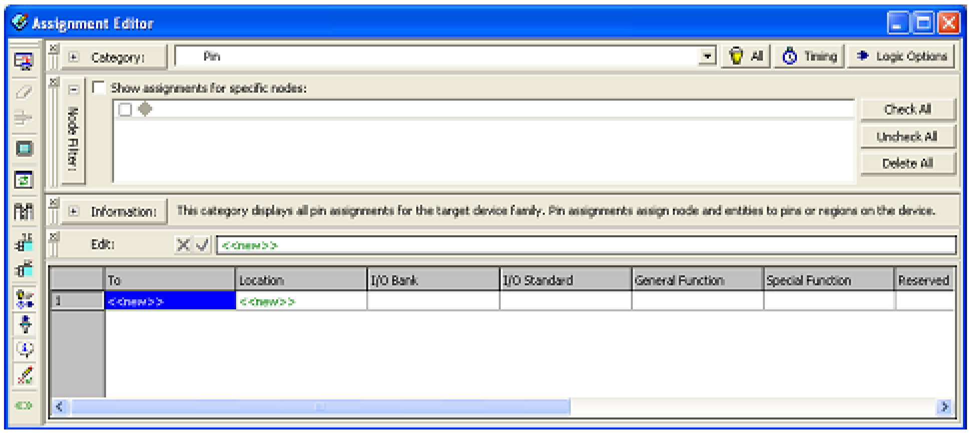
```
library ieee;
    use ieee.std_logic_1164.all;
    --use ieee.std_logic_arith.all;
    --use ieee.std_logic_unsigned.all;
entity ps2kb is
    port (
           ps2c: in  std_logic;
           ps2d: in  std_logic;
           clk: in  std_logic;
--           keycode: out std_logic_vector( 7 downto 0);
           btns: out  std_logic_vector (6 downto 0)
           );
end ps2kb;

architecture Behavioral of ps2kb is
    signal ps2rx: std_logic; -- ps2 received one byte
    signal ps2rxd1: std_logic_vector(7 downto 0); -- last byte
    signal ps2rxd2: std_logic_vector(7 downto 0); -- pre last byte
    signal ps2rxd3: std_logic_vector(7 downto 0); -- pre pre last byte
    signal ps2temp: std_logic_vector(7 downto 0); -- temp register for ps2rx
    signal ps2creg: std_logic_vector (15 downto 0); -- clock debounce shift reg
    signal ps2cdeb: std_logic; -- filtered ps2 clock
    signal ps2rxstate: integer range 0 to 11;
```

assign leg

Step 1: Click Assignments> Assignment Editor. in Category select Pin. Double-click the new << >>. Click signals from lists spread assigned to do leg. Next, double-click the box to the right box for

for the signal to be assigned (column Location). Choose from the list PIN_XYZ leg

or you can type directly into the box Location.



This category displays all pin assignments for the target device family. Pin assignments assign node and entities to pins or regions on the device.

<<new>>

	To	Location	I/O Bank	I/O Standard	General Function	Special Function	Reserved
1	<<new>>	<<new>>					

Figure 6. Assignment Editor window used to assign pins

Step 2: Same as above, perform assigned to the other input pins

Step 3: After completing the assigned pins, click File> Save. Close the door Assignment Editor window, click Yes and recompile the circuit.

Attention : Choose a name that coincides with the signals in the table DE1_pin_assigment.csv when we just need to assign pins assigment> import assigment, in our path to that file and press OK DE1_pin_assigment.csv and follow step 3 is xong.Khong take time to assign pins manually.

Translate

Once finished writing code for a program that you need to compile to create the files used to load up KIT DE1

Step 1: Click select Processing> Start Complication. Compiled into public (or unsuccessful) will be notified on the following dialog box unbent when compilation ends. Confirm by clicking the OK button.

Step 2: When the translation is complete, a compiler report is given. Door This window can also be opened at any time by clicking choose Processing> Complication Report. In this report includes a number of categories in the left window, click on the categories to see the details of the list

This item shows up in the right window.

Step 3: Fix the bugs

Select Analysis & Synthesis> Messages to display error messages. Click

Double-click the first error message, the command line errors will be marked on the

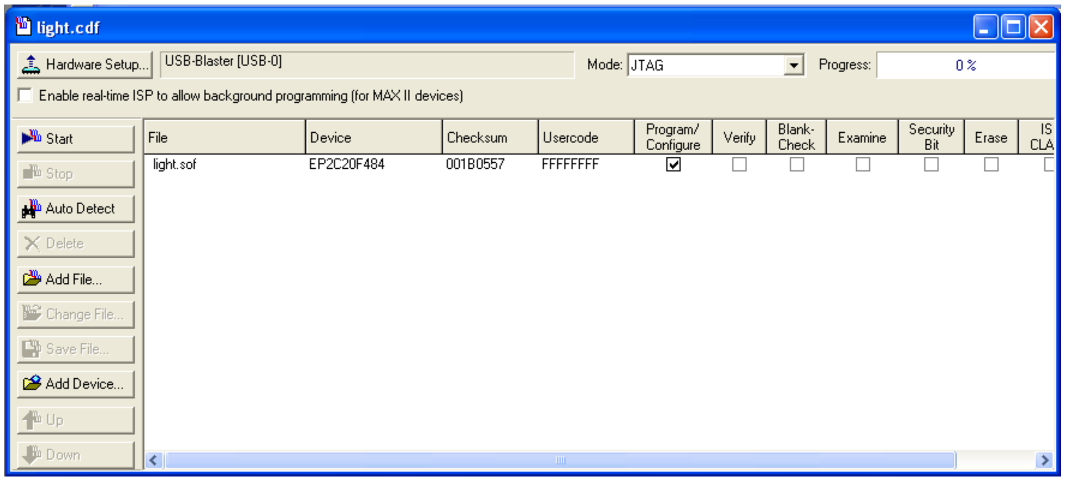
text editor, correct it and then recompile the project.

Load up KIT

Step 1: Brushing switching RUN / PROG switch to RUN. Click Tools>

Programmer to the window as shown in Figure 11. Check the options

Program / Configue to allow xxxxxx.sof loaded configuration file.



USB-Blaster [USB-0] Mode: JTAG Progress: 0 %

☐ Enable real-time ISP to allow background programming (for MAX II devices)

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine	Security Bit	Erase	IS CLA
light.sof	EP2C20F484	001B0557	FFFFFFFF	☑	☐	☐	☐	☐	☐	☐

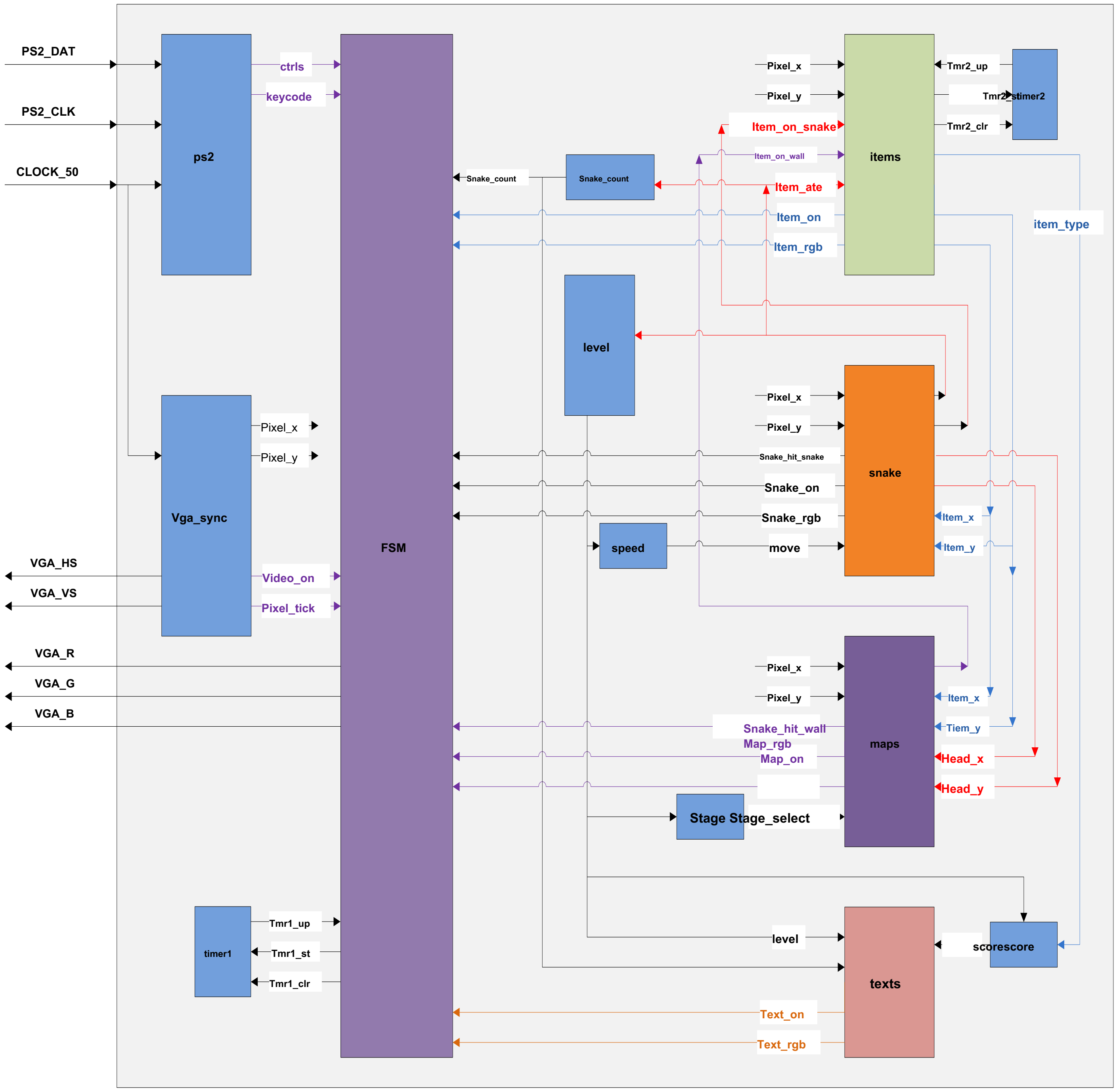
Step 2: Click the Start button on the left window to load a configuration file down

FPGA. After successfully loaded into the FPGA, check this circuit

implementation on FPGA to run exactly as official breeds of desire or not.

1.6. System block diagram

1.6.1. Extensive system



1.6.2. Blocks and functions.

Grade level	Function
PS2	Receiving signals from the keyboard, sending control signals.
VGA_SYNC	Control the display on VGA
ITEMS	Creating prey for snakes.
SNAKE	Control block of solid behavior.
MAPS	Creating various obstacles in the game screen.
texts	Displays information to the user.
FSM	Volume control system, linked to other blocks.
TIMER	Timer, create latency.
SPEED	Regulators moving speed of the snake.
LEVEL	Calculation rank players.

2. Detailed design

2.1. VGA block

About CRT 640x480

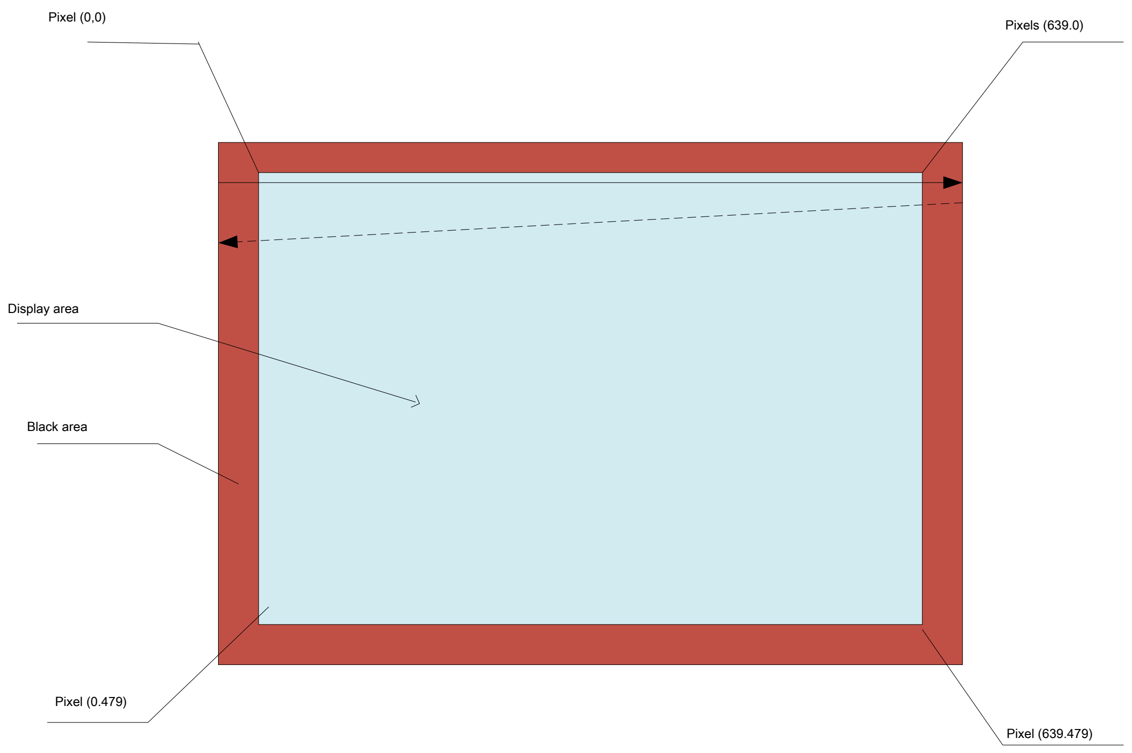
640x480 is the number of pixels and lines that can be displayed on screen hinh.Moi

including 640 horizontal lines of pixels, 480 is the number of horizontal scan lines but the fact

are each composed of 800 horizontal lines of pixels and the number of horizontal scan lines is 525.So

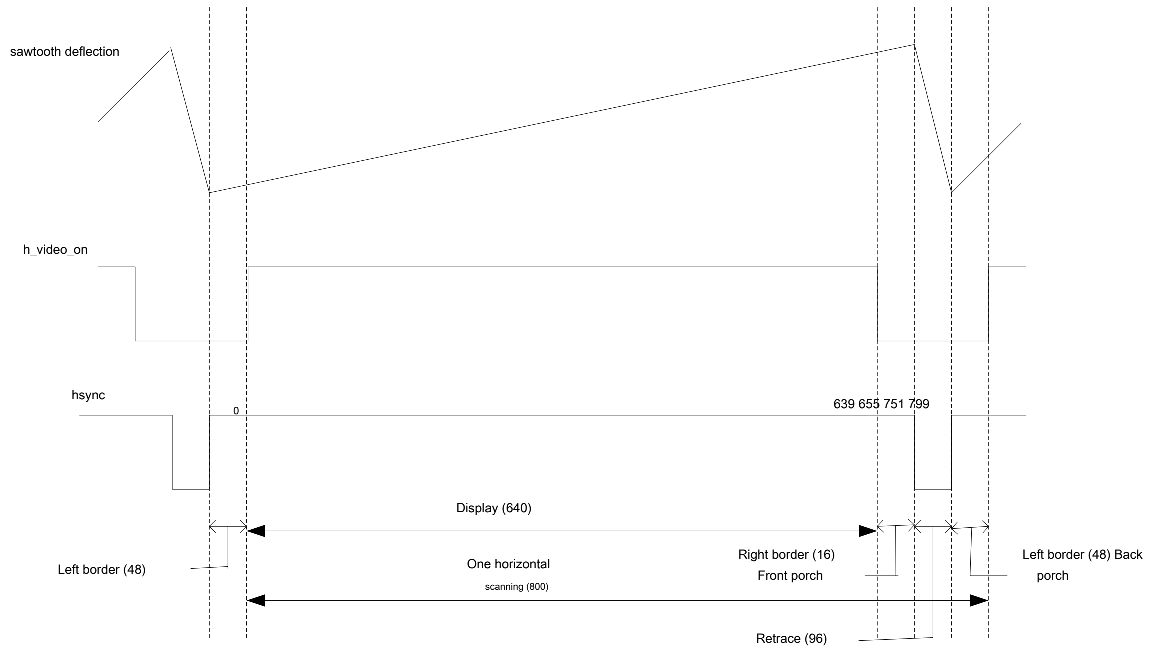
pixels and lines not shown is called *black border (* black border) .Tan

the operation is 25MHz.



1.1.1. Horizontal axis synchronization

800 pixels horizontal axis is divided into 4 regions



- Dislay: regions where the pixels are shown with the length 640px.

- Retrace: cover regions where electron back left corner, the video signal should be off, length 96px.
- Right border (right border) regions forming right border of the display area is called the front porch (front gate), the signal should be switched off, length 16px
- Left border (the left): left border region form of the display area is called back porch (tailgate), the signal should be switched off, length 48px.

The following code determines the quantities for the horizontal axis:

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

The length of the left and right border may vary between different screens.

Hsync signal needs more counter circuit 800 pixels and a resolution ma.Khi arrested first displaying the timer starts counting and also signal the signal component pixel_x.Tin hsync low signal when the output signal of the counter in between 656 and 751.

We used to adjust signal video_on show / not show when the count value less than 640

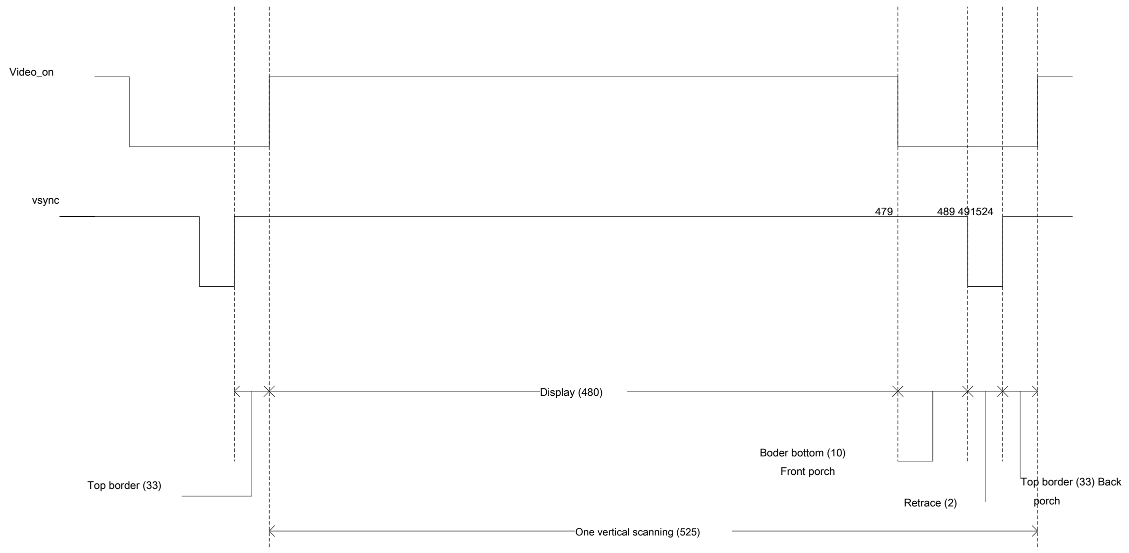
CRT monitors should be placed in the left and right black and transparent process retrace.

1.1.2 Vertical axis synchronization

During Vertical, electron beam moving steadily from beginning to end the screen, then came back and head curtain require similar hinh.Su an interval to refresh the vsync hinh.Cau similar architecture with hsync.

One cycle of the signal vsync is 525 lines and is divided into 4 regional seed as hsync.

The characteristics of each region are similar hsync



Vsync signal line 525 to one count and a count circuit when the first ma.Bat start area thi.tin current output signal of the counter is pixel_y.Tin vsync signal in low when the counter flow in lines 490 or 491.

Like hsync, we use video_on to show / not show when the count value is less than 480.

1.1.3 Calculation of VGA signals time synchronization

In the scope of this project we are using frequencies that are selected 25MHz.Su decided by three quantities:

- p: number of pixels on a horizontal scan line. p = 800 pixels / line
- l: total sugar monitor. l = 525 lines / screen
- s: number of frames per second. s = 60 screens / second

Choose s = 60 here is because the human eye works well in this frame and against

the flashing.

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

1.1.4 Complete VGA Graphic

Above we have designed 2 sets dem.Van design problem here is that only support KIT DE1

50MHz frequency support that request is 25MHz.Vi by design requirements so we created

1 of 25MHz allowing the marker to pause or enable the dem.Tin

p_tick signal is a signal to perform this work and coordinate with the activities

circuit of the pixel generation.

Use 2 effective h_end progress and v_end to check the completion of the horizontal scanning and

vertical.

Also to avoid interference situation we need to use more of the

Buffer was inserted hsync and vsync signals.

The following code used to create the mod-2 counter used to mark time:

```
-- 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';
```

The following code used to identify the completion of the Horizontal:

```
h_end <= - end of horizontal counter
    '1' when h_count_reg = (HD + HF + HB + HR-1) --799 else '0'; // subtract 1 because
we count from 0 //
```

The following code is used to eliminate noise:

```
-- 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. block PS2

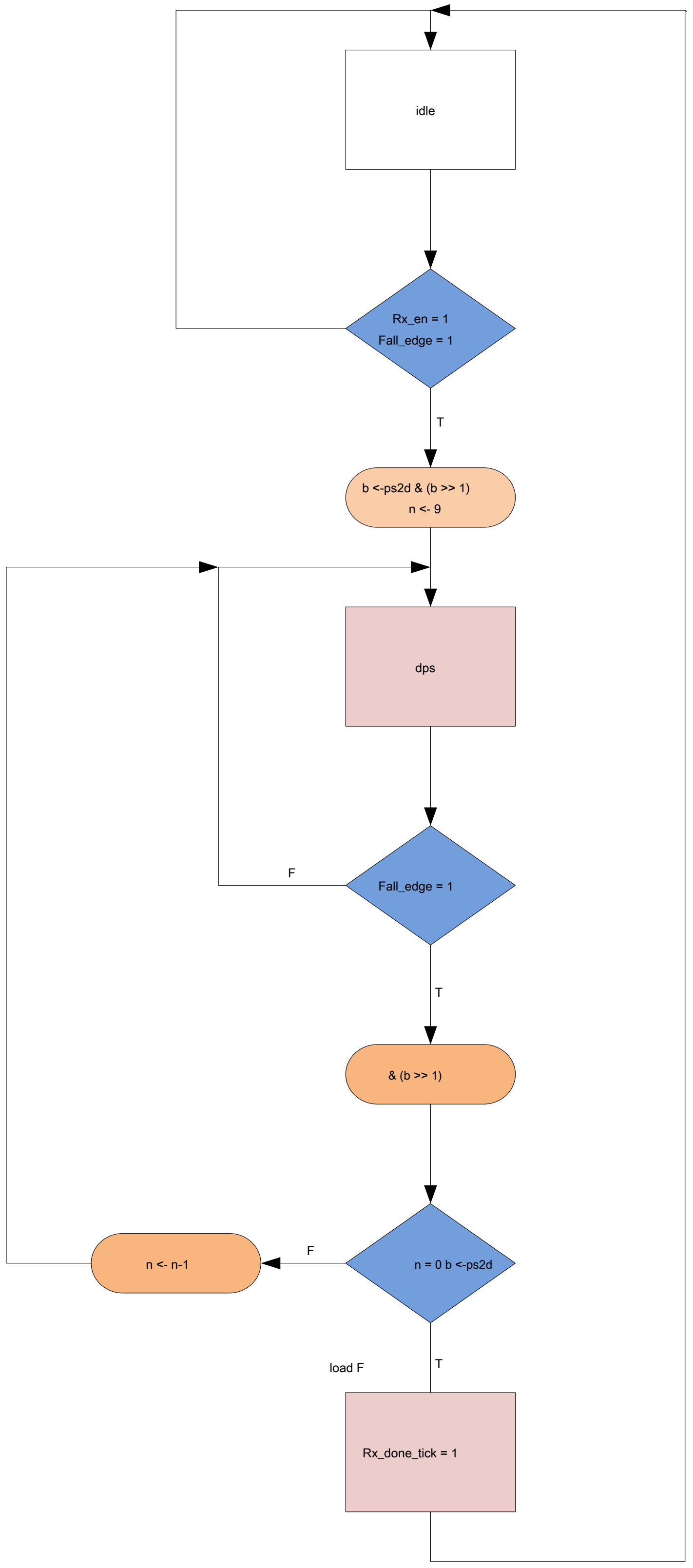
On the circuit includes a receiver which can insert a block against signal interference, a block FSM to dog transmission scan code, FIFO blocks and transmit data to burn out under the first in first out (first in first out).

2.1 Block receiving data (PS2_rx)

Use the following code to prevent interference to the signal:

```
ps2cdeb <= '0' when ps2creg = "1111111111111111" else // returns 0 if all
                                                   bits is 1
'1' when ps2creg = "0000000000000000" else // returns 1 when all bits are 0;
```

To understand the mechanism of PS2 receive data we consider the following flowchart:



ASMD chart of PS2 port receiver

When the signal allows received positive signals rx_en and chops down the state

Status will translate to start bit and move to state DPS.

Since the data is received in a block has been checked, we will translate

10bit again in a separate status rather than using a separate state

as data, parity, stop.

Then will go to the state circuit load in which a clock signal cycle

distributed more to complete the process of translating the stop bit, and signal

rx_done_tick be inserted after the first cycle to notice you have received the data

Data.

2.3. block ITEMS

2.3.1. General description

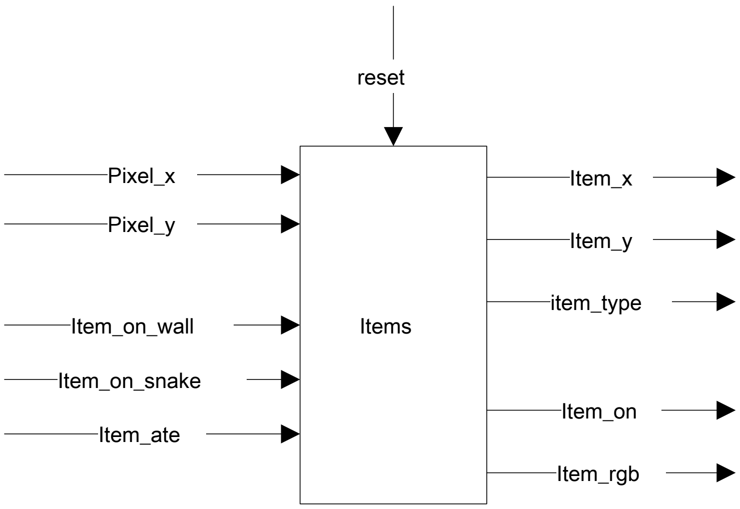


Figure 2. Block ITEMS

ITEMS tasked blocks born prey to snakes, the new location meets the

conditions:

- New bait is created if the bait is eaten, or as soon as they were created

  The new bug a certain solid fuels, or centered around obstructions.

- Bait is generated at random positions, does not depend on location primer before.
- New Predator born in the region must move the snakes.
- New primer is the position does not coincide with any solid fuel, if it solid grasp on, to recreate a new bait.
- New born bait obstacles must not be identical, if not, to create the new prey.
- The special baits will disappear after a period of time regulations.

Type	Shape Time	Features The women	disappeared	
first	Apple		first	
2	Star	5s	first	Deceleration
3	Heart	5s	first	Increase plays again
--	--	--	--	--

The inputs and outputs are declared as follows:

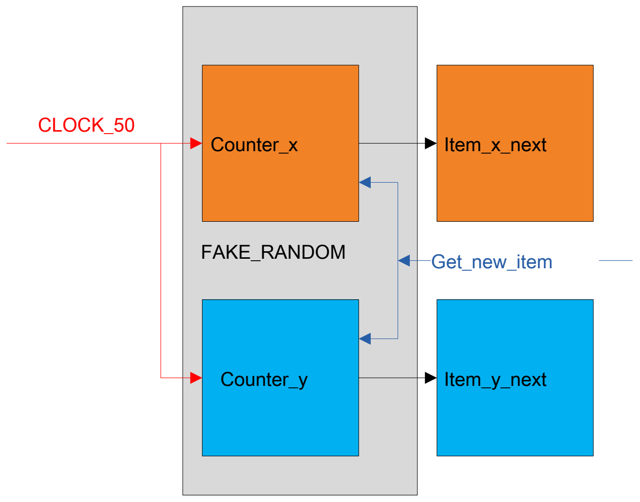
```
     Items entity is
     ports (
     -- Main clock
     clk: in std_logic;
     reset: in std_logic;
     -- coordinates of the pixel being scanned
     pixel_x: in std_logic_vector (9 downto 0);
pixel_y: in std_logic_vector (9 downto 0);
     -- current level to appear different primer suitable level
```

```
        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;
```

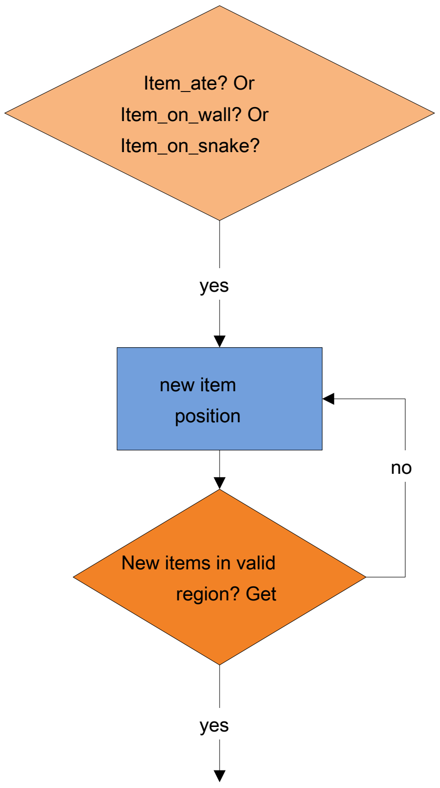
Create rom for Items:

We describe ROM through the array of bits, as follows:

Random number generator:



We use two counters as Random Number Generator. with 50Mhz clock, and unknown time items ate khi,
value of counter Seem to be random.



Item checked untill position Phải WE HAVE a valid value

To generate random numbers we use the counter rotation and counter values

be retrieved at any time we want, to ensure randomness, it

counters turnaround time is much smaller than the average time

get value from the counter

I use the counter with a small value, but repeated with frequency quickly, then

is 50Mhz clock.

Counter 1 is used for the range of values according to the coordinates x

cross-priming

Counters 2 used to create y coordinate values vertically

of prey

Combine 2 values are going to get the coordinates of the new bait was born,

and seems almost random.

code:

2.4. SNAKE block

2.4.1. diagram

2.4.2. code

2.5. MAPS block

2.6. block 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;
      ieee.std_logic_1164.all use;
      ieee.numeric_std.all use;

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

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


architecture arch of font_rom is
    ADDR_WIDTH constant: integer: = 11;
    DATA_WIDTH constant: integer: = 8;
    addr_reg signal: std_logic_vector (1 downto ADDR_WIDTH-0);
    rom_type 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
    -- x01 code
    "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. FSM block

2.7.1. diagram

2.7.2. code

2.8. sub block

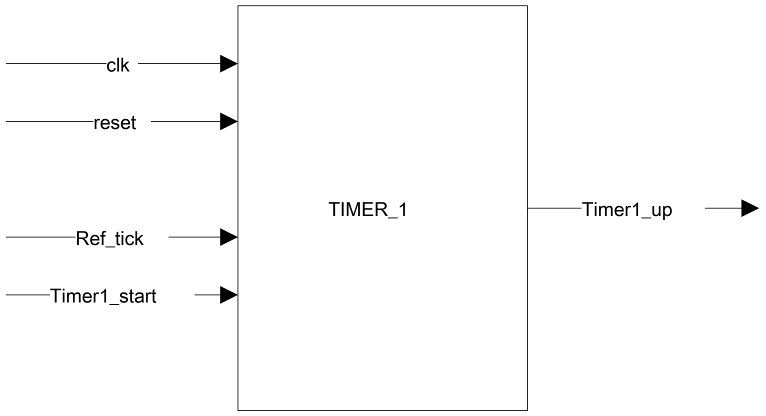
2.8.1. TIMER_1

This is the timer 2s blocks to create delay between the machine's status

FSM states, useful when you want notice something on the screen. IN

Here is the announcement GAME OVER, CLEAR STAGE, LEVEL UP ...

Model:



Declare the input, output:

```
entity timer is
        ports (
                -- Main-clock at 50MHz input
                CLK:                        in std_logic;
                -- Synchronous reset
                reset:          in std_logic;
                -- 60Hz-clock rate of the monitor is input
                refr_tick:      in std_logic;
                -- start signal
                timer_start: in std_logic;
                -- output
                timer_up:       out std_logic
        );
end timer;
```

We will use the counter to make timed quest, so we saved

the count value in a register, can count rising, but how arbitrary count reduction

count reduction is more beneficial when comparing the value of the register with the value 0.

Here we need about 2 seconds timer, input a signal

refresh the screen at about 60Hz to 2 seconds would correspond to values

of registers 120.

We selected 7-bit registers, meaning its max value is 127, ie the period

127/60 = 2.1 lag is seconds, uncertainty still acceptable.

code:

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

The next code section counting processing time, first processor registers,

using 50Mhz clock as main clock to increase synchronization system

system

We deploy counter state machine format but without any

created state ram register only thing worth divided into two bars

record: timer_reg only current counter value, and the counter value only timer_next

then, if there happens clock.

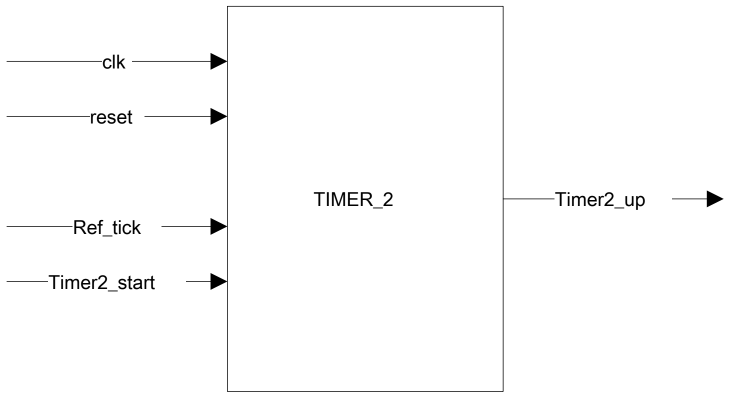
code:

```
-- registers
process (clk, reset)
begin
    if (clk'event and clk = '1') then
        if reset = '1' key
            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

Similarly TIMER_1, TIMER_2 also the counter, but it is used in other purposes, it is time display of a special kind of bait in the game, such LIVE_UP, SPEED_DOWN ... this new species displayed in a certain time period and then disappear if during that time could not eat solid food.



Like TIMER_1, we count the last time with the beat clock register 60Hz through the value of the register. The only difference is the length of the bar ie max value recorded that storage registers.

Here registers will be:

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

10 bit value stored max = 511, so the biggest delay is 511/60 = 9 seconds.

The handling TIMER_2 identical TIMER_1 should not say here again.

2.8.3. LEVEL

Block Level monitor the number of solid bait has been eaten, so that when solid food certain amounts of bait, the rank of the player will be increased, accompanied whereby the movement speed increased targeting solid increase the level of difficulty for screen play.

Once players start at level 1 and reach level 5, screen play will change, we need to control when the game screen changes, then This should add some code for an examination of the level achieved OK.

The necessary signal pins are described as follows:

```
Entity level is
  ports (
     -- Main clock
      clk: in std_logic;
       restart: in std_logic_vector (1 downto 0);
     -- bait has been eaten
       item_ate: in std_logic;
     -- notification signal level up
       level_up: out std_logic;
```

```
    -- signal over screens.
      stage_clear: out std_logic;
    -- Current game screen of the player.
      stage_select_in: in integer range 1 to 7;
      stage_select_out: out integer range 1 to 7;
    -- inform current level of the player.
      level: out integer range 1 to 7
  );
end level;
```

Block-level code to control the player based on the bait was eaten.

Some bait to eat the next level is 5 times higher than current levels, for example from level

level 2, level 3 should want to eat 2 * 5 = 10 primers.

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

This code will control the game screen if the player has reached level 6, the

Players will be raised screen play harder

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

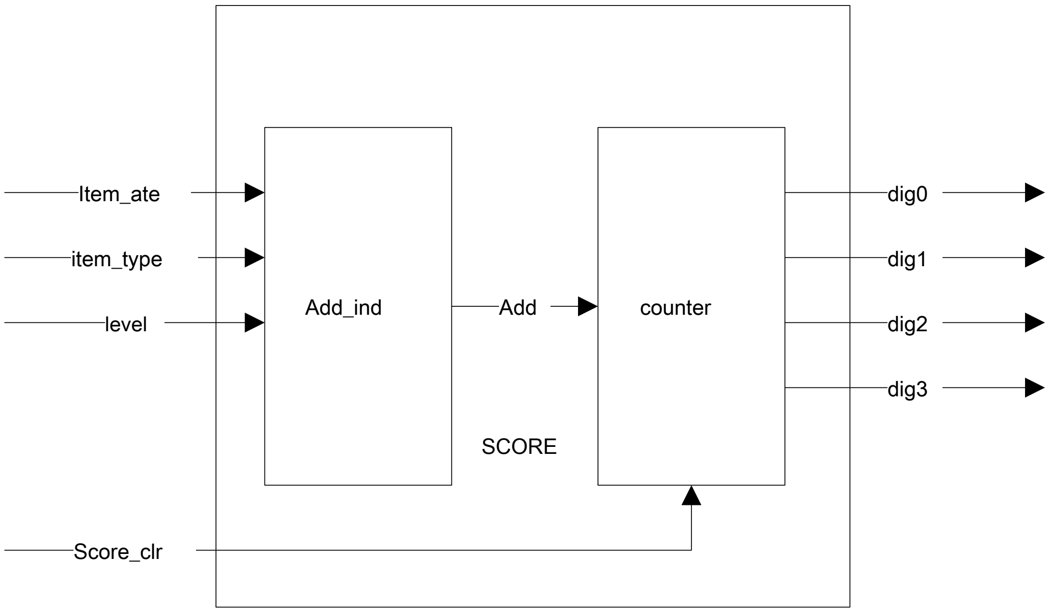
and ranks in the new game screen will be started again from level 1.

```
                                    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. diagram



- This unit is responsible for grading the player, and the score received will depend on the type of bait that snakes eat, current level of the solid (speed the faster, the higher the level, the more points received understand).
- Blocks receive signals from investors including Item_ate ITEMS, and item_type, after when calculating block Add_ind play plus point that nguii further, it will create the Add to block Counter pulse points. For example when Item_ate = 1, item_type = 3, Level = 2, then the score is 3 + 2 people = 5, this time will generate 5 cubic Add_Ind signal Add to the Counter.
- Score_lcr aimed reset when the score of the 0000 game screen newly captured head

2.8.4.2. Counter_9999

As mentioned above, plus block signals received via COUNTER_9999 d_inc foot, each time this signal from a low to a high value of the counter will increase by 1.

The legs of the block are described as follows:

```
counter9999 entity is
  ports (
        -- Main clock
        clk: in std_logic;
        -- reset
        reset: in std_logic;
        -- Addition indicate
        d_inc: in std_logic;
        -- output is the number of distinct advantages for the inclusion of
Texts to turn into digit number, serves the display on the screen.
```

```
            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)
    );
counter9999 end;
-----------------------------------------------------------
architecture arch of counter9999 is
    -- registers save the current value and the next 4 count.
    dig0_reg signal, dig1_reg, dig2_reg, dig3_reg: unsigned (3 downto 0);
    dig0_next signal, 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' key
                            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' key
            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 bars
                dig0_next <= (others => '0');
                if dig1_reg = 9 bars
                    dig1_next <= (others => '0');
                    if dig2_reg = 9 bars
                        dig2_next <= (others => '0');
```

```
                                if dig3_reg = 9 bars
                                    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

This mass is the mass nature of the control block and counter999 communicate with the higher-level block.

Declare the signal pins as follows:

```
library ieee;
    ieee.std_logic_1164.all use;
entity score is
    ports (
            -- Main clock
            clk: in std_logic;
            -- remove about 0000 signal point
            score_clr: in std_logic;
            -- Newspaper bait was eaten
            item_ate: in std_logic;
            -- baits
            item_type: in integer range 1 to 7;
            -- Current levels of players
            level: in integer range 1 to 7;
            -- put out for the texts digit
            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;
```

we declare a component for the block counter9999

and perform signal connector block.

```
    counter_u0: Entity work.counter9999
    port map (...);
```

Important part here is how to create urban rhythm corresponding point count

will be added to the player, this count rate takes into

counter9999 through turn signal counter and d_inc.

We handled through a process as follows:

```
process (clk, level, item_ate, item_type)
        -- variable flow conditions
        add variable: std_logic;
        -- stored value to community
        variable counter: integer range 0 to 63;
  begin
                if clk'event and clk = '1' key
                -- if the bait is eaten
                        if item_ate = '1' key
                -- signal output of the counter
                                add: = '1';
                        end if;
                        -- if the signal for the counter
                        if add = '1' key
                                -- calculate the amount plus
                                if counter = level + item_type Athens
                                        -- if enough writers community newspaper
community
                                        add: = '0';
                                        -- put counter to 0
                                        counter: = 0;
                                else
```

```
                                    -- if not, the next count
                                    counter: = counter + 1;
                                end if;
                            end if;
                            -- and provide signals for counter9999
                            d_inc <= add;
                        end if;
                end process;
            end arch;
```

2.8.5. Speeds

2.8.6. SNAKE_COUNTER