International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 05 Issue: 05 | May-2018 www.irjet.net p-ISSN: 2395-0072

FPGA Implementation of Snake Game Using Verilog HDL

Nishant Singla[1], Mandeep Singh Narula[2]

[1]Student, Dept. of E.C.E., JIIT, Noida, India,
[2]Professor, Dept. of E.C.E., JIIT, Noida, India.

Abstract—*A video game is simulation of a scenario, traditionally implemented on a digital systems such as a computer or a dedicated hardware gaming console, generally using traditional sequential programming languages, mainly C/C++.*

In this project we implement the electronic Snake game on FPGA (Field Programmable Gate Array) using Verilog HDL[9]. Implementation in HDL (Hardware Description Language) is quite different from implementation in sequential languages mainly because of the parallel nature of the HDLs, where sequential language code is executed by a digital core step by step, HDL code describes the functioning of a digital hardware, it is taken by synthesis tools that try to find a digital hardware implementation of the description, thus there is no step by step execution of the statement of HDL, each statement is a smaller circuit in itself. The system is implemented on FPGA, which are modern programmable logic devices, i.e. we can program almost any digital function in it. Newer FPGA are capable of holding complete systems on them. They are called field programmable as they can be reprogrammed after manufacturing when they are in the field.

Key Words—Verilog; FPGA,DE0 Nano; Snake Game

1. INTRODUCTION

The system is a virtual implementation of SNAKE game. The game is played by 1 player with a handheld controller. The controller interface directly with FPGA. The FPGA drives the VGA monitor.

Fig - 1 : Block Diagram of the System

The complete game is implemented in digital logic using the Verilog HDL on the Terasic DE0-Nano FPGA development board based around Altera Cyclone IV FPGA.

The controller consist of 4 pull – downed push buttons that are used to move the 'Snake' in the game. The state of the buttons are read by the FPGA which then updates the direction of the snake.

FPGA drives the VGA monitor by generating 5 signals, 'RED', 'GREEN', and 'BLUE' which are used to turn the colours of individual pixels on or off, and 'H-Sync' and 'V-Sync', which are used to synchronize the Horizontal and Vertical scans of the monitor[1].

2. SNAKE GAME

2.1 Description of the game

"Snake" is a simple game where the user controls a snake to eat items generated at random locations in the play area. The snake gets longer and harder to control the more items it consumes. You lose if the head of the snake collides with its own body, or if the snake hits one of the borders.

This game was first released in 1976, and variations on the game are still around today. For example, in the YouTube website, you can play snake over a video while it loads, and Google has made a doodle in which you can play a slightly updated version of snake.

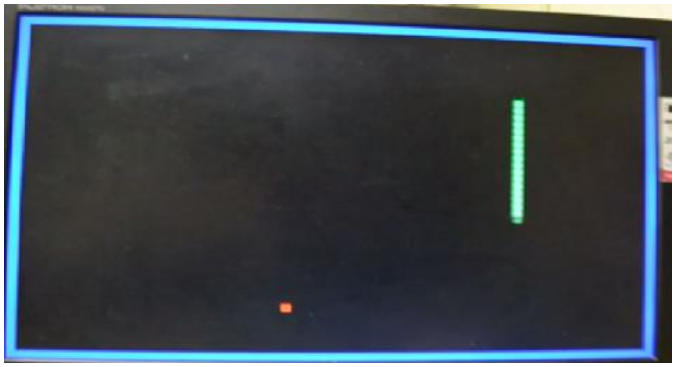


Fig - 2: Final look of the Game

Rules of the game

- The game consists of a Snake, Border and Apples.
- Snake move on screen with a constant velocity.
- The player has to use the Controller to change the direction of Snake so as to make it hit the apples.
- If player fail to prevent the head of 'Snake' from hitting the Border or its own body he losses.
- When the Snake hit any Apple its length increases.

3. Implementation

The game is implement on FPGA using Verilog, which is Hardware Description Language. The code written in Verilog describes the behaviour of the desired hardware. The code is taken by Altera's synthesis tool (we use Altera's toolchain because we have an Altera Cyclone IV FPGA) which 'try' to find an implementation of the description of the code. The word try is empathised as the tools, advance as they may be, might not be able to find a correct implementation for given description or might produce poor or overly complex implementation, thus it is our job as designer to write synthesizable code.The main idea is to have a general view of the kind of circuit implementation will be derived by the tools and to logically partition the module in the code, it is easy to correctly implement small module interconnected then a very big and complex block. In view of above ideas, we have tried to best partition our implementation, using many modules[6].

Hardware

The system is implemented on a Terasic DE0-Nano[8] FPGA development board. As the board does not have many peripherals we need to make our own expansion board to connect the FPGA to controller, VGA monitor . All the hardware built along with the development board is explained in this section.

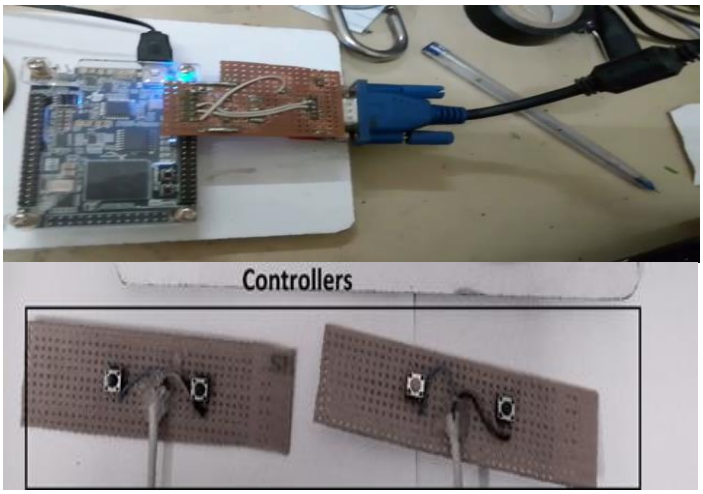


Fig - 3: Final Hardware of the Project

Terasic DE0-Nano FPGA Development Board

This Project uses a Terasic DE0-Nano FPGA Development Board, it introduces a compact-sized FPGA development platform suited for to a wide range of portable design projects.

The DE0-Nano features a powerful Altera Cyclone IV FPGA (with 22,320 logic elements), 32 MB of SDRAM, 2 Kb EEPROM, and a 64 Mb serial configuration memory device. For connecting to real-world sensors the DE0-Nano includes a National Semiconductor 8-channel 12-bit A/D converter, and it also features an Analog Devices 13-bit, 3-axis accelerometer device.

The DE0-Nano board includes a built-in USB Blaster for FPGA programming, and the board can be powered either from this USB port or by an external power source. The board includes expansion headers that can be used to attach various Terasic daughter cards or other devices, such as motors and actuators. Inputs and outputs include 2 pushbuttons, 8 user LEDs and a set of 4 dip-switches.

The key feature of the board are listed below –

- Featured device
  - Altera Cyclone IV EP4CE22F17C6N FPGA
  - 153 maximum FPGA I/O pins
- Configuration status and set-up elements
  - On-board USB-Blaster circuit for programming
  - Spansion EPCS64
- Expansion header
  - Two 40-pin Headers (GPIOs) provide 72 I/O pins, 5V power pins, two 3.3V power pins and four ground pins
- Memory devices
  - 32MB SDRAM
  - 2Kb I2C EEPROM
- General user input/output
  - 8 green LEDs
  - 2 debounced pushbuttons
  - 4-position DIP switch
- G-Sensor
  - ADI ADXL345, 3-axis accelerometer with high resolution (13-bit)
- Clock system
  - On-board 50MHz clock oscillator
- Power Supply
  - USB Type mini-AB port (5V)
  - DC 5V pin for each GPIO header (2 DC 5V pins)
  - 2-pin external power header (3.6-5.7V)

VGA Connector

The FPGA displays the game on VGA monitor, the DE0-Nano board lacked a VGA port we made our own VGA expansion board to connect to the GPIO-00 expansion header of the DE0-Nano board. A VGA Connector is a DSub-15 connector[2], the connector use 3 analog signals for red, green and blue and 2 digital signal for h-sync and v-sync, it also has 2 I2C buses which we don't use.

Table 1 : Pin Usage of VGA Connector

Pin	Function	Type	Maximum
1	Red video	Analog	0.7
2	Green video	Analog	0.7
3	Blue video	Analog	0.7
5	Ground (h-	-	-
10	Ground (v-	-	-
13	HSync	Digital	5
14	VSync	Digital	5

As the 3 video signal are analog and our FPGA cannot output an analog voltage, we use it as a digital signal, this make the circuit easy as we just need to shift the voltage level from 3.3V of FPGA to 0.7V of VGA Standard, but allows us to only have the colour either completely ON or OFF, as there are 3 colours we get a total of 8(2^3 = 8) colour for each pixel, which is more than enough for our application[3].

The colour are summarized as following –

Table 2 : 3 Bit Colour Combination

Red	Green	Blue	Resulting
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

The problem of shifting level is easy with voltage divider, as there is already 75Ω pull-down resistance in the monitor according to VGA Standard, we just need a 270Ω resistance in series of connector[5].

$$\frac{75\Omega}{75\Omega + 270\Omega} \times 3.3V = 0.717V$$

Table 3 : FPGA to VGA Connector Connections

Connector Pin (Pin	DE0-Nano Pin	FPGA
V-Sync (14)	GPIO_05 (8)	B4
H-Sync (13)	GPIO_07 (10)	B5
Red (1)	GPIO_00 (2)	D3
Green (2)	GPIO_01 (4)	C3
Blue (3)	GPIO_03 (6)	A3
GND (5,6,7,8,10)	GND (12)	-

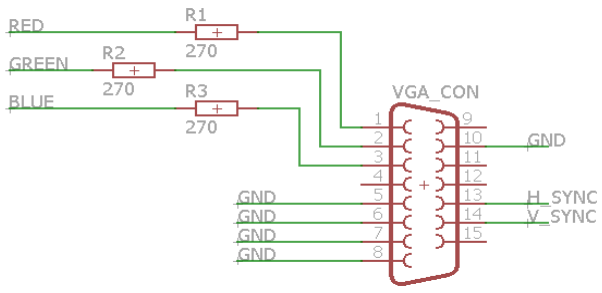


Fig- 4 : Circuit Diagram of VGA Connector Circuit

Controller

The game calls for a controller for the player to play the game, the controller needs 4 button for moving snake up , down, right and left. Each button is implemented as a momentary pull-up switch.

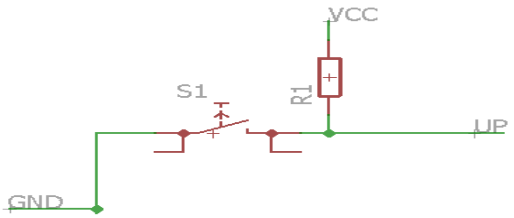


Fig - 5 : Circuit Diagram for a Button on Controller

Verilog HDL Hardware

The complete system is implemented in Verilog. As a successful implementation in Verilog calls for good logical partitioning of the circuit, various modules are created that are interconnected to make the whole system.

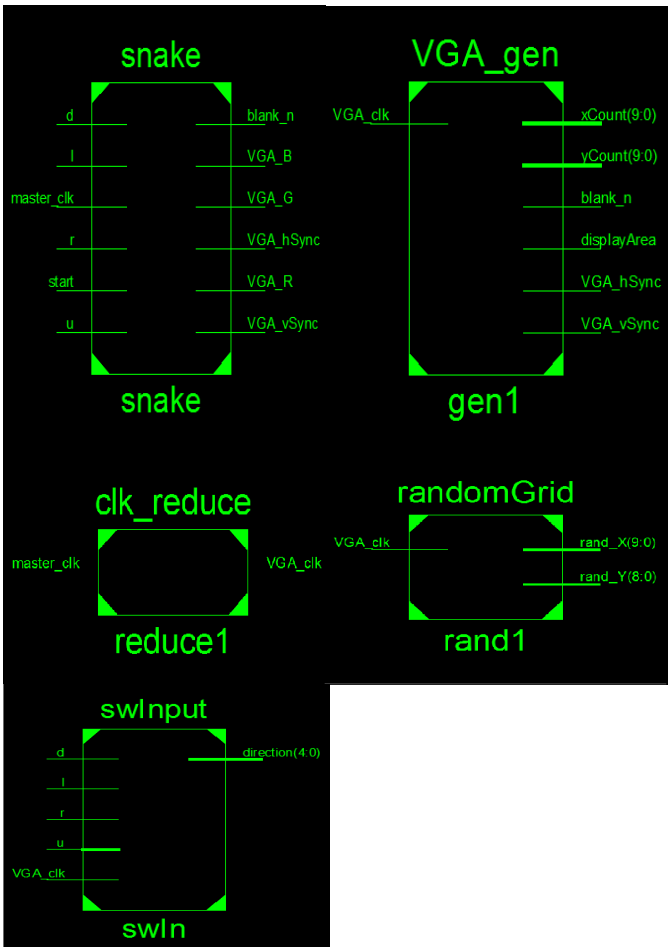
Module Name: Snake: - This is the top module which takes input from controller, clock and start button and is interconnected to all the other modules. This module implement the game logic and gives the output to the VGA monitor driving circuit.

Module Name: VGA_gen: - It generates the H-Sync and V-Sync signals needed by the VGA monitor to synchronize the horizontal and vertical scans of the monitor. The pixel_x and pixel_y counter signals that represent the current scan coordinates and are used by other modules to generate their 'on' signals.

Module Name: CLK_reduce: - This module generates a 25 MHz clock by reducing the on-board 50MHz clock as the main module operates on a reduced clock[7].

Module Name: swInput: - This module is responsible for controlling the input to the main unit. As we need to attach a 4 output controller to the game, this module handles the input to move the 'Snake' in specified direction.

Module Name: RandomGrid: - This module generates a randon position of pixel x and pixel y for the apple's new position.



3 Architecture

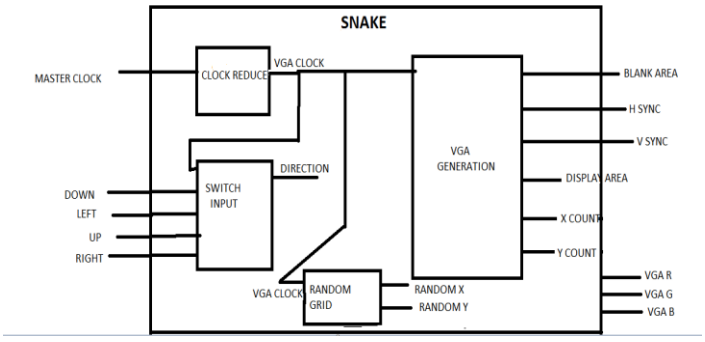


Fig - 6 : Interconnections between modules

4 REFERENCES

[1] H. Dong and H. Guo, "Design of VGA display controller based on FPGA and VHDL," International Conference on Electric Information and Control Engineering, 2011.

[2] Olivito, C. Gonzalez, and J. Resano, "FPGA implementation of a strong Reversi player," International Conference on Field-Programmable Technology, 2010.

[3] B. Muralikrishna, V. G. S. Swaroop, K. G. Deepika, and H. Khan, "PS2-VGA Peripheral based Character Display using FPGA," International Journal of Computer Applications, vol. 48, no. 9, pp. 1–5, 2012.

[4] J. Eldon, "Video to VGA and back," Proceedings of 27th Asilomar Conference on Signals, Systems and Computers.

[5] S. Palnitkar, Verilog HDL: a guide to digital design and synthesis. Taipei: Pearson Education Taiwan, 2009.

[6] P. P. Chu, FPGA prototyping using verilog examples: xilinx spartan -3 version. Hoboken, NJ: Wiley, 2008.

[7] VGA Signal 640 x 480 @ 60 Hz Industry standard timing. [Online]. Available:http://tinyvga.com/vga-timing/640x480@60Hz.