Electrical and Computer Engineering

ECE-5225

Project Report

THE STEALTH BOT

-Ajith Muthu

Date: Dec 16, 2021.

Title:

The title of the project is *The Stealth Bot* designed by *Ajith Muthu* on 16th of December 15, 2021, as a final project for Hardware Software Codesign course in Electrical and Computer Engineering, Saint Louis University.

Abstract:

The use of vivado application to program Verilog code-based game and test it on a VGA monitor using Basys3 FPGA processor is being experimented in this project. The frame creation, object creation, providing control to the object, movement of the object, flashing the objects and interfacing a seven-segment display to the FPGA is being tested in this project. The major constraints of flashing the object, moving the object, interfacing counters and seven-segment display on the processor, and providing empty spaces for a specified pixel areas of the frame is detailed in this report.

Introduction:

In this project I created a game called The Stealth Bot Basys3 FPGA board programmed using Verilog and displayed on the Monitor using VGA cable. I have used Vivado 2019.1 Xilinx based system application to develop and execute the game. The game is played in a rectangular region having blue borders. A green colored small square shaped object is used as a bot that can be moved vertically and horizontally using up(btnU), down(btnD), left(btnL) and right(btnR) button of the Basys3 FPGA board. The center button(btnC) is used to reset the game when the game is over.

I have used 7 horizontal and & vertical obstacles that must be avoided by the bot. There are gaps provided in the obstacle which is subjected to move up and down on vertical and horizontal obstacles respectively. The controller must move the bot in vertical and horizontal direction using the buttons and cross the obstacles through the gap to remaining in the game.
I have interfaced a 7-segment display to record the duration of the gameplay using up-counter which increments its value every second and stores it as a duration of minutes followed by seconds on the display. The difficulty of the game can be changed by changing the size of the gap.
I faced a problem of determining the frame size as the simulator took longer time to run, so I generated a bitstream and figured out the errors in my code by running and testing the game for bounds and collision logic.
Other constraint I faces was, flashing the obstacle and the bot to indicate the end of game which was later resolved using a clock signal on the obstacle and the bot it fails to move through the gaps defined. The flash was reached by toggling the signal on the bot and the vertical or horizontal obstacle it contacted by passing the clock signal derived by flipflop and wired it through the register to reset the game.

Design/Work Performed:

Basys3:

Basys3 FPGA XC7A35T-1CPG236C board is used to develop and execute the Stealth Bot game displayed over a VGA Monitor. The high-capacity board featuring 450MHz clock speed,3 Pmod ports, 1.8Kb RAM, inbuilt 4 digit seven-segment display, 12-bit VGA output, 5 push buttons and USB interaction facilitates enough space to design and implement the proposed game. My game uses 5 push buttons to control the movement of the bot and the USB port to output the 10-bit VGA output signal to the monitor. It provides flash memory of 1.8Kb to store the program and display the interactive game. The 10-bit input signal is supported on the board to process it with the 333Mhz clock required to execute the Verilog program. The Xilinx based FPGA board allows Verilog programming through Vivado software to build the behavioral, schematic structure and provides the clock to drive the interactive game.
The Basys3 FPGA board is shown in appendix apx.12

VGAController:

I created two counters to create my VGA controller, one for the current row and other for the current column in use. The row counter will increment only after column counter reaches the last pizel. The output of the Horizontal counter is called Hcount and the output of the vertical counter is called Vcount. After verifying the working of the counters, I outputted Hsync, Vsync, active region (AR) and frame as follows. The frames were defined high for one pixel per frame to depict the bot.

Hsync = (Hcount <= 10b'1010001110) | (Hcount >= 10b'1011101111)
Vsync = (Vcount <= 10b'0111101000) | (Vcount >= 10b'0111101011)
AR = (Hcount <= 10b'1001111111) | (Vcount <= 10b'0111011111)
Frame = (Hcount == 10b'1010010100) | (Vcount >= 10b'0111101010)

Hsync and Vsync are used to define the gap region of the obstacles, while AR specifies the active region of pixels in which the bot is located, and the frame defines the frame of the region.

The schematic diagram of the VGA Controller is enclosed in appendix apx.1

Stealthy Bot:

A state machine was built to reset the bot to initial position (coordinate (10, 10)) every time the game is restarted. The machine only consists of 2 states namely initialize and play. To start with the program to run the game btnC is pressed where the states remain in initialize state (10,10) and loaded as starting position for the bot. Once the movement in detected through up, down, left, or right button the counter stops loading in initial position and state machine Is driven to play until center button or the reset is pressed.

The state machine diagram for stealthy bot is shown in appendix apx.2
The schematic diagram for bot movement is enclosed in appendix apx.3

Obstacle:

A state machine was built to develop each obstacle individually. A gap counter was placed to keep the track of left reference pixel to show the beginning of each gap in the respective obstacle module. The state machine consists of 4 states namely, initialize, up, down and flash. The input to the state machine is provided by the counters Vcount, Hcount and the output is a single bit signal that makes the current pixel green indicating the bot is passing through the obstacle. When the bot is in line with the obstacle, then the entire obstacle is displayed using colors for the obstacle and the bot without having any gap in it. If obstacle changes to its original design whether the bot is moved up or down is determined by the direction variable. IF the flash state is encountered, the gap stops moving and the obstacle as well as the bot begins to flash.

The state machine diagram for obstacle is shown in appendix apx.4
The schematic diagram for Horizontal Obstacle is enclosed in appendix apx.5
The schematic diagram for Vertical Obstacle is enclosed in appendix apx.6

Flash Clocks:

Flash clocks are used to flash the bot and the obstacle when the bot fails to pass through the obstacle. I created a model that would count a certain number of frames and reset when the center button (btnC) is pressed. A flipflop is used to produce the signal that is alternated between high and low whenever the register storing the end of game is ON. This register is wired to reset counter to initialize the game to the starting position when reset button is pressed. This way the signal passing through the register is used to flash bot and the obstacle impacted by passing high and low for a certain number of frames.

The schematic diagram for flashing clock is enclosed in appendix apx.7

Time Counter:

Every time the game is started, a timer is activated on the & segment display which depicts the seconds and minutes of the game runtime. A time counter is created that counts to 60 frames (1 second) and then resets where the second counter starts to increment for every reset from counter 1. If the module

receives a flash signal the 7-segment display begins to flash until the game is reset.

The schematic diagram for Timer is enclosed in appendix apx.8

Debugging:

VGA monitor is debugged using simulator. It is run in a particular manner where the Hsync and Vsync counters count every clock cycle and changes the output signal precisely. As the simulator took longer time to run it was difficult to determine the frame size, so I generated a bitstream and figured out the errors in my code by running and testing the game for bounds and collision logic.

Conclusion:

Overall, I had learnt new concepts and overcame many constraints before completing the project. The liberty of designing The Stealth Bot game was immensely enjoyable. I could apply many of the concepts that I learnt in the coursework and implement them on my own design. I was able to design the VGA Controller to get Hsync and Vsync working followed by displaying a solid color on screen. Then I developed a blue border for the region and the state machine to implement the behavioral structure of the bot to move around the screen and the obstacle to challenge the movement of the bot.
The snapshot of the gameplay is enclosed as follows:

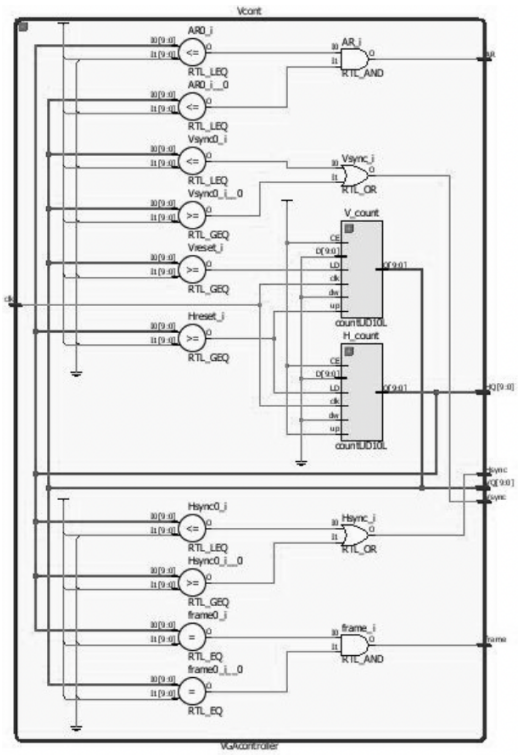
The game initial position is shown in appendix apx.9
The gameplay is shown in appendix apx.10
The Basys3 FPGA involved in the gameplay is shown in appendix apx.11

References:

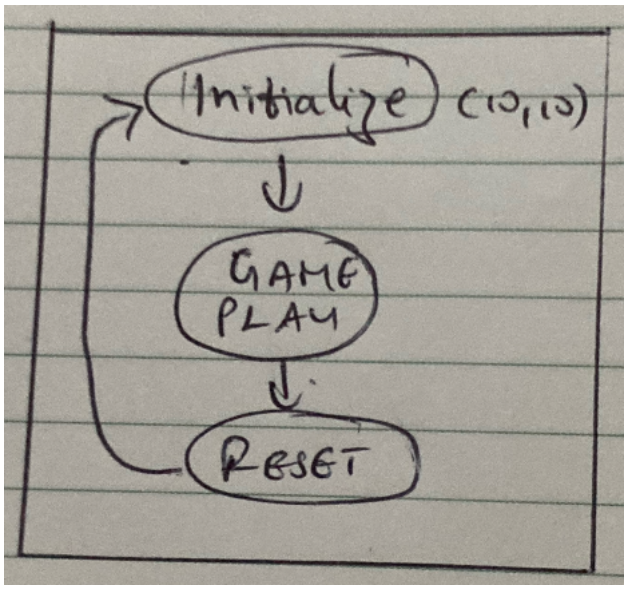
- https://www.xilinx.com/support/documentation/university/VivadoTeaching/HDLDesign/2015x/Basys3/Supporting%20Material/Basys3_RM.pdf -- to study the features about BASYS3 FPGA board.
- https://www.fpga4student.com/2017/09/seven-segment-led-display-controller-basys3-fpga.html ---- for interfacing seven-segment display.

Appendix:

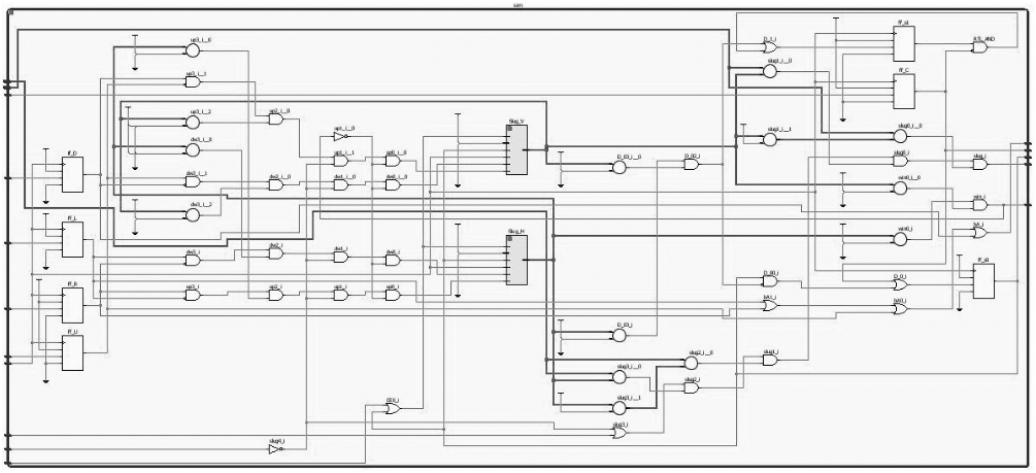
Appendix apx.1: The schematic diagram of the VGA Controller.



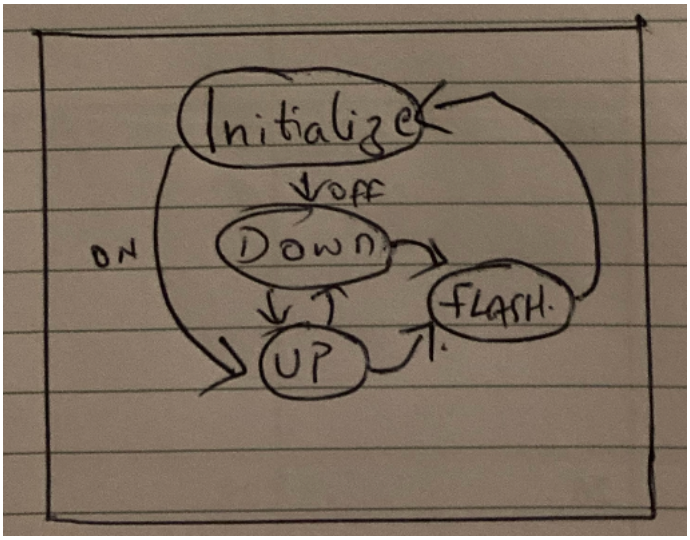
Appendix apx.2: The state machine diagram for stealthy bot.



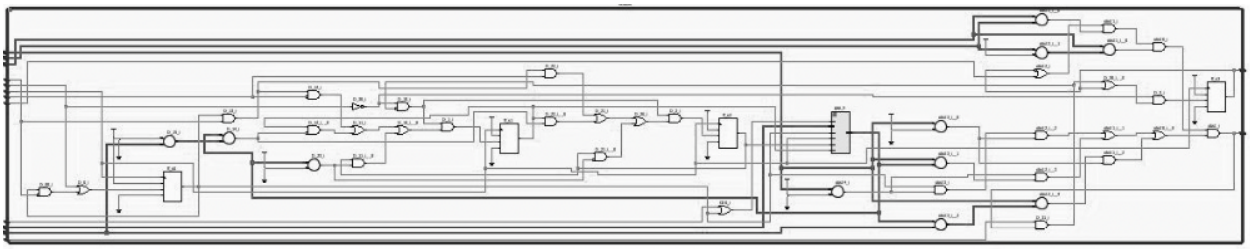
Appendix apx.3: The schematic diagram for bot movement



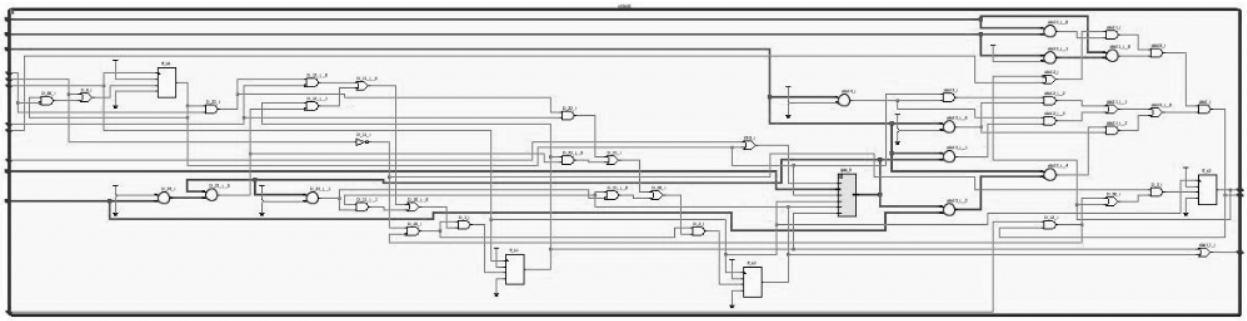
Appendix apx.4: The state machine diagram for obstacle.



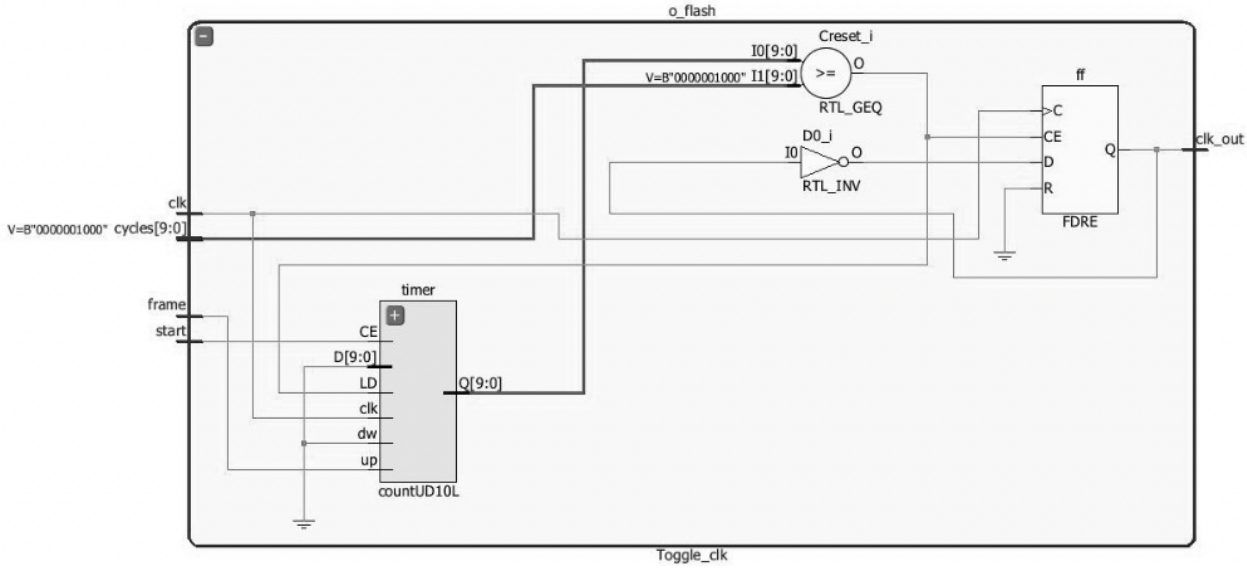
Appendix apx.5: The schematic diagram for Horizontal Obstacle.



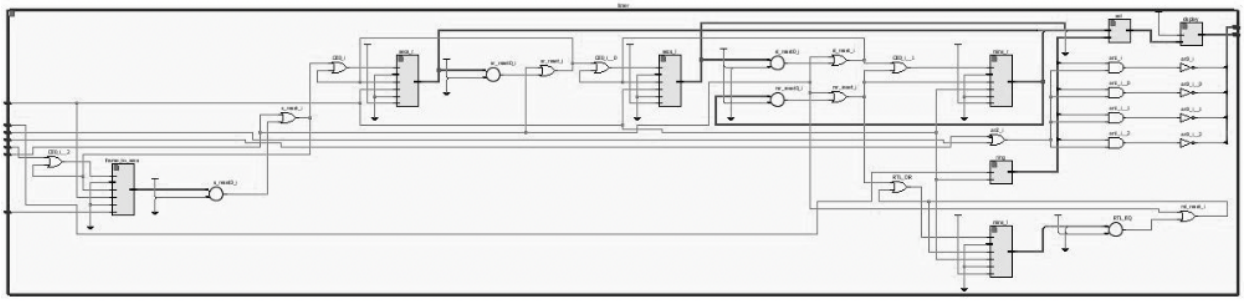
Appendix apx.6: The schematic diagram for Vertical Obstacle.



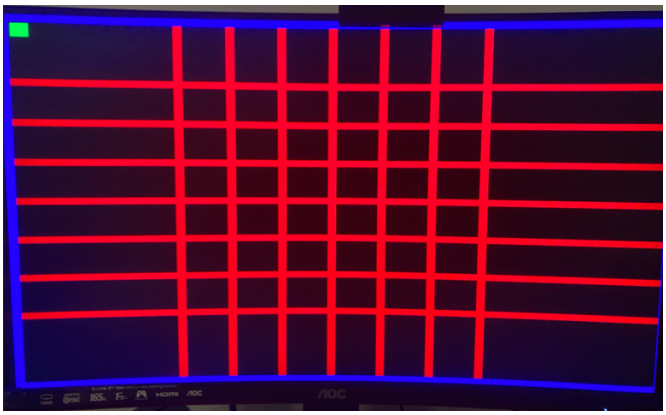
Appendix apx.7: The schematic diagram for flashing clock.



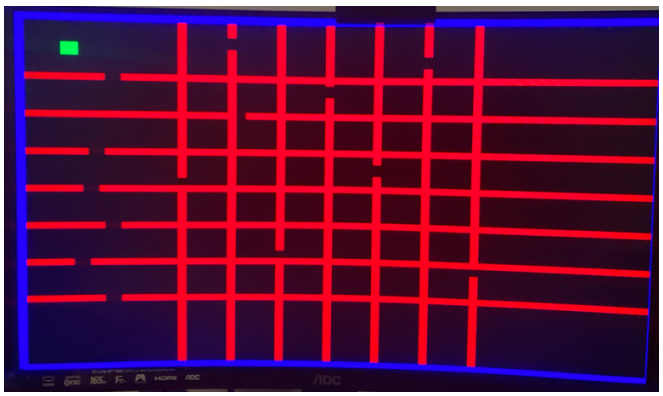
Appendix apx.8: The schematic diagram for Timer.



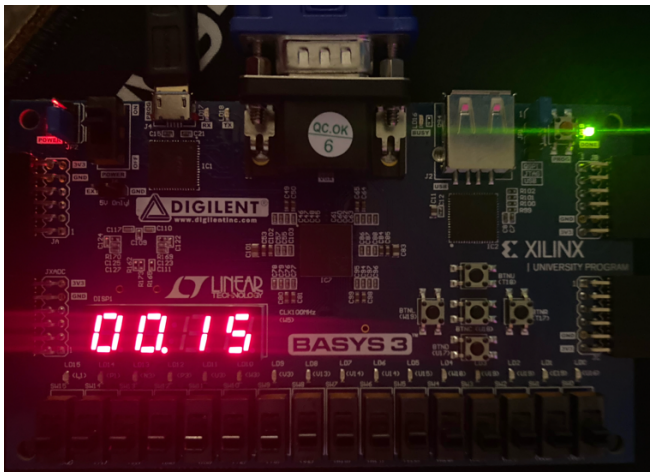
Appendix apx.9: Initial stage of the game.



Appendix apx.10: Gameplay.



Appendix apx.11: The Basys3 FPGA during in the gameplay.



Appendix apx.12: Basys3 FPGA board.