EHB 432E Embedded System Design Final Project, 2017

Project S

Osman Can Başaran, Ali Gür Güvenilir, Can Özbaş

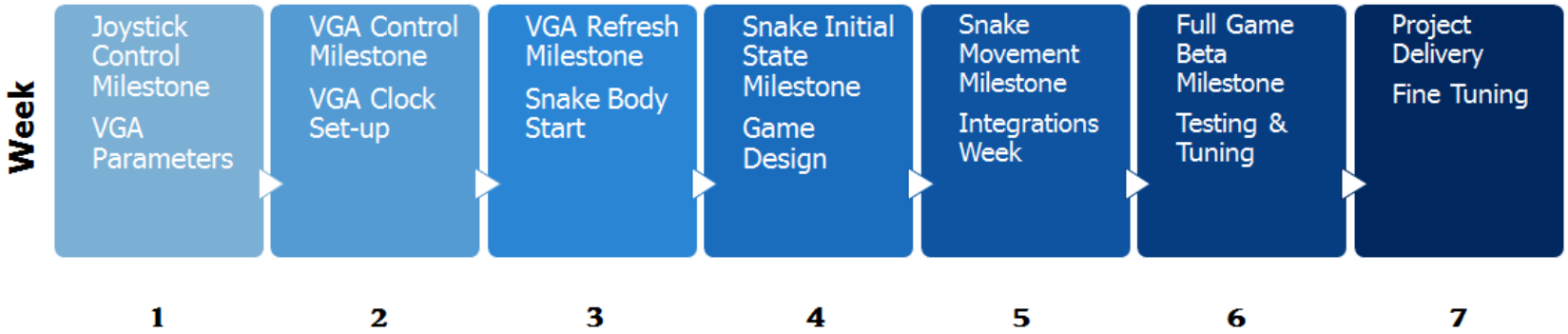
Introduction

For the Y generation game concept started with SEGA, NES and Nintendo games like Tetris, Sonic, Zelda and so forth. In this list there is a game that has a very special place for every 90s kids' heart which is Snake Game. Snake Game is a very simple yet so addicting game of a snake chasing a target (apple) without touching the borders or its tail. Whenever snake eats a target (apple) its tail grows accordingly and game speeds a pinch, this results in a very hard game mod on the later stages of the game. This project is a nostalgia project bringing our favorite childhood game into PC monitor with a FPGA controlled joystick. The system uses a Digilent PMOD-JSTK joystick module as snake control input and a VGA port output for the monitor display. We developed this project on Digilent Nexys 2 FPGA and uploaded the snake algorithm developed on Xilinx's ISE Project Suite in several Verilog modules.

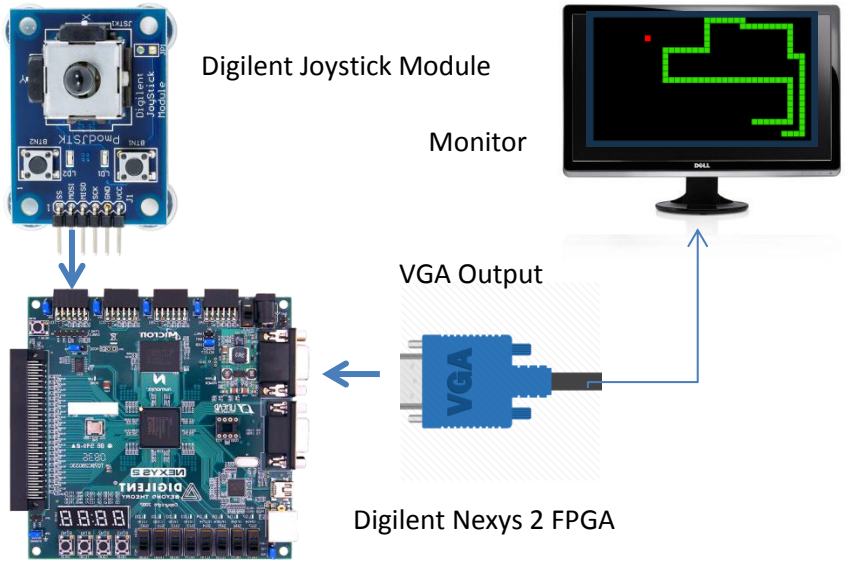
Work Packages

Group	Sub-Group	Sub²-Group	Work Package	Owner	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7
1	1.1	1.1.1	VGA Parameters	Ali Gür							
		1.1.2	VGA Generation	Ali Gür							
	1.2	1.2.1	VGA Clock	Can							
		1.2.2	VGA Horizontal&Vertical Synchronization	Can							
2	2.1	2.1.1	Joystick Control	Can							
	2.2	2.2.1	JoyStick CLK Division	Osmancan							
		2.2.2	JoyStick SPI Control	Can							
		2.2.3	JoyStick Master	Osmancan							
3	3.1	3.1.1	Snake Body Generation	Ali Gür							
		3.1.2	Snake Movement	Ali Gür							
	3.2	3.2.1	Random Grid	Osmancan							
		3.2.2	Apple Generator	Osmancan							
		3.2.3	Collisions (Lethal/Nonlethal)	Ali Gür							
4	4.1	4.1.1	Joystick & VGA Integration	Can							
	4.2	4.2.1	Algortihm Integration	Ali Gür							
5	5.1	5.1.1	Peripheral Test	Osmancan							
	5.2	5.2.1	Algorithm Test	Osmancan							
	5.3	5.3.1	Fine Tuning	Can							
6	6.1	6.1.1	Poster Preperation	Osmancan Ali Gür							
Milestones				All	JoyStick Control	VGA Control	VGA Refresh	Snake Initial State	Snake Movement	Full Game Play	Successful Fine Tuning

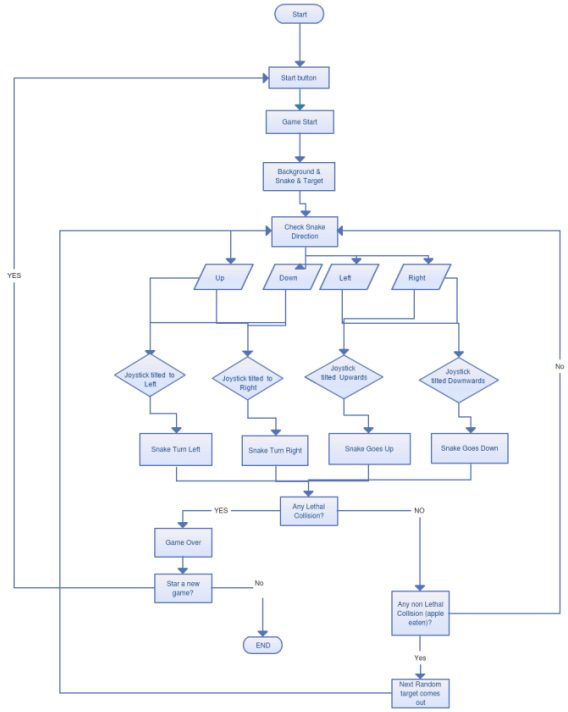
Work Flow



System Architecture



ASM Diagram



Technical Architecture

Digilent PMOD-JSTK Joystick Module:

The PmodJSTK is designed to be a versatile peripheral module that can be used in a wide variety of projects. It contains a resistive twin axis joystick that includes a center push button along with two additional push buttons. The PmodJSTK also has two programmable LEDs located on the board that can provide additional information to the user. The PmodJSTK is ideally suited for Digilent microcontroller or FPGA based projects that require proportional control from the user, such as robotic applications.

Features:

- 2-axis resistive joystick with central push button
- Two additional user push buttons
- Two user indicator LEDs
- Small PCB size for flexible designs 1.8" × 1.3" (4.6 cm × 3.3 cm)
- 6-pin Pmod connector with SPI interface
- Follows Digilent Pmod Interface Specification Type 2

Functional Description:

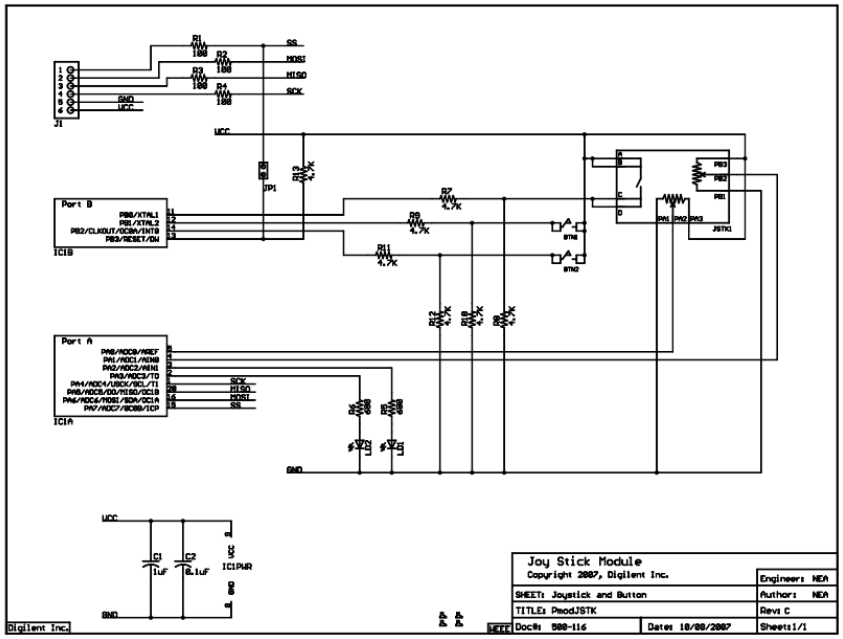
The PmodJSTK uses an Atmel® ATtiny24 microcontroller in a MLF20 package to collect information about its peripherals. The twin axis joystick uses two potentiometers to measure the current position in the x and y coordinate directions and stores the information in two 10-bit values ranging from 0 to 1023.

Pin Descriptions:

Pin	Signal	Description
1	~CS	Chip Select (active low)
2	MOSI	Master-Out-Slave-In
3	MISO	Master-In-Slave-Out
4	SCK	Serial Clock
5	GND	Power Supply Ground
6	VCC	Power Supply (3.3V/5V)

Table 2. Pin descriptions for the PmodJSTK.

Schematic:

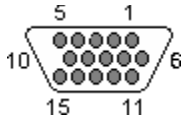


VGA Output:

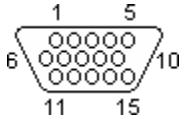
A Video Graphics Array (VGA) connector is a three-row 15-pin DE-15 connector. The 15-pin VGA connector was provided on many video cards, computer monitors, laptop computers, projectors, and high definition television sets. On laptop computers or other small devices, a mini-VGA port was sometimes used in place of the full-sized VGA connector.

Pin Descriptions:

Videotype: Analogue. VGA (31.5 KHz - 640x480) SVGA (35-37 KHz - 800x600)



15 PIN HIGHDENSITY D-SUB FEMALE at the videocard



15 PIN HIGHDENSITY D-SUB MALE at the monitor cable

PIN Descriptions:

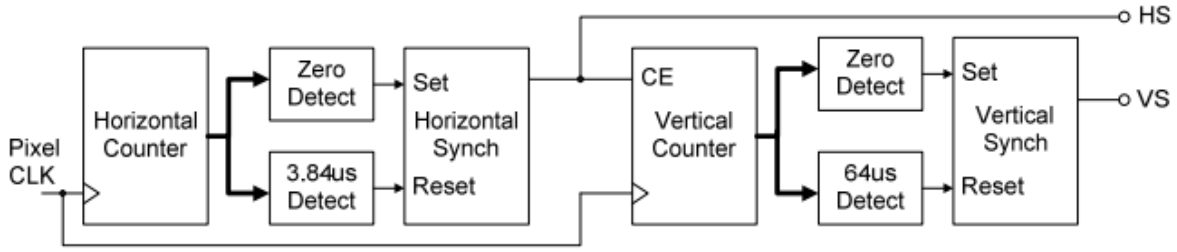
Pin	Name	Dir	Description
1	RED	OUT	Red video
2	GREEN	OUT	Green video
3	BLUE	OUT	Blue video
4	ID2 / RES	IN	Reserved for E-DDC
5	GND	-	Ground (HSynch)
6	RED_RTN	-	Red return
7	GREEN_RTN	-	Green return
8	BLUE_RTN	-	Blue return
9	KEY / PWR	-	+5 V DC

10	GND	-	Ground (VSynch, DDC)
11	ID0 / RES	IN	Reserved for E-DDC
12	ID1 / SDA	IN	PC Data
13	HSynch	OUT	Horizontal synch
14	VSynch	OUT	Vertical synch
15	ID3 / SCL	IN	PC Clock

VGA Sync:

Symbol	Parameter	Vertical Sync			Horiz. Sync	
		Time	Clocks	Lines	Time	Clks
T_S	Sync pulse	16.7ms	416,800	521	32 us	800
T_{disp}	Display time	15.36ms	384,000	480	25.6 us	640
T_{pw}	Pulse width	64 us	1,600	2	3.84 us	96
T_{fp}	Front porch	320 us	8,000	10	640 ns	16
T_{bp}	Back porch	928 us	23,200	29	1.92 us	48

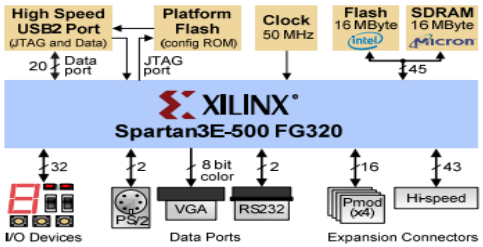
VGA Controller Schematic:



Digilent Nexys 2 FPGA:

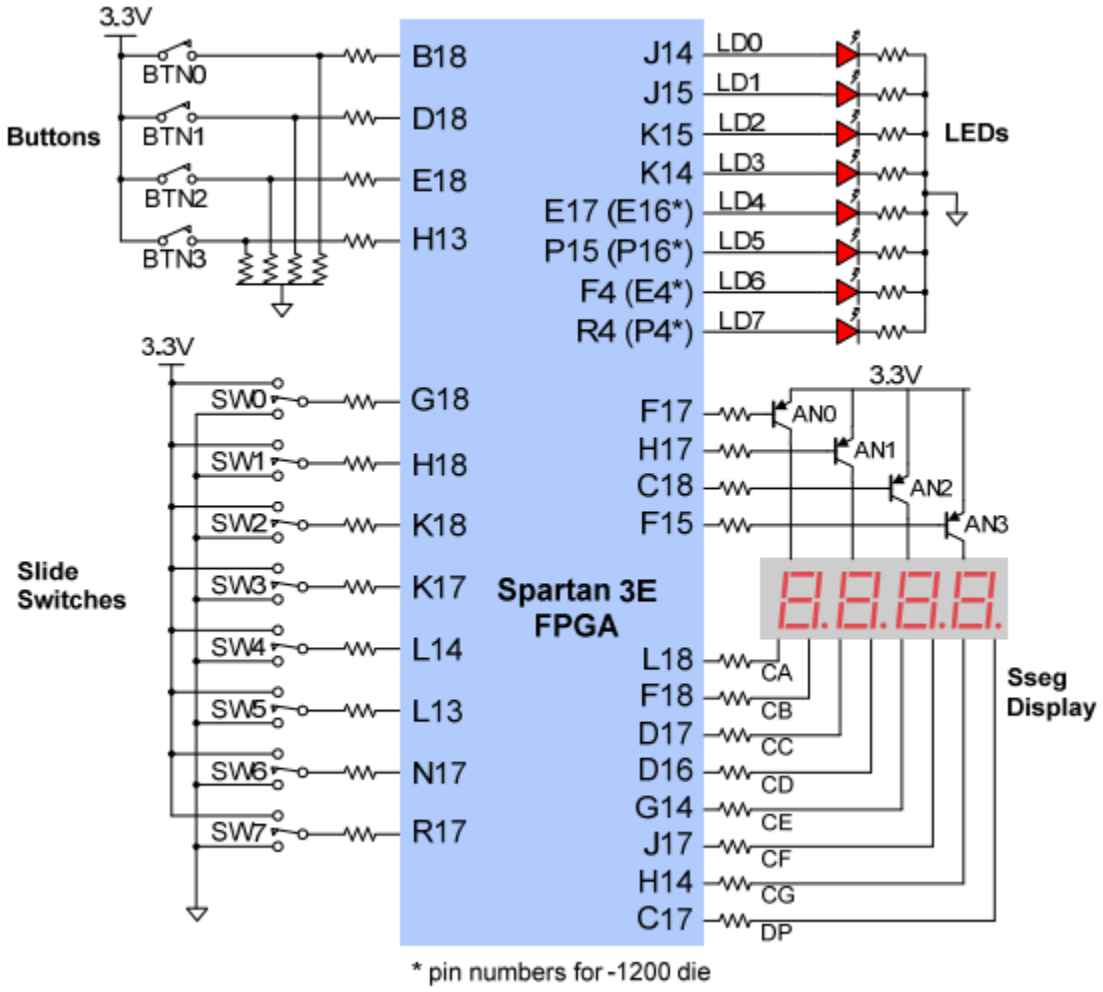
The Nexys2 circuit board is a complete, ready-to-use circuit development platform based on a Xilinx Spartan 3E FPGA. Its onboard high-speed USB2 port, 16Mbytes of RAM and ROM, and several I/O devices and ports make it an ideal platform for digital systems of all kinds, including embedded processor systems based on Xilinx's MicroBlaze. The USB2 port provides board power and a programming interface, so the Nexys2 board can be used with a notebook computer to create a truly portable design station.

It can host countless FPGA-based digital systems, and designs can easily grow beyond the board using any or all of the five expansion connectors. Four 12-pin Peripheral Module (Pmod) connectors can accommodate up to eight low-cost Pmods to add features like motor control, A/D and D/A conversion, audio circuits, and a host of sensor and actuator interfaces



- *500K-gate Xilinx Spartan 3E FPGA*
- *USB2-based FPGA configuration and high-speed data transfers (using the free Adept Suite Software)*
- *USB-powered (batteries and/or wall-plug can also be used)*
- *16MB of Micron PSDRAM &16MB of Intel StrataFlash ROM*
- *Xilinx Platform Flash for nonvolatile FPGA configurations*
- *Efficient switch-mode power supplies (good for battery powered applications)*
- *50MHz oscillator plus socket for second oscillator*
- *60 FPGA I/O's routed to expansion connectors (one high-speed Hirose FX2 connector and four 6-pin headers)*
- *8 LEDs, 4-digit 7-seg display, 4 buttons, 8 slide switches*
- *Ships in a plastic carry case with USB cable*

Nexys 2 I/O:


3.3V
BTN0
BTN1
BTN2
BTN3
Buttons
B18
D18
E18
H13
J14
J15
K15
K14
E17 (E16*)
P15 (P16*)
F4 (E4*)
R4 (P4*)
LD0
LD1
LD2
LD3
LD4
LD5
LD6
LD7
LEDs
3.3V
SW0
SW1
SW2
SW3
SW4
SW5
SW6
SW7
Slide Switches
G18
H18
K18
K17
L14
L13
N17
R17
Spartan 3E FPGA
F17
H17
C18
F15
AN0
AN1
AN2
AN3
L18
F18
D17
D16
G14
J17
H14
C17
CA
CB
CC
CD
CE
CF
CG
DP
Sseg Display
* pin numbers for -1200 die

Milestone Steps

This part explains briefly what has been done in each milestone during the project development phases. Each milestone adds up to each other and forms a structured project plan to design a snake game for Nexys 2 FPGA module using Digilent Joystick module as input and VGA as output.

Milestone #1

The purpose of the demo was to be able to control the Digilent Joystick module signals to differentiate 2-axis movement by using the Cartesian coordinate values generated by the Joystick module to trigger different LEDs. This milestone also leans on the two buttons on the module and trigger different LEDs with press signal. In the demo code of the milestone data is sent and received to and from PmodJSTK at 5 Hz. The positional data of the Joystick ranges from 0 to 1023 for both x and y axis. We trigger the initial LED if the x value generated from the joystick is less than 400, if it's greater than 600 we trigger the second LED. The in between values do not trigger any LEDs. For the y axis we perform the same logic for third and fourth LEDs. With the two buttons also we trigger another pair of LEDs, so that during the project development we can implement Joystick module buttons to the architecture easily. For the parameters, registers and wires the demo code uses an active low wiring, a data transfer from master to slave, a serial clock, a status of PmodJSTK buttons that is displayed on LED, a data to be sent to PmodJSTK, a signal to send/receive to/from PmodJSTK and a signal to carry output data that user selected registers and wires. In order to operate with FPGA, we use the input 100MHz clock of the module to convert to a 66.67 kHz as clock signal.

Milestone #2

For the milestone #2, VGA output is tested. Nexys 2 FPGA works with a clock of 50MHz but we need to reduce it to 25Mhz for pixel clock of VGA. We used the VGA output to generate 640x480 pixels screen resolution. We used 7 pins of the VGA controller such as Ground, VCC, R, G, B, Vsync, Hsync. The Vsync and hsynch pins would be the VGA control for the monitor that scans the screen at 25MHz starting from top left corner. Certain pixels are spared for different purposes on the VGA controller and were not visible on monitor. So the algorithm to change colour and but a stationary block on the screen pixel adjustments were made. Also for the ease of further controlling the VGA output, we divided the 640x480 pixel screen into 8x8 blocks and stored it as a matrix. In the matrix our 8x8 pixels would be just one block that is put on a screen of 80x60 blocks resolution. We randomly picked differentiated colors and ended up with bright bluish background and a pinkish block in the middle of the screen.

Milestone #3

For the third milestone we studied to bring PmodJSTK module and VGA controller to work together on the Nexys 2 FPGA. For this we wired the PmodJSTK pins with the VGA controller and FPGA pins. Our PmodJSTK were working on 5Hz and our VGA controller on 25MHz clocks; that's why we made delays in our output pixel movement algorithms with 10^6 nanoseconds delays. For this milestone we decided to be able to initialize the stationary block from milestone 1 with a direction and constant movement speed. We adjusted the movement for the gameplay and we changed the 5 bit direction parameter with joystick input of x and y coordinates. If the block movement direction and joystick direction are the same then nothing happened, otherwise we changed the direction only movement algorithm was stable. Since there are non-active video pixels on VGA, boundary conditions must be set to prevent excessive horizontal & vertical synchronization duration. That's why we decided to prevent block from moving our of the boundary active pixels. If the block reaches our 80x60 block wide resolution, then the direction is reversed and to get the block out of boundary condition we move it in the same clock timing one block away from the boundary. This created vivid bounce-back effect on the monitor output.

Milestone #4

For the fourth milestone we created a random x and y coordinate generator for the apple (target). The module worked with the PosEdge of the VGA clock at 25MHz and each time the apple is eaten the next coordinate of apple is made visible on the monitor. The snake head block is controlled inside the snake module if the x and y coordinates of the snake is matched the algorithm called for the next apple location. The target is 8x8 pixels same size as the snake head and the color is adjusted to be differentiated from the snake. Joystick module still works efficiently for the snake head. The next milestone will be based on the addition of the snake tail and the whole snake movement algorithm is needed to be updated in order to allow size growth.

Milestone #5

For the fifth milestone we added a tail to the snake head and stored the whole snake body on an array of registers at the maximum size of 50. The initial state of the snake size is three and after each good collision with the apple snake size is increased in a for loop that is controlled with a size parameter. The snake movement is updated in order to create seamless movement effect. The array values is shifted from heads towards the tail according to current direction of the snake in each clock cycle. This created a fluent movement visual and easier adaptation of size growth for the full body snake. The maximum size is limited to 50 in 8x8 blocks because Nexys 2 FPGA logic

gates enables a smooth operation with a maximum of 50 snake size because of the loops included in the control and application of the game.

Milestone #6

For sixth and the last milestone the game is updated to include the full features such as game pause, game end, scoreboard, bad collisions with tail and borders. The whole game pause, game end and bad collisions are done inside snake module in order to accommodate variable transfer errors between modules in Verilog. The game update cycle is controlled via 1 bit parameter that is connected to a switch on the FPGA. The snake head if collided with its body which is checked inside the snake body movement loop with an additional loop. The update module also checks whether snake's next coordinates collide with our 8 pixel length border. At the end of the game or at any bad collisions the game freezes and snake body flashes. The game can be restarted with restart button where it resets the game to its initial state where apple's current location stays the same. The restart button is connected onto a button on the FPGA. The PicoBlaze is utilized on the keeping the score of the game via 7-segment display on the FPGA. The PicoBlaze takes the good collisions on the game and calculates the point of the player according to the current size; the output score is transferred onto FPGA's 7-segment display to feature the score to the player.

Algorithm

https://drive.google.com/file/d/0ByBSlGm9JEGvZEpXamRlNWgzbW8/view?usp=sharing

Conclusion

As a final project Project S, retro snake game on first generation mobile phones on the FPGA platform enabled the team to utilize different peripherals and algorithm approaches to understand and learn to work with FPGA and micro controller. The project was a challenging one with a joystick module as an input rather than a keyboard or buttons and a very challenging output module, VGA controller. The project was a success and as a team we created fun to play and an addicting game that is created extensively on Verilog with Xilinx ISE Software. In the future projects we can easily utilize the skills and the know-how we have gathered in this project to deliver visual outputs, get complex input modules and create high level algorithm to control our peripherals.