MICROPROCESSOR SYSTEM DESIGN

FINAL PROJECT

TETRIS

Burak Mert GONULTAS

040140003

Ilayda YAMAN

040140057

Talip Tolga SARI

040140007

INTRODUCTION

Tetris is a tile-matching puzzle game, created and programmed by a Russian game designer Alexey Pajitnow. The game made its debut on June 6, 1984, while Pajitnov was working for the Dorodnitsyn Computing Centre of the Academy Science of the Soviet Union in Moscow. The game is available for nearly every video game console and computer operating system as well as on many digital devices such as graphing calculators, mobile phones, portable media players, network music players and even on oscilloscopes.

The purpose of the project is to design and implement a simple platform to play Tetris game on a "Digilent Nexys 2 FPGA" with the help of a soft microprocessor, PicoBlaze, a VGA unit and switches which will be used for input types. Except the control of the VGA unit, most of the game rules were controlled by PicoBlaze and switches were used for left and right shifts, rotation and speedup.

1. Implementation of VGA is done to adjust H_synch and V_synch for a static VGA output of a block and a background.
2. Switch inputs and VGA output modules are combined to deliver a 22x10 block control over a 640x480 resolution monitor.
3. Blocks are generated with a random generator algorithm.
4. Game mechanics are added such as game finish, game pause, rotation, collision with right, left and bottom boarders by programming the PicoBlaze.

GENERAL SYSTEM ARCHITECTURE

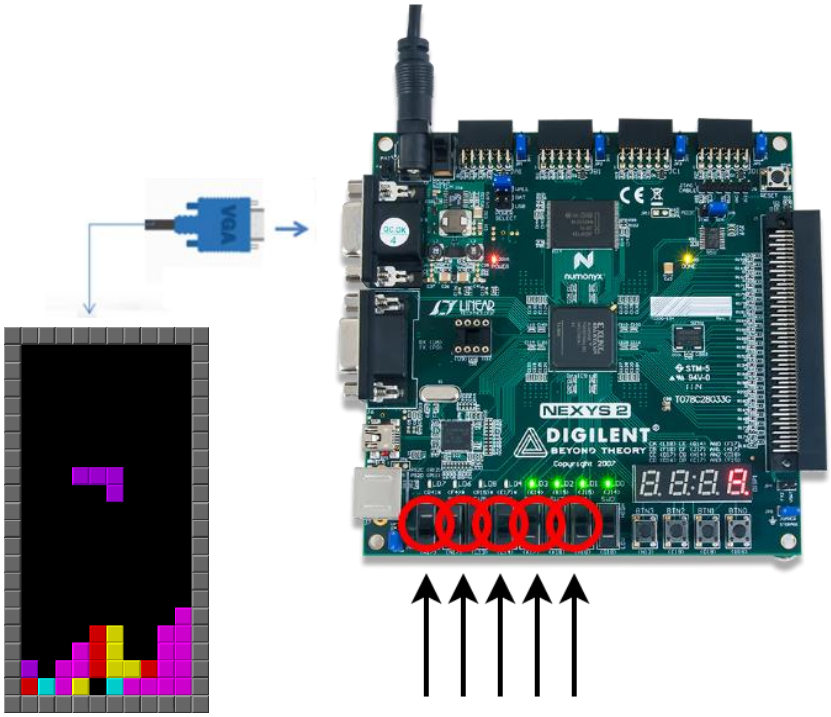


Figure 1: showing the general architecture of the system

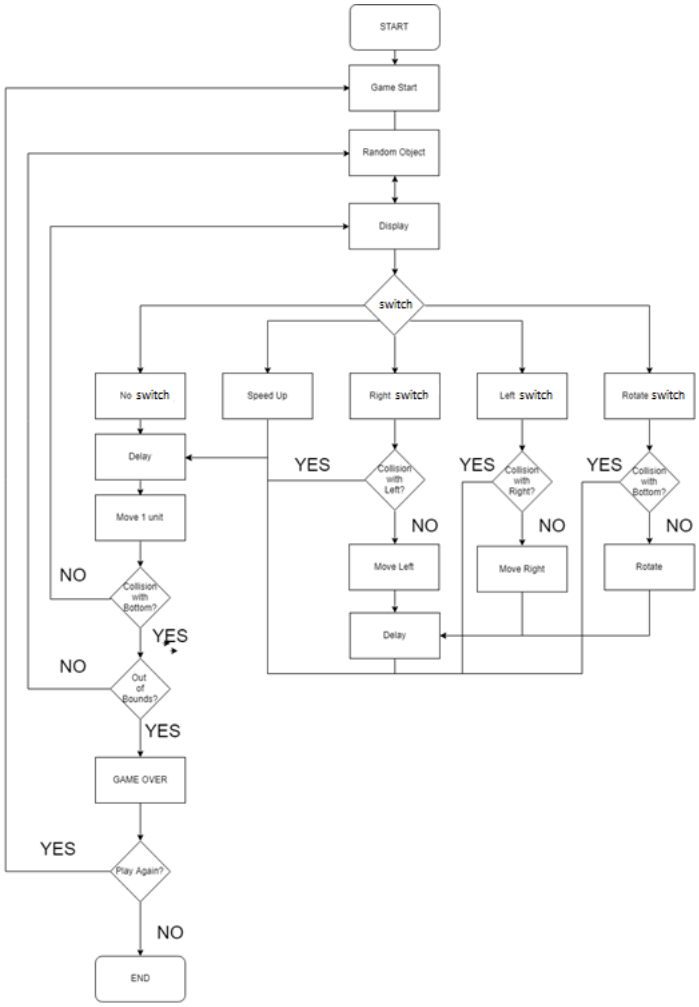


Figure 2 ASM of the system

OVERALL DATA PATH

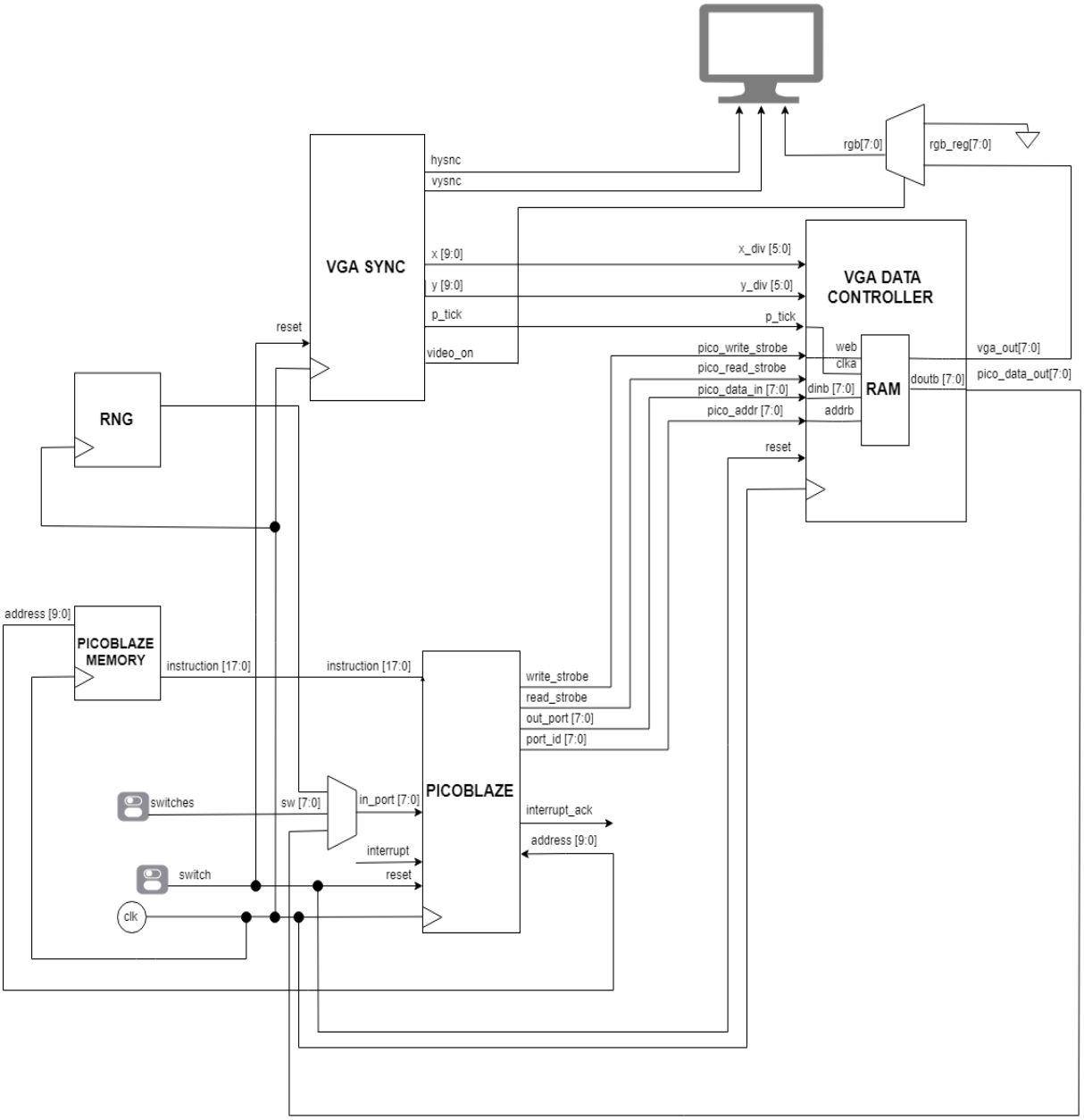


Figure 3: showing the overall data path of the system

GANTT CHART

GROUP	SUBGROUP	WORK PACKAGE	PERSON	20.04.2018	27.04.2018	04.05.2018	11.05.2018	18.05.2018	08.06.2018
	1.1	VGA Parameters	I/T/M						
	1.2	VGA Clock	I/T/M						
	1.3	VGA Generation	I/T/M						
1	1.4	VGA Horizantal and Vertical Synchronization	I/T/M						
		Switch Input Read	I						
2	2.1	Switch Functions	T/M						
	3.1	Random Generation	T						
	3.2	Block Representation	M						
3	3.3	Simple Game Rules	I						
	4.1	Game Start/Over	M						
	4.2	Sliding Blocks	T						
4	4.3	Edges	I						
	5.1	Shift	I						
	5.2	Rotate	T						
5	5.3	Speedup	M						
	6.1	Software Test	M						
	6.2	Hardware Test	T						
6	6.3	Poster	I						
MILESTONES				INPUT CONTROL	OUTPUT CONTROL	BLOCKGEN	GAME DESIGN	BLOCK DYNAMICS	PRESENTATION

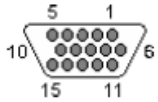
WORK FLOW

TECHNICAL ARCHITECTURE

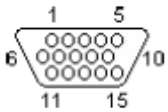
VGA Output

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. Switch inputs and VGA output modules are combined to deliver a 22x10 block control over a 640x480 resolution monitor in our project.

Pin Descriptions:
Video Type: Analogue. VGA (31.5 KHz - 640x480) SVGA (35-37 KHz - 800x600)



15 PIN HIGHDENSITY D-SUB FEMALE at the Video Card



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 (H_Synch)
6	RED_RTN	-	Red return
7	GREEN_RTN	-	Green return
8	BLUE_RTN	-	Blue return
9	KEY / PWR	-	+5 V DC
10	GND	-	Ground (V_Synch, DDC)
11	ID0 / RES	IN	Reserved for E-DDC
12	ID1 / SDA	IN	PC Data
13	H_Synch	OUT	Horizontal synch
14	V_Synch	OUT	Vertical synch
15	ID3 / SCL	IN	PC Clock

VGA Sync:

Display area on the screen using VGA has a resolution of 640 by 480 pixels. The origin of the display area is (0,0) and is located in the upper left corner. The x dimension increases from left to right, while y increases from top to bottom.

Figure 4: showing the function of H_Sync

Figures are showing the timing signals produced by the VGA controller. The controller contains two counters. One counter increments on pixel clocks and controls the timing of the h_sync (horizontal sync) signal. By setting it up such that the display time starts at counter value 0, the counter value equals the pixel's column coordinate during the display time. The horizontal display time is followed by a blanking time, which includes a horizontal front porch, the horizontal sync pulse itself, and the horizontal back porch, each of specified duration. At the end of the row, the counter resets to start the next row.

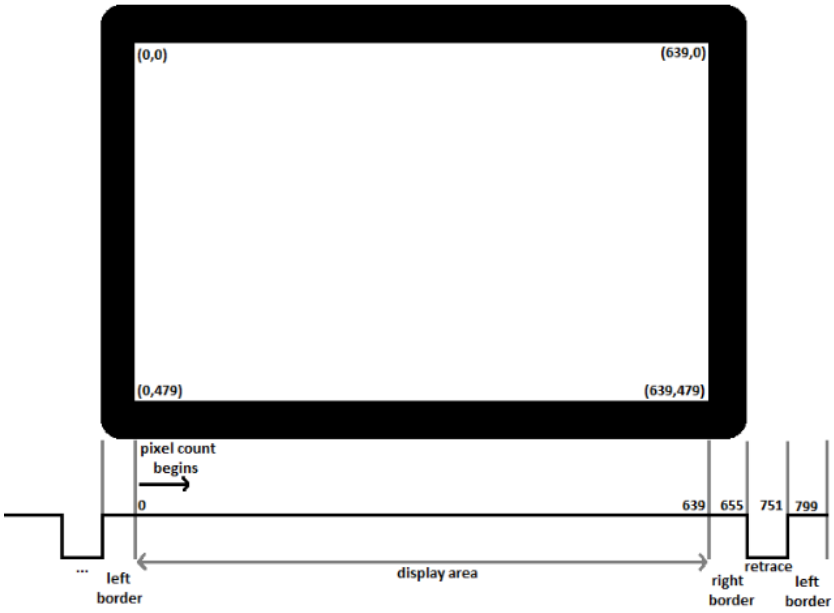
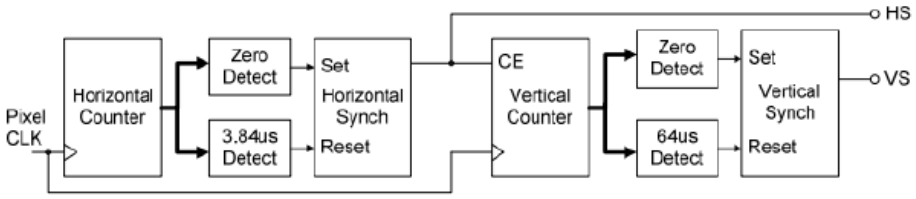


Figure 5 : showing the function of V_Sync

The other counter increments as each row completes, therefore controlling the timing of the v_sync (vertical sync) signal. Again, this is set up such that the display time starts at counter value 0, so the counter value equals the pixel's row coordinate during the display time. As before, the vertical display time is followed by a blanking time, with its corresponding front porch, sync pulse, and back porch. Once the vertical blanking time completes, the counter resets to begin the next screen refresh. A display enable is defined by the logical AND of the horizontal and vertical display times.

Using these counters, the VGA controller outputs the horizontal sync, vertical sync, display enable, and pixel coordinate signals. The sync pulses are specified as positive or negative polarity for each VGA mode. The GENERIC parameters h_pol (horizontal polarity) and v_pol (vertical polarity) set the polarity of the VGA controller's h_sync and v_sync outputs, respectively.

VGA Controller Schematic:



Digilent Nexys 2 FPGA:

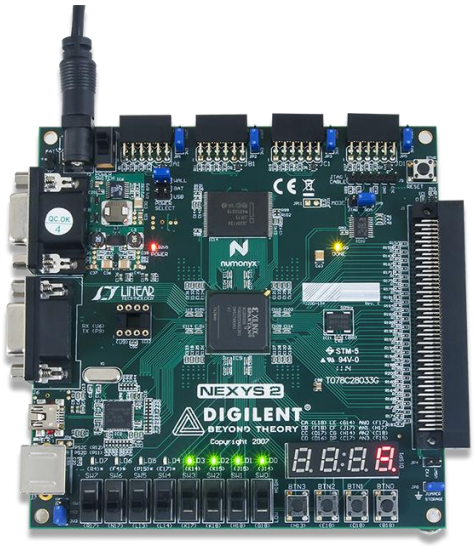


Figure 6: showing the Nexys2 circuit board

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, therefore it is possible to use Nexys2 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.

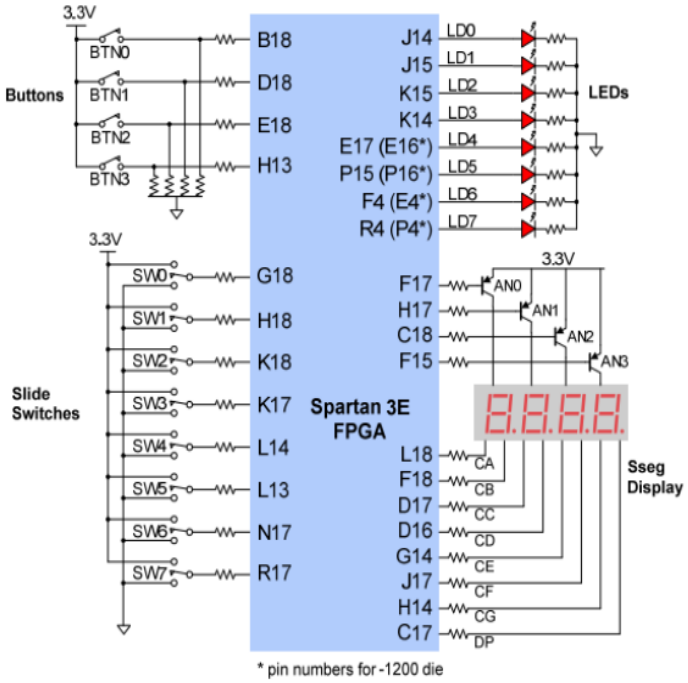


Figure 7: showing I/O pins of Nexys 2

MILESTONE STEPS

This part explains briefly what has been done in each milestone during the project development phases. Each milestone adds on top of each other and forms a structured project plan to design a Tetris game for Nexys 2 FPGA module using PicoBlaze as the control unit, switches as input and VGA as output.

Milestone #1

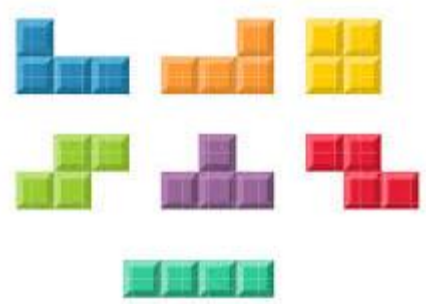
VGA Output synchronization has been accomplished in this milestone. Classical Tetris game requires 22x10 blocks so they are created from 640x480 resolution monitor. V_Sync which controls the coordinates in vertical and H_Sync which controls the y coordinate in horizontal is divided to create 22x10 playground where some of the monitor parts have not been used. The color of the environment is also set in this milestone.

Milestone #2

Second milestone is about switch inputs and VGA output modules and combining them to deliver the game environment. General module is arranged to register switch inputs which are right, left, speedup, rotation and reset. User Constraint File is added to the project. A VGA data controller module has been created since VGA output module requires a true dual port block memory which has been created by block memory generator. Colors of the background and playground have been set.

Milestone #3

In this milestone, blocks are generated with a random generator algorithm. All blocks have 4 parts and there are 7 unique blocks. The form of different parts has been coded manually.



Since the game requires, different blocks need to fall randomly to game environment. A module in Verilog has been coded to generate pseudo-random numbers and give to PicoBlaze so it will use as a seed. As a result, the blocks are generated randomly and created in the game environment.

Milestone #4

First part of the game mechanics is added such as game finish, game pause, rotation, collision with right, left and bottom boarders by programming the PicoBlaze. Collisions are handled starting with bottom boarder, and then right and left boarders. Afterwards the collisions of overlapping blocks have been handled. Rotation also required a second layer of collision control. All these steps were controlled by PicoBlaze and Assembly language is used.

Milestone #5

In the last milestone, the game is updated to include the full features such as the reset, deletion and speedup. These are also done in Assembly language to be controlled by PicoBlaze. The game can be restarted with restart button where it resets the game to its initial state where the game environment is clear. The restart button is connected onto a switch on the FPGA. The speed of the game can be put on fast track by another switch. Deletion of the blocks in the same row when all the blocks are filled caused many bugs such as collision of parts without a reason or unpredictable behavior of the system. Most of the bugs have been fixed.

CONCLUSION

At the end of this project, we designed and implemented a simple platform to play Tetris game on a FPGA board with VGA output and five switches which were used for five input types. For future work, we realized it is a little hard to play with switches or buttons so a joystick module can also be implemented and used for better user ergonomics. Overall, the project is working as it was planned and as a team we extended our knowledge on controlling systems with micro controllers and gained knowledge on how to work with a VGA. In the future projects we can easily utilize these skills to deliver visual outputs, get complex input modules and create high level algorithm to control our peripherals.

The codes that are used in this project can be found in the following link:

https://www.dropbox.com/sh/ss0d8edzks96o80/AAA68RpyTHzqSq45G9Ua0IFva?dl=0

References

[1] https://embeddedthoughts.com/2016/07/29/driving-a-vga-monitor-using-an-fpga/

[2] https://eewiki.net/pages/viewpage.action?pageId=15925278