

# **ECSE323 Report 5- Group 26**

# **BrickBreaker**

**Authors: Muhammad Ammar Raufi, Murtaza Rizvi**

## Contents

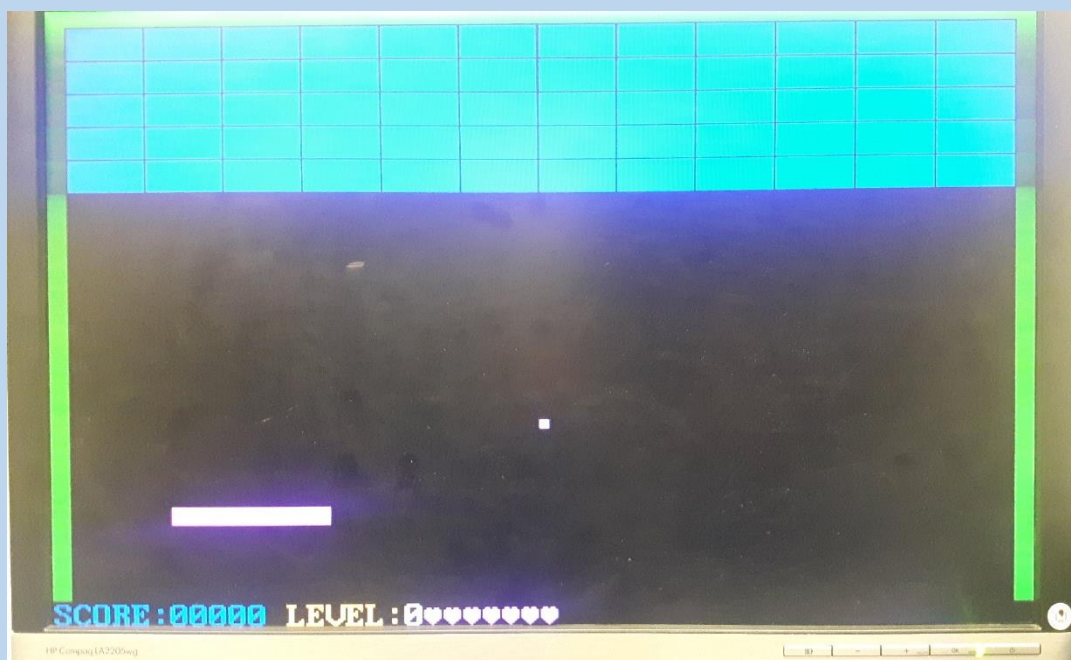
<b>Description of the System's Features .....</b>	<b>3</b>
<b>Block Diagram of the System.....</b>	<b>4</b>
<b>Detailed Description of System.....</b>	<b>5</b>
Inputs .....	5
Counters.....	6
Ball .....	7
Paddle .....	7
Blocks.....	7
Score Bar .....	8
Extra Functionality .....	9
Output .....	10
<b>User Interface .....</b>	<b>11</b>
<b>Testing .....</b>	<b>12</b>
<b>Summary of FPGA Resource Utilization and Timing.....</b>	<b>13</b>
<b>Conclusion .....</b>	<b>15</b>
Problems or Significant Issues .....	15
Possible Enhancements or Extensions .....	15
References .....	16

## Description of the System's Features

Breakout is a popular arcade game developed in 1972 by Atari, Inc. Our version of the game called BrickBreaker starts with 60 bricks at the top of screen and a ball and paddle below it. The concept behind the game is that the moving ball breaks the bricks, and the score depends on how many bricks are broken. The sides of the screen deflect the ball and the paddle at the bottom of the screen is used to deflect the ball and prevent it from falling. The idea of this game was first pitched by Nolan Bushnell and Steve Bristow, and was based off the 1972 Atari arcade game Pong. <sup>[1][2]</sup>

This arcade game can be entertaining and challenging at the same time. It features 8 levels which keeps the players involved all the time. It is suitable for all ages and the colourful display always keeps the players engaged. Multiple people can play in turns to beat each other's high score - making this a fun activity at parties or while hanging out with friends and family.

The game is designed in VHDL and is ported to the Altera DE1 board. A picture of the VGA display is shown in figure 1.



**Figure (1) – Screen at start of game**

## Block Diagram of the System

Figure 2 shows a high level Block Diagram of the Brick Breaker system. It shows the main inputs i.e. clock, rst and paddle. The outputs are R,G,B, HSYNC and VSYNC. Block logic, score logic, ball logic, paddle logic and wall logic drive the R, G and B output to the VGA output. They are explained in more detail in the next sections.

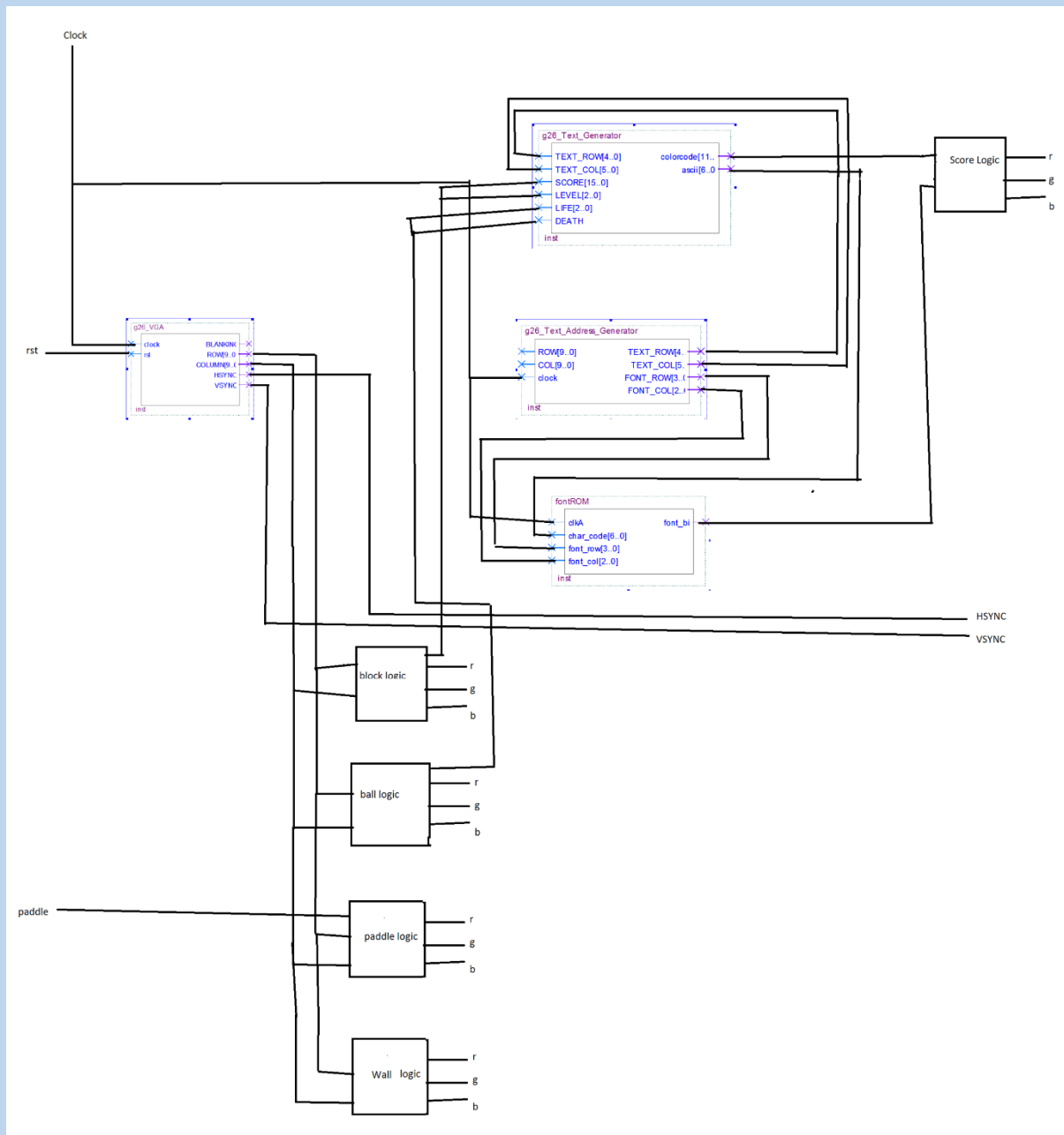


Figure (2) – High Level Block Diagram

## Detailed Description of System

### Inputs

Figure 3 shows the three external inputs to the system. The paddle, which is controlled by two buttons on the Altera board, a 50 MHz clock and a reset button. The input\_main was used for testing various functionality e.g. score, life and also to pause the game.

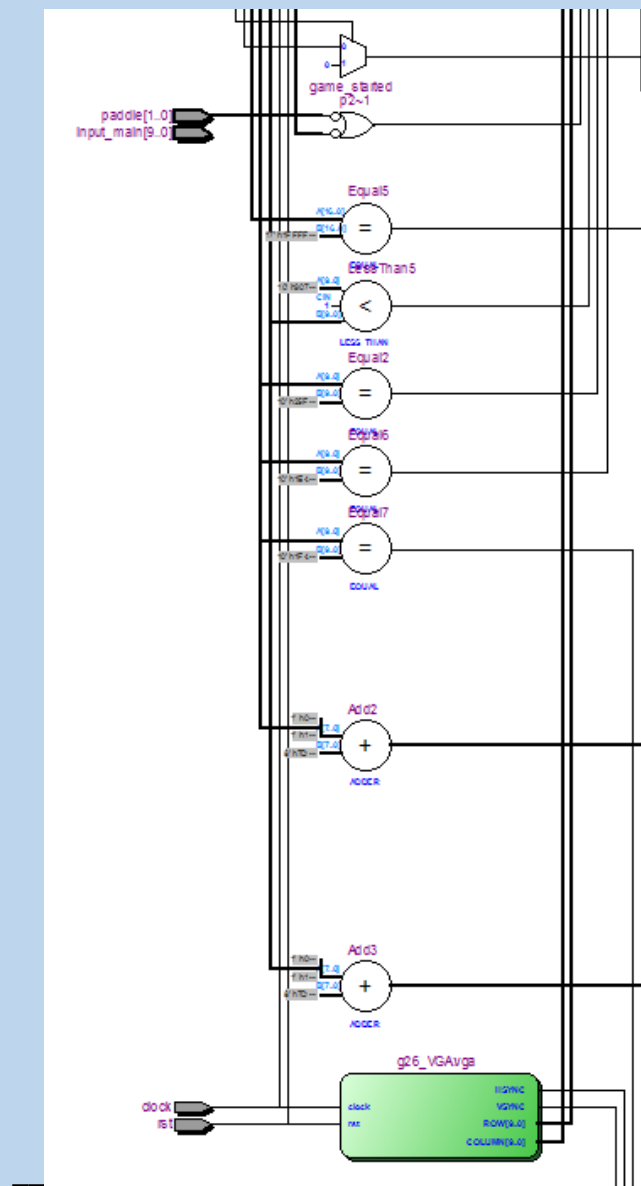


Figure (3) – External Inputs

## Counters

There are three counters in the system.

Counter\_enabler: A counter to slow down the two other counters. You can see the inputs and outputs in figure 4 (a)

Ball\_row\_counter: A counter to control the ball row position. You can see the inputs and outputs in figure 4 (b)

Ball\_col\_counter: A counter to control the ball column position. You can see the inputs and outputs in figure 4 (c)

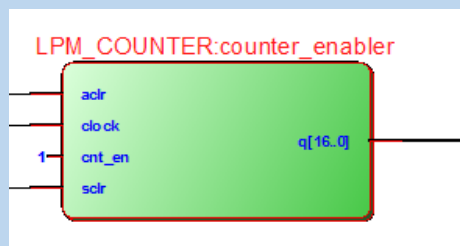


Figure 4 (a) – Counter Enabler

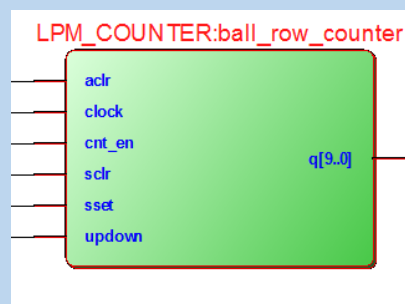


Figure 4 (b) – Ball Row Counter

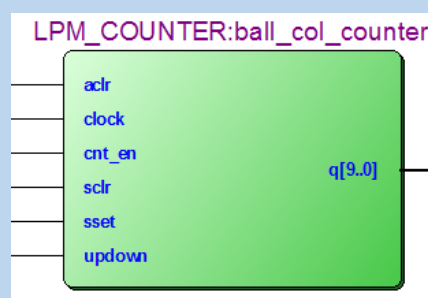


Figure 4 (c) – Ball Column Counter

The counter\_enabler counter is incremented on every clock cycle and reset at the binary value "1111111111111111". The rest of the counters are incremented when counter\_enabler reaches the binary value "1111111111111110". The output of ball\_row\_counter and ball\_col\_counter are set to "400" when their respective sset values are high. Updown of the ball\_row\_counter and ball\_col\_counter counters are used to control the direction of the ball. The counter\_enabler is also used to control the movement of the paddle whose position is only incremented or decremented when the output of the counter reaches the value "1111111111111110".

## Ball

The output of ball\_row\_counter and ball\_col\_counter are driving the ball row and column, and the output to the vga is white if the col and row output from the g26\_VGA component are less than eight more from the output of ball\_row\_counter and ball\_col\_counter.

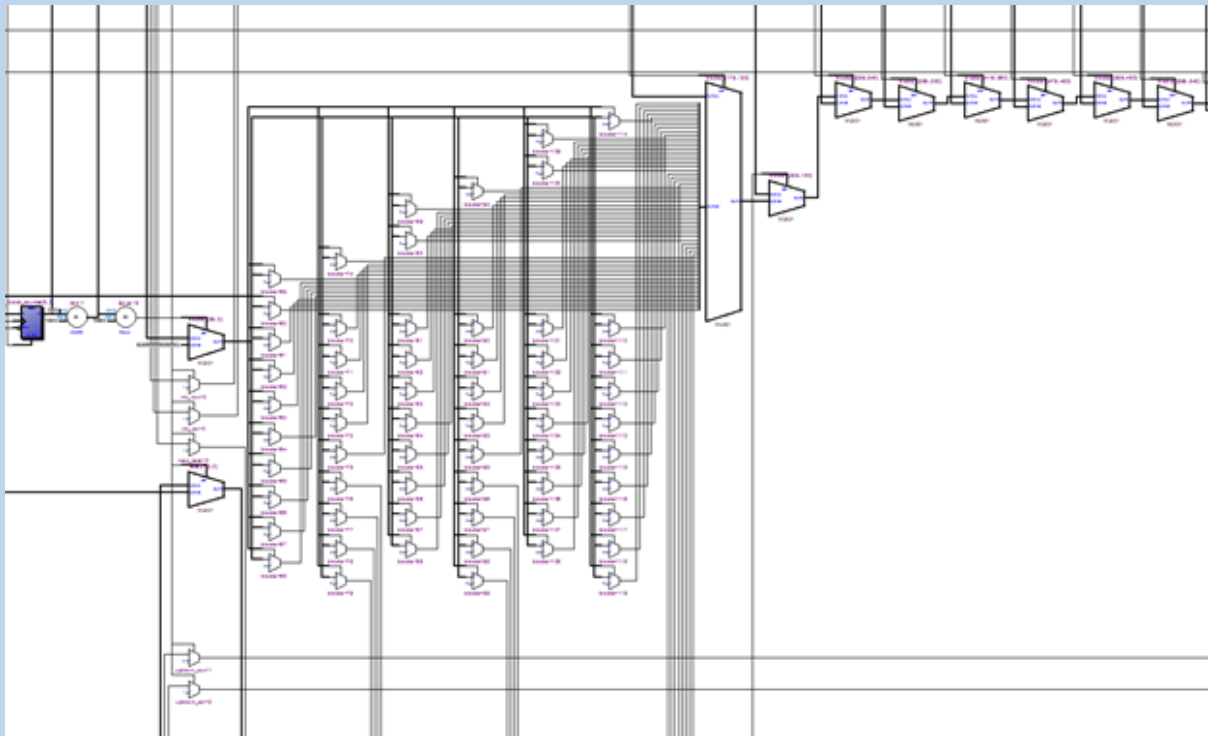
The direction of the ball is controlled by the updown property of the counters. If the ball hits a wall i.e. if ball\_row\_counter output is 16 or ball\_col\_counter output is either 16 or 784, then the respective counter is reversed depending on the wall that is hit. If the ball hits the paddle, its row\_counter is reversed. If the ball hits a block, its row\_counter is reversed if it hits a horizontal edge of the block, and col\_counter is reversed if it hits a vertical edge of the block.

## Paddle

The movement of the paddle is controlled by the 2 bit input logic vector paddle. If the first bit is low, the paddle moves to the left, if the second bit is low the paddle moves to the right. If they are both high, the block is stationary. The logic vector is mapped to two push buttons on the Altera board which go low when you press them and turn high when they are released. The top left position of the paddle is controlled in this way and when a button is pressed the counter is incremented and decremented accordingly and the output to the vga is green for columns 128 more than the top left value and row 16 more than the top left value.

## Blocks

The Blocks are controlled by a 60 bit std\_logic\_vector, where each bit represents a block and is high when the block is present and is low when the block has been hit. Each block is 32 bit by 64 bit so the output of the vga is blue from pixel (16,16) to (784,176), with a 1 pixel border at  $((\text{pixel}-16) \bmod 32)$  and  $((\text{pixel}-16) \bmod 64)$ . If the ball hits anywhere on this border and the block is active, the ball direction is reversed and the block is deactivated by turning that bit of the logic\_vector to low. The block number to deactivate is calculated by the formula  $((((\text{row}-16)/32))*12)+(((\text{column}-6)/64)+1))$ . The block logic, implemented through MUXs can be seen in figure 5

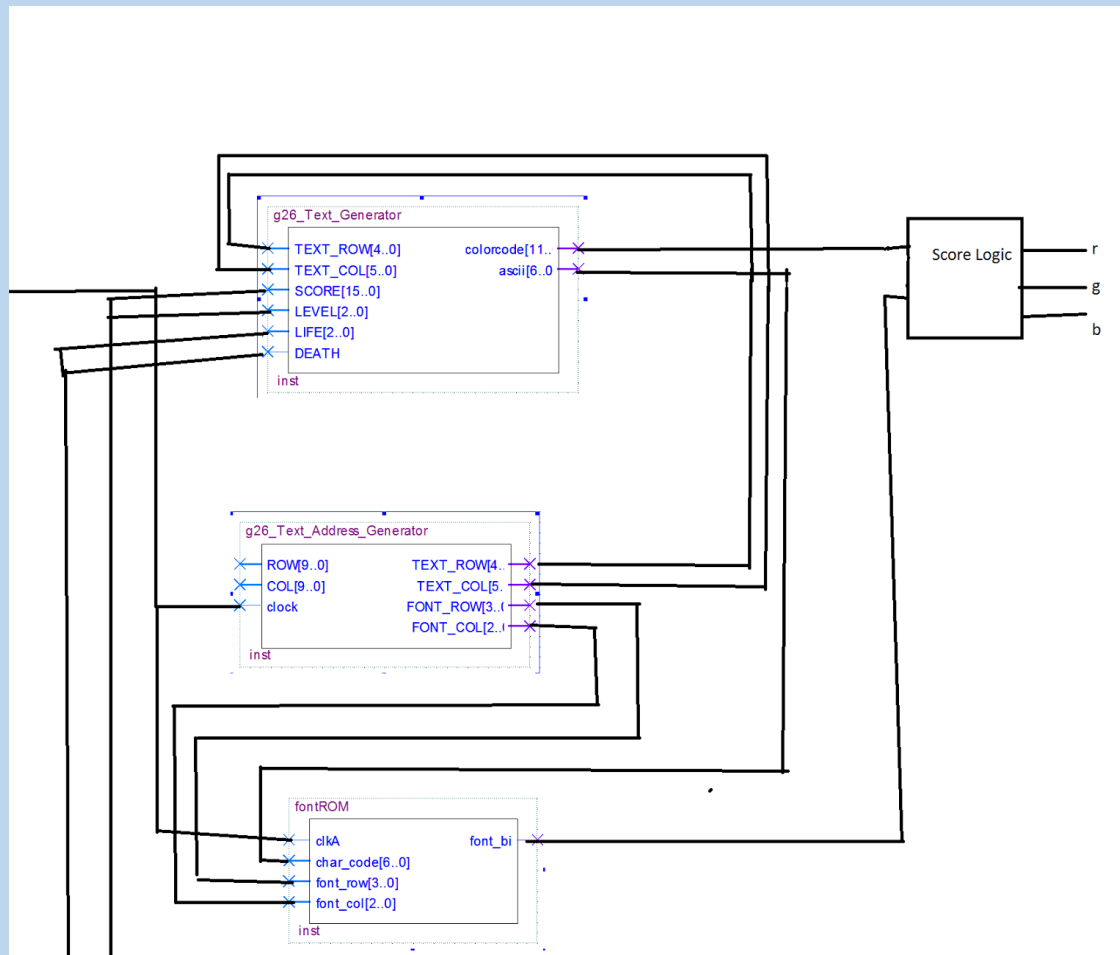


**Figure 5 – Block Logic Implementation**

## Score Bar

The score bar is driven by the text generator, fontROM and text\_address generator components. An overview can be seen in figure 6. The text\_address\_generator component gets the row and col from the VGA component and gives the output text\_row and Text\_col to Text\_generator and font\_row and font\_col to the fontROM component. The Text\_generator gets the score, life and level and outputs the colorcode and ascii code, and the fontROM component gives the font\_bit output which indicates when the output RGB values should be enabled to show an ascii character.





**Figure 6 – Scroll Bar Implementation**

There are three main parts to the score bar. the score, the life and the level. Whenever the ball hits a block, the score value is incremented by 10. Whenever the ball hits the bottom edge of the playing screen i.e. row is 584, the life is decremented by one and the ball is reset to a center position. The ball counter stops counting until a paddle button is pressed on the Altera board. When all the blocks have been hit, the level is incremented by 1 and all the blocks are reset.

## Extra Functionality

We also added the functionality that displayed the text “You are dead” whenever the user loses a life. The text continues to show until the user moves their paddle to start playing again. This is achieved by setting a signal ‘death’ high when the ball falls to the pixel 568 and only goes low when the user moves their paddle. The Text\_generator file takes this “death” signal as input and while it is high it shows the text “You are dead” when the Text\_row is equal to 9.

## Output

The five outputs which drive the vga display are the R, G, B, VSYNC and HSYNC signal. The R, G and B signals are stored in a register before they are output to ensure that no timing problems occur. HSYNC and VSYNC come from the g26\_VGA component. The outputs can be seen in figure 7.

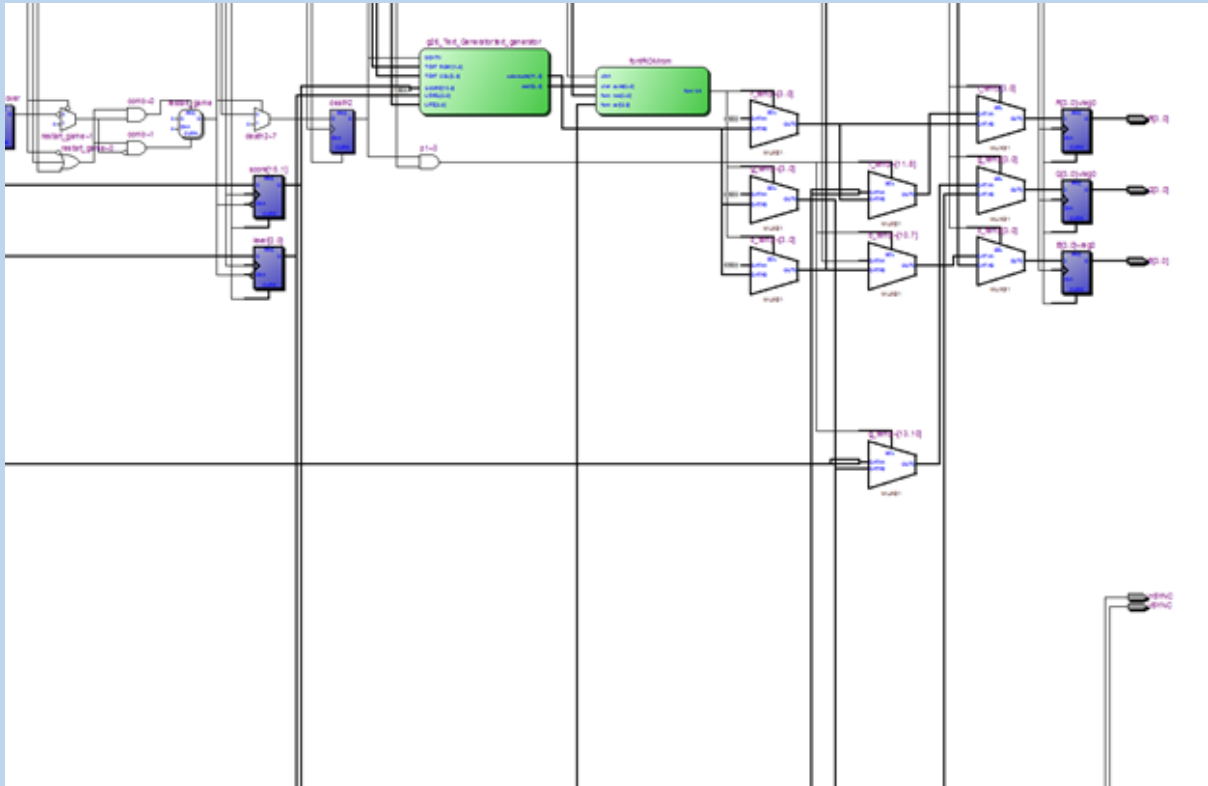


Figure 7 - Output

## User Interface

Our game starts with 60 cyan bricks on the top of screen – split into 5 rows of 12 bricks in each. The objective of the game is to eliminate as many bricks as possible without loosing the ball. The green side bars of the screen and the pink paddle at the bottom deflect the ball and change the direction of its motion in multiples of 45 degrees. If the ball hits a brick, the brick breaks and the player gets 10 points. The paddle at the bottom can be moved right and left to catch the ball after rebound. The paddle is moved right by pressing key0 on the Altera board and left by pressing key1 on the board.

The bar at the bottom of the screen displays the Score, Level and Life. Each ♥ represents a life. If the ball misses the paddle, it is lost and a life is decreased. A 'YOU ARE DEAD' message is also displayed on the screen. If the user still has remaining lives, he or she can continue the game from the same stage by moving the paddle. 7 lives are given to the player at the beginning of the game. Figure 8 displays game play just after a life is lost. The player now has 2 remaining lives, they are on Level 0 and their score is 160.

Once all the bricks are eliminated from the screen, the player moves on to the next level. The game ends once the player eliminates all bricks till level 7, or once he or she is out of lives.

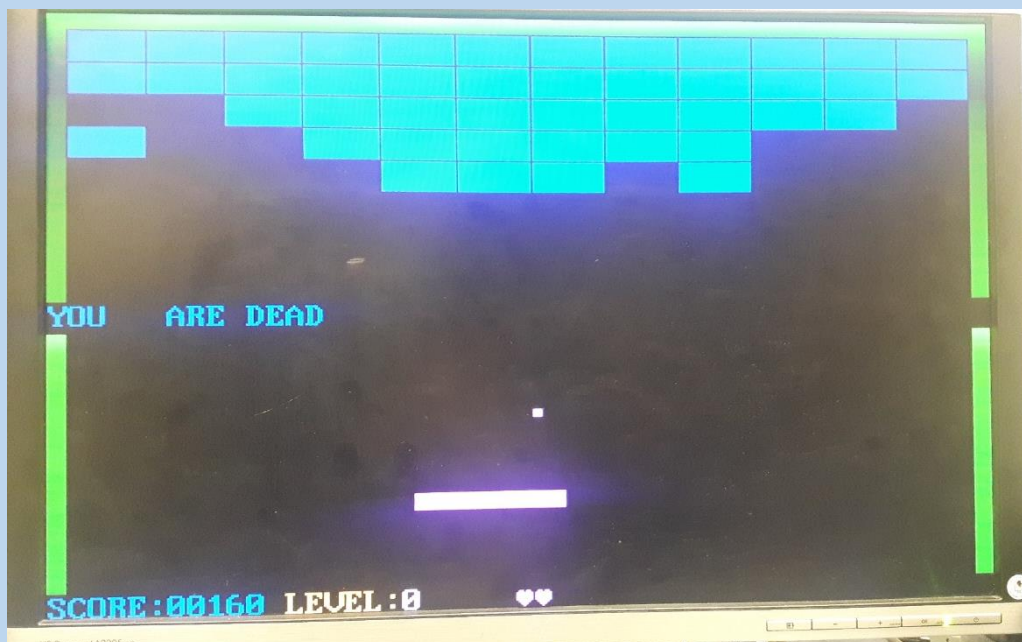


Figure 8 – Game Interface

## Testing

To test the system, we had to run multiple ModelSim simulations to get a better understanding of how all the signals were behaving. The biggest issue we had was implementing the ball so we had to run simulations to check how the signals driving the ball were behaving.

We also ran a simulation to check if all the blocks would be disabled, when the ball was in its correct position.

A screenshot of the simulation wave for approximately one VSYNC cycle is shown below in figure 9.

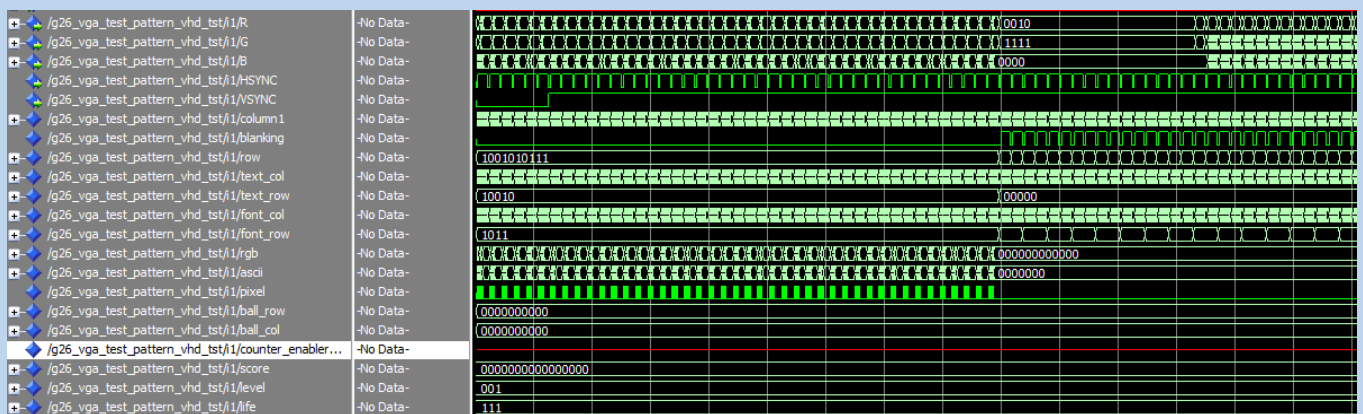


Figure 9 – ModelSim Simulation

After we had completed all aspects of the game, we played the game through to the end to make sure that everything was working correctly.

## Summary of FPGA Resource Utilization and Timing

A summary of the FPGA resource utilization can be seen in figure 10

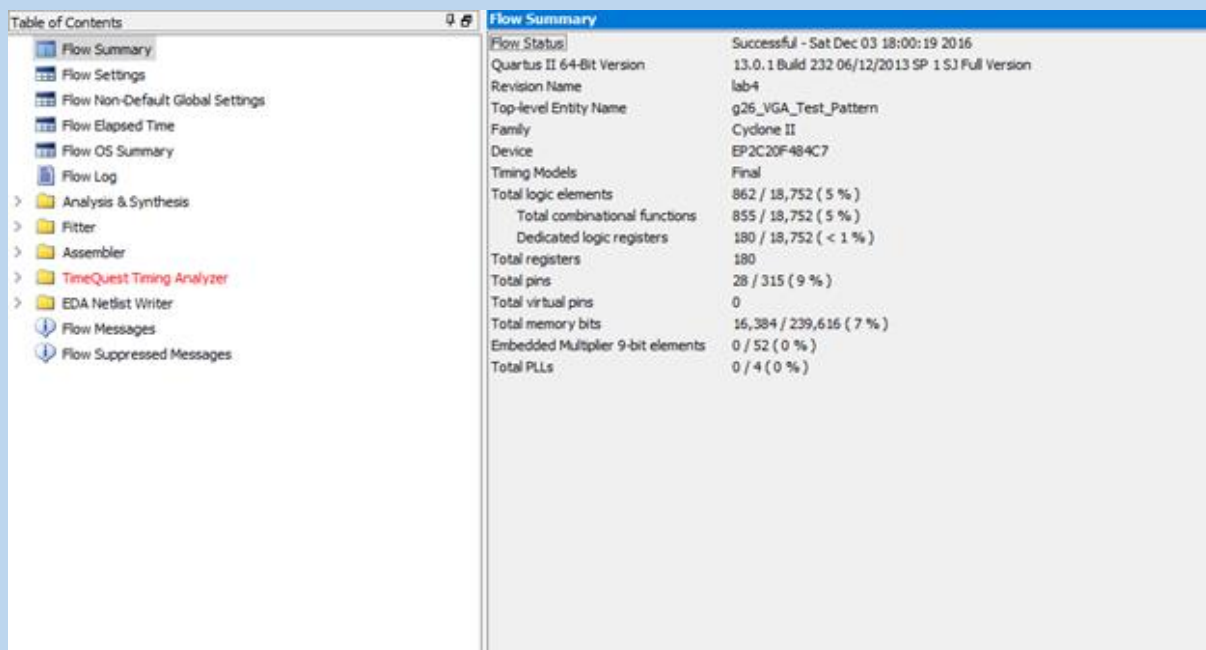
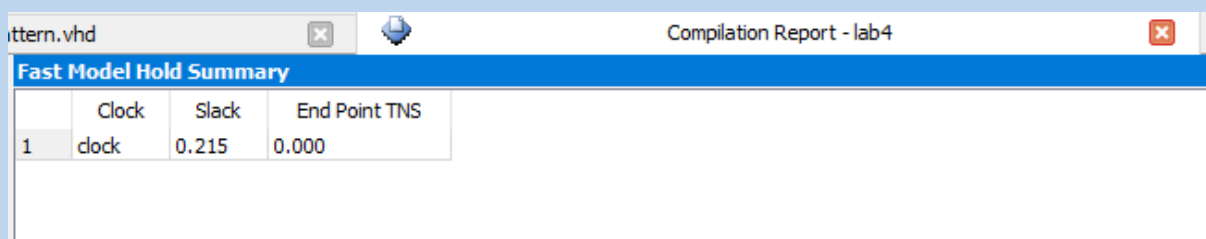


Table of Contents		Flow Summary	
Flow Summary		Flow Status	Successful - Sat Dec 03 18:00:19 2016
Flow Settings		Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Full Version
Flow Non-Default Global Settings		Revision Name	lab4
Flow Elapsed Time		Top-level Entity Name	g26_VGA_Test_Pattern
Flow OS Summary		Family	Cyclone II
Flow Log		Device	EP2C20F484C7
Analysis & Synthesis		Timing Models	Final
Fitter		Total logic elements	862 / 18,752 ( 5 % )
Assembler		Total combinational functions	855 / 18,752 ( 5 % )
TimeQuest Timing Analyzer		Dedicated logic registers	180 / 18,752 ( < 1 % )
EDA Netlist Writer		Total registers	180
Flow Messages		Total pins	28 / 315 ( 9 % )
Flow Suppressed Messages		Total virtual pins	0
		Total memory bits	16,384 / 239,616 ( 7 % )
		Embedded Multiplier 9-bit elements	0 / 52 ( 0 % )
		Total PLLs	0 / 4 ( 0 % )

Figure 10 Resource Utilization

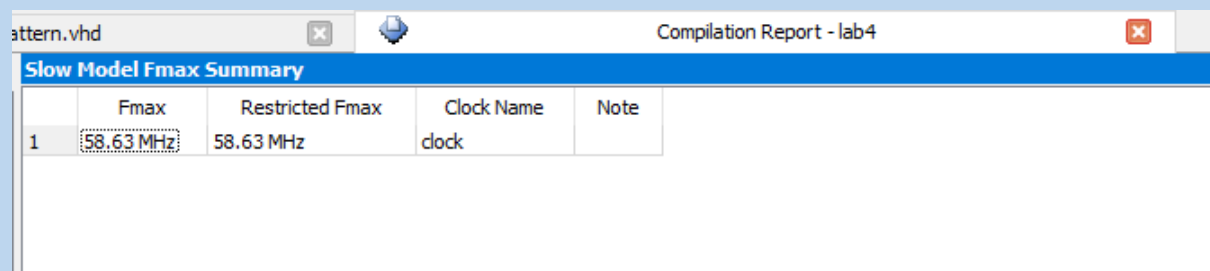
The timing analysis diagrams shows that Fmax is larger than 50MHz for the slow model but the slack value for Slow Model is negative. However, no significantly visible lag was seen in our game display.

The timing summary can be seen below in figure 11 (a), (b) and (c)



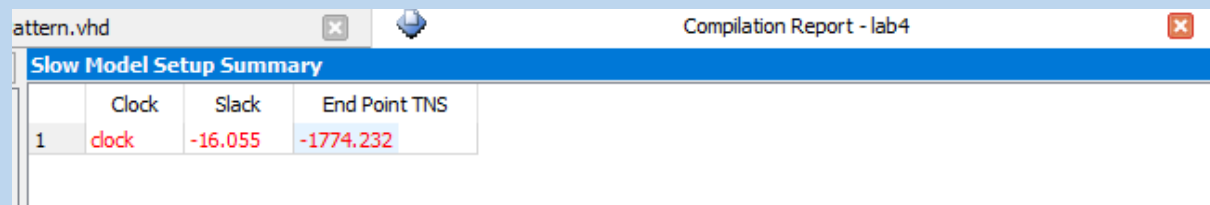
Fast Model Hold Summary			
	Clock	Slack	End Point TNS
1	clock	0.215	0.000

Figure 11 (a) – Fast Model Hold Summary



	Fmax	Restricted Fmax	Clock Name	Note
1	58.63 MHz	58.63 MHz	clock	

Figure 11 (b) – Slow Model Hold Summary



	Clock	Slack	End Point TNS
1	clock	-16.055	-1774.232

Figure 11 (c) – Slow Model Setup Summary

# Conclusion

## Problems or Significant Issues

The biggest issue we had was in driving the ball. Initially we had difficulty realizing that the counter had to be controlled to be able to see the ball and then we had difficulty in initializing the ball in a center position, so that the counters would start the ball at a value of zero, whereas that was the value of the wall, which reversed the direction of the counter so the ball would never be visible on the screen. After we realized the use of the LPM\_SVALUE property of the counter this problem was fixed.

We also had some timing problems. We had fixed the raggedness issue of the score bar in lab four, but this problem returned after we had implemented much of the game. The blocks were also leaving behind some of their color after they had been disabled. We realised that this must be because we are not setting the outputs of the r,g,b output values on the same rising edge so we created temporary values for the three of these and added a register which allowed all three to be added on the same clock edge. Sure enough, this resolved problem.

## Possible Enhancements or Extensions

A possible enhancement that can be made to our game is to increase the difficulty as the Level goes up. This can be achieved by adding several extensions. A great idea would be to increase the speed of the ball and decrease the speed of the paddle as the level got higher. The size of the paddle could also be decreased for increases difficulty. Furthermore, distance between the brick layer and the paddle could also decrease for additional difficulty and excitement. To make the game more engaging, sounds effects can also be added. A sound would be associated to each time a brick is broken, a life is lost or the player goes to a higher level.

Some other possible enhancements are to have different number of points associated with each layer of bricks. The layer with higher points would need more than one hit before the brick breaks. Each layer could also have a different colour to indicate the points associated with that layer.

We also thought about adding a multiplayer option which would automatically keep track of each player's score and make the game more fun and competitive. There could also be an option to have multiple balls on the screen at the same time. This could be implemented with having either one or two paddles as well.

## References

- [1] <http://www.atarigames.com/atarinumbers90s.pdf>
- [2] [https://en.wikipedia.org/wiki/Breakout\\_\(video\\_game\)#cite\\_note-2](https://en.wikipedia.org/wiki/Breakout_(video_game)#cite_note-2)