Group 30
Gabriel Chootong 260 637 105
Sean Stappas 260 630 512

December 5, 2016
ECSE-323

Crystal Ball

Table of Contents

1 Game Description

1.1 Advertising

Crystal Ball is a challenging game where the player needs to be fast and clever to win. The game is a version of Breakout, and involves a ball that breaks blocks. The goal is to successfully make the ball break all blocks in order proceed to the next level. The player controls a paddle that allows him or her to keep the ball in play, and depending on their relative positions, the ball might have unexpected changes of direction. But that is not all! There are several powerups available to the player that can be real game-changers. These can help clear a level but can also greatly increase the difficulty! In addition to that, each level differs in difficulty; the higher the level, the faster the speed of the ball!

1.2 User's Guide

1.2.1 Controls

Figure 1 shows the basic controls of the game on the DE1 board: 4 pushbuttons (KEY3 to KEY0, from left to right). KEY3 moves the paddle left, and KEY2 moves the paddle right. KEY1 launches the ball at start-up, and KEY0 resets the game (LEVEL reset to 1, SCORE to 0, LIFE to 5).

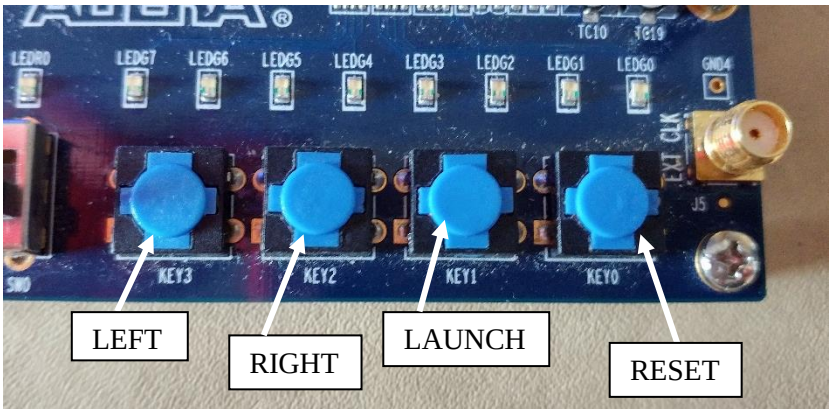

Figure 1. Controls.

1.2.2 Colours

Each level has its own colour gradient (various shades of a colour). Figure 2 shows the colour gradient for each level of the game. For example, every column of blocks in level 1 is a different shade of blue.

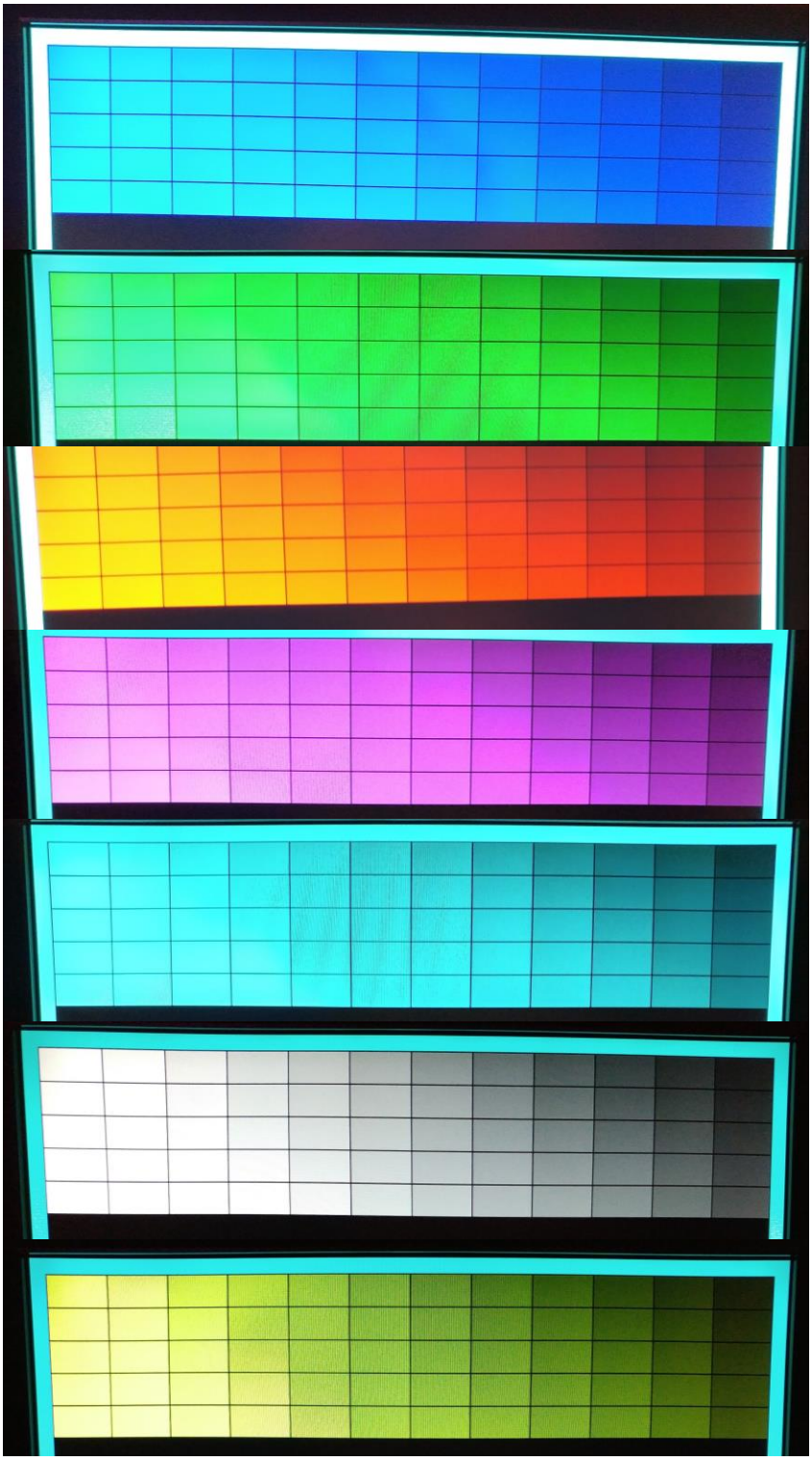

Figure 2. Level Colour Gradients (Top: Level 1, Bottom: Level 7).

1.2.3 Powerups

Powerups fall periodically from broken blocks and must be caught by the user with the paddle. A description of the powerups can be found in Table 1. Even though some powerups have negative effects, those powerups award more points than normal. Similarly, powerups that are already extremely useful, such as the Penetrator, offer a small amount of points. The effects of the Shrinker, Penetrator and Flipper powerups are lost upon losing a life. The Penetrator powerup also applies a trailing rainbow effect to the ball, as can be seen in Figure 3, and in the following video: https://youtu.be/KxH68TEDN7Y. Note that the gradient level colours were not implemented at the time of filming of the video.

Table 1. Powerup Descriptions.

Powerup	Name	Effect	Points Awarded
	1-UP	Extra life (maximum of 7 lives)	10
	Shrinker	Halve paddle size	50
	Penetrator	Ball penetrates blocks (also applies a rainbow effect)	5
	Flipper	Left and right paddle controls flipped	25

Figure 3. Penetrator Rainbow Effect.

Points are awarded in the game from breaking blocks and from obtaining powerups. Bonus points are also awarded if the player hits multiple blocks before hitting the paddle again (1 point for the first extra block hit, 2 points for the second block hit, and so on). This rewards the player for making interesting plays, where the ball can end up hitting multiple blocks in a row. Note that this mechanic allows the player to receive a large amount of points as long as he or she keeps the Penetrator powerup.

1.2.4 Ball & Paddle Mechanics

Another interesting mechanic the game provides is that the ball will bounce off the paddle at different angles depending on where it hit the paddle. This is illustrated in Figure 4. If the ball hits the paddle to the left of the paddle's midpoint, it will reflect to the left. If it hits the paddle on the right, it will bounce to the right. Furthermore, if the ball hits the midpoint of the left half-paddle, it will reflect at exactly 45 degrees to the left. Similarly, if it hits the midpoint of the right half-paddle, it will reflect at 45 degrees to the right. In between these points, the ball will reflect approximately as shown in Figure 4. Note that to achieve this result, the ball changes speed, going faster in whatever direction appropriate.

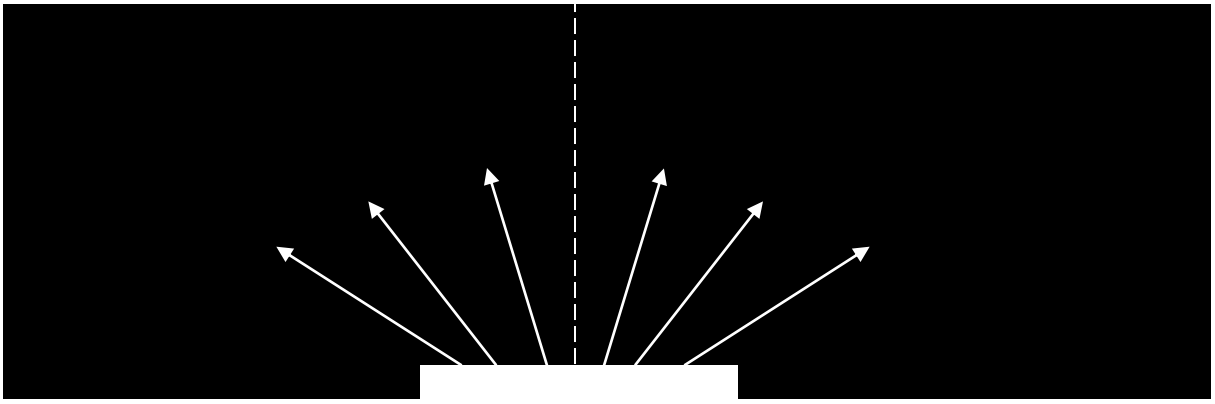

Figure 4. Ball Bounce Direction Dependence on Location on Paddle.

One last feature of the game is that the speed of the ball increases with every level, making the game slightly harder as it progresses, so that only the best of the best can reach and beat level 7. The approximate ball speeds at every level are shown in Table 1.

Table 2. Ball Speeds at Every Level.

Level	Ball Speed (pixels/second)
1	381
2	394
3	407
4	421
5	436
6	452
7	470

2 System Description

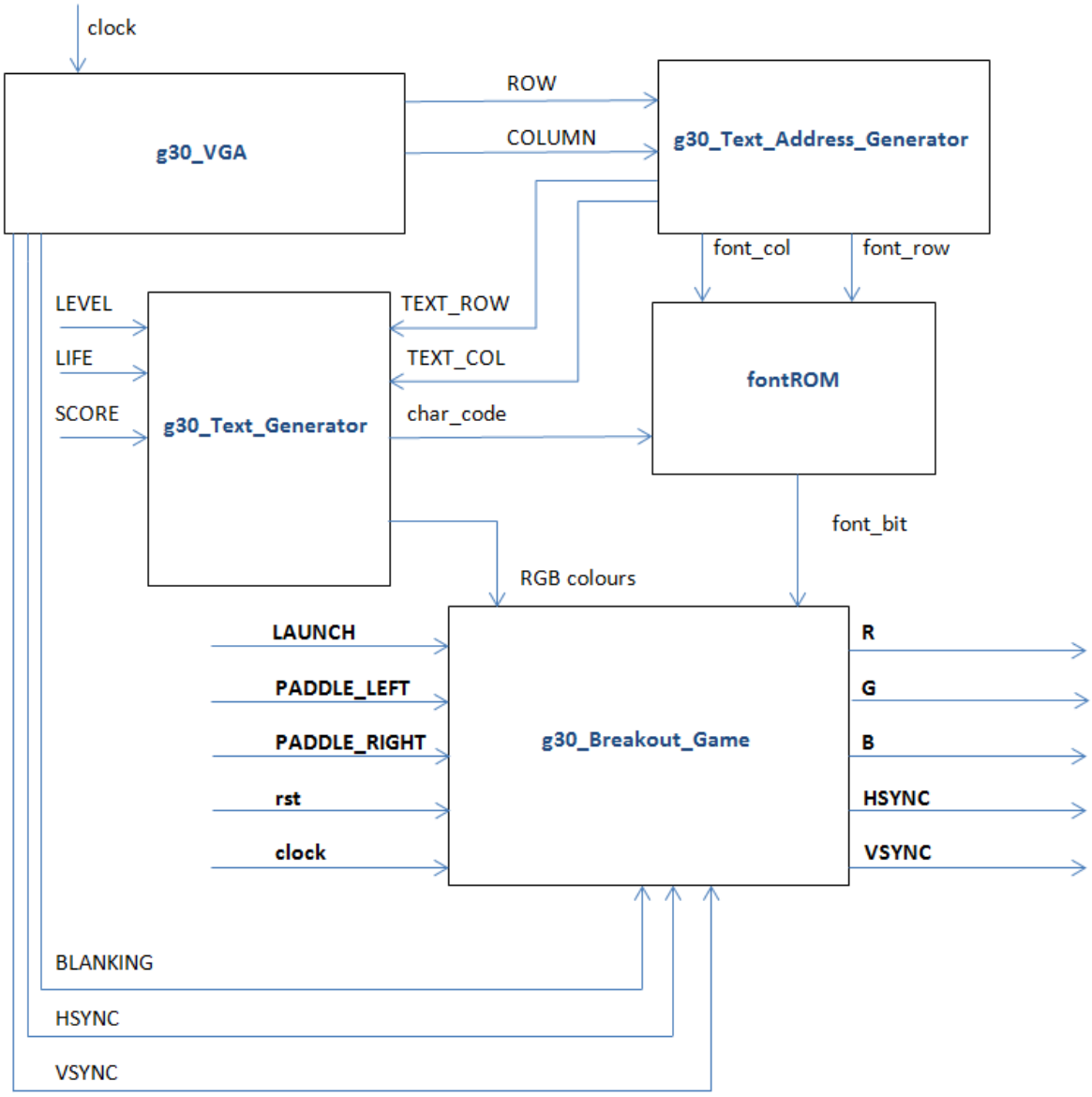

Figure 5. Block Diagram of the System.

Figure 5 shows a simplified, high-level block diagram of our system, and will be used as reference in this section. The input and output signals of g30_Breakout_Game are highlighted in bold in the diagram, and the rest of the signals are internal to the circuit. For a more complete system description, see the attached RTL diagram, and the g30_Breakout_Game VHDL file. The RTL diagram was attached separately (g30_Breakout_Game.png), since it was too large to present in this report directly. Some internal signals from the g30_Breakout_Game VHDL file will also be referenced in this section.

2.1 VGA Circuit

The g30_VGA circuit uses clock counters to generate signals for row and column pixel position (‘ROW’ and ‘COLUMN’). The row and column pixel position ranges are 600 and 800, respectively. It also generates the ‘BLANKING’, ‘HSYNC’ and ‘VSYNC’ signals which correspond to blanking, horizontal synchronization and vertical synchronization signals, respectively. The blanking signal is set to ‘0’ when the row and column position are outside the visible pixel range. ‘HSYNC’ and ‘VSYNC’ indicate the start of a horizontal line and of a video frame, respectively.

2.2 Text Address Generator

The g30_Text_Address_Generator circuit takes ‘ROW’ and ‘COLUMN’ as inputs and determines the location of a character to be drawn. It does so by dividing the 800 by 600-pixel grid into 19 rows of 32 pixel rows each and 50 columns of 16 pixel columns each. The 19th row is not fully displayed since 600 is not a multiple of 19. This new mapping of row and column position is represented by the ‘TEXT_ROW’ and ‘TEXT_COL’ signals as outputs of the g30_Text_Address_Generator. The component also outputs signals ‘font_row’ and ‘font_col’ which carry the positions of individual pixels within a character.

2.3 Text Generator

The g30_Text_Generator component takes the ‘TEXT_ROW’ and ‘TEXT_COLUMN’ signals as inputs, as shown in Figure 5. The other inputs for this component are the ‘LIFE’, ‘SCORE’ and ‘LEVEL’ signals, which carry the current values for the life, score and level. This component produces the last text row of the screen, which is where the life, the score and the level are displayed. It outputs the red, green and blue colour signals, as well as the ‘char_code’ signal, which carries the ASCII code of the current character being drawn. The text generator produces the ASCII code and the colours of the characters to be displayed on the 19th text row (TEXT_ROW = 18, since the signal ranges from 0 to 18) and the blank character ASCII code “20h” for all other character locations on the screen.

2.4 Font ROM

The fontROM component then takes the ASCII code from the g30_Text_Generator as input, along with the ‘font_row’ and ‘font_column’ signals from the g30_Text_Address_Generator as illustrated in Figure 5. This component is driven by the same clock as the g30_VGA component. The fontROM component outputs the ‘font_bit’ signal, which is set to ‘1’ if the current pixel at the positions determined by ‘font_row’ and ‘font_col’ must be on (according to the current character being drawn as specified by the input ASCII character code). The ‘font_bit’ signal is set to ‘0’ otherwise.

2.5 Breakout Game

2.5.1 Inputs and Outputs

The g30_Breakout_Game circuit is driven by the ‘clock’ signal. Its other inputs are the reset, launch, right paddle and left paddle buttons, which correspond to the signals ‘rst’, ‘LAUNCH’, ‘PADDLE_RIGHT’ and ‘PADDLE_LEFT’, respectively. The outputs are colour signals ‘R’, ‘G’ and ‘B’ and ‘HSYNC’ and ‘VSYNC. Both ‘HSYNC’ and ‘VSYNC’ are passed down from the g30_VGA component. The colour signals, however, depend on what is happening on the game.

2.5.2 RGB Signals

The g30_Breakout_Game circuit uses several signals from the other components in its logic to determine what colour signals to output. It is important to note that the signal colours are carried into a single 12-bit internal signal ‘RGB’ for ease of manipulation. Bits 11 to 8 of ‘RGB’ correspond to red, 7 to 4 to green and 3 to 0 to blue. A pixel should not be displayed when ‘BLANKING’ from g30_VGA is set to 0. If ‘font_bit’ from fontROM is 0, then the circuit is not currently drawing a pixel for a character on the 19th row (in other words, if ‘TEXT_ROW’ is 18) and should therefore output a black pixel. If ‘ROW’ and ‘COLUMN’ indicate edge positions, then ‘RGB’ is set to have the value of a light blue colour to represent the walls of the game.

2.5.3 Paddle and Ball Counters

The circuit also uses several counters to keep track of the parameters of the paddle and the ball. Cascaded counters are used to slow down the paddle and the ball’s movements. This is done so that the ball and paddle positions do not increment at the speed of the 50 MHz clock, which would be too fast. The cascaded counter is referred as a “slow counter” in the code. The actual position counter of the paddle or ball increments every time the slow counter reaches a chosen value, and it will be slower the higher value is. In other words, the clock of the actual count is determined by the slow counter. In this manner, the speed of the ball and the paddle movement can be controlled.

For example, the row and column position of the ball are tracked by the internal signals ‘ball_row_value’ and ‘ball_col_value’ which are updated by ‘slow_ball_row_counter’ and ‘slow_ball_col_counter’. The paddle speed is controlled in a similar manner by the signals ‘slow_paddle_col_counter’ and ‘paddle_col_value’. Moreover, the speed of the ball is increased in each higher level by modifying the corresponding slow counters. It is important to note that ‘slow_paddle_col_counter’ is only enabled if either ‘PADDLE_LEFT’ or ‘PADDLE_RIGHT’ is set to 1, but not both. This is to ensure the user is either holding down the button for ‘PADDLE_LEFT’ or

‘PADDLE_RIGHT’, to move the paddle left or right. This condition is implemented with an XOR operator, which outputs 1 when only one input is 1. The result of the XOR is then connected to an AND gate along with ‘paddle_inside_borders’ which is set to 1 when the user is not attempting to move the paddle past the left or right wall.

2.5.4 Paddle and Ball Display

Using the signals that track the position of the ball and paddle (‘ball_col_value’, ‘ball_row_value’ and ‘paddle_col_value’), the g30_Breakout_Game circuit determines what to draw depending on the current values for ‘ROW’ and ‘COLUMN’. For instance, if the ‘ROW’ and ‘COLUMN’ values overlap with ‘ball_row_value’ and ‘ball_col_value’ within an 8-pixel margin (the ball’s dimensions are 8x8 pixels), the circuit will output colour signals corresponding to white (which is the colour of the ball). The paddle is drawn in a similar manner and its dimensions are 128 by 16 pixels (64 by 16 if the Shrinker powerup is obtained).

2.5.5 Block Display

The g30_Blockout_Game also displays 60 blocks that the player must break in order to proceed to the next level. Each block is represented by a unique bit in the 60-bit signal ‘blocks’, and a 1 at the n^{th} bit signifies that the n^{th} block is intact. If the n^{th} bit is 0, the n^{th} block is broken. If ‘COLUMN’ and ‘ROW’ correspond to a pixel position inside the area of the blocks, the circuit will draw the block at the current position if the bit with index that is associated to that position in the ‘blocks’ signal is set to 1. The circuit also separates blocks by drawing blank pixels where ‘ROW’ and ‘COLUMN’ are not multiples of 32 and 64, respectively. In order to determine this, the circuit uses the signal ‘block_count’ which determines the index of the block at the current position using offset and modulus operations as shown in the code. Therefore, ‘block_count’ carries the value of the index of the current block.

2.5.6 Collision Detection

For hardware efficiency, a collision is checked only when the ball changes position since the clock’s speed is much faster than the ball. When the ball is determined to be inside the area of the blocks, the g30_Breakout_Game will determine if there is an intact block at the ball’s position (using the signals ‘blocks’ and ‘block_ball_count’) as shown in the code. In the case that there is an intact block at the ball’s position, the ball’s direction is changed depending on where there is a collision. If the ball hits either the right or left side of a block, then only its horizontal direction is changed by changing the value of ‘ball_col_increment’. If the ball hits the top or the bottom side of a block, then the modified signal is

'ball_row_increment'. Moreover, the index bit in the 'blocks' signal corresponding to the block involved in the collision is then set to 0 to indicate that it has been destroyed.

2.5.7 Reset Mechanism

The g30_Breakout_Game circuit also implements a reset mechanism that can be triggered by a button on the board (with the use of a pin assignment) or by losing or winning the game. The score, level and life values are restored when either of these events occur.

2.5.8 Powerups

The system also constantly provides game-changing powerups to the player. Each time 7 blocks have been broken (while there are no other powerups on the screen), the circuit will display a falling block that, if the player chooses to catch, will enable a powerup. For instance, the Penetration power-up lets the ball break through blocks without bouncing off them, awarding the player a large amount of points, as explained previously. The Penetration power-up is implemented in a simple manner: if the signal 'ball_penetrate' is set to 1, the circuit will not change the directions of the ball when it hits a block. Moreover, the ball will leave a colourful trail in its path when this power-up is in enabled.

3. Testing and Development

3.1 Manual Testing

3.1.1 Ball Testing

First, the ball bouncing was tested, using the pin assignment in Figure 6, with the reset input mapped to one of the pushbutton switches (PIN_R22). As described in the lab description, the reset places the ball in the lower center of the screen (coordinates ROW=300, COL=400) and makes it move towards the left and top of the screen. The results can be seen here: https://youtu.be/eeVzYnx50Qs. Note that an invisible bottom wall was implemented purely for testing the bouncing of the ball.

3.1.2 Blocks Testing

Next the blocks and block breaking were tested, with the same pin assignments. Now, the reset also refreshes all the blocks on the screen. The results can be seen here: https://youtu.be/_umMgFNsil4.

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Differential Pair
B[3]	Output	PIN_B10	3	B3_N0	PIN_H11	3.3-V LV...default)		24mA (default)	
B[2]	Output	PIN_A10	3	B3_N0	PIN_B9	3.3-V LV...default)		24mA (default)	
B[1]	Output	PIN_D11	3	B3_N0	PIN_B14	3.3-V LV...default)		24mA (default)	
B[0]	Output	PIN_A9	3	B3_N0	PIN_B13	3.3-V LV...default)		24mA (default)	
clock	Input	PIN_L1	2	B2_N1	PIN_M1	3.3-V LV...default)		24mA (default)	
G[3]	Output	PIN_A8	3	B3_N0	PIN_A13	3.3-V LV...default)		24mA (default)	
G[2]	Output	PIN_B9	3	B3_N0	PIN_G11	3.3-V LV...default)		24mA (default)	
G[1]	Output	PIN_C10	3	B3_N0	PIN_F10	3.3-V LV...default)		24mA (default)	
G[0]	Output	PIN_B8	3	B3_N0	PIN_C10	3.3-V LV...default)		24mA (default)	
HSYNC	Output	PIN_A11	3	B3_N0	PIN_F11	3.3-V LV...default)		24mA (default)	
LEVEL[2]	Input	PIN_M22	6	B6_N0	PIN_H10	3.3-V LV...default)		24mA (default)	
LEVEL[1]	Input	PIN_L21	5	B5_N1	PIN_W11	3.3-V LV...default)		24mA (default)	
LEVEL[0]	Input	PIN_L22	5	B5_N1	PIN_A11	3.3-V LV...default)		24mA (default)	
LIFE[2]	Input	PIN_U12	8	B8_N0	PIN_B10	3.3-V LV...default)		24mA (default)	
LIFE[1]	Input	PIN_W12	7	B7_N1	PIN_A10	3.3-V LV...default)		24mA (default)	
LIFE[0]	Input	PIN_V12	7	B7_N1	PIN_H9	3.3-V LV...default)		24mA (default)	
R[3]	Output	PIN_B7	3	B3_N1	PIN_D11	3.3-V LV...default)		24mA (default)	
R[2]	Output	PIN_A7	3	B3_N1	PIN_A8	3.3-V LV...default)		24mA (default)	
R[1]	Output	PIN_C9	3	B3_N1	PIN_E11	3.3-V LV...default)		24mA (default)	
R[0]	Output	PIN_D9	3	B3_N0	PIN_A9	3.3-V LV...default)		24mA (default)	
rst	Input	PIN_R22	6	B6_N0	PIN_M2	3.3-V LV...default)		24mA (default)	
VSYNC	Output	PIN_B11	3	B3_N0	PIN_B11	3.3-V LV...default)		24mA (default)	

Figure 6. Ball Testing Pin Assignment.

3.1.3 Paddle Testing

Afterwards, the paddle and game were tested together, using the pin assignment in Figure 7. Notice that we have now removed the LIFE and LEVEL from the inputs, since they should be generated based on the game state. PADDLE_LEFT and PADDLE_RIGHT were added as inputs to be used as pushbutton controls for the user. The reset pushbutton now restarts the entire game, resetting the LIFE and LEVEL values, as well as resetting the blocks, the paddle, and the ball on the screen. A video of a user playing the game can be seen here: https://youtu.be/bU9AzsO9vik.

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Differential Pair
rst	Input	PIN_R22	6	B6_N0	PIN_R22	3.3-V LV...default)		24mA (default)	
clock	Input	PIN_L1	2	B2_N1	PIN_L1	3.3-V LV...default)		24mA (default)	
PADDLE_RIGHT	Input	PIN_T22	6	B6_N0	PIN_T22	3.3-V LV...default)		24mA (default)	
PADDLE_LEFT	Input	PIN_T21	6	B6_N0	PIN_T21	3.3-V LV...default)		24mA (default)	
VSYNC	Output	PIN_B11	3	B3_N0	PIN_B11	3.3-V LV...default)		24mA (default)	
R[3]	Output	PIN_B7	3	B3_N1	PIN_B7	3.3-V LV...default)		24mA (default)	
R[2]	Output	PIN_A7	3	B3_N1	PIN_A7	3.3-V LV...default)		24mA (default)	
R[1]	Output	PIN_C9	3	B3_N1	PIN_C9	3.3-V LV...default)		24mA (default)	
R[0]	Output	PIN_D9	3	B3_N0	PIN_D9	3.3-V LV...default)		24mA (default)	
HSYNC	Output	PIN_A11	3	B3_N0	PIN_A11	3.3-V LV...default)		24mA (default)	
G[3]	Output	PIN_A8	3	B3_N0	PIN_A8	3.3-V LV...default)		24mA (default)	
G[2]	Output	PIN_B9	3	B3_N0	PIN_B9	3.3-V LV...default)		24mA (default)	
G[1]	Output	PIN_C10	3	B3_N0	PIN_C10	3.3-V LV...default)		24mA (default)	
G[0]	Output	PIN_B8	3	B3_N0	PIN_B8	3.3-V LV...default)		24mA (default)	
B[3]	Output	PIN_B10	3	B3_N0	PIN_B10	3.3-V LV...default)		24mA (default)	
B[2]	Output	PIN_A10	3	B3_N0	PIN_A10	3.3-V LV...default)		24mA (default)	
B[1]	Output	PIN_D11	3	B3_N0	PIN_D11	3.3-V LV...default)		24mA (default)	
B[0]	Output	PIN_A9	3	B3_N0	PIN_A9	3.3-V LV...default)		24mA (default)	

Figure 7. Left and Right Paddle Pin Assignment.

3.1.4 Extra Features Testing

After this, some extra features were added to the game, such as different angles of ball bouncing on the paddle depending on where the ball hits, launching the ball from the paddle at every life, and power-

ups, as described in the previous section. To launch the ball, a new pushbutton pin assignment was made, as shown in Figure 8. A video of the system at this point can be found here: https://youtu.be/zVbmLscevx4.

3.1.5 Final Testing

Then, some finishing touches were added, such as the gradient colour schemes for each level, and the rainbow effect on the Penetrator powerup. The complete game in action can be seen tested in this video: https://youtu.be/I9pZ3tcOD4k.

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Differential Pair
clock	Input	PIN_L1	2	B2_N1	PIN_L1	3.3-V LV...default)		24mA (default)	
LAUNCH	Input	PIN_R21	6	B6_N0	PIN_R21	3.3-V LV...default)		24mA (default)	
rst	Input	PIN_R22	6	B6_N0	PIN_R22	3.3-V LV...default)		24mA (default)	
PADDLE_LEFT	Input	PIN_T21	6	B6_N0	PIN_T21	3.3-V LV...default)		24mA (default)	
PADDLE_RIGHT	Input	PIN_T22	6	B6_N0	PIN_T22	3.3-V LV...default)		24mA (default)	
R[2]	Output	PIN_A7	3	B3_N1	PIN_A7	3.3-V LV...default)		24mA (default)	
G[3]	Output	PIN_A8	3	B3_N0	PIN_A8	3.3-V LV...default)		24mA (default)	
B[0]	Output	PIN_A9	3	B3_N0	PIN_A9	3.3-V LV...default)		24mA (default)	
B[2]	Output	PIN_A10	3	B3_N0	PIN_A10	3.3-V LV...default)		24mA (default)	
HSYNC	Output	PIN_A11	3	B3_N0	PIN_A11	3.3-V LV...default)		24mA (default)	
R[3]	Output	PIN_B7	3	B3_N1	PIN_B7	3.3-V LV...default)		24mA (default)	
G[0]	Output	PIN_B8	3	B3_N0	PIN_B8	3.3-V LV...default)		24mA (default)	
G[2]	Output	PIN_B9	3	B3_N0	PIN_B9	3.3-V LV...default)		24mA (default)	
B[3]	Output	PIN_B10	3	B3_N0	PIN_B10	3.3-V LV...default)		24mA (default)	
VSYNC	Output	PIN_B11	3	B3_N0	PIN_B11	3.3-V LV...default)		24mA (default)	
R[1]	Output	PIN_C9	3	B3_N1	PIN_C9	3.3-V LV...default)		24mA (default)	
G[1]	Output	PIN_C10	3	B3_N0	PIN_C10	3.3-V LV...default)		24mA (default)	
R[0]	Output	PIN_D9	3	B3_N0	PIN_D9	3.3-V LV...default)		24mA (default)	
B[1]	Output	PIN_D11	3	B3_N0	PIN_D11	3.3-V LV...default)		24mA (default)	

Figure 8. Final Pin Assignment.

3.2 FPGA Resource Utilization

A summary of the FPGA resource utilization can be seen in Figure 9. Despite the complexity of the game and its circuit, still only a relatively small percentage of the resources are used. Only 12% of the total logic elements are used, 6% of pins, and 7% of memory bits.

Flow Summary	
Flow Status	Successful - Thu Dec 01 21:15:01 2016
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	g30_Breakout_Game
Top-level Entity Name	g30_Breakout_Game
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Total logic elements	2,298 / 18,752 (12 %)
Total combinational functions	2,271 / 18,752 (12 %)
Dedicated logic registers	425 / 18,752 (2 %)
Total registers	425
Total pins	19 / 315 (6 %)
Total virtual pins	0
Total memory bits	16,384 / 239,616 (7 %)
Embedded Multiplier 9-bit elements	1 / 52 (2 %)
Total PLLs	0 / 4 (0 %)

Figure 9. Quartus II Flow Summary.

3.3 Timing Analysis

A summary of the timing analysis can be seen in Figure 10 and Figure 11, showing the fast model hold summary and slow model Fmax summary from TimeQuest in Quartus II. As can be seen in the figures, the timing requirements are met.

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

Figure 10. TimeQuest Fast Model Hold Summary.

Slow Model Fmax Summary				
	Fmax	Restricted Fmax	Clock Name	Note
1	46.58 MHz	46.58 MHz	clock	

Figure 11. TimeQuest Slow Model Fmax Summary.

4 Conclusion

We have designed our version of the Breakout game, with a set of unique features: powerups, angled ball bouncing on the paddle, level colour schemes, increased speed at every level, and a trailing "rainbow" effect for one of the powerups.

Not many significant difficulties arose during the design process. There were a few minor bugs here and there, such as block breaking not working the way we intended, or various timing problems, causing the display to not function as intended. These problems were resolved relatively quickly, allowing the implementation of many features.

Given more time, many more features could have been added. For instance, different types of blocks could be implemented. There could be blocks that award more points (higher on the screen), blocks that are invisible until hit, blocks that that multiple hits to break, or blocks that explode on hit, causing other blocks to break. Also, more types of powerups could have been added. We could have implemented more positive powerups such as one that makes the ball stick to the paddle, a powerup that gives the user the ability to shoot missiles from the paddle, or a powerup that makes the ball "explosive", causing multiple blocks to break when hitting a block. Some more negative ones could also have been added, such as one that speeds up the ball, one that automatically removes a life, or one that rotates the display, disorienting the player.

See the attached files for all the VHDL files, the .sof file, and a picture of the RTL circuit representation of the circuit.

5 Grade Sheet

Grade Sheet for Lab #5 Fall 2016.

Group Number: 30 .

Group Member Name: Gabriel Chodos Student Number: 260637105 .

Group Member Name: Sean Stappas. Student Number: 260 639 512 .

Marks			TA Signatures
2	1.	VHDL Description of the walls and moving ball	AL .
2	2.	Demonstration of the walls and moving ball	AL .
2	3.	VHDL description of the blocks	AL .
2	4.	Demonstration of the blocks and block breaking	AL .
2	5.	VHDL for the moving paddle	AL .
2	6.	Demonstration of the moving paddle	AL .
2	7.	Demonstration of the complete breakout game	AL .

Each part should be demonstrated to one of the TAs who will then give a grade and sign the grade sheet. Grades for each part will be either 0, 1, or 2. A mark of 2 will be given if everything is done correctly. A grade of 1 will be given if there are significant problems, but an attempt was made. A grade of 0 will be given for parts that were not done at all, or for which there is no TA signature.