FPGA Breakout

Atari 2600 Video Game FPGA Reproduction

Doug Cumbie, *CpE MS Major, UCF Fall 2008*
EEL 5772 – FPGA Design, Jooheung Lee
December 2, 2008

***Abstract*—Video games have come a long way in the passed 30 years. Original video games of the 1970s consisted of analog and digital circuits combined with an input device for game control and a video display for visual representation. These early designs did not use a microprocessor, mostly because microprocessors were not yet available. The limited technology of the 1970s forced game designers of the time to work a little harder to develop a user friendly and fun game. As the technology improved and the popularity for games greatly increased over the years, the video game industry has become on par with movies in sales and demand.**

The following report will document a redesign of the video game *Breakout* (1) using an FPGA (Field Programmable Gate Array). FPGAs have advanced in their power and ability to allow for the development of digital products. FPGAs provide designers with a clean slate of reconfigurable digital logic to develop many types of digital electronic circuits. Designs are implemented by using a Hardware Description Language (HDL), which consists of specific instructions that define a digital design. An FPGA is an ideal way to redevelop an early digital video game because of these unique features. The Breakout game design documented in this report will cover the various components necessary to create the game. The methods used in the design as well as the challenges met will be presented.

I. Introduction

The video game Breakout was envisioned by the founder of Atari, Nolan Bushnell (1). The inspiration came from Atari's first video game, Pong, which was released in 1972 (2). Pong was a coin-operated, two-player video game that resembled table tennis. Two paddles were controlled by two players to bounce a ball from each end of the screen. Points were scored when the ball was missed and moved passed a player's paddle. Pong was developed by Atari's Al Alcorn, and consisted entirely of discrete logic circuits.

There was no financial support for this document. The work within this text represents information collected from independent research on FPGA design and development and the Breakout video game. Please see the References (Section IV) for more detailed information.

This paper was created for the FPGA Design course (EEL 5772) of the University of Central Florida's Computer and Electrical Engineering department, held during the Fall term of 2008. The professor of this course was Dr. Jooheung Lee.

The concept of Breakout was similar to Pong, but presented a one-player variation. Development of the game was assigned to Steve Jobs (founder of Apple Computer) in 1975, who was an employee at Atari at this time. Alcorn, who was project manager of the game, offered the task of developing a Breakout prototype for $700 with a $100 bonus for every Transistor-Transistor Logic (TTL) chip saved in the design. Jobs won the contract, but had most of the work done by his friend, and inventor of the Apple computer, Steve Wozniak. Wozniak at the time was an employee at Hewlett Packard and had exceptional skill is electronic design. Jobs, for personal reasons, fooled Wozniak into believing that the task was expected in four days time, and that if developed under 50 chips they'd get $700, if under 40 chips they'd get $1000. Wozniak and Jobs spent those four days without sleep developing the prototype Breakout and in the end, delivered a design of 42 chips - better than half the total number of chips of most games. Almost all the design and development was done by Wozniak, with Jobs profiting about $5000 and only giving $350 of it to Wozniak, who assumed that the total pay was the originally told $700 (1).

Wozniak's design was very impressive at only 42 discrete logic chips. However, the minimal design was difficult to understand and therefore not ideal for testing and reproduction, so it never was sold. Instead, Atari redesigned it, keeping most of the original game-play developed by Wozniak. Breakout was released in 1976 as a coin-op arcade, and later on the Atari 2600 home system and has since been one of Atari's biggest games. The design by Wozniak was impressive by the chip count used, but was an example of the common non-microprocessor games of the time. Each component of the game (video generation, input device controllers, and game logic) was completely contained in the digital hardware logic.

A. Gameplay

The objective of Breakout is to bounce a ball off of a moveable paddle towards a wall of bricks. The wall consists of 6 rows of colored bricks (108 bricks total). When the ball collides with the wall, bricks are successively removed and points are scored. The player must avoid having the ball fall passed the paddle (at the bottom of the screen), which will cost one life. Different points are earned from hitting the bricks in higher rows. The top two rows give 7 points per

brick, middle 2 rows give 4 points, and bottom 2 rows give 1 point. When all bricks are destroyed, the player advances to the next level (3).

Figure 1. Breakout Screenshot

II. DESIGN

The design and methods of implementation for the FPGA recreation of Breakout consists of various modules and components that work together to obtain the gameplay realized in the original game. The main components of the design are listed below:

- Video Generation
- User Input Controller
- Internal Logic
  - Graphics Display
  - Collision Logic
  - Score/Lives Logic
  - Paddle Frame Counter
  - Ball Frame Counter
  - Ball Movement

In the figure below, the design flow of each component is visually represented.

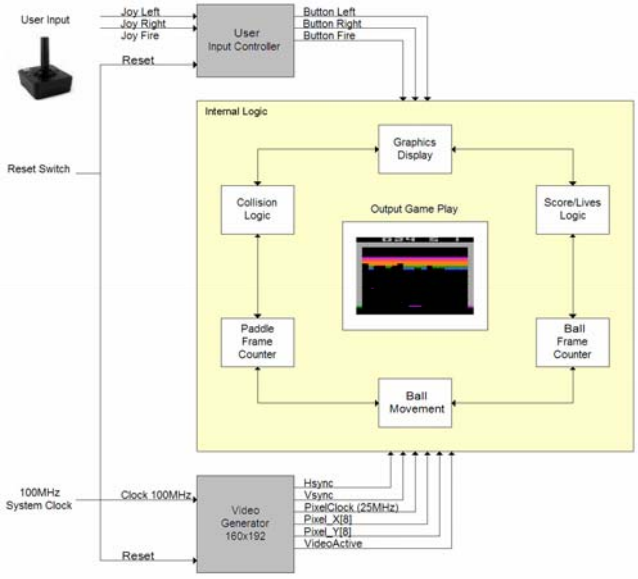

Figure 2. Breakout Design

The FPGA Breakout video game was designed and developed using the Xilinx XUPVP2 Virtex-II development board. The code was written entirely in Verilog. The following sections outline the design and development of the Breakout FPGA video game.

A. Video Generation

The output of any video game is always to some form of video display. Early arcade game used a CRT similar to a television as the video game's visual output. In this course, Lab 2 introduced the concept of video generation on a VGA display. This project utilized the results of Lab 2 to create a video display for Breakout that was consistent with the early Atari 2600 game version.

The Atari 2600 video display consisted of a region of pixels at overall resolution of 228x262 pixels (6). Generation of the video used a television's electron beam scanned to the screen; the same way that VGA monitors generate their video. The active video region consists of a visual block of 160x192 pixels. This is the standard resolution of the Atari 2600, and the resolution desired for the FPGA Breakout design. To generate video at this resolution, the 640x480 VGA module developed in Lab 2 was used (7). The module was extended by scaling the horizontal and vertical counters such that X and Y pixel counters would be returned that were within the 160x192 range. The module is displayed below.

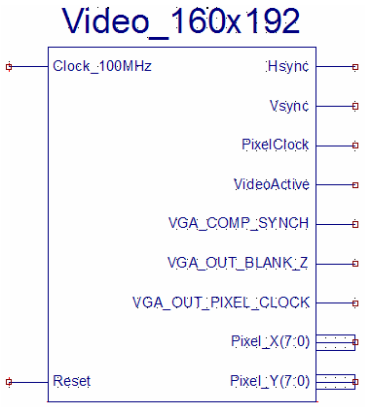

Figure 3. 160x192 Video Generation Module

The Video_160x192 module uses instantiations of 2 modules to generate the output video. These modules are listed below:

- **ClockDivider**: Used to divide the 100MHz system clock to 25MHz for the 640x480 VGA display.
- **VGA_Sync**: Generates the horizontal and vertical VGA signals for a 640x480 display.

The Video_160x192 module takes the output horizontal and vertical counters of the **VGA_Sync** module, and returns scaled 8-bit values for X and Y pixel positions in the 160x192

range. This scaling is done by using the following simple operations.

$$Pixel\_X = \frac{HorizontalCount}{4}$$

$$Pixel\_Y = \{\frac{VerticalCount - 48}{2} \mid VerticalCount > 48\}$$

The horizontal pixel range of 160 is obtained by simply dividing the Horizontal Count by 4 (640/4 = 160). For the vertical pixel range, a simple division was not available to obtain the 192 pixel range. So from the 480 pixels, 48 were removed from the top and bottom leaving 384 pixels. By dividing this by 2, the 192 pixel range is obtained. The *VideoActive* signal is asserted when the scan lines are in the active video region. For the 160x192 resolution, this had to be modified to move the 48 pixels above and below the vertical region to the blanking region. The following Verilog code demonstrates this.

```
assign VideoActive = (Hcount > 0 && Hcount <= 640) &&
                     (Vcount > 47 && Vcount < 432);
```

The resulting video output is generated by apply colors to the pixels pointed to by the *Pixel_X* and *Pixel_Y* signals. At this resolution the relatively large, rectangular pixel shape that was standard in the Atari 2600 system is realized.

1) Graphics Display

The graphics for FPGA Breakout are displayed by applying specific colors to the *Red*, *Green*, and *Blue* VGA output signals when the *Pixel_X* and *Pixel_Y* values are at certain positions. The game graphics are simple and consist of the following objects:

- **Walls**
- **Paddle**
- **Ball**
- **Bricks**
- **Score Board**

The dimensions for each object was obtained by visually estimating the relative sizes of each object. These dimensions dictate when each color is applied to the display to paint the objects. For example, the left and right walls are gray in color and located at the edge of the screen and are each 8 pixels wide. To draw these objects the 24-bit RGB output (represented by a single 24-bit signal, *PixelColor*) is set to gray when *Pixel_X* < 8 AND *Pixel_X* >= 152. Similar logic is used to draw the ball, paddle bricks, and score board.

2) Graphical Animations

The animated graphics in FPGA Breakout consists of the moving ball and the user controllable paddle. These objects positions change at specific rates creating movement as their graphics are redrawn to the screen. To maintain the update rates of the ball and paddle positions, a **FrameCounter** module was developed. The **FrameCounter** is an adjustable, circular counter that asserts a flag when the count has reached its max. The counter counts at the rising edge of the input 25MHz clock. To define the count, an input 32-bit value is supplied. This input may be changed dynamically, which allows for a change in the rate of ball movement during gameplay, for example. The exact animation rates of the original game were not known, so the following rates were implemented.

- **Paddle**: Constant at 200Hz (5ms)
- **Ball**: Variable rates depending on the number of bricks remaining:
  - 108 to 95: 33Hz (30ms)
  - 94 to 80: 50Hz (20ms)
  - 79 to 20: 100Hz (10ms)
  - 19 to 0: 125Hz (8ms)

a) Paddle Movement

The Paddle position is update when the user moves the joystick left or right. When a left or right movement is detected, the paddles x-position is decremented or incremented by 1.

b) Ball Movement

The ball movement consists of updating the ball's X, Y, and Slope values during movement in open space or after collisions have been detected. The ball's position is always updated when its associated ball **FrameCounter** asserts. The ball maintains its trajectory until is collides with a wall, the paddle or bricks. At this point the ball's slope is modified so that it will assume a new trajectory. The ball's slope is simply reversed in the X or Y direction when it collides with the wall or bricks. However, collisions with the paddle result in modification of the slope angle depending on where the ball collides with the paddle's surface. The following figure shows the resulting ball angle based on its collision with the paddle.

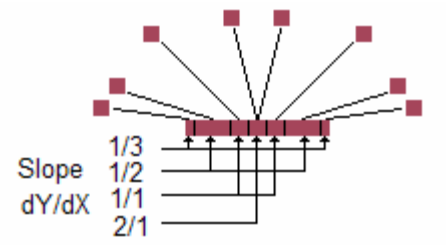

Figure 4. Ball Trajectory

3) Collision Logic

Collisions encompass most of the gameplay action of the video game. Collisions occur when the ball encounters another object and is forced to alter its current trajectory. Points are scored when the player collides the ball with the bricks. The ball's movement is altered depending on how it hits the edge of another object. The method used for altering the ball's trajectory consists of reversing the direction when

colliding with the face of an object. For example, when colliding with the left wall, the ball's x-movement is reversed while the y-movement is preserved so as to alter its motion away from the wall. The ball's slope is sotred in 2 variables called *xBallDelta* and *yBallDelta*. When a specific collision occurs, these variables are altered to modify the ball's movement.

Collisions with bricks include additional logic for interpreting what kind of brick was hit so that the appropriate points can be added to the current score. There are 6 rows of 18 bricks for a total of 108 bricks. Each brick's state is maintained in a 108-bit register, such that a 1 in a bit position indicates that the brick is currently present on the screen. The ball's position is periodically checked against these brick-related combinational logic. When the ball passes over a region that is computed as the edge of a brick (top, bottom, left, or right), a collision is detected. Bricks can only be destroyed if the ball collides with the top or bottom edge. Collision with a side edge will only ricochet the ball. Upon detecting a top or bottom collision with a brick, the following actions are performed:

```
if (BRICKS[18*yBallBrick + xBallBrick])
begin
  // 1. Clear brick state
  BRICKS [18*(yBallBrick) + xBallBrick] = 0;
  // 2. Decrement total brick count
  brickCount = brickCount - 1;
  // 3. Alter ball's motion
  yBallDelta = -yBallDelta;
  // 4. Update score
  Score = CalcScore(Score, yBallBrick);
end
```

If the ball fails to collide with any of the available objects, it falls into the open hole located at the bottom of the screen. This deducts one life from the players remaining lives.

4) Score and Lives Logic

FPGA Breakout's main objective is to destroy all the bricks across two levels before losing all 5 available lives. Points are scored by destroying the bricks with a total of 864 points obtained if all bricks are destroyed. The score board is displayed at the top of the screen above the main game area. The score is maintained in an internal 10-bit register which is used to control the drawing of the numerical fonts at the top of the screen. These fonts are held in a Xilinx Coregen Single Port ROM of 16 bit width by 140 bit depth. During refresh of the game graphics, the current score is converted to a graphical 3-digit number by performing lookups in the number ROM and then painting the number's pixels gray when a 1 is read from the ROM. Each digit contains its own number ROM module instantiation. Handling of the current lives value is maintained in the same fashion.

To convert the 10-bit score to a valid 3-digit number, a module was created that takes the Score as an input and returns a BCD representation of the hundreds, tens, and ones digits. The **Score3BCD** module uses 2 Coregen Divider modules to generates the Score's 3 digits, which are repainted during every graphical cycle.

Figure 5. Score3BCD Verilog Module

B. User Input Controller

The input device for many early video games was the joystick. A basic joystick provided a small set of inputs that would command movement for a player along with an action input, typically for jumping or firing. The original Breakout game was played using the Atari paddle controllers, which consisted of a rotating control used to sweep the paddle left and right. An Atari paddle controller was not available for this design, but a standard Atari joystick was. This joystick was chosen as the user input device.

The inputs from the Atari joystick consist of simple normally open switches. These signals are routed to a physical interface consisting of a female 9-pin D-sub connector. When the joystick is moved in one direction or the fire button is pressed, a circuit is closed to ground, passing a logic low to the switch's output. The pinouts for an Atari Joystick is listed in the table below (4).

Table 1. Atari Joystick Pinouts

Pin	Signal
1	Up
2	Down
3	Left
4	Right
5	Unused
6	Fire
7	Unused
8	Ground
9	Unused

A hardware interface combined with a Verilog module was developed to communicate with the joystick. The hardware interface consisted of a male 9-pin D-sub connector, some wire, and a 60-pin expansion header used for connecting the Atari joystick to the Xilinx Virtex-II development board. The Xilinx I/O expansion headers were used to accept the joystick's input commands. The interface is displayed in the schematic below.

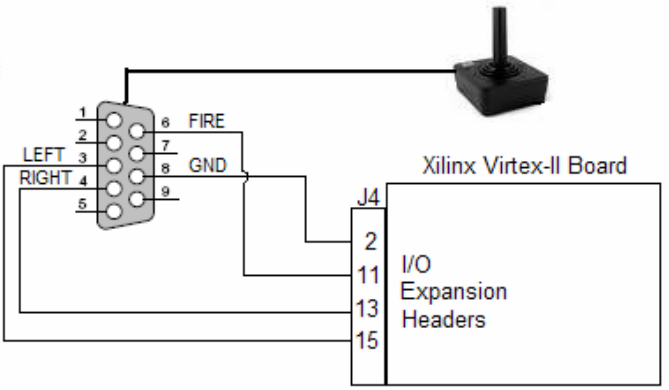

Figure 6. Joystick to Xilinx Board Hardware Interface

To interpret the commands from the joystick a Verilog module was developed. Since the commands are received as negative logic values, the Verilog modules basically converts the levels to positive logic at the output. The joystick commands are then used by the main internal logic to move the paddle and start the ball movement. The Verilog module is displayed in the figure below.

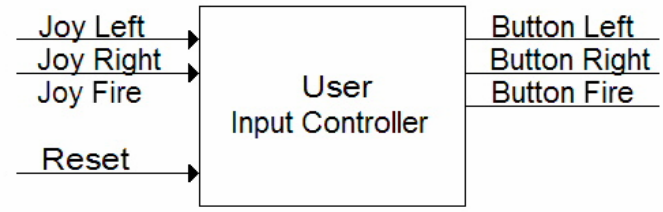

Figure 7. Verilog User Input Controller Module

III. CONCLUSION

The final design of the FPGA Breakout video game was successful in meeting the requirements desired in that the completed product was a discrete logic circuit in the form of an FPGA. The overall presentation matched most of the original game's look and feel. Each component of the design proved to be a challenge with the collision logic presenting the most difficulty. Some portions of the collision logic between the ball and bricks still contained issues consisting of incorrect responses in ball movement. This resulted in some unpredictable behavior, such as the ball destroying more bricks instead of deflecting away. This behavior however, is not often and therefore does not interfere with the overall game functionality.

The Xilinx FPGA design summary for the FPGA Breakout design generated by the Xilinx ISE development environment is displayed below.

Table 2. FPGA Breakout Design Summary for Xilinx Virtex-II XC2VP30

Logic Utilization	**Used**	**Available**	**Utilization**
Total Slice Registers	928	27,392	3%
Flip Flops Slice Registers	882		
Number of 4-input LUTs	3,593	27,392	13%
Occupied Slices	2,171	13,696	15%
Slices containing only related logic	2,171	2,171	100%
Slices containing only unrelated logic	0	2,171	0%
Total 4-input LUTs	3,776	27,392	13%
Total equivalent gate count	305,960		

The summary above reveals that the resources used were relatively minimal for the Virtex-II board. Overall, the design of the FPGA Breakout video game proved successful by meeting the requirements and at the same time producing an entertaining result.

IV. REFERENCES

Web-based Resources:

[1] A Complete History of Breakout. Marty Goldberg. http://classicgaming.gamespy.com/View.php?view=Articles.Detail&id=395

[2] Breakout (arcade game). Wikipedia.. http://en.wikipedia.org/wiki/Breakout_(arcade_game)

[3] Breakout (2600). Online Manual. http://strategywiki.org/wiki/Breakout_%282600%29/Getting_Started

[4] Atari Joystick Pinout. Pinouts.ru website. http://pinouts.ru/Inputs/JoystickAtari2600_pinout.shtml

Books:

[5] Xilinx University Program Virtex-II Pro Development System. Hardware Reference Manual. Xilinx. March 8, 2005.

[6] Stella Programmer's Guide. Steve Wright. Dec 3, 1979

Articles, Journal, Reports:

[7] Lab 2. Displaying Colors on a VGA Monitor. Doug Cumbie. Sept 16-30, 2008

[8] Lab 3. Input from a PS/2 Keyboard. Doug Cumbie. Sept 30-Oct 14, 2008

[9] Lab 4. Displaying PS/2 Keyboard Input on a VGA Display. Doug Cumbie. Oct 14-28, 2008