Abstract

The purpose of this project was to learn the functionality of video-output subsystem and Video Graphic Adapter (VGA) standard as well as on-chip (FPGA) to I/O device interfacing. Another goal of this project was to get practical experience in design and implementation of the real-time high performance signal generators implemented on-chip (FPGA) with custom logic circuits. Getting hands-on experience in interfacing the on-chip circuits with external real-time I/O devices (VGA Monitor). Finally the final goal was to learn VHDL-coding technique for real-time applications in the environment of Xilinx ISE CAD system and hardware evaluation platform based on Xilinx Spartan-3E FPGA.

Introduction

The primary design goal was to implement a simple video game based on Atari's pong. The game has a 'ball' which can 'fly' around the field. When in contact with the wall or a player the ball was to 'bounce' ±90°. When entering the net the ball was to reset position to the middle of the field, and then continue in motion. We were to use four input switches to control player movement up and down.

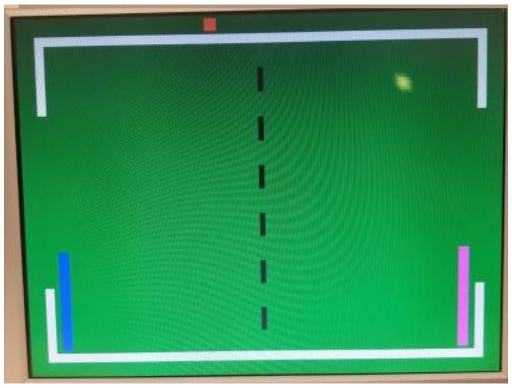


Figure 1: Game Field

The basic layout of the game can be seen above in Figure 1. The blue and pink rectangles are the 'players', the yellow square is the 'ball' and the white outer edge is the 'border'. The black dotted line is purely aesthetic and has no impact on the game.

In addition to these basic requirements we also included the ability to increase or decrease the player and ball speed as well as increase or decrease the player size (vertically). These game modifications are made through the use of push buttons. Also we included a score indicator. This can be seen as the orange square at the top of the screen in Figure 1. This square moves left or right each time a point is scored based on who the scorer was.

System Specifications

DAC_CLK (Functional Specifications)[1]:

1. Clock Speed of the VGA Controller
2. The DAC_CLK monitors the rising edge of the clk signal and asserts and de-asserts itself based on the previous DAC_CLK value

DAC_CLK (Technical Specifications):

1. The DAC_CLK is synchronized to the rest of the system via a clock with a frequency of 50 MHz.
2. DAC_CLK signal:
   a) DAC_CLK stays asserted for one clock cycle
   b) DAC_CLK stays de-asserted for one clock cycle
   c) steps a) and b) are repeated ad infinitum

VGA Sync (Functional Specifications):

1. Monitors external DAC_CLK signal
2. On the rising edge of the DAC_CLK signal, an integer that is used to keep track position on the line is incremented
3. This line integer along with another integer which keeps track of the vertical position in the frame are used to assert and de-assert Hsync and Vsync signals

VGA Sync (Technical Specifications):

1. The VGA Sync is synchronized to the rest of the system via a DAC_CLK with a frequency of 25 MHz.
2. Hsync signal:
   a) Line width & front porch: Hsync is asserted for 656 DAC_CLK cycles
   b) Sync pulse: Hsync is de-asserted for 96 DAC_CLK cycles
   c) Back porch: Hsync is asserted for 48 DAC_CLK cycles, steps a), b) and c) are repeated ad infinitum
3. Vsync signal:
   a) Frame width & front porch: Vsync is asserted for 490 Hsync cycles
   b) Sync pulse: Vsync is de-asserted for 2 Hsync cycles
   c) Back porch: Vsync is asserted for 33 Hsync cycles, steps a), b) and c) are repeated ad infinitum

[1]For easier comprehension, each process in PongV2 was given its own System Specifications

Game Logic and Visuals (Functional Specifications):

1. Displays static and dynamic elements onto the Monitor
2. The position of all dynamic elements are calculated at the end of each frame
3. The ball is redrawn every frame in a new position depending upon its speed, and whether it is colliding with any objects on the screen
4. The Players are redrawn every frame in a new position depending upon player input

Game Logic and Visuals (Technical Specifications):

1. The Game Logic and Visuals are synchronized to the rest of the system via a DAC_CLK with a frequency of 25 MHz.
2. If $0 \leq \text{hsync} \leq 639$ and $0 \leq \text{vsync} \leq 479$ then set Rout, Gout and Bout signals

Device Descriptions

Symbols

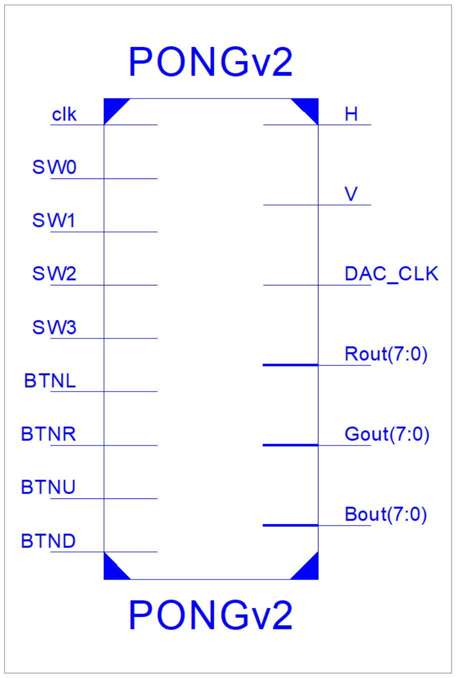


Figure 2: PongV2 Symbol

VGA Specifications

The VGA standard uses three signals to display colour on a VGA monitor, red, green and blue. This colour information is then used to drive a electron gun which emits electrons which paints one primary colour on a point on the monitor.

A typical VGA monitor consists of an array of 640 by 480 pixels. The complete pictures is usually referred to as a frame while a single 640 by 1 horizontal section is referred to as a line. To complete a frame multiple deflection circuits are used to move the electrons across and down the screen. These circuits need two synchronization signals H and V to stop the deflection at the correct times so that the lines orient correctly.

Every VGA monitor has an internal clock which determines how fast each pixel is updated. The monitor refreshes the screen depending on the H and V signals which in themselves directly or indirectly depend on the internal clock of the monitor.

The VGA monitor will update the screen at precise times with these specific synchronization pulses. This behavior is described in the tables below.

Table 1: VGA Horizontal Parameters

Parameters	**Complete Frame**	**Front Porch**	**Sync Pulse**	**Back Porch**	**Image Area**
Clock Cycles	800	16	96	48	640
Time	31.77 μs	0.64 μs	3.81 μs	1.91 μs	25.42 μs

Table 2: VGA Vertical Parameters

Parameters	**Complete Frame**	**Front Porch**	**Sync Pulse**	**Back Porch**	**Image Area**
Clock Cycles	525	10	2	33	480
Time	16.6 ms	1.04 ms	64 μs	0.32 ms	15.25 ms

Block Diagrams

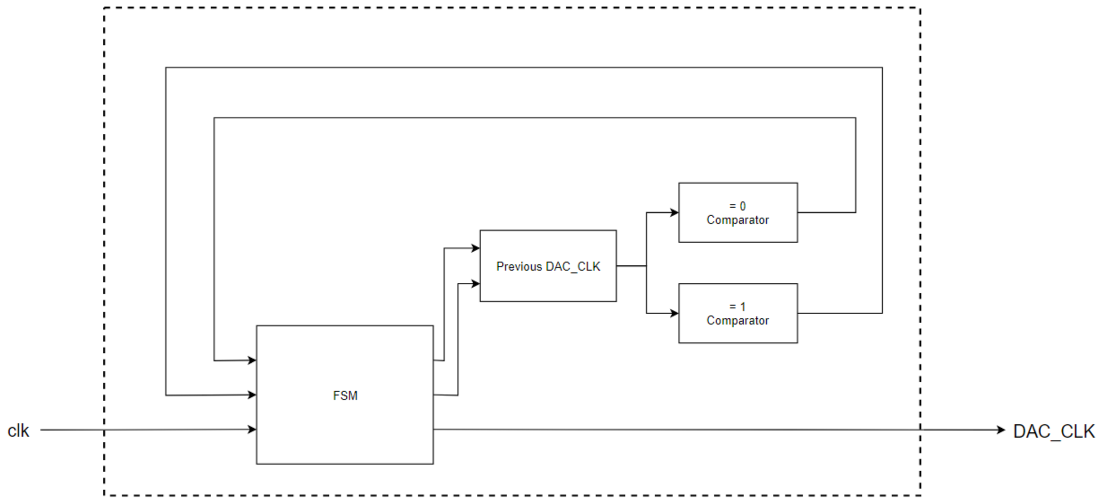


Figure 3: DAC_CLK Block Diagram

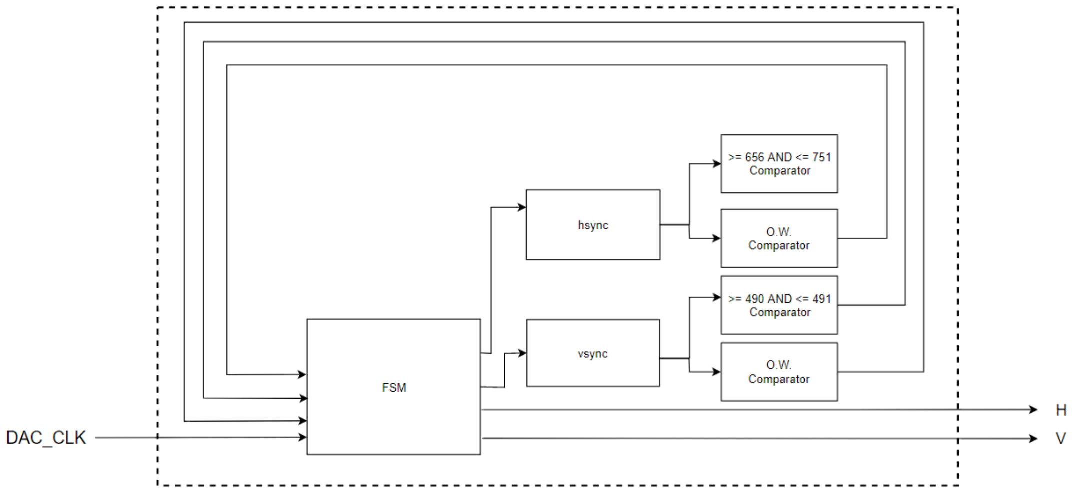


Figure 4: Synchronization Block Diagram

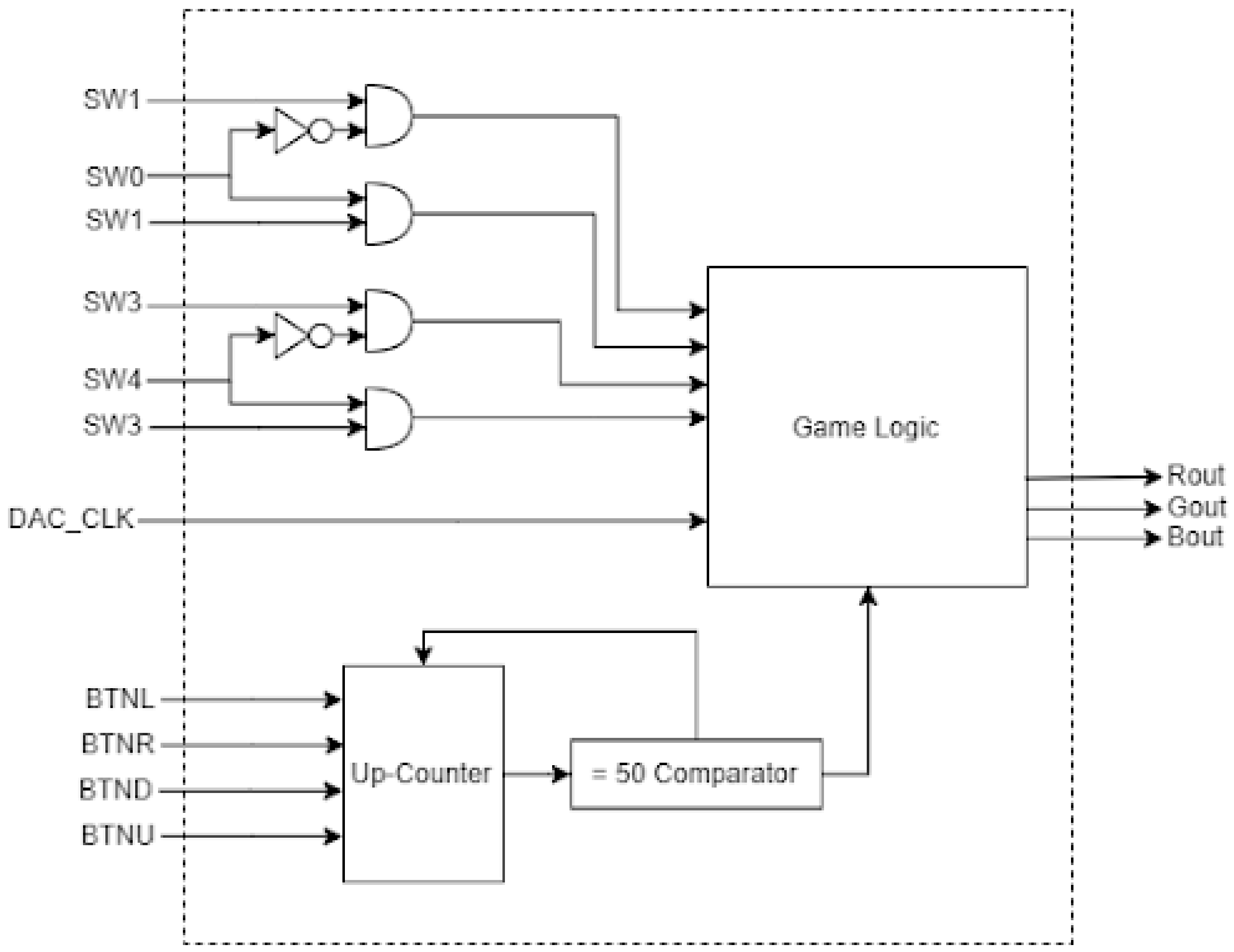


Figure 5: Game Logic and Visuals Block Diagram

State Diagrams

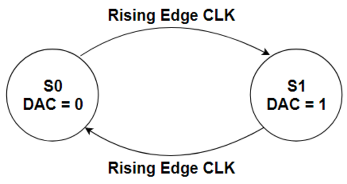
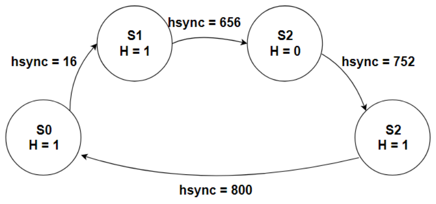


Figure 6: DAC_CLK FSM



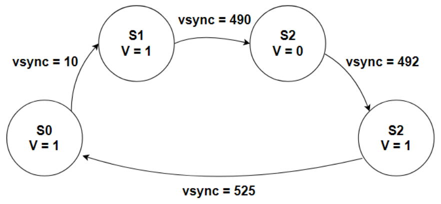


Figure 7: Synchronization FSMs

Process Diagram

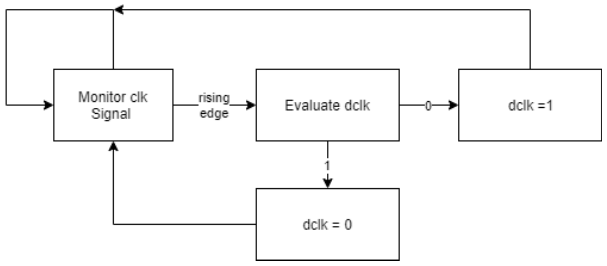


Figure 8: dclk synchronization process

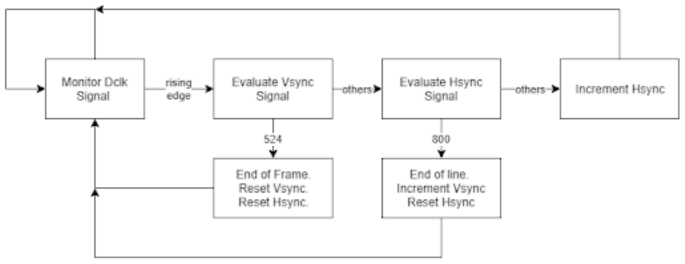


Figure 9: Synchronization Process diagram

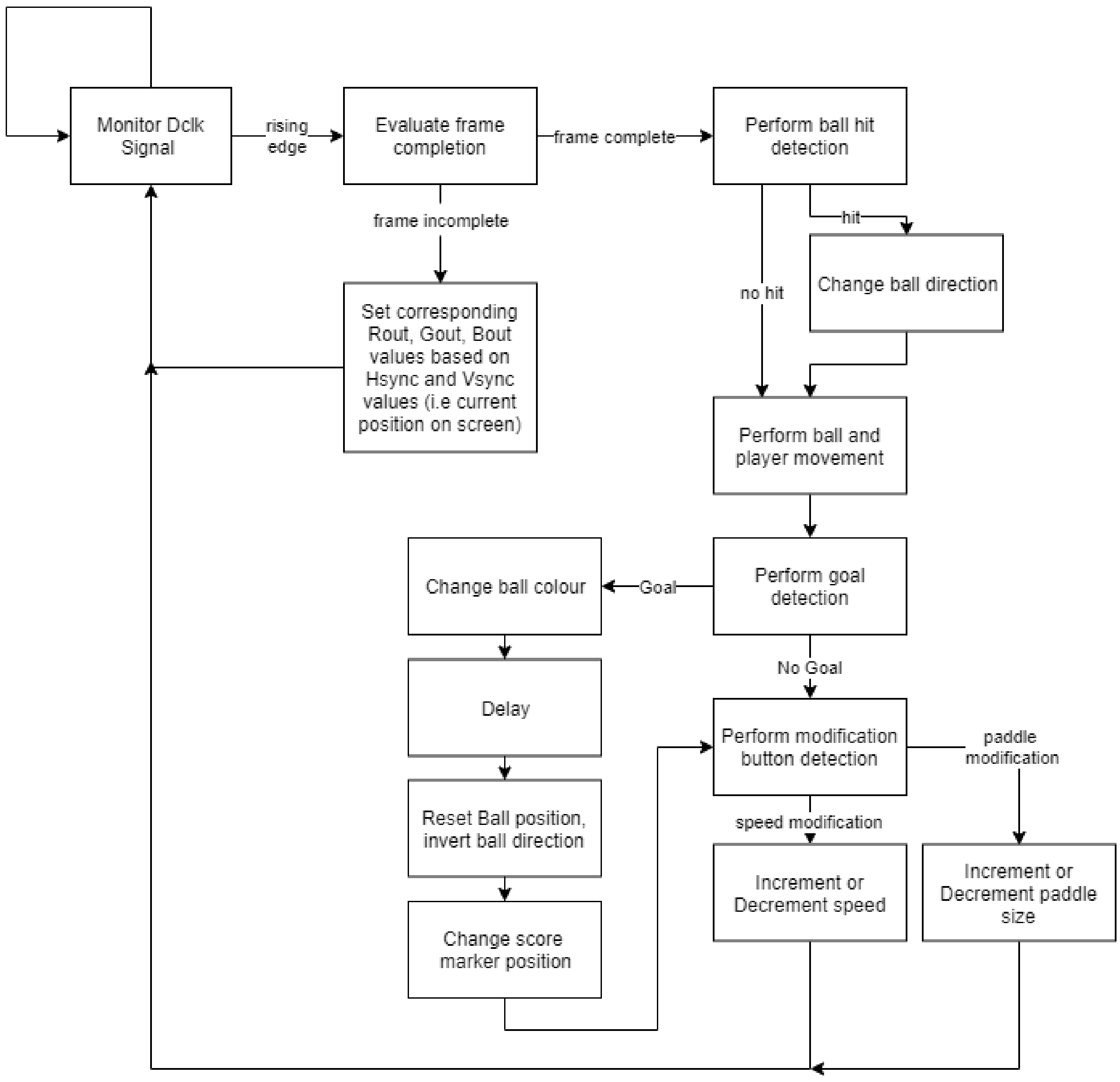

Monitor Dclk Signal
rising edge
Evaluate frame completion
frame complete
Perform ball hit detection
frame incomplete
Set corresponding Rout, Gout, Bout values based on Hsync and Vsync values (i.e current position on screen)
hit
no hit
Change ball direction
Perform ball and player movement
Change ball colour
Goal
Perform goal detection
No Goal
Delay
Perform modification button detection
paddle modification
speed modification
Reset Ball position, invert ball direction
Increment or Decrement speed
Increment or Decrement paddle size
Change score marker position

Figure 10: Game Logic and Visuals Process diagram

Results

Timing (Simulation) Diagrams

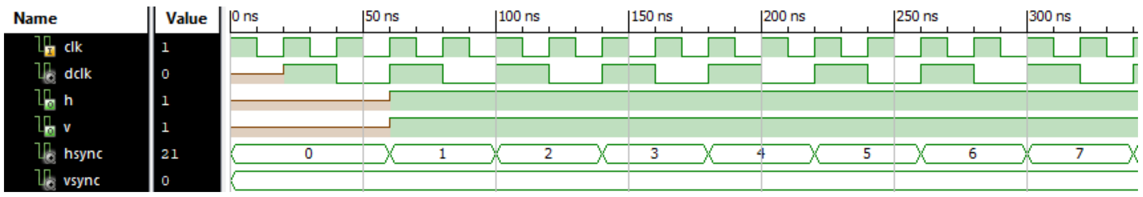


Figure 11: Visible Dclk Synchronization and Hsync frame increase

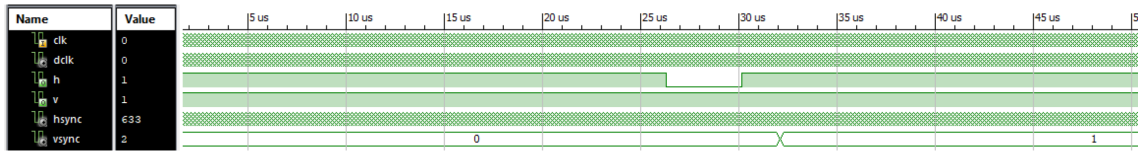


Figure 12: Visible H-Pulse and subsequent Vsync frame increase

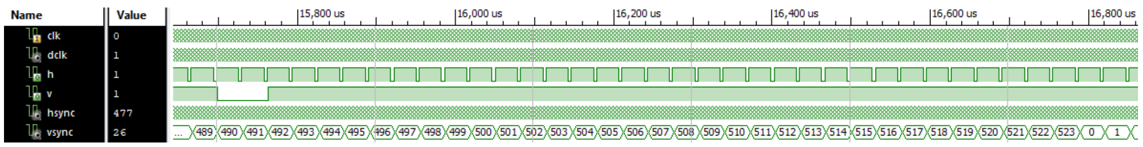


Figure 13: Visible V-Pulse and subsequent Vsync frame reset

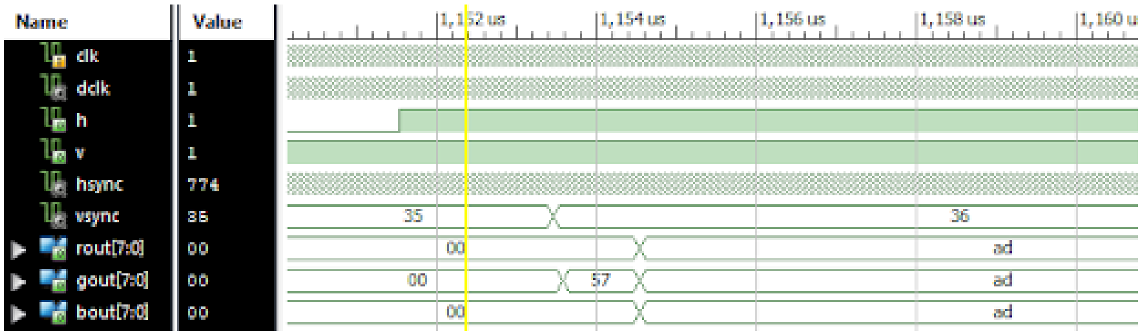


Figure 14: Visible RGB Static Image colour changing

Screen Captures

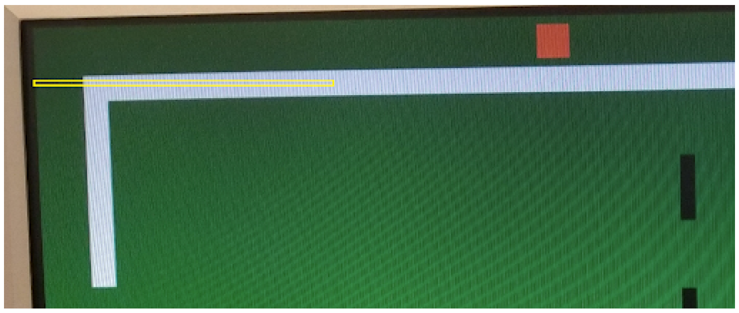


Figure 15: Static Image Colour Region

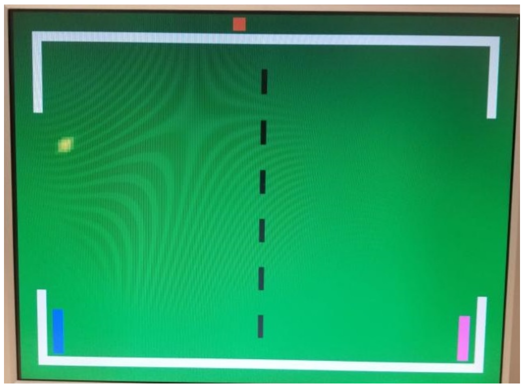


Figure 16: Game Start

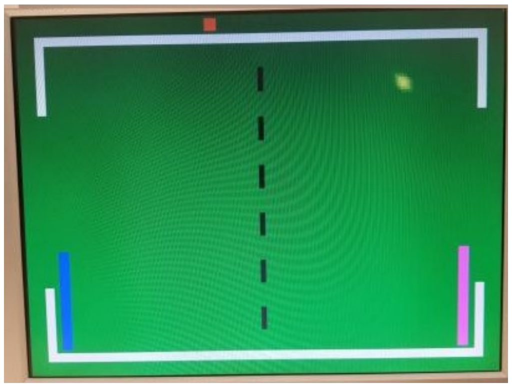


Figure 17: Dynamic Player Adjustment(long)

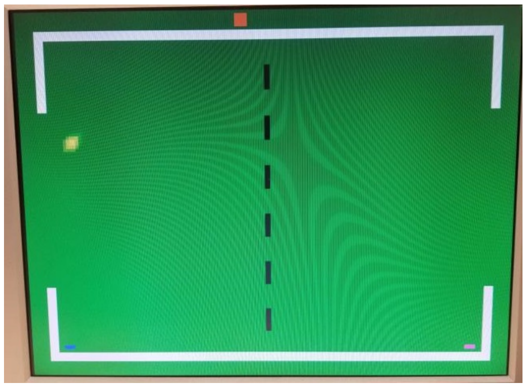


Figure 18: Dynamic Player Adjustment(short)

Brief Explanation

As seen above in Figure 11, the Dclk was generated to have a period of 40ns, this was done by toggling it on the rising edge of the main clock. The state machine for this can be seen above in Figure 6, and it's process diagram can be seen in Figure 8.

We can see the process of frame synchronization being carried out in Figure 12 and Figure 13. After 656 Dclk cycles we can see the Sync Pulse of signal H in Figure 12 being de-asserted for 96 Dclk cycles. 48 Dclk cycles later we can see the vsync line counter increment. In Figure 13 we can see the Sync Pulse of signal V being de-asserted for 2 lines (as seen by the H-Sync pulses above it) We then see the 'back porch' of 33 lines (as seen by Vsync signal counter below it) before the Vsync Signal resets. The state machines for this synchronization can be seen above in Figure 7 and the process diagram can be seen in Figure 9.

It can be seen from Figure 14 that across a number of Dclk cycles there is output colour change. At 1,154μs we can see the colour output is 0x005700 (Dark Green, playing field). Then at 1,156μs the colour output is 0xadadad (Light Grey, border region). Based on the vsync signal value of 36, we can see that this is the 36th pixel-row from the top of the monitor, and it is being coloured green on the leftmost edge, followed by a light grey border region. The approximate region of the monitor being displayed is outlined in yellow above in Figure 18.

In addition to the required functionality of the game we allowed for scoring, dynamic player adjustment, and dynamic game speed adjustment. We can see from Figure 15 that the players are "medium" in size and the orange square at the top of the monitor is slightly left of center. In Figure 16 the paddles have been adjusted to be longer through the use of a push button, and we can see that the score tracking cube has moved farther left, indicated that the pink player has scored an additional goal. In Figure 17 we can see that the player sizes have been drastically reduced (they are slightly visible at the bottom of the screen). This is the minimum player size. Two other push buttons can be used to increase and decrease the player speed, however this is not visible in images and has been demonstrated during the tutorial period. The general process for the game functionality has been outlined above in Figure 10.

In order to control the players, each user has two switches. One of these switches stops or starts the player motion, the other controls the movement direction. This can be seen above in Figure 5, where SW1 would control movement, and SW0 would control direction. Also in order to control the speed and size functions a counter loop needed to be implemented on each pushbutton in order to not have a single button press cause multiple speed or size changes. This can also be in above in Figure 5, where BTNL, BTNR, BTNU and BTND are the 4 used push-buttons.

Conclusion

In this lab, the process of creating a simple video game and learning the functionality of video-output subsystem and Video Graphic Adapter was accomplished. By following the VGA Design specifications a video adapter was created which allowed for the CPU to represent information on the VGA monitor in the form of a picture. Then by programming the Game's logic, which is tied to the Dynamic elements on the screen, a simple video game based on Atari's pong was created.

There is one slight caveat, the simulation diagrams Figures 11 through 14 are slightly off. The clock of the VGA monitor is actually 25.125 MHz while the simulation is running at a consistent 25MHz. While the difference does not appear to be that large, over time timing diagrams become less and less precise.

The secondary goals of learning and expanding the knowledge of VHDL within the Xilinx ISE CAD environment and gaining experience in the design and implementation of custom logic controllers was also achieved.

References

1. D. A. Patterson, J. L. Hennessy, and P. Alexander, Computer organization and design: the hardware/ software interface. Amsterdam: Morgan Kaufmann, 2015.

2. [Online]. Available: https://www.ee.ryerson.ca/ lkirisch/ele758/labs/SimpleVideoGame [11-11-11].pdf. [Accessed: 25-Nov-2019].