Game development using VGA controller in Basys 3 FPGA

[1]Joel Varughese, [2]S. Ewins Pon Pushpa

[1] PG Student, [2] Assistant Professor, DECE, CEG, Anna University,
Guindy, Chennai, Tamil Nadu,India 600025

Abstract:

This paper deals with the design and implementation of Pong game on Basys3 FPGA development board. Pong game design and implementation unfolds on Basys3 FPGA development board utilizing a VGA controller for display output mercilessly. System built entirely in Verilog HDL leverages capabilities of Xilinx Artix-7 FPGA and gets synthesized using Vivado Design Suite very efficiently. Real-time paddle control and ball movement with collision detection are featured alongside score tracking on a standard VGA monitor. A modular approach was employed dividing project into functional blocks including VGA signal generation object rendering game logic and user input handling rather haphazardly. VGA controller generates synchronization signals and pixel data at 640×480 resolution at 60 Hz enabling smooth visuals quite remarkably. FPGA-based game design showcases educational value through integration of digital design principles with video interfacing and real-time control pretty effectively nowadays. Future enhancements might encompass two-player mode support and sound integration thereby deepening user engagement with increasingly complex graphics.

Keywords: VGA controller; FPGA; Verilog HDL.

1. Introduction

Evolution of digital design and embedded systems enables implementation of quite complex real-time applications on various Field Programmable Gate Arrays fairly rapidly nowadays. Integration of visual interfaces like VGA with interactive digital logic offers a gnarly platform to explore fundamentals of hardware design thoroughly nowadays. Pong game design unfolds on Basys3 FPGA board with Verilog HDL describing hardware and VGA controller driving real-time video output rapidly. Pong provides a straightforward way to grasp key digital design principles such as finite state machines and collision detection through real-time user input effectively.

Its inherent simplicity makes it an excellent candidate for educational hardware projects and experimental setups paired with VGA monitors or display devices. Project development occurs on Basys3 FPGA board utilizing Xilinx Artix-7 FPGA pretty deeply inside its core architecture. Programmable logic cells and onboard peripherals like push buttons are available on this platform alongside block RAM and clocking capabilities. Pong system gets created with paddle controls and ball dynamics alongside collision mechanics scoring and user interface on VGAcompatible monitor. VGA controller generates synchronization signals and pixel coordinates necessary for rendering graphics on screen pretty quickly nowadays.

VGA being an archaic analog video standard necessitates rigorous timing specs pretty much everywhere for a relatively stable image on screen. Display operates at 640x480 pixel resolution and 60 Hz refresh rate necessitating pixel clock frequency of around 25.175 MHz somehow precisely. Designing hardware modules involves wielding precise control over pixel drawing locations and frame buffering amidst stringent horizontal and vertical sync signal timing requirements. System architecture is split into multiple Verilog modules ensuring clarity and reusability in design with fairly complex module interactions. Key modules include VGA Timing Generator which produces hsync and vsync signals along with active video signals and maintains pixel coordinate counters x and y. Game Logic Controller manages motion of ball and paddles rather sloppily while detecting collisions with walls or paddles and updating scores quickly. Input Handling Module interfaces with physical buttons on Basys3 board allowing real-time user control of paddles effectively always.

Renderer draws game elements like ball and paddles pretty quickly based on current pixel location in a rather haphazard manner. Recreating playable Pong and showcasing fundamental digital logic orchestrated into a complete interactive visual system is this project's multifaceted goal somehow. Entire system operates sans microprocessor or software intervention showcasing sheer power of pure hardware design through RTL-level implementation effectively.

This project serves quite valuably as an educational tool for myriad purposes downstream. Students and hobbyists get hands-on experience with gnarly hardware timing issues and modular design sync problems by tinkering with this contraption. Project merits extend far beyond education providing foundation for super advanced FPGA-based designs like bespoke game engines or graphics accelerators and AI-driven decision-making in games. System design and module descriptions are detailed in subsequent sections alongside simulation results and hardware implementation specifics.

2.Literature Review

Several research efforts have explored design and implementation of VGA controllers on FPGA platforms highlighting educational and practical digital system applications recently. Niveditha Yadav and others have collaborated on various projects occasionally with considerable success emerging over time quite mysteriously.(2016) presented a modular algorithm for VGA controller design implemented on an Altera DE2-115 FPGA using Verilog HDL.

A modular algorithm for VGA controller design was presented in 2016 utilizing Verilog HDL on Altera DE2-115 FPGA. Their work centered around 640×480 resolution support emphasizing synchronization logic and scanline management accurately generating VGA signals while showcasing portability and reusability of modules across various FPGA platforms using Quartus II. A rather flexible VGA controller design was explored on Xilinx Virtex-4 FPGA at 2012 International Conference on Electronics and Communication Engineering. It featured support for multiple display resolutions up to 1280×800 and interfaced with system via PLB bus and FIFO-based frame buffer underlining capability for higher-level applications like graphical display with software integration. Fangqin Ying and Xiaoqing Feng remarkably furthered VGA timing comprehension quite extensively in 2012 with VHDL-based controller design on Altera EPIC6Q240C8 platform. Their work simulated and verified a complete controller rendering static images text Chinese characters and color patterns with accurate VGA timing while consuming fairly minimal resources at relatively low power.

Wasu and Wadhankar in 2015 built a VGA controller with Verilog on Altera Cyclone FPGA focusing on real-time simulation heavily.(2017) proposed a more advanced and configurable VGA graphics controller that supports standard resolutions from VGA to WXGA, with built-in resolution selectivity, internal testing, and scalable timing generators. Gujarathi and others have presented some findings recently. A rather advanced VGA graphics controller was proposed in 2017 featuring WXGA support standard resolutions and internal scalability with built-in testing generators somehow. A recent work published in IJARIIE 2023 implemented a basic yet efficient VGA signal generator on Cyclone-II FPGA using VHDL focusing heavily on timing accuracy. These studies collectively highlight a prevailing focus on modularity and synchronization accuracy alongside efficient utilization of FPGA resources in designing VGA controllers. They predominantly focus on static displays or test patterns and fail to integrate complex systems that interact in real time dynamically.

Our work leverages foundational VGA timing and rendering techniques from prior research and embeds a full interactive system deeply into VGA output. This elevates VGA controller's role from display peripheral to core visual engine in real-time hardware-based game environments somehow very significantly. Fig. 1 represents the work flow according to the literature review done.

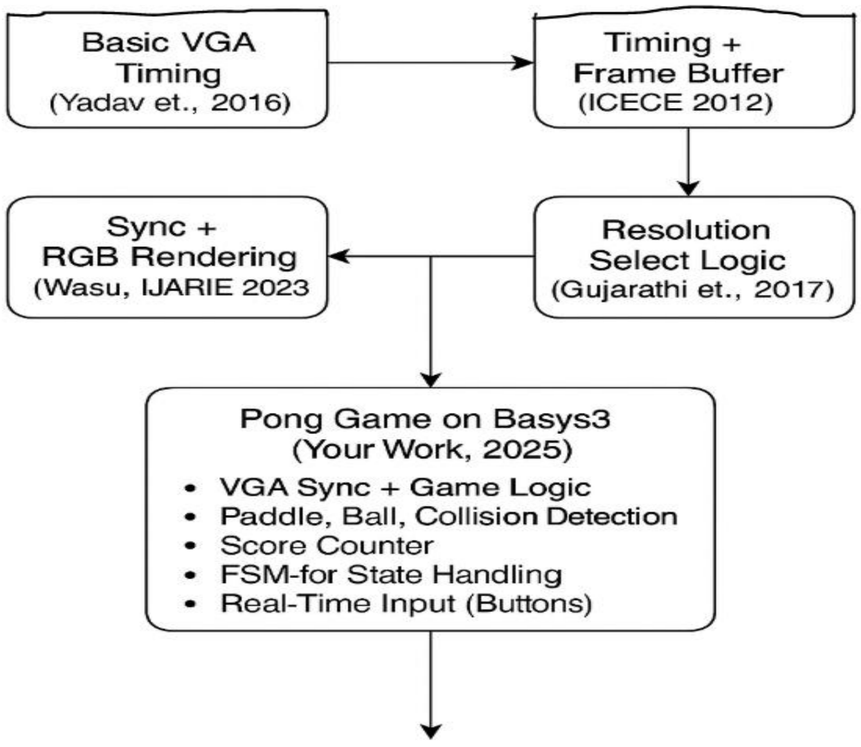
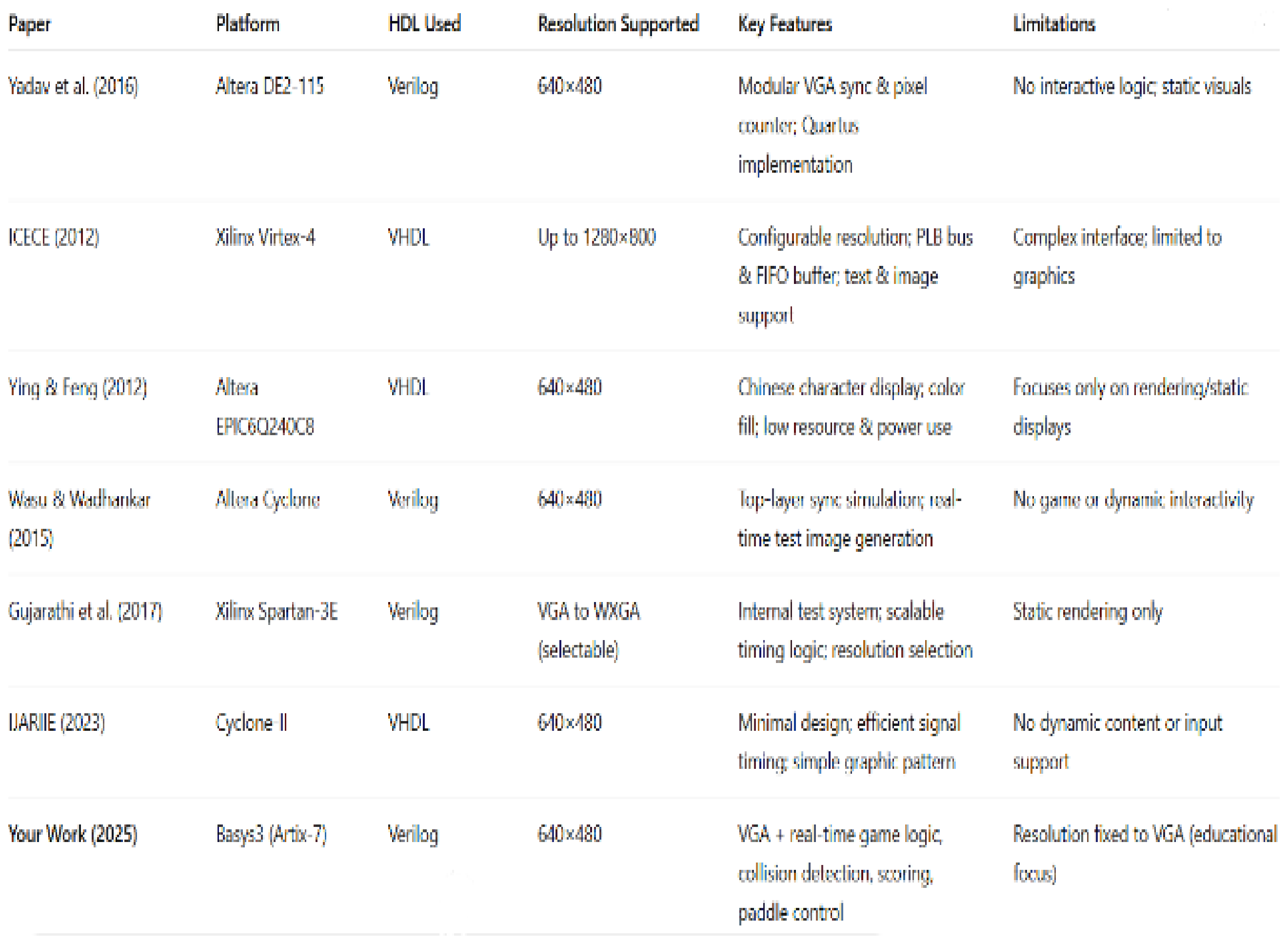

Basic VGA Timing (Yadav et., 2016)
Timing + Frame Buffer (ICECE 2012)
Sync + RGB Rendering (Wasu, IJARIE 2023
Resolution Select Logic (Gujarathi et., 2017)
Pong Game on Basys3 (Your Work, 2025)
• VGA Sync + Game Logic
• Paddle, Ball, Collision Detection
• Score Counter
• FSM-for State Handling
• Real-Time Input (Buttons)

Fig.1 Work flow according to the literature review that is done.



Paper	Platform	HDL Used	Resolution Supported	Key Features	Limitations
Yadav et al. (2016)	Altera DE2-115	Verilog	640×480	Modular VGA sync & pixel counter; Quartus implementation	No interactive logic; static visuals
ICECE (2012)	Xilinx Virtex-4	VHDL	Up to 1280×800	Configurable resolution; PLB bus & FIFO buffer; text & image support	Complex interface; limited to graphics
Ying & Feng (2012)	Altera EPIC6Q240C8	VHDL	640×480	Chinese character display; color fill; low resource & power use	Focuses only on rendering/static displays
Wasu & Wadhankar (2015)	Altera Cyclone	Verilog	640×480	Top-layer sync simulation; real-time test image generation	No game or dynamic interactivity
Gujarathi et al. (2017)	Xilinx Spartan-3E	Verilog	VGA to WXGA (selectable)	Internal test system; scalable timing logic; resolution selection	Static rendering only
IJARIIE (2023)	Cyclone II	VHDL	640×480	Minimal design; efficient signal timing; simple graphic pattern	No dynamic content or input support
Your Work (2025)	Basys3 (Artix-7)	Verilog	640×480	VGA + real-time game logic, collision detection, scoring, paddle control	Resolution fixed to VGA (educational focus)

Table. 1 Comparison Table of Literature on VGA Controller Implementations

Existing literature on VGA controller implementations reveals evolution in design complexity and application scope ultimately yielding quite integrated real-time hardware systems. Niveditha Yadav et al spearhead research endeavors vigorously. (2016), who implemented a VGA controller on the Altera DE2 115 using Verilog, the emphasis was on modular design—specifically separating VGA synchronization and pixel-counting modules—targeted for educational purposes and limited to static pattern rendering at 640×480 resolution.

In 2016 someone implemented a VGA controller using Verilog on Altera DE2-115 with emphasis on modular design specifically separating VGA sync and pixel counting modules. A 2012 ICECE paper built upon Xilinx Virtex 4 platform advancements by incorporating multi-resolution support up to 1280×800 and a FIFO-based frame buffer linked via PLB bus enabling display of graphics and text simultaneously yet still missing real-time interactivity or dynamic input. Fangqin Ying and Xiaoqing Feng added complexity through VHDL-based design in 2012 for Altera platforms and demonstrated display of various image types vividly. Wasu and Wadhankar in 2015 perpetuated this inclination on an Altera Cyclone FPGA focusing heavily on simulation and test pattern generation. (2017) then introduced configurability in resolution selection from VGA to WXGA using a Xilinx Spartan 3E, along with built-in testing systems, making the design scalable and more flexible, yet still bound to static display outputs.

Gujarathi and others have presented findings elsewhere. In 2017 configurability was introduced in resolution selection ranging from VGA to WXGA using a Xilinx Spartan 3E and built-in testing systems made design pretty scalable. A fairly recent VHDL-based design published in IJARIIE 2023 utilized Cyclone-II prioritizing signal timing accuracy for rendering basic shapes thus reinforcing dominance of non-interactive applications heavily in VGA-based research. These works collectively showcase educational and technical feasibility of VGA controller design on various FPGA platforms using mostly Verilog and VHDL. Present Pong game implementation distinguishes itself precisely here with a uniquely quirky yet captivating gameplay style somehow. Using Verilog on Basys3 FPGA it builds upon foundational VGA signal generation techniques and introduces a system blending VGA synchronization with game state logic and real-time paddle movement tightly coupled to frame refresh cycle alongside collision detection score display and responsive control inputs.

2. Methodology

The methodology for implementing the Pong game using a VGA controller on the Basys 3 FPGA board begins with the initialization of the VGA display system. This step involves configuring the VGA controller to generate accurate synchronization signals required for displaying graphics on a standard monitor. On the Basys 3 board, the Artix-7 FPGA is programmed using Verilog or VHDL to create horizontal and vertical sync pulses (HSYNC and VSYNC), which are essential for proper raster scanning on a 640x480 VGA display at 60 Hz. Additionally, a pixel clock—typically set to 25 MHz by dividing the board's 100 MHz master clock—is used to time the scanning of pixels. The VGA controller must also distinguish between the visible display area and the blanking intervals, during which synchronization pulses are transmitted.

Once the VGA system is configured, the next phase involves rendering the game elements on the screen. The ball and paddles, which are the core visual components of the Pong game, are drawn by comparing the current pixel location (x, y) against the coordinates of each object. If the pixel falls within a defined region of the ball or a paddle, the FPGA outputs a specific RGB colour value for that pixel, making the object appear on screen. The central dividing line (net) may also be added for aesthetics and gameplay orientation. This entire rendering process is synchronized with the VGA timing to ensure smooth visual output.
The game logic then proceeds to dynamically update the positions of the ball and the paddles. The ball's movement is handled by incrementing or decrementing its x and y coordinates based on predefined direction variables. Paddle movement, on the other hand, is controlled by user inputs received through the Basys 3's push buttons or switches. These inputs are processed and debounced in hardware to ensure clean, responsive control, and the paddle positions are adjusted accordingly. Collision detection plays a crucial role in determining the behavior of the game elements. The FPGA continuously checks whether the ball has reached the screen boundaries or made contact with the paddles. If the ball hits the top or bottom edge, its vertical direction is inverted. If it hits a paddle, its horizontal direction is reversed, simulating a bounce. Failure to intercept the ball with a paddle—resulting in the ball reaching the left or right edge—triggers a scoring event for the opposing player.

Finally, when a score is registered, the updated score is displayed using the 7-segment display on the Basys 3 board. This hardware element shows the current points for each player in real time. The system may include a brief pause or visual feedback before restarting play, allowing users to view the result of the last round. This final phase ensures interactive gameplay with continuous feedback, making the system a complete hardware-based video game solution. The below Figure 2 represents the methodology for implementing a pong game in Basys3.

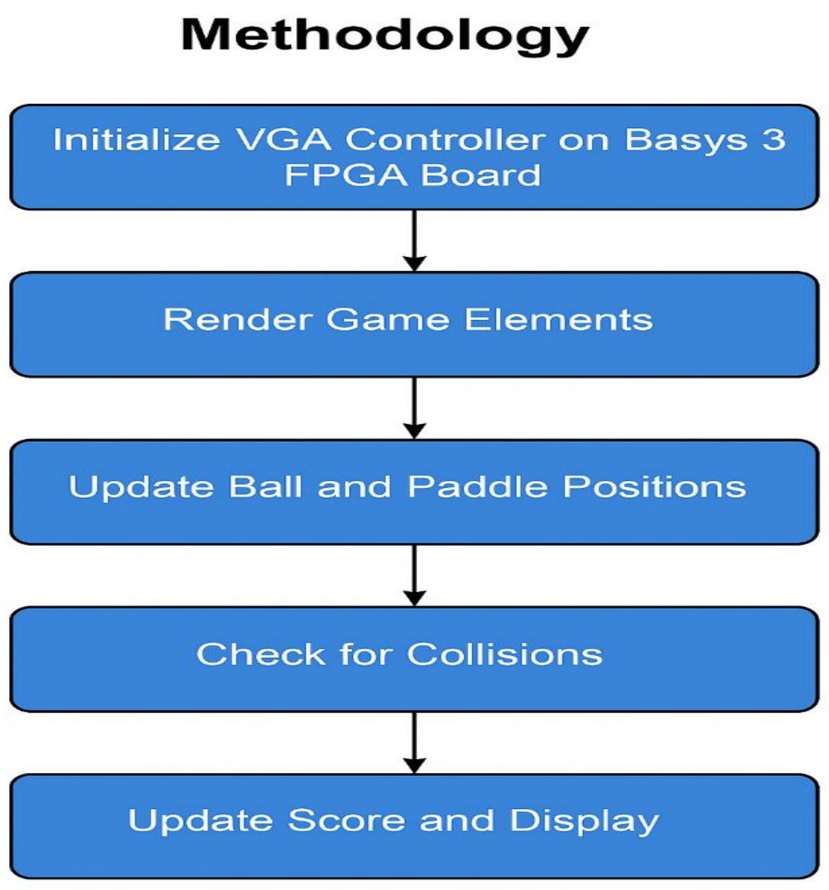

Methodology
Initialize VGA Controller on Basys 3 FPGA Board
Render Game Elements
Update Ball and Paddle Positions
Check for Collisions
Update Score and Display

Fig. 2 Methodology for implementing a pong game in Basys3

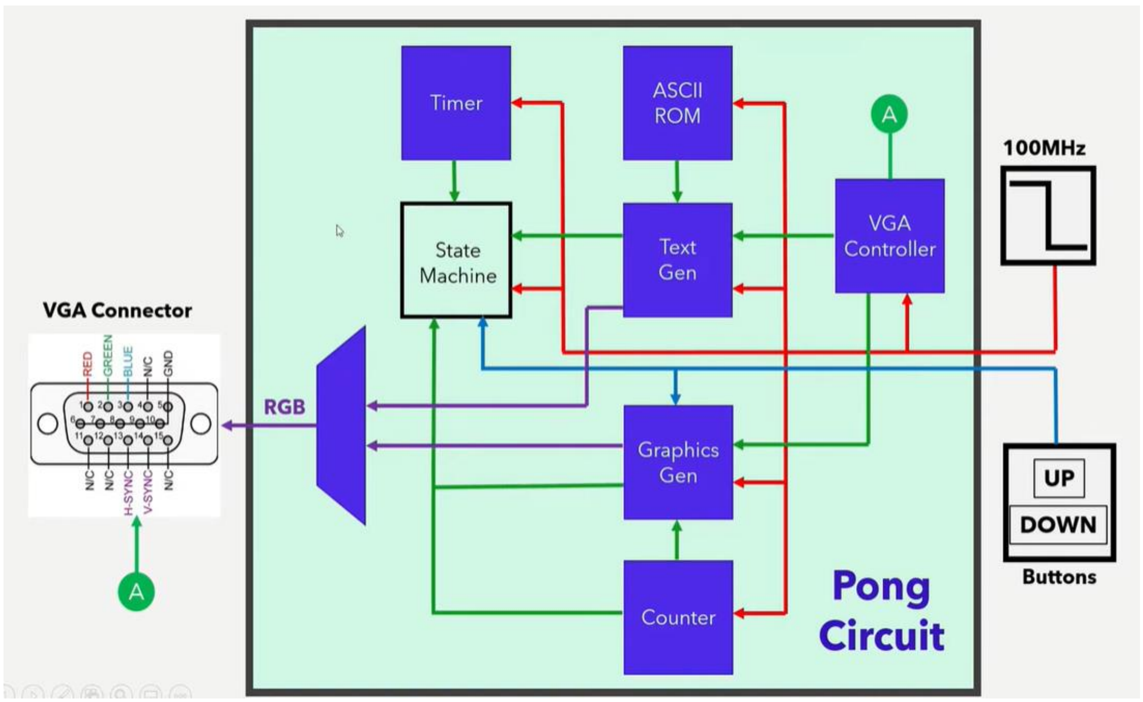


Fig. 3 Overall Block Diagram of the Pong game in Basys3

Fig 3. depicts the overall block diagram for the designed pong game. A block diagram illustrates an architecture implementing classic Pong using VGA controller on Basys3 FPGA board effectively. A VGA controller sits at core of system responsible for generating precise horizontal and vertical sync signals needed for displaying graphics on standard VGA monitor. It operates using a 100 MHz clock signal from FPGA and produces precise timing necessary somewhat awkwardly to drive display resolutions like 640x480 pixels at 60 Hz. Game graphics are drawn correctly on screen in each successive frame. State machine serves as control unit governing entire game logic pretty much.

It dictates precisely when various operations happen including moving paddles rapidly updating ball's position suddenly and detecting numerous collisions effectively between distinct game states. State machine interacts with almost every block in circuit orchestrating behavior based on current conditions and player inputs very dynamically. Periodic signals arrive from a timer that regulates frequency of game state updates pretty regularly. This timer ensures consistent pacing of game by controlling speed at which ball moves swiftly and refreshing game logic fairly frequently. Paddle movement gets controlled via UP and DOWN inputs provided by push buttons awkwardly situated on Basys3 board.

Button signals get processed quickly and get dispatched off to a state machine which interprets them and tells graphics generator update paddle positions. Graphics generator draws ball and paddles along with score boundaries or other elements with coordinates provided by some counter module rapidly. Counter tracks timing and position data synchronizing game element updates rapidly with VGA frame refresh cycle pretty accurately apparently. Design incorporates an ASCII ROM and text generator effectively displaying scores or game messages rapidly within game environments. ASCII ROM stores bitmap representations of characters while text generator utilizes this data rendering text elements like player scores or cryptic status messages. Game graphics or textual information gets displayed at each pixel based on current screen location and game state via a multiplexer.

Finally, all visual data including color values and synchronization pulses are routed via circuitry to a VGA connector which drives an external monitor. This connector includes dedicated pins for red green and blue color signals and horizontal and vertical sync pulses. FPGA logic hosts a seamless gaming experience via state machine synchronization and clock where each module plays its specific role effectively.

3. Code - working Principle.

Game.vhd file implements Pong game on Basys3 FPGA with VHDL and VGA controller driving visual output on display pretty smoothly. Several interconnected logic components and processes handling user input object movement collision detection and visual rendering make up this design. Game accepts inputs like Player1_up and Player2_down which vertically manipulate paddles of each player along screen pretty freely. A key part of logic involves finite state machine and timing mechanisms synchronized with system clock against VGA display refresh rate.

VGA interface gets managed via counters horizontal and vertical that reckon active pixel and line positions generating sync pulses and drawing game objects. Paddle and ball positions update rapidly based on user inputs and internal velocity registers while collision detection logic kicks in. Scoring logic resets ball at center and updates player scores quickly whenever ball crosses left or right boundary lines suddenly. Output video signals red green blue hsync and vsync are precisely timed complying with VGA 640×480 standard allowing smooth display of various game elements. Modularity and clarity are heavily emphasized in design which demonstrates successful real-time game implementation using FPGA and novel hardware description techniques effectively.

4. Results and Discussion.

The game has been developed in three stages and each stage is more complicated than the previous stage. Before getting into that let’s first look at the RTL schematic for the VGA interface. The figure 4 below shows the schematic.

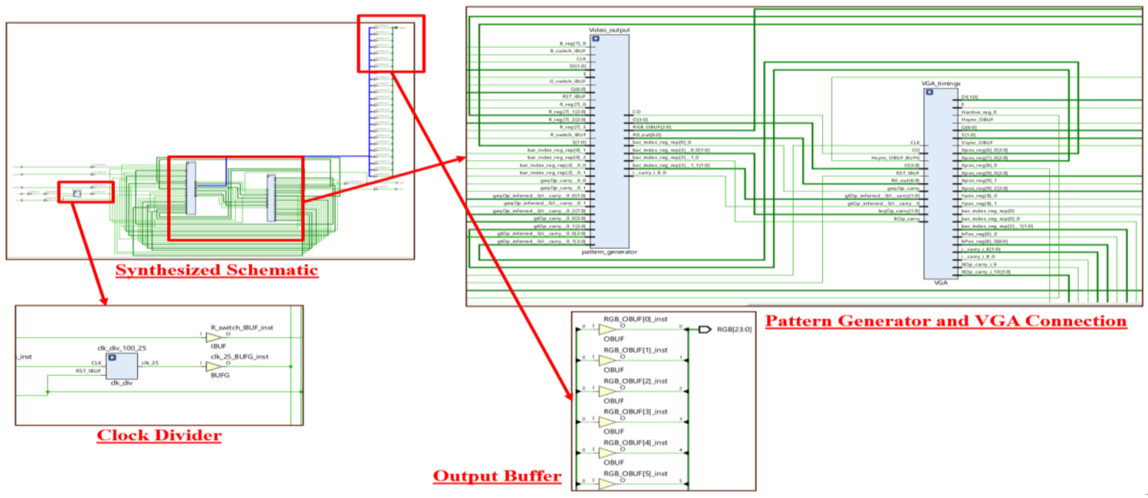


Fig. 4 RTL Schematic for the VGA Interface

Schematic illustrates top-level hardware architecture of Pong game implementation on FPGA focusing heavily on VGA output pipeline functionality somehow. Synthesized schematic lies at core of design integrating all lower-level modules such as pattern generator and VGA sync logic quite neatly. Clock divider module reduces system primary clock typically 100 MHz on Basys3 down rather slowly to a 25 MHz signal fitting standard 640×480 VGA timing. Divided clock feeds into main pattern generator and VGA connection block containing pixel rendering logic alongside VGA timing generator circuitry.

Various inputs such as paddle positions and ball coordinates are utilized inside this module generating RGB color outputs based on active screen region while hsync and vsync signals get managed according to VGA protocol specifications. Video output signals pass through output buffer represented as RGB_OUTPUT where individual bits of red green and blue are routed via OBUFs to physical FPGA output pins. Signals from graphics cards feed actual VGA connectors directly on most computer systems nowadays. Schematic represents a clean modular flow from system clock input to VGA video signal generation pretty effectively embodying a classic rendering pipeline suitable for retro-style games implemented in digital logic on FPGA.

Stage 1 of the project implementation of the pong game has a single paddle and a wall. The ball bounces between the paddle and the wall and the user controls only one paddle and moves it up and down to catch the ball. In this stage a counter is not used and the score is not kept and the game ends as soon as the paddle misses the ball. The bellow figure 5 shows us the said implementation.

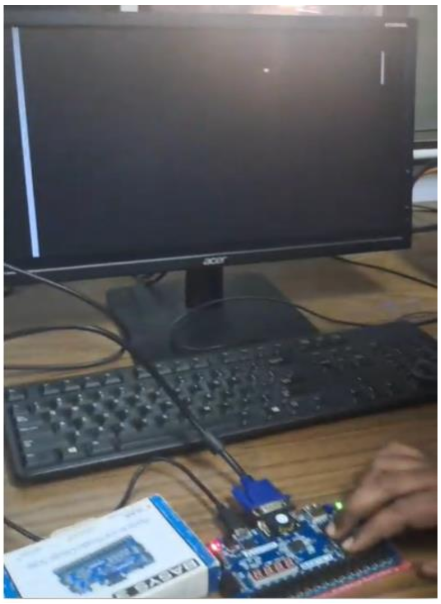


Fig. 5 Stage 1 output shown on the monitor.

` In stage 2 the game is improved more complex by introducing an extra level. Initially, when the game starts, it starts with level 1 and then as the player reaches a score of 5, a new level is reached where the ball moves faster and the game becomes slightly more difficult. A counter has been introduced to keep track of the score and this is also important for the program to know when the level is changed. Also, the player is given 3 lives and the number of lives left is also shown in the monitor. The bellow figure 6 shows how the stage 2 output is seen on the monitor.

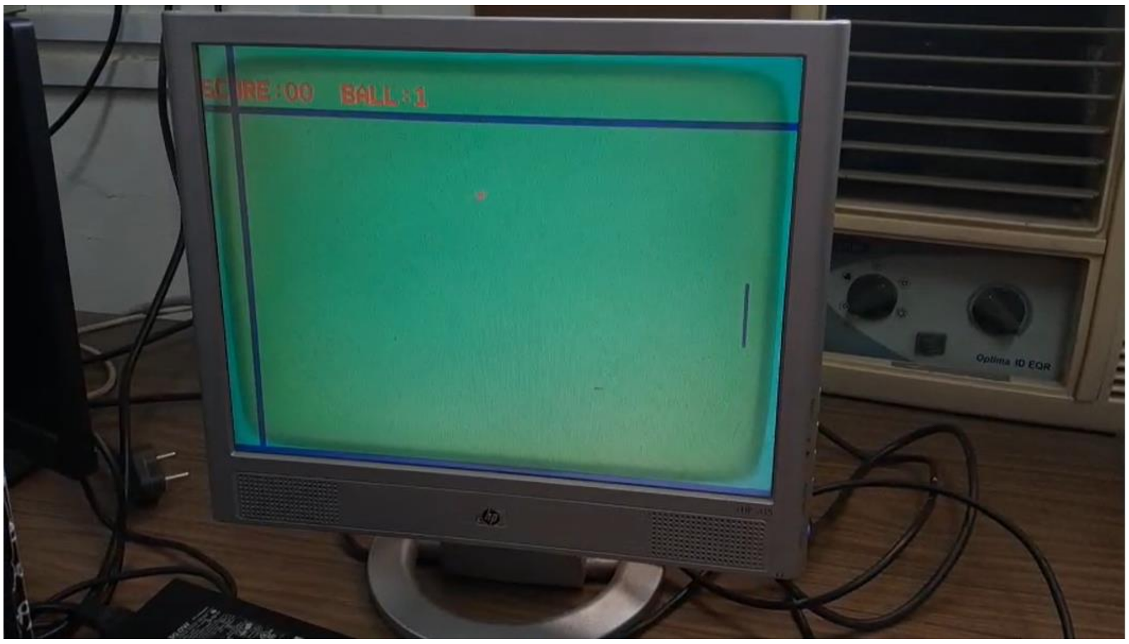


Fig. 6 Stage 2 output shown in the monitor.

For stage 3 introduces an extra player. Here there are two players playing against each other. Each player is assigned with two buttons on the basys3 FPGA. The player who reaches the score of 10 wins. The score of each player is displayed in their respective colors in the top middle portion of the window being displayed. The figure 7, bellow shows how the basys3 displays the output of stage 3 on the monitor.

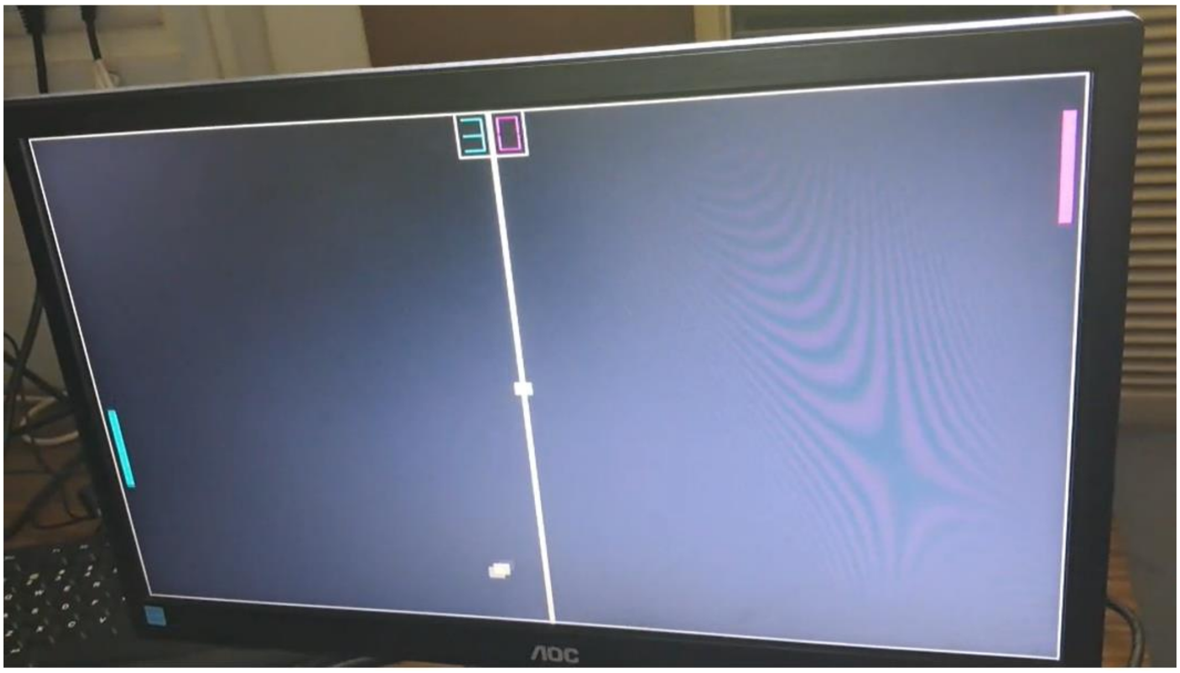


Fig. 7 Stage 3 output shown in the monitor.

5. Conclusion and Future Scope.

Successfully demonstrating capability of FPGAs in handling real-time video generation and interactive digital system design Pong game was implemented using VGA controller on Basys 3 FPGA board. Pong gameplay gets recreated entirely in hardware via integration of modules like VGA controller state machine graphics generator and input interface. Verilog and VHDL facilitate design of systems interacting with human input and display hardware rapidly in real-time with considerable effectiveness. This project reinforces understanding of digital logic design and finite state machines while providing hands-on experience with quirky embedded game logic.

FPGA-based designs achieve unruly efficiency and parallelism through synchronization of various blocks pretty seamlessly underneath. This project lays foundational groundwork quite heavily for rather advanced graphical applications based on FPGA and also somewhat intricate gaming. Several enhancements might be seamlessly integrated later increasing functionality significantly and adding layers of intricacy simultaneously. Pong game functionality expands quite irregularly with multiple difficulty levels and AI opponents for two-player mode using onboard audio modules. Support for higher VGA resolutions and sophisticated graphics could be implemented by modifying VGA timing and utilizing multiple image buffers effectively. Exploring integration with external memory modules or HDMI interfaces could enable broader display compatibility pretty effectively nowadays. This architecture lends itself nicely beyond just some simple game to educational tools that teach digital design and embedded systems pretty effectively.

Pong game using VGA controller on Basys3 FPGA has extensive future scope offering multiple paths for academic enhancement and significant technical advancement. Expanding this project involves introducing multiplayer or networked gameplay via protocols such as UART or Bluetooth and Wi-Fi quite rapidly. Advanced timing and encoding techniques are utilized when implementing higher-resolution VGA or HDMI support involving rather complex processes.

Design integrates rather seamlessly with soft-core processors such as MicroBlaze or RISC-V enabling hybrid hardware-software systems and embedded C programming somehow. Game logic gets enriched with quirky features like power-ups or AI-controlled paddles and difficulty levels introducing complex state machines quite dynamically. On-chip memory utilization for framebuffers enables smoother animations and graphical enhancements while hardware acceleration for physics introduces pipelining very effectively. Optimizing for low power and resource utilization aligns nicely with fundamental VLSI design principles and certain ASIC methodologies perfectly. Lastly this project can morph into a broader educational game platform supporting multiple retro games and making it quite ideal for teaching digital systems embedded design and complex real-time graphics processing with FPGA tech.

6. References.

1. Algorithm to Design VGA Controller on FPGA Board Niveditha Yadav M, Yaseen Basha , Rohith. S , Venkateshkumar.H. IOSR Journal of VLSI and Signal Processing (IOSR-JVSP) 2016.
2. FPGA IMPLEMENTATION OF VGA CONTROLLER January 2012 Conference: International Conference on Electronics and Communication Engineering (ICECE) -16 th Sept, 2012, Pune.
3. Design and Implementation of VGA Controller Using FPGA Fangqin Ying, Xiaoqing Feng. 2002.
4. R. Wasu, V. Wadhankar, "Design and Implementation of VGA Controller on FPGA," *IJIRCCE*, vol.3, no.7, Jul 2015.
5. Gujarathi et al., "FPGA-Based Graphics Controller," *IJERT*, 2017.
6. *FPGA Based VGA Signal Generation*, *IJARIIE*, 2023.