Labyrinth Game on Altera DE2-115 FPGA architecture using VHDL

Jills Pulickakunnel Manuvel
Student ID: 33060728
Engineering and Mathematics Department
Sheffield Hallam University
Sheffield, United Kingdom
C33060728@hallam.shu.ac.uk

***Abstract*- The popularity of Field-Programmable Gate Array (FPGA) technology is growing among designers specializing in Application-Specific Integrated Circuits (ASIC). FPGAs are favoured for their ease of development and maintenance, making them an attractive choice for applications where speed and efficiency are crucial. This project focuses on exploring the capabilities of FPGAs by developing a simple labyrinth game with a VGA display. The entire implementation was carried out on a DE2-115 development board, utilizing the VHDL hardware description language. For the implementation of the game the most important task is its VHDL programming. Labyrinth game is typically a classic pong game in a closed maze. It is a two-player game, where each player tries to thit a ball towards the opponent using paddles. Also, several fundamental functionalities, including enhancing the ball's speed, generating sound, and displaying textual information, have been incorporated. Game controlling was implemented through four keys in a PS-2 keyboard and the resolution of the monitor also must be set at 640 x 480 to meet the requirement.**

Keywords- ASIC, FPGA, Labyrinth Game, Altera DE2-115, VHDL,* Quartus Prime*, Cyclone-IV, VGA

I. INTRODUCTION

Nowadays FPGAs are used to implement digital logic circuits and are highly flexible compared to traditional Application-Specific Integrated Circuits (ASICs) because they can be reprogrammed for different tasks. The attractiveness of FPGA lies in its ease of creation and maintenance, making it a compelling solution for high-speed and efficient applications. An FPGA typically contains a matrix of programmable elements, also known as, Configurable Logic Blocks (CLBs). CLBs contain Look Up Tables (LUTs), that can be used as logic or storage elements. The configuration data is stored in the memory. There are various models of FPGA available in the market, in which the two biggest manufacturers are Xilinx and Altera. The project implemented an Altera DE2-115 FPGA development board with the usage of VHDL as the hardware description language.

The primary purpose of this project is to explore the world of FPGA by developing a simple labyrinth game. It is basically two player Ping Pong game in a closed field. The all rules and conditions of labyrinth game are same as that of ping pong game. The system design was mainly focused on hardware and software design. The hardware part involved the host computer for design configuring and programming, FPGA for game operating as well as the VGA monitor for display. The software design included the design of overall system, input keyboard controller module, VGA display module, and the game control module.

II. LITERATURE REVIEW

2.1 Overview

The Fourth Industrial Revolution emphasizes the transformative role of technologies in driving societal digitization, organization, and enhanced manufacturing productivity. Within the realm of electronics, the Integrated Circuit (IC) stands out as a key technology, contributing to the reduction in the size of electronic products through heightened logic gate density per chip. Certain ICs are adaptable and reprogrammable for various applications, while others, categorized as Application Specific Integrated Circuits (ASICs), are tailored for specific and dedicated uses. ASIC is favoured over general-purpose chips because of its compact size, rapid response, and low energy consumption, making it well-suited for complex larger systems. Its distinctive features make it a preferred choice for high-level applications, commonly deployed in private data centres, public clouds, and shared devices globally. Categorization of ASIC is demonstrated in Fig.1

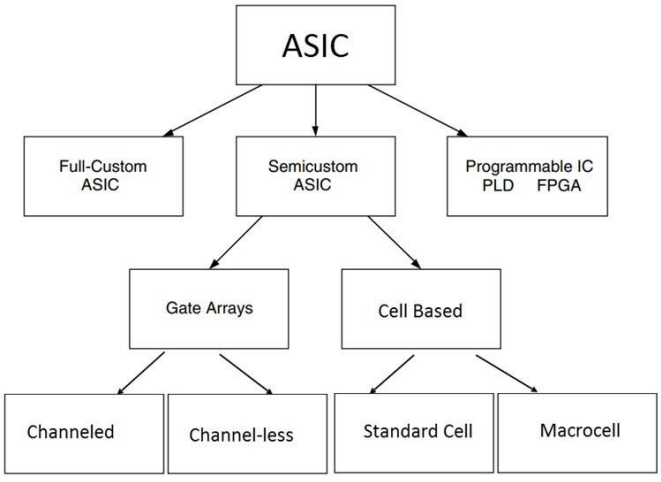

Fig. 1 Categories of ASIC

In full custom design, each interconnection mask layer is tailored to specific requirements, while semi-custom design involves partial customization of masks with the rest drawn from a pre-designed library. Chips designed with full customization provide minimal flexibility for programming since they lack the ability to be re-programmed or modified after the manufacturing process. This challenge is notably addressed by using Programmable ASIC, known as a Programmable Logic Device (PLD). PLD is an integrated circuit with many flip-flops and gates that can be configured with basic software to execute the logic for a specific function or perform a complex logic function. It offers significant flexibility throughout the design cycle, allowing for programming or reconfiguration to define functions based on design considerations. This stands in contrast to logic gates, which have fixed functions and lack the adaptability provided by PLDs. There are three types of PLD, namely Simple PLD (SPLD), Complex PLD (CPLD), and Field Programmable Gate Array (FPGA).

2.2 Introduction to FPGA

Field Programmable Gate Arrays (FPGAs) are silicon devices manufactured in advance, capable of being electronically programmed on-site to transform into virtually any type of digital circuit or system. In low to medium volume productions, FPGAs offer a more cost-effective solution and a quicker time to market compared to ASICs, which typically demand substantial resources in terms of time and money for the initial device acquisition.

The primary advantage of FPGAs, namely their flexibility, is also the primary factor contributing to their drawbacks. Their flexible nature results in FPGAs being notably larger, slower, and more power-consuming compared to ASICs. These disadvantages largely stem from the programmable routing interconnect of FPGAs, which constitutes almost 90% of their total area. However, despite these drawbacks, FPGAs remain an attractive option for digital system implementation due to their faster time to market and lower costs for low-volume production. Normally FPGAs comprise of components like Programmable logic blocks which implement logic functions, Programmable routing that connects these logic functions. I/O blocks that are connected to logic blocks through routing interconnect and that make off-chip connections. FPGA provides high speeds and a range of capacities, making it applicable to random logic, device controllers, filtering and encoding of communication, multiple SPLDs integration, and small to medium systems with SRAM blocks.

2.3 FPGA programming using VHDL

VHDL is a programming language utilized to delineate a logic circuit based on its function, data flow behaviour, and/or structure. This hardware description serves the purpose of configuring a programmable logic device (PLD), such as a field programmable gate array (FPGA), to implement a tailored logic design. The fundamental structure of a VHDL program revolves around the notion of BLOCKS, which serve as the foundational building units of a VHDL design. Within these design blocks, it is straightforward to articulate the description of a logic circuit or function.

The commencement of a VHDL design involves an ENTITY block, outlining the design's interface. This interface specifies the input and output logic signals for the intended circuit. The subsequent ARCHITECTURE block provides a depiction of the internal functioning of the design. These blocks encapsulate various other functional units employed in constructing the elements of the logic circuit being developed. The structure of VHDL programming is demonstrated in Fig. 2

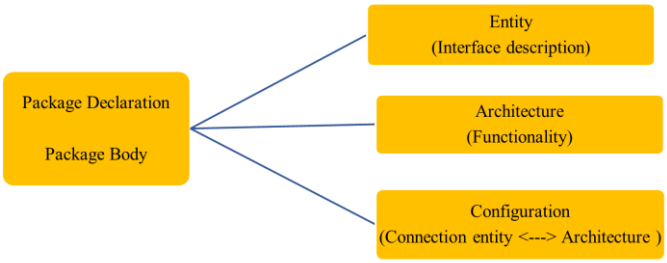

Fig. 2 The structure of VHDL programming

Primarily, VHDL finds application in the creation of Application Specific Integrated Circuits (ASICs). Tools facilitating the automated conversion of VHDL code into a gate-level net list were established quite early in its development. This conversion process, known as synthesis, is a crucial component of contemporary design workflows.

VHDL is primarily utilized in the development of Application-Specific Integrated Circuits (ASICs). Early in its evolution, tools were introduced to automatically transform VHDL code into gate-level net lists. This transformation, referred to as synthesis, is an integral step in modern design processes. VHDL can be employed to model system behaviour without being tied to a specific technology. This capability is valuable for creating standardized solutions, such as for microcontrollers or error correction decoders. Additionally, behavioural models of microprocessors and RAM devices are utilized to simulate a new device within its intended environment.

III DESIGN AND DEVELOPMENT

In this project, a complex FPGA board, Altera DE2-115 from Terasic, is implemented. The board composes a set of Cyclone-IV FPGA, SRAM, SDRAM, Flash memories, seven-segment displays, Light Emitting Diodes (LEDs), switches, pushbuttons, and LCD screen.

The game consists of three core parts, as shown in the block diagram shown in Fig.3 clearly explain the related hardware and the corresponding function of these parts.

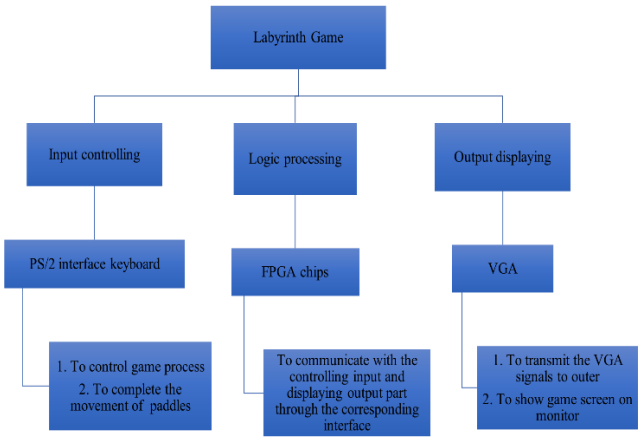

Fig. 3 Categorization of the main body of the Labyrinth game

3.1 Port declaration

The system VHDL is coded, and the entity ports description is labelled as Fig.4. It portrays that the entity "LabyrinthGame" consists of three inputs and five outputs

```
entity LabyrinthGame is
    Port (
      clk : in STD_LOGIC;
      hsync_out : out STD_LOGIC;
      vsync_out : out STD_LOGIC;
      vga_clk : out STD_LOGIC;
      red_out : out STD_LOGIC;
      green_out : out STD_LOGIC;
      blue_out : out STD_LOGIC;
      soundPin : buffer STD_LOGIC := '0';
      kb_clk : in STD_LOGIC;
      kb_data : in STD_LOGIC
   );
end LabyrinthGame ;
```

Fig. 4 Input – Output declaration

clk: Clock input signal - The main clock signal used to synchronize the operations in the design.
kb_clk: Keyboard clock input - A clock signal specifically for the keyboard controller.
kb_data: Keyboard data input - Data from the keyboard, likely representing key presses.
hsync_out: Horizontal synchronization output - This signal is used to synchronize the horizontal timing of the VGA display
vsync_out: Vertical synchronization output - This signal is used to synchronize the vertical timing of the VGA display.
vga_clk: VGA clock output - A clock signal for the VGA display, usually derived from the main clock.
red_out, **green_out**, **blue_out**: RGB colour outputs - These signals determine the colour of each pixel on the VGA display.
soundPin: Sound output - This is likely connected to a speaker or an audio output to produce sound. This port allows both reading and writing to the **soundPin**. It's commonly used for bidirectional communication.

3.2 Input Controlling

The game control was implemented using a PS2 keyboard which is interfaced with FPGA board. Keys A and Z are used for the left paddle movement. Meanwhile keys K and M are used for the right paddle movement. Components of input controlling is shown in Fig.5. Following ports are declared in component "**KeyboardController"**

```
component KeyboardController is
    Port ( Clock : in STD_LOGIC;
           KeyboardClock : in  STD_LOGIC;
           KeyboardData : in  STD_LOGIC;
           LeftPaddleDirection : out  integer;
           RightPaddleDirection : out  integer
   );
end component;
```

Fig. 5 Input – Output declaration of "KeyboardController"

Clock: This is the main clock signal that synchronizes the operations of the keyboard controller module with the rest of the system.
KeyboardClock: This signal is likely a clock signal specific to the keyboard interface. It's used to synchronize the communication between the FPGA and the keyboard.
KeyboardData: This signal carries data from the keyboard. It's likely a serial data stream representing the keys being pressed or released.
LeftPaddleDirection and RightPaddleDirection: This is an output signal that indicates the direction in which the left/right paddle should move. It's likely an integer value representing the direction of movement (e.g. up or down).

3.3 Clock Timing Processes

For the smooth working of the game logic, synchronization of ball and paddle movements is essential. For that **ballMovementClockScaler** and **paddleMovementClockScaler** processes are initiated. Both processes use counters to keep track of the number of clock cycles. The clock signals (**ballMovementClock** and **paddleMovementClock**) are toggled when the counters reach specific values and these toggled clock signals control the timing of specific operations related to ball and paddle movements.

3.4 VGA synchronization

To extract the game output in a VGA screen, processes **signalTiming** and **vgaSync** are used. Declaration of both processes are shown in Fig.6 and Fig.7 respectively. Both processes are process is sensitive to the rising edge of the **halfClock** signal. If the **horizontalPosition** is within the front porch region (between 0 and 97), the **hsyncEnable** is set to '0', indicating the horizontal synchronization period. Otherwise, it is set to '1'. If the **verticalPosition** is within the front porch region (between 0 and 3), the **vsyncEnable** is set to '0', indicating the vertical synchronization period. Otherwise, it is set to '1'.

```
signalTiming : process(halfClock)
begin
    if halfClock'event and halfClock = '1' then
        if horizontalPosition = 800 then
            horizontalPosition <= 0;
            verticalPosition <= verticalPosition + 1;

            if verticalPosition = 521 then
                verticalPosition <= 0;
            else
                verticalPosition <= verticalPosition + 1;
            end if;
        else
            horizontalPosition <= horizontalPosition + 1;
        end if;
    end if;
end process signalTiming;
```

Fig. 6 Components of process **"signalTiming"**

```
vgaSync : process(halfClock, horizontalPosition, verticalPosition)
begin
    if halfClock'event and halfClock = '1' then
        if horizontalPosition > 0 and horizontalPosition < 97 then
            hsyncEnable <= '0';
        else
            hsyncEnable <= '1';
        end if;

        if verticalPosition > 0 and verticalPosition < 3 then
            vsyncEnable <= '0';
        else
            vsyncEnable <= '1';
        end if;
    end if;
end process vgaSync;
```

Fig. 7 Components of process **"vgaSync"**

3.5 Game Logic

The game follows the basic Pong gameplay mechanics, including paddle movement, ball movement, collisions, scoring, and game over conditions. The VGA display is synchronized to visually represent the game, and sound is generated based on specific events. Keyboard input is used to control the movement of the paddles. The game continues until one of the players runs out of lives, at which point the game over state is reached. Various signals are declared to manage the game state, positions, speeds, and other parameters

3.5.1 Paddle movement

The **leftPaddleMovement** and **rightPaddleMovement** processes handle the movement of the left and right paddles, respectively. Paddle movements are controlled by the **paddleMovementClock**, and the direction of movement is determined by keyboard input (**leftPaddleDirection** and **rightPaddleDirection**). Implementation of both processes are shown in Fig. 8.

```
leftPaddleMovement : process(paddleMovementClock)
begin
    if paddleMovementClock'event and paddleMovementClock = '1' then
        if leftPaddleY + leftPaddleDirection < paddleBottomLimit - paddleHalfHeight
            and leftPaddleY + leftPaddleDirection > paddleTopLimit + paddleHalfHeight then
            leftPaddleY <= leftPaddleY + leftPaddleDirection;
        end if;
    end if;
end process leftPaddleMovement;

rightPaddleMovement : process(paddleMovementClock)
begin
    if paddleMovementClock'event and paddleMovementClock = '1' then
        if rightPaddleY + rightPaddleDirection < paddleBottomLimit - paddleHalfHeight
            and rightPaddleY + rightPaddleDirection > paddleTopLimit + paddleHalfHeight ther
            rightPaddleY <= rightPaddleY + rightPaddleDirection;
        end if;
    end if;
end process rightPaddleMovement;
```

Fig. 8 The **leftPaddleMovement** and **rightPaddleMovement**

3.5.2 Ball Movement

The **ballMovement** process updates the position of the ball based on its speed and direction. Collisions with paddles are detected, and the ball's speed is adjusted accordingly. If the ball reaches the left or right boundary, the corresponding player loses a life, and the ball is reset to the centre. **ballSpeedX** and **ballSpeedY** represent the speed of the ball in the X and Y directions, respectively. This process is demonstrated in Fig. 9.

```
ballMovement : process(ballMovementClock,gameOver,soundPlingCounter)
begin
    if gameOver = '1' then
        ballX <= 319;
        ballY <= 239;
        ballSpeedX <= 0;
        ballSpeedY <= 0;
    elsif soundPlingCounter >= 10 then
        playSound <= '0';
    elsif ballMovementClock'event and ballMovementClock='1' then
        if resetBall = '1' then
            if resetCounter = 100 then
                resetCounter <= 0;
                ballX <= 319;
                ballY <= 239;
                resetBall <= '0';
            else
                resetCounter <= resetCounter + 1;
            end if;
        else

            if ballX+4 > rightPaddleFrontX and ballX < rightPaddleBackX
                and ballY+4 > rightPaddleY-paddleHalfHeight and ballY-4 < rightPaddleY+paddleHalfHeight ther
                ballX <= rightPaddleFrontX - 4;
                ballSpeedY <= (ballY - rightPaddleY) / 8;
                ballSpeedX <= -ballMaxSpeed + ballSpeedY;
                playSound  <= '1';
            elsif ballX-4 < leftPaddleFrontX and ballX > leftPaddleBackX
                and ballY+4 > leftPaddleY-paddleHalfHeight and ballY-4 < leftPaddleY+paddleHalfHeight then
                ballX <= leftPaddleFrontX + 4;
                ballSpeedY <= ((ballY - leftPaddleY) / 8);
                ballSpeedX <= ballMaxSpeed - ballSpeedY;
                playSound  <= '1';
            elsif ballX + ballSpeedX < 4 then
                leftLifes <= leftLifes - 1;
                ballX <= -20;
                ballY <= -20;
                resetBall <= '1';
            elsif ballX + ballSpeedX > 635 then
                rightLifes <= rightLifes - 1;
                ballX <= -20;
                ballY <= -20;
                resetBall <= '1';
            else
                ballX <= ballX + ballSpeedX;
            end if;

            if ballY > 470 then
                ballY <= 470;
                ballSpeedY <= -ballSpeedY;
                playSound  <= '1';
            elsif ballY < 10 then
                ballY <= 10;
                ballSpeedY <= -ballSpeedY;
                playSound  <= '1';
            else
                ballY <= ballY + ballSpeedY;
            end if;
        end if;
    end if;
end process ballMovement;
```

Fig. 9 Components of process **"ballMovement"**

3.5.3 Game Over and Lifes

The game has a finite number of lives for each player (**leftLifes** and **rightLifes**). When a player misses the ball, their life count is decremented. The **finishGame** process checks if either player has run out of lives. If a player loses all lives (**leftLifes** or **rightLifes** becomes zero), the game is considered over (**gameOver** is set to '0'). In that time the display will shows the text "GAME OVER". The process for executing following conditions is shown in Fig.10 and VGA display corresponding to this condition is illustrated in Fig.11

```
finishGame : process(leftLifes, rightLifes)
begin
if leftLifes = 0 or rightLifes = 0 then
    gameOver <= '1'; -- Game over
else
    gameOver <= '0'; -- Continue the game
 end if;
end process finishGame;

colorSetter : process(photonX, photonY, halfClock)
begin
        Paddle handling
```

Fig. 10 Game Over process declaration

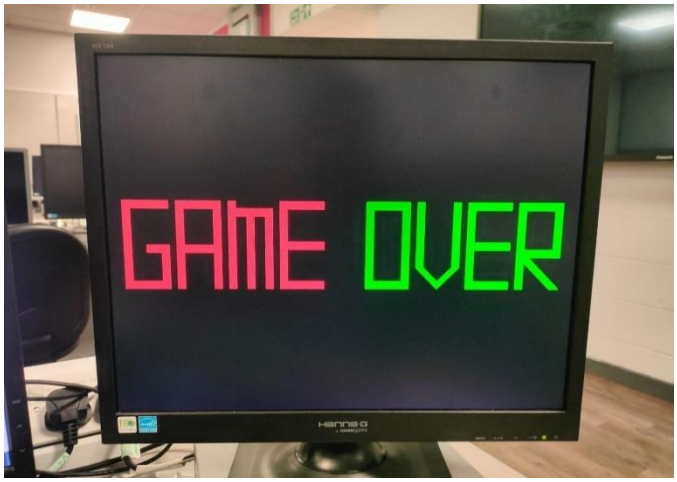

Fig 11. Game Over on VGA screen

IV EXPERIMENTAL PLANNING

4.1 Overview

Altera-Quartus is used as the CAD tool for the implementation of VHDL on the DE2-115 Board. Once the Labyrinth game design was coded, the next step is creating a new project in Quartus prime and add the Labyrinth game VHDL description into the new project. Before compilation and verification, the user must select the model of the device which suits the real FPGA board used. Next, syntax verification is required to remove syntax errors, ensuring that the project can later be compiled successfully. After checking the syntax, the compilation must be done to generate the circuit netlist. Then through the Quartus-Pin Planner, pins assignment is done by assigning the circuit ports to the physically FPGA pins on the DE2-115 board according to their respective functions as taking the DE2-115 hardware design manual as reference. Hence, the circuit is computed, and a file of binary configuration is created for the Cyclone-IV FPGA. Lastly, the final step is to configure the Cyclone-IV FPGA by completing the setup of USB-Blaster and the chosen binary configuration file. The entire system implementation is given in Fig 12.

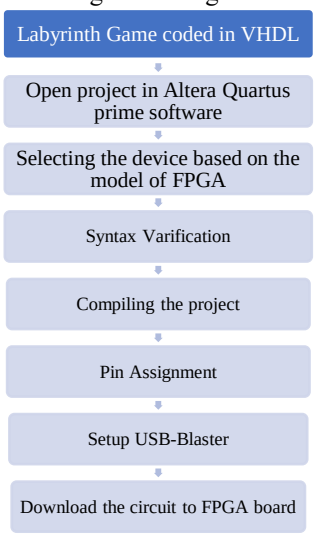

Fig. 12 System implementation flow

4.2 Block Diagram of FPGA DE2-115

The block diagram of DE2-115 with components is provided with reference in Fig.13

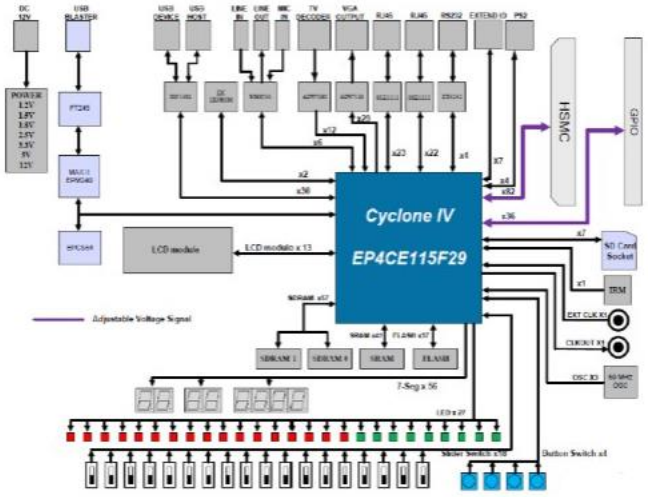

Fig. 13 Block diagram of DE2-115 (Altera Corporation, 2010)

4.3 Quartus Prime software

Fig. 14 illustrates the user interface of the Quartus prime software. In Quartus Prime, design entry can be performed using various file types such as VHDL, Verilog HDL, Block Design File, System Files, State Machine File, and others. This flexibility allows users to engage in diverse tasks, encompassing the creation, simulation, compilation, and configuration stages of the design process

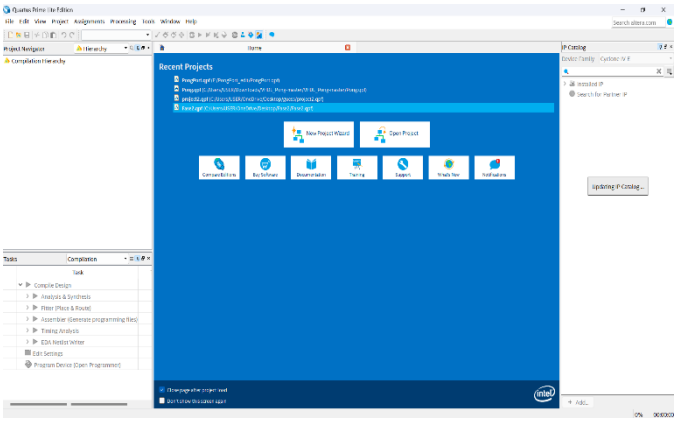

Fig. 14 User interface of the Quartus prime software

V GAME STIMULATION AND RESULTS

Before proceeding with a particular task, it is essential to conduct preliminary tests to ensure the functionality of the equipment intended for use in the project. Three basic tests were performed, specifically targeting the CAD Tool, FPGA board, and VGA monitor. The objective was to assess their performance and verify their suitability for integration into the game design.

5.1 Program Compilation

To enable the designed circuit to run on the FPGA, it must undergo programming and configuration. The configuration file was generated using the Assembler module of the Quartus prime Compiler. Altera's DE2-115 configuration is facilitated through the JTAG mode, where JTAG, an acronym for Joint Test Action Group, provides a straightforward method for testing digital circuits and loading configured data into the development board, adhering to IEEE standards.

For the configuration process, a cable linking a USB port on the host computer (running Quartus prime software) to the leftmost USB connector on the board was utilized. It is imperative to have the USB-Blaster driver installed for successful connection. In JTAG mode, the configuration data was directly loaded into the FPGA device. With this approach, the FPGA retains its configuration as long as power is consistently supplied.

The VHDL code for Labyrinth game was successfully compiled in Quartus prime software. Its LUT utilization of Labrinth game is mentioned in Fig. 15.

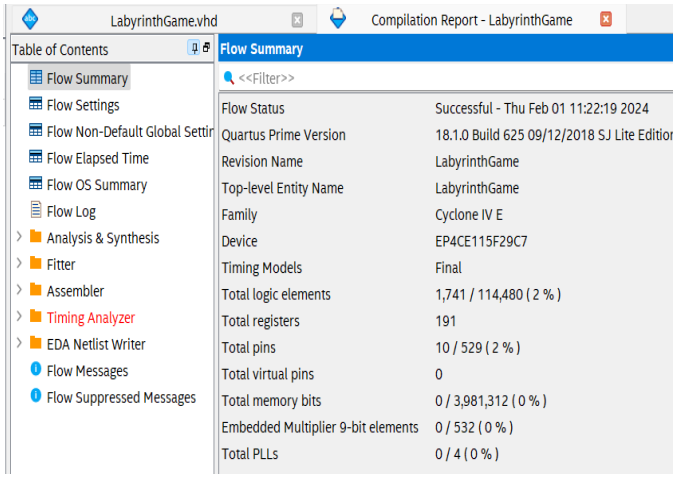

Fig. 15 LUT utilization of Labyrinth game

5.2 Hardware requirement

Following equipment's are required for the successful testing of the Labyrinth game. A typical hardware setup is shown in Fig. 16.

1. Host Computer
2. Altera's DE2-115 board (Device: EP4CE115F29C7)
3. VGA monitor
4. VGA cable
5. Power cable for DE2-115
6. USB blaster
7. PS/2 Keyboard

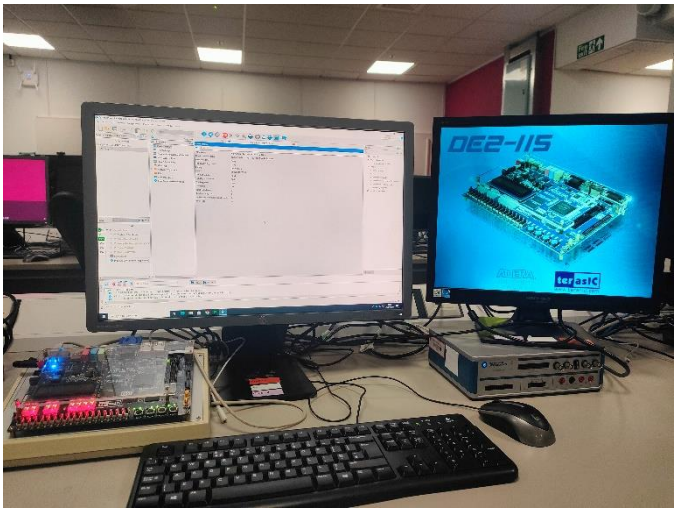

Fig.16 Hardware setup for Labyrinth game

5.3 Pin Assignment

In the realm of FPGA design, pin assignment encompasses the task of linking designated physical pins on the FPGA device to the logical signals and functionalities within the digital circuit being developed. FPGA devices are equipped with a matrix of programmable logic elements, interconnect resources, and I/O pins, which can be tailored to suit the requirements of a given application. The assignment of pins is a pivotal stage in the FPGA design workflow, requiring the allocation of specific logical signals in the design to pins on the physical FPGA device. This procedure holds significance as it determines the routing of signals among diverse components of the FPGA and establishes how these signals will interact with the external environment. This meticulous process is instrumental in optimizing the performance, power consumption, and reliability of the final FPGA design, ensuring seamless integration with external components and systems. The pin assignments which are used in Labyrinth game is shown in Fig. 17.

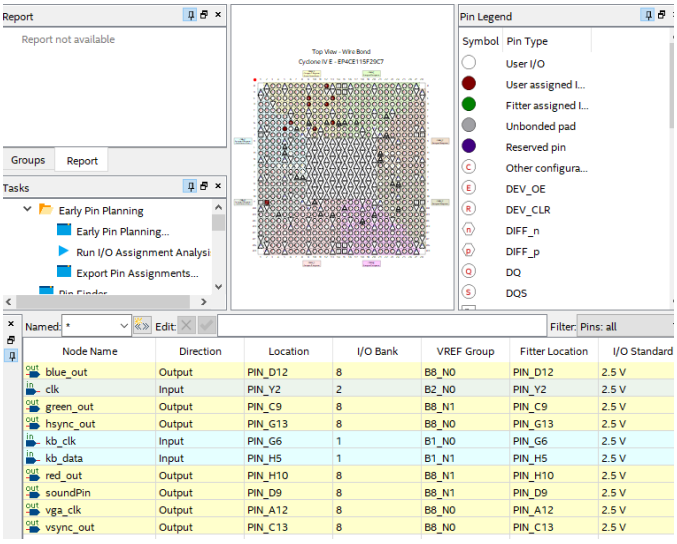

Fig. 17 Pin assignment in Labyrinth Game

5.4 JTAG configuration in FPGA board

The DE2-115 incorporates the Cyclone EP4CE115 device, boasting an impressive array of specifications. This FPGA device is distinguished by its substantial capacity, featuring 114,480 logic elements, making it the most extensive among the Cyclone IV E series. Furthermore, it offers up to 3.9-Mbits of RAM and includes a remarkable 266 multipliers. Beyond its sheer capability, the Cyclone EP4CE115 stands out for its exceptional balance of affordability and functionality, coupled with improved energy efficiency when compared to earlier generations of Cyclone devices. This amalgamation of advanced features positions the Cyclone EP4CE115 as a highly competitive and cost-effective choice in the realm of FPGAs.

Connection between host computer and DE2-115 board is implemented through USB blaster port. For that installation of Altera USB Blaster driver software is necessary. The result after successful running of board shown in Fig. 18 and board status after configuration is illustrated in Fig. 19.

Fig. 18 Process of loading configuration data into FPGA

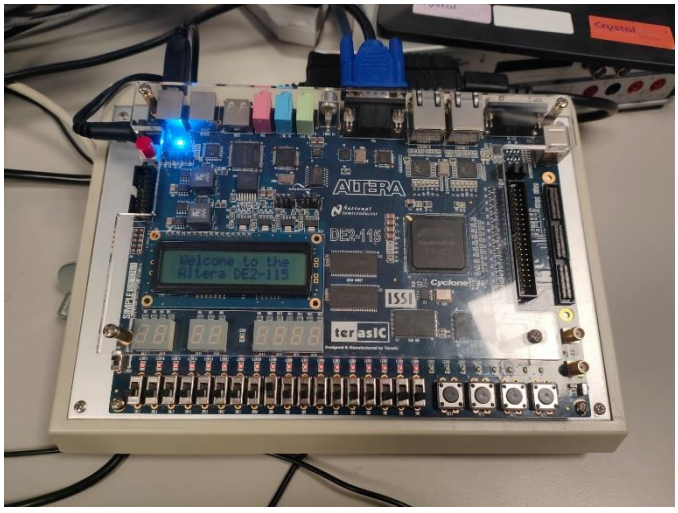

Fig. 19 DE2-115 board status after configuration

5.5 Synchronisation in VGA monitor

Creating a VGA display on an FPGA entail generating the essential timing signals and RGB colour data to produce video frames. VGA serves as a prevalent standard for linking displays to computers and various devices. A typical VGA display for the game is mentioned in Fig.20.

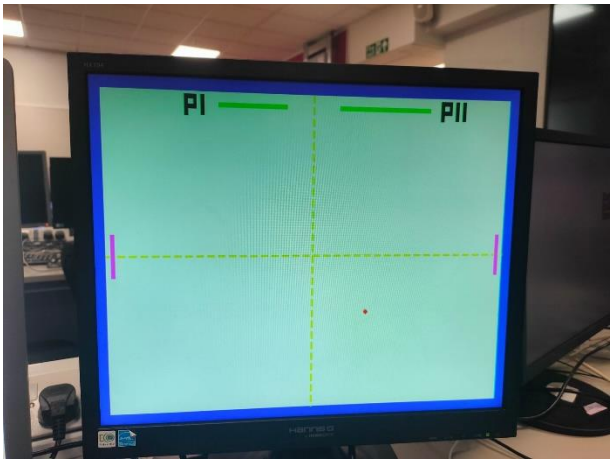

Fig.20 VGA display of Labyrinth game

VI CONCLUSION

The VHDL-based implementation of the Labyrinth game has offered significant insights into the realms of digital game design and FPGA applications. This endeavour effectively integrates expertise in hardware description languages with the principles of game development. Despite encountering challenges along the way, the final result stands as a fully operational and entertaining Labyrinth game, laying the groundwork for ongoing exploration and enhancement.

In the ever-evolving landscape of technology, the convergence of hardware and gaming opens up compelling avenues for innovation and creativity. This project acts as a foundational step, propelling us toward a more profound comprehension of digital design within the sphere of entertainment applications. As advancements in technology persist, the opportunities for refining and expanding upon this intersection are both promising and exciting.

The project commenced with acquiring foundational knowledge in FPGA development, progressing to the creation of straightforward applications, and subsequently advancing to the exploration of VGA graphics and game development.

Several modules have been created to enhance the functionality of the games. These encompass logic for ball and paddle movement, a VGA converter, and a colour interface for various game objects. Although most of the predetermined objectives have been met, there is still an opportunity for additional enhancements and experimentation.

REFERENCES

[1] Altera Corporation, 2010. U. Manual, “Altera DE2 Board”, s.l.: Terasic Technologies.

[2] Michael Santarini.Altera CPLD targets portable-system applications[J]. Electrical Design News,2008,53(1):16-16.

[3] Jiang Chunmao,Zhang Guoyin,Huang Chunmei.Research of Embedded Operating System Based on Multi-core Processor[C].//2010 3rd IEEE International Conference on Computer Science and Information Technology

[4] ALTERA CORP.Apparatus and Methods for Optimizing the Performance of Programmable Logic Devices: US, US201113326082[P].2012-4-12.

[5] Stephen D. Brown, Zvonko G. Vranesic, Fundamentals of Digital Logic with VHDL Design, 2nd ed., McGraw-Hill Professional, 2005.

[6] Pong P. Chu, FPGA Prototyping by VHDL examples, New-Jersey,USA: John Wiley & Sons, 2008.

[7] Sadiye Nergis Tural-Polat, Pong Game as an Embedded System, Proc. of the Second Intl. Conf. on Advances in Computing, Electronics and Communication - ACEC 2014

[8] R. Szabó, and A. Gontean, “Pong game on FPGA with CRT or LCD display and push button controls,” In Proc.Federated Conference on Computer Science and Information Systems (FedCSIS)’14, 2014, pp. 729-734.

[9] D. Crookes, K. Benkrid, A. Bouridane, K. Alotaibi, and A. Benkrid, “Design and implementation of a high level
programming environment for FPGA-based image processing,” In Proc. IEE Vision, Image and Signal Processing’00,2000, pp. 377-384.

[10] Myers, Robert L., 2002. Display interfaces: fundamentals and standards. Chichester: John Wiley and Sons