Embedded Systems - Final Project

Oleksandr Kuprii, Nikolay Ichov

December 2024

Contents

1 Introduction **3**

2 Requirements **3**

3 Game Logic **4**

4 System Design and Implementation **5**
4.1 Hardware Components Mapping 5
4.2 VHDL Program Architecture 5
4.3 Program Pseudocode 6

5 Issues **8**
5.1 Display Horizontal/Vertical lines 8
5.2 Entity separation 9
5.3 Update screen on a falling edge 9
5.4 7-segment display 9

6 Demonstration **9**

7 Conclusion **10**

8 Resources **11**

1 Introduction

This document explains the final assignment of the Embedded Systems course. The chosen assignment is creating a Tetris game and outputting it on a screen using VGA.

It was done with a DE1-SoC board and the game is following the basic rules of Tetris. The ways of implementation, functionalities, and encountered issues can be found further in the document. The importance of the assignment is understanding the logic of VHDL programming, and choosing a project like this can benefits a lot to the goal due to its complexity and exploration of various components and aspects.

The rules followed to create the game are the basic rules of Tetris which are as follows:
Shape is generated on each move and the goal of the player is to align the shapes in a way that they fit each other. If one full line is filled, this line disappears and points are increased. Players can move the shapes left and right, and to rotate them at 90 degrees. However, the rotation is not implemented in this game. The goal of the game is to survive as long as possible by filling lines and reaching as high score as possible. Example of a game can be seen here.

2 Requirements

In the beginning of the project, the team outlined the requirements for the system to set expectations of the final result and show what additional features can be built. The requirements were split into "must" – the crucial baseline features for the minimum working product. Additionally, there is a list of "should" features - extra things that make the game more appealing and add extra functionality.
Must:

- the game is displayed on the screen with VGA port
- game grid 20x10 squares, placed in the middle of the screen
- black and white colors
- there are 7 defined shapes
- the shape is generated in the top center of the screen
- the shape can be moved to left/right with 2 buttons
- the completed row is detected and removed
- the score is calculated based on a number of completed rows
- the "game over" is detected when a new shape cannot be generated

Should:

- the game can be paused or resumed with a switch
- the shape can be rotated clockwise with a button
- the shape can be dropped faster by a press of a button
- the game can be resetted and started over with a button press
- there is a preview of the next shape in order
- there is a game timer
- there is a preview of a placement of an active shape on the screen

3 Game Logic

The Tetris game that was created follows the rules of Tetris itself. The only difference that was not implemented was the rotation of the shapes. All shapes exist - O, I, S, Z, L, J, T.

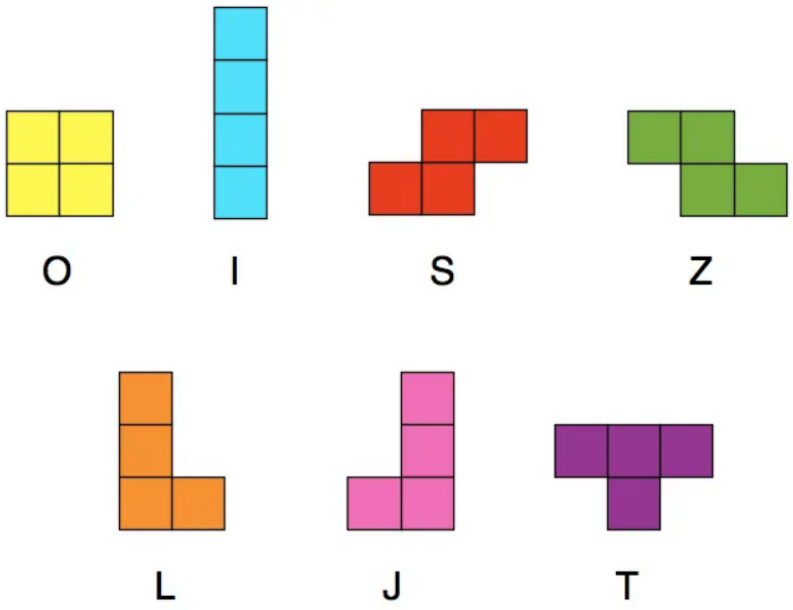

Figure 1: Tetris shapes used in the game (ref. 1)

The game is built using 2 buttons (KEY3 and KEY2) to move left and right, a reset button (KEY0) that can be used at any point of the game but is implemented to reset the game when it is over. The reset button resets the grid and the score, giving the player a completely fresh start. Another component that is used is an LED (LED9) that is ON to indicate the game is over. The final part of the implementation is a switch (SW9) that pauses the game. When the game is paused LED (LED7) is on, showing that the pause switch is active.

Since the goal of Tetris is to reach as high score as possible by removing the fully filled rows, a scoring system is included as well. When a row is filled with shapes, this row is removed and 1 point is added to the score. The score is displayed on two 7-segments displays (HEX0 and HEX1).

The shape is generated from an array of shapes and it is always displayed in the middle of the grid. It moves down with a speed of 1 second. It stops when it reaches the bottom or the surface of another shape.

4 System Design and Implementation

This section describes the design of the systems, hardware devices mapping and the architecture of the VHDL program.

4.1 Hardware Components Mapping

Before implementing the project the team created a mapping of hardware components to the required features. The mapping below shows what buttons, switches and LEDs with corresponding component coding match to what features:

```
most left switch (SW9) - start/pause
1 button (KEY0) - reset
2 button (KEY1) - rotate
3 button (KEY2) - right move
4 button (KEY3) - left move
2 7-seg (HEX0, HEX1) - score
4 7-seg (HEX2 - HEX5) - timer
1 led (LED9) - game over
1 led (LED7) - pause
```

4.2 VHDL Program Architecture

The program was split into separate components to decouple its logic into components and reduce the complexity of each of them. Moreover, it helped to overcome issues with synchronising signals in the architecture. The entity diagram is on 2 which also shows the input and output signals in the system as

well as which signals are transmitted between entities. The overview of each entity is given below:

VGA Controller. It generates signals to control data transmission to the screen through VGA, such as 'pixel_clk', 'Hsync', 'Vsync', 'n_blanck', 'nsync'.

Game Controller. It is the core of the game as it implements the game logic and is responsible for the whole functionality of it. This controller manages the play grid, active shape and score, while maintaining multiple states such as start, pause, and game over.

Visible Grid Controller. While 'updated_grid' keeps track of what cells are already filled with the items in the game, visible grid controller prepares this grid to be displayed on the screen by also adding there an active shape to the screen, which is updated every game cycle.

Pixel Generator. It receives VGA control signals together with 'visible_grid' from the visible grid controller and transforms this grid into a range of red, green, and blue signals to visualise the grid as pixels on the screen.

Seven Segments Controller. This entity is responsible for displaying the current game score on two 7-segment displays by converting an integer value into a series of active segments on the displays.

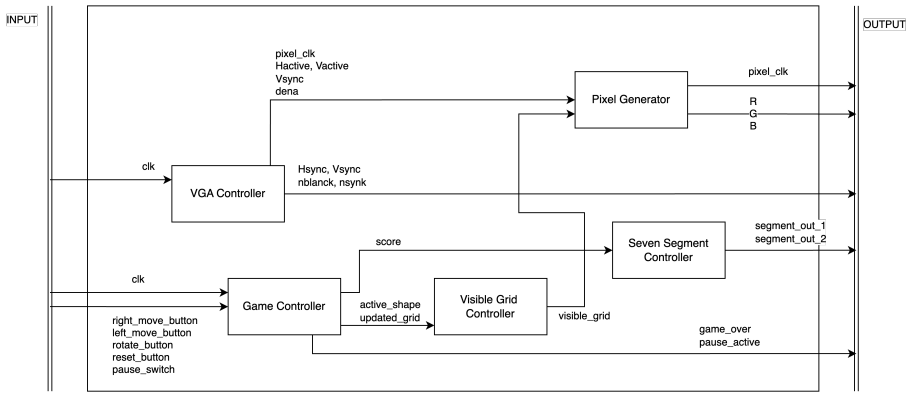

Figure 2: Entity Diagram

4.3 Program Pseudocode

The following pseudocode was written during the planning phase to outline the main software components and functions that must be implemented in the game controller:

```
if pause:
    continue

if not pause:
    every step_time seconds:
```

```
    if not shape:
        # Generate shape and place it above the screen, y>=20
        generate_shape()
        continue

    if shape:
        # Check if the the shape can be moved down
        if check_move_shape_down():
            # move all coordinates down
            move_down_shape()
        else:
            # Check if any piece of the shape is above the screen
            if check_game_over():
                finish_game()
            else:
                # Update the grid with coordinates of the shape
                # Clear the shape
                add_shape_to_grid()

                # Check if the bottom row is completed
                if check_bottom_row_complete():
                    # move all cells down
                    move_grid_down()

                    increase_score()

every clock cycle:
    # Render grid on VGA
    show_grid()

    # Show score on 7-segment displays
    show_score()

trigger shape left transition:
    # Create a new shape by left transition
    new_shape = shape_left_transition()

    # Check if new_shape does not interfere with the grid
    if check_new_shape_on_grid():
        # Update the shape
        shape = new_shape

trigger shape right transition:
    # Create a new shape by right transition
    new_shape = shape_right_transition()
```

```
        # Check if new_shape does not interfere with the grid
        if check_new_shape_on_grid():
            # Update the shape
            shape = new_shape

    trigger start/pause:
        # Update start/pause state

    trigger reset:
        # Reset the game
        # - grid
        # - shape
        # - score

        # Start a new game
        pause = false
```

5 Issues

5.1 Display Horizontal/Vertical lines

After connecting the board to a screen via VGA, the first step was outputting something on the screen. This took a great amount of time and experimenting until the result was achieved. Initially, only horizontal lines could be seen and the vertical ones created further trouble.

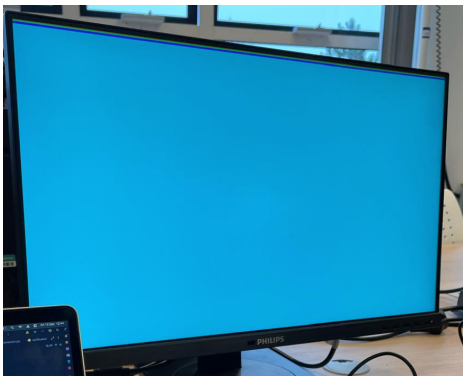

Figure 3: Horizontal lines can be observed on the top of the screen

Eventually, the problem was fixed by implementing a pixel counter that counted pixels of the display instead of dividing it into horizontal and vertical lines. Having this fixed, the project could continue with creating the grid, shapes, etc.

5.2 Entity separation

The first major issue encountered during the project was having all the functionality under one entity. The occurrence of this issue happened during the implementation of an array of shapes to be displayed on the screen one after another. An integer variable had to increase its value and display the corresponding shape after each clock cycle. However, the clock was interfering with the different processes and could not change the proper values. Therefore, the code that was created until now was separated into new files and new entities, all referenced in a top-level file.

5.3 Update screen on a falling edge

Adding the functionality of moving the shapes left and right came with the need to change some other parts of the code. The update of a screen was happening during the process of moving down shape. The left and right movement was implemented in the same process. However, this gave some distorted shape results. To keep the stable shape a new variable for updating the grid was created. This variable took the value of the grid and it was updated and displayed during the falling edge of the clock. This way it stopped interfering with the shape movement.

5.4 7-segment display

One of not minor issues that were encountered was displaying the score on the 7-segment displays. Even though, it was a straightforward implementation of giving pins and values to the 7-segment displays there were complications with that. Throughout solving it, the pins were checked several times ensuring that they were correct. Eventually, after some changing of the values the desired result was accomplished.

6 Demonstration

Some of the results are shown here. Includes images of a partly played game where some of the shapes are displayed on top of each other. Leading to a further continued game with the grid being half-filled. Images Fig.6 and Fig.7 shows the usage of the LEDs and the switch for pausing the game. When the switch is turned to pause the game, an LED is ON. The same happens when the game is over, it is indicated on another LED.

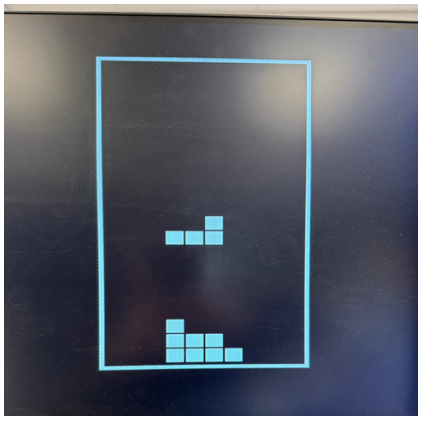

Figure 4: Partly played game

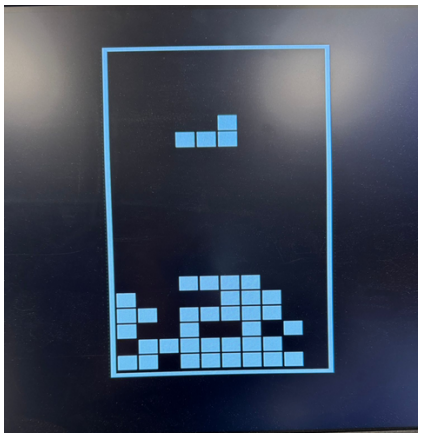

Figure 5: Half filled grid

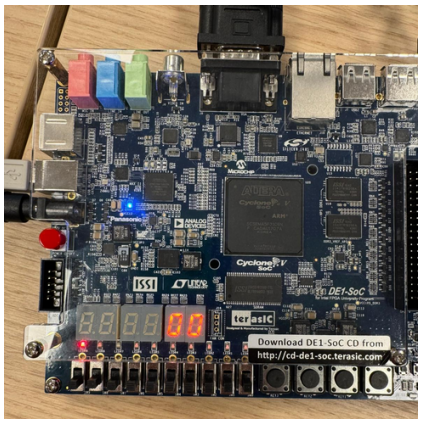

Figure 6: Game over indication with LED ON

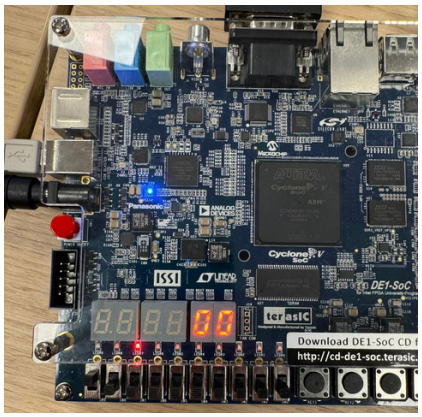

Figure 7: Paused game indicated with LED ON and switch ON

7 Conclusion

The team completed all "must" requirements, resulting in a fully working Tetris game. Additionally, a few "should" features were implemented to enhance the game and improve its functionality, particularly: pausing/resuming the game with a switch and resetting the game with a button press.

The content of the project brought many issues that further into development got fixed. This came with a better understanding of the architecture of VHDL and its way or functioning. During the work aspects like creating an entity, separating entities and combining them into one file, creating constant variables, and understanding the usage of the clock were explored and studied. The

difference between function and procedure were observed and implemented depending of the need.

Overall, the project successfully achieved its goal of giving a good amount of knowledge to the students.

8 Resources

1. Mark M Liu (01.01.2018), The Tetris Proof, *Medium*