Puzzle Game “2048”:
Implementation by Using Max10 FPGA

Authors

Andrei Markov, Leila Khaertdinova, Albert Khazipov

1 Introduction

The Puzzle Game named “2048” is one of the most popular puzzle video games, along with Tetris, etc. It was first introduced by Italian web developer Cirulli in 2014. The game stimulates the development of analytical skills.

In this project, we implement the “2048” game in the Quartus Prime programming environment, by using Verilog HDL and Max10 FPGA hardware board. These tools allow us to simplify the game design and create a concise game implementation.

2 Project goal

The main project goal is to implement a “2048” game using Max10 FPGA.

General objectives

- Time-scale: two weeks;
- Graphical interface: the game implementation should be concise, fast, and user-friendly;
- Run-speed: a reasonable speed is important, but not in prejudice of accuracy, the required time to compile the program code is expected to be approximately fifteen minutes;
- Visual display: a simple display (640×480).

Section 1. Game mechanics

The game field is a 4×4 grid with a collection of tiles. Each of them can contain 2^n value, where $1 \leq n \leq 11$. Initially, all tiles are empty, as illustrated in *Figure 1a*. When the game starts, a tile with a randomly chosen value “2” or “4” enters the game grid. A random number appears after each game move, as shown in *Figure 1b*. The details on random generating are provided in Section 4.2.

There are 4 possible options for the game move: move to the left, move to the right, move up, and move down. For instance, *Figure 1c* demonstrates down move.

While a cell is shifted to the selected side, it either reaches the grid border or gets stopped by another cell. If two tiles have equal values and collide during one game move, they are merged and their values are added.

Figure 1d illustrates up game move and random "2" value appearance. In *Figure 1e*, we provided an example when two tiles are merged. Similarly, we demonstrate how cells move to the right and sum the equal numbers (*Figure 1f*).

Finally, the game is finished when one of the following situation occurs:

1. If a player manages to get a tile with a value of 2048, he or she becomes a winner (*Figure 2a*);
2. If a game grid is fully occupied and the player cannot make any game moves, the game is over (*Figure 2b*).

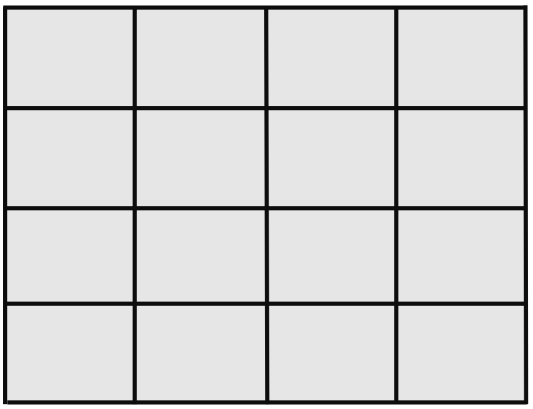

Figure 1a. Empty game field

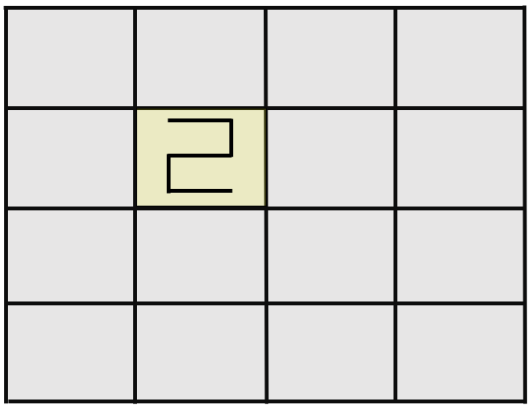

Figure 1b. The first random value appears

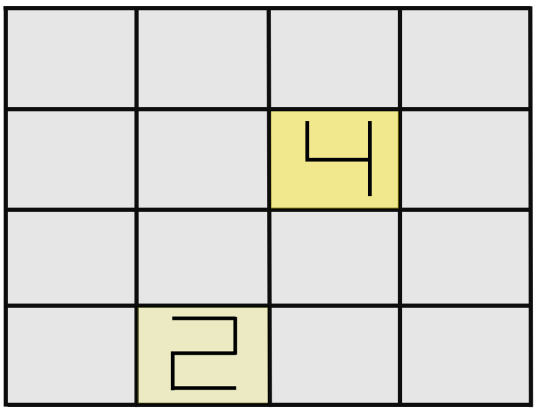

Figure 1c. "Down" game move and appearance of value "4" in a random tile

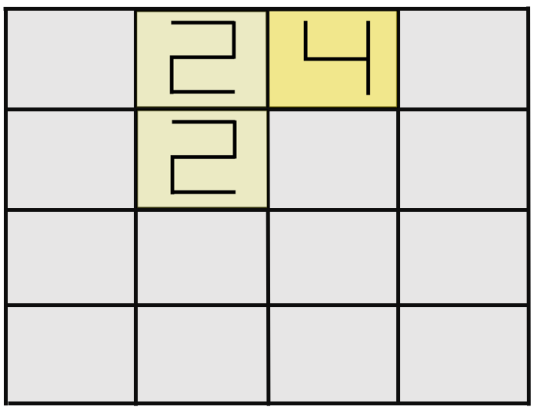

Figure 1d. "Up" game move

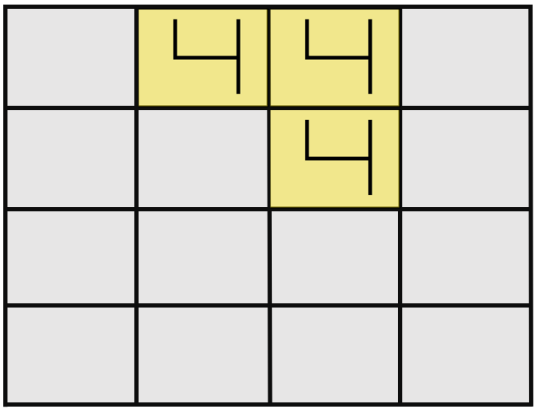

Figure 1e. Tiles move up and two cells with “2” value are merged

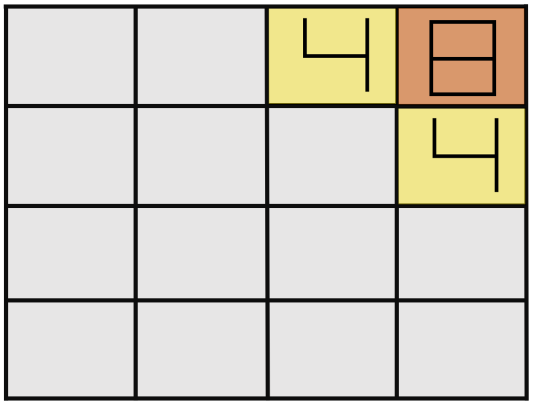

Figure 1f. Move to the right and merging of cells with equal values “4”

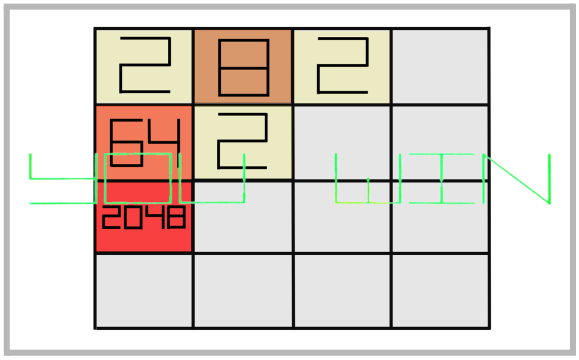

Figure 2a. End of the game (win)

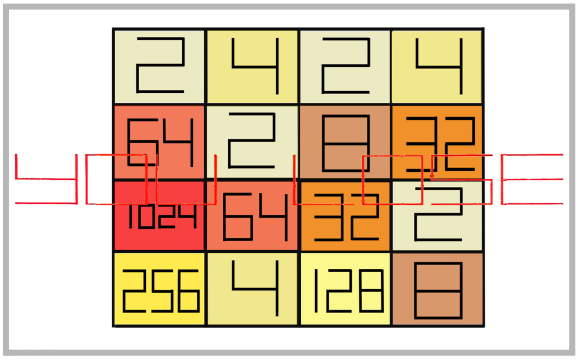

Figure 2b. End of the game (lose)

Section 2. Devices and tools used

2.1. Our project implementation requires the following hardware devices:

- Max10 FPGA Board

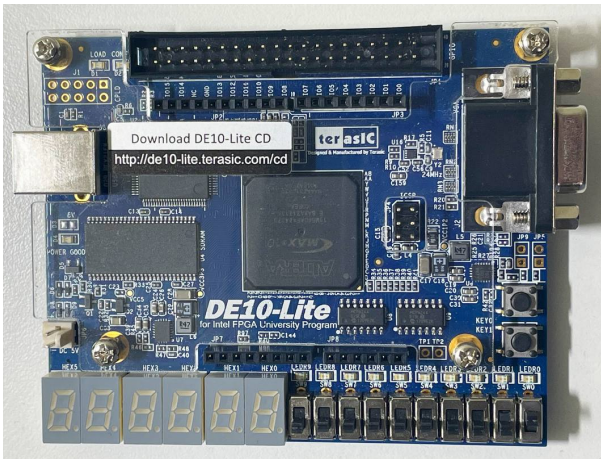

Figure 3. FPGA Max10

In this project, we use Max10 FPGA Board as an executing device and an user interface. More detailed information about FPGA is provided in Section 4.1.

- 2D Display
- VGA connector

Figure 4. Video Graphics Array

VGA connector is used to connect Max10 FPGA Board to display.

2.2 We use the following software tools:

- Quartus Prime Lite Edition 20.1.1.
- Verilog HDL

Section 3. Technical implementation

To implement the “2048” Game, we first develop a user interface with the Max10 FPGA board (*Figure 5*). Then, to visualize the game, we connect FPGA to a display by using the VGA connector (*Figure 6*). We next determine the game grid marking (*Figure 7*) and choose a special design for cells and numbers. Finally, we write the code on Verilog to implement the game logic.

3.1 Max10 FPGA

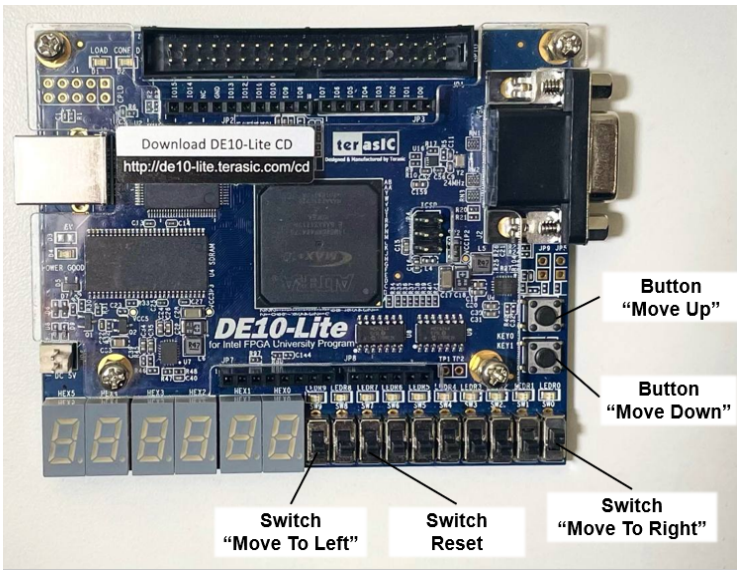

Figure 5. User interface with FPGA

3. 2 VGA

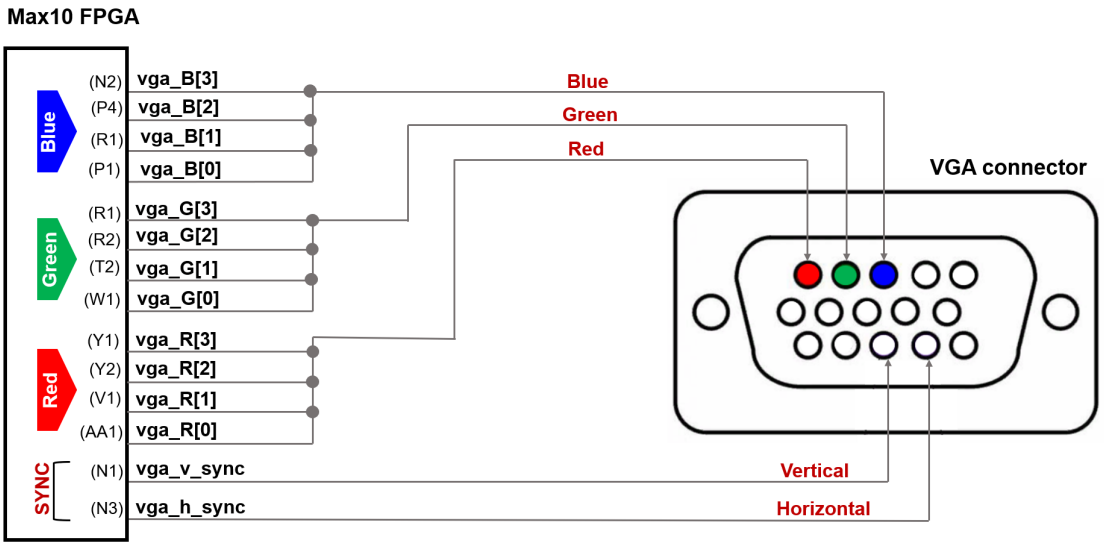

Figure 6. Scheme of VGA connecting

The VGA connector requires a clock frequency of almost 25 MHz. The FPGA has 50 MHz clocks. So, we need half of the clocks of the FPGA oscillator. The available palette contains 4096 colors. With a frequency of 25MHz, we are able to transmit colors for a single pixel during one clock cycle.

3.3 Game grid design

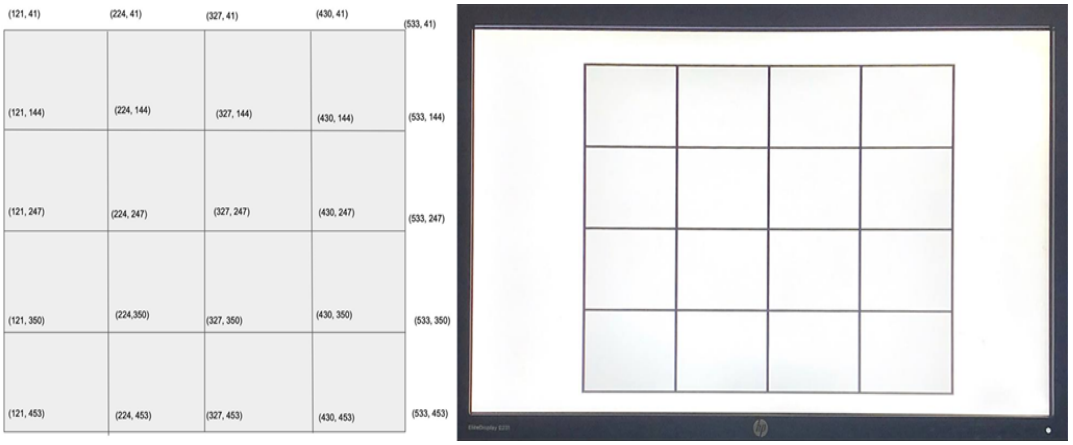

Figure 7. Coordinates of the game grid and numbers

The picture in *Figure 7* shows the coordinates of the grid and its elements. First, we determine the coordinates of the game grid and tiles. Second, we define the positions of the numbers in cells and then determine each number coordinates respectively.

3.4 Schematic representation

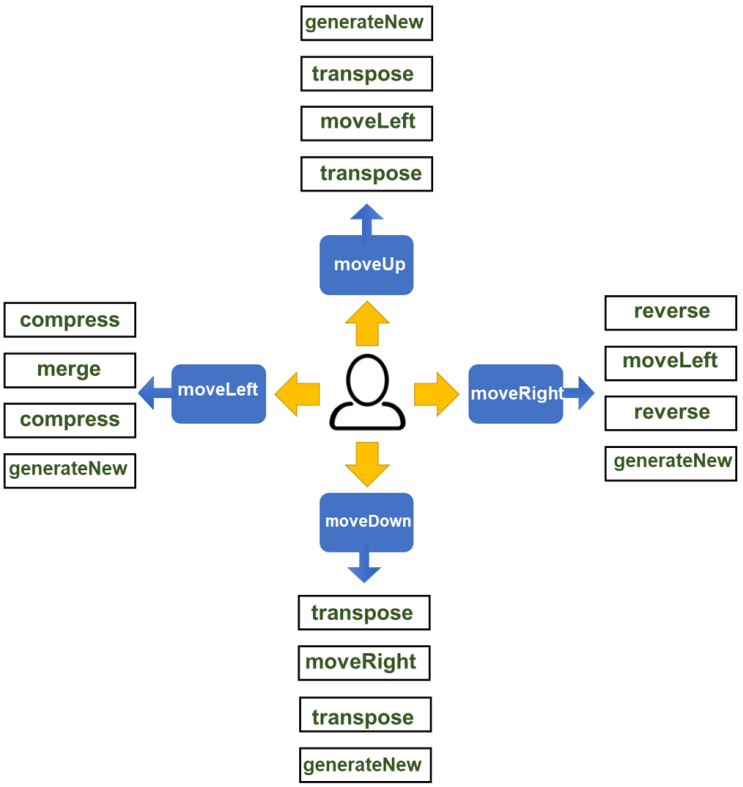

Figure 8. User interaction with the program code

The following functions are defined to implement the logic of the game:

- **compress**: shifts all cells to the left side replacing free ones
- **merge**: finds colliding identical cells and merging their values
- **reverse**: flips the game grid horizontally
- **transpose**: flips with respect to the main diagonal
- **generateNew**: generates a new tile in a random position

According to the scheme, **MoveLeft**, **MoveRight**, **MoveUp**, **MoveDown** functions are combinations of previous functions.

Section 4. Additional information

4.1 FPGA

FPGA (Field Programmable Gate Array) is an integrated circuit that allows designers to program customized digital logic in the field. There are various FPGA models, for example, Cyclone 4, Cyclone 5 and Max10.

The main advantage of using FPGA boards is that they enable people to build special hardware and do operations in a simpler and more energy-efficient way.

4.2 Random generation

After each game move, a new value “2” or “4” appears in a random empty tile. Number of the cell that contains a new value updates every clock cycle.

In this project, we consider 75% probability to “2” value appearance. Likewise, we pick out 26% probability to “4” value appearance. What is more, if one of the central tiles is empty, we put a new number into it with the highest opportunity.

Conclusion

We described the key stages of the game “2048” implementation by using an FPGA Board, display, Quartus Prime software, and Verilog programming language. Our project was completed as a part of the «Computer Architecture» course at Innopolis University. The implemented game can be used for entertainment and also for educational purposes.

Acknowledgments

We extend gratitude to our professors Artem Burmyakov, Alexander Tormasov, and our teacher assistant Kirill Poletkin for giving us in-depth knowledge and the possibility to use it in practice.

References

1. 2048 Game
2. FPGAs for Dummies by Andrew Moore with Ron Wilson
3. Quartus Prime Lite Edition
4. Driver Installation
5. Button Debouncer
6. VGA adapter

Step-by-step instruction

Here is the step-by-step instruction, how to use our implementation:

1. Install Quartus Prime 20.1.1 and drivers

2. Go to our GitHub repository: Game2048_FPGA

3. Click on the button “Code” and then “Download ZIP”

4. Unpack downloaded archive

5. Open Quartus Prime 20.1.1.

6. Press “Open Project” and find the file “OptionalProject.qpf”

7. Choose your FPGA model in “Assignments” → “Device”

8. Go to “Assignments” → “Pin planner” and assign the following pins (only for Max10 Board):

9. Start compilation (around 15 minutes)

10. Connect your FPGA Board and go to “Programmer”, then press “Start”

11. Connect your Board to display using a VGA connector

12. Done! You can enjoy the “2048” Game!