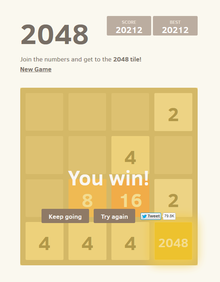
Final Project 2048

Abstract

Our project is to duplicate the game 2048 on the Nexys 3 board and a screen with VGA display. By pressing respective buttons on Nexys 3 board, users are able to play the popular game 2048 on any screen connected to the board. 2048 is a sliding block game on a 4X4 grid, every turn, a new block with number 2 would appear. Two block of same value will merge and add together when moving. When there is a block with 2048, user will win the game.



Introduction and Background:

In this project, we have referred to a lot of previous lab work we did. We are using a divclock from the introduction lab to get a slower scanning of the screen displayed. Also, by studying the VGA module provided and referring to the previous seven segment display, we are able to display our numbers by smaller segment.

Design:

Our objective is to duplicate the game 2048 using Verilog coding on Nexys-3 board. To achieve this, we must enable the function of moving blocks to blocks and merging blocks of the same number. In order to achieve this, we design the following state diagram.

Array[15:0]=0

Start

reset

After reset, the 4\*4 grid (we set it as a 15:0 array) will be initialized at the Initial State, all the value are set as 0. (0 is considered blank in this game). After a start signal was generated, the system will generated a number 2 at a random position in the array. Since the random function in Verilog is not synthesizable, so we just use a counter with its last four digits to keep counting to get a random position in the array. After a two was generated, the user will press any of the BUTN (UP, DOWN, LEFT, RIGHT) to make a move. Then the state will entered to move state, if there is no two blocks of the same number were arranged in the moving direction, we will go back to generate state to get another 2 at any random blank position. If the condition is able to fulfill the merge condition, the state will transfer from move to merge and every possible pair of blocks will merge once. After merge, the state will go back to generate state to generate another 2 at random empty position in the array. The state will go to Done if we are able to get a number 2048 at merge state. Otherwise, if all the blanks have a number, that means if after generate state, there is no mover or merge occurs. The state will go to Done state to end the game. Then the Ack signal will shift the state to initial state and once reset the gam will start over.

In order to play this game, User have several important input. Sw0 for to reset, Sw1 to Start, BTNU as upward movement signal, BTND as downward movement signal, BTNL as leftward movement signal and BTNR as rightward movement signal. The output is mostly on the VGA display, we are able to display the 4X4 grid with numbers in the center of the blocks to display to the user. Most of the output signals are just the number signal to display. The user will be acknowledged after the game is a win or a loss.

Test Methodology

In order to test and debug our code, we created our own testbench to run simulation and to see the value and state transition. Moreover, we are able using the bit file produced to implement it on the board and connect board to a screen using VGA port. By using VGA module we are able to visually see the flaw in our design and edit based on what we have seen.

Conclusion and Future Work

When we first pick this 2048 game, we thought it was easy to implement our design on the Nexys Board through Verilog coding since 2048 is not a hard or complex project if we are using Java or C++. However, we start to understand the huge difference between software design language and hardware design language. We have to consider the synthesizability of the code and the moreover, the most obstacles we meet is to simulate and debugging. However, we are still able to get our way out by our knowledge from the previous lab like designing a testbench file to stimulate and self-study on VGA module. We think the overall lab session are great, however, we are really getting to little help from TA due to schedule conflict and no TA office hour. We think if there was an office hour for lab TA to help us out to explain further on Verilog coding we can cooperate with this course in a better way.

Timing

Resource Consumption

Number of Slice Registers: 341 out of 18224 1%

Number of Slice LUTs: 4668 out of 9112 51%

Number used as Logic: 4668 out of 9112 51%

Timing: maximum frequency: 34.768MHz