İhsan Doğramacı Bilkent University

Computer Science

Digital Design 223

PUYO POP

SECTION 6

İREM URAL  
21502278

EMRE DİKENELLİ

21501975

26.12.2016

**BLOCK DIAGRAM OF THE PROJECT**

start

reset

clk

puyoPopGame

clk

Buzzer

buzz

buzzer

gameOver

stop

clk

m, n, o, p, r, s, t, dp

SevSeg\_4digit

Seven   
Segment  
Display

[3:0] in0, in1, in2, in3

f\_clk

[3:0] an

ButtonSynchronizer

left

leftButton

right

downButton

rightButton

down

rotate

rotateButton

8x8 RGB  
Dot Matrix

MR

OE

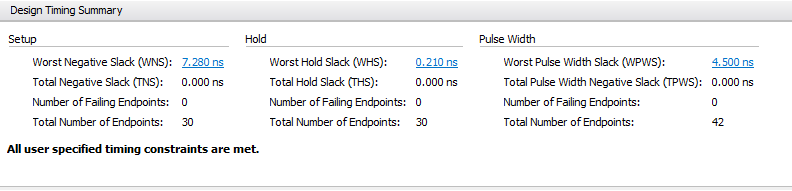
shiftclk

storeclk

DS

[7:0] rowselect

**Functions of the Blocks:**

1. PuyoPopGame Block:
2. Function of the block: The function of the block is to display the game and the steps of game are included as well.
3. Timing of the block regarding inputs and outputs: The system clock is divided by using an 8 bit counter; hence the clock frequency is 390625 Hz in this module. The system clock is 100 MHz.
4. Description of the block: In order to display, 6 variables are sent to 8x8 rgb dot matrix, the rowselect is a 8 bit variable to activate a row. serialdata is the 24 bit signal sent to each row. In each shift\_clk each bit of serialdata is shifted to the shift register by sending the data to DS. Each bit is stored in storage register and it is updated according to store\_clk. Store\_clk is opposite of shift\_clk.

When 24 bit serial data is ready, the oe signal is set to 0 which is active low signal hence it allows data to pass. When the data is passed for a one row, we select the second row to pass the serialdata. The f\_clk is also used in game part(in the always statement of the state transitions) and send to buttonSynchronizer as well. Because when we used system clock there were some glitches, some green leds turned on without any reason as we do not use green color in our code. Furthermore, the movement of the pair was disabled and it sometimes stuck in anywhere of the matrix. However we fixed these problems by using f\_clk.

First of all, a 2 dimensional array which holds 2 bits for each cell is used in order to locate colors in each cell. The 2 bits represent the color values 00 represents an empty led, which shouldn’t light any colors. 01 represents a blue color, 10 pink and 11 shows that red color should light up. The row number is increased by 1 from bottom to top starting from 0. Additionally, the column number is increased by 1 from left to right starting from 0. The usage of matrix ease our programming logic however, it creates difficulties to design, as we could not pass the matrix from one module to another. Hence we could not separate the display and game apart. Maybe because of that, while programming we come up with several difficulties as well. Furthermore, the locations of the pair are stored with 4 variables, loc1i, loc1j, loc2i, loc2j, and the values of these variables are changed while the pair is moving. Mainly the state transitions are used in this module, in the positive edge of f\_clk and the positive edge of reset signal the following state transitions occur. There are basically six states in the module. These steps will be explained briefly in the following part of the paragraph. With the reset signal the game is initialized with an empty matrix each cell has 00 values, score and gameover signal is set to zero. In the INITALIZE state, if the start signal is given by the switch it will go to GENERATE\_PAIR state, or it will wait in this state. In this part, at the top middle of the matrix new pairs appear. There are three main colors used in the program. However the colors of the pairs are the combinations of these three colors. For instance, the colors of the pairs can be red at left, blue at right or both might have pink color. The change of the upcoming pair is determined by a counter, when a new pair is generated the counter is incremented by 1, according to the value of the counter the color of the new pair is determined in a case statement. When system resetted, the color of the new pair changes all the time.The locations are set to (7,3) and (7,4) ( top middle) and the orientation starts as horizontal. In the MOVEMENT state locations are changed with respect to the orientation of the pair and push buttons. For instance if the pair is horizontal and the player enters right button, loc1j and loc2j are incremented by one if there is no ball at the right of the pair and if there is no boundary, loc2j should be less than 8. When the ball reaches at the bottom it will go to GENERATE PAIR state, else it will go to CLEAR state, to delete if there is any adjacent 4 balls with the same color. In this state it traverses the matrix and looks for 4 adjacent same color balls, they can form square or they might neighbour each other horizontally or vertically. In these three cases the balls are deleted from the matrix. Then if they are deleted, the next state is REARRANGE. In this part of the code segment, again it traverses the matrix and if there is no ball at the bottom of the ball it moves down until there is a ball or until it reaches to the bottom of the matrix (row number with 0). And finally, while generating the new pair if one or both of the top middle seats is filled with ball then it moves to the GAMEOVER state and gameOver signal becomes 1. While programming states sequential logic is used.

1. Buzzer:
2. Function of the block: This block generates the one bit output signal if the player loses the game.
3. Timing of the block regarding inputs and outputs: The output signal is generated each positive edge of the clock, which has a 100MHz frequency, the frequency of the clock of the Basys3. There is no delay between output and input.
4. Description of the block: When the game ends the gameOver signal is sent from PuyoPopGame block to buzzer and at each positive edge of the clock if stop signal is high, which is obtained from the switch on Basys3, the output signal is 0. Otherwise if the gameOver signal is 1 then a high signal sent to the port. We take the output and connect it to the positive end of the buzzer. Therefore a buzzing sound is generated.
5. Sev\_Seg4digit:
6. Function of the block: The score of the player is displayed by using this module.
7. Timing of the block regarding inputs and outputs: The system clock is divided and slowed down by using an 18 bit counter. Therefore, the clock frequency is nearly 381.5 Hz in this module. The system clock is 100MHz, and it works within this frequency without any delay.
8. Description of the block: In the block, according to 4 bit 4 decimal values obtained from PuyoPopGame module, 8 outputs and 4 bit signal is sent to the 7 segment led on basys3. The four bit four signals (in0, in1, in2, in3) are set to each value to be displayed and send to this block from game module. By using most significant bits of the counter the values of digit\_en signal and digit\_val are set, to select one of the four values on the 7 segment. Case statement is used to send signals to display. In order to light a led low signal should be send.
9. ButtonSychronizer:
10. Function of the block: This block is used to convert a level signal, which is entered by push buttons, to a pulse. It turns a continuous high signal to a one clock cycle signal when the input goes from low to high.
11. Timing of the block regarding inputs and outputs: The output signal is generated each positive edge of the f\_clk, which has a 390625 Hz frequency, the clock slowed down by the puyoPopGame module. The f\_clk is sent to synchronize the button with the game logic. However as human presses the button in an arbitrary time, it might not be at the positive edge of the clock; therefore it might not be recognized by the program as the input does not go from low to high at the positive edge of the clock. Furthermore it can be received as more than one signal, there might be debouncing.
12. Description of the block: The input signals are taken from push buttons on the Basys3. As these signals are asynchronous we need to synchronize these signals with our program. At positive edge of the clock the signal is synchronized and the previous state of the button is stored. The output signal is generated according to the last state and previous state of the button. This statement in the code corresponds to the assignment part, hence it recognizes the input when it goes from low to high and assigns 1 to the output signal.

**References:**

1. Sev\_Seg4digit block is the code which is given us in lab04. It is directly used in the code in order to display the score of the player.
2. The buttonSychronizer is used to convert level to a pulse. The following website is used to

<http://web.mit.edu/6.111/www/f2012/handouts/L05.pdf>

It is used in the buttonSychronizer module.

1. In puyoPopGame module the code which is about 8x8 rgb dot matrix in Beti Fpga Education Set is translated into SystemVerilog code, which is written in VHDL. However we made some little changes as well, to adapt it to our program. We used it to display our code on 8x8 rgb dot matrix. The translated part is specified by comments and located between stars.
2. The buzzer is obtained from outside. It is taken from one of our friend who is in third grade now. The output signal of buzzer module is sent to the buzzer and sound is generated when the game ends.

**Appendices:**

1. The names of the module correspond to the block names.

**puyoPopGame block –** puyoPopGame.sv

Funtion : display and game module

**buzzer –** buzzer.sv

Function: generate sound

**buttonSynchronizer –** buttonSynchronizer.sv

Function: synchronize buttons, level to pulse converter

**Sev\_Seg4digit –** Sev\_Seg4digit.sv

Function: display the score of the player

**Circuit**



GND

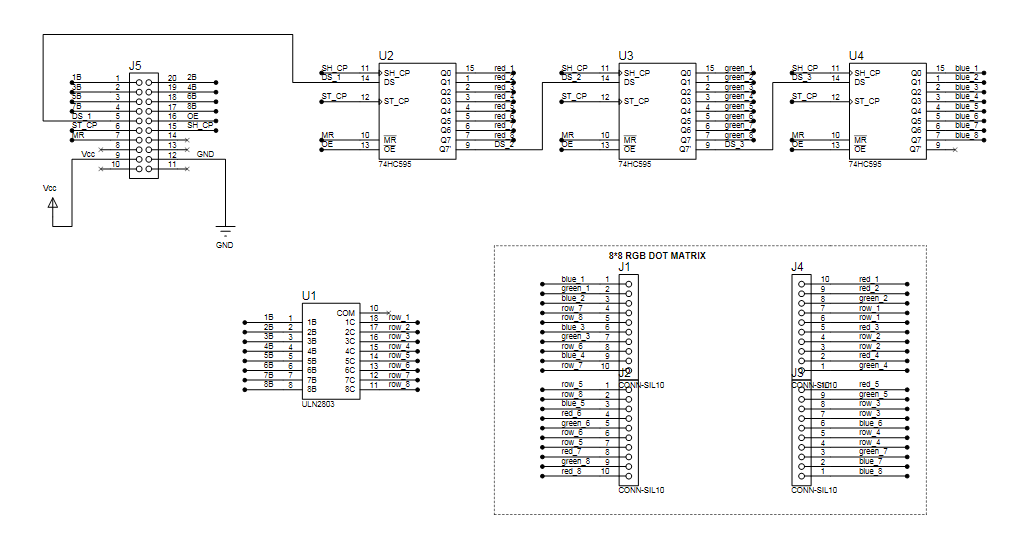
gameOver

+3.3 V

1. Datasheet of the buzzer



Datasheet of the 8x8 RGB Matrix (taken from Unilica)



Datasheet of the seven segment display (taken from Basys3\_rm.pdf file on Unilica)

