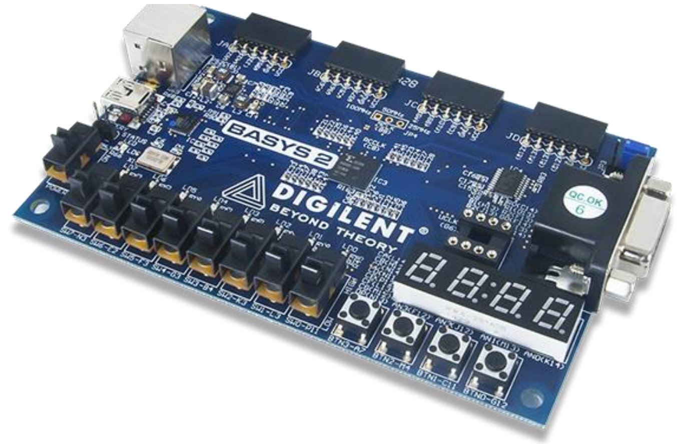
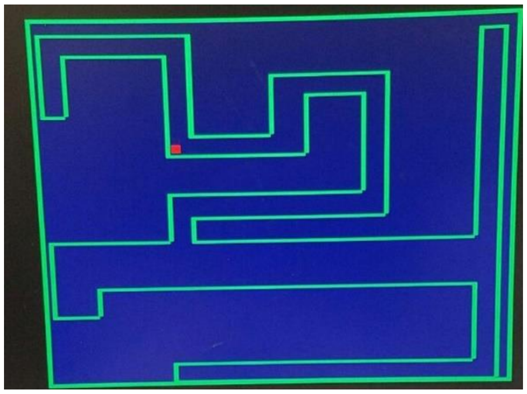




Dungeon Solver on the Basys 3 FPGA

Group 4:

Team Members	*Course*
Anthony Marando	EENG-483
Maximus Rizk	EENG-483
Priyank Kashyap	EENG-483
Vitaliy Vorobets	EENG-483

Table of Contents

1. Introduction

1.1. Dungeon Solver

There are some different versions of traditional dungeon solving games such as Doom. Such games have the user control a character through user POV and navigate a maze whilst collecting coins and fighting monsters. Should the user fail to do so in a given time frame the game resets the user to a checkpoint and after 3 tries the user dies in the game.

We have used the above as a way to set up our game. The user in our case is red block that is generated in a maze that is specified by us. Once the user finishes the first maze, the user is guided to a second maze. All the while there is a timer present which gives the user 50 seconds to complete each maze with 3 trials for the entire game. Once the timer counts the user is reinitialized to the beginning of the maze and when the user completes the maze, the screen turns green and red if the user loses.

1.2. FPGA

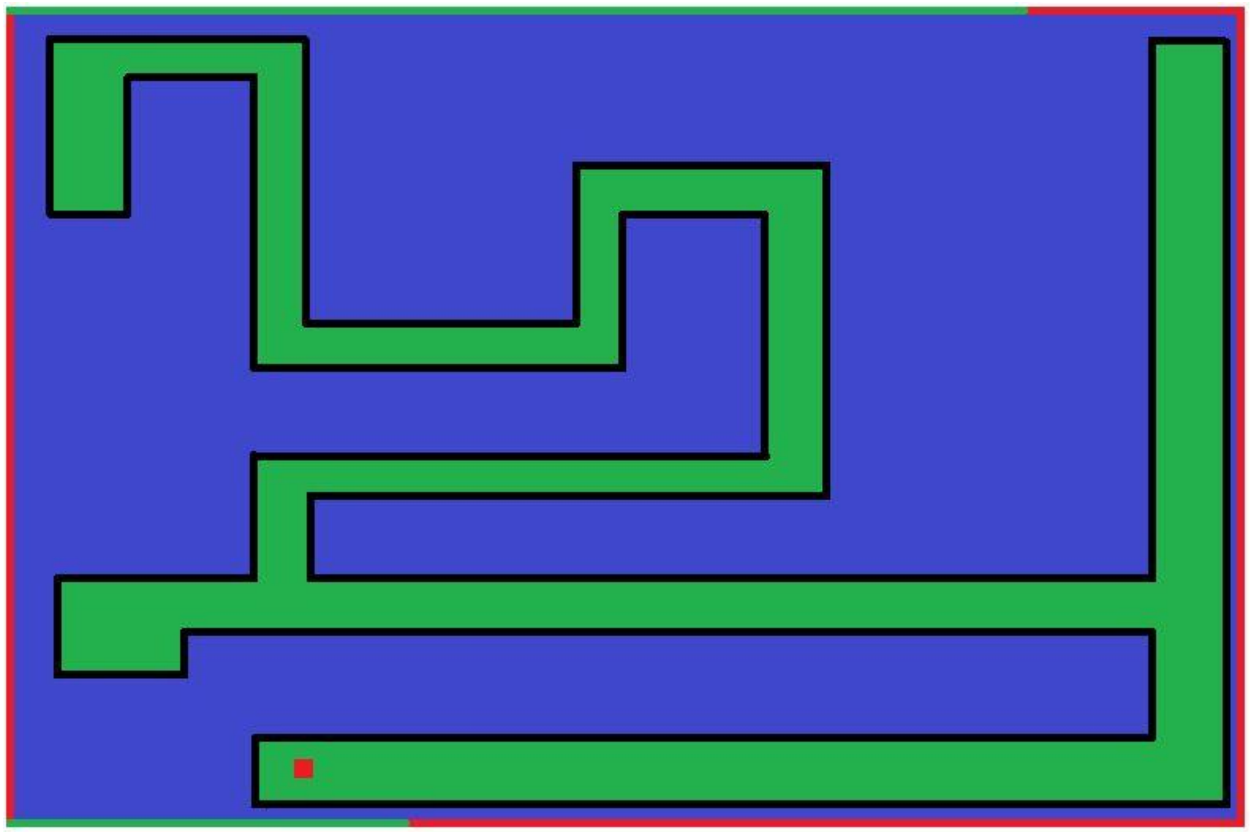
A FPGA or a field programmable gate array are devices that allow user to program their own digital circuits. It is easy to consider a FPGA as a blank slate that does nothing on its own, once the user configures it through a bit file, the FPGA will behave like the digital circuit the user had in mind. Through FPGAs the user has full control over the hardware that is connected to the board and has the advantages over ASIC (Application Specific Integrated Circuit) devices as an FPGA can be reconfigured as many times as it might be needed. For the purpose of this project, NYIT's School on Engineering provided us with a Xilinx Basys 3 FPGA board to implement and test our project on.

2. Planning

In our initial planning phase, we planned to implement the user walking through the maze using POV rather than showing the player moving and solving the 2 mazes. However, we were unable to implement such system due to the movements involved in changing the POV and we resorted to a 2D maze, where the user would have to solve the maze and by checking every corner of the maze to progress and finish the game.

We used the code given to us by Dr.Artan to play Pong a primary resource to implement our game. We split up our dungeon map into a grid system to keep the overall maze dimensions consistent between the different maps and player movements.

3. Implementation



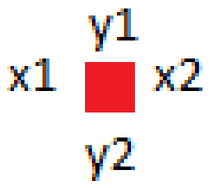
3.1. Maze

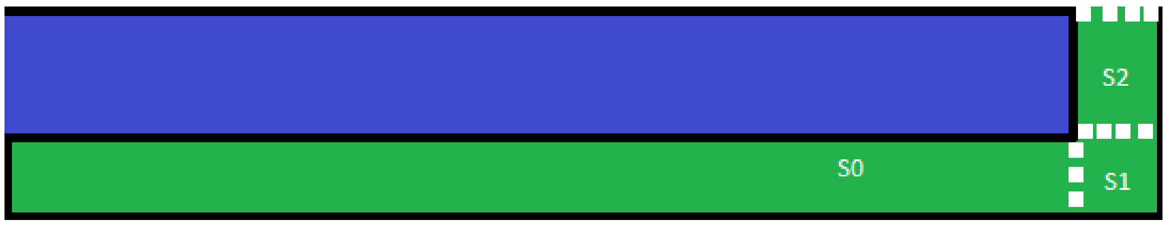
For our maze, we split the screen into grids so as to make drawing the maze easier, as many pixels would start and begin/terminate at the same point. We used VGA Sync to trace the standard monitor with 640by 480 resolution at 25 MHz's Through VGA Sync we obtain the current x and y position, which is then used to construct the maze. We encountered issues with our maze when we generated maze borders at random points due to the current x and y positions not being at the particular pixel. Our code for both maze follow a similar pattern with lines being drawn at increasing current y values. The player is generated in the maze at the bottom, however it can be changed to be generated at the top by changing the offset of the player in the player process.

3.2. Timer and Trial

As it can be seen in the above picture of our maze, we made the choice of borders of the game red. This was done so that rather than changing the signal that controls the length of the timer to change color it changes length. To do so we created a 27-bit counter, which uses the clock on the Basys 3 which clocks every 10^{-9} ns, to have an internal clock which clocks at every 1 second. At every second we decrement 8 pixels from the border length and once the timer reaches a value of 123, corresponding to the 123rd pixel the timer will reset. Should the user run out of time, the player has three tries to solve the maze. Every time the timer counts down to 123, the trial signals losses 1/3 of its length. One the trial signal is also 123-px, the game background will become red to indicate the game is over.

3.3. Collisions





Our code handles collisions similarly to state machines. In the above figures, we see an example of three states or “hallways” that are used to track the positon of the ball and therefore determine the surroundings of the wall around it. Given that the player spawns in hallway/state 0, we will use it to illustrate the example. The restriction on the ball are only in 3 directions them being top, bottom and left. If the balls y1 is greater than the boundary of the maze, the balls movement is stopped. The same principal applies for both the bottom and left. The movement is unrestricted on the right button as x1 of the player crosses the boundary dividing the 2 hallways, it switches to hallway/state 1 which has the restriction on the right and bottom direction. This was made possible with the help of the pong code from Dr.Artan which restricted the player’s movement as well as Gilbert Bogen who helped us realize the intersection problem that we were facing.

3.4. Scoring

Due to issues we had were unable to implement a scoring system to collect coins. However, we use our timer to give the player a score. With every decrement in the timer, the users score would go up and if would be displayed on the Basys 3 using the LED's on the board, the lowest score would be the winner. The idea of using the timer of the game to keep track of score was given to us by Hendra Bei.

4. Appendix

4.1. Code Snippet

A. Code for Maze Generation (1st Row)

```
    if(current_y > x"3F" and current_y <= x"8F" and current_x >= x"7E" and current_x <= x"81") then
      colorOut <= GREEN;
    end if;
    if(current_y > x"3F" and current_y <= x"42" and current_x >= x"7E" and current_x <= x"FB") then
      colorOut <= GREEN;
    end if;
    if(current_y > x"3F" and current_y <= x"A3" and current_x >= x"FB" and current_x <= x"FF") then
      colorOut <= GREEN;
    end if;
    if(current_y > x"3F" and current_y <= x"111" and current_x >= x"1F1" and current_x <= x"1F4") then
      colorOut <= GREEN;
    end if;
    if(current_y > x"3F" and current_y <= x"42" and current_x >= x"1F1" and current_x <= x"205") then
      colorOut <= GREEN;
    end if;
    if(current_y > x"3F" and current_y <= x"1A5" and current_x >= x"205" and current_x <= x"208") then
      colorOut <= GREEN;
    end if;
```

B. Code for Timer & Trial

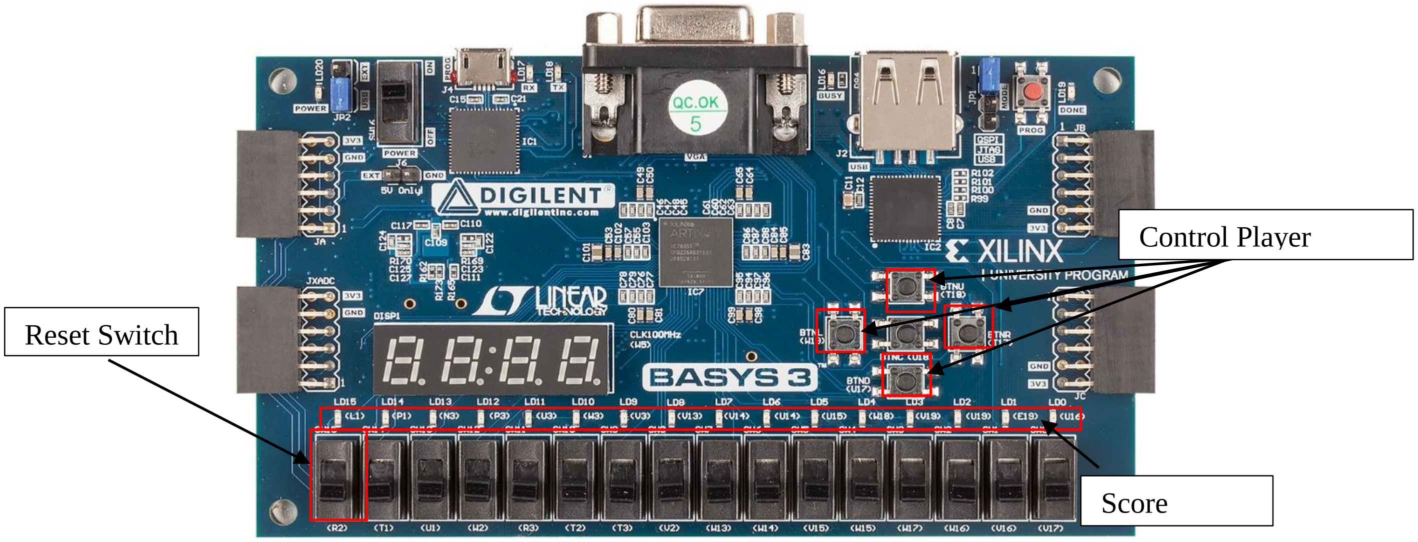
```
Count: process(clk, rst)
begin
  if(rst='1')then
    counterSec <= (others=> '0');
    top_time <= timer_trial;
    trial <= timer_trial;
  elsif(clk'event and clk = '1') then
   counterSec <= counterSec +'1';
   if(counterSec = "000000000000000000000000000") then
      top_time <= top_time - x"008";
      if(top_time = x"7B") then
        top_time <= timer_trial;
        trial <= trial-x"87";
      end if;
    end if;
  end if;
end process;
```

C. Code for Collisions

```
                if(hallway= "00000") then
          if(btnL = '1' and  bottom_player_x1 >= x"EA") then
            bottom_player_x1 <= bottom_player_x1 - '1';
          elsif(btnR = '1') then
            bottom_player_x1 <= bottom_player_x1 + '1';
          end if;
          if(btnD = '1' and  bottom_player_y2 <= x"1A5") then
            bottom_player_y1 <= bottom_player_y1 + '1';
          elsif(btnU = '1' and  bottom_player_y1 >= x"199") then
              bottom_player_y1 <= bottom_player_y1 - '1';
          end if;
          if(bottom_player_x1 >= x"1F4") then
          hallway <= "00001";
          end if;
     end if;

   if(hallway= "00001") then
            if(btnL = '1' and  bottom_player_x1 >= x"EA") then
              bottom_player_x1 <= bottom_player_x1 - '1';
            elsif(btnR = '1' and bottom_player_x2 <= x"205") then
              bottom_player_x1 <= bottom_player_x1 + '1';
            end if;
            if(btnD = '1' and  bottom_player_y2 <= x"1A5") then
              bottom_player_y1 <= bottom_player_y1 + '1';
            elsif(btnU = '1') then
                bottom_player_y1 <= bottom_player_y1 - '1';
            end if;
            if(bottom_player_x2 <= x"1F4") then
            hallway <= "00000";
            end if;
            if(bottom_player_y2 <= x"199") then
               hallway <= "00010";
               end if;
      end if;
```

4.2. Control



4.3. Contributions

	Anthony Marando	Maximus Rizk	Priyank Kashyap	Vitaliy Vorobets
Planning	✓	✓	✓	✓
Maze Design	✓	✓	✓	
Timer/ Trial Design		✓	✓	✓
Maze Collisions	✓		✓	✓
Scoring			✓	
Slides		✓	✓	✓
Report		✓	✓	