Unicor Explosion-The Game

By Changcan(Benjamin) Hu

[changcanhu@cpp.edu](mailto:changcanhu@cpp.edu)

version 1.0, 2018-11-25

## Table of Contents

## Sequential Circuit

## Implementation

## Purpose

## Logic

## Restriction

Sequential Circuit

This is a synchronous sequential circuit that uses combinational logic and more than one flip flops, where the outputs depend on the past behavior of the circuit, as well as on the present values of inputs, where clock signal is used to control the operation. In this case. The Game is a type of Finite State Machine (FSM)

Implementation

Purpose:

* The Physics Engine is responsible for receiving all the necessary initialized signals and output from the map generator to process its logic, which is depended on the user inputs and circuit stored position value then output score value for the scoring machine.

Logic:

* Position Constraint
  + User need to pass a low block in the next few phases, that requires a jump, as clock arrives at the block phase, Physics Engine checks user position based on when the user input jumps, if condition is met, Physics Engine will output Score Engine positive and jumpout positive to signal display its current position.
  + User need to go under a high block, which require user position to be on ground. Physics Engines checks user position based on its initial inputs, outputs score positive and position ground, if constraint has met, or output is\_dead positive to signal the display, the player is lost to terminate the game.
* Restriction
  + In order to control user from continuously jumping in the air in the event of avoiding any necessary low block phase, we control the setting of jump, user cannot extends jumping duration in the air, and jump will only be stored till position has reached ground level.
  + Based on the dungeon map design, there will be phases where two consecutive low blocks are built next to each other’s. Therefore, we give the user the in-air jumping duration to 2 second, so that user can jump early to avoid jumping too late from a single low block, or successful jump far enough to go over the two consecutive low blocks.

|  |
| --- |
| `timescale 1ns / 1ps |
|  | ////////////////////////////////////////////////////////////////////////////////// |
|  | // Company: Unicorn Explosion |
|  | // Engineer: Benjamin(Changcan) Hu |
|  | // |
|  | // Create Date: 11/25/2018 02:57:30 PM |
|  | // Design Name: |
|  | // Module Name: Brain |
|  | // Project Name: Uniorn Explosion |
|  | // Target Devices: NEXYS4-DDR |
|  | // Tool Versions: Artix-XC7A100T-CSG324 |
|  | // Description: FPGA |
|  | // |
|  | // Dependencies: |
|  | // |
|  | // Revision: |
|  | // Revision 0.01 - File Created |
|  | // Additional Comments: |
|  | // |
|  | ////////////////////////////////////////////////////////////////////////////////// |
|  |  |
|  |  |
|  | module physics\_engine( |
|  | input clk, |
|  | input start, |
|  | input [15:0] map, |
|  | input jumpin, |
|  | output reg is\_dead, |
|  | output reg score, |
|  | output reg jumpout |
|  | ); |
|  | reg [1:0] position; |
|  | reg [1:0]Q; |
|  |  |
|  | always@(posedge clk) |
|  | begin |
|  | if(~start) |
|  | begin |
|  | is\_dead = 0; |
|  | end |
|  | else |
|  | begin |
|  | case(position) |
|  | 0: |
|  | begin |
|  | if(jumpin==1) |
|  | begin |
|  | position<=2'b10; |
|  | end |
|  | else |
|  | begin |
|  | position<=2'b00; |
|  | end |
|  | end |
|  | 1: |
|  | begin |
|  | position<=2'b00; |
|  | end |
|  | 2: |
|  | begin |
|  | position<=2'b01; |
|  | end |
|  | 3: |
|  | begin |
|  | position=2'b01; |
|  | end |
|  | endcase |
|  |  |
|  | case(map[15:14]) |
|  | 0: //No block |
|  | begin |
|  | score=0; |
|  | end |
|  | 1: //Low block |
|  | if(position > 2'b00) |
|  | begin |
|  | score = 1; |
|  | end |
|  | else |
|  | begin |
|  | is\_dead=1; |
|  | end |
|  | 2: //High block |
|  | if(position==0) |
|  | begin |
|  | score = 1; |
|  | end |
|  | else |
|  | begin |
|  | is\_dead=1; |
|  | end |
|  | 3: //Don't exist |
|  | begin |
|  | score = score; |
|  | end |
|  | endcase |
|  | end |
|  | end |
|  |  |
|  | always@(position) |
|  | begin |
|  | if(position == 0) |
|  | jumpout = 0; |
|  | else |
|  | jumpout = 1; |
|  | end |
|  |  |
|  | endmodule |