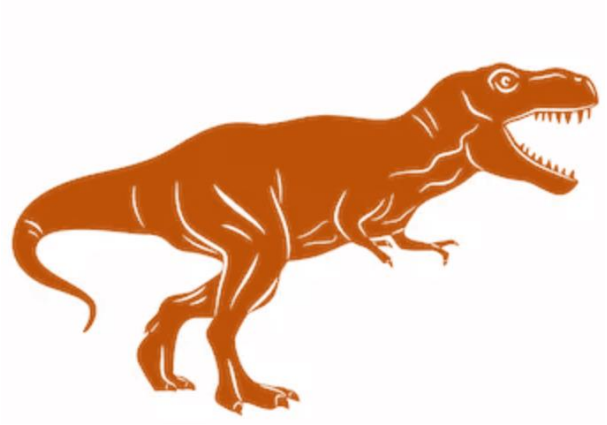
EECS 2070 02 Digital Design Labs 2019 Group04 Final Project
學號：104021215　姓名：熊磊



Amazing Dinosaur

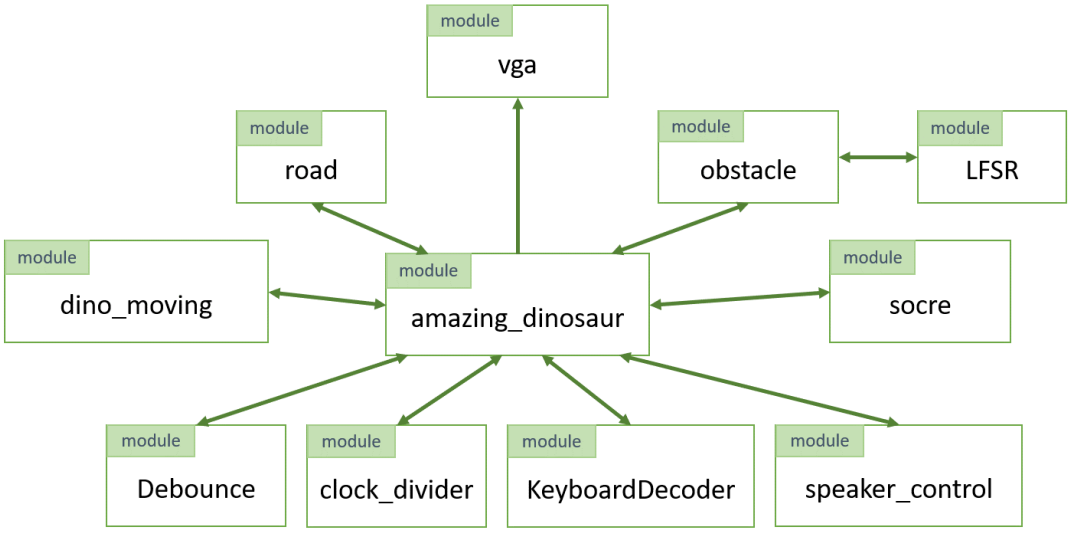
1. IDEA

   In this project, I implement a dinosaur running game on FPGA board. The rule of the game is that the dinosaur must avoid obstacles, which are randomly generated. I consider several parts of the game to finish this project, i.e., the **body of the dinosaur**, different **obstacles**, **jumping** scheme, **scoring** scheme, running **speed**, **music**, **led indication**, **vga control**, and the **collision detection**.

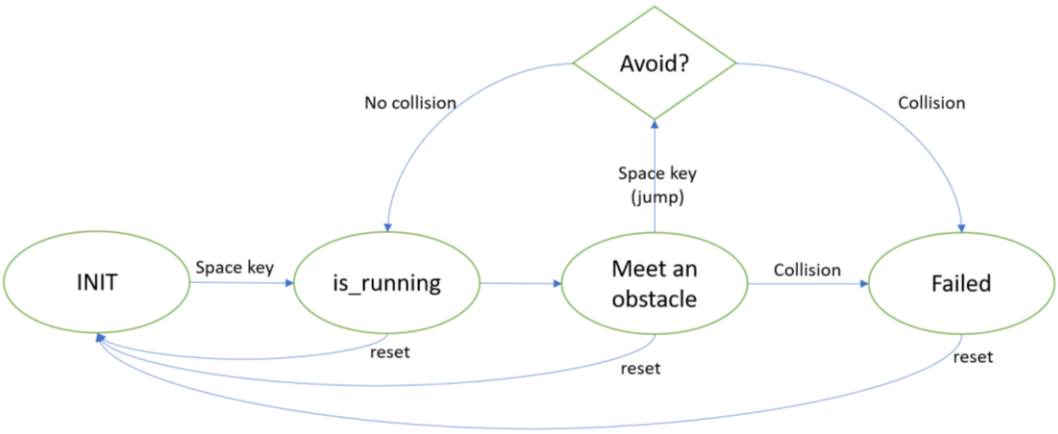
2. Detail and Block Diagram

   First I list the relationship of modules I used, and I will explain main function of each modules. In this project, I used eleven types of modules.

   - The relationship between each module is followed by this diagram:



- The dinosaur has following status:



Amazing Dinosaur Module

This module is the main module. It collects each submodule's pixel signal (i.e. *px_dinosaur, px_ground, px_obstacle, px_scoring*) and transmit the signal to ***vga*** module.

- Detect Collision
  If *px_dinosaur* and *px_obstacle* has both value 1 at the same time, which means a collision occurs, then *stop_trigger* will be set to 1.
- Analyze the keyboard signal
  Determine if the player hit the "space" key on key board or not.
- Start or Reset the game
  In the beginning, one can use "space" key on keyboard to start the game.
  If the game is stopped by *stop_trigger* or just willing to restart the game, one can hit the center button on FPGA board to reset.
- Sending Jump signal
  When the dinosaur is running, one can use the "space" key on keyboard to make the dinosaur jump.
- Music Selection
  Before the game start, there is no music playing. While the player starts the game, the music 1 will be played. If the player trigger *jump*, the FPGA will play a jumping sound effect, which is another music track. When the dinosaur hit an obstacle, all music will stop.
- 7-segment and LED
  Transmit the *score* signal to 7-segment Display.
  LED will indicate whether the dinosaur is running, hit, or not begin yet.
- VGE Display
  Transmit the *score* signal to 7-segment Display.

dino_moving Module

This module is used to output Dinosaur Body Pixel and control Jumping scheme. It will decide HOW high and HOW long it should jump. Notice that the jumping must happened when the dinosaur was on the road, we used *jump_time* and *is_jumping* to detect if the dinosaur finished last jumping and running on the road or was still up in the air. Then transmit *px_dinosaur* to *amazing_dinosaur* for displaying.

```
40      always @(negedge frame) begin
41          counter <= counter+1;
42          if (is_running) begin
43              if (jump && is_jumping==1'b0) is_jumping<=1'b1;
44              if (is_jumping) begin
45                  if (jump_time>=12'd40) begin
46                      jump_time<=12'b0;
47                      is_jumping<=1'b0;
48                  end
49                  else jump_time<=jump_time+1'b1;
50              end
51          end else begin
52              if (rst || start) begin
53                  jump_time <= 12'b0;
54                  is_jumping <= 1'b0;
55                  counter <= 4'b0;
56              end
57          end
58      end
```

obstacle

To generate a random position of obstacle onto the road. To accomplish this function, I used LFSR to generate a random number for *waiting_time*. I used a counter until counting to the *waiting_time*. Once the counting finished, it will generate an obstacle on the road by transmitting *px_obstacle* to *amazing_dinosaur* for displaying.

```
47      always @(negedge frame) begin
48          if (my_rst) begin
49              position <= 10'b0;
50              wait_counter <= 10'b0;
51              wait_time <= 10'b0;
52          end
53          if (is_running) begin
54              if (position == 10'b0) begin
55                  if (wait_counter == 10'b0) begin
56                      wait_time <= rand * 10'd7;
57                      obstacle_sel <= rand % 3;
58                      wait_counter <= 1;
59                  end else begin
60                      wait_counter <= wait_counter + 1;
61                      if (wait_counter >= wait_time) begin
62                          position <= screen_speed;
63                          wait_counter <= 10'b0;
64                      end
65                  end
66              end else begin
67                  position <= (position + screen_speed) % 700;
68              end
69          end
70      end
```

LFSR

Generate a *pseudo* random number for *waiting_time of obstacle.*

```
module LFSR(random, clk, rst);
    input clk;
    input rst;
    output [3:0] random;
    reg [3:0] random;

    always @(posedge clk or posedge rst) begin
        if (rst == 1'b1) random[3:0] <= 4'b1000;
        else begin
            random[2:0] <= random[3:1];
            random[3] <= random[1] ^ random[0];
        end
    end
endmodule
```

road

This module is used to generate the ground layout and control HOW Fast the screen is moving. The road should move from right to left repeatedly if *is_running* == 1. Also, output *px_road* to *amazing_dinosaur* for displaying road .

```
    always @(negedge frame) begin
        if (is_running) position <= (position + screen_speed) % 10'd160;
        else position <= 10'b0;
    end
```

keyboard

This module is from TA's providing IP module.

score

When the dinosaur is running, the score will auto increase by 1 in a given period. And output px_score to *amazing_dinosaur* module for displaying score .

```
always @ (posedge clk or posedge rst or posedge start) begin
    if(start||rst) begin
        score_data <= 0;
        score_data_cnt <= 0;
    end
    else if(is_running==0) score_data <= score_data;
    else if(score_data_cnt < 1000000) score_data_cnt <= score_data_cnt + 1;
    else begin
        score_data_cnt <= 0;
        if(score_data[3:0] == 4'b1001) begin
            score_data[3:0] <= 0;
            if(score_data[7:4] == 4'b1001) begin
                score_data[7:4] <= 0;
                if(score_data[11:8] == 4'b1001) begin
                    score_data[11:8] <= 0;
                    if(score_data[15:12] == 4'b1001) begin
                        score_data[15:12] <= 0;
                    end
                    else score_data[15:12] <= score_data[15:12] + 1;
                end
                else score_data[11:8] <= score_data[11:8] + 1;
            end
            else score_data[7:4] <= score_data[7:4] + 1;
        end
        else score_data[3:0] <= score_data[3:0] + 1;
    end
end
```

audio

This module used to play music when dinosaur is running, also it will play jumping sound.

vga

Only output one color on screen for simplicity, and output when px == 1, which means *px_road* == 1 or *px_dinosaur*==1 or *px_obstacle*==1 or *px_score*==1.

```
56      assign r = (tmp) ? 4'h0 : px ? 4'b0000 : 4'b1111;
57      assign g = (tmp) ? 4'h0 : px ? 4'b0000 : 4'b1111;
58      assign b = (tmp) ? 4'h0 : px ? 4'b0000 : 4'b1111;
```

Use SW[0] to decide whether turning on the vga signal.

```
37      always @ (posedge clk or negedge vga_enable) begin
38          if (!vga_enable) v_count <= 10'h0;
39          else if (h_count == 10'd799) begin
40              if (v_count == 10'd524) v_count <= 10'h0;
41              else v_count <= v_count + 10'h1;
42          end
43      end
```

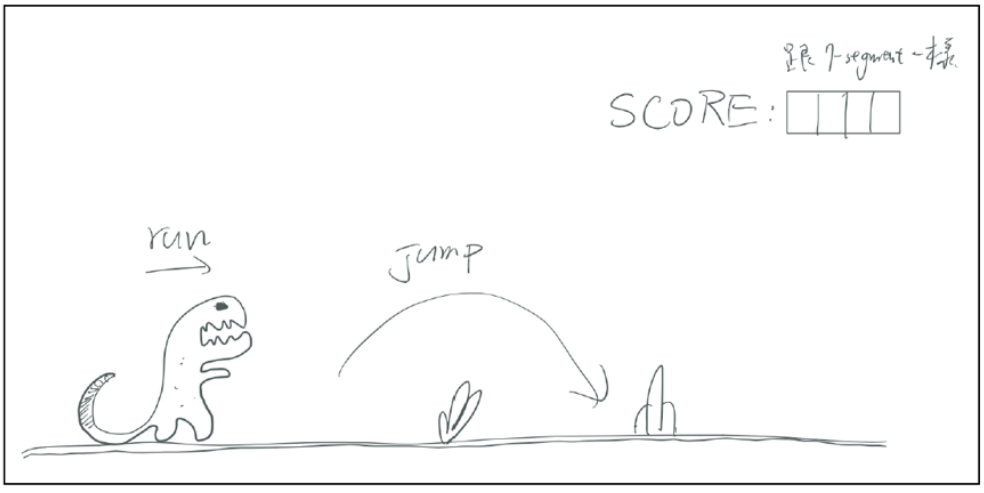
3. Completeness

Based on my proposal, I accomplished almost 90% functions. However, the sound effect for jumping was not like in real world. It should be able to be refined.

Project Description:

1. **概念圖、功能描述與使用到的 I/O Devices 或額外的機構設計**

我預計設計一個恐龍遊戲，並且包含音效、進入、結束等畫面。

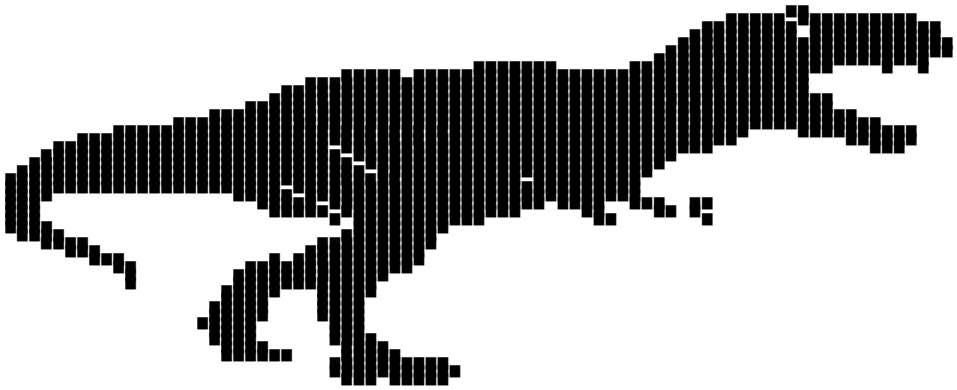


- Switch 用來控制是否啟動遊戲，輸出 VGA done
- LED 用來顯示目前的狀態（初始、遊戲中、失敗） done
- Push button 用來 Reset done
- 7-segment Display 可以顯示目前的分數 done
- 鍵盤 用來控制恐龍可以跳起來避開障礙物 done
- Audio 遊戲背景音樂 和 失敗的音效，還有恐龍跳起來的音效 80% done
- VGA 輸出遊戲畫面 done

4. Challenges
   - When I try to generate obstacles on the road, I have no idea how to do it completely random. I found, however, if I using LFSR it will sometimes generate a consecutive sequence of obstacles, which makes dinosaur cannot avoid them by single jumping.

     I extend the time by multiple it by a number, hence, it ensures a certain distance between two obstacles.

   - To have high resolution via vga, I choose not display colorful picture on screen, instead, all color on screen are in dark gray, which is decided by 0 or 1.



   - Determine clock_divider for Keyboard, jump and vga is a tough process. Since I have to synthesis for a long time the check whether it works. But finally it looks great.

5. Final Thoughts
   This is really a good course and I also be more familiar with Verilog by accomplishing this project. Thank to TAs and the professor.

6. References

   [1] Google ChromeDino game chrome://dino/
   [2] https://github.com/wayou/t-rex-runner
   [3] https://github.com/topics/dinosaur-game