Team 6	**Project: Plinko Game**
Name: 鄭聰明 **Jason Theodorus**	**ID:109006236**
Name: 鐵特德 **Tuguldur Tserenbaljir**	**ID:109006271**

TABLE OF CONTENTS

I. Introduction 2
II. Motivation 3
III. Budgeting 3
IV. System Specification 3
V. Problems and their solutions 7
VI. Experimental Results & Design 12
VII. Block and State Diagram 16
VIII. Proposal and Final Project Comparison 17
IX. Future Improvements 18
X. Conclusion 18

I. Introduction

For This year's final project for Logic Design Laboratory Class, we decided to make an "**Arcade Version of the Plinko Game".** In a nutshell, a Plinko game is played on a large board with a series of barriers arranged in a triangular shape, After that, a player drops an object, which in our case is a ping pong ball, and bounces down the board, hitting the barriers until it falls into a slot. Where each space corresponds to a different prize, where the more centered the slots, the higher the rewards. The goal of this game is for the player to land in a high-reward slot. The game itself is often seen in night markets, amusement parks, and carnivals.

Since the big picture of our final project is an arcade machine of the Plinko game, we must take money for it to be able to start running, or give out the rewards. Where the reward for playing is as we so call "**tickets**". Where on arcades they can be traded for counter rewards.

II. Motivation

Our main motivation to create an arcade version of Plinko is because it is actually fun and addicting, and it can relieve stress because it is so satisfying to watch the balls go through the barriers.

Second, Plinko has been around since the 1980s making it the most popular game that has prizes. But, it has been a while since Plinko appeared in the community, so we would want to promote

the more advanced, or we can say “Arcade Version of Plinko” that has adapted to the modern world.

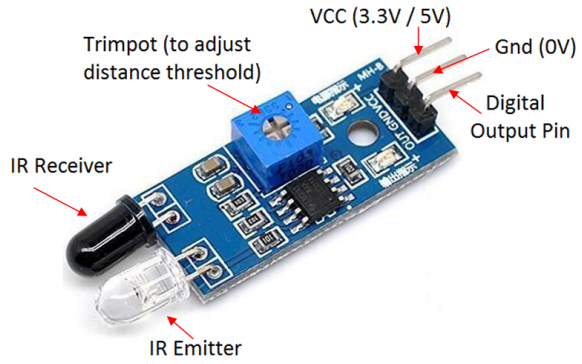
Third, It is fun for all ages. The game Plinko was popularized by the TV-Series Stranger Things, and frankly, we both knew Plinko from that certain episode. We wanted to play Plinko, but we didn’t have the board, and it is very hard to find them, as it becomes more rare each day.

III. Budgeting

Item x Quantity	**Price /u**	**Total**
IR Sensor x 6	40 NTD	240 NTD
Servo Motor x 1	130 NTD	130 NTD
Infraboard x 2	40 NTD	80 NTD
Miscellaneous	100 NTD	100 NTD
TOTAL		550 NTD

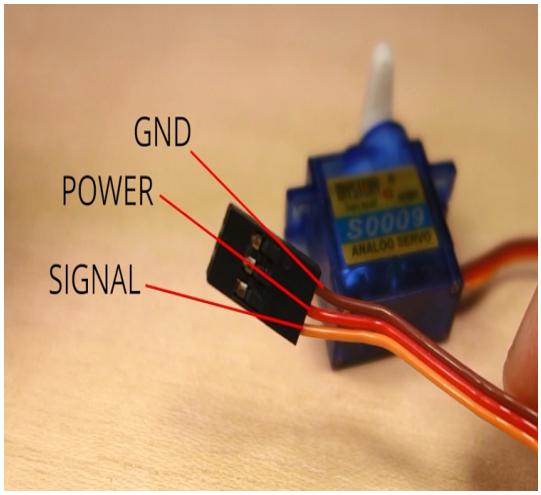
IV. System Specification

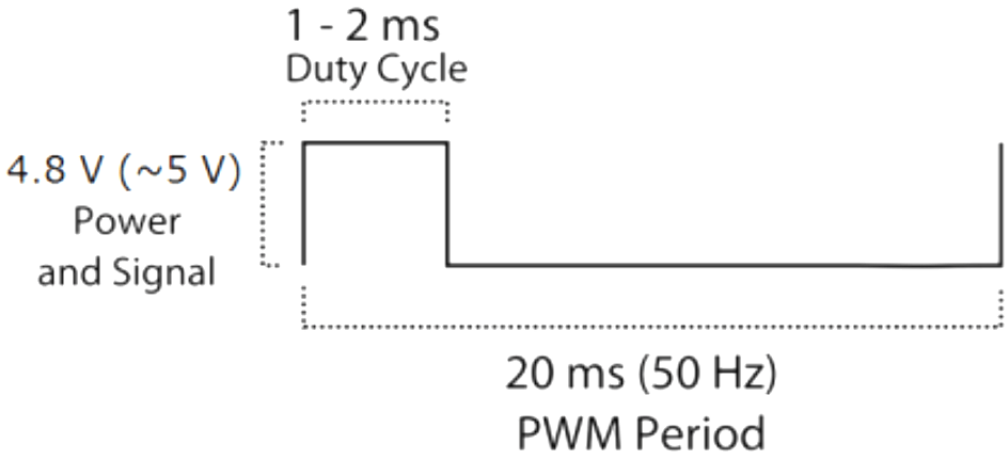
IR sensor (X6): Infrared Sensor Module



- VCC Pin is power supply input (connect to VC of FPGA)
- GND Pin is power supply ground (connect to GND of FPGA)
- OUT is an active-high o/p – 3.3V & 5V (output Pin)
- Range is up to 20 centimeters
- The supply current is 20mA

Motor(X1): MG90s





- Red wire is the power supply input (connect to VCC of FPGA)
- Brown is the power supply ground (connect to GND of FPGA)
- Signal is an active-high o/p (output Pin)
- Rotation: 0°-180°
- Operating Voltage: 4.8V to 6V
- Operating speed is 0.1s/60° (4.8V)

Wire(at least X21): Male to Female

- 40P color jumper wire
- Compatible with 2.54mm spacing pin headers.
- Pin to pin

V. Problems and their solutions

Since we had limited budget, and our sensors were already so expensive. So we decided to go DIY with our project, and spend more of the money on the electronics which are the IR Sensors and the Motor.

To do our project couple measures have been taken to achieve our final prototype product. Where we divide it into 5 steps :

1. Detecting the Coin

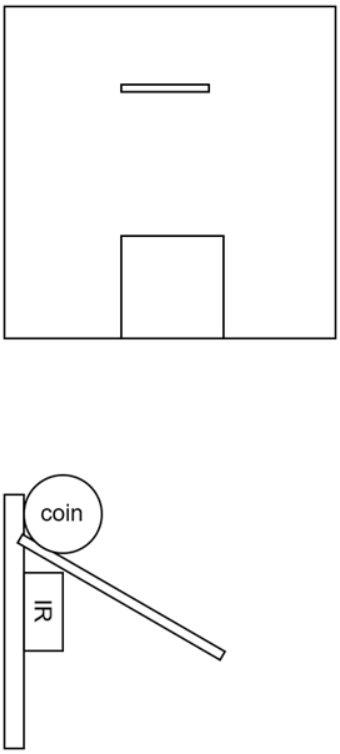


Figure 5.1.1 Top = Front Look of the **Ball Dispenser** , Bottom, IR Sensor to detect the coin inserted laying flat.

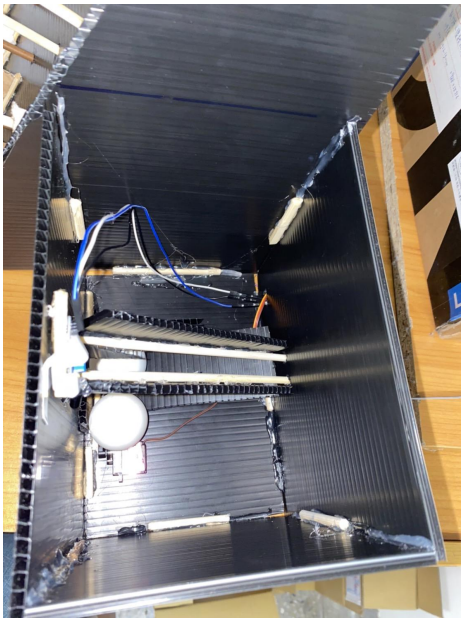


Figure 5.1.2 Prototype Ball Dispenser

Basically, We made a coin slot, for the coin to make way and store it in using chopsticks, and once the coin is inserted it will be detected by the IR Sensor at the top, which captures light, and since there is a difference on the light sensitivity, therefore it detects a track signal.

2. **Dispensing the Ball**

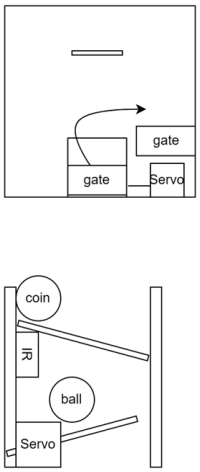


Figure 5.2.1 Inside look of the Ball Dispenser

During the ball dispensing, we made a gate using servo motor that can lift up. So it lets the ball roll out. To move the servo motor itself, the signal is caught from the IR, So once the the coin has been detected the gate will open, and the player can get the ball to play Plinko.

3. Creating the Plinko Board

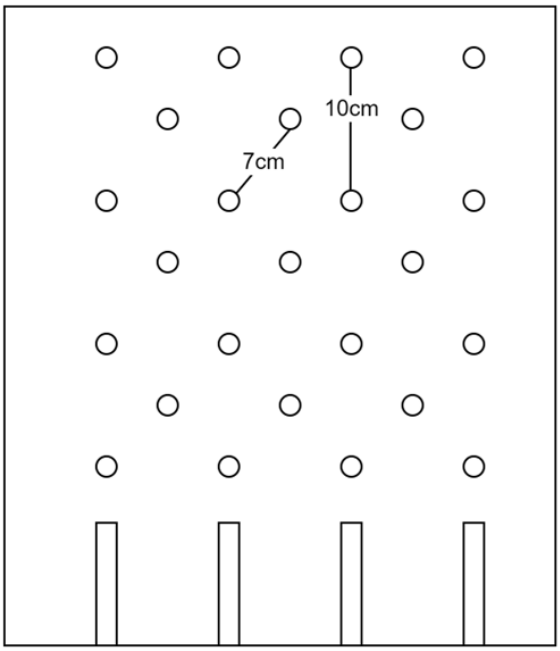


Figure 5.3.1 Plinko Board Schematic

Making the plinko board itself is quite challenging, because we had to calculate for each dot, where we are going to put the barrier, and it is very mathematical and you have to be very careful when you place the dots. Otherwise, if we are not careful and the dots are misplaced, the pingpong ball will not be able to go through, and resulting in failure of the mechanism of the game.

As for the small circles in Figure 5.31, we used a series of chopsticks where we drill holes and stab into. We thought of what we're going to use but we find that chopsticks are the best, because it is strong and not wobbly. As predicted, it turns out it works really great.

Figure 5.3.2 The Plinko Board with Ball Dispenser attached to the right and IR Sensors at the bottom

4. **The Gameplay of The Plinko**

   To support the gameplay of the plinko which gives out rewards. Since we are an arcade version of Plinko we have to make it digital. Where on the conventional plinko it is just paper, well on Logic Design Lab Class we have to do more than that. So, we decided on each slot at the bottom, we put IR Sensors, so we can detect where the ball lands.

   Since there are 5 slots, so we use 5 IR Sensors to detect the ball. Look at **Figure 5.3.2**, Even though the IR Sensors were quite fragile, and there has been multiple problems about the sensor where it was detecting objects even though there were nothing, but we were able to fix it.

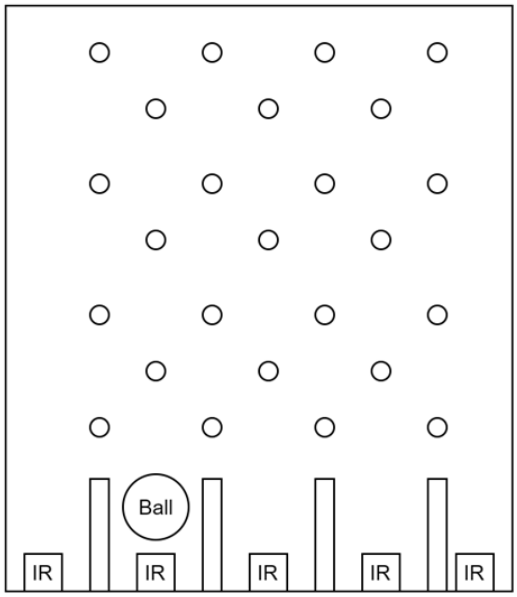


Figure 5.4.1 IR Sensors at the bottom

5. The Scoring System of the Plinko

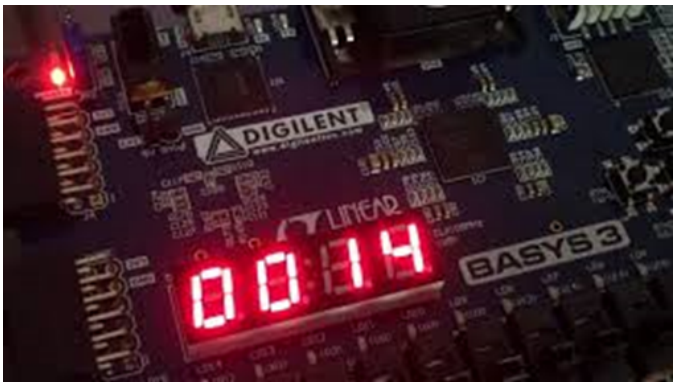


Figure 5.5.1 7-Segment Display of Basys-3 FPGA

Arcades have a very interesting way of giving rewards which attracts customers into playing their machines, which are tickets. Where these tickets later on can be converted into redeemable rewards like Playstation or Xbox and many other things.

So for the scoring system of our arcade plinko, we also deciced to use a ticketing kind of simulator. For example, Lets say the ball landed in the middle slot where the score is 50, so 50 tickets will be given, so the 7-segment display will decrement from 50 1 by 1 until the end goal which is 0.

VI. Experimental Results & Design

Before putting every module together we coded and tested them separately. Namely IR sensor, motor, and motor with IR sensor.

To start our project we coded our IR sensor first, making sure that the IR sensor that we selected worked with our FPGA board. In addition, we learned from online open source but with the Arduino board. We learned that the IR will only take one input but GND and VCC. Thus, we realized from testing with the display that unless we save the value of the IR sensor, the sensed value will keep updating due to the clock and system design of the IR sensor. From then, we knew that we should create an FSM and make it stuck in a state where the value does not change.

Secondly, for the motor, we had no idea how it works at first. After searching online and reading the data sheet of the servo motor, we found some online sources from youtube, GitHub, and so on. We tested the servo motor source code and we saw it turns 180 degrees from the original position and back to the original position which is counter 180 degrees from the changed position.

Finally part before putting everything in a top module for our Plinko game, we need to implement the previous servo motor open-source code to our design. Whereas our servo motor will be enabled when the sixth IR sensor in the coin machine detects a coin. This was one of the challenging parts of our project because after the IR sensor detects the coin to enable the motor, the motor only rotates 180 degrees from the original position rather than rotating 180 degrees and counter 180 degrees. After testing numerous code designs for our servo motor, we were left with no choice but to manually counter 180 degrees instead of the servo motor.

Since all of the modules for our top module were done, it was time to code and design the final project. Firstly, we give inputs for the sensors namely track 1 to track 6 (1 to 5 for Ball, 6 for motor), inputs of the clock, reset, and output 4-bit anode, 7-bit cathode (4 digits 7 segment display), and motor_servo for the motor. Secondly, before configuring the modules with wires and registers we create a reset button using debounce and one pulse modules from previous labs. The purpose of the reset button is to have a safe option to reset everything in the code with care.

```
        debounce DB1(.s(rst), .s_db(rst_db), .clk(clk));
        onepulse OP1(.s(rst_db), .s_op(rst_op), .clk(clk_d22));
```

Then, it was time to design the configure the parts we implemented into one whole code. Namely, apart from the reset button, our top module will consist of 4 modules and they are sensor1 (for bottom 5 sensors), sensor6(for coin detection), motor(opening dispenser box), and anothecathode (printing out on the 4 digits 7 segment display).

Starting with sensor1, its purpose is to save the corresponding score and give it to anothecathode module when either of the five senses detects the ball.

```
sensor1 SS1(.clk(clk), .rst(rst), .track1(track1), ...
.track5(track5),.score(score_dummy));
```

Just like the traffic light controller from the previous lab, we set up the FSM by case statement with next_state and state where we set the parameters for S1 and S2 to 3'b000 and 3'b001. In the first S1 state, we check whether either pair of track1, track5, and track2, track4 because their score is the same 10 and 20 respectively. In contrast, the track5 score is alone 50 because of our hardware.

```
                    if(!track1 ||!track5)begin
                       next_state = S2; //10
                       next_score = 7'd10;
```

We use the if statement to check for the !track instead of track for all of the sensors because the bits are switched just like the anode where 0 means on and 1 means off. Furthermore, after setting the respective score to the next_score, we move to the next state. At first, our idea was to make it so that it stays in the second state when the ball is sensed (shown below) and when the reset button is pressed it sets back to 0 and waits for the next ball sense.

```
                S2:begin
                        next_state = S2;
                        next_score = score;
               end
```

However, talking with each other we decided to create an arcade experience where the 4 digits display will display the number of tickets to give out to the player and set back to zero. So the purpose of the second state of this module was to output the score to anothecathode and countdown to 0 as it is outputting the score. At first, we thought it is easy as we implemented our code (shown below).

```
                S2:begin
                    if(score > 4'd0)begin
                        next_score = score - 1'd1;
```

```
                        next_state = S2;
                    end else begin
                        next_score = 4'd0;
                        next_state = S1;
                    end
                end
```

But we were running into bugs where the score did not show up or the score is not changing from 0 or the score is just 10 and numbers from our scoring table. After debugging for days, we found out that forgot to change the clock cycle of this always block where the outputting score is too fast for the human eye to see even though we updated our clock cycle in the anodecathode. In addition, the reason why the score was that the IR emitter was too close to the IR receiver (pictures IR emitter and receiver shown above in specs)where it kept sensing and outputting the score.

The second module for our top module was the sensor6 module which it sends enables the motor to give out the ball when a coin is entered into the coin dispenser (the flag variable is a 1-bit wire).

```
        sensor6 SS2(.clk(clk), .rst(rst), .track6(track6),
.flag_motor(flag));
```

The sensor6 module works similarly to the sensor1 module where it also uses FSM and an IR sensor to detect the coin. However, the main difference is that it will give an output of 1 or 0 to enable the motor in the motor module. Thus, it also has a counter to count clock cycles. Our first initial thought was that keep the enable enabled for enough time so that the motor can turn 180 and counter 180 degrees. But the motor only turns 180, not 180, and the counter 180. It leaves us the option to do the counter 180 degrees manually. So, we could say this state of the FSM is there just in case.

```
                S2: begin
                    //enable the motor 69 countdown
                    if(count >= 69)begin
                        next_state = S1;
                        next_flag_motor = 1'b0;
                        next_count = 0;
                    end else if(count < 69) begin
                        next_count = count + 1;
                        next_state = S2;
                        next_flag_motor = 1'b1;
                    end
                end
```

The second to last module of our top module is the motor module where it will take the enable(flag) from the sensor6 module and have clk and output motor_servo.

```
motor M1 (.mclk(clk) , .flag_m(flag), .servo(motor_servo));
```

Our implemented source code has 20-bit counter, 1-bit servo_reg, and control registers.We set the max value of count registers counting up to be 'd999999 and after it reaches the max it will be set back to zero to control the position of the servo motor. The servo_reg register stores the current state of the servo motor and run_servo stores the value of this module's output which is the servo. Thus, the counter is compared with the value ('d50000 + control) to set the value of the servo_reg. In addition, the second always block uses the flag given by the senso1 module to enable and disable the value of the run_servo register.

Given our option to manually turn the motor counter 180 degrees, we tried our last test to give another input of reversed flag to the module. Our idea was that if the flag is first enabled and turns 180 degrees then reverse the flag and turn the counter 180 degrees. However, we were not able to succeed where our final choice was to manually turn the motor counter 180 degrees.

The module of our top module is the anodecathode which outputs the score values to the 4 digits 7-segment display.

```
anodecathode disp0(.score_an(score_dummy), .clk(clk),
.rst_n(rst), .anode(anode), .cathode(cathode));
```

Similar to our PingPong lag, we reused our code with implementation. We only use the first two digits of the 4 digits in the seven segments and output the corresponding numbers by /10 and %10 the score we get from the sensor6 module. Then, the one-bit digit is passed on to the cathode module which selects the respective 8-bit value to display.

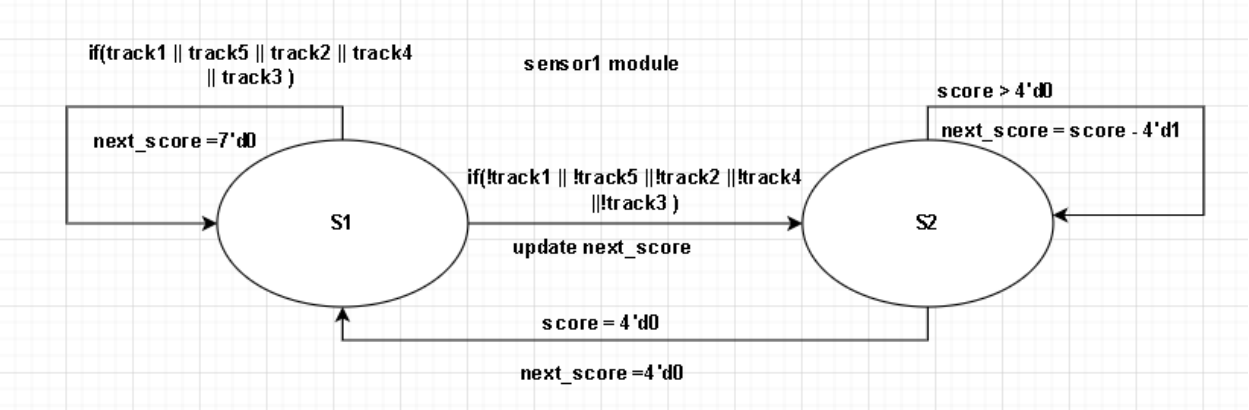
```
2'b10: begin //1's value
    anode = 4'b1011;
    digit = score_an%10;
end
2'b11: begin //10's value
    anode = 4'b0111;
    digit = score_an/10;
```

In conclusion, our final project consist of 4 main modules.

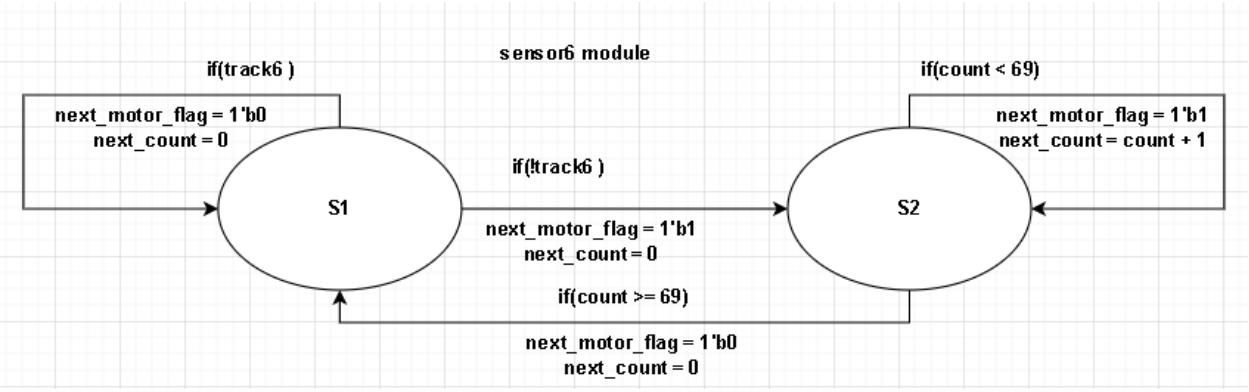
```
sensor1 SS1(.clk(clk),
.rst(rst),.track1(track1),.track5(track5),.score(score_dummy));
sensor6 SS2(.clk(clk), .rst(rst), .track6(track6), .flag_motor(flag));
```

```
motor M1 (.mclk(clk) , .flag_m(flag), .servo(motor_servo));
anodecathode disp0(.score_an(score_dummy), .clk(clk), .rst_n(rst),
.anode(anode), .cathode(cathode));
```

VII. Block and State Diagram



(Figure 7.1 sensor1 module FSM)



(Figure 7.2 sensor6 module FSM)

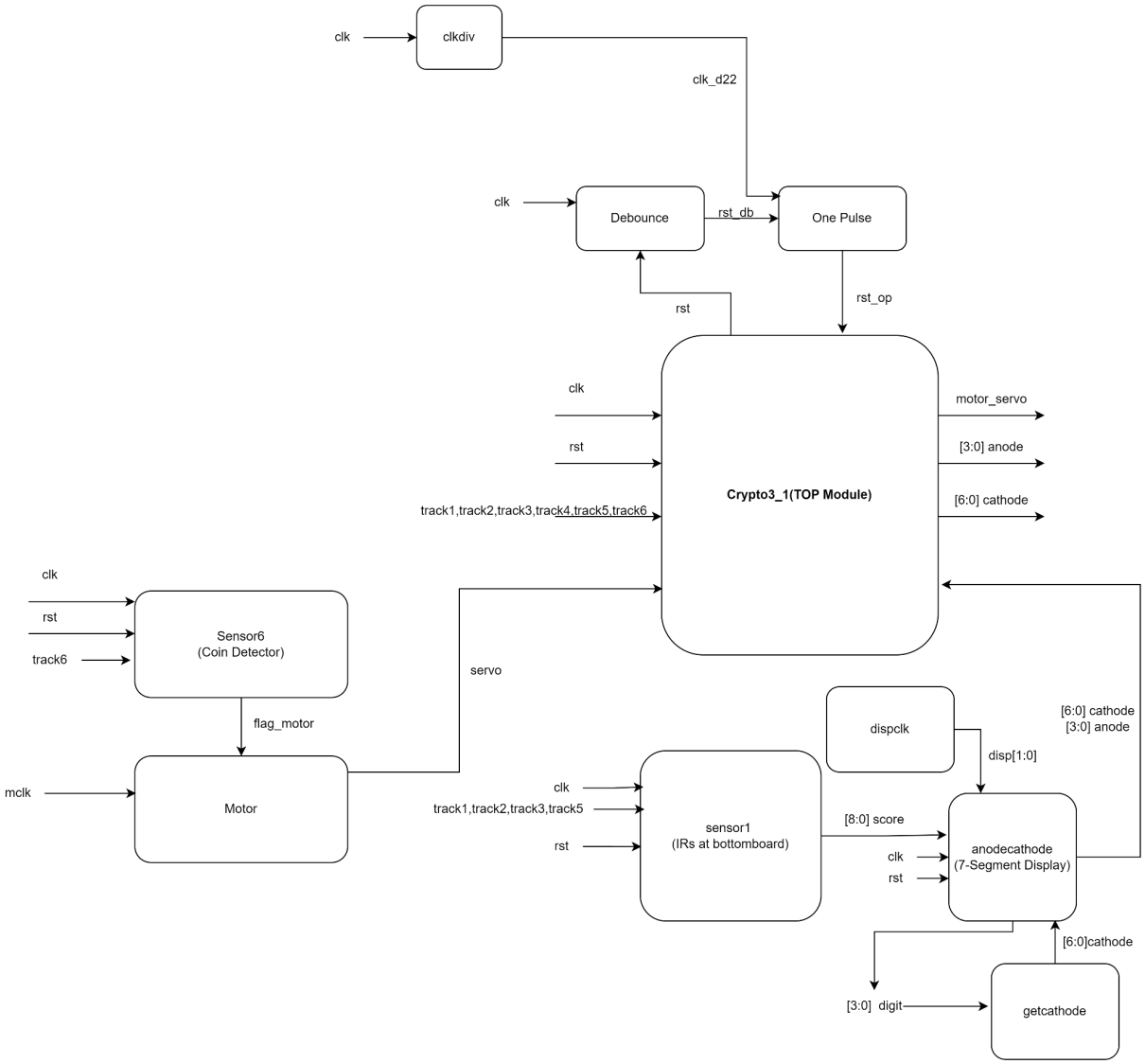


Figure 7.3 Block Diagram of Plinko Game

VIII. Proposal and Final Project Comparison

Although the proposal and final project are alike, there are a few differences. After I mention what we accomplished from our proposal, I will mention the differences.

Firstly, our picture of the design stayed the same as well as the timeline for finishing various parts of the project. Thus, we figured out how to design the coin detection. We used another IR sensor to detect the ball and a motor to give out a ball to play. Finally, when we had trouble we asked TAs during Thursday lab hours.

On the other hand, we were not able to fit the music playing and LED for the Plinko game. As we connected a motor, we ran out of PMODs so we decided to use the last PMOD for

the LED instead of the music. Theoretically, turning on a LED using FPGA is possible but it was too difficult on our part because we were stuck debugging our code to make it better most of the time and there is scarce source code online.

IX. Future Improvements

If we would make future improvements to this prototype of the Arcade Plinko Game, we would want to make :

1. **Automated System**
    - Which means that once you inserted the coin, you just have to watch where the ball goes, and you don't need to manually drop the ball. But, once we rethinked about this it removes the satisfying element of plinko, because most people get satisfaction from dropping the ball and watching the ball go to the bottom slots.
2. **Improve Coin Detecting**
    - We have researched online and found out that there is a sensor that can detect metal, which is the Inductive Proximity Sensor, where it can sense metallic elements, like coins. Currently, in our prototype the coin detector can sense any object, including non-coin looking things, and it is really faulty after lengths of use.
3. **Create Ticketing System**
    - Since the ticketing system is just a simulation of a ticket dispenser. We can improve our prototype by actually creating a ticket dispenser, where it will dispense tickets, but we find that it is really tricky, since we have to literally create **"tickets"** and create a **"conveyor"** to roll out the tickets.
4. **Improve the Gate**
    - Implement the servo motor code that it does not need manual help to turn counter 180 degrees.

X. Conclusion

Logic Design Laboratory has been a really fun class for both of us. Especially, during the final project where we finally get to make what our hopes during 1st year would be, creating something that exists and is usable. Although, on this semester we have our ups and downs, but we always keep our hopes up that we could try to do it, and actually do it.

We want to thank you Professor Lee and the TAs for this wonderful semester, and we hope that we will see you guys again in the future!