Design Assignment #2 Report

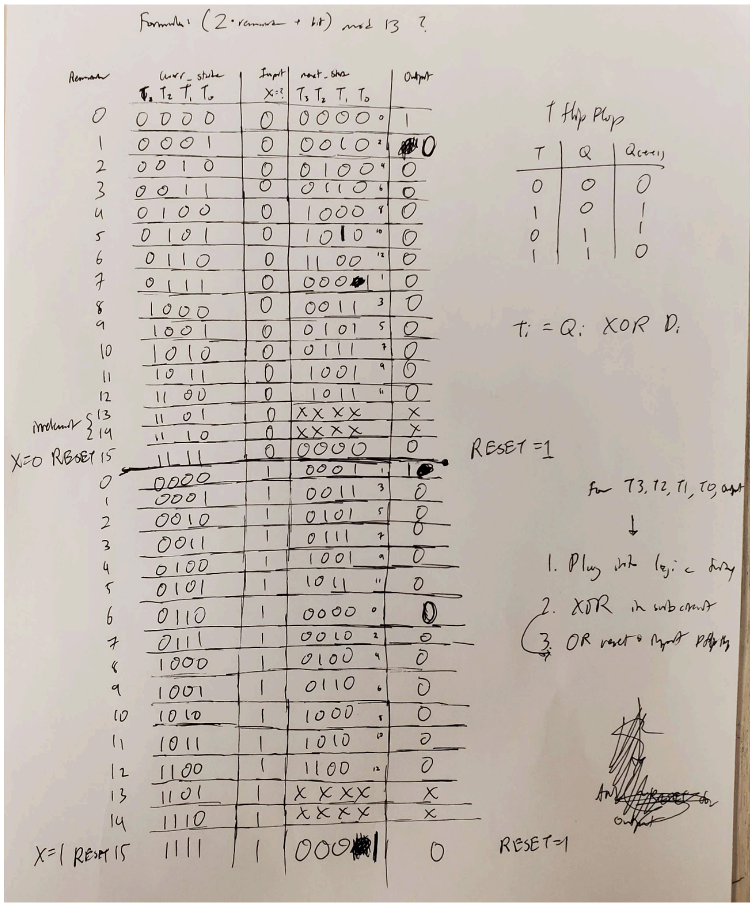
Student Name: Arthur Zhou

Task 1: Detecting Unlucky Numbers

Controller FSM Design

*My approach for the design was to utilize 4 T flip flops to be able to store the states 0 to 12 for the remainder of the value after being divided by 13. The controller would solve the equation of (2 * remainder + input_bit) mod 13 and find the remainder after to determine which state to enter after given a bit. The states were represented by 0000 for 0 remainder, 0001 for 1 remainder, … 1100 for 12 remainder. 1101 and 1110 were marked as "don't care" values while 1111 was marked as the reset. I would implement the reset by OR-ing the final result from each of the T flip flop next state equations with the reset bit. If the reset bit were to be one we would be transported to the reset state which is 1111 which would give us the proper outputs and the proper next state depending on the input.*

State Transition Diagram



Formula: (2·remainder + bit) mod 13 ?

Remainder	curr_state T_3 T_2 T_1 T_0	Input X=?	next_state T_3 T_2 T_1 T_0		Output
0	0000	0	0000	0	1
1	0001	0	0010	2	0
2	0010	0	0100	4	0
3	0011	0	0110	6	0
4	0100	0	1000	8	0
5	0101	0	1010	10	0
6	0110	0	1100	12	0
7	0111	0	0001	1	0
8	1000	0	0011	3	0
9	1001	0	0101	5	0
10	1010	0	0111	7	0
11	1011	0	1001	9	0
12	1100	0	1011	11	0
irrelevant 13	1101	0	XXXX		X
irrelevant 14	1110	0	XXXX		X
X=0 RESET 15	1111	0	0000		0
0	0000	1	0001	1	1
1	0001	1	0011	3	0
2	0010	1	0101	5	0
3	0011	1	0111	7	0
4	0100	1	1001	9	0
5	0101	1	1011	11	0
6	0110	1	0000	0	0
7	0111	1	0010	2	0
8	1000	1	0100	4	0
9	1001	1	0110	6	0
10	1010	1	1000	8	0
11	1011	1	1010	10	0
12	1100	1	1100	12	0
13	1101	1	XXXX		X
14	1110	1	XXXX		X
X=1 RESET 15	1111	1	0001		0

T flip flop

T	Q	Q(t+1)
0	0	0
1	0	1
0	1	1
1	1	0

$T_i = Q_i$ XOR D_i

RESET =1

For T3, T2, T1, T0, output

↓

1. Plug into logic diagram
2. XOR in subcircuit
3. OR reset & input [illegible]

RESET=1

Although I did not explicitly use a state transition diagram, I drew out the state table and used it to consistently achieve my results.

State Transition Table

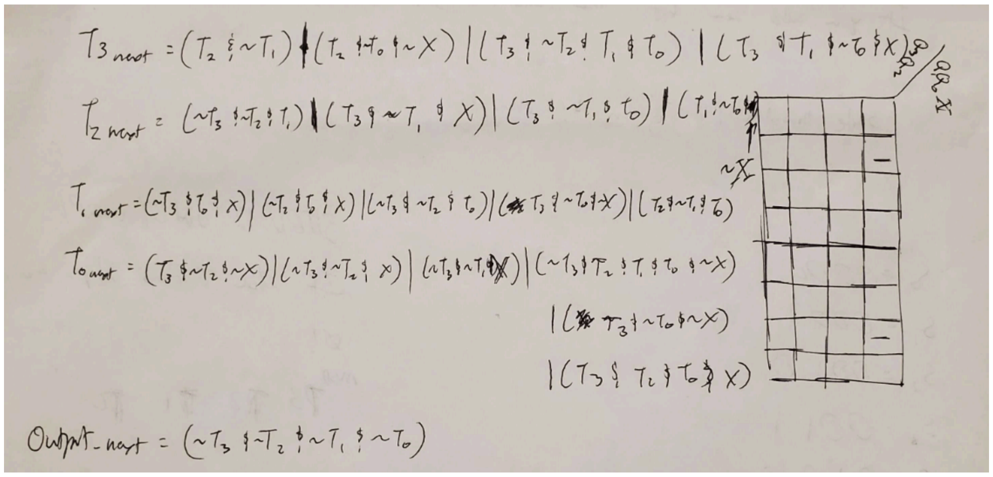
Provide a state transition table showing the output and next state for each combination of current state and inputs. Add rows as needed. For Current and Next States use the same symbolic names as you did in the State Transition Diagram.

	Current State	X	RST	Next State	Z
0	0000	0	0	0000	1
1	0001	0	0	0010	0
2	0010	0	0	0100	0
3	0011	0	0	0110	0
4	0100	0	0	1000	0
5	0101	0	0	1010	0
6	0110	0	0	1100	0
7	0111	0	0	0001	0
8	1000	0	0	0011	0
9	1001	0	0	0101	0
10	1010	0	0	0111	0
11	1011	0	0	1001	0
12	1100	0	0	1011	0
13	1101	0	0	XXXX	X
14	1110	0	0	XXXX	X
15	1111	0	1	0000	0
16	0000	1	0	0001	1
17	0001	1	0	0011	0
18	0010	1	0	0101	0
19	0011	1	0	0111	0
20	0100	1	0	1001	0
21	0101	1	0	1011	0

22	0110	1	0	0000	0
23	0111	1	0	0010	0
24	1000	1	0	0100	0
25	1001	1	0	0110	0
26	1010	1	0	1000	0
27	1011	1	0	1010	0
28	1100	1	0	1100	0
29	1101	1	0	XXXX	X
30	1110	1	0	XXXX	X
31	1111	1	1	0001	0

State Encoding

Briefly describe how you encoded the states into bit vectors and why you picked this encoding.



T3 next = (T2 & ~T1) | (T2 & ~T0 & ~X) | (T3 & ~T2 & T1 & T0) | (T3 & T1 & ~T0 & X)

T2 next = (~T3 & ~T2 & T1) | (T3 & ~T1 & X) | (T3 & ~T1 & T0) | (T1 & ~T0 & ~X)

T1 next = (~T3 & T0 & X) | (~T2 & T0 & X) | (~T3 & ~T2 & T0) | (T3 & ~T0 & ~X) | (T2 & ~T1 & T0)

T0 next = (T3 & ~T2 & ~X) | (~T3 & ~T2 & X) | (~T3 & ~T1 & X) | (~T3 & T2 & T1 & T0 & ~X) | (T3 & ~T0 & ~X) | (T3 & T2 & T0 & X)

Output_next = (~T3 & ~T2 & ~T1 & ~T0)

Q3Q2
Q1Q0X
~X

What I did was that I had a subcircuit for each of the T flip flops that stored a state value. I then plugged each 32-row truth table for T3, T2, T1, T0, and output Z into Logic Friday's circuit generator and was able to acquire all the minterms as shown in the photo. I went on to create gates for each of the T flip flop values along with the output Z. I chose to encode this way because 4 T flip flops would've been perfect to store the 13 remainder states + 1 reset state as log2(14) ceiling is 4. This way also minimized the cost. However, when the reset bit was activated I would OR it with the result of each of the T flip flop next-state inputs as OR-ing a reset bit that is equivalent to 1 with all the other input bits would set the next state as 1111 which brings us to the proper reset condition. This way I was able to cover for all cases.

Other Details

Provide any other details that would help grade.

One thing to be noted is that I actually misread the prompt at first and read it as executing the entire circuit with only D flip flops. As a result I had to XOR the next state of the D flip flop with the current value of the flip flop to get the T flip flop's next input (T_i = Q_i XOR D_i as seen on my state table paper). As a result that is why I XOR'ed all my results at the very end.