Forgiving Lock Part 2

EXTRA-Forgiving Lock Motivation

- **ISSUE:** I am STILL struggling with typing a correct password.
- **SOLUTION:** Let's give the user some HINTS!

EXTRA-Forgiving Lock Design Problem (Hint #1)

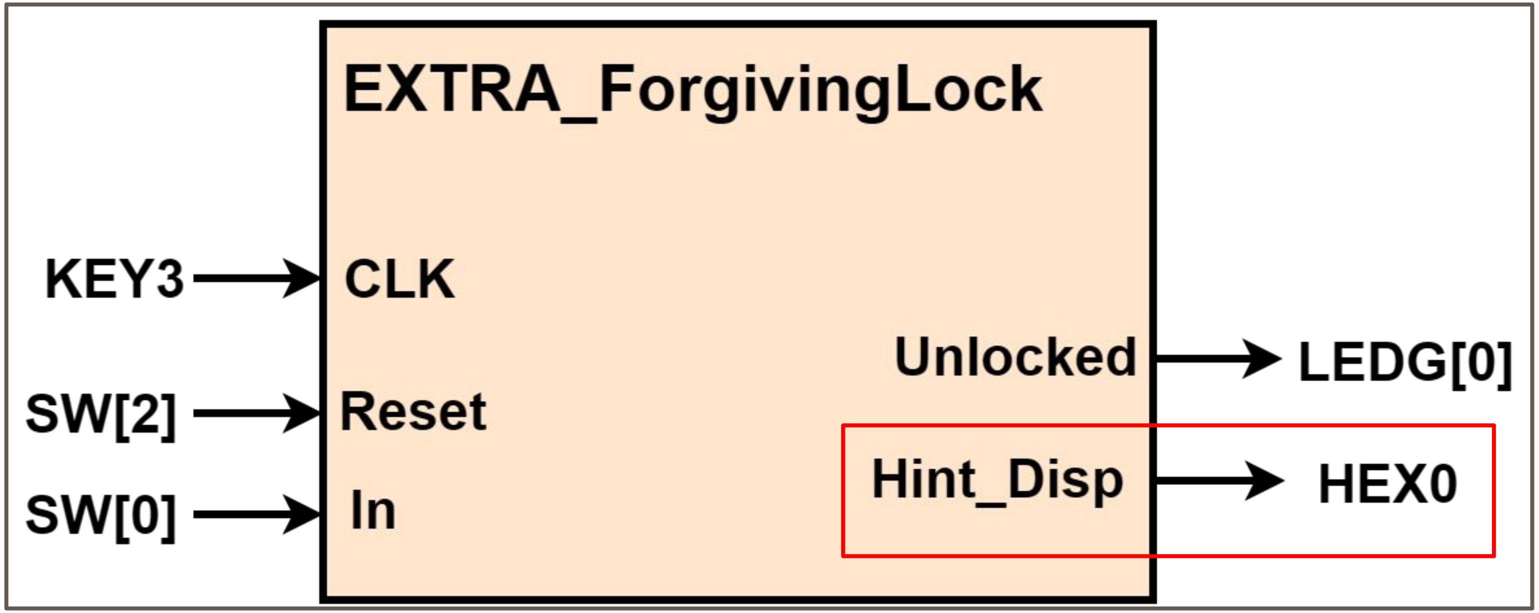
- **Hint #1 (Password Auto-Fill):** We will inform the user what the next digit should be to get closer to a correct password

- Using the same password constraint from before
  - Allow user to unlock if the current code has alternating numbers in at least 3 digits
  - Ex. 010x, 101x, x010, …

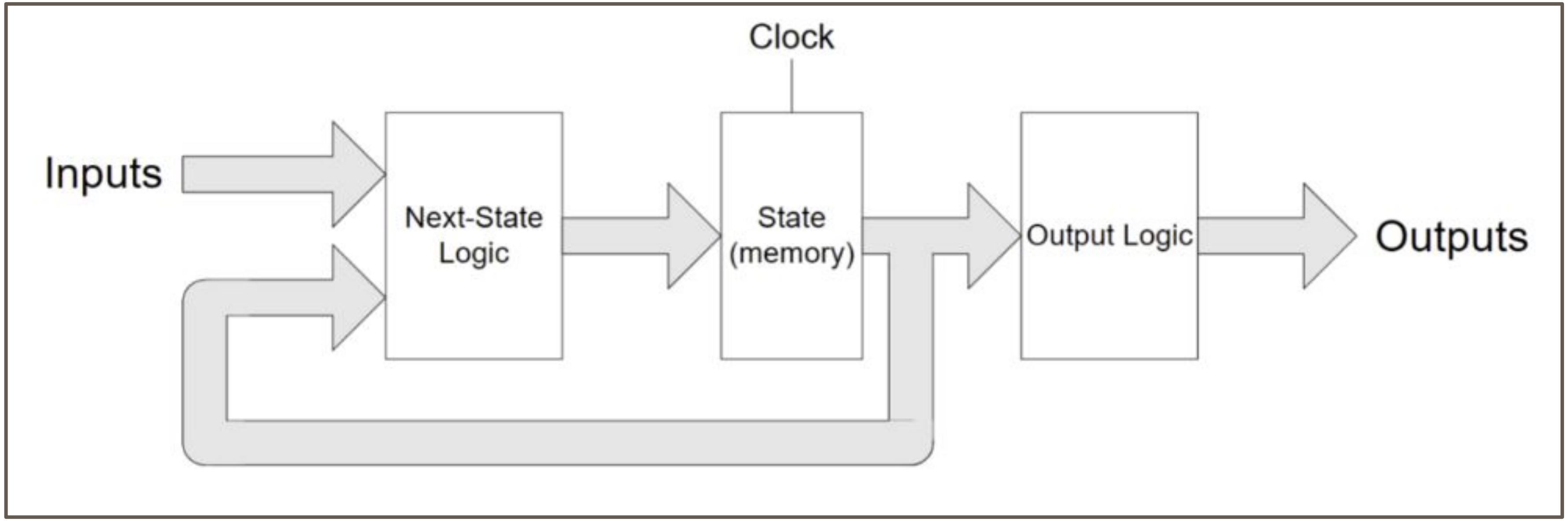
- **NEW: Will display the number the user should provide to get towards a correct password**
  - Ex. Current Stream 1110 → Display ‘1’
  - Ex. Current Stream 1001 → Display ‘0’
  - Ex. Current Stream 1010 → Display ‘1’
  - Ex. Current Stream 1111 → Display ‘0’

EXTRA-Forgiving Lock Design Diagram

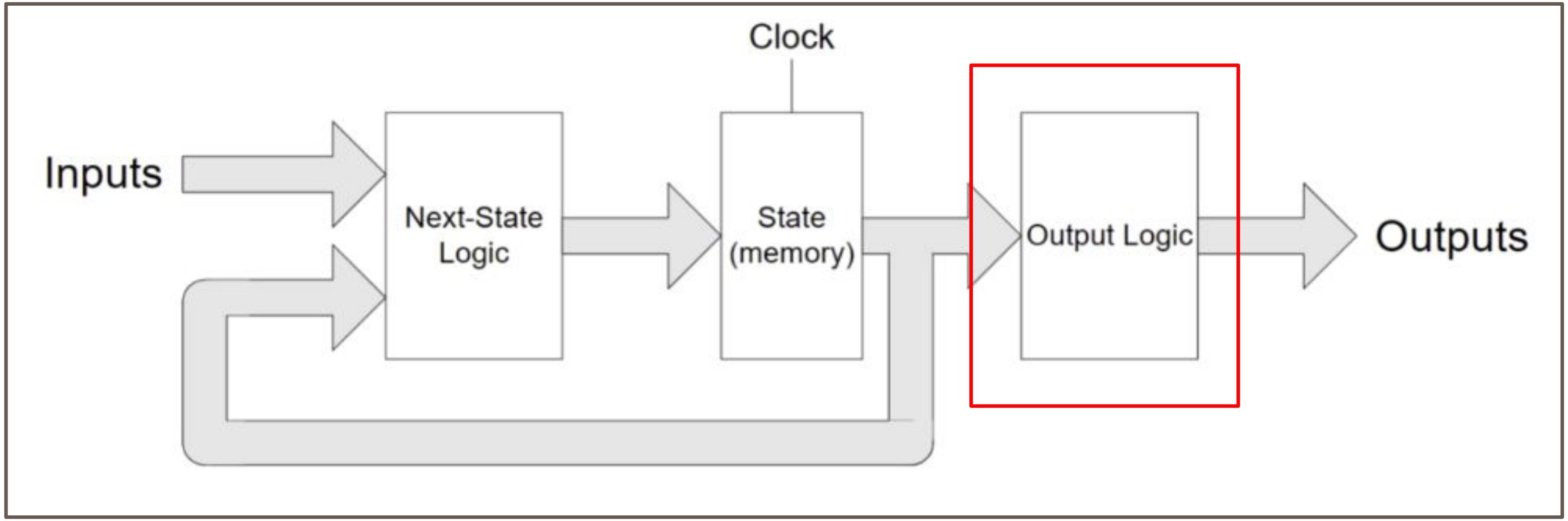
- Addition of a HEX display output



What’s Going to Change?



What's Going to Change?



EXTRA-Forgiving Lock Output Table / Equation

Code	Hint_Disp
0000	
0001	
0010	
0011	
0100	
0101	
0110	
0111	

Code	Hint_Disp
1000	
1001	
1010	
1011	
1100	
1101	
1110	
1111	

Hint_Disp = ?

EXTRA-Forgiving Lock Output Table / Equation

Code	Hint_Disp
0000	“1”
0001	
0010	
0011	
0100	
0101	
0110	
0111	

Code	Hint_Disp
1000	
1001	
1010	
1011	
1100	
1101	
1110	
1111	

Hint_Disp = ?

EXTRA-Forgiving Lock Output Table / Equation

Code	Hint_Disp
0000	“1”
0001	“0”
0010	
0011	
0100	
0101	
0110	
0111	

Code	Hint_Disp
1000	
1001	
1010	
1011	
1100	
1101	
1110	
1111	

Hint_Disp = ?

EXTRA-Forgiving Lock Output Table / Equation

Code	Hint_Disp
0000	“1”
0001	“0”
0010	“1”
0011	
0100	
0101	
0110	
0111	

Code	Hint_Disp
1000	
1001	
1010	
1011	
1100	
1101	
1110	
1111	

Hint_Disp = ?

EXTRA-Forgiving Lock Output Table / Equation

Code	Hint_Disp
0000	“1”
0001	“0”
0010	“1”
0011	“0”
0100	“1”
0101	“0”
0110	“1”
0111	“0”

Code	Hint_Disp
1000	
1001	
1010	
1011	
1100	
1101	
1110	
1111	

Hint_Disp = ?

EXTRA-Forgiving Lock Output Table / Equation

Code	Hint_Disp
0000	“1”
0001	“0”
0010	“1”
0011	“0”
0100	“1”
0101	“0”
0110	“1”
0111	“0”

Code	Hint_Disp
1000	“1”
1001	“0”
1010	“1”
1011	“0”
1100	“1”
1101	“0”
1110	“1”
1111	“0”

Hint_Disp = ?

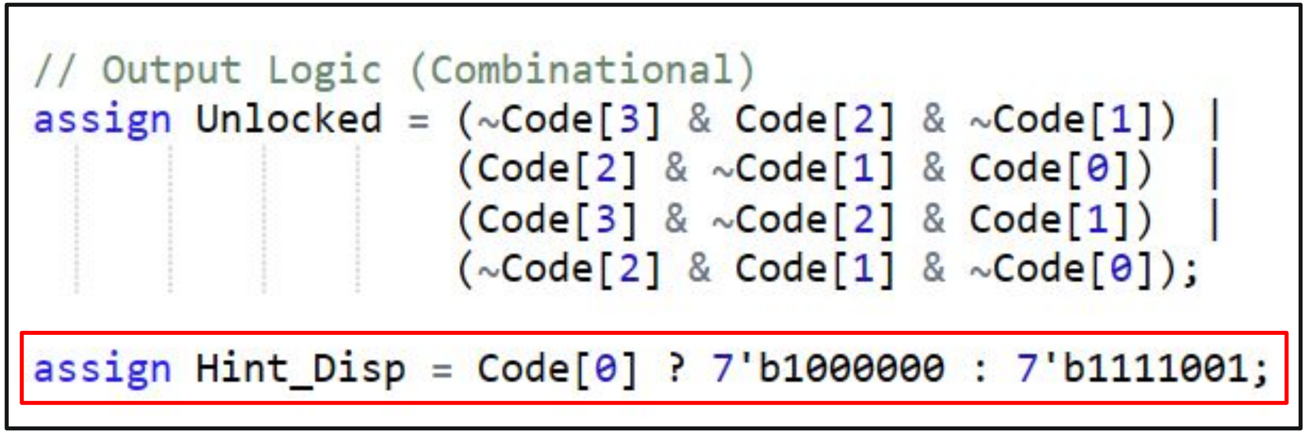
EXTRA-Forgiving Lock Output Table / Equation

Code	Hint_Disp
0000	“1”
0001	“0”
0010	“1”
0011	“0”
0100	“1”
0101	“0”
0110	“1”
0111	“0”

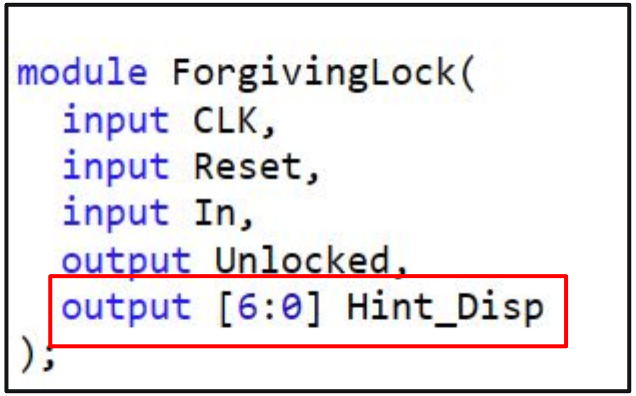
Code	Hint_Disp
1000	“1”
1001	“0”
1010	“1”
1011	“0”
1100	“1”
1101	“0”
1110	“1”
1111	“0”

Hint_Disp =
Code[0] ? “0” : “1”

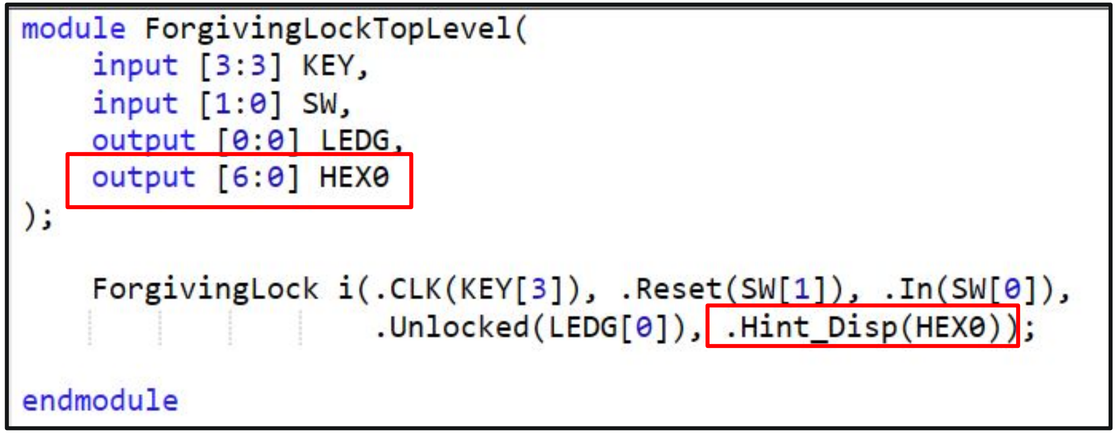
Small Changes to Code



```
// Output Logic (Combinational)
assign Unlocked = (~Code[3] & Code[2] & ~Code[1]) |
                  (Code[2] & ~Code[1] & Code[0])  |
                  (Code[3] & ~Code[2] & Code[1])  |
                  (~Code[2] & Code[1] & ~Code[0]);

assign Hint_Disp = Code[0] ? 7'b1000000 : 7'b1111001;
```



```
module ForgivingLock(
  input CLK,
  input Reset,
  input In,
  output Unlocked,
  output [6:0] Hint_Disp
);
```



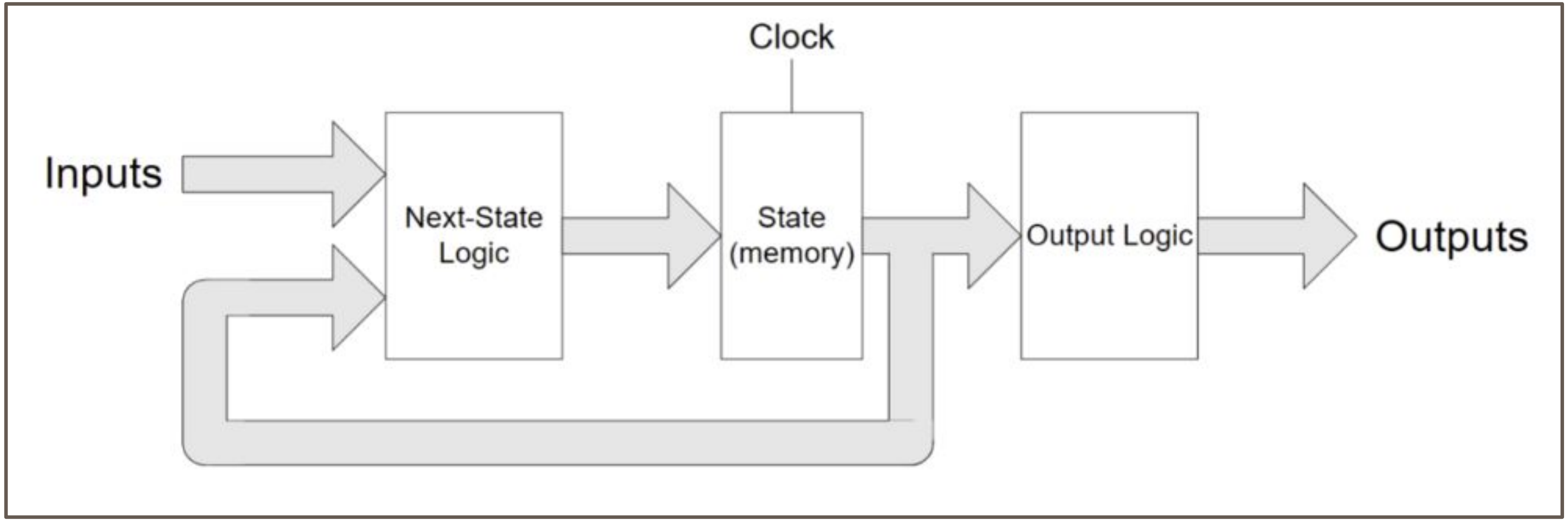
```
module ForgivingLockTopLevel(
    input [3:3] KEY,
    input [1:0] SW,
    output [0:0] LEDG,
    output [6:0] HEX0
);

    ForgivingLock i(.CLK(KEY[3]), .Reset(SW[1]), .In(SW[0]),
                    .Unlocked(LEDG[0]), .Hint_Disp(HEX0));

endmodule
```

EXTRA-Forgiving Lock Design Problem (Hint #2)

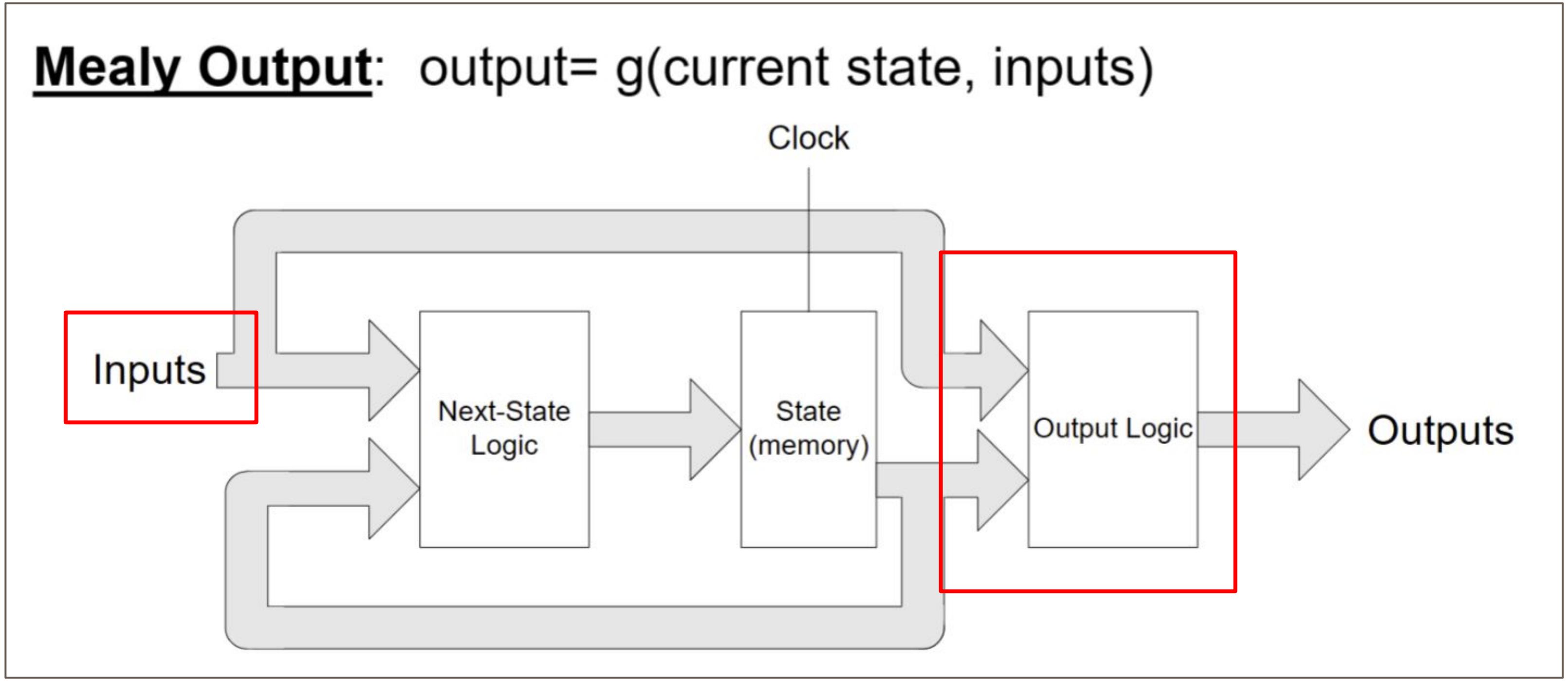
- **Hint #2 (Almost There!):** We want to let the user know if the input they are ABOUT to select will result in a correct password state
  - The user has not yet actually chosen the given input at this time

- **Will turn on an indicator to alert the user that choosing this value will result in a correct password state**
  - Ex. (Current Stream = 1110) & (In = 0) → **Indicator OFF (Will not be correct password)**
  - Ex. (Current Stream = 1001) & (In = 0) → **Indicator ON (Will BECOME correct password)**
  - Ex. (Current Stream = 1010) & (In = 0) → **Indicator ON (Will STAY correct password)**
  - Ex. (Current Stream = 0100) & (In = 1) → **Indicator OFF (Will not be correct password)**

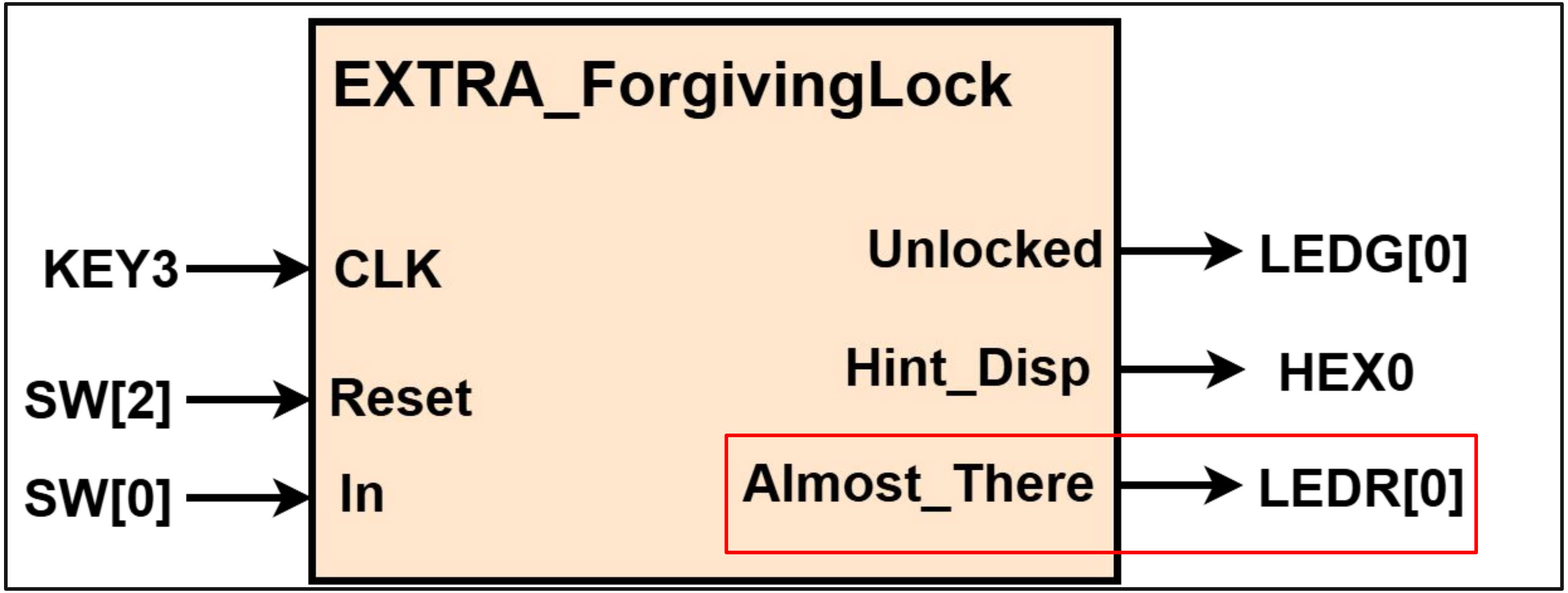
What’s Wrong with This Model?



The Indicator is a Mealy Output!!



EXTRA-Forgiving Lock Design Diagram (Hint #2)



Now it's your turn to update the code!