Lab 8 Submission

**Multi-Output 4-bit Up/Down Counter**

CPE 133 - 03

Michael Hegglin

Jonathan Skelly

**Executive Summary:**

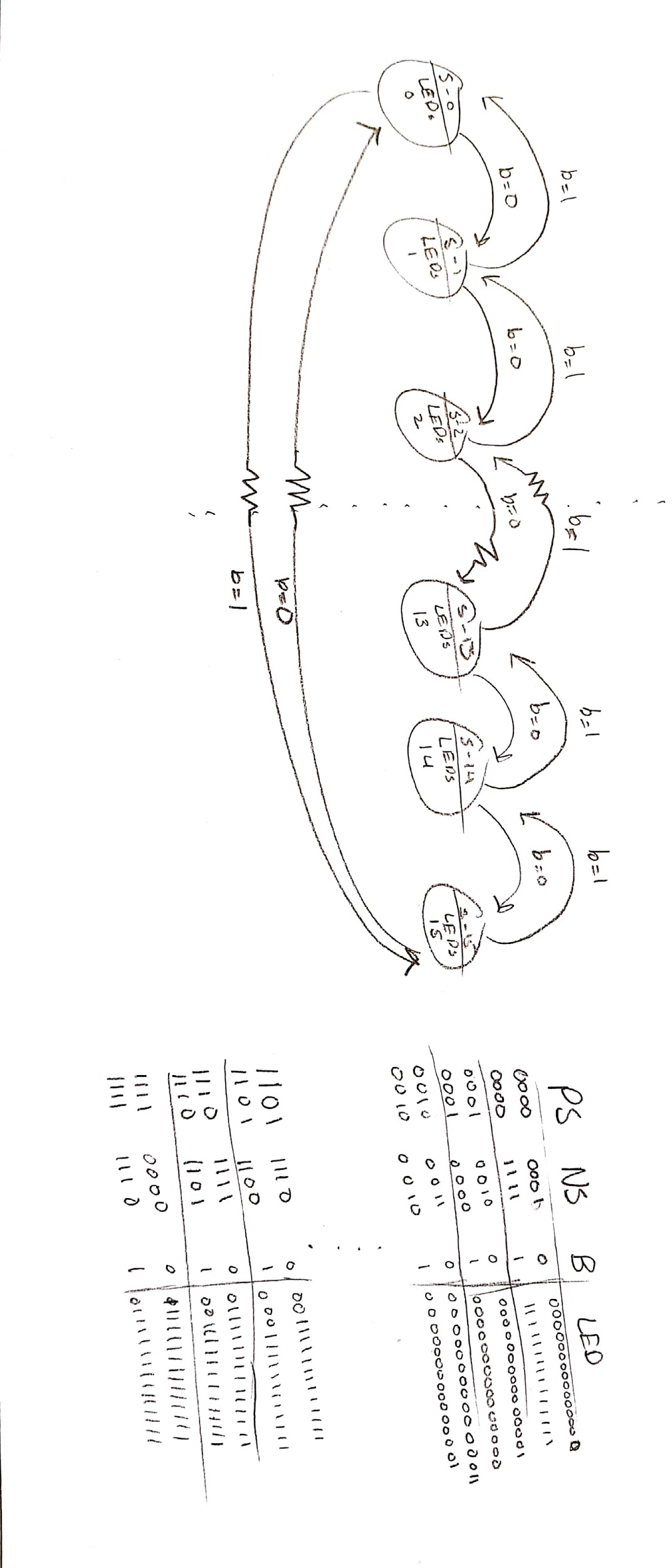
We designed a 4-bit up/down counter with an unsigned binary number. The counter used a clock, a register, and two logic modules. The register held the data, while the logic modules helped display the LED and segment outputs, and decide which number was happening next.



BBD of our Multi-Output 4-bit Up/Down Counter



Next-level down diagram of our Multi-Output 4-bit Up/Down Counter



State diagram and table of our Multi-Output 4-bit Up/Down Counter

**Questions**:

1. Briefly describe the difference between a flip-flop and a latch.

* Latches are asynchronous while flip-flops are synchronous.

2. Sequential circuits are referred to as having “state”. In your own words, briefly describe what exactly that means.

* Having a state means the current value currently being held by the state machine.

3. Briefly describe why state diagrams generally do not include any notion of a clock signal.

* With state diagrams it is implied that the clock signal governs each transition.

4. This experiment only asked you to draw a state diagram using six states. Briefly describe how many states the FSM in this experiment actually contains.

* The FSM in this experiment actually contains 16 states, one for each value.

5. Briefly but completely describe the three basic modules of a FSM. Make sure your description includes which of the two classes of digital circuits the modules represent.

* There are mealy (Their output is decided upon an outside variable), and moore digital circuits(simply output the data inputted into it with whatever decoding the user desires). These modules are flip-flops(synchronous output), latches(asynchronous output), and counters(stores the number of times something has happened).

6. The state registers in a FSM are considered a synchronous circuit. Briefly describe what this means in context of the FSM.

* It means it’s being timed by a clock, and that state variables are stored, then changed simultaneously.

7. This experiment used a clock divider to slow down the dev board’s clock to something around 2Hz. Show how exactly you slowed down the clock, including any associated calculations.

* We slowed down the clock by dividing by a certain amount of hertz. The clock displays at around 100MHz. By using n = 26 in the clock divider module. We are dividing that 100MHz by ~50MHz. This means the display slows down to displaying at a rate of around 2Hz. This enables us to see the numbers.
* 2^26 = 67,108,864

8. Sequential circuits are known to contain memory, but it is not clear from the word “sequential” where the memory comes from. For this problem, briefly describe the relation between the word “sequential” and memory.

* The memory in this lab was in the register module. While the first logic module decided what the next state would be, the register module keeps track of the past state, and provides an output of that state. It does this through the clock signal which it is synchronized with. It stores the outputs in the sequence they are created.

9. The FSM you designed in this experiment has status inputs and control outputs. Make a table showing the FSM’s inputs and output; briefly describe the function of each input and output in terms of status and control signals.

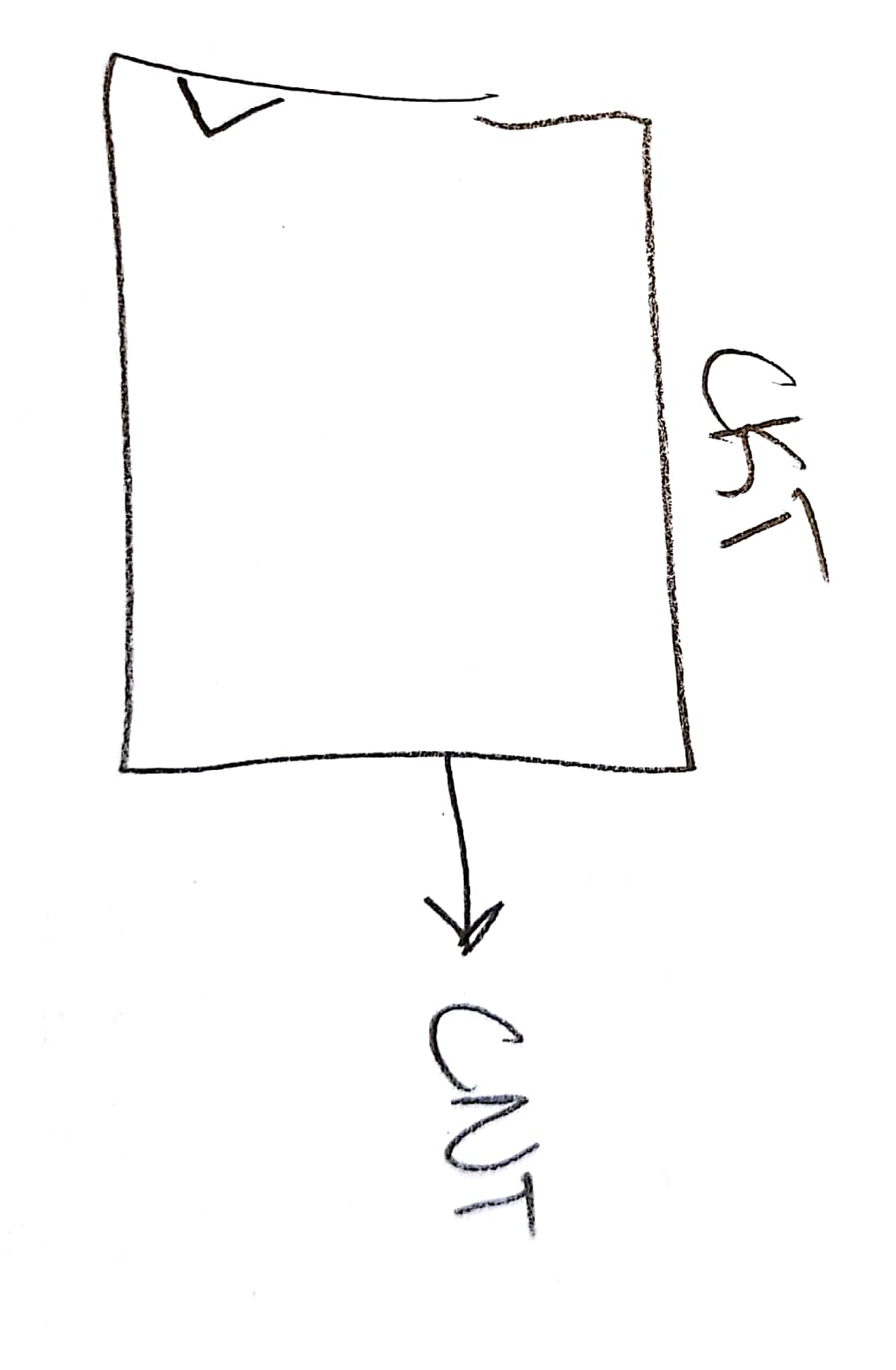
* To summarize, each input is the last output which then decides the next output signal. The button being pressed flips the inputs and the outputs so it starts counting down. These outputs can be seen in the binary form in our source code.

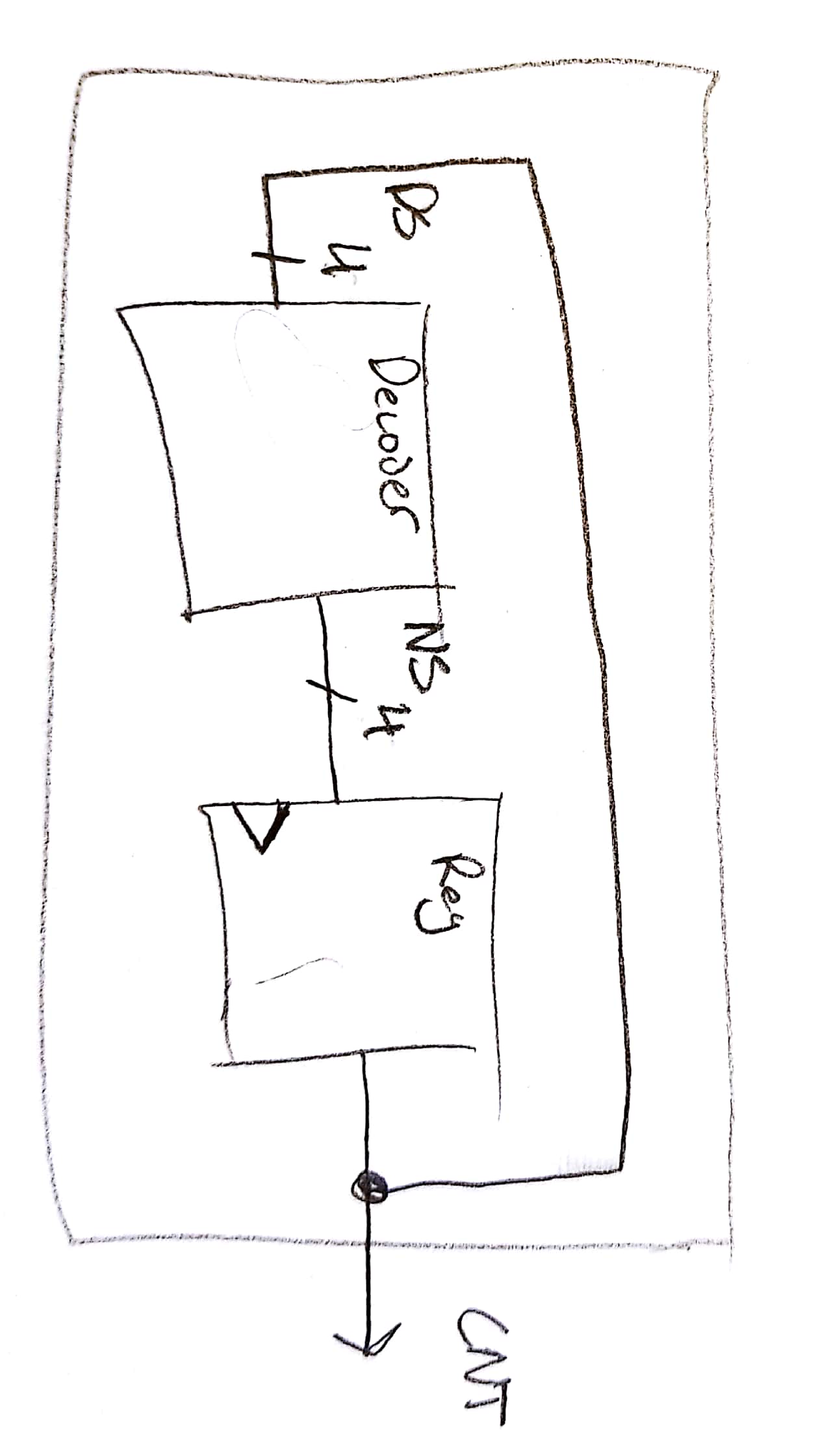
|  |  |
| --- | --- |
| Inputs | Outputs |
| 0 | 1 |
| 1 | 2 |
| 2 | 3 |
| 3 | 4 |
| 4 | 5 |
| 5 | 6 |
| 6 | 7 |
| 7 | 8 |
| 8 | 9 |
| 9 | 10 |
| 10 | 11 |
| 11 | 12 |
| 12 | 13 |
| 13 | 14 |
| 14 | 15 |
| 15 | 0 |

**Design Problems:**

1. Design a 4-bit synchronous down counter. This output of this counter should reflect a binary output that represents RC numbers, which effectively means the counter counts as listed below. This counter should follow the standard FSM model that includes three basic modules, though you may not need all three modules for this problem. State how the circuit is controlled.

The circuit is controlled internally, by the past output.

BBD of our 4-bit RC counter.

Next-Level down of our 4-bit RC counter. Outputs the state as count, as the state and CNT are represented by the same binary numbers. Features no control.

**MainCode:**

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company: DogsWithJobs

// Engineer: Hegglin/Skelly

//

// Create Date: 11/2/2018 12:00 PM

// Design Name: Counter

// Module Name: Lab\_8\_source

// Project Name: Multi-Output 4-bit Up/Down Counter

// Target Devices: Digilent Board

// Tool Versions: Verilog

// Description: Counts up via LEDs in stoneage binary while displaying // decimal equivalent on 7-segment display.

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module MainLogic(clk, button, led, seg, an);

input clk, button;

wire[3:0] run\_around, run\_out;

wire nclk;

output [14:0] led;

output [7:0] seg;

output [3:0] an;

clk\_divder\_nbit #(26) MY\_DIV (

.clockin (clk),

.clockout (nclk)

);

initialDecoder D1(

.run\_out(run\_out),

.button(button),

.run\_around(run\_around)

);

reg\_nb #(4) MY\_REG (

.data\_in (run\_out),

.ld (1),

.clk (nclk),

.clr (0),

.data\_out (run\_around)

);

mealyMoore myMM (

.reg\_output (run\_around),

.led\_nums (led)

);

univ\_sseg my\_univ\_sseg (

.cnt1 (run\_around),

.cnt2 (0),

.valid (1),

.dp\_en (0),

.dp\_sel (0),

.mod\_sel (0),

.sign (0),

.clk (clk),

.ssegs (seg),

.disp\_en (an)

);

endmodule

//output decoder

module mealyMoore(reg\_output, led\_nums);

input [3:0] reg\_output;

output reg [14:0] led\_nums;

always @ (reg\_output)

begin

case (reg\_output)

0: led\_nums = 15'b000000000000000;

1: led\_nums = 15'b000000000000001;

2: led\_nums = 15'b000000000000011;

3: led\_nums = 15'b000000000000111;

4: led\_nums = 15'b000000000001111;

5: led\_nums = 15'b000000000011111;

6: led\_nums = 15'b000000000111111;

7: led\_nums = 15'b000000001111111;

8: led\_nums = 15'b000000011111111;

9: led\_nums = 15'b000000111111111;

10: led\_nums = 15'b000001111111111;

11: led\_nums = 15'b000011111111111;

12: led\_nums = 15'b000111111111111;

13: led\_nums = 15'b001111111111111;

14: led\_nums = 15'b011111111111111;

15: led\_nums = 15'b111111111111111;

endcase

end

endmodule

//initial decoder

module initialDecoder(run\_out, button, run\_around);

input button;

input [3:0] run\_around;

output reg [3:0] run\_out;

always @ (run\_around, button)

begin

if(button == 0)

begin

case (run\_around)

0: run\_out = 4'b0001;

1: run\_out = 4'b0010;

2: run\_out = 4'b0011;

3: run\_out = 4'b0100;

4: run\_out = 4'b0101;

5: run\_out = 4'b0110;

6: run\_out = 4'b0111;

7: run\_out = 4'b1000;

8: run\_out = 4'b1001;

9: run\_out = 4'b1010;

10: run\_out = 4'b1011;

11: run\_out = 4'b1100;

12: run\_out = 4'b1101;

13: run\_out = 4'b1110;

14: run\_out = 4'b1111;

15: run\_out = 4'b0000;

default: run\_out = 4'b0000;

endcase

end

else

begin

case (run\_around)

2: run\_out = 4'b0001;

3: run\_out = 4'b0010;

4: run\_out = 4'b0011;

5: run\_out = 4'b0100;

6: run\_out = 4'b0101;

7: run\_out = 4'b0110;

8: run\_out = 4'b0111;

9: run\_out = 4'b1000;

10: run\_out = 4'b1001;

11: run\_out = 4'b1010;

12: run\_out = 4'b1011;

13: run\_out = 4'b1100;

14: run\_out = 4'b1101;

15: run\_out = 4'b1110;

0: run\_out = 4'b1111;

1: run\_out = 4'b0000;

default: run\_out = 4'b0000;

endcase

end

end

endmodule