CE 100 Lab Report 6

Turkey Traffic

Armando Silva

Tutor: Dawn Hustig-Schultz

Lab Section: Tuesday, Thursday 2:00pm – 4:00pm

Date: February 26th 2016

Description:

The purpose of this lab which was to get more familiarized with state machines, use of two's compliment, and use of Verilog to demonstrate the turkey's grade depending on the system clock. Through the turkey traffic game, the logical design of each symbol was able to be verified and tested in series of simulations, complied files, and downloading on the Basys board for demonstration. In all, this lab consisted of the following parts:

Part A- The State Machine

In order to construct the state machine in Figure 1, I will analyze and explain my concept behind the automaton. In the first state "Idle", this state is my start state; essentially as long as Push Button 0 and Push Button 2 are not pressed or Push Button 0 and Push Button 2 are pressed at the same time, I stay in this state and display the score, also known as "Gradestop". Second, I move to another state depending on which Push Button was pressed, this means that the turkey crossed one of the infrared sensors. If the turkey crossed or pressed Button 0 then I go to state check and if pressed Button 2 then I go to state checkminus. Essentially, if we passed the first sensor and then cross the next sensor we will increment or decrement depending on which way the turkey is going. Therefore, after check, if Push Button 0 and Push Button 2 are both pressed we follow through to the next state. Then we can observe that the other states are the final crossing; as well we need to cover all cases in which the turkey may fly away or decide to go back; therefore, the rest of the transitions are described in the figure. After creating my automaton for my state machine, I then build the state machine using Verilog.

Part B- Keeping track

I began with using an up-down counter provided by us by Xilinx as CB8CLED. This is only incremented or decremented on the clock enabled, which then results in an 8-bit number. As for when incremented or decremented is high or low, we can check the state machine from the previous part. You can observe the counter is incremented or decremented when the turkey does a full cross. Then here we do some arithmetic to determine whether the 8-bit number is negative or positive, depending on which side the turkey crossed by using the two-compliment by checking the last bit.

Furthermore, I will go more in depth with the two compliment symbol. Here, I have taken in the 8-bit bus from our counter and I check the last bit or q(7). This last bit is my selector to determine whether the number is positive or negative. If the last bit is 0, then that means I can output that same 8-bit bus; however, if that number is 1; then I do the two compliment of that 8-bit bus.

After determining if I needed the two compliment, I send this 8 –bit bus out to the selector that was used in Lab 5 and the ring counter then determines when to display the bits as it passes through the hex7seg.

Part C – One last detail

To monitor the health of the turkey, I determine the amount of time that it takes for a turkey to cross the sensor, beginning with the first time either sensor is blocked until both sensors are again unblocked is recorded. The way I went about this was to use a built in symbol in Xilinx called CB8RE, which is a 8 bit counter with a reset. I count up when the turkey has crossed the first sensor, this is determined in my state machine, which I called ~grade stop. When grade stop is finally high, that means that the turkey has finally finished crossing so we can stop counting and don't reset until the next passing of the sensor. What do we do with this bits? Well based on the CE100 UCSC page, we assign the grade of the turkey passing as follows: less than 1 second (A), at least 1 second but less than 2 seconds (B), at least 2 seconds but less than 4 seconds (C), at least 4 seconds but less than 6 seconds (D), and finally at least 6 seconds (F). Therefore, here I decided to do assign values based on the similar inputs of the grade, given that we have 8-bits. I then wrote this in Verilog and outputted each letter grade, with the criteria that only one output is guaranteed to be high at a time.

Next, I was to transform this one output letter grade into a 4-bit to send to the selector in order to display on the anode. Here, I used logic again using hot encoding and used sop of products to determine when the 4-bits are high or low. Given the four-bits, I then send them as the last 4 bits to my selector to send out to the seven segment display to display on the Basyas board. This selector and seven segment display are reused from previous labs; therefore, I won't go their details again.

Finally, we have three of the four outputs to display; however, we need to determine when to display the minus sign. This is ultimately made from my negsign, which takes in a 7-bit bus, from the seven segment display to choose which segments to light up or turn off when anode 2 is displayed. And in the end put all the pieces together along with a clock that was provided to us in order to have a functioning game working.

Part D- The End

Lastly we use sensors that are from the JB-1 socket of the Basys2 board and socket JB-4 to have actual sensors for the game. Here we have the sensors connected as the figure shown in CE100 lab6 page. These sensor inputs are asynchronous and in order to make them asynchronous I would use D flip flops to synchronous with the clock.

Part E- The Simulation

Testing the state machine should be after part A; however, for writing purposes I saved it til the end so that we have a clear understanding of how my schematic is implemented. With that being said we can see from Figure 12, when Push Button 0 and Push Button 2 are not pressed, the state is in 0000001, which is state 1. However, when push button 0 is pressed we enter state 0000010, which is state two as described as check. And we can simulate further to demonstrate that this is properly working.

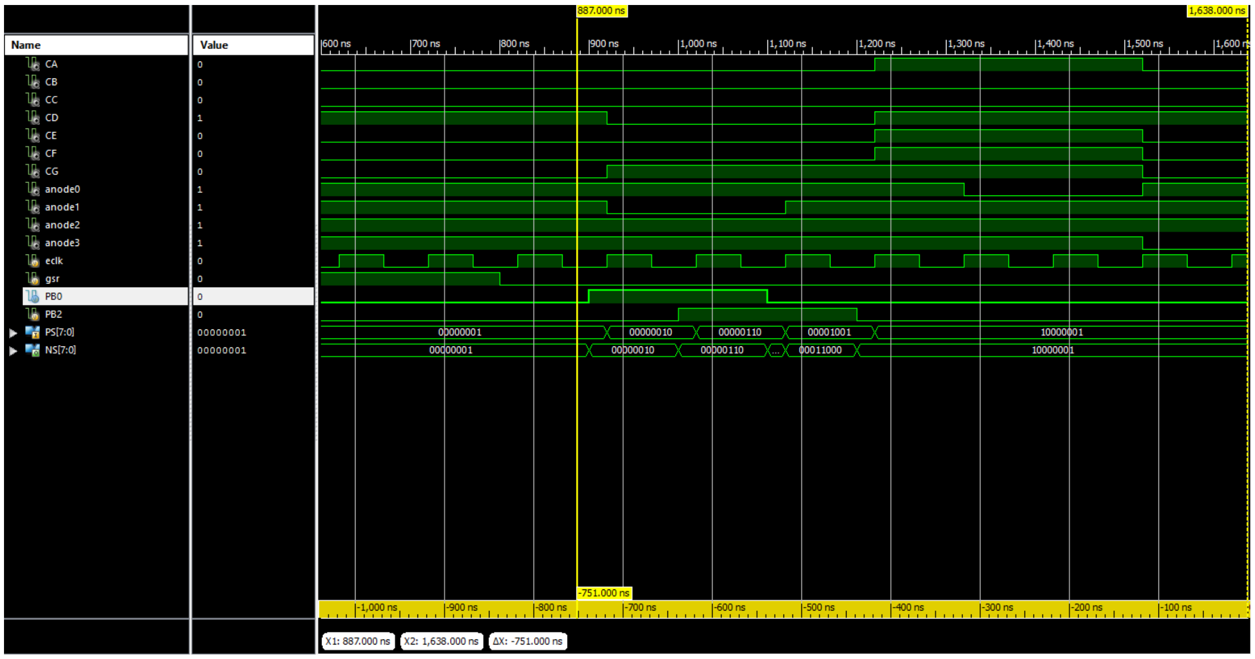


Figure 12: The state machine simulation

Conclusion

This lab proved to be tedious, which is explains why I am so burned out; especially from study for midterm, doing homework, finishing this lab and getting the bonus for lab 7. This explains why I would not recreate the automaton of figure 1-11 on a software program to use as reference. As well, instead of using the grader's recommend "professional template" from Microsoft word, I will continue to use the template given to use from CE100 sample lab reports.

Along with that, for my conclusion of the lab, I learned how to determine when to display a minus sign or a positive sign, along with the correct score. I also was able to make a successful state machine that corresponds to the turkey game and covered all possible cases that it would not crashed even if the user would do something crazy like let go of both push buttons, as if the turkey flew away. I would try to simplify my top level schematic by creating more symbols if I had dedicated more time.

Appendix:

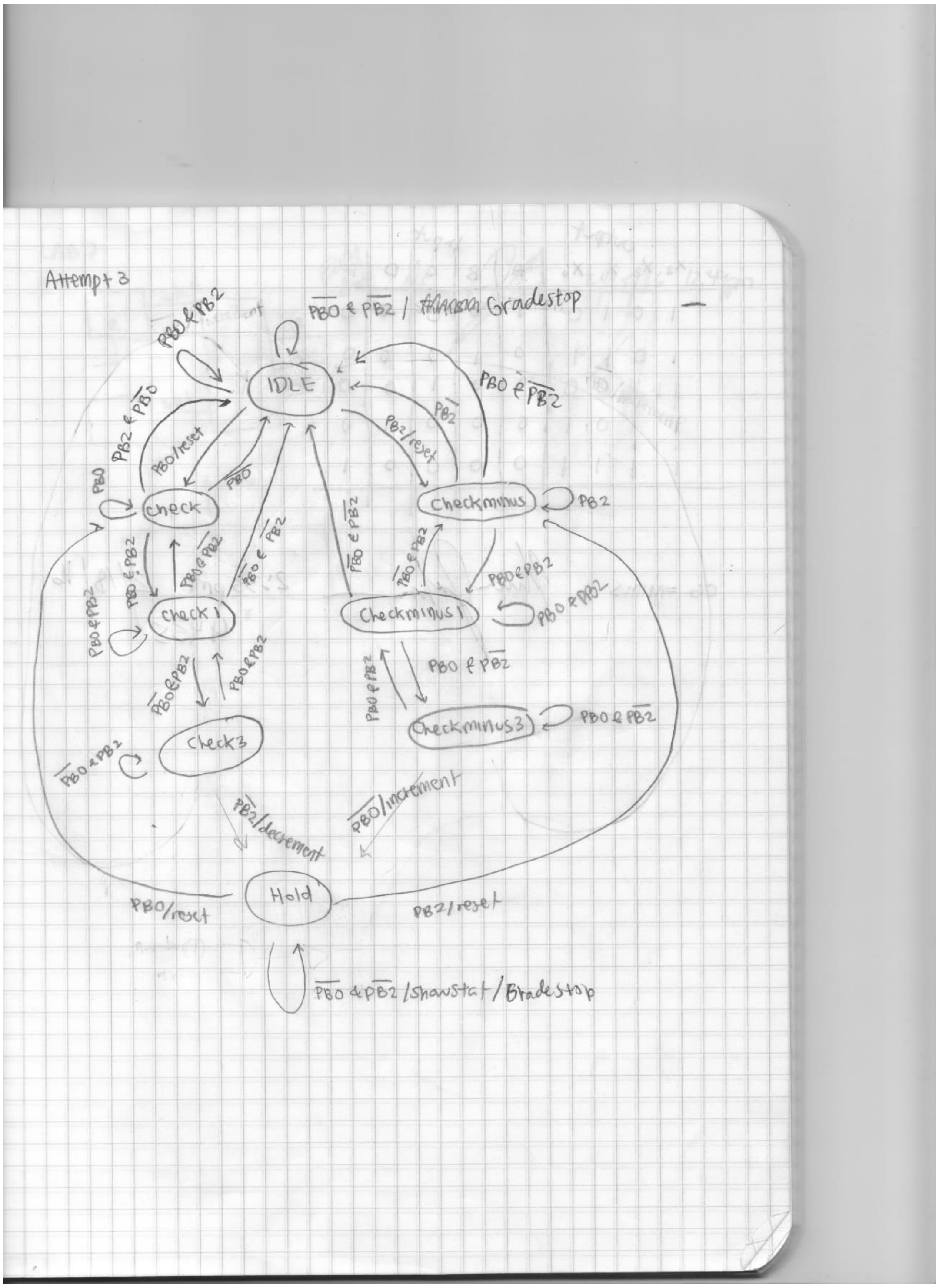
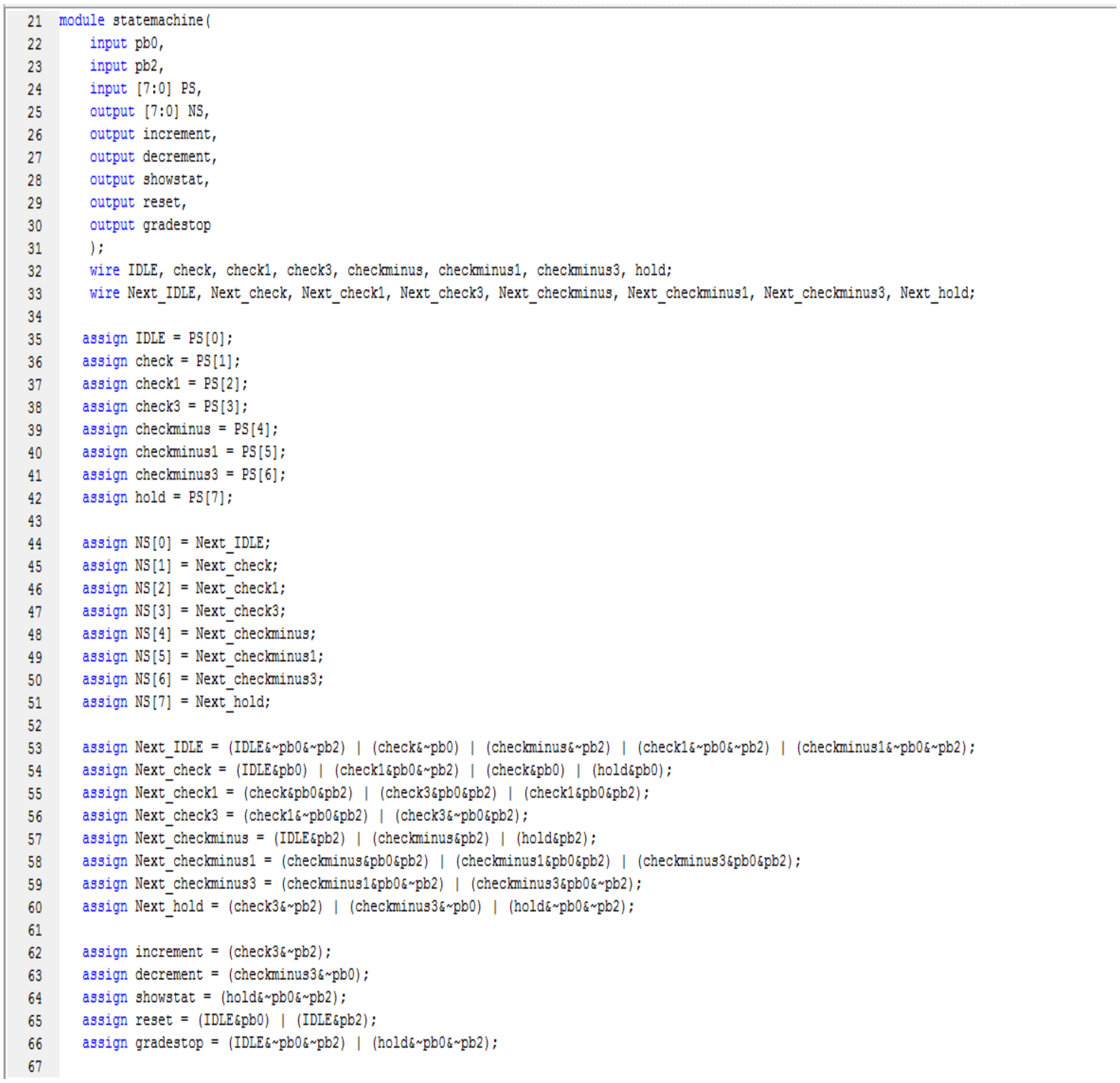


Figure 1: The 3rd and final automaton for state machine



```
module statemachine(
    input pb0,
    input pb2,
    input [7:0] PS,
    output [7:0] NS,
    output increment,
    output decrement,
    output showstat,
    output reset,
    output gradestop
    );
    wire IDLE, check, check1, check3, checkminus, checkminus1, checkminus3, hold;
    wire Next_IDLE, Next_check, Next_check1, Next_check3, Next_checkminus, Next_checkminus1, Next_checkminus3, Next_hold;

   assign IDLE = PS[0];
   assign check = PS[1];
   assign check1 = PS[2];
   assign check3 = PS[3];
   assign checkminus = PS[4];
   assign checkminus1 = PS[5];
   assign checkminus3 = PS[6];
   assign hold = PS[7];

   assign NS[0] = Next_IDLE;
   assign NS[1] = Next_check;
   assign NS[2] = Next_check1;
   assign NS[3] = Next_check3;
   assign NS[4] = Next_checkminus;
   assign NS[5] = Next_checkminus1;
   assign NS[6] = Next_checkminus3;
   assign NS[7] = Next_hold;

   assign Next_IDLE = (IDLE&~pb0&~pb2) | (check&~pb0) | (checkminus&~pb2) | (check1&~pb0&~pb2) | (checkminus1&~pb0&~pb2);
   assign Next_check = (IDLE&pb0) | (check1&pb0&~pb2) | (check&pb0) | (hold&pb0);
   assign Next_check1 = (check&pb0&pb2) | (check3&pb0&pb2) | (check1&pb0&pb2);
   assign Next_check3 = (check1&~pb0&pb2) | (check3&~pb0&pb2);
   assign Next_checkminus = (IDLE&pb2) | (checkminus&pb2) | (hold&pb2);
   assign Next_checkminus1 = (checkminus&pb0&pb2) | (checkminus1&pb0&pb2) | (checkminus3&pb0&pb2);
   assign Next_checkminus3 = (checkminus1&pb0&~pb2) | (checkminus3&pb0&~pb2);
   assign Next_hold = (check3&~pb2) | (checkminus3&~pb0) | (hold&~pb0&~pb2);

   assign increment = (check3&~pb2);
   assign decrement = (checkminus3&~pb0);
   assign showstat = (hold&~pb0&~pb2);
   assign reset = (IDLE&pb0) | (IDLE&pb2);
   assign gradestop = (IDLE&~pb0&~pb2) | (hold&~pb0&~pb2);

```

Figure 2: The Verilog that corresponds to the automaton in Figure 1

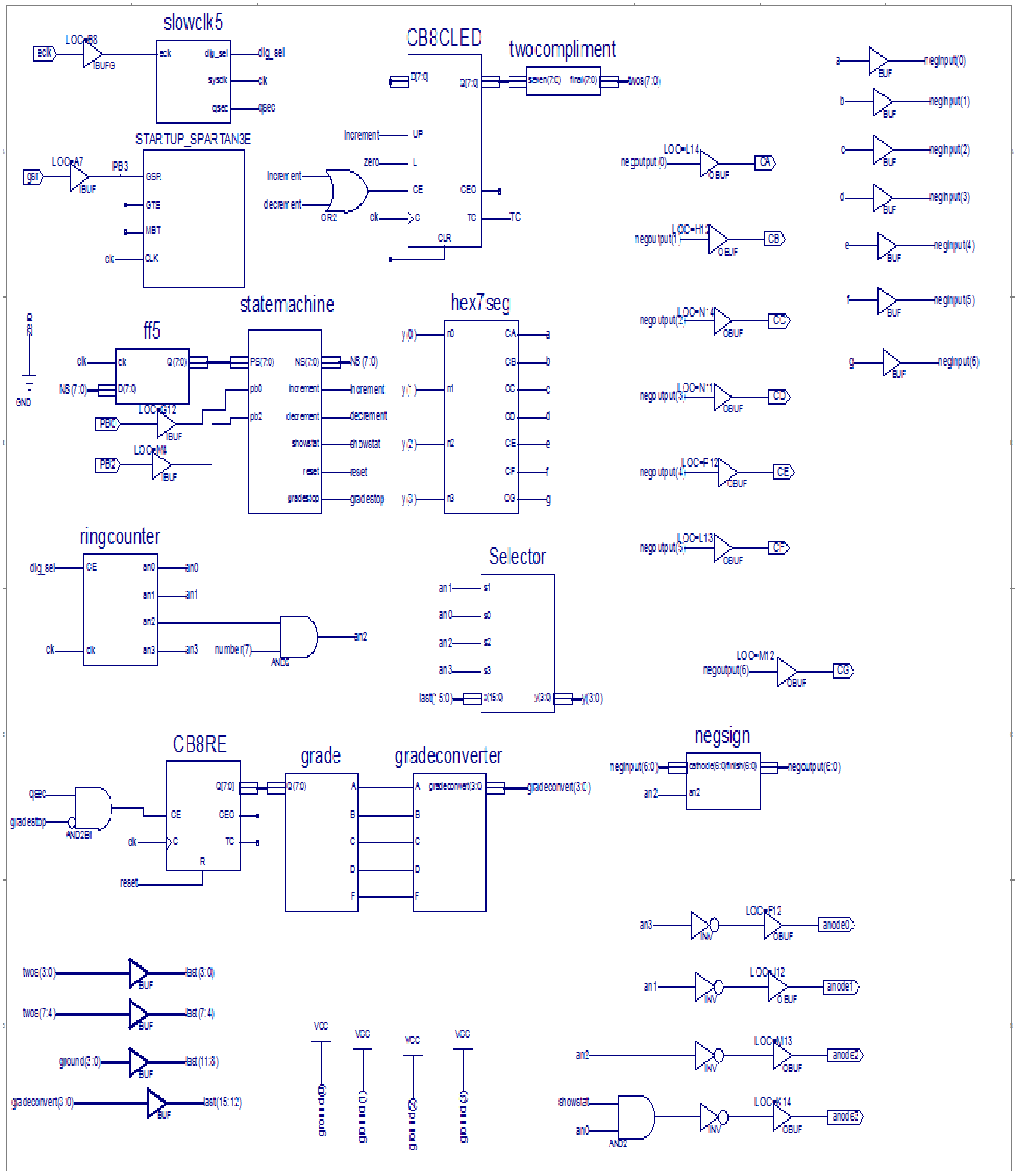


Figure 3: The top schematic of the turkey traffic

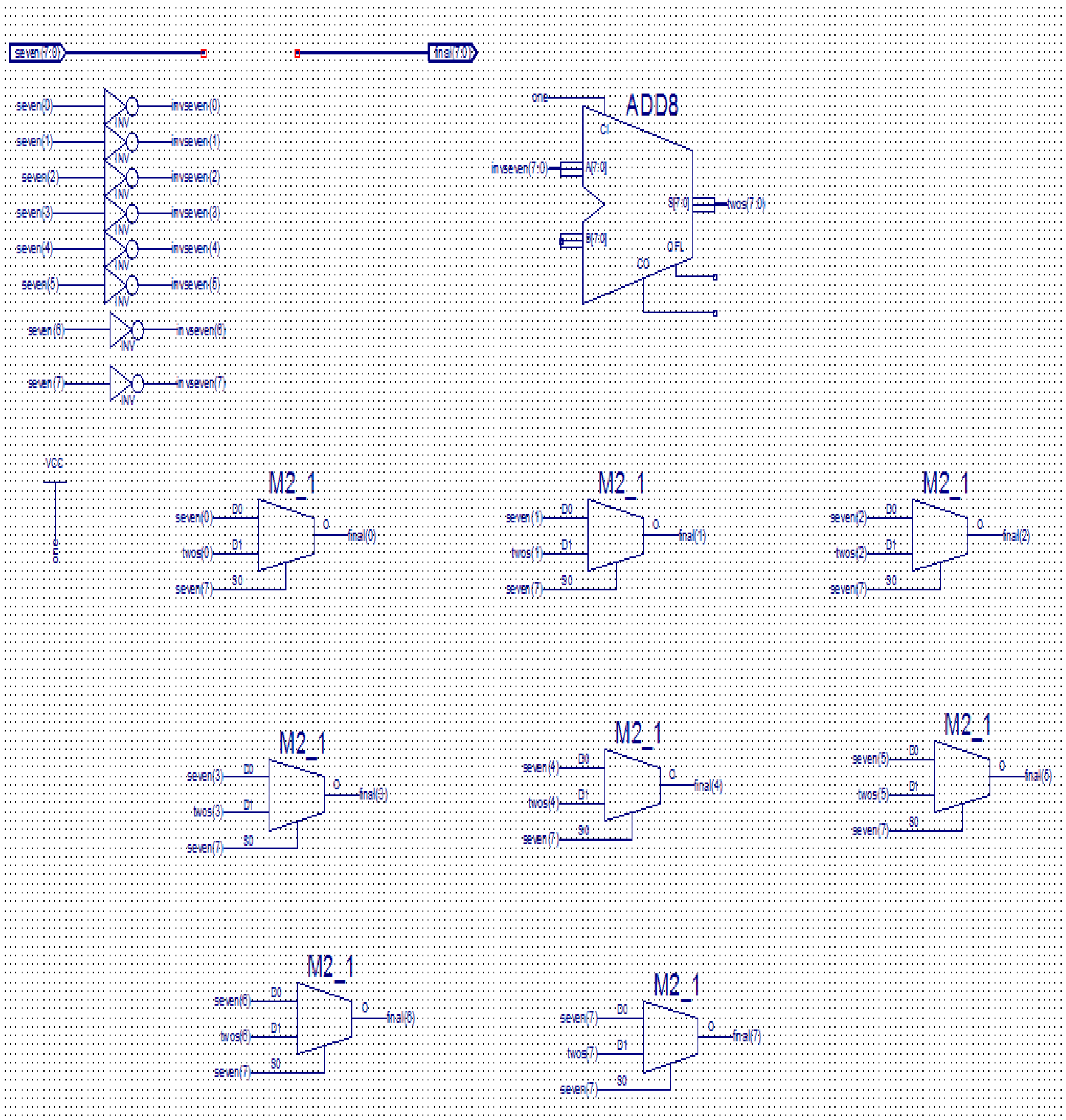
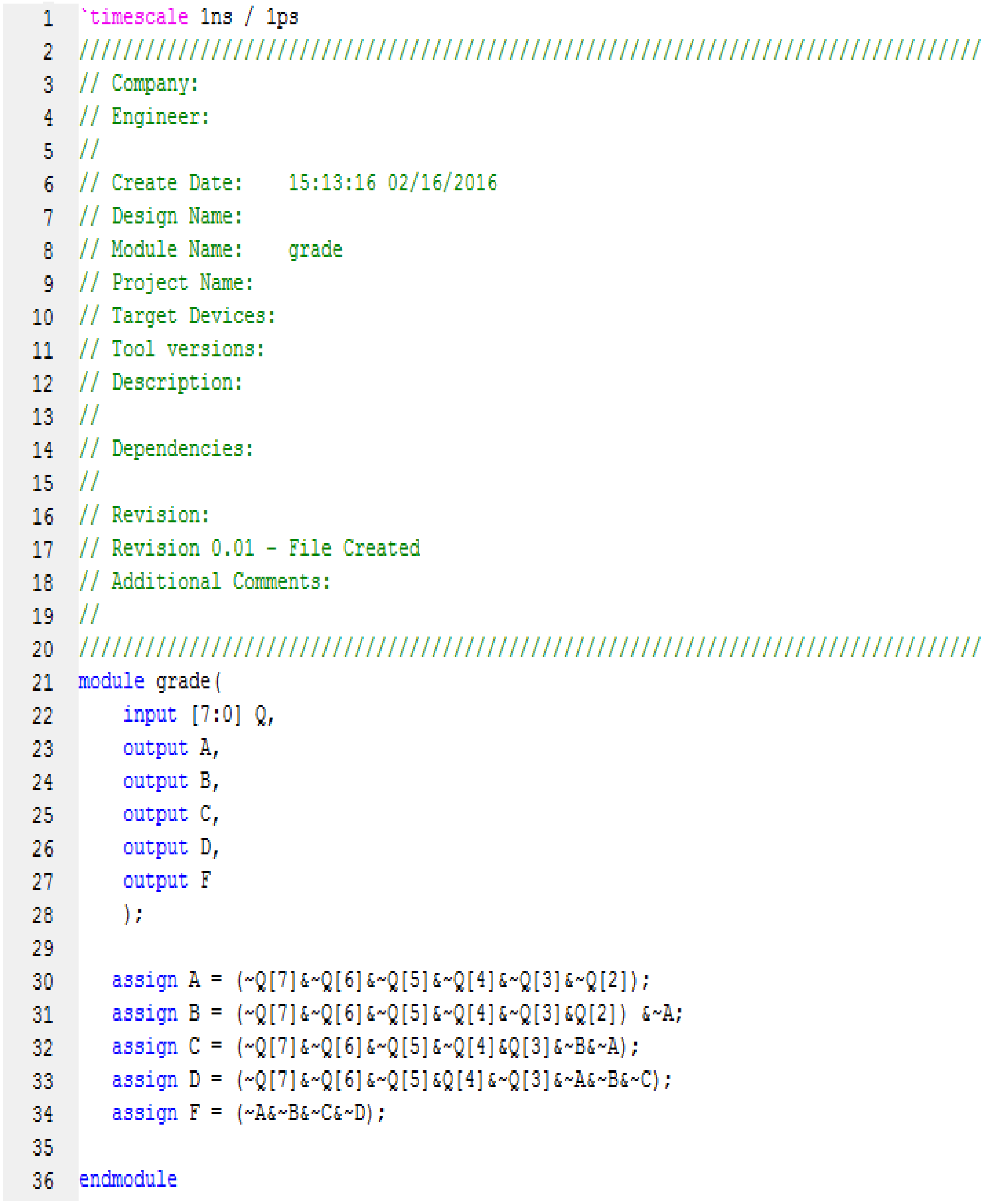


Figure 4: The two compliment symbol that takes in the 8-bit bus from CB8CLED

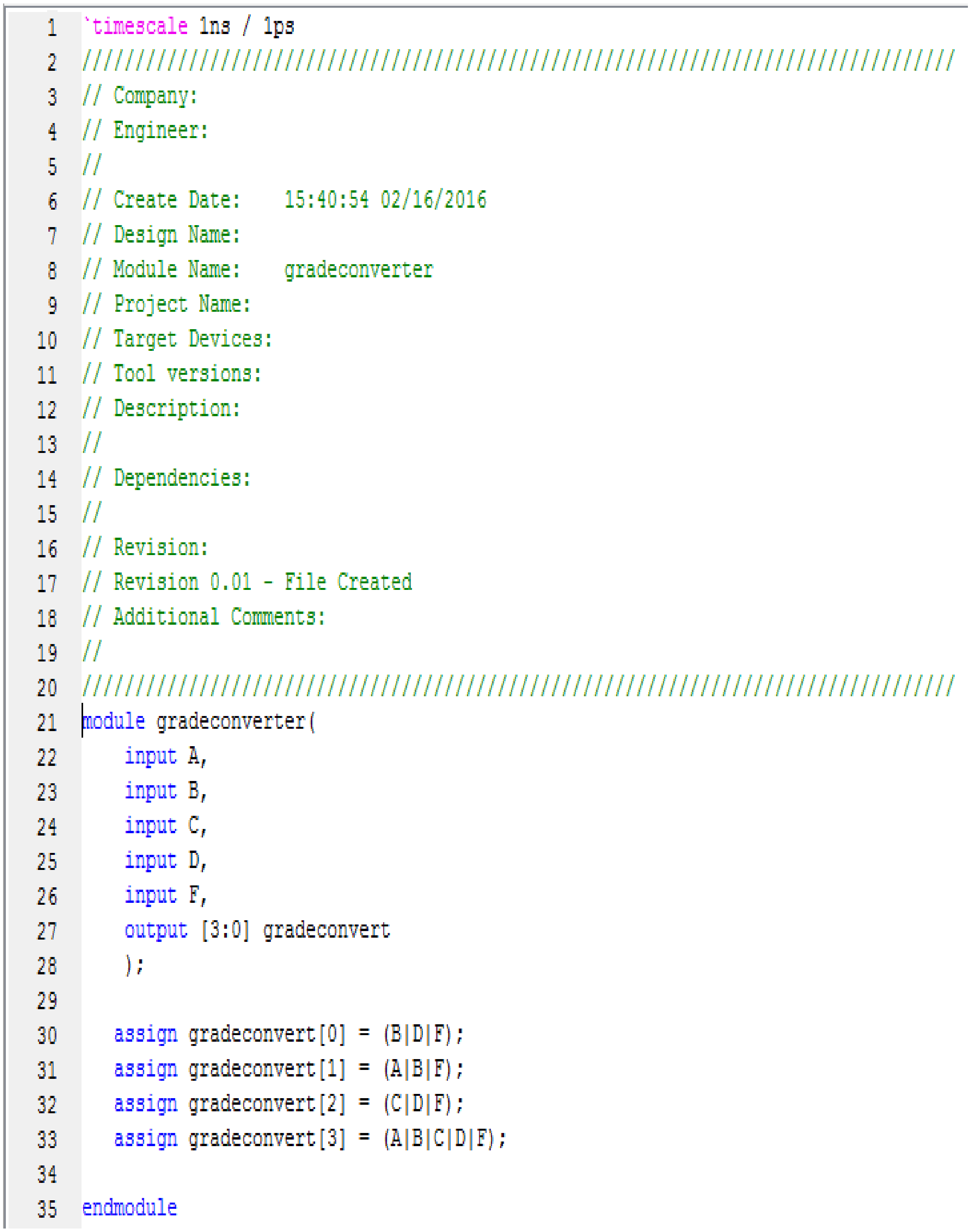


```verilog
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    15:13:16 02/16/2016
// Design Name:
// Module Name:    grade
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module grade(
    input [7:0] Q,
    output A,
    output B,
    output C,
    output D,
    output F
    );

   assign A = (~Q[7]&~Q[6]&~Q[5]&~Q[4]&~Q[3]&~Q[2]);
   assign B = (~Q[7]&~Q[6]&~Q[5]&~Q[4]&~Q[3]&Q[2]) &~A;
   assign C = (~Q[7]&~Q[6]&~Q[5]&~Q[4]&Q[3]&~B&~A);
   assign D = (~Q[7]&~Q[6]&~Q[5]&Q[4]&~Q[3]&~A&~B&~C);
   assign F = (~A&~B&~C&~D);

endmodule
```

Figure 5: Verilog corresponding to 8-bit bus input to determine grade



```verilog
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    15:40:54 02/16/2016
// Design Name:
// Module Name:    gradeconverter
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module gradeconverter(
    input A,
    input B,
    input C,
    input D,
    input F,
    output [3:0] gradeconvert
    );

   assign gradeconvert[0] = (B|D|F);
   assign gradeconvert[1] = (A|B|F);
   assign gradeconvert[2] = (C|D|F);
   assign gradeconvert[3] = (A|B|C|D|F);

endmodule
```

Figure 6: The Verilog of hot-encoding to send as a 4-bit bus

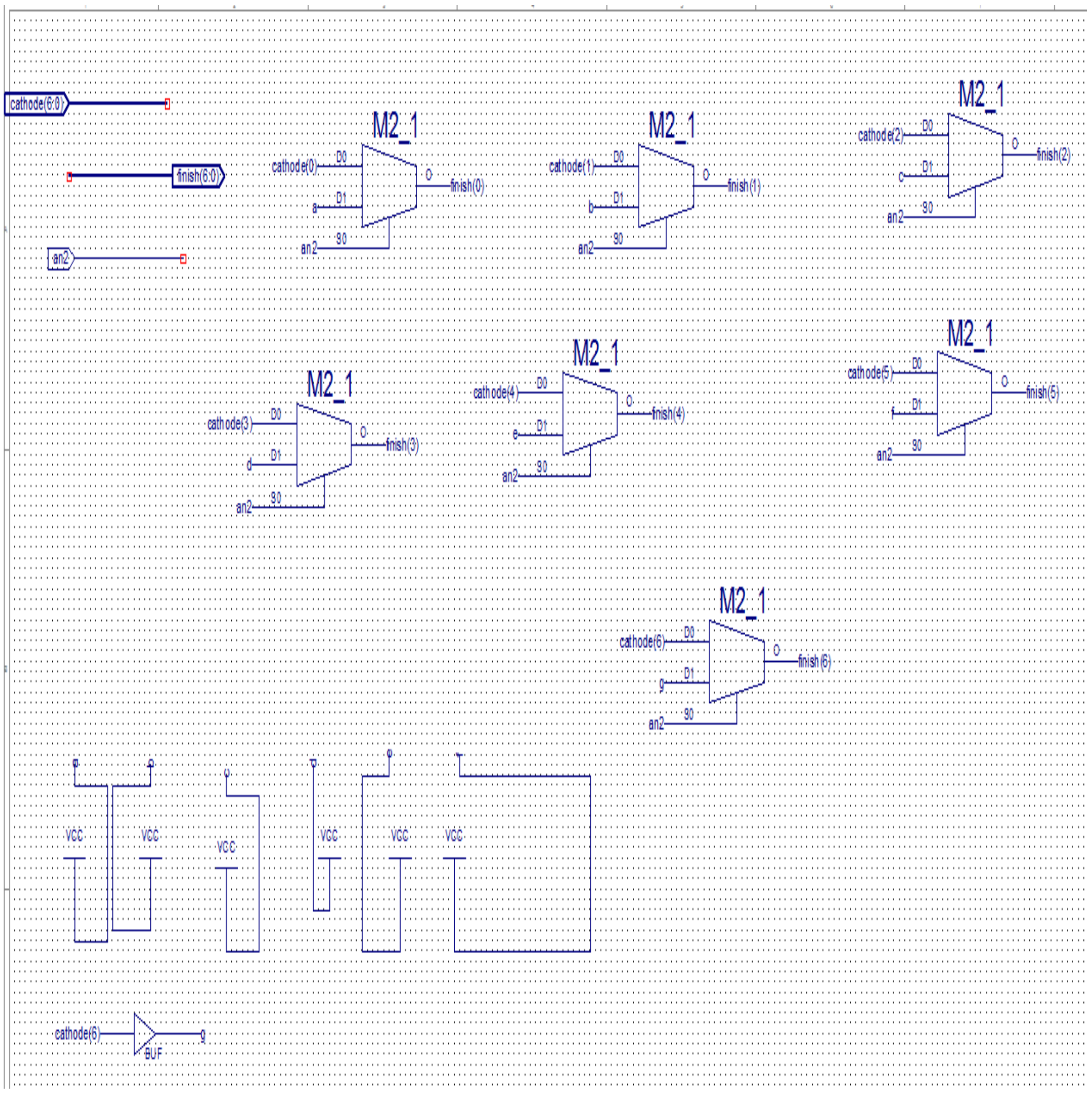


Figure 7: Negative Symbol to display when counter is negative decimal

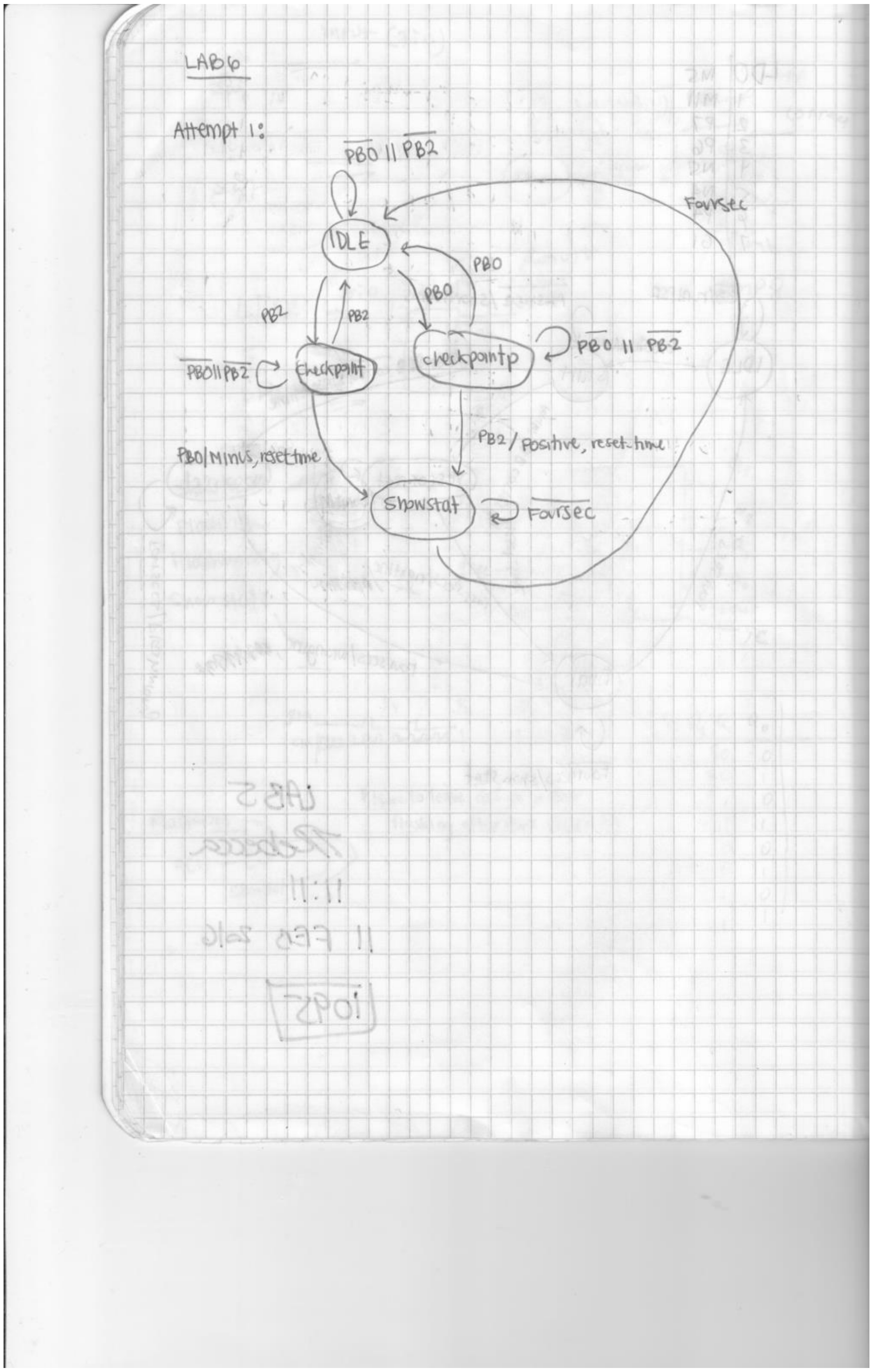


Figure 8: Attempt 1 of my state machine

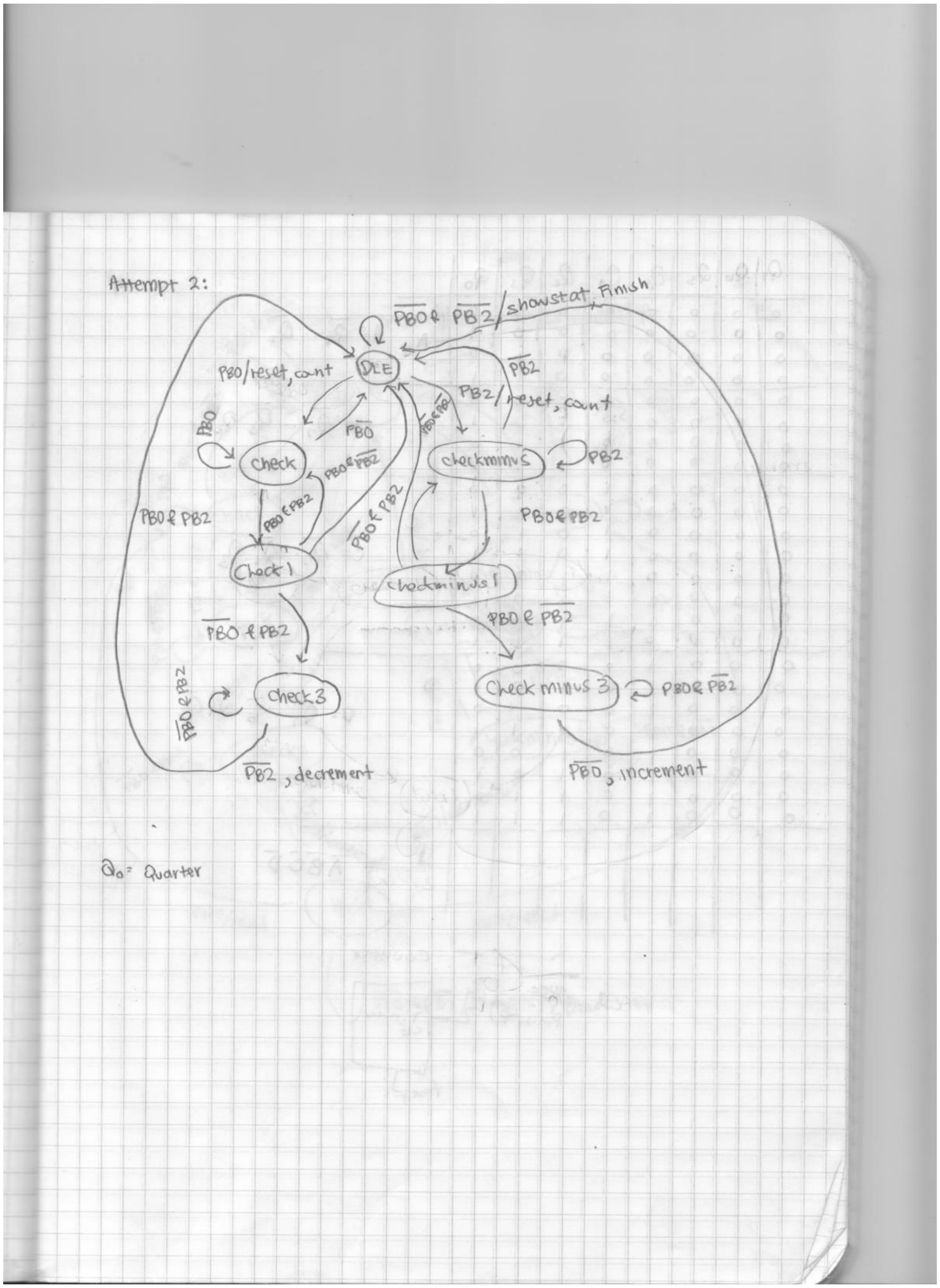
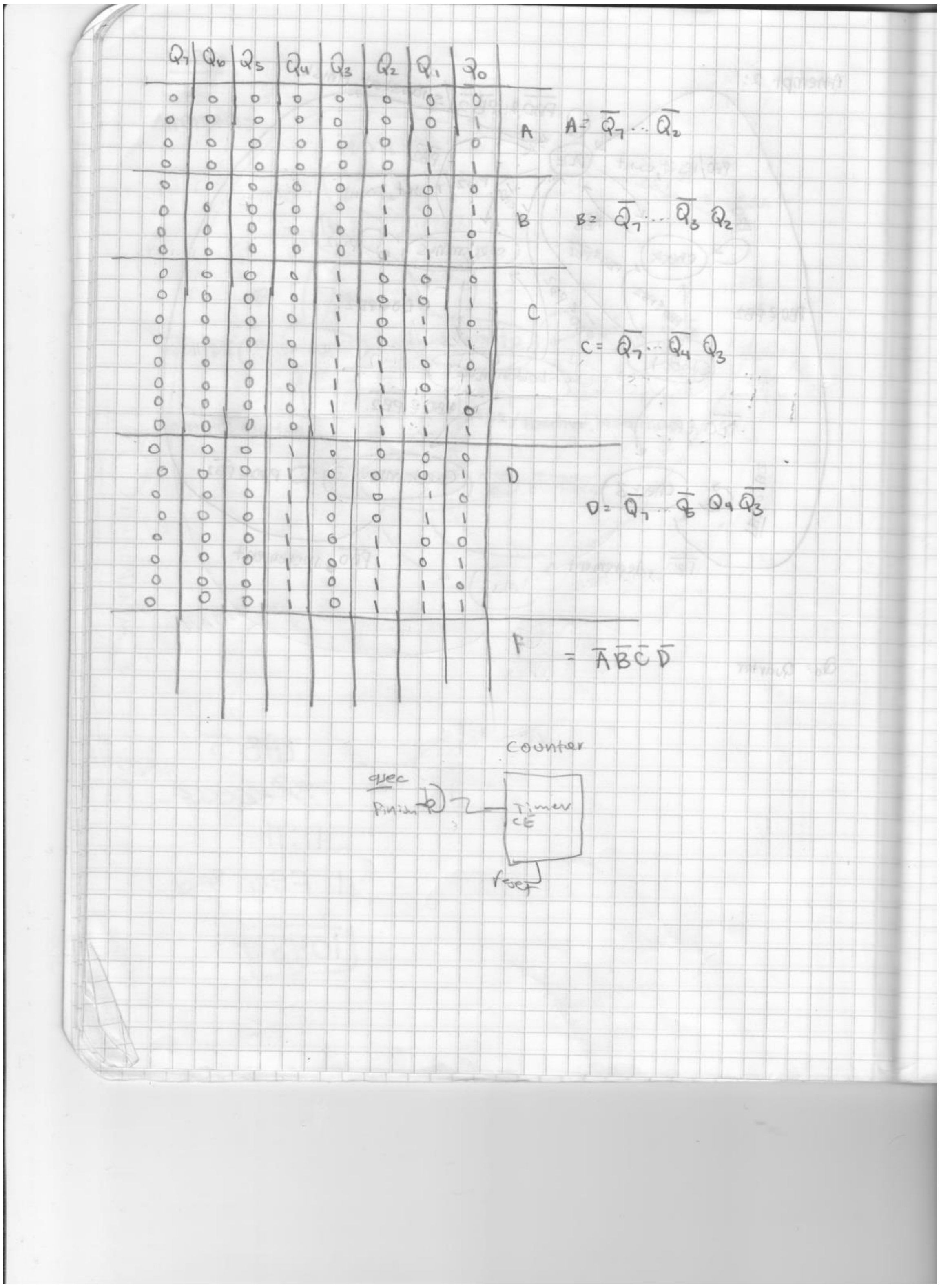


Figure 9: Attempt 2 of my state machine



Q_7	Q_6	Q_5	Q_4	Q_3	Q_2	Q_1	Q_0	
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	1	A
0	0	0	0	0	0	1	0	
0	0	0	0	0	0	1	1	
0	0	0	0	0	1	0	0	
0	0	0	0	0	1	0	1	B
0	0	0	0	0	1	1	0	
0	0	0	0	0	1	1	1	
0	0	0	0	1	0	0	0	
0	0	0	0	1	0	0	1	
0	0	0	0	1	0	1	0	C
0	0	0	0	1	0	1	1	
0	0	0	0	1	1	0	0	
0	0	0	0	1	1	0	1	
0	0	0	0	1	1	1	0	
0	0	0	0	1	1	1	1	
0	0	0	1	0	0	0	0	
0	0	0	1	0	0	0	1	D
0	0	0	1	0	0	1	0	
0	0	0	1	0	0	1	1	
0	0	0	1	0	1	0	0	
0	0	0	1	0	1	0	1	
0	0	0	1	0	1	1	0	
0	0	0	1	0	1	1	1	
								F

$A = \overline{Q_7} \cdots \overline{Q_2}$

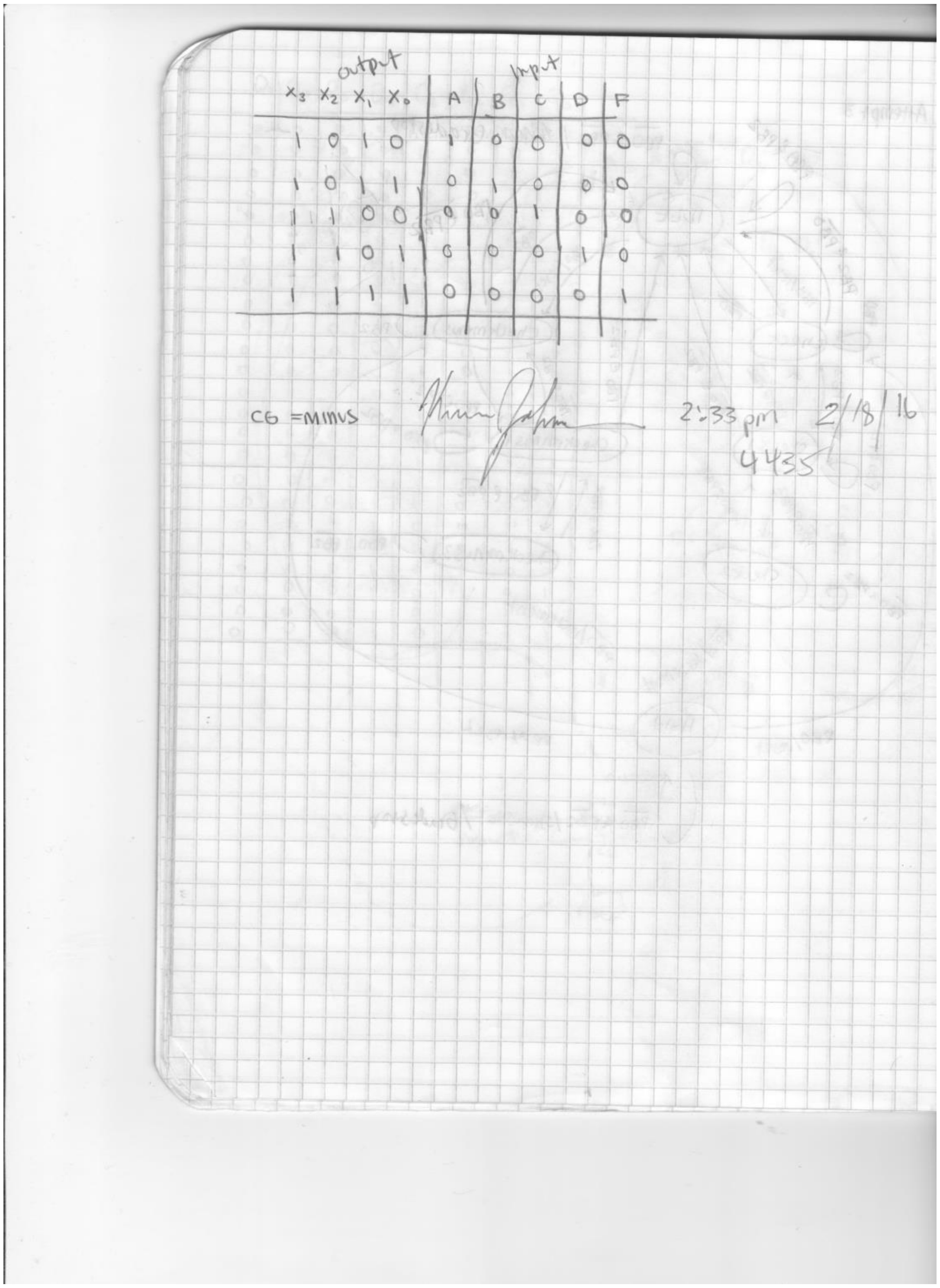
$B = \overline{Q_7} \cdots \overline{Q_3}\, Q_2$

$C = \overline{Q_7} \cdots \overline{Q_4}\, Q_3$

$D = \overline{Q_7} \cdots \overline{Q_5}\, Q_4 \overline{Q_3}$

$F = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$

Figure 10: How to determine grade from an 8-bit input



output				input				
X_3	X_2	X_1	X_0	A	B	C	D	F
1	0	1	0	1	0	0	0	0
1	0	1	1	0	1	0	0	0
1	1	0	0	0	0	1	0	0
1	1	0	1	0	0	0	1	0
1	1	1	1	0	0	0	0	1

CG =MINUS [illegible] 2:33 pm 2/18/16
4435

Figure 11: One Hot encoding input and a 4-bit output along with Kevin's beast signature