CE100 Lab Report 5

Add It

Justin Fortner

Lab 1C

TU/TH 1:30-3:20

11/15/17

**Description:**

The purpose of this lab was to implement an adding game. When a Go button was pushed a random number was given. The players would have to guess the answer and race to submit it. The score would go up if the answer is correct and down if wrong. All of these stages are implemented through the use of a state machine.

**Methods:**

**Part 1 - Random Number Generator:**

The random number generator was implemented as follows. 8 Dflip flops were hooked together in series. The output of one going into the input of the next. Then four flip flop outputs were fed into an xor gate. The output of this gate is the input of the first D flip flop inn the series mentioned above. A random 8 bit number is output from the circuit described.

**Part 2 - Time Counter:**

The time counter is a five bit up counter. The qsec signal is the clock for this module. Every increment in the count represents a quarter second. For this lab we needed a two second and an eight second counter. A high output was given when the counter reached 1000 to represent two seconds and 100000 to represent eight seconds.

**Part 3 - Score Keeping and Display:**

The score keeping module is a four bit up down counter capable of displaying the numbers -8 to 7. The counter is made up of four D flip flops. The counter will count up with the input of a 1 and down with the input of a 0. The negative numbers in this counter will be kept track of in two's complement and will need to be converted in order to be displayed correctly by the hex 7 seg.

**Part 4 - State Machine:**

My state machine is comprised of seven states. One for the start of the game to display the scores. One to show the number required to enter. One to accept the inputs. Then from this state either the left player can answer, the right player can answer, neither can answer or they can answer from the same time. No matter what all these states end back at the start of the state machine.

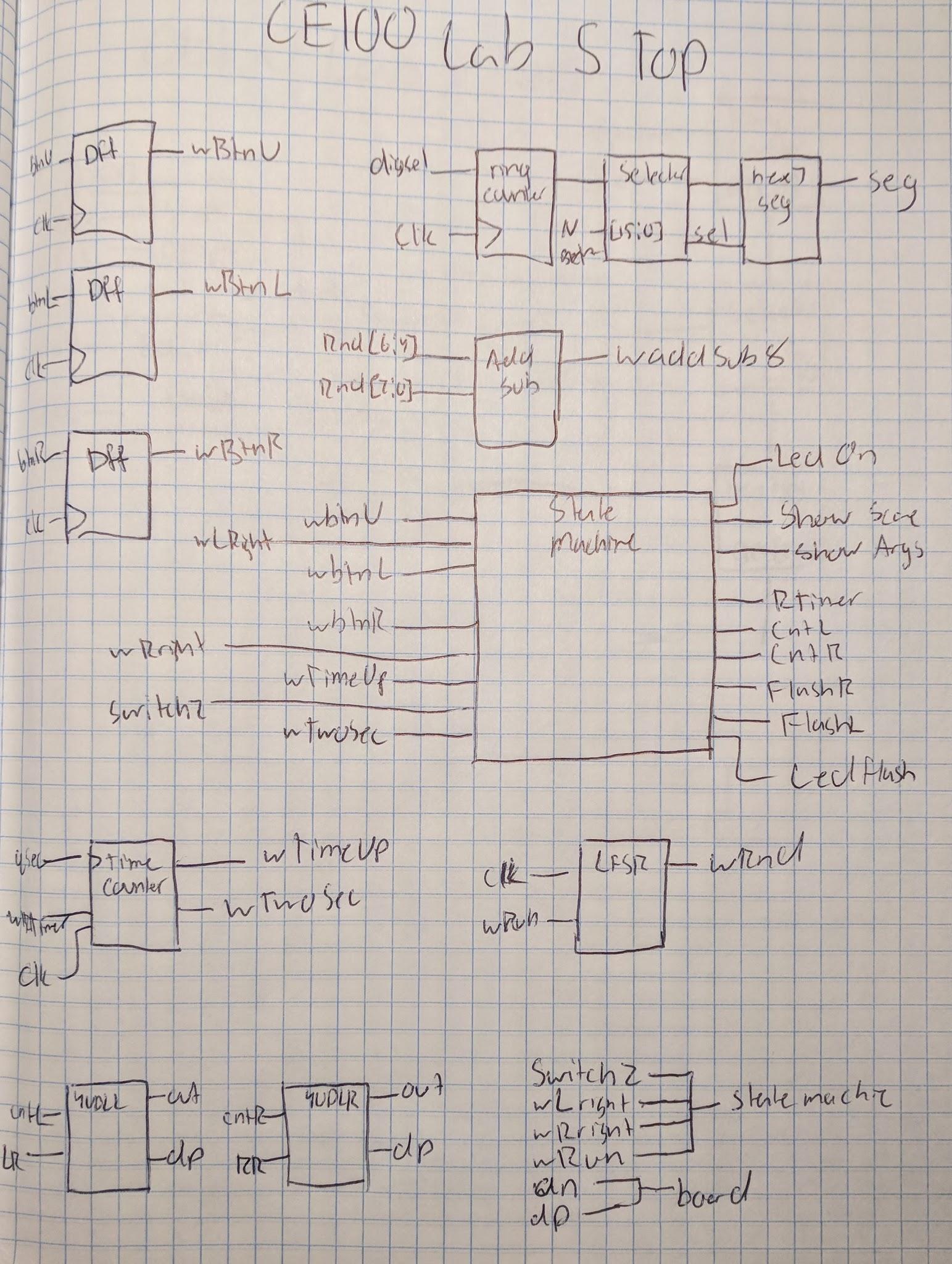
**Part 5 - Top Level:**

The top level for this lab tied all the different modules of this lab together. Allowed for the input of switches to an adder. Compared the random number to the input given by the player who answers. As well as keeping track of the score for both players. Also within this top module is converting the score from 2s complement to its positive counterpart if need be. Lastly this module contains the logic to display the decimal place when a negative score and flash the score when a player gives the correct answer.

**Results:**

**Design:**

**Top Level:**

****

wSwitchZ = ~sw[15] & ~sw[14] & ~sw[13] & ~sw[12] & ~sw[11] & ~sw[10] & ~sw[9] & ~sw[8] & ~sw[7] & ~sw[6] & ~sw[5] & ~sw[4] & ~sw[3] & ~sw[2] & ~sw[1] & ~sw[0];

wAddSu8\_4 = wAddSub8[3:0];

wLright = (sw[15] ~^ wAddSub8[3]) & (sw[14] ~^ wAddSub8[2]) & (sw[13] ~^ wAddSub8[1]) & (sw[12] ~^ wAddSub8[0]);

wRright = (sw[3] ~^ wAddSub8[3]) & (sw[2] ~^ wAddSub8[2]) & (sw[1] ~^ wAddSub8[1]) & (sw[0] ~^ wAddSub8[0]);

wRun = btnU & wShowScore;

led [11] = wLedOn | (wTimeCounter[1] & wLedFlash);

led [10] = wLedOn | (wTimeCounter[1] & wLedFlash);

led [9] = wLedOn | (wTimeCounter[1] & wLedFlash);

led [8] = wLedOn | (wTimeCounter[1] & wLedFlash);

led [7] = wLedOn | (wTimeCounter[1] & wLedFlash);

led [6] = wLedOn | (wTimeCounter[1] & wLedFlash);

led [5] = wLedOn | (wTimeCounter[1] & wLedFlash);

led [4] = wLedOn | (wTimeCounter[1] & wLedFlash);

an [3] = ~((wShowScore & wRingCounter[3] & ~wFlashL)|(wRingCounter[3] & wShowScore & wTimeCounter[1] & wFlashL));

an [2] = ~(wShowArgs & wRingCounter[2]);

an [1] = ~(wShowArgs & wRingCounter[1]);

an [0] = ~((wShowScore & wRingCounter[0] & ~wFlashR)| (wRingCounter[0] & wShowScore & wTimeCounter[1] & wFlashR));

dp = ~((wDpL & wRingCounter[3]) | (wDpR & wRingCounter[0]));

**Random Number Generator:**

The random number generator was implemented as follows. 8 Dflip flops were hooked together in series. The output of one going into the input of the next. Then four flip flop outputs were fed into an xor gate. The output of this gate is the input of the first D flip flop inn the series mentioned above. A random 8 bit number is output from the circuit described.

**Time Counter:**

Equations are obtained from the table below.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CE** | **Q5** | **Q4** | **Q3** | **Q2** | **Q1** | **Q0** | **Q5+** | **Q4+** | **Q3** | **Q2+** | **Q1+** | **Q0+** |
| **1** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** |
| **1** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **0** | **1** | **0** |
| **1** | **0** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **0** | **0** | **1** | **1** |
| **1** | **0** | **0** | **0** | **0** | **1** | **1** | **0** | **0** | **0** | **1** | **0** | **0** |
| **1** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **1** |
| **1** | **0** | **0** | **0** | **1** | **0** | **1** | **0** | **0** | **0** | **1** | **1** | **0** |
| **1** | **0** | **0** | **0** | **1** | **1** | **0** | **0** | **0** | **0** | **1** | **1** | **1** |
| **1** | **0** | **0** | **0** | **1** | **1** | **1** | **0** | **0** | **1** | **0** | **0** | **0** |
| **1** | **0** | **0** | **1** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **0** | **1** |
| **1** | **0** | **0** | **1** | **0** | **0** | **1** | **0** | **0** | **1** | **0** | **1** | **0** |
| **1** | **0** | **0** | **1** | **0** | **1** | **0** | **0** | **0** | **1** | **0** | **1** | **1** |
| **1** | **0** | **0** | **1** | **0** | **1** | **1** | **0** | **0** | **1** | **1** | **0** | **0** |
| **1** | **0** | **0** | **1** | **1** | **0** | **0** | **0** | **0** | **1** | **1** | **0** | **1** |
| **1** | **0** | **0** | **1** | **1** | **0** | **1** | **0** | **0** | **1** | **1** | **1** | **0** |
| **1** | **0** | **0** | **1** | **1** | **1** | **0** | **0** | **0** | **1** | **1** | **1** | **1** |
| **1** | **0** | **0** | **1** | **1** | **1** | **1** | **0** | **1** | **0** | **0** | **0** | **0** |
| **1** | **0** | **1** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **1** |
| **1** | **0** | **1** | **0** | **0** | **0** | **1** | **0** | **1** | **0** | **0** | **1** | **0** |
| **1** | **0** | **1** | **0** | **0** | **1** | **0** | **0** | **1** | **0** | **0** | **1** | **1** |
| **1** | **0** | **1** | **0** | **0** | **1** | **1** | **0** | **1** | **0** | **1** | **0** | **0** |
| **1** | **0** | **1** | **0** | **1** | **0** | **0** | **0** | **1** | **0** | **1** | **0** | **1** |
| **1** | **0** | **1** | **0** | **1** | **0** | **1** | **0** | **1** | **0** | **1** | **1** | **0** |
| **1** | **0** | **1** | **0** | **1** | **1** | **0** | **0** | **1** | **0** | **1** | **1** | **1** |
| **1** | **0** | **1** | **0** | **1** | **1** | **1** | **0** | **1** | **1** | **0** | **0** | **0** |
| **1** | **0** | **1** | **1** | **0** | **0** | **0** | **0** | **1** | **1** | **0** | **0** | **1** |
| **1** | **0** | **1** | **1** | **0** | **0** | **1** | **0** | **1** | **1** | **0** | **1** | **0** |
| **1** | **0** | **1** | **1** | **0** | **1** | **0** | **0** | **1** | **1** | **0** | **1** | **1** |
| **1** | **0** | **1** | **1** | **0** | **1** | **1** | **0** | **1** | **1** | **1** | **0** | **0** |
| **1** | **0** | **1** | **1** | **1** | **0** | **0** | **0** | **1** | **1** | **1** | **0** | **1** |
| **1** | **0** | **1** | **1** | **1** | **0** | **1** | **0** | **1** | **1** | **1** | **1** | **0** |
| **1** | **0** | **1** | **1** | **1** | **1** | **0** | **0** | **1** | **1** | **1** | **1** | **1** |
| **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **0** | **0** | **0** | **0** |
| **1** | **1** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |

wD[0] = (~CE&t[0])|(CE&~t[0]);

wD[1] = (~CE&t[1])|(t[1]&~t[0])|(CE&~t[1]&t[0]);

wD[2] = (~CE&t[2])|(t[2]&~t[1])|(t[2]&~t[0])|(CE&~t[2]&t[1]&t[0]);

wD[3] = (~CE&t[3])|(t[3]&~t[2])|(t[3]&~t[1])|(t[3]&~t[0])|(CE&~t[3]&t[2]&t[1]&t[0]);

wD[4] = (~CE&t[4])|(t[4]&~t[3])|(t[4]&~t[2])|(t[4]&~t[1])|(t[4]&~t[0])|(CE&~t[4]&t[3]&t[2]&t[1]&t[0]);

wD[5] = (~CE&t[5])|(t[5]&~t[4])|(t[5]&~t[3])|(t[5]&~t[2])|(t[5]&~t[1])|(t[5]&~t[0])|(CE&~t[5]&t[4]&t[3]&t[2]&t[1]&t[0]);

TimeUp = t[5] & ~t[4] & ~t[3] & ~t[2]& ~t[1]& ~t[0];

TwoSec = ~t[5] & ~t[4] & t[3] & ~t[2]& ~t[1]& ~t[0];

**Score Keeping and Display:**

Equations obtained from the table below.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CE** | **UD** | **Q3** | **Q2** | **Q1** | **Q0** | **Q3+** | **Q2+** | **Q1+** | **Q0+** |
| **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **0** | **0** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **0** |
| **0** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **0** | **1** |
| **0** | **0** | **0** | **0** | **1** | **1** | **0** | **0** | **0** | **1** |
| **0** | **0** | **0** | **1** | **0** | **0** | **0** | **0** | **1** | **0** |
| **0** | **0** | **0** | **1** | **0** | **1** | **0** | **0** | **1** | **0** |
| **0** | **0** | **0** | **1** | **1** | **0** | **0** | **0** | **1** | **1** |
| **0** | **0** | **0** | **1** | **1** | **1** | **0** | **0** | **1** | **1** |
| **0** | **0** | **1** | **0** | **0** | **0** | **0** | **1** | **0** | **0** |
| **0** | **0** | **1** | **0** | **0** | **1** | **0** | **1** | **0** | **0** |
| **0** | **0** | **1** | **0** | **1** | **0** | **0** | **1** | **0** | **1** |
| **0** | **0** | **1** | **0** | **1** | **1** | **0** | **1** | **0** | **1** |
| **0** | **0** | **1** | **1** | **0** | **0** | **1** | **1** | **0** | **0** |
| **0** | **0** | **1** | **1** | **0** | **1** | **1** | **1** | **0** | **1** |
| **0** | **0** | **1** | **1** | **1** | **0** | **1** | **1** | **1** | **0** |
| **0** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **0** | **1** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **0** | **1** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **1** |
| **0** | **1** | **0** | **0** | **1** | **0** | **0** | **0** | **1** | **0** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **0** | **0** | **1** | **1** | **0** | **0** | **1** | **1** |
| **0** | **1** | **0** | **1** | **0** | **0** | **0** | **1** | **0** | **0** |
| **0** | **1** | **0** | **1** | **0** | **1** | **0** | **1** | **0** | **1** |
| **0** | **1** | **0** | **1** | **1** | **0** | **0** | **1** | **1** | **0** |
| **0** | **1** | **0** | **1** | **1** | **1** | **0** | **1** | **1** | **1** |
| **0** | **1** | **1** | **0** | **0** | **0** | **1** | **0** | **0** | **0** |
| **0** | **1** | **1** | **0** | **0** | **1** | **1** | **0** | **0** | **1** |
| **0** | **1** | **1** | **0** | **1** | **0** | **1** | **0** | **1** | **0** |
| **0** | **1** | **1** | **0** | **1** | **1** | **1** | **0** | **1** | **1** |
| **0** | **1** | **1** | **1** | **0** | **0** | **1** | **1** | **0** | **0** |
| **0** | **1** | **1** | **1** | **0** | **1** | **1** | **1** | **0** | **1** |
| **0** | **1** | **1** | **1** | **1** | **0** | **1** | **1** | **1** | **0** |
| **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **1** | **0** | **0** | **0** | **0** | **0** | **1** | **1** | **1** | **1** |
| **1** | **0** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **0** |
| **1** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **0** | **1** |
| **1** | **0** | **0** | **0** | **1** | **1** | **0** | **0** | **1** | **0** |
| **1** | **0** | **0** | **1** | **0** | **0** | **0** | **0** | **1** | **1** |
| **1** | **0** | **0** | **1** | **0** | **1** | **0** | **1** | **0** | **0** |
| **1** | **0** | **0** | **1** | **1** | **0** | **0** | **1** | **0** | **1** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **0** | **0** | **1** | **1** | **1** | **0** | **1** | **1** | **0** |
| **1** | **0** | **1** | **0** | **0** | **0** | **0** | **1** | **1** | **1** |
| **1** | **0** | **1** | **0** | **0** | **1** | **1** | **0** | **0** | **0** |
| **1** | **0** | **1** | **0** | **1** | **0** | **1** | **0** | **0** | **1** |
| **1** | **0** | **1** | **0** | **1** | **1** | **1** | **0** | **1** | **0** |
| **1** | **0** | **1** | **1** | **0** | **0** | **1** | **0** | **1** | **1** |
| **1** | **0** | **1** | **1** | **0** | **1** | **1** | **1** | **0** | **0** |
| **1** | **1** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** |
| **1** | **1** | **0** | **0** | **0** | **1** | **0** | **0** | **1** | **0** |
| **1** | **1** | **0** | **0** | **1** | **0** | **0** | **0** | **1** | **1** |
| **1** | **1** | **0** | **0** | **1** | **1** | **0** | **1** | **0** | **0** |
| **1** | **1** | **0** | **1** | **0** | **0** | **0** | **1** | **0** | **1** |
| **1** | **1** | **0** | **1** | **0** | **1** | **0** | **1** | **1** | **0** |
| **1** | **1** | **0** | **1** | **1** | **0** | **0** | **1** | **1** | **1** |
| **1** | **1** | **0** | **1** | **1** | **1** | **1** | **0** | **0** | **0** |
| **1** | **1** | **1** | **0** | **0** | **0** | **1** | **0** | **0** | **1** |
| **1** | **1** | **1** | **0** | **0** | **1** | **1** | **0** | **1** | **0** |
| **1** | **1** | **1** | **0** | **1** | **0** | **1** | **0** | **1** | **1** |
| **1** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **0** | **0** |
| **1** | **1** | **1** | **1** | **0** | **0** | **1** | **0** | **1** | **1** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **1** | **1** | **1** | **0** | **1** | **1** | **1** | **0** | **0** |
| **1** | **1** | **1** | **1** | **1** | **0** | **1** | **1** | **0** | **1** |
| **1** | **1** | **1** | **1** | **1** | **1** | **0** | **0** | **0** | **0** |

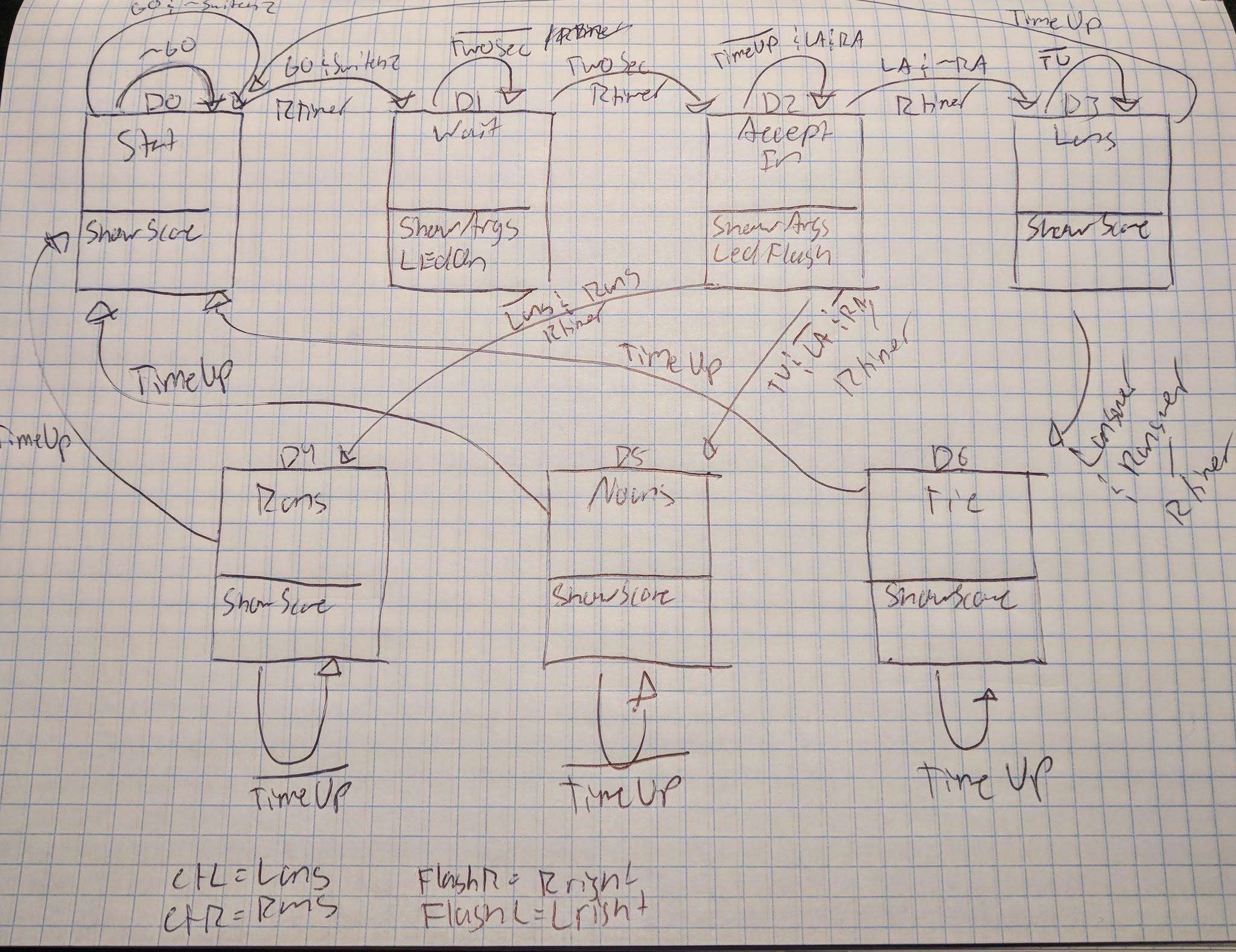
wD[0] = (~CE&c[0])|(CE&~c[3]&~c[0])|(CE&c[1]&~c[0])|(CE&c[2]&~c[0])|(CE&UD&~c[0])|(UD&~c[3]&c[2]&c[1]&c[0]);

wD[1] = (~CE&c[1])|(~UD&c[1]&c[0])|(UD&c[1]&~c[0])|(~c[3]&c[2]&c[1]&c[0])|(CE&UD&~c[1]&c[0])|(CE&~UD&~c[3]&~c[1]&~c[0])|(CE&~UD&c[2]&~c[1]&~c[0]);

wD[2] = (~CE&c[2])|(~UD&c[2]&c[0])|(~c[3]&c[2]&c[0])|(c[2]&c[1]&~c[0])|(UD&c[2]&~c[1])|(CE&UD&~c[2]&c[1]&c[0])|(CE&~UD&~c[3]&~c[2]&~c[1]&~c[0]);

wD[3] = (~CE&c[3])|(~UD&c[3])|(c[3]&~c[2])|(c[3]&~c[1])|(c[3]&~c[0])|(CE&~UD&~c[2]&~c[1]&~c[0]);

**State Machine:**

****

D[0] = (Q[0] & ~Go)|(Q[3] & TimeUp)|(Q[4] & TimeUp)|(Q[5] & TimeUp)|(Q[6] & TimeUp)|(Q[0] & Go & ~SwitchZ);

D[1] = (Q[0] & Go & SwitchZ)|(Q[1] & ~TwoSec);

D[2] = (Q[1] & TwoSec)|(Q[2] & ~TimeUp & ~Lanswer & ~Ranswer);

D[3] = (Q[2] & Lanswer & ~Ranswer)|(Q[3] & ~TimeUp);

D[4] = (Q[2] & ~Lanswer & Ranswer)|(Q[4] & ~TimeUp);

D[5] = (Q[2] & ~Lanswer & ~Ranswer & TimeUp)|(Q[5] & ~TimeUp);

D[6] = (Q[2] & Lanswer & Ranswer)|(Q[6] & ~TimeUp);

ShowScore = Q[0]|Q[3]|Q[4]|Q[5]|Q[6];

ShowArgs = Q[1]|Q[2];

RTimer = (Q[0] & Go & SwitchZ)|(Q[1] & TwoSec)|(Q[2] & Lanswer & ~Ranswer)|(Q[2] & ~Lanswer & Ranswer)|(Q[2] & ~Lanswer & ~Ranswer & TimeUp)|(Q[2] & Lanswer & Ranswer);

cntL = Lanswer & Q[2];

cntR = Ranswer & Q[2];

FlashR = Rright & Q[4];

FlashL = Lright & Q[3];

LedOn = Q[1];

LedFlash = Q[2];

**Testing & Simulation:**

I tested my state machine in order to make sure it made it successfully through all states. This was done by going selecting accurate values that could be input by the board in order to trigger states as I had designed them. The most important state that needed to be tested was the tie because that is nearly impossible to test on the board.

**Lab Questions:**

No lab questions for this lab.

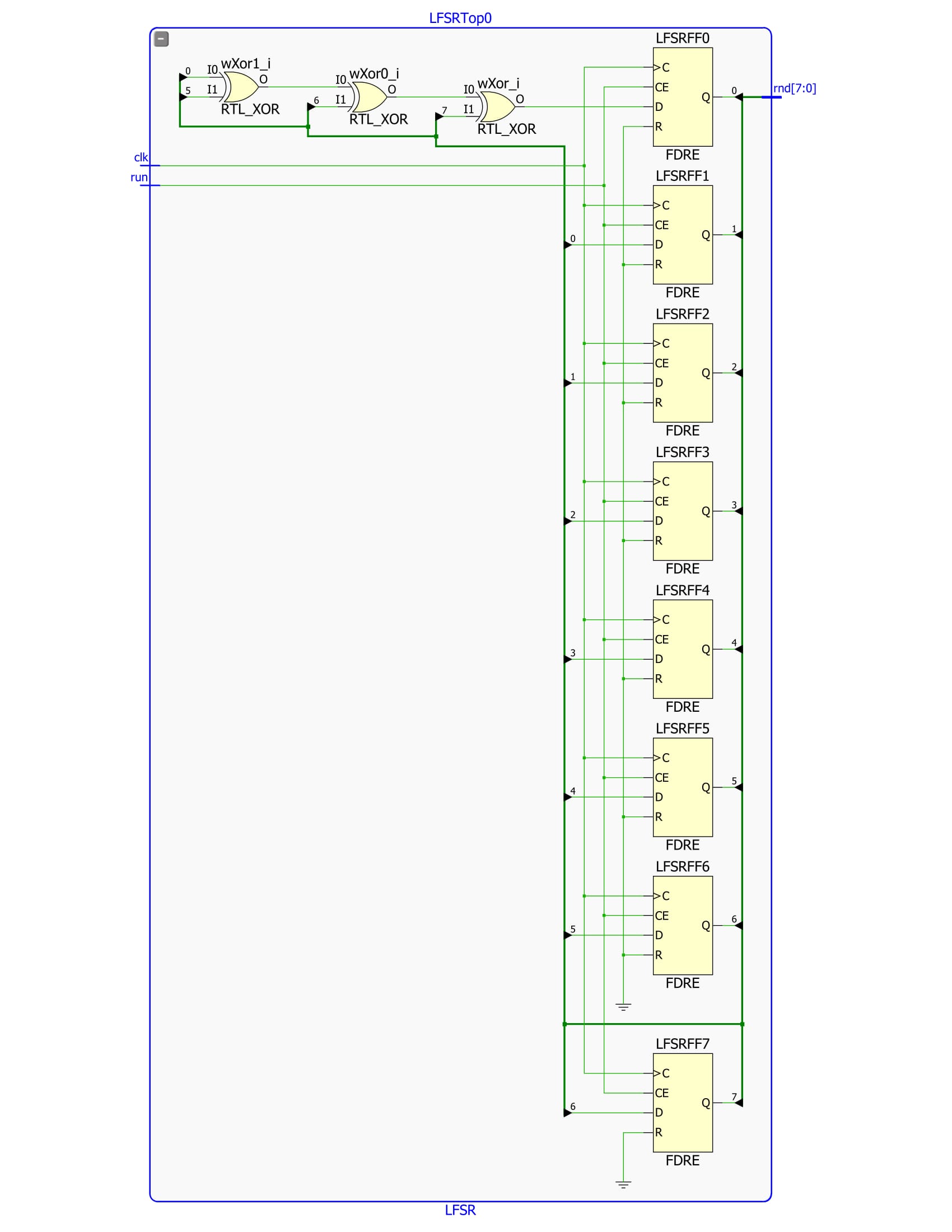
**Conclusion:**

My main take away from this lab was the state machine. I learned how to design and implement a state machine. I understand fully that a state machine can take many iterations and hours of perfecting in order for it to perform exactly how it is intended. However, I also believe that because I have put in the time I will be able to create state machines more quickly or create more complex ones. The state machine was the biggest difficulty that I had with this lab and is the main reason why it took me so many hours to complete. I would love to optimize my state machine further as well as my top level.

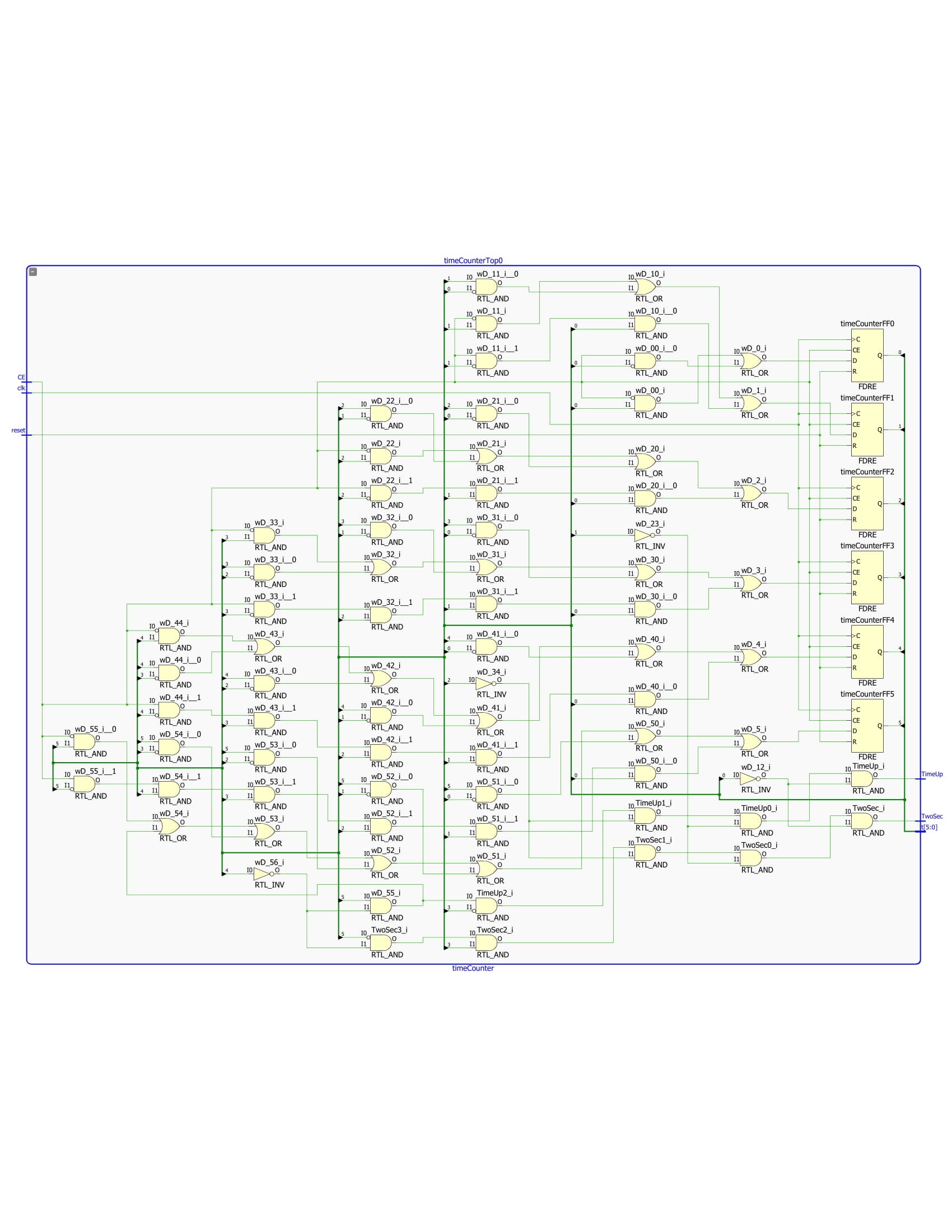
**Appendix:**

**Schematics:**

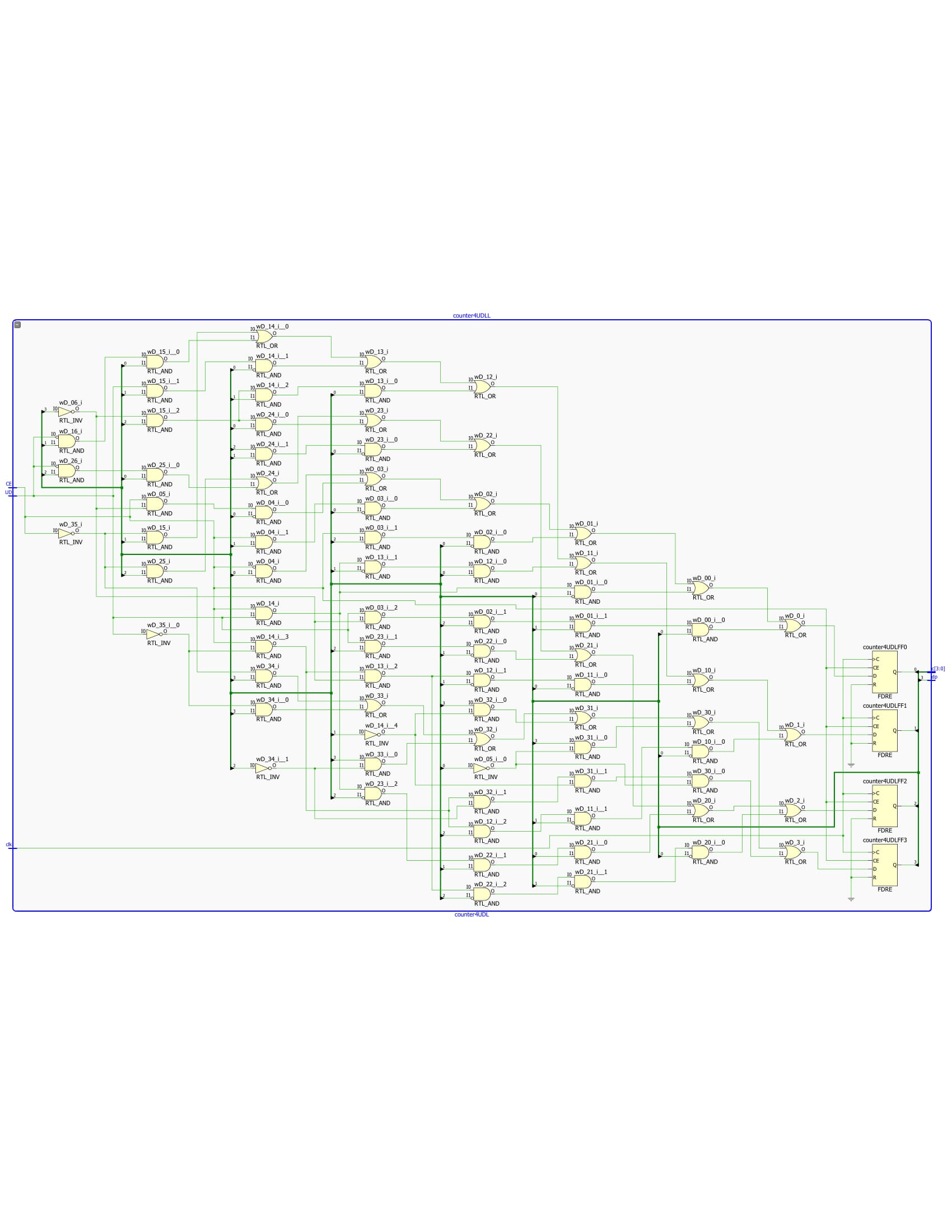
**Random Number Generator:**

****

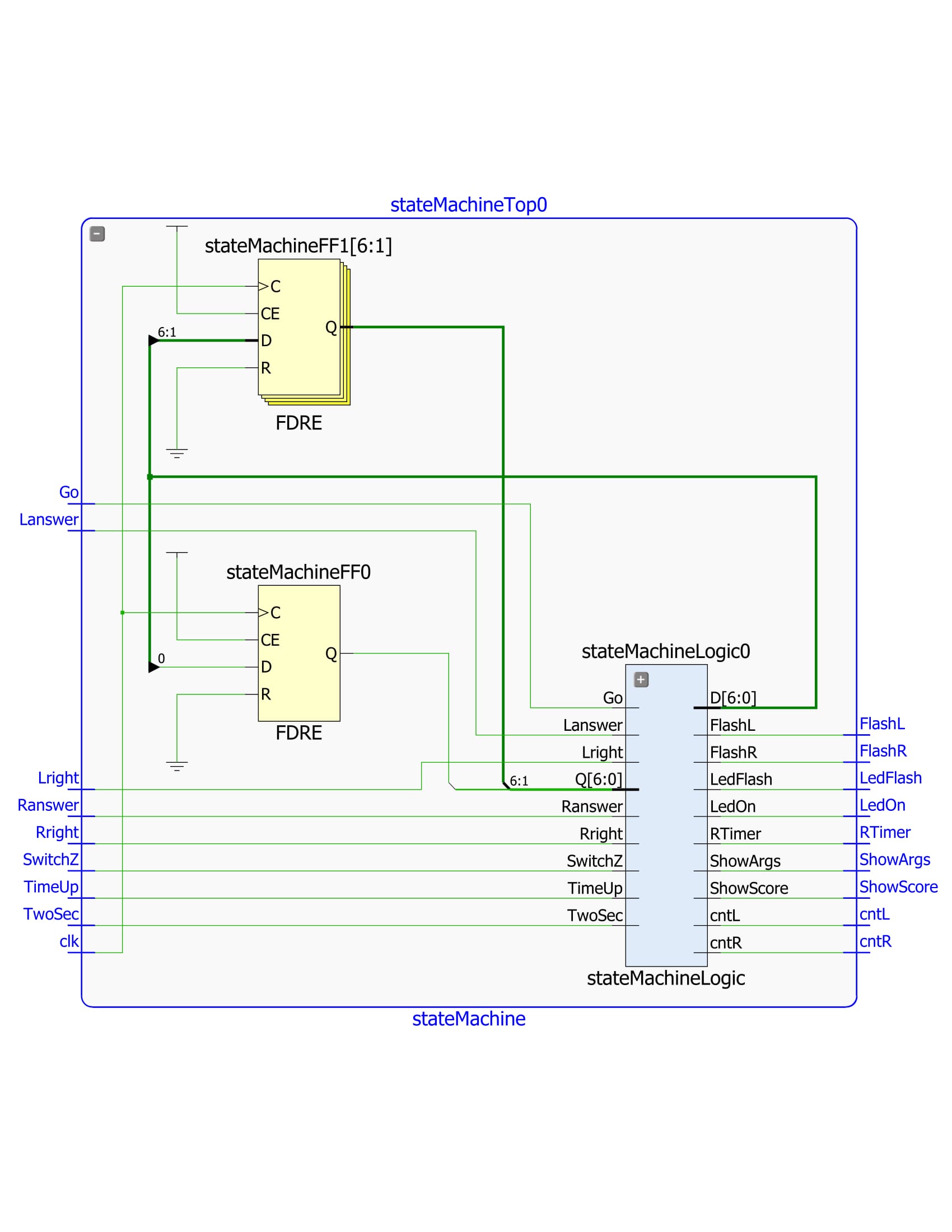
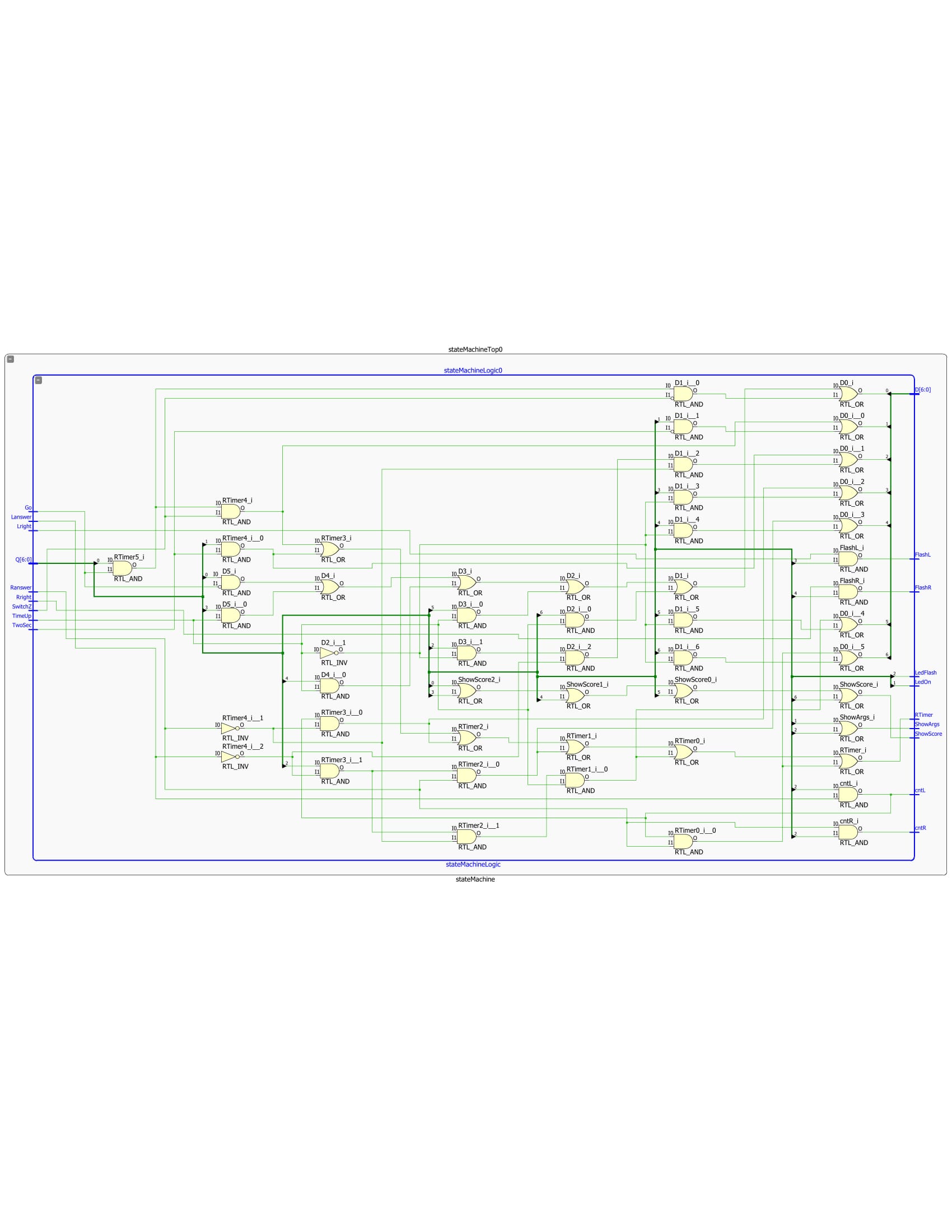
**Time Counter:**

****

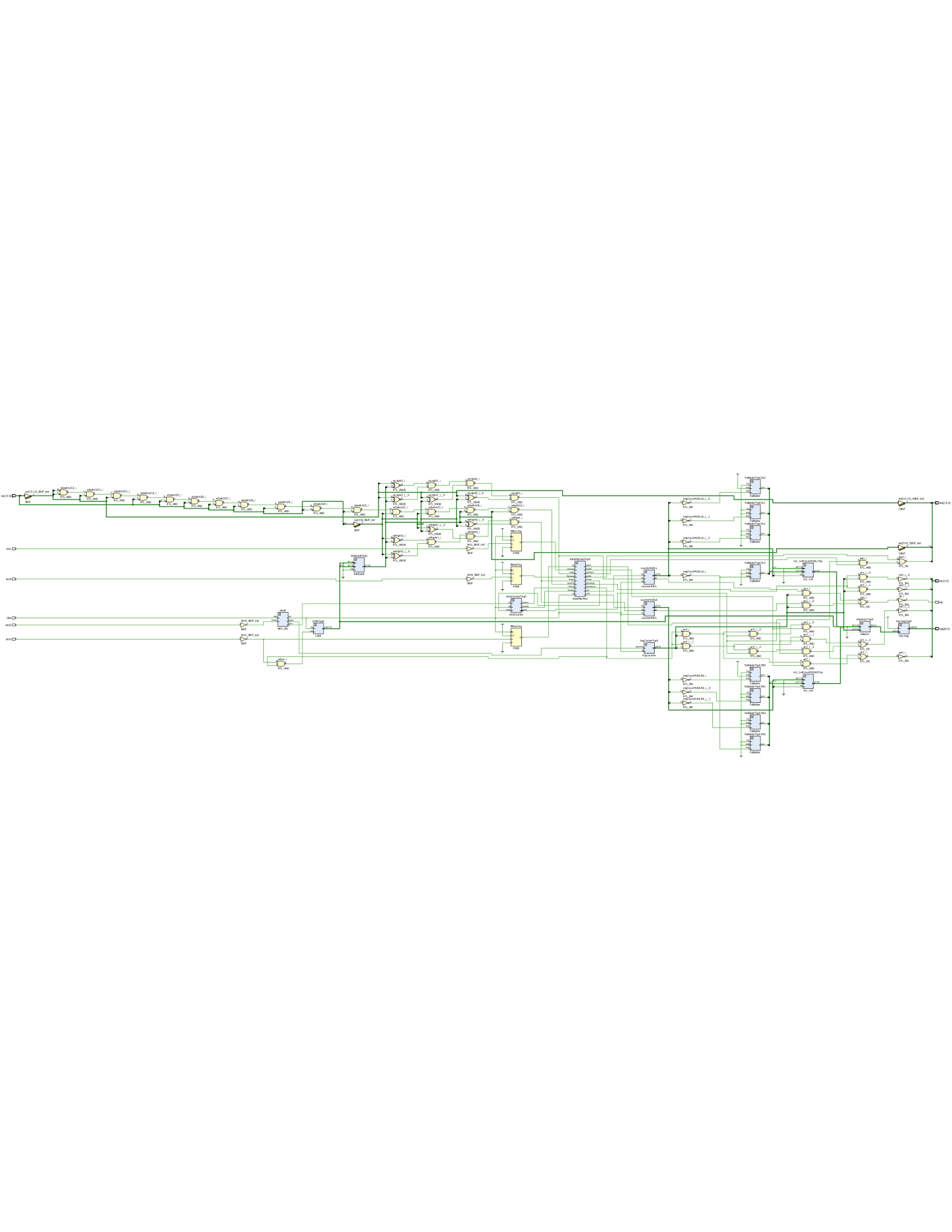
**Score Keeping and Display:**

****

**State Machine Top & Logic:**

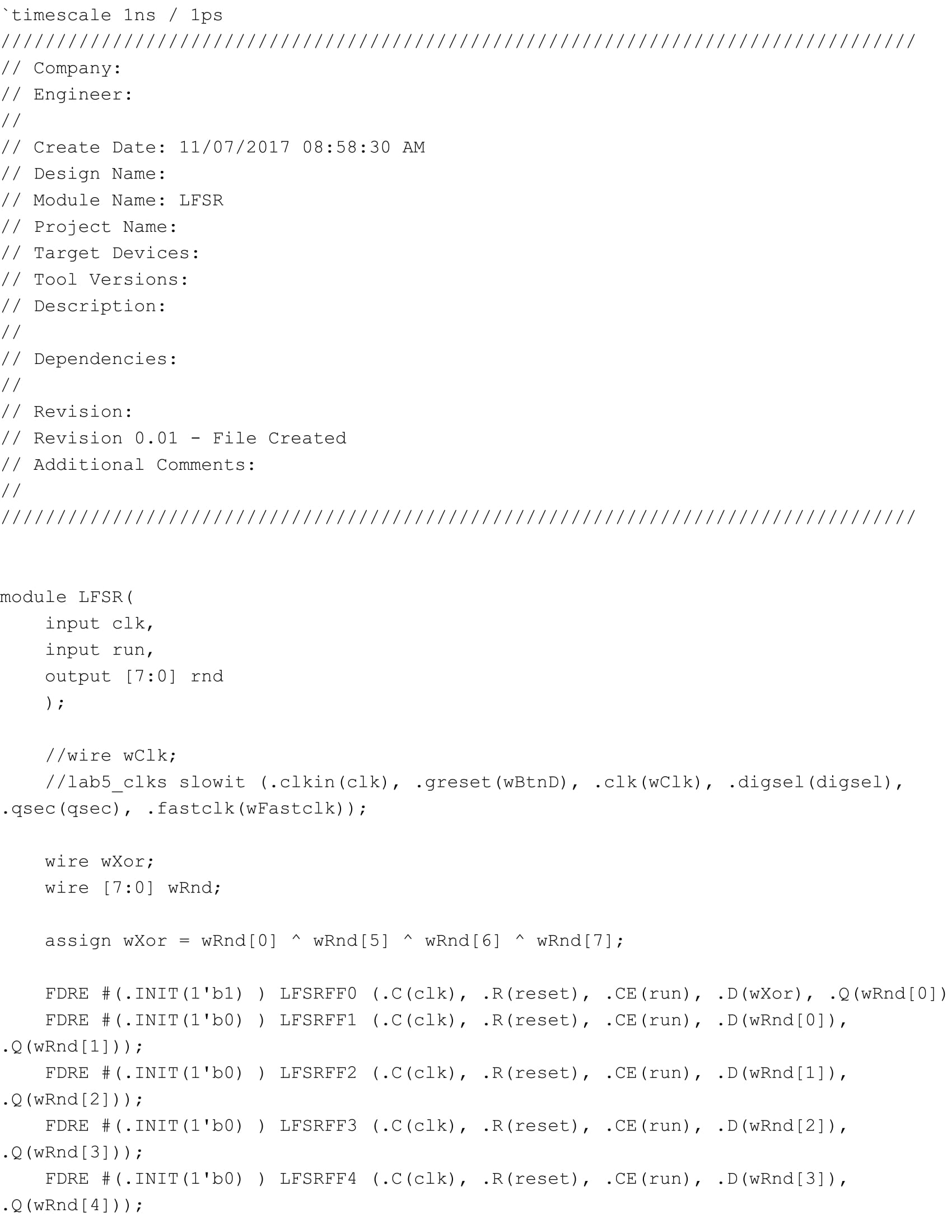


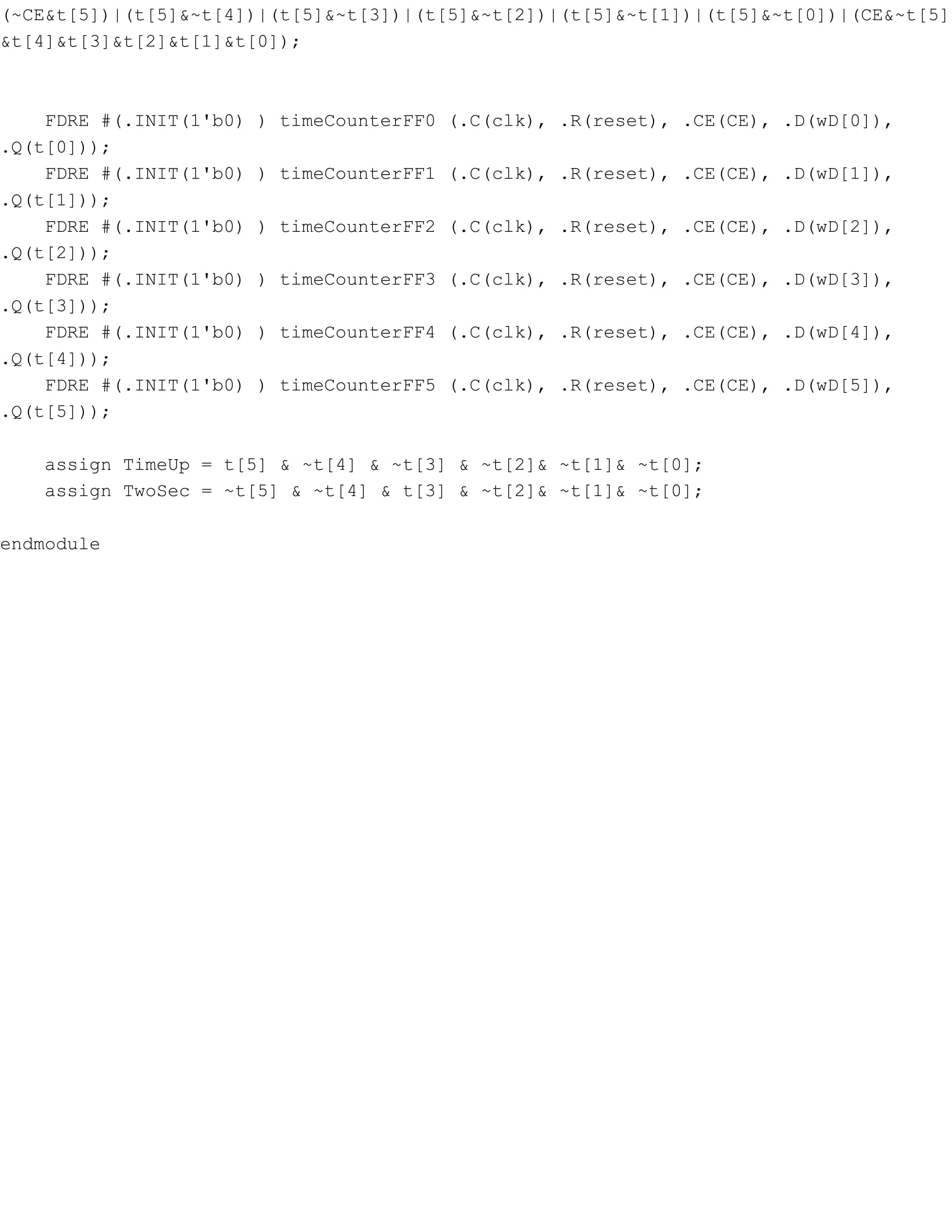
**Top Level:**

****

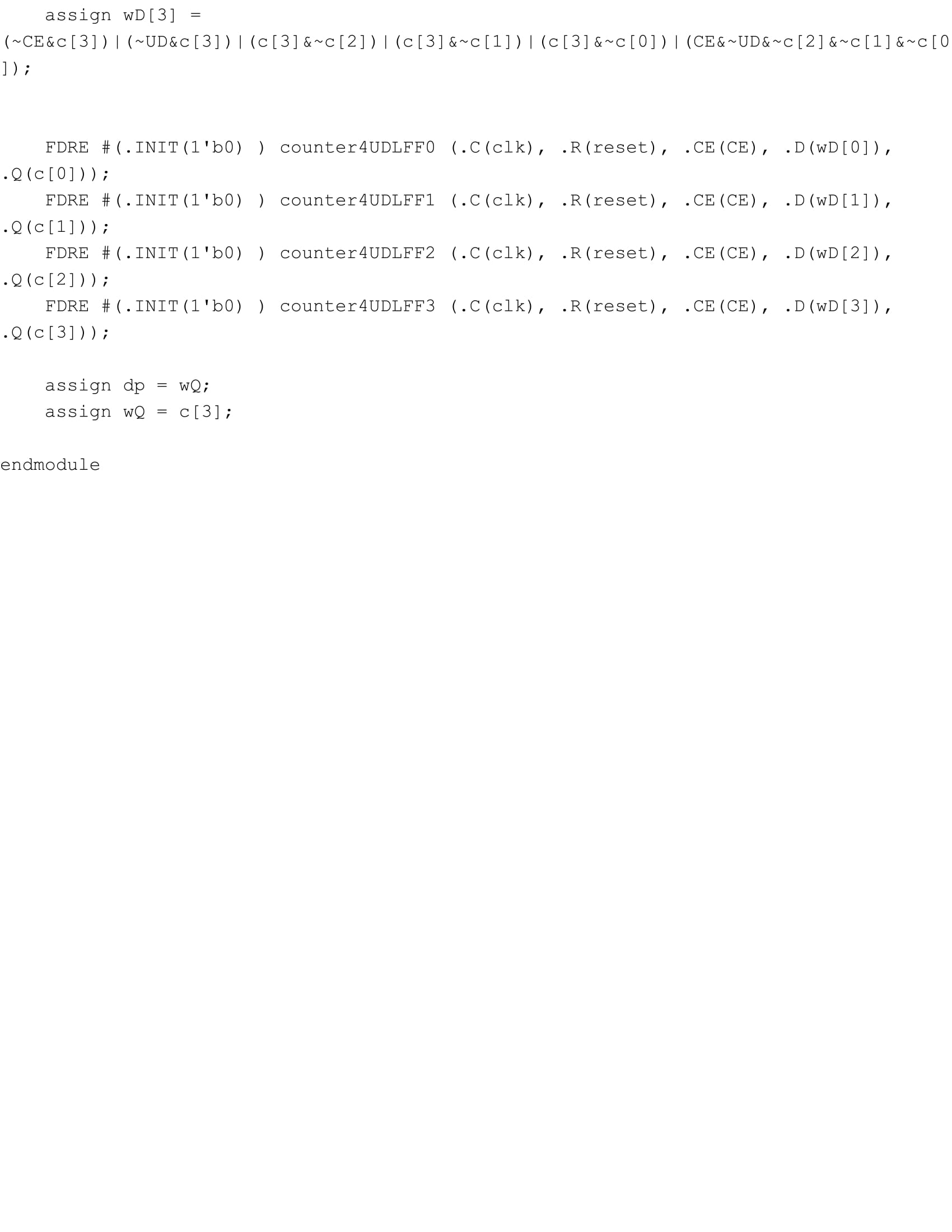
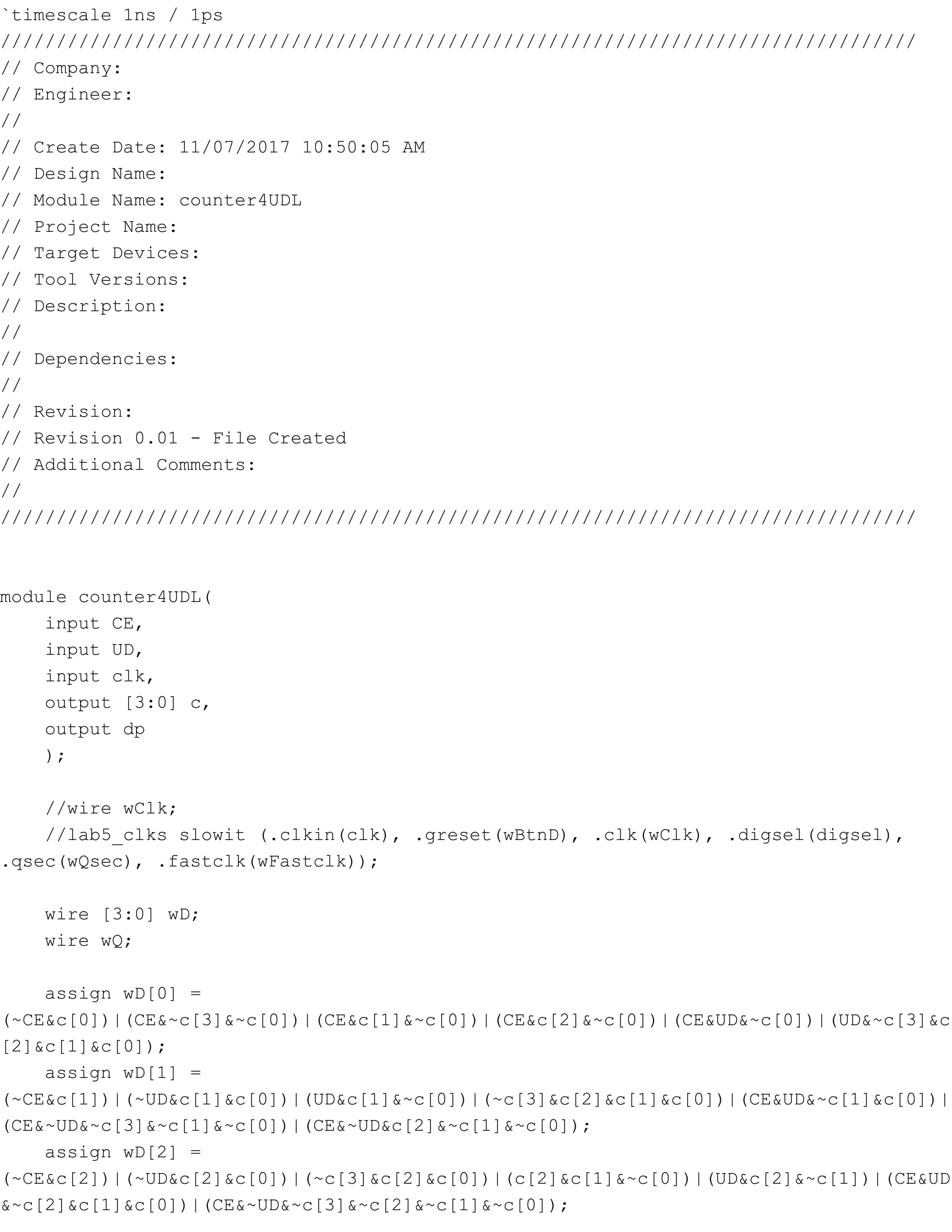
**Verilog:**

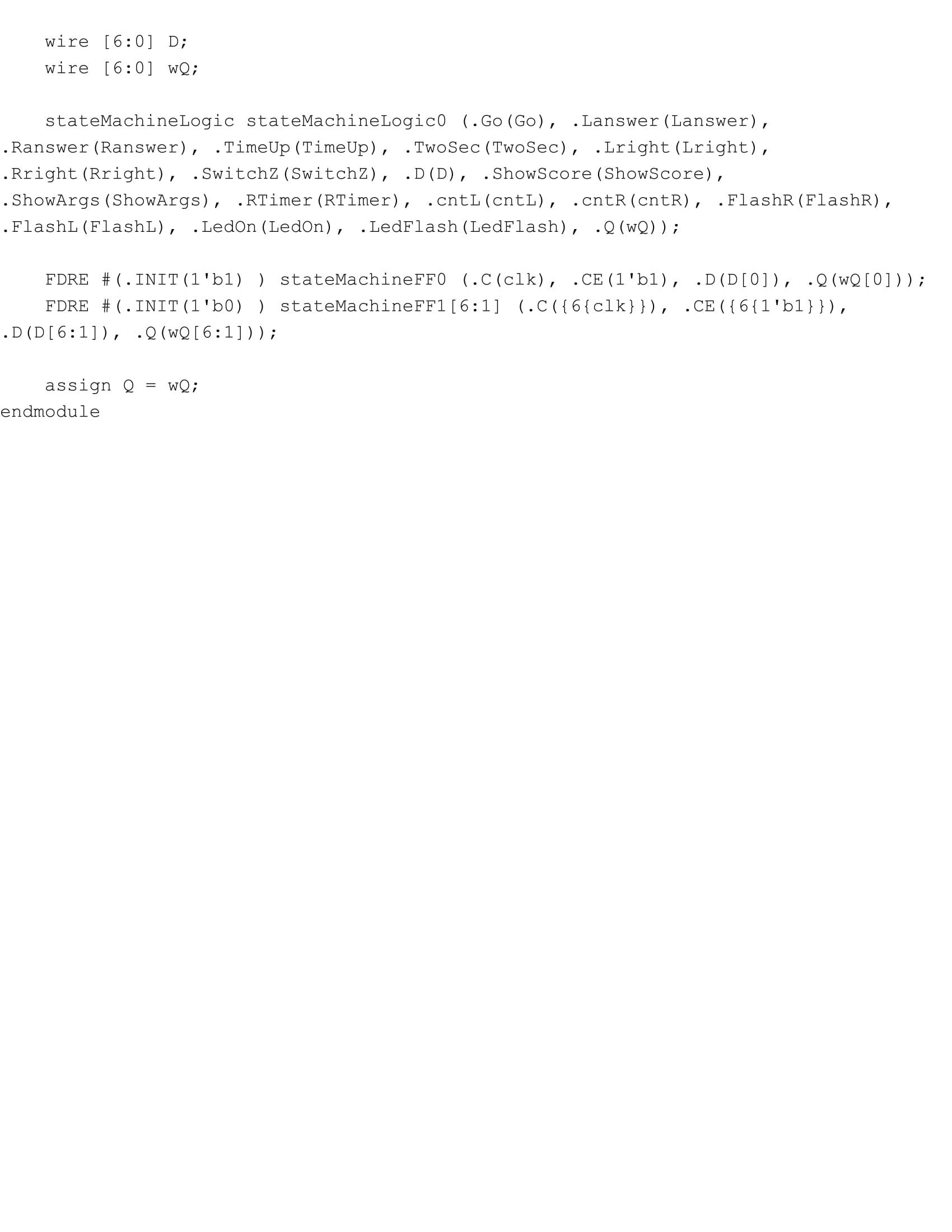
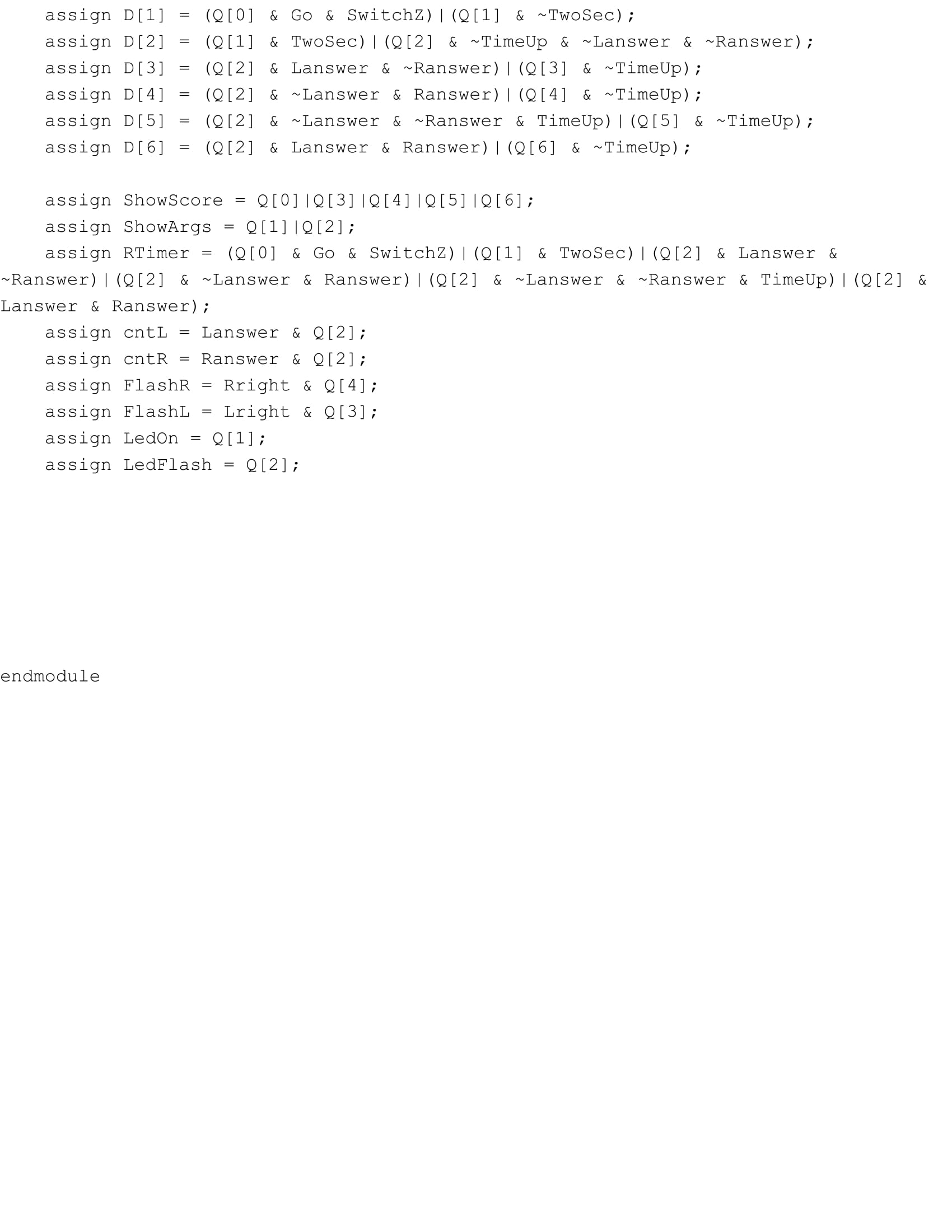
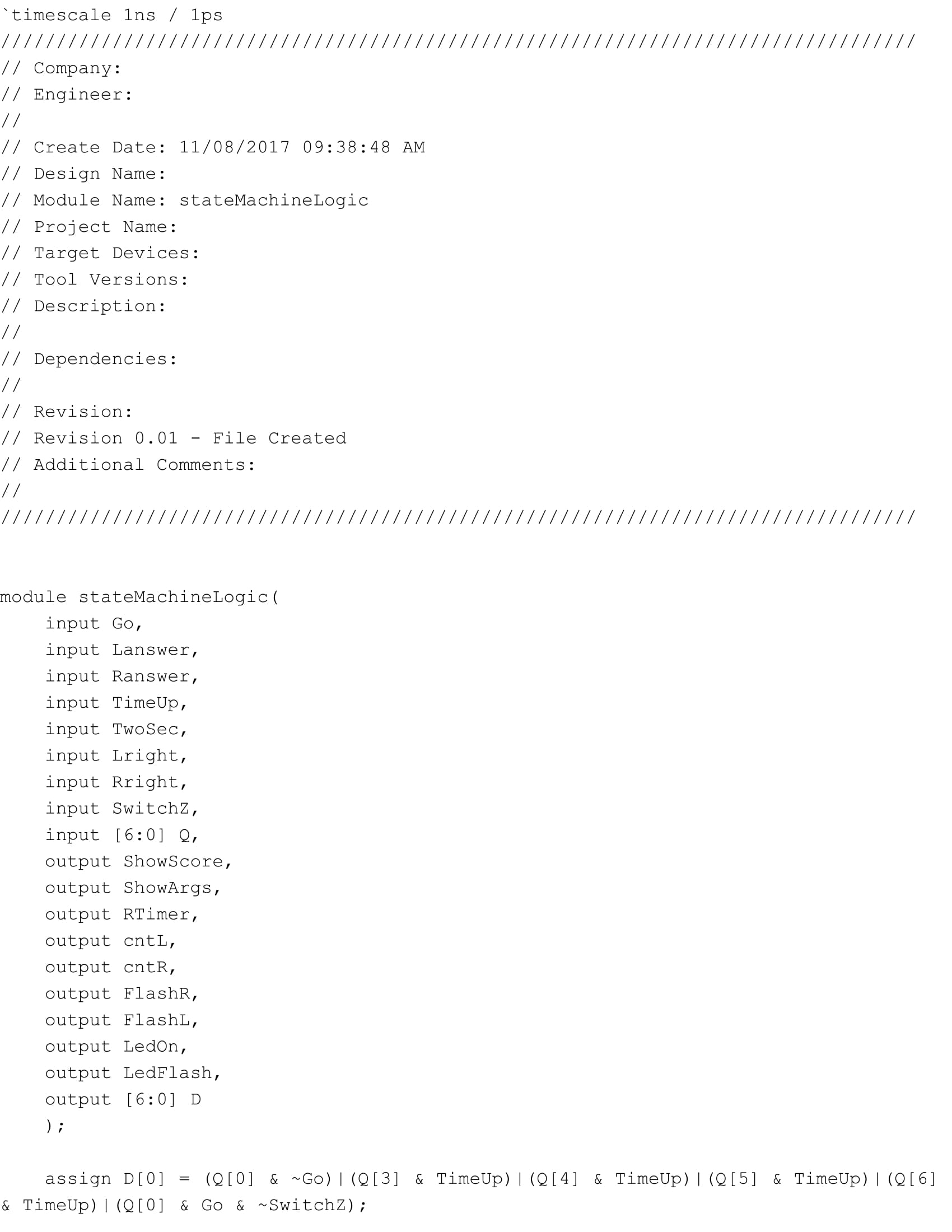
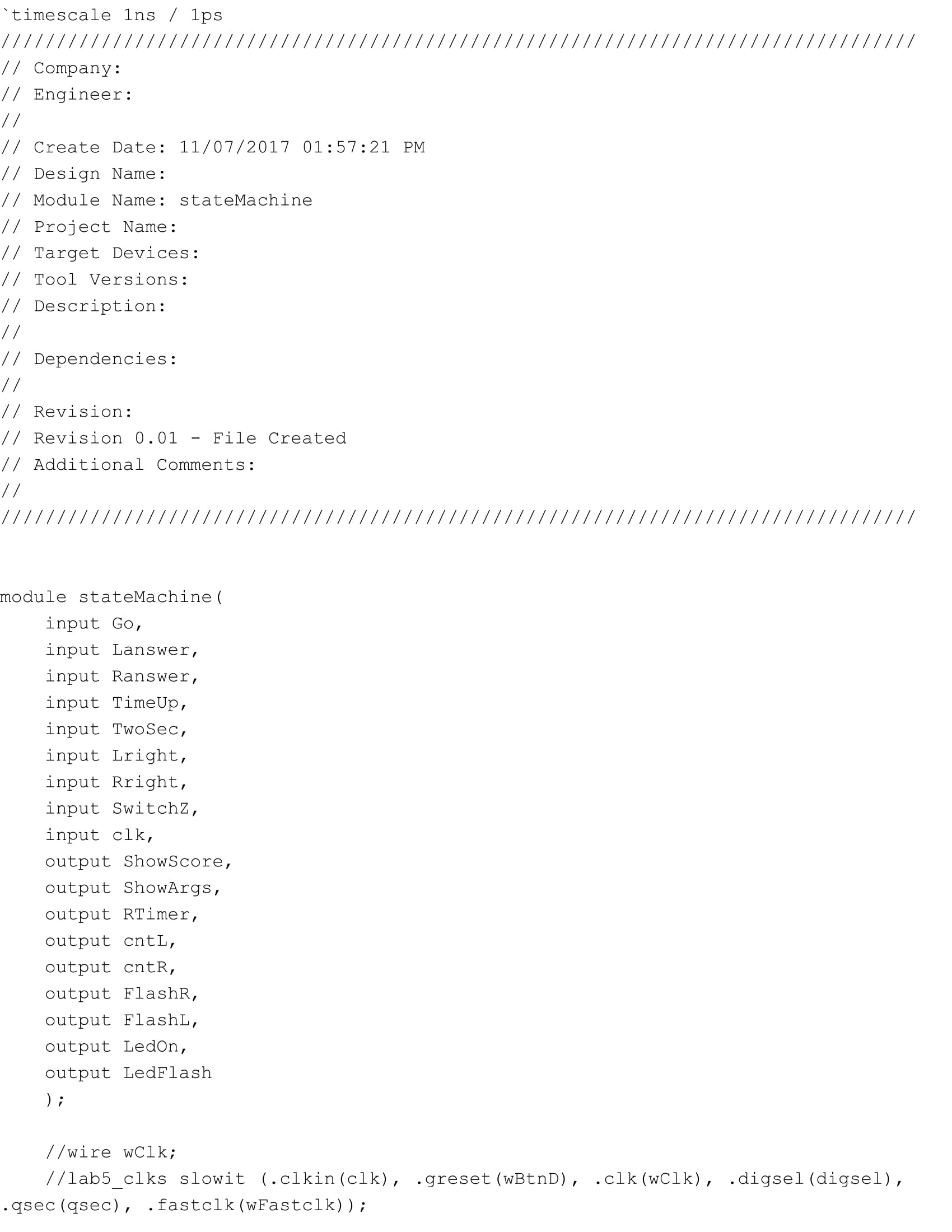
**Random Number Generator:**

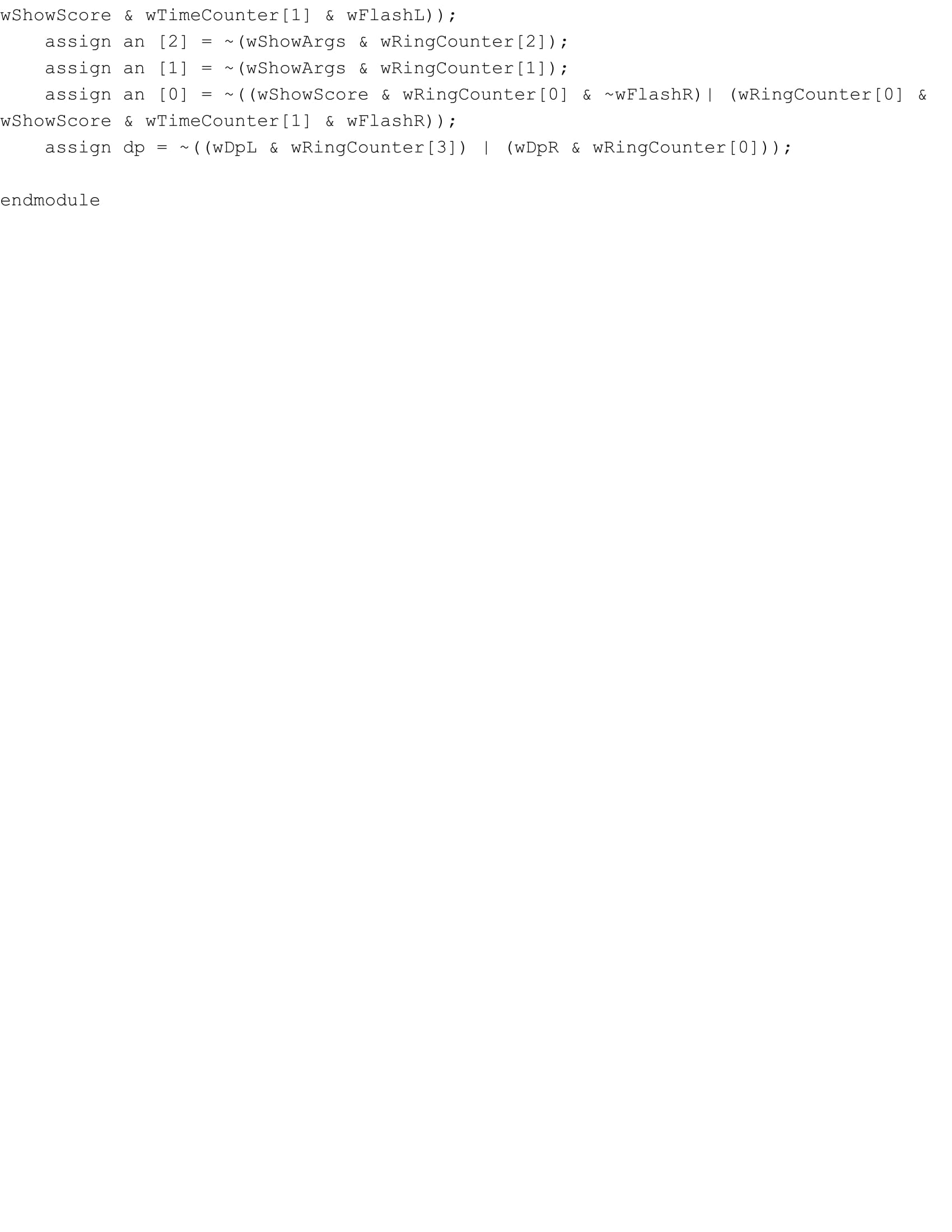
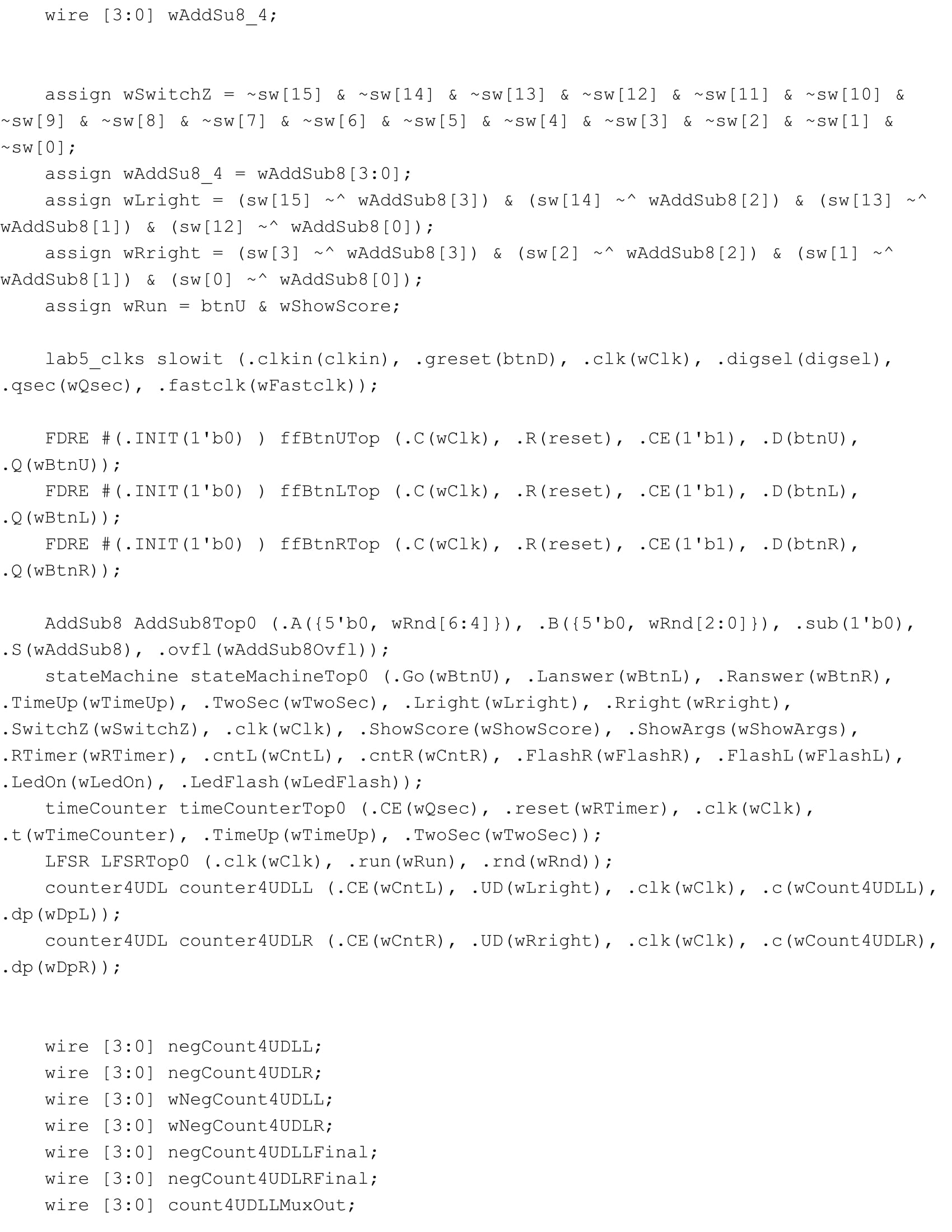
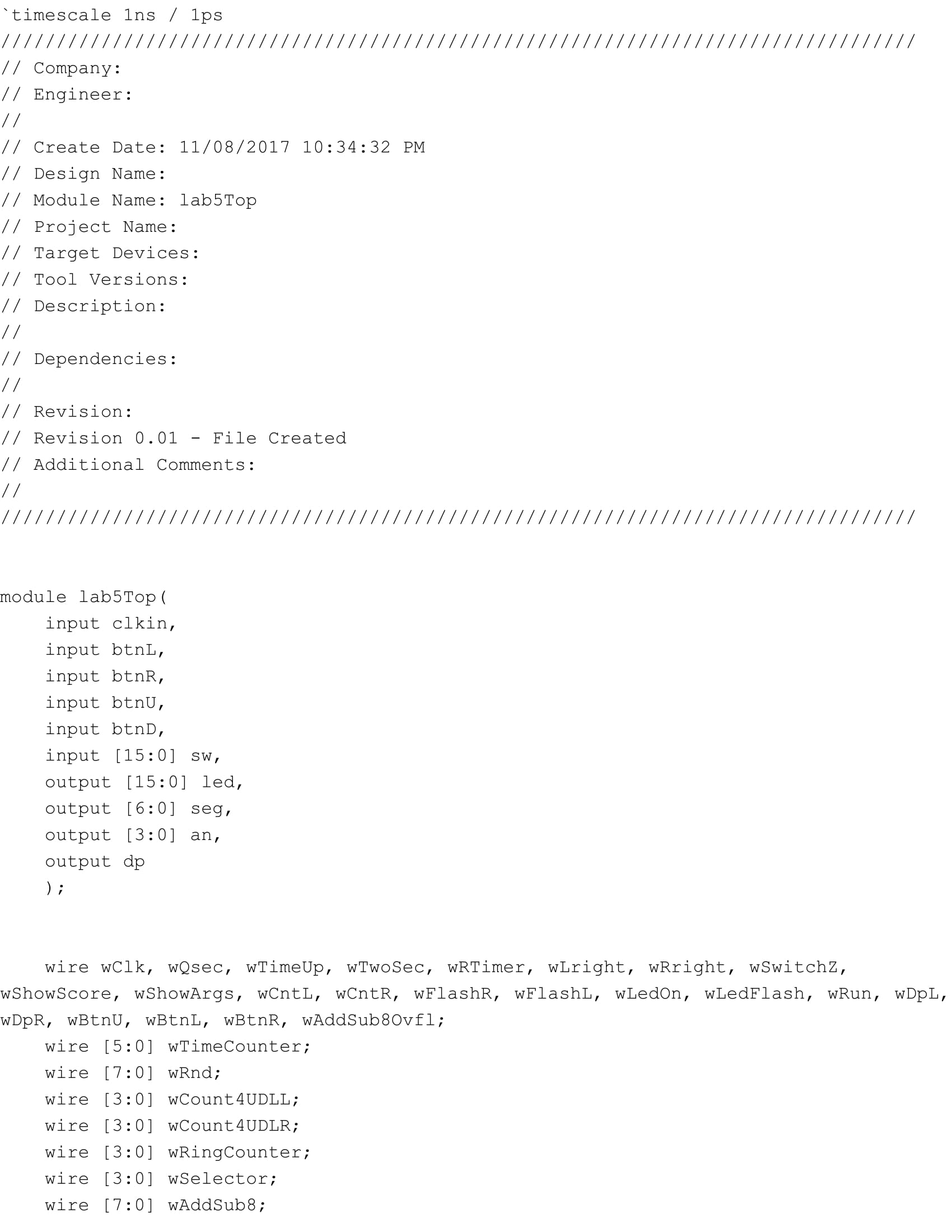
****

**Time Counter: **

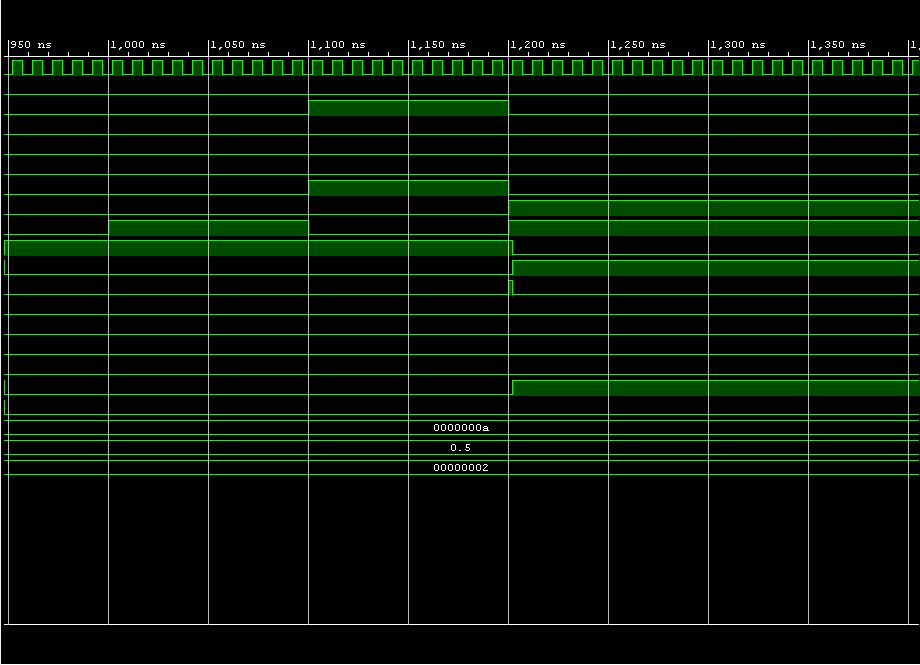
**Score Keeping and Display:**

****

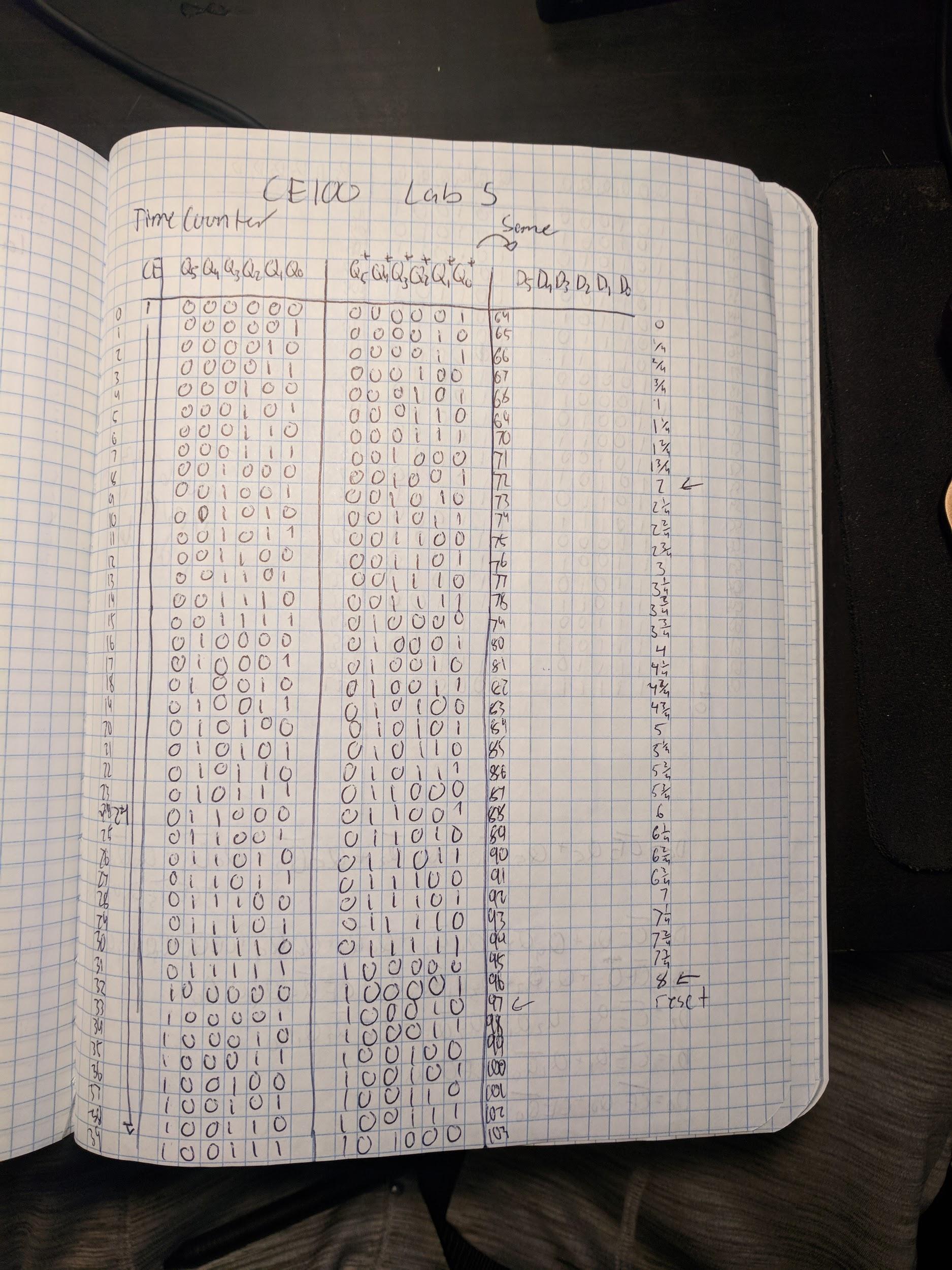
**State Machine Top & Logic:** 

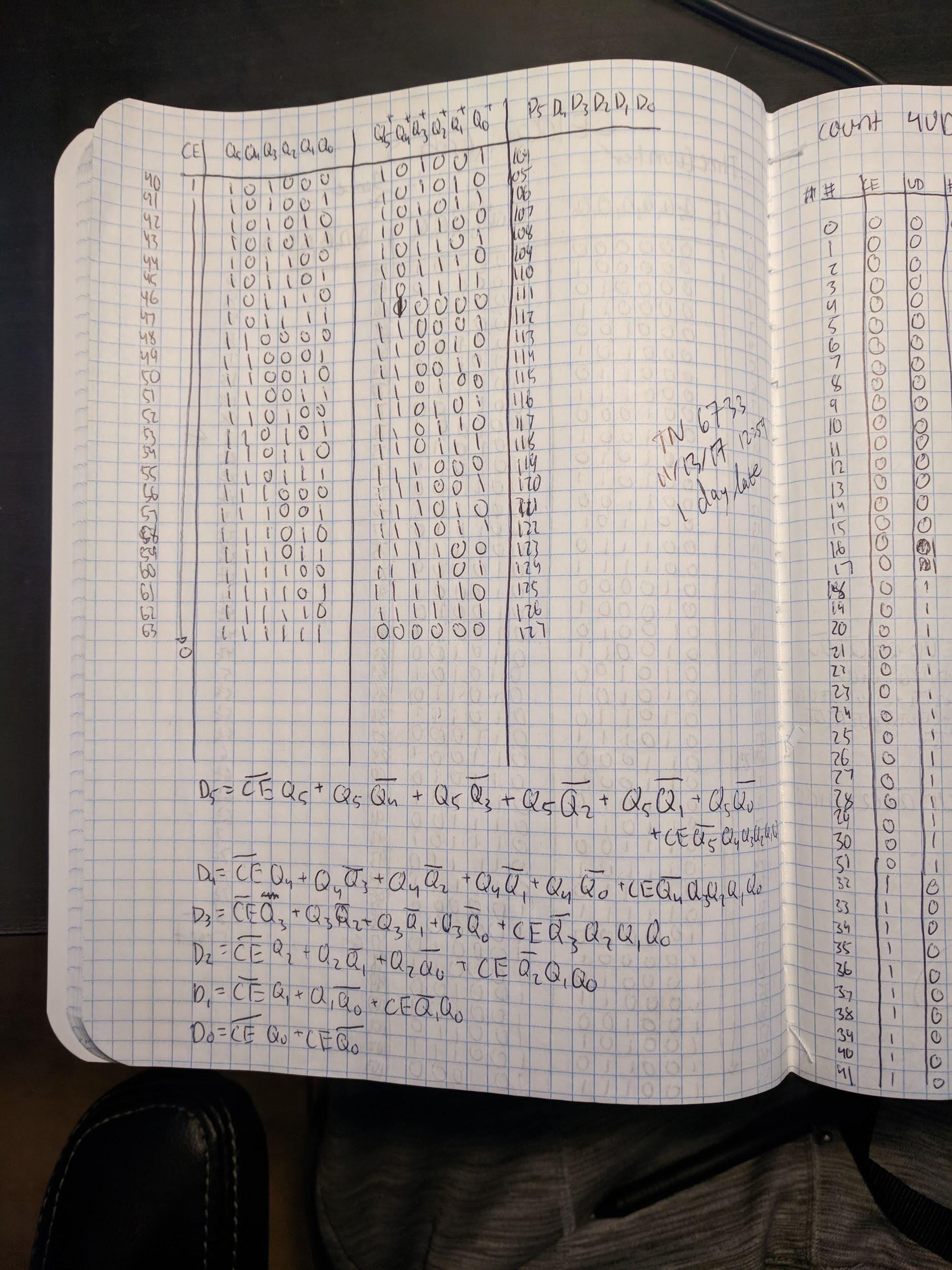
**Top Level:**

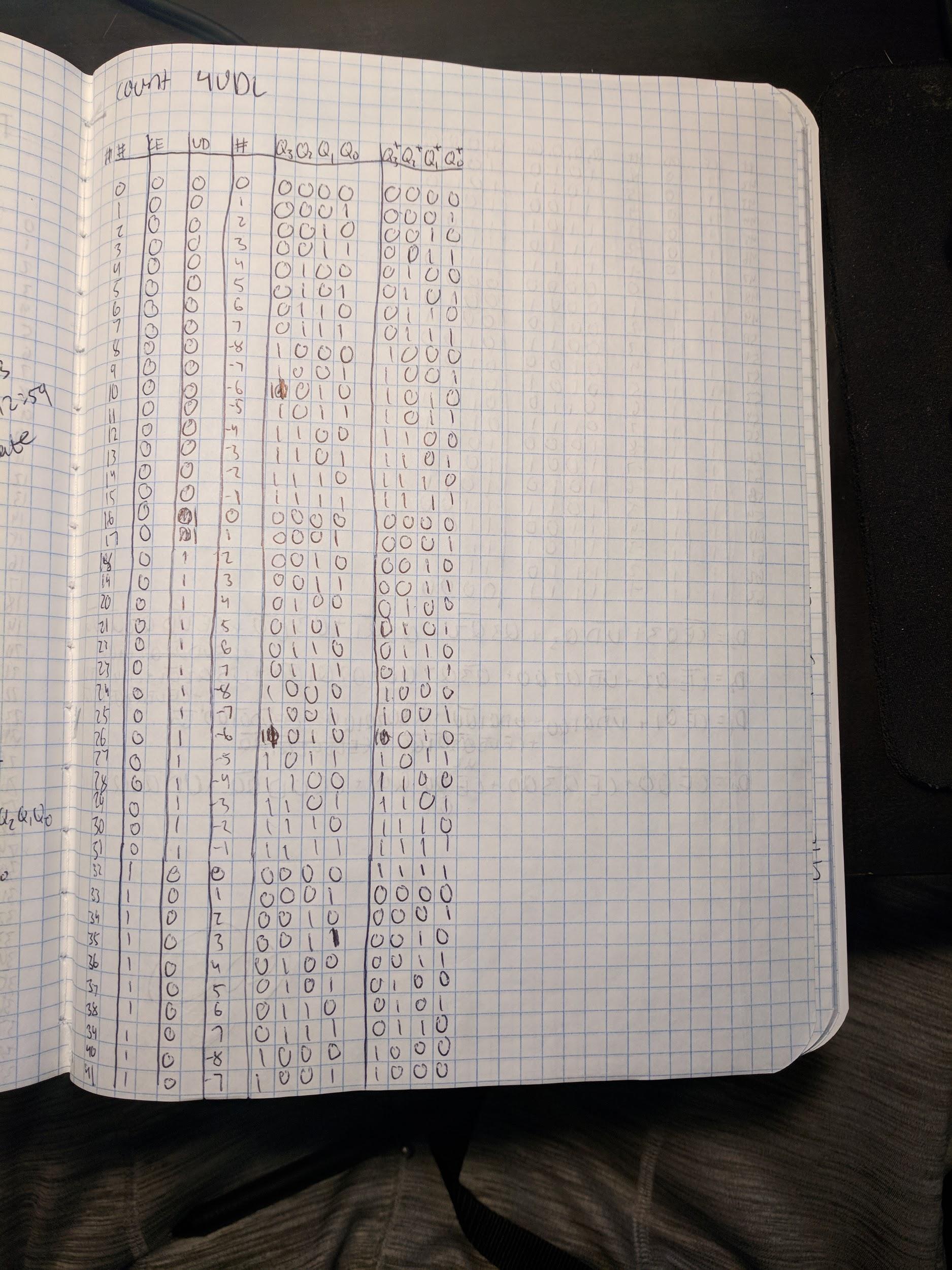
**Simulation Waveform:**

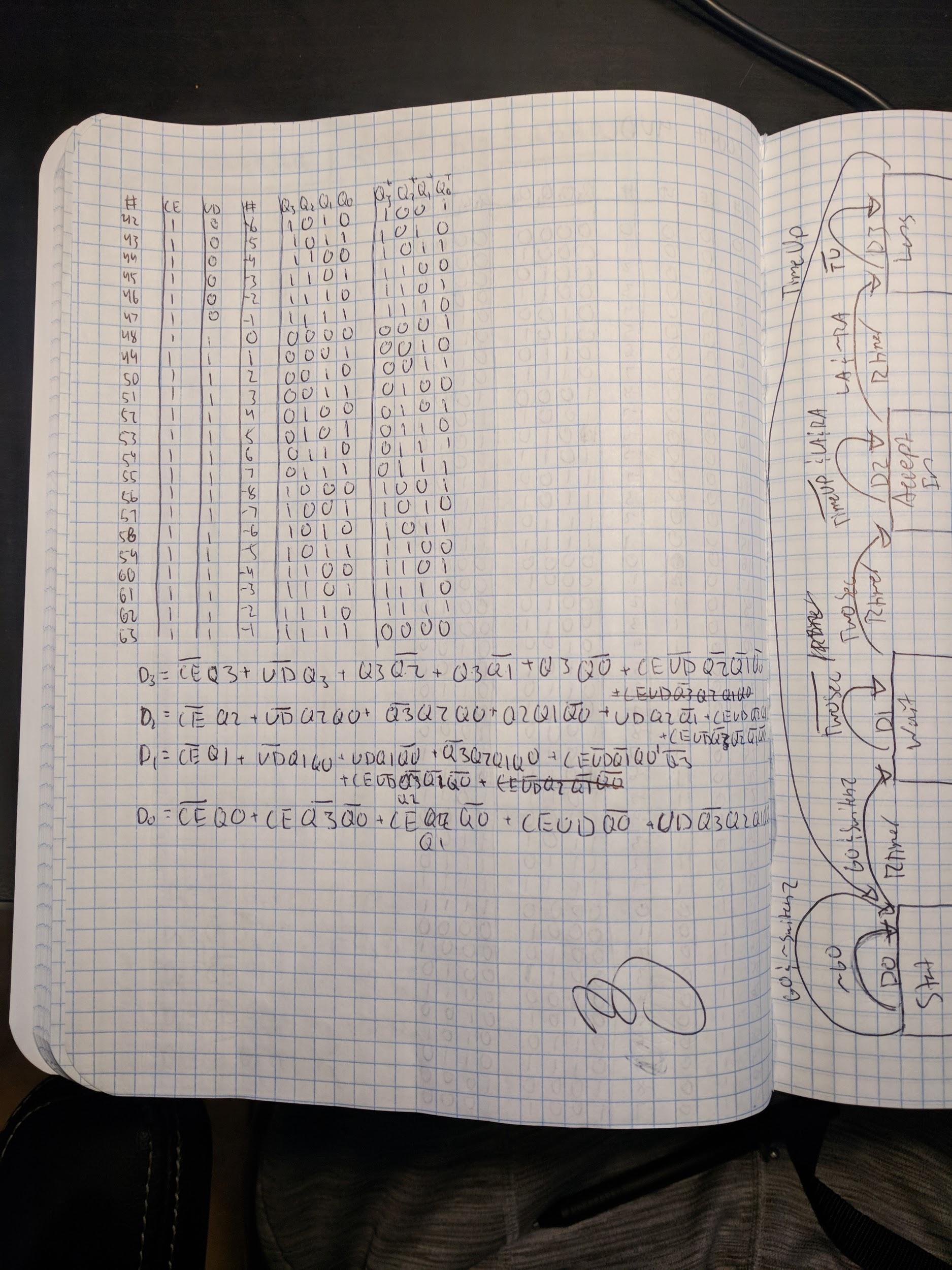
****

**Notebook Pages:**

****

****

****

****