CSE 2441 – Introduction to Digital Logic
Fall 2012
Term Design Project
Belachew Haile-Mariam

- **Introduction**

  - Overview of the project

    The term design project for this semester was to design a lock/display controller for a digital lock. It should display “open” on a series of seven-segment boards if the code sequence **X1X0:** 00-10-01 is entered, “deny” if any other code sequence is entered, and “lock” if the digital lock is still in its locked state.

    The design requirements specified the construction and use of a finite state machine controller that:

    1. Satisfied the above input and output definitions
    2. Uses the minimum number of flip-flops and combinational logic elements, and
    3. Does not produce any unwanted or undefined outputs.

    The final implementation of this project was to be designed using the Quartus II software, tested thoroughly using the QSim Simulation Software, and implemented using the Altera DE-1 board.

  - Extra Features

    As an extra feature I decided to implement in this project, I decided that my seven-segment display panels should initially be turned off, only displaying input to the user once a 2 bit sequence has been entered, and returning to the off state afterwards. The current state of the lock (lock, deny, open) would only be available to the user while he/she is pressing the enter button.

    Not only does this approach demonstrate a different way to output information to the user, but it is also more energy efficient than the alternative—leaving the state of the lock constantly displayed on the seven segment display panels. While not a major factor when powering such a device through a USB cable or from a wall outlet, this consideration could come particularly in handy if powering this device through other, less abundant, means (batteries, solar-power, etc.)

  - Overview of the Results

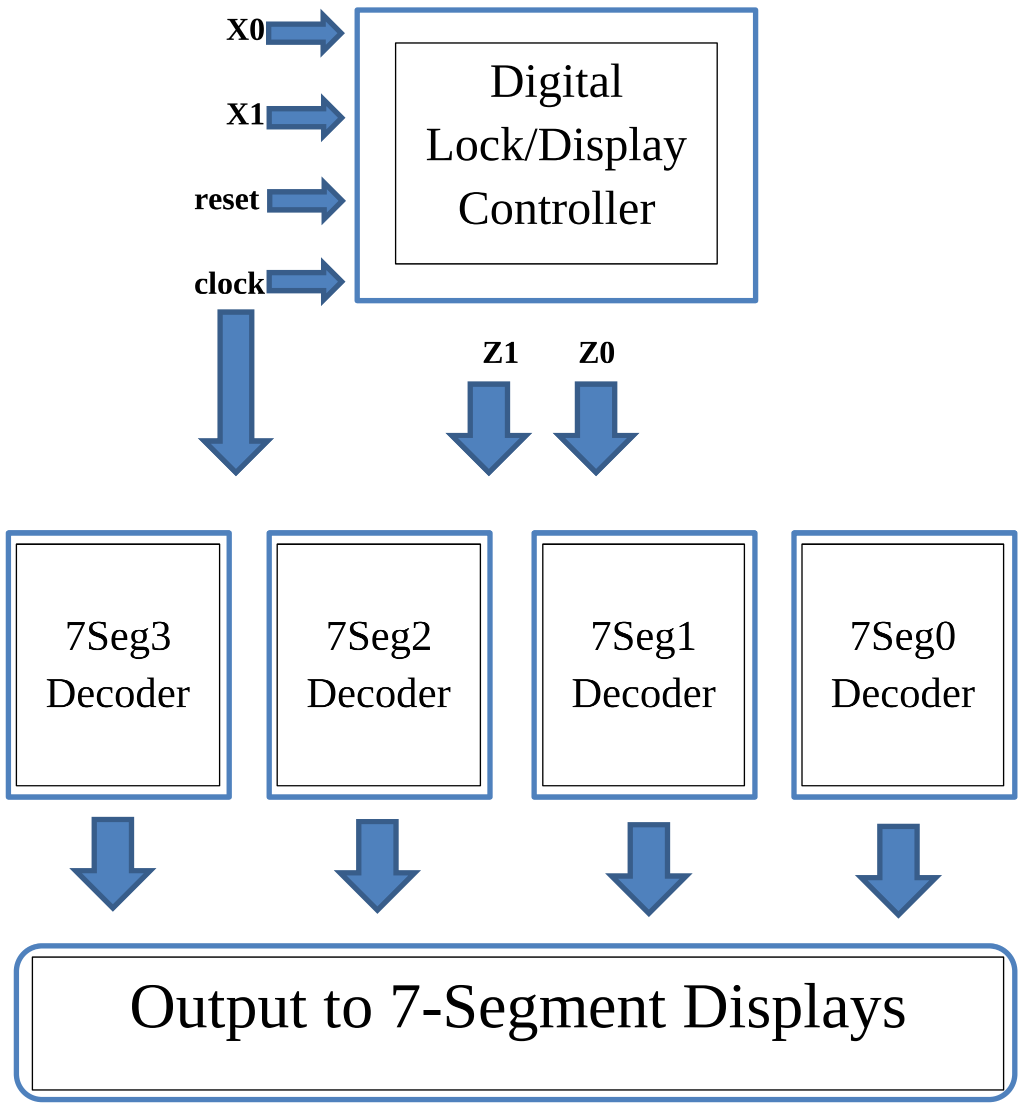
    The results of this project came out as expected. Its design and implementation all follow the required criteria, and the display shows the right outputs to the user while in the correct states. Also, as stated above, the screen only shows the output to the user once the input has been entered, and shuts of as soon as the button to enter the 2 bit sequence has been depressed.

- Unresolved Problems and Future Plans

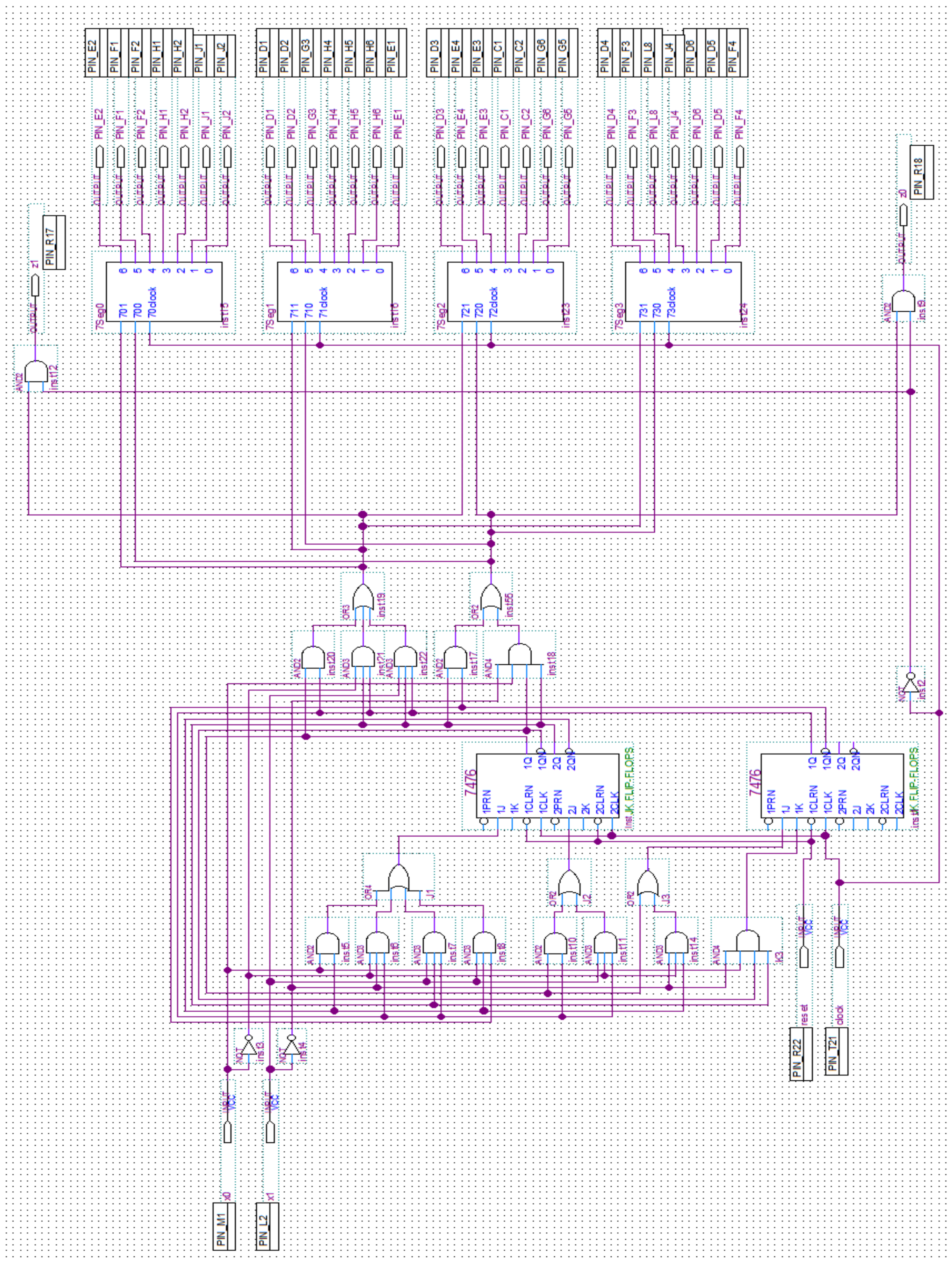
  There were no unresolved problems in the construction and implementation of my design. Although, there was a small issue in which the state of the lock would switch to deny during the second 2 bit input. However, this problem was remedied right before our class project demonstration, and all outputs and states are now in working order.

- **Final Design Description**

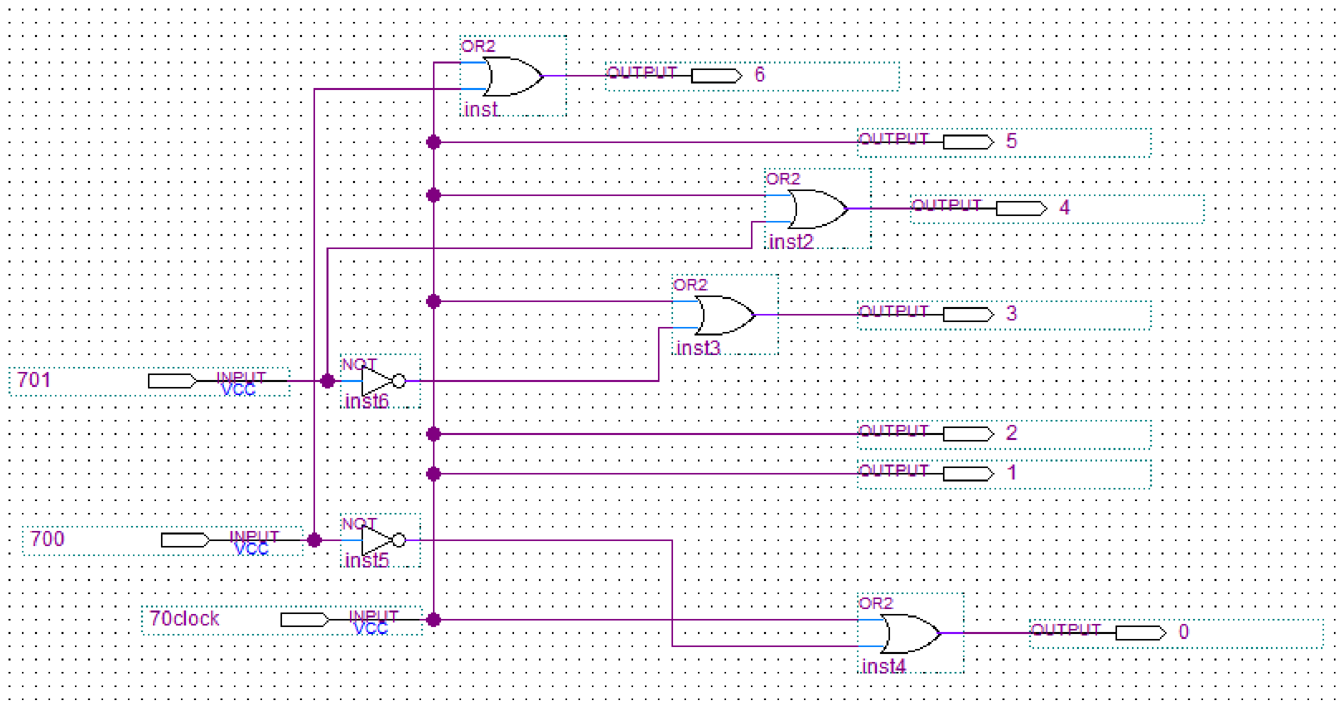
  - Block Diagram



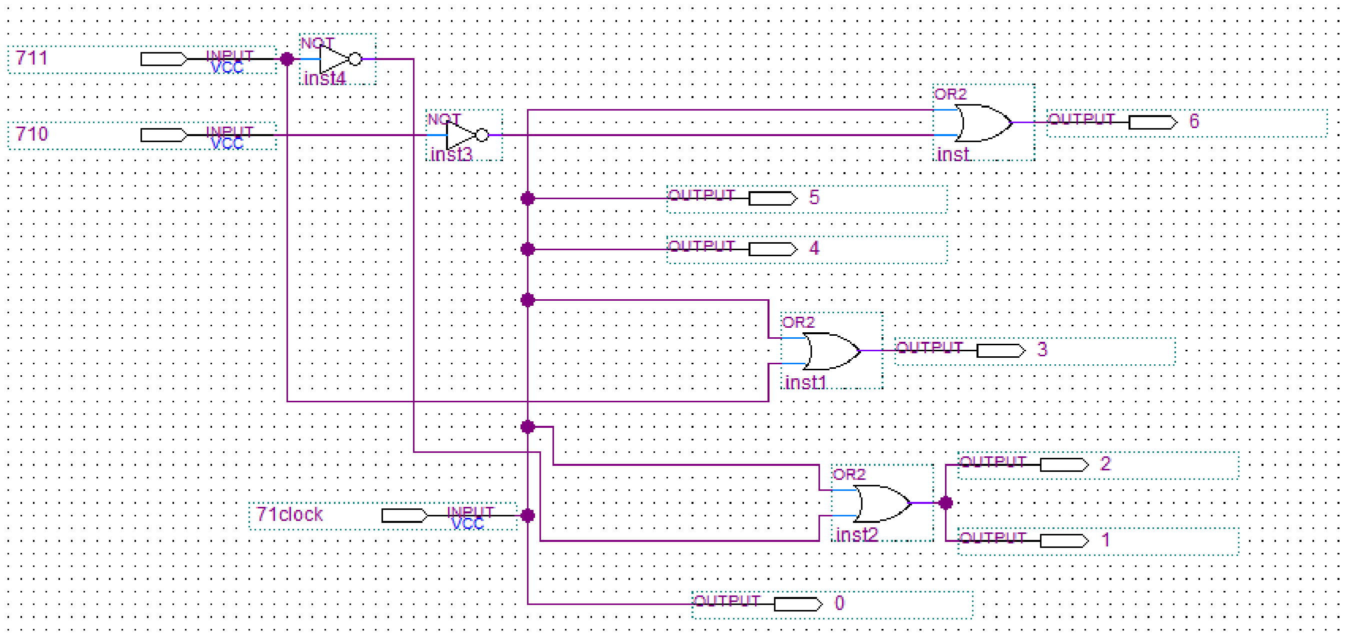
- Circuit schematic diagram from Quartus II



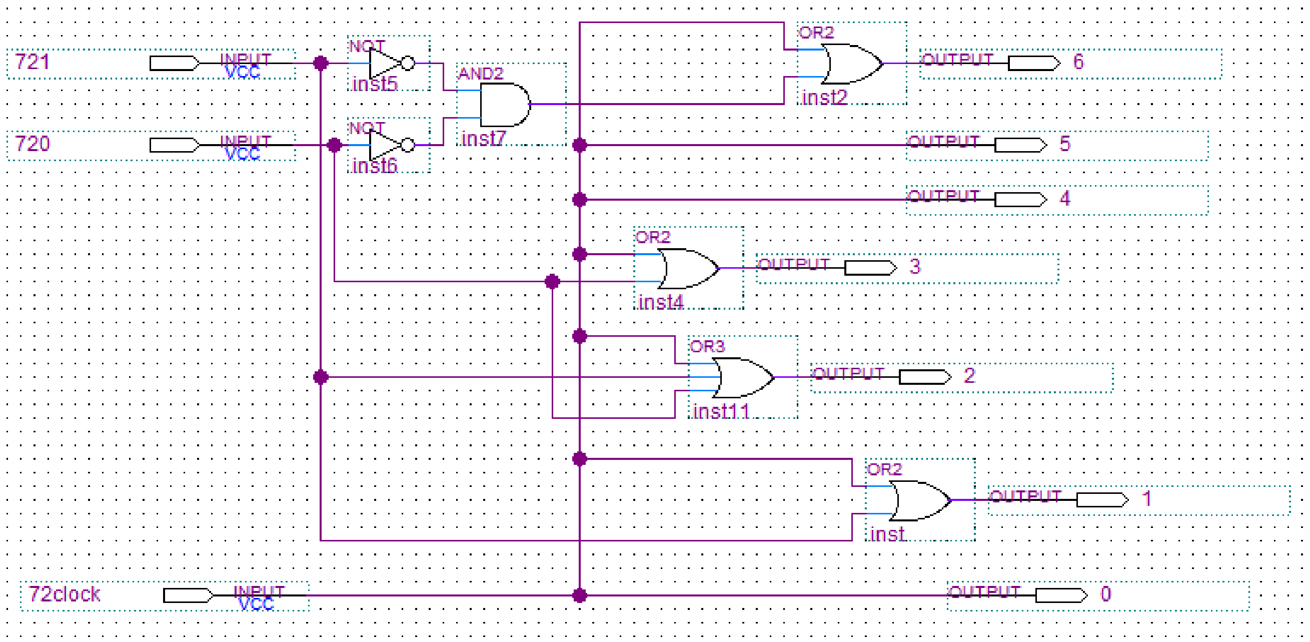
Lock/Display Controller (29 logical elements)



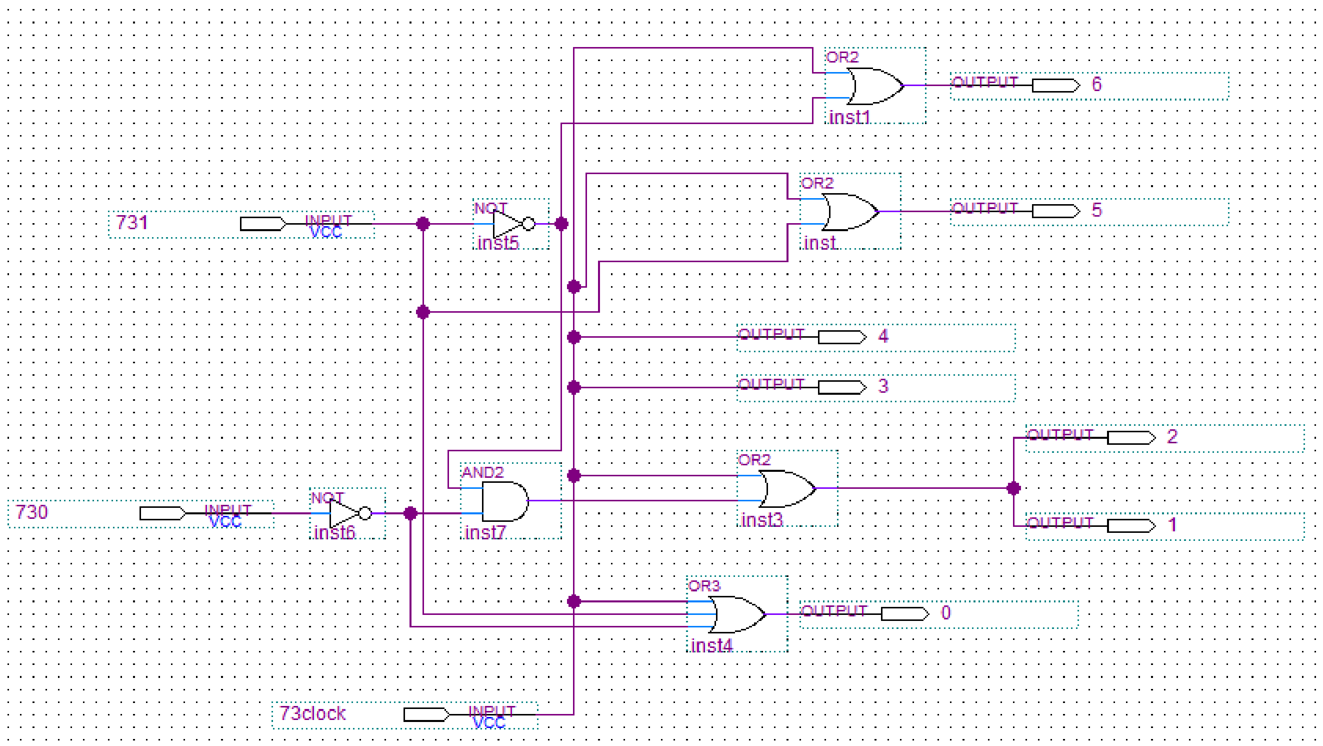
7Seg0 Decoder (6 Logical Elements)



7Seg1 Decoder (5 Logical Elements)

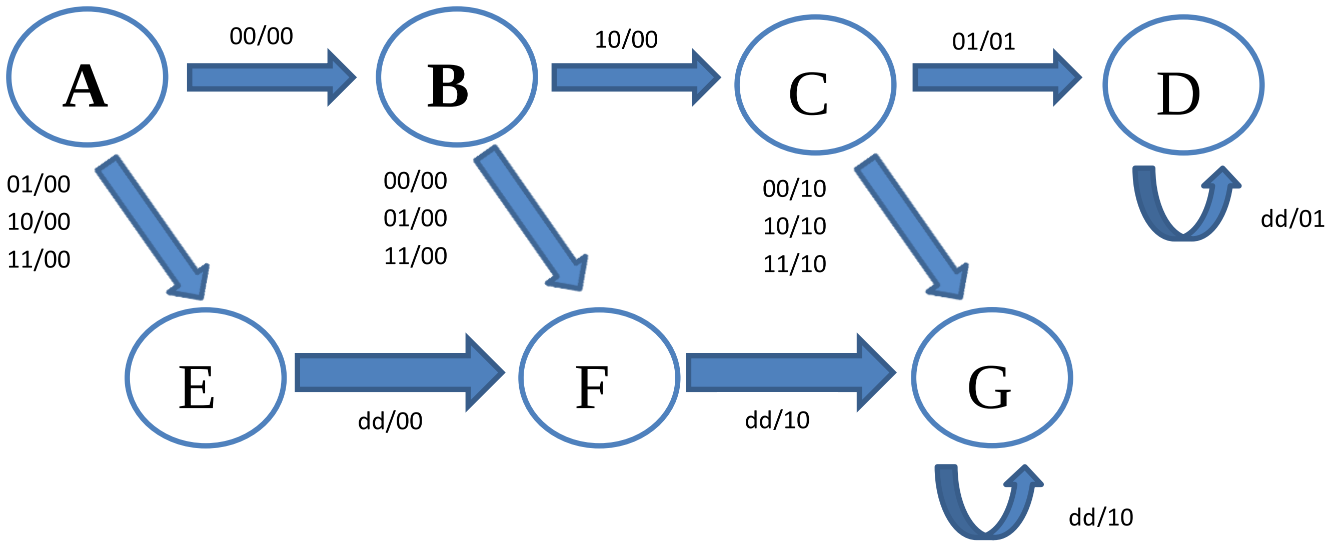


7Seg2 Decoder (7 Logical Elements)



7Seg3 Decoder (7 Logical Elements)

- State Diagram of the Controller



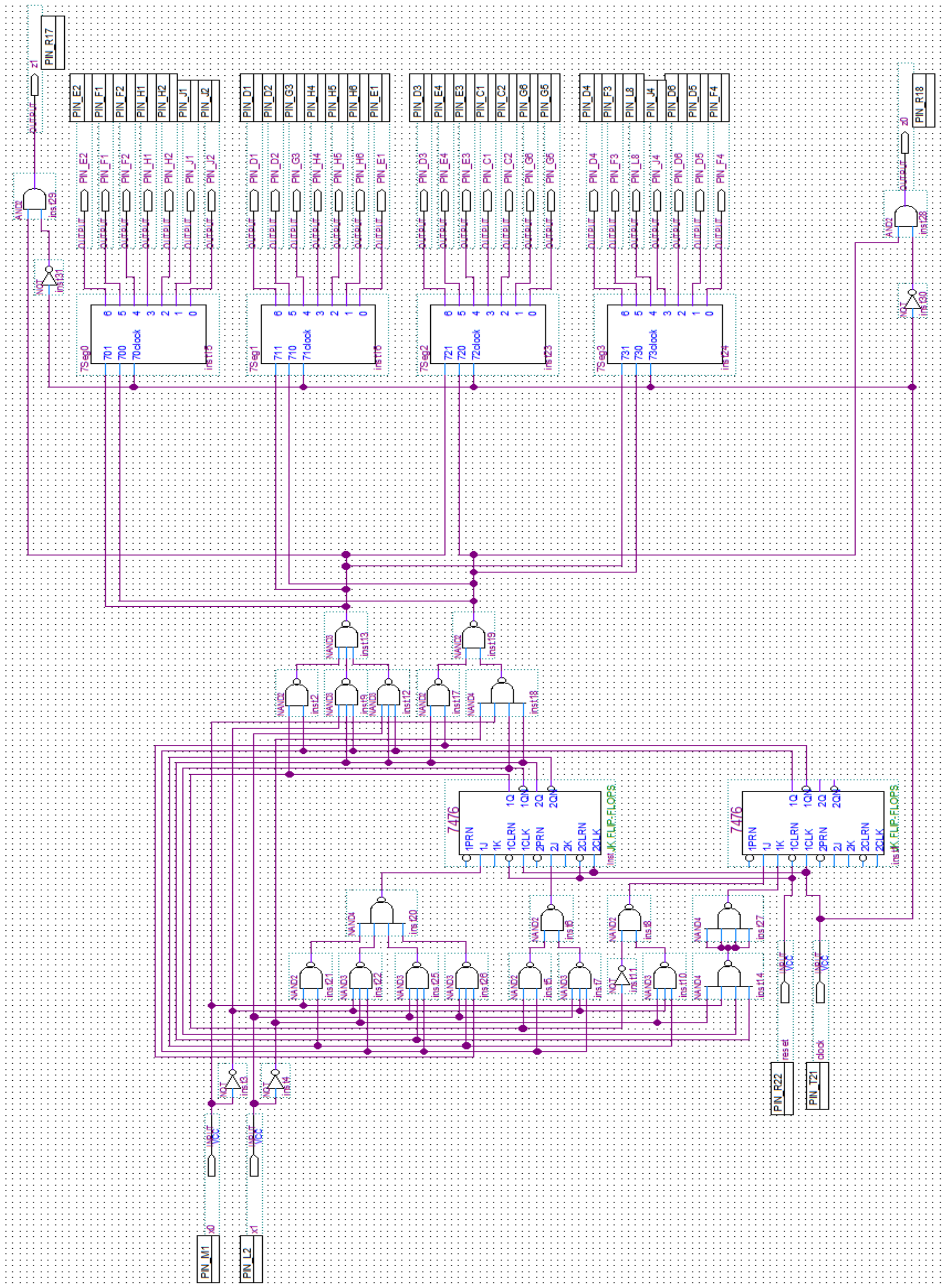
- **Alternative Design Considerations**

  - Alternatives Considered

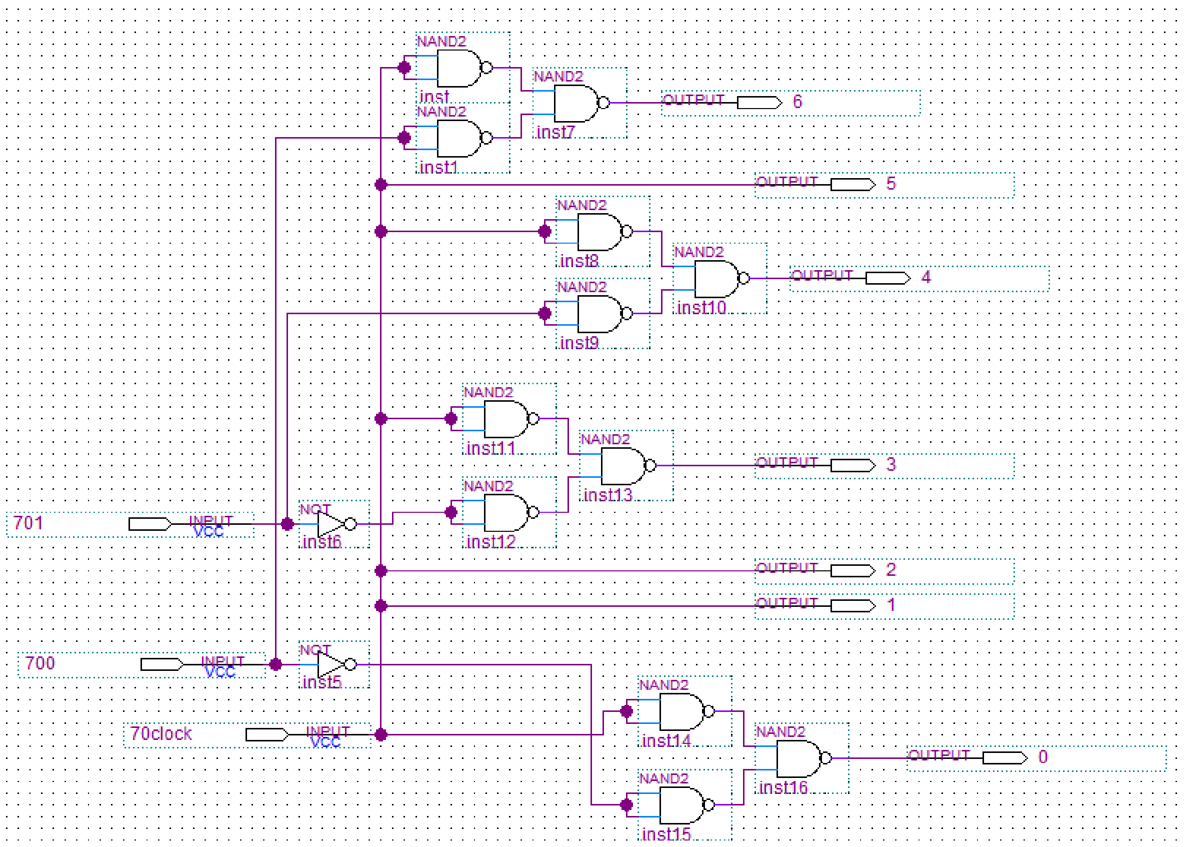
    The difference between my main design and my alternative is in the way that I chose to realize my logic functions. My alternative design is a NAND realization of my main design, as illustrated below, while my original design is realized using a variety of different logical units.

    I chose to use a NAND realization as my alternative design because NAND gates are functionally complete, meaning that they can be combined in a variety of different fashions to express all possible logical equations.

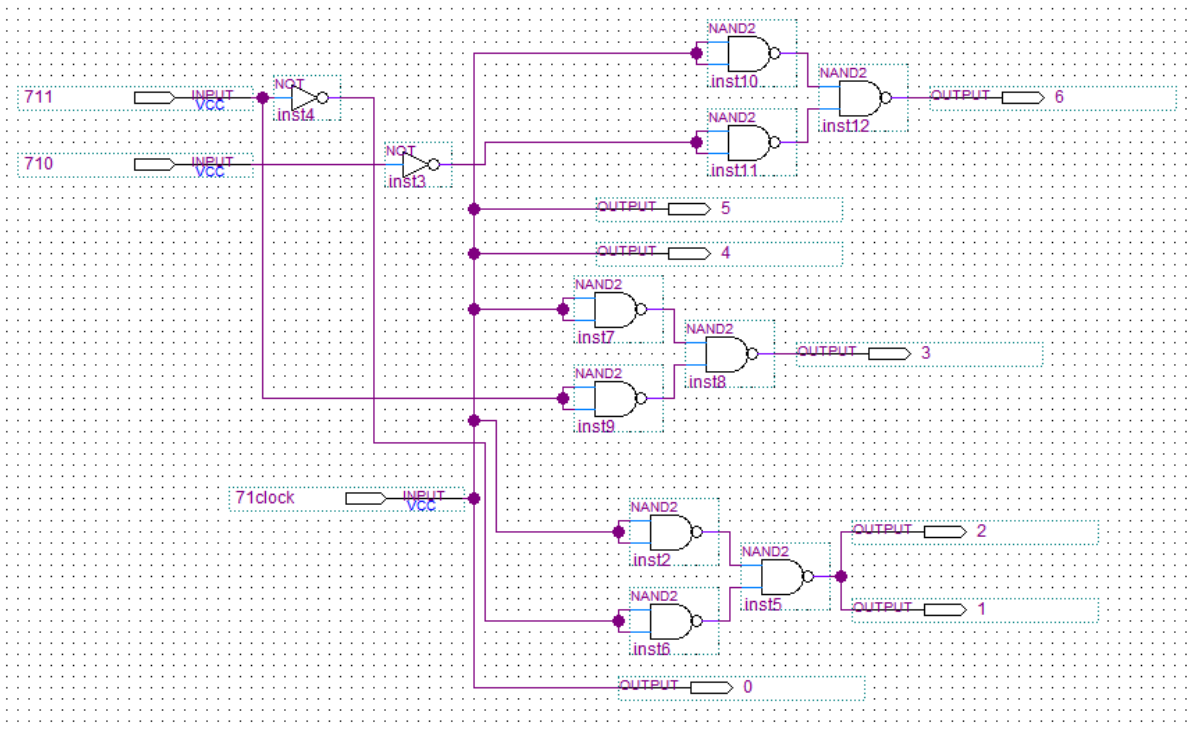
- Alternative Circuit schematic diagram from Quartus II



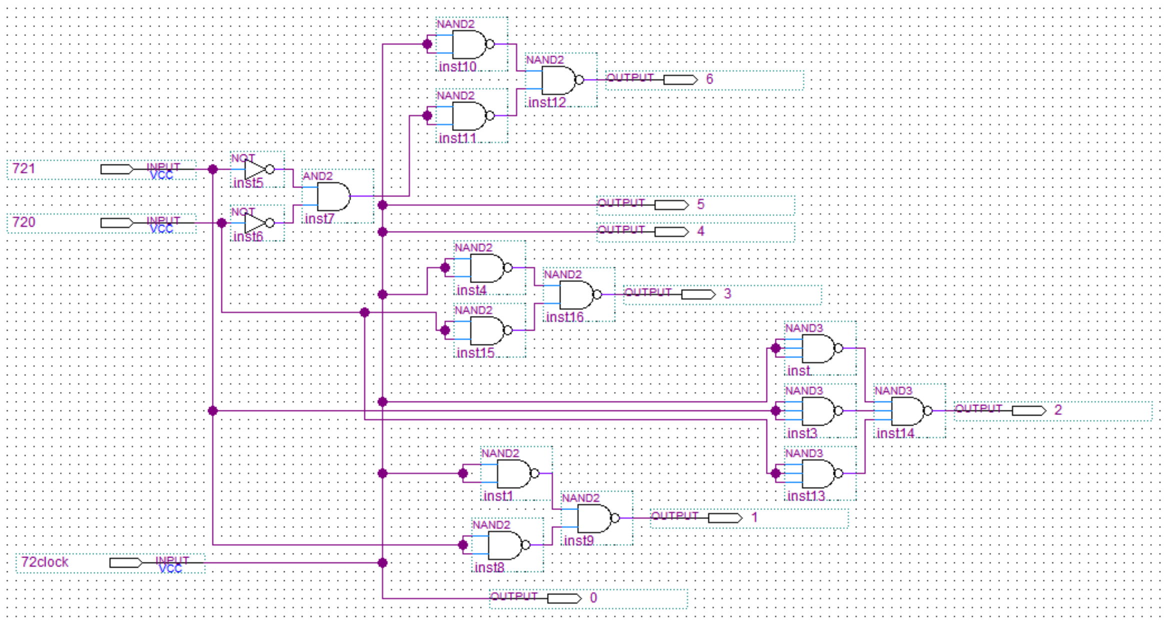
Alternative Lock/Display Controller (32 Logical Elements)



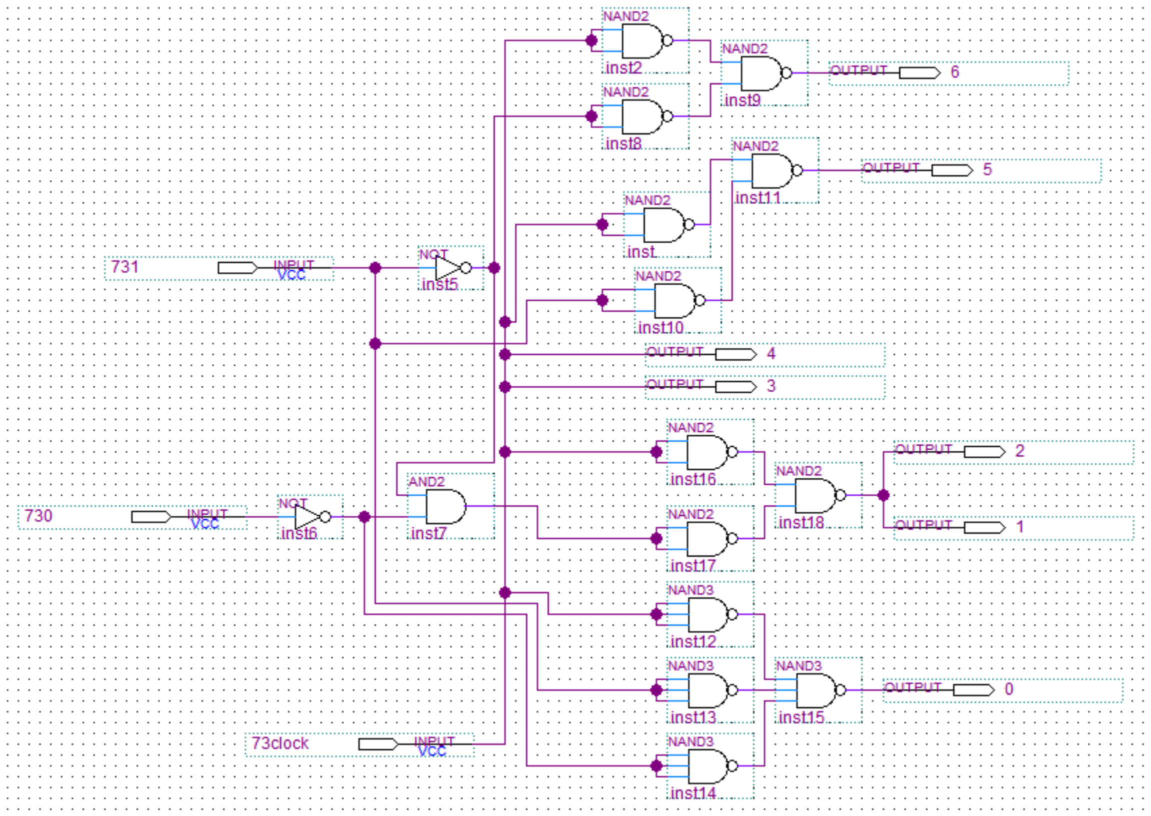
Alternative 7Seg0 Decoder (14 Logical Elements)



Alternative 7Seg1 Decoder (11 Logical Elements)



Alternative 7Seg2 Decoder (16 Logical Elements)



Alternative 7Seg3 Decoder (16 Logical Elements)

- Reasons for Selection of Final Design

  I chose the NAND realization as my final design because although not a reduction in total number of logical units (89 vs. 54), I believe that there are distinct advantages to be had by using only one type of logical element. If my design were to be mass produced to be sold, it would be easier to locate and buy NAND gates in bulk, rather than locating different specific types of combinational logic gates separately. Also, since chips usually have more than a single gate on them, only using NAND gates would help in minimizing the wasted gates that could arise from using several types of gates.

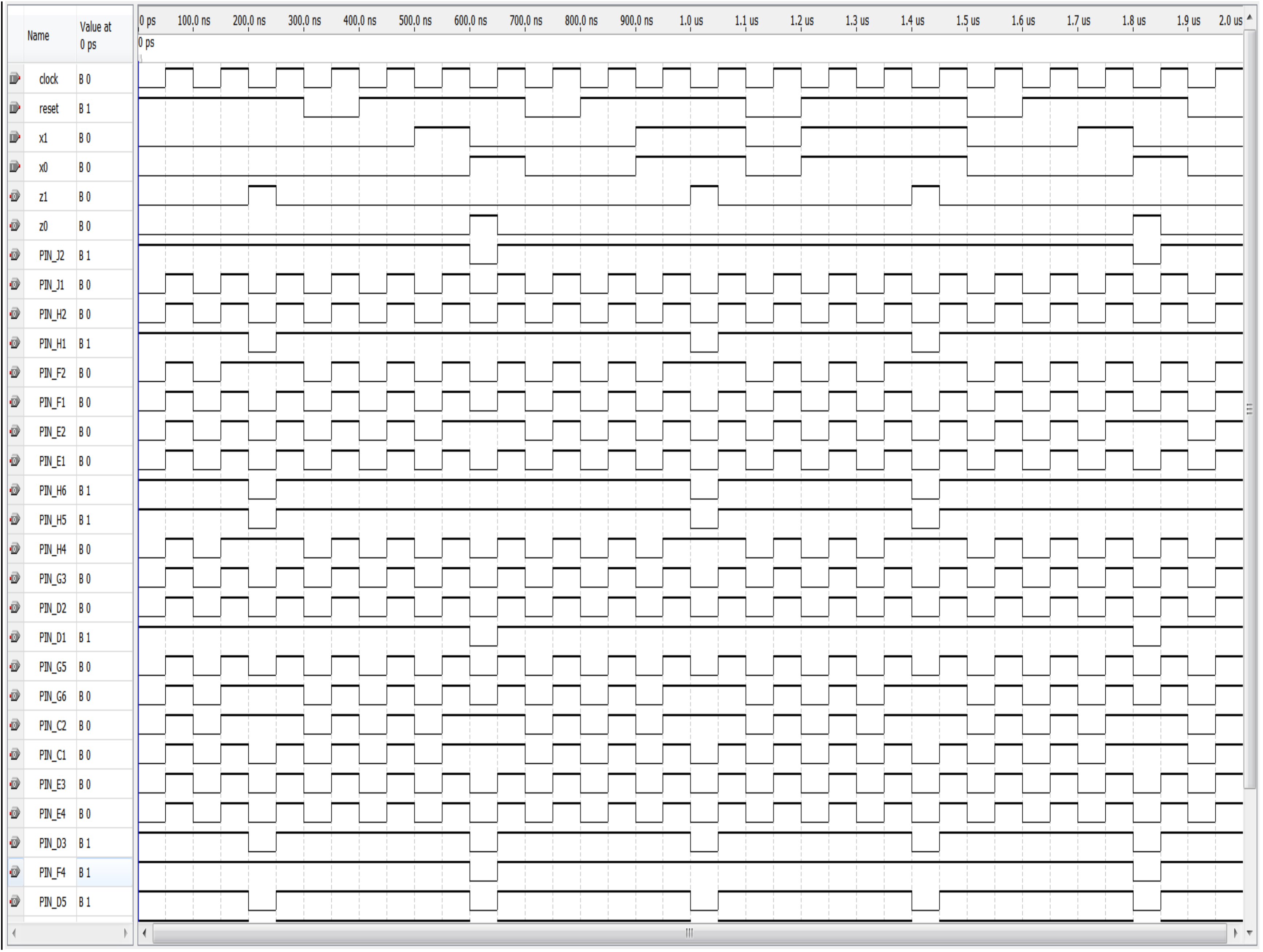
- **Test Plan**

  - Test Strategy

    Both of my designs were tested on the QSim Simulator software and on the Altera DE-1 board.

  - Simulation Results from QSim

    Because both designs are logically equivalent, the simulations results from them both should be identical. Therefore, only 1 Simulation result from the QSim software needs to be shown.



Name	Value at 0 ps
clock	B 0
reset	B 1
x1	B 0
x0	B 0
z1	B 0
z0	B 0
PIN_J2	B 1
PIN_J1	B 0
PIN_H2	B 0
PIN_H1	B 1
PIN_F2	B 0
PIN_F1	B 0
PIN_E2	B 0
PIN_E1	B 0
PIN_H6	B 1
PIN_H5	B 1
PIN_H4	B 0
PIN_G3	B 0
PIN_D2	B 0
PIN_D1	B 1
PIN_G5	B 0
PIN_G6	B 0
PIN_C2	B 0
PIN_C1	B 0
PIN_E3	B 0
PIN_E4	B 0
PIN_D3	B 1
PIN_F4	B 1
PIN_D5	B 1

- Test Results from DE-1 Implementation

  The DE-1 tested correctly for all of the different inputs that were tried. The seven segment displays were in working order, and displayed the correct messages to the user. The output of the circuit (Z1Z0) was also tested and verified on the DE-1 through the use of two led lights.

- **Other Considerations**

  - How many different code sequences can be applied to the lock controller specified above?

    (2^2) * (2^2) * (2^2) = 64 different code sequences

  - Describe two different approaches for changing the specifications to improve security.

    To improve the security of the lock, you could either:

    1. Change the specifications from using a 2-bit binary sequence to a 3-bit binary sequence. The result would be 512 different code sequence possibilities as opposed to 64 different code sequence possibilities

       ex. **X2X1X0**: 000-001-100

    2. Change the specifications from using a length-three sequence to using a length 4 sequence. The result would be 256 different code sequence possibilities as opposed to 64 different code sequence possibilities.

       ex. **X1X0**: 00-10-01-11

    Both of these approaches would make the lock harder to open via brute force attempts.

  - Which approach is the least costly to implement relative to the original? Explain and justify.

    It would be less costly to implement the 1st method relative to the original. This is due to the fact that it wouldn’t require the use of any additional states than were already used in the original construction. The 2nd approach would require the use of 2 more states than were used in the original construction, directly resulting in the addition of extra logic unites.