**Cover Page**

**CSCI 2272 - Computer Organization and Lab**

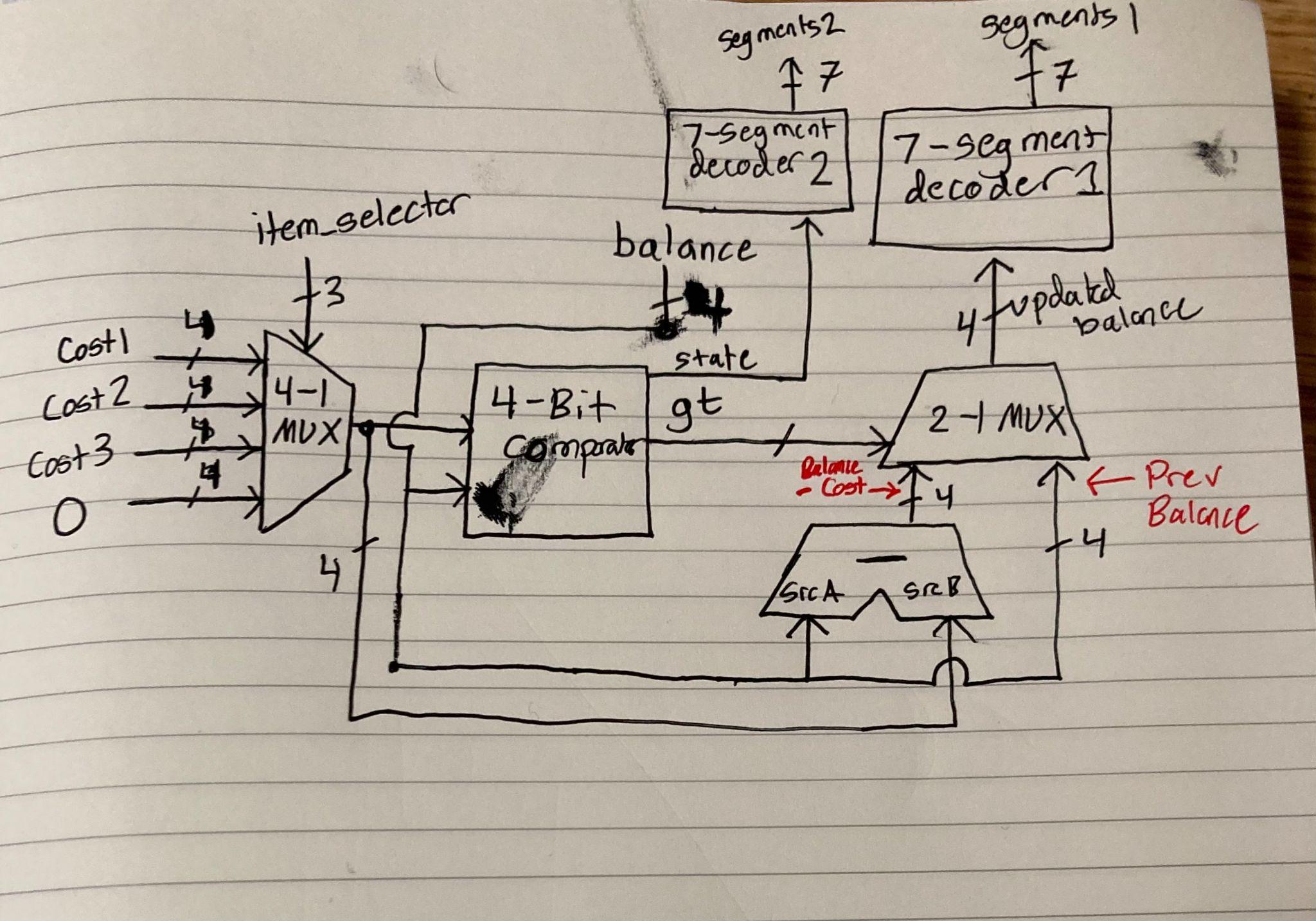
**Optional Project: Vending Machine Controller (Cyclone V Implementation)**

**Brendan Sherman: shermabk@bc.edu**

**Vending Machine Controller**

2) Project description

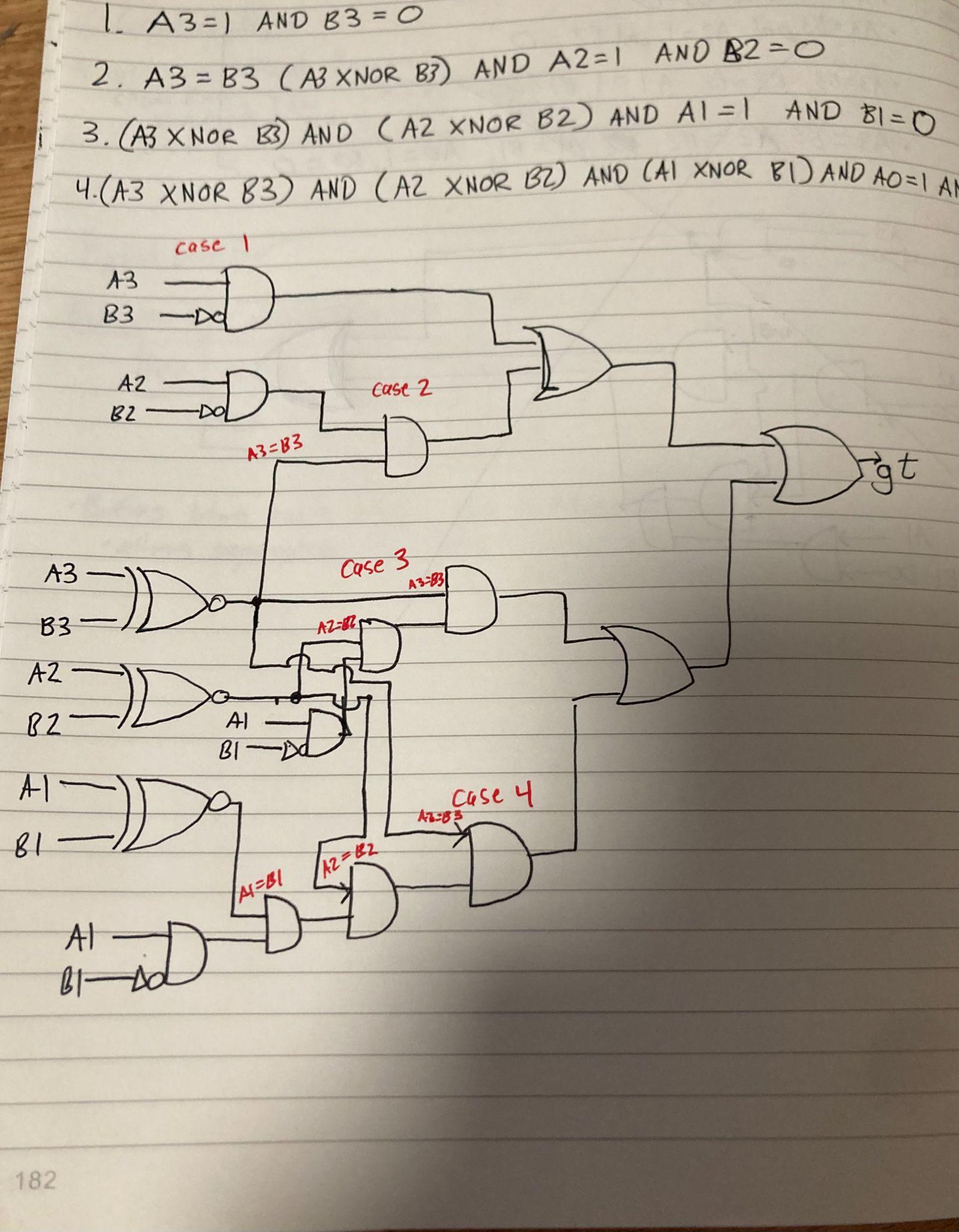
My project involved designing the schematic for a vending machine controller, implementing it in Verilog, and then exporting the Quartus project onto the Cyclone V board for demonstration, which is viewable in the video attached with this project. The controller takes a user-inputted balance, set by the switches on the board, and then displays it to the user on the seven-segment display as well as a “b” indicating balance. Also, there are 3 button inputs representing the items offered by the machine, each with its own associated cost. The user can select an item for purchase by pressing the respective button, at which point if the balance is too low an “E” for error will be displayed as well as the original balance. If the balance was sufficient, the controller calculates the change, displaying it along with a “c” indicating change.



**Figure 1:** Schematic Design for the controller

Above is the schematic for the vending machine controller, which I planned before starting verilog coding. As shown, each of the item costs are inputted into a 4-1 mux, with the chosen item being selected by item\_selector, as set by the user’s button inputs. Importantly, if no item is selected the mux outputs a cost of 0. This cost, along with the user-inputted balance, is then fed into a 4-bit comparator, which I implemented structurally in verilog. If the cost is greater than the balance, meaning an error should be indicated, the comparator outputs 1, otherwise 0. The output of the comparator is then fed into one of the seven-segment decoders, which uses this output to determine whether to display “E” or “c”. The output of the comparator is also used to update the balance, serving as the selector for a 2-1 mux that is connected to the other seven segment display. if the cost was too high the previous balance is displayed, while otherwise the change (cost-balance), calculated using a 4-bit subtractor, is displayed . The code that accomplishes all of this is viewable in the video, as you can see I was able to exactly copy the above schematic design into verilog using a combination of structural and dataflow implementations. The final module, VendingMachine, exactly reflects Figure 1 structurally.

The comparator works by examining the 4 cases, one of which must always be true, where 4 digit signal A can be greater than 4 digit signal B. These cases are when the leftmost bit of A (A[3]) = 1 and the B[3] = 0, when A[3]=B[3] and A[2] = 1 and B[2] = 0, when A[3]=B[3] A[2]=B[2], A[1] = 1 and B[1] = 0, and finally when A[3]=B[3] A[2]=B[2] A[1]=B[1] A[0] = 1 and B[0] = 0. These 4 cases are then combined into an OR statement (accomplished using 3 individual OR gates), which outputs true any time one of the cases is true. Therefore, the comparator outputs 1 if A>B, 0 otherwise, making it important to determine whether the selected cost is greater than the balance for the controller. My sketch of the Comparator structure, which I used as a guideline for the structural code, is shown below.



**Figure 2: Final Design for 4-bit comparator**

3) Relevance to the course

This project helped to reinforce my understanding of several concepts that we discussed this semester. Designing the schematic, no easy task, was a very good refresher in sequential logic, as I had to employ several of the designs we talked about this semester, including MUXes, decoders, and the 4-bit subtractor to accomplish my goal of creating a vending machine controller. Also, writing the entire project in verilog was very useful towards increasing my knowledge of the language; I had to refresh my memory on structural and dataflow implementations, as well as look up new verilog concepts such as global constants (which I used for the prices). Additionally, when designing the 4-bit comparator I used a K-map to simplify the expression, then being able to generate a circuit to represent this expression, a 4-bit comparator. The design of the 4-bit comparator was one of the hardest parts of this project, but in doing so I was able to practice boolean logic and more realize its practical use, without simplification the circuit would have been much too complex to draw. Another challenge was creating custom outputs for the seven-segment display, namely the “E”, “c”, and “b”, state outputs. I had to restudy the seven-segment display and decoder to implement these. Finally, I had never loaded a project onto the board without predefined lab instructions, so figuring out pin assignments as well as looking up ones that we hadn’t used in the course (e.g. for a second seven-segment display) was a challenge. Therefore, new information I learned from this project included verilog concepts, implementation of a 4-bit comparator, and the overall process of planning and implementing a digital circuit.

4) **A video link:** [**https://youtu.be/jyxaF6LsS3g**](https://youtu.be/jyxaF6LsS3g)