EE 648 – VLSI Design

Binary Coded Hexadecimal to Seven-Segment Display Converter

Ryker Dial

Cody Gossel

Zach Krehlik

March 10, 2017

Contents

1 Introduction

The objective of this project is to design and fabricate an integrated circuit that takes four input bits representing a hexadecimal number and displays that number on a seven-segment display. Such a circuit will be useful for a microcontroller because it reduces the number of pins required to use the display from seven to four, freeing up GPIO pins for other tasks.

2 Top Level Design

To facilitate the design of this circuit, the logic for each segment is separated into its own module. This converter is being designed for an active-low seven-segment display, so the output of each module will be used to switch a FET that pulls the segment to ground when switched on, turning on the segment. The top-level design for the converter circuit with these modules included is shown in Figure 1.

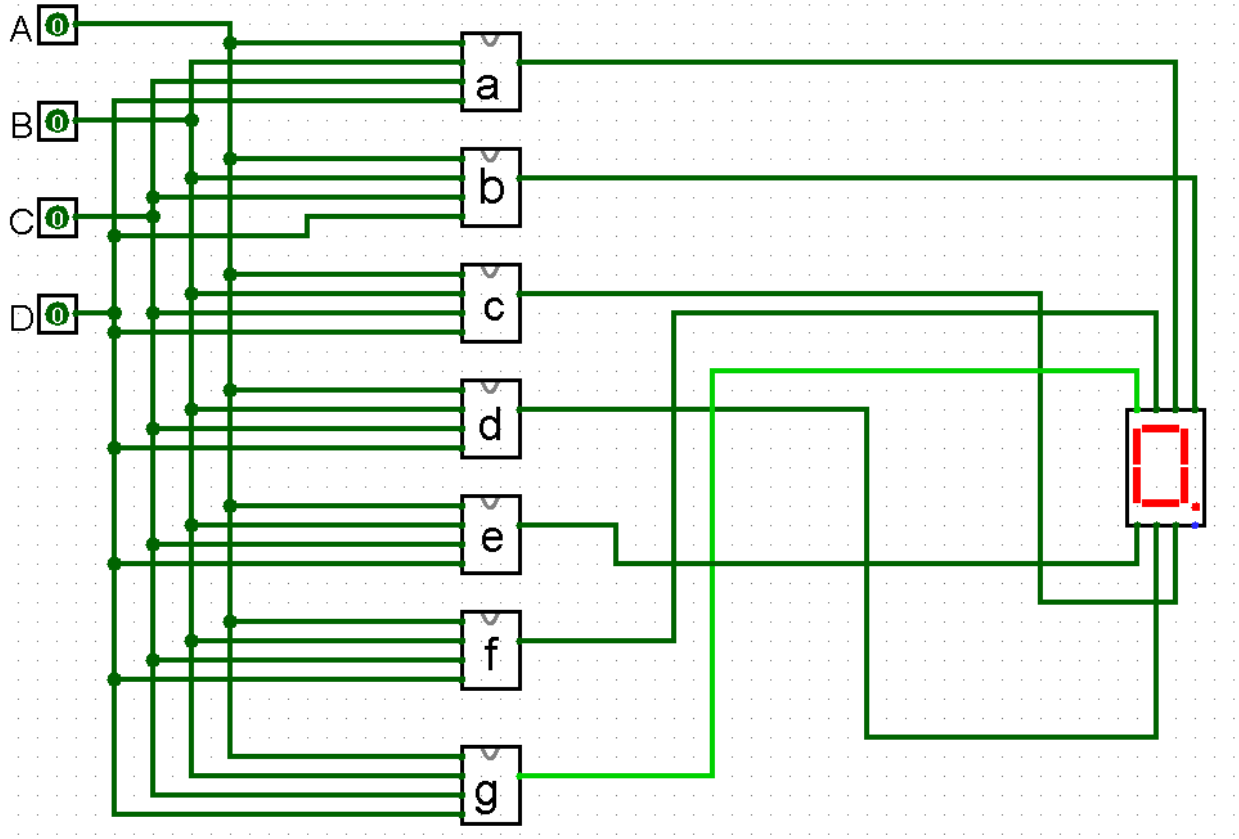

Figure 1: Converter Top Level layout

3 Detailed Design

3.1 Gate Level

The truth table for the converter, which includes its four inputs and seven outputs, is shown in Table 1. A value of zero corresponds to the segment being on, and vice-versa. From this, truth tables for each individual output were obtained and transferred to a program called Logisim, which is a free tool for designing and simulating logic circuits. This tool was used to generate minimized NAND-only Boolean expressions for each module, as using only inverting logic will reduce the number of inverters required to realize the converter circuit. Logisim was also used to generate the gate-level schematics for each module; the gate-level schematics for each module can be found in Appendix A, and the corresponding logic functions can be found in Appendix B.

Table 1: Converter Truth Table

Inputs				**Outputs**						
A	**B**	**C**	**D**	**a**	**b**	**c**	**d**	**e**	**f**	**g**
0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	1	1	1	1
0	0	1	0	0	0	1	0	0	1	0
0	0	1	1	0	0	0	0	1	1	0
0	1	0	0	1	0	0	1	1	0	0
0	1	0	1	0	1	0	0	1	0	0
0	1	1	0	0	1	0	0	0	0	0
0	1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	1	1	0	0
1	0	1	0	0	0	0	1	0	0	0
1	0	1	1	1	1	0	0	0	0	0
1	1	0	0	0	1	1	0	0	0	1
1	1	0	1	1	0	0	0	0	1	0
1	1	1	0	0	1	1	0	0	0	0
1	1	1	1	0	1	1	1	0	0	0

From the logic equations and gate-level schematics, note that the complement of any particular input is used many times. In the actual implementation of the circuit, as opposed to what is shown in the gate-level schematics, the complements of the input signals will be produced only once and reused as needed, greatly reducing the number of inverters in the circuit. Additionally, there are several terms in the logic equations that appear more than once: specifically $\overline{(\bar{A}B\bar{C}\bar{D})}$, $\overline{(AB\bar{C}D)}$, $\overline{(AB\bar{D})}$, and $\overline{(\bar{B}\bar{C}D)}$; the outputs of these gates will be reused as well.

To verify the functionality of the circuit, Logisim was used to simulate the output for each of the sixteen possible inputs. The input and output waveforms were then plotted and are shown in Figure 2. Comparing the simulated output to the converter circuit's truth table verifies that the design is valid.

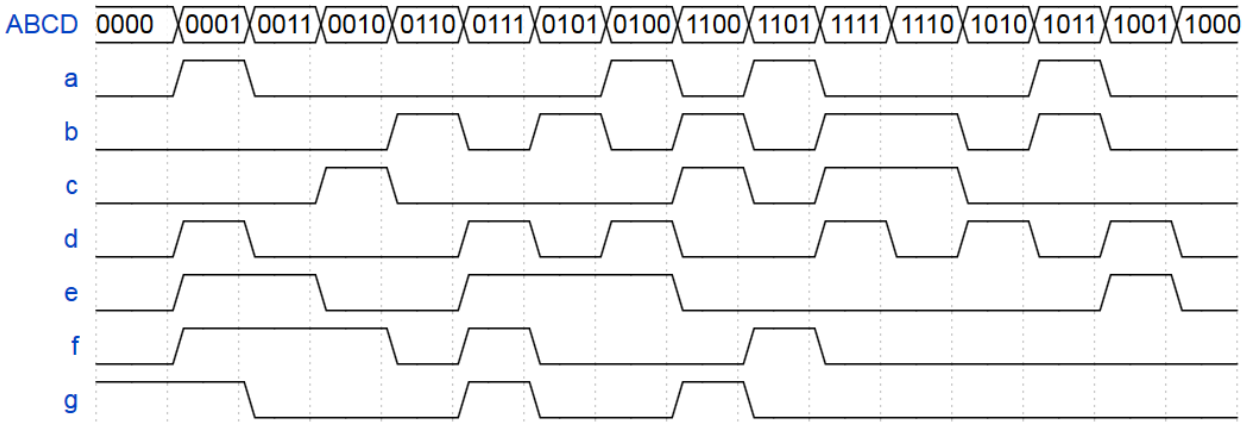

Figure 2: Simulated Input and Output of Converter

3.2 Transistor Level

3.2.1 Transistor Level Schematics

By using Logisim to implement each module with only NAND and inverter gates, the converter circuit can be realized with only a few gates: specifically two, three, and four input NAND gates, and an inverter. Efficient implementations of these gates can be designed once and reused as needed, though different sizes of these gates will need to be made to optimize for delay and power consumption. The transistor level design of these gates is shown in Figures 3, 4, 5, and 6.

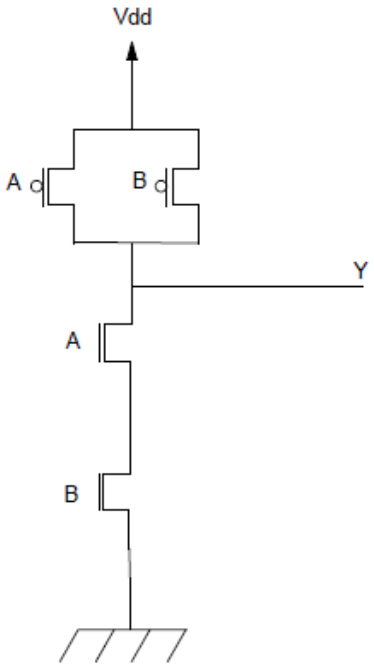

Figure 3: Transistor Level Schematic for Two-Input NAND

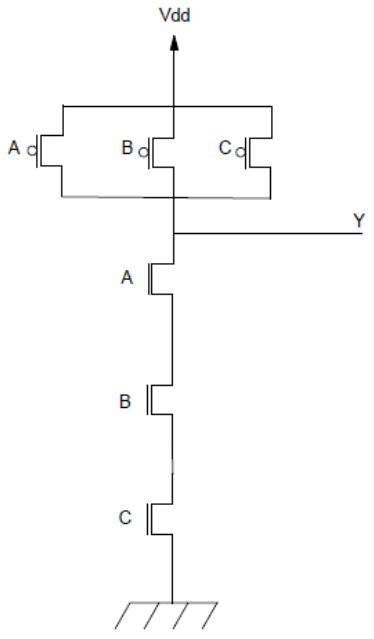

Figure 4: Transistor Level Schematic for Three-Input NAND

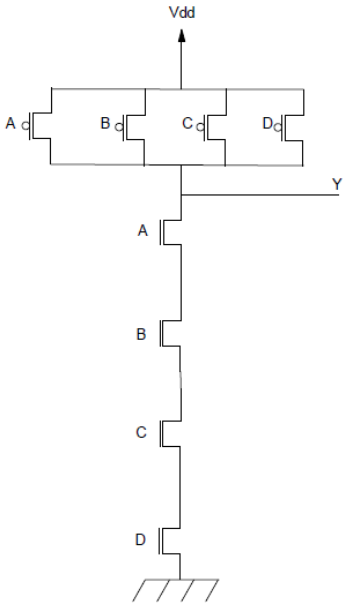

Figure 5: Transistor Level Schematic for Four-Input NAND

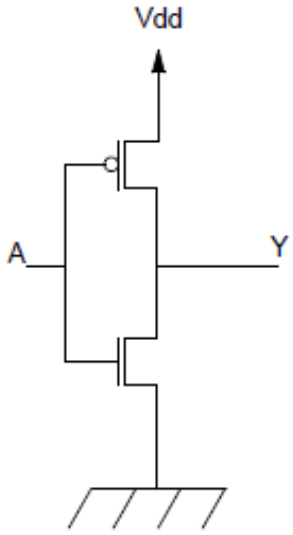

Figure 6: Transistor Level Schematic for Inverter

3.2.2 Transistor/Gate Sizing

Sizing the transistors to optimize for delay first requires knowing the input and output capacitance for each module of the converter circuit. In addition to the physical capacitance

of the external connections to the circuit it is necessary to determine the capacitance of one normalized unit, generally referred to as C. Knowledge of these two values will allow for the application of the linear delay model. The value of C is determined by constructing a unit inverter in Magic, and extracting it to a spice model. A unit inverter has an input capacitance of $3C$, and spice calculates the physical capacitance of the inverter to be 221 fF. Dividing this number by 3 yields the normalization factor of 74 fF.

The input capacitance to the circuit is based off of an MSP430 output pin. The capacitance of the output pin is readily availiable on a device datasheet, and is roughly 5 pF. This can be normalized to $67C$ for the input to the circuit.

The output capacitance is based off of the open drain transistor which forms the output of the IC. An example of an open drain circuit is shown in

With values for the input and output capacitances is it possible to calculate the critical path delay for each module. For modules a, b, d, and f the critical path is an un-inverted input feeding into a four-input NAND, the output of which is fed into another four-input NAND which produces the module output. For modules c, e, and g the critical path is an un-inverted input feeding into a three-input NAND, the output of which is fed into another four-input NAND which produces the module output. The delay for the worst case path can be found using the equation

$$D = NF^{1/N} + P \tag{1}$$

where N is the number of stages, F is the total path effort, and P is the combined parasitics of each gate in the path. F is calculated by the equation

$$F = GBH \tag{2}$$

where G is the path logical effort, B is the branching effort, and H is the path electrical effort, which is the output capacitance of the path divided by the input capacitance of the path.

G is the product of the individual logical efforts for each gate. The logical effort for a three-input NAND is 5/3, and the logical effort for a four-input NAND is 6/3. Thus, G for the critical path in modules a, b, d, and f is four, and the G for the critical path in modules c, e, and g is 10/3. Since there is no branching in any of the circuits, the branching effort B for each path is one. Each module has an input capacitance of 67C and an output capacitance of 550C, so H for each path is 8.2. Using these values, the critical path effort for modules a,

b, d, and f is 32.8 and the critical path effort for modules c, e, and g is 27.363.

With values for F it is possible to calculate the delay of the critical path for each circuit. The parasitic for a three-input NAND is three and the parasitic for a four-input NAND is four, so the path parasitic for modules a, b, d, and f is eight, and the path parasitic for modules c, e, and g is seven. The number of stages in each critical path is two, so using Equation 2 the critical path delay for modules a, b, d, and f is 19.46 and the critical path for modules c, e, and g is 17.46.

To check whether these delays are close to optimal, the path effort F can be used to calculate the optimal number of stages, $\hat{N}$ with the equation

$$\hat{N} = log_4(F) \tag{3}$$

For each module, the resulting optimal number of stages is three. The three-stage critical path for each module is simply the two-stage critical path with an inverter placed at the input, and the calculated delays for these paths are 18.6 for modules a, b, d, and f and 17.0 for modules c, e, and g. Comparing these delays to the delays of the two-stage paths, the effect of adding another stage is so marginal that it is not worth it to add extra stages to the critical path, which would decrease the path effort and thus decrease the delay at the cost of higher power consumption.

Since the decrease in delay is not worth the increase in power, the critical paths remain unmodified in the circuit. Thus, the gates in the critical paths can be sized using the equation

$$C_{in} = \frac{C_{out} * g}{F^{1/N}} \tag{4}$$

where C_{in} and C_{out} are the input and output capacitance of the gate and g is the logical effort of the gate. Starting at the last gate, the output capacitance of the path is used to calculate the input capacitance, which determines the size of the gate; these calculations cascade until the end of the path is reached. For the critical path for modules a, b, d, and f, stage one ends up having an input capacitance of 67C and stage 2 ends up having an input capacitance of 192C. By dividing these by the input capacitances of the corresponding unit-sized gates, this results in stage one having a size of 11 and stage two having a size of 32. Similarly, stage one for modules c, e, and g has an input capacitance of 67C and a size of 11, and stage two has an input capacitance of 175C and a size of 29. Since stage one for each critical path is a four-input NAND of size 11, only one four-input NAND gate will have to be designed for this stage. The results of this section are summarized in Table 2.

Table 2: Critical Path Delay and Stage Sizes

Modules	Critical Path Delay	Stage One Size	Stage Two Size
a, b, d, f	19.5	11	32
c, e, g	17.5	11	29

A Segment Logic Diagrams

A.1 Segment a

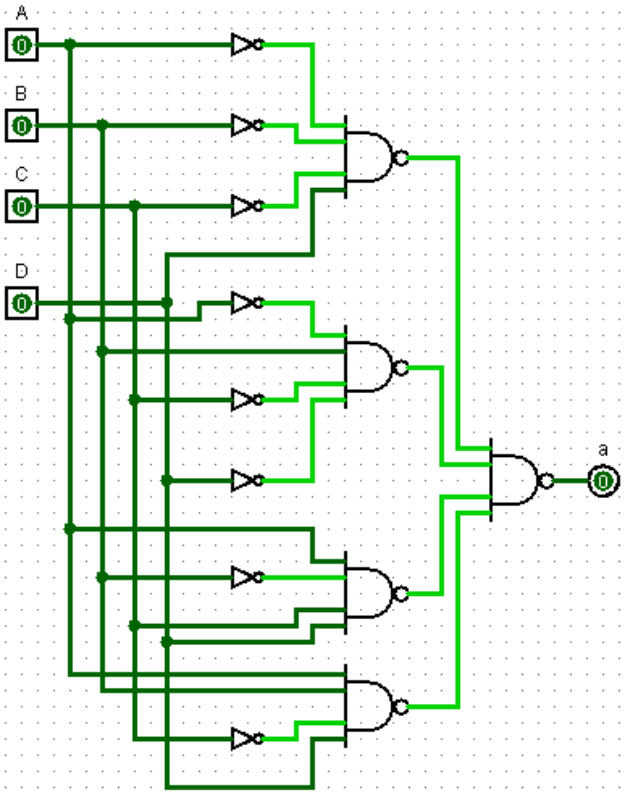

Figure 7: Block a Gate Level Schematic

A.2 Segment b

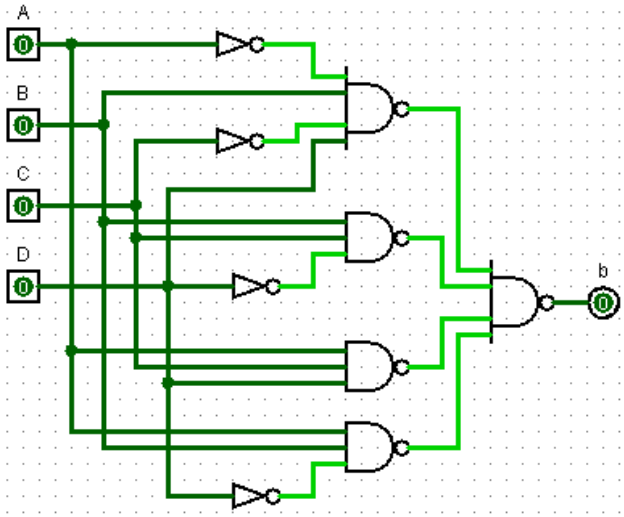

Figure 8: Block b Gate Level Schematic

A.3 Segment c

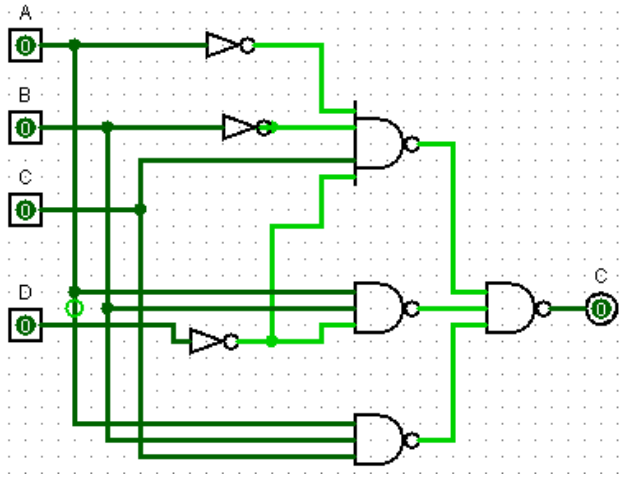

Figure 9: Block c Gate Level Schematic

A.4 Segment d

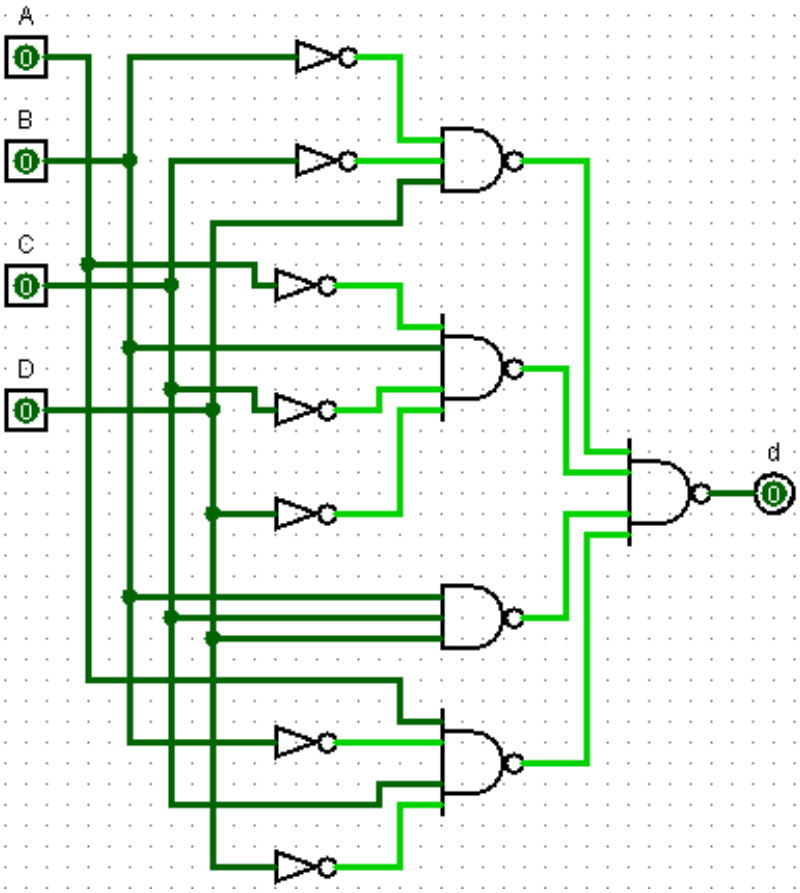

Figure 10: Block d Gate Level Schematic

A.5 Segment e

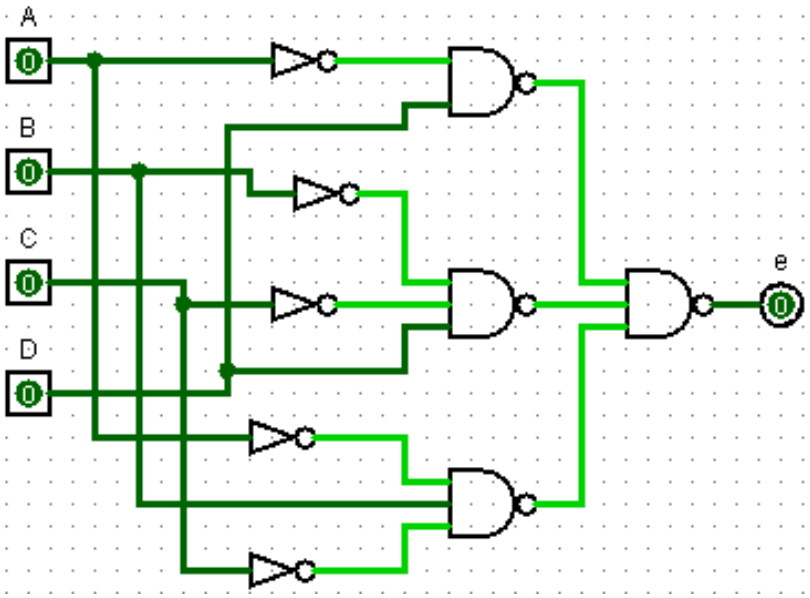

Figure 11: Block e Gate Level Schematic

A.6 Segment f

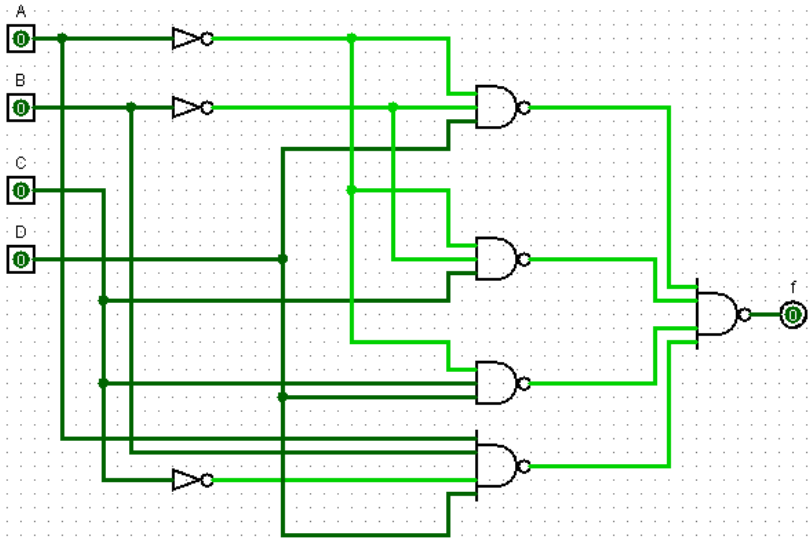

Figure 12: Block f Gate Level Schematic

A.7 Segment g

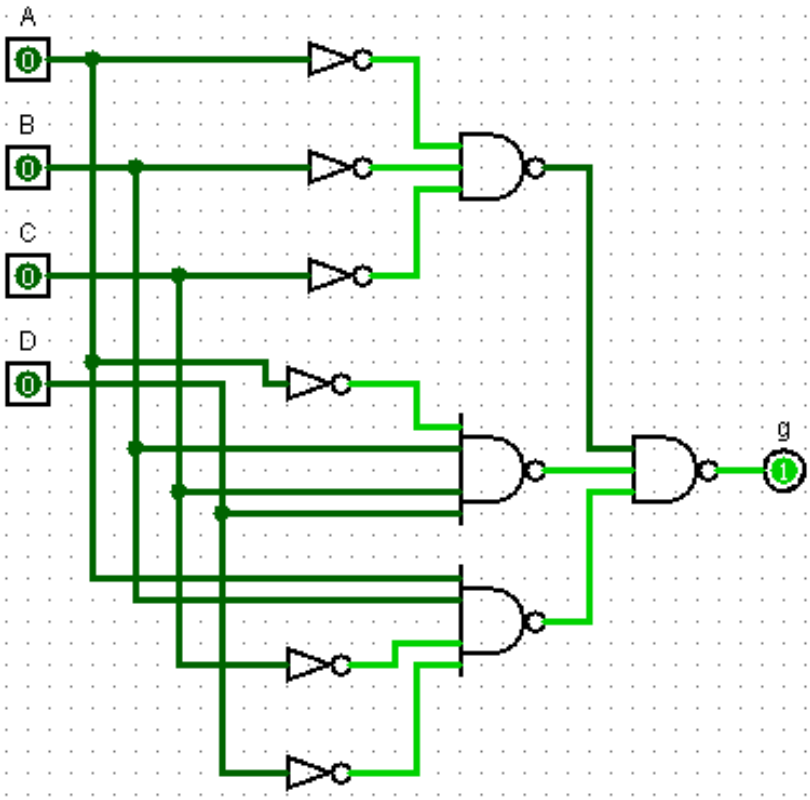

Figure 13: Block g Gate Level Schematic

B Logic Equations

$$a = \overline{\overline{(\bar{A}\bar{B}\bar{C}D)}\,\overline{(\bar{A}B\bar{C}\bar{D})}\,\overline{(A\bar{B}CD)}\,\overline{(AB\bar{C}D)}} \tag{5}$$

$$b = \overline{\overline{(\bar{A}B\bar{C}D)}\,\overline{(BC\bar{D})}\,\overline{(ACD)}\,\overline{(AB\bar{D})}} \tag{6}$$

$$c = \overline{\overline{(\bar{A}\bar{B}CD)}\,\overline{(AB\bar{D})}\,\overline{(ABC)}} \tag{7}$$

$$d = \overline{\overline{(\bar{B}\bar{C}D)}\,\overline{(\bar{A}B\bar{C}\bar{D})}\,\overline{(BCD)}\,A\bar{B}C\bar{D}} \tag{8}$$

$$e = \overline{\overline{(\bar{A}D)}\,\overline{(\bar{B}\bar{C}D)}\,\overline{(\bar{A}B\bar{C})}} \tag{9}$$

$$f = \overline{\overline{(\bar{A}\bar{B}D)}\,\overline{(\bar{A}\bar{B}C)}\,\overline{(\bar{A}CD)}\,\overline{(AB\bar{C}D)}} \tag{10}$$

$$g = \overline{\overline{(\bar{A}\bar{B}\bar{C})}\,\overline{(\bar{A}BCD)}\,\overline{(AB\bar{C}\bar{D})}} \tag{11}$$