

EE 648 – VLSI Design

Binary Coded Hexadecimal to Seven-Segment  
Display Converter

---



---

RYKER DIAL  
CODY GOSSEL  
ZACH KREHLIK

April 18, 2017

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Top Level Design</b>	<b>1</b>
<b>3</b>	<b>Detailed Design</b>	<b>2</b>
3.1	Gate Level . . . . .	2
3.2	Transistor Level . . . . .	4
3.2.1	Transistor Level Schematics . . . . .	4
3.2.2	Transistor/Gate Sizing . . . . .	7
<b>4</b>	<b>Magic VLSI Layout</b>	<b>10</b>
<b>A</b>	<b>Segment Logic Diagrams</b>	<b>12</b>
A.1	Segment a . . . . .	12
A.2	Segment b . . . . .	13
A.3	Segment c . . . . .	13
A.4	Segment d . . . . .	14
A.5	Segment e . . . . .	15
A.6	Segment f . . . . .	15
A.7	Segment g . . . . .	16
<b>B</b>	<b>Logic Equations</b>	<b>17</b>
<b>C</b>	<b>Magic VLSI Layouts</b>	<b>18</b>

# 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.

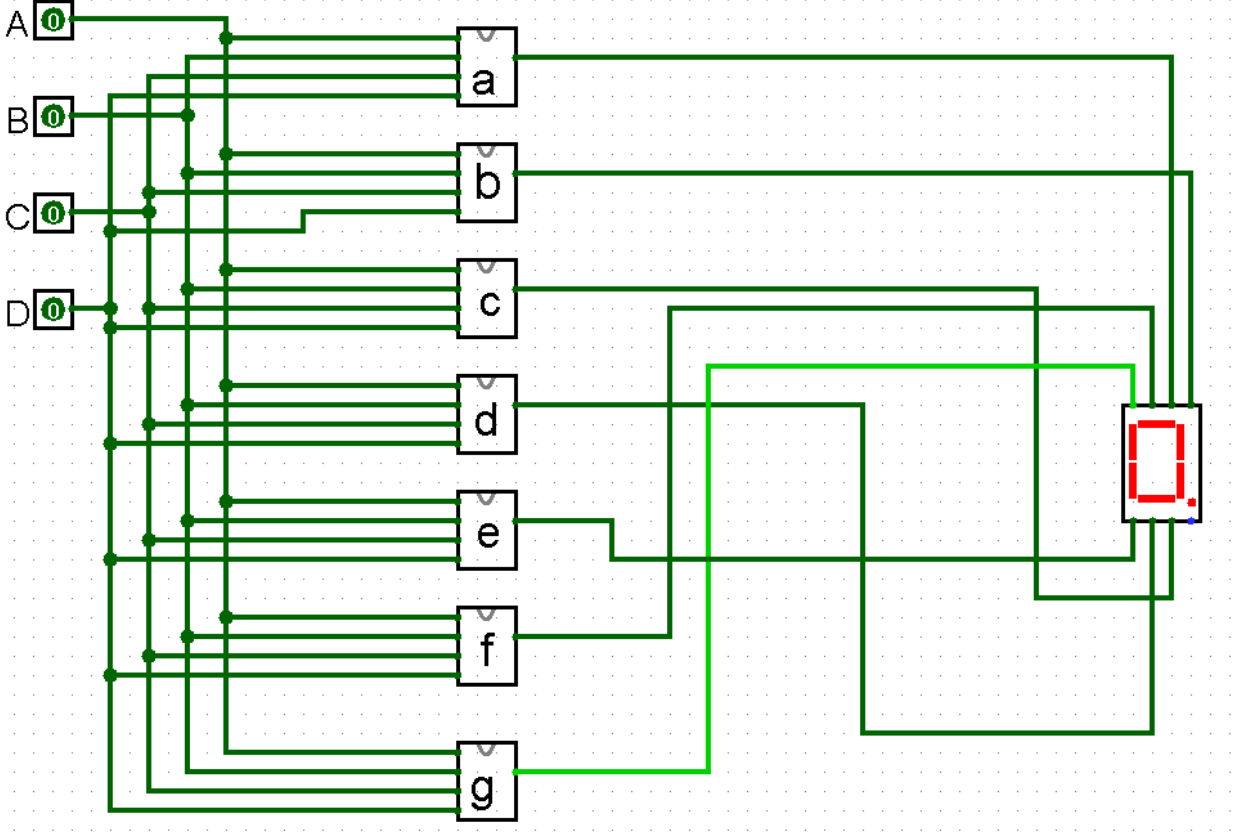


Figure 1: Converter Top Level layout

### 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  $(\overline{A}\overline{B}\overline{C}\overline{D})$ ,  $(\overline{A}\overline{B}\overline{C}D)$ ,  $(\overline{A}\overline{B}D)$ , and  $(\overline{B}\overline{C}D)$ ; 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.

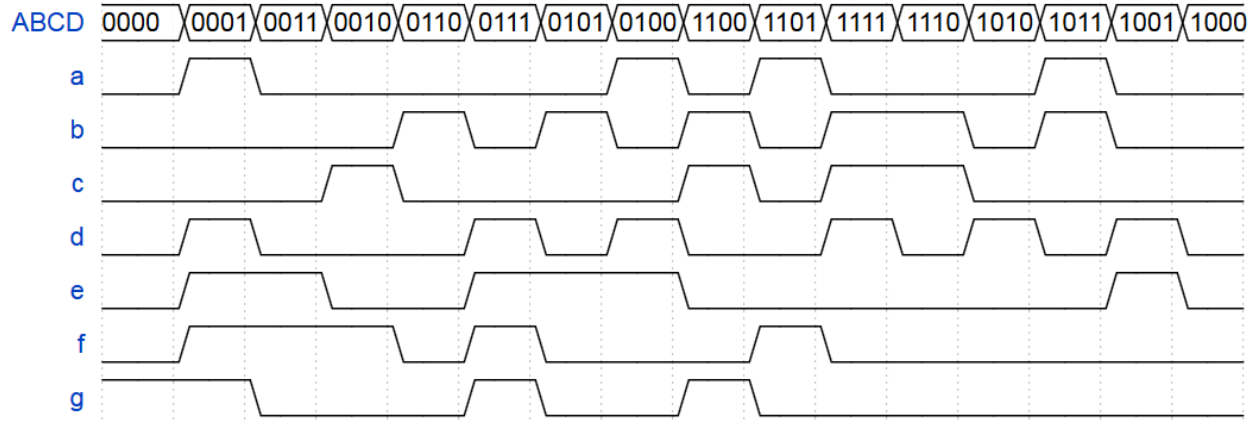


Figure 2: Simulated Input and Output of Converter

## 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.

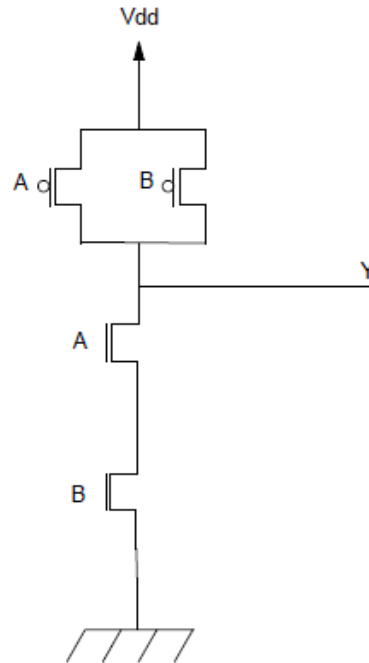


Figure 3: Transistor Level Schematic for Two-Input NAND

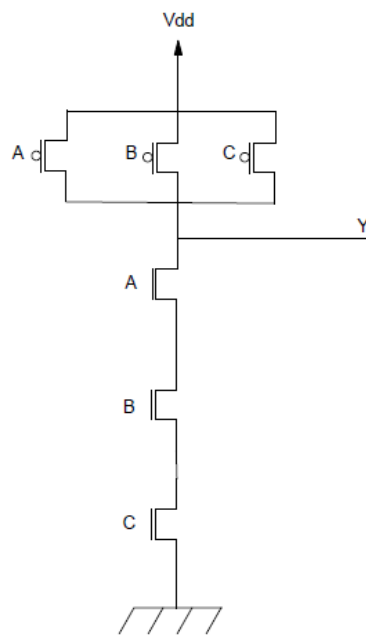


Figure 4: Transistor Level Schematic for Three-Input NAND

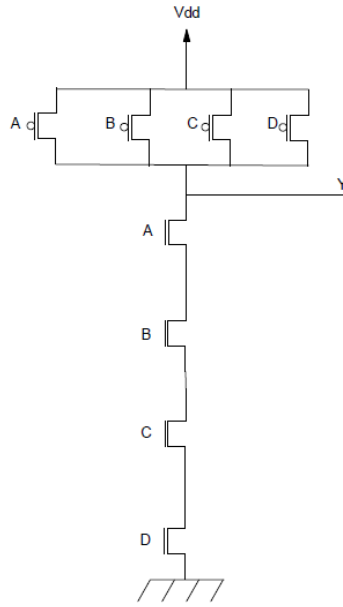


Figure 5: Transistor Level Schematic for Four-Input NAND

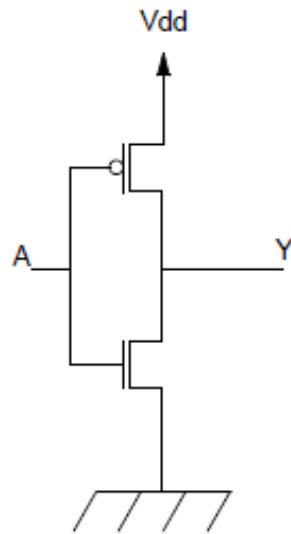


Figure 6: Transistor Level Schematic for Inverter



### 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 available 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 Figure 7.

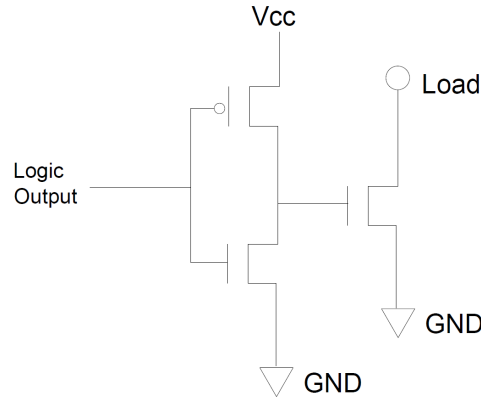


Figure 7: Block a Gate Level Schematic

The nmos will present either a hi-z or low voltage to the external device. One end of the external device is attached to an external voltage source, and the other is attached to the load pin of the nmos. When the nmos is turned on it allows for current to flow through the LED, which causes it to light up.

Sizing of the open drain transistor is determined by considering the current necessary for an LED segment and a desired voltage drop. In order to keep dissipated power through the IC low, a  $V_{ds}$  of 0.2 V is selected for the transistor. The maximum current required will be

approximately 20 mA. These two values can be used with the long channel mosfet equations to determine a required channel width, assuming that the channel length will be held at the minimum. The relation between transistor current and channel width is given by Equation (1).

$$I_{ds} = \mu C_{ox} \frac{W}{L} (V_{gt} - V_{ds}) V_{ds} \quad (1)$$

There are several given values in the equation,  $\mu$ , the electron mobility can be found in the spice technology file and is  $275 \text{ cm}^2 \text{ V}^{-1} \text{ s}^{-1}$ .  $C_{ox}$  is the per area capacitance of the oxide layer, and is derived from the  $40 \text{ \AA}$  oxide thickness used by TSMC.  $V_{gt}$ ,  $V_{ds}$ , and  $I_{ds}$  are based on the 3.3 V logic level of the IC less the threshold voltage of 0.7 V, the desired  $V_{ds}$  of 0.2 V, and the maximum expected current of 20 mA respectively.

Rearranging the equation and solving for  $W$  when  $L$  is fixed at 180 nm results in a width of  $3.3 \times 10^{-5} \text{ m}$ , which is approximately  $550\lambda$  in magic. The input capacitance of a transistor is proportional to the width, with each  $\lambda$  resulting in  $C$  worth of capacitance, so the open drain output is said to present  $550C$  worth of capacitance to the logic circuit.

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

$$D = NF^{1/N} + P \quad (2)$$

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

$$F = GBH \quad (3)$$

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 3 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

$$\hat{N} = \log_4(F) \quad (4)$$

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

$$C_{in} = \frac{C_{out} * g}{F^{1/N}} \quad (5)$$

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

## 4 Magic VLSI Layout

Implementing the circuit in bare silicon required first constructing VLSI layouts for each gate. To ease the assemblage of the full layout, the layout was started by designing every gate to be unit sized; this allows for expedient construction of the circuit and verification of the layout. Additionally, the gates were laid out such that increasing the transistor size increases the width of the layout, while the height stays constant. The  $V_{dd}$  and  $gnd$  rails of the transistor are at the top and bottom of the layout, respectively, so this means that resizing the transistors does not move the voltage rails. These design choices make it easier to size the gates for optimal delay at a later date.

Figures 15, 16, 17, and 18 in Appendix C show the VLSI layouts for the inverter and the two, three, and four input NAND gates. The functionality of each gate was verified by simulation with ngspice. During these simulations, the gate delay, rise, and fall times were also measured. Table 3 shows these measurements, as well as the physical dimensions of each gate.

Table 3: Select Transistor Properties

Gate	Delay	Rise	Fall	Width ( $\lambda$ )	Height ( $\lambda$ )	Area ( $\lambda^2$ )
INV				128	86	11,008
2NAND				128	90	11,520
3NAND				128	94	12,032
4NAND				128	98	12,544

## A Segment Logic Diagrams

### A.1 Segment a

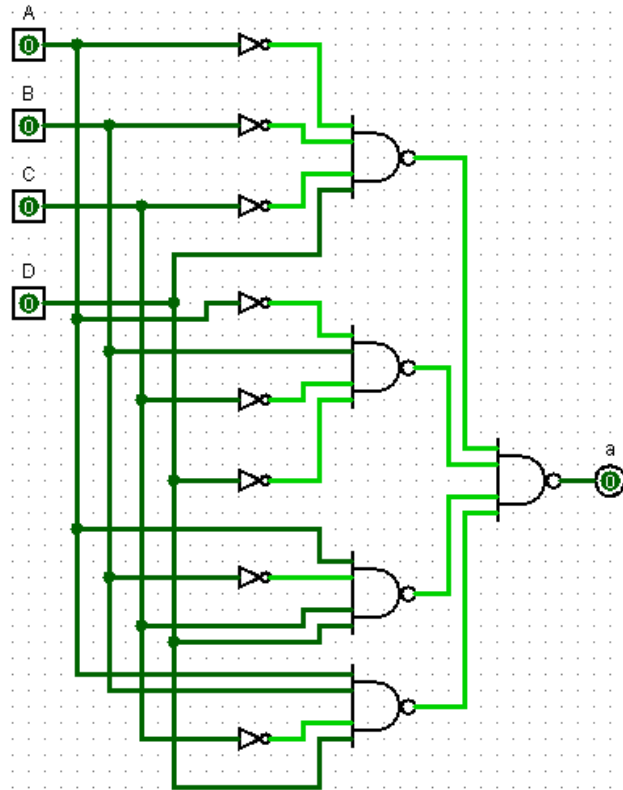


Figure 8: Block a Gate Level Schematic

## A.2 Segment b

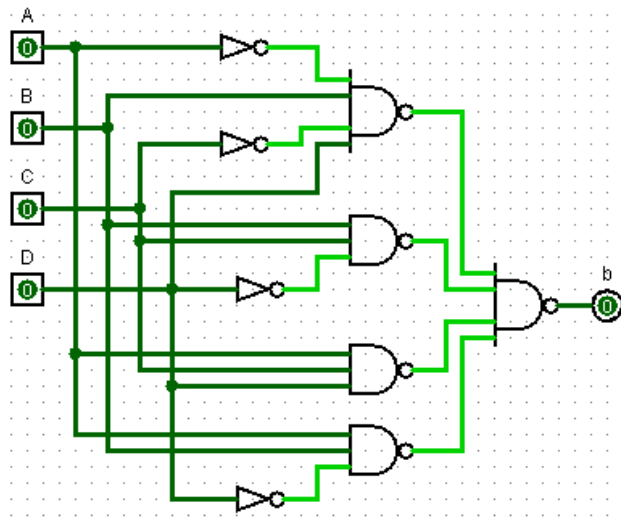


Figure 9: Block b Gate Level Schematic

## A.3 Segment c

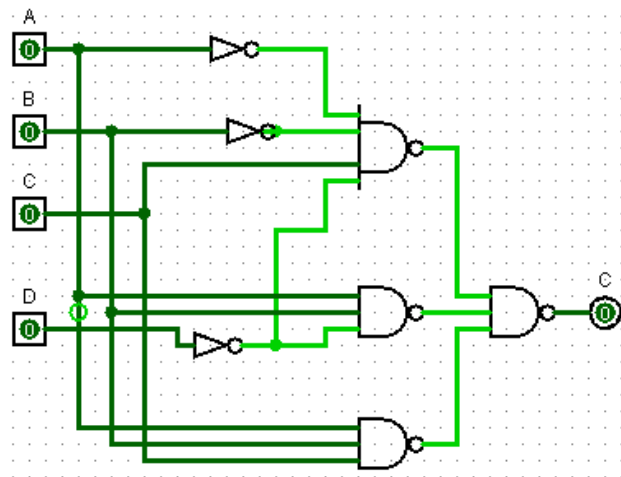


Figure 10: Block c Gate Level Schematic

### A.4 Segment d

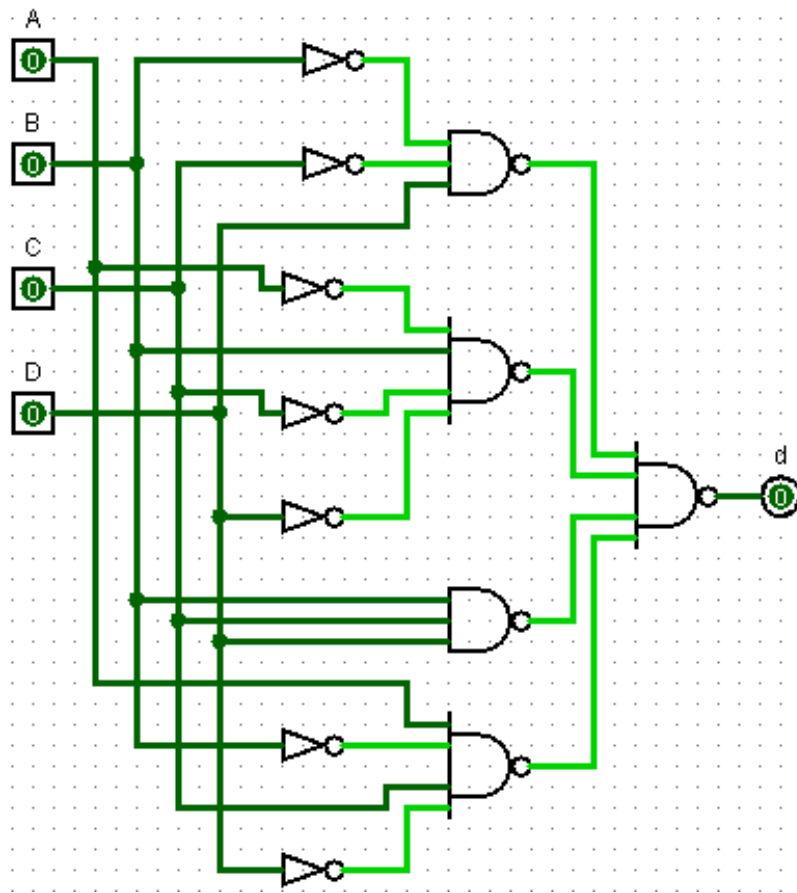


Figure 11: Block d Gate Level Schematic



### A.5 Segment e

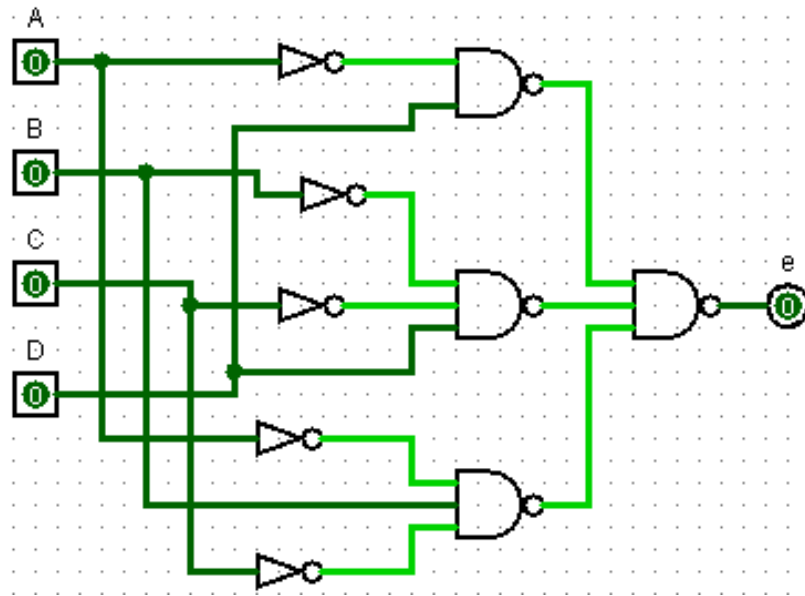


Figure 12: Block e Gate Level Schematic

### A.6 Segment f

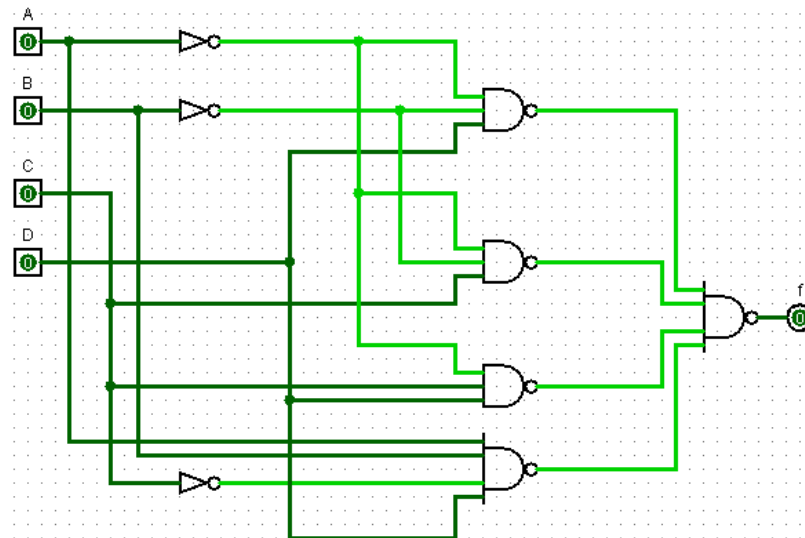


Figure 13: Block f Gate Level Schematic

## A.7 Segment g

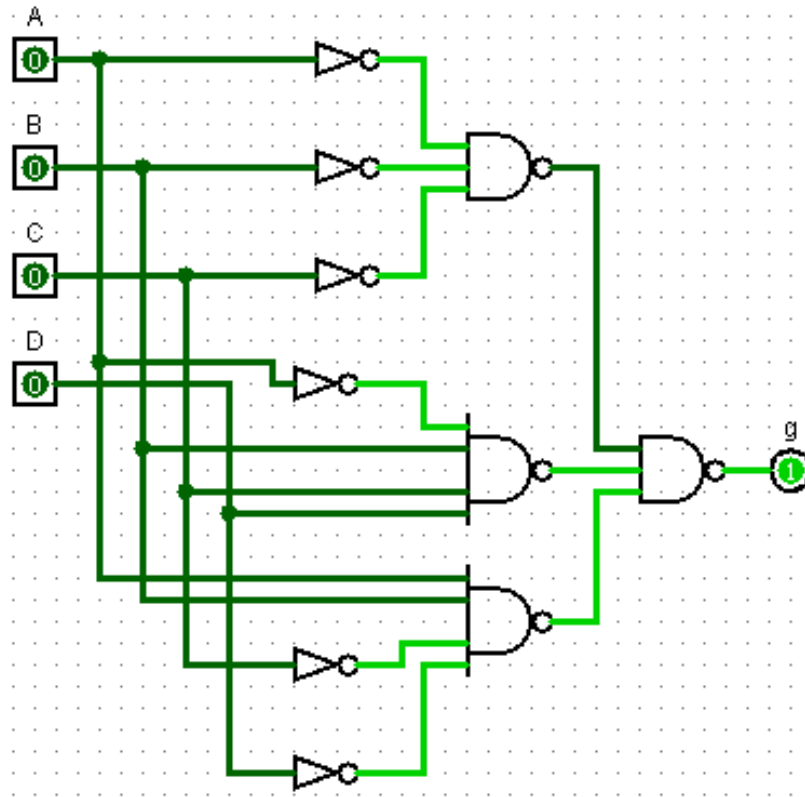


Figure 14: Block g Gate Level Schematic

## B Logic Equations

$$a = \overline{\overline{(\bar{A}\bar{B}\bar{C}D)(\bar{A}\bar{B}\bar{C}\bar{D})(A\bar{B}CD)(AB\bar{C}D)}} \quad (6)$$

$$b = \overline{\overline{(\bar{A}\bar{B}\bar{C}D)(\bar{B}\bar{C}\bar{D})(ACD)(AB\bar{D})}} \quad (7)$$

$$c = \overline{\overline{(\bar{A}\bar{B}\bar{C}\bar{D})(AB\bar{D})(ABC)}} \quad (8)$$

$$d = \overline{\overline{(\bar{B}\bar{C}D)(\bar{A}\bar{B}\bar{C}\bar{D})(\bar{B}CD)\bar{A}\bar{B}\bar{C}\bar{D}}} \quad (9)$$

$$e = \overline{\overline{(\bar{A}D)(\bar{B}\bar{C}D)(\bar{A}\bar{B}\bar{C})}} \quad (10)$$

$$f = \overline{\overline{(\bar{A}\bar{B}D)(\bar{A}\bar{B}C)(\bar{A}CD)(AB\bar{C}D)}} \quad (11)$$

$$g = \overline{\overline{(\bar{A}\bar{B}\bar{C})(\bar{A}\bar{B}CD)(AB\bar{C}\bar{D})}} \quad (12)$$

## C Magic VLSI Layouts

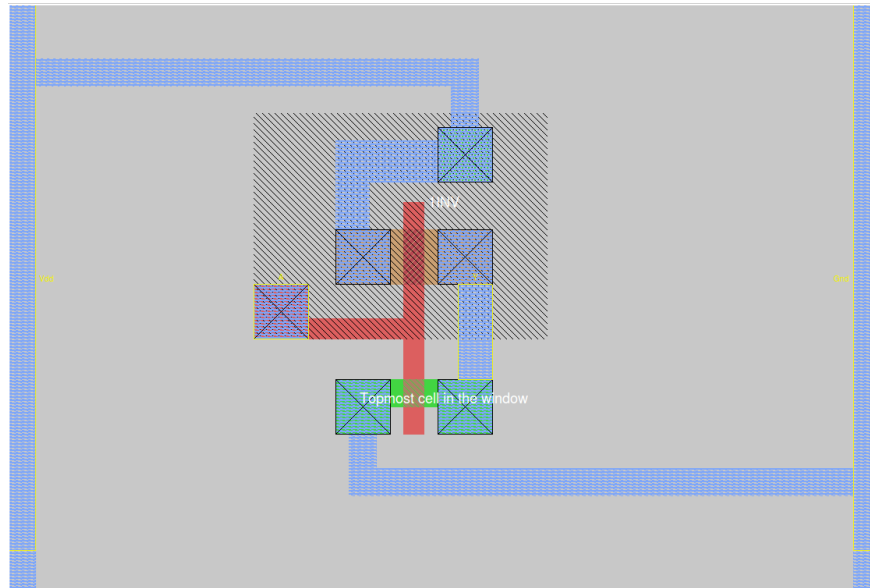


Figure 15: VLSI Layout for Inverter

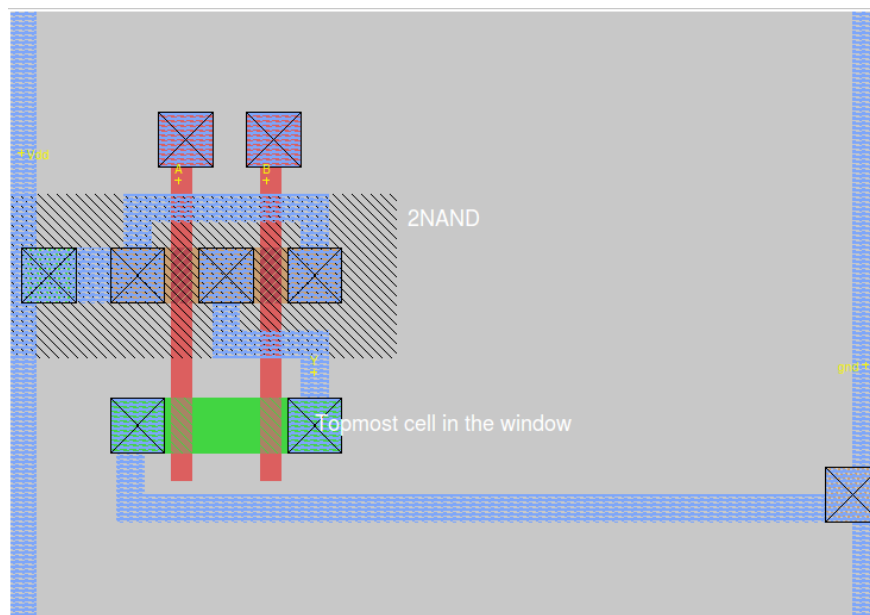


Figure 16: VLSI Layout for 2-Input NAND

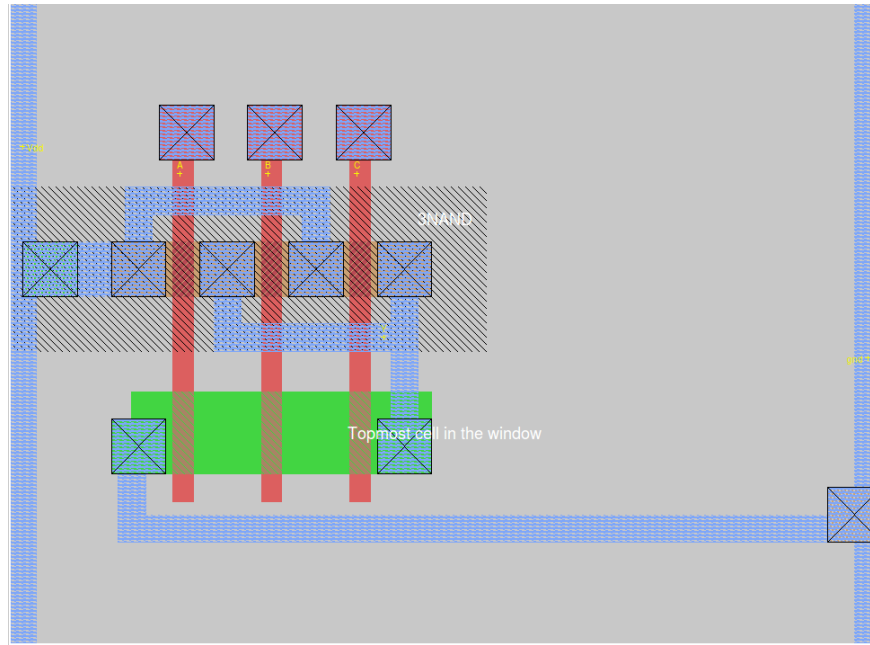


Figure 17: VLSI Layout for 3-Input NAND

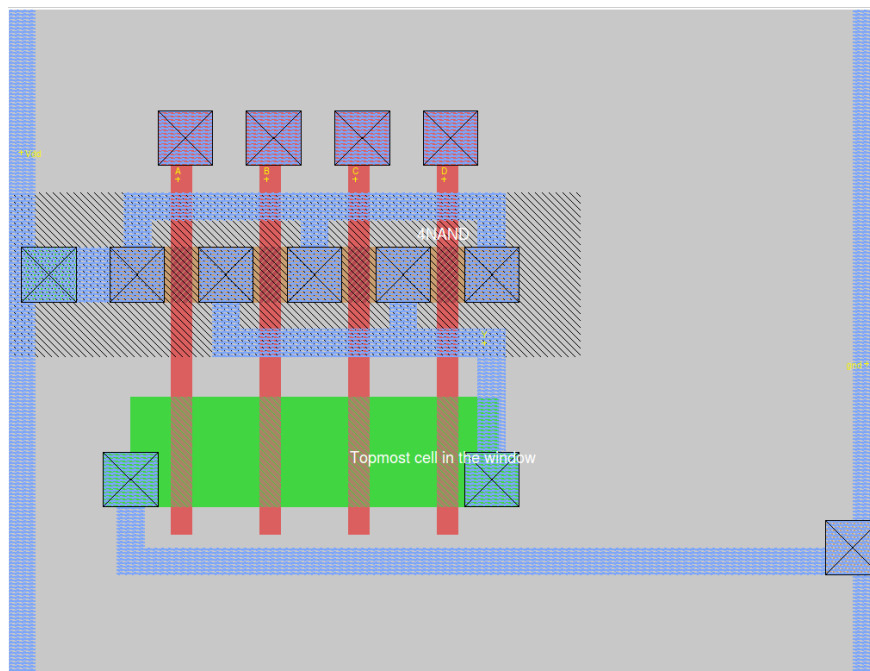


Figure 18: VLSI Layout for 4-Input NAND