EE 648 – VLSI Design

Binary Coded Hexadecimal to Seven-Segment Display Converter

Ryker Dial

Cody Gossel

Zach Krehlik

April 18, 2017

Contents

1 Introduction **1**

2 Top Level Design **1**

3 Detailed Design **2**

3.1 Gate Level 2

3.2 Transistor Level 3

3.2.1 Transistor Level Schematics 3

3.2.2 Transistor/Gate Sizing 6

4 Magic VLSI Layout **9**

4.1 Individual Gate Layout 9

4.2 Full Circuit Layout 10

4.3 Spice Simulations 10

A Segment Logic Diagrams **14**

A.1 Segment a 14

A.2 Segment b 15

A.3 Segment c 15

A.4 Segment d 16

A.5 Segment e 17

A.6 Segment f 17

A.7 Segment g 18

B Logic Equations **19**

C Magic VLSI Layouts **20**

D SPICE Simulations **24**

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.

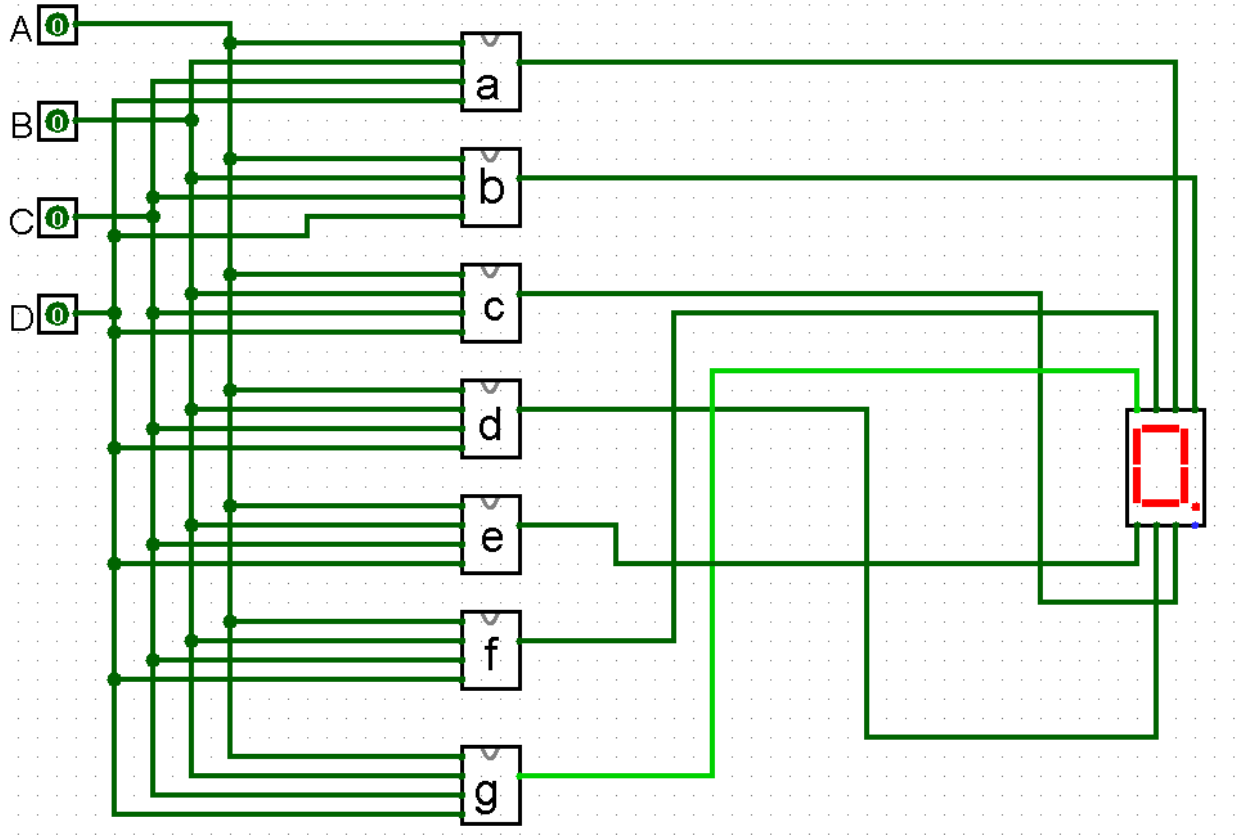


Figure 1: Converter Top Level layout

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.

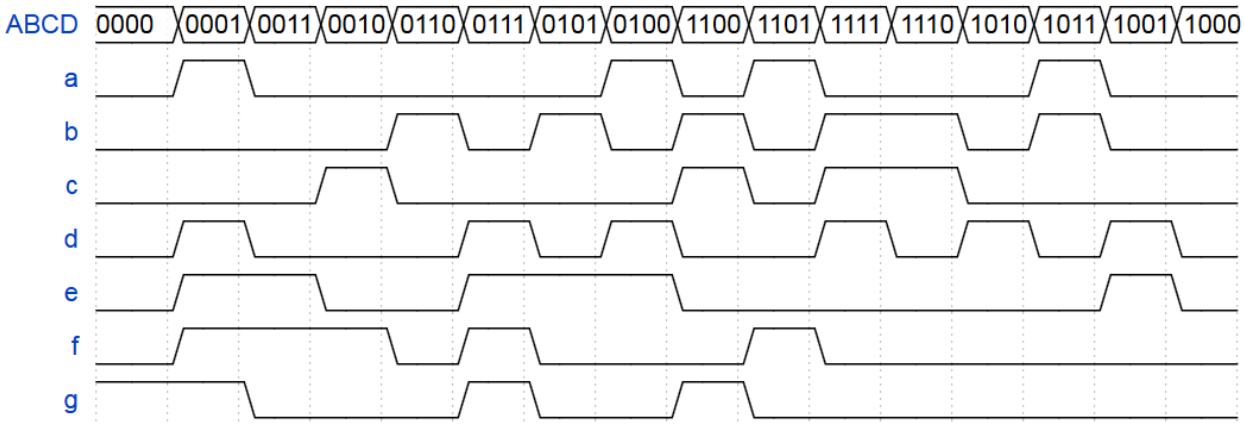


Figure 2: Simulated Input and Output of Converter

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.

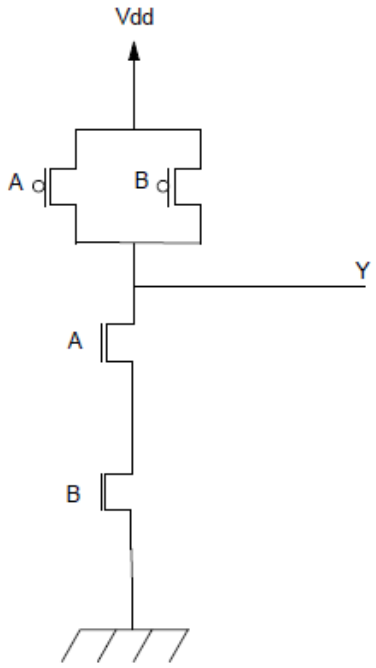


Figure 3: Transistor Level Schematic for Two-Input NAND

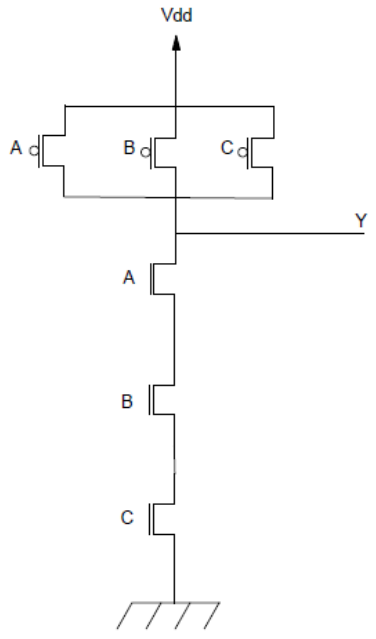


Figure 4: Transistor Level Schematic for Three-Input NAND

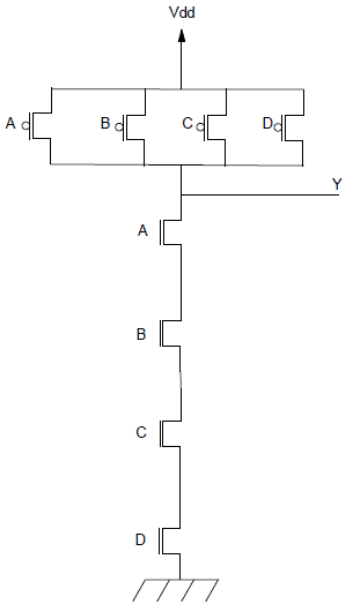


Figure 5: Transistor Level Schematic for Four-Input NAND

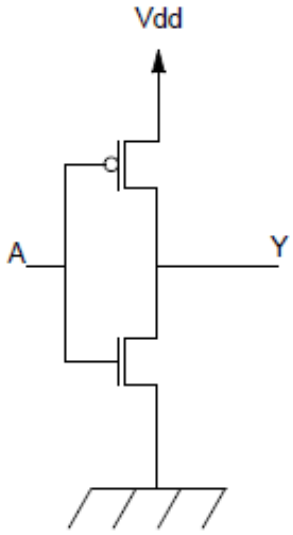


Figure 6: Transistor Level Schematic for Inverter

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 Figure 7.

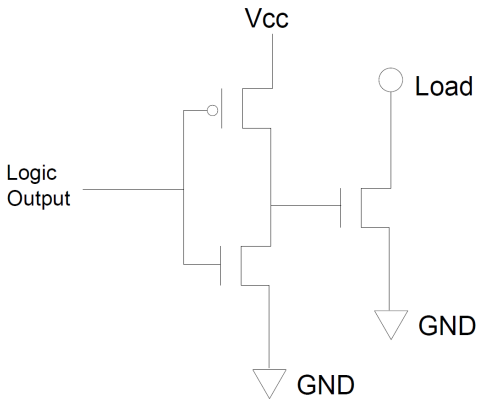


Figure 7: Block a Gate Level Schematic

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 Vds 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).

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

There are several given values in the equation, μ, the electron mobility can be found in the spice technology file and is $275\,\mathrm{cm\,V^{-2}}$. C_{ox} is the per area capacitance of the oxide layer, and is derived from the 40 Å 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 3.3×10^{-5} m, which is approximately 550λ in magic. The input capacitance of a transistor is proportional to the width, with each λ 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

$$D = NF^{1/N} + P \tag{2}$$

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

$$F = GBH \tag{3}$$

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

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

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

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

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

4.1 Individual Gate 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 19, 20, 21, and 22 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. Figures 25, 26, 27, and 28 in Appendix D show these simulations.

During simulations the delay, rise, and fall times were also measured from visual inspection of the waveform. Delay is measured from 50% on the input to 50% on the output, the rise time is measured from 20% on the output to 80% on the output, and fall time is measured from 80% on the output to 20% on the output; Figure 8 shows an example of how these were determined. For the two, three, and four input NAND gates, the circuits used to test the delay, rise, and fall times consisted of the device under test, with a unit inverter on

each input and the output driving four inputs to circuits of the same type. Figure 10 shows this setup for the 2-input NAND.

For the inverter circuit, the fan-out-of-four delay (FO4) was measured; the setup for this is shown in Figure 11. Table 3 shows these measurements, as well as the physical dimensions of each gate.

Table 3: Transistor Delay, Rise, and Fall Times

Gate	Rise	Fall	Falling Delay	Rising Delay	Width	Height	Area
INV	50 ps	55 ps	40 ps	40 ps	128λ	86λ	11,008λ
2NAND	135 ps	120 ps	120 ps	90 ps	128λ	90λ	11,520λ
3NAND	215 ps	130 ps	135 ps	100 ps	128λ	94λ	12,032λ
4NAND	265 ps	155 ps	185 ps	110 ps	128λ	98λ	12,544λ

4.2 Full Circuit Layout

After constructing the individual gates, the next step was to lay them out into the full converter circuit. To accomplish this, first each segment was laid out

4.3 Spice Simulations

The circuit was simulated at the individual segment level and also as a complete layout. With 4 inputs it is reasonable to do a spanning sweep of all input possibilities and verify the functionality in every case. A tool was created in C# to generate the input waveforms in a manner similar to the layout of a truth table, resulting in a binary count from 0 to 15. NG spice was then used to stimulate the circuits with the input sweep, and export the results into a ASCII file. The ASCII file is then loaded into Matlab where an output figure showing both the input stimuli and outputs from the circuit.

Figure 9 shows the output from the spice simulation for the entire circuit. Input D is the least significant bit of the input number and input A is the most significant bit. The signals beginning with "Seg" are the outputs to the 7 segment display. The outputs are active low, meaning that a 0 value will turn on the LED segment. The first time block corresponds to 0,

and every segment except for segment G is active. A zero on a 7 segment display lights every segment except for the middle horizontal segment, which matches the output from spice.

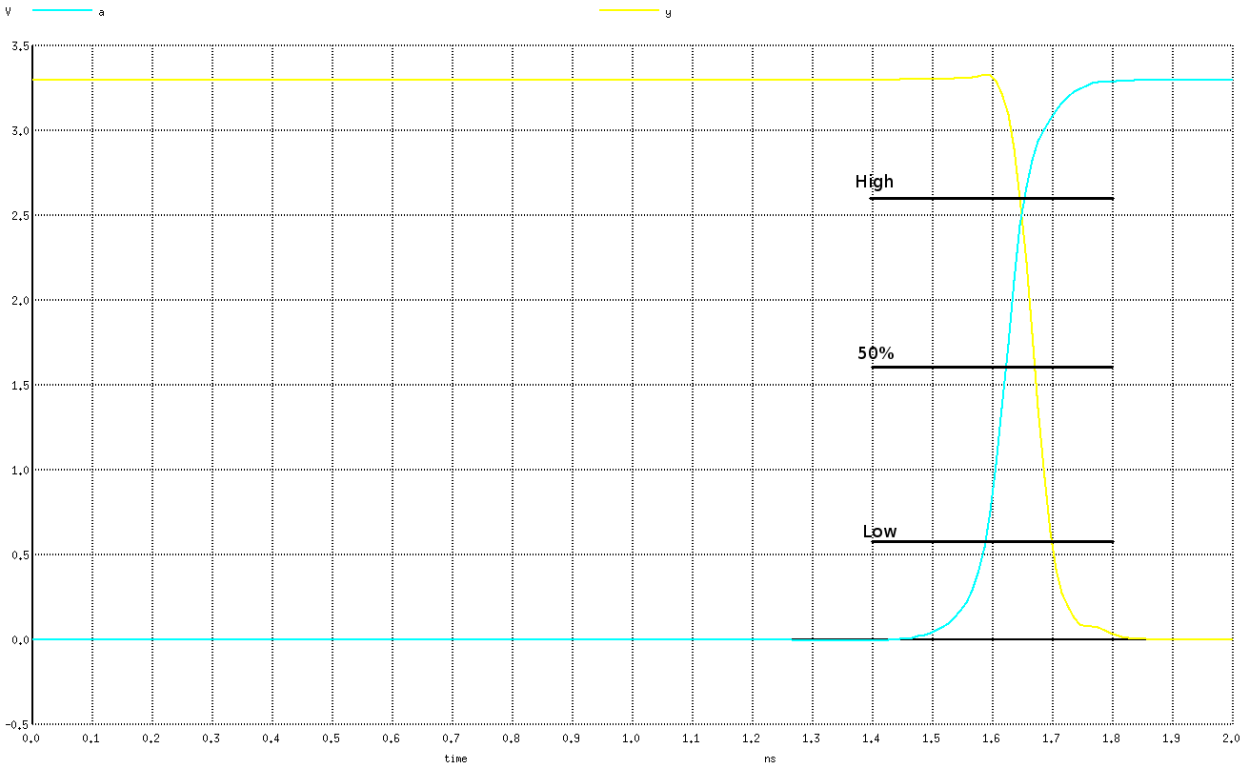


Figure 8: Measuring FO4 Delay and Rise/Fall Times of Inverter

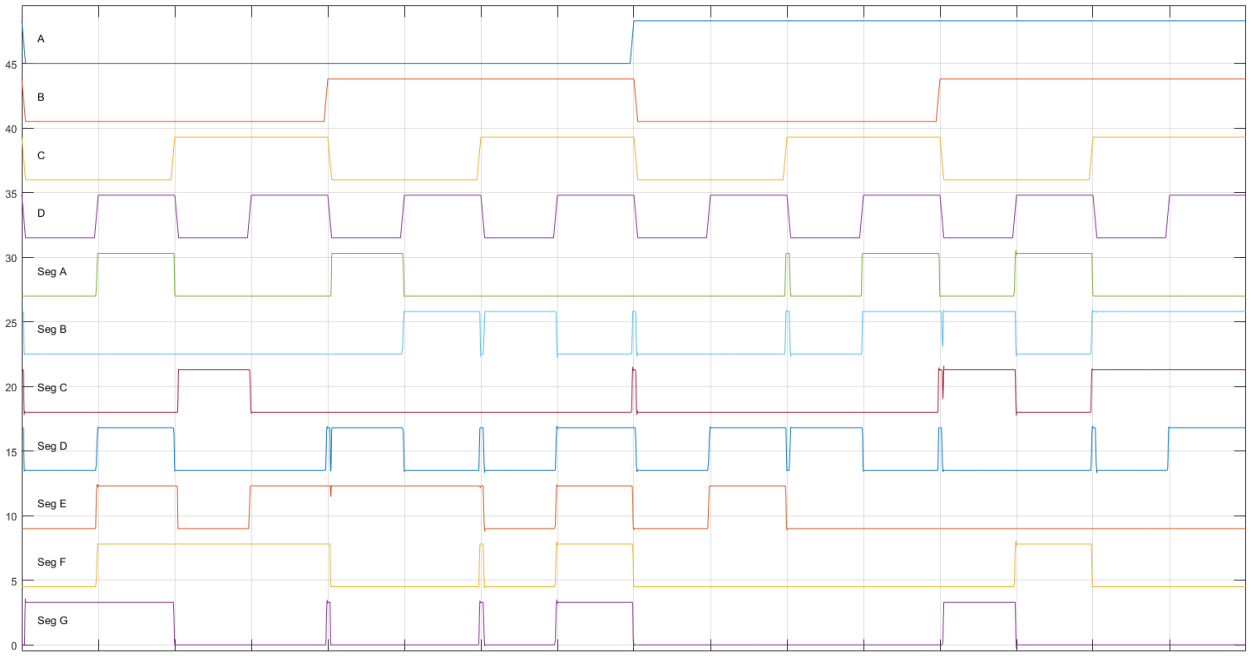


Figure 9: Spice Simulation of Complete Circuit

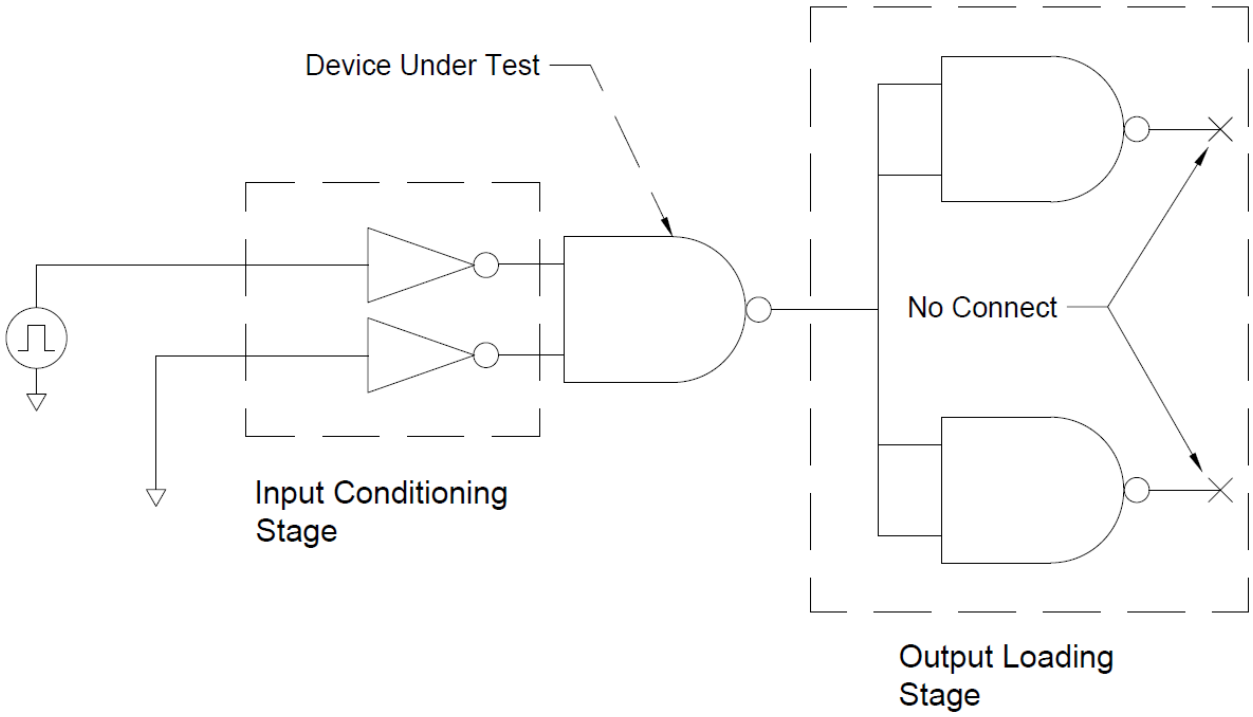


Figure 10: Delay and Rise/Fall Time Measurement Setup for NAND Gates

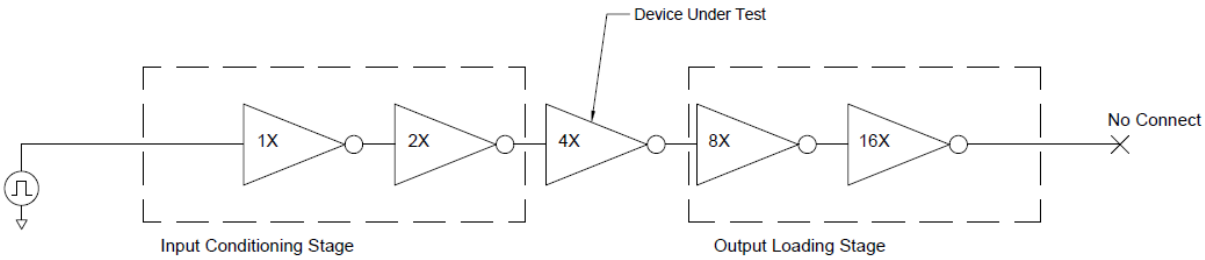


Figure 11: Delay and Rise/Fall Time Measurement Setup for Inverter

A Segment Logic Diagrams

A.1 Segment a

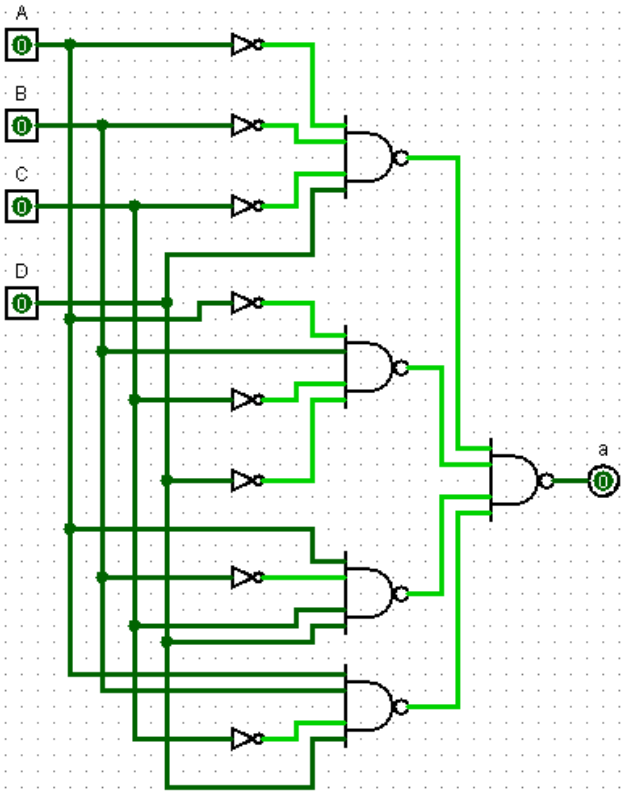


Figure 12: Block a Gate Level Schematic

A.2 Segment b

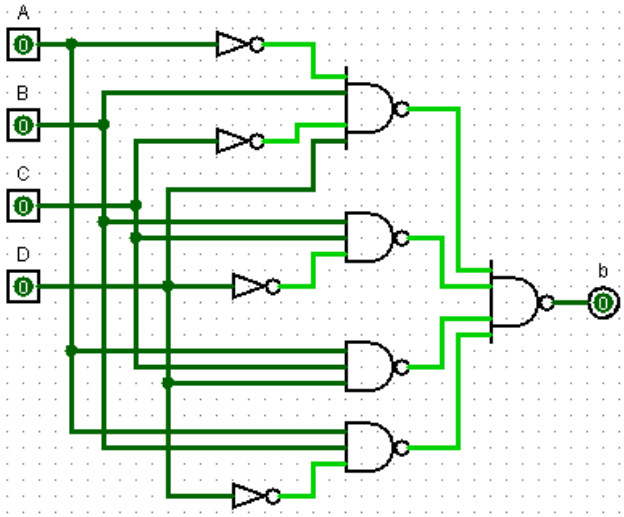


Figure 13: Block b Gate Level Schematic

A.3 Segment c

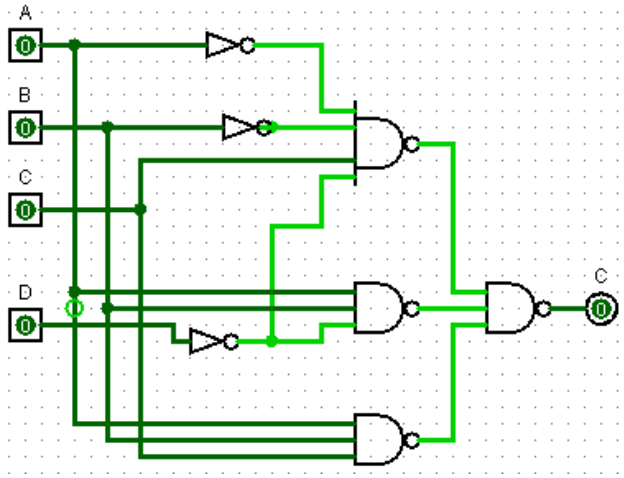


Figure 14: Block c Gate Level Schematic

A.4 Segment d

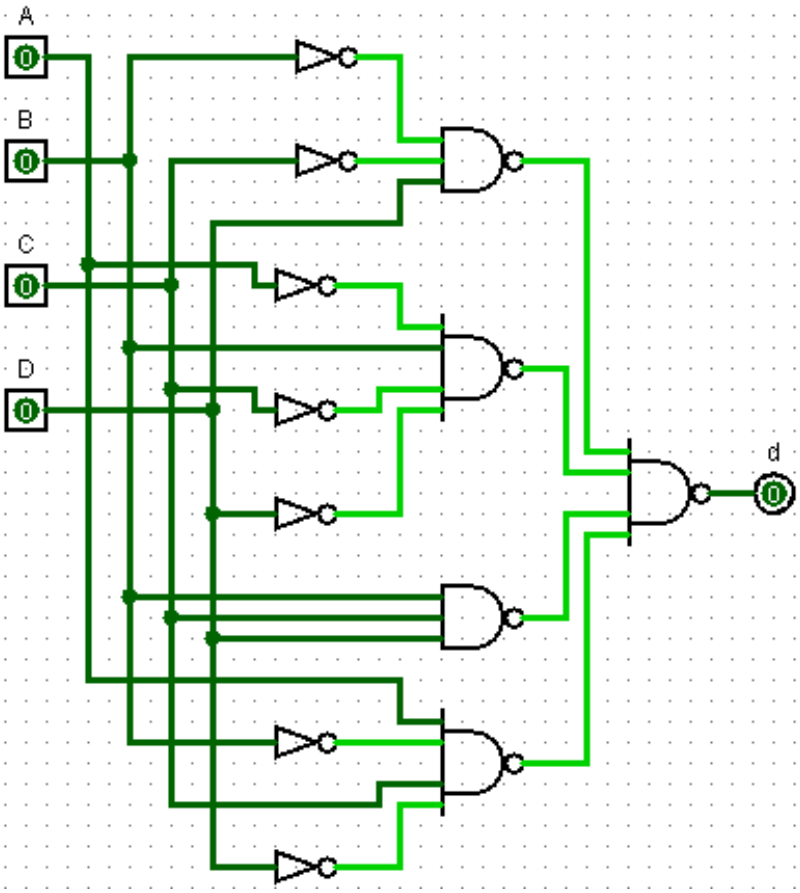


Figure 15: Block d Gate Level Schematic

A.5 Segment e

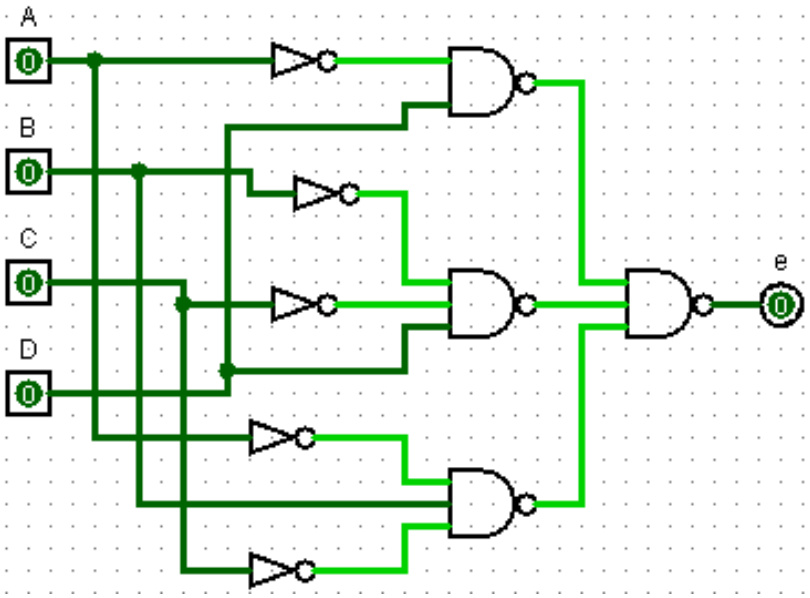


Figure 16: Block e Gate Level Schematic

A.6 Segment f

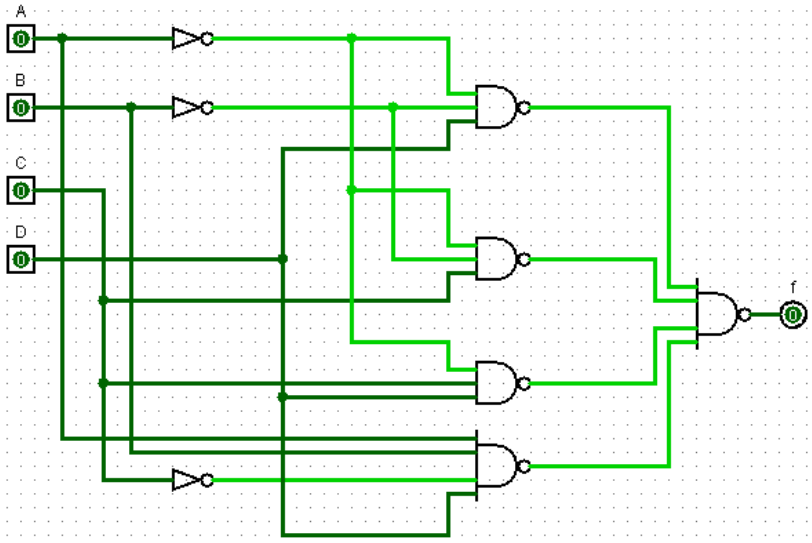


Figure 17: Block f Gate Level Schematic

A.7 Segment g

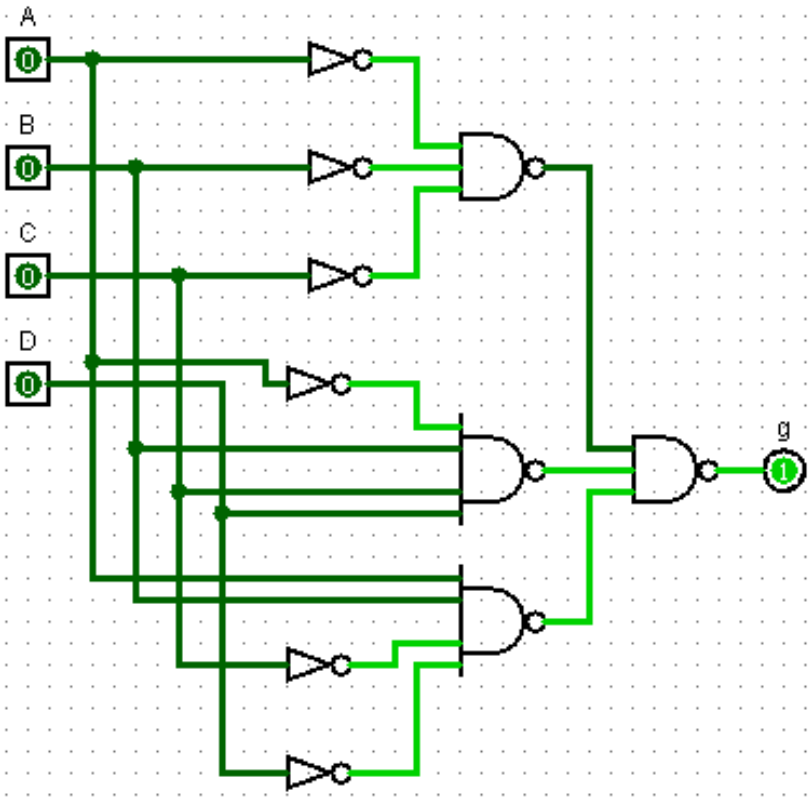


Figure 18: Block g Gate Level Schematic

B Logic Equations

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

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

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

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

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

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

$$g = \overline{\overline{(\bar{A}\bar{B}\bar{C})}\,\overline{(\bar{A}BCD)}\,\overline{(AB\bar{C}\bar{D})}} \tag{12}$$

C Magic VLSI Layouts

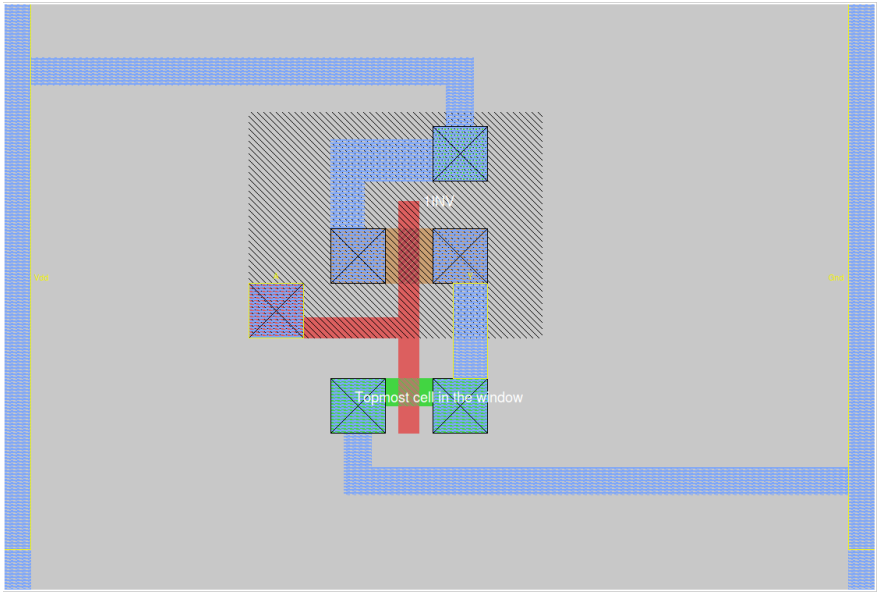


Figure 19: VLSI Layout for Inverter

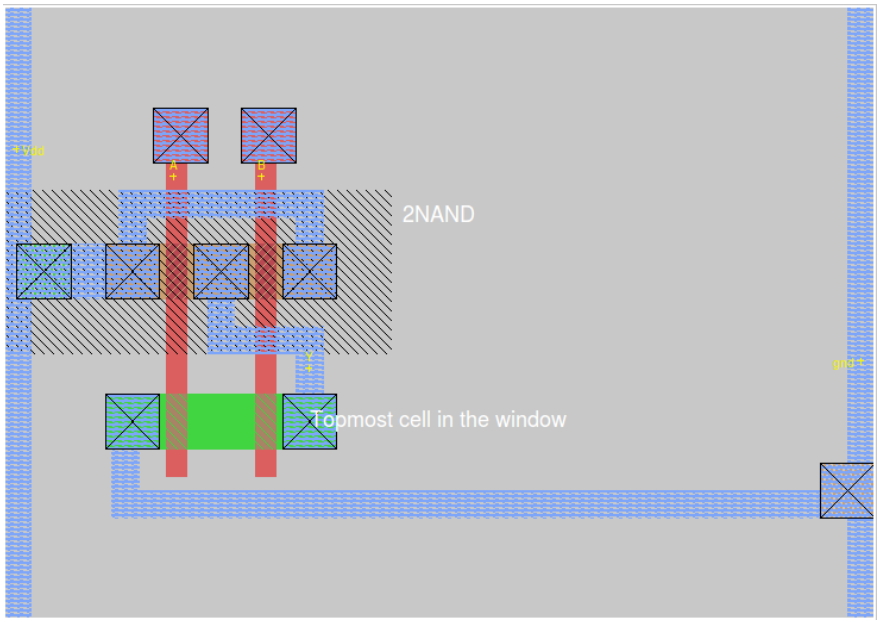


Figure 20: VLSI Layout for 2-Input NAND

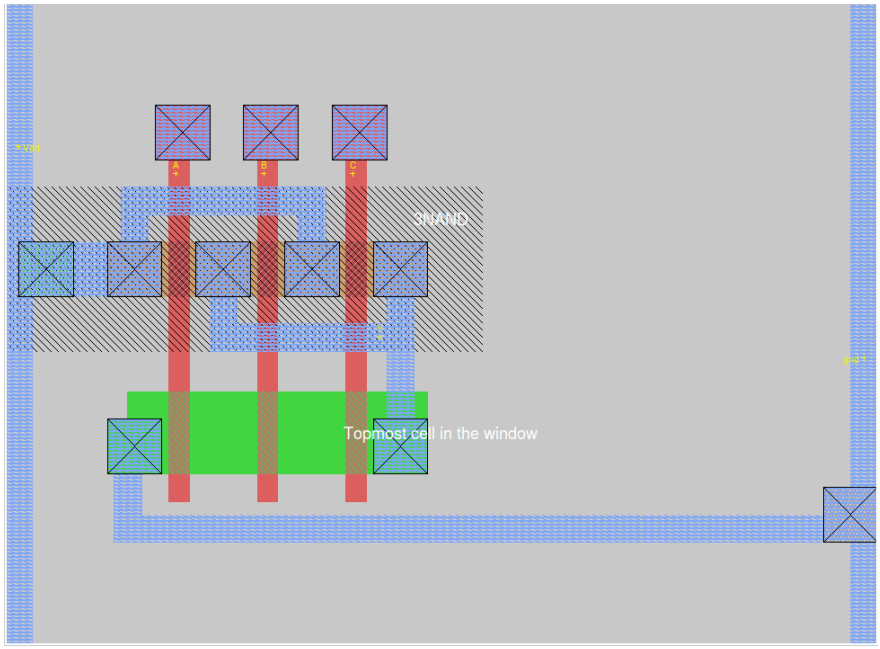


Figure 21: VLSI Layout for 3-Input NAND

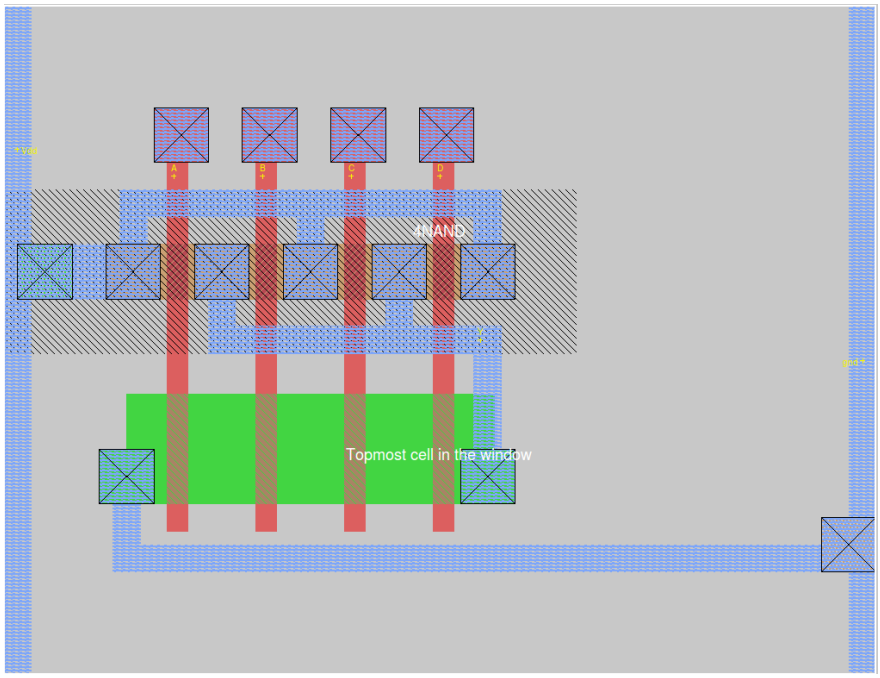


Figure 22: VLSI Layout for 4-Input NAND

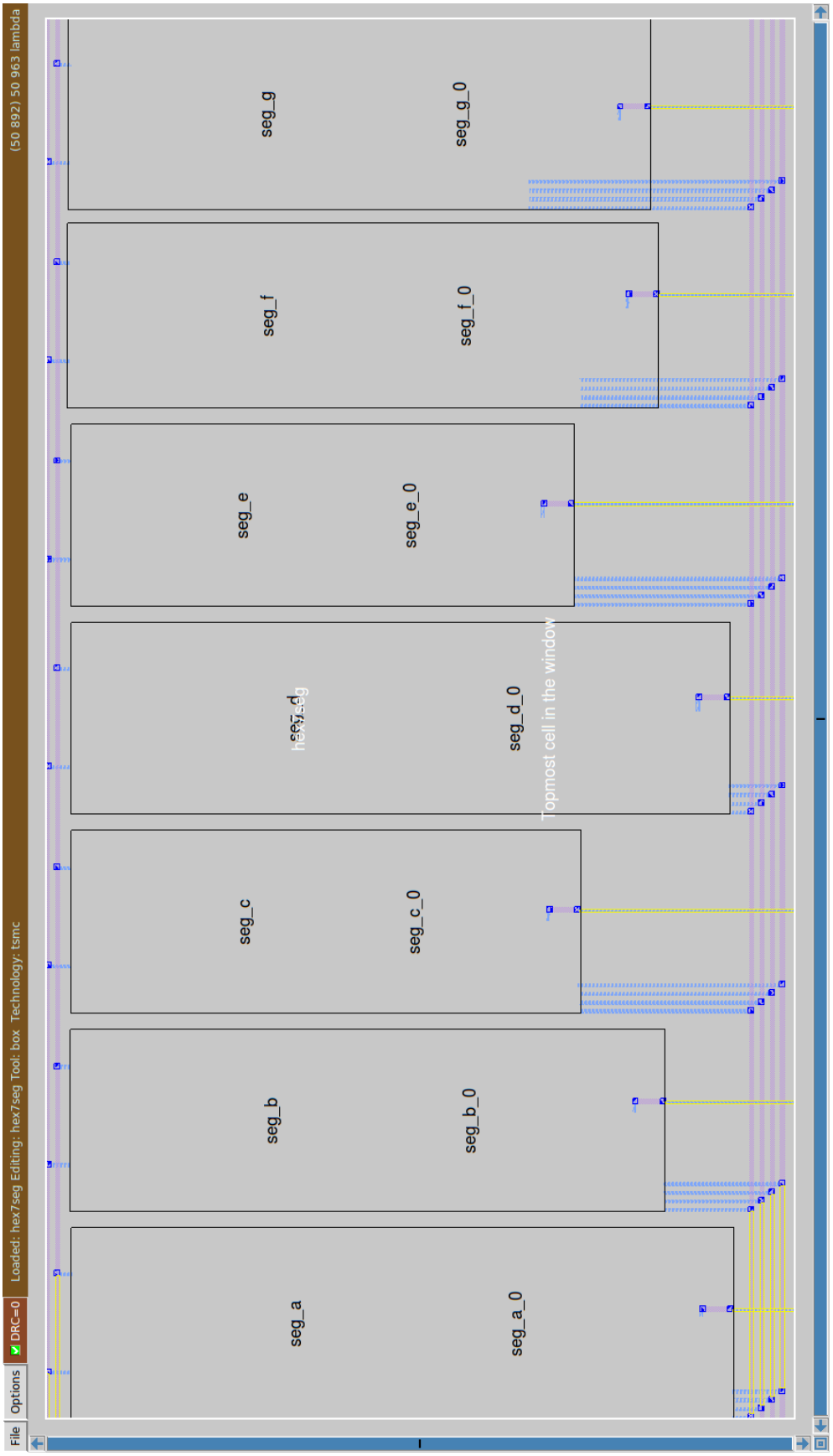


Figure 23: Unexpanded Full Circuit Layout

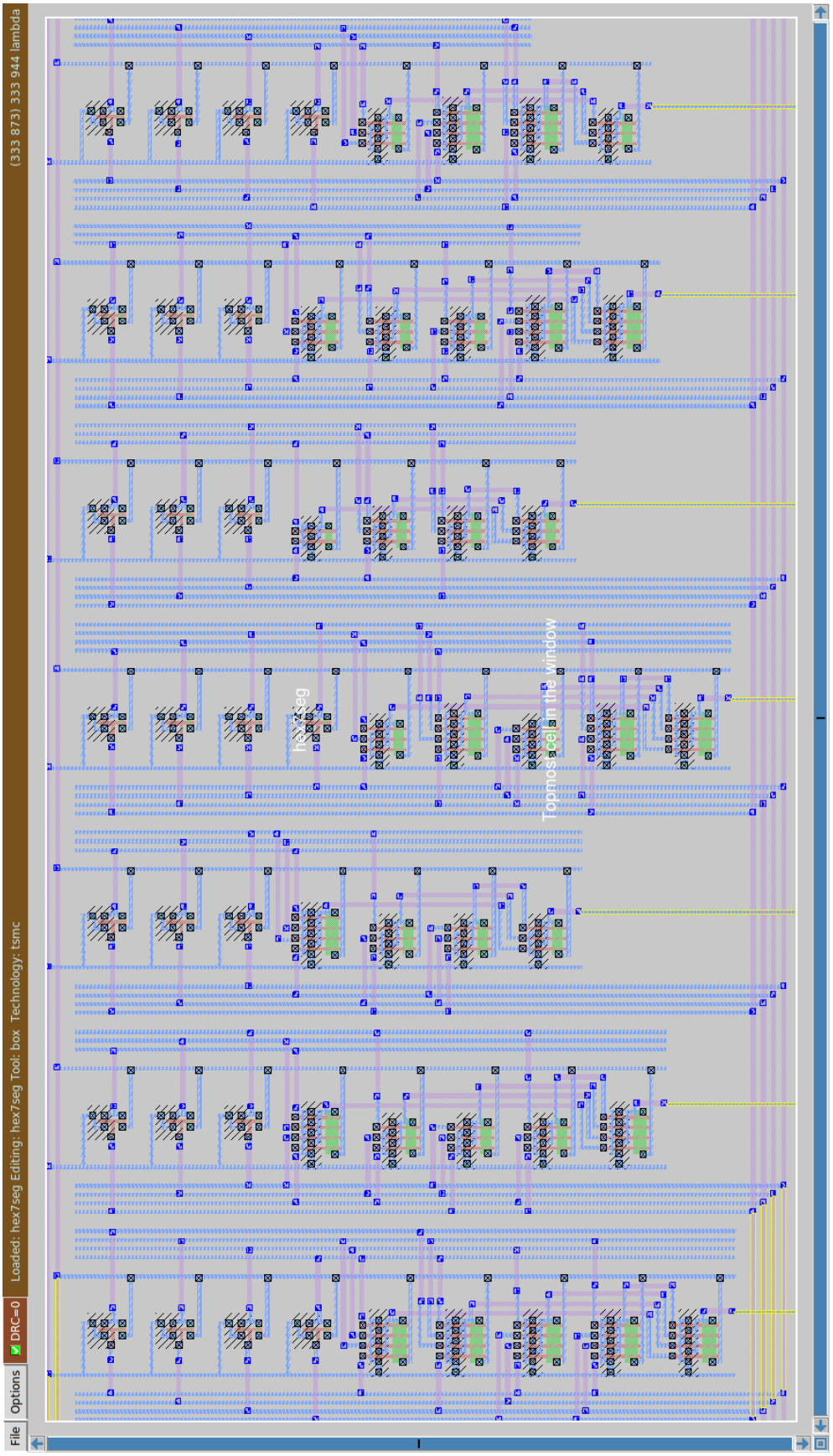


Figure 24: Expanded Full Circuit Layout

D SPICE Simulations

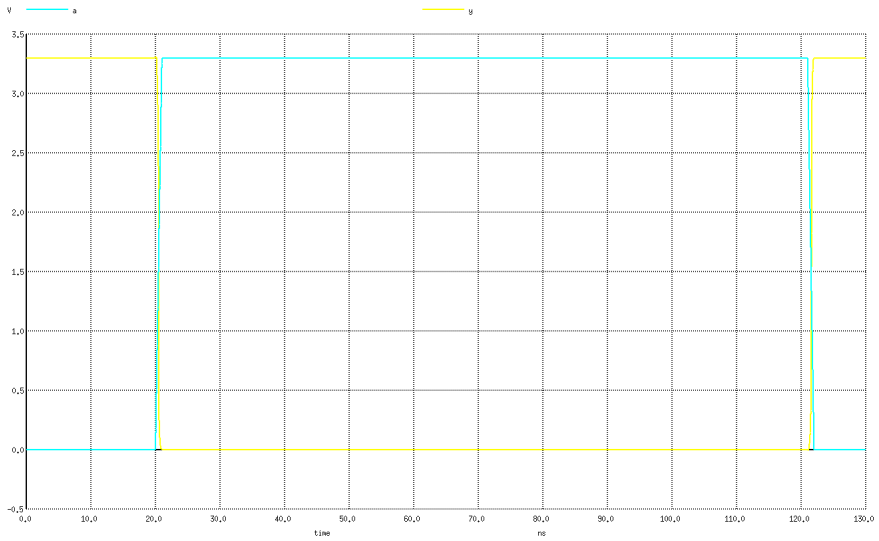


Figure 25: Spice Simulation of Unit Inverter

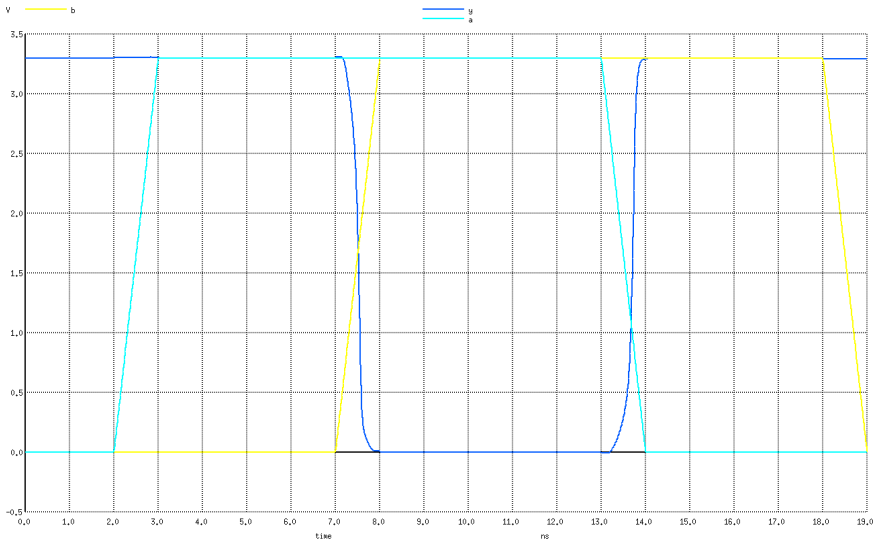
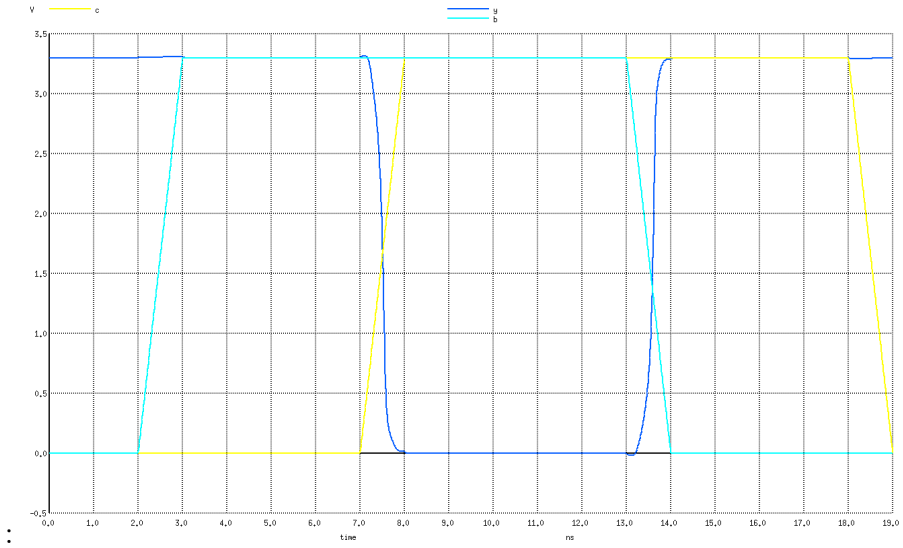


Figure 26: Spice Simulation of 2-Input NAND



:

Figure 27: Spice Simulation of 3-Input NAND

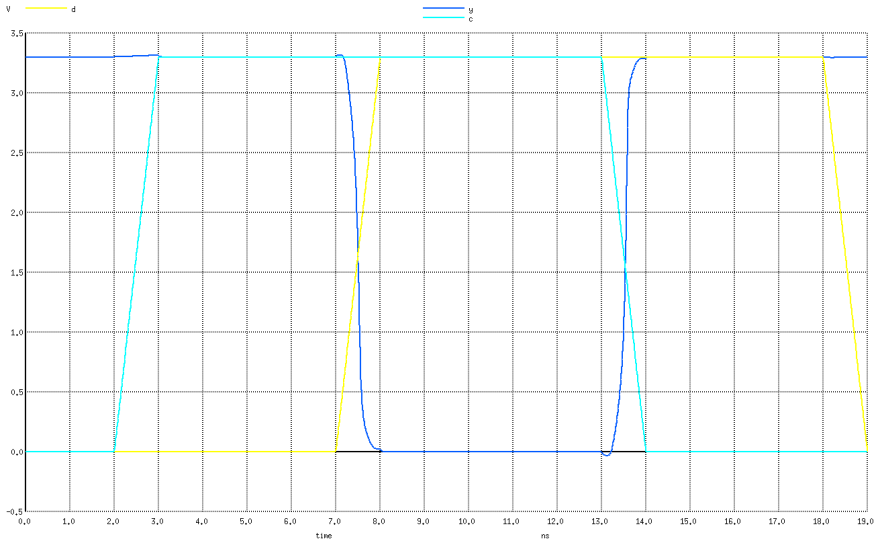


Figure 28: Spice Simulation of 4-Input NAND