EE 241B HW2 Writeup

Vighnesh Iyer

Contents

1 Extracting and Simulating a Synthesized Design

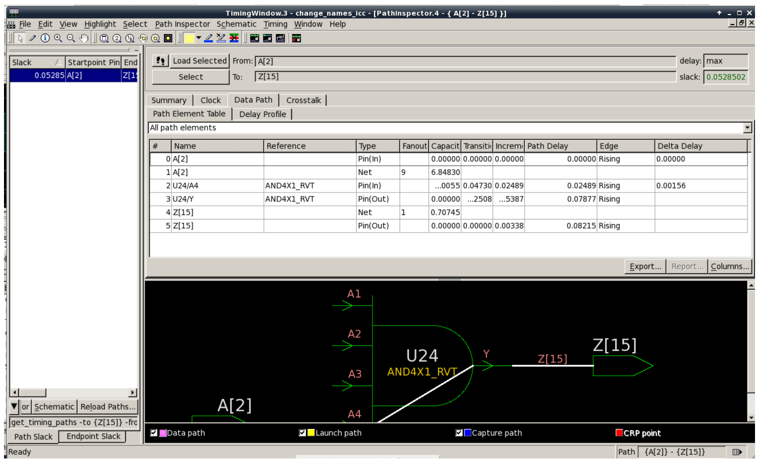
We start with an already placed and routed 4-bit decoder and we go through the flow of extracting parasitics, running LVS, and running SPICE simulations of the extracted schematic.

1.1 Delay of a Path

For the path from A[2] to Z[15] rise, what is the delay measured by IC Compiler, SPICE simulation without extracted parasitics, and SPICE simulation with parasitics?

- IC Compiler: 0.08215 ns = 82.15 ps

  This number was derived from the placed and routed design loaded into ICC and the timing path inspector was used to extract the relevant path and display its delay.


TimingWindow.3 - change_names_icc - [PathInspector.4 - { A[2] - Z[15] }]

Slack	Startpoint Pin	End
0.05285	A[2]	Z[15]

From: A[2] delay: max
To: Z[15] slack: 0.0528502

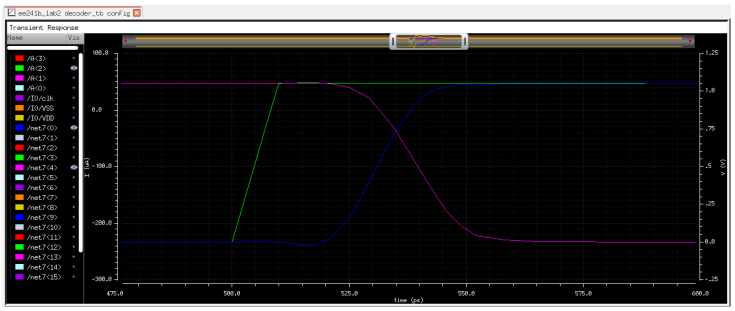
All path elements

#	Name	Reference	Type	Fanout	Capacit	Transiti	Increm	Path Delay	Edge	Delta Delay
0	A[2]		Pin(In)		0.00000	0.00000	0.00000	0.00000	Rising	0.00000
1	A[2]		Net	9	6.84830					
2	U24/A4	AND4X1_RVT	Pin(In)		...0055	0.04730	0.02489	0.02489	Rising	0.00156
3	U24/Y	AND4X1_RVT	Pin(Out)		0.00000	...2508	...5387	0.07877	Rising	
4	Z[15]		Net	1	0.70745					
5	Z[15]		Pin(Out)		0.00000	0.00000	0.00338	0.08215	Rising	

get_timing_paths -to {Z[15]} -frc
Path {A[2]} - {Z[15]}

- SPICE without parasitics: 26.297 ps

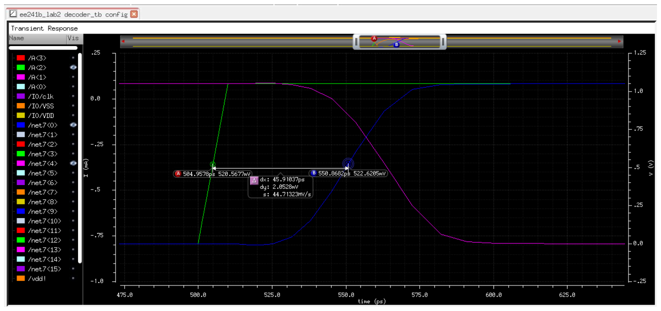
  This number was derived by importing the layout and Verilog netlist into Virtuoso, performing LVS to check that the schematic and layout were representative of the same circuit, and then using the schematic to perform a transistor level SPICE simulation of rising A[2] to rising Z[15].



In the figure you can see the rise of A[2] (green) followed by the rise of Z[15] (blue) and the fall of Z[11] (pink). The delay in this part is significantly lower than the delay as reported by ICC; it could have to do with parasitics, which will be simulated in the next part.

- SPICE with parasitics: 46.91 ps

  This number was derived using the same testbench, but using the 'starrc' parasitic extracted view of the decoder schematic rather than the raw schematic with ideal circuitry. We can see that the delay increased by almost 2x due to parasitics.

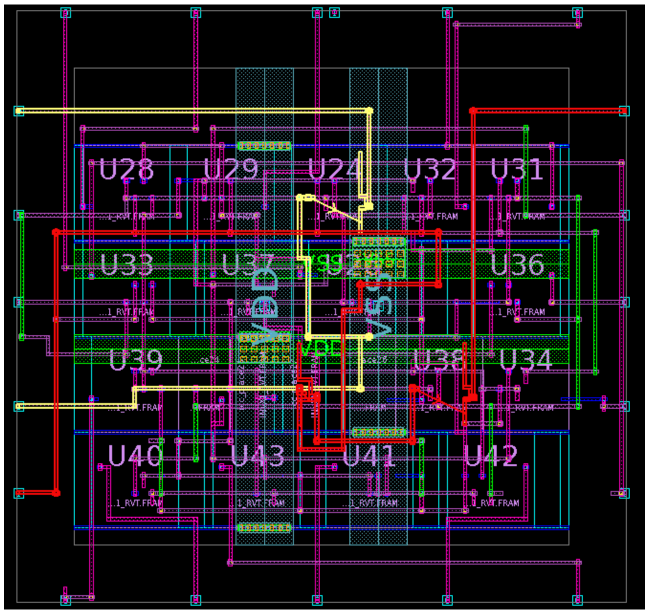


This figure has the same color scheme as the previous one. The path delay is still roughly half of what ICC reported. This probably has to do with some inherent pessimism in the tool.

1.2 ICC Critical Path + Special Path

We display the critical path in ICC and also highlight the special path noted above from the rise of A[2] to the rise of Z[15].

The critical path is from rising A[3] to falling Z[5], and it has a negative slack of -0.00263 ns. It is highlighted in red. The special path above is highlighted in yellow.



1.3 Power Estimate Accuracy Analysis

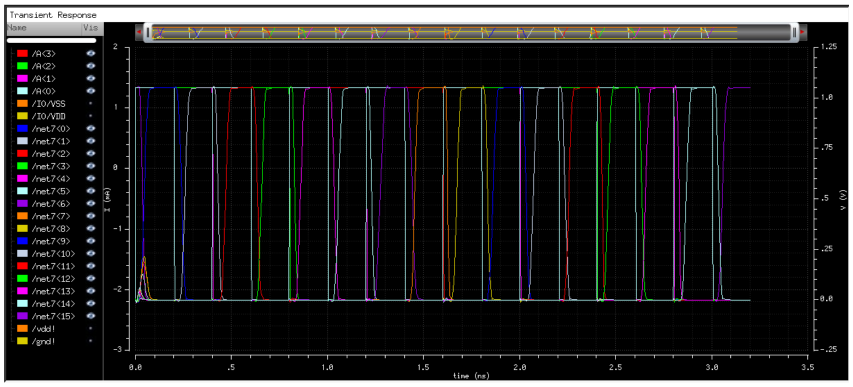
We run the functional testbench that counts from A=0000 to A=1111 and report the power measured by Primetime, the SPICE simulation, and the mixed-signal simulation.

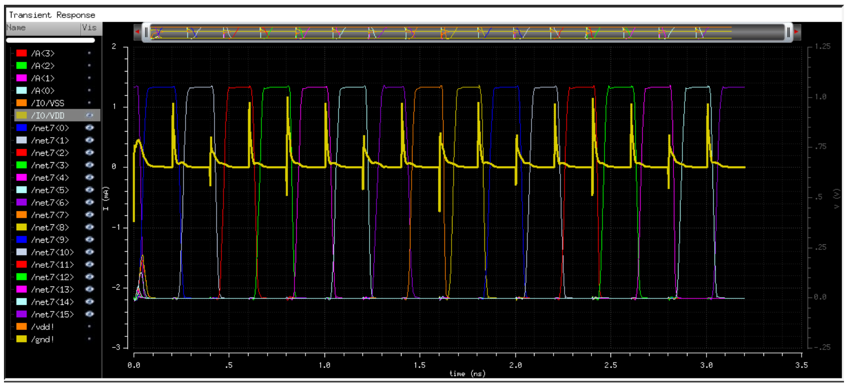
- Primetime: 3.97e-5 (switching), 4.35e-5 (int), 1.90e-6 (leakage), 8.51e-5 (total) = 85 uW

We run a post-PAR simulation using VCS and extract the switching activity SAIF file from the output of the testbench. This file and the Verilog netlist is fed into Primetime which estimates the power usage for the testbench.

- SPICE: 73.46 uW

  We re-create the Verilog testbench in SPICE by using pulsed voltage sources for the `A<3:0>` input to the decoder and we measure the average power consumption by integrating the current drawn from the `VDD` supply during the transient SPICE simulation.



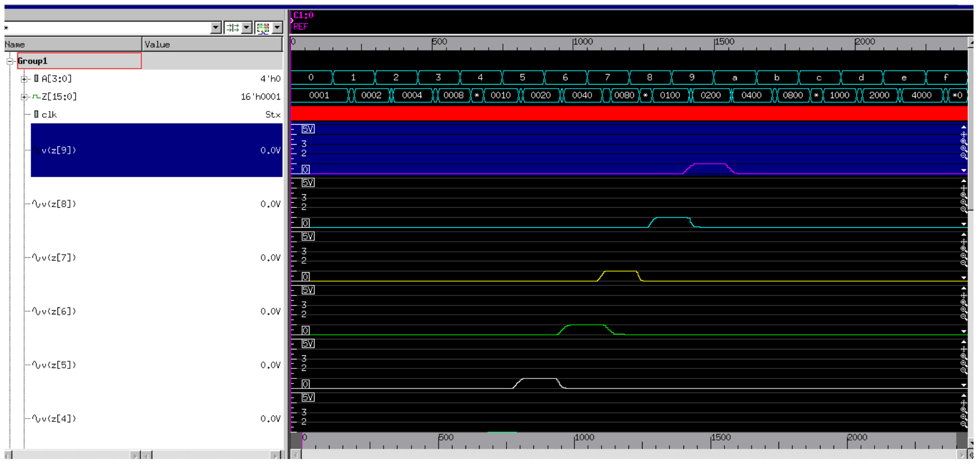


  The first plot shows the switching activity of both the inputs and the outputs of the decoder. The second plot shows the current drawn from the `VDD` input (bold yellow line) of the decoder and each output bit of the decoder going high as its respective code is applied to the decoder's input. The power consumed is measured by using the `average` function on the transient current on the decoder's `VDD` port and multiplying by the supply voltage of 1.05 V.

  The power consumed as reported by SPICE with parasitics is a little lower than what was reported by PrimeTime.

- Mixed-Signal: 75.79 uW

  We then set up a mixed-signal testbench by exporting the SPICE netlist of the extracted decoder. We create a wrapper for the decoder in Verilog and instantiate it in a testbench. The testbench is used to drive the inputs and check the outputs of the decoder, while the actual function of the decoder is simulated at the transistor level. This combines the best parts of the Primetime and SPICE flows by letting us write stimulus at a high-level (in Verilog), but still retain the precision of a transistor level simulation using SPICE.



This snapshot from DVE shows the digital stimulus applied to the decoder and the SPICE simulated decoder's analog outputs.

1.4 Voltage Scaling Power Estimates

We run the mixed-signal simulation at 1.05V, 0.8V, 0.6V, and 0.4V and measure the average power for each voltage. We then convert average power to energy/op in terms of J/op and uW/Mhz. These results are compared to the theoretically predicted active energy savings for voltage scaling based on the 1.05V result.

The (fake) clock period will have to increase for lower supply voltages to compensate for increased delay.

We theoretically predict that the dynamic power scales according to this equation:

$$P_{dyn} = C_{eff} \cdot V_{DD}^2 \cdot f_{clk} \tag{1}$$

with C_{eff} being fixed for each supply voltage.

Here are my summarized results that were produced by varying the supply voltage to the decoder and measuring average power in the same manner. The number of operations is $2^4 = 16$ and $P_{scale,pred}$ is the predicted power scaling, P_{pred} is the predicted power consumption, and P_{sim} is the mixed-signal simulated actual power consumed. t_{sim} is used to measure the time of the testbench. The clock period was changed manually until signal settling was sufficiently fast.

V_{DD}	T_{clk}	$P_{scale,pred}$	P_{pred}	P_{sim}	t_{sim}	J/op	uW/Mhz
1.05 V	0.15 ns	1.0	75.79 uW	75.79 uW	2.4 ns	11.37 pJ	1.14e-8
0.8 V	0.17 ns	0.5122	38.82 uW	50.27 uW	2.72 ns	8.546 pJ	8.55e-9
0.6 V	0.5 ns	0.098	7.4274 uW	13.56 uW	8 ns	6.78 pJ	6.78e-8
0.4 V	decoder doesn't work (5ns)	0.0044	0.333 uW	10 uW	80ns	50 pJ	5e-8

2 FF and Latch Based Timing

3 Variability and Timing Simulations

We begin by recording the nomimal (TT) FO4 low to high and high to low delays of an inverter. We size the inverters for minimum average delay.

Using SPICE and the 32nm LP model with a 1.05V supply, we find the optimal width of the PMOS transistor that minimizes the propagation delay ($t_p = (t_{pHL} + t_{pLH})/2$) for a CMOS inverter. We fix the NMOS transistor width at 100nm and the lengths of both transistors are fixed at the process' 32nm minimum.

In order to solve this problem with simulation, I used the F04 (fanout of 4) method of inverter delay characterization. I designed a simulation based on the paper "The Fanout-of-4 Inverter Delay Metric" by Harris, Horowitz, et al. The circuit setup from the paper is shown below.

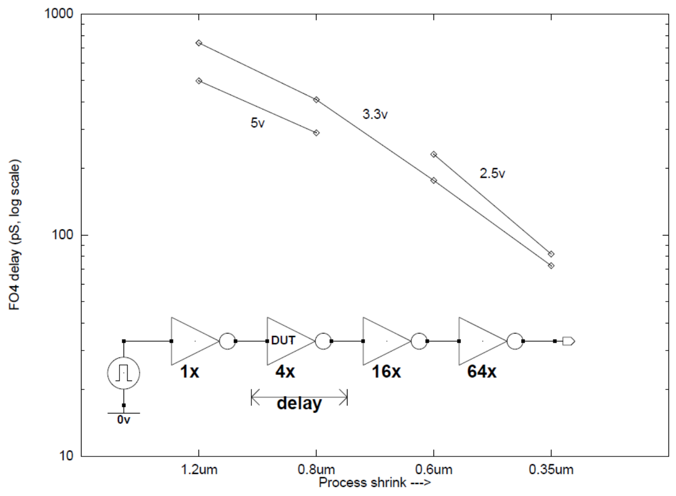
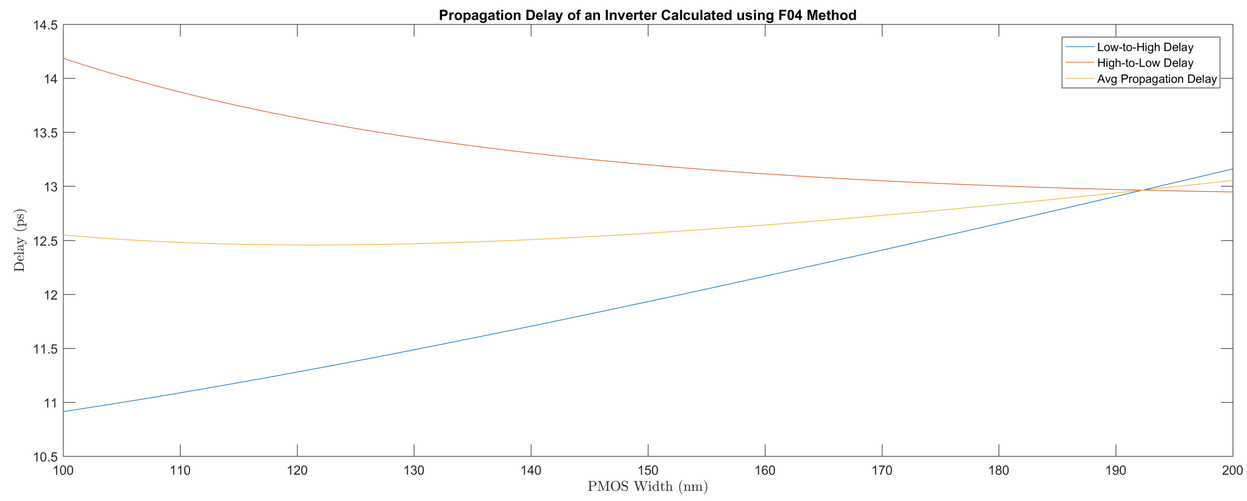


Figure 1: Fanout-of-4 inverter delays

I swept the width of the PMOS from 100n to 200n while performing transient simulations. I measured the propagation delays (high-to-low and low-to-high) using '.measure' statements in SPICE. Propagation delay was measured from 50% of the input to 50% of the output. The SPICE file is provided in the appendix. The results are summarized below:



We find that the average propagation delay t_p is minimized when the PMOS width is 121nm and the minimal delay is 12.4560 ps. The PMOS width that results in an equal low-to-high and high-to-low delay is 192.3nm and the equal delay is 12.96 ps. In conclusion, for the instructions in this part:

$$\begin{aligned} W_{pmos,optimal} &= 121 \text{ nm for } W_{nmos} = 100 \text{ nm} \\ t_{p,optimal} &= 12.4560 \text{ ps} \end{aligned}$$

A PMOS/NMOS DC Characterization SPICE Sim