Assignment 4 - Place and Route & Domino Logic

ELEC 402

Isabelle André – 12521589

1.0 Project Description

This project consisted in using the Cadence Design tools to synthesize a Verilog DNSLookup state machine previously designed in Project 1 using a GPDK 45nm standard cell library. The synthesized design is then laid out and verified using the Cadence Innovus tool by auto-routing. In the next part of this project, domino logic is explored by determining voltage reduction under worst case charge sharing conditions. Minimum capacitance and delays are calculated for a transmission gate circuit, and static and dynamic power consumption is solved analytically and graphically. Finally Interconnect properties are explored by computing the resistance, capacitance, and delay for a distributed RC wire.

2.0 Cell Library Layout

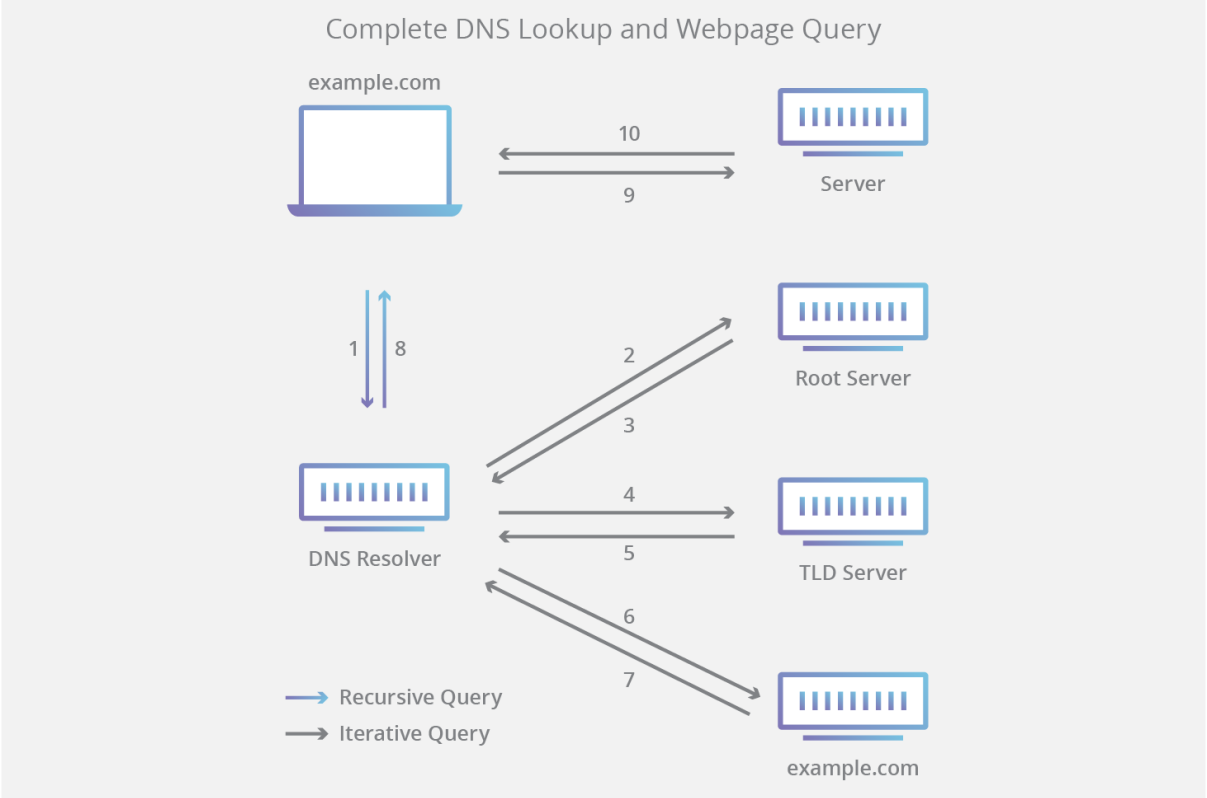
2.1 FSM Design Description

A DNS Lookup process begins with the user entering a web address into a browser such as google.ca, received by a DNS recursive resolver. The resolver uses this address to query a DNS root nameserver, which in turn responds with the address of a TLD DNS server, such as .ca. Once the resolver then queries to the .ca TLD server, and receives the IP address of the domain's nameserver for google.ca. The resolver sends another query to the domain's nameserver, which returns the corresponding IP address for google.ca. Finally, the DNS server is able to return the corresponding IP address to the web browser for the requested domain.

Once the client browser receives a response, it makes an HTTP request to the IP address. The server at the IP address returns the webpage data and components to render the website.

The state machine includes a caching state in which the browser is able to cache the IP address of the latest website accessed. If the same website is accessed on the next request, the FSM skips the DNS lookup process, and is able to directly make an HTTP request to the cached IP address. This is a very simplified approach to simulate the caching abilities of browsers to reduce request latency, and does not include OS caching or caching server addresses.

Figure 2.1 describes this process and the iterative nature of server requests from the DNS resolver. Note that this is not a complete state transition diagram, and simply a representation of the servers and client interactions.



Cloudflare, What is DNS?, 2022, https://www.cloudflare.com/learning/dns/what-is-dns/
Figure 2.1: Complete DNS Lookup and Webpage Query

2.2 States Description

- **IDLE:** Initial state. If a client request arrives, reset execution time counter
- **CLIENT START:** Start execution time counter and begin to process request
- **CLIENT RESOLVER REQ:** Browser cache is queried to see if web address matches a cached IP address. If so, the state machine skips to CLIENT SERVER REQ. If not, the client query is sent to the DNS Resolver for a DNS Lookup.
- **RESOLVER ROOT REQ:** Resolver request is sent to a Root Nameserver. A query bit is enabled to allow a mock TLD address to be processed and returned.
- **ROOT RES:** The mock TLD address is generated and returned.
- **RESOLVER TLD REQ:** The Resolver sends a request to the TLD server. A query bit is enabled to allow a mock Domain Nameserver address to be processed and returned.
- **TLD RES:** The mock Domain Nameserver address is generated and returned.
- **RESOLVER DOMAIN REQ:** The Resolver sends a request to the Domain Nameserver. A query bit is enabled to allow a mock web IP address to be processed and returned.
- **DOMAIN RES:** The mock web IP address is generated and returned.
- **RESOLVER RES:** The DNS Resolver responds to the client with the IP address of the domain requested. The IP address was found through DNS Lookup and resolved.

- **CLIENT SERVER REQ:** The client browser makes an HTTP request to the IP address. If the IP address was found through DNS Lookup, use this address, if not, use the cached address. A query bit is enabled to allow mock web data to be processed and returned.
- **SERVER RES:** The mock web data is generated and returned.
- **CACHING:** The last used IP address is cached in the browser. The execution time counter is terminated and outputted, and the client request has been resolved. The state machine returns to the IDLE state until the next client request arrives.

2.3 Inputs Description

- **clk:** Input clock signal.
- **rst:** Input reset signal.
- **client req:** Flag indicating an incoming client request.
- **web addr:** Mock client web address requesting corresponding IP address.

2.4 Outputs Description

- **webpage idx out:** Mock web page data output corresponding for website rendering.
- **tld addr out:** Mock Top Level Domain address.
- **domain ip out:** Mock Domain Nameserver IP address.
- **web ip out:** Mock web IP Address corresponding to initial client query.
- **exec time:** Total execution time for DNS Lookup and caching process, simulated by counting cycles.
- **ip resolved:** Mock web IP address was found during DNS Lookup.
- **client res:** Web data has been returned and rendered, state machine is complete.

2.5 Modules

- **DNSLookup:** Main state machine.
- **ExecCounter:** A simple up-counter to count mock execution time in cycles.
- **WebAddrToTLDAddr:** A simple shifting module to generate a mock TLD address returned to the Resolver. This simulates a Root Nameserver, returning the corresponding TLD address to a Web Address.
- **TLDAddrToDomainIP:** A simple XOR module to generate a mock Domain address returned to the Resolver. This simulates a TLD Server, returning the corresponding Domain address to a Web Address.

- **DomainIPToWebIP:** A module of combined operations to generate a mock IP address returned to the Resolver. This simulates a Domain Nameserver, returning the corresponding IP address to a Web Address.
- **WebIPToWebdata:** A 4 to 16 Decoder module generating mock web data upon an HTTP request to an IP address. This simulates the web data being returned and rendered in a browser. In a normal rendering process, multiple HTTP requests would be made until the website is fully rendered.

2.6 Testbench

Figure 2.2 describes the input and output connections of the DNSLookup top level module DUT and the testbench. This testbench stimulates a clock signal every 5 ns, a reset signal, a client request flag signal, and a mock web address chosen arbitrarily.
The objective of this testbench is not only to check the correctness of the output signals, but also to observe the state transitions when a client requests multiple addresses consecutively as the functioning of the browser caching mechanism can be verified. the Verilog code for the testbench can be seen in Appendix A.

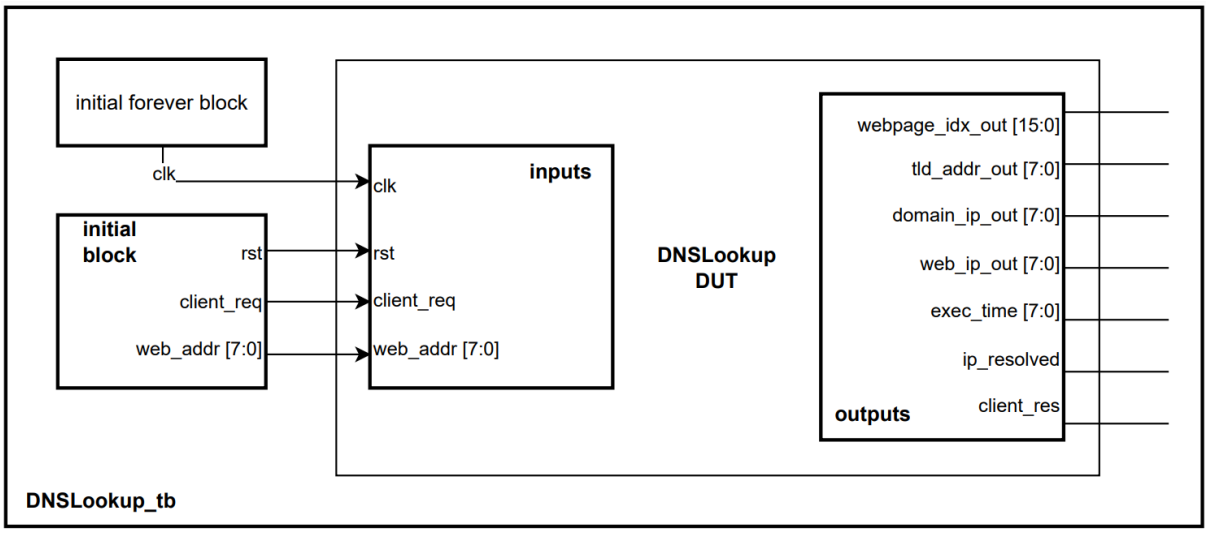


Figure 2.2: DNSlookup Testbench and Top Level Module Connections

2.7 Testing Procedures

A SystemVerilog Testbench was designed in order to test the FSM design and simulate various cases. In our test bench, three mock web addresses were simulated to test the caching and address mapping behaviors of the DNS Lookup state machine.

The first stimulated client query is an arbitrary Web Address (A) represented by 8 bits. In Figure 2.3 a), the query reaches the DNS Resolver, which queries the Root, TLD, and Domain servers one by one until the correct IP address is found in Figure 2.3 b). The IP Address is returned to the client, initiating an HTTP request, then cached for future use once webdata is returned and client query is resolved. The total execution time is counted to be 12.

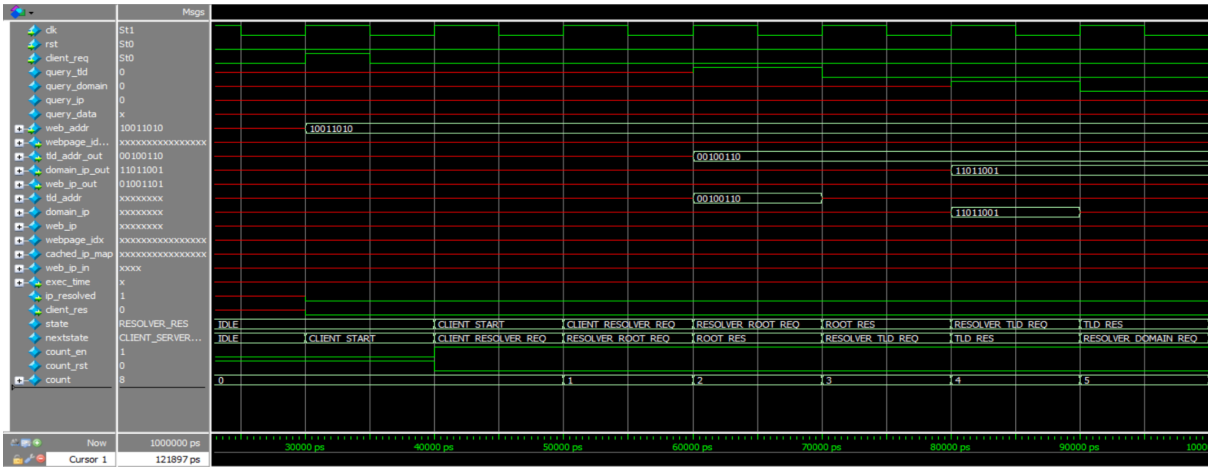


Figure 2.3 a): Web Address (A) is queried by client. Servers return corresponding addresses.

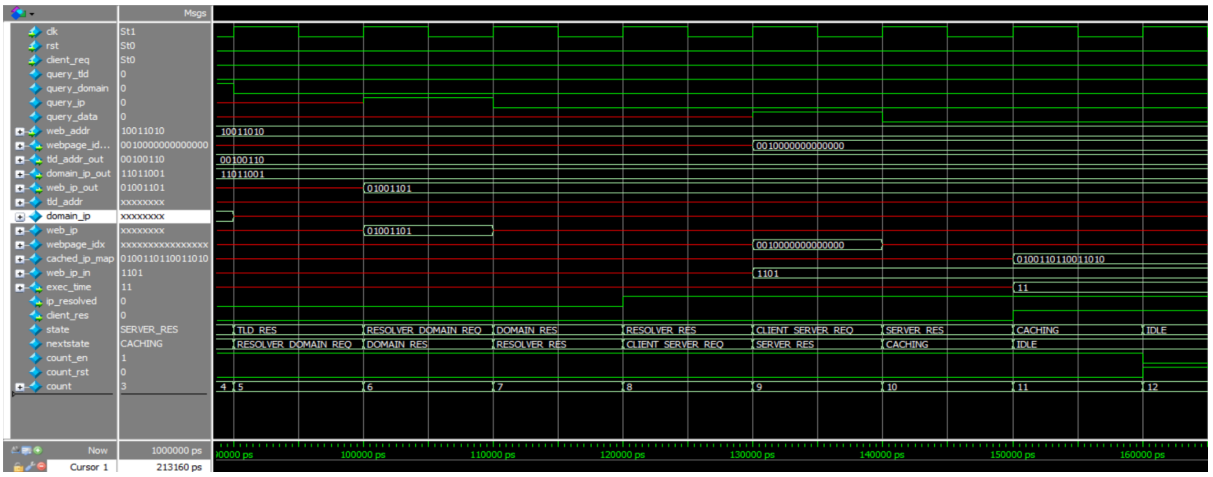


Figure 2.3 b): Corresponding IP Address to Web Address (A) is resolved, cached, and returned to client.

Once again, we use the same client web query to check that its IP address has been successfully cached. The query reaches the DNS Resolver, which maps the web address to a cached IP address as shown in Figure 2.4. The state machine directly moves to the HTTP request state, before caching again and resolving the client request as before. The total execution time is counted to be only 5 this time.

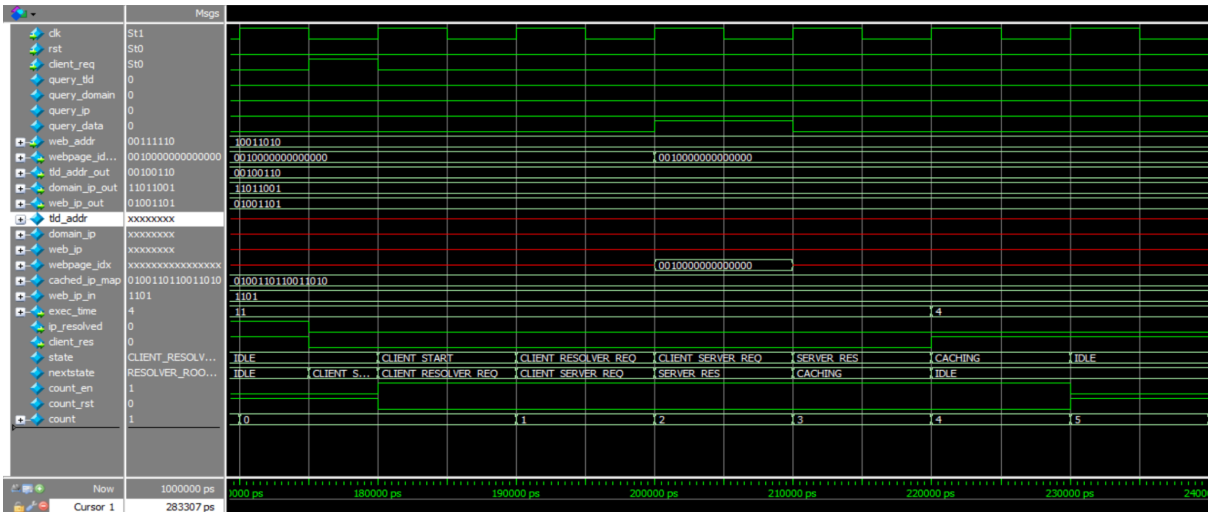


Figure 2.4: Web Address (A) is queried again by client. The client can directly query the server to render the page.

This next client query is a different arbitrary web address from the last query, represented by 8 bits. In Figure 2.5 a), the query reaches the DNS Resolver. As Web Address (B) is not cached by the browser. The Resolver proceeds to query the DNS server similarly to Figures 62.5 a) and b). A new IP address is cached, and the total execution time is 12.

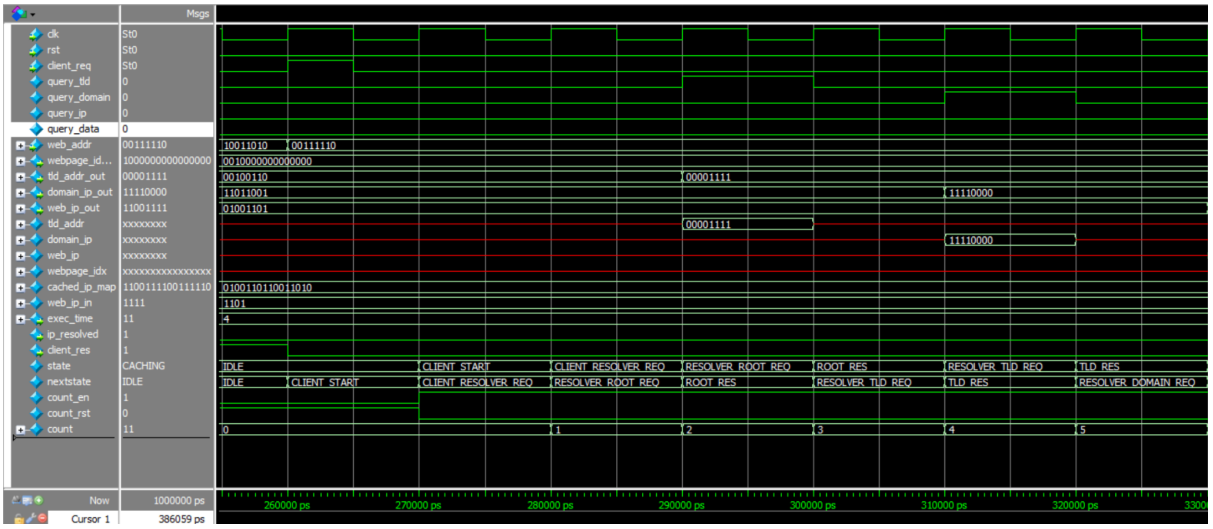


Figure 2.5 a): Web Address (B) is queried by client. Address is not cached, servers return corresponding addresses.

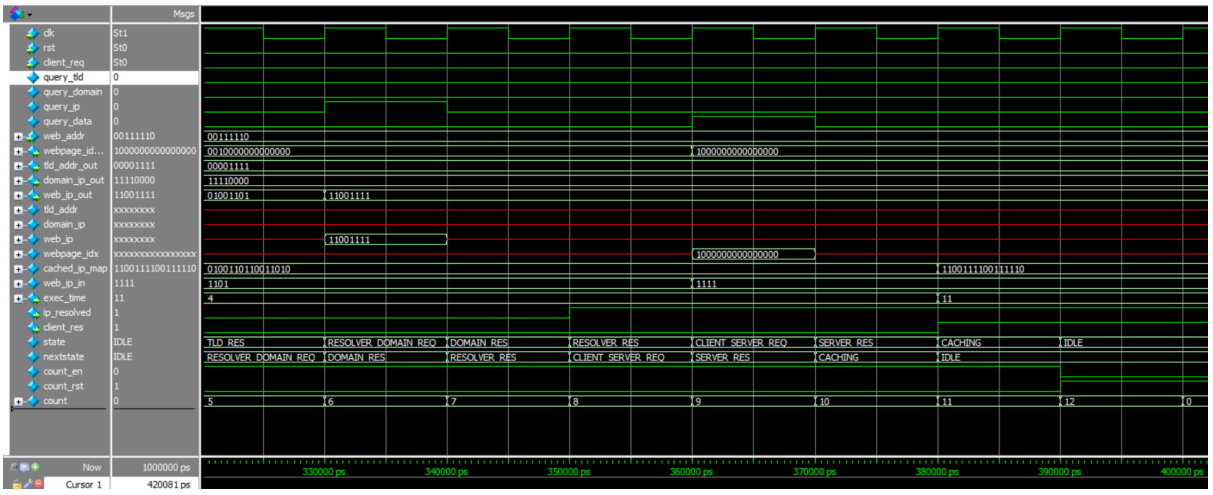


Figure 2.5 b): Corresponding IP Address to Web Address (B) is resolved, cached, and returned to client.

2. 8 Cadence Layout

Once the Verilog code was simulated and tested on modelsim, the mapped Verilog file was synthesized then auto-routed by running Innovus. The 45 nm PDK was used to synthesize Verilog code and run PnR. While running Innovus Place and Route, the input, output, VSS, and VDD pins were placed on the layout and power nets were configured.

After adding VSS and VDD rings, setting up power nets, and generating the layout, the layout in Figure 2.6 is obtained.

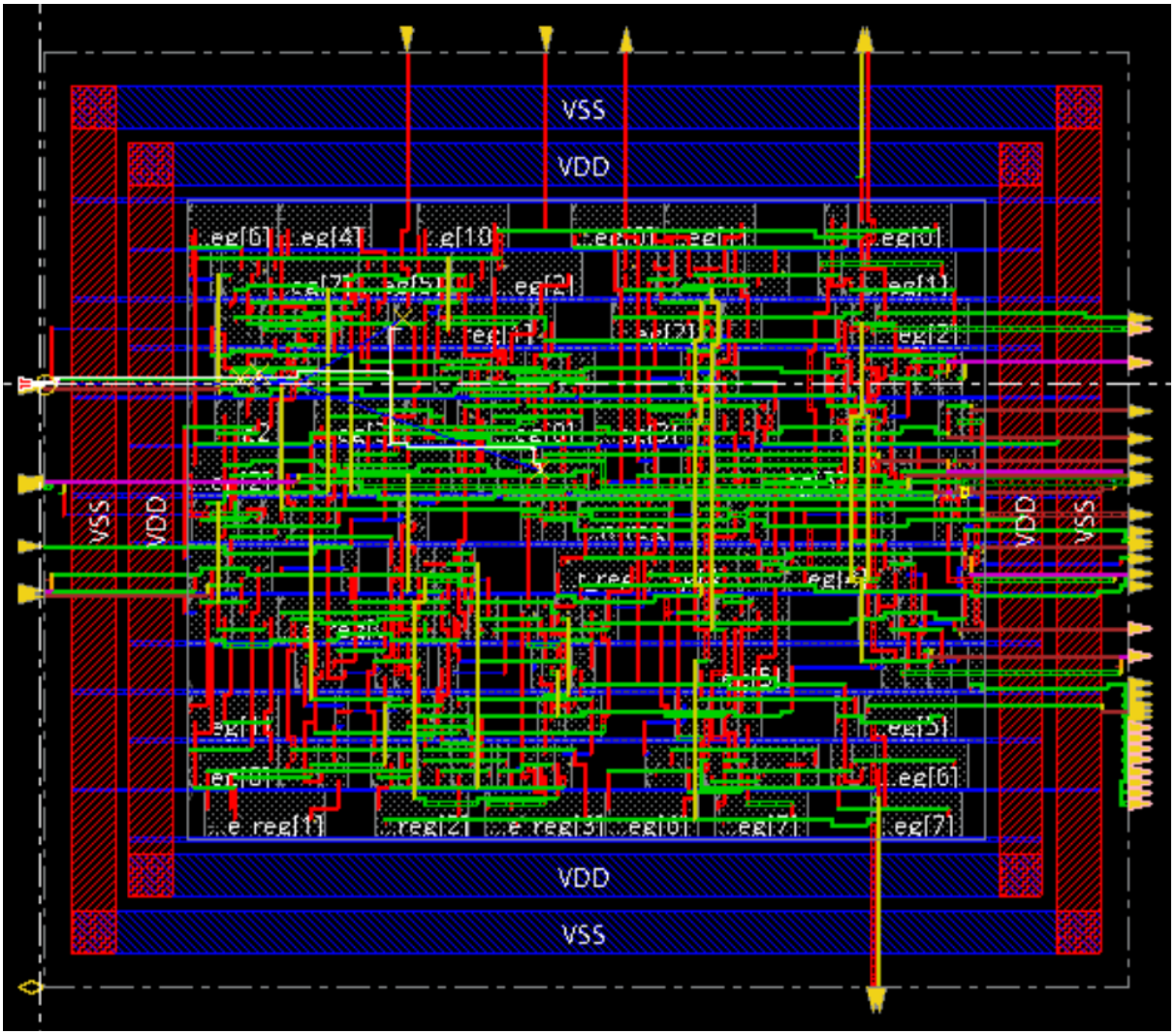


Figure 2.6: Innovus Place and Route

After exporting the layout to GDS and saving the Verilog file as a netlist, we begin the Post Place and Route on Virtuoso. After importing the GDS file, additional cells are generated as part of the layout, shown in Figure 2.7.

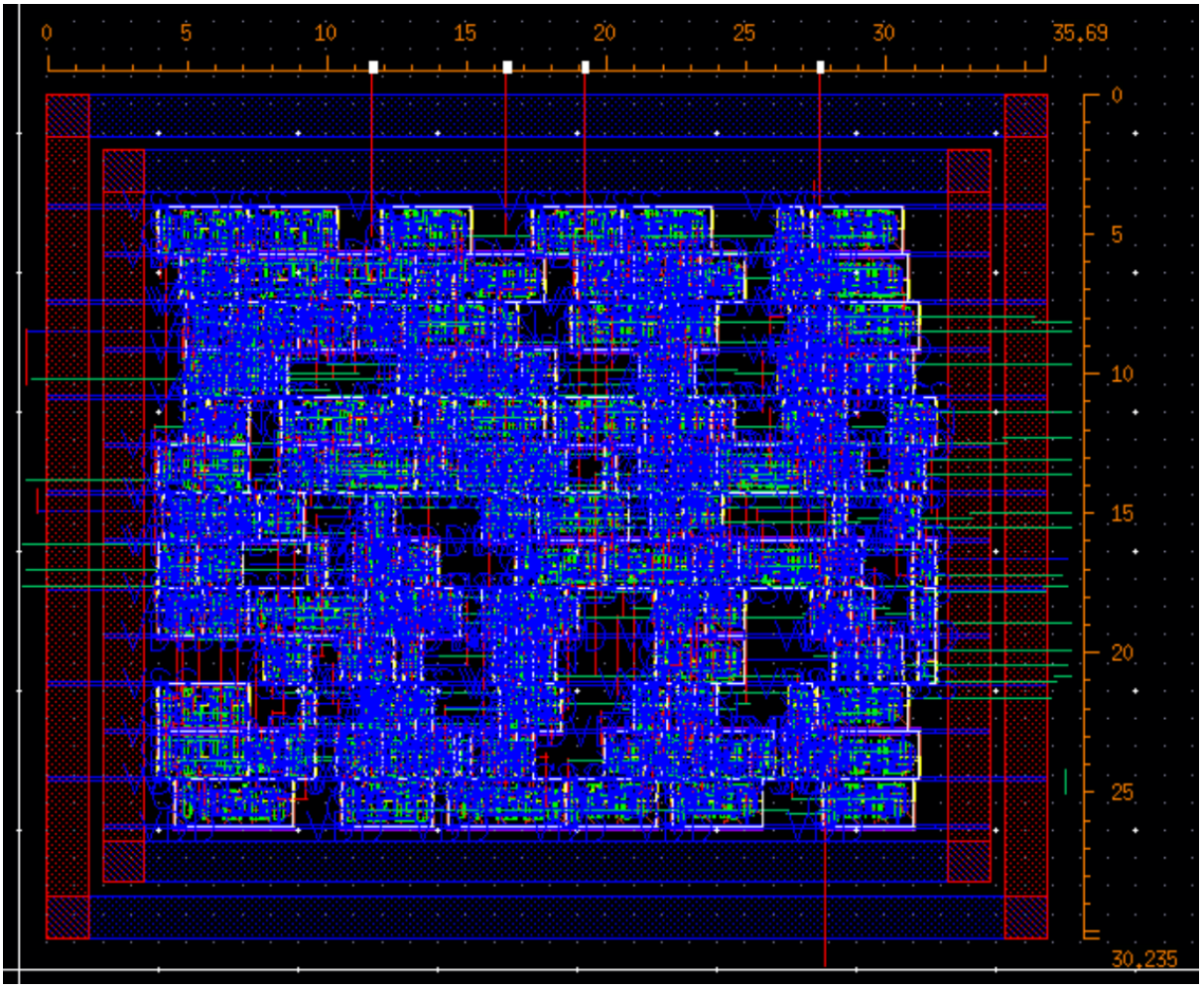


Figure 2.7: Virtuoso Place and Route Cells

To simulate the layout, we must import the generated Verilog file. A schematic as shown in Figure 2.8 and a symbol for the layout are generated.

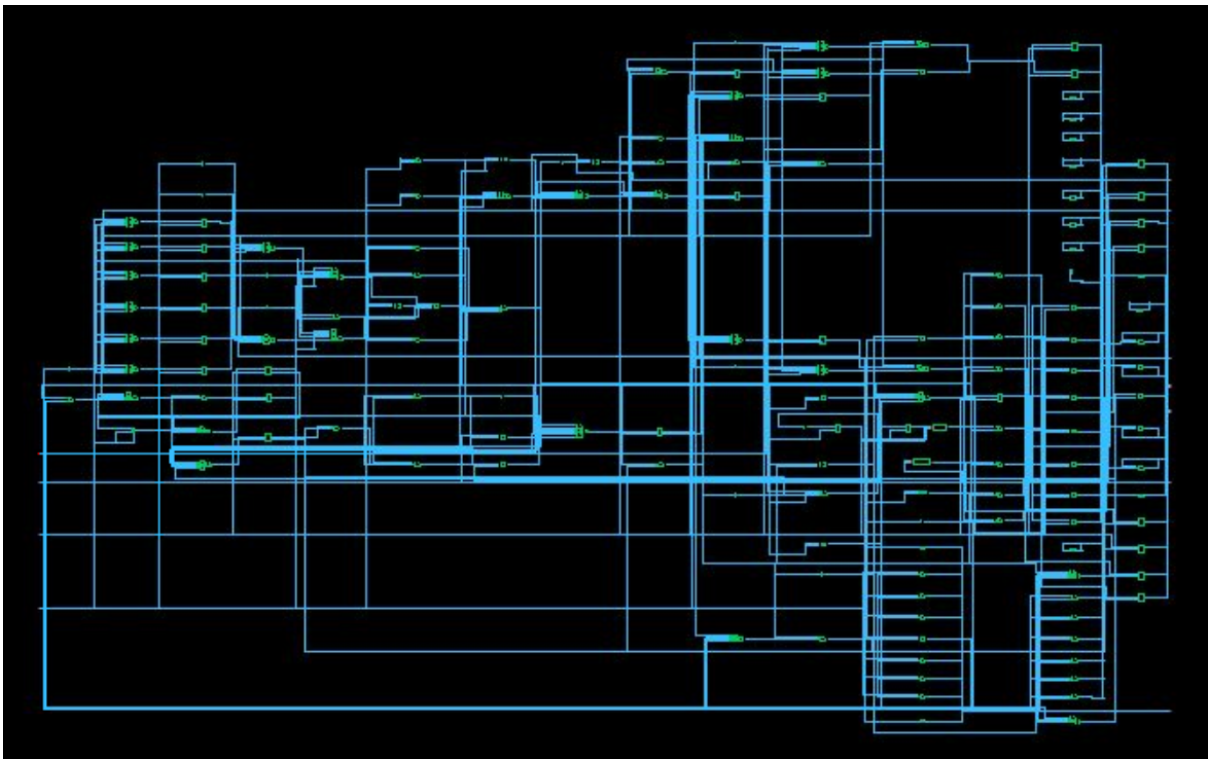


Figure 2.8: Generated Schematic From Imported Verilog

From the generated symbol, the following testbench shown in Figure 2.9 is created. Each input bus is connected to a voltage source in order to simulate signals as described in the Verilog testbench created above, mocking the modelsim simulation. VSS is grounded using the cds_thru cell, and VDD is connected to a 1V DC source. Each individual signal from the block's output buses are connected to a 10 fF capacitor. The clocking period is chosen as defined in the timing.sdc file, set to 100 ns.

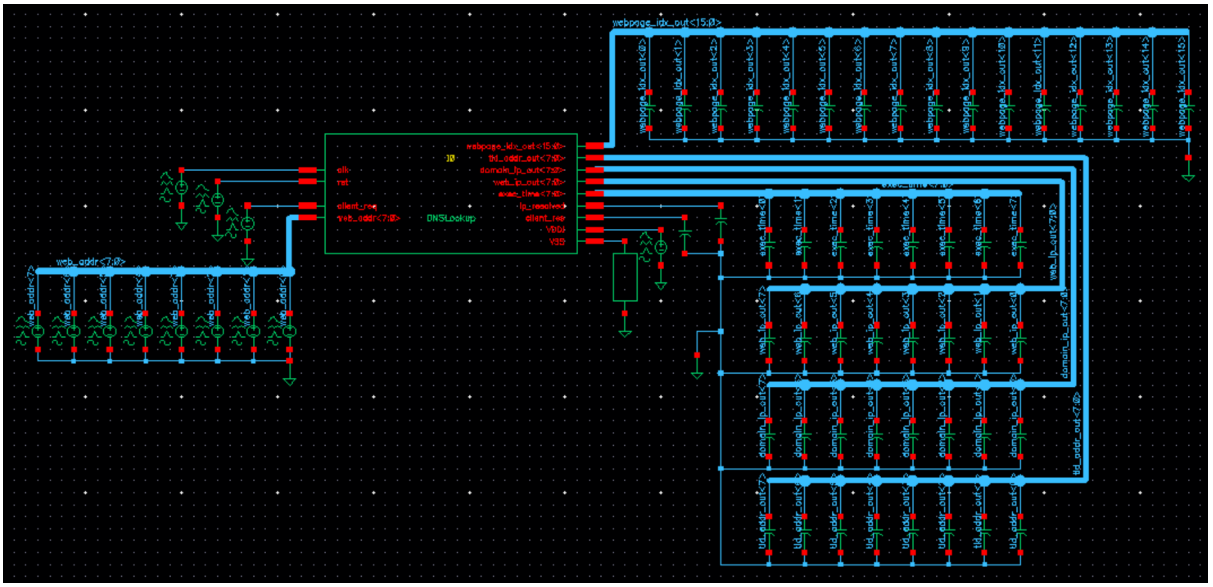


Figure 2.9: Layout Simulation Testbench

2.9 Output Waveforms

Running a transient simulation on the testbench above for 1 us, the waveform shown in Figure 2.10 is obtained. Comparing this waveform to the start of the initial modelsim waveform in Figure 2.11, the initial ip_resolved and client_res outputs are set to pulsing values at the clock frequency until the reset signal is off to allow client requests to arrive. Once a client request arrives, the web_addr bus is set to a value, indicating the start of the state machine. While the clock period in the layout testbench is set to 100 ns according to the timing. sdc file, the clock period used in the Verilog testbench is 10 ns, likely contributing to some timing accuracies.

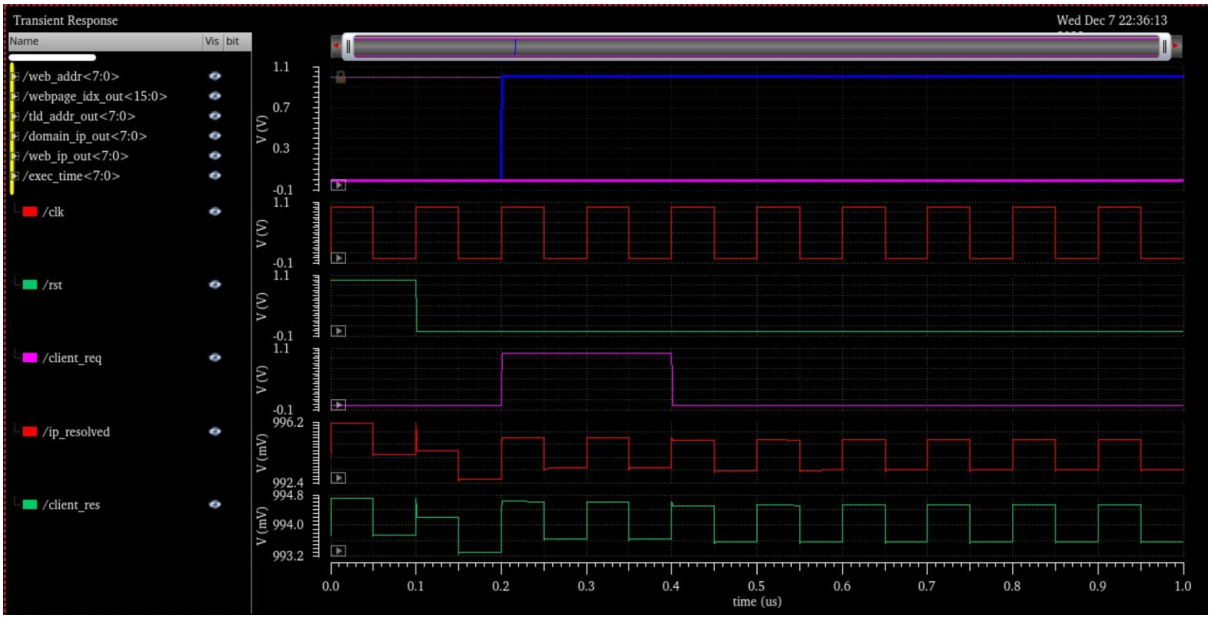


Figure 2.11: Layout Testbench Simulated Waveform

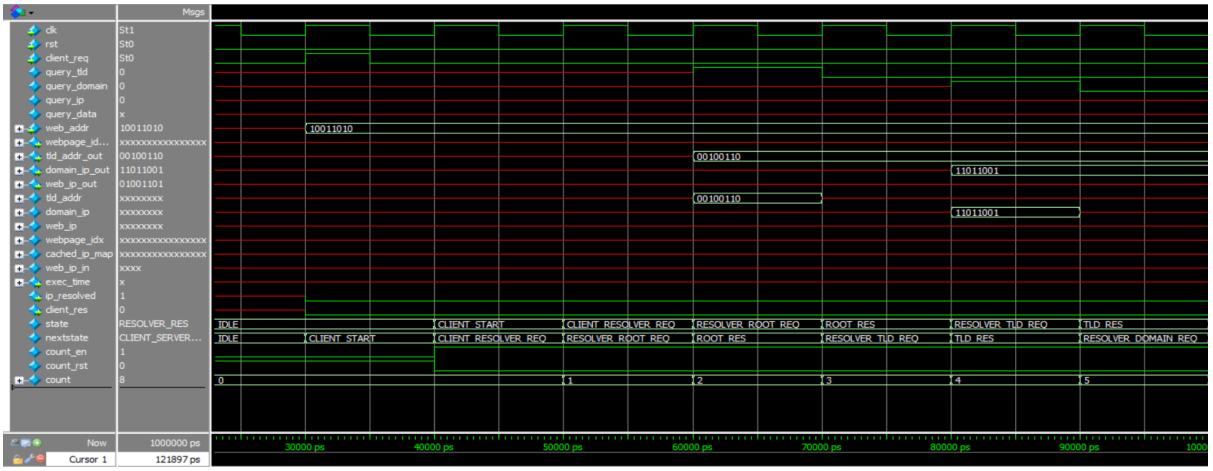


Figure 2.12: Modelsim Verilog Testbench Simulated Waveform

The rise and fall delays are set to 10 ps and may result in some additional inaccurate signal behavior such as the sudden spike in the ip_resolved and client_res signals as the rst signal is turned off. Some discrepancies remain in the layout simulation that were not fixed and shall remain unfixed due to the ungodly hour at which this report is being written.

The Verilog testbench can be seen in Appendix A. The DNS Lookup timing.sdc file can be seen in Appendix B.

3.0 Domino Logic

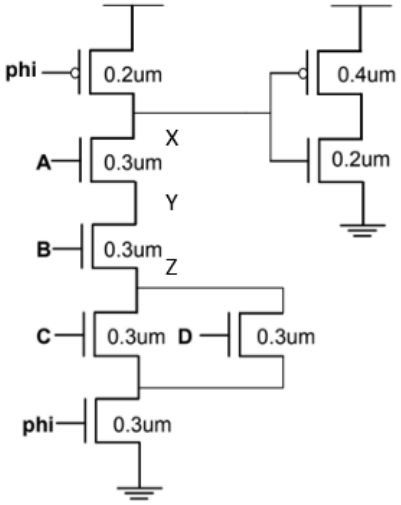


Figure 3.1: Logic Function Diagram

3.1 Logic Function OUT

The logic function for the circuit in Figure 3.1 can be found knowing that during the pre-charge phase, PMOS pulls-up the OUT. During the evaluation phase, the pull-down network is activated to discharge to 0. In order to discharge from VDD to 0, we set A and B, and either C or D to 1, giving the following logic function:

$$OUT = AB(C + D)$$

3.2 Charge Sharing

To calculate the voltage drop at the input of the inverter under worst case charge sharing conditions, we can assume that transistors A and D are still on after charge-sharing. The load capacitance from the inverter to the first stage is Cg*4Wn. With $C_{eff} = 1\ fF/\mu m$ and $C_g = 2\ fF/\mu m$:

$$C_X = C_{eff}(W_N + W_P) + C_g(W_{invN} + W_{invP}) = 1.7\ fF$$

$$C_Y = C_{eff} * W_N = 0.3\ fF$$

$$C_Z = C_{eff} * 2W_N = 0.6\ fF$$

$$V_X = V_Y = V_Z = \frac{C_X V_{DD}}{C_X + C_Y + C_Z} = 0.65\ V$$

At saturation, VX > VDD - VT, therefore we adjust VX to be 0.6V. For the remaining charge on node X after charge sharing: $Q_X = Q_{tot} - Q_Y - Q_Z = VDD * C_X - V_X * C_Y - V_X * C_Z = 1.16\ fC$.

Therefore, the reduction in voltage at the input of the inverter under worst case charge sharing condition is $V = \frac{Q_X}{C_X} = 0.68\,V$.

4.0 Capacitance and Delay Calculations

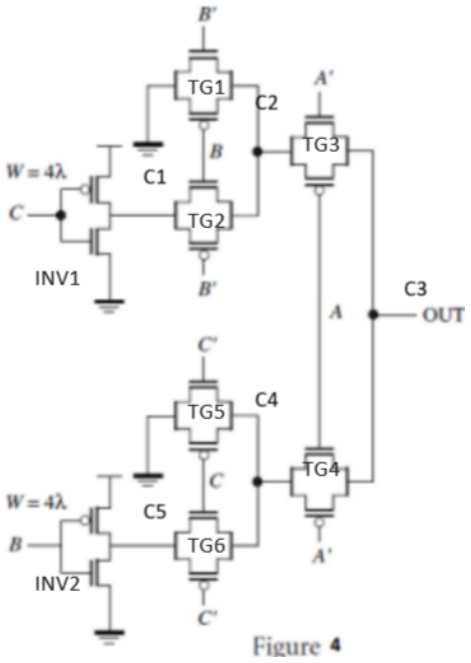


Figure 4.1: Transmission Gates Logic Circuit

To find the minimum and maximum delays of the transmission gates logic circuit shown in Figure 4.1, we first find the total capacitance at each node of the circuit. The equivalent resistance and capacitances of the top half of the circuit can be simplified to the RC circuit in Figure 4.2.

4.1 Capacitance

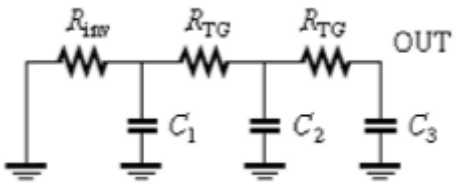


Figure 4.2: Equivalent RC Circuit of Top Half

Given $L = 2\lambda$ and $W = 2\lambda$ and assuming that $R_{eqn} = 12.5\,k\Omega$, $C_g = 2\,fF/\mu m$ and $C_{eff} = 1\,fF/\mu m$, the capacitances at each node are calculated as follows.

The capacitance at Node 1 denoted by C1 includes both the preceding inverter input and output capacitance as well as the next transmission gate, assuming that all transmission gates along this path are ON:

$$C_1 = C_5 = C_{invout} + C_{TGin}$$

$$C_1 = C_5 = [C_g(W_N + W_P) + C_{eff}(W_N + W_P)] + [2C_{eff}W + C_gW]$$

$$C_1 = C_5 = [C_g(2\lambda + 4\lambda) + C_{eff}(2\lambda + 4\lambda)] + [2C_{eff}2\lambda + C_g2\lambda] = 14\lambda\, fF$$

The capacitance at Node 2 denoted by C2 includes the output capacitances from TG1 and TG2 as well as the input capacitance of TG3, assuming that all transmission gates along this path are ON:

$$C_2 = C_4 = C_{TGout} + C_{TGout} + C_{TGin}$$

$$C_2 = C_4 = (2C_{eff}W + C_gW) + (2C_{eff}W + C_gW) + (2C_{eff}W + C_gW)$$

$$At\ W = 4\lambda,\ C_2 = C_4 = 20\lambda\, fF$$

The capacitance at Node 3 denoted by C3 includes the output capacitances from TG3 and TG4, assuming that all transmission gates along this path are ON:

$$C_3 = C_{TGout} + C_{TGout}$$

$$C_3 = (2C_{eff}W + C_gW) + (2C_{eff}W + C_gW)$$

$$At\ W = 4\lambda,\ C_3 = 16\lambda\, fF$$

4.2 Maximum Delay

For the maximum delay, we take the path of most transistors, including C1, C2 and C3 along the path. Using Elmore delay:

$$t_{dmax} = R_{inv}C_1 + (R_{inv} + R_{TG})C_2 + (R_{inv} + 2R_{TG})C_3 = 1.125\lambda\, ns$$

where $R_{inv} = R_{eqn}\frac{L_N}{W_N} = 12.5k\frac{2\lambda}{2\lambda} = 12.5\ k\Omega$ and C1, C2, and C3 are as shown above.

4.3 Minimum Delay

For the minimum delay, we take the path of least transistors, including only C2, and C3 along the path. Using Elmore delay:

$$t_{dmin} = R_{TG} * C_2 + 2R_{TG} * C_3 = 0.550\lambda\, ns$$

where $R_{inv} = R_{eqn}\frac{L_N}{W_N} = 12.5k\frac{2\lambda}{2\lambda} = 12.5\ k\Omega$ and C1, C2, and C3 are as shown above.

5.0 Power Consumption

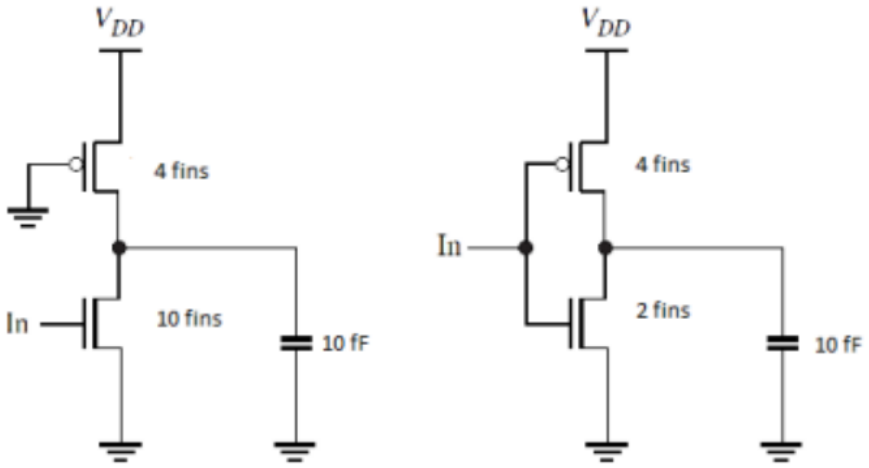


Figure 5.1: Inverter Circuits

5.1 Static Power Consumption

Given 15 nm technology and the number of fins for each transistor as shown in Figure 5.1, the Static Power Consumption for the left circuit can be calculated. To calculate the Static and Dynamic Power Analytically, it is assumed that
$V_{DD} = 1V,\ V_{THP} = 0.4V,\ E_{Cp} = 24V/\mu m,\ v_{SAT} = 8E6\ cm/s,\ C_{OX} = 1.6E-6\ F/cm^2,\ \alpha = 0.5,$
$C_{eff} = 1\ fF/\mu m,\ C_{load} = 10\ fF,\ L = 1\ fin = 2\lambda$. Therefore for the transistor widths, 2 fins = 30 nm, 4 fins = 60 nm, and 10 fins = 150 nm.

For Static Power Consumption, $P_{static} = \alpha V_{DD} I_{DC}$ where the activity factor α is assumed to be 0.5 due to the clock switching and DC current only flowing once per cycle.

To find IDC while the inverter is off and outputs VOl, we must find the states of the NMOS and/or PMOS when Vin = VDD = 1V. For PMOS saturation:

$$V_{SG} > |V_{TH}|,\ V_{SD} > V_{SG} - |V_{TH}|$$

Since VSG = VDD and VSD = VDD - VOL = VOH, the PMOS is in saturation.

Using the PMOS long channel saturation current equation:

$$I_{DC} = I_{SD}(sat) = \frac{W_P v_{sat} C_{OX} (|V_{SG}| - |V_{TH}|)^2}{(|V_{SG}| - |V_{TH}| + E_C L)} = 288\ \mu A$$

Therefore the Static Power Consumption can be calculated for the left circuit:

$$P_{static} = \frac{1}{2} V_{DD} I_{DC} = 144\ \mu A$$

5.2 Dynamic Power Consumption

Sources of power dissipation for dynamic power include switching load capacitances and short-circuit current. The dynamic power consumption can be calculated using an input frequency of 100 MHz and neglecting wire capacitance and short-circuit current:

$$P_{dynamic} = P_{switching} + P_{shortcircuit} = \alpha C V_{DD}^{2} f$$

Neglecting wire capacitance, the total capacitance can be calculated as follows:

$$C = C_{eff}(W_P + W_N) + C_{load}$$

Therefore, for the left circuit:

$$P_{dynamic} = \frac{1}{2}[C_{eff}(W_{4\,fins} + W_{10\,fins}) + C_{load}]V_{DD}^{2} f = \frac{1}{2} * 10.21fF * 1^2 * 100\,MHz = 510\,nW$$

And for the right circuit:

$$P_{dynamic} = \frac{1}{2}[C_{eff}(W_{4\,fins} + W_{2\,fins}) + C_{load}]V_{DD}^{2} f = \frac{1}{2} * 10.09\,fF * 1^2 * 100\,MHz = 504\,nW$$

5.3 Power Consumption Parameter Simulation

As opposed to Sections 5.1 and 5.2, if the MOS parameters were to be found by running a DC or Transient simulation to calculate IDC, a different result would likely occur. Using the same IDC expression for the PMOS at saturation, the parameters W_P, v_{sat}, C_{OX}, $E_C L$, V_{TH} must be found by simulation.

$$I_{DC} = I_{SD}(sat) = \frac{W_P v_{sat} C_{OX}(|V_{SG}| - |V_{TH}|)^2}{(|V_{SG}| - |V_{TH}| + E_C L)}$$

To find the current expression parameters, the testbench in Figure 5.2 is created.

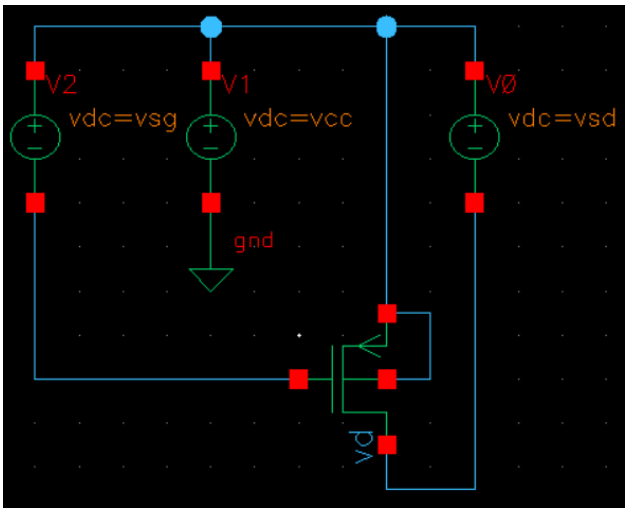


Figure 5.2: Testbench for IDC Parameter Measurement

In order to find VTH, VSD and VCC are set to 1V while conducting a voltage sweep on VSG from 0-1V. VTP can be measured at the point at which the current enters the linear region to determine VTH. As shown in Figure 5.3, VTH = 0.242 V.

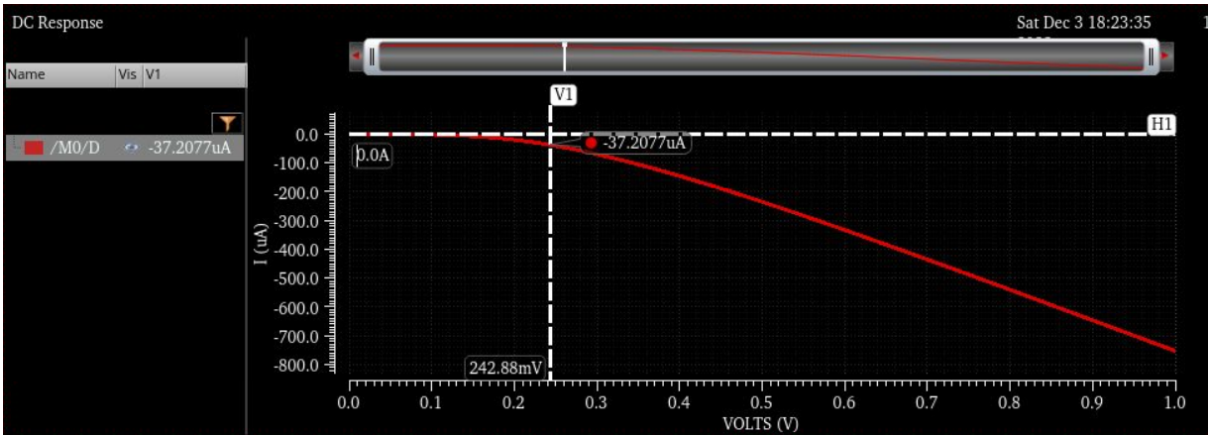


Figure 5.3: VTH Point on PMOS IV curve

Next, the parameters $W_P v_{sat} C_{OX}$ and $E_C L$ can be found by sweeping VDS from 0-2V at VSG = 0.5V and VSG = 1V, resulting in the plot in Figure 5.4.

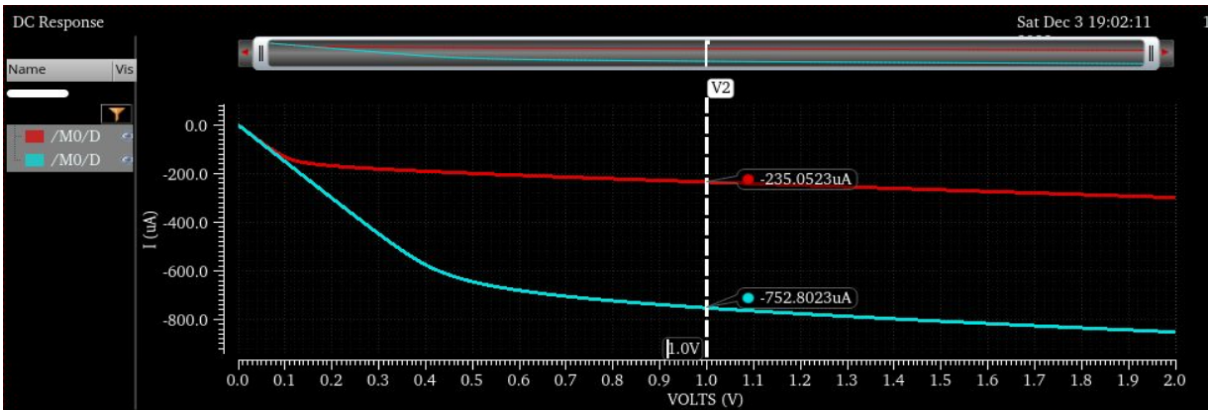


Figure: 5.4: VDS Voltage Sweep on PMOS IV curve

Using the above VSG and IDS values, the remaining missing parameters can be solved:

$$I_{DC} = I_{SD}(sat) = -\ 235.05\ \mu A = \frac{W_P v_{sat} C_{OX}(0.5-0.242)^2}{(0.5-0.242+E_C L)}$$

$$I_{DC} = I_{SD}(sat) = -\ 752.8\ \mu A = \frac{W_P v_{sat} C_{OX}(1-0.242)^2}{(1-0.242+E_C L)}$$

Solving the two current equations, we find that $W_P v_{sat} C_{OX} = -\ 1.01708E\ -\ 3$ and $E_C L = 1.8274E\ -\ 2$.

Therefore, IDC at VSG = 1V is found to be $I_{DC} = 173\ \mu A$. Using the static power expression similarly to Section 5.1 with VDD = 1V:

$$P_{static} = \frac{1}{2} V_{DD} I_{DC} = 86.5\ \mu A$$

As Dynamic Power is independent of IDC, the Dynamic Power consumption remains the same as in Section 5.2

5.4 VTC Curves

A VTC sketch of the two Inverter outputs are shown in Figure 5.2. The Pseudo NMOS inverter results in 3 distinct current regions including the Subthreshold region when the inverter is on, Short Circuit Current as the transistor is transitioning, and DC Current as the transistor turns off. The CMOS inverter results in a Subthreshold current region when the transistor is on, a Short Circuit region as the transistor is transitioning, and another Subthreshold Current region as the transistor turns off.

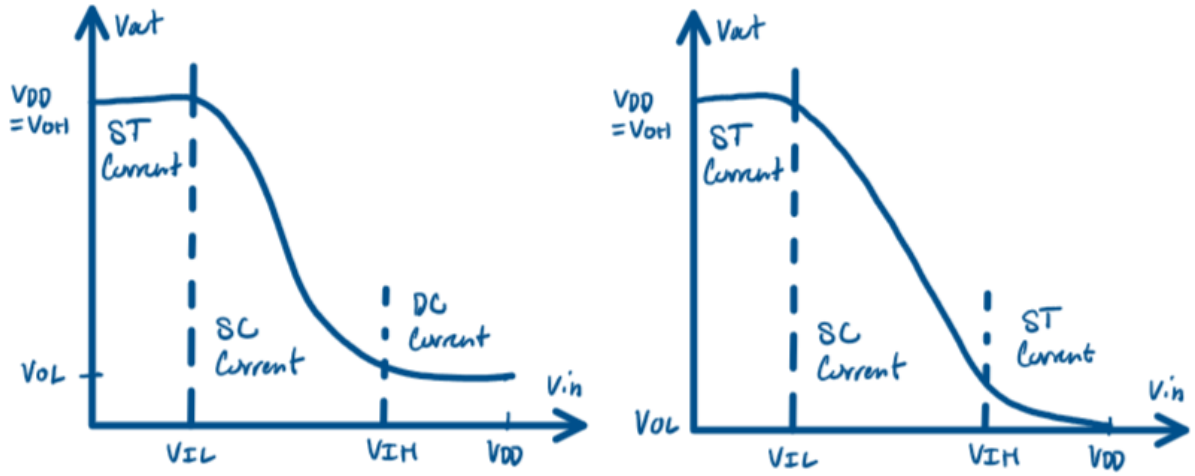


Figure 5.2: a) Pseudo-NMOS Inverter VTC, b) CMOS Inverter VTC

A Cadence simulation is conducted next to compare the voltage curves with the above VTC sketches for the two inverters.

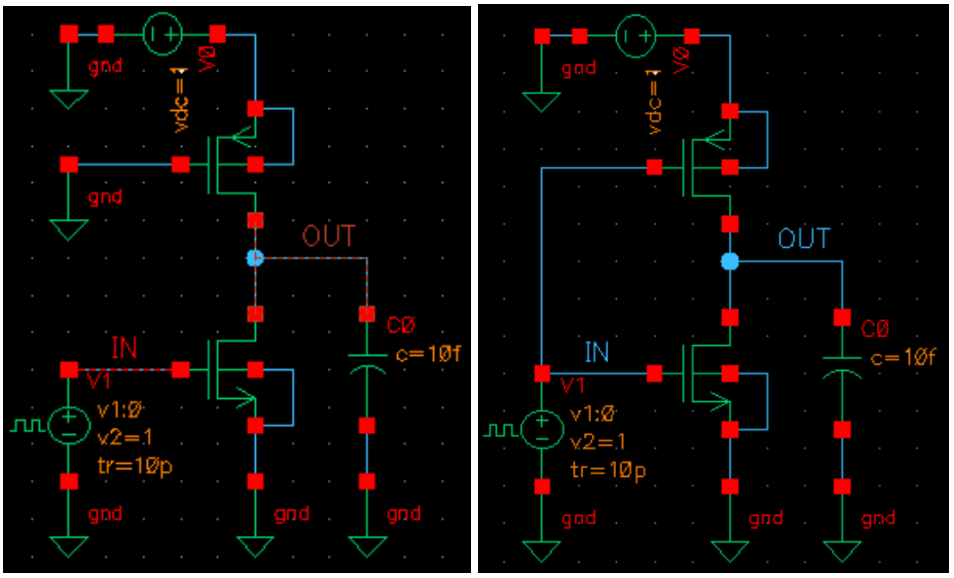


Figure 5.3: Cadence Simulation of Inverter Circuits

Using a rise and fall time of 10 ps and a 10 ns period, the Voltage curves in Figure 5.4 are obtained. It is observed that the top voltage plot matches the behavior of Figure 5.2 a) for the Pseudo NMOS inverter, as the Voltage does not reach 0 in the DC current region, while the bottom plot matches the behavior of Figure 5.2 b) for the CMOS inverter, as the Voltage reaches 0 in the Subthreshold Current region.

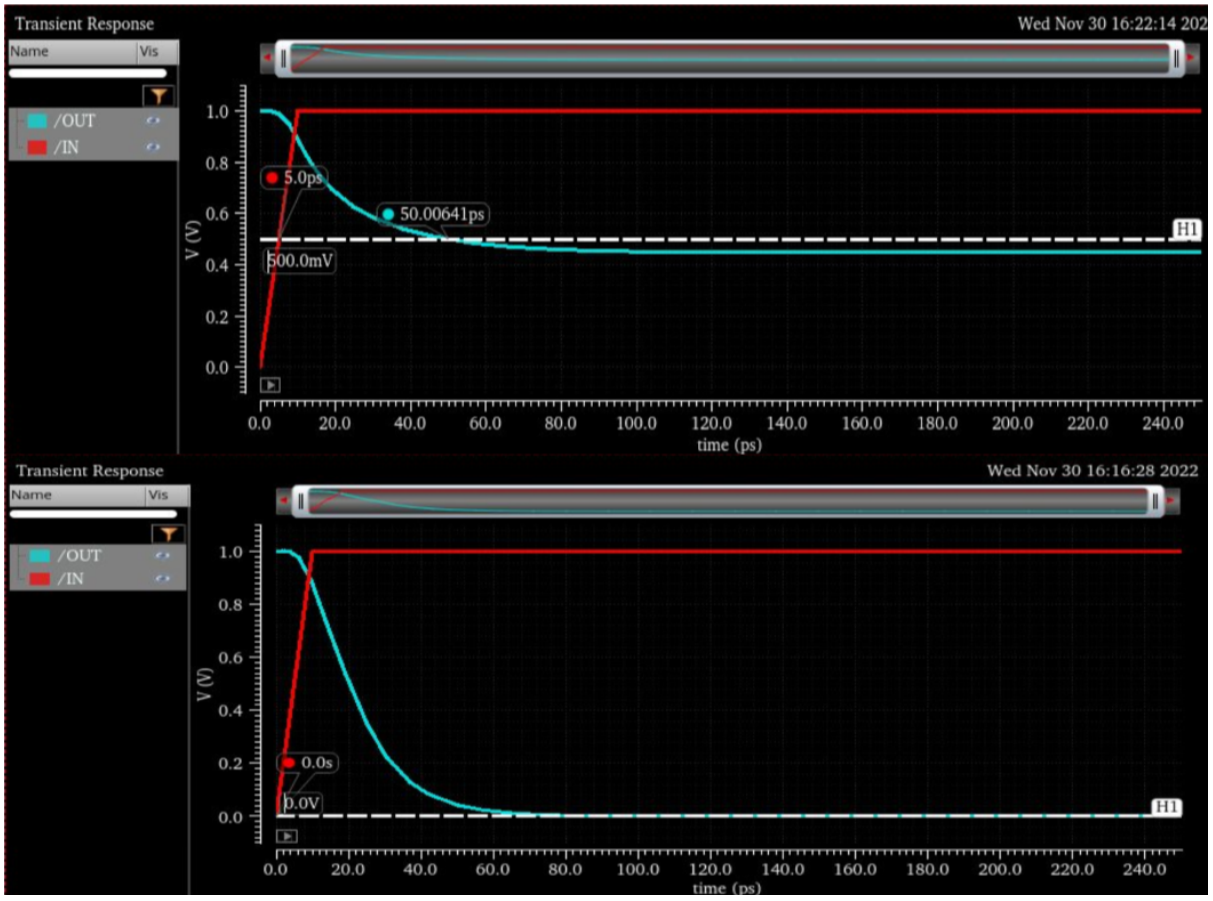


Figure 5.4: Resulting Voltage Curves from Cadence Simulation

6.0 Activity Factor

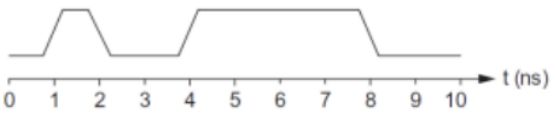


Figure 6.1: Signal at 1 GHz clock rate

With the system clock frequency f = 1 GHz, the switching frequency $f_{SW} = \alpha f$ where α is the activator factor and probability that the circuit node transitions from 0 to 1. First we find the period of the clock signal: $T = 1/f = 1\,ns$.

Figure 6.1 shows 10 periods of the clock signal. The given signal switches 2 times from 0 to 1. Therefore the Activity Factor:

$$\alpha = \frac{\#\,transitions}{total\ cycles} = \frac{2}{10} = 0.2$$

7.0 Interconnects

The following interconnect is given:

- 18 mm Metal 7 wire
- 40 nm technology
- W = 0.4 um wide and T = 0.8 um thick wire, spacing to adjacent wires of D = 2 um
- Height above and below to Metal 8 and Metal 6 is 0.5 um
- Cu for interconnect ($\rho = 0.017\ \Omega - \mu m$)
- low-k dielectric material ($\epsilon_r = 3.0$)
- Worst-case coverage of Metal 6 below, Metal 8 above, Metal 7 for adjacent lines that are shielding wire of interest

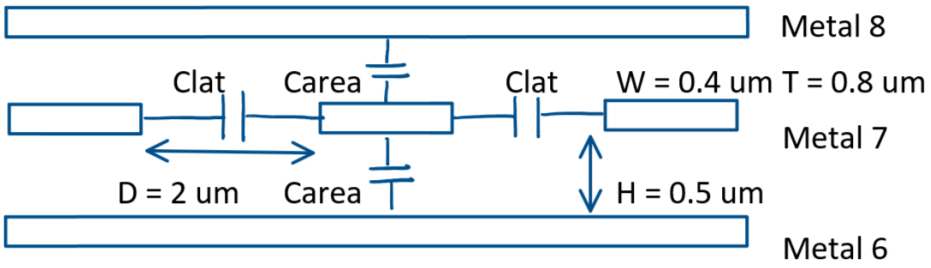


Figure 7.1: Wire Capacitances of Interconnect

7.1 Wire Resistance and Capacitance

The resistance per unit length and capacitance per unit length for the above wire can be found using Figure 7.1.

The resistance of an interconnect or wire is given by: $R = \frac{\rho L}{A} = \frac{\rho L}{TW}$. For resistance of an interconnect or wire per unit length:

$$R_{eq} = \frac{\rho}{TW} = 0.0531\ \Omega/\mu m$$

The area capacitance caused by the distance between different Metal layers such as Metal 8 and 7 and Metal 6 and 7 can be found as follows:

$$C_{area} = \epsilon_{OX}\frac{W}{H} = \epsilon_r\epsilon_0\frac{W}{H} = 0.02125\ fF/\mu m$$

The lateral capacitance caused by the distance between Metal 7 wires can be found as follows:

$$C_{lateral} = \epsilon_{OX}\frac{T}{D} = \epsilon_r\epsilon_0\frac{T}{D} = 0.0106\ fF/\mu m$$

The fringing capacitance can be found as follows:

$$C_{fringe} = \epsilon_{OX} ln(1 + \frac{T}{H}) = \epsilon_r\epsilon_0 ln(1 + \frac{T}{H}) = 0.02538\ fF/\mu m$$

Therefore for the total capacitance per unit length:

$$C_{TOT} = 2C_{area} + 2C_{lateral} + 2C_{fringe} = 0.11446\, fF/\mu m$$

7.2 Delay of Distributed RC Wire

Assuming that the driver is a perfect voltage source and that the load capacitance is 50 fF, the delay of the distributed RC wire can be determined using the corresponding RC T-Model circuit in Figure 7.2.

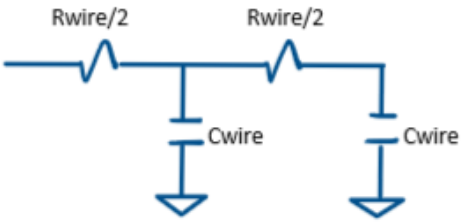


Figure 7.2: RC T-Model Circuit of Wire with Driver as a Perfect Voltage Source

Using Req and Ctotal from Section 7.1, the wire resistance and capacitance can be calculated as follows:

$$R_{wire} = R_{eq}L = 0.0531 * 18E3 = 955.8\,\Omega$$

$$C_{wire} = C_{TOT}L = 0.11446 * 18E3 = 2060.28\, fF$$

Using the wire resistance and load capacitance, the Elmore delay of the wire can be calculated as follows:

$$t_{delmore} = C_{wire}(\frac{R_{wire}}{2}) + C_{load}(2 * \frac{R_{wire}}{2}) = 1.03239\, ns$$

7.3 Delay of Wire Driven by 25X Inverter

The delay of the distributed RC wire driven by a 25X inverter can be determined using the corresponding RC Π-Model circuit in Figure 7.3 and Rpeq = 30 kΩ, $2\lambda = 40\, nm$ for a 40 nm technology.

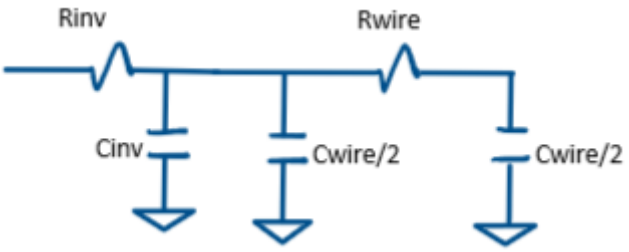


Figure 7.3: RC Π-Model of Circuit or Wire with Driver as 25X Inverter

Rinv can first be found as $R_{inv} = \frac{1}{25} * R_{Peq} * \frac{L_P}{W_P} = \frac{1}{25} * 30k * \frac{2\lambda}{4\lambda} = 600\ \Omega$

where $L_P = 2\lambda$ and $W_P = 4\lambda$.

For Cinv: $C_{inv} = 25 * C_{eff}(W_N + W_P) = 25 * C_{eff}(2\lambda + 4\lambda) = 3\ fF$

Using the same Rwire and Cwire values as in Section 7.2, the Elmore delay can be calculated as:

$$t_{delmore} = (C_{inv} + \frac{C_{wire}}{2}) * (R_{inv}) + (\frac{C_{wire}}{2}) * (R_{wire} + R_{inv}) = 2.2219\ ns$$

Appendix A: Verilog Testbench

```
`timescale 1ns/1ps

module DNSLookup_tb();
	logic clk, rst, client_req;
	logic [7:0] web_addr, tld_addr, domain_ip, web_ip, exec_time;
	logic [15:0] webpage_idx;
	logic ip_resolved, client_res;

	// Instantiate the module to test
	DNSLookup DUT (
		.clk(clk),
		.rst(rst),
		.client_req(client_req),
		.web_addr(web_addr),
		.webpage_idx_out(webpage_idx),
		.tld_addr_out(tld_addr),
		.domain_ip_out(domain_ip),
		.web_ip_out(web_ip),
		.exec_time(exec_time),
		.ip_resolved(ip_resolved),
		.client_res(client_res)
	);

	// Initialize signals
	initial begin
		{clk, rst} = 2'b01;
		client_req = 1'b0;

	forever
	begin
		clk = ~clk;
		#5;
	end
	end
```

```
	// inputs
	initial begin
		#15;
		rst = 1'b0;
		#15;

		// query first web addr
		client_req = 1'b1;
		web_addr = 8'b1001_1010;
		#15;
		client_req = 1'b0;
		#140;

		// test that first web addr is cached by querying again
		client_req = 1'b1;
		web_addr = 8'b1001_1010;
		#15;
		client_req = 1'b0;
		#80

		// query second web addr
		client_req = 1'b1;
		web_addr = 8'b0011_1110;
		#15;
		client_req = 1'b0;
		#140;
	end
endmodule
```

Appendix B: timing.sdc File

```
current_design up_counter

create_clock [get_ports {clk}] -name clk -period 100 -waveform {0 50}
```