Simulation of Logic Primitives and Dynamic D-latch with Verilog-XL

November 30, 2011

Robert D’Angelo

Tufts University

Electrical and Computer Engineering

EE-103 Lab 3: Part I&II

Professor:

Dr. Valencia Joyner Koomson
Abstract

In this lab logic primitives NOT, NAND2, NOR2, NAND3, NOR3, and XOR2 were constructed from functional views written in Verilog. These primitives are to be used for the design of a multiplier in Part II of this exercise. Additionally, a dynamic d-latch was simulated from a functional view. This circuit exploits leakage effects of transistors to store the previous input until the next clock transition. It is shown that a large leakage time relative to the clock period is necessary for proper functioning of the latch.
Contents

Abstract ...................................................... 2

1 Logic Primitives ........................................... 4
   1.1 Logical Effort ........................................ 4
   1.2 Simulations ......................................... 6

2 Dynamic D Latch .......................................... 8
   2.1 Design ............................................. 8
   2.2 Simulation ......................................... 9

3 Conclusions ............................................. 10

List of Figures

   1 Logic Primitives Test Schematic ......................... 6
   2 Logic Primitives Simulation Results ..................... 7
   3 D Latch Circuit ....................................... 8
   4 D Latch Schematic .................................... 9
   5 D Latch Simulation, 20ns Decay Time ................... 9
   6 D Latch Simulation, 2ns Decay Time ................... 9
   7 Leakage Independent D Latch Schematic ................ 10

List of Tables

   1 Summary of Gate Delays ................................ 4
   2 D Latch Truth Table ................................... 8
1 Logic Primitives

1.1 Logical Effort

Logical effort can be used to estimate the delay of logic gates from the following formula.

\[ d = gh + p \]  \hspace{1cm} (1.1)

where \( d \) is the delay, \( g \) is the logical effort of the gate, \( h \) is the electrical effort, and \( p \) is the parasitic effort. For simplicity we assume that \( h = 1 \), implying that the input capacitance is equal to load capacitance, and that \( p = 0 \), implying that parasitics are insignificant to the delay estimate. Therefore, the gate delay is equal to the logical effort, \( g \).

The logical effort of a gate is dependent upon the aspect ratio of the transistors used \( (r) \), the number of gate inputs, and the type of gate.

\[
\begin{align*}
    g_{\text{NOT}} &= 1 \\
    g_{\text{NAND}} &= \frac{n + r}{1 + nr} \\
    g_{\text{NOR}} &= \frac{1}{1 + r} \\
    g_{\text{XOR}} &= n2^{n-1}
\end{align*}
\]  \hspace{1cm} (1.2 - 1.5)

From Lab 2 it was determined that \( r = 1.8 \) for equal rise and fall time of an inverter (assuming equal gate lengths):

\[ r = \frac{W_n}{W_p} = \frac{2.7\mu m}{1.5\mu m} = 1.8 \]  \hspace{1cm} (1.6)

This inverter will serve as the reference for determining the delays through other gates. Therefore, the NOT gate will have a delay of 1 unit. Table 1 summarizes the equations and estimated delays for the remaining gates. The gate delays are rounded to the nearest integer so that they can be programmed using Verilog.

<table>
<thead>
<tr>
<th>Gate</th>
<th>Logical Effort</th>
<th>( g ) (exact)</th>
<th>( g ) (estimate)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOT</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>NAND2</td>
<td>( \frac{n+2r}{1+nr} )</td>
<td>1.35</td>
<td>1</td>
</tr>
<tr>
<td>NAND3</td>
<td>( \frac{n+3r}{1+nr} )</td>
<td>1.71</td>
<td>2</td>
</tr>
<tr>
<td>NOR2</td>
<td>( \frac{1+2r}{1+nr} )</td>
<td>1.64</td>
<td>2</td>
</tr>
<tr>
<td>NOR3</td>
<td>( \frac{1+3r}{1+nr} )</td>
<td>2.29</td>
<td>2</td>
</tr>
<tr>
<td>XOR2</td>
<td>( 2 * 2^{n-1} )</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

Using these delays the functional views for each gate were written in Verilog. Programming Listings 1-6 below show the functional views.
// Verilog HDL for "ee103", "inv" "functional"
module inv (Y, A);
  output Y;
  input A;
  not #1 (Y, A);
endmodule

Listing 1: Booth Encoder Verilog Stimulus

// Verilog HDL for "ee103", "NAND2" "functional"
module NAND2 ( vout , va , vb );
  input va;
  input vb;
  output vout;
  nand #1 (vout, va, vb);
endmodule

Listing 2: Booth Encoder Verilog Stimulus

// Verilog HDL for "ee103", "NAND3" "functional"
module NAND3 (vout , va , vb , vc);
  output vout;
  input va , vb , vc;
  nand #2 (vout, va, vb, vc);
endmodule

Listing 3: Booth Encoder Verilog Stimulus

// Verilog HDL for "ee103", "NOR2" "functional"
module NOR2 ( vout , va , vb );
  input va;
  input vb;
  output vout;
  nor #2 (vout, va, vb);
endmodule

Listing 4: Booth Encoder Verilog Stimulus

// Verilog HDL for "ee103", "NAND3" "functional"
module NOR3 (vout , va , vb , vc);
  output vout;
  input va , vb , vc;
  nor #2 (vout, va, vb, vc);
endmodule

Listing 5: Booth Encoder Verilog Stimulus
1.2 Simulations

The logic primitive symbol views can be seen in the schematic in Figure 1. This schematic was used to simulate the logic gates in Verilog XL. The simulation results are shown in Figure 2. The Verilog stimulus file is shown in Program Listing 7. The inputs are swept from 0b000 to 0b111. It can be seen that the gates are operating correctly. Note that the delays shown in the simulation match those in the functional view.

---

// Verilog HDL for "ee103", "XOR2" functional
module XOR2 (vout, va, vb);
    output vout;
    input va, vb;
    xor #4 (vout, va, vb);
endmodule

Listing 6: Booth Encoder Verilog Stimulus
1.2 Simulations

Figure 2: Logic Primitives Simulation Results

// Verilog stimulus file.
// Please do not create a module in this file.

initial
begin
    IN2=1'b0; IN1=1'b0; IN0=1'b0;
    #50 IN2=1'b0; IN1=1'b0; IN0=1'b1;
    #50 IN2=1'b0; IN1=1'b1; IN0=1'b0;
    #50 IN2=1'b0; IN1=1'b1; IN0=1'b1;
    #50 IN2=1'b1; IN1=1'b0; IN0=1'b0;
    #50 IN2=1'b1; IN1=1'b0; IN0=1'b1;
    #50 IN2=1'b1; IN1=1'b1; IN0=1'b0;
    #50 IN2=1'b1; IN1=1'b1; IN0=1'b1;
    #50 $finish;
end

Listing 7: Booth Encoder Verilog Stimulus
2 Dynamic D Latch

2.1 Design
A dynamic D Latch is a device that produces the input, D, at the output Q when the clock is asserted high, but holds the previous value when the clock is low. This functionality is shown in the truth table in Table 2. An implementation of a D Latch is shown in Figure 3. When the clock is high, the transmission gate passes the signal D, which charges the parasitic capacitance at the input of the buffer. This capacitance is prone to leakage; therefore, the clock period must be smaller than the leakage time so that the signal will not decay at the output.

<table>
<thead>
<tr>
<th>Clk</th>
<th>D</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>x</td>
<td>Q prev</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 3: D Latch Circuit

The transmission gate is modeled using “rcmos” and the parasitic capacitance is modeled using “trireg” command with delay. If the delay is too fast compared to the clock period, the output will decay before the next transition. The functional view of the D Latch is shown in Program Listing 8.

```verilog
// Verilog HDL for "ee103", "dlatch" "functional"
module dlatch (d, clk, q, q_);
  output q, q_;
  input d, clk;
  wire clk_;
  not #1 (clk_, clk);
  trireg(medium) #(0,0,20) storage;
  rcmos #1 (storage, d, clk, clk_);
  not #1 (q_, storage);
  not #1 (q, q_);
endmodule
```

Listing 8: Booth Encoder Verilog Stimulus
2.2 Simulation

A schematic view of the D latch is shown in Figure 4. The functional view was simulated with two different decay rates to demonstrate the effects of having leakage that is faster than the clock rate (2ns delay) and leakage that is slower than the clock rate (20ns delay). The simulations agree well with common sense, showing that it is more desirable to have a long decay time so that the output does not fall below the threshold before the next clock transition. For comparison, Figure 7 shows the typical D latch schematic that does not depend on leakage. This implementation is slower and larger, however.
3 Conclusions

This lab involved the construction of logical primitives using Verilog. All of the gates were successfully simulated using Verilog-XL. The calculated delay times were observed. A dynamic d-latch was implemented at the functional level. Two different schematic implementations are shown. It was verified that a large leakage time relative to the clock period is necessary so that the latch can hold the previous value until the next transition.