Cadence Tutorial 6

Verilog-XL Simulation for Dynamic Logic

EE577b Fall 98

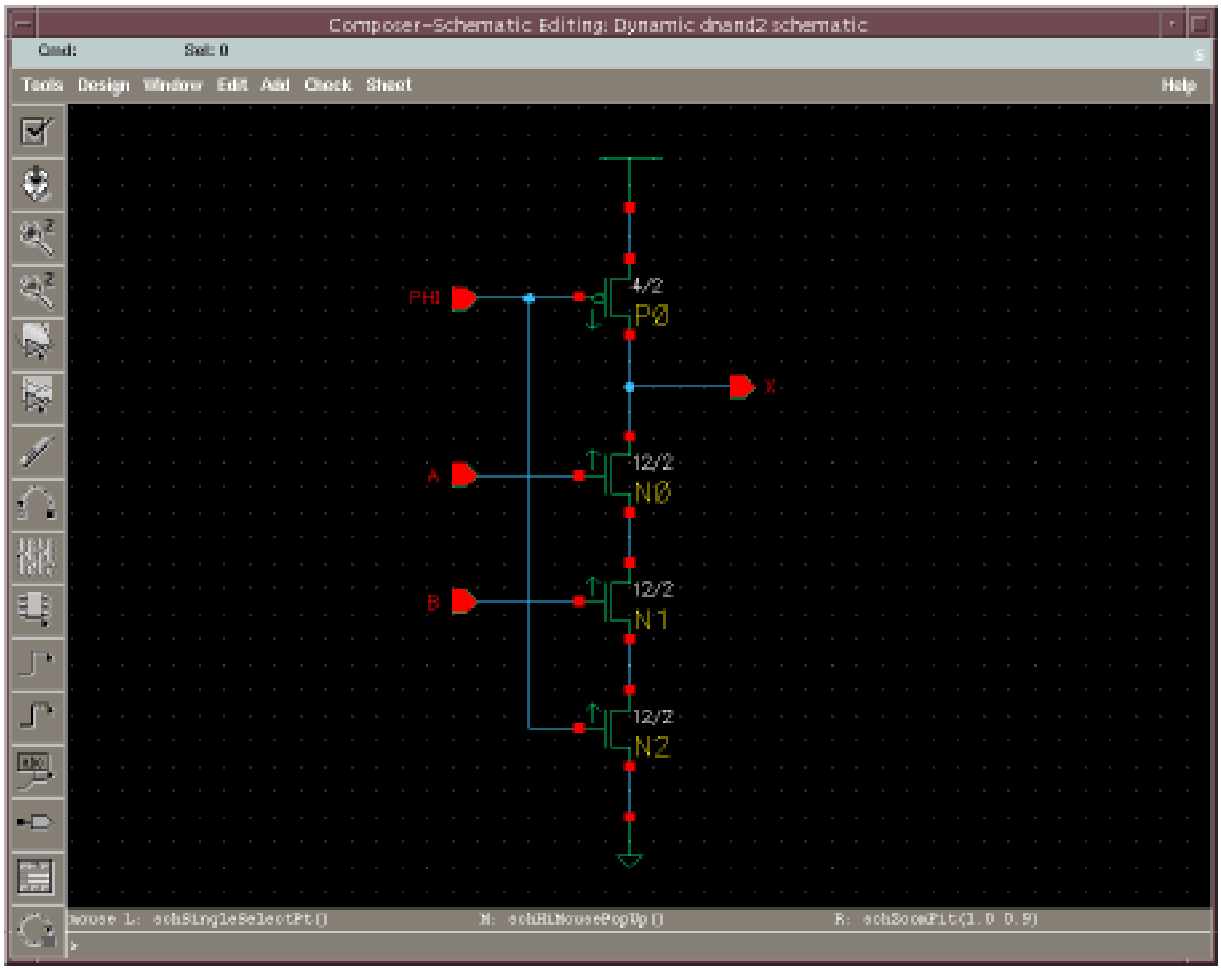
In this tutorial, I am going to demonstrate how to design and simulate the domino style dynamic logic.

1. Tutorial Setup

No previous tutorial is required to start this tutorial.

2. 2-input Dynamic NAND Gate Schematic

Finish the following 2-input dynamic nand gate schematic with the name *dnand2* in *Dynamic* library. (choose other names for cellview and library if you wish)

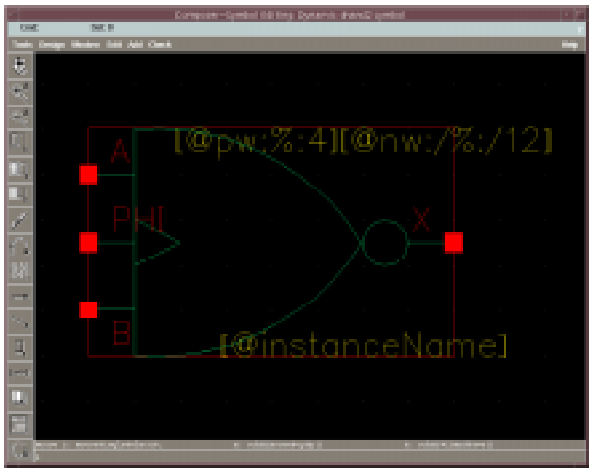


- I used @pw=4, @nw=12 for transistor sizing. All lengths of transistor are “2” (string type). See tutorial 1 & 2 for detailed explanation about parameterized design.

jsmoon@ISI.EDU

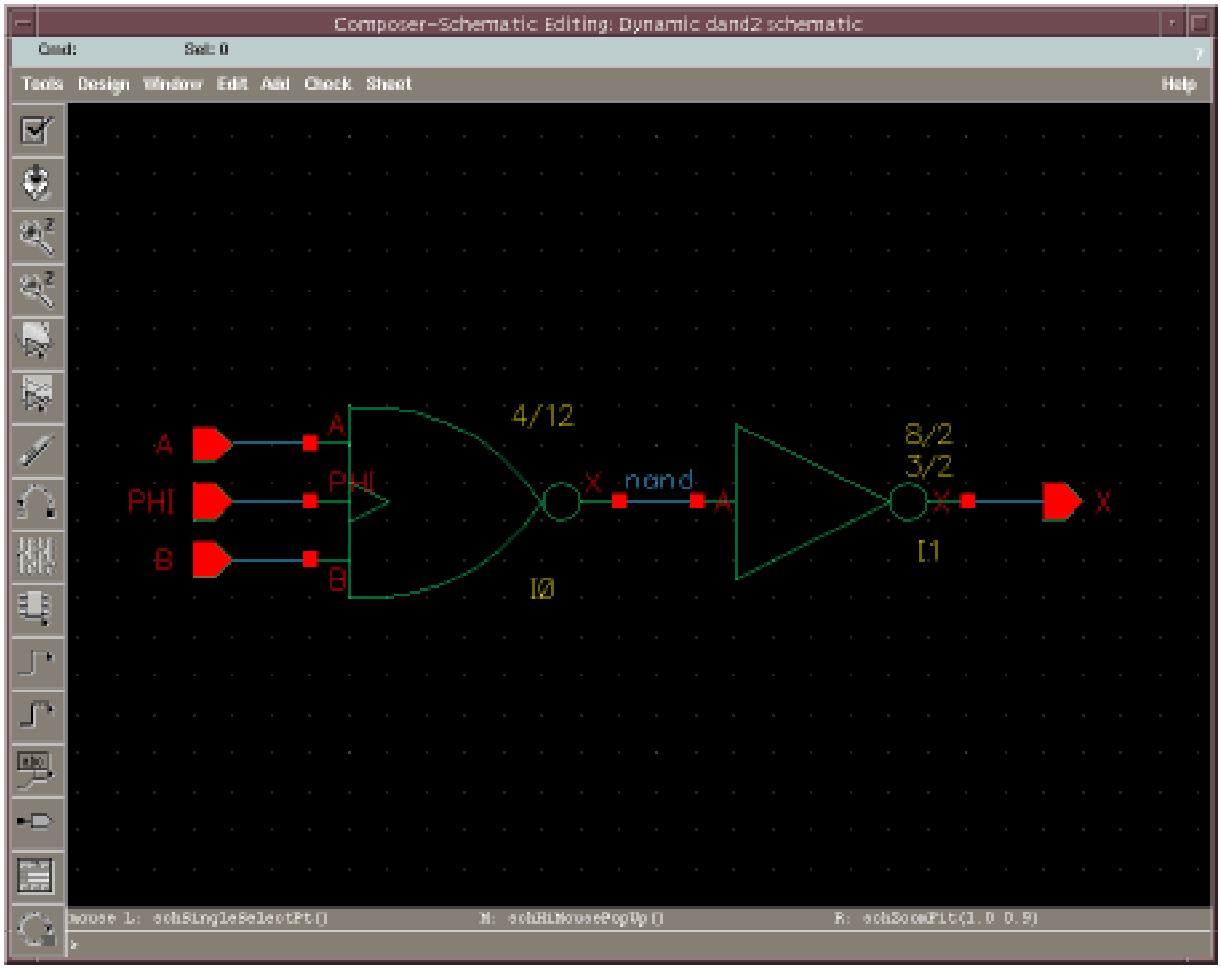
3. 2-input Dynamic NAND Gate Symbol

Complete the following 2-input dynamic nand gate symbol with the name *dnand2* in *Dynamic* library.



4. 2-input Domino Style AND Gate Schematic

Complete the following 2-input domino style and gate schematic with the name *dand2* in *Dynamic* library. (It uses *inverter* from *Cell* library and *dnand2* from *Dynamic* library)
Make sure you put *nand* wire name on the output of nand gate.



jsmoon@ISI.EDU

5. Verilog-XL Simulation Based on the Netlist from Schematic

Run Verilog-XL simulation with the following testfixture.template and testfixture.verilog.
You need to generate netlist from Verilog-XL integration tool before starting simulation.
See tutorial 4 for Verilog-XL simulation procedure for schematic.
In my case, I created *dand2.run1* directory and copied *testfixture.template* and *testfixture.verilog* in *wave* subdirectory under *dand2.run1*.

In *testfixture.template* file, insert the following lines under *timescale*.

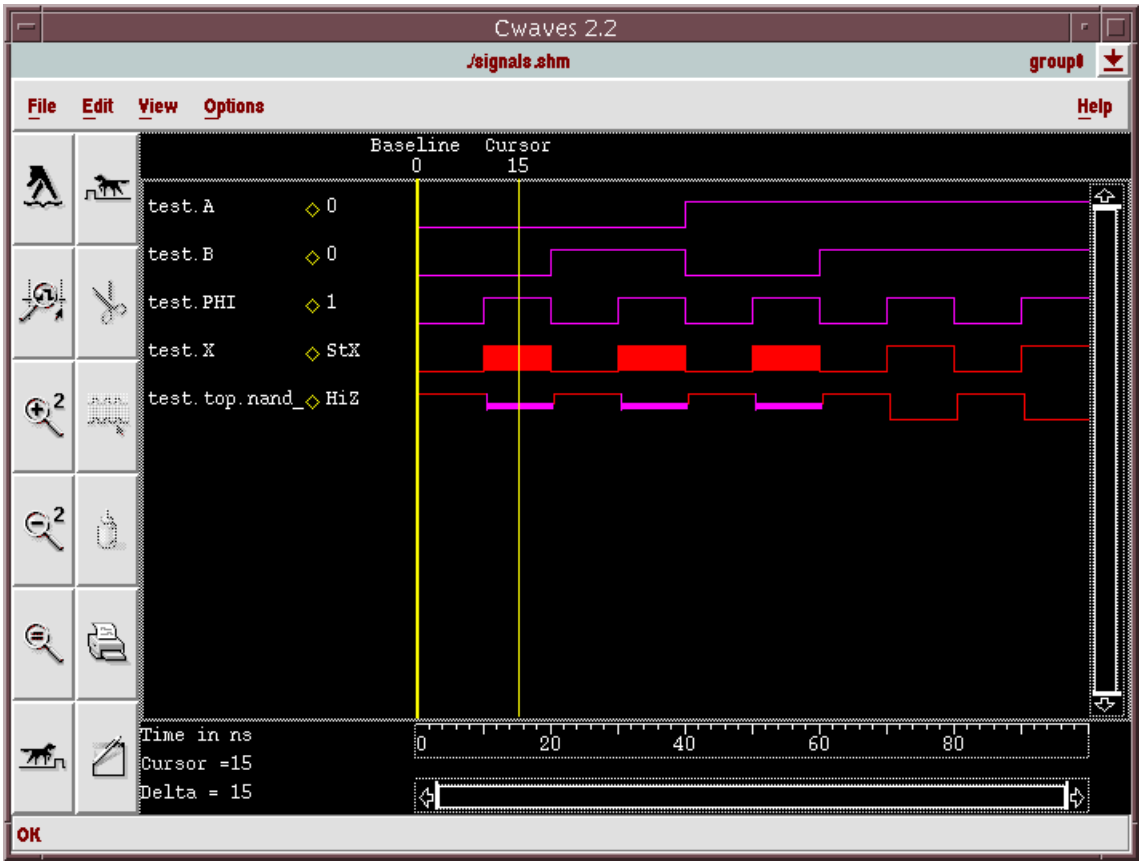
```
`timescale 1ns / 1ns
`include "../hdlFilesDir/cds_globals.v"
`include "../ihnl/cds0/netlist"
`include "../ihnl/cds1/netlist"
`include "../ihnl/cds2/netlist"
```

In *testfixture.verilog* file, type as follows.

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


// Default verilog stimulus.

initial
begin

   A = 1'b0;
   B = 1'b0;
   PHI = 1'b0;

   #20 A = 1'b0;
       B = 1'b1;
   #20 A = 1'b1;
       B = 1'b0;
   #20 A = 1'b1;
       B = 1'b1;
end

always
begin
    #10 PHI = ~PHI;
end

initial
begin
   $shm_open("signals.shm");
   $shm_probe("AS");
   #100 $shm_close();
   $finish;
end
```


jsmoon@ISI.EDU

Run Verilog simulation and see waveform using cwaves.
See if you get the following result.



As you can see, top.nand_ signal has high impedance value when PHI='1' (evaluation phase) and input A and B are not "11". Why? Physical wire *nand* has capacitance so that it stores precharged value when there is no pull-down path during evaluation phase. However, in verilog modeling, node *nand* goes to "Z" because both paths (pull-up and pull-down) are blocked. If A and B are "11", *nand* value is "0" because pull-down path is established. In next step, I will show how to resolve this using "trireg" type.

6. Verilog-XL Simulation Based on the Modified Netlist for Precharged Node

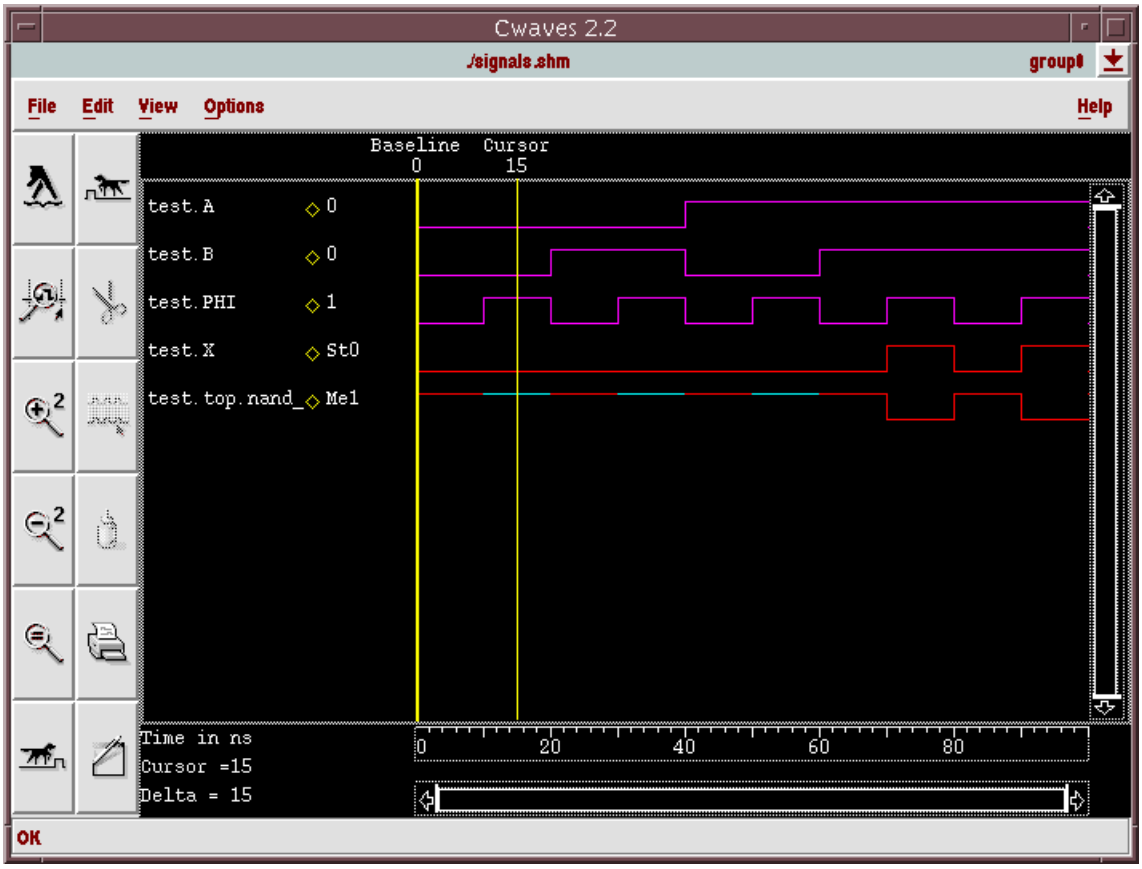
Try to find where you have netlist file under *ihnl* directory. In my case, *cds1/netlist* has netlist for dynamic nand2 gate.
Modify *netlist* file by adding line "*trireg X;*" as follows. (Do not touch any other lines!!)

```
// Library - Dynamic, Cell - dnand2, View - schematic
// LAST TIME SAVED: Dec  3 15:27:27 1998
// NETLIST TIME: Dec  3 15:35:33 1998
`timescale 1ns / 1ns

module dnand2 ( X, A, B, PHI );
output  X;
trireg  X;
```


jsmoon@ISI.EDU

trireg is one of the reg type which can hold the previous value. It can be interpreted as the wire with capacitance.

Run Verilog simulation again and see the waveform.



You can see that *top.nand_* file holds the precharged value correctly even if there is no pull-down path in evaluation phase. (It is Me1 - medium 1 strength)

* In summary, after you generate the netlist from schematic, you need to type “trireg nodename” inside netlist file wherever you want to define precharge node.
* Whenever Verilog-XL re-generate the netlist, your definition of trireg will be gone. So, make a copy of netlist file for dynamic gate.

I would recommend to include “*cds1/netlist.dyn*” in *testfixture.template* instead of *netlist*.

In *netlist.dyn*, you should have definition of trireg. (*netlist.dyn* won’t be overwritten so that you don’t lose trireg information.)