SIMULATION 13

Key terms and concepts: Engineers used to prototype systems to check designs • Breadboarding is feasible for systems constructed from a few TTL parts • It is impractical for an ASIC • Instead engineers turn to **simulation**

13.1 Types of Simulation

Key terms and concepts: **simulation modes** (high-level to low-level simulation–high-level is more abstract, low-level more detailed): Behavioral simulation • Functional simulation • Static timing analysis • Gate-level simulation • Switch-level simulation • Transistor-level or circuit-level simulation

13.2 The Comparator/MUX Example

Key terms and concepts: using **input vectors to** test or **exercise** a behavioral model • simulation can only prove a design does not work; it cannot prove that hardware will work

```
// comp_mux.v
module comp_mux(a, b, outp); input [2:0] a, b; output [2:0] outp;
function [2:0] compare; input [2:0] ina, inb;
begin if (ina <= inb) compare = ina; else compare = inb; end
endfunction
assign outp = compare(a, b);
endmodule
```

```
// testbench.v
module comp_mux_testbench;
integer i, j;
reg [2:0] x, y, smaller; wire [2:0] z;
always @(x) $display("t    x y actual calculated");
initial $monitor("%4g",$time,,x,,y,,z,,,,,,,smaller);
initial $dumpvars; initial #1000 $finish;
initial
```

```
begin
  for (i = 0; i <= 7; i = i + 1)
  begin
    for (j = 0; j <= 7; j = j + 1)
    begin
    x = i; y = j; smaller = (x <= y) ? x : y;
    #1 if (z  != smaller) $display("error");
    end
  end
end
comp_mux v_1 (x, y, z);
endmodule
```

13.2.1 Structural Simulation

Key terms and concepts: logic synthesis produces a structural model from a behavioral model • reference model • derived model • **vector-based simulation** (or **dynamic simulation**)

```
`timescale 1ns / 10ps // comp_mux_o2.v
module comp_mux_o (a, b, outp);
input  [2:0] a; input  [2:0] b;
output [2:0] outp;
supply1 VDD; supply0 VSS;
mx21d1 b1_i1 (.i0(a[0]), .i1(b[0]), .s(b1_i6_zn), .z(outp[0]));
oa03d1 b1_i2 (.a1(b1_i9_zn), .a2(a[2]), .b1(a[0]), .b2(a[1]),
  .c(b1_i4_zn), .zn(b1_i2_zn));
nd02d0 b1_i3 (.a1(a[1]), .a2(a[0]), .zn(b1_i3_zn));
nd02d0 b1_i4 (.a1(b[1]), .a2(b1_i3_zn), .zn(b1_i4_zn));
mx21d1 b1_i5 (.i0(a[1]), .i1(b[1]), .s(b1_i6_zn), .z(outp[1]));
oa04d1 b1_i6 (.a1(b[2]), .a2(b1_i7_zn), .b(b1_i2_zn),
  .zn(b1_i6_zn));
in01d0 b1_i7 (.i(a[2]), .zn(b1_i7_zn));
an02d1 b1_i8 (.a1(b[2]), .a2(a[2]), .z(outp[2]));
in01d0 b1_i9 (.i(b[2]), .zn(b1_i9_zn));
endmodule
```

```
`timescale 1 ns / 10 ps
module mx21d1 (z, i0, i1, s); input i0, i1, s; output z;
  not G3(N3, s);
  and G4(N4, i0, N3), G5(N5, s, i1), G6(N6, i0, i1);
  or G7(z, N4, N5, N6);
specify
  (i0*>z) = (0.279:0.504:0.900, 0.276:0.498:0.890);
```

```
  (i1*>z) = (0.248:0.448:0.800, 0.264:0.476:0.850);
  (s*>z)  = (0.285:0.515:0.920, 0.298:0.538:0.960);
endspecify
endmodule
```

```
`timescale 1 ps / 1 ps // comp_mux_testbench2.v
module comp_mux_testbench2;
integer i, j; integer error;
reg [2:0] x, y, smaller; wire [2:0] z, ref;
always @(x) $display("t       x y derived reference");
// initial $monitor("%8.2f",$time/1e3,,x,,y,,z,,,,,,,,ref);
initial $dumpvars;
initial begin
  error = 0; #1e6 $display("%4g", error, " errors");
  $finish;
end
initial begin
  for (i = 0; i <= 7; i = i + 1) begin
    for (j = 0; j <= 7; j = j + 1) begin
    x = i; y = j; #10e3;
    $display("%8.2f",$time/1e3,,x,,y,,z,,,,,,,,ref);
    if (z  != ref)
    begin $display("error"); error = error + 1;end
    end
  end
end
comp_mux_o v_1 (x, y, z); // comp_mux_o2.v
reference v_2 (x, y, ref);
endmodule
```

```
// reference.v
module reference(a, b, outp);
input [2:0] a, b;output [2:0] outp;
  assign outp = (a <= b) ? a : b; // different from comp_mux
endmodule
```

13.2.2 Static Timing Analysis

Key terms and concepts: “What is the longest delay in my circuit?” • timing analysis finds the critical path and its delay • timing analysis does not find the input vectors that activate the critical path • **Boolean relations** • **false paths** • a timing-analyzer is more logic calculator than logic simulator

13.2.3 Gate-Level Simulation

Key terms and concepts: differences between functional simulation, timing analysis, and gate-level simulation

```
# The calibration was done at Vdd=4.65V, Vss=0.1V, T=70 degrees C
Time = 0:0   [0 ns]
                            a  =  'D6  [0]  (input)(display)
                            b  =  'D7  [0]  (input)(display)
                         outp  =  'Buuu   ('Du)  [0]  (display)
                         outp --> 'B1uu   ('Du)  [.47]
                         outp --> 'B11u   ('Du)  [.97]
                         outp --> 'D6  [4.08]
                            a --> 'D7  [10]
                            b --> 'D6  [10]
                         outp --> 'D7  [10.97]
                         outp --> 'D6  [14.15]
  Time = 0:0 +20ns   [20 ns]
```

13.2.4 Net Capacitance

Key terms and concepts: **net capacitance** (interconnect capacitance or wire capacitance) • **wire-load model**, **wire-delay model**, or **interconnect model**

```
@nodes
a R10 W1; a[2] a[1] a[0]
b R10 W1; b[2] b[1] b[0]
outp R10 W1; outp[2] outp[1] outp[0]
@data
         .00              a -> 'D6
         .00              b -> 'D7
         .00           outp -> 'Du
         .53           outp -> 'Du
         .93           outp -> 'Du
        4.42           outp -> 'D6
       10.00              a -> 'D7
       10.00              b -> 'D6
       11.03           outp -> 'D7
       14.43           outp -> 'D6
### END OF SIMULATION TIME = 20 ns
@end
```

13.3 Logic Systems

Key terms and concepts: **Digital simulation** • **logic values** (or **logic states**) from a **logic system** • A **two-value logic system** (or two-state logic system) has logic value `'0'` (**logic level** 'zero') and a logic value `'1'` (logic level 'one') • logic value `'X'` (unknown logic level) or **unknown** • an unknown can **propagate** through a circuit • to model a three-state bus, we need a **high-impedance state** (logic level of 'zero' or 'one') but it is not being driven • A **four-value logic system**

A four-value logic system

Logic state	Logic level	Logic value
0	zero	zero
1	one	one
X	zero or one	unknown
Z	zero, one, or neither	high impedance

13.3.1 Signal Resolution

Key terms and concepts: **signal-resolution function** • commutative and associative

A resolution function R{A, B} that predicts the result of two drivers simultaneously attempting to drive signals with values A and B onto a bus

R{A, B}	B=0	B=1	B=X	B=Z
A=0	0	X	X	0
A=1	X	1	X	1
A=X	X	X	X	X
A=Z	0	1	X	Z

13.3.2 Logic Strength

Key terms and concepts: n-channel transistors produce a logic level 'zero' (with a forcing strength) • p-channel transistors force a logic level 'one' • An n-channel transistor provides a weak logic level 'one', a **resistive 'one'**, with **resistive strength** • **high impedance** • **Verilog logic system** • **VHDL signal resolution** using **VHDL signal-resolution functions**

A 12-state logic system

	Logic level		
Logic strength	**zero**	**unknown**	**one**
strong	S0	SX	S1
weak	W0	WX	W1
high impedance	Z0	ZX	Z1
unknown	U0	UX	U1

Verilog logic strengths

Logic strength	Strength number	Models	Abbreviation	
supply drive	7	power supply	supply	`Su`
strong drive	6	default gate and assign output strength	strong	`St`
pull drive	5	gate and assign output strength	pull	`Pu`
large capacitor	4	size of trireg net capacitor	large	`La`
weak drive	3	gate and assign output strength	weak	`We`
medium capacitor	2	size of trireg net capacitor	medium	`Me`
small capacitor	1	size of trireg net capacitor	small	`Sm`
high impedance	0	not applicable	highz	`Hi`

The nine-value logic system, IEEE Std 1164-1993.

Logic state	Logic value	Logic state	Logic value
`'0'`	strong low	`'X'`	strong unknown
`'1'`	strong high	`'W'`	weak unknown
`'L'`	weak low	`'Z'`	high impedance
`'H'`	weak high	`'-'`	don’t care
		`'U'`	uninitialized

```
function "and"(l,r : std_ulogic_vector)return std_ulogic_vector
is                                                            --1
  alias lv : std_ulogic_vector (1to l'LENGTH ) is l;          --2
  alias rv : std_ulogic_vector (1to r'LENGTH ) is r;          --3
variable result : std_ulogic_vector (1to l'LENGTH );          --4
```

```
constant and_table : stdlogic_table := (
-----------------------------------------------------
--|  U    X    0    1    Z    W    L    H    -       |   |
-----------------------------------------------------
  ( 'U', 'U', '0', 'U', 'U', 'U', '0', 'U', 'U' ), -- | U |
  ( 'U', 'X', '0', 'X', 'X', 'X', '0', 'X', 'X' ), -- | X |
  ( '0', '0', '0', '0', '0', '0', '0', 'U', '0' ), -- | 0 |
  ( 'U', 'X', '0', '1', 'X', 'X', '0', '1', 'X' ), -- | 1 |
  ( 'U', 'X', '0', 'X', 'X', 'X', '0', 'X', 'X' ), -- | Z |
  ( 'U', 'X', '0', 'X', 'X', 'X', '0', 'X', 'X' ), -- | W |
  ( '0', '0', '0', '0', '0', '0', '0', '0', '0' ), -- | L |
  ( 'U', 'X', '0', '1', 'X', 'X', '0', '1', 'X' ), -- | H |
  ( 'U', 'X', '0', 'X', 'X', 'X', '0', 'X', 'X' ), -- | - |);
begin
  if (l'LENGTH /= r'LENGTH) then assert false report
"arguments of overloaded 'and' operator are not of the same
length"
    severity failure;
  else
    for i in result'RANGE loop
      result(i) := and_table ( lv(i), rv(i) );
    end loop;
  end if;
  return result;
end "and";
```

13.4 How Logic Simulation Works

Key terms and concepts: event-driven simulator • event • event queue or event list • evaluation • **time step** • interpreted-code simulator • compiled-code simulator • native-code simulator • evaluation list • simulation cycle, or an event–evaluation cycle • time wheel

```
model nd01d1 (a, b, zn)
function (a, b) !(a & b); function end
model end
```

```
nand nd01d1(a2, b3, r7)
```

```
struct Event {
  event_ptr fwd_link, back_link; /* event list */
  event_ptr node_link; /* list of node events */
  node_ptr event_node; /* node for the event */
  node_ptr cause; /* node causing event */
  port_ptr port; /* port which caused this event */
  long event_time; /* event time, in units of delta */
  char new_value; /* new value: '1' '0' etc. */
};
```

13.4.1 VHDL Simulation Cycle

Key terms and concepts: **simulation cycle** • elaboration • a **delta cycle** takes **delta time** • **time step** • postponed processes

A VHDL simulation cycle consists of the following steps:

1. The current time, t_c is set equal to t_n.

2. Each active signal in the model is updated and events may occur as a result.

3. For each process P, if P is currently sensitive to a signal S, and an event has occurred on signal S in this simulation cycle, then process P resumes.

4. Each resumed process is executed until it suspends.

5. The time of the next simulation cycle, t_n, is set to the earliest of:
   a. the next time at which a driver becomes active or
   b. the next time at which a process resumes

6. If $t_n = t_c$, then the next simulation cycle is a delta cycle.

7. Simulation is complete when we run out of time (t_n = `TIME'HIGH`) and there are no active drivers or process resumptions at t_n

13.4.2 Delay

Key terms and concepts: **delay mechanism** • **transport delay** is characteristic of wires and transmission lines • **Inertial delay** models the behavior of logic cells • a logic cell will not transmit a pulse that is shorter than the switching time of the circuit, the default **pulse-rejection limit**

```
Op <= Ip after 10 ns;                                    --1
Op <= inertial Ip after 10 ns;                           --2
Op <= reject 10 ns inertial Ip after 10 ns;              --3
```

```
--  Assignments using transport delay:                   --1
Op <= transport Ip after 10 ns;                          --2
Op <= transport Ip after 10 ns, not Ip after 20 ns;      --3
```

```
--  Their equivalent assignments:                              --4
Op <= reject 0 ns inertial Ip after 10 ns;                     --5
Op <= reject 0 ns inertial Ip after 10 ns, not Ip after 10 ns;      --6
```

13.5 Cell Models

Key terms and concepts: delay model • power model • timing model • primitive model

There are several different kinds of logic cell models:

- Primitive models, produced by the ASIC library company and describe the function and properties of logic cells using primitive functions.
- Verilog and VHDL models produced by an ASIC library company from the primitive models.
- Proprietary models produced by library companies that describe small logic cells or functions such as microprocessors.

13.5.1 Primitive Models

Key terms and concepts: **primitive model** • a designer does not normally see a primitive model; it may only be used by an ASIC library company to generate other models

```
Function
(timingModel = oneOf("ism","pr"); powerModel = oneOf("pin"); )
Rec
Logic = Function (A1; A2;  )Rec ZN = not (A1 AND A2); End; End;
miscInfo = Rec Title = "2-Input NAND, 1X Drive"; freq_fact = 0.5;
tml = "nd02d1 nand 2 * zn a1 a2";
MaxParallel = 1; Transistors = 4; power = 0.179018;
Width = 4.2; Height = 12.6; productName = "stdcell35"; libraryName =
"cb35sc"; End;
Pin = Rec
A1 = Rec input;  cap = 0.010;  doc = "Data Input";  End;
A2 = Rec input;  cap = 0.010;  doc = "Data Input";  End;
ZN = Rec output;  cap = 0.009;  doc = "Data Output";  End; End;
Symbol = Select
timingModel
On pr Do Rec
tA1D_fr = |( Rec prop = 0.078; ramp = 2.749;  End);
tA1D_rf = |( Rec prop = 0.047; ramp = 2.506;  End);
tA2D_fr = |( Rec prop = 0.063; ramp = 2.750;  End);
tA2D_rf = |( Rec prop = 0.052; ramp = 2.507;  End); End
On ism Do Rec
```

```
tA1D_fr = |( Rec A0 = 0.0015; dA = 0.0789; D0 = -0.2828;
dD = 4.6642; B = 0.6879; Z = 0.5630; End );
tA1D_rf = |( Rec A0 = 0.0185; dA = 0.0477; D0 = -0.1380;
dD = 4.0678; B = 0.5329; Z = 0.3785; End );
tA2D_fr = |( Rec A0 = 0.0079; dA = 0.0462; D0 = -0.2819;
dD = 4.6646; B = 0.6856; Z = 0.5282; End );
tA2D_rf = |( Rec A0 = 0.0060; dA = 0.0464; D0 = -0.1408;
dD = 4.0731; B = 0.6152; Z = 0.4064; End ); End; End;
Delay = |( Rec from = pin.A1; to = pin.ZN;
edges = Rec fr = Symbol.tA1D_fr; rf = Symbol.tA1D_rf;  End; End, Rec
from = pin.A2; to = pin.ZN; edges = Rec fr = Symbol.tA2D_fr;rf =
Symbol.tA2D_rf;  End; End );
MaxRampTime = |( Rec check = pin.A1; riseTime = 3.000;  fallTime =
3.000;  End, Rec check = pin.A2; riseTime = 3.000;  fallTime =
3.000;  End, Rec check = pin.ZN; riseTime = 3.000;  fallTime =
3.000;  End );
DynamicPower = |( Rec rise = { ZN }; val = 0.003;  End); End; End
```

13.5.2 Synopsys Models

Key terms and concepts: **vendor models** • each logic cell is part of a file that also contains wire-load models and other characterization information for the cell library • not all of the information from a primitive model is present in a vendor model

```
cell (nd02d1) {
/* title : 2-Input NAND, 1X Drive */
/* pmd checksum : 'HBA7EB26C */
area : 1;
  pin(a1) { direction : input; capacitance : 0.088;
    fanout_load : 0.088; }
  pin(a2) { direction : input; capacitance : 0.087;
    fanout_load : 0.087; }
  pin(zn) { direction : output; max_fanout : 1.786;
  max_transition : 3; function : "(a1 a2)'";
  timing() {
    timing_sense : "negative_unate"
    intrinsic_rise : 0.24 intrinsic_fall : 0.17
    rise_resistance : 1.68 fall_resistance : 1.13
    related_pin : "a1" }
  timing() { timing_sense : "negative_unate"
    intrinsic_rise : 0.32 intrinsic_fall : 0.18
    rise_resistance : 1.68 fall_resistance : 1.13
```

```
    related_pin : "a2"
} } } /* end of cell */
```

13.5.3 Verilog Models

Key terms and concepts: Verilog timing models • SDF file contains back-annotation timing delays • delays are calculated by a **delay calculator** • `$sdf_annotate` performs back-annotation • **golden simulator**

```
`celldefine
`delay_mode_path
`suppress_faults
`enable_portfaults
`timescale 1 ns / 1 ps
module in01d1 (zn, i); input i; output zn; not G2(zn, i);
specify specparam
InCap$i = 0.060, OutCap$zn = 0.038, MaxLoad$zn = 1.538,
R_Ramp$i$zn = 0.542:0.980:1.750, F_Ramp$i$zn = 0.605:1.092:1.950;
specparam cell_count = 1.000000; specparam Transistors = 4 ;
specparam Power = 1.400000; specparam MaxLoadedRamp = 3 ;
  (i => zn) = (0.031:0.056:0.100, 0.028:0.050:0.090);
endspecify
endmodule
`nosuppress_faults
`disable_portfaults
`endcelldefine
```

```
`timescale 1 ns / 1 ps
module SDF_b; reg A; in01d1 i1 (B, A);
initial begin A = 0; #5; A = 1; #5; A = 0; end
initial $monitor("T=%6g",$realtime," A=",A," B=",B);
endmodule
```

```
T=     0 A=0 B=x
T= 0.056 A=0 B=1
T=     5 A=1 B=1
T=  5.05 A=1 B=0
T=    10 A=0 B=0
T=10.056 A=0 B=1
```

```
(DELAYFILE
  (SDFVERSION "3.0") (DESIGN "SDF.v") (DATE "Aug-13-96")
  (VENDOR "MJSS") (PROGRAM "MJSS") (VERSION "v0")
  (DIVIDER .) (TIMESCALE 1 ns)
  (CELL (CELLTYPE "in01d1")
    (INSTANCE SDF_b.i1)
    (DELAY (ABSOLUTE
      (IOPATH i zn (1.151:1.151:1.151) (1.363:1.363:1.363))
    ))
  )
)
```

```
`timescale 1 ns / 1 ps
module SDF_b; reg A; in01d1 i1 (B, A);
initial begin
$sdf_annotate ( "SDF_b.sdf", SDF_b, , "sdf_b.log", "minimum", , );
A = 0; #5; A = 1; #5; A = 0; end
initial $monitor("T=%6g",$realtime," A=",A," B=",B);
endmodule
```

Here is the output (from MTI V-System/Plus) including back-annotated timing:

```
T=     0 A=0 B=x
T= 1.151 A=0 B=1
T=     5 A=1 B=1
T= 6.363 A=1 B=0
T=    10 A=0 B=0
T=11.151 A=0 B=1
```

13.5.4 VHDL Models

Key terms and concepts: VHDL alone does not offer a standard way to perform back-annotation.

- VITAL

```
library IEEE; use IEEE.STD_LOGIC_1164.all;
library COMPASS_LIB; use COMPASS_LIB.COMPASS_ETC.all;
entity bknot is
  generic (derating : REAL := 1.0; Z1_cap : REAL := 0.000;
    INSTANCE_NAME : STRING := "bknot");
  port (Z2 : in Std_Logic; Z1 : out STD_LOGIC);
end bknot;
```

```
architecture bknot of bknot is
constant tplh_Z2_Z1 : TIME := (1.00 ns + (0.01 ns * Z1_Cap)) *
derating;
constant tphl_Z2_Z1 : TIME := (1.00 ns + (0.01 ns * Z1_Cap)) *
derating;
begin
  process(Z2)
  variable int_Z1 : Std_Logic := 'U';
  variable tplh_Z1, tphl_Z1, Z1_delay : time := 0 ns;
  variable CHANGED : BOOLEAN;
  begin
  int_Z1 := not (Z2);
  if Z2'EVENT then
    tplh_Z1 := tplh_Z2_Z1; tphl_Z1 := tphl_Z2_Z1;
  end if;
  Z1_delay := F_Delay(int_Z1, tplh_Z1, tphl_Z1);
  Z1 <= int_Z1 after Z1_delay;
  end process;
end bknot;
configuration bknot_CON of bknot is for bknot end for;
end bknot_CON;
```

13.5.5 VITAL Models

Key terms and concepts: VITAL • VHDL Initiative Toward ASIC Libraries, IEEE Std 1076.4 [1995] • . **sign-off** quality ASIC libraries using an approved cell library and a golden simulator

```
library IEEE; use IEEE.STD_LOGIC_1164.all;
use IEEE.VITAL_timing.all; use IEEE.VITAL_primitives.all;
entity IN01D1 is
  generic (
  tipd_I : VitalDelayType01 := (0 ns, 0 ns);
  tpd_I_ZN : VitalDelayType01 := (0 ns, 0 ns) );
  port (
  I : in STD_LOGIC := 'U';
  ZN : out STD_LOGIC := 'U' );
attribute VITAL_LEVEL0 of IN01D1 : entity is TRUE;
end IN01D1;
architecture IN01D1 of IN01D1 is
attribute VITAL_LEVEL1 of IN01D1 : architecture is TRUE;
signal I_ipd : STD_LOGIC := 'X';
begin
WIREDELAY:block
  begin VitalWireDelay(I_ipd, I, tipd_I);end block;
```

```
VITALbehavior : process (I_ipd)                                      --18
variable ZN_zd : STD_LOGIC;                                          --19
variable ZN_GlitchData : VitalGlitchDataType;                        --20
begin                                                                --21
ZN_zd := VitalINV(I_ipd);                                            --22
VitalPathDelay01(                                                    --23
  OutSignal => ZN,                                                   --24
  OutSignalName => "ZN",                                             --25
  OutTemp => ZN_zd,                                                  --26
  Paths => (0 => (I_ipd'LAST_EVENT, tpd_I_ZN, TRUE)),                --27
  GlitchData => ZN_GlitchData,                                       --28
  DefaultDelay => VitalZeroDelay01,                                  --29
  Mode => OnEvent,                                                   --30
  MsgOn => FALSE,                                                    --31
  XOn => TRUE,                                                       --32
  MsgSeverity => ERROR);                                             --33
  end process;                                                       --34
end IN01D1;                                                          --35
```

```
library IEEE; use IEEE.STD_LOGIC_1164.all;                            --1
entity SDF is port ( A : in STD_LOGIC; B : out STD_LOGIC );           --2
end SDF;                                                              --3
architecture SDF of SDF is                                            --4
component in01d1 port ( I : in STD_LOGIC; ZN : out STD_LOGIC );      --5
end component;                                                        --6
  begin i1: in01d1 port map ( I => A, ZN => B);                       --7
end SDF;                                                              --8
```

```
library STD; use STD.TEXTIO.all;                                      --1
library IEEE; use IEEE.STD_LOGIC_1164.all;                            --2
entity SDF_testbench is end SDF_testbench;                            --3
architecture SDF_testbench of SDF_testbench is                        --4
component SDF port ( A : in STD_LOGIC; B : out STD_LOGIC );           --5
end component;                                                        --6
signal A, B : STD_LOGIC := '0';                                       --7
begin                                                                 --8
  SDF_b : SDF port map ( A => A, B => B);                             --9
  process begin                                                      --10
    A <= '0'; wait for 5 ns; A <= '1';                               --11
    wait for 5 ns; A <= '0'; wait;                                   --12
  end process;                                                       --13
  process (A, B) variable L: LINE; begin                             --14
    write(L, now, right, 10, TIME'(ps));                             --15
    write(L, STRING'(" A=")); write(L, TO_BIT(A));                   --16
    write(L, STRING'(" B=")); write(L, TO_BIT(B));                   --17
    writeline(output, L);                                            --18
```

```
  end process;                                                        --19
end SDF_testbench;                                                    --20
```

```
(DELAYFILE
  (SDFVERSION "3.0") (DESIGN "SDF.vhd") (DATE "Aug-13-96")
  (VENDOR "MJSS") (PROGRAM "MJSS") (VERSION "v0")
  (DIVIDER .) (TIMESCALE 1 ns)
  (CELL (CELLTYPE "in01d1")
    (INSTANCE i1)
    (DELAY (ABSOLUTE
      (IOPATH i zn (1.151:1.151:1.151) (1.363:1.363:1.363))
      (PORT i (0.021:0.021:0.021) (0.025:0.025:0.025))
    ))
  )
)
```

```
<msmith/MTI/vital> vsim -c -sdfmax /sdf_b=SDF_b.sdf sdf_testbench
...
#       0 ps A=0 B=0
#       0 ps A=0 B=0
#    1176 ps A=0 B=1
#    5000 ps A=1 B=1
#    6384 ps A=1 B=0
#   10000 ps A=0 B=0
#   11176 ps A=0 B=1
```

13.5.6 SDF in Simulation

Key terms and concepts: SDF is also used to describe forward-annotation of timing constraints from logic synthesis

```
(DELAYFILE
  (SDFVERSION "1.0")
  (DESIGN "halfgate_ASIC_u")
  (DATE "Aug-13-96")
  (VENDOR "Compass")
  (PROGRAM "HDL Asst")
  (VERSION "v9r1.2")
  (DIVIDER .)
  (TIMESCALE 1 ns)
```

```
  (CELL (CELLTYPE "in01d0")
    (INSTANCE v_1.B1_i1)
    (DELAY (ABSOLUTE
      (IOPATH I ZN (1.151:1.151:1.151) (1.363:1.363:1.363))
    ))
  )
  (CELL (CELLTYPE "pc5o06")
    (INSTANCE u1_2)
    (DELAY (ABSOLUTE
      (IOPATH I PAD (1.216:1.216:1.216) (1.249:1.249:1.249))
    ))
  )
  (CELL (CELLTYPE "pc5d01r")
    (INSTANCE u0_2)
    (DELAY (ABSOLUTE
      (IOPATH PAD CIN (.169:.169:.169) (.199:.199:.199))
    ))
  )
)
```

```
(DELAYFILE
  ...
  (PROCESS "FAST-FAST")
  (TEMPERATURE 0:55:100)
  (TIMESCALE 100ps)
(CELL (CELLTYPE "CHIP")
  (INSTANCE TOP)
  (DELAY (ABSOLUTE
    (INTERCONNECT A.INV8.OUT B.DFF1.Q (:0.6:) (:0.6:))
)))
```

```
  (INSTANCE B.DFF1)
  (DELAY (ABSOLUTE
    (IOPATH (POSEDGE CLK) Q (12:14:15) (11:13:15))))
```

```
(DELAYFILE
(DESIGN "MYDESIGN")
(DATE "26 AUG 1996")
  (VENDOR "ASICS_INC")
  (PROGRAM "SDF_GEN")
(VERSION "3.0")
  (DIVIDER .)
```

```
  (VOLTAGE 3.6:3.3:3.0)
  (PROCESS "-3.0:0.0:3.0")
  (TEMPERATURE 0.0:25.0:115.0)
(TIMESCALE )
(CELL
  (CELLTYPE "AOI221")
  (INSTANCE X0)
  (DELAY (ABSOLUTE
  (IOPATH A1 Y (1.11:1.42:2.47) (1.39:1.78:3.19))
  (IOPATH A2 Y (0.97:1.30:2.34) (1.53:1.94:3.50))
  (IOPATH B1 Y (1.26:1.59:2.72) (1.52:2.01:3.79))
  (IOPATH B2 Y (1.10:1.45:2.56) (1.66:2.18:4.10))
  (IOPATH C1 Y (0.79:1.04:1.91) (1.36:1.62:2.61))
))))
```

13.6 Delay Models

Key terms and concepts: timing model describes delays outside logic cells • delay model describes delays inside logic cells • **pin-to-pin delay** is a delay between an input pin and an output pin of a logic cell • **pin delay** is a delay lumped to a certain pin of a logic cell (usually an input) • **net delay** or **wire delay** is a delay outside a logic cell • **prop–ramp delay model**

```
specify specparam                                                    //1
InCap$i = 0.060, OutCap$zn = 0.038, MaxLoad$zn = 1.538,             //2
R_Ramp$i$zn = 0.542:0.980:1.750, F_Ramp$i$zn = 0.605:1.092:1.950;  //3
specparam cell_count = 1.000000; specparam Transistors = 4 ;        //4
specparam Power = 1.400000; specparam MaxLoadedRamp = 3 ;           //5
  (i=>zn)=(0.031:0.056:0.100, 0.028:0.050:0.090);                   //6
```

13.6.1 Using a Library Data Book

Key terms and concepts: **area-optimized library** (small) • **performance-optimized library** (fast)

Input capacitances for an inverter family (pF)

Library	inv1	invh	invs	inv8	inv12
Area	0.034	0.067	0.133	0.265	0.397
Performance	0.145	0.292	0.584	1.169	1.753

Delay information for a 2:1 MUX

		Propagation delay			
		Area		Performance	
From input	To output	Extrinsic/ $nspF^{-1}$	Intrinsic / ns	Extrinsic / ns	Intrinsic / ns
D0\	Z\	2.10	1.42	0.5	0.8
D0/	Z/	3.66	1.23	0.68	0.70
D1\	Z\	2.10	1.42	0.50	0.80
D1/	Z/	3.66	1.23	0.68	0.70
SD\	Z\	2.10	1.42	0.50	0.80
SD\	Z/	3.66	1.09	0.70	0.73
SD/	Z\	2.10	2.09	0.5	1.09
SD/	Z/	3.66	1.23	0.68	0.70

Process derating factors

Process	Derating factor
Slow	1.31
Nominal	1.0
Fast	0.75

Temperature and voltage derating factors

	Supply voltage				
Temperature/°C	4.5V	4.75V	5.00V	5.25V	5.50V
–40	0.77	0.73	0.68	0.64	0.61
0	1.00	0.93	0.87	0.82	0.78
25	1.14	1.07	1.00	0.94	0.90
85	1.50	1.40	1.33	1.26	1.20
100	1.60	1.49	1.41	1.34	1.28
125	1.76	1.65	1.56	1.47	1.41

13.6.2 Input-Slope Delay Model

Key terms and concepts: submicron technologies must account for the effects of the rise (and fall) time of the input waveforms to a logic cell • nonlinear delay model

The input-slope model predicts delay in the fast-ramp region, D_{ISM} (50 %, FR), as follows (0.5 trip points):

D_{ISM} (50%, FR)

$= A_0 + D_0 C_L + 0.5 O_R = A_0 + D_0 C_L + d_A/2 + d_D C_L/2$

$= 0.0015 + 0.5 \times 0.0789 + (-0.2828 + 0.5 \times 4.6642)\ C_L$

$= 0.041 + 2.05 C_L$

13.6.3 Limitations of Logic Simulation

Key terms and concepts: pin-to-pin delay model • timing information for most gate-level simulators is calculated once, before simulation • **state-dependent timing**

Switching characteristics of a two-input NAND gate

		Fanout					
Symbol	**Parameter**	**FO = 0 /ns**	**FO = 1 /ns**	**FO = 2 /ns**	**FO = 4 /ns**	**FO = 8 /ns**	**K /nspF^{-1}**
t_{PLH}	Propagation delay, A to X	0.25	0.35	0.45	0.65	1.05	1.25
t_{PHL}	Propagation delay, B to X	0.17	0.24	0.30	0.42	0.68	0.79
t_r	Output rise time, X	1.01	1.28	1.56	2.10	3.19	3.40
t_f	Output fall time, X	0.54	0.69	0.84	1.13	1.71	1.83

Switching characteristics of a half adder

Symbol	Parameter	Fanout FO = 0 /ns	FO = 1 /ns	FO = 2 /ns	FO = 4 /ns	FO = 8 /ns	K /nspF^{-1}
t_{PLH}	Delay, A to S (B = '0')	0.58	0.68	0.78	0.98	1.38	1.25
t_{PHL}	Delay, A to S (B = '1')	0.93	0.97	1.00	1.08	1.24	0.48
t_{PLH}	Delay, B to S (B = '0')	0.89	0.99	1.09	1.29	1.69	1.25
t_{PHL}	Delay, B to S (B = '1')	1.00	1.04	1.08	1.15	1.31	0.48
t_{PLH}	Delay, A to CO	0.43	0.53	0.63	0.83	1.23	1.25
t_{PHL}	Delay, A to CO	0.59	0.63	0.67	0.75	0.90	0.48
t_r	Output rise time, X	1.01	1.28	1.56	2.10	3.19	3.40
t_f	Output fall time, X	0.54	0.69	0.84	1.13	1.71	1.83

13.7 Static Timing Analysis

Key terms and concepts: static timing analysis • pipelining • critical path

```
Instance name        in pin-->out pin    tr      total   incr    cell
--------------------------------------------------------------------
END_OF_PATH
outp_2_                                  R       27.26
OUT1                : D--->PAD           R       27.26   7.55    OUTBUF
I_1_CM8             : S11--->Y           R       19.71   4.40    CM8
I_2_CM8             : S11--->Y           R       15.31   5.20    CM8
I_3_CM8             : S11--->Y           R       10.11   4.80    CM8
IN1                 : PAD--->Y           R       5.32    5.32    INBUF
a_2_                                     R       0.00    0.00
BEGIN_OF_PATH
```

```
// comp_mux_rrr.v                                                      //1
module comp_mux_rrr(a, b, clock, outp);                                //2
input [2:0] a, b; output [2:0] outp;input clock;                       //3
reg [2:0] a_r, a_rr, b_r, b_rr, outp;reg sel_r;                        //4
wire sel = ( a_r <= b_r ) ? 0 : 1;                                     //5
always @ (posedge clock) begin a_r <= a; b_r <= b; end                 //6
always @ (posedge clock) begin a_rr <= a_r; b_rr <= b_r;end            //7
always @ (posedge clock) outp <= sel_r ? b_rr : a_rr;                  //8
always @ (posedge clock) sel_r <= sel;                                 //9
endmodule                                                             //10
```

```
---------------------INPAD to SETUP longest path---------------------
Rise delay, Worst case
Instance name        in pin-->out pin    tr      total   incr    cell
--------------------------------------------------------------------
```

```
END_OF_PATH
D.a_r_ff_b2                             R       4.52    0.00    DF1
INBUF_24          : PAD--->Y            R       4.52    4.52    INBUF
a_2_                                    R       0.00    0.00
BEGIN_OF_PATH

---------------------CLOCK to SETUP longest path--------------------
Rise delay, Worst case

Instance name       in pin-->out pin    tr      total   incr    cell
--------------------------------------------------------------------
END_OF_PATH
D.sel_r_ff                              R       9.99    0.00    DF1
I_1_CM8           : S10--->Y            R       9.99    0.00    CM8
I_3_CM8           : S00--->Y            R       9.99    4.40    CM8
a_r_ff_b1         : CLK--->Q            R       5.60    5.60    DF1
BEGIN_OF_PATH


---------------------CLOCK to OUTPAD longest path--------------------
Rise delay, Worst case

Instance name       in pin-->out pin    tr      total   incr    cell
--------------------------------------------------------------------
END_OF_PATH
outp_2_                                 R       11.95
OUTBUF_31         : D--->PAD            R       11.95   7.55    OUTBUF
outp_ff_b2        : CLK--->Q            R       4.40    4.40    DF1
BEGIN_OF_PATH
```

A timing analyzer examines the following types of paths:

1. An **entry path** (or input-to-D path) to a pipelined design. The longest **entry delay** (or input-to-setup delay) is 4.52 ns.

2. A **stage path** (register-to-register path or clock-to-D path) in a pipeline stage. The longest **stage delay** (clock-to-D delay) is 9.99 ns.

3. An **exit path** (clock-to-output path) from the pipeline. The longest **exit delay** (clock-to-output delay) is 11.95 ns.

13.7.1 Hold Time

Key terms and concepts: Hold-time problems occur if there is clock skew between adjacent flip-flops • To check for hold-time violations we find the clock skew for each clock-to-D path

```
timer> shortest
 1st shortest path to all endpins
Rank Total Start pin      First Net     End Net       End pin
  0    4.0 b_rr_ff_b1:CLK b_rr_1_       DEF_NET_48    outp_ff_b1:D
  1    4.1 a_rr_ff_b2:CLK a_rr_2_       DEF_NET_46    outp_ff_b2:D
... 8 similar lines omitted ...
```

13.7.2 Entry Delay

Key terms and concepts: Before we can measure clock skew, we need to analyze the entry delays, including the clock tree

13.7.3 Exit Delay

Key terms and concepts: exit delays (the longest path between clock-pad input and an output) • critical path and operating frequency

13.7.4 External Setup Time

Key terms and concepts: external set-up time • internal set-up time • clock delay

Each of the six chip data inputs must satisfy the following set-up equation:

t_{SU} (external) > t_{SU} (internal) – (clock delay) + (data delay

13.8 Formal Verification

Key terms and concepts: logic synthesis converts a behavioral model to a structural model • How do we know that the two are the same? • **formal verification** can prove they are equivalent

13.8.1 An Example

Key terms and concepts: **reference model** • **derived model** • (1) the HDL is parsed • (2) a **finite-state machine compiler** extracts the states • (3) a **proof generator** automatically generates formulas to be proved • (4) the **theorem prover** attempts to prove the formulas

```
entity Alarm is                                                       --1
  port(Clock, Key, Trip : in bit; Ring : out bit);                    --2
end Alarm;                                                            --3
```

```
architecture RTL of Alarm is                                          --1
  type States is (Armed, Off, Ringing); signal State : States;        --2
begin                                                                 --3
  process (Clock) begin                                               --4
  if Clock = '1' and Clock'EVENT then                                 --5
    case State is                                                     --6
      when Off => if Key = '1' then State <= Armed; end if;           --7
      when Armed => if Key = '0' then State <= Off;                   --8
                elsif Trip = '1' then State <= Ringing;               --9
                end if;                                               --10
```

```
      when Ringing => if Key = '0' then State <= Off; end if;
    end case;
  end if;
  end process;
  Ring <= '1' when State = Ringing else '0';
end RTL;
```

```
library cells; use cells.all; // ...contains logic cell models
architecture Gates of Alarm is
component Inverter port(i : in BIT;z : out BIT) ; end component;
component NAnd2 port(a,b : in BIT;z : out BIT) ; end component;
component NAnd3 port(a,b,c : in BIT;z : out BIT) ; end component;
component DFF port(d,c : in BIT; q,qn : out BIT) ; end component;
signal State, NextState : BIT_VECTOR(1downto 0);
signal s0, s1, s2, s3 : BIT;
begin
  g2: Inverter port map ( i => State(0), z => s1 );
  g3: NAnd2 port map ( a => s1, b => State(1), z => s2 );
  g4: Inverter port map ( i => s2, z => Ring );
  g5: NAnd2 port map ( a => State(1), b => Key, z => s0 );
  g6: NAnd3 port map ( a => Trip, b => s1, c => Key, z => s3 );
  g7: NAnd2 port map ( a => s0, b => s3, z => NextState(1) );
  g8: Inverter port map ( i => Key, z => NextState(0) );
  state_ff_b0: DFF port map
  ( d => NextState(0), c => Clock, q => State(0), qn => open );
  state_ff_b1: DFF port map
  ( d => NextState(1), c => Clock, q => State(1), qn => open );
end Gates;
```

13.8.2 Understanding Formal Verification

Key terms and concepts: The **formulas** to be proved are generated as **proof statements** • An **axiom** is an explicit or implicit fact (signal of type `BIT` may only be `'0'` and `'1'`) • An **assertion** is derived from a statement placed in the HDL code • **implication** • **equivalence**

```
assert Key /= '1' or Trip /= '1' or NextState = Ringing
  report "Alarm on and tripped but not ringing";
```

Implication and equivalence

A	**B**	**A B**	**A ⇔ B**
F	F	T	T
F	T	T	F
T	F	F	F
T	T	T	T

13.8.3 Adding an Assertion

Key terms and concepts: “The axioms of the reference model do not imply that the assertions of the reference model imply the assertions of the derived model.” Translation: “These two architectures differ in some way.”

```
<E>  Assertion may be violated
SEVERITY: ERROR
REPORT: Alarm on and tripped but not ringing
FILE: .../alarm-rtl3.vhdl
FSM: alarm-rtl3
STATEMENT or DECLARATION: line8
.../alarm-rtl3.vhdl (line 8)
Context of the message is:
(key And trip And memoryofdriver__state(0))
```

```
    case State is                                                   --1
      when Off => if Key = '1' then State <= Armed; end if;         --2
      when Armed => if Key = '0' then State <= Off;                 --3
                  elsif Trip = '1' then State <= Ringing;           --4
                  end if;                                           --5
      when Ringing => if Key = '0' then State <= Off; end if;       --6
    end case;                                                       --7
```

```
Prove (Axiom_ref => (Assert_ref => Assert_der))
Formula is NOT VALID
But is VALID under Assert Context of alarm-rtl3
```

13.8.4 Completing a Proof

```
...
case State is
  when Off =>             if Key = '1' then
              if Trip = '1' then NextState <= Ringing;
              else NextState <= Armed;
              end if;
            end if;
  when Armed => if Key = '0' then NextState <= Off;
            elsif Trip = '1' then NextState <= Ringing;
            end if;
  when Ringing => if Key = '0' then NextState <= Off; end if;
end case;
...
```

13.9 Switch-Level Simulation

Key terms and concepts: The **switch-level simulator** is a more detailed level of simulation than we have discussed so far • Example: a true single-phase flip-flop using true single-phase clocking (TSPC)

13.10 Transistor-Level Simulation

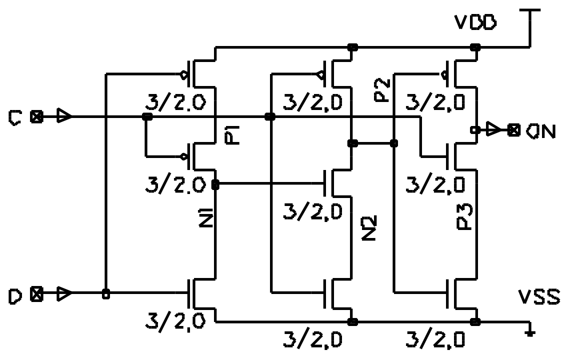
Key terms and concepts: **transistor-level simulation** or **circuit-level simulation** • **SPICE** (or **Spice**, **Simulation Program with Integrated Circuit Emphasis**) developed at UC Berkeley

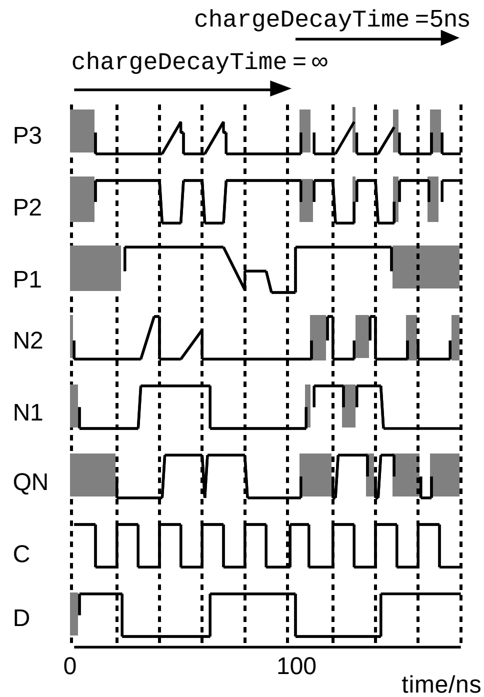
13.10.1 A PSpice Example

Key terms and concepts: **PSpice** input **deck**

```
OB September 5, 1996 17:27
.TRAN/OP 1ns 20ns
.PROBE
 cl  output  Ground  10pF
 VIN  input  Ground   PWL(0us 5V 10ns 5V 12ns 0V 20ns 0V)
 VGround  0  Ground   DC 0V
 Vdd  +5V  0   DC 5V
 m1  output  input  Ground Ground NMOS W=100u L=2u
```

(a) **(b)**





A TSPC (true single-phase clock) flip-flop

(a) The schematic (all devices are W/L=3/2)

(b) The switch-level simulation results

The parameter `chargeDecayTime` sets the time after which the simulator sets an undriven node to an invalid logic level (shown shaded).

```
 m2  output  input  +5V +5V PMOS W=200u L=2u
.model nmos nmos level=2 vto=0.78 tox=400e-10 nsub=8.0e15 xj=-0.15e-6
+ ld=0.20e-6 uo=650 ucrit=0.62e5 uexp=0.125 vmax=5.1e4 neff=4.0
+ delta=1.4 rsh=37 cgso=2.95e-10 cgdo=2.95e-10 cj=195e-6 cjsw=500e-12
+ mj=0.76 mjsw=0.30 pb=0.80
.model pmos pmos level=2 vto=-0.8 tox=400e-10 nsub=6.0e15 xj=-0.05e-6
+ ld=0.20e-6 uo=255 ucrit=0.86e5 uexp=0.29 vmax=3.0e4 neff=2.65
+ delta=1 rsh=125 cgso=2.65e-10 cgdo=2.65e-10 cj=250e-6 cjsw=350e-12
+ mj=0.535 mjsw=0.34 pb=0.80
.end
```

13.10.2 SPICE Models

Key terms and concepts: SPICE parameters • LEVEL=3 parameters

SPICE transistor model parameters (LEVEL=3)

parameter	n-ch. value	p-ch. value	Units	Explanation
CGBO	4.0E-10	3.8E-10	Fm^{-1}	Gate–bulk overlap capacitance (CGBoh, not CGBzero)
CGDO	3.0E-10	2.4E-10	Fm^{-1}	Gate–drain overlap capacitance (CGDoh, not CGDzero)
CGSO	3.0E-10	2.4E-10	Fm^{-1}	Gate–source overlap capacitance (CGSoh, not CGSzero)
CJ	5.6E-4	9.3E-4	Fm^{-2}	Junction area capacitance
CJSW	5E-11	2.9E-10	Fm^{-1}	Junction sidewall capacitance
DELTA	0.7	0.29	m	Narrow-width factor for adjusting threshold voltage
ETA	3.7E-2	2.45E-2	1	Static-feedback factor for adjusting threshold voltage
GAMMA	0.6	0.47	$\mathrm{V}^{0.5}$	Body-effect factor
KAPPA	2.9E-2	8	V^{-1}	Saturation-field factor (channel-length modulation)
KP	2E-4	4.9E-5	AV^{-2}	Intrinsic transconductance (μC_{ox}, not $0.5\mu C_{ox}$)
LD	5E-8	3.5E-8	m	Lateral diffusion into channel
LEVEL	3		none	Empirical model
MJ	0.56	0.47	1	Junction area exponent
MJSW	0.52	0.50	1	Junction sidewall exponent
NFS	6E11	6.5E11	$\mathrm{cm}^{-2}\mathrm{V}^{-1}$	Fast surface-state density
NSUB	1.4E17	8.5E16	cm^{-3}	Bulk surface doping
PB	1	1	V	Junction area contact potential
PHI	0.7		V	Surface inversion potential
RSH	2		Ω/square	Sheet resistance of source and drain
THETA	0.27	0.29	V^{-1}	Mobility-degradation factor
TOX	1E-8		m	Gate-oxide thickness
TPG	1	-1	none	Type of polysilicon gate
U0	550	135	$\mathrm{cm}^{2}\mathrm{V}^{-1}\mathrm{s}^{-1}$	Low-field bulk carrier mobility (Uzero, not Uoh)
XJ	0.2E-6		m	Junction depth
VMAX	2E5	2.5E5	ms^{-1}	Saturated carrier velocity
VTO	0.65	-0.92	V	Zero-bias threshold voltage (VTzero, not VToh)

PSpice parameters for process G5 (PSpice `LEVEL=4`)

```
.MODEL NM1 NMOS LEVEL=4
+ VFB=-0.7, LVFB=-4E-2, WVFB=5E-2
+ PHI=0.84, LPHI=0, WPHI=0
+ K1=0.78, LK1=-8E-4, WK1=-5E-2
+ K2=2.7E-2, LK2=5E-2, WK2=-3E-2
+ ETA=-2E-3, LETA=2E-02, WETA=-5E-3
+ MUZ=600, DL=0.2, DW=0.5
+ U0=0.33, LU0=0.1, WU0=-0.1
+ U1=3.3E-2, LU1=3E-2, WU1=-1E-2
+ X2MZ=9.7, LX2MZ=-6, WX2MZ=7
+ X2E=4.4E-4, LX2E=-3E-3, WX2E=9E-4
+ X3E=-5E-5, LX3E=-2E-3, WX3E=-1E-3
+ X2U0=-1E-2, LX2U0=-1E-3, WX2U0=5E-3
+ X2U1=-1E-3, LX2U1=1E-3, WX2U1=-7E-4
+ MUS=700, LMUS=-50, WMUS=7
+ X2MS=-6E-2, LX2MS=1, WX2MS=4
+ X3MS=9, LX3MS=2, WX3MS=-6
+ X3U1=9E-3, LX3U1=2E-4, WX3U1=-5E-3
+ TOX=1E-2, TEMP=25, VDD=5
+ CGDO=3E-10, CGSO=3E-10, CGBO=4E-10
+ XPART=1
+ N0=1, LN0=0, WN0=0
+ NB=0, LNB=0, WNB=0
+ ND=0, LND=0, WND=0
* n+ diffusion
+ RSH=2.1, CJ=3.5E-4, CJSW=2.9E-10
+ JS=1E-8, PB=0.8, PBSW=0.8
+ MJ=0.44, MJSW=0.26, WDF=0
*, DS=0
```

```
.MODEL PM1 PMOS LEVEL=4
+ VFB=-0.2, LVFB=4E-2, WVFB=-0.1
+ PHI=0.83, LPHI=0, WPHI=0
+ K1=0.35, LK1=-7E-02, WK1=0.2
+ K2=-4.5E-2, LK2=9E-3, WK2=4E-2
+ ETA=-1E-2, LETA=2E-2, WETA=-4E-4
+ MUZ=140, DL=0.2, DW=0.5
+ U0=0.2, LU0=6E-2, WU0=-6E-2
+ U1=1E-2, LU1=1E-2, WU1=7E-4
+ X2MZ=7, LX2MZ=-2, WX2MZ=1
+ X2E= 5E-5, LX2E=-1E-3, WX2E=-2E-4
+ X3E=8E-4, LX3E=-2E-4, WX3E=-1E-3
+ X2U0=9E-3, LX2U0=-2E-3, WX2U0=2E-3
+ X2U1=6E-4, LX2U1=5E-4, WX2U1=3E-4
+ MUS=150, LMUS=10, WMUS=4
+ X2MS=6, LX2MS=-0.7, WX2MS=2
+ X3MS=-1E-2, LX3MS=2, WX3MS=1
+ X3U1=-1E-3, LX3U1=-5E-4, WX3U1=1E-3
+ TOX=1E-2, TEMP=25, VDD=5
+ CGDO=2.4E-10, CGSO=2.4E-10, CGBO=3.8E-
10
+ XPART=1
+ N0=1, LN0=0, WN0=0
+ NB=0, LNB=0, WNB=0
+ ND=0, LND=0, WND=0
* p+ diffusion
+ RSH=2, CJ=9.5E-4, CJSW=2.5E-10
+ JS=1E-8, PB=0.85, PBSW=0.85
+ MJ=0.44, MJSW=0.24, WDF=0
*, DS=0
```

13.11 Summary

Key terms and concepts: Behavioral simulation can only tell you only if your design will not work • Prelayout simulation estimates of performance • Finding a critical path is difficult because you need to construct input vectors to exercise the model • Static timing analysis is the most widely used form of simulation • Formal verification compares two different representations. It cannot prove your design will work • Switch-level simulation can check the behavior of circuits that may not always have nodes that are driven or that use logic that is not complementary • Transistor-level simulation is used when you need to know the analog, rather than the digital, behavior of circuit voltages • trade-off in accuracy against run time