Avant!

Chapter 20

Signal Integrity

The performance of an IC design is no longer limited to how many million transistors a vendor fits on a single chip. With tighter packaging space and increasing clock frequencies, packaging and system level performance issues such as crosstalk and transmission lines are becoming increasingly significant.

At the same time, with the popularity of multichip packages and increased I/O counts, package design itself is becoming more and more like chip design.

This chapter describes how to maintain signal integrity for you design, and covers the following topics:

Preparing for Simulation

To simulate a PC board or backplane with Star-Hspice, you must consider models for:

- A driver cell, including the parasitic pin capacitances and package lead inductances
- Transmission lines
- A receiver cell with its parasitic pin capacitances and package lead inductances
- Terminations or other electrical elements on the line

It is important to model the transmission line as closely as possible— that is, to include all electrical elements exactly as they are laid out on the backplane or printed circuit board, to maintain the integrity of the simulation.

With readily available I/O drivers from ASIC vendors and Star-Hspice's advanced lossy transmission lines, you can simulate the electrical behavior of the board interconnect, bus, or backplane to analyze the transmission line behavior under various conditions.

Simulation is possible because the critical models and simulation technology exist.

- Many manufacturers of high-peed components use Star-Hspice already.
- The complexity can be hidden from the systems level.
- The necessary electrical characteristics are preserved with full transistor level library circuits.

Star-Hspice has been enhanced for systems simulation with:

- Systems level behavior, such as local component temperature and independent models, to allow accurate prediction of electrical behavior
- Automatic inclusion of library components via the SEARCH option
- Lossy transmission line models that:
- Support common mode simulation
- Include ground plane reactance

- Include resistive loss of conductor and ground plane
- Allow multiple signal conductors
- Require minimum CPU computation time

The following vendor models are currently available in Star-Hspice:

- Signetics FAST Library
- Xilinx 3000/4000 Series FPGA
- Intel's Peripheral Component Interconnect (PCI) high-speed local bus

Signal Integrity Problems

Some signal integrity problems that can cause failures in high-speed designs are listed in Table 20-1:.

Table 20-1: High-Speed Design Problems and Solutions

Signal Integrity Problem	Causes	Solution
Noise: delta I (current)	Multiple simultaneously switching drivers; high-speed devices create larger delta I	Adjust or evaluate location, size, and value of decoupling capacitors.
Noise: coupled (crosstalk)	Closely spaced parallel traces	Establish parallel line length design rules.
Noise: reflective	Impedance mismatch	Reduce the number of connectors and select proper impedance connectors.
Delay: path length	Poor placement and routing; too many or too few layers; chip pitch	Choose MCM or other high density packaging technology.
Propagation speed	Dielectric medium	Choose dielectric with lowest dielectric constant.
Delay: rise time degradation	Resistive loss and impedance mismatch	Adjust width, thickness and length of line.

Analog Side of Digital Logic

Circuit simulation of a digital system only becomes necessary when the analog characteristics of the digital signals become electrically important. Is the digital circuit a new design, or simply a fast version of the old design? Many new digital products are really faster versions of existing designs. The transition from a 100 MHz to a 150 MHz Pentium PC may not require extensive logic simulations. However, the integrity of the digital quality of the signals may require very careful circuit analysis.

The source of a signal integrity problem is the digital output driver. A high speed digital output driver can only drive a few inches before the noise and delay due to the wiring become a problem. To speed up circuit simulation and modeling, you can create analog behavioral models that mimic the full analog characteristics at a fraction of the traditional evaluation time. The simulation of the output buffer in Figure 20-1: demonstrates the analog behavior of a digital gate simulated in the Star-Hspice circuit simulator.

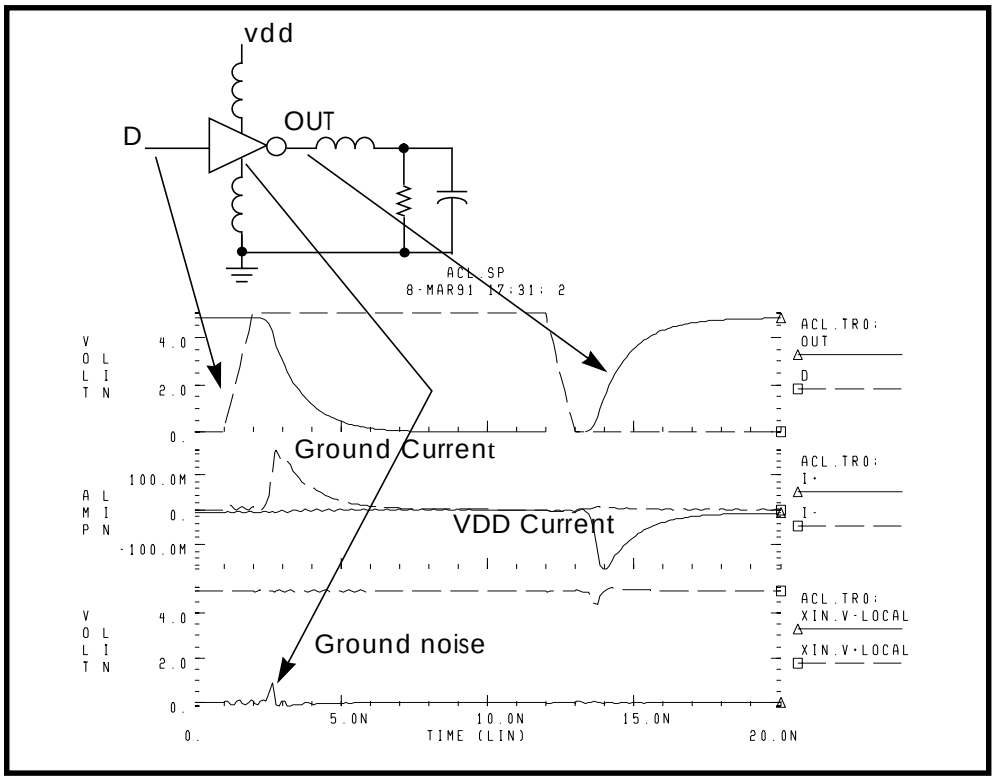

Figure 20-1: Simulation of Output Buffer with 2 ns Delay and 1.8 ns Rise and Fall Times

The roadblocks to successful high-speed digital designs are noise and signal delays. Digital noise can come from several sources. The fundamental digital noise sources are:

- Line termination noise
- Ground bounce noise
- Coupled line noise

Line termination noise is the additional voltage that is reflected from the load back to the driver because of impedance mismatch. Digital output buffers are not designed to have accurately controlled output impedance and most buffers have different rising and falling edge impedances.

Ground bounce noise is generated where leadframes or other circuit wires cannot be formed into transmission lines. The resulting inductance creates an induced voltage in the ground circuit, the supply circuit, and the output driver circuit. The ground bounce noise lowers the noise margins for the rest of the system. Coupled line noise is the noise induced from lines that are physically adjacent. This noise is generally most severe for data lines that are next to clock lines.

Circuit delays become critical as timing requirements become tighter. The key circuit delays are:

- Gate delays
- Line turnaround delays for tristate buffers
- Line length delays (clock skew)

Logic analysis only addresses gate delays. You can compute the variation in the gate delay from circuit simulation only if you understand the best case and worst case manufacturing conditions. The line turnaround delays add to the gate delays because extra margin must be added so that multiple tristate buffer drivers do not simultaneously turn on. The line length delay affects the clock skew most directly in most systems. As system cycle times approach the speed of electromagnetic signal propagation for the printed circuit board, consideration of the line length becomes critical. The system noises and line delays interact with the electrical characteristics of the gates and may require circuit level simulation.

Analog details find digital systems problems. Exceeding the noise quota may not cause a system to fail. Only when a digital input is being accepted does the maximum noise become a problem. If a digital systems engineer can decouple the system, much higher noise can be accepted.

Common decoupling methods are:

- Multiple ground and power planes on the PCB, MCM, PGA
- Separating signal traces with ground traces
- Decoupling capacitors
- Series resistors on output buffer drivers
- Twisted pair line driving

In present systems designs, you must select the best packaging methods at the printed circuit board level, the multi-chip module level, and the pin grid array level. Extra ground and power planes are often necessary to lower the supply inductance and provide decoupling. Decoupling capacitors must have very low internal inductance in order to be effective for high speed designs. Newer designs frequently use series resistance in the output drivers to lower circuit ringing. Finally, in critical high speed driver applications, twisted differential pair transmission lines are used.

The systems engineer must determine how to partition the logic. The propagation speed of signals on a printed circuit board is about 6 in/ns. As digital designs become faster, the wiring interconnect becomes a factor in deciding how to partition the logic. The critical wiring systems are:

- IC level wiring
- Package wiring for SIPs, DIPs, PGAs, MCMs
- Printed circuit board wiring
- Backplane and connector wiring
- Long lines – power, coax, twisted pair

Systems designers who use ASIC or custom integrated circuits as part of their system logic partitioning strategy find that they must make decisions about integrated circuit level wiring. The more familiar decisions involve the selection of packages and the arrangement of packages on a printed circuit board. Large systems generally have a central backplane that becomes the primary challenge at the system partition level.

Use the following equation to estimate wire length when transmission line effects become noticeable:

```
critical length=(rise time)*velocity/8
```

For example, if rise time is 1 ns and board velocity is 6 in/ns, distortion becomes noticeable at a wire length of 3/4 in. The Star-Hspice circuit simulator automatically generates models for each type of wire to define full loss transmission line effects.

ECL logic design engineers typically partitioned the system by calculating the noise quota for each line. Now, most high speed digital logic must be designed with respect to the noise quota so that the engineer knows how much noise and delay can be accepted before the timing and logic levels fail.

To solve the noise quota problem, you must calculate the noise associated with the wiring. Large integrated circuits can be separated into two parts: the internal logic and the external input and output amplifiers.

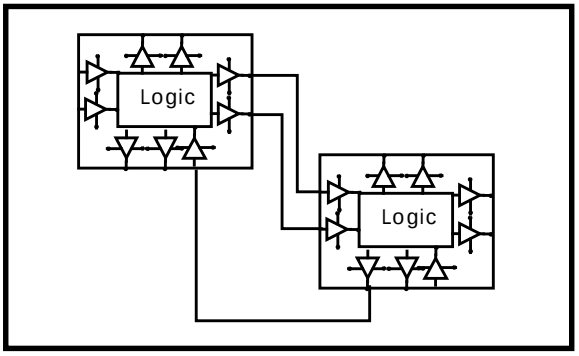

Figure 20-2: Analog Drivers and Wires

Using mixed digital and analog tools such as Avant!'s Star-Hspice and Viewlogic's Viewsim A/D, you can merge a complete system together with full analog quality timing constraints and full digital representation. You can simultaneously evaluate noise quota calculation subject to system timing.

Optimizing TDR Packaging

Packaging plays an important role in determining the overall speed, cost, and reliability of a system. With today's small feature sizes and high levels of integration, a significant portion of the total delay comes from the time required for a signal to travel between chips.

Multilayer ceramic technology has proven to be well suited for high speed GaAs IC packages.

The multichip module minimizes the chip-to-chip spacing and reduces the inductive and capacitive discontinuity between the chips mounted on the substrate with a more direct path (die-bump-interconnect-bump-die), thus eliminating wirebonding. In addition, narrower and shorter wires on the ceramic substrate have much less capacitance and inductance than the PC board interconnections.

Time domain reflectometry (TDR) is the closest measurement to actual digital component function. It provides a transient display of the impedance versus time for pulse behavior.

With a digitized TDR file, you can automatically select design components using the Star-Hspice optimizer. You can extract critical points from digitized TDR files using the Star-Hspice .MEASURE statement, and use the results as electrical specifications for optimization. This process eliminates recurring design cycles to find component values to curve-fit the TDR files.

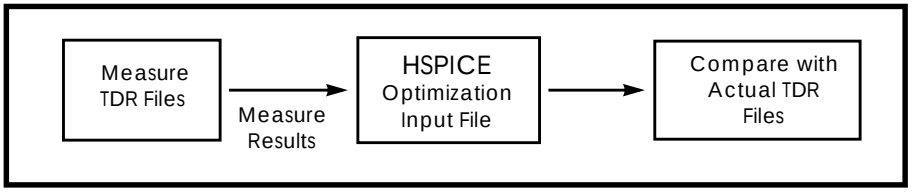

Figure 20-3: Optimization Process

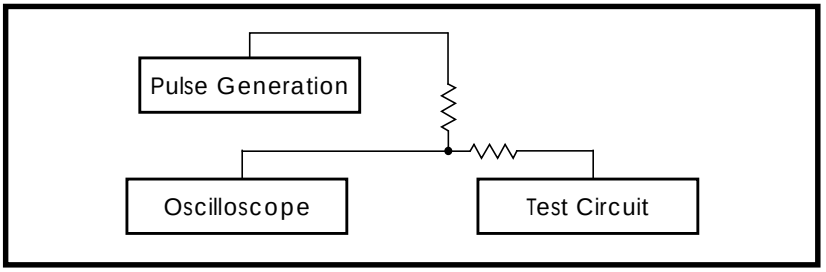

Figure 20-4: General Method for TDR Optimization

The following is a method used for realistic high-speed testing of packaging.

Test fixtures were designed that closely emulate a high-speed system environment. A Star-Hspice model was constructed for measurements using ideal transmission lines and discrete components.

The circuit tested contained the following components:

- Signal generator
- Coax connecting the signal generator to ETF (engineering test fixture) board
- ETF board
- Package pins
- Package body

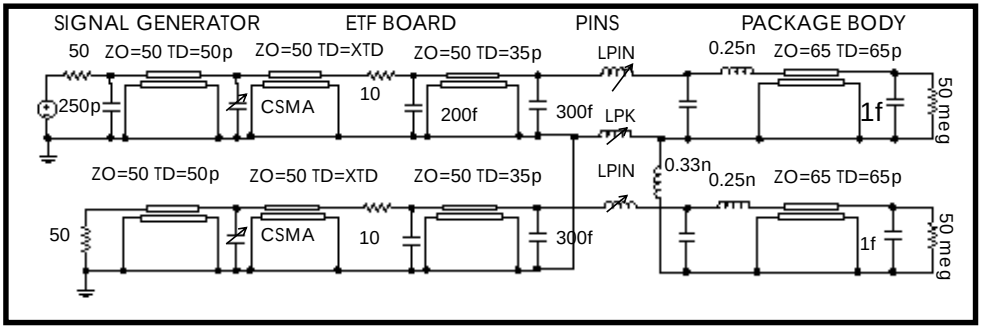

Figure 20-5: SPICE Model for Package-Plus-Test Fixture Optimized Parameters: XTD, CSMA, LPIN, and LPK

The package tests performed traditional time domain measurements using a digital sampling oscilloscope. Tests were designed to observe the reflected and transmitted signals derived from the built-in high-speed pulse generator and translated output signals into digitized time domain reflectometer files (voltage vs. time).

A fully developed SPICE model was used to simulate the package-plus-test fixture. The simulated and measured reflected/transmitted signals were compared.

The input netlist file for this experiment is shown on the following pages. Output plots are shown in Figures 20-6 through 20-9.

You can further investigate this experiment using Star-Hspice's advanced lossy transmission lines to include attenuation and dispersion.

TDR Optimization Procedure

Measure Critical Points In The TDR Files

```
Vin 1 0 PWL(TIME,VOLT)
.DATA D_TDR
    TIME    VOLT
    0       0.5003mV
    0.1n    0.6900mV
     ...
    2.0n    6.4758mV
.ENDDATA
.TRAN DATA=D_TDR
.MEAS .....
.END
```

Set up an Input Optimization File

```
$ SPICE MODEL FOR PACKAGE-PLUS-TEST FIXTURE
$ AUTHOR: DAVID H. SMITH & RAJ M. SAVARA
.OPTION POST RELV=1E-4 RELVAR=1E-2

$ DEFINE PARAMETERS
.PARAM LV=-0.05 HV=0.01 TD=1P TR=25P TF=50P TPW=10N TPER=15N
```

```
$ PARAMETERS TO BE OPTIMIZED
.PARAM CSMA=OPT1(500f,90f,900f,5f)
+      XTD=OPT1(150p,100p,200p)
+      LPIN=OPT1(0.65n,0.10n,0.90n,0.2n)
+      LPK=OPT1(1.5n,0.75n,3.0n,0.2n)
+      LPKCL=0.33n
+      LPKV=0.25n
```

Signal Generator

```
VIN   S1 GND PULSE LV HV TD TR TF TPW TPER
RIN   S1 S2  50
CIN1  S2 GND 250f
TCOAX S2 GND SIG_OUT GND ZO=50 TD=50p
```

ETF Board

```
CSNAL SIG_OUT GND CSMA
TEFT2 SIG_OUT GND E3 GND ZO=50 TD=XTD
RLOSS1 E3 E4 10
CRPAD1 E4 GND 200f
TLIN2 E4 GND ETF_OUT GND ZO=50 TD=35p
CPAD2 ETF_OUT GND 300f
TLIN1 E5 GND E6 GND ZO=50 TD=35p
CPAD1 E5 GND 300f
CRPAD2 E6 GND 200f
RLOS1 E6 E7 10
TEFT1 E7 GND E8 GND ZO=50 TD=XTD
CSMA2 E8 GND CSMA
TCOAX2 E8 GND VREF1 GND ZO=50 TD=50p
RIN1 VREF1 GND 50
```

Package Body

```
LPIN1 ETF_OUT P1 LPIN
LPK1 GND P5 LPK
LPKGCL P5 NVOUT2 LPKCL
CPKG1 P1 P5 250f
LPKV1 P1 P2 LPKV
TPKGL P2 NVOUT2 VOUT NVOUT2 ZO=65 TD=65P
CBPL VOUT NVOUT 1f
ROUT1 VOUT NVOUT 50meg
LPIN2 E5 P3 LPIN
```

```
CPKG2 P3 NVOUT2 250f
LPKV2 P3 P4 LPKV
TPKG2 P4 NVOUT2 VOUT2 NVOUT2 ZO=65 TD=65p
CBPD1 VOUT2 NVOUT2 1f
ROUT2 VOUT2 NVOUT2 50meg

$ BEFORE OPTIMIZATION
.TRAN .004NS 2NS

$ OPTIMIZATION SETUP
.MODEL OPTMOD OPT ITROPT=30

.TRAN .05NS 2NS SWEEP OPTIMIZE=OPT1
+                RESULTS=MAXV,MINV,MAX_2,COMP1,PT1,PT2,PT3
+                MODEL=OPTMOD

$ MEASURE CRITICAL POINTS IN THE REFLECTED SIGNAL
$ GOALS ARE SELECTED FROM MEASURED TDR FILES

.MEAS TRAN COMP1 MIN V(S2) FROM=100p TO=500p GOAL=-27.753
.MEAS TRAN PT1 FIND V(S2) AT=750p GOAL=-3.9345E-3 WEIGHT=5
.MEAS TRAN PT2 FIND V(S2) AT=775p GOAL=2.1743E-3 WEIGHT=5
.MEAS TRAN PT3 FIND V(S2) AT=800p GOAL=5.0630E-3 WEIGHT=5

$ MEASURE CRITICAL POINTS IN THE TRANSMITTED SIGNAL
$ GOALS ARE SELECTED FROM MEASURED TDR FILES

.MEAS TRAN MAXV FIND V(VREF1)AT=5.88E-10 GOAL=6.3171E-
+WEIGHT=7
.MEAS TRAN MINV FIND V(VREF1) AT=7.60E-10 GOAL=-9.9181E-3
.MEAS TRAN MAX_2 FIND V(VREF1) AT=9.68E-10 GOAL=4.9994E-3

$ COMPARE SIMULATED RESULTS WITH MEASURED TDR VALUES

.TRAN .004NS 2NS
.PRINT C_REF=V(S2) C_TRAN=V(VREF1)

.END
```

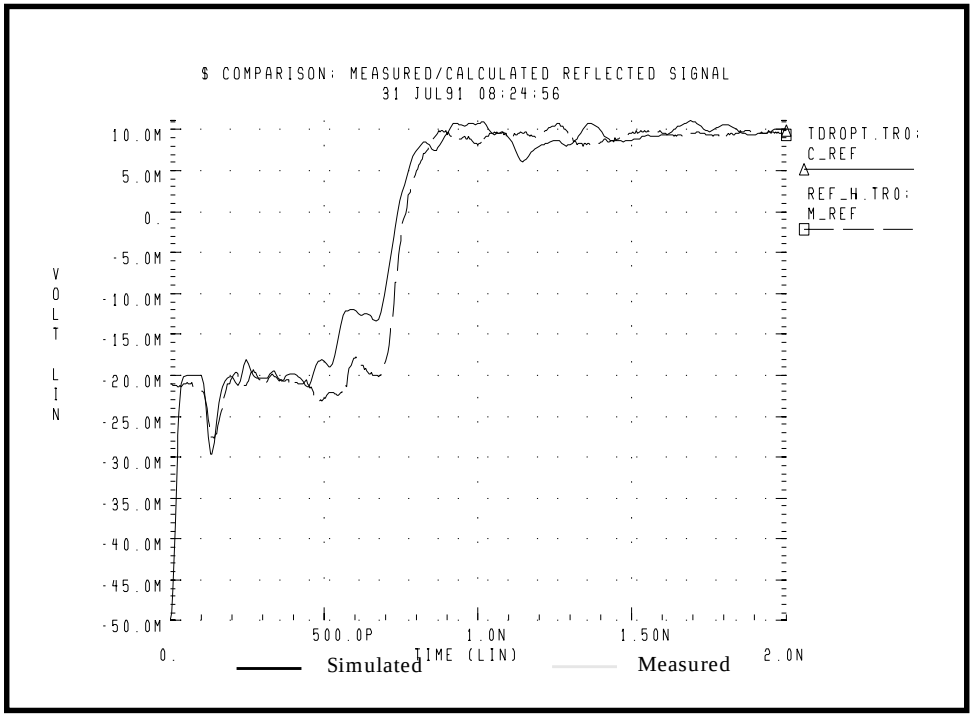

Figure 20-6: Reflected Signals Before Optimization

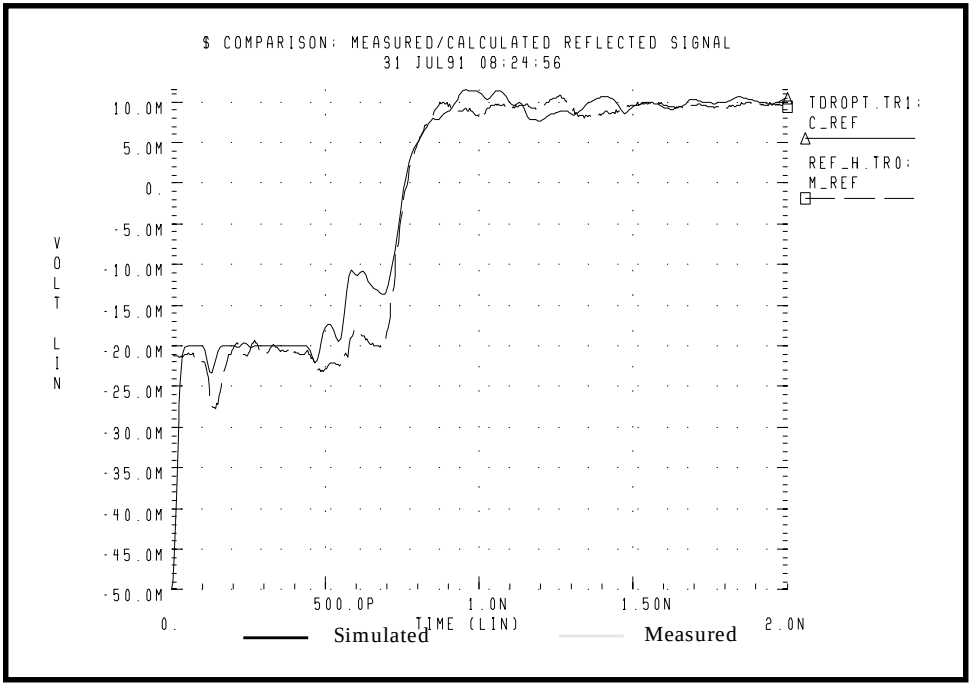

Figure 20-7: Reflected Signals After Optimization

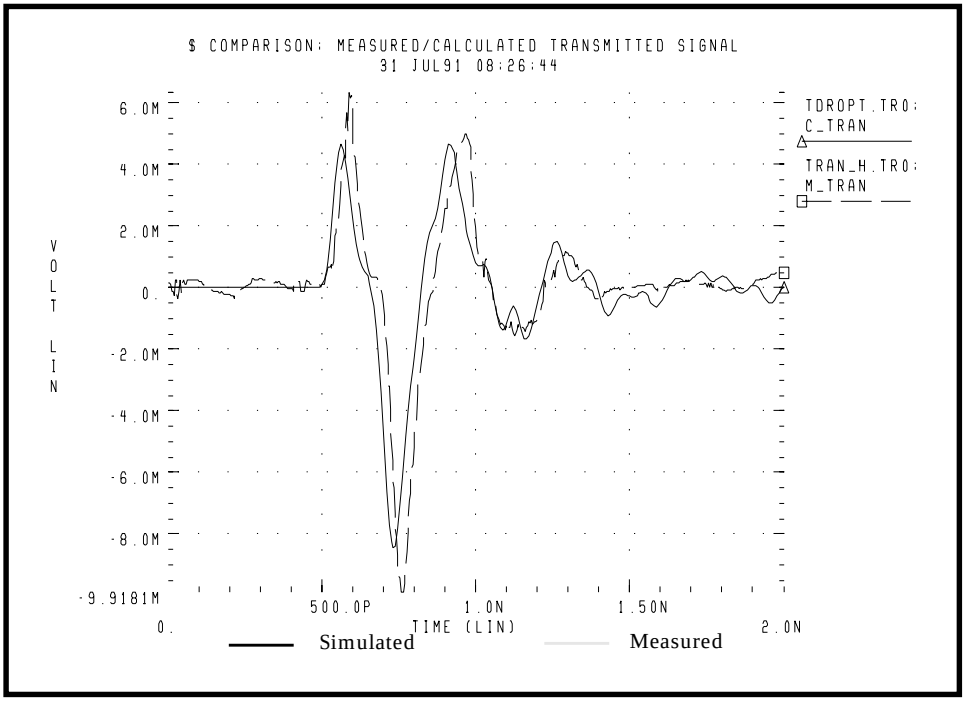

Figure 20-8: Transmitted Signals Before Optimization

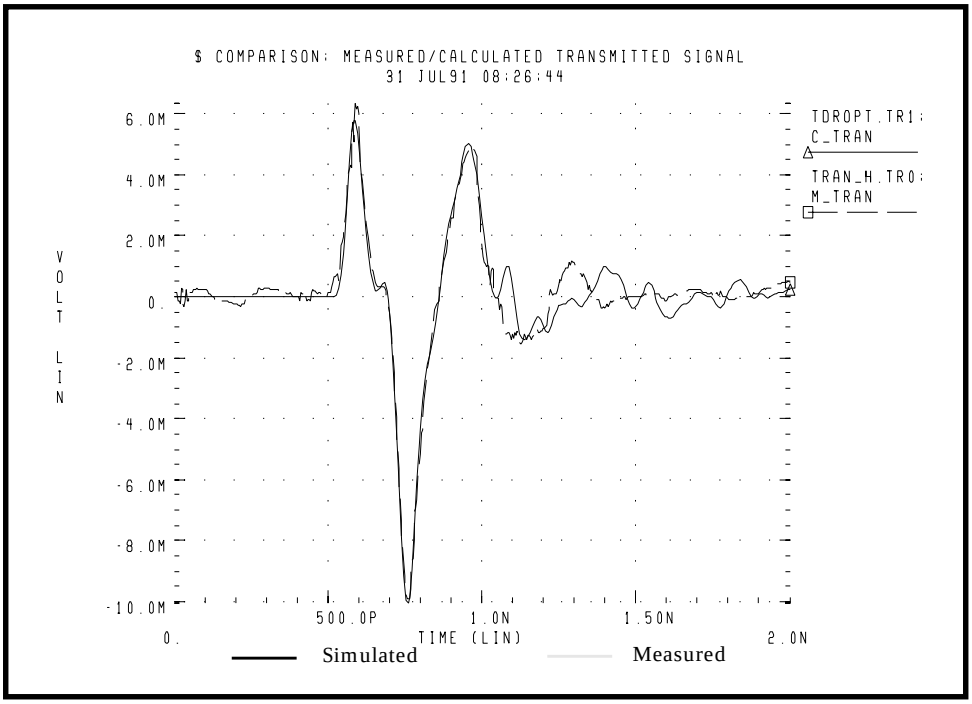

Figure 20-9: Transmitted Signals after Optimization

Simulating Circuits with Signetics Drivers

The Signetics I/O buffer library is distributed with Star-Hspice in the *$installdir/parts/signet* directory. These are high performance parts used in backplane design. The transmission line model used describes two conductors.

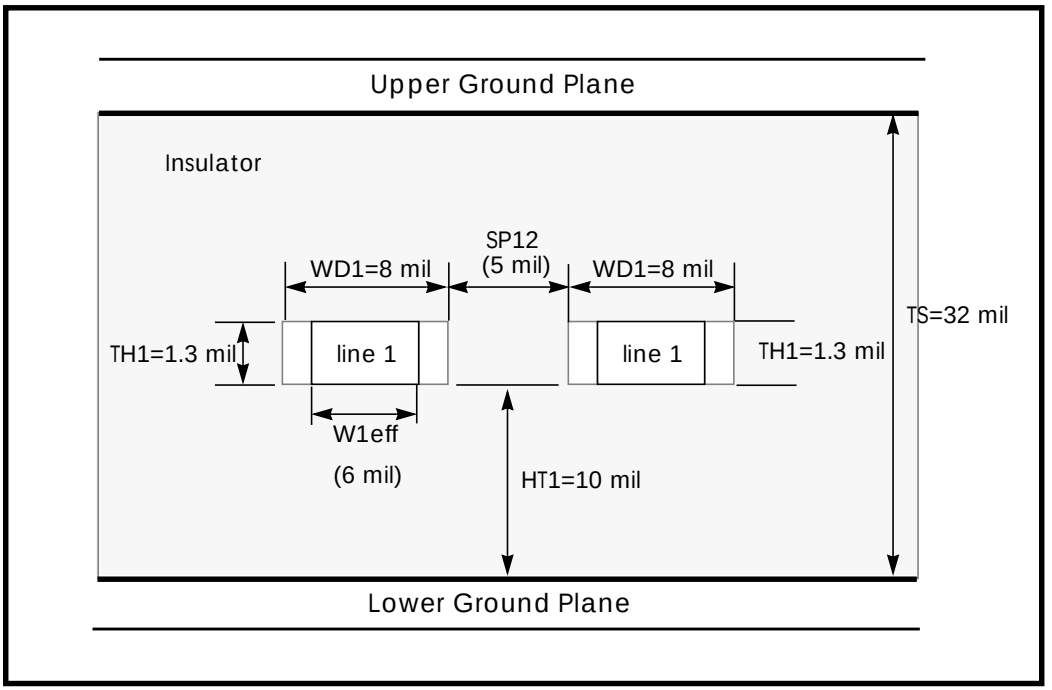

Figure 20-10: Planar Transmission Line DLEV=0: Microstrip Sea of Dielectric

In the following application, a pair of drivers are driving about 2.5 inches of adjacent lines to a pair of receivers that drive about four inches of line.

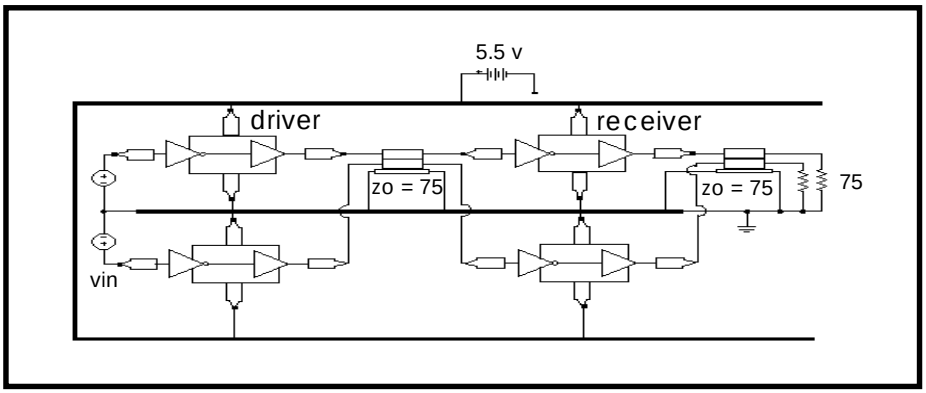

Figure 20-11: I/O Drivers/Receivers with Package Lead Inductance, Parallel 4" Lossy Microstrip Connectors

Example of Connecting I/O Chips With Tlines

```
.OPTIONS SEARCH='$installdir/parts/signet'
.OPTIONS POST=2 TNOM=27 NOMOD LIST METHOD=GEAR
.TEMP 27

$ DEFINE PARAMETER VALUES
.PARAM LV=0 HV=3 TD1=10n TR1=3n TF1=3n TPW=20n TPER=100n
+ TD2=20n TR2=2n TF2=2n LNGTH=101.6m

$ POWER SUPPLY
VCC VCC 0 DC 5.5

$ INPUT SOURCES
VIN1 STIM1 0 PULSE LV HV TD1 TR1 TF1 TPW TPER
VIN2 STIM2 0 PULSE LV HV TD2 TR2 TF2 TPW TPER

$ FIRST STAGE: DRIVER WITH TLINE
X1ST_TOP STIM1 OUTPIN1 VCC GND IO_CHIP PIN_IN=2.6n PIN_OUT=4.6n
X1ST_DN STIM2 OUTPIN2 VCC GND IO_CHIP PIN_IN=2.9n PIN_OUT=5.6n
U_1ST OUTPIN1 OUTPIN2 GND TLOUT1 TLOUT2 GND USTRIP L=LNGTH

$ SECOND STAGE: RECEIVER WITH TLINE

X2ST_TOP TLOUT1 OUTPIN3 VCC GND IO_CHIP PIN_IN=4.0n PIN_OUT=2.5n
X2ST_DN TLOUT2 OUTPIN4 VCC GND IO_CHIP PIN_IN=3.6n PIN_OUT=5.1n
U_2ST OUTPIN3 OUTPIN4 GND TLOUT3 TLOUT4 GND USTRIP L=LNGTH
```

```
$ TERMINATING RESISTORS
R1 TLOUT3 GND 75
R2 TLOUT4 GND 75
$ IO CHIP MODEL - SIGNETICS
.SUBCKT IO_CHIP IN OUT VCC XGND PIN_VCC=7n PIN_GND=1.8n

X1 IN1    INVOUT VCC1 XGND1 ACTINPUT
X2 INVOUT OUT1   VCC1 XGND1 AC109EQ
```

Package Inductance

```
LIN_PIN IN IN1 PIN_IN
LOUT_PIN OUT1 OUT PIN_OUT
LVCC VCC VCC1 PIN_VCC
LGND XGND1 XGND PIN_GND
.ENDS

$ TLINE MODEL - 2 SIGNAL CONDUCTORS WITH GND
$ PLANE

.MODEL USTRIP U LEVEL=3 ELEV=1 PLEV=1
+ TH1=1.3mil HT1=10mil TS=32mil KD1=4.5 DLEV=0 WD1=8mil
+ XW=-2mil KD2=4.5 NL=2 SP12=5mil
$ ANALYSIS / PRINTS
.TRAN .1NS 100NS
.GRAPH IN1=V(STIM1) IN2=V(STIM2) VOUT1=V(TLOUT1) VOUT2=V(TLOUT2)
.GRAPH VOUT3=V(TLOUT3) VOUT4=V(TLOUT4)
.END
```

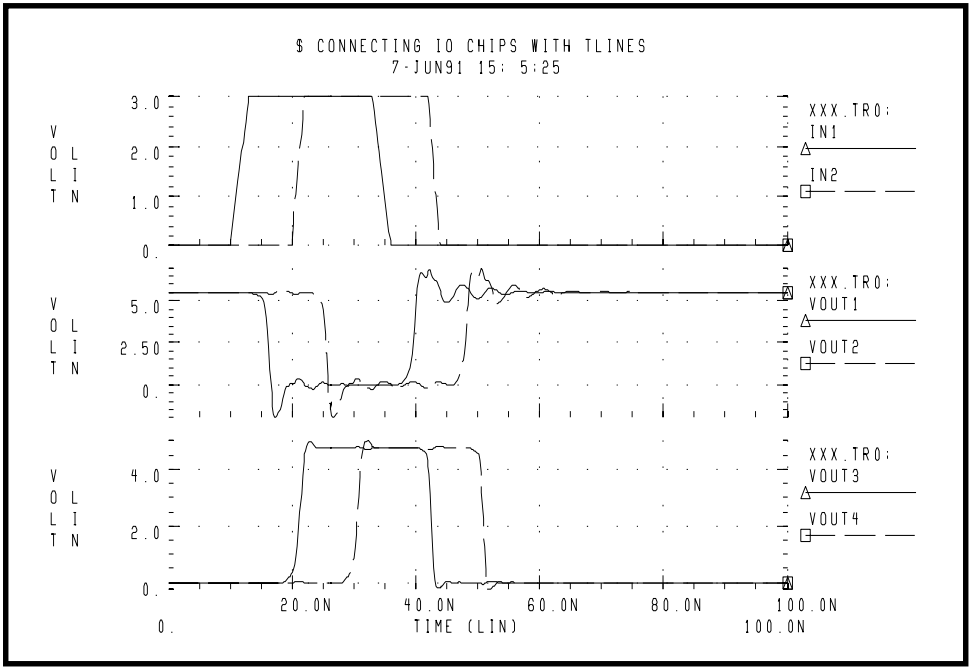

Figure 20-12: Connecting I/O Chips with Transmission Lines

Simulating Circuits with Xilinx FPGAs

Avant!, in conjunction with Xilinx, maintains a library of Star-Hspice transistor level subcircuits for the 3000 and 4000 series Field Programmable Gate Arrays (FPGAs). These subcircuits model the input and output buffer.

The following simulations use the Xilinx input/output buffer (*xil_iob.inc*) to simulate the ground bounce effects for the 1.08µm process at room temperature and nominal model conditions. The IOB and IOB4 are parameterized Star-Hspice subcircuits that allow you to specify:

- Local temperature
- Fast/slow/typical speed selections
- Technology 1.2µ/1.08µ

These choices allow the system designer to perform a variety of simulations to measure:

- Ground bounce as a function of package, temperature, part speed and technology
- Coupled noise, both on-chip and chip-to-chip
- Full transmission line effects at the package and printed circuit board levels
- Peak current and instantaneous power consumption for power supply bussing considerations and chip capacitor placement

Syntax for IOB (xil_iob) and IOB4 (xil_iob4)

```
* EXAMPLE OF CALL FOR 1.2U PART:
* X1  I O PAD TS FAST PPUB TTL VDD GND  XIL_IOB
*+      XIL_SIG=0 XIL_DTEMP=0 XIL_SHRINK=0
* EXAMPLE OF CALL FOR 1.08U PART:
* X1  I O PAD TS FAST PPUB TTL VDD GND  XIL_IOB
*+      XIL_SIG=0 XIL_DTEMP=0 XIL_SHRINK=1
```

Nodes	*Description*
I (IOB only)	output of the TTL/CMOS receiver
O (IOB only)	input pad driver stage
I1 (IOB4 only)	input data 1

I2 (IOB4 only)	input data 2
DRIV_IN (IOB4 only)	
PAD	bonding pad connection
TS	three-state control input (5 V disables)
FAST	slew rate control (5 V fast)
PPUB (IOB only)	pad pull-up enable (0 V enables)
PUP (IOB4 only)	pad pull-up enable (5 V enables)
PDOWN (IOB4 only)	pad pull-down enable (5 V enables)
TTL (IOB only)	CMOS/TTL input threshold select (5 V selects TTL)
VDD	5 volt supply
GND	ground

Parameters	*Description*
XIL_SIG	model distribution: (default 0) -3 ==> slow 0 ==> typical +3==> fast
XIL_DTEMP	buffer temperature difference from ambient. The default = 0 degrees if ambient is 25 degrees and the buffer is 10 degrees hotter than XIL_DTEMP=10
XIL_SHRINK	old or new part; (default is new) 0==>old 1==>new

All grounds and supplies are common to the external nodes for ground and VDD. Star-Hspice allows you to redefine grounds for the addition of package models.

Example of Ground Bounce Simulation

The ground bounce simulation presented duplicates Xilinx internal measurements methods; 8 to 32 outputs are simultaneously toggled. Each output is loaded with a 56-pF capacitance. The simulation uses an 84-pin package mode and an output buffer held at chip ground to measure the internal ground bounce.

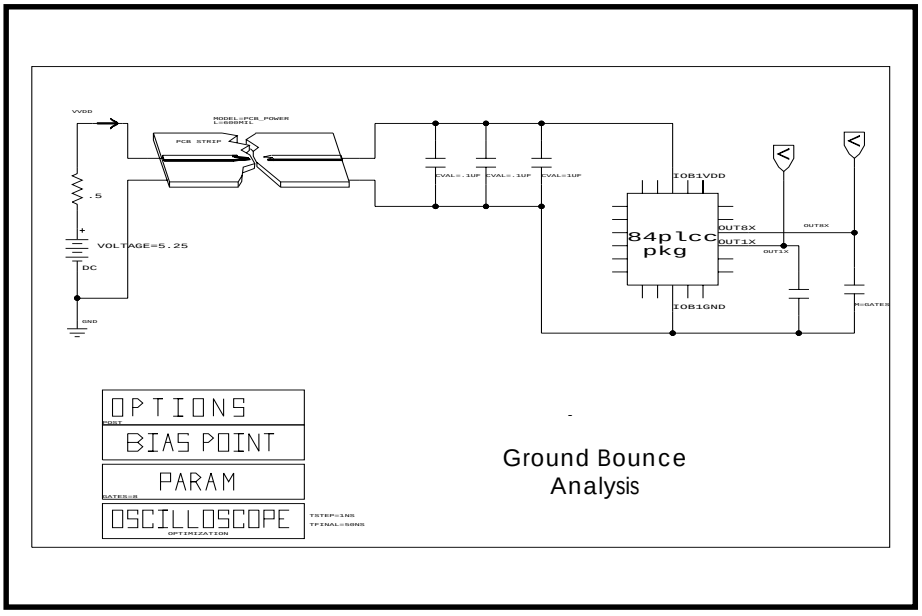

Figure 20-13: Ground Bounce Simulation

The simulation model is adjusted for the oscilloscope recordings for the two-bond wire ground.

Star-Hspice Input File for Ground Bounce

```
qabounce.sp test of xilinx   i/o buffers

* The following is the netlist for the above schematic(fig. 10-13)
.op
.option  post list
.tran 1ns 50ns  sweep gates 8 32 4
.measure bounce  max v(out1x)
*.tran .1ns 7ns
.param gates=8
.print v(out1x) v(out8x)  i(vdd) power
$.param xil_dtemp=-65  $ -40 degrees c  (65 degrees from +25 degrees)
vdd  vdd gnd 5.25
vgnd return gnd 0
upower1 vdd return   iob1vdd iob1gnd   pcb_power  L=600mil
* local power supply capacitors
xc1a iob1vdd iob1gnd cap_mod cval=.1u
```

```
xc1b iob1vdd iob1gnd cap_mod cval=.1u
xc1c iob1vdd iob1gnd cap_mod cval=1u
xgnd_b   iob1vdd iob1gnd out8x out1x  xil_gnd_test
xcout8x out8x iob1gnd cap_mod  m=gates
xcout1x out1x iob1gnd cap_mod  m=1

.model pcb_power u level=3 elev=1 plev=1 nl=1 llev=1
+ th=1.3mil ht=10mil kd=4.5  dlev=1 wd=500mil xw=-2mil

.macro cap_mod node1 node2  cval=56p
Lr1 node1 node1x L=2nh R=0.05
cap node1x node2x c=cval
Lr2 node2x node2 L=2nh R=0.05
.eom
.macro xil_gnd_test  vdd gnd outx outref
+ gates=8
* example of 8 iobuffers simultaneously switching
* through approx.  4nh lead inductance
* 1 iob is active low for ground bounce measurements

vout drive  chipgnd pwl 0ns 5v, 10ns 5v, 10.5ns 0v,
$+  20ns 0v, 20.5ns 5v,  40ns 5v R
x8  I8 drive PAD8x TS FAST PPUB TTL chipvdd chipgnd  xil_iob
+   xil_sig=0 xil_dtemp=0  xil_shrink=1  M=gates
x1  I1 gnd PAD1x TS FAST PPUB TTL chipvdd chipgnd  xil_iob
+   xil_sig=0 xil_dtemp=0  xil_shrink=1  m=1
```

Control Settings

```
rts    ts    chipgnd 1
rfast  fast  chipvdd 1
rppub  ppub  chipgnd 1
rttl   ttl   chipvdd 1
* pad model plcc84 rough estimates
lvdd  vdd chipvdd L=3.0nh r=.02
lgnd  gnd chipgnd L=3.0nh r=.02
lout8x  outx pad8x L='5n/gates'  r='0.05/gates'
lout1x  outref pad1x L=5nh r=0.05
c_vdd_gnd  chipvdd chipgnd 100n
.eom
.end
```

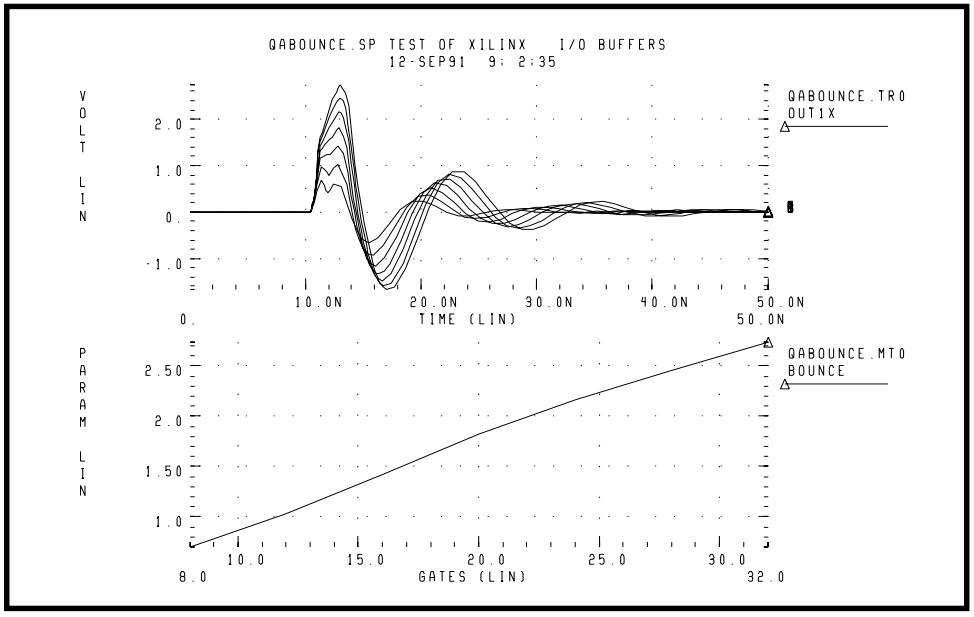

Figure 20-14: Results of Ground Bounce Simulation

Example of Coupled Line Noise

This example uses coupled noise to separate IOB parts. The output of one part drives the input of the other part through 0.6 inch of PCB. The example also monitors an adjacent quiet line.

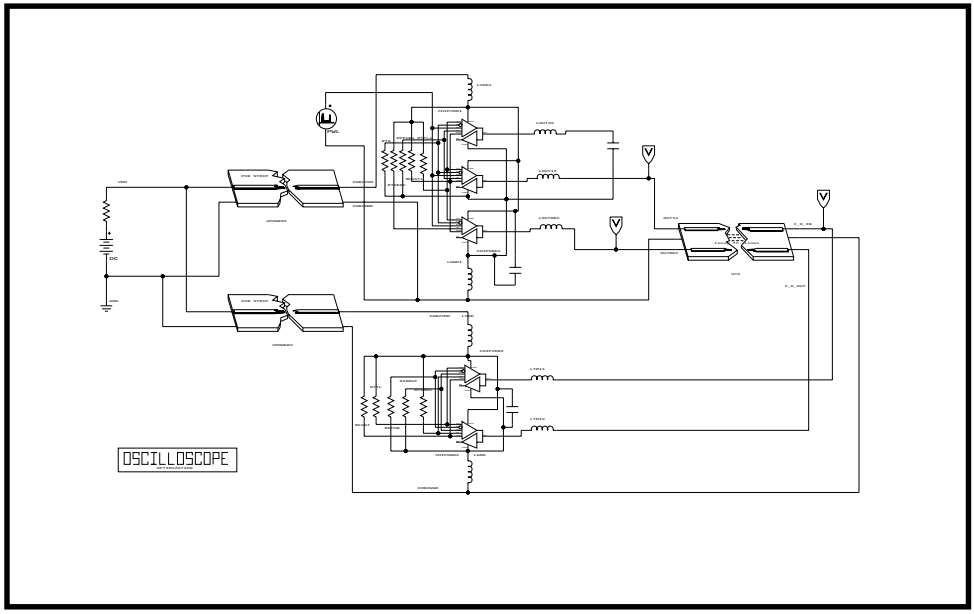

Figure 20-15: Coupled Noise Simulation

Coupled Noise Star-Hspice File

```
qa8.sp test of xilinx 0.8u  i/o buffers

* The following is the netlist for the above schematic ( fig 10-15)
.op
.option  nomod post=2
*.tran .1ns 5ns sweep xil_sig -3 3 3
.tran .1ns 15ns
.print v(out1x) v(out3x)  i(vdd) v(irec)

vdd  vdd gnd 5
vgnd return gnd 0
upower1 vdd return   iob1vdd iob1gnd   pcb_power  L=600mil
upower2 vdd return   iob2vdd iob2gnd   pcb_power  L=600mil

x4io   iob1vdd iob1gnd out3x out1x outrec irec  xil_iob4
cout3x out3x iob1gnd  9pf

u1x  out1x outrec iob1gnd   i_o_in i_o_out  iob2gnd  pcb_top
L=2000mil
```

```
xrec iob2vdd iob2gnd i_o_in i_o_out  xil_rec
.ic i_o_out 0v
.model pcb_top u level=3 elev=1 plev=1 nl=2 llev=1
+ th=1.3mil ht=10mil sp=5mil kd=4.5  dlev=1 wd=8mil xw=-2mil
.model pcb_power u level=3 elev=1 plev=1 nl=1 llev=1
+ th=1.3mil ht=10mil kd=4.5  dlev=1 wd=500mil xw=-2mil

.macro xil_rec  vdd gnd  tri1 tri2
* example of 2 iobuffers in tristate

xtri1  Irec O pad_tri1 TSrec FAST PPUB TTL    chipvdd chipgnd xil_iob
+   xil_sig=0 xil_dtemp=0 xil_shrink=1 m=1
xtri2  Irec O pad_tri2 TSrec FAST PPUB TTL    chipvdd chipgnd xil_iob
+   xil_sig=0 xil_dtemp=0 xil_shrink=1 m=1
```

Control Setting

```
rin_output    O    chipgnd 1
rtsrec tsrec chipvdd 1
rfast  fast  chipvdd 1
rppub  ppub  chipgnd 1
rttl   ttl   chipvdd 1
* pad model plcc84 rough estimates
lvdd  vdd chipvdd L=1nh r=.01
lgnd  gnd chipgnd L=1nh r=.01
ltri1  tri1 pad_tri1 L=3nh r=0.01
ltri2  tri2 pad_tri2 L=3nh r=.01
c_vdd_gnd chipvdd chipgnd 100n
.eom

.macro xil_iob4  vdd gnd out3x out1x outrec Irec
* example of 4 iobuffers simultaneously switching through approx.
*  3nh lead inductance
* 1 iob is a receiver (tristated)

vout O chipgnd pwl 0ns 0v, 1ns 0v, 1.25ns 4v, 7ns 4v, 7.25ns 0v,
12ns 0v R
x3  I3 O PAD3x TS FAST PPUB TTL    chipvdd chipgnd  xil_iob
+   xil_sig=0 xil_dtemp=0 xil_shrink=1 m=3
x1  I1 O PAD1x TS FAST PPUB TTL    chipvdd chipgnd  xil_iob
+   xil_sig=0 xil_dtemp=0 xil_shrink=1 m=1
xrec  Irec O PADrec TSrec FAST PPUB TTL    chipvdd chipgnd xil_iob
+   xil_sig=0 xil_dtemp=0 xil_shrink=1 m=1
* control settings
rts    ts    chipgnd 1
rtsrec tsrec chipvdd 1
```

```
rfast  fast  chipvdd 1
rppub  ppub  chipgnd 1
rttl   ttl   chipvdd 1
* pad model plcc84 rough estimates
lvdd  vdd chipvdd L=1nh r=.01
lgnd  gnd chipgnd L=1nh r=.01
lout3x  out3x pad3x L=1nh r=.0033
lout1x  out1x pad1x L=4nh r=0.01
loutrec  outrec padrec L=4nh r=.01
c_vdd_gnd chipvdd chipgnd 100n
.eom
.end
```

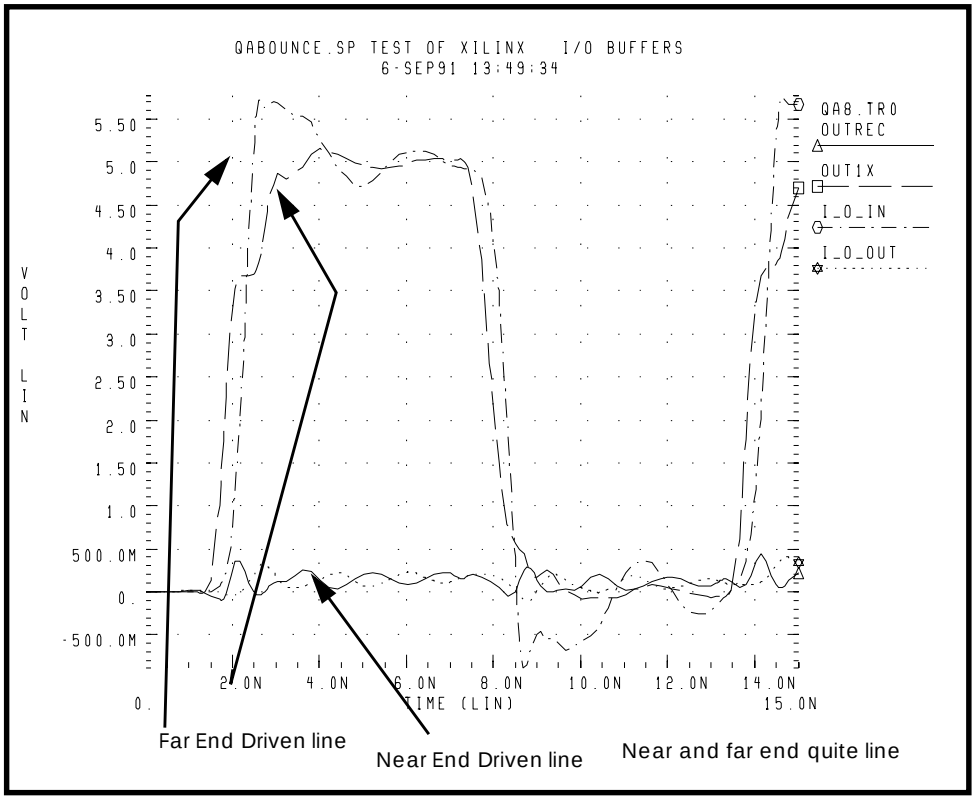

Figure 20-16: Results of Coupled Noise Simulation

Sample of IOB Buffer Module

```
* XILINX IOB INPUT/OUTPUT CIRCUIT
* NAME:   XIL_IOB.INC
* PURPOSE: XILINX INPUT/OUTPUT BLOCK MODEL
* EXAMPLE OF CALL FOR 1.2U PART:
* X1  I O PAD TS FAST PPUB TTL VDD GND  XIL_IOB
*+      XIL_SIG=0 XIL_DTEMP=0 XIL_SHRINK=0
* EXAMPLE OF CALL FOR 1.08U PART:
* X1  I O PAD TS FAST PPUB TTL VDD GND  XIL_IOB
*+      XIL_SIG=0 XIL_DTEMP=0 XIL_SHRINK=1
*
* NAME:    XIL_IOB.INC
* PURPOSE: XILINX INPUT/OUTPUT BLOCK MODEL
* PINS:
*       I     OUTPUT OF THE TTL/CMOS INPUT RECEIVER.
*       O     INPUT TO THE PAD DRIVER STAGE.
*       PAD   BONDING PAD CONNECTION.
*       TS    THREE-STATE CONTROL INPUT. HIGH LEVEL
*             DISABLES PAD DRIVER.
*       FAST  SLEW RATE CONTROL. HIGH LEVEL SELECTS
*             FAST SLEW RATE.
*       PPUB  PAD PULLL-UP ENABLE. ACTIVE LOW.
*       TTL   CMOS/TTL INPUT THRESHOLD SELECT. HIGH
*             SELECTS TTL.
*       VDD   POSITIVE SUPPLY CONNECTION FOR INTERNAL
*             CIRCUITRY.
*   ALL THE ABOVE SIGNALS ARE REFERENCED TO NODE 0.
*   THIS MODEL DOES CAUSE SOME DC CURRENT TO FLOW
*   INTO NODE 0 THAT IS AN ARTIFACT OF THE MODEL.
*       GND   CIRCUIT GROUND
```

Description

```
*  THIS SUBCIRCUIT MODELS THE INTERFACE BETWEEN XILINX
*  3000 SERIES PARTS AND THE BONDING PAD. IT IS NOT
*  USEFUL FOR PREDICTING DELAY TIMES FROM THE OUTSIDE
*  WORLD TO INTERNAL LOGIC IN THE XILINX CHIP. RATHER,
*  IT CAN BE USED TO PREDICT THE SHAPE OF WAVEFORMS
*  GENERATED AT THE BONDING PAD AS WELL AS THE RESPONSE
*  OF THE INPUT RECEIVERS TO APPLIED WAVEFORMS.
*
*  THIS MODEL IS INTENDED FOR USE BY SYSTEM DESIGNERS
```

```
*  WHO ARE CONCERNED ABOUT TRANSMISSION EFFECTS IN
*  CIRCUIT BOARDS CONTAINING XILINX 3000 SERIES PARTS.
*
*  THE PIN CAPACITANCE AND BONDING WIRE INDUCTANCE,
*  RESISTANCE ARE NOT CONTAINED IN THIS MODEL. THESE
*  ARE A FUNCTION OF THE CHOSEN PACKAGE AND MUST BE
*  INCLUDED EXPLICITLY IN A CIRCUIT BUILT WITH THIS
*  SUBCIRCUIT.
*
*  NON-IDEALITIES SUCH AS GROUND BOUNCE ARE ALSO A
*  FUNCTION OF THE SPECIFIC CONFIGURATION OF THE
*  XILINX PART, SUCH AS THE NUMBER OF DRIVERS WHICH
*  SHARE POWER PINS SWITCHING SIMULTANEOUSLY. ANY
*  SIMULATION TO EXAMINE THESE EFFECTS MUST ADDRESS
*  THE CONFIGURATION-SPECIFIC ASPECTS OF THE DESIGN.
*
.SUBCKT XIL_IOB I O PAD_IO TS FAST PPUB TTL VDD GND
+  XIL_SIG=0 XIL_DTEMP=0 XIL_SHRINK=1
.prot FREELIB
;]= $.[;qW.261DW3Eu0
VO\;:n[ $.[;qW.2’4%S+%X;:0[(3’1:67*8-:1:\[
kp39H2J9#Yo%XpVY#O!rDI$UqhmE%:\7%(3e%:\7\5O)1-5i# ;

.ENDS XIL_IOB
```

PCI Modeling Using Star-Hspice

Peripheral Component Interconnect (PCI) is an interconnect specification for standard personal computer architectures. PCI enables low-cost, high-performance standard I/O functions (Figure 20-17:). PCI provides a component-level standard, contrasted to EISA/ISA board-level standards. Both standards coexist with higher performance functions integrated into the system on PCI, while EISA/ISA bring end users the flexibility of adding lower bandwidth functions.

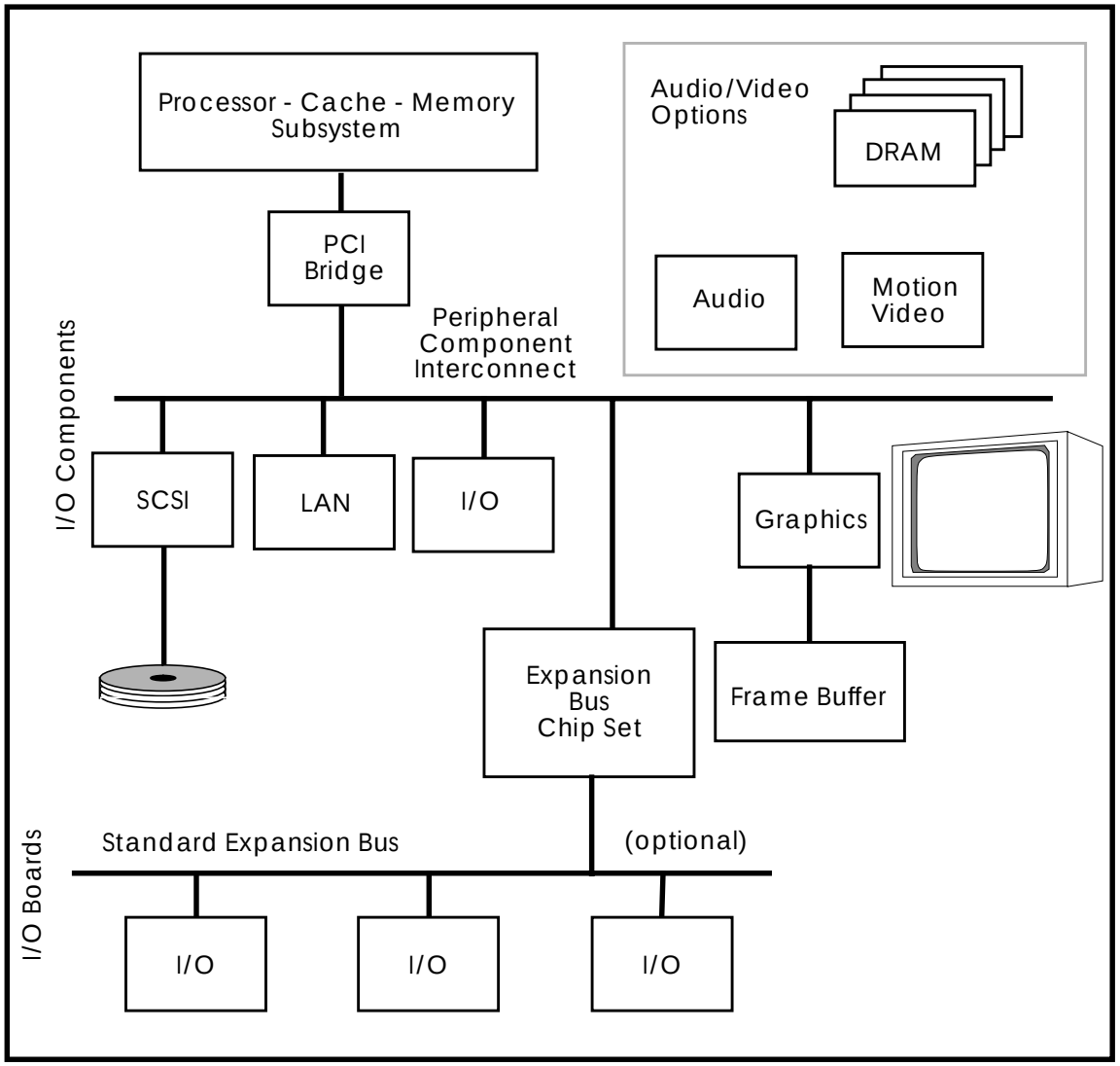

Figure 20-17: PCI System Block Diagram

Importance of Star-Hspice Simulation to PCI Design

PCI's targeted frequency allows it to achieve higher performance. At its target speed, a significant part of the cycle time is spent in the actual propagation of the signals through the system. Use an analog simulation of the interconnect to understand this phenomenon. Star-Hspice is ideally suited to resolving PCI design issues because of the following capabilities:

- Geometric representation of printed circuit traces as lossy transmission lines, to provide excellent correlation between simulation and actual hardware
- Analog behavioral modeling elements, to simplify output buffer models and decrease simulation time
- Monte Carlo analysis, to perform exhaustive random simulations
- An Automatic Measure command, to quickly determine delays allowing thousands of simulations to be analyzed quickly and efficiently

All of these advanced features are used extensively throughout the models and processes described here.

The PCI Speedway Star-Hspice Model

Intel's PCI Speedway is a recommended method for interconnecting PCI devices, as shown in Figure 20-18:. The Speedway is not the only way to interconnect PCI devices, but it has proven to be a robust solution.

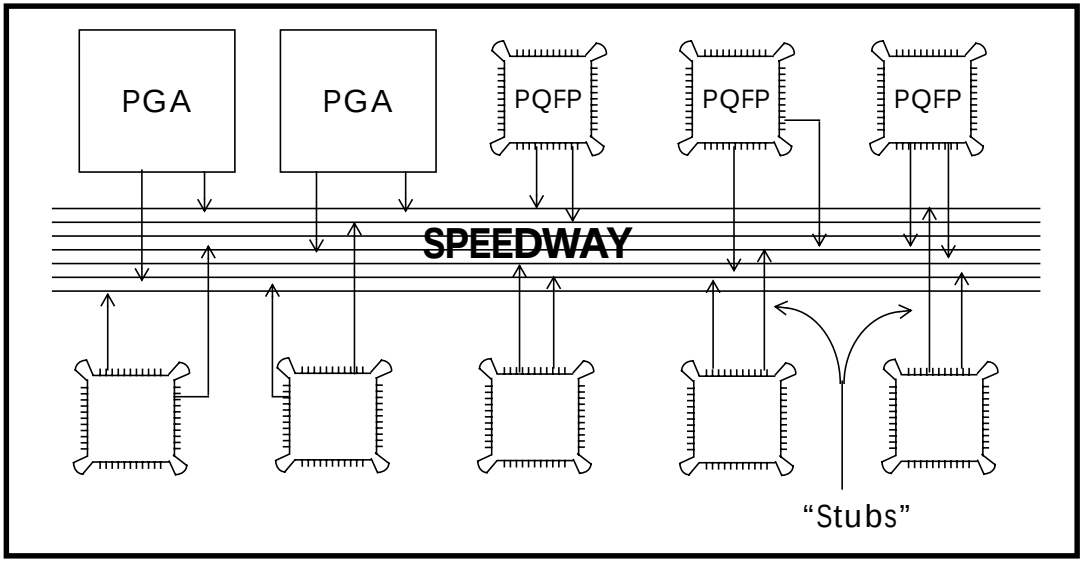

Figure 20-18: PCI Speedway

Figure 20-18: shows ten PCI devices connected together through a narrow band of traces referred to as the "Speedway." Each device's PCI signals connect to the Speedway through a short trace on an orthogonal layer known as a "stub". If the stubs are short enough, they do not cause any unwanted interruptions to a signal traveling on the Speedway except to slow it down.

The PCI Speedway implementation places components on both sides of the Speedway, spaced every two inches, causing a signal traveling on the Speedway to see a new load every inch (on average). This stub-to-stub element is referred to as a "line". Consequently, the Speedway trace is actually a collection of "lines". Both lines and stubs are fully expressed in the Star-Hspice model named and described below.

The Speedway is a behavioral representation of integrated outputs and inputs to speed the simulations and achieve a greater amount of investigation. These elements simulate 25 times faster than silicon models and allow the invention of an optimal PCI buffer characteristic and specification. These buffers are currently provided for use in encrypted form.

Available Files

The PCI files provided under Star-Hspice are listed in Table 20-2:. These are the same files that were derived at Intel Corporation. You can find these files under the *\meta\<release>\parts\pci* directory in the PC version of Star-Hspice, or the *$installdir/parts/pci* directory in other version of Star-Hspice. They are available on all common computers and operating systems.

Table 20-2: PCI-HSPICE Files

File Name	Description
Circuit files	
pci_wc.sp	worst-case PCI Speedway circuit file, ten devices
pci_mont.sp	example Monte Carlo file
pci_lab.sp	file prepared for the lab outlined by this document
Subcircuit “include” files	
pci_in_w.inc	worst-case PCI input load, called by *.sp* file
pci_ii_win.inc	worst-case PCI output driver, called by *.sp* file
pci_ii_t.inc	typical PCI output driver
pci_ii_b.inc	best-case PCI output driver
trace.inc	printed circuit trace subcircuit, called by *.sp* file

The *.inc* files are all called by the *.sp* files during simulation. Consequently, they need to be in the current directory or in a directory referenced by the *hspice.ini* file search statements.

Using the Reference PCI Speedway Model, PCI_WC.SP

The worst-case PCI Speedway reference file, *pci_wc.sp*, can be used directly, or customized to match other PCI implementations. The file serves as a template for experimentation to investigate other configurations. The file also serves as a quick guide to learning the advanced features of Star-Hspice, and seeing them in action.

The reference PCI Speedway file is well documented, to make it simple to use and customize. Hardcopy of this file is listed in "PCI Simulation Example Files" on page 20-49. The file is broken into six major sections, as listed below:

- Parameters
- Star-Hspice Control/Analysis Statements
- Measure Commands
- PCI Driver Selection
- PCI Speedway Subsections
- File Alter Commands

This is the same order that sections appear in the file. The following sections provide a short explanation of each file section, along with actual examples from the file.

Parameters

The PCI model makes extensive use of "parameter" (.PARAM) statements to allow you to describe the environment in a few specific places rather than throughout the file. An example of a parameter is the system voltage. Once the parameter is defined in the Parameter section of the file, you can use it in numerous places throughout the file without further editing. This greatly simplifies the creation of new files, with little or no understanding of the actual circuit section.

Three subsections define the system, line, and stub parameters. The system parameters are listed here:

```
.param vccdc=5.00V     $ system voltage
.param per=60ns        $ period of pulse generator
.param v0=0V vp=5V     $ amplitude of pulse generator
```

```
.param trp=2.5ns        $ rise time of pulse generator
.param tfp='trp'        $ fall time of pulse generator
.param tw='(per/2)-trp' $ pulse width of pulse generator
.param td=2ns           $ delay time of pulse generator
.param cvia=0.5pF       $ via capacitance where stub hits speedway
.param Cin=8.0pF        $ input capacitance of buffer
.param Ci_pkg=2.0pF     $ package capacitance on input of buffer
.param Li_pkg=8nH       $ input bond wire inductance of buffer
.param Ri_pkg=0.03      $ input pin/bond-wire resistance
```

As the comments show, you can set various system parameters such as the DC voltage and the frequency/waveshape of the applied pulse generator. This waveform is applied to the input of the output buffer under test. You can also set the amount of a capacitance applied to a via (the connection of a trace from one layer to the next) using the appropriate parameter.

The last four parameters are passed to the input load model and allow you to see the effects of various packages and loads on Speedway performance. The reference file has ten input loads on the Speedway at the end of the “stub” traces.

The next set of parameters describe the printed circuit board fabrication of the “line” elements. As previously described, the “line” traces make up the Speedway length and connect the “stubs” together. Star-Hspice uses these dimensions to develop a lossy transmission line model of the printed circuit traces. This model accurately characterizes the intrinsic impedance and propagation velocity of the trace, as fabricated.

The line parameters are:

```
.param line=1.0         $ line length, in inches
.param linewd=6         $ line width, in mils
.param lineht=16        $ line height from ground plane, in mils
.param lineth=2.0       $ line thickness, in mils
.param linelyr=0        $ line layer, 1=outer 0=inner
```

The values shown here represent a 6 mil trace on an inner layer, 16 mils from the ground plane. Assuming the components are spaced every two inches on both sides of the speedway, the line length would be one inch as shown. If the components were more spread out, you could set the length using the parameter “line”. The stub parameters are similar to the line parameters, and are listed below:

```
.param stub=1.5           $ length of stub, in inches
```

```
.param stubwd=6           $ stub width, in mils
.param stubht=20          $ stub height from ground plane, in mils
.param stubth=1.8         $ stub thickness, in mils
.param stublyr=1          $ stub layer, 1=outer 0=inner
```

Here, the stubs are set on an outer layer, with a length of 1.5 inches (the recommended maximum).

Star-Hspice Control and Analysis Statements

The following statements cause Star-Hspice to perform a transient analysis, ending the simulation at 60 ns:

- .TRAN 0.1ns 60ns
- .PROBE load1=V(load1)
- .PROBE load2=V(load2)

The .PROBE statements store the transient voltage observed during simulation at the load specified and are similar to placing an oscilloscope probe at that point on a physical board. The waveforms at all ten loads are saved, in addition to the 50 pF reference.

.MEASURE Statement

During simulation, Star-Hspice automatically measures *T_prop* (as defined by the PCI specification), using the .MEASURE statement. The reference file contains .MEASURE commands for rising edge and falling edge measurements. The simulation measures and saves the time delay in a file with a *.mt0* extension. Note that if you run a falling edge simulation, the rising edge measurements are invalid. Similarly, if you run a rising edge simulation, the falling edge measurements are invalid. This is important to remember when referring to the *.mt0* file after a simulation.

Examples of the .MEASURE statements from the file are listed here.

```
*************************************************************
*             Rising edge T_prop measurements               *
*************************************************************
.MEAS tran tr1_val TRIG V(ref_50pf) val=1.5v td='per/2' cross=1
+ TARG V(load1) val=2.0v rise=last
.MEAS tran tr2_val TRIG V(ref_50pf) val=1.5v td='per/2' cross=1
+ TARG V(load2) val=2.0v rise=last
...
.MEAS tran tr10_val TRIG V(ref_50pf) val=1.5v td='per/2' cross=1
+ TARG V(load10) val=2.0v rise=last
*************************************************************
     Falling edge T_prop measurements             *
*************************************************************
.MEAS tran tf1_val TRIG V(ref_50pf) val=1.5v td='per/2' cross=1
+ TARG V(load1) val=0.8v fall=last
...
.MEAS tran tf10_val TRIG V(ref_50pf) val=1.5v td='per/2' cross=1
+ TARG V(load10) val=0.8v fall=last
```

The file provides .MEASURE statements to measure *T_prop* from the ref_50pf waveform to each of ten loads. Since each load is measured, you can quickly determine the worst-case *T_prop* for a given configuration by finding the largest value.

The .MEASURE statements work by triggering on the ref_50pf signal as it crosses 1.5 volts and ending the measurement when the target waveform crosses the specified voltage for the last time. For rising edge measurements, this value is 2.0 volts. For falling edge measurements, the value is 0.8 volts.

PCI Driver Selection

Use the Speedway file to quickly test numerous buffers by acquiring or creating a subcircuit file of the desired buffer and inserting the name into the file. Then you can drive the Speedway from any load position.

The following file shows that the driver chosen is a worst-case Class II buffer. Note that the file uses two buffers; one drives the Speedway from load1, and the other drives a simple 50 pf reference load.

```
*********************************************************************
*                   PCI Driver Selection                            *
*                                                                   *
*   To drive the Speedway from another load position, change the    *
*   next line.  For example the statement:                          *
*                                                                   *
*        Vdrvout load2 drvout $ driver position and current         *
*                                                                   *
*   would drive the Speedway from position number two.  The driver  *
*   model should be formed into a subcircuit, called from the lines *
*                                                                   *
*    Xdriver  sqwave drvout   VCC GND  xxxx  $ place driver here    *
*    Xref_drv sqwave ref_50pf VCC GND  xxxx  $ place driver here too*
*                                                                   *
*   where "xxxx" represents the driver subcircuit name.  The nodes  *
*   must be placed in the order:  input  output vcc gnd.            *
*                                                                   *
*                                                                   *
*  change                                                           *
*  position                                                         *
*  here                                                             *
*********************************************************************
Vdrvout  load1  drvout                     $ driver pos'n and current
Xdriver  sqwave drvout   VCC GND  PCI_II_W $ place driver here
Xref_drv sqwave ref_50pf VCC GND  PCI_II_W $ place driver here too
*          in     out    vcc gnd  name     $ driver format
Xref_clamp ref_50pf VCC 0 PCI_IN_W         $ need input structure
Cref_cap ref_50pf 0 '50e-12-(Cin+Ci_pkg)'  $ total cap. = 50pf
Vpulse   sqwave   0 PULSE v0 vp td trp tfp tw per
Vsupply  VCC      0 DC VCCDC
```

PCI Speedway Subsections

To understand the reference driver and the Speedway subsections, refer to Figure 20-19:.

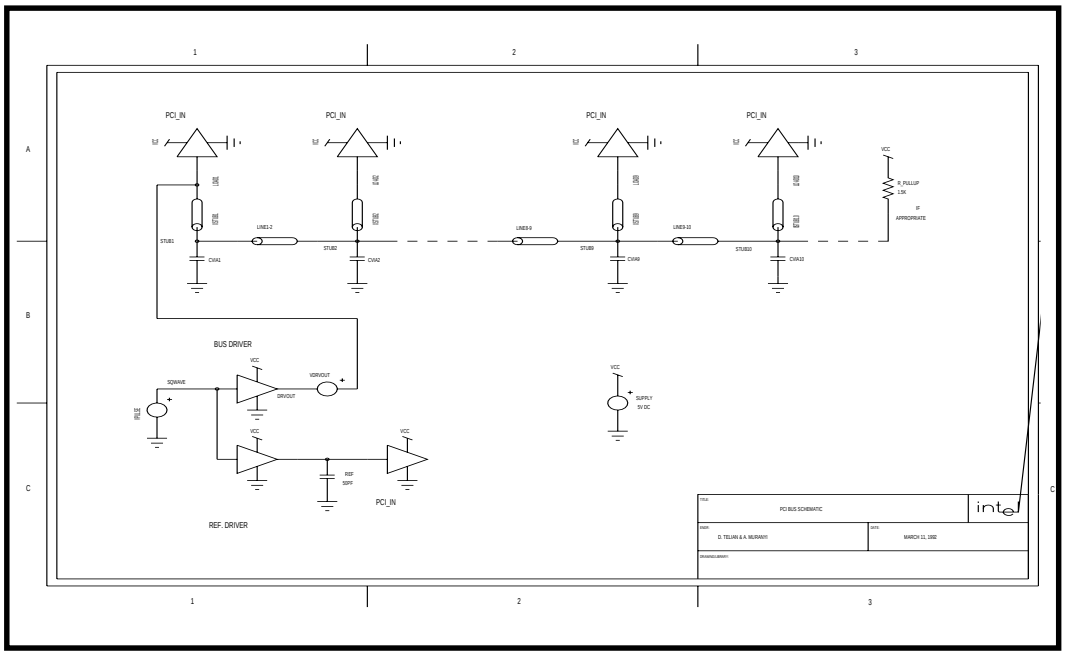

Figure 20-19: PCI Speedway Circuit Schematic

This figure shows the overall topology of the Speedway, and how the individual elements are interconnected.

Aside from the driver section, the Speedway is made up of repetitive subsections, represented in the file as listed below (for the Load 2 and 3 subsections):

```
**********************************************************************
*               Speedway Sub-section Load 2                          *
**********************************************************************
Xline1_2    stub1    stub2    TRACE     LENGTH=line
+      W=LINEWD H=LINEHT T=LINETH DLEVOUT=LINELYR CORRECT=1
Xstub2      stub2    load2    TRACE     LENGTH=STUB
+      W=STUBWD H=STUBHT T=STUBTH DLEVOUT=STUBLYR CORRECT=1
Xload2      load2    VCC      0         PCI_IN_W
Cvia2       stub2    0        CVIA
**********************************************************************
*               Speedway Sub-section Load 3                          *
**********************************************************************
Xline2_3    stub2    stub3    TRACE     LENGTH=LINE
+      W=LINEWD H=LINEHT T=LINETH DLEVOUT=LINELYR CORRECT=1
Xstub3      stub3    load3    TRACE     LENGTH=STUB
+      W=STUBWD H=STUBHT T=STUBTH DLEVOUT=STUBLYR CORRECT=1
Xload3      load3    VCC      0         PCI_IN_W
Cvia3       stub3    0        CVIA
```

As shown in the figure and the listing, each subsection consists of a “line” from the previous subsection (for example, Xline2_3 joins subsections two and three together), a “stub” to the load, a load model at the end of the stub, and a trace via where the stub meets the line. Each element is defined by parameters at the top of the file, but you could adjust it by replacing the parameter name with a value instead. This flexibility is particularly useful since the stubs and lines do not normally have the same length.

You can simulate PCI implementations with fewer than ten loads by either deleting subsections or commenting them out with an asterisk (*) at the beginning of the line

File .ALTER Statement

When the .ALTER statement is invoked, Star-Hspice automatically alters a file and resimulates. The reference file uses this feature at the end to force a rising edge simulation. For the worst-case rising edge simulation, reset the system voltage to its minimum tolerance. In addition, invert the applied input pulse as shown in the following code excerpt.

```
*********************************************************************
*                Alter for Class_II (rising edge)                   *
*********************************************************************
.alter
.param vccdc=4.75V        $ set system voltage here
.param v0=5V vp=0V        $ amplitude of pulse generator
```

Additional .ALTER statements can change the driver type, as shown here to test the falling edge of a best-case PCI driver.

```
*********************************************************************
*        Alter for Best-Case Class_II Driver (falling edge)         *
*********************************************************************
.alter
.param vccdc=5.25V                        $ set system voltage here
.param v0=0V vp=5V                        $ amplitude of pulse gen.
Xdriver sqwave drvout VCC GND PCI_II_B    $ place driver here
Xref_drv sqwave ref_50pf VCC GND PCI_II_B  $ place driver here too
*********************************************************************
```

Star-Hspice/Excel Graph Conversion Macro

To develop a PCI bus, you must run simulations of the board architecture to ensure that the *T_prop* timing matches the PCI specification. However, it is a long and tedious process to manually extract the important *T_prop* timings from the Star-Hspice data file (**.mt0*) produced from running these simulations, especially if there are many Monte Carlo iterations. An Excel macro has been developed to automate this process and extract the *T_prop* timings from the Star-Hspice data file. The macro creates a bar graph in Excel from the *T_prop* timings according to the specifications given.

There are two programs; the first is a demo called *PCIDEMO.XLA*. This demo shows tables of impedance calculations made on the Speedway, describes how the Star-Hspice/Excel graph conversion program works, and shows the *T_prop* graphs made on the Speedway using the conversion program. The second program is the Star-Hspice/Excel graph conversion program called *D_XLCONV.XLA*.

If you have the disk with the two programs, use the following steps to access the data on the disk:

1. Verify that Microsoft Excel™ is installed on your computer.
2. Start Microsoft® Windows™ and run "D_XLCONV.XLA" using either File-Run or the file manager.

   You can open Excel first and then open the conversion program inside Excel, but it is not necessary. The ".XLA" extensions are associated with Excel and automatically open Excel when they are run.
3. Enter the file name of your Star-Hspice data file.

   It is not necessary to add the ".MT0" extension to the file name since the program assumes the file is in the current directory and searches for it there. If your file is not in the current directory, Excel lets you know it cannot find it. If Excel cannot find the file, enter the path with the file name (ex: C:\DATA\EXAMPLE.MT0). You only need to enter the path one time because the program stores the path in memory. To erase this path from memory and switch back to using the current directory, type a period in front of the file name when you enter it.
4. Enter the combination of rising and falling edges your data file has in the main section and its alters.

   For example, if the first run in the *.SP file is a rising edge and the three alters are all falling edges, you would type in "RFFF."
5. Enter what position you are driving from in your simulation runs. Simply enter the number of the position (ex: "1").

   At this point, the program opens your Star-Hspice data file and prompts you for the size of divisions you want to make for your X-axis on the graph(s) to be created.

6. Choose the X-axis division size.

   Your choice affects the level of detail you want for the actual data that make up your bar graph. To use the default value (0.1 ns), press the ‘enter’ key. To change the tick-mark divisions on the X or Y axis, select the scale you want to change, choose “Format - Scale” and change the major and/or minor unit accordingly.

7. If your data file has one or more alters, the program prompts you to make a separate graph for each alter, or to combine all your alters into one graph. If you press ‘return’, Excel assumes you want a separate graph for each alter. If you want to combine all your alters into one graph, type in “2” and press ‘return.’

8. After your graph has been created, enter the title of the graph.

   Type in the title you want and press ‘return.’ The charts are in the order of how they were run in your HSPICE *.SP file. The first run is the chart with the lowest number and the last alter is the chart with the highest number.

9. To rerun the Star-Hspice/Excel graph conversion program without exiting Excel, press “CTRL - D.”

Note: This Star-Hspice/Excel graph conversion program automatically sets up your graph “Page Setup” for printing. The total number of runs is displayed in the top right hand corner of each graph when printed.

PCI Simulation Process

The following outline and examples of simulations help you to understand PCI Star-Hspice simulation. These acquaint you with the simulation process and demonstrate how to adapt the file to simulate other topologies or variations.

The simulation process is outlined as follows:

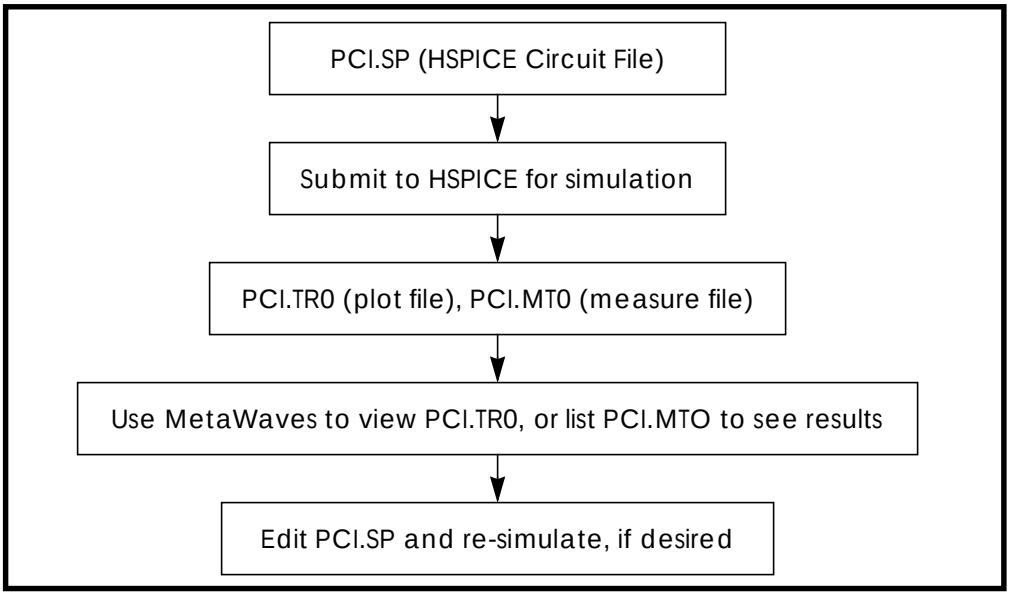

Figure 20-20: PCI Simulation Process

The following list provides ideas for simulation options. Use any combination of the changes listed here, or try your own. Values to change are underlined.

To drive the Speedway from another location (other than position #1)

change:

```
Vdrvout  load1  drvout          $ driver pos'n and current
```

to:

```
Vdrvout  load4  drvout          $ driver pos'n and current
```

To convert the topology from a “Speedway” to a “Subway” (traces run under the components)

change:

```
.param stub=1.5                 $ length of stub, in inches
.param line=1.0                 $ line length, in inches
```

to:

```
.param stub=0.25                $ length of stub, in inches
```

```
.param line=2.0                    $ line length, in inches
```

To change the primary printed circuit fabrication parameters on the Speedway traces

change:

```
.param linewd=6                    $ line width, in mils
.param lineht=16                   $ line height from ground
plane, in mils
```

to:

```
.param linewd=10                   $ line width, in mils
.param lineht=8                    $ line height from ground
plane, in mils
```

To make one of the stubs abnormally long (example would make the stub to device #7 8”)

change:

```
Xstub7      stub7    load7    TRACE     LENGTH=STUB
```

to:

```
Xstub7      stub7    load7    TRACE     LENGTH=8
```

PCI Simulation Example Files

```
                  Hardcopy of PCI_WC.SP Simulation File
PCI Speedway, 10-load Reference Model, Worst-Case  (file=PCI_WC.SP)
**********************************************************************
**********************************************************************
**  COPYRIGHT 1992 Intel Corporation                                  **
**                          Version:  1.7                             **
**********************************************************************
**********************************************************************
*                                                                    *
*  This is a base model of the PCI Speedway environment developed    *
*  under HSPICE.  Most pertinent environment attributes have been    *
*  reduced to HSPICE "parameters."  For example, system voltage can  *
*  be set simply by typing the desired voltage on the line:          *
*                                                                    *
*       .param vccdc=??.?V       $ set system voltage here           *
*                                                                    *
*  The file is structured with the following sections (in order):    *
*                                                                    *
*                  1.  Parameters                                     *
*                  2.  HSPICE Control/Analysis Statements             *
*                  3.  Measure Commands                               *
*                  4.  PCI Driver Selection                           *
*                  5.  PCI Speedway Subsections                       *
*                  6.  File Alter Commands                            *
*                                                                    *
**********************************************************************
*              Interconnect Topology Explanation                      *
*                                                                    *
*  The PCI Speedway interconnects 10 integrated circuit components   *
*  through a network of "stubs" and "lines" as shown:                *
*                                                                    *
*             1    2    3    4    5    6    7    8    9    10      *
*             |____|____|____|____|____|____|____|____|____|        *
*                                                                    *
*  where,                                                            *
*                   stub  =  |                                       *
*                   line  =  ____                                    *
*                                                                    *
*  Each IC load "stubs" onto the Speedway, which is really just a    *
*  collection of "lines". "Line" length represents the physical      *
*  part-to-part spacing. "Stub" length is the distance from the      *
*  component lead to appropriate trace on the speedway.  On a        *
*  printed circuit board, "lines" will typically be routed on        *
*  horizontal layers, and "stub" on vertical layers.  As such, the   *
*  geometric parameters for both "stubs" and "lines" (width,         *
*  distance to the ground plane, ...) are adjustable below.          *
**********************************************************************
```

```
*                File Control Parameters                          *
*****************************************************************
.param vccdc=4.75V        $ set system voltage here
.param per=60ns           $ period of pulse generator
.param v0=5V vp=0V        $ amplitude of pulse generator
.param trp=2.5ns          $ rise time of pulse generator
.param tfp='trp'          $ fall time of pulse generator
.param tw='(per/2)-trp'   $ pulse width of pulse generator
.param tdly=2ns           $ delay time of pulse generator
.param cvia=0.5pF         $ via capacitance where stub hits speedway
.param Cin=8.0pF          $ input capacitance of buffer
.param Ci_pkg=2.0pF       $ package capacitance on input of buffer
.param Li_pkg=8nH         $ input bond wire inductance of buffer
.param Ri_pkg=0.03        $ input pin/bond-wire resistance
*****************************************************************
* Cin + Ci_pkg should equal 10 pF max (PCI Spec C_i/o).
*****************************************************************
*                Line Trace Parameters                            *
*****************************************************************
.param line=1.0           $ line length, in inches
.param linewd=6           $ line width, in mils
.param lineht=16          $ line height from ground plane, in mils
.param lineth=2.0         $ line thickness, in mils
.param linelyr=0          $ line layer, 1=outer 0=inner
*****************************************************************
*                Stub Trace Parameters                            *
*****************************************************************
.param stub=1.5           $ length of stub, in inches
.param stubwd=6           $ stub width, in mils
.param stubht=20          $ stub height from ground plane, in mils
.param stubth=1.8         $ stub thickness, in mils
.param stublyr=1          $ stub layer, 1=outer 0=inner
*****************************************************************
*                Output Control Statements                        *
*****************************************************************
.TRAN 0.1ns 60ns
.OPTIONS ACCT RELTOL=.001 POST=1 PROBE
.PROBE ref_50pf=V(ref_50pf)
.PROBE load1=V(load1)
.PROBE load2=V(load2)
.PROBE load3=V(load3)
.PROBE load4=V(load4)
.PROBE load5=V(load5)
.PROBE load6=V(load6)
.PROBE load7=V(load7)
.PROBE load8=V(load8)
.PROBE load9=V(load9)
.PROBE load10=V(load10)
*****************************************************************
* The following lines can be used to measure the output current of  *
```

```
* the driver and the impedance seen by the driver.  Note that the    *
* impedance of rising and falling edges are calculated differently.  *
* (Remove comment "*" if you want to use this feature.)              *
*.PROBE drvcur=I(vdrvout)                                           *
*.PROBE tdr_rise=par('abs(V(drvout)/I(vdrvout))')                    *
*.PROBE tdr_fall=par('abs((V(vcc)-V(drvout))/I(vdrvout))')           *
**********************************************************************
*                Rising edge T_prop measurements                    *
**********************************************************************
.MEAS tran tr1_val TRIG V(ref_50pf) val=1.5v td='per/2' cross=1
+ TARG V(load1) val=2.0v rise=last
.MEAS tran tr2_val TRIG V(ref_50pf) val=1.5v td='per/2' cross=1
+ TARG V(load2) val=2.0v rise=last
.MEAS tran tr3_val TRIG V(ref_50pf) val=1.5v td='per/2' cross=1
+ TARG V(load3) val=2.0v rise=last
.MEAS tran tr4_val TRIG V(ref_50pf) val=1.5v td='per/2' cross=1
+ TARG V(load4) val=2.0v rise=last
.MEAS tran tr5_val TRIG V(ref_50pf) val=1.5v td='per/2' cross=1
+ TARG V(load5) val=2.0v rise=last
.MEAS tran tr6_val TRIG V(ref_50pf) val=1.5v td='per/2' cross=1
+ TARG V(load6) val=2.0v rise=last
.MEAS tran tr7_val TRIG V(ref_50pf) val=1.5v td='per/2' cross=1
+ TARG V(load7) val=2.0v rise=last
.MEAS tran tr8_val TRIG V(ref_50pf) val=1.5v td='per/2' cross=1
+ TARG V(load8) val=2.0v rise=last
.MEAS tran tr9_val TRIG V(ref_50pf) val=1.5v td='per/2' cross=1
+ TARG V(load9) val=2.0v rise=last
.MEAS tran tr10_val TRIG V(ref_50pf) val=1.5v td='per/2' cross=1
+ TARG V(load10) val=2.0v rise=last
**********************************************************************
*                Falling edge T_prop measurements                   *
**********************************************************************
.MEAS tran tf1_val TRIG V(ref_50pf) val=1.5v td='per/2' cross=1
+ TARG V(load1) val=0.8v fall=last
.MEAS tran tf2_val TRIG V(ref_50pf) val=1.5v td='per/2' cross=1
+ TARG V(load2) val=0.8v fall=last
.MEAS tran tf3_val TRIG V(ref_50pf) val=1.5v td='per/2' cross=1
+ TARG V(load3) val=0.8v fall=last
.MEAS tran tf4_val TRIG V(ref_50pf) val=1.5v td='per/2' cross=1
+ TARG V(load4) val=0.8v fall=last
.MEAS tran tf5_val TRIG V(ref_50pf) val=1.5v td='per/2' cross=1
+ TARG V(load5) val=0.8v fall=last
.MEAS tran tf6_val TRIG V(ref_50pf) val=1.5v td='per/2' cross=1
+ TARG V(load6) val=0.8v fall=last
.MEAS tran tf7_val TRIG V(ref_50pf) val=1.5v td='per/2' cross=1
+ TARG V(load7) val=0.8v fall=last
.MEAS tran tf8_val TRIG V(ref_50pf) val=1.5v td='per/2' cross=1
+ TARG V(load8) val=0.8v fall=last
.MEAS tran tf9_val TRIG V(ref_50pf) val=1.5v td='per/2' cross=1
+ TARG V(load9) val=0.8v fall=last
```

```
.MEAS tran tf10_val TRIG V(ref_50pf) val=1.5v td='per/2' cross=1
+ TARG V(load10) val=0.8v fall=last
*********************************************************************
*                    PCI Driver Selection                           *
*                                                                   *
*   To drive the Speedway from another load position, change the    *
*   next line.  For example the statement:                          *
*                                                                   *
*        Vdrvout load2 drvout $ driver position and current         *
*                                                                   *
*   would drive the Speedway from position number two.  The driver  *
*   model should be formed into a subcircuit, called from the lines *
*                                                                   *
*    Xdriver  sqwave drvout   VCC GND  xxxx  $ place driver here    *
*    Xref_drv sqwave ref_50pf VCC GND  xxxx  $ place driver here too*
*                                                                   *
*   where "xxxx" represents the driver subcircuit name.  The nodes  *
*   must be placed in the order:  input  ouput vcc gnd.             *
*                                                                   *
*                                                                   *
*  change                                                           *
*  position                                                         *
*  here                                                             *
*********************************************************************
Vdrvout  load1  drvout                    $ driver pos'n and current
Xdriver  sqwave drvout   VCC GND  PCI_II_W $ place driver here
Xref_drv sqwave ref_50pf VCC GND  PCI_II_W $ place driver here too
*          in     out    vcc gnd  name     $ driver format
Xref_clamp ref_50pf VCC 0 PCI_IN_W         $ need input structure
Cref_cap ref_50pf 0 '50e-12-(Cin+Ci_pkg)'  $ total cap. = 50pf
Vpulse   sqwave   0 PULSE v0 vp tdly trp tfp tw per
Vsupply  VCC      0 DC VCCDC
*********************************************************************
* Rpullup     load10   VCC      1.5K       $ use if appropriate
*********************************************************************
*               Speedway Sub-section Load 1                         *
*********************************************************************
Xstub1      stub1    load1    TRACE     LENGTH=STUB
+      W=STUBWD H=STUBHT T=STUBTH DLEVOUT=STUBLYR CORRECT=1
Xload1      load1    VCC      0         PCI_IN_W
Cvia1       stub1    0        CVIA
*********************************************************************
*               Speedway Sub-section Load 2                         *
*********************************************************************
Xline1_2    stub1    stub2    TRACE     LENGTH=line
+      W=LINEWD H=LINEHT T=LINETH DLEVOUT=LINELYR CORRECT=1
Xstub2      stub2    load2    TRACE     LENGTH=STUB
+      W=STUBWD H=STUBHT T=STUBTH DLEVOUT=STUBLYR CORRECT=1
Xload2      load2    VCC      0         PCI_IN_W
Cvia2       stub2    0        CVIA
```

```
**********************************************************************
*               Speedway Sub-section Load 3                          *
**********************************************************************
Xline2_3    stub2    stub3    TRACE     LENGTH=LINE
+      W=LINEWD H=LINEHT T=LINETH DLEVOUT=LINELYR CORRECT=1
Xstub3      stub3    load3    TRACE     LENGTH=STUB
+      W=STUBWD H=STUBHT T=STUBTH DLEVOUT=STUBLYR CORRECT=1
Xload3      load3    VCC      0         PCI_IN_W
Cvia3       stub3    0        CVIA
**********************************************************************
*               Speedway Sub-section Load 4                          *
**********************************************************************
Xline3_4    stub3    stub4    TRACE     LENGTH=LINE
+      W=LINEWD H=LINEHT T=LINETH DLEVOUT=LINELYR CORRECT=1
Xstub4      stub4    load4    TRACE     LENGTH=STUB
+      W=STUBWD H=STUBHT T=STUBTH DLEVOUT=STUBLYR CORRECT=1
Xload4      load4    VCC      0         PCI_IN_W
Cvia4       stub4    0        CVIA
**********************************************************************
*               Speedway Sub-section Load 5                          *
**********************************************************************
Xline4_5    stub4    stub5    TRACE     LENGTH=LINE
+      W=LINEWD H=LINEHT T=LINETH DLEVOUT=LINELYR CORRECT=1
Xstub5      stub5    load5    TRACE     LENGTH=STUB
+      W=STUBWD H=STUBHT T=STUBTH DLEVOUT=STUBLYR CORRECT=1
Xload5      load5    VCC      0         PCI_IN_W
Cvia5       stub5    0        CVIA
**********************************************************************
*               Speedway Sub-section Load 6                          *
**********************************************************************
Xline5_6    stub5    stub6    TRACE     LENGTH=LINE
+      W=LINEWD H=LINEHT T=LINETH DLEVOUT=LINELYR CORRECT=1
Xstub6      stub6    load6    TRACE     LENGTH=STUB
+      W=STUBWD H=STUBHT T=STUBTH DLEVOUT=STUBLYR CORRECT=1
Xload6      load6    VCC      0         PCI_IN_W
Cvia6       stub6    0        CVIA
**********************************************************************
*               Speedway Sub-section Load 7                          *
**********************************************************************
Xline6_7    stub6    stub7    TRACE     LENGTH=LINE
+      W=LINEWD H=LINEHT T=LINETH DLEVOUT=LINELYR CORRECT=1
Xstub7      stub7    load7    TRACE     LENGTH=STUB
+      W=STUBWD H=STUBHT T=STUBTH DLEVOUT=STUBLYR CORRECT=1
Xload7      load7    VCC      0         PCI_IN_W
Cvia7       stub7    0        CVIA
**********************************************************************
*               Speedway Sub-section Load 8                          *
**********************************************************************
Xline7_8    stub7    stub8    TRACE     LENGTH=LINE
+      W=LINEWD H=LINEHT T=LINETH DLEVOUT=LINELYR CORRECT=1
```

```
Xstub8      stub8    load8    TRACE     LENGTH=STUB
+      W=STUBWD H=STUBHT T=STUBTH DLEVOUT=STUBLYR CORRECT=1
Xload8      load8    VCC      0         PCI_IN_W
Cvia8       stub8    0        CVIA
*********************************************************************
*               Speedway Sub-section Load 9                         *
*********************************************************************
Xline8_9    stub8    stub9    TRACE     LENGTH=LINE
+      W=LINEWD H=LINEHT T=LINETH DLEVOUT=LINELYR CORRECT=1
Xstub9      stub9    load9    TRACE     LENGTH=STUB
+      W=STUBWD H=STUBHT T=STUBTH DLEVOUT=STUBLYR CORRECT=1
Xload9      load9    VCC      0         PCI_IN_W
Cvia9       stub9    0        CVIA
*********************************************************************
*               Speedway Sub-section Load 10                        *
*********************************************************************
Xline9_10   stub9    stub10   TRACE     LENGTH=line
+      W=LINEWD H=LINEHT T=LINETH DLEVOUT=LINELYR CORRECT=1
Xstub10     stub10   load10   TRACE     LENGTH=STUB
+      W=STUBWD H=STUBHT T=STUBTH DLEVOUT=STUBLYR CORRECT=1
Xload10     load10   VCC      0         PCI_IN_W
Cvia10      stub10   0        CVIA
*********************************************************************
*                Alter for Class_II (falling edge)                  *
*********************************************************************
.alter
.param vccdc=5.25V        $ set system voltage here
.param v0=0V vp=5V        $ amplitude of pulse generator
*********************************************************************
*            Alter for Best-Case Class_II (rising edge)             *
*********************************************************************
*.alter
*.param vccdc=4.75V        $ set system voltage here
*.param v0=5V vp=0V        $ amplitude of pulse generator
*Xdriver     sqwave   drvout   VCC     GND
*+  PCI_II_B              $ place driver here
*Xref_drv    sqwave   ref_50pf VCC     GND
*+  PCI_II_B              $ place driver here too
*********************************************************************
*           Alter for Best-Case Class_II (falling edge)             *
*********************************************************************
*.alter
*.param vccdc=5.25V        $ set system voltage here
*.param v0=0V vp=5V        $ amplitude of pulse generator
*Xdriver     sqwave   drvout   VCC     GND
*+  PCI_II_B              $ place driver here
*Xref_drv    sqwave   ref_50pf VCC     GND
*+  PCI_II_B              $ place driver here too
*********************************************************************
.END
```

Analyzing Board Signal Integrity

This section describes the features of a tool station that analyzes printed circuit boards using Star-Hspice and PADS-LPE, a layout extraction module. A number of board layouts from different routers, such as PADS, Allegro, Mentor, PDIF and Racal, can be read in. Specify critical net names to generate Star-Hspice equivalent statements. These are transmission line (U-line) element statements for board traces, and Buffer calls for the input/output and glue buffers. The model name is automatically generated for each U-line, based on the layer number, width of the line, and whether it a stripline or microstrip. You must then create the .MODEL statements. Obtain the needed inputs from the board stack-up and the model name itself. A simple mechanism allows you to choose net names classified under different signal classes. The signal class is a group of signals with common delay specification.

To maintain accuracy, local ground is represented. Handle direct terminations to power supplies carefully. Comment lines about using local ground/power are generated routinely in each partial input file. Use a field solver to get the board power/ground characteristics, then use it to degrade the supplies for direct terminations. Specify the ground/power node for each net before you run the extraction.

Special template input files for delay and overshoots/undershoots are provided. Use these files to create input files for the Star-Hspice runs. Complete the partial Star-Hspice decks and include .MEASURE, .PRINT, .OPTION, and .SOURCE statements as demonstrated in the templates.

Input Files for the Extraction Process

This section describes the input files needed for the Star-Hspice extraction process.

DEMO.DAT

This file contains the name of the routed database file with the *.SRC* extension from the router, and net names for the creation of partial Star-Hspice input netlist files. You must specify the ground node for each net. After the extraction, the ground/power nodes appear as comments. Instead of using values of 5 V or 0 V, the node names are used in the resulting netlist. To get a more accurate representation of the supplies, use at least a resistor to get realistic levels.

```
DA[00]          GND=node0 PWR=node12 (Net names list)
DA[06]          GND=node1 PWR=node14
TRLW.EXE /PADS (call to the PADS Translator)
x%n i%n o1 %node tsi1 fast1 ppub1 ttl1 vdd1 gnd1 %mdl_iob
+ %mdl_sig=0 %mdl_dtemp=0 %mdl_shrink=0
```

(This is a prototype statement used to specify the pin order)

DEMO.SRC

This routed database file from a router such as PADS is generated using the ASCII OUT option in the PADS environment.

DEMO.MDD

This file contains path names to model files(*.MDL) used to instantiate buffer calls.

```
.BEGIN
C:\DEMOBRD\MDB\XIL3.MDL
C:\DEMOBRD\MDB\EB3.MDL
C:\DEMOBRD\MDB\DBX1.MDL
.END
```

DEMO.MDL

These model files contain pin types and part names.

```
.BEGIN
.NAME
part name (or package name)
.TYPE
part type (can be Internal for all ACTIVE devices except
diodes, resistor, capacitor, inductor, connector, diode,
package)
.INPUT PINS
pin_name list
.OUTPUT PINS
pin_name_list
.BI PINS
bidirectional_pins_list
.TRI PINS
tristate_pins_list
.PWR PINS
power_pins
.GND PINS
gnd pins
.NC PINS
unused_pin _list
.DCTERM PINS
pin_name resistance voltage
.ANODE PIN
pin_name
.CATHODE PIN
pin_name
   .VALUE
pin1 pin2 value (for resistors, inductors, capacitors)
```

DEMO.SNF

This special nets file should contain names for global power/ground.

```
.POWERBUS
VCC 5.0
GND 0.0
```

DEMO.BST

The board stack-up file describes the board as a series of structures such as microstrip or stripline.

```
.UNITS
mils (or cm)
.STRUCTURE
microstrip (or stripline)
.DIELECTRIC
air
T=55   ER=4.5
.ENDS
.SIGNAL
L=1 H=15 T=1.4
.ENDS
.ENDS
.THRU_VIA (or .BURIED_VIA, if applicable)
1pf
.STRUCTURE
microstrip
.DIELECTRIC
T=55 ER=4.5
air
.SIGNAL
L=2 H=15 T=1.4
.ENDS
.ENDS
.RESISTIVITY value (This allows the resistivity to be
specified in ohm-cm.Default is copper)
.WID_ADJ value (Allows different built width vs. drawn width)
```

Output Files from the Extraction Process

This section describes the output files you get from the Star-Hspice extraction process.

DEMO.NET

This file is a partial Star-Hspice-compatible netlist. One file is created for each net specified in the *.DAT* file. The netlist describes a Xilinx driver, driving a transmission line into a through via (represented by a capacitor) to the bottom side of the board into a Xilinx receiver. The board is made of four layers. Using the templates provided, complete the input files to run Star-Hspice.

```
x1 i1 o1 n1 tsi1 fast1 ppub1 ttl1 vdd gnd xil_iob
+ xil_sig=0 xil_dtemp=0 xil_shrink=1
x2 i2 n3 pad1 tsi2 fast1 ppub1 ttl1 vdd1 gnd1
+ xil_iob  xil_sig=0 xil_dtemp=0 xil_shrink=1
u1 n1 gnd n2 out_ref_1 umod1_4 l=1200mil
u2 n2 out_ref_1 n3 out_ref_2 umod4_4 l=300mil
c1 n2 out_ref_1 0.01pf
```

DEMO.CRF

Generate this cross-reference file containing names of buffer instances found in the database for which there are no models pointed to by the *.MDL* file. Use this list as a reference to create the missing *.MDL* files.

Running PADS-LPE (The Extraction Program)

The program can be invoked from within Microsoft® Windows™ as “PADS-LPE”.

You must add the path name to the file *autoexec.bat* so it can be invoked without the complete path name.

Delay Analysis Template

Use this Star-Hspice input file as a template for DELAY analysis of a net from a PADS file.

```
*  Database

*supplies
v1 vdd gnd 5.0v

v3 a gnd 0v
v4 b gnd 0v
v5 c gnd 5v
v6 d gnd 0v
v9 e gnd 5v

* Piece-wise linear statement - a 0.5ns rise time is assumed

vdat o1 gnd pwl 0ns 0v 9.5ns 0v 10ns 5v 19.5ns 5v 20ns 0v
+ 30ns 0v
```

Control Settings

```
r1 a tsi1 1
r2 b tsi2 1
r3 c fast1 1
r4 d ttl1 1
r5 e ppub1 1
c6 i2 gnd 0.5pf
v7 vdd1 gnd 4.5v
v8 gnd1 gnd 0.5v
```

Default Netlist Created Automatically Per Net

```
x1 i1 o1 n1 tsi1 fast1 ppub1 ttl1 vdd gnd xil_iob
+ xil_sig=0 xil_dtemp=0 xil_shrink=1
x2 i2 n3 pad1 tsi2 fast1 ppub1 ttl1 vdd1 gnd1
+ xil_iob  xil_sig=0 xil_dtemp=0 xil_shrink=1
u1 n1 gnd n2 out_ref_1 umod1_4 l=1200mil
u2 n2 out_ref_1 n3 out_ref_2 umod4_4 l=300mil
c1 n2 out_ref_1 0.01pf
```

.Measure Statements For Rise And Fall Delay Measurement

```
.measure tran rise1 trig v(o1) val=2.5ns rise=1
+ targ v(n3) val=2.5ns rise=1
.measure tran fall1 trig v(o1) val=2.5ns fall=1
+ targ v(n3) val=2.5ns fall=1

.model umod1_4 u level=3 elev=1 plev=1 llev=1 th=1.4mil
+ ht=10mil kd=5 dlev=1 wd=4mil

.model umod4_4 u level=3 elev=1 plev=1 llev=1 th=1.4mil
+ ht=10mil kd=5 dlev=1 wd=4mil

.option post list
.tran 0.1ns 30ns
.end
```

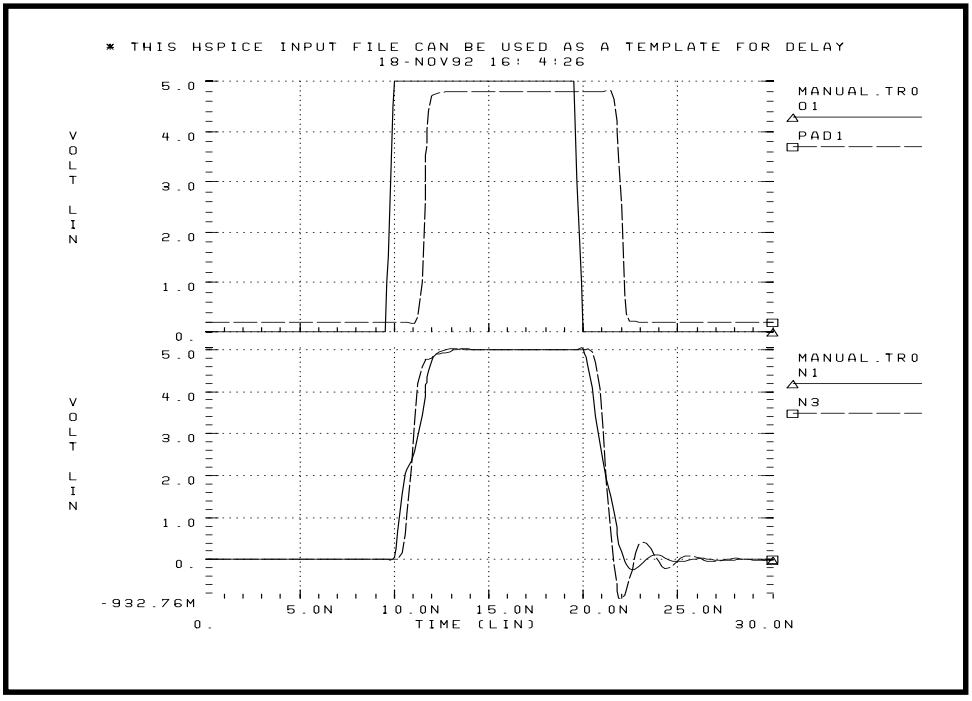

Figure 20-21: Delay Template Output

Overshoot/Undershoot Analysis Template

```
* This HSPICE input file can be used as a template for OVER/
UNDER SHOOTS
* analysis of a net from a PADS Database

*supplies
v1 vdd gnd 5.0v

v3 a gnd 0v
v4 b gnd 0v
v5 c gnd 5v
v6 d gnd 0v
v9 e gnd 5v

* Piece-wise linear statement - a 0.5ns rise time is assumed

vdat o1 gnd pwl 0ns 0v 9.5ns 0v 10ns 5v 19.5ns 5v 20ns 0v
+ 39.5ns 0v 40ns 5v 50ns 5v
```

Control Settings

```
r1 a tsi1 1
r2 b tsi2 1
r3 c fast1 1
r4 d ttl1 1
r5 e ppub1 1
*c6 i2 gnd 0.5pf
v7 vdd1 gnd 4.8v
v8 gnd1 gnd 0.2v
```

Default Netlist Created Automatically Per Net

```
* The netlist describes a Xilinx driver, driving a
transmission line
* into a through via(represented by a capacitor) to the
bottom side
* of the board into a Xilinx receiver. The board is made of 4
layers.

x1 i1 o1 n1 tsi1 fast1 ppub1 ttl1 vdd gnd xil_iob
+ xil_sig=0 xil_dtemp=0 xil_shrink=1
x2 i2 n3 pad1 tsi2 fast1 ppub1 ttl1 vdd1 gnd1
+ xil_iob  xil_sig=0 xil_dtemp=0 xil_shrink=1
```

```
u1 n1 gnd n2 out_ref_1 umod1_4 l=1200mil
u2 n2 out_ref_1 n3 out_ref_2 umod4_4 l=1200mil
c1 n2 out_ref_1 0.01pf

* measure statements for overshoots and undershoots
* Node n3 was chosen since it is the input directly to the
receiver,
* and may cause * the receiver to switch falsely. Overshoot
is more
* of a concern after the first fall,
* Undershoot should be examined after the first rise.

* Over/Under shoots after the first rise of node n3

.measure tran osh1 when v(n3)='0.5*5' cross=1
.measure tran osh2 when v(n3)='0.5*5' cross=2
.measure tran tmid_o param='(osh1+osh2)/2'
.measure tran vmid_o find v(n3) AT='tmid_o'
.measure tran o_from when v(n3)='vmid_o' rise=1
.measure tran overshoot1 MAX v(n3) from='o_from' to='tmid_o'
.measure tran undershoot1 MIN v(n3) from='o_from' to='tmid_o'

* Over/Under shoots after the first fall of node n3

.measure tran ush1 when v(n3)='0.5*5' cross=2
.measure tran ush2 when v(n3)='0.5*5' cross=3
.measure tran tmid_u param='(ush1+ush2)/2'
.measure tran vmid_u find v(n3) AT='tmid_u'
.measure tran u_from when v(n3)='vmid_u' fall=1
.measure tran overshoot2 MAX v(n3) from='u_from' to='tmid_u'
.measure tran undershoot2 MIN v(n3) from='u_from' to='tmid_u'

.model umod1_4 u level=3 elev=1 plev=1 llev=1 th=1.4mil
+ ht=10mil kd=5 dlev=1 wd=40mil

.model umod4_4 u level=3 elev=1 plev=1 llev=1 th=1.4mil
+ ht=10mil kd=5 dlev=1 wd=40mil

.option post
.tran 0.1ns 50ns
.end
```

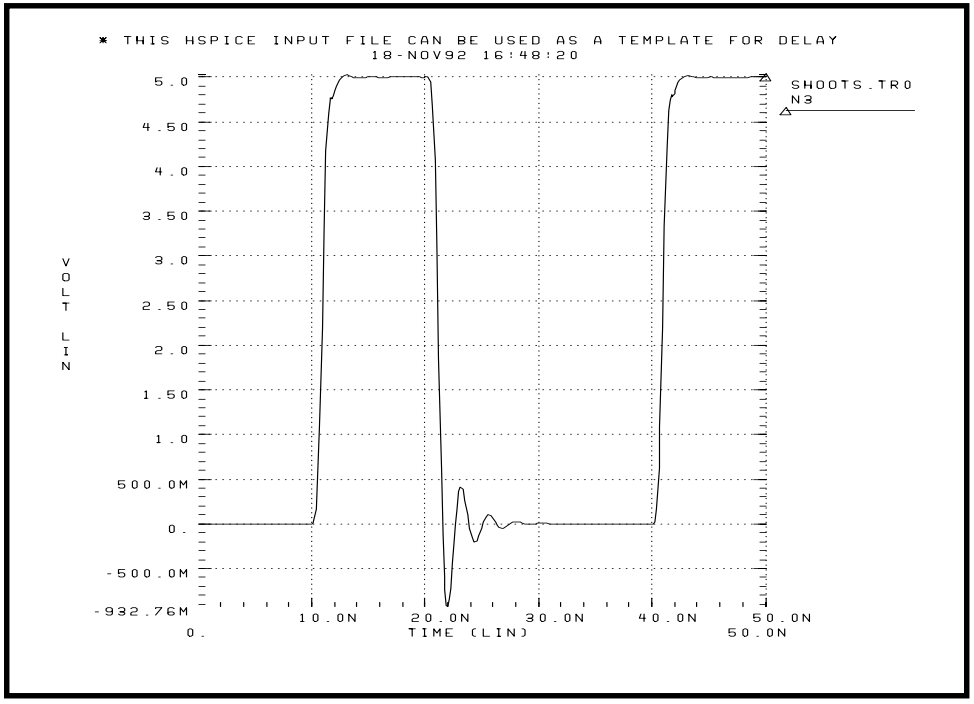

Figure 20-22: Overshoot Template Example