Avant!

Chapter 27

Timing Analysis Using Bisection

To analyze circuit timing violations, a typical methodology is to generate a set of operational parameters that produce a failure in the required behavior of the circuit. Then when a circuit timing failure occurs, you can identify a timing constraint that can lead to a design guideline. You must be able to perform an iterative analysis to define the violation specification.

Typical types of timing constraint violations include:

- Data setup time before clock
- Data hold time after clock
- Minimum pulse width required to allow a signal to propagate to the output
- Maximum toggle frequency of the component(s)

This chapter describes how to use the Star-Hspice bisection function in timing optimization. The general topic of optimization with Star-Hspice is covered in depth in Chapter 11, “Optimizing Performance”.

The following topics are covered in this chapter:

Understanding Bisection

Formerly, engineers built external drivers to submit multiple parameterized Star-Hspice jobs, with each job exploring a region of the operating envelope of the circuit. In addition, the driver needed to provide part of the analysis by post-processing the Star-Hspice results to deduce the limiting conditions.

Because characterization of circuits in this way is associated with small jobs, the individual analysis times are relatively small compared with the overall job time. This methodology is inefficient because of the overhead of submitting the job, reading and checking the netlist, and setting up the matrix. Efficiency in analyzing timing violations can be increased with more intelligent methods of determining the conditions causing timing failure. The bisection optimization method was developed to make cell characterization in Star-Hspice more efficient.

Star-Hspice bisection methodology saves time in three ways:

- Reduction of multiple jobs to a single characterization job
- Removal of post-processing requirements
- Use of accuracy-driven iteration

Figure 27-1: illustrates a typical analysis of setup time constraints. A cell is driven by clock and data input waveforms. There are two input transitions, rise and fall, that occur at times T_1 and T_2. The result is an output transition, when V(out) goes from low to high. The following relationship between times T_1(data) and T_2 (clock) must be true in order for the V(out) transition to occur:

$$T_2 > (T_1 + \text{setup time})$$

The goal of the characterization, or violation analysis, is to determine the setup time. This is done by keeping T_2 fixed while repeating the simulation with different values of T_1 and observing which T_1 values produce the output transition and which values do not.

Previously, it was necessary to do very tight sweeps of the delay between the data setup and clock edge, looking for the value at which the transition fails to occur. This was done by sweeping a value that specifies how far the data edge precedes a fixed clock edge. This methodology is time consuming, and is not

accurate unless the sweep step is very small. The setup time value cannot be determined accurately by linear search methods unless extremely small steps from T_1 to T_2 are used to simulate the circuit at each point while monitoring the outcome.

For example, even if it is known that the desired transition occurs during a particular five nanosecond period, searching for the actual setup time to within 0.1 nanoseconds over that five nanosecond period takes as many as 50 simulations. Even after this, the error in the result can be as large as 0.05 nanoseconds.

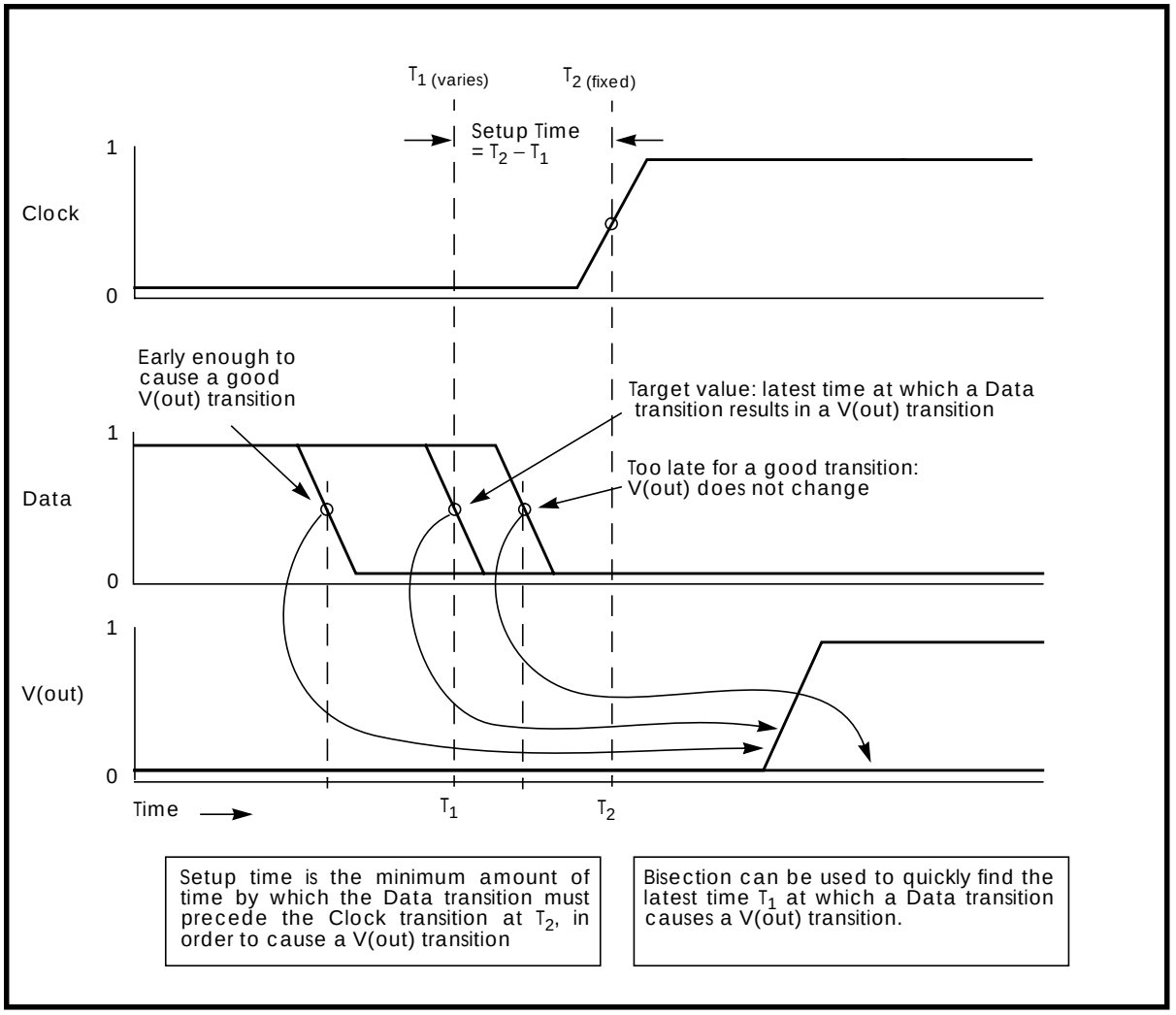

Figure 27-1: Determining Setup Time with Bisection Violation Analysis

The Star-Hspice Bisection feature greatly reduces the amount of work and computational time required to find an accurate solution to this type of problem. The following pages show examples of using this feature to identify setup, hold, and minimum clock pulse width timing violations.

Understanding the Bisection Methodology

Bisection is a method of optimization which employs a binary search method to find the value of an input variable (target value) associated with a “goal” value of an output variable. The input and output variables may be of various types – for example, voltage, current, delay time or gain– related by some transfer function. In general, use a binary search to locate the output variable goal value within a search range of the input variable by iteratively halving that range to converge rapidly on the target value. At each iteration the “measured value” of the output variable is compared with the goal value. Bisection is employed in both the “pass/fail” method and the “bisection” method (see “Using Bisection” on page -6). The process is largely the same for either case.

The Star-Hspice Bisection procedure involves two steps when solving the timing violation problem. First, the procedure detects whether the output transition occurred. Second, the procedure automatically varies the input parameter (T_1 in Figure 27-1:) to find the value for which the transition barely occurs. The Star-Hspice Measurement and Optimization features handle these two steps.

Measurement

Use the Star-Hspice MAX measurement function to detect success or failure of an output transition. In the case of a low-to-high output transition, a MAX measurement produces zero on failure, or approximately the supply voltage V_{dd} on success. This measurement, using a goal of V_{dd} minus a suitable small value to ensure a solution, is sufficient to drive the optimization.

Optimization

The bisection method is straightforward, given a single measurement with a goal and known upper and lower boundary values for the input parameter. The characterization engineer should be able to specify acceptable upper and lower boundary values.

Using Bisection

To use bisection, the following is required:

- A user-specified pair of upper and lower boundary input variable values. For a solution to be found, one of these values must result in an output variable result |goal value| and the other in a result< |goal value|
- Specified goal value
- Error tolerance value. The bisection process stops when the difference between successive test values ≤ error tolerance. If the other criteria are met, see below.
- Related variables. Variables must be related by a monotonic transfer function, where a steadily progressing time (increase or decrease) results in a single occurrence of the "goal" value at the "target" input variable value

The error tolerance is included in a relation used as a process-termination criterion.

Figure 27-2: shows an example of the binary search process used by the bisection algorithm. This example is of the "pass/fail" type, and is appropriate for a setup-time analysis that tests for the presence of an output transition as shown in Figure 27-1. Here, a long setup time T_S (= T_2 - T_1) results in a VOUT transition (a "pass"), and a too-short setup time (where the latch has not stabilized the input data before the clock transition) results in a "fail." A "pass" time value, for example, might be defined as any setup time TS that produces a VOUT output *"minimum high"* logic output level of ≥ 2.7 V – the "goal" value. The "target" value is that setup time that *just* produces the V_{OUT} value of 2.7 V. Since finding the exact value is impractical, if not impossible, an error tolerance is specified to give a solution arbitrarily close to the target value. The bisection algorithm performs tests for each of the specified boundary values to determine the direction in which to pursue the target value after the first bisection. In this example, the upper boundary value is a "pass" value, and the lower boundary value is a "fail" value.

To start the binary search, a lower boundary and upper boundary are specified. The program tests the point midway between the lower and upper boundaries (see Figure 27-2:).

If the initial value passes the test, the target value must be less than the tested value (in this case), so the bisection algorithm moves the upper search limit to the value it just tested. If the test fails, the target value must greater than the tested value, so the bisection algorithm moves the lower limit to the value it just tested.

Then the algorithm tests a value midway between the new limits. The search continues in this manner, moving one limit or the other to the last midpoint, and testing the value midway between the new limits. The process stops when the difference between the latest test values is less than or equal to the user-specified error tolerance (normalized by multiplying by the initial boundary range).

Examining the Command Syntax

```
.MODEL <OptModelName> OPT METHOD=BISECTION ...
```

or

```
.MODEL <OptModelName> OPT METHOD=PASSFAIL ...
```

OptModel- the model to be used. Refer to Chapter 11, “Optimizing Performance” for *Name*information on specification of optimization models in Star-Hspice.

METHOD keyword to indicate which optimization method to use. For bisection, the method may be one of the following:

BISECTION When the difference between the two latest test input values is within the error tolerance and the latest measured value exceeds the goal, bisection has succeeded, and stops. The process reports the optimized parameter that corresponded to the test value that satisfies this error tolerance, and this goal (passes).

PASSFAIL When the difference between the two latest test input values is within the error tolerance and one of the values ≥ goal (passes) and the other fails, bisection has succeeded and stops. The process reports the value the input parameter value associated with the “pass” measurement.

OPT keyword to indicate optimization is to be performed

The parameters are passed in a normal optimization specification:

```
.PARAM <ParamName>=<OptParFun> (<Initial>, <Lower>, <Upper>)
```

In the BISECTION method, the measure results for <Lower> and <Upper> limits of <ParamName> must be on opposite sides of the GOAL value in the .MEASURE statement. For the PASSFAIL method, the measure must pass for one limit and fail for the other limit. The process ignores the value of the <Initial> field.

The error tolerance is a parameter in the model being optimized.

Note that the bisectional search is applied to only one parameter.

When the OPTLST option is set (.OPTION OPTLST=1), the process prints the following information for the BISECTION method:

```
bisec-opt iter = <num_iterations>  xlo = <low_val>  xhi =
<high_val>
x = <result_low_val>  xnew = <result_high_val>
err = <error_tolerance>
```

where x is the old parameter value and xnew is the new parameter value.

When .OPTION OPTLST=1, the process prints the following information for the PASSFAIL method:

```
bisec-opt iter = <num_iterations>  xlo = <low_val>  xhi =
<high_val>
x = <result_low_val>  xnew = <result_high_val>
measfail = 1
```

(measfail = 0 for a test failure for the x value).

Example: transient analysis .TRAN statement:

```
.TRAN <TranStep> <TranTime> SWEEP OPTIMIZE=<OptParFun>
+ RESULTS=<MeasureNames> MODEL=<OptModelName>
```

Example: transient .MEASURE statement:

```
.MEASURE TRAN <MeasureName> <MeasureClause> GOAL=<GoalValue>
```

Example 1 – Setup Time Analysis

This example uses a bisectional search to find the minimum setup time for a D flip-flop. The circuit for this example is */bisect/dff_top.sp* in the Star-Hspice *$installdir/demo/hspice* demonstration file directory. The files in Figures 27-2 and Figure 27-3 show the results of this demo. Note that setup time is not optimized directly, but is extracted from its relationship with the DelayTime parameter (the time preceding the data signal), which is the parameter being optimized.

Input listing

```
File: $installdir/demo/hspice/bisect/dff_top.sp
* DFF_top Bisection Search for Setup Time
*
* PWL Stimulus
*
v28 data  gnd PWL
+ 0s        5v
+ 1n        5v
+ 2n        0v
+ Td = “DelayTime”         $ Offsets Data from time 0 by
DelayTime
v27 clock gnd PWL
+ 0s        0v
+ 3n        0v
+ 4n        5v
*
* Specify DelayTime as the search parameter and provide
* the lower and upper limits.
*
.PARAM DelayTime= Opt1 ( 0.0n, 0.0n, 5.0n )
*
* Transient simulation with Bisection Optimization
*
.TRAN 1n 8n Sweep    Optimize = Opt1
+                    Result   = MaxVout $ Look at measure
+                    Model    = OptMod
*
* This measure finds the transition if it exists
```

```
*
.MEASURE Tran MaxVout Max v(D_Output) Goal = ‘v(Vdd)’
*
* This measure calculates the setup time value
*
.MEASURE Tran SetupTime    Trig v(Data)  Val = ‘v(Vdd)/2’
Fall = 1
+                          Targ v(Clock) Val = ‘v(Vdd)/2’
Rise = 1
*
* Optimization Model
*
.MODEL OptMod Opt
+ Method = Bisection
.OPTIONS Post Brief NoMod
**********************************
* AvanLink to Cadence Composer by Avant!
* Hspice Netlist
* May 31 15:24:09 1994
**********************************
.MODEL nmos nmos level=2
.MODEL pmos pmos level=2
.Global vdd gnd
.SUBCKT XGATE  control in n_control out
m0 in n_control out vdd pmos l=1.2u w=3.4u
m1 in control out gnd nmos l=1.2u w=3.4u
.ends
.SUBCKT INV  in out  wp=9.6u wn=4u l=1.2u
mb2 out in gnd gnd nmos l=l w=wn
mb1 out in vdd vdd pmos l=l w=wp
.ends
.SUBCKT DFF  c d nc nq
Xi64  nc net46 c net36 XGATE
Xi66  nc net38 c net39 XGATE
Xi65  c nq nc net36 XGATE
Xi62  c d nc net39 XGATE
Xi60  net722 nq INV
Xi61  net46 net38 INV
Xi59  net36 net722 INV
Xi58  net39 net46 INV
c20 net36 gnd c=17.09f
```

```
c15 net39 gnd c=15.51f
c12 net46 gnd c=25.78f
c4 nq gnd c=25.28f
c3 net722 gnd c=19.48f
c16 net38 gnd c=16.48f
.ENDS
*---------------------------------------------------------------
----------
* Main Circuit Netlist:
*---------------------------------------------------------------
----------
v14 vdd gnd dc=5
c10 vdd gnd c=35.96f
c15 d_output gnd c=21.52f
c12 dff_nq gnd c=11.73f
c11 net31 gnd c=42.01f
c14 net27 gnd c=34.49f
c13 net25 gnd c=41.73f
c8 clock gnd c=5.94f
c7 data gnd c=7.93f
Xi3  net25 net31 net27 dff_nq DFF l=1u wn=3.8u wp=10u
Xi6  data net31 INV
Xi5  net25 net27 INV
Xi4  clock net25 INV
Xi2  dff_nq d_output INV wp=26.4u wn=10.6u
.END
```

Results

The top plot in Figure 27-3: shows the relationship between the clock and data pulses that determine the setup time. The bottom plot shows the output transition.

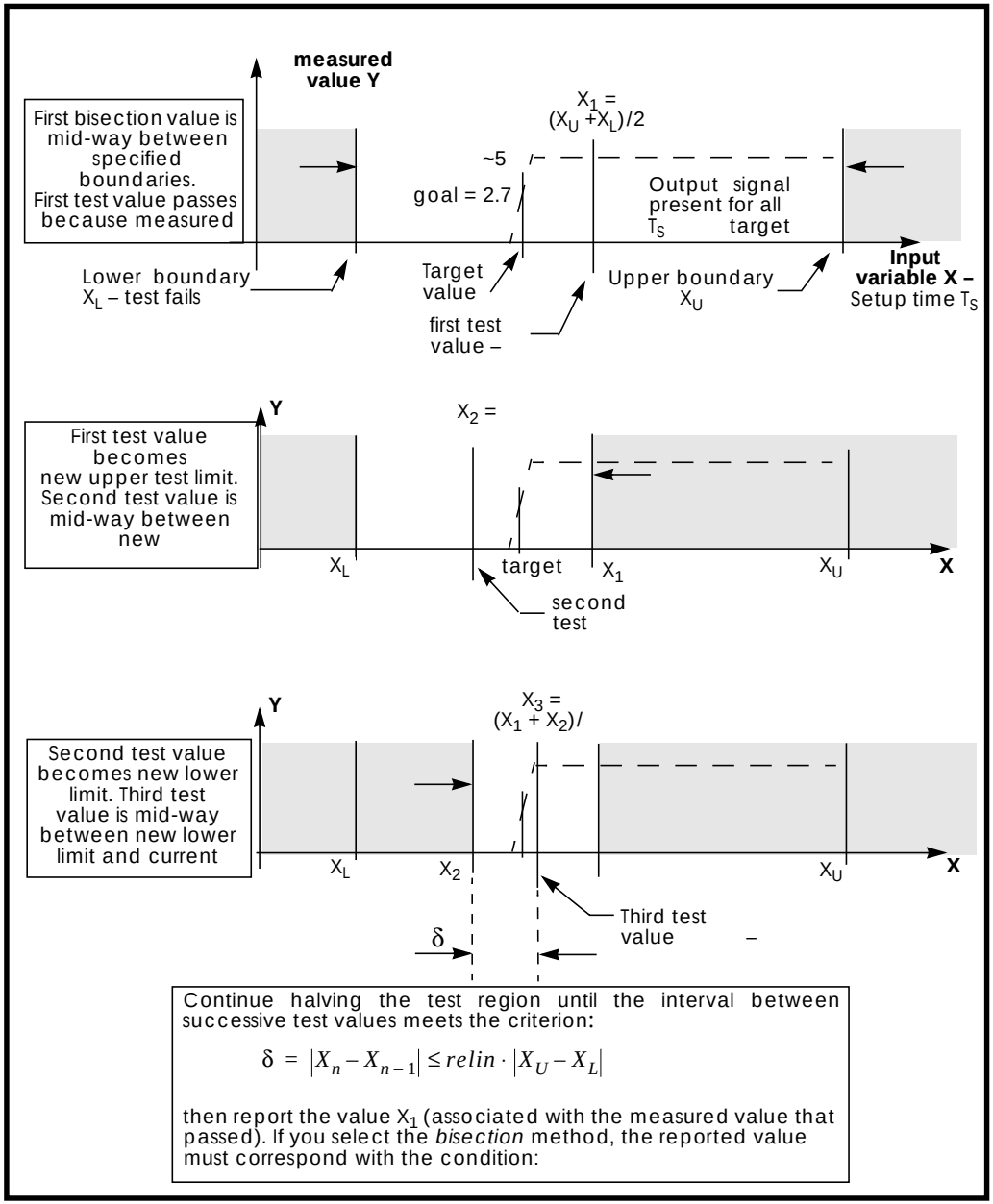

Figure 27-2: – Bisection Example for Three Iterations

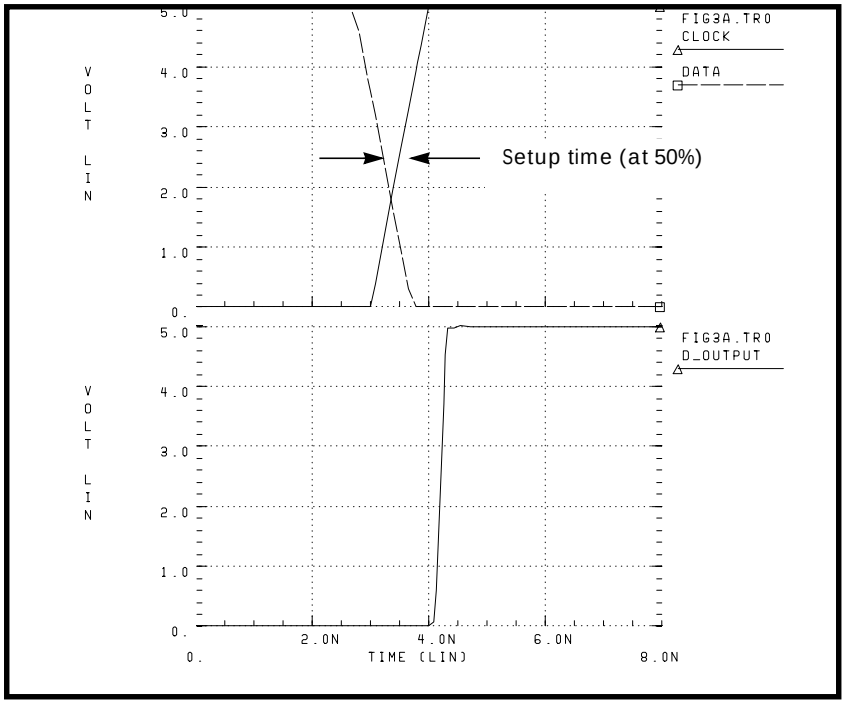

Figure 27-3: – Transition at Minimum Setup Time

Find the actual value for the setup time in the “Optimization, Results” section of the Star-Hspice listing file:

```
          optimization completed, the condition
          relin =  1.0000E-03 is satisfied
**** optimized parameters opt1
.PARAM DelayTime = 1.7188n
...
   maxvout  =  5.0049E+00    at=  4.5542E-09
   from     =   .0000E+00    to=  8.0000E-09
   setuptime=  2.8125E-10  targ=  3.5000E-09   trig=  3.2188E-
09
```

This listing file excerpt shows that the optimal value for the setup time is 0.28125 nanoseconds.

The top plot in Figure 27-4: shows examples of early and late data transitions, as well as the transition at the minimum setup time. The bottom plot shows how the

timing of the data transition affects the output transition. These results were produced with the following analysis statement:

```
* Sweep 3 values for DelayTime          Early Optim Late
*
.TRAN 1n 8n Sweep DelayTime Poi 3       0.0n  1.7188n 5.0n
```

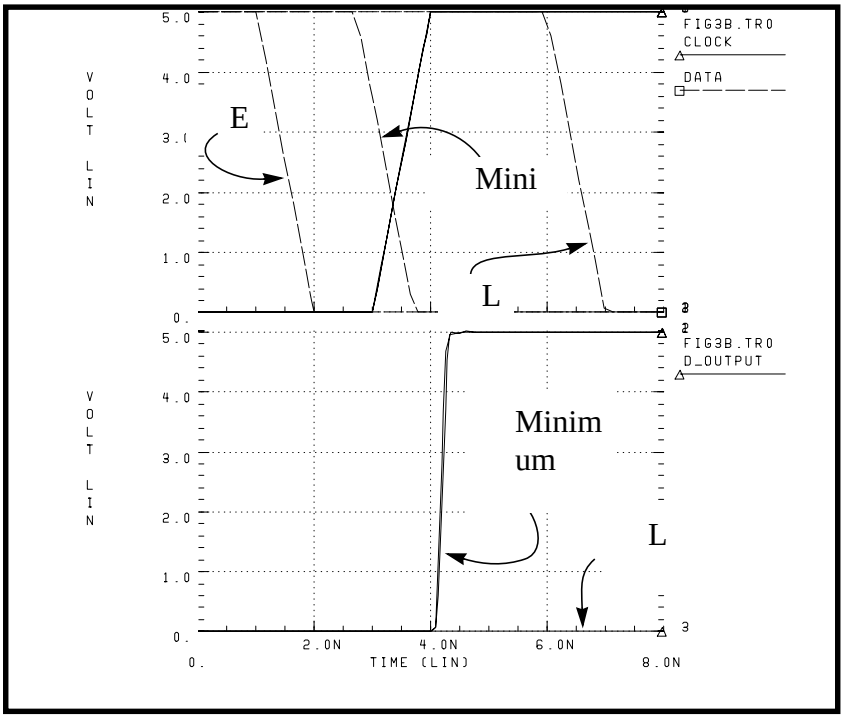

Figure 27-4: Early, Minimum, and Late Setup and Hold Times

This analysis produces the following results:

```
*** parameter DelayTime =     .000E+00 *** $ Early
setuptime       =  2.0000E-09  targ=  3.5000E-09   trig=
1.5000E-09
*** parameter DelayTime =    1.719E-09 *** $ Optimal
setuptime       =  2.8120E-10  targ=  3.5000E-09   trig=
3.2188E-09
*** parameter DelayTime =    5.000E-09 *** $ Late
setuptime       = -3.0000E-09  targ=  3.5000E-09   trig=
6.5000E-09
```

Example 2 – Minimum Pulse Width Analysis

This example uses a pass/fail bisectional search to find a minimum pulse width required to allow the input pulse to propagate to the output of an inverter. The circuit for this example is */bisect/inv_a.sp*. in the *$installdir/demo/hspice* directory. The results of this demo are shown in Figure 27-5:.

Input listing

```
File: $installdir/demo/bisect/inv_a.sp
$ Inv_a.sp testing bisectional search, cload=10p & 20p
*
* Parameters
.PARAM Cload   =10p                    $ See end of deck for
Alter
.PARAM Tpw     =opt1(0,0,15n)          $ Used in Pulsed
Voltage Source, v1
*
* Transient simulation with PassFail Optimization
*
.TRAN .1n 20n Sweep  Optimize    = Opt1
+                    results     = Tprop
+                    Model       = Optmod
.MODEL OptMod Opt    Method =    PassFail
.MEASURE Tran Tprop  Trig v(in)  Val=2.5 Rise=1
+                    Targ v(out) val=2.5 Rall=1
.OPTION nomod acct=3 post autostop
.GLOBAL 1
*
* The Circuit
*
vcc 1 0 5
vin in 0 pulse(0,5 1n 1n 1n Tpw 20n)
rin in 0 1e13
rout out 0 10k
cout out 0 cload
x1 in out inv
.SUBCKT Inv in out
    mn out in 0 0 nch W=10u L=1u
    mp out in 1 1 pch W=10u L=1u
.ENDS
```

```
*
* Models
*
.PARAM
+ mult1=1 xl=0.06u xwn=0.3u xwp=0.3u
+ tox=200 delvton=0 delvtop=0 rshn=50
+ rshp=150
.MODEL nch nmos
+ level=28
+ lmlt=mult1 wmlt=mult1 wref=22u lref=4.4u
+ xl=xl xw=xwn tox=tox delvto=delvton rsh=rshn
+ ld=0.06u wd=0.2u acm=2 ldif=0 hdif=2.5u
+ rs=0 rd=0 rdc=0 rsc=0 js=3e-04 jsw=9e-10
+ cj=3e-04 mj=.5 pb=.8 cjsw=3e-10 mjsw=.3 php=.8
+ fc=.5 capop=4 xqc=.4 meto=0.08u
+ tlev=1 cta=0 ctp=0 tlevc=0 nlev=0
+ trs=1.6e-03 bex=-1.5 tcv=1.4e-03
* dc model
+ x2e=0 x3e=0 x2u1=0 x2ms=0 x2u0=0 x2m=0
+ vfb0=-.5 phi0=0.65 k1=.9 k2=.1 eta0=0
+ muz=500 u00=.075 x3ms=15 u1=.02 x3u1=0
+ b1=.28 b2=.22 x33m=0.000000e+00
+ alpha=1.5 vcr=20 n0=1.6 wfac=15 wfacu=0.25
+ lvfb=0 lk1=.025 lk2=.05 lalpha=5
.Model pch pmos
+ level=28
+ lmlt=mult1 wmlt=mult1 wref=22u lref=4.4u
+ xl=xl xw=xwp tox=tox delvto=delvtop rsh=rshp
+ ld=0.08u wd=0.2u acm=2 ldif=0 hdif=2.5u
+ rs=0 rd=0 rdc=0 rsc=0 rsh=rshp js=3e-04 jsw=9e-10
+ cj=3e-04 mj=.5 pb=.8 cjsw=3e-10 mjsw=.3 php=.8
+ fc=.5 capop=4 xqc=.4 meto=0.08u
+ tlev=1 cta=0 ctp=0 tlevc=0 nlev=0
+ trs=1.6e-03 bex=-1.5 tcv=-1.7e-03
* dc model
+ x2e=0 x3e=0 x2u1=0 x2ms=0 x2u0=0 x2m=5
+ vfb0=-.1 phi0=0.65 k1=.35 k2=0 eta0=0
+ muz=200 u00=.175 x3ms=8 u1=0 x3u1=0.0
+ b1=.25 b2=.25 x33m=0.0 alpha=0 vcr=20
+ n0=1.3 wfac=12.5 wfacu=.2 lvfb=0 lk1=-.05
*
```

```
* Alter for second load value
*
.ALTER $ repeat optimization for 20p load
.PARAM Cload=20p
.END
```

Results

Figure 27-5: shows the results of the pass/fail search for two different capacitive loads.

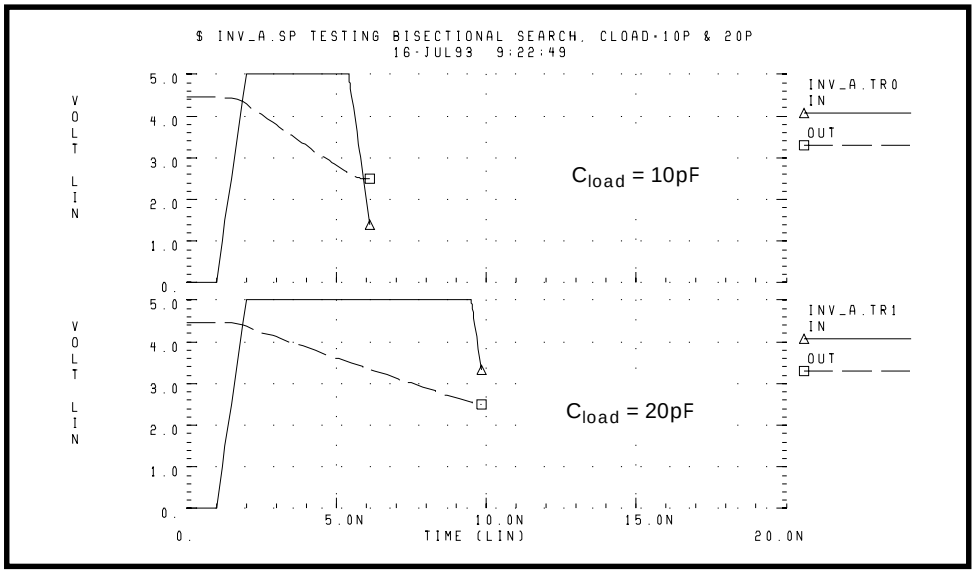

Figure 27-5: Results of Bisectional Pass/Fail Search