2.7.3 Modeling Delays

In SNL, delays may be specified either locally, in the PART or TYPE statement, or globally, in a **delay table**. Delays may depend on signal loading. Delay tables are specified in the **`!DELAY`** section of the SNL description, while pin and net loading are specified in PART and TYPE statements of the **`!LOGICAL`** section. This section describes these SNL constructs.

The SIMIC compiler adds all pin loading on each net (i.e., the loading at each element pin connected to the net) to the net's contribution (wiring capacitance) to obtain the total net loading. It then references the driver delay characteristics to compute the rise and fall delays of the net's driver(s).

Delays can also be changed at run time using the **`SET`** command.

2.7.3.1 SIMIC Time-Units

In SIMIC, a time-unit is the smallest quantum of time that can be processed. All SIMIC delays are expressed in time-units. Time-units are normalized time measurements; they can represent any real-time value (e.g., picoseconds, nanoseconds), depending on the selected scale factor used to define delays. Care must be taken in this selection; if a technology has gate delays in the nanosecond range, and delays have been scaled so that 1 time-unit represents 1 picosecond, then each signal delay will be thousands of time-units. During simulation, cumulative time will increase rapidly. Since SIMIC maintains elapsed simulated time internally as a 32 bit integer, the maximum possible simulation time is 2,147,483,647 time-units. Large internal delays would unnecessarily limit the number of stimuli that can be simulated before maximum time is reached.

The correspondence between time-units and real-time can be optionally entered in the **`!DELAY`** statement's **`TIME-UNITS`** keyword-field. For example:

```
!DELAY time-units=1e-9
```

specifies that one time-unit corresponds to one nanosecond. If this correspondence is specified, SIMIC reports its value at run time.

2.7.3.2 Delay Table Coefficients

Delays are specified with fixed-point or floating-point numbers having up to six significant digits. A fixed-point number is a decimal number, possibly containing a decimal point (a rightmost decimal point is implicit for integers). A floating-point number is a fixed-point number multiplied by an integral power of 10; the letter "E" (or "e") separates this exponent from the fixed-point number. The plus (+) sign for positive exponents is optional. For example, the number 72 can be represented as:

`72, 72., 7200e-2, .0072e4,` etc.

2.7.3.3 Linear Delay Curves

For some technologies, propagation delays depend only on, and vary linearly with, capacitive loading at cell outputs. This relationship, illustrated in Figure 2.7-4, can be described as either:

1. A line that goes through the coordinates (2,4) and (6,8), or,
2. A line that has a y-intercept of 2 and a slope of 1.

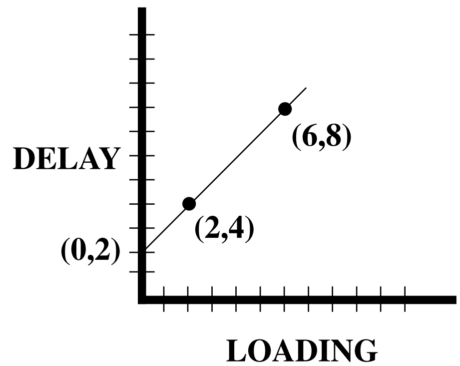


Figure 2.7-4 Typical Delay vs. Loading Relation

The two-point form is:

`(`*<load1>*`,`*<delay1>*`) (`*<load2>*`,`*<delay2>*`)`

In this example, the two-point representation would therefore be:

`(2,4)(6,8)`

which defines a delay whose value is 4 when the loading is 2, and 8 when the loading is 6. Optional whitespace may be placed between the right parenthesis of the first point and the left parenthesis of the second.

The intercept-slope form is:

`[`*<intercept>*`,`*<slope>*`]`

In this example, the intercept-slope representation would be:

`[2,1]`

Two-point and intercept-slope forms may be used interchangeably.

Constant delays are a special case, with zero slope. For example, a constant delay of 4 could be represented as:

`4` or `(1,4)(10,4)` or `[4,0]` etc.

Note: if propagation delays depend on input rise and fall times, or if they vary non-linearly with output capacitance, the simple linear delay vs. loading relationship described here would not be applicable. The delay specification formats described in Section 2.7.3.6 (page 2.7-11) should be used instead.

2.7.3.4 Global Delays

Global delays are specified in the **!DELAY** section of the SNL description. Rise and fall delays are independently specifiable. Each global delay characteristic is assigned a unique name that is referenced by individual PART and TYPE statements in the **!LOGICAL** section. Each global delay definition has the form:

DELAY=<*name*> RISE=<*delay*> FALL=<*delay*>

for delay characteristics having different rise and fall delays, or

DELAY=<*name*> CHANGE=<*delay*>

for delay characteristics having identical rise and fall delays. In both forms, ***<delay>*** is a delay specification. For example:

```
DELAY=del5 RISE=[2,1] FALL=7
```

defines a global delay having a rise delay characteristic as shown in Figure 2.7-4 (page 2.7-8) and a constant fall delay of 7, independent of loading.

The <***delay***> specification may contain minimum and maximum delays as well as typical delays. Its complete format is:

<*typical-delay*>;<*minimum-delay*>;<*maximum-delay*>

where each delay specification is either a two-point or intercept-slope description. For example:

```
DELAY=del12_8_15  CHANGE=[12,1];8;(0,15)(2,19)
```

The typical delay must be specified. If either the minimum or maximum delay is omitted, it defaults to the typical delay.

Delays are assigned to TYPE and PART outputs with the **OUTPUT-DELAY (ODEL)** keyword-field, and to busses with the **BUS-DELAY (BDEL)** keyword-field. The value part of these keyword-fields, which contains the global delay names, is in a one-to-one correspondence with the respective outputs and busses. For example,

```
P=a2  T=full-adder  I=a[2],b[2],c[1] $
   O=s[2],c[2]  ODEL=del5,del12_8_15
```

assigns the global delay **del5** to output **s[2]** and **del12_8_15** to **c[2]**.

If a signal's delay is unspecified, the delay defaults to 0. Thus, if the first delay in the above PART statement were omitted:

```
P=a2  T=full-adder  I=a[2],b[2],c[1] $
     O=s[2],c[2]  ODEL=,del12_8_15
```

the rise and fall delays of signal **s[2]** default to 0. Note the placeholder comma preceding the delay name **del12_8_15**. If this comma were missing, then delay **del12_8_15** would be assigned to **s[2]** and the rise and fall delays of **c[2]** would default to 0.

As mentioned above, the global delay definitions are specified in the **!DELAY** section, and are referenced by PART and TYPE statements in the **!LOGICAL** section:

```
!DELAY time-units=1e-9
     .........
DELAY=del5 RISE=[2,1] FALL=7
DELAY=del12_8_15  CHANGE=[12,1];8;(0,15)(2,19)
     .........
     .........
     .........
!LOGICAL
     .........
 P=a2  T=full-adder  I=a[2],b[2],c[1] $
     O=s[2],c[2]  ODEL=del5,del12_8_15
     .........
```

The **!LOGICAL** section could be in the same network description file as the **!DELAY** section, or it could be in a different file. Delays defined in a **!DELAY** section are truly global; that is, available to PART and TYPE statements in all network description files explicitly selected with the **GET FILE** keyword option or referenced by an **!INCLUDE** statement. Furthermore, the global delay tables can be contained in multiple **!DELAY** sections, possibly in different files. Regardless of how the delay tables are organized, however, global delay names must be unique. Assigning the same name (e.g., **del5**) to two *different* delay characteristics is a fatal error.

Global delays are very convenient when constructing simulation libraries for variants of a particular technology. Here, the functions of the logic cells and loading characteristics are identical, but each variant requires a different set of delay-vs.-loading curves. This is accomplished very simply by creating separate files, each containing a **!DELAY** section for one variant, and then referencing the appropriate delay table file with the **GET FILE** option.

2.7.3.5 Local Delays

Delays may also be assigned locally, within a PART or TYPE statement, without referencing a named global delay. Local delays are unnamed; they are specified directly. This is especially useful for generating SNL descriptions with automated netlisters. The format for defining local delay characteristics is identical to the ***<delay>*** format, described above, for the **DELAY** statement's **RISE**, **FALL**, and **CHANGE** keywords. The corresponding keywords for PART and TYPE busses are **BUS-RISE (BRISE)**, **BUS-FALL (BFALL)**, and **BUS-CHANGE (BCHANGE)**. For outputs, the corresponding keywords are **OUTPUT–RISE (ORISE)**, **OUTPUT–FALL (OFALL)**, and **OUTPUT–CHANGE (OCHANGE)**.

For example, assume that a type block named **my_type** is being defined for a subcircuit that has three input pins, two bidirectional pins, and two outputs pins. Then the TYPE statement:

```
T=my_type  I=a,b,c  B=d,e  O=f,g  BCHANGE=3,4  $
  ORISE=(5,3)(20,4)  OFALL=[3,1]  OCHANGE=,2
```

assigns:

bus **D** rise and fall delays of 3
bus **E** rise and fall delays of 4
output **F** a rise delay of (5,3)(20,4) and a fall delay of [3,1]
output **G** rise and fall delays of 2.

2.7.3.6 Delays Dependent On Input Slew-Rate

If input rise and fall times (slew-rates) can significantly affect an element's propagation delays, or if propagation delays vary non-linearly with output capacitance, use of the simple linear delay vs. loading curve (see Section 2.7.3.3 on page 8) may introduce unacceptable errors in computed delays. To handle such technologies, SIMIC supports an alternative, flexible method of describing the (generally non-linear) dependence of propagation delays on output loading and input slew-rates.

Figure 2.7-5 illustrates the SIMIC model for slew-rate dependent delays. As shown in Figure 2.7-5(a), the actual waveforms are represented by idealized rail-to-rail ramps. In practice, the ramps would be constructed by connecting the points at which the waveform passes through two predefined thresholds[1] with a straight line, and extending this line to the rails. The rail-to-rail ramp interval, TR, depicts the waveform's transition time.

In order to obtain the required characterization data, you would perform sets of measurements or simulations, linearize the waveforms, and obtain per-cell data on the following three variables (see Figure 2.7-5(b)):

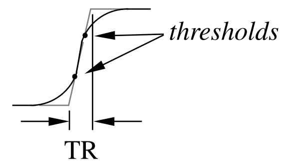
TR_{in} – the ramp interval of the cell's input transition (generated by a driving cell or primary signal),

T0 – the time interval from the start of the input ramp to the start of the output ramp, and

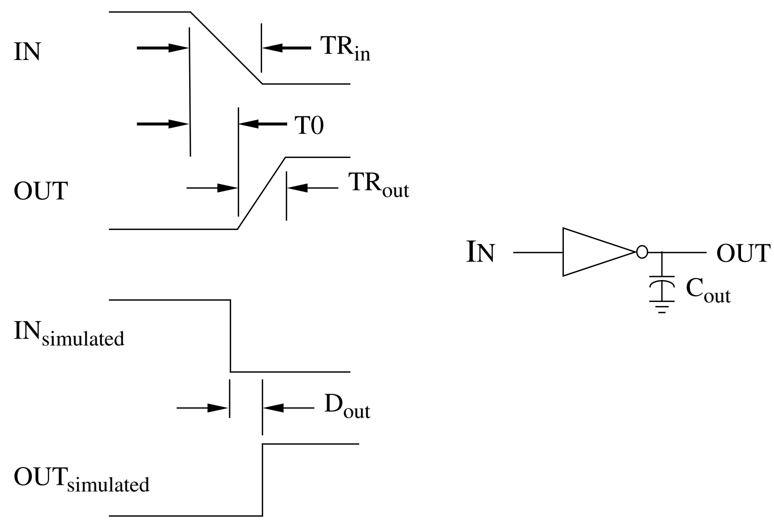
TR_{out} – the ramp interval of the cell's output.

The measured values should then be scaled to SIMIC time-units.

1. Typical threshold values might be the 30% and 70% points, the 35% and 65% points, etc. In general, the threshold values should not be critical, as long as they are used consistently and are well-selected, i.e., they should bracket the 50% point and be far enough apart to avoid numerical errors, yet close enough to exclude most of the waveform tails.



(a) Waveform Linearization



(b) Delay Model Variables

Figure 2.7-5 Slew-Rate Delay Model

The delay model assumes that the output measurements, T0 and TR_{out}, are linearly related to the input ramp interval, TR_{in}, and the output loading capacitance, C_{out}. These relations are described with six parameters; *a*, *b*, *c*, *d*, *e*, *f*:

$$T0 = a + b \times C_{out} + c \times TR_{in}$$
$$TR_{out} = d + e \times C_{out} + f \times TR_{in}$$

After obtaining the measurements, you[2] would then calculate values for the six parameters to fit the data and generate the corresponding SIMIC delay tables. The format for this delay specification is:

(*maxload* [*a*, *b*, *c* ,*d*, *e*, *f*])

2. Since the operations of obtaining the measurements, linearizing the waveforms, obtaining per-cell values for the three variables, calculating the six coefficients to fit the data, and printing them in the proper format are all regular and repetitive, they can readily be automated, and the amount of actual manual effort need be minimal.

where ***a***, ***b***, ***c***, ***d***, ***e***, and ***f*** are the values of the six coefficients, and ***maxload*** is the maximum value of C_{out} for which measurements were taken.

For example:

```
Delay=Example1 Change=$
 (3e-12 [-6.28, 6e12, .47, 25.3, 1.5e13, .24])
```

If ***maxload*** is exceeded in the circuit being simulated, SIMIC will issue a warning.

After reading the circuit description and delay tables, SIMIC computes the values of T0 and TR_{out} for each element output. It then assigns rise and fall delays to each signal (which will execute delayed instantaneous transitions) that extend from the 50% point of the input ramp to the 50% point of the output ramp. For example, referring to Figure 2.7-5(b), the square waveforms illustrate the simulated events corresponding to the waveforms of the actual circuit. The inverter's rise delay would be:

$$D_{out} = T0 + 0.5\times(TR_{out} - TR_{in})$$

In general, the actual dependence of T0 and TR_{out} on loading or input ramp may be non-linear. Thus, SIMIC supports a piecewise linear table format to describe either or both relationships. To describe a non-linear loading relationship, you specify the coefficients within the regions of the curve, and the breakpoints from one region to the next in the **Ctable** format:

(*maxload1* [*a1*,*b1*,*c1, d1*,*e1*,*f1*] *maxload2* [*a2*,*b2*,*c2, d2*,*e2*,*f2*] ...)

where ***maxload1***, ***maxload2***, etc. are the upper breakpoints for the linear regions, and ***a1-f1***, ***a2-f2***, etc. are the lists of six coefficients that apply for the first region, the second region, etc. For example:

```
Delay=Example2 Change=($
 1e-13 [.4, -2.5e-12, .6, 3.1, 4.3e13, -.3] $
 3e-13 [.6, -2.3e-12, .5, 3.6, 4.6e13, -.4] $
)
```

In this example, the first set of coefficients are used if the loading is less than or equal to 1e-13, and the second set of coefficients are used if the loading is above 1e-13 and less than or equal to 3e-13. Because this is the last region described, it will also be used for loadings above 3e-13, but a warning will be issued.

Note that as a special case, the **Ctable** format can also be used to handle non-linear delay vs. loading characteristics for technologies in which output loading is the only significant factor affecting propagation delays. This is accomplished by setting the ***c***, ***d***, ***e***, and ***f*** coefficients (those associated with input slew-rates) to 0. Essentially, each entry specifies the intercept and slope of the piecewise linear approximation for a range of capacitive loads. For example:

```
Delay=Example2 Change=($
 1e-13 [.4, -2.5e-12, 0, 0, 0, 0] $
 3e-13 [.6, -2.3e-12, 0, 0, 0, 0] $
)
```

To describe non-linear input ramp relationships in a piecewise linear fashion, use the **Rtable** format:

(*maxramp1 Ctable1 maxramp2 Ctable2* ...)

where ***maxramp1***, ***maxramp2***, etc. are the upper breakpoints for the linear regions, and ***Ctable1***, ***Ctable2***, etc. are in the **Ctable** format described above. The following is an example of a segmented input slew table:

```
Delay= N1P1 $
      Rise= ( 5     (7e-13 $
                      [0.4,-2.5e12,0.6,3.2,4.3e13,-0.4])$
             10     (3e-12 $
                      [0.6,-2.3e12,0.5,3.6,4.6e13,-0.4])$
            250     (3e-12 $
                      [8.9,-4.8e12,0.7,-4.8,4.6e13,0.1]))$
      Fall= ( 5     (7e-13 $
                      [0.2,-1.4e12,0.6,2.5,2.8e13,-0.4])$
             10     (3e-12 $
                      [0.03,-9.8e11,0.6,3.3,2.9e13,-0.6])$
            100     (3e-12 $
                      [1.1,-1.1e12,0.6,9.4,2.5e13,0.06])$
            250     (3e-12 $
                      [-6.3,6e12,0.5,25.3,1.5e13,0.2]))
```

Note: like the linear two-point and intercept-slope delay forms, the **Ctable** and **Rtable** forms can be used in local as well as global delay specifications.

Since the cells driven by the primary inputs must also be assigned delays, primary inputs should be assigned a reasonable “default” ramp interval. This is accomplished with **`RAMP`** keyword in the **`!DELAY`** section. For example:

```
!DELAY RAMP=5
```

assigns the ramp interval TR = 5 time-units at each of the primary signals. The specified ramp interval is also assigned at all internal signals whose delay tables are not in **Rtable** format to allow intermixing of input slew-rate dependent and input slew-rate independent cells in the same design with reasonable results. The **`RAMP`** keyword may also be specified on a separate line within the **`!DELAY`** section.

2.7.3.7 Specifying Pin Loading

Loading can be specified for all pins of a TYPE statement and all pins of an instantiated component in a PART statement. A load value can also be specified for each signal, typically representing wiring capacitance. Load values are specified as fixed-point or floating-point numbers. If no loading is specified for a signal or a pin, then the loading value is defaulted to 0.

Like delays, the general format for specifying loading is a three-tuple of values—typical, minimum, and maximum—separated by semicolons.

The keywords for specifying pin loads in PART and TYPE statements are **`BUS-LOADS (BLOD)`**, **`INPUT-LOADS (ILOD)`**, and **`OUTPUT-LOADS (OLOD)`** for bus, input, and output pins, respectively. The specified values are in a one-to-one correspondence with the respective signals. For example, adding loading to the above TYPE statement for **`my_type`**:

```
T=my_type  I=a,b,c  B=d,e  O=f,g  BCHANGE=3,4  $
  ORISE=(5,3)(20,4)  OFALL=[3,1]  OCHANGE=,2  $
   ILOD=1,,3 BLOD=5;4;6,7.89 OLOD=10,11
```

assigns:

input pin **A** a load of 1
input pin **B** a load of 0 (since no loading is specified)
input pin **C** a load of 3
bus pin **D** typical,minimum,maximum loading of 5;4;6
bus pin **E** a load of 7.89
output pin **F** a load of 10
output **G** a load of 11.

Nets are assigned loading by instantiating the built-in LOAD element:

P=<*name*> T=load O=<*net-name*> OLOD=<*value*>

or

P=<*net-name*> T=load OLOD=<*value*>

where:

- ***<net-name>*** is the hierarchical name of the net
- ***<value>*** is the load value assigned to the net.

In the first form, with the **`OUTPUT`** keyword-field explicitly specified, the assigned part name, ***<name>***, is *arbitrary* (this is the only case where a part name has no significance).

As an example of the second form, the statement:

```
PART=a.b.c  TYPE=load  OLOD=3;2;5
```

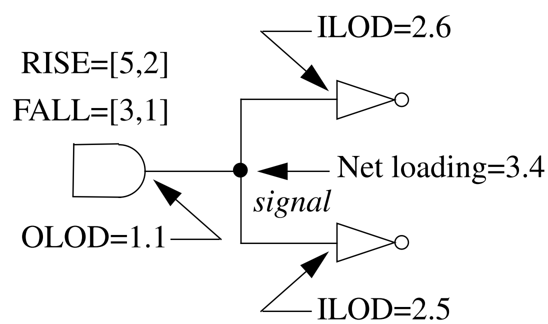
assigns typical,minimum,maximum loading of 3,2,5, respectively, to the signal named **`a.b.c`**.

2.7.3.8 Resultant Delays

When a load-dependent driver delay is specified, either by reference to a global delay or by a local delay, the SIMIC compiler totals all loading associated with the driven net to obtain the resultant delay from the specified delay vs. loading characteristic. Calculations are performed in floating point, and the results rounded to the nearest integer.

For example, in the circuit shown in Figure 2.7-6, the total loading on the net named **`signal`** is (1.1 + 2.6 + 2.5 + 3.4 =) 9.6. Thus, the AND gate's rise delay is 24 (5 + 2×9.6 = 24.2 rounded down), and its fall delay is 13 (3 + 1×9.6 = 12.6 rounded up).

If the result after interpolation is negative, SIMIC sets the delay to 0.




```
P=driver t=and i=a,b o=signal orise=[5,2] $
   ofall=[3,1]  olod=1.1
p=load1  t=inv  i=signal  ilod=2.6
p=load2  t=inv  i=signal  ilod=2.5
p=signal t=load olod=3.4
```

Figure 2.7-6 Delay Computation Example

2.7.3.9 Delays At Paralleled Elements

Drivers are sometimes paralleled to decrease rise and fall delays for heavily-loaded busses. Usually, the drivers are either identical or at least close in their drive capabilities. The SIMIC compiler considers elements paralleled when (1) their outputs are tied together, (2) they are all the same type, which must be one of the following built-in primitives: INV, AND, NAND, OR, NOR, BTGRN, BTGRP, UTGRN, or UTGRP, and (3) they have the same inputs (though not necessarily in the same order). When these criteria are met, SIMIC replaces the paralleled elements with a single element whose delay characteristic has:

1. a y-intercept that is the minimum y-intercept of all the paralleled elements, and

2. a slope obtained by treating the slope of each element's delay vs. loading characteristic as a resistance and computing the equivalent resistance of the parallel combination

The rise and fall delay constructions are performed independently. Figure 2.7-7 illustrates this replacement.

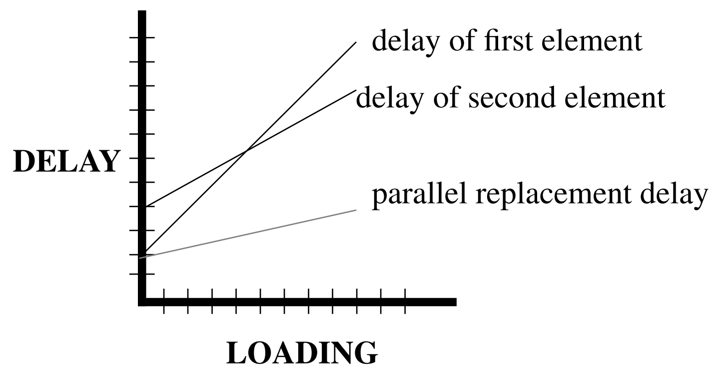


Figure 2.7-7 Illustration of Parallel Element Delay Reduction

2.7.3.10 Modifying Delays At Run Time

The **`SET`** command can be used to modify delays at run time. This is useful for circuit debugging and experimentation. The options that can be specified are:

1. **<n>** - set delay to the specified value, **<n>** (an integer)
2. **+<p>%** or **-<p>%** - increase or decrease delay by the specified percentage, **<p>** (an integer). (Note: the + sign is optional)

The **`RISE`**, **`FALL`**, and **`CHANGE`** keywords are used to specify that rise delays, fall delays, or both, respectively, be modified in the requested manner. The **`LIST`** keyword specifies which signals should be affected. For example,

```
SET RISE=30 LIST=a,b,c
```

sets the rise delay of the three signals to 30, while

```
SET CHANGE=-10% LIST:
```

reduces all delays by 10% (note that specification of a percentage less than, or equal to, -100% will set the affected delays to 0).

The Section *Modifying Delay Values* in Chapter 2.6 contains a more complete description of interactive delay modification.