

Guide to Writing Stimulus Files

(ver. S06: modified for using *spectre*)

A Cadence EDA Tools Help Document

Document Contents

Introduction

Voltage Source Definitions

Defining Sources for Specific Simulations

RCL Component Definitions

Introduction

Cadence Tutorial A introduces functional simulation of digital circuits by using transient simulations and Tutorial C describes additional simulation techniques. This document describes the syntax for defining voltage sources in text-based stimulus files for the Cadence *spectre* simulator.

General notes about including your stimulus files

- If you include your file by using “Setup -> Simulation Files -> Stimulus File” in the Affirma analog environment (as in Tutorial A), you must follow these rules when writing your stimulus file:

1. The first line of your stimulus file must read:

```
simulator lang=spectre
```

If it does not, you will likely have errors when you try to generate the netlist from your design.

2. You must use the backslash character ‘\’ before any left square bracket ‘[’. A backslash will ‘escape’ the character so that it is interpreted in its *spectre* statement. If you do not do this, you will encounter a syntax error during circuit netlisting which flags the left square bracket as an invalid character.

General Rules

- Voltage source and RLC component names may be any alphanumeric string, and do not need to start with any particular letter or case.
- All voltage sources and RLC components must be defined between a positive (+) and negative (-) node using node names established by pins/nets in the schematic.
- Node names may be alphanumeric strings (like ‘A1’ or ‘input’) or node numbers.
- The ground node is preset to node number 0. Global gnd! is the same as node 0.
- All lines in a *spectre* stimulus file, including the last line, must be terminated by the newline character (i.e. pressing enter).
- If you want to extend a command or component definition to multiple lines for the sake of improving readability, you must use the ‘+’ character at the beginning of each extended line, followed by a space.
- You can comment a line in a stimulus file by using two consecutive forward slashes ‘//’.
- When specifying any numeric value, you can use exponential abbreviations. For instance, a value of 5 nanounits could be written as ‘5n’
 - ‘M’ for mega

- ‘K’ for kilo
- ‘m’ for milli
- ‘u’ for micro
- ‘n’ for nano
- ‘p’ for pico
- ‘f’ for femto

Voltage Source Definitions

DC Source

Format:

```
<name> (+terminal -terminal) vsource dc=value
```

A DC source defines a constant (DC) voltage between two nodes (*+terminal and -terminal*) with a *value* in volts. It is commonly used to define the global power supply voltage, vdd!, or other fixed sources.

Example:

```
vdd (vdd! 0) vsource dc=3
```

This statement applies 3 volt source named “vdd” between the vdd! and 0 (gnd!) terminals.

Voltage Pulse Source

Format:

```
<name> (+terminal -terminal) vsource type=pulse  
val0=v1 val1=v2 delay=td rise=tr fall=tf width=pw period=per
```

A pulse source is a periodic voltage that changes between a high and low value with timing characteristics defined in Figure 1. Timing characteristics are delay, rise time, fall time, pulse width, and pulse period.

Example:

```
vA (A 0) vsouce type=pulse val0=0 val1=3 delay=0  
rise=0.05n fall=0.05n width=5n period=10n
```

This statement will apply a 3V square(approximately) pulse between nodes A and 0 with a 0.05nsec rise and fall time, a pulse width of 5nsec and a period of 10nsec (50% duty cycle). This could be used to generate a 100MHz clock with 0.05nsec rise/fall time.

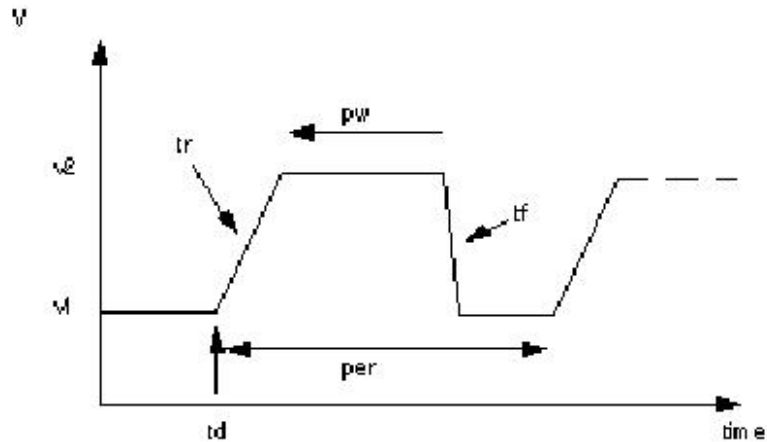


Figure 1. Pulse waveform generated by *spectre* (source openbook)

Piecewise Linear Source

Format:

```
<name> (+terminal -terminal) vsource type=pwl
      wave=[ t1 v1 t2 v2 t3 v3 t4 v4 ... tn vn ]
```

This statement allows a voltage source to be defined by a series of time/voltage points as shown in Figure 2. It can be used to generate complex source signals as shown in the bit-stream generation example below. This example also illustrates how to set variables and use them within source statements to simplify defining quasi-periodic waveforms.

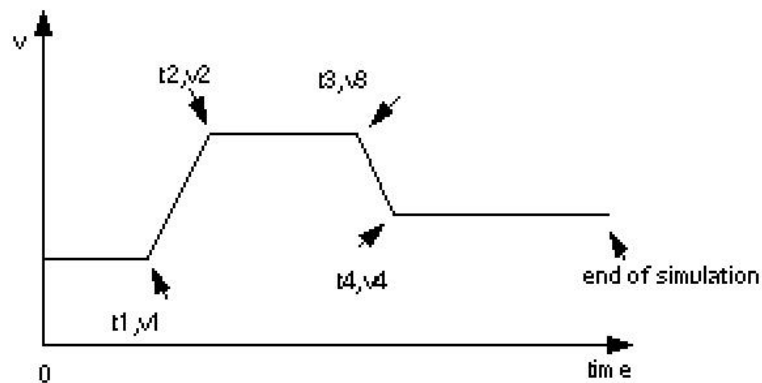


Figure 2. Piecewise Linear waveform (source openbook)

(note: If you are running into syntax errors when using PWL sources, use a backslash character '\ ' before every left square bracket '['. See note at top of file.)

Defining Voltage for Specific Simulations

Testing Logic Truth Tables

It is always desirable to check how a circuit will work for all possible logic combinations. Setting pulse waveforms with width and period twice of another will simulate this condition. For three inputs, the third source should have four times the period, etc.

Example:

```
vdd (vdd! 0) vsource dc=3
vA (A 0) vsource type=pulse val0=0 val1=3 delay=0 rise=0.05n fall=0.05n
+ width=10n period=20n
vB (B 0) vsource type=pulse val0=0 val1=3 delay=0 rise=0.05n fall=0.05n
+ width=20n period=40n
```

(note: the '+' symbol extends component declaration to multiple lines. If you wish to keep the entire declaration of any voltage source in a single line, omit the '+' symbol)

This code generates a DC supply voltage and the waveform shown in Figure 3.

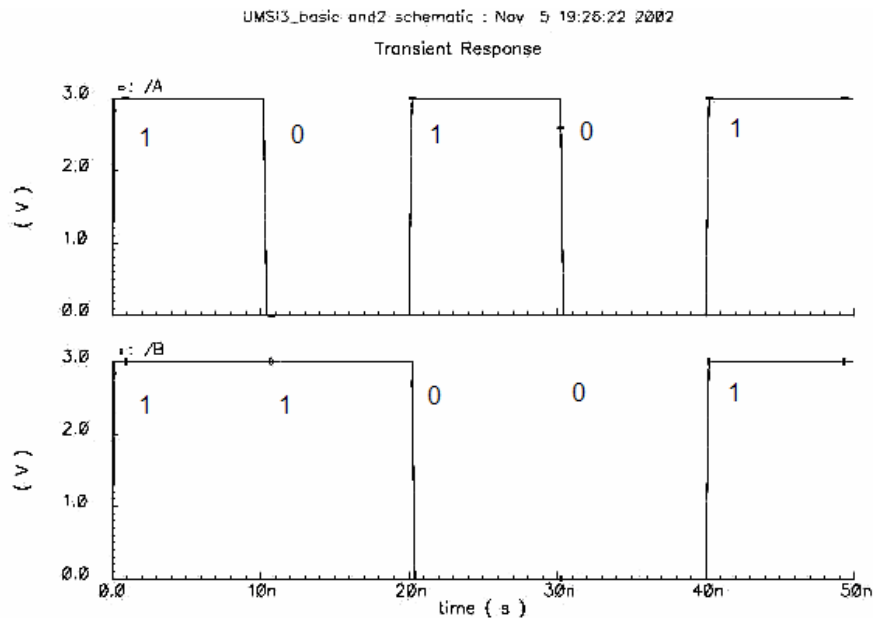


Figure 3. Input for truth table analysis

This example illustrates that all possible 2-input logic combinations (“00”, “01”, “10”, “11”) are tested every 40n.

Timing Analysis for Multi-Input Circuits

When simulating a multi-input circuit to study the effect of internal capacitances (timing analysis), it is desirable to have non-overlapping waveforms. This is effectively done by using 2 pulses of same width and period by just giving a delay to one of them. Notice however that this does not test the 11 → 00 or 00 → 11 case. You will need a different stimulus file for these cases.

Example:

```
vdd (vdd! 0) vsource dc=3
```

```

vA (A 0) vsource type=pulse val0=0 vall=3 delay=0 rise=0.05n fall=0.05n
+ width=20n period=40n
vB (B 0) vsource type=pulse val0=0 vall=3 delay=10n rise=0.05n fall=0.05n
+ width=20n period=40n

```

These sources will apply the waveform shown in Figure 4 to a two-input circuit and allow you to test the timing characteristics.

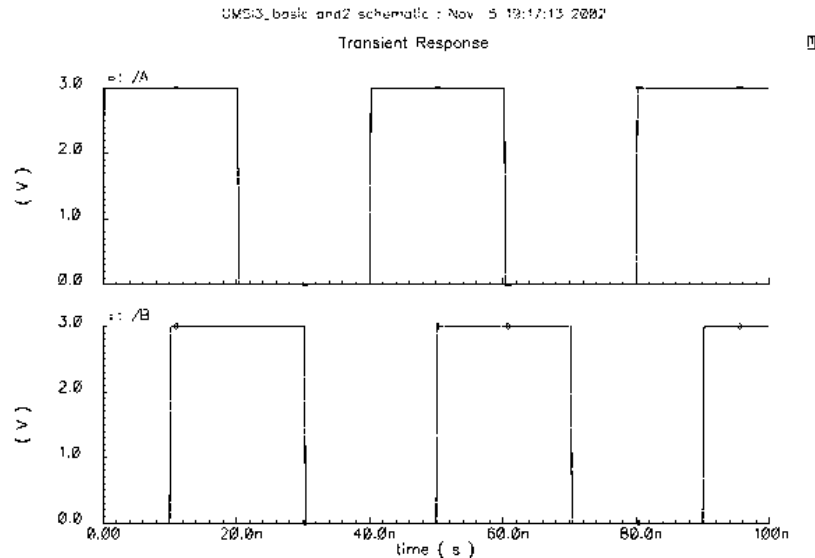


Figure 4. Input for timing analysis

Multiple Synchronized Signals

Often you need to generate a series of signals that are synchronized but have transitions at different times. For example, when working with microprocessor simulations, you might want to apply voltage inputs as a bus. One way to achieve this is by using multiple pulse sources and varying the pulse width and/or period.

Example:

```

vdd (vdd! 0) vsource dc=3
vA0 (A_0 0) vsource type=pulse val0=0 vall=3 delay=0 rise=0.05n fall=0.05n
+ width=10n period=20n
vA1 (A_1 0) vsource type=pulse val0=0 vall=3 delay=0 rise=0.05n fall=0.05n
+ width=20n period=40n
vA2 (A_2 0) vsource type=pulse val0=0 vall=3 delay=0 rise=0.05n fall=0.05n
+ width=10n period=20n
vA3 (A_3 0) vsource type=pulse val0=0 vall=3 delay=0 rise=0.05n fall=0.05n
+ width=20n period=40n

```

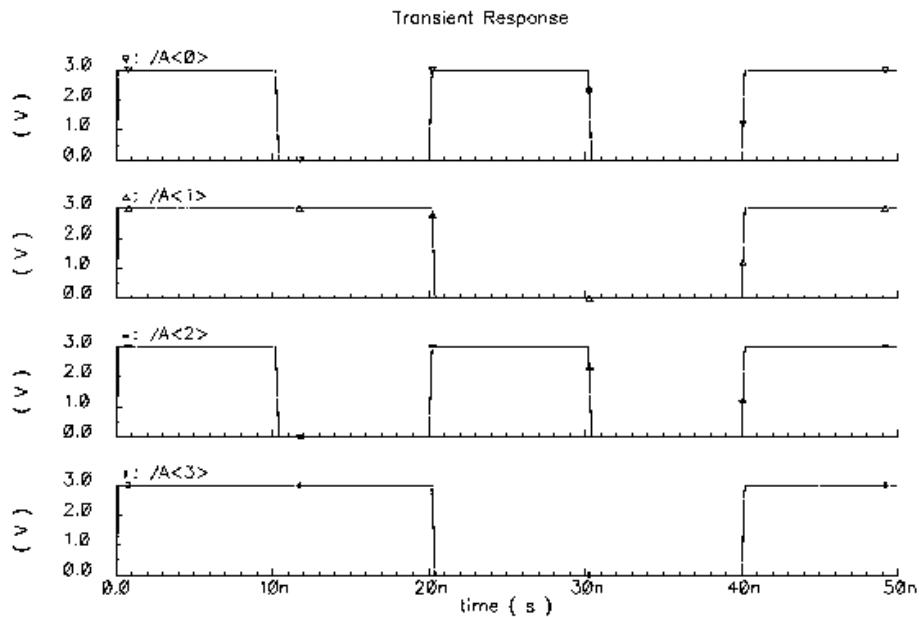


Figure 5. Input for a bus simulation

Bit-stream generation

When you are testing circuits like a serial to parallel converter, it is desirable to give an input bit sequence like 10110111 or 10011101 etc. The piecewise linear (pwl) waveform will help you to test such circuits.

Example:

```
vdd (vdd! 0) vsource dc=3
```

```
//K is the signal period
```

```
parameters K = 20n
```

```
//D1 is the delay of vA0
```

```
parameters D1 = 0n
```

```
//D2 is the delay of vA1
```

```
parameters D2 = 20n
```

```
vA0 (A_0 0) vsource type=pwl wave=[
+ 0n 0 D1+0.0001n 0
+ 1*K+0.05n+D1 0 2*K-0.05n+D1 0
+ 2*K+0.05n+D1 3 4*K-0.05n+D1 3
+ 4*K+0.05n+D1 0 5*K-0.05n+D1 0
+ 5*K+0.05n+D1 3 8*K-0.05n+D1 3
+ ]
```

```
vA1 (A_1 0) vsource type=pwl wave=[
+ 0n 0 D2+0.0001n 0
+ 1*K+0.05n+D2 0 3*K-0.05n+D2 0
+ 3*K+0.05n+D2 3 6*K-0.05n+D2 3
+ 6*K+0.05n+D2 0 7*K-0.05n+D2 0
+ 7*K+0.05n+D2 3 8*K-0.05n+D2 3
+ ]
```

In this example a bit stream with a width of 20 ns is generated. The rise and fall times of the wave are 0.1 ns. For the second waveform the delay is controlled by changing D2. Figure 6 illustrates the waveforms in this example.

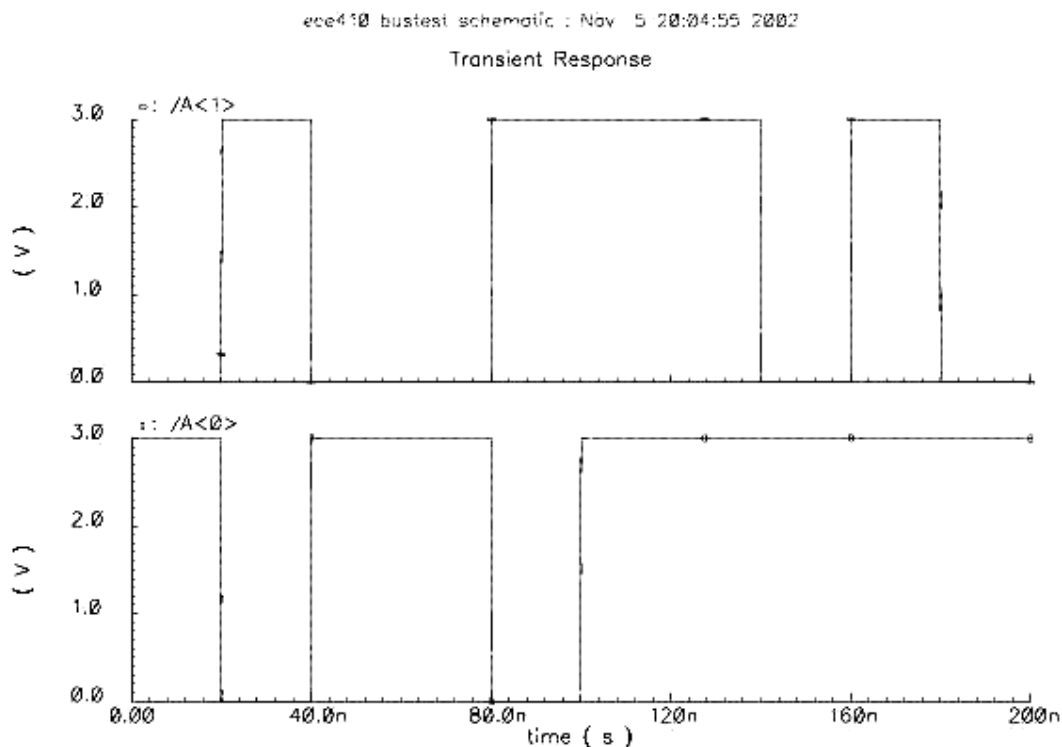


Figure 6. Bit sequence generation

(note: If you are running into syntax errors when using PWL sources, use a backslash character '\ ' before every left square bracket '['. See note at top of file.)

RLC Component Definitions

Resistor

Format:

```
<name> (+terminal -terminal) resistor r=value
```

Places a resistor between nodes '+terminal' and '-terminal' of 'value' Ohms (Ω)

Example:

```
R1 (Y 0) resistor r=3K
```

This statement applies a $3\text{K}\Omega$ resistor between the Y and 0 (gnd!) terminals.

Capacitor

Format:

```
<name> (+terminal -terminal) capacitor c=value
```

Places a capacitor between nodes '+terminal' and '-terminal' of 'value' Farads (F).

Example:

```
C1 (Y 0) capacitor c=3f
```

This statement applies a 3fF capacitor between the Y and 0 (gnd!) terminals.

Inductor

Format:

```
<name> (+terminal -terminal) inductor l=value
```

Places an inductor between nodes '+terminal' and '-terminal' of 'value' Henries (H).

Example:

```
L1 (Y 0) inductor l=3u
```

This statement applies a 3uH inductor between the Y and 0 (gnd!) terminals.