Altium

Руководство по
Digital SimCode

Комплексное руководство, описывающее язык Digital SimCode, применяемый для описания математических моделей цифровых элементов.

РУКОВОДСТВО ПО DIGITAL SIMCODE

Данный документ содержит комплексное руководство, описывающее язык Digital SimCode, применяемый для описания математических моделей цифровых элементов. Руководство содержит углублённое описание для каждой функции языка.

Вследствие сложности цифровых устройств, в основном не практично имитировать их работу, используя стандартный подход с использованием SPICE без событийного моделирования. По этой причине имитатор смешанных сигналов Altium Designer включает в себя специальный описательный язык, который позволяет выполнить имитацию цифровых устройств, используя расширенную версию событийно-управляемого языка XSPICE. Этот язык, используемый для моделирования цифровых устройств, называется Digital SimCode.

SimCode является «C-подобным» описательным языком. Его используют для определения характеристик и поведения моделируемых устройств. Он использует функции для определения таких параметров, как задержки распространения, нагрузочные характеристики, силы и т.п. Поведение устройства описывают с помощью таблиц истинности, математических функций и условных операторов, таких как оператор IF... THEN.

Исходные коды на языке SimCode пишут в виде плоских ASCII файлов и сохраняют с расширением .txt.

Примечание: Digital SimCode является патентованным языком - модели устройств, созданные с его помощью, не совместимы с другими имитаторами, даже если модели цифровых компонентов, созданные с для других имитаторов, совместимы с имитатором смешанных сигналов Altium Designer.

НАЧАЛО ОПРЕДЕЛЕНИЯ SIMCODE МОДЕЛИ

Определение устройства на SimCode начинают с высказывания # xxxx source. Это высказывание имеет вид:

```
# <имя модели> source
```

где <имя модели> - имя имитирующей функции, используемой для указания блока определения модели. Блок определения модели заканчивается высказыванием EXIT. Таким образом, SimCode модель конкретного устройства будет иметь следующий вид:

```
# MyDevice source
...
...
...
EXIT;
```

В фрагменте кода выше высказывание .MODEL, содержащееся в .MDL-файле, на который ссылается условное графическое обозначение, вызовет функцию MyDevice, чтобы имитировать устройство.

Один текстовый файл с описанием на SimCode может содержать любое количество описаний моделей цифровых компонентов.

СИМВОЛ ЗАВЕРШЕНИЯ ВЫСКАЗЫВАНИЙ SIMCODE

SimCode использует символ точки с запятой «;» для указания конца высказывания SimCode.

В SimCode высказывание может занимать одну строку кода или несколько строк. Символ завершения указывает на завершение полного высказывания SimCode и расположен сразу после последнего предложения в высказывании, как показано в примерах ниже:

```
DELAY Q1 Q2 Q3 Q4 = 10n;
DELAY Q QN =
CASE (TRAN_LH) : tplh_val
CASE (TRAN_HL) : tphl_val
END;
data = (E0_1 && (CHANGED(D0) || CHANGED(D1)));
DELAY Q1 Q0 =
CASE (data && TRAN_LH) : tplh_D_Q
CASE (data && TRAN_HL) : tphl_D_Q
CASE (TRAN_LH) : tplh_E_Q
CASE (TRAN_HL) : tphl_E_Q
END;
```

КОММЕНТАРИИ В SIMCODE-ФАЙЛАХ

Вы можете включать комментарии в SimCode-файл, предваряя комментарии двумя символами правой наклонной «//». Условно эту пару символов можно назвать директивой комментирования. Всё, что находится правее директивы комментирования будет игнорировано. Комментарии часто встречаются как индивидуальная строка, или после строки SimCode, как показано в примерах ниже:

```
EVENT = (present_time + 1e-6); //return in 1us
DELAY Q QN =
// Case 1
CASE (TRAN_LH) : tplh_val
// Case 2
CASE (TRAN_HL) : tphl_val
END;
```

Комментарии должны быть расположены в моделях SimCode, чтобы сделать код более читаемым и в качестве помощи для отладки.

ОПИСАНИЕ МОДЕЛИ SIMCODE

Следующие элементы составляют язык Digital SimCode, и применяются для конструирования имитационной модели цифрового устройства. Каждый отдельный пункт темы даёт полное описание синтаксиса и примеры использования. При описании синтаксиса языка используется следующий стиль:

<> - значение/переменная/вывод/выражение;

[] - дополнительный параметр;

{}|{} - выбор (следует выбрать один из параметров).

Функции определения устройства

Функции, используемые для определения выводов устройства и т.п.:

- INPUTS
- OUTPUTS
- INTEGERS
- REALS
- PWR_GND_PINS
- IO_PAIRS

Функции установки устройства

Функции, используемые для задания определённых характеристик для выводов устройства:

- VIL_VIH_VALUE
- VIL_VIH_PERCENT
- VOL_VOH_MIN

Функции тестирования устройства

Функции, используемые для тестирования любых нарушений в устройстве, которые могут произойти в схеме:

- SUPPLY_MIN_MAX
- RECOVER
- SETUP_HOLD
- WIDTH
- FREQUENCY(FMAX)

Функции выходных выводов

Функции, используемые для программирования выходных выводов устройства:

- STATE
- STATE_BIT
- LEVEL
- STRENGTH
- TABLE
- EXT_TABLE
- LOAD
- DRIVE
- DELAY
- NO_CHANGE
- EVENT

Функции и операторы выражений

Используются в выражениях для управления данными и выполнения сравнений, которые контролируют ход исполнения программы:

- OPERATORS
- MATH FUNCTIONS
- PARAM_SET
- PWL_TABLE
- SELECT_VALUE
- MIN_TYP_MAX
- NUMBER
- VALUE
- CHANGE_TIME
- WIDTH_TIME
- INSTANCE
- CHANGED_XX
- READ_DATA

Управление программой

Функции, используемые для управления ходом исполнения программы:

- # XXXX SOURCE
- IF ... THEN
- WHILE ... DO
- GOTO
- GOSUB
- RETURN
- EXIT

Текстовый вывод

Функции, используемые для отображения сообщений в ходе имитации и отладки модели:

- PROMPT
- MESSAGE

Отладка

Функции, используемые для трассировки в ходе выполнения задач отладки SimCode:

- STEP_ON
- STEP_OFF

xxxx source

Синтаксис:

 # <имя функции> source

Описание:

Определяет начало исходной функции SimCode. Этот оператор идентифицирует функцию SimCode, чтобы ее можно было вызвать из имитатора смешанных сигналов Altium Designer. Это должно быть первое высказывание каждой функции устройства Digital SimCode.

Параметры:
<имя функции> - Имя функции устройства SimCode.

Примеры:

```
//================================
# MyDevice source
//================================
INPUTS VCC, GND, IN1, IN2;
OUTPUTS VCC_LD, IN1_LD, IN2_LD, OUT;
EXIT;
```

Примечания:
Имитатор имеет возможность читать либо нескомпилированные исходные модели SimCode, либо компилировать модели SimCode. Ключевое слово source идентифицирует это как исходную модель SimCode для имитатора, которая автоматически компилирует модель на лету, когда выполняется имитация.

Если вы хотите, чтобы скомпилированная информация о модели была записана в файл, убедитесь, что параметр Create compiled SimCode output file включён в диалоговом окне Simulation Preferences. Доступ к этому диалогу осуществляется нажатием кнопки Preferences в диалоговом окне Analyses Setup (Design -> Simulate -> Mixed Sim). Скомпилированный вывод записывается в текстовый файл ASCII - <имя функции> simlist.txt, который по умолчанию хранится в том же каталоге, что и сама исходная модель ASCII SimCode (\Library\Sim). Скомпилированную информацию о модели можно извлечь, чтобы создать скомпилированный файл модели SimCode (*.SCB).

CHANGE_TIME

Синтаксис:

```
CHANGE_TIME(<вывод>)
```

Описание:
Функция возвращает действительное значение, которое показывает последнее время изменения состояния указанного входного или выходного вывода.

Параметры:
<вывод> - Имя входного или выходного вывода.

Примеры:

```
T1 = (CHANGE_TIME(INA));
```

CHANGED_XX

Синтаксис:

```
CHANGED_xx(<вывод> [{<}|{<=}|{>}|{>=} <переменная/время/значение>])
```

Описание:
Эта функция используется, чтобы определить, изменился ли указанный вывод.

Суффикс _XX, который следует за ключевым словом CHANGED, может быть исключён (для указания любого типа изменения) или XX может быть установлен в:
LH, LX, HL, HX, XL, XH, LZ, ZL, ZH, ZX, HZ или XZ
для указания конкретного типа изменения. Для проверки более конкретного изменения будет включён необязательный оператор сравнения (<, <=,>,> =) и <переменная/ время / значение>. Если они не включены, функция вернёт 1, если вывод изменился на текущем шаге имитации.

Параметры:

<выход>	-	Имя входного или выходного вывода.
<переменная/время/значение>	-	Объект, с которым сравнён <вывод>.

Примеры:

IF (CHANGED_LH(CLK)) THEN ...
IF (CHANGED(DATA < 10n)) THEN ...

DELAY

Синтаксис 1:

DELAY <выход> [<выход> ...] = <задержка>;

Синтаксис 2:

DELAY <выход> [<выход> ...] =
CASE (<условное выражение>) : <задержка>
CASE (<условное выражение>) : <задержка>
[CASE (<условное выражение>) : <задержка> ...]
END;

Описание:

Устанавливает задержку распространения на определённые выходы. Команда DELAY выполняется один раз для каждого перечисленного вывода и назначает задержку распространения для каждого вывода, который изменил свой уровень. Дополнение CASE позволяет указать более одного параметра <задержка>. Затем <условное выражение> определяет, какая <задержка> будет использоваться. Если для вывода, который не изменился, установлена задержка, то вывод будет помечен как NO-CHANGE, и задержка не будет отправлена. <Задержка> может быть действительной константой, действительной переменной или действительным выражением.

Параметры:

<выход>	-	Имя или переменный индекс выходного вывода.
<условное выражение>	-	Условное выражение, которое определяет задержку, которая будет использована.
<задержка>	-	Время задержки распространения до выходного вывода.

Примеры:

DELAY Q1 Q2 Q3 Q4 = 10n;
DELAY Q QN =
CASE (TRAN_LH) : tplh_val
CASE (TRAN_HL) : tphl_val
END;
data = (E0_1 && (CHANGED(D0) || CHANGED(D1)));
DELAY Q1 Q0 =
CASE (data && TRAN_LH) : tplh_D_Q
CASE (data && TRAN_HL) : tphl_D_Q
CASE (TRAN_LH) : tplh_E_Q
CASE (TRAN_HL) : tphl_E_Q
END;

В этом примере, если данные отличны от нуля, а Q1 меняется с High на Low, задержка tphl_D_Q будет назначена Q1. Затем, если Q0 меняется с Low на High, задержка tplh_D_Q будет назначена Q0.

Примечания:
Команда DELAY должна выполняться ровно один раз для каждого выходного вывода, то есть для каждого вывода, объявленного в операторе OUTPUTS, который НЕ указан в высказываниях LOAD или NO_CHANGE. Порядок, в котором установлены задержки, основан на порядке, в котором эти контакты указаны в команде DELAY (то есть для первого перечисленного вывода, задержка устанавливается первой). Каждое <условное выражение> оценивается в том порядке, в котором оно перечислено в команде DELAY, пока одно из выражений не будет оценено как TRUE. Когда это происходит, значение <задержка>, связанное с выражением TRUE, назначается для установки выводу. При использовании опции CASE по крайней мере одно <условное выражение> должно оцениваться как TRUE для каждого выходного вывода. Если ни одно <условное выражение> не имеет значения TRUE, назначается <задержка>, связанная с последним оператором CASE.

В дополнение к стандартным функциям выражений следующие термы могут быть применены только к назначенному выходному выводу и могут быть использованы в <условном выражении>:

- TRAN_LH – переход L->H;
- TRAN_LX – переход L->X, где X – другое состояние;
- TRAN_HL – переход H->L;
- TRAN_HX – переход H->X, где X – другое состояние;
- TRAN_HZ – переход H->Z;
- TRAN_XL – переход X->L, где X – другое состояние;
- TRAN_XH – переход X->H, где X – другое состояние;
- TRAN_LZ – переход L->Z;
- TRAN_ZL – переход Z->L;
- TRAN_ZH – переход Z->H;
- TRAN_ZX – переход Z->X, где X – другое состояние;
- TRAN_XZ – переход X->Z;
- TRAN_XX – переход X->Y, где Y – другое состояние, не X.

Если значение <задержки> меньше или равно 0.0, будет отображаться сообщение об ошибке времени выполнения. Выходные выводы могут быть определены с помощью имени выходного вывода или целочисленной переменной, содержащей индекс выходного вывода. Названия выводов и переменные не могут быть смешаны в одном высказывании DELAY. Ссылки на выходы должны быть либо все именами выводов, либо все именами переменных.

DRIVE

Синтаксис:
DRIVE <выход> [<выход> ...] = (v0=<значение> v1=<значение> ttlh=<значение> tthl=<значение>);

Описание:
Объявляет выходные характеристики выходных выводов. Команда DRIVE используется для объявления выходных характеристик выходного вывода. Когда выход установлен в состояние LOW, выходной вывод подключается к значению напряжения v0 через сопротивление rol_param. Когда выход установлен в состояние HIGH, выходной вывод подключается к значению напряжения v1 через сопротивление roh_param. Время перехода от низкого к высокому задаётся с помощью ttlh, а время перехода от максимума к минимуму устанавливается с помощью tthl.

Параметры:

<выход>	-	Имя или переменный индекс выходного вывода.
v0=<значение>	-	низкий уровень (VOL) напряжения питания выходного вывода. <Значение> может быть действительным значением или переменной.
v1=<значение>	-	высокий уровень (VOH) напряжения питания выходного вывода. <Значение> может быть действительным значением или переменной.
tthl=<значение>	-	Время перехода 1->0 выходного вывода. <Значение> может быть действительным значением или переменной.

Примеры:

rol_param = (MIN_TYP_MAX(drv_param: 62.5, 43.75, NULL);
roh_param = (MIN_TYP_MAX(drv_param: 262.5, NULL, 52.5);
DRIVE Q QN = (v0=vol_param,v1=voh_param,ttlh=ttlh_val,
tthl=tthl_val);

Примечания:

Имена и переменные выводов нельзя смешивать в одном заявлении DRIVE. Ссылки на выходы должны быть либо все именами выводов, либо все именами переменных.

Значения, используемые для rol_param, должны быть получены с использованием спецификаций для VOL. Это значение представляет собой полное сопротивление насыщения подтягивающей структуры выхода устройства. Для стандартного LS выхода в состоянии LOW, например, падение напряжения при нагружении 8 мА не будет превышать 0,5 В, обычно ближе к 0,35 В. Следовательно:

для типичного LOW состояния :
rol_param = VOLtyp / IOLmax
rol_param = 0.35В / 8мА
rol_param = 43.75 Ом.

для минимального LOW состояния:
rol_param = VOLmax / IOLmax
rol_param = 0.5В / 8мА
rol_param = 62.5 Ом.

Значения, используемые для roh_param должны быть получены, используя спецификацию для IOS, если это значение доступно. Это значение представляет собой полное сопротивление насыщения подтягивающей структуры выхода устройства. Для стандартного LS выхода в состоянии HIGH при закорачивании вывода на минус питания и питании Vcc=5.25В будет доступно не менее 20мА, но не более 100 мА. Следовательно:

для минимального HIGH состояния:
roh_param = VCCmax / IOSmin
roh_param = 5.25В / 20мА
roh_param = 262.5 Ом.

для максимального HIGH состояния:
roh_param = VCCmax / IOSmax
roh_param = 5.25В / 100мА
roh_param = 52.5 Ом.

EVENT

Синтаксис:

EVENT = ({<время>}|{<выражение>});

Описание:

Сообщает имитатору цифровое событие. Во многих случаях цифровое событие сообщается, когда изменяется состояние одного или более входных выводов модели SimCode. Когда событие обрабатывается, вызывается и запускается SimCode для указанного события.

Эта инструкция позволяет модели SimCode сообщить цифровое событие в назначенное <время>. Если указанное время EVENT больше, чем время имитатора (его можно узнать из внутренней переменной present_time), тогда цифровое событие будет сообщено. Если более одного события сообщено в одном вызове SimCode модели, то будет использована только инструкция EVENT с более продолжительным значением <время>. Эта функция позволяет имитировать генераторы одиночных импульсов и другие подобные модели устройств.

Параметры:

<время>	-	Время, в которое должно возникнуть событие.
<выражение>	-	Выражение, показывающее время, в которое должно возникнуть событие.

Примеры:

EVENT = (present_time + 1e-6); //возврат в 1мкс, событие будет сообщено через 1 мкс после текущего вызова.

Примечания:

Если цифровое событие возникает для определённой модели до того, как инструкция EVENT <время> сообщена SimCode, инструкция EVENT <время> должна быть сообщена повторно. Например, если текущее время имитации равно 1мкс, SimCode модель устанавливает EVENT=2мкс и входной вывод в модели SimCode изменяет состояние в 1,5 мкс, тогда событие на время 2мкс должно быть сообщено заново.

EXIT

Синтаксис:

EXIT;

Описание:

Завершает исполнение SimCode

Примечания:

Это последняя строка SimCode модели, но инструкция также может быть расположена в других местах для прерывания исполнения оставшегося SimCode.

EXT_TABLE

Синтаксис:

EXT_TABLE <строка>
<входной вывод> [<состояние входа> ...] <выходной вывод> [<выходной вывод> ...]
<входной вывод> [<состояние входа> ...]
<выходное состояние> [<выходное состояние> ...];

Описание:

Устанавливает логические состояния на основе расширенной таблицы истинности. Высказывание EXT_TABLE является функцией расширенной таблицы истинности, используемой для установления уровня и силы указанных выходных выводов.

Разрешёнными входными состояниями являются:

0 - низкий (входное напряжение <= vil_param)
1 - высокий (входное напряжение >= vih_param)
^ - переход от низкого к высокому
v - переход от высокого к низкому
X - неопределённое состояние

Разрешёнными выходными состояниями являются:

L - ZERO (установить выходной уровень в vol_param)
H - HIGH (установить выходной уровень в voh_param)
Z - UNKNOWN (установить выходной уровень в v3s_param)

Также для имён входных и выходных выводов допускается применить дополнительные префиксы для указания выходных выводов. Префиксы следующие:

- - Предыдущее состояние
~ - Инверсное состояние
-~ - Инверсное предыдущее состояние

Буквы выходных состояний могут заканчиваться двоеточием и буквой, указывая силу:

s - STRONG (установить выход в rol_param для L and roh_param для H)
z - HI_IMPEDANCE (установить выход в r3s_param)

Если буква силы не указана после состояния выхода, тогда будет использовано значение STRONG для L и H состояний и значение HI_IMPEDANCE будет использовано для Z состояния.

Параметры:

<строка>	-	Переменная, в которую помещается номер строки, используемый в таблице
<input pin>	-	Имя входного вывода
<output pin>	-	Имя выходного вывода
<input state>	-	Состояния отдельных входов
<output state>	-	Состояния отдельных выходов на основе входных условий

Примеры:

EXT_TABLE tblIndex
PRE CLR CLK DATA Q QN
0 1 X X H L
1 0 X X L H
0 0 X X H H
1 1 ^ X DATA ~DATA
1 1 X X Q ~Q;

Этот пример является показательным для 1/2 триггера D-типа 7474. Если входные выводы PRE, CLR, и DATA имеют высокое состояние (>=vih_param) и CLK совершает переход L->H, Q примет значение H (voh_param) и STRONG (roh_param), Qn примет состояние L (vol_param) и STRONG (rol_param) и tblindex будет иметь значение 4.

Примечания:

Каждая строка проверяется последовательно сверху-вниз пока не будут достигнуты заданные условия для входов. Выход устанавливается в значение первой встретившейся строки, удовлетворяющей входным условиям. <Строка> устанавливается в значение номера строки таблицы, которая была использована. Если соответствий не было найдено, тогда <строке> устанавливается значение 0. Имена выводов, использованные для указания выходных состояний, не нуждаются в нахождении в шапке таблицы. В отличии от высказывания TABLE входные переменные не допускаются.

FREQUENCY (FMAX)

Синтаксис:

FREQUENCY(<вход> [<вход>...] MIN=<частота> MAX=<частота> [«<сообщение>»])

Описание:

Проверяет входы на нарушение минимальной и максимальной частот. Функция FREQUENCY сравнивает период <входа> (время от одного перехода L->H до следующего перехода L->H) с величиной, обратной указанной <частоте> (1/freq). Если длительность периода для <входа> меньше, чем величина, обратная к указанной MAX частоте, или длительность периода для <входа> больше, чем, величина, обратная к MIN частоте, тогда будет выведено <сообщение>.

Параметры:

<вход>	-	Имя или индекс переменной контролируемого входного вывода.
MIN=<частота>	-	Минимальная допускаемая частота для контролируемого вывода.
MAX=<частота>	-	Максимальная допускаемая частота для контролируемого вывода.
<сообщение>	-	Текстовая строка, которая будет выведена при возникновении нарушения условий.

Примеры:

FREQUENCY(CLK MAX=10MEG «CLK»); //проверяется только fmax

Примечания:

Для данной функции должны быть использованы данные спецификации имитируемого компонента. Выводы и имена переменных могут быть смешаны в одном высказывании FREQUENCY. Сообщается только о первом сбое в высказывании FREQUENCY для каждого перечисленного вывода.

GOSUB

Синтаксис:

GOSUB <метка>;

Описание:

Переход к подпрограмме в SimCode. Инструкция GOSUB используется для выполнения непоследовательного исполнения в SimCode. Однако, в отличие от высказывания GOTO, исполнение SimCode будет продолжаться от инструкции, следующей за инструкцией GOSUB, когда исполнение встретит инструкцию RETURN.

Параметры:

<метка> - Положение в SimCode, где возобновится исполнение программного потока.

Примеры:

```
GOSUB Shift_Left;
.
.
Exit;
Shift_Left:
.
.
RETURN:
```

GOTO

Синтаксис:

GOTO <метка>;

Описание:

Переход к новому положению в SimCode. Инструкция GOTO используется для непоследовательного исполнения SimCode.

Параметры:

<метка> - Положение в SimCode, где возобновится исполнение программного потока.

Примеры:

```
GOTO Shutdown;
.
.
Shutdown:
.
.
Exit;
```

Примечания:

Исполнение программного потока возобновляется в положении в SimCode, в котором встречается <метка>: . <Метка> должна начинаться с буквы латинского алфавита, за которой следует любое количество буквенно-цифровых символов или символ подчёркивания (_). В том месте, где <метка> встречается в коде, она должна завершаться символом двоеточия (:).

IF ... THEN

Синтаксис 1:

```
IF (<выражение>) THEN BEGIN ... [ELSE ...] END;
```

Синтаксис 2:

```
IF (<выражение>) THEN GOTO <метка>;
```

Описание:

Условно контролирует программный поток SimCode.

Высказывание IF ... THEN используется для управления потоком программы в зависимости от того, оценивается ли выражение <expression> true или false. Несколько IF ... THEN могут быть вложенными.

Параметры:

<выражение>	-	Любое выражение, которое может быть оценено как истина или ложь.
<метка>	-	Расположение в SimCode, где возобновляется поток программы.

Примеры:

```
IF (EN) THEN
BEGIN
STATE Q0 = UNKNOWN;
ELSE
IF (IN2) THEN
BEGIN
STATE Y2 = ONE;
ELSE
STATE Y2 = ZERO;
END;
END;
IF (x = -2) THEN GOTO Do_Neg2;
...
...
Do_Neg2:
...
...
```

Примечания:
Когда используется форма BEGIN ... ELSE ... END этого выражения, а <выражение> принимает значение истина, поток программы возобновляется из инструкции BEGIN и пропускает любой необязательный SimCode между высказываниями ELSE и END. Если <выражение> принимает значение ложь, то поток программы возобновляется из необязательного оператора ELSE, если он существует или после оператора END, если он не существует.

Когда используется форма GOTO этого оператора, а <выражение> - истина, поток программы возобновляется с места, где <метка>: появляется в SimCode. <Метка> должна начинаться с буквенного символа, за которым следует любое количество буквенно-цифровых символов или символ подчёркивания (_). Если в коде появляется <метка>, за ней должно сразу следовать двоеточие (:).

INPUTS

Синтаксис:

INPUTS <входной вывод>[, <входной вывод>, ...];

Описание:
Объявляет входные выводы (выводы, которые контролируют схему). Тип данных INPUTS используется для определения выводов, которые контролируют сигналы, внешние по отношению к устройству. Обычно они включают в себя вход, вход/выход, питание и заземляющий выводы.

Параметры:
<входной вывод> - Имя входа

Примеры:

INPUTS VCC, GND, PRE, DATA, CLK, CLR;

Примечания:
Идентификаторы входных выводов должны начинаться с буквы и определяться до их использования.

INSTANCE

Синтаксис:

INSTANCE(«<имя экземпляра>»)

Описание:
Проверяет, является ли это указанным экземпляром устройства. Функция INSTANCE возвращает 1, если данный экземпляр устройства SimCode соответствует указанному <имя экземпляра>. В противном случае он возвращает 0.

Параметры:
<имя экземпляра> - Текстовая строка, показывающая имя экземпляра.

Примеры:

```
IF (INSTANCE(«AU23»)) THEN
BEGIN
MESSAGE(«U23-Q0 = %d», Q0);
END;
```

Примечания:
Схема может содержать более одного устройства. Во время моделирования может быть важно знать, является ли имитируемое устройство в данный момент тем, что вас интересует. Это позволит вам, например, печатать сообщения для одного конкретного элемента NAND без необходимости выводить сообщения для всех других NAND. Имя экземпляра - это обозначение устройства (Designator), которому предшествует символ префикса SPICE (буква A).

INTEGERS

Синтаксис:

INTEGERS <переменная>[, <переменная>, ...];

Описание:

Тип данных INTEGERS используется для объявления целочисленных переменных и массивов.

Параметры:

<переменная> - Имя переменной.

Примеры:

INTEGERS tblIndex, count, data[64];

Примечания:

Целочисленные переменные и массивы должны начинаться с буквы и определяться до их использования. Целочисленные массивы определяются следующим за именем массива левой скобкой ([), целым числом, которое определяет размер массива и правой скобкой (]). Целочисленные массивы могут быть установлены и/или использованы в выражениях.

Ниже перечислены целые переменные SimCode, которые не нужно объявлять:

Переменная	Использование
tp_param	Индекс tplh/hl
tt_param	Индекс ttlh/hl
ld_param	Индекс LOAD
drv_param	Индекс DRIVE
i_param	Индекс ICC
user_param	Индекс USER
warn_param	Предупреждающие сообщения
init_sim	Единожды в ходе инициализации SimCode
tran_pin	Индекс вывода в TRAN_xx

Ожидается, что первые шесть переменных в этом списке будут иметь значение 1, 2 или 3. Эти значения представляют собой индекс в массивах min/typ/max:

Значение	Смысл
1	индекс минимального значения
2	индекс типичного значения
3	индекс максимального значения

Параметр цифровой модели может быть установлен независимо для каждого цифрового устройства, используя соответствующие параметры, найденные на вкладке Parameters диалогового окна Sim Model. Доступ к этому диалогу осуществляется двойным щелчком по записи для ссылки имитационной модели в области Models на панели Properties.

Если дополнительный параметр SPICE задан на странице Advanced Options диалогового окна Analyses Setup, этот параметр будет глобально переопределять значения параметров цифровой модели для всех цифровых устройств. Если переменная задана явно в SimCode, этот параметр отменяет все остальные настройки. Таким образом, области размещения параметров можно расположить по убыванию приоритета в следующем порядке: SimCode->Advanced Options->Model parameters.

Параметр warn_param может быть настроен на любое положительное значение, чтобы условно отображать предупреждающие сообщения для устройства. Программист устройства может создавать различные уровни предупреждений, доступ к которым осуществляется путём ввода разных положительных значений. Значение init_sim устанавливается 1 во время инициализации SimCode, в противном случае оно равно 0. Значение tran_pin устанавливается на индекс вывода, который устанавливается во время высказывания DELAY CASE. Этот индекс используется для определения того, к какому выводу применяется инструкция TRAN_xx.

IO_PAIRS

Синтаксис:

IO_PAIRS (<входной вывод:выходной вывод>[, <входной вывод:выходной вывод>, ...]);

Описание:

Объявляет ассоциации контактов входа / выхода для вывода нагрузки. Оператор IO_PAIRS определяет, какие из выводов INPUTS связаны с каким из выводов OUTPUTS. Эта ассоциация используется оператором LOAD.

Параметры:

<входной вывод :выходной вывод> - Имена ассоциируемых входов и выходов

Примеры:

IO_PAIRS (IN1:IN1_LD, IN2:IN2_LD);

В этом примере IN1 и IN2 имеют тип INPUTS, а IN1_LD и IN2_LD - тип OUTPUTS. IN1 и IN1_LD оба соответствуют одному и тому же физическому выводу устройства.

Примечания:

Каждый физический входной вывод устройства состоит из входного вывода и выходного вывода в SimCode. Для определения входных нагрузочных характеристик требуется выходной вывод. Это высказывание можно использовать только один раз в SimCode. Выводы питания не указаны в высказывании IO_PAIRS.

LEVEL

Синтаксис 1:

LEVEL <выход> [<выход> ...] = (<выражение>);

Синтаксис 2:

LEVEL <выход> [<выход> ...] = {ZERO}|{ONE}|{UNKNOWN};

Описание:

Устанавливает уровень выходного состояния. Состояние выходного вывода определяется его уровнем и силой. Следует использовать команду LEVEL для установки уровня одного или нескольких выходных выводов.

<выражение>	Состояние	Уровень
<= vol_param	ZERO	vol_param
>= voh_param	ONE	voh_param
другое	UNKNOWN	v3s_param

Параметры:

<выход> - Имя или переменный индекс выхода.

<выражение> - Любое выражение сравнивается с VOL или VOH.

Примеры:

LEVEL Q = ONE;
LEVEL Q1 Q2 Q3 Q4 = ZERO;
LEVEL OUT = ((1+2)/3);

В последнем примере OUT будет следующий результат:
ZERO если vol_param > 1,
UNKNOWN если vol_param < 1 и voh_param > 1,
ONE если voh_param < 1.

Примечания:

Выходные выводы могут быть указаны, используя имя выходного вывода или целочисленной переменной, которая содержит индекс выходного вывода.

Выводы и имена переменных не могут быть смешаны в одном высказывании LEVEL. Ссылки на выходы должны быть либо все именами выводов, либо все именами переменных.

LOAD

Синтаксис:

LOAD <выход> [<выход> ...] = (v0=<значение> r0=<значение> [v1=<значение> r1=<значение>] [io=<значение>] t=<значение>);

Описание:

Объявляет нагрузочные характеристики входных выводов. Команда LOAD обычно используется с входными или силовыми выводами для обеспечения нагрузки для управляющей устройством схемы. Поскольку только выходные выводы могут обеспечивать нагрузку, каждый вход должен иметь соответствующий выход. Они назначаются с помощью оператора IO_PAIRS.

Если для разных входов требуются разные нагрузки, могут использоваться несколько операторов LOAD. Силовые контакты должны быть размещены в отдельной инструкции LOAD, которая не включает нагрузку v1 / r1 или io. Выводы питания не включены в оператор IO_PAIRS. Высказывание IO_PAIRS должно быть введено перед любыми высказываниями LOAD, которые содержат io.

Параметры

<выход>	-	Имя переменной, содержащей индекс выходного вывода.
v0	-	Напряжение нагрузки для входа в состоянии HIGH. <значение> может быть действительной константой или действительной переменной.
r0	-	Сопротивление нагрузки для входа в состоянии HIGH. <значение> может быть действительной константой или действительной переменной.
v1	-	Напряжение нагрузки для входа в состоянии LOW. <значение> может быть действительной константой или действительной переменной.
r1	-	Сопротивление нагрузки для входа в состоянии LOW. <значение> может быть действительной константой или действительной переменной.
io	-	Неактивное сопротивление нагрузки для неиспользуемой нагрузки. <значение> должна быть действительной постоянной.
t	-	Задержка времени, которое будет применено перед загрузкой. <значение> должна быть действительной постоянной.

Примеры:

r0_val = (MIN_TYP_MAX(ld_param: NULL, NULL, 125k);
r1_val = (MIN_TYP_MAX(ld_param: NULL, NULL, 10.5k);
ricc_val = (MIN_TYP_MAX(ld_param: NULL, 833, 525);
LOAD PRE_LD DATA_LD CLK_LD CLR_LD = (v0=vol_param, r0=r0_val,
v1=voh_param, r1=r1_val, io=1e9, t=1p);
LOAD VCC_LD = (v0=gnd_param, r0=ricc_val, t=1p);

Примечания:
Входная нагрузка состоит из напряжения и сопротивления (v0 / r0 или v1 / r1). Уровень напряжения входящего сигнала определяет, какая нагрузка будет использоваться. Если уровень напряжения переходит ниже VIL и остаётся ниже VIH, тогда вход считается в состоянии LOW и применяется v1 / r1. Если уровень напряжения переходит выше VIH и остаётся выше VIL, тогда вход считается в состоянии HIGH и применяется v0 / r0. io - сопротивление входа в выключенном состоянии. Неиспользуемая нагрузка по существу удаляется из схемы, изменяя её значение r на значение, указанное для io.

Значения для v0, r0, v1 и r1 могут быть либо действительными константами, либо действительными переменными. Значения для io и t должны быть действительными константами. Имена выводов и переменные выводов не могут быть смешаны в одном и том же высказывании LOAD. Ссылки на выходы должны быть либо все именами выводов, либо все именами переменных.

Для входных выводов значения, используемые для r0, должны быть получены с использованием данных для IIH (Iinput_high) из спецификаций. Например, стандартный вход LS будет поглощать максимум 20 мкА при Vin = 2,7 В. Поэтому, если vol_param = 0.2 В, то:

для максимальной нагрузки HIGH:
r0 = (Vin - vol_param) / IIHmax
r0 = (2,7 В - 0,2 В) / 20 мкА
r0 = 125 кОм

Значения, используемые для r1, должны быть получены с использованием спецификаций данных для IIL (Iinput_low). Например, стандартный вход LS будет выдавать максимум 400 мкА при Vin = 0,4 В. Поэтому, если voh_param = 4.6 В, то:

для максимальной несущей нагрузки:
r1 = (voh_param - Vin) / IILmax
r1 = (4.6 В - 0.4 В) / 400 мкА
r1 = 10,5 кОм

Для силовых контактов значение, используемое для r0, должно быть выведено с использованием спецификаций данных для ICC (Icc). Для 74LS151 тип Icc равен 6 мА при Vcc = 5 В, а Icc max - 10 мА при Vcc = 5.25 В. Следовательно:

для Icc typ : r0 = 5 В / 6 мА = 833 Ом
для Icc max : r0 = 5,25 В / 10 мА = 525 Ом

Если вы создаёте устройство с несколькими частями на элемент, такое как четырёхвентильный И-НЕ элемент 74LS00, вы должны соответственно настроить нагрузку Icc для отдельных частей.

Математические функции

В SimCode модели могут быть использованы следующие математические функции:

Function	Description	Example
POW	Возведение в степень	X= (12 POW(3));
ABS	Округление значения	X= (ABS(-12));
SQRT	Квадратный корень	X= (SQRT(2));
EXP	Экспонента	X= (EXP(10));
LOG	Натуральный логарифм	X= (LOG(0.1));
LOG10	Логарифм по основанию 10	X= (LOG10(0.1));
SIN	Синус	X= (SIN(0.1));
COS	Косинус	X= (COS(0.1));
TAN	Тангенс	X= (TAN(0.1));
ASIN	Арксинус	X= (ASIN(0.1));
ACOS	Арккосинус	X= (ACOS(0.1));
ATAN	Арктангенс	X= (ATAN(0.1));
HSIN	Гиперболический синус	X= (HSIN(0.1));
HCOS	Гиперболический косинус	X= (HCOS(0.1));
HTAN	Гиперболический тангенс	X= (HTAN(0.1));

MESSAGE

Синтаксис:

MESSAGE(«<сообщение>»[, <значение/вывод>...]);

Описание:

Отображает сообщение без остановки исполнения программного потока. Высказывание MESSAGE используют для вывода информации, указанной в строке <сообщение>. Это работает без прерывания имитации. Сообщение отображается в окне статуса в ходе имитации.

Параметры:

<сообщение>	-	Строка сообщения, в том числе символы форматирования.
<значение>	-	Переменная или константа.
<вывод>	-	Имя вывода или индекс вывода.

Примеры:

MESSAGE(«device instance= %s»,INSTANCE);

Примечания:

Строка формата в MESSAGE похожа на формат, который может использоваться в инструкции printf в языке C. Допустимые символы форматирования включают (но не ограничиваются ими):

\t	-	табуляция
\n	-	переход на новую строку
\r	-	возврат каретки в начало строки
%d	-	Отображение в десятичной системе исчисления для короткой переменной или текущего состояния входа / выхода.
%D	-	Отображение в десятичной системе исчисления для короткой переменной или предыдущего состояния входа / выхода.
%x	-	Отображение в шестнадцатеричной системе исчисления для короткой переменной или текущего состояния входа / выхода.
%X	-	Отображение в шестнадцатеричной системе исчисления для короткой переменной или предыдущего состояния входа / выхода.
%c	-	Отображение в символьном виде для короткого переменного или текущего состояния ввода / вывода.
%C	-	Отображение в символьном виде для короткого переменного или предыдущего состояния ввода / вывода.
%e	-	Отображение в экспоненциальном виде для действительной переменной.
%f	-	Отображение в инженерном виде с плавающей запятой для действительной переменной.
%g	-	Отображение в укороченном виде (%e или %f) для действительной переменной.
%s	-	Отображение строковой постоянной.

Допустимыми строковыми постоянными являются:

INSTANCE	-	Настоящее имя экземпляра устройства SimCode
FUNC	-	Настоящее имя функции устройства SimCode
FILE	-	Настоящее имя файла устройства SimCode

MIN_TYP_MAX

Синтаксис:

MIN_TYP_MAX(<индекс>: <минимум>, <типично>, <максимум>);

Описание:

Возвращает значение из таблицы поиска MIN_TYP_MAX. Функция MIN_TYP_MAX аналогична функции SELECT_VALUE, за исключением того, что необходимо ввести три значения / переменные. Ключевое слово NULL может быть заменено вместо одного или двух неизвестных значений. Если в качестве индекса <индекс> используется предопределённая целочисленная переменная (см. INTEGERS), неизвестные (NULL) значения рассчитываются из известных значений следующим образом:

Известное значение	Формула
<минимум>, <максимум>	типично = (<максимум> + <минимум>) / 2
Только <минимум>	типично = (<минимум> / <масштабный множитель минимума>)
	максимум = (<минимум> / <масштабный множитель минимума>) * <масштабный множитель максимума>
только <типично>	минимум = (<типично> * <масштабный множитель минимума>)
	максимум = (<типично> * <масштабный множитель максимума>)
только <максимум>	минимум = (<максимум> / <масштабный множитель максимума>) * <масштабный множитель минимума>
	типично = (<максимум> / <масштабный множитель максимума>)

Параметры:

<индекс>	-	Входная переменная (индекс выбора минимального (1), типичного (2) или максимального (3) значения).
<минимум>	-	Минимальное значение из спецификации.
<типично>	-	Типичное значение из спецификации.
<максимум>	-	Максимальное значение из спецификации.

Примеры:

tplh_val = (MIN_TYP_MAX(tp_param: NULL, 5n, NULL));

В этом примере, если предположим, что SPICE настройки PROPMNS и PROPMXS установлены в значения по-умолчанию (см. Примечания), тогда:
if tp_param = 1 (т. е. вернуть минимальное значение), тогда tplh_val = 2.5 нс
if tp_param = 2 (т. е. вернуть типичное значение), тогда tplh_val = 5 нс
if tp_param = 3 (т. е. вернуть максимальное значение), тогда tplh_val = 7.5 нс

ricch_val = (MIN_TYP_MAX(i_param: NULL, 2500, 1250));

В этом примере, если мы предположим, что значения SPICE настроек CURRENTMNS и CURRENTMXS установлены в значения по умолчанию (см. Примечания), тогда:
if i_param = 1 (т. е. вернуть минимальное значение), тогда ricch_val = 3750
if i_param = 2 (т. е. вернуть типичное значение), тогда ricch_val = 2500
if i_param = 3 (т. е. вернуть максимальное значение), тогда ricch_val = 1250

Примечания:

Если <индекс> имеет ни одно из предопределённых переменных, перечисленных ниже, тогда <масштабный множитель минимума>=0.5 и <масштабный множитель максимума>=1.5.

<Масштабный множитель минимума> и <масштабный множитель максимума> для каждой из этих предустановленных переменных можно изменить на странице Advanced Options диалога Analyses Setup (Design -> Simulate -> Mixed Sim).
<Масштабный множитель минимума> и <масштабный множитель максимума> обратны для ld_param, drv_param и i_param, т.к. эти параметры управляют значением сопротивления, а не значением тока (т.е. максимальная нагрузка приравнивается к минимальному сопротивлению).

Переменная	SPICE параметр	Значение параметра по умолчанию
tp_param	PROPMNS	<масштабный множитель минимума> = 0.5
	PROPMXS	<масштабный множитель максимума> = 1.5
tt_param	TRANMNS	<масштабный множитель минимума> = 0.5
	TRANMXS	<масштабный множитель максимума> = 1.5
ld_param	LOADMNS	<масштабный множитель минимума> = 1.5
	LOADMXS	<масштабный множитель максимума> = 0.5
drv_param	DRIVEMNS	<масштабный множитель минимума> = 1.5
	DRIVEMXS	<масштабный множитель максимума> = 0.5
i_param	CURRENTMNS	<масштабный множитель минимума> = 1.5
	CURRENTMXS	<масштабный множитель максимума> = 0.5
vth_param	VTHMNS	<масштабный множитель минимума> = 0.5
	VTHMXS	<масштабный множитель максимума> = 1.5
user_param	USERMNS	<масштабный множитель минимума> = 0.5
	USERMXS	<масштабный множитель максимума> = 1.5

NO_CHANGE

Синтаксис:

NO_CHANGE <выход> [<выход> ...];

Описание:

Функция NO_CHANGE используется, чтобы указать отсутствие изменений для указанных выходных выводов. Высказывание используется для двунаправленных выводов, когда двунаправленный вывод обрабатывается как вход.

Параметры:

<выход> - Имя или индекс переменной выходного вывода.

Примеры:

NO_CHANGE Q1 Q2 Q3 Q4;

Примечания:

Имена выводов и переменные нельзя смешивать в одном и том же высказывании NO_CHANGE. Ссылки на выходы должны быть либо все именами выводов, либо все именами переменных.

NUMBER

Синтаксис:

NUMBER(<MSB вывод>, [<вывод>, ...] <LSB вывод>);

Описание:

Возвращает число, основанное на двоичных взвешенных состояниях выводов. Функция NUMBER возвращает короткое целое число, которое представляет десятичное значение двоичного числа, представленного списком <выводов>. Каждый бит (представленный <выводом>) установлен в 1, если <вывод> не равен нулю, в противном случае он установлен в 0.

Параметры:
<вывод> - Имя или индекс вывода.

Примеры:

A = (NUMBER(D3,D2,D1,D0));

В этом примере, если D3=HIGH, а D2, D1 и D0 = LOW (1000), тогда A=8

Примечания:
Первый <вывод> в перечне представляет старший разряд (MSB), а последний <вывод> в перечне представляет младший разряд (LSB).

ОПЕРАТОРЫ

В выражениях SimCode могут быть использованы следующие операторы:

=	Равенство, присвоение значения.
+	Сложение
-	Вычитание
*	Умножение
/	Деление
~	Логическое отрицание
!	Поразрядное дополнение
&&	И (AND)
||	ИЛИ (OR)
^^	ИСКЛЮЧАЮЩЕЕ ИЛИ (XOR)

Поразрядные операторы:

&	AND
|	OR
^	XOR
<<	Сдвиг влево
>>	Сдвиг вправо

Операторы сравнения:

=	Равно
!=	Не равно
<	Меньше
<=	Меньше или равно
>	Больше
>=	Больше или равно

Описание:
Операторы используются для установки и управления переменными и выражениями.

Примеры:

```
clk_twl = (25n);
reg = (reg + 1);
vx = (vol_param - 10m);
C = (A * B);
val = (xval / 2);
X = (A && ~(B));
Y = (!(X)); //если X=1 тогда Y=FFFFFFFE
A = (X & 1); //если X=1 тогда A=1, если X=2 тогда A=0
B = (X | 8); //если X=1 тогда B=9, если X=2 тогда B=10
C = (X >> 2); //если X=1 тогда C=0, если X=2 тогда C=0
D = (2 >> X); //если X=1 тогда D=1, если X=2 тогда D=0
E = (X << 2); //если X=1 тогда E=4, если X=2 тогда E=8
F = (2 << X); //если X=1 тогда F=4, если X=2 тогда F=8
IF (A >= B) THEN ...
IF ((A < 2) && (B > 3)) THEN ...
IF ((C < 2) || (X > 4)) THEN ...
```

Примечания:

Выражения должны быть заключены в круглые скобки (). Выражения всегда оцениваются слева направо в круглых скобках. Требуется использовать круглые скобки для определения приоритета операторов внутри выражения. При использовании унарных операторов (логическое NOT и побитовое дополнение) значения, переменные, выражения, значения, переменные, выражения и т.д. должны быть в круглых скобках ().

OUTPUTS

Синтаксис:

OUTPUTS <выходной вывод>[, <выходной вывод>, ...];

Описание:

Объявляет выходные выводы (выводы, которые управляют или нагружают схему). Тип данных OUTPUTS используется для определения выводов, которые влияют на работу схем, внешних по отношению к устройству. Обычно они включают в себя вход, выход, ввод-вывод и силовые контакты. Входные и силовые контакты включены в этот список, поскольку их присутствие представляет нагрузку на управляющую схему.

Параметры:

<выходной вывод> - Имя выходного вывода.

Примеры:

```
OUTPUTS VCC_LD, PRE_LD, DATA_LD, CLK_LD, CLR_LD, QN, Q;
```

Примечания:

Идентификаторы выходных выводов должны начинаться с буквы и определяться до их использования.

PARAM_SET

Синтаксис:

PARAM_SET(<имя параметра>)

Описание:

Функция PARAM_SET позволяет установить, было ли определено значение заданного параметра в определении модели SimCode. Она возвращает 1, если параметр был определён (например, vil_param=0.8), иначе она возвращает значение 0.

Параметры:
<имя параметра> - параметр определения модели SimCode.

Примеры:

A = PARAM_SET(ld_param);
IF (PARAM_SET(voh_param)) THEN ...

Примечания:
Перечень описания параметров определения модели SimCode и имена их ассоциированных переменных (имена параметров) см. в INTEREGRS и REALS.

PROMPT

Синтаксис:

PROMPT(«<сообщение>»[, <значение/вывод>...]);

Описание:
Оператор PROMPT используется для приостановки моделирования и отображения информации, указанной строкой <сообщения>. Сообщение отображается в окне состояния во время моделирования. Пользователь должен нажать кнопку, чтобы продолжить выполнение SimCode.

Параметры:

<сообщение>	-	Строка сообщения, содержащая символы форматирования при необходимости.
<значение>	-	Значение переменной или константы.
<вывод>	-	Имя вывода или индекс вывода.

Примеры:

PROMPT(«input=%d time=%f device=%s», D1, t1, INSTANCE);

Примечания:
Строка формата в PROMPT похожа на формат, который может использоваться в инструкции printf в языке C. Допустимые символы форматирования включают (но не ограничиваются ими):

\t	-	табуляция
\n	-	переход на новую строку
\r	-	возврат каретки в начало строки
%d	-	Отображение в десятичной системе исчисления для короткой переменной или текущего состояния входа / выхода.
%D	-	Отображение в десятичной системе исчисления для короткой переменной или предыдущего состояния входа / выхода.
%x	-	Отображение в шестнадцатеричной системе исчисления для короткой переменной или текущего состояния входа / выхода.
%X	-	Отображение в шестнадцатеричной системе исчисления для короткой переменной или предыдущего состояния входа / выхода.
%c	-	Отображение в символьном виде для короткого переменного или текущего состояния ввода / вывода.
%C	-	Отображение в символьном виде для короткого переменного или предыдущего состояния ввода / вывода.
%e	-	Отображение в экспоненциальном виде для действительной переменной.
%f	-	Отображение в инженерном виде с плавающей запятой для действительной переменной.
%g	-	Отображение в укороченном виде (%e или %f) для действительной переменной.
%s	-	Отображение строковой постоянной.

Допустимыми строковыми постоянными являются:

INSTANCE	-	Настоящее имя экземпляра устройства SimCode
FUNC	-	Настоящее имя функции устройства SimCode
FILE	-	Настоящее имя файла устройства SimCode

PWL_TABLE

Синтаксис:

PWL_TABLE(<входная переменная>: <IN1>, <OUT1>, <IN2>, <OUT2> [,...<INn>, <OUTn>])

Описание:

Эта функция является кусочно-линейной функцией, по существу таблицей соответствия. Значение <входной переменной> используется для поиска записи в таблице, состоящей из пар значений. Первое значение каждой пары является значением сравнения для входной переменной, а второе значение является соответствующим выходным значением. Если значение <входной переменной> меньше, чем первое значение <IN>, то возвращается первое <OUT> значение. Если значение <входной переменной> больше, последнее <INn> значение, тогда возвращается последнее <OUTn> значение. Между промежуточными записями таблицы выполняется линейная интерполяция по следующей формуле:

value = (((OUTA-OUTB)/(INA-INB))*(<IN var>-INA)+OUTA),

где <входная переменная> лежит между входными сравниваемыми значениями INA и INB. Актуальное значение выхода будет лежать между соответствующими значениями OUTA и OUTB.

Параметры:

<входная переменная>	-	входная переменная (целочисленная или действительная)
<INx>	-	значение для сравнения входа
<OUTx>	-	выходное значение для <INx>

Примеры:

twh = (PWL_TABLE(var: 5,180n,10,120n,15,80n));

В этом примере, если var = 10, тогда twh = 120n, а если var = 12, тогда twh = 104.

Примечания:

Должны быть введены две или более пары значений данных IN / OUT, а значения IN должны быть введены в порядке возрастания. Максимальное количество пар данных IN / OUT не ограничено.

PWR_GND_PINS

Синтаксис:

PWR_GND_PINS (<вход питания положительной полярности>, <вход питания отрицательной полярности>);

Описание:

Оператор PWR_GND_PINS определяет, какие из входных выводов являются питанием и землёй, и устанавливает параметры питания и земли устройства на абсолютные напряжения следующим образом:

pwr_param = напряжение на <входе питания положительной полярности>

gnd_param = напряжение на <входе питания отрицательной полярности>

Параметры:

<вход питания положительной полярности>	-	имя вывода питания положительной полярности.
<вход питания отрицательной полярности>	-	имя вывода питания отрицательной полярности.

Примеры:

PWR_GND_PINS(VCC, GND);

Примечания:

Это высказывание может быть использовано в SimCode только единожды. Только один вывод может быть определён как вывод питания положительной полярности и один как вывод питания отрицательной полярности.

READ_DATA

Синтаксис:

READ_DATA(<массив>[, <массив>, ...])

Описание:

Функция READ_DATA открывает файл, указанный параметром «data=» в высказывании .MODEL (шаблон SPICE для определения типового цифрового устройства в *.mdl файле) и читает ASCII текстовые данные в один или более массивов. Количество и тип значения (целый/действительный) в строке, которое будет прочитано, основано на количестве и типе переменных массива, указанных в вызове функции. Количество прочитанных строк данных определяется по количеству строк данных в указанном файле и/или размером наименьшего массива в вызове функции.

Функция READ_DATA возвращает количество прочитанных строк. В случае ошибки возвращается отрицательное число:

-1	Неверное имя файла
-2	Не удаётся найти файл
-3	Недопустимый массив
-4	Недопустимый доступ к массиву
-5	Тип данных
-6	Ожидается значение данных

Параметры:

<массив> - Имя массива, в которое помещено значение.

Примеры:

MYDEVICE.MDL file:
.MODEL AMYDEVICE XSIMCODE(file=«{MODEL_PATH}MYDEVICES.SCB»
+ func=MyDevice data=«{MODEL_PATH}MYDEVICE.DAT» {mntymx})
MYDEVICE.DAT file:
8, 8E-6
9, 9E-6
10, 1E-5
11, 1.1E-5
MyDevice SimCode:
nlines = READ_DATA(int_array, real_array);

В этом примере открывается файл с названием MYDEVICE.DAT. Читается 2 столбца данных из файла, где первый столбец содержит целочисленные значения, а второй столбец содержит действительные значения. Если массив объявлен как int_array[3] и real_array[5], тогда будут прочитаны только первые 3 строки данных и nlines примет значение 3.

Примечания:

Несколько значений в строке в файле данных должны быть разделены запятыми.

Действительные числа должны быть записаны в файле в научной нотации.

Высказывание .MODEL, которое содержит параметр «data=» должно быть расположено в *.mdl файле, который подключается УГО с помощью шаблона.

REALS

Синтаксис:

REALS <переменная>[, <переменная>, ...];

Описание:

Тип данных REAL применяется для объявления действительных переменных и массивов.

Параметры:

<переменная> - имя переменной

Примеры:

REALS tplh_val, tphl_val, ricc_val, vbias, values[64];

Примечания:

Действительные переменные и массивы должны начинаться с буквы и быть определены до их использования. Действительные массивы определяются следующим именем массива с левой скобкой ([), целым числом, которое определяет размер массива и правой скобкой (]). Действительные массивы могут быть определены и / или использованы в выражениях.

Ниже перечислены действительные переменные SimCode, которые не нужно объявлять:

Переменная	Значение	Имя параметра цифровой модели
vil_param	состояние низкого уровня входа	VIL
vih_param	состояние высокого уровня входа	VIH
vol_param	состояние низкого уровня	VOL
voh_param	состояние высокого уровня входа	VOH
v3s_param	третье состояние входа	N/A
rol_param	значение низкого уровня силы выхода	N/A
roh_param	значение высокого уровня силы выхода	N/A
r3s_param	третье состояние уровня силы выхода	N/A
pwr_param	напряжение на выводе питания положительной полярности	PWR
gnd_param	напряжение на выводе питания отрицательной полярности	GND
present_time	текущее время имитации	N/A
previous_time	предыдущее время имитации	N/A
sim_temp	рабочая температура схемы	N/A (SPICE Option: TEMP)

Параметр цифровой модели может быть установлен независимо для каждого цифрового устройства, используя соответствующие параметры, найденные на вкладке Parameters диалогового окна Sim Model. Доступ к этому диалогу осуществляется двойным щелчком по записи для ссылки имитационной модели в области Models панели Properties. Ввод значения для любого из PWR, GND, VIL, VIH, VOL и VOH в диалоговом окне Sim Model переопределит любое значение, указанное в модели SimCode.

Значения pwr_param и gnd_param устанавливаются каждый раз при выполнении инструкции PWR_GND_PINS. Значение current_time и previous_time устанавливаются каждый раз, когда изменяется временной шаг. Значение sim_temp - это текущая рабочая температура схемы, которая может быть установлена параметром TEMP имитатора SPICE.

RECOVER

Синтаксис:

RECOVER(<clk вход> = {LH}|{HL}
<mr вход> [<mr вход> ...]
{TREC=<время>}|{TRECL=<время> TRECH=<время>} [«<сообщение>»];

Описание:

Тестирует входы на нарушения времени восстановления. Функция RECOVER сравнивает разницу во времени между изменением уровня (LH или HL) на входе <clk вход> и изменением уровня на входе <mr вход> с указанным временем тестирования. Время тестирования RECOVER указывается объединённо с использованием TREC = <время> (которое устанавливает TRECL и TRECH на одно и то же значение) или индивидуально с использованием TRECL = <время> и TRECH = <время>. Если время сравнения меньше заданного <времени>, во время имитирования будет отображаться WARNING. Необязательная строка <сообщение> может быть включена в высказывание RECOVER, и будет выводиться, если отображается WARNING.

Параметры:

<clk вход>	-	Имя или индекс входного тестируемого вывода тактирования/опоры.
<mr вход>	-	Имя или индекс входного тестируемого вывода установки/сброса.
TREC	-	Время восстановления как для низких, так и для высоких переходов <mr вывода>.
TRECL	-	Время восстановления для низких переходов <mr вывод>.
TRECH	-	Время восстановления для высоких переходов <mr вывод>.
<время>	-	Указанное проверочное время.
<сообщение>	-	Текстовая строка, которая будет показана в случая возникновения предупреждения.

Примеры:

RECOVER(CLK=LH PRE CLR TREC=trec_val
«CLK->PRE or CLR»);

Примечания:

С этой функцией следует использовать дынные из спецификации. TRECL = <время> и TRECH = <время> могут быть введены в одни и те же тесты RECOVER. Тест RECOVER будет выполнен только в том случае, если состояние <mr входа> соответствует параметру времени (TRECL = LOW, TRECH = HIGH), когда <clk вход> выполняет указанный переход (LH или HL). Например, если <clk вход> = LH и TRECL определён, то <mr вход> должен быть LOW, когда <clk вход> переходит из LOW в HIGH, для того, чтобы выполнить тест RECOVER.

Названия и переменные выводов могут быть смешаны в одном высказывании RECOVER.

RETURN

Синтаксис:

RETURN

Описание:

Возвращает из подпрограммы в SimCode. Инструкция RETURN используется для возврата потока программы к инструкции, которая следовала за последней инструкцией GOSUB.

ELECT_VALUE

Синтаксис:

SELECT_VALUE (<индекс>: <значение/вывод/переменная>, <значение/вывод/переменная> [,<значение/вывод/переменная>,...]);

Описание:

Возвращает значение из простой таблицы сопоставления. Функция SELECT_VALUE возвращает значение числа или переменной, обозначенных значением индекса.

Параметры:

<индекс>	-	входная переменная (индекс для <значение/вывода/переменной>).
<значение/вывод/переменная>	-	выходное значение, вывод или переменная.

Примеры:

A = (SELECT_VALUE(B: 16, 8, 4, 2, 1));

В этом примере, если B=2, тогда A=8 (второе значение).

Примечания:

Количество используемых значений и / или переменных не ограничено.

SETUP_HOLD

Синтаксис:

SETUP_HOLD(<clk вход> = {LH}|{HL}
<data вход> [<data вход> ...]
{TS=<время>}|{TSL=<время> TSH=<время>}
{TH=<время>}|{THL=<время> THH=<время>} [«<сообщение>»];

Описание:

Тестирует входы на нарушения установки и удержания. Функция SETUP_HOLD сравнивает разницу во времени между изменением уровня (LH или HL) на <clk входе> и уровнем изменения на <data входе> с заданным временем тестирования. Время тестирования SETUP указывается объединённо с использованием TS = <время> (которое устанавливает TSL и TSH на одно и то же значение) или индивидуально с использованием TSL = <время> и TSH = <время>. Время тестирования HOLD указывается объединённо с использованием TH = <время> (которое устанавливает THL и THH на одно и то же значение) или индивидуально с использованием THL = <время> и THH = <время>. Если время сравнения меньше заданного <времени>, будет отображаться WARNING. Необязательная строка <сообщение> может быть включена в инструкцию SETUP_HOLD, и будет выводиться, если отображается WARNING.

Параметры:

<clk вход>	-	Имя или индекс тестируемого входного вывода тактирования/опоры.
<data вход>	-	Имя или индекс тестируемого входного вывода данных.
TS	-	Время установления для обоих высокого и низкого переходов <data входа>.
TSL	-	Время установления для низкого перехода <data входа>.
TSH	-	Время установления для высокого перехода <data входа>.
TH	-	Время удержания для обоих высокого и низкого переходов <data входа>.
THL	-	Время удержания для низкого перехода <data входа>.
THH	-	Время удержания для высокого перехода <data входа>.
<время>	-	Указанное тестирующее время.
<сообщение>	-	Текстовая строка, которая будет отображена, если возникнет предупреждение.

Примеры:

SETUP_HOLD(CLK=LH DATA Ts=ts_val Th=th_val «CLK->DATA»);

Примечания:

С этой функцией следует использовать данные спецификации. TSL = <время>, TSH = <время>, THL = <время> и THH = <время> могут быть введены в одном и том же высказывании SETUP_HOLD. Тест SETUP и / или HOLD будут выполняться только в том случае, если состояние <data входа> соответствует параметру времени (TSL или THL = LOW, TSH или THH = HIGH), когда <clk ввод> выполняет указанный переход (LH или HL). Например, если указан <clk вход> = LH и TSL, то <data вход> должен быть LOW, когда <clk вход> переходит из LOW в HIGH для теста SETUP.

Названия выводов и переменные могут быть смешаны в одном и том же заявлении SETUP_HOLD.

STATE

Синтаксис 1:

STATE <выход> [<выход>...] = (<выражение>);

Синтаксис 2:

STATE <выход> [<выход>...] = {ZERO}|{ONE}|{UNKNOWN};

Описание:

Состояние выходного вывода определяется его уровнем и силой. Команда STATE устанавливает уровень и силу для одного или нескольких выходных выводов или переменных. Если <выражение> меньше или равно vol_param, выход будет установлен на ZERO. Если <выражение> больше или равно voh_param, выход будет установлен в ONE. В противном случае на выходе будет установлено значение UNKNOWN. Значения уровня и силы задаются в соответствии с состоянием:

<выражение>	Состояние	Уровень	Сила
<= vol_param	ZERO	vol_param	rol_param
>= voh_param	ONE	voh_param	roh_param
другое	UNKNOWN	v3s_param	r3s_param

Параметры:

<выход>	-	Имя или переменный индекс входного вывода.
<выражение>	-	Любое выражение, сравниваемое с VOL или VOH.

Примеры:

STATE Q = ONE;
STATE Q1 Q2 Q3 Q4 = ZERO;
STATE OUT = ((1+2)/3);

В последнем примере OUT будет:
ZERO, если vol_param > 1
UNKNOWN, если vol_param < 1 и voh_param > 1
ONE, если voh_param < 1

Примечания:

Выходные выводы могут быть заданы с помощью имени выходного вывода или целочисленной переменной, содержащей индекс выходного вывода. Имена выводов и переменных нельзя смешивать в одном высказывании STATE. Ссылки на выходы должны быть либо все именами выводов, либо все именами переменных.

STATE_BIT

Синтаксис:

STATE_BIT <вывод> [<вывод> ...] = (<выражение>);

Описание:

Состояние выходного вывода определяется его уровнем и силой. Команда STATE_BIT используется для установки уровня и силы для одного или нескольких выходных выводов на основе значения <выражения>. Состояние первого перечисленного вывода задаётся в соответствии с первым (наименьшим значащим разрядом) значения выражения, состояние второго перечисленного вывода устанавливается в соответствии со вторым битом значения выражения и так далее. Значения уровня и силы задаются значением бит:

Значение разряда	Состояние	Уровень	Сила
0	ZERO	vol_param	rol_param
1	ONE	voh_param	roh_param

Параметры:

<выход>	-	Имя или переменный индекс выходного вывода.
<выражение>	-	Любое выражение, которое может быть побитно сравнено с выходами.

Примеры:

STATE_BIT Q1 Q2 Q3 Q4 = (internal_reg);

В этом выражении, если internal_reg = 11 (1011 binary), тогда Q1 (LSB) = ONE, Q2 = ONE, Q3 = ZERO and Q4 (MSB) = ONE.

Примечания:

Выходные выводы могут быть заданы с помощью имени выходного вывода или целочисленной переменной, содержащей индекс выходного вывода. Названия выводов и переменных не могут быть смешаны в одном и том же высказывании STATE_BIT. Ссылки на выходы должны быть либо все именами выводов, либо все именами переменных.

Максимальное количество выходных выводов / переменных ограничено 16.

STEP_OFF

Синтаксис:

STEP_OFF

Описание:

Отключает режим трассировки SimCode.

STEP_ON

Синтаксис:

STEP_ON

Описание:

Включает режим трассировки SimCode. Это приводит к тому, что SimCode отображает номер счётчика программы (ПК) и каждую инструкцию SimCode до её исполнения.

STRENGTH

Синтаксис 1:

STRENGTH <выход> [<выход> ...] = (<выражение>);

Синтаксис 2:

STRENGTH <выход> [<выход> ...] = {STRONG}|{HI_IMPEDANCE};

Описание:

Состояние выходного вывода определяется его уровнем и силой. Команда STRENGTH используется, чтобы установить силу одного или нескольких выходных выводов.

Значение	Состояние	Сила
STRONG	ZERO	rol_param
STRONG	ONE	roh_param
HI_IMPEDANCE	N/A	r3s_param
<выражение>	N/A	<выражение>

Параметры:

<выход>	-	Имя или переменный индекс выходного вывода.
<выражение>	-	Любое выражение, которое должно использоваться непосредственно как сила.

Примечания:

Выходные выводы могут быть заданы с использованием имени выходного вывода или целочисленной переменной, содержащей индекс выходного вывода. Имена выводов и переменных нельзя смешивать в одном и том же высказывании STATE. Ссылки на выходы должны быть либо все именами контактов, либо все именами переменных.

SUPPLY_MIN_MAX

Синтаксис:

SUPPLY_MIN_MAX(<минимальное значение>,<максимальное значение>)

Описание:

Функция SUPPLY_MIN_MAX проверяет разницу напряжений между выводами питания и земли (положительной и отрицательной полярностями питания), определёнными в PWR_GND_PINS. Если параметр WARN установлен в ON на закладке Parameters диалога SimModel и разница напряжений (pwr_param-gnd_param) меньше, чем <минимальное значение> или больше чем <максимальное значение>, то в ходе имитации будет отображено предупреждение.

Параметры:

<минимальное значение>	-	Минимальное рекомендуемое напряжение питания.
<максимальное значение>	-	Максимальное рекомендуемое напряжение питания.

Примеры:

SUPPLY_MIN_MAX(4.75, 5.25);

Примечания:

Данные для этой функции должны быть взяты из спецификации элемента. Функция PWR_GND_PINS должна быть определена для использования данной функции.

TABLE

Синтаксис:

TABLE <строка>
<вход> [<вход>...] <выходной вывод> [<выходной вывод>...]
<состояние входа> [<состояние входа>...] <состояние выхода> [<состояние выхода>...]

Описание:

Высказывание TABLE работает как таблица истинности, позволяя установить силу и вывод указанных выводов.

Допустимыми состояниями входов являются:

0 низкий уровень (входное напряжение <= vil_param).
1 высокий уровень (входное напряжение >=vih_param).
X входное напряжение не важно.

Допустимыми состояниями выходов являются:

L ZERO (установить уровень в vol_param).
H ONE (установить уровень в voh_param).
Z UNKNOWN (установить уровень в v3s_param).

Сила выхода может быть определена через двоеточие после символа выходного состояния:

s STRONG (установить выход в rol_param для L и roh_param для H)
z HI_IMPEDANCE (установить силу в r3s_param)

Если символ силы вывода не указан после состояния вывода, тогда будет использовано значение STRONG для L и H состояний и HI_IMPEDANCE для Z состояний.

Параметры:

<строка>	-	Переменная, в которую помещается номер строки таблицы.
<вход>	-	Имя входного вывода или его индекс.
<выходной вывод>	-	Имя выходного вывода.
<входное состояние>	-	Состояние отдельных входов.
<выходное состояние>	-	Состояние отдельных выходов, основанное на входных условиях.

Примеры:

TABLE tblIndex

INA	*INB*	*OUT*
0	*0*	*H*
0	*1*	*H*
1	*0*	*H*
1	*1*	*L;*

В этом примере указана таблица истинности представителя 1/4 для 7400 2-входового вентеля типа 2И-НЕ. Если входные выводы INA и INB оба в высоком состоянии (>=vih_param), тогда OUT устанавливается в ZERO (vol_param) и STRONG (rol_param), а tblIndex устанавливается в 4.

Примечания:

Каждая строка проверяется последовательно сверху-вниз до тех пор, пока не будут выполнены условия для входов. Выходы устанавливаются для первой строки для соответствия входным условиям. <Строка> указывает на строку таблицы, которая была использована. Если соответствия не было найдено, то <строка> устанавливается в 0.

Имена входов и их индесы (индексные переменные) не могут быть смешаны в одном высказывании TABLE. Сслыки на входы должны все быть именами выводов или именами переменных.

VALUE

Синтаксис:

VALUE(<вывод>)

Описание:

Возвращает значение указанного вывода. Функция VALUE возвращает действительное число, которое указывает уровень напряжения указанного вывода.

Параметры:

<вывод> - Имя или индекс вывода

Примеры:

v = (VALUE(D3));

VIL_VIH_PERCENT

Синтаксис:

VIL_VIH_PERCENT (<уровень входа низкий%>,<уровень входа высокий%>);

Описание:

VIL и VIH не используют мин/тип/макс массив для выбора их значений, но должны быть объявлены явно для каждого цифрового устройства. Высказывание VIL_VIH_PERCENT устанавливает значения параметров VIL и VIH устройства в процентах от напржения питания следующим образом:
vil_param = (pwr_param - gnd_param)*<уровень входа низкий%>
vih_param = (pwr_param - gnd_param)*<уровень входа высокий%>

Параметры:

<уровень входа низкий%>	-	Проценты от напряжения питания, определяющие низкий уровень входа.
<уровень входа высокий%>	-	Проценты напряжения питания, определяющие высокий уровень входа.

Примеры:

VIL_VIH_PERCENT(33,67)

Примечания:

Высказывание PWR_GND_PINS должно быть определено для использования данной функции.

Значения <уровень входа высокий> и <уровень входа низкий> должны лежать в дипапазоне от 0 до 100.

Значения параметров vil_param и vih_param, установленные VIL_VIH_PERCENT переопределяются любыми значениями параметров VIL и VIH, заданными на закладке Parameters диалога Sim Model.

VIL_VIH_VALUE

Синтаксис:

VIL_VIH_VALUE (<уровень входа низкий>,<уровень входа высокий>);

Описание:

VIL и VIH не используют мин/тип/макс массив для выбора их значений, но должны быть объявлены явно для каждого цифрового устройства. Высказывание VIL_VIH_VALUE устанавливает значения параметров VIL и VIH устройства в абсолютных значениях напряжения следующим образом:
vil_param = <уровень входа низкий>
vih_param = <уровень входа высокий>

Параметры:

<уровень входа низкий>	-	Абсолютное значение напряжения, которое определяет низкий уровень входа.
<уровень входа высокий>	-	Абсолютное значение напряжения, которое определяет высокий уровень входа.

Примеры:

VIL_VIH_VALUE(1.25, 1.35);

Примечания:

Чтобы более точно смоделировать фактические характеристики переключения цифрового входа, значения параметров VIL и VIH обычно не устанавливаются в значения из спецификации элемента. Исключениями являются устройства с указанным гистерезисом, такие как 74LS14. Как правило, гистерезис цифрового устройства мал, порядка 100 мВ, но никогда не 0 В.

Значения параметров vil_param и vih_param, установленные VIL_VIH_PERCENT переопределяются любыми значениями параметров VIL и VIH, заданными на закладке Parameters диалога Sim Model.

VOL_VOH_MIN

Синтаксис:

VOL_VOH_MIN (<смещение низкого уровня выхода>,<смещение высокого уровня выхода>,<минимальная разница верхнего и нижнего уровней выхода>);

Описание:

VOL и VOH не используют мин/тип/макс массив для выбора их значений, но должны быть объявлены явно для каждого цифрового устройства. Высказывание VOL_VOH_MIN устанавливает значения параметров VOL и VOHустройства следующим образом:
vol_param = gnd_param+<смещение низкого уровня выхода>
voh_param = pwr_param+<смещение высокого уровня выхода>

Параметры:

<смещение низкого уровня выхода>	-	Смещение напряжения относительно отрицательного вывода питания для получения низкого уровня выхода.
<смещение высокго уровня выхода>	-	Смещение напряжения относительно положительного вывода питания для получения высокого уровня выхода.
<минимальная разница верхнего и нижнего уровней выхода>	-	Минимальная допустимая разница между вычисленными низким и высоким уровнями напряжения выхода.

Примеры:

VOL_VOH_MIN(0.2, -0.4, 0.1);

В этом примере:
Если gnd_param = 0 В и pwr_param = 5.0 В, тогда vol_param = 0.2 В и voh_param = 4.6.
Если gnd_param = 0 В и pwr_param = 0.5 В, тогда vol_param = 0 В и voh_param=0.1 В.

Примечания:

Чтобы более точно смоделировать фактические цифрового вывода, VOH обычно не устанавливается равным значению в спецификации на элемент. Причиной этого является то, что значения в спецификации для VOH указаны для максимальной нагрузки IOH. В SimCode значения VOL и VOH заряктеризуются при отсутствии нагрузки. Для использования VOL_VOH_MIN должно быть определено PWR_GND_PINS.

Значения параметров vol_param и voh_param, установленные VOL_VOH_MIN, переопределяются значениями параметров VOL и VOH, заданными на закладке Parameters диалога Sim Model. Это смещённые значения, а не абсолютные значения.

<Смещения высокого уровня выхода> отрицательно, таким образом, когда оно добавляется к pwr_param, результирующее значение VOH не будет больше, чем pwr_param. Если разница между результирующим vol_param и voh_param менее, чем <минимальная разница верхнего и нижнего уровней выхода>, тогда vol_param будет установлено значение gnd_param, а voh_param будет установлено значение gnd_param+<минимальная разница верхнего и нижнего уровней выхода>.

WHILE ... DO

Синтаксис:

WHILE (<выражение>) DO BEGIN ... END;

Описание:

Высказывание WHILE ... DO используется для создания циклов в SimCode, пока <выражение> не будет вычислено как ложное.

Параметры:

<выражение> - Любое выражение, которое вычисляется как истина или ложь.

Примеры:

```
i = 1;
WHILE (i<=5) DO
BEGIN
data[i] = data[i+1];
i = i + 1;
END;
```

Примечания:

Поток программы остаётся между высказываниями BEGIN и END, пока <выражение> не будет вычислено как ложное, программа будет продолжена после высказывания END.

WIDTH

Синтаксис:

WIDTH(<вход> [<вход>...] {TWL=<время>}|{TWH=<время>} [«<сообщение>»]);

Описание:

Функция WIDTH сравнивает ширины импульсов каждого <входа> с указанным в WIDTH временем. Для контроля времени низкого уровня указывается TWL=<время>, а для контроля времени высокого уровня указывается TWH=<время>. Если сравниваемое время меньше, чем указанное <время>, то отображается WARNING. Дополнительная строка <сообщения> может быть включена в высказывание WIDTH, и показано во время отображения WARNING.

Параметры:

<вход>	-	Имя или индекс тестируемого входного вывода.
TWL	-	Условие ширины импульса низкого уровня.
TWH	-	Условие ширины импульса высокого уровня.
<сообщение>	-	Текстовая строка, которая будет показана, если возникнет нарушение условия.

Примеры:

```
WIDTH(CLK TWL=clk_twl TWH=clk_twl «CLK»);
WIDTH(PRE CLR TWL=pre_clr_twl «PRE or CLR»);
```

Примечания:
Для данной функции должны быть использованы данные спецификации элемента. Входные выводы могут быть именами входных выводов или целочисленными переменными, которые содержат индекс входного вывода. Имена выводов и переменные могут быть смешаны в одном высказывании WIDTH.

WIDTH_TIME

Синтаксис:

WIDTH_TIME(<вход>);

Описание:
Функция возвращает действительное значение, которое указывает на ширину последнего импульса, возникшего на указанном <входе>.

Параметры:
<вход> - Имя или индекс входного вывода.

Примеры:

PW = (WIDTH_TIME(CP2));