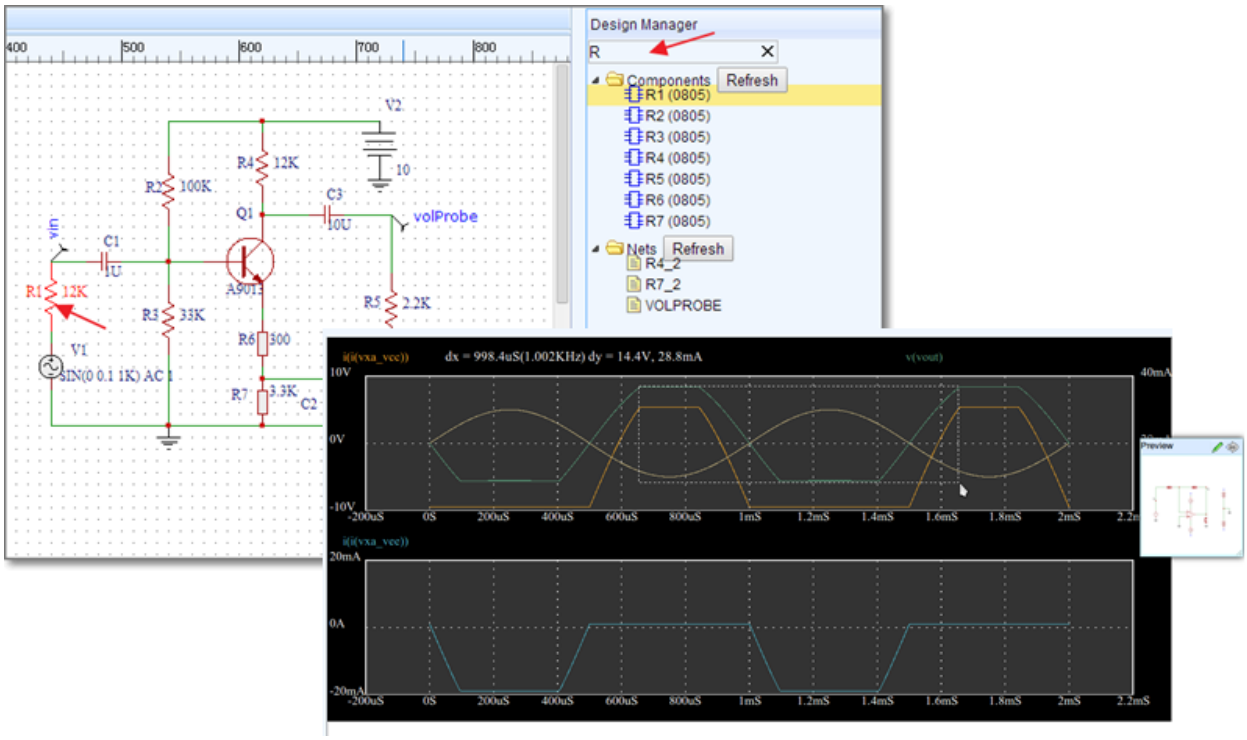
EasyEDA. Simulation eBook



Апрель 2016

Оглавление

Введение

EasyEDA – это не только возможность нарисовать схему и развести печатную плату. Это также и симулятор электрических цепей.

Симуляция схем электроники основана на специализированных математических программах, оптимизированных для конструирования, а затем решения уравнений, которые определяют поведение схемы, описываемой этими уравнениями по чертежу схемы.

Программа моделирования, которую использует EasyEDA, называется ngspice. Ngspice – это свободная и Open Source программа (FOSS), которая в свой черёд основана на симуляторе, называемом SPICE, оригинал которого написан Larry Nagel.

Для чего эта книга

Эта книга – введение в моделирование электрических схем в EasyEDA, использующей ngspice.

Она начинается с основ того, как избежать наиболее часто встречающихся ошибок, которые приводят к сбою, а затем мы перейдём к иллюстрации того, как создать схему, которая будет моделироваться успешно и даст значимые результаты. Так же в ней обсуждается ряд аспектов, объясняющих, как может возникать то, что может показаться неожиданным или даже бессмысленным результатом.

Затем книга поясняет и иллюстрирует более развитые техники, такие как:

- сигналы пробников, таких как измерители напряжения, тока, мощности и сопротивления;
- конфигурация источников сигналов;
- задание разных типов анализа;
- проведение измерений, таких как время возрастания, действующее значение и полоса пропускания;
- определение значений компонента, используя параметры;
- использование выражений для расчёта значений компонента, таких как приведённый ток в нагрузке или ёмкость для заданной частоты среза фильтра;
- использование моделей устройств от производителей;
- задание для сложной симуляции, включающей произвольные источники напряжения и тока.

Для чего книга не предназначена

- Это не книга о том, как изучать использование EasyEDA для рисования схем. Для основной информации об использовании EasyEDA, пожалуйста, обратитесь к «Руководству EasyEDA».
- Эта книга не учит электронике. Хотя могут быть примеры схем и пояснения, которые могут быть полезны для понимания электроники, но подразумевается, что пользователь уже имеет достаточные познания в области электроники, чтобы понять содержание книги.
- Хотя ngspice похожа на другие варианты spice, и большое количество информации и техник применимы к некоторым из этих вариантов, эта книга написана исключительно о моделировании электрических схем в EasyEDA, используя ngspice.
- Исключая места, где необходимо пояснить некоторые аспекты поведения симуляции, эта книга не рассматривает детали того, как работает симуляция в общем, а в ngspice в частности. За информацией по этим вопросам обратитесь к ссылкам, данным ниже:

Больше информации о Nagel и SPICE вы найдёте здесь:

http://www.omega-enterprises.net/The%20Origins%20of%20SPICE.html

Larry Nagel's PhD Dissertation:

Laurence W. Nagel., "SPICE2: A Computer Program to Simulate Semiconductor Circuits,"

Memorandum No. ERL-M520, University of California, Berkeley, May 1975.

http://www.eecs.berkeley.edu/Pubs/TechRpts/1975/ERL-520.pdf

реально очень читабельно и поучительно.

Больше информации о симуляции электронных схем, и spice в частности, вы найдёте:

http://en.wikipedia.org/wiki/Electronic_circuit_simulation

и:

http://en.wikipedia.org/wiki/SPICE

Больше информации по ngpsice доступно здесь:

http://ngspice.sourceforge.net/presentation.html

Для кого эта книга

Все инструменты симуляции и то, как они взаимодействуют с инструментами создания схем, разные, так что даже для людей, имеющих опыт в использовании инструментов симуляции, есть смысл хотя бы бегло просмотреть разделы этой книги, чтобы ознакомится с тем, где можно найти информацию о неожиданно возникающих ситуациях.

Однако для тех, у кого опыт работы с предметом ограничен или вовсе отсутствует, эта книга станет существенным чтением.

Почему?

Потому, что для любого новичка в мире моделирования электрических схем, очень заманчиво сразу начать с попыток создания и запуска моделирования множества интересных и увлекательных примеров.

Увы, это может стать очень неприятным и обескураживающим опытом, поскольку многие из этих попыток осуществить моделирование либо не будут работать, либо будут работать неправильно, либо приведут к неожиданным результатам! Это случается по ряду простых причин, но новичкам без опыта работы с симуляторами эти причины кажутся совершенно непонятными.

Так что, во избежание разочарований и даром потраченного времени, лучше потратить немного времени, чтобы познакомиться с наиболее общими причинами сбоев при симуляции или получения неожиданных ответов.

Как структурирована книга

Эта книга отнюдь не законченный том с множеством слов, диаграмм и примеров кода. Она написана как живой, интерактивный документ. Вместо чтения обширного текста с последующим переходом в EasyEDA для черчения схемы и каких-то проб, мы предлагаем живые примеры симуляции, встроенные непосредственно в текст для иллюстрации обсуждаемых моментов.

Достаточно перейти к началу или заголовку раздела, прочитать несколько параграфов, а затем открыть и запустить встроенный пример симуляции, чтобы въявь увидеть то, о чём написано в тексте.

Каждый встроенный пример моделирования должен быть сохранён перед запуском. Пользователи, которые присоединились к EasyEDA, могут сохранить эти файлы в собственной папке для проектов. Незарегистрированные пользователи могут хранить файлы в Anonymous Files. В любом случае примеры могут копироваться и редактироваться, так что, пользователи могут опробовать разные подходы, которые помогут им понять то, что пример пытается им объяснить, могут создать их персональные библиотеки для освоения примеров, поскольку те проходят через всю книгу.

Новичкам рекомендуется работать с книгой последовательно, поскольку каждый раздел построен на знаниях и идеях, полученных из всех предыдущих. Пропуск разделов оставляет «дыры», в которых могут «провалиться» начинающие, и затруднит понимание последующих разделов книги.

- Книга начинается с введения в концепции и термины, которые важны для базового осмысления того, что такое симулятор и как использовать его эффективно.
- Затем, используя простые интерактивные примеры симуляции, она поясняет и иллюстрирует, как избежать наиболее часто встречающихся подводных камней в построении, запуске и интерпретации полученных при моделировании результатов.
- Книга сосредоточена на показе путей по созданию схем, демонстрирует использование соединений и меток в схеме. Затем обсуждается техника использования пробников напряжения и тока и то, как они связаны, могут влиять или зависеть от имён (netnames), которые присваиваются соединениям либо автоматически EasyEDA, либо вручную пользователем, размещающим метки (netlabels).
- Описана более прогрессивная техника по применению пробников напряжения и тока, использующая probe-команду, чтобы с помощью этой команды измерять рассеиваемую мощность, сопротивления и проводимости.

Этот раздел описывает использование источников 0V для измерения токов, и является вводным к пониманию E,F, G и H зависимых источников, и очень мощных произвольных или B Dependent Sources (зависимых источников).

В этом разделе также поясняется концепция «Spice Directive, директив spice», описывающих, как превратить неактивный текст комментария (comment) в активный «spice» текст, который впоследствии распознаётся симулятором в качестве инструкции к действиям.

- Концепция «probe» затем расширяется описанием того, как несколько разных probe-команд могут быть добавлены к схеме – просто выбором, которая из них переключается с comment-текста на spice-текст – как они могут использоваться для обмена между разными выбранными значениями и узлами для наблюдения.

- Чтобы использовать команду «let» для дальнейшего расширения и упрощения наблюдения за сигналами, есть введение в виде простого примера. Эта команда будет шире представлена в последующих разделах, где рассказано о том, как производить измерения на основе результатов моделирования.

- Предыдущие интерактивные примеры симуляции поясняют основной способ установок и запуска симуляции, используя иконку с зелёным бегущим человечком «Simulate...» (the Green Man) через опцию «Run the Document...», открывающую диалог «Run your simulation, запустить вашу симуляцию», где вы выбираете тип анализа, который следует запустить.

- Последующие примеры показывают, как использовать горячие клавиши **CTRL+R** в качестве быстрого доступа к диалогу «Run your simulation».

Эта идея затем расширяется концепцией использования «spice directives, директив spice» для запуска симуляции прямо горячими клавишами **CTRL+R** с помощью нескольких простых ключевых приёмов; это облегчает переключение между несколькими разными симуляциями из одной моделируемой схемы.

- DC operating point, DC Transfer function, DC Sweep, AC (частотная область) и Transient (временная область) анализы описаны более детально.

Использование DC Sweep (развёртка на постоянном токе) для развёртки значений резисторов в схеме и развёртка температуры окружающей среды для схемы рассматриваются в этом разделе.

Есть детальный раздел по настройке каждого из разных источников сигналов во временной области, которые доступны, таких как SINE, PULSE и т.д.

Рассматривается конфигурирование и использование во временной области AC source (источник переменного напряжения), который встроен в каждый независимый источник сигналов.

- Продемонстрирована необходимость и разные способы задания начальных условий, таких как напряжение на соединении и переходном конденсаторе и токи в индуктивности.

Использование PULSE, EXP, PWL и B Sources для схем типа «начните» также проиллюстрировано в этом разделе.

- Описана концепция использования параметров (то есть, переменных, которые используются при симуляции) для определения таких вещей, как значения многокомпонентных элементов и установки источников сигналов.

- Использование параметров в выражениях (то есть, в формулах или уравнениях) для вычисления значений, таких как ёмкость для заданной временной постоянной RC цепи с заданным сопротивлением, не только рассматривается, но и расширяется до использования их в B Sources.

- Концепция предопределённых функций и их использование в выражениях описана детально, включая расширение к созданию определённых пользователем функций с помощью инструкции .func.

- Концепция и рамки моделей устройств для симуляции обсуждается детально.
- Детально рассматриваются пути добавления моделей и подсхем, сделанных сторонними организациями, в создаваемые схемы, чтобы использовать их с предопределёнными символами схем из EasyEDA Libs.
- Использование моделей сторонних организаций расширено до добавления их к пользовательским символам схем.
- Использование выражений и функций при создании пользовательских моделей поведения рассматриваются со ссылкой на некоторые модели от EasyEDA (EE suffix).

Вводные понятия Spice симуляции

Для каждой схемы, подлежащей симуляции, EasyEDA преобразует схему в текстовое описание этой схемы, которое передаётся в симулятор.

Текстовое описание схемы называется **spice netlist.**

- Заметьте, что spice netlist не то же самое, что генерируется из схемы, и что передаётся в PCB layout (разводка печатной платы).

Netlist – это список всех использованных компонентов, того, как они соединяются вместе, и их описание, называемое **models** (модели), так что, симулятор знает поведение каждого компонента, который должен моделироваться. Netlist также включает инструкции симулятору, которые называются **spice directives** (директивы spice), чтобы поведать ему, какой тип **analysis** (анализа) должен быть выполнен. Также может быть включён список значений, называемых **parameters**, которые используются для непосредственного определения значений компонентов или как часть более обширных вычислений, называемых **expressions** (выражения).

Подобно реальным компонентам компоненты схемы для симуляции требуют источников питания и часто источников сигналов. Поэтому netlist будет включать любые **voltage sources** и **current sources** (источники напряжения и тока). Это могут быть **Independent Sources** (независимые источники) или **Dependent Sources** (зависимые источники). Независимые источники генерируют только DC voltages или currents (постоянное напряжение или ток), или могут генерировать сигналы заданной амплитуды и фазы во временной области для симуляции (фактически они могут конфигурироваться для генерации всех трёх, но это будет рассмотрено позже). Зависимые источники могут быть также использованы для генерации DC напряжений или токов, но их основное использование в генерации выхода, что оказывается не чем иным, как **functions** этих других сигналов в схеме, например, использование функции для описания выхода источника как тока, который эквивалентен сумме квадратов двух входных напряжений.

Netlists большинства расширенных симуляций может включать инструкции, которые называются **measure statements** (выражения измерения), позволяющие симулятору выполнить вычисления по результатам симуляции, например, измерение времени нарастания импульса, RMS значение сигнала или полосу пропускания схемы фильтра по уровню 3dB.

О соглашении по именам

Перед тем как продолжить, очень важно понять соглашение по именам, используемое в этой книге и всех симуляциях.

Поля в строке отделяются одним или более пробелов, запятыми, знаками равенства (=), левой или правой скобками; дополнительные пробелы игнорируются. Строка может быть продолжена вводом знака «+» (плюс) в колонке 1 следующей строки; ngspice продолжает чтение с начала колонки 2. Имя поля должно начинаться с буквы (от A до Z) и не может содержать какие-либо разделители. Числовое поле может быть целым (12, -44), числом с плавающей точкой (3.14159), либо целым или с плавающей точкой числом с последующей целой экспонентой (1e-14, 2.65e3), либо целым или с плавающей точкой числом с последующим одним из масштабирующих коэффициентов:

Суффикс	Имя	Коэффициент
T	Tera	1e12
G	Giga	1e9
Meg	Mega	1e6
K	Kilo	1e3
mil	Mil	25.4×1e-6
m	milli	1e-3
u	micro	1e-6
n	nano	1e-9
p	pico	1e-12
f	femto	1e-15

Буквы, сразу следующие за числом, но не коэффициенты масштабирования, игнорируются, и буквы, следующие за коэффициентами, также игнорируются. Так что, 10, 10V, 10Volts и 10Hz – все представляют одинаковое число, а M, MA, MSec и MMhos – все представляют одинаковый коэффициент масштабирования. Заметьте, что 1000, 1000.0, 1000Hz, 1e3, 1.0e3, 1kHz и 1k – все представляют одинаковые числа. Заметьте, что M или m означают «милли», то есть, 1e-3.

- Суффикс Meg ДОЛЖЕН использоваться как 1e6.

Имена узлов не чувствительны к регистру.

Имена узлов могут быть либо просто числами, либо произвольными строками символов, не начинающимися с цифры.

Общий провод (узел земли) может быть назван «0» (ноль). Из соображений совместимости «gnd» допускается для обозначения узла земли, и будет внутренне рассматриваться, как глобальный узел, и преобразован в «0».

Отметьте разницу с ngspice, где узлы трактуются как строки символов, а не рассматриваются как числа, так что, «0» и «00» разные узлы в ngspice (в то время как это не так в SPICE2).

Ngspice, а следовательно EasyEDA, требует, чтобы удовлетворялись следующие топологические ограничения:

- Каждая схема должна иметь узел земли (gnd или 0)!
- Схема не должна иметь петель источников напряжения и/или индуктивностей и не должна содержать последовательно соединённых источников тока и/или конденсаторов.
- Каждый узел в схеме должен иметь путь по постоянному току к земле.
- Каждый узел должен иметь хотя бы два подключения, исключая узлы линии передачи (для линий передачи с открытым концом) и MOSFET узлы подложки (которые имеют два внутренних подключения в любом случае).

Эти ограничения в деталях будут рассмотрены позже.

Введение в использование симулятора

Использовать симулятор – не то же самое, что собрать реальную схему. Есть множество вещей, которые могут подловить новичка при симуляции, поскольку ни реальные, ни большинство компонентов симулятора не идеальны. Отклонения от идеальных этих двух воплощений компонентов часто различаются, и если различия между реальными компонентами и их представлением для симуляции не вполне осознаются, результаты могут приводить к путанице, когда результаты моделирования даже простой схемы очень отличаются от ожидаемых.

Часть проблем в том, что ожидания пользователей формируются под воздействием опыта реальных измерений. Поэтому важно точно понять, как проводились измерения, и как симуляция схемы представляет результаты. В тех случаях, когда есть разница между реальными и полученными при моделировании результатами, может потребоваться более глубокое понимание того, что происходит в данной схеме, и какие предположения делаются – часто не осознано – об этом.

Изучение использования симулятора означает больше размышлений о том, на что на самом деле похож реальный мир; чем он отличается от проблем теоретического мира учебников и простого схематического представления электрических цепей, и, поэтому, каковы могут быть результаты измерений.

Как избежать общих ошибок

Следующий раздел описывает и иллюстрирует некоторые из наиболее общих, часто встречающихся ошибок и недопонимания, которые приводят к путанице.

Prefix conflict error (ошибка конфликта префиксов)

Это сообщение об ошибке появится, если:

Есть две копии одной и той же схемы в той же папке проекта: CctA и Copy_of_CctA.

В этом случае префиксы компонентов (позиционные обозначения) будут одинаковы в обоих случаях: CctA и Copy_of_CctA.

Или:

Есть две разных схемы в той же папке проекта, CctA и CctB, но есть ряд компонентов в CctB, которые имеют те же префиксы, что и в CctA.

Тогда:

Если симуляция запущена с помощью:

Green Man > Run the Project

тогда EasyEDA попытается соединить все схемы проекта в одну схему, и затем запустить симуляцию получившейся большой схемы.

Произойдёт сбой, поскольку EasyEDA будет повторять образцы компонентов с одинаковыми префиксами.

Иначе:

если вы запускаете симуляцию с помощью:

Green Man > Run the Document

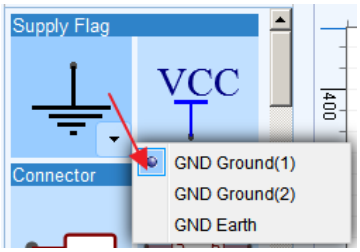
или:

CTRL+R

тогда EasyEDA запустит симуляцию только схемы на текущей активной закладке в окне редактора (если у вас открыта только одна схема, тогда она будет использована для моделирования, но если у вас более одной открытой схемы, тогда будет симулироваться только одна на выбранной закладке).

Если вы дали уникальные префиксы для каждого компонента на выделенном листе, тогда симуляция будет работать, поскольку EasyEDA найдёт только один образец каждого компонента в схеме.

Все схемы для симуляции ДОЛЖНЫ иметь узел земли



Особенность метода работы симуляторов в том, что они ДОЛЖНЫ иметь узел заземления (также может быть ссылка на 0 net) где-нибудь в схеме. Все пробники напряжения в схеме, исключая *явные* пробники разностного напряжения, должны проводить измерения относительно этого узла заземления (ground node). Этот узел может быть помещён в любом месте схемы, которое подходит для целей моделирования, но он обязательно должен быть в схеме.

СИМУЛЯЦИЯ СХЕМЫ, У КОТОРОЙ НЕТ ЗАЗЕМЛЕНИЯ, НЕ БУДЕТ РАБОТАТЬ.

Это иллюстрируют следующие два примера:

Без символа заземления симуляция не запустится:

All simulation schematics MUST have a ground 01

Как только любой доступный символ заземления будет добавлен, симуляция заработает:

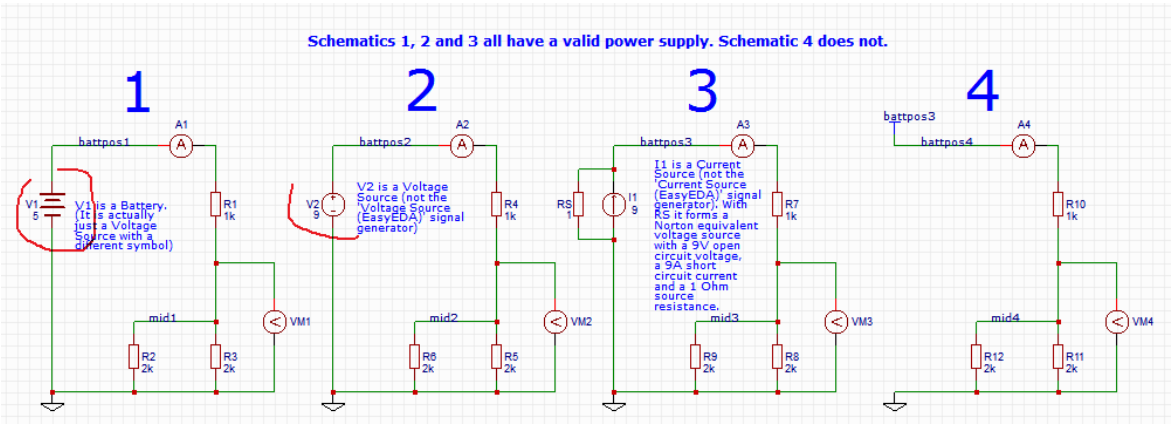
All simulation schematics MUST have a ground 02

Все схемы для симуляции ДОЛЖНЫ иметь питание и/или источник сигнала

Как реальная схема, схема для симуляции должна иметь питающее устройство какого-то рода, даже если источник питания окажется источником сигнала (например, детекторный приёмник или регулятор громкости) или ранее заряженным до какого-то напряжения конденсатором, или ранее заряженной индуктивностью до какого-то тока. Схема для симуляции должна иметь либо источник питания и/или источник сигнала, либо заданные начальные условия, такие как предварительно заряженный до какого-то напряжения конденсатор или индуктивность, заряженная до какого-то тока, перед запуском симуляции.

Источником питания может быть идеальный, с нулевым внутренним сопротивлением источник напряжения. Просто, но большинство реальных моделей батареек и источников регулируемого напряжения могут быть заменены источником напряжения (voltage sources) с последовательно включённым сопротивлением, или источником тока (current sources) с параллельным сопротивлением (то есть, Thevenin или Norton эквивалентным источником).

EasyEDA предлагает большой выбор источников сигналов (signal sources). Они будут рассмотрены в соответствующем разделе, но базовое использование некоторых из этих источников, обеспечивающих питание схемы, иллюстрируется ниже:



All simulation schematics MUST have a power supply too

Каждая точка в схеме для симуляции ДОЛЖНА иметь DC путь к заземлению

Подобно реальным схемам каждая точка симулируемой схемы ДОЛЖНА иметь путь к земле по постоянному току (или к 0 net). Попытка запустить моделирование с узлом, который не имеет пути по постоянному току к земле, вызовет сбой с сообщением об ошибках.

Перед тем, как обсудить значение пути к земле по постоянному току, важно взглянуть на некоторые базовые компоненты и источники, которые используются почти во всех симуляциях, и понять, как они влияют на эти пути постоянного тока.

Значения и DC пути RLC компонентов

Этот раздел о значениях – и путях прохождения постоянного тока – резисторов, индуктивностей и конденсаторов.

- Индуктивности в ngspice имеют последовательно включённое нулевое сопротивление (т.е., ESR = 0).
    - Таким образом, индуктивности имеют DC путь через них.
    - Индуктивности в ngspice не имеют паразитной параллельной ёмкости или сопротивления.
- Конденсаторы в ngspice не имеют паразитной параллельной проводимости (т.е., они имеют бесконечное DC сопротивление).
    - Поэтому конденсаторы в ngspice не имеют DC пути.
    - Конденсаторы в ngspice имеют нулевое последовательное сопротивление (т.е., ESR = 0).
    - Конденсаторы в ngspice не имеют паразитной последовательной индуктивности (т.е., ESL = 0).
- Индуктивности и конденсаторы в ngspice могут задаваться положительными и отрицательными значениями и нулём. Будьте осторожны при задании отрицательных значений – это может стать причиной того, что цепь, соединённая с ними, потеряет устойчивость или будет осциллировать. Это, в свой черёд, может привести к сбою симуляции.
- Резисторы, как правило, имеют DC путь через них.
- Резисторы в ngspice могут задаваться положительными и отрицательными значениями, но не могут принимать точно нулевое значение: они ДОЛЖНЫ иметь не нулевое значение. Задание нулевого значения резистору приведёт к тому, что при запуске симуляции произойдёт сбой с сообщением об ошибке. В основном это из-за того, что любое падение напряжения на резисторе с нулевым сопротивлением (например, это может обнаружиться в цепи, или даже как цифровой «шум», при расчётах моделирования) будет генерировать бесконечный ток, который, очевидно, приведёт к отказу в верхней части динамического диапазона расчётов симулятора. Это применимо ко всем резисторам в ngspice, включая значения Ron и Roff переключателей и сопротивления, заданные в моделях устройств. Остерегайтесь задавать отрицательные значения резисторов, что может вызывать

неустойчивость или осцилляцию цепей, подключенных к ним. Результатом будет сбой симуляции.

- Общей причиной путаницы при симуляции может стать неожиданное поведение схемы, использующей обрыв цепи выключателем. Как и выключатели реального мира, выключатели в EasyEDA имеют не нулевое ON сопротивление (Ron). Они также имеют конечное OFF сопротивление (Roff). Если выключатель EasyEDA подключен последовательно с источником напряжения в разомкнутой нагрузочной цепи (такой как представлено пробником VolProbe), тогда выходное напряжение выключателя будет эквивалентно обрыву *источника напряжения, независимо от того, разомкнут или замкнут выключатель*. Это не так, как в реальном эксперименте, поскольку в реальной цепи при измерении напряжения вольтметром или осциллографом, имеет место меньшее сопротивление, чем сопротивление цепи, оборванной выключателем.
  Чтобы проиллюстрировать это, давайте рассмотрим простой пример с выключателем, подключенным последовательно к идеальному источнику напряжения в 1V, а затем измерим выходное напряжение V(Output) выключателя, когда он разомкнут и замкнут.
  Когда выключатель замкнут, выходное напряжение V(Output) = 1V.
  И это то, что показывает симуляция.
  OK. Когда выключатель разомкнут, выходное напряжение V(Output) = 0V.
  А вот то, что на самом деле показывает симуляция, V(Output) = 1V.
  О чём это? Неужели выключатель моделируется неправильно и его «заклинило»?
  Нет, конечно, выключатель и напряжение делают в точности то, что и должны. Это означает только, что результаты неожиданные, поскольку имеет место недопонимание того, чем симулируемая цепь отличается от реальной схемы.
  Ни один реальный выключатель не имеет бесконечного сопротивления в состоянии OFF, и, аналогично, нет такой вещи, как реальный разрыв цепи, представленный бесконечным импедансом нагрузки.
  Управляемые напряжением и током выключатели (Voltage и Current Controlled Switches) и статические выключатели (Static Switches) в EasyEDA не идеальны: они все имеют конечное сопротивление в состоянии OFF. Для выключателей, управляемых напряжением и током, сопротивление в состоянии OFF определяется задаваемым пользователем R(off)(Ω) параметром на правой панели свойств (Properties) с предопределённым значением 1GΩ. Статические выключатели имеют фиксированное сопротивление состояния OFF 10GΩ. Хотя это сопротивление большое, но оно не бесконечное.
  Однако эффективное сопротивление к земле выходного узла при использовании Volprobe или выражения для построения графика действительно бесконечно. Поэтому при операциях с бесконечным сопротивлением нагрузки нет разницы между ON и OFF состояниями для напряжения на выходном узле (Output node).
  Заметьте, что если сопротивление нагрузки 1GΩ, и она подключена между выходом и землёй, тогда будет разница между чистыми состояниями ON/OFF.
  Это иллюстрируется следующим примером симуляции:

EasyEDA switches are not ideal

Некоторые примеры эффекта от конечного сопротивления выключателя иллюстрирует следующая симуляция:

Effects of finite switch resistances

Заметьте, что в ngspice значения Ron и Roff, используемые для выключателей, могут быть заданы так, чтобы Ron > Roff. Это полезный способ инвертировать логический смысл выключателя.

На данный момент стоит отметить, что если важно, чтобы результаты симуляции были как можно ближе к полученным при использовании реальных измерительных инструментов в реальной схеме, тогда полезно добавлять реалистичную нагрузку к проводу в схеме для симуляции туда, где измерения напряжения должны быть сделаны в реальной схеме. Аналогично и с измерением токов, последовательно с проводом следует добавить сопротивление реального амперметра. Во избежание нежелательной нагрузки симулируемой цепи добавляйте эти сопротивления только в те места, где должны подключаться измерительные инструменты реальной схемы, и только в том количестве, в каком измерительные инструменты должны использоваться одновременно. Например, если есть два пробника осциллографа, подключённые к схеме, но один из них переносится по схеме; подключайте только два осциллографических пробника в двух местах симулируемой цепи. Если нужны измерения в разных местах схемы, перемещайте пробники и перезапускайте симуляцию.

Пути DC через Voltage и Current Sources (источники)

Этот раздел о путях постоянного тока через источники напряжения и тока.

- Voltage sources (источники напряжения) в ngspice (включая независимые V и зависимые B и E источники напряжения) имеют нулевое сопротивление источника и не имеют ограничений по току. Выходное напряжение будет постоянным для любой нагрузки по току. Это справедливо для входящего и выходящего тока. Но это не то же самое поведение, что и у регулируемых и имеющих ограничители тока источников питания постоянного напряжения. Исключая специализированные источники, такие как Source Measure Units (источники, питающие и измеряющие одновременно), источники обычно могут только обеспечивать током нагрузку с положительным или отрицательным напряжением, работая вплоть до значения тока, заданного пользователем.
    - Источник напряжения в ngspice, таким образом, имеет DC путь через него.

Как следствие их нулевого внутреннего сопротивления, через них могут протекать разрушающе большие токи, если батарейки соединить параллельно без какого-либо сопротивления между ними. Это справедливо, даже если они настроены так, что у них были в точности одинаковые напряжения. При попытке сделать это запуск симуляции вызовет сбой с сообщением об ошибке.

Похожим образом такие разрушающе большие токи будет протекать при соединении индуктивности непосредственно с реальным источником питания без некоторого сопротивления между ними, индуктивность в схемах симуляции не может соединяться параллельно с источником напряжения без некоторого сопротивления между ними. Попытка сделать это приведёт к сбою симуляции с сообщением об ошибке.

- Источники тока в ngspice (включая независимые I и зависимые B и F источники) имеют бесконечное сопротивление и не имеют ограничения по напряжению. Выходной ток будет

постоянным для любого импеданса нагрузки и напряжения источника тока. Это не такое же поведение, что у регулируемых источников питания и источников питания с ограничением тока, когда они настраиваются на выход с постоянным заданным током. Исключение представляют специализированные источники, как Source Measure Units; обычные источники тока обеспечивают положительное или отрицательное напряжение на выходе, как это задано пользователем.

- Источники тока в ngspice по этой причине не имеют DC пути через них.

Как следствие генерации постоянного тока через их бесконечное внутреннее сопротивление, могут появляться разрушающе большие напряжения, если два источника тока соединены последовательно без какого-либо резистора с конечным сопротивлением для каждого источника; источники тока симулятора не могут соединяться последовательно без некоторого резистора с конечным сопротивлением, включённым параллельно с каждым источником. Попытка сделать это приведёт к сбою симулятора с сообщением об ошибке.

- При симуляции подключение конденсатора параллельно источнику тока, генерирующему ток I, без подключенного параллельно резистора R приведёт к тому, что напряжение на конденсаторе поднимется до бесконечности. Даже с параллельным резистором напряжение поднимется до I*R, что может всё ещё быть очень большим напряжением. Если такая цепь есть в симулируемой схеме, пока она не будет закорочена выключателем с некоторым подходящим резистором с низким сопротивлением, напряжение на конденсаторе будет уже увеличено до бесконечности в момент t=0, то есть, когда стартует симуляция. Это может привести к неожиданно большому напряжению сразу после запуска симуляции.

Эффект добавления DC путей

Этот раздел о создании DC путей и некоторых последствиях, которые это может иметь.

- Излюбленное в классах элементарной инженерной электрики – и пример очень сложной для решения проблемы при использовании симулятора – это цепь, где два конденсатора соединены последовательно. В этом случае необходим DC путь к земле от общей точки между двумя конденсаторами, но путь к земле не должен быть непосредственным. Он может быть через другие элементы схемы. В качестве примера, если один конец конденсатора подключен к земле, когда другой конец другого конденсатора подключен непосредственно к заземлённому источнику напряжения или к Thevenin Source (источник напряжения с последовательно включённым резистором), или к Norton Source (источник тока с параллельно включённым резистором), тогда добавления резистора с большим сопротивлением параллельно с другим конденсатором или просто к точке, где оба конденсатора заземлены, будет достаточно.

Следующий простой пример не будет работать, поскольку нет DC пути к земле от общего узла B между двумя конденсаторами:

Capacitors in series 01

- Резисторы, размещённые на конденсаторах и образующие путь DC, могут быть масштабированы так, чтобы константа R_C time (постоянная времени), которую они

создают с конденсаторами, стала очень большая по сравнению с интервалом времени, через который симуляция переходного процесса (Transient) будет работать. Другой вариант взглянуть на это для AC Analysis (анализ на переменном токе) в том, чтобы 1/(2_pi_R_C) частота, которую формирует такая цепь, была далеко за пределами частотного диапазона, который нас интересует.

Значение этих резисторов может быть где-то между milliΩ (1e-3 Ω) и вплоть до 1G Ω (1e9 Ω). Во многих случаях значения могут быть меньше или больше, чем этот диапазон, но в некоторых цепях это может стать причиной того, что симуляция терпит неудачу с ошибками. Обычно это случается из-за того, что отношение наибольшего к наименьшему напряжению или току в цепи слишком велико для поддержки симуляции. Основное правило – поддерживать отношение между наибольшим и наименьшим значениями компонентов для любого заданного типа пассивных компонентов не более, чем 1e12, что должно помочь избежать этого рода сбоев при симуляции.

- Однако следует отметить, что в то время, когда чисто теоретический анализ напряжения на конденсаторах, включённых последовательно, из-за тока, проходящего через них, основан на Q=I_t=C_V и может дать удобные значения, настройка как реальных, так и симулируемых схем для демонстрации этого может оказаться довольно сложной. Измерение напряжения в реальной схеме может быть затруднено, в то время как необходимость в DC путях для симуляции может создать различные, но не менее сложные проблемы.

- Аналогично получается и со схемой с трансформаторной развязкой, где первичная обмотка подключена к фазе и нейтрали, а одна из вторичных обмоток соединена с землёй (возможно, там, где символ земли добавлен к диаграмме схемы). Схема должна иметь DC путь обратно к этой земле от одного или обоих концов первичной обмотки трансформатора (или от центрального отвода, если таковой есть).

Как и при размещении резисторов на конденсаторах, резисторы DC пути, используемые с индуктивностями, можно масштабировать так, чтобы постоянная времени R/L_time, которую они создают, будет очень большой сравнительно с интервалом времени, через который заработает симуляция переходного процесса. Другой способ взглянуть на это для AC Analysis в том, чтобы 1/(2_pi_R*C) частота, которую они формируют, была далеко за пределами интересующего нас частотного диапазона.

Другой пример, где резисторы с большим значением сопротивления будут добавляться – это трансформаторы развязки в коммуникационных сетях данных, таких как 100BaseTx или 1000BaseT Ethernet. Где связь между двумя передатчиками плавающая, и, соответственно должна иметь DC пути к земле, добавленные для сохранения удачной симуляции. Использование резисторов, сопротивление которых велико в сравнении с характеристическим импедансом сетевого кабеля, а, значит, с нагрузочным сопротивлением, важно в плане их несоответствия с импедансом, что, таким образом, может нарушить целостность сигнала. На практике это упрощается добавлением двух резисторов: по одному от каждого из двух сигнальных проводов к земле. Это сохраняет симметричность сигналов на проводах и удваивает эффективное сопротивление между ними.

Во многих схемах с развязывающими трансформаторами самое простое решение – это заземлить один из выводов первичной и один из выводов вторичной обмотки в схеме. Это может выглядеть

непривычно, поскольку схема теперь осталась без гальванической развязки с помощью трансформатора, но причина такого соединения в его необходимости только для симуляции (и удаляется или модифицируется перед переходом к разводке печатной платы!) и остаётся простым и очень полезным решением. И оно не только убирает проблемы с тем, какими, большими или маленькими, должны быть резисторы для DC путей, но и относится также к напряжению и первичной, и вторичной обмотки относительно земли, упрощая измерение множества напряжений на обеих сторонах трансформатора, которые иначе следовало бы измерять как разностные.

Некоторые примеры подобного использования возвратных резисторов для пути к земле могут быть показаны в следующей коллекции примеров трансформаторов и развязывающих индуктивностей, включая простой пример обратноходового преобразователя с открытой петлёй:

Transformers and coupled inductors

Transformers and coupled inductors 1

Transformers and coupled inductors 2

Transformers and coupled inductors 3

Open loop flyback converter

- Чтобы избежать путаницы между пассивными схемами, предназначенными для передачи в PCB layout, и схемами для симуляции, которые могут иметь добавочные резисторы для DC путей, хорошей будет идея чётко определить, какие резисторы добавлены к схеме только для образования DC путей к земле. Это можно сделать, используя резисторы с именами, такими как *RDCPATH1*, или маркируя их, например, как *For simulation only*.

- Следует помнить, что все напряжения в схеме измеряются относительно земли. Это особенно важно помнить при работе с сигналами из разряда плавающих, такими как в примерах с развязывающими трансформаторами, которые обсуждались выше. И тогда, когда B или E Sources используются для измерения двух плавающих напряжений или вычитания одного из другого для создания разностного напряжения между ними.

Общие проблемы с DC путями

DC путь к земле часто предлагается для остальной схемы, но есть некоторые случаи, на которые часто не обращают внимания:

- Все источники тока (независимые и зависимые B и F sources) идеальны: они имеют бесконечное сопротивление по постоянному току (и AC импеданс), так что постоянный ток через них не проходит. Подключение источника тока к конденсатору, имеющему один конец на земле, приведёт к ошибке, даже если источник тока установлен на ноль (напряжение на конденсаторе будет стремиться к бесконечности, если задан ненулевой ток, и это приведёт к ошибке другого рода!). Так что, к источнику тока должен быть подключен резистор, чтобы организовать DC путь к земле для другого конца источника тока/конденсатора. Похожая проблема возникает, если два источника тока включены последовательно, даже если они идентичны (если не так, тогда общий узел вновь улетает в бесконечность).

- Выключатели имеют их собственные внутренние пути по постоянному току между контактами выключателя, поскольку у них конечное сопротивление в выключенном состоянии. Однако **voltage controlled switches** (выключатели, управляемые напряжением) не имеют DC пути между их входами управления.

- **Current controlled switches**, **Current Controlled Current Sources (CCCS) и Current Controlled Voltage Sources (CCVS)** – все имеют нулевое сопротивление короткого замыкания между их управляющими входами. На первый взгляд может показаться, что подключение выхода источника тока к входу управляемого током выключателя не приведёт к возникновению проблем, поскольку вход управления выключателем добавляет короткое замыкание на выход источника тока. Однако в этом случае отсутствует DC путь к земле от этих стыков, так что нужен резистор, соединяющий вывод входа управляемого источника тока с землёй. Если выход/управляющий вход выключателя цепи не соединён с чем-либо ещё, тогда резистор может быть заменён прямым соединением с землёй.

Это иллюстрируется размещением RDC_PATH_TO_GROUND 1k резистора, подключенным к выходу CCCS F1 и входу управляемого током выключателя W1 в примере ниже:

Controlling EasyEDA switches

- Конденсаторы (пока не используются более «продвинутые» опции ngspice) идеальные: они не имеют паразитного (утечки) DC сопротивления параллельно с ними. По этой причине оба конца конденсатора должны иметь DC путь к земле либо через внешнюю цепь, либо через явно добавленный резистор.

- Хотя первичная и вторичная (или вторичные) обмотки трансформатора имеют DC пути, конечно, нет DC пути от первичной обмотке к вторичной. Это должно быть дополнено либо соединением одного конца первичной и конца вторичной с землёй непосредственно, либо через резистор.

- Источники напряжения (включая независимый V и зависимые B и E voltage sources) имеют обратную проблему. Они идеальные, так что, они имеют нулевое внутреннее сопротивление. То же справедливо для простых индуктивностей. Если вы включаете источники напряжения параллельно, тогда ngspice даст ошибку, даже если источники напряжения заданы, как имеющие одинаковое значение напряжения (если это не так, тогда будет протекать бесконечно большой ток, что приведёт к другой ошибке). Та же проблема возникает, если вы соединяете источник напряжения параллельно с простыми, идеальными индуктивностями, поскольку источник напряжения пытается пропустить бесконечно большой ток через индуктивность.

Подобная проблема часто обнаруживается, когда устраивают трансформатор из источника напряжения. Добавленный последовательно резистор с небольшим сопротивлением устраняет эту маленькую ловушку (на практике всегда будет конечное сопротивление обмоток!).

Компоненты соединены netnames (именами соединений)

Важно понимать, что хотя компоненты в схеме могут быть показаны как соединённые проводами или метками (netlabels), в spice netlist они соединены чисто названиями соединений, данных им либо автоматически EasyEDA, либо вручную пользователем, поместившим метки. Это включает

соединения, связанные любыми из трёх NetFlag GND заземляющих символов. Так же включает соединения, связанные Net Port, NetFlag VCC и NetFlag +5 символами, как показано ниже:

Nets can be joined by netnames 01

Точно так же важно понимать, что когда провод между двумя компонентами в схеме прерван, тогда – исключая, что проводам с обеих сторон обрыва явно даны одинаковые имена вручную с помощью меток с одинаковым именем – тогда *провода по обеим сторонам обрыва получат разные метки*, поскольку даже в случае, когда одна сторона уже имеет присвоенное вручную имя, другая сторона будет автоматически поименована заново произвольным образом, присвоенным ей EasyEDA именем.

Значение ручного присвоения имён соединениям станет очевидным чуть позже, когда будет рассматриваться то, как используются пробники напряжений и используется напряжение на соединении в выражениях для B источников.

Nets can be joined by netnames 02

Как только что показано, прерванное соединение создаёт два сегмента соединения, которые более не связаны вместе. Это оттого, что EasyEDA автоматически назначает разные метки каждому из концов прерванного соединения.

Чтобы вернуть связь, добавьте метки с одинаковыми именами каждому из сегментов оборванного провода. NetPorts и NetFlags могут задаваться в любом месте соединения, но позаботьтесь, чтобы маленькие серые точки соединения попали на провод.

Чтобы избежать случайного соединения проводов, которые не должны быть связаны вместе, при ручном задании меток (netlabels) позаботьтесь о том, чтобы присваиваемые имена были уникальны для каждого соединения. Например, в следующем примере использование имени «mid» вместо «ammeterneg» соединит отрицательную сторону A1 с «mid» соединением и закоротит R1.

Nets can be joined by netnames 03

Наблюдение за сигналами

После затраченного на построение схемы времени пришёл момент для финального аккорда: первого запуска симуляции. Окно **Simulation Results...** открывается и там…

Ничего.

Нет предупреждений.

Нет сообщений об ошибках.

Нет ужасных сообщений о *Timestep too short (временной шаг слишком короткий)* или *Trouble with node X (проблемы с узлом X).*

Что пошло не так?

Более всего это похоже на широко распространённую ошибку начинающих, которые забывают, что на схеме нет ни одного пробника. Симуляция прошла удачно. А то, что она не дала результатов, просто потому, что её об этом и не просили!

В плане получения каких-либо полезных выходных данных при симуляции схема должна иметь хотя бы один пробник напряжения или амперметр. При базовой симуляции это будут VolProbe или Ammeter символы из EasyEDA Libs. При более развитой симуляции это может быть использование встроенных команд **probe**. В любом случае хотя бы один тип пробника должен быть на схеме для симуляции. Если нет ни одного пробника, симуляция будет работать, но окно **Simulation Results...** останется пустым, а окно WaveForm (сигналов) не откроется.

Пробники напряжения

Все измерения напряжения в реальной схеме на самом деле – это измерения разностных напряжений. Во многих случаях это имеет место, когда наблюдение за напряжением происходит с помощью осциллографа, и очень легко забыть, что напряжение измерялось в действительности как разница напряжения между щупом и тем местом, где щуп соединяется с землёй. Точно так же легко забыть, что измерение напряжения в одном месте симулируемой схемы соотносится с местом, где располагается узел земли.

В реальной схеме измерение напряжения между двумя точками добавляет активное сопротивление между ними. С вольтметром хорошего качества это сопротивление может быть очень большим, порядка сотен MegΩ. С щупом осциллографа x10 оно будет 10MegΩ. И будет паразитная ёмкость с этим сопротивлением. И будет также паразитная индуктивность. Если измеряемое напряжение – это сигнал переменного тока (AC signal), тогда импеданс из-за этих утечек и паразитных компонентов будет также нагружать схему.

Заметьте, что при симуляции пробник напряжения имеет бесконечное сопротивление и не имеет ни паразитной ёмкости, ни паразитной индуктивности. Как результат, измерение напряжения имеет бесконечно широкую полосу пропускания.

Следующий пример иллюстрирует некоторые техники измерения, описанные выше:

Probing voltages 01

Следующий пример показывает несколько путей измерения напряжения по отношению к земле или разностного напряжения:

- EasyEDA Voltmeter;
- E source (a.k.a. Voltage Controlled Voltage Source или VCVS, источник напряжения, управляемый напряжением) с Voltage probe для измерения выхода.
- A B, source (a.k.a. поведенческий или зависимый источник), сконфигурированный как VCVS при использовании Voltage probe для измерения выхода.

Схема также демонстрирует важность:

1. Задания имён пробников, которые идентичны соединениям, на которых они установлены.
2. Именования всех соединений на схеме.

Probing voltages 02

Измерение токов

В реальной схеме измерение тока в проводнике добавляет активное сопротивление в него. Это приводит к наличию падения напряжения на амперметре. С амперметром высокого качества это падение напряжения может быть очень маленьким, порядка милливольт. Будет иметь место и некоторая паразитная ёмкость на добавленном сопротивлении и от места подключения амперметра к земле. И будет также паразитная индуктивность. Если ток измерялся как сигнал переменного тока (AC signal), тогда импеданс из-за этих утечек и паразитных компонентов будет также нагружать схему.

Заметьте, что при симуляции пробник тока имеет нулевое добавляемое сопротивление и не имеет паразитных ёмкостей и индуктивностей. Как результат, пробник тока имеет бесконечно широкую полосу пропускания.

Следующий пример показывает некоторые пути для измерения токов относительно земли или использования разности токов:

- Использование для измерения тока символа Ammeter.
- Как линейно масштабируемое напряжение с использованием H Current Controlled Voltage Source (CCVS), источник напряжения, управляемый током (или F Current Controlled Current Source (CCCS), источник тока, управляемый током, с резистором).
- Как линейно масштабируемый ток с использованием F Current Controlled Current Source (CCCS), источник тока, управляемый током, управляемый Ammeter.
- Как напряжение, которое может быть произвольной функцией тока, проходящего через 0V Voltage Source, использующей BV source (или a BI source с резистором).
- Как ток, который может быть произвольной функцией тока, проходящего через 0V Voltage Source, использующей BI source, управляемый Ammeter.

Однако заметьте, что хотя источник с 0V может использоваться для *наблюдения* за током, его нельзя использовать для *измерения* тока, так что он может непосредственно отображаться в окне Simulation Results... или графиком в Waveform.

Probing currents 01

Конфигурирование источников напряжения и тока

EasyEDA Libs предлагает целую линейку того, что относится в ngspice к **Dependent Sources** (зависимые источники).

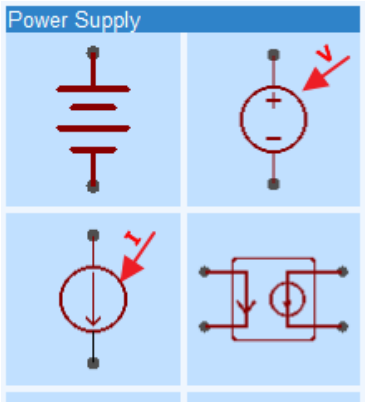
Это источники напряжения и тока, чьи выходы определены списком значений или параметров. Выходы не зависят более ни от чего.

Эти источники уже обсуждались в терминах их сопротивления DC source и DC путей.

Пока в большинстве примеров было использовано питающее напряжение либо как идеальный источник напряжения, либо как Thevenin или Norton Sources.

Их использование предполагает работу с сигналами во временной области (изменение во времени), использованными во введении и в примерах о трансформаторах, но до сих пор оно не объяснено.

Вы найдёте это ниже.



Этот раздел описывает в деталях, как любые зависимые источники могут настраиваться для поддержки следующих типов источников сигналов:

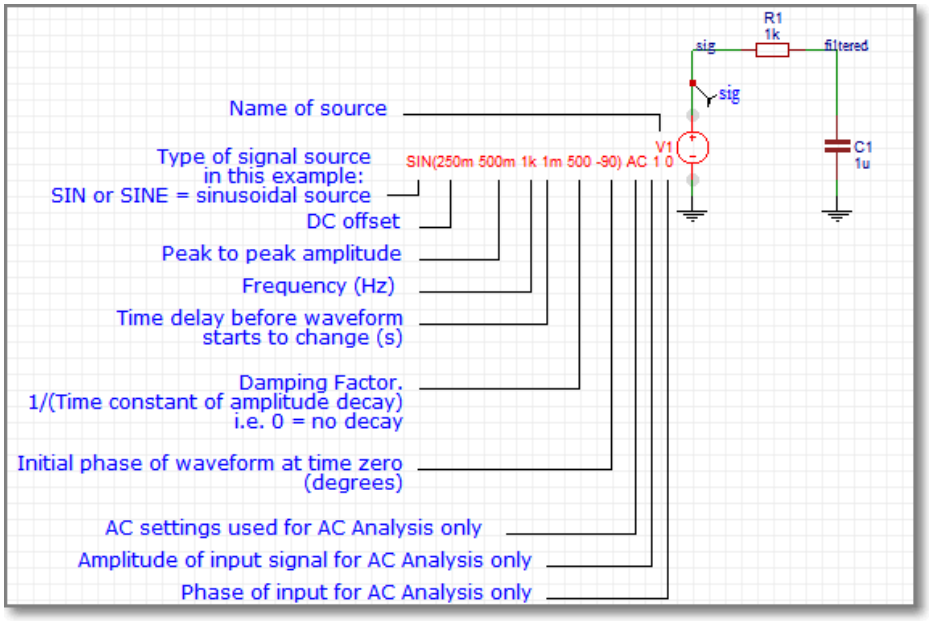
- SINE или SIN: синусоидальный сигнал.
- PULSE: импульс общей формы.
- EXP: единичный импульс с экспоненциальными нарастающим и спадающим фронтом.
- SFFM: (Single Frequency Frequency Modulated) сигнал с синусоидальной несущей и частотной модуляцией синусоидальным сигналом.
- AM: (Amplitude Modulated) сигнал с синусоидальной несущей и амплитудной модуляцией единственной частотой.
- PWL: (PieceWise Linear sources} источник сигнала произвольной формы с сигналом, созданным как список времён и уровней с последующей линейной интерполяцией между каждой временной точкой.

Хотя примеры этого раздела иллюстрируют только то, как конфигурировать зависимые источники напряжения (Dependent Voltage Sources), зависимые источники тока (Dependent Current Sources) конфигурируются точно так же.

Конфигурирование SIN или SINE опции

Конфигурирование SINE опции для создания источника немодулированного, синусоидального сигнала одной частоты.

Spice Sinusoidal Source example



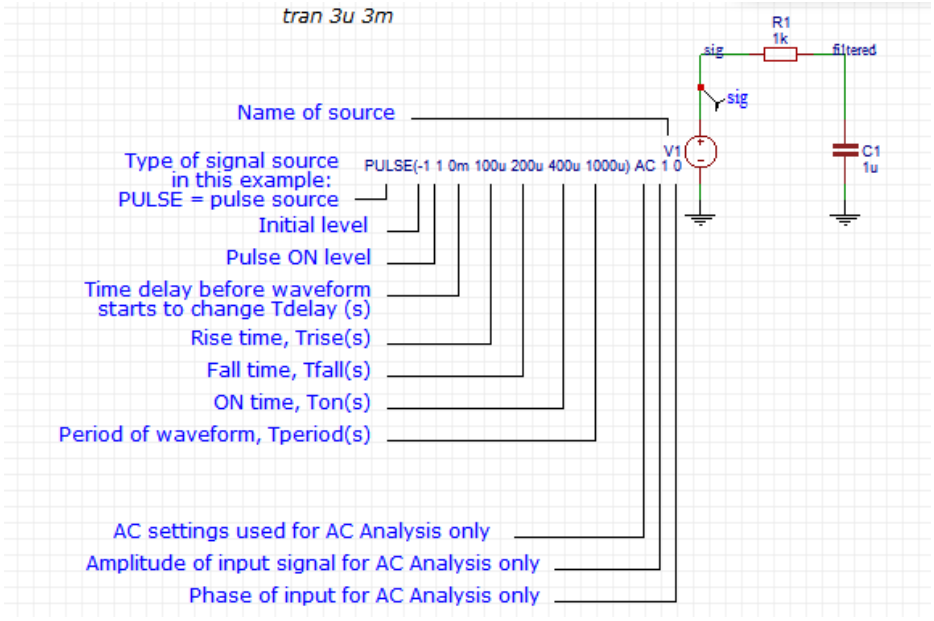
Больше способов использования SIN (или SINE) опции

Spice Sinusoidal Source: more examples

Конфигурирование PULSE опции

Конфигурирование PULSE опции для создания источника импульсного сигнала.

Spice PULSE Source



Больше способов использования PULSE опции

Spice PULSE Source: more examples

Конфигурирование EXP опции

Конфигурирование EXP опции для создания источника единичного импульса с экспоненциальными фронтами.

Spice EXP Source

Конфигурирование SFFM опции

Конфигурирование SFFM опции для создания источника простого, единственной частоты частотно-модулированного синусоидального сигнала.

Spice SFFM Source

Конфигурирование AM опции

Конфигурирование AM опции для создания источника простого, единственной частоты амплитудно-модулированного синусоидального сигнала.

Spice AM Source

Конфигурирование PWL опции

Конфигурирование PWL опции для создания произвольного кусочно-линейного источника сигнала.

Spice PWL Source

Конфигурирование AC source опции

Очень трудно увидеть, как задавать Amplitude и Phase в AC Source опциях независимых источников напряжения и тока (Independent Voltage и Current Sources), и что они реально означают, если сигналы при AC Analysis нельзя видеть во временной области переходного процесса (Transient Analysis). Чтобы попытаться визуализировать эти настройки и то, что они представляют, следующий набор примеров демонстрирует эти настройки так, чтобы можно было вернуться к их эквивалентам во временной области.

Первый пример показывает, насколько больше, чем один AC Source, можно конфигурировать в схеме для представления разных источников сигналов той же частоты, но разной фазы. Пример также показывает, как настройки фазы относятся к тому же сигналу во временной области.

В этом примере оба AC Sources настроены на одинаковую амплитуду 1. Они могут быть настроены на разные амплитуды: попробуйте это и сравните результаты с изменениями при одной амплитуде во временной области части источников сигналов.

Configuring AC Sources 01

Однако заметьте, что AC Analysis предполагает, что цепь совершенно линейная, так что, если даже амплитуда AC Source была задана как 100, выход будет выглядеть так, как после прохождения чисто линейной цепи. Сравните это с тем, что случится, если на части временной области источник будет установлен к 100!

DC смещение синфазного AC Source в этом примере важно, поскольку оно основано на Q1 в диапазоне, где оба, эмиттер и коллектор, работают в линейной области.

Это можно очень ясно увидеть с помощью пробников V(Q1E) и V(Q1C) в Transient Analysis. Если смещение DC увеличивается, тогда V(Q1C) как V(Q1E) будет расти до насыщения Q1, а V(Q1E) начнёт тянуть V(Q1C) обратно вверх. В точке, где это происходит, небольшой прирост сигнала на выходе с коллектора проходит через ноль, а затем становится неинвертирующим коэффициентом усиления меньше единицы.

Если напряжение смещения DC уменьшается почти до земли или даже ниже, Q1 выключается, так что малосигнальное усиление и коллектора, и эмиттера фактически падает до нуля. Вновь это можно ясно увидеть в Transient Analysis.

Что не так очевидно, так это то, что хотя эти эффекты всё-таки ещё обнаруживаются при AC Analysis, поскольку они представлены в частотной области, их подчас трудно интерпретировать.

Так что информация, полученная из всего этого, такова, что если AC Analysis показывает усиление меньше ожидаемого, тогда лучше всего проверить рабочую точку схемы по постоянному току, не находится ли какая-то часть её в насыщении или отсечке. Один пример этого – неправильное смещение входа ОУ, так что, выход достигает напряжения одного или другого питающего напряжения. Забыть подать питающее напряжение – это другая широко распространённая ошибка.

Следует также понимать, что хотя некоторое количество AC Sources может быть размещено на схеме, каждый со своей собственной амплитудой и фазой, все источники будут работать на той частоте, что определена установками в AC Analysis, а не самими источниками.

В цепи с несколькими независимыми источниками AC sources могут просто добавляться к ней, удаляться из неё или перемещаться по ней простым добавлением значений AC амплитуд и фаз к требуемому источнику. Так в примере выше характеристика синфазной и противофазной стороны по всей схеме может наблюдаться просто заданием одного AC Source, или другого с нулевой амплитудой, или только удалением AC частей конфигурации источника.

Другим примером этого может быть частотная характеристика усилителя от входа до выхода, которую можно нарисовать, используя AC Source на входе, в то время как частотная характеристика усилителя от пульсаций питания до выхода может изображаться качанием установки AC Source до напряжения источника, используемого для питания.

Вторая симуляция показывает также, что может конфигурироваться более одного AC Source в схеме для представления разных источников сигналов одной частоты, но с разными амплитудами и фазами. Этот пример также показывает, как задание амплитуд и фаз сказывается на одном и том же сигнале во временной области.

Configuring AC Sources 02

Настройки анализов

Что такое анализ (Analyses)?

Анализ – это просто инструкция для симулятора, указывающая ему, какой тип симуляции от него требуется при создании spice netlist.

Есть несколько разных типов анализа, которые могут реализоваться при запуске симулятора.

EasyEDA, в сущности, поддерживает подмножество анализов spice, которые доступны в ngspice. Поддерживаемые через диалог Simulate... (доступный с помощью «бегущего зелёного человечка» green running man > Run the Document или Run the Project опции или прямо через **CTRL+R**) типы анализа описаны ниже.

SPICE Analyses доступные с помощью кнопки Green Man/Simulate...

Щёлкните по иконке с зелёным человечком (Simulate...), откроется меню опций Simulation.

- Run the Document...

Предназначается для запуска симуляции схемы на единственном листе.

- Run the Project...

Это для запуска симуляции всех листов схемы в проекте.

Заметьте, что префиксы компонентов (позиционные обозначения) должны быть уникальны для всех листов схемы: то есть, не должно быть дублирующих позиционных обозначений.

Попытка запуска симуляции любого листа схемы или подборки листов, содержащих дубликаты позиционных обозначений, приведёт к ошибке «Prefix conflict error».

С помощью:

Simulate... > Run the Document... / Run the Project...

следующие SPICE анализы могут запускаться из простого диалога. Для получения доплнительной информации о том, что они делают, пожалуйста, пролистайте книгу до соответствующих разделов.

1) DC op pnt

2) DC Transfer

3) DC Sweep

4) AC Analysis

5) Transient

SPICE Analyses и Control Statement Syntax (синтаксис выражений управления)

Следующие SPICE analyses могут быть заданы непосредственно в текстовом поле схемы.

Несколько выражений для анализа можно ввести в единственную схему, но одна и только одна может быть сделана активной для одного запуска моделирования. Чтобы активировать выражение для анализа, сделайте:

Properties > Text Attributes > Texttype > spice

Для деактивации выражения анализа сделайте:

Properties > Text Attributes > Texttype > comment

Затем достаточно сделать просто:

CTRL+R

Несколько выражения для анализа можно ввести в единственную схему, но одно и только одно может быть сделано активным для единственного запуска моделирования.

1) OP: выполняет анализ рабочей точки

Основная форма:

op

Пример:

```
op
```

Заставляет SPICE выполнить анализ рабочей точки, чтобы определить спокойное состояние схемы с закороченными индуктивностями и оборванными конденсаторами. Результаты этого анализа используются для расчёта значений линеаризованных, малосигнальных моделей нелинейных устройств.

2) TF: выполняет анализ передаточной функции на постоянном токе

DC transfer function анализ – фрагмент SPICE расчётов следующих малосигнальных характеристик:

отношение выходной переменной к входной переменной (усиление или передаточное усиление);

сопротивление относительно источника входного сигнала;

сопротивление относительно выхода;

Выражение TF может быть использовано для поиска Thevenin эквивалентного малосигнального сопротивления. (Thevenin напряжение задаётся узловым напряжением схемы с открытым выходом, как результат OP выражения).

Основная форма:

tf OUTvar inSRC

Примеры:

```
tf V(5, 3) VIN
```

```
tf I(VLOAD) VIN
```

Команда TF определяет малосигнальный выход и вход для DC малосигнального анализа. OUTvar – это малосигнальная выходная переменная, и в SRC является малосигнальным входным источником. Если эта строка включена, SPICE просчитывает DC малосигнальное значение передаточной функции (output/input), входное сопротивление и выходное сопротивление.

3) DC: выполняет DC-Sweep Analysis (развёртку на постоянном токе)

В процессе DC-sweep analysis SPICE пошагово меняет значение заданного независимого источника напряжения или тока в определённом пользователем диапазоне и выполняет анализ рабочей точки для каждого из значений. Это позволяет оценить передаточную функцию на постоянном токе, и также предоставляет механизм для рисования графиков характеристических кривых устройств и моделей.

Основная форма:

dc Source-Name Vstart Vstop Vincr [Source2 Vstart2 Vstop2 Vincr2]

Примеры:

```
dc vin 0.25 5.0 0.25

dc vin 0 10 .5 vgs 0 5 1

dc vce 0 10 .25 ib 0 10u 1u

dc R1 0 1k 100

dc TEMP 0 100 1
```

Параметры определяют источник DC передаточных кривых и ограничения развёртки. Source-Name – это имя независимого источника напряжения или тока, сопротивления или температуры схемы. Vstart, Vstop и Vincr – это начальное, конечное значения и шаг соответственно. Первый пример заставляет значение источника напряжения развёртываться от 0.25 вольт до 5.0 вольт шагами по 0.25 вольт. Второй источник (Source2) может дополнительно задаваться для развёртки с ассоциированными параметрами развёртки. В этом случае первый источник «качается» в его диапазоне для каждого из значений второго источника.

Стоит подчеркнуть, что DC Sweep Spice Analysis позволяет не только использовать источники напряжения и тока для развёртки, но также температуру и сопротивление, поскольку Source-Name может относиться и резистору в схеме или ключевому слову TEMP, означающему температуру в градусах Цельсия.

Следующие симуляции иллюстрируют развёртку:

- источника напряжения;

Plot and compare diode forward currents vs. voltage

- значения резистора;

Sweep a resistor value

- температуры окружающей среды, воздействующей на каждый компонент при симуляции;

Sweep the ambient temperature

4) AC: выполняет малосигнальный AC (в частотной области) анализ

AC малосигнальный фрагмент SPICE расчётов выходных AC переменных как функции частоты. Программа вначале рассчитывает рабочую точку схемы и определяет линеаризованные, малосигнальные модели для всех нелинейных элементов в схеме. Результирующая линейная схема затем анализируется в заданном пользователем диапазоне частот. Нужный выход ac small-signal analysis, как правило, это передаточная функция (усиление по напряжению, передаточный импеданс и т.д.). Если схема имеет только один сигнальный вход, тогда он преобразуется к единичному сигналу с нулевой фазой, так что, выходная переменная имеет то же значение, что и передаточная функция выходной переменной по отношению к входу.

Основная форма:

ac (DEC | OCT | LIN) N Fstart Fstop

Примеры:

```
ac dec 10 1 10K

ac dec 10 1k 100Meg

ac lin 100 1 100HZ
```

Используйте:

'**dec**' для декадного варьирования, в этом случае N – это количество точек на декаду;

'**oct**' для октавного варьирования, в этом случае N – это количество точек на октаву;

'**lin**' для линейного варьирования, когда N – это общее количество точек.

Fstart – начальная частота, а Fstop – конечная частота.

5) TRAN: выполняет Transient (во временной области) анализ

Анализ переходного процесса – фрагмент SPICE расчёта выходной переменной как функции времени с заданным пользователем интервалом. Начальные условия автоматически определяются DC анализом. Все источники, которые не зависят от времени (например, источник питания), принимают их DC значение.

Основная форма:

tran Tstep Tstop [Tstart [Tmax]] [uic]

Примеры:

```
tran 1ns 100ns 0ns 2ns

tran 1ns 1000ns 500ns 10ns

tran 10ns 1us 0us 20ns uic
```

Tstep – это предлагаемый шаг увеличения для расчёта.

Tstop – это конечное время.

Tstart – это начальное время.

Если Tstart опущено, оно приравнивается к нулю. Сам Transient analysis всегда начинается с времени равного нулю.

В интервале <zero, tstart=""> схема анализируется (до достижения устойчивого состояния), но выходы не запоминаются. В интервале <tstart, tstop=""> схема анализируется и выходы запоминаются. Tmax – это максимальный шаг, который использует SPICE; попробуйте начать с Tmax=(Tstop-Tstart)/50.0.

Дополнительное ключевое слово «uic» (использовать начальные условия) показывает, что пользователь не хочет, чтобы симулятор ngspice рассчитывал рабочую точку устойчивого состояния до начала анализа переходного процесса. Если ключевое слово задано, ngspice использует значения, заданные с использованием IC=... на разных элементах в качестве начальных условий transient и продолжает анализ. Если управляющая строка .ic была задана, тогда узловые напряжения в строке .ic используются для расчёта начальных условий устройств. IC=... будет иметь приоритет перед значениями, данными в управляющей строке .ic. Если либо IC=... , либо управляющая строка .ic задана для отдельного узла, узловое напряжение принимается равным нулю.

Пожалуйста, посмотрите описание ниже управляющей строки .ic по её интерпретации, когда uic не задано.

6) IC: задаёт начальные условия

Основная форма:

.IC V(n1)=VAL <V(n2)=VAL><...>

Пример:

```
.IC V(11)=5 V(4)=-5 V(2)=2.2
```

Строка IC задаёт начальные условия анализа переходного процесса. Она имеет две разные интерпретации, зависящие от того, будет ли задан параметр UIC в управляющей строке .TRAN. Не следует путать эту строку со строкой .NODESET. Строка .NODESET помогает только сходимости DC и не сказывается на конечном решении смещения (исключая мульти-стабильные схемы). Эти две интерпретации строки следующие:

- Когда задан параметр uic в строке .tran, тогда узловое напряжение, задаваемое в .ic управляющей строке, используется для вычислений с начальными условиями конденсаторов, диодов, BJT, JFET и MOSFET. Это эквивалентно заданию ic=... параметра в каждой строке устройства, но это гораздо удобнее. Ещё ic=... параметр может быть задан и иметь приоритет перед значениями .ic. Если dc bias (смещение) не задано (начальный переходной процесс), решение рассчитывается до начала transient анализа. Следует обратить внимание на то, чтобы были заданы все источники постоянного напряжения в

управляющей строке .ic, если они используются при расчёте начальных условий устройства.

- Когда параметр uic не задан в управляющей строке .tran, dc bias (инициализация transient) решение вычисляется перед анализом переходного процесса (transient analysis). В этом случае узловое напряжение, задаваемое в управляющей строке .ic, приводится к нужному начальному значению при расчёте смещения (bias). В процессе transient analysis ограничение на эти узловые напряжения удаляются. Этот метод является предпочтительным, поскольку позволяет ngspice найти совместимое dc решение.

- Заметьте, что «uic» опция должна использоваться с осторожностью.

Как правило, анализ рабочей точки выполняется перед началом анализа переходного процесса. Результат этого анализа рабочей точки передаётся в начальные условия для схемы при t=0.

«uic» spice директива подавляет эту инициализацию.

Начальные условия для некоторых элементов схемы могут задаваться на основе «от экземпляра к экземпляру». Например, транзисторы могут настраиваться так, что они в начальной стадии будут выключены (OFF); выключатели могут быть в состоянии ON или OFF в начальный момент; .IC spice директива позволяет задавать напряжение на соединении в момент t=0.

Если опция «uic» добавлена в spice директиву tran, тогда используются все заданные начальные условия.

Однако важно понимать, что если директива «uic» используется без явно указанных начальных условий, тогда, поскольку анализ рабочей точки на постоянном токе опущен, используются значения по умолчанию. Это может вызвать проблемы при ряде симуляций, поскольку значения по умолчанию могут приводить к нефизическим начальным условиям во всей схеме. Например, рассмотрим идеальный источник напряжения, подключённый параллельно идеальному конденсатору. Пока не задано иное, начальное значение по умолчанию источника напряжения принимается равным нулю. Таким образом, напряжение на конденсаторе будет нулевым при t=0. Затем, на первом временном шаге источник напряжения устанавливается к рабочему напряжению, что приводит к бесконечному току от источника к конденсатору, заряжаемому до этого рабочего напряжения. Симулятор не может найти наименьший временной шаг, чтобы сделать этот ток конечным, а в результате вы получите сообщение «time step too small convergence fail».

Заметьте, что когда опция «uic» не используется, любая директива .IC , включённая в симуляцию, всё-таки используется.

Начальные условия и запуск схемы

Некоторые фоновые и основные техники запуска

Все симуляции во временной области начинаются с t=0.

Хотя можно начинать прорисовку графика результатов симуляции в любое время после t=0, сама симуляция всегда стартует с t=0.

Начальные DC условия для схемы полностью определены начальными уровнями или DC смещениями любых источников, присутствующих в схеме. Так что, например, питающее напряжение, которое задано как 9V, трактуется симулятором как имеющее 9V всё время до t=0.

Аналогично, источник SINE (синусоидального напряжения) со смещением -1V DC или источник PULSE с начальным уровнем -1V трактуются симулятором, как имеющие -1V DC всё время до t=0.

Поэтому напряжения в схеме, такие как смещения баз биполярных транзисторов от делителя и т.д., будут заданы установившимися значениями DC для всего времени t<0. Следовательно, напряжения на конденсаторах и токи через все индуктивности в схеме будут соответствовать их установившимся значениям до старта моделирования в момент t=0.

Даже если схема – это осциллятор, до момента t=0 он будет иметь устоявшееся, не осциллирующее состояние. В момент t=0 схема будет запущена при этих начальных DC условиях. Она будет либо продолжать оставаться в этом стабильном, не осциллирующем состоянии, либо будет медленно выходить из этого стабильного состояния, пока не возникнут условия осцилляции.

Начальное состояние осцилляторов, основанных на таких схемах, как использующих фазовый сдвиг, как мост Вина и осцилляторы с кварцем, будет определяться их условиями DC смещения. Если нет источников шума в схеме (состояние по умолчанию для всех компонентов пока не задано иное, как, например, для резисторов, которые могут вносить свой вклад в шумы), тогда нет ничего, что могло бы вывести схему из равновесия, и, таким образом, схема никогда не начнёт осциллировать.

Хотя в большинстве подобных случаев осцилляция возникает благодаря «скрытым» источникам шума, которые существуют как математический шум из-за конечного разрешения и ошибок округления при расчётах запущенного моделирования, это может занять больше времени сравнительно со временем запуска осциллятора за несколько циклов в стабильном режиме генерации. Кварцевые осцилляторы, в частности, могут потребовать сотни тысяч периодов осцилляции, чтобы запуститься и войти в стабильный режим.

Чтобы сократить при симуляции время ожидания начала осцилляции, полезно ввести некоторые начальные условия запуска, чтобы «подтолкнуть» схему к генерации.

Самый простой способ подтолкнуть схему к генерации – это заменить простой источник питания импульсным (PULSE source) с установленным начальным уровнем напряжения равным требуемому, но сконфигурированному так, чтобы получился короткий импульс питающего напряжения плюс-минус небольшое напряжение. Так, например, схема с 9V вольтовым питающим напряжением получает небольшой импульсный провал до 8.5V на 1us с 100ns фронтами. Или начальный уровень 8.5V с задержкой 1us перед 100ns фронтом импульса до уровня 9V.

Такой же приём можно выполнить, используя источник напряжения, добавленный в схему почти в любой точке цепи. Просто для внесения небольшой шаговой или импульсной добавки к смещению. Но следует помнить, что если шаговое напряжение или импульс не возвращаются к нулю после «подталкивания», тогда они станут источником дополнительного смещения для этой части схемы.

Примером этого может стать принудительный старт кварцевого осциллятора.

Кварцевые осцилляторы требуют большого времени для запуска, поскольку у них очень большое значение Q кристалла.

То же самое справедливо и для их симуляции. Так что, во избежание затягивания времени начала симуляции и генерации массива данных в файл, они нуждаются в запуске в состоянии с увеличенными временами начала и остановки, но с небольшим значением (Tstop - Tstart).

Как обсуждалось ранее, добавление небольшого шага напряжения или источника напряжения, или внутреннего узла схемы может помочь при запуске осцилляции чуть раньше, но идея добавления небольшого источника шагового или импульсного напряжения к схеме, чтобы подтолкнуть его к жизни, рассматривается как крайняя мера для EasyEDA модели кварца.

Техника, используемая в XTALfast subckt (подсхеме с кварцем), заключена в добавлении внутреннего PULSE source (импульсного источника) к модели кристалла, чтобы ввести импульс очень большой амплитуды при t=0 внутри модели. Это запускает осциллятор почти мгновенно, а поскольку импульс имеет очень широкий спектр в отличие от используемого при шаге (или всплеск присущей кристаллу резонансной частоты), осциллятор быстро запускается на резонансной частоте кристалла в данной схеме.

Время запуска примера ниже, использующего XTALfast subckt, можно сравнить с той же моделью кристалла, но без исправленного запуска, просто изменением имени модели кристалла с XTALfast на XTALnofast.

Crystal oscillator using the EasyEDA quick starting crystal model

Источники EXP и PWL можно тоже использовать как источники питания с «подталкиванием».

Большинство релаксационных осцилляторов, таких как классический мультивибратор на двух транзисторах или на таймере 555, имеют два стабильных состояния. Осцилляция обычно происходит между этими двумя состояниями, однако под воздействием DC смещения перед t=0 эти схемы часто переходят в одно или другое из этих стабильных состояний, где и остаются в момент t=0. Что означает, что они никогда не начнут осциллировать.

Вот пример простого RC релаксационного осциллятора, который не запускается сам:

Relaxation oscillator startup 01

Этому типу схем может потребоваться несколько больший «пинок», чтобы заставить их работать. Это можно сделать с помощью PULSE source (импульсный источник), но вместо добавления маленького шага источник устанавливает нужное питающее напряжение с нуля. Так, например, при задании начального уровня 0 и уровня импульса 9 с нулевым временем задержки и временем фронта 200us, схема начинает работу при всех внутренних узлах, установленных в ноль:

Relaxation oscillator startup 03

Другая возможность, которая позже будет рассмотрена подробнее, в использовании выражения как функции по времени в B source.

Например, вот выражение в B source:

V=9_(1-exp(-1_time/100u))

генерирующее напряжение, которое начинается с нуля и поднимается экспоненциально к конечному значению 9V с постоянной времени 100us:

Relaxation oscillator startup 04

В симметричных схемах, как в мультивибраторе с двумя транзисторами, хотя этого может быть недостаточно для нарушения равновесия, может быть достаточно для начала осцилляции. Может потребоваться ввести некоторую умышленную асимметрию или дисбаланс в схему, например, сделав одну базу с подтягивающим резистором, или сделать время задающий конденсатор частично отличным от другого конденсатора. Даже разницы меньшей, чем допуск реального конденсатора, может оказаться достаточно, чтобы ввести схему в самостоятельные колебания с нуля до номинального значения возрастающего напряжения питания.

Иногда сложные схемы – или простые схемы со сложными моделями – могут отказываться моделироваться, поскольку симулятор не может найти рабочую точку предшествующую моменту t=0. Такие схемы очень часто симулируются хорошо, если питание (или источники питания) поднимается с нуля. Тем же способом, как слегка разбалансированные симметричные компоненты помогают запуску, слегка разбалансированные напряжения, паузы или время увеличения питающего напряжениям могут помочь при запуске более «трудных» симуляций.

Однако есть и обратная сторона использования нарастающего с нуля напряжения питания. Как уже было сказано, при начале симуляции переходного процесса напряжение на всех конденсаторах и ток через все индуктивности в схеме достигнет их значений в установившемся режиме до начала симуляции при t=0. Если симуляция начинается с нулевого питающего напряжения при t=0, тогда, очевидно, все внутренние напряжения и токи также должны быть нулевыми, исключая некоторые небольшие смещения из-за начальных уровней каких-нибудь источников сигналов, а иногда плохо построенных источников внутри моделей, которые не переходят в ноль при нулевом питающем напряжении.

Если все внутренние напряжения и токи нулевые при t=0, тогда, возможно, потребуется больше времени, чем выбранное время остановки симуляции, поскольку всем внутренним узлам требуется время для достижения их DC стабильного состояния

Простое решение в этом случае – запустить симуляцию на время достаточное для установки всего, а строить графики результатов только для времени достаточного для работы сигналов, которые вас интересуют. Так, например, схема, которая управляется источником 1kHz, но которой необходимо 95ms для урегулирования, может наблюдаться, скажем, с t=98ms до t=100ms заданием в transient anaylsis: максимальное время шага 1us, время остановки 100ms и время начала 98ms, как это:

```
tran 1u 100m 98m
```

Такое решение хорошо работает, но в этом примере около 95% времени симуляции используется только для перехода схемы в устойчивое состояние перед тем, как будут получены какие-либо полезные результаты. Это очень расточительно для времени симуляции, а для сложных симуляций, таких как симуляция импульсных источников питания (SMPS), где конкретно эта ситуация может возникнуть, симуляция может занять много минут реального времени.

Это то, где иногда могут быть полезнее другие техники задания начальных условий в схеме.

Задание начальных напряжений на соединениях и токов через компоненты

Есть случаи, когда требуется начать симуляцию в некотором предопределённом состоянии схемы. Например, может потребоваться, чтобы симуляция начиналась в момент t=0 с предварительно заряженным конденсатором до некоторого напряжения. Аналогично ток в индуктивности может иметь некоторое значение к моменту t=0. При симуляции больших схем может оказаться полезным зарядить выходной сглаживающий конденсатор блока питания до примерно правильного напряжения питания, чтобы сохранить время, требуемое для его зарядки с нуля. Если конденсатор выходной в SMPS (импульсный источник питания), тогда полезным может стать также и изменение в индуктивности(тях) в SMPS тока до (их) среднего рабочего тока через неё.

Использование spice ic директивы

Используйте .ic spice директиву для установки начального напряжения на соединении.

Больше информации о .ic spice директиве вы найдёте в:

About-spice-analyses-in-EasyEDA

Использование .ic spice директивы иллюстрируется в этих двух примерах:

Setting initial circuit conditions 01

Relaxation oscillator startup 02

Следующие примеры показывают два способа использования добавленных «uic» опций к Transient Analysis, но по причинам уже данным в описании «uic» опции в разделе «IC: задаёт начальные условия» в «Настройки анализов», следует проявлять осторожность в использовании этой опции.

Relaxation oscillator startup 05

Relaxation oscillator startup 06

Использование источника тока для задания начального значения через индуктивность

Директива .ic spice может использоваться только для задания начального напряжения на одном или нескольких соединениях. Её нельзя использовать для задания начального тока через компонент. Этот пример иллюстрирует простой способ использования источника тока для задания начального тока через компонент.

Setting initial circuit conditions 02

Задание напряжения на конденсаторе использованием XSPICE модели конденсатора

В качестве альтернативы использования директивы .ic: задание начального напряжения на конденсаторе с использованием модели конденсатора XSPICE вместо предопределённого конденсатора EasyEDA.

Setting initial circuit conditions 03

Задание тока индуктивности использованием XSPICE модели индуктивности

В качестве альтернативы источнику тока параллельно индуктивности: задание начального тока через индуктивность использованием XSPICE модели индуктивности вместо предопределённой индуктивности EasyEDA.

Setting initial circuit conditions 04

Некоторые схемы могут запускаться самостоятельно, но простая замена модели компонента может вызвать сбой при запуске. Не бойтесь пробовать эти методы перед тем, как тратить время на попытки найти какие-то другие неясные причины сбоя.

Использование источника 1V для помощи при запуске

Это широко распространённая техника, которая имеет ограниченное применение, но которая при правильных обстоятельствах может оказаться очень полезной.

Разместив источник напряжения 1V в схеме, дав имя выходу, скажем, «unity», а затем умножив, используя выражение в B Sources V(unity), можно заставить все источники B sources в симуляции стартовать с нуля в течение начальной части (DC рабочей точки) переходного процесса.

Это может иной раз помочь произвести чистый старт с нулевыми внутренними начальными состояниями.

Однако следует иметь ввиду, что как и с другими техниками, которые производят симуляцию с нуля, любая попытка запустить OP, TF или AC симуляцию может, как правило, вернуть нулевые результаты, поскольку все начальные состояния (и во многих случаях усиление) B sources будут установлены к нулевым начальным значениям.

Замена идеальных и Thevenin источников напряжения на Norton Sources с ограниченным диапазоном для помощи при запуске

Эта техника очень широко применяется, поскольку помогает улучшить сходимость симуляции, а не только поведение при запуске.

Идеальные источники напряжения при симуляции способны дать бесконечные токи, так что нагрузка, которая выглядит емкостной в любой момент симуляции, имеет возможность вызвать мгновенный бесконечный ток. Это может стать причиной переполнения при симуляции или сбою при поиске походящего следующего шага для продолжения. В подобном случае симуляция аварийно завершится из-за расходимости. Чтобы избежать этого, хорошей практикой будет добавление последовательного сопротивления к каждому источнику напряжения, будет ли это независимый источник V, зависимые E или H источники, или зависимый источник B, сконфигурированный как источник напряжения.

Это превращает все идеальные источники напряжения в Thevenin источники, то есть, источники напряжения с конечным внутренним сопротивлением.

Однако внутренне симулятор оказывается более эффективен в расчётах токов и напряжений с Norton Source, чем с Thevenin источником. Так что, следующий шаг процесса – это преобразование всех Thevenin источников для симуляции в Norton источники.

Norton Source – это точная эквивалентная схема Thevenin Source. Чтобы преобразовать Thevenin источник в эквивалентный Norton, источник напряжения Thevenin замещается источником тока равным току короткого замыкания Thevenin источника, а последовательное сопротивление источника Thevenin переносится параллельно этому источнику тока. Оба источника после этого имеют одинаковые внутренние сопротивления, напряжение при разомкнутой цепи и токи короткого замыкания.

Сделанные до этого шаги преобразования идеальных источников в Thevein источники, а затем в Norton эквивалентные источники, как правило, увеличивают скорость работы большинства симуляций, но конечный шаг ограничения полосы пропускания Norton источников может существенно улучшить запуск и сходимость многих симулируемых схем.

Ограничение полосы пропускания (или band-limiting) источника Norton реализуется легко: всё, что требуется, это добавить конденсатор параллельно внутреннему сопротивлению источника, которое уже было добавлено параллельно источнику тока. Этим создаётся фильтр нижних частот на выходе Norton источника, уменьшающий выход на высоких частотах. Что означает, что, даже если источник пропускает или генерирует высокочастотные нелинейные сигналы, которые и могут производить такого рода очень быстрые и даже бесконечные сигналы, с которыми spice с трудом справляется в своих внутренних расчётах. Поскольку частотное наполнение этих сигналов и периоды превышают частоту среза фильтра нижних частот, они уменьшаются до сравнительно медленного обрамления (то есть, вырезается высокочастотное содержимое) сигналов. Такие сигналы имеют конечные времена фронтов, так что, spice более не затрудняется в борьбе с ними. Они становятся гораздо более дружелюбными к spice непрерывными сигналами.

Даже лучше, что есть такие источники с ограниченной частотной полосой, рассыпанные по схеме (такие как входные источники сигналов), поскольку очень часто и производные от таких сигналов становятся непрерывными. Это большая подмога для spice, поскольку большая часть внутренних расчётов в spice оценивается не только там, где есть сигнал, но также осуществляется оценка по всему пути прохождения сигналов и того, как быстро они там окажутся.

Несколько слов-предупреждений уместных здесь. Замещение источников напряжения на Thevein и Norton источники, ограниченные по частотной полосе – это прекрасная техника, но требующая высокой степени понимания того, что действительно происходит при симуляции схемы.

Выражение о сказанном выше:

«...всё, что требуется, это соединить конденсатор параллельно с внутренним сопротивлением источника, которое уже добавлено параллельно источнику тока. Этим создаётся фильтр нижних частот на выходе Norton источника, который уменьшает выход на высоких частотах»

это всё очень хорошо, но обходит вопрос, какой должна быть ёмкость этого конденсатора? И нет единственного ответа на этот вопрос, поскольку частота среза, требуемая для проблем сходимости или запуска, будет сильно отличаться от одной схемы к другой, и даже от одного места в схеме к другому. Единственный важнейший фактор, влияющий на значение конденсатора, это то, что частота среза должна быть много выше, чем рабочие частоты не самой схемы, а устройств в ней. Например, схема усилителя звука, разрабатываемого для полосы до 20kHz, может использовать операционный усилитель, который работает с замкнутой петлёй ОС в полосе до 200kHz, но который внутренне может иметь усиление в полосе до 2MHz или даже до 20MHz.

На практике задание частоты среза с RC постоянной времени в 1ps (1e-12s), то есть, приблизительно 160GHz должно быть безопасно для большинства «обычных» симуляций усилителей, схем линейных и импульсных источников питания, но для высокочастотных и RF цепей рамки полосы частот должны быть соответственно раздвинуты.

Значение резистора в этой параллельной RC цепи может оказаться трудным для выбора. Для большинства мало нагруженных источников напряжения значения в 1 Ω вполне достаточно, и для упрощения расчётов параллельного конденсатора примите значение 1pF. Однако для некоторых источников может оказаться необходимым использовать большее (или меньшее) значение параллельного сопротивления. Это, в свою очередь, потребует использования пропорционально меньшей (или большей) ёмкости конденсатора, чтобы сохранить частоту среза.

Эта техника обрезания полосы пропускания не должна применяться огульно к любому существующему источнику тока в схеме. Поместив RC цепь параллельно чисто источнику тока, можно причинить больше вреда работе схемы из-за уменьшения импеданса в полосе источника тока, чем помощи от ограничения полосы пропускания!

Хотя нет примеров с проблемами запуска или инициализации, преобразование идеальных и Thevein источников в Norton демонстрируют шаги (i), (ii) и (iii) следующей симуляции:

Parameters, expressions, functions and B Sources

Использование опции 'OFF' для помощи в запуске

Некоторые компоненты, такие как выключатели, bjts, jfets, MOSFET и MESFET имеют опцию 'OFF', чтобы установить устройство в OFF начальном состоянии.

Выключатели также имеют опцию 'ON', задающую начальное состояние устройства ON.

Эта опция может быть очень полезна для обеспечения, например, того, чтобы одна сторона двух транзисторной бистабильной или моностабильной схемы или нестабильного мультивибратора была выключена, тем самым избегая ситуации, описанной выше, где оба транзистора включены в начальном состоянии. Хотя это само по себе не гарантирует, что схема будет запущена с t=0, но это может упростить любые другие меры, которые должны быть приняты, чтобы обеспечить запуск.

Эти состояния просто применяются добавлением ключевого слова OFF (или для выключателей только ON) после имени устройства. Это можно сделать либо непосредственно, редактируя имя в месте его расположения на схеме, либо через диалог на правой панели Properties (свойства). Например, для установки выключателя с именем MYSWITCH в начальное состояние ON имя должно выглядеть так:

```
MYSWITCH ON
```

Аналогично для установки bjt с именем 2N2222 в начальное состояние OFF его имя должно быть:

```
2N2222 OFF
```

Выражения

Выражения могут использоваться для определения значений компонентов, и чтобы помочь сконфигурировать Voltage и Current Sources.

Операторы

В выражениях скобки выполняются прежде других операторов. Операторы выполняются, следуя списку приоритетности (Precedence), показанному ниже. Для одинакового старшинства бинарных операций выполнение идёт слева-направо. Функции оперируют только с реальными значениями!

Operator	Alias	Precedence	Description
-		1	unary negate (см. Прим. 1 ниже)
!		1	unary not
**	^	2	power (но также см. pow(x,a), pwr(x,a) and pwrs(x,a) functions)
*		3	multiply
/		3	divide
%		3	modulo (не работает в B Source выражениях)
\		3	integer divide (не работает в B Source выражениях)
+		4	add
-		4	subtract
==		5	equality
!=	<>	5	non-equal
<=		5	less or equal
>=		5	greater or equal
<		5	less than
>		5	greater than
&&		6	boolean and
||		7	boolean or
c?x:y		8	ternary operator (см. также if(x,y,z) и ifx(x,y,z) функции)

Примечание 1

На время написания (141021) ngspice реализация unary negate или subtract symbol работают как ожидается , когда используются в таком типе выражений:

```
.param myparameter={A-B}
```

но могут производить неожиданный результат, когда используются в выражениях для B Source (или в .func выражениях) подобным этому:

V=5*(1-exp(-time/1m))

Это выражение вместо показанного выше должно быть записано так:

V=5_(1-exp(-1_time/1m))

Это особенность ngspice. Будем надеяться, что эта аномалия будет скорректирована в какой-то момент в будущем, и что это будет включено в следующие выпуски EasyEDA.

Число ноль используется для представления Булева значения False. Любое другое число представляет Булево True. Результат логических операций – это 1 или 0.

Некоторые примеры логических операторов, используемых для определения значения источников напряжения:

```
V1or 1 0 {1 | | 0}

V2and 2 0 {1 &amp;&amp; 0}

V3not 3 0 { ! 1}

V4mod 4 0 {5 % 3}

V5div 5 0 {5 \ 3}

V6not 6 0 { ! 0}
```

Заметьте, что при использовании непосредственно в компоненте и полях значения источника, выражения ДОЛЖНЫ быть в одной строке. При использовании подобных этим выражений их нельзя упаковывать более чем в одну строку.

Использование выражения для определения значений компонента

fc, частота по уровню -3dB фильтра RC низших частоты первого порядка выглядит так:

fc = 1/(2_pi_R*C)

Аналогичное выражение есть для RC фильтра высших частот первого порядка.

Если fc задано для 10kHz, а R выбрано в 1k, тогда:

C = 1/(2_pi_1k*10k)

Положим, что выход фильтра высших частот требуется уменьшить на коэффициент A:

Полное значение R для фильтра высших частот всё ещёl 1k, но оно должно быть разделено с нижним резистором, имеющим значение, заданное так:

Rlower = (Rupper+Rlower)*1/A

в то время как верхнее значение задаётся как:

Rupper = (Rupper+Rlower)*(1-1/A)

Если выбрать A=3, тогда для выбранного значения R=1k

Rlower = 1k*1/3

и:

Rupper = 1k*2/3

Просто введите правую часть этих выражений в поле значений компонента, заключив их в фигурные скобки, как это:

{expression}

значение этих компонентов будет определено непосредственно этими выражениями, как иллюстрируется Rupper и Rlower в этом примере.

Использование выражений для конфигурирования источников напряжения и тока

В этом примере PULSE source (импульсный источник) V1 конфигурируется для генерации сигнала с 20us фронтами, частотой 5kHz и равными временами в высоком и низком состоянии: другими словами, импульсы прямоугольной формы с покатыми фронтами, периодом 200us и 50% скважностью.

Поскольку источник импульсов определён в терминах Trise и Ton, может оказаться полезным подумать о временном интервале от начала переднего фронта до начала спада (заднего фронта) в качестве «pulse width, ширины импульса», Twidth:

Twidth = Trise+Ton

Тогда совсем просто определить PULSE source в терминах Trise и Twidth без использования ручного вычисления значения Ton поскольку:

Ton = Twidth-Trise

Из этого мы также можем определить, что, если «duty cycle, скважность» определяется как:

D = Twidth/Tperiod

тогда для данного D:

Ton = D*Tperiod - Trise

И, наконец, иногда удобно определять период PULSE source в терминах частоты:

Frequency = 1/Tperiod

Чтобы использовать выражение в настройках источника, просто введите его в то место, где вы хотели бы вычислить значение, заключив его в фигурные скобки, как здесь:

{expression}

Использование выражений иллюстрируется следующим примером:

Using expressions 01

Параметры

Обычно значения компонентов задаются непосредственно в полях значений компонентов. Однако есть случаи, когда желательно задать или изменить значение нескольких компонентов одновременно, но без редактирования отдельных значений компонентов.

Простая цепь резистивного делителя, использованного для иллюстрации нескольких предыдущих примеров, имеет один резистор 1k и два по 2k. Вместо введения значения 1k в один резистор, а 2k в каждый из других, можно задать две переменные, представляющие эти значения.

Для создания двух переменных, R1val=1k и R2val=2k, выражение .param помещено на схему и включено в spice директиву (сделано так: **Properties > Text type = spice)**:

.param R1val=1k R2val=2k

Параметры затем используются для определения значений компонентов в их полях значений.

Поместив более одного выражения .param на схеме, выполним:

Properties > Text type = comment

и

Properties > Text type = spice

Можно переключать задание значений без редактирования их каждый раз для отдельных компонентов.

Поэтому использование выражений .param делает возможным:

- изменение значения нескольких компонентов единственной правкой;
- определение параметров в терминах других параметров;
- определение параметров в терминах функций других параметров.

Заметьте, что можно также иметь несколько.param активными при симуляции одновременно, но, чтобы избежать конфликтных ситуаций дублированием определений, имя идентификации параметра должно быть уникальным для каждого назначения.

Синтаксис выражения .param:

.param

<param_name1>=<value1> <param_name2>=<value2> ... <param_nameN>=<valueN>

Выражения в «.param» могут помещаться более, чем в одну строку при использовании символа продолжения «+»:

.param

- =
- =
- ... +
- ... +
- ...
- =

Параметры могут быть числами, другими определёнными параметрами или выражениями, сделанными из любой комбинации чисел и заданных параметров.

Имена идентификаторов параметров должны начинаться с буквенного символа. Другие символы должны быть либо буквами, цифрами или специальными символами ! # $ % [ ] _.

Заметьте, что при использовании выражений в .param, они могут размещаться более, чем на одной строке, если использовать символ «+» продолжения:

```
.param
+ <param_name>=<value1>
+ <param_name2>=<value2>
+ <param_name3>=<{expression1}>
+ <param_name4>=<{part of expression2
+ continuation of expression2}>
+ ...
+ <param_nameN>=<valueN>
```

При выборе точки прерывания строки следует быть внимательным, чтобы чётко использовать символ «+» в качестве продолжения строки, отличая его от математического использования символа «+» при операции сложения в выражении.

Например:

```
.param x=3 y=4
+ hypotenuse={sqrt(x^2+
+ y^2)}</xmp>
```

и:

```
.param x=3 y=4
+ hypotenuse={sqrt(x^2 +
+y^2)}
```

что правильно помещает выражение при задании в .param и даст ожидаемый результат, тогда как:

```
.param x=3 y=4
+ hypotenuse={sqrt(x^2
+ y^2)}
```

или:

```
.param x=3 y=4
+ hypotenuse={sqrt(x^2+
y^2)}
```

или:

```
.param x=3 y=4
+ hypotenuse={sqrt(x^2
+y^2)}
```

могут привести к неожиданному результату ил могут привести к сбою с ошибками.

- **Заметьте, что переменные TIME, TEMPER или TEMP и HERTZ будут НЕПРАВИЛЬНЫМИ именами идентификаторов.**

Заметьте, что использование параметра в поле значения компонента должно быть заключено в фигурные скобки:

{...}

Параметр также должен быть заключён в фигурные скобки, если он используется при определении значения другого параметра, как в примере:

R3val={R2val}

Хотя здесь и не было показано, использование фигурных скобок для заключения выражений, содержащих параметры, которые затем используются для определения других параметров, является обязательным, даже при том, что использование фигурных скобок в примере выше не обязательно, хорошим правилом будет всегда заключать в фигурные скобки любые параметры или выражения, используемые в определении.

Однако заметьте, что параметры в выражениях, используемых в B Sources, не должны заключаться в фигурные скобки.

Базовое использование параметров иллюстрируется ниже:

Using parameters 01

Использование параметров в выражениях

Выражения и параметры могут комбинироваться для упрощения и автоматизации расчётов компонентов и значений конфигурации источников, как иллюстрирует следующий пример:

Using parameters in expressions 01

Функции

В разделе о выражениях и параметрах было показано, как значения компонентов и источников могут быть определены с помощью арифметических уравнений. В примерах, использованных для иллюстрации, использовались только простые линейные выражения. В этом разделе вводится понятие функции.

Функции очень расширяют мощность параметров и выражений, позволяя создавать выражения, включающие нелинейные функции других параметров.

Предопределённые функции

EasyEDA имеет некоторые предопределённые функции. Многие из них сразу доступны для использования в выражениях, поскольку они встроены в ngspice или они автоматически добавляются EasyEDA в spice netlist во время первого сохранения схемы.

Другие не добавляются автоматически в netlist, так что их определения должны быть вставлены вручную до того, как их можно будет использовать. Эти определения функций явно показаны в примерах, иллюстрирующих каждую функцию.

Эти функции скоро будут добавлены к списку тех, что автоматически добавляются в netlist.

Все доступные в настоящее время предопределённые функции включены в список вместе с примерами, иллюстрирующими их, в таблице ниже.

Заметьте, что все функции в этом списке могут быть использованы в любом контексте EasyEDA: в полях значений и в выражениях для значений компонентов, независимых источников и B Sources.

Таблица функций

Функция	Описание	ngspice native или EasyEDA special	Где можно использовать
abs(x)	Абсолютное значение x	ngspice	E, I, B
acos(x)	Арккосинус x. Не сходится для x < -1 and x > +1. Использовать invcos(x) вместо.	ngspice	E, I, B
acosh(x)	Реальная часть обратного гиперболического косинуса x, т.е., acosh(.5) возвращает 0, не 1.0472i	EasyEDA	E, I, B
arctan(x)	Альтернативный синтаксис для atan(x)	ngspice	E, I, B
asin(x)	Арксинус x. Не сходится для x < -1 and x > +1.	ngspice	E, I, B

	Использовать invsin(x) вместо.		
asinh(x)	Обратная функция гиперболического синуса x	EasyEDA	E, I, B
atan(x)	Арктангенс x	ngspice	E, I, B
atan2(y,x)	4 квадратный арктангенс x/y (tan^-1(x/y))	EasyEDA	E, I, B
atanh(x)	Обратная функция гиперболического тангенса. (Ограничивает размах выхода, чтобы избежать числового отказа из-за переполнения/понижения).	EasyEDA	E, I, B
buf(x)	Возвращает 1, если x > 0.5, иначе 0	EasyEDA	E, I, B
ceil(x)	Целое, эквивалентное или больше, чем x	ngspice	E, I, B
cos(x)	Косинус x	ngspice	E, I, B
cosh(x)	Гиперболический косинус x	ngspice	E, I, B
exp(x)	e от x	ngspice	E, I, B
floor(x)	Целые, эквивалентные или меньше, чем x	ngspice	E, I, B
if(x,y,z)	IF x > 0.5, THEN y ELSE z	EasyEDA	E, I, B
ifx(x,y,z)	IF x, THEN y ELSE z	EasyEDA	E, I, B
int(x)	Преобразует x в целое	EasyEDA	E, I, B
inv(x)	Возвращает 0, если x > 0.5, иначе 1	EasyEDA	E, I, B
invcos(x)	Действительная часть арккосинуса x, т.е., acos(-5) возвращает 3.14159, а не 3.14159+2.29243i	EasyEDA	E, I, B
invsin(x)	Действительная часть арксинуса x, asin(-5) возвращает -1.57080, а не -1.57080+2.29243i	EasyEDA	E, I, B
invtan(x)	Альтернативный синтаксис для atan(x)	EasyEDA	E, I, B
limit(x, L, U)	Значение x, ограниченное через L и U	EasyEDA	E, I, B
ln(x)	Натуральный логарифм x. Сбой с ошибками для отрицательного x. Использовать log(x) вместо.	ngspice	E, I, B
log(x)	Натуральный логарифм x. Генерирует действительное значение выхода для всех x, ограниченных до примерно минимума -230.5 для x< = 1e-100.	EasyEDA	E, I, B
log(x)	Логарифм по основанию 10. Генерирует действительное значение выхода для всех x,	EasyEDA	E, I, B

	ограниченных до минимума -100 для x <= 1e-100.		
max(x,y)	Наибольшее из x и y	ngspice	E, I, B
min(x,y)	Наименьшее из x и y	ngspice	E, I, B
pow(x,a)	Действительная часть x в степени a. Ноль для отрицательного x и дробного a.	EasyEDA	E, I, B
pwr(x,a)	Абсолютное значение x в степени a.	EasyEDA	E, I, B
pwrs(x,a)	pwr(x) умноженное на знак x	EasyEDA	E, I, B
sgn(x)	Знак x. Возвращает -1 для x < 0, 0 для x == 0 (где == означает «точно равно») и 1 для x > 0.	ngspice	E, I, B
sin(x)	Синус x	ngspice	E, I, B
sinh(x)	Гиперболический синус x	ngspice	E, I, B
softlim(ip, lo, hi, sharp)	Значение ip, ограниченное lo и hi с обрезанием перехода между линейной и ограниченной областями, определёнными через «sharp».	EasyEDA	E, I, B
sqr(x)	Квадрат x	EasyEDA	E, I, B
sqrt(x)	Действительная часть квадратного корня из x. Ноль для отрицательных x.	EasyEDA	E, I, B
stp(x)	Альтернативный синтаксис для u(x)	EasyEDA	E, I, B
tan(x)	Тангенс x	ngspice	E, I, B
tanh(x)	Гиперболический тангенс x	ngspice	E, I, B
u(x)	Единичный шаг, т.е., 1, если x > 0, иначе 0.	EasyEDA	E, I, B
u2(x)	Возвращает 1 для x >= 1, x для x между 0 и 1, 0 для x <= 0.	EasyEDA	E, I, B
uramp(x)	X, если x > 0, иначе 0.	EasyEDA	E, I, B

Функции, определённые пользователем

Может быть много случаев, когда функция может потребоваться в разных местах схемы, или она полезна для нескольких разных схем. Чтобы сохранить возможность копировать и вставлять полные выражения, как текстовый блок, всякий раз, когда это нужно, есть выражение **.func**, делающее возможным создание функции, определённой пользователем.

Синтаксис выражения .func очень простой:

.func myfunctionname(a,b,c, ...n) {expression of functions of a, b, c ... n}

Например:

```
.func hypotenuse(x,y) {sqrt(x^2+y^2)}
```

определяет функцию, которая вычисляет длину гипотенузы прямоугольного треугольника с длиной сторон x и y.

Когда функция так определена в схеме проекта, она может быть использована в любом месте любой схемы этого проекта, то есть, она не должна определяться отдельно на каждом листе, где она используется в проекте. Однако она должна определятся на странице каждого проекта, в котором предполагается её использовать, но, если она реально полезная функция, дайте нам знать, возможно, мы добавим её к растущему списку предопределённых функций!

Чтобы использовать функцию, всё что требуется, это вставить **hypotenuse(x,y)** туда, где она нужна, и заменить «x» и «y» их значениями. Так что, например, для использования функции в параметрическом выражении:

```
.param a=3 b=4 hypot=hypotenuse(a,b)
```

или для выходного тока B Source, произведённого двумя напряжениями, V(oneside) и V(otherside):

```
I=hypotenuse(V(oneside), V(otherside))
```

Есть много примеров определения функций выражением .func при симуляции, связанным с тем, что в таблице выше, и во всех EasyEDA spice netlists для автоматического применения предопределённых функций.

Заметьте, что когда используются в выражениях .func, выражения могут занимать более одной строки с использованием символа продолжения «+».

Есть несколько примеров этого в симуляциях, связанных с тем, что в таблице выше, и во всех EasyEDA spice netlists для автоматического применения предопределённых функций. Например:

```
.func POW(x,a)
+ {(((a-(int(a)))==0)||(sgn(x)>=0))_( max(exp(ln(uramp(x))_a),0) +
+ (2_(0.5-ABS((int(a))-2_int(a/2))))_max(exp(ln(uramp(-1_x))*a),0) )}
```

может быть применено к каждому EasyEDA spice netlist.

B sources spice симуляция

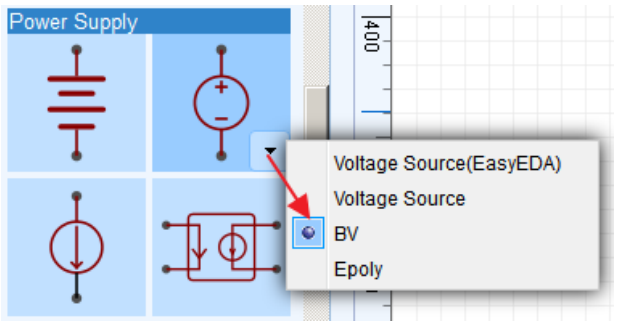
B sources – это один из наиболее мощных компонентов в EasyEDA. Его можно использовать и как BV voltage source и как BI current source (хотя фактически на уровне spice netlist они являются теми же самыми устройствами, только определёнными для получения выходного напряжения или тока).

Функция для каждого B source определяется уравнением.

Левая часть уравнения определяет, будет ли выход источником напряжения или тока.

Правая сторона – это выражение, созданное из чисел, базовых арифметических операторов и функций не только параметров, но, что важно, из динамических напряжений и токов, появляющихся при симуляции схемы. Другими словами, B sources могут выполнять ряд функций при симуляции, которые ограничены только воображением разработчика симуляции схемы.

Как добавить BV source в EasyEDA



Синтаксис уравнения для определения BV source (источника напряжения) в EasyEDA очень прост:

V=expression

Например:

```
V=3*V(a,b)
```

определяет BV source, который генерирует выходное напряжение эквивалентное трёхкратной разности между напряжением на соединении «a» (V(a)) и напряжением на соединении «b» (V(b)).

```
V=scale*uramp(V(a,b))/ABS(I(Vimon))
```

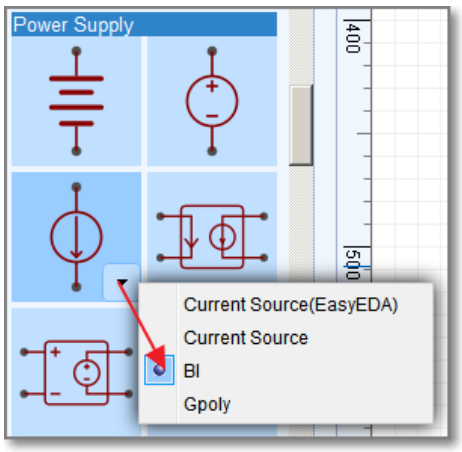
определяет BV source, который генерирует выходное напряжение равное параметру scale, умноженному на положительную разность между напряжением на соединении «a» (V(a))и напряжением на соединении «b» (V(b)), делённую на абсолютное значение тока через 0V source Vimon (I(Vimon)).

```
V=Vswing_tanh(V(a,b)_Avol)
```

определяет дифференциальное усиление блока с малосигнальным усилением Avol и выходным размахом напряжения, ограниченным функцией tanh до +/-Vswing.

Синтаксис уравнения, определяющего BI source в EasyEDA также прост.

Как добавить BI source в EasyEDA



I=expression

```
I=V(a)*I(Vimon)
```

определяет BI source, который генерирует выходной ток равный напряжению на соединении «a» (V(a)), умноженному на ток через 0V source Vimon (I(Vimon)).

```
I=LIMIT(V(a), 3, minval^2)
```

определяет BI source, который генерирует выходной ток равный напряжению на соединении «a» (V(a)), но сжатый до значения 3 и квадрату значения minval параметра для всех значений V(a) вне диапазона, определённого через 3 и minval^2.

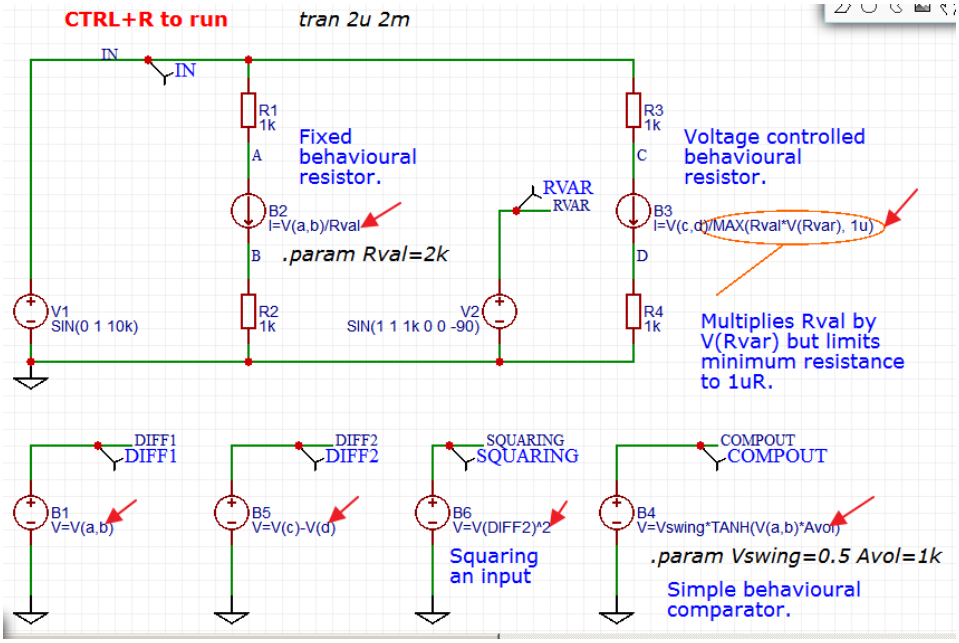
```
I=V(a,b)/Rval
```

когда «-» и «+» выводы B source названы «a» и «b» соответственно, тогда это выражение определяет резистор со значением Rval.

Заметьте, что фигурные скобки не используются в выражениях для B Sources.

Есть несколько примеров использования B Sources в следующих симуляциях:

1. B Sources 01 example->



2. limit-x-L-U

3. Parameters, expressions, functions and B Sources

Заметьте, что когда оно введено непосредственно в поле значения B Source на схеме, выражение ДОЛЖНО быть на единственной строке. Когда используются подобным образом, они не могут располагаться более, чем на одной строке.

Однако выражения, введённые в netlist, такие как в определении .subckt model, могут занимать более одной строки, если использовать символ продолжения «+». Несколько примеров этого можно найти, если просмотреть netlists схем, использующих некоторые EasyEDA .subckt models. Например, в netlist для opamp5pEE. Параметризованная 5и-выводная модель операционного усилителя, где есть эти B sources:

```
Bipbias1 inp isum I=(ibias+ios)*V(supply_ok)
+ ( uramp(V(inn)-(V(vp)+inmax)) - uramp(-V(inn)+(V(vn)-inmin)) )/Rser
```

и

```
Bstg1 0 stage1 I=Islew_tanh(V(indiff)_Kg)
+ - ( uramp(V(stage1)-(V(vp)-outhi)) - uramp(-V(stage1)+(V(vn)+outlo)) )/Rser
+ - sel*( uramp(V(stage1)-(V(out)+ooooclmphi)) - uramp(-V(stage1)+(V(out)-
oooclmplo)) )/Rser
```

и в opamp_ANF01 .subckt примере, найденном в другом месте этого документа, есть ещё пример:

```
B1 out 0
+ V=(TANH((V(inp)-V(inn))_{Avol}_2/(V(vcc)-V(vee)))*(V(vcc)-V(vee))
+ +(V(vcc)+V(vee)))/2
```

Модели устройств

В плане симуляции поведения индивидуальных компонентов они могут быть описаны математически. Базовые уравнения, описывающие поведение компонентов, написаны внутри программы симулятора (иногда они могут быть добавлены пользователем).

Уравнения, которые описывают базовые компоненты, такие как резисторы (I = V/R), конденсаторы (I = C_dV/dt) и индуктивности (V = L_di/dt), могут быть достаточно простыми. Уравнения, которые описывают диоды, биполярные транзисторы (bjt) и разнообразные полевые транзисторы (jfet) и MOS транзисторы, становятся возрастающе сложными, иногда несколько уравнений описывают поведение разных аспектов работы устройства при разных операциях.

Поскольку эти системы уравнений более всего базируются на физике полупроводников и процессе их производства, для некоторых групп устройств, таких как MOSFET, разные системы уравнений могут использоваться для описания моделей одного класса. Разные системы уравнений могут использоваться и потому, что производитель предпочитает описывать операции собственных устройств с большей или меньшей степенью точности.

Хотя сами уравнения глубоко скрыты в исходном коде симулятора, в целом показатели систем уравнений собраны вместе в форме списка. Отдельные устройства любых классов обычных устройств могут описываться в списке показателей.

Подобный список показателей называют **model** .

Отдельные показатели в модели называются **model parameters** (параметры модели).

Модель устройства, написанная подобным образом, называется **.model** statement (выражение модели).

Некоторые устройства, такие как Thyristors (тиристоры), opamps (операционные усилители), linear regulators (линейные стабилизаторы) и микросхемы импульсных источников питания, созданы из некоторого набора других устройств, соединённых вместе в форме подсхем.

Spice netlist устройства, определённого подсхемой, также относится к **model**.

Модель устройства, написанная подобным образом, называется **.subckt**.

Подсхемы моделей могут сами содержать .model statements.

Подсхемы могут также содержать параметры и могут иметь параметры, которые им передаются, чтобы изменить характеристики, например, чтобы адаптировать их к конкретному варианту устройства.

Почему есть разные модели для одного устройства?

Поскольку каждый класс устройств (resistors, diodes, bjts, jfets, MOSFETs и т.д.) описывается одним или более уравнений, каждый класс имеет одну или более моделей, применимых к ним.

Одна из причин существования разных моделей, доступных для устройств одного класса, в том, что производители дают модели устройств бесплатно. Таким образом, они не хотят тратить слишком много времени на разработку моделей устройств, больше, чем им нужно. В основном, чем сложнее модель, тем больше времени нужно производителю, чтобы провести нужные

измерения для получения параметров модели. В итоге, если они чувствуют, что устройство может быть адекватно описано простой моделью, тогда они используют её вместо более аккуратной, но более сложной и трудоёмкой модели.

Другая причина того, что существует разница между моделями одного и того же устройства, заключается в том, что есть разница в процессе производства полупроводников у разных производителей.

Аналогичным образом, почему может быть доступно более одной .model для устройства? Потому что могут быть разные .subckt, определяющие модели.

Может быть разница между .subckt моделей из-за того, что есть разница в реализации моделей устройств и/или физических устройств у разных производителей. Например, есть небольшая разница во внутренней синхронизации и даже во внутреннем устройстве блока осцилляции в группе UC384x контроллеров SMPS между разными производителями.

Иногда есть разница в моделях из-за защиты прав производителя. Некоторые различия могут появиться из-за оптимизации модели для конкретного симулятора, а некоторые детали просто могут быть опущены из-за предпочтений автора.

.model statements

В spice netlist схемы пользователь может найти модели, перечисленные в .model statements. Когда схема сохраняется, эти выражения .model укладываются в netlist EasyEDA, распознающей символы и их связь с именами устройств, данных в схеме. Каждая модель либо добавляется в форме библиотеки, либо, для устройств, которых нет в библиотеках EasyEDA, загрузкой модели от производителя с его web-сайта, с последующей ручной вставкой её непосредственно в схему (то, как это осуществляется будет описано позже).

Типы моделей Ngspice

Чтобы помочь в идентификации типов моделей, а особенно если они для устройств типа N или P, следующая таблица типов моделей может оказаться полезна.

Код	Тип модели
R	Модель полупроводникового резистора
C	Модель полупроводникового конденсатора
L	Модель индуктивности
SW	Управляемый напряжением выключатель
CSW	Управляемый током выключатель
URC	Модель равномерно распределённых RC
LTRA	Модель линии передачи с потерями
D	Модель диода

NPN	Модель NPN BJT
PNP	Модель PNP BJT
NJF	Модель N-канального JFET
PJF	Модель P-канального JFET
NMOS	Модель N-канального MOSFET
PMOS	Модель P-канального MOSFET
NMF	Модель N-канального MESFET
PMF	Модель P-канального MESFET

Хотя это выходит за рамки данного документа, входить в детали, есть ряд других моментов, касающихся моделей, которые следует упомянуть.

- Модели для базовых резисторов, конденсаторов и индуктивностей, использованных в схеме, обычно не показаны в netlist.
- Некоторые модели устройств имеют полный список параметров, другие могут иметь частично заполненный список. Пропущенные параметры в моделях просто замещаются значениями по умолчанию.
- Разные симуляторы поддерживают разный набор моделей, так что в некоторых случаях симулятор может предупреждать пользователя, что какие-то параметры не распознаны и игнорируются. Это, как правило, мало сказывается на результатах симуляции, но если пользователь частично заинтересован в их влиянии, тогда единственный путь – изменить использованный симулятор на тот, что поддерживает все нужные параметры.

.subckt определения

Не все устройства описываются через .model statements.

Модели более сложных устройств, таких как Thyristor (SCR, Triac и также Diac), Insulated Gate Bipolar Transistors (IGBT, биполярные транзисторы с изолированным затвором), операционные усилители (opamps) и даже многие MOSFET, часто сделаны соединением устройств более низкого уровня, чтобы создать схему, которая хорошо отвечает поведению нужного устройства. Это называется **subcircuit** (подсхема). Spice netlist этой подсхемы затем используется для создания такого типа модели, который определяется так называемым **.subckt**. Низкоуровневые компоненты в подсхеме описываются той же разновидностью моделей (списком параметров или показателями), что и для базовых диодов и т.д., уже была ссылка на это. .subckt часто содержит список выражений .model, описывающий устройства, которые используются для создания самой .subckt. Сложные .subckt могут даже вызвать другие .subckt.

Поведенческие модели

Используя Behavioural Voltage и Current Sources (поведенческие источники) и выражения, можно создать то, что называется **behavioural models** (поведенческие модели) компонентов. Есть модели, которые ведут себя как устройства, но которые имеют мало общего с базовыми реальными схемами и, в основном, или, возможно, полностью, описываются исключительно определёнными выражениями (уравнениями). Модели большинства устройств внутри содержат более одного активного компонента, то есть, IC, являясь более всего поведенческими. Это способ скрыть детальную информацию о процессе и технологии производства, которая открывается на низкоуровневом spice моделировании.

Использование выражений и поведенческих источников в EasyEDA поясняется в книге позже.

Что делать, если нет доступной модели устройства?

Не все устройства имеют spice-модели, которые можно запустить в ngspice. И есть ряд причин для этого.

1. Некоторые модели зашифрованы и могут работать только с отдельными коммерческими симуляторами.
2. Некоторые коммерческие симуляторы поддерживают модели, которые недоступны в ngspice.
3. Некоторые устройства имеют модели, которые работают только со специфическими не основанными на spice симуляторами, и которые по этой причине не могут быть преобразованы в spice-модели.
4. Некоторые устройства не имеют доступных опубликованных моделей.
5. Многие устройства, предварившие создание оригинальных spice программ, не имеют моделей.
6. Моделей для некоторых устройств просто нет в природе, поскольку производители их не создавали.
7. Некоторые модели могут быть недоступны в EasyEDA, поскольку они удалены из-за защиты авторских прав для лицензий конечных пользователей, так что, они могут работать только с некоторыми коммерческими симуляторами или не могут распространяться публично.

Для случаев от (1) до (3) нет способа заставить их работать в ngspice. Они должны запускаться в симуляторах, для которых были написаны.

Для случаев от (4) до (7) иной раз можно найти эквивалент, альтернативу или похожее устройство, для которого spice модель доступна. Пользователь должен проявлять осторожность и использовать свои суждения при решении, если замена даёт удовлетворительные результаты симуляции.

Следует отметить, что spice не было оригинально написано с поддержкой для вакуумных устройств (ламп или трубок), так что, модели для таких устройств существуют только в форме

.subckt. Обычно они создаются энтузиастами, а не производителями, так что их, порой, (a) трудно найти и (b) их следует использовать с осторожностью. EasyEDA имеет библиотеку моделей ламп, полученных из источников, модели у которых, мы верим, были написаны достаточно точно.

Заметьте, что модели, полученные от производителей, часто ограничены авторскими правами. Пожалуйста, уважайте любые примечания copyright, содержащиеся либо в соглашении о лицензии для конечного пользователя, которое следует принять перед загрузкой копии модели, либо в самой модели.

Аналогично, модели содержащиеся в библиотеках коммерческих симуляторов – это предмет copyright ограничений.

Часто можно найти модели устройств, предлагаемых на форумах, в дискуссионных группах и разных online коллекциях моделей. Вновь, пользователь должен быть внимателен и полагаться на свои суждения о применимости этих моделей. Часто невозможно выяснить, откуда они взялись, так что, трудно проверить их правильность. Также, возможно, что такие модели скопированы без соблюдения оригинальных авторских прав.

Взаимоотношение между spice-моделями и справочными данными устройств

Хотя некоторые модели устройств в EasyEDA были написаны специально для того, чтобы пользователи могли легко адаптировать их к симуляции ряда устройств, редактируя параметры, которые можно найти непосредственно в – или вывести из – справочных данных (datasheet) устройства (см.: «О взаимоотношении между spice-моделями и поведением в реальном мире» ниже), большая часть из них является моделями «с полки» от производителей устройств.

Важно понимать, что для многих из этих моделей «с полки» базовые уравнения и, поэтому, параметры .model и определения .subckt имеют малое отношение к информации, которую даёт типовая справка к компоненту. Таким образом, обычно невозможно взять datasheet устройства и просто написать модель устройства на основе полученной из справки информации.

Хотя можно извлечь spice параметры для разных устройств и из их справочных данных, и из реальных измерений, описание того, как это сделать, выходит за рамки данного документа.

Больше информации о параметрах моделей, применённых для diodes, bipolar transistors и MOSFET можно найти:

http://www3.imperial.ac.uk/pls/portallive/docs/1/56133736.PDF

с индивидуальным набором слайдов:

http://www3.imperial.ac.uk/pls/portallive/docs/1/7292571.PDF

http://www3.imperial.ac.uk/pls/portallive/docs/1/7292572.PDF

http://www3.imperial.ac.uk/pls/portallive/docs/1/7292573.PDF

Для более детального ознакомления с bjt, в частности, посмотрите книгу:

Modelling the Bipolar Transistor by Ian Getreu

доступную на:

http://www.lulu.com/spotlight/iangetreu

и

http://www.amazon.com/Modeling-Bipolar-Transistor-Ian-Getreu/dp/B000EYPQLU

Другая прекрасная (и свободная) книга о моделировании транзисторов доступна:

http://www.aeng.com/spice_modeling.htm

и, чтобы получить её, зарегистрируйтесь:

Definitive Handbook of Transistor Modeling

Больше информации о ngpsice доступно здесь:

http://ngspice.sourceforge.net/presentation.html

Больше информации о Larry Nagel и SPICE доступно:

http://www.omega-enterprises.net/The%20Origins%20of%20SPICE.html

Larry's PhD диссертация:

Laurence W. Nagel., "SPICE2: A Computer Program to Simulate Semiconductor Circuits,"

*Memorandum No. ERL-M520, University of California, Berkeley, May 1975.

http://www.eecs.berkeley.edu/Pubs/TechRpts/1975/ERL-520.pdf

хорошо читается и поучительна.

Больше информации о симуляции электрических цепей и spice частично здесь:

http://en.wikipedia.org/wiki/Electronic_circuit_simulation

и:

http://en.wikipedia.org/wiki/SPICE

Взаимоотношение между spice-моделями и реальным поведением

Не все spice-модели создаются одинаково. Вот только несколько вещей, относительно которых следует быть в курсе.

Модели одного и того же устройства от разных производителей могут отличаться разной степенью точности. Иногда модели остаются простыми, чтобы ускорить симуляцию за счёт точности. Иногда они сложны, поскольку точность предпочтительнее скорости моделирования. Модели могут содержать некоторый текст в самом начале, который описывает некоторые из её ограничений или её особенности. Часто бывает полезно прочитать эту информацию, поскольку она может помочь в улучшении сходимости симуляции при использовании модели.

Не все модели диодов симулируют обратное напряжение пробоя.

Модели стабилитрона (Zener diode) могут быть разной точности, и лучше всего протестировать стабилитрон, чтобы получить график, который можно сравнить с datasheet. Стабилитроны иногда используют в качестве источника белого шума. Модели стабилитрона не генерируют ни точного уровня, ни спектра шума, наблюдаемого у реальных устройств.

Модели bjt не моделируют пробивное обратное напряжение база-эмиттер. Очень немногие модели поддерживают пробивное напряжение коллектор-эмиттер или коллектор-база.

Некоторые модели, особенно высокоскоростных и высокочастотных устройств, могут включать паразитные параметры, такие как паразитные индуктивность и ёмкость. Такие модели всегда имеют .subckt в определении устройства, заданного выражением .model, но с паразитными элементами, подключёнными из subckt. Если высокочастотное поведение не столь важно, скорость симуляции можно увеличить, используя только выражение .model без паразитных элементов. Эта .model может быть вырезана и вставлена без определения .subckt, но часто собственную .model для транзистора можно найти где-нибудь ещё на сайте производителя или что-то похожее у другого поставщика.

Модели Thyristor и Triac могут быть разной степени точности или моделировать только ограниченный набор параметров устройства.

EasyEDA имеет домашнюю поведенческую макромодель Thyristor и поведенческую макромодель Triac.

Насколько это возможно, домашние модели от EasyEDA для Thyristor и Triac моделируют почти все параметры из datasheet целевого устройства, но есть исключение для di/dt поведения с индуктивной нагрузкой. Эти устройства могут адаптироваться к модели, хотя любые устройства могут просто использовать значения, полученные из справочных данных нужного устройства.

Metal Oxide Varistors (MOV, металл-оксидные варисторы) являются ночным кошмаром для моделирования, и их лучше избегать! Даже модели коммерческого происхождения, порой, работают ненадёжно в оговорённых условиях.

Некоторые модели opamp (операционные усилители) очень подробны, и могут быть весьма точны, но следует обратить внимание на их синтаксическую совместимость с ngspice. Устройства адаптированные для некоторых коммерческих симуляторов не будут работать в ngspice без дополнительной синтаксической корректировки. Другие могут потребовать специальных **.option**s, чтобы работать с симулятором.

И обратите внимание, что некоторые, даже очень сложные модели opamp, не симулируют потребляемый ток хотя бы в виде простого DC тока покоя, не говоря уже о динамическом поведении с токовой нагрузкой. Это может стать преимуществом, поскольку уменьшает токовые сигналы, которые должны моделироваться. Также это означает, что нет смысла включать в цепи питания элементы развязки для этих устройств, поскольку модель не потребляет ток: она использует напряжение питания для определения диапазона синфазных сигналов или размаха выходного напряжения.

Вот пример модели ОУ стороннего производителя, которая не моделирует токи питания или выхода:

LM108 test jig

Некоторые модели opamp могут не предпринимать попыток точно моделировать поведение выходного каскада под действием токовой нагрузки. Аналогично, многие модели устройств не симулируют поведение входов и выходов, когда они оказываются выше или ниже питающего напряжения.

Несколько моделей устройств симулируют чрезмерный питающий ток при отключённом реверсном питании или при правильно включённом устройстве, но с питающим напряжением выше заявленного абсолютного максимально допустимого.

Есть несколько моделей устройств в EasyEDA, специально написанных для воспроизведения поведения реальных устройств, которые они моделируют.

Например, EasyEDA домашняя opamp поведенческая макромодель может быть настроена так, чтобы размах выходного напряжения был почти от шины до шины, но его можно настроить и на более ограниченный размах выходного напряжения, которое может быть и ассиметричным.

Входное сопротивление, смещение и ток смещения, входное напряжение смещения – всё это моделируется.

Моделируется входной разностный и синфазный режимы напряжения.

Поведение тока потребления устройства, если вход или выход оказываются за верхним или нижним пределом питания, или если питающее напряжение подключено реверсно, всё это моделируется. Полярность выходного напряжения из-за перегрузки по входу моделируется для устройств, показывающих такое поведение.

Моделируется зависимость частоты от общего режима и уменьшения питающего напряжения.

В настоящее время не моделируются эффекты температурной зависимости и шумов.

EasyEDA имеет поведенческие макромодели, которые могут быть адаптированы к моделям в широком спектре линейных 3х-выводных и постоянных, и настраиваемых, как положительного, так и отрицательного напряжения, стабилизаторов, которые функционируют подобно моделям ОУ реального мира.

Обо всех домашних моделях EasyEDA вы сможете найти больше информации в определении .subckt, просматривая spice netlist любой сохранённой схемы, где была добавлена эта модель.

Как изменить модель, прикреплённую к символу

Заметьте, пожалуйста, что пред тем, как приступить к редактированию модели, вы должны быть *абсолютно* уверены в своих знаниях относительно того, как связаны между собой spice имена выводов и их нумерация, описанные в разделе «Символы схемы: префиксы и нумерация выводов».

Мы работаем над тем, чтобы предоставить такую функцию для моделей симуляции доступных в EasyEDA.

В настоящем есть ряд методов изменения модели для устройства.

1) Поместите устройство из библиотеки EasyEDA Libs, а затем отредактируйте имя модели устройства либо там, где модель в схеме, либо на панели свойств справа.

Например, когда в схему добавляется NPN bjt (биполярный транзистор npn типа), он появляется с предопределённым именем, пригодным для редактирования, 2DC2412R. Это имя предполагает связь с предопределённой моделью 2DC2412R в spice netlist. Изменение имени с 2DC2412R на 2N2222 приведёт к тому, что модель 2N2222 из библиотеки EasyEDA spice моделей будет помещена в netlist.

Проблема здесь в том, что когда функция поиска модели заработает и запустит этот подход, она может выбрать случайную цель, пропустив все похожие варианты, поскольку нет способа увидеть все доступные для выбора модели.

2) Вторая опция несколько менее удобная, но всегда позволяет запустить в симуляции любую незашифрованную модель устройства. Этот процесс похож и на .model , и на .subckt определение модели.

Этот процесс детально описан в «Связь spice моделей с символами схемы», как части следующего раздела «Символы схемы: префиксы и нумерация выводов», но мы очень советуем прочитать весь раздел полностью для полного понимания детального описания.

Символы схемы: префиксы и нумерация выводов

Заметьте, пожалуйста, что пред тем, как приступить к редактированию модели, вы должны быть абсолютно уверены в своих знаниях относительно того, как связаны между собой spice имена выводов и их нумерация, описанные в этом разделе.

Символы устройств и подсхем (или иерархического блока), созданные для использования в схемах, предназначены к запуску в spice моделировании и дополнительно имеют PCB Prefix (префикс для разводки), который используется для позиционного обозначения в схеме, а также имеют Spice Prefix. Ещё они имеют два набора номеров выводов: PCB выводы и Spice выводы.

PCB и Spice Prefix

Правила присваивания PCB Prefix или позиционного обозначения символа на схеме – это то, что зависит от программы EDA и пользовательских предпочтений. В зависимости от того, как устройство представлено графически схемным символом, оно может иметь разные PCB Prefix или позиционные обозначения. Например, единственное отдельное устройство MOSFET может иметь префикс PCB как Q, M или, возможно, TR, тогда как, если оно часть транзисторной сборки, оно может иметь PCB Prefix в виде U или IC.

Правила присваивания Spice Prefix для схемного символа достаточно строгие. Причина этого в том, что Spice Prefix используется для того, чтобы сообщить симулятору, какой элемент схемы представляет этот символ, а потому, какую симулятор должен использовать модель.

Модели симуляции для большинства элементов spice цепей существуют в форме единственной строки с выражением .model, однако некоторые из них могут быть в форме многострочного определения подсхемы, .subckt. Например, некоторые MOSFET могут описываться как выражения .model, и в этом случае их Spice Prefix будет M, но многие MOSFET определяются как .subckt, и поэтому их Spice Prefix будет X.

Таким образом, независимо от выбора PCB Prefix для символа схемы, Spice Prefix для него представляет данный элемент схемы и должен совпадать с типом модели, требуемой для симуляции этого представителя элемента в вашей схеме.

Например, если есть два разных n-канальных экземпляра MOSFET в схеме, Q1 как BSS123 представленного выражением .model:

```
*SRC=BSS123;DI_BSS123;MOSFETs N;Enh;100V 0.170A 1.00ohms Diodes Inc. MOSFET
.MODEL DI_BSS123 NMOS( LEVEL=1 VTO=1.00 KP=6.37m GAMMA=1.24
+ PHI=.75 LAMBDA=625u RD=0.140 RS=0.140
+ IS=85.0f PB=0.800 MJ=0.460 CBD=19.8p
+ CBS=23.7p CGSO=36.0n CGDO=30.0n CGBO=124n
* -- Assumes default L=100U W=100U --
```

и Q2 как BSS127S, который моделируется с помощью .subckt:

```
*---------- BSS127S Spice Model ----------
.SUBCKT BSS127S 10 20 30
* TERMINALS: D G S M1 1 2 3 3 NMOS L = 1E-006 W = 1E-006
RD 10 1 84.22
RS 30 3 0.001
RG 20 2 29
CGS 2 3 1.958E-011
EGD 12 0 2 1 1
VFB 14 0 0
FFB 2 1 VFB 1
CGD 13 14 2E-011
R1 13 0 1
D1 12 13
DLIM DDG 15 14
DCGD R2 12 15 1
D2 15 0 DLIM
DSD 3 10 DSUB
.MODEL NMOS NMOS LEVEL = 3 VMAX = 8E+005 ETA = 1E-012 VTO = 3.419
+ TOX = 6E-008 NSUB = 1E+016 KP = 0.127 U0 = 400 KAPPA = 1.044E-015
.MODEL DCGD D CJO = 1.135E-011 VJ = 0.9232 M = 0.9816
.MODEL DSUB D IS = 2.294E-010 N = 1.601 RS = 0.1079 BV = 65
+ CJO = 1.956E-011 VJ = 1.514 M = 0.8171
.MODEL DLIM D IS = 0.0001
.ENDS
*Diodes BSS127S Spice Model v1.0 Last Revised 2012/6/6
```

тогда даже тот факт, что оба имеют одинаковый PCB Prefix Q: Q1 должен иметь Spice Prefix M, а Q2 должен иметь Spice Prefix X.

Список Spice Prefixes и связанных с ними элементов схемы дан в таблице ниже.

Описание элемента	Spice Prefix	Комментарий
A	XSPICE code model	analogue, digital, mixed signal
B	Behavioural (arbitrary) source	
C	Capacitor	
D	Diode	
E	Voltage-controlled voltage source (VCVS)	linear, non-linear
F	Current-controlled current source (CCCS)	linear
G	Voltage-controlled current source (VCCS)	linear, non-linear
H	Current-controlled voltage source (CCVS)	linear
I	Current source	
J	Junction field effect transistor (JFET)	spice pin order: D G S
K	Coupled (Mutual) Inductors	
L	Inductor	
M	Metal oxide field effect transistor (MOSFET)	spice pin order: D G S
N	Numerical device for GSS	
O	Lossy transmission line	
P	Coupled multiconductor line (CPL)	
Q	Bipolar junction transistor (BJT)	spice pin order: C B E
R	Resistor	
S	Switch (voltage-controlled)	
T	Lossless transmission line	
U	Uniformly distributed RC line	
V	Voltage source	
W	Switch (current-controlled)	
X	Subcircuit	spice pin order: depends on subckt

Y	Single lossy transmission line (TXL)	
Z	Metal semiconductor field effect transistor (MESFET)	spice pin order: D G S

За большей информацией по элементам схем в Ngspice, пожалуйста, обратитесь к:

http://ngspice.sourceforge.net/docs/ngspice-manual.pdf#subsection.2.1.2

PCB и Spice номера выводов

Есть два набора номеров выводов:

1. PCB pin number: это нумерация выводов реального физического устройства в корпусе. Она требуется, поскольку выводы устройства на схеме могут быть востребованы для площадок под выводы при разводке печатной платы. Другими словами, чтобы соединения на схеме были правильно повторены медными дорожками палаты (PCB).

2. Spice pin number или pin order: это нумерация, которая представляет выводы символа для функционирования в spice модели или подсхеме.

В реальности порядок выводов в spice (pin ordering) имеет более глубокий смысл.

В Spice нет такого понятия, как символ компонента: это конструкция редактора схем.

Когда генерируется spice netlist, символ в редакторе схем либо – в случае определения устройства моделью, как для резисторов, конденсаторов, индуктивностей, диодов, транзисторов и источников – отображается непосредственно соответствующей моделью (определённой префиксом устройства, таким как R, C, L, D, Q и т.д.), либо в случае подсхемы преобразуется в вызов подсхемы.

Spice порядок выводов для подавляющего большинства встроенных моделей, таких как резисторы, конденсаторы, индуктивности, диоды, транзисторы и источники, определен и обычно становится заботой редактора схем, однако больше внимания должно быть уделено spice порядку выводов подсхем.

Это можно проиллюстрировать примером простого операционного усилителя с 5 выводами: инвертирующий и неинвертирующий входы, выход и выводы положительного и отрицательного напряжения питания, но принцип применим ко всем spice подсхемам.

Вызов подсхемы для этого операционного усилителя может выглядеть похожим на этот в spice netlist:

```
X1 input feedback vpos vneg output opamp_ANF01
```

где:

X1 – это имя подсхемы верхнего уровня (вызываемой) схемы;

input feedback vpos vneg output – это имена соединений в вызываемой схеме (содержимое) подсхемы и

opamp_ANF01 – это имя подсхемы, которая вызывается.

- Обратите особое внимание на порядок имён соединений в вызываемой подсхеме.

Spice порядок выводов для большинства подсхем opamp похож на тот, что показан в примере ниже:

```
******************************
* opamp_ANF01 *
* Simplified behavioural opamp
* Node assignments
*                     noninverting input
*                     | inverting input
*                     | | positive supply
*                     | | | negative supply
*                     | | | | output
*                     | | | | |
* spice pin order:    1   2   3   4   5
*                     |   |   |   |   |
.subckt opamp_ANF01 inp inn vcc vee out ; these are the netnames
* used internally to the subcircuit.
*
B1 out 0 + V=(TANH((V(inp)-V(inn))*{Avol}*2/(V(vcc)-V(vee)))*(V(vcc)-V(vee))
+ +(V(vcc)+V(vee)))/2
.ends opamp_ANF01
******************************
```

Заметьте, что spice pin order (порядок выводов) вызываемой подсхемы – это точно тот же порядок, что и в подсхеме.

Хотя физическая нумерация выводов любого устройства критична именно для успешного представления выводов символов схемы для контактных площадок под корпус на печатной плате, spice знает только об одном устройстве и не заботится о том, как оно упаковано. Каждый образец любого устройства на spice схеме должен быть нумерован теми же выводами, что представлены в spice модели или подсхеме, без внимания к физическому корпусу устройства.

Таким образом, для физической нумерации выводов по корпусу устройства для четырёх операционных усилителей, упакованных, скажем, в SOIC14 или DIP14 корпус, как показано ниже, при работе с примером подсхемы выше, spice порядок выводов будет:

Opamp A	**PCB Pin Number**	**Spice Pin Order**
OUT	1	5
IN-	2	2
IN+	3	1
V+	4	3
V-	11	4

Opamp B	**PCB Pin Number**	**Spice Pin Order**
OUT	7	5
IN-	6	2
IN+	5	1
V+	4	3
V-	11	4
Opamp C	**PCB Pin Number**	**Spice Pin Order**
OUT	8	5
IN-	9	2
IN+	10	1
V+	4	3
V-	11	4
Opamp D	**PCB Pin Number**	**Spice Pin Order**
OUT	14	5
IN-	13	2
IN+	12	1
V+	4	3
V-	11	4

Порядок нумерации физического корпуса отражается на каждом операционном усилителе в упаковке.

Spice порядок выводов одинаков для каждого образца отдельного операционного усилителя.

Конечно, есть только один физический образец каждого из выводов питания на схемном символе для четырёх opamp в этом примере, но каждая spice подсхема должна иметь явно определённый вывод.

Именно от того, как это поддержано уровнем схемного символа, зависит то, что при захвате схемы получится с корпусом, имеющим несколько устройств с общими выводами питания, но генерация spice netlist из схемы будет всегда создавать полный набор выводов, требуемых для вызова подсхемы.

В случае, когда подсхема создана пользователем не так, как предусмотрено поставщиком для конкретного устройства, применимы те же самые правила, исключая то, что в задачу пользователя входит определить порядок выводов подсхемы и соответствующим образом построить символ.

Хотя, как написано выше, встроенные spice модели обычно имеют определённый spice порядок выводов, не все подсхемы имеют одинаковую нумерацию. Таким образом, если ваша spice схема выдаёт ошибки, особенно, если есть предупреждение о нумерации выводов или именах выводов, стоит вспомнить о проверке того, что порядок выводов символов в netlist, формирующий выражение вызова, совпадает с тем, что в вызываемой подсхеме!

Связь spice моделей с символами схемы

Для моделей, определённых как .MODEL

1. Найдите spice .model для нужного вам устройства.
2. Скопируйте и вставьте в текстовую метку (горячая клавиша T) на вашей схеме (но, пожалуйста, соблюдайте EULA и copyright коммерческих файлов).
3. На панели свойств справа измените тип текста с comment на spice:

   Properties > Text type > spice
4. Поместите символ устройства из набора библиотеки EasyEDA Libs на схему.
5. Отредактируйте имя модели, чтобы оно совпадало с именем модели в исправленном файле.
6. Готово!

Вот пример этого:

Playing with model parameters

Вот другой пример, показывающий использование обычного режима истощения MOSFET.

Он также показывает способ взломать MOSFET, определённый как LEVEL 3 в выражении .model, но который имеет проблемы с некоторыми параметрами, которые не распознаются в ngspice как часть модели, но всё ещё может использоваться непосредственно с символом MOSFET.

В этом примере L и W параметры оригинальной модели распознаются как часть выражения .model. Заметьте также, что некоторые другие параметры просто не распознаются в ngspice.

Вот, что нужно сделать:

1. Найдите spice .model для нужного вам устройства.
2. Скопируйте и вставьте выражение .model в рабочее поле схемы.
3. Превратите его в spice директиву:

   Properties > Text Attributes > Text type > spice

4. Поместите N-канальный в истощённом режиме MOSFET символ на схему.

5. Отредактируйте атрибут «model» для M1, чтобы включить не распознанные или модифицированные L и W параметры, так чтобы это выглядело как:

```
IXTT20N50D L=2E-6 W=5.5
```

Это может быть сделано либо размещением, либо через:

Part Attributes > Model > IXTT20N50D L=2E-6 W=5.5

Заметьте, что добавление звёздочки в начале двух строк в выражении .model, что определяет L и W параметры (и любые другие нераспознанные параметры по мере необходимости), закомментирует их. Это остановит без объявления этих параметров модели как выходных в сообщении симуляции, но это и не требуется.

Этот процесс иллюстрируется следующим примером:

N channel depletion mode MOSFET using a .model statement

Для моделей, определённых как .SUBCKT

Процесс, описанный выше, работает хорошо для простых моделей, определённых как .model, но для моделей, определённых как .subckt, это немного сложнее, поскольку вам нужно сказать EasyEDA, что модель именно .subckt, а не простая .model.

Заметьте, что даже некоторые модели скромных диодов – это, фактически, определения .subckt для включения таких вещей, как паразитные параметры корпуса. Например, сравните 1N4148 и 1N4148W-V модели в netlist.

Есть три стадии в прикреплении .subckt к символу, которые уже имеет spice prefix «X», и ожидает вызова выражения .subckt.

- Поместите текст .subckt на схему и активизируйте его.
- Поместите символ на схему.
- Измените имя символа на точно то же, что и имя .subckt.

Детальные шаги для связи новой модели .subckt с символом:

1. Найдите spice .SUBCKT для нужного вам устройства.

2. Скопируйте и вставьте в текстовое окно (горячая клавиша T) на вашей схеме (но, пожалуйста, учтите EULA и copyright коммерческих файлов).

3. На правой панели свойств измените тип текста с comment на spice:

   Properties > Text type > spice

4. Поместите символ устройства из набора библиотеки EasyEDA Libs на схему.

5. Отредактируйте имя модели для точного совпадения с именем модели в исправленном файле.

6. Нажмите горячую клавишу **I** или:

Щёлкните по синей кнопке **Edit Symbol...** на панели Properties:

Properties > Edit Symbol...

или сделайте:

Super Menu > Miscellaneous > Edit Symbol

7. В диалоговом окне **Modify your symbol information** отметьте, что **Spice Prefix** это X.

8. Проверьте, что NUMBER выводов в «Edit Pin Map information» точно такое же, что и в .SUBCKT, вставленной в схему: если это не так, то был выбран неправильный символ для выбранной .SUBCKT (или наоборот), тогда должен быть выбран другой символ (или .SUBCKT).

Заметьте, что «number of pins» здесь означает количество выводов, не номера выводов или имена, и использовано для описания соединений, которыми выводы соединяются в .subckt.

9. Проверьте, что ORDER выводов в **Edit Pin Map information** точно такой, что и в .SUBCKT, вставленной в схему. Это может очень запутывать, поскольку NAMES выводов могут различаться у символа и .SUBCKT, так что вначале необходимо примирить два набора имён до того, как подтвердить их порядок.

10. Щёлкните по **OK** в диалоговом окне **Modify your symbol information**.

11. Готово!

Этот процесс иллюстрируется в следующем примере:

Attaching a .subckt to a symbol 01

Некоторые EasyEDA символы из библиотеки EasyEDA Libs имеют Spice Prefix, который ожидает вызова выражения .model (то есть, любой символ с Spice Prefix иным, чем «X»!). Например, bjts символы имеют Spice Prefix «Q», хотя MOSFET символы имеют Spice Prefix «M». Есть некоторые диоды, например, ряд стабилитронов (zener diod), определены через .models, а другие диоды определены как .subckts. Даже скромный 1N4148 может иметь .model от одного производителя и .subckt от другого. Некоторые модели bjt (биполярных транзисторов) определены через .subckts, такие как большинство транзисторов Дарлингтона и некоторые лавинные транзисторы (Avalanche transistor). Кроме части маломощных MOSFET и тех, что используются в разработке IC, большинство моделей MOSFET определены как .subckts. Таким образом, общепринято связывать .subckt с символом через Spice Prefix, который будет вызваться выражением .model.

Это можно сделать в четыре приёма, как иллюстрирует следующий пример: NMOS_E символ размещён в схеме из набора библиотеки EasyEDA Libs; он должен редактироваться, чтобы изменить «Spice Prefix» символа с «M» (для части, которая определена как .model) на «X» (для части определённой как .subckt).

- Поместите .subckt текст на схему и активизируйте его.

- Поместите символ на схему.
- Измените имя символа, сделав его точно таким, как имя .subckt.
- Измените «Spice Prefix» символа с «M» (для .model) на «X» (для .subckt).

Детально шаги по связыванию новой модели с символом, и чтобы сказать EasyEDA, что модель устройства – это .subckt, а не просто .model:

1. Найдите spice .SUBCKT для нужного устройства.
2. Скопируйте и поместите в текстовое окно (горячая клавиша **T**) на вашей схемы (но с учётом EULA и copyright коммерческих файлов).
3. На панели свойств справа измените тип текста с comment на spice.

   Properties > Text type > spice

4. Поместите символ устройства из набора библиотеки EasyEDA Libs на схему.
5. Исправьте имя модели, чтобы оно было точно таким, как во вставленном файле.
6. Нажмите горячую клавишу **I** или:

Щёлкните по синей кнопке **Edit Symbol...** на панели Properties (свойства):

Properties > Edit Symbol...

или сделайте:

Super Menu > Miscellaneous > Edit Symbol

7. В диалоговом окне «**Modify your symbol information**» измените **Spice Prefix** с M (для .model) на X (для .subckt).
8. Проверьте, что NUMBER выводов в **Edit Pin Map information** точно такое же, как в .SUBCKT, вставленной в схему: если это не так, то был выбран неправильный символ для выбранной .SUBCKT (или наоборот), тогда должен быть выбран другой символ (или .SUBCKT).

Заметьте, что «number of pins» здесь означает количество выводов, а не порядок выводов или их имена, использованные для описания соединений, которые они осуществляют в .subckt netlist.

9. Проверьте, что ORDER выводов в **Edit Pin Map information** точно такой, что и в .SUBCKT, вставленной в схему. Это может очень запутывать, поскольку NAMES выводов могут различаться у символа и .SUBCKT, так что вначале необходимо примирить два набора имён до того, как подтвердить их порядок.
10. Щёлкните по **OK** в диалоговом окне **Modify your symbol information.**
11. Готово!

Этот процесс иллюстрирует следующий пример:

Attaching a .subckt to a symbol 02

Другой пример процесса, описанного выше для изменения Spice Prefix символа, иллюстрируется с EasyEDA символом N-канального в режиме истощения MOSFET из библиотеки EasyEDA Libs, что был ранее использован в выражении IXTT20N50D .model. В этом примере MOSFET символ прикреплён к .subckt, которая была создана из оригинального выражения IXTT20N50D .model в порядке упаковки L=2E-6 W=5.5 параметров, что упрощает использование оригинальной модели.

An N-channel depletion mode MOSFET using an EasyEDA .subckt

Привязка моделей к пользовательским символам

В основном это то же, что привязка модели к любому из предопределённых символов из библиотеки EasyEDA Libs за исключением того, что символ тот же, что создавался непосредственно «с нуля» или редактированием существующего символа. Правила те же, что для назначения и проверки того, что spice prefix совпадает с типом модели, к которой прикреплён (то есть, «X» для .subckt или любой Spice Prefix отличный от «X» для .model), и проверки того, что spice pin numbering совпадает с типом устройства, определённого выражением .model, или последовательностью выводов, определённых в .subckt модели.

Пользовательское моделирование

Это обширный раздел, который будет заполнен, когда позволит время.

Однако, просматривая некоторые домашние модели EasyEDA, можно получить достаточно информации, пусть не в части описания «Как это работает», но всё-таки в части определения .subckt, что может стать окном в дикий и прекрасный мир пользовательского моделирования.

Некоторые примеры при просмотре netlist:

LDR test

Electret microphone model

Electret microphone model .subckt

Electret microphone model test jig

LM56EE demo jig

How to include a 5 Pin Comparator in a schematic

И этот проект:

An EasyEDA logic family

Проведение измерений по результатам симуляции

Некоторым образом цифровой запоминающий осциллоскоп (DSO) позволяет измерять сигналы, отображаемые на дисплее, с помощью курсоров, и непосредственно считывая значения в том месте, где это нужно. С помощью математического анализа данных графиков, сохранённых в

памяти DSO, EasyEDA позволяет обрабатывать результаты симуляции как непосредственно с помощью курсоров, так и анализа данных, полученных при создании WaveForm отображения, если использовать команду **meas**.

Использование курсоров отображения WaveForm

WaveForm X и Y функции курсора – это простой и быстрый способ провести измерения в интересующих точках сигналов, а также определить разность между этими точками.

WaveForm позволяет отображать графики на любой выбранной из доступных трёх вертикальных панелей. Ось Y автоматически масштабируется для заполнения в выбранных единицах всего отображаемого диапазона. Графики могут быть скрыты, но хотя бы один из них должен быть виден. Данные графиков X и Y могут быть видны на экране, достаточно провести курсором мышки по области построения диаграммы, при этом по оси Y график будет адаптирован к наилучшему чтению на каждой панели.

Дельта X и дельта Y данных графика могут быть показаны на экране, если использовать Left-Click и протаскивание курсора по области выбора, имеет место и адаптация к наилучшему чтению по оси Y для каждой панели. Возврат курсора в маленькую окружность начальной точки окна выбора, без отпускания Left-Click, вернёт к показаниям данных графика по X и Y.

Произведённое размещение курсора и полученные результаты являются временными, что означает – они не могут копироваться и вставляться. Они не сохраняются как часть файла WaveForm. Однако, используя утилиту снимка экрана, можно сохранить рисунок дисплея WaveForm, показывающий положение курсора и связанные с ним показания.

Заметьте, что утилита снимка экрана должна иметь определяемую пользователем задержку, чтобы позволить переместить курсор между моментами запуска снимка и фактического его выполнения.

Больше информации по отображению результатов симуляции вы найдёте в разделе WaveForm руководства EasyEDA.

Использование команды meas

Команда meas используется для анализа выходных данных dc, ac, tran или fft (или spec) симуляций. Команда выполняется сразу по завершении симуляции.

Значение терминов в команде MEAS

Тип meas **{DC|AC|TRAN|SP}** зависит от данных, которые получены в результате DC, AC analysis, TRANsient или SPectrum analysis (используя fft (или spec) анализ) симуляции.

result будет вектором, содержащим результат измерения.

trig*variable*, **targvariable** и **out_variable** – это векторы, возникающие при симуляции, например, вектор напряжения v(out).

VAL=val ожидается действительное число val. val может также быть действительным параметром или выражением, заключённым в [] или {}, расширенным до действительного числа.

TD=td и **AT=time** ожидается значение времени, если тип meas это **tran**. Для **ac** и **sp**, **AT** будет значением частоты, а **TD** игнорируется.

Для **dc** анализа **AT** – это напряжение (или ток), а **TD** игнорируется.

CROSS=# требует целого числа #. **CROSS=LAST** также допустимо. То же получается при **RISE** и **FALL**.

Значения частоты и времени могут начинаться с 0 и продолжаться до некоторого положительного действительного числа. Напряжение (или ток) входа для независимой (scale) оси при dc анализе может начинаться или заканчиваться произвольным действительным числом.

Примеры форм и синтаксиса команд MEAS

Trig Targ

Команда meas использует Trig Targ синтаксис **General Form 1** при измерении разности dc напряжения, частоты или времени между двумя точками, выбранными по одному или двум выходным векторам. Следующие примеры все используют симуляцию переходного процесса (transient simulation). Измерения при tran анализе стартует после времени td. Если другие примеры запускаются при ac симуляции или спектральном анализе, время может быть замено частотой, хотя после dc симуляции независимая переменная может стать напряжением или током.

General Form 1:

```
MEAS {DC | AC | TRAN | SP} result TRIG trig_variable VAL=val
<TD=td> <CROSS=# | cross="LAST">
<RISE=# | rise="LAST"> <FALL=# | fall="LAST">
<TRIG at="time"> TARG targ_variable VAL=val <TD=td>
<CROSS=# | cross="LAST">
<RISE=# | rise="LAST"> <FALL=# | fall="LAST">
<TARG at="time">
```

Пример выражения измерения:

```
meas tran tdiff TRIG v(1) VAL=0.5 RISE=1 TARG v(1) VAL=0.5 RISE=2
```

измеряет разность времени между напряжением v(1), достигшем 0.5 V от первоначального возрастающего склона (TRIG), до достижения 0.5 V вновь на втором возрастающем склоне (TARG); то есть, измеряется период сигнала.

Выход:

```
tdiff = 1.000000e-003 targ= 1.083343e-003 trig= 8.334295e-005
```

Пример выражения измерения:

```
meas tran tdiff TRIG v(1) VAL=0.5 RISE=1 TARG v(1) VAL=0.5 RISE=3
```

измеряет разность времени между напряжением v(1), достигшем 0.5 V от первоначального возрастающего склона, до достижения 0.5 V на возрастающем склоне, но в третий раз (то есть, два периода).

Пример выражения измерения:

```
meas tran tdiff TRIG v(1) VAL=0.5 RISE=1 TARG v(1) VAL=0.5 FALL=1
```

измеряет разность времени между напряжением v(1), достигшем 0.5V от первоначального возрастающего склона, до достижения 0.5 V на первом спадающем склоне.

Пример выражения измерения:

```
meas tran tdiff TRIG v(1) VAL=0 FALL=3 TARG v(2) VAL=0 FALL=3
```

измеряет разность времени между напряжением v(1), достигшем 0V его третьего спадающего склона, до напряжения v(2), достигшего 0 V на его третьем спадающем склоне.

Пример выражения измерения:

```
meas tran tdiff TRIG v(1) VAL=-0.6 CROSS=1 TARG v(2) VAL=-0.8 CROSS=1
```

измеряет разность времени между напряжением v(1), пересекающем -0.6 V впервые (любой склон), и напряжением v(2), пересекающим -0.8 V впервые (любой склон).

Пример выражения измерения:

```
meas tran tdiff TRIG AT=1m TARG v(2) VAL=-0.8 CROSS=3
```

измеряет разность времени между временной точкой 1ms и временем, когда напряжение v(2) пересекает -0.8 V в третий раз (любой склон).

Find … When

Функция FIND и WHEN позволяет измерять любое зависимое или независимое время, частоту или dc параметр, когда два сигнала пересекаются или сигнал пересекает заданное значение.

Измерение начинается после задержки TD и может быть ограничено диапазоном от FROM до TO.

General Form 2:

```
MEAS {DC | AC | TRAN | SP } result WHEN out_variable=val
<TD= td> <FROM=val> <TO= val>
<CROSS=# | cross="LAST">
<RISE| rise="LAST"> <FALL=# | fall="LAST">
```

Пример выражения измерения:

```
meas tran teval WHEN v(2)=0.7 CROSS=LAST
```

измеряет точку времени, когда напряжение v(2) пересекает 0.7 V в последний раз (любой склон).

General Form 3:

```
MEAS {DC | AC | TRAN | SP } result WHEN out_variable=out_variable2
<TD= td> <FROM= val> <TO= val>
<CROSS=# cross="LAST">
<RISE | rise="LAST"> <FALL=# | fall="LAST">
```

Пример выражения измерения:

```
meas tran teval WHEN v(2)=v(1) RISE=LAST
```

измеряет временную точку, когда напряжения v(2) и v(1) равны, v(2) возрастает в последний раз.

General Form 4:

```
MEAS {DC | AC | TRAN | SP } result FIND out_variable WHEN out_variable2=val
<TD= td> <FROM= val> <TO= val>
<CROSS=# | cross="LAST"> <RISE | rise="LAST">
<FALL=# | fall="LAST">
```

Пример выражения измерения:

```
meas tran yeval FIND v(2) WHEN v(1)=-0.4 FALL=LAST
```

возвращает зависимую переменную (y), нарисованную по v(2) во временной точке, когда v(1) равно значению -0.4, v(1) спадает в последний раз.

General Form 5:

```
MEAS {DC | AC | TRAN | SP } result FIND out_variable WHEN
out_variable2=val=out_variable3
<TD= td> <CROSS=# | cross="LAST"> <RISE | rise="LAST"> <FALL | fall="LAST">
```

Пример выражения измерения:

```
meas tran yeval FIND v(2) WHEN v(1)=v(3) FALL=2
```

возвращает зависимую переменную (y), нарисованную по v(2) во временной точке, когда v(1) пересекает v(3), v(1) спадает во второй раз.

General Form 6:

```
MEAS {DC | AC | TRAN | SP } result FIND out_variable AT=val
```

Пример выражения измерения:

```
meas tran yeval FIND v(2) AT=2m
```

возвращает зависимую переменную (y), нарисованную по v(2) во временной точке 2 ms (заданной AT=time).

AVG | MIN | MAX | PP | RMS | MIN_AT | MAX_AT

General Form 7:

```
MEAS {DC | AC | TRAN | SP } result {AVG | MIN |MAX| PP | RMS | MIN_AT |
MAX_AT} out_variable
<TD= td> <FROM=val> <TO=val>
```

Пример выражения измерения:

```
meas tran ymax MAX v(2) from=2m to=3m
```

возвращает максимальное значение v(2) внутри временного интервала между 2 ms и 3 ms.

Пример выражения измерения:

```
meas tran tymax MAX_AT v(2) from=2m to=3m
```

возвращает временную точку максимального значения v(2) внутри интервала от 2 ms до 3 ms.

Пример выражения измерения:

```
meas tran ypp PP v(1) from=2m to=4m
```

возвращает значение от пика до пика v(1) внутри интервала между 2 ms и 4 ms.

Пример выражения измерения:

```
meas tran yrms RMS v(1) from=2m to=4m
```

возвращает среднеквадратичное значение v(1) внутри интервала от 2 ms до 4 ms.

Пример выражения измерения:

```
meas tran yavg AVG v(1) from=2m to=4m
```

возвращает среднее значение v(1) внутри интервала между 2 ms и 4 ms.

INTEG

Означает выражение с формой INTEG, возвращающей область под out_variable внутри временного интервала, заданного FROM val и TO val.

General Form 8:

```
MEAS {DC | AC | TRAN | SP} result INTEG<RAL> out_variable
<TD=td> <FROM=val> <TO=val>
```

Пример выражения измерения:

```
meas tran yint INTEG v(2) from=2m to=3m
```

возвращает область под v(2) внутри интервала между 2 ms и 3 ms.

DERIV

Пожалуйста, отметьте, что meas {DC|AC|TRAN|SP} результат DERIV<ATIVE> ... всё ещё не доступен в ngspice.

Больше выражений измерения

```
meas tran inv_delay 2 trig v(in) val='vp/2' td=1n fall=1 targ v(out)
val='vp/2' rise=1

meas tran test_data1 trig AT=1n targ v(out) val='vp/2' rise=3

meas tran out_slew trig v(out) val=' 0.2_vp' rise=2 targ v(out) val=' 0.8_vp'
rise=2

meas tran skew when v(out)=0.6

meas tran skew2 when v(out)=skew_meas
```

```
meas tran skew3 when v(out)=skew_meas fall=2

meas tran skew4 when v(out)=skew_meas fall=LAST

meas tran skew5 FIND v(out) AT=2n

meas tran v0_min min i(v0) from='dfall' to='dfall+period'

meas tran v0_avg avg i(v0) from='dfall' to='dfall+period'

meas tran v0_integ integ i(v0) from='dfall' to='dfall+period'

meas tran v0_rms rms i(v0) from='dfall' to='dfall+period'

meas dc is_at FIND i(vs) AT=1

meas dc is_max max i(vs) from=0 to=3.5

meas dc vds_at when i(vs)=0.01

meas ac vout_at FIND v(out) AT=1MEG

meas ac vout_atd FIND vdb (out) AT=1MEG

meas ac vout_max max v(out) from=1 k to=10MEG

meas ac freq_at when v(out)=0.1

meas ac vout_diff trig v(out) val=01 rise=1 targ v(out) val=01 fall=1

meas ac fixed_diff trig AT=10k targ v(out) val=0.1 rise=1

meas ac vout_avg avg v(out) from=10k to=1MEG

meas ac vout_integ integ v(out) from=20k to=500k

meas ac freq_at2 when v(out)=01 fall=LAST

meas ac vout_rms rms v(out) from=10 to=1G
```

Следующие примеры иллюстрируют некоторые измерения, которые могут выполняться таким способом.

Measuring WaveForm parameters 01

Measuring WaveForm parameters 02

Measuring settling time

Find gain and bandwidth