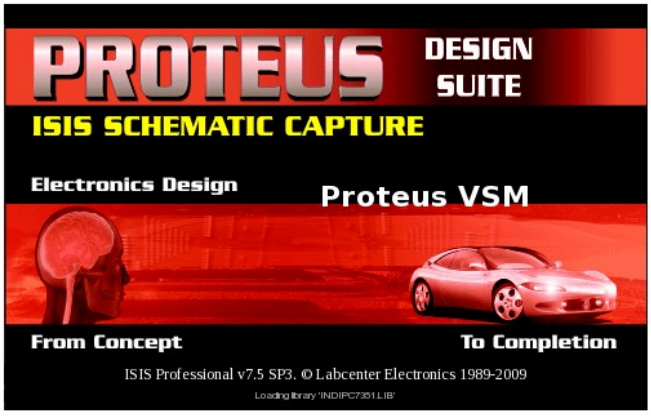


Intelligent Schematic Input System

Руководство пользователя

Адаптировано к версии 2010 г.

Issue 6.0 - November 2002

Оглавление

ВВЕДЕНИЕ

О PROTEUS VSM

Традиционно симуляция схемы была делом не интерактивным. Некогда netlist (спецификация элементов и соединений) делали вручную, и вывод состоял из кучи цифр. Если вам везло, вы получали псевдографический вывод, нарисованный с помощью звёздочек для показа формы напряжения или тока.

Ближе к нашим дням ввод схемы и экранная графика становятся нормой, но процесс симуляции всё ещё остаётся не интерактивным — вы рисуете схему, нажимаете «go», и изучаете результат в какого-нибудь рода пост-процессоре. Это прекрасно, если схема, которую вы проверяете, достаточно статична, не более осциллятора, который в ней есть и генерирует, просто, 1 МГц. Однако, если вы разрабатываете охранную сигнализацию, и хотите найти, что случится, когда будет введён неверный пароль с клавиатуры, требуемые установки становятся довольно непрактичны, и кто-то должен прибегнуть к прототипу. И это — позор, поскольку работа «в кибер-пространстве» может предложить очень много с точки зрения производительности разработки.

Только в образовательных кругах пытались создать симуляцию электрических цепей похожую на работу с электроникой в реальной жизни, когда можно интерактивно взаимодействовать со схемой в процессе симуляции. Проблема в том, что анимация компонентов трудно поддаётся кодированию в программе. Появилось только ограниченное число простых устройств, таких как переключатели, лампочки, электрические моторы и т.п., а это мало используется профессиональными пользователями. Вдобавок, качество симуляции схем часто оставляло желать лучшего. Например, один из ведущих продуктов этого типа не имел временной информации в цифровых моделях.

PROTEUS VSM даёт вам лучшее, комбинируя великолепный симулятор смешанных цепей, основанный на стандарте SPICE3F5, с анимированными моделями компонентов. И он предоставляет архитектуру, в которой дополнительные анимированные модели могут создаваться кем угодно, включая конечных пользователей. Действительно, множество типов анимированных моделей может быть выполнено без обращения к кодированию. Следовательно, PROTEUS VSM позволяет профессиональным инженерам запускать интерактивную симуляцию реальных проектов, и получать в награду результат, подходящий к симуляции схемы. А, кроме того, если этого не достаточно, мы создали ряд моделей симуляции для популярных микроконтроллеров и набор анимированных моделей для периферийных устройств, таких как светодиоды, жидкокристаллические дисплеи, клавиатуры, RS232 терминалы и т.д. Итог — вы можете симулировать полные микроконтроллерные системы, а, следовательно, разрабатывать программы для них без обращения к физическим прототипам. В мире, где время для рынка становится все более и более важным, это реальное преимущество.

Следует отметить, что производительность современных персональных компьютеров, действительно, потрясающая. 300MHz Pentium II PC может симулировать простые микроконтроллерные схемы в реальном времени и в ряде случаев даже быстрее. И в тех случаях, когда скорости не хватает, время реального отклика чаще всего не используется при

программной разработке. Если же вы серьёзно озабочены этим — идите и покупайте 2GHz двух процессорный PC, который, несомненно, много быстрее. Этим полностью развенчивается очевидное возражение против интерактивной симуляции, что она не будет достаточно быстра.

О ДОКУМЕНТАЦИИ

Это руководство предназначено восполнить информацию, предоставляемую в подсказке (on-line help). Руководство содержит дополнительную информацию и примеры, тогда как подсказка предлагает контекстно чувствительную информацию, относящуюся к отдельным иконкам, командам и диалогам. Подсказка для большинства объектов в пользовательском интерфейсе может быть получена после указания курсором и нажатия клавиши F1.

PROTEUS VSM может использоваться двумя путями — либо для интерактивной симуляции, либо для симуляции, основанной на графиках, что и отображено в структуре руководства. Обычно вы используете интерактивную симуляцию, чтобы увидеть, работает ли проект в целом, а графическую симуляцию, когда исследуете, почему нет, или когда нужны детальные измерения. Однако нет надёжных правил. Некоторые элементы системы, такие как генераторы, важны для обоих режимов использования, и имеют отдельные главы по этой причине.

Есть детальные, шаг за шагом руководства, которые позволят вам поупражняться в обоих типах симуляции. Мы очень рекомендуем, чтобы вы поработали с ними, поскольку они показывают все основные техники, требуемые для работы с программой.

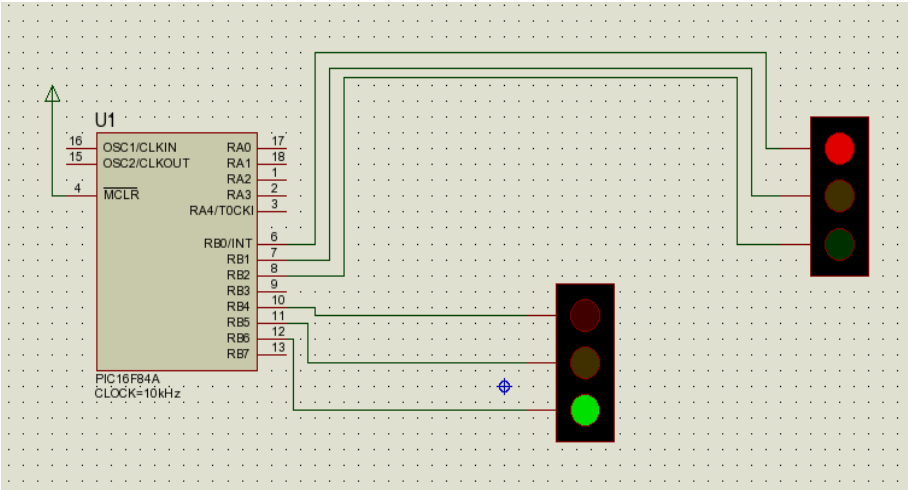
Информация о создании VSM моделей предоставлена отдельно в *VSM Software Development Kit (SDK) на он-лайновом ресурсе, но устанавливается стандартно с профессиональной версией Proteus*.

РУКОВОДСТВО ПО ИНТЕРАКТИВНОЙ СИМУЛЯЦИИ

Введение

Цель этого руководства показать вам, создавая простые схемы, как использовать интерактивную симуляцию в Proteus VSM. Пока мы сосредоточимся на использовании Active Components (активных компонентов) и возможностях отлаживания с помощью ISIS Editor, кроме того коснёмся вопроса основ планирования схем и работы с основными схемами. Полную информацию по этим вопросам можно найти в руководстве к ISIS.

Схема, которую мы используем для симуляции — это пара светофоров, соединённых с микроконтроллером PIC16F84, как показано ниже.



Чтобы не рисовать всю схему сначала, можно найти её в папке примеров «Samples\Tutorials\Traffic.DSN», устанавливаемую вместе с Proteus. Пользователи, которые знакомы с основными процедурами работы в ISIS, могут использовать этот готовый проект и перейти к секции программирования микроконтроллеров. Заметьте, однако, что этот файл проекта содержит преднамеренную ошибку — прочитайте об этом.

Если вы плохо знаете ISIS, интерфейс и основы использования обсуждаются во «Введении» руководства к ISIS. И, хотя мы коснёмся этих вопросов в следующих разделах, постарайтесь выкроить время для знакомства с программой, перед тем как продолжать.

Рисование схем

Размещение компонентов

Мы начнём с размещения двух наборов огней светофоров и PIC16F84 на новом листе схемы. Начните с нового проекта, выберите основной режим и иконку *Component Mode* (все иконки имеют контекстно чувствительную подсказку их назначения), а затем щёлкните по букве «P» над *окном выбранных объектов*. Появится диалог *Device Library* в *окне редактирования* (загляните в руководство ISIS, где больше информации).

Traffic Lights (светофор) можно найти в категории *Miscellaneous*, а PIC микроконтроллер в категории *Microprocessor ICs*. Для выбора компонента для проекта подсветите имя компонента, щёлкнув по его имени в окне *Results*, и нажмите кнопку **ОК**. Переместите курсор в нужное место рабочего поля (курсор похож на карандаш) и щёлкните левой клавишей, пока не появится нужный компонент; он же появится в *окне выбранных компонентов*.

Перемещение и ориентация

У нас появились нужные для схемы блоки, но, возможно, они размещены не совсем удачно. Чтобы переместить компонент, поместите курсор мышки на компонент, щёлкните и, когда рядом с курсором (который стал похож на руку) появится перекрестие, нажмите и удержите левую клавишу мышки, перенося компонент в новое место. К курсору будет «привязан» контур компонента. Обратите внимание, что в этот момент компонент все ещё выделен (подсвечен), щёлкните левой клавишей мышки на любом свободном месте рабочего поля, и компонент приобретёт свой привычный вид.

Для переориентации компонента щёлкните правой клавишей мышки по нему и из выпадающего меню выберите нужную операцию: поворот или отражение. Не забудьте вновь щёлкнуть по свободному месту, чтобы вернуть компонент к нормальному виду.

Потренируйтесь с перемещением и ориентацией компонентов (возможно, используя данный пример). Если у вас появятся проблемы, мы советуем вам обратиться к руководству по ISIS.

Масштабирование

В плане соединения схемы будет полезно иметь возможность менять вид (масштабировать) компонентов к заданной области. Нажатие на клавишу **F6** увеличит вид у текущей позиции курсора или, альтернативно, можно удерживать клавишу **SHIFT** и прорисовать прямоугольник левой клавишей мышки, что увеличит изображение в этой области. Чтобы уменьшить изображение, нажмите клавишу **F7** или, если хотите уменьшить так, чтобы видеть весь чертёж, нажмите клавишу **F8**. Соответствующие команды можно найти в разделе *View* основного меню.

Соединение

Самый простой способ соединять компоненты — это использовать опцию *Wire Auto Router* в разделе *Tools* основного меню. Убедитесь, что эта опция выбрана (иконка слева в меню должна быть «нажата»). Больше информации по работе WAR можно найти в руководстве к ISIS.

Увеличьте PIC пока все его выводы будут удобно видны и затем подведите курсор мышки к концу вывода 6 (RB0/INT). Вы увидите красный квадратик на конце вывода, а курсор станет похож на карандаш. Это означает, что мышка в нужном месте для проведения соединения с этим выводом. Щёлкните левой клавишей мышки, чтобы начать соединение, а затем перемещайте мышку к выводу от красного окна одного из светофоров. Когда на том выводе появится красный квадратик, щёлкните левой клавишей мышки ещё раз, чтобы завершить соединение. Повторите процесс для подключения обоих светофоров, как это сделано на схеме примера.

Есть ряд важных моментов, относящихся к процессу соединения:

- Вы можете проводить соединения в любом режиме — ISIS достаточно сообразительная программа, чтобы определить, что вы делаете.
- Когда выбрана опция *Wire Autorouter*, соединение обходит помехи и, как правило, находит путь между другими соединениями. Что означает для вас — все, что нужно сделать, это два щелчка на двух выводах, и дать возможность ISIS найти путь между этими выводами.
- ISIS автоматически панорамирует экран, если вы пересекаете край рабочего окна при создании соединения. Это означает, что вы можете увеличивать изображение до нужного уровня и, поскольку знаете нужное целевое место, просто проходите по экрану до того момента, когда увидите объект. Альтернативно, вы можете увеличивать и уменьшать изображение, размещая провода (используя клавиши **F6** и **F7**).

И, наконец, мы должны соединить вывод 4 с контактом питания. Щёлкните правой клавишей мышки на свободном месте, выберите из выпадающего меню *Place-Terminal-Power*; теперь щёлкните левой клавишей мышки в подходящем месте для размещения контакта. Определитесь с нужной ориентацией и соедините вывод 4 с контактом питания, как это описано выше.

Больше полезной информации о процессе соединения вы найдёте в руководстве к ISIS.

В этом месте мы рекомендуем загрузить полную версию схемы — это избавит вас от любых недоразумений, если версия, которую нарисовали вы отличается чем-то от нашей!

Написание программы

Листинг исходного кода

Для целей нашего руководства мы приготовили следующую программу, которая позволит PIC управлять светофорами. Эта программа предоставлена в файле, названном TL.ASM и может быть найдена в директории «Samples\Tutorials».

```
        LIST    p=16F84 ; PIC16F844 целевой процессор
        #include "P16F84.INC" ; Включить файл заголовков

        CBLOCK 0x10   ; Временное хранилище
          state
          l1,l2
        ENDC

        org     0               ; Начало вектора.
        goto    setports        ; Переход к началу кода.

          org     4             ; Вектор прерывания.
halt      goto    halt          ; Задаём бесконечный цикл и ничего не делаем.

setports  clrw                  ; Ноль в W.
        movwf   PORTA           ; Убедимся, что PORTA обнулён, прежде чем использовать.
        movwf   PORTB           ; Убедимся, что PORTB обнулён, прежде чем использовать.
        bsf     STATUS,RP0      ; Выбираем Bank 1
        clrw                    ; Маскируем все биты для выхода.
        movwf   TRISB           ; Устанавливаем регистр TRISB.
```

```
        bcf     STATUS,RP0      ; Переключаемся к Bank 0.

initialise clrw                 ; Начальное состояние.
        movwf   state           ; Устанавливаем его.

loop      call    getmask       ; Конвертируем состояние в bitmask.
        movwf   PORTB           ; Записываем его в порт.
        incf    state,W         ; Увеличиваем на единицу состояние в W.
        andlw   0x04            ; Оборачиваем его.
        movwf   state           ; Возвращаем в память.
        call    wait            ; Ждем :-)
        goto    loop            ; И цикл :-)

        ; Функция возвращения bitmask для выхода порта для текущего состояния.
        ; Верхний кусок содержит биты для одной установки света и
        ; нижний кусок биты для другой установки. Бит 1 красный, 2 жёлтый
        ; а бит три зелёный. Бит четыре не используется.

getmask    movf    state,W      ; Берём статус в W.
        addwf   PCL,F           ; Добавляем компенсацию в W для PCL для расчета. goto.
        retlw   0x41            ; state==0 это зелёный и красный.
        retlw   0x23            ; state==1 это жёлтый и красный/жёлтый.
        retlw   0x14            ; state==3 это красный и зелёный.
        retlw   0x32            ; state==4 это красный/жёлтый и жёлтый.

        ; Функция, использующая два цикла для реализации паузы.

wait      movlw   5
        movwf   l1

w1        call    wait2
        decfsz  l1
        goto    w1
        return

wait2     clrf    l2
w2        decfsz  l2
        goto    w2
        return
        END
```

Здесь есть, фактически, предумышленная ошибка в коде выше, но об этом чуть позже...

Добавление исходного файла

Следующий этап — добавление программы в проект, чтобы мы могли успешно симулировать её поведение. Мы сделаем это, используя команды *Source* основного меню. Перейдите к разделу *Source* и выберите команду *Add/Remove Source Files*. Щёлкните по кнопке **New**, переместитесь к директории «Samples\Tutorials» для выбора TL.ASM файла. Щёлкните по кнопке **Открыть** и файл появится в выпадающем списке *Source Code Filename*.

Теперь нам нужно выбрать транслятор для файла. Для нашей цели подходит MPASM. Эта опция доступна из выпадающего списка, а щелчок левой клавишей мышки выберет её обычным образом. (Заметьте, что если вы планируете использовать новый ассемблер или компилятор в этот момент, вы должны зарегистрировать его, используя команду *Define Code Generation* в разделе *Tools*).

И, наконец, необходимо указать, какой файл процессор должен запускать. В нашем примере это будет tl.hex (hex-файл производит MPASM при ассемблировании tl.asm). Чтобы добавить этот файл в процессор, откройте двойным щелчком по компоненту PIC диалоговое окно *Edit Component*, в котором есть поле для *Program File*. Если это уже не задано, как tl.hex, укажите путь к файлу вручную или перейдите к месту хранения файла с помощью кнопки «**?**» справа от поля. Когда вы задали hex-файл, нажмите **ОК** для выхода из диалога.

Теперь мы прикрепили исходный файл к проекту и указали, какой транслятор использовать. Больше информации о системе управления исходным кодом вы найдёте далее.

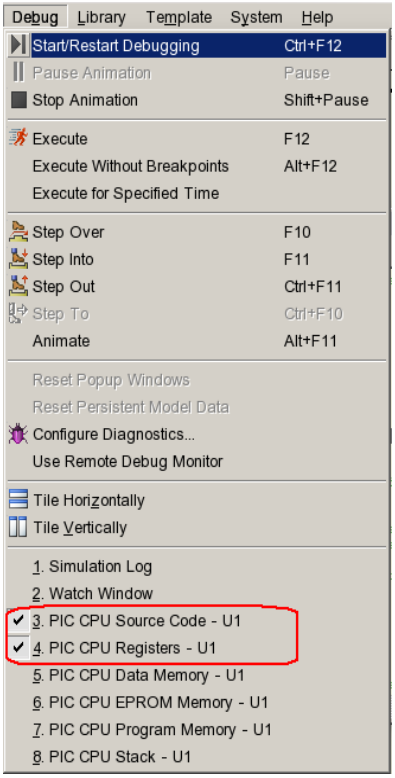
Отладка программы

Симуляция схемы

Чтобы симулировать схему, поместите указатель мышки на кнопку **Play** (►) панели анимации в левом нижнем углу экрана и нажмите её. Панель состояния покажет время активности анимации. Вы должны также отметить, что один из светофоров зелёный, тогда как другой красный, и логическое состояние выводов должно быть видно на схеме. Вместе с тем, вы заметили, что состояние светофоров не меняется. Это произошло из-за преднамеренной ошибки, которую мы сделали в коде. На этом этапе самое время отладить программу, чтобы избавиться от проблемы.

Режим отладки

Чтобы убедиться, что мы находимся в режиме отладки схемы, мы остановим текущую симуляцию. Сделав это (кнопка Stop — ■ на панели анимации) вы можете начать отладку, нажав клавиши **CTRL+F12**. Появятся два всплывающих окна — одно показывает текущие значения регистров, а другое исходный код программы. Если нет, их можно активировать из меню *Debug*, как и остальные информационные окна.



Нам также хотелось бы активировать Watch Window, в котором можно увидеть изменение в состоянии переменных.

Полное описание этих возможностей вы найдёте в разделе, озаглавленном «Watch Window — окно наблюдения».

Сконцентрируемся сейчас на окне Source, обратив внимание на красную стрелку слева. Это, вместе с подсвеченной строкой, показывает текущую позицию счётчика программы. Чтобы задать здесь точку останова (breakpoint), нажмите правую клавишу мышки (точка останова всегда устанавливается на подсвеченной строке) и из выпадающего меню выберите *Toggle (Claear/Set) Breackpoint*. Если вы хотите очистить точку останова, вы можете сделать это повторив операцию, но мы пока её оставим.

Задание точек останова

Взглянув на программу, можно заметить, что цикл возвращается к себе. Неплохо бы было установить точку останова в начале цикла, до того как мы стартуем. Вы можете сделать это подсветив линию (по адресу 000E) мышкой и затем нажав **F9**. Затем нажмите **F12**, чтобы запустить программу. Вы должны увидеть сообщение на панели состояния (Status Bar), показывающее, что цифровая точка останова достигнута и адрес программного счётчика (Program Counter, PC). Это относится к адресу первой точки останова, которую мы задали.

Список ключей отладки есть в разделе *Debug* основного меню, но для большей части работы мы используем клавишу **F11** для пошагового прохождения программы. Нажмите клавишу **F11** и заметьте, что красная стрелка слева перемещается к следующей инструкции. Все, что мы реально сделали — это выполнили команду «clrw», а затем остановились. Вы можете проверить это, взглянув на регистр W в *окне регистров* (PIC CPU Registers) и увидев, что регистр обнулился.

Все, что нам сейчас нужно сделать, это определить, что мы предполагаем должно произойти при выполнении следующей инструкции, а затем проверить, так ли это? Например, следующая инструкция должна переместить содержимое регистра «W» в PORTA. То есть, порт A должен очиститься. Выполним эту инструкцию и проверим окно регистров, убедимся, что это так. Продолжая выполнение, пока не будет достигнута второй точки останова, можно отметить, что оба порта были очищены и готовы для вывода (как это диктуется регистром TRISB), и что переменная state корректно установлена в 0.

Поскольку далее следует функция вызова (call), мы должны выбрать шаг через функцию (Stepping Over, нажатием клавиши **F10**), но для полноты мы пройдём через все инструкции. Нажатие клавиши F11 в этом месте заставляет «нас» перепрыгнуть на первую выполняемую

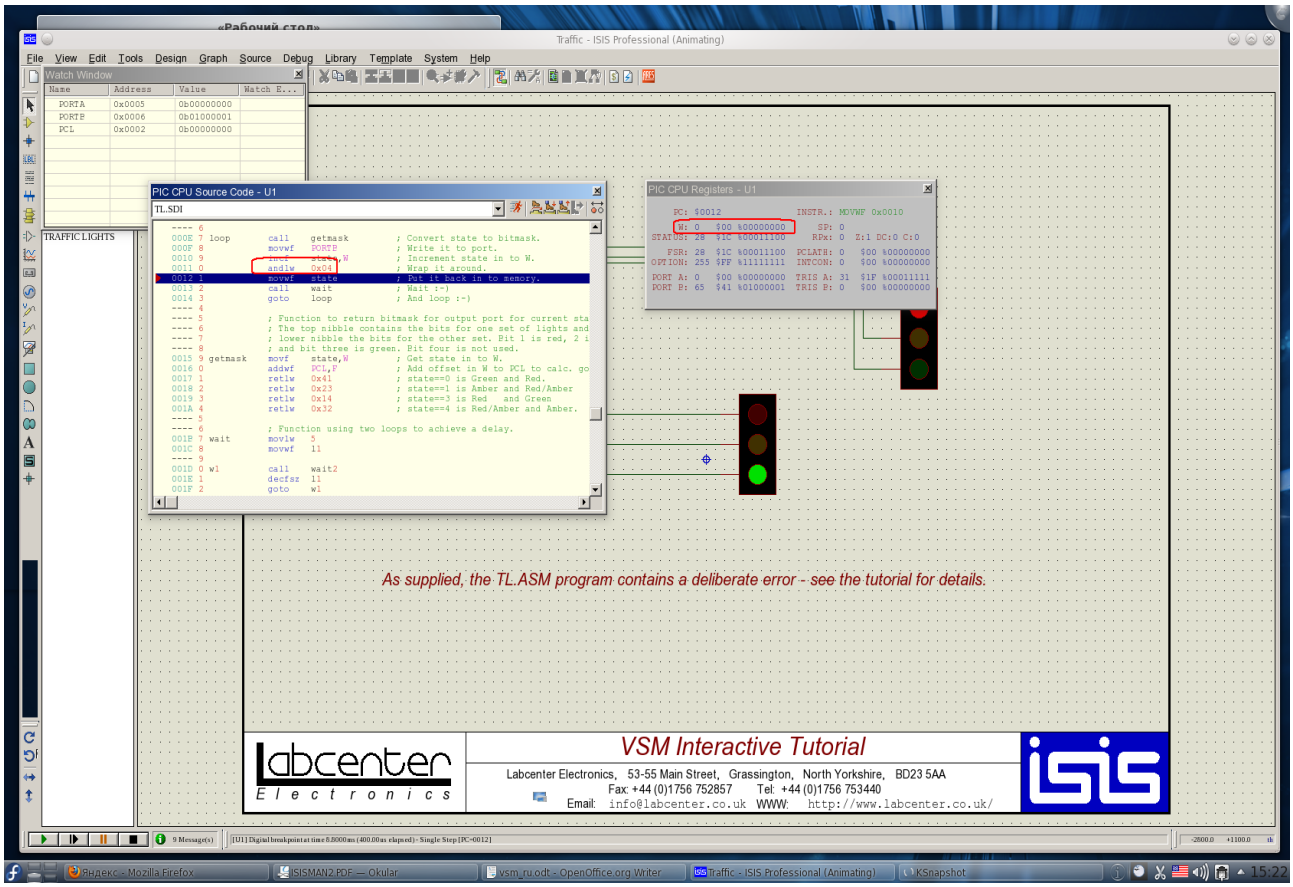
строку функции getmask. Передвигаясь вперёд, мы увидим, что операция move успешна, и что мы «приземлились» в нужном месте для добавления маски в нашу таблицу обозрения. Когда мы вернёмся в основную программу, мы получим маску, которую и предполагали. Шагая дальше и записывая маску в порт, мы увидим правильный результат на схеме. Следующий шаг увеличит на единицу state, это тоже успешно, о чем свидетельствует *окно наблюдения*, где значение регистра «W» увеличивается на 1.

Следующий шаг приводит нас к инструкции, выполняющей возврат state в ноль, когда она увеличивается больше 3. Это, как можно видеть в *окне наблюдения*, не выполняется, хотя должно бы. Переменная state остаётся со значением 1 для маски, которая должна быть правильно установлена при следующем выполнении цикла.

Поиск ошибки

Завершение расследования показывает, что проблема обнаруживается при операции AND с 4 вместо 3. Мы хотели бы, чтобы состояния были 0, 1, 2, 3 и любое из них, когда AND 4, даёт ноль. Вот почему при запуске симуляции состояние светофоров не меняется. Решение простое — заменить проблемную инструкцию AND со state с 4 на 3. Это будет означать, что когда state будет увеличиваться до 3, и когда регистр «W» увеличится до 4, переменная state будет обнуляться. Альтернативное решение — просто проверить в этом случае регистр «W», и когда он станет равен 4, обнулить его.

Этот короткий пример отладки в Proteus VSM иллюстрирует, в основном, что есть множество дополнительных функций. Очень рекомендуем, чтобы вы заглянули в раздел «Отладка на уровне исходного кода» позже в этом документе, чтобы ознакомиться со все этим детальнее.



РУКОВОДСТВО ПО СИМУЛЯЦИИ, ОСНОВАННОЙ НА ГРАФИКАХ

Введение

Цель этого руководства показать вам, используя схему простого усилителя, как выполнять симуляцию, основанную на графиках с PROTEUS VSM. Шаг-за-шагом мы пройдём через:

- Размещение графиков, пробников и генераторов.
- Выполнение актуальной симуляции.
- Использование графиков для отображения результатов и получения замеров.
- Обзор некоторых типов анализа, которые доступны.

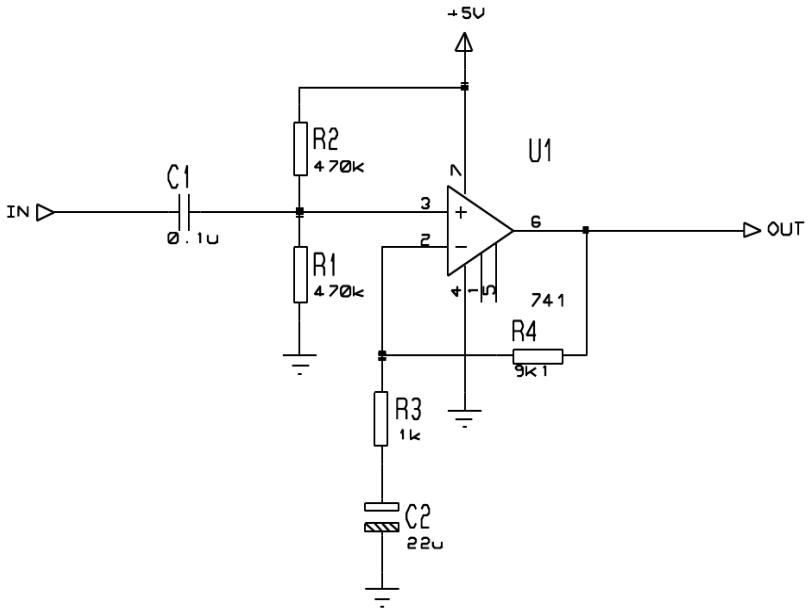
Руководство не отражает полного использование ISIS, таких процедур, как размещение, соединение компонентов, выделение объектов и т.д. Полнее это описано в руководстве «По интерактивной симуляции» и ещё детальнее в руководстве к ISIS. Если вы ещё не освоились с использованием ISIS, сделайте это до того, как обратитесь к этому руководству.

Мы очень советуем вам проработать это руководство до попыток самостоятельно использовать симуляцию, основанную на графиках: освоив концепции, много легче понять и весь материал в соответствующих разделах, что избавит вас от непроизводительных затрат времени и разочарования в дальнейшем.

Начнём

Схема, которую мы собираемся симулировать — это аудио усилитель на основе ОУ 741, как показано ниже. На ней ОУ 741 в обычной конфигурации с питанием от единственного источника 5 В. Резисторы обратной связи R3 и R4 задают усиление порядка 10. Входные компоненты R1, R2 и C1 создают «фальшивое» заземление на не инвертирующем входе, который «развязан» от сигнала.

Как и принято в подобных случаях, мы используем анализ переходных процессов (transient analysis) электрической цепи. Этот вид анализа очень полезен, даёт большой объем информации о схеме. После завершения описания симуляции с анализом переходных процессов, будет сопоставление с другими видами анализа.



Если хотите, вы можете нарисовать схему самостоятельно, но можете загрузить готовый проект: «Samples\Tutorials\ASIMTUT1.DSN». Каков бы ни был ваш выбор, убедитесь сейчас, что ISIS запущен и схема нарисована.

Генераторы

Чтобы проверить схему, нам нужно снабдить её подходящим входным сигналом. Мы будем использовать источник напряжения прямоугольной формы в качестве тестового сигнала. Для генерации требуемого напряжения будет использован объект «генератор».

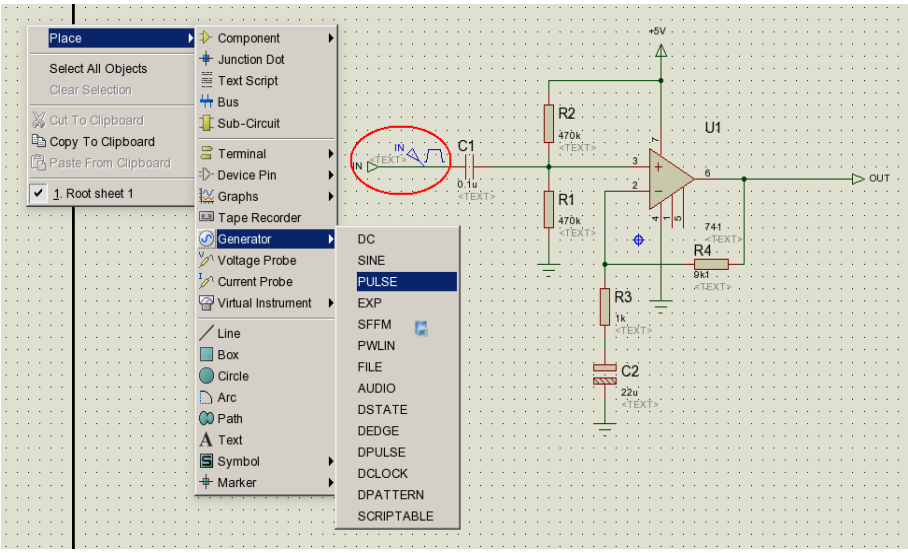
Чтобы поместить на схему генератор, щёлкните правой клавишей мышки на свободном месте рабочего поля, выберите *Place-Generator-PULSE*: для нашей симуляции нам понадобится Pulse-генератор. Выберите генератор, переместите курсор мышки в окне редактора правее контакта IN, и щёлкните левой клавишей мышки по проводу, чтобы поместить генератор.

Объект генератор похож на другие объекты в ISIS — такие же процедуры для предварительного просмотра и ориентации, редактирования генератора, перемещения или удаления (см. «Основные приёмы редактирования» в руководстве к ISIS или «Генераторы и пробники» в этом руководстве).

Также, как мы «прицепили» генератор к существующему проводу, как мы это делаем, мы можем размещать генераторы на листе и соединять со схемой обычным образом. Если оттащить генератор от соединения, тогда ISIS «решит», что вы хотите его отсоединить, и не будет «тащить» провод за ним, как это делает для обычных компонентов.

И, заметьте, как генератор автоматически присваивает ссылку — имя контакта IN.

Если генератор соединяется с объектом (или помещается непосредственно на существующий провод) он присваивает имя цепи, к которой подключён. Если цепь не имеет имени, тогда имя ближайшего вывода компонента будет использоваться по умолчанию.



И, наконец, мы должны отредактировать генератор, чтобы определить параметры импульсов, которые нам нужны. Чтобы отредактировать генератор, щёлкните по нему правой клавишей мышки, выделяя, и выберите из выпадающего меню *Edit Properties*, открывая диалог свойств. Выберите поле Pulsed (High) Voltage и задайте значение 10mV. Также установите ширину импульса 0.5s.

Нажмите клавишу ОК, чтобы изменения вступили в силу. Раздел «Генераторы и пробники» даёт исчерпывающую информацию о свойствах, распознаваемых всеми типами генераторов. Для этой схемы нужен только один генератор, но их количество для размещения не ограничено.

Пробники

После того, как мы определили входной сигнал для нашей схемы, используя генератор, мы должны теперь поместить пробники в точки, за которыми хотим наблюдать. Более всего нас интересует выход, и вход, после того, как он был настроен, тоже полезная точка для пробника. Если нужно, можно добавить ещё пробники в ключевые точки и повторить симуляцию.

Для размещения пробников щёлкните правой клавишей мышки на свободном месте чертежа и выберите из выпадающего меню *Place-Voltage Probe* (убедитесь, что выбрали ненароком не *Current Probe* — мы вернёмся к этому позже). Пробники должны помещаться поверх проводов или помещаться, а затем соединяться с проводами, так же, как и генераторы. Переместите курсор мышки в *окне редактирования* левее вывода 3 U1 и щёлкните левой клавишей мышки, чтобы добавить пробник к проводу, соединённому с выводом 3 и резисторами R1 и R2. Убедитесь, что пробник помещён на провод, поскольку он не может быть помещён непосредственно на вывод. Заметьте, что пробник приобрёл имя ближайшего компонента, к которому он присоединён, с номером вывода в скобках. Теперь поместите второй пробник левее контакта OUT на провод между точкой соединения и контактом.

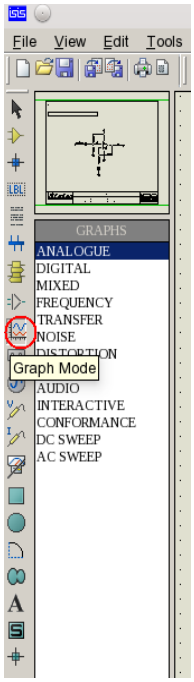
Пробник, как объект, подобен генераторам и большинству других объектов в ISIS — те же процедуры для просмотра, ориентации пробника перед размещением, редактирования, перемещения, переориентации или удаления после размещения (см. раздел « см. «Основные приёмы редактирования» в руководстве к ISIS или «Пробники» в этом руководстве).

Пробники можно редактировать, изменяя их этикетки. В нашем случае прекрасно подходят названия по умолчанию, но при необходимости используйте кончик пробника, а не тело.

Теперь, когда схема готова к симуляции, нам нужно разместить график для отображения результатов.

Графики

Графики играют важную роль в симуляции: они не только служат средой для отображения результатов, но действительно отражают все, что выполняет симуляция. Размещая один (или более) график и показывая какой из типов данных вы ожидаете увидеть на графике (цифровой, напряжение, импеданс и т.п.), вы даёте знать ISIS, какой тип симуляции следует применить и какую часть схемы следует включить в симуляцию. Для анализа переходных процессов нам нужен аналоговый (Analogue) тип графика. Он назван аналоговым, а не **transient**, чтобы отличать его от цифрового (Digital) типа, который используется для отображения результатов при цифровом анализе, действительно специализированной форме **transient** (анализ переходного процесса) анализа. Оба могут изображаться на той же оси времени, если использовать Mixed (смешанный) график.



Чтобы поместить график, вначале выберите иконку *Graph Mode*: в окне выбора компонентов отобразится список всех доступных типов графиков. Затем выберите *Analogue* тип, поместите курсор мышки на свободном месте окна редактирования, щёлкните левой клавишей мышки и «растащите» прямоугольник, чтобы он стал нужного размера, и щелкните ещё раз, размещая график.

График ведёт себя подобно другим объектам в ISIS, хотя есть и некоторые тонкости. Мы снабдили необходимым пример для руководства, но соответствующий раздел будет лучше почитать. Вы можете выделять график обычным образом, затем, используя «ручки» и левую клавишу мышки растягивать график, или поместив курсор на график послед выделения, когда курсор приобретает вид руки, перемещать сам график

Нам сейчас нужно добавить наш генератор и пробники на графике. Каждый генератор имеет пробник, связанный с ним, так что нет необходимости помещать пробник непосредственно на генераторе, чтобы увидеть входной сигнал. Есть три способа добавить генератор и пробники на график:

- Первый — выделить каждый пробник/генератор по очереди и перетащить их на график — точно так, как вы их перемещаете на новое место. ISIS обнаружит, что вы пытаетесь поместить пробник/генератор на графике, возвращает их на место и добавляет кривые на графике с теми же ссылками, что и на пробнике/генераторе. Кривые могут быть связаны с левой или правой осями в аналоговом графике, а пробник/генератор будут добавлены к ближайшей оси со стороны размещения.

Независимо от того, куда вы поместили пробник/генератор, новая кривая всегда добавляется ниже уже существующих графиков.

Второй и третий методы добавления пробников/генераторов на график оба используют команду *Add Trace* из меню графика; эта команда всегда добавляет пробники к текущему графику (если их более одного, текущий график, он один, выбирается через раздел основного меню *Graph*).

- Если команда *Add Trace* выполняется без выделенного пробника или генератора, тогда появляется диалоговая форма *Add Transient Trace*, и пробник может быть выбран из списка всех пробников проекта (включая пробники на других листах).
- Если есть выделенные пробники/генераторы, выполнение команды *Add Trace* (из основного меню) приведёт к тому, что появится приглашение *Quick Add* (быстро добавить) выделенные пробники к текущему графику; при выборе опции **Cancel**, появляется диалоговая форма *Add Transient Trace*, как описано выше. Выбор OK добавляет все выделенные пробники/генераторы к текущему графику в алфавитном порядке.

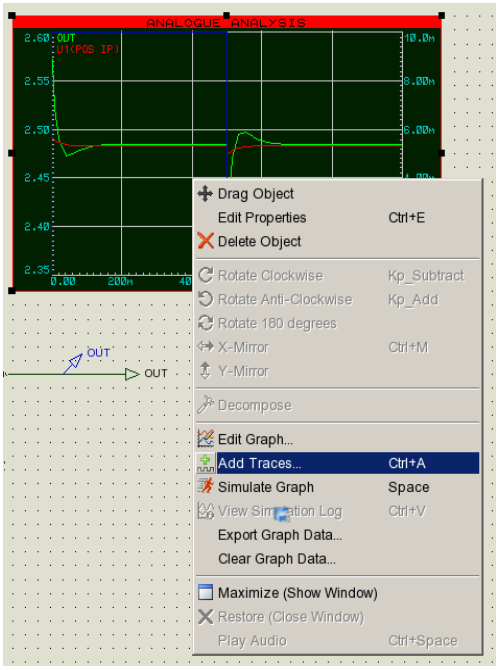
Мы используем *Quick Add* для добавления наших пробников и генератора к графику. Либо выделите пробники и генераторы индивидуально, или, ещё быстрее, сделайте прямоугольное выделение всей схемы — механизм *Quick Add* проигнорирует все выделенные объекты, кроме пробников и генераторов. Выберите команду *Add Trace* из раздела *Graph* основного меню и нажмите кнопку **ОК** в приглашении. Кривые (их заголовки) появятся на графике (поскольку график один, и он последним использовался, понятно, что он — текущий график). В этот момент кривые состоят из имени (на левой оси) и из области пустых данных (основная область графика). Если кривые не появились, возможно, они слишком малы, чтобы программа ISIS смогла нарисовать их. Увеличьте график, выделив его и растаскивая углы, чтобы получить нужный размер.

Как только это произойдёт, наши кривые (размещённые в алфавитном порядке) появятся. Однако мы можем их перемешать. Чтобы это сделать, убедитесь, что график не выделен, щёлкните левой клавишей мышки по имени кривой и перетащите, удерживая левую клавишу, её в нужное место. Кривая подсвечивается, чтобы показать, что она выделена. Кривую можно удалить, выбрав команду *Delete Trace* из выпадающего меню после щелчка правой клавишей мышки по имени кривой. Чтобы снять выделение со всех кривых, щёлкните мышкой в свободном месте чертежа.

Остался последний штрих установок перед тем, как выполнить симуляцию — это задать время симуляции. ISIS симулирует схему согласно со временем окончания на шкале x графика, а для нового графика это задано в 1 секунду. Для нашей цели мы хотели бы, чтобы входной сигнал был близко к высшей рабочей частоте, скажем, 10 кГц. Это потребует общего периода в100µs. Выделите график и щёлкните по нему левой клавишей мышки, чтобы появился диалог *Edit Transient Graph*. Форма эта имеет поля, которые позволяют вам озаглавить график, задать время начала и окончания симуляции (это относится к левому и правому наибольшим значениям оси x), озаглавить левую и правую оси (это не отображается на цифровых графиках), а также задать основные свойства для запуска симуляции. Всё, что нужно сделать нам, это изменить время окончания симуляции с 1.0 на 100u (вы можете ввести символы 100u — ISIS конвертирует это в 100E-6) и нажать **ОК**.

Теперь проект готов к симуляции. В этом месте, пожалуй, лучше загрузить нашу версию

проекта (Samples\Tutorials\ASIMTUT2.DSN), чтобы избежать появления каких-либо проблем при симуляции и для следующих разделов. Но вы можете и продолжить с проектом, который правили сами, а загрузить файл ASIMTUT2.DSN только в том случае, если проблемы появятся.



Симуляция

Чтобы симулировать схему, всё что вам нужно сделать, это дать команду *Simulate Graph* в разделе *Graph* основного меню (или использовать клавиатуру — пробел). Команда выполнит симуляцию и текущий график (тот, что отмечен в меню *Graph*) будет обновлён результатами симуляции.

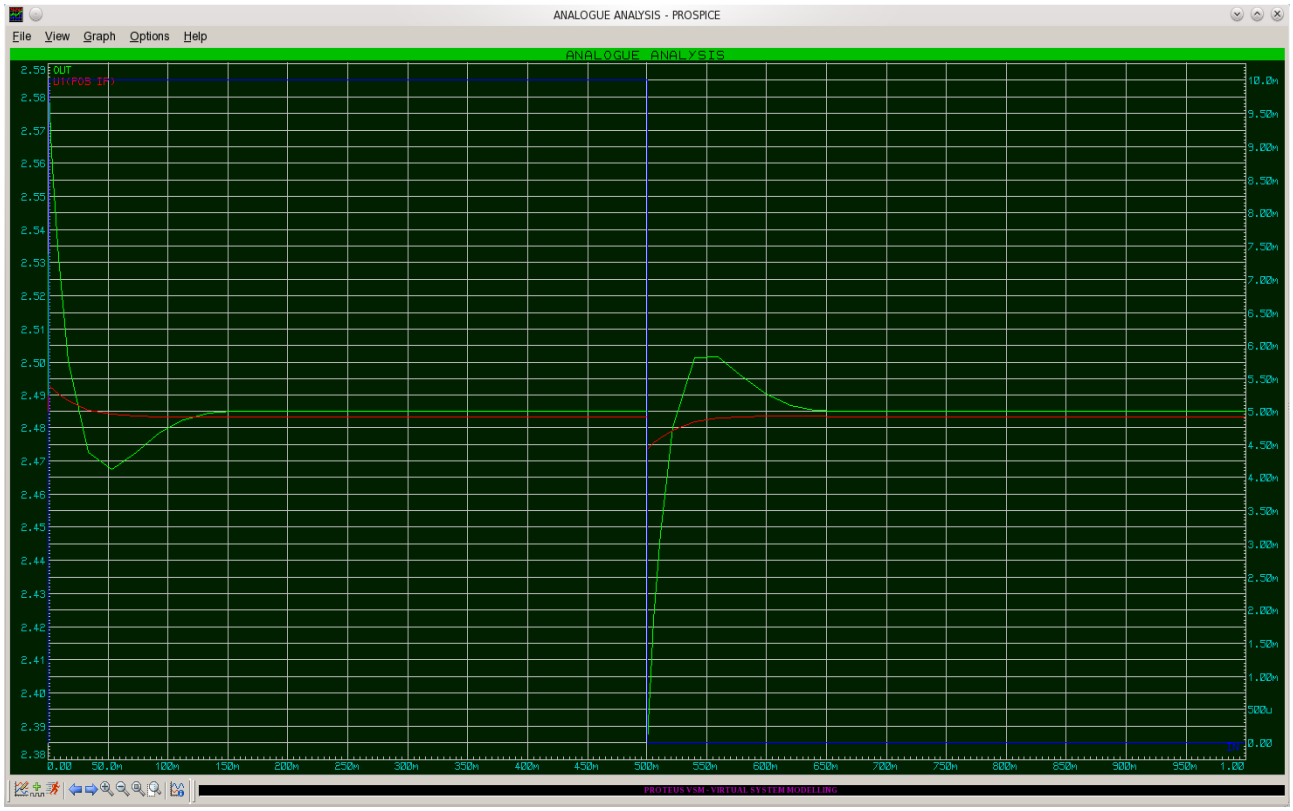
Выполните это сейчас. Панель состояния показывает, до какого места дошёл процесс симуляции. Когда симуляция завершена, график перерисовывается, используя новые данные. Для текущей версии ISIS (версии 6.0) и ядра симулятора начало графика игнорируется — симуляция всегда начинается с нулевого времени и продолжается до достижения времени остановки или до момента, когда симуляция доходит до неизменного состояния. Вы можете прервать симуляцию на середине, нажав клавишу **ESC**.

Журналирование симуляции ведётся для последней выполненной симуляции. Вы можете увидеть лог (запись журнала), используя команду *View Log* из раздела *Graph* основного меню (или с помощью горячих клавиш **CTRL+V** клавиатуры). Лог при аналоговой симуляции редко появляется для немедленного чтения, только тогда, когда есть предупреждения или ошибки, о которых нужно знать, и в этом случае вы можете просмотреть детали — что пошло не так. В некоторых случаях, однако, запись симуляции может предоставить полезную информацию, которую нелегко найти на кривых графика.

Итак, первая симуляция завершена. Бросив взгляд на график, трудно разглядеть какие-то детали. Чтобы проверить, работает ли схема должным образом, нам нужно выполнить некоторые замеры...

Выполнение измерений

График, созданный на схеме, минимизирован. Чтобы провести измерения, мы вначале должны максимизировать его. Для этого вначале убедитесь, что график не выделен, а затем щёлкните левой клавишей мышки по его заголовку — график перерисуется в своём собственном окне. В верхней части окна появится меню. Под ним в левой части экрана есть область, в которой отображаются заголовки кривых, а справа от этого сами кривые. В нижней части окна, слева, есть инструментальная панель, правее которой панель состояния, отображающая информацию о курсоре (времени/состоянии). Поскольку это новый график, и мы не проводили никаких измерений, на графике не видно курсоров, а панель состояния просто отображает заглавное сообщение.



Кривые повторяют цвета своих заголовков. Кривые OUT и U1(POS IP) собраны в верху, тогда как IN внизу. Чтобы разглядеть детали кривых, нам нужно отделить кривую IN от остальных двух. Это можно сделать, перетащив левой клавишей мышки заголовок кривой на правую часть экрана (как на рисунке выше). В этом случае появляется правая ось y, которая масштабирована независимо от левой. Кривая IN теперь выглядит гораздо больше, но это потому, что ISIS выбирает наилучший масштаб для обзора на правой оси. Для того, чтобы сделать график яснее, возможно, лучше удалить кривую IN, а U1(POS IP) пока ещё полезна. Щёлкните правой клавишей мышки по заголовку IN, и выберите *Delete Trace* из выпадающего меню. График вернулся к единственной, расположенной слева оси y.

Мы измеряем два значения:

- Усиление по напряжению схемы.
- Подходящее время спада на выходе.

Эти измерения выполняются с помощью *Cursors*.

Каждый график имеет два курсора, названные как *Reference* и *Primary* курсоры. Ссылочный (reference) отображается красным, а первичный (primary) зелёным. Курсор всегда «привязан» к кривой; кривая, к которой привязан курсор, обозначается маленьким крестиком «x», который перемещается по кривой. Маленькие маркеры на обоих осях x и y следуют за этим перекрестием «x», чтобы дать возможность правильно прочитать значение на осях. Если при перемещении использовать клавиатуру, то курсор будет перемещаться по оси x маленькими шагами.

Давайте начнём с размещения *Reference* курсора. Для доступа к обоим, *Reference* и *Primary*, курсорам используются одинаковые клавиши. Какой из них, определяется клавишей **CTRL** на клавиатуре; *Reference* курсор, используемый меньше, всегда требует нажатия клавиши **CTRL**. Чтобы поместить курсор, всё что от вас требуется, это указать точку на кривой (не этикетку кривой — она используется для другой цели), где вы хотите «прицепить» курсор, и щёлкнуть левой клавишей мышки. Если нажата клавиша **CTRL**, вы поместите (или будете перемещать) *Reference* курсор; если же клавиша **CTRL** не нажата, тогда вы поместите (или переместите) *Primary* курсор. Пока нажата клавиша мышки (и клавиша **CTRL** для *Reference* курсора), вы можете перетаскивать курсор. Итак, нажмите (и удержите) клавишу CTRL, поместите указатель мышки справа на графике над двумя кривыми и нажмите левую клавишу мышки. Появится красный *Reference* курсор. Протащите курсор (всё ещё с нажатой клавишей **CTRL**) между 70u и 80u по оси x. Заголовок в строке состояния пропадает, и теперь строка состояния отображает время (под курсором, красным, слева) и напряжение (под курсором) с именем кривой (справа). Это кривая OUT, которую мы хотели видеть.

Вы можете перемещать курсор по оси X, используя курсорные клавиши клавиатуры (влево и вправо), и вы можете «прицепить» курсор к предыдущей или следующей кривой, используя курсорные клавиши вверх и вниз. Клавиши **ВЛЕВО** и **ВПРАВО** перемещают курсор к левому или к правому ограничителю оси X, соответственно. С нажатой клавишей **CTRL** попробуйте перемещать клавишами влево-вправо *Reference* курсор маленькими шагами по оси времени.

Теперь поместите *Primary* курсор по кривой OUT между 20u и 30u. Процедура такая же, что и для Reference курсора выше, исключая то, что вам не нужно удерживать клавишу **CTRL**. Время и напряжение (зелёным) для первичного курсора добавляются теперь на панель состояния.

Также отображается разность и по времени (DX), и по напряжению (DY) между положениями обоих курсоров. Разность напряжения составляет около 100mV. Входной импульс был 10mV, так что усиление по напряжению составляет около 10. Заметьте, что значение положительное, поскольку *Primary* курсор над *Reference* курсором — дельта выхода это значение Primary минус Reference.

Мы можем также измерить время спада, используя значение разности во времени между позициями курсоров на падающем участке выходного импульса. Это можно сделать либо перетаскивая курсоры мышкой, либо курсорными клавишами (не забудьте про клавишу **CTRL** для *Reference* курсора). *Primary* курсор должен быть правее по кривой, на её спрямлении, а *Reference* курсор на сгибе начального участка спада импульса. Вы можете определить, что время спада чуть меньше10µs.

Использование пробников тока

Теперь мы должны закончить с измерениями, мы можем вернуться к схеме — достаточно закрыть окно графика обычным образом или для ускорения можно нажать клавишу **ESC** на клавиатуре. Мы используем теперь токовый пробник для проверки тока в петле обратной связи, измеряя ток через R4.

Пробники тока используются аналогично пробникам напряжения, но с одним важным отличием. Токовый пробник нуждается в направлении, связанном с ним. Пробники тока работают фактически в разрыве провода, куда они вставляются, так что они должны «знать» пути, их окружающие. Это выполняется просто способом размещения. В ориентации по умолчанию (в направлении направо) пробник тока измеряет ток в горизонтальном проводе слева направо. Чтобы измерить ток в вертикальном проводе, пробник нужно повернуть на 90° или 270°. Размещать пробник под неверным углом — это ошибка, о чем будет сообщено при выполнении симуляции. Если есть сомнения, посмотрите на стрелку на символе. Она указывает на направление тока.

Выберите пробник тока, щёлкнув по иконке *Current Probe Mode*. Щёлкните по иконке поворота по часовой стрелке, так чтобы стрелка показывала вниз. Затем поместите пробник на вертикальный провод между правым выводом R4 и выводом 6 U1. Добавьте пробник к правому краю графика, выделив и перетащив его на правую сторону минимизированного графика. Правая сторона — хороший выбор для пробника тока, поскольку обычно масштаб его амплитуды иной, чем у пробника напряжения, так что отдельная ось понадобится для детального отображения. В этот момент нет кривой, отображаемой пробником тока. Нажмите пробел для новой симуляции графика, и кривая появится.

Даже из минимизированного графика мы можем видеть, что ток в петле обратной связи следует форме выхода, как вы могли бы ожидать от операционного усилителя. Ток изменяется между 10μA и 0 в верхней и нижней части кривой соответственно. Если вы хотите, график можно максимизировать для детальной проверки кривой.

Частотный анализ

Как и анализ переходных процессов, при симуляции аналоговых цепей возможны несколько других типов анализа. Во многом они используются таким же образом: для графиков, пробников и генераторов, — но все они по разному варьируют эту тему. Следующий тип анализа, который мы проведём, это частотный анализ. При частотном анализе по оси x откладывается частота (в логарифмическом масштабе), а амплитуда и фаза могут отображаться по оси y.

Для выполнения частотного анализа требуется график FREQUENCY. Щёлкните по иконке Graph Mode, чтобы вновь отобразить список типов графиков в окне выбора, и щёлкните по типу FREQUENCY. Затем поместите график на схему, как и раньше, растащите окно графика левой клавишей мышки. Нет необходимости удалять уже существующий график переходного процесса, но вы может это сделать, в плане освободить как можно больше места (щёлкните по графику правой клавишей мышки и выберите из выпадающего меню *Delete Object*).

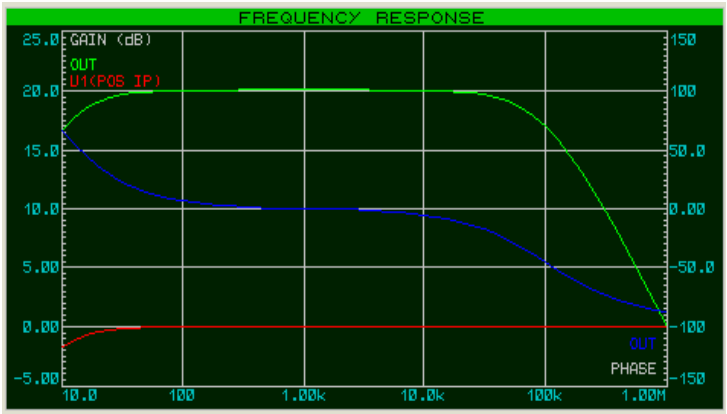
Теперь добавим пробники. Мы добавим оба пробника напряжения OUT и U1(POS IP). На частотном графике две y-оси (левая и правая) имеют специальное назначение. Левая ось y используется для отображения амплитуды сигнала, а правая ось для отображения фазы. Чтобы видеть обе, следует добавить пробники на обе стороны графика. Выделите и перетащите пробник OUT на левую сторону графика, затем перетащите его на правую

сторону. Обычно каждый график имеет свой цвет, но обе кривые имеют одинаковое имя. А теперь выделите и перетащите U1(POS IP) пробник, но только на левую сторону графика.

Значения амплитуды и фазы должны быть заданы по отношению к некоторому значению. В ISIS это выполняется заданием *Reference Generator*. Он всегда имеет выход в 0dB (1 вольт) при 0°. Любой существующий генератор может быть задан, как *reference generator*. Все другие генераторы в схеме игнорируются при частотном анализе. Чтобы задать IN генератор как «относительный» в нашей схеме, просто, выделите и перетащите его на график, как вы это делали с пробниками. ISIS «поймёт», поскольку это генератор, что он должен быть относительным генератором, о чём и сообщит вам. Удостоверьтесь, что вы это сделали, или симулятор будет работать некорректно.

Нет необходимости править свойства графика, поскольку частотный диапазон, выбираемый по умолчанию, вполне подходит для наших целей. Однако, если вы сделаете это (указывая график и нажимая **CTRL-E**), вы увидите диалог Edit Frequency Graph, который слегка отличается от аналогичного для анализа переходного процесса. Нет нужды маркировать оси, и их свойства фиксированы, но есть флажок, который позволяет отображать амплитуду в децибелах или в обычных единицах. Эту опцию лучше оставить в dB, поскольку отображаемые абсолютные значения не будут реальными значениями, присутствующими в схеме.

Теперь нажмите пробел (когда курсор помещён поверх частотного графика), чтобы начать симуляцию. Когда она закончится, щёлкните левой клавишей мышки по заголовку графика, чтобы максимизировать график. Просматривая вначале кривую амплитуды OUT, мы можем заметить, что в полосе пропускания усиления составляет 20dB (как и ожидалось), а диапазон рабочих частот лежит в полосе от 50Hz до 20kHz. Курсоры в данном случае работают так же, как и в предыдущем случае — вы можете проверить предыдущие утверждения с помощью курсоров. Кривая фазы OUT показывает ожидаемые фазовые искажения на краях диапазона, правая ветвь подходит к -90° при единичном усилении. Эффект фильтра на высоких частотах обнаруживается, если сравнить кривую OUT с U1(POS IP). Заметьте, что ось x логарифмическая, и для чтения значений на оси лучше использовать курсоры.



Анализ развёртки переменной

В ISIS можно наблюдать, как сказывается на схеме изменение некоторых параметров цепи. Есть два типа анализа, позволяющие сделать это — *DC Sweep* и *AC Sweep* (развёртка на постоянном и переменном токе). График *DC Sweep* отображает ряд значений рабочих точек в

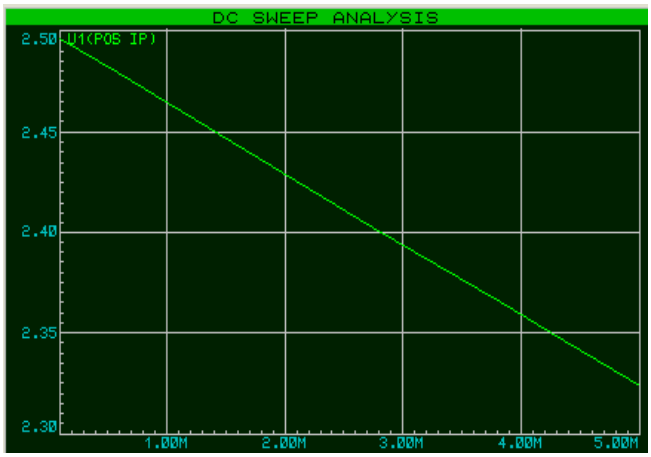
зависимости от переменной развёртки, а график *AC Sweep* отображает последовательность значений единственной точки частотного анализа амплитуды и фазы, подобно графику Frequency.

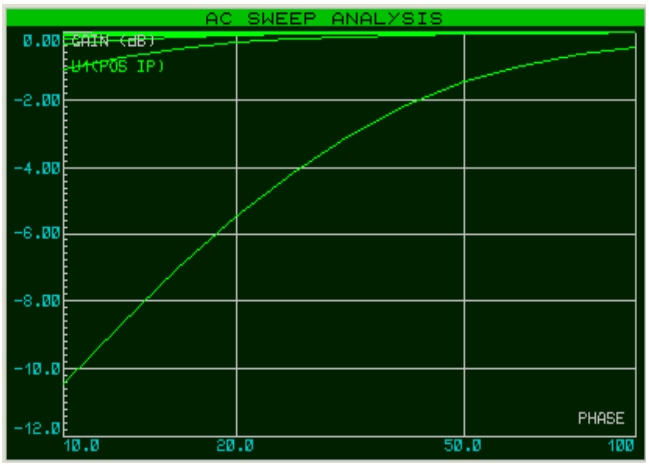
Поскольку обе формы схожи, мы обратимся к одной — *DC Sweep*. Резисторы смещения, R1 и R2, создают маленький ток для U1. Используем *DC Sweep*, чтобы увидеть, как сказывается значение этих резисторов на рабочей точке.

Для начала разместим график *DC Sweep* на свободном месте схемы. Затем выделим пробник U1(POS IP) и перетащим его на левую сторону графика. Нам нужно задать значение развёртки, а это делается с помощью редактирования графика (поместите курсор мышки на него и нажмите **CTRL-E**). Появится диалоговая форма *Edit DC Sweep Graph*, включающая поля для задания имени переменной качания, её начальное и конечное значения и количество шагов при развёртке. Мы хотели бы развернуть значения резисторов в диапазоне, скажем, от 100kΩ до 5MΩ, так что задаём для поля *Start* 100k, а для поля *Stop* 5M. Щёлкаем по **ОК**, чтобы изменения вступили в силу.

И, конечно, резисторы R1 и R2 следует подготовить к развёртке — изменить фиксированные значения, которые они имеют. Чтобы это сделать, редактируем R1 (щелчок правой клавишей мышки по резистору и выбор *Edit Properties* из выпадающего меню), меняя поле Value с 470k на X. Заметьте, что переменная развёртки на графике остаётся X. Щёлкаем по кнопке **ОК**, и повторяем все это для резистора R2.

Теперь можно симулировать схему: курсор мышки на график, нажать пробел. Затем, максимизировав график, вы можете отметить, что уровень смещения уменьшается, тогда как сопротивление рабочей цепи увеличивается. При 5MΩ все значительно меняется. Конечно, изменение этих резисторов также сказывается на частотном режиме. Мы должны провести AC Sweep анализ, чтобы посмотреть, что происходит в районе нижних частот.





Анализ шумов

Последняя форма анализа, о которой мы расскажем, это *Noise* анализ. В этом анализе симулятор показывает количество тепловых шумов, которые генерируются каждым элементом схемы. Все эти шумы затем суммируются (квадратично) в каждой испытуемой точке схемы. Результат выводится для полосы пропускания шумов.

Есть несколько важных особенностей анализа шумов:

- Время симуляции прямо пропорционально количеству пробников напряжения (и генераторов) в схеме, поскольку учитывается каждый из них.
- Пробники тока не учитываются при анализе шумов, следовательно, игнорируются.
- Значительная часть информации содержится в лог-файле симуляции.
- PROSPICE вычисляет и входной, и выходной шум. Чтобы сделать модель, вход должен быть определён как reference — это выполняется перетаскиванием генератора на график, как и при частотном анализе. Изображение входного шума при этом показывает эквивалентный шум на входе для каждого выходного пробника.

Чтобы выполнить анализ шумов нашей схемы, мы должны вначале вернуть резисторам R1 и R2 их прежнее значение 470kΩ. Сделайте это сейчас. Затем выберите тип графика Noise и поместите новый график на свободном месте чертежа. Нас реально интересует только выходной шум, так что перетащите OUT пробник напряжения на график. Как и раньше, предопределённые значения для симуляции нас вполне устраивают, но вам нужно задать reference для входного генератора IN. Диалоговая форма *Edit Noise Graph* имеет флажок для отображения результатов в dB. Если вы используете эту опцию, то убедитесь, что 0dB соответствует 1 вольту r.m.s. Щёлкните по **Cancel**, чтобы закрыть диалог.

Симулируйте график, как прежде. Когда график будет масимизирован, вы можете увидеть, что значения, производимые этим видом анализа, очень малы (nV в нашем случае), как вы можете убедиться из анализа этого типа. Но как вы можете просмотреть источники шума в вашей схеме? Ответ лежит в логе симуляции. Просмотрите запись журнала, нажав **CTRL+V**. Используйте иконку перемещения вниз (со стрелкой вниз), и вы должны увидеть линию, которая начинается словами:

Total Noise Contributions

Это список всех вкладов в шумы (во всем рабочем диапазоне частот) для каждого элемента схемы, производящего шум. Большинство элементов, фактически, внутри операционного усилителя, и имеют префикс U1_. Если вы установите флажок *Log Spectral Contributions* в диалоге *Edit Noise Graph*, то вы получите больше данных в журнале, показывающих вклад каждого компонента на каждой из частот.

ИНТЕРАКТИВНАЯ СИМУЛЯЦИЯ

БАЗОВЫЕ НАВЫКИ

Панель управления анимацией

Интерактивная симуляция управляется с панели, похожей на панель управления видеомагнитофоном, и работает как обычный пульт управления. Эта панель размещена в нижней левой части экрана. На панели четыре кнопки, которые вы используете для управления симуляцией схемы.

- Кнопка **PLAY** запускает симуляцию.
- Кнопка **STEP** позволяет вам выполнить пошаговую анимацию с заданным шагом. Если кнопка нажата и отпущена, тогда симуляция происходит на одном временном интервале; если кнопка нажата и удерживается, то анимация происходит до тех пор, пока вы не отпустите кнопку. Увеличение времени единичного шага может быть настроено в диалоге *Animated Circuit Configuration* (*System-Set Animation Options*). Возможность менять время шага позволяет более внимательно просматривать работу схемы и происходит медленнее, чем в действительности.
- Кнопка **PAUSE** останавливает анимацию, которая может возобновиться, если повторно нажать кнопку **PAUSE**, или может выполняться в пошаговом режиме при использовании кнопки **STEP**. Симуляция также приостанавливается, если обнаруживается точка останова.

  Симуляция также может быть приостановлена нажатием клавиши **PAUSE** на клавиатуре компьютера.
- Кнопка **STOP** говорит PROSPICEЮ, что нужно остановить симуляцию реального времени. Вся анимация останавливается и симулятор выгружается из памяти. Все индикаторы сбрасываются в их неактивное состояние, но приводы (выключатели и т.п.) остаются в их существующем положении.

  Симуляция может также останавливаться с клавиатуры клавишей **SHIFT-BREAK**.

Во время анимации текущее время симуляции и средняя загрузка CPU отображаются на панели состояния. Если не хватает мощности CPU для работы симулятора в реальном времени, время симуляции перестанет продвигаться в реальном времени. Отвлекаясь от этого, нет вредных последствий при симуляции очень быстрых схем, поскольку система автоматически регулирует количество симуляции, приходящейся на анимационный кадр.

Индикаторы и приводы

В отличие от обычных электронных компонентов, интерактивная симуляция обычно использует специальные Active Components (активные компоненты). Они имеют несколько графических состояний и проявляются в двух ипостасях: индикаторы и актураторы. Индикаторы отображают графическое состояние, которое меняется согласно измеряемому

параметру в схеме, тогда как Actuators (приводы) позволяют пользователю определять их состояние, что изменяет некоторые характеристики схемы.

Приводы сделаны с символами, небольшими красными маркерами, которые можно нажать мышкой для управления. Если у вас есть мышка с колёсиком, то вы можете также управлять актуратором, указав на него и прокручивая колесо в нужном направлении.

Установки интерактивной симуляции

Большей частью навык нужен в установках и запуске интерактивной симуляции, сконцентрированной на чертеже схемы в ISIS. Этот аспект наилучшим образом становится понятен, если работать с примерами в руководстве к ISIS. Однако процесс можно суммировать следующим образом:

- Укажите компонент, который вы хотите использовать, в библиотеке элементов, используя кнопку **P** в *окне выбора*.
- Разместите компоненты на схеме.
- Отредактируйте их — щелчок правой клавишей мышки и выпадающее меню или **CTRL-E** — чтобы присвоить подходящие значения и свойства. Множество моделей предоставляют контекстно чувствительную подсказку, так что информацию об индивидуальных свойствах можно увидеть, выделив поле и нажав **F1**.
- Микропроцессорный исходный код можно применить под управлением PROTEUS VSM, используя команды раздела *Source* основного меню. Не забудьте присвоить объектный код (hex-файл) компоненту на схеме.
- Соединяйте схему, щелчками по выводам.
- Удаляйте объекты двойным щелчком правой клавиши мышки.
- Перемещайте компоненты, выделив их и перетаскивая левой клавишей мышки.
- Щёлкните по клавише PLAY на панели управления анимацией, чтобы запустить симуляцию.

При использовании виртуальных инструментов или моделей микропроцессоров всплывающее окно, относящееся к компоненту, может отображаться с помощью команд раздела *Debug* основного меню.

ЭФФЕКТЫ АНИМАЦИИ

Обзор

Кроме эффектов для активных компонентов в схеме, можно создать ряд других эффектов анимации, что поможет вам изучать операции со схемами. Эти опции можно задать, используя команду *Set Animation Options* в разделе *System*. Все установки сохраняются с проектом.

Состояние логических выводов

Pin Logic States — эта опция отображает цветные квадратики на каждом выводе, который

подключён к цифровой или смешанной цепи. По умолчанию квадратики синие для логического 0, красные для логической 1 и зелёные для плавающего состояния. Цвета могут быть изменены командой *Set Design Defaults* раздела *Template* основного меню.

Установка опции довольно умеренно нагружает симулятор, но может быть очень и очень полезна, когда используется совместно с точками останова и в пошаговом режиме, поскольку позволяет вам изучить состояние выходных выводов порта контроллера при каждом шаге по коду программы.

Отображение напряжения на проводах цветом

Show Wire Voltage as Colour — эта опция вызовет то, что любой провод, как часть аналоговой или смешанной цепи, будет отображаться цветом, показывающим напряжение на нём. По умолчанию цветовой спектр задан от синего при –6V через зелёный при 0V к красному при +6V. Напряжение можно изменить в диалоге *Set Animation Options*, а цветовую гамму командой *Set Design Defaults* в разделе *Template*.

Установка этой опции не очень сильно загружает симулятор, но может быть очень эффективна, помогая понять происходящее в схеме, особенно в сочетании с опцией *Show Wire Current as Arrows* (показывать ток стрелками).

Показывать ток в проводах стрелками

Show Wire Current as Arrows — эта опция вызывает появление стрелок на всех проводах с током. Направление стрелки отражает направление тока, и она появляется, если амплитуда тока превышает пороговое значение. Пороговое значение по умолчанию — 1uA, хотя это можно изменить в диалоге *Set Animation Options*.

Вычисление тока в проводах вызывает вставку источника нулевого напряжения в каждый сегмент провода в схеме (отдельно от внутренних моделей), и это может создать большое число дополнительных узлов. Соответственно нагрузка на симулятор может стать значительной. Однако стрелки очень полезны в технике основ электричества и электроники, где схемы, как правило, достаточно просты.

УПРАВЛЕНИЕ ШАГОМ ВРЕМЕНИ ПРИ АНИМАЦИИ

Обзор

Два параметра управляют тем, как выполняется интерактивная симуляция в реальном времени. *Animation Frame Rate* (показатель кадра анимации) определяет, сколько раз в секунду будет обновляться экран, а *Animation Timestep* (шаг времени анимации) определяет, как много симуляции произойдёт в каждом из кадров. Для операций реального времени, timestep должно быть установлено обратно к показателю кадра.

Количество кадров в секунду (Frames Per Second)

Обычно нет необходимости подстраивать показатель кадров, поскольку значение по умолчанию 20 кадров в секунду даёт гладкую анимацию без перегрузки (значительной) графики современных PC. Но иногда полезно замедлить анимацию при отладочных операциях компьютерных моделей.

Шаг времени на кадр (Timestep Per Frame)

Animation Timestep можно использовать, чтобы сделать быстрое выполнение симуляции медленнее, или чтобы медленное выполнение ускорить. Для операций реального времени timestep должно быть установлено обратно к показателю кадра (frame rate). Конечно, введённое значение timestep всегда будет таким, каким оно желательно. Выбор зависит от того, достаточно ли мощности CPU для вычислений, необходимых при симуляции событий внутри времени, отведённого для каждого кадра. Процессор загружает фигуры, отображаемые в процессе анимации, представляя отношение этих двух времён. Если доступной мощности процессора не хватает, загрузка CPU будет 100%, и выполняемый шаг времени на кадр будет уменьшаться.

Время одного шага (Single Step Time)

Оставшийся параметр управления временем шага — это *Single Step Time*. Это время, за которое симуляция будет выполнена, когда кнопка одного шага на панели анимации будет нажата.

ПОЛЕЗНЫЕ СОВЕТЫ

Шкала времени схемы

Интерактивная симуляция будет обычно выглядеть в реальном времени, так что не используйте схему с тактовой частотой 1 МГц или синусоидальным сигналом 10 кГц, пока не зададите значение *Timestep per Frame* в диалоге *Animated Circuits Configuration*.

Если вы собираетесь симулировать что-то, что работает очень быстро, то держите в памяти некоторые детали:

- Мощность процессора не бесконечна, только некоторое количество времени симуляции может уложиться в фиксированное количество реального времени. PROTEUS VSM разработан, чтобы в любом случае поддерживать показатель кадра анимации (количество кадров в секунду) и сокращать некоторые кадры, которые не помещаются в заданное время. Результатом будет то, что очень быстрые схемы будут симулироваться медленнее (относительно реального времени), но ровнее.
- Модели аналоговых компонентов симулируются намного медленнее, чем цифровые. На быстрых компьютерах вы можете симулировать цифровые схемы, работающие на частоте в несколько мегагерц в реальном времени, но аналоговые электронные схемы будут работать только до частоты 10-20 кГц.

  Следовательно, не разумно пытаться симулировать тактовый генератор для цифровой схемы в аналоговой области, вместо этого используйте *Digital Clock* (цифровой тактовый генератор) и установите флажок *Isolate Before* для него, чтобы исключить симуляцию аналогового генератора.

Масштаб напряжений

Если вы решили использовать расцвеченные провода для показа напряжения узлов, вы должны уделить некоторое внимание диапазону напряжений в вашей схеме. Предопределённый диапазон для показа — это +/-6V, так что, если схема работает с

совершенно другими напряжениями, вам понадобится изменить значение *Maximum Voltage* в диалоге *Animated Circuits Configuration*.

Заземление

PROSPICE пытается определить чувствительную точку заземления для любой активной цепи, которая осталась без специально заданного земляного контакта. На практике обычно выбирается средняя точка батарейки или центральный вывод, если цепь имеет раздельное питание. Из этого следует, что положительный контакт батарейки будет над землёй, а отрицательный под ней, соотносясь с красным и синим цветом проводов. Однако, если такое поведение не то, что требуется, вы всегда можете воспользоваться возможностью явного задания точки заземления, используя контакт земли в ISIS.

Точки высокого импеданса

Автоматическое логическое заземление также проверяется для любого вывода устройства, которое не подключено к земле, и автоматически будет добавлен резистор большого сопротивления, чтобы выполнить сходимость для SPICE симуляции. Это означает, что плохо оформленные компоненты схемы (то есть, с не соединёнными или частично соединёнными элементами) будут, в общем, симулироваться — хотя временами это даст странные результаты.

ВИРТУАЛЬНЫЕ ИНСТРУМЕНТЫ

ВОЛЬТМЕТРЫ И АМПЕРМЕТРЫ

Несколько интерактивных вольтметров и амперметров предлагаются в библиотеке активных устройств. Они оперируют в реальном времени и могут быть подключены к схеме, как любые другие компоненты. Когда симуляция начинается, они отображают напряжение на их выводах или ток, протекающий через них, в удобном для чтения формате.

Предложенные модели покрывают FSD в 100, 100m и 100u с разрешением в 3 значащих цифры и максимальным числом в два десятичных разряда. Таким образом объект *VOLTMETER* может отображать значения от 0.01V до 99.9V , а *AMMETER-MILLI* от 0.01mA до 100mA и т.д.

Модели вольтметров поддерживают внутреннее сопротивление до 100 МОм, но его можно изменить, редактируя компонент обычным образом. Если оставить значение пустым, это убирает сопротивление модели.

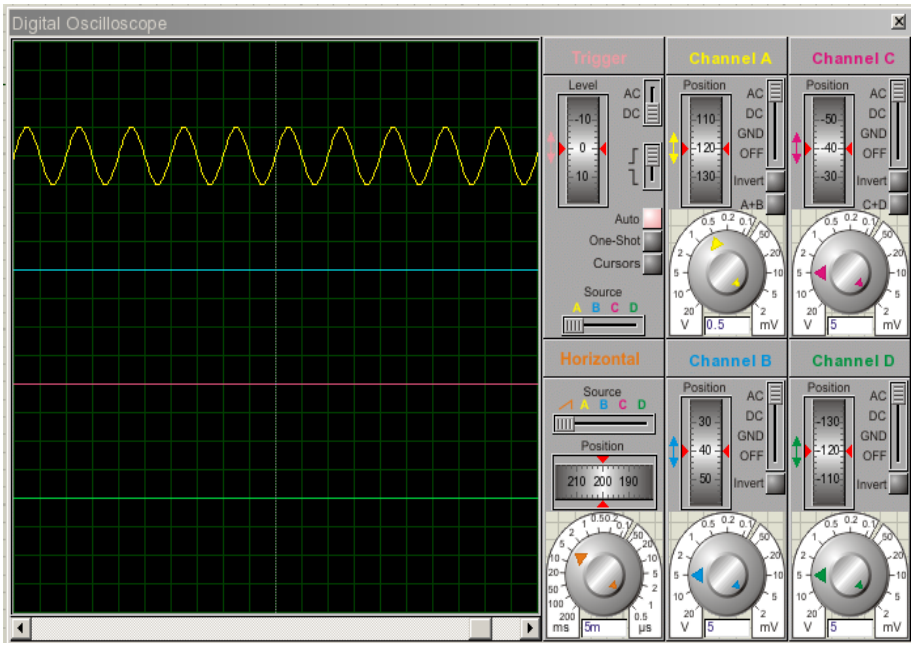
Вольтметры и амперметры переменного тока отображают RMS значения, интегрированные через определимую пользователем постоянную времени.

ОСЦИЛЛОГРАФ

Обзор

VSM осциллограф выполнен как стандартный для всех версий ProSPICE и базовых моделей аналоговых устройств, и имеет следующую спецификацию:

- Четыре канала, X-Y операции.
- Усиление канала от 20V/div до 2mV/div.
- Базовые времена от 200ms/div до 0.5us/div.
- Автоматический уровень переключения, заданный для любого канала.
- AC или DC входы.



Использование осциллографа

Чтобы отобразить аналоговый сигнал:

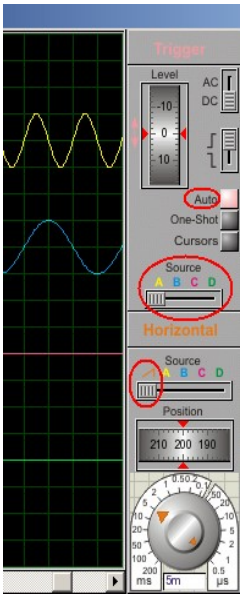
1. Щёлкните по кнопке Virtual Instruments Mode; выберите в окне выбора OSCILLOSCOPE; щёлкните на свободном месте рабочего поля, чтобы добавить символ осциллографа; добавьте соединение с точкой схемы, в которой хотите наблюдать сигнал.
2. Запустите интерактивную симуляцию, нажав на кнопку Play панели управления анимацией. Появится осциллограф.
3. Задайте время развёртки и подходящее напряжение для удобного наблюдения. Вам следует подумать о частоте сигнала, и конвертировать её в обратную величину времени развёртки.
4. Если отображаемый сигнал имеет постоянную составляющую, выберите режим AC на одном или всех каналах.
5. Подстройте усиление по Y и позицию Y, чтобы сигнал имел удобный для наблюдения вид. Если сигнал состоит из переменного напряжения малой амплитуды на большой постоянной составляющей, вам необходимо включить конденсатор между точкой наблюдения и осциллографом, так как, изменение позиции по Y имеет ограниченные возможности.
6. Решите, каким каналом вы будете запускать развёртку.
7. Поверните ручку уровня переключения (Trigger), пока на дисплее не отобразится требуемый участок входного сигнала. Он фиксируется на переднем фронте, если дисковая шкала повёрнута вверх, и на заднем фронте, если вниз.

См. «Дисковая шкала» далее, где подробнее описано, как этим пользоваться.

Режимы операций

Осциллограф может работать в режимах, обозначенных ниже:

- Отдельные каналы — при этом развёртка запускается автоматически, выбранным каналом.



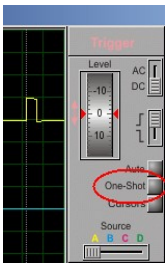
Уровень запуска (Level) определяет, при каком уровне входного сигнала выбранного канала происходит запуск развёртки (панель Trigger).

Положение каналов по оси X на экране определяется дисковой шкалой Position.

Время развёртки определяется ручкой выбора масштаба (время/деление) в нижней части блока управления развёрткой.

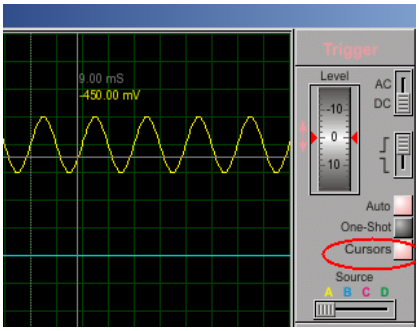
В этом режиме удобно наблюдать за фазовыми и временными сдвигами сигналов (например, на входе и на выходе).

- Снимок экрана — для этого режима на панели Trigger используется кнопка One-Shot.

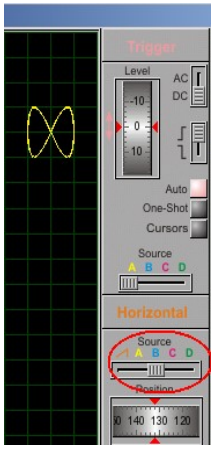


В сложных случаях, когда не удаётся синхронизировать изображение, когда «картинка» на экране не статична, можно произвести снимок экрана и разобраться с происходящим.

- Последняя кнопка в этом ряду, Cursors, позволяет уточнить величину сигнала и время его появления. При нажатой кнопке на экране появляются курсоры, перемещая которые в нужные точки, можно уточнить данные.



- Режим разностного сигнала позволяет наблюдать, например, фигуры Лиссажу, если включить развёртку в этот режим.



Четыре канала осциллографа имеют разные цвета, которым соответствуют цвета надписей на панелях и переключателях, что облегчает поиск ручек управления и положения переключателей для каждого из каналов.

Каждый из каналов может быть выключен, достаточно перевести его переключатель в положение OFF. Это даёт возможность более внимательно просматривать сигналы, если их один или два, используя смещение положения луча работающих каналов с помощью дискового регулятора Position.

Запуск (Triggering)

Осциллограф VSM поддерживает механизм автоматического запуска, что позволяет синхронизировать базовое время с входящим сигналом.

- Какой входной канал используется для запуска, определяется переключателем на панели Horizontal.
- Диск переключения (Level на панели Trigger) последовательно вращается на 360 градусов и задаёт уровень, на котором происходит переключение. Фронт переключения определяется переключателем рядом с диском.
- Если переключение не обнаруживается в течение одного базового интервала времени, базовый интервал будет «освобождён».

Входное соединение

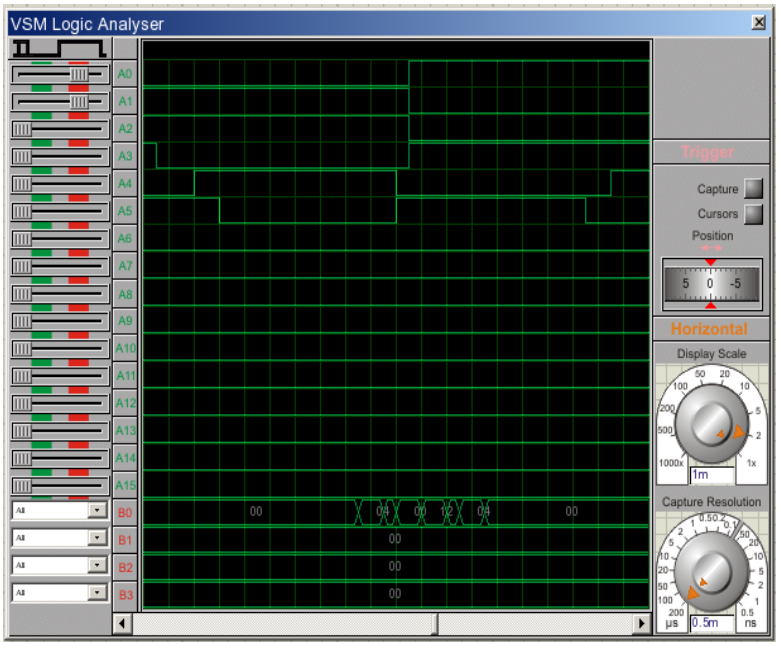
Каждый входной канал может быть непосредственно подключён к точке наблюдения (DC соединение) или подключён через конденсатор (AC соединение). Последний способ полезен тогда, когда входной сигнал имеет маленький уровень переменной составляющей и большой уровень постоянной.

Вход может также кратковременно подключаться к земле (переключатель в положении GND) при настройке координатной сетки перед измерением.

ЛОГИЧЕСКИЙ АНАЛИЗАТОР

Обзор

Логический анализатор (Logic Analyser) представлен как стандартный в PROTEUS VSM Professional, но как дополнительная опция в PROTEUS VSM Lite.

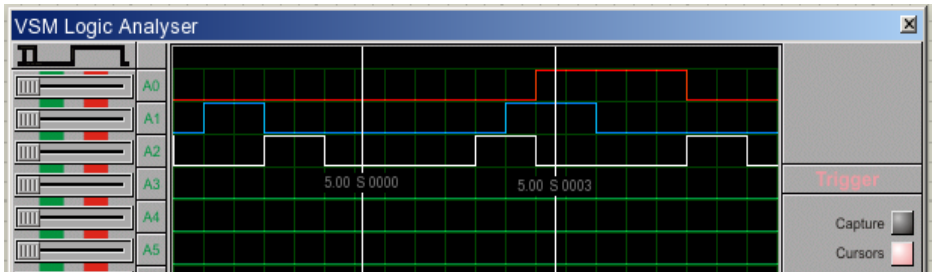


Логический анализатор оперирует последовательно записанными в большой *буфер захвата* входными цифровыми данными. Это процесс отбора, так что есть подстройка разрешения (resolution), которая определяет самый короткий импульс, который может быть записан. На панели запуска (Trigger) есть кнопка **Capture**, которой запускается процесс захвата данных, а спустя некоторое время после выполнения условий переключения останавливается; кнопка меняет свой цвет при записи и после её завершения. Результат, содержимое *буфера захвата* и до, и после переключения, отображается на дисплее. Поскольку *буфер захвата* очень большой (10000 образцов в данном случае), предусмотрено масштабирование и панорамирование изображения. И, наконец, измерительные маркеры (кнопка **Cursors**) позволяют точно определить параметры импульсов и т.п.

Логический анализатор имеет следующую спецификацию:

- 16 x 1 бит и 4 x 8 бит шины трассировки.
- 10000 x 24 бит буфер захвата.
- Разрешение при захвате от 200us на образец до 0.5ns с соответствующим временем захвата от 2s до 5ms.
- Масштаб отображения от 1000 образцов на деление до 1 образца на деление.
- Переключение с комбинацией AND входных состояний и/или фронтов и значениями на шине.

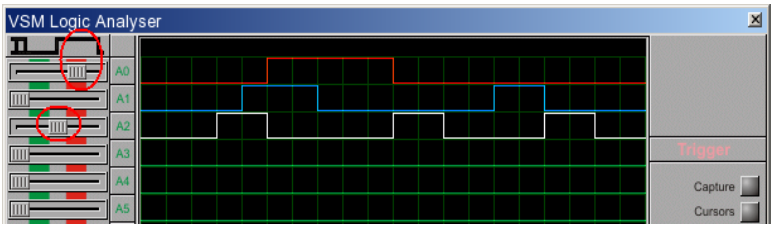
- Позиционирование переключения на 0, 25, 50, 75 и 100% *буфера захвата*.
- Курсоры для точных измерений времени.
- Возможность изменить цвета отображаемых кривых, курсоров, текста и т.д. (щелчок правой клавишей мышки по дисплею и выбор из выпадающего меню *Colours Setup*).



Использование логического анализатора

Чтобы захватить и отобразить цифровые данные:

1. Щёлкните по иконке *Virtual Instruments Mode*; выберите LOGIC ANALYSER, разместите объект на чертеже, соедините входы анализатора с сигналами, которые вы намерены записать.
2. Запустите интерактивную симуляцию, используя кнопку **Play** на *панели управления анимацией*. Появится окно анализатора.
3. Задайте разрешение, подходящее к вашему примеру. Этим устанавливается самый короткий импульс, который будет записан. Чем лучше разрешение, тем короче время захвата данных.
4. Установите на левой панели требуемые условия переключения. Например, если вы хотите переключать инструмент, когда сигнал, подключённый к каналу 1 в высоком состоянии, а сигнал, подключённый к каналу 3 переходит в высокое состояние, вы должны задать первый как «High», а третий как «Low-High».



5. Решите, хотите ли вы видеть данные преимущественно до или после выполнения условий переключения, установите шкалу Position в нужное положение переключения.
6. Когда вы готовы, нажмите кнопку **Capture**. Индикатор загорится, и анализатор будет захватывать входные данные непрерывно, пока не обнаружатся входные условия для переключения триггера. Когда это случится, индикатор станет зелёным, захват данных

продолжится до тех пор, пока не заполнится часть буфера после переключения. Индикатор погаснет, а на дисплее отобразятся анализируемые кривые.

Панорамирование и масштабирование

Поскольку *буфер захвата* удерживает 10000 образцов, а отображение имеет ширину только в 250 пиксел, появляется необходимость в панорамировании и масштабировании *буфера захвата*. *Display Scale* определяет количество образцов на деление, а полоса прокрутки позволяет перемещать изображение влево и вправо.

Заметьте, что считанные данные под *Display Scale* отображают текущее время на деление в секундах, но не являются актуальными установками собственно шкалы. Время деления вычисляется умножением установок шкалы на разрешение.

Измерения

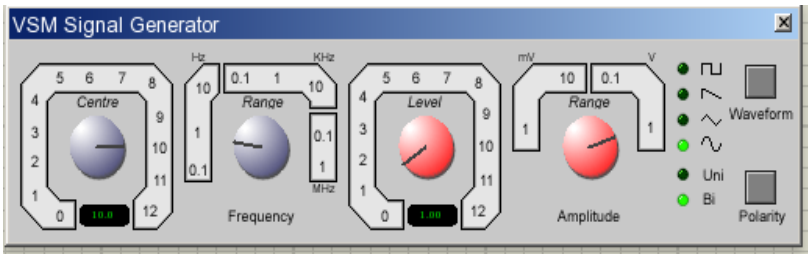
Для точного измерения времени используются маркеры. Каждый маркер можно размещать, используя соответственно окрашенную шкалу. Показания отображают точку времени, отмеченную маркером, относительно времени переключения, а показания Delta A-B — разницу между маркерами.

См. раздел «Дисковые шкалы», чтобы подробнее ознакомиться с этим управлением в логическом анализаторе.

СИГНАЛ-ГЕНЕРАТОР

Обзор

VSM Signal Generator входит как стандартный и для ProSPICE Professional, и для ProSPICE Lite.



VSM Signal Generator представляет собой простой функциональный аудио генератор со следующими возможностями:

- Генерирует выходные импульсы прямоугольной, пилообразной, треугольной и синусоидальной формы.
- Выходная частота 0-12MHz в 8 диапазонах.
- Выходная амплитуда 0-12V в 4 диапазонах.
- Входы для амплитудной и частотной модуляции.

Использование сигнал генератора

Чтобы установить простой аудио сигнал:

1. Щёлкните по иконке *Virtual Instruments Mode*; выберите SIGNAL GENERATOR в окне выбора объектов, поместите его на схему и соедините его выход со входом схемы. В большинстве случаев (то есть, когда схема, которую вы рисуете, требует баланса входного сигнала), вам понадобится заземлить клемму -ve генератора. И самое простое — использовать заземление (правая клавиша мышки, в выпадающем меню *Place-Terminal-GROUND*).

   Входы амплитудной и частотной модуляции можно оставить без подключения, пока оно вам не потребуется.

2. Запустите интерактивную симуляцию, используя кнопку **Play** на *панели управления анимацией*. Появится всплывающее окно генератора.

3. Установите частотный диапазон, который вам подходит. Значение диапазона показывает частоту, которая генерируется, когда центр верньера управления находится в положение 1.

4. Установите амплитуду сигнала, подходящую для вашей схемы. Значение диапазона показывает амплитуду, которая генерируется, когда верньер управления установлен в положение 1. Значение амплитуды представляет пиковый уровень выхода.

5. Нажимайте кнопку **Waveform**, пока индикатор рядом с изображением подходящей формы сигнала не засветится.

Использование входов AM & FM Modulation

Модель сигнал-генератора поддерживает и амплитудную, и частотную модуляцию выходного сигнала. Оба входа, амплитудной и частотной, модуляции имеют следующие возможности:

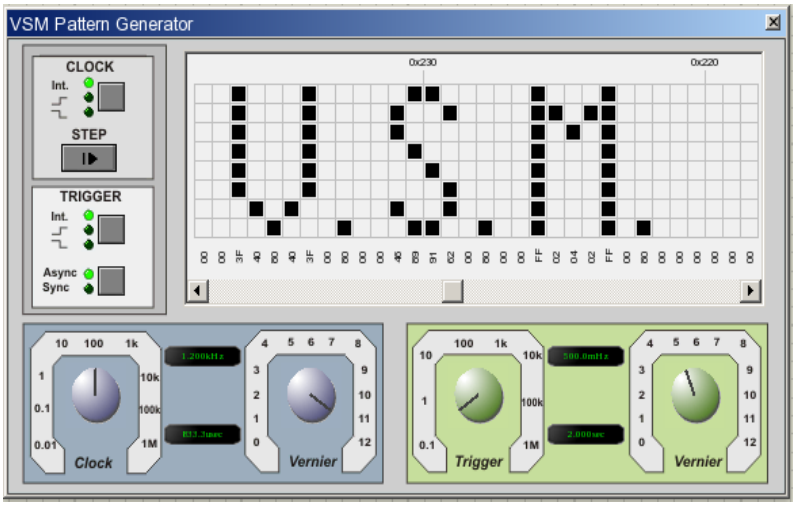
- Усиление входа модуляции в терминах Hz/V или V/V задаются с помощью управления Frequency Range и Amplitude Range, соответственно.
- Входное напряжение модуляции ограничено в пределах +/- 12V.
- Входы модуляции имеют бесконечное входное сопротивление.
- Напряжение на входе модуляции добавляется к установкам управления верньером до умножения установками диапазона, чтобы определить мгновенное значение частоты амплитуды.

Например, если частотный диапазон выбран в 1KHz, а частотный верньер установлен в 2.0, тогда уровень 2V частоты модуляции даст выходную частоту 4kHz.

ЦИФРОВОЙ ГЕНЕРАТОР ШАБЛОНА

Обзор

VSM Pattern Generator — это цифровой эквивалент аналогового сигнал-генератора и предоставляется как стандартный для всех профессиональных версий симуляторов Proteus.



VSM Pattern Generator предназначен для 8-битовых шаблонов до 1 Кбайта и поддерживает следующие возможности:

- Запускается в графике, основанной на интерактивном режиме.
- Имеет внутренний и внешний режим тактирования и переключения.
- Верньерную подстройку для тактовой частоты и шкалы переключения.
- Режим отображения шестнадцатеричной и десятичной сетки.
- Непосредственный ввод значения для большей точности.
- Загрузку и сохранение скрипта шаблона.
- Ручную спецификацию длительности периода шаблона.
- Пошаговое управление позволяет вам использовать шаблон постепенно.
- Окно указателя позволяет вам видеть непосредственно, где вы находитесь на сетке.
- Возможность внешней поддержки шаблона в его текущем состоянии.
- Команды Block Editing на сетке для облегчения конфигурирования шаблона.

Использование генератора шаблонов

Вывод шаблона Pattern Generator'a в режиме интерактивной симуляции:

1. Щёлкните по иконке *Virtual Instruments Mode*; выберите PATTERN GENERATOR в окне выбора объектов, поместите его на схему и соедините с остальной схемой.
2. Инициализируйте интерактивную симуляцию, нажав на кнопку **Play** на *панели управления анимацией*. Появится окно Pattern Generator.
3. Задайте шаблон, который вы хотите вывести, с помощью левой клавиши мышки, щелчки которой по квадратикам сетки переключают их логическое состояние.
4. Решите, будете ли вы тактировать генератор внутренним или внешним образом, установите режим с помощью кнопки **Clock**, последовательно нажимая её до свечения индикатора, соответствующего вашему выбору.
5. Если вы выбрали режим внутреннего тактирования, подстройте частоту с помощью шкалы Clock.
6. Решите, будете ли вы переключаться внутренним или внешним событием, и используйте кнопку **Trigger** для выбора соответствующего режима. Если вы будете управлять переключением от внешнего источника, вам нужно подумать, будет ли переключение синхронным или асинхронным с тактовым генератором.
7. Если вы решили переключаться внутренним генератором, подстройте шкалу Trigger к нужной частоте.
8. Нажмите кнопку **Play** на *панели управления анимацией*.
9. Чтобы использовать шаблон в одном такте, нажмите кнопку паузы на *панели управления анимацией*, а затем нажимайте кнопку **Step** слева от сетки.

📖 См. раздел «Дисковые шкалы» далее, чтобы лучше понять, как пользоваться этим в Pattern Generator.

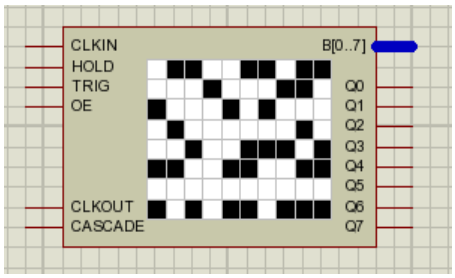
📖 См. раздел «Режимы переключения» ниже, где детально описаны разные режимы переключения.

📖 См. раздел «Дополнительные функции» ниже, где больше информации о конфигурировании и использовании Pattern Generator.

Вывод шаблона Pattern Generator'a в режиме графической симуляции

1. Задайте схему обычным образом.
2. Поставьте пробники на схему в интересующих вас точках и добавьте эти пробники на график.
3. Щёлкните правой клавишей мышки по Pattern Generator на схеме и вызовите диалог *Edit Component*.
4. Сконфигурируйте опции *Trigger* и *Clock*.
5. Загрузите нужный шаблон в поле *Pattern Generator Script*.
6. Покиньте диалоговую форму Pattern Generator и нажмите пробел, чтобы запустить симуляцию.

Выводы компонента Pattern Generator



Выводы выхода данных (выход с тремя состояниями, Q0-Q7, B[0-7])

Pattern Generator может выводить либо на шину и/или на индивидуальные выводы.

Выводы тактового генератора (CLKOUT)

Когда Pattern Generator тактируется внутренним генератором, вы можете сконфигурировать этот вывод для отображения импульсов внутреннего генератора. Это задаётся, как свойство, и может изменяться через диалоговую форму *Edit Component*. По умолчанию эта опция не установлена, поскольку это слегка затрудняет выполнение, обычно на высоких частотах.

Выход Cascade

Cascade Pin переходит в высокое состояние, когда первый бит шаблона выполняется, и остаётся в высоком состоянии, пока не пришёл второй бит (на один такт позже). Это означает, что он в высоком состоянии для первого такта при старте симуляции и вновь для первого цикла последующего сброса.

Вывод триггера (вход)

Этот вход используется для подачи внешнего импульса запуска в Pattern Generator. Есть четыре режима переключения, которые обсуждаются в разделе «Режимы сброса».

Вывод CLKIN (вход)

Этот вход используется для подключения внешнего тактового генератора к Pattern Generator. Есть два режима внешнего тактирования, которые обсуждаются более подробно в разделе «Режимы тактирования».

Вывод Hold (вход)

Этот вывод, когда переходит в высокое состояние, может использоваться для остановки (паузы) Pattern Generator. Шаблон останется в той же точке, пока не будет освобождён вывод hold. Для внутреннего тактового генератора и/или триггера тактирование будет возобновлено относительно точки, в которой было приостановлено. Например, при 1Hz внутреннего генератора, если шаблон остановлен на времени 3.6 секунды и запущен в 5.2 секунды, тогда следующий спад импульса будет в 5.6 секунд.

Вывод разрешения выхода (вход)

Этот вывод (OE) должен быть установлен в высокое состояние, чтобы разрешить работу выводам выхода. Если это вывод не в высоком состоянии, тогда Pattern Generator, продолжая работать с заданным шаблоном, не будет передавать шаблон на выходные выводы.

Режимы тактирования

Внутреннее тактирование

Внутреннее тактирование переключается отрицательным фронтом. Из чего следует, что состояние импульсов будет «низкое-высокое-низкое» за один такт.

Внутреннее тактирование может быть задано либо до симуляции через диалог *Edit Component*, либо в процессе симуляции с помощью кнопки выбора режима **Clock**.

Вывод CLKOUT, когда разрешён, отображает внутренние импульсы. По умолчанию эта опция не выбрана, из-за возможных затруднений (обычно на высокой частоте) в работе, но может быть установлена через диалог Edit Component генератора шаблонов.

Внешнее тактирование

Есть два режима внешнего тактирования — отрицательным фронтом (состояния низкое-высокое-низкое) и положительным (высокое-низкое-высокое).

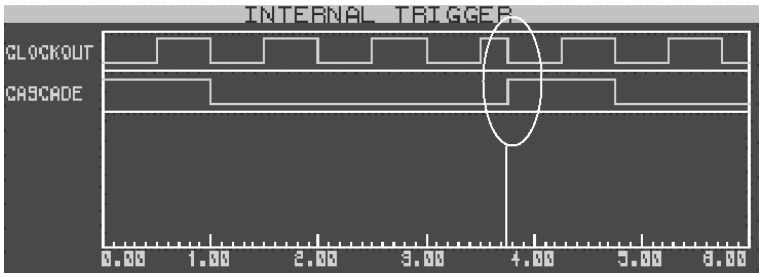
Чтобы тактировать внешним генератором, соедините выходные импульсы с выводом CLKIN и выберите один из двух режимов.

Как и при внутреннем тактировании, вы можете изменить режим либо редактированием компонента до симуляции, либо с помощью кнопки выбора режима **Clock** при паузе в симуляции.

Режимы переключения

Внутренний триггер

Режим Internal Trigger Mode (внутреннее переключение) в Pattern Generator переключает шаблон в заданных интервалах. Если тактирование внутреннее, тактовый импульс в этот момент сбрасывается. Это поведение показано на рисунке ниже.

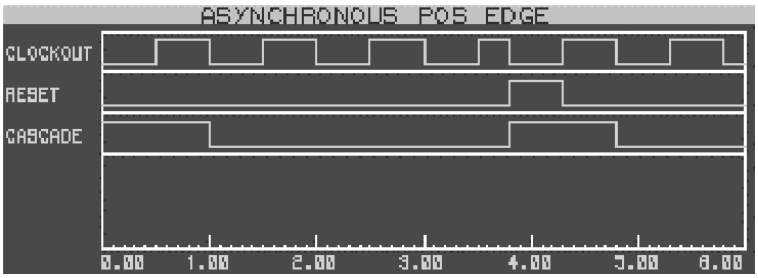


Внутренний генератор работает на частоте 1Гц и время внутреннего триггера установлено в 3.75 сек. Вывод Cascade в высоком состоянии, когда первый бит шаблона поступает на выходы и в низком всё остальное время.

Заметьте, что в момент переключения внутренний тактовый генератор сбрасывается. Первый бит шаблона выводится на выходные выводы (как показывалось при переходе вывода Cascade в высокое состояние).

Переключение внешним асинхронным положительным фронтом

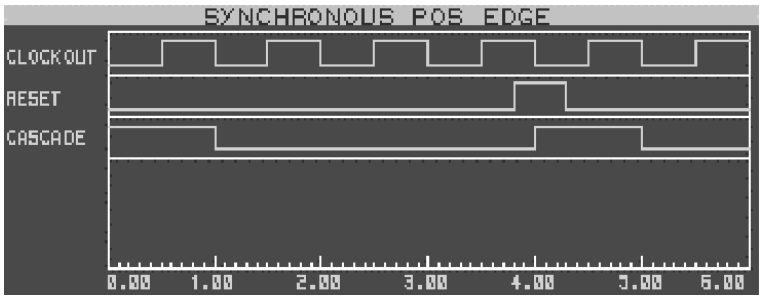
Переключение задаётся переходом положительным фронтом на выводе Trigger. Переключение происходит немедленно и следующий фронт тактового импульса будет восходящий в момент bitclock/2, следующий за временем сброса, как показано ниже.



Внутренний генератор работает на частоте 1Гц и вывод триггера переходит в высокое состояние в момент 3.75 сек. Немедленно на положительном фронте вывода триггера тактовый генератор сбрасывается и первый бит шаблона появляется на выходах.

Переключение внешним синхронным положительным фронтом

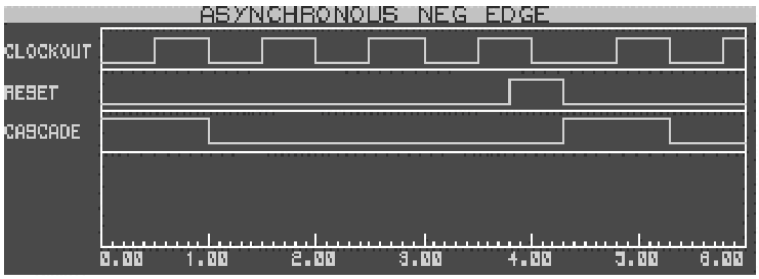
Переключение задаётся переходом положительным фронтом на выводе Trigger. Триггер защёлкивается и будет синхронизирован следующим спадающим фронтом тактового импульса, как показано ниже.



Внутренний генератор работает на частоте 1Гц. Заметьте, что на тактовом генераторе не сказывается работа триггера, и что переключение на спадающем фронте такта следует за положительным фронтом импульса.

Переключение внешним асинхронным отрицательным фронтом

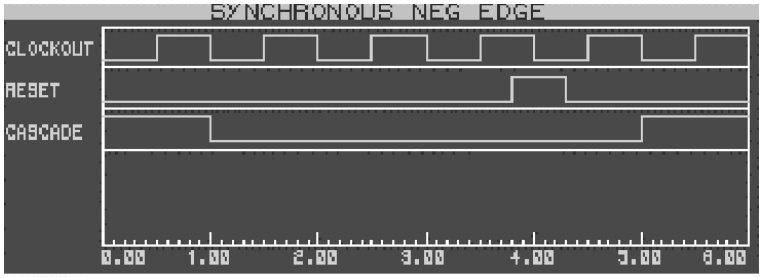
Триггер устанавливается в режим переключения отрицательным фронтом на выводе Trigger. Переключение происходит немедленно и первый бит шаблона выводится на выходные выводы.



Внутренний генератор работает на частоте 1Гц. Можно видеть, что тактовый генератор сбрасывается при отрицательном фронте импульса переключения, а первый бит шаблона выводится в этот момент.

Переключение внешним синхронным отрицательным фронтом

Переключение задано отрицательным фронтом на выводе Trigger. Триггер защёлкивается и действует синхронно со следующим спадающим фронтом тактового генератора, как показано ниже.

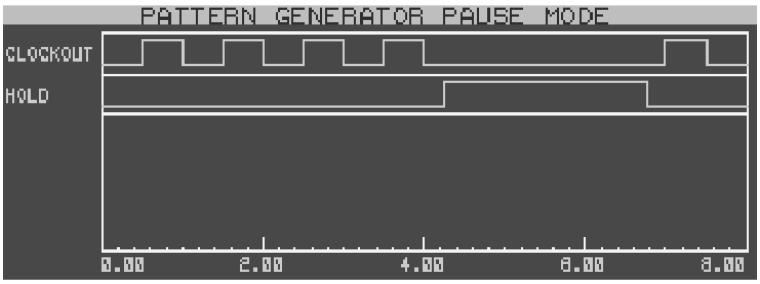


Внутренний генератор работает на частоте 1Гц. Заметьте, что переключение имеет место на спаде импульса переключения, и шаблон не сбрасывается, пока следом не приходит спадающий фронт импульса тактового генератора.

Внешнее удержание

Удержание шаблона в его текущем состоянии

Если вам нужно удержать шаблон на некоторое время, вы можете сделать это с помощью вывода Hold, который должен быть установлен в высокое состояние на время паузы, которая вам нужна. Перевод вывода Hold в низкое состояние синхронно вернёт движение шаблона, если вы используете внутренний тактовый генератор. Вместе с тем, если сигнал на выводе Hold переходит в высокое состояние на середине тактового цикла, тогда при его переходе в низкое состояние следующий бит появится на выходных выводах на полтакта позже.

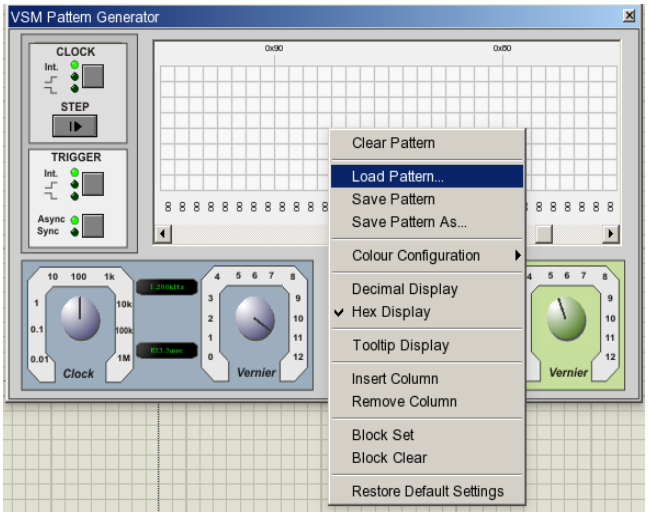


Когда сигнал на выводе Hold переходит в высокое состояние, внутренний тактовый генератор останавливается. При переходе вывода Hold в низкое состояние тактовый генератор запускается относительно точки в цикле, где он был остановлен.

Дополнительная функциональность

Загрузка и сохранение скрипта шаблона

Pattern Scripts могут загружаться или сохраняться, если щёлкнуть правой клавишей мышки по сетке при интерактивной симуляции и выбрать команду из выпадающего меню.



Это полезно, когда выполненный шаблон предполагается использовать в других проектах.

Скрипты шаблонов — это просто текст и простой, разделённый запятыми, список байт, где каждое значение байта представляет колонку на сетке. Любая линия, начинающаяся с точки с запятой, трактуется как линия комментария и игнорируется анализатором программы. По умолчанию формат байта шестнадцатеричный, хотя, если вы создаёте свой собственный скрипт, вы можете вводить десятичные значения, двоичные или шестнадцатеричные.

Установка специальных значений для шкал

Вы можете задавать внешние значения и для бит, и для частоты триггера после двойного щелчка мышки по соответствующей шкале. Этим вызывается появление плавающего окна редактирования, в которое вы можете ввести значение. По умолчанию введённое значение рассматривается как частота, но вы можете задать значение в секундах или как дробь, добавив подходящий суффикс к значению (sec, ms и т.д.). Дополнительно, если вы захотите, чтобы триггер точно работал кратно битам тактовой частоты, вы можете добавить суффикс «bits» к нужному множителю (то есть, 5bits).

Для подтверждения ввода нажмите клавишу **Enter** или, если решили отказаться от ввода, нажмите клавишу **Escape** или, просто, щёлкните где-нибудь в окне Pattern Generator.

Эти значения можно также задать до симуляции через соответствующие свойства в диалоговой форме *Edit Component*.

Установка специальных значений для сетки шаблона

Вы можете задать специальные значения для любой колонки на сетке, щёлкнув левой клавишей мышки по тексту, отображающему текущее значение для этой колонки. Появится плавающее текстовое окно, в которое вы можете вписать нужное значение. В можете задать значение в десятичном (то есть, 135), шестнадцатеричном (0xA7) или двоичном (0b10110101) виде.

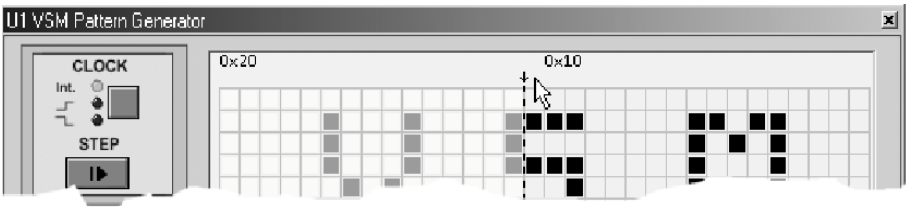
Для подтверждения ввода нажмите клавишу **Enter**, для отказа клавишу **Escape** или щёлкните левой клавишей мышки где-нибудь в окне *Pattern Generator*.

Для удобства вы можете использовать задание в колонке через горячие клавиши **CTRL+1**

клавиатуры и очистить колонку клавишами **CTRL+SHIFT+1**, когда мышка находится на нужной колонке.

Задание длины периода шаблона вручную

Вы можете вручную задать период, щёлкнув левой клавишей мышки над сеткой и колонкой, где вы хотели бы, чтобы шаблон заканчивался. Чтобы снять период, щёлкните правой клавишей мышки один раз в том же месте. Всё это показано ниже.



Мышка расположена в области, где может быть задан нужный период. Мы можем видеть, что метка периода показывает точку прерывания шаблона, и что всё слева от выбранного места побледнело, чтобы показать, шаблон будет остановлен у метки периода.

Пошаговое выполнение в продвижении по шаблону

Кнопка **STEP** может использоваться для продвижения при симуляции на время периода, эквивалентное заданному bitclock либо внутреннего, либо внешнего тактового генератора. Симуляция будет идти, пока не завершится следующий цикл тактового генератора, а затем вновь остановится.

Изменение режима отображения сетки

Отображение сетки может переключаться между шестнадцатеричным и десятичным режимом. Это можно сделать, либо щёлкнув правой клавишей мышки по сетке и выбрав нужную опцию из меню, либо используя горячие клавиши **CTRL+X** (шестнадцатеричное отображение) и **CTRL+D** (десятичное отображение).

Задание выхода

Выделите элемент схемы *Pattern Generator*, чтобы войти в диалог *Edit Component*. Свойство внизу (*Output Configuration:*) позволяет вам конфигурировать, как вы хотите видеть шаблон: и на шине, и на выводах; только на выводах; только на шине.

Дисплей указателя контекстного окна

Вы можете разрешить указатель контекстного окна, который будет следовать за мышкой, показывая информацию о текущий строке и столбце. Это можно переключать, включая и выключая, либо через контекст мышки по щелчку правой клавиши, либо через горячие клавиши **CTRL+Q**. Заметьте, что режим указателя отменяется в процессе *Block set* или *clear*.

Редактирование блока

Вы можете использовать команды Block Set и Block Clear, что поможет вам быстро сконфигурировать сетку к нужному шаблону. Это доступно через контекстное меню по щелчку правой клавиши мышки или через горячие клавиши (**CTRL+S** для *Block set* и **CTRL+C** для *Block clear*). Заметьте, что команды Block Editing запрещены в режиме указателя (tooltip mode).

ПОЛЬЗОВАТЕЛЬСКИЕ ЭЛЕМЕНТЫ ИНТЕРФЕЙСА VSM

Дисковые шкалы

VSM Virtual Instruments используют управляемые мышкой дисковые шкалы (ручки) для настройки некоторых параметров. Процедура настройки следующая.

Чтобы установить дисковую шкалу:

1. Установите курсор где-нибудь на шкале.
2. Нажмите левую клавишу мышки и удержите её.
3. Переместите указатель мышки от этой точки вокруг центра шкалы, выписывая дугу, чтобы повернуть ручку к нужному значению шкалы.
4. Шкала будет поворачиваться на угол, следующий за указателем мышки от её центра. Чем дальше вы будете перемещать мышку, тем на большее число градусов повернётся ручка.
5. Отпустите клавишу мышки, чтобы зафиксировать положение шкалы.

РАБОТА С МИКРОПРОЦЕССОРАМИ

ВВЕДЕНИЕ

Комбинация из смешанного режима симуляции и анимации схемы становится наиболее привлекательна, когда используется в контексте систем, основанных на микроконтроллерах. Множество таких систем затрагивают интерфейс пользователя, или, хотя бы, комплексную последовательность внешних событий, которые не могут легко симулироваться вне интерактивного окружения, и PROTEUS VSM в первую очередь был создан, адресуясь к этим проблемам. Следовательно, значительный кусок его функциональности сосредоточен вокруг разработки на основе микропроцессоров.

На практике, редактирование и компиляция исходного кода, интегрируется в окружение проекта так, чтобы вы могли редактировать код и просматривать результат внесённых изменений максимально легко. Исходный файл может быть вызван для редактирования двумя нажатиями клавиш, а симуляция возобновлена с этого места двумя другими.

СИСТЕМА УПРАВЛЕНИЯ ИСХОДНЫМ КОДОМ

Обзор

Система управления исходным кодом поддержана двумя основными функциями:

- Регистрацией файлов исходного кода в ISIS, так чтобы они могли быть вызваны для редактирования без ручного переключения на другое приложение.
- Определением правил для компиляции исходного кода в объектный код. Однажды заданные, эти правила появляются каждый раз при выполнении симуляции, так что объектный код привязывается к датам.

Заметьте, что нет необходимости использовать систему управления исходным кодом для симуляции проекта, основанного на микропроцессорах. Действительно, если вы выбрали ассемблер или компилятор, имеющий свой собственный IDE, вы можете прекрасно работать в нем, а переключаться в Proteus, когда проверяете выполнение программы. Если вы планируете работать таким образом, тогда перейдите к следующему разделу «Использование IDE других производителей».

Добавление исходного кода в проект

Чтобы добавить файл исходного кода в проект:

1. Выберите команду *Add/Remove Source Files* из раздела *Source* основного меню.
2. Выберите *Code Generation Tool* для исходного файла. Если вы планируете использовать новый ассемблер или компилятор впервые, вам нужно зарегистрировать их, используя команду *Define Code Generation Tools*.
3. Щёлкните кнопку **New** и выберите или введите имя для файла исходного кода с помощью селектора файлов. Вы можете ввести текстовое имя файла, если он не существует.

4. Установите флажки, требуемые для задания работы с этим исходным файлом в поле *Flags*. Флажки, необходимые каждый раз при использовании обычных инструментов, могут быть введены при регистрации инструмента.
5. Щёлкните **ОК** для добавления исходного файла в проект.

Не забудьте отредактировать микропроцессор и присвоить имя файлу объектного кода (обычно это hex-файл) в его *PROGRAM* свойстве. ISIS не может сделать этого автоматически, поскольку у вас может быть не один процессор на схеме!

Работа над вашим исходным кодом

Чтобы отредактировать исходный код:

1. Нажмите **ALT-S**.
2. Нажмите порядковый номер файла исходного кода в меню *Source*.

Если вы предпочитаете использовать более развитый текстовый редактор, посмотрите соответствующий раздел.

Чтобы переключиться обратно в ISIS, оттранслируйте (build) исходный код и запустите симуляцию:

1. Из текстового редактора нажмите **ALT-TAB**, чтобы вернуться в ISIS.
2. Нажмите **F12** для выполнения или **CTRL-F12** для начала отладки.

В любом случае ISIS проинструктирует текстовый редактор сохранить свои файлы, проверит дату файлов исходного и объектного кодов, и запустит подходящий инструмент трансляции для создания объектного кода.

Чтобы перетранслировать весь объектный код:

1. Выберите команду *Build All* из раздела *Source* основного меню.

ISIS вызовет все инструменты генерации, требуемые для трансляции в объектный код, безотносительно меток время/дата файлов объектного кода. Командная строка вывода инструмента будет отображена в окне. Этим превосходно поддерживается проверка, что все оттранслировано без ошибок и предупреждений.

Установка инструментов генерации кода других производителей

Некоторое количество свободных ассемблеров и компиляторов может быть установлено в директорию TOOLS с системного CD, и они будут автоматически заданы, как средства генерации кода программой установки PROTEUS. Однако, если вы хотите использовать другие инструменты, вам понадобится команда *Define Code Generation Tools* в разделе *Source* основного меню.

Чтобы зарегистрировать новый инструмент генерации кода:

1. Выберите команду *Define Code Generation Tools* в разделе *Source* основного меню.
2. Щёлкните **New** и используйте селектор файлов для указания пути к исполняемому файлу инструмента. Вы можете также зарегистрировать batch-файлы (командные файлы), как инструмент генерации кода.
3. Введите расширения для файлов исходного и объектного кода. Этим определяется тип файла, по которому ISIS будет определять при решении, запускать ли инструмент для отдельного исходного файла или нет. Если вы установили Always Build, тогда инструмент трансляции запускается всегда, а расширение объектного кода не требуется.
4. Задайте нужную командную строку для инструмента. Используйте %1 для представления файла исходного кода и %2 для файла объектного кода. Вы можете также использовать %$ для пути к директории PROTEUS и %~ для директории, в которой размещается DSN файл.

И хорошо одновременно поместить флаги командной строки, которые нужны для фонового запуска инструмента трансляции (то есть, без паузы для пользовательского ввода), и хорошо бы задать пути к директории *include* файлов заголовков процессоров и т.п.

Если вы хотите использовать уровень отладки исходного кода PROTEUS VSM, вам понадобится Debug Data Extractor для вашего ассемблера или компилятора. Это маленькая программа командной строки, которая выбирает строку объектного кода/исходного из информации кросс-ссылок списка файлов, произведённых ассемблером или компилятором. Мы надеемся поддерживать все популярные ассемблеры и компиляторы, так что проверяйте наш сайт на предмет последней информации.

Если у вас есть DDX программа, введите расширение для файла списка или символьного отладочного файла, который производится инструментом генерации кода, и щёлкните **Browse**, чтобы выбрать путь и имя файла DDX программы.

Использование программы MAKE

В некоторых случаях простые правила, встроенные в ISIS, могут быть недостаточны для обслуживания вашего приложения — особенно, если происходит выполнение множества исходных и объектных файлов, и происходит компоновка. В таких случаях вам нужно использовать внешнюю программу MAKE, и вы можете сделать это следующим образом:

Чтобы установить использование внешней программы Make в проекте:

1. Установите вашу программу Make (обычно поставляемую с ассемблер/компилятор окружением), как *code generation tool*. Задайте расширение исходника как MAK и установите флажок *Always Build*. Для типичной программы Make задайте командную строку:

   ```
   -f%1
   ```

2. Используйте команду *Add/Remove Source Files* для добавления makefile (то есть, MYPROJECT.MAK) к проекту.

3. Также добавьте файлы исходного кода, но для *Code Generation Tool* выберите <NONE>. Каждый раз при построении проекта (built), ISIS запустит внешнюю программу MAKE с файлом проекта makefile в качестве параметра. Затем идёт обращение к программе MAKE, чтобы определиться, какой следует запустить инструмент генерации кода. Хорошая make-программа подразумевает большую гибкость.

Использование редактора исходного кода других производителей

PROTEUS VSM имеет простой текстовый редактор — SRCEDIT, который можно использовать для редактирования файлов исходного кода. SRCEDIT — это модифицированная версия NOTEPAD, которая может открывать множество исходных файлов и может реагировать на запросы DDE, чтобы сохранить модифицированные буферы.

Если у вас есть более совершенный текстовый редактор, как, например, UltraEdit, вы можете проинструктировать ISIS использовать этот редактор вместо штатного. Заметьте, что IDE окружение, возможно, не будет реагировать на DDE команды, и подобное интегрирование может оказаться неподходящим. Однако вы всегда можете запросить производителя добавить эту поддержку — это не так сложно.

Чтобы установить альтернативный редактор исходного кода:

1. Выберите команду *Setup External Text Editor* из раздела *Source*.
2. Щёлкните по кнопке Browse и используйте селектор файлов для указания исполняемого файла вашего текстового редактора.
3. ISIS инструктирует текстовый редактор открывать и сохранять файлы, используя DDE протокол. Обратитесь к документации на текстовый редактор или к поставщику, чтобы разобраться с синтаксисом команд. Если вы не уверены в имени сервиса, попробуйте использовать имя продукта, то есть, ULTRAEDIT.

ИСПОЛЬЗОВАНИЕ IDE ДРУГИХ ПРОИЗВОДИТЕЛЕЙ

Большинство профессиональных компиляторов и ассемблеров имеют собственное интегрированное окружение разработки или IDE. Примеры этого — IAR Embedded Workbench, Keil uVision 2, Microchip MP-LAB и Atmel AVR studio. Если вы разрабатываете ваш код с одним из этих инструментов, вы можете обнаружить, что легче выполнить шаги редактирования и компиляции в этих IDE, а затем переключиться в Proteus VSM, но только тогда, когда вы получили исполняемый образ (то есть, HEX или COD файл), и вы готовы симулировать их.

Proteus VSM поддерживает два способа работы с внешними IDE:

- Использование Proteus в качестве внешнего отладчика — управление отладкой из ISIS, в основном, как если бы при работе с нормальной CAD симуляцией.
- Использование Proteus, как дополнительного симулятора — управление отладкой из отладчика IDE. Proteus работает, как разновидность виртуального сетевого симулятора, контактируя с IDE через TCP/IP. В этом режиме вы запускаете ваш IDE отладчик на одном компьютере, а симулятор Proteus на другом.

Использование Proteus VSM в качестве внешнего отладчика

Для использования Proteus в качестве внешнего отладчика требуется, чтобы формат символьной отладки, произведённый вашим компилятором, поддерживался как один из доступных для загрузки в Proteus. Загрузчик извлекает адреса каждой строки исходника в программу языка высокого уровня, и, где возможно, локализацию программных переменых.

Общие форматы отладчика — это COD (используемый в мире PIC), UBROF — для всех компиляторов IAR и OMF (используемый для 8051). Мы также предоставляем загрузчики для других коммерческих форматов, как файлы списков, производимых Crownhill PICBasic. Загляните на наш сайт в раздел поддержки «3 rd Party Compilers», где есть последняя информация.

При условии, что загрузчик выбранного вами компилятора есть, процедура загрузки программы в симулируемый микропроцессор совсем проста.

Чтобы загрузить программу, произведённую во внешнем IDE:

1. Убедитесь, что программа откомпилирована и скомпонована без ошибок.
2. Отредактируйте свойство PROGRAM модели CPU, чтобы было имя образа выполняемого файла, произведённого компилятором или компоновщиком, то есть, MYPROG.COD.

   Не вводите имя исходного файла — Proteus VSM не симулирует «C» или «ASM» файлы; CPU модели загружают и выполняют двоичные машинные коды.
3. Нажмите кнопку **PLAY** на панели управления анимацией, чтобы начать симуляцию в реальном времени или нажмите кнопку STEP, чтобы открыть инструкцию первого уровня исходника. В последнем случае появится Source Window (окно исходника) и вы можете начать пошаговую отладку вашего кода.

Использование Proteus VSM в качестве виртуального встроенного эмулятора (ICE)

К моменту написания только Keil uVision 2 для 8051поддерживает виртуальный ICE, хотя мы работаем с IAR, Microchip and Atmel, да и другими, чтобы интегрировать с ними Proteus VSM. Это быстро развивающаяся область, и мы настоятельно рекомендуем заглядывать на наш сайт в раздел «3 rd Party Compilers» за последней информацией и документацией. Инструкции по использованию установок uVision2 для работы с Proteus можно найти именно там.

ВСПЛЫВАЮЩИЕ ОКНА

Большинство моделей микропроцессоров, написанных для PROTEUS VSM, будут создавать несколько всплывающих окон, которые могут отображаться и скрываться с помощью раздела *Debug* основного меню. Эти окна есть трёх основных типов:

- *Status Windows* (окна состояния) — модель процессора будет, как правило, использовать одно из них для отображения значений его регистров.
- *Memory Windows* (окна памяти) — обычно есть одно из них для каждой области памяти в архитектуре процессора. Устройства памяти (RAM и ROM) также создают эти окна.
- *Source Code Windows* (окна исходного кода) — по одному из них будет создано для каждого процессора в схеме.

Чтобы отобразить всплывающее окно:

1. Запустите режим отладки, нажав **CTRL-F12**, или, если он уже запущен, щёлкните на кнопку **Pause** панели анимации (Animation Control Panel).
2. Нажмите **ALT-D**, а затем порядковый номер требуемого окна в разделе *Debug* основного меню.

Эти типы окон могут отображаться только тогда, когда симуляция приостановлена и скрываются автоматически, когда она запущена, чтобы дать вам лучший доступ к активным компонентам на схеме. Когда симуляция приостановлена (вручную или с помощью точки останова, breakpoint) окна, которые были показаны, будут вновь открыты.

Все окна отладки имеют контекстное меню — если вы поместите курсор мышки на окно и нажмёте правую клавишу мышки, появится меню, в котором вы можете управлять появлением и форматом данных в этом окне.

Положение и видимость окон отладки сохраняются автоматически в PWI файле с тем же именем, что и текущий проект. Файл PWI также содержит положение любых точек останова, которые были заданы, и содержание окна наблюдения (watch window).

ОТЛАДКА ИСХОДНОГО КОДА ВНУТРИ PROTEUS VSM

Обзор

PROTEUS VSM поддерживает отладку исходного кода через использование загрузчика отладки для поддерживаемых ассемблеров и компиляторов. Текущий набор загрузчиков отладки содержится в системном файле LOADERS.DLL, а количество инструментальных средств, поддерживаемых ISIS, растёт довольно быстро. Последняя информация на это счёт доступна на сайте в разделе «3 rd Party Compilers».

Если вы используете поддерживаемый ассемблер или компилятор, PROTEUS VSM создаёт окно исходного кода для каждого исходного файла в проекте, и эти окна появятся в меню *Debug*.

Окно исходного кода

Окно исходного кода имеет некоторые особенности:

- Синее выделение представляет текущую строку, на которой можно установить точку останова (breakpoint), нажав **F9** (Toggle Breakpoint), и до которой дойдёт программа, если вы нажмёте **CTRL-F10** (Step To).
- Красный указатель показывает текущее положение программного счётчика процессора.
- Красная окружность маркирует линию, на которой была установлена точка останова.

Контекстное меню, вызываемое правой клавишей мышки, поддерживает ряд опций, включающий: *Goto Line, Goto Address, Find Text* и переключение отображения номеров строк, адресов и байт объектного кода.

Единичные шаги

Поддерживается ряд опций для пошагового режима, все они доступны на инструментальной панели окна исходного кода или из раздела *Debug* основного меню.

- *Step Over* — продвижение вперёд по одной строке, пока строка не инструкция вызова подпрограммы, в этом случае вся подпрограмма выполняется.
- *Step Into* — выполнение одной инструкции кода. Если окно исходного кода не активировано, выполняется инструкция одного машинного кода. Обычно это одно и то же, если вы не отлаживаете на языке высокого уровня.
- *Step Out* — выполняется до возврата из подпрограммы.
- *Step To* — выполняется, пока программа не достигнет выделенной строки. Эта опция доступна только при активном окне исходного кода.

Заметьте, что, исключая Step To, пошаговые команды будут работать без активации *окна исходного кода*. Это возможно, хотя и не так легко, использовать для отладки исходного кода, сгенерированного программой, для которой нет поддержки загрузки.

Использование точек останова (Breakpoints)

Точки останова предлагают очень мощное средство выявления проблем в программе или программно-аппаратном взаимодействии в проекте. Обычно, вы устанавливаете точки останова в начале подпрограммы, вызывающей проблемы, начинаете выполнение симуляции, и затем работаете с проектом, пока программа не достигнет точки останова. Здесь симуляция приостанавливается. И теперь вы можете пошагово пройти код программы, проверяя значения регистров, положение в памяти и другие условия в схеме, пока вы движетесь. Обращение к *Show Logic State of Pins effect* также может быть очень поучительно.

Когда *окно исходного кода* активно, точки останова могут устанавливаться или сбрасываться на текущей строке нажатием на клавишу **F9**. Вы можете устанавливать точки останова только на строке, имеющей объектный код.

Если исходный код меняется, PROTEUS VSM попытается переставить точку останова, основываясь на адресах подпрограммы в файле и на совпадении с байтами объектного кода.

Очевидно, если вы меняете код радикально, это может увести «в сторону», но обычно это работает довольно хорошо, и вам нет нужды об этом думать.

Окно переменных (Variables Window)

Большинство загрузчиков, приходящих с Proteus VSM, способны извлечь положение программных переменных, как и адреса номеров строк исходного кода. Когда это возможно, Proteus отображает *окно переменных* вместе с *окном исходного кода*.

Есть некоторые особенности, на которые следует обратить внимание, относящиеся к окну переменных:

- При пошаговой отладке любые переменные, которые меняют значение, подсвечиваются.
- Формат, в котором каждая переменная отображается, может быть настроен, если щёлкнуть правой клавишей мышки по переменной и выбрать альтернативный формат из контекстного меню.
- Хотя окно переменных скрыто, пока программа работает, вы можете drag & drop (взять и перетащить) переменные в *окно наблюдения*, где они будут оставаться видимы.
- В зависимости от того, как компилятор «переиспользует» память, локальные переменные, выходящие за границы переменных, могут отображать неверные значения.

ОКНО НАБЛЮДЕНИЯ (WATCH WINDOW)

В то время, как *окно памяти* и *окно регистров* принадлежат модели процессора, и отображаются только во время паузы симуляции, *окно наблюдения* существует для отображения значений, которые обновляются в реальном времени. Это также подразумевает присваивание имён индивидуальным областям памяти, которые могут легче обнаруживаться, чем при поиске в *окне памяти*.

Для добавления объекта в окно наблюдения:

1. Нажмите **CTRL-F12**, чтобы начать отладку или приостановите симуляцию, если она уже запущена.
2. Отобразите *окно памяти* (memory window), содержащее объект для наблюдения, и окно наблюдения и окно наблюдения, используя нумерованные опции в разделе *Debug* основного меню.
3. Отметьте позицию в памяти или диапазон памяти, используя левую клавишу мышки. Выделенный диапазон появляется в инверсных цветах.
4. Перетащите выделенный объект(ы) из *окна памяти* в *окно наблюдения*.

Вы можете также добавлять объекты в *окно наблюдения*, используя его команду *Add Item* из контекстного меню после щелчка правой клавишей мышки.

Модификация объектов в окне наблюдения

Получив объект или объекты в окне наблюдения, вы теперь можете выбрать объект левой клавишей мышки, а затем:

- Переименовать его, нажав **CTRL-R** или **F2**.
- Изменить размер данных по любой из опций, полученных из контекстного меню после нажатия правой клавиши мышки. Для размеров данных, которые содержат несколько байт (то есть, 16 или 32 битовых слов или строки), второму и последующим байтам присваиваются следующие за объектом адреса. Следовательно, для отображения многобайтных слов или строк вам нужно только перетащить первый байт из *окна памяти*.
- Изменить формат чисел на двоичный, восьмеричный, десятичный или шестнадцатеричный.

ТРИГЕРЫ ТОЧЕК ОСТАНОВА

Обзор

Некоторые компонентные объекты поддерживают то, какой триггер приостанавливает симуляцию, когда достигаются особые условия. Что особенно полезно, когда симуляция комбинируется с пошаговым режимом, поскольку схема может симулироваться нормально до выполнения частных условий, после чего можно продолжить работу в пошаговом режиме, чтобы увидеть непосредственно, что происходит дальше.

Устройства переключения точек останова можно найти в библиотеке REALTIME устройств.

Триггер напряжения точки останова — RTVBREAK

Это устройство доступно с одним и двумя выводами, и переключает точку останова, когда напряжение на его единственном выводе или напряжение между двумя выводами больше заданного значения. Вы можете связать это устройство с произвольно управляемым источником напряжения (arbitrary controlled voltage source, AVCS) для переключения точки останова при сложных, задаваемых формулами условиях, содержащих множество напряжений, токов и т.д.

Когда напряжение превысило напряжение срабатывания, устройство не повторит переключение до тех пор, пока напряжение не опустится ниже порогового и не повысится вновь.

Триггер тока точки останова — RTIBREAK

Это устройство имеет два вывода и переключает точку останова, когда ток, проходящий через них, больше заданного значения.

Когда ток превысил ток срабатывания, устройство не повторит переключение до тех пор, пока ток не опустится ниже порогового и не повысится вновь.

Цифровой триггер точки останова — RTDBREAK

Это устройство доступно с разным количеством выводов, и вы можете его менять, если это нужно. Переключение точки останова происходит, когда двоичное значение на его входах эквивалентно значению, присвоенному компоненту. Например, задание значения RTDBREAK_8

```
0x80
```

приведёт к переключению, когда D7 в высоком состоянии, а D0-D6 в низком.

Когда условия переключения наступают, устройство не повторит переключения, пока напряжение с другим входным значением не будет получено.

СИМУЛЯЦИЯ НА БАЗЕ ГРАФИКОВ

ВВЕДЕНИЕ

Хотя интерактивная симуляция имеет много преимуществ, остаётся ещё ряд ситуаций, в которых полезнее провести симуляцию в графиках и изучить полученные результаты без спешки. В частности, возможно увеличивать отдельные события симуляции и получить возможность более детального измерения. Симуляция, основанная на графиках, также единственный способ проведения анализа, который нельзя осуществить в реальном времени, как, например, малосигнальный анализ (small-signal AC analysis), анализ шумов (Noise Analysis) и измерения при развёртке параметров (swept parameter).

Симуляция, основанная на графиках, не доступна в PROTEUS Lite.

УСТАНОВКА СИМУЛЯЦИИ НА БАЗЕ ГРАФИКОВ

Обзор

Симуляция, основанная на графиках, выполняется в пять основных этапов. Всё это суммировано ниже с детальными пояснениями на каждом этапе в последующих разделах:

- Ввод схемы для симуляции.
- Размещение сигнал-генераторов в точках, требующих стимулов, и пробников в точках, которые требуют проверки.
- Размещение графиков, соответствующих типу анализа, который вы хотите выполнить — то есть, например, частотный график для отображения АЧХ.
- Добавление генераторов и пробников на график, чтобы отобразить данные, которые они генерируют/фиксируют.
- Установка параметров симуляции (таких, как время выполнения) и выполнение симуляции.

Ввод схемы

Ввод схемы, которую вы хотите симулировать, точно такой же, что и в других проектах ISIS; техника выполнения этого детально описана в руководстве к ISIS.

Размещение пробников и генераторов

Второй этап процесса симуляции — это установка сигнал-генераторов в точках, требующих стимулов, и пробников в точках проверки. Установка сигнал-генераторов и пробников совершенно проста, поскольку выполняется подобно тому, как это делается для других компонентов, контактов или точек соединения в ISIS. Всё, что нужно сделать, это выбрать подходящую иконку, указать тип генератора или пробника в *окне селектора* и поместить объект на схему там, где вам нужно. Это можно сделать непосредственно на существующий провод, или поступить, как с другими компонентами — поместить в рабочее поле, а затем соединить с нужной точкой схемы. Определение сигнала, производимого генератором, это предмет редактирования свойств объекта и задания требуемых установок в его диалоге.

На этом этапе вы можете также изолировать часть проекта, так что только некоторые компоненты будут вовлечены в симуляцию. Это выполняется установкой флажков *Isolate Before* (изолировать до) и *Isolate After* (изолировать после) в пробниках и генераторах. Использование пробников и генераторов таким путём не только ускоряет симуляцию, но также означает, что ошибки из-за удалённых проводов, о которых забыли и заменили, не будут вкрадываться в результаты.

Размещение графиков

Третий этап процесса симуляции — это определение, какой тип анализа или типы вы хотите выполнить. Типы анализа включают аналоговый и цифровой анализ переходных процессов, частотный анализ, анализ «качания» параметров и т.д. В ISIS определение типа анализа — это синоним размещения объекта график требуемого типа анализа. И вновь, поскольку график подобен большинству других объектов в ISIS, его размещение сводится к выбору подходящей иконки, выбору требуемого типа графика и размещение его в проекте рядом со схемой. Но не только так можно увидеть несколько типов анализа одновременно, при подготовке можно использовать метод «drag-and-drop, перетаскивания», принятый в ISIS, что даёт дополнительные удобства, и вы можете увидеть (и вывести в виде твёрдой копии) графики рядом со схемой, которая генерирует их.

Добавление кривых на графики

После размещения одного или больше графиков, вы должны задать, какие данные пробников/генераторов вы хотите видеть на графиках. Каждый график отображает несколько кривых. Данные для построения кривой обычно получены от единственного пробника или генератора. Однако ISIS предлагает для кривой, отображающей данные, до четырёх отдельных пробников/генераторов, скомбинированных математически с помощью *Trace Expression* (выражение для кривой). Например, кривая может быть задана для отображения данных, полученных от пробника напряжения и тока (оба контролируют ту же точку), так что эффективно отображается мощность в контрольной точке.

Задание кривых, отображаемых на графике, может быть сделано несколькими путями: вы можете выделить и перетащить пробник на график или можете выделить несколько пробников/генераторов и добавить их все на график за одну операцию, или для кривых, требующих выражений (trace expressions), вы можете использовать диалоговую форму для выбора пробников и задания выражений.

Процесс симуляции

Симуляция, основанная на графиках, это *Demand Driven* (вывод по запросу). Что означает, что акцент сделан на установке генераторов, пробников и графиков в плане определения, что вы хотите измерять, а не на установках симуляции с последующим, какого-либо рода, постпроцессором, который проверял бы результаты. Любые параметры, специфические для для запуска данной симуляции, задаются либо редактированием подходящих свойств самого графика (то есть, время начала и окончания для симуляции) или присвоения дополнительных свойств графику (то есть, для цифровой симуляции вы можете передать симулятору «randomise time delays, случайные временные задержки»).

Итак, что происходит, когда вы начинаете симуляцию? Вкратце, действие проходит через следующие шаги:

- *Netlist generation* (генерация спецификации схемы) — это обычный процесс трассировки соединений от вывода к выводу и создание списка компонентов, а также списка групп соединений выводов или сетей (nets). Вдобавок, любые компоненты в проекте, которые должны симулироваться файловыми моделями, заменяются компонентами, содержащимися в этих файлах.
- *Partitioning* (разбиение) — ISIS просматривает точки, где вы поместили пробники и обратные пути от них к тем, где вы внедрили сигналы. Результатом этого анализа станет создание одной или более частей, которые могут нуждаться в симуляции, а, следовательно, которые должны симулироваться. Когда это произойдет, результаты сохраняются в новом файле этих частей.
- *Results Processing* (результаты обработки) — наконец ISIS берёт файлы частей для построения разных кривых на графике. График после этого обновляется и может быть максимизирован для измерения и т.д.

Если в процессе выполнения любого из этих этапов обнаруживаются ошибки, тогда детали этого записываются в файл журнала симуляции (simulation log). Некоторые ошибки фатальны, а другие требуют только предупреждения. В случае фатальных ошибок, запись журнала отображается для немедленного обозрения. Если обнаруживаются только предупреждения, тогда график обновляется, а за вами остаётся выбор, хотите ли вы видеть запись в журнале (log). Большинство ошибок относятся либо к плохо нарисованной схеме (которая не может, по тем или иным причинам, математически обработаться), либо к пропущенным или некорректно присоединённым файлам моделей.

ОБЪЕКТ ГРАФИК

Обзор

График — это объект, который можно разместить в проекте. Его назначение — управлять частичной симуляцией и отображать результаты этой симуляции. Тип анализа, выполняемый симуляцией, определяется типом размещённого графика. Часть проекта, которая симулируется, и данные, которые отображены на графике, определяются такими объектами, как пробники и генераторы, которые были добавлены на график.

Текущий график

Все специфические команды, относящиеся к графику, находятся в разделе меню *Graph*. В нижней части этого меню также содержится список всех графиков дизайна, с текущим графиком, отмеченным маленьким маркером слева от имени. Текущий график — это последний график, который симулировался или был выполнен командой.

Любые команды из меню Graph могут быть выполнены для заданного графика, если указать на него мышкой и использовать горячие клавиши (показанные в меню справа от команды). Если указатель мышки не находится на графике, или если вы выбрали команду непосредственно из меню, команда будет выполнена для текущего графика.

Размещение графика

Чтобы разместить график:

1. Получите список типов графиков, выбрав иконку *Graph Mode*. Список типов графиков отображён в *окне селектора*.
2. Выберите тип графика, который вы хотите разместить из *окна селектора объектов*.
3. Поместите указатель мышки в *окне редактора* в точке, где вы хотели бы видеть верхний левый угол графика. Нажмите левую клавишу мышки и «растащите» прямоугольник такого размера, какой вам нужен для графика, а затем отпустите клавишу мышки.

Редактирование графиков

Все графики могут быть перемещены, масштабированы или отредактированы, чтобы изменить их свойства, используя стандартную технику редактирования ISIS.

Свойства графика можно изменить через диалог *Edit...*, вызываемый, как любой объект в ISIS, либо первоначальным выделением с последующим щелчком левой клавишей мышки по графику, либо указав на него мышкой и нажав **CTRL+E** на клавиатуре.

Добавление кривых к графику

Каждый график отображает одну или более кривых. Каждая кривая обычно отражает данные, ассоциированные с одним генератором или пробником. Однако для аналоговых или смешанных типов графиков есть возможность отобразить на отдельном графике данные между одним и четырьмя пробниками, скомбинированными математической формулой, которую мы назвали *trace expression* (выражение кривой).

Каждая кривая имеет этикетку, которая отображается возле оси y, к которой относится кривая. Некоторые типы графиков имеют только одну ось y, и нет опции, чтобы соединить кривую с отдельной осью y. По умолчанию, имя новой кривой то же, что и имя пробника (или первого пробника в выражении кривой) — это можно изменить, отредактировав кривую.

Кривые могут быть определены тремя путями:

- Перетаскиванием отдельных генераторов или пробников на соответствующий график.
- Выделением пробников и использованием инструмента *Quick Add* (быстрое добавление) команды *Add Trace* (добавить кривую).
- Использованием диалоговой формы команды *Add Trace*.

Первые два метода легче в использовании, но ограничивают вас в добавлении новых кривых для каждого генератора или пробника, заданных на графике. Третий метод кажется более сложным, но даёт больше возможностей в контроле над типом кривой, добавленной на график. На практике кривые, которые требуют выражений, должны добавляться к графику через диалоговую форму *Add Trace*.

Чтобы «перетащить» пробник или генератор на график:

1. Выделите генератор или пробник, который вы хотите добавить на график.
2. Щёлкните и удержите левую клавишу мышки на генераторе или пробнике и перетащите пробник на график, где отпустите клавишу мышки.

Новые кривые создаются и добавляются на график; новая кривая отображает данные, ассоциированные с индивидуальным пробником/генератором. Заметьте, что любая существующая кривая может уменьшиться, чтобы график приспособился к новой кривой.

Для графиков с двумя осями y, добавление пробника или генератора на левую или правую половину графика связывает новую кривую с соответствующей осью. Более того, для смешанных, аналоговых и цифровых, типов графиков переходных процессов добавление генератора или пробника к уже существующим цифровым или аналоговым кривым создаёт новую кривую соответствующего типа (первые пробник или генератор, добавленные на график смешанного типа, всегда создают аналоговую кривую).

Чтобы быстро добавить несколько генераторов или пробников на график:

1. Убедитесь, что график, на который вы хотите перенести генераторы или пробники, это текущий график. Текущий график — это график, заголовок которого показан выбранным в меню *Graph*.
2. Выделите каждый генератор или пробник, который вы хотите добавить на график.
3. Выберите команду *Add Trace* в разделе *Graph* основного меню. В результате для выделенных пробников и генераторов команда вначале отобразит приглашение «Quick add tagged probes?».
4. Выберите кнопку **Yes**, чтобы добавить выделенные генераторы и пробники к текущему графику.

Новая кривая создаётся для каждого выделенного генератора или пробника и добавляется к графику в алфавитном порядке; каждая новая кривая отображает данные, ассоциированные со своим генератором или пробником. Кривые всегда связываются с левой осью y для тех типов графиков, которые поддерживают две оси.

Диалоговая форма команды Add Trace

Если вы не выполняли *Quick Add*, команда *Add Trace* отобразит диалоговую форму Add Trace (каждый тип графика имеет некоторые отличия). Эта форма позволяет вам выбрать имя новой кривой, её тип (аналоговая, цифровая и т.д.), ось y (левая или правая), к которой она будет добавлена, до четырёх генераторов или пробников, чьи данные она использует, и выражение, которое комбинирует данные выделенных генераторов или пробников.

Чтобы добавить кривую к графику, используя команду Add Trace:

1. Выполните команду *Add Trace* из раздела *Graph* основного меню напротив графика, к которому вы хотите добавить новую кривую.

   Если есть выделенные пробники или генераторы, последует приглашение «Quick add tagged probes?». Откажитесь, используя кнопку **No**.

2. Выберите тип кривой, которую вы хотите добавить на график, выбрав подходящий с помощью кнопки **Trace Type**. Только те типы кривых, которые допустимы для графика, будут присутствовать в списке.
3. Выберите ось y, к которой новая кривая будет относиться. Только те оси, которые допустимы для графика и выбранного типа кривой, будут присутствовать в списке.
4. Выберите одну из кнопок **Selected Probes** от P1 до P4, и затем выберите пробник или генератор из списка в окне *Probes*, чтобы связать его с выбранной кривой. Имя выбранного генератора или пробника появится рядом с кнопкой **Selected Probes** и имя *Selected Probe* (от P1 до P4) появится в поле *Expression*, если его ещё нет.

   Если выражения кривой не допустимо, будет разрешена только P1 кривая — и новая кривая будет отражать данные, ассоциированные с выбранным P1 пробником.
5. Повторите шаги от 3 до 4, пока вы выберете все генераторы и пробники, которые вам требуются для кривой.
6. Введите выражения кривой в поле *Expression*. В выражении выделенные пробники будут представлены именами P1, P2, P3 и P4, которые соответствуют выбранным пробникам рядом с кнопками **P1**, **P2**, **P3** и **P4**.

   Если выбранный тип кривой не поддерживает выражения, содержимое поля *Expression* будет игнорироваться.
7. Выберите кнопку **ОК**, чтобы добавить новую кривую на график.

Редактирование кривых графика

Индивидуальные кривые на графике можно редактировать, чтобы изменить их имена или выражения, определяющие их.

Чтобы выделить, отредактировать и снять выделение с кривой:

1. Убедитесь, что график, отображающий кривую, которую следует редактировать, не выделен.
2. Выделите кривую, щёлкнув по её имени. Выделенная кривая отображается подсвеченным именем.
3. После щелчка по имени кривой, появляется её диалоговое окно Edit Graph Trace.
4. Отредактируйте имя кривой или выражение, как это требуется. Для типа кривой, не поддерживающей выражения, любые изменения в поле *Expression* будут игнорироваться.
5. Выберите ОК, чтобы принять изменения.
6. Чтобы снять выделение с кривой, щёлкните по графику, но не по имени кривой.

Изменение последовательности и/или цвета кривой на графике

Последовательность кривых на графике может быть настроена перетаскиванием этикеток с помощью левой клавиши мышки. Кривая может быть выделена или наоборот, как вы хотите.

- Для цифровых кривых цель их перемещения, просто, в получении особой последовательности расположения.
- Для аналоговых графиков, вы можете перетаскивать кривые с левой на правую ось и назад.

  Актуальные цвета, назначенные по позиции в последовательности, могут быть реконфигурированы после максимизации графика с последующим выполнением команды *Set Graph Colours* из раздела *Template*.

ПРОЦЕСС СИМУЛЯЦИИ

Симуляция, выполняемая по требованию

Когда мы разрабатывали Proteus, одной из наших главных целей было сделать использование симуляции, как можно более интуитивным, более удобным, чем это было прежде. Множество проблем с прежними пакетами симуляции произрастало из того факта, что ряд фрагментов было написано раньше, а другие аспекты, такие как графическое отображение результатов, были добавлены позже, скорее, как запоздалые раздумья. Результатом этой тенденции стал фрагментарный вид работы, при которой вы запускаете одну программу, чтобы нарисовать схему, другую для симуляции схемы, и третью для отображения результатов.

Proteus очень отличается в этом, нарисовав схему, вы начинаете с ответа на вопрос, что вы хотите видеть, размещая и настраивая график. Этот график затем сохраняется до тех пор, пока вы не удалите его, и каждый раз, когда вы хотите видеть эффект от изменения схемы, вы только обновляете график, указав на него мышкой и нажав пробел. Мы назвали это Demand Driven Simulation (симуляция по запросу), поскольку ISIS должен работать с графиком, который действительно требует симуляции, и чтобы вам не приходилось делать этого с помощью ввода текстов. Более того, вы можете поместить несколько графиков, которые отображают разные эксперименты, и каждый «запоминает» разные установки для симуляторов.

Наиболее важное преимущество имеет место от того, что данный график определяет ряд интересующих точек в проекте через пробники, которые связаны с кривыми. ISIS затем может использовать эту информацию для выявления, какие части проекта действительно нуждаются в симуляции, вместо того, чтобы заставлять вас вручную «прошерстить» весь проект. В результате, весь проект, готовый к разводке печатной платы, может симулироваться без интенсивного редактирования, когда вам и нужно-то было симулировать только отдельные его части.

Ещё одно преимущество, нет риска, если вы забыли отменить модификации, сделанные для целей симуляции, поскольку нет такой отмены; гаджеты пробника и генератора, которые вы размещали, игнорируются при генерации netlists для разводки печатной платы (PCB layout).

Выполнение симуляций

Когда график размещён с пробниками и генераторами, связанными с ним, вы можете инициировать симуляцию, указав график и нажав пробел. ISIS определит, какие части проекта следует симулировать, с тем чтобы обновить график, запустит симуляцию и отобразит новые данные.

В процессе симуляции формируется отчёт (simulation log) о симуляции. Обычно эта запись не представляет особого интереса. Однако, если при симуляции обнаруживается ошибка, или вы запрашивали очёт о netlist при симуляции (см. «Редактирование графиков»), или результаты анализа получены в виде данных, которые не могут быть отображены графически (то есть, аналоговый анализ шумов), тогда вам нужно просмотреть запись в журнале (log) по окончании симуляции.

Чтобы обновить график и увидеть отчёт о симуляции:

1. Вначале убедитесь, что график, который вы хотите обновить, это текущий график, а затем выберите команду *Simulate* из раздела *Graph* или установите указатель мышки на график, нужный вам, и нажмите пробел.
2. По окончании симуляции, если обнаруживается ошибка, вы получите приглашение: «Load partial results?». Если вы ответите **YES**, загрузятся данные симуляции до момента возникновения ошибки и отобразятся на графике. Если вы ответите **NO**, во всплывающем окне просмотра текста отобразится отчёт о симуляции.

Если ошибок не обнаруживается или если вы выбрали **YES**, вы всё ещё можете посмотреть отчёт о симуляции либо выбрав команду *View Log* из раздела Graph основного меню, либо используя горячие клавиши **CTRL+V**.

Что происходит, когда вы нажимаете пробел

Когда вы обновляете график, либо используя команду *Simulate* из разделе *Graph*, либо с помощью горячих клавиш команды (пробел) — будет проделана большая работа по анализу и во время, и до, собственно, аналогового или цифрового анализа. Сейчас мы вкратце покажем разные шаги этого процесса:

- *Netlist compilation* (компиляция спецификации) — это обычный процесс трассировки связей от вывода к выводу и составление списка компонентов и списка групп соединения выводов или сетей. Результирующий netlist на этом этапе хранится в памяти.
- *Netlist linking* (компоновка спецификации) — некоторые компоненты моделируются с использованием подсхем (sub-netlists) или файлов моделей, которые обычно хранятся в директории, заданной в поле *Module Path* диалоговой формы команды *Set Paths*. Множество моделей приходят с Proteus, и вы можете, конечно, создавать свои собственные. Вы можете представлять себе модели, как подлисты иерархического дизайна, в котором родительский объект — это компонент, который нужно моделировать.

  Каждый раз, когда компонент моделируется таким образом, оригинальный элемент удаляется из netlist и замещается содержимым файла модели со входами и выходами модели, соединёнными в том месте, где соединялись выводы оригинального компонента.

  В конце этого процесса каждый компонент, остающийся в netlist, будет иметь свойство PRIMITIVE, которое означает, что он может непосредственно симулироваться PROSPICE.
- *Partitioning* (разбиение на части) — ISIS просматривает точки, где вы поместили пробники и обратные связи отсюда ко входным сигналам. Сигналы считаются

подключёнными к любому из: шина питания, генератор с установленным флагом *Isolate Before* или выходом записи. Этот анализ результируется в создании одной или более частей, которые нужно симулировать.

Дальнейший анализ затем работает в порядке, в котором они должны симулироваться. Например, если часть A имеет выход, который соединён со входом части B, тогда ясно, что A должна симулироваться первой. Если же выясняется, что B имеет выход, приходящий к A, тогда все теряется — и это уже к вам, используйте запись объектов таким образом, чтобы циклические зависимости не появлялись.

Всё же больше анализов устанавливается там, где есть существующие результаты, в директории, заданной через поле *Results Path* (командой *Set Paths* меню *System*), которые относятся к идентичным запускам симуляции для любого текущего набора частей. Если они есть, и если участвующие части не стимулированы теми, что сами будут заново симулироваться, тогда ISIS не спешит пересимулировать части, но использует вместо этого существующие результаты. Из чего следует, если вы работаете на другом конце проекта, нет нужды поддерживать пересимуляцию сначала.

Вы можете убедиться, что эта часть системы очень умна!

- *Simulator Invocation* (вызов симулятора) — ISIS теперь вызывает PROSPICE для каждой части в порядке выполнения действительной симуляции.

  Каждый вызов симулятора приводит к созданию нового частичного файла, а информация о прогрессе симуляции также добавляется к отчёту о симуляции, который поддерживается во время всего процесса симуляции.

- *Results Processing* (завершение процесса) — окончательно, ISIS проходит по частичным файлам, чтобы выстроить разные кривые на графике. График после этого обновляется и может быть максимизирован для проведения измерений и т.д.

Если в процессе выполнения любого из этих этапов обнаруживаются ошибки, тогда детали этого записываются в отчёт (simulation log). Некоторые ошибки фатальны, а другие вызывают только предупреждения. В случае фатальных ошибок отчёт о симуляции отображается для немедленного просмотра. Если обнаруживается только необходимость в предупреждениях, тогда график обновляется и у вас остаётся возможность посмотреть отчёт, если вы хотите этого. Большая часть ошибок относится либо к плохо сделанному чертежу (которые не могут, по тем или иным причинам, быть разрешены математически), либо из-за пропущенных или плохо связанных файлов моделей.

ТИПЫ АНАЛИЗА

ВВЕДЕНИЕ

Есть тринадцать типов графиков; каждый отображает результаты разных типов анализа схемы, поддержанных PROSPICE.

Analogue (аналоговый) Выводит напряжения и/или токи, как функцию времени, и более всего похоже на работу осциллографа. Дополнительно, могут выводиться выражения, выполняемые несколькими пробниками; например, ток может умножаться на напряжение, давая соответствующую мощность. Этот режим анализа часто называют анализом переходных процессов (Transient Analysis).

Digital (цифровой) Выводит логические уровни, как функцию времени, и более всего это похоже на логический анализатор. Кривые могут представлять единственные биты данных или бинарное значение шины.

Цифровые графики вычисляются с использованием Event Driven Simulation (событийно осуществляемая симуляция).

Mixed Mode (смешанный режим) Комбинирует оба, аналоговый и цифровой, сигналы на одном и том же графике.

Frequency (частотный) Выводит малосигнальное напряжение или ток, как функцию частоты. Известен также, как Bode plot (плоттер Боде), и может отображать и амплитуду, и фазу. Дополнительно, используя Trace Expressions, вы можете выводить входной и выходной импеданс.

Заметьте, что выведенные значения зависят от заданного входного генератора.

DC Sweep (развёртка на постоянном токе) Устоявшееся состояние рабочей точки по напряжению или току, как функция переменной развёртки. Как и в аналоговом анализе, могут выводиться выражения, комбинирующие несколько пробников.

AC Sweep (развёртка на переменном токе) Создаёт семейство частотных характеристик с одним откликом для каждого значения переменной развёртки.

Transfer (передаточный) Выводит характеристические кривые или семейство кривых, «качая» значение одного или двух входных генераторов и выводя установившиеся состояния напряжения или тока. Значение переменной первого генератора отображается по оси x; отдельная кривая производится для каждого значения второй переменной.

Noise (шумы) Входное или выходное напряжение шумов, как функция частоты. Также производит список отдельных шумовых составляющих.

Distortion (искажения) Выводит 2 и 3 гармоники искажений, как функцию частоты. Может также использоваться для вывода интермодуляционных искажений.

Fourier (Фурье)	Показывает гармонический спектр переходных процессов. Похоже на использование анализатора спектра вместо осциллографа.
Audio (аудио)	Выполняет анализ переходных процессов, а затем проигрывает результат через вашу звуковую карту. Может также генерировать Windows WAV файлы с выхода вашей схемы.
Interactive (интерактивный)	Выполняет интерактивную симуляцию и отображает результаты на графике. Этот тип анализа позволяет вам комбинировать преимущество интерактивного и графического видов симуляции.
Conformance (согласования)	Выполняет цифровую симуляцию, а затем сравнивает результаты с сохранёнными результатами предыдущего запуска. Это особенно важно при создании программных тестов для приложений на базе микроконтроллеров.

Дополнительно, все типы анализа начинаются с вычисления рабочей точки, то есть, начальных значений для всех узлов напряжения, ветвей тока и состояния переменных в момент времени равный 0. Информация, относящаяся к рабочей точке, доступна в ISIS через интерфейс *point and shoot*.

АНАЛОГОВЫЙ АНАЛИЗ ПЕРЕХОДНЫХ ПРОЦЕССОВ

Обзор

Этот тип графика представляет то, что вы могли бы увидеть на экране осциллографа. Ось x показывает продвижение по времени, а ось y отображает напряжение или ток. Мы часто ссылаемся на этот тип анализа, как на Transient Analysis, поскольку он имеет место во временной зоне.

Transient analysis, возможно, наиболее общеупотребительный из всех форм анализа. Все, что вы можете измерить с реальным осциллографом, можно измерить на графике *analogue*. Он может использоваться для проверки, что схема работает ожидаемым образом, для быстрого измерения усиления, для визуальной оценки искажений, для измерения тока от источника или проходящего через отдельный компонент и т.д.

Метод вычисления

Технически говоря, этот тип симуляции можно отнести к нелинейному узловому анализу, а это форма вычисления, используемая всеми SPICE симуляторами. В каждый момент времени каждый компонент в схеме представляется, как комбинация источников тока и/или резисторов. Это представление затем описывается системой совместных уравнений, использующих закон Ома и законы Кирхгофа, а уравнения решаются Гауссовым методом исключения. Для каждого момента времени, для которого решаются уравнения, значения источников тока и резисторов задаются законами, встроенными в модели компонентов и процесс повторяется, пока не будет получен стабильный набор результирующих значений.

Для симуляции, связанной с продвижением по времени, есть два отдельных этапа. На первом задача состоит в вычислении рабочих точек схемы — это напряжения по всей цепи в начальный момент симуляции. При этом учитывается эффект изменения времени для схемы, для чего идёт пересчёт напряжения на каждом шаге (итерируемом до схождения, как описано выше). Шаг по времени критичен для стабильности вычислений, и поэтому PROSPICE погоняет его автоматически, в рамках, определённых пользователем. Схема, которая меняется быстро, например, ряд быстро переключающихся ключей, нуждается в меньших временных шагах, а, значит, и большем количестве попыток, чем схема, которая меняется более плавно. Использование управления временными шагами и итеративный расчёт — все это добавляет вычислений, которые могут сделать transient analysis сравнительно медленным.

Поскольку алгоритм использует итерации, всегда существует вероятность, что решение не сойдётся. Чаще всего это происходит в начальной точке времени, означая, что симулятор не может установить стабильный или уникальный набор значений для рабочей точки. Временами, однако, это может обнаружиться в дальнейшем, когда обычное поведение схемы становится совершенно непредсказуемо. PROSPICE снабжён рядом техник, помогающих решить подобные проблемы, но не способных их полностью убрать. Скажем, использование в качестве основы Berkeley SPICE3F5 настолько хорошо, насколько вы можете этим воспользоваться.

Использование analogue графиков

Графики analogue могут иметь либо только левую ось y, либо и левую, и правую. Пробники могут размещаться на отдельных осях двумя способами:

- Выделив и перетащив пробник к подходящей стороне графика.
- Используя диалоговую форму *Add Transient Trace*.

См. раздел «Добавление кривых на график», где больше информации о том, как добавить кривые на существующий график.

Очень часто оказывается удобно использовать левую и правую оси y для отдельных кривых с разными единицами измерения. Например, если отображаются оба пробника: и напряжения, и тока, — тогда левая сторона может использоваться для отображения напряжения, а правая для тока. Можно размещать цифровые пробники на аналоговом графике, но *Mixed* график, как правило, удобнее.

Для выполнения анализа переходного процесса (transient analysis) аналоговой цепи:

1. Добавьте в схему генераторы, как необходимо для получения входных сигналов.

См. разделы «Размещение генераторов» и «Генераторы и пробники», где подробнее описаны разные типы аналоговых генераторов.

2. Поместите пробники на схему в тех точках, что вас интересуют. Это могут быть и точки внутри схемы, и очевидные выходы схемы.
3. Поместите *Analogue* график.
4. Добавьте пробники (и генераторы, если нужно) на график.
5. Отредактируйте график (укажите на него и нажмите **CTRL+E**, задайте требуемое

время остановки симуляции, этикетки на график и управляемые свойства, если это нужно.

6. Запустите симуляцию либо выбором команды *Simulate* раздела *Graph*, либо нажатием пробела.

Определение Analogue Trace Expressions (выражений)

Обычно вам достаточно добавить пробники напряжения и тока на график, чтобы увидеть процессы в схеме. Однако в аналоговом анализе переходных процессов можно создавать кривые, которые используют математические выражения, основанные на показаниях пробников. Например, положим, что схема имеет и пробник напряжения, и пробник тока, размещённые в схеме. Умножая значения этих пробников, мы можем нарисовать график выходной мощности в этой точке.

Чтобы отрисовать график выходной мощности:

1. Добавьте пробники тока и напряжения на выход схемы.
2. Вызовите диалоговую форму *Add Transient Trace* (нажмите **CTRL+A**) для графика.
3. Присвойте пробник напряжения P1.
4. Присвойте пробник тока P2.
5. Измените выражение (expression) на P1*P2.
6. Щёлкните по кнопке **ОК**.
7. Нажмите пробел, чтобы выполнить симуляцию.

См. раздел «Диалоговая форма команды *Add Trace*», где детально описано использование формы *Add Transient Trace*.

ЦИФРОВОЙ АНАЛИЗ ПЕРЕХОДНЫХ ПРОЦЕССОВ

Обзор

Цифровой график отображает то, что вы могли бы увидеть на экране логического анализатора. Ось x отображает время, а ось y показывает вертикальный стек сигналов. Это могут быть либо одиночный биты данных, либо двоичное представление вывода на шину.

Метод вычисления

Цифровой анализ переходных процессов использует технику, известную как событийно-ориентированная симуляция (Event Driven Simulation). Она отличается от аналоговой тем, что процесс осуществляется только тогда, когда какой-нибудь элемент схемы меняет своё состояние. Вдобавок, только дискретные логические уровни принимают в нем участие, что позволяет представить функционирование компонентов на более высоком уровне. Например, мы можем представить себе счётчик в терминах регистра значений, которые увеличиваются на единицу при каждом тактовом импульсе, а не в терминах нескольких сотен транзисторов. Это делает событийно-ориентированную симуляцию на несколько порядков быстрее, чем аналоговую симуляцию этой же схемы.

Этап загрузки (Boot Pass)

Цель этапа загрузки — определить начальные состояния всех цепей схемы, в первую очередь для выполнения симуляции. Boot pass выполняется следующим образом:

- Все входные выводы, присоединённые к цепям VCC и/или VDD, считаются имеющими высокий уровень.
- Все входные выводы, присоединённые к цепям GND и/или VSS, считаются имеющими низкий уровень.
- Все входные выводы, присоединённые к цепям, к которым присоединён генератор, считаются имеющими то же состояние, что и в свойстве *INIT* генератора.
- Все остальные выводы считаются имеющими плавающее начальное значение.
- Все модели запрашиваются (не в установленном порядке) для вычисления их входных и установки их выходных выводов соответственно. Когда каждый выходной вывод задан, состояние цепи, к которой вывод подключён, перерассчитывается.
- Все изменения состояния цепей, моделей, присоединённых к ним, запрашиваются для перерасчёта их выходов. Этот процесс продолжается, пока не находится устойчивое состояние.

Циклический процесс событий

Следом за этапом загрузки DSIM начинает собственно процесс симуляции. Симуляция выполняется в цикле, который проходит повторно через следующие два шага:

- Все события, меняющие состояние на текущий момент, прочитываются по очереди и применяются к существенным цепям. Этот процесс завершается новым набором состояний цепей.
- Там, где цепь меняет состояние, все модели, входы которых к ней присоединены, симулируются вновь. Там, где их выходы меняются, это создаёт новые события, которые помещаются в последовательность событий.

Конечно, разные модели создадут события, которые могут закончиться неудачно для обработки в разное время. Поэтому ядро DSIM должно упорядочить все новые события, созданные в конце каждого прохода цикла.

Также ценно то, что наша схема довольно удачно поддерживает модели, которые имеют нулевые временные задержки. В этом контексте события, генерируемые с одинаковым временным кодом, происходят в пакете (один пакет эквивалентен одному проходу по циклу), согласно тому, как они сгенерированы.

Условия прерывания

Симуляция останавливается, когда обнаруживается одно из следующего:

- Достигается заданное время остановки.
- Запрос о событиях становится пуст. Это означает, что схема достигла устойчиво стабильного состояния.
- Логический парадокс с нулевой задержкой времени обнаруживается так, что текущее время перестаёт меняться, несмотря на повторение циклов прохода при обработке событий в цикле.
- Системная ошибка, как выбег запросов событий за пределы области памяти. Это не обнаруживается при нормальной работе, если не возникает нечто нестабильное в вашем проекте, возможно, склонность к высокочастотной (то есть, 100 МГц) осцилляции где-нибудь.

Использование цифровых графиков

Чтобы выполнить анализ переходных процессов для цифровой схемы:

1. Добавьте генераторы, как нужно для работы схемы, чтобы запитать входы.
2. Поместите пробники на схеме в интересующих вас точках. Это могут быть как точки внутри схемы, так и очевидные выходы.
3. Поместите *Digital* график.
4. Добавьте пробники (и генераторы, если нужно) на график.

Можно использовать только *Digital* генераторы для генерации цифровых сигналов — заметьте, что обычно *Pulse* (импульсный) генератор рассматривается, как источник аналоговых импульсов.

Только цифровые пробники должны использоваться в цифровых цепях — использование токовых пробников будет заставлять PROSPICE выполнять симуляцию смешанного режима.

5. Отредактируйте график и задайте требуемое время остановки (Stop Time) симуляции, а также этикетки графика и управляемые свойства, если это требуется.
6. Выполните команду *Simulate* из раздела *Graph* основного меню напротив графика или нажмите пробел.

См. раздел «Процесс симуляции» с общей информацией о том, как симулировать графики.

См. разделы «Размещение генераторов» и «Размещение пробников» по размещению генераторов и пробников.

См. раздел «Добавление кривых на график», где описано, как добавить пробники на графики.

Как отображаются цифровые данные

Нормальные кривые

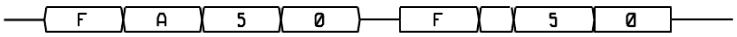
Все выходные значения от симулятора DSIM получаются в терминах шести цифровых состояний:

Тип состояния	**Ключевое слово**	**Появление на графике**
Строго высокое	**SHI**	Высокий уровень, голубой
Слабо высокое	**WHI**	Высокий уровень, синий
Плавающее	**FLT**	Средний уровень, белый
Спорное	**CON**	Средний уровень, жёлтый
Слабое низкое	**WLO**	Низкий уровень, синий
Строгое низкое	**SLO**	Низкий уровень, голубой

Вы можете, таким образом, определить состояние кривой в отдельное время, либо посмотрев на цвет и позицию линии, либо, максимизировав график, установить курсор в выбранном месте и прочитав состояние в строке состояния. Логические уровни в позиции курсора также отображаются справа от этикетки кривой.

Кривые шины

Кривые шины (результат от размещения пробника напряжения на проводе шины) отображают шестнадцатеричные значения битов шины между пересечениями линий:



Когда один или более битов шины не имеет ни высокого, ни низкого уровня, линии шины рисуются на среднем уровне. Также, если переход уровней слишком узкий для отображения шестнадцатеричного значения, тогда оно опускается. Значение ещё может быть прочитано, если на него поместить курсор.

АНАЛИЗ ПЕРЕХОДНОГО ПРОЦЕССА СМЕШАННОГО РЕЖИМА

Обзор

График смешанного режима (mixed mode) позволяет и цифровым кривым, и аналоговым отображаться над той же осью x, на которой представлено время. Анализ смешанного режима — фактически, применим тогда, когда в схеме есть и аналоговые, и цифровые модели, и график смешанного режима единственный тип, способный отображать оба вида сигналов вместе.

Метод вычисления

Смешанный режим анализа переходных процессов комбинирует вместе нелинейный узловой анализ SPICE3F5 и событийно-ориентированную симуляцию DSIM. Детальная реализация этого весьма сложна, но основной алгоритм может быть суммирован в следующем:

- Перед симуляцией каждая цепь анализируется на предмет, соединён ли с ней аналоговый, цифровой или оба типа моделей. Там, где аналоговые источники приходят на цифровые входы, PROSPICE вставляет объекты АЦП интерфейса, а там, где цифровые выходы управляют аналоговыми компонентами, вставляет объекты ЦАП интерфейса.

  Там, где несколько цифровых входов запитываются от одной цепи, создаются множественные АЦП объекты, так что переключения разных логических уровней и загружаемых характеристик может быть смоделировано для каждого входа. Аналогично, в случае, когда несколько цифровых выходов соединяются вместе, создаются множественные объекты ЦАП.

- Рабочие точки формируются так, как описано ниже.

- Симуляция затем протекает, как для обычного аналогового анализа, исключая то, что АЦП генерирует цифровые события, когда их входное напряжение пересекает порог переключения. Когда это происходит, аналоговая симуляция приостанавливается, а начинается цифровая симуляция для обработки эффекта от этих новых событий.

- Когда цифровая симуляция сказывается на изменении состояния для объектов интерфейса ЦАП, аналоговая симуляция возобновляется для тщательной симуляции около времени переключения. Фактически, выходы ЦАП моделируют время перехода (восходящее или спадающее) и несколько временных точек будет симулироваться за этот период.

Поиск рабочей точки

Обработка рабочей точки — это особый «фокус» для симуляции смешанного режима, поскольку состояние аналоговой цепи сказывается на состоянии цифровых компонентов и наоборот.

В сущности, PROSPICE делает следующее:

- Значения начальных условий (Initial condition, IC) присваиваются обоим, аналоговым и цифровым, цепям; остальные цепи считаются стартующими с нулевого напряжения.
- Узловые матрицы затем конструируются и рассчитываются, как при обычной SPICE симуляции.
- Для каждой итерации цифровая схема заново обрабатывается, чтобы изменить входы АЦП. Когда эти изменения распространяются через несколько цифровых моделей, цифровой схеме позволяется произвести итерацию до стабильного состояния.
- Объекты ЦАП заново присваивают свои выходы согласно любым изменениям в состоянии цифровой схемы. Если цифровая схема не «успокоилась», это игнорируется, поскольку это может быть кратковременным всплеском.
- Проходы через цикл продолжаются до обнаружения полной устойчивости состояния или по достижении ограничений на итерации.

Использование смешанных графиков

Смешанные (Mixed) графики используются таким же образом, что и аналоговые графики, исключая то, что вы можете добавлять на них цифровые кривые. Чтобы добавить первую цифровую кривую на смешанный график, вы должны использовать диалог *Add Trace*. Затем перетащите пробник на аналоговую часть графика, создавая новую аналоговую кривую, тогда как перетаскивание на цифровую часть создаст новую цифровую кривую.

ЧАСТОТНЫЙ АНАЛИЗ

Обзор

При частотном анализе (Frequency Analysis) вы можете видеть, как схема ведёт себя на разных частотах. Только одна частота рассматривается в один момент времени, так что вывод не похож на тот, что даёт спектральный анализ, где все частоты рассматриваются вместе. Скорее, это похоже на то, что вы подключаете генератор ко входу и следите за выходом, к которому подключён вольтметр переменного напряжения. На графике кроме амплитуды может отображаться и фаза сигнала пробника.

Частотный анализ производит частотную характеристику или диаграмму Боде (Bode plots). Это полезно, когда вы проверяете, ведёт ли себя фильтр так, как ожидается, или проверяете усилитель, работает ли он корректно в требуемой полосе частот.

Частотный график (Frequency) может также быть использован для отображения входного и выходного импеданса как функции частоты на малом сигнале.

Метод вычисления

При выполнении частотного анализа вначале находится рабочая точка схемы, затем все активные компоненты заменяются линейными моделями. Внутренние ёмкости активных устройств вычисляются в рабочей точке, принимается, что рабочее напряжение в схеме отклоняется не на много. Все генераторы, отдельно опорный генератор (см. ниже), замещаются их внутренними сопротивлениями. В этом случае линии питания эффективно соединены вместе, что нормально для частотных вычислений. Затем анализ выполняется с комплексными числами в линейном виде. Частота постепенно увеличивается от начальной до конечной с одинаковым приращением. Линейная природа этого анализа делает его более быстрым, чем анализ переходных процессов, не смотря на использование комплексных чисел.

Очень важно помнить, что частотный анализ применим к линейным цепям. Это означает, что чистый синусоидальный сигнал на входе производит чистый синусоидальных сигнал на выходе на всех частотах. Конечно, нет реальных активных цепей, которые были бы чисто линейными, но многие очень к ним близки, позволяя использовать эту форму анализа. Есть также схемы, которые в принципе нелинейны, как, например, линейные повторители с триггерами Шмитта на входе. Для нелинейных цепей термин «частотная характеристика» не имеет реального математического значения, поэтому частотная симуляция не даст значимых результатов. Если вас интересует поведение таких схем в частотной области, тогда вам больше подойдёт Фурье анализ.

Использование частотных графиков

Чтобы рассчитать амплитуду синусоидального сигнала на выходе, мы должны добавить на вход опорное синусоидальное напряжение. PROSPICE сделает это автоматически, но ему нужно знать, где вход схемы. Чтобы это сделать, вы должны задать для каждого частотного графика *Reference Generator*, чтобы дать знать симулятору, где расположен опорный сигнал. Есть три способа сделать это:

- Использовать поле *Reference* диалоговой формы *Edit Frequency Graph* для выбора входного генератора. Этот объект может быть обычным, одновыводным генератором или примитивом FREQREF.
- Выделить и перетащить любой аналоговый генератор на частотный график.
- Использовать диалоговую форму *Add Trace* для добавления генератора как REF.

Примитив FREQREF полезен для задания моделей, которые приводят в действие схему, например, модели микрофона. Он будет содержать явно названный генератор, используемый как опорный, тогда как эффект от остального в модели микрофона (как моделирование частотной характеристики) потребует уточнения.

Второй и третий приёмы будут использоваться чаще. Действие по перетаскиванию генератора на частотный график отличается от перетаскивания его на график переходного процесса. Вместо добавления пробника, генератор нужен как опорная точка схемы. Нужен генератор не синусоидальный — DC, Pulse и Pwlin, однако, вполне подойдут. Если вы все-таки хотите добавить генератор как пробник, это все ещё можно сделать, используя диалоговую форму Add Trace.

Амплитуда опорного источника всегда 1 вольт, фаза всегда 0 градусов. Внутреннее

сопротивление опорного генератора будет зависеть от того, что определено для генератора на первом месте. Это используется для расчётов децибел, то есть, 0dB = 1 volt. ISIS ограничивает очень маленькие значения, избегая log(0) ограничением до -200dB.

Частотный график всегда имеет обе, левую и правую, оси у. Левая ось используется для отображения амплитуды сигнала, а правая для отображения фазы. Если вы перетаскиваете пробник на левую сторону графика, будет отображаться амплитуда, а если на правую, фаза. Ось x используется для отображения частоты опорного генератора (reference generator). Для отображения частоты всегда используется логарифмическая шкала. Левая ось может иметь либо dB, либо обычные единицы, а правая всегда проградуирована в градусах.

Чтобы получить частотную характеристику цепи:

1. Разместите пробники на схеме в точках, которые вас интересуют.
2. Поместите график Frequency.
3. Добавьте на него пробники: для отображения амплитуды к левой стороне, для отображения фазы к правой. Вы можете добавлять пробник дважды, чтобы получить и фазу, и амплитуду.
4. Отредактируйте график (укажите на него и нажмите **CTRL+E**), задайте начальную и конечную частоту, и все требуемые *Simulation Control Properties*.
5. Для вызова PROSPICE нажмите пробел.

Файлы примеров ZIN.DSN и ZOUT.DSN показывают, как комбинировать частотный график с выражениями кривой для получения входного и выходного импеданса.

АНАЛИЗ РАЗВЁРТКИ НА ПОСТОЯННОМ ТОКЕ

Обзор

При анализе развёртки на постоянном токе (DC Sweep analysis) вы можете видеть, как изменения цепи скажутся на её работе. Это выполняются с помощью *Property Expression Evaluation* симулятора PROSPICE. График развёртки определяет переменную, которая будет «развёрнута» в несколько шагов в заданном пользователем диапазоне. Переменная развёртки может появиться в свойствах любого элемента схемы, такого как значение сопротивления, усиление транзистора или температура схемы.

Кривая DC Sweep показывает уровень установившегося состояние напряжения (или тока) в точках наблюдения на схеме при увеличении переменной развёртки. Она может использоваться для вывода передаточной характеристики цепи на постоянном токе, если присвоить переменную качания значению генератора, или для вывода эффекта влияния значения компонента на рабочую точку цепи.

Метод вычисления

При анализе развёртки на постоянном токе PROSPICE многократно определяет рабочую точку цепи, увеличивая переменную качания между вычислениями. PROSPICE будет пересчитывать все переменные, используемые между шагами, так что переменная развёртки

может использоваться так часто, как это нужно, а также могут использоваться переменные, основанные на переменной развёртки. Любые параметры инициализации схемы будут приняты во внимание для каждого вычисления рабочей точки.

Использование графика DC Sweep

Как и при аналоговом анализе переходных процессов (см. «Аналоговый анализ переходных процессов»), можно использовать либо левую, либо правую, либо обе оси y. По оси x откладывается переменная развёртки. Выражения кривой, описанные для диалоговой формы команды *Add Trace*, могут использоваться и для *DC sweep*. При выборе количества шагов не забывайте, что время симуляции прямо пропорционально выбранному количеству шагов.

Для вывода передаточной функции схемы:

1. Поместите *DC generator* на входе схемы. Задайте его значение как X.
2. Поместите один или несколько пробников на выход схемы.
3. Поместите график *DC Sweep* и добавьте пробники на него — см. «Добавление кривых на график».
4. Отредактируйте график (**CTRL+E**) и задайте начальное и конечное значения в пределах входной развёртки, которые вам нужны. Проверьте, что переменная развёртки — это X.
5. Нажмите пробел для вызова PROSPICE.
6. Если полученные кривые выглядят не связанно или угловато, когда вы увеличиваете график, увеличьте количество шагов в диалоге *Edit DC Sweep Graph*.

Чтобы увидеть эффект от альтернативных значений цепи:

1. Подготовьте схему, как сделали бы для анализа переходных процессов. Добавьте пробники и генераторы в подходящих точках схемы.
2. Отредактируйте компоненты, влияние значений которых вас интересует. Задайте их значения, как выражения, содержащие X, переменную развёртки. Вы можете редактировать только один или несколько компонентов.
3. Разместите график DC Sweep и добавьте пробники и генераторы на нем — см. «Добавление кривых на график».
4. Отредактируйте график и задайте параметры качания параметров.
5. Нажмите пробел для вызова PROSPICE.

АНАЛИЗ РАЗВЁРТКИ НА ПЕРЕМЕННОМ ТОКЕ

Обзор

Этот тип анализа создаёт семейство кривых АЧХ для разных значений переменной развёртки.

Основное использование этого типа графика — посмотреть, как значения отдельных компонентов влияют на амплитудно-частотную характеристику (АЧХ) вашей схемы.

Метод вычисления

При этом анализе вычисления происходят, как при обычном частотном анализе, исключая то, что выполняются много раз, по одному расчёту для каждого значения переменной качания.

Ограничение на линейность схемы, присущие этому анализу, такие же, как и для обычного частотного анализа — см. «Частотный анализ».

Использование графиков AC Sweep

Как и при Frequency анализе, левая и правая оси отображают, соответственно, усиление и фазу. Также входной генератор должен быть задан, как опорная точка для расчёта усиления.

Чтобы увидеть эффект от альтернативных параметров на частотной характеристике:

1. Подготовьте схему, как для проведения частотного анализа.
2. Отредактируйте нужные вам компоненты. Задайте их свойства, обратив внимание на то, чтобы выражения содержали X, переменную развёртки. Вы можете редактировать один или несколько компонентов.
3. Поместите график AC Sweep и добавьте пробники на него — см. «Добавление кривых на график».
4. Если у вас нет генератора на входе схемы, добавьте его.
5. Выделите и перетащите генератор, который на входе схемы, на график, как опорный генератор.
6. Отредактируйте график (**CTRL+E**) и задайте параметры качания. Задайте параметр *Freq* с интересующей вас частотой.
7. Нажмите пробел для вызова PROSPICE.

АНАЛИЗ ПЕРЕДАТОЧНОЙ КРИВОЙ НА ПОСТОЯННОМ ТОКЕ

Обзор

Этот тип графика был специально разработан для получения семейства характеристических кривых полупроводниковых устройств, хотя подчас он применим и к другим приложениям. Каждая кривая состоит из выводимых напряжений или токов рабочей точки, как функции установленного входного генератора, который «качается» от одного DC значения до другого. Может также устанавливаться дополнительный генератор для получения набора кривых.

Метод вычисления

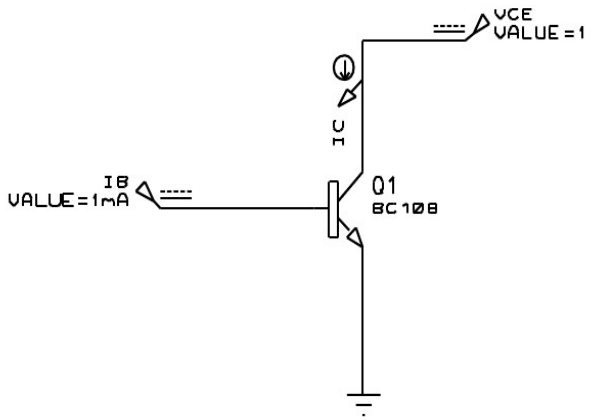
Он очень похож на вычисления для DC Sweep анализа, исключая то, что могут «качаться» два значения. Рабочая точка находится для начального значения каждого генератора, а первый затем проходит по шагам, заданным в установленном диапазоне. После каждого шага первого генератора, значение второго генератора увеличивается; новая кривая выводится для каждого отдельного значения второго генератора.

Использование передаточных (Transfer) графиков

Как и при аналоговом анализе переходных процессов (см. «Аналоговый анализ переходных процессов»), может использоваться либо левая, либо правая, либо обе оси y. Ось x показывает первую переменную качания, а отдельная кривая результат для каждого значения второго генератора. Выражения кривой, описанные в «Диалоговая форма команды Add Trace», могут также использоваться при необходимости.

Для вывода передаточных кривых для транзистора:

1. Постройте схему, похожую на изображённую ниже:



IB — это источник тока, а *VCE* — источник напряжения. *IC* измеряет ток коллектора.

2. Разместите график *Transfer* и добавьте пробник *IC* на него.

3. Укажите на график и нажмите **CTRL+E,** чтобы отредактировать его.
4. Присвойте *Source 1* источнику *VCE*, а *Source 2* источнику *IB*.
5. Задайте диапазоны напряжения и тока в пределах, чувствительных для транзистора — то есть, 0 -> 10V для *VCE* и 100uA -> 1mA для *IB*.
6. Подстройте число шагов для *IB*, дающих значимые промежуточные результаты. Не забывайте, что они дадут на одну кривую больше, чем количество шагов, поскольку один шаг подразумевает два отдельных значения.
7. Закройте диалоговую форму и выберите **ОК** для симуляции графика.

АНАЛИЗ ШУМОВ

Обзор

Движок симулятора SPICE может моделировать тепловой шум, генерируемый в резисторах и полупроводниковых компонентах. Индивидуальный вклад от шумов суммируется на пробнике напряжения в схеме для некоторого диапазона частот. Напряжение шумов, нормализованное корнем квадратным из полосы частот, может затем выводиться как функция частоты. Кривые графика шумов всегда имеют единицы V/ √Hz.

Вычисляются два типа шумов — выходные шумы (Output Noise) и эквивалентные входные шумы (Equivalent Input Noise). Расчёт последних возможен относительно уровня входного сигнала или входных шумов, поскольку представляет уровень шумов на входе, которые потребуют создания действительных шумов на выходе, взятых с усилением схемы на отдельных частотах.

Размещение пробника на левой оси отображает выходной шум, а приведённый ко входу шум можно отобразить, если перетащить пробник на правую сторону графика.

Анализ шумов имеет тенденцию к выводу крайне малых значений (порядка нановольт). По этой причине есть опция отображения их в dB. 0dB относится к 1V/ √Hz.

Корректное моделирование шумов для схем, использующих макромодели IC, не гарантировано, поскольку эти модели могут использовать линейные управляемые источники для моделирования только базового поведения устройства.

Метод вычисления

Рабочая точка цепи вычисляется обычным образом, а затем модель цепи модифицируется для корректного суммирования вкладов шумов. Все генераторы, исключая опорный входной, игнорируются при анализе шумов (исключая момент вычисления рабочей точки), и их можно не удалять перед анализом. PROSPICE будет вычислять тепловой шум для каждого пробника напряжения в схеме, включая те, что связаны с генераторами и записывающими устройствами, поскольку нет способа узнать, какой пробник интересует вас при анализе. Шумовые токи не поддерживаются, поэтому PROSPICE игнорирует все токовые пробники. При анализе шумов производятся отдельные симуляции для каждого пробника, поэтому время симуляции прямо пропорционально количеству размещённых пробников напряжения.

Использование графика шумов

При использовании анализа шумов важно помнить, что эффекты от вклада внешних электрических и магнитных полей не моделируются. Во многих ситуациях шумы, привносимые на вход, могут превосходить шумы, производимые системой.

Если вы разбиваете схему, используя записи (tapes, см. «Записи и разбиение»), вы должны быть уверены, что только текущая часть учитывается PROSPICE симулятором. Вид шума, взятый в изоляции, всё ещё полезен, но самописцы (tapes) должны удаляться, чтобы увидеть эквивалентный шум всей цепи.

В порядке определения источника шума в цепи отдельные напряжения шумов добавляются в журнал симуляции (simulation log). По порядку рассматривается каждый пробник, и для каждого компонента получается вклад шума. Вклады обсчитываются и выводятся как квадраты значений. Этот процесс выполняется для начальной и конечной частот. Если вы имеете много пробников, может быть, лучше выводить результирующие данные.

Чтобы провести анализ шумов схемы:

1. Разместите график *Noise* и отредактируйте его, чтобы задать нужный диапазон частот и входной опорный (reference) генератор.
2. Добавьте пробники напряжения на выходы схемы или в другие интересующие вас точки, и на график (см. «Добавление кривых на график»).
3. Нажмите пробел, чтобы запустить симулятор PROSPICE.
4. Если уровень шума требуется уменьшить, проверьте отчёт о симуляции (simulation log, **CTRL+V**), чтобы определить источник шума.

АНАЛИЗ ИСКАЖЕНИЙ

Обзор

Анализ искажений (Distortion analysis) определяет уровень гармонических искажений, получаемых в схеме при тестировании. Это могут быть либо 2я, либо 3я гармоника основного сигнала или интермодуляционные продукты двух тестовых сигналов.

Искажения создаются нелинейностями передаточной функции схемы — схемы, составленные только из линейных компонентов (резисторов, конденсаторов, индуктивностей, линейных управляемых источников) не будут создавать искажений. Анализ искажений SPICE моделей подразумевает искажения на диодах, биполярных транзисторов, JFET и MOSFET.

Корректное моделирование искажений для цепей, включающих IC макро модели, не гарантировано, поскольку эти модели могут использовать линейные управляемые источники для моделирования только основного поведения устройства.

Похожая информация может быть получена при использовании Фурье анализа (Fourier analysis), но *Distortion* анализ также способен показать, как меняются искажения, когда основная частота «качается».

Метод вычисления

Этот анализ основан на малосигнальных (AC) моделях для устройств в схеме, так что первый шаг — это расчёт рабочей точки. Каждая модель нелинейного компонента затем добавляет комплексные значения искажений для подходящих гармоник, в зависимости от того, как много устройств влияет на входной основной сигнал. Величины, которой достигают эти гармоники, появляются на выходе и определяют выводимые значения. Процесс повторяется в заданном диапазоне входных частот. Фактически, математика этого анализа крайне сложна и подразумевает конструкции и манипуляции с рядами Тэйлора, чтобы представить нелинейные устройства.

Заметьте, что используются комплексные значения, так что анализ предоставляет информацию и об амплитуде, и о фазе каждой гармоники.

Для одночастотного гармонического анализа искажений получаются две кривые на графике — одна для 2й гармоники, а вторая для 3й.

Для интермодуляционных искажений используются две входные частоты, заданные в терминах отношения между 2й частотой (F2) и основной (F1). Отображаются три кривые, показывающие интермодуляционные артефакты на F1+F2, F1-F2 и 2F1-F2.

Некоторого внимания потребует выбор отношения F2/F1, поскольку иначе могут обнаружиться математические странности. Например, если F2/F1 равно 0.5, то значение F1-F2 равно F2, а вывод значения F1-F2 будет бессмысленным, поскольку оно совпадает со значением второй основной частоты. В общем, лучше остановить свой выбор для этого отношения на иррациональном числе. Так F2/F1=49/100 будет много лучше. Заметьте, что $F2/F1 < 1$.

Использование графика Distortion

График искажений показывает амплитуду гармоник на левой оси и фазу (обычно менее интересующую) на правой оси. Тестовая частота (F1) выводится на оси x.

Для гармонического анализа (Harmonic Distortion) при одной частоте на входе (F1) выводятся две кривые — по одной для каждой составляющей, 2F1 и 3F1.

Для интермодуляционного анализа (две входные частоты, F1 и F2) выводятся три кривые, показывающие составляющие на F1+F2, F1-F2 и 2F1-F2.

В любом случае, какая кривая есть какая, может быть определено, если вывести курсор на график — кривая, на которую вы указали, будет идентифицирована на правой стороне строки состояния.

ФУРЬЕ АНАЛИЗ

Обзор

Фурье анализ (Fourier Analysis) — это процесс преобразования данных временной области в частотную, а результаты похожи на те, что получаются при подключении спектрального анализатора вместо осциллографа. Он особенно полезен при анализе гармонических составляющих сигнала, возможно, при исследовании отдельных типов искажений, но и имеет множество других приложений.

Метод вычисления

Фурье анализ начинается с выполнения анализа переходных процессов (Transient), а затем выполняется быстрое преобразование Фурье над полученными данными. Этот процесс выполняется дискретным по времени выбором образов из данных временной области с последующим применением критерия Найквиста. Положим просто, что наивысшая частота, которая рассматривается, равна половине выбранной частоты. Однако могут обнаружиться другие, вводящие в заблуждение эффекты при наложении выбранной частоты с гармониками входного сигнала, которые выше половины выбранной частоты. Чтобы минимизировать подобные эффекты, можно применить к входным данным разного типа интервалы до быстрого преобразования Фурье.

Использование графиков Fourier

В первом приближении вам нужно приготовить вашу схему, как для анализа переходного процесса, исключая то, что вы должны использовать график *Fourier* вместо (или также как!) графика *Analogue*.

Для анализа частотного содержимого сигнала:

1. Приготовьте схему, как для анализа переходных процессов.
2. Добавьте график *Fourier* в проект и перетащите на него пробник, соединённый с нужными вам точками.
3. Выберите начальное и конечное время и значение частота/разрешение (frequency/resolution), подходящее для сигнала, который вы анализируете. Если возможно, выберите временной интервал и частотное разрешение, которые соответствуют основной частоте анализируемого сигнала.
4. Нажмите пробел для вызова PROSPICE.

Если время начала и окончания одинаковы и для графика переходного процесса, и для графика анализа Фурье, PROSPICE не нужно будет повторять симуляцию схемы между этими двумя анализами. Вместо этого ISIS выполнит только быстрое преобразование Фурье с данными на существующей временной области.

АУДИО АНАЛИЗ

Обзор

PROTEUS VSM включает ряд возможностей, которые позволяют вам услышать выход вашей схемы (при наличии, конечно, звуковой карты!). Главный компонент этого — график *Audio*. Это, в основном, то же, что и график *Analogue*, исключая то, что после симуляции генерируется файл Windows WAV из данных временной области, который и воспроизводится через вашу звуковую карту.

Файлы WAV могут также экспортироваться для использования в других приложениях.

Метод вычисления

Аудио анализ выполняется точно так же, как и анализ переходных процессов, исключая то, что после симуляции данные заново обрабатываются на одной из стандартных для PC частот дискретизации (11025, 22050 или 44100 Гц), а затем записываются в формат WAV, используя стандартную функцию Windows, предназначенную для этой цели. В завершение подаётся команда проиграть WAV файл на вашем аудио оборудовании.

Использование графика Audio

В первом приближении вам нужно приготовить схему, как для анализа переходного процесса, исключая то, что вы должны использовать график *Audio* вместо *Analogue*.

Чтобы услышать аудио выход схемы:

1. Подготовьте схему, как для анализа переходных процессов.
2. Добавьте график *Audio* в проект и перетащите пробник с выхода схемы на график.
3. Выберите начальное, конечное время и время цикла для генерации сигнала подходящей длительности, но с минимумом актуальной симуляции. Создание одной секунды аудио сигнала с анализом 1 мс и циклом в 1000 раз значительно быстрее, чем анализ схемы для целой секунды.
4. Выберите разрешение для образца и диапазон, подходящие для естественного прослушивания сигнала.

   Используйте 16-битовое разрешение, если вы не создаёте большие файлы и/или при нехватке дискового пространства.

   Большинство звуковых систем PC не работают с частотой дискретизации больше, чем 44.1 кГц.
5. Нажмите пробел, чтобы вызвать PROSPICE.
6. Нажмите **CTRL-SPACE** для повторного воспроизведения без повторения симуляции.

ИНТЕРАКТИВНЫЙ АНАЛИЗ

Обзор

Интерактивный анализ комбинирует преимущества симуляции, основанной на интерактивности и графиках. Симулятор стартует в интерактивном режиме, а результаты записываются и отображаются на графике, как при анализе переходных процессов.

Этот вид анализа особенно полезен при проверке того, что случилось при проведении отдельных операций в проекте, и его можно рассматривать, как комбинацию запоминающего осциллографа и логического анализатора в одном устройстве.

Метод вычисления

Метод вычисления идентичен тому, что выполняется для анализа переходных процессов в смешанном режиме, исключая то, что симулятор работает в интерактивном режиме. Следовательно, операции переключения, операции с клавиатурой и другими приводами в схеме будут сказываться на результатах. Также симуляция будет происходить со скоростью, определяемой параметром *Animation Timestep*, а не с наиболее возможной скоростью.

- Остерегайтесь захвата очень большого количества данных. Тактовый генератор процессора, работающий на реальной скорости, генерирует миллионы событий в секунду, а это будет занимать много мегабайт, если будет захвачено и отображено на графике — особенно, если вы просматриваете шину данных или адресную шину. Вы можете легко «подвесить» вашу систему, если ISIS загрузит 20 или 30 Мбайт результирующих данных.

  Возможно, лучше использовать в этом случае *Logic Analyser*, если нет возможности получить требуемые данные за относительно короткий период симуляции.

- Как и с обычной интерактивной симуляцией, многофрагментарные схемы не поддерживаются, а любые записывающие объекты, не установленные в режим проигрывания, будут автоматически удалены из схемы.

Использование интерактивных графиков

Обычный сценарий для интерактивного анализа таков, что вы проверяете проект с помощью интерактивной симуляции и находите, что происходит нечто странное при обычных управляющих операциях. В первом приближении вы можете попробовать использовать виртуальные инструменты, чтобы увидеть, что происходит, но в некоторых случаях есть необходимость захватить результаты для графика и внимательно исследовать их.

Чтобы выполнить интерактивный анализ:

1. Добавьте пробники в интересующие вас точки.
2. Разместите график *Interactive* в свободной области схемы и перетащите на него пробник.
3. Отредактируйте график и выберите подходящие времена начала и окончания.

Заметьте, что PROSPICE не будет захватывать данные от датчика до момента старта — это избавляет от «горы» данных, которые получит ISIS.

4. Установите любые интерактивные компоненты на схему в подходящем начальном состоянии.
5. Приготовьтесь к выполнению интерактивных операций и затем нажмите клавишу **SPACE** (пробел). Вы должны уложиться с операциями с активными компонентами в промежуток времени, который задали на шаге 4, если эффект от этих действий должен быть записан. Если это трудно выполнить, вы можете либо увеличить время останова, либо увеличить параметр *Timestep Per Frame* (время шага на кадр).

ЦИФРОВОЙ АНАЛИЗ СООТВЕТСТВИЯ

Обзор

Анализ соответствия (conformance analysis) сравнивает один набор результатов цифровой симуляции с другим. Идея в том, что проект, который был ранее признан рабочим, может быть быстро перепроверен после модификации, с тем чтобы удостовериться в отсутствии нежелательных сторонних эффектов, как результата изменений. Это обычно относится к приложениям на основе микроконтроллеров, где всё приложение может нуждаться в перепроверке после изменений, которые были внесены в код программы.

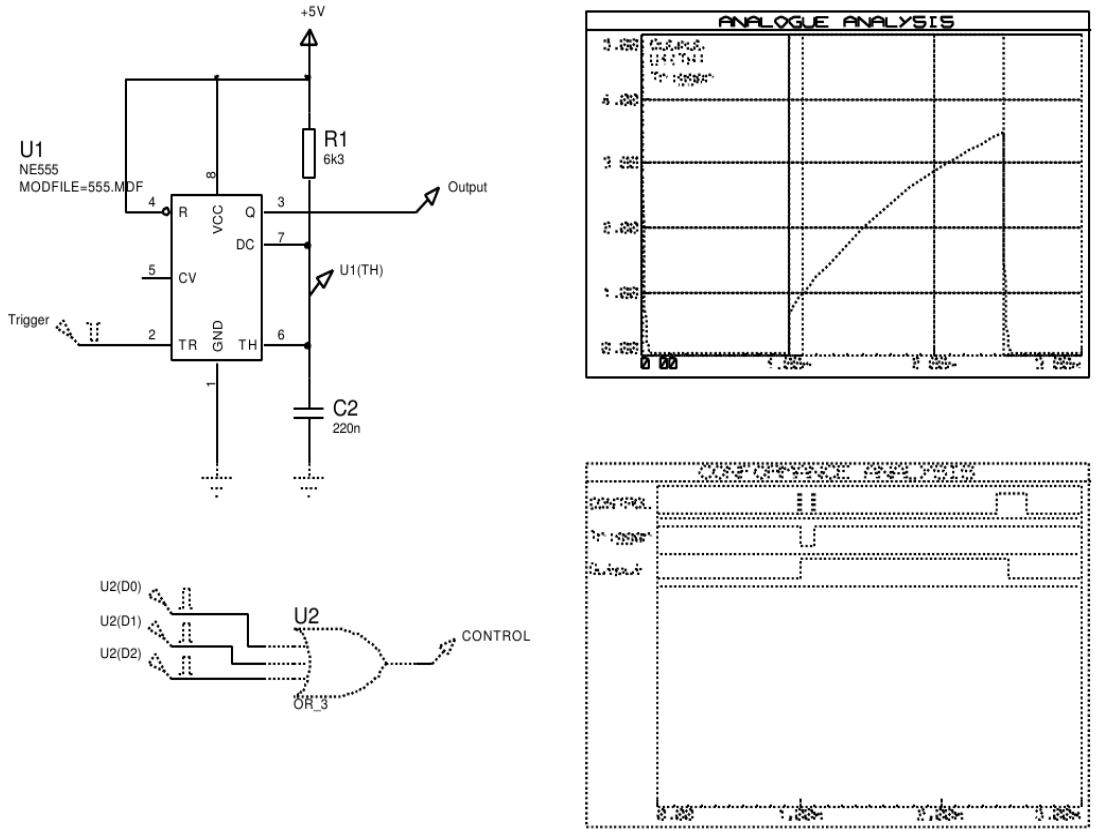
Метод вычисления

Метод вычисления идентичен тому, что выполняется для цифрового анализа переходных процессов, исключая то, что могут запоминаться два набора результатов на графике. Мы назвали два набора результатов: проверяемые результаты и опорные результаты.

Соответствие или нет — определяется сравнением проверяемых и опорных результатов на каждом фронте первой кривой. Мы назвали эту кривую контрольной, и она отображается иным цветом, чтобы отличать её от остальных. Очень важно, что нет требования для фронтов в тестовой и опорной копиях контролируемой кривой быть совпадающими по времени. Это означает, что изменения в абсолютном времени событий внутри данных результата, не свидетельствуют несомненно об отсутствии соответствия. Это обычно относится к приложениям с микроконтроллерами, где любые изменения кода будут ограничены во влиянии на абсолютное время событий внутри системы. В этих случаях контрольная кривая может быть сгенерирована самим кодом на входе и/или выходе в программные процедуры при тестировании.

Использование графиков Conformance

Обычно анализ соответствия используется, как часть стратегии тестирования, и чаще во встроенных системных приложениях, хотя мы также используем их в нашей фирме для тестирования наших моделей симуляторов. На простом примере ниже будет показано, как использовать график *Conformance* для тестирования операций моностабильного 555.



Проверяемая схема состоит из U1, R1 и C2, которые соединены в классическую схему моностабильного 555.

Схема запускается в 1 мс цифровым импульсным генератором, и выходное напряжение отображается на графике аналогового анализа. Чтобы проверить корректность работы схемы, было бы правильно установить следующие факты:

- Что выходной сигнал в низком состоянии до прихода управляющего импульса.
- Что выходной сигнал переходит в высокое состояние вскоре после перехода управляющего сигнала в низкое состояние.
- Что выходной сигнал остаётся в высоком состоянии, когда управляющий сигнал становится высоким.
- Что выходной сигнал остаётся в высоком состоянии на время около 1.5 мс.
- Что выходной сигнал переходит в низкое состояние после этого интервала времени.

Чтобы получить это, используя график *Conformance*, нам нужно отобрать выходные данные около каждого фронта перехода сигнала управления, а также с любой стороны выходного сигнала. Мы можем определить допуск для времени задержки одновибратора, выбирая промежуток между последними двумя точками пробы.

Это получается, если использовать последовательность импульсов цифрового генератора,

выход которого комбинируется с вентилем ИЛИ. Ширина каждого импульса генератора определяет временной допуск для каждого ожидаемого события. Например, третий импульс генератора задаёт переключение между 2.4 мс и 2.6 мс, давая допуск в +/- 100us на ширину ожидаемого выходного импульса.

Заметьте, что вы должны использовать цифровой генератор или генератор шаблонов для управляющего сигнала; использование простого тактового генератора приведёт к проверке в регулярные временные интервалы, которые, фактически, будут достаточны для многих приложений.

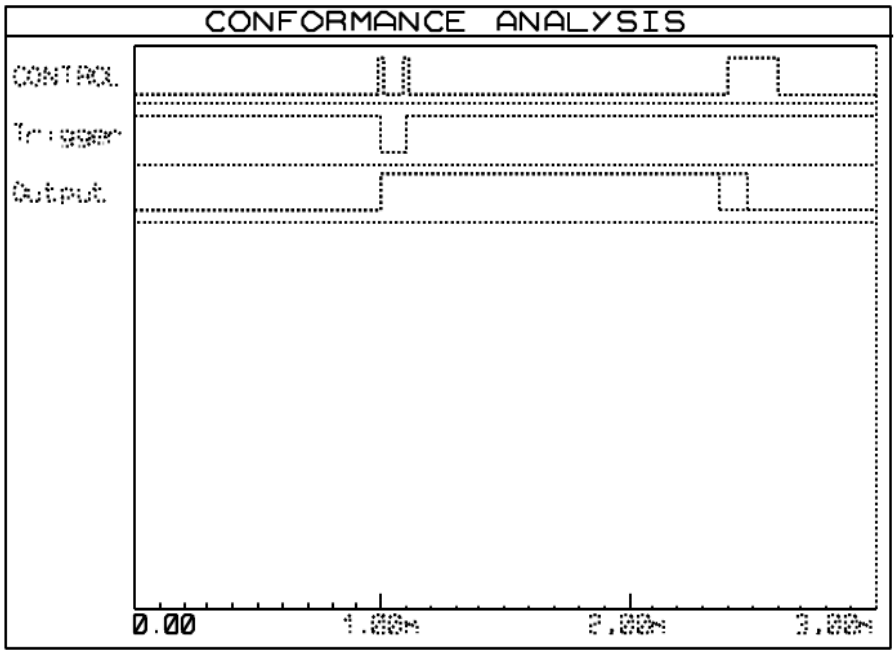
В основном процедура может быть суммирована следующим образом.

Чтобы подготовить анализ соответствия:

1. Решите, что вы хотите проверить и в какое время.
2. Постройте тестовую схему, которая генерирует выходной сигнал, который вы проверяете и управляющий сигнал, меняющий состояние каждый раз, когда вы хотите подтвердить результаты.
3. Поместите график *Conformance* и подготовьте его точно так, как вы сделали бы для цифрового анализа, удостоверившись, что контрольная кривая находится в верхней части графика.
4. Запустите симуляцию и проверьте, что результаты таковы, как ожидалось, и что переходы на контрольной кривой обнаруживаются в те моменты, когда вы хотели бы проверить результаты.
5. Отредактируйте график *Conformance* и нажмите кнопку **Store Results** (сохранить результаты). Это сделает текущие отображаемые результаты опорными результатами. Затем это отобразится приглушённым цветом и график будет сравниваться с любыми новыми результатами, когда он будет заново симулирован.
6. Сохраните проект — теперь он пригоден для целей проверки в будущем.

Теперь давайте предположим, что из-за нехватки компонентов, значения R1 и C2 должны быть изменены. Если мы сделаем C2 равной 100nf, то можем ли взять R1 равным 15k, и будет ли проект ещё работать с такой спецификацией?

Результаты этого эксперимента показаны ниже:



Ответ — нет. Выходной импульс теперь стал слишком коротким, и в отчёте симуляции для графика появляется следующая информация:

```
Comparing Results...
Data mismatch at time 2.40m in trace 'Output'.
Data signatures were 'L' and 'H'.
Conformance analysis FAILED.
```

Заметьте, что курсор на графике в позиции времени первой разницы, то есть, 2.4 мс.

Есть фактически три пути запуска анализа соответствия:

- Новой симуляцией графика обычным образом, то есть, нажатием на пробел или использованием команды *Simulate Graph* из раздела *Graph*.
- Использованием команды *Conformance Analysis* из раздела *Graph* основного меню. Это заново симулирует все графики соответствия в проекте и сообщит о любых ошибках.
- Только для опытных пользователей — можно использовать командную строку. Ввод

  ISIS <filename> /V

  выполнит глобальный анализ соответствия, как выше. Если какой-то из графиков станет причиной отказа, ISIS установит уровень ошибки 1. Это позволяет использовать пакетные файлы для автоматического выполнения множественных тестов.

РАБОЧАЯ ТОЧКА НА ПОСТОЯННОМ ТОКЕ

Это единственный тип анализа (*DC OPERATING POINT*), для которого нет соответствующего графика. Тем не менее, вычисленную информацию можно увидеть, используя интерфейс «point and shoot, прицелься и стреляй» в ISIS.

Важно понимать, что рабочая точка вычисляется для графика тока (*Current Graph*), и что на схеме должен быть график для расчёта рабочей точки. Основанием для этого служит то, что без графика у системы нет понимания, какие части схемы следует симулировать. Вдобавок, функциональная зависимость от *Property Expression Evaluation* (оценка выражений свойств) и возможности задать свойства на графике означает, что актуальные значения компонентов могут зависеть от того, какой используется график.

Чтобы увидеть значения рабочей точки:

1. Настройте схему как для анализа переходных процессов.
2. Добавьте график на схему и добавьте пробники и генераторы на него, соответствующие секциям схемы, которые вы хотите симулировать.
3. Симулируйте рабочую точку, нажав **ALT+SPACE**. Пробники напряжения и тока будут отображать их значения рабочей точки на постоянном токе немедленно.
4. Щёлкните по иконке *Multimeter*.
5. Укажите на любой компонент и щёлкните левой клавишей мышки, чтобы увидеть его информацию о рабочей точке.

Пара замечаний:

- Можно вычислять рабочую точку без графика. В этом случае выполняется симуляция всей схемы, как единой секции — то есть, независимо от любых пробников, генераторов или самописцев. Это может окончиться неудачей, если есть объекты, не имеющие моделей.
- Компоненты, которые фактически являются родительскими для подсхем, или которые моделируются файлами MDF или SPICE, будут отображать только информацию о базовых рабочих точках, то есть, узловые напряжения.

ГЕНЕРАТОРЫ И ПРОБНИКИ

ГЕНЕРАТОРЫ

Обзор

Генератор — это объект, который может быть установлен, чтобы дать сигнал в той точке, к которой он подключён. Есть множество типов генераторов, каждый из которых генерирует разного рода сигналы:

- *DC* — источник постоянного напряжения.
- *Sine* — генератор синусоидального напряжения с управляемыми амплитудой, частотой и фазой.
- *Pulse* — аналоговый импульсный генератор с управляемыми амплитудой, периодом и временем нарастающего и спадающего фронтов.
- *Exp* — экспоненциальный импульсный генератор; производит импульсы такого же вида, какой имеют RC цепи при заряде/разряде.
- *SFFM* — частотномодулированный одночастотный генератор — производит сигнал, определяемый частотной модуляцией одного синусоидального сигнала другим.
- *Pwlin* — генератор кусочно линейных сигналов; для создания импульсов и сигналов произвольной формы.
- *File* — как и выше, но данные берутся из ASCII файла.
- *Audio* — используются Windows WAV файлы для входных сигналов. Особый интерес представляет комбинация с графиком *Audio*, поскольку вы можете прослушать полученный от вашей схемы эффект на аудио выходе компьютера.
- *DState* — установившийся логический уровень.
- *DEdge* — единственный логического уровня переход или фронт.
- *DPulse* — единственный цифровой тактовый импульс.
- *DClock* — цифровой тактовый сигнал.
- *DPattern* — произвольная последовательность логических уровней.

Первые восемь типов генераторов предполагаются быть аналоговыми компонентами, и симулируются ядром SPICE симулятора, тогда как остальные относятся к цифровым цепям и поддерживаются DSIM.

Размещение генераторов

Чтобы разместить генератор:

1. Получите список типов генераторов, выбрав иконку *Generator Mode*. Список типов генераторов отображается в *окне выбора*.
2. Выберите тип генератора, который вы хотите поместить, в окне выбора. ISIS покажет вид генератора в *окне просмотра*.
3. Используйте иконки поворота и отражения (Rotation и Mirror), чтобы придать нужное положение генератора, как считаете удобным.
4. Переместите мышку в *окно редактирования*, и щёлкните левой клавишей в нужном вам месте.

Вы можете разместить генератор непосредственно на существующем проводе, расположив его так, чтобы его точка соединения касалась провода. В качестве альтернативы — размещение нескольких генераторов на свободном месте с последующим их соединением со схемой.

Когда генератор размещён без соединения с существующими проводами, он имеет имя по умолчанию со значком вопроса (?), чтобы показать, что он не отмечен. Когда генератор соединён с сетью (возможно, когда размещён на ней, если помещается непосредственно на существующий провод), он получает имя этой сети, или, если сеть сама не помечена, ссылку на компонент и имя вывода первого, что соединён с сетью. Имя генератора будет автоматически обновляться, когда он отсоединяется или когда перетаскивается от одной сети к другой. Вы можете задать ваше собственное имя генератору, отредактировав объект, в последнем случае имя принадлежит только ему и не обновляется.

См. «Имена сетей» в руководстве к ISIS, где рассказано о сетях и их наименовании.

Редактирование генераторов

Любой генератор можно отредактировать, используя одну из основных техник редактирования ISIS, самая простая из которых — щёлкнуть правой клавишей мышки по объекту и выбрать из выпадающего меню *Edit Properties*, или указать на генератор и нажать **CTRL+E**.

Диалоговая форма Edit Generator предлагает ряд общих полей, а затем дальнейший набор полей, который меняется согласно типу генератора. Общие поля описаны ниже:

Name Имя генератора.

Вы можете изменить имя генератора, вписав новое в поле *Name*. Когда вы меняете имя генератора вручную, ISIS никогда не изменит его, даже если вы переместите генератор к новой цепи.

Если вам удобнее вернуться к автоматическому наименованию, очистите это имя, оставив строку пустой, нажав один раз **ESC**, а затем щёлкнув **ОК**.

Type Тип генератора. Вы можете изменить тип генератора (от размещённого типа), выбрав кнопку для нового требуемого типа.

Это управление определяется специальным полем генератора, которое отображается на правой стороне диалога.

Current Source За исключением DIGITAL генератора все остальные типы способны оперировать либо с источником напряжения, либо с источником тока. Установка последнего флажка превращает генератор в источник тока.

Isolate Before Этот флажок управляет, будет или нет генератор размещён в середине провода, образуя обрыв провода, к которому он подключён, изолируя сеть, отходящую от генератора, от сети за генератором.

Установка флажка не сказывается на генераторе, соединённом непосредственно с проводом сети из единственного провода.

Manual Edits Когда флажок установлен, свойства генератора отображаются как текстовый список свойств. Это поддерживается вручную для обратной совместимости с предыдущими версиями программы. Однако опытные пользователи могут использовать выражения свойств в свойствах генератора, а это возможно только в ручном режиме редактирования.

Генераторы постоянного тока

DC generator используется для генерации постоянного уровня аналогового напряжения или тока. Генератор имеет единственное свойство, которое задаёт выходной уровень.

Генератор синуса

Sine generator используется для создания непрерывного синусоидального сигнала одной частоты. Некоторые параметры могут настраиваться:

- Выходной уровень задаётся как пиковая амплитуда (*VA*) с дополнительной постоянной составляющей (*VO*). Амплитуда может также задаваться в терминах действующего (RMS) или от пика до пика значений.
- Частота генерации может задаваться в Гц (*FREQ*) или как период (*PER*), или в терминах числа циклов на всем графике.
- Фазовый сдвиг может задаваться либо в градусах (*PHASE*), либо временной задержкой (*TD*). В последнем случае генерация не начнётся раньше заданного времени.
- Экспоненциальный спад кривой после старта осцилляции задаётся коэффициентом затухания *THETA*.

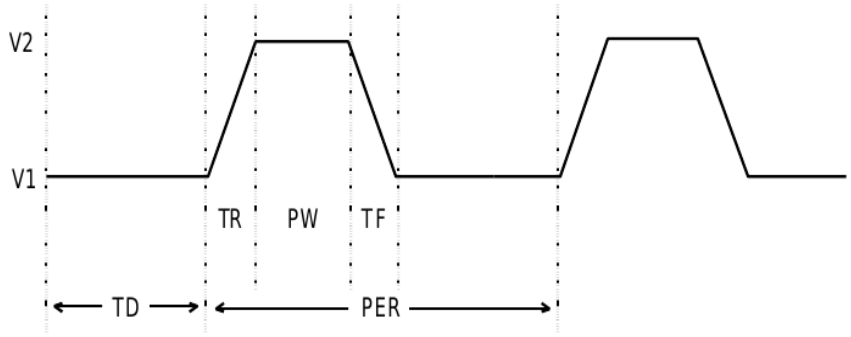
Математически выход задаётся:

$$V = VO + VAe^{-(t-TD)\,THETA}\sin(2\pi FREQ(t + TD))$$

для t >= TD. Для t < TD выход — просто постоянная составляющая (DC offset) *VO*.

Импульсный генератор

Pulse generator используется для получения разных повторяющихся входных сигналов для аналогового анализа. могут быть заданы прямоугольные, пилообразные и треугольные импульсы, равно как и единичные короткие импульсы. Заметьте, что времена подъёма и спада не могут быть нулевыми, так что реально прямоугольные импульсы не позволительны. Причина этого в том, что мгновенные изменения в основном не позволительны в PROSPICE. Операции с импульсным генератором лучше описываются следующей диаграммой:

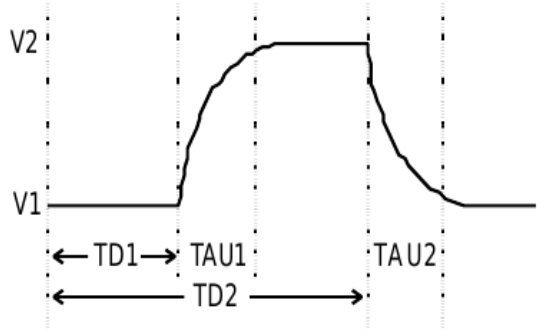


где

PER	Период сигнала. Если не задан, то используется **FREQ**.
FREQ	Частота сигнала. Она предопределена для одного периода анализа переходного процесса.
V1	Значение низкого уровня выходного сигнала.
V2	Высокий уровень выхода.
PW	Время за которое выход становится **V2** в каждом цикле. Это не включает **TR** и **TF**.
TR	Время переднего фронта — время между LOW и HIGH в каждом цикле. Это предопределено как1ns.
TF	Время заднего фронта — время между HIGH и LOW в каждом цикле. Предопределено как **TR**.
TD	Время задержки. Выход генератора стартует при **V1** и будет оставаться на этом уровне в течение **TD** секунд.

Экспоненциальные генераторы

Exp generator производит сигналы похожие на заряд и разряд RC цепи. Параметры лучше всего описываются диаграммой ниже.



где параметры это:

V1	Значение низкого уровня выходного сигнала.
V2	Высокий уровень выхода.
TD1	Начальное время подъёма сигнала.
TAU1	Постоянная времени подъёма сигнала. Это время, за которое напряжение увеличивается примерно до 0.63 значения полного потенциала.
TD2	Начальное время спада сигнала. Заметьте, что **TD2** отсчитывается от нуля.
TAU2	Постоянная времени спада сигнала.

Математически сигнал описывается на трёх интервалах:

0 to TD1 $V1$

TD1 to TD2 $V1+(V2-V1)\left(1-e^{\frac{-(t-TD1)}{TAU1}}\right)$

TD2 to TSTOP $V1+(V2-V1)\left(1-e^{\frac{-(t-TD1)}{TAU1}}\right)+(V1-V2)\left(1-e^{\frac{-(t-TD2)}{TAU2}}\right)$

Генераторы частотно моделированной частоты

SFFM generator производит сигнал, который представляет результат частотной модуляции одной синусоидой другой. Математически это представлено так:

$$V = VO + VA\sin\left(2\pi FCt + MDI\sin(2\pi FSt)\right)$$

где параметры следующие

VO	Постоянная составляющая (DC offset) напряжения.
VA	Амплитуда несущей.
FC	Частота несущей.
FS	Частота сигнала.
MDI	Индекс модуляции.

Генераторы кусочно-линейных форм сигнала

Piece-wise Linear generator используется для создания аналоговых сигналов, которые слишком сложны для импульсного генератора или для воспроизводства измеренных сигналов. Выходной сигнал описывают, используя пары значений для времени и выходной амплитуды. Выход генератора в промежутках времени затем линейно интерполируется.

Диалоговая форма генератора кусочно-линейных форм состоит из графического редактора на котором вы можете перетаскивать точки данных. Операции можно суммировать так:

- Щелчком левой клавиши мышки поместите новую точку данных.
- Чтобы переместить точку данных, перетащите её, используя левую клавишу мышки.
- Чтобы удалить точку данных щёлкните правой клавиши мышки.

Есть следующие ограничения:

- Всегда должна быть точка при нулевом времени, хотя её у-значение может быть изменено, то есть, вы можете только перетаскивать их вверх или вниз.
- Если вы хотите создать вертикальный фронт, графический редактор разделит две точки данных минимальным значением времени подъёма/спада.

Если у вас есть табличные данные, легче может оказаться режим ручного редактирования. В этом случае каждая точка данных задаётся свойством формы V(t). Следующий пример показывает, как это работает.

```
V(0)=0
V(2n)=0
V(3n)=1
V(5n)=1
V(6n)=0
```

Если количество данных очень велико, может оказаться легче использовать *FILE generator*.

Файловые генераторы

File generator используется для испытания схемы аналоговым сигналом, который задаётся серией временных точек и значений данных, содержащихся в ASCII файле. Это очень похоже на работу с кусочно-линейным генератором, исключая то, что значения данных находятся вовне, а не в свойствах устройства.

Диалоговая форма имеет только одно поле, в котором задаётся имя файла данных. Нет предопределённых выражений для этих файлов, и файл должен располагаться в той же директории, что и файл проекта, если не задан полный путь к файлу.

Формат файла данных

Файл ASCII данных должен быть отформатирован парами время/напряжение для каждой линией, разделёнными пробелом (пробелы или табуляция, а не запятые). Значения времени должны последовательно возрастать и все значения должны быть просто числами с

плавающей точкой (суффиксы не допустимы).

Пример

Следующий пример файла данных производит три цикла пилообразных импульсов с временем подъёма 0.9ms, временем спада 0.1ms и амплитудой 1V.

```
0          0
9E-4       1
1E-3       0
1.9E-3     1
2E-3       0
2.9E-3     1
3E-3       0
```

Аудио генераторы

Audio generator используется для испытания схемы файлом Windows WAV. В соединении с *Audio graphs* (аудио графиками) возможно услышать результаты воспроизведения звукового сигнала через симулируемую схему.

- Имя файла подразумевается с предопределённым расширением WAV, файл должен располагаться в той же директории, что DSN файл, если не задан полный путь к файлу.
- Амплитуда может быть задана либо в терминах максимального абсолютного значения для положительной и отрицательной полуволн, либо как значение от пика до пика.
- Может быть задана постоянная составляющая (DC offset); если она нулевая, выходное напряжение осциллирует около нуля.
- Для стерео WAV файлов вы можете выбрать, какой канал будет проигрываться, либо определить данные как моно.

Цифровые генераторы

Есть пять подтипов цифровых генераторов (digital generator):

- *Single Edge* — единственный переход от низкого к высокому или от высокого к низкому уровням.
- *Single Pulse* — пара переходов в противоположных направлениях, которые вместе формируют положительный или отрицательный импульс. Вы можете задать либо времена каждого фронта (начальное и конечное время), либо начальное время и ширину импульса.
- *Clock* — последовательность импульсов, разделённых паузами. Вы можете задать начальное значение и время первого фронта, но можно задать период или частоту. Период задаёт время целого цикла, не ширину единственного импульса или паузы.

- *Pattern* — наиболее гибкий и может, фактически, генерировать любые другие типы. *Pattern generator* определён в терминах следующих параметров:

  Initial State Это значение в нулевой момент времени, а также значение, используемое при поиске рабочей точки схемы в смешанном режиме.

  First Edge Это время, когда шаблон действительно стартует; выход будет оставаться в значении начального состояния, пока не придёт время.

  Timing Каждый шаг шаблона может получить то же время (равное времени импульс/пауза) или может иметь разные времена для высокого и низкого уровня. В этом случае ширина *Pulse* задаётся временем для значений логической «1» и *Space Time* задаётся для значений логического «0».

  Transitions Выход может быть определён для постоянного сигнала до окончания симуляции с повторением шаблона или для только фиксированного числа переходов.

  Bit Pattern Предопределённый шаблон — это просто альтернатива последовательности высокое-низкое состояние. Альтернативно, может быть задана строка шаблона. Строка шаблона может состоять из следующих символов:

  0,L Выходной сигнал переходит на строго низкий уровень (заметьте, «L» в верхнем регистре).

  1,H Выходной сигнал переходит на строго высокий уровень (заметьте, «H» в верхнем регистре).

  1 Выходной сигнал переходит на слабо низкий уровень (заметьте, «l» в нижнем регистре).

  h Выходной сигнал переходит на слабо высокий уровень (заметьте, «h» в нижнем регистре).

  F,f Выходной сигнал на плавающем уровне.

- *Script* — генератор будет управляться скриптом DIGITAL BASIC. Генератор получает доступ к скрипту через декларацию переменной PIN с тем же именем, что и ссылка генератора.

ПРОБНИКИ

Обзор

Пробник (*probe*) — это объект, который может быть задан для записи состояния цепи, к которой он подключён.

Есть два типа пробников:

- *Voltage probes* (пробники напряжения) — они могут использоваться и для аналоговой, и для цифровой симуляции. В шаблоне они записывают действительные уровни напряжения, хотя в последнем случае они записывают логические уровни и их длительность.
- *Current probes* (пробники тока) — они могут использоваться только для аналоговой симуляции и должны располагаться на проводе так, чтобы провод проходил через пробник. Направление измерения отображается на графике токового пробника.

  Вы не можете использовать токовый пробник для цифровой симуляции или на шине.

Пробники в большинстве случаев используются при симуляции, основанной на графиках (Graph Based Simulations), но могут использоваться и при интерактивной симуляции для отображения данных рабочей точки и для части схемы.

Размещение пробников

Чтобы разместить пробник:

1. Вначале выберите либо *Voltage Probe Mode*, либо *Current Probe Mode* иконку. ISIS покажет предварительный вид пробника в *окне предпросмотра*.
2. Используйте иконки поворота и отражения для придания пробнику нужного положения, в зависимости от того, как вы их намерены разместить.

   Заметьте, что для пробника тока ориентация важна. Направление измерения тока указано стрелкой, заключённой в кружок, формирующий символ пробника.
3. Переместите мышку в окно редактирования, нажмите левую клавишу мышки и перетащите пробник в нужное место. Щёлкните левой клавишей мышки ещё раз.

   Вы можете размещать пробник непосредственно на существующем проводе так, чтобы его точка подключения касалась провода. Но вы можете разместить несколько пробников на свободном месте вашего проекта и соединить со схемой позже.

Когда пробник размещён так, что не соединяется ни с одним из существующих проводов, он получает предопределённое имя в виде знака вопроса «?», чтобы показать, что он не подключён. Когда пробник присоединён к цепи (возможно, при размещении его непосредственно на проводе), ему присваивается имя цепи или, если цепь сама не имеет имени, он получает ссылку на компонент и имя вывода, который первый подключён к цепи. Имя пробника будет автоматически обновляться, когда он отсоединяется или когда перетаскивается от одной цепи к другой. Вы можете присвоить пробнику собственное имя,

отредактировав пробник, но в этом случае имя остаётся постоянным и не будет обновляться.

Установки пробника

Диалоговая форма *Edit Probe* позволяет подправить два параметра:

LOAD Resistance (сопротивление нагрузки)

Пробник напряжения может быть задан так, чтобы иметь сопротивление нагрузки для схемы. Это полезно, там где нет пути протекания постоянного тока на землю от точки измерения.

Record Filename

Оба побника, и напряжения и тока, могут записывать данные в файл, который может затем проигрываться с помощью *Tape Generator* (генератор записи). Эта возможность позволяет вам создавать тестовые сигналы, используя одну схему, а затем проигрывать её в другой.

См. «Записи и сегментирование», где информации больше.

ИСПОЛЬЗОВАНИЕ SPICE МОДЕЛЕЙ

ОБЗОР

Многие из основных производителей компонентов сегодня поддерживают SPICE совместимые модели для симуляции для своей линейки продуктов. Поскольку симулятор PROSPICE основан на оригинальной версии Berkeley SPICE, у вас могут быть некоторые проблемы с использованием таких моделей. Однако есть и несколько решений, с которыми вам следует познакомиться до попыток использовать модели других производителей:

- SPICE модель — это ASCII файл спецификации (netlist). Он совсем не содержит графической информации и не может быть непосредственно размещён на схеме. Это означает, что вам нужен элемент библиотеки ISIS для этого устройства, связанный с этой SPICE моделью. К сожалению, нет стандартного формата файлов для любой библиотечной части схемы, так что вам обычно следует нарисовать её самостоятельно.

- В netlist SPICE модель вызывается ссылкой на подсхему, используя X карту. Связывание узлов схемы с узлами модели определяется порядком, в котором узлы схемы задаются в X карте. Например,

  ```
  XU1 46 43 32
  ```

  означает, что узел 46 схемы соединён с узлом 1 модели. Аналогично узел 43 соединён с узлом 2, а узел 32 соединён с узлом 3. К сожалению, эта схема означает, что узлы модели не именованы. И что ещё хуже, крайне редко есть некая согласованность между номерами узлов модели и номерами физических выводов устройства — хотя бы то, что физические корпуса могут иметь не соединённые с кристаллом выводы, и это сложно оформить при нумерации узлов SPICE модели. Есть также ряд трудностей с многоэлементными устройствами, как TL074 — имеет ли модель один ОУ или четыре?

  На практике это означает, что нужны некоторого рода явные кросс-ссылки, чтобы сказать ISIS, какие номера узлов SPICE модели использовать для каждого вывода устройства, поскольку ни имена выводов ISIS, ни номера выводов не могут использоваться по причинам, указанным выше. Было введено специальное свойство устройства SPICEPINS, чтобы сделать это как можно более безболезненно.

- Существует множество вариантов SPICE, начиная со SPICE 2, для стандарта которого были написаны большинство появившихся моделей. Однако значительное число производителей объявило, что их модели имеют совместимость с PSPICE™, коммерческой версии SPICE 2, которая была разработана в значительной мере далеко от стандарта оригинала. Пока SPICE3F5 имеет много особенностей, не поддерживаемых PSPICE™, PSPICE ™ имеет ряд типов примитивов, которые различны, и также использует разный синтаксис для некоторых из нововведений, которые были добавлены в оба продукта. Реализация этого, конечно, означает, что модели специально написанные для PSPICE™ могут не работать с PROSPICE.

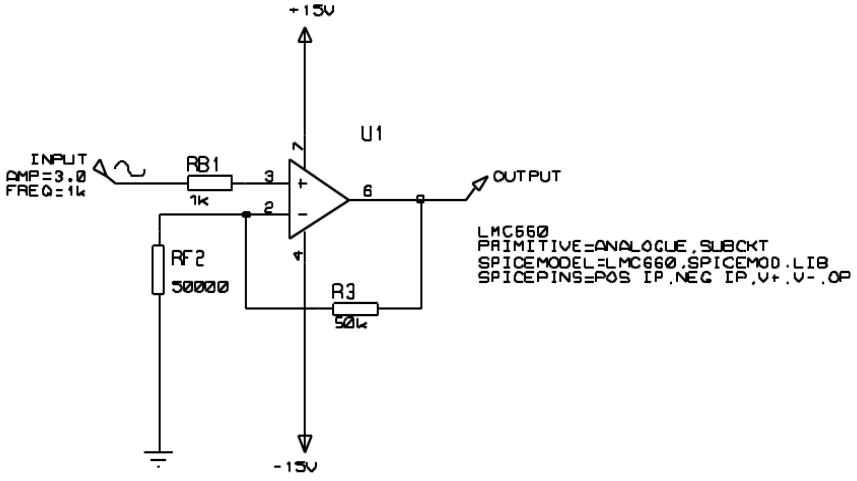
  Вкратце, если модель других производителей не работает правильно, первое, что следует проверить, это для какой версии, SPICE 2 или SPICE 3, она была написана.

Действительно хорошая новость — это то, что у нас собрано вместе большое количество (более 1500 к моменту написания этого руководства) моделей от производителей, и мы создали соответствующие элементы библиотеки ISIS для них.

Вы должны, конечно, понимать, что мы можем не согласиться нести ответственность за точность и даже функциональность этих моделей, и что и мы, и производители озабочены этим отказом от ответственности за потери любого рода возникающие при их использовании. В любом случае, никогда нет гарантий, что симуляция схемы будет в точности отражать работу реального устройства, и что всегда рекомендуется создать и протестировать физический прототип до начала массового производства.

ИСПОЛЬЗОВАНИЕ SPICE ПОДСХЕМ (SUBCKT ОПРЕДЕЛЕНИЕ)

Наш первый пример — это модель LMC660 операционного усилителя, которая есть в SPICE файле SPICEMOD.LIB. Этот пример можно найти также в проекте SPICE1.DSN в директории примеров, и он показан ниже.



Компонент операционный усилитель имеет три присвоенных свойства — PRIMITIVE, SPICEMODEL и SPICEPINS. Рассмотрим каждое из них по порядку:

PRIMITIVE=ANALOGUE,SUBCKT

Это задание всегда одинаково для компонентов, которые моделируются как SPICE подсхемы. Оно означает, что компонент будет симулироваться непосредственно SPICE3F5 и что он представлен подсхемой SPICE, а не действительным SPICE примитивом.

SPICEMODEL=LMC660,SPICEMOD.LIB

Эта строка показывает имя используемой подсхемы и имя ASCII файла, который содержит определение подсхемы. Альтернативно вы можете использовать

SPICEMODEL=LMC660

SPICEFILE=SPICEMOD.LIB

Некоторые производители добавляют много моделей в один файл, другие только одну в файле. Это ничего не значит. Имя подсхемы должно в точности совпадать с тем, что используется в файле модели, так что, если подсхема была названа как LMC660/NS, вы должны включить именно этот текст в свойство SPICEMODEL.

Файлы SPICE моделей ищутся в текущей директории и в *Module Path*, как задано в диалоговой форме *Set Paths*.

```
SPICEPINS=POS IP,NEG IP,V+,V-,OP
```

Свойство SPICEPINS задаётся с целью привязать имена выводов ISIS к номерам выводов SPICE модели. Чтобы понять, как это свойство используется, вам нужно взглянуть на комментарии в оригинальном файле SPICE модели:

```
*//////////////////////////////////////////////////
*LMC660AM/AI/C CMOS Quad OP-AMP MACRO-MODEL
*//////////////////////////////////////////////////
*
* connections:      non-inverting input
*                   |   inverting input
*                   |   |   positive power supply
*                   |   |   |   negative power supply
*                   |   |   |   |   output
*                   |   |   |   |   |
*                   |   |   |   |   |
.SUBCKT LMC660      1   2  99  50  41
*
```

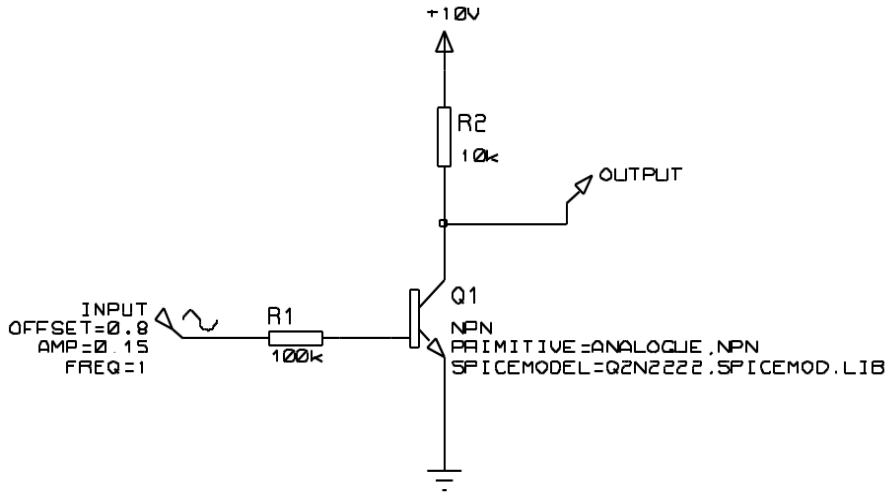
На «диаграмме» перечисляются функции выводов и их последовательность в строке SUBCKT. Существенно, что номера, присвоенные относительно внутренних определений модели, фактически не важны для наших целей. Ключевая информация — это то, что неинвертирующий вход первый вывод, инвертирующий вход второй и т.д. Этим определяется, как вы сконструируете значение свойства SPICEPINS, а именно в том, что вы дадите имена выводам ISIS в соответствующей последовательности. Имена выводов с пробелами допустимы, но будьте внимательны, чтобы не оставлять пробелов перед запятыми. Альтернативно, вы можете заключить каждое имя вывода в кавычки — это важно, если у вас есть имена выводов, которые содержат запятые.

Вам нужно, конечно, знать имена выводов ISIS, которые часто бывают скрытыми. Это решается перемещением мышки на конец вывода компонента, а затем следует прочитать строку состояния.

ИСПОЛЬЗОВАНИЕ SPICE МОДЕЛИ (КАРТА МОДЕЛИ)

Для полупроводниковых приборов, особенно диодов, транзисторов, JFET и MOSFET, SPICE модели обычно задаются в терминах единственной MODEL card (карты модели). Она перечисляет значения параметров для подходящего SPICE примитива. Процедура похожа на ту, что есть для SUBCKT модели, но в действительности несколько проще, поскольку не происходит преобразования имён выводов. Дополнительно разделы библиотеки ISIS уже содержат разные типы SPICE примитивов — вы найдёте их в ASIMMDLS.LIB.

Наш второй пример — это транзистор 2N2222, он вновь есть в ASCII файле SPICEMOD.LIB. Также можно найти пример SPICE2.DSN в директории *Samples*.



Транзистор имеет два присвоенных свойства — PRIMITIVE и SPICEMODEL.

Рассмотрим каждое из них последовательно:

```
PRIMITIVE=ANALOGUE,NPN
```

Это присваивание показывает, что устройство будет симулироваться непосредственно PROSPICE, как примитив типа NPN. Если вы выбрали диод, транзистор, JFET или MOSFET из библиотеки ASIMMDLS, он уже будет иметь подходящее присвоение PRIMITIVE.

```
SPICEMODEL=Q2N2222,SPICEMOD.LIB
```

Значение SPICEMODEL такое же, что и для SUBCKT модели. Оно задаёт имя используемой модели и SPICE netlist файл, который её содержит. Большинство производителей включают много определений MODEL в единственный файл.

Файлы SPICE model ищутся в текущей директории и через *Module Path*, как он задан в диалоговой форме *Set Paths*.

БИБЛИОТЕКИ SPICE МОДЕЛЕЙ

Библиотеки других производителей SPICE моделей, приходящие с PROSPICE, содержатся в двоичных библиотечных файлах, похожих на те, что есть для устройств ISIS и библиотек символов. Сделано это было из двух соображений:

- Многие файлы моделей исключительно малы и многочисленны, и на всё тратится огромное количество места на жёстком диске с большим размером кластера, если оно хранится индивидуально.
- Когда много моделей содержатся в единственном ASCII файле, PROSPICE вынужден проходить весь файл, чтобы использовать только одну модель, а это довольно медленно.

Когда SPICE модель содержится в библиотеке SPICE моделей (SML), синтаксис для её поддержки для раздела ISIS библиотеки сильно отличается от нужного. Например, LMC660 в первом примере будет иметь вместо нужных следующие свойства:

PRIMITIVE=ANALOGUE,SUBCKT

SPICEMODEL=LMC660

SPICEPINS=POS IP,NEG IP,V+,V-,OP

SPICELIB=NATSEMI

(как ранее)

(как ранее)

Свойство SPICELIB задаёт имя для файла SPICE модели библиотеки, которое содержит элемент. Эти файлы ищутся в текущей директории, и в *Module Path*, как задано в диалоговой форме *Set Paths*.

Библиотеками SPICE model можно манипулировать в командной строке инструментов PUTSPICE.EXE и GETSPICE.EXE, хотя мы не ожидаем, что обычные пользователи создадут или соберут достаточное количество моделей, чтобы обеспечить их использование. Запуск любой из этих программ без параметров даст информацию об их использовании.

**SPICE SCRIPTS*

Реально возможно ввести определение типа SPICE модели непосредственно в ISIS, а затем вызвать модель для подходящего компонента на схеме. Например, модель транзистора для 2N2222 должна быть вписана в ISIS скрипт следующим образом:

```
*SCRIPT SPICE
.MODEL Q2N2222 NPN(IS=3.108E-15 XTI=3 EG=1.11 VAF=131.5 BF=300
NE=1.541
+ISE=190.7E-15 IKF=1.296 XTB=1.5 BR=6.18 NC=2 ISC=0 IKR=0 RC=1
+CJC=14.57E-12 VJC=.75 MJC=.3333 FC=.5 CJE=26.08E-12 VJE=.75
+MJE=.3333 TR=51.35E-9 TF=451E-12 ITF=.1 VTF=10 XTF=2)
*ENDSCRIPT
```

Свойство транзистора SPICEMODEL будет затем изменено на

SPICEMODEL=Q2N2222

то есть, без задания имени файла.

Эта возможность может быть особенно полезна, когда модель получена на бумаге, или когда вы хотите интерактивно поэкспериментировать со значениями параметров.

ПОДДЕРЖКА АВАРИЙ ПРИ СИМУЛЯЦИИ МОДЕЛИ

Модели подсхем очень различаются по сложности и разработке. И как результат, одни модели симулировать легче, чем другие. В качестве основного правила — если PROSPICE не может симулировать вашу схему с её преобразованной моделью, тогда и ничто не сможет.

Мы часто находили, что проблемы в большинстве своём усугубляются плохим проектом схемы. Если у вас есть примечания в справочном листке к устройству, тогда сравните его с вашей схемой, чтобы убедиться, например, нет ли несоответствия во входном токе для входа в рабочей точке. Разработчики моделей всегда стремятся сделать модель работающей с

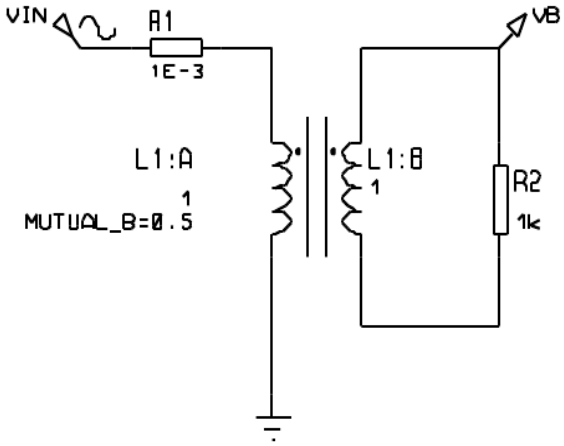
симулятором, что отмечают в примечаниях, а не в общем случае.

ProSPICE можно сделать более стабильным (но менее точным), если увеличить значение GMIN. По умолчанию это 1E-14, так что есть смысл попробовать1E-13 и 1E-12. При значениях около 1E-6, похоже, любые результаты симуляции совсем не будут иметь отношения к реальности, так что указывайте наинизшие из значений, но из возможных.

ШИНЫ ПИТАНИЯ И ЗЕМЛИ

Почему вам нужна земля

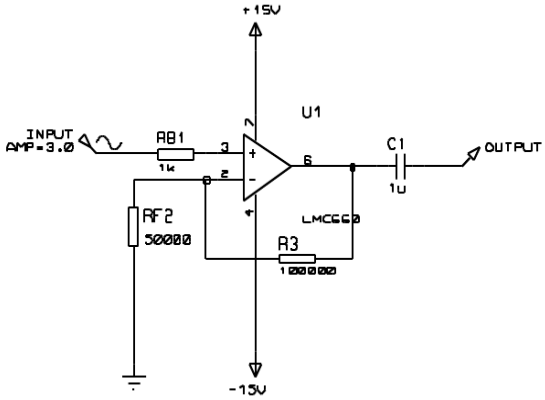
Все симуляции требуют, чтобы была задана земля, иначе размещение пробника на участке цепи не имеет смысла, так как напряжение на этом участке должно определяться в терминах опорной точки. Фактически есть ещё требование, чтобы все части схемы имели путь по постоянному току к земле, поскольку зондирование точки на «плавающем» участке цепи, бессмысленно. Например, на схеме ниже измерение напряжения в VB бессмысленно,



поскольку вторая сторона — плавающая. Теоретически, мы можем предложить двух выводные пробники (наподобие реального мультиметра), но возникают серьёзные математические трудности при расчёте схемы без земли, и, главное, Berkeley University не обращается к ним в SPICE3F5.

Ключевым, таким образом, будет то, что все секции вашей схемы должны иметь путь по постоянному току к земле. Хорошая новость в том, что ProSPICE проверяет это для вас и сообщит в предупреждении обо всех цепях, которые не отвечают этому критерию. В большинстве случаев симуляция на этом прекращается.

Другая конфигурация схемы, которая могла иметь трудности в прошлом, показана ниже:



Здесь наличие разделительного конденсатора C1 означает, что пробник на выходе не имеет пути для постоянного тока к земле. Следовательно, симулятор не может рассчитать рабочую точку для выхода, поскольку при вычислении рабочей точки все конденсаторы выбрасываются из цепи. Мы выбрали решение для этой проблемы в использовании конденсаторов с очень маленькой утечкой. Поэтому, в отсутствии любых других путей для постоянного тока, рабочая точка на выходе вычисляется с полностью разряженным C1.

Конденсаторы без утечки определяются свойством управления симуляцией GLEAK, которое имеет значение по умолчанию 1E-12Mho. Установка этого значения в нуль устраняет утечку конденсатора, как в традиционных SPICE симуляторах.

Исключая симуляцию внутренней схемы DRAM памяти, мы не видим каких-то проблем с этой схемой, и она предохраняет начинающих от множества странных сообщений об ошибках. В любом случае, реальные конденсаторы обычно имеют утечку значительно большую, чем миллион мегаом.

Цепь земли в схеме может быть определена либо явно, либо скрыто. Явная земля определяется размещением не именованного *Ground* контакта, как на нижнем конце RF2 на рисунке выше. Вы также можете обозначить провод текстом GND, если места мало.

Неявная земля может быть создана с помощью питающей шины, генератора с единственным выводом, нагрузочного пробника или цифрового выхода, как, впрочем, и использованием модели, которая имеет внутренний заземлённый узел.

Шины питания

PROSPICE распознаёт некоторое количество объектов, как шины питания (power rails); правила следующие:

- Любая сеть, названная GND или VSS, обосновывается как опорная земля. А насколько это затрагивает PROSPICE, GND и VSS означают то же самое.
- Цепи, названные VCC и VDD, считаются логическими шинами питания, и будут трактоваться симулятором DSIM как логическая «1». Как и с GND и VSS, PROTEUS VSM принимает, что эти два имени относятся к той же цепи, если вы действительно хотите отделить логическое питание, вы должны выбрать разные имена для цепей.
- Контакты питания с именами в виде +5, -5 или +10V, -10V принимаются как предопределённые фиксированные шины питания с опорой на землю. Символы «+» и «-» критичны, а этикетки, как 5.0V нет.

  Соединение двух таких этикеток с разными значениями с одной и той же цепью — это ошибка.

  Создание шины питания также неявно задаёт землю.

Дополнительная возможность, добавляемая как часть схемы для симуляции смешанного режима, такова — модели логических микросхем могут рассматриваться как уже включённые. Это позволяет вам рисовать схему, как показанная ниже, и получать ощутимые результаты без прорисовки явных цепей питания.

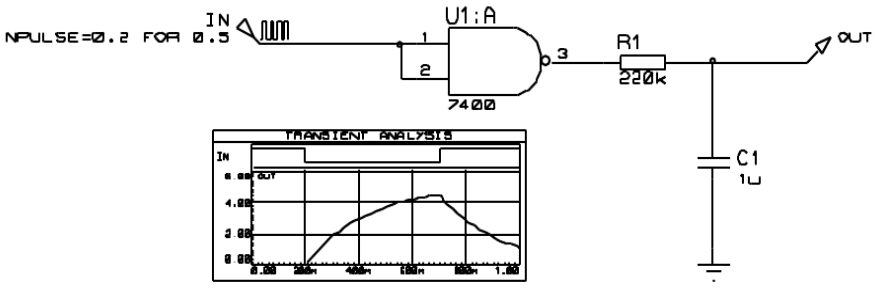


Схема работает, поскольку Interface Model, определённая для 7400, включает свойство VOLTAGE. Что подразумевает создание 5V батарейки между скрытыми выводами питания 7400 (VCC и GND), таким образом, цепь VCC приобретает потенциал 5V. Остальные элементы, соединённые с VCC, включая DAC объект на выходе 7400, будут видеть это напряжение.

Дальнейшее обсуждение Interface Models (модели интерфейса) вы найдёте в соответствующем разделе.

НАЧАЛЬНЫЕ УСЛОВИЯ

Обзор

Какой бы тип симуляции не был выбран, первое, что делает PROSPICE, это рассчитывает рабочую точку схемы — условия устойчивого состояния, предшествующие появлению сигналов на входе. Есть два отдельных режима операций, ассоциированных с расчётом этих значений:

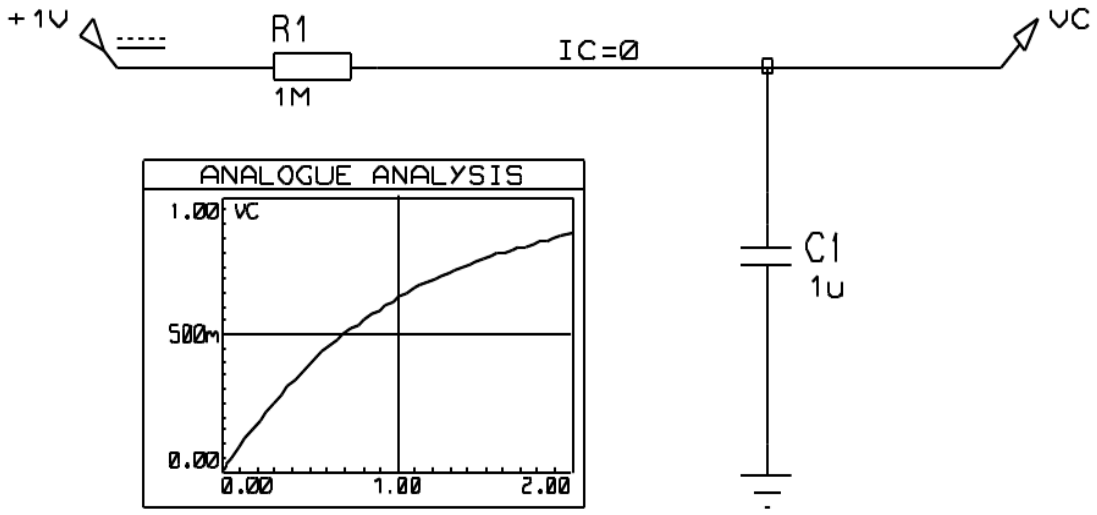
- При поиске рабочей точки считается, что все конденсаторы заряжены, а через индуктивности протекает ток. В этом случае Transient Analysis будет показывать поведение схемы, как если бы питание было подано непосредственно до времени начала симуляции, так что при старте симуляции всё уже успокоилось. Это режим операций по умолчанию, который даёт возможность установить начальные условия для отдельных компонентов и/или узловых напряжений, подходящие для многих целей.
- При поиске рабочей точки все конденсаторы считаются закороченными, а индуктивности оборванными. В этом случае Transient Analysis покажет поведение схемы таким, как если бы питание было подано в нулевой момент времени. Этот режим операций может быть выбран сбросом флажка *Compute Operating Point* на графике.

В любом случае возможно задать начальные условия для отдельных компонентов или узловых напряжений. Это особенно полезно для схем осцилляторов или таких, где работа схемы зависит от того, чтобы отдельные конденсаторы были разряжены до начала испытания. Действительно, концепция устойчивого состояния рабочей точки не имеет

смысла для осцилляторов, а симуляция может полностью остановиться, если некоторые начальные условия не заданы.

Задание начальных условий для цепей

Самый лёгкий способ задать начальные условия, чаще всего, обозначить начальное напряжение для отдельной цепи. На схеме ниже это выполнено добавлением этикеток к проводам с текстом IC=0 для испытуемой цепи. Без этого присваивания PROSPICE будет вычислять значение устойчивого состояния напряжения на C1, то есть, 1 вольт, и график будет показывать VC, как горизонтальную прямую линию.



Для цепей, которые имеют внутреннее соединение только цифровых компонентов, вы должны использовать логические состояния для начальных условий; то есть, 1,0,H,L,HIGH,LOW,SHI,WHI,SLO,WLO или FLT, и присваивать им BS свойство (Boot State, состояние загрузки). Для смешанных цепей вы должны задавать начальное напряжение — это будет автоматически распространяться как логический уровень на цифровые компоненты.

Задание начальных условий компонентов

Эта опция доступна только тогда, когда PROSPICE не вычисляет начальную рабочую точку, то есть, когда флажок Initial DC Solution на графике сброшен. В этом случае все узловые напряжения будут нулевыми в нулевой момент времени, исключая цепи с заданным свойством «начальные условия», как это описано выше. В этом случае можно задать начальные условия для отдельных компонентов. Например, вы можете добавить свойство

IC=1

для C1 в предыдущем примере, так что C1 начнёт работу со значением устойчивого состояния в 1 вольт.

Детали свойства IC поддерживаются разными SPICE примитивами, и даны в разделе о моделировании.

Некоторым образом неприятно, что эта опция не доступна при вычислении рабочей точки, но это то, как SPICE3F5 было кодировано в Berkeley. Однако мы добавили свойство

PRECHARGE в качестве лекарства от этой болезни.

Свойство NS (NODESET, установка узла)

Иногда при отказе симуляции на стадии поиска рабочей точки может быть полезно дать SPICE «подсказку» в виде начальных значений для отдельных цепей. Это отличается от установки начальных условий в том, что значения даются только для использования в первой итерации, а затем цепь становится «плавающей» до получения значения, к которому сходятся уравнения матриц. Это не более, чем помощь в сходимости, и не скажется на определении реальной рабочей точки.

Такая подсказка для сходимости может быть задана при использовании свойства цепи NS, так что размещая этикетку провода с текстом NS=10, вы задаёте начальное значение в 10 вольт для этой цепи.

Свойство PRECHARGE

Ещё одна опция для задания начального условия — это свойство PRECHARGE (предзаряд). Оно может присваиваться любому конденсатору или индуктивности в цепи и задавать либо напряжение на элементе, либо ток, протекающий через него, соответственно.

Свойство PRECHARGE — это специфическое добавление Labcenter к SPICE, и отличается от свойства IC в том, что оно применимо в зависимости от того, установлен ли флажок *Initial DC Solution*.

МОДЕЛИРОВАНИЕ ТЕМПЕРАТУРЫ

Ядро SPICE3F5 предлагает широкую поддержку для моделирования температурных эффектов. Схема работает следующим образом:

- Есть глобальное свойство TEMP, которое, если ничего другого не сделано, будет отнесено ко всем компонентам схемы.
- Температура индивидуальных компонентов может быть задана через их собственное свойство TEMP.
- Когда параметры модели устройства меняются под действием температуры, отличной от той, при которых они были измерены, эта температура может быть задана глобально и/или индивидуально через свойство TNOM.

Моделирование температуры поддерживается у резисторов, диодов, JFET, MESFET, BJT (полевые и биполярные транзисторы), и уровня 1, 2 и 3 MOSFET. BSIM модели (ожидаемо) были созданы для заданной температуры. В цифровые примитивы температурная зависимость не встраивалась.

Дополнительно, очень важно уяснить, что большинство эквивалентных моделей схем IC не имеют моделей корректных температурных эффектов. Причина этого в том, что эти модели используют идеальные управляемые источники и другие макро-модельные примитивы, а не построены из точных копий внутренних компонентов. А когда такие примитивы используются, маловероятно, что модели покажут корректную температурную зависимость поведения, пока кто-то не возьмёт на себя труд сделать это.

ПАРАДИГМА ЦИФРОВОЙ СИМУЛЯЦИИ

Модель девяти состояний

Вы можете думать, что цифровая симуляция будет моделировать только высокое и низкое состояние, но, фактически, DSIM моделирует всего девять отдельных состояний:

Тип состояния	Ключевое слово	Описание
Питающее высокое	**PHI**	Питающая шина логической 1.
Сильное высокое	**SHI**	Активный выход логической 1.
Слабое высокое	**WHI**	Пассивный выход логической 1.
Плавающее	**FLT**	Плавающий выход — высокий импеданс.
Неопределённое	**WUD**	Среднее напряжение от аналогового источника.
Спорное	**CON**	Среднее напряжение от цифрового конфликта.
Слабое низкое	**WLO**	Пассивный выход логического 0.
Сильное низкое	**SLO**	Активный выход логического 0.
Питающее низкое	**PLO**	Питающая шина логического 0.

В сущности, данные состояния содержат информацию об их полярности: high, low или mid-way, — и их силе. Сила измеряется величиной выходного тока как источника или для потребителя, и становится существенной, когда два или более выхода соединяются с той же цепью.

Например, если выход с открытым коллектором присоединяется через резистор к VCC, тогда при переходе выхода в низкое состояние и Слабое высокое (Weak High), и сильное низкое (Strong Low) состояния применимы к цепи. Сильное низкое состояние победит, и цепь перейдёт в низкое состояние. С другой стороны, если два выхода с трёхстабильным состоянием становятся активы в цепи, а переходят в противоположные состояния, ни один из выходов не «побеждает», а результатом будет состояние Спорное (Contention).

Эта схема позволяет DSIM симулировать выходные цепи с открытым коллектором или открытым эмиттером и подтягивающими резисторами, а также цепи, в которых трёхстабильные выходы противостоят друг другу через резисторы — разновидность мультиплексора бедняков, если вам так нравится. Однако важно помнить, что DSIM — это только цифровой симулятор и не может моделировать поведение, которое становится бесспорно аналоговым. Например, подключение избыточно большого резистора к TTL входам будет работать хорошо в DSIM, но откажет на практике, из-за недостаточного тока подтяжки от входов.

Неопределённое состояние

Когда вход цифровой модели не определён, это распространяется через модель согласно тому, что может быть описано как правила здравого смысла. Например, если вентиль И имеет

низкий уровень, тогда выход будет в низком состоянии, но если все, кроме одного, входы в высоком состоянии, и этот вход не определён, тогда выход будет неопределённым.

- Фронт переключающегося устройства требует перехода от положительно определённого логического 0 к логической 1 (или наоборот), чтобы фронт был обнаружен. Переход от 0 к 1 не определён и не возвращается для вычисления фронта.
- Более сложные последовательные логические устройства (счётчики, защёлки и т.п.) будут часто иметь не определённые входы ни к логическому 0, ни к логической 1, согласно устройству их внутренней логики. Это не похоже на реальную жизнь!

Поведение плавающих входов

Обычная практика, если нет иного опыта, следует доверять тому факту, что свободные TTL входы ведут себя так, как если бы были подключены к логической 1. Эта ситуация может возникнуть и как результат пропуска соединения, и в том случае, когда вход присоединён к неактивному выходу с тремя состояниями. DSIM должен что-то предпринять в подобной ситуации, поскольку внутренние модели принимают истинное логическое поведение со входами, находящимися исключительно в высоком или низком состоянии.

Это учитывается через свойство FLOAT. Оно может присваиваться либо непосредственно компоненту, либо Interface Model. В частности, модель интерфейса (interface model) для элементов TTL имеет присваивание:

```
FLOAT=HIGH
```

которое гласит, плавающий вход должен интерпретироваться как уровень логической 1.

Чтобы задать, что плавающие входы должны быть логическим 0, используйте

```
FLOAT=LOW
```

Иначе плавающий вход будет считаться в неопределённом (Undefined) состоянии (см. выше).

Поддержка проблем

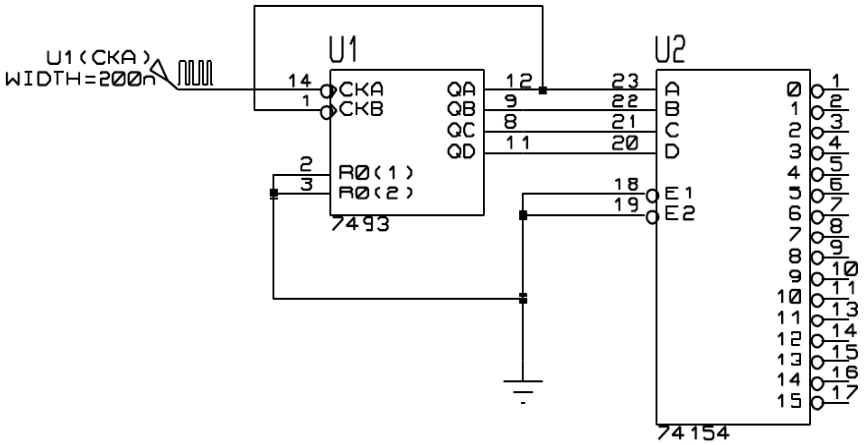
При разработке DSIM мы долго обсуждали, как поддерживать симуляцию моделей, подверженных очень коротким импульсам. Фундаментальная проблема в том, что при этих условиях основное положение парадигмы DSIM — модели ведут себя как чисто цифровые — начинает проваливаться. Например, реальная модель 7400 под действием входного импульса в 5ns генерирует на выходе какого-то рода импульс, но не тот, который встречается в спецификации логических уровней для TTL. Будет ли такой выходной импульс приводить в действие счётчик, это зависит от очень многих аналоговых явлений.

Самое лучшее, что можно предложить для учёта крайностей, это:

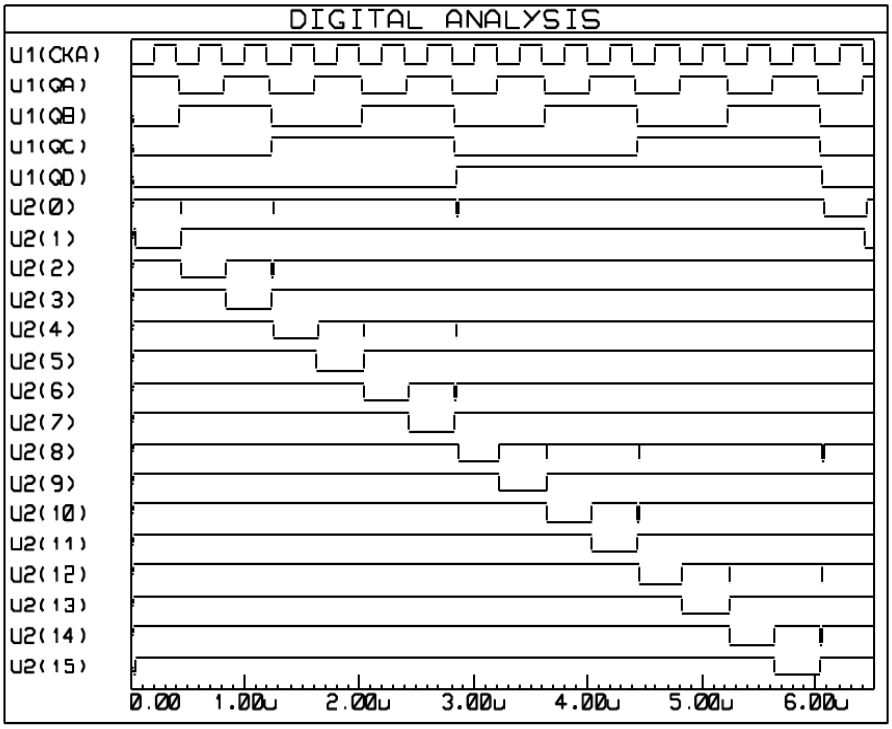
- Входной импульс в 1ns не передавать совсем.
- 20ns импульс передавать хорошо.

Где-то между ними вентиль перестанет передавать импульсы правильно и, скажем, будет сдерживать проблему. Это приводит нас к концепции *Glitch Threshold Time* (пороговое время проблемы), которая может быть дополнительным свойством модели наряду с обычными TDLH и TDHL.

Другая тонкая штука — будет ли проблема касаться входа или выхода модели. Чтобы как-то решить это, обратимся к дешифратору 4-16 счётчика сквозного переноса, как показано ниже.



Выходы счётчика сквозного переноса «расположены уступами», следовательно, возникает возможность того, что дешифратор будет генерировать фальшивые импульсы, когда входы проходят через промежуточные состояния. Эта ситуация показана на следующей диаграмме:



The above graph was produced with TGQ=0 for the 74154

Рассмотрим первую проблему примера на событии U1(QA); спад вначале побеждает подъём U1(QB) и промежуточное состояние входа проходит в дешифратор, примерно, в течение 10ns. Вопрос в том, что может ли дешифратор действительно отреагировать на это или нет, и даже более того, что случится, если колебания на входе продлятся только 1ns или 1ps? Ясно, в последних двух случаях реальное устройство не будет реагировать, и это говорит нам, что мы должны поддерживать проблему на выходе, а не на входе, поскольку в примере выше

входные импульсы относительно долгие и не будут рассматриваться проблемными при любых разумных критериях. Некоторые конкурирующие продукты могут внести в это путаницу, можно прогнозировать отклик даже в ситуации с 1ps!

Действительно интересная часть этой истории в том, что если вы соберёте схему, приведённую выше, она, возможно, не будет иметь проблем. Это очень плохая, естественно, разработка, но TDLH и TDHL модели '154 приблизительно 22ns, и это делает условия отклика на входное воздействие в 10ns порядком выше её возможностей. С индивидуальными компонентами мы пробовали, нет выходных импульсов, иных чем, возможно, лёгкие «просадки» питающего напряжения, насколько это измеримо.

Для поддержания управления поддержкой проблем все DSIM примитивы предлагается использовать с определяемым свойством *Glitch Threshold Time*, названным *TGxx*, где xx — это имя соответствующего выхода. Наши TTL модели определены так, чтобы эти свойства могли быть отменены для TTL компонентов, а значения затем переопределены с тем, чтобы *Glitch Threshold Times* усреднялось по основным задержкам распространения «low-high и high-low». Задание нулевого *Glitch Threshold Times* позволит выявить все проблемы, если вам удобнее такое поведение. График, показанный выше, был создан именно так, присвоением `TGQ=0` модели 74154.

И, наконец, важно уяснить, что если *Glitch Threshold Time* больше, чем либо low-high, либо high-low задержка распространения, тогда *Glitch Threshold Time* будет игнорироваться. Это для того, чтобы после входного фронта по истечении значимого времени задержки выход вентиля изменил своё значение — он не может «заглянуть» в будущее и увидеть, произойдёт ли событие на другом входе, которое может сбросить выход. Рассмотрите симметричный вентиль с временем задержки распространения в10ns и *Glitch Threshold Time* равным 20ns. В момент t=0ns вход переходит в высокое состояние, а при t=15ns переходит в низкое. Вы можете исключить это из распространения, переведя выход в высокое состояние при t=10ns и переходом в низкое при t=25ns, так что произведённый импульс будет 15ns шириной, и был бы подавлен, поскольку меньше, чем *Glitch Threshold Time*. Причина, по которой этого не произойдёт, в том, что при t=10ns выход должен перейти в высокое состояние, и он не может стать низким в ближайшие 20ns без шансов (как в нашем примере), чтобы второй фронт, приходящий при этом, произвёл выходной импульс, он должен быть подавлен! Когда же выход становится высоким при t=10ns, тогда второй фронт (при t=25ns) свободно может сбросить его. Вы должны хорошо это всё обдумать, чтобы понять это.

МОДЕЛИ ИНТЕРФЕЙСОВ СМЕШАННОГО РЕЖИМА (ITFMOD)

Обзор

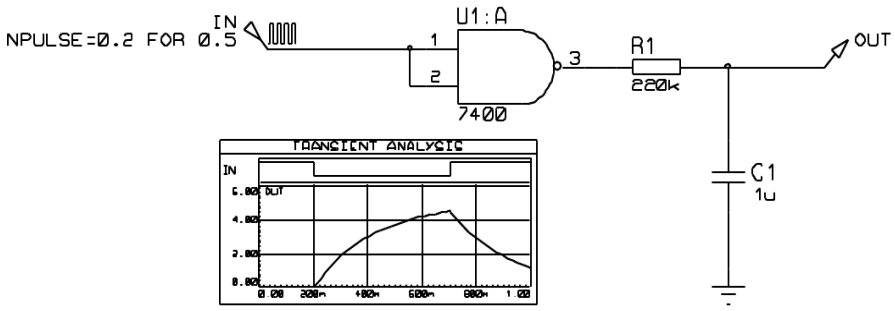
При разработке нашей схемы для симуляции смешанного режима в PROSPICE мы уделили много внимания проблеме того, как задать аналоговые характеристики устройствам цифровых семейств. Характеристики включают:

- Входной и выходной импеданс устройства.
- Логические пороги для входов устройств.
- Уровни напряжения для высокого и низкого состояний выхода.

- Времена подъёма и спада для выходов устройств.
- Определённое логическое состояние «плавающих» входов.

Схема, которая учитывает задание всех этих параметров для каждого из устройств в библиотеке TTL, скажем, была бы крайне громоздка.

Вдобавок, возникает важная проблема (для начинающих, во всяком случае) в задании питающего напряжения — есть тенденция выкладывать схемы, подобные той, что ниже, и ожидать разумных результатов. Здесь проблема, конечно, в неявном соглашении, что 7400 имеет 5В питающее напряжение, подводимое к его скрытым выводам, которые соединены с VCC/GND.



Все эти проблемы решаются введением в свойство компонента ITFMOD. Это очень похоже на свойство MODEL в том, что поддерживает ссылку на набор значений свойств, но также активирует специальный механизм в компиляторе netlist. По существу это работает следующим образом:

- Для любого устройства, которое имеет свойство ITFMOD, дополнительное определение модели вызывается в процессе создания netlist, которое задаёт управляемые параметры для ADC и DAC объектов, и также имена положительного и отрицательного выводов подключения питания. В схеме выше U1:A будет иметь ITFMOD=TTL.
- Получив имена выводов питания (VCC, GND в данном случае), ISIS создаёт специальный примитив и соединяет его с выводами питания. ISIS имена этого объекта похожи на те, что у объектов производных листов или моделей, так что схеме выше питание объекта будет вызвано U1:A_#P.
- Когда PROSPICE симулирует схему смешанного режима, он создаёт ADC и DAC объекты и подразумевает, что они «принадлежат» объектам, к которым они подключены. В случае вышеприведённой схемы DAC объект будет создан с именем U1:A_DAC#0000, поскольку он формирует интерфейс для выхода U1:A.

  Смысл в том, что при выполнении этого также просматриваются объекты интерфейса питания с тем же основанием имени, то есть, U1:A, и ищется U1:A_#P. Затем приходит инструкция U1:A_DAC#0000 — взять свойство из U1:A_#P, которое в свою очередь наследует свойства от заданной модели в оригинальном присваивании ITFMOD. Так объект DAC оперирует с параметрами, определёнными для семейства TTL логики.

- Каждый объект интерфейса питания также содержит батарею, которая назначается свойству VOLTAGE, данному в определении модели интерфейса. Модель TTL интерфейса имеет заданное VOLTAGE=5V.

  Это означает, что в вышеприведённой схеме батарея 5V включена между VCC и GND, поскольку это цепь, обозначенная как выводы питания для устройства 7400.

- Батареи имеют маленький внутренний импеданс (1 миллиом). А это означает, что если вы назначаете реальные шины питания VCC/VDD (разместив контакты питания или источник напряжения), тогда этим будет отменён уровень, определённый внутренними батареями — в мире симуляции большой ток через батареи ничего не значит!

Использование свойств ITFMOD

Существующие модели интерфейсов определены следующим образом:

TTL	Стандартная TTL серия (74 серия)
TTLLS	Низковольтная Шотки TTL (74LS серия)
TTLS	Стандартная Шотки TTL (74S серия)
TTLHC	Высокоскоростная CMOS TTL (74HC серия)
TTLHCT	Высокоскоростная CMOS TTL с TTL выходами (74HCT)
CMOS	4000 серия CMOS
NMOS	Микропроцессорного типа MOS схемы
PLD	PLD типа MOS схемы

Из этого следует, что любая новая цифровая модель может быть назначена фамилии устройств добавлением свойства, такого как

```
ITFMOD=TTL
```

Определения фамилии поддерживаются файлом ITFMOD.MDF, который хранится в директории моделей.

Каждое определение может содержать любое или все свойства, определённые для интерфейсов примитивов ADC и DAC. Дополнительно может быть задано следующее:

V+	-	Имя вывода плюса питания.
V-	-	Имя вывода минуса питания.
VOLTAGE	5V	Заданное по умолчанию рабочее напряжение.
RINT	1mΩ	Заданный импеданс внутренней батареи. Нулевое значение отключает батарею.
FLOAT	-	Задаёт HIGH или LOW значение для плавающего входа.

И, наконец, важно уточнить, что любое заданное свойство, например, TRISE, может быть отменено в родительском устройстве, так что, если вы хотите симулировать микросхему 4000 серии с большим временем переднего фронта, вы должны добавить TRISE=10u непосредственно в её список свойств.

ПОСТОЯННЫЕ ДАННЫЕ МОДЕЛИ

Некоторые модели симулятора, частично те, что для EPROMS и микропроцессоров, содержащие EEPROM или флэш память, способны «запомнить» данные между запусками симуляции, подобно их реальным прототипам. Это действие облегчается свойством MODDATA. Постоянные блоки данных модели сохраняются в файле DSN, и не удерживаются между сессиями PROTEUS, пока вы не сохраните проект.

Чтобы сбросить постоянные данные модели до начального состояния, используйте команду *Reset Persistent Model Data* из раздела *Debug* основного меню.

ЗАПИСЬ И РАЗБИЕНИЕ

Обзор

Уникальная особенность системы VSM в её способности делить большой проект на одну или более секций или частей и симулировать каждую из них индивидуально.

Есть два принципиальных преимущества, связанных с этим:

- В полностью интегрированных CAD схема всего проекта будет содержать несколько секций, которые вам не захочется или вы не сможете симулировать. Для предотвращения необходимости дробить проект требуется некий механизм для определения, какие компоненты в проекте действительно влияют на данный эксперимент при симуляции.

  ISIS делает это с учётом точек измерения и, в дальнейшем отступая от проекта, точек, управляемых тестовыми источниками и/или шинами питания.

- Если проект состоит из нескольких этапов, будет общим требованием видеть, как более поздние этапы выполняются, когда управляются выходами ранних этапов. Хотя это может быть получено симуляцией всех этапов вместе, но будет осуществляться неоправданно медленно.

  Разбиение позволяет результаты симуляции предыдущих этапов записать на «плёнку» (tapes) и воспроизвести как входной сигнал на следующих этапах.

  ISIS включает логику, чтобы сделать это либо с ручным управлением, либо автоматически. В последнем случае ISIS определяет, когда схема с заданным разбиением изменилась и новой симуляции повергается только то, что изменилось, или то, на чём скажутся произведённые изменения.

Операции с единственной частью

Многие эксперименты при симуляции будут состоять из тестирования единственной части проекта, изолированной от остального. Это обычно требует сигнала или сигналов на входе (или входах) этой секции, и затем наблюдения за происходящим в разных её точках.

Должно быть ясно, что сигналы могут добавляться размещением генераторов на подходящих входных проводах, и что работа схемы может наблюдаться путём размещения пробников. Однако эти действия сами по себе не изолируют тестируемую секцию от остальной части проекта.

Чтобы сделать это, вам нужно установить флажки *Isolate Before* на генераторах и флажки *Isolate After* на пробниках. Когда это сделано, только секция между точками изоляции будет компилирована в netlist и будет симулироваться.

Чтобы выполнить симуляцию секции, ISIS просматривает пробники, которые связаны с графиком, и проходит по компонентам и проводам (включая контакты внутренних соединений) наружу, пока не встретит одно из следующих условий:

- Больше нет компонентов для процесса.
- Достигнуты изолированные пробники или генераторы.
- Достигнуты записывающих устройства (tape) на входе или выходе.
- Достигнуты шины питания. Цепь превращается в питающую, если присоединена к контактам POWER или GROUND. GND или VCC этикетки на проводах НЕ делают этого.

Объекты записи

Объекты записи (Tape objects) имеют два отдельных применения в системе:

- Чтобы определить точки, в которых есть смысл игнорировать схему правее, когда схема левее симулируется. Такие точки обычно там, где низкий импеданс выходов управляет высоким импедансом входов. Это может быть также достигнуто изоляцией пробников.
- Чтобы поддержать средства для захвата состояния выходов одного из этапов проекта и использовать эти состояния для управления следующим этапом без предварительной симуляции первого этапа.

Чтобы разместить запись (Tape):

1. Выберите иконку *Tape Recorder Mode*.
2. Используйте иконки поворота и отражения для придания нужного положения записывающему устройству.
3. Переместите мышку в окно *редактирования* и нажмите левую клавишу мышки, перетащите записывающее устройство в нужное место и нажмите клавишу ещё раз.

   Вы можете разместить записывающее устройство (tape) непосредственно на

существующем проводе, разместив его так, чтобы точка соединения касалась провода. В качестве альтернативы вы можете разместить несколько записывающих устройств на свободном месте и соединить со схемой позже.

Жизненно важно размещать записывающее устройство в чувствительных местах, там где низкий импеданс управляет высоким. Если вы разместите записывающее устройство в другом месте, вы можете изменить свойства вашей схемы и испортить результаты, которые будут произведены.

Режимы записи

Чтобы дать максимальные возможности для пользовательского управления, записывающее устройство имеет три режима: AUTO, PLAY и RECORD. В последующем рассказе мы используем термины left и righ в смысле логической зависимости от структуры разбиения, чтения проекта от входа к выходу, слева-направо.

Режим AUTO

Это схема по умолчанию и определяет режим операций, при котором ISIS решает, какие части следует пересимулировать, а для каких можно использовать прежде полученные данные. Для большинства симуляций, которые требуют записи, режим AUTO оказывается наиболее полезен.

Это автоматическое определение основано на учёте всего текста, обнаруженного в sub-netlist, относящегося к секции проекта. Из чего следует, что если любые имена частей, значения, свойства, соединения и т.п. в этой секции менялись, будет выполнена пересимуляция, если только файл раздела уже не содержит этот набор информации.

Заметьте, что ВСЕ модели, скрипты, глобальные свойства проекта и т.д. включены в sub-netlist (под-спецификацию), так что изменение любого такого объекта вызовет пересимуляцию всех автоматически управляемых секций. ISIS не задаётся вопросом, будет ли отдельная модель, скрипт и т.п. использоваться компонентами в отдельной секции. Если вам нужно обойти это, вы можете использовать ручное управление в режимах RECORD и PLAY.

Заметьте, что объект TAPE в автоматическом режиме будет удалён из схемы, если выполняется интерактивная симуляция.

Режим PLAY

Этот режим даёт вам возможность проиграть именованный файл, который вы прежде записали либо с помощью записывающего устройства, либо добавлением свойства RECORD пробнику. Имя файла данных для проигрывания должно быть введено в поле *Filename* записывающего устройства; вы не можете использовать режим PLAY пока не введёте имя файла.

Когда записывающее устройство в режиме PLAY, схема слева отключена и игнорируется, если только не включает пробники, которые также включены в текущий график.

Не забывайте, что вы можете проигрывать только данные, которые были записаны для тех типов анализа, которые выполняются в данный момент. Попытка выполнения других типов вызовет ошибку.

Режим RECORD

Этот режим приведёт к тому, что данные, присутствующие на входе записывающего устройства, будут записаны в файле с именем в поле *Filename* устройства; вы не можете использовать режим RECORD, пока имя файла не введено.

Другой эффект режима записи в форсировании пересимуляции секции схемы слева от записывающего устройства, независимо от того, будет ли она меняться или нет. Если есть часть схемы справа от устройства записи, которая тестируется, тогда она тоже будет пересимулирована, поскольку она зависит от той части, что слева.

Режим RECORD записывающего устройства наиболее полезен при симуляции, в которой вы хотите записать сигнал, а затем использовать его как входной в дальнейших экспериментах.

СВОЙСТВА УПРАВЛЕНИЯ СИМУЛЯТОРОМ

Обзор

Есть огромное количество параметров, которые сказываются на деталях выполнения симуляции. Это включает такие элементы, как максимальное количество итераций, разрешённых для нахождения рабочей точки, точность, которая определяет момент сходимости, используемый метод интеграции и т.д.

Эти опции, общие для всех типов анализа, могут настраиваться отдельно для каждого графика с помощью его редактирования и выбора кнопки **SPICE Options**.

Свойства точности (Tolerance)

Эта группа параметров определяет, как точно SPICE будет вычислять решение. Чем выше точность, тем, обычно, длительнее время симуляции, а в некоторых обстоятельствах схема может вовсе не симулироваться из-за расходимости, если вы выбрали слишком высокий набор параметров точности.

Наиболее полезное значение здесь — это *Truncation Error Estimation* фактор, или TRTOL в традиционной номенклатуре SPICE. Если вы получили результаты, которые излишне «изрезаны» или страдают отклонениями, вы должны попробовать уменьшить это значение.

Значение минимальной электропроводности, GMIN, определяет утечку обратно включённых полупроводниковых переходов или других теоретических точек бесконечного импеданса. Вырезание этого значения может помочь выполнить сходимость для цепи, которая не смогла симулироваться, хотя это может уменьшить точность симуляции. См. раздел далее, где больше сказано о проблемах сходимости.

Утечка проводимости, GLEAK, похожа на то, что мы описывали, когда говорили о решении для схем с блокированием конденсаторов по постоянному току. Она определяет утечку по постоянному току конденсаторов и может, обычно, оставаться в стороне, пока вы не симулируете что-то похожее на ячейки CMOS памяти или нечто в этом роде.

Свойства Mosfet

SPICE симуляция MOSFET (канальный полевой униполярный МОП-транзистор) основана на допущении, что вы выполняете IC проект и, следовательно, реализуете схему для

масштабируемой геометрии. На практике это означает, что есть некоторое количество параметров, которые определяют предопределённые физические размеры элементов устройства MOSFET. Это значения, определяемые здесь.

Дополнительно были созданы некоторые модели, которые зависят от поведения старых версий SPICE, и это поведение MOSFET можно включать и выключать здесь, если вы используете старые MOSFET модели.

Свойства итерации

Свойства на закладке *Iteration* определяют, как SPICE обходится с цепями, которые плохо сходятся.

Метод интеграции может быть либо *Gear*, либо *Trapezoidal*. Последний поддерживается в основном для обратной совместимости с предыдущими версиями SPICE, хотя метод интеграции *Gear* обычно даёт более точные результаты для заданного количества временных шагов. При интеграции по Gear возможны порядки более второго; это заставляет SPICE использовать больше истории проходов точки времени, чтобы предсказать, что случится в следующей временной точке.

Когда SPICE терпит неудачу при поиске сходящегося решения для рабочей точки, программа делает попытки в двух приближениях: Gmin Stepping и Source Stepping. Количество шагов попыток в каждом методе может быть задано здесь.

Три следующих опции определяют, какое максимальное количество итераций будет использовано для каждой из рабочих точек: шагов в *Transfer* анализе, временных точек в *Transient* анализе. Увеличение этого может помочь в получении результата для сложных или близких к нестабильности схем.

Наконец, две опции дают возможное ускорение симуляций. LTRA уплотняет использование только для схем, использующих линии передач с потерями (LOSSYLINE модель). Идея в том, что близкие к одинаковым значения в данных канала отбрасываются так, что эти точки данных проходятся. Прохождение не изменяющихся элементов в общем оптимизируется, что предохраняет SPICE от пересчёта значений полупроводниковых устройств, чьи узловые напряжения не менялись с последней оценки.

Температурные свойства

Есть два глобальных температурных свойства: TEMP — предопределённая рабочая температура, и TNOM — параметр измеряемой температуры. TEMP определяет актуальную температуру схемы, тогда как TNOM — это значение, при котором зависящие от температуры параметры устройства берутся для выполнения измерения. Более детальное обсуждение моделирования температуры в PROSPICE смотрите в соответствующем разделе.

Свойства цифрового симулятора

TDSCALE, TDSEED, TDLOWER и TDUPPER

Переменная TDSCALE используется для управления масштабированием всех временных свойств, используемых моделями при запуске симуляции, если модель не была явно

отмечена, как не масштабируемая. Переменной TDSCALE может быть присвоено либо значение постоянной с плавающей точкой, либо ключевое слово RANDOM.

Если задано постоянное значение, тогда все временные свойства, определяемые для моделей, используемых при симуляции, умножаются на это значение; значение меньше, чем 1.0 уменьшают временные свойства, а значения больше 1.0 увеличивают их. Например, присвоение переменной:

```
TDSCALE = 1.1
```

даёт эффект удлинения всех временных свойств, используемых при симуляции на 10%. Значение по умолчанию для TDSCALE — это постоянное значение 1.0; это сказывается на всех временных свойствах так, что они не модифицируются.

Если переменной TDSCALE присвоить ключевое слово RANDOM, DSIM будет случайным образом масштабировать каждое временное свойство, умножая его на случайное значение с плавающей точкой. Диапазон значений временного масштабирования, выбираемый симулятором, может быть ограничен присвоением переменным TDLOWER и TDUPPER; этим определяются допустимые наинизшее и наивысшее случайные значения соответственно. По умолчанию значения TDLOWER и TDUPPER 0.9 и 1.1, что ограничивает случайное масштабирование значениями ±10%.

Последовательность генерируемых случайных значений, скажем, псевдослучайная — каждое последующее сгенерированное значение, которое должно быть случайным, фактически определяется предыдущим значением (и сложной формулой). Из этого следует, что любая последовательность случайных чисел определяется начальным значением, и что для заданного начального значения генерируемые последовательности случайных чисел всегда одинаковы. Свойство TDSEED позволяет вам задать начальное значение для генерации случайных значений временного масштабирования и гарантирует, что последовательность, выбранная для одной симуляции, при другом запуске будет всегда той же. Это устраняет проблему ошибок с временами в проекте, обусловленных случайными задержками распространения, которые появляются, но исчезают при последующих запусках симуляции.

Переменная TDSEED может получать только положительные целые значения в диапазоне 1-32767. Предопределённое значение само по себе случайное число (основанное на дате и времени), и этим поддерживается метод генерации случайных чисел от одной симуляции к следующей.

Например, присвоение:

```
TDSCALE = RANDOM
TDLOWER = 2.00
TDUPPER = 3.00
TDSEED = 723
```

даст эффект случайного увеличения всех временных свойств примерно на 200-300% со случайной последовательностью значений масштабирования. Начальное значение 723 приведёт к тому, что та же последовательность псевдослучайных чисел будет генерироваться при каждом запуске симуляции.

INITSEED

Свойство INITSEED используется для начальной случайной инициализации значения генератора, используемого моделями примитивов DSIM, чьё свойство инициализации было присвоено ключевому слову RANDOM.

Как и со свойствами TDSCALE и TDSEED, описанными выше, если последовательность значений, сгенерированная случайным значением инициализации, случайна, последовательность в целом конечна и определена. Свойство INITSEED поддерживает метод выбора, которым последовательность случайных значений используется симулятором DSIM.

Свойство INITSEED может принимать только положительные целые значения в диапазоне 1-32767. Предопределённое значение для INITSEED само по себе случайное число (основанное на дате и времени), и этим поддерживается метод генерации случайных чисел от одной симуляции к следующей.

ТИПЫ МОДЕЛЕЙ СИМУЛЯТОРА

Как сказать, имеет ли компонент модель

PROTEUS приходит с более, чем 8000 элементами библиотеки, из которых около 6000 имеют модели симулирования. Устройства, которые не имеют моделей, очень существенны для использования при разводке печатной платы (PCB design), и идея, что каждый элемент должен иметь модель, абсолютно несостоятельна. Это предполагало бы, между прочим, что мы должны создать модель для 68020 процессора — задача совсем не тривиальная.

Итак, для целей симуляции схем, вам нужно определиться, имеет ли компонент, который вы используете, модель симулирования. Иначе попытка симуляции может окончиться неудачей, если модели нет, и обычно с сообщением похожим на следующее:

```
ERROR [PSM] : No model specified for 'U1'.
```

Ошибка обнаруживается на этапе разбиения (PSM), поскольку до этого может получиться так, что компонент без модели окажется вне части схемы, которая должна симулироваться.

Иногда вы получите:

```
ERROR [U1] :
Value '74F00' of VALUE not found in parameter mapping table.
```

Это означает, что есть файл модели для устройства, но вы должны изменить значение типа элемента, который не моделируется MDF файлом. В примере выше мы должны изменить значение вентиля на 74LS00, который моделируется, для вентиля 74F00, который не моделируется.

Тип модели симулятора (если есть), доступный для компонента, отображается слева вверху *окна предварительного просмотра* в обозревателе устройств библиотеки. Например, если вы открываете обозреватель библиотеки и выбираете устройство 74LS00 в библиотеке 74LS, вы увидите

```
Schematic Model [74NAND.MDF]
```

Дальнейшая информация о разных типах моделей дана в следующих разделах.

Модели примитивов

Значительная часть базовых типов компонентов построены непосредственно в PROSPICE. Эти типы устройств названы Primitives и включают резисторы, конденсаторы, диоды, транзисторы, вентили, счётчики, память и много другое.

Модели примитивов не требуют внешних файлов для симуляции, и они не определяются наличием единственного свойства PRIMITIVE. Дополнительные свойства задаются непосредственно в компоненте и передаются в PROSPICE через netlist. Например, резистор будет иметь:

```
PRIMITIVE=ANALOG,RESISTOR
```

Это определяет элемент, как SPICE Resistor примитив.

Стандартный набор примитивов симулятора можно найти в библиотеках ASIMMDLS и DSIMMDLS. Для этих элементов есть контекстно чувствительная подсказка (help) их свойств, а примеры их использования можно найти в документации VSM SDK.

Схемные модели

Когда симулируются более сложные устройства, общий подход таков — нарисовать схему, которая подражает нужным действиям, используя примитивы симуляции. Эта схема может быть составлена из актуальных внутренних электронных компонентов устройства, но более общим случаем было бы использование идеальных источников тока, напряжения и переключателей для ускорения процесса.

Схемные модели задаются свойством MODFILE, которое по соглашению мы задаём со свойством «только для чтения». Например, операционному усилителю 741 присваивается:

```
MODFILE=OA_BIP
```

Вы могли заметить, что файл модели доступен для нескольких моделей устройств — фокус в использовании *Parameter Mapping Table*.

Схемные модели создаются вычерчиванием схемы в ISIS с последующей компиляцией её *Model Compiler* для производства MDF файла. Дальнейшие детали процесса описаны в документации к VSM SDK.

VSM модели

VSM модели в действительности модели примитивов, которые реализованы во внешних DLL, а не в самом PROSPICE. Они поддерживают функциональность симулируемых устройств, используя язык программирования по вашему выбору, хотя обычно лучше использовать C++.

VSM модель будет иметь и свойство PRIMITIVE (поскольку и ISIS, и PROSPICE трактуют их как примитивы), и свойство MODDLL, которое задаёт имя DLL файла, в котором находится код модели.

Например, модель 8052 имеет:

PRIMITIVE=DIGITAL,8052

MODDLL=MCS8051

Заметьте, что DLL модели может реализовать более одного типа примитива — MCS8051.DLL реализует несколько вариантов 8051.

VSM модели могут также реализовать функциональность, которая относится к анимации, так что электрические и графические аспекты операций компонента могут комбинироваться весьма спокойным образом. Модель LCD дисплея прекрасное тому подтверждение.

Создание VSM моделей вращается вокруг нескольких классов C++ Interface (похожих на COM). Всё это задокументировано в руководстве к VSM SDK.

SPICE модели

Поскольку PROSPICE основан на Berkeley SPICE3F5, он непосредственно совместим со стандартными SPICE моделями, и множество компонентов в библиотеках PROTEUS моделируются, используя SPICE файлы, полученные от производителей компонентов. SPICE модели могут быть заданы либо блоком SUBCKT, либо набором параметров в записи MODEL. SUBCKT модели будут иметь присваивание свойств, как следующие:

PRIMITIVE=ANALOG,SUBCKT

SPICEMODEL=CA3140

где примитив модели SPICE для транзистора может получить следующие свойства:

PRIMITIVE=ANALOG,NPN

SPICEMODEL=BC108

Собственно модель может хранится либо в ASCII (текстовом) файле, либо в библиотеке *SPICE Model*. Имена этих файлов задаются либо свойством SPICEFILE, либо SPICELIB.

Различные детали того, как использовать SPICE модели от производителей, описаны в разделе «Использование SPICE моделей».

НЕПОЛАДКИ

ЖУРНАЛ (LOG) СИМУЛЯЦИИ

При каждой симуляции создаётся запись в файле журнала (Simulation Log). Файл отчёта содержит всю информацию, предупреждения и сообщения об ошибках и самого симулятора, и индивидуальных моделей. Когда сообщение создаётся моделью, оно имеет префикс ссылки модели компонента в квадратных скобках, так что вы можете увидеть:

```
[U1] Loaded 26 files from PROGRAM.HEX
```

В процессе интерактивной симуляции содержимое этого файла может быть отображено во всплывающем окне из раздела *Debug* основного меню, тогда как для симуляции, основанной на графиках, файл может быть просмотрен, если указать на график и нажать **CTRL+V**.

В случае, когда симуляция прекращается полностью, файл отчёта будет отображаться автоматически, чтобы вы могли видеть причину возникающей проблемы немедленно.

ОШИБКИ NETLIST

Эти ошибки обнаруживаются, как результат проблем попыток ISIS создать netlist схемы — вы также столкнётесь с этим, если попытаетесь экспортировать схему в ARES для разводки печатной платы. Общим будет:

- Наличие двух компонентов с одинаковыми именами или компонентов без имени, например, двух резисторов с этикеткой R?.
- Плохо сформированные файлы скриптов, как MAP ON таблицы и т.д. Обратитесь к определениям синтаксиса в руководстве ISIS, если вы не можете выявить проблему немедленно.

ОШИБКИ КОМПОНОВКИ (LINKING)

Компоновка модели — это процесс в котором ISIS вызывает файлы MDF для компонентов, которые моделируются эквивалентными схемами. Безоговорочно, наиболее общей проблемой служит ситуация, когда заданный файл модели отсутствует. Файл модели должен быть в текущей директории или в *Module Path*, как указано в диалоге *Set Paths*.

Другие общие ошибки компоновки включают:

- Значение не найдено в таблице карты параметров. Это означает, что тип элемента — скажем, CA3140, не включён в список в таблице карты заданного файла модели. Файлы модели, такие как OA_MOS.MDF разработаны для моделей нескольких разных компонентов, использующих ту же схему, но с разными значениями. Эта ошибка означает, что заданный файл модели не имеет параметров для типа устройства, которое вы используете — вы можете открыть MDF файл с помощью текстового редактора, чтобы увидеть, какие устройства моделируются.
- Неразрешённый вывод модуля. Это, скорее, вызовет предупреждение, а не сообщение об ошибке, и означает, что корпус родительского компонента имеет вывод, который отсутствует в модели. Часто это не препятствие — например, большинство моделей

операционных усилителей не моделируют отвод от нулевого вывода, но это может быть ошибкой, и, если новая модель не работает, в этом общем случае появляется предупреждение «No DC path to Ground».

ОШИБКИ РАЗБИЕНИЯ

Разбиение (Partitioning) — это механизм, с помощью которого ISIS решает, какая часть(и) схемы нуждаются в симулировании. Проблемы, которые могут возникнуть, это:

- Обнаруживается циклическая зависимость. Это означает, что расположение записывающих устройств (tapes) таково, что секции за и перед записывающим устройством взаимно зависимы. При условии, что вы корректно задали все *Isolate Before* и *Isolate After* флажки на пробниках и генераторах, ваше простейшее действие в этом случае будет, возможно, таким — удалить все объекты записывающих устройств и симулировать всю схему за один заход.
- Не задана модель — это означает, что обнаружен компонент, который не имеет свойств PRIMITIVE, MODFILE, и появился в части схемы, которая нуждается в симуляции. Если устройство неподходящее (например, коннектор), тогда вы должны задать PRIMTIVE=NULL, иначе модель вам нужна!

В руководстве есть раздел, где больше сказано о типах моделей симулятора. Обратитесь к нему.

ОШИБКИ СИМУЛЯЦИИ

Ошибки симуляции генерируются PROSPICE, а не ISIS, и поэтому обнаруживаются после того, как файл netlist был успешно сгенерирован. Общие проблемы, которые при этом обнаруживаются, включают:

- Тип устройства не распознаётся. Это означает, что вы задали тип примитива, который не поддерживается, или что файл модели и уже использовался.
- Нет пути постоянного тока к земле. Это обсуждается в разделе «Шины питания и земли».
- Не находится пробник — вы пытаетесь обратиться к пробнику или генератору напряжения, который не существует. Помните, что вы должны использовать объект IPROBE из ASIMMDLS.LIB — вы не можете ссылаться на гаджет токового пробника.
- Не открывается исходный SPICE файл. Исходный файл, заданный свойством SPICEMODEL, не может быть локализован. Он должен быть в текущей директории или задан в *Module Path* из диалоговой формы *Set Paths*.
- Не находится библиотечная модель. SUBCKT или MODEL, которые вы задали, не существуют в указанной библиотеке или на диске.
- Не находится DLL модель. Заданная VSM модель DLL не может быть локализована. Она должна быть в текущей директории или задана в *Module Path* из диалоговой формы *Set Paths*.

ПРОБЛЕМЫ СХОДИМОСТИ

Этот последний набор ошибок относится к тому, что случается, если SPICE сам терпит неудачу при симуляции. Есть три основных сообщения об ошибке, которые говорят об этом:

- Сингулярная матрица. Похоже, неизвестных больше, чем уравнений, чаще всего это относится к схемам, которые «недорисованы», или в которых некоторые начальные условия нуждаются в том, чтобы давались в порядке, определяющем стартовое состояние.

  Эта ошибка часто предваряется предупреждением «No DC path to ground», и вам нужно проверить соединения вокруг выводов, обозначенных в списке после этого предупреждения. Если часть вашей схемы не заземлена, симулятор не может определиться с напряжением относительно земли — это столь же просто, как это.

- Слишком много итераций без схождения. Это означает, что решение схемы нестабильно. Цепи с примитивами VSWITCH или CSWITCH могут легко создавать такие условия, но любые цепи, передаточные функции которых разрывны, могут создавать серьёзные проблемы для SPICE.

- Временной шаг (Timestep) слишком мал. Это означает, что схема переключается таким образом, что прохождение по времени даже при очень маленьких значениях (обычно 1E-18 сек) все ещё не производит сколь-нибудь заметных малых изменений напряжения в цепи.

  Часто это связано с плохо разработанной моделью, или с отсутствием задания подходящих параметров для моделей диодов или транзисторов. На практике, если ёмкость перехода не выбрана правильно, такое устройство будет показывать нулевое время переключения, которое может непосредственно вызвать это сообщение об ошибке.

Большинство ошибок схождения происходят из-за плохо нарисованной схемы или плохих моделей — время от времени мы получаем схемы, присланные по причине «не симулируется», но только для того, чтобы обнаружилось что-то, что не было соединено. Пожалуйста, проверьте отчёт о симуляции на ошибки, перепроверьте вашу схему, прежде чем делать заключение о неработоспособности PROSPICE. *В тех случаях, когда используются SPICE или VSM модели других производителей, мы не можем тратить время на их отладку, пока не сможем предоставить простую схему, демонстрирующую проблему.*

Остерегайтесь также использования SPICE моделей других производителей, которые не поддерживают стандарт SPICE 2 или SPICE 3. Модели разработанные для PSPICE™ могут включать и элементы, и синтаксические конструкции, которые не являются стандартом SPICE.

Осцилляторы могут стать причиной особых проблем, поскольку начальное решение для рабочей точки потерпит неудачу. В конце концов, осцилляторы не имеют устойчивого состояния! Используйте IC, NS или OFF свойства для определения начального состояния, как это обсуждалось в разделе «Начальные условия».

Если проблема в действительности в числовом расхождении, вы можете попробовать следующие тактики:

- Увеличьте значение GMIN. Это сопротивление утечки для обратно смещённого перехода полупроводника, и наименьшее значение сделает цепь более похожей на цепочку резисторов (которые всегда решаются). Но это ухудшает точность. По умолчанию, это 1E-12; значения больше 1E-9 дадут довольно бессмысленные результаты.

  Заметьте, что в любом случае SPICE3F5 попробует то, что называется продвижением GMIN, если вначале схема не сходится. Это означает, что большое значение GMIN используется для нахождения начального решения, а значение затем постепенно возвращается к его оригинальному значению, чтобы сохранить точность.

- Увеличьте значение ABSTOL и/или RELTOL. Эти значения управляют точностью, которая требуется для симуляции, чтобы она считалась сходящейся. Однако, чем больше вы делаете допуск, тем меньше точные получаете результаты.

- Если схема использует операционные усилители, попробуйте задать MODFILE=OA_IDEAL вместо специфического типа устройства — эту модель намного легче симулировать.

- Уменьшите значение TRTOL. Это заставит SPICE использовать меньшие временные шаги, так что уменьшит вероятность «потерять» сходящееся решение, но это может увеличить время симуляции. Это сработает только тогда, когда симуляция отказывает в части анализа переходного процесса.

  Вы также должны попробовать уменьшить TRTOL, если нарисованные кривые выглядят «изрезанными» или содержат математический шум. Это часто проявляется в виде колебаний значения после быстрой смены уровня.