Разработка моделей цифровых элементов Digital SimCode для Altium Mixed Sim

Современные САПР электронных устройств, например Altium Designer, обладают колоссальной функциональностью в части управления данными проектов, проектирования схем, проектирования печатных плат, инженерного анализа. В Altium Designer инженерный анализ выполняется различными расчетными модулями, применяемыми в зависимости от поставленной задачи. Существуют возможности выполнять пред- и посттопологический анализ целостности сигналов, расчет посттопологических статических электрических нагрузок на цепи питания, расчеты электрических режимов функционирования элементов схемы. Одна из составляющих Altium Designer — высококачественный имитатор электрических схем Altium Mixed Sim. Чтобы упростить программирование разработки моделей цифровых компонентов, в Altium Mixed Sim введена универсальная модель цифрового компонента с использованием языка Digital SimCode.

Юрий ЛЕГАН
yuri.legan@altium.com

Введение

Данный имитатор позволяет выполнять различные виды расчетов электрических режимов схемы на этапе проектирования. В том числе он имеет алгоритмы моделирования смешанных сигналов, реализующих расчет поведения электрической схемы, содержащей как аналоговые, так и цифровые компоненты. Расчет поведения электрической схемы выполняется как расчет переходного процесса во временной области.

Данный подход давно известен и реализован во многих распространенных имитаторах электрических схем. Естественным затруднением этого подхода становится описание моделей цифровых компонентов. Такие компоненты характеризуются обычно большим количеством внутренних элементов, функционирующих в ключевом, или иначе — вентильном, режиме и реализующих такие внутренние состояния своих схем, которые обычно приближены по напряжениям к потенциалам питания. Моделировать схемы на уровне транзисторов — мероприятие чрезвычайно сложное и затратное с точки зрения ресурсов вычислительной системы. Поэтому все цифровые компоненты моделируют на так называемом логическом уровне, интерпретируя значения внутренних функций цифровых компонентов в логической форме, имеющей несколько допустимых статических состояний — 0, 1, X и Z, а также известные динамические взаимные переходы между статическими состояниями. Для решения задачи моделирования цифровых компонентов в имитаторе Altium Mixed Sim реализована поддержка специфических моделей имитатора XSPICE, описывающих цифровые компоненты.

Изначально XSPICE обладал открытой программной спецификацией и допускал «достраивание» своей функциональности за счет разработки дополнительных моделей компонентов в виде программных модулей на языке C++, обладающих определенной совместимостью по программному интерфейсу с ядром имитатора. Этот подход предоставлял широкие возможности для построения моделей, но требовал от инженера нетривиальной специализации дополнительно как программиста. Кроме того, каждое расширение функциональных возможностей имитатора предполагало новую сборку, а значит, доступ к программным библиотекам его реализации и накладывал зависимость на применяемый компилятор: используя разные компиляторы, разработчики расширений ядра имитатора могли получать несколько различающиеся результаты, даже при одном исходном коде.

Для того чтобы снизить требования к навыкам программирования разработки моделей, в ядро имитатора Altium Mixed Sim введена универсальная модель цифрового компонента. Она реализует определяемую пользователем функциональность через восприятие плоского текстового файла, содержащего описание цифрового элемента на языке Digital SimCode. Такой подход позволил отказаться от необходимости расширения имитатора дополнительными пользовательскими программами, а значит, и снизить требования к навыкам разработки ПО со стороны пользователей. Пользователю, разрабатывающему модель цифрового компонента, теперь не нужно иметь специальность программиста, знать язык C++ на хорошем уровне, а достаточно овладеть гораздо более простым языком Digital SimCode.

Назначение Digital SimCode

По своей сути Digital SimCode является языком описания аппаратуры (HDL), эквивалентным по основной функциональности другим языкам подобного назначения. Он предназначен для описания функционирования цифровых элементов на логическом уровне.

Работа модели Digital SimCode основана на принципе событийного моделирования — имитатор отслеживает входящие сигналы и проверяет их на изменение экви-

валентного логического состояния, а также отслеживает внутренние состояния модели. При наличии события изменения состояния входящего сигнала или внутреннего состояния модели формируется обработка события. Измененные выходные сигналы отражаются во внешних цепях моделируемого цифрового компонента. Сигналы модели, обращенные к внешней схеме, называют выводами, или портами модели.

Основные возможности

Digital SimCode позволяет:

- описывать перечни входных и выходных сигналов, указать выводы питания, выводы типа вход/выход, формировать подгружающие эффекты входов;
- указывать выходные нагрузочные характеристики для выходных выводов;
- указывать законы преобразования электрических уровней в логические состояния и логических состояний в электрические уровни;
- описывать логические функции устройства на основе логических термов;
- реализовать циклы, использовать условные ветвления;
- объявлять целочисленные и действительные переменные для вычисления промежуточных результатов или внутренних состояний;
- указывать задержки передачи сигналов;
- формировать события, привязанные к определенному времени исполнения;
- и многое другое.

Модель допускает формировать три типа характеристик задержки распространения сигнала — минимальную, типовую и максимальную. При реализации модели можно указать нужный тип характеристики или назначить его какому-либо параметру имитатора Mixed Sim. Это увеличивает гибкость процесса моделирования, позволяя имитировать технологический разброс быстродействия при изготовлении микросхем.

Функции языка Digital SimCode описывают четыре статических состояния — 1, 0, X и Z, а также динамические переходы между ними. Кроме того, они описывают эквивалентные внутренние сопротивления выходов, что дает возможность учитывать перекрестное взаимодействие выводов двух и более цифровых моделей, имеющих общие внешние цепи. Этот механизм позволяет аккуратно вычислять актуальные действующие напряжения и токи.

Также Digital SimCode содержит функции, предназначенные для улучшения качества отладки моделей за счет формирования сообщений потока моделирования.

В своей работе модель цифрового устройства может опираться на файлы дополнительных данных, содержащие переменные данные для работы модели. Например, это могут быть коды команд цифрового устройства в определенном порядке. Меняя последовательность кодов в файле данных, можно влиять на порядок операций, выполняемых моделируемым цифровым устройством, примерно так, как это происходит в однокристальных микроконтроллерах и подобных устройствах. Файлы с такими последовательностями данных можно формировать в иных программах или вручную. Это позволяет менять поведение модели без изменения ее собственного описания на Digital SimCode.

Все функции языка в совокупности разрешают формировать сколь угодно сложное поведение моделируемого цифрового устройства.

Кодированные модели

Обычно модель цифрового устройства описывают в формате плоского текстового файла в формате ASCII. Для файла незашифрованного описания модели цифрового устройства на Digital SimCode применяют расширение *txt*, что позволяет в полной мере использовать и модифицировать модель. В процессе запуска расчета имитатором производится предварительная сборка модели, которая получает сжатый зашифрованный вид и быстрее исполняется. Этот вид можно извлечь из *log*-файла запуска расчета и перенести в плоский файл в формате ASCII. Для файлов зашифрованного описания модели цифрового устройства следует использовать расширение *scb*.

Файлы описания моделей в зашифрованном формате позволяют защитить модель от преднамеренного внесения изменений. Однако это порождает необходимость держать совместно с зашифрованной моделью описание способа назначения портов модели к выводам УГО элемента (иначе называемого шаблоном назначения) для имитатора Mixed Sim, так как при отсутствии сведений о перечне, порядке и назначении внешних выводов модели нет возможности получить доступ к ее функциональности.

Назначение модели Digital SimCode для Altium Mixed Sim

Требования разработки модели

Использование модели осуществляют, указав свободный шаблон модели устройства Altium Mixed Sim. Для этого в диалоге Sim Model на закладке Model Kind нужно выбрать General в поле model Kind и Generic Editor в поле Model Sub-Kind. Затем следует выбрать значение A в поле SPICE Prefix, сформировать шаблон на нижней закладке Netlist Template и указать путь к файлу, содержащему описание модели-прототипа цифрового устройства имитатора Altium Mixed Sim в поле Model Name. Если файл с описанием модели-прототипа указан корректно, то на нижней закладке Model File будет приведено содержание файла.

Шаблон модели Mixed Sim

Шаблон должен содержать хотя бы одну строку вида:

```
@designator [<перечень имен внешних цепей для входных выводов>] [<перечень имен внешних цепей выходных выводов>] @model
```

Причем перечни имен цепей должны быть указаны в том же порядке, в котором они приводятся в описании модели цифрового устройства на языке Digital SimCode (из этого следует необходимость для зашифрованной модели предоставлять еще и шаблон). Для упрощения формирования шаблона можно сделать предварительное сопоставление вида <вывод УГО>-<порт модели Mixed Sim>-<вывод модели Digital SimCode>.

Модель Mixed Sim xsimcode (параметры модели)

Файл описания модели-прототипа цифрового устройства Altium Mixed Sim должен содержать одну строку вида:

```
.model <имя прототипа> xsimcode(file=<имя файла> func=<имя прототипа> [data=<имя файла данных>] [{mntpmx}])
```

- <имя файла> — полное имя файла, содержащего описание прототипа устройства на Digital SimCode;
- <имя прототипа> — наименование функции-прототипа описания устройства Digital SimCode;
- <имя файла данных> — полное имя файла данных, содержащего исходные данные, обрабатываемые моделью.
- {mntpmx} — указание передачи следующих параметров внутрь модели:
  - Propagation — задержка распространения в устройстве. Установка значений MIN или MAX указывает соответственно на минимальное и максимальное значения, которые обычно отражены в спецификации компонента. По умолчанию используется типичное значение.
  - Loading — входная нагрузочная характеристика. Установка значений MIN или MAX указывает соответственно на минимальное и максимальное значения, которые обычно описаны в спецификации компонента. По умолчанию используется типичное значение.

– Drive — выходная нагрузочная характеристика. Установка значений MIN или MAX указывает соответственно на минимальное и максимальное значения, которые обычно определены в спецификации компонента. По умолчанию используется типичное значение.
– Current — ток потребления устройства. Установка значений MIN или MAX указывает соответственно на минимальное и максимальное значения, которые обычно отражены в спецификации компонента. По умолчанию используется типичное значение.
– PWR value — напряжение питания положительной полярности. Определение параметра перегружает определение зарезервированной переменной в модели-прототипе. Если данный параметр указан, то совместно с ним должен быть определен параметр GND value.
– GND value — напряжение питания отрицательной полярности. Определение параметра перегружает определение зарезервированной переменной в модели-прототипе. Если данный параметр установлен, то совместно с ним должен быть определен параметр PWR value.
– VIL value — входное напряжение, интерпретируемое как сигнал низкого уровня. Определение параметра перегружает определение зарезервированной переменной в модели-прототипе.
– VIH value — входное напряжение, интерпретируемое как сигнал высокого уровня. Определение параметра перегружает определение зарезервированной переменной в модели-прототипе.
– VOL value — выходное напряжение, эквивалентное сигналу низкого уровня. Определение параметра перегружает определение зарезервированной переменной в модели-прототипе.
– VOH value — выходное напряжение, эквивалентное сигналу высокого уровня. Определение параметра перегружает определение зарезервированной переменной в модели-прототипе.
– WARN — установка значения ON является флагом для ошибок: времени установки сигнала, времени удержания, ширины импульса, нарушения минимальной/максимальной частоты и нарушения минимального/максимального напряжения питания. Сообщение об ошибке появляется в течение всей имитации после возникновения нарушения.

Особенности разработки модели Digital SimCode

При разработке модели Digital SimCode необходимо учитывать следующее:

- описание функции-прототипа устройства на Digital SimCode не должно содержать кириллических символов;
- <имя файла> должно быть обрамлено двойными кавычками;
- <имя файла> не должно содержать кириллических символов;
- расширение <имени файла> должно быть *txt* или *scb*;
- <имя файла данных> должно быть обрамлено двойными кавычками;
- <имя файла данных> не должно содержать кириллических символов;
- расширение <имени файла данных> должно быть *dat*;
- <имя файла> и <имя файла данных> может содержать в своем составе специальный префикс {MODEL_PATH}
- признак {mntpmx} просто указывается в описании модели для Mixed Sim при необходимости передать специальные параметры модели.

Специальные параметры модели имеют соответствия среди перечня параметров симулятора (они задаются в разделе Advanced диалога Analyses Setup). В результате приоритет определений возрастает от структурно располагающихся наиболее близко к коду модели: самый низкий у специальных переменных в модели-прототипе Digital SimCode, затем приоритет за значениями параметров симулятора, для передачи значений параметров симулятора должен быть указан флаг {mntpmx}, самый высокий уровень приоритета имеют определения специальных параметров модели Mixed Sim, должен быть указан флаг {mntpmx}.

Прием реализации назначения модели

Назначение модели Digital SimCode не очень сложный процесс сам по себе, но при его выполнении можно быстро запутаться, так как нужно согласовать выводы УГО, выводы контейнерной модели-прототипа типа xsimcode для Mixed Sim и выводы модели-прототипа Digital SimCode. Для этого используются данные шаблона экземпляра модели и данные на закладке Port Map.

Однако процедуру можно значительно облегчить, используя простой прием назначения: нужно привести обозначения выводов УГО в соответствие с номерами выводов модели Mixed Sim. Таким образом, на этапе сопоставления на закладке port mapping диалога Sim Model получился результат вида «первый к первому, второй ко второму» и т. д. При этом следует учитывать, что выводы модели Mixed Sim появятся лишь после того, как будут указаны в шаблоне экземпляра. Поэтому предварительной целью для операции назначения портов является разработка шаблона.

Разработать шаблон экземпляра можно либо на основании исходного описания модели Digital SimCode либо на основании сведений о порядке перечисления и назначении портов в этой модели в разделах Inputs и Outputs. Удобно разрабатывать шаблон уже после создания *mdl*-файла, содержащего описание модели-прототипа Mixed Sim. Отсюда можно вывести следующий порядок действий:

- найти файл модели-прототипа Digital SimCode;
- выписать из него перечень входных и выходных сигналов, пар выводов вход/выход и пары выводов питания;
- разработать описание модели-прототипа для Mixed Sim;
- разработать шаблон экземпляра;
- выполнить назначение портов модели-прототипа Mixed Sim и выводов УГО;
- проверить результат.

Практический пример 1 (назначение модели)

Рассмотрим пример назначения модели на основе модели микросхемы АЦП ADC0800.

Предварительно имеем разработанное УГО элемента. Рассмотри набор шагов, которые позволят выполнить настройку назначения модели Digital SimCode элементу:

1. В качестве первого шага следует определиться с местоположением файла модели-прототипа Digital SimCode и именем файла. Удобно будет скопировать его в пустой каталог поближе к корневому каталогу — это позволит использовать более короткие пути в определениях модели-контейнера (рис. 1).
2. Отрыть файл модели в текстовом редакторе (рис. 2).
3. Создать *mdl*-файл с описанием модели-прототипа Mixed Sim. Следует указать путь к файлу и его имя, а также название модели, описывающей АЦП (рис. 3).

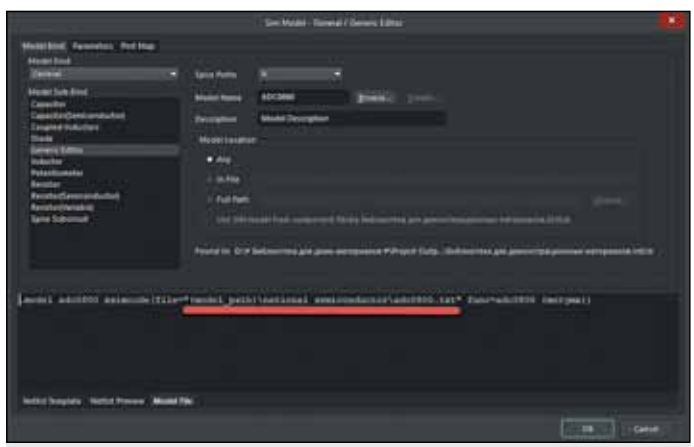

Рис. 1. Рекомендуемый способ сохранения файла модели-прототипа Digital SimCode

Рис. 2. Файл модели Digital SimCode

Рис. 3. Пример mdl-файла с описанием модели-прототипа Mixed Sim

4. Перенести перечень входных и выходных выводов в электронную таблицу. Электронная таблица удобна для представления данных (рис. 4).
5. Перенести в ту же электронную таблицу перечень парных соответствий входных и выходных выводов (рис. 5).
6. Выполнить сопоставление. Перечень входных и выходных выводов модели Mixed Sim должен быть дан в том же порядке, в котором перечислены входные и выходные порты модели Digital SimCode (рис. 6).

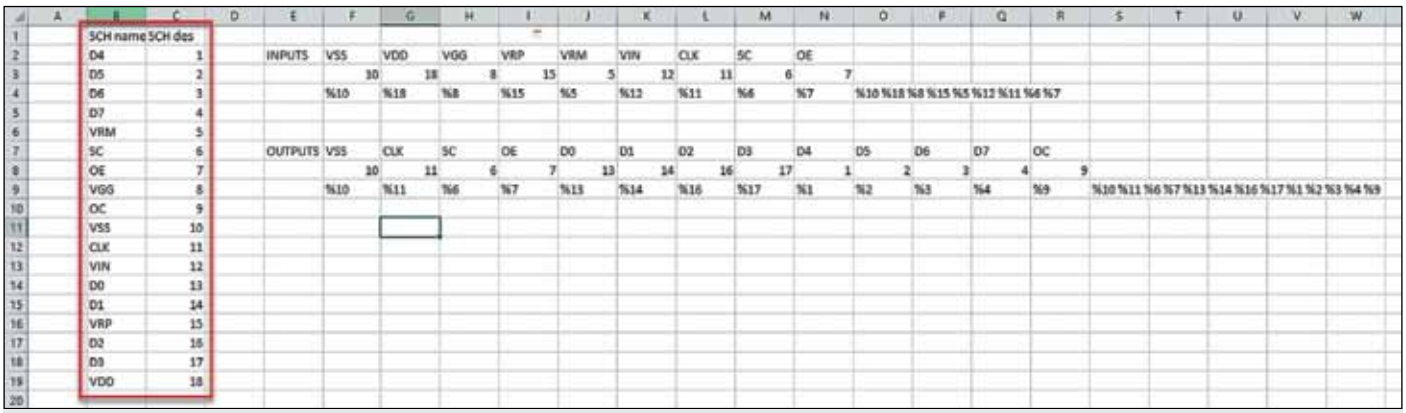

Рис. 4. Перечень входных и выходных выводов в электронной таблице

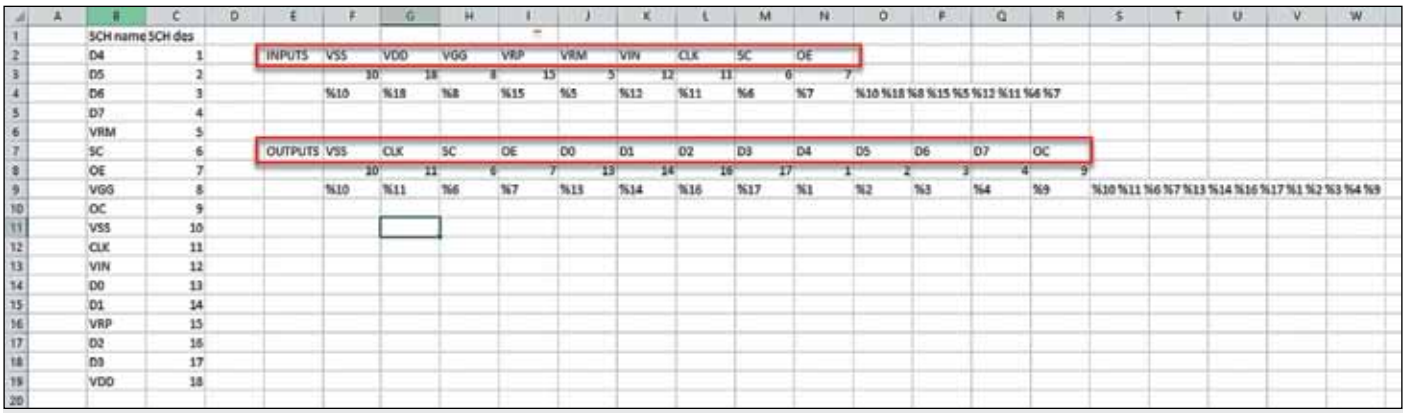

Рис. 5. Перечень парных соответствий входных и выходных выводов в электронной таблице

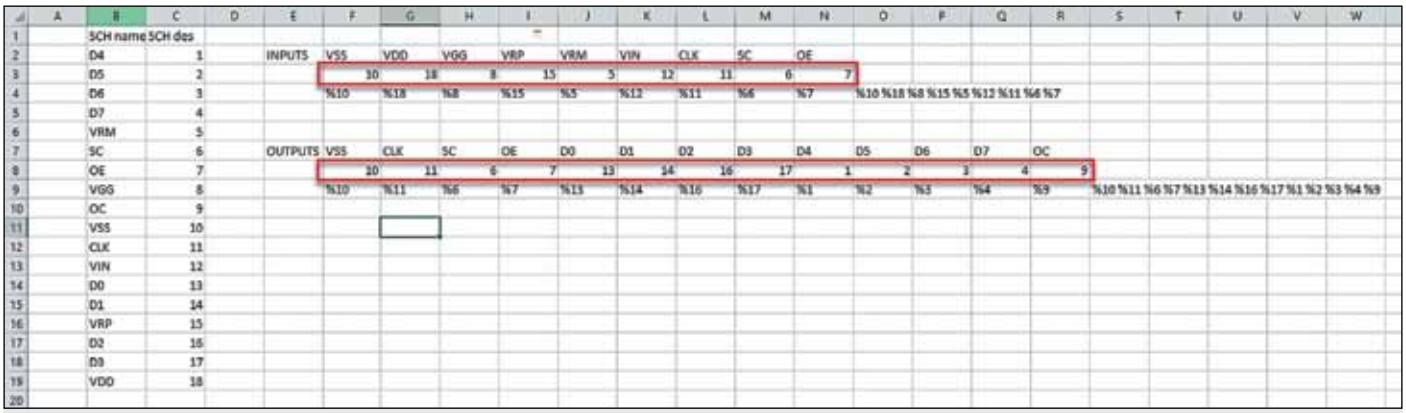

Рис. 6. Порядок входных и выходных выводов модели Mixed Sim должен совпасть с перечнем входных и выходных портов модели Digital SimCode

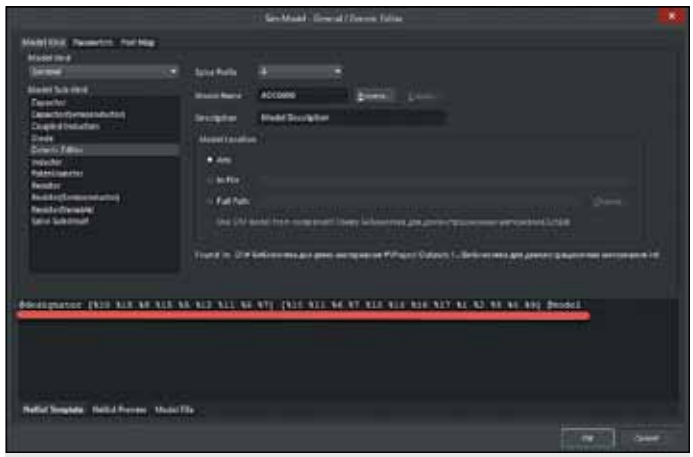

Рис. 7. Шаблон экземпляра модели Mixed Sim

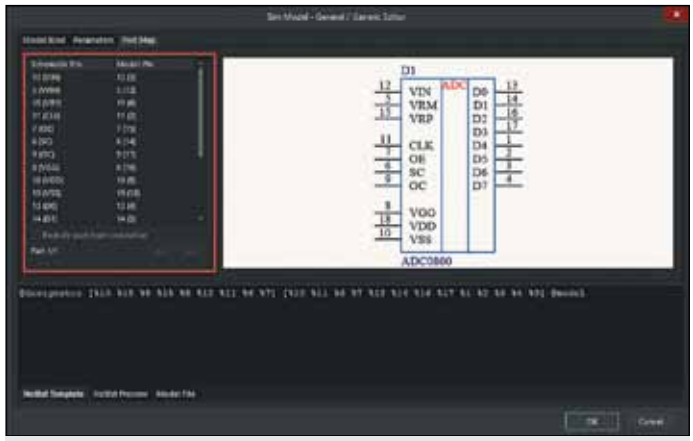

Рис. 8. Закладка Port Map диалога Sim Model

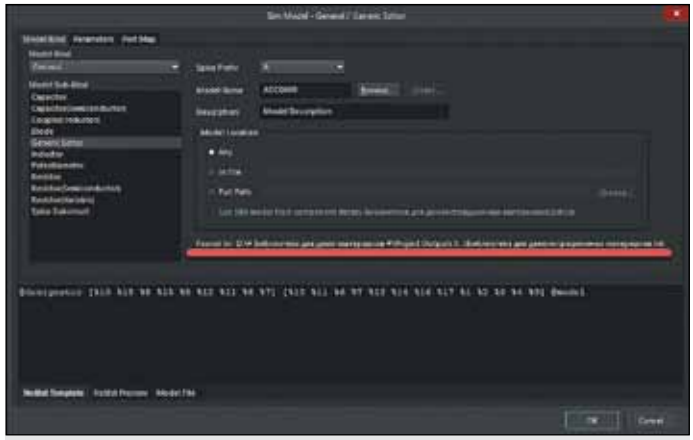

Рис. 9. Закладка Model File диалога Sim Model

7. Сформировать шаблон экземпляра модели Mixed Sim (рис. 7).
8. Проверить информацию на закладке Port Map диалога Sim Model и откорректировать при необходимости (рис. 8).
9. Проверить корректность назначения файла *mdl* на закладке Model File диалога Sim Model (рис. 9).

Условная структура описания модели Digital SimCode

Описание модели должно состоять из следующих условных разделов:
- Идентификация функции устройства.
- Объявление данных.
- Инициализация модели.
- Определение нагрузочных характеристик.
- Функциональность.
- Проверка нарушений.
- Определение задержек выходных сигналов и сообщений событий.

Идентификация функции устройства

Идентификацию устройства следует выполнять с помощью следующей конструкции языка:

```
# <имя модели> source,
```

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

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

Остальные операторы (высказывания) языка, описывающие модель, должны располагаться внутри приведенного описания.

Объявление данных

Digital Sim Code позволяет описывать данные таких типов, как входы и выходы устройства, и внутренние переменные. Взаимодействие модели с окружением производится через входные и выходные порты и порты питания. Каждый порт должен быть перечислен как вход и/или выход, а порты, выполняющие функции ввода и вывода информации в модель одновременно, должны присутствовать в обоих перечислениях. Для описания входов и выходов служат операторы:
- INPUTS — с именами входящих портов;
- OUTPUTS — с именами выходящих портов.

Описание портов следует выполнять через запятую.

Указание портов питания (один порт положительной полярности и одни — отрицательной полярности питания) можно перечислить в INPUTS, а в OUTPUTS — с добавкой _LD к имени (для ясности), поскольку порты питания должны принимать значение входного уровня (напряжение или тока) и оказывать влияние на соединенные с ними цепи. Кроме того, требуется указать порты питания с помощью оператора PWR_GND_PINS — оператора указания портов питания. Благодаря этому имитатор сможет вычислять входные уровни для входных портов и выходные уровни для выходных портов, если не выполнено явное или полное их описание, что обеспечивает зависимость от напряжения питания.

Аналогично двунаправленные порты или порты, оказывающие влияние на входные цепи, требуется указывать с помощью оператора IO_PAIRS — оператора указания двунаправленных портов.

Объявление вспомогательных переменных следует выполнять с помощью операторов INTEGERS (для целочисленных данных), и REALS (для действительных данных). Существуют переменные, зарезервированные в модели в системных целях. Значения таких переменных не допускается изменять явным способом (при помощи операций присвоения).

Инициализация модели

Инициализация модели строится при помощи оператора ветвления:

```
IF (init_sim) THEN
BEGIN
...
...
...
EXIT;
END;
```

Внутри данной конструкции следует указать операторы инициализации. Условный оператор должен содержать зарезервированную целочисленную переменную init_sim, которая равна 1 при первоначальном вызове кода модели и равна 0 при последующих обращениях имитатора к модели. Завершение инициализации должно содержать оператор EXIT, что позволяет выполнить завершение работы модели после выполнения ее инициализации. Дальнейшее обращение имитатора к модели не будет приводить к выполнению блока операторов инициализации.

Определение нагрузочных характеристик

Определение нагрузочных характеристик следует выполнять с помощью:

- оператора DRIVE:

```
<выход1> [<выход2> …] = (v0=<значение> v1=<значение> ttlh=<значение> tthl=<значение>)
```

для указания выходной характеристики (питающей способности) выходных портов;

- оператора LOAD:

```
<вход> [<вход> …] = (v0=<значение> r0=<значение> [v1=<значение> r1=<значение>] [io=<значение>] t=<значение>)
```

для указания нагрузочной характеристики входных портов.

Указание параметров питающей способности и нагрузочной характеристики с помощью DRIVE и LOAD позволяет очень детально и аккуратно описать взаимодействие модели с окружением и добиться высокой реалистичности и точности расчета.

Функциональность

Описание функциональности цифрового устройства Digital SimCode можно выполнять, используя операторы ветвления, циклов, таблиц истинности и иных привычных методов, подобных другим HDL-языкам. Можно выполнять безусловные переходы к определенным меткам, что позволяет применять подпрограммы.

Также можно использовать операторы подключения к внешнему (по отношению к модели) файлу данных при помощи оператора READ_DATA. Например, таким образом удается реализовать цифровой автомат с выбором состояния из внешнего файла и подобные устройства.

Проверка нарушений

В Digital SimCode существует возможность формировать утверждения, которые позволяют реализовать отладку модели — выполнять вывод сообщений в панель Messages.

Примерами операторов проверки нарушений являются:

- FREQUENCY — проверяет входные порты на нарушение минимальной и максимальной частот;
- CHANGED_XX — для проверки изменения порта, сравнивает текущее состояние вывода с состоянием при предыдущем вызове модели;
- RECOVER — тестирует порты на нарушения времени восстановления;
- SETUP_HOLD — тестирует порты на нарушения времени установки/удержания;
- SUPPLY_MIN_MAX — проверяет разницу напряжений на входах питания;
- WIDTH — сравнивает ширины импульсов для входных портов с указанным временем.

Определение задержек выходных сигналов и сообщений событий

Эффекты запаздывания/распространения выходного сигнала можно имитировать с помощью оператора:

```
DELAY <выход1> [<выход2>...] = <задержка>.
```

Вместо <длительность> можно использовать несколько связанных операторов CASE, тогда выражение будет иметь вид:

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

Таким образом оператор DELAY может иметь два вида синтаксиса.

Формировать события можно при помощи оператора EVENT.

Если производится отладка модели или модель разрабатывается с внедрением контроля исполнения, то для формирования сообщений можно использовать оператор PROMPT. При этом происходит остановка программного потока модели. Действие оператора MESSAGE подобно оператору PROMPT, за исключением того, что не происходит остановки программного потока.

Практический пример 2 (разбор модели ADC0800)

В качестве примера условного описания структуры модели Digital SimCode рассмотрим листинг модели микросхемы АЦП National Instruments ADC0800.

```
//=======Идентификация функции устройства============================
# adc0800 source
//============================================================
//========Объявление данных=========================================
INPUTS VSS, VDD, VGG, VRP, VRM, VIN, CLK, SC, OE; // объявление входных портов модели — принимающих сигналы
OUTPUTS VSS_LD, CLK_LD, SC_LD, OE_LD, D0, D1, D2, D3, D4, D5, D6, D7, OC; //объявление выходных портов модели — формирующих сигналы
INTEGERS internal_reg, clk_count, oc_bit, ck_vgg; // объявление внутренних целочисленных переменных
REALS tt_val, tp_val, rin_val, vin_val, ridd_val, tmp_val; // объявление внутренних действительных переменных
//==Смешанное объявление данных и определение контроля нарушений===========
PWR_GND_PINS (VSS,VDD); // установка значений параметров pwr_param и gnd_param
SUPPLY_MIN_MAX (4.75,10.25); // контроль минимального напряжения питания =4.75 и максимального напряжения питания =10.25
VOL_VOH_MIN (0,0,0.1); // установка минимума напряжения для выходных портов vol_param=gnd_param+0, максимума напряжения для выходных портов voh_param=pwr_param-0
VIL_VIH_PERCENT (33,66); // определение значений параметров vil=33% от размаха питания, vih=66% от размаха питания для P-структуры МДП
IO_PAIRS (CLK:CLK_LD, SC:SC_LD, OE:OE_LD); // определение портов CLK, SC, OE в качестве входов-выходов
//============================================================
//==============Инициализация=======================================
IF (init_sim) THEN
BEGIN
//MESSAGE("time\tVin\treg\tD7\tD6\tD5\tD4\tD3\tD2\tD1\tD0\tSC\tOC");
// Приведены параметры сопротивлений (системные параметры для модели)
//Примечание: значения ttlh и tthl одинаковы
tt_val = (MIN_TYP_MAX(tt_param: NULL, 60n, 120n));
//Примечание: значения tplh и tphl одинаковы
tp_val = (MIN_TYP_MAX(tp_param: NULL, 200n, NULL));
//Параметры выходных портов для низкого уровня напряжения IOL max=2 мА @ vol=0.4 В => rol_param=(0.4-vol_param)/2 мА
rol_param = (MIN_TYP_MAX(drv_param: NULL, 100, NULL));
//Параметры выходных портов для высокого уровня напряжения IOH max=-200 мкА @ voh=4.5 В => roh_param=(voh_param-4.5)/200 мкА
roh_param = (MIN_TYP_MAX(drv_param: NULL, 500, NULL));
//Утечка входа @ 25 °C: IIN=1 мкА @ VDD=5 В => rin= (5/1 мкА);
rin_val = (MIN_TYP_MAX(ld_param: NULL, 5E6, NULL));
IF (oc_bit) THEN // если oc_bit не равно 0, тогда
BEGIN
STATE OC = ONE; // установить порт OC в 0
ELSE
STATE OC = ZERO; // иначе, установить порт OC в 1
END;
IF (OE) THEN // если значение порта OE не равно 0
BEGIN
STATE_BIT D0 D1 D2 D3 D4 D5 D6 D7 = (internal_reg); // вывести значение на выходные порты внутреннего регистра internal_reg, предварительно разложив его на битовые компоненты.
STATE_BIT D0 = (!(D0)); // инвертировать порт D0
STATE_BIT D1 = (!(D1)); // инвертировать порт D1
STATE_BIT D2 = (!(D2)); // инвертировать порт D2
STATE_BIT D3 = (!(D3)); // инвертировать порт D3
STATE_BIT D4 = (!(D4)); // инвертировать порт D4
STATE_BIT D5 = (!(D5)); // инвертировать порт D5
STATE_BIT D6 = (!(D6)); // инвертировать порт D6
STATE_BIT D7 = (!(D7)); // инвертировать порт D7
ELSE
STATE D0 D1 D2 D3 D4 D5 D6 D7 = UNKNOWN; // иначе, установить неизвестное состояние
END;
IF (warn_param && (present_time > 0)) THEN // если warn_param не равен 0 и present_time (текущее время модели) больше 0, тогда
```

```
BEGIN
IF (SC && (clk_count = 4)) THEN // если порт SC не равен 0 и clk_count равен 4, тогда
BEGIN
MESSAGE ("WARNING: Possible conversion problem — SC was high for more than 3 CLK cycles");
// Сообщение «ПРЕДУПРЕЖДЕНИЕ: Возможна проблема преобразования — SC имел высокий уровень больше 3 циклов входа CLK»
END;
//=========Проверка различных нарушений — проверка VGG============
IF (ck_vgg) THEN // если ck_vgg не равен 0, тогда
BEGIN
ck_vgg = (0); // присвоить ck_vgg в 0
vin_val = (VALUE(VIN)); // присвоить переменной vin_val значение напряжения на входе VIN
IF (vin_val > pwr_param) THEN // если vin_val больше pwr_param (напряжения питания), тогда
BEGIN
MESSAGE ("WARNING: VIN(%fV) > VSS(%fV)",vin_val,pwr_param); // Сообщение «ПРЕДУПРЕЖДЕНИЕ: Напряжение на входе превышает напряжение питания»
ELSE
tmp_val = (VALUE(VGG)+7); // присвоить переменной tmp_val значение напряжения порта VGG+7
IF (vin_val < tmp_val) THEN // если vin_val меньше tmp_val, тогда
BEGIN
MESSAGE ("WARNING: Inadequate voltage to analog switches\n\tVIN(%fV) < VGG+7(%fV)",vin_val,tmp_val);
// Сообщение «ПРЕДУПРЕЖДЕНИЕ: Неправильное напряжение на аналоговых ключах»
//Ток потребления питания Idd @ 5 В: 250= 5/20 мА макс
ridd_val = (MIN_TYP_MAX(i_param: NULL, NULL, 250));
// Инициализация переменных
oc_bit = (1);
ck_vgg = (1);
clk_count = (0);
internal_reg = (0);
STATE OC = ONE; // инициализация выходного порта OC
STATE D0 D1 D2 D3 D4 D5 D6 D7 = ZERO; // инициализация выходных портов
EXIT;
END;
//=====================================================
//=======Описание нагрузочных характеристик==================
DRIVE D0 D1 D2 D3 D4 D5 D6 D7 OC = (v0=vol_param,v1=voh_param,ttlh=tt_val,tthl=tt_val);
// определение питающей способности для перечисленных выходных портов
LOAD CLK_LD SC_LD OE_LD = (v0=vol_param,r0=rin_val,v1=voh_param,r1=rin_val,io=1e12,t=1p);
// определение нагружающей способности для двунаправленных портов
//=====================================================
//=========Описание функционирования=====================
IF (CHANGED_LH(SC)) THEN // если произошло изменение L->H для порта SC (start conversion) — запуска цикла преобразования
BEGIN
oc_bit = (0); // установить oc_bit  в 0
clk_count = (0); // установить clk_count в 0
END;
IF (CHANGED_LH(CLK)) THEN // если произошло изменение L-H для порта CLK
BEGIN
clk_count = (clk_count + 1); // увеличить clk_count на 1
END;
IF (clk_count = 40) THEN // если значение clk_count достигло 40
BEGIN
oc_bit = (1); // установить oc_bit в 1
tmp_val = (VALUE(VRP) - VALUE(VRM)); // вычислить значение вспомогательной переменной tmp_val на основании значений напряжения
IF (tmp_val <= 0) THEN // если tmp_val <=0, тогда
BEGIN
MESSAGE ("WARNING: Vref+ <= Vref-"); // вывод сообщения «ПРЕДУПРЕЖДЕНИЕ: на входы опорного напряжения подана неверная полярность»
ELSE
internal_reg = ((VALUE(VIN) * 256)/tmp_val); // иначе, вычислить значение переменно internal_reg (внутренний регистр) на основании операции квантования
END;
END;
ELSE
vin_val = (VALUE(VRM)); // присвоить переменной vin_val значение напряжения на порте VRM
IF (vin_val < tmp_val) THEN // если vin_val меньше tmp_val (напряжения на порте VRM), тогда
BEGIN
MESSAGE ("WARNING: Inadequate voltage to analog switches\n\tVref-(%fV) < VGG+7(%fV)",vin_val,tmp_val);
// Сообщение «ПРЕДУПРЕЖДЕНИЕ: Неправильное напряжение на аналоговых ключах»
ELSE
ck_vgg = (1); //нет ошибок — продолжение проверки VGG
END;
END;
END;
END;
FREQUENCY (CLK MIN=50k MAX=800k "CLK"); // контроль минимальной и максимальной частоты для входного порта CLK
END;
//=====================================================
//=======Описание нагрузочной характеристики==================
LOAD VSS_LD = (v0=gnd_param,r0=ridd_val,t=1p); // определение нагружения порта VSS
//=======Определение задержек выходных сигналов================
DELAY D0 D1 D2 D3 D4 D5 D6 D7 OC = (tp_val);
//=====================================================
EXIT;
```

Литература

1. Руководство по Digital SimCode. **www.ru.resources.altium.com/russian-guide-books/руководство-digital-simcode**
2. Вебинар «Разработка моделей цифровых элементов Digital SimCode для Altium Mixed Sim» (видео). **www.youtube.com/watch?v=PNC9QIWlJPw**