1 Проблемы проектирования систем на кристалле

Современные тенденции развития заключается в применении встраиваемых систем на основе систем на кристалле (System-on-Chip (SoC)) или (СБИС СнК) в разработке электронных систем. Такие SoC-решения обычно состоят из встроенного процессора (процессоров), встроенных памяти, аппаратных ускорителей (или IP-ядер), высокоскоростных коммуникационных интерфейсов и реконфигурируемой логики. Вследствие этого разработки таких электронных систем становятся все более сложными, поскольку они предъявляют более жесткие требования к более низкой стоимости, более высокой производительности, качеству продукции, безопасности и скорости продаж. Кроме того, в соответствии с законом Мура при дальнейшем развитии возможностей цифрового оборудования существует потребность, чтобы увеличенное число функциональных возможностей содержались в более ограниченном пространстве прибора.

Традиционно во всех проектах предусматривалось создание сложной интегральной схемы (ИС), состоящей из 500000 вентилей. По существу сложные схемы состояли из основной логики и некоторых жестких макросов, таких как встроенная память. С быстрым прогрессом в технологии обработки полупроводников плотность вентилей на кристалле увеличилась в соответствии с тем, что предсказывал закон Мура. Это помогло реализовать более сложные конструкции на одной и той же ИС.

За последние несколько лет с появлением таких передовых технологий, как услуги мобильной связи, предоставляющие доступ в Интернет через мобильные телефоны, возрастает потребность в том, чтобы разместить традиционные микропроцессоры, память и периферийные устройства на одной микросхеме. Это было отмечено как начало эпохи SoC. Обычно SoC содержит один или более программируемых процессоров, встроенную память, таймеры, контроллеры прерываний, шины, специально разработанные сложные аппаратные модули и встроенное программное обеспечение.

Проектирование конструкции системы на кристалле (SoC) имеет особенности:

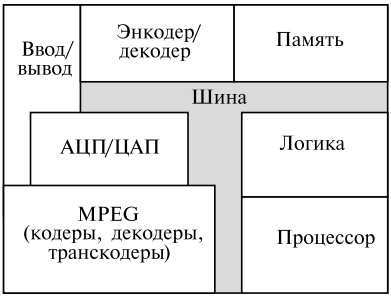

Рис. 1.1. Типичная конструкция SoC

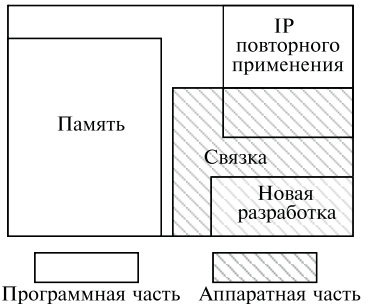

Рис. 1.2. Соотношение аппаратной и программной части в SoC

- несколько областей проектирования: аппаратное, программное, аналоговое;
- множество исходных компонентов: DSP (Digital signal processor), ASIC (Application-Specific Integrated Circuit), IP (Intellectual Property)-ядра;
- жесткие ограничения: работа в реальном времени с низким энергопотреблением.

Существенные ресурсы каждого SoC составляют IP-ядра. Ядро IP — это сложный модуль, выполняющий определенную задачу и созданный для повторного использования. Эти IP-ядра являются строительными блоками для SoC и занимают очень большой процент площади SoC.

Одна из ключевых задач проекта SoC — разбиение системной функциональности на аппаратное обеспечение (HW) и программное обеспечение (SW) или совместные HW/SW. Функциональность, однажды отнесенная к программному обеспечению, теперь для лучшей производительности может быть реализована на аппаратном уровне, а компоненты аппаратного обеспечения должны интегрироваться с программными API (Application Programming Interface) более высокого уровня. В текущей методологии САПР делается априорное разделение, и таким образом создаются отдельные аппаратные и программные спецификации. Изменения в разделении HW/SW требуют обширной реорганизации, которая обычно заканчивается неоптимальными проектами. Кроме того, при внедрении встроенных процессоров в ПЛИС дизайнеры цифровых устройств знакомятся с новой областью САПР, которая включает в себя одновременную разработку как аппаратного, так и программного обеспечения (программа выполняется на встроенном процессоре).

На рис. 1.2 условно показано соотношение аппаратной и программной части в SoC.

Еще одним критическим недостатком текущей методологии САПР является то, что она является RTL (register transfer level)-ориентированной и в связи с увеличением сложности схемы время моделирования возрастает и постепенно становится неприемлемым. Понятно, что скорость моделирования полной системы является решающим фактором при разработке сложной цифровой системы, такой как SoC. Эта проблема проверки еще более усугубляется, когда число тестовых векторов, необходимых для проверки, возрастает в 100 раз каждые шесть лет, что в 10 раз превышает число вентилей на чипе, указаное законом Мура. Кроме того, полная верификация (проверка соответствия техническим условиям) и валидация (проверка соответствия поставленным функциональным целям) системы часто невозможна до тех пор, пока полностью не построен рабочий прототип. Это особенно актуально для распределенной и гетерогенной среды, такой как сетевая встроенная система.

Очевидно, что для сокращения времени разработки и затрат компьютерные инструменты проектирования (САПР) теперь должны включать функции, которые облегчают проектирование пространственное и архитектурное, а также дают более высокий уровень абстракции. Среди решений вышеупомянутых вопросов проектирования, которые сегодня активно обсуждаются, есть проектирование на уровне абстракции электронной системы (ESL — Electronic System Level) и примение стандартного языка разработки SystemC.

Этот подход потребует совместного проектирования и совместного моделирования аппаратно-программного обеспечения (HW/SW), он облегчает исследования в области проектирования и дает высокую скорость моделирования. Предложены методологии совместного моделирования, основанные на SystemC, с симулятором набора инструкций в качестве модели процессора в общем исходном файле.

Перечислим основные факторы, которые привели к созданию новой методологии проектирования, основанной на SystemC:

- сложность SoCs продолжает увеличиваться;
- использование моделирования на уровне вентилей или даже RTL для характеристики и изучения полного SoC для типичных сценариев прецедентов не представляется возможным;
- создание моделей на этих уровнях занимает довольно много времени;
- полные модели обычно готовы на поздней стадии разработки системы, и внести в них изменения затруднительно или невозможно;
- симуляция идет слишком медленно.

Для решения проблемы требуется:

- модель с более высоким уровнем абстракции на основе SystemC;

- меньше архитектурных деталей в начальном потоке проектирования;
- более высокая скорость моделирования;
- возможность моделирования более сложных систем.

1.1. Цели SystemC

Одной из основных задач SystemC является моделирование на системном уровне. Это моделирование систем над уровнем RTL-абстракции, в том числе систем, которые могут быть реализованы в программном или аппаратном обеспечении или некоторой комбинации этих составлющих. Модели RTL имеют целью реализацию системы с использованием регистров и комбинационной логики. Одной из проблем в создании языка проектирования на уровне системы является то, что существует широкий спектр дизайнерских моделей вычисления, дизайна уровней абстракции и дизайна методологии, которые используются в конструкции системы. Для решения в SystemC этой небольшой, но очень важной задачи, к языку было добавлено системное моделирование. На вершине этого языкового фундамента мы можем добавить более конкретные модели вычислений, библиотек проектирования, методических указаний по моделированию и методологии дизайна, которые необходимы для проектирования системы. Небольшой, универсальный фундамент моделирования в SystemC называется ядром языка и является центральным компонентом стандарта SystemC 2.0.

Стандарт SystemC 2.0 был введен OSCI в 2001 г., и эта книга использует спецификацию к данному стандарту. В настоящее время распространяется версия SystemC-2.3.1, выпущенная в 2014 г., и все примеры взяты нами для этой версии.

Другие компоненты стандарта SystemC 2.0 включают в себя элементарные модели и библиотеки, построенные на ядре языка (например, таймеры, FIFO, сигналы и т. д.), которые широко применяются. Следует признать, что много различных моделей вычислений и проектных методик могут быть использованы в сочетании с SystemC. По этой причине расчетные библиотеки и модели, необходимые для поддержания этих специфических методик проектирования, целесообразно отделять от ядра стандартного языка SystemC 2.0.

SystemC 2.0 — это язык моделирования, основанный на C++. Он расширяет возможности C++, позволяя моделировать описания аппаратных средств, добавляет моделирование аппаратных описаний, библиотеку классов функций, типов данных и другие языковые конструкции на C++. Эта библиотека классов содержит мощные новые механизмы, которые позволяют проектным группам моделировать и проверять конструкции, выраженные в истинных системных уровнях

абстракции, уточнить их, чтобы отразить варианты реализации и, наконец, связать модель системы с реализацией и проверкой аппаратного обеспечения.

SystemC является надстройкой C/C++ и содержит специальные библиотеки для моделирования HW.

От C/C++ SystemC включает в себя следующие функции:

- классы и объекты;
- инкапсуляция — данные и поведение;
- перегрузка операторов — новые типы и поведение;
- усиление печати — дополнительная защита;
- наследование — повторное использование декларации;
- шаблоны — шаблоны построения.

Преимущества процесса проектирования на основе C/C++ состоят в следующем:

- производительность;
- технические характеристики между этапами разработки архитектуры и исполнением являются изменяемыми;
- высокоскоростное и высокоуровневое моделирование и прототипирование;
- уточнение, отсутствие перевода на аппаратное обеспечение (отсутствие «семантического разрыва»);
- уровень системного моделирования;
- завтрашние разработчики систем будут разрабатывать больше программного обеспечения и меньше аппаратного обеспечения;
- совместная разработка, совместное моделирование, совместная проверка, совместная отладка;
- аспект повторного использования;
- оптимальная поддержка повторного использования объектно-ориентированными методами;
- эффективное повторное использование testbench;
- особенно широко распространен и широко используется C/C++!

Недостатки процесса проектирования на основе C/C++:

- C/C++ не был создан для проектирования аппаратного обеспечения;
- C/C++ не поддерживает:
  - аппаратную коммуникацию (сигналы, протоколы);
  - понятие времени (тактирование, операции с чередованием по времени);
  - параллельность (аппаратное обеспечение является одновременно параллельным, работает параллельно);
- реактивность (аппаратура по своей сути реактивна, реагирует на воздействия, взаимодействует со своей средой, требует обработки исключений);

- типы аппаратных данных (тип бита, тип вектора битов, многозначные логические типы, целые типы со знаком и без знака, типы с фиксированной точкой).

SystemC — это единый язык для определения, совместного моделирования, уточнения системы аппаратных и программных компонентов вплоть до уровня передачи регистров для синтеза. SystemC предоставляет ядро для моделирования исполняемой системы, написанное в SystemC. Он может быть использован для описания циклических точных моделей на системном уровне, разработки алгоритмов программного обеспечения и аппаратной архитектуры.

SystemC отвечает следующим требованиям:

- разрешает совместное проектирование аппаратного и программного обеспечения и совместную проверку;
- быстрое моделирование для проверки и оптимизации;
- плавный переход на аппаратное и программное обеспечение;
- поддержка повторного использования проекта и архитектуры;
- тактирование.

Рис. 1.3. Изучение конструкций на разных уровнях абстракции

Проектирование систем на кристалле (SoC) необходимо выполнять на четырех уровнях абстрактного описания:

- архитектура;
- поведенческий уровень;
- RTL-уровень;
- уровень вентилей.

Скорость итераций увеличивается при проектировании на системном уровне (рис. 1.3).

К языку программирования на системном уровне предъявляются следующие требования:

- спецификация и дизайн на различных уровнях абстракции;
- создание исполняемых спецификаций конструкции;
- создание исполняемых платформ моделей, представляющих возможную реализацию архитектуры;
- быстрое моделирование для того, чтобы исследовать дизайн-пространство;
- конструкции для разделения функциональности от коммуникаций.

Сравнение применимости различных языков проектирования для решения задач разного уровня показано на рис. 1.4. Как видно, SystemC имеет наиболее широкие возможности моделирования систем на разных уровнях описания.

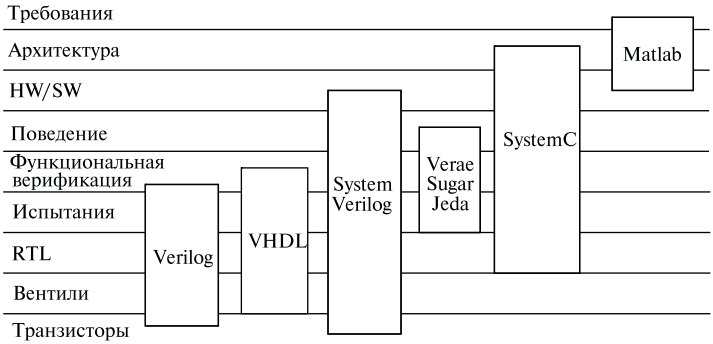

Рис. 1.4. Сравнение применимости различных языков проектирования

Сравнение языков. В связи с быстро возрастающей сложностью конструкции и ростом стоимости ошибки или отказа, разработчикам системы в большинстве областей продукции требуется похожий подход проектирования сверху вниз, но с улучшенной методологией. Эта возникшая методология основана на моделировании на уровне транзакций (Transaction level model — TLM) и используется в SystemC.

Моделирование на уровне транзакций является новой концепцией без точного определения. Рабочая группа Open SystemC Initiative (OSCI) в настоящее время дала определение набора терминологии для TLM и в конечном счете развивает TLM-стандарты. На самом деле, когда инженеры говорят о TLM, они, вероятно, говорят об одном или нескольких из четырех различных стилей моделирования.

Использование TLM позволяет испытывать модели на ранних этапах процесса разработки системы. TLM является концепцией, которая не зависит от языка. Тем не менее для реализации и результативности TLM модели полезно иметь такой язык, как SystemC, который имеет функции поддержки независимого уточнения функциональных возможностей и коммуникаций, что имеет решающее значение для эффективного развития TLM.

1.2. Уровни моделирования в SystemC

В SystemC используют семь уровней моделирования:

1. *Executable specification* (исполняемая спецификация) — это модель, которая прямо передает спецификацию проекта в SystemC. Модель предусматривает функционирование конструкции таким образом, что она не зависит от возможного выполнения. Если присутствуют временные задержки, они учитываются в исполняемой модели.

2. *Untimed functional model* подобна предыдущей, но временные задержки не присутствуют в модели. Обычно коммуникации модели-

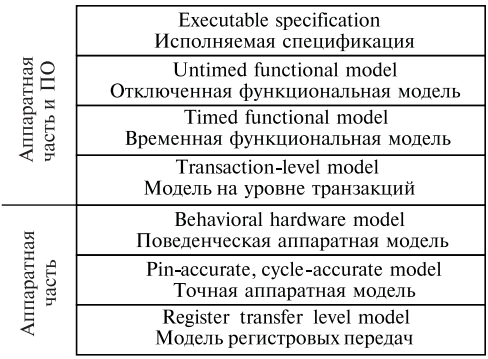

Рис. 1.5. Уровни моделирования в SystemC

руются с использованием FIFO с блокировкой записи и чтения, так что данные надежно передаются между моделями.

3. *Timed functional model* — по-прежнему прямая передача данных, задержки добавлены к процессам и отражают особенности функционирования модели. Применяют на ранних стадиях компромиссного анализа аппаратного и программного обеспечения.

4. *Transaction-level model.* Связи между модулями моделируются с использованием функций вызовов. В такой модели коммуникации обычно моделируются в терминах функционирования и в терминах времени, но структурно точно коммуникации не моделируются. Например, в TLM для платформы SoC мы можем моделировать различные типы транзакций, которые поддерживаются на шине чипа (короткие транзакции чтение/запись), но мы не можем моделировать реальные проводники шины или пины, которые соединяют модуль с шиной.

Когда используют термин *platform transaction-level model*, показывают, что модель использует TLM-стиль с целью моделировать инфраструктурные связи в платформе SoC и что модули вне такой конструкции структурно соответствуют блокам вне целевой реализации. Этот метод используют для уточнения эффектов в модели, и он является более качественным, точным и эффективным путем моделирования взаимосвязи HW и SW на самом раннем этапе проектирования.

5. Behavioral hardware model — это модель, имеющая контактную и функциональную точность на границах, тактовая точность на границах не учитывается. Эта модель может использоваться как входная для средств поведенческого синтеза.

6. *Pin-accurate, cycle-accurate hardware model* — эта модель обеспечивает контактную и тактовую точность на границах в добавление к функциональной точности. Для модели не требуется внутренняя структура, которая влияет на целевую реализацию.

7. *Register-transfer level model* имеет на своих границах контактную и тактовую точность. В добавление внутренняя структура RTL-модели точно отражает регистры и комбинационную логику целевой реализации.

1.3. Краткая история создания и развития SystemC

SystemC является результатом эволюции многих концепций в исследовательских и коммерческих сообществах EDA (Electronic Design Automation). Многие исследовательские группы и EDA компании внесли свой вклад в этот язык.

SystemC начиналась как очень ограниченный циклический симулятор и «слегка отличный» от языка RTL. Язык эволюционировал и развивался до истинного языка проектирования системы, который включает как программные, так и аппаратные концепции. Хотя SystemC специально не поддерживает аналоговое оборудование или механические компоненты, нет никаких причин, почему эти аспекты системы не могут быть смоделированы с помощью SystemC-конструкций или методами co-simulation. В последние году выпущена расширенная версия языка SystemC-AMS для проектирования аналоговых и смешанных систем на кристалле.

Некоторые из организаций, которые внесли значительный вклад в изучение языка разработки, очень рано поняли, что любой новый язык дизайна должен быть открытыми для сообщества и не быть частным или собственным.

В результате в 1999 г. была создана Open SystemC Initiative. OSCI была сформирована для того, чтобы:

- развивать и стандартизировать язык;
- облегчить общение между пользователями и поставщиками языка;
- улучшить адаптацию;
- обеспечить механизмы для разработки программ с открытым исходным кодом и их поддержки.

Основные даты в развитии SystemC:

- открытая инициатива SystemC (OSCI) — платформа языка и моделирования на основе C++;
- сентябрь 1999 г. — SystemC 0.9 — первая версия, основанная на цикле;
- март 2000 г. — SystemC 1.0 — широко доступный основной выпуск, набор конструкций для RTL и поведенческого моделирования;
- август 2002 г. — SystemC 2.0 — каналы и события, более чистый синтаксис, предоставление возможности моделирования на системном уровне для программных и аппаратных реализаций;

- апрель 2005 г. — SystemC TLM 1.0 (моделирование уровня транзакций);
- сентябрь 2005 г. — SystemC 2.1;
- июль 2006 г. — SystemC 2.2 (обновлено в марте 2007 г.);
- июнь 2008 г. — SystemC TLM 2.0.0 (библиотека);
- июль 2009 г. — LRM SystemC TLM-2.0 (библиотека TLM-2.0.1);
- март 2010 г. — SystemC AMS 1.0 LRM;
- ноябрь 2011 г. — стандарт IEEE 1666-2011;
- июль 2012 г. — SystemC 2.3 (интегрировано с TLM);
- март 2013 г. — завершена разработка SystemC AMS 2.0;
- апрель 2014 г. — SystemC 2.3.1 (интегрирована с TLM).

SystemC состоит из языковых и потенциально-методологических библиотек. Создана библиотека верификации SystemC. Авторы рассматривают SystemC Verification (SCV) как наиболее значимую из этих библиотек. Эта библиотека добавляет поддержку современного языка проверки высокого уровня и такие концепции, как ограниченная рандомизация, самоанализ и запись транзакций. Первый выпуск библиотеки SCV произошел в декабре 2003 г. после более чем годового бета-тестирования.

Несмотря на то что цифровые схемы являются специальным типом аналоговых схем и методов, для анализа и понимания аналоговых и смешанных сигналов (AMS) до настоящего времени для проектировщиков аналоговой системы использовался слишком сложный и отнимающий много времени аппарат. Расширение SystemC-AMS для C++ — важнейшая отправная точка в решении этого вопроса и основывается на существующих версиях SystemC. Для удовлетворения особых требований аналоговых систем и цифрового оборудования с программным обеспечением требуется взаимодействие с их физической аналоговой средой. Например, когда цифровое аппаратное/программное обеспечение взаимодействует с радиочастотными системами, датчиками и исполнительными механизмами и силовой электроникой, анализ должен не только решать проблемы, характерные для чисто аналоговых и чисто цифровых систем, а также моделировать их взаимодействие в режиме реального времени. Этим специальным требованиям отвечает SystemC-AMS.

Для изучения книг по SystemC требуется, чтобы читатель владел практическими знаниями C++ и минимальными знаниями аппаратного обеспечения. Для навыков C++ не требуется, чтобы читатель являлся «мастером». Надо, чтобы Вы хорошо знали синтаксис, объектно-ориентированные функции и методы использования C++.

Краткие сведения о языке C++ и объектно-ориентированном программировании приведены в приложении к этой книге.

Чтобы полностью понять примеры, читателю потребуется минимальное понимание цифровой электроники.

1.4. Методология проектирования SystemC

Чтобы понять методологию проектирования SystemC и ее преимущества, важно сначала понять методологию, отличную от SystemC.

В методологии non-SystemC (рис. 1.6) разработчики системы должны написать исполняемые спецификации в C или C++, а затем проверить и отладить этот проект. Обнаружив, что эта конструкция удовлетворяет всем техническим требованиям, она передается в группы разработки RTL. Затем группа проекта RTL перезаписывает проект в RTL, чтобы синтезировать его для вентилей. В такой методологии функциональное RTL-описание иногда зависит от выполняемых спецификаций и, следовательно, становится склонным к ошибке. Кроме того, возникает реальная проблема, если на уровне RTL обнаруживается, что что-то в концептуальной модели не может быть реализовано, так как нет общей среды проектирования между разработкой системы и ее реализацией.

В методологии SystemC разработчику системы нужно только написать модель SystemC. Разработчик может итеративно уточнять исполняемые спецификации вплоть до уровня регистровой передачи, который все еще находится в SystemC до синтеза. Испытательные

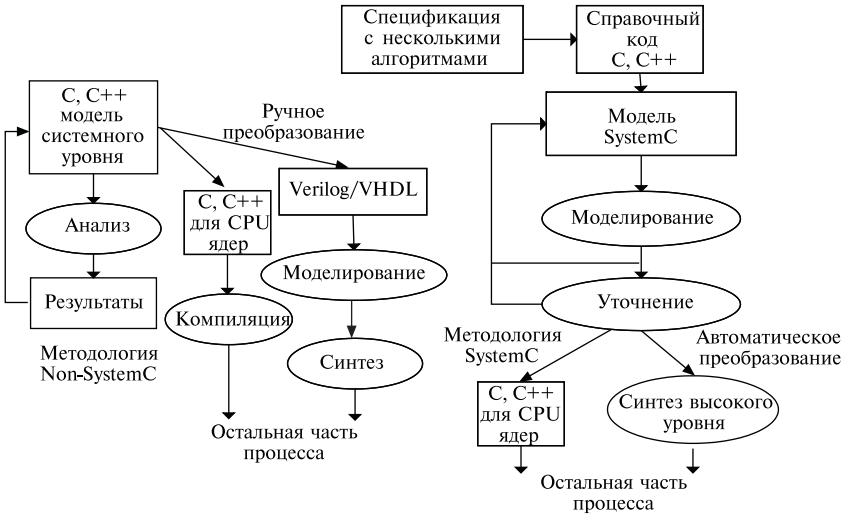

Рис. 1.6. Сравнение методологии проектирования

программы (Testbench) можно использовать повторно, чтобы гарантировать, что итерационный процесс не имеет ошибок. Если во время реализации RTL обнаружено, что что-то концептуально неправильно, гораздо проще его переписать.

SystemC использует следующие типы моделей:

- архитектурная модель системы;
- модель производительности системы;
- модель уровня транзакций (TLM);
- функциональная модель;
- модель передачи на уровне регистров (RTL).

Некоторые проекты SystemC начинаются как функциональные модели, в то время как большая часть кода SystemC делается на уровне TLM. Почти всегда TLM выступает в качестве исполняемой платформы, которая является достаточно точной для запуска программного обеспечения.

Главной причиной использования SystemC является значительное увеличение производительности симуляции на уровне TLM по сравнению с исполняемыми платформами, смоделированными на уровне RT с использованием Verilog или VHDL. Модели SystemC TL достаточно быстры, чтобы служить платформой для разработки программного обеспечения, что позволяет выполнить раннюю разработку программного обеспечения и совместное моделирование аппаратного и программного обеспечения. И модель TL, и функциональная модель достаточно быстры для архитектурного моделирования и анализа на системном уровне.

Типичные числа для модели System on Chip составляют примерно от 1K циклов в секунду до более высоких, ≈300–400K циклов в секунду. Этот диапазон зависит от того, как процессор моделируется в системе. Если использован симулятор набора инструкций (ISS), то производительность обычно составляет от 1 до 10 тыс. циклов. Если процессор моделируется как прямое подключение к системной шине, то производительность переходит в диапазон 100K и выше.

Вторая причина использования SystemC — функциональная проверка. Одна и та же исполняемая платформа, которая используется для разработки программного обеспечения, часто используется и для проверки всей системы. Эта проверка происходит на раннем этапе проекта, и TLM становится прекрасной проверкой для всей системы.

Поскольку SystemC — это C++, у него есть ряд неотъемлемых свойств, таких как классы, шаблоны и наследование, которые поддаются проверке. Эти возможности дополняются SystemC Verification Library (SCV), что делает SystemC мощным языком проверки, а также языком моделирования.

Ключевые характеристики SystemC:

- параллельность — синхронные и асинхронные процессы;
- понятие времени — несколько тактовых генераторов с произвольным фазовым соотношением;
- типы данных — битовые векторы, целые числа с произвольной точностью;
- типы данных фиксированной точки произвольной точности;
- связь — сигналы, каналы;
- расширенные протоколы связи;
- реактивность — просмотр событий;
- поддержка отладки — вывод осциллограмм;
- поддержка моделирования;
- поддержка множества уровней абстракции и итеративной обработки;
- поддержка создания функциональной модели.

Потоки SystemC и использование. На рис. 1.7 показан типичный системный поток SystemC. TLM часто служит как исполняемая платформа для программного обеспечения и как ценная контрольная модель для проверки. Путь к вентилям от TLM либо прямой с использованием поведенческого синтеза, либо TLM служит для ручного перевода либо в SystemC на уровень RTL-модели, либо в HDL (Verilog или VHDL) RTL-модель с последующим синтезом.

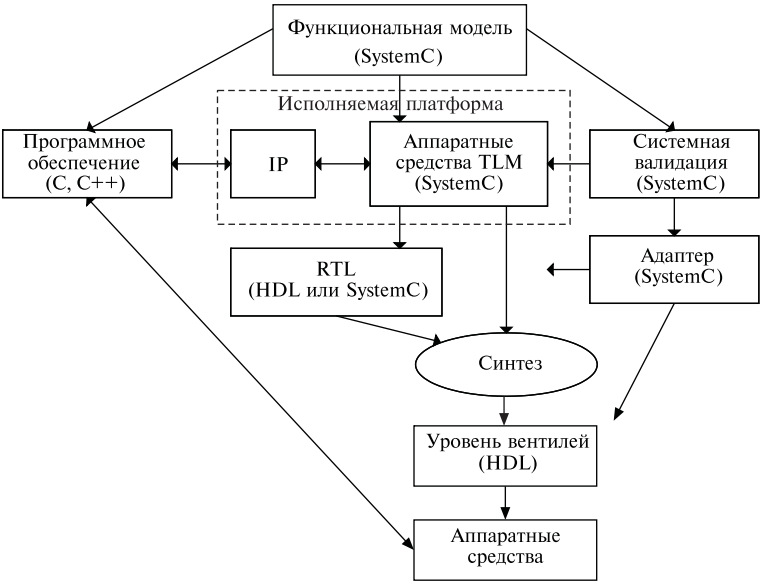

Рис. 1.7. Поток системного проектирования SystemC

Верификация выполняется на уровне TLM. Поскольку TLM переводится на уровни RTL или вентилей (gates), та же самая проверка может быть использована с помощью адаптеров (иногда называемых транзакторами).

Модели вычислений. Наиболее известные модели вычислений в SystemC включают следующие:

- статический многоскоростной поток — включает три фазы: чтение входных данных, выполнение вычисление внутри процесса, вывод всех данных, количество всех данных является известным и неизменным;
- динамический многоскоростной поток;
- сетевой процесс Кана — эффективная модель вычислений для создания алгоритмических моделей в применении к сигнальным процессорам, в которых процессы выполняются параллельно и каналы FIFO имеют определенную длину.

Дискретные события, которые используют:

- RTL-моделирование HW, связанное с цифровой аппаратурой, синхронизированной во времени;
- сетевое моделирование (стохастические или ожидающие модели).

TLM (transaction-level model) — платформа моделирования SoC, основанная на транзакциях, моделирует коммуникации между модулями, используя функции вызова, которые представляют транзакции, типично поддерживаемые целевой платформой. Каналы sc_signal используются вместо данных. Эти сигналы обменяются между различными процессами, путем чтения и записи общих переменных данных.

TLM проекты более краткие и быстрые, чем соответствующие RTL-проекты.

Перечислим основные уровни абстракции и модели использования SystemC:

- функциональное моделирование системных алгоритмов;
- моделирование системных архитектур на уровне транзакций;
- моделирование на уровне RTL и привязка SystemC к маршрутам реализации;
- недавние добавления к SystemC на тему верификации: библиотека SCV верификации SystemC.

SystemC является языком, идеально подходящим для моделирования встроенных систем, и именно тем языком, с помощью которого можно не только описывать сами модели, но и разрабатывать тестовое окружение для этих моделей и использовать его для тестирования реальных плат.

1.5. Стандартные графические обозначения

Для иллюстрации моделей в SystemC применяют стандартные графические обозначения, как показанные на рис. 1.8. Их терминология будет представлена по мере изучения книги.

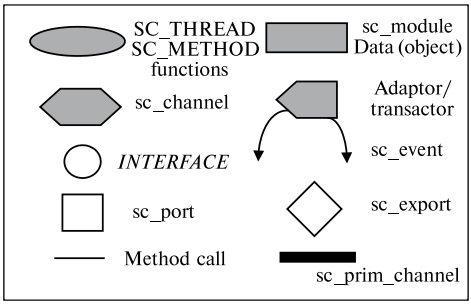

Рис. 1.8. Графические обозначения в SystemC

SystemC использует соглашение об именах, где большинство идентификаторов SystemC имеет префикс sc_ или SC_. Это соглашение зарезервировано для библиотеки SystemC, и Вы не должны использовать эти префиксы в коде конечного пользователя.

1.6. Как изучать эту книгу

Успешно изучить язык программирования Вы сможете, только постоянно подкрепляя новые теоретические сведения самостоятельной практической работой на компьютере. Полезные фрагменты листингов программ будут лучше запоминаться, поиск и устранение ошибок даст Вам навыки работы в интерактивных средах программирования, которые показывают ошибки и дают подсказки, как их исправить.

Поэтому эта книга иллюстрирует теорию множеством практических примеров листингов программ. Причем почти все примеры проверены автором моделированием в средах Eclipse или Microsoft Visual Studio. Скрин-шоты результатов моделирования подтверждают это. Большинство примеров были найдены в руководствах по SystemC от разработчиков из OSCI и разработчиков ASIC (Application-Specific Integrated Circuit — интегральные схемы специального назначения). Многие материалы, найденные мною в Интернете, я перевел на русский и отредактировал. Полагая, что читатели владеют техническим английским языком, я не стал переводить на русский некоторые комментарии в программах и общепринятые термины.

Выполняя важное правило, по которому, приступая к сложной работе, надо подготовить хорошие инструменты и научиться пользоваться ими, во второй главе Вы научитесь устанавливать на своих

компьютерах программные средства: компилятор Cygwin, библиотеки SystemC, среду разработки Eclipce, программу GTKWave для отображения результатов моделирования. В Приложении Б показано, как установить и настроить для SystemC среду разработки Microsoft Visual Studio.

В третьей главе Вы изучите основы языка SystemC-2.3.1. Все теоретические сведения подкрепляются практическими примерами программ, и я рекомендую Вам самостоятельно выполнить моделирование и добиться совпадения результатов с приведенными в книге.

Четвертая глава содержит более сложные практические примеры моделирования, в которых в комплексе применены сведения о SystemC, полученные в третьей главе.

Пятая глава посвящена моделированию на уровне транзакций с использованием стандарта TLM-2.0. Этот метод применяют для разработки на системном уровне с исследованием передач между моделями и объектами сложной системы.

Я надеюсь, что эта книга станет для Вас первым шагом в освоении сложной современной технологии проектирования «систем на кристалле».

Желаю Вам успехов!

Профессор *В.А. Алехин*