ISSN 2500-316X (Online)

https://doi.org/10.32362/2500-316X-2020-8-4-79-95

УДК 004.434; 004.415.2

Проектирование электронных систем с использованием SystemC и SystemC–AMS

В.А. Алехин

МИРЭА – Российский технологический университет, Москва 119454, Россия
@Автор для переписки, e-mail: alekhin@mirea.ru

Современные тенденции в проектировании электронных систем и устройств заключаются в применении встраиваемых систем на основе «систем на кристалле» (System-on-Chip (SoC)) или (СБИС СнК). В работе рассмотрены особенности проектирования электронных систем на кристалле с использованием языка проектирования и верификации SystemC. Для совместного проектирования и моделирования аппаратно-программного обеспечения цифровых систем представлены и обсуждаются семь уровней моделирования: исполняемая спецификация, отключенная функциональная модель, временная функциональная модель, модель на уровне транзакций, поведенческая аппаратная модель, точная аппаратная модель, модель регистровых передач. Изложена методология проектирования SystemC с функциональной проверкой, сокращающая сроки разработки. Показаны архитектура языка SystemC и его главные компоненты.

Рассмотрено расширение SystemC–AMS для аналоговых и смешанных аналого-цифровых сигналов и варианты его использования в проектировании электронных систем. Обсуждаются модели вычислений: временной поток данных (TDF), линейный поток сигналов (LSF) и электрические линейные сети (ELN). Представлена архитектура стандарта языка SystemC–AMS и приведены примеры его применения.

Показано, что языки проектирования SystemC и SystemC–AMS широко применяются ведущими разработчиками систем автоматизированного проектирования электронных устройств.

Ключевые слова: системы на кристалле, проектирование электронных систем, моделирование, SystemC, SystemC–AMS.

Для цитирования: Алехин В.А. Проектирование электронных систем с использованием SystemC и SystemC–AMS. *Российский технологический журнал.* 2020;8(4):79-95. https://doi.org/10.32362/2500-316X-2020-8-4-79-95

Designing Electronic Systems Using SystemC and SystemC–AMS

Vladimir A. Alekhin

MIREA - Russian Technological University, Moscow 119454, Russia
@Corresponding author, e-mail: alekhin@mirea.ru

Current trends in the design of electronic systems is the use of embedded systems based on systems on a chip (System-on-Chip (SoC)) or (VLSI SoC). The paper discusses the design features of electronic systems on a chip using the SystemC design and verification language. For the joint design and simulation of digital systems hardware and software, seven modeling levels are presented and discussed: executable specification, disabled functional model, temporary functional model, transaction-level model, behavioral hardware model, accurate hardware model, register transfer model. The SystemC design methodology with functional verification is presented, which reduces development time. The architecture of the SystemC language and its main components are shown.

The expansion of SystemC–AMS for analog and mixed analog-digital signals and its use cases in the design of electronic systems are considered. Computing models are discussed: temporary data stream (TDF), linear signal stream (LSF) and electric linear networks (ELN). The architecture of the SystemC–AMS language standard is shown and examples of its application are given.

It is shown that the design languages SystemC and SystemC–AMS are widely used by leading developers of computer-aided design systems for electronic devices.

Keywords: systems on a chip, design of electronic systems, modeling, SystemC, SystemC–AMS.

For citation: Alekhin V.A. Designing Electronic Systems Using SystemC and SystemC–AMS. *Rossiiskii tekhnologicheskii zhurnal = Russian Technological Journal.* 2020;8(4):79-95 (in Russ.). https://doi.org/10.32362/2500-316X-2020-8-4-79-95

Введение

В настоящее время широко развивается проектирование встраиваемых электронных систем на основе «систем на кристалле» (System-on-Chip (SoC)) или (СБИС СнК). Они содержат встроенный процессор (процессоры), аппаратные ускорители (или IP-ядра – Intellectual Property), встроенную память, коммуникационные интерфейсы и другие цифровые и аналоговые блоки. К устройствам на основе СнК предъявляют жесткие требования по стоимости, производительности, качеству, безопасности.

В последние несколько лет растет потребность в том, чтобы разместить традиционные микропроцессоры, память и периферийные устройства – все в одной микросхеме. Появление этой тенденции было отмечено как начало эпохи SoC и привело к разработке новых программных средств проектирования электронных устройств.

Целью данной работы является анализ возможностей перспективных средств проектирования систем на кристалле SystemC и SystemC–AMS. Эти вопросы обсуждаются в многочисленных работах зарубежных авторов и в работах автора [1, 2]. Для изучения методологии SystemC в [1] рассмотрены вопросы практического использования языка для моделирования сложных электронных систем на разных уровнях абстракции: системное описание, уровень транзакций, уровень регистровых передач и т.д. Изложена методика установки библиотек, дано подробное описание языка и многочисленные примеры программ с решениями в средах Eclipse и Microsoft Visual Studio.

Особенности проектирования системы на кристалле

Одна из ключевых задач проекта SoC – разбиение системной функциональности на аппаратное обеспечение (HW) и программное обеспечение (SW) или совместно HW/SW [3].

Сопряжённое или совместное проектирование аппаратуры и ПО, представляет собой процесс параллельной и скоординированной разработки, который называют «co-design».

В текущей методологии систем автоматизированного проектирования (САПР) делается априорное разделение на аппаратные и программные средства, 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 (программируемые логические интегральные схемы) 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).

Еще одним критическим недостатком текущей методологии САПР стало то, что она является RTL ориентированной (register transfer level) – в связи с увеличением сложности схемы время моделирования возрастает и постепенно становится неприемлемым.

Для сокращения времени разработки и затрат компьютерные инструменты проектирования (САПР) теперь должны включать проектирование на уровне абстракции электронной системы (ESL – Electronic System Level).

С этой целью Открытой инициативой SystemC (Open SystemC Initiative – OSCI) был введен Стандарт SystemC 2.0 [4]. В настоящее время распространяются версия SystemC-2.3.2, выпущенная в 2017 году [5], и SystemC-2.3.3 (2019 г.). Созданы новые расширения языка: SystemC-Verification-2.0 (библиотека для верификации систем) [6] и SystemC-AMS-2.0 [7, 8] (для моделирования систем с аналоговыми и смешанными сигналами).

Разработки языка SystemC спонсируют крупнейшие производители САПР для электроники: Synopsys, Mentor Grapfics, Cadence.

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

В связи с быстро возрастающей сложностью конструкции и ростом стоимости ошибки или отказа разработчикам системы в большинстве областей продукции требуется подход проектирования сверху вниз, но с улучшенной методологией.

Несколько позже SystemC возникла методология моделирования на уровне транзакций (Transaction-level model – TLM), которая используется в SystemC [9, 10] и существенно повышает скорость моделирования на ранних этапах разработки.

Архитектура языка SystemC

SystemC обращается к моделированию программного и аппаратного обеспечения, используя язык C++. Диаграмма (рис. 1) иллюстрирует основные компоненты SystemC, которые базируются на стандартном языке C++. Поскольку C++ уже решает большинство задач программного обеспечения SystemC фокусируется прежде всего на проблемах, свя-

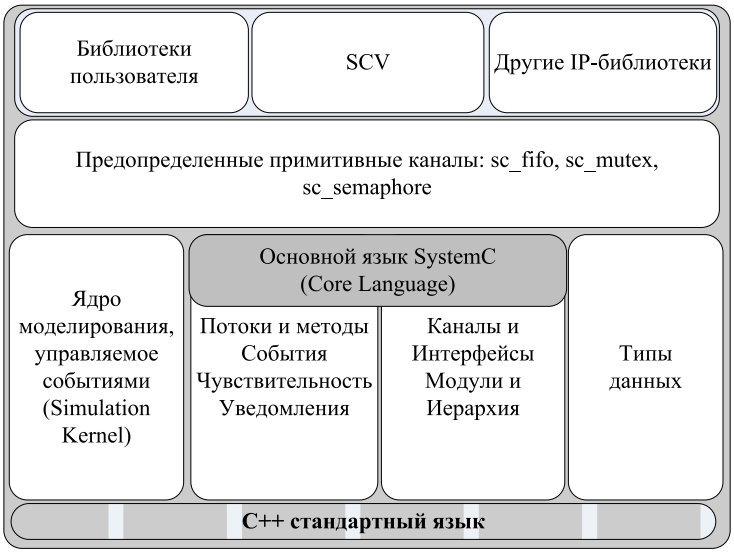

Рис. 1. Архитектура и главные компоненты языка SystemC.

занных как с программным обеспечением, так и с аппаратной реализацией. Основной областью применения SystemC является разработка электронных систем, но SystemC применяется к неэлектронным системам. Имеются публикации по моделированию электромеханических систем, датчиков, систем геологоразведки и пр.

Ядро моделирования (Kernel) и главная программа

Рассмотрим работу ядра моделирования Simulation Kernel.

Принцип работы ядра моделирования SystemC схож с языками VHDL и Verilog. При использовании Verilog и VHDL проходит некоторое время между инициализацией кода и началом моделирования. В SystemC, также как и в C/C++, есть строго определённая точка входа в программу. В случае SystemC – это главная программа sc_main().

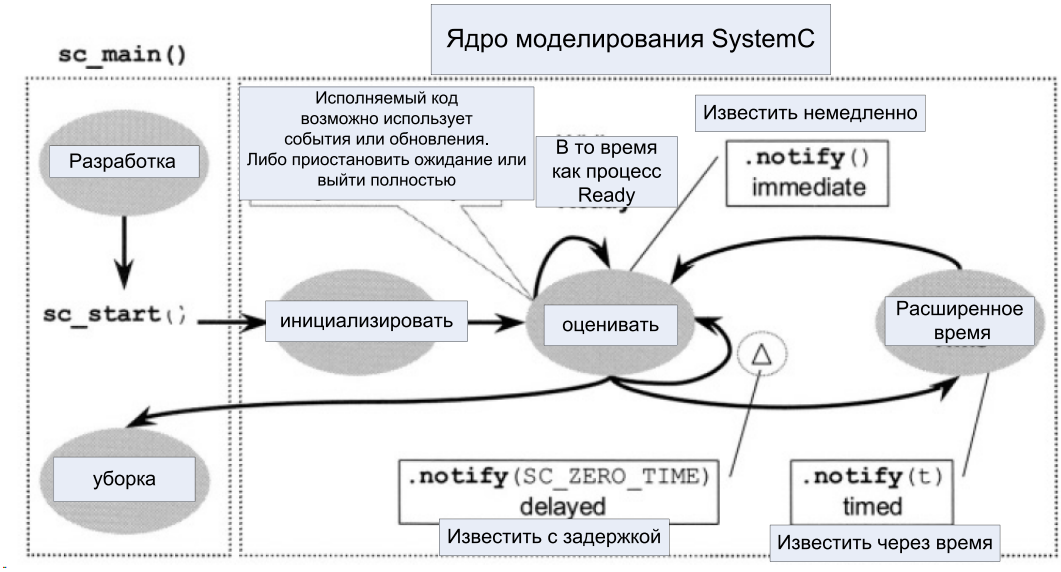

Рис. 2. Работа ядра моделирования SystemC.

Исполнение команд перед вызовом функции в sc_start () относится к этапу разработки. Эта фаза характеризуется инициализацией структур данных, установлением соединений, а также подготовкой ко второму этапу – выполнению. Контроль фазы выполнения передан ядру моделирования SystemC, которое руководит выполнением процессов, чтобы создать иллюзию параллелизма.

После команды sc_start () (рис. 2) все процессы моделирования (кроме нескольких исключений) вызываются случайным образом во время инициализации. После инициализации запускается процесс моделирования, когда происходит событие, к которому процесс чувствителен. Несколько процессов моделирования могут начаться в один и тот же момент времени в симуляторе. В связи с этим все процессы моделирования оцениваются, и затем их данные на их выходах обновляются. Оценки данных с последующим обновлением называются термином «дельта-цикл». Если нет никаких дополнительных процессов моделирования, которые следует оценить в настоящее время (в результате обновления), то время моделирования продвигается вперед. При этом, если не требуется запускать дополнительные процессы, когда отработаны все события, к которым чувствительны процессы, моделирование заканчивается.

Состав ядра языка SystemC (Core Language)

Ядро языка SystemC (Core Language), как показано на рис. 1, кроме ядра моделирования включает в себя следующие компоненты: модули, порты, процессы, события, интерфейсы, каналы. Эти компоненты оперируют с различными типами данных, как правило, аналогичными языку C++.

Основные понятия ядра языка SystemC классифицируются по следующей терминологии:

Основные термины

Method	Метод C++, функция член класса
Module	Конструктивный субъект, который может содержать процессы, порты, каналы и другие модули. Модули позволяют выразить структурную иерархию
Interface	Интерфейс предоставляет набор объявлений методов, но не дает никаких реализаций метода и нет полей данных
Channel	Канал реализует один или несколько интерфейсов, а также служит в качестве контейнера для функциональных возможностей связи
Port	Порт представляет собой объект, через который модуль может получить доступ к его каналу через интерфейс. Модули могут также получить доступ к интерфейсу канала напрямую
Primitive Channel	Примитивный канал является атомарным, он не содержит процессы или модули, и он не может непосредственно получить доступ к другим каналам
Hierarchical Channel	Иерархический канал представляет собой модуль, он может содержать процессы и другие модули, и он может непосредственно получить доступ к другим каналам
Event	Событие. Процесс может приостанавливаться событием или быть чувствительным к одному или нескольким событиям. События позволяют возобновить и активизировать процессы
Sensitivity	Чувствительность процесса определяет, когда этот процесс будет возобновлен или активирован. Процесс может быть чувствительным к набору событий. Всякий раз, когда одно из соответствующих событий инициируется, процесс возобновляется или активируется
Static Sensitivity	Чувствительность процесса объявляется статически во время разработки и не может быть изменена после начала моделирования. Так называемый список чувствительности используется для определения статического набора событий
Dynamic Sensitivity	Чувствительность процесса может быть изменена во время моделирования
IMC	Метод вызова интерфейса
RPC	Удаленный вызов процедур

Терминология процессов

Процессы играют центральную роль в SystemC. Они описывают функциональные возможности системы и позволяют выразить параллелизм в системе. Процессы содержатся в модулях и имеют доступ к интерфейсам внешнего канала через порты модуля. Существуют различные типы процессов и различные способы активации процессов. Объясним термины, связанные с процессами.

Thread	Поток SystemC имеет свой собственный поток исполнения, но он не является упреждающим
Automatically activated	Некоторые методы модуля (процессы) активируются автоматически, когда происходят события, к которым процессы чувствительны
Explicitly activated	Некоторые методы модуля, которые должны быть вызваны явно другим кодом, чтобы активироваться
wait()	Метод, который приостанавливает выполнение потока. Аргументы, передаваемые wait(), определяют, когда выполнение потока возобновляется
SC_THREAD	Модуль-способ, который имеет свой собственный поток исполнения и который может вызвать код, инициирующий ожидание (). SC_THREADs автоматически активируются. Также известен как процесс потока
SC_METHOD	Модуль метод, который не имеет своего собственного потока исполнения, и который не может вызвать код, инициирующий wait(). SC_METHODs автоматически активируется. Также известен как процесс метода
SC_CTHREAD	Модуль метод, который имеет свой собственный поток исполнения и который в его списке чувствительности имеет только положительное или отрицательное событие по фронту тактового импульса. Он может вызвать код wait() с ограниченным списком аргументов. SC_CTHREADs активируется автоматически. Также известен как процесс с тактовой частотой

Следующей более высокой надстройкой являются понятия элементарных каналов.

На вершине архитектуры языка SystemC добавлены более специфические модели вычислений, библиотеки проектирования, руководства по моделированию, методология проектирования, которые полезны при проектировании.

Нижний слой подчеркивает то, что SystemC создан полностью на языке C++. Это означает, что любая программа, написанная на SystemC, может быть преобразована компилятором C++ в исполняемую программу.

Модули SC_MODULE

Основной единицей проектирования является модуль SC_MODULE.

Сложные системы состоят из множества независимо функционирующих компонентов. Они могут представлять собой аппаратные средства, программное обеспечение или любой объект, могут быть большими или маленькими. Компоненты часто содержат иерархии более мелких. Самые маленькие компоненты представляют поведение и состояние. В SystemC для представления компонентов используют концепцию, известную как SC_MODULE.

Определение: Модуль – это конструктивный субъект, который может содержать процессы, порты, каналы и другие модули. Модули позволяют выразить структурную иерархию.

Модуль SystemC является самым маленьким контейнером функциональности с состоянием, поведением и структурой для иерархического подключения.

Модуль SystemC соответствует определению класса C++. Как правило, макрос SC_MODULE используется для объявления класса:

```
#include <systemc.h>
SC_MODULE(Adder) {
//Тело модуля
//порты, процессы, внутренние данные и т.д.
SC_CTOR(Adder){
//Тело конструктора
//объявление процессов, чувствительностей и т.д.
}
};
```

Отметим, что SC_MODULE – это простой макрос C++ и на языке C++ его можно определить так:

```
#define SC_MODULE (module_name)
struct module_name: public sc_module
```

Модуль является старшим в иерархии элементов SystemC. Наличие модуля позволяет строить SystemC модели в соответствии с имеющимися у разработчиков аппаратуры представлениями (документацией) об архитектуре и функционировании будущего изделия. Обычно каждый отдельно взятый модуль представляет в модели функционально законченный узел разрабатываемого изделия.

Модули объявляются с ключевым словом SC_MODULE.

На рис. 3 изображён модуль, который включает в себя несколько процессов.

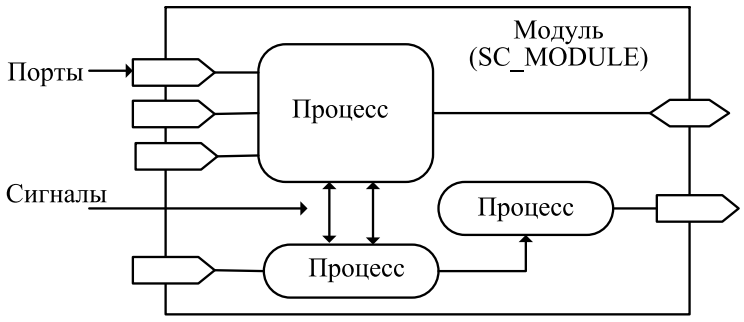

Рис. 3. Модуль, содержащий несколько процессов.

Структура модуля

```
// Заголовочный файл
SC_MODULE(module_name) {
Объявление портов
Объявление локальных каналов
Объявление переменных
Объявление процессов
Объявление других методов

Конкретизации модуля
SC_CTOR(module_name){
```

Регистрация процесса
Лист статической чувствительности
Инициализация переменных модуля
Связывание экземпляра модуля / канала}
} ;

Коммуникации в SystemC

Рассмотрим способы подключения портов, каналов, модулей и процессов. Диаграмма (рис. 4) иллюстрирует типы соединения, которые возможны с SystemC.

Вначале рассмотрим фрагменты по имени, а затем обсудим правила их взаимосвязи.

Имеется три модуля, изображенные прямоугольниками. Внешний охватывающий экземпляр модуля имеет имя top. Два экземпляра подмодуля внутри top называются M1 mi1 и M2 mi2.

Каждый из модулей имеет один или несколько портов, представленных квадратами p. Портами модуля top являются p1, p2, p3, p4, p5 и p6, которые используют интерфейсы с именами if1, if2, if3, if4, if5 и if6, соответственно.

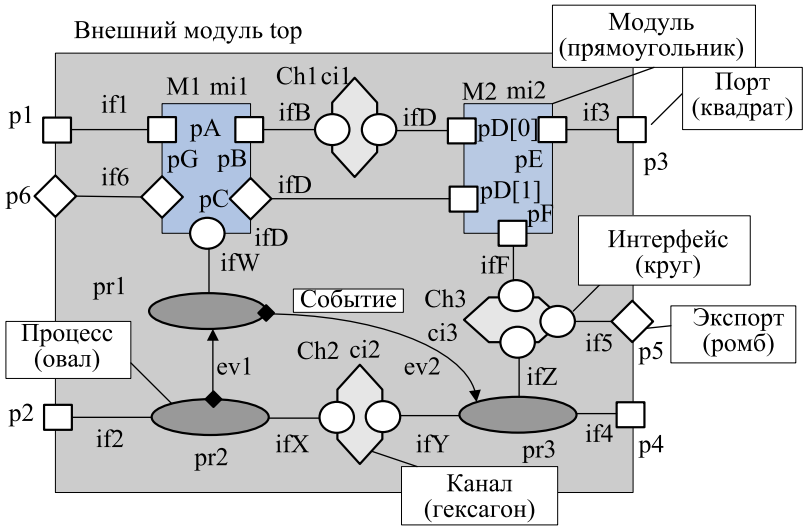

Рис. 4. Возможности подключения портов, каналов, модулей и процессов.

Портами для модуля M1 mi1 являются pA, pB, pC и pG, которые подключены к интерфейсам, названным if1, ifB, ifD и if6, соответственно.

Модуль M1 mi1 также предоставляет интерфейсы ifW и if6.

Порты для модуля M2 mi2 представляют собой pD[0], pD[1], pE и pF, которые подключены к интерфейсам с именами if3, ifD и ifF, соответственно. Имеется три экземпляра каналов, представленных гексагональными формами, они существуют внутри модуля top и называются c1i, c2i и c3i. Каждый канал реализует один или несколько интерфейсов, представленных кружками и стрелками. Стрелка предназначена для указания возможности вызова, возвращающего значение. Канал c1i реализует интерфейсы ifB и ifD. Канал c2i реализует интерфейсы ifX и ifY. Канал c3i реализует интерфейсы if5, ifF и ifZ.

Существуют три процесса с именами pr1, pr2 и pr3. Есть два явных события ev1 и ev2, используемые для сигнализации между процессами. На диаграмме рис. 4 можно

наблюдать несколько правил. Процессы могут связываться с процессами:

– на том же уровне через каналы, через синхронизацию, через события;

– вне локального модуля проектирования через порты, привязанные к каналам посредством интерфейсов;

– в экземплярах подмодулей через интерфейсы к каналам, подключенным к подмодулю портов или посредством интерфейсов через сам модуль sc_export; любая другая попытка межпроцессного общения либо запрещена, либо опасна и приводит к ошибкам.

Порты могут подключаться через интерфейсы только к локальным каналам, портам подмодулей или косвенно к портам процессов. Есть несколько интересных особенностей. Во-первых, модуль экземпляр mi1 реализует интерфейс ifW. Во-вторых, порт pD представляется массивом размера 2. Наконец, порт p5 и порт pC иллюстрируют sc_export.

Уровни моделирования в SystemC

В SystemC используют семь уровней моделирования [11], представленных на рис. 5.

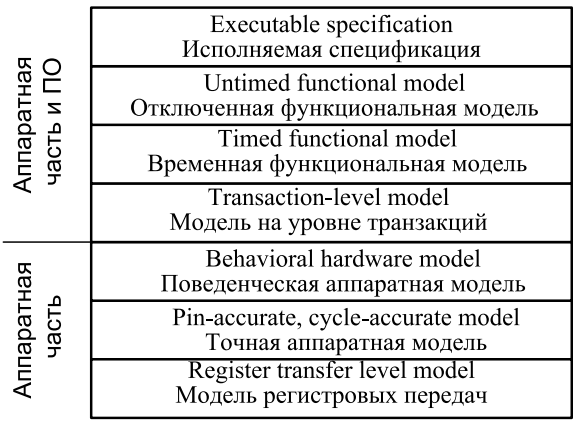

Рис. 5. Уровни моделирования в SystemC.

1. ***Executable specification*** (исполняемая спецификация) прямо передаёт спецификацию проекта в SystemC. Временные задержки учитываются в исполняемой модели.

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

3. ***Timed functional model*** применяют на ранних стадиях анализа аппаратного и программного обеспечения. Задержки добавлены к процессам и отражают особенности функционирования модели.

4. ***Transaction-level model (TLM)***. В этой модели связи между модулями смоделированы с использованием функций вызовов.

Термин ***platform transaction-level model*** показывает, что модель использует TLM стиль с целью моделирования инфраструктурных связей в платформе SoC. Этот метод является более качественным, точным и эффективным путём моделирования взаимосвязи HW и SW на самом раннем этапе проектирования.

Модели для аппаратной части:

5. ***Behavioral hardware model*** – 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*** – 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 (RTL) model*** имеет на своих границах контактную и тактовую точность. Внутренняя структура RTL модели точно отражает регистры и комбинационную логику целевой реализации.

Методология проектирования SystemC

Типичную методологию моделирования в среде SystemC представлена рис. 6.

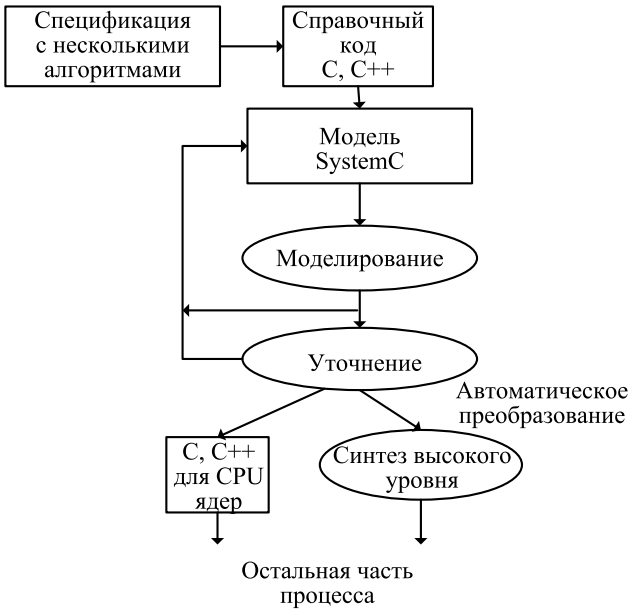

Рис. 6. Методология проектирования в SystemC.

Дизайнер пишет модели SystemC на системном, поведенческом уровне или на уровне RTL с использованием C/C++ с библиотекой классов SystemC. Библиотека классов обслуживает две важные цели. Во первых, возможность реализации многих типов объектов, которые зависят от аппаратного обеспечения, такие как параллельные и иерархические модули, порты и такты. Во-вторых, она содержит ядро для планирования процессов. Пользовательский код SystemC можно скомпилировать и связать вместе с библиотекой классов с любым стандартным компилятором C++ (например, GNU GCC). Полученный исполняемый файл служит симулятором пользовательского дизайна.

Испытательный стенд для проверки правильности конструкции также будет написан в SystemC и скомпилирован вместе с проектом.

Исполняемый файл может быть отлажен в любой среде отладки C++. Кроме того, файлы трассировки также могут генерироваться для просмотра истории выбранных сигналов с использованием стандартного инструмента отображения формы сигнала.

Поскольку SystemC – это надстройка к C++, у него есть ряд неотъемлемых свойств, таких как классы, шаблоны и наследование, которые поддаются проверке. Эти возможности дополняются SystemC Verification Library (SCV) [12], что делает SystemC мощным языком проверки, а также языком моделирования.

Использование SystemC для RTL синтеза устройств

SystemC компилятор синтезирует SystemC RTL-модули или проекты с интегрированными RTL- и поведенческими модулями в список соединений на уровне затворов. Он также может синтезировать системный модуль SystemC в RTL или Netlist. После синтеза вы можете использовать этот список соединений в качестве входных данных для других инструментов компании Synopsys, такие как Design Compiler и Physical Compiler.

Технология Synopsys дает усовершенствованные кремниевые чипы, которые являются более умными благодаря специальному программному обеспечению, которое ими управляет, и помогает заказчикам внедрять инновации.

SystemC Compiler – это инструмент, который может принимать как поведенческие, так и RTL-модели SystemC и выполнять поведенческий или RTL-синтез, так как необходимо создать список соединений на уровне затворов. На рис. 7 показаны процессы поведенческого синтеза и RTL-синтеза для описания на уровне затворов с помощью SystemC Compiler.

Высокоуровневая система SystemC может содержать абстрактные порты, типы которых не могут быть переведены на оборудование. Для каждого абстрактного порта необходимо определить порт или набор портов для замены каждого терминала абстрактного порта и заменить все обращения к абстрактному порту или терминалу с доступом к вновь определенным портам.

Для компилятора SystemC требуется иметь описание SystemC RTL, технологическую библиотеку и синтетическую библиотеку.

На рис. 8 показан процесс синтеза в компиляторе SystemC и выход из него.

Описание SystemC RTL проводят, используя SystemC Class Library. Технологическая библиотека предоставляется поставщиком ASIC в Synopsys .db Format базы данных. Поставщик предоставляет области, сроки, модели загрузки соединений и условия эксплуатации.

Синтетическая библиотека является технологически независимой библиотекой логики и включает такие компоненты, как сумматоры и множители. Компилятор SystemC сопоставляет проектные операторы с синтетической библиотекой логических компонентов.

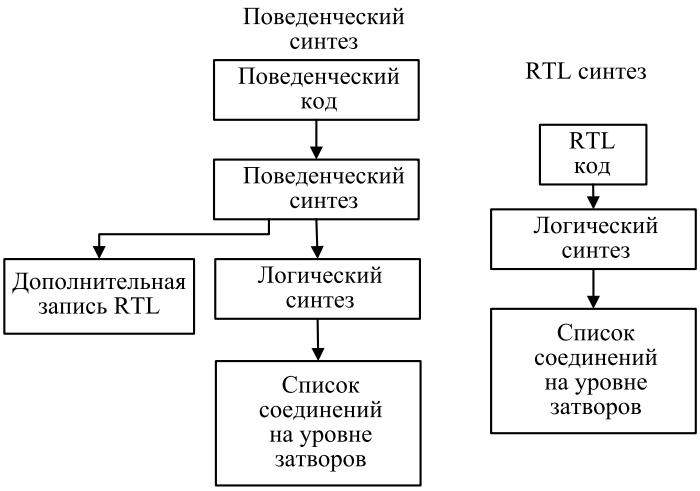

Рис. 7. Сравнение поведенческого синтеза и RTL-синтеза.

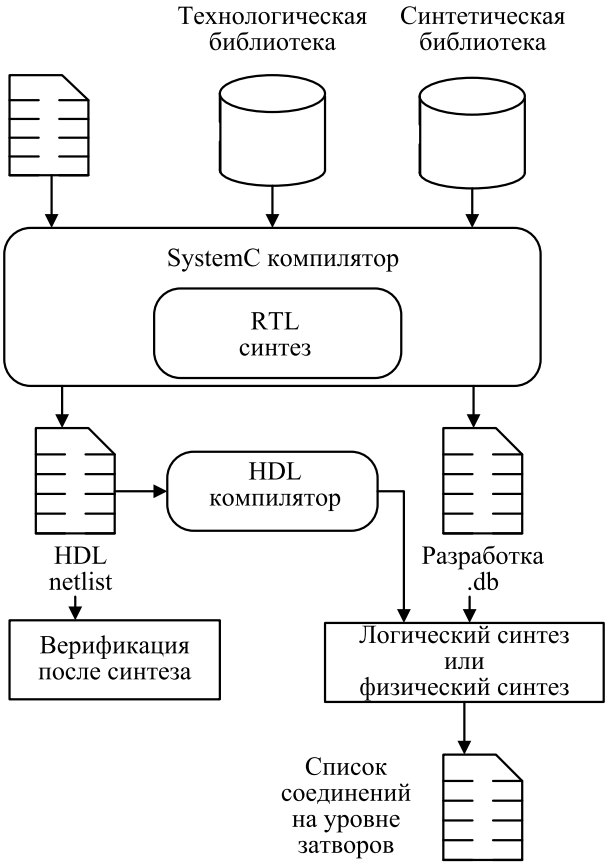

Рис. 8. Процесс RTL-синтеза в SystemC.

Дизайнер предоставляет путь к выбранным синтетическим библиотекам для своего дизайна, определяя переменную синтетическую библиотеку в dc_shell.

SystemC Compiler создает расширенный .db-файл для ввода в инструмент проектирования компилятора. Он также генерирует RTL файлы HDL (например, Verilog), которые могут использоваться в потоках на основе HDL.

Последние версии языка SystemC предназначены для моделирования и проектирования цифровых и микропроцессорных устройств. В первых публикациях [12–15] приведены примеры моделирования логических цифровых устройств, триггеров, счетчиков, регистров, конечных автоматов. В [16] описаны модели счетчиков в коде Грея, декодеров Джонсона.

SystemC–AMS для аналоговых и смешанных сигналов

Многие технические системы состоят из цифровых и аналоговых подсистем, в которых цифровые подсистемы контролируются программным обеспечением. Моделирование смешанного сигнала, т.е. комбинированное цифровое и аналоговое моделирование имеет очень важное значение при разработке таких гетерогенных систем. К сожалению, имитаторы смешанных сигналов на несколько порядков медленнее, чем эффективное системное моделирование. Кроме того, совместное моделирование оборудования со смешанным сигналом со сложным программным обеспечением, обычно написанным на C или C++, поддерживается недостаточно.

Перспективным подходом к преодолению этих трудностей является применение SystemC и его расширения для аналоговых и смешанных сигналов SystemC–AMS, созданные для обеспечения единой и стандартизированной методологии для моделирования систем E AMS [17, 18].

Варианты использования SystemC–AMS и требования

Как показано на рис. 9, расширения SystemC–AMS могут использоваться для самых разных вариантов, таких как исполняемая спецификация, виртуальное прототипирование, изучение архитектуры, проверка интеграции.

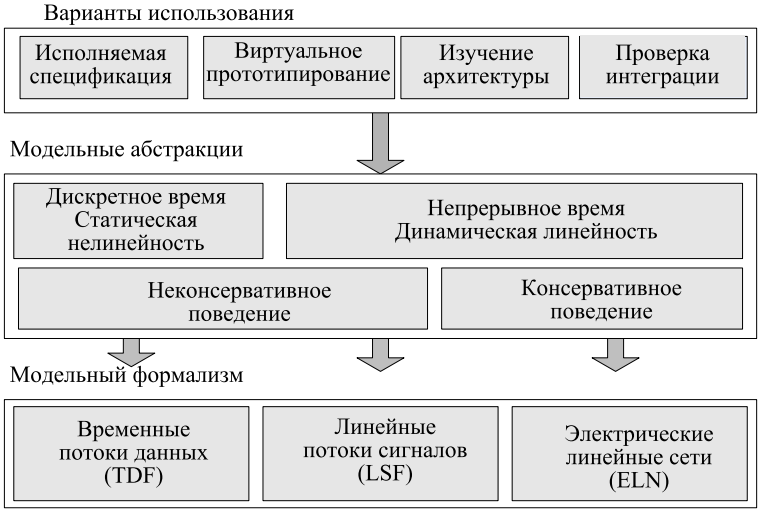

Рис. 9. Варианты использования SystemC–AMS, модельные абстракции и модельный формализм.

Исполняемая спецификация выполняется для проверки правильности требований к системе посредством создания исполняемого описания системы с помощью моделирования. Для этого варианта используются модели на высоком уровне абстракции, которые необязательно должны быть связаны с физической архитектурой или внедрением системы. Поэтому они называются функциональными или алгоритмическими.

Виртуальное прототипирование направлено на предоставление разработчикам программного обеспечения высокоуровневых или временных моделей, которые представляют собой аппаратную архитектуру и обеспечивают высокую скорость моделирования. Специально для E-AMS систем, в которых программное обеспечение или встроенное ПО взаимодействуют напрямую с оборудованием AMS, обеспечена совместимость с использованием расширения SystemC Transaction-Level Modeling (TLM). Подсистема AMS может моделировать часть виртуального прототипа для дальнейшей разработки подсистемы HW/SW.

После определения архитектуры и проектирования аналоговых и цифровых компонентов HW/SW эти компоненты будут интегрированы, и их правильность проверяется в рамках всей системы. Интерфейсы и типы данных, используемые в модели, должны соответствовать физической реализации. Для аналоговых схем это относится к электрическим узлам. Для цифровых схем это относится к точности контактов на шинах. Для систем HW/SW могут подходить интерфейсы TLM.

Модели вычислений в SystemC–AMS

SystemC–AMS определяют основные формализмы моделирования, необходимые для поддержки AMS поведенческого моделирования на разных уровнях абстракции. Эти формализмы моделирования реализуются с использованием различных моделей вычислений: временной поток данных (TDF), линейный поток сигналов (LSF) и электрические линейные сети (ELN).

А. *Временной поток данных (TDF).* Семантика выполнения на основе TDF представляет собой моделирование непрерывного и дискретного времени без дополнительных расходов на динамическое планирование, наложенное ядром дискретного события SystemC. Моделирование ускоряется путем определения статического графика, который вычисляется до начала моделирования и который выполняется функцией обработки запланированных модулей TDF в соответствии с направлением потока данных. Образцы дискретного времени, которые распространяются через модули TDF, могут представлять любой C++ тип. Если используется, например, вещественный тип, такой как double, сигнал TDF может представлять собой напряжение или ток на данный момент времени.

Б. *Линейный поток сигналов (LSF).* Формализм линейного сигнала поддерживает моделирование непрерывного поведения, предлагая согласованный набор примитивных модулей, таких как сложение, умножение, интеграция или задержка. Модель LSF состоит из соединения таких примитивов посредством сигналов вещественной временной области, представляющих любые виды непрерывного количества. Модель LSF определяет систему линейных уравнений, которая решается линейным вычислителем DAE.

В. *Электрические линейные сети (ELN).* Моделирование электрических сетей поддерживается путем создания заданных линейных сетевых примитивов, таких как резисторы или конденсаторы, которые используются в качестве макромоделей для описания непрерывных отношений между напряжениями и токами. Доступен ограниченный набор линейных примитивов и переключателей для моделирования энергосберегающего поведения.

Архитектура стандарта языка AMS

На рис. 10 показана архитектура стандарта языка AMS [17, 18].

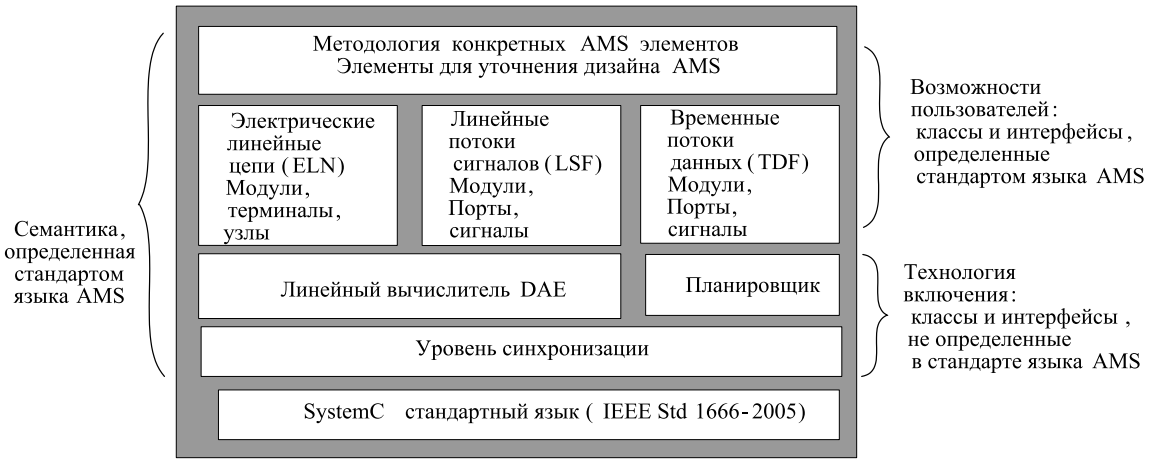

Рис. 10. Архитектура стандарта языка AMS.

Расширения SystemC–AMS полностью совместимы со стандартом языка SystemC. Стандарт языка AMS определяет семантику выполнения вычислений для моделей TDF, LSF и ELN и дает представление о основополагающих технологиях, таких как линейный вычислитель, планировщик и уровень синхронизации. В настоящее время включение интерфейсов и определение классов для этих основополагающих технологий определяется реализацией. Разработчик AMS (конечный пользователь) может использовать классы и интерфейсы для создания моделей TDF, LSF или ELN с использованием предопределенных модулей, портов, терминалов, сигналов и узлов.

Примеры применения SystemC–AMS

В [19, 20] приведены примеры использования SystemC–AMS для моделирования активных фильтров на операционных усилителях, аналого-цифрового преобразователя с интегрированием входного сигнала, квадратурного фазового модулятора с цифро-аналоговым преобразователем, дельта-игма модулятора, фильтра Баттерворта 5-го порядка, который моделируется как линейная электрическая цепь, сигма-дельта модулятора с полосовым фильтром, позиционно-чувствительного детектора, считывателя CD-ROM и т.п.

В [21] отмечается, что задачей разработки систем E-AMS является понимание взаимодействия между HW/SW и подсистемами аналогового и смешанного сигналов на уровне архитектуры. Это требует некоторых средств моделирования взаимодействующих аналоговых/смешанных сигналов и систем HW/SW на функциональном и архитектурном уровнях. В [21] приведена спецификация простого бинарного приемника с двоичной амплитудной манипуляцией (BASK), состоящего из микшера, фильтра нижних частот и демодулятора BASK, и рассмотрена последовательность проектирования устройства.

Для подготовки дизайна и проверки интеграции всех подсистем должны быть точно смоделированы интерфейсы и типы данных, используемые в моделях. Для аналоговых схем это относится к электрическим узлам. Для цифровых схем – к точному выводу на шины. Для систем HW/SW интерфейсы TLM могут быть наиболее подходящими. После определения и проектирования аналоговых, цифровых компонентов HW и SW эти компоненты будут интегрированы, и их правильность проверяется в общей системе.

Виртуальное прототипирование предоставляет разработчикам программного обеспечения высокоуровневую или временную модель, которая представляет собой аппаратную архитектуру. Для систем E-AMS, где программное обеспечение взаимодействует с оборудованием AMS, особенно важно взаимодействие с расширениями SystemC TLM.

Заключение

Язык разработки SystemC в настоящее время широко применяется для разных уровней проектирования и верификации цифровых систем на кристалле.

SystemC–AMS расширяют доступный стандарт SystemC для поддержки линейных электрических сетей, линейного потока сигналов и временных моделей потока данных, чтобы эффективно моделировать аналоговые архитектуры смешанного сигнала. Новые языковые конструкции поддерживают создание аналоговых и смешанных сигналов на более высоких уровнях абстракции, вводя функциональное и архитектурное моделирование. Это позволяет использовать методологию уточнения дизайна для исполняемой

спецификации, исследования архитектуры, проверки интеграции и виртуального прототипирования систем E-AMS. Расширение SystemC–AMC поддерживает особые требования, например, для моделирования нелинейных или радиочастотных систем.

Расширение уникальных возможностей SystemC с новыми функциями AMS предлагает мощную систему моделирования и симуляции, позволяющую разрабатывать и проверять широкий спектр приложений и систем.

Литература:

1. Алехин В.А. SystemC. Моделирование электронных систем. Учебное пособие для вузов. М.: Горячая линия – Телеком, 2018. 320 с.

2. Алехин В.А., Быков И.А. Применение SystemC для проектирования электронных систем. Сборник трудов Международной научно-практической конференции «Наука, образование, общество». Тамбов, 31 марта 2018. С. 7-11.

3. Devalapalli S. Development of SystemC Modules from HDL for System-on-Chip Applications. Master's Thesis, University of Tennessee, 2004.
URL: http://trace.tennessee.edu/utk_gradthes/2119

4. Functional specification for SystemC 2.0. April 5, 2002.
[Электронный ресурс]. http://www.systemc.org

5. Technical Tutorial: "SystemC Design and Verification – Solidifying the Abstraction Above RTL". Feb. 2, 2017. [Электронный ресурс]. http://videos.accellera.org/systemc2017/

6. Ma A., Zacharda A. SystemC. Utilizing SystemC for Design and Verification. USA: Mentor Grapfics, 2005. p. 33.

7. Requirements specification for SystemC Analog Mixed Signal (AMS) extensions. Version 2.1. March 8, 2010. [Электронный ресурс]. http://www.systemc.org

8. Banerjee A., Sur B. SystemC and SystemC – AMS in Practice. SystemC 2.3, 2.2 and SystemC – AMS 1.0. New York, Dordrecht, London: Springer Cham Heidelberg; 2014. 462 p.

9. OSCI TLM-2.0 Language reference manual. Software version: TLM 2.0.1 Document version: JA32.: Open SystemC Initiative (OSCI). 2009. 194. p.

10. Aynsley J. Getting Started with TLM-2.0: Doulos.Ltd., 2017. [Электронный ресурс].
https://www.doulos.com/knowhow/systemc/tlm2/tutorial__1/

11. Grotker Th., Liao S., Martin G., Swan S. System Design with SystemC. New York, Boston, Dordrecht, London, Moscow: Kluwer Academic Publishers; 2002. 236 p.

12. Kalo G. Functional Verification with SystemC. Magnus Ljung, Integrated Systems Scandinavia AB. Christer Albinsson, KTH Syd. 2006. 29 p.

13. Yuan Wen and Hani Mohamed Khalil. SystemC-based electronic system level design methodology for SoC design-space exploration. In: Advances In Microelectronics. Penerbit UTM, Skudai, Johor Bahru, 2008. P. 9-34.

14. Black D. C., Donovan J., Bunton B., Keist A. SystemC: From the Ground Up. Second Edition. New York, Dordrecht, Heidelberg, London: Springer; 2010. 291 p.

15. Describing Synthesizabl RTL in SystemC. Printed in the U.S.A.: Synopsys, Inc., 2001. 116 p.

16. Bhasker J. A SystemC Primer. USA: Star Galaxy Publishing; 2002. 283 p.

17. Standard SystemC – AMS extensions Language Reference Manual. Open SystemC Initiative (OSCI). 2010. 152 p. [Электронный ресурс]. http://www.systemc.org

18. Requirements specification for SystemC Analog Mixed Signal (AMS) extensions Version 2.1. Open SystemC Initiative (OSCI). March 8, 2010. 27 p.
[Электронный ресурс]. http://www.systemc.org

19. SystemC – AMS extensions User's Guide. Open SystemC Initiative (OSCI). 2010. 166 p.
[Электронный ресурс]. http://www.systemc.org

20. Banerjee A., Sur B. SystemC and SystemC – AMS in Practice. SystemC 2.3, 2.2 and SystemC – AMS 1.0. New York, Dordrecht, London: Springer Cham Heidelberg; 2014. 462 p.

21. Grimm Ch., Barnasconi M., Vachoux A., Einwich K. An Introduction to Modeling Embedded Analog/Mixed-Signal Systems using SystemC – AMS Extensions. Open SystemC Initiative (OSCI). 2008. 12 p.
[Электронный ресурс]. https://publik.tuwien.ac.at/files/PubDat_171466.pdf

References:

1. Alekhin V.A. *Modelirovanie jelektronnyh sistem: uchebn. posobie dlya VUZov* (SystemC. Modeling of electronic systems). Moscow: Hotline – Telecom; 2018. 320 p. (in Russ.).

2. Alekhin V.A., Bykov I.A. Application SystemC for the design of electronic systems. Proceedings of the International Scientific and Practical Conference "Science, Education, Society". Tambov, March 31, 2018. P. 7-11 (in Russ.).

3. Devalapalli S. Development of SystemC Modules from HDL for System-on-Chip Applications. Master's Thesis, University of Tennessee, 2004.
URL: http://trace.tennessee.edu/utk_gradthes/2119

4. Functional specification for SystemC 2.0. April 5, 2002.
[Electronic resource]. http://www.systemc.org

5. Technical Tutorial: "SystemC Design and Verification – Solidifying the Abstraction Above RTL". Feb. 2, 2017. [Electronic resource]. http://videos.accellera.org/systemc2017/

6. Ma A., Zacharda A. SystemC. Utilizing SystemC for Design and Verification. USA: Mentor Grapfics. 2005. 33 p.

7. Requirements specification for SystemC Analog Mixed Signal (AMS) extensions. Version 2.1. March 8, 2010. [Electronic resource]. http://www.systemc.org

8. Banerjee A., Sur B. SystemC and SystemC – AMS in Practice. SystemC 2.3, 2.2 and SystemC – AMS 1.0. New York, Dordrecht, London: Springer Cham Heidelberg; 2014. 462 p.

9. OSCI TLM-2.0 Language reference manual. Software version: TLM 2.0.1 Document version: JA32.: Open SystemC Initiative (OSCI). 2009. 194 p.

10. Aynsley J. Getting Started with TLM-2.0: Doulos.Ltd., 2017. [Electronic resource].
URL: https://www.doulos.com/knowhow/systemc/tlm2/tutorial__1/

11. Grotker Th., Liao S., Martin G., Swan S. System Design with SystemC. New York, Boston, Dordrecht, London, Moscow: Kluwer Academic Publishers; 2002. 236 p.

12. Kalo G. Functional Verification with SystemC. Magnus Ljung, Integrated Systems Scandinavia AB. Christer Albinsson, KTH Syd. 2006. 29 p.

13. Yuan Wen and Hani Mohamed Khalil. SystemC-based electronic system level design methodology for SoC design-space exploration. In: Advances In Microelectronics. Penerbit UTM, Skudai, Johor Bahru, 2008. P. 9-34.

14. Black D. C., Donovan J., Bunton B., Keist A. SystemC: From the Ground Up. Second Edition. New York, Dordrecht, Heidelberg, London: Springer; 2010. 291 p.

15. Describing Synthesizabl RTL in SystemC. Printed in the U.S.A.: Synopsys, Inc., 2001. 116 p.

16. Bhasker J. A SystemC Primer. USA: Star Galaxy Publishing; 2002. 283 p.

17. Standard SystemC – AMS extensions Language Reference Manual. Open SystemC Initiative (OSCI). 2010. 152 p. [Electronic resource]. http://www.systemc.org

18. Requirements specification for SystemC Analog Mixed Signal (AMS) extensions Version 2.1. Open SystemC Initiative (OSCI). March 8, 2010. 27 p.
[Electronic resource]. http://www.systemc.org

19. SystemC – AMS extensions User's Guide. Open SystemC Initiative (OSCI). 2010. 166 p.
[Electronic resource]. http://www.systemc.org

20. Banerjee A., Sur B. SystemC and SystemC AMS in Practice. SystemC 2.3, 2.2 and SystemC – AMS 1.0. New York, Dordrecht, London: Springer Cham Heidelberg; 2014. 462 p.

21. Grimm Ch., Barnasconi M., Vachoux A., Einwich K. An Introduction to Modeling Embedded Analog/Mixed-Signal Systems using SystemC – AMS Extensions. Open SystemC Initiative (OSCI). 2008. 12 p.
[Electronic resource]. https://publik.tuwien.ac.at/files/PubDat_171466.pdf

Об авторе:

Алехин Владимир Александрович, доктор технических наук, профессор кафедры вычислительной техники Института информационных технологий ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78). ResearcherID: B-4747-2016, http://orcid.org/0000-0001-8291-6052

About the author:

Vladimir A. Alekhin, Dr. Sci. (Engineering), Professor, Department of Computing Engineering, Institute of Information Technologies, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow 119454, Russia). ResearcherID B-4747-2016, http://orcid.org/0000-0001-8291-6052

Поступила: 18.12.2019; Получена после доработки: 23.01.2020; Принята к опубликованию: 22.06.2020.