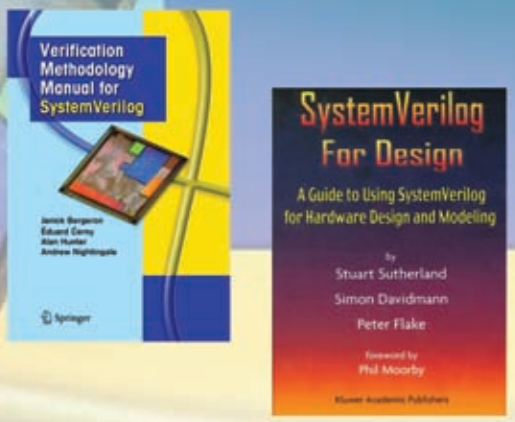
ЯЗЫК SystemVerilog

ПРОЕКТИРОВАНИЕ СБИС И СИСТЕМ



А. Пеженков
alexander.pejenkov@alt-s.com

Д. Радченко
dmitry.radchenko@alt-s.com

Развитие компонентной базы современной электроники требует создания новых средств проектирования. Когда-то основным инструментом разработчика интегральных микросхем был редактор принципиальных схем. При переходе к цифровым СБИС размерность схем стала такой, что ее описание средствами графического редактора принципиальных схем стало практически невозможным. Появилась идея проектирования цифровых СБИС на уровне межрегистровых передач с помощью специализированных языков, таких как Verilog и VHDL. Сегодня в рамках концепции "система на кристалле" перед разработчиками микросхем возникают задачи системного уровня. Идет поиск подходов, которые позволили бы решать их параллельно с традиционными задачами проектирования цифровых СБИС. Одним из перспективных направлений здесь может быть использование языка SystemVerilog. Компания Synopsys активно пропагандирует преимущества этого языка и использует его в своих средствах проектирования.

В поисках более эффективной организации процесса проектирования цифровых систем в микроэлектронной индустрии все больше внимание уделяется средствам проектирования системного уровня. Важная роль здесь принадлежит языкам описания проекта с более высоким, по сравнению с традиционными Verilog и VHDL, уровнем абстракции. Так, за последнее время появилось много проектов, работающих с такими языками, как C/C++, SystemC, "e" (система Vericity), "m" (система Matlab). В компании Synopsys большие надежды связывают с использованием языка SystemVerilog как наиболее органичного при переходе от проектирования СБИС к задачам системного проектирования. Этот язык хорошо воспринимается разработчиками, уже работающими с языками Verilog и VHDL, он удобен при решении задач функциональной верификации и обеспечивает возможность эффективного логического синтеза проекта из описаний на этом языке. Рассмотрим некоторые особенности языка SystemVerilog и историю его создания.

ЯЗЫК SystemVerilog. КРАТКАЯ СПРАВКА

На рис.1 можно проследить развитие языков, ориентированных на проектирование интегральных схем. Версия языка Verilog 1995 года включала возможности событийного моделирования, четырехзначную логику (1, 0, X, Z), поддержку аппаратного параллелизма, модульной организации, моделирования на уровне логических вентилей и переключателей (с учетом времени переключения), временного моделирования. Язык VHDL позволил разработчикам использовать новые типы данных, такие как многомерные массивы, структуры, автоматические и строковые переменные, целочисленные переменные со знаком, перечисляемые и пользовательские типы, указатели. Были добавлены механизмы динамического распределения памяти, перегрузки операторов, организации пакетов (независимо транслируемых модулей). Появилась возможность конфигурации архитектуры и динамической аппаратной генерации. Адаптация языков, базирующихся на языке C, к проектированию аппаратуры стимулировала движение в направлении объектно-ориентированного подхода. Стандарт Verilog 2001 подтянул возможности этого языка к возможностям языка VHDL, но поддержка развитых структур данных в нем все еще не была предусмотрена.

Язык SystemVerilog [1] – расширение стандарта языка Verilog 2001 (IEEE Std. 1364-2001). Первый официальный стандарт языка SystemVerilog был принят в ноябре 2005 года (IEEE Std. 1800-2005). Основой стандарта является руководство SystemVerilog 3.1a LRM, разработанное компанией Accellera. При его подготовке использовался положительный опыт следующих продуктов различных компаний:

- SUPERLOG ESS HDL (компания Co-Design Automation);
- OpenVera HVL (компания Synopsys);
- PSL Assertions (компания IBM) – язык описания и проверки свойств проекта;
- OpenVera Assertions (компания Synopsys) – другой вариант языка описания и проверки свойств проекта;
- DirectC и API (компания Synopsys) – языковые интерфейсы C/Verilog;
- принцип раздельной компиляции (компания Mentor Graphics).

SystemVerilog представляет собой новый тип языка – унифицированный язык описания и верификации аппаратуры (Hardware Description and Verification Language, HDVL). Включает в себя два подмножества конструкций:

- конструкции для описания проекта (Design Under Test, DUT);
- конструкции для создания тестового окружения (verification envronment) и проверки функционирования проекта (functional verification).

К наиболее важным новым возможностям, предоставляемым языком SystemVerilog, относятся:

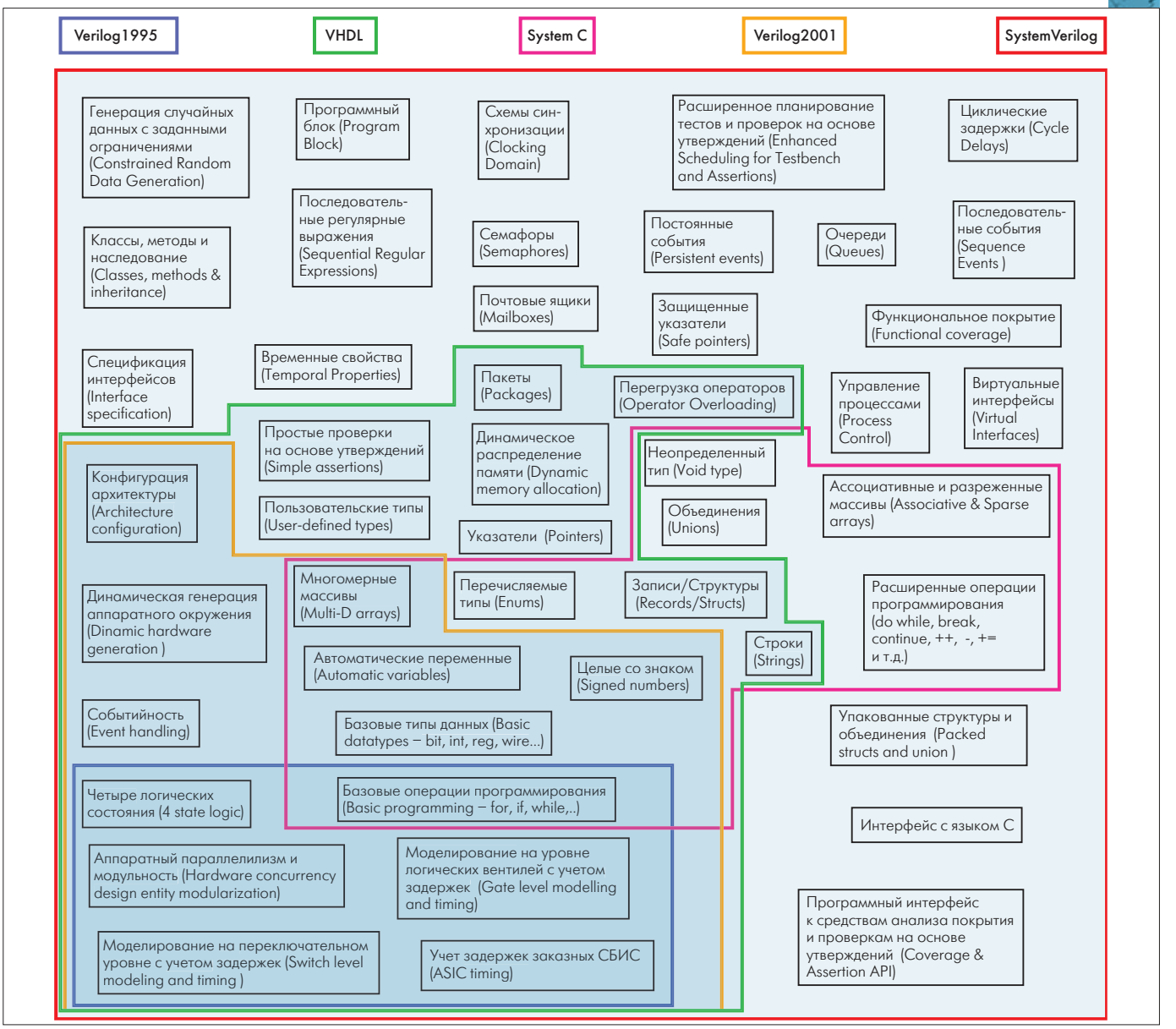


Рис.1. Эволюция языков описания аппаратуры

- объектно-ориентированное программирование (ООП);
- описание интерфейсов;
- новые базовые типы данных, перечисляемые типы (enum), многомерные массивы, структуры и объединения (union);
- строки, динамические и ассоциативные массивы, списки;
- описание новых типов (typedef) и преобразование типов;
- внешняя область объявлений (extern);
- операторы ++, --, += и другие;
- передача параметров процедур по ссылке (ref);
- специализированные блоки always (always_ff, always_latch, always_comb);
- специализированные операторы if и case (unique if/case, priority if/case);
- средства взаимодействия между параллельными процессами (semaphore, mailbox);
- унифицированный язык утверждений (assertions) для динамической и формальной верификации;
- генерация случайных данных, управляемая ограничениями (constraints);
- описание точек анализа функционального покрытия и их мониторинг;
- програмные блоки (program blocks) и схемы тактирования (clocking domain);
- прямой програмный интерфейс (Direct Programm Interface, DPI) C/Verilog.

SystemVerilog предназначен для разработки и проверки сложных аппаратных проектов большого объема (в том числе систем на кристалле). Этот язык позволяет значительно ускорить подготовку описания проекта благодаря наличию новых более удобных и компактных языковых конструкций (рис.2). Эффективность верификации повышается благодаря использованию единой унифицированной языковой среды, поддерживающей не только описание проекта, но и описание тестового окружения. Поддерживается общая семантика проверок на основе утверждений (assertions), причем как при моделировании, так и в тестовом окружении, а также при анализе функционального покрытия и в средствах динамического и статического формального анализа.

SystemVerilog В ПРОДУКТАХ SYNOPSYS

На сегодняшний день язык SystemVerilog поддерживается в таких продуктах компании Synopsys, как системы логического и смешанного моделирования **VCS** и **VCS MX**, система синтаксического и

Рис.2. Основные преимущества использования языка SystemVerilog

семантического анализа HDL-описаний на основе заданных правил и стандартов кодирования Leda, система формальной проверки логической эквивалентности описаний, представленных на вентильном или RTL-уровне Formality, гибридная система Magellan (функциональная верификация RTL-описаний на основе заданных утверждений с использованием формальных методов и динамического моделирования) и система логического синтеза Design Compiler. Таким образом, обеспечивается поддержка полноценного маршрута проектирования, использующего преимущества SystemVerilog.

Системы логического и смешанного моделирования VCS и VCS MX – промышленный стандарт в области традиционных средств функциональной верификации. В традиционном процессе функциональной верификации для заданого набора тестов проводится моделирование поведения системы на логическом уровне, затем полученные результаты анализируются на предмет совпадения с ожидаемыми. Приходится работать с огромными объемами кода описания проектов и не меньшим объемом кода, описывающего тестовые воздействия. При этом тестовые воздействия часто необходимо формировать в другой языковой среде. Поддержка языка SystemVerilog в системах VCS и VCS MX позволяет разработчику значительно сократить объем кода благодаря более удобным и компактным конструкциям этого языка. Это относится и к коду описания самого проекта, и к коду описания тестового окружения. Проект и тесты создаются в рамках единой языковой среды, что упрощает отладку и самого проекта, и системы тестов, а также обеспечивает постоянную взаимно согласованную отладку этих описаний.

Язык SystemVerilog позволяет перейти на новую ступень организации тестового окружения проекта благодаря использованию принципов объектно-ориентированного программирования, поддержке механизмов случайной генерации тестовых сигналов в рамках заданных ограничений и встроенных средств анализа покрытия. Специалистами компаний Synopsys и ARM на базе языка SystemVerilog разработана методология создания тестового окружения VMM (Verification Methodology Manual), следуя которой можно достигнуть наибольшей эффективности при разработке реальных промышленных проектов [2].

В этой методологии, помимо других возможностей языка, активно используется механизм встроенных проверок на основе утверждений. Такие проверки можно вставлять как в систему тестов, так и в описание самого проекта. Они намного повышают эффективность процесса отладки. Встраивание проверок в разрабатываемые IP-блоки позволяет в последующем предотвратить использование этих блоков в несанкционированных режимах. В рамках работы с языком SystemVerilog системы VCS и VCS MX поддерживают стандартные тестирующие модели на SystemC и интерфейс к программам на языке C, обеспечивают различные типы анализа покрытия (покрытие кода, функциональное покрытие), а также другие традиционные механизмы отладки, свойственные системам логического моделирования.

Система **Leda** предназначена для проверки правильности RTL-описаний с точки зрения стандартов кодирования. Помимо поддержки традиционных Verilog и VHDL поддерживается стандарт SystemVerilog. Система позволяет формулировать собственные правила и формировать на их основе фирменный стандарт описания, учитывающий специфику проектов. Предусмотрена возможность создания настраиваемых шаблонов правил. В описаниях на SystemVerilog система позволяет находить сложные ошибки, связанные с использованием нескольких схем синхронизации (clocking domain). Важное преимущество работы с SystemVerilog-описаниями – то, что здесь можно сформулировать правила проверки не только описания проекта, но и его тестового окружения. При работе в маршруте проектирования компании Synopsys могут быть использованы специальные шаблоны правил, соблюдение которых позволяет добиваться наилучших результатов логического синтеза в системе Design Compiler и обеспечивает наилучшую производительность моделирования в системе VCS.

Система **Formality** позволяет провести статический анализ функциональной эквивалентности различных представлений одного и того же проекта. Проверка производится на основе методов теории доказательств. Тестовых векторов не требуется. Система очень полезна при поиске оптимального варианта реализации функционирования схемы, например выборе оптимальной расстановки регистров, минимизации потребления, подборе вариантов схемы тактирования. Formality позволяет быстро убедиться, что новые варианты эквивалентны исходному. Продукт полностью поддерживает описания на SystemVerilog, позволяет проверять эквивалентность не только для СБИС на базе стандартных ячеек, но и полностью заказных СБИС и схем памяти.

Система **Magellan** – гибрид моделирования и формальной верификации. Поскольку формальные методы позволяют провести

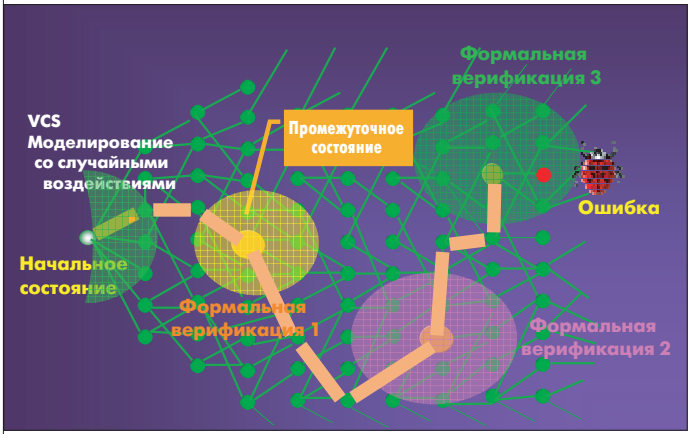


Рис.3. Гибридная технология верификации

верификацию RTL-описаний только в некоторой ограниченной области состояний, ошибки, расположенные вне этой области, остаются невыявленными. Идея гибридного подхода, который использует система Magellan, проиллюстрирована на рис.3. В данном случае формальный анализ в точке исходного состояния не позволяет выявить "спрятанную в углу" ошибку. Но если провести формальный анализ для множества промежуточных состояний, вероятность нахождения "спрятанной" ошибки значительно возрастает. Для перехода из одного состояния в другое используются средства моделирования (в систему встроено ядро VCS) и случайные (с учетом заданных ограничений) воздействия.

Возможности, которые предоставляет язык SystemVerilog в области поддержки механизма утверждений, чрезвычайно важны для работы системы Magellan. Утверждения языка SystemVerilog могут использоваться как при формальной верификации (для описания характеристик правильности функционирования), так и в качестве индикаторов нарушений при моделировании. Кроме того, что важно для гибридной технологии верификации, утверждения языка SystemVerilog позволяют описать ограничения на воздействия, генерируемые для передачи в моделирующее ядро. Помимо утверждений, создаваемых разработчиками, SystemVerilog предусматривает наличие расширенной собственной библиотеки стандартных утверждений (табл.1).

Система **Design Compiler** – один из признанных лидеров в области средств логического синтеза. Для разработчиков цифровых СБИС и систем на кристалле, использующих преимущества SystemVerilog в области создания RTL-описания и его верификации, в конечном счете важно, чтобы из отлаженного RTL-описания можно было автоматически синтезировать принципиальную схему проекта и результат при этом не уступал бы результатам, полученным при использовании описаний на языках Verilog или VHDL. На сегодняшний день можно сказать, что задача поддержки логического синтеза из описаний на языке SystemVerilog в системе Design Compiler в целом успешно решена. Для проведения логического синтеза используется то же самое описание, что и для верификации, включая утверждения, вставленные для формальной верификации и проверок в процессе моделирования, а также части описания, формирующие тестовое окружение. Использование новых конструкций языка поддерживается средствами логического синтеза.

В табл.2 приведены сравнительные результаты логического синтеза для одной и той же тестовой схемы, реализованной на традиционном Verilog и на SystemVerilog с использованием новых компактных конструкций (typedefs, structures, interfaces, modports, always_*) [3]. Как можно заметить, качество полученных результатов для этой схемы практически одно и то же. Однако не следует забывать, что, как и при использовании традиционных языков, результат в значительной мере зависит от того, насколько разработанное описание удобно для проведения логического синтеза. В этом отношении SystemVerilog имеет свои особенности, и их, конечно, надо учитывать при создании RTL-описания. Маршрут проектирования компании Synopsys предусматривает сквозную поддержку создания описания на языке SystemVerilog, удобного для последующего логического синтеза.

Таблица 1. Встроенная библиотека утверждений языка SystemVerilog

Проверка правильности значений	Проверка протоколов	Проверка правильности состояния	Проверка последовательностей
assert_bits	assert_arbiter	assert_code_distance	assert_hold_value
assert_value	assert_data_used	assert_driven	assert_reg_loaded
assert_odd_parity	assert_dual_clk_fifo	assert_mutex	assert_time
assert_one_cold	assert_fifo	assert_next_state	assert_transition
assert_one_hot	assert_memory_sync	assert_no_transition	assert_unchange
assert_even_parity	assert_memory_async	assert_proposition	assert_width
assert_always_on_edge	assert_no_contention	assert_quiescent_state	assert_win_change
assert_decrement	assert_req_ack_unique	assert_never	assert_win_unchange
assert_delta	assert_stack	assert_next	assert_window
assert_zero_one_hot	assert_valid_id	assert_always	assert_implication
assert_no_underflow	assert_fifo_index		assert_cycle_sequence
assert_no_overflow	assert_proposition		assert_change
assert_no_underflow	assert_frame		
assert_increment	assert_handshake		
assert_range			

Таблица 2. Результаты логического синтеза тестовой схемы на языках Verilog и на SystemVerilog

	Verilog	SystemVerilog
Число логических уровней (Levels of Logic)	10	10
Длина критического пути (Critical Path Length)	6,38 нс	6,38 нс
Превышение по критическому пути (Critical Path Slack)	-6,38 нс	-6,38 нс
Обшее превышение (Total Negative Slack)	-701,23 нс	-701,91 нс
Число путей с нарушениями (No. of Violating Paths)	204	204
Число иерархических ячеек (Hierarchical Cell Count)	242	242
Число ячеек нижнего уровня (Leaf Cell Count)	6261	6266
Комбинационная логика (Combinational Area)	10590 эквивалентных вентилей	10590 эквивалентных вентилей.
Некомбинационная логика (Noncombinational Area)	14903 эквивалентных вентилей	14907 эквивалентных вентилей
Цепи (Net Area)	0	0
Область ячеек (Cell Area)	25493 эквивалентных вентилей	25497 эквивалентных вентилей
Рабочая область (Design Area)	25493 эквивалентных вентилей	25497 эквивалентных вентилей
Время работы компилятора (Overall Compile Time)	105,60 с	105,67 с

SystemVerilog В МАРШРУТЕ ПРОЕКТИРОВАНИЯ SYNOPSYS

На рис. 4 представлен маршрут проектирования аппаратуры на базе языка SystemVerilog с помощью инструментов компании Synopsys. В отдельных проектах может быть использован не полный набор инструментов, однако каждое из этих средств весьма важно для создания работоспособного проекта. Так, использование системы Leda позволяет проверить стиль кодирования и скорректировать описание таким образом, чтобы предотвратить потенциальную возможность функционального рассогласования RTL-описания, подаваемого на этап логического синтеза, и описания на уровне логических вентилей, полученного в результате синтеза. Кроме того, Leda помогает обеспечить эквивалентность моделей на RTL-уровне и на уровне вентилей. Важно также то, что система Design Compiler обычно не применяется для всего проекта целиком, синтез проводится поблочно. Семантический анализ конструкций SystemVerilog проводится для отдельного блока. Использование же VCS, Magellan и Leda позволяет проверить проект в целом. Моделирование на уровне логических вентилей в системе VCS и формальная верификация в системе Formality также позволяют проверить соответствие исходного RTL-описания проекта полученным результатам для всего проекта.

Используя данный маршрут проектирования, нужно учитывать, что не все конструкции языка могут быть абсолютно адекватно преобразованы в аппаратное описание. Простой пример – применение битовых переменных обеспечивает поддержку только двух состояний сигнала (1, 0), в то время как для описания состояния на вхо-

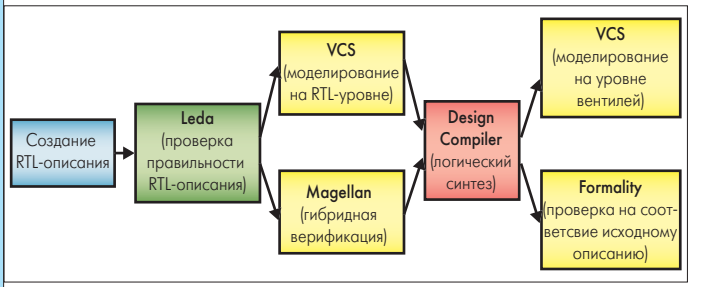


Рис.4. Маршрут компании Synopsys для языка SystemVerilog

Табл.3. Конструкции языка SystemVerilog и логический синтез

Ключевые слова, поддерживаемые при синтезе	Ключевые слова, игнорируемые при синтезе	Ключевые слова, не поддерживаемые при синтезе	Системные функции, поддерживаемые при синтезе	Системные функции, не поддерживаемые при синтезе
always_comb, always_ff, always_latch, bit, break, byte, const, continue, do, endinterface, enum, importint, interface, logic, longint, modport, packed priority, return, shortint, struct, typedef, union, unique, void	assert, endsequence, sequence, assume, expect, timeprecision, endproperty, property, timeunit	alias, before, bind, bins, binsof, chandle, class, clocking, constraint, context, cover, covergroup, coverpoint, cross, dist, endclass, endclocking, endgroup, endpackage, endprogram, export, extends, extern, final, first_match, foreach, forkjoin, iff, ignore_bins, illegal_bins, inside, intersect, join_any, join_none, local, matches, new, null, package, program, protected, pure, rand, randc, randcase, randsequence, ref, shortreal, solve, static, string, super, tagged, this throughout, type, uwire, var, virtual, wait_order, wildcard, with, within	$bits, $countones, $dimensions, $high, $increment, $left, $low, $onehot, $onehot0, $right, $size, $unit	$assertkill, $assertoff, $asserton, $bitstoshortreal $cast, $coverage_control, $coverage_merge, $coverage_save, $error, $exit, $fatal, $fell, $get_coverage, $info, $isunbounded, $isunknown, $load_coverage_db, $past, $root, $rose, $sampled, $set_coverage_db_name, $shortrealtobits, $stable, $typename, $urandom, $urandom_range, $warning, $writememb, $writememh

де/выходе реального логического элемента необходимо по крайней мере четыре значения (1, 0, X, Z). Подобное утверждение справедливо для всех языков, но SystemVerilog содержит новые возможности, к которым разработчики еще не привыкли, поэтому имеет смысл определить набор конструкций, работа с которыми в предлагаемом маршруте проектирования обеспечит создание удобного для логического синтеза описания. Можно, например, использовать вариант такого набора, предложенный разработчиками компании Synopsys [4]. С определенной степенью условности эти рекомендации могут быть отражены в виде таблицы, характеризующей элементы языка SystemVerilog (табл.3). Надо заметить, что таблица отражает только текущее состояние, в перспективе возможности использования SystemVerilog в маршруте проектирования Synopsys будут постоянно расширяться.

На сегодняшний день стандарт языка SystemVerilog в той или иной мере поддерживается всеми ведущими производителями САПР микроэлектроники. Активно развиваются средства проектирования на его основе. У этого языка есть хорошие шансы в ближайшее время стать базовым средством проектирования СБИС и систем на кристалле.

ЛИТЕРАТУРА

1. S.Sutherland, S.Davidmann, P.Flake. SystemVerilog for Design. – Springer-Verlag, Boston, Massachusetts. Copyright 2004. ISBN: 1-4020-7530-8.

2. J.Bergeron, E. Cerny, A. Hunter, A. Nightingale. Verification Methodology Manual for SystemVerilog. – Springer, 2005, 510 p., ISBN: 0-387-25538-9

3. K.Pieper. SystemVerilog from a Synthesis Perspective. – Design and Verification Conference (DVCon), 2004.

4. S.Sutherland. Modeling with SystemVerilog in a Synopsys Synthesis Design Flow Using Leda, VCS, Design compiler and Formality. – Sutherland HDL, Inc. (www.synopsys.com)