Блок САР позволяет настраивать параметры блоков системы от уровня входного/выходного сигнала до глубины задержки блока эффектов. Схема содержит регуляторы (усилители) уровня входного сигнала для Увх, регуляторы уровня сигнала по каждой из полос пропускания Фn для САФ, регуляторы задержки и глубины у САЭ и регулятор уровня выходного сигнала для Увых.

На этапах макетирования разработчик строит функциональную схему звуковой аппаратуры и осуществляет её создание. Если схема удовлетворяет ряду тестов, то выходной сигнал соответствует желаемому. Программирование требует оформление полученной схемы в программный вид. Так же стоит отметить, что на этапе программирования требуется спроектировать отдельную модель системы (схемы) и привести к функциональному виду.

Библиографический список использованной литературы

1. Моделирование активных фильтров [Текст] / Под ред. Г.Мошиц, П.Хорн. — Л.: Мир, 1984. — 318 с.

Е.А. Чернова, студентка
Научный руководитель: В.Е. Стиканов, доц., канд. техн. наук
Ннациональный технический университет Украины «Киевский политехнический институт»
E-mail: h2sif6@mail.ru

ВЕРИФИКАЦИЯ В SYSTEMVERILOG

The report contains information about the methods of the hardware verification with SystemVerilog.

Развитие компонентной базы современной электроники требует создания новых средств проектирования. Раньше базовыми инструментами разработчиков интегральных микросхем были разнообразные редакторы принципиальных схем. Но, при переходе к цифровым сверхбольшим интегральным схемам, их размерность стала сильно возрастать, а описание этих схем средствами одного лишь графического редактора принципиальных схем стало очень сложным и, в некоторых случаях, просто невозможным. Появилась идея проектирования цифровых сверхбольших интегральных схем на уровне межрегистровых передач с помощью специализированных языков, таких как Verilog и VHDL.

Поскольку микросхемы становятся все более и более дешевыми, а количество транзисторов в них все больше и больше увеличивается, то это способствует увеличению сложности проектов. Как следствие, это неизбежно приводит к требованию применять еще более производительные программные инструменты для отладки проектов. Верификация проектов — это самая трудоемкая операция.

Верификацией называется проверка соответствия результатов, полученных на отдельных этапах проектирования, требованиям и ограничениям, установленным для них на предыдущих этапах. На начальном этапе проверяют, соответствуют ли результаты исходным требованиям (техническим заданием). Основная задача верификации - контроль качества проектирования, включая такие его аспекты, как функциональная корректность, надежность, удобство использования и многие другие.

Именно плохая верификация проектов приводит к наибольшим потерям при разработке проектов. Чем более сложными становятся проекты, и чем больше у них функциональных возможностей, тем больше пропасть между сложностью проекта и моментом окончания отладки такого проекта.

Широкие возможности языка SystemVerilog позволяют ему соответствовать многим методикам проверки, и все эти методики можно наиболее эффективно использовать вместе.

Базовыми методиками верификации можно назвать:

— генерацию случайных воздействий, выполненных с учетом ограничений;
— верификацию, управляемую функциональным охватом;
— утверждения;
— множественные testbench использующие транзакции;
— общий testbench для всех тестов;
— отдельный от testbench специальный испытательный код и другие.

Все эти принципы связаны между собой. Генерация случайных значений, выполненная с учетом ограничений, имеет решающее значение для осуществления сложных конструкций. Направленный тест находит те ошибки, которые вы ожидаете в проекте, в то время как случайный тест может найти ошибки, которые вы не ожидаете. При использовании генерации случайных воздействий, выполненных с учетом ограничений, требуются функциональные покрытия для измерения проверки прогресса. Создание testbench структуры, в том числе само прогнозирования, представляет собой значительный объем работы. Множественные testbench позволяют управлять сложностью, разбивая задачу на управляемые части. Транзакции обеспечивают выгодную модель для построения этих частей. При наличии соответствующего планирования, можно построить testbench структуру, которая может быть общей для всех тестов и не должна будет постоянно изменяться. Нужно оставить зацепки для тестов, где могут выполнять определенные действия, такие как формирование последовательностей возбуждения и внутренних смещений. С другой стороны, код, специфический для простого теста должен храниться отдельно от testbench, чтобы не усложнять структуру.

В языке SystemVerilog есть новые мощные средства верификации языковых конструкций, такие как ограничения, функциональный охват и объектно-ориентированное программирование.

В 2005 году SystemVerilog был принят как стандарт IEEE 1800-2005 [1]. В 2009 году стандарт был объединен с базовым стандартом Verilog (IEEE 1364-2005) и создан стандарт IEEE 1800-2009. Полное название стандарта IEEE Std 1800-2009: IEEE Standard for SystemVerilog — Unified Hardware Design, specification, and Verification Language, который является текущей и актуальной версией.

Язык SystemVerilog обеспечивает все конструкции и имеет все возможности для того, чтобы построить расширенную среду проверки, охватывающую генерацию случайных воздействий, выполненных с учетом ограничений, проверку, управляемую охватом, и утверждения. Применение языка SystemVerilog может улучшить тестирование, автоматизировать рутинные операции по составлению тестов и увеличить охват тестированием проекта.

Библиографический список использованной литературы

1. IEEE IEEE Standard for SystemVerilog — Unified Hardware Design, Specification, and Verification Language. New York: IEEE 2005

2. Spear Chris. SystemVerilog for Verification: A Guide to Learning the Testbench Language Features / Chris Spear. — Norwell, MA: Springer, 2006. — 301 c.