УДК 621.3.049.771.14 DOI: 10.22184/NanoRus.2019.12.89.159.161

НАГРУЗОЧНОЕ ТЕСТИРОВАНИЕ В МАРШРУТЕ РАЗРАБОТКИ СБИС

STRESS TESTING IN THE VLSI DEVELOPMENT ROUTE

ЛИСТОПАДОВ АНДРЕЙ ДМИТРИЕВИЧ
ФГУ ФНЦ НИИСИ РАН
117218, Москва, Нахимовский просп., 36, к. 1
listopad@cs.niisi.ras.ru

LISTOPADOV ANDREY D.
SRISA RAS
bld. 1, 36 Nakhimovskiy Ave., Moscow, 117218
listopad@cs.niisi.ras.ru

Описана система нагрузочного тестирования Stress для исследования систем на кристалле. Изложены основные задачи, решаемые системой, и способы решения, ее состав и структура.
Ключевые слова: нагрузочное тестирование; тестовая система; система на кристалле.
The paper highlights a stress testing system for SOC research, as well as presents design details and the main tasks solved by the Stress system.
Keywords: stress testing; testing system; system on chip.

Разработка микроэлектронных устройств — это сложный и многоуровневый процесс, требующий от разработчиков достижения таких параметров, как высокие показатели производительности, высокий процент выхода годных микросхем, широкий набор выполняемых функций, низкая себестоимость и др.

Маршрут разработки микросхем от технического задания до серийного производства включает множество различных этапов, на каждом из которых требуется верификация проекта. Для устройств с микропроцессором, систем на кристалле (СнК), на первый план выходит верификация программными тестами. Программное тестирование СнК и отдельных микропроцессоров применимо и необходимо на всех этапах разработки: поведенческое моделирование, разработка RTL, NetList, отработка на ПЛИС, отбраковка микросхем на производстве и проверка в составе изделия — электронного модуля или ЭВМ.

Помимо программной верификации отдельных компонентов разрабатываемой системы, проводятся испытания с изменением тактовых частот, напряжения, температуры микросхемы. Нагрузочное тестирование позволяет расширить границы исследовательского процесса, увеличив количество возможных состояний схемы. Необходимость проведения нагрузочных тестов возникает уже на этапе прототипирования с использованием программируемых логических интегральных схем (ПЛИС), а также на платах для функционального контроля микросхемы. Во время нагрузочного тестирования все компоненты схемы работают в самом интенсивном режиме продолжительные отрезки времени. Интенсивный режим работы влечет за собой изменения в энергопотреблении и температуре исследуемой схемы. Результаты нагрузочного тестирования позволяют учитывать эти параметры при разработке новых изделий.

Существует довольно большое количество систем для проведения нагрузочного тестирования, например тесты, написанные для операционной системы GNU/Linux, такие как stress-ng [1], Phoronix Test Suite [2] и др. Отдельные тесты могут загрузить различные компоненты системы, однако другие будут простаивать. Существуют и тестовые системы, загружающие все системные компоненты, но операционная система сама по себе забирает часть процессорного времени, что может понизить общую степень нагрузки. Стоит отметить, что далеко не все производимые изделия рассчитаны на работу с типовой операционной системой (GNU/Linux), поэтому возникает необходимость разработки системы стресс-тестирования СнК, способной работать без операционной системы.

Американская компания Cadence Design Systems представила комплексное решение Perspec System Verifier [3], предназначенное для автоматизации разработки тестов и упрощения процессов проектирования тестовых сценариев. Данный программный комплекс предоставляет возможность написания тестов на языке SLN, из которых впоследствии строятся тестовые сценарии, что вызывает необходимость написания абсолютно новых тестов и библиотек. Поскольку тестовая система с довольно обширным пакетом тестов для различных проектов уже существует в ФГУ ФНЦ НИИСИ РАН, было решено создать систему стресс-тестирования на основе уже существующей базы.

Основой для проекта Stress стала инфраструктура системы тестирования логических моделей KMDTESTKIT (далее KMD). Подробное устройство тестовой системы описано в работе [4]. KMD содержит большое количество тестов компонентов разрабатываемых модулей, имеет механизм поддержки тестирования нескольких проектов (платформ) и позволяет вести разработку кросс-платформенных тестов компонентов СнК. Каждый тест представляет собой автономный исполняемый двоичный код, который можно загрузить в электронный модуль и передать ему управление. Тесты в KMD рассчитаны на тестирование как логической модели (RTL), так и ПЛИС-прототипов и опытных образцов и представляют собой тесты отдельных компонентов СнК. Поскольку на данный момент существует довольно обширная библиотека тестов для различных проектов, тесты отдельных компонентов можно переиспользовать для создания кросс-платформенных нагрузочных тестов.

Тестовая система Stress представляет собой генератор, предназначенный для формирования исходного исполняемого файла из набора нагрузочных тестов. Данный генератор опирается на конфигурационный файл, содержащий в себе список тестов, и параметры тестирования. Исходный текст стресс-теста формируется из заготовок тестов, указанных в конфигурационном файле, после чего кросс-компилируется для исследуемой платформы. Впоследствии будет произведена загрузка в модуль и передача управления.

МЕТОДИКА ТЕСТИРОВАНИЯ

Нагрузочный тест пишется для отдельного устройства как автономная единица компиляции. В тесте производятся все настройки, специфичные для конкретного устройства: настраиваются регистры, программируется DMA (если есть), или задание пишется непосредственно в память устройства, например МКИО. После настройки производится запуск и регулярная проверка состояния контроллера. Если устройство закончило выполнять задачу, в зависимости от настроек генератора выполняются проверки и производится перезапуск задачи. После тестирования производится остановка незавершившихся задач и на экран выводится лог тестирования.

Построение нагрузочного теста выполняется в соответствии с конфигурационным файлом, закрепленным за платформой. В данном файле перечисляются тесты, которые будут включены в нагрузочный тест, и параметры тестирования, такие как количество итераций нагрузочного цикла, наличие и глубина проверки отдельных контроллеров и поведение теста в случае обнаружения ошибки. Например, мы можем захотеть получить результаты за самый короткий промежуток тестирования, который приводит к ошибке, и в конфигурационном файле задается условие остановки тестирования в случае обнаружения ошибки в любом из контроллеров. Или наоборот, если нам интересно, как будет себя вести система дальше, остановка после обнаружения ошибки выключается. Иногда необходимости проверки на ошибки нет в принципе, поэтому ее можно отключить, освободив таким образом часть процессорного времени.

Основная задача процессора при данном виде тестирования — убедиться, что каждый контроллер не простаивает. Процессор опрашивает каждый контроллер на каждой итерации нагрузочного цикла на предмет выполнения задания и проверки ошибок. Так как нагрузочный тест создается специально для интенсивной работы схемы, задания контроллерам программируются также достаточно продолжительные, что в результате освобождает довольно большое количество процессорного времени, и частоту опроса устройств можно понизить. Чтобы процессор не простаивал, опрашивая устройства впустую, производится нагрузка на различные сопроцессоры: FPU, DSP и векторный сопроцессор. Так как целочисленные вычисления не являются энергоемкими в достаточной степени, процессор в качестве основной задачи выполняет мониторинг, предпочтение отдается сопроцессорам. Благодаря тому что время выполнения задач каждым контроллером может сильно отличаться, запущенные одновременно задачи будут достаточно сильно распределены во времени, чтобы не мешать друг другу во время проверок, и простоев центрального процессора удастся избежать в большей степени.

После того как необходимые тесты написаны, они передаются генератору Stress, который является основой данной системы. Генератор использует конфигурационный файл, в котором перечисляются тесты и параметры, для формирования исходного исполняемого файла, который будет загружен в память исследуемой микросхемы. В ходе генерации формируется список вызовов всех функций пролога, после которого создается основной нагрузочный цикл. В данном цикле последовательно вызываются нагрузочные функции, в которых происходит опрос устройств. Условия завершения цикла задаются в конфигурационном файле. После нагрузочного цикла идет последовательный вызов функций эпилога для каждого тестируемого компонента схемы. Производится печать статистики, и программа завершает выполнение.

Во время прохождения тестирования на исследуемом устройстве производятся замеры энергопотребления, температуры, из которых формируются характеристики микросхемы.

УСТРОЙСТВО НАГРУЗОЧНОГО ТЕСТА

Все тесты имеют три этапа работы: prologue, stress_test и epilogue, образующие три основные функции в исходном коде. Такое разделение помогает распределить обязанности на этапе генерации. Каждая функция в тесте принимает на вход два параметра: номер устройства и уровень проверки, и именуется следующим образом: <device_name>_<stage_name>(), например, timer_prologue(), can_stress_test(), spw_epilogue() и т.д.

Функция device_name_prologue() выполняет все необходимые для устройства инициализации и проверяет, что устройство готово к работе. Некоторым устройствам старт может быть дан в данной функции, например если задача достаточно длинная или если устройство никогда не останавливается (циклический таймер). Впоследствии генератор сформирует список из вызовов функций пролога, подготавливающий все устройства перед тестированием.

Далее идет функция тестирования device_name_stress_test(). Это очень простая и быстрая функция, которая запускает контроллер, если тот не был запущен в прологе, и проверяет, не закончил ли он работу. Если контроллер закончил выполнение своей задачи, данная функция дает ему команду повторить эту задачу. Также данная функция выполняет сбор статистики работы каждого контроллера, если в конфигурационном файле указано, какие контроллеры нуждаются в проверке, и насколько глубокой она должна быть. Поддерживается три вида проверок: быстрая проверка, поверхностно проверяющая регистры на наличие ошибок и сохраняющая состояние регистра, более глубокая проверка, разбирающая регистр на предмет ошибок, сохраняя каждую ошибку как отдельное сообщение для последующей выдачи на экран после завершения теста, и длительная проверка, зависящая от типа устройства, требующая большего процессорного времени, например проверка корректности передачи данных. Длительные проверки в большинстве случаев опускаются, поскольку снижают общую нагрузку и увеличивают интервалы простоя контроллеров. Некоторые устройства можно проверять прямо во время работы, другие — только между перезапусками. Устройства, которые можно проверять когда угодно, проверяются раз в фиксированное количество циклов. Устройства, которые проверяются во время перезапусков, проверяются каждый раз после N перезапусков, количество которых задается в конфигурационном файле. Стресс-функция может возвращать три типа значений:

- 0 — тестовая функция удостоверилась, что контроллер выполняет свое задание;
- 1 — тестовая функция удостоверилась, что контроллер закончил свое задание и перезапустила контроллер;
- отрицательный код — тестовая функция обнаружила ошибку в работе тестируемого контроллера.

Во время работы тестовой функции производится сбор данных о холостых заходах в функцию — количество раз, когда функция вернула 0, количество заходов с перезапусками и количество ошибок. Данные о количестве ошибок также являются данными о количестве перезапусков, так как проверка

на ошибки происходит пред перезапуском. В таком случае приоритет отдается сообщению об ошибке, так как в конфигурационном файле может быть дано указание останавливать выполнение стресс-теста при нахождении определенного количества ошибок. Данные об ошибках сохраняются отдельно для каждого устройства в виде коротких кодов, каждый из которых специфичен для конкретного устройства.

Первичные прогоны стресс- тестов используются для сбора информации о том, сколько времени процессор простаивает между подталкиванием контроллеров после завершения их текущего задания. Стоит отметить, что не все компоненты СнК целесообразно включать в тест. Например, I2C является достаточно низкочастотным контроллером и не способен создать должной нагрузки, однако требует времени на обслуживание. Свободное процессорное время используется для нагрузки сопроцессоров: FPU, DSP, векторный сопроцессор.

После того как основной цикл тестирования завершен, выполняются функции завершения тестирования device_name_epilogue() для каждого контроллера. Производится остановка незавершившихся заданий, преобразование собранных сообщений об ошибках в осмысленные сообщения об ошибках.

ЗАКЛЮЧЕНИЕ

Разработанная система нагрузочного тестирования Stress предоставляет дополнительные исследовательские возможности и позволяет выявить узкие места в проектировании СнК. В ходе проведения исследований разработчиками были получены результаты, наглядно демонстрирующие важность внедрения нагрузочных испытаний в маршрут разработки СБИС.

ЛИТЕРАТУРА

1. stress-ng https://wiki.ubuntu.com/Kernel/Reference/stress-ng.
2. Phoronix Test Suite — Open-Source, Automated Benchmarking https://www.phoronix-test-suite.com.
3. Perspec System Verifier — SoC verification with portable stimulus. https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/system-design-verification/perspec-system-verifier-ds.pdf.
4. Сидоров С. А., Слепов А. Б. Система тестирования логических моделей KMDTESTKIT // Труды НИИСИ РАН. Т. 8 № 1. Математическое и компьютерное моделирование сложных систем: теоретические и прикладные аспекты. — М.: ФГУ ФНЦ НИИСИ РАН, 2018. — С. 37–43 (7 стр.) ISSN 2225-7349.