УДК 004.312.44

Сопроцессоры вещественной и комплексной арифметики и их тестирование

Н.В. Николина, П.С. Зубковский, П.А. Чибисов

Учреждение Российской академии наук Институт системных исследований РАН,

nikolina@cs.niisi.ras.ru

***Аннотация* — В статье представлено описание встроенных арифметических сопроцессоров, а также подходы, применяемые для тестирования этих сопроцессоров.**

***Ключевые слова* — Сопроцессор, тестирование.**

I. Введение

При разработке микропроцессоров наиболее важным направлением является создание универсального, хорошо программируемого устройства. Это гарантирует, что изделие может использоваться в разных областях и адаптироваться к быстро меняющимся требованиям по цифровой обработки сигналов. Хотя вычислительные возможности процессора позволяют обрабатывать значительный поток данных в реальном времени, ядро процессора желательно интегрировать на кристалл с набором сопроцессоров, позволяющих существенно повысить производительность ряда вычислительных задач. Сопроцессоры могут обрабатывать задачи с жесткими временными требованиями. Например, эффективность регулярных вычислений, требующих высокой скорости передачи данных, будет выше при выполнении на арифметическом сопроцессоре, нежели чем на ядре процессора. Сопроцессоры взаимодействуют между собой и процессором по локальной шине и разделяют доступ к общей внешней памяти. Таким образом, ядро процессора можно комбинировать с различным набором таких внутрикристальных сопроцессоров, достигая тем самым как наибольшей универсальности, так и хорошей производительности, что является чрезвычайно важным для того, чтобы конечный продукт стал востребованным на рынке.

В данной статье рассматриваются два сопроцессора, которые могут быть интегрированы на кристалл вместе с ядром процессора. Это сопроцессор вещественной арифметики и сопроцессор комплексных вычислений. Будут рассмотрены как характеристики и возможности данных сопроцессоров, так и методы их тестирования в составе микропроцессора.

II. Блок вещественной арифметики (FPU)

Блок вещественной арифметики является сопроцессором (FPU) центрального ядра процессора и предназначен для выполнения арифметических операций над числами с плавающей запятой одинарной и двойной точности, а также может осуществлять обмен данными с центральным ядром процессора и памятью. Блок вещественной арифметики может одновременно выполнять две инструкции процессора – одну арифметическую инструкцию и одну инструкцию обмена данными с ядром процессора или памятью.

Арифметический тракт блока вещественной арифметики включает в себя вычислительные модули, выполняющие операции умножения, сложения, деления, извлечения квадратного корня над числами с плавающей запятой одинарной и двойной точности. Помимо этого, могут быть выполнены операции преобразования форматов чисел, сравнения. Все операции выполняются в соответствии со стандартом IEEE 754. Блоки умножения, сложения, преобразования форматов, сравнения построены по полностью конвейерной схеме и позволяют запускать новую операцию вычисления в каждом такте. Предусмотрена возможность выполнения операций умножения, сложения и умножения с накоплением над сдвоенными числами одинарной точности. Для этого в состав арифметического тракта блока вещественной арифметики включены по два параллельных модуля умножения и сложения одинарной точности.

Блок вещественной арифметики включает в себя три 32-х разрядных управляющих регистра. Регистр идентификации доступен только для чтения и содержит идентификационный номер блока. Регистр конфигурации позволяет разрешать или запрещать выполнение инструкций деления и извлечения квадратного корня. Регистр управления и состояния позволяет разрешать или запрещать прерывания при возник-

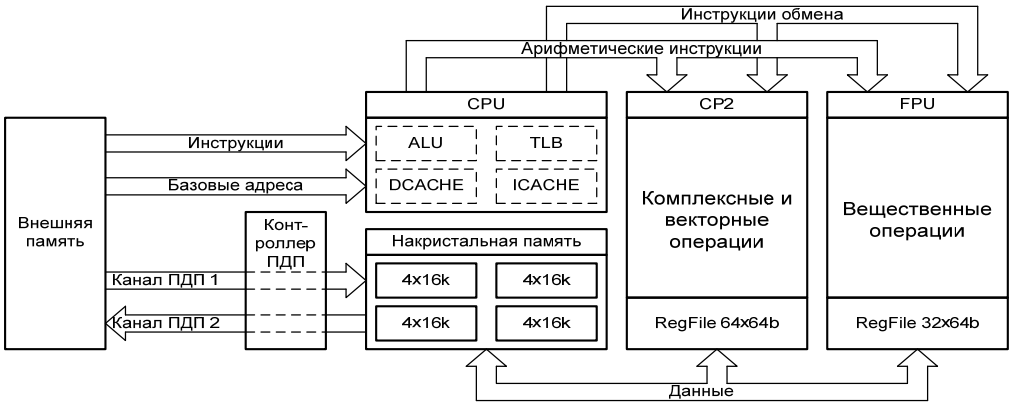

Рис. 1. Взаимодействие сопроцессоров вещественной и комплексной арифметики с внутрикристальной памятью

новении исключительных ситуаций, содержит флаги возникших исключительных ситуаций и коды сравнения для выполнения условных переходов и сдвигов.

III. Сопроцессор комплексных вычислений

Сопроцессор комплексных вычислений (CP2) представляет собой специализированный сопроцессор, ориентированный на выполнение арифметических операций над комплексными числами с плавающей запятой.

Операнды инструкций CP2 – это, в основном, 64-х разрядные слова, которые в зависимости от типа инструкции, могут рассматриваться как два вещественных числа в формате Paired Single (сдвоенные числа одинарной точности), либо могут рассматриваться как комплексное число, где старшая часть числа Paired Single является действительной частью комплексного числа, а младшая часть Paired Single – мнимой частью комплексного числа.

Также существует возможность оперировать 128-ми разрядными словами и получать 128-ми разрядный результат операции, например, для инструкции суммирования и вычитания комплексных чисел. Каждая инструкция работает с одним, двумя или тремя операндами из регистрового файла с 64-мя 64-х разрядными регистрами. Каждая арифметическая инструкция может выполняться параллельно с инструкцией обмена данными.

Операции над вещественными и комплексными числами включают в себя: сложение и вычитание, умножение с накоплением, умножение с вычитанием, умножение с накоплением и вычитанием, изменение знака числа независимо для старшей и младшей частей, сравнение чисел, пересылка данных между регистрами. Только над комплексными числами предусмотрены следующие действия: вычисление квадрата модуля комплексного числа; суммирование и вычитание, производимые одной инструкцией.

IV. Взаимодействие с внешней и внутрикристальной памятью

Сопроцессоры FPU и CP2 взаимодействуют с внешней памятью с помощью специальных команд загрузки/сохранения, при этом используя таблицу трансляции адресов (TLB), а также общую для ядра и сопроцессоров кэш-память данных.

Существенное влияние на производительность процессора оказывает подсистема памяти [1]. Поэтому, для минимизации времени доступа к данным, в процессоре предусмотрена более быстрая внутрикристальная память.

Внутрикристальная память процессора – физически это кэш-память данных второго уровня, состоящая из 16-ти банков объемом 16 Кбайт каждый, разделенная на 2 секции. Независимость секций позволяет одновременно выполнять загрузки/выгрузки из одной секции в регистровые файлы сопроцессоров и заполнять из внешней памяти другую секцию (либо считывать информацию из этой секции во внешнюю память) по каналам прямого доступа к памяти (ПДП). Сопроцессоры FPU и CP2 взаимодействуют с внутрикристальной памятью с помощью специальных команд загрузки/сохранения, не используя таблицу трансляции адресов (TLB) и кэш-память данных. Команды обращения к внутрикристальной памяти, а также механизм ПДП становятся доступны при отключении кэш-памяти второго уровня в специализированных управляющих регистрах.

Контроллер ПДП управляет пересылкой данных по каналам прямого доступа к памяти между внутрикристальной памятью и внешней памятью. Управление осуществляется набором управляющих регистров, которые отображены в адресном пространстве внутрикристальной памяти или доступны как управляющие регистры сопроцессора комплексных вычислений. Контроллер позволяет осуществлять трехмерную выборку данных из исходной области памяти и независимое трехмерное размещение в целевой области

памяти. Обмен производится 64-х разрядными словами. На рис. 1 изображена структурная схема описываемой системы при взаимодействии с внутрикристальной памятью.

V. Тесты разработчика

Первый этап в тестировании сопроцессоров в составе микропроцессора – это тесты разработчика, написанные вручную. Эти тесты позволяют проверить правильность реализации команд, корректность вызова исключений от отдельных инструкций, а также, в первом приближении, проверить некоторые зависимости между инструкциями и отмену инструкций сопроцессора при возникновении исключительной ситуации.

Ручная разработка тестовых программ достаточно часто используется для проверки, так называемых, крайних случаев. Инженеры-верификаторы, разрабатывающие такие тесты, должны знать детали реализации верифицируемого сопроцессора, чтобы создать соответствующие ситуации и проверить правильность поведения микропроцессора в них.

В случае проверки зависимостей все инструкции делятся на классы эквивалентности. При этом считается, что инструкции из одного класса абсолютно равноценны для тестирования. В каждый класс входят инструкции, имеющие общие параметры или характеристики, например, совпадающие по длительности выполнения инструкций, по количеству операндов, по количеству регистров результата. Есть также классы инструкций обмена данными с внешней памятью, с внутрикристальной памятью, с управляющими регистрами сопроцессоров, с регистрами общего назначения, а также инструкции обмена данными между регистрами сопроцессоров. При тестировании зависимостей инструкций разработчик перебирает комбинации команд из всех классов эквивалентности. Конечно, такой перебор нельзя считать исчерпывающим, так как аккуратно перебрать все варианты вручную, даже с учетом существования классов эквивалентности, практически невозможно. Тем не менее, эти тесты незаменимы для верификации проекта, так как на коротких тестах разработчику легче отладить модель.

VI. Программный эмулятор

Для проверки результатов прохождения теста необходимо их сравнить с некими «эталонными» результатами. Под «эталонными» результатами в данном случае понимаются:

- результаты выполнения инструкций FPU/CP2, не вызвавших исключение (результаты арифметических инструкций, команд загрузки/сохранения, модификации регистров; отмена выполнения команд при возникновении прерывания, при возникновении исключения от предыдущей инструкции, так как рассматриваемый процессор является суперскалярным).

- результаты выполнения инструкций FPU/CP2, вызвавших исключение (корректность вызова исключения, правильность установки статусных регистров, отсутствие изменения регистра результата инструкции, вызвавшей исключение).

Проверка всех этих результатов в тесте не всегда возможна в полной мере из-за большого количества тестовых случаев, сложности расчета результатов арифметических операций, и так далее. Тем не менее, для полноценной верификации проекта осуществление проверки результатов всех инструкций является обязательным условием.

Поэтому существенная часть методов верификации опирается на использование некоторых эталонных моделей процессора [2]. Среди этих методов наиболее распространен параллельный запуск тестовых программ на RTL-модели процессора и его эталонной модели и сравнение содержимого регистров и памяти в обеих моделях после каждого такта или после каждой инструкции. В качестве эталонной модели используются, как правило, эмуляторы, реализованные на языке программирования высокого уровня. Такие эмуляторы описывают процессор на уровне его программной модели и являются относительно простыми в разработке и отладке.

VII. Переборные тесты

Как было сказано ранее, тесты разработчика нельзя считать исчерпывающими. Тем не менее, перебрать все ситуации необходимо, ведь ошибка может содержаться в любой комбинации команд. Для этой цели используются генераторы переборных тестов, которые можно написать на языке высокого уровня, таких как Java, Perl. Можно также воспользоваться уже готовой технологией генерации тестовых программ, например, технологией UniTesk (Unified Testing & specification toolKit) [3], разработанной на языке Java. Данный подход использует высокоуровневые описания системы команд для формальной спецификации поведения микропроцессора и комбинаторные методы построения тестовых воздействий в тестовых программах.

Генератор создает набор тестовых программ. Каждая тестовая программа начинается с инициализирующей части и оканчивается заключительной частью. Внутренняя часть программы содержит набор тестовых вариантов, каждый из которых связан с некоторой частной целью тестирования. Основной частью тестового варианта является тестовое воздействие, то есть специально подготовленная последова-

тельность инструкций, нацеленная на создание определенной ситуации в работе микропроцессора. Тестовые воздействия генерируются с использованием комбинаторной техники – путем систематического перебора всех возможных комбинаций конкретных инструкций, тестовых ситуаций и зависимостей.

Генератор использовался для тестирования зависимостей, механизма ПДП, а также для перебора всех инструкций с некоторыми «специальными» операндами.

Существенным недостатком данной технологии является трудоемкий процесс проверки полноты покрытия (то есть проверки наличия всех возможных комбинаций команд в тестовой программе или наборе тестовых программ, запускаемых на проекте), так как критерии перебора задаются человеком нетривиальным способом, а проверка полученного результата не осуществляется. Требуемый уровень покрытия был достигнут с 23-ей итерации генерации тестовых программ. Тем не менее, преимуществом данной технологии является то, что тестовый набор со всеми логами занимает всего около 750 Мб и удобен для дальнейшего регрессионного тестирования.

VIII. Тестирование механизма ПДП

Отдельно рассмотрим тестирование контроллера ПДП, так как его верификация всеми изложенными выше способами имеет ряд особенностей. Это связано с тем, что, во-первых: невозможно перебрать все ситуации из-за их большого количества; во-вторых: случайные тесты могут задать некорректную комбинацию числа и величины шагов при вычислении адреса передачи.

Прежде, чем приступить к верификации механизма ПДП, нужно решить, какие требуются тесты, чтобы утверждать, что механизм ПДП работает корректно, изолированно от всей системы в целом. Одним из путей решения этого вопроса является определение узких мест в тестируемом блоке. В данной системе узкие места связаны с минимальным размером передаваемых данных. Для увеличения производительности при обращении во внешнюю память происходит выборка сразу восьми 64-х разрядных слов, а минимальный размер передаваемых по ПДП данных – это одно 64-х разрядное слово. Следовательно, контроллер ПДП должен отменить все ненужные передачи. Проверка корректности этой отмены и является основной задачей верификации механизма ПДП. Таким образом, осуществляется перебор не всех возможных комбинаций адресов и размера передаваемых данных, а перебор всех возможных смещений адреса приема/передачи данных по всем циклам передачи. При этом перебираемые размеры передачи не должны быть слишком большими.

IX. Генератор псевдослучайных тестов

Еще одним методом верификации является использование генератора псевдослучайных тестов.

Программы, построенные случайным образом, позволяют быстро обнаруживать сравнительно простые ошибки. Другое достоинство метода состоит в том, что с его помощью можно создать ситуации, которые сложно представить, но представляющие интерес для тестирования.

Именно, генератор псевдослучайных тестов является наиболее мощным средством верификации. Первая итерация запуска шаблонов на вычислительном кластере Московского Государственного Университета дала 75% измеряемого покрытия за трое суток работы кластеров. При этом были найдены ошибки на ситуациях, не отслеживаемых имеющимися средствами оценки покрытия. После усовершенствования шаблонов, используемые методы оценки покрытия показали 93% .

Кроме того, в шаблоны псевдослучайных тестов добавлен запуск ПДП передачи как линейной, так и трехмерной. Таким образом, на фоне случайных тестов с командами обращения во внутрикристальную память осуществлялась ПДП передача. Это увеличило процент найденных ошибок, возникавших из-за наложения адресов обращения во внутрикристальную память командами загрузки/сохранения и ПДП передачей.

X. Заключение

Таким образом, в ядро микропроцессора были интегрированы сопроцессоры, позволяющие решать такие вычислительные задачи, как преобразование Фурье, умножение матрицы на вектор, арифметика с комплексными числами и так далее.

Описанный подход к тестированию сопроцессоров в составе микропроцессора позволил в короткий срок получить работоспособную версию, готовую для дальнейшего тестирования на ПЛИС прототипе.

Литература

[1] Cragon H.G. Memory Systems and Pipelined Processors // Jones and Bartlett Publishers. – 1996. – 576 p.

[2] Лесных А. А., Широков И.А. Покомандная модель проектируемого 64-битного процессора / Сб. статей под ред. академика РАН В.Б. Бетелина. – М.: НИИСИ РАН, 2005. – С. 96 – 116.

[3] http://www.citforum.ru/SE/testing/unitesk_hard/.

[4] Аряшев С. И., Зубковский П.С., Николина Н.В., Чибисов П.А. Основные подходы к верификации блока вещественной арифметики // Проблемы разработки перспективных микроэлектронных систем – 2005. Сб. научных трудов / под общ. ред. А.Л. Стемпковского. – М.: ИППМ РАН, 2005. – С. 269 – 274.