# 10 причин перехода на конфигурируемые процессорные ядра

**СТИВ РОДДИ (STEVE RODDY)**, вице-президент по маркетингу, Tensilica Inc.

***В статье обсуждаются преимущества конфигурируемых процессоров, которые позволяют повысить гибкость, улучшить эффективность архитектуры, оптимизировать процесс проектирования и уменьшить потребляемую мощность конкретного приложения. Статья представляет собой перевод [1].***

Конфигурируемые процессорные ядра впервые появились на рынке в конце 1990-х гг., и с тех пор внедрено множество более совершенных методов настройки процессора под задачи конкретных приложений. Многие процессы в настоящее время автоматизированы, что позволило ускорить цикл адаптации процессора к требованиям заказчика и гарантировать безопасность процессорного ядра в процессе настройки. В чем преимущество создания собственной конфигурации процессорного ядра перед использованием серийно выпускаемого процессора?

### ПРИЧИНА 1: УМЕНЬШЕНИЕ ПОТРЕБЛЯЕМОЙ МОЩНОСТИ

Некоторые специалисты считают, что все RISC-процессоры обеспечивают примерно одинаковую производительность в расчете на тактовый цикл. Это предположение неверно для процессорных ядер, сконфигурированных специально для конкретного приложения. С помощью настройки процессора можно существенно улучшить производительность для каждого тактового цикла.

Разработчик может ввести специализированные команды в структуру набора команд (instruction set architecture — ISA), которые увеличат размер ядра процессора, что, в свою очередь, вызовет рост средней мощности, рассеиваемой процессорным ядром за тактовый цикл. Однако если новые команды существенно уменьшат общее число тактовых циклов, необходимых для выполнения данного задания, то общая потребляемая мощность (мощность за тактовый цикл, умноженная на общее время цикла) может быть в значительной степени снижена. Например, 20%-е увеличение рассеиваемой мощности за тактовый цикл, скомпенсированное за счет 3-кратного увеличения скорости выполнения задания, снижает общее энергопотребление на 60%.

### ПРИЧИНА 2: ИСКЛЮЧЕНИЕ НЕОБХОДИМОСТИ В RTL-ВЕРИФИКАЦИИ

Какова альтернатива встраиванию блока ускорения в процессор? При использовании стандартных RISC-процессоров ответом на этот вопрос является создание собственных RTL-блоков ускорения. Это приводит к затратам на длительный и сложный процесс верификации. В большинстве RTL-блоков конечный автомат содержит только команды управления. И большая часть риска при проектировании и верификации системы связана именно с конечным автоматом из-за его сложности.

Внесение изменений в RTL-блоки ускорения на последних стадиях разработки скорее повлияет на конечный автомат, чем на канал данных, т.к. конечный автомат представляет собой наиболее сложную часть проекта. Конфигурируемые процессоры, подобно RTL-блокам, содержат информационные каналы с большим числом разрядов, что гарантируется вендорами процессоров, и в то же время снижаются риски, связанные с проектированием конечных автоматов, т.к. они на базе процессоров создаются на микропрограммном уровне.

Введение дополнительных функций в процессорное ядро компании Tensilica не нарушает базовый набор команд Xtensa, гарантируя тем самым возможность применения надежной экосистемы прикладного ПО и инструментов разработки сторонних производителей. Конфигурируемые ядра процессора Xtensa совместимы с основными ОС, инструментами отладки и ICE-решениями. Они поставляются с полным, автоматически генерируемым комплектом программных инструментов разработки, который содержит усовершенствованную интегрированную среду разработки на базе Eclipse; оптимизированный векторизующий компилятор мирового класса; точный SystemC-совместимый симулятор набора команд; полный набор инструментов GNU, принятый в качестве отраслевого стандарта, а также скрипты EDA-синтеза.

Конфигурируемые процессоры могут быть использованы в качестве альтернативы RTL-блокам, кодируемым вручную, благодаря созданию тех же конфигураций каналов данных, которые были реализованы в RTL-блоках ускорителей. Такие конфигурации каналов данных включают длинные конвейеры (deep pipelines), блоки параллельного исполнения, специализированные регистры состояния и шины данных с большим числом разрядов для связи с локальной и глобальной памятью.

### ПРИЧИНА 3: ПРОЦЕССОР ВАШ И ТОЛЬКО ВАШ

Если вы создали процессор, сконфигурированный уникальным образом, то конкурентам значительно труднее скопировать ваши идеи. Вы получили более высокую производительность и более низкую потребляемую мощность процессора. Никто кроме вас не сможет получить автоматически сгенерированный комплект программных инструментов, поэтому никто не сможет запрограммировать процессоры на вашей ASIC, если вы не позволите сделать это.

Даже если кто-нибудь получит сконфигурированный вами процессор, он не сможет воспользоваться преимуществами оптимизированной вами структуры команд, если вы не предоставите необходимые для этого программные инструменты. К тому же, ваш оптимизированный процессор будет иметь лучшую производительность, работать на более низкой тактовой частоте и потреблять меньше энергии, чем стандартные микропроцессорные ядра с фиксированной структурой набора команд.

Беспокоитесь, что другие не знают, как запрограммировать процессор? Вы будете удивлены узнав, насколько редко к конфигурируемым процессорам обращается основная ОС, после того как установлено микропрограммное обеспечение. Например, созданная однажды микропрограмма для видеоприложения продолжает работать без каких-либо изменений. То же самое относится и к программам для аудиоприложений или коммуникационным программам. После того как вы

оптимизировали микропрограммное обеспечение, уникально сконфигурированные процессоры способны выполнять такие задачи, которые больше не нужно программировать.

### ПРИЧИНА 4: АВТОМАТИЗИРОВАННЫЙ ПРОЦЕСС КОНФИГУРИРОВАНИЯ

Процесс создания ядра процессора для специализированного приложения автоматизирован. Доступны компиляторы, которые могут проверить код на языке C для специальной задачи или алгоритма и предложить расширение процесса, позволяющее ускорить это задание или алгоритм. Такие компиляторы способны обеспечить почти мгновенную обратную связь с командой разработчиков, что может значительно сократить цикл проектирования.

Например, компилятор XPRES от Tensilica может автоматически анализировать код C, идентифицировать критичные внутренние циклы. Компилятор создает граф, показывающий различные варианты соотношения скорости исполнения кода и числа необходимых вентилей для реализации анализируемого кода. Это позволяет разработчикам принимать обоснованное решение при выборе между использованием дополнительных логических элементов (площадь) и увеличением производительности (число циклов). Процесс разработки включает следующие этапы.

1. Компиляция исходного приложения на C/C++ и запуск компилятора XPRES.

2. Выбор наилучшей конфигурации для заданного числа вентилей и при желании — оптимизация сгенерированной конфигурации вручную.

3. Построение процессора с использованием стандартного набора инструментов компании Tensilica — Xtensa Processor Generator. Комплект инструментов Xtensa Processor Generator создает RTL-описание конфигурируемого процессора и генерирует специальные версии всех необходимых программных инструментов разработки, включая компилятор, ассемблер, отладчик и симулятор набора команд. Кроме того, он генерирует логическую модель на C или SystemC для процессора и скрипты EDA-синтеза. Чтобы получить полное соответствие между программными инструментами разработки и процессором, никакой ручной работы не требуется.

4. Компилирование исходного немодифицированного кода на C для запуска на сконфигурированном ядре процессора.

Заметим, что если разработчик ASIC использует компилятор XPRES для создания конфигурируемого процессора, нет необходимости модифицировать исходный код или любой другой код на C для того, чтобы использовать новые команды. Компилятор автоматически создаст эти новые команды.

### ПРИЧИНА 5: БЕЗОПАСНОСТЬ

Когда вы создаете свой проект или его часть, то на стандартном процессорном ядре намного легче сделать его копию. В конечном итоге кто-нибудь может использовать то же самое процессорное ядро в своем проекте. Но когда вы конфигурируете процессор под свое приложение, то каким образом кто-либо сможет скопировать те изменения и настройки, которые сделали вы? Ядро ваше и только ваше.

Когда вы конфигурируете процессор Xtensa компании Tensilica, то получаете соответствующий комплект программных инструментов, которые оптимизированы под ваши изменения. В то время как компилятор для других процессоров способен запустить ваше ПО, эти процессоры не могут достичь уровня производительности, реализуемого с помощью оптимизированного вами компилятора на оптимизированном вами же процессорном ядре.

В процессе оптимизации вводятся специальные регистры (размер которых соответствует типам данных для заданий, которые необходимо выполнить) и блоки исполнения, которые эффективно выполняют специализированные алгоритмы, зачастую сразу за два тактовых цикла. Такой метод проектирования помогает сохранить тактовую частоту и энергопотребление на низком уровне.

Если вы не разработчик процессоров и не знаете, каким образом ввести регистры и блоки исполнения в процессор, то, возможно, вы знакомы с Verilog. Используя автоматизированный процесс от компании Tensilica, необходимо написать несколько строк кода, подобного Verilog.

### ПРИЧИНА 6: ПРОЕКТИРОВАНИЕ БЕЗ СИСТЕМНЫХ ШИН

Главная шина процессорного ядра является, как правило, узким местом передачи данных. Однако имеется новое поколение процессоров, которые позволяют уменьшить загрузку главной шины путем использования других средств повышения производительности линий ввода/вывода. Процессоры Xtensa от Tensilica позволяют достичь скорости передачи данных, которая может быть сравнима со скоростью вручную спроектированных RTL-блоков. Есть четыре способа непосредственной коммуникации без использования главной системной шины.

1. XLMI (Xtensa Local Memory Interface) — это простая, быстрая, однотактная шина, которая выполняет пересылку данных намного быстрее, чем главная системная шина, потому что она не предназначена для поддержки множества ведущих устройств на шине (bus master). Шина может быть сконфигурирована вплоть до 128-битной разрядности, что обеспечивает пиковую полосу пропускания 3,2 Гбит/с с малой задержкой передачи данных.

2. Порты действуют как линии ввода/вывода общего назначения (GPIO) и связаны таким образом, чтобы непосредственно соединить два процессора Xtensa или подключить процессор Xtensa к внешнему RTL-блоку. Порты могут содержать до 1024 линий, что позволяет передавать данные с большим числом разрядов без необходимости выполнения множества операций загрузки/хранения. Порты особенно полезны для передачи управляющей информации и данных о состоянии.

3. Список очередности обслуживается по принципу FIFO и обеспечивает высокоскоростной механизм передачи потоков данных без буферизации (см. рис. 1). Входные и выходные очереди работают, с точки зрения программиста, подобно обычным регистрам процессора без ограничений по полосе пропускания для доступа к локальной и системной памяти. Очереди могут поддерживать скорости передачи данных до одной команды пересылки за каждый тактовый цикл или более 350 Гбит/с для каждого списка очередности, введенного в процессор Xtensa.

4. Интерфейсы просмотра памяти (Memory Lookup Interfaces) полезны при подсоединении RAM для поиска данных по таблице или при подсоединении аппаратных блоков вычислений с длительной задержкой (см. рис. 2). Память, подсоединенная к этим просмотровым интерфейсам, может быть считана или записана непосредственно из канала данных процессора без использования команд загрузки и хранения.

Все эти функции, легко определяемые разработчиком, автоматически вводятся в процессор Xtensa и полностью моделируются инструментальным средством Xtensa Processor Generator компании Tensilica. Процесс автоматизирован с помощью технологии, запатентованной компанией Tensilica, он изначально проверен, и повторной верификации процессора не требуется.

### ПРИЧИНА 7: СООТВЕТСТВИЕ ПРОЕКТА ТРЕБОВАНИЯМ ЗАВТРАШНЕГО ДНЯ

Во многих отраслях часто происходят изменения стандартов, и в следующих или даже в существующих проектах необходимо реализовывать новые алгоритмы. Если вы создаете жёстко заданные алгоритмы в RTL, то после того как чип изготовлен, вносить изменения в него нельзя. Однако если вы реализуете проект в процессоре, можно ввести

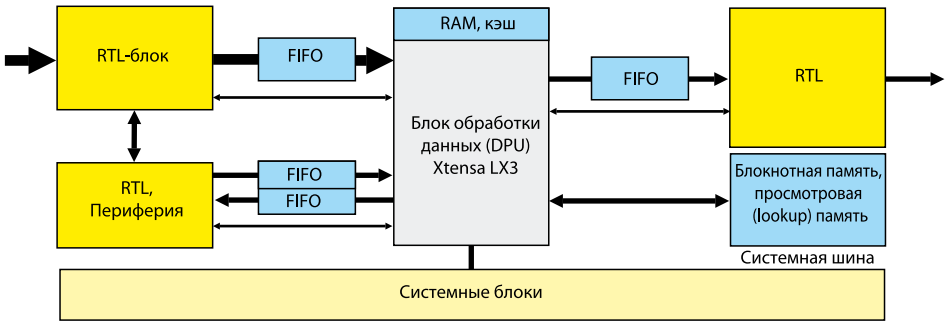


*Рис. 1.* ***Пример непосредственных соединений FIFO и порта с использованием TIE*** *(Tensilica Instruction Extension)* ***очередей и TIE-портов***

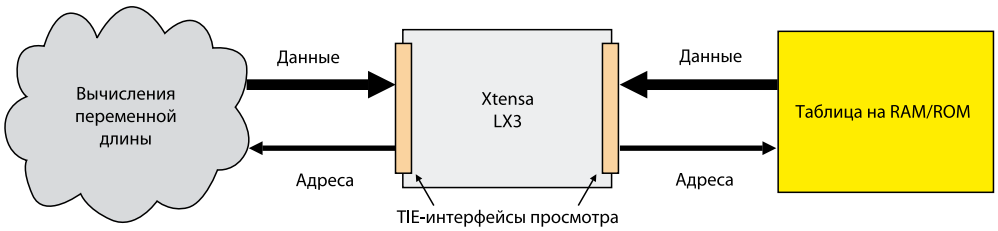


*Рис. 2.* ***Пример выполнения TIE-поиска данных по таблице, показывающий соединение памяти с логикой***

изменения в микропрограммное обеспечение после того, как кристалл изготовлен. Обычно это является причиной использования процессора вместо RTL. Но почему это особенно важно при использовании конфигурируемого процессора? Потому что можно встроить в него логику, которую предполагалось поначалу реализовать во внешнем RTL-блоке, прямо в канале данных процессора. Можно сконфигурировать канал данных точно на ту разрядность, которая требуется. Если необходимо обрабатывать 56-разрядные данные, то это можно делать одновременно, не используя две 32-разрядные операции. В результате процессор становится намного более эффективным, и необходимость выгрузки заданий в специализированные аппаратные RTL-блоки существенно снижается, если совсем не исчезает. Следовательно, те функции, которые раньше были реализованы в RTL-блоках, теперь являются программируемыми в самом процессоре.

### ПРИЧИНА 8: ЛУЧШЕ СООТНОШЕНИЕ ЗАНИМАЕМОЙ ПЛОЩАДИ И ПРОИЗВОДИТЕЛЬНОСТИ

Думаете о переходе на более крупное ядро из-за необходимости увеличения производительности? Более крупный процессор не только занимает большую площадь на кристалле, но и потребляет больше энергии. Что, если вместо этого начать с высокоэффективной базовой архитектуры и увеличить производительность там, где это требуется? Вы получите процессор с оптимальной производительностью вместо более крупного процессора общего назначения с чрезмерной вычислительной мощностью.

### ПРИЧИНА 9: ВАШИ ИНЖЕНЕРЫ БУДУТ РАБОТАТЬ БОЛЕЕ ЭФФЕКТИВНО

Конечно, вы могли бы спроектировать собственный процессор или DSP или разработать собственные функции ускорения в RTL. Но цикл разработки обычно занимает более года, плюс потребуется время для разработки соответствующего комплекта программных средств — компилятора, отладчика, симулятора набора команд и т.д. И нет гарантии, что характеристики процессора или RTL-проекта, которые вы выбрали для реализации, являются наилучшими.

Вместо этого используются конфигурируемый процессор и сопоставляются разные варианты архитектур. Затем, когда вы получили желаемое соотношение между производительностью, занимаемой площадью, потребляемой мощностью, можно положиться на автоматизированный процесс от компании Tensilica, чтобы сгенерировать полный комплект программных инструментов, точно соответствующий спроектированному процессору. Сотни разработчиков прошли этот путь, и компания Tensilica гарантирует, что программные инструменты, а также процессор не содержат ошибок.

При этом численность инженеров, занятых в процессе проектирования и верификации заказных RTL-блоков ускорения, существенно уменьшается. Метод разработки ASIC на базе процессоров допускает постепенное восстановление графика работ по проекту, когда обнаруживается ошибка. Кроме того, трудоемкий цикл RTL-верификации исключается. Следует только проверить, что процессор выполняет те функции, которые вы задали, а не заниматься процессом верификации аппаратно-реализованных RTL-блоков.

### ПРИЧИНА 10: ПОТОМУ ЧТО ВЫ ЭТО УМЕЕТЕ

Почему бы, используя автоматизированные инструменты разработки, не попытаться сконфигурировать процессор, точно отвечающий требованиям вашего приложения? Вас несомненно порадуют гибкие возможности, которые обеспечат требуемую эффективность решения на базе программируемого процессора. При помощи инструментов компании Tensilica вы сможете проверить различные конфигурации и варианты проекта для того, чтобы добиться наилучшего сочетания занимаемой площади, потребляемой мощности и производительности, которые точно подходят для приложения.

Больше нет необходимости беспокоиться, что вы не являетесь специалистом по разработке процессоров. Даже если вы никогда раньше не проектировали процессор, но знаете, что необходимо приложению (то, что вам следовало реализовать в RTL-блоке в старом проекте), вы сможете использовать автоматизированный процесс от компании Tensilica для создания процессора.

Вам не нужно обращаться к RTL-описанию процессора и делать какие-либо правки. Вместо этого достаточно использовать язык, подобный Verilog, для описания функций, которые вы хотели бы ввести. Затем автоматизированный процесс сгенерирует расширения процессора, которые будут отвечать вашим требованиям, с соответствующим комплектом программных инструментов.

***ЛИТЕРАТУРА***

*1. Steve Roddy. 10 Reasons to Customize A Processor Core//www.electronicscomponentsworld.com.*