Мультиклеты – новое слово в микропроцессорах

Борис Зырянов, Николай Стрельцов, Дмитрий Кукушкин, Михаил Власов, Михаил Бахтерев, Наталья Горностаева (г. Екатеринбург)

В статье представлен новый тип процессора с мультиклеточной архитектурой, позволяющей увеличить производительность в 4–5 раз при одновременном снижении энергопотребления и уменьшении площади кристалла.

История создания

Работы по созданию прототипа мультиклеточного процессора, получившего название синпьютер (автор архитектуры – Николай Стрельцов), велись с 2000 г. В марте 2003 г. разработка была представлена на ежегодной международной конференции по цифровой обработке сигналов International Signal Processing Conference (ISPC) в Далласе (США) от имени фирмы SYCS ApS – Synergetic Computing Systems A/S (Дания) и была отмечена в номинации «Лучший продукт года». С 2004 г. работы продолжились в Уральской архитектурной лаборатории, и в 2006 г. проект мультиклеточной архитектуры стал победителем российского конкурса инноваций в номинации «Белая книга» как новаторский проект, имеющий прорывной характер.

В 2008 г. Николай Стрельцов представил на рассмотрение фонда «Инновационные технологии» проект создания процессоров с принципиально новой, универсальной мультиклеточной архитектурой, и руководством фонда было принято решение об участии в проекте. Год спустя прототип мультиклеточного процессора был показан на Женевском салоне инноваций (Salon international des inventions de Geneve), который является партнёром фонда «Инновационные технологии», а ещё через год, в 2010 г., была основана корпорация «Мультиклет» на основе объединения интеллектуальной собственности Уральской архитектурной лаборатории и фонда «Инновационные технологии». Компанию возглавил доктор технических наук Борис Зырянов, должность технического директора занял Николай Стрельцов. Началась подготовка к выпуску мультиклеточного процессора. В 2011 г. код RTL был опробован на FPGA spartan-6.

Опытная партия 4-клеточных процессоров на кристалле МСр0411100101 (получившая впоследствии название MULTICLET P1) была выпущена по технологии 180 нм в июне 2012 г. В разработке топологии участвовал дизайн-центр «Цифровые решения» (г. Москва). Мультиклеточный процессор MULTICLET P1 успешно прошёл испытания в температурном диапазоне –60...+125°С в ОАО «ВЗПП-С» (г. Воронеж) на выборке из 20 микросхем.

С 2011 г. ОАО «Мультиклет» является резидентом инновационного центра «Сколково» с проектом создания процессоров с мультиклеточной архитектурой (кластер «Космические технологии и телекоммуникации»).

Введение в мультиклеточную архитектуру

Появление мультиклеточной архитектуры было обусловлено тем, что основополагающая фон-неймановская архитектура, по мнению аналитиков International Technology Roadmap for Semiconductors (ITRS), исчерпала свой потенциал развития. В последние два десятилетия эффективность использования кремниевых ресурсов постоянно снижается, а производительность в пересчёте на один транзистор падает. Резко возрастает сложность проектирования, а затраты на создание новых моделей процессоров приближаются к стоимости разработки современных самолётов. Каждая новая топологическая норма усугубляет эту ситуацию и актуализирует поиск новых процессорных архитектур.

Начиная со времён первых вычислительных машин, основной тенденцией развития фон-неймановской архитектуры было увеличение уровня параллелизма при выполнении потока команд, т.е. стремление обойти главный принцип этой архитектуры – упорядоченное, последовательное размещение команд в программе и их исполнение в порядке размещения. Необходимым условием реализации этого принципа и, как следствие, опосредованной формы информационных связей между командами в фон-неймановской архитектуре является отчуждение результата очередной команды с последующей записью в общедоступную память машины (регистры, ЗУ). Только после этого результат доступен программисту и может использоваться им в качестве операнда для последующих команд.

Наиболее известные попытки уйти от фон-неймановской архитектуры – это потоковые и редукционные машины. Принцип построения потоковых процессоров – адресная рассылка. Каждое командное слово такого процессора, как правило, содержит код операции, поля операндов, адрес командного слова, которому передаётся результат выполненной операции, и номер операнда, в поле которого будет помещено полученное значение. В результате последовательность команд в программе не упорядочена и, в принципе, может быть размещена в памяти программ произвольным образом. Команда выбирается и выполняется «по готовности операндов», т.е. когда получены все результаты выполнения других команд, необходимые для её выполнения. Принцип исполнения команд потокового процессора «по готовности операндов» позволяет реализовать параллелизм «естественным» образом – без решения задачи распараллеливания, но требует использования специальных языков программирования (с однократным присваиванием).

Редукционная модель процессора также использует адресную рассылку, но она задаётся в командном слове указанием адреса команды, результат выполнения которой используется в качестве операнда. В этом случае команды выби-

раются из памяти и исполняются «по запросу команды». Такая модель также позволяет отказаться от упорядоченности команд – они могут быть размещены в памяти произвольным образом. Как и потоковая, редукционная модель обеспечивает «естественную» реализацию параллелизма, но при этом требует использования специальных языков функционального программирования. Обе модели не смогли составить реальной конкуренции фон-неймановским процессорам, хотя их отдельные решения, например исполнение команд «по готовности», используются в современных проектах.

На данный момент существует достаточно много вариантов реализации фон-неймановской архитектуры, которые обеспечивают повышение производительности процессора, но делают это разными методами. Например, конвейерные процессоры – путём совмещения выполнения потока команд во времени при использовании одного исполнительного устройства; RISC-процессоры – сокращая объём реализуемых операций и время выполнения одной команды; CISC-процессоры – увеличивая объём операций одной команды, но уменьшая общее число выполняемых команд.

Суперскалярные, многоядерные и VLIW-процессоры обеспечивают повышение производительности за счёт совмещения выполнения команд в пространстве, т.е. на нескольких исполнительных устройствах. При этом в суперскалярных процессорах задача распределения потока команд по исполнительным устройствам решается преимущественно аппаратным способом, а в многоядерных и VLIW-процессорах – программным.

В конкретных реализациях эти и другие методы могут сочетаться. Процессоры RISC и CISC обычно имеют конвейерную организацию, но отличаются длиной конвейера. Конвейер широко используется для сокращения времени выполнения потока команд, поступающего на исполнительное устройство в суперскалярных процессорах. Наиболее совершенные способы организации конвейера не только реализуют совмещение выполнения потока команд во времени, но и обеспечивают переупорядочивание команд для устранения конфликтов данных между ступенями конвейера. В этом случае команда выдаётся на исполнение не по очереди, а по готовности. Наиболее известны две схемы, реализующие такое переупорядочивание команд: централизованное окно команд и распределённая схема Томасуло.

Сочетанием методов отличается и проект TRIPS, который авторы относят к архитектуре EDGE, – явное исполнение графа потока данных. Кристалл имеет два функциональных блока, каждый из которых фактически является VLIW-процессором, содержащим 16 (4 × 4) 64-разрядных АЛУ с плавающей точкой, которые работают как потоковая машина. От традиционной потоковой машины они отличаются тем, что команды загружаются во все АЛУ одновременно, но выполняются как в потоковой машине – асинхронно, по готовности операндов.

Следует отметить, что использование методов распараллеливания в суперскалярных или VLIW-процессорах и методов переупорядочивания команд в конвейере, которые меняют очерёдность исполнения команд, или методов исполнения команд «по готовности» не отменяет ключевого принципа фон-неймановской модели – упорядоченного изменения состояния процессора. Вне зависимости от очерёдности исполнения команд, формируемые ими состояния процессора реализуются в той последовательности, в которой эти команды были размещены в памяти, а не исполнены. Этот принцип ограничивает возможности указанных методов и увеличивает аппаратные затраты на их реализацию, а также требует решения сложной задачи распараллеливания при использовании более чем одного процессора для выполнения программы.

Принципиальное отличие мультиклеточной архитектуры от фон-неймановской и потоковой – использование непосредственных информационных связей между командами и широковещательная рассылка их результатов. Как известно, моделью вычислений в императивных языках высокого уровня является выполнение упорядоченной последовательности операторов. Каждый оператор представляет собой неделимую и целостную языковую конструкцию, описывающую процесс преобразования данных. Порядок выполнения операций внутри оператора задаётся путём их ранжирования и расстановки скобок, т.е. указанием непосредственных информационных связей между операциями. Промежуточные результаты вычислений внутри оператора не отчуждаются, а программисту виден только результат выполнения оператора. Следовательно, для абстрактной машины, непосредственно реализующей некоторый язык высокого уровня, оператор языка является командой.

Традиционно под командой понимается то неделимое и целостное действие вычислительной машины, которое видимо и доступно программисту. Оно не может быть задано или выполнено частично (неделимость и целостность). Результат данного действия фиксируется путём изменения состояния машины, которое далее может использоваться другими командами (видимость). Программист может задавать это действие в процессе программирования (доступность).

В машинах с фон-неймановской архитектурой система команд, которая используется при программировании, полностью эквивалентна той, которая реализует исполнительные устройства данной машины. Для виртуальной машины, реализующей некоторый язык высокого уровня на аппаратном уровне, ситуация принципиально иная.

Так, исходное множество операций любого алгоритмического языка зафиксировано и конечно. Множество операторов, которые теоретически могут быть сконструированы с использованием данных операций, является потенциально бесконечным. Поэтому машина с архитектурой, непосредственно реализующей язык высокого уровня, будет иметь конечный набор команд исполнительных устройств для множества операций языка, но не будет иметь фиксированной системы команд на уровне машины. Например, команда сложения, исполняющая соответствующую операцию, является таковой только внутри машины, но она не является оператором и, соответственно, командой при программировании, где она может быть только составной частью арифметического выражения, используемого в условных операторах или операторах присваивания.

Устранение этого семантического разрыва между уровнем программирования (оператор) и уровнем исполнения (последовательность команд) для фон-неймановских машин традиционно решается в процессе компиляции. Однако решение этой задачи аппаратными средствами в процессе исполнения более эффективно, поскольку появляется степень свободы в формировании потока команд, поступающего на

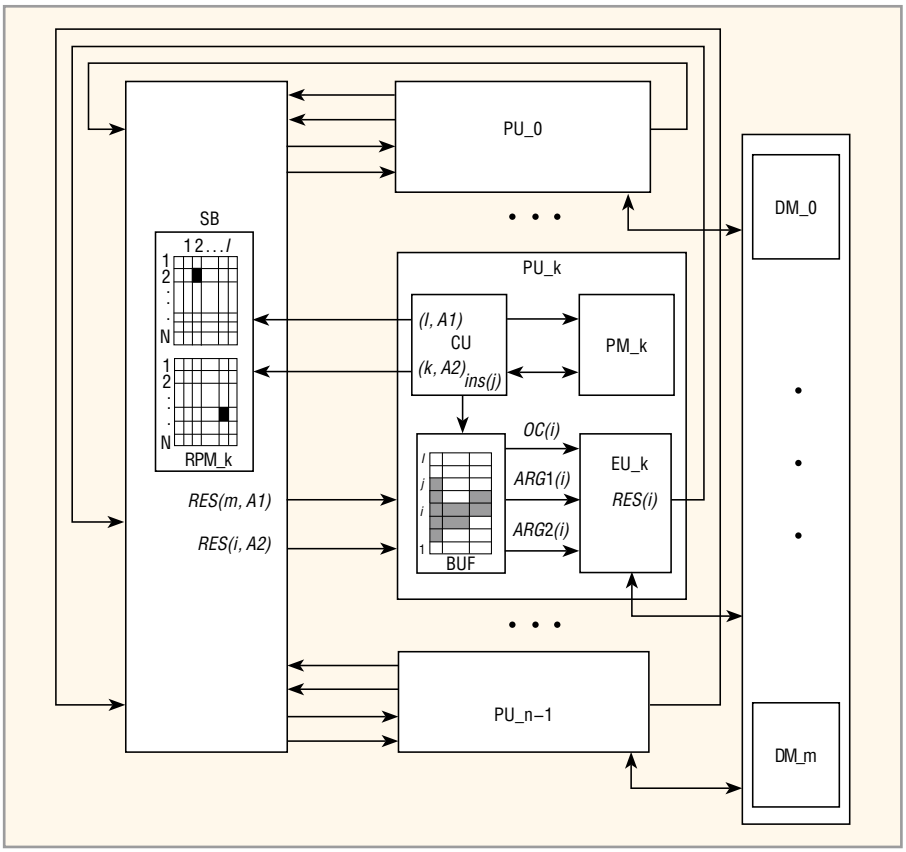

Рис. 1. Структурная схема мультиклеточного процессора

исполнение. Это позволяет сформировать поток оптимальным образом, исходя из текущего состояния машины.

Описанное решение легло в основу принципиально нового направления – мультиклеточных процессоров, качественные и количественные характеристики которых позволяют говорить о появлении нового класса архитектур. Система команд мультиклеточного процессора основана на промежуточном представлении компилируемой программы после синтаксического анализа (триады) и, фактически, является аппаратной реализацией входного языка программирования.

Каждая команда именуется, и информационные связи между ними задаются указанием имени команды, результат которой используется в качестве операнда. Имя команды формируется динамически при её выборке и представляет собой индивидуальный цифровой тег (признак). Фактически, значение тега – это номер команды в последовательности выбранных команд. При таком способе формирования тега указание на команду может задаваться относительным смещением команды, т.е. если *j*-команда использует результат ранее выраженной *i*-команды, то в поле операнда будет записано значение (*j* – *i*). Максимальное значение смещения и окно видимости результатов определяются размерностью поля операнда. Значение тега присваивается циклически, и оно должно быть больше окна видимости.

Основной программируемой единицей в мультиклеточном процессоре является параграф – замкнутая, информационно связанная группа команд, которые не имеют ссылок на результаты команд других параграфов. Информационный обмен между параграфами осуществляется только через память. Команды в параграфе могут располагаться в любой последовательности, но для минимизации аппаратных затрат на их размещение наложено требование частичной упорядоченности: все команды параграфа размещаются последовательно, а источники результатов – раньше их потребителей.

Состояние мультиклеточного процессора, видимое программисту, формируется по завершении параграфа. В этом смысле параграф эквивалентен команде в фон-неймановском процессоре или оператору (линейной последовательности операторов) в виртуальной машине, непосредственно реализующей язык высокого уровня. Параграфы в программе не упорядочены. Переход от одного параграфа к другому выполняется только командами передачи управления. Поскольку состояние процессора меняется по завершении параграфа, то команда, формирующая адрес следующего параграфа, может занимать любое место в последовательности команд, образующих параграф.

Структурная схема мультиклеточного процессора выглядит следующим образом (см. рис. 1), выполняя функции:

- рассылка всех результатов всем;
- отбор требуемых результатов;
- согласованная выборка команд;
- информирование всех устройств управления о состоянии выборки (начало и конец линейного участка, адрес следующего участка);
- сквозная адресация PM и DM, а также доступ к DM любой клетки.

Команды параграфа размещаются последовательно в памяти программ. Выбираются команды клетками – группами по *n* команд, где *n* – количество клеток, выполняющих данную программу. Сначала одновременно выбираются первые *n* команд, потом вторые и т.д. Очередной выбранной группе присваивается очередное значение тега группы (T_g). Выбранные команды размещаются в буферах клеток до исполнения. Каждая выбранная команда и, соответственно, её результат именуются: результату присваивается индивидуальный номер, равный $T_g\&N_c$, где N_c – логический номер клетки, выбравшей данную команду – число в диапазоне от 0 до (n – 1). По значению ссылки формируются номера запрашиваемых результатов, которые сообщаются коммутационному узлу клетки. Для мультиклеточных процессоров с $n = 2^{**}k$ и частично упорядоченным размещением команд (команда-источник размещается раньше команды-приёмника), номер запрашиваемого результата определяется следующим образом: $N_r = ((T_g\&N_c)\text{-}P_{op}) \bmod (T_g^{max})$, где N_r – номер запрашиваемого результата; T_g – максимальное значение тега группы; P_{op} – значение ссылки. Следует отметить, что младшие k бит N_r содержат физический номер клетки, от которой должен поступить запрашиваемый результат. Этот номер используется при выдаче запроса узлу коммутации на отбор результата.

Каждый полученный результат поступает всем коммутационным узлам, которые из общего потока по номеру отбирают результаты, требуемые данной клетке, и передают их процессорному устройству. Процессорное устройство исполняет команду, когда выполнены два условия: получены

запрошенные результаты и выбраны все команды-приёмники её результата.

Особенности мультиклеточной архитектуры

Принципиальными отличиями мультиклеточной архитектуры от известных фон-неймановской и потоковой архитектур являются использование непосредственного указания информационных связей между командами и полная независимость объектного программного кода от количества клеток, а также снятие требования упорядоченного размещения команд в программе.

Непосредственное указание информационных связей между операциями обеспечивает возможность одновременного чтения и исполнения нескольких команд без анализа их очерёдности выполнения, т.е. обеспечивает «естественную» реализацию параллелизма, которая изначально обусловлена видом и механизмами исполнения команд. В мультиклеточном процессоре нет аппаратных средств выявления информационных связей между выбранными операциями (командами) и их распределения по функциональным устройствам, т.е. динамическое распараллеливание отсутствует. Нет и статического распараллеливания, т.к. программа в виде триад хотя и описывает информационные связи, но имеет линейную форму и не содержит каких-либо указаний, что и как можно выполнять параллельно. В этом состоит принципиальное отличие от фон-неймановской модели.

Благодаря этой особенности потенциально обеспечивается живучесть мультиклеточного процессора, т.е. возможность непрерывного исполнения программы без перекомпиляции или перезагрузки при отказах его отдельных клеток (деградации процессора). Деградация связана с потерей производительности и увеличением времени решения задач. Но для целого ряда встроенных применений живучесть мультиклеточного процессора позволяет управляемому объекту выполнять основные функции либо за счёт снижения их качества, либо за счёт отказа от решения второстепенных задач [1]. Подобная независимость кода от используемых ресурсов создаёт основу для непрерывной самоадаптации процессора к потоку задач, а также его самовосстановления после сбоев или подключения новых ресурсов.

Неупорядоченность команд в параграфе позволяет при необходимости получать после каждой компиляции индивидуальный объектный код для каждого процессора. Это резко ограничивает возможности незаметного и несанкционированного вмешательства извне в работу системного программного обеспечения. Индивидуальность системного кода позволяет создать эффективную защиту от вредоносных программ.

Полносвязная интеллектуальная коммутационная среда, работающая в режиме «широковещательной» рассылки, не накладывает каких-либо топологических ограничений на межклеточный обмен данными и обеспечивает эффективную реализацию любого класса задач, что придаёт универсальность архитектуре и допускает эффективное масштабирование процессора. При увеличении количества клеток и наличии потенциального параллелизма алгоритма рост производительности процессора практически пропорционален количеству клеток.

Применение в качестве прототипа языка триад делает возможным использование общепринятых языков высокого уровня, таких как C и C++, что наряду с ассемблером минимизирует риск, связанный с адаптацией наработанного программного обеспечения к новой аппаратной платформе.

Асинхронная и децентрализованная организация мультиклеточного процессора как на системном уровне – между клетками (при реализации параллелизма), так и на внутриклеточном уровне – между блоками клетки (при реализации команд), дополнительно обеспечивает:

- минимизацию номенклатуры объектов проектирования и уменьшение их сложности;
- уменьшение площади кристалла процессора (объём оборудования при децентрализованном управлении меньше, чем при централизованном);
- увеличение производительности и сокращение энергопотребления за счёт реализации более эффективного вычислительного процесса;
- использование индивидуальной системы синхронизации для каждой клетки при реализации на одном кристалле десятков и сотен клеток.

В результате получается хорошо структурированная модульная система, позволяющая резко уменьшить сложность процессора, снизить затраты и повысить качество проектирования. При этом, по сравнению с фон-неймановской моделью, улучшаются количественные характеристики процессора.

Программное обеспечение

Программное обеспечение для мультиклеточных процессоров представляет собой набор бинарных утилит, разрабатываемых компанией «Мультиклет», а также привлеченных разработок, который включает:

- ассемблер;
- редактор связей (компоновщик);
- препроцессор C;
- компилятор C99;
- драйвер сборки;
- функциональную модель процессора;
- программный загрузчик;
- отладчик.

Ассемблер и редактор связей являются разработками компании «Мультиклет», хотя общие условия их использования схожи с аналогичными бинарными утилитами GNU, информация о которых доступна по адресу http://sourceware.org/binutils. Препроцессор C является полностью сторонней разработкой (см. http://mcpp.sourceforgr.net).

Разрабатываемый компанией «Мультиклет» компилятор C99, в основе которого лежит новая технология LiME, имеет следующие особенности:

- унифицированную обработку исходных текстов на языках программирования с разными грамматиками. Подход LiME, основанный на выводе графа программы по специальным формам и описанию синтаксической структуры текста на универсальном языке, должен существенно упростить разработку интерфейсов других языков программирования;
- возможность расширения языка пользователем путём создания библиотеки собственных языковых конструкций;
- промежуточное представление программ в LiME более абстрактно, чем в традиционных процессорах с фон-неймановской архитектурой. Это позволяет реализовать более эффективные компиляторы для процессоров с принципиально новыми архитектурами.

Представленные выше утилиты разрабатывались как кросс-платформенные и на сегодняшний день могут быть использованы для управления операционными системами Windows и Linux.

В ближайших планах компании – сопряжение инструментария разра-

Сравнение мультиклеточного процессора MULTICLET P1 с аналогами

№ п/п	Параметры / Процессор		Мультиклет	Отечественные аналоги		Зарубежные аналоги				
			Мультиклет МСр0411100101	Элвис 1892ВМ3Т (МС-12)	Элвис 1892ВМ5Я (МС-0226)	Texas Instruments OMAP L-138	Analog Devices ADSP-21469	Texas Instruments TMS320C6701	Analog Devices ADSP-TS201S	Intel Pentium 4
Технологические характеристики										
1	Ядро		Мультиклет	Мультикор	Мультикор	C674+ARM9	SHARC	TMS320C6000	TigerSHARC	Willamette
2	Кол-во клеток / кол-во ядер / кол-во исполнительных устройств		4 / – / –	– / 2 / –	– / 3 / –	– / 1 / 8	– / – / 6	– / – / 8	– / – / 6	– / 1 / –
3	Архитектура		Мультиклеточная	MIPS32 + RISC	MIPS32 + RISC	Гарвардская RISC	Гарвардская	Гарвардская	Гарвардская	IA-32
4	Тип корпуса		QFP-208	PQFP-240	BGA-416	PBGA-361	PBGA-324	BGA-352	Ball BGA-576	FC-PGA2-423
5	Технологический процесс, мкм		0,18	0,25	0,25	0,065	0,065	0,18	0,13	0,18
6	Температурный диапазон, °С		–60…+125[1]	–65…+85	–65…+85	–40…+90	–40…+85	–40…+105	–40…+85	–40…+85
7	Площадь кристалла, мм^2		100	НД[2]	НД	НД	НД	НД	НД	217
8	Разрядность, бит		32/64	32	32	32	32	32	32/64	32
9	Тактовая частота, МГц		100	80	100	456	400	167	600	2000
10	Производительность, Гфлопс		2,4	0,24	1,2	2,7	2,4	1	3,6	2
11	Память, Кбит	ПЗУ	Внешняя[3]	134	128	64	500	64	НД	НД
		ОЗУ	ПД – 128 (4 × 4К × 64), ПП – 128 (4 × 4К × 64)	16	16	8	625	64	3000	НД
12	Напряжение, В	Ядра	1,8	2,5	2,5	1,8	1,8	1,8	1,2	1,8
		Периферии	3,3	3,3	3,3	3,3	3,3	3,3	3,3	НД
13	Максимальная потребляемая мощность, Вт		1,08	1,2	1,6	1	1	1,8	3,4	100
14	Арифметика с плавающей запятой		+	+	+	+	+	+	+	+
Периферийные устройства										
15	SPI		3	–	–	2	2	McBSPs[4]	–	–
16	I^{2}S		1	–	–	1	1	–	–	–
17	I^{2}C		2	–	–	2	1	–	–	–
18	USART		–	–	–	–	–	–	–	–
19	UART		4	1	1	3	1	–	–	–
20	USB		1.1 FS (device)	–	–	2	1.1 FS (host) + 2.0	–	–	–
21	Контроллер Ethernet, Мбит/с		10/100	–	–	10/100	–	–	–	–
22	ШИМ		1	–	–	2	4	–	–	–
23	GPIO		104	НД	НД	НД	НД	НД	НД	НД
24	Системный таймер, разрядность		1/32	–	–	–	1/32	–	–	НД
25	Таймер общего назначения, разрядность		7/32	1/32	1/32	3/64	2/32	2/32	2/64	НД
26	Сторожевой таймер		1	1	1	1	1	1	1	1
Программное обеспечение										
28	Ассемблер		+	+	+	+	+	+	+	+
29	Компилятор С		+	+	+	+	+	+	+	+
30	Отладчик, JTAG		+	+	+	+	+	+	+	+
31	Операционная система		FreeRTOS	Linux, QNX	Linux, QNX	Linux, BIOS, Win CE	НД	НД	НД	Windows, Linux
32	Среда разработки		+	+	+	+	+	+	+	+
Прочие параметры										
33	Область применения		Общепромышленная				Обработка звуковых сигналов	Общепромышленная		
34	Стоимость, руб.		от 565	по запросу	по запросу	600	1200	по запросу	9 000	по запросу

[1] В пластмассовом корпусе.

[2] Нет данных.

[3] Список рекомендуемых флэш-ПЗУ для использования с процессором МСр0411100101: XCF04S, XCF08P, XCF16P, XCF32P.

[4] Многоканальный последовательный порт.

Примечание. Данные взяты из Интернета, авторы не несут ответственности за их достоверность.

Рис. 2. Диаграмма распределения специализированных процессоров

ботки ПО для процессоров MULTICLET с существующей свободно распространяемой средой разработки, адаптация ОСРВ (в частности, eCos; портирование на FreeRTOS завершено), а также окончание работ над альфа-версией компилятора Си-99 на базе каркаса для построения событийно-управляемых трансляторов LiME [2]. В 2014–2015 гг. ОАО «Мультиклет» планирует завершение двух потребительских проектов по выпуску мультиклетов – устройств с широким спектром применения.

Сравнение мультиклеточного процессора MULTICLET P1 с аналогами

Большинство существующих на сегодня процессоров имеют фоннеймановскую архитектуру (CISC, RISC, VLIW и пр.) и при использовании идентичных технологических процессов и методологий проектирования уступают мультиклеточным процессорам как по быстродействию и энергопотреблению (в удельных показателях), так и по надёжности и отказоустойчивости (см. таблицу).

Конкурентные преимущества мультиклетов:

- увеличение производительности в 4–5 раз при одновременном снижении энергопотребления (в 2–4 раза по сравнению с аудиопроцессорами; в 10–15 раз по сравнению с процессорными ядрами со сверхнизким энергопотреблением компаний TI, ARM);
- «естественная» реализация параллелизма (без решения задачи распараллеливания);
- уменьшение площади кристалла;
- эффективная реализация любого класса задач (коммутационная среда не накладывает ограничений на межклеточный обмен данными);
- выполнение программы без перекомпиляции на любом количестве клеток;
- непрерывное выполнение программы при деградации аппаратной среды (отказ клеток);
- естественный иммунитет к вредоносному коду.

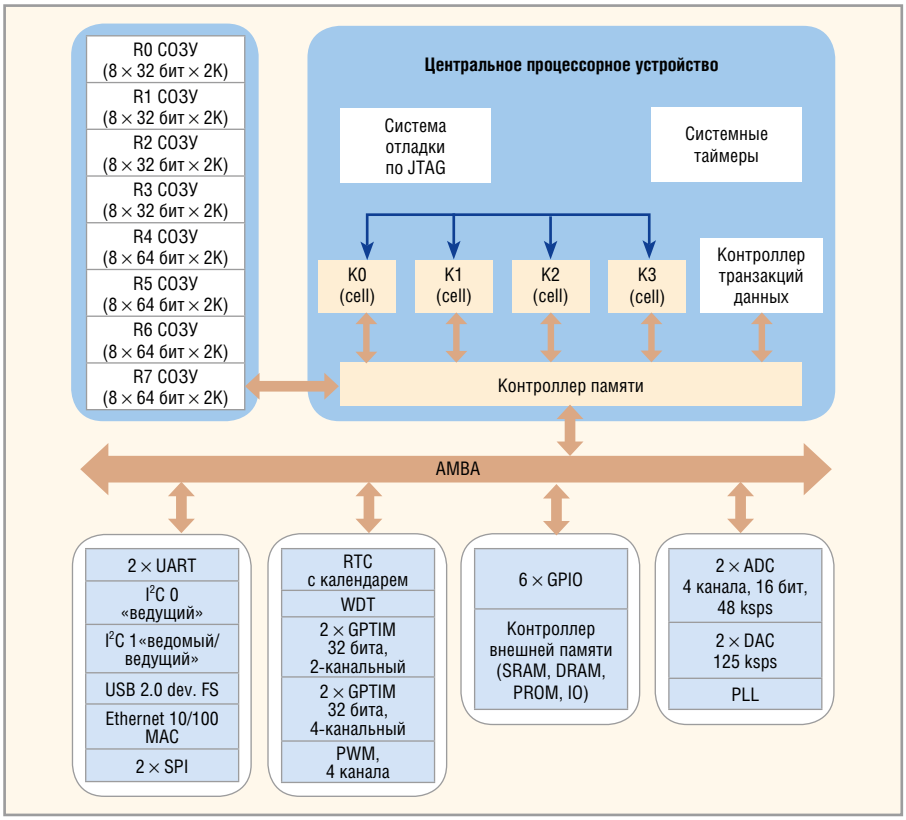

Рис. 3. Структурная схема процессора MULTICLET P2

Сфера применения мультиклетов

Мультиклеточная архитектура является универсальной архитектурой, что обеспечивает широкую сферу применения – от процессоров для персональных компьютеров и суперкомпьютеров до специализированных промышленных систем. Структуру мирового рынка специализированных процессоров иллюстрирует рисунок 2.

Мультиклеты – самостоятельно или в составе сборки – могут успешно использоваться в различных областях техники:

- космическое и авиационное оборудование бортового и наземного базирования;
- промышленная аппаратура;
- специальные приложения на FPGA;
- автомобильная электроника («интеллектуальные» бортовые системы);
- настольные суперкомпьютеры терафлопного класса;
- траст-процессоры «Антихакер» для банковских приложений;
- приёмники ГЛОНАСС/GPS/Galileo;
- звуковые процессоры;
- 3D-телевидение;
- мобильная и видеосвязь.

Перспективы развития

На начало 2014 г. запланирован выход мультиклеточного процессора MULTICLET P2 (серии P – Performance), который ориентирован на максимальную производительность при одновременном снижении энергопотребления. В процессоре MULTICLET P2 увеличены тактовая частота и объём памяти на кристалле, расширен состав периферийных устройств (см. рис. 3).

Также в ближайших планах компании – выпуск процессора с динамической реконфигурацией MULTICLET R1 серии R (Reconfiguration). Процессоры MULTICLET P2 и MULTICLET R1 создаются для общепромышленного применения, а также могут использоваться в автомобильной и специальной электронике. Выпуск «живучего» микропроцессора MULTICLET L1 серии L (Liveness), способного выполнять без перезагрузки и перекомпиляции свои программы даже при выходе из строя одной, двух и более клеток, намечен на конец 2014 г.

Литература

1. *Зырянов Б.А., Стрельцов Н.В.* Обеспечение живучести мультиклеточного процессора. Труды РАСО. ИПУ РАН. 2012. с.126.
2. *Бахтерев М.О., Косенко В.В.* Разработка компилятора для языка программирования RiDE.L. Саров. 2010. с.16–17.