Стековые процессоры

или Новое — это хорошо забытое новое

Данная статья продолжает цикл статей по описанию структур микропроцессоров для встроенного применения [5-8]. Были приведены описания регистрового процессора, процессора, работающего по схеме память — аккумулятор. Настала очередь стекового процессора. Цель данной статьи — показать для чего это надо и «как это сделано внутри». Кроме того, приводится краткое описание языка программирования Форт и Форт-систем.

Иосиф Каршенбойм

iosifk@narod.ru

Автор заранее просит прощения у читателей за очень сжатый, иногда почти «телеграфный» стиль изложения. Материалов по данной теме очень много, некоторые материалы заслуживают краткого описания, а некоторые хочется хотя бы упомянуть, чтобы читатели смогли все остальное узнать сами. Возможно, что в статье что-то не отражено, или у вас есть какие-то материалы по этой тематике — автор всегда готов обсудить заинтересовавшую вас тему. Возможно, результат обсуждения будет отражен в следующей статье.

Автор выражает свою благодарность всем фирмам и частным лицам, предоставившим свои материалы для написания данной статьи.

Где можно увидеть работающие Форт-системы? Начиная от космических Шаттлов на самом верху и до «Спасибо за покупку» на чеке из ближайшего магазина — вот таков диапазон применений Форт-систем.

Блок-схема стекового процессора приведена на рис. 1. В отличие от «обычного» микропроцессора, стековый процессор содержит два стека — стек данных и стек возвратов. Стек возвратов используется для возвратов из подпрограмм, как и у «обычных» микропроцессоров, а вот стек данных — это «привилегия» стекового процессора. Именно через стек данных производится передача параметров при вычислениях.

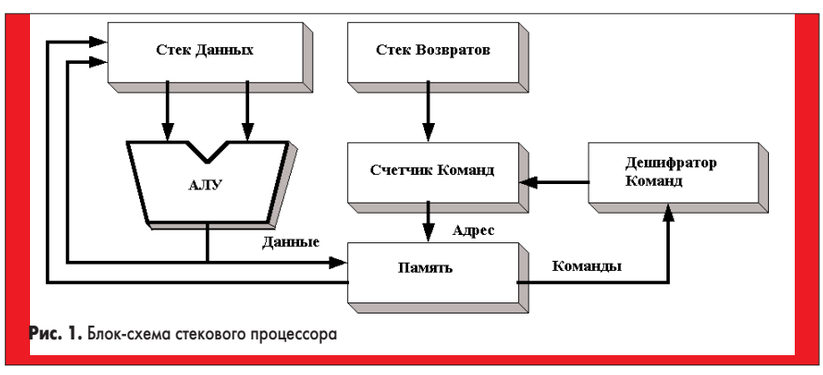

Рис. 1. Блок-схема стекового процессора

Исторический экскурс

Нет еще такой машины времени, чтобы можно было заглянуть вперед и сказать, что и как будет применяться через 100–200 лет. Но на лекциях по философии нас учили, что развитие в природе идет по спирали. И, если хочется узнать, что будет нас ждать впереди, лет через 10–20, то нужно оглянуться назад, посмотреть, что было там и «перенести» это на сегодняшнюю действительность. Итак, попробуем провести очень краткий исторический экскурс в эпохи «компьютерных динозавров», чтобы узнать, что нас ждет в будущем. Поскольку большинство читателей журнала «ходить, говорить и программировать начали одновременно», то за начало «исторической эры», естественно выберем появление первых IBM PC. При этом все получится довольно привычно и «наукообразно», как в археологии. Итак, начнем наши раскопки.

Представим, что компьютерная история состоит из следующих эпох:

А) эпохи первых компьютеров и «доисторических компьютеров»;
Б) эпоха первых однокристальных микропроцессоров;
В) эпоха появления IBM PC;
Г) эпоха появления однокристальных микроконтроллеров;
Д) эпоха расцвета однокристальных микропроцессоров и IBM PC, появление RISC-микропроцессоров и микроконтроллеров;
Е) эпоха появления первых «систем на кристалле».

Как и любая другая историческая формация, каждая компьютерная эпоха проходит стадию зарождения, расцвета, спада и, затем, гибели. Но при этом все свои достижения каждая эпоха передает следующей за ней эпохе, где эти достижения трансформируются под текущие нужды, развиваются, и, затем либо отмирают, либо наследуются следующими поколениями. Но поскольку талант инженеров, программистов, бизнесменов не зависит от той эпохи, в которую они творили, то нам будет полезно внимательно сопоставить на-

правление деятельности людей, определявших уровень эпохи, с теми ресурсами, которые были в тот период у них в распоряжении.

Чтобы не утомлять читателя точными техническими данными, относящимися к тому или иному экспонату, позволим себе приводить только качественные оценки ресурсов процессоров. Поскольку основная сфера интересов автора — микропроцессорные системы и встроенные системы, то за базу для отсчета будем брать те ресурсы, которыми располагают современные микропроцессорные и встроенные системы.

Эпоха первых компьютеров и «доисторических компьютеров»

Погружаемся в доисторические времена. Программистов — всего только сотни. Языков программирования нет, Интернета нет, телеконференций, конечно, тоже нет. Как в таких условиях можно было что-то делать — с сегодняшней точки зрения совершенно не понятно. Ресурсов у машин крайне мало. Однако компьютеры производятся и продаются на коммерческих условиях. В обществе возникает потребность в Программном Обеспечении к компьютерам. Именно эта эпоха характеризуется тем, что у программистов очень много энтузиазма и желания работать. Происходит зарождение языков высокого уровня. Это приводит к облегчению труда программистов и к облегчению отладки программ. Поскольку аппаратных ресурсов мало, то возникает потребность в компиляции программ для систем с ограниченными ресурсами. Здесь необходимо отметить возникновение двух языков программирования, ориентированных на системы реального времени, имеющие ограниченные ресурсы — языки С и Форт. Разрабатываются первые операционные системы.

Эпоха первых однокристальных микропроцессоров

«Доисторические компьютеры» продолжают бурно развиваться, их аппаратные ресурсы значительно выросли. Соответственно, спрос на «доисторические компьютеры» вызвал рост электронной технологии, а это в свою очередь позволило совершить качественный скачок в развитии — создать однокристальные микропроцессоры. Наиболее известный представитель этого класса — Intel 8080. Были, как положено, и другие представители, но у нас в стране о них мало что известно. Итак, что можно сказать о первых однокристальных микропроцессорах? Да почти тоже, что и об их старших братьях. Ресурсов у машин также крайне мало. И они также производятся и продаются на коммерческих условиях. И снова в обществе возникает потребность в Программном Обеспечении именно к этим компьютерам. И снова эта эпоха характеризуется тем, что у пользователей-программистов очень много энтузиазма и желания работать. Разрабатываются первые операционные системы, ориентированные на 8-разрядные микропроцессоры и на системы с минимальными аппаратными ресурсами. Появляются кросскомпиляторы.

Эпоха появления IBM PC

Развитие технологии электронной техники приводит к очередному скачку — появлению 16-разрядной однокристальной машины (кристалл Intel 8086 и IBM PC на его основе). Присмотритесь внимательно к вашей сегодняшней машине, и вы увидите, что могучий P4 унаследовал черты и Intel 8086, и даже Intel 8080. Поскольку далее начинается «наша эра», то остальное сравнение этой эпохи с другими читатель может провести сам. Отметим только, что Программное Обеспечение довольно быстро подгоняется под имеющиеся ресурсы.

Эпоха появления однокристальных микроконтроллеров

Итак мы добрались до очень интересной эпохи — эпохи появления однокристальных микроконтроллеров. В чем же суть этого явления и зачем фирма Intel за них взялась? Как говорится, «а ларчик просто открывался». Все дело в стоимости выполнения одной вычислительной операции. Одна отдельно взятая операция должна выполняться на все более и более дешевых аппаратных средствах, но объем продаж этих средств должен невероятно увеличиваться. Посмотрите предыдущие эпохи — при переходе от одной эпохи к другой происходит удешевление вычислений, при этом объем продаж аппаратных средств резко увеличивается.

Первые микроконтроллеры. Ресурсов настолько мало, что и говорить об этом трудно. Отладка занудная, память команд — по карте заказа или однократно программируемая. У некоторых представителей память команд внешняя.

Здесь мы вплотную подошли к стековым процессорам. NC4000, в дальнейшем переименованный в NC4016 (Novix, 1985), стал первым чипом, разработанным для непосредственного выполнения команд Форта. Он был реализован на матричном кристалле, два массива аппаратных стеков были вынесены наружу и подключались через дополнительные стековые шины. Два верхних элемента стека данных T и N, а также один элемент стека возвратов R были расположены во внутренних регистрах. Процессор выполнял большинство команд, включая вызов подпрограмм, за один такт, что позволяло в одной 16-разрядной команде упаковывать несколько Форт-примитивов.

Эпоха расцвета однокристальных микропроцессоров и IBM PC, появление RISC-микропроцессоров и микроконтроллеров

Этот пункт хочется выделить, чтобы не упустить один довольно важный момент. К этой эпохе развитие электронной техники позволило интегрировать на кристалле значительные ресурсы. И что же происходит? Появляются первые микрокомпьютеры на кристалле. Это аналоги популярных микропроцессоров Intel 80186 и Intel 80386. То есть наступает момент, когда определяющим критерием при разработке микроконтроллера становится уже не технологический предел по числу вентилей в кристалле, а разработанное программное обеспечение.

Увеличение аппаратных ресурсов микроконтроллеров позволяет использовать языки программирования высокого уровня. А это, в свою очередь, приводит к возможности быстрого переноса отлаженных программ с одной аппаратной платформы на другую. Появляется множество структур микропроцессоров. Идет интенсивное развитие программных средств, в том числе и операционных систем реального времени.

Было еще одно знаменательное событие, которое произошло в эту эпоху. Однако это событие было довольно мало известно даже у нас в стране. В эту эпоху в СССР появились первые интегральные стековые процессоры, а если говорить более точно, то Форт-процессоры. Эти процессоры поддерживали набор команд, существующий только в языке Форт. В следующей публикации мы подробно рассмотрим принцип работы Форт-процессора, а в историческом исследовании необходимо отметить, что эта эпоха в нашей стране отмечена появлением первого отечественного микропроцессора, разработанного и внедренного в производство отечественной негосударственной фирмой. Это была фирма «Форт-Инфо», проект Дофин-1610, год выпуска — 1990.

Стековые процессоры сегодня

Типичный представитель — набор микросхем ***TDS9092 FORTH CHIPS***, производимый фирмой Triangle Digital Services (**www.triangledigital.com**, рис. 2).

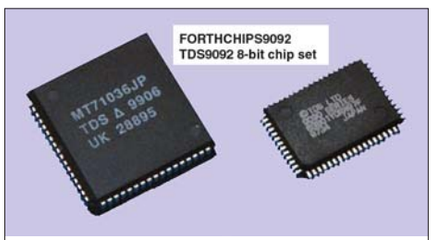

Рис. 2. Набор микросхем TDS9092 FORTH CHIPS

63B01Y0FP — микропроцессор с масочно-зашитым Фортом и символическим ассемблером, с поддержкой многозадачного режима, часами реального времени, экранным редактором и поддержкой прерываний в Форте или ассемблере. Есть драйверы для шины I²C, алфавитно-цифровых ЖКИ, параллельных портов, двух последовательных портов, сторожевого таймера, клавиатуры и режимов малой мощности. Используется версия языка с расширенной 32-разрядной арифметикой, тригонометрией и функциями измерения скорости в реальном масштабе времени.

TDS9 — вентильная матрица, обеспечивающая 16 дополнительных параллельных портов (всего 35). Есть аппаратная поддержка для портов ЖКИ и для сканирования клавиатуры. Сторожевой таймер приводит систему в необходимое состояние в случае аварийного отказа. Сигналы выбора адреса используются для подключения внешней оперативной памяти и ППЗУ. Кроме того, есть резервные декодированные строки адреса для того, чтобы можно было непосредственно подключать внешние устройства типа АЦП.

В состав комплекта входят микросхемы RAM и PROM. PROM содержит коды прикладной программы, написанной на Форте

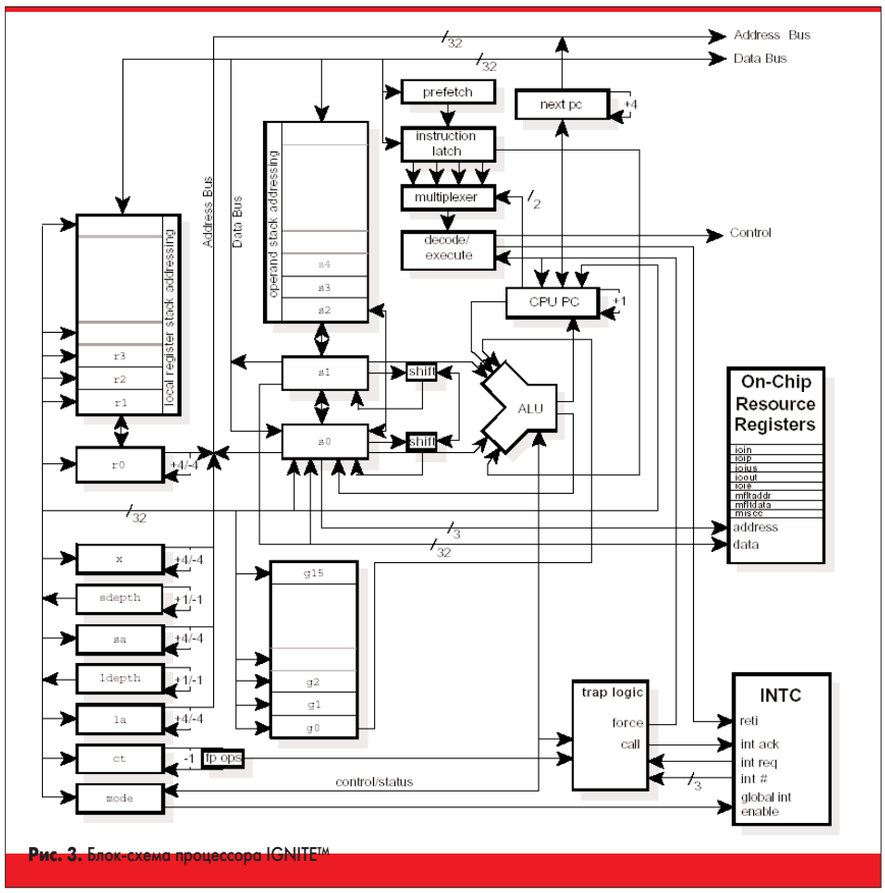

Рис. 3. Блок-схема процессора IGNITE™

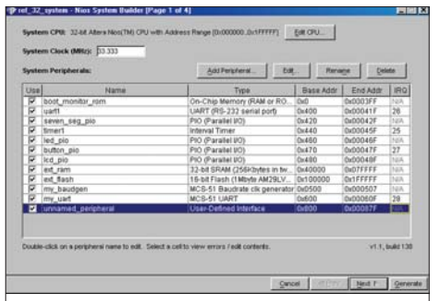

Рис. 4. Окно программы конфигурации периферии встроенного процессора Nios

или ассемблере. (Обратите внимание! В память программ пользователь загружает коды программы на языке высокого уровня!)

Кроме 8-разрядных решений фирма имеет также и 16-разрядные решения для Форт-систем, выполненные на базе стандартных микроконтроллеров.

Микропроцессор IGNITE™, разработанный фирмой Patriot Scientific Corporation (PTSC, www.ptsc.com) — это мощный 32-разрядный микропроцессор с очень высокой производительностью и плотностью упаковки команд, прекрасно исполняющий байт-коды Java, Forth и сгенерированные компиляторами языков C/C++ бинарные программы. Блок-схема приведена на рис. 3. Он имеет «безадресную архитектуру» ROSC (Removed Operand Set Computer — компьютер с безоперандным набором команд). Если его сравнить с RISC-микропроцессорами, то можно заметить, что у RISC-микропроцессоров часть кода команды (до 15 битов или больше в команде) тратится на определение трех возможных операндов для каждой команды. Архитектуре с нулевым операндом (стеком) эти биты операнда не нужны, таким образом, получаются намного более короткие команды, а следовательно, и меньший размер программы. Код операции этого микропроцессора «укладывается» в 8 бит, и за один цикл чтения 32-битного слова выполняется 4 команды. Кроме того, применение стека также позволяет сократить число регистров, выполнять сохранение в стеке и извлечение данных из стека в пределах одной процедуры, таким образом позволяя более иметь короткие последовательности команды и выполнять код быстрее.

В таблице 1 приведен пример выполнения программы для вычисления функции g5 = g1–(g2+1)+g3–(g4×2) на RISC-процессоре и на процессоре IGNITE.

Таблица 1. Сравнение выполнения программы на RISC-процессоре и на процессоре IGNITE

Номер команды	RISC MPU	IGNITE
1	add #1, g2, g5	push g1
		push g2
		inc #1
2	sub g1, g5, g5	sub
3	add g5, g3, g5	push g3
		add
4	shl g4, #1, temp	push g4
		shl #1
5	sub g5, temp, g5	sub
		pop g5
	Всего 20 байт	Всего 10 байт

Эпоха появления первых систем на кристалле, IP-ядра

Наконец, уважаемый читатель, мы выбрались из дебрей истории и дошли до сегодняшнего дня. Оглядываемся — и видим настольные компьютеры с невероятными ресурсами. Микроконтроллеры — любых сортов и мастей. И их ресурсы таковы, что иная «доисторическая машина» и мечтатьоб этом не могла. Однако, как зеленый росток из-под асфальта, упрямо лезет что-то новое, называемое «системой на кристалле» [4]. Причем, суть этого явления все та же — сделать каждую вычислительную операцию дешевле, но объем продаж этих вычислительных средств невероятно увеличить. И вот опять ситуация напоминает то, что уже было, но, как мы помним, на новом, более высоком уровне. Рост технологии — появление FPGA, увеличение ресурсов FPGA, удешевление встроенных в FPGA микропроцессоров. И снова мы проходим по тому же кругу спирали. Ресурсов у встроенных машин также крайне мало. И снова в обществе возникает потребность в Программном Обеспечении именно к этим встроенным микроконтроллерам. И снова эта эпоха характеризуется тем, что у программистов очень много энтузиазма и желания работать. Только теперь в «игре» не одни любители-энтузиасты. В дело вступают мощные фирмы-производители микросхем FPGA. Разрабатываются первые инструментальные системы конфигурирования, отладки и тестирования, ориентированные на встроенные микропроцессоры и микроконтроллеры, работающие в системах с минимальными аппаратными ресурсами [7]. На рис. 4 показано одно из окон программы конфигурации периферии встроенного процессора Nios фирмы Altera.

Как и следовало ожидать, кроме различных ядер с RISC-процессорами и IP-ядер, повторяющих стандартные микроконтроллеры, появляется множество различных разработок IP-ядер Форт-процессоров. Разработки выполняются в университетах студентами, любителями-одиночками, фирмами. В таблице 2 приведены описания двух типичных Форт-процессоров, выполненных в FPGA. Оба имеют типичные для Форт-процессоров архитектуры.

Таблица 2. Ядра Форт-процессоров, выполненные в FPGA

Название	Разрядность	Число логических ячеек в FPGA	Синхрочастота
MSL16	16	Xilinx 4006E-1, 175 CLBs	33~ МГц
b16	16	Flex10K30E About 600 LCs,	25 МГц

Необходимо добавить, что для процессора b16 его автор Bernd Paysan [16], отмечает, что в случае выполнения ядра процессора по технологии TSMC 0,5 < 0,4 мм^2 с 3 слоями металлизации при питании 5 В (то есть на сегодняшний день это далеко не передовая технология) и со стеком, имеющим 8 элементов, может быть получена тактовая частота 100 МГц.

Однако встречаются и проекты, где преследуются цели получения максимальной производительности. Тогда на свет появляются продукты, подобные 4stack.

Ядро процессора 4stack (http://www.jwdt.com/~paysan/4stack.html) использует набор команд, ориентированных на работу со стеками для четырех узлов обработки, вы-

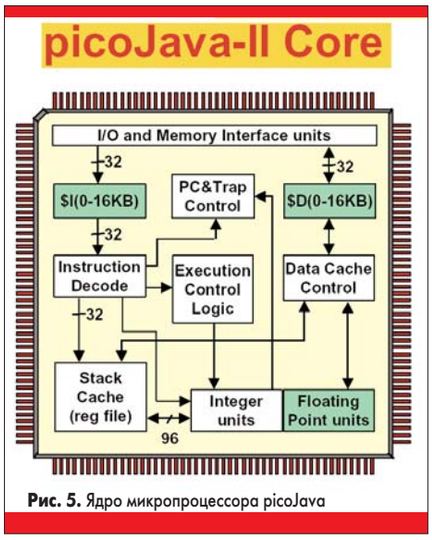

Рис. 5. Ядро микропроцессора picoJava

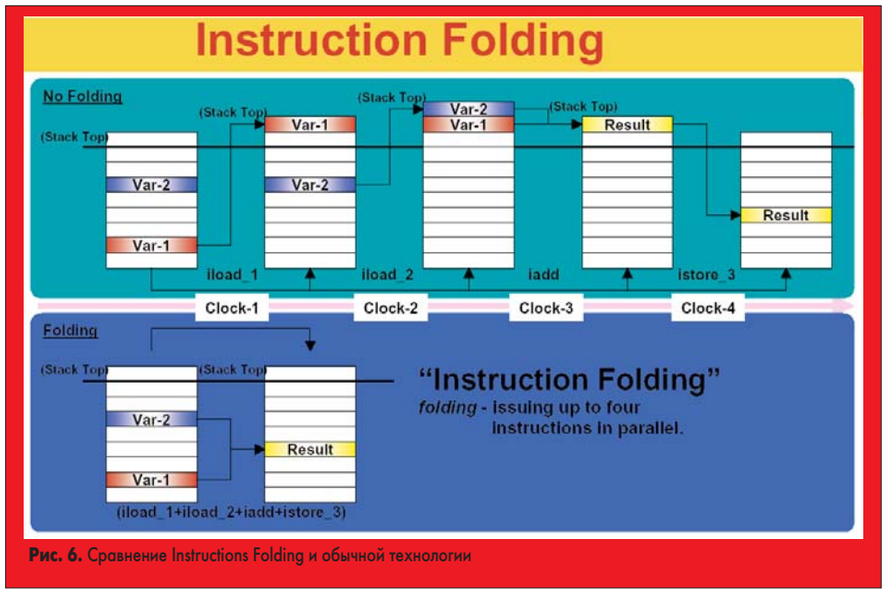

Рис. 6. Сравнение Instructions Folding и обычной технологии

полненных как VLIW-сопроцессоры. Если бы процессор 4stack был выполнен по самой современной технологии, то он значительно превзошел бы по быстродействию такие высокопроизводительные DSP-устройства как TMS 320C6x или TigerSHARC.

Процессор имеет четыре арифметико-логических устройства и выполняет четыре операции со стеками, две операции load/store и две операции модификации адреса. Два блока DSP MAC и два сопроцессора для операций с плавающей точкой (один сумматор и один умножитель), позволяют производить высокоэффективную обработку сигналов и трехмерные геометрические вычисления. Именно поэтому процессор 4stack имеет достаточную производительность, которая может быть использована в блоках цифрового телевидения.

Упрощение архитектуры VLIW приводит, с одной стороны, к сокращению количества транзисторов, а с другой — к сокращению связей внутри кристалла и уменьшению энергопотребления. Для сравнения можно сопоставить Athlon с его приблизительно с 20 миллионами транзисторов. Он потребляет 40 Вт при 1 ГГц, в то время как ядро процессора 4stack, выполненное по той же самой технологии, по предварительной оценке, будет потреблять 1 Вт при той же частоте в 1 ГГц.

Применение стека, как было сказано выше, очень увеличивает плотность команды. В то время как нормальный RISC-процессор использует для одной команды 32 бита, процессор 4stack одновременно выполняет 8 операций в 64 битах. Архитектура VLIW может не всегда заполнять каждый слот операции, однако в том случае, когда заполнено не менее двух слотов операции, процессор 4stack уже имеет выигрыш по производительности. Это позволяет лучше использовать память программы, сокращая цену чипа (меньший кэш команд) и системные затраты (меньше оперативной памяти). Выполнение переходов без дополнительных циклов ожидания позволяет производить быстрое выполнение обычной и объектно-ориентированной рабочей программы (быстро обработать перетранслированный Java-код). Процессор 4stack имеет защиту памяти, виртуальную память, различие режимы supervisor/user, и другие вещи, которые обычно имеют центральные процессоры ПК. В то же время большинство центральных процессоров, применяемых в ПК, не имеют аппаратной поддержки DSP.

Появление Java-процессоров — новый виток гонки

Распространение Интернет-технологий вызвало потребность создания устройств широкого применения, таких как справочные терминалы, домашние устройства для доступа в Интернет и т. д. Для этих целей были созданы процессоры, непосредственно исполняющие байт-код Java. Java-процессоры также используют архитектуру стека. Далее приведены описания только двух представителей из семейства Java-процессоров.

Ядро микропроцессора picoJava фирмы Sun microsystems (рис. 5, [15–18]) было разработано в конце 1990-х годов, оно предназначено для выполнения байт-кода Java так, как определено в Java Virtual Machine (JVM). Оно может также выполнять код C/C ++ с эффективностью сопоставимых архитектур RISC-процессоров.

При разработке ядра процессора picoJava были применены очень интересные технические решения, направленные на повышение быстродействия. Полное описание работы ядра — это тема отдельной статьи, поэтому здесь хочется только кратко упомянуть одно решение — технологию Instructions Folding. Традиционная работа стековой машины представлена на рис. 6, на верхнем, зеленом поле. Для того чтобы обработать две переменные Var-1 и Var-2, выполняются следующие действия: сначала одна, затем вторая переменная заносится на вершину стека, после чего над ними выполняется требуемая операция, результат которой заносится в определенное место памяти. На обработку тратится 4 такта.

На нижнем, синем поле, представлена диаграмма работы ядра процессора picoJava. После того, как ядро получает очередную команду, производится анализ этой команды на «совместимость». До 4 команд может быть объединено в один folder, после чего они «совмещаются» в одну команду и выполняются. На диаграмме показано, что обработка двух переменных Var-1 и Var-2 объединена вместе с записью результата в одну команду, которая выполняется за один такт.

Примером применения технологии picoJava-II может служить микропроцессор Fujitsu MB86799. Он состоит из ядра picoJava-II, внешней интерфейсной шины и интерфейса шины PCI. Он имеет кэш команд 8 КБ, кэш данных 8 КБ, кэш стека с 64 входами и сопроцессор для операций с плавающей точкой (рис. 7). MB86799 может работать с внешней максимальной частотой 33 МГц и внутренней частотой 66 МГц. Соотношение внутренней и внешней частоты может быть от 2 до 5. Чип потребляет 360 мВт от источника питания 2,5 В при тактовой частоте 66 МГц.

Дальнейшим развитием этого направления стал микропроцессор MB92901 (рис. 8). Он выполнен по технологии «0.25µm/4 layered Al», имеет питание 2,5/3,3 В, кэш команд 8 КБ, кэш данных 8 КБ и шину данных для подключения памяти с разрядностью в 32 бита. Синхрочастота — 66 МГц. Периферийные блоки: контроллер прерываний (до 16 каналов), сторожевой таймер, таймеры (3 канала), системный таймер, порт для последовательной связи, UART, 8-разрядный порт I/O. Корпус — LQFP-208P.

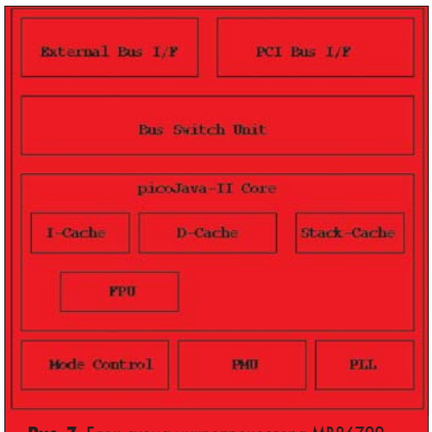

Рис. 7. Блок-схема микропроцессора MB86799

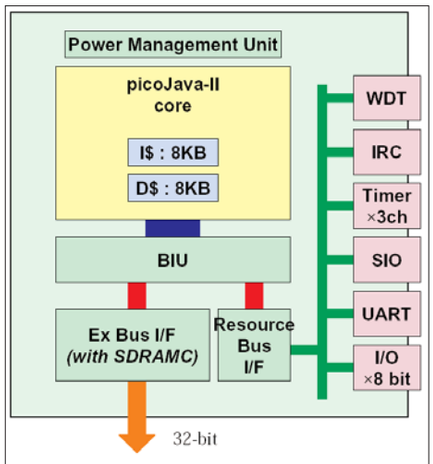

Рис. 8. Блок-схема микропроцессора MB92901

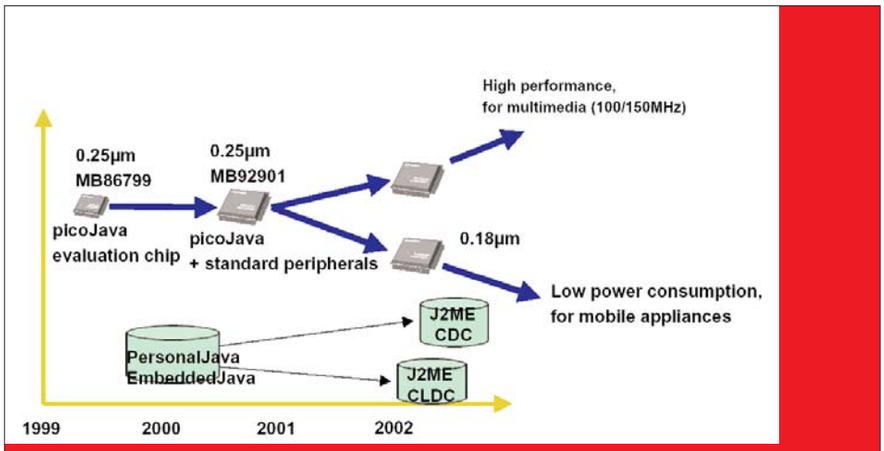

Рис. 9. Тенденция развития Java-микропроцессоров на базе ядра picoJava

Тенденция развития Java-микропроцессоров на базе ядра picoJava представлена на рис. 9. Микропроцессоры будут развиваться в двух направлениях: первое — для высокопроизводительных устройств, второе — для переносных устройств с малым энергопотреблением.

LavaCORE — 32-разрядное ядро Java-процессора, предназначенное для реконфигурируемых аппаратных средств и оптимизированное для применения в Xilinx Virtex-II. Оно разработано фирмой Derivation Systems, Inc. (DSI) (www.derivation.com), являющейся партнером фирмы Xilinx, и может быть загружено с сайта Xilinx IP Center (www.xilinx.com/ipcenter). Ресурсы, занимаемые ядром процессора на кристалле, и характеристики микропроцессора приведены в таблице 3.

Ядро процессора непосредственно выполняет байт-код Java аппаратным способом. LavaCORE выполнено как soft-ядро процессора с набором программных инструментальных средств для выполнения Java-приложения и конфигурирования процессора.

На рис. 10 приведена блок-схема ядра LavaCORE с дополнительными модулями. Ядро процессора состоит из блока целочисленных вычислений, программируемых таймеров, регистрового файла, и контроллера прерывания. Дополнительные модули включают местную память, внутреннюю память, расположенную в FPGA, аппаратный блок кодирования, блок вычислений с плавающей точкой и сборщик мусора. Варианты конфигурации позволяют произвести выбор того, какие команды байт-кода Java могут быть опущены или перенесены из аппаратных средств в программное обеспечение, а также выбрать размеры кэша и разрядность шины данных (32, 16 или 8 бит).

Таблица 3. Ресурсы, занимаемые ядром процессора LavaCORE

Примечание:

Supported Family	Device Tested	CLB Slices	Clock IOBs*	IOBs*	Performance (MHz)	Xilinx Tools
Virtex II	2V1000-5	1926	1	112	20	ISE 3.3
Virtex-E	V300-8	1903	1	112	16	ISE 3.3

* Все входы-выходы ядра процессора подключены ко входам-выходам кристалла.

На рис. 11 изображена диаграмма интерфейса LavaCORE. Сигналы интерфейса (табл. 4) включают в себя сигналы синхрочастоты, сброс, захват, подтверждение захвата, интерфейс памяти и флажки состояния. Ядро содержит в себе регистровый файл 32×32 бита, три 8-битовых регистра команд, буфер выборки команд (что позволяет уменьшить время доступа к памяти), поддержку кэширования и 32-разрядное арифметико-логическое устройство, выполняющее целочисленные и логические операции.

Появляются первые гибридные «системы на кристалле»

Так можно было бы назвать кристалл, состоящий из двух областей — аппаратного микроконтроллера и FPGA, который может конфигурироваться пользователем. Поскольку в качестве аппаратного микроконтроллера в таких системах стековые процессоры пока не используются, то в данном разделе ограничимся упоминанием сайтов фирм Altera, Triscend, Atmel и др., где имеются достаточно подробные описания.

Таблица 4. Сигналы ядра процессора LavaCORE

Signal	Direction	Description
CK	Input	System Clock
RST_LO	Input	System Reset (active low)
HOLD	Input	Hold processor state; idle after current instruction
HOLD_ACK	Output	Hold request acknowledge
IOPP	Output	Supervisory memory read/ write for peek/poke
INTR	Input	Interrupt request
FS[7:0]	Output	Processor status byte
MEMWR	Output	Memory Write Strobe
MEMRD	Output	Memory Read Strobe
MEMRDY	Input	Memory Ready Strobe
ADDR[31:0]	Output	Memory Address
DATAIN[31:0]	Input	Memory Data Input
DATAOUT[31:0]	Output	Memory Data output

Заканчивая обзор

Как мы видим из проведенного небольшого исторического исследования, постепенное развитие технологии приведет к удешевлению «систем на кристалле» и гибридных «систем на кристалле». На первых порах ресурсов для «систем на кристалле» будет мало (это мы видим уже сейчас). Поскольку аппаратные ресурсы встроенных микроконтроллеров ограничены, то ситуация будет напоминать эпоху первых микропроцессоров.

И еще несколько слов о том, «кто заказывает музыку». Как упоминалось выше, на-

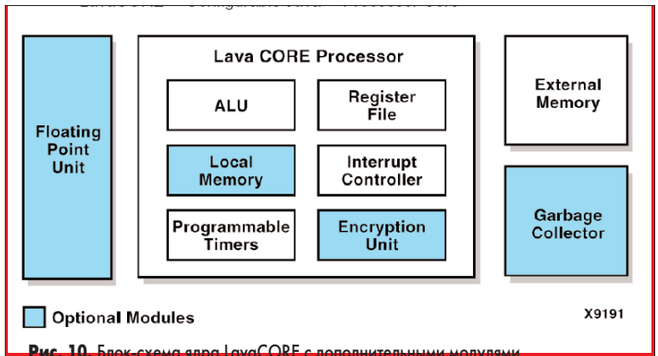

Рис. 10. Блок-схема ядра LavaCORE с дополнительными модулями

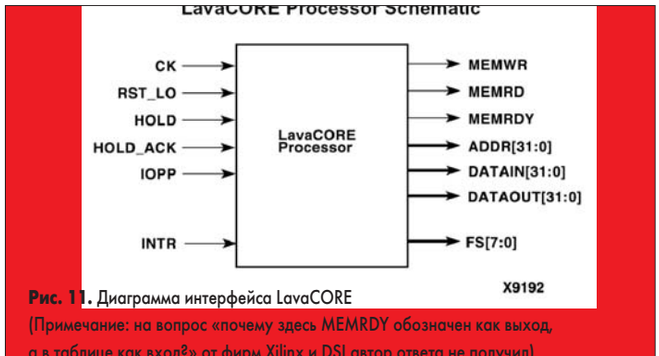

Рис. 11. Диаграмма интерфейса LavaCORE

(Примечание: на вопрос «почему здесь MEMRDY обозначен как выход, а в таблице как вход?» от фирм Xilinx и DSI автор ответа не получил)

ступает некоторый уровень развития технологии электронной техники, когда уже не требования технологии определяют, сколько и каких ресурсов можно разместить на кристалле. Теперь архитектуру кристалла будет определять необходимость поддержки определенного языка программирования с определенным набором команд. И чем проще архитектура микропроцессора, требуемая для поддержания языка высокого уровня, тем раньше это произойдет. Можно ожидать, что удешевление ресурсов для «систем на кристалле» приведет к еще большему увеличению рынка сбыта этих устройств.

Таким образом, применение стековых машин становится наиболее целесообразно для небольших встроенных применений, так как позволяет получить поддержку языка высокого уровня и упростить отладку системы. Однако применение Форт-процессоров будет, как и на предыдущих этапах, тормозиться отсутствием квалифицированных программистов, с одной стороны, а с другой стороны — отсутствием свободно распространяемого программного обеспечения для встроенных систем.

Литература

1. Баранов С.Н., Ноздрунов Н.Р. Язык Форт и его реализация. Л.: Машиностроение. 1988.
2. Бураго А.Ю., Кириллин В.А., Романовский И.В.. Форт — язык для микропроцессоров. Ленинградская организация общества «Знание» РСФСР. 1989.
3. Броуди Л. Начальный курс программирования на языке ФОРТ. М.: «Финансы и статистика». 1990.
4. Кривченко И. Системы на кристалле: общее представление и тенденции развития // Компоненты и технологии. 2001. № 6.
5. Каршенбойм И. Микропроцессор своими руками // Компоненты и Технологии. 2002. № 6, 7.
6. Каршенбойм И. Микропроцессор своими руками — 2. Битовый процессор // Компоненты и Технологии. 2003. № 7, 8.
7. Каршенбойм И. Микропроцессор для встроенного применения Nios. Система команд и команды, определяемые пользователем // Компоненты и Технологии. 2002. № 8, 9.
8. Каршенбойм И. Микропроцессор для встроенного применения Nios. Конфигурация шин и периферии // Компоненты и Технологии. 2002. № 2, 3, 4, 5.
9. Зотов В. PicoBlaze — семейство восьмиразрядных микропроцессорных ядер, реализуемых на основе ПЛИС фирмы Xilinx // Компоненты и Технологии. 2003. № 4.
10. Купман Ф. Stack Computers. **http://www.ece.cmu.edu/~koopman/stack.html**.
11. Leong P.H.W., Tsang P.K., Lee T.K.. MSL16. Китайский Университет Гонконга.
12. b16 — A Forth Processor in an FPGA. Bernd Paysan. **http://www.jwdt.com/paysan/b16.html**.
13. Зубинский А. Прагматичные процессоры // Компьютерное обозрение. 2001. № 17.
14. PicoJava Overview. **http://www.sun.com/microelectronics/picoJava/overview.html**.
15. В. Коржов. PicoJava I — первый процессор Java. Сети. 1997. № 1.
16. Харлан Макгэн, Майк О'Коннор. PicoJava: Механизм непосредственного выполнения байт-кода Java // Открытые системы. 1999. № 1. **http://osp.admin.tomsk.ru/os/1999/01/17.htm**.
17. Роберт Макмиллан. Sun готовит новые микросхемы Java. SunWorld. **http://www.airport.sakhalin.ru/ospru/java/1997/05/05.htm**.