УДК 004.2

АРХИТЕКТУРА ПРОЦЕССОРОВ В СОСТАВЕ ГЕТЕРОГЕННОЙ СИСТЕМЫ НА КРИСТАЛЛЕ ДЛЯ ВСТРАИВАЕМЫХ ПРИМЕНЕНИЙ

Тарасов И.Е.

МИРЭА - Российский технологический университет, 119454, Россия, г. Москва, проспект Вернадского, 78, e-mail: tarasov_i@mirea.ru

Рассмотрен подход к проектированию сверхбольших интегральных схем класса «система на кристалле», предназначенных для использования во встраиваемых приложениях начального и среднего уровня. В микроэлектронике наблюдается, с одной стороны, увеличение плотности компонентов на полупроводниковом кристалле, а с другой – усложнение изделий и функциональных требований к ним. В этих условиях становится возможным размещение на кристалле нескольких процессорных ядер, которые следует адаптировать к соответствующим подклассам задач, типичных для встраиваемых систем. На практике это означает применение системы на кристалле с гетерогенной архитектурой, интегрирующей процессорные ядра различного назначения. В статье рассмотрены возможные микроархитектуры процессорных ядер, применимых в системах на кристалле встраиваемого класса, а также подход динамического перераспределения накристальной памяти и периферийных устройств между вспомогательными процессорными ядрами, работающими под управлением центрального процессора.

Ключевые слова: процессор, микроконтроллер, система на кристалле, совместная оптимизация, аппаратная архитектура.

ARCHITECTURE OF PROCESSORS IN A HETEROGENEOUS SYSTEM-ON-CHIP FOR EMBEDDED APPLICATIONS

TARASOV I. E.

MIREA - Russian Technological University, 119454, Moscow, 78 Vernadskogo Avenue, Russia, e-mail: tarasov_i@mirea.ru

The article discusses an approach to designing system-on-a-chip ultra-large-scale integrated circuits intended for use in entry-level and mid-level embedded applications. In microelectronics, on the one hand, an increase in the density of components on a semiconductor crystal is observed, and on the other hand, a complication of products and functional requirements for them. Under these conditions, it becomes possible to place several processor cores on a chip, which should be adapted to the corresponding subclasses of tasks typical of embedded systems. In practice, this means using a system on a chip with a heterogeneous architecture that integrates processor cores for various purposes. The article considers possible microarchitectures of processor cores applicable in systems on a chip of an embedded class, as well as the approach of dynamic redistribution of on-chip memory and peripheral devices between auxiliary processor cores operating under the control of a central processor.

Keywords: processor, microcontroller, system on chip, co-optimization, hardware architecture.

Введение

Повсеместное распространение информационных технологий обуславливает

повышенную потребность в цифровых управляющих системах, в качестве которых на протяжении последних десятилетий традиционно выступают микропроцессорные устройства. Можно обратить внимание, что на протяжении развития вычислительной техники понятие «процессор» стало относиться к широкому классу устройств, существенно отличающихся производительностью, энергопотреблением, функциональными возможностями и т.д. Это стало следствием внедрения процессоров в устройства начального уровня, такие как распределенные системы сбора данных, устройства Интернета вещей (IoT), в том числе носимые, которые ставят приоритеты в виде снижения потребляемой мощности, тогда как производительность и объем памяти находятся на относительно низком уровне. При этом такой низкий уровень в действительности означает тактовую частоту 20-80 МГц и объемы памяти уровня 16-256 кб флеш-памяти и 4-16 кбайт статической памяти. Такие платформы, как Arduino, в действительности являются достаточными для многих практических приложений, где от процессора не требуется высокая по современным меркам производительность, а речь идет преимущественно о поддержке датчиков и исполнительных устройств. Платформа Arduino [1], представленная процессорами на основе ядер AVR и ARM, часто рассматривается как стандарт де-факто для обучения информационным технологиям и разработки прототипов устройств робототехники, элементов умного дома, а также для управления промышленными устройствами. Ключевым фактором при этом является умеренная скорость работы самих устройств, что обусловлено механической инерцией движущихся частей или эргономическими факторами (например, при чтении показаний датчиков частота смены значений ограничена скоростью человеческого восприятия).

Важным вопросом для Arduino является применение достаточно высокой степени абстрагирования от аппаратуры. Это достигается за счет снижения производительности вычислений, поскольку используется динамическое определение параметров аппаратной части вместо статически скомпилированных библиотек, ориентированных на конкретную модель микроконтроллера. Такой подход показывает, что для приложений с невысокими требованиями к производительности приоритетом является удобство разработки, с преимущественной ориентацией на «программирующих профессионалов».

Наиболее распространенным подходом является применение встраиваемой системы на базе одноядерного процессора. Традиционно применяющиеся системы основаны на архитектурах 8051, AVR, PIC, ARM32, STM8, MIPS, RISC-V и, частично, на x86 (в вариантах встраиваемых решений). Эти процессоры в подавляющем большинстве относятся к одноядерным, а поддержка нескольких задач осуществляется путем их временного мультиплексирования. Несмотря на умеренные требования к производительности, это создает ряд неудобств в практической деятельности. В ряде источников [2] рассматриваются вопросы подключения периферийных контроллеров к процессорному ядру и их взаимодействие в процессе работы.

1. Поддержка протоколов обмена, критичных к временным интервалам, может нарушаться из-за обработки прерываний и промахов при доступе к кэш-памяти.

Большое количество датчиков и исполнительных устройств может ставить одноядерный процессор в достаточно сложную ситуацию с точки зрения алгоритмического разрешения конфликта одновременного доступа к устройствам. Медленные протоколы при их программной реализации могут существенно перегружать

процессор непродуктивной работой, заставляя исполнять циклы задержки для воспроизведения временных диаграмм обмена данными. В этой связи большинство современных микроконтроллеров имеют в своем составе большое количество аппаратных контроллеров периферийных устройств, аппаратно поддерживающих протоколы обмена данными.

Тесно связанной проблемой является т.н. «шторм прерываний», под которым понимается сценарий одновременного формирования большого количества запросов на прерывание от аппаратных контроллеров. Это может объясняться особенностями реализуемых алгоритмов или являться следствием аварийной ситуации со срабатыванием большого количества контрольных датчиков и сторожевых устройств.

2. Реализация алгоритмов цифровой обработки сигналов и задач управления в реальном времени существенно зависит от постоянства временного интервала между итерациями обработки. Например, алгоритмы численного дифференцирования, широко применяемые в системах управления, непосредственно основаны на предположении об известном и постоянном интервале времени дискретизации анализируемого сигнала. Спектральные и статистические характеристики сигналов также будут определены неверно при нестабильном во времени процессе их приема.

Вопросы выбора архитектуры процессорного ядра активно исследуются в литературе. Это относится как к широко известным фундаментальным трудам [3], так и к частным работам по оптимизации отдельных показателей [4] или же использованию специфичных программных моделей [5]. Направление встраиваемых приложений получает отдельное внимание [6]. Возможным решением проблемы является применение многоядерной процессорной системы, где отдельные критичные процессы будут поддерживаться выделенными процессорными ядрами.

Архитектура для встраиваемых приложений на базе гетерогенной многопроцессорной системы

Применение многоядерных процессоров требует соответствующих архитектурных решений и методической поддержки. В отличие от понятия «мультипроцессорность» (multicore), под которым понимается размещение в системе нескольких одинаковых ядер, термин «многопроцессорность» (manycore) подразумевает использование неидентичных процессорных ядер. Это в целом соответствует постановке задачи поддержки встраиваемых решений, где можно выделить следующие процессы:

1. Потоковая цифровая обработка сигналов и алгоритмы управления в реальном времени. Эти задачи характеризуются жесткими требованиями к постоянству времени одной итерации и относительно несложными вычислениями в сочетании с небольшим объемом программ.

2. Обмен данными с датчиками и внешними интерфейсными устройствами, для которого типичны медленные или нерегулярные процессы обмена данными с относительно жесткими требованиями к соблюдению временных интервалов обмена. В зависимости от используемых протоколов обмена данными объем программ может изменяться от небольшого до среднего.

3. Обмен данными с внешними вычислительными устройствами, включая реализацию интерфейса пользователя. Эти алгоритмы могут требовать среднего или большого объема памяти, например, для поддержки графического интерфейса на внешнем дисплее.

Таким образом, при построении архитектуры многопроцессорной системы необходимо предусмотреть специализированные процессорные ядра для независимой поддержки нескольких одновременно протекающих процессов управления. Кроме этого, для поддержки алгоритмов управления системой, сложность которых заранее неизвестна, требуется центральное управляющее ядро общего назначения. Пример такой системы показан на рис. 1.

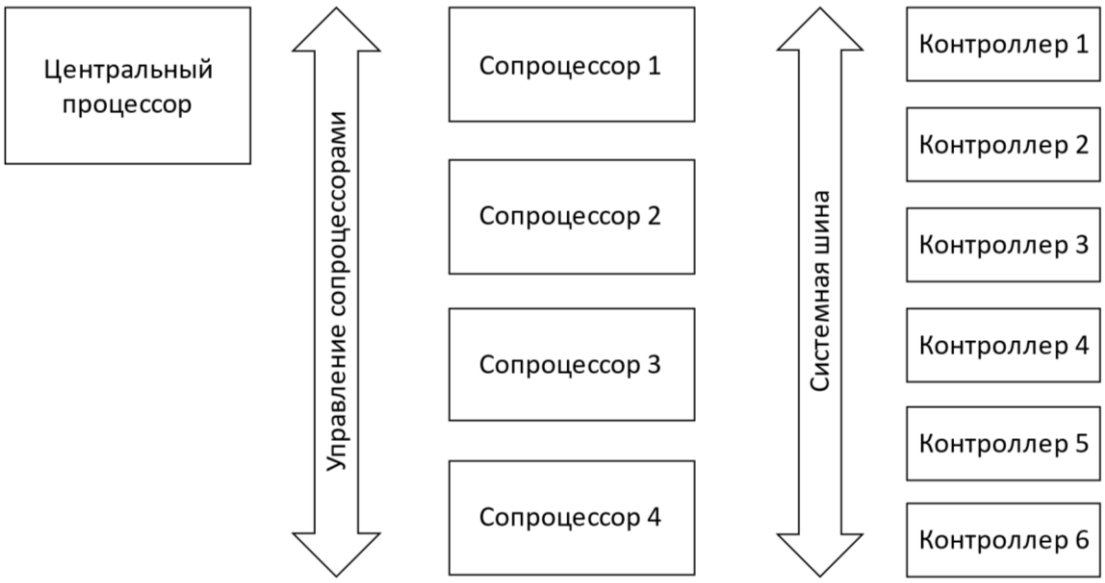

Рис.1. Пример гетерогенной многопроцессорной системы

Динамическое перераспределение ресурсов

Многопроцессорная система при практическом использовании формирует важную задачу распределения ресурсов системы между задачами. Очевидной проблемой является возможный недостаток памяти для одного из вспомогательных ядер при ее избытке для других ядер (при этом общий объем памяти в системе может быть достаточен). Кроме этого, при наличии нескольких подключенных устройств необходимое распределение вычислительной мощности неочевидно на этапе проектирования и не может быть зафиксирована на этапе проектирования.

Альтернативным подходом является применение динамического мультиплексирования доступа к ресурсам со стороны процессорных ядер. В предельном варианте это достигается за счет использования т.н. полного коммутатора, позволяющего подключить шину управления каждого периферийного контроллера к любому из присутствующих на кристалле процессоров. В этом случае можно пойти на определенный компромисс, увеличивая площадь такого коммутатора и снижая общую тактовую частоту системы. Сопутствующим приемом, предлагаемым в данной работе, является аналогичная динамическая коммутация доступа к блокам памяти, которые могут выступать в качестве как памяти программ, так и памяти данных. Такой подход позволяет выделять память отдельным ядрам по мере необходимости, реализуя программные комплексы с неравномерной загрузкой отдельных ядер.

Пример подключения периферийных контроллеров и блоков памяти к сопроцессорам с применением коммутаторов показан на рис. 2.

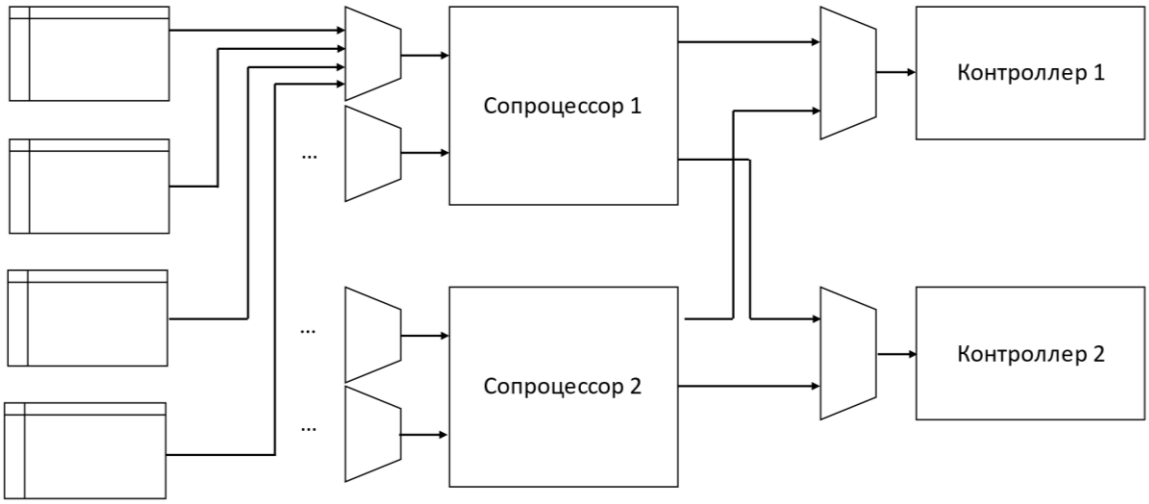

Рис.2. Пример коммутатора для динамического перераспределения ресурсов между процессорными ядрами

Показанный пример коммутатора обеспечивает большую гибкость в динамическом распределении ресурсов, однако имеет ряд недостатков. Первым из них является плохая масштабируемость, поскольку при увеличении числа процессорных ядер и периферийных блоков возрастает также и сложность коммутаторов, что ведет к увеличению их размера на кристалле и возрастанию задержки распространения сигналов.

Вторым недостатком является потенциальная уязвимость к алгоритмическим ошибкам, которая может выразиться в нарушении работы периферийных устройств при принудительном переключении доступа между процессорными ядрами. Этот недостаток, однако, устраняется на стадии разработки программного обеспечения. В целом, особенностью встраиваемых систем является отсутствие возможности загрузки программ конечным потребителем, поэтому потенциальные алгоритмические проблемы могут быть достаточно эффективно устранены разработчиками.

Подобные подходы применительно к ПЛИС были ранее опробованы в ходе предшествующих проектов. Например, в [7] описано серийное изделие на основе многопоточного процессора в ПЛИС Xilinx Spartan-6, где управление набором независимых периферийных контроллеров осуществлялось на основе временного мультиплексирования выполняемых на одном ядре потоков.

При наличии в составе системы процессоров с различной архитектурой встает задача инструментального обеспечения разработки. Можно отметить, что современное состояние в области компиляторов [8, 9] допускает быструю разработку по крайней мере ассемблера для оригинальной процессорной архитектуры.

Для встраиваемых применений в настоящее время наблюдается использование относительно простых языков программирования на основе Си или Питона, поэтому этот вопрос не представляется критически важным и позволяет достаточно свободно экспериментировать с программной моделью разрабатываемых процессорных ядер, оптимизируя их для работы с определенными подклассами алгоритмов.

Прототипирование описанных систем осуществлялось на базе ПЛИС с использованием САПР Xilinx Vivado [10]. Возможностей САПР ПЛИС достаточно для проведения поведенческого моделирования на системном уровне, которое включает в

себя инициализацию памяти примерами программ с последующим наблюдением поведения системы в процессе их выполнения. Этот подход более информативен по сравнению с тестированием отдельных узлов процессора и позволяет выявить потенциальные ошибки на уровне интеграции подсистем, а также отсутствие необходимых функциональных возможностей для реализации типичных задач, для которых предназначается система в целом.

Заключение

Рассмотренная архитектура гетерогенной системы на кристалле позволяет проектировать цифровые интегральные схемы с процессорными ядрами для решения задач управления во встраиваемых системах с учетом современных требований по поддержке нескольких одновременно протекающих процессов обмена данными и управления.

Список литературы

1. What is Arduino? | Arduino [Электронный ресурс]. - URL: https://www.arduino.cc/en/Guide/Introduction (дата обращения: 27.03.2022)
2. Currie, Edward. (2021). Microcontroller Subsystems. April 2021. DOI: 10.1007/978-3-030-70312-7_2. In book: Mixed-Signal Embedded Systems Design.
3. John L. Hennessy, David A. Patterson. Computer Architecture. 6th Edition. A Quantitative Approach. (The Morgan Kaufmann Series in Computer Architecture and Design), 2017, 936 p.
4. James Balfour, William J. Dally, David Black-Schaffer, Vishal Parikh, JongSoo Park. An Energy-Efficient Processor Architecture for Embedded Systems. IEEE COMPUTER ARCHITECTURE LETTERS, VOL. 7, NO. 1, JANUARY-JUNE 2008. P.P. 29-32
5. LaForest C. E. Second-generation stack computer architecture: дис. – University of Waterloo, 2007.
6. Nohl A., Schirrmeister F., Taussig D. Application specific processor design architectures, design methods and tools //Proceedings of the International Conference on Computer-Aided Design. – IEEE Press, 2010. – С. 349-352.
7. Тарасов И.Е., Потехин Д.С., Хренов М.А., Советов П.Н. Автоматизация проектирования многопроцессорной системы на базе ПЛИС для управления во встраиваемых приложениях // Экономика и менеджмент систем управления, 2017, №3.1(25) с. 179 — 184.
8. Puschel M. et al. SPIRAL: Code generation for DSP transforms //Proceedings of the IEEE. – 2005. – Т. 93. – №. 2. – С. 232-275.
9. Ragan-Kelley J. et al. Halide: a language and compiler for optimizing parallelism, locality, and recomputation in image processing pipelines //ACM SIGPLAN Notices. – 2013. – Т. 48. – №. 6. – С. 519-530.
10. Тарасов И.Е. ПЛИС Xilinx. Языки описания аппаратуры VHDL и Verilog, САПР, приемы проектирования. 2019 г. 538 стр.