УДК 004.27

А.Ф. Кургаев, И.В. Савченко

О проектировании специализированного процессора в системе автоматизированного проектирования

Решена задача проектирования специализированного процессора, предполагающая: расширение и выявление набора команд, их планирование, выделение аппаратных ресурсов, связывание команд, декомпозиция устройства управления, организация структуры памяти и распределения в ней.

Solved in the paper is problem of dedicated processor development which is focused on: expansion of the instruction set extension, the processor instructions detection, instructions planning, the allocation of hardware resources, instructions binding, the control unit decomposition, memory structure organization and data distribution.

Розв'язано задачу проектування спеціалізованого процесора, яка передбачає: розширення та виявлення набору команд, планування команд, виділення апаратних ресурсів, зв'язування команд, декомпозиція пристрою управління, організація структури пам'яті а розподілу у ній даних.

Введение. Компьютеры с фон-неймановской архитектурой были и остаются основой вычислительной техники, однако их архитектура не предназначена для эффективной аппаратной поддержки систем обработки знаний. Это связано с тем, что компьютерам с фон-неймановской архитектурой присущ большой семантический разрыв между понятиями операций и объектов в языках высокого уровня и понятиями операций и объектов, определяющими архитектурой компьютера [1].

Объем и сложность задач, решаемых современными компьютерными системами постоянно растет. Это вынуждает разработчиков аппаратных средств искать пути для создания более мощных, быстрых, надежных и одновременно экономичных архитектур компьютерных систем. Принципиальными задачами по-прежнему остаются [2]: повышение производительности процессора, быстродействия памяти и портов ввода-вывода. При этом наиболее актуальна задача повышения производительности процессора.

Повышение производительности процессора достигается решением следующих задач [2]:

– уменьшением размеров логических элементов и плотности их размещения;

– повышением размера, быстродействия и числа уровней кэш-памяти;

– внесением изменений в архитектуру и организацию процессора (использование различных форм параллелизма, конвейеризации команд, введение дополнительного набора специализированных устройств и команд).

Существенным вариантом повышения производительности процессора остается расширение его набора команд [3]. Для повышения эффективности работы систем обработки знаний (СОЗ) в современных процессорах вводятся дополнительные наборы команд. Примером этого является набор команд *SSE*4.2, применяемый для: более эффективной обработки строк и текста в *XML*-файлах, синтаксического разбора предложения, создания меток, оценки регулярных выражений и поиска вирусов [4].

В [5] представлен широкий обзор аппаратных реализаций специализированных процессоров (СП), использующих адаптивные реконфигурируемые аппаратные средства, генетические алгоритмы, нечеткую логику, нейронные сети, параллельные алгоритмы. Актуальны исследования в таких направлениях разработки СОЗ [6]: экспертные системы, системы принятия решения, нейронные сети, нечеткая логика, генетические алгоритмы, рассуждение в условиях неопределенности и др.

К основным этапам проектирования СП можно отнести: анализ спецификации, разработку набора команд и архитектуры (функциональной структуры) процессора, моделирование набора команд, оценку производительности, разработку структуры на *RTL*-уровне (уровне регистровых передач), верификацию [7–11].

Из всего спектра задач проектирования СП выделим следующие: расширение набора команд, автоматическое выявление команд процессора [5], планирование команд, выделение аппаратных ресурсов, связывание команд [12], декомпозиция устройства управления [13], организация памяти и разделение данных в памяти [14].

Цель статьи – рассмотрение сути перечисленных задач.

Разработка набора команд процессора

Набор команд обеспечивает интерфейс между аппаратным и программным обеспечением процессора и, в конечном итоге, определяет его производительность. При создании новой команды необходимо обеспечить минимальную зависимость по данным между ней и другими командами или полностью ее исключить. Если процессор содержит набор команд с невысокой зависимостью по данным, то такой набор считают эффективным. Среди команд можно выделить стандартные и специальные. Стандартные команды выполняют базовые операции с регистрами и памятью, тогда как специальные зависят от специфики предметной области. Определение специального набора команд процессора основано на выявлении в графе операций и данных (ГОД) некоторой последовательности примитивных операций (шаблонов команд), аппаратная реализация которых в форме одной сложной команды существенно повышает эффективность работы СП.

ГОД строится согласно спецификации алгоритма СП. Поиск нового набора команд заключается в итеративном порождении подграфа ГОД и его анализе на соответствие установленным ограничениям по числу входных операндов, объему аппаратных затрат на реализацию и длительность выполнения. Во время работы алгоритма множеству вершин порожденного подграфа ГОД ставится в соответствие множество операций, совместное выполнение которых реализует соответствующий подграф и служит кандидатом на порождение новой, более сложной команды. Полученное множество вершин назовем шаблоном команды. Если шаблон команды отвечает установленным ограничениям, то он сохраняется во множестве шаблонов команд, а вершины ГОД, отвечающие обнаруженному шаблону, объединяются в одну кластерную вершину модифицированного ГОД. Модифицированный ГОД используется на следующей итерации поиска шаблона новой команды, при этом кластерные вершины не принимают участия в поиске шаблонов новых команд. Итеративный поиск шаблонов новых команд длится до тех пор, пока не будут рассмотрены все возможные варианты реализации ГОД или не будут исчерпаны аппаратные ресурсы, доступные для реализации аппаратных модулей поддержки новых сложных команд. Полученное таким образом множество шаблонов команд есть оптимальный набор команд СП.

Разработка архитектуры специализированного процессора

Разработка архитектуры СП выполняется в рамках некоторой его обобщенной структуры (рис. 1) [11] и состоит в разработке функциональных структур двух основных блоков: устройства управления (УУ) и устройства обработки данных (УОД) в составе арифметико-логического устройства (АЛУ), блока регистров и оперативной памяти (ОП).

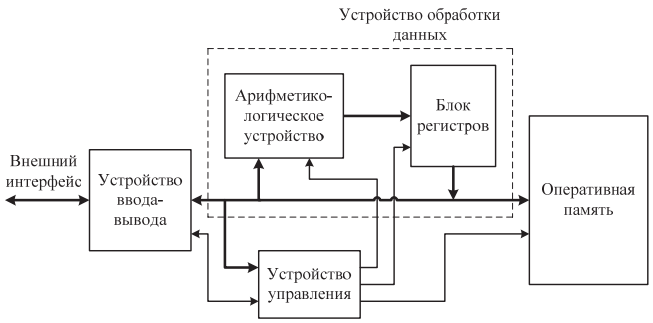

Рис. 1. Пример обобщенной структуры специализированного процессора

Разработка устройства обработки данных

Проектирование УОД выполняется на поведенческом уровне и включает решение задач: планирования – установление последовательности использования аппаратных ресурсов, выделения аппаратных ресурсов и связывания команд. Планирование обеспечивает исключение конфликтов при одновременной работе всех модулей УОД, связывание команд отображает

множество команд в ГОД на множество функциональных модулей (ФМ).

При установлении последовательности использования аппаратных ресурсов необходимо выполнить такие условия:

- Каждая новая команда, полученная в результате анализа ГОД, может выполняться одним или несколькими ФМ. При этом суммарное время выполнения множества операций, которое входит в состав каждой новой команды СП, должно быть меньше, чем время выполнения аналогичного множества операций на универсальном процессоре. Суммарное время выполнения команды состоит из времени: загрузки данных в процессор, выполнения команды и считывания результатов.
- Каждый ФМ может выполнять не более одной команды в один момент времени.
- Команды выполняются в порядке, отмеченном в ГОД.
- Общий объем аппаратных затрат не должен превышать максимальный объем доступных аппаратных ресурсов.

В процессе планирования, а также при выделении аппаратных ресурсов, используются два типа ограничений: относительно зависимости по данным и по времени срабатывания предыдущего модуля. Ограничения относительно зависимости по данным состоят в выполнении команды в момент времени готовности всех ее операндов. Эти ограничения вводятся в ГОД в виде времени срабатывания команды, определяемого по результатам анализа времени выполнения предыдущих команд. Ограничение по времени срабатывания предыдущего модуля зависит от типа и реализации ФМ. Если планируется применить ФМ для выполнения нескольких команд, то выполнение этих команд необходимо организовать в последовательность. Поэтому каждая команда может быть выполнена лишь после выполнения предшествующей команды. Эти ограничения вводятся установлением времени срабатывания каждой команды.

Разработка УОД выполняется итеративно, поиском наилучшей его реализации – с наименьшими аппаратными затратами и наибольшей производительностью.

Разработка устройства управления

Структура синхронного УУ состоит из: логики переходов, памяти состояний и выходной логики. Память состояний представляется набором запоминающих элементов, сохраняющих текущее состояние автомата. Логика переходов определяет следующее состояние автомата и является функцией от входной команды УУ и текущего состояния. Выходная команда определяется средствами выходной логики и есть функцией от входной команды УУ и текущего состояния (для автомата Мили) или только от текущего состояния (для автомата Мура) [15].

В случае СП со сложным алгоритмом функционирования, представляемым большим числом состояний, целесообразно разбить УУ на множество подавтоматов в форме сети автоматов. Декомпозиция УУ на множество подавтоматов сопровождается уменьшением состояний основного УУ. При этом, кроме упрощения общей структуры УУ, достигается повышение быстродействия, надежности и уменьшаются аппаратные затраты.

Декомпозиция УУ выполняется решением задачи формирования на множестве состояний УУ множества кластеров. Эта задача решается в два этапа. На первом этапе формируется множество кластеров, каждый из которых объединяет состояния, вероятность переходов между которыми наибольшая. Граф ГОД модифицируется с учетом найденных кластеров. Каждый кластер представляет множество состояний и переходов между ними, является подграфом ГОД и реализует поведенческую модель соответствующего подавтомата. Полученные таким образом автоматы объединяются в сеть автоматов, с организацией их синхронной работы. На втором этапе выполняется синтез реализации кластеров в форме множества ведущего и ведомых УУ. После этого оцениваются показатели качества реализации полученного УУ. Если полученная реализация УУ не удовлетворяет установленным ограничениям относительно допустимого времени работы и/или затрат аппаратных ресурсов, то выполняется новый вариант декомпозиции УУ [16]. Структура полученного таким образом распределенного ус-

тройства управления (РУУ) представлена на рис. 2.

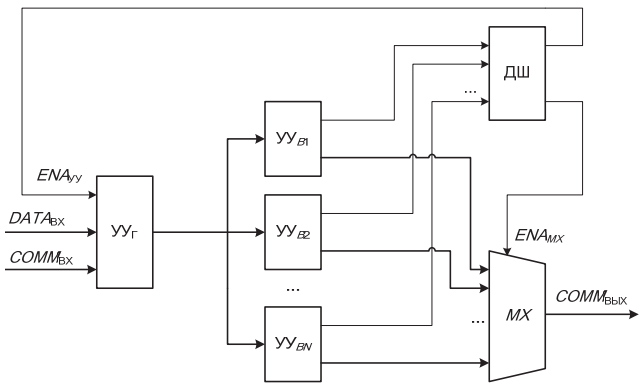

Рис. 2. Функциональная структура распределенного устройства управления

РУУ состоит из главного УУ$_Г$, множества ведомых УУ$_{Вi}$, дешифратора (ДШ) и мультиплексора (*MX*). На вход РУУ поступают входные команды $COMM_{ВХ}$, входные данные $DATA_{ВХ}$, а на выходе формируются выходные команды $COMM_{ВЫХ}$. УУ$_Г$ управляет работой ведомых УУ за счет анализа данных, которые поступают от ДШ. Мультиплексор *MX* коммутирует команды с выходов ведомых УУ на выходную шину $COMM_{ВЫХ}$.

Организация структуры памяти и размещения данных

Доступная на кристалле СП оперативная память не превышает объема от десятков КБ до единиц МБ, не всегда достаточного для решения практически значимых задач. Поэтому доступную память структурируют для оперирования знаниями необходимого объема. Для этого в СП, кроме регистровой памяти, дополнительно вводят кэш-память. Поскольку размер и возможность совместной реализации и кэш-памяти и регистровой памяти ограничивается объемами оперативной памяти, то структура памяти СП может быть реализована в таких вариантах.

• Отсутствует кэш-память и регистровая память. В этом случае доступ к информации во внешней памяти обеспечивается при помощи контроллера внешней памяти. Поскольку длительность запроса к внешней памяти составляет от десятков до сотен процессорных тактов, общая производительность СП остается невысокой.

• Отсутствует кэш-память. В этом случае СП оперирует с регистровой памятью, а при отсутствии в ней необходимых данных – с внешней памятью при помощи контроллера памяти. За счет отсутствия кэш-памяти может быть увеличен размер регистровой памяти. Данная структура обеспечивает более высокую (относительно предыдущего варианта) производительность СП, однако узким местом остается доступ к информации, находящейся во внешней памяти.

• Отсутствует регистровая память. В этом случае СП оперирует с информацией использованием кэш-памяти, размер которой может быть увеличен за счет регистровой памяти. Данная структура обеспечивает существенный рост производительности СП, за исключением тех случаев, когда необходимые данные отсутствуют в кэш-памяти.

• Имеются как регистровая, так и кэш-память. При этом СП оперирует с регистровой и кэш-памятью, объем которых будет меньшего размера, чем в предыдущих двух случаях. Данная структура обеспечивает существенный рост производительности СП. Исключение составляют те случаи, когда необходимые данные отсутствуют в обеих памятях.

Кроме структурирования памяти по времени доступа к данным на производительность СП влияет структурирование памяти по видам хранимых данных и/или выполняемым функциям (функциональная структура).

Для решения задачи разработки функциональной структуры необходимо все константы и переменные, используемые в процессе работы СП, разместить в регистровой памяти, а те, что не размещены в регистровой памяти, в дальнейшем размещаются во внешней памяти. На очередном шаге выполняется группирование оставшихся данных и сортировка сформированных кластеров в порядке уменьшения оценки вероятности доступа к каждому из них. На следующем шаге в порядке уменьшения оценки вероятности доступа анализируется соответствие каждого кластера данных ограничениям по

максимально возможному объему. Эти ограничения вводятся на основе оставшегося свободного участка в регистровой памяти. Оптимальным считается кластер, удовлетворяющий ограничениям на максимальный размер кластера. Если данное условие выполнено, то найденный кластер размещается в регистровой памяти. В результате получим распределение данных в памяти, обеспечивающее максимальную производительность СП.

Параметры спроектированного процессора баз знаний (ПБЗ)

На основе изложенного подхода спроектирована структура ПБЗ, содержащая блоки согласно (рис. 1). Блок регистров содержит 16 функционально специфичных регистров. Арифметико-логическое устройство (АЛУ) выполняет такие простые операции как: инкремент, декремент, инверсию, сброс в «0», установку в «1».

Для этих параметров структуры синтезированы структура и класс команд СП, представленные соответственно на рис. 3 и в табл. 1.

16	13	12	9	8	4	3	0
Поле 4		Поле 3		Поле 2		Поле 1	

Рис. 3. Структура команды процессора баз знаний

Таблица 1. Набор команд процессора баз знаний

Класс команд	Описание класса команд
1	Загрузка внешних данных в $R_{Б1}$
2	Загрузка данных из ОП в $R_{Б1}$
3	Загрузка данных из $R_{Б2}$ в $R_{Б1}$
4	Инкремент $R_{Б2}$ и запись в $R_{Б1}$
5	Декремент $R_{Б2}$ и запись в $R_{Б1}$
6	Запись 0x0 в $R_{Б1}$
7	Запись 0x1 в $R_{Б1}$
8	Инверсия $R_{Б2}$ и запись в $R_{Б1}$
9	Запись данных из $R_{Б1}$ в $R_{01} - R_{15}$, $R_{СМ}$
10	Запись данных из $R_{01} - R_{09}$, $R_{СМ}$ в $R_{ПИ}$
11	Запись данных из $R_{Б2}$ в R_{RA}
12	Запись данных из $R_{Б2}$ в R_{WA}
13	Запись данных из $R_{Б2}$ в R_{DT}
14	Запись данных из R_{DT} в ОП
15	Считывание данных из ОП

Каждая команда ПБЗ имеет длину 17 бит и состоит из 4-х полей. Активные биты поля 1 предназначены для управления загрузкой первого буферного регистра $R_{Б1}$, хранящего промежуточные данные. Активные биты поля 2 управляют загрузкой данных из первого буферного регистра в регистры ПБЗ: $R_{01} - R_{15}$, $R_{СМ}$. Активные биты поля 3 управляют загрузкой данных из регистров $R_{01} - R_{09}$, $R_{СМ}$ во второй буферный регистр $R_{Б2}$. Активные биты поля 4 управляют работой оперативной памяти (ОП), входящей в состав ПБЗ.

В результате синтеза ПБЗ с помощью инструментального средства *Synplify Pro*, входящего в состав САПР *Libero IDE* корпорации *Actel*, получены данные, представленные в табл. 2.

Таблица 2. Аппаратные затраты, необходимые для реализации процессора баз знаний

Наименование	Использовано, шт.	Всего, шт.	Использовано, %
Количество логических ячеек	1506	13824	11
Количество портов ввода–вывода *IO*	102	235	43,4
Количество блоков памяти *RAM*/*FIFO*	8	24	33

С учетом параметров кристалла, размещенного на плате *M*1*AGL-DEV-KIT-SCS*, выбрана структура памяти (рис. 4), включающая внешнюю память и внутреннюю оперативную память.

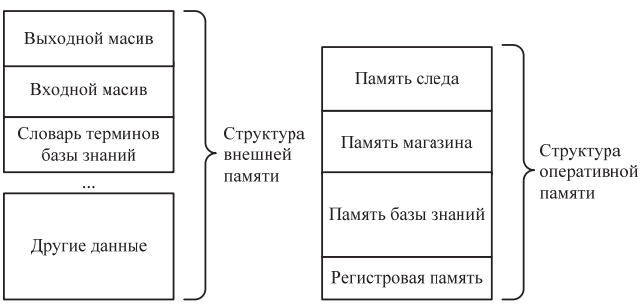

Рис. 4. Структура памяти процессора баз знаний

Внешняя память содержит входной массив ($M_{ВХ}$), выходной массив ($M_{ВЫХ}$), словарь терминов базы знаний и структуру памяти данных, специфичную для системы терминалов приложения. $M_{ВХ}$ и $M_{ВЫХ}$ – линейные массивы однородных данных разрядностью 8 бит каждое, первый из них хранит исходные данные всякой задачи (аргумент вычисляемой функции), второй – результаты решения любой (разрешимой) задачи (вычисляемой функции). Словарь терминов базы знаний хранит индексную таблицу ссылок на машинное представление знаний о соответствующих терминах.

Оперативная память содержит: регистровую память (РП), память БЗ, память магазина (ПМ) и память следа (ПС). Память БЗ хранит машинное представление знаний некоторой проблемы. ПМ хранит текущий процесс вывода решения задачи. ПС хранит структуру вывода решения задачи.

Заключение. Специализация процессоров разработкой и реализацией специальной системы команд обеспечивает существенное повышение эффективности системы. Суть процесса разработки специальной системы команд определена последовательностью задач синтеза набора команд, компонент архитектуры процессора и структуры памяти.

В свою очередь суть синтеза набора команд представлена последовательностью задач:

- синтез ГОД согласно спецификации алгоритма СП;
- итеративное порождение подграфа ГОД или выход, если исчерпаны возможные варианты реализации ГОД;
- анализ подграфа ГОД на соответствие установленным ограничениям по числу входных операндов, объему аппаратных затрат на реализацию и длительность выполнения;
- накопление шаблонов команд (подграфов ГОД), удовлетворяющих установленным ограничениям;
- модификация ГОД и возврат ко второму пункту.

Для синтеза компонент архитектуры процессора определены условия и ограничения планирования аппаратных ресурсов УОД и суть решения задачи реализации УУ в форме иерархической сети подавтоматов, эффективной применительно к СП со сложным алгоритмом функционирования.

Определены варианты структурирования памяти по времени доступа к данным и по видам хранимых данных и/или выполняемым функциям.

Полученные результаты нашли практическое применение при проектировании системы команд и структуры памяти ПБЗ, реализованного на базе кристалла *M*1*AGL*600*V*2-*FGG*484 корпорации *Actel* с использованием 11% логических ячеек и 33% блоков памяти *RAM*.

1. *Амамия М.*, *Танака Ю.* Архитектура ЭВМ и искусственный интеллект. – М.: Мир, 1993. – 400 с.
2. *Stallings W.* Computer Organization and Architecture. Designing for performance. Eight edition / Pearson Education, Inc., 2010. – 763 p.
3. *Galuzzi C.*, *Bertels K.* The Instruction-Set Extension Problem: A Survey // In Proc. of The 4th Intern. Workshop, ARC 2008 London, UK, March 26–28, 2008. – 2008. – P. 209–220.
4. *Intel*® 64 and IA-32 Architectures. Software Developer's Manual. Volume 1: Basic Architecture. Copyright © 2010 Intel Corporation, 2010. – 512 p.
5. *Teodorescu H.N.* Hardware Implementation of Intelligent systems / H.N. Teodorescu, L.C. Jain, A. Kandel. Copyright © Physica-Verlag Heidelberg, 2010. – 282 p.
6. *KES International*. – *http://www. kesinternational.org/aim.php*
7. *Суворова Е.А.*, *Шейнин Ю.Е.* Проектирование цифровых систем на *VHDL*. – СПб.: БХВ-Петербург, 2003. – 576 с.
8. *Грушвицкий Р.И.*, *Мурсаев А.Х.*, *Угрюмов Е.П.* Проектирование систем на микросхемах программируемой логики. – СПб.: БХВ-Петербург, 2002. – 608 с.
9. *Weng Fook Lee.* VLIW Microprocessor Hardware Design For ASIC and FPGA / The McGraw-Hill Comp., Inc., 2008. – 219 p.
10. *Shen J.P.*, *Lipasti M.H.* Modern processor design. Fundamentals of superscalar processors / The McGraw-Hill Comp., Inc., 2005. – 642 p.
11. *Enoch O. Hwang.* Digital logic and Micro-processor design with VHDL / Copyright © Brooks, 2005. – 513 p.
12. *An ILP* Solution for Optimum Scheduling, Module and Register Allocation, and Operation Binding in Datapath Synthesis / T.C. Wilson, N. Mukherjee, M.K. Garg et al. // VLSI Design – 1995. – **3**, №. 1. – P. 21–36.
13. *Automatic* FSM Synthesis for Low-power Mixed Synchronous/Asynchronous Implementation / B. Oelmann, K. Tammemae, M. Kruus et al. // VLSI DESIGN – 2001. – **12**, №. 2. – P. 167–186.
14. *Macii A.* Memory Organization for Low-Energy Embedded Systems // Low-Power Processors and Systems on Chips Edited by Christian Piguet CRC Press 2006. – P. 9–12.
15. *Кургаєв О.П.*, *Савченко І.В*. Проектування управляючої частини IP-блоків на ПЛІС // Радіоелектронні і комп'ютерні системи. – 2010. – № 6. – С. 298–302.
16. *Hasan Z.*, *Ciesielski M.J.* FSM decomposition and functional verification of FSM Networks // VLSI Design. – 1995. – **3**, № 3–4. – P. 249–265.

Поступила 13.01..2012
Тел. для справок: (050) 881-6218, (097) 209-8304 (Киев)
E-mail: *afkurgaev@ukr.net*, *savchenko_ivan@ukr.net*