УДК 681.3; 004.274

Ю.С. ЯКОВЛЕВ

РЕКОНФИГУРИРУЕМЫЕ PIM–СИСТЕМЫ: МЕТОДОЛОГИЯ ПОСТРОЕНИЯ, ПРИМЕРЫ МОДЕЛЕЙ

Abstract: *The substantive provisions of methodology of construction and planning of the reconfigurable PIM-systems are considered by two methods: by programmable logical charts (for example, FPGA) and by the program-driven of communication environment for the choice of resources under the realized applications from scienter entered in the PIM-system of surplus hardware and software tools. The models of the reconfigurable PIM-systems using these methods and the estimations of their parameters are resulted.*
Key words: *"processor-in-memory", reconfigurable systems, methodology of construction .*

Анотація: *Розглянуто основні положення методології побудови та проектування двома способами PIM-систем, що можуть реконфігуруватися за допомогою програмувальних логічних схем (наприклад, FPGA) і за допомогою програмно-керованого комунікаційного середовища для вибору ресурсів під реалізовані додатки зі свідомо уведених в PIM-систему надлишкових програмно-апаратних засобів. Наведено моделі PIM-систем, що можуть реконфігуруватися, які використовують ці способи, а також оцінки їхніх параметрів.*
Ключові слова: *“процесор-в-пам'яті”, реконфігуровані системи, методологія побудови.*

Аннотация: *Рассмотрены основные положения методологии построения и проектирования реконфигурируемых PIM-систем двумя способами: с помощью программируемых логических схем (например, FPGA) и с помощью программно-управляемой коммуникационной среды для выбора ресурсов под реализуемые приложения из заведомо введенных в PIM-систему избыточных программно-аппаратных средств. Приведены модели реконфигурируемых PIM-систем, использующих эти способы, а также оценки их параметров.*
Ключевые слова: *“процессор-в-памяти”, реконфигурируемые системы, методология построения.*

1. Введение

Развитие полупроводниковой технологии обеспечило плотное размещение динамической оперативной памяти (DRAM) и логики КМОП на том же самом кристалле (чипе), что привело к появлению нового класса компьютерных систем, известного как класс “Процессор-в-памяти” (Processor-in-memory – PIM) [1, 2].

PIM отличается от классической системы на чипе тем, что реализуется широкая полоса пропускания память-процессор на уровне 100Гб/с, обеспечивая производительность до 10Гипс (32-разрядные операнды) на чипе памяти с емкостью 16Мбайт. При этом логические схемы для обработки информации имеют прямой доступ ко всем битам строки памяти (например, 2048бит), используя арифметико-логические устройства большой разрядности и соответствующие системы команд [3].

Реконфигурация средств вычислительной техники – это перспективное направление создания и применения компьютерных систем и комплексов, обеспечивающее перенастройку их архитектуры на оптимальное решение пользовательских задач [4]. Поэтому перенесение принципов реконфигурации на PIM-архитектуру при построении компьютерных систем может обеспечить существенный скачок в улучшении их параметров.

Можно выделить три основных подхода к решению проблем, сопутствующих реконфигурации:

1. Реконфигурация с использованием FPGA [4]. Под FPGA (Field Programmable Gate Arrays) понимают кремниевое изделие (чип), состоящее из массива коммутируемых элементов и некоммутируемых ресурсов, подлежащих конфигурации пользователем с помощью средств программирования.

2. Реконфигурация без использования FPGA. Такая реконфигурация осуществляется за счет заведомо заложенной в архитектуру PIM-систем аппаратной и программной избыточности и выбора необходимых ресурсов путем коммутации для их распределения под реализуемые приложения.

3. Комбинированный способ реконфигурации, использующий как заведомо внесенную избыточность ресурсов и выбор их под реализуемое приложение, так и применение FPGA.

Огромный выигрыш в достижении высоких пользовательских характеристик PIM-систем по сравнению с классическими системами (в частности, по производительности и потребляемой мощности) определяет высокий уровень актуальности проблемы построения и применения реконфигурируемых PIM-систем.

2. Основы методологии построения реконфигурируемых PIM-систем

Реконфигурация архитектуры PIM-систем существенно отличается от реконфигурации архитектуры компьютерной системы (КС) в классическом исполнении вследствие того, что при выполнении процедуры реконфигурации пытаются сохранить преимущества PIM-систем перед обычными системами по ширине полосы пропускания процессор-память и, следовательно, по производительности, потребляемой мощности и пр. Поэтому в методологическом плане появилась необходимость ввести новые трактовки таких понятий, как процесс, приложение, операционная система, система адресации, система коммуникации процессов и др.

Реконфигурируемая PIM-система (РК-PIM) включает реконфигурируемую среду, например, специально организованную память FPGA, средства коммуникации как необходимую компоненту для создания требуемой архитектуры системы, а также специальный инструмент для реконфигурации. Микропроцессор либо существует в пределах той же самой структуры, обычно известной как твердое ядро микропроцессора, или конфигурируется на программируемой логике, известной как мягкое ядро.

Топология взаимосвязи PIM-системы при реконфигурации традиционно делится на две группы: статическая и динамическая. Статическая реконфигурируется заранее перед работой системы, динамическая – в процессе её работы.

Описание процесса применительно к РК-PIM включает описание аппаратных средств в виде потокового графа с узлом источника и узлами выдачи данных, описание адресного пространства, а также среды коммуникации между процессами для формирования сообщения и передачи их другим процессам через сеть на чипе и контроллер памяти. Взаимосвязь нескольких процессов может осуществляться как через память с помощью арбитра и контроллера памяти, так и с помощью использования сети с различной топологией (общая шина, звезда, тор и т.д.). Процедура коммуникации состоит из формирования сообщений, которые направляются коммуникационным процессам. При этом она поддерживается предварительно сконфигурированной так называемой *коммуникационной архитектурой*, состоящей, например, из контроллера памяти и коммуникационной сети, размещенной на чипе.

Приложение РК-PIM состоит из двух частей: файла потока двоичных данных (ПДД), который конфигурирует среду (например, FPGA), и хост-программы, которая взаимодействует с платформой PIM-системы.

Операционная система (ОС/РК-PIM) предназначена для управления реконфигурируемым вычислением в РК-PIM и содержит три основных составляющих, определяемых такими понятиями, как процесс, адресное пространство и коммуникация между процессами [5].

В качестве ядра операционной системы (условно обозначим ЯДОС/РК-PIM) можно принять усеченную по функциям ОС/РК-PIM, из которой, например, исключены функции передачи между уровнями.

Общий вид блок-схемы ОС РК-PIM (с использованием FPGA) приведен на рис. 1 [5]. Оболочка в этой операционной системе подобна традиционной и обеспечивает интерфейс между пользователем, операционной системой и аппаратными средствами.

Выполнение приложения начинается с сообщения операционной системе (через оболочку), что есть приложение, ждущее выполнения.

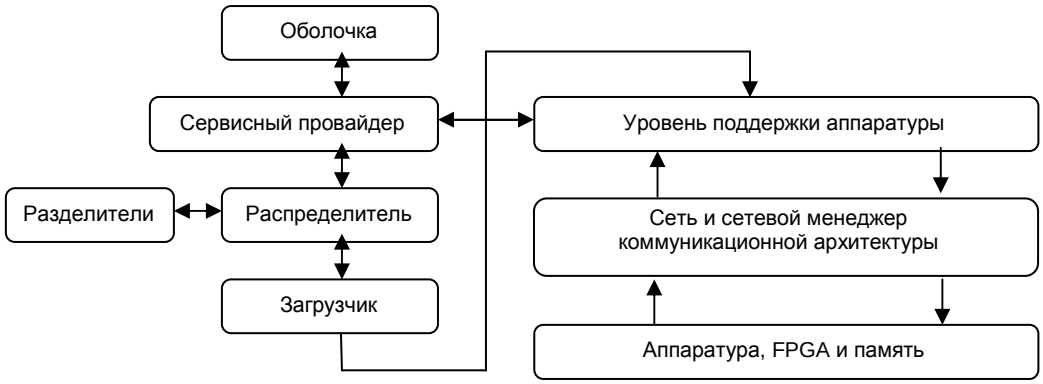


Рис. 1. Укрупненная блок-схема операционной системы (ОС/РК-PIM) с использованием FPGA

Приложение описывается в формате потокового графа данных и передается через сервисный провайдер к программе распределения (Allocation). Алгоритм распределения определяет наличие и местоположение области на FPGA, в которую можно разместить поступающее приложение соответствующего размера. Если есть такая область, но она не находится в одной непрерывной зоне свободного пространства, то приложение будет передано программе разделения (Partitioner), которая разделяет приложение на части (процессы), вмещаемые в распределенную область. При этом алгоритм разделяет приложение, структурированное как потоковый граф, на любое количество частей различного размера. Если вся область использована и приложение не было полностью распределено, то такое приложение блокируется и помещается в соответствующую очередь.

Распределенное приложение передается загрузчику для продуцирования двоичного потока, который будет сконфигурирован на FPGA через уровень поддержки аппаратной части. Если реконфигурируемый компьютер имеет приложения, конкурирующие за ресурсы аппаратных средств, соответствующие механизмы обязаны распределить эти ресурсы таким образом, чтобы они не пересекались с ресурсами уже распределенных (выполняемых) приложений. К таким ресурсам относятся: логическая область FPGA, матрица маршрутизации, контакты ввода – вывода

и внешняя память. Среди известных алгоритмов распределения, которые могут быть использованы в РК-PIM, можно выделить следующие [5]: Greedy based (каскадно основанный), Bottom left (нижний левый), Minkowski Sum (сумма Минковского).

Область FPGA является весьма дорогим ресурсом, поэтому должен быть найден компромисс между временем выполнения приложения и использованием области. Для этого был проделан эксперимент [5], анализ результатов которого позволил сделать следующие выводы:

* Максимальное количество приложений, которое было распределено на FPGA, различно для каждого из указанных выше алгоритмов распределения (табл. 1) и трех размеров приложений (маленькое, типовое и большое по отношению ко всей распределяемой области FPGA).

Таблица 1. Количество приложений, размещенных на FPGA

Алгоритм	Размеры приложений		
	маленькое	типовое	большое
Minkowski Sum	40	23	14
Bottom left	43	25	16
Greedy based	45	28	18

* Времена выполнения для каждого из алгоритмов распределения мало отличаются по значению для всех трех размеров приложений. Тем не менее время выполнения алгоритма для маленьких размеров приложений достигло наибольшего значения по сравнению с другими.

* Самый высокий процент фрагментации был получен при распределении на FPGA приложений маленького размера для всех трех указанных выше алгоритмов распределения.

* Максимальное использование области FPGA изменяется в зависимости от размеров приложений: для маленьких размеров приложений максимальный процент области, которая занята приложениями, по отношению ко всей области FPGA – наибольший. Оставшаяся область FPGA, не занятая приложениями, занимает фрагментация.

* В большинстве случаев размер области, занимаемой на FPGA, увеличился из-за фрагментации.

* Каждый из алгоритмов распределил поступающие приложения в различные местоположения (области) на FPGA.

На основе совокупности экспериментальных результатов сделан вывод, что алгоритм Minkowski Sum является наиболее подходящим из трёх алгоритмов распределения для использования в РК-PIM.

Процедура разделения приложения используется, когда необходимо разделить приложение на части специфического размера для размещения его в областях FPGA с соответствующей геометрической конфигурацией. Одним из популярных алгоритмов является алгоритм разделения приложения по времени реализации (Temporal partitioning). Согласно этому алгоритму первоначально назначается каждый узел в потоковом графе в виде уровня выполнения по принципу “как можно скорее” (“As Soon As Possible” – ASAP). При этом приложение разделяется на множество предопределенных, установленных по размеру частей разделения, которые соответствуют текущему размещению FPGA. Результаты эксперимента отражены на графике (рис. 2).

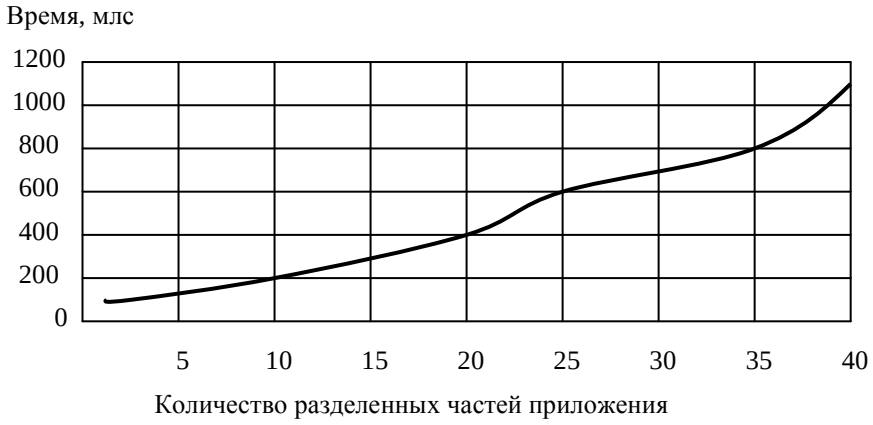
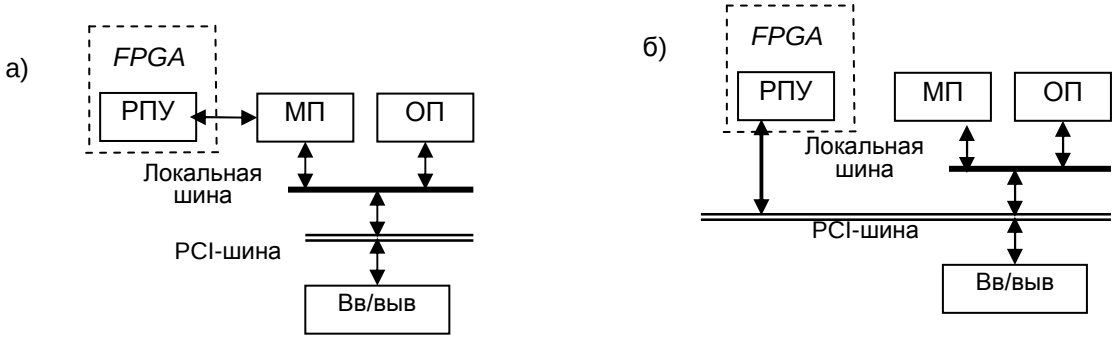


Рис. 2. Текущее время, полученное от алгоритма разделения

По результатам эксперимента определено, что основная часть времени была потрачена алгоритмом разделения на вычисление ASAP уровней каждого узла в потоковом графе, так как приложение имело большое количество узлов (40 узлов) на различных уровнях.

Помимо указанных алгоритмов распределения и разделения логической области FPGA в реконфигурируемых компьютерах, имеются и другие алгоритмы, однако высокая сложность и невысокое качество распределения (с точки зрения потраченной области на FPGA) являются проблемой для некоторых их них.



РПУ – реконфигурируемое процессорное устройство (сопроцессор);
МП – микропроцессор (центральное процессорное устройство);
ОП – основная память;
Вв/выв – устройство ввода/вывода.

Рис. 3. Блок-схема возможных вариантов подключения РПУ

Возможные варианты построения архитектуры PK-PIM. Два возможных варианта построения архитектуры PK-PIM, использующей FPGA, приведены на рис. 3. В первом случае микропроцессор (МП) соединен напрямую с реконфигурируемым процессорным устройством (РПУ), которое можно рассматривать как сопроцессор (рис. 3a), во втором (рис. 3b) – реконфигурируемый сопроцессор соединен с главным процессором через шину ввода – вывода PCI (слабосвязанная архитектура).

Логическая ячейка на FPGA, идентифицируемая как конфигурируемый логический блок (CLB), выполняет логические операции приложения, обычно осуществляемые через специальную

поисковую таблицу. Матрица маршрутизации соединяет CLB друг с другом, используя определенную структуру [4 – 6]. Ячейки ввода-вывода, часто называемые блоками ввода-вывода, соединяются непосредственно с контактами устройства и используются для чтения и записи сигналов, поступающих в/из чипа.

Для управления вводом-выводом и передачей информации между реконфигурируемыми вычислительными приложениями и микропроцессором используется мультиплексирование множества сигналов (множества приложений) через каждый контакт FPGA. Однако количество приложений, которое может совместно использовать контакт FPGA, ограничивается тактовой частотой реконфигурируемого приложения на FPGA. Альтернатива – мультиплексирование контактов с помощью сети и арбитра, размещенных на чипе.

Укрупненная блок-схема проектирования аппаратной части РК-PIM на FPGA может быть представлена в виде, приведенном на рис. 4 [4 – 6].

Для описания работы схемы, как правило, используется язык описания аппаратных средств HDL, который содержит подмножества языков программирования стандартного программного обеспечения типа C, используя подобный синтаксис и соответствующие расширения.

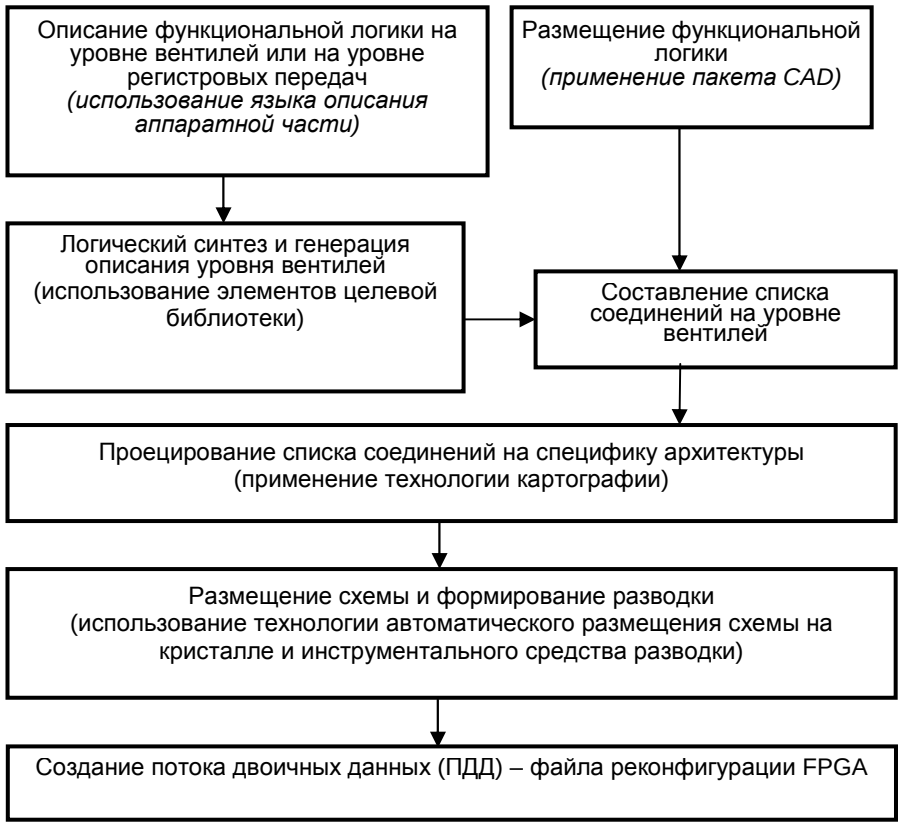


Рис. 4. Укрупненная блок-схема проектирования аппаратной части РК-PIM на FPGA

Следует особо отметить, что для установления конфигурации FPGA и выполнения необходимого взаимодействия между хост-компьютером и реконфигурируемой средой (вычислительной платформой) генерируется поток двоичных данных (ПДД), который загружается на FPGA ведущей хост-программой.

Этот поток устанавливает уровень тактовых сигналов, используя драйвер платформы и программируемый интерфейс, чтобы выполнить необходимые задачи управления. Однако при динамической реконфигурации, что является особенностью применения в РК-PIM современного FPGA, необходимо генерирование дополнительных ПДД, чтобы можно было осуществить конфигурирование на FPGA разных приложений одновременно. Это требует соответствующих изменений традиционного проектируемого потока. Таким образом, создается начальный ПДД для всего проекта и затем индивидуальные ПДД для каждого реконфигурируемого модуля.

Указанные выше особенности архитектуры РК-PIM отражены в предложенной методологии их построения, компоненты которой представлены в табл. 2.

Таблица 2. Общая методология построения реконфигурируемых PIM-систем

Исходная информация	Компоненты методологии	Получаемый результат
1	2	3
1. Выбор подхода к решению проблемы конфигурации		
* Описание предметной области и пользовательской задачи. * Требования к параметрам конфигурируемой системы. * Информация о конфигурируемых системах. * Информация о возможностях интегральной технологии и элементной базы. * Наличие необходимых инструментальных средств проектирования в соответствии с выбранным подходом к конфигурации	* Поиск аналогов систем с различными подходами к решению проблемы конфигурации. * Анализ архитектурно-структурных особенностей аналогов и выбор подхода к построению реконфигурируемой системы (с применением FPGA, без применения FPGA, комбинированный подход). * Определение принципов взаимодействия реконфигурируемой области с другими компонентами системы	* Выбранный подход к решению проблемы реконфигурации проектируемой системы. * Предварительный вариант архитектуры реконфигурируемой системы. * Принципы взаимодействия реконфигурируемой области с другими компонентами архитектуры в составе системы. * Выбранный набор инструментальных средств проектирования
2. Уточнение архитектуры системы и проектирование потока управления реконфигурацией		
* Технологии реконфигурации для выбранного подхода. * Известные алгоритмы функционирования систем подобного типа. * Существующие операционные системы. * Существующий поток управления реконфигурацией	* Анализ и выбор технологии реконфигурации (при необходимости её доработка). * Выбор и доработка операционной системы. * Анализ существующих потоков управления реконфигурацией, выбор и доработка потока в соответствии с подходом к реконфигурации	* Новый поток управления реконфигурацией. * Модифицированная операционная система. * Проект архитектуры системы. * Спецификации алгоритмов функционирования системы
3. Распределение ресурсов и разделение приложений		
* Информация по размещению и разделению приложений. * Метрики программного обеспечения	* Выбор и доработка алгоритмов распределения памяти, размещения и разделения приложений. * Измерение параметров выбранных алгоритмов	* Алгоритмы распределения памяти, размещения и разделения приложений. * Показатели программного обеспечения
4. Построение модели реконфигурируемой системы		
* Информация по программному обеспечению и разработанной ОС для проектируемой системы. * Метрики потока управления конфигурацией. * Спецификации алгоритмов функционирования системы. * Параметры архитектуры проектируемой системы	* Разработка программной (либо аппаратно-програм-мной) модели реконфигурируемой системы. * Определение метрик (показателей) приложений. * Определение состава проектируемой системы	* Модель проектируемой реконфигурируемой вычислительной системы. * Метрики разработанной ОС. * Перечень аппаратно-программных средств реконфигурируемой системы

Продолжение табл. 2

1	2	3
5. Оценка параметров модели		
* Модель проектируемой вычислительной системы. * Приложение FPGA или другой области конфигурации. * Набор инструментальных средств для оценки параметров модели	* Построение теста конфигурации. * Измерение параметров приложений. *Оценка фрагментации. * Анализ корреляции результатов. * Оценка параметров по результатам моделирования на удовлетворение исходным требованиям по основным показателям системы	* Тест конфигурации. * Параметры результатов моделирования. * Корреляционные факторы. * Выводы о целесообразности перехода на другой подход к конфигурации и тип модели
6. Проектирование аппаратной части реконфигурируемой области (например, на FPGA, см. рис. 4)		
* Перечень аппаратно-программных средств проектируемой системы. * Параметры результатов моделирования. * Пакет CAD, язык описания аппаратной части, целевая библиотека элементов, технология картографии, инструментальные средства разводки и др. (рис. 4)	* Описание логики на уровне вентилей и её размещение. * Логический синтез и генерация описания уровня вентилей, составление списка соединений. * Проецирование списка соединений на специфику архитектуры системы. * Размещение схемы и формирование разводки. * Создание файла реконфигурации	* Файл реконфигурации (например, FPGA)

Доработка операционной системы, а также алгоритмов распределения и разделения приложений, отмеченные в табл. 2, объясняется тем, что имеется несколько уникальных особенностей реконфигурируемого компьютера, которые не позволяют стандартные программы использовать без модификации в реконфигурируемой вычислительной среде.

Во-первых, в стандартной операционной системе процесс состоит из последовательных команд и данных, которые распределяются в памяти и обращаются через линейное адресное пространство. В PK-PIM *адресное пространство*, используемое в операционной системе, состоит из двумерного адресного пространства для области конфигурации (например, FPGA) и одномерного адресного пространства для встроенной памяти.

Во-вторых, в стандартной операционной системе память является главным ресурсом, подлежащим распределению. В системе PK-PIM распределению подлежат логические ячейки (CLB), память, штырьки ввода – вывода микросхем, провода (линии) маршрутизации и др., распределения которых могут требовать процессы. Поэтому алгоритмы распределения адресного пространства должны быть изменены, чтобы удовлетворить этой комплексной среде.

В-третьих, способность совместно использовать логическую схему на реконфигурируемом компьютере намного труднее, чем совместное использование программы, сохраненной в памяти. В реконфигурируемом компьютере все схемы не могут быть разделены между процессами. Примером такой схемы является контроллер памяти, который управляет чтением и записью информации отдельного банка оперативной памяти для нескольких процессов. Однако только один процесс одновременно может читать информацию или писать её в память, и, как только эти процедуры будут выполнены, следующий процесс может использовать общедоступную схему. Эти три уникальных особенности PK-PIM не позволяют без модификации использовать адресное пространство программного домена классической операционной системы.

3. Примеры моделей реконфигурируемых PIM-систем

Ниже рассматриваются типовые модели реконфигурируемых PIM-систем: модель реконфигурируемого PIM-компьютера типа ReConfigME, использующая FPGA [5]; модель PIM-системы, использующая реконфигурацию информационного канала [7], и модель реконфигурируемой процессорной памяти [8]. В каждом из обозначенных РК-PIM реализован соответствующий подход к решению проблемы реконфигурации.

Модель РК-PIM типа ReConfigME, использующая FPGA

В табл. 3 представлен состав модели ReConfigME, которую условно можно разделить на три уровня: уровень операционной системы, уровень пользователя и уровень платформы. ReConfigME поддерживает множество приложений путем распределения ресурсов аппаратных средств FPGA, логического разделения приложения, генерации двоичного потока и непосредственно выполнения реконфигурации.

ReConfigME использует статическое распределение памяти приложения и имеет предельное значение количества параллельных (конкурирующих) приложений. Пользователи соединяются с ReConfigME через специальный интерфейс пользователя, который дает им возможность загрузить приложения, передавать данные, информацию о конфигурации и монитор состояния реконфигурируемой вычислительной платформы.

FPGAs и их средства проектирования не поддерживают динамическую реконфигурацию приложений произвольного размера, поэтому ReConfigME при необходимости моделирует динамическую реконфигурацию FPGAs.

Уровень операционной системы ReConfigME (табл. 3) содержит компоненты, которые реализуют распределение и разделение приложения, генерацию двоичного потока FPGA, передачу данных приложения и информации о конфигурации между уровнем платформы и пользовательским уровнем. Вершина уровня компонент операционной системы дублируется ядром ОС (ЯДОС), которое содержит необходимые субкомпоненты и инструментальные средства генерации двоичного потока. ЯДОС управляет передачей данных приложения, получает входные данные от пользователя, загружает их во встроенную память, читает выходные данные от встроенной памяти и передает их назад пользователю. Эта задача прежде всего содержит адресную трансляцию. Локальная схема адресации транслируется в глобальную схему адресации платформы, чтобы гарантировать правильное местоположение памяти платформы при любом чтении или записи.

ЯДОС также передает специфические синхроимпульсы и информацию платформы между HAL-клиентом и пользовательским сервером.

Приложение и его предварительно определенные геометрические размеры поступают на программный распределитель и вместе с программным разделителем определяется, может ли приложение быть конфигурированным на FPGA или блокировано и помещено в очередь из-за нехватки свободной области памяти.

Еще одним компонентом уровня операционной системы являются клиент HAL и пользовательский сервер (табл. 3). Клиент HAL реализует подключение к желаемой платформе и передает информацию ввода-вывода, двоичные потоки и информацию о конфигурации.

Таблица 3. Уровни модели ReConfigME и их состав

Наименование основных компонентв модели	Состав компонентов модели по уровням		
	операционной системы	пользователя	платформы
FPGA с дополнительной памятью	–	–	+
Библиотека платформы	–	–	+
Виртуальная машина Java	+	–	+
HAL - сервер	–	–	+
HAL - клиент	+	–	–
Средства взаимосвязи HAL – клиента и HAL - сервера	+	–	+
Ядро операционной системы ReConf ME	+	–	–
Сервер пользователя	+	–	–
Пользователь	–	+	–
Средства взаимосвязи пользователя с сервером пользователя	+	+	–
Операционная система основного компьютера типа PC	+	+	+
Средства стандартной сети (TCP/IP)	+	+	+
Интерфейс пользователя с ReConf ME	–	+	–

Пользовательский сервер реализует связь между пользователем и ЯДОС, подключая к ReConfigME через стандартные гнезда TCP/IP многочисленные отдаленно расположенные клиенты.

Пользовательский уровень обеспечивает интерфейс пользователя и подключение его к уровню операционной системы. Он содержит обычный (стандартный) интерфейс, командную строку для ввода и графический интерфейс, чтобы геометрически отображать размещение выполняемого приложения на реконфигурируемой вычислительной платформе. С помощью командной строки пользователи загружают приложения, реализуют передачу данных к встроенной платформе памяти и специфические параметры настройки конфигурации платформы типа значений синхросигналов.

Графический пользовательский интерфейс отображает результаты распределения и разделения приложений, когда они загружены в ReConfigME. Пользовательский клиент имеет доступ к ЯДОС через стандартные гнезда TCP/IP пользовательского сервера на уровне операционной системы и преобразовывает пользовательские запросы командной строки интерфейса в API для взаимосвязи между пользователем и компонентами сервера.

Уровень платформы. Аппаратная часть уровня платформы ReConfigME размещена на макетной плате Celoxica RC1000pp в типичной конфигурации сопроцессора. Платформа состоит из FPGA, слабосвязанной с современным микропроцессором через стандартную шину PCI, и имеет четыре банка двухпортовой памяти большой емкости. Процессы могут общаться друг с другом и внешним микропроцессором через встроенную платформу памяти. Внешние данные ввода-вывода могут быть загружены в банк памяти через шину PCI и затем переданы в процесс через контакты FPGA и контроллер памяти. Этот тип передачи ввода-вывода требует двухпортовой памяти, так как хост и FPGA могут общаться непосредственно с банком памяти.

Приложения имеют доступ к 1МБ памяти, при этом каждое стартует на виртуальный адрес. Затем контроллер памяти преобразует этот виртуальный адрес в реальный адрес, основанный на статическом распределении внешней памяти.

Контроллер памяти, размещенный на чипе, совместно с сервером ReConfigME организует чтение и запись данных к/от приложений и соответствующего местоположения памяти. Этот интерфейс позволяет приложениям быть запрограммированными в VHDL или в Handel-C.

Коммуникационная часть уровня платформы ReConfigME конфигурируется на FPGA перед любыми пользовательскими приложениями. Она состоит из контроллера памяти и сетевых терминаторов. Контроллер памяти предоставляет доступ к памяти по требованию приложения и управляет передачей ввода-вывода приложения. Он должен гарантировать, что хост и приложения FPGA не выполнят одновременно запись в тот же самый банк памяти. Для простоты выполнения контроллер памяти логически делит память на блоки фиксированного размера.

Для выполнения приложения необходимо создать два типа файлов, которые загружаются на реконфигурируемый компьютер с помощью операционной системы ReConfigME: файл EDIF непосредственно с приложением для каждого узла потокового графа и файл класса Java, который определяет, как каждый из этих файлов EDIF связан вместе в потоковой модели графа.

Для проверки функциональных возможностей модели ReConfigME на данной аппаратной платформе были реализованы следующие приложения [5]:

– Blob tracking, которое обычно используется в системе технического зрения и реализует процесс обнаружения местоположения известного объекта в ряду изображений;

– Edge enhancement, обычно используемое для идентификации граней объектов в изображении на первой стадии построения шаблона соответствия или целевого распознавания.

Для измерения времени ответа и пропускной способности ReConfigME было определено эталонное приложение и разработана испытательная среда, генерирующая моделируемые рабочие нагрузки. В качестве эталонного приложения был выбран алгоритм шифрования данных (Data Encryption Algorithm – DES) с некоторыми доработками [5]. По результатам экспериментов сделаны следующие выводы:

* Самое низкое пользовательское время ответа было получено, когда DES-приложение было распределено без разделения на пустом FPGA.

* Пользовательское время ответа и сигнал задержки значительно увеличивается, когда приложение разделено, при этом тем больше, чем больше количество разделений.

* Общим фактором, который вызывает потерю времени ответа и пропускной способности, является фрагментация FPGA.

* При разделении приложения время ответа увеличивается, потому что при этом используется большое количество межпроцессных маршрутов коммуникации.

* Важным полезным свойством систем типа ReConfigME является возможность динамической реконфигурации, когда вновь поступающее на FPGA приложение может быть принято, распределено в памяти и при необходимости разделено на множество частей различного размера и конфигурации, что позволяет использовать РК-PIM для решения различных классов задач.

Модель PK-PIM с реконфигурацией информационного канала

В [7] представлена схема модели PIM, основанного 32-разрядного реконфигурируемого информационного канала с низким энергопотреблением, который оптимизирован для приложений мультимедиа. Архитектура PK-PIM, использующая этот канал, представлена четырьмя 8-битовыми элементарными процессорами (PE) невысокой сложности, которые можно отнести к категории интеллектуальных ОЗУ типа RAM (IRAM), интегрирующих логику обработки и блоки памяти на том же самом чипе. Схема структурирована в виде двух ступеней логической обработки, управляемых операндным диспетчером, который при необходимости реконфигурации выбирает соответствующие 8-битовые подсегменты загруженных 32-разрядных слов для параллельной обработки каждым PE.

Входной узел первой логической ступени содержит восемь 8-разрядных регистров вместе с логикой, необходимой для форматирования входных данных. Все биты строки блока памяти (то есть 32-разрядное слово) могут быть получены одновременно за одно обращение к памяти.

Вторичная ступень логики обработки используется для того, чтобы объединять частичные результаты, сгенерированные PE, для повышения точности операций умножения. Она также включает схему для нормализации и вычисления признака для операций с плавающей запятой.

Сеть подключения, представленная в виде ступеней мультиплексирования, посылает операнды конечному сумматору, который является быстрым частичным сумматором и может выполнять одно 64-разрядное или два независимых 32-разрядных сложения.

PE работают по принципам SIMD, эффективно выполняя за счет реконфигурации канала обработки параллельные арифметические операции над 8, 16, или 32-разрядными целочисленными данными или над 32-разрядными данными с плавающей запятой. Четыре PE связываются, используя мультиплексор, основанный на логике управления переносом.

Информационный канал конфигурируется через соответствующие SIMD команды, поступающие от внешней логики управления. Каждая SIMD команда определяет код операции для PE, чтобы выполнять различные операции.

Чтобы определить эффективность рассмотренной схемы, было выбрано приложение мультимедиа – сжатие изображения. Результаты экспериментов показали, что компьютерная система с реконфигурируемым каналом вследствие высокой модульности, низкой сложности и низкого энергопотребления может быть эффективно использована для обработки интенсивных алгоритмов, требующих высокой пропускной способности по каналам процессор-память-устройства ввода/вывода особенно при необходимости настройки на обработку 8 или 16-разрядных слов с фиксированной запятой или 32-разрядных слов с плавающей запятой.

Реконфигурируемая процессорная память

В [8] описана система памяти с реализацией функций хранения и обработки информации на одном кристалле (чипе), обладающая свойствами реконфигурации архитектуры. Такой чип памяти, содержащий процессоры (до нескольких тысяч), каждый из которых соединен с соответствующим столбцом оперативной памяти и размещен на одной интегральной схеме (ИС), может обеспечить увеличение производительности на несколько порядков по сравнению с обычными компьютерными системами с классическими архитектурами. Множество процессоров на чипе управляются

параллельно, соединены в группы или системы банков памяти, расширяя или заменяя существующие подсистемы памяти в компьютерах различных классов (от персональных компьютеров до супер-ЭВМ). Укрупненная блок-схема одного PIM-чипа процессорной памяти, поясняющая взаимосвязь основных функциональных компонентов, приведена на рис. 5.

PIM-чип может работать в двух режимах: как обычная память для чтения – записи и для вычисления (режим PIM), при этом адрес представляется декодеру строки через контакты чипа.

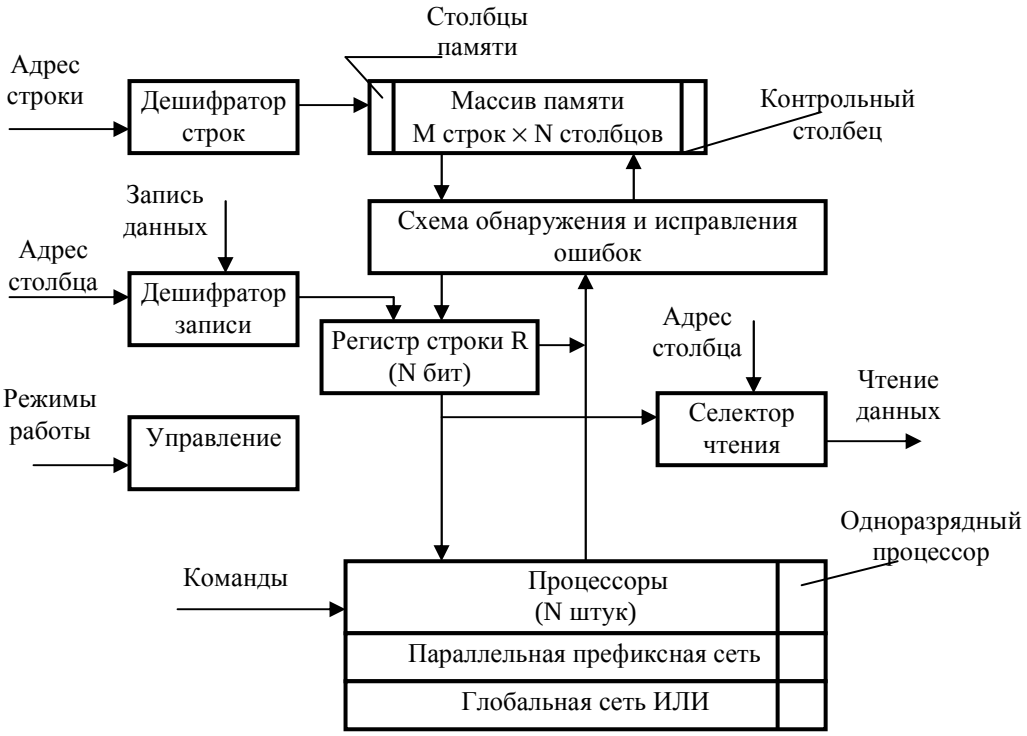


Рис. 5. Блок-схема PIM-чипа процессорной памяти

При записи информации в память и чтении информации из памяти всегда происходит считывание полного N-разрядного слова, хотя для выполнения конкретной арифметической или логической операции часто требуются r-разрядные слова $(r << N)$. При этом команду выполняют все процессоры над всеми разрядами длинной N-разрядной строки, но большинство процессоров (кроме задействованных с нужными данными) не сохраняют результаты вычислений, т.е. часть процессоров записывает обратно в память только что считанную из неё информацию.

Здесь селектор чтения используется для выделения считанного слова данных разрядностью r из считанной N – разрядной строки $(r << N)$, которая при чтении заносится на регистр строки, а дешифратор записи – для размещения при записи r – разрядного слова данных на соответствующую позицию N – разрядного регистра строки.

Когда чип используется для записи в режиме обычной памяти, данные сначала читаются из памяти, исправляется ошибка и затем помещаются в регистр строки R. После этого содержание регистра строки R с измененными данными направляется через логику исправления ошибки к памяти. Когда чип используется для чтения в режиме обычной памяти, строка данных берется из

памяти, исправляется ошибка и помещается в регистр строки R. В следующем тактовом цикле биты адреса столбца выбирают требуемое подмножество битов данных, которые будут выведены из чипа.

“Глобальная” сеть ИЛИ (GOR) предназначена для связи процессоров между собой по принципу “многие – к одному” или “один – ко многим”, а “префиксная” сеть (PPN) – для связи процессоров по принципу “многие – ко многим”.

Метод управления банком памяти PIM-чипов реализуется с помощью шин адреса и данных. В обычном режиме памяти для выполнения чтения-записи банк PIM-чипов получает адреса строки и столбца на адресной шине и данные для чтения-записи на шине данных. В режиме PIM банк PIM чипов получает адрес строки на адресной шине, а на шине данных – команду SIMD, которая должна быть выполнена.

PPN-сеть реализована в виде нескольких логических уровней, управляемых процессорами. Первый уровень позволяет отправлять данные одного процессора налево при получении данных от процессора справа. Следующий уровень позволяет определенным процессорам отправлять данные на следующие два процессора слева и т.д. Все процессоры получают данные от всех уровней. Требуемый уровень выбирается исполняемой программой с использованием соответствующих управляющих шин.

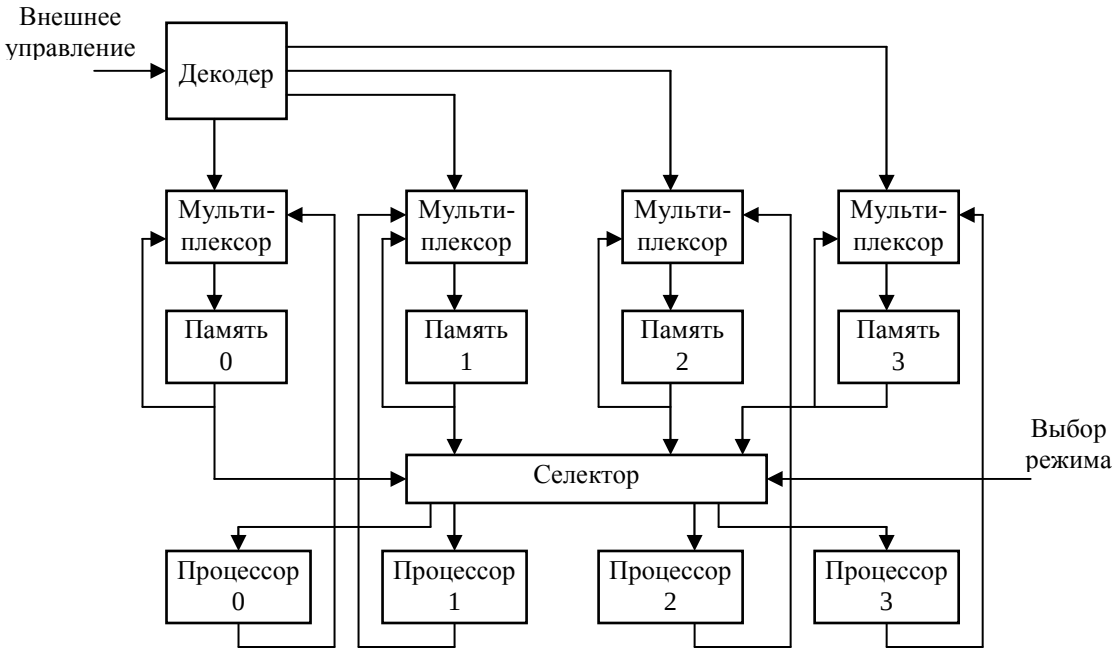


Рис. 6. Блок-схема реконфигурируемой процессорной памяти

Организация архитектуры реконфигурируемой PIM-системы выполнена таким образом, что множество устройств памяти и процессоров объединены в группы, каждая группа содержит μ устройств памяти, μ процессоров, μ мультиплексоров, один селектор и один дешифратор мультиплексоров, соединенные между собой и с другими узлами системы соответствующими связями (рис. 6).

Каждый процессор в пределах группы работает с теми же самыми данными. Когда данные должны быть сохранены, процессор, соответствующий адресу запоминаемых данных, посылает

недавно вычисленный результат в память, в то время как процессоры в пределах группы, которые не соответствуют адресу памяти записываемого слова, записывают обратно предварительно выбранные старые данные.

Таким образом, множество устройств памяти имеют соответствующее множество связанных процессоров. При этом селектор подключает выходы устройств памяти ко входам процессоров так, чтобы каждый процессор имел как вход, так и выход от одного из блоков памяти.

4. Выводы

Реконфигурация архитектуры PIM-систем существенно отличается от реконфигурации архитектуры системы в классическом исполнении, что привело в методологическом плане к необходимости ввести новые трактовки сущностей глобальных классических понятий, таких как процесс, приложение, операционная система, система адресации, система коммуникации процессов и др., а также доработать известные алгоритмы распределения памяти для размещения приложений в реконфигурируемую среду и алгоритмы разделения приложений на составные части для оптимального использования памяти.

Возможны несколько вариантов построения архитектуры РК-PIM, использующей FPGA: например, микропроцессор соединяется напрямую с реконфигурируемым процессорным устройством, которое можно рассматривать как сопроцессор, или реконфигурируемый сопроцессор соединяется с главным процессором через шину ввода-вывода PCI.

Важным свойством модели системы типа ReConfigME является возможность динамической реконфигурации, когда вновь поступающее на FPGA приложение может быть принято, распределено в памяти и при необходимости разделено на множество частей различного размера и формы, не затрагивая области уже распределенных приложений. Это позволяет использовать РК-PIM для решения различных классов задач.

Поскольку плотность FPGA увеличивается за 10 миллионов конфигурируемых вентилей и при этом имеется возможность динамической реконфигурации, то целесообразно использовать единственное устройство FPGA высокой плотности для реализации нескольких приложений, которые когда-то требовали одного FPGA каждое. Если реконфигурируемый компьютер имеет приложения, конкурирующие за ресурсы аппаратных средств, механизмы и методика обязаны распределять эти ресурсы таким образом, что они не должны пересекаться с выполнением других приложений. Эти ресурсы включают логическую область FPGA, матрицу маршрутизации, контакты ввода-вывода и внешнюю память.

Модель РК-PIM с реконфигурацией информационного канала так же, как и модель реконфигурируемой процессорной памяти, использующие для целей реконфигурации программно-управляемые коммуникационные сети, вследствие высокой модульности, высокой производительности и низкого энергопотребления могут быть эффективно использованы для обработки интенсивных алгоритмов, требующих высокой пропускной способности по каналам процессор-память особенно при необходимости настройки на обработку слов различной разрядности или слов с плавающей запятой.

СПИСОК ЛИТЕРАТУРЫ

1. Палагин А.В., Яковлев Ю.С., Тихонов Б.М. Основные принципы построения вычислительных систем с архитектурой “Процессор-в-памяти” // Управляющие системы и машины. – 2004. – № 5. – С. 30–37.
2. Палагин А.В., Яковлев Ю.С., Тихонов Б.М., Першко И.М. Архитектурно-структурная организация компьютерных средств класса “Процессор-в-памяти” // Математичні машини і системи. – 2005. – № 3.– С.3–16.
3. Яковлев Ю.С., Тихонов Б.М. Об оптимизации размещения данных в PIM-системе // Математичні машини і системи. – 2006. – № 3. – С. 24–35.
4. Палагин А.В., Опанасенко В.Н. Реконфигурируемые вычислительные системы: Основы и приложения. – К.: Просвіта, 2006. – 280 с.
5. Brian G.W. An Operating System for Reconfigurable Computing. April 2005. –
– http://www.library.unisa.edu.au/adt-root/uploads/approved/adt-SUSA-03062005-155342/public/02whole.pdf.
6. Грушвицкий Р.И., Мурсаев А.Х., Угрюмов Е.П. Проектирование систем на микросхемах программируемой логики. – СПБ.: БХВ-Петербург, 2002. – 608 с.
7. Marco Lanuzza, Мартин Margala, Pasquale Corsonello Cost-Effective Low-Power Processor-In-Memory-based Reconfigurable Datapath for Multimedia Applications.– http: // portal.acm.org/ft_gateway.cfm?id=1077645& type=pdf.
8. Kenneth W. Resnick, David R. Wallgren, Kenneth R. Reconfigurable memory processor. United States Patent №5.396.641, Intern'l Class: G06F 013/00, U.S. Class: 713/100. 07.03.95 March 7, 1995. – 14 p.