7. «The HPC Challenge (HPCC) Benchmark Suite» / P. Luszczek [et al.] SC06 Conference Tutorial, IEEE, Tampa, Florida, November 12, 2006.

8. CUDA Zone. – URL: http://www.nvidia.ru/object/cuda_home_new_ru.html.

[1]И.И. Левин, [2]А.И. Дордопуло, [1]В.А. Гудков

[1]Научно-исследовательский институт многопроцессорных вычислительных систем имени академика А.В. Каляева Южного федерального университета, г. Таганрог

[2]Южный научный центр РАН, г. Ростов-на-Дону

ПРИНЦИПЫ РАЗРАБОТКИ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ ДЛЯ РЕКОНФИГУРИРУЕМЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ НА ЯЗЫКЕ ВЫСОКОГО УРОВНЯ COLAMO

Реконфигурируемые вычислительные системы демонстрируют высокую реальную производительность при решении расчетоемких сильносвязанных задач различных классов. Вместе с тем сдерживающим фактором для широкого распространения реконфигурируемых вычислительных систем (РВС) являются сложность и трудоемкость их программирования, а также отсутствие удобных высокоуровневых средств разработки приложений.

Для программирования РВС наибольшее распространение получили языки HDL-группы, позволяющие описать структуру и функционирование вычислительной системы, а также моделировать обмен данными между блоками системы. Наиболее популярным языком этой группы является язык VHDL, обладающий высоким функциональным уровнем абстракции. Однако громоздкое текстовое описание устройств на VHDL обусловило популярность и широкое использование систем автоматизированного проектирования в виде специализированных графических редакторов, таких как Xilinx ISE, Altera MaxPlus или OrCad. Такой подход к программированию РВС требует участия в программировании системы как специалиста-схемотехника,

создающего конфигурацию вычислительной системы с учетом особенностей ее архитектуры и элементной базы, так и прикладного программиста, создающего параллельную программу, описывающую потоки данных в созданной схемотехником виртуальной вычислительной структуре.

Естественное желание программировать РВС на языке высокого уровня привело к созданию ряда процедурных языков, таких как ImpulseC, Mitrion-C, Catapult C, Handel-C и др. Для языков этой группы характерен привычный для большинства программистов персональных ЭВМ синтаксис языка С, а различия проявляются в семантических особенностях вызова и использования специализированных параллельных операторов. При этом для описания параллельных процессов в этих языках используется изначально последовательный язык С, семантика которого ориентирована на взаимодействие последовательных процессов, что не позволяет в полной мере использовать все возможности РВС. Это приводит к семантическому разрыву между исходным информационным графом алгоритма задачи, его описанием на языке высокого уровня и созданной транслятором схемотехнической реализацией, что выражается в существенном снижении эффективности прикладной программы. Как правило, созданные с помощью указанных языков приложения имеют в 3–5 раз более низкую производительность по сравнению с их аналогами, разработанными на более низких уровнях абстракции.

Поэтому разработка новых предложений в области средств программирования РВС, сочетающих удобство описания информационных графов задач на языке высокого уровня с эффективностью и возможностями языка VHDL, представляется актуальной научной задачей, решение которой позволит расширить традиционные области применения РВС.

Принципы программирования на языке высокого уровня COLAMO. В Научно-исследовательском институте многопроцессорных вычислительных систем имени академика А.В. Каляева Южного федерального университета более 20 лет

развивается оригинальное научное направление по созданию реконфигурируемых вычислительных систем с динамически программируемой архитектурой различной конфигурации и их программного обеспечения. Программирование созданных образцов осуществляется на языке программирования высокого уровня COLAMO [1].

Язык COLAMO предназначен для описания реализации и создания в архитектуре РВС специализированной вычислительной структуры на основе принципов структурно-процедурной организации вычислений [2], которая предполагает последовательную смену структурно (аппаратно) реализованных фрагментов информационного графа задачи, каждый из которых является вычислительным конвейером потока операндов. Таким образом, приложение (прикладная задача) для РВС состоит из структурной составляющей, представленной в виде аппаратно реализованных фрагментов вычислений, и процедурной составляющей, представляющей собой единую для всех структурных фрагментов управляющую программу последовательной смены вычислительных структур и организации, в каждой структуре соответствующих потоков данных. Для представления такой организации вычислений в языке используется понятие «кадр» [1,2], являющийся неразрывной совокупностью вычислительной структуры фрагмента задачи и множества операций чтения-записи входных и результирующих потоков данных.

Более строго, кадром является программно-неделимый компонент, представляющий собой совокупность операторов, которые реализуются в виде арифметико-логических команд и команд чтения/записи, выполняемых на различных функциональных устройствах, соединенных между собой в соответствии с информационной структурой алгоритма.

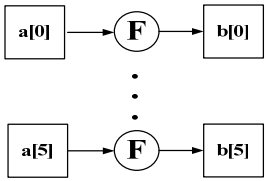
В языке COLAMO отсутствуют явные формы описания параллелизма, а распараллеливание достигается с помощью объявления типов доступа к переменным и индексации элементов массивов, что характерно для языков потока данных. Для обращения к данным используются два основных метода доступа: параллельный доступ (задаваемый типом Vector) и последо-

вательный доступ (задаваемый типом Stream). Степень параллелизма определяется по минимальному значению параметра распараллеливания. Для типа доступа Stream степень параллелизма равна 1, а для типа доступа Vector определяется наименьшим значением векторной составляющей каждого массива, участвующего в вычислениях. Для параллельного типа доступа возможна одновременная обработка всех размерностей массивов, заданных типом Vector, при этом повышается аппаратный ресурс на обработку, но снижается время обработки.

Многомерные массивы данных могут иметь множество измерений, каждое из которых может иметь последовательный или параллельный тип доступа, задаваемый ключевым словом Stream или Vector соответственно. Если вычисления связывают между собой массивы с различным типом доступа, то заданная пользователем вычислительная структура может оказаться несбалансированной, что приведет к затрате дополнительного оборудования и снизит скорость обработки потоков данных.

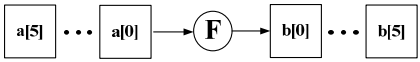
На рисунке представлены примеры простейших программ на языке COLAMO, отличающихся только типом доступа к массиву, и графы синтезируемых вычислительных структур. Данный пример показывает, что одна и та же программа в зависимости от типа доступа к данным при описании массива может быть реализована как параллельно, так и последовательно, причем синтез соответствующей описанию вычислительной структуры осуществляет транслятор.

```
Var a,b: Array Integer [10 : Vector] Mem;
Var i : Number;
Cadr FVector;
   For i := 0 to 5 do
      b[i] :=F(a[i]);
EndCadr;
```



а

```
Var a,b : Array Integer [10 : Stream] Mem;
Var i : Number;
Cadr FStream;
   For i := 0 to 5 do
      b[i] :=F(a[i]);
EndCadr;
```



б

Рис. Параллельная (*а*) и последовательная (*б*) обработка массивов

Изменение типа доступа позволяет достаточно просто управлять как степенью распараллеливания вычислений на уровне описания структур данных, так и скоростью обработки и занимаемым ресурсом, что позволяет программисту описывать различные виды параллелизма в достаточно сжатом виде.

Помимо типа доступа для переменной в языке COLAMO определен также и тип ее хранения: мемориальный (Mem), регистровый (Reg) и коммутационный (Com).

Мемориальной переменной называется величина, хранящаяся в ячейке распределенной памяти и, следовательно, сохраняющая свое значение до очередного переприсваивания. Для мемориальной переменной возможно одновременное выполнение только одного процесса. Поэтому в семантике языка COLAMO для мемориальной переменной в теле кадра действуют правило однократного присваивания и правило единственной подстановки. Правило однократного присваивания указывает на то, что мемориальная переменная в кадре изменяет свое значение только один раз. Правило единственной подстановки определяет, что переменная в выражении может использоваться только для одного процесса: чтения или записи.

Для описания связей между элементами информационного графа задачи в языке COLAMO предназначена коммутационная переменная. Поскольку коммутационная переменная описывает информационные связи, она не требует никакого вычислительного аппаратного ресурса для своей реализации. Доступ к значению коммутационной переменной после выполнения кадра невозможен. Коммутационная переменная необходима транслятору для указания информационных зависимостей при построении вычислительной структуры задачи. На коммутационную переменную, как и на мемориальную переменную, действует правило однократного присваивания, но не действует правило единственной подстановки. Использование коммутационных переменных позволяет легко разветвлять и дублировать потоки данных, но не позволяет реализовать рекурсию.

Для организации рекурсии в языке COLAMO используется регистровая переменная, которая представляет собой регистр

на аппаратном уровне и используется для хранения промежуточных данных, полученных в процессе вычислений. Единственным ограничением для регистровой переменной в теле кадра является правило однократного присваивания.

Для управления порядком выполнения операций и реализации ветвлений в языке COLAMO предусмотрен оператор условного перехода IF, который может быть внутренним или внешним по отношению к кадру. Внутренний по отношению к кадру оператор условного перехода, расположенный в теле кадра, реализуется на структурном уровне: обе ветви условия выполняются параллельно, а для выбора результата используется мультиплексор. Реализации внешнего оператора условного перехода осуществляются процедурной составляющей путем вызова того или иного кадра в зависимости от условия.

Расширения языка высокого уровня COLAMO. Программирование РВС на уровне логических ячеек ПЛИС (LUT), когда программист использует специализированные аппаратно-реализованные блоки, соответствующие арифметико-логическим операциям программы, и создает связи между ними в соответствии с информационным графом задачи, позволяет создать более компактные аппаратные решения для конкретной задачи, учитывающие специфику проблемной области, необходимый формат представления информации, а также требования к производительности задачи.

Для объявления постоянной вычислительной структуры при программировании на уровне ячеек ПЛИС в стандарте языка COLAMO используется конструкция LET, которая описывает функционально законченный неизменяемый подграф в информационном графе задачи. При этом программа может содержать единственное описание вычислительной структуры LET, вызов которой обязательно (по умолчанию) осуществляется в теле каждого кадра программы. В этом случае кадры отличаются друг от друга информационными потоками, подключенными на входы и выходы конструкции LET.

Переключение информационных потоков при вызове конструкции LET в кадре параллельной программы может осуществ-

ляться двумя основными способами: сдвоенным контроллером распределенной памяти (КРП), для автоматического формирования которого транслятором необходимо выполнение ряда условий, или с помощью дополнительных коммутаторов, реализация которых требует как затрат оборудования, так и предварительной настройки коммутаторов перед выполнением каждого кадра.

Современная архитектура ПЛИС позволяет использовать внутреннюю память ПЛИС, которая обладает достаточной емкостью, высокой скоростью доступа, а также возможностью организации одновременного доступа к памяти нескольких независимых процессов чтения и записи по разным портам. Поддержка работы с внутренней памятью ПЛИС в языке COLAMO осуществляется с помощью нового типа хранения переменных – InterMem **<N>**, где N – количество доступных портов при работе с внутренней памятью ПЛИС.

Для решения задач символьной обработки в языке COLAMO предусмотрена поддержка битовых переменных и соответствующего им набора операций и функций. Обработка битовых переменных, которые объявляются ключевым словом Bit, в языке осуществляется вычислительными операциями, выполняющими логические функции (and, or, xor, not), и логическими поразрядными операциями (операции сдвига влево и вправо, операции циклического сдвига влево и вправо и др.). Языковые средства позволяют получить доступ к любому биту в слове высокоуровневыми конструкциями и адресовать биты операторами индексного доступа к элементам массива. Также в языке поддерживаются групповые операции («срезы») для произвольного измерения массива любого типа, позволяющие сократить запись.

Для сокращения объема программы и повышения ее читабельности предназначены макросы обработки битовых переменных, позволяющие выполнять битовые операции не на уровне элементов битовых массивов, а на уровне 32-разрядных или 64-разрядных слов.

Язык программирования реконфигурируемых вычислительных систем COLAMO с учетом описанных расширений позволяет рационально использовать ресурсы ПЛИС при про-

граммировании РВС на уровне логических ячеек и обеспечивает пользователя набором необходимых средств для быстрой разработки эффективных параллельных программ.

Список литературы

1. Каляев А.В., Левин И.И. Модульно-наращиваемые многопроцессорные системы со структурно-процедурной организацией вычислений. – М.: Янус-К, 2003. – 380 с.

2. Реконфигурируемые мультиконвейерные вычислительные структуры / И.А. Каляев [и др.]. – изд. 2-е, перераб., доп. / под общ. ред. И.А. Каляева. – Ростов н/Д: Изд-во ЮНЦ РАН, 2009. – 344 с.

3. Левин И.И. Многопроцессорная система с программированием архитектуры на нескольких уровнях // Методы и средства обработки информации: тр. Первой Всерос. науч. конф. – М.: Изд-во МГУ, 2003. – С. 111–118.

4. Семейство многопроцессорных вычислительных систем с динамически перестраиваемой архитектурой / А.И. Дордопуло [и др.] // Высокопроизводительные вычислительные системы: материалы IV Междунар. науч. молодежн. школы. – Таганрог: Изд-во ТТИ ЮФУ, 2007. – С. 68–74.

5. Левин И.И. Ресурсонезависимое параллельное программирование // Искусственный интеллект. – Донецк: Наука і освіта, 2002. – № 3. – С. 277–285.

С.Ю. Лесовой, А.В. Панюков

Южно-Уральский государственый университет, г. Челябинск

ПРИМЕНЕНИЕ МАССИВНО-ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ ДЛЯ РЕАЛИЗАЦИИ ОСНОВНЫХ ОПЕРАЦИЙ ЦЕЛОЧИСЛЕННОЙ АРИФМЕТИКИ

Существенным недостатком вычислений с помощью аппаратных типов данных с плавающей точкой является погрешность. Данную проблему позволяет решить использование производных дробно-рациональных типов данных, в которых числа