Гудков В.А.

СРЕДСТВА АВТОМАТИЗАЦИИ РАЗРАБОТКИ ПРОГРАММ ДЛЯ РЕКОНФИГУРИРУЕМЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

НИИ многопроцессорных вычислительных систем имени академика А.В. Каляева Южного федерального университета, г. Таганрог
E-mail: Slava_Gudkov@mail.ru

Повышение скорости разработки и отладки параллельных программ для многопроцессорных вычислительных систем (МВС) является одной из наиболее актуальных в настоящее время задач в области высокоскоростных вычислений.

Особо остро данная проблема стоит для реконфигурируемых вычислительных систем (РВС), для которых необходимо не только создание параллельных программ, но и разработка конфигурации вычислительной системы для выполнения программы. Разработка параллельной программы для МВС с реконфигурируемой архитектурой и создание конфигурационных файлов ПЛИС для данной задачи требует участия как квалифицированных программистов, так и специалистов-схемотехников.

Новая версия транслятора языка с неявным описанием параллелизма COLAMO [1] обеспечивает перевод параллельной программы в загрузочный модуль РВС и генерацию необходимых структур данных для среды разработки специальных конфигураций.

Для разработки специальных конфигураций можно использовать среду Fire!Constructor [2], которая позволяет распределить вычислительный граф задачи на произвольную архитектуру МВС.

Перед генерацией структур данных для среды Fire!Constructor транслятор выполняет оптимизацию вычислений. Для оптимизации вычислений в языке COLAMO используется представление графа задачи в ярусно-параллельной форме (ЯПФ). При оптимизации графа задачи выделяется подграф, образующий цепочку ассоциативных операций (умножение, сложение, конъюнкция, дизъюнкция) и выполняет ее представление в пирамидальной форме. Если при работе с числами с плавающей запятой необходима абсолютная точность производимых вычислений, то выполнение оптимизации вычислительного подграфа, содержащего неассоциативные операции или операции сложения, не производится.

На рисунках 1 и 2 представлены информационные графы выражения $k := A_1 * A_2 * A_3 * A_4 * A_5 * A_6 * A_7 * A_8$ без оптимизации и с оптимизацией, соответственно.

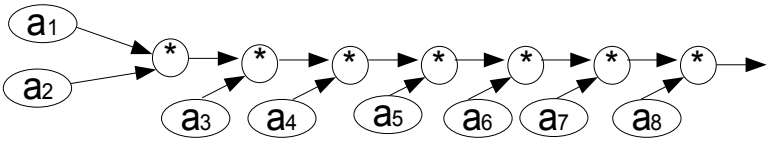

Рисунок 1 - Неоптимизированный граф

Математическое представление изображенного на рисунке 1 графа описывается следующей формулой:

$$h = \bigotimes_{i=1}^{N} A_i ,$$

где θ - ассоциативная операция, N – количество входных вершин графа.

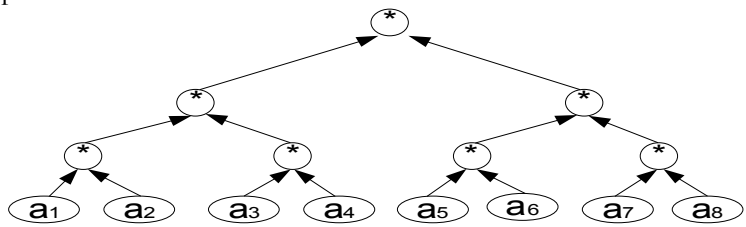

Рисунок 2 - Оптимизированный граф

Математическое представление изображенного на рисунке 2 графа описывается следующей формулой:

$$h = \bigotimes_{i=0}^{\log_2 N} \bigotimes_{j=1}^{N/(2^i)} A_k ,$$

где $k = 2 * N - \left[\frac{N}{2^{i-1}} \right] + j$.

После построения ЯПФ выполняется синхронизация вычислительных потоков. Для этого в информационном графе выделяются подграфы, и для каждой дуги подграфа рассчитывается задержка относительно критического пути подграфа.

Рассмотрим пример программы на языке COLAMO.

```
Program Primer;
Include Library;
Var A,B,C,D : array [N : Stream, K : Stream] : Mem;
Define N=5;
```

```
Define K=6;
Cadr
  For I := 0 to N-1 do
    For J := 0 to K-1 do
      B[I, J] := (A[I,J] + A[I,J + 1])* (A[I,J + 2] * C[I,J+2])+(C[I,J]
+ D[I,J+1]);
EndCadr;
End_Program.
```

В языке программирования COLAMO массивы различаются по способу обработки на векторы (Vector) и потоки (Stream). К элементам векторов можно обращаться параллельно, а к элементам потоков только последовательно.

Замена хотя бы одной из размерностей массива со Stream на Vector позволяет мультиплицировать информационный граф (т.е. повысить степень распараллеливания по данным).

Результатом трансляции программы является синхронизированный информационный граф, показанный на рисунке 3.

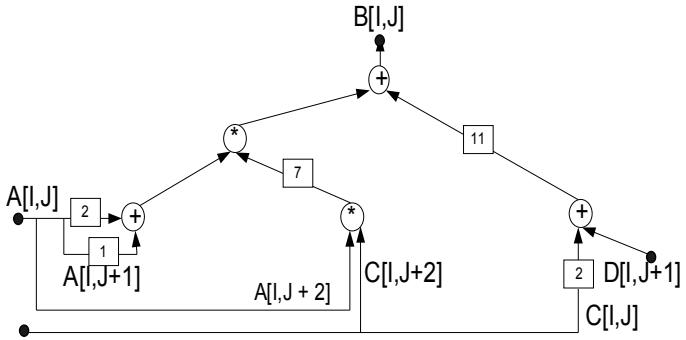

Рисунок 3 - Синхронизированный информационный граф задачи

Здесь вершина в виде зачерненного кружка отображает входную/выходную вершину графа, которая соответствует контроллеру распределенной памяти (КРП).

Вершины графа, обозначенные светлыми кружками, являются операциями.

Вершины графа, обозначенные прямоугольниками, указывают на задержку информации на определенное число тактов.

Синхронизация информационного графа заключается в вычислении информационных задержек для дуг данного графа. Информационные задержки рассчитываются, исходя из задержек, вносимых операционными вершинами графа, и из смещения данных,

которое требуется при обращении к данным, расположенным в одном канале памяти.

Полученный информационный граф задачи с синхронизированными вычислительными потоками передается в среду Fire!Constructor с указанием подключаемых библиотечных элементов (список подключаемых библиотек указывается в операторе Include языка COLAMO).

Среда Fire!Constructor представляет полученный граф в виде вычислительных блоков и интерфейсов, после чего спроектированный вычислительный граф (рисунок 4) укладывается на ресурс МВС с реконфигурируемой архитектурой.

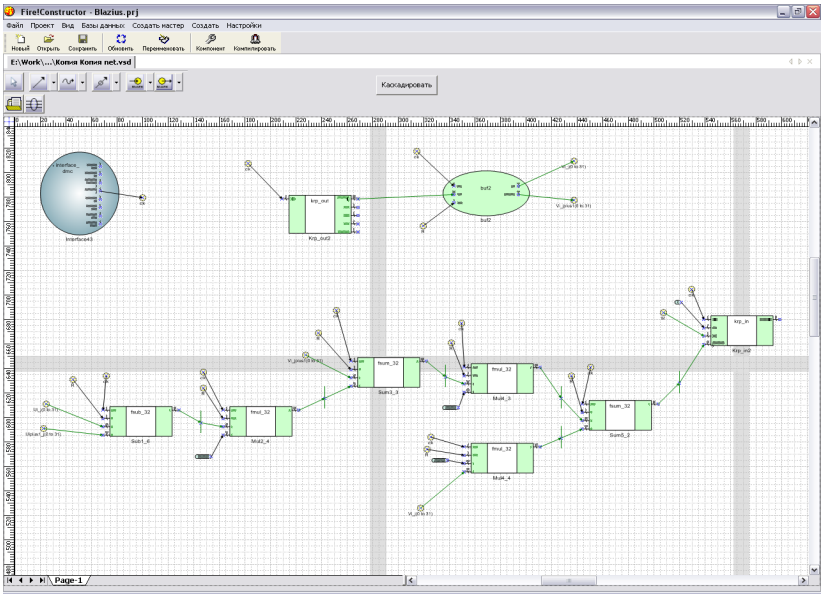

Рисунок 4 - Спроектированный вычислительный граф задачи

Помимо генерации структур данных для среды Fire!Constructor транслятор выполняет перевод с языка COLAMO в параллельную программу на языке структурно-процедурного программирования ARGUS. Язык ARGUS позволяет не только отобразить вычислительный граф задачи на специализированную конфигурацию, полученную из среды Fire!Constructor, но и организовать потоки данных для данной конфигурации.

Фрагмент сгенерированной программы на языке ARGUS выглядит следующим образом:

```
Define N = 5;
Define K = 6;
Define M = 1;

DMC[0.A]#
Metka : Read A step M repeat K; Add A, K;
Loop N Goto Metka;

DMC[0.C]#
Metka1 : Read C step M repeat K; Add C, K;
Loop N Goto Metka1;

DMC[0.D]#
Metka2 : Read D step M repeat K; Add D, K;
Loop N Goto Metka2;

DMC[0.B]#
Metka3 : Write B step M repeat K; Add B, K;
Loop N Goto Metka3;
```

Здесь Read – оператор, обеспечивающий чтение потока данных из КРП в специализированную конфигурацию (т.е. реализуется внутренний цикл транслируемой программы), Add – оператор приращения адреса, реализующий переход на следующую строку двумерного массива, Loop – оператор цикла, реализующий внешний цикл транслируемой программы, Write – оператор записи потока данных в КРП (т.е. реализуется внутренний цикл программы). Имена переменных соответствуют именам переменных, используемых в транслируемой программе на языке COLAMO.

Таким образом, благодаря новой версии транслятора впервые становится возможным создание параллельных прикладных программ для реконфигурируемых систем в едином языковом пространстве.

В результате различного взаимодействия транслятора и других компонентов системного программного обеспечения программист освобождается от сложных механистических процедур, вследствие чего уменьшаются временные затраты на создание эффективных параллельных программ.

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

2. Гуленок А.А. Среда разработки масштабируемых структурных компонентов для реконфигурируемых вычислительных систем. Проблемы информационно-компьютерных технологий и мехатроники // Международная научно-техническая конференция. Таганрог: Изд-во ТТИ ЮФУ, 2007.