УДК 004.272.2

РАСПАРАЛЛЕЛИВАНИЕ И ПЛАНИРОВАНИЕ ВЫЧИСЛИТЕЛЬНЫХ И ИНФОРМАЦИОННЫХ ПРОЦЕССОВ

А.А. ПРИХОЖИЙ, М.В. СОЛОМЕННИК

Высший государственный колледж связи
Староборисовский тракт, 8/2, Минск, 220114, Беларусь

Поступила в редакцию 15 июля 2003

Представлены результаты разработки и исследования сквозной технологии распараллеливания и планирования информационно–вычислительных процессов, специфицированных на алгоритмических языках описания аппаратуры и программ. Технология включает: 1) измерение вычислительной сложности поведенческого описания, его критического пути и предельно возможного ускорения за счет распараллеливания; 2) оптимизационный синтез последовательно-параллельного плана; 3) оптимизационный синтез сетевого плана; 4) эквивалентное преобразование поведенческого описания с целью улучшения параметров параллельной системы. Экспериментальные результаты показывают реальную применимость технологии для решения практических задач защиты и сжатия информации, обработки аудио- и видео-информации, обработки информации в компьютерных сетях и сетях телекоммуникаций, других приложений.

Ключевые слова: распараллеливание, критический путь, планирование, преобразование, оптимизация.

Введение

Современные алгоритмы обработки и сжатия информации, коммуникационные протоколы и мультимедиасистемы достигли чрезвычайно высокой степени сложности. Понимание таких систем и приведение их к эффективным параллельным реализациям возможно лишь посредством использования современных абстрактных алгоритмических языков описания аппаратуры, программ и аппаратно-программных систем. Описания на алгоритмических языках, таких, как C, C++, VHDL, Verilog и др., часто являются огромными и включают десятки, сотни тысяч и миллионы строк кода. В связи с этим большей частью они разрабатываются в последовательном варианте. Получение эффективных параллельных реализаций невозможно без автоматизированных методов, технологий и инструментов оптимизационного распараллеливания и планирования.

Оптимизация в ходе распараллеливания вычислений — важнейшая задача, решаемая в процессе разработки и использования высокопроизводительной информационно-вычислительной системы. Результаты ее решения определяющим образом влияют на конечные параметры системы. Обычно задача оптимизации формулируется в виде проблемы планирования или построения расписания. Многообразие в способах организации параллельных систем и требованиях, предъявляемых к системам, влечет многообразие в формальной постановке задачи планирования. Пространство возможных распараллеливаний системы увеличивается также благодаря принципиальному существованию и возможности конструктивного построения различных описаний для одного и того же поведения. Существуют описания, на базе которых организация параллельных процессов является

наиболее эффективной и производительной. Нашей целью является также выявление правил перехода к такого рода описаниям.

Проблема распараллеливания вычислительных и информационных процессов

Проблема параллелизма находится в поле пристального внимания и исследования на протяжении последних десятилетий. В ее рамках сформулированы и решены разнообразные теоретические и практические задачи, в которых нашло отражение современное на текущий момент времени состояние развития: 1) компьютеров и средств телекоммуникаций, 2) вычислительной науки, 3) программного обеспечения, 3) средств программирования, 4) средств обработки информации и информационного обмена и т. д. Переход от реализации вычислительных процессов на синхронно работающем компьютере к их реализации на множестве компьютеров, объединенных в сеть и работающих асинхронно параллельно, привел к необходимости разработки новых подходов к построению моделей параллелизма и отображению существующих или вновь создаваемых алгоритмов к их адекватным параллельным версиям. На практике важен не только сам переход, но и то, насколько эффективно он выполнен. Исходя из этого, разработчиками технологий и средств распараллеливания сформулированы разнообразные оптимизационные задачи, ставящие целью нахождение оптимальных значений одних параметров реализации параллельных вычислений при заданных ограничениях на другие параметры. Наиболее важными параметрами являются время выполнения алгоритма или производительность системы, а также объем используемых вычислительных ресурсов.

С целью выявления направлений в области распараллеливания вычислительных и информационных процессов выполним анализ тематик Международного симпозиума по параллельной и распределенной обработке [1] (International Parallel and Distributed Processing Symposium — IPDPS) и Международной конференции по параллельным вычислениям в электронике и электротехнике [2] (Parallel Computing in Electrical Engineering — PARELEC). Представленные доклады можно отнести к пяти главным направлениям (рис. 1):

параллельные и распределенные архитектуры;
средства разработки параллельных и распределенных систем;
параллельные и распределенные алгоритмы и модели;
параллельные и распределенные приложения;
сетевые вычислительные и информационные процессы.

Проблема оптимального планирования параллельных процессов

Планирование или построение расписания (scheduling) [3–8] является наиболее важной задачей, решаемой при реализации параллельных вычислительных процессов, так как ее результаты в значительной степени влияют на конечные параметры проектируемой или эксплуатируемой информационно-вычислительной системы. Современные методы последовательно-параллельного планирования синтезируют план, вводя шаги управления и состояния в поведенческое описание, распределяют операторы и операции по шагам с целью их параллельного выполнения в пределах одного шага, выполняют минимизацию числа шагов или объема вычислительных ресурсов, необходимых для обеспечения установленного уровня параллелизма.

Сетевое планирование [3–5] не вводит шагов управления. Оно определяет два взаимодополняющих бинарных отношения между операторами: отношение последовательного выполнения и отношение параллельного выполнения. Построенный сетевой план однозначно устанавливает поток данных, поток операций и поток ресурсов, которые обеспечивают корректные вычисления при заданном уровне распараллеленности. Целью сетевого планирования является оптимизация асинхронных систем, которые могут быть представлены с помощью сетевых моделей.

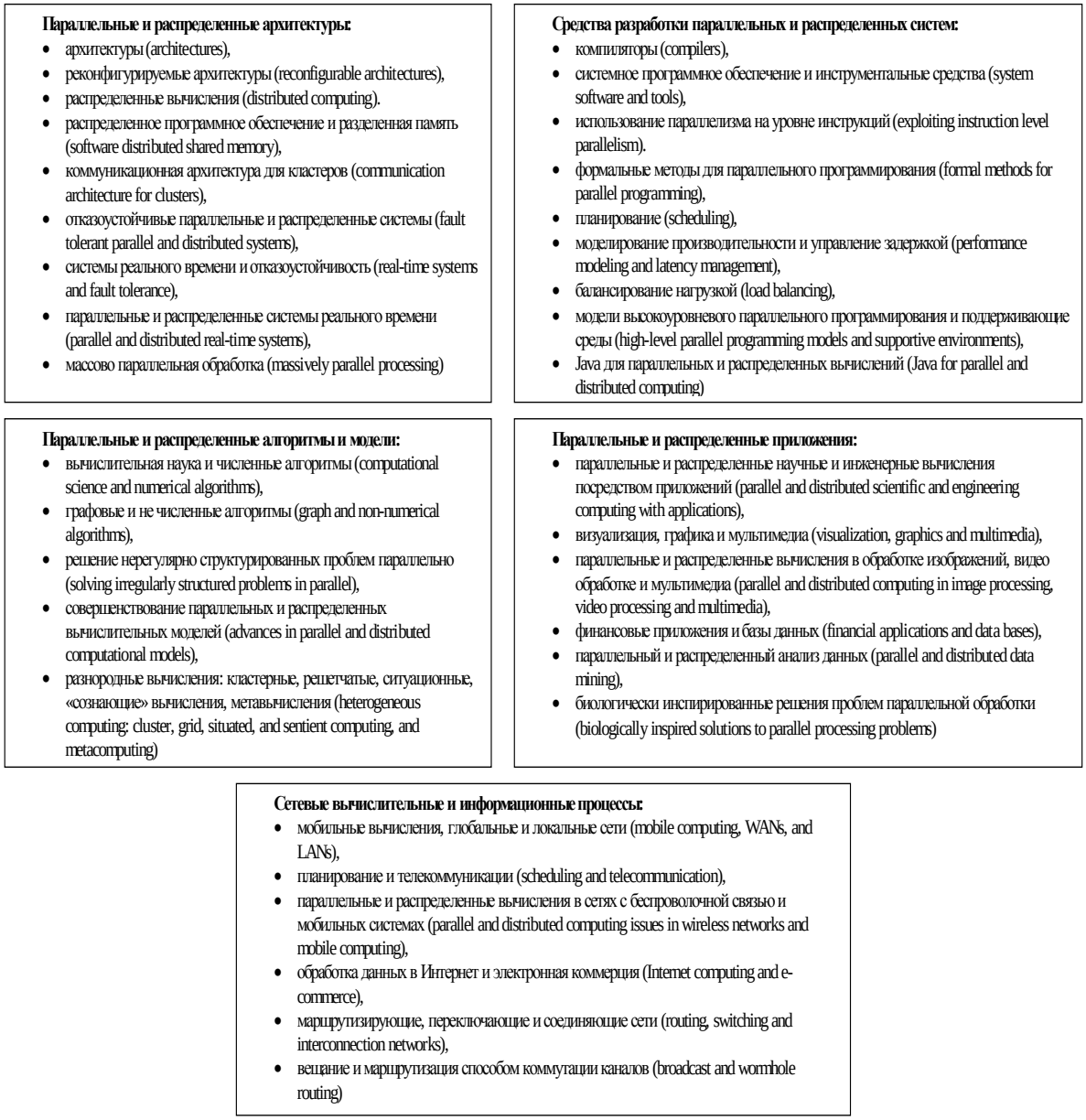

Рис. 1. Классификация направлений и тем в области параллельной и распределенной обработки

Известны два глобальных подхода к разработке параллельных информационно-вычислительных систем. Первый состоит в "ручном" создании параллельных описаний с использованием подходящих языков, технологий и сред программирования. Второй базируется на автоматическом распараллеливании и планировании информационно-вычислительных процессов. Каждый из подходов имеет свои преимущества, однако очевидно, что второй подход является менее трудоемким и более мобильным.

В основе автоматического распараллеливания и планирования вычислений лежит модель параллелизма, выразительная сила которой является достаточной для представления рассматриваемого класса целевых параллельных архитектур.

Ряд технологий распараллеливания использует граф потока данных (DFG — data flow graph) и граф потока управления (CFG — control flow graph) в качестве внутренней или промежуточной модели [9] при синтезе параллельных планов и реализаций. Объединенная модель называется графом потока управления данными (CDFG — control-data flow graph). Модель высокоуровневого конечного автомата является популярной и эффективной моделью при синтезе последовательно-параллельных планов. Особенностью модели является

связывание параллельно выполняемых операций и операторов из поведенческого описания, представленного на одном из алгоритмических языков, с состояниями автомата и переходами из одного состояния в другое.

Наиболее абстрактной моделью, представляющей параллельное асинхронное поведение, является Сети Петри. Граф управления CASCADE [10] может описывать и синхронный конечный автомат и асинхронное устройство управления, в котором есть параллелизм. Сети предикат/транзакция (Predicate/Transition Nets) [10] являются моделью системного уровня, включающей параллелизм и охватывающей и управление и вычисления. Асинхронная методология и модель проектирования "Микроконвейеры" (Micropipelines) представляют асинхронную систему с использованием унифицированного набора блоков, управляемых событиями.

Оптимизационные задачи планирования параллельных вычислений имеют три классические постановки [6].

1. Минимизировать время выполнения распараллеливаемого поведенческого описания при заданных ограничениях на информационно-вычислительные ресурсы (resource-constrained scheduling).

2. Минимизировать используемые информационно-вычислительные ресурсы при заданном ограничении на время выполнения распараллеливаемого поведенческого описания (time-constrained scheduling).

3. Проверить принципиальную возможность построения параллельного плана, выполняемого за время, не превышающее установленное и не нарушающее заданные ограничения на информационно-вычислительные ресурсы (feasible-constrained scheduling).

Критический путь, его профиль и предельное распараллеливание

Проблема идентификации одного из самых длинных взвешенных по времени путей в информационно-вычислительной системе называется проблемой критического пути [7, 8, 11–18]. Анализ критического пути является эффективным механизмом, используемым для решения разнообразных задач на различных уровнях разработки и эксплуатации системы: схемном, логическом, архитектурном, поведенческом, системном. На системном уровне процесс разделения на аппаратные и программные части формулируется как сложная оптимизационная задача [19]. Анализ критического пути обеспечивает более эффективное разбиение на части и оптимизацию параметров каждой из частей.

Существует два направления использования критического пути на системном, алгоритмическом и архитектурном уровнях разработки и эксплуатации параллельной системы. Первым является построение профиля критического пути [13] с целью настройки уже существующего параллельного описания для эффективного выполнения на ранее разработанной архитектуре (рис. 1,*a*). Другим направлением, развиваемым в работе [11], является измерение критического пути с целью оценки потенциала распараллеливания последовательного алгоритма (кода) и использование этого потенциала для синтеза будущей архитектуры (рис. 1,*b*). Достоинствами первого направления являются возможность улучшения уже разработанного параллельного кода и возможность сокращения времени вычислений на известной архитектуре. Достоинства второго направления состоят в обеспечении оценивания предельной возможности распараллеливания и, как следствие, предельной возможности ускорения вычислений в самом начале разработки параллельной системы, сокращения длины критического пути посредством эквивалентных преобразований поведенческого описания, выбора наиболее подходящего для распараллеливания алгоритма из множества альтернатив.

При проектировании архитектур СБИС сложное поведение представляется в виде ориентированного графа задач. Концепция критического пути на графе используется в работе [9] для решения проблемы оптимального размещения буферов, которая формулируется как проблема декомпозиции с использованием метода целочисленного линейного программирования.

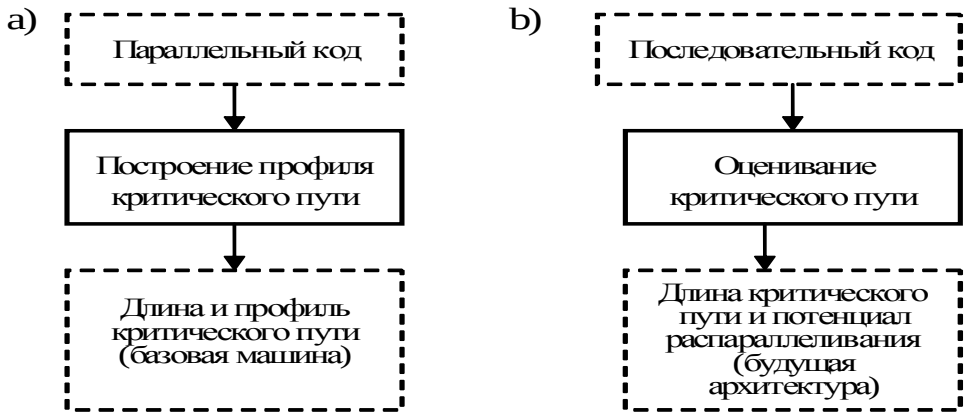

Рис. 2. Построение профиля критического пути (*a*) на параллельном коде и оценивание потенциала распараллеливания (*b*) на последовательном коде

В высокоуровневом синтезе статическая модель графа потока данных широко используется для решения задач планирования, размещения, связывания. В процессе планирования критический путь используется для установления достижимого итерационного периода. В работе [16] предложены некоторые преобразования статического графа с целью сокращения критического пути.

В работе [18] дополнительный код для анализа параллелизма встраивается в последовательный код программы событийного моделирования. Выполнение программы учитывает причинно-следственные ограничения, при этом соотношения между событиями представляются графом предшествования событий. В результате выполнения оценивается временная сложность параллельного моделирования, основанного на протоколе Чанди-Мисра.

В работе [12] анализ критического пути рассматривается как метод выявления того, какие задержки в передаче данных имеют место в Интернет. Построение профиля критического пути позволяет выявить долю, вносимую в общую задержку передачи данных такими элементами Интернет, как сервер, клиент, средства передачи пакетов, трафик и др.

В параллельных программах, передающих сообщения и разделяющих общую память [13], события взаимодействия и синхронизации формируют разнообразные пути при выполнении программы. Критический путь определяется как самая длинная последовательность взвешенных по времени событий. Параллельные вычисления представляются графом активности программы. Критический путь программы есть наидлиннейший путь на графе.

Профиль критического пути [13] — это список процедур параллельной программы и временной вклад каждой процедуры в общий критический путь. Это механизм, позволяющий выявить части параллельной программы, ограничивающие ее производительность, понять причину дисбаланса в нагрузке на вычислительные ресурсы, настроить код на ранних стадиях разработки системы и принципиально проверить возможность завершения программы в установленное время.

В основе методологии оценивания потенциального распараллеливания поведенческого описания лежат следующие главные принципы [11]:

критический путь определяется на потоке данных, поток последовательного управления исключается из рассмотрения;

длина критического пути и предельное распараллеливание определяются в терминах вычислительной сложности базовых операций языка, на котором представлено поведение. Параметры машины, на которой выполняется оценивание, не принимаются во внимание;

критический путь оценивается динамически на исполнительном графе потока данных, получаемом в результате частичного выполнения поведенческого описания на входных данных, типичных для распараллеливаемой системы;

потенциал распараллеливания оценивается как суммарная вычислительная сложность поведенческого описания, деленная на длину критического пути.

На рис. 3 приведен пример чисто последовательного С-кода, а на рис. 4 — исполненный граф потока данных. Жирным выделены вершины и дуги графа, лежащие на критическом пути. Вычислительная сложность С-кода (174) и длина критического пути (30) измерены в числе взвешенных операций. Предельно возможное ускорение описанных С-кодом процессов за счет распараллеливания составляет 5.8.

```
void main() {
      unsigned long n=21414;
      int m[5], k=0, i, j;
      for(i=0; i<5; i++) m[i]=0;
      while(n) {m[n%10]=1; n/=10;}
      for(j=0; j<5; j++) if(m[j]) k++;
}
```

Рис. 3. С-код для подсчета цифр 1, 2, 3, 4, 5 в целом числе *n*

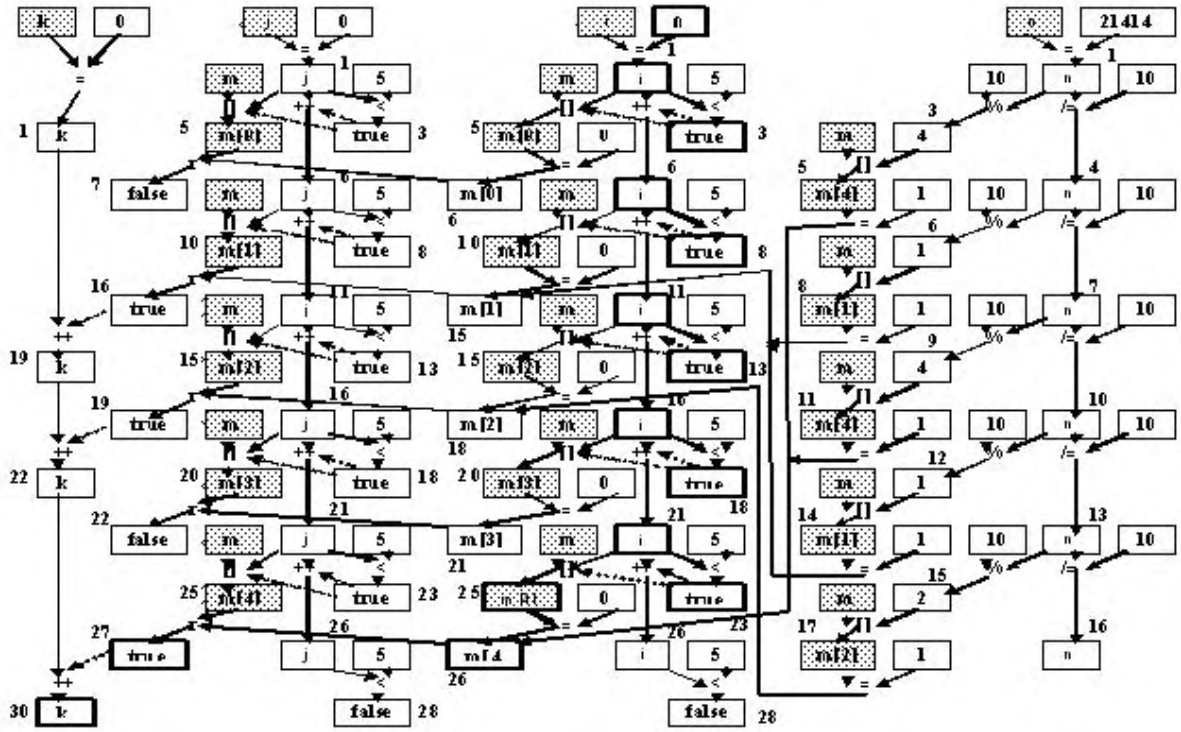

Рис. 4. Граф исполнения потока данных и критический путь для С-кода

Анализ критического пути составляет также эффективный базис для планирования вычислений. В работе [7] предлагается алгоритм планирования, основанный на перестройке критического пути. Работа [8] описывает метод планирования, выполняющий динамический анализ критического пути на не циклическом, не ветвящемся графе задач. Работы [3–5, 20, 21] определяют уровень параллелизма посредством взаимно параллельно выполняемых пар операторов. Метод минимизации критического пути, оцениваемого в виде клики максимального веса на графе последовательно/параллельного выполнения операторов, является наиболее перспективным методом оптимизации сетевых параллельных планов.

Стратегии оптимального синтеза последовательно-параллельных планов

Среди стратегий последовательно-параллельного планирования [6, 7, 23], минимизирующих время выполнения при неограниченном объеме вычислительных ресурсов, наиболее известными являются "как можно раньше" (ASAP — as soon as possible), "как можно позже" (ALAP — as late as possible). В стратегии ASAP (ALAP) оператор включается

в очередной шаг управления сразу, как только все его предшественники (последователи) включены в предыдущие (последующие) шаги. Примеры планов, синтезированных этими стратегиями, выполняемых на 4 шагах, приведены на рис. 5,*a* (ASAP) и 5,*b* (ALAP). Стратегии спискового планирования (list scheduling) являются эвристическими, строятся на базе стратегий ASAP и ALAP, используют список операций, которые могут быть включены в очередной шаг, и минимизируют время выполнения плана при наличии ограничений на ресурсы на каждом шаге управления. На рис. 5,*c* приведен план, синтезированный расширенной списковой стратегией ASAP в расчете на наличие 1 устройства сложения и 1 устройства умножения и выполняемый на 5 шагах управления.

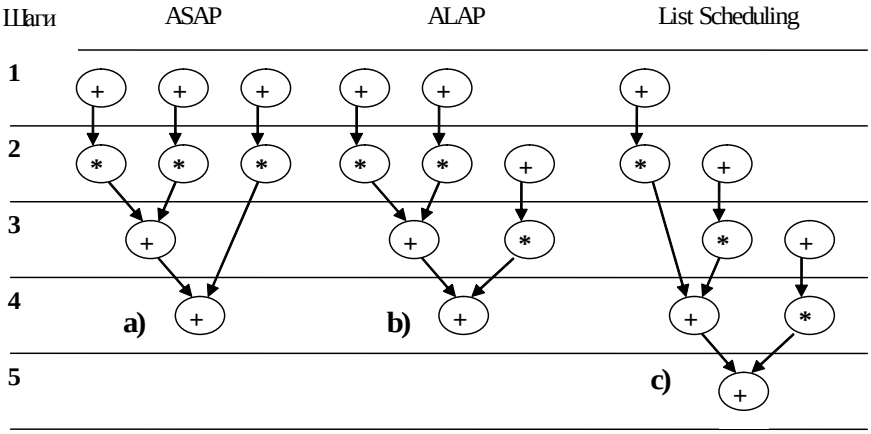

Рис.5. Последовательно-параллельные планы, синтезированные стратегиями ASAP (3 сумматора, 3 умножителя), ALAP (2 сумматора, 2 умножителя), List Scheduling (1 сумматор, 1 умножитель)

Точные стратегии планирования [6] строятся на базе метода целочисленного линейного программирования. Пусть $N=\{1,\ldots n\}$ — множество номеров операторов поведенческого описания. Бинарное отношение H непосредственного предшествования операторов (если $(i,j)\in H$, то i предшествует j) строится в результате анализа информационных зависимостей. Минимальный шаг S_i выполнения оператора i определяется по плану, генерируемому ASAP, максимальный шаг L_i определяется по плану, генерируемому ALAP. Число используемых в плане функциональных узлов типа k есть b_k, а число шагов управления равно T. Стоимость функционального узла типа k определяется величиной s_k. Обозначим через x_{ij} двоичную переменную, принимающую значение 1, если оператор i выполняется на шаге j. Задача целочисленного линейного программирования для минимизации ресурсов.

Целевая функция:

$$\min \sum_{k=1}^{N_{FU}} (s_k \cdot b_k). \qquad (1)$$

Ограничения:

$$\left(\sum_{\substack{i\in N \\ fu(i)=k}} x_{ij} \right) - b_k \le 0 \quad \text{для } 1\le j\le T,\ \ 1\le k\le N_{FU}, \qquad (2)$$

$$\sum_{j=S_i}^{L_i} x_{ij} = 1 \quad \text{для } 1\le i\le n, \qquad (3)$$

$$\sum_{r=S_i}^{L_i}(r \cdot x_{ir}) - \sum_{r=S_j}^{L_j}(r \cdot x_{jr}) \leq -1 \qquad \text{для } (i,j) \in N. \tag{4}$$

Ограничение (2) вытекает из того факта, что на каждом шаге число используемых функциональных узлов типа k не должно превышать b_k. Ограничение (3) выражает возможность выполнения любого оператора ровно на одном шаге управления. Ограничение (4) формализует то, что оператор j должен выполняться на шаге с номером, большим, чем шаг, на котором выполняется оператор i, предшествующий оператору j.

Еще одной эффективной стратегией оптимального синтеза последовательно-параллельных планов является метод свертывания графа распараллеленности операторов, предложенный в работе [23].

В рассмотренных стратегиях планирования один оператор размещался на одном шаге управления, а весь план обрабатывал один набор данных. Обобщение описанных стратегий [6] приводит к рассмотрению многошагового (Multicycling), цепочечного (Chaining) и конвейерного (Pipelining) планирования.

Эти виды планирования улучшают параметры синтезируемых планов.

Стратегии оптимального синтеза сетевых планов

В работах [3–5, 20, 21, 23] предложены и развиваются методы и стратегии синтеза асинхронных параллельных сетевых планов из последовательных алгоритмических поведенческих моделей с целью оптимизации асинхронных сетевых параллельных информационно-вычислительных систем. Методы охватывают три основных стадии перехода от произвольного последовательного алгоритмического описания к сетевому плану:

преобразование алгоритмического описания в модель с одним линейным участком (OBBM — one basic block model);

отображение OBBM в сетевой план с учетом лишь одного потока данных;

синтез и оптимизацию сетевого плана с учетом потока ресурсов.

Пример сетевого плана приведен на рис. 6. Это сеть, представленная ориентированным графом, построенным на вершинах-переменных и вершинах-операциях. Переменными могут быть как простые, так и произвольные сложные структуры данных, реализуемые файлами, а операциями могут быть как базовые операции алгоритмических языков, так и произвольные сложные макрооперации, реализуемые функциями, исполнительными модулями или пакетами программ. Первоначально граф строится на базе механизма квитирования. Дуги запроса и подтверждения, возможно, помеченные маркерами, вводятся в граф. В последующем часть дуг исключается из графа посредством построения антитранзитивного отношения. Вершины графа могут метиться условными булевыми выражениями, определяющими, выполняется операция или присваивание значения переменной или нет. Сетевой план описывает асинхронный алгоритм, функционирующий посредством сгорания вершин, приводящего к перемещению маркеров с входных дуг на выходные. Временная задержка на узле зависит от значения условного выражения, ассоциируемого с узлом. Граф выражает одновременно установленный алгоритмом поток данных, требуемый поток ресурсов, ограничения на время выполнения и объем имеющихся ресурсов.

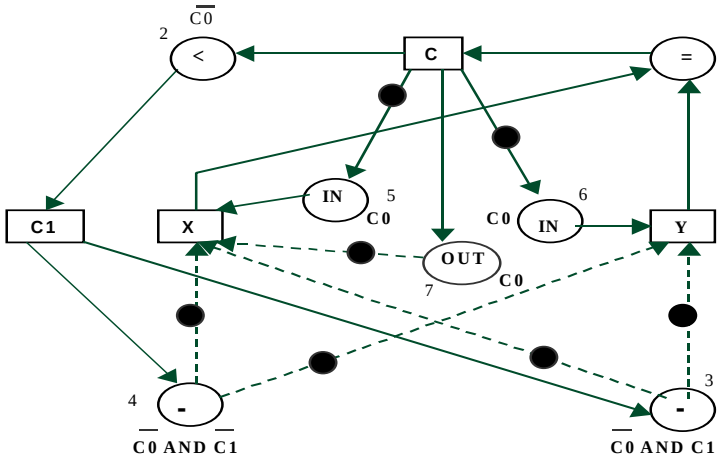

Рис. 6. Сетевой план алгоритма вычисления наибольшего общего делителя

Проблема оптимизационного синтеза графов сетевых планов является сложной комбинаторной проблемой структурного синтеза при параметрических ограничениях. Методы оптимизации планов предложены в работах [3, 5, 20]. Проблема синтеза разбита на задачи, включающие:

выявление и анализ зависимостей данных и операций;

генерацию отношений ортогональности на множестве тестовых сигналов и переменных;

генерацию множества пар взаимоисключающих операторов;

анализ операторов на совместимость и соседство;

генерацию отношения предшествования операторов;

оценивание параметров сетевого плана посредством взвешенных клик графа распараллеленности;

оптимизацию уровня параллелизма;

решение проблемы существования сети с учетом уровня параллелизма;

генерацию графа сетевого плана;

свертывание графа с целью разделения ресурсов.

Разработанные методы применимы для распараллеливания и планирования информационно-вычислительных процессов в программах, аппаратуре и смешанных аппаратно-программных системах.

Распараллеливание посредством преобразования поведенческих описаний

Методология распараллеливания и планирования посредством эквивалентных преобразований исходного поведенческого описания исследовалась и разрабатывалась в работах [22, 23]. Преимущества методологии проявляются в следующих принципиальных возможностях.

Одно и то же поведенческое описание может быть приведено к виду, адекватному той или иной параллельной архитектуре.

Предельные границы распараллеливания и предельные параметры реализации системы могут быть расширены.

Преобразованное поведенческое описание может быть более эффективным в отношении применимости методов, стратегий и алгоритмов решения оптимизационных задач автоматического распараллеливания и планирования.

При одних и тех же ограничениях на одни параметры системы можно получить более выгодные значения других параметров.

Следующие правила преобразования алгоритмического описания на языке представления аппаратуры (VHDL, Verilog) или программ (C/C++) приводят к "выгодной" для распараллеливания перестройке графов потока управления и потока данных:

реструктуризации, расщепление и преобразование операторов;

вынесению опережающих вычислений из управляющих структур;

слиянию операторов и выражений;

алгебраическим преобразованиям арифметических, логических и других типов выражений;

развертыванию циклов и др.

Во-первых, применение этих правил приводит к значительному сокращению длины критического пути и повышению возможного уровня распараллеливания. Во-вторых, расщепление всех управляющих последовательных структур делает исходное описание описанием "чистый поток данных" и значительно повышает эффективность стратегий планирования.

Экспериментальные результаты и заключение

Нами выполнены вычислительные эксперименты с целью оценки эффективности рассмотренных выше задач, методов, стратегий и алгоритмов распараллеливания и планирования. Оценивание вычислительной сложности, критического пути и потенциала распараллеливания С-кода криптографического пакета RSAREF [11] показало, что для отдельных составляющих пакета и типичных входных данных сложность меняется в пределах от $5{,}0\times10^6$ до $21{,}9\times10^9$, длина критического пути меняется в диапазоне 77×10^3–806×10^6, а достижимое ускорение за счет распараллеливания лежит в переделах от 27,12 до 136,93. Исследования С-кода WAVELET кодека выявили еще более впечатляющее возможное ускорение — до $1{,}1\times10^6$. Эксперименты по эквивалентному преобразованию С-кода с целью увеличения потенциала распараллеливания доказали, что сокращение критического пути в 3 раза и более является вполне достижимым.

Переход от последовательно-параллельной синхронной реализации системы к асинхронной сетевой реализации влечет двукратное ускорение вычислений [5] при изменяющемся времени выполнения операций. Были выполнены также эксперименты с целью выявления степени влияния эквивалентных преобразований VHDL-кода на число шагов управления, вводимых при решении задачи планирования. Число шагов сокращается до двух раз в процессе высокоуровневого синтеза [22].

Полученные результаты доказывают, что развиваемая нами парадигма "измерение параметров–преобразование поведения–оптимизация параллелизма" является наиболее эффективной в отношении распараллеливания и планирования информационно-вычислительных процессов. Итерационное применение парадигмы приводит к пошаговому улучшению параметров распараллеливаемой системы.

PARALLELIZATION AND SCHEDULING OF COMPUTATION AND INFORMATION PROCESSES

A.A. PRIHOZHY, M.V. SOLOMENNIK

Abstract

Results of research and development of the overall parallelization and scheduling technology for computation and information processes specified on algorithmic programming and hardware description languages are presented. The technology includes: (1) the measurement of computational complexity, critical path, parallelization potential and bound acceleration of a behavioral description (2) the optimal synthesis of a sequential-parallel schedule, (3) the optimal synthesis of a concurrent schedule, and (4) the equivalent transformation of the behavioral description in order to improve the parallel system parameters. The experimental results prove the applicability of the technology to real tasks of encryption and compression of data, audio and video information processing, information processing in computer and telecommunication networks and others.

Литература

1. IPDPS 2001, San Francisco, California, IEEE Press, 2001.
2. PARELEC 2002, IEEE Computer Society Press, 2002, 448 p.
3. *Prihozhy A.* // IEEE Design & Test of Computers. Spring, 1996. P. 24–33.
4. *Prihozhy A.* // Proc. DATE 98, IEEE CS Press. CA. 1998. P. 649–651.
5. *Prihozhy A., Solomennik M.* // Proc. FDL'2001, France, 2001. ECSI. P. 237–243.
6. *Hwang T., Lee J., Hsu Y.* // IEEE Transactions on CAD, 1991, Vol.10, No.4, P. 479–495.
7. *Kobayashi S., Sagi S.* // ISS, 2000. Vol. J81-D-I, No.2. P. 187–194.
8. *Kwong Y.-K., Ahmad I.* // IEEE Transactions on Parallel and Distributed Systems, 1996, Vol. 7, No. 5. P. 506–521.
9. *Chang P., Lee C.S.* // IEEE Transactions on Computers, 1990, Vol. 39, No. 1. P. 34–46.
10. Fundamentals and Standards in Hardware Description Languages. / Ed. J.P. Mermet. Kluwer Academic Publishers, Norwell, Mass., 1993.
11. *Prihozhy A., Mattavelli M., Mlynek D.* // Integrated Circuit and System Design, LNCS 2799. Springer, 2003. P. 569–579.
12. *Barford P., Crovella M.* // IEEE/ACM Transactions on Networking. 2001. Vol. 9, No. 3. P. 238–248.
13. *Hollingsworth J.* // IEEE Transactions on Parallel and Distributed Systems. 1998. Vol. 9, No. 10. P. 1029–1040.
14. *Li Y.S., Malik S.* // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 1997. Vol. 16. P. 1477–1487.
15. *Liu L., Du D., Chen H.-C.* // IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems. 1994.Vol. 13. No. 7. P. 909–919.
16. *Lucke L., Parhi K.* // IEEE Transactions on Computer Aided Design. 1993.Vol. 12, No. 7. P. 1063-1068.
17. *Pushner P., Koza C.* // Journal of Real-Time Systems. 1989. Vol. 1, Sept. P. 160–176.
18. *Wong Y.-C., Hwang S.-Y., Lin Y.* // IEEE Transactions on Parallel and Distributed Systems. 1995. Vol. 6, No. 6. P. 628–638.
19. *Henkel J., Ernst R.* // IEEE Transactions on VLSI Systems. 2001. Vol. 9, No 2. P. 273–289.
20. *Prihozhy A., Merdjani R., Iskandar F.* // Proc. PARELEC 2000, Canada, IEEE CS Press, CA, 2000. P.24-28.
21. *Prihozhy A., Mlynek D., Solomennik M., Mattavelli M.* // IEEE CS Press. 2002. P. 211–216.
22. *Prihozhy A.* // System-on-Chip Methodologies and Design Languages. Kluwer Academic Publishers, 2001. P. 135–146.
23. *Прихожий А.А., Мищенко В.А.* Теория эквивалентных преобразований алгоритмов в САПР СБИС, 1991, С. 120–151.