УДК 004.031.6

Д.И. Лазоренко

Институт проблем моделирования в энергетике им. Г.Е. Пухова НАН Украины, Киев

ОБЪЕДИНЕНИЕ МНОГОМЕРНЫХ ЦИКЛОВ ПРИ ПРОЕКТИРОВАНИИ ВСТРОЕННЫХ СИСТЕМ СО СНИЖЕННЫМ ЭНЕРГОПОТРЕБЛЕНИЕМ

Предлагается метод объединения многомерных циклов, который может использоваться с автоматизированных системах проектирования встроенных систем со сниженным энергопотреблением. Предлагаемый алгоритм позволяет одновременно объединять многомерные циклы со связями по данным и по выходу. В результате объединения циклов, выполненного согласно предлагаемому методу, можно существенно сократить количество операций обращения к основному запоминающему устройству, и уменьшить её размер.

Ключевые слова: *преобразования циклов, энергопотребление, встроенная система.*

Введение

В последнее время энергопотребление стало важным параметром в процессе проектирования электронных систем. Существует несколько причин для снижения энергопотребления:

- увеличение времени работы устройства от автономного источника энергии;
- уменьшение стоимости охлаждения устройства, а, следовательно, цены конечного продукта;
- увеличения надёжности проектируемого устройства;
- уменьшение эффекта электромиграции и электромагнитного излучения;
- упрощение разводки шин питания микросхем.

Основной вклад (до 80%) в энергопотребление вносит динамическое рассеяние энергии, которое происходит из-за зарядки/разрядки узлов схемы [1 – 3].

В настоящее время проектирование цифровой системы начинается с создания текста её описания на языках высокого уровня, например, C/C++, SystemC, VHDL, Verilog.

Решения, влияющие на энергопотребление проектируемого устройства, могут быть приняты на любом этапе процесса проектирования. Наибольший потенциальный выигрыш возможен в самом его начале, поэтому требуется соответствующим образом преобразовать исходный текст описания цифровой системы [4].

Встроенные системы предназначены для решения небольшого набора предопределённых задач. Знание алгоритмов этих задач позволяет провести оптимизацию создаваемых устройств по требуемому параметру, например, энергопотреблению.

Память и энергопотребление

Современные цифровые приложения обрабатывают большие объемы данных по сложным алгоритмам. В данное время схемы памяти могут занимать от 50 до 80 % площади полупроводникового кристалла. В будущем данная величина будет только увеличиваться. Известно также, что схемам памяти присущи большие паразитные токи утечки [5].

Следовательно, в процессе проектирования современных цифровых приложений необходимо уменьшать объем памяти и количество операций обращения к ней.

В тексте исходного описания цифровой системы циклы «for» представляют именно ту часть описания, которая ответственна за использование массивов, а, значит, определяет количество обращений к памяти и её объём [6 – 8]. Операция объединения циклов позволяет уменьшить количество обращений к памяти и её объём [6, 7].

Объединениемногомерныхциклов

Алгоритм объединения многомерных циклов был разработан на основе алгоритма объединения одномерных циклов [9].

В предлагаемом алгоритме каждому вычислению массива в исходном тексте описания ставится в соответствие вершина графа. Связи между циклами представлены ребрами графа. Ребра, обозначенные буквой «f», соответствуют связям по данным, а обозначенные буквой «o» – связям по выходу. Каждой вершине и «f»-ребрам ставится в соответствие набор весов, количество весов в наборе соответствует размерности вычисляемых массивов. Каждый отдельный вес в наборе отвечает определённой итерационной переменной (рис. 1).

Веса вершин в исходном редуцированном графе зависимостей равны 0.

Веса рёбер вычисляются следующим образом:

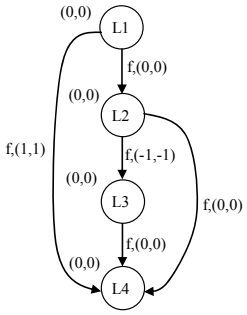
– пусть элементы массива «b» вычисляются при помощи элементов массива $a[i_1-d_1,i_2-d_2,\ldots,i_n-d_n]$;

– вес ребра, соответствующий i_k итерационной переменной, вычисляем, как разница k-х значений индексных выражений для элементов массива «a» при вычислении массива «a» и «b»;

– набор весов данного ребра будет $(d_1,d_2,\ldots,d_n)$.

Аналогично случаю с одномерными циклами, необходимо трансформировать граф таким образом, чтобы в нём не осталось «f»-рёбер с отрицательными значениями весов. Для вершин и «f»-рёбер изменение каждого веса в наборе, соответствующего одной из итерационных переменных, производится точно так же, как и в случае с одномерными циклами.

```
for (int i = 0; i < N; i++)
 for (int j = 0; j < N; j++)
  a1[i][j] = f1(i,j);
...
for (int i = 0; i < N; i++)
 for (int j = 0; j < N; j++)
  a2[i][j] = f2(a1[i][j]);
...
for (int i = 0; i < N - 1; i++)
 for (int j = 0; j < N - 1; j++)
  a3[i][j] = f3(a2[i+1][j+1]);
...
for (int i = 1; i < N; i++)
 for (int j = 1; j < N; j++)
a4[i][j] =
f4(a1[i-1][j-1], a2[i][j],a3[i][j]);
```



А Б

Рис. 1. Исходный текст программы (А), редуцированный граф зависимостей (Б)

Вес, соответствующий итерационной переменной j, ребра L2→L3 был увеличен на 1 и стал равным 0. Как следствие, вес, соответствующий итерационной переменной j, ребра L2→L4 увеличился на 1 и стал равным 1. Вес, соответствующий итерационной переменной j, ребра L1→L2 уменьшился на 1 и стал равен -1. Вес, соответствующий итерационной переменной j, вершины L2 уменьшился на 1 и стал равен –1. Таким образом, если вес, соответствующий k-й итерационной переменной, какого-либо ребра увеличивается на определённую величину, то вес, соответствующий k-й итерационной переменной, вершины, из которой выходит данное ребро, уменьшается на данную величину. Вес, соответствующий k-й итерационной переменной, рёбер, входящих в эту вершину, также уменьшается на ту же величину, а вес, соответствующий k-й итерационной переменной, рёбер, выходящих из этой вершины, увеличивается на упомянутую величину (рис. 2).

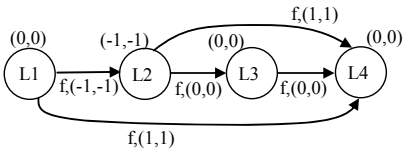


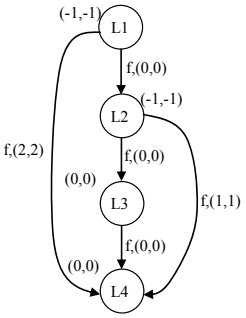
Рис. 2. Граф в процессе преобразований

Двигаясь к вершине L1 (началу графа), проводим аналогичные преобразования для всех весов в наборах (рис. 3).

Теперь рассмотрим объединение циклов, если между ними присутствует связь по выходу.

Пусть исходный текст программы содержит три цикла (рис. 4, А). Каждой вершине редуцированного графа (рис. 4, Б) присваивается набор нулевых весов. Веса «f»-рёбер вычисляются также, как это было сделано выше. Ребро с ярлыком «o» отображает наличие связи по выходу и не имеет веса. Между первым и третьим циклами существует связь по выходу.

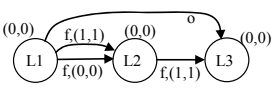
```
for (i = 2; i < N; i++)
 for (i = 2; j < N; j++) {
  a1[i][j] = f1(i,j);
  a2[i][j] = f2(a1[i][j]);
  a3[i-1][j-1] = f3(a2[i][j]);
  a4[i-1][j-1] =
f4(a1[i-2][j-2],a2[i-1][j-1],a3[i-1][j-1]);
}
```



А Б

Рис. 3. Текст объединённого цикла (А), преобразованный граф зависимостей (Б)

```
for (int i = 0; i < N; i++)
 for (int j = 0; j < N; j++)
  a[i][j] = f1(i,j);
...
for (int i = 1; i < N; i++)
 for (int j = 1; j < N; j++)
  b[i][j] = f2(a[i][j],a[i-1][j-1]);
...
for (int i = 0; i < N - 1; i++)
 for (int j = 0; j < N - 1; j++)
  a[i+1][j+1] = f3(b[i][j]);
```

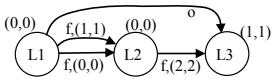


А Б

Рис. 4. Исходный текст программы (А), редуцированный граф зависимостей (Б)

В исходном виде данные три цикла объединить невозможно. Например, значение элемента a[1,1] необходимо для вычисления b[2,2], но до этого оно уже перезаписывается во время выполнения третьего цикла.

Чтобы сделать объединение циклов возможным, необходимо перезаписывать значение a[i,j] уже после того, как оно было использовано для вычисления b[i+1,j+1]. Для этого требуется трансформировать граф следующим образом. Увеличиваем каждый k-й вес вершины L3, на которой заканчивается «o»-ребро, на величину, равную максимальному значению из всех k-х весов всех «f»-рёбер, исходящих из вершины L1, из которой исходит та же «o»-ребро. Например, из вершины L1 исходят две «f»-ребра, первый вес из набора одной из них равен 1, для другой – эта величина равна 0. Максимальное значение из приведённых двух равно 1, поэтому первый из набора весов вершины L3 должен увеличиться на 1. При этом веса всех «f»-рёбер, входящих в L3, должны увеличиться на ту же величину, на которую увеличились соответствующие веса вершины L3, веса же всех исходящих из данной вершины «f»-дуг должны уменьшиться на упомянутую величину (рис. 5).




```
for (int i = 2; i < N; i++)
 for (int j = 2; j < N; j++){
  a[i][j] = f1(i,j);
  b[i][j] = f2(a[i][j],a[i-1][j-1]);
  a[i-1][j-1] = f3(b[i-2][j-2]);
 }
```

А Б

Рис. 5. Редуцированный граф зависимостей (А), исходный текст программы (Б)

Для обобщения ниже приводятся алгоритмы построения и преобразования графического отображения циклов.

Алгоритм построения графа исходного текста программы:

1. Каждому вычисляемому в циклах массиву ставится в соответствие вершина графа с начальным набором нулевых весов.

2. Вершинами соединяются направленными дугами. Дуги исходят из вершин, соответствующих тем массивам, которые вычисляются первыми в исходном тексте программы.

2.1. Рёбра, соответствующие связи по выходу, не имеют веса и обозначаются ярлыком «o».

2.2. Рёбра, соответствующие связи по данным, обозначаются ярлыком «f». Каждому f-ребру ставится в соответствие набор весов. Размерность набора равна размерности массивов. K-й вес в наборе соответствует k-й итерационной переменной. Начальное значение k-го веса в наборе определяется как разница между индексными выражениями для k-й индексной переменной массива, который вычисляется в левой части выражения, соответствующего вершине, из которой выходит ребро, и правой части выражения, использующего этот массив и соответствующего вершине, на которой ребро заканчивается.

Алгоритм преобразования графа:

1. Если существуют «f»-рёбра с отрицательными значениями весов в наборе, тогда, двигаясь от такого ребра к началу графа, сделать все значения весов «f»-рёбер неотрицательными следующим образом: пусть k-й вес имеет отрицательное значение, увеличить k-й вес ребра до нуля на необходимую величину, затем уменьшить k-й вес вершины, из которой вышло данное ребро, на упомянутую величину; все другие «f»-рёбра, исходящие из данной вершины, увеличивают свой k-й вес на указанную величину, а «f»-рёбра входящие в эту вершину, наоборот, на данную величину свой k-й вес уменьшают.

2. Если есть циклы со связями по выходу, то k-й вес вершины, в которую входит «o»-ребро, должен быть увеличен на величину, равную максимальному значению из всех k-х весов «f»-рёбер, выходящих из вершины исходной для данной «o»-ребра. При этом веса всех f-рёбер, входящих в вершину, в которую входит «o»-ребро, должны увеличиться на ту же величину, на которую увеличились соответствующие веса этой вершины, веса же всех исходящих из данной вершины f-рёбер должны уменьшиться на упомянутую величину.

Выводыя

Предлагаемый алгоритм позволяет одновременно объединять многомерные циклы со связями по данным и по выходу. В результате объединения циклов, выполненного согласно предлагаемому методу, можно существенно сократить количество операций обращения к основному запоминающему устройству, и уменьшить её размер. Данный алгоритм позволяет объединять больше циклов, чем метод, описанный в [7].

Списокяитературыя

1. Veendrick H.J.M. Deep-Submicron CMOS ICs. From Basics to ASICs / H.J.M. Veendrick. – Kluwer academic publishers, 2000. – 539 p.

2. Poppen F. Low Power Design Guide / F. Poppen. – OFFIS Research Institute [Электронный ресурс]. – Режим доступа: http://www.offis.de.

3. Kim H.S. Effect of compiler optimizations on memory energy / H.S. Kim, M.J. Irwin, N. Vijaykrishnan, M. Kandemir // 2000 IEEE Workshop on Signal Processing Systems. SiPS 2000. – 2000. – P. 663-672.

4. Sproch J. High Level Power Analysis and Optimization. Tutorial / J. Sproch // 1997 International Symposium on Low Power Electronics and Design. – 1997.

5. International Technology Roadmap for Semiconductors [Электронный ресурс]. – Режим доступа: http://public.itrs.net/

6. Fraboulet A. Loop Alignment for Memory Accesses Optimization / A. Fraboulet, G. Huard, A. Mignotte // Twelfth International Symposium on System Synthesis. Proceedings (ISSS'99). IEEE Computer Society Press. – 1999. – P. 71-77.

7. Fraboulet A. Loop fusion for memory space optimization / A. Fraboulet, K. Kodary, A. Mignotte // The 14th International Symposium on System Synthesis. Proceedings 2001. – 2001. – P. 95-100.

8. Catthoor F. Global communication and memory optimizing transformations for low power signal processing systems / F. Catthoor, F. Franssen, S. Wuytack, L. Nachtergaele, H. De Man // Workshop on VLSI Signal Processing, VII. – 1994. – P. 178-187.

9. Лазоренко Д.И. Алгоритм объединения одномерных циклов исходного текста описания цифровых систем с целью снижения их энергопотребления / Д.И. Лазоренко // Системи обробки інформації. – Х.: ХУПС, 2007. – Вип. 8(66). – С. 45-49.

Рецензент: д-р техн. наук, проф**.** А.Н. Давиденко, Институт проблем моделирования в энергетике НАН Украины, Киев.

Поступила в редколлегию 3.06.2009

ОБ'ЄДНАННЯ БАГАТОВИМІРНИХ ЦИКЛІВ ПРИ ПРОЕКТУВАННІ ВБУДОВАНИХ СИСТЕМ З ПОНИЖЕНИМ ЕНЕРГОСПОЖИВАННЯМ

Д.І. Лазоренко

У даній статті пропонується метод об'єднання багатовимірних циклів, який може використовуватися з автоматизованих системах проектування вбудованих систем з пониженим енергоспоживанням. Пропонований алгоритм дозволяє одночасно об'єднувати багатовимірні цикли із зв'язками по даним і по виходу. В результаті об'єднання циклів, виконаного згідно пропонованому методу, можна істотно скоротити кількість операцій звернення до основного пристрою, що запам'ятовує, і зменшити її розмір.

Ключові слова: *перетворення циклів, енергоспоживання, вбудована система.*

MULTIDIMENSIONAL LOOP FUSION FOR LOW-POWER EMBEDDED SYSTEMS DESIGN

D.I. Lazorenko

A merge of multidimensional cycles technique is offered in this article, which can be used with the automated systems of planning of the embedded systems with the reduced energy consumption. The offered algorithm allows simultaneously to unite multidimensional cycles with connections to on to information and on an output. As a result of uniting cycles, executed in obedience to the offered method, it is possible substantially to shorten the amount of operations of address to basic data storages, and to decrease its size.

Keywords: *transformations of cycles, energy consumption, embedded system.*