УДК 004.383

А.М. Сергиенко, В.П. Симоненко, доктора техн. наук
Национальный технический университет Украины
«Киевский политехнический ин-т»
(Украина, 03056, Киев, пр-т Победы, 37,
тел. 4549337, e-mail: aser@comsys.ntu-kpi.kiev.ua)

Методы оптимизации графов синхронных потоков данных

Выполнено сравнения методов ресинхронизации графов синхронных потоков данных (ГСПД), отображаемых в конвейерные вычислительные устройства (ВУ). Методы основаны на минимизации задержек прохождения сигналов между регистрами. Предложен метод ресинхронизации пространственного ГСПД, обеспечивающий максимальное отношение производительность—стоимость в полученном ВУ.

Виконано порівняння методів ресинхронізації графів синхронних потоків даних (ГСПД), які відображаються в конвеєрні обчислювальні пристрої (ОП). Методи базовані на мінімізації затримок проходження сигналів між регістрами. Запропоновано метод ресинхронізації просторового ГСПД, який забезпечує максимальне відношення продуктивність—вартість в отриманому ОП.

К л ю ч е в ы е с л о в а: граф потоков данных, цифровая обработка сигналов.

Современные высокопроизводительные средства вычислительной техники работают с высокими тактовыми частотами, которые достигают нескольких гигагерц вследствие конвейерной организации обработки и передачи данных. Существующие в настоящее время методы проектирования и оптимизации конвейерных вычислительных устройств (ВУ) сводятся к структурному синтезу ВУ, описанию его на уровне регистровых передач и дальнейшему преобразованию до уровня вентилей. Основой многих методов структурного синтеза конвейерных ВУ является представление алгоритма в виде графа синхронных потоков данных (ГСПД) и его трансформация [1]. Такие методы оптимизации ГСПД, как ресинхронизация, сворачивание, разворачивание и конвейеризация ГСПД, существуют более двух десятилетий и часто используются в микроэлектронике [2], при программировании систем реального времени [3], при проектировании устройств цифровой обработки сигналов [4].

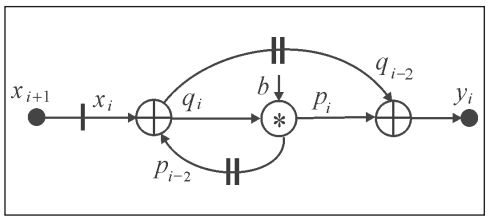


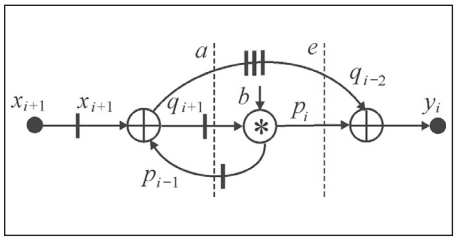
Рис. 1. Граф синхронизации потоков данных: ⊕ — вершина оператора сложения; ⊛ — вершина умножения на коэффициент; ❙ — метка задержки на один цикл данного, передаваемого по дуге

Методы оптимизации ГСПД. При высокоуровневом или системном синтезе ВУ циклически повторяемый алгоритм, такой как алгоритм цифровой обработки сигналов, обычно представляют в виде ГСПД. В нем вершины-акторы соответствуют операторам алгоритма, а дуги — передачам переменных между акторами с задержкой на заданное число циклов. При этом каждым актором в течение одного цикла генерируется и используется число переменных, неизменное от цикла к циклу [5, 6].

Различают однородные ГСПД (homoheneous SDF (synchronous data flow)) и неоднородные ГСПД (multirate SDF). В последних число переменных, использованных и сгенерированных каждой вершиной в течение одного цикла, может быть больше одной, вследствие чего они имеют более компактную форму задания алгоритма [7]. Для упрощения анализа и синтеза на его основе ВУ неоднородный ГСПД преобразуют в эквивалентный однородный [6, 8, 9]. Далее будем рассматривать только однородные ГСПД.

Фазовому фильтру с амплитудо-частотной характеристикой $H(z) = b + z^{-2}/(1 + b\,z^{-2})$ соответствуют разностные уравнения $p_i = b\,q_i$, $q_i = x_i - p_{i-2}$, $y_i = q_{i-2} + p_i$ и ГСПД, представленный на рис. 1. Алгоритм, заданный ГСПД, следующий. При появлении на входе первой вершины сложения операндов x_i и p_{i-2} сразу же или с заданной задержкой d на ее выходах появляется результат q_i, который немедленно поступает на вход смежной вершины или с задержкой на число циклов, равной числу меток, которыми нагружена дуга, входящая в эту вершину. Аналогичные шаги выполняются на остальных вершинах. Цикл алгоритма является элементарным вычислительным процессом в ГСПД, который оканчивается тогда, когда состояния выходов всех вершин принимают новые значения.

Особое свойство ГСПД состоит в том, что он изоморфен графу структуры некоторого конвейерного вычислителя, выполняющего заданный алгоритм. При этом его вершины соответствуют вычислительным ресурсам — сумматорам, умножителям, дуги — линиям связи, а метки, которыми они нагружены, — регистрам, на которые поступает синхросигнал от одного источника. Следовательно, ГСПД — это направленный граф $G = (V, E)$, представляющий вычислительную структуру, где вершина $v \in V$ отвечает комбинационной схеме, имеющей задержку d еди-



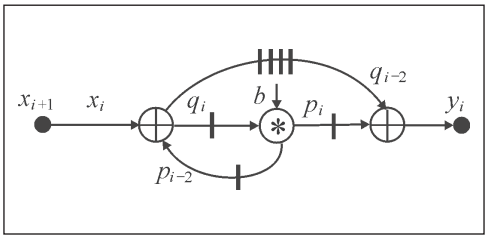


Рис. 2. Пример элементарной ресинхронизации

Рис. 3. ГСПД с минимальным тактовым интервалом $T_C = \max(d_A, d_M)$

ниц времени. Дуга $e \in E$ графа соответствует линии связи, нагруженной метками, число которых равно числу регистров $w[e]$, задерживающих переменную на w циклов или тактов синхросигнала.

Период синхросигнала определяет длительность выполнения цикла алгоритма. Минимальная длительность этого цикла равна максимальной задержке прохождения сигнала от выхода одного регистра или входа устройства до входа другого регистра или выхода устройства, т.е. критическому пути, проходящему через смежные вершины с задержками d, соединенные дугами e, для которых $w[e]=0$. Для ГСПД, представленного на рис. 1, эта длительность составляет $T_C = 2d_A + d_M$, где d_A — задержка сложения, d_M — задержка умножения. Следует заметить, что при таком единичном отображении ГСПД в структуру ВУ длительность выполнения цикла алгоритма T_A совпадает с длительностью тактового интервала, т.е. $T_A = T_C$, что при других отображениях не соблюдается.

Ресинхронизация ГСПД — это такое изменение меток в дугах, т.е. положения регистров, которое не влияет на результаты выполнения алгоритма. Обычно оно реализуется как последовательность элементарных ресинхронизаций, каждая из которых состоит в перенесении одной группы меток с входных дуг некоторой вершины v на ее выходные дуги.

На рис. 2 представлен пример элементарной ресинхронизации — переноса группы меток через вершину, где вычисляется значение q_i, в результате чего минимальная длительность цикла уменьшилась до $T_C = 2d_A + d_M$. Поэтому ресинхронизация является распространенным приемом минимизации тактового интервала конвейерных вычислительных структур. Целью ресинхронизации также является минимизация общего числа регистров, часто используемая при программной реализации ГСПД.

После ресинхронизации посредством перемещения $r[v]$ регистров из дуг, входящих в вершину v, в дуги, выходящие из этой вершины, новое значение числа регистров $w'[u,v]$ для дуги $e=(u,v)$ рассчитываем по формуле

$$w'[u,v] = w[u,v] + r[v] - r[u] \geq 0. \qquad (1)$$

При этом тот факт, что $w \geq 0$ для всех дуг ГСПД, является естественным условием корректности ресинхронизации [10].

Если между вершинами u и v находятся один или несколько маршрутов, в дугах которых стоят регистры, то следует определять параметр $W[u, v]$, равный минимальному числу регистров в этих маршрутах. Тогда из подграфа, которому принадлежат эти маршруты, можно вынести наружу не более, чем $W[u, v]$ регистров, чтобы не нарушить условие (1). Для решения задачи оптимизации необходимо также определить задержку $D[u, v]$ от вершины u до вершины v, которая равна максимальной задержке, определенной по всем маршрутам, проходящим между этими вершинами, для которых $W[u, v]$ — минимально.

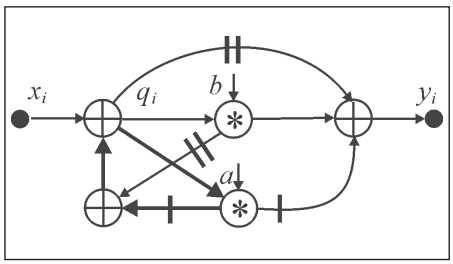
В графе G не должно быть цикла, во всех дугах которого отсутствуют регистры, т.е. $W[u, u] = 0$. Другими словами, если вершины u и v, принадлежащие подграфу с параметром $W[u, v]$, соединены дугой, замыкающей кольцевой маршрут, то число регистров в таком маршруте не может быть меньше единицы, и для такого графа справедливо следующее:

$$W[u, v] + r[v] - r[u] \geq 1. \qquad (2)$$

Из параметров $W[u, v]$ и $D[u, v]$ формируются соответственно матрицы W и D. В работе [4] предложен алгоритм поиска оптимальной ресинхронизации, обеспечивающей минимальный период T_C. Он основан на том, что из матриц W и D и условий (1) и (2) составляется и решается задача целочисленного линейного программирования с использованием алгоритма Бэллмана—Форда для поиска кратчайшего пути [2]. При этом фактически выполняется проверка осуществимости вычислительной структуры для заданного значения T_C, которое итерационно уточняется. Сложность выполнения такого алгоритма оценивается как $o(|V|^3 \log |V|)$.

Другой известный алгоритм [2] основан на выделении в графе G максимального ациклического подграфа, его ресинхронизации посредством увеличения параметра r в дугах, принадлежащих критическому пути, до получения требуемого периода T_C. В конце итерации алгоритма проверяются условия (1) и (2) для всего графа G. Сложность этого алгоритма оценивается как $o(|V||E|\log |V|)$.

Известны способы упрощенного выполнения ресинхронизации. Наиболее известен способ ресинхронизации по сечению ГСПД, согласно которому в графе выделяются дуги, разрезающие его на две несвязанные части. Тогда регистры, находящиеся в дугах, направленных в другую часть, могут быть изъяты, а в дуги, ведущие в обратном направлении, — добавлены. Такой разрез показан на рис. 2 штриховой линией a. Результатом ресинхронизации в этом случае будет граф, представленный на рис. 1.



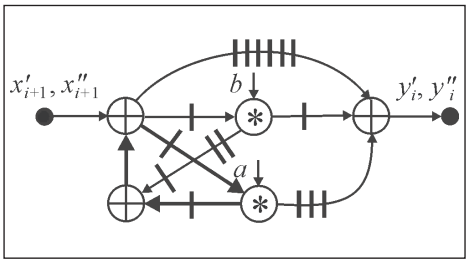


Рис. 4. ГСПД для фазового фильтра с функцией (3)

Рис. 5. ГСПД, представленный на рис. 4, после ресинхронизации и замедления при $c = 2$

Строго говоря, при ресинхронизации допускается добавление регистров в дуги только по правилу (1), иначе — изменяется алгоритм. Однако в большинстве случаев допускается увеличение латентной задержки выполнения алгоритма и вставляются дополнительные регистры на входах или выходах ГСПД. После ресинхронизации такого модифицированного ГСПД получается конвейеризованная схема с пониженным значением T_C. Такой метод называют конвейеризацией.

Простой способ конвейеризации — разрезание ГСПД так, чтобы разрез проходил через одинаково направленные дуги, как, например, разрез, показанный на рис. 2 штриховой линией *e*. В каждую из дуг разреза вставляется по регистру (рис. 3), вследствие чего для данного ГСПД минимальный тактовый интервал уменьшается до $T_C = \max(d_A, d_M)$.

На рис. 4 представлен ГСПД, соответствующий фазовому фильтру с амплитудо-частотной характеристикой

$$H(z) = \frac{b + a\,z^{-1} + z^{-2}}{1 + a\,z^{-1} + b\,z^{-2}}. \qquad (3)$$

Жирными стрелками выделен цикл, определяющий критический путь длительностью $T_C = 2d_A + d_M$. Для этого цикла $W[u,u] = 1$ и эта величина не может быть увеличена с помощью ресинхронизации. Таким образом, число регистров в цикле $W[u,u]$ определяет возможности ресинхронизации. Поэтому следует выбирать такие алгоритмы, для которых это число максимально [9]. Например, для увеличения $W[u,u]$ часто ГСПД трансформируют посредством увеличения числа регистров во всех дугах в c раз. Полученный ГСПД выполняет c исходных алгоритмов параллельно, но с периодом в c тактов. Вместе с тем, в таком ГСПД можно до c раз снизить длительность тактового интервала. Этот метод называют ресинхронизацией с c-кратным замедлением. После замедления при $c = 2$ и ресинхронизации получаем ГСПД, представленный на рис. 5. В результате дли-

тельность тактового интервала уменьшилась до $T_C = \max(2d_A, d_M)$, но длительность цикла алгоритма увеличилась до $T_A = cT_C = \max(4d_A, 2d_M)$.

В работе [11] предложен метод, обеспечивающий теоретически минимальный период вычислений ГСПД. Метод состоит в ресинхронизации исходного ГСПД с получением минимального числа регистров задержки, раскручивании оптимизированного ГСПД и минимизации задержек логических функций в критическом пути. При этом раскручивание ГСПД означает такое его преобразование, когда в течение одного цикла нового ГСПД вычисляется c циклов исходного ГСПД. Такое преобразование эквивалентно операции раскручивания циклов, применяемого в программировании [12]. Но при этом длина критического пути может увеличиться в c раз, так как число $W[u,u]$ остается прежним.

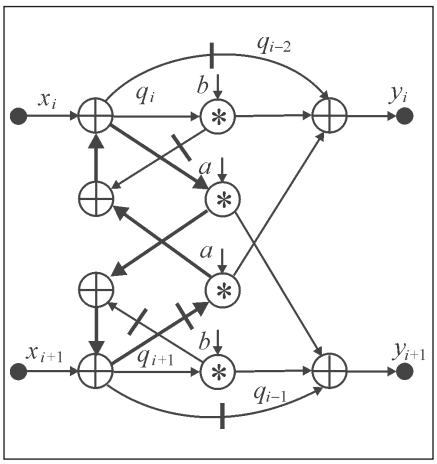


Рис. 6. ГСПД, представленный на рис. 5, после раскручивания в $c = 2$ раза

Пример ГСПД, изображенного на рис. 5, после раскручивания представлен на рис. 6. В маршруте критического пути этого ГСПД вычисляется выражение

$$q_{i+1} = x_{i+1} - b\,q_{i-1} - a\,(x_i - b\,q_{i-2} - a\,q_{i-1})\,. \qquad (4)$$

В результате оптимизации раскрученного ГСПД получаем структуру с увеличенными в c и более раз аппаратурными затратами, однако вычисляющую за один такт c циклов исходного ГСПД. При этом длительность одного такта увеличивается менее, чем в c раз вследствие логической оптимизации вычислительных модулей, находящихся в критическом пути. Такая оптимизация достигается в результате того, что для многих алгоритмов цепочку из n двухвходовых функциональных элементов, стоящую в критическом пути, удается заменить деревом, имеющим высоту $\log_2 n$. В этом случае длительность цикла алгоритма уменьшается до $T_A \approx T_C / c$. Например, выражение (4) можно представить в виде $q_{i+1} = x_{i+1} - ((bq_{i+1} - ax_i) + (pq_{i-2} + sq_{i-1}))$, где $p = ab$, $s = a^2$, и вычислить с помощью дерева сумматоров и умножителей с четырьмя уровнями, а не с шестью, как для выражения (4). Длительность тактового интервала для такого развернутого ГСПД составит $T_{\text{Ср}} = 3d_A + d_M$, а длительность цикла — $T_A = 1{,}5d_A + 0{,}5d_M$. При этом величина T_A оказывается меньше, чем для

исходного ГСПД или ГСПД после ресинхронизации с замедлением. Таким образом, минимально возможное значение периода вычислений T_A достигается при существенном увеличении аппаратных затрат. В данном примере число умножителей увеличено с двух до семи.

Рассмотрим также метод сворачивания ГСПД, позволяющий увеличить число регистров в замкнутых циклах. Он состоит в том, что ГСПД, имеющий множества вершин, каждое из которых содержит до c вершин одного типа, заменяется на ГСПД с теми же множествами вершин, но в каждом из них оставлено по одной вершине. При этом потоки данных направлены так, чтобы в новом ГСПД цикл алгоритма выполнялся за c тактов посредством разделения вершин во времени. В новом ГСПД число регистров задержки оказывается таким же, как и в исходном ГСПД.

В общем случае метод сводится к синтезу модели конвейерного вычислителя согласно алгоритму ГСПД. При этом выполняются этапы выбора ресурсов (множества вершин процессорных элементов), составления расписания срабатывания ресурсов и назначения операций, т.е. вершин ГСПД, на ресурсы. Часто стратегия назначения на ресурсы обусловлена тем, что во многих случаях несколько изоморфных подграфов исходного ГСПД являются изоморфными подграфу результирующего ГСПД. Для необходимого перенаправления потоков данных в граф вводятся искусственные вершины коммутаторов, имеющих до c входов. Если регистр должен хранить данное в течение нескольких тактов, то ему назначают метку номера такта записи.

Название метода объясняется тем, что он напоминает метод оптимизации программных циклов сворачиванием соседних однородных операций [13]. Метод ресинхронизации с замедлением является частным случаем метода сворачивания ГСПД. Рассмотрим его применение для построения цифрового фильтра (рис. 7), имеющего амплитудо-частотную характеристику

$$H(z) = H_1(z) + H_2(z)\, z^{-1}, \qquad (5)$$

где $H_1(z)$, $H_2(z)$ — характеристики, такие же, как (3). Как видно из рис.7, свернутый ГСПД обеспечивает в $c \approx 2$ раза меньшие аппаратные затраты и минимизированную до $T_{Cc} = \max(2d_A, d_M)$ длительность тактового интервала, так как на вычисление одного цикла алгоритма требуется два такта. Поскольку при сворачивании ГСПД длительность тактового интервала T_{Cc} уменьшается в c раз, можно достичь ситуации, когда ВУ, полученное сворачиванием, имеет период вычислений T_A, равный длительности тактового интервала T_C до сворачивания, т.е. $T_A = T_C = cT_{Cc}$. В этом случае полученное ВУ будет иметь существенно меньшие (до c раз) аппаратные затраты.

Из рассмотренных методов метод ресинхронизации является наиболее конструктивным. Он всегда позволяет достичь намеченной цели с

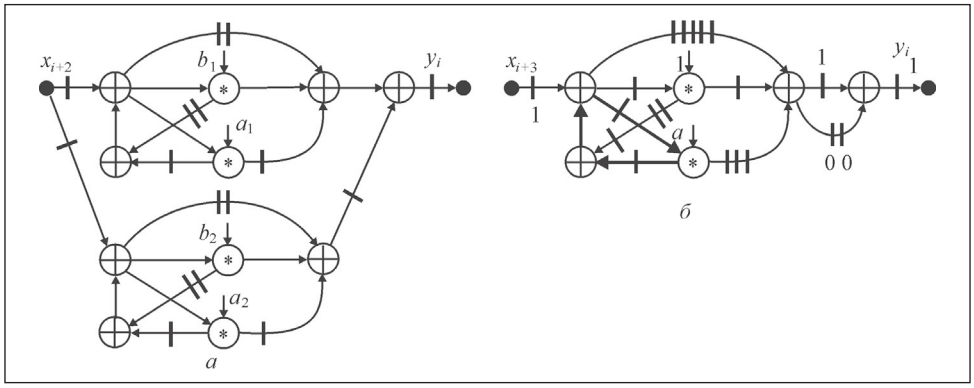


Рис. 7. ГСПД цифрового фильтра после конвейеризации (*а*) и ресинхронизации сворачиванием с периодом $c = 2$ (*б*): 0 или 1 — номер такта двухтактового цикла, в котором в этот регистр записывается данное

помощью соответствующего алгоритма оптимизации, сложность которого сравнительно невелика. Поэтому метод ресинхронизации успешно используется во многих САПР микроэлектроники. Рассмотрим недостатки этих методов.

Расписание выполнения ГСПД и синтез ВУ. В рассмотренных выше методах оптимизации проведена прямая аналогия между ГСПД и реальной конвейерной схемой. В действительности метка на дуге ГСПД означает, что данные в дуге должны быть задержаны на w циклов алгоритма в порядке их следования, т.е. по принципу FIFO. Замена этих меток на регистры, управляемые синхросигналом, позволяет простейшим способом получить одно из многих корректных расписаний выполнения алгоритма. Цикл выполнения такого расписания равен одному такту при единичном отображении ГСПД в структуру или c тактов в случае свернутого ГСПД, хотя другие возможные расписания могут продолжаться в течение иного числа тактов.

Поиск возможных расписаний выполнения ГСПД является одним из этапов задачи синтеза конвейерного ВУ, которая, согласно общепринятой методологии, решается путем выполнения, кроме этапа расписания, этапов выбора множества ресурсов, назначения операций на ресурсы, построения структуры ВУ и его блока управления, а также выбора ВУ, оптимального по заданному критерию.

Сложность многоэтапного синтеза ВУ состоит в том, что различные аспекты синтеза и этапы разработки ВУ существенно зависят один от другого. Каждый из этапов проектирования не может быть выполнен независимо так, чтобы не уменьшить возможности глобальной оптимиза-

ции ВУ. Например, минимизация аппаратных затрат при выборе ресурсов входит в противоречие с минимизацией длительности цикла алгоритма T_A при составлении расписания. Кроме того, если при единичном отображении алгоритма в структуру, как, например, при ресинхронизации ГСПД, сложность синтеза ВУ оценивается полиномиальной сложностью задачи ресинхронизации, то при отображении с замедлением в c раз, как при сворачивании ГСПД, задача синтеза становится *NP*-полной [13].

Таким образом, по всем рассмотренным методам синтезируется ограниченное множество ВУ, способных к исполнению алгоритма с периодом один или c тактов. При этом, если следовать методу сворачивания ГСПД, то фактически выполняются все этапы синтеза конвейерного ВУ, имеющего указанные выше недостатки.

Метод оптимизации пространственного ГСПД. Существенно упростить и улучшить процесс синтеза конвейерного ВУ можно, если этапы синтеза — выбор ресурсов, составление расписания и назначение на ресурсы — выполнять на одном комбинированном этапе.

Задание расписания для ГСПД означает назначение вершинам моментов срабатывания. Структуру ВУ можно получить путем гомоморфного преобразования ГСПД в граф структуры. Такое преобразование означает склеивание вершин, операторы которых выполняются в одном процессорном элементе (ПЭ) ВУ, но в разные моменты времени. Линии межпроцессорной связи при этом соответствуют дугам ГСПД после гомоморфного преобразования [14].

Гомоморфное преобразование графа выполняют, назначив вершинам тэги с номерами вершин ПЭ. Тогда в вершину ПЭ склеиваются вершины с одинаковыми номерами. Если рассматриваются не универсальные ПЭ, то в них должны быть отображены операторы, соответствующие типу данного ПЭ. Так, в блок умножения отображаются только операторы умножения. Поэтому вершины ГСПД должны иметь тэги, указывающие тип оператора. Следовательно, вершина ГСПД должна иметь тэг с такими параметрами, как момент выполнения оператора, его тип и номер ПЭ, где он будет выполняться.

Отображение алгоритма заключается в назначении тегам определенных значений и построении структуры ВУ с вычислением критерия ее оптимальности. Такой набор тэгов означает одно структурное решение ВУ. Тогда оптимизация ВУ заключается в построении множества таких наборов и выборе наиболее предпочтительного набора по критерию оптимальности.

Тэги вершин представимы многомерными векторами. Если вершина имеет тэг размерности n, то считается, что она находится в n-мерном

пространстве. Целесообразно представить координаты векторов — моменты срабатывания, номера ПЭ, тип операторов — целыми числами. Тогда ГСПД с такими вершинами будет представлен в n-мерном целочисленном пространстве $\mathbb{Z}^n$. При этом момент срабатывания задается как номер такта.

В [15] описан метод синтеза ВУ, в котором ГСПД представлен в трехмерном целочисленном пространстве в виде конфигурации алгоритма (КА) $\mathsf{K}_G = (K, D, A)$, где K — матрица векторов-вершин $\mathbf{K}_i$, соответствующих операторам алгоритма; D — матрица векторов-дуг $\mathbf{D}_j$, соответствующих непосредственным информационным связям между операторами; A — матрица инцидентности ГСПД. В векторе-вершине $\mathbf{K}_i = (k_i, s_i, t_i)^T$ координаты k_i, s_i, t_i соответствуют типу оператора, номеру процессорного элемента, где выполняется данный оператор, и такту, в котором записывается в регистр результат этого оператора. Таким образом, векторы $\mathbf{K}_i$ представляют собой тэги, кодирующие свойства вершин ГСПД, и такой граф называется пространственным ГСПД.

Можно считать, что матрица K кодирует некоторое допустимое решение, так как матрица D вычисляется из уравнения $D = KA$. Поиск оптимального структурного решения заключается в нахождении такой матрицы K, которая минимизирует заданный критерий качества. При этом можно сначала задать координаты векторов матрицы D_{o}, обеспечивающие условия минимального значения T_C, а координаты векторов $\mathbf{K}_i$ найти из соотношения $K = D_{\text{o}} A_{\text{o}}^{-1}$, где D_{o} — матрица векторов-дуг; A_{o} — матрица инцидентности остова ГСПД.

При поиске эффективных структурных решений необходимо руководствоваться следующими закономерностями.

Конфигурация алгоритма является корректной, если в матрице K нет двух одинаковых векторов, т.е.

$$\forall \mathbf{K}_i, \mathbf{K}_j \ (\mathbf{K}_i \neq \mathbf{K}_j, i \neq j). \tag{6}$$

Расписание выполнения алгоритма корректно, если операторы, отображаемые в один и тот же ПЭ, выполняются в различных тактах:

$$\forall \mathbf{K}_i, \mathbf{K}_j \ (k_i = k_j, s_i = s_j) \Rightarrow t_i \not\equiv t_j \bmod c. \tag{7}$$

Однотипные операторы следует отображать в ПЭ того же типа:

$$\mathbf{K}_i, \mathbf{K}_j \in K_{p,q} (k_i = k_j = p, s_i = s_j = q), |K_{p,q}| \leq c, \tag{8}$$

где $K_{p,q}$ — множество векторов-вершин операторов p-го типа, отображаемых в q-й ПЭ p-го типа ($q = 1, 2, \ldots, q^p_{\max}$).

Если ГСПД — циклический, т.е. в нем есть циклы межитерационной зависимости, то должна быть равна нулю сумма векторов-дуг $\mathbf{D}_j$, входя-

щих в любой из циклов графа, включая дуги межитерационной зависимости $\mathbf{D}_{Dj}$. Тогда для *i*-го цикла

$$\sum b_{i,j}\mathbf{D}_j = (0,0,0)^T, \qquad (9)$$

где $b_{i,j}$ — элемент *i*-й строки цикломатической матрицы ГСПД. Обратные векторы-дуги или дуги межитерационной зависимости $\mathbf{D}_{Dj} = (0, 0, -wc)^T$ означают задержку, равную *w* циклов (итераций). Таким образом, КА составляет объединение ациклической конфигурации, выполняющей вычисления одной итерации и множества дуг $\mathbf{D}_{Dj}$, представляющих межитерационную задержку переменных на *wc* тактов.

Исходный ГСПД ресинхронизируется с целью получения минимального числа задержек, представляемых обратными векторами-дугами $\mathbf{D}_{Dj}$. Эффективная КА получается в два этапа.

На первом этапе вершины ГСПД вместе с дугами размещаются в трехмерном пространстве как множества векторов $\mathbf{K}_i$ и $\mathbf{D}_j$ с учетом условий (6) — (9), т.е. формируется исходная КА. Минимизируется число ПЭ в искомой структуре выполнением требования $|K_{p,q}| \to c$. При этом вектору $\mathbf{D}_j$ придаются значения $(k_i, s_i, t_i)^T$, $t_i \geq 1$, если инцидентная началу этого вектора вершина генерирует переменную, которая должна быть запомнена в одном или нескольких регистрах. Назначив всем или большинству векторов $\mathbf{D}_j$ минимальные ненулевые значения с учетом (9), а также принимая во внимание задержки *d* в схемах, отвечающих инцидентным вершинам, получаем пространственный ГСПД, в котором выполнена ресинхронизация, обеспечивающая минимизированную длительность тактового интервала T_C [16, 17]. При этом на одном этапе получаем и расписание срабатывания операторов алгоритма, и очертания искомой структуры, позволяющие достаточно точно оценить как быстродействие, так и аппаратные затраты ВУ.

На втором этапе выполняется уравновешивание КА. Для этого рассматривается ациклический подграф ГСПД без векторов-дуг $\mathbf{D}_{Dj}$. Во все его дуги включаются промежуточные вершины операторов задержки (регистров). В результирующей уравновешенной КА все векторы-дуги, кроме векторов $\mathbf{D}_{Dj}$, имеют вид $\mathbf{D}_j = (a_j, b_j, 1)^T$ или $\mathbf{D}_j = (a_j, b_j, 0)^T$. При этом вершины-операторы образуют ярусы, расстояние между которыми по координате времени *t* равно одному такту. В уравновешенной КА в результате взаимных перестановок векторов-вершин из одного яруса минимизируется число регистров и входов мультиплексоров результирующего ВУ.

Структуру ВУ и расписание выполнения алгоритма можно получить, расщепив КА K_G на конфигурацию структуры K_S и конфигурацию пред-

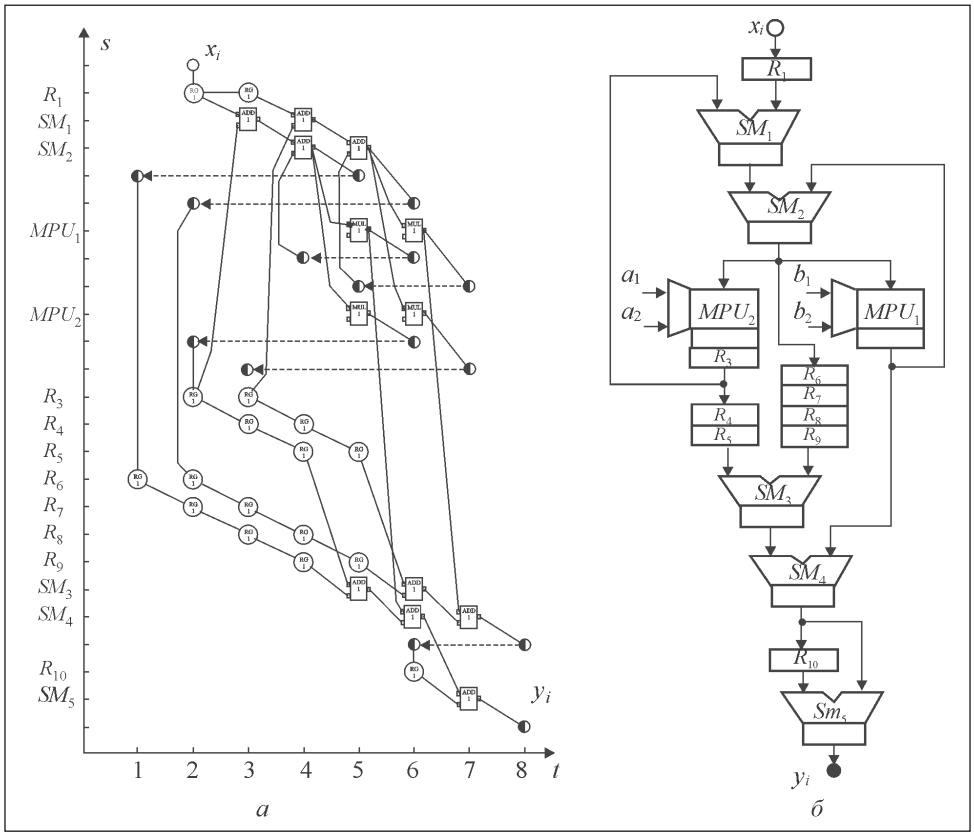


Рис. 8. Пространственный ГСПД после второго этапа синтеза ГСПД, представленного на рис. 7, *а* (*а*) и блок-схема результирующей структуры (*б*): ← — векторы $\mathbf{D}_{Dj}$; ○ — вершина ввода; ● — вершина вывода; ◐ — вершина передачи переменной межитерационной зависимости; RG, ADD и MUL — вершины задержки, сложения и умножения

шествования, которые имеют ту же матрицу A, а векторы матрицы K_S координат ПЭ и матрицы моментов срабатывания K_T соответствуют координатам векторов матрицы K, т.е. $(k_i, s_i)^T$ и t_i.

Пространственный ГСПД однозначно преобразуется в описание ВУ на языке VHDL, которое транслируется в проект, описанный на уровне вентилей компилятором-синтезатором. Поэтому структуру ВУ и расписание для пространственного ГСПД составлять не обязательно [18].

На рис. 8, *а*, показан пространственный ГСПД после второго этапа синтеза ГСПД, представленного на рис. 7, *а*. Как видно из рис. 8, *б*, длительность тактового интервала в полученной структуре является минимальной: $T_C = \max(d_A, d_M)$.

Результаты эксперимента. Согласно ГСПД, представленному на рис. 7, *а*, и ГСПД, полученным различными методами оптимизации, были построены пять различных структур фильтров. Эти структуры описаны языком VHDL и транслированы в ПЛИС Xilinx Kintex-7 с помощью САПР ISE. При этом использованы одинаковые требования к проекту, а параметр ограничения минимального периода синхросигнала подобран так, чтобы минимизировать значения T_C.

Результаты проектирования пяти структур фильтров представлены в таблице, в которой аппаратные затраты фильтра равны числу занимаемых логических таблиц S_L и блоков умножения S_M в ПЛИС. Критерий качества проекта, составляющий $Q_L = S_L T_A$ и $Q_M = S_M T_A$, равен объему аппаратных затрат, приходящихся на 1 МГц частоты дискретизации входного сигнала. При этом предполагается, что структура, в которой параметры Q_L и Q_M имеют меньшие значения, более предпочтительна по критерию производительность-стоимость.

Из таблицы видно, что применение методов оптимизации ГСПД позволило улучшить качество проекта в 1,1—2,5 раза. Метод раскручивания ГСПД обеспечивает получение ВУ с максимальной пропускной способностью (минимальным значением T_A) вследствие существенного увеличения аппаратурных затрат, особенно числа блоков умножения. Метод ресинхронизации пространственного ГСПД предназначен для получения структур ВУ с таким же значением коэффициента замедления c, как известный метод сворачивания ГСПД, но позволяет уменьшить аппаратурные затраты S_L и длительность тактового интервала T_C. В данном эксперименте этот метод обеспечил наилучшее качество проектного решения.

Следует заметить, что период алгоритма T_A ВУ, построенного по пространственному ГСПД с замедлением $c = 2$, оказался меньшим, чем ВУ, спроектированного по неоптимизированному ГСПД, для которого $c = 1$. Это означает, что с помощью ресинхронизации и многотактового выпол-

Метод проектирования ВУ на основе ГСПД	Номер рисунка	S_L	S_M	T_C, нс	T_A, нс	Q_L, МГц	Q_M, МГц
Конвейеризация	7, *а*	259	4	8,39	8,39	2,17	0,034
Реконвейеризация	7, *а*	212	4	5,78	5,78	1,23	0,023
Сворачивание	7, *б*	184	2	5,43	10,86	1,99	0,022
Раскручивание	6	510	14	6,94	3,47	1,77	0,049
Ресинхронизация пространственного ГСПД	8, *а*	144	2	3,10	6,20	0,89	0,012

нения алгоритма можно получить такую же пропускную способность, как у однотактового ВУ при существенно меньших аппаратурных затратах. Параметры Q_L и Q_M, характеризующие уровень энергопотребления ВУ, также свидетельствуют о том, что ВУ, спроектированные по пространственному ГСПД, имеют меньший уровень энергопотребления.

Данный метод показал высокую эффективность также при синтезе структур рекурсивных фильтров [15, 19], процессоров для быстрого преобразования Фурье, двумерного дискретного косинусного преобразования [20], авторегрессионного анализа [21], решения задач линейной алгебры [22] и других конвейерных ВУ.

Выводы

Методы ресинхронизации уже более двух десятилетий успешно используются как в микроэлектронике и при проектировании ПЛИС, так и в программировании. Эти методы основаны на минимизации задержек прохождения сигналов между регистрами посредством такой перестановки регистров, которая функционально не изменяет алгоритм. Особенно важно выполнить такую минимизацию в маршрутах распространения переменных в обратной связи алгоритма, так как в этом случае рамки оптимизации жестко ограничены числом регистров в маршрутах. Предложенный метод ресинхронизации пространственного ГСПД высокоэффективен при синтезе конвейерных структур. В отличие от других методов ресинхронизации он обеспечивает максимальное отношение производительность-стоимость при отображении ГСПД с обратными связями.

СПИСОК ЛИТЕРАТУРЫ

1. *Сергиенко А.М., Симоненко В.П.* Алгоритмические модели обработки потоков данных // Электрон. моделирование. — 2008. — **30**, № 6. — С. 49—60.
2. *Handbook* of Algorithms for Physical Design Automation / Editors C.J. Alpert, D.P. Mehta, S.S. Sapatnekar — Auerbach Publications, 2008. — 1024 p.
3. *Bhattacharyya S.S., Murthy P.K., Lee E.A.* Software Synthesis from Dataflow Graphs. — Kluwer Academic Publ, 1996.
4. *Khan S.A.* Digital Design of Signal Processing Systems. — John Wiley & Sons, 2011. — 586 p.
5. *Ли Е.А., Мессершмитт Д.Г.* Вычисления с синхронными потоками данных // ТИИЭР. — 1987. — **75**, № 9. — С. 107—119.
6. *Edwards S., Lavagno L., Lee E.A., Sangiovanny-Vincentelli A.* Design of Embedded Systems: Formal Models, Validation and Synthesis // Proc. of the IEEE. — 1997. — Vol. 85, No 3. — P. 366—390.
7. *Lee E.A., Messerschmitt D.G.* Static scheduling of synchronous data flow programs for digital signal processing // IEEE Trans. on Computers. — 1987. — Vol. 36, No 1. — P. 24—35.
8. *O'Neil T.W., Sha E.H.M.* Retiming synchronous data-flow graphs to reduce execution time // IEEE Trans. on Signal Processing. — 2001. — Vol. 49, No 10, — P. 2397 — 2407.

9. *Ito K., Parhi K.K.* Determining the Iteration Bounds of Single-Rate and Multi-Rate Data-Flow Graphs // Proc. 1994 IEEE Asia-Pacific Conf. on Circuits and Systems. Taipei, Taiwan. 5—8 Dec. 1994. — 1994. — P. 163—168.
10. *Leiserson C.E., Saxe J.B.* Retiming Synchronous Circuitry // Algorithmica. — 1991. — No 6. — P. 5 —35.
11. *Potkonjak M., Rabaey J.M.* Maximally and Arbitrarily Fast Implementation of Linear and Feedback Linear Computations // IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems. — 2000. — Vol. 19, No 1. —P. 30—43.
12. *Petersen W.P., Arbenz P.* Introduction to Parallel Computing. — Oxford University Press, 2004. — 259 p.
13. *The Systhesis* Approach to Digital System Design / Editors P. Micheli, U. Lauther, P. Duzy. — Kluwer Academic Pub, 1992. — 415 p.
14. *Кун С.* Матричные процессоры на СБИС. — М. : Мир, 1991. — 672 с.
15. *Сергиенко А.М., Симоненко В.П.* Отображение периодических алгоритмов в программируемые логические интегральные схемы //Электрон. моделирование. — 2007. — **29**, № 2. — С. 49—61.
16. *Каневский Ю.С., Овраменко С.Г., Сергиенко А.М.* Отображение регулярных алгоритмов в структуры специализированных процессоров // Там же. — 2002. — **24**, № 2. — С. 46—59.
17. *Сергієнко А.М.* Досконалий кістяк графа алгоритму //Вісн. Нац. технічного університету України «КПИ». Сер. Інформатика і обчислювальна техніка. — 2007. — **46**. — С. 62—67.
18. *Сергиенко А.М.* Методика проектирования цифровых фильтров с помощью VHDL // Зб. наук. праць. Ін-т проблем моделювання в енергетиці ім. Г.Є. Пухова. НАНУ «Моделювання та інформаційні технології». — 2002. — **12**. — С. 99—107.
19. *Сергиенко А.М., Лесик Т.М.* Динамически перестраиваемые цифровые фильтры на ПЛИС // Электрон. моделирование. — 2010. — **32**, № 6. — С. 47—56.
20. *Сергієнко А.М., Лепеха В.Л. Лесик Т.М.* Спецпроцесори для двовимірного дискретного косинусного перетворення // Вісн. НТУУ «КПІ». Інформатика і обчислювальна техніка. Зб. наук. праць. — 2007. — **47**. — С. 49—52.
21. *Сергиенко А.М.* Спецпроцессоры для авторегрессионного анализа сигналов // Электрон. моделирование. — 2010. — **32**, № 2. — С. 87—96.
22. *Sergiyenko A., Maslennikow O., Lepekha V. et al*. Parallel Implementation of Cholesky LL^T Algorithm in FPGA — Based Processor // Lecture Notes in Computer Science. — Berlin: Springer, 2008. — Vol. 4967. — P. 137—147.

A.M. Sergiyenko, V.P. Simonenko

OPTIMIZATION METHODS FOR SYNCHRONOUS DATAFLOW GRAPHS

Methods for the synchronous dataflow graph (SDF) retiming, and mapping it into pipelined datapaths are considered. A method of retiming the spatial SDF is proposed. The method is based on the SDF representation in the multidimensional space. The dimensions of this space are spatial coordinate of the processing unit, coordinate of the operator firing, and operator type. At the first stage of the datapath synthesis the operator nodes are placed in the space according to a set of rules and theorems providing the minimum hardware volume and minimum clock period for the given number of clock periods in the algorithm cycle. At the second stage of the synthesis this spatial SDF is balanced and optimized providing the minimum register and multiplexor number in the resulting datapath. The resulting spatial SDF is described in VHDL language and is modeled and compiled using proper CAD tools. The method is successfully proven by the synthesis of a set of infinite impulse response filters for FPGA.

K e y w o r d s: retiming, synchronous dataflow, scheduling, pipelining, folding, datapath.

REFERENCES

1. *Sergiyenko A. M., Simonenko V. P.* Algorithmic models of dataflow processing // Electronic Modeling. — 2008. — Vol. 30, No 6. — P. 49—60 (in Russian).
2. *Handbook* of Algorithms for Physical Design Automation / Editors C.J. Alpert, D.P. Mehta, S.S. Sapatnekar — Auerbach Publications, 2008. — 1024 p.
3. *Bhattacharyya S.S., Murthy P.K., Lee E.A.* Software Synthesis from Dataflow Graphs. — Kluwer Academic Publ., 1996.
4. *Khan S.A.* Digital Design of Signal Processing Systems. — John Wiley & Sons, 2011. — 586 p.
5. *Lee E.A., Messerschmitt D.G.* Synchronous data flow // Proc. IEEE — 1987.— Vol. 75, No 9. — P. 107—119 (in Russian).
6. *Edwards S., Lavagno L., Lee E.A., Sangiovanny-Vincentelli A.* Design of Embedded Systems: Formal Models, Validation, and Synthesis // Proc. of the IEEE. — 1997. — Vol. 85, No 3. — P. 366—390.
7. *Lee E.A., Messerschmitt D.G.* Static scheduling of synchronous data flow programs for digital signal processing // IEEE Trans. on Computers. — 1987. — Vol. 36, No 1. — P. 24—35.
8. *O'neil T.W*., *Sha E.H.M.* Retiming synchronous data-flow graphs to reduce execution time // IEEE Trans. on Signal Processing. — 2001. — Vol. 49, No 10. — P. 2397—2407.
9. *Ito K., Parhi K.K.* Determining the Iteration Bounds of Single-Rate and Multi-Rate Data-Flow Graphs // Proc. 1994 IEEE Asia-Pacific Conf. on Circuits and Systems. Taipei, Taiwan. — 5—8 Dec. 1994. — P. 163—168.
10. *Leiserson C.E., Saxe J.B.* Retiming Synchronous Circuitry // Algorithmica. — 1991. — No 6. — P. 5—35.
11. *Potkonjak M., Rabaey J.M*. Maximally and Arbitrarily Fast Implementation of Linear and Feedback Linear Computations // IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems. — 2000. — Vol. 19, No 1. — P. 30—43.
12. *Petersen W.P., Arbenz P.* Introduction to Parallel Computing.— Oxford University Press, 2004. — 259 p.
13. *The Systhesis* Approach to Digital System Design / Editors P. Micheli, U. Lauther, P. Duzy. — Kluwer Academic Pub., 1992. — 415 p.
14. *Kun S.* VLSI Array Processors. — Moscow: Mir, 1991. — 672 p. (in Russian).
15. *Sergiyenko A.M., Simonenko V.P.* Mapping of periodic algorithms onto programmed logic integral circuits // Electronic Modeling. — 2007. — Vol. 29, No 2. — P. 49—61 (in Russian).
16. *Kanyevskiy Ju.S., Ovramenko S.G., Sergiyenko A.M.* Mapping of regular algorithms onto structures of specialized processors // Ibid. — 2002. — Vol. 24, No 2. — P. 46—59 (in Russian).
17. *Sergiyenko A.M.* Perfect frame of the algorithm graph // Visnyk NTUU «KPI». Informatics, operation and computer science. — 2007. — Vol. 46. — P. 62—67 (in Ukrainian).
18. *Sergiyenko A.M.* Methods of design of digital filters with the help of VHDL // Collected Scientific Works. Pukhov Institute for Modeling in Energy Engineering. – Modeling and Information Technologies. — 2002. — Vol. 12. — P. 99—107 (in Russian).
19. *Sergiyenko A.M., Lesyk T.M.* Dynamic modified digital filters on PLIS // Electronic Modeling. — 2010. — Vol. 32, No 6. — P. 47—56 (in Russian).

20. *Sergiyenko A.M., Lepekha V.L., Lesyk T.M.* Special processors for two-dimensional discrete cosinus transformation // Visnyk NTUU «KPI». Informatics, operation and computer science. — 2007. — Vol. 47. — P. 49—52 (in Ukrainian).
21. *Sergiyenko A.M*. Special processory for autoregression analysis of signals // Electronic Modeling. — 2010. — Vol. 32, No 2. — P. 87—96 (in Rusian).
22. *Sergiyenko A., Maslennikow O., Lepekha V. et al.* Parallel Implementation of Cholesky LL^T Algorithm in FPGA—Based Processor // Lecture Notes in Computer Science. — Berlin: Springer, 2008. — Vol. 4967. — P. 137—147.

Поступила 07.08.14

СЕРГИЕНКО Анатолий Михайлович, д-р техн. наук, ст. науч. сотр. кафедры вычислительной техники Национального технического университета Украины «Киевский политехнический ин-т». В 1981 г. окончил Киевский политехнический ин-т. Область научных исследований — отображение алгоритмов в структуры вычислительных средств, цифровая обработка сигналов.

СИМОНЕНКО Валерий Павлович, д-р техн. наук, профессор Национального технического университета Украины «Киевский политехнический ин-т». В 1965 г. окончил Киевский политехнический ин-т. Область научных исследований — организация вычислительных процессов в вычислительных системах.