АВТОМАТИЗАЦИЯ ПОСТРОЕНИЯ ВЫЧИСЛИТЕЛЬНОГО ЯДРА ДЛЯ СПЕЦВЫЧИСЛИТЕЛЕЙ НА БАЗЕ FPGA

Ю.А. Милюкин

Юго-Западный госуниверситет, Курск

Рассматриваются методы адаптивных вычислений. Вычисления, при которых аппаратная архитектура вычислителя подстраивается под структуру заданного вычислительного процесса, может быть основана на математической модели вычислительного процесса и её математического образа, задающего двумерную топологию вычислительного ядра. Приведен алгоритм распараллеливания вычислений, позволяющий распознавать по маске математические формулы.

Введение

Характеристики современных ПЛИС и структурных ASIC, включающих встроенные процессоры и блоки памяти, позволяют реализовать значительное число приложений в виде системы на программируемом кристалле.

FPGA представляет собой микросхему высокого уровня интеграции, содержащую во внутренней области матрицу функциональных блоков и систему их межсоединений, размещенную между столбцами матрицы, а в периферийной области – блоки ввода-вывода.

Современные FPGA предоставляют встроенную память и встроенные логические блоки для арифметических операций. Преимущество встроенных логических блоков – лучшая скорость и пространство. Кроме того, встроенная память проще, чем внешний интерфейс памяти. FPGA часто связаны с микропроцессорами. Также основное из преимуществ встроенных микропроцессоров – уменьшение задержки связи между микропроцессором и FPGA.

Основным достоинством классических FPGA является то, что при проведении разработки и в процессе отладки у разработчика имеется возможность многократно загружать проект в микросхему и проверять результаты своей работы непосредственно на «живом» изделии. При этом стоимость отладки любого этапа проекта невелика. Недостатком работы классических FPGA является сама система загрузки данных в микросхему. Во-первых, для работы таких FPGA необходимо внешнее устройство, в котором хранится файл инициализации. Во-вторых, информация, загружаемая в FPGA, может быть скопирована и использована для получения пиратских копий разрабатываемого изделия. Защититься от такого копирования принципиально невозможно, поскольку информация о конфигурации передается от одной микросхемы к другой через внешние выводы микросхем по линиям связи, расположенным на печатной плате. Есть и еще один существенный недостаток у классических FPGA, правда, он не так часто бросается в глаза. Во время работы, особенно в такой аппаратуре, которая работает в долговременном непрерывном режиме без перезагрузки, могут происходить частичные сбои конфигурации. Определить такой сбой не всегда возможно. Исправить частичные сбои конфигурации можно только полной перезагрузкой конфигурации, так как частичная загрузка конфигурации в классических FPGA невозможна.

Задача заключается в том, чтобы разработать технологию автоматического конструирования вычислительного ядра для заданной в символьном виде вычислительной задачи.

FPGAs непосредственно не программируются. Синтез инструментов преобразует код в поток битов, который загружается в память FPGA. Как правило, для настройки устройства используются языки описания оборудования (VHDL и Verilog). Основная проблема программирования FPGA состоит в том, что и VHDL, и Verylog (то есть языки программирования аппаратуры) предполагают побитовую обработку информации. Но также есть возможность использовать языки высокого уровня (C++). Реализация на языке C++ позволяет обработку информации с заданной разрядностью (8, 16, 24, 32 и т.д.), кратной байту, что является более предпочтительным для автоматического конструирования вычислительного ядра.

На сегодняшний день задача автоматического конструирования топологии вычислительного ядра частично решена только для аналоговых интегральных схем Anadigm Designer® 2 на аппаратной платформе FPAA.

1. Конструирование ядра через кортеж и маску

Для автоматизации конструирования спецвычислительных систем на программируемом кристалле необходимо решить две задачи:

1) синтез топологии многопроцессорного вычислительного ядра (вычислителя), в котором осуществлено распараллеливание вычислений под ту или иную вычислительную задачу;

2) создание программного кода, доставляющего решение вычислительной задаче на синтезированном вычислителе.

Такие задачи были поставлены по причине проблем, возникающих при практическом использование этих платформ:

1) производители «бортовых» вычислителей и/или их компонентов выпускают эти изделия с заведомо избыточной функциональной структурой. При этом разработчикам, как правило, известен математический функционал, подлежащий технической реализации, и, как следствие, известна функциональная структура вычислителя, которая может быть заведомо задана производителю под заказную СБИС, что влечёт за собой дальнейшую миниатюризацию воплощения её в «железе»;

2) не меньшую трудность у разработчиков вызывает собственно программирование выбранной для технической реализации встраиваемой системы, поскольку, например, электрическая принципиальная схема современного аналогового вычислителя может иметь размеры футбольного стадиона, аналогично цифровой вычислитель требует сотен (и даже тысяч) страниц программного кода.

Для решения возникающих проблем был предложен автоматический синтез топологии вычислительного ядра встраиваемой системы по заданному вычислительному процессу на заданной аппаратной платформе.

Физическая постановка задачи адаптивных вычислений:

Дано:

1) математическая модель заданного вычислительного процесса

$$u(a_1, a_2, \dots, a_n) = \sum_{i=1}^{k} \pm c_i \prod_{j=1}^{n} \varphi_{i,j,\alpha}(a_j^{m_j}), \quad (1)$$

где u – выход вычислительного процесса; a_i $(i = \overline{1,n})$ – входы вычислительного процесса (сигналы с датчиков в аналоговой и/или цифровой форме); c_i – коэффициенты усиления (числа); $\varphi_{i,j,\alpha}$ – обозначение операции взятия функции, где α – обозначение класса функции ($\alpha = 0$ – алгебраические функции, $\alpha = 1$ – тригонометрические функции, $\alpha = 2$ – показательные функции и т.д.); m_j – обозначение математической операции ($m_j = 1$ – деление; $m_j = p$ – возведение в степень $p > 0$; $m_j = {}^{1}/_{p}$ – извлечение арифметического корня степени $p > 0$; $m_j = (p)$ – операция взятия производной степени p; $m_j = \{p\}$ – операция взятия интеграла кратности p.

2) заданная аппаратная платформа, на которой процесс (1) нужно реализовать.

Требуется автоматически сконструировать топологию вычислительного ядра, реализующую заданный вычислительный процесс (1) на заданной аппаратной платформе.

Для решения задачи неравенство меняется на его математический образ, формализующийся двумя понятиями: кортежем и маской. Тем самым предложенный образ вычислительного процесса задаёт двумерную топологию многопроцессорного вычислительного ядра организации параллельных вычислений слагаемых заданной алгебраической суммы, каждое из которых вычисляется в результате последовательного перемножения результатов выполнения математических операций, формализуемых соответствующей строкой маски.

Можно провести процесс распараллеливания вычисления, распознав содержащиеся в маске математические формулы (если они в ней есть) (рис. 1).

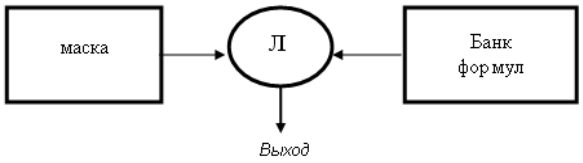

Рис. 1. Функциональная схема алгоритма автоматического распознавания математических формул

Здесь Л – логический блок для работы с банком формул, представляющим собой набор масок распознаваемых формул.

В случае графического представления технической реализации математических операций необходимо рассматривать вычислительное ядро с распараллеливанием вычислений, которое будет реализовывать вычислительный процесс (1). Такая топология является отправной точкой для разработки интерпретатора (конвертера) для преобразования элементов маски в эквивалентные им фрагменты технических устройств в интегральном исполнении, реализующих ту или иную математическую операцию (сложение, вычитание, умножение, деление, возведение в степень, извлечение арифметического корня заданной степени, дифференцирование, интегрирование, а также взятие действительной функции одной переменной).

Для синтезирования электрической принципиальной схемы бортового вычислителя с использованием обозначений пакета по программированию аналоговых интегральных схем Anadigm Designer® 2 на аппаратной платформе FPAA, позволяющего «прошивать» FPAA с использованием алфавита команд разработчиков Anadigm, будем применять рефлексивную семантику преобразования модели математической операции в модель технической реализации. Другими словами интерпретировать один формализованный язык (язык математических операций) в категории другого языка (языка технической реализации математических операций в аналоговой и/или цифровой форме). Техническая реализация математической операции в аналоговой форме, как правило, представляется посредством электрической принципиальной схемы.

Некоторые из электрических принципиальных схем, автоматически сформированных описанной в [6] программой-конвертером языка кортежа и маски в обозначения пакета Anadigm Designer® 2, то есть в файл с расширением *.cpp, который затем компилируется внешним C++-компилятором (платный пакет производства компании Anadigm), предназначенным для «физической прошивки» FPAA, приведены на рис. 2-8.

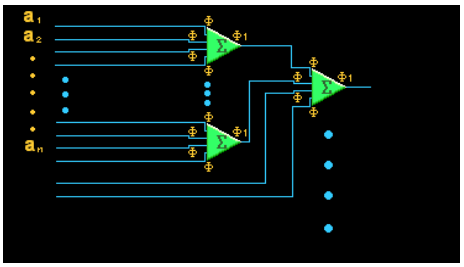

Рис. 2. Электрическая принципиальная схема реализации маски (сумматора)

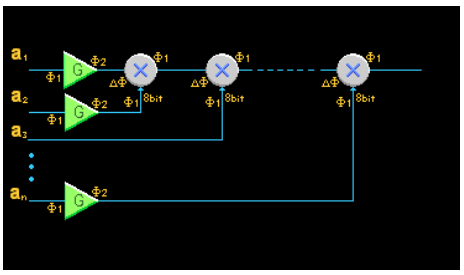

Рис. 3. Электрическая принципиальная схема реализации строки маски (умножителя в общем случае *n*-сигналов с коэффициентом усиления, отличным от 1)

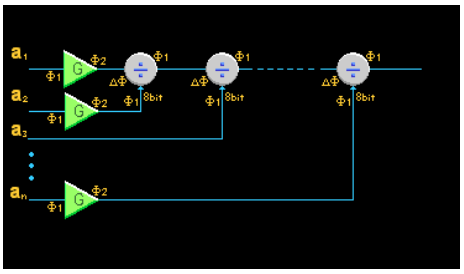

Рис. 4. Электрическая принципиальная схема реализации строки маски (делителя в общем случае *n*-сигналов с коэффициентом усиления, отличным от 1)

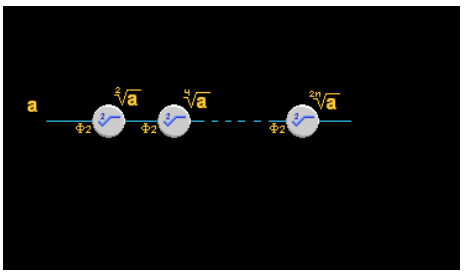

Рис. 5. Электрическая принципиальная схема взятия арифметического корня

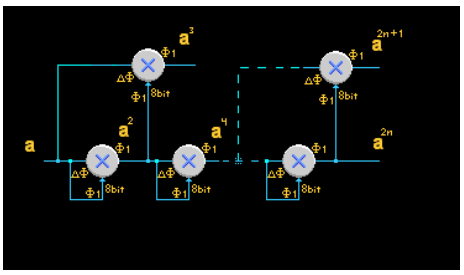

Рис. 6. Электрическая принципиальная схема возведения в степень

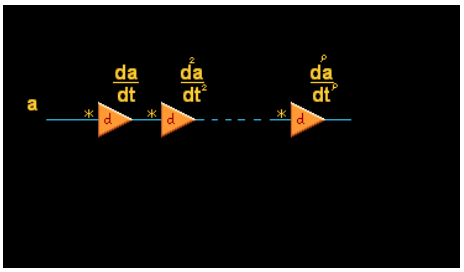

Рис. 7. Электрическая принципиальная схема взятия производной степени ρ

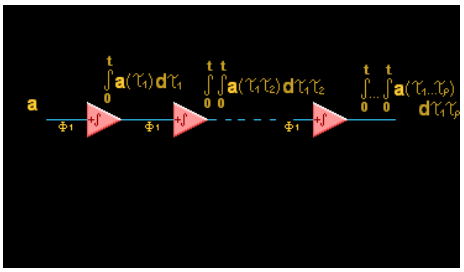

Рис. 8. Электрическая принципиальная схема взятия ρ-кратного интеграла

Литература

1. Подчукаев В.А., Милюкин Ю.А., Филонович А.В. Методы адаптивных вычислений и техническая реализация современных систем автоматического управления // Сборник научных статей по материалам III Международной научно-практической конференции «Применение инновационных технологий в научных исследованиях». Курск, 2011. С. 6.
2. Подчукаев В.А., Милюкин Ю.А., Филонович А.В., Кутуев А.Н. Автоматическое проектирование интегральных микросхем // Сборник научных статей по материалам II Международной научно-практической конференции «Применение инновационных технологий в научных исследованиях».Курск, 2011. С. 137.
3. Подчукаев В.А., Милюкин Ю.А., Филонович А.В., Кутуев А.Н. Вычислительный кластер как аналог его функциональной схемы // Сборник научных статей по материалам II Международной научно-практической конференции «Применение инновационных технологий в научных исследованиях». Курск, 2011. С. 142.
4. Применение программируемых логических интегральных схем в электронной аппаратуре // Электроника и системы управления, 2009. №4.
5. Подчукаев В.А., Кулаков К.А. Аналитическое проектирование топологии интегральных микросхем // Матер. Междунар. науч.-техн. конф. «Мехатроника, автоматизация, управление – 2007». Таганрог: Изд-во ТТИ ЮФУ, 2007. С. 260-264.
6. Подчукаев В.А., Шевченко Д.С. Автоматическое проектирование технической реализации законов управления на платформе программируемых аналоговых интегральных схем производства компании Anadigm // Цифровые системы управления и обработки информации: приложение к журналу «Мехатроника, автоматизация, управление». 2008. № 7. С. 7-11.