Обработка радиолокационной информации: ПЛИС или графические процессоры?

Майкл ПАРКЕР (Michael PARKER)

Хотя графические процессоры общего назначения дают высокую пиковую производительность для операций с плавающей точкой, ПЛИС обладают более привлекательными уровнями производительности для таких операций. Более того, ПЛИС корпорации Altera теперь поддерживают стандарт OpenCL — используемый в графических процессорах.

Введение

ПЛИС и процессоры (CPU) уже давно являются неотъемлемой частью устройств обработки радиолокационной информации. ПЛИС традиционно используют для первичной обработки информации, а процессоры (CPU) — для ее окончательной обработки. Радиолокационные системы (РЛС) наращивают свои возможности и усложняются, что приводит к резкому росту требований в области обработки информации. ПЛИС сохранили темпы увеличения производительности обработки информации и пропускную способность, в то время как процессоры отставали в обеспечении производительности обработки сигнала в РЛС нового поколения. Поэтому что бы обеспечить работу при столь больших нагрузках пришлось прийти к использованию процессорных ускорителей, таких как графические ускорители (GPU).

За последние несколько лет GPU получили мощные платформы для операций с плавающей запятой, известные как GP-GPU, которые дают высокую пиковую производительность для операций с плавающей запятой в секунду (FLOPs). ПЛИС, традиционно использовавшиеся для обработки цифрового сигнала (DSP) с фиксированной точкой, теперь предлагают конкурирующие уровни производительности для операций с плавающей точкой, что делает их подходящими для быстрой конечной обработки информации с РЛС.

Высокопроизводительные ПЛИС корпорации Altera следующего поколения будут иметь производительность как минимум 5 TFLOPs, используя ядро Intel 14 Нм Tri-Gate. Можно ожидать получение производительности до 100 GFLOPs/Вт, используя эту передовую технологию изготовления полупроводниковых приборов. Более того, ПЛИС корпорации Altera теперь поддерживают язык OpenCL, используемый в графических процессорах.

Оценка ПЛИС с пиковой производительностью в GFLOPs

Современные ПЛИС имеют производительность в 1+ пиковая TFLOPs [1], в то время как новейшие графические процессоры AMD и Nvidia обладают более высокой производительностью, примерно до 4 TFLOPs. Однако для ПЛИС с пиковой производительностью в GFLOPs или TFLOPs имеется мало информации о производительности данного устройства в конкретном применении. На нем просто указывается общее число теоретических сложений и умножений с плавающей запятой, которые могут быть выполнены в секунду. Этот анализ показывает, что ПЛИС во многих случаях превышают пропускную способность GPU по алгоритмам и объемам данных при обработке информации РЛС.

Общим алгоритмом средней сложности является быстрое преобразование Фурье (FFT). Поскольку РЛС зачастую выполняют большую часть обработки информации в частотной области, очень часто используется алгоритм FFT. Например, расчет FFT на 4096 точек может быть выполнен с помощью операций с плавающей запятой одинарной точности. Это дает возможность ввода и вывода четырех комплексных выборок за один временной цикл. Каждое ядро с алгоритмом FFT может работать с производительностью 80 GFLOPs, а большая ПЛИС, созданная по 28-нм технологии, может заменить семь таких ядер.

Однако, как показано на рис. 1, производительность алгоритма FFT на этой ПЛИС равна почти 400 GFLOPs. Этот результат основан на компиляции в среде OpenCL, без опробования на ПЛИС. При использовании логики блокирования и Design Space Explorer (DSE) при оптимизации семиядерная архитектура сможет достичь f_{max} одноядерной архитектуры, повышая эту величину бо-

Точек	4096	**Архитектура**	Radix 2^2 с прямой связью
Данные	Комплекс	**Параллельных маршрутов**	4

	ALMs	Регистры	Блоки ЦОС	Блоки M20K	f_{MAX} (МГц)	GFLOPs
Одно ядро с FFT	36,7K	73K	60	136	340	81,6
Ускоритель семи ядер с FFT /ПЛИС	252K	457K	420	1,134	230	386,1
Полная оптимизация семи ядер с FFT /ПЛИС	257K	511K	420	952	340	571,2

Рис. 1. Производительность ПЛИС с алгоритмом FFT Stratix V 5SGSD8 и плавающей запятой

лее чем до 500 GFLOPs, то есть с более чем 10 GFLOPs/Вт при использовании ПЛИС, изготовленной по 28-нм технологии.

С точки зрения сравнения параметров графических процессоров, они не являются эффективными на этих длинах быстрого преобразования Фурье, и поэтому таблицы истинности здесь не представлены. Использование GPU становится эффективным с длинами быстрого преобразования Фурье на нескольких сотнях тысяч точек, когда оно может дать полезное ускорение для процессора. Однако короткие длины преобразования Фурье широко распространены при обработке радиолокационной информации, где длины 512–8192 точек являются нормой.

В целом полезная производительность в GFLOPs зачастую представляет собой долю пиковой или теоретической производительности. По этой причине для сравнения лучше всего использовать такой алгоритм, который дает корректное представление характеристик для типовых применений. Хотя алгоритм таблиц истинности и увеличивает сложность, но он дает более представительную фактическую производительность РЛС.

Алгоритм таблиц истинности

Вместо того чтобы полагаться на приводимую производителем пиковую производительность в GFLOPs, в технологических устройствах обработки информации следует использовать альтернативу в виде сторонних оценок с использованием примеров достаточной сложности. Общим алгоритмом для обработки данных с пространственно-временной адаптацией (Space-Time Adaptive Processing, STAP) РЛС является разложение Холецкого. Этот алгоритм часто используется в линейной алгебре для эффективного решения множественных уравнений и может быть использован в корреляционных матрицах.

Алгоритм Холецкого имеет высокую числовую сложность и почти всегда требует численного представления с плавающей запятой для получения корректных результатов. Необходимые вычисления пропорциональны $N3$, где N — размер матрицы. Поскольку РЛС обычно работают в режиме реального времени, обязательным требованием является высокая пропускная способность. Результат будет зависеть от размера матрицы и пропускной способности для требуемой обработки матрицы, что часто составляет более 100 GFLOPs.

В таблице 1 представлены результаты сравнительного анализа графического процессора Nvidia с номиналом в 1,35 TFLOPs и использованием различных библиотек, и Xilinx Virtex6 XC6VSX475T, оптимизированной ПЛИС для обработки цифрового сигнала с фиксированной запятой (DSP) с плотностью распределения 475K LCs. Эти устройства схожи по плотности с ПЛИС корпорации Altera, использующими таблицы истинности Холецкого. Библиотеки LAPACK и MAGMA являются коммерческими библиотеками, в то время как ядро графического процессора с производительностью в GFLOPs использует язык программирования OpenCL, разработанный в университете штата Теннесси [2]. Приведенные ниже данные со всей очевидностью показывают больший уровень оптимизации на матрицах меньших размеров.

Таблица 1. Ядро графического процессора и ПЛИС Xilinx с таблицами истинности Холецкого

Матрица	Библиотека LAPACK GFLOPs	Библиотека «Магма», GFLOPs	Ядро графического процессора, GFLOPs	ПЛИС, GFLOPs
512 (SP) 512 (DP)	19,49 11,99	22,21 20,52	58,4 57,49	19,23
768 (SP) 768 (DP)	29,53 18,12	38,53 36,97	81,87 54,02	20,38
1024 (SP) 1024 (DP)	36,07 22,06	57,01 49,6	67,96 42,42	21
2048 (SP) 2048 (DP)	65,55 32,21	117,49 87,78	96,15 52,74	–

Примечание.
SP — одинарная точность; DP — двойная точность.

ПЛИС Stratix V корпорации Altera среднего размера (460K логических элементов (LEs)) сравнивалась с ПЛИС Altera с использованием алгоритма Холецкого на операциях с плавающей запятой одинарной точности. Как показано в таблице 2, производительность ПЛИС Stratix V с алгоритмом Холецкого намного выше, чем у Xilinx. Сравнительный анализ Altera также включает QR-разложение, представляющее собой алгоритм обработки другой матрицы разумной сложности. Алгоритм Холецкого и QR-разложение используются как параметризуемые ядра от корпорации Altera.

Таблица 2. ПЛИС Altera с таблицами истинности Холецкого и QR

Алгоритм (комплекс, одинарная точность)	Размер матрицы	Векторное множество	f_{max}, МГц	GFLOPs
Холецкий	360×360	90	190	92
	60×60	60	255	42
	30×30	30	285	25
QR	450×450	75	225	135
	400×400	100	201	159
	250×400	100	203	162

Следует отметить, что размеры матрицы для сравнительного анализа неодинаковы. Результаты, полученные в университете Теннесси, начинаются с размеров матрицы 512×512, в то время как сравнительный анализ корпорации Altera начинается с размеров 360×360 для алгоритма Холецкого и 450×450 для QR-разложения. Причина в том, что ядро графического процессора очень неэффективно при меньших размерах матриц, поэтому нет смысла использовать его для ускорения процессора в этих случаях. Напротив, ПЛИС может эффективно работать с гораздо меньшим размером матрицы. Эта эффективность имеет решающее значение, поскольку радиолокационным системам нужна довольно высокая пропускная способность, выражаемая в тысячах матриц в секунду. Таким образом, используются меньшие размеры матриц, даже за счет разделения большой матрицы на более мелкие с последующей обработкой.

В дополнение ко всему сравнительный анализ корпорации Altera проводился по каждому ядру с алгоритмом Холецкого. Каждое параметризуемое ядро с алгоритмом Холецкого позволяет выбрать размер матрицы, векторное множество и глубину канала. Векторное множество примерно определяет ресурсы ПЛИС. Больший (360×360) размер матрицы использует большее векторное множество, что позволяет использовать одно ядро в этом ПЛИС с 91 GFLOPs. Меньший (60×60) размер матрицы использует меньше ресурсов, поэтому два ядра могут быть реализованы, в общей сложности, на 2×42 = 84 GFLOPs. Наименьший (30×30) размер матрицы допускает использование трех ядер, в общей сложности 3×25 = 75 GFLOPs.

ПЛИС гораздо лучше подходит для решения задач с меньшими размерами данных, что приемлемо для многих РЛС. Пониженная эффективность ядра графических процессоров связана с вычислительными нагрузками, увеличивающимися как N3, данными ввода/вывода, возрастающими как N2, и, в конечном счете, узкими местами ввода/вывода ядра графического процессора, создавая меньше проблем при увеличении набора данных. Кроме того, из-за увеличения размеров матрицы ее пропускная способность в секунду резко падает из-за увеличения времени обработки каждой матрицы. В какой-то момент пропускная способность становится слишком низкой и непригодной для работы в реальном времени в РЛС.

При использовании преобразования Фурье нагрузка вычисления увеличивается до Nlog2N, в то время как данные ввода/вывода возрастают как N. Опять же, при очень больших объемах данных ядро графического процессора становится эффективной вычислительной системой. Напротив, ПЛИС является эффективным вычислительным устройством для всех размеров данных и лучше подходит для большинства радиолокационных операций, где размеры преобразования Фурье небольшие, а пропускная способность является основным параметром.

Методология проектирования графического процессора и ПЛИС

Ядро графического процессора программируется с использованием либо собственного языка CUDA от Nvidia, либо открытого стандарта языка программирования OpenCL. Эти языки очень похожи по своим возможностям, с той единственной разницей, что CUDA может использоваться только на графических процессорах Nvidia.

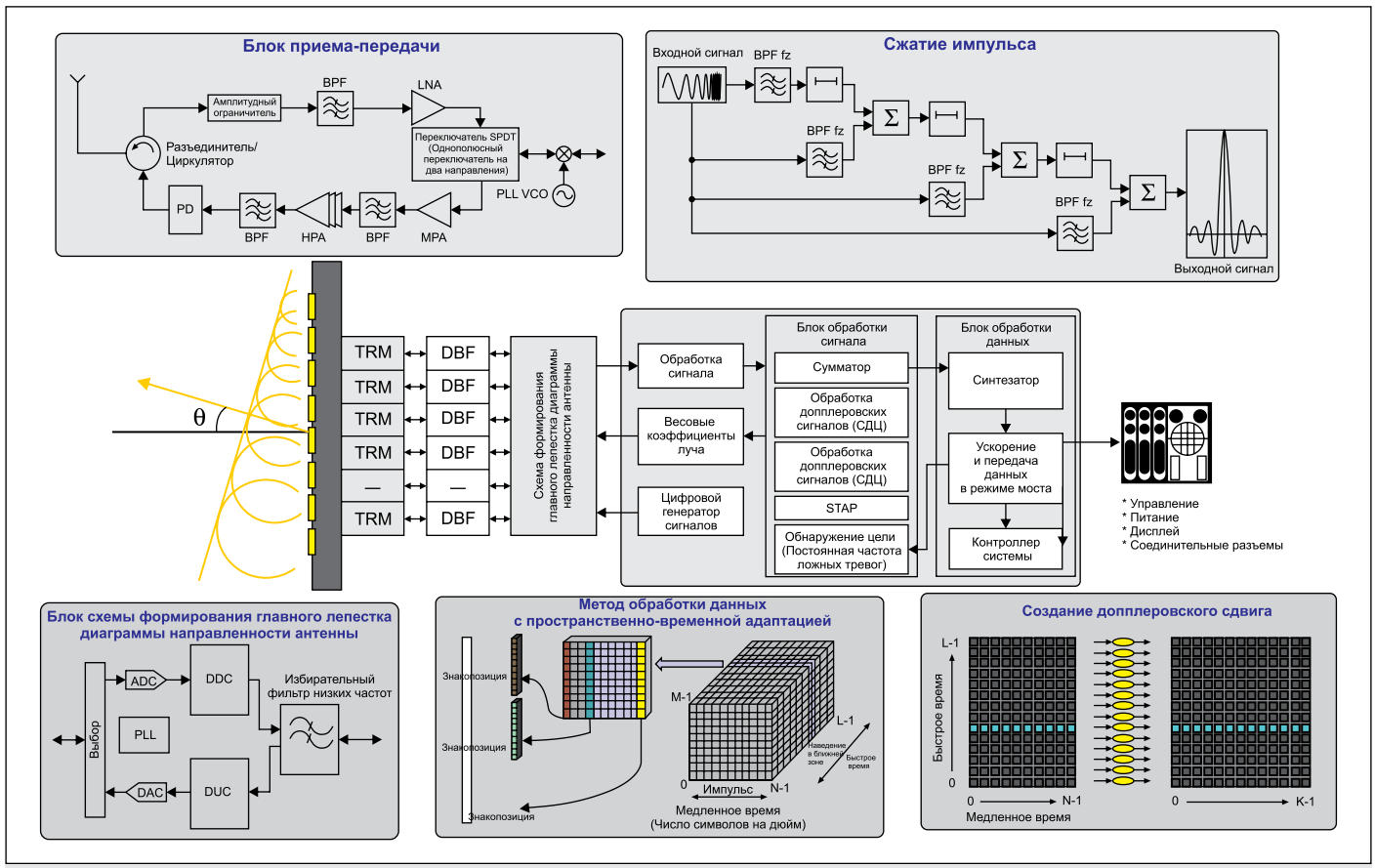

Рис. 2. Основная блок-схема обработки сигнала РЛС

ПЛИС, как правило, программируются с помощью языков описания аппаратуры Verilog или VHDL. Ни один из этих языков полностью не подходит для поддержки программирования с плавающей запятой, хотя последние версии включают определение, пусть и не обязательного обобщения, чисел с плавающей запятой. Например, в языке описания Verilog простая вещественная переменная является аналогом числа одинарной точности стандарта IEEE, а вещественная переменная — аналогом числа двойной точности стандарта IEEE.

Библиотека DSP Builder Advanced Blockset

Синтез трактов данных с плавающей запятой в ПЛИС с использованием традиционных методов очень неэффективен, что подтверждено низкой производительностью ПЛИС фирмы Xilinx на алгоритме Холецкого, реализованного с помощью генератора параметризированных модулей Core Generator фирмы Xilinx с универсальным ядром, предназначенным для автоматизированной подготовки описаний элементов, выполняющих различные арифметические операции с плавающей запятой. Тем не менее корпорация Altera предлагает две альтернативы. Первая заключается в использовании библиотеки DSP Builder Advanced Blockset с технологией проектирования компании Mathworks на базе Model-Based Design. Этот инструмент поддерживает числа как с фиксированной, так и с плавающей запятой, а также семь различных уровней точности обработки с плавающей запятой, включая половинную, одинарную и двойную точность по стандарту IEEE. Он также поддерживает векторизацию, которая необходима для эффективного применения линейной алгебры. Самое главное заключается в его способности эффективно отображать цепи с плавающей запятой на сегодняшних архитектурах ПЛИС с фиксированной запятой, о чем свидетельствует их сравнительный анализ производительности в 100 GFLOPs на алгоритме Холецкого при среднем размере ПЛИС по 28-нм технологии. Для сравнения, использование алгоритма Холецкого на тех же размерах ПЛИС фирмы Xilinx без этой возможности синтеза дает только 20 GFLOPs производительности по тому же алгоритму [2].

Язык программирования OpenCL для ПЛИС

Язык программирования OpenCL сходен с языками программирования ядра графического процессора. Компилятор OpenCL [3] для ПЛИС означает, что код OpenCL, написанный для AMD или Nvidia GPU, может быть скомпилирован на ПЛИС. Кроме того, компилятор OpenCL корпорации Altera позволяет программам ядра графического процессора использовать ПЛИС без необходимости разработки типового набора приемов проектирования ПЛИС.

Использование OpenCL с ПЛИС дает несколько ключевых преимуществ по сравнению с ядром графического процессора. Важно, что ядра графических процессоров, как правило, ограничены по входу/выходу. Все входные и выходные данные должны быть переданы центральным процессором через интерфейс PCI Express (PCIe). Получаемые задержки могут привести к потере скорости обработки ядром процессора, что снижает производительность.

Расширения языка программирования OpenCL для ПЛИС

ПЛИС хорошо известны благодаря высокой пропускной способности ввода/вывода, что позволяет осуществлять поточную передачу данных в ПЛИС и из него через Gigabit Ethernet (GbE), Serial RapidIO (SRIO) или непосредственно через АЦП и ЦАП. Корпорация Altera определила специфическое для производителя расширение стандарта OpenCL для поддержки непрерывного режима передачи данных. Это расширение является важнейшей особенностью в РЛС, так как позволяет передавать данные непосредственно от формирования диаграммы направленности, с предварительной обработкой с фиксированной запятой, и цифро-

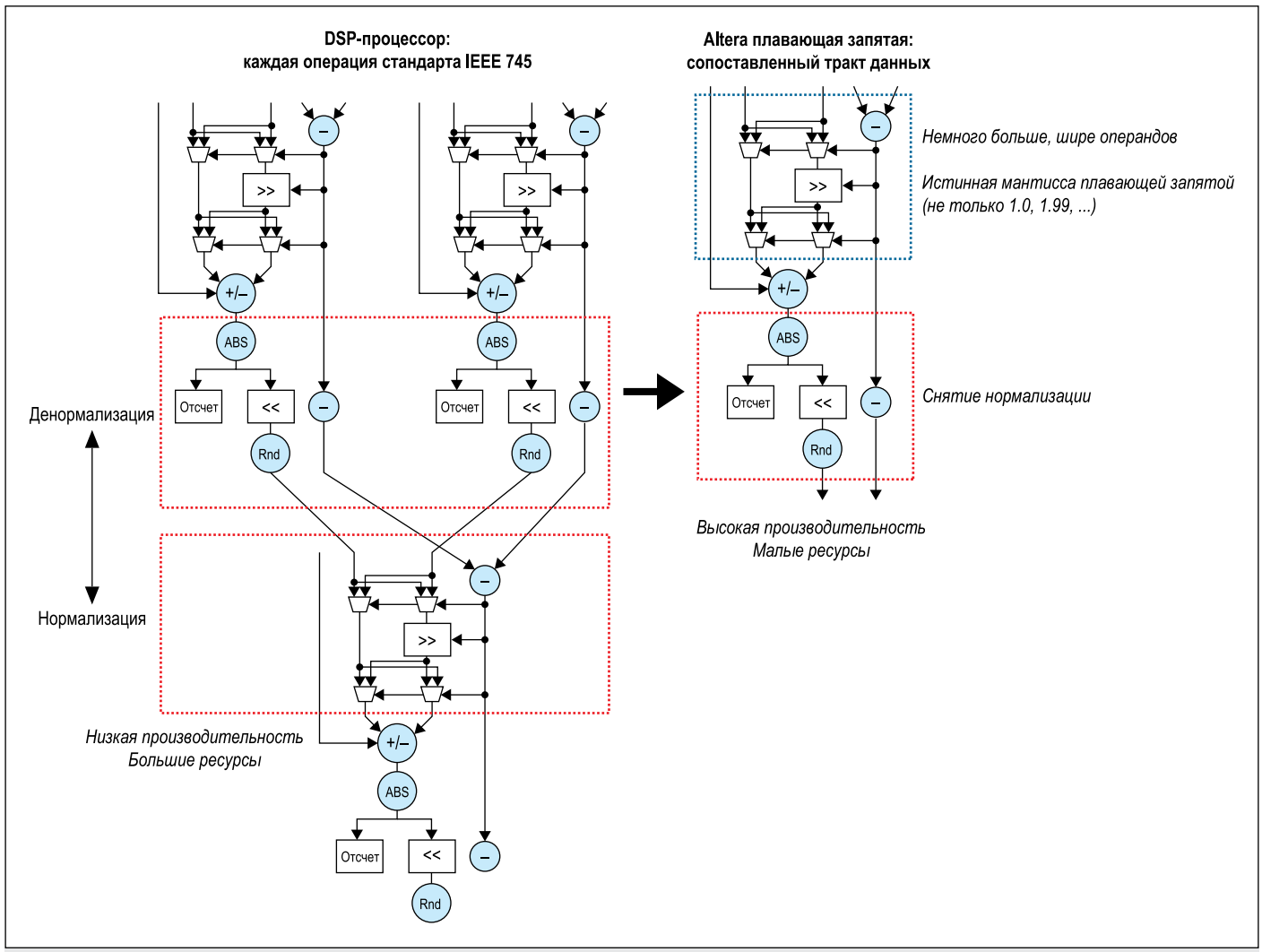

Рис. 3. Реализация сопоставленного тракта данных в операции с плавающей запятой

вого преобразования с понижением частоты на стадии обработки с плавающей запятой для сжатия импульса. Эффект Доплера, пространственно-временная адаптация, индикатор типа сообщения (MTI) движущейся цели и другие функции показаны на рис. 2. Таким образом, поток данных избегает узких мест процессора перед передачей в ускоритель ядра графического процессора, так что в целом задержка обработки уменьшается.

ПЛИС также может предложить гораздо более низкое время задержки обработки, чем ядро графического процессора, даже независимо от узких мест ввода/вывода. Хорошо известно, что ядра графических процессоров для эффективной работы должны обрабатывать многие тысячи потоков из-за чрезвычайно длинных задержек при вводе и выводе из памяти и даже между несколькими ядрами графического процессора. По сути, ядро графического процессора должно обработать множество операций, чтобы удержать ядра процессора от потери скорости обработки при ожидании данных, что приводит к очень длительному времени задержки для любой конкретной задачи.

ПЛИС использует вместо этого архитектуру «крупнозернистого параллелизма», с помощью которой создается множество оптимизированных и параллельных трактов данных, каждый из которых выдает один результат за один временной цикл. Количество трактов данных зависит от ресурсов ПЛИС, но, как правило, их гораздо меньше, чем количество ядер графического процессора. Тем не менее каждый тракт данных имеет гораздо более высокую пропускную способность, чем ядро графического процессора. Основное преимущество такого подхода заключается в низкой временной задержке, критическом значении для производительности во многих приложениях.

Еще одно преимущество ПЛИС состоит в гораздо более низком энергопотреблении, в результате чего резко снижено соотношение GFLOPs/Вт. Измерения производительности ПЛИС с использованием прототипной платы показывают 5–6 GFLOPs/Вт для таких алгоритмов, как алгоритм Холецкого и QR-разложение, и около 10 GFLOPs/Вт для простых алгоритмов, таких как быстрое преобразование Фурье. Измерения энергоэффективности графического процессора найти гораздо труднее, но с использованием ядра графического процессора с производительностью 50 GFLOPs с алгоритмом Холецкого и среднего потребления энергии в 200 Вт получается 0,25 GFLOPs/Вт, что в двадцать раз больше энергии, потребленной на полезный FLOPs.

Для бортовой самолетной РЛС или станции, установленной на автомобиле, размер, вес и потребляемая мощность (SWaP) имеют первостепенное значение. Можно легко представить себе радиолокационный пеленг беспилотников производительностью в десятки TFLOPs в будущих системах. Доступная величина вычислительной мощности коррелирует с допустимым разрешением и охватом современной РЛС.

Сопоставленный тракт данных

OpenCL и DSP Builder используют технологию, известную как «сопоставленный тракт данных» (рис. 3), где операции с плавающей запятой осуществляются в целях резкого сокращения количества требуемых многорегистровых схем циклического сдвига, что, в свою очередь, позволяет осущест-

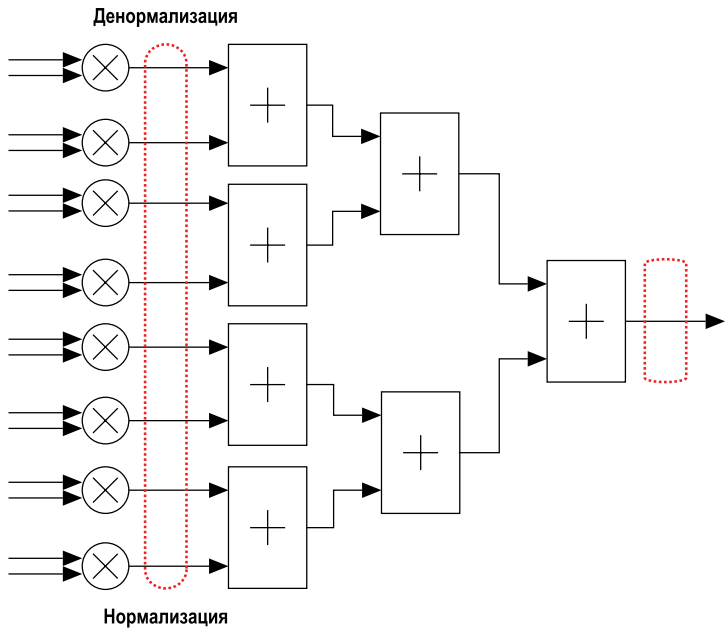

Рис. 4. Оптимизация скалярного произведения векторов

влять встраивание с использованием ПЛИС для крупномасштабного и высокопроизводительного программирования с плавающей запятой.

Для уменьшения частоты реализации многорегистровой схемы циклического сдвига процесс синтеза ищет возможности компенсации потребности в частой нормализации и денормализации с помощью большей ширины мантиссы. Возможность использования фиксированных множителей 27×27 и 36×36 допускает бóльшие множители, чем 23 бит, необходимые для реализаций одинарной точности, а также создание множителей 54×54 и 72×72 допускает бóльшие множители, чем 52 бита, необходимые для реализаций двойной точности. Для реализации больших схем суммирования с фиксированной запятой, с включением встроенных схем ускоренного переноса, логика ПЛИС оптимизируется.

Там, где требуется нормализация и денормализация, альтернативным способом является использование мультипликаторов, что позволяет избежать низкой производительности и чрезмерной маршрутизации. Для мантиссы одинарной точности в 24 бита (включая знаковый бит) множитель 24×24 сдвигает ввод путем умножения на 2*n*. Опять же, возможность использования фиксированных множителей 27×27 и 36×36 позволяет применять расширенные размеры мантиссы в реализациях одинарной точности и может быть использована для создания размеров множителя для реализаций двойной точности.

Скалярное произведение векторов является операцией, потребляющей основную часть производительности в FLOPs, и используется во многих алгоритмах линейной алгебры. Реализация скалярного произведения векторов одинарной точности длиной 64 потребует 64 множителя с плавающей запятой, с последующим суммированием на 63 сумматорах с плавающей запятой. Такая реализация потребует большого количества многорегистровых схем циклического сдвига.

Вместо этого выходы 64 множителей можно денормализовать до общего показателя, самого крупного из 64-х. Затем эти 64 выхода можно суммировать с помощью схемы суммирования с фиксированной запятой и окончательной нормализацией в конце. Эта операция с плавающей запятой в локализованном блоке обходится без промежуточной нормализации и денормализации, необходимой для каждого отдельного суммирования, и показана на рис. 4. Даже при операции с плавающей запятой стандарта IEEE 754 число с наибольшим показателем определяет показатель в конце, поэтому это изменение просто перемещает выравнивание показателя к более ранней точке в расчете.

Однако при обработке сигнала лучшие результаты получаются при наиболее точном выполнении округления результатов в конце расчета. Такой подход компенсируется путем использования наибольшей ширины бита мантиссы сверх того, что требуется при операции с плавающей запятой одинарной точности, как правило, 27–36 бит. Расширение мантиссы выполняется множителями с плавающей запятой, чтобы устранить необходимость нормализации произведения на каждом этапе.

Этот подход может также давать один результат за один временной цикл. Архитектуры ядра графического процессора могут производить все операции умножения с плавающей запятой параллельно, но не могут эффективно выполнять параллельное суммирование. Эта неспособность обусловлена требованиями того, что разные ядра должны передавать данные через локальную память (для связи друг с другом), тем самым уменьшая гибкость в соединениях архитектуры ПЛИС.

Подход «сопоставленный тракт данных» генерирует результаты, которые являются более точными, чем обычные результаты с плавающей запятой стандарта IEEE 754, как показано в таблице 3.

Таблица 3. Точность разложения Холецкого (одинарная точность)

Размер комплексной входной матрицы (n×n)	Векторное множество	Ошибка при использовании MATLAB. Настольный компьютер	Ошибка при использовании DSP Builder Advanced Blockset с генерацией RTL
360×360	50	$2{,}1112\times10^{-6}$	$1{,}1996\times10^{-6}$
60×60	100	$2{,}8577\times10^{-7}$	$1{,}3644\times10^{-7}$
30×30	100	$1{,}5488\times10^{-6}$	$9{,}0267\times10^{-8}$

Эти результаты были получены путем реализации обращения большой матрицы с использованием алгоритма разложения Холецкого. Этот же алгоритм был реализован тремя различными способами:

- в MATLAB+Simulink при операции с плавающей запятой одинарной точности стандарта IEEE 754;
- в RTL при операции с плавающей запятой одинарной точности, используя подход «сопоставленный тракт данных»;
- в MATLAB при операции с плавающей запятой двойной точности.

Реализация двойной точности точнее примерно в один миллиард раз (10^9), чем реализация одинарной точности.

Сравнение ошибок, полученных в MATLAB с одинарной точностью, в RTL с одинарной точностью и в MATLAB с двойной точностью, подтверждает достоверность подхода сопоставленного тракта данных. Этот подход продемонстрирован для нормализованной ошибки во всех сложных элементах в выходной матрице и матричного элемента с максимальной погрешностью. Суммарная ошибка или норма рассчитывается с использованием нормы Фробениуса:

$$\|E\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\left|e_{ij}\right|^2}.$$

Поскольку норма включает ошибки всех элементов, то она часто бывает много больше, чем отдельные ошибки.

Кроме того, инструменты потоков DSP Builder Advanced Blockset and OpenCL со всей очевидностью поддерживают и оптимизируют принятые методы расчета для архитектуры ПЛИС следующего поколения. Пиковую производительность в 100 GFLOPs/Вт можно ожидать благодаря архитектурным технологическим инновациям.

Заключение

Высокопроизводительные РЛС получили теперь новые процессорные платформы. В дополнение к значительно улучшенному SWaP, ПЛИС может обеспечить меньшее время ожидания и более высокую производительность в GFLOPs, чем решения на базе процессоров. Эти преимущества будут еще более явными с введением нового поколения ПЛИС, оптимизированного для высокопроизводительных вычислений.

Компилятор OpenCL корпорации Altera обеспечивает почти прямой путь программистам ядра графических процессоров для оценки достоинства этой новой архитектуры обработки сигнала. Язык программирования OpenCL 1.2 совместим с полным набором математической библиотеки поддержки. Это позволяет абстрагироваться от традиционных проблем ПЛИС по времени свертывания, управления памятью DDR и сопряжения с хост-процессором PCIe.

Для разработчиков безъядерных графических процессоров корпорация Altera предлагает инструмент потока DSP Builder Advanced Blockset, который позволяет разработчикам осуществлять DSP-программирование с высоким f_{max} и фиксированной или плавающей запятой, сохраняя при этом преимущества моделирования на основе Mathworks и среды разработки. Этот инструмент использовался в течение многих лет разработчиками РЛС с использованием ПЛИС для более производительного рабочего процесса и моделирования, который дает ту же производительность f_{max}, как и ручное программирование в коде HDL.

Литература

1. Получение одного TFLOPs на ПЛИС с технологией 28 нанометров. **www.altera.com/literature/wp/wp-01142-teraflops.pdf**
2. Depeng Yang, Junqing Sun, Jun Ku Lee, Getao Liang, David D. Jenkins, Gregory D. Peterson, and Husheng Li. Сравнение производительности разложения Холецкого на графическом процессоре и ПЛИС. **http://saahpc.ncsa.illinois.edu/10/papers/paper_45.pdf**
3. Реализация программирования ПЛИС языком OpenCL Standard. **www.altera.com/literature/wp/wp-01173-opencl.pdf**