УДК 004.415.2, 004.383

DOI: 10.15827/0236-235X.133.098-105

Дата подачи статьи: 18.01.21

2021. Т. 34. № 1. С. 098–105

Применение технологии высокоуровневого синтеза и аппаратных ускорителей вычислений на ПЛИС в задачах идентификации белков

Г.К. Шмелев¹, *заведующий отделом, shmelevgk@cps.tver.ru*

М.А. Лихачев¹, *заведующий отделом, likhachevma@cps.tver.ru*

В.И. Аржаев¹, *к.т.н., заместитель генерального директора по ИТС,
arzhaevvi@cps.tver.ru*

¹ НИИ «Центр программ систем», г. Тверь, 170024, Россия

В работе рассмотрено применение технологии высокоуровневого синтеза с использованием аппаратных ускорителей на базе программируемых логических интегральных схем в задаче идентификации белков.

В настоящее время существует значительное количество аппаратных решений с высокой производительностью и пропускной способностью, предназначенных для решения различных прикладных задач. Одним из таких решений являются аппаратные ускорители вычислений, построенные на базе программируемых логических интегральных схем, которые обладают рядом преимуществ в сравнении с ускорителями, построенными на графических процессорах, а также на интегральных схемах специального назначения. Однако существует определенная сложность, препятствующая широкому применению подобных устройств, которая заключается в трудоемкости и специфичности традиционного пути разработки приложений с применением специализированных языков программирования под этот тип ускорителей. Использование технологии высокоуровневого синтеза с применением одного из популярных языков программирования открывает новые горизонты для широкого применения подобного рода ускорителей.

В данной работе описывается один из вариантов построения вычислительной аппаратно-программной платформы с применением ускорителя вычислений на программируемой логике. Особое внимание уделяется основным шагам разработки архитектуры приложений, разворачиваемых на аппаратных средствах, и методологии разработки высокопроизводительного вычислительного ядра аппаратно ускоряемых программных функций. Демонстрируются результаты повышения вычислительной производительности программного приложения *de novo* секвенирования пептидных последовательностей, а также эффективность применяемой аппаратно-программной платформы и выбранного пути разработки в сравнении с исходным программным приложением.

Ключевые слова: *высокоуровневый синтез, ускорители вычислений, ПЛИС, идентификация белков, de novo секвенирование.*

При решении значительной части задач биоинформатики возникает проблема оптимизации реализаций алгоритмов и вычислительных процедур по времени выполнения и объему требуемых вычислительных ресурсов.

Например, в задачах попарного выравнивания биологических последовательностей применяются различные алгоритмы динамического программирования, которые требуют значительных вычислительных ресурсов даже при ограниченной размерности задачи. Это же касается и задач определения аминокислотных последовательностей пептидов и белков без использования поисковых программ и БД (*de novo* секвенирование). Метод множественного локального выравнивания последовательностей, являющийся основой алгоритмов поиска сходных последовательностей в больших БД, тоже

относится к задачам динамического программирования и имеет слишком большую временную и пространственную сложность. В связи с постоянным ростом объемов БД и сложности алгоритмов построения биологических последовательностей требования к вычислительной мощности компьютерных систем существенно расширяются.

К настоящему времени создан значительный арсенал аппаратных решений для обеспечения вычислений с высокой пропускной способностью, а также программных библиотек, предоставляющих интерфейсы прикладного программирования для использования на различных аппаратных платформах при выполнении прикладных задач. Параллельные вычисления стали доминирующей парадигмой в архитектуре компьютеров.

В силу этого универсальным способом повышения производительности вычислений является использование параллельных алгоритмов, реализованных и выполняемых на параллельных вычислительных системах, построенных на базе гетерогенных вычислителей [1]. В таких вычислительных системах обычные центральные процессоры (CPU) дополняются специализированными ускорителями, повышающими производительность и энергоэффективность решения различных ресурсоемких задач, уже привычных в современном мире. В качестве таких ускорителей вычислений, помимо графических процессоров (GPU) и специализированных *интегральных схем специального назначения* (ASIC), все более широкое применение находят ускорители вычислений, построенные на базе *программируемых логических интегральных схем* (ПЛИС). К основным преимуществам ускорителей на ПЛИС в отличие от ускорителей, основанных на CPU или GPU, следует отнести более высокую производительность в приложениях реального времени, сокращение задержек вывода машинного обучения, лучшее соотношение производительности на ватт потребляемой мощности, а также высокий уровень гибкости в сравнении с ASIC. В работах [2, 3] демонстрируется эффективность применения ПЛИС в решении ряда задач биоинформатики.

В качестве основы ускорителей на ПЛИС используются микросхемы программируемой логики типа FPGA. Вычислительная система, включающая в свой состав ускорители на FPGA, – это, как правило, классическая серверная платформа, содержащая процессоры и несколько карт ускорителей на FPGA, либо специализированные промышленные серверы. В таких системах ускоритель принимает на себя выполнение основных ресурсоемких задач, в которых требуются параллельные вычисления, повышающие производительность всей системы в целом.

Технология ПЛИС обеспечивает создание гибко конфигурируемых цифровых электронных схем. Аппаратные ресурсы микросхемы FPGA программируются пользователем непосредственно под саму задачу, что позволяет реализовать пользовательскую программу как имплементацию алгоритма в кремнии, используя как базовые элементы (триггеры, логические элементы «И», «ИЛИ», «НЕ» и пр.), так и специализированные аппаратные ядра (умножители, блочную память, коммуникационные интерфейсы и пр.), и достичь тем самым высо-

кого быстродействия за счет высокого параллелизма и тактовых частот.

Традиционный путь разработки под FPGA предполагает написание программного кода каждого из модулей проектируемого устройства с использованием языков описания аппаратуры интегральных схем (HDL), таких как VHDL и Verilog, применение готовых стандартных программных и аппаратных модулей (IP-ядер), входящих в состав САПР. Используя эти инструменты, разработчик вынужден тратить значительную часть времени на написание и отладку низкоуровневого поведенческого программного кода каждого функционального модуля проекта, а также на создание и настройку всей периферии, необходимой для интеграции разработанного программного кода и обеспечения взаимодействия с внешними периферийными устройствами. Данный путь оправдан в случае выполнения несложных задач, требующих небольших по объему микросхем FPGA. Современные микросхемы программируемой логики последних поколений обладают огромным количеством предоставляемых пользователю аппаратных ресурсов и предназначены для решения ресурсоемких задач и построения сложных программных алгоритмов, например, таких, как секвенирование генов [4].

Таким образом, традиционный путь разработки неоптимален или сложно реализуем в связи с необходимостью разработки на языках низкого уровня, что требует значительного времени на написание и отладку программных алгоритмов.

В современных реалиях традиционный путь разработки уходит на второй план и становится лишь вспомогательным. На первый план выходит новая парадигма – высокоуровневый синтез (HLS), она заключается в разработке программных алгоритмов на языках высокого уровня, таких как C, C++ и OpenCL [4, 5], что сильно упрощает и ускоряет разработку и отладку программного кода. Компилятор HLS преобразует программный код в синхронную цифровую схему на уровне регистровых передач (RTL-дизайн), который впоследствии будет исполняться на FPGA. Надо отметить, что компилятор имеет ряд ограничений по сравнению со стандартными компиляторами кода C/C++, такие как отсутствие системных вызовов, динамически выделяемой памяти, рекурсивных функций. Также стоит учитывать, что пропускная способность полученного таким образом программного модуля сильно зависит

от используемых в процессе его написания конструкций и подходов в коде. На рисунке 1 представлена обобщенная схема, описывающая основные шаги разработки архитектуры приложений, разворачиваемых на программно-аппаратных средствах.

Рассмотрим применение технологии HLS на примере решения задач биоинформатики, в частности, портирования и повышения вычислительной производительности алгоритма PepNovo, реализованного в программном приложении PepNovo+. За основу взята методология разработки, описанная в руководстве [6].

Программное приложение PepNovo+ осуществляет de novo пептидное секвенирование, в котором пептидная аминокислотная последовательность определяется по результатам tandemной масс-спектрометрии. PepNovo – алгоритм, основанный на динамическом программировании и предназначенный для восстановления аминокислотной последовательности путем поиска антисимметричных путей в графе спектра. Алгоритм проверяет две гипотезы: суть первой в том, что масса фрагмента образована при фрагментации пептида, которому соответствует исследуемый спектр; второй – все пики в спектре были получены в результате случайного процесса. В соответствии с первой гипотезой могут быть описаны правила фраг-

ментации пептида. В результате каждой массе фрагмента ставится в соответствие величина, равная логарифму отношения правдоподобия этих двух гипотез. Для каждой вершины вычисляют несколько значений, которым соответствуют различные комбинации аминокислот. В итоге учитываются значения, рассчитанные для каждой величины.

Временной и функциональный анализ программного кода приложения PepNovo+ выявил ряд функций, наиболее пригодных для размещения на аппаратном ускорителе на базе ПЛИС. В результате анализа принято решение разбить приложение PepNovo+ на две функциональные составляющие, расположенные на аппаратном модуле, на базе CPU и на ПЛИС. В основе аппаратной платформы для решения поставленной задачи использовались вычислительные модули со следующими характеристиками:

- процессорный модуль на базе процессора Intel Xeon D-1548 (тактовая частота 2 ГГц, 8 вычислительных ядер, 16 потоков), объем оперативной памяти составляет 16 Гб DDR4;
- модуль цифровой обработки на базе ПЛИС Xilinx Kintex UltraScale XCKU085T (1,088 млн логических ячеек и 4 100 аппаратных умножителей), объем оперативной памяти составляет 2 Гб DDR3.

Структурная схема программно-аппаратной платформы, отражающая основные компоненты и внутренние связи, представлена на рисунке 2.

Разработка программных компонентов, исполняемых на аппаратном ускорителе, осуществлялась в средах Xilinx Vivado и Vivado HLS версии 2019.1.

Оценка производительности и определение целевых показателей исходного приложения PepNovo+ осуществлялись на ЭВМ с центральным процессором Intel Core i5 9400 и 16 Гб оперативной памяти DDR4 2400, под управлением ОС Debian 10. Выполнялась данная процедура с применением профайлера функций, общее время определялось миллисекундным таймером. При работе с приложением использовались результаты tandemной масс-спектрометрии образца клеточной линии, полученного из эмбриональных почек человека, содержащие 90 887 масс-спектров, время обработки которых составило 2 часа 38 минут и 53 секунды при работе приложения в однопоточном режиме.

Определение пропускной способности приложения PepNovo+ определялось по формуле

Рис. 2. Структурная схема программно-аппаратной платформы

Fig. 2. Hardware and software platform block diagram

$$T_{sw} = \frac{\max(V_{INPUT}, V_{OUTPUT})}{RunningTime},$$

где V_{INPUT} – объем входных данных; V_{OUTPUT} – объем выходных данных, $Running\ Time$ – время выполнения (сек.).

Для неоптимизированного приложения PepNovo+, работающего в однопоточном режиме, уровень пропускной способности измерялся в спектрах в секунду. Используя результаты времени обработки, получаем 9,53 спектра в секунду. Исходя из полученных результатов принято решение увеличить максимальную производительность аппаратно-ускоряемых функций приложения PepNovo+ в 100 раз, что соответствует возможности обработки 953 спектров в секунду.

В качестве ускоряемой была выбрана функция, генерирующая пептидные последовательности из сгенерированного графа, используя алгоритмы обхода в ширину, обхода в глубину с методом ветвления и границ. Данная функция идеально подходит для ускорения на FPGA, поскольку зависит только от значений оценок стоимости вершин и ребер входного графа.

Расчет пропускной способности программного ядра, разрабатываемого на основе описанной выше функции, осуществлялся по формуле

$$T_{hw} = \frac{Frequency}{SampleRate},$$

где $Frequency$ – частота тактирующего сигнала ядра; $SampleRate$ – временной интервал, измеряемый в циклах тактирующего сигнала.

Требуемый параллелизм ядра определялся по формуле

$$ParallelismNeeded = \frac{T_{GOAL}}{T_{HW}},$$

где T_{GOAL} – требуемая пропускная способность; T_{HW} – расчетная пропускная способность ядра.

Был выбран требуемый параллелизм для ускоряемой функции, равный 1, значит, если принять требуемую пропускную способность T_{GOAL} равной 953 спектрам в секунду, то расчетная пропускная способность ядра T_{HW} также должна равняться 953 спектрам в секунду. Если частоту ядра принять равной 200 MHz, тогда время обработки одного спектра составит 1.05 мс, а $SampleRate$ – 209 863 цикла тактирующего сигнала.

Как было сказано выше, не все выделенные функции ускоряются с применением ускорителя на базе FPGA. Для достижения целевых показателей пропускной способности требуется обеспечить Software-параллелизм. Тогда, если программно ускорить вычисления по генерации пептидных последовательностей, про-

пускная способность программы улучшится до 36 спектров/сек. Достижение показателя в 953 спектра в секунду для программной части приложения PepNovo+ достижимо весьма условно и требует многопоточной реализации и оптимизации вычислений с применением OpenMP [7].

Методология разработки вычислительного ядра функции, генерирующей пептидные последовательности из сгенерированного графа с применением технологии HLS, представлена на рисунке 3.

В рамках данной методологии разработан протокол обмена между программной и аппаратной ускоряемой частями приложения PepNovo+. Для его реализации использован паттерн «загрузить–вычислить–сохранить», включающий три основные функции: загрузку и разбор входных данных по протоколу обмена, обработку входных данных, сохранение и запаковку результатов обработки по протоколу обмена.

Второй шаг предполагает разбиение основного блока обработки на отдельные функции. Основной алгоритм функции генерации пептидных последовательностей из сгенерированного графа разбивается на три блока:

- поиск в ширину;
- поиск в глубину с ветвлением и границами;

- постобработка пептидной последовательности.

В ходе доработки программного кода портируемой на ускоритель функции применялись следующие подходы к организации оптимизации вычислений:

- уменьшение числа итераций циклов;
- уменьшение длительности итераций циклов;
- уменьшение интервала перекрытия конвейера обработки.

Уменьшение числа итераций осуществлялось с помощью директивы HLS UNROLL, которая приводит к раскрытию исходного цикла (рис. 4а) и одновременному выполнению большего числа операций за один период тактового сигнала (рис. 4б).

Раскрытие программных циклов накладывает определенные требования к обеспечению поддержки нескольких одновременно выпол-

няемых операций ввода/вывода или доступа к переменным. Подобная задача решалась посредством директивы HLS ARRAY_PARTITION, осуществляющей разбиение массива на более мелкие массивы или отдельные элементы и обеспечивающей повышение пропускной способности и увеличение количества портов чтения и записи.

Интервал перекрытия исходного конвейера обработки (рис. 5а) конфигурировался с помощью директивы HLS PIPELINE, обеспечивающей одновременное выполнение операций (рис. 5б).

Результатом компиляции в среде разработки Vivado HLS доработанного и оптимизированного исходного программного кода приложения PepNovo+ по описанной выше методологии является вычислительное ядро функции, генерирующей пептидные последовательности из сгенерированного графа. При частоте вычислительного ядра в 200 MHz пропускная способность неоптимизированной

версии ядра составила 75 спектров в секунду, а оптимизированной – около 900 спектров в секунду. Целевая пропускная способность вычислительного ядра выбиралась с учетом входных данных, содержащих 90 вершин на граф. Пропускная способность программной составляющей приложения PepNovo+ в многопоточном исполнении позволила повысить пропускную способность приложения до 187 спектров в секунду, что значительно ограничивает потенциал аппаратного ускорителя.

В заключение можно сделать следующие выводы. Результаты портирования на аппаратный ускоритель ряда функций приложения de novo секвенирования PepNovo+ с применением технологии HLS показали значительное повышение вычислительной производительности функций, которая в отдельных случаях достигает 100 раз. Однако, несмотря на аппаратное ускорение отдельных функций, общая вычислительная производительность оптимизированного приложения PepNovo+, развернутого на программно-аппаратной платформе, ограничена программной составляющей, располагающейся на процессорном модуле, и зависит от входных данных, а именно от количества вершин в графе.

Стоит отметить влияние на общую вычислительную производительность пропускной способности и типа используемой на ускорителе внешней памяти [5, 8], применение оптимальных стратегий оптимизации в среде разработки Vivado 2019.1, расположение и трассировку разработанных вычислительных ядер непосредственно в кристалле FPGA [9], а также необходимость сбалансированного подхода в отношении пропускной способности и масштабирования вычислительных ядер по причине ограниченности внутренних ресурсов FPGA.

Представляет значительный интерес построение более сложных гетерогенных аппаратных платформ, включающих в себя, помимо процессорных модулей, несколько ускорителей вычислений на основе FPGA, а также с применением графических процессоров [10], что является темой дальнейших работ.

Литература

1. Reich O., Ozkan M.A., Hannig F., Teich J., Schmid M. Loop parallelization techniques for FPGA accelerator synthesis. Journal of Signal Processing Systems, 2018, vol. 90, no. 1, pp. 3–27. DOI: 10.1007/s11265-017-1229-7.
2. Oliver T., Schmidt B., Nathan D., Clemens R., Maskell D. Using reconfigurable hardware to accelerate multiple sequence alignment with ClustalW. Bioinformatics, 2005, vol. 21, pp. 3431–3432. DOI: 10.1093/bioinformatics/bti508.

3. Bogaraju V.C.S., Paul K., Lavenier D., Balakrishnan M. Hardware acceleration of de novo genome assembly. *IJES*, 2017, vol. 9, no. 1, pp. 74–89. DOI: 10.1504/IJES.2017.081729.
4. Rauer C., Powley G.S., Ahsan M., Finamore N. Accelerating Genomics Research with OpenCL and FPGAs. URL: <https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/wp/wp-accelerating-genomics-opencl-fpgas.pdf> (дата обращения: 13.01.2021).
5. Fang J., Mulder Y.T.B., Hidders J., Lee J., Hofstee P.H. In-memory database acceleration on FPGAs: a survey. *The VLDB Journal*, 2020, vol. 29, pp. 33–59. DOI: 10.1007/s00778-019-00581-w.
6. Xilinx: SDAccel Methodology Guide UG1346 (v2018.3). April 03, 2019. URL: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_3/ug1346-sdaccel-methodology-guide.pdf (дата обращения: 13.01.2021).
7. Горобец А.В., Суков А.С., Железняков А.О., Богданов П.Б., Четверушкин Б.Н. Расширение двухуровневого распараллеливания MPI+OpenMP посредством OpenCL для газодинамических расчетов на гетерогенных системах // Вестн. ЮУрГУ. Сер.: Матем. моделирование и программирование. 2011. Т. 25. С. 76–86.
8. Cadenelli N., Jaksic Z., Polo J., Carrera D. Considerations in using OpenCL on GPUs and FPGAs for throughput-oriented genomics workloads. *Future Generation Computer Systems*, 2019, vol. 94, pp. 148–159. DOI: 10.1016/j.future.2018.11.028.
9. Farooq F., Parvez H., Mehrez H., Marrakchi Z. Exploration of heterogeneous FPGA architectures. *International Journal of Reconfigurable Computing*, 2011, vol. 2011, pp. 1–18. DOI: 10.1155/2011/121404.
10. Inta R., Bowman D.J., Scott S.M. The “Chimera”: An Off-The-Shelf CPU/GPGPU/FPGA Hybrid Computing Platform. *International Journal of Reconfigurable Computing*, 2012, vol. 2012, pp. 1–10. DOI: 10.1155/2012/241439.

Software & Systems
DOI: 10.15827/0236-235X.133.098-105

Received 18.01.21
2021, vol. 34, no. 1, pp. 098–105

Application of high-level synthesis technology and hardware accelerators on FPGA in protein identifications

G.K. Shmelev¹, Chief of Department, shmelevgk@cps.tver.ru

M.A. Likhachev¹, Chief of Department, likhachevma@cps.tver.ru

V.I. Arzhaev¹, Ph.D. (Engineering), Deputy General Director for Intelligent Systems, arzhaevvi@cps.tver.ru

¹ R&D Institute Centerprograms, Tver, 170024, Russian Federation

Abstract. The paper considers the use of high-level synthesis technology using hardware accelerators based on FPGA in the identifying proteins problem.

Currently, there are a significant number of hardware solutions with high performance and bandwidth designed to solve various applications. One such solution is hardware-based computation accelerators based on field-programmable gate array (FPGA), which have a number of advantages over accelerators built on both graphics processing unit (GPU) and application-specific integrated circuit (ASIC). However, there is a certain complexity in the wide application of such devices, which consists in the laboriousness and specificity of the traditional way of developing applications using specialized programming languages for this type of accelerator. Using high-level synthesis technology using one of the popular programming languages opens up new horizons in the wide use of such accelerators.

This paper describes one embodiment of a computational hardware and software platform using a hardware accelerator on a FPGA. Special attention is paid to considering the major steps of developing the architecture of applications deployed on hardware and the methodology for developing a high-performance computing core of hardware-accelerated software functions. The results of improving the computational performance of the de novo peptide sequence sequencing software application and the effectiveness of the used hardware platform and the chosen development path in comparison with the original software application are demonstrated.

Keywords: high-level synthesis, hardware acceleration, FPGA, protein identification, de novo sequencing.

References

1. Reich O., Ozkan M.A., Hannig F., Teich J., Schmid M. Loop parallelization techniques for FPGA accelerator synthesis. *Journal of Signal Processing Systems*, 2018, vol. 90, no. 1, pp. 3–27. DOI: 10.1007/s11265-017-1229-7.

2. Oliver T., Schmidt B., Nathan D., Clemens R., Maskell D. Using reconfigurable hardware to accelerate multiple sequence alignment with ClustalW. *Bioinformatics*, 2005, vol. 21, pp. 3431–3432. DOI: 10.1093/bioinformatics/bti508.
3. Bogaraju V.C.S., Paul K., Lavenier D., Balakrishnan M. Hardware acceleration of de novo genome assembly. *IJES*, 2017, vol. 9, no. 1, pp. 74–89. DOI: 10.1504/IJES.2017.081729.
4. Rauer C., Powley G.S., Ahsan M., Finamore N. *Accelerating Genomics Research with OpenCL and FPGAs*. Available at: <https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/wp/wp-accelerating-genomics-opencl-fpgas.pdf> (accessed January 13, 2021).
5. Fang J., Mulder Y.T.B., Hidders J., Lee J., Hofstee P.H. In-memory database acceleration on FPGAs: a survey. *The VLDB Journal*, 2020, vol. 29, pp. 33–59. DOI: 10.1007/s00778-019-00581-w.
6. Xilinx: *SDAccel Methodology Guide UG1346* (v2018.3). April 03, 2019. Available at: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_3/ug1346-sdaccel-methodology-guide.pdf (accessed January 13, 2021).
7. Gorobets A.V., Soukov S.A., Zheleznyakov A.O., Bogdanov P.B., Chetverushkin B.N. Extension with OpenCL of the two-level MPI+OpenMP parallelization for CFD simulations on heterogeneous systems. *Vestn. YuUrGU. Ser. Mat. Model. Progr.*, 2011, vol. 25, pp. 76–86 (in Russ.).
8. Cadenelli N., Jaksic Z., Polo J., Carrera D. Considerations in using OpenCL on GPUs and FPGAs for throughput-oriented genomics workloads. *Future Generation Computer Systems*, 2019, vol. 94, pp. 148–159. DOI: 10.1016/j.future.2018.11.028.
9. Farooq F., Parvez H., Mehrez H., Marrakchi Z. Exploration of heterogeneous FPGA architectures. *International Journal of Reconfigurable Computing*, 2011, vol. 2011, pp. 1–18. DOI: 10.1155/2011/121404.
10. Inta R., Bowman D.J., Scott S.M. The “Chimera”: An Off-The-Shelf CPU/GPGPU/FPGA Hybrid Computing Platform. *International Journal of Reconfigurable Computing*, 2012, vol. 2012, pp. 1–10. DOI: 10.1155/2012/241439.

Для цитирования

Шмелев Г.К., Лихачев М.А., Аржаев В.И. Применение технологии высокоуровневого синтеза и аппаратных ускорителей вычислений на ПЛИС в задачах идентификации белков // Программные продукты и системы. 2021. Т. 34. № 1. С. 098–105. DOI: 10.15827/0236-235X.133.098-105.

For citation

Shmelev G.K., Likhachev M.A., Arzhaev V.I. Application of high-level synthesis technology and hardware accelerators on FPGA in protein identifications. *Software & Systems*, 2021, vol. 34, no. 1, pp. 098–105 (in Russ.). DOI: 10.15827/0236-235X.133.098-105.