Информатика, телекоммуникации и управление. 2022. Том 15, № 4. С. 51–63.
Computing, Telecommunication and Control, 2022, Vol. 15, No. 4, Pp. 51–63.

Научная статья
DOI: https://doi.org/10.18721/JCSTCS.15404
УДК 004.312.44

ИССЛЕДОВАНИЕ И СРАВНИТЕЛЬНЫЙ АНАЛИЗ ЭФФЕКТИВНОСТИ ПРОГРАММНОЙ И АППАРАТНЫХ РЕАЛИЗАЦИЙ ОПЕРАЦИИ СУММИРОВАНИЯ ТРАНСПОНИРОВАННЫХ МАТРИЦ

А.П. Антонов[1] ✉ , *Д.С. Беседин*[2], *А.С. Филиппов*[3]

[1,2,3] Санкт-Петербургский политехнический университет Петра Великого, Санкт-Петербург, Российская Федерация

✉ antonov@eda-lab.ftk.spbstu.ru

Аннотация. Статья посвящена исследованию и сравнительному анализу программной и аппаратной реализации операции суммирования транспонированных матриц и её модифицированного варианта: операции транспонирования суммы матриц. Особенностью исследования является использование для получения аппаратной реализации средств высокоуровневого синтеза. Актуальность исследования обусловлена широким использованием матричных операций для решения задач различных классов, степенной асимптотической сложностью матричных вычислений и отсутствием данных об использовании данного инструментария в задачах создания аппаратных устройств для матричных вычислений. Предложен пошаговый метод синтеза и оптимизации аппаратного устройства. Проведено сравнительное исследование программных и аппаратных реализаций двух вычислительных задач. Показано, что большой выигрыш производительности аппаратных реализаций получается за счет увеличения степени параллелизма вычислений. Дополнительно сделаны выводы о неэффективности попыток достичь высоких тактовых частот, а также об увеличении затрачиваемых ресурсов при увеличении быстродействия за счет распараллеливания.

Ключевые слова: аппаратная реализация, производительность, аппаратные затраты, FPGA, параллельные вычисления, конвейеризация

Финансирование: Работа выполнена при поддержке Госзадания СПбПУ FSEG-2022-0001.

Для цитирования: Антонов А.П., Беседин Д.С., Филиппов А.С. Исследование и сравнительный анализ эффективности программной и аппаратных реализаций операции суммирования транспонированных матриц // Computing, Telecommunications and Control. 2022. Т. 15, № 4. С. 51–63. DOI: 10.18721/JCSTCS.15404

Research article
DOI: https://doi.org/10.18721/JCSTCS.15404
UDC 004.312.44

RESEARCH AND COMPARATIVE ANALYSIS OF THE EFFECTIVENESS OF SOFTWARE AND HARDWARE IMPLEMENTATIONS OF THE OPERATION OF SUMMING TRANSPOSED MATRICES

A.P. Antonov[1] ✉ , *D.S. Besedin*[2], *A.S. Filippov*[3]

[1,2,3] Peter the Great St. Petersburg Polytechnic University,
St. Petersburg, Russian Federation

✉ antonov@eda-lab.ftk.spbstu.ru

Abstract. The article is devoted to the study and comparative analysis of the software and hardware implementation of the operation of summing transposed matrices and its modified version – the operation of transposing the sum of matrices. A feature of the study is the use of high-level synthesis tools to obtain a hardware implementation. The relevance of the study is due to the widespread use of matrix operations for solving problems of various classes, the power asymptotic complexity of matrix calculations and the lack of data on the use of this toolkit in the tasks of creating hardware devices for matrix calculations. A step-by-step method of synthesis and optimization of a hardware device is proposed. A comparative study of software and hardware implementations of two computational tasks is carried out. It is shown that a large gain in the performance of hardware implementations is obtained by increasing the degree of parallelism of calculations. Additionally, conclusions are drawn about the inefficiency of attempts to achieve high clock frequencies, as well as about the increase in resources spent with increased speed due to parallelization.

Keywords: hardware implementation, performance, hardware costs, FPGA, parallel computing, pipelining

Funding: The work was carried out with the support of the Goszadanie of SPbPU FSEG-2022-0001.

Citation: Antonov A.P., Besedin D.S., Filippov A.S. Research and comparative analysis of the effectiveness of software and hardware implementations of the operation of summing transposed matrices. Computing, Telecommunications and Control, 2022, Vol. 15, No. 4, Pp. 51–63. DOI: 10.18721/JCSTCS.15404

Введение

Матричные вычисления широко используются для решения задач различных классов. При этом асимптотическая сложность матричных вычислений имеет степенную зависимость от числа строк/столбцов обрабатываемых матриц, например, в общем случае: $O(n^3)$ – для умножения матриц, $O(n^2)$ – для сложения матриц. Поэтому задача повышения производительности вычислительных систем при выполнении базовых матричных операций актуальна [1, 2].

Матричные вычисления – это тип вычислений, позволяющий осуществлять пространственное распараллеливание вычислений, что является следствием того, что вычисления сводятся к операциям с независимыми, с точки зрения вычислительного процесса, элементами исходных матриц. Кроме того, отсутствие зависимости по данным для большинства базовых матричных операций позволяет использовать конвейеризацию (распараллеливание во времени), обеспечивающую параллельное (одновременное) выполнение операций считывания элементов матриц, арифметических действий над ними и записи результата одновременно для P элементов результирующей матрицы.

Очевидно, что теоретически пространственное распараллеливание вычислений может дать кратный степени распараллеливания рост производительности. Соответственно, конвейеризация – кратный коэффициенту конвейеризации, т. е. числу реализованных этапов конвейера. Однако такие теоретические оценки являются излишне оптимистичными, поскольку не учитывают ограничения, связанные с пропускной способностью подсистем, обеспечивающих считывание/запись данных, т. е. считывание и запись элементов матриц, а также с доступными логическими ресурсами для реализации арифметических операций над элементами матриц [3].

Средства и подходы, традиционно используемые для повышения производительности вычислений, основанных на матричных операциях, хорошо известны. Это и многопоточное, многоядерное и/или многопроцессорное пространственное распараллеливание, реализуемое на традиционных процессорах, имеющих N физических ядер и позволяющих реализовать до N*2 потоков. Это и графические карты, работающие в режиме вычислителя, в англоязычной литературе называемом General Purpose Graphic Processing Unit (GPGPU), и имеющие систолическую SIMD (Single Instruction Multiple Data) – одна инструкция много данных – архитектуру. Это и FPGA (Field Programmable Gate Arrays) – СБИС (Сверхбольшие Интегральные Схемы) с внутренней архитектурой, аппаратно реконфигурируемой под выполняемый алгоритм [4–7].

В вычислителях с фиксированной архитектурой (многоядерные/многопотоковые процессоры и GPGPU) возможности повышения производительности ограничены особенностями конкретного вычислителя: числом вычислительных блоков; существующими связями между вычислительными блоками и их быстродействием; объёмом локальной/распределенной памяти, шириной каналов доступа к памяти; быстродействием и производительностью памяти. Эти ограничения являются фиксированными для конкретного вычислителя с фиксированной архитектурой и определяют зависимость между производительностью вычислителя и типом решаемой на нем задачи.

Вычислители с реконфигурируемой внутренней архитектурой, подстраиваемой под задачу, традиционно реализуемые на FPGA, во многом лишены указанных выше недостатков фиксированных архитектур. Однако одним из недостатков использования ускорителей вычислений на FPGA как для матричных операций, так и для других задач, является сложность традиционной процедуры разработки, основанной как на схемном описании, так и на использовании языков описания аппаратуры, например, языков VHDL, Verilog HDL, System Verilog [8–11].

Процесс создания вычислителя, аппаратно оптимизированного под алгоритм решаемой задачи, трудоёмок и требует существенных временных затрат как на этапе его разработки, так и на этапе его отладки [5]. Это зачастую не позволяет провести исследование и сравнительный анализ разных вариантов аппаратных реализаций алгоритма решаемой задачи. Что приводит к аппаратным решениям по производительности близким к решениям на основе вычислителей с фиксированными архитектурами [12, 13].

К настоящему времени существует ряд работ, связанных с созданием аппаратных реализаций вычислительно сложных алгоритмов и демонстрирующих указанные выше особенности и недостатки.

Так, в [3] описана попытка использования FPGA для выполнения операций с матрицами и векторами и сравнение эффективности с реализациями на базе Цифровых Сигнальных Процессоров (DSP), GPGPU и Специализированных Интегральных Схем (ASIC). Сделан вывод о применимости FPGA для решения подобных вычислительных задач, а основным недостатком использования FPGA названа сложность разработки устройства с использованием языков описания аппаратуры HDL. При этом в работе, в силу заявленной сложности создания вариантов аппаратных реализаций, не проведены исследования и оптимизация аппаратных реализаций.

В [4] сравнивается производительность FPGA и GPGPU при решении задачи умножения разреженной матрицы на вектор. Для GPGPU используется решение, основанное на технологии

CUDA. В работе приведена сравнительная таблица производительности решения поставленной задачи на GPGPU и на базе FPGA, показывающая, что предложенная аппаратная реализация алгоритма умножения разреженной матрицы на вектор несколько производительнее реализации на GPGPU.

В [5] представлено сравнение производительности GPGPU, многоядерных систем и аппаратной реализации на FPGA при решении задачи перемножения матриц. Показано, что максимальная производительность достигается для решений на базе GPGPU и FPGA. При этом FPGA показали лучший результат по критерию энергоэффективности.

В [6] рассмотрены две реализации одного алгоритма умножения матриц на FPGA, созданные с использованием традиционного подхода к проектированию, включающему использование языков описания аппаратуры и схемного ввода. В статье показано, что полученные аппаратные реализации по производительности сравнимы с реализациями аналогичных алгоритмов на многоядерных системах.

В работе [7] предложена конвейерная архитектура матричного умножения на FPGA, созданная на базе схемного ввода. Особое внимание уделено оценке производительности передачи данных между подмодулями устройства. Оценка производительности проводилась в сравнении с оценкой производительности на базе моделирования в пакете MATLAB. В результате показано, что созданная реализация на FPGA имеет схожую производительность.

Современным подходом при создании аппаратной реализации алгоритма решаемой задачи является использование возможностей средств высокоуровневого синтеза – синтеза аппаратных решений описаний, созданных на высокоуровневых языках программирования, обычно на языках Си и C++. Подобные средства предоставляют как ведущие производители FPGA, такие как компания Xilinx и Intel PSG, так и компании, занимающиеся созданием средств разработки электронных устройств, например, компания Mentor Graphics. Подобные средства позволяют не только создать некоторое, базовое, не оптимизированное аппаратное решение на базе существующего описания на Си (или C++), но и провести исследование и сравнительный анализ различных вариантов аппаратных решений, отличающихся степенью параллелизма, количеством ступеней конвейера, интерфейсом к памяти данных, структурной организацией памяти данных и другими параметрами, связанными с эффективностью (критерии: производительность, аппаратные затраты) создаваемых аппаратных решений [14–16].

Эффективность, производительность и аппаратные затраты конечного результата, аппаратной реализации алгоритма решаемой задачи, существенно зависят от выбранного алгоритма решения задачи и подобранных параметров процедуры высокоуровневого синтеза. Процедура получения оптимального конечного результата не формализована, эвристическая и требует проведения исследований с использованием имитационного моделирования и сравнительного анализа [17–19].

Объект, предмет, методы, цель и средства исследования

Объектом исследования, результаты которого приведены в данной статье, является способ повышения производительности матричных вычислений.

Предмет исследования – операция суммирования транспонированных двумерных матриц размером N столбцов и M строк:

$$\mathrm{C}_{N\times M} = \mathrm{A}^{\mathrm{T}}_{M\times N} + \mathrm{B}^{\mathrm{T}}_{M\times N},$$

где C – двумерная выходная матрица размером $N\times M$; A и B – двумерные входные матрицы размером $N\times M$.

Пользуясь известными свойствами операции транспонирования матриц, алгоритм предмета исследования может быть представлен в модифицированном виде:

$$C_{N\times M} = \left(A_{M\times N} + B_{M\times N}\right)^{T},$$

где C – двумерная выходная матрица размером $N\times M$; A и B – двумерные входные матрицы размером $N\times M$.

Описание на языке Си прямого алгоритма предмета исследования приведено на рис. 1, где A_in и B_in – двумерные входные матрицы, размером $N\times M$; C_out – двумерная выходная матрица размером $M\times N$.

Функция transponse транспонирует входную матрицу, функция sum_matrix вычисляет сумму двух матриц с помощью двух вложенных циклов поэлементно, а T_SUM представляет собой функцию для решения целевой задачи с помощью предыдущих двух. При этом в функции T_SUM используются два временных (промежуточных, буферных) массива A_t и B_t для хранения транспонированных копий входных матриц A_in и B_in.

Описание на языке Си модифицированного алгоритма предмета исследования приведено на рис. 2, где A и B – двумерные входные матрицы, размером $N\times M$; C – двумерная выходная матрица размером $M\times N$.

В модифицированном алгоритме транспонирование выполняется при записи результатов суммирования двух матриц путем изменения порядка индексов. Поэтому в данной реализации алгоритма нет необходимости в буферных массивах, используемых для хранения промежуточных результатов.

Методы исследования:

• имитационное моделирование программной и аппаратной реализаций матричной операции суммирования двух транспонированных двумерных матриц (исходный алгоритм предмета исследования) и её модифицированной формы: транспонирования суммы двух двумерных матриц;

• сравнительный анализ по критериям: производительность и аппаратные затраты (для аппаратных реализаций).

Методика исследования включает следующие основные этапы:

• оценку производительности программной реализации суммирования двух транспонированных двумерных матриц;

• оценку производительности аппаратной реализации суммирования двух транспонированных двумерных матриц;

• оценку производительности программной реализации модифицированного алгоритма – операции транспонирования суммы двух двумерных матриц;

• оценку производительности аппаратной реализации модифицированного алгоритма – операции транспонирования суммы двух двумерных матриц;

• сравнительный анализ оценок, полученных при проведении исследования.

Цель исследования – выбор оптимального, по критерию производительность с учетом аппаратных затрат для аппаратных реализаций, способа реализации матричной операции суммирования двух транспонированных двумерных матриц.

Мера измерения производительности – промежуток времени, измеряемый в наносекундах, через который может быть запущена новая задача суммирования двух транспонированных матриц.

Мерами измерения аппаратных затрат являются: число встроенных модулей памяти (BRAM), необходимых для аппаратной реализации алгоритма, измеряемое в штуках; сумма числа триггеров (FF) и логических элементов (LUT), необходимых для аппаратной реализации алгоритма, измеряемая в штуках.

Исследования проводились для следующих наборов числа элементов квадратных матриц: 256×256, 512×512, 1024×1024, 2048×2048, 3072×3072 элементов. Пояснения по выбору числа элементов для проведения исследования: число элементов строк/столбцов выбрано равным степени двойки т. к. это максимальное число элементов для соответствующей разрядности адреса стро-

```
void transpose(data in_m[N][M], data out_m[M][N])
{
Loop_T1: for (int i = 0; i < N; i++)
      Loop_T2 : for (int j = 0; j < M; j++)
      out_m[j][i] = in_m[i][j];
}
void sum_matrix(data A_m[M][N], data B_m[M][N], data C_m[M][N])
{
      Loop_Sum1: for(int j=0;j<N;j++)
            Loop_Sum2: for( int i=0;i<M;i++)
              C_m[i][j]=A_m[i][j] + B_m[i][j];
}

void T_SUM(data A_in[N][M], data B_in[N][M], data C_out[M][N]){
      data A_t[M][N], B_t[M][N];

      transpose(A_in, A_t);
      transpose(B_in, B_t);
      sum_matrix(A_t, B_t, C_out);
}
```

Рис. 1. Прямая форма алгоритма суммирования транспонированных двумерных матриц
Fig. 1. The direct form of the algorithm for summing transposed two-dimensional matrices

```
void SUM_T(data A[N][M], data B[N][M], data C[M][N]) {
    SM_L2: for (int i = 0; i < N; i++) {
        SM_L1: for (int j = 0; j < M; j++) {
            C[j][i] = A[i][j] + B[i][j];
        }
    }
}
```

Рис. 2. Модифицированная форма алгоритма суммирования транспонированных двумерных матриц
Fig. 2. Modified form of the algorithm for summing transposed two-dimensional matrices

ки/столбца; полученные результаты практически не зависят от задания числа элементов строк/столбцов не равными степени двойки, например, такими как 255×255, 511×511 и далее, соответственно. Тип элементов матриц – целые, знаковые числа, разрядность 32 бита.

Средством для имитационного моделирования при программной реализации был выбран персональный компьютер (ПК) со следующими характеристиками: процессор – AMD Ryzen7 6800HS; оперативная память – 64 Гбайт, DDR5. Оболочка разработки (компилятор) для создания исполняемых файлов имитационных программных моделей – Microsoft Visual Studio 2022 Community.

Средством для аппаратной реализации, определяющим ограничения на доступные аппаратные ресурсы и задержки выполнения аппаратных функций, была выбрана микросхема (FPGA) компании Xilinx: xcku115-flva1517-2-e.

Средством для синтеза аппаратных реализаций на основе описания алгоритма на языке Си являлся пакет Vitis HLS 2022.2. Пакет позволяет осуществлять синтез устройства, получать оценки времени решения задачи, тактовой частоты устройства, а также выполнять имитационное моделирование для получения оценок производительности.

Процедура и результаты исследования

Для проведения исследования аппаратной и программной реализаций двух вариантов алгоритма суммирования двух транспонированных двумерных матриц на языке Си созданы описания, алгоритмические основы которых приведены на рис. 1, 2.

Для проведения исследования программной реализации создана имитационная модель, осуществляющая для каждого набора числа элементов (256×256, 512×512, 1024×1024, 2048×2048, 3072×3072) квадратных матриц:

– формирование двумерных квадратных матриц A и B, и их заполнение случайными данными;

– многократный (*K* раз) запуск функций SUM_T и T_SUM, алгоритмы которых приведены на рис. 1, 2 соответственно;

– оценку времени выполнения функций SUM_T и T_SUM при каждом запуске и запись оценок в соответствующие массивы;

– нахождение K/2+1 статистики (медианы) среди элементов массивов с результатами оценок времени выполнения функций SUM_T и T_SUM, а также минимальной и максимальной оценок времени;

– контроль алгоритмической правильности работы функций SUM_T и T_SUM, осуществляемый путем сравнения результатов, полученных этими функциями;

– отображение минимальной, максимальной и медианной оценок времени выполнения функций SUM_T и T_SUM для текущего набора числа элементов в матрице.

Для аппаратной реализаций каждого из двух вариантов алгоритма суммирования двух транспонированных двумерных матриц в рамках пакета Vitis HLS 2022.2: созданы наборы решений, отличающиеся степенью параллелизма, коэффициентом конвейеризации и структурной организацией памяти хранения данных; проведены исследования и выбран период тактового сигнала, осуществляющего синхронизацию элементов аппаратной реализации; получены оценки производительности и аппаратных затрат созданных наборов аппаратных решений.

Проведен сравнительный анализ результатов исследований и выбран оптимальный, по критериям производительность/аппаратные затраты, алгоритм реализации суммирования двух транспонированных двумерных матриц и способ его реализации.

При создании набора аппаратных решений использованы следующие директивы пакета Vitis HLS:

– решения sol_16_6 (sol_16_8): степень параллелизма внутреннего цикла – 16, коэффициент конвейеризации внутреннего цикла – 3, входные массивы данных структурно преобразованы в 16 массивов по второму индексу, выходные массивы данных структурно преобразованы в 16 массивов по первому индексу. Период тактового сигнала задан равным 6 нс (8 нс);

– решения sol_32_6 (sol_32_8): степень параллелизма внутреннего цикла – 32, коэффициент конвейеризации внутреннего цикла – 3, входные массивы данных структурно преобразованы в 32 массива по второму индексу, выходные массивы данных структурно преобразованы в 32 массива по первому индексу. Период тактового сигнала задан равным 6 нс (8 нс).

Результаты исследования программной и аппаратной реализаций прямого алгоритма суммирования двух транспонированных двумерных матриц, представленные в виде графиков, приведены на рис. 3.

По оси абсцисс отложено число столбцов и строк матриц, использованных в исследовании. По оси ординат, представленной в логарифмическом масштабе, отложено время выполнения алгоритма в наносекундах.

На рисунке показаны графики зависимости времени выполнения алгоритма от размера матрицы для PC – программной реализации; base (8 ns) – аппаратной реализации без оптимизации, период тактового сигнала 8 нс; sol_16_6 (sol_16_8) и sol_32_6 (sol_32_8) – оптимизированных аппаратных решений, описанных выше.

Из анализа данных, представленных на рис. 3, следует:

– графики для решений sol_16_6 и sol_16_8 практически совпали;

– графики для решений sol_32_6 и sol_32_8 практически совпали;

– решение base является самым медленным;

– решение PC показывает большую производительность относительно решений sol_16_6 (sol_16_8) для матриц с числом строк = столбцов <= 512, и меньшую производительность относительно решений sol_32_6 (sol_32_8) для любого числа строк/столбцов матриц.

Оценка аппаратных затрат для указанных выше аппаратных решений приведено на рис. 4.

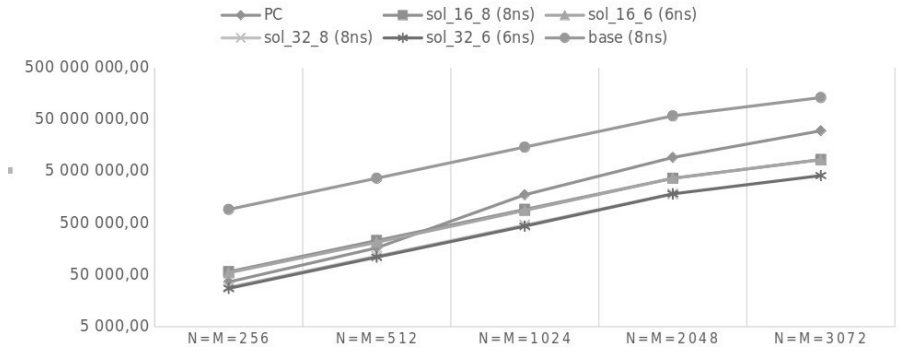

Рис. 3. Производительность для прямой формы алгоритма суммирования транспонированных двумерных матриц

Fig. 3. Performance for the direct form of the algorithm for summing transposed two-dimensional matrices

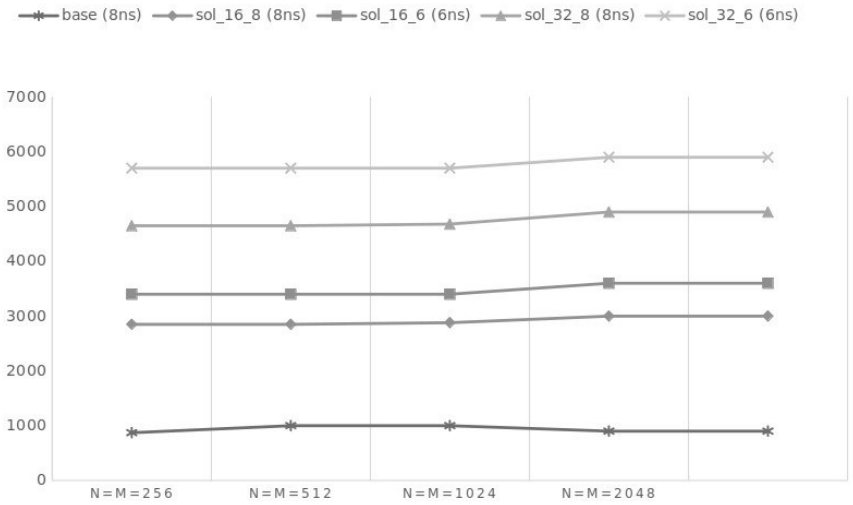

Рис. 4. Аппаратные затраты для прямой формы алгоритма суммирования транспонированных двумерных матриц

Fig. 4. Hardware costs for the direct form of the algorithm for summing transposed two-dimensional matrices

На оси ординат показана сумма использованных логических элементов и триггеров для соответствующей аппаратной реализации.

Из анализа рисунка можно заметить то, что потенциально более производительные варианты (у которых период тактовой частоты задан равным 6 нс) требуют больших аппаратных затрат. Однако из анализа рис. 3 следует, что оценки производительности, полученные после имитационного моделирования, зависят только от степени параллелизма и практически не зависят от установленного требования к периоду тактовой частоты.

Оценка числа внутренних блоков памяти, использованных для буферных массивов (см. рис. 1), показывает его квадратичную зависимость от числа строк/столбцов матриц. Поэтому, приведенные на рис. 3, 4 решения могут быть реализованы на микросхеме xcku115-flva1517-2-e, выбранной для имитационного моделирования, только для матриц с количеством строк = = столбцов <= 1024. Что ограничивает применимость прямого алгоритма суммирования двух транспонированных двумерных матриц для аппаратной реализации. Лучшей аппаратной реализацией прямой формы алгоритма, по критериям производительность/аппаратные затраты, является решение sol_32_8.

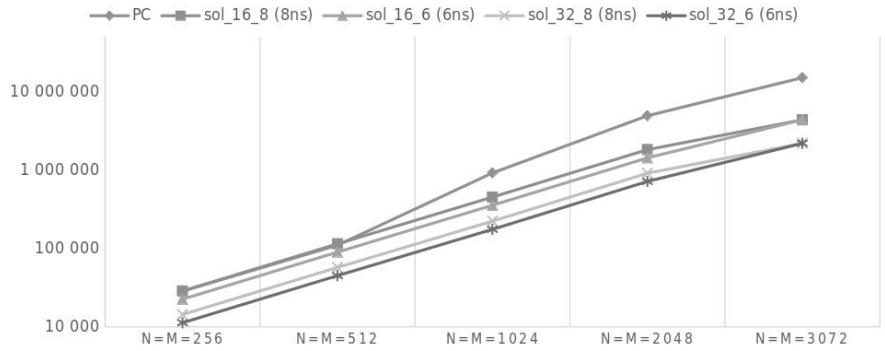

Рис. 5. Производительность для модифицированной формы алгоритма суммирования транспонированных двумерных матриц

Fig. 5. Performance for a modified form of the algorithm for summing transposed two-dimensional matrices

Результаты исследования программной и аппаратной реализаций модифицированного алгоритма суммирования двух транспонированных двумерных матриц, представленные в виде графиков, приведены на рис. 5.

По оси абсцисс отложено число столбцов и строк матриц, использованных в исследовании. По оси ординат, представленной в логарифмическом масштабе, отложено время выполнения алгоритма в наносекундах.

Из анализа полученных результатов следует, что программное решение (PC) показывает меньшую производительность относительно всех аппаратных решений для любого числа строк/столбцов матриц; решения sol_32_6 (sol_32_8) показывают большую производительность в сравнении с решениями sol_16_6 (sol_16_8).

Оценки затрат аппаратных ресурсов, требуемых для аппаратной реализации решений модифицированного алгоритма, показывают аналогичный рис. 4 характер зависимости от числа строк/столбцов обрабатываемых матриц и заданных требований к периоду тактовой частоты. Абсолютные значения количества использованных логических элементов и триггеров для соответствующей аппаратной реализации модифицированного алгоритма примерно в два раза меньше, чем для аппаратной реализации прямого алгоритма (см. рис. 4). При этом модифицированный алгоритм не требует буферных матриц. Лучшей аппаратной реализацией модифицированной формы алгоритма, по критериям производительность/аппаратные затраты, является решение sol_32_8.

Обобщение полученных в результате исследования результатов приведено на рис. 6.

По оси абсцисс отложено число столбцов и строк матриц, использованных в исследовании. По оси ординат, представленной в логарифмическом масштабе, отложено время выполнения алгоритма в наносекундах.

На рисунке показаны графики зависимости времени выполнения алгоритма от размера матрицы для T_SUM_PC – программной реализации прямой формы алгоритма; SUM_T_PC – программной реализации модифицированной формы алгоритма; T_SUM_32_8 – аппаратной реализации прямой формы алгоритма (базируется на решении sol_32_8, приведенном на рис. 3); SUM_T_32_8 – аппаратной реализации модифицированной формы алгоритма (базируется на решении sol_32_8, приведенном на рис. 5).

Анализ полученных обобщенных результатов показывает:

• Модифицированный алгоритм суммирования двух транспонированных двумерных матриц позволяет как для программной, так и для аппаратной реализаций достичь примерно в два раза большей производительности относительно производительности прямого алгоритма для соответствующих решений.

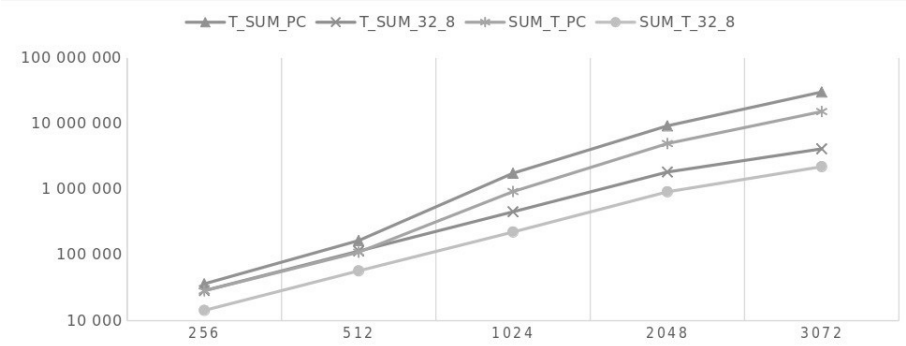

Рис. 6. Обобщенные результаты исследования
Fig. 6. Generalized results of the study

• Максимальную производительность дает аппаратное решение, основанное на модифицированной форме алгоритма – решение SUM_T_32_8. При этом выигрыш производительности аппаратного SUM_T_32_8 решения относительно лучшего программного решения увеличивается с увеличением числа строк/столбцов матриц, а сложность, число использованных логических элементов и триггеров аппаратного решения практически не изменяется.

Заключение

Цель проведения исследования достигнута – определен способ реализации операции суммирования двух транспонированных двумерных матриц, обеспечивающий максимальную производительность для заданных ограничений, и создано соответствующее аппаратное решение SUM_T_32_8.

Показано, что алгоритм реализации операции суммирования двух транспонированных двумерных матриц имеет существенное влияние как на производительность программного и аппаратных решений, так и на аппаратные затраты при аппаратной реализации.

В качестве дальнейшей работы в данном направлении целесообразно провести аналогичные исследования других часто используемых матричных операций.

СПИСОК ЛИТЕРАТУРЫ

1. **Antonov A., Zaborovskiy V., Kiselev I.** The reconfigurable computational modules in network-centric supercomputer systems // High Availability Systems. 2018. vol. 14. No. 3. Pp. 57–62. DOI: 10.18127/j20729472-201803-09

2. **Kobayashi R., Oobata Y., Fujita N., Yamaguchi Y., Boku T.** OpenCL-ready high speed FPGA network for reconfigurable high performance computing // Proc. of the Internat. Conf. on High Performance Computing in Asia-Pacific Region. 2018. Pp. 192–201. DOI: 10.1145/3149457.3149479

3. **Qasim S.M., Telba A.A., Al Mazroo A.** FPGA design and implementation of matrix multiplier architectures for image and signal processing applications // Internat. J. of Computer Science and Network Security. 2010. No. 10. Pp. 168–176.

4. **Zhang Y., Shalabi Y.H., Jain R., Nagar K.K., Bakos J.D.** FPGA vs. GPU for sparse matrix vector multiply // Internat. Conf. on Field-Programmable Technology. 2009. Pp. 255–262. DOI: 10.1109/FPT.2009.5377620

5. **Tan Y., Imamura T., Mukunoki D.** Design of an FPGA-based matrix multiplier with task parallelism // Parallel Computing: Technology Trends. 2020. vol. 36. Pp. 241–250. DOI: 10.3233/APC200047

6. **Kumar V., Joshi S., Patkar S., Narayanan H.** FPGA based high performance double-precision matrix multiplication // Internat. J. of Parallel Programming. 2010. No. 38. Pp. 322–338. DOI: 10.1007/s10766-010-0131-8

7. **Jiang J., Mirian V., Tang K., Chow P., Xing Z.** Matrix multiplication based on scalable macro-pipelined FPGA accelerator architecture // Internat. Conf. on Reconfigurable Computing and FPGAs. 2009. Pp. 48–53. DOI: 10.1109/ReConFig.2009.30

8. **Abbaszadeh A., Iakymchuk T., Bataller-Mompeán M., Frances-Villora J.V., Rosado A.** Anscalable matrix computing unit architecture for FPGA and SCUMO user design interface // Electronics. 2019. vol. 8. P. 94. DOI: 10.3390/electronics8010094

9. **Antonov A., Zaborovskij V., Kisilev I.** Developing a new generation of reconfigurable heterogeneous distributed high performance computing system // Proc. of Internat. Scientific Conf. on Telecommunications, Computing and Control. Smart Innovation, Systems and Technologies. Springer, 2021. vol 220. DOI: 10.1007/978-981-33-6632-9_22

10. **Antonov A., Besedin D., Filippov A.** Research of the efficiency of high-level synthesis tool for FPGA based hardware implementation of some basic algorithms for the big data analysis and management tasks // 2020 26th Conf. of Open Innovations Association (FRUCT). Yaroslavl, Russia, 2020. Pp. 1–7. DOI: 10.23919/FRUCT48808.2020.9087355

11. **Kalyaev I., Antonov A., Zaborovskij V.** Architecture of reconfigurable heterogeneous distributed supercomputer system for solving problems of intelligent data processing in the era of digital transformation of the economy // Cybersecurity Issues. 2019. Pp. 2–11. DOI: 10.21681/2311-3456-2019-5-02-11

12. **Asch M., Moore T., Badia R., et al.** Big data and extreme-scale computing: Pathways to Convergence-Toward a shaping strategy for a future software and data ecosystem for scientific inquiry // The Internat. J. of High Performance Computing Applications. 2018. vol. 4. No. 32. Pp. 435–479. DOI: 10.1177/1094342018778123

13. **Haidar A.** Investigating power capping toward energy-efficient scientific applications // Concurrency and Computation Practice and Experience. 2018. Pp. 1–14. DOI: 10.1002/cpe.4485

14. **Antonov A., Zaborovsky V., Polyanskiy V.** Neural computations in control problems: Aspects of computability and spatial-time characterization of cognitive functions // J. of Physics: Conference Series. 2021. vol. 1864. No. 1. DOI: 10.1088/1742-6596/1864/1/012104

15. **Dongarra J.** Race to exascale // Computing in Science and Engineering. 2019. vol. 21. No. 1. Pp. 4–5. DOI: 10.1109/MCSE.2018.2882574

16. **Harris A.** Exascale models of stellar explosions: Quintessential multi-physics simulation // The Internat. J. of High Performance Computing Applications. 2021. DOI: 10.1177/10943420211027937

17. **Le Fèvre V.** Comparing the performance of rigid, moldable and grid-shaped applications on failure-prone HPC platforms // Parallel Computing. 2019. vol. 85. Pp. 1–12. DOI: 10.1016/j.parco.2019.02.002

18. **Usman A., Fathy A., Aiiad A., Abdullah A.** Performance and power efficient massive parallel computational model for HPC heterogeneous exascale systems // IEEE Access. 2018. No. 6. Pp. 23095–23107. DOI: 10.1109/ACCESS.2018.2823299

19. **Mantovani F., Calore E.** Performance and power analysis of HPC workloads on heterogeneous multi-node clusters // Low Power Electron. 2018. vol. 2. No. 8. Pp. 1–14. DOI:10.3390/jlpea8020013

REFERENCES

1. **Antonov A., Zaborovskiy V., Kiselev I.** The reconfigurable computational modules in network-centric supercomputer systems. *High Availability Systems*, 2018, vol. 14, No. 3, Pp. 57–62. (rus). D0I: 10.18127/j20729472-201803-09

2. **Kobayashi R., Oobata Y., Fujita N., Yamaguchi Y., Boku T.** OpenCL-ready high speed FPGA network for reconfigurable high performance computing. *Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region*, 2018, Pp. 192–201. DOI: 10.1145/3149457.3149479

3. **Qasim S.M., Telba A.A., Al Mazroo A.** FPGA design and implementation of matrix multiplier architectures for image and signal processing applications. *International Journal of Computer Science and Network Security*, 2010, No. 10, Pp. 168–176.

4. **Zhang Y., Shalabi Y.H., Jain R., Nagar K.K., Bakos J.D.** FPGA vs. GPU for sparse matrix vector multiply. *International Conference on Field-Programmable Technology*, 2009, Pp. 255–262. DOI: 10.1109/FPT.2009.5377620

5. **Tan Y., Imamura T., Mukunoki D.** Design of an FPGA-based matrix multiplier with task parallelism. *Parallel Computing: Technology Trends*, 2020, vol. 36, Pp. 241–250. DOI: 10.3233/APC200047

6. **Kumar V., Joshi S., Patkar S., Narayanan H.** FPGA based high performance double-precision matrix multiplication. *International Journal of Parallel Programming*, 2010, No. 38, Pp. 322–338. DOI: 10.1007/s10766-010-0131-8

7. **Jiang J., Mirian V., Tang K., Chow P., Xing Z.** Matrix multiplication based on scalable macro-pipelined FPGA accelerator architecture. *International Conference on Reconfigurable Computing and FPGAs*, 2009, Pp. 48–53. DOI: 10.1109/ReConFig.2009.30

8. **Abbaszadeh A., Iakymchuk T., Bataller-Mompeán M., Frances-Villora J.V., Rosado A.** Anscalable matrix computing unit architecture for FPGA and SCUMO user design interface. *Electronics*, 2019, vol. 8. Pp. 94. DOI: 10.3390/electronics8010094

9. **Antonov A., Zaborovskij V., Kisilev I.** Developing a new generation of reconfigurable heterogeneous distributed high performance computing system. *Proceedings of International Scientific Conference on Telecommunications, Computing and Control. Smart Innovation, Systems and Technologies*. Springer, 2021, vol. 220. DOI: 10.1007/978-981-33-6632-9_22

10. **Antonov A., Besedin D., Filippov A.** Research of the efficiency of high-level synthesis tool for FPGA based hardware implementation of some basic algorithms for the big data analysis and management tasks. *2020 26th Conference of Open Innovations Association* (*FRUCT*), Yaroslavl, Russia, 2020, Pp. 1–7. DOI: 10.23919/FRUCT48808.2020.9087355

11. **Kalyaev I., Antonov A., Zaborovskij V.** Architecture of reconfigurable heterogeneous distributed supercomputer system for solving problems of intelligent data processing in the era of digital transformation of the economy. *Cybersecurity Issues*, 2019, Pp. 2–11. DOI: 10.21681/2311-3456-2019-5-02-11

12. **Asch M., Moore T., Badia R., et al.** Big data and extreme-scale computing: Pathways to Convergence-Toward a shaping strategy for a future software and data ecosystem for scientific inquiry. *The International Journal of High Performance Computing Applications*, 2018, vol. 4, No. 32, Pp. 435–479. DOI: 10.1177/1094342018778123

13. **Haidar A.** Investigating power capping toward energy-efficient scientific applications. *Concurrency and Computation Practice and Experience*, 2018, Pp. 1–14. DOI: 10.1002/cpe.4485

14. **Antonov A., Zaborovsky V., Polyanskiy V.** Neural computations in control problems: Aspects of computability and spatial-time characterization of cognitive functions. *Journal of Physics: Conference Series*, 2021, vol. 1864, No. 1. DOI: 10.1088/1742-6596/1864/1/012104

15. **Dongarra J.** Race to exascale. *Computing in Science and Engineering*, 2019, vol. 21, No. 1, Pp. 4–5. DOI: 10.1109/MCSE.2018.2882574

16. **Harris A.** Exascale models of stellar explosions: Quintessential multi-physics simulation. *The International Journal of High Performance Computing Applications*, 2021. DOI: 10.1177/10943420211027937

17. **Le Fèvre V.** Comparing the performance of rigid, moldable and grid-shaped applications on failure-prone HPC platforms. *Parallel Computing*, 2019, vol. 85, Pp. 1–12. DOI: 10.1016/j.parco.2019.02.002

18. **Usman A., Fathy A., Aiiad A., Abdullah A.** Performance and power efficient massive parallel computational model for HPC heterogeneous exascale systems. *IEEE Access*, 2018, No. 6, Pp. 23095–23107. DOI: 10.1109/ACCESS.2018.2823299

19. **Mantovani F., Calore E.** Performance and power analysis of HPC workloads on heterogeneous multi-node clusters. *Low Power Electron*, 2018, vol. 2, No. 8, Pp. 1–14. DOI:10.3390/jlpea8020013

INFORMATION ABOUT AUTHORS / СВЕДЕНИЯ ОБ АВТОРАХ

Антонов Александр Петрович
Alexander P. Antonov
E-mail: antonov@eda-lab.ftk.spbstu.ru

Беседин Денис Сергеевич
Denis S. Besedin
E-mail: nero1310@yandex.ru

Филиппов Алексей Семенович
Alexey S. Filippov
E-mail: alexey.s.filippov@gmail.com

Поступила: 03.12.2022; Одобрена: 26.12.2022; Принята: 12.01.2023.
Submitted: 03.12.2022; Approved: 26.12.2022; Accepted: 12.01.2023.