УДК 004.021 DOI: 10.31114/2078-7707-2021-3-76-80

Оценка использования систолических массивов при реализации алгоритмов умножения матриц на ПЛИС

С. В. Пасынков, Р. Ф. Ильясов

Автономная некоммерческая организация высшего образования «Университет Иннополис» (АНО ВО «Университет Иннополис»), r.iliasov@innopolis.ru

***Аннотация* — Фундаментальным строительным блоком многих алгоритмов, таких как анализ данных и нейронные сети, является умножение матриц. В течение последних десятилетий систолические массивы зарекомендовали себя в качестве оптимального и высокоэффективного решения, и в последнее время интерес к ним только увеличился. Цель данной статьи - определить оптимальный метод умножения двух матриц в зависимости от максимально возможной тактовой частоты и ресурсов ПЛИС для реализации проекта. Для проведения измерений мы реализовали два дизайна на языке Verilog для умножения матриц 3 на 3 и 10 на 10 с использованием систолического массива. В дополнение к вышеперечисленным дизайнам мы также реализовали дизайны на языке Verilog для умножения матриц 3 на 3 и 10 на 10 без использования систолических массивов. ПЛИС Cyclone IV включают комбинацию встроенных ресурсов, которые помогают повысить производительность и известны как выделенные блоки цифровой обработки сигналов (DSP). Поэтому в дополнение к вышеуказанным дизайнам умножения матриц на устройствах FPGA было решено реализовать конструкцию для умножения матриц с использованием встроенных блоков умножения. Сравнение двух реализаций умножения проводилось с использованием встроенных функций программы Quartus Prime. Результаты этой работы показали, что алгоритм систолического массива до сих пор является оптимальным и высокоэффективным решением для использования в приложениях линейной алгебры. Используя его, можно ускорить выполнение операций перемножения, а также уменьшить количество требуемых элементов для реализации дизайна на языке Verilog.**

***Ключевые слова* — Verilog, умножение, матрицы, систолический массив, блоки умножения, DSP block.**

I. Введение

Искусственный интеллект и машинное обучение активно используются в современном мире [1]. Алгоритмы и модели для этих приложений становятся все более сложными, а наборы данных становятся все больше. Таким образом, потребности в вычислениях растут в геометрической прогрессии. Совсем недавно достижения в области компьютерного зрения и глубоких нейронных сетей (DNN) повысили потребность в увеличении скорости выполнения этих операций [2]. Ускорение вычислений, требуемых для AI и ML, является серьезной проблемой. Было предложено множество решений для ускорения глубоких нейронных сетей в аппаратном обеспечении, начиная от ASIC и заканчивая полностью программируемыми графическими процессорами и конфигурируемыми решениями на основе ПЛИС [3]. Конструкции на основе ASIC обладают наилучшими характеристиками скорости и мощности (быстрая и низкая мощность), но им не хватает конфигурируемости и адаптивности, что имеет решающее значение в быстро меняющемся мире AI/ML. Проекты на основе графических процессоров и процессоров, хотя они легко программируются и адаптируются, не так быстры и энергоэффективны, как ASIC [3]. Конструкции на основе ПЛИС обеспечивают лучшее из обоих миров. Они обеспечивают массовый параллелизм, будучи гибкими и легко настраиваемыми, а также быстрыми и энергоэффективными [3].

Перемножение матриц — одна из фундаментальных операций линейной алгебры с многочисленными приложениями в математике, статистике, физике, экономике, информатике и искусственном интеллекте [4]. Систолические массивы с многообещающими атрибутами, такими как высокая степень параллельных вычислений и высокая скорость повторного использования данных, являются привлекательными решениями для многих приложений линейной алгебры например, QR-разложение комплексной матрицы [5] , матричная триангуляризация [6], перемножение матриц. В последнее время систолические массивы часто стали использовать для ускорения вывода глубоких нейронных сетей (DNNs) [7]. Повторное использование данных сводит к минимуму потребность в пропускной способности памяти за счет взаимодействия потоков данных в вычислительных массивах [8]. В результате систолические массивы хорошо работают для вычисления линейных повторений и вычислений линейной алгебры. Данная статья преследует цели определения наиболее оптимального способа перемножения двух матриц в зависимости от максимальной возможной тактовой частоты и ресурсов ПЛИС для реализации дизайна.

II. Методология

Рассмотрим понятие систолического массива. Систолический массив — это сеть вычислительных элементов, которые ритмично вычисляют и передают данные через систему [9]. Помимо этого каждый вычислительный элемент может сохранять промежуточные результаты. Подобно тому, как кровь

ритмично течет через биологическое сердце, данные ритмично вытекают из памяти, проходя через многие элементы, прежде чем вернуться в память [10]. Систолический массив — это отличный пример конвейеризации наряду с параллельными вычислениями [11].

Преимущества использования систолических массивов:

1. Они легко реализуемы.
2. Данные в систолическом массиве используются последовательно без записи в память, благодаря чему увеличивается пропускная способность, без увеличения пропускной способности памяти.
3. Данные используются многократно, поэтому избегаются стадии fetch и refetch.

Недостатки: узкая направленность. Систолические массивы чаще всего используются в приложениях линейной алгебры таких как, перемножение матриц, задача о наименьших квадратах и другие.

Вычислительный элемент систолического массива для перемножения матриц принимает данные сверху и слева, обрабатывает, а затем передает вниз и направо соответственно принятые элементы без изменений. Схема показана на рис. 1.

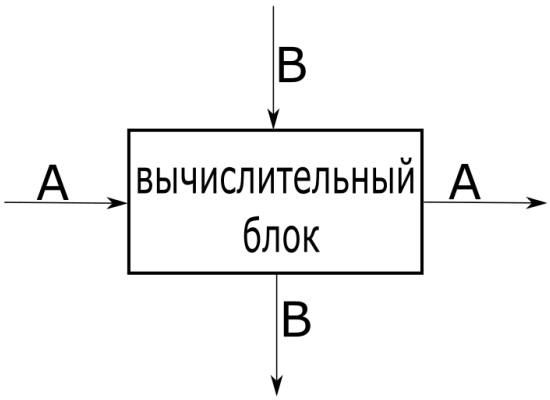

Рис. 1. Передача данных через вычислительный элемент

Элементы обработки расположены в виде массива. На рис. 2 показано, как матрицы A и B подаются в систолический массив.

Для проведения измерений были реализованы два дизайна на языке программирования Verilog для перемножения матриц 3 на 3 и 10 на 10 с использованием систолических массивов. В данной статье для перемножения двух матриц был использован систолический массив типа a (рис. 3).

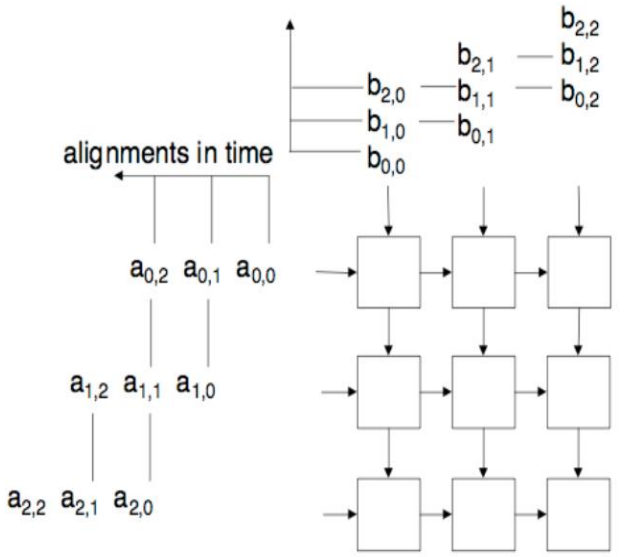

Рис. 2. Схема систолического массива [12]

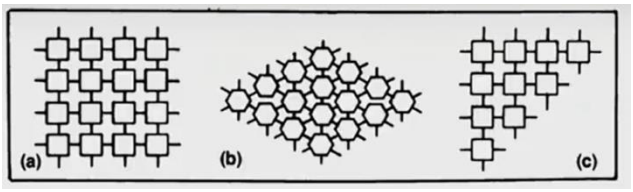

Рис. 3. Типы двумерных систолических массивов [13]

На рис. 4 показан обрабатывающий элемент систолического массива.

```
module pe(clk,reset,in_a,in_b,out_a,out_b,out_c);

 parameter data_size=8;
 input wire reset,clk;
 input wire [data_size-1:0] in_a,in_b;
 output reg [2*data_size:0] out_c;
 output reg [data_size-1:0] out_a,out_b;

 always @(posedge clk)begin
    if(reset) begin
      out_a<=0;
      out_b<=0;
      out_c<=0;
    end
    else begin
      out_c<=out_c+in_a*in_b;
      out_a<=in_a;
      out_b<=in_b;
    end
 end

endmodule
```

Рис. 4. Вычислительный элемент (processing element), используемый в систолическом массиве

Генерация соединений систолических массивов 3 на 3 и 10 на 10 была сделана с помощью программы, написанной на языке программирования Python. В этой статье мы использовали программы на языке программирования Python вместо generate блоков, так наш вариант более интуитивно понятен. Кроме того структура сгенерированного дизайна недостаточно однородна, поэтому использование generate блока

только усложнило бы задачу. Код генерации дизайна для перемножения матриц в зависимости от аргумента, который определяет размер квадратной матрицы с использованием систолического массива, приведен на рис. 6.

В дополнении к предыдущим дизайнам мы также реализовали два дизайна на языке программирования Verilog для перемножения матриц 3 на 3 и 10 на 10 без использования систолических массивов. Дизайн на языке описания аппаратуры Verilog для перемножения матриц 3 на 3 приведен на рисунке 5.

```
module top(clk,reset,a1, a2, a3, a4, a5, a6, a7, a8, a9, b1,
           b2, b3, b4, b5, b6, b7, b8, b9,c1,c2,c3,c4,c5,c6,c7,c8,c9);

 parameter data_size=8;
 input wire clk,reset;
 input wire [data_size-1:0] a1, a2, a3, a4, a5, a6, a7, a8, a9
 input wire [data_size-1:0] b1, b2, b3, b4, b5, b6, b7, b8, b9;
 output reg [2*data_size:0] c1,c2,c3,c4,c5,c6,c7,c8,c9;

  always @(posedge clk)begin
    if(reset) begin
      c1<=0;
      c2<=0;
      c3<=0;
      c4<=0;
      c5<=0;
      c6<=0;
      c7<=0;
      c8<=0;
      c9<=0;
    end
    else begin
      c1<=((a1 * b1) + (a2 * b4) + (a3 * b7));
      c2<=((a1* b2) + (a2 * b5) + (a3 * b8));
      c3<=((a1* b3) + (a2 * b6) + (a3 * b9));
      c4<=((a4* b1) + (a5 * b4) + (a6 * b7));
      c5<=((a4* b2) + (a5 * b5) + (a6 * b8));
      c6<=((a4* b3) + (a5 * b6) + (a6 * b9));
      c7<=((a7* b1) + (a8 * b4) + (a9 * b7));
      c8<=((a7* b2) + (a8 * b5) + (a9 * b8));
      c9<=((a7* b3) + (a8 * b6) + (a9 * b9));
    end
 end
endmodule
```

Рис. 5. Дизайн для перемножения матриц 3 на 3 без использования систолического массива

Дизайн для перемножения матриц 10 на 10 без использования систолического массива реализован по аналогии с дизайном, показанным на рис. 5.

Также устройства Cyclone IV включают в себя комбинацию встроенных ресурсов, которые помогают повысить производительность, снизить стоимость системы и снизить энергопотребление систем цифровой обработки сигналов (встроенные DSP блоки) [14]. Встроенные DSP блоки ПЛИС особенно часто используются для оптимизации приложений, которые извлекают выгоду из большого количества ресурсов параллельной обработки [15]. Поэтому в дополнение к вышеперечисленным дизайнам перемножения матриц на устройствах ПЛИС, мы решили реализовать дизайн для перемножения матриц с использованием встроенных DSP блоков. Дизайн для перемножения матриц 3 на 3 был реализован с использованием DSP блоков LPM_MULT и PARALLEL_ADD.

Все измерения проводились в программе Quartus Prime с выбранной платой ПЛИС Cyclone IV E: EP4CE115F29I8L, так как для симуляции дизайна было необходимо более 110,000 логических элементов ПЛИС.

В данной статье для компилирования всех дизайнов были использованы virtual pins, благодаря чему мы показали максимальную возможную частоту модулей вычисления вне зависимости от вариаций автоматического соединения пинов в Quartus'e.

Для всех дизайнов был выбран режим оптимизации "Balanced (Normal flow)" в программе Quartus. В таблицу были занесены данные о максимальной частоте из графы Slow 1000mV 100C. Результаты сравнения различных дизайнов для перемножения матриц показаны в таблице 1.

Таблица 1

Сравнение реализаций перемножения

Реализация	**Логические элементы**	**Макс. частота (MHz)**
с систолическим массивом		
10x10	3854	93.7
3x3	249	118.19
без использования систолического массива		
10x10	60061	63.2
3x3	272	105.72
с использованием встроенных блоков умножения		
3x3	432	201.33

```
size = 3
par = size ** 2 + 1
[print("wire [data_size-1:0] c" + str(i) + ";") for i in range(1, par)]
for i in range(1, par):
    ia = ''
    ib = ''
    oa = ''
    ob = ''
    oc = 'c' + str(i)
    if i % size == 1:
        ia = "a" + str(i // size + 1)
    else:
        ia = "a{}{}".format(str(i - 1), str(i))
    if i <= size:
        ib = "b" + str(i)
    else:
        ib = "b{}{}".format(str(i - size), str(i))
    if (i + 1) % size == 1:
        oa = "a" + str(i)
    else:
        oa = "a{}{}".format(str(i), str(i + 1))
    if i % size == 0:
        oa = ''
    ob = "b{}{}".format(str(i), str(i + size))
    if i > par - size - 1:
        ob = ''
    print("pe pe{} (.clk(clk), .reset(reset),"
          " .in_a({}), .in_b({}), .out_a({}),"
          " .out_b({}), .out_c({}));".format(str(i), ia, ib, oa, ob, oc))
```

Рис. 6. Код на языке программирования Python для генерации соединений дизайна с использованием систолического массива

III. Заключение

В данной работе мы сравнили разные дизайны на языке описания аппаратуры Verilog для перемножения матриц и пришли к выводу, что алгоритм систолического массива, который был впервые описан в статье 1979 года, до сих пор является оптимальным и высокоэффективным решением для использования в приложениях линейной алгебры. Используя его, можно ускорить выполнение операций перемножения, а также уменьшить количество требуемых элементов для реализации дизайна на языке описания аппаратуры Verilog.

Благодарности

Выражаем благодарность Юрию Панчулу за помощь в организации обучения микроархитектуре цифровых схем в Иннополисе с 2018 года.

Литература

[1] R. Novickis et al. An Approach of Feed-Forward Neural Network Throughput-Optimized Implementation in FPGA //Electronics. – 2020. – Т. 9. – №. 12. – С. 2193.

[2] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, no. 7553, p. 436, 2015.

[3] A. Arora, Z. Wei, L. K. John, Hamamu: Specializing FPGAs for ML Applications by Adding Hard Matrix Multiplier Blocks //2020 IEEE 31st International Conference on Application-specific Systems, Architectures and Processors (ASAP). – IEEE, 2020. – С. 53-60.

[4] J. M. Landsberg, “Tensors: geometry and applications,” Representation theory, vol. 381, no. 402, p. 3, 2012.

[5] А.А. Мальцев, В.А. Пестрецов, А.В. Хоряев, Р.О. Масленников, Метод QR-разложения комплексной матрицы с помощью треугольного систолического массива // Проблемы разработки перспективных микроэлектронных систем - 2006. Сборник научных трудов / под общ. ред. А.Л.Стемпковского. М.:ИППМ РАН, 2006. С. 91-96.

[6] W. M. Gentleman, H. T. Kung, Matrix triangularization by systolic arrays //Real-time signal processing IV. – International Society for Optics and Photonics, 1982. – Т. 298. – С. 19-26.

[7] B. Asgari et al. Eridanus: Efficiently running inference of dnns using systolic arrays //Ieee micro. – 2019. – Т. 39. – №. 5. – С. 46-54.

[8] И.А. Калмыков, М.И. Калмыков, Е.П. Степанова, А.В. Велигоша, В. Бороденко , Отказоустойчивый систолический процессор цифровой обработки сигналов, функционирующий в модулярном коде // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). 2016. № 1. С. 242-248.

[9] URL: https://www.geeksforgeeks.org/parallel-processing-systolic-arrays/ (дата обращения: 28.07.2021)

[10] K. T. Johnson, A. R. Hurson , B. Shirazi, General-purpose systolic arrays //Computer. – 1993. – Т. 26. – №. 11. – С. 20-31.

[11] P. Quinton,The systematic design of systolic arrays : дис. – INRIA, 1983.

[12] URL: http://ashanpeiris.blogspot.com/2015/08/digital-design-of-systolic-array.html (дата обращения: 1.07.2021)

[13] H. T. Kung, Why systolic architectures? //Computer. – 1982. – Т. 15. – №. 01. – С. 37-46.

[14] URL: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/an/an306.pdf (дата обращения: 5.07.2021)

[15] URL: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/cyclone-iv/cyiv-51004.pdf (дата обращения: 1.07.2021)

Evaluation of Systolic Arrays Using for the Implementation of Matrix Multiplication Algorithms in FPGAs

S. Pasynkov, R. Ilyasov

Innopolis University, Innopolis, Russia, r.iliasov@innopolis.ru

***Abstract* — Matrix multiplication is the fundamental building block of many algorithms, such as data analysis and neural networks. Over the past decades, systolic arrays have proven themselves as the optimal solution for using data with high efficiency, and recently there has been an increased interest in them. This article compares the multiplication operations of 2 designs for multiplying matrices using a systolic array and without using it, as well as using embedded multiplication blocks (DSP blocks). This article aims to determine the most optimal method for multiplying two matrices, depending on the maximum possible clock frequency and FPGA resources for implementing the design. To produce the measurements, we implemented two designs for matrix multiplication using a systolic array. In addition to these, we implemented two designs in the Verilog hardware description language for multiplying matrices 3 by 3 and 10 by 10 without using systolic arrays. Cyclone IV devices also include a combination of built-in resources that help improve performance, known as dedicated digital signal processing (DSP) blocks. Therefore, in addition to the abovementioned matrix multiplication designs on FPGA devices, we implemented a design for matrix multiplication using built-in multiplication blocks. All measurements were performed in the Quartus Prime program with the selected Cyclone IV E: EP4CE115F29I8L board, since more than 110,000 FPGA logic elements were needed to simulate the design. In this article, virtual pins were used to compile all the designs. Owing to this, we showed the maximum possible frequency of calculation modules, regardless of the variations of the automatic connection of pins in Quartus. The comparison of all multiplication**

implementations was done by using the embedded functions of the Quartus Prime program. For all the designs, we selected " Balanced (Normal flow)" optimization mode in the Quartus program. The data on the maximum frequency from the graph Slow 1000mV 100C were entered in the table. The results of this work have shown that the systolic array algorithm is still an optimal and highly efficient solution for use in linear algebra applications and usage of systolic arrays can improve the execution of multiplication operations and reduce the number of required elements for implementing a design in the Verilog hardware description language.

***Keywords* — Verilog, multiplication, matrices, systolic array, DSP block, embedded multiplication block.**

ACKNOWLEDGMENTS

We express our gratitude to Yuri Panchul for his help in organizing the microarchitecture of digital circuits course in Innopolis since 2018.

REFERENCES

[1] R. Novickis et al. An Approach of Feed-Forward Neural Network Throughput-Optimized Implementation in FPGA //Electronics. – 2020. – Т. 9. – №. 12. – С. 2193.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, p. 436, 2015.

[3] A. Arora, Z. Wei, L. K. John, Hamamu: Specializing FPGAs for ML Applications by Adding Hard Matrix Multiplier Blocks //2020 IEEE 31st International Conference on Application-specific Systems, Architectures and Processors (ASAP). – IEEE, 2020. – С. 53-60.

[4] J. M. Landsberg, "Tensors: geometry and applications," Representation theory, vol. 381, no. 402, p. 3, 2012.

[5] Maltsev A.A., Pestretsov V.A., Khoryaev A.V., Maslennikov R.O. Method of QR-decomposition of a complex matrix by means of triangular systolic array // Problems of Perspective Microelectronic Systems Development - 2006. Proceedings / edited by A. Stempkovsky, Moscow, IPPM RAS, 2006. P. 91-96.

[6] W. M. Gentleman, H. T. Kung, Matrix triangularization by systolic arrays //Real-time signal processing IV. – International Society for Optics and Photonics, 1982. – Т. 298. – С. 19-26.

[7] B. Asgari et al. Eridanus: Efficiently running inference of dnns using systolic arrays //Ieee micro. – 2019. – Т. 39. – №. 5. – С. 46-54.

[8] Kalmykov I.A., Kalmykov M.I., Stepanova E.P., Veligosha A.V., Borodenko V. Fault-tolerant systolic processor for digital signal processing in modular code // Problems of Perspective Micro- and Nanoelectronic Systems Development - 2016. Proceedings / edited by A. Stempkovsky, Moscow, IPPM RAS, 2016. Part 1. P. 242-248.

[9] URL: https://www.geeksforgeeks.org/parallel-processing-systolic-arrays/ (access date: 28.07.2021)

[10] K. T. Johnson, A. R. Hurson , B. Shirazi, General-purpose systolic arrays //Computer. – 1993. – Т. 26. – №. 11. – С. 20-31.

[11] P. Quinton,The systematic design of systolic arrays : дис. – INRIA, 1983.

[12] URL: http://ashanpeiris.blogspot.com/2015/08/digital-design-of-systolic-array.html (access date: 1.07.2021)

[13] H. T. Kung, Why systolic architectures? //Computer. – 1982. – Т. 15. – №. 01. – С. 37-46.

[14] URL: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/an/an306.pdf (дата обращения: 5.07.2021)

[15] URL: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/cyclone-iv/cyiv-51004.pdf (access date: 1.07.2021)