moving of component parts of commodity roller the **Scientific novelty** of the got results consists in further development and improvement of methods of planning of mechanisms of rolling-up of linen of knitting machines.

Practical meaningfulness of researches consists in development of fundamentally new construction of commodity roller, able to promote efficiency of work of mechanism of rolling-up of linen.

Keywords: *knitting machine, knitting linen, mechanism of rolling-up of linen, commodity rolle.*

УДК 004.27

ЗАХАРЧЕНКО Т.Л.
Национальный технический университет Украины «Киевский политехнический институт»

ИССЛЕДОВАНИЕ ВОЗМОЖНОСТИ РЕАЛИЗАЦИИ ЭФФЕКТИВНОЙ ВЫЧИСЛИТЕЛЬНОЙ АРХИТЕКТУРЫ НА РЕКОНФИГУРИРУЕМЫХ ЛОГИЧЕСКИХ УСТРОЙСТВАХ

Цель. *Статья призвана доказать эффективность применения реконфигурируемой архитектуры для ускорения вычислений. Проверка этой гипотезы осуществлялась экспериментально.*

Методика. *Для этого в настоящей публикации приведены примеры популярных процедур ускорения, и получено количественное сравнение производительности вычислительных архитектур с применением реконфигурируемой логикой и без. В качестве примеров использовалась реализация алгоритмов фильтрации изображения и умножения матрицы на вектор.*

В результате *проверки предложенной гипотезы показано, что использование реконфигурируемой логики имеет смысл. На упомянутых примерах достигнут многократный прирост производительности. Вместе с этим следует понимать, что рассмотрен идеальный случай, так как реализацию алгоритмов вычислений на реконфигурируемых устройствах проводил человек, а не компьютер, автоматический перенос алгоритмов на реконфигурируемую логику может давать худшие результаты. Метрики в статье для оценки качества переноса алгоритма на реконфигурируемую платформу использованы впервые, а вопрос об ускорении вычислений и более экономичном использовании площади кристалла микросхемы стоит очень остро.*

Ключевые слова: *компьютерная архитектура, ускорение, реконфигурируемая логика, вычислительная техника.*

Введение. Современные вычислительные архитектуры, которые строятся на базе микропроцессоров с классическими архитектурами, невозможно назвать универсальными решением вычислительных задач. К этому утверждению ведет сама суть этого понятия, сформированного множеством устоявшихся типов архитектур[1]. Каждая такая архитектура предусматривает команды, на извлечение которых из памяти уходит время, а так же наличие особенностей, которые, возможно, будут не нужны для выполнения некоторых задач и которые будут большинство машинного времени попусту занимать место на кристалле. С появлением реконфигурируемых логических

устройств отпадает необходимость использовать команды и жестко специфицированные архитектуры. Пользователь свободен создавать собственные вычислительные архитектуры для ускорения вычислений [2]. Вместе с этим все-таки существуют определенные приемы при конструировании архитектур. Более того, все вычислительные архитектуры тесно связаны с разного рода компромиссами и конструктору приходится их разрешать для каждой из конкретных задач. Так постоянно ведется борьба между скоростью и количеством транзисторов, между этими двумя критериями и финансовыми средствами и т.д.

Объектом исследования является вычислительная архитектура на базе реконфигурируемых логических устройств. Предмет исследования – ее производительность и преимущества перед классической вычислительной архитектурой. К методам исследования относятся эксперимент, анализ и синтез.

Постановка задания. Экспериментальным путем попробовать доказать, что вычислительная архитектура на базе реконфигурированных логических устройств имеет преимущество в производительности над классическими вычислительными архитектурами. Получить количественные результаты.

В ходе исследования разобраны две типовые задачи: умножение матрицы на вектор и фильтрация изображения.

Умножение матрицы на вектор с использованием обычного микропроцессора может быть представлено двумя вложенными циклами типа for, при условии, что матрица хранится в формате «по столбцам». Такая простая реализация несет в себе высокую нагрузку на процессор, количество операций при его выполнении может быть огромным, для каждого элемента матрицы нужно выполнить операцию извлечения из памяти как самого элемента матрицы, так и элемента вектора, на который умножается матрица, операцию умножения, операцию инкрементирования итератора, так же есть определенное количество других команд. Так же следует учесть промахи кэша микропроцессора и получится довольно внушительное число. Так умножение матрицы 10x10 на вектор будет занимать приблизительно 621 такт. Это число можно значительно уменьшить, используя реализацию такого умножителя матрицы на вектор на реконфигурируемой логике. Во-первых в задачи присутствует параллелизм уровня данных. Конечно, его эксплуатация возможна и на современных многоядерных микропроцессорах, но она ограничена каналом обмена данными с памятью. Использование же реконфигурируемой логики позволит закешировать все данные на кристалл логического устройства и организовать к ним тип доступа, который лучше всего подходит для данной конкретной задачи. Такую задачу легко распараллелить на число подзадач, равное количеству строк и для каждой строки создать свой банк памяти.

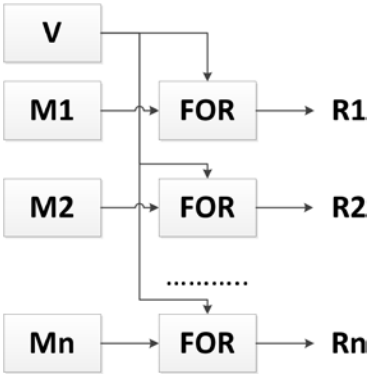

Рис. 1. **Схема принципа работы умножителя матрицы на вектор**

На рис. 1. изображен принцип реализации умножителя матрицы на вектор. FOR – блок, который выполняет цикл for, количество блоков for равно количеству строк матрицы, количество столбцов матрицы равно количеству итераций блока for. M1..Mn – банки памяти, содержащие по одной строке матрицы. V – банк памяти, содержащий векторный операнд. R1..Rn – элементы вектора результата. Адресные входы банков памяти, тактовые сигналы и другие служебные проводники не изображены для упрощения понимания. Умножитель в такой реализации позволяет получить результат за 21 такт. Это тридцатикратное ускорение вычислений (в дискретном времени), которое в принципе не достижимо с использованием микропроцессора, даже в случае с параллельным программированием. Кроме того можно утверждать, что цикл for имеет некоторую реализацию, которую можно повторно использовать в других проектах как типовой функциональный блок в других проектах. Давайте ближе рассмотрим блок FOR. Его можно представить как в следующем виде (рис. 2).

В целях лучшего восприятия некоторые служебные сигналы не показаны. Сигнал IN_DATA попадает на блок инициализации INIT, где комбинационной логикой производится начальная инициализация переменных цикла. Блок проверки условия выхода из цикла TST сразу анализирует нужно ли выполнять цикл, в случае если нужно, по сигналу ST проинициализированные данные попадают на вход тела цикла BODY, управляющее устройство подает сигнал BS и ожидает сигнала BR, после чего итерация считается завершенной. Второй блок проверки условия выхода из цикла TST используется для проверки условия выхода между итерациями.

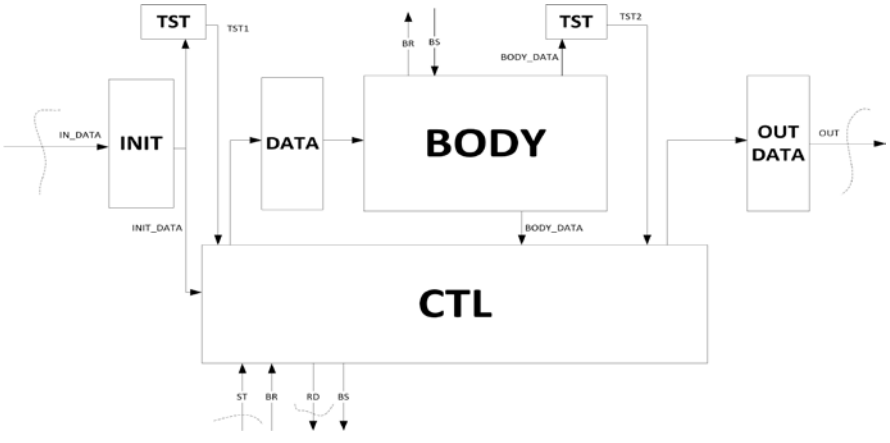

Рис. 2. **Общая схема блока циклирования for**

Блок проверки условия выхода из цикла TST сразу анализирует нужно ли выполнять цикл, в случае если нужно, по сигналу ST проинициализированные данные попадают на вход тела цикла BODY, управляющее устройство подает сигнал BS и ожидает сигнала BR, после чего итерация считается завершенной. Второй блок проверки условия выхода из цикла TST используется для проверки условия выхода между итерациями. Первый и второй блоки идентичны. После успешного выполнения всего цикла, взводится флаг RD, что означает, что цикл выполнился. Легко заметить, что набор входных и выходных сигналов тела цикла соответствует набору входных и выходных сигналов цикла в целом. Это позволяет создавать вложенные циклы произвольной степени вложенности. Диаграмма состояний управляющих сигналов выглядит следующим образом (рис. 3). Реализация такого конечного автомата осуществлена на языке Verilog.

Фильтрация изображения. Для примера рассмотрим фильтрацию изображения простейшим фильтром размытия (усреднения) изображения с ядром размером 3x3. Для фильтрации таким фильтром усредняется значение яркости на области 3x3, оно и является новым значением яркости центра области. Все изображение можно пройти с помощью двух циклов: один по строкам, а второй – по столбцам.

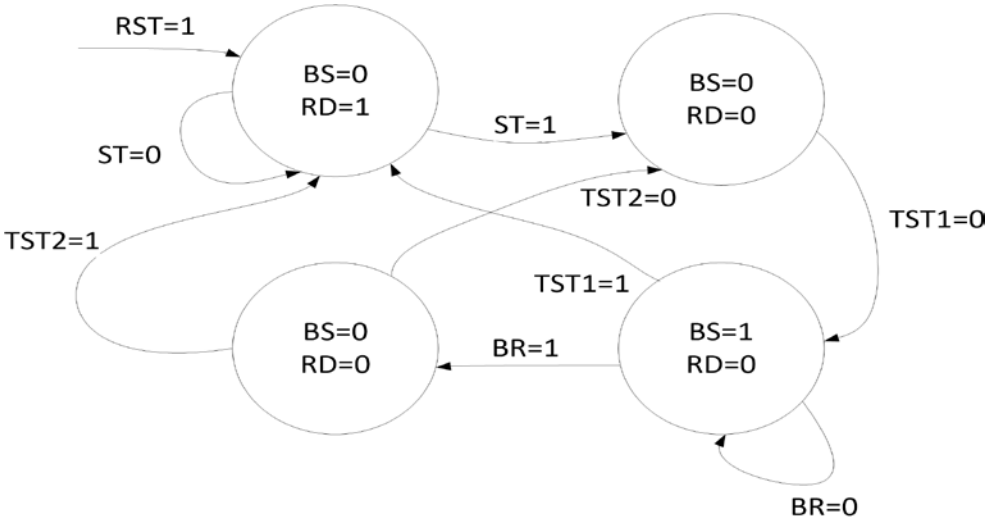

Рис 3. **Диаграмма состояния блока циклирования**

На реконфигурируемом логическом устройстве вычисление среднего уровня яркости блока клеток можно организовать за один такт (рис. 4).

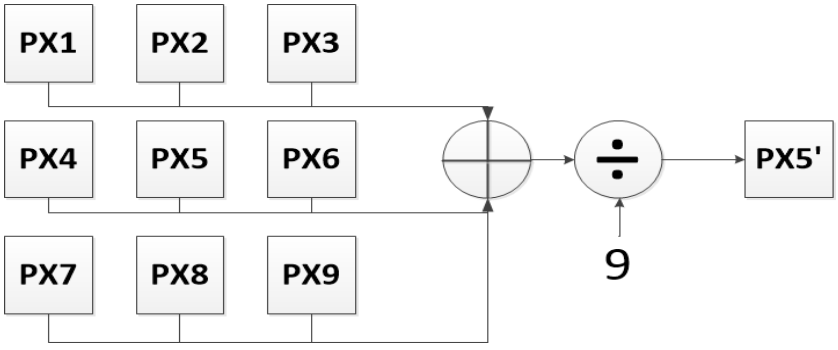

Рис 4**. Вычислитель среднего уровня яркости**

PX1..9 – значение яркостей пикселей, + – блок деления, ÷ – блок вычисление целочисленного частного, PX5’ – новое значение пикселя.

Для организации прохода по каждому из пикселей (кроме самых крайних, которые остаются неизмененными), используются рассмотренные раньше блоки циклирования, вложенные один в другой. Что подтверждает возможность повторного использования таких базовых блоков, которые аналогичны тем, которые присутствуют во многих языках программирования.

В ходе испытаний выяснилось, что для обработки изображения 16x16 пикселей таким алгоритмом, необходимо 16790 тактов для ПК на базе процессора Intel Core i7 и 1160 тактов для реализации на языке Verilog.

Выводы. Использование реконфигурируемой логики – это один из способов преобразования площади кристалла в вычислительную мощность. Притом, один из самых эффективных, так как есть возможность учесть специфические особенности задачи и при этом в некоторой мере шаблонизировать решение. Степень шаблонизации решения целиком зависит от разработчика. Таким образом при использовании реконфигурируемой логики при проектировании существует двумерное конструкторское пространство, где одна его ось – ось компромисса быстродействие-площадь, а вторая – затраты на проектирование-шаблонизацию (которая ведет к уменьшению производительности)[3]. К недостаткам использования реконфигурируемой логики можно отнести относительно низкие тактовые частоты. Таким образом хоть количество выполненной работы в дискретном времени для реконфигурируемой логики намного больше чем для классического микропроцессора, в метрическом времени этот выигрыш не такой большой. Автор выражает надежду, что в скором будущем ситуация изменится в лучшую сторону.

Список используемой литературы

1. J.L. Hennessy Computer Architecture, 5th Edition A Quantitative Approach / J.L.Hennessy, D.A. Patterson. – Morgan Kaufmann, 2011. – 856 p.

2. A. DeHon "Design patterns for reconfigurable computing,"/ J. Adams, M.deLorimier, N. Kapre, and others. – Field-Programmable Custom Computing Machines, 2004. FCCM 2004. 12th Annual IEEE Symposium on. – pp.13,23.

3. M. Holzer "Design Space Exploration for Real-Time Reconfigurable Computing,"/M. Holzer, B. Knerr, M. Rupp. – Signals, Systems and Computers, 2007. ACSSC 2007. Conference Record of the Forty-First Asilomar Conference on. – pp.1981,1985.

Рекомендовано до публікації проф., д.ф-м.н. Редько И.В.
Стаття надійшла до редакції 03.12.2013

ДОСЛІДЖЕННЯ МОЖЛИВОСТІ РЕАЛІЗАЦІЇ ЕФЕКТИВНОЇ ОБЧИСЛЮВАЛЬНОЇ АРХІТЕКТУРИ НА РЕКОНФІГУРОВАНІ ЛОГІЧНІ ПРИСТРОЇ

ЗАХАРЧЕНКО Т.Л.
Національний технічний університет України «Київський політехнічний інститут»

Мета. Стаття призначена довести ефективність застосування реконфігурованої архітектури задля прискорення обчислень.

Методика. Перевірка цієї гіпотези виконувалася експериментально. Для запропонованої публікації наведені приклади популярних процедур прискорення та отримано кількісне порівняння продуктивності обчислювальних архітектур з застосуванням реконфігурованої логіки та без. У якості прикладів використовувалась реалізація алгоритму фільтрації зображення та множення матриці на вектор. В результаті перевірки запропонованої гіпотези показано, що використання реконфігурованої логіки має сенс.

Результати дослідження. На згаданих прикладах досягнуто багаторазовий приріст продуктивності. Разом з цим слід усвідомлювати, що розглянуто ідеальний випадок, так як реалізацію алгоритмів обчислень на реконфігурованих пристроях проводила людина, а не комп'ютер, автоматичне перенесення алгоритмів на реконфігуровану логіку може давати гірші результати. Метрики в статті для оцінки якості переносу алгоритму на реконфігуровані пристрої використані вперше, а проблема прискорення обчислень та більш економного використання площі кристалу мікросхеми стоїть гостро.

Ключові слова: *комп'ютерна архітектура, прискорення, реконфігурована логіка, обчислювальна техніка.*

REVAMP STUDY OF RECONFIGURABLE LOGIC USAGE FOR COMPUTING ACCELERATION

ZAKHARCHENKO T.
National Technical University of Ukraine "Kyiv Polytechnic Institute"

The aim. The article is to prove effectiveness of reconfigurable logic usage for computing acceleration. In order to prove this hypothesis, some experiments were conducted.

Methodology. For this purpose, in the following paper examples of popular acceleration procedures were shown and quantitative comparison of different acceleration procedures with and without usage of reconfigurable logic devices was found. As examples an implementations of image filtering algorithm and implementations of matrix-vector multiplication were used.

As the result of the hypothesis examination, the author found that usage of reconfigurable logic makes sense. For examples mentioned above, there is significant productivity increase was achieved. However, one should understand that an ideal case was examined, when implementations of algorithms mentioned above was made by man, not computer. Automatic algorithm porting to reconfigurable logic may yield much worse results. Metrics used for porting to reconfigurable platform quality evaluation were using for the first time. A challenge of computing acceleration and effective utilization of chip area is very actual nowadays.

Keywords: *computing architecture, acceleration, reconfigurable logic, computing devices.*