ИССЛЕДОВАТЕЛЬСКИЙ ПРОЕКТ МАССОВО-ПАРАЛЛЕЛЬНОГО ПРОЦЕССОРА НА БАЗЕ МУЛЬТИТРЕДОВЫХ ЯДЕР СО СПЕЦИАЛИЗИРОВАННЫМИ УСКОРИТЕЛЯМИ

Эйсымонт Л.К [1], Никитин А.И.[2], Биконов Д.В.[3], Бражкин А.А.[4], Пеплов И.С.[5], Федоренко П.П.[6], Ермаков С.С.[7], Эйсымонт А.Л.[8], Комлев А.А.[9]

Цель статьи. *Представить исследовательский проект разработки отечественного массово-параллельного процессора mPX, образцы которого могли бы использоваться в качестве процессоров-ускорителей наряду с отечественными универсальными процессорами при построении вычислительных узлов суперкомпьютеров. Целями проекта являются также информационно-аналитическая работа и отработка технических решений для создания высокоскоростной элементно-конструкторской базы.*

Метод. *Формирование функционала процессора mPX и оценка его ожидаемой производительности, анализ по результатам информационно-аналитических исследований.*

Полученный результат*. Проработаны принципы работы создаваемого в рамках проекта процессора и вопросы его реализации. Базовый компонент процессора — тайл, образуемый 64-х тредовым ядром (mt-LWP), локальной памятью в 256 KB, подключенными специализированными ускорителями (SFU). Процессор mPX должен включать несколько сотен тайлов, соединенных внутрикристальной сетью, несколько линков межкристального взаимодействия, PCI-e интерфейс с хост-процессором. Интерфейс с внекристальной памятью пока не рассматривается. На данном этапе уделено внимание ядру mt-LWP, чему и посвящена статья. По идеологии архитектуры mPX похож на процессор Colossus английской фирмы Graphcore, ориентированный на задачи машинного обучения, однако процессор mPX — это платформа, на базе которой можно разрабатывать разные варианты.*

Ключевые слова: *мультитредовость, управление потоком данных, графовое представление программ, специализированные ускорители.*

DOI: 10.21681/2311-3456-2020-03-22-39

Введение

В работе [1] разбиралось проблемное состояние в области отечественной высокоскоростной ЭКБ. Такая ЭКБ применяется в суперкомпьютерах для решения научно-технических задач (HPC) и обработки больших данных (Big Data), а в последнее время все в большей и большей степени в суперкомпьютерах для решения задач искусственного интеллекта (AI). Технологии создания таких суперкомпьютеров, включая ЭКБ, в последние 10-15 лет оказались востребованными в системах высокоточного оружия и изделиях гражданского назначения – томографы, мобильные телефоны и компьютеры.

Для улучшения ситуации в работе [1] были предложены 5 стратегических архитектурных направлений исследований и разработок. В качестве направления N1 была определена разработка одного или нескольких вариантов замены GPU Volta (V100), который в отечественных системах в силу его уникально высокой производительности используется для решения задач от HPC до AI, но заменить его нечем.

GPU Volta (V100) – мощный специализированный процессор массово-параллельного типа на мультитредовых ядрах, который вдобавок является результатом 10-летней эволюции. Отставание от него отечественных образцов универсальных и специализированных процессоров по возможностям для задач из HPC до 30 раз (сравнение производительностей для FP64, 64-х раз-

1 Эйсымонт Леонид Константинович, кандидат физико-математических наук, научный консультант ЗАО НТЦ «Модуль», ФГУП НИИ«Квант», г.Москва, Россия. E-mail: verger-lk@yandex.ru
2 Никитин Анатолий Иванович, независимый эксперт, г.Москва, Россия. E-mail: nikitin95006@outlook.com
3 Биконов Дмитрий Владиленович, начальник сектора ЗАО НТЦ «Модуль», г.Москва, Россия. E-mail: d.bikonov@module.ru
4 Бражкин Алексей Алексеевия, инженер-программист, ЗАО НТЦ «Модуль», г.Москва, Россия. E-mail: a.brazhkin@module.ru
5 Пеплов Илья Сергеевич, начальник научно-исследовательской группы, ФГУП НИИ «Квант», г.Москва, Россия. E-mail: i.peplov@yandex.ru
6 Федоренко Петр Павлович, практикант ФГУП НИИ «Квант», студент, Национальный исследовательский университет МИЭТ, г.Зеленоград, Россия. E-mail: petyan.fedorenko@yandex.ru
7 Ермаков Семен Сергеевич, практикант ФГУП НИИ «Квант», студент, Национальный исследовательский университет МИЭТ, г.Зеленоград, Россия. E-mail: s_s_ermakov@mail.ru
8 Эйсымонт Алексей Леонидович, начальник сектора ЗАО НТЦ «Модуль», г.Москва, Россия. E-mail: eisymont@module.ru
9 Комлев Арсений Александрович, ведущий инженер ЗАО НТЦ «Модуль», г.Москва, Россия. E-mail: a.komlev@module.ru

рядных чисел с плавающей точкой), а по возможностям для задач AI – более 200 раз (для чисел FP16/ FP32). Появившийся в мае этого года GPU Ampere увеличил отрыв соответственно до 50 раз для задач HPC и от 500 до нескольких тысяч раз для задач AI. Еще сильнее отрыв в области AI произошел в связи с появлением в середине июля этого года второго поколения процессора Colossus.

В статье [2] рассмотрены три возможных архитектурных варианта создания отечественной замены GPU V100. Массово-параллельный процессор mPX – это третий вариант. Один или несколько таких процессоров могли бы быть подключены к отечественным универсальным процессорам в качестве ускорителей при построении вычислительных узлов суперкомпьютеров.

1. Высокоскоростные массово-параллельные процессоры, общее рассмотрение особенностей процессора mPX

При разработке архитектуры любого массово-параллельного процессора имеется ряд конкретных позиций (вопросов), по которым надо принять архитектурные решения. Далее выделены семь таких позиций. Рассмотрим, какие архитектурные решения принимаются в других процессорах из числа наиболее успешных. Сведения о них даны в таблице 1.

Процессоры GPU Volta [3, 4] и Ampere [5], а также PEZY-SC2 (Япония [12]) ориентированы на решение широкого круга задач из областей HPC, BigData и AI.

Процессор Colossus (фирма Graphcore [9]) ориентирован только на задачи машинного обучения (ML-задачи, область AI), за счет этого выигрывает на них у GPU [10]. Более того, Colossus реализует новую архитектурную идеологию организации подсистемы памяти и выполняемых параллельных программ [7, 8], которая обеспечивает большую производительность и энергоэффективность. Хоть разработчики Colossus и подчеркивают, что он создан для задач ML, но очевидно, что его можно применить и для решения других задач. Например как специализированный процессор [6] для решения задач информационной безопасности. Во всяком случае, символично в этом плане выбранное название процессора и его посвящение памяти Томми Фловерса (Tommy Flowers) [11].

Таблица 1.

Сравнение основных характеристик массово-параллельных процессоров

Характеристика	Массово-параллельные микропроцессоры			
	GPU Volta (2017)	GPU Ampere (2020)	Colossus (2018)	PEZY-SC2 (2017)
Технология	12 нм	7 нм	16 нм	16 нм
Площадь кристалла	815 мм2	826 мм2	~ 800 мм2	620 мм^2
Количество транзисторов	21.1x10^9	54.2x10^9	23.6x10^9	9x10^9
Частота	1.455 GHz	1.410 GHz	1.6 GHz	1 GHz
Потребление	300 W	400 W	120W	130-180W
Производительность (FP16)	31.4 TF	78 TF	н/д	16.4 TF
Производительность (FP32)	15 TF	19.5 TF	31.3 TF	8.2 TF
Производительность (FP64)	7.4 TF	9.7 TF	-	4.1 TF
Производительность TC SFU (FP16/FP32)	120 TF	312(624) TF	124.5 TF	-
Производительность TC SFU (INT8/INT32)	-	624 (1248) TOPS	-	-
Количество управляющих CPU	-	-	-	8xMIPS 64 P6600
Организация иерархии «кластер-тайл-ядро»	6 GPC 14 SM/GPC 4 ядра/SM	7 GPC 16 SM/GPC 4 ядра/SM	2x8x19 blocks 4 tiles/block 1 core/tile	128 City 4 Village/City 4 ядра/Village

DOI:10.21681/2311-3456-2020-03-22-39

Характеристика	Массово-параллельные микропроцессоры			
	GPU Volta (2017)	**GPU Ampere (2020)**	**Colossus (2018)**	**PEZY-SC2 (2017)**
Количество тайлов	84 (SM)	108 (SM)	304 (blocks)	512 (Villadge)
Количество ядер	336	432	1216	2048
Мультитредовость ядра	32 A-треда x 32 S-треда	32 A-треда x 32 S-треда	6 A-тредов	8 A-тредов
Вычислительные ресурсы ядра	16 FP32 16 INT32 8 FP64 8 LD/ST SFU 2 TPU, TPU: **128** FP16.32/clk	16 FP32 16 INT32 8 FP64 8 LD/ST SFU TPU: **1024** FP16.32/clk, + варианты	ALU AMP: 16 FP32/clk 64 FP16.32/clk	ALU: 8 FP16/clk 4 FP32/clk 2 FP64/clk нет
Локальная память ядра	L0i – 12 KB	L0i – 12 KB	SMEM-256 KB	L1i — 2 KB SMEM — 16 KB
Общая память ядер тайла	L1i – 128 KB L1d – 128 KB	L1i – 128 KB L1d – 192 KB	-	L1d — 2x2 KB
Общая память тайлов в кластере	-	-	-	L2d – 64 KB
Общая память кластеров кристалла	L2 – 6144 KB	L2 – 40960 KB	-	LLC – 40960 KB
Внутрикристальная память (V, BW)	32 MB	81 MB	304 MB, 45 TB/s	47 MB
Внекристальная память (тип, V, BW)	HBM 32 GB, 0.8 TB/s	HBM 40 GB, 1.6 TB/s	-	HBM – 2 TB/s DDR4 – 102GB/s
Внутрикристальная сеть (тип, BW)	н/д	н/д	2xSwitch 8x8 8 TB/s	н/д
Внешние линки (N, duplex BW линка)	6 x 50GB/s	12x50 GB/s	10 x 64GB/s	-
Интерфейсы host-CPU (N, тип)	PCIe-3.0	PCIe-4.0	PCIe-3.0	есть, н/д

1.1. Количество ядер и их вычислительная мощность

В массово-параллельных процессорах количество процессорных ядер особенно велико, от нескольких сотен до тысяч, в таблице 1 это строка «Количество ядер». Очевидно, что количество ядер зависит от площади кристалла и используемой технологии, но если обратить внимание на строку «Вычислительные ресурсы», то можно заметить зависимость и от имеющихся в ядре функциональных устройств, т.е. мощности ядра. Возникает вопрос, насколько оправдано иметь мощные ядра?

Это старая архитектурная проблема выбора между «одним слоном и тысячами мышей». GPU Volta и Ampere с одной стороны, а Colossus и PEZY-SC2 с другой – примеры таких разных подходов.

Архитектура GPU сложилась более 10 лет назад, потом шло ее эволюционное развитие. В проекте GPU Echelon, он рассмотрен в [2] в сравнении с GPU Volta, была сделана принципиальная попытка упростить мультитредовое ядро GPU, но при этом значительно увеличить их количество и одновременно увеличив вдвое асинхронную мультитредовость каждого такого ядра. Это ожидалось увидеть в 2020 году, но в GPU Ampere это не произошло. Важно, что мысли по этому поводу в NVIDIA были! Пока не сложилось, но это будет.

Если еще обратить внимание на эволюцию GPU, то видно, что GPU Ampere по сравнению с GPU Volta, значительно усилился по возможностям специализированного устройства выполнения тензорных операций TPU

(см. строку «Вычислительные средства ядра»).

По-видимому, такие изменения GPU говорят о тенденции развития в сторону более простых ядер, но в большем количестве и усиленных специализированными функциональными ускорителями. Пример такого решения – ядра процессора Colossus.

Эффективная работа даже таких ядер, особенно их ускорителей, выполняющих много операций за такт, — это проблема, ведь их надо успевать обеспечивать данными и получать результаты. Это сделать сложно для однопотоковой программы, поэтому в ядрах используется выполнение множества вычислительных процессов (тредов) одной программы – мультитредовость. В строке «Мультитредовая организация ядра» показан реализуемый тип и степень мультитредовости: A-треды – асинхронные треды; S-треды – синхронные треды, выполняемые по одной и той же команде действия, но над разными данными.

При разработке mPX принято решение выбрать 64-х тредовые ядра mt-LWP средней мощности. Для каждого такого ядра предусмотрено подключение через стандартный интерфейс одного или нескольких специализированных ускорителей SFU.

Возможна критика по поводу того, что для массово-параллельного процессора нужен кристалл большой площади, а тут отечественные возможности явно ограничены. Это не совсем так. Например, известно о заказе процессора по технологии 16 нм с площадью 400 мм^2. Это половина площади процессора Colossus (16 нм). Если на этой же площади реализовать половину процессора Colossus, то для решения нейросетевых задач был бы получен процессор с производительностью 15 TF даже на частоте 1.6 GHz.

1.2. Организация иерархии внутрикристальной сети

В массово-параллельных процессорах ядра подключены к внутрикристальной сети, она обычно имеет иерархическую структуру. Типична трехуровневая иерархия (см. строку «Организация иерархии «кластер-тайл-ядро») — несколько ядер посредством прямых соединений образуют тайл, тайлы образуют кластер, кластеры образуют процессор.

Об используемых в GPU и PEZY-SC2 сетях неизвестно, но во время их создания были популярны сети передачи сообщений с топологиями «двунаправленное кольцо», 2D-тор, 2D-сетка и другие, даже с ограниченным диаметром типа “dragonfly”.

В последние несколько лет стал заметен интерес к иерархическим сетям на коммутаторах. Пример такой сети – сеть процессора Colossus, если судить по результатам ее исследования на тестах-микробенчмарках [9]. Для этого процессора эффективность внутрикристальной сети из-за большого количества ядер особенно важна, и она достигнута, что подтверждают работы [9, 10]. В этой сети для одного процессора средняя задержка передачи сообщения между ядрами составляет 130 нсек (208 тактов), но внутри одной секции из 19 4-х ядерных блоков (всего 76 ядер, второй уровень иерархии) она составляет 37-59 нсек (59-95 тактов). Пропускная способность внутрикристальных соединений – 6.3 GB/s. Под нагрузкой, когда одновременно выполняется много передач сообщений между ядрами, эти показатели не сильно меняются, деградация задержек приблизительно в 1.2 раза.

В проекте mPX решения по организации внутрикристальной сети еще не приняты, но ее организация в процессоре Colossus рассматривается как приоритетная.

1.3 Организация подсистемы памяти

Для массово-параллельных процессоров эта архитектурная позиция наиболее чувствительна, можно выделить два крайних варианта решений: первый, когда к процессору подключена внешняя память с высокой пропускной способностью, а внутрикристальная память – это иерархия кэш-памятей (используется в GPU Volta и Ampere); второй, когда внешней памяти нет, а используется много модулей внутрикристальной памяти (используется в Colossus). Второй подход не такой общий и не так прост для программирования, но обладает высокой энергоэффективностью и обеспечивает на порядок большую суммарную пропускную способность памяти. Это убедительно объясняется в докладах [7, 8], обосновывающих принятые в Colossus решения, да и заметно по характеристикам (см. строки «Локальная память ядра», «Внутрикристальная память ядра», «Внекристальная память ядра»).

Первый подход уже оказывается явно недостаточен, это видно по доработкам при создании GPU Ampere. В этом процессоре значительно увеличен объем внутренних кэш-памятей, особенно уровня L2. Более того, в L2 используется компрессия при хранении данных, это увеличивает его эффективный объем до 4-х раз и во столько же раз повышает эффективную пропускную способность обменов с внешней памятью. Дополнительно введенная возможность запрещения выталкивания наиболее часто используемых строк в L2, повышает до 2.5 раз производительность вычислений.

При разработке mPX выбран второй подход, но использование внекристальной памяти не исключалось, даже была введена дополнительная возможность доступа к памяти через расширенный 62-х разрядный адрес и глобальное адресное пространство.

1.4. Специальные вычислительные ресурсы ядра, типы данных и модели вычислений

Применение специальных ускорителей для повышения производительности было хорошо известно, однако появление тензорного процессорного блока (TPU) в составе GPU Volta и резкое повышение за счет этого производительности удивило. В каждом ядре этого процессора были добавлены два TPU, которые могли выполнять по 128 операций над FP16 и FP32 за такт. Это повысило производительность с 31.4TF (FP16) и 15TF (FP32), развиваемой на обычных устройствах, до 120 TF.

В GPU Ampere главные улучшения сделаны именно в блоке TPU, что повысило его производительность по отношению GPU Volta от 2.5 до 20 раз, в зависимости от используемых типов данных. Разнообразие данных, обрабатываемых в TPU, значительно расширено, введены даже операции над целыми в 8, 4 и 1 бит, обеспе-

DOI:10.21681/2311-3456-2020-03-22-39

чена и работа и с числами FP64. Введена возможность работы TPU с разреженными матрицами, подготовленными по специальным алгоритмам, но представленными в специальном сжатом формате. Результат одной и другой доработки TPU — в строках «Производительность TC SFU (FP16/FP32)" и «Производительность TC SFU (INT8/INT32)". Взятый в скобки показатель производительности – это производительность, которая бы достигалась, если бы вычисления выполнялись над плотно заполненными матрицами с нулевыми коэффициентами.

Введение мощных блоков тензорных вычислений стало типичным для массово-параллельных процессоров и достигло таких предельных показателей, что появились сомнения в целесообразности дальнейшего роста. Например, в TPU GPU Ampere за такт выполняется уже 1024 операций над FP16 и FP32. Такие блоки трудно обеспечить данными, требуется память с исключительно высокой пропускной способностью, велико энергопотребление.

Варианты решений – оптимизация используемых типов данных, что и было сделано в GPU Ampere, второй подход – изменение в организации вычислений, что было реализовано в процессоре Colossus и объясняется в [7, 8]. Отмечается, что был сделан сознательный отказ от тензорных вычислений в пользу вычислений над графовыми представлениями сетей. Это означает представление вычислений в виде статических графов потоков данных, в которых в узлах находятся вычислительные процедуры, а дуги – это информационные связи для передачи данных. Активизация вычислений в узлах происходит по готовности данных, поставляемых в узлы по дугам. Кстати, такая модель была поддержана в свое время в процессорах разрабатывавшегося суперкомпьютера «Ангара» [16, 17, 18], сокращенное название этой архитектуры было MTDF – мультитредовая с управлением потоком данных.

В системе программирования CUDA для GPU такая модель также обеспечивается, а в GPU Ampere ее поддержка даже была усилена — в 11 раз повышена пропускная способность памяти при выполнении атомарных операций с ней.

В процессоре mPX для поддержки графовых моделей вычислений были введены такие средства, как почтовые ящики и передача сигналов между тредами.

Если возвратиться к теме ускорителей в ядрах, то надо заметить, что TPU были введены для повышения вычислений на нейронных сетях. Очевидно, что для других областей приложений могут потребоваться другие специализированные функциональные ускорители [19], в то время, как мультитредовое ядро окажется инвариантом и обеспечит эффективную работу с ними по данным. Поскольку процессор mPX рассматривается как платформа для разработки ее модификаций для разных прикладных областей, то в нем был зафиксирован стандартный интерфейс со специализированными ускорителями, который базируется на передаче сигналов и локальной памяти с высокой пропускной способностью.

1.5. Межкристальные линки

Межкристальные линки в коммерчески доступных микропроцессорах фирм AMD, IBM и Intel появились более десяти лет назад. Благодаря им появилась возможность разрабатывать многосокетные (многопроцессорные) платы, которые оказались в свое время исключительно важным явлением. Такие узлы одновременно оказались интереснейшим объектом исследований, а их освоение произошло с определенными трудностями. В работах по их освоению довелось поучаствовать и авторам данной статьи [22, 23].

Приложения, связанные с нейронными сетями, настолько динамично растут по требованиям к производительности, что оказалось актуальным строить на основе массово-параллельных процессоров специализированные суперкомпьютеры. По этой причине в таких процессорах стало важным наличие большого количества высокоскоростных межкристальных линков (см. строку «Внешние линки»).

Наивысшее качество по количеству линков и их дуплексной пропускной способности имеют GPU Ampere и Colossus. Для линков Colossus в [9] имеются результаты их исследования на тестах-микробенчмарках, демонстрирующие их высокое качество. Например, если нет нагрузки от множества одновременных обменов, то задержка передачи сообщения между ядрами разных процессоров на одной плате составляет лишь 633 нсек (это всего лишь в 4 раза хуже, чем для ядер внутри кристалла), а пропускная способность составляет 5.46 GB/s, что вообще мало отличается от пропускной способности в 6.3 GB/s между ядрами внутри кристалла. При нагрузке ситуация объективно сложнее. Задержка возрастает до 2524 нсек, а вот пропускная способность падает в 125 раз. Это естественно, поскольку взаимодействующих через линк ядер много, а он один. Из этого просто надо сделать вывод при разработке программного и прикладного программного обеспечения.

В процессоре mPX вопрос внешних линков пока открыт. Отечественных блоков (IP-блоков) для построения линков такого уровня нет, хотя есть менее мощные линки, применяемые в NeuroMatrix NM6408MP (1879ВМ8Я) [20, 21] (далее — NM6408MP), что можно использовать в экспериментах и макетах.

1.6. Интерфейс с хост-процессором

Некоторые массово-параллельные процессоры включают обычные ядра для решения задач управления и реализации вычислений со скалярами. Например, в PEZY-SC2 имеется 8 ядер с MIPS-архитектурой, а в проекте GPU Echelon предполагалось иметь 8 обычных ядер. В других процессорах это не используется, зато в любом из них имеется высокоскоростной PCI-интерфейс с управляющим host—процессором. Такой интерфейс используется при построении узлов суперкомпьютеров с множеством массово-параллельных процессоров. Известны 8-ми и 16-ти процессорные платы с GPU Volta и Ampere, а также платы с 16-ю процессорами Colossus. Это очень мощные узлы, они уже применяются, например, даже позволили NVIDIA построить суперкомпьютер на GPU Ampere с производительностью около 700 петафлопс.

В процессоре mPX предполагается иметь один PCI-интерфейс. Опыт его применения получен при разра-

ботке системы параллельного программирования для NM6408MP[21].

1.7 Разнообразие масштабируемости

Наличие мощных линков и PCI-интерфейсов, материнских плат с большим количеством PCI-разъемов позволяют создавать вычислительные узлы. Такие узлы объединяются друг с другом при построении суперкомпьютеров уже через обычные межузловые сети типа, например, Mellanox Infiniband.

Между тем, проявляется интерес к вычислительным узлам “host-less” типа, т.е. без хост-процессоров. В таких узлах соединения между массово-параллельными процессорами реализуются через их линки напрямую.

В проекте по mPX рассматривается разработка многопроцессорных плат с host-процессорами и без хост-процессоров, а также их системного программного обеспечения. Отработка решений по этим вариантам возможна на базе имеющегося 21-ядерного гибридного процессора NM6408MP с четырьмя линками, а также разработанной системы параллельного программирования [20, 21].

2. Массово-параллельный процессор mPX и ядра mt-LWP

После рассмотрения массово-параллельных процессоров и общей характеристики принятых в процессоре mPX архитектурных решений перейдем к его более детальному описанию.

2.1. Структура и главные особенности

Структура процессора mPX дана на рис.1а. Тайлы объединены в секции, а секции соединяются коммутатором. Как отмечалось, такое решение пока принято условно.

Структура тайла изображена на рис.1б. Пока принят простой вариант, когда в тайл входит одно ядро mt-LWP, локальная память (ЛП) и блок специальных устройств-ускорителей (SFU_1, SFU_2 , ...SFU(NI)). SFU(NI) – это устройство связи с внутрикристальной сетью, оно включено в блок ускорителей и имеет стандартный интерфейс с mt-LWP через блок Интерфейс SFU этого ядра (см. раздел 2.3).

Ядро mt-LWP содержит четыре секции, в каждой из которых имеются 16 тредовых устройств (TU), на которых могут выполняться процессы-треды. Каждое TU – это набор регистров общего назначения, слов локальной памяти треда и системных регистров треда. Переход от выполняемой на одном TU программы на программу другого TU происходит без каких-либо накладных расходов на запоминание/восстановление состояний программ.

Каждая секция содержит общее для всех TU секции АЛУ для выполнения простых арифметических и логических операций, а также умножитель и блок связи с локальной памятью ЛП. Переключение между выполнением программ разных TU производятся аппаратно либо при возникновении задержки выполнения команды на TU, либо в результате выполнения специальных команд работы с тредами.

Секции TU взаимодействуют друг с другом, а также с блоком общих специальных регистров БОР, блоком Интерфейса SFU через однотактовую «шину» межтредовых сообщений (Шина МТС), реализуемую как коммутатор. В каждой секции работу с шиной по приему/выдаче сообщений и выполнению специальных операций с TU выполняют блоки специальных операций БСО.

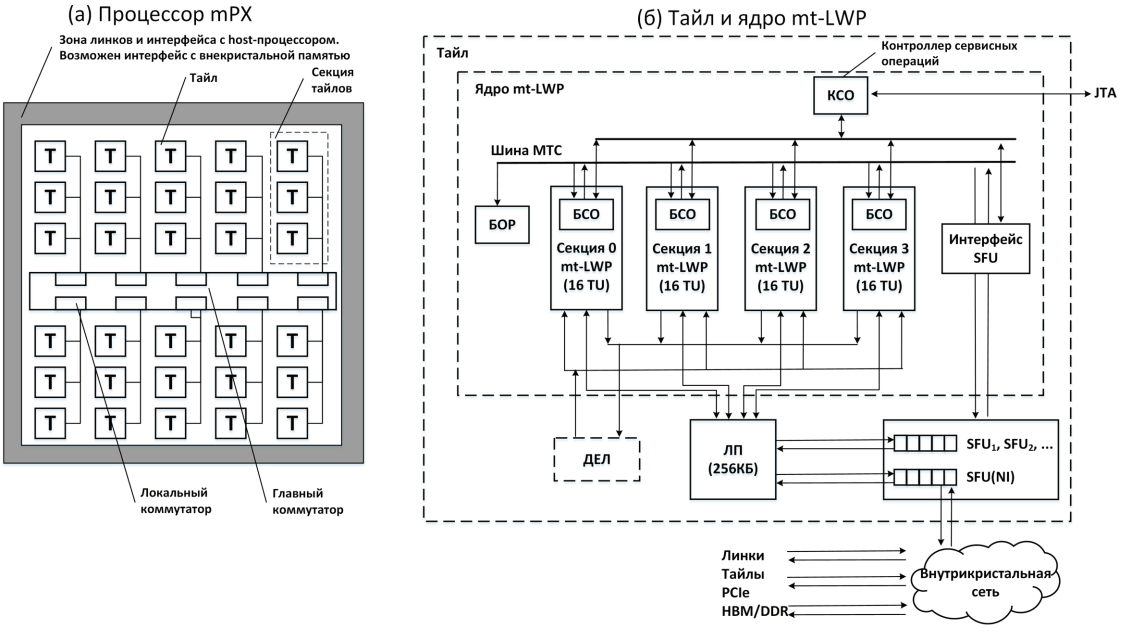

Рис.1. Структура массово-параллельного процессора mPX и мультитредового ядра mt-LWP

DOI:10.21681/2311-3456-2020-03-22-39

Главные особенности ядра mt-LWP состоят в следующем:

- это 64-х потоковый (тредовый) RISC-процессор, обрабатываемые данные в ядре mt-LWP: 32-х и 16-разрядные целые со знаком, двоичные коды, байты и биты, обработка чисел с плавающей точкой и других типов данных в подключенных специальных устройствах – ускорителях (SFU, их тип не фиксируется);
- имеются 12 форматов команд, два из них длинноформатные (32 разряда), остальные – 16-разрядные, количество команд 141, из них 10 длинноформатные (см. рис.2);
- в программе каждого треда пользователю доступны 8 32-х разрядных регистров общего назначения, с которыми можно работать и как с 16 16-разрядными регистрами, 32 слова сверхбыстрой памяти треда (ЛПТ), несколько системных регистров;
- команды треда выполняются по дисциплине “in-order”, информационные зависимости команд отслеживаются аппаратно, в одном треде возможно выполнение до 8 обращений к памяти одновременно, т.е. всего в mt-LWP допускается выдача до 512 незавершенных обращений к памяти, в треде возможна барьерная синхронизация для завершения выданных им обращений к памяти;
- для взаимодействия тредов доступна аппаратная поддержка передачи сигналов, почтовые ящики с full/empty-битами, реализуемые на словах сверхбыстрой памяти тредов, счетчики барьерной синхронизации, атомарные операции с памятью;
- взаимодействие тредов обеспечивается внутри mPX, а также в сети из таких процессоров, построенной с применением линков mPX;
- работа с памятью обеспечивается в обычном и расширенном режиме, основные возможности:
- в обычном режиме работа ведется над локальной памятью тайла объемом 256 KB через 16-разрядный адрес, адресация данных до 32-х разрядного слова, а команд – до полуслова, если подключена внекристальная память – то используется 32-х разрядный адрес;
- в расширенном режиме аппаратно-программно поддержан доступ к памяти по 62-х разрядному адресу до байта (64-х разрядный адрес за вычетом двух теговых битов, см. раздел 2.4), обеспечивается работа с глобальным логическим адресным пространством (GAS) объемом до 4.6 экзабайт, на которое отображена память всех mPX, образующих мультипроцессорную систему;
- обеспечивается асинхронная пересылка блоков разной фиксированной длины между локальной (внекристальной) памятью и памятью, доступной через GAS;
- обеспечен унифицированный интерфейс с SFU, основанный на передаче сигналов.

2.2. Система команд ядра mt-LWP

Для экономии обращений к памяти при выборке команд используется их очень плотная 16-разрядная кодировка. На рис.2 приведено одно из трех деревьев кодировки команд (дерево А) с примерами команд на языке Ассемблера, а также 6 форматов команд из имеющихся 12. В этих форматах: КОП – основной код операции команды; F4, F8 – дополнительные коды операций команд, их использование видно в дереве кодировки А; r1, r2, r3 – это ссылки на регистры общего назначения TU; imm8, imm32, imm256 – это непосредственно задаваемые в команде константы соответствующей разрядности.

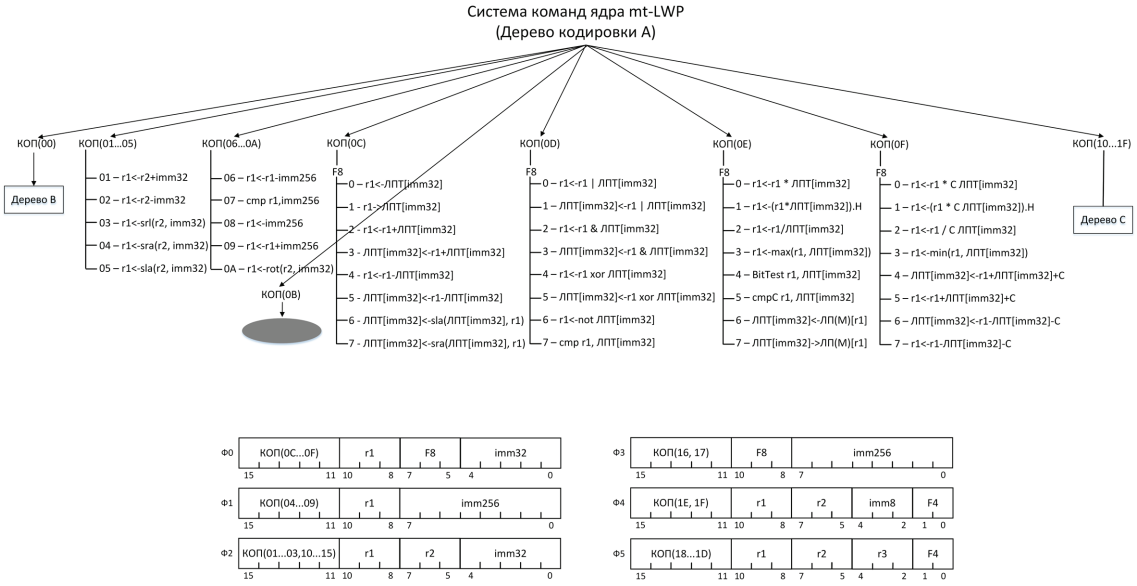

Рис.2 Форматы и кодировка команд

Набор команд mt-LWP можно разделить на четыре основных группы: арифметико-логические команды; команды пересылки регистров и операций с памятью; команды передач управления и синхронизации; команды управления тредами и специальных операций. Есть еще несколько длинноформатных команд разного типа, особой смысловой нагрузки они не имеют.

Группа арифметико-логических команд – это обычные команды RISC-процессора, которые образуют следующие подгруппы: операции сложения/вычитания, сравнения с 8-разрядной константой (6 команд); арифметические операции сложения/вычитания, сравнения (21 команда, операнды – регистры общего назначения и слова локальной памяти треда, обозначены на рис.2 как ЛПТ[imm32]); логические операции (11 команд); операции арифметических и логических сдвигов, циклического сдвига (12 команд); арифметические операции умножения и деления (9 команд); операции над полусловами, байтами и битами (12 команд). Всего в этой группе 71 команда из 141, имеющихся в mt-LWP.

При работе с полусловами регистров их можно адресовать непосредственно по номерам. Например, можно использовать команду вида

$$r1(Np) <- r1(N1) + r2(N2)$$

где r1 и r2 – это ссылки собственно на 32-х разрядные регистры, а Np, N1 и N2 – это указатели на их правую (=0) или левую (=1) половину.

При работе с байтами их также можно нумеровать в регистре справа налево, от 0 до 3-х. Имеется команда пересылки байта одного регистра в байт другого, сравнения заданных байтов разных регистров.

При работе с битами регистров их можно проверять по маске, упаковывать и распаковывать по маске, определять номер первого единичного разряда слева, производить зеркальную перестановку битов регистра.

Короткие 16-разрядные команды с плотной кодировкой повлияли на ограничения по типам обрабатываемых данных в арифметических командах mt-LWP. Сейчас это целые со знаком и двоичные коды. Другие типы данных оставлены для реализации в SFU. Не трудно представить критику, что хорошо бы иметь непосредственно в ядре mt-LWP еще и операции над другими типами данных, например, над числами с плавающей точкой.

Это можно исправить, сохранив 16-разрядную кодировку команд и сохранив многие аппаратные решения, введя в разных секциях mt-LWP немного отличающуюся по обрабатываемым данным систему команд, но сохранив кодировку. Например, в такой измененной системе команд 32-х разрядные целые со знаком можно исключить, заменив их 32-х разрядными числами с плавающей точкой, и сохранив двоичные коды для адресной арифметики.

Такой подход использования разнородных секций в mt-LWP, а также и возможность применения разных SFU в разных тайлах благодаря зафиксированному интерфейсу с SFU не противоречит базовой идее, заложенной в процессорах, примерами которых являются mPX, Colossus и Celerity [13, 14, 15]. Эту базовую идею китайские специалисты удачно охарактеризовали как «зеленый великан с асимметричными потоками», где термин «зеленый» означает «энергоэффективный».

Группа команд пересылки регистров и операций с памятью содержит следующие подгруппы: пересылки между регистрами общего назначения, словами локальной памяти треда и системными регистрами (10 команд); пересылки между регистрами и памятью (12 команд); пересылки между участками памяти (2 команды). Всего 24 команды.

В системе команд mt-LWP заложены возможности работы: с локальной памятью ЛП ядра (16-разрядный адрес); внекристальной памятью М (32-х разрядный адрес, хотя доступ к ней в конкретной реализации mPX может и не поддерживаться); произвольной физической памятью mPX и вычислительной системы в целом через 62-х разрядный адрес и глобальное адресное пространство (GAS, см. раздел 2.4).

Обмен данными с ЛП и М производится 32-х разрядными словами, имеются атомарные операции с памятью.

Обмен данными с памятями, адресуемыми через GAS, может быть байтами, полусловами и словами. Кроме этого, возможна выдача запросов на асинхронную пересылку блоков памяти. Такая возможность важна для обменов между ЛП или М конкретного mPX и какой-либо физически доступной памятью, которой может быть: ЛП своего тайла; ЛП удаленного тайла этого же процессора mPX; память М своего mPX; ЛП или М удаленного процессора mPX вычислительной системы.

Концепция работы с памятью предполагает, что основная работа из mt-LWP будет с ЛП своего тайла. Остальные возможности принципиально иметь, но требования по эффективности их реализации не предъявляются. Допускается даже программно-аппаратная реализация работы с памятью через GAS. Жизненность такой концепции на реальных прикладных алгоритмах в проекте mPX предполагается исследовать в первую очередь, для чего начата разработка имитационной модели с потактовой отработкой выполнения команд.

Напомним, что в выполняемом на TU треде может быть выдано до 8 обращений к памяти, которые выполняются асинхронно по отношению к другим командам треда, но информационная зависимость с этими обращениями к памяти других команд отслеживается аппаратно.

Группа команд передач управления и синхронизации содержит следующие подгруппы: команды переходов (6 команд); команды синхронизации (12 команд). Всего 18 команд.

В подгруппу команд переходов входят команды условного и безусловного перехода, команда обращения к подпрограмме и выхода из нее; управления циклом; команда обращения к обработчику прерывания в треде и выхода из него.

В подгруппу команд синхронизации входят две команды специфических барьеров: ISB – ожидание завершения выданных на выполнение команд данного треда; DMB – ожидание завершения выданных ранее до этой команды в треде обращений к памяти.

Имеются также три команды работы с сигналами: Send_Signal – посылка в адресуемый TU сигнала с за-

DOI:10.21681/2311-3456-2020-03-22-39

данным номером; Wait_Signal – ожидание получения в текущем TU сигналов, заданных в этой команде битовой маской; Wait_Any_Signal – ожидание получения в текущем треде хотя бы одного сигнала из заданных битовой маской.

Каждое TU имеет 16-разрядный системный регистр TSIG, именно в нем битами отображаются получаемые TU сигналы. Дополнительно имеется системный 16-разрядный регистр TESC, биты которого соответствуют сигнальным битам TSIG и указывают, при поступлении каких сигналов должно возникнуть прерывание «Получение сигнала». Таким образом, в TU реализуются два способа получения сигнала: активный – по прерыванию; пассивный – по командам Wait_Signal и Wait_Any_Signal.

Наличие команд работы с сигналами значительно облегчает реализацию статических графовых моделей вычислений с управлением потоком данных. Напомним, что на такие модели ориентирован процессор Colossus и есть их поддержка в GPU.

Для взаимодействия тредов дополнительно введены команды работы с почтовыми ящиками, которые реализуются на словах локальной памяти треда (ЛПТ). Это подтвержденное многолетним опытом средство одновременно синхронизации и передачи данных между процессами. Каждое из слов ЛПТ снабжено дополнительным full/empty теговым битом. Имеются две команды работы с этими ящиками.

Команда fe_write – запись в заданное слово ЛПТ удаленного TU, если его теговый бит в состоянии empty, после чего установка его в состояние full. Если состояние тега full, то производятся попытки повторения этой записи, пока либо теговый бит не станет empty, либо будет достигнуто заданное в TU возможное количество повторений такой записи. Во втором случае возникнет соответствующее прерывание.

Команда fe_read – чтение из заданного слова ЛПТ текущего TU, если его теговый бит в состоянии full и установка его после этого в состояние empty. Если состояние тега empty, то производится повторение чтения, пока теговый бит не станет full, либо не будет достигнуто заданное в TU ограничение на допустимое количество повторений чтения. Во втором случае возникнет соответствующее прерывание.

Почтовые ящики, как и сигналы, могут использоваться для реализации статических графовых моделей вычислений, а также для других моделей взаимодействий.

В подгруппе команд синхронизации еще имеются три команды для работы со шкалами сигналов выходных и входных каналов Интерфейса SFU (см.раздел 2.3). Это команды типа посылки и ожидания сигналов для тредов.

Группа команд управления тредами и специальных операций содержит следующие подгруппы: команды работы с тредами (8 команд); команды барьерной синхронизации тредов (4 команды); специальные команды системного уровня и отладки (6 комад). Всего 18 команд.

Команды работы с тредами позволяют выполнять с TU и, соответственно, с выполняемым на нем процессом-тредом (если он есть) разные действия, включая диагностику TU. Кроме того, имеются два режима выполнения тредов на TU – USER (режим пользователя) и SYSTEM (системный режим, привилегированный и защищенный). В эту подгруппу входят следующие команды.

Reserve – резервирование заданного в команде количества TU для последующего их использования при выполнении тредов, такие TU получают состояние IDLE.

Create – порождение процесса-треда на свободном TU из множества ранее зарезервированных, в команде задаются адреса начала программы и области данных этого треда. TU с порожденным на нем тредом переходит в состояние PASSIVE (нет выборки и выполнения команд, это точка останова, тред не реагирует на прерывания).

Delete – освобождение адресуемого в команде удаленного TU от выполняемого им процесса-треда. Это TU должно находиться в состоянии PASSIVE, а в результате выполнения команды переходит в состояние IDLE и возвращается в множество зарезервированных TU.

Activate – активация в адресуемом в команде удаленном TU треда, который должен находиться в состоянии PASSIVE, т.е. на точке останова. В результате успешного выполнения команды в этом треде возобновляется выборка и выполнение команд, он переводится вместе с его TU в состояние ACTIVE.

Passivate – в адресуемом в команде TU (удаленном или текущем) происходит останов выборки и выполнения команд, фиксируется точка останова. Пассивируемый тред должен находиться в состоянии ACTIVE или WAIT (ожидание сигнала) и переводится в состояние PASSIVE. Если в треде происходит обработка прерывания или он работает в привилегированном режиме, то перед переводом в состояние PASSIVE ожидается доработка прерывания и/или выход из привилегированного режима.

Имеются три команды, которые позволяют диагностировать адресуемое TU, работая с ним на физическом уровне: Execute – выполнение из текущего TU заданной двоичным кодом команды в адресуемом TU, если он находится в состоянии PASSIVE; Read – чтение-произвольного регистра из адресуемого TU; Write – запись в произвольный регистр адресуемого TU.

Команды подгруппы средств барьерной синхронизации в mt-LWP немного отличаются от обычно используемых. В программе для mt-LWP треды могут работать с до 8 счетчиками барьерной синхронизации, каждый из них может использоваться для определяемого пользователем подмножества тредов программы. Имеются команды: Reserve_Barrier – резервирование заданного в команде количества счетчиков барьерной синхронизации; Create_Barrier – активация свободного барьерного счетчика из ранее зарезервированных; Barrier – выход в барьерную синхронизацию текущего TU по заданному в команде счетчику барьерной синхронизации; Delete_Barrier – освобождение указанного в команде счетчика барьерной синхронизации если он активирован, но не участвует в данный момент в барьерной синхронизации тредов, после успешного выполнения счетчик переводится в множество зарезервированных.

Подгруппа специальных команд системного уровня и отладки включает прежде всего 4 команды, связанные с организацией работы треда в привилегированном режиме SYSTEM.

Переход в режим SYSTEM и выход из него выполняется соответственно командами System_Call и System_Ret. Выход в режим SYSTEM производится только для выполнения конкретно функции, точка входа в которую с дополнительной информацией находится в специальном системном векторе, на который указывает системный регистр GVS_call. Для загрузки и чтения этого регистра введены две привилегированные команды.

Команда System_Call может использоваться тредом как в режиме USER, так и SYSTEM. Механизмы обеспечения безопасности применения этой команды пока не прорабатывались.

Дополнительно в этой подгруппе имеются: команда KT задания контрольной точки, она применяется для трассировки параллельных программ; пустая команда NOP, применяется для выравнивания размещения команд в программе; команда ERR для вызова прерывания «Запрещенная команда», это применяется для организации «ловушек» в отлаживаемых программах.

2.3 Интерфейс со специализированными устройствами-ускорителями

Интерфейс взаимодействия тредовых устройств (TU) со специализированными функциональными устройствами (SFU) основан на средствах передачи сигналов, имеющихся в TU. Общая схема интерфейса показана на рис.3.

В *прямом направлении* для связи по управлению от TU к SFU введено 8 сигнальных каналов, общих для всех SFU ядра mt-LWP. Сигналы этих каналов отображаются в левом 8-разрядном поле Channel_Out (шкале выходных сигналов) некоторого 16-разрядного архитектурного регистра. Считается, что эти сигнальные каналы сопоставлены с *выходными каналами* интерфейса от TU к SFU по данным. Такие каналы реализуются буферами выдачи в ЛП (Buf_Out_j на рис.3), а привязка этих буферов выдачи данных к сигнальным каналам реализуется через таблицу ADR, входом в которую служит номер сигнального канала. Жесткого закрепления SFU за выходными каналами интерфейса нет — для привязки к выходному каналу в каждом SFU имеется регистр Chan_In, в котором хранится номер выходного канала, к которому считается подключенным данное SFU. Между выходными каналами и SFU должно соблюдаться взаимно-однозначное соответствие, т.е. к одному SFU не могут вести несколько выходных каналов интерфейса.

В *обратном направлении*, для связи по управлению от SFU к TU, также введено 8 сигналов, общих уже для всех SFU ядра mt-LWP. Эти сигналы отображены в 8-разрядном поле Channel_In (шкале входных сигналов) 16-разрядного архитектурного регистра, в котором находится и шкала Channel_OUT. Сигналы поля Channel_In однозначно связаны с *входными каналами* связи по данным от SFU к TU (Buf_In_j на рис.3) через таблицу ADR. Жесткого закрепления SFU за выходными каналами также нет. Для привязки к конкретному входному каналу в SFU имеется регистр Chan_Out, в котором хранится номер соответствующего входного канала. Между входными каналами и SFU должно быть взаимно-однозначное соответствие.

Номера выходного и входного канала интерфейса, связанные с одним и тем же SFU, могут быть разными.

Управление приемом в TU сообщения от SFU более сложное. Для входного канала можно программно назначить TU, которое должно принимать сообщение от SFU, а также задать номер сигнала, передачей которого в этот TU инициируется прием и обработка этого сообщения (см.таблицы N_{SIG} и N_{TP} на рис.3).

Все таблицы интерфейса настраиваемые, их состояние фиксируется при начальной загрузке mPX. В качестве примера рассмотрим протокол выдачи по каналу выдачи j данных из TU в SFU для последующей их обработки.

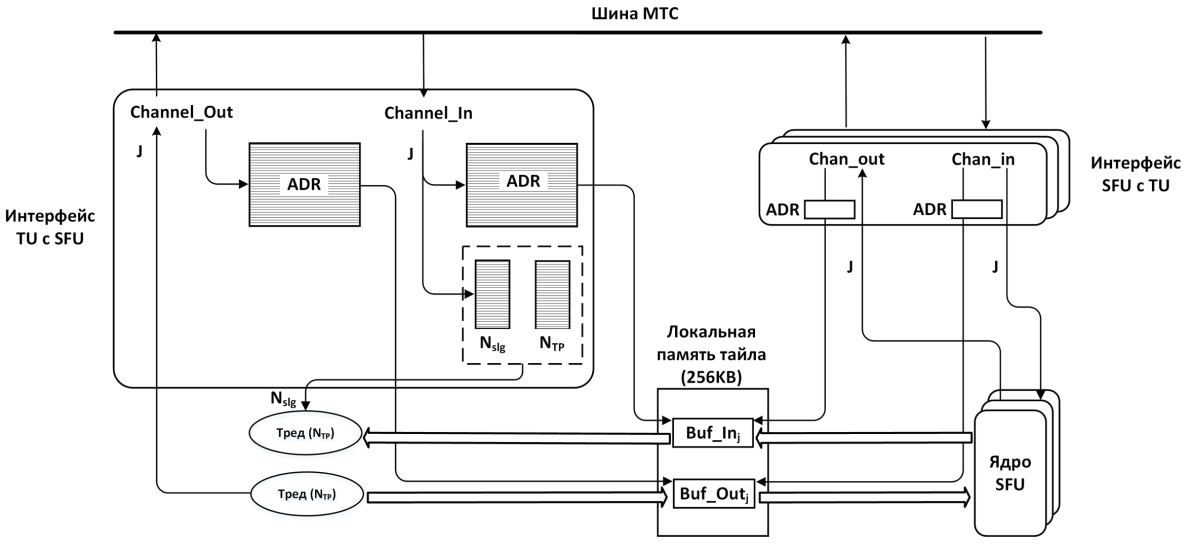

Рис.3 Интерфейс TU секций mt-LWP и SFU

DOI:10.21681/2311-3456-2020-03-22-39

А. Действия со стороны треда на TU

А.1. Проверка состояния бита j в шкале сигналов Channel_Out. Ожидание нулевого состояния этого бита, что означает готовность подключенного к каналу j SFU к приему.

А.2. Записать данные из TU в буфер выдачи Buf_Out_j этого канала выдачи j. Первым словом сообщения должно быть слово, содержащее в младших 16 разрядах длину сообщения в словах.

А.3. Выдать команду DMB в TU, чтобы все обращения к памяти из треда на этом TU были завершены.

А.4. Выдать бит запроса Chanel_Out [j], сигнализирующий о необходимости передачи по выходному каналу j в требуемое SFU. Для этого используется команда, которая этот бит устанавливает атомарно.

В. Действия со стороны SFU по приему данных от TU через канал j

В.1. В интерфейсе TU с SFU данным SFU производится ожидание единичного состояния бита с номером из регистра Chan_In (это канал j) в шкале сигналов каналов выдачи Channel_Out.

В.2. Если j-й бит шкалы Channel_Out единичный, то в буфере выдачи этого канала j уже находятся данные для этого SFU. Средствами SFU через канал прямого доступа к локальной памяти ЛП mt-LWP это сообщение считывается из буфера и затем начинается его обработка в SFU.

В.3. Если в SFU реализована возможность совмещения приема и обработки данных, то блок приема этого SFU после считывания производит сброс бита Chan_In в шкале Channel_Out, что означает готовность SFU к приему новых данных.

В.4. Если в SFU возможность совмещения приема и обработки не реализована, то сброс бита Chan_In в шкале сигналов Channel_Out будет только после обработки в SFU.

Действия по выдаче данных от SFU в TU похожи, только используется еще возможность выбора TU и сигнала в нем для обработки сообщения от SFU, это за счет настройки таблиц N_{SIG} и $N_{TP.}$

2.4. Адресация памяти и глобально адресуемая память

Имеются четыре команды работы с памятью через GAS. Две первых команды – это считывание или запись байта, полуслова или слова. Команда считывания имеет следующий вид на языке Ассемблера

r1 <- GAS [r2||r3], L

где L = {B, H, W}, указывает элемент памяти, к которому производится доступ, а 62-х разрядный адрес образуется объединением (конкатенацией) регистров r2 и r3. Старшие два бита r2 означают: 00 – доступ к байту; 01 – доступ к полуслову; 10 – доступ к слову. Эти биты устанавливаются в r2 заранее. Результат считывания записывается в регистр r1.

Команда вида

r1 -> GAS [r2||r3], L

производит запись в память из регистра r1, она похожа на команду считывания.

Следующие две команды формируют запросы на асинхронную пересылку из GAS в память ЛП (М) и обратно. Команда пересылки в ЛП(М) имеет вид

ЛП(М)[r1] <- GAS [r2||r3], L

где L = {32, ... 4096}. Это команда копирования блока слов фиксированной длины L. Длина блока задается в поле r1[31..29], а r1[28..0] – это адрес в ЛП(М), по которому копируется блок. Адрес в GAS формируется как r2||r3, но в данном случае старшие два бита r2 должны быть единичными. Кодировка длины блока в r1[31..29] : 000 – 32 байта; 001 – 64 байта, 010 – 128 байтов, 011 – 256 байт, 100 – 512 байт, 101 – 1024 байта, 110 – 2048 байт, 111 – 4096 байт.

Команда вида

ЛП(М)[r1] -> GAS [r2||r3], L

формирует заявку на обратную пересылку из ЛП(М) в память GAS.

Очевидно, команды работы с GAS выглядят экзотически, но такие решения применялись ранее в проектах перспективных суперкомпьютеров, например в [18] и не только (DARPA HPCS), а сейчас – в проектах суперкомпьютеров экзафлопсного уровня [24, 25] и даже в студенческих учебно-исследовательских проектах [15].

3. Базовая временная диаграмма

Выполнение одной арифметико-логической команды АЛУ занимает 7 однотактовых фаз, это базовая временная диаграмма.

Такт 1 (ВТ) – выбор TU для сеанса обслуживания, команда программы которого будет выполняться. На этом такте формируется 4-х битовый номер этого TU.

Такт 2 (РС) –по выбранному на такте ВТ номеру TU считывается его слово состояния программы, а из него выбирается текущее значение счетчика команд РС.

Такт 3 (ВК) – выборка команды по РС из буфера командных слов секции, если эта команда там имеется. Если команды в буфере нет, то данное TU теряет активность и заказывается подкачка команды. После подкачки команды TU продолжит работу с такта ВК. Если команда в буфере есть, то она передается на дешифрацию ее кода операции.

Такт 4 (ДШ) – дешифрация кода операции команды, а потом чтение микрокоманд для нее. Одновременно производится спекулятивное чтение регистра на буферный регистр TU, номер регистра выбирается из поля “r2” команды, хотя это может оказаться неправильным, поскольку до дешифрации ещё неизвестно, находится ли в поле “r2” номер регистра или нет. Если после дешифрации выяснится, что поле “r2” не содержит номер регистра-первого операнда команды, то считанное значение этого регистра игнорируется.

Такт 5 (IS) – выдача команды непосредственно на выполнение в АЛУ, но предварительно выполняется следующее:

- перепись первого операнда с буферного регистра на буферный регистр первого операнда, если в результате дешифрации подтверждена правильность его считывания;
- чтение второго операнда по номеру из поля “r3

или r1" команды на буферный регистр второго операнда;
- проверка по таблице резервирования регистров доступности значений регистров-операндов команды для выполнения с ними операции АЛУ;
- проверка возможных конфликтов, препятствующих выполнению данной команды.

Если обнаруживается недоступность регистров-операндов и/или конфликты, то производится сброс дальнейшего выполнения команды, что далее приводит либо к попытке повторного выполнения этой команды в следующем сеансе обслуживания данного TU, либо в случае ожидаемых больших времен неготовности операндов или существования конфликтов к дезактивации текущего TU до того момента, когда неготовность операндов или конфликт исчезнет.

Если недоступность операндов или конфликты не обнаружены, то происходит выдача команды на выполнение в АЛУ.

Такт 6 (ОП) – выполнение операции в АЛУ над операндами из буферных регистров первого и второго операнда. Установка счетчика команд PC данного TU на следующую команду, т.е. для этой следующей команды фактически выполнен такт PC. Одновременно проверяется, может ли данное TU выбрано максимально быстро для выполнения следующей команды этого треда, т.е. возможно ли фиктивное выполнение фазы ВТ для этого TU.

Такт 7 (ЗР) – запись результата выполнения операции в АЛУ в регистр результата с адресом из поля "r1" выполняемой команды. Если нет конфликта типа «запись-запись» текущей команды с какой-либо ранее запущенной, но еще не завершившейся командой данного треда, то запись в регистр-результат выполняется. Если конфликт обнаружен, то действия этого такта на следующем сеансе обслуживания TU повторятся, а следующая команда из этого TU тогда выполняться пока не будет.

На рис.4 демонстрируется выполнение команд тредов T_A, T_B , T_C , T_D и T_E на TU одной 16-тредовой секции. Видно, что в моменты времени T_1 и T_2 одновременно в одной секции выполняется много команд от разных TU. Это иллюстрирует основной прием совмещения выполнения команд. Чтобы усилить представление об уровне параллелизма выполнения команд в mt-LWP, надо разместить четыре рис.4 друг под другом. Получается 4-х канальное ядро с выполнением до 4-х команд за такт (это фазы ОП из каждой секции, ведь каждая из секций имеет свое АЛУ).

Используя рис.4, далее рассмотрим некоторые проблемные особенности. Видно, что конвейер в 7 тактов и выбранная простейшая дисциплина выполнения команд треда в порядке следования не позволяют вернуться к выполнению следующей команды после текущей ранее, чем через 4 такта. Это видно на примере треда T_A, см. строки 00 и 03.

Такой быстрый возврат в тред не всегда возможен. Например, тред T_B (строка 01) выдал команду с многотактовой фазой ОП, но возникли конфликты, которые позволили лишить этот тред активности на ближайшее время, т.е. сеанс обслуживания ему предоставляться

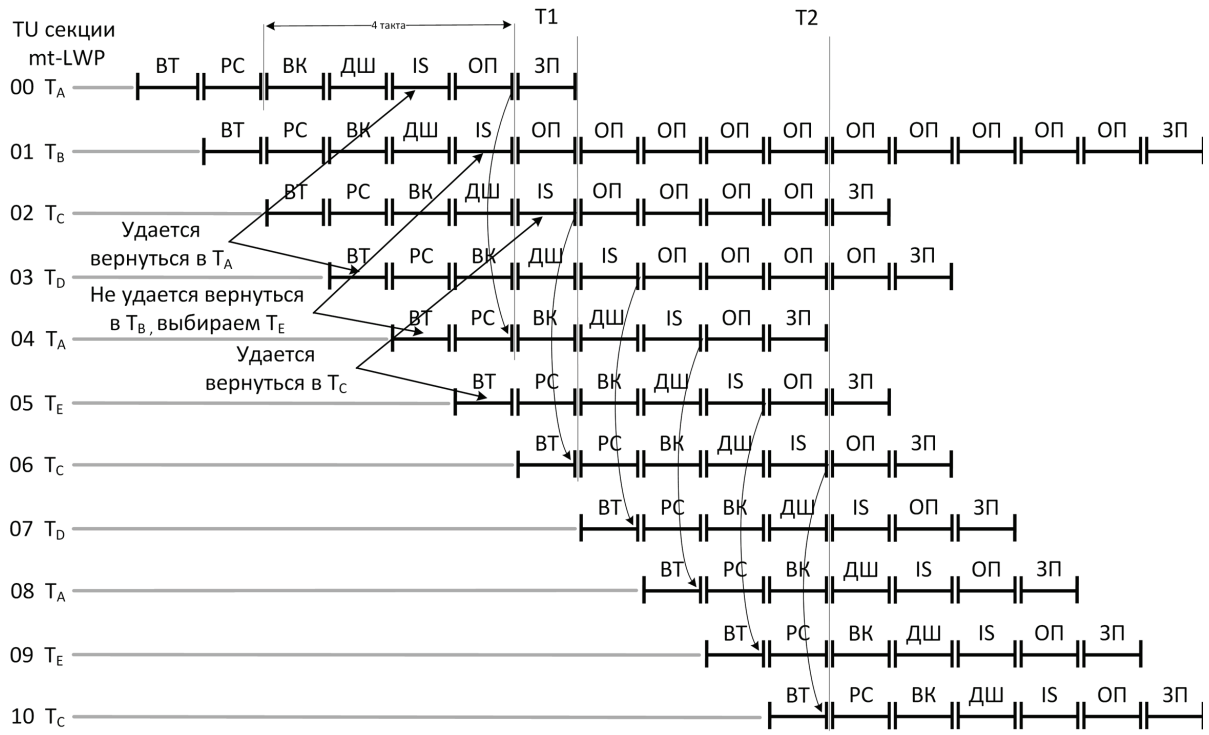

Рис.4 Пример совмещения выполнения команд разных активных тредов, выполняемых на TU в одной секции mt-LWP

DOI:10.21681/2311-3456-2020-03-22-39

не будет. Дело в том, что алгоритм предоставления сеанса обслуживания какому-либо треду из множества активных чувствителен к разным ситуациям в треде. Например, если для треда T_B после выдачи команды на многотактовую фазу ОП активность теряется, то в треде T_C после выдачи команды с 4-х тактовой фазой ОП (строка 02) эта активность не теряется, планируется выдача на следующем такте активности следующей команды, что и происходит (строка 06).

Для тредов T_A, T_C, T_D и T_E удается возвращаться в каждый из них максимально быстро, т.е. через 4 такта. Однако если количество таких активных тредов будет больше, то задержка выполнения очередной команды треда будет все больше и больше 4-х тактов.

Может оказаться, что активных тредов с однотактовыми фазами ОП будет не так много, тогда они будут быстро выполняться, а остальные треды будут типа T_B, которые лишают активности TU, на котором выполняются. Если таких потерявших активность TU будет слишком много, то это также проблемно. Например, при моделировании суперкомпьютера «Ангара» было замечено, что команды типа обращения к памяти через GAS могут занимать тысячи тактов. В данных случаях может оказаться полезным смена треда на таком TU. Пока такая диспетчеризация тредов на TU в проекте mPX не рассматривается, но имеется в виду и, конечно, будет исследоваться.

В общем, вопросов с оптимизацией временных диаграмм команд и алгоритма диспетчеризации тредов (фаза BT) предвидится еще много, для этого и запланировано создание наряду с разработкой микроэлектронных реализаций блоков потактовой имитационной модели.

4. Предварительные оценки и обсуждение

Процессор mPX, как и другой массово-параллельный процессор, должен иметь много ядер. Возникает вопрос, сколько ядер mt-LWP с разными ускорителями можно разместить на заданной площади кристалла при использовании той или иной технологии?

Составить некоторое представление об этом можно, сравнивая с похожими по архитектуре уже реализованными процессорами. Для mPX, как было установлено ранее в статье, это процессор Colossus. Полезные сведения по реализации микроархитектурных блоков еще имеются в материалах по реализованному с использованием технологии 16 нм 511-ядерному массово-параллельному процессору Celerity [13, 14, 15] (см. также сайт www.opencelerity.org). Этот микропроцессор реализован командой из 20 студентов из четырех американских университетов по учебно-исследовательскому проекту в рамках программы DARPA CRAFT (создание новой технологии быстрой разработки СБИС). Также оказалось особенно полезным использование опыта и результатов разработки по технологии 28 нм отечественного гибридного 21-ядерного микропроцессора NM6408MP [20, 21].

Далее проведем оценки на основе собственных исследований прошлых лет, учитывая и сведения по другим процессорам, о которых говорилось выше.

Более пяти лет назад авторами статьи была проведена эскизная проработка схемы 64-тредового процессора (далее – процессора K4, частота 1 GHz), похожего на ядро mt-LWP. Была проведена разработка его микроархитектурных блоков на уровне регистровых передач и сделана оценка этого процессора K4 по площади и энергетике с использованием таблиц затрат разных их компонентов (регистры, коммутаторы, памяти, мультиплексоры и демультиплексоры, АЛУ, вентильные логические схемы), которые были получены раздельно синтезом в САПР Cadence с использованием технологии 65 нм.

В результате такой оценки было получено, что одна 16-тредовая секция K4 размещается на площади 0.6763 мм^2 (26% от этого занимает самый большой блок K4, регистровый файл, что и ожидалось), а 64 KB статической памяти для K4 занимают 0.740 мм^2. Если воспользоваться этим для оценки тайла с ядром mt-LWP с четырьмя 16-тредовыми секциями и памятью в 256 KB, то получим (4x0.6763) + 4x0.740 ~ 5.7 мм^2. Если это пересчитать для технологии 28 нм (используем грубый коэффициент сжатия в 5.4 раза), то получим ~ 1 мм^2, а само ядро при этом будет занимать ~ 0.5 мм^2. Если продолжить пересчет уже для технологии 16 нм (грубый коэффициент сжатия 3.0), то для тайла получим оценку площади в ~ 0.35 мм^2.

Далее предположим, что имеется возможность реализации mPX на площади в 400 мм^2 по технологии 16 нм. Если учесть опыт разработки процессора Colossus, то 12% этой площади отнесем к расходам на линки и PCIe интерфейс, а 12.7% — на центральный коммутатор. Принимая это для mPX, получим, что на тайлы останется площадь кристалла в ~ 300 мм^2. Тогда, по грубой оценке, на такой площади можно разместить (300/0.35) ~ 860 тайлов, каждый из которых содержит 64-х тредовое ядро и 256 KB статической памяти. Поскольку в одном таком тайле выполняется до 4-х операций за такт, то производительность такого mPX на частоте 1 GHz будет (4x860) ~ 3.4 TOPS.

Это нижняя оценка производительности, поскольку по концепции создания mPX достижение высокой производительности должно получиться за счет введения в тайлах SFU. Рассмотрим далее разные варианты этого.

Первый вариант SFU – пусть в качестве SFU будет подключен реконфигурируемый блок FPU из процессора NM6408MP, способный выполнять до 8 операций над FP32 за такт, его площадь 0.025 мм^2 (технология 28 нм), если пересчитать с использованием грубого коэффициента сжатия, то на технологии 16 нм площадь будет 0.008 мм^2.

Добавление одного такого блока SFU увеличит площадь тайла до ~ 0.36 мм^2, тогда на площади 300 мм^2 можно будет разместить 830 тайлов, а пиковая производительность на частоте 1 GHz станет 830x8 ~ 6.6 TF. При двух подключенных таких SFU в тайле производительность будет 810x16 ~ 13 TF, при четырех SFU в тайле производительность станет 770x32 ~ 24.6 TF, а при восьми подключенных SFU производительность окажется 697x64 ~ 44.6 TF. Такой рост ограничен, поскольку при увеличении количества SFU в тайле возникает проблема обмена с ними данными даже для 64-тредового ядра mt-LWP. По-видимому, реально рассчитывать на 4 SFU в тайле.

Ранее в статье высказывалось, что при реализации Colossus на площади в 400 мм^2 может получиться производительность в 15 TF. Выше для похожего на Colossus процессора mPX, но в котором ядро обладает в 10 раз большей мультитредовостью (64 против 6), другими методами была получена близкая оценка в 13 TF.

Оценивавшийся выше тайл mPX с четырьмя SFU похож на NMPU-узел процессора NM6408MP [20, 21]. В этом узле вместо mt-LWP имеется однопотоковый RISC-процессор (площадь 0.041 мм^2), четыре векторных ускорителя (в каждом по FPU и набор из 8-ми векторных регистров) и специальный блок сжатия/растяжки векторов (общая площадь блоков векторной обработки 0.321 мм^2). RISC-процессор и блоки векторной обработки образуют ядро nmc4. В NMPU-узле ядро nmc4 подключено через специальный блок «Системный интегратор» (блок SI) к расслоенной на 8 памяти в 512 KB. В блок SI для поддержки быстрого считывания и записи векторов встроены 8 адресных генераторов, которые можно рассматривать как 8 специализированных тредовых устройств генерации адресов элементов векторов. Таким образом, тайл в mPX можно рассматривать как некоторое обобщение NMPU-узла в NM6408MP. Еще отметим, что 64-тредовое ядро mt-LWP оказалось больше по площади RISC-процессора ядра nmc4 более, чем в 10 раз (0.5 мм^2 против 0.041 мм^2), это плата за то, что оно выполняет не один поток команд, а 64, но над этим надо задуматься при реализации mt-LWP.

Второй вариант SFU – в качестве SFU можно взять реконфигурируемый и программируемый конвейерный ускоритель DRCP [19], разработанный для решения задач информационной безопасности и обладающий повышенной энергоэффективностью за счет применения микроархитектурных методов, аналогичных методам работы [26]. В таком ускорителе выполняется до 16 операций над INT32 за такт. Его площадь при реализации по технологии 28 нм оценивается в 0.15 мм^2 (0.05 мм^2 для 16 нм). Такое SFU отличается тем, что не требует высокого темпа обмена данными с ним, поскольку конвейерные процессы обработки очень длинные. Тайл mPX с одним таким SFU (технология 16 нм) будет иметь размер 0.40 мм^2, т.е. на кристалле в 400 мм^2 с учетом потерь на линки и PCIe-интерфейс можно будет разместить (300/0.40) ~ 750 тайлов, а пиковая производительность будет 12 TOPS. При подключении двух SFU будем иметь (667x32) ~ 21.3 TOPS, а для четырех SFU (545x64) ~ 35 TOPS, для 8 SFU – (400x128) ~ 51 TOPS. Реально иметь 8-12 SFU, требуемый темп обмена данными это позволяет. Новая архитектура DRCP+ позволяет на той же площади поднять производительность до 4-х раз, т.е. можно достигнуть ~ 200 TOPS.

Второй вариант SFU по производительности на целочисленных операциях также есть с чем сравнивать. Во-первых, в GPU Ampere, правда на операциях INT8/INT32 был достигнут уровень нескольких POPS (POPS – 10^{15} операций в секунду). Во-вторых, еще раньше о достижении этого уровня на операциях с INT8 в ускорителе TSP было объявлено новой американской фирмой Groq [27] (кристалл 725 мм^2, технология 14 нм, 0.9 GHz, по специфической конвейерной архитектуре можно этот процессор охарактеризовать как «супер DRCP++"). Итак, известные рекордные процессоры по TOPS имеют кристаллы 700-800 мм^2, используют технологии 7-14 нм. Применительно к mPX эти возможности дали бы масштабирование по производительности не менее, чем в 4 раза, т.е. в mPX также возможно было бы достижения POPS-го уровня.

Приведенные оценки могут показаться слишком приближенными и оптимистичными, но они есть, их можно критиковать, важно, что на них можно ориентироваться. Доля истины в них все-таки есть. Например, был проведен синтез схемы блока регистрового файла mt-LWP по его описанию на языке Verilog. Использовалась технология 28 нм. Этот блок по структуре мало отличается от одноименного блока процессора K4, который оценивался по таблицам для технологии 65 нм. Оказалось, что синтезированный блок занимает в 5.4 раза меньше площади (как предсказывалось) и в 6 раз экономнее по энергетике.

Возможны и другие SFU, поле исследований огромно, на это и был расчет. Важно, что мультитредовость mt-LWP, а она может из-за секционности как увеличиваться, так и уменьшаться, позволяет обеспечивать как эффективную работу с памятью, так и с множеством SFU разного типа, это ядро, в большей степени, лишь менеджер вычислительного процесса, происходящего в тайле.

Заключение

Данная статья – первая публикация по проекту mPX. Задача состояла в описании основных архитектурных решений, сравнения их с решениями других массово-параллельных процессоров, представить систему команд ядра mt-LWP и основы его функционирования.

Создание системы команд нового процессора всегда было интересным и достойным делом, но всегда сопровождалось сильнейшей критикой и спорами, которые в конечном итоге оказывались крайне полезными и улучшали решение. Собственно говоря, авторы статьи рассчитывали как раз инициировать такое обсуждение предложенной архитектуры ядра mt-LWP и процессора mPX как в среде участвующих в проекте коллег из разных организаций, так и других. Ситуация с высокоскоростной отечественной ЭКБ сейчас такова, что чем больше будет активных людей, вовлеченных в данную область, тем лучше. В связи с этим, если данная статья в результате согласия или несогласия с ней воодушевит на открытие новых проектов, то это было бы полезным для данной области.

Что касается ближайших задач в проекте mPX, то на первый план ставится доработка системы команд mt-LWP, формализация алгоритмов выполнения команд в виде временных диаграмм, разработка имитационной потактовой модели, базового системного программного обеспечения для проведения на этой модели исследований на фрагментах прикладных программ. Планируется выбрать два типа SFU, что рассмотрены в статье. Одновременно с этим будет продолжена реализация имеющихся схем блоков ядра mt-LWP и SFU с целью получения их оценок по площади и энергетике, что вместе с результатами, полученными на имитационной

модели, даст реальное представление о достижимых характеристиках такого процессора и возможном его практическом применении.

В данной статье неоднократно отмечалась близость архитектурных подходов построения процессоров mPX и Colossus. В таблице 1 были приведены сведения по процессору Colossus первого поколения выпуска 2018 года. Однако все быстро меняется, 15 июля 2020 года было объявлено о выпуске процессора Colossus второго поколения – GC200 [28]. Этот процессор уже изготовлен по технологии 7 нм, площадь кристалла 823 мм^2, количество транзисторов 59.4x10^9, еще больше, чем в GPU Ampere. Количество ядер в нем увеличено до 1472 (было 1216), а вот внутрикристальная память увеличена значительно, в 3 раза, до 900 MB. Пиковая производительность тензорных вычислений (FP16/FP32) увеличена в 2 раза, до 250 TF. Реальная же производительность при обучении нейронных сетей и их использовании за счет увеличенной внутренней памяти и тензорной производительности увеличилась в 8 раз. В работе [28] также уделено существенное внимание разработке удачного вычислительного узла в конструктиве 1U (высота 5 см) с четырьмя GC200, подключенных через разработанный этой же фирмой мост с DDR4 памятью объемом до 450 GB и пропускной способностью в 180 TB/s. Такой блок объединяется с другими в суперкомпьютере в соответствии с подходом “host-less”, линками процессора GC200. На уровне стоек применяется топология сети 3D тор. Это реальный конкурент вычислительным узлам на основе GPU A100, сравнение также имеется в работе [28].

Приведенный факт еще раз показывает, насколько стремительно идет развитие зарубежной высокоскоростной ЭКБ и создания на ее основе суперкомпьютеров для искусственного интеллекта, а также как быстро увеличивается их отрыв от отечественной ЭКБ. Что еще должно произойти для того, чтобы активизировались отечественные работы в области высокоскоростной ЭКБ?

Литература

1. Адамов А.А., Фомин Д.В., Эйсымонт Л.К. Главные проблемные направления в области отечественной элементной базы суперкомпьютеров // Вопросы кибербезопасности. 2019. № 4. С. 2-12. DOI: 10.681/2311-3456-2019-4-02-12
2. Адамов А.А., Павлухин П.В., Биконов Д.В., Эйсымонт А.Л., Эйсымонт Л.К. Альтернативные современным GPGPU перспективные универсальные и специализированные процессоры-ускорители // Вопросы кибербезопасности. 2019. №4. С.13–21. DOI: 10.681/2311-3456-2019-4-13-21
3. Durant L. [et al.] Inside Volta: The World’s Most Advanced Data Center GPU, 10 may 2017 // https://devblogs.nvidia.com/parallelforall/inside-volta/
4. Jia Z., Maggioni M., Staiger B., Scarpazza D.P. Dissecting the NVIDIA Volta GPU Architecture via Microbenchmarking //Technical report, High Performance Computing R&D Team Citadel, April 18, 2018 — 66 pp.
5. NVIDIA A100 Tensor Core GPU Architecture, May 2020, 83 pp.
6. Елизаров С.Г., Лукьянченко Г.А., Марков Д.С., Монахов А.М., Сизов А.Д., Роганов В.А. Программируемые на языках высокого уровня энергоэффективные специализированные СБИС для решения задач информационной безопасности // Системы высокой доступности. 2018. т.14. №3. С.40-48.
7. Knowless S. How to build processor for machine intelligence// RAAIS 30th June 2017// Видеозапись, 34 мин, 17 слайдов // https://www.youtube.com/watch?v=T8DvHnb3Y9g
8. Knowless S. Scaling Throughput Processors for machine Intelligence// ScaledML Conference, Stanford, 24 May 2018 // Видеозапись, 36 мин, 25 слайдов // https://www.youtube.com/watch?v=GgFUiN81QKo
9. Jia Z., Tillman B., Maggioni M., Scarpazza D.P. Dissecting the Graphcore IPU Architecture via Microbenchmarking. // Technical report, High Performance Computing R&D Team Citadel, 7 Dec 2019 — 91 pp.
10. Performance Benchmarks. https://www.graphcore.ai/benchmarks
11. Tommy Flowers // https://en.wikipedia.org/wiki/Tommy_Flowers
12. PEZY-SC2-PEZY// https://en.wikichip.org/wiki/pezy/pezy-scx/pezy-sc2
13. Ajavi T. [et al]. Experiences Using the RISC-V Ecosystems to Design an Accelerator-Centric SoC in TSMC 16 nm. // First Workshop on Computer Architecture Research with RISC-V, August 2017, 6 pp.
14. Taylor M. Celerity: An Open Source 511-core RISV-V Tiered Accelerator Fabric, 2017, 50 slides, http://www.opencelerity.org.
15. Schor D. A Look at Celerity’s Second-Gen 496-core RISC-V Mesh NoC // Wiki Chip Fuse, Jan 12, 2020, https://fuse.wikichip.org/news/3217/a-look-at-celeritys-second-gen-496-core-risc-v-mesh-noc/
16. Слуцкин А.И., Эйсымонт Л.К. Российский суперкомпьютер с глобально адресуемой памятью // Открытые системы. 2007. №9. С. 42-51.
17. Митрофанов В.В., Слуцкин А.И., Эйсымонт Л.К. Суперкомпьютерные технологии для стратегически важных задач. // Электроника: НТБ. 2008. №7. С. 66-79.
18. Семенов А.С., Соколов, А.А. Эйсымонт Л.К. Архитектура глобально адресуемой памяти мультитредово-потокового суперкомпьютера // Электроника: НТБ. 2009. №1. С. 50-56.
19. Эйсымонт Л.К., Бутов А.А., Пеплов И.С. и др. Реконфигурируемый вычислительный модуль // Патент на изобретение. RU 2 686 017. Дата регистрации 23.04.2019, 24 с.
20. Эйсымонт А.Л., Черников В.М., Черников Ан.В., Черников Ал.В., Косоруков Д.Е., Насонов И.И., Комлев А.А. Гетерогенная многопроцессорная система на кристалле с производительностью 512 Gflops // Системы высокой доступности. 2018. т.14. №3. С.49-56.

21. Биконов Д.В., Пузиков А.Д., Сивцов А.С., Эйсымонт Л.К. Трехуровневая система параллельного программирования гибридного 21-ядерного скалярно-векторного микропроцессора NM6408MP. // Вопросы кибербезопасности. 2019.№4. С.22-34. DOI: 10.681/2311-3456-2019-4-22-34
22. Кудрявцев М., Эйсымонт Л., Мошкин Д., Полунин М.. Суперкластеры — между прошлым и будущим // Открытые системы. 2008. №8 С. 40-47.
23. Речинский А., Горбунов В., Эйсымонт Л. Суперкластер с глобально адресуемой памятью // Открытые системы. 2011. №7. С. 21-25.
24. Горбунов В.С., Эйсымонт Л. К. Экзафлопсный барьер: проблемы и решения // Открытые системы. 2010. №6. С. 12-15.
25. Горбунов В.С., Елизаров Г.С., Эйсымонт Л.К. Экзафлопсные суперкомпьютеры: достижения и перспективы //Открытые системы. 2013. №7. С.10-14.
26. Dally W., Balfour J. et al. An Energy-Efficient Processor Architecture for Embedded Systems. IEEE Computer Architecture Letters, Vol.7, No 1, January-June 2008, pp.29-32.
27. Abts D. [et al] Think Fast: A Tensor Streaming Processor (TSP) for Accelerating Deep Learning Workloads // 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA), 14 pp
28. Toon N. Introducing 2nd generation IPU systems for AI at scale //15 july 2020//https://www.graphcore.ai/posts/introducing-second-generation-ipu-systems-for-ai-at-scale

Рецензент: *Сигов Александр Сергеевич, академик Российской академии наук, доктор физико-математических наук, профессор, Президент Российского технологического университета (РТУ МИРЭА), Москва, Россия.*
E-mail: sigov@mirea.ru

RESEARCH PROJECT OF MASS-PARALLEL PROCESSOR BASED ON MULTITREAD CORE WITH SPECIALIZED ACCELERATORS

Eisymont L.K.[10], Nikitin A.I.[11], Bikonov D.V.[12], Brazhkin A.A.[13], Peplov I.S.[14], Fedorenko P.P.[15], Ermakov S.S.16, Eisymont A.L.[17], Comlev A.A.[18]

Purpose. *Present a training and research project of the development a domestic mass-parallel processor mPX, samples of which could be used as processor-accelerators along with domestic universal processing frames when building computer nodes of supercomputers.*

Method. *Generation of mPX processor functionality and evaluation of its expected performance, analysis based on the results of information and analytical studies.*

Result. *The principles of operation of the processor and issues of its implementation have been developed at project. The basic component of the processor is a tile consisting of a 64-thread core (mt-LWP), 256 KB local memory, connected with special-purpose accelerators (SFU). The mPX processor must include several hundred of tiles connected by an intra-chip network, several links of inter-chip connection, and a PCI-e interface with the host-processor. The interface with off-chip memory is not yet considered. At this stage, attention is paid to the mt-LWP core ISA, which the article is devoted. The ideology of architecture, mPX is similar to the Colossus processor of the English company Graphcore, focused on machine learning tasks, but the mPX processor is a platform on the basis of which you can develop different variants. The objectives of the project are also information and analytical work, theoretical and practical edication of young specialists, and the development of technical solutions for creating a high-speed chips.*

Keywords: *mass-parallel processor, multithreading, data flow control, graph representation of programs, specialized accelerators.*

References

1. Adamov A.A., Fomin D.V., Eysymont L.K. Glavnye problemnye napravleniya v oblasti otechestvennoj elementnoj bazy superkomp'yuterov // Voprosy kiberbezopasno-sti, N4, 2019. — S. 2-12.
2. Adamov A.A., Pavluhin P.V., Bikonov D.V., Eysymont A.L., Eysymont L.K. Al'ternativnye sovremennym GPGPU perspektivnye universal'nye i specializiro-vannye processory-uskoriteli // Voprosy kiberbezopasnosti, N4, 2019 – S.13 – 21// http://cyberrus.com/wp-content/uploads/2019/07/13-21-432-19_2.-Bikonov.pdf
3. Durant L. [et al.] Inside Volta: The World's Most Advanced Data Center GPU, 10 may 2017 // https://devblogs.nvidia.com/parallelforall/inside-volta/
4. Jia Z., Maggioni M., Staiger B., Scarpazza D.P. Dissecting the NVIDIA Volta GPU Architecture via Microbenchmarking //Technical report, High Performance Computing R&D Team Citadel, April 18, 2018 — 66 pp.
5. NVIDIA A100 Tensor Core GPU Architecture, May 2020, 83 pp.
6. Elizarov S.G., Luk'yanchenko G.A., Markov D.S., Monahov A.M., Sizov A.D., Ro-ganov V.A. Programmiruemye na yazykah vysokogo urovnya energoeffektivnye speciali-zirovannye SBIS dlya resheniya zadach in-formacionnoj bezopasnosti // Sistemy vyso-koj dostupnosti, 2018, t.14, №3 – S.40-48.
7. Knowless S. How to build processor for machine intelligence// RAAIS 30th June 2017// Videozapis', 34 min, 17 slajdov // https://www.youtube.com/watch?v=T8DvHnb3Y9g
8. Knowless S. Scaling Throughput Processors for machine Intelligence// ScaledML Con-ference, Stanford, 24 May 2018 // Videozapis', 36 min, 25 slajdov // https://www.youtube.com/watch?v=GgFUiN81QKo

10 Leonid Eisymont, Ph.D. (in Math), Scientific consultant, Research Center "Module". Moscow, Russia. E-mail: verger-lk@yandex.ru
11 Anatoly Nikitin, independent expert. Moscow, Russia. E-mail: nikitin95006@outlook.com
12 Dmitriy Bikonov, Chief of sector, Research Center "Module". Moscow, Russia. E-mail: d.bikonov@module.ru
13 Alexey Brazhkin, software engineer, Research Center "Module", Moscow, Russia. E-mail: a.brazhkin@module.ru
14 Ilya Peplov, graduate student, National Research University "MIET". Zelenograd, Russia. E-mail: i.peplov@yandex.ru
15 Petr Fedorenko, student, National Research University "MIET". Zelenograd, Russia . E-mail: petyan.fedorenko@yandex.ru
16 Semyon Ermakov, student, National Research University "MIET". Zelenograd, Russia. E-mail: s_s_ermakov@mail.ru
17 Alexey Eisymont, Chief of sector, Research Center "Module". Moscow, Russia. E-mail: eisymont@module.ru
18 Arseny Comlev, Lead engineer, Research Center "Module". Moscow, Russia. E-mail: a.komlev@module.ru

9. Jia Z., Tillman B., Maggioni M., Scarpazza D.P. Dissecting the Graphcore IPU Architecture via Microbenchmarking. // Technical report, High Performance Computing R&D Team Citadel, 7 Dec 2019 — 91 pp.
10. Performance Benchmarks. https://www.graphcore.ai/benchmarks
11. Tommy Flowers // https://en.wikipedia.org/wiki/Tommy_Flowers
12. PEZY-SC2-PEZY// https://en.wikichip.org/wiki/pezy/pezy-scx/pezy-sc2
13. Ajavi T. [et al]. Experiences Using the RISC-V Ecosystems to Design an Accelerator-Centric SoC in TSMC 16 nm. // First Workshop on Computer Architecture Research with RISC-V, August 2017, 6 pp.
14. Taylor M. Celerity: An Open Source 511-core RISV-V Tiered Accelerator Fabric, 2017, 50 slides, http://www.opencelerity.org.
15. Schor D. A Look at Celerity's Second-Gen 496-core RISC-V Mesh NoC // Wiki Chip Fuse, Jan 12, 2020, https://fuse.wikichip.org/news/3217/a-look-at-celeritys-second-gen-496-core-risc-v-mesh-noc/
16. Sluckin A.I., Eysymont L.K. Rossijskij superkomp'yuter s global'no adresuemoj pamyat'yu // Otkrytye sistemy, №9 – 2007 — C. 42-51.
17. [17] Mitrofanov V.V., Sluckin A.I., Eysymont L.K. Superkomp'yuternye tekhnologii dlya strategicheski vazhnyh zadach. // Elektronika: NTB, №7,2008. — C. 66-79.
18. Semenov A.S., Sokolov, A.A. Eysymont L.K. Arhitektura global'no adresuemoj pamyati mul'titredovo-potokovogo superkomp'yutera // Elektronika: NTB, №1, 2009. — C. 50-56.
19. Eysymont L.K., Butov A.A., Peplov I.S. i dr. Rekonfiguriruemyj vychislitel'nyj modul' // Patent na izobretenie, RU 2 686 017, Data registracii 23.04.2019, 24 str.
20. Eysymont A.L., CHernikov V.M., CHernikov An.V., CHernikov Al.V., Kosorukov D.E., Nasonov I.I., Komlev A.A. Geterogennaya mnogoprocessornaya sistema na kristalle s proizvoditel'nost'yu 512 Gflops // Sistemy vysokoj dostupnosti, 2018, t.14, №3, s.49-56.
21. Bikonov D.V., Puzikov A.D., Sivcov A.S., Eysymont L.K. Trekhurovnevaya sistema parallel'nogo programmirovaniya gibridnogo 21-yadernogo skalyarno-vektornogo mikroprocessora NM6408MP. // Voprosy kiberbezopasnosti, nomer 4, 2019, str.22-34, https://cyberrus.com/wp-content/uploads/2019/07/22-34-432-19_3.-Puzikov.pdf
22. Kudryavcev M., Eysymont L., Moshkin D., Polunin M.. Superklastery — mezhdu proshlym i budushchim // Otkrytye sistemy, №8 – 2008
23. Rechinskij A., Gorbunov V., Eysymont L. Superklaster s global'no adresuemoj pamyat'yu // Otkrytye sistemy, №7, 2011. – S. 21-25.
24. Gorbunov V.S., Eysymont L. K. Ekzaflopsnyj bar'er: problemy i resheniya // Otkrytye sistemy, №6, 2010 – S. 12-15.
25. Gorbunov V.S., Elizarov G.S., Eysymont L.K. Ekzaflopsnye superkomp'yutery: dostizheniya i perspektivy //Otkrytye sistemy, N7, 2013 — S.10-14.
26. Dally W., Balfour J. et al. An Energy-Efficient Processor Architecture for Embedded Systems. IEEE Computer Architecture Letters, Vol.7, No 1, January-June 2008, pp.29-32.
27. Abts D. [et al] Think Fast: A Tensor Streaming Processor (TSP) for Accelerating Deep Learning Workloads // 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA), 14 pp
28. Toon N. Introducing 2nd generation IPU systems for AI at scale //15 july 2020// https://www.graphcore.ai/posts/introducing-second-generation-ipu-systems-for-ai-at-scale