УДК 004.27 DOI: 10.31114/2078-7707-2020-1-64-69

Влияние особенностей модели вычислений и архитектуры на надежность параллельной потоковой вычислительной системы

Д.Н. Змеев, Н.Н. Левченко, А.С. Окунев, А.Л. Стемпковский

Институт проблем проектирования в микроэлектронике РАН, г. Москва, nick@ippm.ru

***Аннотация* — Необходимость распределения вычислений на сотни тысяч и даже миллионы вычислительных элементов заставляет разработчиков искать новые подходы к разработке архитектуры суперкомпьютеров, в частности, использовать потоковую модель вычислений. Решение вопросов надежности вычислений на таких системах часто требует нестандартных подходов. В потоковых вычислительных системах из-за сложной структуры параллельных вычислительных процессов возникают трудности с локализацией источника ошибки, которая может распространиться по всей системе до момента ее обнаружения. Также возникают проблемы при восстановлении нормального функционирования вычислительного ядра после сбоя либо при реконфигурировании системы после отказа. В статье описываются особенности параллельной потоковой вычислительной системы, оказывающие влияние на её надежность: использование аппаратной ассоциативной памяти в качестве ассоциативной памяти ключей, применение парадигмы «раздачи», выполнение программы вычислительного узла, не прерываемое на подкачку данных и др. Эти особенности потоковой модели вычислений и архитектуры ППВС следует учитывать при создании подходов к повышению надежности потоковой системы.**

***Ключевые слова* — потоковая модель вычислений, параллельная потоковая вычислительная система, надежность вычислений.**

I. Введение

За рубежом и в нашей стране специалисты формулируют требования к архитектурным особенностям высокопроизводительных вычислительных систем экзафлопсной производительности. Многие из них склоняются к тому, что для создания таких систем необходим «революционный» подход, использующий новые парадигмы вычислений, новые параллельные языки программирования и нетрадиционные архитектурные решения.

Кризис в развитии элементной базы и методов распределения вычислений на все большее число процессоров вынуждает разработчиков вновь обратиться к потоковым вычислительным системам. До последнего времени большинство проектов по созданию вычислительных систем с управлением потоком данных ограничивалось разработкой специализированных устройств, ускоряющих выполнение отдельных частей программы, которые с трудом распараллеливаются традиционными средствами программирования на классических вычислительных системах. Созданием же универсального суперкомпьютера с управлением потоком данных занимаются лишь немногие группы исследователей.

В одной из статей авторами утверждалось, что «сегодня крайне важен (для экзафлопсных суперкомпьютеров) поиск новых методов, алгоритмов и способов организации управления имеющимся вычислительным ресурсом. Кроме того, на повестке дня остаются следующие проблемы: разработка новых алгоритмов решения задач, учитывающих новые архитектуры; создание новой модели параллельных вычислений; снижение энергопотребления; повышение отказоустойчивости» [1].

По мнению же известного западного ученого профессора Университета Луизианы Томаса Стерлинга, системы экзафлопсной производительности должны обладать следующими свойствами (с точки зрения архитектуры), важными для обеспечения отказоустойчивости [2]:

- легкими механизмами синхронизации для быстрой передачи управления вместо принципа глобальных барьеров;
- динамическими средствами управления вычислительными ресурсами, а не статическим распределением, выполненным при компиляции;
- использованием микроархитектуры, построенной на идеях обработки потока данных (dataflow), а не на классических процессорных ядрах;
- созданием локальных микроконтрольных точек для восстановления вычислений прямо в памяти ядра в отличие от традиционного рестарта с дисков для восстановления при сбоях.

Все перечисленные свойства в той или иной степени реализуются в параллельной потоковой вычислительной системе (ППВС) [3], работа над которой ведется в ИППМ РАН.

В потоковых вычислительных системах из-за сложной структуры параллельных вычислительных процессов, основанных на принципе «активации по готовности данных», возникают трудности с локализацией источника ошибки, а также опасность распространения ошибки по всей системе до момента ее обнаружения и изоляции отказавшего модуля. Также возникают проблемы реконфигурирования системы после отказов и восстановления ее нормального функционирования.

Вообще, обеспечению надежности суперкомпьютерных систем в настоящее время уделяется большое внимание как с точки зрения отказоустойчивых алгоритмов, так и со стороны аппаратуры. Особенно это относится к вычислительным системам экзафлопсного уровня производительности, имеющим в своем составе миллионы и более вычислительных ядер.

II. Потоковая модель вычислений с динамически формируемым контекстом

В основе потоковой модели вычислений с динамически формируемым контекстом лежит активация программных узлов по готовности данных. В отличие от классических моделей вычислений программный узел в такой модели вычислений выполняется с начала и до конца при отсутствии приостановок вычислительного процесса и ожидания каких-либо дополнительных внешних данных, то есть программный узел оперирует только входными данными и контекстом (рис. 1). В потоковой модели вычислений по коммуникационной сети «циркулируют» исключительно токены. Токены представляют собой структуры данных, которые содержат непосредственно данное (операнд), набор служебных полей и ключ (контекст или индекс), который однозначно определяет местоположение этого операнда в виртуальном адресном пространстве задачи. Данные в виде токенов поступают на вход рабочего пространства токенов (рис. 1), в котором происходит сопоставление всех имеющихся токенов между собой по определенным правилам. При положительном результате сопоставления происходит формирование активной комбинации, а токены, участвующие в ее формировании, удаляются (в общем случае) из рабочего пространства. Активная комбинация исполняется на вычислителе. В результате обработки активной комбинации формируются новые токены, которые либо выдаются в качестве результата на выход, либо подаются в рабочее пространство токенов. Подобный цикл вычислений продолжается до полного выполнения программы.

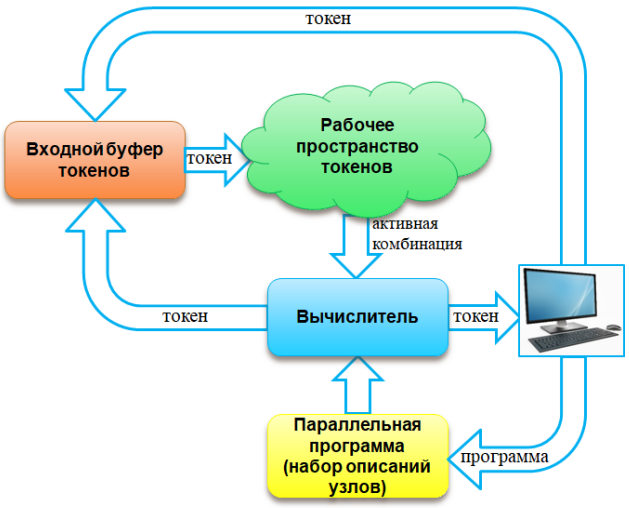

Рис. 1. Потоковая модель вычислений с динамически формируемым контекстом

Потоковая модель вычислений с динамически формируемым контекстом воплощена в параллельном языке программирования DFL [4]. Программа для параллельной потоковой вычислительной системы (на DFL) представляет собой набор описаний программных узлов. Каждый программный узел состоит из заголовка (входные данные и контекст) и тела (набора инструкций, обрабатывающих входные данные). Активация программного узла происходит в том случае, если все его входные данные присутствуют в рабочем пространстве токенов.

После активации программного узла происходит выполнение последовательности команд, в результате чего вычисляются новые данные (токены). Результаты выполнения программного узла в виде токенов отсылаются либо в другие программные узлы, либо выдаются на ХОСТ-машину. В операторе посылки токена, помимо операнда, указывается имя программного узла и вход, на который будет направлен токен, а также контекст токена, вычисляемый в коде программного узла. В потоковой модели вычислений с динамически формируемым контекстом адреса «получателей» данных определяются непосредственно в программе узла в динамике, этим она отличается от классических потоковых моделей вычислений, которые разрабатывались ранее [5-7].

В этой модели вычислений в качестве рабочего пространства токенов (абстрактного устройства сопоставления) концептуально применима ассоциативная память. Ассоциативная память отвечает за сопоставление токенов (операндов) по определенным правилам и формирование пакета, представляющего собой структуру, содержащую операнды токенов, общий ключ, контекст и ряд служебных полей. Процесс сравнения (сопоставления) одного токена одновременно с множеством других токенов является имманентным свойством ассоциативной памяти, что и послужило поводом для выбора ассоциативной памяти в качестве основного сопоставляющего устройства.

III. Архитектура параллельной потоковой вычислительной системы

Параллельная потоковая вычислительная система реализует потоковую модель вычислений с динамически формируемым контекстом. Базовая архитектура (многомодульная вычислительная среда) параллельной потоковой вычислительной системы представляет собой набор объединенных глобальным коммутатором токенов вычислительных модулей [8]. К этому коммутатору подключен процессор ввода/вывода, через который осуществляется связь с «внешним миром» (рис. 2).

Между ядрами в системе передаются единицы информации в виде токенов. Коммутация между ядрами осуществляется по значению номера ядра, вырабатываемого блоком хэширования на основе настраиваемой функции распределения вычислений.

Вычислительные ядра в рамках одного кристалла организуются в вычислительные модули.

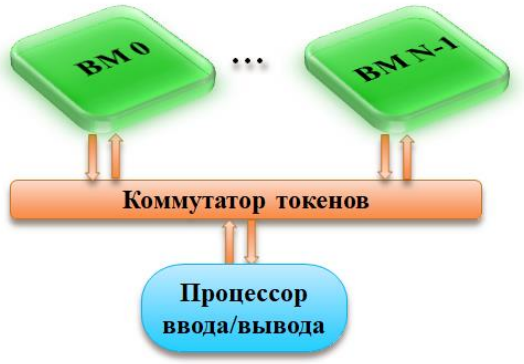

Рис. 2. Базовая архитектура ППВС «Буран»

В состав вычислительного модуля входят следующие основные узлы и блоки (рис.3):

- внутримодульный коммутатор токенов;
- процессор сопоставлений (ПС);
- внутримодульный коммутатор пакетов;
- исполнительное устройство (ИУ).

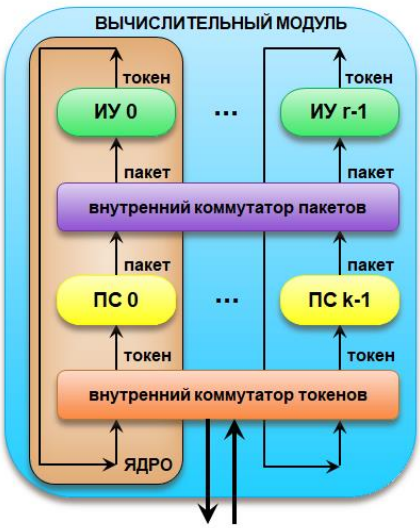

Рис. 3. Структура базового вычислительного модуля ППВС «Буран»

На вход процессора сопоставления каждого вычислительного ядра от исполнительного устройства или из коммуникационной среды поступают токены, после чего запускается операция поиска по ключу токена в ассоциативной памяти ключей (АПК).

Поиск предназначен для выявления готовности операндов к обработке. В процессе выполнения поиска происходит сравнение ключа пришедшего токена с ключами всех уже находящихся в процессоре сопоставлений токенов. Каждый токен имеет поле маски, которая предназначена для маскирования отдельных полей ключа. Замаскированные поля ключа при его сравнении с другими ключами всегда считаются совпадающими. Сравнение ключей сопоставляемых токенов производится с учетом обеих масок: при сравнении ключей используется результирующая маска, получаемая наложением друг на друга масок пришедшего и хранящегося токенов.

Если результат поиска отрицательный (среди находящихся в ПС отсутствуют токены, совпадающие по ключу с пришедшим токеном), то этот токен сохраняется в процессоре сопоставлений. Запись токена производится в любую свободную ячейку, адрес которой выдает схема формирования свободных адресов.

При положительном результате поиска, т.е. если в процессе сопоставлений найден токен, «парный» пришедшему токену, на выходе АПК формируется адрес ячейки, в которой был записан совпавший токен. При множественном результате совпадений на выходе АПК последовательно формируются адреса ячеек, в которых было зафиксировано совпадение. Факт наличия нескольких совпадений называется множественным откликом.

После формирования пакет поступает на обработку на одно из «свободных» исполнительных устройств. Результатом обработки этого пакета являются новые токены, которые поступают либо в процессор сопоставлений (своего или «чужого» ядра), либо передаются из системы на ХОСТ-машину.

IV. Особенности ППВС и надежность

Потоковая модель вычислений с динамически формируемым контекстом и её аппаратная реализация обладают целым рядом особенностей, которые необходимо учитывать при оценке надежности вычислительной системы.

Наиболее кардинальным отличием от классических вычислительных систем является использование аппаратной ассоциативной памяти в качестве ассоциативной памяти ключей в процессоре сопоставлений вычислительного ядра. Использование аппаратной ассоциативной памяти объясняется тем, что она наиболее естественным образом обеспечивает активацию программных узлов, а также автоматическую синхронизацию вычислительных процессов по данным. Ассоциативная память отличается от прямоадресуемой памяти, которая используется в традиционных вычислительных системах, адресацией по содержимому, возможностью записи на свободное место, а также одновременным просмотром всех ячеек в режиме «Поиск».

Поскольку методы работы ассоциативной памяти отличаются от прямоадресуемой, стандартные способы решения задачи восстановления работы после сбоя или отказа при проектировании средств обеспечения надежности работы вычислительного ядра могут быть использованы лишь частично. Тем не менее, развитая система команд процессора сопоставлений, а также сама его структура, состоящая из различных по назначению узлов и блоков, позволяют повысить надежность работы вычислительной системы за счет следующих аппаратно-программных решений: это уменьшение размера непосредственно ассоциативной памяти ключей, выделение отдельной памяти токенов, система команд

с расширенным набором признаков аварийного останова (АВОСТ), аппаратный контроль запрещенных состояний, построение иерархии памяти и др.

Особенностью потоковой модели вычислений с динамически формируемым контекстом также является применение парадигмы «раздачи», которая в архитектуре ППВС обеспечивается путем использования принципа однократного присваивания. Принцип парадигмы «раздачи» заключается в том, что «производитель» (программа узла) каждого нового значения (токена) определяет в процессе вычисления (в динамике) всех его «потребителей» и самостоятельно обеспечивает рассылку по требуемым адресам (вычисляется требуемый контекст токена). Другими словами, это динамический выбор целевых узлов в узлах-источниках данных. «Получатель», в отличие от традиционной парадигмы «сбора», не знает об источнике пришедших ему данных. Данное свойство позволяет, во-первых, сократить число обращений к данным лежащим в памяти, во-вторых, обеспечить отсутствие проблемы когерентности кэш-памяти, свойственной для традиционных моделей вычислений, и связанных с ее решением оборудования, и в-третьих, отказаться от глобальной синхронизации (а также глобальных контрольных точек) и перейти к обеспечению надежности вычислительной системы с использованием локальных контрольных точек.

Можно отметить другую особенность модели вычислений: одним из ее базовых принципов является то, что выполнение программы вычислительного узла не прерывается на подкачку каких-либо дополнительных данных, а зависит исключительно от входных данных, элементов контекста и констант. Учитывая же, что размер программного узла, а, следовательно, и время его выполнения «ограничены», появляется возможность с малыми затратами при обнаружении сбоев или отказов в процессе выполнения перезапускать обработку пакета либо на том же самом ИУ, либо на свободном соседнем. Обеспечению реализации данного свойства способствует принцип обезличенности исполнительных устройств, который заключается в том, что готовый к активации пакет может быть выполнен на любом свободном ИУ, что повышает надежность вычислительной системы. Для реализации этого предназначена аппаратура, обеспечивающая «перелив» пакетов или токенов между «свободными» исполнительными устройствами или внутримодульным коммутатором токенов. Кроме того, это повышает технологичность изготовления кристаллов.

Еще одним важным свойством системы программирования и модели вычислений (а не архитектуры ППВС) является то, что содержательное описание алгоритма отделено от управления порядком вычислений (распределение по времени), распределением и локальностью (по пространству). Для этого задается функция распределения (локализации в пространстве и времени), у которой аргументом является виртуальный адрес (вектор индексов). Это позволяет, во-первых, на уже отлаженной программе «экспериментировать» с другими функциями распределения без внесения изменений в алгоритм параллельной программы, во-вторых, производить отладку алгоритма программы на одном вычислительном ядре, в-третьих, запускать отлаженную на одном вычислительном ядре программу запускать на любом количестве ядер без изменений.

Кроме того, в ППВС обеспечивается поддержка мелкозернистого параллелизма на уровне аппаратуры: типовые операции по обработке токенов и пакетов также поддержаны аппаратно. Эта особенность архитектуры позволяет определять порции данных/вычислений исходя из свойств задачи, а не из требований исполняющей системы. Мелкозернистый параллелизм позволяет существенно упростить процедуру перезапуска обработки программного узла в случае возникновения сбоя или отказа исполнительного устройства, а также обеспечить аппаратный контроль по выдаче токенов.

Особенностью модели вычислений и архитектуры ППВС также является возможность начала вычислений до прихода полного пакета входных данных. Это означает, что работа вычислительной системы может начинаться по мере поступления входных данных. Это касается как начала работы системы на конкретной задаче, так и перехода к новой итерации в процессе вычислений в рамках одной задачи, что позволяет отказаться от глобальной синхронизации и сосредоточиться на локальном уровне. Данная особенность неразрывно связана с применением парадигмы «раздачи» и принципа однократного присваивания.

Можно отметить, что основную проблему в поиске неисправности в процессе работы параллельной программы на ППВС создает недетерминированность самого вычислительного процесса. И это еще одна особенность системы - недетерминированный процесс вычислений, связанный с асинхронностью выполнения программы. Прохождение вычислительного процесса в ППВС не детерминировано, хотя результат его выполнения безусловно детерминирован. Такой недетерминизм на аппаратном уровне позволяет извлекать «неявный» параллелизм, заложенный в самом алгоритме задачи, который программист изначально может и «не замечать». Сам недетерминизм образуется в результате организации вычислений по готовности данных, что вместе с многоядерностью и возможными неравномерностями при передаче данных по коммуникационной сети обеспечивает его проявление.

С точки зрения повышения надежности программного обеспечения для таких систем нужно отметить, что семантический разрыв между параллельным языком высокого уровня ППВС и самой архитектурой ППВС значительно ниже, чем в современных системах. В традиционных системах, где этот разрыв большой, растет стоимость разработки ПО и снижается его надежность. Особенности проектируемой системы позволят, в противоположность этому, обеспечить рост надежности программного обеспечения, повысить продуктивность программирования и эффективность использования аппаратуры, сократить время решения задачи, а также снизить стоимость создания программного обеспечения.

Перечисленные выше особенности потоковой модели вычислений и архитектуры ППВС должны учитываться при создании подходов к повышению надежности потоковой системы. Эти подходы предполагают разработку оригинальных алгоритмов сбора информации для формирования локальных контрольных точек, способов фиксации сбоя и отказа, а также реализацию новых аппаратных решений при создании локальных средств восстановления для вычислительного ядра системы.

V. Заключение

При создании высокопроизводительных вычислительных систем одной из важнейших проблем является проблема надежности вычислений. Это касается и разрабатываемой в ИППМ РАН параллельной потоковой вычислительной системы. Архитектура ППВС реализует потоковую модель вычислений с динамически формируемым контекстом, поэтому применяемые в классических вычислительных системах методы обеспечения надежности вычислений не всегда подходят для этой архитектуры. С другой стороны, особенности как модели вычислений, так и реализующей ее архитектуры безусловно влияют на разрабатываемые новые методы обеспечения надежности системы.

Описанные в статье особенности потоковой модели вычислений и архитектуры, которые рассмотрены с точки зрения их влияния на надежность вычислений, требуют разработки оригинальных алгоритмов сбора информации для формирования локальных контрольных точек, способов фиксации сбоев и отказов, а также включения новой аппаратуры в состав вычислительного ядра и модуля для автоматического восстановления работы после сбоя. Помимо приведенных в статье аппаратно-программных решений, для повышения надежности функционирования вычислительной системы могут использоваться и другие методы [9]. Реализованные в ППВС вышеперечисленные средства позволят обеспечить высокую степень надежности, необходимую для подобных вычислительных систем.

Литература

[1] Горбунов В.В., Елизаров Г.А., Эйсымонт Л.К. Экзафлопсные суперкомпьютеры: достижения и перспективы // Открытые системы. 2013. № 7. С. 10-15.

[2] Многоточие Стерлинга // Суперкомпьютер. 2010. №3. С. 17-20.

[3] Ivannikov A.D., Levchenko N.N., Okunev A.S., Stempkovsky A.L., Zmejev D.N. Dataflow Computing Model – Perspectives, Advantages and Implementation // Proceedings of IEEE East-West Design & Test Symposium (EWDTS'2017), Novi Sad, Serbia, Sept 29 - Oct 2. 2017. P. 187-190.

[4] Климов А.В., Окунев А.С. Графический потоковый метаязык для асинхронного распределенного программирования // Проблемы разработки перспективных микро- и наноэлектронных систем - 2016. Сборник трудов / под общ. ред. академика РАН А.Л. Стемпковского. М.: ИППМ РАН, 2016. Часть II. С. 151-158.

[5] Jack B. Dennis and David P. Misunas. A preliminary architecture for a basic data-flow processor // in Proceedings of the 2nd annual symposium on Computer architecture (ISCA '75), ACM, New York, NY, USA. 1974. P.126-132. doi: 10.1145/642089.642111

[6] Böhm A.P.W. Dataflow and hybrid dataflow architecture summary // in Parallel computer systems, Rebecca Koskela and Margaret Simmons (Eds.), ACM, New York, NY, USA. 1990. P. 281-286. doi: 10.1145/100215.100286

[7] Silc J., Robic B., Ungerer T. Asynchrony in parallel computing: From dataflow to multithreading // Parallel and Distributed Computing Practices. 1998. Vol. 1. No. 1. P. 3-30.

[8] Стемпковский А.Л., Левченко Н.Н., Окунев А.С., Цветков В.В. Параллельная потоковая вычислительная система – дальнейшее развитие архитектуры и структурной организации вычислительной системы с автоматическим распределением ресурсов // ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ. 2008. №10. С. 2-7.

[9] Гаврилов С.В., Иванова Г.А., Рыжова Д.И., Стемпковский А.Л. Методы повышения надежности комбинационных микроэлектронных схем на основе мультиинтервального анализа быстродействия // Системы высокой доступности. 2015. Т. 11. № 4. С. 69-76.

Impact of Features of the Computing Model and Architecture on the Parallel Dataflow Computing System Reliability

D.N. Zmejev, N.N. Levchenko, A.S. Okunev, A.L. Stempkovsky

Institute for Design Problems in Microelectronics of RAS, Moscow, ippm@ippm.ru

***Abstract* — When creating high-performance computing systems, especially with non-traditional architecture, the problem of increasing reliability is associated with the presence of a large amount of parallel computing. With the increase in the computational cores number this problem is getting worth. While distributing computations to hundreds of thousands and millions of computational cores, the dataflow computing model provides significant advantages over the traditional von Neumann approach in terms of increasing real performance and increasing the efficiency of parallelizing tasks with a large volume of hard-structured data. The features of the non-traditional computing model and the architecture implementing this model have a significant impact on the new methods being developed to ensure system reliability.**

The architecture of the parallel dataflow computing system currently being developed in the IPPM RAS, is based on the dataflow computing model with a dynamically formed context with activation of program nodes by data readiness. In such systems due to the complex structure of parallel computing processes it is difficult to localize the source of the error. It is also necessary to solve the problem of reliable restoration of normal functioning after a fault and reconfiguring of the system after a failure.

The article describes the features of the dataflow computing model and architecture, which are considered from the point of view of their impact on the reliability of computing. The reliability of the computing system is enhanced by the developed hardware and software solutions for the matching processor: reducing the size of the content addressable memory of keys, using a separate memory of tokens, creating a command system with an extended set of AVOST flags, hardware control of forbidden states, building a memory hierarchy, etc.

These and other features of the parallel dataflow computing system require the original algorithm development for collecting information for the formation of local checkpoints, methods for faults and failures registration as well as adding new hardware to the computational core for the automatic operation restoration after a fault. The above-mentioned tools implemented in the PDCS will make it possible to provide a high reliability level required for such computing systems.

***Keywords* — dataflow computing model, parallel dataflow computing system, computation reliability**

REFERENCES

[1] Gorbunov V.V., Elizarov G.A., Jejsymont L.K. Jekzaflopsnye superkomp'jutery: dostizhenija i perspektivy (Exaflops supercomputers: achievements and prospects) // Otkrytye sistemy. 2013. № 7. S. 10-15.

[2] Mnogotochie Sterlinga (Etcetera of Sterling) // Superkomp'juter. 2010. №3. S. 17-20.

[3] Ivannikov A.D., Levchenko N.N., Okunev A.S., Stempkovsky A.L., Zmejev D.N. Dataflow Computing Model – Perspectives, Advantages and Implementation // Proceedings of IEEE East-West Design & Test Symposium (EWDTS'2017), Novi Sad, Serbia, Sept 29 - Oct 2. 2017. P. 187-190.

[4] Klimov A.V., Okunev A.S. Graficheskij potokovyj metajazyk dlja asinhronnogo raspredelennogo programmirovanija (A graphical dataflow meta-language for asynchronous distributed programming) // Problemy razrabotki perspektivnyh mikro- i nanojelektronnyh sistem - 2016. Sbornik trudov / pod obshh. red. akademika RAN A.L. Stempkovskogo. M.: IPPM RAN, 2016. Chast' II. S. 151-158.

[5] Jack B. Dennis and David P. Misunas. A preliminary architecture for a basic data-flow processor // in Proceedings of the 2nd annual symposium on Computer architecture (ISCA '75), ACM, New York, NY, USA. 1974. P.126-132. doi: 10.1145/642089.642111

[6] Böhm A.P.W. Dataflow and hybrid dataflow architecture summary // in Parallel computer systems, Rebecca Koskela and Margaret Simmons (Eds.), ACM, New York, NY, USA. 1990. P. 281-286. doi: 10.1145/100215.100286

[7] Silc J., Robic B., Ungerer T. Asynchrony in parallel computing: From dataflow to multithreading // Parallel and Distributed Computing Practices. 1998. Vol. 1. No. 1. P. 3-30.

[8] Stempkovskij A.L., Levchenko N.N., Okunev A.S., Cvetkov V.V. Parallel'naja potokovaja vychislitel'naja sistema – dal'nejshee razvitie arhitektury i strukturnoj organizacii vychislitel'noj sistemy s avtomaticheskim raspredeleniem resursov (Parallel Dataflow Computing System - the Further Development of Architecture and the Structural Organization of the Computing System with Automatic Distribution of Resources) // INFORMACIONNYE TEHNOLOGII. 2008. №10. S. 2-7.

[9] Gavrilov S.V., Ivanova G.A., Ryzhova D.I., Stempkovskij A.L. Metody povyshenija nadezhnosti kombinacionnyh mikrojelektronnyh shem na osnove mul'tiinterval'nogo analiza bystrodejstvija (Methods for improving the reliability of combinational microelectronic curcuits based on multiinterval timing analysis) // Sistemy vysokoj dostupnosti. 2015. T. 11. № 4. S. 69-76.