УДК 681.3

С.Д. Погорелый
Киевский национальный университет имени Тараса Шевченко
sdp@univ.net.ua

К вопросу формализации проектирования приложений для кластерных систем с GPU

Предложен метод повышения производительности кластерных систем, основанный на использовании видеоадаптеров компании NVIDIA с архитектурой FERMI в составе вычислительных узлов. Обоснована необходимость формализованного проектирования алгоритмов для таких систем, основанная на использовании математического аппарата модифицированных систем алгоритмических алгебр. Предложена организационная структура инструментальной системы алгоритмического проектирования с возможностью последующего синтеза программ, ассоциированных с параллельными алгоритмами. Описана предлагаемая методика проектирования и рассмотрен пример создания приложения. Приведены результаты ускорения при использовании предложенной методики и инструментальных средств.

SMP-архитектура, MPP-архитектура, CUDA-архитектура, инструментарий программирования для CUDA-архитектуры, модифицированные системы алгоритмических алгебр, организационная структура инструментальной системы проектирования параллельных алгоритмов, сеть Петри

Введение

Выводы экспертов в области компьютерных технологий совпадают в том, что ресурс экстенсивного роста производительности компьютерных систем вследствие повышения сложности и тактовой частоты микропроцессоров себя исчерпал. В то же время существует большое разнообразие архитектур, которые могут обеспечить существенное повышение быстродействия алгоритмов, и важным классом среди них есть параллельные:

- SMP-архитектура (Рис. 1, отражает один уровень параллелизма);
- MPP-архитектура (Рис. 2, отражает два уровня параллелизма);
- CUDA-архитектура (при использовании в MPP-системах отражает три уровня параллелизма).

Эти аргументы делают актуальной задачу создания для них параллельных алгоритмов.

Рисунок 1 - SMP-архитектура

Рисунок 2 - MPP-архитектура

Последние исследования и разработки компании NVIDIA привели к созданию архитектуры FERMI [1], которая является наиболее продвинутой в истории вычислений на GPU и предлагает ряд новых возможностей:

- 512 ядер обеспечивают супервычисления и высокую производительность с минимальными затратами;
- позволяет широко использовать вычисления на GPU и CPU, поддерживая возможность работы с C ++ и совместимость со средой разработки Visual Studio, что упрощает параллельное программирование для этой архитектуры;
- имеет развитую иерархию системы памяти: регистровую, локальную, глобальную, коллективную, константную и текстурную.

Существуют различные технологии и соответствующие им инструментальные средства для создания программных средств для этих архитектур.

1. Инструментарий программирования CUDA SDK позволяет реализовывать на специальном упрощенном диалекте языка программирования ANSI C99 алгоритмы, которые будут выполняться на графических процессорах NVIDIA, и включать специальные функции в текст программы на C. Инструментарий дает разработчику возможность по своему усмотрению организовывать доступ к набору инструкций графического ускорителя, управлять его памятью, организовывать на нем сложные параллельные вычисления. Первая версия CUDA SDK представлена 15.02.2007 г. Пакет использует grid-модель памяти, кластерное моделирование потоков и SIMD инструкции. Создан для платформ Linux, Mac OS X, Windows.

2. Инструментарий программирования OpenCL предназначен для написания компьютерных программ, связанных с параллельными вычислениями на различных графических (GPU) и центральных (CPU) процессорах. Включает язык программирования, который базируется на стандарте ANSI C99, интерфейс программирования приложений (англ. API), библиотеки и динамическая система для поддержки разработки программного обеспечения. OpenCL обеспечивает параллелизм на уровне инструкций и на уровне данных и является реализацией технологии GPGPU. OpenCL является полностью открытым стандартом.

3. Инструментарий программирования DirectX 11. В конце 2008 года на ежегодной конференции Gamefest компания Microsoft анонсировала новую версию популярного в мире графического API - DirectX 11. Эта технология позволяет разработчикам использовать преимущества новейших разработок в области аппаратного обеспечения (это касается как CPU, так и GPU), а также упростить их работу. Официальная финальная версия вышла 22 октября 2009 года в составе Windows 7.

Динамика роста количества ядер в GPU *существенно выше*, чем в CPU. В этой связи возникла новая технология – GPGPU. GPGPU (GPU общего назначения) – техника использования графического процессора видеокарты для выполнения расчётов в приложениях для общих вычислений, которые обычно проводит CPU. Это обосновывает актуальность использования видеокарт с GPU и технологии GPGPU в системах MPP-архитектуры.

Проектирование параллельных алгоритмов

Инструментальные системы программирования, существующие в настоящее время, обеспечивают средства проектирования параллельных алгоритмов и синтеза соответствующих программ, в основном, одной парадигмы параллельного программирования. Это значительно затрудняет сравнение результатов применения различных подходов к разработке параллельных приложений. А, учитывая то, что таких подходов существует достаточно много, процесс анализа и выбора наилучшего становится практически невыполнимым.

Алгоритмический этап проектирования параллельных приложений является начальным и *наименее технологически обеспеченным*. Однако, он влияет на весь дальнейший процесс разработки и определяет ее успех в целом. Инструментарий этого этапа проектирования должен обеспечивать:

- эффективные средства описания алгоритмов функционирования систем параллельной обработки данных с учетом их специфики;
- возможность формальных эквивалентных преобразований алгоритмов с целью получения новых более продуктивных версий;
- средства описания последовательных и параллельных фрагментов алгоритмов, возможность генерации ассоциированных с исследуемыми алгоритмами программ для различных подходов параллельного программирования;
- удобные визуальные средства проектирования алгоритмов;
- возможность оценки темпоральных характеристик работы алгоритмов перед их использованием на целевых параллельных аппаратно-программных платформах.

Перечисленные требования выдвигают задачу формализованного проектирования параллельных алгоритмов, для чего предлагается использовать математический аппарат модифицированных систем алгоритмических алгебр [2 – 6]. Этот выбор целесообразен в силу следующих причин:

- математический аппарат САА-М ориентирован на проектирование параллельных алгоритмов и ассоциированных с ними программ;
- любой алгоритм, записанный в САА-М, представляет собой алгебраическую формулу, позволяющую применять систему развитых в САА-М формальных эквивалентных преобразований для оптимизации и дальнейшего автоматизированного синтеза программного обеспечения;
- САА-М соответствуют концепции структурного программирования;

- методология проектирования должна базироваться не только и не столько на коммерческих продуктах, сколько на научном фундаменте.

Опишем методику формирования параллельной схемы для SMP-архитектуры на примере алгоритма Данцига (широко применяется для решения задачи маршрутизации в компьютерных сетях, основывается на матричных вычислениях и имеет полиномиальную сложность $O(n^3)$. Идея заключается в последовательном формировании с помощью рекуррентной процедуры подматриц кратчайших путей растущей размерности. Каждая такая матрица фактически является матрицей кратчайших путей подграфов с вершинами от *0* до *m-1*.

1. Вначале следует сформировать последовательную схему алгоритма. С учетом приведенных в статье обозначений ее можно записать в виде соотношения:

$$Dancig = \{\underset{m>N}{} SubDanc1(m)\times SubDanc2(m)\times SubDanc3(m)\times(++m)\}$$

2. Необходимо определить концепции распараллеливания, причем, в связи стем, что выполняется распараллеливание алгоритма по данным, размещенным в памяти (каждый поток обрабатывает определенную часть информационной множества), следует решить вопрос их разбиения.

3. Необходимо сформировать параллельную схему алгоритма, основываясь на необходимости обеспечения максимального параллелизма в работе потоков.

4. Основываясь на системе эквивалентных преобразований в САА-М и формируя различные критерии трансформаций, получить совокупность параллельных схем, обеспечивающих различные параметры соответствующих вычислительных процедур.

5. Исследовать полученные схемы (анализ, моделирование и т.д.) с целью выбора максимально соответствующей нужным требованиям.

Выполняя указанные действия, получим следующие две схемы алгоритма (методика с указанием всех обозначений описана в [7]):

$$Dancig = \{\underset{m>N}{}\left(\dot{\bigvee_{l=1}^{2N}}\left(PSub1(m,2N,l)\times B_{2N}(l)\right)\right)\times\left(\dot{\bigvee_{l=1}^{2N}}\left(PSub2(m,2N,l)\times B_{2N}(l)\right)\right)\times$$
$$\times\left(\dot{\bigvee_{l=1}^{2N}}\left(PSub3(m,2N,l)\times B_{2N}(l)\right)\right)\times(++m)\} \quad (1)$$

$$Dancig = \{\underset{m>N}{}\dot{\bigvee_{l=1}^{2N}}\left(PSub1(m,2N,l)\times B_{2N}(l)\times PSub2(m,2N,l)\times B_{2N}(l)\times PSub3(m,2N,l)\times B_{2N}(l)\right)\times$$
$$\times(++m)\} = \dot{\bigvee_{l=1}^{2N}}\{\underset{m>N}{}PSub1(m,2N,l)\times B_{2N}(l)\times PSub2(m,2N,l)\times B_{2N}(l)\times PSub3(m,2N,l)\times$$
$$\times B_{2N}(l)\times(++m)\} \quad (2)$$

Выполнив эквивалентные преобразования схемы (1) в САА-М, получим схему (3):

$$Dancig = \dot{\bigvee_{l=1}^{2N}}\{\underset{m>N}{}\left(\left(\underset{\omega}{}\left(PSub1(m,N,l)\vee PSub2(m,N,l)\right)\times B_{2N}(l)\right)\right)\times$$
$$\times\left(\left(PSub3(m,2N,l)\times B_{2N}(l)\right)\right)\times(++m)\} \quad (3)$$

Схемы (1) и (2), фактически, отражают две модели синхронизации потоков (рис. 3). Преимуществом схемы (3) является меньшее количество синхронизаций, что повышает эффективность её работы.

Благодаря параллельности выполнения первых двух операций можно существенно уменьшить общее количество операций синхронизации, а поскольку это операции, которые нельзя распараллелить, получим выигрыш во времени за счет некоторого увеличения нагрузки на отдельные потоки.

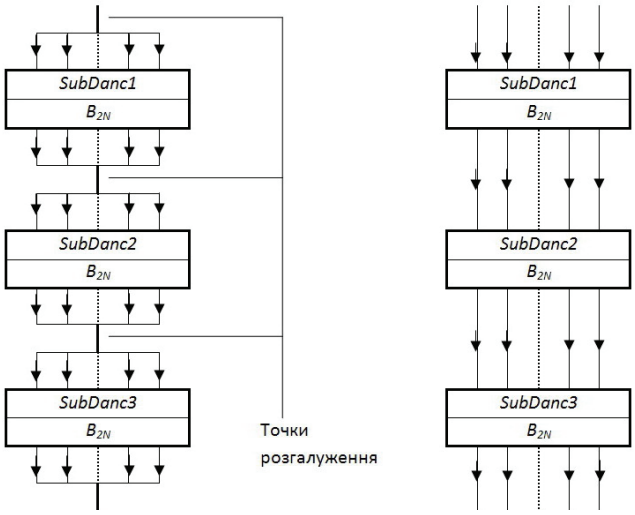

Рисунок 3 - Модели синхронизации потоков

Организационная структура инструментальной системы проектирования параллельных алгоритмов

На рис. 4 представлена организационная структура инструментального комплекса для проектирования параллельных алгоритмов и ассоциированных с ними программ.

Модуль синтеза САА-М-схем на основе UML-диаграмм позволяет осуществить два уровня параметризации входных данных:

- на уровне САА-М-схем алгоритмов;
- на уровне UML-диаграмм.

Модуль трансформации САА-М-схем алгоритмов на основе развитой в алгоритмических алгебрах системы эквивалентных трансформаций позволяет получить совокупность схем параллельного алгоритма с различным поведением и характеристиками.

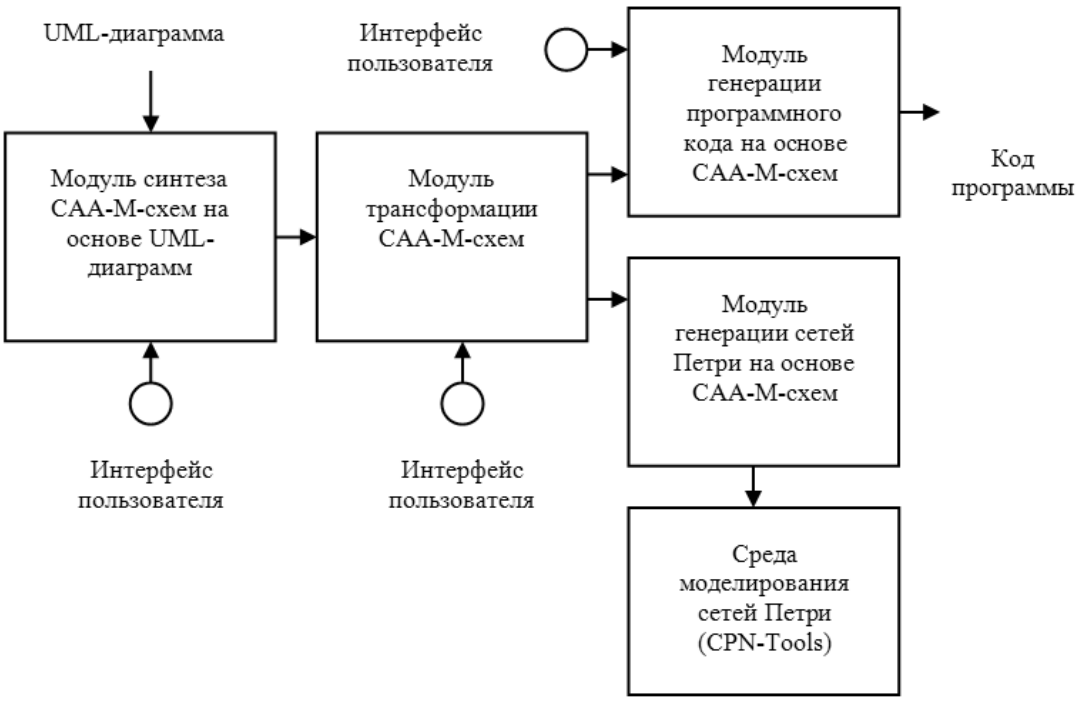

Рисунок 4 - Организационная структура инструментального комплекса

Для любой из полученных САА-М-схем алгоритмов соответствующий модуль даёт возможность сгенерировать цветную сеть Петри, моделирование которой позволяет получить темпоральные оценки параллельных алгоритмов. На рис. 5 представлена цветная сеть Петри для параллельного алгоритма Данцига, соответствующая схеме (3). В конечном счете, для выбранной схемы (схем) осуществляется синтез параллельной программы (программ) для определенной парадигмы параллельного программирования.

Параллельная схема алгоритма Флойда-Уоршала для MPP-архитектуры с GPU

Методику создания приложения для MPP-архитектур с использованием видеоадаптеров с архитектурой FERMI рассмотрим на примере алгоритма Флойда-Уоршала, который решает задачу определения кратчайших путей на взвешенном графе, основывается на матричных вычислениях и имеет полиномиальную сложность $O(n^3)$. На видеоадаптерах целесообразно реализовать ту его часть, которая касается регулярных матричных вычислений: обработку матриц весов кратчайших путей D и предшествования P [8]. После применения предложенного подхода получим фрагменты схем алгоритма, которые выполняются на GPU:

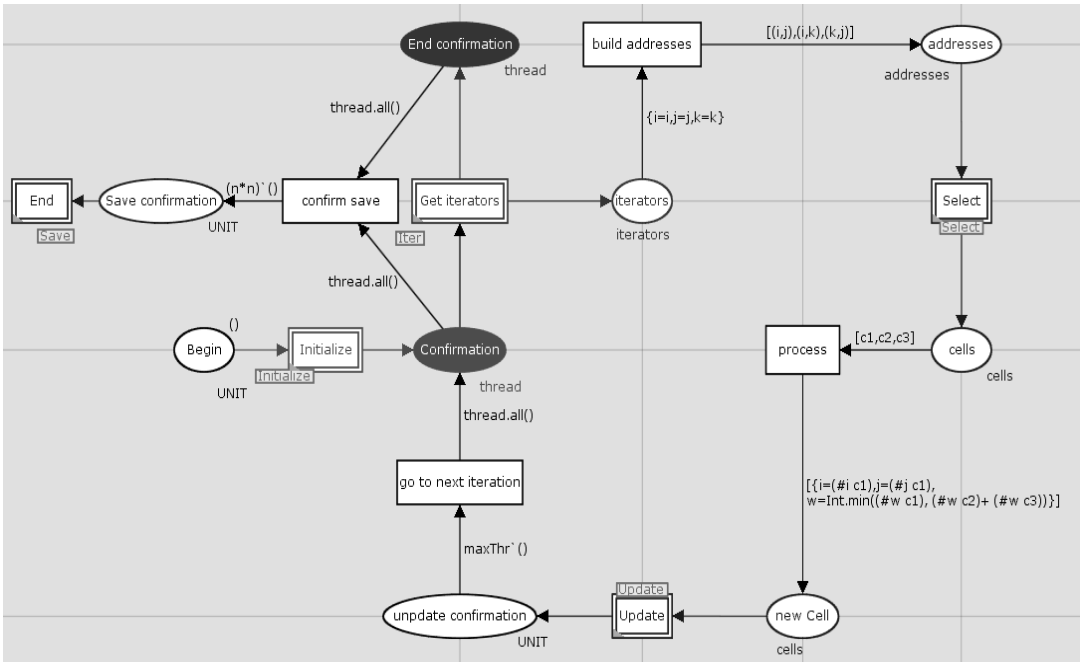

Рисунок 5 - Цветная сеть Петри для алгоритма Данцига

$$GPUBlock1 := S(\beta) * \left[\dot{\bigvee_{s=1}^{nThread}} (Cell\,\mathrm{Re}aliz) \right] * T(\alpha)) \quad (4)$$

$$GPUBlock2 := S(\beta) * \left[\dot{\bigvee_{s'=1}^{nThread}} (Column\,\mathrm{Re}aliz) \right] * T(\alpha)) \quad (5)$$

Подставляя (4) и (5) в схему алгоритма Флойда-Уоршала, получаем общую схему, реализующую MPI-парадигму кластерных вычислений с использованием видеоадаптеров [X]. По полученной схеме алгоритма было синтезировано приложение, которое выполнялось на кластере, содержащем 4 узла с видеоадаптерами GEForce430GT (архитектура FERMI). Общее ускорение быстродействия для различных реализаций алгоритма Флойда-Уоршала:

- последовательная реализация на одном компьютере,
- (MPI-парадигма с видеоадаптерами, 4 узла)/(MPI-парадигма, 4 узла),

- (MPI-парадигма с видеоадаптерами, 4 узла)/(последовательная реализация на одном компьютере)

представлено на рис. 6 (по вертикальной оси отложено ускорение (в разах), по горизонтальной – количество *n* вершин графа). Приведенные зависимости позволяют сделать оценку, что при использовании старших моделей видеоадаптеров ряда с архитектурой FERMI компании NVIDIA, может быть достигнуто ускорение быстродействия в 80 – 100 раз.

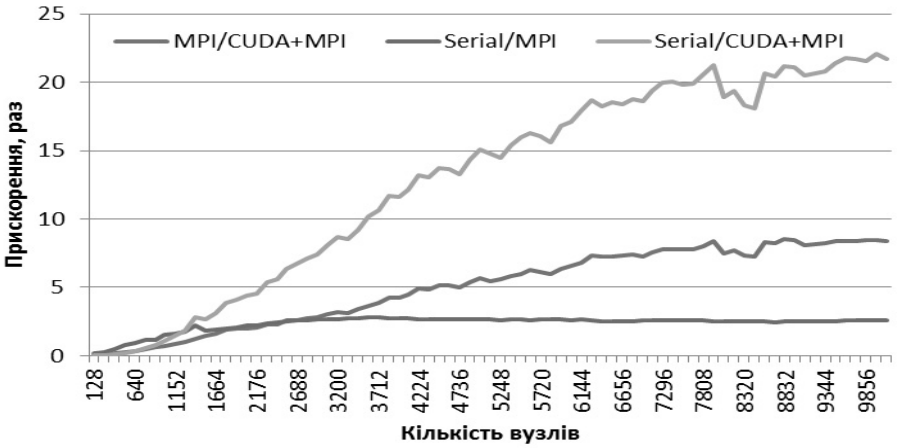

Рисунок 6 - Ускорение вычислений для различных реализаций алгоритма

Выводы

1. Разработаны концепция и организационная структура системы проектирования параллельных алгоритмов (технология GPGPU), включающая следующие возможности:

- нотации исходных (проектируемых) алгоритмов в виде:
  - САА-М-схем,
  - UML-диаграмм;
- эквивалентных преобразований схем алгоритмов в САА-М;
- моделирования схем параллельных алгоритмов, представленных в виде сетей Петри в среде CPN Tools;
- генерации параллельных приложений для различных парадигм параллельного программирования и используемых инструментальных средств.

2. Математический аппарат САА-М В. М. Глушкова является гибким, удобным и эффективным средством *аналитического исследования и анализа* параллельных схем алгоритмов, что позволяет сосредоточиться на методологии проектирования системы, концепциях и методах трансформации параллельных схем, абстрагируясь от конкретных деталей реализации. В результате полученные схемы являются базой для создания соответствующего программного обеспечения.

Список литературы

1. Анализ методов повышения производительности компьютеров с использованием графических процессоров и программно-аппаратной платформы CUDA / С.Д. Погорелый, Ю.В. Бойко, М.И. Трибрат, Д.Б. Грязнов // Математичні машини і системи. – 2010. – №1. – С. 40 - 54.

2. Pogorilyy S.D. Finding Strongly Connected Components In Parallel / S.D. Pogorilyy, S.I. Lozytskyi // Applied and Computational Mathematics. – 2007. – V 6 – No 2. – P.121-130.

3. Formalized Methods of paralleling the Goldberg-Tarjan Algorithm / S.D. Pogorillyy, Yu.V. Boyko, S.I. Lozitskiy, A.D. Gusarov // Journal of Automation and Information Sciences. – 2008. – V 40. – No 9. – P. 64-75.

4. Pogorilyy S.D. Paralleling Of Edmonds-Karp Net Flow Algorithm / Pogorilyy S.D. Y, Gusarov A.D. // Appl. Comput. Math. – 2006. – V. 5. – No.2. – P.121-130.

5. An approach to the parallel solution of a high-dimensional basic flow problem / S. D. Pogorilyy, Yu. V. Boyko, A. D. Gusarov, S. I. Lozytskyi // Cybernetics and Systems Analysis, Volume 45, Issue 2 (March 2009) pages: 291 – 296. Springer Science and Business Media Inc. ISSN:1060-0396.

6. Pogorilyy S.D. A Conception for Creating a System of Parametric Design of Parallel Algorithms and their Software Implementations / S.D. Pogorilyy and I. Yu. Shkulipa // Cybernetics and System Analysis, Volume 45, Issue 6 (November 2009) pages: 952 - 958. Springer Science and Business Media Inc. ISSN:1060-0396.

7. Дослідження паралельних схем алгоритму Данцига для обчислювальних систем зі спільною пам'яттю / С.Д. Погорілий, В.А. Мар'яновський, Ю.В. Бойко, О.А. Верещинський // Математичні машини і системи. – 2009. – №4. – С. 27-37.

8. С.Д. Погорілий Вітель. Формування узагальнених паралельних схем алгоритму Флойда-Уоршала / С.Д. Погорілий, В.А. Мар'яновський, Ю.В. Бойко // Системні дослідження та інформаційні технології. – 2010. – №1. – С. 52 – 68.

Надійшла до редколегії 10.10.2011

С.Д. ПОГОРІЛИЙ
Київський національний університет імені Тараса Шевченка

До питання формалізації проектування додатків для кластерних систем з GPU

Запропоновано метод підвищення продуктивності кластерних систем, заснований на використанні відеоадаптерів компанії NVIDIA з архітектурою FERMI в складі обчислювальних вузлів. Обгрунтовано необхідність формалізованого проектування алгоритмів для таких систем. Запропонована організаційна структура інструментальної системи алгоритмічного проектування з можливістю подальшого синтезу програм, асоційованих з паралельними алгоритмами. Описана пропонована методика проектування і розглянуто приклад створення програми. Наведено результати прискорення при використанні запропонованої методики та інструментальних засобів.

SMP-архітектура, MPP-архітектура, CUDA-архітектура, інструментарій програмування для CUDA-архітектури, модифіковані системи алгоритмічних алгебр, організаційна структура інструментальної системи проектування паралельних алгоритмів, мережа Петрі

S. D. POGORILY
Kyiv National Taras Shevchenko University

On the Problem of Formalization of Designing Applications for Cluster Systems with GPU

A method of improving performance of cluster systems based on the use of video cards from NVIDIA's FERMI architecture consisting of computing nodes is proposed. Necessity of formal design algorithms for such systems is shown. Organizational structure of algorithmic design tool with the possibility of further synthesis of programs associated with parallel algorithms is proposed. The proposed design procedure is described and an example of the program is considered. The results of acceleration when using the proposed methodology and tools are provided.

SMP-architecture, MPP-architecture, CUDA-architecture, programming tools for CUDA-architecture, modified systems of algorithmic algebras, the structure of the instrumental design of parallel algorithms, Petri nets