OSTIS-2014
(Open Semantic Technologies for Intelligent Systems)

УДК 004.272:43+004.272.32

НЕКОТОРЫЕ ЗАКОНОМЕРНОСТИ И ОБЪЕКТИВНЫЕ ОГРАНИЧЕНИЯ РЕАЛИЗАЦИИ АЛГОРИТМОВ СЕМАНТИЧЕСКОЙ ОБРАБОТКИ НА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ С МАССОВЫМ ПАРАЛЛЕЛИЗМОМ

Вереник Н.Л.[*], Сейткулов Е.Н.[**], Гирель А.И.[*], Татур М.М.[*]

[*] *Белорусский государственный университет информатики и радиоэлектроники, г. Минск, Республика Беларусь*

nick.verenik@gmail.com

tatur@i-proc.com

[**] *Евразийский национальный университет имени Л.Н. Гумилева г. Астана, Казахстан*

seitkulov_y@enu.kz

В настоящей работе приводится описание разрабатываемой архитектуры проблемно-ориентированного семантического процессора, рассматривается формат данных и система команд процессора, основные принципы функционирования. Приводится ряд закономерностей и объективных ограничений архитектуры. Вкратце рассмотрен пример использования процессора для решения задачи поиска пути в графе.
Ключевые слова: семантическая обработка информации; семантическая сеть; проблемно-ориентированный процессор; массовый параллелизм.

Введение

Одним из наиболее перспективных способов формализации функционирования произвольной интеллектуальной системы является использование семантических сетей, с помощью которых могут быть представлены как знания системы (база знаний системы), так и алгоритмы интеллектуальной обработки информации, используемые данной системой. Семантическая сеть по своей сути является некоторой графовой структурой, элементы которой наделяются дополнительным смыслом. Вычислительная система, на базе которой функционирует семантическая сеть, как правило, является, так называемой, графодинамической машиной [Голенков и др., 2012] – программной либо аппаратной системой, внутреннее состояние которой представлено графом. Процесс обработки информации в такой системе трактуется как графодинамический процесс, т.е. процесс изменения внутренней графовой структуры системы, который может включать в себя не только изменение внутреннего состояния элементов графа, но и изменение конфигурации графа (добавление либо удаление вершин и дуг в графе).

На практике при создании прикладной интеллектуальной системы, как правило, используется программная модель семантической сети, реализованная на универсальной вычислительной системе (персональном компьютере), что, очевидно, существенно снижает общую стоимость системы и нередко является подходящим решением. Однако использование универсальных процессоров имеет ряд ограничений по масштабируемости [Amdahl, 1967], [Gustafson, 1988], [Воеводин, 2010] и не подходит для реализации сверх сложных систем. В случае такого рода систем, предъявляющих особо высокие требования к производительности либо максимально возможному объему знаний, могут быть использованы спецпроцессоры (суперкомпьютеры) [Hillis, 1989], [Kitano et al., 1992], [Каляев и др., 2011], однако это зачастую невозможно из-за большой стоимости таких систем. К другим явным недостаткам можно отнести массогабаритные характеристики суперкомпьютеров, повышенные затраты на обслуживание (выделение специального помещения, охлаждение и т.п.) и обслуживающий персонал.

Другим хорошо известным решением является использование универсальных параллельных

систем, таких как GPU [Brodtkorb et al., 2013], кластеров, облачных вычислений, что позволяет достичь гораздо большей производительности по сравнению с обычными универсальными процессорами при сохранении относительно низкой стоимости всей системы в целом. Одним из недостатков такого рода систем является тот факт, что изначально они создавались для решения совершенно другого класса задач, соответственно в случае интеллектуальных систем они зачастую не могут быть использованы с максимальной эффективностью.

Альтернативное решение, предлагаемое авторами настоящей работы, заключается в разработке проблемно-ориентированного процессора [Байрак и др., 2012]. Из-за изначальной ориентации на решение задач семантической обработки является возможным достижение лучшей эффективности по сравнению с универсальными решениями (универсальными процессорами и универсальными параллельными процессорами) при сохранении низкой общей стоимости. В статье, в частности, приводится краткое описание разработанной архитектуры, системы команд и формата данных процессора, рассматриваются объективные закономерности и ограничения реализации процессора.

1. Архитектура процессора

В работе [Вереник и др., 2012] было показано, как задачи семантического анализа могут быть сведены к задачам из теории графов, в частности как произвольная семантическая сеть может быть представлена некоторым графом регулярной структуры. В свою очередь, задачи из теории графов в большинстве своем относятся к комбинаторным задачам, которые на практике могут быть решены только за счет введения ряда ограничений в условия исходной задачи. Подобные ограничения вводятся до тех пор, пока время решения задачи не достигнет необходимого допустимого значения. В результате полученное решение имеет смысл только в контексте прикладной задачи, для которой оно было разработано.

С другой стороны, стоит задача обеспечения рентабельности разработки проблемно-ориентированного процессора, которая предполагает определенные временные, а значит и материальные затраты. Такого рода проект никогда не достигнет рентабельности, если будет ориентирован только на одну частную задачу. Следовательно, с самого начала в разработку архитектуры процессора необходимо заложить определенную унификацию, т.е. ориентировать процессор на решение определенного класса схожих задач [Байрак и др., 2012].

Авторами предлагается реализовать аппаратную поддержку базовых операций, используемых в алгоритмах из теории графов, в частности операций поиска элементов графа и операции над множествами элементов графа. Фактически, процессор предлагаемой архитектуры может рассматриваться как сложная ассоциативная память, где операции чтения соответствует выполнение некоторого поискового запроса по графу.

На рис. 1 изображена структурная схема процессора. В качестве базовой архитектуры предлагается использовать SIMD-архитектуру (по классификации Флинна) магистрального типа. Из очевидных преимуществ данной архитектуры можно назвать простоту связей между модулями, наименьшие аппаратные затраты, высокую степень модульности и, что важно, возможность наращивания. Известные недостатки, такие как большое количество связей в схеме и связанная с этим невысокая надежность (в случае повреждения одной из шин данных), не являются существенными при реализации схемы на ПЛИС. Наибольшую проблему составляет ограниченная пропускная способность шин данных, в частности при чтении данных из памяти.

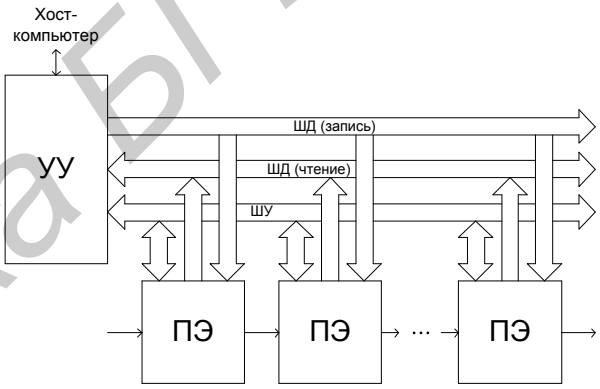

Рисунок 1 – Структурная схема процессора

На схеме процессор представлен одним общим устройством управления (УУ), которое считывает и декодирует последовательность команд исходной программы (одиночный поток команд), и множеством процессорных элементов (ПЭ), на которых в последствие параллельно исполняются эти команды. УУ взаимодействует с линейкой ПЭ через глобальные шины данных (ШД) трех видов:

- ШД записи – шина, по которой исполняемая команда параллельно транслируется на все ПЭ;
- ШД чтения – шина, по которой выполняется последовательное чтение данных с процессорных элементов (ШД с временным разделением);
- шина управления (ШУ) – множество управляющих сигналов.

Каждый ПЭ содержит свою собственную локальную память, за взаимодействие с которой отвечает только он один. ПЭ соединены между собой однонаправленными локальными связями, посредством которых осуществляется возможность приоритетного считывания данных из памяти.

Совокупность локальной памяти всех ПЭ составляет общую память процессора.

Рассмотрим общий механизм взаимодействия с памятью процессора. На данный момент система команд процессора представлена всего одной командой (см. рис. 2), используемой как для чтения, так и для записи данных в процессор.

одна ячейка	тип операции	M_A, маска-адрес	D_A, данные-адрес	M_D, маска-данные	D_D, данные-данные

Рисунок 2 – Формат команды процессора

Для адресации (выбора одного или нескольких элементов памяти процессора, с которыми будет произведено действие) в состав команды входит битовая маска M_A и битовое поле D_A, размерность которых совпадает с разрядностью процессора. Если в ячейке находится значение D_0, то она считается адресованной (выбранной, активной) в случае, когда все биты D_A и D_0, отмеченные маской M_A, соответственно равны между собой, т.е. $D_{Ai} = D_{0i}$ для $\forall i$, $i = \overline{0, b-1}$ (где b – разрядность процессора), такого что $M_{Ai} = 1$.

Часто требуется (например, в случае операции чтения) гарантировано применить действие к не более чем одному элементу памяти. Для выбора этого режима используется бит “одна ячейка”. В этом случае, если была адресована более чем одна ячейка, выбор той, которая выполнит операцию, будет осуществлен по схеме приоритетов. Собственно приоритет определяется порядком следования в схеме. Для простоты можно считать, что сработает та ячейка, которая занимает наиболее левое положение в линейке ПЭ (см. рис. 1).

Операция записи данных в ячейку памяти также использует доступ через маску, в частности значение M_D определяет те биты ячейки, которые будут изменены в зависимости от текущего значения бита согласно некоторой функции $f(D_{0i}, D_{Di}) \Rightarrow D_{0i}$ для $\forall i$, $i = \overline{0, b-1}$, такого что $M_{Di} = 1$. Выбор функции осуществляется посредством поля “тип операции”. На текущий момент предполагается реализовать простое присваивание и, как минимум, основные функции бинарной логики (И, ИЛИ, НЕ).

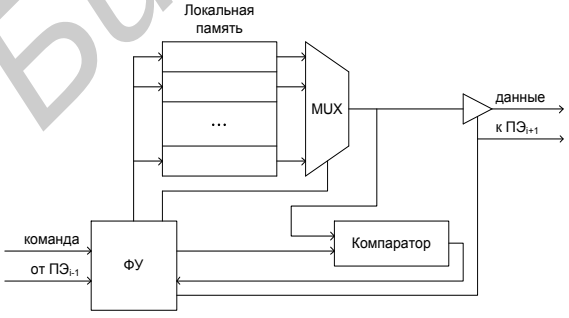

Рисунок 3 – Структурная схема ПЭ

На рисунке 3 представлена структурная схема ПЭ. В состав ПЭ входит локальная память и комбинационная схема, отвечающая за:

- последовательное сравнение ячеек памяти с входным значением (реализуется механизм адресации описанный выше);
- побитовую запись данных в ячейку (в соответствие со значением маски M_D);
- реализацию логической функции, вычисляющей значение для записи в ячейку;
- взаимодействие с соседними ПЭ (схема приоритетов);
- чтение данных.

2. Некоторые закономерности и объективные ограничения

Основная идея при построении процессора заключается в том, чтобы сохранить функциональную сложность ПЭ максимально простой, что в результате позволит реализовать огромное их количество на одном чипе. В то же время процессор должен быть в некотором роде универсальным (для обеспечения рентабельности проекта, как объяснялось ранее), что не может не повлиять на структуру ПЭ. В дополнение ко всему, требования предъявляемые конкретной прикладной системой, для которой изготавливается процессор, могут существенно отличаться от одной системы к другой. Некоторые системы требуют максимальной производительности (скорости выполнения), для других, в первую очередь, важен максимальный объем хранимой памяти, для третьих главным ограничивающим фактором будет являться стоимость.

Для решения данной проблемы предполагается поддержка процессором определенной настройки под конкретную прикладную систему (прикладную задачу), для которой этот процессор будет изготавливаться. Для предлагаемой архитектуры такого рода настройка может быть выполнена на нескольких концептуальных уровнях. В частности, можно выделить аппаратную настройку процессора (выбор разрядности данных процессора, количества ПЭ, объем локальной памяти) и программную настройку процессора (различная интерпретация формата данных процессора). Остановимся на аппаратной настройке процессора, рассмотрим некоторые закономерности и ограничения, присущие предлагаемой архитектуре.

1. Разрядность процессора b – в полной мере зависит от решаемой на системе задачи. Например, для хранения семантической сети в памяти процессора ячейка памяти должна иметь возможность вместить представление любого произвольного элемента сети. Так, для представления вершины графа необходимо хранить некоторый уникальный идентификатор вершины и ряд значений ее свойств (атрибутов). Для представления дуги графа – идентификаторы двух

инцидентных ей вершин и некоторые свойства дуги. Увеличение разрядности процессора *b* приводит к:

- увеличению объема локальной памяти ПЭ и, соответственно, увеличению схемы памяти ПЭ;
- увеличению комбинационной схемы управляющей логики ПЭ;
- уменьшению скорости исполнения управляющей логики ПЭ (дополнительные задержки на мультиплексорах и т.п.);
- увеличению размера команды (в 4 раза быстрее, чем разрядность *b*, при грубой оценке);
- увеличению количества шин данных и, вместе с тем, общей сложности трассировки схемы;
- росту стоимости одной ячейки памяти.

Стоимость одной ячейки памяти процессора можно принять равной стоимости изготовленного чипа деленной на общее количество ячеек памяти, которые и представляют собой основную ценность. Фактически, изменение стоимости (рост или падение) определяется отношением площади комбинационной схемы (КС) процессора (всей управляющей логики) к площади, занимаемой собственной элементами памяти (ЭП) процессора:

$$K = \frac{S_{KC}}{S_{ЭП}}.$$

Увеличение значения величины *K* приводит к росту стоимости одной ячейки памяти, уменьшение значение – к падению стоимости.

2. Количество элементов локальной памяти *M*, приходящееся на один ПЭ. Увеличение значения *M* приводит к:

- увеличению объема локальной памяти ПЭ и, соответственно, увеличению схемы памяти ПЭ;
- увеличению комбинационной схемы управляющей логики ПЭ;
- увеличению времени выполнения операции процессорным элементом (чем больше ячеек памяти содержится в ПЭ, тем больше времени требуется для их последовательной обработки);
- уменьшению производительности (количество операций выполняемых в единицу времени);
- падению стоимости одной ячейки памяти.

Очевидно, что для обеспечения максимальной производительности системы необходимо максимально уменьшить значение переменной *M*, т.е. уменьшить количество ячеек памяти, приходящееся на ПЭ и, соответственно, увеличить общее количество ПЭ на чипе. Однако это будет сопровождаться ростом стоимости одной ячейки памяти, что не всегда является допустимым.

3. Общее количество ПЭ на чипе *N*. В большинстве случаев является производной величиной от значения разрядности процессора *b*, размера локальной памяти *M* и максимальной допустимой площади схемы S_{MAX}, т.е. после выбора конкретного значения основных показателей системы мы размещаем на чипе максимально возможное количество соответствующих ПЭ:

$$S = S_{KC} + S_{ЭП} \rightarrow S_{MAX}.$$

В общем случае увеличение количества ПЭ приводит к:

- увеличению общего объема памяти процессора;
- увеличению эффективности процессора (больший объем памяти обрабатывается за то же самое время);
- увеличению количества шин данных и, вместе с тем, общей сложности трассировки схемы.

Отметим, что максимальное количество ПЭ ограничено некоторым максимальным числом N_{MAX}, что обусловлено наличием локальных связей между ПЭ (задержками на последовательной схеме). Решение данной проблемы в настоящей статье не рассматривается.

3. Пример использования

Вкратце рассмотрим пример программной “настройки” процессора, основной смысл которой заключается в построении дополнительного уровня абстракции над системой команд процессора, который [уровень] жестко ориентирован на конкретную решаемую задачу. Для этого определяется свой собственный формат данных процессора (конкретные биты ячеек памяти наделяются нужным смыслом) и вводятся новые команды процессора, работающие в терминах решаемой задачи. Реализация такой расширенной системы команд может быть выполнена как прикладным программистом на уровне его среды разработки, так и системным программистом на уровне УУ процессора, который может представлять собой перепрограммируемый контроллер.

Used	Cell type	ID	Атрибуты вершины				
			v	n	f	$distance_{min}$	prevID
	0	k-1 0				m-1 0	k-1 0

Used	Cell type	ID1	ID2	Атрибуты дуги	
				c	cost
	1	k-1 0	k-1 0		m-1 0

Рисунок 4 – Формат данных процессора для решения задачи поиска кратчайшего пути в графе

В работах [Вереник и др., 2013], [Tatur et al., 2013] на базе программной модели предлагаемой архитектуры процессора была решена задача поиска кратчайшего пути в графе. Для решения задачи использовался параллельный алгоритм, разработанный на основе алгоритмы Дейкстры и волнового алгоритма. На рис. 4 показан выбранный формат данных процессора (ячейка памяти может содержать 2 различных типа данных – вершины и дуги), а ниже приведен расширенный список команд:

- инициализация (устанавливается начальное значение для всех ячеек памяти в соответствии с используемым алгоритмом)
- создать вершину с заданным идентификатором;
- создать дугу с заданным идентификатором и стоимостью перехода;
- установить для заданной вершины минимальное найденное расстояние и идентификатор предыдущей вершины (для возможности восстановить траекторию пути после);
- установить для заданной вершины флаг вхождения во фронт волны;
- считать вершину по ее идентификатору;
- считать следующую вершину из фронта волны;
- сдвинуть фронт волны;
- найти все дуги выходящие из заданной вершины;
- считать следующую дугу из найденных ранее.

Заключение

В настоящей работе была представлена архитектура процессора, предназначенная для решения задач на графах и, в частности, эффективного исполнения алгоритмов семантической обработки информации. Были рассмотрены основные принципы функционирования процессора, формат данных и система команд, структура процессорного элемента, некоторые закономерности и объективные ограничения реализации процессоров данной архитектуры. В частности, были описаны наиболее важные параметры архитектуры процессора, выбор значения которых и представляет собой настройку архитектуры для решения конкретной прикладной задачи.

В рамках разработки предложенной архитектуры следующим этапом исследования планируется реализовать архитектуру на существующих параллельных платформах, таких как GPU (технология CUDA), кластере и FPGA для оценки реального прироста производительности и эффективности при решении различных типовых задач из области семантического анализа по сравнению с активно используемыми на сегодня решениями.

Библиографический список

[Байрак и др., 2012] Параллельные процессоры для построения интеллектуальных систем / С. А. Байрак, Д. Н. Одинец, М. М. Татур, Ф. Филипов, М. Мунос // Открытые семантические технологии проектирования интеллектуальных систем : материалы II Междунар. научн.-техн. конф. (Минск, 16-18 февраля 2012 г.) / редкол. : В. В. Голенков (отв. ред.) [и др.]. – Минск : БГУИР, 2012. – С. 135–140.

[Вереник и др., 2012] Разработка проблемно-ориентированных процессоров семантической обработки информации / Н. Л. Вереник, Е. Н. Сейткулов, М. М. Татур // Электроника инфо. – 2012. – № 8. – С. 95–98.

[Вереник и др., 2013] Имитационная модель векторного процессора на примере задачи поиска пути в графе / Н. Л. Вереник, А. И. Гирель, Е. Н. Сейткулов, М. М. Татур // Искусственный интеллект. – 2013. – №4. – С. 89–100.

[Воеводин, 2010] Воеводин В. В. Вычислительная математика и структура алгоритмов. – М.: Издательство Московского университета, 2010. – 168 с.

[Голенков и др., 2012] Графодинамические модели параллельной обработки знаний: принципы построения, реализации и проектирования / В. В. Голенков, Н. А. Гулякина // Открытые семантические технологии проектирования интеллектуальных систем : материалы II Междунар. научн.-техн. конф. (Минск, 16-18 февраля 2012 г.) / редкол. : В. В. Голенков (отв. ред.) [и др.]. – Минск : БГУИР, 2012. – С. 23–52.

[Каляев и др., 2011] Высокопроизводительные реконфигурируемые вычислительные системы нового поколения / И. А. Каляев, А. И. Дордопуло, И. И. Левин, Е. А. Семерников // Вычислительные методы и программирование: новые вычислительные технологии. – 2011. – Т. 12. – №2. – С. 82 – 89.

[Amdahl, 1967] Gene M. Amdahl. Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities // AFIPS Conference Proceedings. – Volume 30, April 18-20, 1967. – pp. 483–485

[Brodtkorb et al., 2013] André R. Brodtkorb, Trond R. Hagen, Martin L. Sætra. Graphics processing unit (GPU) programming strategies and trends in GPU computing // Journal of Parallel and Distributed Computing. – Volume 73, Issue 1, January 2013. – pp. 4–13.

[Gustafson, 1988] John L. Gustafson. Reevaluating Amdahl's Law // Communications of the ACM. – Volume 31, Issue 5, May 1988. – pp. 532–533.

[Hillis, 1989] W. D. Hillis. The Connection Machine. – Cambridge : The MIT Press, 1989. – 208 p.

[Kitano et al., 1992] Hiroaki Kitano, Dan Moldovan “Semantic Network Array Processor as a massively parallel computing platform for high performance and large-scale natural language processing”, Proc. Int’l Conf. on Computational Linguistics (COLING '92), vol. 2, pp. 813–819, 1992.

[Tatur et al., 2013] M.M. Tatur, Y.N. Seitkulov, N.L. Verenik, A.I. Girel "Pathfinding on a Specialized Vector Processor", Proc. Int'l Conf. Parallel and Distributed Processing Techniques and Applications (PDPTA '2013), vol. 1, pp. 711–716, 2013.

SOME REGULARITIES AND OBJECTIVE LIMITATIONS WHEN IMPLEMENTING SEMANTIC PROCESSING ALGORITHMS ON COMPUTER SYSTEMS WITH MASSIVE PARALLELISM

Nick L. Verenik*, Yerzhan N. Seitkulov**, Alexey I. Girel*, Mikhail M. Tatur*

** Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus*

nick.verenik@gmail.com

tatur@i-proc.com

*** L.N. Gumilyov Eurasian National University, Astana, Kazakhstan*

seitkulov_y@enu.kz

In the article description of semantic ASIP architecture which is in development is given, processor’s data format and instruction set, basic principles of functioning are considered. Several architecture limitations and objective regularities are given. Usage example of processor for solution of the path in graph problem is briefly considered.

Key words: semantic information processing; semantic network; ASIP; massive parallel processing.

Introduction

Using of semantic networks is one of the most perspective approaches to formalize functioning of custom intelligent system. They can be used to represent both system's knowledge (knowledge database) and algorithms of intelligent information processing which are used in this system.

In practice when building applied intelligent systems then program model of semantic network is used and usually implemented on general-purpose computer system (PC). Obviously this approach significantly decreases the total cost of the system and frequently appears as a suitable solution. However using of general-purpose processors has number of scalability limitations. When implementing really complex intelligent systems, which requires especially high performance or huge value of knowledge database, supercomputers can be used. But this is often impossible because of high cost of such systems, weight and size characteristics or difficulties in maintenance.

Another well-known solution is using of general-purpose parallel systems such as GPU, clusters, and clouds computing. It allows reach much more performance in comparison with common general-purpose processors and keep relatively low cost of the entire system. The main disadvantage of this approach is the fact that such computer systems were developed to solve different kind of problems but semantic processing and as the result it's very difficult to achieve maximum of efficiency.

Authors have proposed alternative solution which is development of ASIP. Its problem orientation allows achieve better performance and efficiency in comparison with general-purpose processors (including parallel processors) while keeping low total cost. In the article brief description of developed architecture, its instruction set and data format are given, objective regularities and limitations of implementation are given.

Main Part

It's known that problems of semantic analysis can be reduced to problems in graph theory. Then it's known that problems in graph theory are mostly combinatorial problems and in practice they can be solved only after some limitations in original problem are introduced. Such limitations are introduced until the duration of problem solution reaches the required value. As the result the final solution has sense only in context of this specific applied problem.

On the other hand there's problem to ensure profitability of ASIP development which supposes some time and hence the material spending. Such project will never reach profitability if it's focused on only one specific applied problem. Therefore from the beginning the certain unification should be taken as basis while developing processor architecture, i.e. there's the necessarily to orient it on the solution of some kind of similar problems.

Authors have proposed to implement hardware support of basic operations used in graph theory algorithms, in particular graph elements searching and working with set of graph elements. Practically, the processor with proposed architecture can be considered as a complex associative memory where read operation corresponds to execution of some search request within graph structure.

Schematic diagram of processor is presented in fig.1. The original architecture belongs to SIMD-kind where processor elements (PEs) are connected by three global buses (data bus for reading, data bus for writing and control bud) and locally connected with each other by unidirectional links (to perform mechanism of reading in order of priority). Each PE contains some value of local memory and is responsible for access to it. The approximate schematic diagram of PE is presented in fig.2.

The basic idea of processor is to provide bitwise access to memory cells which means to address memory by comparison of selected bits in cell and in input command. Selection mechanism is implemented via special mask field in command (see command format in fig.3). Writing operation is performed by bits as well.

The processor architecture is designed to support two-level customization. First of all number of low level parameters can be adjusted – such as processor capacity, number of memory cells per one PE, number of PEs. The regularities while changing these parameters are considered in the article. The next level of customization is performed by application or system engineers and includes the introduction of custom data format and high level instruction set which are completely determined by and formulated in terms of the problem to be solved. In fig.4 the example of such customization is presented where path finding problem is solved.

Conclusion

In the given paper the result of semantic processor development is presented. The description of proposed architecture, instruction-set and data format, basic principles of functioning are given. The solution of path finding problem is considered as the example of processor usage.

Further study and simulations are necessary for definition the problems to be solved on the proposed architecture. As the next step simulations on existing parallel systems are expected – such as GPU (CUDE technology), clusters and FPGA.