УДК 519.687

М. Б. Остапкевич [2], **С. В. Пискунов** [1, 2]

[1] Институт вычислительной математики
и математической геофизики СО РАН
пр. Акад. Лаврентьева, 6, Новосибирск, 630090, Россия

[2] Новосибирский государственный университет
ул. Пирогова, 2, Новосибирск, 630090, Россия

E-mail: ostap@ssd.sscc.ru

СИСТЕМА ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ АЛГОРИТМОВ С МЕЛКОЗЕРНИСТЫМ ПАРАЛЛЕЛИЗМОМ WinALT

Описана система имитационного моделирования алгоритмов с мелкозернистым параллелизмом WinALT. Представлены ее пользовательские характеристики, архитектура и механизмы расширения функциональных возможностей в рамках проблемной области алгоритмов с мелкозернистым параллелизмом, показаны возможности адаптации системы к новым разновидностям алгоритмов с мелкозернистым параллелизмом.

Ключевые слова: мелкозернистый параллелизм, имитационное моделирование, системы моделирования, языки моделирования, алгоритм параллельных подстановок, открытая архитектура.

Введение

Имитационные модели исполнения алгоритмов и работы вычислителей с мелкозернистым параллелизмом (МЗП) используются для исследования физических, химических, биологических, социальных и информационных процессов и структур вычислительных систем [1; 2]. Они позволяют изучать явления, для которых построение аналитических моделей затруднительно или невозможно. Большой объем данных для описания реалистических моделей требует одновременного выполнения большого числа преобразований данных и большого числа итераций.

Известны более двадцати систем имитационного моделирования МЗП алгоритмов [1], но всем им присуще ограничение – ориентация на работу клеточного автомата и его модификаций, у большей части которых закрытая архитектура. Ни одна из них не дает возможности композиционных построений сложных моделей. Вместе с тем к МЗП структурам относятся не только клеточные автоматы и их расширения, но и матричные и конвейерные (включая систолические) архитектуры, мультимикропроцессорные архитектуры, ассоциативные процессоры, клеточно-нейронные сети, однородные перестраиваемые вычислители и т. п.

Все сказанное выше делает *актуальным* построение программного комплекса с открытой архитектурой, обеспечивающего конструирование МЗП алгоритмов и структур в самой широкой трактовке термина.

Такой комплекс разработан на кафедре параллельных вычислений НГУ и в Институте вычислительной математики и математической геофизики СО РАН и назван системой имитационного моделирования WinALT [2]. Система WinALT и ее предшественница ALT [3; 4] базируются на алгоритме параллельных подстановок – алгоритмической системе, являющейся

[1] Cellular automata software: http://www.cafaq.com/soft/index.php; *Пискунов С. В., Остапкевич М. Б.* Сайт системы WinALT. URL: http://winalt.sscc.ru/

[2] *Пискунов С. В., Остапкевич М. Б.* Система имитационного моделирования мелкозернистых алгоритмов и структур WinALT. Фонд алгоритмов и программ СО РАН. PR11053, 2011. URL: http://fap.sbras.ru/node/2455

пространственной моделью для представления параллельных мелкозернистых алгоритмов и структур [4]. Алгоритм параллельных подстановок (АПП) концептуально объединяет в себе подстановочный характер алгоритма Маркова [5] с пространственной параллельностью клеточного автомата [6] и основывается на общем для них ассоциативном механизме выполнения операций. АПП позволяет описать вычисления в максимально параллельном виде, когда на каждом шаге выполняются все допустимые действия над всеми имеющимися данными.

В системе WinALT был использован ряд свойств системы ALT. Ключевыми достоинствами ALT являются: 1) возможность конструирования алгоритмов и структур с разнообразными видами МЗП; 2) визуальный подход в описании правил преобразований данных в моделях. Наряду с названными свойствами в системе WinALT появились новые полезные свойства. Главные из них перечислены ниже.

1. Реализован язык моделирования, позволяющий строить моделирующие программы из модулей и имеющий средства композиции объектов данных и средства их преобразования для моделей из всего спектра классов МЗП.

2. WinALT построена как открытая система. Ее функциональные возможности могут расширяться не только разработчиками, но и пользователями. В частности, в систему можно добавлять новые форматы данных, виды отображения данных, режимы моделирования, внешние функции, используемые в моделирующей программе.

3. Средства интерфейса системы дают пользователю возможность конструирования и модификации как данных модели, так и моделирующих программ, позволяют следить за динамикой изменения преобразуемых в модели данных.

4. Система WinALT позволяет работать с объектами данных больших размеров и, как следствие, с практически полезными моделями.

В работе представлены пользовательские характеристики системы, описаны ее архитектура и механизмы расширения функциональных возможностей в рамках проблемной области МЗП алгоритмов и структур, показаны возможности адаптации системы к новым разновидностям МЗП алгоритмов и структур.

Пользовательские характеристики системы WinALT

Система WinALT обладает следующими необходимыми для конструирования моделей и адаптации к нуждам пользователя характеристиками:

- свободное распространение;
- дружественность и адекватность пользовательского интерфейса;
- коллекция моделей на сайте системы;
- язык моделирования, соединяющий средства параллельного, последовательного программирования и средства адаптации к запросам пользователе;
- наличие графического режима, реализующего визуальное программирование;
- расширяемая модульная архитектура системы.

Свободное распространение. На сайте системы содержится раздел «инсталляция системы», включающий инструкции по установке, обновлению, удалению системы и ссылку на ее дистрибутив.

Дружественность и адекватность пользовательского интерфейса. Графический интерфейс системы WinALT обеспечивает возможность строить, отлаживать и исполнять модели любого класса МЗП.

Коллекция моделей на сайте системы. Доступны модели арифметических устройств, ассоциативных процессоров, универсальных однородных структур, ряда классических клеточных автоматов, модель диффузии, которая предложена Т. Марголусом [1], модели арифметических и геометрических фракталов, модель визуальной криптографии на основе алгоритма Шамира, модель управляющей сети Петри.

Язык моделирования, соединяющий средства параллельного, последовательного программирования и средства адаптации к запросам пользователей. Центральная часть языка служит для описания МЗП вычислений. Кроме нее имеется набор конструкций, типичных для языка структурного программирования. Они применяются, например, для инициализа-

ции объектов данных, взаимодействия с пользователем. Язык моделирования позволяет использовать процедуры, реализованные вне системы моделирования, в моделях.

Наличие графического режима, реализующего визуальное программирование. Наряду с развитыми средствами аналитического описания моделей, система позволяет описывать данные и некоторые правила их преобразования визуально.

Расширяемая модульная архитектура системы. Набор функций системы может расширяться как разработчиками системы, так и ее пользователями, имеющими навыки программирования, с помощью внешних модулей. При их добавлении в систему не требуется производить никаких модификаций в ней самой.

Архитектура системы WinALT

Основными подсистемами системы являются консольная версия, графическая оболочка и четыре библиотеки: модулей поддержки форматов данных, режимов моделирования, модулей языковых функций и модулей видов отображения (рис. 1).

Консольная версия системы. Назначение консольной версии системы – быть ядром всех прочих компонентов WinALT, которые расширяют ее функциональные возможности по адаптации к предметным областям пользователей. Главные компоненты самой консольной версии – менеджер данных, компилятор языка моделирования и виртуальная машина. Менеджер данных реализует общие для всех остальных компонент WinALT функции по организации модульной архитектуры всей системы, по работе со структурами данных. Компилятор проверяет синтаксическую корректность моделирующей программы, написанной на языке моделирования WinALT, и выдает пользователю сообщения об ошибках. Затем он генерирует код программы, который исполняется виртуальной машиной. Консольная версия системы написана на языке C, чтобы быть выполнимой как в ОС Windows, так и в ОС Unix.

Язык моделирования системы WinALT содержит три взаимосвязанных части. Первая часть предназначена для описания МЗП вычислений. Вторая часть предназначена для описания вспомогательных, технологических частей моделей. Третья – обеспечивает импорт в моделирующую программу встроенных в систему библиотек.

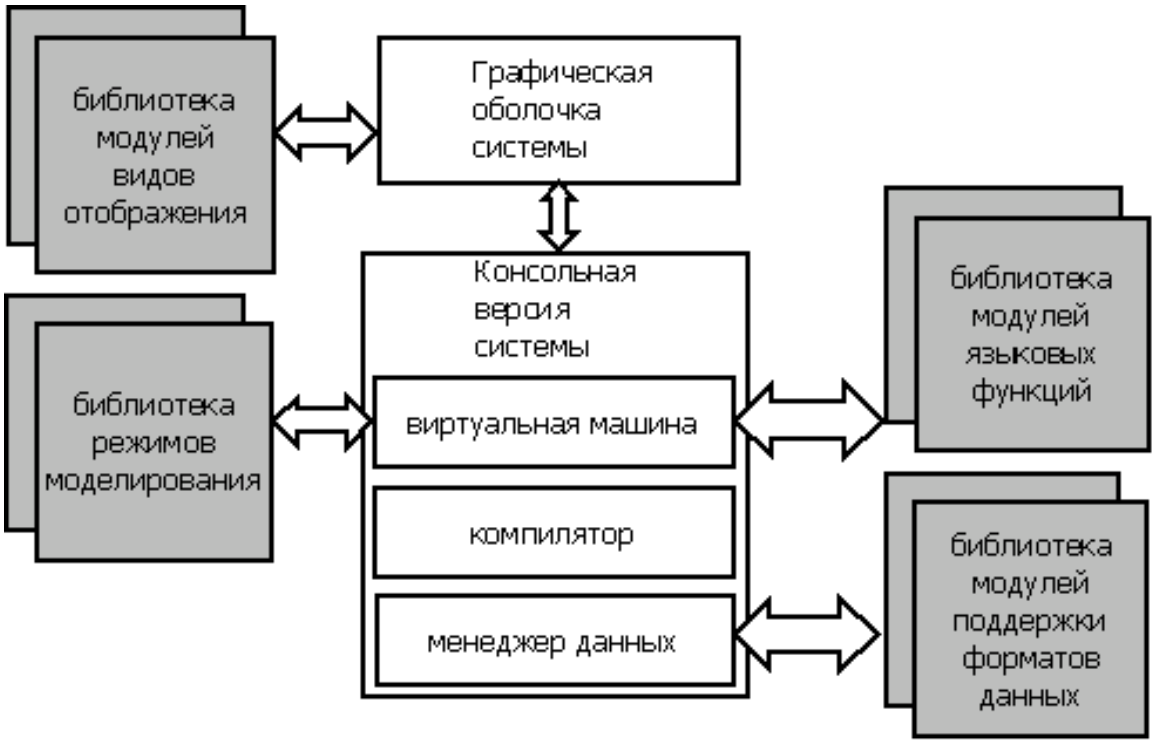

Рис. 1. Основные подсистемы WinALT

Алгоритм параллельных подстановок. Теоретическим фундаментом первой части языка является АПП. Главная идея АПП заключена в следующих трех положениях.

- Обрабатываемая информация задается в виде клеточного массива S – множества клеток, в котором каждая клетка имеет состояние и имя. Состояние – это данное (бит, символ, число и т. д.) из некоторого конечного алфавита A; имя – это место данного в массиве, например, в двумерном клеточном массиве, который представляется прямоугольной таблицей, клетка именуется парой координат. Множество имен обозначается M.
- Алгоритм задается множеством параллельных подстановок вида П: $S_1 * S_2 \to S_3$, где S_1, S_2 и S_3 – клеточные подмассивы, звездочка между S_1 и S_2 – знак композиции, т. е. подмассивы S_1 и S_2 рассматриваются вместе. Левая часть $S_1 * S_2$ определяет условие применимости подстановки и состоит из двух частей – базы S_1 и контекста S_2. Правая часть S_3 задает новые состояния клеткам базы при выполнении подстановки. Состояния клеток своего контекста подстановка не изменяет. Выразительные возможности АПП увеличиваются, если ввести в алфавит A, кроме битов, символов, чисел, *переменные* и *функции*. Подстановка, использующая *переменные и функции*, называется функциональной в отличие от подстановки простой, использующей биты, числа и символы. Использование функциональных подстановок позволяет: а) сделать запись АПП существенно компактней, чем при использовании простых параллельных подстановок; б) представить в виде отдельной клетки достаточно сложные устройства, например, АЛУ.
- Процесс вычисления имеет итерационный характер: на каждом временном шаге выполняются одновременно все применимые в клеточном массиве параллельные подстановки. Вычисление заканчивается тогда, когда к полученному на предыдущем шаге клеточному массиву не применима ни одна подстановка. Этот массив и есть результат работы АПП.

Формальное описание АПП представлено в [4]. Мы проиллюстрируем идею АПП простым примером АПП Φ_σ сложения многих целых положительных двоичных чисел.

$A = \{0, 1\}$, $M = N \times N$, $N = \{0, 1, 2, \ldots\}$. Перерабатываемый клеточный массив – это двумерная прямоугольная таблица, чьи клетки перенумерованы парами чисел в соответствии с левой системой координат (i – абсцисса; j – ордината). Имя m клетки таблицы – пара чисел i, j. Двоичные слагаемые расположены в строках таблицы. АПП Φ_σ состоит из двух простых параллельных подстановок:

$$\Phi_\sigma = \begin{cases} \Theta_1^\sigma: \ \{(1,\langle i, j\rangle)(1,\langle i, j+1\rangle)(0,\langle i+1, j\rangle)\} * \{(0,\langle i, j-1\rangle)(0,\langle i+1, j-1\rangle)\} \to \\ \qquad \{(0,\langle i, j\rangle)(0,\langle i, j+1\rangle)(1,\langle i+1, j\rangle)\}, \\ \Theta_2^\sigma: \ \{(1,\langle i, j\rangle)(0,\langle i, j+1\rangle)\} * \{(0,\langle i, j-1\rangle)\} \to \{(0,\langle i, j\rangle)(1,\langle i, j+1\rangle)\}. \end{cases}$$

В записи подстановок в угловых скобках записаны функции сдвига, задающие положение клеток левой и правой частей подстановки относительно друг друга. Описание клетки заключено в круглые скобки. Состояние клетки показано слева от запятой. При подстановке в функции сдвига конкретных значений i, j мы получаем ассоциированные с этим именем клеточные массивы. Такое описание подстановок допускает их графическое представление (рис. 2). В этом случае левые и правые части подстановок задаются с помощью шаблонов (геометрических образов), а поиск вхождений левых частей подстановок выполняется при перемещении их шаблонов по таблице. Пример исходной таблицы содержит слагаемые 9, 15, 5 (рис. 3). После четырех шагов преобразования этой таблицы в верхней строке таблицы будет получена сумма, все остальные строки станут нулевыми.

Рис. 2. Графическое представление команд АПП клеточного сумматора

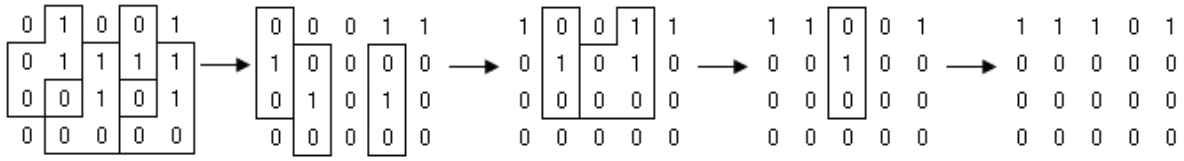

Рис. 3. Шаги выполнения алгоритма клеточного сумматора

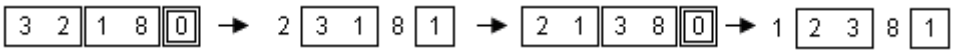

Рис. 4. Исполнение алгоритма сортировки на четырех элементах

Замечание. Поскольку в АПП допускается произвольный шаблон подстановки, то возможна ситуация, в которой одна и та же клетка оказывается в зоне применимости двух подстановок. Если эта клетка в обеих подстановках является контекстной или в одной – контекстной, а в другой – базовой, то не возникает проблемы с изменением ее состояния. Не возникает проблемы и в том случае, когда клетка является базовой в обеих подстановках и ее новое состояние является одним и тем же в этих подстановках. Противоречие в применимости подстановок возникает, если состояние общей клетки изменяется двумя подстановками по-разному. АПП должен содержать непротиворечивое множество подстановок. В [4; 7] сформулированы критерии непротиворечивости и даны способы проверки множества параллельных подстановок на непротиворечивость.

В качестве примера АПП с функциональными подстановками рассмотрим АПП Φ_Δ сортировки данных четно-нечетными перестановками [4], который применяется к компонентам вектора: $X = \{x_1,...,x_n\}$, где x_i – действительные числа. На каждом нечетном шаге сравниваются пары компонентов $(x_{2i-1},\ x_{2i})$ (разной величины иксы здесь и в предыдущих записях). Если $x_{2i-1} > x_{2i}$, то компоненты меняются местами. На каждом четном шаге сравниваются пары компонентов $(x_{2i},\ x_{2i+1})$. Если $x_{2i} > x_{2i+1}$, то компоненты меняются местами. Множество имен задано как $M = \{1, 2, \ldots, n, n+1\}$. Клетка с именем $n+1$ введена для организации чередования подстановок на четных и нечетных шагах. В подстановках используются функции f_1 и $f_2 : f_1(x,y) = y$, $f_2(x,y) = x$, если $x > y$, иначе $f_1(x,y) = x$, $f_2(x,y) = y$. Одномерный клеточный массив в начальном состоянии определен как $W = \{(x_1, 1)...(x_n, n)(0, n+1)\}$. Подстановки, выполняющие сортировку, определены следующим образом:

$$\Phi_\Delta = \begin{cases} \Theta_1 :\ \{(x,2i-1)(y,2i)(0,n+1)\} \to \{(f_1(x,y),2i-1)(f_2(x,y),2i)(1,n+1)\}, \\ \Theta_2 :\ \{(x,2i)(y,2i+1)(1,n+1)\} \to \{(f_1(x,y),2i)(f_2(x,y),2i+1)(0,n+1)\}. \end{cases}$$

где *i* принимает значение от 1 до *n*.

Рассмотрим пример исполнения алгоритма сортировки в одномерном клеточном массиве при *n* = 4 для начального состояния $W = \{(3,1)(2,2)(1,3)(8,4)(0,5)\}$ (рис. 4). Двойная рамка у клетки (0, 5) означает, что эта клетка используется одновременно в двух подстановках.

Язык моделирования. В первой части языка, которая служит для описания МЗП-вычислений и является нотацией АПП, присутствует отчетливое разделение на графическую и аналитическую части. Графическими объектами являются одно-, двух-, трехмерные клеточные массивы и шаблоны подстановок. Образ объекта составляется из цветных клеток, в общем случае расположенных вдоль вертикальной и горизонтальной осей, а также оси, уходящей от пользователя вглубь экрана. Цвет – это отражение состояния клетки. Состояние

клетки может задаваться любым типом данных из библиотеки форматов данных системы (см. ниже). Пустая клетка в шаблоне обозначается косым крестом и имеет тип *void.*

Для описания простой параллельной подстановки, меняющей состояния клеток внутри одного клеточного массива, в языке WinALT используется конструкция

in C
    at L
        do R

Оператор *in* задает клеточный массив *C*, в котором параллельная подстановка должна выполняться. Единственным параметром оператора является имя *C* этого клеточного массива. Оператор *at* задает шаблон *L* левой части подстановки, а оператор *do* – шаблон *R* правой части подстановки. Одна из клеток шаблона *R* назначается центральной (центр шаблона). Подстановка выполняется в два такта. На первом такте при перемещении центра шаблона левой части подстановки вдоль осей по перерабатываемому клеточному массиву отмечаются все вхождения этого шаблона. На втором такте во всех вхождениях левой части подстановки состояния клеток заменяются на состояния соответствующих клеток из шаблона правой части.

Для описания подстановки, меняющей состояния клеток более чем одного массива, используется конструкция

in (C_1; ... C_i)
    at (L_1; ... L_j)
        do (R_1; ... R_k)

Длины списков такой подстановки, называемой векторной, связаны отношением $i \geq j \geq k$. Параметром оператора *in* является список имен клеточных массивов C_1, … C_i, в которых выполняется подстановка. Параметры операторов *at* и *do* – это списки имен шаблонов левой и правой частей подстановки. Объединение шаблонов в список означает, что перемещение шаблонов по изображениям соответствующих клеточных массивов осуществляется согласованно путем подстановки в центры всех шаблонов из операторов *at* и *do* одного и того же набора значений координат. Наборами служат значения координат клеток клеточного массива, занимающего первую позицию в списке оператора *in*.

Функциональная подстановка описывается конструкцией

in C
    at L
        do F(*c*)

Она всюду применима в клеточном массиве *C* и вычисляет новые состояния клеток с помощью функции *F*. Отличается от простой тем, что: 1) все клетки шаблона *L* находятся в состоянии «пусто»; 2) как минимум одна его клетка поименована некоторым символом переменной *c*; 3) параметр оператора *do* – имя функции *F*, вычисляющей новые состояния клеток; 4) в скобках после имени функции *F* приводится список имен тех переменных, которые данная функция изменяет; 5) функции *F* допускается использование переменных с именами, упомянутыми в шаблоне *L*.

Для описания определенной группы параллельно исполняемых подстановок применяется блок параллельного исполнения, называемый синхроблоком. В языке имеется три вида синхроблоков.

Синхроблок, имитирующий однократное параллельное исполнение всех содержащихся в нем подстановок, описывается конструкцией

change
    …
end

Синхроблок, задающий циклическое параллельное исполнение всех содержащихся в нем подстановок с указанием числа итераций, описывается конструкцией:

clock E;
    …
end

Целочисленный результат выражения *E* задает число итераций. В простейшем случае оно может быть константой.

Синхроблок, циклически исполняющий подстановки до их неприменимости, описывается конструкцией:

exhaust

…

end

Если в подобном синхроблоке на некоторой итерации не найдено ни одной применимой подстановки, происходит выход из него.

Замечание. Для описания композиций параллельных и последовательных вычислений синхроблок может содержать не только операторы подстановок, но и операторы структурного программирования. Отметим также, что в языке системы WinALT могут использоваться вложенные друг в друга синхроблоки. Описанные возможности позволяют конструировать любые параллельно-последовательные композиции синхронных преобразований клеточных массивов.

Моделирующая программа состоит из перечня включаемых в программу библиотек, описаний констант, переменных, клеточных объектов, процедур, функций и тела программы.

Графическая оболочка системы. Является надстройкой над консольной версией системы, позволяет исследователю пройти всю цепочку работы с моделью, от ее создания и наполнения данными, через отладку, до ее исполнения и получения конечных результатов.

Графическая оболочка системы позволяет:

- создавать проект, представляющий модель;
- создавать в проекте произвольное количество окон с моделирующими программами и полотнами (рис. 5) для работы с клеточными массивами;
- создавать и размещать на полотне любое количество произвольно расположенных клеточных массивов;
- загружать в проект существующее полотно;
- просматривать дерево всех объектов, добавленных в проект (окон, полотен и клеточных массивов), переходить к окну или полотну, или редактированию при нажатии на соответствующее имя объекта;
- загружать и сохранять клеточный объект как файл в некотором формате, обеспечивая совместную работу с различными приложениями и системами моделирования;
- менять состояние отдельной клетки или состояния клеток в группах, инициализировать состояния клеток в группах псевдослучайными числами, копировать и перемещать состояния клеток группы как внутри одного клеточного объекта, так и между разными клеточными объектами;
- менять размер клеточного массива по всем осям, вставлять или удалять слой клеток по выбранной оси;
- менять режим отображения клеточного объекта, поворачивать, задавать положение клеточного объекта или группы клеточных объектов на полотне;
- давать имя части клеточного объекта, обеспечивая возможность работы с ней как с полноценным клеточным объектом (создавать виртуальный клеточный массив);
- редактировать тексты программ, производить их синтаксическую раскраску, переходить от идентификатора в программе к редактированию соответствующего объекта данных;
- запускать и останавливать исполнение или отладку модели;
- вводить данные с помощью диалоговых окон, показывать содержимое объектов данных, осуществлять консольный вывод в любой момент исполнения модели;
- выводить сообщения об ошибках при компиляции программ в окне вывода компилятора, переходить от этих сообщений к соответствующим местам в программе;
- приостанавливать и возобновлять исполнение, производить пошаговое исполнение, наблюдать за стеком вызовов процедур, состояниями переменных и клеток в процессе отладки модели.

Графическая оболочка, как и вся система моделирования, имеет открытую архитектуру и предоставляет разработчикам и пользователям механизм для добавления новых режимов отображения и редактирования данных в систему, который основан на использовании библиотек внешних модулей.

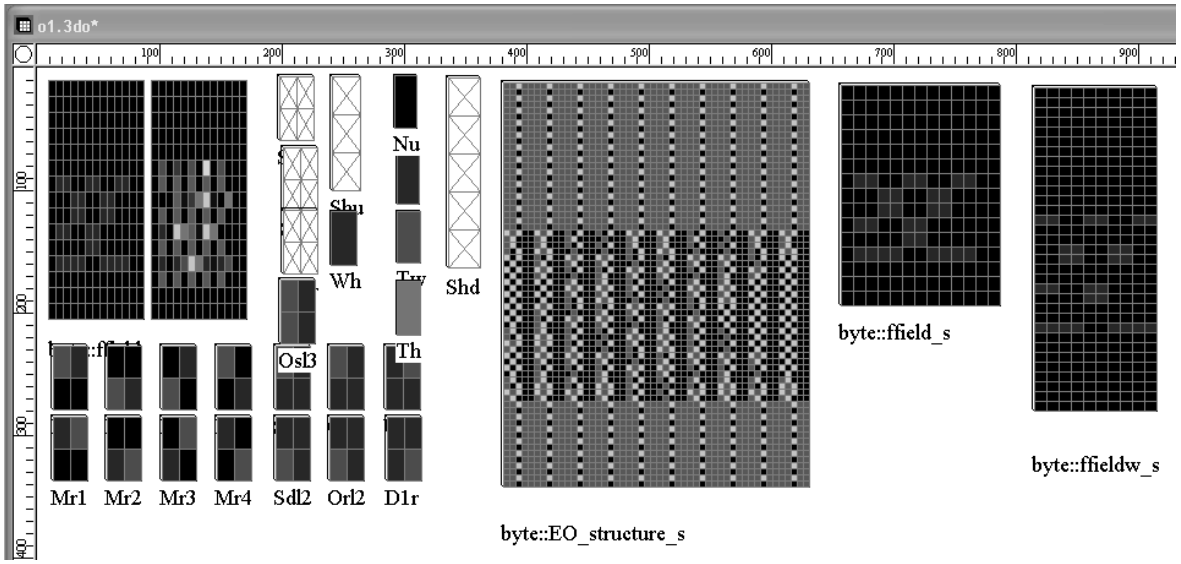

Рис. 5. Полотно с клеточными массивами

Библиотеки внешних модулей. Расширение возможностей системы выполняется с помощью внешних модулей. Они объединены в следующие смысловые группы, называемые библиотеками: 1) модулей поддержки форматов данных; 2) языковых функций; 3) режимов отображения; 4) режимов моделирования.

Библиотека модулей поддержки форматов данных позволяет использовать расширяемое множество форматов для представления клеточных объектов. Изначально имеющееся в ней множество включает форматы для представления клеточных массивов с определенным типом клеток (булева, символьная, целая беззнаковая и знаковая 8-, 16-, 32-битная, с плавающей запятой одинарной или удвоенной точности, строковая) и массивов с клетками различных типов (формат по умолчанию). Это, во-первых, позволяет использовать такие типы клеток, которые, с одной стороны, удобны в модели и, с другой стороны, экономны по расходу памяти. Во-вторых, библиотека реализует набор распространенных форматов, например, форматов растровых изображений (*bmp*) и аудиоданных (*wav*). Это позволяет использовать изображения и аудиоданные как входные, промежуточные или результирующие данные для моделирования. Формат объекта явно указывается в имени объекта в виде префикса. Например, имена объектов с клетками типа 8-битное целое начинаются с префикса *byte::* (см. рис. 5). При использовании формата по умолчанию префикс опускается.

Библиотека модулей языковых функций обеспечивает возможность использования функций, написанных на языках C/C++ и собранных во внешних модулях, в моделирующих программах. Набор изначально реализованных модулей этой библиотеки содержит функции по работе с клеточными объектами (создание, удаление, модификация, изменение размеров), диалоговыми окнами графического пользовательского интерфейса (создание диалоговых окон и ввод пользователем информации с их использованием), математическими функциями (*sin*, *cos*, *atan*, *cosh*, *log*, *j0* и т. д.), функциями консольного ввода / вывода (*WriteLn*, *ReadLn*), файлового ввода / вывода (*fopen*, *fgets*, *fread*, *feof*, *fclose*) и прочими функциями (*max*, *min*, *null*, *typeof*, *StringLength*, *Time* и т. д.). Вызов библиотек из моделирующей программы осуществляется с помощью операторов *use* и *import* языка моделирования системы WinALT.

Библиотека модулей видов отображения обеспечивает возможность визуализировать данные разнообразными способами, в том виде, который привычен пользователю или общепринят в интересующей его прикладной области. Каждый модуль этой библиотеки реализует один вид отображения и содержит необходимые для него процедуры отрисовки срезов клеточных массивов или их отдельных клеток. Изначально реализованные виды обеспечивают

показ состояний клеток цветом, числом и в виде стрелки с направлением на одном слое или на всех слоях в развертке трехмерного объекта на полотне.

Библиотека режимов моделирования позволяет менять такие параметры исполнения модели, как синхронность (синхронный, асинхронный, блочно-синхронный режимы), механизм поиска применимых подстановок (сплошной, по зонам активности). С использованием этой библиотеки строится реализация работы с графовыми структурами.

Пользователь, имеющий навыки программирования, может пополнять наборы модулей всех библиотек, создавая новые внешние модули как с использованием функций, доступных в исходных текстах, так и используя библиотеки функций, доступных только в бинарном виде. Это позволяет адаптировать систему к требованиям пользователя, к классам моделей или обеспечить стыковку с пакетом программ, который использует собственные форматы.

Все внешние модули представляют собой *dll*-файлы в Windows или shared object-файлы в LINUX. Каждый внешний модуль содержит функцию инициализации модуля и шлюзовые функции, реализующие возможность вызова интерфейсных функций из WinALT. Состав остальных интерфейсных функций зависит от того, к какой библиотеке относится модуль. Например, модуль реализации формата содержит такие интерфейсные функции, как создание объекта заданных размеров, чтение значения клетки, запись значения клетки. Для упрощения реализации собственного модуля доступны исходные тексты внешних модулей каждой из библиотек.

Конструирование имитационных моделей МЗП алгоритмов в системе WinALT

Имитационная модель оформляется как проект, содержащий набор подокон – полотен главного окна проекта. Каждое отдельное полотно содержит либо графические объекты модели, либо текстовые.

Создание нового проекта содержит следующие шаги: а) запускается система моделирования WinALT; б) выбирается команда меню *File / New*; в) в открывшемся окне *New* выбирается закладка *Project*; г) в тестовое поле *Project Name* вводится имя проекта; д) в текстовом поле *Location* выбирается место для размещения папки проекта; е) нажимается кнопка *OK*. Проект создан. Для создания полотна с клеточными объектами – файла с расширением *.3do*, в рабочей области проекта: а) в окне *New* выбирается закладка *File*; б) на закладке выбирается строка *Cellular Objects*; в) имя полотна вводится в текстовое поле *File name*; г) нажимается кнопка *OK*. При создании полотна с текстом моделирующей программы – файла с расширением *.src*: а) на закладке *File* выбирается строка *Simulated Program*; б) имя полотна вводится в текстовое поле *File name*; в) нажимается кнопка *OK*. Клеточные объекты размещаются на полотнах с расширением *.3do*. Создание клеточного объекта на полотне включает следующие шаги: а) выбирается команда меню *Object / New*; б) в открывшемся диалоге *Create Object* в поле *Name* в первом окне выбирается тип клеточного массива, например, *default*; во второе окно (после двойного двоеточия) вписывается имя клеточного объекта *myobj1*; в) в поле *Size* устанавливаются размеры объекта; г) в поле *Margins* вводится размер клетки в окне *Cell Size* и указывается положение массива на полотне в окнах *Top* и *Left*; д) нажимается кнопка *OK*. В окне *myproj1.3do* появился клеточный объект *default::myobj1*.

Опишем пример проекта WinALT-модели (рис. 6) алгоритма визуальной криптографии Наора и Шамира [8]. Алгоритмы визуальной криптографии применяются для получения набора зашифрованных изображений, которые по отдельности не позволяют восстановить исходное изображение, но, будучи собраны вместе, обеспечивают возможность его получения. Зашифрованные изображения можно, например, напечатать на прозрачные листы и раздать нескольким людям. Никто из этих людей не может восстановить исходное изображение, пользуясь своим прозрачным листом. Но если все они соберутся вместе и сложат стопкой свои прозрачные листы, то будет видно дешифрованное изображение.

В WinALT-модели алгоритма визуальной криптографии Наора и Шамира реализованы процедуры шифрации и дешифрации изображения. Исходное изображение, которое необходимо зашифровать, хранится в клеточном массиве *bmp2::img_secret_input* слева на полотне *secret.3do*. Оно является входными данными для процедуры шифрации. Результатом этой

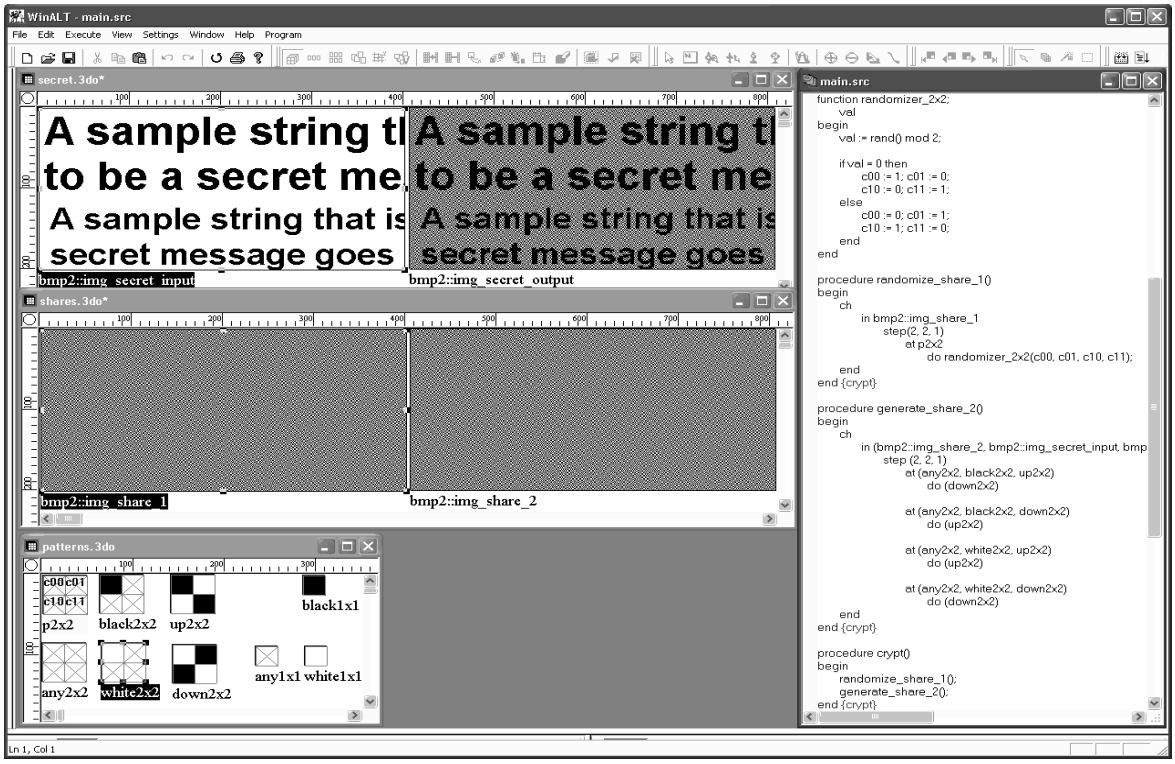

Рис. 6. Пример модели визуальной криптографии

процедуры являются два зашифрованных изображения *bmp2::img_share_1* и *bmp2::img_share_2*, которые показаны на полотне *shares.3do*. Они являются входными данными для процедуры дешифрации. Результатом этой процедуры является расшифрованное изображение *bmp2::img_secret_output*, показанное на полотне *secret.3do* справа. Все шаблоны подстановок модели показаны в полотне *patterns.3do*.

Рассмотрим процедуру шифрации *crypt*, которая вместе со всеми используемыми ею процедурами приведена в подокне *main.src*. Первое зашифрованное изображение *bmp2::img_share_1* формируется функциональной подстановкой, описанной в процедуре *randomize_share_1*. Функциональная подстановка использует шаблон *p2x2* с четырьмя именованными клетками (*c00*, *c01*, *c10*, *c11*) и подстановочную функцию *randomizer_2x2*. Каждый вызов подстановочной функции заполняет группу *2x2* точек изображения *bmp2::img_share_1*, выбирая случайно одну из двух заданных четырьмя операторами присваивания комбинаций двух черных и двух белых точек. Клетка, соответствующая черной точке, находится в состоянии 0, а клетка, соответствующая белой точке, – в состоянии 1. В результате однократного параллельного исполнения рассмотренной подстановки в синхроблоке *change* (записанном в сокращенной форме как *ch*) происходит установка значений всех клеток массива *bmp2::img_share_1*.

Входными данными для второго зашифрованного изображения (*bmp2::img_share_2*) являются изображения *bmp2::img_secret_input* и *bmp2::img_share_1*. Оно формируется четырьмя векторными подстановками, описанными в процедуре *generate_share_2*. Имена клеточных массивов, хранящих входные данные и результат, перечислены в операторе *in*. Шаблоны задают условия применимости каждой из подстановок. Для применимости векторной подстановки требуется выполнение условий, заданных каждым из ее шаблонов. Шаблон *any2x2* присутствует во всех четырех подстановках и описывает всегда истинное условие: подстановка применима при любых значениях клеток массива *bmp2::img_share_2*. Шаблон *black2x2* описывает условие, истинное, когда обрабатываемая точка исходного изображения черная, а *white2x2* – когда она белая. Первые две подстановки обрабатывают случай, когда точка ис-

ходного изображения – черная, вторые две – когда она белая. Для белой точки в исходном изображении соответствующая группа из *2x2* клеток первого зашифрованного изображения копируется во второе зашифрованное изображение. Для черной точки в исходном изображении белые клетки первого зашифрованного изображения заменяются на черные, а черные – на белые во втором зашифрованном изображении.

Входными данными для процедуры дешифрации *decrypt* являются изображения *bmp2::img_share_1* и *bmp2::img_share_2*. Для каждой точки изображения *bmp2::img_share_1* эта процедура выполняет операцию конъюнкции с точкой с теми же самыми координатами из изображения *bmp2::img_share_2*. Результаты конъюнкции записываются в точки изображения *bmp2::img_secret_output*.

В [9] приводится описание примера модели FHP газа на обычной и гексагональной решетке. Рассматривается построение модели, использующей функциональные подстановки, в графической оболочке системы WinALT и ее исполнение на кластере. В [10] даны примеры моделей сетей Петри. Для их построения использованы векторные подстановки и виртуальные массивы. Некоторые из них выполняются в асинхронном режиме моделирования. В [11] описывается пример модели 8-битного пирамидального сумматора, в которой использованы все виды подстановок (скалярные, векторные, функциональные). Там же приведено описание модели двухслойной оптоэлектронной матрицы, реализующей этот сумматор. В [12] рассматриваются модели конфигурируемых многослойных конвейерных структур с электрооптической реализацией. Описание модели устройства ассоциативной обработки данных дано в [13]. Там же описан пример модели алгебраического фрактала Julia.

Заключение

Система WinALT использована для построения широкого спектра моделей с МЗП, часть которых представлена на сайте системы (http: // winalt.sscc.ru). Опыт эксплуатации показал адекватность системы проблемной области и ее удобство.

С помощью системы WinALT заложены основы клеточной технологии построения 3D-вычислительных структур, ориентированных на электрооптическую реализацию [14].

Система WinALT с документацией доступна на сайте для открытого использования.

Список литературы

1. *Тоффоли Т., Марголус Н.* Машины клеточных автоматов. М.: Мир, 1991. 280 с.

2. *Ilachinski A.* Cellular Automata. A Discrete Universe. World Scientific. Singapore, 2001. 808 p.

3. *Pogudin Yu.* Simulating of Fine-Grained Parallel Algorithm with ALT (Animated Language Tools) System // Proc. of the First Intern. Workshop on Distributed Interactive, Simulation and Time Applications / Ed. by A. Boukerche, S. K. Das, V. Malyshkin. Eilat, Israel, 1997. P. 22–27.

4. *Achasova S. M., Bandman O.L., Markova V.P., Piskunov S.V.* Parallel Substitution Algorithm. Theory and Application. World Scientific. Singapore, 1994. 220 p.

5. *Марков А. А.* Теория Алгорифмов // Тр. Мат. ин-та им. Стеклова. М., 1954. Т. 42. 377 с.

6. *Von Neumann J.* Theory of self-reproducing automata / Ed. by A. W. Burks. Urbana; L.: University of Illinois Press, 1966. 324 p.

7. *Ачасова С. М., Бандман О. Л.* Корректность параллельных вычислительных процессов. Новосибирск: Наука, 1990. 252 с.

8. *Naor M., Shamir A.* Visual Cryptography // Proc. of EUROCRYPT'94, Lecture Notes in Computer Science. Heidelberg: Springer-Verlag, 1995. Vol. 950. P. 1–12.

9. *Ostapkevich M. B., Piskunov S. V.* Imitational Simulation of Fine-grain Algorithms and Structures // Bull. Nov. Comp. Center. Ser. Comp. Sci. 2002. Vol. 17. P. 89–103.

10. *Piskunov S. V., Ostapkevich M. B.* The Construction and Simulation of Petri Nets in WinALT // Bull. Nov. Comp. Center, Ser. Comp. Sci. 2012. Vol. 32. (will be published)

11. *Kostsov E. G., Piskunov S. V., Ostapkevich M. B.* 3D ICs with Optical Interconnections // Optical Communication. Rijeka: InTech, 2012.

12. *Kostsov E. G., Piskunov S. V., Umrikhina E. V.* Configurable Multilayer Pipeline Electro-optical Structures // Optoelectronics Instrumentation and Data Processing C/C of Avtometriia. Allerton Press, 2005. Vol. 6. P. 62–71.

13. *Ostapkevich M., Piskunov S.* The Construction of Simulation Models of Algorithms and Structures with Fine-Grain Parallelism in WinALT // Lecture Notes in Computer Science. Heidelberg: Springer-Verlag, 2011. Vol. 6873. P. 192–203.

17. *Косцов Э. Г., Пискунов С. В.* Трехмерные интегральные схемы с оптическими межсоединениями // 3D-лазерные информационные технологии / Под ред. П. Е. Твердохлеба. Новосибирск: Офсет, 2003. С. 168–242.

Материал поступил в редколлегию 13.07.2012

M. B. Ostapkevich, S. V. Piskunov

WinALT SIMULATING SYSTEM FOR ALGORITHMS WITH FINE-GRAIN PARALLELISM

A description of a WinALT simulating system of algorithms with fine-grain parallelism is considered in the article. Its main user properties, architecture and the extensibility of its functionality are presented within the problem domain of algorithms with fine-grain parallelism. The capabilities of the system to adapt to new kinds of algorithms with fine-grain parallelism are demonstrated.

Keywords: fine-grain parallelism, simulation system, simulation language, parallel substitution algorithm, open architecture.