УДК 681.03

ВИНОГРАДОВ Ю.Н.

СПОСОБ ФОРМАЛЬНОГО ОПИСАНИЯ ФУНКЦИОНИРОВАНИЯ РЕКОНФИГУРИРУЕМЫХ ВЫЧИСЛИТЕЛЬНЫХ УСТРОЙСТВ

В статье предложена формализованная модель построения и функционирования реконфигурируемых вычислительных устройств, которая в полной мере соответствует их структурной и алгоритмической организации. Исследована взаимная согласованность и корректность всех введенных в модели понятий. Предложенная модель может быть основой для анализа и синтеза реконфигурируемых вычислительных устройств.

The formalized model of construction and functioning of configurable computing devices is offered in the article, which to a full degree corresponds their structural and algorithmic organization. Mutual coordination and correctness of all of the concepts entered in models is investigational. The offered model can be basis for an analysis and synthesis of reconfigurability computing devices.

1. Введение

Реконфигурируемость структуры вычислительного устройства является естественной потребностью при создании сложных систем, которые должны обладать высокой надежностью, гибкостью и адаптацией к решаемым задачам.

В настоящее время превалирует тенденция выполнения научных исследований и практических разработок в области реконфигурируемых вычислительных устройств (РВУ) с перспективной элементной базой (в частности, ПЛИС), которые призваны удовлетворить требования, возникающие при создании высокопроизводительных устройств обработки данных, цифровой обработки сигналов, поддержки телекоммуникаций, сопряжением комплексов и систем и т.д..

Исследования комплекса прикладных задач анализа, синтеза и оптимизации системного и логического проектирования реконфигурируемых устройств [1 - 3] показали, что сдерживающим фактором построения РВУ является отсутствие системы формальных математических моделей, которые учитывали бы характеристики особенности их функционирования, определяемые особенностями новой элементной базой (в частности, ПЛИС).

Таким образом, задача построения формализованной системы согласованных математических понятий, адекватно описывающих характерные особенности функционирования РВУ, является актуальной и важной для современного этапа развития теории проектирования вычислительных устройств.

2. Анализ существующих моделей функционирования РВУ

Принцип реконфигурируемости, положенный в основу проектирования РВУ, прошел определенную эволюцию и переосмысление, что, в первую очередь, связано с прогресом элементной базы. Слово реконфигурируемый трактуется двояко: есть функциональные элементы структуры, которые объединяются тем или иным способом, при помощи каналов связи, для решения конкретной задачи, либо в системе для каждой конкретной задачи используется новая структура с новыми функциональными элементами. Такая форма реконфигурации выражается в том, что для реализации различных процессов возникают структуры в произвольном порядке.

В [1] рассмотрена классификация и проанализированы основные особенности РВУ, которые в наибольшей мере определяют способ их функционирования:

- отсутствие заранее известной структуры решения задачи;
- отсутствие единого событийного времени;
- внутренний и внешний недетерминизм при организации вычислительного процесса;
- наличие конфликтов и совместно используемых ресурсов;
- независимость структурной организации от времени;
- наличие двух видов рекофигурирования.

Ни одна из известных моделей функционирования РВС [2] [3] и изложенная в [4] не учитывает все эти особенности РВС в полной мере. Упомянутые модели создавались с целью спецификации структуры и исследования вычислительных процессов при решении задач в ней. Основные недостатки указанных моделей – это отсутствие связи с количественным и алгоритмическим анализом вычислительных устройств. Модели частично описывают влияние функциональных элементов структуры на динамику взаимодействия структурных элементов системы. Они не содержит множественного времени как количественной сущности, что затрудняет описание процессов временной реконфигурации. В известных моделях отсутствует иерархичность и структуризация функциональной организации вычислительной системы с точки зрения ее реконфигурируемости.

Таким образом, проанализированные выше модели лишь частично формализуют процесс описания РВУ и требуют дополнительных инструментов для логического и системного проектирования вычислительных устройств.

Целью данной статьи является построение математической модели динамики функционирования реконфигурируемых вычислительных устройств, которая достигается путем разработки согласованной системы математических понятий, адекватно описывающих их характерные особенности.

3. Разработка модели функционирования реконфигурируемых вычислительных устройств.

Основу модели функционирования реконфигурируемых вычислительных устройств, которые учитывают влияние функциональных элементов структуры на динамику взаимодействия структурных элементов, составляют три понятия:

1. поведение – модель динамики организации реконфигурируемых вычислительных процессов;
2. среда – модель аппаратных средств с реконфигурируемой структурой;
3. реализация – исчисление, определяющее интерпретацию поведения в конкретной среде при заданных исходных данных.

3.1. Основные понятия и определения предлагаемой модели

В составе РВУ будем выделять три основные части:

- физическую среду – аппаратные средства, определяемые составом функциональных элементов и коммутирующей сетью;
- логическую среду – управляющие средства, которые поддерживают работоспособность физической среды и реализуют выполнение той или иной задачи и способ реконфигурации;
- поведенческую среду, которая определяет конкретную реализацию вычислительного процесса на РВУ.

В предлагаемой модели под словом реконфигурация понимается децентрализация управления и вычислений в физической и логической средах. Это значит, что в физической среде есть устройства способные функционировать автономно, т.е. решать индивидуальную “перегружаемую” задачу, а логическая среда обеспечивает использование реконфигурируемости физической среды. В качестве таких устройств мы будем иметь ввиду ПЛИС. Такое понимание охватывает широкий класс вычислительных систем, к которым относятся: измерительно-вычислительные системы, бортовые вычислительные комплексы, системы управления экспериментом, технологическими процессами, высокопроизводительные вычислительные комплексы на базе спецпроцессоров.

Динамику функционирования РВУ определяет взаимодействие процессов реализации конкретной вычислительной подзадачи при ее решении и совместная работа с логической и физической средами. Схематично наше представление о динамике можно охарактеризовать так. Каждый отдельно взятый вычислительный процесс определяет логическую последовательность действий. Эти действия есть либо реализация статического, либо динамическое использование логического ресурса. Использование сопровождается передачей параметров. Так выглядит выполнение совокупности процессов в РВУ с точки зрения глобальной организации вычислительного процесса.

Результатом выполнения действия по отношению к логической среде является формирование сообщения определенного типа и имени процесса, которому необходимо передать управление и само сообщение. Под сообщением мы будем понимать совокупность параметров обращения. Тип сообщения – это класс эквивалентности на множестве допустимых сообщений.

Передача управления может происходить по инициативе самого процесса, а может произойти и в результате воздействия на исполнителя извне (прерывания), например со стороны другого исполнителя под влиянием выполняемого на нем процесса. Одному исполнителю может быть сопоставлено несколько процессов. Их количество ограничено физическими характеристиками исполнителя.

Исполнитель – это модель физической среды. Исполнители бывают динамические и статические. Статический исполнитель может выполнять только фиксированные задачи из заранее определенного конкретного набора. Динамический может менять набор задач. Исполнители, объединенные каналами связи, могут обмениваться сообщениями и образуют реконфигурируемый исполнитель. Канал передает эти сообщения без каких либо изменений от одного исполнителя к другому, внося определенную временную задержку. Величина этой задержки и объем единовременно передаваемой информации есть постоянные характеристики данного канала.

Каждый реконфигурируемый исполнитель связан с определенным набором исполнителей. Этот набор ограничен и не изменяется в процессе реализации задачи. Последовательность логических действий по реализации процесса также фиксирована.

Так кратко, на содержательном уровне, можно описать наше представление о существе рассматриваемого явления и те концепции, на которых создается модель. Подчеркнем, что мы задаем модель не какой то конкретной вычислительной системы, а модель класса реконфигурируемых вычислительных устройств, характеристики которого были сформулированы в [1].

3.2 Математическая модель функционирования реконфигурируемых вычислительных устройств

Конкретизируем понятие поведения, т.е. детально опишем модель динамики организации реконфигурирования вычислительного процесса.

Поведение процесса будем описывать в форме тензорного уравнения как умножение со сверткой подмножества области отправления отображения на график этого отображения. Результатом при этом, является подмножество области прибытия отображения. [5]

Использование такого описания поведения процесса позволяет отказаться от меток шагов преобразования, т.к. их роль представляют обозна-

чения подмножеств областей отправления. Более того, в этом случае отпадает необходимость в описании момента передачи управления при описании поведения процесса, т.к. каждое отображение начинает выполняться только тогда, когда в результате выполнения других отображений сформируется подмножество области отправления данного отображения.[6].

Таким образом, поведение процесса можно описать тензорным уравнением преобразования его шагов, что позволяет рассматривать множество взаимодействий реализации алгоритмов конкретной задачи.

Пусть M – множество типов сообщений в РВУ и P – множество наблюдаемых процессов в системе. Множество P состоит из элементов, которые принадлежат либо множеству процессов логической среды P_L, либо множеству подзадач вычислительного алгоритма P_A, причем P = P_L $\bigcup$ P_A и P_L $\bigcap$ P_A = 0.

Выполнение шага процесса P_i определяется следующим образом:

1. инициализируется воздействие, которое заключается в получении сообщения типа a из множества M_I и управления от процесса P_j;
2. выполняется реализация, которая состоит из последовательности внутренних действий $\{q_i\}$, множество которых обозначим через Q, направленных на определение отклика на инициализируемое воздействие.
3. формируется реакция, которая заключается в посылке нового сообщения b из множества M_O.

Шаг процесса S определим как тройку S = (M_I, M_O, P), где M_I - множество исходных сообщений, M_O – множество выходных сообщений (M = M_I $\bigcup$ M_O), P – множество состояний наблюдаемых процессов.

В этом случае, выполнение шага описывается тензорным уравнением

$$S^{M\_O,P_j} = S^{M\_I,P_i} Q^{M\_O,P_j}_{M\_I,P_i} \qquad (1).$$

Важной характеристикой шага является его сложность. Сложность шага – величина, определяющая время выполнения его внутренних действий. Эти действия суть использования статических логических ресурсов, поэтому можно определить тензор W_i^{qi}, который однозначно задает сложность $W_i^{qi} = W_1^{q1} + W_2^{q2} + \ldots = W^q$. Очевидно, что если внутренние действия отсутствуют, то $W_i^{qi} = 0$.

Для оценки времени на передачу сообщений между исполнителями мы введем такую характеристику сообщения, как временная сложность шага. Определим ее с помощью тензорного уравнения $C\_T^b = M\_I_a^i Q_i^{ab}$,

которое реализует график M_O^b на множестве типов сообщений с областью значений – множество натуральных чисел *i*.

Историей процесса P назовем непустую конечную последовательность его шагов, замкнутую слева, т.е. если $H_P S^P_i = S_1\ S_{2....}\ S_k$ история, то $\forall i : i \le k$ и последовательности $S_1\ S_{2....}\ S_k$ – история. Наше предположение о конечности истории основано на представлении о реализуемости вычислений под надзором наблюдателя, а всякое наблюдение конечно.

Для выполнения процесса P потребуется фиксированное число шагов его истории, которые можно охарактеризовать длиной истории K_P. Истории процессов P_1 и P_2 равны тогда и только тогда, когда равны их длины $K_{P1} = K_{P2}$ и $\forall i : 1 \le i \le K_{P1}$, $S_{1i} = S_{2j}$, где S_{ij} –

i-тый шаг j-го процесса.

Два шага S_1 и S_2 равны тогда и только тогда, когда совпадают типы сообщений и типы процессов. Равенство сложностей шагов в этом случае не требуется.

Подисторией процесса P называется любой постфикс этой истории. Цепь – это любое подслово слова H_P. Поведением процесса P назовем множество всевозможных историй этого процесса, которое мы будем обозначать BH_P.

Проанализируем структуру поведения процесса BH_P с целью его описания. Для этого историю процесса $H_P S^P_i = S_1\ S_{2...}\ S_k$, правая часть которой представляет цепь $k = S_1\ S_{2...}\ S_k$ представим в блочном виде: $S_1\ S_{2...}\ S_l = l_1$, $S_{l+1}\ S_{l+2...}\ S_k = l_2$.

Введем две операции: следования $\rightarrow$ и альтернативы $\neg$. Цепь $k = l_1 \rightarrow l_2$ такова, что

$k = l_1 \neg l_2$, а l_1 – префикс, l_2 – постфикс k, т.е. операция следования есть конкатенация цепей, рассматриваемых как последовательность символов в алфавите S^P_i. Нетрудно видеть, что операция $\rightarrow$ ассоциативна, но не коммутативна.

Если поведение процесса P задается выражением $H_{P1} \neg H_{P2}$, то сам процесс, в зависимости от тех или иных условий, может быть реализован либо историей H_{P1} либо H_{P2}. Правило выбора способа реализации операция $\neg$ не определяет. Основные свойства операции $\neg$ следующие:

1. $l_1 \neg l_2 = l_2 \neg l_1$;
2. $(l_1 \neg l_2) \neg l_3 = l_1 \neg (l_2 \neg l_3)$;
3. $l_1 \rightarrow (l_2 \neg l_3) = l_1 \rightarrow l_2 \neg l_1 \rightarrow l_3$.

Таким образом, операция $\neg$ коммутативна, ассоциативна и дистрибутивна справа относительно операции $\rightarrow$.

Теорема 1. Любой процесс P, который имеет историю H_P ($P : H_P \in BH_P$) может быть представлен в виде: $H_P = H_{P1} \rightarrow \sim H_P$.

Теорема 2. Пусть BH_P' множество всевозможных историй $H_P' = H_{P0} \to \sim H_P'$, причем $\sim H_P' \in BH_P'$, где BH_P' – множество подисторий историй из BH_P'. Тогда $BH_P' = H_0 \to \Sigma \sim H_{Pi}$, где сумма берется по $\sim H_{Pi} \in \sim BH_P'$.

Обозначим элементы $t^* H_{P0}$ такими, что они не принадлежат $H_{P0} \to l$, где $l \in S^*_P$.

Теорема 3. Для любого фиксированного $H_{P0} \in BH_P$ представление $t^* H_{P0} \in BH_P$ в виде $H_0 \to \Sigma \sim H_{Pi}$ единственно с точностью до ассоциативной перестановки слагаемых.

Следствие. Префикс H_P' любой истории из BH_P задает на истории класс эквивалентности $t^* H_P' = \{H_P \mid H_P = H_P' \to \sim H_P\}$. Причем для $\forall\ H_P'$, H_P'' из BH_P таких, что $H_P' \neq H_P''$ следует, что $t^* H_P' \cap t^* H_P'' = \varnothing$.

Теорема 4. Для любого множества BH_P существует тензор преобразования (возможно инвариант), который устанавливает взаимнооднозначное соответствие между множествами путей историй процесса и множества историй из BH_P:

$$S_{P_i}^{k} T_k^{P_i} = BH_{P_i}$$

Одной из основных особенностей поведения реконфигурируемых вычислительных процессов является недетерминизм. Можно определить два его вида: внутренний и внешний. Будем говорить, что выражение $S_i + S_j$ описывает внешний недетерминизм, если множество M_I из процесса S_i не равно соответствующему множеству исходных сообщений из процесса S_j и внутренний, если они равны.

Способ реализации задачи определяется множеством поведений процессов $BH_p = \bigcup_{P_i \in p} bhp_i$ образующих ее решения. Множество шагов $S^P{}_i$ в поведении BH_P частично упорядочено. Это отношение отражает взаимодействие между процессами и порядок следования шагов в истории каждого процесса. Проанализируем это отношение более подробно, т.к. оно описывает причинно-следственные связи на множестве действий при решении задачи и во многом определяет семантику реконфигурации в РВУ. По определению, из цепочки S_i следует цепочка S_j тогда и только тогда когда

$\exists\ H_P \in BH_P : (S_i \neg S_j) \in H_P$ и , $S_j \in S^P{}_k$; где S_i – причина, а S_j – следствие. Это отношение обозначим R □ (P) = { (S_i, S_j) | из S_i следует S_j }. Как видно, оно означает причинно-следственные связи на $S^P{}_k$, где $k \in P$.

Это отношение можно распространить на случай событий. Выполнение шага процесса P согласно (1) состоит в инициализации воздействия путем получения сообщения из M_I и реакции, которая заключается в формировании нового сообщения из M_O. Если обозначить событие через *e,* с надлежащим индексом, то получим: из цепочки e_i следует цепочка e_j тогда и только тогда когда

$\exists$ S_k : (e_i = M_I(S^P_k) & e_j = M_O(S^P_k) $\vee$ $\exists$ S_l : $e_j \in S_k$ & $e_j \in S_l$ & (S_k, S_l) $\in$ R □ (P))

Определим отношение

R~(P) = {(S_i, S_j) | $\exists$ Hp_1, $\exists$ Hp_2 : $S_i \in Hp_1$ & $S_j \in Hp_2$ & M_O(S_i^P) = M_I (S_j^P)},

т.е. реакция на шаге S_i в поведении процесса P_1 есть обращение к шагу S_j в поведении процесса P_2. Если (S_i, S_j) $\in$ R~(P), то этот факт будем записывать как тензорное уравнение $S_j = S_i R^i_j$. Отношение R~(P) несимметрично, иррефлексивно.

Обозначим R > (P) = R~(P) $\bigcup$ { $\bigcup_{P_i=P} R$ □ (P) }. Не трудно показать, что эти отношение также несимметрично и иррефлексивно.

По аналогии с ранее сделанным, распространим отношение R~(P) на случай событий:

из цепочки e_i следует цепочка e_j тогда и только тогда когда

$\exists$ S_k $\exists$ S_l : (S_k S_m) $\in$ R~(P) & $e_i \in S_k$ & $e_j \in S_m$

Это отношение обозначим Re~(P).

Историей выполнения вычисления называется тройка (H^i_P, $M\_I_i{}^j$, R~P), т.е. это совокупность историй процессов H^i_P , удовлетворяющих отношению R~(P); $M\_I_i{}^j$ – сообщение типа a_i из множества исходных сообщений процесса P^j , которое задает первый шаг в том процессе, с которого начинаются вычисления.

Рассмотрим различия в описании поведения процессов логической среды и описания вычислений, проводящихся в ней.

1. В поведении прикладных процессов вместо некоторого шага может быть указан тензор процесса, который свернут до шага. Это означает, что процесс начиная с этого места ведет себя так как свернутый процесс.

2. В уравнениях выполнения процессов из логической среды, последовательности шагов вычислений представляются переменными с именами специального фиксированного вида, которые образуют множество Dm с элементами dmi. Реакция в которой указано dmi, означает передачу управления прикладному процессу, сопоставленному этому имени. Соответствие между Dm и процессами из P устанавливается уравнением SHD = P^i Dm_i. Здесь предполагается, что SHD не зависит от времени. Однако, наделяя SHD различными свойствами, например, вводя индексы времени, можно описывать динамические процессы развивающиеся во времени.

3. Считаем, что множества Dm и P_L фиксированы для задачи. В этом случае, исходя из ограничения Dm, следует ограничение на множество допустимых к реализации вычислений, а именно | Dm | $\geq$ | P |.

4. В логической среде есть процесс с именем Stop, обращение к которому завершает процесс. В реакциях шагов может присутствовать предо-

пределенная переменная back, которую можно представить, как вершину стека. Для работы с ней есть две функции put – поместить в стек и get – выдать значения.

Проанализируем виды реконфигурирования и их семантику. Есть два вида реконфигурации: функциональная и полная. Первая соответствует случаю, когда функциональные элементы, объединяются при помощи коммутационной сети для решения той или иной подзадачи. В этом случае, выполнение вычислений происходит одним исполнителем, т.е. процессы разделяют ресурсы одной и той же логической и физической сред. Во втором случае, вычисления выполняются независимо на разных процессах, обмениваясь при необходимости сообщениями.

Введем две операции композиции процессов: $P_1 \sim P_2$ – совместное использование функциональных элементов для решения ряда подзадач; $P_1 \parallel P_2$ – полная реконфигурация. Существует несколько подходов в описании способа реконфигурирования. Например, один из самых распространенных это использования описания типа семантики применяемой в языках программирования, таких как CSP [7]. Другой, это семантики описания операторов работы с разделяемыми переменными, который чаще всего используется в теории сетей [8].

Для описания процессов при объединении функциональных элементов с помощью коммутирующей сети будем использовать семантику чередования. Это означает, что события которые задаются процессами HP_1 и HP_2, образуют цепочку, в которой они не имеют предпочтения в порядке следования между собой. Ограничение на чередование определяет некоторое отношение независимости, задаваемое на S^P. Такая семантика учитывает единство временной оси исполнителя. Это выражается в том, что все события, происходящие на данном исполнителе, собираются в цепочки.

Семантику независимых реконфигурируемых структур, организующих выполнение вычислительных процессов, будем описывать отношениями частичного порядка на множестве их событий. Запись $P_1 \parallel P_2$ означает, что цепочка событий, определяемая историей HP_1 , появляется между точками взаимодействия вместо и независимо от цепочки, определяемой историей HP_2. Анализ взаимодействия этих видов реконфигурации и их эквивалентность будет проведен ниже.

Конкретизируем понятие физической среды, которая в предлагаемой модели реконфигурируемых устройств представляет исполнитель. Прежде всего, рассмотрим реконфигурируемые устройства у которых функциональные элементы структуры объединяются тем или иным способом при помощи коммуникационных каналов. Для этого введем понятие пространственно реконфигурируемый исполнитель.

Пространственно реконфигурируемый исполнитель SE – это математическая структура:

SE=<Pa, Tm_n^e, $M^P{}_i$, C_S , A_{Map}^{Tm} > (2)

у которой Pa – атомарный процесс; Tm_n^e – тензор модельного времени; $M^P{}_i$ – память исполнителя; C_S – пространственно реконфигурируемый вычислительный узел; A_{Map}^{Tm} – арбитр. Определим строго эти понятия.

В множестве наблюдаемых процессов P можно выделить подмножество атомарных процессов Pa. Атомарным называется процесс, который для конкретного устройства имеет минимальное количество шагов истории процесса. В этом случае атомарный процесс задает частичное отображение множества исходных сообщений M_I в множество выходных сообщений M_O и образует множество типов атомарных сообщений. M_A.

Основные свойства атомарных процессов таковы:

1) $\forall a_i : \exists M\_I : \exists! M\_O : a_i \in M\_A$,

$\forall b_i : \exists M\_O : \exists! M\_I : b_i \in M\_A$

т.е. для любого $a_i \in M\_A$ определено множество исходных сообщений множество выходных сообщений. Между M_I и M_O тензор $S^{M\_I,Pa}$ задает однозначное соответствие. Поэтому недетерминизма на уровне исполнителя нет.

2) Выполнение атомарного процесса не прерывается, т.е. начавшись, атомарный процесс будет закончен за определенное фиксированное, конечное время без останова.

3) После выполнения атомарного процесса управление всегда возвращается тому процессу, который инициировал его выполнение, если ему на вход поступило сообщение допустимого типа.

4) У атомарного процесса моменты передачи сообщения результата и моменты возврата управления совпадают; если управление передано, то сообщение доступно.

Каждый атомарный процесс характеризуется сложностью, которая определяет время выполнения его внутренних действий в тактах. Суть этих действий, заключается в использовании статических физических ресурсов сети. Поэтому можно определить тензор W_i^a который однозначно задает время выполнения атомарного процесса множеством натуральных единиц времени (тактов) по часам исполнителя. В этом случае можно отметить, что время выполнения любого атомарного процесса конечно, постоянно и равно $P_a W_i^a$.

Модельное время есть тензор $\mathrm{Tm}_{\mathrm{n}}^{\mathrm{e}}$, который сопоставляет событию e количество тактов прошедших к моменту его возникновения на данном исполнителе. Момент наступления события считается с момента начала выполнения первого атомарного процесса. Область отправления отображения модельного времени определена на множестве всех событий e_i в процессах P_a, приписанных данному исполнителю и представляет собой, в общем случае, множество натуральных чисел. Постулируем свойства тензора Tm.

Постулат 1. $\forall e_i, e_j : e_i, e_j \in BHp_m \,\&\, e_i \sim\!\!\rightarrow e_j$, т.е. в одном процессе причина всегда наступает раньше следствия $\mathrm{Tm_i} < \mathrm{Tm_j}$.

Постулат 2.

$$\forall e_i, e_j : e_i \in BHp_k \,\&\, e_j \in BHp_m \,\&\, e_i \sim\!\!\rightarrow e_j \,\&\, p_m \sim p_k,$$

т.е. если причина и следствие принадлежат разным $\mathrm{Tm_i} < \mathrm{Tm_j}$ процессам и эти процессы выполняются на одном и том же исполнителе (устройстве), то по часам этого исполнителя причина наступит всегда раньше следствия.

Из приведенных выше постулатов можно сформулировать следующие следствия:

Следствие 1. Каждый процесс имеет длительность, т.е. между двумя последовательными событиями одного процесса (между причиной и следствием) всегда происходит хотя бы один такт времени.

Следствие 2. $\forall e_i, e_j : e_i \rightarrow e_j$, и $\mathrm{Tm_i} < \mathrm{Tm_j}$ т.е. любой процесс выполняется последовательно.

Следствие 3. $\forall \mathrm{e_j} : \mathrm{e_0} \rightarrow \mathrm{e_j}$ и $\mathrm{Tm_0} < \mathrm{Tm_j}$, т.е. обращение к процессу всегда наступает раньше (по часам данного исполнителя), чем возникает какое либо событие в этом процессе.

Следствие 4. Если

$$\forall \mathrm{e_i}, \mathrm{e_j} : (\mathrm{e_i}, \mathrm{e_j}) \in \mathrm{R} \sim \mathrm{P} \,\&\, \mathrm{e_i} \in \mathrm{BHp_k} \,\&\, \mathrm{e_j} \in \mathrm{BHp_m} \,\&\, \mathrm{P_k} \sim \mathrm{P_m},$$

то из этого следует $\mathrm{Tm_i} < \mathrm{Tm_j}$, причем R~P транзитивное замыкание. Данное следствие означает, что если два события связаны причинно – следственной связью, принадлежат разным процессам, которые выполняются на одном и том же исполнителе, то причина по часам этого исполнителя наступает всегда раньше следствия.

Теорема 5. Тензор $\mathrm{Tm}_{\mathrm{n}}^{\mathrm{e}}$ не нарушает отношения причинно – следственного порядка цепочки процессов, которые выполняются одним исполнителем.

Обозначим через t переменную на множестве вещественных чисел R, значение которой равно величине астрономического времени, код которо-

го можно определить с помощью астрономических наблюдений. Такое задание времени никоим образом не зависит от свойств вычислительного устройства, но это время важно для организации конкретных вычислений.

Рассмотрим определение множества отправлений тензора Tm^{e}_{n}, а именно, обозначим через Tm_t количество тактов наступивших к моменту времени t. При этом постулируем следующие свойства:

Постулат 3. $\forall t, \exists P, \forall \Delta t : \rho, t, \ \Delta \in R \ \& \ 0 < \Delta < \rho$

В этом случае Tm_t(t + Δt) – Tm_t = 1& Tm_t(t + Δt) – Tm_t = 0. т.е. часы идут с минимально различимым тактом ρ. На практике точность измерения ρ флюктуирует. Поэтому для практического применения Постулат 3 удобнее представить в следующем виде:

Постулат 3_1. $\forall t, \forall \Delta t \forall \rho, \exists \zeta : 0 < \Delta < (\rho - \zeta) \ \& \ \rho - \zeta < \rho < \rho + \zeta$

В этом случае Tm_t(t + Δt) – Tm_t = 0& Tm_t(t + ρ) – Tm_t = 1.

При таком подходе можно однозначно обеспечить показания часов в разных экспериментах только лишь в рамках определенного интервала наблюдений δt, такого, что $|[\frac{\delta t}{\rho - \zeta}|]$ = $|[\frac{\delta t}{\rho + \zeta}|]$, где |[…]| - целая часть выражения. Отсюда, зная ρ и ζ можно вычислить δt, либо зная заранее δt, определить длительность шага моделирования.

Для хранения и правильного функционирования процесса требуется определенный объем памяти в котором хранятся параметры реконфигурации. Определим тензор M^{P}_{i}, который в качестве области отправления отображения на график этого отображения имеет множество процессов и множество натуральных чисел соответственно. Такое задание тензора M^{P}_{i} позволяет определить необходимый объем памяти для хранения и выполнения процесса. В тоже время сам исполнитель может иметь определенное (возможно большее) количество единиц памяти N.

На данном конкретном исполнителе может выполняться только такое множество процессов, у которых наличная память N будет больше чем необходимая.

Понятие вычислительного узла является ключевым для построения модели исполнителя. Пространственно реконфигурируемый вычислительный узел C_S это математическая структура C_S = <IN, ENT^{n}_{IN}, OUT, EXT^{OUT}_{k}>, где IN – множество полюсов, которые мы будем называть входами, тензор ENT^{n}_{IN} – определяет назначение входов; тем самым, каждому входу ставится во взаимно – однозначное соответствующие полюсы n вычислительного узла, OUT – множество полюсов,

которые мы будем называть выходами, тензор EXT_k^{OUT} – определяет назначение выходов.

Полюса $out_i \in OUT$ обеспечивают передачу воздействий на процессы, размещенные на других исполнителях. Каждому out_i по определенному правилу или закону сопоставляется атомарный процесс P_out_i. Время срабатывания полюса out_i определяется тензором $Tm_i^{P-out_i}$. Также с каждым полюсом связана функция Map(n):N $\rightarrow \{0,1\}$; Map(n) = 1, если $n \in [Tm\_t_0, Tm\_t_0 + P_{P\_out_i} \bullet Tm_i^{P-out_i}]$, в противном случае – 0. Здесь Tm_t_0 задает воздействие на P_out_i. Назовем эту функцию характеристической.

Один out_i может быть соединен с несколькими входами in_j своего исполнителя или других исполнителей. Всем этим in_j одновременно будет передано одно и тоже сообщение. Время передачи того или иного сообщения определяется характеристикой соответствующей связи. (Способ задания характеристики будет рассмотрен отдельно). Объем и тип передаваемого за одно обращение к out_i сообщения определяет атомарный процесс P_out_i.

Каждому входу in_j можно сопоставить (способ задания сопоставления будет рассмотрен отдельно) только один выход out_i своего либо другого исполнителя и один процесс из множества типов сообщений M, управление которому будет передано, если данный вход возбужден и выиграл арбитраж. Вход возбужден, если Map(n)=1 у выхода, связанным с этим входом.

Тензор A_{Map}^{Tm} описывает процедуру арбитража входов пространственно реконфигурируемого исполнителя. Область отправления отображения это множество значений тензор Tm, а множество характеристических функций определяет график этого отображения. Таким образом, в зависимости от состояния выходов, арбитраж позволяет однозначно передать управление на вход исполнителя.

Более сложная модель временного реконфигурируемого исполнителя. Конкретизируем эту модель.

Временный реконфигурируемый исполнитель DE это математическая структура:

$$DE = < \{SE_i\}, C\_D, M\_S_M^E > \qquad (3)$$

где $\{SE_i\}$ – множество пространственно реконфигурируемых исполнителей, C_D – распределенный вычислительный узел, M_S – поток сообщений.

Понятие распределенного вычислительного узла C_D определим как гиперграф, множество вершин которого образуют полюса C_S, входящие в состав соответствующего CE_i, т.е. это математическая структура: $C\_D = <\{C\_S_i\}, E, ENT^n_{IN}, EXT^{OUT}_k>$,

где $\{C\_S_i\}$ – множество реконфигурируемых вычислительных узлов, которое обозначим через V. По сути, множество вычислительных узлов, определяется следующим образом:

$V = \cup\{OUT_i \cup IN_i \mid OUT_i \in C\_S_i \,\&\, OUT_i \in C\_S_i\}$; $C\_S_i \in C\_D$;

E – множество дуг, которые мы будем обозначать ARC, и которое определятся, как

$E = \{ARC_j \mid ARC_j \in V \,\&\, \forall j \exists! OUT \in ARC_j\}$, т.е. дугу в гиперграфе образует множество полюсов, среди которых есть только один полюс типа выход;

ENT^n_{IN} – тензор разметки входов C_D : IN – вход в узел C_D, если

$\exists C\_S_j : C\_S_j \in C\_D \Rightarrow \forall ARC : IN \notin ARC \,\&\, IN \in IN_j \,\&\, IN_j \in C\_S_j$; т.е. это один из входов C_S, не использованных в дугах C_D;

EXT^{OUT}_k тензор разметки выходов C_D : OUT, если

$\exists C\_S_j : OUT_j \in OUT_j \,\&\, OUT_j \in C\_S_j \Rightarrow \forall ARC : OUT_j \notin ARC$, т.е. это один из выходов C_S, не использованных в дугах C_D;

Поток M_S формируется множеством дуг E по которым за время tt блоками m_v передается множество M сообщений. Каждому конкретному сообщению m_v_i соответствует время передачи tt_i, измеряемое в тактах исполнителя. Пару вида $< m\_v_i, tt_i, >$ будем называть пропускной способностью канала.

Модели пространственного и временного реконфигурируемых исполнителей позволяют при помощи конструктивных правил описать любую структуру вычислительного устройства. Конкретизируем и определим алгебру носителей для построения модели структуры такого устройства.

Обозначим через C множество вычислительных узлов: $C = CS \cap CD$, где CS(m,n) – множество пространственно реконфигурируемых вычислительных узлов CS, имеющих не более n входов и m выходов; CD – множество временных реконфигурируемых узлов. Для простоты изложения, без потери общности, введем следующие обозначения: C_S(i,j) – пространственно реконфигурируемый вычислительный узел с i входами и j выходами; Plug – пространственно реконфигурируемый вычислительный узел вида C_S(1,1). Будем считать, что $CD_1 = CD_2$, если:

1) $\forall C\_S_{1i} : C\_S_{1i} \in CD_1 \Rightarrow \exists C\_S_{2j} : C\_S_{2j} \in CD_2 \,\&\, C\_S_{1i} = C\_S_{2j}$;

2) $E_1 = E_2$; $ENT_{IN}^{n1} = ENT_{IN}^{n2}$; $EXT_{k1}^{OUT} = EXT_{k2}^{OUT}$;

т.е. вычислительные узлы равны с точностью до изоморфизма.

Определим на множестве вычислительных узлов: С операции объединения, слияния входов, композиции и замыкания. Везде далее считаем заданными вычислительные узлы $C\_D_1 = <V_1, E_1, ENT_{IN}^{n1}, EXT_{k1}^{OUT}>$, $C\_D_2 = <V_2, E_2, ENT_{IN}^{n2}, EXT_{k2}^{OUT}>$, $C\_D_1, C\_D_2 \in C$.

Объединением вычислительных узлов $C\_D_1 + C\_D_2$ будем называть узел $C\_D_3 = <V_3, E_3, ENT_{IN}^{n3}, EXT_{k3}^{OUT}> \in CD$ такой, что: $V_3 = V_1 \cup V_2$; $E_3 = E_1 \cup E_2$; $n_3 = n_1+n_2$; $k_3 = k_1+ k_2$;

$ENT_{IN}^{n3(i)} = ENT_{IN}^{n1(i)}$, если $1 \le i \le n_1$, и $ENT_{IN}^{n3(i)} = ENT_{IN}^{n2(i-n_1)}$ если $n_1 \le i \le n_1 + n_2$;

$EXT_{k3}^{OUT} = EXT_{k1}^{OUT}$, если $OUT \in V_1$, и $EXT_{k3}^{OUT} = k_1 + EXT_{k2}^{OUT}$, если $OUT \in V_2$.

Рис. 1 иллюстрирует эту операцию графически.

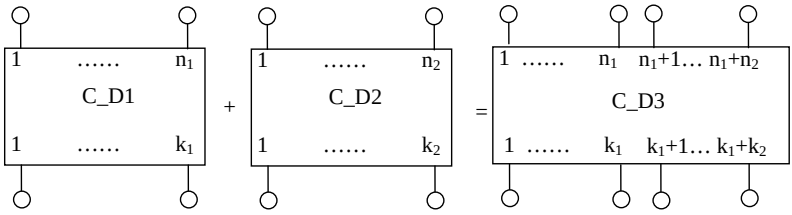

Рис 1.

Слиянием входов вычислительных узлов C_D1 # C_D2 будем называть узел $C\_D_3 = <V_3, E_3, ENT_{IN}^{n3}, EXT_{k3}^{OUT}> \in CD$ такой, что: $V_3 = ENT_{IN}^{n1} \cup \{EXT_{k1}^{OUT} \cup EXT_{k2}^{OUT}\}$; $E_3 = E_1 \cup E_2$; причем $ENT_{IN}^{n2} = ENT_{IN}^{n1}$ и $EXT_{k3}^{OUT} = EXT_{k1}^{OUT}$, если $EXT_{k3}^{OUT} \in EXT_{k1}^{OUT}$, и $EXT_{k3}^{OUT} = EXT_{k2}^{OUT}$, если $EXT_{k3}^{OUT} \in EXT_{k2}^{OUT}$. Эта операция определена для тех вычислительных узлов, у которых количество входов

равно друг другу, т.е. $ENT_{IN}^{n1} = ENT_{IN}^{n2}$. Сущность этой операции состоит в том, что входы C_D_1 отождествляются с входами C_D_2. Тем самым допускается многополюсное возбуждение входов узла C_D_3. Графическое представление этой операции представлено на рис. 2.

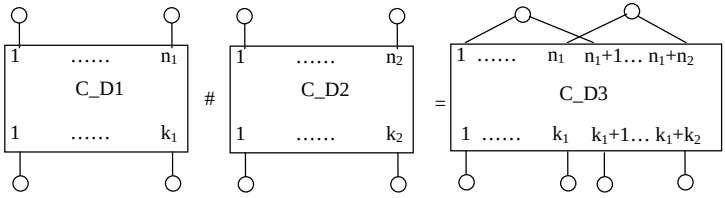

Рис. 2

Композицией носителей $C\_D_1 * C\_D_2$ будем называть вычислительный узел $C\_D_3 = <V_3, E_3, ENT_{IN}^{n3}, EXT_{k3}^{OUT}> \in CD$ такой, что: $ENT_{IN}^{n3} = ENT_{IN}^{n1}$, и $ENT_{IN}^{n3} = ENT_{IN}^{n2}$; $ENT_{IN}^{n1} = ENT_{IN}^{n3}$ $(n_1=n_3)$, $EXT_{k3}^{OUT} = EXT_{k2}^{OUT}$ $(k_3=k_2)$; $E_3 = E_1 \cup E_2 \cup (EXT_{k1}^{OUT} \cup EXT_{k1}^{OUT})$, где $k_1 = n_2$. Эта операция определена только для тех операндов, у которых число выходов первого вычислительного узла равна числу входов второго. Графически эту операцию можно проиллюстрировать рис. 3.

Для определения операции замыкания вычислительных узлов введем предварительные определения. Обозначим через Z_{IM}^{D} тензор, который сопоставляет множеству чисел $D_g \subseteq \{1, \ldots, k\}$ график с областью значений $IM_g \subseteq \{1, \ldots, n\}$. Теперь замыканием C_D_1 по Z_{IM}^{D} называется вычислительный узел: $C\_D_2 = [C\_D_1]_{g(i)} = <V_2, E_2, ENT_{IN}^{n2}, EXT_{k2}^{OUT}>$ такой, что

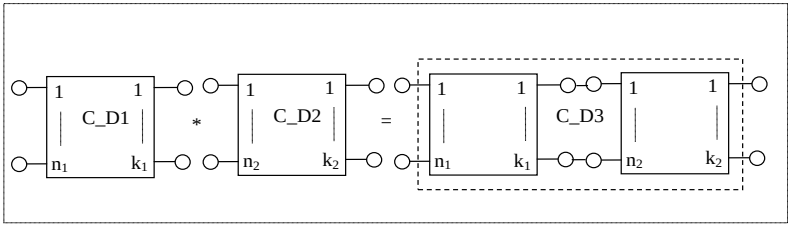

Рис. 3

$$EXT_{k2}^{OUT} = EXT_{k1}^{OUT} - EXT_{k1_{IM}}^{OUT_D} \; Z_{IM}^{D} \subseteq D_g,$$

$$ENT_{IN}^{n2} = ENT_{IN}^{n1} - ENT_{IN_D}^{n1_{IM}} \; Z_{IM}^{D} \subseteq IM_g,$$

$$E_2 = E_1 \cup \{(EXT_{ki}^{OUT}, \; ENT_{IN_D}^{n1_{IM}} \; Z_{IM}^{D}) | \; EXT_{k1_{IM}}^{OUT_D} \subseteq D_g\},$$

Таким образом, при выполнении операции замыкания $[C\_D_1 \;]_{g(i)}$ вычислительного узла C_D_1 по Z_{IM}^{D} происходит перенумерация с уплотнением входов/выходов результирующего вычислительного узла C_D_2 . Графически эта операция представлена на рис. 4.

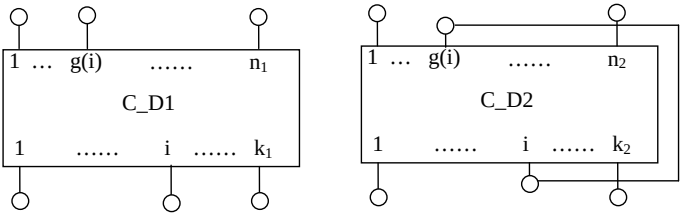

Рис. 4

Из выше приведенного определения вычислительного узла и операций объединений, слияний, композиции, замыкания следует, что структура < C, + , # , * , [] > образует частичную алгебру.

Теорема 6. При фиксированных m и n для любого временного реконфигурируемого вычислительного узла из CD существует разложение на множество пространственно реконфигурируемых вычислительных узлов CS с помощью операций объединений, слияний, композиции и замыкания.

Эта теорема гарантирует возможность представление структуры любого временного реконфигурируемого исполнителя в виде алгебраического выражения над пространственно реконфигурируемыми вычислительными узлами CS. Доопределим понятие исполнителя на случай алгебраического описания его структуры. В этом случае основную проблему представляет проблема переопределения потока сообщений M_S на случай вычислительных узлов типа Plug. Так как определение потока сообщений M_S на множестве дуг E определяет время tt, и блоки m_v, которыми передаются сообщения из M и задает пропускную способность канала (m_v, tt), то на случай узла типа Plug этот поток будет иметь следующий вид: $M\_S(Plug) = (m\_v, tt)_i$, если $Plug \in ARC_i$. Будем говорить, что это

имеет место, т.е. Plug $\in$ ARC_i , если выход узла EXT_i замкнут на вход ENT_j , такой что пара (EXT_i , ENT_j) $\in$ ARC_i.

Теперь постулируем свойства тензора Tm_n^e, который сопоставляет событию *e* количество тактов n наступивших к моменту его появления на данном исполнителе на случай временного реконфигурируемого исполнителя.

Постулат 4.

$\forall\ Tm_n^e : Tm_n^e \in SE_i \in DE_i \Rightarrow (t_0 - tt) - t_0 \geq 1$, где t_0 – момент астрономического времени начала передачи единичного блока m_v (момент начала передачи – возврат управления от процесса сопоставленного EXT)

Функционирование реконфигурируемого вычислителя определяется тем, как в каждом конкретном случае будет происходить интерпретация поведения вычислительного процесса на исполнителе. Поэтому для определения способа функционирования вычислителя необходимо построить историю вычислительного процесса по исходному поведению логической среды, исполнителю и начальному воздействию.

Конкретизируем связь логической среды и исполнителя. Для этого введем тензор привязки логической среды к исполнителю $B_{DE_i}^{LE_i}$, который устанавливает взаимно – однозначное соответствие между множеством поведений процессов логической среды LE = {BHp_i | Pa_i $\in$ {P_L $\cup$ Pa}} и множеством входов / выходов исполнителя .{SE_i} $\in$ DE_i; EXT = {ext_i | $\exists$ }SE : SE $\in$ DE&ext_i $\in$ SE}; .{SE_i} $\in$ DE_i; ENT = {ent_j | $\exists$ }SE : SE $\in$ DE&ent_j $\in$ SE}.

Основные свойства такого отображения следующие: 1) одному входу соответствует один процесс; 2) нет двух или более исполнителей, которым был бы приписан процесс с одним и тем же именем; 3) тензор привязки логической среды к исполнителю $B_{DE_i}^{LE_i}$ не меняется в течении всего периода наблюдений.

Для описания функционирования реконфигурируемого вычислителя определим вычислительное устройство как тройку COM = <BHp, SHD, CE>, где BHp – поведение вычислительного процесса; SHD – значение тензорного уравнения, которое задает соответствие между множеством переменных фиксированного вида Dm и процессами; SHD т.е. это по сути функция распределения вычислительного процесса в логической среде; CE – вычислительная среда. CE = <LE, $B_{DE_i}^{LE_i}$, DE>, а свойства всех этих объектов были уже определены.

Организация вычислительного процесса по выполнению вычислений на вычислительном устройстве есть исчисление, которое будем называть временно реконфигурируемым наблюдателем. Зададим его в виде набора

идентичных алгоритмов и правил взаимодействий между ними. Набор этих алгоритмов будем называть пространственно реконфигурируемым наблюдателем (или просто наблюдателем, если это не вызывает неоднозначности). Каждый из этих алгоритмов определяет выбор очередного шага из поведения каждого из процессов порождаемых прикладными вычислениями и логической средой.

Каждому пространственно реконфигурируемому исполнителю сопоставлен свой наблюдатель, который будем обозначать OBS. По существу каждое OBS задает частичное отображение $M \times P\_A \times A \to S$, которое определяется согласно (1). В этом случае, в зависимости от шага процесса S^*, процессов P и состояния арбитра A_{Map}^{Tm} формируется следующий шаг S следующим образом:

1 Если $A_{Map}^{Tm} \neq 0$, то P = B_{Tm}^{Map} A_{Map}^{Tm} и $S \in S_i \in BHp = S_{P_i}^{k} T_k^{P_i}$, где S_i – очередной шаг процесса P, соответствующий воздействию поступившему на вход A_{Map}^{Tm}. При этом если $p^* \in P\_L$, то put(p^*, s^*), иначе, если $p^* \in P$, то $dm_j = p^* s^*$, где p^* определяется из SHD = $P^j Dm_j$, put(dm_j) и $S^* \in BHp$.

2. Если $A_{Map}^{Tm} = 0$, то

2.1) если переменная специального вида dm_k определяет реакцию на шаге s^*, т.е. $dm_k \in rpl(s^*)$, то p определяется из SHD = $P^k Dm_k$ и выбирается шаг следующий за указанным в dm_k воздействии, которое соответствует $rpl(s^*)$ (если таких шагов несколько, то выбор случаен);

2.2) если back $\in rpl(s^*)$, то p = get(back) и выбирается шаг, который следует за указанным в back и воздействие, которое соответствует $rpl(s^*)$ (если таких шагов несколько, то выбор случаен);

2.3) если p $\in rpl(s^*)$, то берем очередной шаг процесса P, если их несколько, то тот, у которого воздействие соответствует реакции (если таких шагов несколько, то выбор случаен);

2.4) если $rpl(s^*)$ содержит обращение к атомарному процессу, то сообщение в $rpl(s^*)$ преобразуется согласно определению данного атомарного процесса и следующим будет выбран шаг, который следует за s^* в поведении этого процесса, из которого данный s^*;

2.5) если в выбранном процессе нет шага с воздействием, соответствующим $rpl(s^*)$, то работа данного OBS блокируется.

Текущим значением p^* становится s^*.

Рассмотрим свойства наблюдателя на предмет его корректности. Под корректностью наблюдателя мы будем понимать то, что последователь-

ность цепочек шагов, которую порождает OBS, является историей программы, т.е. удовлетворяет отношению R > (P) = R~(P) ∪ R □ (P).

Пусть $S^*(P)$ – множество цепочек в алфавите S(P) и w ∈ $S^*(P)$. Обозначим $[w]_{Pi}$, где $p_i \in P$, последовательность только тех шагов из w, которые принадлежат $S(p_i)$, и в том порядке, в котором они расположены в w, т.е. $[w]_{Pi} \in S^*(p_i)$. Назовем $[w]_{Pi}$ – проекцией w на p_i.

Для обоснования корректности наблюдателя надо показать то, что при $\forall i : p_i \in P, \forall j : OBS_j$ он порождает:

1) цепочку шагов, в которых не нарушены ни R □ (P), (причинно-следственные связи на BHpi), ни R ~ (P) (причинно-следственные связи между процессами), т.е. w удовлетворяет R ~ (P) и $\forall p_i : \exists Hp_i : [w]Pi = Hpi$, где w – цепочка шагов, получаемая при интерпретации. Таким образом, это будет означать, что OBSi корректно воспроизводит реконфигурацию основанную на совместном использовании функциональных элементов для решения ряда подзадач: P1~P2.

2) если процессы прикладных вычислений P распределены между двумя и более OBS, то порождаемые совокупности цепочек $\{w_k\}$ обладают тем свойством, что $p_i : p_i \in P_i, \exists w_k : [w_k]_{P_i} = Hp_i \,\&\, \{w_k\}$ удовлетворяют R ~ (P), т.е. OBS корректно воспроизводит реконфигурацию в форме полной реконфигурации P1 || P2.

Для корректного наблюдателя характерно то, что $\forall i : OBS$ не нарушает отношения R>(P), а это будет тогда, когда для любых SHD и $B_{DE_i}^{LE_i}$ множество OBS корректно порождает историю вычислительного процесса. Таким образом, введенные нами понятия корректны при корректности прикладных вычислений.

4. Заключение

В результате проведенных исследований разработана формализованная модель построения и функционирования реконфигурируемых вычислительных устройств. В строго математическом смысле в качестве модели предложена структура с исчислением, правилами вывода. Разработанная модель описывает все характерные особенности функционирования РВУ. В отличии от модели описанной в[4], предложенный нами временный реконфигурируемый исполнитель позволяет описать децентрализацию управления, конфликты на уровне физической среды и задать оба типа реконфигурации. Предложенная форма описания поведения вычислительного процесса отражает независимость вычислений от функционального элемента и времени. Исследована взаимная согласованность и корректность всех введенных в модели понятий. Все это в целом, являет-

ся основой для создания новой методики проектирования РВУ с более глубокой степенью оптимизации проектных решений.

Таким образом, разработанная формализованная модель построения и функционирования РВУ позволяет описать алгоритмические свойства и поведение реконфигурируемых вычислительных процессов и устройств. Предложенная модель может быть основой для анализа и синтеза реконфигурируемых устройств.

Список использованной литературы

1. Виноградов Ю.Н., Иванов Д.Г., Кушниренко Н.В. Структурный способ иерархической классификации реконфигурируемых вычислительных систем. // Вісник Національного технічного університету України "КПІ" Інформатика, управління та обчислювальна техніка. К.: ТОО "ВЕК+" № 46 . – 2007. С. 133 – 140 .
2. Палагин А.В., Опанасенко В.Н., Сахарин В.Г. Реконфигурируемые структуры на ПЛИС // УсиМ. – 2000. - №3 - С. 32 – 39.
3. Хорошевский В.Г. Инженерный анализ функционирования вычислительных машин и систем. - М.: Радио и связь, 1987.- 256 с.
4. Палагин А.В., Опанасенко В.Н. Реконфигурируемые вычислительные системы. – К.: Просвіта, 2006. – 280 с.
5. Чижухин Г.Н., Панферов В.П. Структурное проектирование вычислительных систем. – Пенза Пенз., политехн. Ин-т, 1985, - 88с.
6. Крон Г. Тензорный анализ сетей: Пер. с англ. / Под ред. Л.Т. Кузина, П.Г. Кузнецова. – М: Сов. Радио 1987. – 720 с.
7. Plotkin G. An operational semantics for GSP // Formal description of programming concepts. N.-H. Press, 1983. p 199 – 223.
8. Котов В.Е. Сети Петри. М.: Наука 1989. – 210 с.