УДК.004.2; 004.272.43

Ю.С. ЯКОВЛЕВ[*], Е.В. ЕЛИСЕЕВА[*]

ПРИМЕНЕНИЕ ПЛИС ДЛЯ СХЕМНОЙ РЕАЛИЗАЦИИ МОДУЛЕЙ ПРОГРАММЫ РАСПРЕДЕЛЕНИЯ ПРИЛОЖЕНИЙ ДЛЯ PIM-СИСТЕМЫ

[*]Институт кибернетики имени В.М. Глушкова НАН Украины, Киев, Украина

Анотація. *Запропоновані для PIM-системи базові положення і основні принципи схемної реалізації модулів програми розподілу додатків з використанням ПЛІС, представленої у вигляді приставки DP1 (Distribution Program) до комп'ютера. Розроблена структурна схема приставки в базисі САПР ф. Xilinx і складових її блоків з прив'язкою до запропонованого авторами алгоритму розподілу програми користувача і до відповідних його вхідних і вихідних сигналів.*
Ключові слова: *PIM-система, ПЛІС, розподіл програми користувача.*

Аннотация. *Предложены для PIM-системы базовые положения и основные принципы схемной реализации модулей программы распределения приложений с использованием ПЛИС, представленной в виде приставки DP1 (Distribution Program) к компьютеру. Разработана структурная схема приставки в базисе САПР ф. Xilinx и составляющих её блоков с привязкой к предложенному авторами алгоритму распределения программы пользователя и к соответствующим его входным и выходным сигналам.*
Ключевые слова: *PIM-система, ПЛИС, распределение программы пользователя.*

Abstract. *Base positions and basic principles of scheme realization the units of distribution program are offered for the PIM-system with the use of FPGA, presented as DP1 adapter (Distribution Program) to the computer. The flow diagram of adapter and constituents of its blocks is developed in a base CAD (computer aided design) of Xilinx with attachment to the distribution algorithm of the program of user offered by authors and to the proper his entrance and output signals.*
Keywords: *PIM-system, FPGA, program distribution of user.*

1. Введение

Сокращение времени разделения программы пользователя на блоки и распределение их по процессорам многопроцессорной системы для их параллельного выполнения является актуальной задачей, поскольку часто процесс распараллеливания требует гораздо большего времени и соответственно больших ресурсов вычислительной системы, чем при решении непосредственно самой задачи. Это привело к тому, что появилось множество вариантов решения этой актуальной задачи, каждый из которых основан либо на итерационных процессах процедур перебора, либо на вмешательстве пользователя в эти процессы, либо на существенных допущениях и ограничениях, либо, как правило, на всей совокупности этих факторов. В связи с этим авторы предложили свой вариант алгоритма распределения приложения, защищенный патентом на изобретение, который в значительной степени не только исключил эти факторы, тем самым уменьшив время распределения, но и дополнительно еще больше сократил его за счет аппаратной реализации отдельных наиболее трудоемких блоков программы пользователя, которые могут выполняться параллельно с остальными программными блоками. При этом на начальном этапе решения этой проблемы использована программируемая логическая интегральная схема (ПЛИС), которая после проверки корректности и правильности инженерно-технических решений может быть заменена на заказную большую интегральную схему (БИС), используемую в составе приставки – ускорителя ЭВМ. Хотя решение этой проблемы рассмотрено применительно к перспективной гетерогенной архитектуре типа PIM-системы (Processor-in-Memory), однако основные положения предложенного подхода могут быть использованы при проектировании гетерогенных распределенных систем различного назначения.

2. Базовые положения реализации модулей программы распределения приложений с использованием ПЛИС для PIM-системы

В основу подхода при реализации модулей программы распределения приложений приняты следующие положения, вытекающие из особенностей архитектурно-структурной организации PIM-систем:

1. Наличие двух типов процессоров – одного достаточно мощного процессора, так называемого ведущего процессора (VP) со своим набором команд и работающего под управлением своей операционной системы ($ОС_1$), и множества упрощенных процессоров, так называемых процессорных ядер (PYa), которые, в отличие от VP, имеют сокращенный набор команд и также работают под управлением собственной операционной системы ($ОС_2$).

2. Наличие распределенной оперативной памяти, организованной в виде многоуровневой системы типа $КЭШ_{VP} \rightarrow ОЗУ_{VP} \rightarrow КЭШ_{PYa}$ (распределенной по процессорам PYa) $\rightarrow ОЗУ$ (в виде внешней памяти по отношению к PIM-кристаллу). При этом память каждого PYa, как правило, существенно меньше памяти VP и имеет ограничения на размещение в ней информации, поскольку в этой памяти необходимо разместить программу, ОС2 и данные.

3. Наличие на кристалле PIM-системы коммуникационной среды, которая обеспечивает передачу информации между памятью всех распределенных PYa, а также между памятью VP и памятью каждого PYa.

Вследствие наличия существенных задержек информации из-за коммуникационной среды, возникает необходимость размещать информацию по месту ее обработки, то есть в памяти процессора, который реализует над этой информацией конкретную операцию. Учитывая ограничения по площади кристалла ПЛИС, используют, как правило, простейшую схему коммутационной среды – общую шину, что делает проблему задержек при прохождении информации по коммутационной среде еще более актуальной.

В связи с этим предложен алгоритм распределения приложений для PIM-систем, который обладает следующими свойствами [1, 2]:

1. В основу алгоритма положен поиск соответствия кодов из систем команд процессоров VP и PYa кодам операций анализируемого блока программы приложения, то есть конкретная операция из блока программы пользователя будет иметь преимущество для выполнения на том процессоре, в системе команд которого найдется команда с кодом, совпадающим с кодом данной операции, и время реализации всего блока программы этим процессором (например, VP) будет меньше времени реализации всего блока другим процессором (например, PYa).

2. Предусмотрено иерархическое распределение приложения в соответствии с п.1 с созданием специфической модели распределения на каждом уровне: на верхнем уровне происходит распределение блоков программы между VP и эквивалентным PYa*, содержащим множество однотипных PYa с одинаковыми системами команд и одинаковыми ограничениями на емкость памяти, приписанной каждому PYa , то есть модель распределения на этом уровне можно условно отобразить $VP \Leftrightarrow PY^*$, где PY^* – множество взаимосвязанных коммутационной средой PYa. На втором уровне происходит распределение блоков программы приложения между всеми PYa, объединенными в PYa*.

3. Применение нового типа коммутационной среды в виде кольцевой шины, разделенной специальными разделителями на секторные участки, к каждому из которых подключены процессоры и память, которые могут работать параллельно с процессорами и памятью других секторов [3].

4. Принцип обеспечения возможности независимого параллельного выполнения логических и арифметических операций по каналу PYa и по каналу VP (рис. 1).

3. Цель и основные принципы реализации программных модулей на ПЛИС

Цель работы: создание приставки DP1 (Distributional Program) к компьютеру для схемной реализации наиболее длительных по времени выполнения циклов программы распределения приложений для PIM-систем.

В данной работе рассмотрена реализация одного модуля программы распределения – модуля оценки веса блоков программы пользователя как с применением программной части, так и схемной части, размещенной на ПЛИС. При этом под оценкой веса каждого блока здесь понимается определение времени выполнения этого блока на VP и отдельно на PYa, а также получение заключения о возможности и приоритетности выполнения блока на каждом из этих процессоров.

Кроме того, под блоком программы пользователя здесь и далее понимается фрагмент программы пользователя, содержащий последовательность команд, расположенных в хронологическом порядке, между которыми имеются зависимости по данным.

Укрупненная блок-схема функционирования приставки DP1, выполненная с использованием условных графических образов (УГО) в базисе САПР фирмы Xilinx типа Web PACK ISE 13.2, приведена на рис. 1, а укрупненные блок-схемы каждого УГО представлены на рис. 2 и 3 (во избежание усложнения блок-схемы цепи тактовых сигналов на этих рисунках не показаны). При этом DP1 содержит:

• блок анализа “BLOCK_AN”, реализующего проверку совпадения кодов команд процесссоров PIM-системы и кодов операций каждого q-го блока программы пользователя при её распараллеливании по этим процессорам и выдачи на выход ПЛИС соответствующих сигналов (признаков);

• “BLOCK_SUM_T_VP/PYa” – блок расчета и оценки полного времени реализации Tq каждого q-го блока программы пользователя при его выполнении на ведущем процессоре PIM-системы Tq(VP) и на упрощенных процессорах – процессорных ядрах Tq(PYa), а также их сравнение для определения места выполнения (на VP или на PYa) конкретного q-го программного блока пользователя;

• “CONTR_BLOCK” – блок управления, предназначенный для формирования сигналов управления (тактовых сигналов), синхронизирующих работу всей приставки DP1.

Обозначения и функциональные назначения управляющих, а также входных и выходных сигналов ПЛИС, представленных на рис. 1–3, приведены в табл. 1 и 2 соответственно. При этом укрупненная блок-схема алгоритма работы приставки DP1 приведена на рис. 4.

При разработке приведенных схем приставки DP1 были использованы следующие основные принципы:

1. Принцип ориентации на массовое использование приставки DP1, разработка которой основана на применении доступного пакета САПР фирмы Xilinx типа Web PACK ISE 13.2 с приемлемыми требованиями к емкости основной оперативной и дисковой памяти, а также к производительности процессора инструментального средства.

2. Принцип максимального использования логических элементов и модулей из библиотеки применяемого пакета САПР фирмы Xilinx.

3. Принцип синхронизации взаимодействия программной части, размещенной в инструментальном компьютере, и схемной части, размещенной на ПЛИС путем формирования входных для ПЛИС и выходных из ПЛИС соответствующих управляющих сигналов, назначения которых приведены в табл. 1 и 2.

4. Принцип обеспечения возможности независимого параллельного выполнения логических и арифметических операций по каналам PYa и VP (рис. 1), что достигается дублированием элементов и модулей принципиальных схем, управляющих тактовых сигналов, а также входной и выходной информации, необходимой для независимой работы этих каналов.

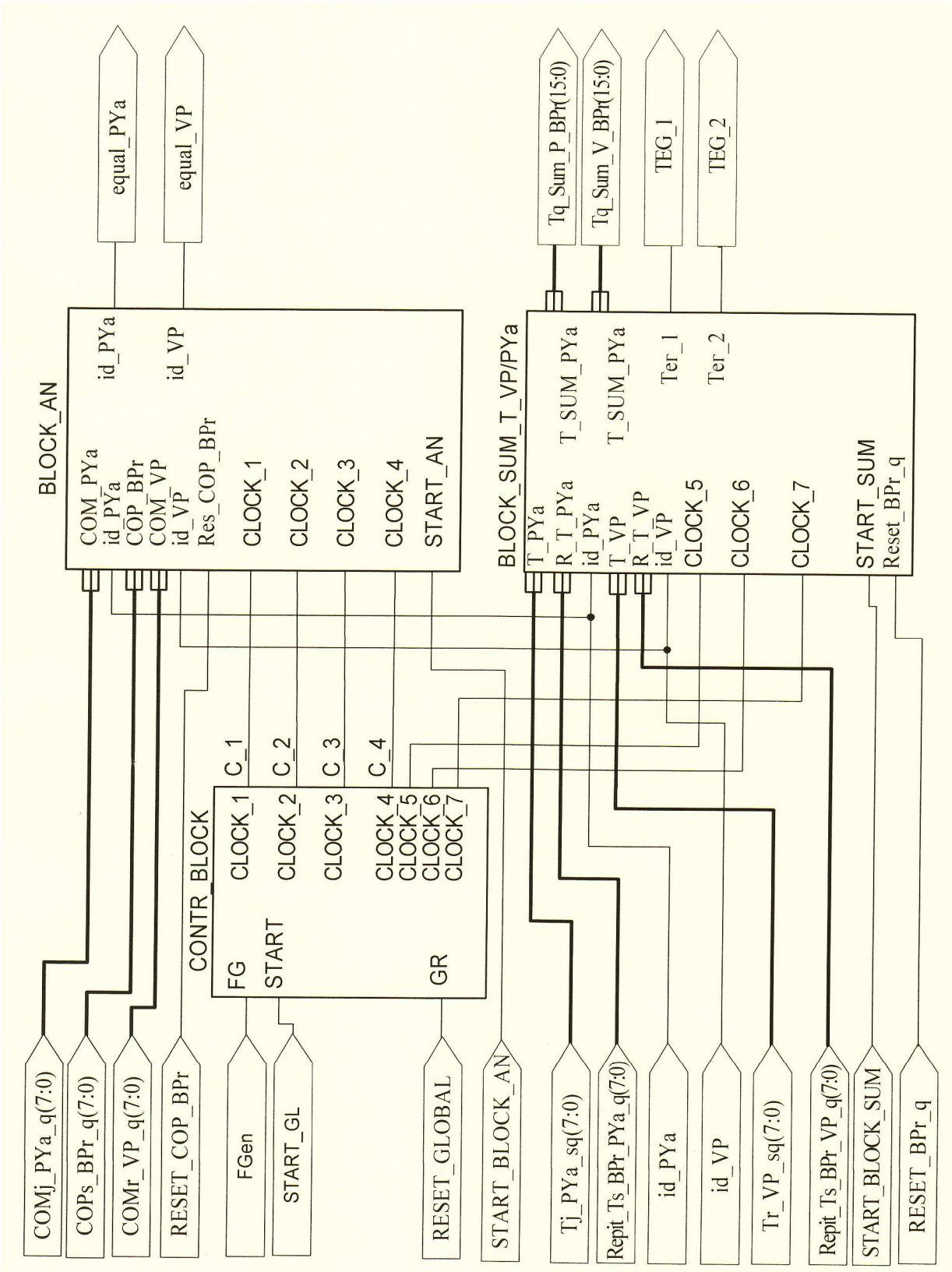

Рис. 1. Укрупненная блок-схема ПЛИС в составе приставки DP1, выполненная с использованием УГО

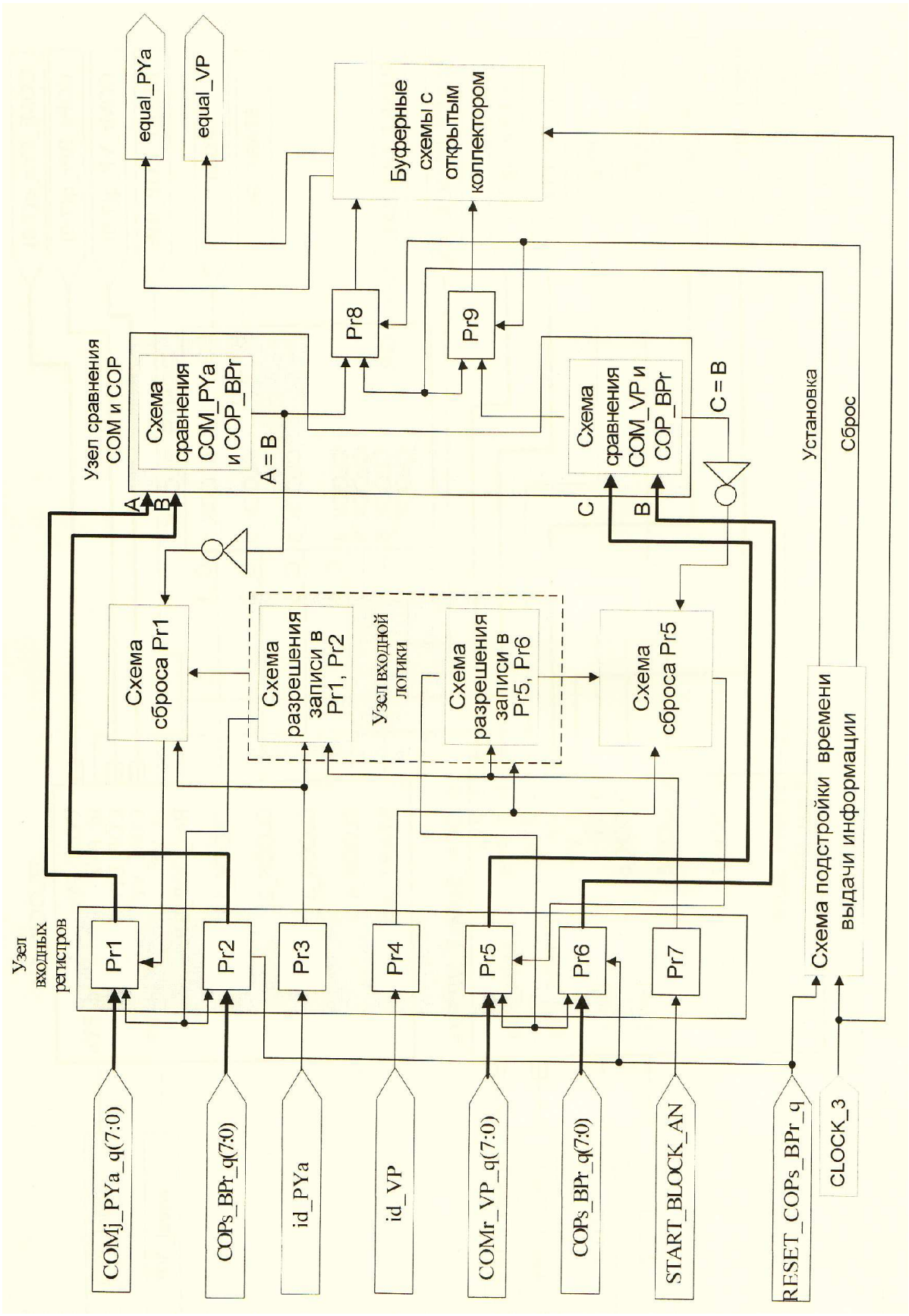

Рис. 2. Укрупненная блок-схема BLOCK_ANALIZE

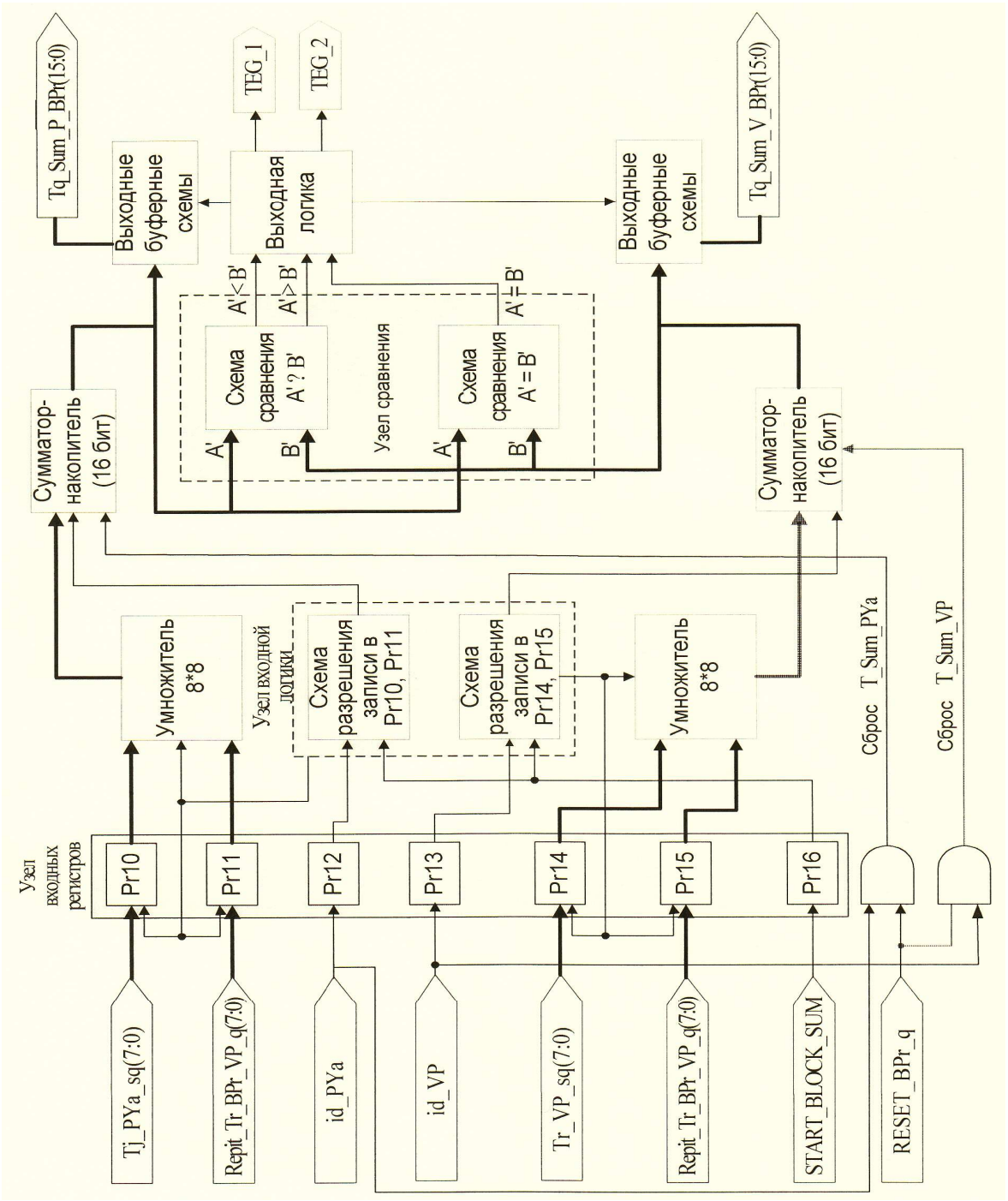

Рис. 3. Блок-схема BLOCK_SUM_T_VP/PYa

Таблица 1. Управляющие сигналы ПЛИС

Входные управляющие сигналы	
START_GL	Глобальный сигнал запуска блока управления ПЛИС (CONTROL_BLOCK), являющийся стартовым сигналом всех схем ПЛИС за счет формирования синхронизирующих последовательностей сигналов типа CLOCK_1 – CLOCK_4, и др.
RESET_GL	Сигнал установки (сброса) в исходное состояние CONTROL_BLOCK, запрещающий формирование последовательностей сигналов типа CLOCK_1 – CLOCK_7

Продолж. табл. 1

START_BLOCK_AN	Запуск блока анализа для поиска соответствия S-ого кода операции q-го блока программы пользователя j-ой команде PYa и (или) r-ой команде VP
START_SUM	Запуск блока накопления (суммирования) весов операций (времен выполнения) для VP и PYa q-го блока программы пользователя
RESET_COP	Стирание на регистрах ПЛИС анализируемого кода операций блока программы при его совпадении с кодом команды PYa или VP
RESET _BPr_q	Сброс информации, полученной аккумуляторами веса для всего q-го блока программы
FGen	Тактовая частота генератора, используемая либо от внешнего источника, либо от генератора внутри ПЛИС со специального выхода ПЛИС, замкнутого на соответствующий ее вход
Выходные управляющие сигналы	
Equal_PYa	Сигнал, указывающий на совпадение j-ой команды PYa с S-м кодом операций q-го блока программы
Equal_VP	Сигнал, указывающий на совпадение r-ой команды VP с S-м кодом операций q-го блока программы
TEG_1	Признак времени выполнения блока программы показывает, что время выполнения блока на PYa меньше, чем на VP
TEG_2	Признак времени выполнения блока программы показывает, что время выполнения блока на PYa больше, чем на VP

Таблица 2. Перечень входной и выходной информации для ПЛИС

Обозначение входных и выходных сигналов	Функциональное назначение сигналов	Обозначение блоков УГО
Входная информация		
COPs_BPr_q(7:0)	S-й код операции q-того блока программы	COP_BPr
id_ PYa	Идентификатор PYa	id_ PYa
COMj_PYa_q(7:0)	Код j-той команды PYa при сравнении с кодами операций q-того блока программы	COM_PYa
COMr_VP_q(7:0)	Код r-той команды VP при сравнении с кодами операций q-того блока программы	COM_VP
id_VP	Идентификатор VP	id_VP
Tj_PYa_sq(7:0)	Код времени выполнения j-той команды PYa при реализации S-той операции q-того блока программы	T_PYa
Tr_VP_sq(7:0)	Код времени выполнения r-той команды VP при реализации S-той операции q-того блока программы	T_VP
Repit_Ts_BPr_PYa_q(7:0)	Код количества повторений S-той операции q-того блока программы при ее реализации на PYa	R_T_PYa
Repit_Ts_BPr_VP_q(7:0)	Код количества повторений S-той операции q-того блока программы при ее реализации на VP	R_T_VP

Продолж. табл. 2

Выходная информация		
Tq_Sum_P_BPr_(15:0)	Код суммарного времени выполнения q-того блока программы при реализации его на PYa	T_Sum_PYa
Tq_Sum_V_BPr_(15:0)	Код суммарного времени выполнения q-того блока программы при реализации его на VP	T_Sum_VP

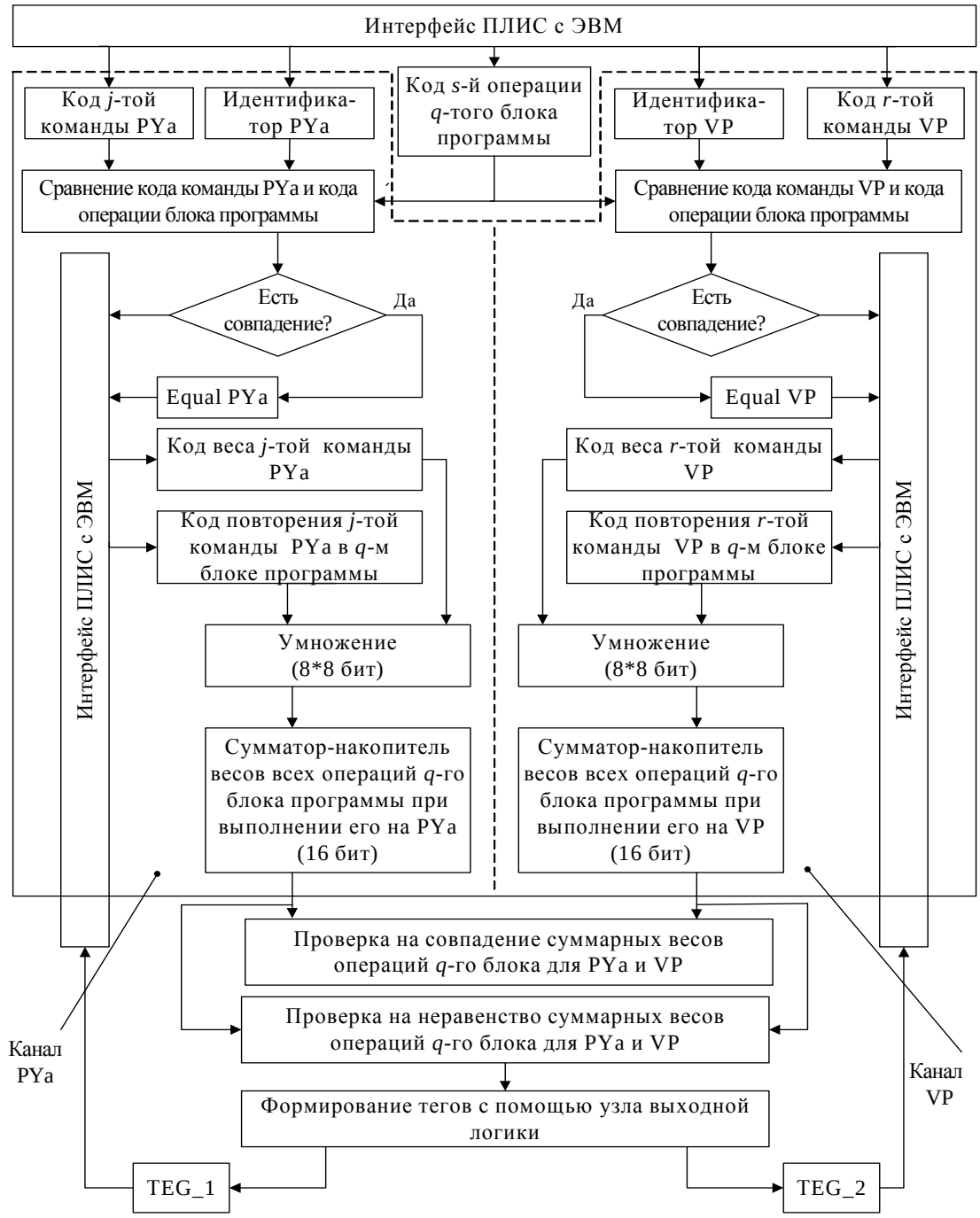

Рис. 4. Укрупненная блок-схема алгоритма функционирования блока анализа (BLOCK_AN) и блока суммирования (BLOCK_SUM_VP/PYA) приставки DP1 на ПЛИС

4. Описание работы приставки DP1

Для каждого блока программы пользователя предварительно проводится лексический анализ содержимого блока, в результате которого выдается список токенов, входящих в данный блок.

Для подготовки ПЛИС к работе на входы подаются соответствующие сигналы RESET_GLOBAL, RESET_COP и RESET_BPr_q для установки блока “Control_Block” и соответствующих регистров блока “Block_ANALIZ” в исходное состояние. Далее на ПЛИС из ЭВМ, где размещена программа распределения, подают сигналы START_Gl и FGen (рис. 1), которые запускают “Control_Block” для формирования последовательности тактовых сигналов CLOCK_1 – CLOCK_7.

Список токенов q-го блока программы пользователя поступает в DP1 из файла, расположенного на внешнем носителе информации. В цикле 1 (рис. 5) просматривается весь список токенов, и для каждого токена выполняется проверка, соответствует ли этот токен какой-либо операции.

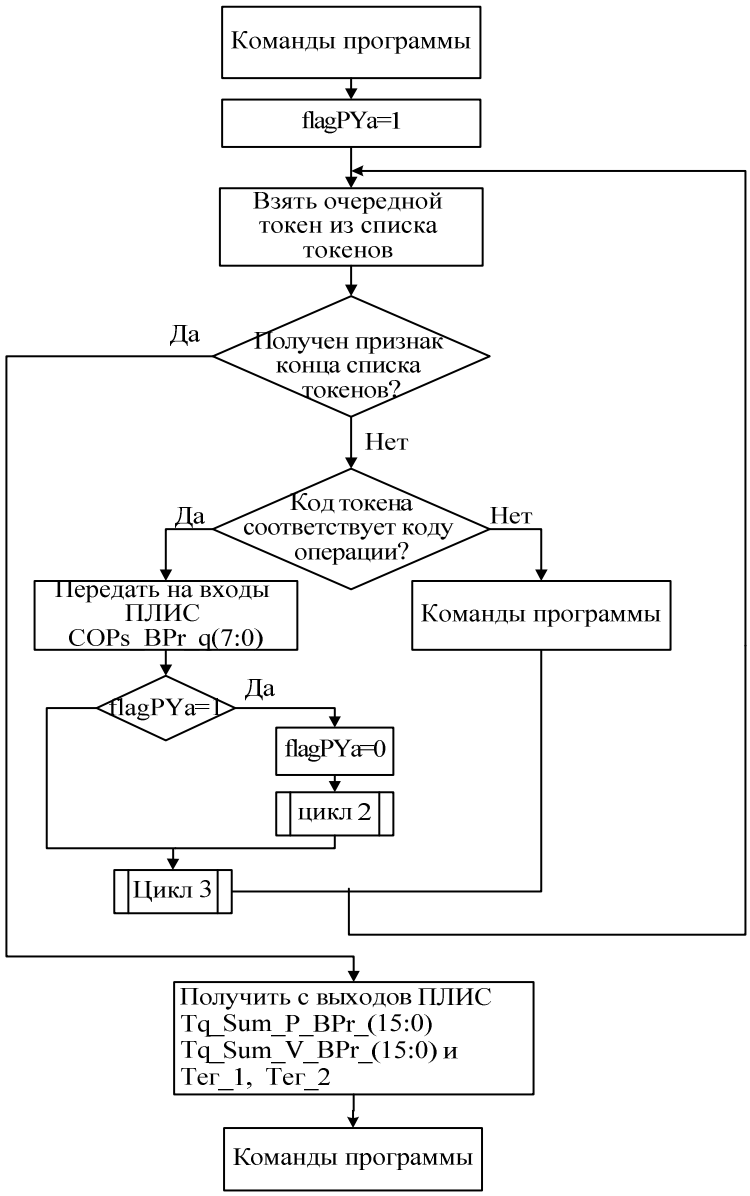

Рис. 5. Укрупненная структурная схема фрагмента алгоритма с указанием входов/выходов в/из ПЛИС. Цикл 1 – перебор всех операций, входящих в блок программы пользователя

Если рассматриваемый токен соответствует операции, то на входы ПЛИС передается значение кода этой операции COPs_BPr_q(7:0) и выполняются циклы 2 и 3 для поиска совпадений в системе команд PYa (СК PYa) и в системе команд VP (СК VP) соответственно (рис. 6). По результатам совпадений делается вывод о возможности выполнения рассматриваемой операции на PYa и (или) VP и при необходимости определяется вес (время выполнения) рассматриваемой операции для каждого из процессоров. При этом учитывается это время при подсчете времени выполнения всего блока.

После завершения работы цикла 1 с выходов ПЛИС получаются значения весов (времен выполнения) рассматриваемого блока для PYa и VP (если выполнение блока на этих процессорах возможно) и сигналы TEG_1 и TEG_2 (возможные значения этих сигналов приведены в табл. 3), которые необходимы для работы последующих модулей программы распределения.

Таблица 3. Таблица значений сигналов TEG_1 и TEG_2

Обозначение	T(PYa) < T(VP)	T(PYa) > T(VP)	T(PYa) = T(VP)
TEG_1	0	1	1
TEG_2	1	0	1

Более подробно рассмотрим реализацию цикла 2 (рис. 6).

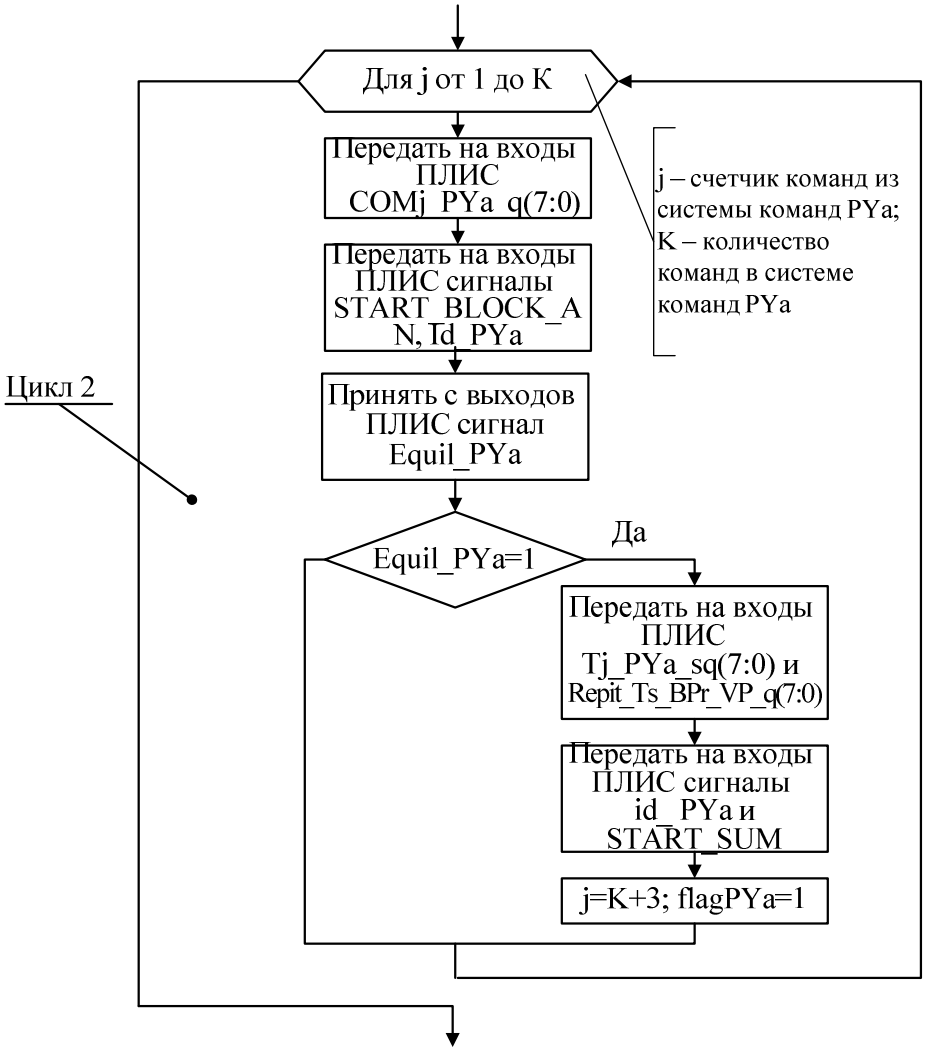

Рис. 6. Укрупненная структурная схема фрагмента алгоритма с указанием входов/выходов в/из ПЛИС. Цикл 2 – поиск соответствия в системе команд PYa

Сначала проводится проверка соответствия кода рассматриваемой операции какой-либо команде из СК_PYa. Для этого в цикле 2 просматривается каждая команда из СК PYA и на входы ПЛИС передается значение 8-разрядного кода этой команды, обозначенного на рис.1 – 3 в виде COMj_PYa_q(7:0).

Кроме того, на соответствующие входы “Block_AN” поступает 8-разрядный код s-той операции q-того блока программы (COPs_BPr_q(7:0)) (цикл1), а в цикле 2 код j-той команды PYa (COMj_PYa_q(7:0)), которые запоминаются на Pr1 и Pr2 узла входных регистров при наличии сигнала разрешения, который вырабатывается соответствующей схемой разрешения (рис. 2) узла входной логики при наличии сигналов id_PYa и START_BLOCK_AN, которые также поступили в такте (CLOCK_1) от ЭВМ и занесены на регистры Pr3 и Pr7 соответственно.

Далее на такте CLOCK_2 сигналы с выходов регистров Pr1 и Pr2 поступают на соответствующие входы A и B схемы сравнения COM_PYa и COP_BPr узла сравнения COM и COP. Если коды COMj_PYa и COPs_BPr не совпадают (рис. 2), то в блоке анализа формируется схема сброса информации на Pr1 с целью приема нового кода COMj_PYa. При равенстве этих кодов (A=B) выдается признак равенства (одноразрядный сигнал), который записывается на выходной регистр Pr8 и с него через буферные схемы с открытым коллектором выдается на выход ПЛИС сигнал “Equal_PYa”, который показывает, было ли совпадение кода операции с кодом рассматриваемой команды из СК PYA.

Если совпадение имело место, то данной операции присваивается такой же вес, как и вес соответствующей ей команды из СК_PYa. При этом на входы ПЛИС из инструментальной ЭВМ передается значение этого веса Tj_PYa_sq(7:0) и значение Repit_Ts_BPr_PYa_q(7:0), которое указывает на количество повторений данной операции в рассматриваемом q-ом блоке программы пользователя. Затем на входы ПЛИС из инструментальной ЭВМ передаются сигналы “id_PYa” и “START_BLOCK_SUM”, запуская тем самым в работу BLOCK_SUM_T_VP/PYa (рис. 1, 3). Эти сигналы запоминаются соответственно на регистрах Pr10 и Pr11, а также Pr12 и Pr16 при наличии сигналов разрешения, поступающих от узла входной логики. Далее сигналы с регистров Pr10 и Pr11 поступают на умножитель (8 × 8) бит, с выходов которого поступают на соответствующие входы сумматора-накопителя (16 бит), в результате работы которого вес данной операции будет добавлен к общей сумме времен выполнения с помощью ПЯ всех предыдущих операций блока, которая будет храниться на ПЛИС на выходных 16-разрядных регистрах до завершения работы цикла1.

Параллельно с работой блока BLOCK_SUM_T_VP/PYa на ПЛИС в программе устанавливается значение flagPYa=1. По завершении работы цикла 1 по значению flagPYa можно будет определить, может ли весь блок быть выполнен на PYa (flagPYa=1) или блок содержит операции, которые на PYa не могут быть выполнены (flagPYa=0). Дальнейший просмотр СК PYa не имеет смысла, поэтому далее работа цикла 2 завершается. Если же совпадения кода операции с кодом рассматриваемой команды из СК PYa не было, то берем следующую команду из СК PYa и повторяем для нее все рассмотренные действия (то есть переходим к следующей итерации цикла 2).

В цикле 3 проводится проверка соответствия кода рассматриваемой операции какой-либо команде из СК_VP. При этом работа в цикле 3 аналогична работе в цикле 2, только вместо сигнала id_ PYa используется сигнал id_ VP.

Необходимо подчеркнуть, что после просмотра систем команд PYa и VP и сравнения кода каждой команды с каждым кодом операций q-того блока программы, ЭВМ выдает на вход ПЛИС сигнал RESET _BPr_q, который при наличии сигналов id_VP и id_ PYa (рис. 3) используется для формирования сигналов сброса сумматора – накопителя канала PYa и отдельно сумматора-накопителя канала VP. Тем самым происходит возвращение на цикл 1. При этом ЭВМ выдает на входы ПЛИС новый q-ый блок программы BPr_(q+1) для

последующего анализа и сравнения с кодами СК_VP и СК_PYa, пока не будут перебраны все существующие блоки, которые входят в состав программы.

При этом с целью получения высокой достоверности сигналов Equal_VP и Equal_PYa на регистрах Pr8 и Pr9 (рис. 2) используется подстройка сигнала разрешения установки и сигнала сброса, которые формируются с помощью специализированной схемы (рис. 2), названной схемой подстройки времени выдачи информации из ПЛИС.

5. Выводы

Рассмотренный в данной статье один из вариантов реализации схемным способом (в частности, на ПЛИС) наиболее трудоемких частей (например, циклов) предложенной ранее авторами программы распределения по процессорам приложений [1, 2] показал целесообразность такого подхода с точки зрения сокращения времени распределения, особенно при замене в дальнейшем ПЛИС (после отладки схем и проверки их корректности) на заказную БИС, представленную в виде приставки DP1 к компьютеру. Это, в свою очередь, позволит уменьшить потребляемую мощность инструментальной ЭВМ и снизить трудоемкость процесса распараллеливания программы пользователя. Приведенные выше структурные схемы, представленные в виде УГО, а также их принципиальные схемы выполнены на ПЛИС и верифицированы в базисе САПР ф. Xilinx типа Web PACK ISE 13.2 . Предполагается, что аналогично могут быть реализованы на ПЛИС (в дальнейшем – на заказных БИС) и другие участки алгоритма распределения приложения, для поиска и обоснования которых необходимо провести дополнительные исследования.

СПИСОК ЛИТЕРАТУРЫ

1. Яковлев Ю.С. О реализации распределения приложения для параллельного выполнения на PIM-системе / Ю.С. Яковлев, Е.В. Елисеева // Інформаційні технології та комп’ютерна інженерія. – 2011. – № 3. – С. 53 – 59.

2. Пат. на винахід 103535, Україна, МПК G06F 9/44, G06F 9/45. Спосіб розподілу програми користувача для комп'ютерної системи / Сергієнко І.В., Палагін О.В., Боюн В.П., Яковлєв Ю.С., Єлісєєва О.В.; Інститут кібернетики імені В.М. Глушкова НАН України; заявл. 03.01.2012; опубл. 25.10.2013, Бюл. № 20.

3. Пат. на винахід 99164 Україна, МПК G06F 15/16, G06F 13/42. Інтелектуальна розподілена система пам'яті з кільцевою шиною / Палагін О.В., Яковлєв Ю.С., Тихонов Б.М., Єлісєєва О.В.; Інститут кібернетики імені В.М. Глушкова НАН України; заявл. 16.07.2010; опубл. 25.07.2012, Бюл. № 14. – 21 с.

Стаття надійшла до редакції 12.11.2014