УДК 680.3

Г.А. ПОЛЯКОВ

Академия наук прикладной радиоэлектроники,Россия

ГАРАНТОСПОСОБНЫЕ АДАПТИВНЫЕ СИСТЕМЫ УПРАВЛЕНИЯ И ТЕХНОЛОГИИ АВТОМАТИЧЕСКОГО ПРОЕКТИРОВАНИЯ ИХ ПАРАЛЛЕЛЬНОГО АППАРАТНО-ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Существенным компонентом гарантоспособных систем управления являются адаптивные параллельные ВС. Формулируются концепция и принципы построения Адаптивных Самоорганизующихся Вычислительных Систем (АСВС), рассматриваются обобщенная архитектура АСВС и проблемы проектирования параллельного аппаратно-программного обеспечения. Дается характеристика аппарата формального синтеза АСВС – Структурно-Временной Дискретной Математики. Описывается технология автоматического синтеза временных мультипараллельных моделей задач – Синтезатор Параллельных Моделей. Рассматривается технология автоматического синтеза временных параллельных программ – Параллельный Интеллектуальный Компилятор. Описывается технология автоматического синтеза мультипараллельных цифровых аппаратных средств – Автоматический Интеллектуальный Проектировщик.

Ключевые слова: *Параллельная вычислительная система (ВС), самоорганизация, адаптация, интеллектуализация, структуры семантико-числовой спецификации (СЧС), структурно-временная дискретная математика, временная параллельная программа, параллельное аппаратно-программное обеспечение, автоматический синтез, верификация, визуализация, Синтезатор Параллельных Моделей (СПМ), Параллельный Интеллектуальный Компилятор (ПИК), Автоматический Интеллектуальный Проектировщик (АИП).*

Введение

Научно – технический прогресс предъявляет постоянно растущие требования к надежности, достоверности и времени решения все более сложных задач управления критическими объектами и системами (например, объектами атомной энергетики, исследования космоса, телекоммуникации, медицины, проектирования, робототехники и т.д.) [1]. В то же время, несмотря на определенные успехи в создании многопроцессорных супер-ЭВМ [2], основные производители и пользователи пришли к выводу, что дальнейшее повышение гарантоспособности, эффективности и расширение областей применения компьютерной техники сдерживается крайне слабыми возможностями параллельных ЭВМ по самоорганизации их архитектуры и адаптации к изменению областей применения, характеру решаемых задач и изменению требований и ограничений.

В настоящее время признается, что «Сложные высокотехнологичные системы следующих десятилетий требуют революционных, а не эволюционных изменений. Широкий взгляд требует широких шагов – нам необходимы космические системы, которые смогут самообучаться и адаптироваться в процессе выполнения поставленных перед ними задач» [3, 4]. Считается, что стратегией развития вычислительных средств в первой половине XXI столетия будет создание динамически перестраиваемых и самопрограммирующихся параллельных ВС, воспроизводящих логику работы мозга человека [5].

По мнению автора, первым этапом реализации этой стратегии является создание Адаптивных Самоорганизующихся Вычислительных Систем (АСВС) с мультипараллельной обработкой данных, способных автоматически, средствами самой системы, целенаправленно изменять или синтезировать (с целью повышения эффективности) свою архитектуру, функционирование и характеристики при изменении областей применения, решаемых задач и предъявляемых внешней средой требований/ограничений [6, 7].

Следующим этапом должен быть переход к созданию Интеллектуально – Самоорганизующихся Вычислительных Систем (ИСВС), объединяющих в рамках единой системы возможности мозга по оценке ситуаций и принятию решений, с одной стороны, и автоматическое создание временных математических моделей параллельного решения задач, автоматический синтез архитектуры мультипараллельных аппаратно-программных средств и собственно параллельное решение задач с выполнением заданных требований и ограничений, с другой стороны.

Цель статьи – изложение концепции и принципов построения АСВС; рассмотрение аппарата формального синтеза параллельного аппаратно-программного обеспечения параллельных процессоров и ВС – структурно-временной дискретной математики; описание технологий автоматического проектирования мультипараллельных аппаратных и программных средств традиционных параллельных ВС и АСВС.

1. Принципы построения, обобщенная архитектура АСВС и проблемы проектирования параллельного аппаратно-программного обеспечения

Концепция АСВС - способность средствами АСВС автоматически синтезировать и изменять в реальном времени, с целью повышения эффективности, архитектуру аппаратно-программных средств, функционирование и характеристики соответственно изменению областей применения, решаемых задач, требований и ограничений [6, 7, 8].

Принципы построения АСВС, обеспечивающие реализацию концепции:

1. Множественность и автоматический выбор состава методов параллельной обработки данных.
2. Совместная обработка семантической и числовой информации.
3. Использование временных мультипараллельных программ.
4. Автоматический выбор оптимального соотношения аппаратно-программных средств.
5. Автоматический синтез мультипараллельных программных средств.
6. Автоматический синтез мультипараллельных аппаратных средств.
7. Принцип «временного управления» на каждом такте параллельного процесса.
8. Автоматический синтез средств контроля и поддержки достоверности вычислений.
9. Автоматическая визуализация параллельного Hard&Soft и показателей эффективности.
10. Спецификация исходных задач с помощью языков последовательного программирования.

Состав компонентов Системы самоорганизации мультипараллельных программных и аппаратных средств АСВС и Системы реализации мультипараллельных процессов АСВС представлен на рис. 1, рис. 2.

Реализация концепции и принципов построения АСВС требует решения ряда математических и инженерных проблем, их состав представляет рис. 3.

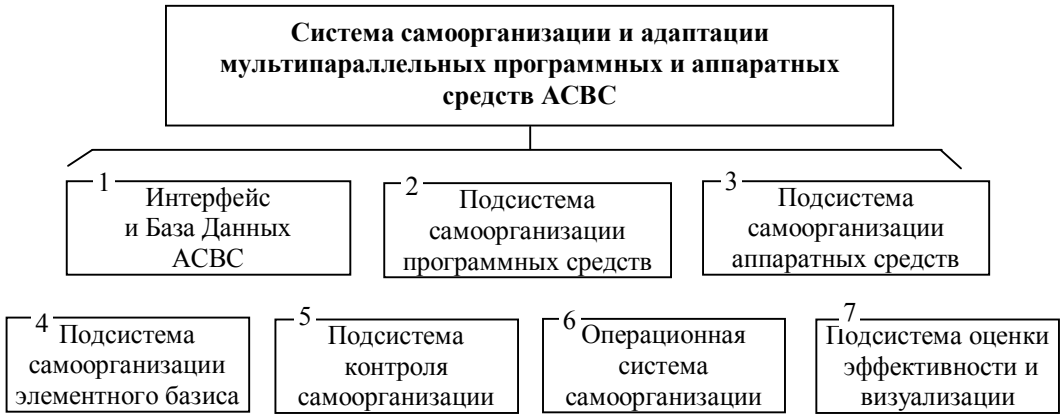

Рис. 1. Состав функциональных подсистем самоорганизации и адаптации параллельных программных и аппаратных средств АСВС

Рис. 2. Состав функциональных подсистем реализации мультипараллельных процессов АСВС

Рис. 3. Основные проблемы теории и технологий синтеза Адаптивных Самоорганизующихся Вычислительных Систем

2. Структурно-Временная Дискретная Математика – аппарат формального синтеза Адаптивных Самоорганизующихся Вычислительных Систем

Математической основой методов формального синтеза и технологий автоматического проектирования мультипараллельного аппаратно-программного обеспечения АСВС является классический аппарат теории множеств и аппарат Структурно-Временной Дискретной Математики (ВДМ), разработанный автором в период 1961…2009 г.г. [6, 8–10]. Задачами аппарата структурно ременной дискретной математики являются:

1. Поддержка создания конструктивной теории формального синтеза и анализа высокопроизводительных параллельных средств вычислительной техники (в том числе параллельных специализированных и универсальных процессоров; традиционных многопроцессорных вычислительных систем – ВС; распределенных вычислительных сетей, РВС/GRID; перспективных Адаптивных Самоорганизующихся Вычислительных Систем.

2. Поддержка создания технологий автоматического проектирования мультипараллельного программного и аппаратного обеспечения двойного использования:

– как самостоятельных автоматических технологий проектирования параллельных аппаратных и программных продуктов цифровой вычислительной техники;

– как технологий автоматического проектирования, включаемых в состав архитектуры АСВС в качестве встроенных средств самоорганизации и адаптации.

Состав компонентов структурно-временной дискретной математики представляет рис. 4.

Рис. 4. Компоненты структурно-временной дискретной математики

Основными составляющими структурно-временной дискретной математики являются:

Алгебра Структур Семантико-Числовой Спецификации (СЧС) представляет собой основное средство семантико-числовой спецификации и формальных преобразований всех объектов теории и технологий синтеза Адаптивных Самоорганизующихся Вычислительных Систем: исходных задач, схем мультипараллельных аппаратных средств

(на системном, функциональном, логическом и вентильном уровнях), параллельных программных средств, топологий ВС и временных моделей их функционирования и т.п. Алгебра Семантико-Числовых Структур включает: статические и временные семантико-числовые структуры, содержащие массивы семантических, числовых, статических и временных характеристик специфицируемого структурой объекта синтеза/анализа; множество операций, обеспечивающих статические и временные преобразования семантико-числовых структур и представляемых ими объектов параллельных аппаратно-программных средств; библиотеку программ, выполняющих автоматические преобразования статических и временных семантико-числовых структур. Аналоги Алгебры СЧС отсутствуют.

Алгебра временных последовательностей кодовых матриц (АКМ) является аппаратом формального синтеза и анализа цифровых устройств на функциональном, логическом и вентильном уровнях проектирования аппаратных средств. АКМ включает: битовые позиционные кодовые матрицы (КМ) и временные последовательности кодовых матриц (ВПКМ); множество операций над КМ и ВПКМ, обеспечивающих формальные арифметические, логические, пространственные и временные преобразования функции и формулы АКМ; библиотеку программ, выполняющих автоматические преобразования функций и формул АКМ и соответствующих им структур семантико-числовой спецификации. Аналоги АКМ автору неизвестны, широко используемый в системах автоматизированного проектирования цифровых устройств классический аппарат булевой алгебры является предельным частным случаем Алгебры Кодовых Матриц.

Кодово-матричный метод обработки данных (КММ) определяет необходимые и достаточные условия корректности представления числовых данных кодовыми матрицами; формулирует достаточные условия возможности выполнения операций над числовыми данными в виде операций с кодовыми матрицами; устанавливает возможность выполнения параллельных алгоритмов, оперирующих с числовыми данными, как процессов временной параллельной обработки двумерных кодовых матриц; обеспечивает повышение быстродействия аппаратных средств за счет сокращения времени выполнения основных алгоритмических операций. Метод является обобщением традиционных способов выполнения операций над числовыми данными и организации вычислительного процесса, применяемых в традиционных последовательных и параллельных ЭВМ (от первых последовательных машин с архитектурой фон Неймана до параллельных супер-ЭВМ) и включает эти способы в качестве частных случаев. Аналоги Кодово-Матричного Метода как общего метода обработки цифровой информации отсутствуют.

Конструктивные Графы аппаратно-программных средств (КГ) – это средство графической визуализации текстов традиционных последовательных и параллельных программ, схем параллельных аппаратных средств (на системном, функциональном, логическом и вентильном уровнях проектирования).

Алгебра временных параллельных граф-схем (ВПГС) – средство визуального представления временных параллельных моделей задач, временных моделей традиционных последовательных и параллельных программ, моделей работы параллельных схем аппаратных средств (на функциональном, логическом и вентильном уровнях детализации), временных параллельных моделей комплексных алгоритмов сложных информационно-управляющих систем.

3. Технология автоматического синтеза временных мультипараллельных моделей задач – Синтезатор Параллельных Моделей (СПМ)

Назначение теории и технологии СПМ [6, 9] – решение проблемы формального и автоматического синтеза временных моно- и мультипараллельных программно и аппаратно ориентированных моделей задач для перспективных Адаптивных Самоорганизующихся Вычислительных Систем, а также параллельных моделей для известных классов параллельных процессоров (например, VLIW, FLOW/DFC – Data Flow Computers) и многопроцессорных ЭВМ (классов SMP, NUMA, MPP, CLUSTER).

Состав поддерживаемых СПМ методов параллельной обработки представляет рис. 5 [7].

Метод совмещения операций заключается в одновременном начале выполнения операторов алгоритма, для которых выполняются следующие условия [6, 7]: операторы не связаны информационными и/или управляющими связями; для каждого из таких операторов к рассматриваемому моменту времени имеются значения соответствующих операндов.

Метод конвейерной обработки заключается в разделении временного алгоритма на "конвейерные" фрагменты равной временной глубины ΔT и последовательной реализации фрагментов во времени с выполнением различными фрагментами операций над различными наборами входных данных.

Декомпозиционный метод параллельной обработки (называемый также мультиконвейерным) заключается в выполнении следующих преобразований над входными данными:

1. Декомпозиция исходной последовательности данных на подпоследовательности, количество которых и номера включаемых в каждую подпоследовательность элементов определяются значением коэффициента декомпозиции $K = \Delta T / \Delta t$, где ΔT – интервал ввода данных в обрабатывающее устройство, равный времени выполнения устройством требуемой операции или функции, Δt – требуемое значение интервала ввода элементов исходной последовательности данных (определяемое, например, заданной шириной спектра обрабатываемых устройством сигналов).

2. Выполнение требуемой операции или функции для элементов каждой из подпоследовательностей данных с интервалом ввода, равным ΔT, и временем относительного сдвига процессов обработки подпоследовательностей данных, имеющих смежные номера, равным Δt.

3. Формирование выходного результата в виде объединения результатов реализации требуемой операции или функции, полученных для каждой из декомпозиционных подпоследовательностей данных.

Сущность метода мультипараллельной смеси (МСА) заключается в создании на операционном уровне смеси алгоритмов и использовании такой смеси для достижения 100%-й загрузки оборудования.

Сущность кодово-матричного метода заключается в представлении обрабатываемых данных в общем случае двумерными кодовыми матрицами (2-х и многорядными кодами) и выполнении алгоритмов как реализации множества кодово-матричных операций.

Архитектура технологии автоматического синтеза временных мультипараллельных моделей задач показана на рис. 6.

В состав Синтезатора Параллельных Моделей (СПМ) входит технология собственно синтеза временных мультипараллельных моделей задач (символы 1…4), верификатор аппаратно/программно ориентированных параллельных моделей и их показателей эффективности (символ 7).

Проиллюстрируем результаты выполнения основных этапов технологии СПМ на конкретных примерах синтеза временных параллельных моделей, использующих метод совмещения независимых операций и метод параллельно-конвейерной обработки данных.

Рис. 5. Методы параллельной обработки и обеспечиваемые ими прикладные эффекты

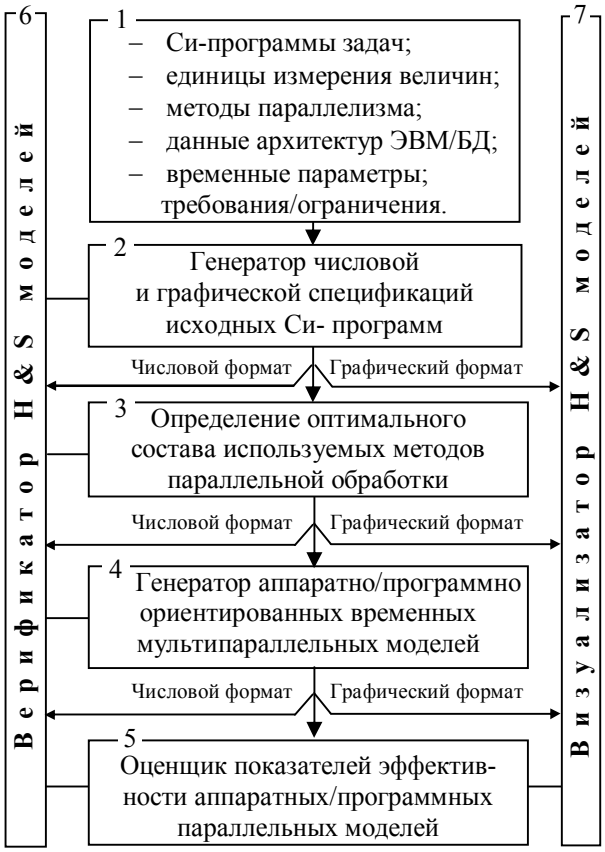

Рис. 6. Технология автоматического синтеза временных мультипараллельных моделей Си – программ (СПМ)

3.1. Пример семантико-числовой и графической спецификации Си-программы

Исходный текст Си-программы представляет рис. 7. Структуры СЧС (символ 2 рис. 6) исходного текста показаны в табл. 1. и 2.

```
#include <stdio.h>
void main(void)
{
    int a, b;
   int k, z, p, s t;
   scanf("%d %d% d",&a, &b);
   k = a / 2;
   z = a * b;
   p = k % z ;
   s = z / k;
   t = p + s;
   printf("%4d\n",t);
}
```

Рис. 7. Исходный текст Си – программы

Базовая структура BF (табл. 1) содержит в своем составе следующие массивы данных:

N – массив номеров инструкций Си-программы (или «вершин» – при работе с конструктивным графом Си-программы);

MET – массив меток инструкций;

TYP – массив типов инструкций Си-программы;

NSJ – массив указателей на начало цепочки номеров входных/сопряженных (для конкретной инструкции) инструкций Си-программы;

SJD – массив количеств входных / «сопряженных» (для рассматриваемой инструкции) инструкций Си-программы;

BJ – массив номеров естественных частей Си-программы (неразветвляющихся фрагментов инструкций Си-программы);

NWJ – массив указателей на начало цепочки номеров выходных/«внешних инструкций Си-программы для рассматриваемой инструкции;

WJD – массив количеств выходных/ «внешних» (для рассматриваемой инструкции) инструкций Си-программы;

VH и VIH – количество входов (операндов) и количество выходов (выходных данных) инструкции/функции Си-программы;

RES – массив имен данных и операций/функций.

Таблица 1

Базовая структура **BF** Си-программы

N	MET	TYP	NSJ	SJD	BJ	NWJ	WJD	MP1	MP2	VH	VIH	REZ
0	0	58	-1	0	0	0	0	0	0	0	1	a_in
1	0	58	-1	0	0	1	0	0	0	0	1	b_in
2	0	47	-1	0	0	2	0	0	0	0	2	a
3	0	47	-1	0	0	3	0	0	0	0	2	b
4	0	47	-1	0	0	4	0	0	0	0	2	k
5	0	47	-1	0	0	5	0	0	0	0	2	z
6	0	47	-1	0	0	6	0	0	0	0	2	p
7	0	47	-1	0	0	7	0	0	0	0	2	s
8	0	47	-1	0	0	8	0	0	0	0	2	t
9	0	12	0	2	0	9	2	0	0	2	1	=
10	0	12	2	2	0	11	0	0	0	2	1	=
11	0	57	-1	0	0	12	0	0	0	0	1	C2_
12	0	4	4	2	0	13	0	0	0	2	1	/
13	0	12	6	2	0	14	2	0	0	2	1	=
14	0	3	8	2	0	16	0	0	0	2	1	*
15	0	12	10	2	0	17	2	0	0	2	1	+
16	0	5	12	2	0	19	0	0	0	2	1	%
17	0	12	14	2	0	20	0	0	0	2	1	=
18	0	4	16	2	0	21	0	0	0	2	1	/
19	0	12	18	2	0	22	0	0	0	2	1	=
20	0	1	20	2	0	23	0	0	0	2	1	+
21	0	12	22	2	0	24	2	0	0	2	2	=
22	0	49	24	1	0	-1	0	0	0	1	0	stop
23	0	48	25	1	0	-1	0	0	0	1	0	t_out

Для понимания соответствия между Си-программой, базовой структурой BF и структурой связей CF (табл. 1, табл. 2) на рис. 8 показано графическое представление Си программы в виде конструктивного Си-графа.

Структура связей CF (табл. 2) содержит в своем составе следующие массивы данных:

NN – номер строки в структуре CF;

SPJD –сопряженное множество для конкретной инструкции Си-программы (либо конкретной вершины графа программы);

Таблица 2

Структура связей **CF** Си – программы

N	JSD	SPJD	SNWIH	SNWHO	TSS	JWD	WPJD	WNWHO	WNWIH	TVS
0	1	0	0	0	0	-1	9	0	0	0
1	-1	2	1	1	2	-1	10	0	0	0
2	3	1	0	0	0	-1	9	1	1	2
3	-1	3	1	1	2	-1	10	1	1	2
4	5	9	0	0	0	-1	13	1	1	2
5	-1	11	0	1	0	-1	15	1	1	2
6	7	4	1	1	2	-1	17	1	1	2
7	-1	12	0	0	0	-1	19	1	1	2
8	9	9	0	0	0	-1	21	1	1	2
9	-1	10	0	1	0	10	12	0	0	0
10	11	5	1	1	2	-1	14	0	0	0
11	-1	14	0	0	0	-1	14	1	0	0
12	13	13	0	0	0	-1	12	1	0	0
13	-1	15	0	1	0	-1	13	0	0	0
14	15	6	1	1	2	15	16	0	0	0
15	-1	16	0	0	0	-1	18	1	0	0
16	17	15	0	0	0	-1	15	0	0	0
17	-1	13	0	1	0	18	16	1	0	0
18	19	7	1	1	2	-1	18	0	0	0
19	-1	18	0	0	0	-1	17	0	0	0
20	21	17	0	0	0	-1	20	0	0	0
21	-1	19	0	1	0	-1	19	0	0	0
22	23	8	1	1	2	-1	20	1	0	0
23	-1	20	0	0	0	-1	21	0	0	0
24	-1	21	1	0	1	25	22	0	1	1
25	-1	21	0	0	0	-1	23	0	0	0

JSD – указатель на продолжение цепочки номеров инструкций Си-программы (или вершин графа), входящих в сопряженное множество SPJD конкретной инструкции Си-программы (либо вершины графа);

SNWIH – номер выхода конкретной сопряженной инструкции (вершины), связанной с входом рассматриваемой вершины;

SNWHO – номер входа инструкции (вершины), связывающего данную вершину с выходом соответствующей сопряженной инструкции (вершины);

WPJD – внешнее множество инструкций (или вершин графа Си-программы;

JWD – указатель на продолжение цепочки номеров внешних инструкций (вершин графа), образующих внешнее множество WPJD конкретной инструкции Си-программы (либо вершины графа);

WNWHO – номер входа инструкции (вершины), являющейся внешней для рассматриваемой инструкции (вершины);

WNWIH – номер выхода рассматриваемой инструкции (вершины), связывающего ее с соответствующей внешней инструкцией Си-программы (внешней вершиной графа).

3.2. Пример синтеза временной максимально параллельной модели и моделей с ограниченной шириной процесса (метод параллелизма – совмещение независимых операций)

Используем для иллюстрации задачу решения системы двух линейных уравнений по алгоритму Крамера, исходный текст Си-программы которой представляет рис. 9, длительности выполнения операций различных типов (в количестве тактов процессора R 10 000 MIPS Technology), используемые при синтезе параллельной временной модели, представляет табл. 3.

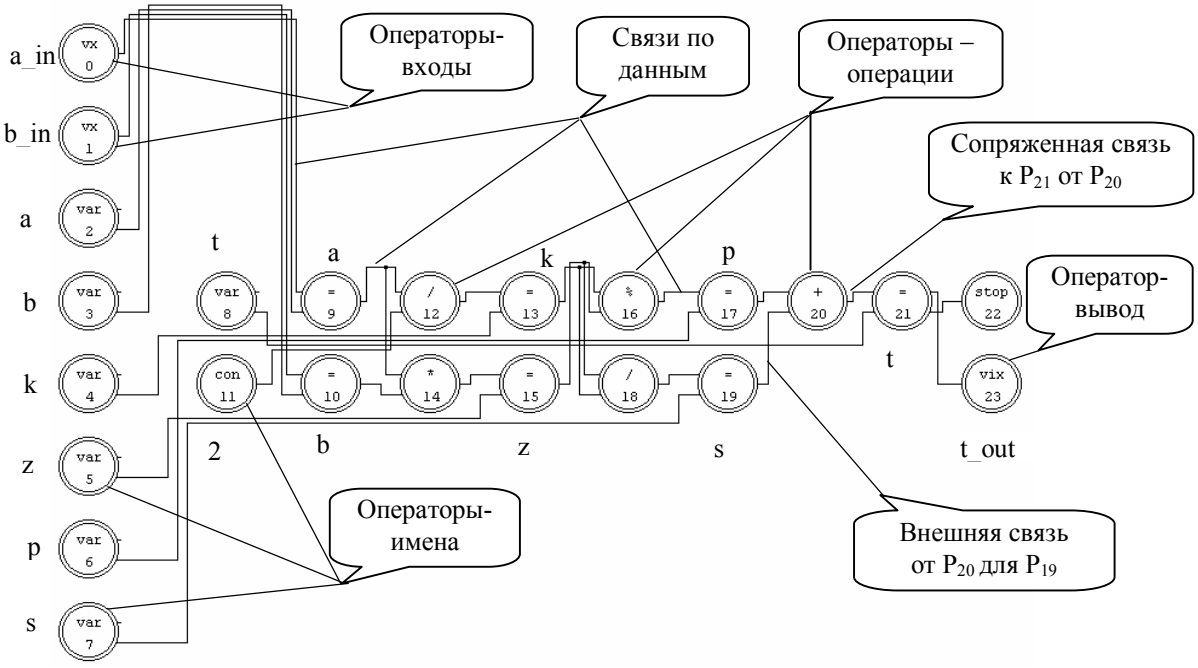

Рис. 8. Конструктивный граф (Си-граф) графической спецификации Си-программы

Таблица 3

Длительность t_j^0 выполнения операций различных типов (такты)

typ	vx	+,-	=	*	%	/	vix	bp	stop	cont
t_j^0	1.00	1.00	2.00	10.00	35.00	35.00	1.00	1.00	1.00	1.00

```
#include <stdio.h>
void main(void )
{
            int a, b, c d, e, f;
            int m,x,y;
scanf("%d %d %d\n", &a, &b, &c);
scanf("%d %d %d\n", &d, &e, &f);
     m = (b * d) - (a * e);
      x = ((b * f) - (c * e)) / m;
      printf("%3d", x);
      y = ((a * f) - (c * d)) / m;
      printf("%3d", y);
}
```

Рис. 9. Исходная Си-программа решения системы двух линейных уравнений по алгоритму Крамера

Содержание этапа синтеза семантико-числовой и графической спецификации Си-программы (рис. 6 символ 2 технологии автоматического синтеза временных мультипараллельных моделей) рассмотрено в Примере 1, поэтому структуры СЧС и Си-граф Си-программы не приводятся. Графическая визуализация (ВПГС) результата синтеза временной максимально параллельной модели (этап 4 рис. 6) показана на рис. 10.

Для оценки показателей эффективности параллельных моделей с различной шириной параллельного процесса синтезированы модели при изменении ширины процесса/количества NM процессоров в диапазоне NM =1,2,…,6. Зависимости времени T = f1(NM) решения задачи и величины загрузки S=f2(NM) оборудования от NM представляют рис. 11 и рис. 12. Пример временной параллельной модели с шириной процесса NM = 2 показан на рис. 13.

Результаты верификации корректности состава операторов временной модели, связей операторов и распределения операторов по временным ярусам модели (символ 6 рис. 6) представляет рис. 14.

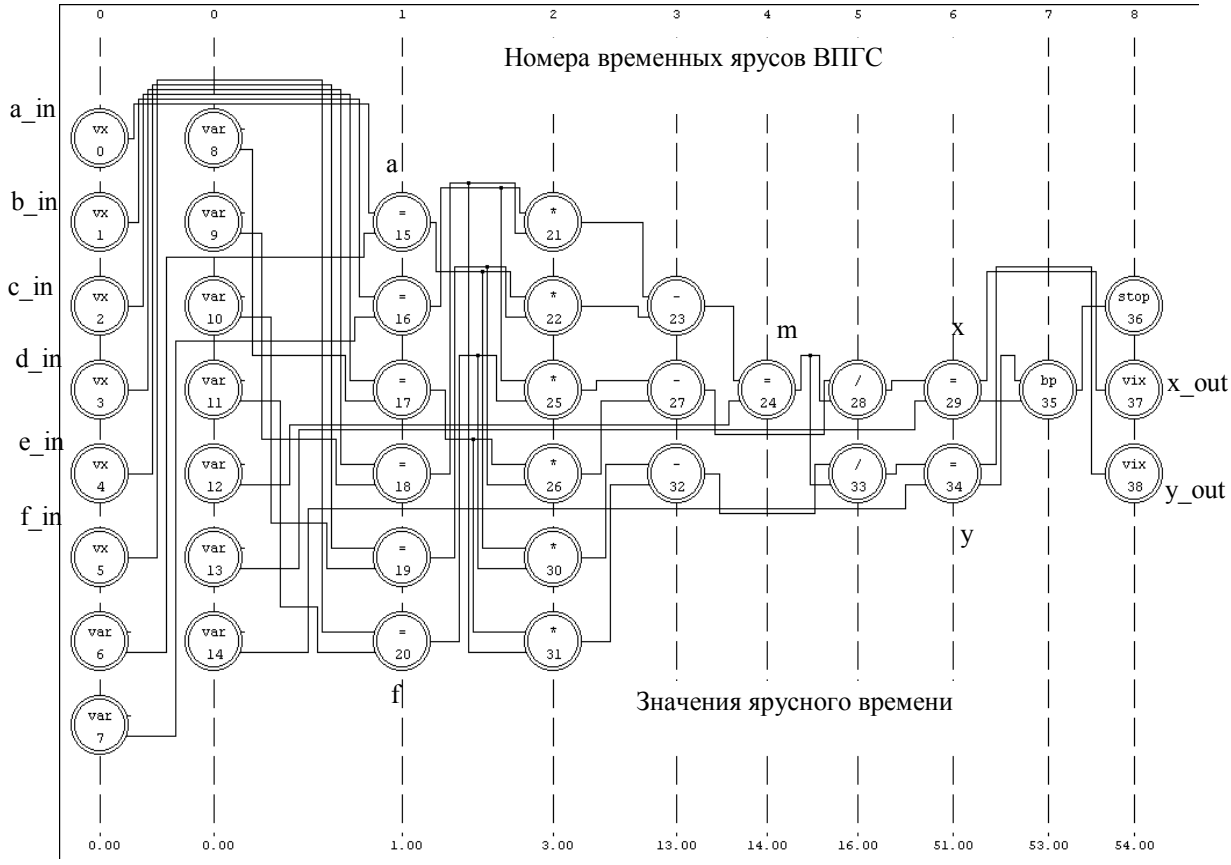

Рис. 10. Временная максимально параллельная модель (ВПГС) решения системы двух линейных уравнений (совмещение операций)

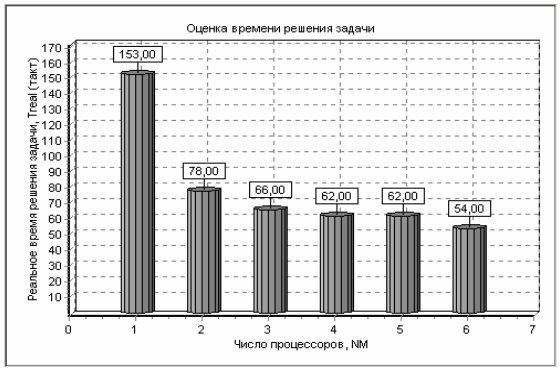

Рис. 11. Зависимости времени T = f1(NM) решения от NM

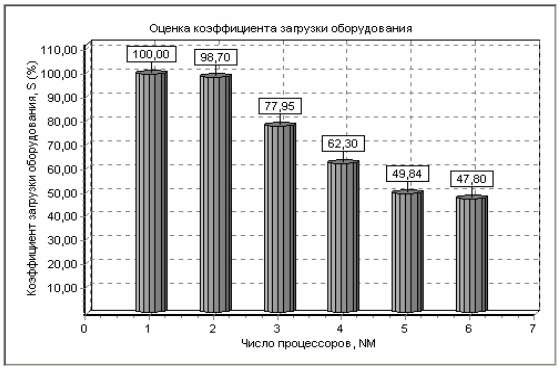

Рис. 12. Зависимости загрузки S=f2(NM) оборудования от NM

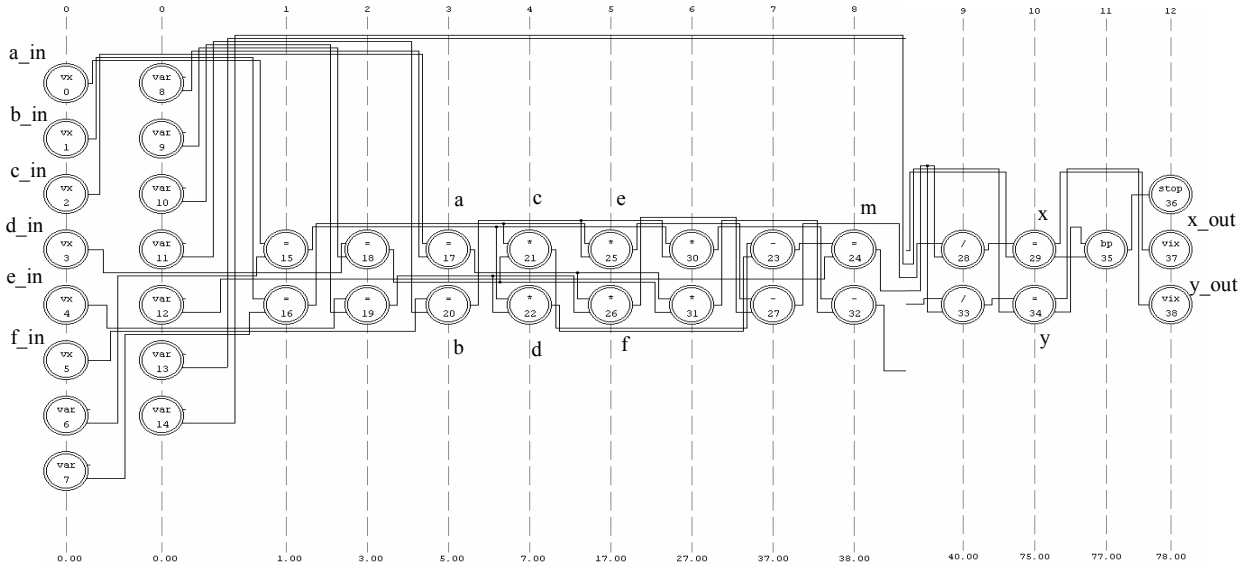

Рис. 13. Временная параллельная модель решения системы линейных уравнений (совмещение операций)

```
файл элементов:                    C:\My_prog\C\Result\TEST2_A\vixvix1.txt
файл связей элементов:             C:\My_prog\C\Result\TEST2_A\vixvix2.txt
файл размещения по ярусам          C:\My_prog\C\Result\TEST2_A\nata.txt

ТЕСТ КОРРЕКТНОСТИ ФАЙЛОВ:
максимальное количество элементов :  0 - 38
максимальное количество связей :     0 - 44
ТЕСТ СООТВЕТСТВИЯ ЧИСЛА СОПРЯЖЕННЫХ И ВНЕШНИХ СВЯЗЕЙ:  OK

ТЕСТ ЧИСЛА СВЯЗЕЙ ПО СОПРЯЖЕННЫМ ЭЛЕМЕНТАМ:  OK

ТЕСТ ЧИСЛА СВЯЗЕЙ ПО ВНЕШНИМ ЭЛЕМЕНТАМ:  OK

ТЕСТ СООТВЕТСТВИЯ ВЫВОДОВ ПО СОПРЯЖЕННЫМ ЭЛЕМЕНТАМ:  OK

ТЕСТ СООТВЕТСТВИЯ ВЫВОДОВ ПО ВНЕШНИМ ЭЛЕМЕНТАМ:  OK

ТЕСТ СООТВЕТСТВИЯ ЯРУСНОГО ВРЕМЕНИ:
ТЕСТ СООТВЕТСТВИЯ ЯРУСНОГО ВРЕМЕНИ:  OK
ТЕСТ СООТВЕТСТВИЯ ЧИСЛА ВХОДОВ ЭЛЕМЕНТА И КОЛИЧЕСТВА ЕГО СОПРЯЖЕННЫХ:  OK
```

Рис.14. Результаты верификации корректности временной параллельной модели задачи

3.3. Пример синтеза временной параллельно - конвейерной модели Си-программы

Исходный текст Си-программы представляет рис. 15. Ограничение на длительность такта TT≤ 44 нс.

Длительности выполнения операций различных типов (в нс), используемые при синтезе параллельной временной модели, представляет табл. 4.

Графическая спецификация Си-программы представлена на рис. 16.

Результат автоматического синтеза параллельно-конвейерной модели с заданным значением такта показан на рис. 17.

Обозначения переменных имеют следующий смысл:

– «a_in», «b_in» – входы исходных данных;

– «a» , «b» – имена исходных данных;

– «clk» – вход тактового сигнала;

– переменные «var» с номерами 5, 7, 20, 4, 24 и их идентификаторами соответствуют вводимым в процессе синтеза конвейерной модели операторам памяти – фиксаторам данных (тип «=»), обеспечивающим передачу данных между конвейерными фрагментами (ступенями) модели.

```
include <stdio.h>
void main (void)
{
  int a, b;
  int z, s;
    scanf("%d", &a);
    scanf("%d", &b);
  if (a == b)
      { z = a * b;
       printf("%3d", z);
      }
 else
      { s = b / a;
       printf("%3d", s);
      }
}
```

Рис. 15. Исходная Си - программа

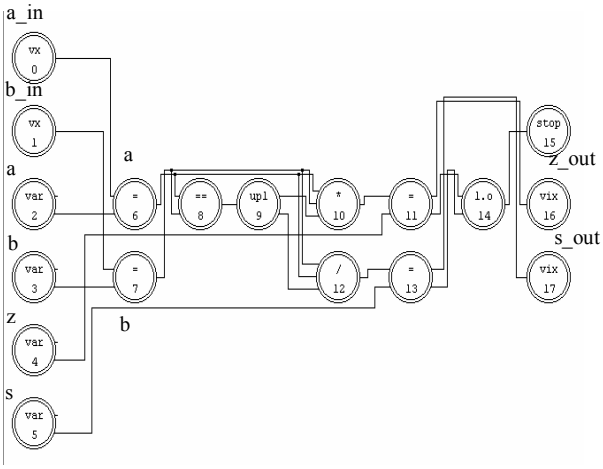

Рис. 16. Си – граф исходной Си – программы

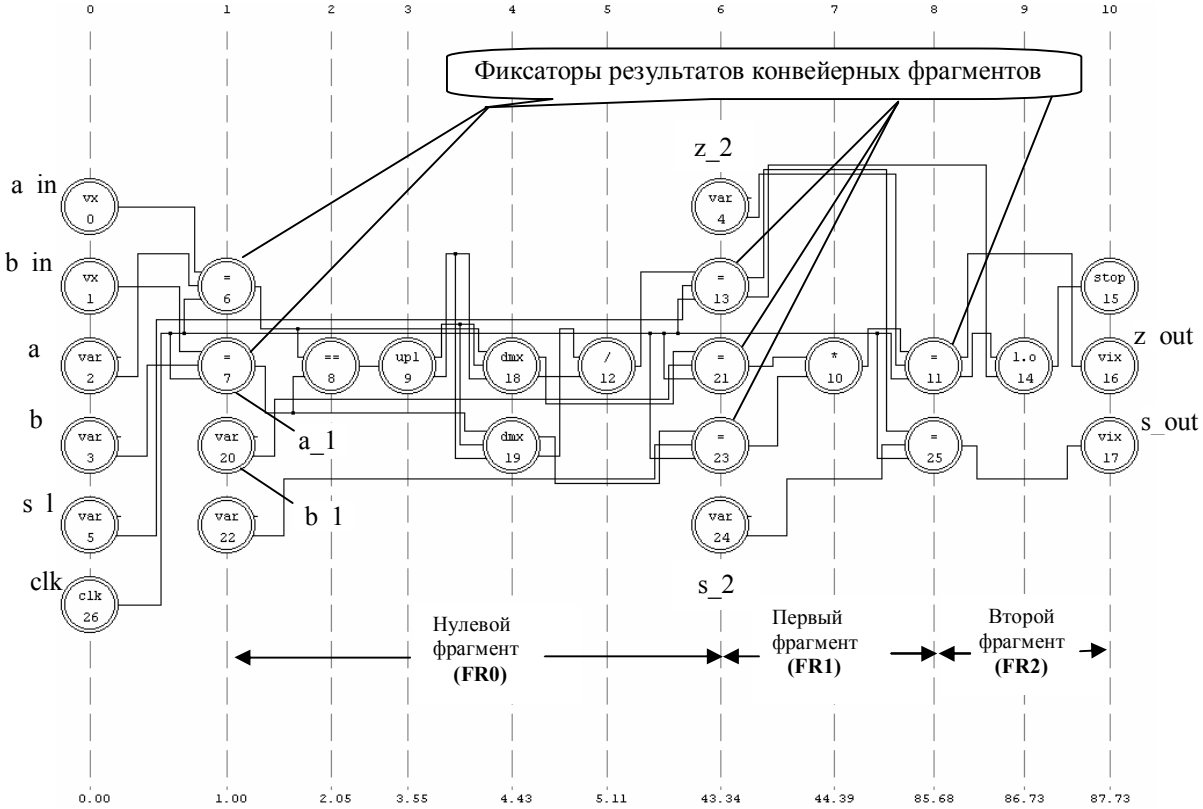

Рис. 17. Параллельно- конвейерная модель разветвляющейся Си – программы (TT = 42.34 нс)

Таблица 4

Длительности $t^o(typ_j)$ выполнения модулями элементной базы различных типов (typ_j) операторов P_j параллельно–конвейерной модели (нс)

typ	vx	var	=	==	upl	dmx	/	*	clk	l.o	vix	stop
t^o(нс)	1.0	1.0	1.05	1.5	0.88	0.68	11.0	41.29	1.0	1.00	1.0	1.0

4. Технология автоматического синтеза временных параллельных программ – Параллельный Интеллектуальный Компилятор, ПИК

Анализ существующих конструкций параллельных программ и известных систем автоматизации параллельного программирования показывает, их основными недостатками, ограничивающими возможности расширения областей применения и повышения эффективности существующих параллельных ВС, являются [11 – 14]:

– использование (заимствованного из последовательных программ фон Неймана) принципа ста-

тичности конструкции программы – отсутствия в программе средств задания (в явном или неявном виде) реального времени как определяющего параметра параллельных процессов;

– ручное выполнение программистом основных наиболее трудоемких этапов параллельного программирования.

Назначение теории и технологии ПИК [6, 9, 15–20, 24] – решение проблемы формального и автоматического синтеза временных моно- и мультипараллельных программ для перспективных Адаптивных Самоорганизующихся Вычислительных Систем, а также традиционных параллельных программ для известных классов параллельных процессоров (например, VLIW, FLOW/DFC – Data Flow Computers) и многопроцессорных ЭВМ (классов SMP, NUMA, MPP, CLUSTER).

Автором предложен новый класс последовательных/параллельных программ – Времяпараметризованные (Временные) Мультипараллельные Программы (ВМП – программы) [15]. Это конструкции, которые содержат в явном виде спецификации следующих категорий информации (рис. 18).

Принципиальными отличиями временных мультипараллельных программ от традиционных параллельных программ, применяемых в известных параллельных процессорах (например, с длинной командной строкой, VLIW, с управлением потоком данных, FLOW) и многопроцессорных ЭВМ (классов SMP, NUMA, MPP CLASTER), являются:

– **использование реального времени** в качестве одного из основных элементов конструкции параллельных программ и соответствующих параллельных процессов;

– **мультипараллелизм программ** – явное отражение в конструкциях программ состава фактически используемых (всех или произвольной части известных) методов параллельной обработки данных;

– **использование в программах единиц измерения (семантики)** обрабатываемых величин (кроме обработки только числовых значений данных, как это имеет место в традиционных последовательных и параллельных программах);

– **поддержка в явном виде** структурой временных мультипараллельных программ различных архитектур параллельных процессоров и ЭВМ;

– **поддержка в явном виде** структурой временных мультипараллельных программ требований и ограничений, задаваемых пользователями (например, обеспечение требуемого времени выполнения программы, заданной тактовой частоты обработки данных и т.д.);

– **поддержка принципа «временного управления на каждом такте»** параллельным вычислительным процессом (в отличие от принципа управления на основе «потока команд» программы и «потока данных» программы, принятых в известных параллельных процессорах и ЭВМ).

Отметим, что применяемые в настоящее время в компьютерной технике «статические» последовательные и параллельные программы являются частными случаями ВМП – программ.

Применение нового класса параллельных программ делает необходимым расширение классификации программ.

Соответствующие классификации по различным параметрам представляют рис. 19 – 21.

Архитектура технологии автоматического синтеза параллельных программных средств (Параллельного Интеллектуального Компилятора, ПИК) показана на рис. 22.

Рис. 18. Состав спецификаций времяпараметризованных (временных) мультипараллельных программ

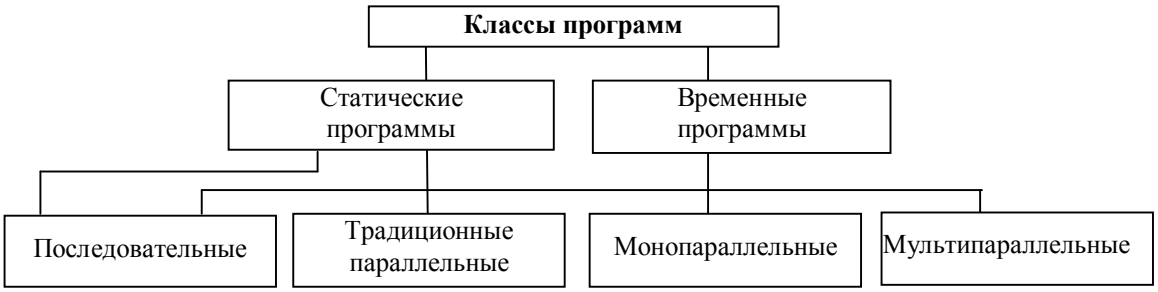

Рис. 19. Классификация программ по временному параметру и виду обработки данных

Рис. 20. Классификация временных параллельных программ на основе состава используемых методов параллельной обработки данных

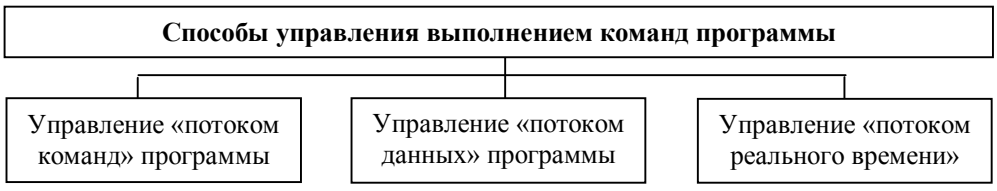

Рис. 21. Классификация временных параллельных программ по способу управления выполнением команд

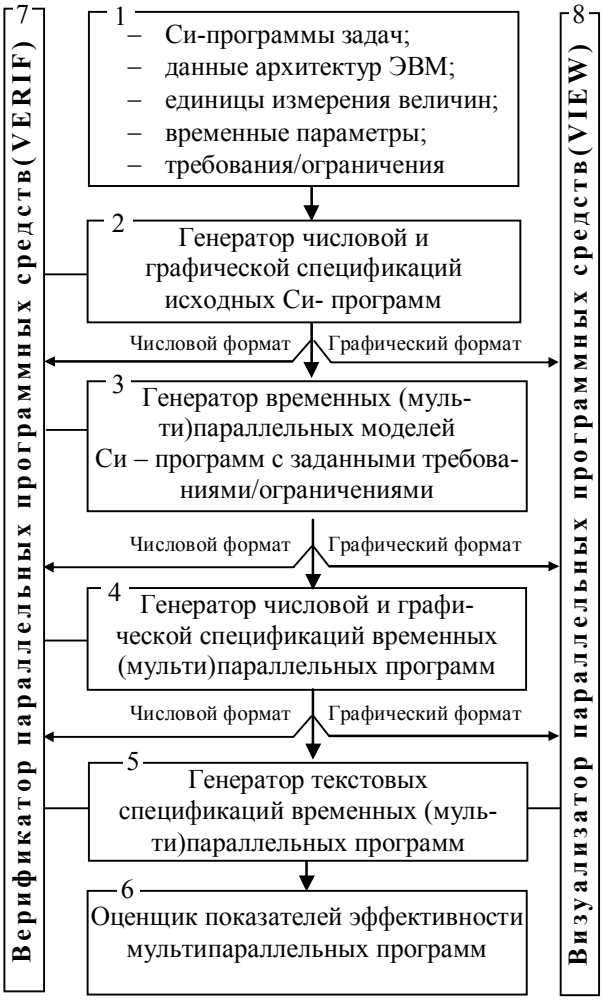

Рис. 22. Архитектура адаптивной технологии автоматического проектирования параллельного программного обеспечения (ПИК)

При проектировании временных мультипараллельных программ обеспечивается учет следующих групп факторов:

– **состав методов параллельной обработки данных**, который может быть использован при синтезе временных параллельных программ: совмещение независимых операций, конвейерный метод, кодово-матричный метод, декомпозиционный метод, метод параллельной смеси алгоритмов;

– **особенности архитектур параллельных процессоров и многопроцессорных ВС**, например, класс параллельной вычислительной системы (SMP, MPP, NUMA), тип процессоров, топологии межпроцессорных коммуникаций (полносвязная система, общая шина, гиперкуб и т.п.); иерархия памяти, возможность и время параллельного доступа к памяти и т.д.;

– **состав конкретной конфигурации параллельной ВС** (количество функциональных блоков/ процессоров, их связность и т.д.);

– **временные характеристики параллельных процессоров и многопроцессорных ВС**: длительности операций, тактовой частоты и т.п;

– **организация межпроцессорного обмена данными,** способов синхронизации работы процессоров, необходимых временных затрат и т.д;

– **система требований и ограничений** (на время решения, тактовую частоту, сложность/стоимость, надежность/достоверность).

Проиллюстрируем результаты применения технологии ПИК на примере синтеза временной параллельной программы для SMP ВС при использовании метода совмещения независимых операций.

4.1. Пример синтеза временной параллельной программы для ВС класса SMP

Исходный текст Си-программы задачи представляет рис. 23 (ЕЧ0… ЕЧ3 – «естественнные части» – неразветвляющиеся участки программы); используемый метод параллельной обработки – совмещение независимых операций; принимаем, что конфигурация ВС содержит NM=2 процессоров, каждый из которых одновременно может выполнять одну операцию; количество портов одновременного (параллельного) приема данных каждым процессором равно 3.

Длительности выполнения операций различных типов (в тактах), используемые при синтезе параллельной временной модели Си-программы, представляет табл. 5.

Результаты распределения данных Си–программы в общей разделяемой памяти SMP ВС показаны на рис. 24.

```
#include <stdio. h>
void main(void)
{
    int a, b, c, r, k, l, m, p;
     int s, t;
    scanf("%d %d %d\n",&a, &b, &c);
              k = a * b;
              l = b % a;                          } ЕЧ0
      if(k <  a - c)
          {
            m = (k % 2) * 2;
              r = l * 2;                          } ЕЧ1
              p = k + l;
          }
       else
          {
              p = 2 * l;                          } ЕЧ2
              r = l - k;
              m = p + l;
          }
        s = p - r;                                } ЕЧ3
         t = (m * 2) / a;
        printf("%4d %4d\n", s, t);
}
```

Рис. 23. Исходная Си-программа

Таблица 5

Длительность t_j^0 выполнения операций различных типов (такты)

typ	vx	+,-	=	*	%	/	vix	bp	stop	scanf	print
t_j^0	1.00	1.00	2.00	10.00	35.00	35.00	1.00	1.00	1.00	1.00	1.00

Адреса данных	Имена данных
0	a
1	b
2	c
3	a1
4	k
5	l
6	C2_
7	r
8	k1
9	p
10	m
11	m1
12	s
13	t

Рис. 24. Распределение данных в разделяемой памяти SMP ВС

Результаты автоматического синтеза временных программ – нитей 1-го и 2-го процессоров ВС показаны на рис. 25 и рис. 26. На рис. 27, табл. 6 и табл. 7 представлены графическая и семантико-числовая спецификации исходной Си-программы на уровне естественных частей. Как видно из рис. 27, задача имеет две возможные ветви решения: выполнение ЕЧ0, ЕЧ1, ЕЧ3, ЕЧ4 или ЕЧ0, ЕЧ2, ЕЧ3, ЕЧ4 с различными временами завершения задачи. Массивы NT (рис. 25 и рис. 26) показывают базовый вариант временных программ процессоров с максимальным временем решения.

NN	Команды 1-го процессора: oper	A1	A2	NT	add_NT
0	scanf	0	-1	1.00	-1
1	scanf	1	-1	1.00	-1
2	scanf	2	-1	1.00	-1
3	*	0	1	3.00	-1
4	=	4	-1	13.00	-1
5	<	4	3	15.00	-1
6	upl	7	12	16.00	-1
7	%	4	6	17.00	-1
8	=	8	-1	52.00	-1
9	*	8	6	54.00	-1
10	=	10	-1	64.00	-1
11	bp	18	-1	66.00	-1
12	*	6	5	17.00	-1
13	=	9	-1	27.00	-1
14	wait (14.00);			29.00	-1
15	+	9	5	43.00	-1
16	=	10	-1	44.00	-1
17	bp	18	-1	46.00	-1
18	*	10	6	67.00	0
19	=	11	-1	77.00	1
20	/	11	0	79.00	2
21	=	13	-1	114.00	3
22	print	13	-1	116.00	4
23	bp	24	-1	117.00	5
24	stop	-1	-1	118.00	6

Рис. 25. Временная программа – нить 1-го процессора SMP ВС

NN	Команды 2-го процессора: oper	A1	A2	NT	add_NT
0	scanf	0	-1	1.00	-1
1	scanf	1	-1	1.00	-1
2	scanf	2	-1	1.00	-1
3	-	0	2	3.00	-1
4	=	3	-1	4.00	-1
5	%	1	0	6.00	-1
6	=	5	-1	41.00	-1
7	<	4	3	43.00	-1
8	upl	9	14	44.00	-1
9	*	5	6	45.00	-1
10	=	7	-1	55.00	-1
11	+	4	5	57.00	-1
12	=	9	-1	58.00	-1
13	bp	17	-1	60.00	-1
14	-	5	4	45.00	-1
15	=	7	-1	46.00	-1
16	bp	17	-1	48.00	-1
17	-	9	7	61.00	0
18	=	12	-1	62.00	1
19	print	12	-1	64.00	2
20	bp	21	-1	65.00	3
21	stop	-1	-1	66.00	4

Рис. 26. Временная программа – нить 2-го процессора SMP ВС

Таблица 6

Базовая структура BFE естественных частей Си-программы

NN	MET	TYP	NSJ	SJD	NWJ	WJD	MP1	MP2	BEG	END
0	0	2	-1	0	0	2	1	2	0	22
1	1	0	0	1	2	1	3	0	23	31
2	2	0	1	1	3	1	3	0	32	38
3	3	0	2	2	4	1	980	0	39	48
4	980	3	4	1	-1	0	0	0	49	51

Таблица 7

Структура CFE связей естественных частей Си-программы

NN	JSD	SPJD	JWD	WPJD
0	-1	0	1	1
1	-1	0	-1	2
2	3	1	-1	3
3	-1	2	-1	3
4	-1	3	-1	4

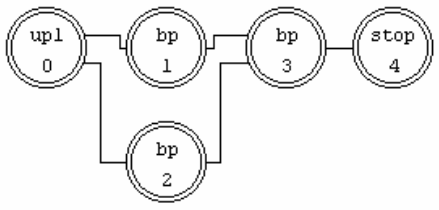

Рис. 27. Граф исходной программы на уровне естественных частей (ЕЧ) Си - программы

Остальные моменты начала выполнения операторов временных программ процессоров задаются в цепочках дополнительных параметров начала каждого оператора (рис. 28). На рис. 28 приняты следующие обозначения:

NN – номера строк, соответствующие значениям указателей add_NT ≥ 0 на рис. 25 и рис. 26 (add_NT = -1 – признак отсутствия у оператора цепочки дополнительных параметров начала),

NT_add – дополнительное время начала,

j_NT$\neq$1 - указатель на продолжение цепочки параметров начала (j_NT = - 1 – признак конца цепочки),

NUM_EC – номера естественных частей Си-программы, при выполнении которых используется конкретный оператор.

```
=====================================
||  1 - й процессор   ||             ||
=====================||  NUM_EC     ||
||  NN  j_NT  NT_add  ||             ||
=====================================
||  0   -1    47.00   ||  0  2  3    ||
||  1   -1    57.00   ||  0  2  3    ||
||  2   -1    59.00   ||  0  2  3    ||
||  3   -1    94.00   ||  0  2  3    ||
||  4   -1    96.00   ||  0  2  3    ||
||  5   -1    97.00   ||  0  2  3    ||
||  6   -1    98.00   ||  0  2  3  4 ||
=====================================

=====================================
||  2 - й процессор   ||
=====================||  NUM_EC
||  NN   j_NT  NT_add ||
=====================================
||  0     -1   49.00   ||  0  2  3    ||
||  1     -1   50.00   ||  0  2  3    ||
||  2     -1   52.00   ||  0  2  3    ||
||  3     -1   53.00   ||  0  2  3    ||
||  4     -1   54.00   ||  0  2  3  4 ||
=====================================
```

Рис. 28. Цепочки значений времени начала операторов параллельной SMP - программы

Рис. 29 и рис. 30 представляют результаты автоматического синтеза Си-текстов, соответствующих временным программам – нитям каждого из процессоров SMP ВС (рис. 25 и рис. 26).

На рис. 31 представлен результат автоматического синтеза традиционной (статической) параллельной SMP-программы (NM = 2).

5. Технология автоматического синтеза мультипараллельных цифровых аппаратных средств – Автоматический Интеллектуальный Проектировщик, АИП

Анализ известных систем автоматизированного проектирования цифровых устройств показывает, что концепцией построения существующих САПР является выполнение человеком наиболее сложных, неформализованных, творческих этапов проектирования, определяющих качество аппаратных средств, сложность, сроки и стоимость проектирования. Это обусловило проблемы САПР, не имеющие до настоящего времени удовлетворительного решения [20 – 22]:

– разрыв между сложностью СБИС, которые может производить электронная индустрия, и предельной сложностью проектов СБИС, поддерживаемых известными автоматизированными системами проектирования (проблема SOC);

– неспособность САПР существенно сократить сроки проектирования аппаратных средств;

– неспособность САПР использовать все известные методы параллельной обработки данных с автоматической оптимизацией их состава для конкретных задач, требований и ограничений.

```
#include <stdio.h>
void main(void)
{
int a,b,c,r, k,l,m,p, s,t,  a1,k1,m1;
            scanf("%d",&a);
            scanf("%d",&b);
            scanf("%d",&c);
            к = a * b;
            if (k < a1)
            {           k1 = к % 2;
                        m = k1 * 2;}
            else
            {           p = 2 * l;
                        wait (14.00);
                        m = p + 1;}
            m1 = m * 2;
            t = m1 / a;
            printf(" %3d ",t);
}
```

Рис. 29. Синтезированная Си – программа 1-го процессора

```
#include <stdio.h>
void main(void)
{
int a,b,c,r;
int k,l,m,p;
int s,t;
int a1,k1,m1;
        scanf("%d",&a) ;
        scanf("%d",&b) ;
        scanf("%d",&c) ;
        a1 = a - c;
        l = b % a;`
        if (k < a1)
              { r=l * 2;
              p=k + l;}
        else
              r=l - k;
        s = p-r;
        printf(" %3d ",s) ;
}
```

Рис. 30. Синтезированная Си – программа 2-го процессора

```
#include-<stdio.h>
void main(void)
int a,b,c,r, k,l,m,p, s,t, a1,k1,m1;
        PAR;
        scanf("%d",&a);
        scanf("%d",&b);
        scanf ("%d",&c);
        END PAR;
        PAR;
        a1 = a - c;
        k = a * b;
        END PAR;
        l = b % a;
        if (k < a1)
                {PAR;
                r = l * 2;
                k1 = k %  2 ;
                END PAR;
                PAR;
                p = k + l;
                m = k1 * 2;
                END PAR;}
        else
                {PAR;
                r = l - k;
                p = 2 * l;
                END PAR;
                m  =  p  *  l ; }
        PAR;
        m1 = m *2;
        s = p - r;
        END PAR;
        printf(" %3d ",s);
        t = m1 / a;
        printf(" %3d ",t);
}
```

Рис. 31. Результат автоматического синтеза традиционной (статической) параллельной SMP – программы (NM = 2)

Назначение теории и технологии АИП [6, 9, 22, 23] – решение проблемы формального и автоматического синтеза временных моно- и мультипараллельных цифровых аппаратных средств для перспективных Адаптивных Самоорганизующихся Вычислительных Систем, а также для универсальных и специализированных параллельных процессоров и многопроцессорных ЭВМ известных классов.

Поддержка технологией АИП всех известных методов параллельной обработки данных приводит к более общей, по сравнению с известной, классификации параллельных аппаратных средств (рис. 32).

Состав типов аппаратных средств, автоматический синтез которых поддерживает АИП:

– типовые функциональные модули (ФМ) цифровой вычислительной техники с мультипараллельной обработкой данных (например, сумматоры, умножители, делители и т.п.), удовлетворяющие заданным требованиям/ ограничениям;

– специализированные мультипараллельные процессоры с неперестраиваемой (жесткой) архитектурой (например, спецпроцессоры цифровой фильтрации, ЦФ; быстрого преобразования Фурье, БПФ и т.п.), предназначенные для решения конкретных задач (с выполнением конкретной системы требований и ограничений);

– специализированные перестраиваемые процессоры с мультипараллельной обработкой данных и программным или аппаратным управлением, обеспечивающие реализацию конкретного множества пользовательских задач (с выполнением требований и ограничений, соответствующих различным областям применения и задачам);

Рис. 32. Классификация аппаратных средств по методам параллельной обработки данных

– мультипараллельные аппаратно реализованные процессоры и системы универсального назначения с самоорганизацией архитектуры применительно к различным задачам и различным требованиям/ ограничениям.

Исходные данные синтеза мультипараллельных аппаратных средств:

– задачи/алгоритмы, представленные исходными текстами Си-программ;

– состав известных методов параллельной обработки данных (рис. 5);

– состав и характеристики элементной базы/библиотеки цифровых аппаратных средств;

– система требований и ограничений (время решения задачи, тактовая частота обработки данных, производительность, надежность / достоверность, сложность/стоимость);

– состав уровней проектирования: архитектурный, функциональный, логический и вентильный.

Архитектура технологии автоматического синтеза параллельных средств (Автоматического Интеллектуального Проектировщика, АИП) показана на рис. 33.

Проиллюстрируем технологию АИП на конкретных примерах синтеза параллельных цифровых устройств: устройства с совмещением независимых операций (глобальный параллелизм) и параллельного устройства, поддерживающего глобальный параллелизм и метод конвейерной обработки.

5.1. Пример автоматического синтеза параллельного спецпроцессора с совмещением независимых операций

Си-программа задачи представлена на рис. 34.

Поясним содержание основных этапов автоматического синтеза параллельных аппаратных средств на примере синтеза функциональной схемы цифрового устройства с совмещением операций при отсутствии ограничений на сложность устройства.

Результаты автоматического синтеза структур семантико-числовой спецификации программы представлены в табл. 8 и 9.

Результат верификации структур [24] СЧС и Си-графа представляет рис. 35.

Структуры СЧС BFE и CFE, представляющие Си-программу на уровне естественных частей (ЕЧ), показаны в табл. 10 и табл. 11, ЕЧ – граф программы – на рис. 36.

Результаты автоматического синтеза графической спецификации Си программы в виде операционного Си-графа показаны на рис. 37.

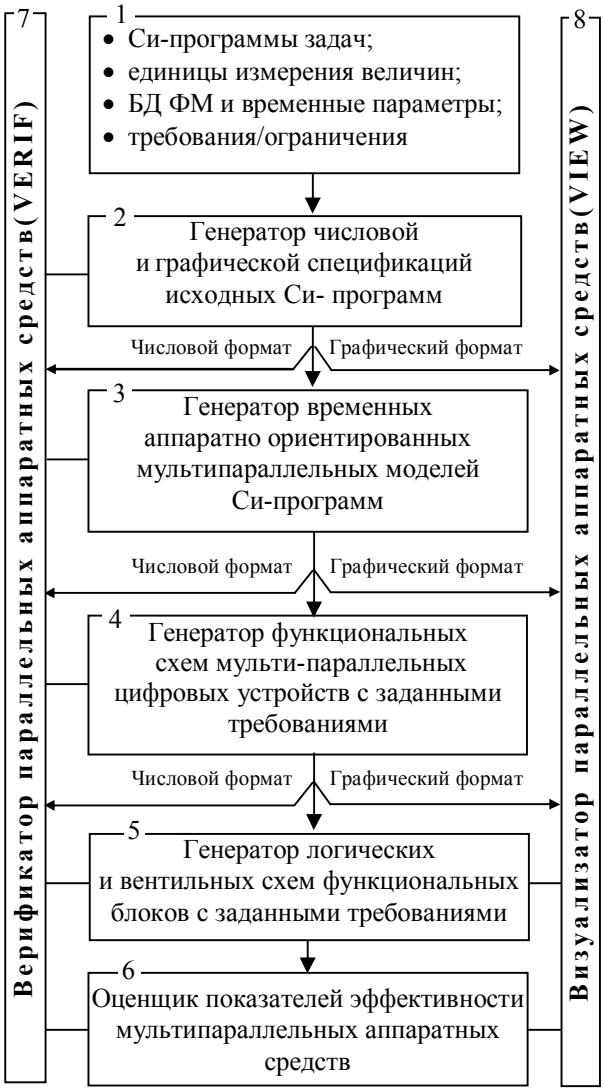

Рис. 33. Архитектура адаптивной технологии автоматического проектирования мультипараллельных аппаратных средств (АИП)

```
#include <stdio.h>
void main(void)
{
    int a,b,  k,z,p,s;
    scanf("%d %d %d %d",&a,&b);
  if(a == b)
   {
     k = a % 2;
     z = a * b;
        printf("%4d\n",k);
     printf("%4d\n",z);
   }
  else
   {
     p = a | b;
     s = b / a ;
        printf("%4d\n",p);
     printf("%4d\n",s);
   }
}
```

Рис. 34. Исходная Си-программа

Таблица 8

Базовая структура **BF** Си – программы

N	MET	TYP	NSJ	SJD	BJ	NWJ	WJD	MP1	MP2	VH	VIH	REZ
0	0	58	-1	0	0	0	1	e	0	0	1	a_in
1	0	58	-1	0	0	1	1	0	0	0	1	b_in
2	0	47	-1	0	0	2	1	0	0	0	2	a
3	0	47	-1	0	0	3	1	0	0	0	2	b
4	0	47	-1	0	0	4	1	0	0	0	2	k
5	0	47	-1	0	0	5	1	0	0	0	2	z
6	0	47	-1	0	0	6	1	0	0	0	2	P
7	0	47	-1	0	0	7	1	0	0	0	2	s
8	0	12	0	2	0	8	5	0	0	2	1	=
9	0	12	2	2	0	3	4	0	0	2	1	=
10	0	23	4	2	0	17	1	0	0	2	1	==
11	0	51	6	1	0	18	4	1	2	1	2	upl
12	0	57	-1	0	1	22	1	0	0	0	1	C2
13	1	5	7	3	1	23	1	0	0	3	1	%
14	0	12	10	2	1	24	2	0	0	2	2	=
15	0	3	12	3	1	26	1	0	0	3	1	*
16	0	12	15	2	1	27	2	0	0	2	2	=
17	0	50	17	2	1	29	1	3	0	2	1	bp
18	2	34	19	3	2	30	1	0	0	3	1	|
19	0	12	22	2	2	31	2	0	0	2	2	=
20	0	4	24	3	2	33	1	0	0	3	1	/
21	0	12	27	2	2	34	2	0	0	2	2	=
22	0	50	29	2	2	36	1	3	0	2	1	bp
23	3	54	31	2	3	37	1	0	0	2	1	1.0
24	0	49	33	1	3	-1	0	0	0	1	0	stop
25	0	48	34	1	3	-1	0	0	0	1	0	k_out
26	0	48	35	1	3	-1	0	0	0	1	0	z_out
27	0	48	36	1	3	-1	0	0	0	1	0	p_out
28	0	48	37	1	3	-1	0	0	0	1	0	s out

```
Файл элементов:                    C:\My_prog\C\Result\ZS_P2\VIXVIX1.TXT
Файл связей элементов:             C:\My_prog\C\Result\ZS_P2\VIXVIX2.TXT


ТЕСТ КОРРЕКТНОСТИ ФАЙЛОВ:
максимальное количество элементов :  0 - 28
максимальное количество связей :     0 - 37
ТЕСТ СООТВЕТСТВИЯ ЧИСЛА СОПРЯЖЕННЫХ И ВНЕШНИХ СВЯЗЕЙ:  OK

ТЕСТ ЧИСЛА СВЯЗЕЙ ПО СОПРЯЖЕННЫМ ЭЛЕМЕНТАМ:  OK

ТЕСТ ЧИСЛА СВЯЗЕЙ ПО ВНЕШНИМ ЭЛЕМЕНТАМ:  OK

ТЕСТ СООТВЕТСТВИЯ ВЫВОДОВ ПО СОПРЯЖЕННЫМ ЭЛЕМЕНТАМ:  OK

ТЕСТ СООТВЕТСТВИЯ ВЫВОДОВ ПО ВНЕШНИМ ЭЛЕМЕНТАМ:  OK

ТЕСТ СООТВЕТСТВИЯ ЧИСЛА ВХОДОВ ЭЛЕМЕНТА И КОЛИЧЕСТВА ЕГО СОПРЯЖЕННЫХ:  OK
```

Рис. 35. Результаты верификации структур СЧС и Си-графа исходной Си-программы

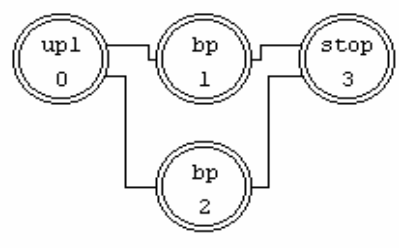

Рис. 36. ЕЧ – граф исходной Си-программы

Таблица 9

Структура связей **CF** Си – программы

N	JSD	SPJD	SNWIH	SNWHO	TSS	JWD	WPJD	WNWHO	WNWIH	TVS
0	1	0	0	0	0	-1	8	0	0	0
1	-1	2	1	1	2	-1	9	0	0	0
2	3	1	0	0	0	-1	8	1	1	2
3	-1	3	1	1	2	-1	9	1	1	2
4	5	8	0	0	0	-1	14	1	1	2
5	-1	9	0	1	0	-1	16	1	1	2
6	-1	10	0	0	0	-1	19	1	1	2
7	8	8	0	0	0	-1	21	1	1	2
8	9	12	0	1	0	9	10	0	0	0
9	-1	11	0	2	1	10	13	0	0	0
10	11	4	1	1	2	11	15	0	0	0
11	-1	13	0	0	0	12	18	0	0	0
12	13	8	0	0	0	-1	20	1	0	0
13	14	9	0	1	0	14	10	1	0	0
14	-1	11	0	2	1	15	15	1	0	0
15	16	5	1	1	2	16	18	1	0	0
16	-1	15	0	0	0	-1	20	0	0	0
17	18	16	1	0	1	-1	11	0	0	0
18	-1	14	1	1	1	19	13	2	0	1
19	20	8	0	0	0	20	15	2	0	1
20	21	9	0	1	0	21	18	2	1	1
21	-1	11	1	2	1	-1	20	2	1	1
22	23	6	1	1	2	-1	13	1	0	0
23	-1	18	0	0	0	-1	14	0	0	0
24	25	9	0	0	0	25	17	1	1	1
25	26	8	0	1	0	-1	25	0	0	0
26	-1	11	1	2	1	-1	16	0	0	0
27	28	7	1	1	2	28	17	0	1	1
28	-1	20	0	0	0	-1	26	0	0	0
29	30	21	1	0	1	-1	23	0	0	1
30	-1	19	1	1	1	-1	19	0	0	0
31	32	22	0	1	1	32	22	1	1	1
32	-1	17	0	0	1	-1	27	0	0	0
33	-1	23	0	0	1	-1	21	0	0	0
34	-1	14	0	0	0	35	22	0	1	1
35	-1	16	0	0	0	-1	28	0	0	0
36	-1	19	0	0	0	-1	23	1	0	1
37	-1	21	0	0	0	-1	24	0	0	1

Таблица 10

Базовая структура **BFE**

NN	MET	TYP	NSJ	SJD	NWJ	WJD	MP1	MP2	BEG	END
0	0	2	-1	0	0	2	1	2	0	11
1	1	0	0	1	2	1	3	0	12	17
2	2	0	1	1	3	1	3	0	18	22
3	3	3	2	2	-1	0	0	0	23	28

Таблица 11

Структура связей **CFE**

NN	JSD	SPJD	JWD	WPJD
0	-1	0	1	1
1	-1	0	-1	2
2	3	1	-1	3
3	-1	2	-1	3

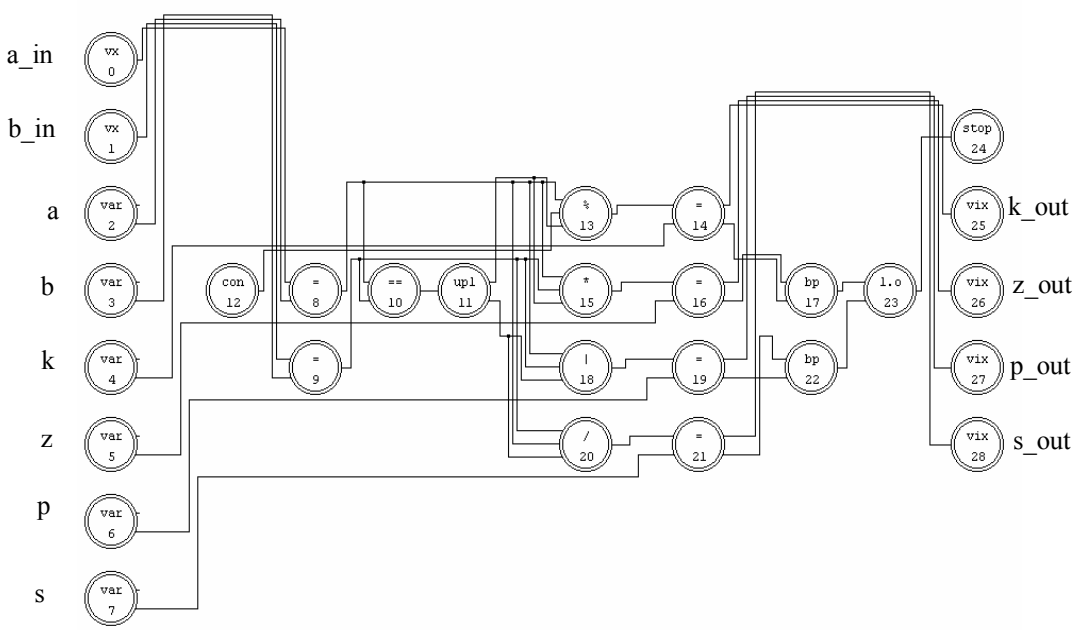

Рис. 37. Графическая спецификация исходной разветвляющейся Си-программы

Результат синтеза функциональной схемы параллельно- конвейерного спецпроцессора представляет рис. 38. В спецпроцессор входят две функционально различные части**:**

– **исполнительное устройство,** содержащее функциональные блоки с номерами 8 …30 (при этом входной интерфейс схемы представляют вход «vx0» (a_in) и вход «vx1» (b_in), выходной интерфейс – представляется выходами «vix25» (k_out), «vix26» (z_out), «vix27» (p_out) и «vix28» (s_out), выход «stop» обеспечивает выдачу признака завершения решения задачи);

– устройство управления, в состав которого входят вход «On»31 сигнала синхронизации, функциональные блоки временной задержки Z_{33}, Z_{34}, Z_{35}, Z_{36} и элемент «ИЛИ» 32.

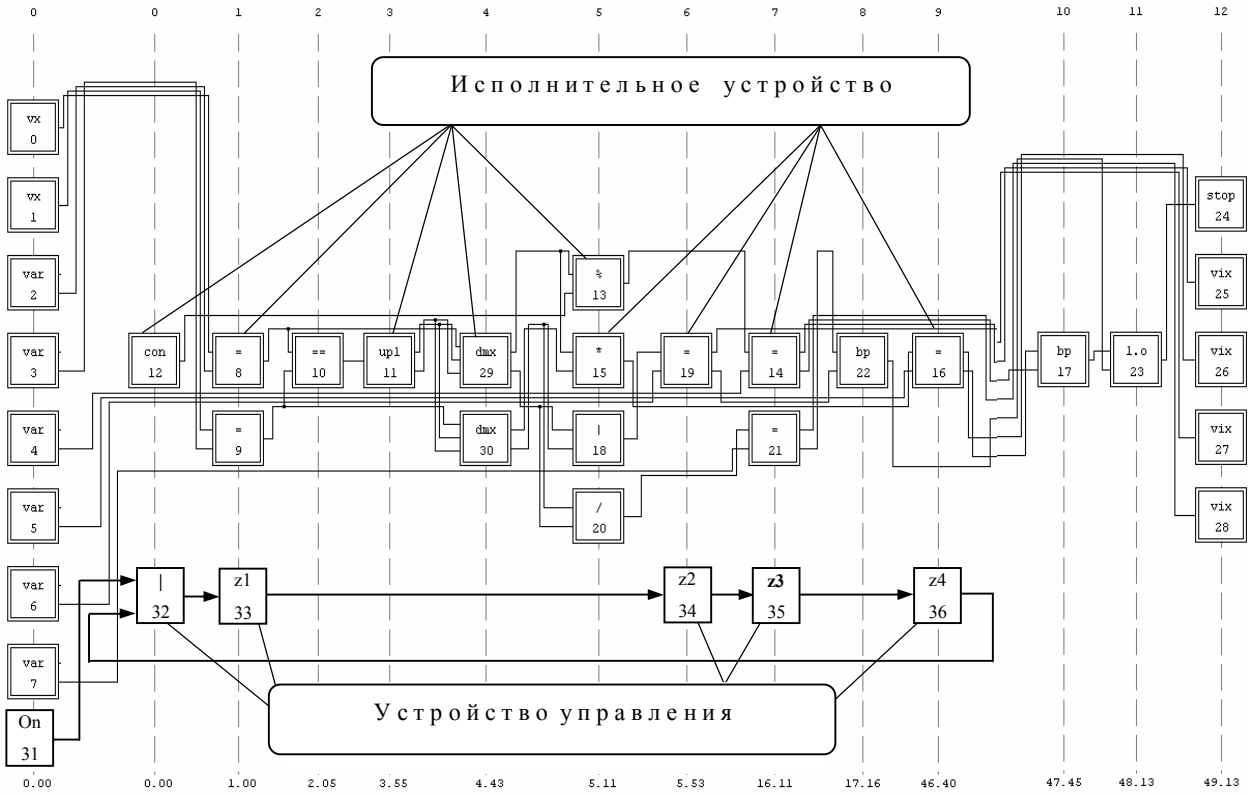

Рис. 38. Функциональная схема параллельного устройства с совмещением независимых операций

Конкретные функциональные блоки устройства выполняют следующие функции:

– **блоки памяти** – в примере это регистры (тип выполняемой операции «=») RG8, RG9, RG19, RG14, RG21, RG16;

– **компаратор** (модуль проверки равенства «==») CMP10 обеспечивающий совместно с модулем UPL11 управление разветвлением вычислительного процесса;

– **блоки арифметических операций** «%» MOD13, «*» MUL15 , «|» OR18, «/» DIV20;

– **блоки коммутации** (**демультиплексоры**) DMX29, DMX30, управляемые блоком UPL11 и обеспечивающие активизацию вычислений по одному из двух возможных дальнейших маршрутов**:** или выполнение операций «%» и «*», либо операций «|» и «/»; блоки типа «bp» BP22, BP17 обеспечивают передачу управления (после завершения реализации фрагментов задачи, связанных с разветвлением) на финальную часть задачи;

– **блок «l.o»**, формирующий признак завершения процесса решения задачи.

Длительности операций представлены в табл. 12.

Таблица 12

Длительность «t_j^0» выполнения операторов типа «typ» параллельной временной модели (нс) с совмещением независимых операций

typ	t_j^0	typ	t_j^0
*	41.29	l.o	1.00
*,=	1.05	dmx	0.68
|	0.42	%	11.00
vx	1.00	/	11.00
stop	1.00	vih	1.00
bp	0.68	==	1.50
upl	0.88		

5.2. Пример автоматического синтеза параллельно-конвейерного спецпроцессора

Рассмотрим задачу, Си-программа которой представлена рис. 39.

Примем, что длительности выполнения операций заданы табл. 12 (а длительность оператора «clk» равна 1.00 нс). Требование к длительности такта конвейера TT ≤ 44 нс, ограничения на сложность устройства отсутствуют.

Примем также, что в связи с подробным рассмотрением (для Си-программы предшествующего примера) содержания большей части этапов проектирования, будут комментироваться только конечные результаты автоматического синтеза – функциональная схема спецпроцессора с совмещением операций и конвейерной обработкой и ее временная параллельная модель.

Синтезированную функциональную схему параллельно-конвейерного спецпроцессора представляет рис. 40. Устройство имеет TT = 42.34 нс и содержит две функциональные части:

– **исполнительное устройство** (функциональные модули с номерами 8 …16, 18…21, 29,30, входной интерфейс схемы представляют вход «vx0» (a_in) и вход «vx1» (b_in), выходной интерфейс представляется выходами «vix25» (k_out), «vix26» (z_out), «vix27» (p_out) и «vix28» (s_out), выход «stop»24 обеспечивает выдачу признака завершения решения задачи);

– **устройство управления**, в состав которого входит опера-тор P_{41} ввода тактового сигнала «clk», задающего моменты передачи значений данных между конвейерными фрагментами путем синхронизации срабатывания памяти (фиксаторов).

```
#include <stdio.h>
void main(void)
{
    int a,b;
    int k,z,p,s;
    scanf("%d%d%d%d",&a,&b);
   if(a == b)
    {
     k = a % 2;
     z = a * b;
          printf("%4d\n",k);
     printf("%4d\n",z);
    }
  else
    {
     p = a | b;
     s = b / a ;
          printf("%4d\n",p);
     printf("%4d\n",s);
    }
}
```

Рис. 39. Исходная Си-программа

Аппаратные блоки устройства имеют следующую функциональность:

– блоки памяти – регистры (тип выполняемой операции «=») исходных данных RG8, RG9, CON12 нулевой ступени конвейера, регистры – фиксаторы RG14, RG19, RG21, RG32, RG34 первой ступени конвейера FR1, регистры – фиксаторы RG16, RG36, RG38, RG40 второй ступени конвейера FR2;

– компаратор 10 (блок проверки равенства «==»), обеспечивающий совместно с блоком UPL11 управление разветвлением вычислительного процесса;

– функциональные блоки коммутации (демультиплексоры) DMX29, DMX30, управляемые

модулем UPL11 и обеспечивающие активизацию вычислений по одному из двух возможных дальнейших маршрутов: или выполнение операций «%» и «*», либо операций «|» и «/»;

– модули BP22, BP17 типа «bp» обеспечивают передачу управления на выходную часть после завершения реализации фрагментов задачи.

Для наглядности функциональная схема синтезированного параллельно - конвейерного устройства объединена с временной диаграммой работы.

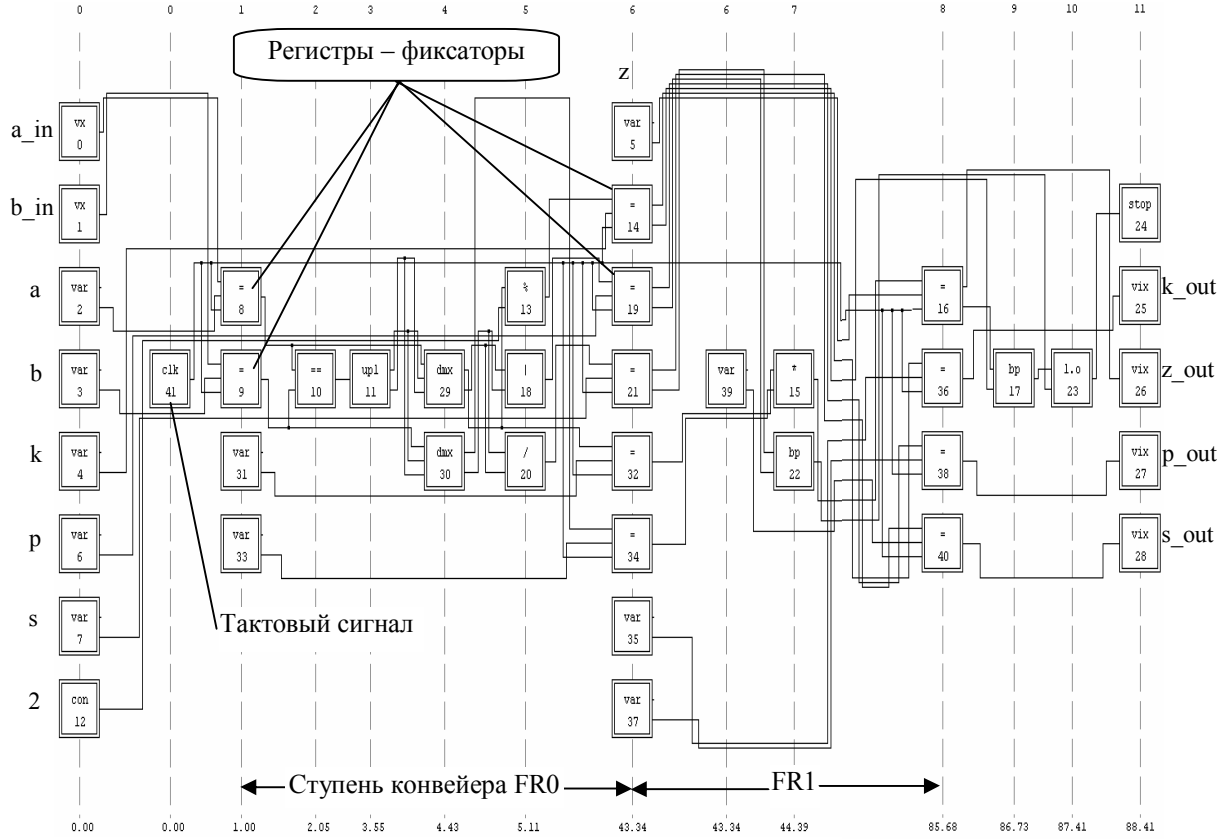

Рис. 40. Функциональная схема параллельно-конвейерного спецпроцессора

Выводы

1. Научно-технический прогресс тесно связан с развитием критических технологий и систем. Характерным для них является широкий состав подлежащих учету факторов, быстротечность процессов и жесткие требования к устойчивости систем и достоверности выходной информации. Одним из перспективных путей создания гарантоспособных систем, способных обеспечить выполнение таких требований, является разработка Адаптивных Самоорганизующихся Вычислительных Систем.

2. В процессе создания теории и методов синтеза Адаптивных Самоорганизующихся Вычислительных Систем решены следующие проблемы:

– разработан аппарат структурно-временной дискретной математики, обеспечивающий формальную спецификацию объектов и этапов проектирования аппаратного и программного обеспечения параллельных вычислительных систем;

– обоснованы методы формального синтеза мультипараллельных аппаратных и программных средств параллельных вычислительных систем;

– созданы адаптивные технологии, обеспечивающие возможность автоматического проектирования временных параллельных (аппаратно и/или программно ориентированных) моделей задач, временных параллельных программ и параллельных аппаратных средств, верификацию и визуализацию результатов и оценку их показателей эффективности.

3. Разработанные аппарат структурно-временной дискретной математики, структурно-временная теория формального синтеза параллельного аппаратного и программного обеспечения, технологии автоматического проектирования и проведенные эксперименты обеспечивают возможность перехода к практической разработке Адаптивных Самоорганизующихся Вычислительных Систем и развития исследований по созданию перспективных Интеллектуально-Самоорганизующихся Вычислительных Систем.

Литература

1. Проблемы многоверсионного проектирования высоконадежных параллельных программных средств для систем управления критическими технологиями и объектами / Г.А. Поляков, В.В. Скляр, Д.А. Толстолужский и др. // Радіоелектронні і комп'ютерні системи. – 2006. -№7(19). – С. 7-16.

2. TOP500 Supercomputer sites [Электронный ресурс]. – Режим доступа к ресурсу: http://www.top500.org.

3. Goldin Daniel S. Beyond Incremental Change / Daniel S. Goldin, Samuel L. Venneri, Ahmed K. Noor // Computer. – 1998. – Vol. 31, № 10. – P. 31-39.

4. Голдин Д.С. На повестке дня – революция / Дэниел С. Голдин, Сэмюель Л. Венери, Ахмед К. Нур // Открытые системы. – 2000. – № 1-2. – С. 66-71.

5. Хокинс Дж. Об интеллекте / Дж. Хокинс, С. Блейксли. – М.: Издательский дом "Вильямс", 2007. – 240 с.

6. Поляков Г.А. Основы построения и автоматического проектирования самоорганизующихся систем параллельной цифровой обработки информации и повышение эффективности комплексов радиолокационного вооружения ПВО / Г.А. Поляков. – Х: ВИРТА ПВО, 1986. – 572 с.

7. Состояние и основные направления развития высокопроизводительных вычислительных средств / Е.Г. Волокитина, Н.В. Матчина, В.В. Онищенко и др.; под ред. Г.А. Полякова. – Х.: ХВУ, НЦ РКИ, НТЦ НКАУ, 1994. – 306 с.

8. Поляков Г.А. Адаптивные самоорганизующиеся системы с мультипараллельной обработкой данных – стратегия развития цифровой вычислительной техники в XXI-м веке / Г.А. Поляков // Прикладная радиоэлектроника. – 2002. – Том 1, № 1. – С. 57 –69.

9.Поляков Г.А. Автоматизация проектирования сложных цифровых систем коммутации и управления / Г.А. Поляков, Ю.Д. Умрихин. – М.: Радио и связь, 1988. – 304 с.

10. Поляков Г.А. Алгебра кодовых матриц – аппарат анализа и синтеза сложных цифровых автоматов / Г.А. Поляков // Радиотехника: Всеукраинский межвед. науч.-техн. сб. – Вып. 72. – Х.: Вища шк.: Изд-во при харьк. ун-те. – 1985. – С. 94-104.

11. Chandy Mani K. Parallel Program Design: A Foundation / K. Mani Chandy, Jayadev Misra. – Massachusetts: Addison Wesley Publishing Company, Inc., 1988. – 544 p.

12. Воеводин В.В. Параллельные вычисления / В.В. Воеводин, Вл.В. Воеводин. – СПб.: БХВ Петербург, 2002. – 608 с.

13. Лацис А. Как построить и использовать суперкомпьютер / А. Лацис. – М.: Бестселлер, 2003. – 240 с.

14. Немнюгин С. Параллельное программирование для многопроцессорных вычислительных систем / С.А. Немнюгин, О.Л. Стесик. – СПб.: БХВ-Петербург, 2002. – 400 с.

15. Поляков Г.А. Глобально-параллельные времяпараметризованные программы – новый подход к синтезу структур и выполнению параллельных программ АСУ реального времени / Г.А. Поляков // Программное обеспечение вычислительных сетей и систем реального времени: тез. докл. – Всесоюз. конф. 15-17 декабря 1981 г. – К., 1981. – С. 130-132.

16. Поляков Г.А. Формальный синтез параллельных программ для высокопроизводительных VLIW – процессоров / Г.А. Поляков, Е.Г. Толстолужская // Системи обробки інформації: збірник наукових праць ХУПС ім. І. Кожедуба. – Х.: ХУ ПС, 2007. – Вип. 8(66). – С. 72-80.

17. Поляков Г.А. Метод формального архитектурно-ориентированного проектирования временных параллельных программ для ЭВМ с симметричной мультипроцессорной обработкой данных / Г.А. Поляков, Е.Г. Толстолужская // Збірник наукових праць Харківського університету Повітряних Сил ім. І. Кожедуба. – Х.: ХУ ПС, 2008. – Вип. 3(18). – С. 118-121.

18. Поляков Г.А. Метод синтеза Си-программ смеси алгоритмов / Г.А. Поляков, Д.А. Толстолужский, Е.Г. Толстолужская // Системи обробки інформації: збірник наукових праць ХУПС ім. І. Кожедуба. – Х.: ХУ ПС, 2008. – Вип. 1(68). – С. 96-100.

19.Поляков Г.А. Технология проектирования времяпараметризованных мультипараллельных программ как стратегия развития систем параллельного проектирования / Г.А. Поляков, Е.Г. Толстолужская // Радіоелектронні і комп'ютерні системи. – 2009. – №6(40). – С. 166-171.

20. Норенков И.П. Основы автоматизированного проектирования: учебник для вузов / И.П. Норенков. – М.: МГТУ им. Н.Э. Баумана, 2002. – 336 с.

21. Максфилд К. Проектирование на ПЛИС. Курс молодого бойца / К. Максфилд. – М.: Издательский дом «Додэка –XX1», 2007. – 408 с.

22. Поляков Г.А. Проблемы создания систем совместного автоматического проектирования аппаратно-программных средств для мультипараллельной цифровой обработки данных / Г.А. Поляков // 1-й Международный радиоэлектронный Форум «Прикладная радиоэлектроника. Состояние и перспективы развития» МРФ – 2002. Часть 2. – Х.: АН ПРЭ, ХНУРЭ, 2002. – С. 241-244.

23. Polyakov G. The hard-and-soft Automatic Design of Self-Organizing Adaptive Systems / G. Polyakov // Радіоелектроніка та інформатика. – 2003. – № 3. – С. 149.

24. Поляков Г.А. Компиляционная методика верификации статических и динамических объектов автоматического проектирования мультипараллельных цифровых устройств/ Г.А. Поляков, Д.А. Толстолужский // Прикладная радиоэлектроника. – 2005. – Т.4, №2. – С. 161-167.

Поступила в редакцию 12.05.2010

Рецензент: д-р техн. наук, профессор, заведующий кафедрой компьютерных систем и сетей В.С. Харченко, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков, Украина.

ГАРАНТОЗДАТНІ АДАПТИВНІ СИСТЕМИ УПРАВЛІННЯ І ТЕХНОЛОГІЇ АВТОМАТИЧНОГО ПРОЕКТУВАННЯ ЇХ ПАРАЛЕЛЬНОГО АПАРАТНО-ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Г.О. Поляков

Констатується, що одним із перспективних шляхів підвищення ефективності критичних технологій і систем є створення і застосування Гарантоздатних Адаптивних Систем з паралельною обробкою даних, що самоорганізуються (ГАС). Формулюються концепція і принципи побудови ГАС, розглядаються узагальнена архітектура і проблеми проектування паралельного апаратно-програмного забезпечення АОС. Наводиться характеристика апарата формального синтезу ГАС – Структурно-Часової Дискретної Математики. Розглядається технологія автоматичного синтезу часових мультипаралельних моделей задач – Синтезатор Паралельних Моделей; технологія автоматичного синтезу часових паралельних програм – Паралельний Інтелектуальний Компілятор; технологія автоматичного синтезу мультипаралельних цифрових апаратних засобів – Автоматичний Інтелектуальний Проектувальник.

Ключові слова: паралельна система управління, гарантоздатність, самоорганізація, адаптація, інтелектуалізація, структури семантико-числової специфікації (СЧС), структурно-часова дискретна математика, часова паралельна програма, паралельне апаратно-програмне забезпечення, автоматичний синтез, верифікація, візуалізація, Синтезатор Паралельних Моделей (СПМ), Паралельний Інтелектуальний Компілятор (ПІК), Автоматичний Інтелектуальний Проектувальник (АІП).

GUARANTABLE ADAPTIVE CONTROL SYSTEMS AND AUTOMATIC DESIGN TECHNOLOGIES OF THEIR PARALLEL HARD-AND-SOFT WARE

G.A. Polyakov

The guarantable adaptive control systems (ASCS) conception and construction principles are formulated. Common architecture and design problems of parallel hard-and-soft ware design are considered. The Structure – Timing Digital Mathematics is described. The functional architectures and working principles of as automatic technology of timing parallel models synthesis (Synthesizer of Parallel Models) as automatic technology of timing multiparallel software synthesis (Parallel Intelligent Compiler) and automatic technology of multiparallel hardware synthesis (Automatic Intelligent Designer) are considered.

Keywords: guarantable parallel control system, self-organizing, adaptation, intellectuality, structures of semanic-numerical specification, structure – timing digital mathematics, timing parallel program, parallel hard-and-software, automatic synthesis, verification, viewing, synthesizer of timing parallel models (SPM), parallel intelligent compiler (PIC), automatic intelligent designer (AID).

Поляков Геннадий Алексеевич – д-р техн. наук, профессор, Академик Академии наук прикладной радиоэлектроники, Россия, e-mail: tda_ua@pochtamt.ru.