УДК 680.3

Г.А. Поляков[1], Е.Г. Толстолужская[2]

[1]*Академия наук прикладной радиоэлектроники, Москва*
[2]*Харьковский университет Воздушных Сил им. И. Кожедуба, Харьков*

ФОРМАЛЬНЫЙ СИНТЕЗ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ ДЛЯ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ VLIW-ПРОЦЕССОРОВ

В статье рассматривается технология формализованного синтеза параллельных программ для VLIW-процессоров, обеспечивающая учет временных характеристик операций, конфигурацию процессоров и систему требований (ограничений) к временным характеристикам выполнения программ.

временная параллельная модель, числовая спецификация программы, граф программы, параллельная программа,VLIW-процессор

Введение

Актуальность исследования. Быстрое расширение областей применения ЭВМ, постоянное возрастание сложности решаемых задач при одновременном повышении требований к времени их решения и достоверности получаемых результатов приводит к необходимости изыскания путей увеличения производительности. Одним из основных путей решения этой проблемы является применение параллельных процессоров и многопроцессорных ЭВМ [1 – 3]. Является общепризнанным, что эффективность параллельных компьютеров существенно зависит от качества реализуемых ими параллельных программ. «В настоящее время проблема разработки эффективного параллельного программного обеспечения оказалась центральной проблемой параллельных вычислений в целом …» … и центральный вопрос параллельных вычислений - «Как создавать эффективные программы для параллельных компьютеров?» [1] .

Цель статьи. Описание технологии формализованного синтеза параллельных программ для высокопроизводительных параллельных процессоров класса VLIW. Рассматривается обобщенный алгоритм синтеза и семантика основных этапов, иллюстрируемые с помощью конкретного примера.

Анализ литературы. Одним из перспективных классов параллельных процессоров являются процессоры со сверхдлинным командным словом – VLIW (Very Large Instruction Word) [1, 4, 5]. Работа VLIW-процессора основана на выявлении параллелизма команд во время трансляции. Транслятор анализирует последовательную программу, определяя, какие операции могут выполняться параллельно. Такие операции «упаковываются» в одну большую команду. Команда VLIW-процессора состоит из набора полей, каждое из которых отвечает за свою операцию, например, за активизацию функциональных устройств, работу с памятью, операции с регистрами и т.п. Если какая-то часть процессора на данном этапе выполнения параллельной программы не востребована, то соответствующее поле «длинной» команды не задействуется. VLIW-программа представляет собой текст, содержащий последовательность длинных команд, которые считываются из командной памяти последовательно одна за другой. После того, как длинная команда выбрана из памяти, составляющие ее обычные команды выполняются параллельно (одновременно) соответствующими функциональными устройствами или универсальными процессорами VLIW-процессора.

Общепринятый подход к созданию параллельных программ определяется следующим образом [6] «разработку параллельной программы можно представить как выполнение трех шагов:

1. Написание традиционной последовательной программы, обеспечивающей необходимые вычисления.

2. Мысленное проектирование схемы параллелизма, то есть распределения данных и вычислений по процессорам.

3. Переписывание программы в параллельном виде с использованием конкретной инструментальной системы параллельного программирования ».

Эти этапы выполняются вручную «…подавляющее большинство параллельных программ, несмотря на все усилия разработчиков систем автоматизации, пишется с использованием традиционной (ручной) технологии» [6].

Состояние решения этой проблемы в целом характеризуется отсутствием средств формализации и автоматизации разработки параллельных программ, обеспечивающих учет требований ко времени выполнения, тактовой частоте, надежности, сложности и возможность применения различных методов параллельной обработки данных для оптимизации по-

 © Г.А. Поляков, Е.Г. Толстолужская

казателей эффективности, с одной стороны, и ограничениями на возможность создания программистами эффективных параллельных программ для параллельных процессоров/ЭВМ с числом функциональных устройств/процессоров большим 3…5, с другой стороны [6 – 8].

Определения и понятия. Постановка задачи исследования

Параллельное решение задач процессорами рассматриваемого класса основано на реализации параллельных время – параметризованных программ, впервые введенных автором в 1980 – 1981 годах [9, 10]. Для достижения предельно высокой производительности параллельных процессоров необходимо обеспечить:

- максимальный параллелизм выполнения задач, соответствующий распараллеливанию на уровне инструкций/команд;
- сведение к минимуму непроизводительных затрат времени на планирование и управление параллельным процессом путем исключения из реального времени затрат на планирование параллельных процессов и уменьшения затрат на управление вычислительным процессом.

Обеспечение этих требований достигается при организации параллельной обработки данных на основе глобально параллельных время – параметризованных (ПВП) программ [9, 10].

Параллельную время – параметризованную программу (ПВП-программу) определим как конструкцию, особенностями которой являются:

- разбиение множества команд программы на подмножества (фрагменты) команд, выполняемых в соответствующие моменты времени (в состав фрагмента входят независимые команды, реализация которых должна начинаться в один и тот же момент времени);
- наличие указаний о порядке реализации фрагментов при естественном порядке их выполнения и при условных (безусловных) передачах управления;
- наличие указаний о моментах времени, в которые должно начинаться выполнение различных фрагментов;
- наличие указаний о перечне этапов, к выполнению которых сводится реализация каждого фрагмента;
- представление данных в виде структур, содержащих подмножества данных, соответствующие фрагментам команд.

Выполнение параллельной программы заключается в реализации в определенные моменты времени соответствующих фрагментов команд с последующим формированием номера (номеров) очередного фрагмента при естественном порядке выполнения фрагментов (или при условных передачах управления). Реализация произвольного фрагмента параллельной программы состоит в параллельном считывании из командной памяти параллельного процессора/многопроцессорной ЭВМ множества команд, входящих в текущий фрагмент, параллельном считывании из памяти данных соответствующего фрагмента данных, распределении команд и данных между обрабатывающими устройствами (процессорами), одновременном выполнении требуемого множества операций с параллельной записью множества результатов в память данных. Как будет оказано ниже, VLIW-программы представляют собой частные случаи ПВП-программ.

Определим числовую спецификацию исходной последовательной программы и ПВП-программы как совокупность следующих структур данных [11].

Базовая структура BF представляет собой следующую структуру вида

BF = (N, MET, TYP, NS, SJ, BJ, NW, WJ, MP1, MP2, VH, VIH, NAME),

где N – номер вершины графа; MET – метка вершины графа; TYP – тип инструкции Си-программы, интерпретируемой соответствующей вершиной графа; NS – указатель на начало цепочки номеров вершин графа, образующих сопряженное множество конкретной вершины графа; SJ – мощность сопряженного множества для конкретной вершины графа); BJ – номер естественной части Си-программы графа, к которой принадлежит конкретная вершина; NW – указатель на начало цепочки номеров вершин графа, образующих внешнее множество конкретной вершины; WJ – мощность внешнего множества для конкретной вершины; MP1, MP2 – метки вершины графа, интерпретирующей операцию перехода «upl»; VH, VIH – количество входов вершины (число операндов операции) и количество выходов вершины; NAME – идентификатор (переменной, константы или операции).

Структура связей CF представляет собой следующую структуру вида

CF = (JS, S, SWIH, SWHO, JW, W, WWHO, WWIH),

где JS – указатель на продолжение цепочки номеров вершин графа, образующих сопряженное множество конкретной вершины графа для рассматриваемой вершины; S – сопряженное множество конкретной вершины графа для рассматриваемой вершины; SWIH – номер выхода конкретной сопряженной вершины, связанной с входом рассматриваемой вершины; SWHO – номер входа вершины, связывающего эту вершину или модуль с входом соответствующей сопряженной вершины; JW – указатель на продолжение цепочки номеров вершин графа, образующих внешнее множество W конкретной вершины графа или конкретного модуля (в случае

F-схемы); W – внешнее множество конкретной вершины графа; WWHO – номер входа вершины, являющейся внешней для рассматриваемой вершины; WWIH – номер выхода рассматриваемой вершины, связывающего ее с внешней вершиной.

Временной файл TF представляет собой следующую структуру вида

$$TF = (N, NT), N = 0, 1, \ldots, p - 1,$$

где p – количество вершин; N – номер вершины графа; NT – момент времени, в который начинается выполнение j-й инструкции параллельной модели алгоритма, интерпретируемой j-й вершиной соответствующей временной параллельной граф-схемой ВПГС [11].

Задача исследований формулируется следующим образом: синтезировать параллельную (в приведенном выше понимании) VLIW-программу, исходя из традиционной последовательной Си-программы задачи и конкретной конфигурации VLIW-процессора.

Результаты исследования

Обобщенный алгоритм синтеза ПВП-программ (и, как их частных случаев, VLIW-программ) показан на рис. 2. При рассмотрении содержания различных этапов алгоритма (рис. 2) используем простую задачу, Си - программа которой дана на рис. 1.

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

Рис. 1. Исходная Си-программы задачи

Содержанием этапа 1 (символ 2) является решение двух задач:

- синтез для исходной Си – программы числовой спецификации в формате базы данных сопряжено-внешних множеств, БД СВМ [11];
- синтез по числовой спецификации Си – программы соответствующего графического представления в виде Си-графа [12].

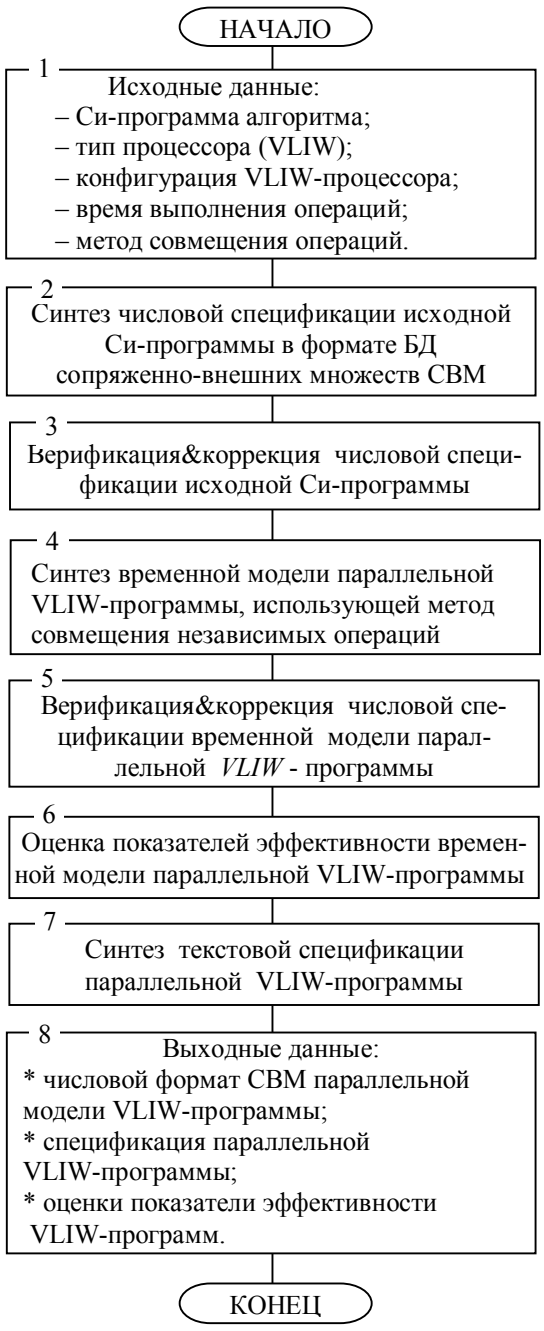

Рис. 2. Обобщенный алгоритм синтеза параллельных программ для высокопроизводительных VLIW-процессоров

Результаты выполнения этапа представляют рис. 3 , табл. 1 и 2 . На рис. 3 показана графическая спецификация исходной Си-программы в виде соответствующего Си-графа. Спецификация содержит следующие элементы:

- множество вершин, имеющих различную семантику: вершины – входы данных, соответствующие входным данным исходной Си-программы (оператор P_0 типа «vx» ввода значения «a», оператор P_1 типа «vx» ввода значения «b»), вершины – имена переменных, имеющие тип «var» (P_2 – для «a», P_3 – для «b», P_4 – для «k», P_5 – для «z», P_6 – для «p»); множество вершин – операций различ-

ных типов (P_7, P_8, … , P_{27}), соответствующее множеству операций исходной Си – программы; вершина – выход результата, имеющая тип «vih» (оператор P_{28});

- множество ребер, соединяющих некоторые вершины и отражающих фактические связи (по данным и по управлению) между операциями исходной Си-программы.

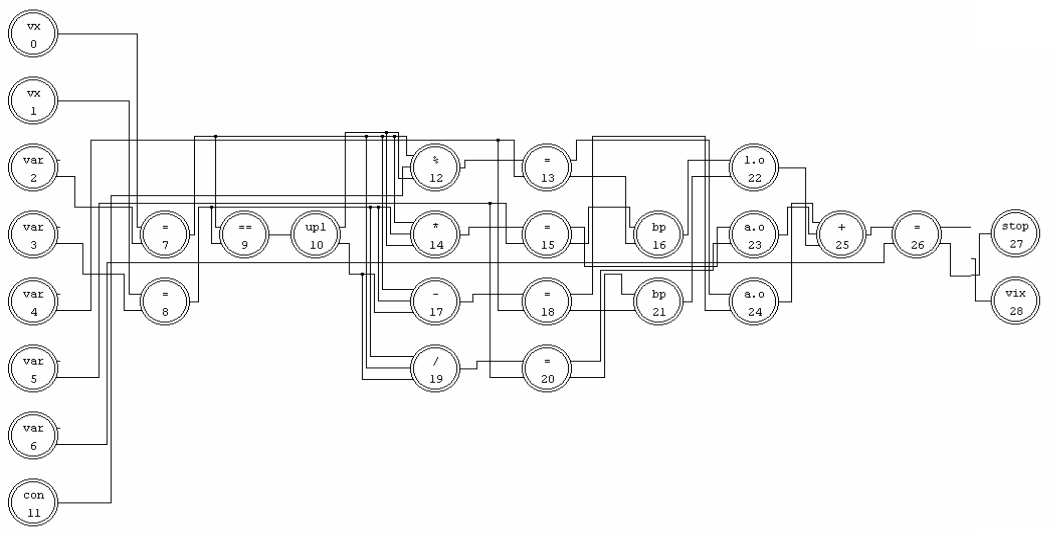

Рис. 3. Графическая модель (Си-граф) исходной Си-программы

Таблица 1

Базовая структура BF числовой спецификации Си-программы задачи

N	MET	TYP	NS	SJ	BJ	NW	WJ	MP1	MP2	VH	VIH	Name	
0	0	58	–1	0	0	0	1	0	0	0	1	a_in	Входы исходных данных
1	0	58	–1	0	0	1	1	0	0	0	1	b_in	
2	0	58	–1	0	0	2	1	0	0	0	1	a	
3	0	47	–1	0	0	3	1	0	0	0	2	b	Переменные
4	0	47	–1	0	0	4	1	0	0	0	2	k	
5	0	47	–1	0	0	6	2	0	0	0	2	z	
6	0	47	–1	0	0	8	1	0	0	0	2	p	
7	0	12	0	2	0	9	5	0	0	2	1	=	Операции исходной Си-программы
8	0	12	2	2	0	14	4	0	0	2	1	=	
9	0	23	4	2	0	18	1	0	0	2	1	==	
10	0	51	6	1	0	19	4	1	2	1	2	upl	
11	0	57	–1	0	1	23	1	0	0	0	1	C2_	
12	1	5	7	3	1	24	1	0	0	3	1	%	
13	0	12	10	2	1	25	2	0	0	2	2	=	
14	0	3	12	3	1	27	1	0	0	3	1	*	
15	0	12	15	2	1	28	2	0	0	2	2	=	
16	0	50	17	2	1	30	1	3	0	2	1	bp	
17	2	2	19	3	2	31	1	0	0	3	1	-	
18	0	12	22	2	2	32	2	0	0	2	2	=	
19	0	4	24	3	2	34	1	0	0	3	1	/	
20	0	12	27	2	2	35	2	0	0	2	2	=	
21	0	50	29	2	2	37	1	3	0	2	1	bp	
22	3	54	31	2	3	38	1	0	0	2	1	l.o	
23	0	53	33	2	3	39	1	0	0	2	1	a.o	
24	0	53	35	2	3	40	1	0	0	2	1	a.o	
25	0	1	37	3	3	41	1	0	0	3	1	+	
26	0	12	40	2	3	42	2	0	0	2	2	=	
27	0	49	42	1	3	–1	0	0	0	1	0	stop	Выход Результат
28	0	48	43	1	3	–1	0	0	0	1	0	p_out	

Таблица 2

Структура связей *CF* числовой спецификации Си-программы задачи

NN	JS	S	SWIH	SWHO	JW	W	WWHO	WWIH
0	1	0	0	0	–1	7	0	0
1	–1	2	1	1	–1	8	0	0
2	3	1	0	0	–1	7	1	1
3	–1	3	1	1	–1	8	1	1
4	5	7	0	0	5	13	1	1
5	–1	8	0	1	–1	18	1	1
6	–1	9	0	0	7	15	1	1
7	8	7	0	0	–1	20	1	1
8	9	11	0	1	–1	26	1	1
9	–1	10	0	2	10	9	0	0
10	11	4	1	1	11	12	0	0
11	–1	12	0	0	12	14	0	0
12	13	7	0	0	13	17	0	0
13	14	8	0	1	–1	19	1	0
14	–1	10	0	2	15	9	1	0
15	16	5	1	1	16	14	1	0
16	–1	14	0	0	17	17	1	0
17	18	15	1	0	–1	19	0	0
18	–1	13	1	1	–1	10	0	0
19	20	7	0	0	20	12	2	0
20	21	8	0	1	21	14	2	0
21	–1	10	1	2	22	17	2	1
22	23	4	1	1	–1	19	2	1
23	–1	17	0	0	–1	12	1	0
24	25	8	0	0	–1	13	0	0
25	26	7	0	1	26	16	1	1
26	–1	10	1	2	–1	24	0	0
27	28	5	1	1	–1	15	0	0
28	–1	19	0	0	29	16	0	1
29	30	20	1	0	–1	23	0	0
30	–1	18	1	1	–1	22	0	0
31	32	21	0	1	–1	18	0	0
32	–1	16	0	0	33	21	1	1
33	34	20	0	1	–1	24	1	0
34	–1	15	0	0	–1	20	0	0
35	36	18	0	1	36	21	0	1
36	–1	13	0	0	–1	23	1	0
37	38	24	0	0	–1	22	1	0
38	39	23	0	1	–1	25	2	0
39	–1	22	0	2	–1	25	1	0
40	41	6	1	1	–1	25	0	0
41	–1	25	0	0	–1	26	0	0
42	–1	26	1	0	43	27	0	1
43	–1	26	0	0	–1	28	0	0

Структуры BF и CF (табл. 2, 3) представляют числовую спецификацию исходной Си – программы, задавая для каждого оператора P_j (j = N = 0, 1, … , 28) соответствующую вектор – строку (N, MET, TYP, NS, SJ, BJ, NW, WJ, MP1, MP2, VH, VIH, Name), семантика элементов которой была определена выше. Отметим, что элементы (NS, SJ) N-й строки структуры BF задают для оператора P_j количество его сопряженных (входных) операторов (SJ), а NS является указателем на строку (с номером NS) структуры CF, с которой начинается цепочка номеров сопряженных для P_j операторов (в массиве S),

указателем на продолжение цепочки является значение JS (при этом значение JS = –1 означает завершение цепочки). Аналогичный смысл имеют элементы (NW, WJ) структуры BF и (JW,W) структуры CF примени-тельно к заданию множества номеров внешних операторов (операторов-приемников), использующих результаты выполнения P_j. Например, для оператора P_9 сопряженное множество операторов $S(P_9)$ = (S(nsj = 4), S(nsj = 5) = (7,8); внешнее множество $W(P_9)$ = (W(nwj = 18)) = 10.

Содержанием этапа 2 (символ 3) является проверка корректности результатов синтеза числовой спецификации (структур BF и CF) исходной Си-программы и соответствующего Си-графа. Верификация включает следующие проверки:

а) равенство суммарных количеств сопряженных связей и внешних связей в числовой спецификации и в Си-графе;

б) соответствие количества входов, количества выходов операторов и их разрядности значениям, определенным в исходной Си-программе;

в) корректность семантики входов и выходов различных операторов;

г) отсутствие “ висячих “ входов у операторов числового формата СВМ Си-программы и Си-графа.

Результаты компиляционной верификации числовой и графической спецификации (Си-графа) исходной Си-программы представлены на рис. 4. Эти результаты подтверждают корректность (ОК) формата СВМ и Си-графа исходной Си-программы. Формализованное описание технологии верификации и семантики основных этапов содержится в работе [14]. На этапе 3 (символ 4) обеспечивается (при использовании метода совмещения независимых операций) синтез параллельной время – параметризованной модели параллельной программы (в виде временной параллельной граф – схемы, ВПГС [10]), удовлетворяющей заданным требованиям и ограничениям (время решения задачи, сложность/количество процессоров). Формализованное описание процедур, обеспечивающих решение задачи синтеза, и семантики различных этапов синтеза содержится в [10, 11, 13]. Табл. 3 задает длительности выполнения различных операций (в тактах), использованные при синтезе.

Результатами выполнения этапа являются:

- графическая спецификация время – параметризованной параллельной модели программы (рис. 5);
- гистограммная спецификация параллельной время – параметризованной модели (рис. 6).

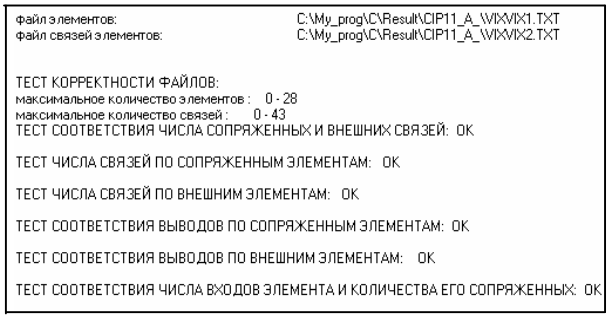

Рис. 4 .Результаты верификации числовой спецификации и Си-графа исходной Си-программы

Состав операторов параллельной модели VLIW-программы и их связи по данным и по управлению определяются структурами BFM , CFM (совпадающими со структурами BF и CF, представленными табл. 1 и 2) и структурой TFM, задающей для каждого оператора P_j значение параметра начала (эта структура не приводится, поскольку идентичная информация представлена на рис. 5).

Содержанием этапа 4 (символ 5) является проверка корректности синтезированной временной параллельной модели исходной Си-программы. Верификация включает следующие проверки [14]:

а) равенство суммарных количеств сопряженных связей и внешних связей в параллельной модели;

б) соответствие количества входов, количества выходов операторов и их разрядности значениям, определенным в исходной Си-программе;

в) корректность семантики входов и выходов различных операторов;

г) корректность разрядности входных и выходных различных операторов;

д) отсутствие “ висячих “ входов у операторов параллельной модели;

е) корректность временных отношений «предшествования – следования» между операторами – источниками, формирующими значения операндов, и операторами – приемниками, использующими эти значения при выполнении последующих операторов параллельной модели.

Результаты верификации представлены на рис. 7. Эти результаты (ОК) подтверждают корректность числового формата СВМ и ВПГС графической спецификации параллельной модели VLIW-программы (состава операторов и их связей – п.»а», временных отношений следования операторов – п.«б»).

Таблица 3

Длительность t^o(typ) выполнения операций различных типов (typ) параллельной модели VLIW-программы (такты)

typ	=	==	upl	%	*	/	bp	l.o, a.o	stop
t^o (нс)	2.00	1.00	1.00	35.00	10.00	35.00	1.00	1.00	1.00

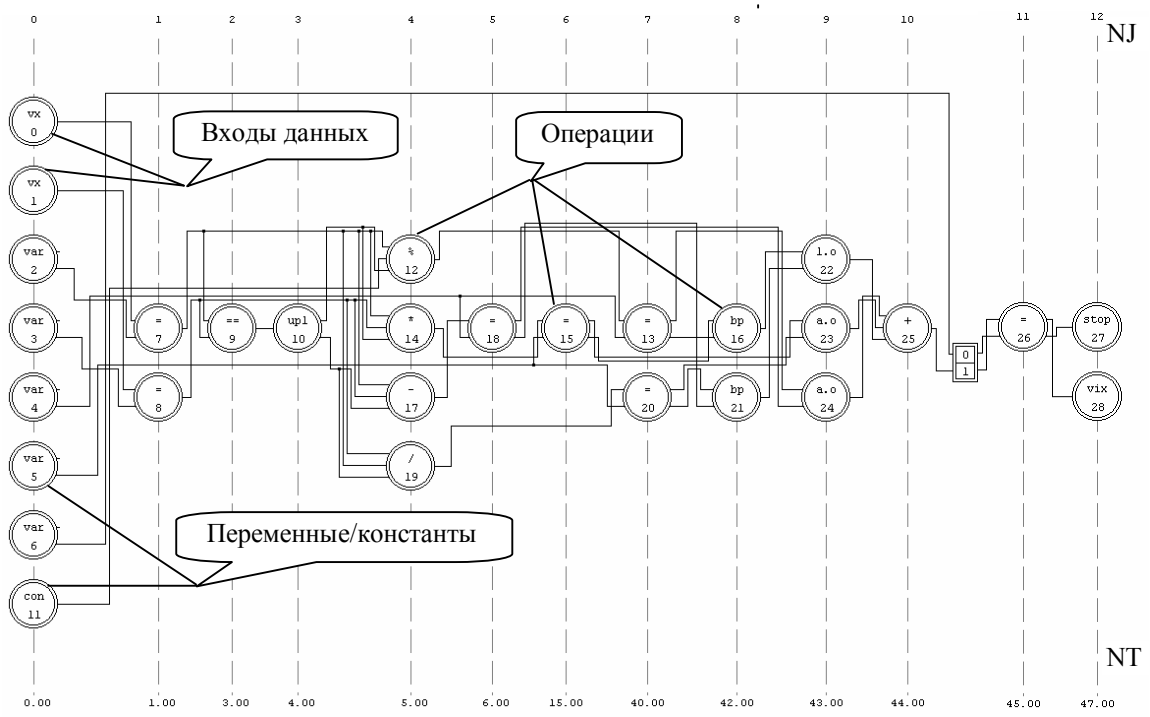

Рис. 5. Максимально параллельная время – параметризованная модель VLIW-программы при совмещении независимых операций

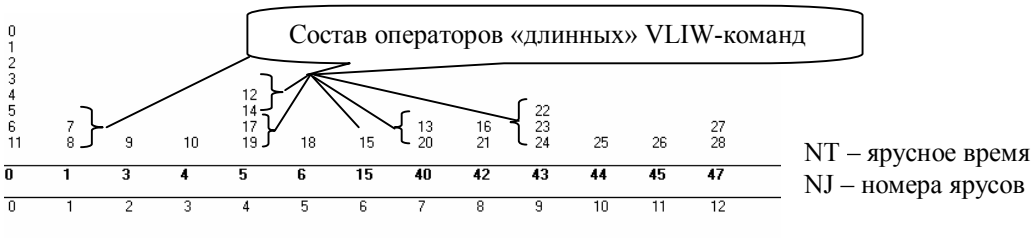

Рис. 6. Гистограммная модель параллельной времяпараметризованной VLIW-программы при совмещении независимых операций

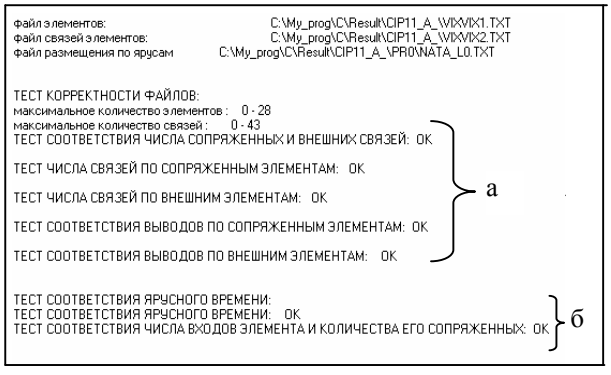

Рис. 7. Результаты верификации временной параллельной модели VLIW-программы

На этапе 5 (символ 6) обеспечивается оценка показателей эффективности синтезированной временной модели параллельной программы: времени выполнения T, сокращения DT времени выполнения параллель-ной модели по сравнению с последовательной реализацией, коэффициента S загрузки процессоров, а также эффективности распараллеливания R. Зависимости показателей эффективности от количества процессоров NM = 1, 2, 3, 4 в составе VLIW-процессора представляют рис. 8 – 11.

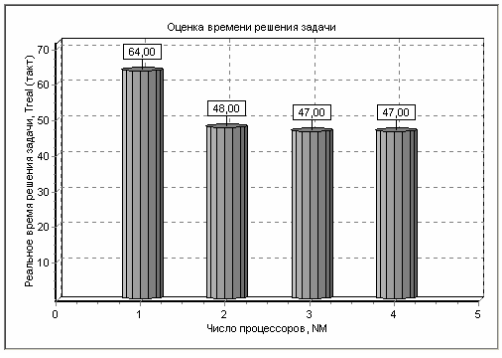

Рис. 8. Зависимость времени выполнения VLIW-модели от числа процессоров NM

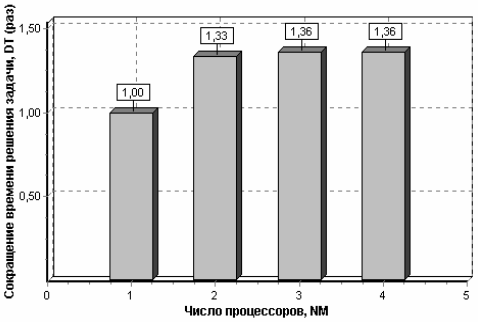

Рис. 9. Зависимость сокращения времени выполнения VLIW-модели от числа процессоров NM

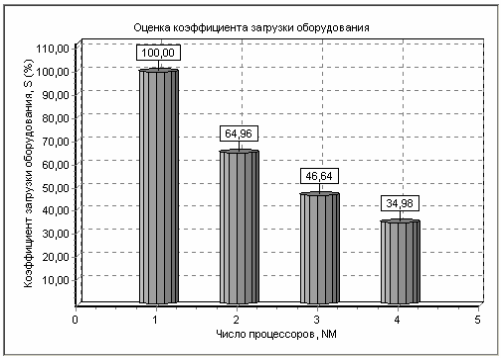

Рис. 10. Зависимость загрузки S от количества NM процессоров

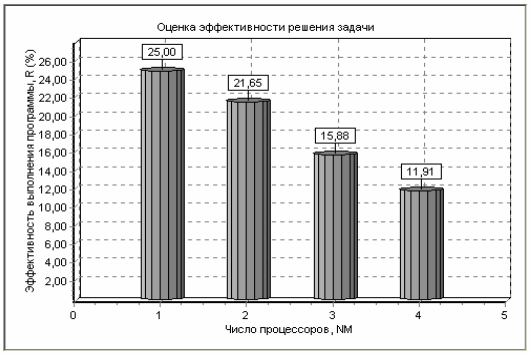

Рис. 11. Зависимость эффективности R = S * DT VLIW-модели от числа NM процессоров

На этапе 6 (символ 7) выполняется распределение данных в памяти данных и синтезируется параллельная VLIW-программа, используя числовую, графическую и гистограммную спецификации временной параллельной VLIW-модели. Этап реализуется путем формального перевода числовой спецификации параллельной модели VLIW-программы в текстовый формат VLIW-программы.

Результаты выполнения этапа представляют табл. 4 и 5. Множество номеров операторов, реализуемых первым процессором (нить 1, thread 1): 9, 10, 12, 13, 14, 15, 16, 25, 26, 27, множество номеров операторов, реализуемых вторым процессором (нить 2): 17, 18, 19, 20, 21.

Таблица 4

Распределение данных в памяти данных VLIW-процессоров ЗУ данных процессора 1

Адреса данных	Имена данных	Значения данных
0	a	«a»
1	b	«b »
2	k	«k»
3	p	«p»
4	C2_	«C2_»
5	w_t	32.0

ЗУ данных процессора 2

Адреса данных	Имена данных	Значения данных
0	a	«a»
1	b	«b »
2	z	«z»
3	w_t	23.0

Таблица 5

Результат синтеза параллельной VLIW-программы (содержание командной памяти VLIW-процессора)

NI	Команды 1-го процессора				Команды 2-го процессора			
	Op	A1	A2	AP	Op	A1	A2	AP
1	scanf	&a	---	0	scanf	&a	---	0
2	scanf	&b	---	1	scanf	&b	---	1
3	==	0	1	---	==	0	1	---
4	upl	5	10	---	upl	5	10	---
5	%	0	4	---	*	0	1	---
6	---	---	---	---	=	---	---	2
7	---	---	---	---	wait	3	---	---
8	=	---	---	2	---	---	---	---
9	bp	15	---	---	bp	12	---	---
10	-	0	1	---	/	1	0	---
11	=	---	---	2	---	---	---	---
12	wait	5	---	---	---	---	---	---
13	---	---	---	---	=	---	---	2
14	bp	15	---	---	bp	15	---	---
15	+	2	3	---	---	---	---	---
16	=	---	---	3	---	---	---	---
17	stop	---	---	---	---	---	---	---

Принятые в табл. 5 обозначения: NI – номер «длинной » VLIW – команды, Op – тип команды, A1 и A2 – адреса первого и второго операндов, AP – адрес записи результата.

Отметим, что наличие в параллельной время – параметризованной модели (рис. 5, 6) моментов $t_j^н$ начала выполнения операторов P_j (j = 0, 1, … ,28) и длительности выполнения операций различных типов (табл. 3) позволяет получить значения NT – моментов начала выполнения длинных команд VLIW-программы.

Выводы

1. Эффективность параллельных процессоров и многопроцессорных компьютеров существенно зависит от качества реализуемых ими параллельных программ. Известные технологии автоматизации параллельного программирования имеют ограниченные возможности повышения качества, увеличения объема и сокращения сроков разработки параллельных программ, обусловленные фактором субъективного творчества программистов.

2. Разработанная методика обеспечивает формализацию синтеза параллельных программ с заданным временем выполнения для высокопроизводительных VLIW-процессоров, исходя из традиционных последовательных Си-программ и имеющейся или определяемой в процессе синтеза оптимальной конфигурации VLIW-процессора.

3. Изложенная методика может рассматриваться как реальная основа создания инструментальных средств автоматического синтеза параллельных программ для высокопроизводительных процессоров класса VLIW, параллельных процессоров с управлением потоком данных и вычислительных систем с симметричной мультипроцессорной обработкой.

Список литературы

1. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления.– С.-Пб.: БХВ-Петербург, 2002. – 608 с.

2. Поляков Г.А. Адаптивные самоорганизующиеся системы с мультипараллельной обработкой данных – стратегия развития цифровой вычислительной техники в XXI веке // Сб. науч. тр. Прикладная радиоэлектроника. – Х.: АНПРЭ, ХНУРЭ. – 2002. – Т. 1, № 1. – С.57-69.

3. Программирование на параллельных вычислительных системах: Пер. с англ. / Р. Бэбб,Дж. Р. МакГроу, Т. Акселрод и др.; Под ред. Р. Бэбба 11. – М.: Мир, 1991. – 376 с.

4. Компьютеры на СБИС: В 2-х кн. Кн.1: Пер. с япон. / Т. Мотоока, С. Томита, Х. Танака и др. – М.: Мир,1988. – 392 с.

5. СуперЭВМ. Аппаратная и программная реализация: Пер. с англ. / Под ред. С. Фернбаха. – М.: Радио и связь. 1991. – 318 с.

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

7. Вычислительные процессы и системы. Вып.2 / Под ред. Г.И. Марчука. – М.: Наука. Главная редакция физико-математической литературы, 1985. – 352 с.

8. Системы параллельной обработки: Пер. с англ. / Под ред.Д. Ивенса.- М.: Мир. 1985.-416 с.

9. Поляков Г.А. Синтез время – параметризованных параллельных программ – новый подход к параллельному программированию для специализированных многопроцессорных вычислительных комплексов АСУ реального масштаба времени // Всесоюзная научно – техническая конференция «Программное обеспечение АСУ». Методология разработку АСУ. Часть 2. – Калинин: НПО «ЦЕНТРПРОГРАММСИСТЕМ»,1980. – С. 87-89.

10. Поляков Г.А. Глобально-параллельные и время-параметризованные – новый подход к синтезу и выполнению параллельных программ в АСУ реального времени // Всесоюзная конференция «Программное обеспечение вычислительных сетей и систем реального времени». – К.: ГК СССР по науке и технике, Академия наук СССР, АН Украинской ССР, ИК АН СССР. – 1981. – С. 130-132.

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

12. Поляков Г.А., Онищенко В.В. Визуализация статико – динамических объектов автоматического проектирования мультипараллельных цифровых устройств // Системи обробки інформації. – Х.: ХВУ. – 2004. – Вип. 7 (35). – С. 169-177.

13. Толстолужская Е.Г. Методика фоормализованного синтеза мультипараллельных архитектурно-ориентированных моделей решения задач // Моделювання та інформаційні технології. – К.: НАНУ, ІПМЕ ім. Г.С. Пухова. – 2003. – Вып. 22. – С. 206-215.

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

Поступила в редколлегию 1.08.2007

Рецензент: д-р техн. наук, проф. В.С. Харченко, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков.