УДК 681.31

В.В. ОНИЩЕНКО

Харьковский университет Воздушных Сил, Украина

ЧИСЛОВАЯ И ГРАФИЧЕСКАЯ СПЕЦИФИКАЦИЯ СИ-ПРОГРАММ ПРИ ПРОЕКТИРОВАНИИ ЦИФРОВЫХ УСТРОЙСТВ

Отмечается, что основной тенденцией развития САПР в настоящее время является снижение субъективной роли человека в процессе проектирования путем перехода к системам автоматического проектирования параллельных цифровых устройств. Важной задачей создания таких систем является разработка единых числовых форматов спецификации и визуализации проекта, обеспечивающих формализацию и оперативность оценки человеком корректности результатов автоматического проектирования. Приводится пример, поясняющий семантику формата СВМ и полученный на его основе Си-граф задачи.

формат сопряженно-внешних множеств, визуализация Си-графов, системы автоматического проектирования параллельных цифровых устройств

Введение

Одним из основных путей повышения эффективности вычислительных средств является применение методов параллельной обработки информации [1 – 3]. Реализация данного направления требует создания и внедрения в практику систем автоматического проектирования параллельных аппаратных средств [3 – 5]. Автоматический характер этих систем делает исключительно актуальной задачу применения на всех этапах проектирования единых форматов спецификации и визуализации проекта, обеспечивающих формализацию оперативной оценки человеком корректности результатов автоматического проектирования [6].

Анализ существующих систем автоматизированного проектирования (САПР) показывает, что при их проектировании широко используется текстовое представление формата (Verilog, VHDL), а применяемые средства визуализации решают специфические задачи и не могут удовлетворить требованиям по универсальности применения, оперативности, наглядности и информативности, предъявляемым к средствам визуализации перспективных систем автоматического проектирования [7, 8].

В работах [3, 9], посвященных созданию систем автоматического проектирования параллельных цифровых устройств обоснован аппарат числовой спецификации – формат сопряженно-внешних множеств (СВМ), который обеспечивает выполнение перечисленных требований.

Цель статьи:

1) описать алгоритм формализованного перехода от текста Си-программы к числовому формату СВМ;

2) описать алгоритм перехода от числового формата СВМ к графовой форме визуализации исходной Си-программы – Си-графу.

Методика формализованного синтеза формата СВМ и Си-графов

Обобщенная блок-схема методики формализованного синтеза формата СВМ и Си-графов представлена на рис. 1.

Исходными данными для этапа синтеза формата СВМ (символ 2) являются:

– текст Си-программы;

– операции и символы языка Си;

– ключевые слова языка Си;

– функции языка Си;

– архитектура формата СВМ.

На выходе этапа получаем формат СВМ, представленный двумя файлами: базовым файлом (БФ) и файлом связей (ФС).

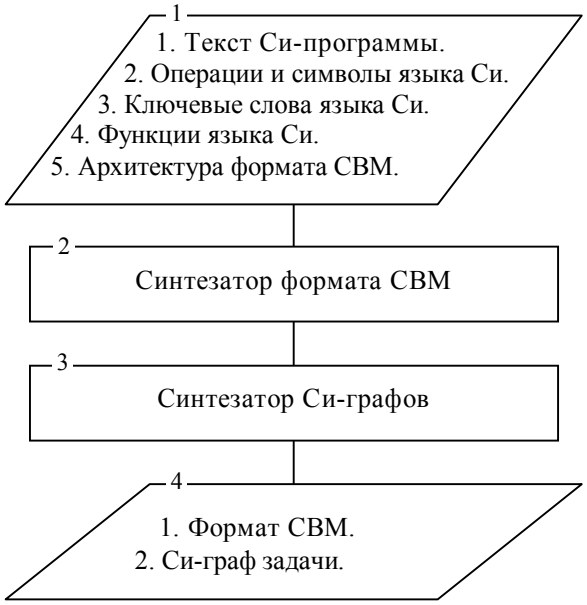

Рис. 1. Обобщенная блок-схема методики формализованного синтеза формата СВМ и Си-графов

Исходными данными для этапа синтеза Си-графов (символ 3) являются:

- формат СВМ;
- таблица идентификаторов и констант.

На выходе этапа получаем Си-граф.

Символ 1 – обеспечивает ввод исходной информации. Если первые из четырех входных данных являются стандартными, то на архитектуре формата СВМ стоит остановиться подробнее.

Архитектура формата СВМ представлена двумя файлами: базовым файлом (БФ) и файлом связей (ФС). Поясним семантику файла БФ:

N – массив номеров *j* оператор P_j (данные и операции считаются операторами и нумеруются подряд от $j = 0$ до $j = n - 1$; n – количество операторов);

MET – массив меток операторов (содержит метки, задаваемые в тексте Си-программы, и автоматически вводимые метки в процессе синтеза формата СВМ);

TYP – массив типов операторов задачи;

NSJ – массив указателей *nsj*[*j*] на младший из номеров *i*-й строки ФС, с которой начинается цепочка сопряженных операторов P_i для оператора P_j;

SJD – массив значений *sjd*[*j*] мощностей сопряженных множеств S_j операторов P_i для оператора P_j;

BJ – массив номеров естественных частей (не разветвляющихся и нециклических фрагментов) Си-программы, значение элемента *bj*[*j*] задает номер естественной части, к которой принадлежит оператор P_j;

NWJ – массив указателей *nwj*[*j*] на младший из номеров *i*-й строки ФС, с которой начинается цепочка внешних операторов P_i для оператора P_j;

WJD – массив значений *wjd*[*j*] мощностей внешних множеств W_j операторов P_i для оператора P_j;

*MP*1 – массив меток операторов безусловного перехода и перехода по значению "истина" Си-программы ("0" означает отсутствие метки);

*MP*2 – массив меток операторов перехода по значению "ложь" Си-программы ("0" означает отсутствие метки);

VH – массив, элементы которого *vh*[*j*] задают количество входов оператора P_j (принимается, что для операторов P_j, интерпретирующих имена входов исходных данных (*a_in*, *b_in*, *c_in*) $vh[j] = 0$; $vih[j] = 1$; переменных (*a*, *b*, *c*) $vh[j] = 0$; $vih[j] = 2$; значение *vh*[*j*] для операторов-операций/функций равно числу операндов соответствующей операции; оператор *stop* имеет один вход $vh[24] = 1$);

VIH – массив, элементы которого *vih*[*j*] задают количество выходов оператора P_j (принимается, что для операторов P_j, интерпретирующих имена выходов (*s_out*); оператор *stop* не имеет выходов $vih[j] = 0$);

RES – массив, содержащий имена входов данных (например, *a_in*, *b_in*), имена исходных переменных, констант, символов операций языка Си, имена промежуточных и окончательных результатов решения задачи, имена выходов (например, *s_out*), а также имя оператора *stop*.

Поясним семантику файла ФС:

N – массив номеров *k* строк (*k* = 0, 1, 2, ... , *sn*–1; *sn* – количество связей по данным и по управлению между "*n*" операторами алгоритма, представляемого исходной Си-программой);

JSD – массив указателей *jsd*[*j*] на множество номеров сопряженных операторов P_i для оператора P_j, начинающийся с указателя *jsd*[*j*] = *nsj*[*j*] строки с номером *nsj*[*j*] массива *SPJD* и заканчивающихся *k*-й строкой массива *JSD*, имеющей *jsd*[*k*] = –1 (при этом каждый указатель *jsd*[*k*] ≠ –1 указывает на некоторый элемент массива *SPJD*, задающий номер очередного оператора P_i, сопряженного для P_j);

SPJD – массив, задающий для каждого оператора P_j (*j* = 0, 1, 2,..., *n*–1) его сопряженное множество S_j, т.е. номера *i* операторов P_i, выходы которых являются входами для P_j (указателями на элементы множества S_j являются значения элементов соответствующей цепочки указателей *jsd*[*j*] массива *JSD*, начинающейся с указателя *jsd*[*j*] = *nsj*[*j*] из массива *NSJ* файла БФ);

SNWIH – массив, задающий для каждой пары операторов P_i, P_j (рассматриваемых соответственно как сопряженный и внешний операторы), соединенных связью по данным, связью по управлению и/или осведомительной связью (либо некоторым сочетанием связей перечисленных типов), номер выхода оператора P_i, соответствующий рассматриваемой "сопряженной связи";

SNWHO – массив, задающий для каждой пары операторов P_i, P_j (рассматриваемых соответственно как сопряженный и внешний операторы), соединенных связью по данным, связью по управлению и/или осведомительной связью (либо некоторым сочетанием связей перечисленных типов), номер входа оператора P_i, соответствующий рассматриваемой "сопряженной связи";

JWD – массив указателей *jwd*[*j*] на множество номеров внешних операторов P_i для оператора P_j, начинающийся с указателя *jwd*[*j*] = *nwj*[*j*] строки с номером *nwj*[*j*] массива *WPJD* и заканчивающихся *k*-й строкой массива *JWD*, имеющей *jwd*[*k*] = –1 (при этом каждый указатель *jwd*[*k*] ≠ –1 указывает на некоторый элемент массива *WPJD*, задающий номер очередного внешнего для P_j оператора P_i);

WPJD – массив, задающий для каждого оператора P_j (*j* = 0,1,2,...,*n*–1) его внешнее множество W_j, т.е. номера *i* операторов P_i, для которых входами являются выходы оператора P_j (указателями на элементы множества W_j являются значения элементов соответствующей цепочки указателей *jwd*[*j*] массива *JWD*, начинающейся с указателя *jwd*[*j*] = *nwj*[*j*] из массива *NWJ* файла БФ);

WNWHO – массив, задающий для каждой пары операторов P_i, P_j (рассматриваемых соответственно как внешний и сопряженный операторы), соединенных связью по данным, связью по управлению и/или осведомительной связью (либо некоторым сочетанием связей перечисленных типов), номер входа оператора P_i, соответствующий рассматриваемой "внешней связи";

WNWIH – массив, задающий для каждой пары операторов P_i, P_j (рассматриваемых соответственно как внешний и сопряженный операторы), соединенных связью по данным, связью по управлению и/или осведомительной связью (либо некоторым сочетанием связей перечисленных типов), номер выхода оператора P_j, соответствующие рассматриваемой "внешней связи".

Символ 2 – представляет собой этап синтеза формата СВМ. Он включает:

– разделение текста Си-программы на лексемы;

– формирование обратной польской записи;

– формирование формата СВМ.

Первые из двух перечисленных пунктов являются стандартными и применяются в большинстве трансляторов. Что касается третьего пункта, то он является новым, по сравнению с существующими трансляторами, где обратная польская запись переводится на машинный язык.

Поясним построение формата СВМ на конкретном примере (текст 1).

```
#include <stdio.h>
void main(void)
{
      int a,b,c;
      int k,s;
      scanf("%d %d %d",&a,&b,&c);
      if( a + b < 0 )
        k = a * c;
      else
        k = b + c;
      s = k % 2;
      printf("%4d\n",s);
}
```

Текст 1. Разветвляющаяся Си-программа

Соответствующий ей формат СВМ представлен в табл. 1 (базовый файл – БФ) и табл. 2 (файл связей – ФС).

Построение формата СВМ можно условно разбить на 5 основных шагов:

1. Заполняются строки файла БФ для переменных, которые подаются на вход программы и записаны явным образом в тексте программы в операторе *scanf*. Строки файла ФС для массивов *JSD*, *SPJD*, *SNWIH, SNWHO* не заполняются, т.к. сопряженные операторы отсутствуют ($nsj[j] = -1$ и $sjd[j] = 0$). В рассматриваемом примере заполняются строки 0÷2 файла БФ. В массиве *RES* при этом к имени переменной дописывается строка *_in*, указывающая на то, что это входные данные (*input*).

2. Заполняются строки файла БФ для переменных, перечисленных в декларативной части (например, *int a,b*;). Строки файла ФС для массивов *JSD*, *SPJD, SNWIH, SNWHO* не заполняются, т.к. сопряженные операторы отсутствуют ($nsj[j] = -1$ и $sjd[j] = 0$). В нашем примере заполняются строки 3÷7 файла БФ, в массивах *RES* которых содержатся те же имена, что и в декларациях. В дальнейшем эти имена будут служить вспомогательной информацией для указания переменной, с которой ведется работа.

3. Заполняются строки файла БФ, соответствующие основным операторам программы (кроме операторов *scanf* и *printf*). Заполнение происходит в соответствии с заданным алгоритмом для конкретной конструкции языка Си посредством обработки обратной польской записи. Заполняются также строки файла ФС для массивов *JSD, SPJD, SNWIH, SNWHO*. Нумерация номеров входов/выходов производится с нуля. Заканчивается 3-й шаг оператором *stop*, являющимся признаком завершения программы.

В нашем примере заполняются строки 8÷24 файла БФ и строки 0÷33 файла ФС. Строки 12 и 21 отражают факт наличия в программе констант, специфицированных в массиве *RES* символами '*C*' и '_'. Строка 14 отражает факт наличия в программе оператора *if* и означает "разветвление" программы на два направления. В массиве *RES* при этом записывается *upl*. Строка 19 отражает факт логического объединения двух управляющих связей, интерпретирующих передачу управления по меткам. Строка 20 отражает факт арифметического объединения двух связей по данным, соответствующих передаче значения переменной *k*.

4. Заполняются строки файла БФ для переменных, которые подаются на выход программы и записаны явным образом в тексте программы в операторе *printf*. Заполняются также и строки файла ФС для массивов *JSD, SPJD, SNWIH, SNWHO*. В рассматриваемом примере заполняется строка 25 файла БФ и строка 31 файла ФС.

В массиве *RES* при этом к имени переменной *s* дописывается строка *_out*, указывающая на то, что это выходное данное (*output*).

5. Осуществляется заполнение всех строк файла БФ для массивов *NWJ* и *WJD*, заполнение которых пропускается на шагах 1–4, а также заполнение всех строк файла ФС для массивов *JWD, WPJD*, *WNWHO* и *WNWIH*. Другими словами, заполняется вся информация о внешних связях.

Таблица 1

Базовый файл (БФ)

N	MET	TYP	NSJ	SJD	BJ	NWJ	WJD	MP1	MP2	VH	VIH	RES
0	0	58	-1	0	0	0	1	0	0	0	1	a_in
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	c_in
3	0	47	-1	0	0	3	1	0	0	0	2	a
4	0	47	-1	0	0	4	1	0	0	0	2	b
5	0	47	-1	0	0	5	1	0	0	0	2	c
6	0	47	-1	0	0	6	2	0	0	0	2	k
7	0	47	-1	0	0	8	1	0	0	0	2	s
8	0	12	0	2	0	9	2	0	0	2	1	=
9	0	12	2	2	0	11	2	0	0	2	1	=
10	0	12	4	2	0	13	2	0	0	2	1	=
11	0	1	6	2	0	15	1	0	0	2	1	+
12	0	57	-1	0	0	16	1	0	0	0	1	C0_
13	0	25	8	2	0	17	1	0	0	2	1	<
14	0	51	10	1	0	18	2	1	2	1	2	upl
15	1	3	11	3	1	20	1	0	0	3	1	*
16	0	12	14	2	1	21	2	3	0	2	2	=
17	2	1	16	3	2	23	1	0	0	3	1	+
18	0	12	19	2	2	24	2	3	0	2	2	=
19	3	54	21	2	3	26	1	0	0	2	1	l.o
20	0	53	23	2	3	27	1	0	0	2	1	a.o
21	0	57	-1	0	3	28	1	0	0	0	1	C2_
22	0	5	25	3	3	29	1	0	0	3	1	%
23	0	12	28	2	3	30	2	0	0	2	2	=
24	0	49	30	1	3	-1	0	0	0	1	0	stop
25	0	48	31	1	3	-1	0	0	0	1	0	s_out

Таблица 2

Файл связей (ФС)

N	JSD	SPJD	SNWIH	SNWHO	JWD	WPJD	WNWHO	WNWIH
0	1	0	0	0	-1	8	0	0
1	-1	3	1	1	-1	9	0	0
2	3	1	0	0	-1	10	0	0
3	-1	4	1	1	-1	8	1	1
4	5	2	0	0	-1	9	1	1
5	-1	5	1	1	-1	10	1	1
6	7	8	0	0	7	16	1	1
7	-1	9	0	1	-1	18	1	1
8	9	11	0	0	-1	23	1	1
9	-1	12	0	1	10	11	0	0
10	-1	13	0	0	-1	15	0	0
11	12	8	0	0	12	11	1	0
12	13	10	0	1	-1	17	0	0
13	-1	14	0	2	14	15	1	0
14	15	6	1	1	-1	17	1	0
15	-1	15	0	0	-1	13	0	0

Таблица 2 (окончание)

Файл связей (ФС)

N	JSD	SPJD	SNWIH	SNWHO	JWD	WPJD	WNWHO	WNWIH
16	17	9	0	0	-1	13	1	0
17	18	10	0	1	-1	14	0	0
18	-1	14	1	2	19	15	2	0
19	20	6	1	1	-1	17	2	1
20	-1	17	0	0	-1	16	0	0
21	22	18	1	1	22	19	0	1
22	-1	16	1	0	-1	20	0	0
23	24	18	0	1	-1	18	0	0
24	-1	16	0	0	25	19	1	1
25	26	20	0	0	-1	20	1	0
26	27	21	0	1	-1	22	2	0
27	-1	19	0	2	-1	22	0	0
28	29	7	1	1	-1	22	1	0
29	-1	22	0	0	-1	23	0	0
30	-1	23	1	0	31	24	0	1
31	-1	23	0	0	-1	25	0	0

Символ 3 – представляет собой этап синтеза Си-графов с сохранением эквивалентности формата СВМ и результирующего графового представления. Данный этап можно также разделить на ряд шагов:

1. Размещаются операторы, не имеющие сопряженных связей. В нашем примере это операторы, записанные в строках 0÷7, 12 и 21. Если все они не помещаются на одном уровне (вертикали экрана), то оставшиеся переносятся на следующий уровень (константы с номерами 12 и 21).

2. На очередной уровень помещаются операторы, для которых их сопряженные операторы уже размещены. Исключение составляют лишь операторы, один из сопряженных операторов которых находится в циклической ветви и еще не размещен. Такие операторы также могут быть размещены на текущем уровне.

3. На последнем уровне размещаются оператор stop и выходные операторы.

4. Используя формат СВМ, проводятся связи между операторами.

Рис. 2 иллюстрирует результат визуализации Си-программы, заданной текстом 1 в виде Си-графа.

Выводы

1. Разработанная методика, представленная в данной статье, обеспечивает полную формализацию перехода от текстового представления Си-программы к числовому формату сопряженно-внешних множеств и от формата сопряженно-внешних множеств к Си-графу, сохраняя логическую эквивалентность Си-программы, ее формата СВМ и Си-графа.

2. Принципиальным отличием разработанной методики от известных систем визуализации, применяемых при проектировании цифровых устройств, является использование числового формата, что обеспечивает потенциальные возможности сокращения сроков проектирования.

3. Разработанные средства формализации синтеза числовых и графических спецификаций Си-программ можно рассматривать как основу для разработки инструментальных средств перспективных систем автоматического проектирования параллельных цифровых устройств.

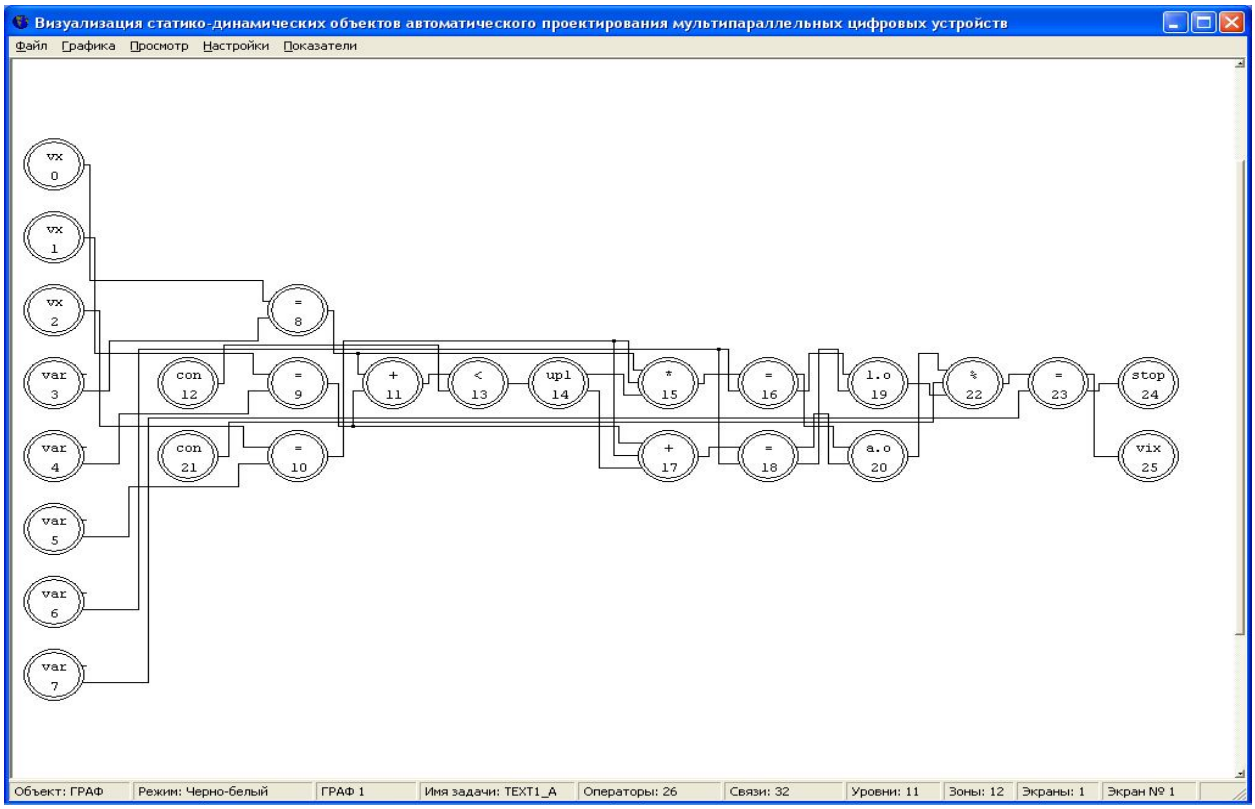

Рис. 2. Си-граф разветвляющейся программы

Литература

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

2. Корнеев В.В. Параллельные вычислительные системы. – М.: Нолидж, 1999. – 320 с.

3. Поляков Г.А. Основы построения и автоматического проектирования самоорганизующихся систем параллельной цифровой обработки информации и повышение эффективности комплексов радиолокационного вооружения ПВО / Под общ. ред. проф. В.К. Стрельникова. – Х.: ВИРТА ПВО, 1986. – 571 с.

4.Поляков Г.А. Проблемы создания систем совместного автоматического проектирования аппаратно-программных средств для мультипараллельной цифровой обработки данных. // 1-й Международный радиоэлектронный Форум "Прикладная радиоэлектроника. Состояние и перспективы развития". МРФ–2002. Ч. 2. – Х.: АН ПРЭ, ХНУРЭ. – 2002. – 656 с.

5. Polyakov G. The Hard-and-Soft Automatic Design of Self-Organizing Adaptive Systems. Radioelectronics&Informatics. – 2003. – № 3.

6. Поляков Г.А., Онищенко В.В. Визуализация статико-динамических объектов автоматического проектирования мультипараллельных цифровых устройств // Системы обработки информации. – Х.: ХВУ. – 2004. – Вып. 7 (35). – 240 с.

7. Кривуля Г.Ф., Хаханов В.И. Новые информационные технологии проектирования цифровых систем. // 1-й Международный радиоэлектронный Форум "Прикладная радиоэлектроника. Состояние и перспективы развития". МРФ – 2002. Ч. 2. – Х.: АН ПРЭ, ХНУРЭ. – 2002. – 656 с.

8. The EDN System Design Series, Part 1. – P. 24, Part 2. – P. 40. – 2001.

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

Поступила в редакцию 05.01.2005

Рецензент: д-р техн. наук, проф. Г.А. Поляков, Академия наук Прикладной Радиоэлектроники, Харьков.