МНОГОФУНКЦИОНАЛЬНЫЕ МОДЕЛИ КОМПОНЕНТОВ НЕОДНОРОДНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Б. И. Борде
Красноярский государственный технический университет, Россия
e-mail: borde@kgtu.runnet.ru

Models of heterogeneous computing system (HCS) components are examined at different levels of abstraction. Multifunction models for a number of applications are proposed accounting for both abstract and specific types of components. The multifunction models allow once only descriptions of the use of HCS in a number of applications.

Введение

На верхнем уровне абстракции (рис. 1) рассматриваем функциональные модели компонентов в целом, которые называются макромоделями компонентов и описываются функциями выходов и переходов. В различных системах моделирования и проектирования функциональные модели компонентов представляются в различных формах и на различных языках [1, 2].

На следующем уровне абстракции находятся структурные модели, в которых отражается внутренняя структура компонентов. Такие модели будем называть микромоделями. На этом уровне можно грубо оценить ресурсы.

Одна и та же структура может быть реализована с помощью различных принципов действия. Физические принципы действия компонентов неоднородных вычислительных

Рис. 1. Иерархия технических решений на различных уровнях абстракции.

систем подробно описаны в [1–4] и будут рассмотрены только при оценке ресурсов или оптимизации.

Зная принцип действия, с учетом уровня техники и технологии можно перейти к уровню технического решения. Техническое решение должно быть выполнимым. После получения технического решения возможно изменение параметров решения с целью повышения качества изделия.

Определив параметры технического решения, можно перейти к конструкции компонента. Например, в зависимости от рассеиваемой кристаллом мощности ему будет соответствовать корпус. На конструкцию существенное влияние оказывает внешняя среда (условия эксплуатации). Таким образом, на верхнем уровне абстракции находятся функциональные модели компонентов, а на нижнем уровне — конструкции компонентов и их геометрия. Геометрические модели компонентов используются при конструировании [1–3]. Абстрактным типам компонентов соответствуют только функциональные модели, а конкретным типам компонентов наряду с функциональными моделями — геометрические и ресурсные модели компонентов. Предлагаются многофункциональные модели для множества приложений с учетом абстрактных и конкретных типов компонентов и однократного описания неоднородных вычислительных систем.

Описание проектного решения, достаточное для автоматизированного ввода и интерпретации формальной системой, называется формализованным заданием, широко используемым в САПР. Формализованные задания (FZ) состоят из множества разделов, а разделы (Pi) — из предложений соответствующего языка:

$$\mathrm{FZ} =< \mathrm{P1,P2...PN} > .$$

Каждый раздел синтаксически и семантически однороден и соответствует отношению. В качестве примера рассмотрим исследовательскую САПР COD, в которой формализованное задание состоит из разделов (табл. 1).

В разделе INPUT описываются отношения между номерами цепей, типом сигнала, тактами начала и конца изменения сигнала.

В разделе UNIT описываются отношения между типами и номерами компонентов, номерами цепей, подключенных к выходам и входам, причем описание производится покомпонентно в виде многофункциональных моделей компонентов.

Модели компонентов могут использоваться на различных уровнях абстракции. Например, функциональные модели могут применяться для автоматического анализа поведения объекта; модели параметров компонентов — для оценки вариантов реализации устройства. Ряд моделей компонентов используется для интерфейса с системами конструкторского проектирования, находящимися на другом уровне абстракции.

Рассмотрим многофункциональные модели компонентов.

Т а б л и ц а 1. Разделы формализованного задания

Имя раздела	Описание раздела
INIT	Начальная установка
INPUT	Описание внешних воздействий
UNIT	Описание устройства
CTRL	Управление
MOD	Модели новых компонентов

1. Модели компонентов

Модели компонентов могут быть функциональными, геометрическими. Модель геометрии компонента может соответствовать символу на принципиальной схеме или геометрии конструктива.

Функциональные модели компонентов MFComp (табл. 2) состоят из разделов PMF:

$$\text{MFComp} =< \text{PMF1,PMF2,..., PMF5} > .$$

Т а б л и ц а 2. Разделы функциональной модели компонента

Имя раздела	Имя раздела (метка)	Описание раздела
1	FINIT	Проверка допустимости параметров, размещение структур данных и их заполнение
2	FTYP	Контроль типов компонентов и выбор возможных реализаций
3	FS	Восстановление состояний компонентов
4	FOUT	Формирование и контроль выходных сигналов
5	FEND	Освобождение ресурсов

Т а б л и ц а 3. Модели компонентов

№	Наименование	Имя процедуры	Имя таблицы параметров
1	Генерация цифровых сигналов	SIGNAL	HSIGN
2	Управление многовариантным анализом	IPRINT	HIPRINT
3	Управление процессом анализа	APRINT	HAPRINT
4	Обобщенный логический элемент	LO	HLO
5	Триггер	MTJK	HMTJK
6	Мультиплексор	MX	HMX
7	Узел мультиплексора многоразрядного	MMX	HMMX
8	Последовательный сумматор-счетчик	MCT	HMCT
9	Последовательный сумматор реверсивный	MCTR	HMCTR
10	Параллельный интерфейсный узел	MIR	HMIR
11	Последовательный интерфейсный узел	MIS	HMIS
12	Модель микроЭВМ	MKM	HMKM
13	Модель процессора	MPU	HMPU
14	Процессор аналоговый	MPA	HMPA
15	Пороговый элемент	PE	HPE
16	Компаратор аналоговый	CA	HCA
17	Цифроаналоговый преобразователь	DAC	HDAC
18	Аналого-цифровой преобразователь	ADC	HADC
19	Мультиплексор универсальный	MXU	HMXU
20	Мажоритарный элемент	MAJ	HMAJ
21	Регистр последовательного приближения	RPP	HRPP
22	Узел оперативной памяти	RAM	HRAM
23	Кодер и декодер	DC	HDC
24	Сумматор цифровой	SUMD	HSUMD
25	Таймер	TIM	HTIM

Для каждого класса компонентов разрабатываются модели с общей структурой данных. В табл. 3 сведены основные классы компонентов, имена процедур и имя таблицы синтаксиса и семантики параметров. В табл. 4–6 приведены синтаксис и семантика основных классов компонентов.

Обязательными в общей структуре данных являются тип компонента, его номер и данные, соответствующие именам выходных и входных цепей. Сначала описываются номера выходных, а затем входных цепей. Состояние сигналов, номера которых соответствуют номерам цепей, хранится в глобальной структуре данных. Для входных цепей процедура только считывает информацию о состоянии сигнала, а для выходных цепей процедура производит запись новых состояний.

С целью сокращения объема описания для каждого класса компонентов используются не одна, а две основные структуры данных. В первой используются массивы номеров входных или выходных цепей. В этом случае на порядок следования номеров цепей отсутствуют ограничения. Для второй структуры указываются только номер цепи старшего разряда (минимальное значение номера) и количество цепей. Но в этом случае номера цепей должны быть расположены в порядке возрастания и процедура заполняет массив номеров цепей, освобождая от этого студента или инженера. Имена для первой точки входа в виде массива номеров цепей образуются из имени класса компонента и окончания NIN, а для второй точки входа — окончания MIN (минимальный).

Структуры данных для этих точек входа должны синтаксически отличаться, а для их использования в формализованное задание должны быть включены файлы описания точек входа. Например, для среды PL/1 файлы имеют имя, составленное из символа W и общего имени процедуры, а для C++ включается файл типа *.H. Примеры описаний для выбора точки входа процедуры моделирования универсального логического элемента приведены на языках PL/1:

```
DCL (LONIN ENTRY (CHAR (*), DEC FIXED (3), DEC FIXED (3),
DEC FIXED (3), (*)DEC FIXED (3)), LOMIN ENTRY (CHAR(*),
DEC FIXED (3), DEC FIXED (3), DEC FIXED (3), DEC FIXED (3),
DEC FIXED (3)));
DCL LO GENERIC (LONIN WHEN (....), LOMIN WHEN (....));
```

на C++:

```
void LO (char*typ, int nel, int noutp, int noutn, IntVector& jin);
void LO (char*typ, int nel, int noutp, int noutn, int ninmin, int nsign);
```

на JAVA:

```
public static void LO (String typ, int nel, int noutp, int noutn, int ninmin, int nsign);
public static void LO (String typ, int nel, int noutp, int noutn, int[] mnin);
```

Семантика параметров приведена в табл. 4.

Модель компонента состоит из ряда разделов и наряду с выполнением основных функций выдает диагностические сообщения.

В первом разделе объявляются в соответствии с классом компонентов структуры данных и независимо от используемой точки входа заполняются массивы номеров входных и выходных цепей. Желательно объявление массивов произвольной размерности с последующим определением размеров фактически переданных массивов по каждому измерению.

Во втором разделе процедуры с именем FTYP проверяется допустимость типа компонента и определяются тип и параметр выполняемой функции. Для недопустимого типа компонента выдается сообщение об ошибке. После проверки типа компонента оператор с меткой FPAR запоминает тип и номер компонента путем вызова функции FPAR. Тип и

Т а б л и ц а 4. Синтаксис и семантика параметров обобщенной модели логического элемента LO

№	Идентификатор	Тип данных PL / 1	Тип данных C, C++, JAVA	Тип данных ADA	Семантика	Примечания
1	TYP	CHAR (*)	char (C++) String (Java)	character	Тип элемента: min и max аналогового сигнала “MIMAA”, для цифровых сигналов ключи приведены в примечании	Ключи ‘&’, ‘|’, ‘ЛА’, ‘ЛЕ’, ‘ЛИ’, ‘ЛП’, ‘М2’,
2	NEL	DEC FIXED(3)	int	integer	Номер элемента	
3	NOUTP	DEC FIXED(3)	int	integer	Выход неинвертирующий	
4	NOUTN	DEC FIXED(3)	int	integer	Выход инвертирующий	
Отличие для точки входа LONIN						
5	NIN	(*) DEC FIXED(3)	IntVector (C++) int[] (Java)	integer	Массив номеров входов	
Отличие для точки входа LOMIN						
6	NMIN	DEC FIXED(3)	int	integer	Номер входа минимальный	
7	NSIGN	DEC FIXED(3)	int	integer	Количество входов	

Т а б л и ц а 5. Синтаксис и семантика процедуры генерации входных сигналов

№	Идентификатор	Тип данных PL / 1	Тип данных C, C++, JAVA	Тип данных ADA	Семантика	Примечания
1	NSIGNAL	DEC FIXED(3)	int	integer	Номер сигнала	
2	STRBIT	BIT (*)	BitString	Символьная строка	Битовая строка	
3	NTMIN	DEC FIXED(6)	int	integer	Номер такта начала изменения сигнала	
4	NTMAX	DEC FIXED(6)	int	integer	Номер такта окончания изменения сигнала	

номер компонента в дальнейшем используются для автоматической оценки параметров устройства с помощью процедуры FPARSUM. Информация о параметрах компонентов хранится в таблице UIPCAD.DBT-UIPCAD.DBM, где также имеются параметры конкрет-

Т а б л и ц а 6. Синтаксис и семантика моделей мультиплексоров цифровых и аналоговых сигналов (MX и MXU)

№	Идентификатор	Тип данных PL / 1	Тип данных C, C++, JAVA	Тип данных ADA	Семантика	Примечания
1	TYP	CHAR (*)	Char (C++) String (Java)	character	Тип мультиплексора 'КП' предназначен для коммутации цифровых сигналов, а 'КН' — для аналоговых	MX, MXU 564КП1 564КП2 К561КП1 К561КП2 К1561КП1 К1561КП2 К590КН1 К590КН6 К591КН3
2	NEL	DEC FIXED(3)	int	integer	Номер элемента	
3	NOUTP	DEC FIXED(3)	int	integer	Номер прямой	
4	NOUTN	DEC FIXED(3)	int	integer	Номер выхода инвертирующий	
5	NC	DEC FIXED(3)	int	integer	Номер входа разрешения	
6	NA	(*) DEC FIXED(3)	int	integer	Массив номеров цепей передачи адреса	Размерность массива должна соответствовать типу мультиплексора
7	JIN	(*) DEC FIXED(3)	IntVector (C++) int[](Java)	integer	Массив номеров входных цепей	
Отличительные особенности параметров точки входа MXMIN						
8	NMIN	DEC FIXED(3)	int	integer	Номер входа минимальный	
9	NSIGN	DEC FIXED(3)	int	integer	Число входных цепей	

ных типов компонентов, для которых известны: потребляемая мощность, стоимость, площадь, масса. Абстрактные типы компонентов имеют произвольную разрядность и произвольное число входов-выходов, но не имеют конкретных параметров. Абстрактные типы компонентов заменяются конкретными при определении элементной базы.

Третий раздел с именем FS служит для оценки состояния компонента, а четвертый раздел с именем FOUT — для вычисления новых значений выходных сигналов. Оценивается допустимость полученных выходных сигналов, и в случае необходимости запрещенные

значения заменяются на допустимые. Например, для троичной модели уровни сигналов обозначены: C0 — уровень нуля, C1 — уровень единичного состояния, CF — фронт, CZ — обозначение запрещенного состояния. Использование таких обозначений облегчает переход из одной языковой среды в другую. Именем завершающего раздела является FEND.

2. Представление многофункциональных моделей

Пример процедуры обобщенного логического элемента для цифровых и аналоговых сигналов на языке C++ с общим именем LO и массивом номеров цепей или объединенных упорядоченных в порядке возрастания номеров цепей с параметрами номера цепи старшего разряда и количества входных цепей приведен ниже:

```
  # include <vextcpp.h>
  void LO(char *typ,int nel,int noutp,int noutn,int nmin,int nsign)
  { IntVector jin (nsign);
    for (int i=0; i<nsign; i++) jin[i] = i + nmin;
  LO(typ,nel,noutp,noutn,jin); }
  void LO(char *typ,int nel,int noutp,int noutn,IntVector& jin)
  { int nsign,nmin,i,j;
  unsigned int newi;
  int mnin=jin.Dim(); IntVector nin(mnin);
  static char *typs[] = {"& ","ЛА","ЛИ","1","ЛЕ","ЛЛ","М2","ИП"};
  const int NTYP = sizeof(typs)/sizeof(*typs);
static struct logic
    { unsigned ex1 : 2;
      unsigned ex2 : 2;
      unsigned ex3 : 2;
      unsigned ex4 : 2;
    } ft[NTYP] =
{ {C0,C0,C0,C1}, {C0,C0,C0,C1}, {C0,C0,C0,C1}, {C0,C1,C1,C1},
  {C0,C1,C1,C1}, {C0,C1,C1,C1}, {C0,C1,C1,C0}, {C1,C0,C0,C1} };
    unsigned int newo[NTYP] =
        { C1,C1,C1,C0,C0,C0,C0,C0 };
    nin = jin;

FTYP: // Begin MINMAX
  if ( strstr(typ,"MIMAA")!=NULL )
  { int ia; float amin,amax;
  amin = as[nin[0]].ANex; amax = as[nin[0]].ANex;
  for (ia=0; ia<mnin; ia++)
  { if (as[nin[ia]].ANex > amax) amax = as[nin[ia]].ANex;
```

```
        if (as[nin[ia]].ANex < amin) amin = as[nin[ia]].ANex; }
      as[noutp].ANew = as[noutp].ANex = amax;
      as[noutn].ANew = as[noutn].ANex = amin;
    } // end MINMAX
    else { for ( j=0; j<NTYP ; j++)
           if (strstr( typ,typs[j] ) != NULL )
           newi = newo[j];
           break; }
    if ( j>>=NTYP )
         { fprintf(stderr,"В процедуре LO отсутствует тип % s,
           №  элемента: % d\n", typ,nel);
           return; }

FPAR: if (nt == 0 & & ppt) { Fpar (typ,nel); }

FOUT: for ( i=0 ; i<mnin ; i++ )
        { if ( out[nin[i]].Nex == CZ )
          out[nin[i]].Nex = CF;
          newi = (
          (ft[j].ex1 & ~newi & ~out[nin[i]].Nex) |
          (ft[j].ex2 & ~newi & out[nin[i]].Nex) |
          (ft[j].ex3 & newi & ~out[nin[i]].Nex) |
          (ft[j].ex4 & newi & out[nin[i]].Nex) ); }
        newi = newi==CZ ? CF:newi;
        out[noutp].New = newi;
        out[noutn].New = (newi == CF) ? CF : ~newi;
        } } // end of LO
```

Примеры формализованных заданий с использованием синтаксиса различных языков программирования приведены в работе [2] и на оптическом диске. Результаты анализа отображаются в виде временных диаграмм цифровых и аналоговых сигналов, а оценка ресурсов и критериев эффективности для всех вариантов заносится в файл сообщений типа msg. Геометрические модели компонентов позволяют получать трехмерное отображение модуля ЭВМ или вычислительной системы в среде виртуальной реальности [2, 4], отображать принципиальную схему с помощью программы просмотра для Интернет. Специализированные модели компонентов позволяют получать командные файлы построения схем в САПР PCAD, ORCAD, CADDy, CATIA [1, 2]. Модели компонентов для различных приложений объединяются в библиотеки и подключаются в зависимости от требуемого результата работы.

Заключение

Рассмотрены модели компонентов на различных уровнях абстракции неоднородных вычислительных систем. Многофункциональные модели компонентов позволили использовать однократное описание НВС в виде формализованного задания для множества приложений. Логические операции и коммутация выполняются как над двух- и трехзначными

моделями сигналов, так и над бесконечнозначными (аналоговыми). Для конкретных типов компонентов автоматически оцениваются ресурсы. Предложенные многофункциональные модели компонентов позволяют расширить множество приложений при единственном описании системы.

Список литературы

[1] Норенков И.П. Основы автоматизированного проектирования: Учебник для вузов. М.: Изд-во МГТУ им. Н.Э. Баумана, 2000. 360 с.

[2] Борде Б.И. Основы САПР неоднородных вычислительных устройств и систем: Учеб. пособие. Красноярск: Изд-во КГТУ, 2001. 351 с.

[3] Корячко В.П., Курейчик В.М., Норенков И.П. Теоретические основы САПР. М.: Энергоиздат, 1987. 400 с.

[4] Титтел Э., Сандерс К., Чарли С., Вольф П. Создание VRML-миров: Пер. с англ. М.: Издательская группа BHV, 1997. 320 с.

Поступила в редакцию 2 ноября 2005 г.