КОНВЕРТИРОВАНИЕ ОПИСАНИЙ ПРОЕКТОВ В САПР НА ЛИНГВИСТИЧЕСКОМ УРОВНЕ

Пилипушко Е.М.
Кафедра МОП ЭВМ, ТРТУ
lena@mop.tsure.ru

Abstract

Pilipushko E. Converting of CAD project descriptions on the linguistical level. *Given article is denoted to the study and method development of converting the project descriptions in the linguistical environment a CAD. Linguistical environment a CAD as a collection of facilities of exchange by information between separate systems a CAD on the semantic level is considered. General feature of compiling process is described. Logical scheme of process of converting on the base of inverse polish record is offered. Statement of the converting problem on base of semantic atoms for description languages of the projects in CAD is formulated. It reflects an ascendancy of semantics in CAD description languages and allows to add new semantic constructions of language without problems.*

1. Лингвистическая среда САПР СБИС

Теоретические исследования в области разработки средств лингвистического обеспечения являются одним из важнейших элементов теории САПР.

На основе методов теории систем САПР высокого уровня следует рассматривать как сложные системы [1]. Они качественно отличаются от обычных технических и программных систем способами и средствами обмена, накопления и хранения информации. Информация, передаваемая в САПР как в системе человеко-машинных коммуникаций, обладает определенным смыслом. Он инвариантен относительно способов ее кодирования и каналов связи человек – машина, т.е. технических средств САПР.

В САПР высокого уровня приемником и источником (передатчиком) информации попеременно являются либо пользователь САПР, либо программная система - ее отдельные подсистемы или пакеты прикладных программ (ППП). Смысл передаваемой информации и ее интерпретация адресатом составляют соответственно семантику и прагматику сообщений. Таким образом, комплексная САПР как сложная система имеет семиотическую (языковую) природу информационных связей между отдельными подсистемами и другими САПР. Качественное отличие САПР высокого уровня от программных систем и пакетов прикладных программ заключается в том, что обмен информацией в ней между отдельными частями происходит на семантическом уровне. В обычных программных системах и пакетах информационные связи осуществляются на синтаксическом уровне.

Процесс проектирования в САПР осуществляется в рамках некоторой информационной и лингвистической среды данной системы. Лингвистическая среда САПР – это совокупность всех средств семантического обмена информацией между разработчиками, пользователями, отдельными подсистемами САПР и ППП. Однако лингвистическая среда конкретной САПР зачастую является закрытой системой для других, что приводит к невозможности передачи проекта из одной САПР, в другую без дополнительных затрат на его новое описание в лингвистической среде новой САПР.

Эту процедуру по своим затратам можно сравнить практически с созданием нового проекта.

С начала 70-х годов стала актуальной проблема создания стандартного средства документации схем и алгоритмов цифровой аппаратуры (ЦА). Этим средством явился язык VHDL. Стандартизация входных языков и внутренних интерфейсов подсистем САПР, в том числе и на базе VHDL, создает общую коммуникационную среду для САПР, позволяет упростить стыковку продуктов различных фирм, обмен библиотеками моделей компонент и проектов, модернизацию отдельных подсистем САПР. Как и все универсальные средства, VHDL как язык представляется весьма громоздким и избыточным, что затрудняет его реализацию и изучение в полном объеме. Естественно выделение подмножеств языка для отдельных групп пользователей. VHDL-модели современных БИС представляют собой большие программы. Например, VHDL-модель МП Z80, сравнимого по числу вентилей с МП К580, состоит из 12000 строк исходного текста, и ее трансляция на рабочей станции типа DN4500 с 16 Мбайт ОЗУ с использованием VHDL-компилятора фирмы VANTAGE выполняется более 5 минут, требуя более 70 Мбайт дискового пространства. Коэффициент замедления при работе модели более 1000 /2/. Если продолжить рассмотрение проблем на примере модели МП К580, то более близкая к реальности модель, учитывающая работу устройства управления с точностью до машинных циклов и микротактов, потребовала больших усилий по весьма неполному анализу документации даже на этот МП. Исходный VHDL-текст модели занимал более 64Кбайт памяти.

Так же, следует отметить, что к настоящему времени разработано множество библиотек базовых матричных кристаллов, банков данных типовых элементов, которые спроектированы в лингвистической среде соответствующей САПР и их перевод на любой другой язык описания требует огромных затрат времени и средств.

Учитывая все это, можно отметить, что для реального создания и моделирования проекта СБИС на языке VHDL необходима прежде всего САПР, поддерживающая ввод проектов на данном языке, большая затрата времени для создания громоздкого описания проекта СБИС и его моделирования, а так же мощные вычислительные средства. Учитывая, что язык VHDL в настоящее время является стандартом для описания и моделирования ЦА, естественно выделение подмножества языка, необходимого и достаточного для проведения проектов в САПР, на основе которого можно построить отображение из любого языка описания. Качественное отличие этого отображения заключается в том, что обмен информацией происходит на семантическом уровне и инвариантен относительно способов ее кодирования.

Таким образом, построив проекцию $P: L_1 \rightarrow L_2$ на основе эквивалентных семантических конструкций, можно автоматически конвертировать модель с языка описания на VHDL, освобождая проектировщика от рутинной работы по переводу проекта. Полученная VHDL-модель будет свободна от каких бы то ни было синтаксических ошибок, и может быть передана в любую САПР, поддерживающую проектирование на языке VHDL, в том числе и кремниевый компилятор.

2. *Общая характеристика процесса компиляции*

Основной задачей компиляции является переработка текста входной программы в эквивалентную последовательность операторов выходной программы. Логически трансляция содержит две основные части: анализ и синтез. Первая часть распознает во входном тексте конструкции абстрактной программы, а вторая реализует собственно перевод – генерацию эквивалентной ей выходной программы. В классической интерпретации, описанной во многих источниках [3,4,5,6] анализ подразделяется на три этапа: лексический анализ, синтаксический анализ и контекстный анализ.

Первый этап процесса компиляции – лексический анализ должен преобразовать исходную программу в последовательность внутренних кодировок и набор таблиц лексем, образующих лексическую свертку входной программы. Он всегда предшествует синтаксическому анализу и заключается в выделение лексических единиц (базовых элементов) - лексем, из которых строится исходная программа. Большинство лексем в языках программирования можно сгруппировать в следующие классы: идентификаторы, служебные слова, числа, операции, однолитерные разделители и двулитерные разделители.

Данной классификации лексем достаточно, чтобы построить лексический анализатор [3,5]. После лексического анализатора программа представляется последовательностью лексических единиц. Для дальнейшей обработки программы, ее необходимо представить в виде таблиц лексем (постоянных для служебных слов и разделителей и формируемых для идентификаторов и констант). Лексемы представляются парой "класс-код", где класс – это признак принадлежности к определенному классу лексем, а код – номер лексической единицы программы в классе. Такое представление программы после лексического анализа называется стандартным или таблицей стандартных символов.

Достоинства такого решения:

- простое обращение к любому стандартному символу, вследствие их фиксированной длины;
- возможность расширения таблиц идентификаторов и констант для включения любой информации об этих лексических единицах (типа, длины, адреса, системы счисления и т.д.).

Таким образом, на этапе лексического анализа исходная программа преобразуется в последовательность стандартных символов, и формируются таблицы идентификаторов и констант. Выходом лексического анализа является лексическая свертка транслируемой программы. Если в программе не существует меток для возврата назад, то лексический анализ осуществляется за один просмотр исходного текста программы.

Наиболее сложные задачи возникают на следующем этапе – этапе синтаксического анализа. Это распознавание и анализ синтаксических единиц программы и их представление (интерпретация) в эквивалентной форме. Часто для этого используется некоторая промежуточная форма. Например, матрица интерпретации /3,4,5/. Ее построение идет параллельно с анализом конструкций исходной программы и выявлением синтаксических ошибок. После синтаксического анализа может быть выполнена машинно-независимая оптимизация, например удаление одинаковых строк из МИ.

Далее выполняется этап распределения памяти. Для каждой переменной из таблиц идентификаторов и констант отводится соответствующее поле памяти, и им присваиваются относительные адреса, а в матрицу интерпретации вводятся команды выделения областей памяти для переменных и констант.

Следующим этапом является генерация кода, который обычно выполняется за два прохода, что связано с разрешением ссылок вперед.

На этапе генерации кода достаточно для каждой команды интерпретации реализовать последовательность команд выходного языка (кодовую продукцию), и тогда каждая строка матрицы интерпретации может быть заменена кодовой продукцией, настроенной в соответствии с использованными в данной строке матрицы операндами. То есть, необходимо построить проекции между семантически эквивалентными конструкциями команд интерпретации и выходного языка. На рисунке 1 показаны логические фазы процесса компиляции.

Таким образом, весь процесс компиляции можно представить в виде суперпозиции следующих отображений:

р: $L_1 \rightarrow$МИ; р: МИ$\rightarrow$ОМИ; р: ОМИ$\rightarrow$ПМИ; р: ПМИ$\rightarrow L_2$,
где L_1 – входной язык,
МИ – матрица интерпретаций,
ОМИ – оптимизированная МИ,
ПМИ – полная МИ,
L_2 – выходной язык.

Рис.1 - Логические фазы процесса компиляции

3. *Особенности процесса конвертирования описаний проектов СБИС*

При конвертировании программ описаний СБИС с одного языка на другой лексический анализ аналогичен описанному ранее. Однако при проведении синтаксического анализа нет необходимости в построении матрицы интерпретации, так как не существует распределения памяти. Этапы 3 и 4, представленные на рис.1 исключаются, поэтому можно отказаться от построения матриц интерпретации и использовать для этого типы данных, эквивалентные конструкциям исходного языка описания. В данной статье предлагается в качестве промежуточного языка использовать обратную польскую запись (ОПЗ).

ОПЗ обладает двумя замечательными свойствами [10]:

- отсутствием скобок;
- возможностью вычисления выражения путем однократного просмотра слева направо.

В этом случае процесс трансляции (конвертирования) можно представить в виде суперпозиции следующих отображений:

р: $L_1 \rightarrow$ ОПЗ; р: ОПЗ $\rightarrow L_2$.

Этот процесс можно расчленить на фазы. Две первые фазы носят подготовительный характер.

Первая фаза заключается в построении таблиц ключевых слов, операций и разделителей, используемых для лексического анализа.
Вторая фаза состоит в построении таблицы приоритетов по принципам Дейкстры для всех конструкций языка, управляющей таблицы синтаксического анализатора и таблицы стандартных символов языка описания проектов на основе ОПЗ и атомарной семантической структуры.
Третья фаза включает разработку семантически эквивалентных кодовых продукций в виде элементарных структур (атомов) для выбранного подмножества конструкций языка VHDL, которое полностью включает возможные семантические структуры входного языка.
Как уже отмечалось ранее, передача информации при описании проектов в САПР осуществляется на семантическом уровне, поэтому построение кодовых продукций должно основываться на семантических атомарных конструкциях.

4. *Формальная постановка задачи конвертирования на основе семантических атомов для языков описания проектов в САПР*

Входной язык L_1 должен быть задан контекстно-свободной (КС) грамматикой /5/ вида:
$G_1 = (V_T, V_N, R_G, C_0)$,
где V_T – множество терминальных символов,
V_N – множество нетерминальных символов, причем $V = V_T \cup V_N$ – алфавит,
R_G – множество правил вывода (продукций),
C_0 – начальный символ (аксиома), определяющий множество $L \subset V^*$ цепочек $\omega \in V_T^*$ (цепочки из терминальных символов) и выводимых из аксиомы $C_0 \in V_N$ путем применения правил из R_G (условие синтаксичности).
Обозначим через λ отображение множества семантических объектов $S_1(P_1)$ во входной язык L_1, рассматриваемый как множество предложений
$\lambda_1: S_1(P_1) \to L_1, \quad S_1(P_1) \neq \{S_e\}$,
где P_1 – предметная область САПР входного языка,
S_e – семантический объект, обозначающий отсутствие смысла.
Язык L_1 удовлетворяет условию семантичности /7/ в заданной предметной области P_1, если существует отображение
$\lambda_1^{-1}: L_1 \to S_1(P_1)$,
которое каждому предложению (слову) на входном языке $\omega \in L_1$ ставит в соответствие семантический объект $I \in S_1(P_1)$.
Будем считать, что конвертирование со входного языка L_1 предметной области P_1 в выходной язык L_2 есть отображение
$$\eta: L_1 \to {}^{\eta}L_2, \qquad (1)$$
задающее перевод ${}^{\eta}\varpi = \eta(\varpi) \in {}^{\eta}L_2$.
Для реализации безошибочного перевода цепочек из входного языка описания в выходной необходимо полное совпадение синтаксиса языка ${}^{\eta}L_2$ синтезируемого конвертором и языка L_2. Важно соблюдение равенства ${}^{\eta}\varpi = \varpi$ для всех возможных цепочек ${}^{\eta}\varpi$. Это условие означает возможность безошибочного синтаксического анализа синтезируемых конвертором цепочек ${}^{\eta}\varpi$ транслятором языка VHDL.
С языками L_1 и L_2 ассоциированы множества семантических объектов $S_1(P_1)$ и $S_2(P_2)$ соответственно, где P_2 - предметная область выходного языка L_2, и такие отображения λ_1 и λ_2, что $\lambda_1: S_1(P_1) \to L_1$ и $\lambda_2: S_2(P_2) \to L_2$. Очевидно, что $P_1 \subseteq P_2$ и $S_1(P_1) \subseteq S_2(P_2)$.

При конвертировании цепочки ϖ_1 в цепочку ϖ_2, необходимо распознать семантическую структуру цепочки ϖ_1 относительно $S_1(P_1)$, найти ее эквивалент в $S_2(P_2)$ и выразить его в языке L_2.

Пусть μ_2: $S_1 \rightarrow S_2$ – отображение семантических объектов предметной области САПР входного языка в семантические объекты предметной области САПР выходного языка. Тогда отображение

$$^{\eta}\varpi = \eta(\varpi) = \lambda_2(\mu_2(\lambda_1^{-1}(\varpi))), \qquad (2)$$

если $S_1 \cap S_2 \neq \varnothing$, где $S_2 = \mu_2(S_1)$ – условие частичного изоморфизма предметных областей (а, следовательно и их языков описания) САПР входного и выходного языков.

Условие $S_1 \cap S_2 \neq \varnothing$ -это условие возможности решения САПР выходного языка некоторой задачи из предметной области P_1. Соотношения (1) и (2) реализуют правила перевода из входного языка L_1 в выходной L_2.

Так как языки описания проектов в САПР имеют структурную организацию на семантическом уровне, то целесообразно осуществлять конвертирование на основе анализа семантических конструкций (атомов) этих языков.

В качестве способа отображения λ_1^{-1} предлагается использовать ОПЗ, построенные для каждого семантического атома языка L_1.

Итак, в данной статье проведены исследования лингвистической среды САПР, рассмотрены существующие методы решения задач компиляции на каждом из этапов. На основании полученных результатов предложена логическая схема конвертирования через ОПЗ и сформулирована постановка задачи конвертирования на основе семантических атомов для языков описания проектов в САПР.

Литература

1. Гридин В.Н. Теоретические основы построения базовых адаптируемых компонентов САПР МЭА/Под ред. Г.Г.Рябова - М.: Наука. Глав. ред. физико-математической литературы, 1989
2. Моделирование цифровых систем на языке VHDL. В 3 книгах. - М: Российский научно-исследовательский институт информационных систем, 1995.
3. Касьянов В.Н., Поттосин И.В. Методы построения трансляторов. Новосибирск: Наука, Сибирское отделение,1986-345с.
4. Хантер Р. Проектирование и конструирование компиляторов. М: Финансы и статистика, 1984-232с.
5. Льюис Ф., Розенкранц Д., Стирнз Р. Теоретические основы проектирования компиляторов. М: Мир,1979-656с.
6. Кауфман В.Т. О технологии создания трансляторов (проекционный подход). Программирование. 1978, №5, с. 36-44.
7. Берестовая С.Н. Об одном подходе к описанию языка программирования // Программирование. 1982, №2, с. 52-58.