УДК 519.7

П.Н. Бибило, В.И. Романов

Продукционная система оптимизации иерархических описаний логических схем

Описана продукционная система, предназначенная для оптимизации иерархически организованных описаний комбинационных логических схем. Комбинированные маршруты оптимизации рассматриваются в виде стратегий, представляющих собой совокупность продукций типа «если → то». В левой части продукции размещается условие ее применения, в правой – исполняемый программный модуль обработки логических описаний либо способ обхода иерархии проекта.

The production rule system for the optimization of hierarchically organized descriptions of combinatorial logic circuits is described. Composite routes of optimization are considered as the strategies represented by a set of productions such as «if → then». Each production consists of two parts: a precondition (or «IF» statement) and an action (or «THEN»). The precondition defines the applicability of the production. The actions is an executing program module of the processing logic descriptions or an algorithm for the project hierarchy traversal.

Описано продукційну систему, призначену для оптимізації ієрархічно організованих описів комбінаційних логічних схем. Комбіновані маршрути оптимізації розглянуто у вигляді стратегій, які являють собою сукупність продукцій типу «якщо → то». У лівій частині продукцій розташовано умову її застосування, у правій – виконуваний програмний модуль обробки логічних описів або спосіб обходу ієрархії проекту.

Введение. Размерности решаемых задач синтеза логических схем постоянно возрастают, так как успехи микроэлектронной промышленности позволяют размещать все больше логических элементов на одном кристалле. Исходные спецификации для проектирования логических схем, как правило, представляют собой иерархические описания на языках высокого уровня, например, *VHDL* (*Very high speed integrated circuits Hardware Description Language* – язык описания аппаратуры сверхскоростных интегральных схем). На этапе высокоуровневого синтеза осуществляется замена конструкций языка *VHDL* [1] элементами памяти (триггерами) и логическими уравнениями в булевом базисе – строится так называемое промежуточное *RTL*-представление логической схемы. Логические уравнения задают комбинационную часть логической схемы и подвергаются технологически независимой оптимизации перед этапом технологического отображения, т.е. этапом непосредственного синтеза логической схемы из элементов заданной технологической библиотеки проектирования.

В статье описана продукционная система *FLC*, предназначенная для оптимизации иерархически организованных описаний комбинационных логических схем, представленных на языке *SF* [2]. Этот язык соответствует уровню *RTL* (*Register Transfer Level* – уровень регистровых передач) языка *VHDL* и имеет средства конвертации в язык *VHDL* (и обратно). Оптимизация осуществляется на основе применения комбинированных методов, реализующих различные базовые оптимизационные приемы – минимизацию функций в классе дизъюнктивных нормальных форм (ДНФ), оптимизацию многоуровневых представлений систем функций на основе декомпозиции и разложения Шеннона и т.д. В системе *FLC* реализован продукционный подход [2] для управления последовательностями проектных процедур, предназначенных для их использования на этапе логического проектирования сложных функциональных блоков заказных цифровых сверхбольших интегральных схем (СБИС).

Представление данных

В качестве языка описания логических схем используется язык *SF* [2], ориентированный на иерархическое описание логической схемы, при этом уровень вложенности компонент не ограничивается. Схема на языке *SF* определяется последовательностью функционально-структурных описаний подсхем (блоков), из которых

состоит схема. Иерархия описания схемы представляется в виде дерева, вершине дерева соответствует отдельный блок.

Любой блок, соответствующий узлу дерева иерархии, выражается заданием связей входящих в него подсхем. Допускается использование двух форматов: формат *CONNECT*_2 – соответствует описанию логической схемы, состоящей из двух уровней, верхний уровень – это структурное описание схемы в виде соединения элементов (листовых блоков), второй (нижний) – функциональные описания листовых блоков; *CONNECT_N* – общий случай иерархии. Функциональные описания листовых блоков дерева иерархии описания представляют собой либо логические уравнения – скобочные выражения в булевом базисе И, ИЛИ, НЕ (*LOG*-формат), либо матричные формы представления систем булевых функций в виде ДНФ (*SDF*-формат). Система ДНФ булевых функций задается парой матриц: строки троичной матрицы *T* задают элементарные конъюнкции, входящие в систему ДНФ, в булевой матрице B единицы задают вхождения конъюнкций в ДНФ функций. При задании логических уравнений в *LOG*-формате символы операций имеют следующую интерпретацию: через * обозначено логическое И (конъюнкция), через + обозначено логическое ИЛИ (дизъюнкция), через ^ обозначено логическое НЕ (отрицание). Если головной блок описан на функциональном уровне, то в этом случае весь проект представляет собой один листовой блок. Возможности использования матричных форм и иерархических описаний – важные отличительные особенности языка *SF* от *RTL*-описаний, используемых в известном синтезаторе логических схем *LeonardoSpectrum* [1] и других промышленных синтезаторах.

Пример 1. На рис. 1 представлена иерархия описания логической схемы (рис. 2), функции некоторых листовых описаний которой приводятся ниже в виде текстовых файлов (листинги 1, 2) на языке *SF*. Заметим, что в листингах опущена служебная информация, не связанная с функционированием схемы и задающая такие ее характеристики, как: автор описания, имя проекта и т.д.

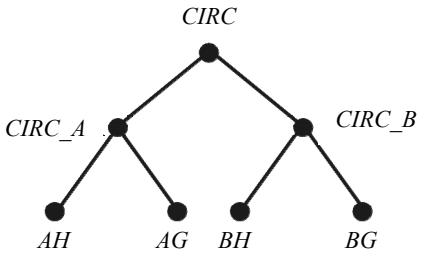

Рис. 1. Иерархия описания схемы *CIRC*

Листинг 1. Описание блока AH в формате *SDF* (*SDF*-лист).

```
INP
x1 x2 x3 x4
OUT
h1 h2 h3
FUNCTION
SDF
4 3 9
0111 001
0001 001
0011 001
0100 010
1100 010
0101 011
1010 100
1011 101
1101 110
END_SDF
```

Для данного примера все элементарные конъюнкции, заданные строками первой матрицы, есть полными – в них входят все литералы входных переменных данного блока.

Листинг 2. Описание блока ***AG*** в формате *LOG* (*LOG*-лист).

```
INP
h1 h2 h3 x5 x6
OUT
f1 f3
INTER
lam1 lam2 lam3
FUNCTION
LOG
5 2 0
f1=^h1*^h2*h3*lam1+^h1*h2*h3*lam2+^h1*
*h2*^h3*lam3+h1*h2*^h3*lam3;
f3=h1*^h2*h3*lam2+h1*h2*^h3*lam2+(^h1*
*h2*^h3+^h1*h2*h3+h1*^h2*^h3)*lam3;
lam1=^x5*x6;
lam2=^x5*x6+x5;
lam3=x5;
END_LOG
```

Булевы переменные *lam*1, *lam*2, *lam*3 являются внутренними для блока *AG* и помещаются в раздел *INTER LOG*-описания. Аналогично задаются и блоки *BH*, *BG* подсхемы *CIRC_B*. Не следует путать число уровней (каскадов) схемы и число уровней в иерархии описания схемы. Схема *CIRC*, изображенная на рис. 2, имеет два уровня, а число уровней в иерархии ее описания равно трем (см. рис. 1).

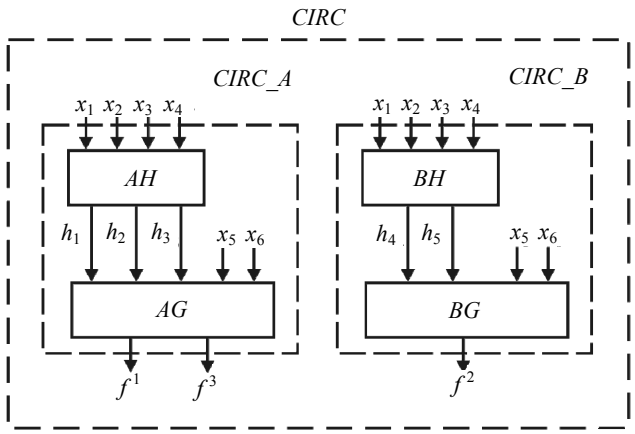

Рис. 2. Комбинационная схема *CIRC*

Программные модули

Программные модули (далее – модули), реализующие оптимизационные преобразования функциональных листовых описаний типа *SDF* и *LOG*, а также модули преобразования иерархии описаний образуют основу системы *FLC*.

Перечислим сначала модули оптимизации листовых описаний и изменения форматов функциональных описаний листьев проекта (табл. 1). Значения атрибутов, не указанные в третьем столбце табл. 1, полагаются равными нулю.

MINIMIZE – модуль минимизации (совместной или раздельной), реализует различные методы и алгоритмы минимизации [3, 4] систем булевых функций в классе ДНФ по различным критериям (числу элементарных конъюнкций, суммарному числу литералов в конъюнкциях, входящих в минимизированную систему ДНФ).

ELIMINATE – модуль, реализующий смену *LOG*-формата функционального описания листа проекта на *SDF*-формат. Данный модуль устраняет (элиминирует) промежуточные внутренние переменные.

Таблица 1

Модуль	Входные/выходные данные	Изменяемые значения атрибутов листьев проекта
MINIMIZE	*SDF*/*SDF*	*MIN* = 1, *TAB* = 0
ELIMINATE	*LOG*/*SDF*	
SDFLOG	*SDF*/*LOG*	
LOGSDF	*LOG*/*SDF*	
SDFROM	*SDF*/*SDF*	*TAB* = 1
TIE_BDD	*SDF*/*LOG*	*BDD* = 1
FACT_BMK	*SDF*/*SDF*, *LOG*/*SDF*	*FACT* = 1, *FACT_MIN* = 1
PRESIN	*LOG*/*LOG*	*ENLARG* = 1
MINRTL	*LOG*/*LOG*	*RTL* = 1

LOGSDF – модуль, реализующий смену формата *LOG* функционального описания листа проекта в формат *SDF* без устранения промежуточных переменных. В результате получается специальная матричная форма задания многоуровневых представлений систем функций.

SDFLOG – модуль, реализующий тривиальную замену *SDF*-описания *LOG*-описанием, при этом никаких оптимизирующих преобразований не выполняется.

SDFROM – модуль преобразования матричной формы системы ДНФ к виду, соответствующему таблице истинности.

TIE_BDD – модуль преобразования системы ДНФ булевых функций в многоуровневое *BDD*-представление [5]. *BDD* (*binary decision diagrams* – диаграммы двоичного выбора) строятся на основе разложения Шеннона. Разложением Шеннона полностью определенной булевой функции $f(x_1,\ldots,x_i,\ldots,x_n)$ по переменной x_i называется ее представление в виде $f(x_1,\ldots,x_n) = x_i f_1 \vee x_i f_0$. Функции f_1, f_0 называются коэффициентами разложения. Они получаются из функции $f(x_1,\ldots,x_n)$ подстановкой вместо переменной x_i константы 1 и 0 соответственно.

FACT_BMK – модуль факторизации [6], т.е. выделения общих частей конъюнкций и дизъюнкций в алгебраических скобочных выражениях. Для факторизации задаются два параметра. В результирующей факторизованной системе логических выражений число литералов в каждой многоместной конъюнкции не должно превосходить число p_k, а число литералов в любой многоместной дизъюнкции – число p_d.

PRESIN – модуль укрупнения [7] многоуровневых «мелкозернистых» *LOG*-описаний, к которым относятся *BDD*-представления и факторизованные выражения с небольшими значениями параметров p_k, p_d. Для работы модуля требуется указание параметра p – ограничение числа переменных, входящих в каждое «укрупненное» результирующее выражение.

MINRTL – получение булевой *RTL*-формы для логических выражений. Понятие такой *RTL*-формы в системе *FLC* отличается от *RTL*-формы в системе *LeonardoSpectrum*. Булевой *RTL*-формой системы функций назовем такое многоуровневое представление системы функций, в котором каждое выражение есть уравнение вида

$$y = x_1 \vee x_2,\ y = x_1 \,\&\, x_2,\ y = \overline{x},$$

в которых любая из булевых переменных x_1, x_2 может быть в прямой либо инверсной форме.

Перечислим программные модули декомпозиции функциональных описаний листьев проекта (табл. 2).

Таблица 2

Модуль	Входные/выходные данные	Устанавливаемые значения атрибутов для новых листьев проекта
DECOMP	*SDF*/*CONNECT_2*	*SF_TYPE = SDF*, *MIN* = 0
RAZ	*SDF*/*CONNECT_2*	*SF_TYPE = SDF*, *MIN* = 0
COVE	*SDF*/*CONNECT_2*, *LOG*/*CONNECT_2*	*SF_TYPE = LOG*
*PART_S*1	*LOG*/*CONNECT_2*	*SF_TYPE = LOG*
*PART_R*1	*LOG*/*CONNECT_2*	*SF_TYPE = LOG*

DECOMP – модуль декомпозиции системы ДНФ булевых функций [8] по входным переменным методом «тождественных отображений в пространстве промежуточных переменных» [9]. Для работы модуля требуется задание параметра p, ограничивающего число входных переменных, от которых зависят функции входного блока в двухэлементной сети, создаваемой по исходному декомпозируемому листовому описанию, заданному в *SDF*-формате. Во втором столбце табл. 2 в строке «*DECOMP*» указывается, что по исходному *SDF*-формату декомпозируемого листового описания строится *SF*-описание типа *CONNECT_2*, равенство *SF_TYPE = SDF* в третьем столбце данной строки свидетельствует о том, что листовые описания построенной подсети имеют формат *SDF*, равенство *MIN* = 0 свидетельствует о том, что данные *SDF*-описания не минимизированы в классе ДНФ.

RAZ – модуль дизъюнктивного разложения системы ДНФ булевых функций на основе группирования «коротких» конъюнкций. Конъюнкция называется короткой, если она содержит не более литералов булевых переменных. Модуль реализует метод, известный в литературе как «метод декомпозиции программируемых логических матриц (ПЛМ) в одноуровневую сеть ПЛМ, допускающих проводные дизъюнкции выходных полюсов» [9]. Модификация этого модуля под именем *RAZ*1 позволяет провести декомпозицию только «по термам».

*PART_R*1 – модуль группирования логических выражений (уравнений) в блоки, причем каждый из блоков имеет ограниченное число входов, выходов и общих конъюнкций, на которых заданы ДНФ функций блока. Реализованный алгоритм группирования экспериментально исследован в работе [10].

*PART_S*1 – модуль укрупнения уравнений, каждое из результирующих уравнений должно иметь не более p переменных [10]. Отличие данного модуля от модуля *PRESIN* [7] состоит в том, что модуль реализует алгоритм из работы [10], кроме того, результат работы модуля оформляется как структурное описание (формат *CONNECT_2*), листовые описания построенной сети при этом имеют формат *LOG*.

Программные модули преобразования иерархии описаний перечислены в таблице 3.

Таблица 3

Модуль	Входные/выходные данные	Устанавливаемые значения атрибутов для листьев проекта
UST	*CONNECT_N*/*LOG*	Нулевые значения всех атрибутов
UST_NEW	*CONNECT_N*/*LOG*	
2*CONNECT*	*CONNECT_N*/*CONNECT_2*	

UST – модуль устранения иерархии *SF*-описания проекта, содержащего несколько уровней [11]. Результирующее описание представляет собой один лист, функции которого даны в формате *LOG*. Редукция переменных, т.е.

устранение уравнений вида $y_1 = y_2$, не производится.

UST_NEW – модуль устранения иерархии *SF*-описания проекта с редукцией переменных [11]. Например, после применения процедуры редукции уравнение $lam3 = x5$; в описании блока *AG* (листинг 2) исчезнет.

2*CONNECT* – модуль преобразования иерархии *SF*-описания проекта, содержащего несколько уровней, в описание, содержащее ровно два уровня [11]. При таком преобразовании описания функций листовых описаний не изменяются. Так посредством установления иерархии структурных описаний выделяются условные границы подсхем, то данный модуль устраняет условные границы выделяемых подсхем, оставляя в описании только блоки (элементы) нижнего уровня описания.

Атрибуты

Одним из базовых понятий продукционно-фреймовой модели в системе *FLC* является понятие атрибута. Атрибуты задают набор существенных (с точки зрения организации процесса проектирования) свойств объектов, технологических ограничений и целей проектирования. Совокупность задействованных в системе атрибутов определяет пространство признаков, необходимых для принятия решений в диалоговом режиме работы системы *FLC*. Атрибуты представляются полным и сокращенным названием, типом и текущим значением. Полное название атрибута отражает его целевое назначение. Сокращенное название используется при записи продукций. Атрибуты могут быть двух типов: числовой (*integer*) и символьный (*text*). В качестве значения символьных атрибутов выступают строки произвольного текста (например, имена объектов). Значения числовых атрибутов – целые числа. По содержанию атрибуты могут быть классифицированы как атрибуты описаний объектов проектирования (*SF*-описаний), атрибуты программных модулей, вспомогательные атрибуты для управления выполнением продукций. Перечислим основные атрибуты, которые будут встречаться.

*Формат_*описания (*SF_TYPE*, <*text*>) – тип *SF*-описания элемента схемы. Его значениями могут быть: *LOG* – *SF*-описание схемы в виде системы логических уравнений; *SDF* – *SF*-описание схемы в матричной форме; *CONNECT_2* – *SF*-описание схемы в виде соединения элементов (листовых блоков); *CONNECT_N* – общий случай иерархического *SF*-описания.

Факторизованные_выражения – (*FACT*, <*integer*>) – в скобочных выражениях в булевом базисе в каждую многоместную конъюнкцию входит ограниченное число p_k литералов и в каждую многоместную дизъюнкцию входит ограниченное число p_d литералов булевых переменных.

Факторизованные_выражения_минимальные – (*FACT_MIN*, <*integer*>) – факторизованные выражения с минимальными значениями $p_k = 2, p_d = 2$.

Таблица_истинности – (*TAB*, <*integer*>) – частный случай формата *SDF*, когда система ДНФ представляет собой таблицу истинности системы функций.

Минимизированная_система_ДНФ (*MIN*, <*integer*>) – матричная форма листового описания минимизирована в классе ДНФ.

BDD-выражения (*BDD*, <*integer*>) – каждое из системы выражений, задающих многоуровневое представление системы функций, представляет собой формулу разложения Шеннона либо ее частный случай.

Укрупненные_выражения – (*ENLARG*, <*integer*>) – укрупненная (по числу входов) форма для «мелкозернистых» логических выражений, представленных в формате *LOG*.

Продукции и стратегии

В качестве условия α продукции $\alpha \to \beta$ выступает предикатная формула $P(P_1,P_2,\ldots,P_n)$, представляющая собой логическое выражение, связывающее операциями конъюнкции и дизъюнкции отдельные предикаты P_i и допускающее также операцию инверсии и скобки. Значениями предикатной формулы $P(P_1,P_2,\ldots,P_n)$ являются значения единица или нуль (*TRUE*/*FALSE*). Отдельный предикат P_i определяется выражением над атрибутами только одного типа. Заключение β (правая часть) продукции $\alpha \to \beta$ определяет выполняемое действие. Заключение продукции характеризуется своим типом:

«*Стратегия*» – реализация продукции такого типа предполагает переключение на анализ продукций стратегии, имя которой использовано в качестве заключения β;

«*Выбор объекта*» – реализация поиска блока проекта, обладающего свойствами, указанными в условиях продукции. Найденный блок делается активным и существует две возможные альтернативы управления: «*Анализ с начала*» – применимость продукций используемой стратегии проектирования будет проверяться с первой из них, «*Анализ продолжать*» – анализу на применимость будет подвергаться очередная (следующая по записи за анализируемой) продукция стратегии;

«*Программный модуль*» – реализация продукции связана с выполнением программного модуля, имя которого указано в качестве заключения;

«*Установка атрибута*» – реализация такого рода продукции предполагает присвоение атрибуту, имя которого выступает в качестве заключения, вычисленного значения условия применимости продукции.

Включение в систему механизма реализации стратегий обеспечивает возможность ее эксплуатации в автоматизированном режиме. Этот механизм основан на некоторых общих положениях. Реализация стратегии осуществляется для части проекта, определенной поддеревом иерархии, возглавляемого блоком, активным в момент старта стратегии. Реализация стратегии осуществляется пошагово. На каждом шаге выполняется анализ условий применимости всех (начиная с первой) ее продукций, осуществляемый последовательно в порядке их расположения в записи стратегии. Завершение шага предполагает возврат на начало стратегии. Анализ условий применимости продукции сводится к проверке истинности предикатной формулы из ее левой части. Соблюдение этого условия приводит к «срабатыванию» продукции – выполнению действия, определенного в правой части продукции, и завершению очередного шага реализации стратегии. Исключения составляют продукции типов «*Установка атрибута*» и «*Выбор объекта – Анализ продолжать*», срабатывание которых не приводит к завершению шага реализации стратегии. Продукции с заключением типа «*Установка атрибута*» или «*Выбор объекта – Анализ продолжать*» реализуются вне зависимости от соблюдения условия применимости, и после их реализации осуществляется анализ применимости следующей продукции рассматриваемой стратегии. Кроме того, если в правой части находится подчиненная стратегия, то при отсутствии условия применимости она выполняется только один раз в процессе выполнения головной стратегии. Реализация стратегии считается завершенной, когда при выполнении отдельного шага оказалась неприменима ни одна из ее продукций (кроме вышеуказанных исключений).

Для осуществления выбора объекта, обладающего заданными свойствами, все блоки иерархического *SF*-описания рассматриваются в виде линейного списка, упорядоченного в соответствии с обходом дерева иерархии проекта в глубину. Активным блоком в дереве иерархии проекта по завершению реализации стратегии является последний активизированный блок.

Примеры стратегий

По характеру своего назначения стратегии разбиты на несколько подмножеств, примерами которых являются:

– однотипная смена форм представления либо оптимизация листовых описаний в графе иерархии проекта;

– изменение существующей иерархии проекта, например, слияние проекта до одного узла (вершины) либо уменьшение числа уровней иерархии проекта;

– введение иерархии (расщепление) для листовых описаний;

– формирование новой иерархии для всего проекта либо его части.

Приведем примеры стратегий, которые, не изменяя иерархии проекта, позволяют оптимизировать либо изменять формат описания всех или некоторых листьев проекта. Для таких преобразований применяются модули, представленные в табл. 1. Заметим, что применение этих

модулей возможно только к одному отдельному листу проекта. Поэтому, если проект содержит много различных листьев, то такие стратегии позволяют обеспечить существенное сокращение «ручной» работы проектировщика, связанной с повторением многочисленных однотипных действий.

«*Представить все листья в SDF*»

```
1. {SF_TYPE="LOG"} → ELIMINATE
2. {SF_TYPE="LOG"} → Выбор объекта
(анализ с начала)
```

Выполнение данной стратегии позволяет найти и преобразовать каждый *LOG*-лист проекта в *SDF*-лист. Если целью является минимизация в классе ДНФ всех *SDF*-листьев, то можно выполнить следующую стратегию.

«*Минимизация SDF листьев*»

```
1. {SF_TYPE="SDF"} & {~MIN} → MINIMIZE
2. {SF_TYPE="SDF"} & {~MIN} → Выбор
объекта (анализ с начала)
```

Данная стратегия позволяет провести минимизацию в классе ДНФ всех листовых описаний, представленных в формате *SDF*: в стратегии ищется не минимизированный (должно быть истинным условие {*~MIN*}) *SDF*-блок (должно быть истинным условие {*SF_TYPE* = "*SDF*"}) и к такому блоку применяется модуль *MINIMIZE* минимизации в классе ДНФ.

«*Построить равноразмерные LOG листья*»

```
1. → Построить все листья в SDF
2. → Построить все листья в BDD
3. {SF_TYPE="LOG"} & {~ENLARG} →
→ PRESIN
4. {SF_TYPE="LOG"} & {~ENLARG} → Выбор
объекта (анализ с начала)
```

После выполнения первых двух продукций (а это безусловно выполняемые стратегии) получается проект, у которого все листовые описания представлены в форме *BDD*. Продукции 3 и 4 с помощью программного модуля *PRESIN* преобразуют логические уравнения в равноразмерные – имеется в виду то, что в каждом из уравнений используются не более *p* переменных. Поскольку продукции, связанные со стартом подчиненных стратегий являются безусловными, то каждая из них будет выполнена только по одному разу.

«*Получить SDF-листья в BDD*»

```
1. {SF_TYPE ="SDF"} → TIE_BDD
2. {SF_TYPE ="SDF"} → Выбор объекта
(анализ с начала)
```

Данная стратегия выполняет преобразование каждого *SDF*-листа в *LOG*-формат, представляющий *BDD*. Если же требуется представить все листья проекта в виде *BDD*, то непосредственно перед выполнением этой стратегии должна быть выполнена стратегия «*Представить все листья в SDF*». Результат применения данной стратегии к подсхеме *CIRC_B* представлен в листингах 3, 4.

Листинг 3. Описание блока *BH* в формате *LOG* (*BDD*-лист).

```
INP
x1 x2 x3 x4
OUT
h4 h5
FUNCTION
LOG
4  2  0
h4 = ^x1*^x3+x1*sf2;
h5 = x1*sf3;
sf2 = x2*x3;
sf3 = ^x2*x4+x2*x3;
END_LOG
```

Листинг 4. Описание блока ***BG*** в формате *LOG* (*BDD*-лист).

```
INP
h4 h5 x5 x6
OUT
f2
FUNCTION
LOG
4  1  0
f2 = ^x6*sf3+x6*sf2;
sf3 = x5*h4;
sf2 = ^h5*sf3+h5*sf4;
sf4 = ^x5*^h4;
END_LOG
```

Заметим, что в листингах 3 и 4 опущены списки внутренних переменных.

«*Получить все листья в RTL*»

```
1. {SF_TYPE ="SDF"} → Получить все ли-
стья в LOG
2. {SF_TYPE ="LOG"} & {~RTL} → MINRTL
3. {SF_TYPE ="LOG"} & {~RTL} → Выбор
объекта (анализ с начала)
```

Сначала все листья строятся в *LOG*-формате, затем программный модуль MINRTL выполняет преобразование формата *LOG* в булеву *RTL* форму. Применение данной стратегии к мини-

мизированному блоку (листу) *BH* позволяет получить *RTL*-лист, показанный в листинге 5.

Листинг 5. Описание блока ***BH*** в формате *LOG* (*RTL*-лист).

```
INP
x1 x2 x3 x4
OUT
h4 h5
INTER
zt_0 zt_4 zt_1 zt_2 zt_5 zt_3
END_PIN
FUNCTION
LOG
4  2  0
zt_0=^x1*^x3;
zt_4=x1*x2;
zt_1=zt_4*x3;
zt_2=zt_4*x3;
zt_5=x1*^x2;
zt_3=zt_5*x4;
h4=zt_1+zt_0;
h5=zt_3+zt_2;
END_LOG
```

«*Получить один лист BDD*»

```
1. → Получить один лист SDF
2 {SF_TYPE="SDF"} & {ROOT} → TIE_BDD
```

Цель данной стратегии – представление проекта в виде одного *BDD*-описания. Это пример стратегии, состоящей из двух продукций. Первая продукция не содержит условия в левой части, правая часть первой продукции является стратегией – это подчиненная стратегия. Вторая продукция позволяет запустить для выполнения модуль *TIE_BDD*.

«*Устранение иерархии*»

```
1. {SF_TYPE ="SDF"} → SDFLOG
2. {SF_TYPE ="SDF"} → Выбор объекта
(анализ с начала)
3. {ROOT} → Выбор объекта (анализ про-
должить)
4. {SF_TYPE="CONNECT_2"} | {SF_TYPE=
="CONNECT_N"} → UST_NEW
```

Цель данной стратегии – представление проекта в виде одного листа. В данной стратегии сначала идет подготовка к запуску модуля *UST_NEW*, для чего все листовые описания преобразуются в *LOG*-формат. В результирующем *LOG*-описании осуществляется редукция внутренних переменных, т.е. устраняются переобозначения одной и той же внутренней переменной. Результат стратегии – единственный *LOG*-лист проекта.

«*Получить один лист SDF*»

```
1. {SF_TYPE="SDF"} & {~sin1} → SDFLOG
2. {SF_TYPE="SDF"} & {~sin1} → Выбор
объекта (анализ с начала)
3. {ROOT} → Выбор объекта (анализ про-
должить)
4. {sin1=sin1} → sin1
5. {SF_TYPE="CONNECT_N"} | {SF_TYPE=
="CONNECT_2"} → UST_NEW
6. {SF_TYPE="LOG"} & {sin1} → ELIMINATE
```

В целом данная стратегия похожа на предыдущую, но результатом ее работы есть *SDF*-лист. В стратегии используется вспомогательный атрибут *sin*1, позволяющий управлять порядком выполнения продукций. В процессе выполнения данной стратегии первые две продукции преобразуют все *SDF*-листья в *LOG*-листья, затем находится головной (*ROOT*) блок *SF*-описания. После этого работает модуль устранения иерархии проекта и затем для скомпилированного в один лист проекта с помощью модуля *ELIMINATE* выполняется переход к *SDF*-описанию.

Для рассматриваемой схемы *CIRC* результатом является *SDF*-проект схемы (рис. 3), заданный в листинге 6.

Листинг 6. Описание всей схемы ***CIRC*** в формате *SDF* (*SDF*-лист).

```
INP
x1 x2 x3 x4 x5 x6
OUT
f1 f2 f3
FUNCTION
SDF
6  3  17
110-1- 100
011101 100
000101 100
001101 100
01011- 100
0101-1 100
0-0-11 010
10-101 010
111-10 010
0-0-10 010
10101- 001
10111- 001
1011-1 001
11011- 001
1101-1 001
010-1- 001
-1001- 101
END_SDF
```

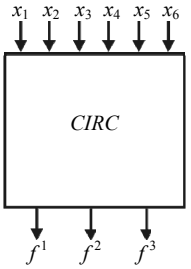

Рис. 3. Схема *CIRC* в виде одного блока

После устранения иерархии описания схемы *CIRC* оказывается, что ДНФ функций заданы на 17-ти элементарных конъюнкциях и функции не минимизированы. В практических ситуациях число конъюнкций может достигать сотен и тысяч. В таких случаях можно выполнить приведенную далее стратегию «*Конвейерная минимизация*».

«*Декомпозиция SDF-листьев*»

```
1. {SF_TYPE="SDF"}&{PORT_INP >SINQINP}&
&{TERM > SINQTERM} → RAZ1
2. {~ MIN} & {SF_TYPE ="SDF"} →
→ MINIMIZE
3. {~ MIN} & {SF_TYPE ="SDF"} → Выбор
объекта (анализ с начала)
4. {MIN} &{SF_TYPE ="SDF"} & {PORT_INP >
> SINQINP} → DECOMP
5. {MIN} & {SF_TYPE ="SDF"} &
& {PORT_INP > SINQINP}
      → Выбор объекта (анализ с начала)
```

Стратегия предназначена для декомпозиции «по входам» *SDF*-листьев. Результатом применения данной стратегии является проект, все листовые *SDF*-описания которого представляют собой системы функций, зависящие не более чем от $p = SINQINP$ переменных. Известно [9], что декомпозиция системы ДНФ по входным переменным методам «тождественных отображений» может приводить к возрастанию числа конъюнкций выходного блока построенной двухэлементной сети (в сравнении с числом конъюнкций декомпозируемого *SDF*-листа). Поэтому целесообразно в этой стратегии перед декомпозицией каждого *SDF*-листа выполнять «декомпозицию по термам» (модуль *RAZ*1) и минимизировать в классе ДНФ (модуль *MINIMIZE*) полученные блоки сети. Это позволяет надеяться на уменьшение числа перекодируемых конъюнкций и получать меньшее число промежуточных переменных при декомпозиции «по входам».

«*Минимизация глобальная*»

```
1. {SF_TYPE="CONNECT_N"} | {SF_TYPE=
="CONNECT_2"} → UST_NEW
2. {SF_TYPE="LOG"} → ELIMINATE
3. {SF_TYPE="SDF"} & {~MIN} → MINIMIZE
```

Целью стратегии является представление проекта в виде одного *минимизированного SDF*-листа. Поэтому сначала устраняется иерархия (модуль *UST_NEW*) , после чего осуществляется переход к *SDF* (модуль *ELIMINATE*) , и минимизация в классе ДНФ (модуль *MINIMIZE*.

«*Конвейерная минимизация*»

```
1. {SF_TYPE="SDF"} & {~MIN} & {ROOT} &
& {~sin1} → RAZ
2. {SF_TYPE="SDF"} & {~MIN} & {~sin1} →
→ MINIMIZE
3. {SF_TYPE="SDF"}&{~MIN}&{~sin1} → Вы-
бор объекта (анализ с начала)
4. {SF_TYPE="SDF"} & {MIN} & {~sin1} →
→ SDFLOG
5. {SF_TYPE="SDF"}&{MIN} & {~sin1} →
→ Выбор объекта (анализ с начала)
6. {sin1=sin1} → sin1
7. {ROOT} → Выбор объекта (анализ про-
должить)
8. {SF_TYPE="CONNECT_N"} | {SF_TYPE=
="CONNECT_2"} → UST_NEW
9. {SF_TYPE="LOG"} & {ROOT} & {sin1} →
→ ELIMINATE
10. {SF_TYPE="SDF"} & {~MIN} & {ROOT} &
& {sin1} → MINIMIZE
```

Выполнение первой продукции приводит к разбиению исходной системы на заданное число подсистем, после чего в продукциях 2, 3 осуществляется независимая минимизация каждой из подсистем в классе ДНФ. Затем осуществляется перенос активности на корневой блок (*ROOT*) в иерархии проекта. Выполнение восьмой продукции позволяет устранить иерархию описания (подсистемы объединяются в одну систему в *LOG*-формате), продукция девять преобразует *LOG*-формат в *SDF*-формат, выполнение десятой продукции ведет к дополнительной минимизации системы в целом, полученный результат минимизации и есть итог выполнения стратегии. Минимизированный блок, представляющий весь проект, получит значение атрибута $MIN = 1$, после чего все продукции описываемой стратегии становятся неприменимыми, и она заканчивает работу. В

листинге 7 представлен результат конвейерной минимизации для ДНФ функций, заданных в листинге 5.

Листинг 6. Минимизированная система ДНФ функций схемы *CIRC* в формате *SDF*.

```
INP
x1 x2 x3 x4 x5 x6
OUT
f1 f2 f3
FUNCTION
SDF
6  3  8
1101-1 001
1011-1 001
111-10 010
10-101 010
101-1- 001
0--101 100
0-0-1- 010
-10-1- 101
END_SDF
```

«*Конвейерная минимизация иерархическая*»

```
1. {SF_TYPE="SDF"} & {~MIN} → RAZ
2. → Минимизация SDF-листьев
3. {ROOT} → Выбор объекта (анализ продолжить)
4. → Минимизация глобальная
```

Выполнение данной стратегии состоит из трех важных этапов: разбиение на подсистемы, их отдельная минимизация, затем выполнение стратегии глобальной минимизации, в которой осуществляется объединение всех блоков проекта в один блок и его минимизация в классе ДНФ. Это пример иерархически устроенной стратегии, так как в нее входят две другие стратегии – «*Минимизация SDF-листьев*» и «*Минимизация глобальная*».

«*Эквивалентные преобразования*»

```
1. → Получить все листья в BDD
2. {SF_TYPE="LOG"}&{~work} & {~work2}&
& {PORT_INP>SINQINP} → PART_S1
3. {SF_TYPE ="LOG"} & {~work} &
&{~work2} & {PORT_INP > SINQINP}
      → Выбор объекта (анализ с начала)
4. {work = work} → work
5. {SF_TYPE="LOG"}&{~work2}&{~MIN}→Выбор объекта (анализ продолжать)
6. {SF_TYPE="LOG"} & {~work2} & {~MIN}→
→ ELIMINATE
7. {SF_TYPE="SDF"} & {~work2} & {~MIN} →
→ MINIMIZE
8. {SF_TYPE="SDF"}&{~work2}&{~MIN} →
→ Выбор объекта (анализ с начала)
9. {work2 = work2} → work2
10. {SF_TYPE="CONNECT_2"} & {work &
& work2} → UST_NEW
11. {SF_TYPE="CONNECT_2"}&{work&work2}→
      → Выбор объекта (анализ с начала)
```

Стратегия позволяет оптимизировать все листья проекта на основе диаграмм двоичного выбора (продукция 1), провести группирование уравнений (продукции 2, 3), преобразовать листовые описания в формат *SDF* (продукции 5, 6), выполнить для каждого листа совместную минимизацию (продукции 7, 8), постепенно (конвейером) устранить иерархию (продукции 10, 11). Продукции 4, 9 формируют значения управляющих переменных. В правой части первой продукции записано не имя исполняемого программного модуля, а исполняемая стратегия «*Получить все листья в BDD*». Этот пример показывает возможности иерархически организованной стратегии, обеспечивающей оптимизацию многоуровневых представлений систем булевых функций большой размерности со сменой иерархии описания проекта.

«*Укрупнить блоки по входам*»

```
1. {SF_TYPE ="SDF"} → SDFLOG
2. {SF_TYPE ="SDF"}  → Выбор объекта
(анализ с начала)
3. {SF_TYPE="CONNECT_2"} & {PORT_INP <=
<= SINQINP} → UST_NEW
4. {SF_TYPE="CONNECT_2"} & {PORT_INP <=
 <= SINQINP}
      → Выбор объекта (анализ с начала)
```

В стратегии находятся блоки типа *CONNECT_2*, у которых число входов *PORT_INP* удовлетворяет ограничению *PORT_INP* <= *SINQINP*, для данных блоков устраняется иерархия, полученные листья проекта представляются в *LOG*-формате. Данная стратегия может быть полезной при замене «мелкозернистых» проектов более крупным базисом, например, при реализации схемы в базисе ПЛМ.

«*Синтез ПЛМ по иерархии*»

```
1. → Декомпозиция SDF-листьев
2. → Декомпозиция по выходам и термам
```

В данной стратегии осуществляется синтез сети ПЛМ по иерархии: каждый из листовых *SDF*-блоков декомпозируется «по входам» (в первой продукции), затем (во второй продукции) полученные листовые *SDF*-блоки декомпозируются «по выходам и термам». Недос-

татком данной стратегии является то, что получаемые блоки, реализуемые на ПЛМ, не минимизируются «по промежуточным шинам», т.е. совместно не минимизируются в классе ДНФ.

«*Синтез ПЛМ с минимизацией*»

```
1. {SF_TYPE="SDF"} & {PORT_INP >
> SINQINP} & {~MIN} → DECOMP
2. {SF_TYPE="SDF"}&({PORT_OUT>SINQOUT}|
{TERM>SINQTERM)&{~MIN} → RAZ
3. {SF_TYPE ="SDF"} & {~MIN} → MINIMIZE
4. {SF_TYPE ="SDF"} & ({PORT_INP >
> SINQINP} | {PORT_OUT > SINQOUT} |
      {TERM > SINQTERM ) → Выбор объекта
(анализ с начала)
5. → Минимизация SDF-листьев
```

Синтез в базисе ПЛМ с ограниченными параметрами сводится к последовательному применению процедур «*декомпозиции по входам*» (модуль *DECOMP*), «*декомпозиции по выходам и термам*» (модуль *RAZ*) и совместной минимизации в классе ДНФ (модуль *MINIMIZE*) полученных блоков разложения, и тем самым устраняется недостаток предыдущей стратегии.

«*Синтез FPGA по иерархии*»

```
1. → Получить все листья в BDD
2. {SF_TYPE="LOG"}&({PORT_INP>SINQINP}
|{PORT_OUT>SINQOUT}) → PART_S1
3. {SF_TYPE="LOG"}&({PORT_INP>SINQINP}
|{PORT_OUT>SINQOUT})
      → Выбор объекта (анализ с начала)
4. {ROOT} → Выбор объекта (анализ про-
должать)
5. {SF_TYPE ="CONNECT_N"} → 2CONNECT
```

Сначала все листья представляются в виде *BDD*, затем укрупняются по параметру *p*, равному числу входов в настраиваемую (программируемую) ячейку *LUT* (*Look-Up Table* – таблица, реализующая логическую функцию), из которых состоит схема *FPGA* (*Field Programmable Gate Array* – программируемая пользователем вентильная матрица). В пятой продукции с помощью модуля *2CONNECT* осуществляется переход к структурному *SF*-описанию схемы.

«*Синтез FPGA по RTL*»

```
1. → Получить один лист RTL
2. {~ENLARG} → PRESIN
3. {ENLARG} & {SF_TYPE ="LOG"} →
→ PART_S1
```

Безусловно, выполняемая стратегия в первой продукции преобразует проект в *RTL*-лист, после чего происходит «укрупнение» уравнений по числу входных переменных и каждое уравнение помещается в один блок, реализуемый на одном *LUT*.

«*Синтез на одном ПЗУ*»

```
1. → Получить один лист SDF
2. {ROOT} & {~TAB} → SDFROM
```

В данной стратегии предполагается, что параметры пассивного запоминающего устройства (ПЗУ) соответствуют параметрам получающейся таблицы истинности, т.е. что таблица истинности помещается в соответствующее ПЗУ. Первая из продукций безусловно инициирует работу стратегии, обеспечивающей преобразование проекта в один *SDF*-лист. Вторая – строит таблицу истинности.

«*Синтез БМК по иерархии*»

```
1. → Получить все листья в SDF
2. → Минимизация SDF-листьев
3. {SF_TYPE ="SDF"} → COVE
4. {SF_TYPE ="SDF"} → Выбор объекта
(анализ с начала)
5. {ROOT} → Выбор объекта (анализ про-
должать)
6. {SF_TYPE ="CONNECT_N"} → 2CONNECT
```

Синтез осуществляется для листовых описаний, которые предварительно минимизируются в классе ДНФ. Реализованная стратегия заключается в поиске листовых описаний (типа *SDF*) и синтезе схемы в библиотеке проектирования базовых матричных кристаллов (БМК), т.е. в базисе библиотечных элементов для каждого *SDF*-листа. Размерность задачи синтеза сокращается, так как синтез каждого блока ведется независимо от других. При «ручном» проектировании проектировщику потребовалось бы поочередно перебрать все листовые блоки и выполнить процедуру (*COVER*) синтеза для каждого из них. Если листья проекта представляют собой небольшие по размерности блоки, например, полученные в результате синтеза *FPGA*, то данная стратегия будет неэффективной. Требуется сначала «укрупнить» листья проекта. Это и осуществляется в следующей стратегии.

«*Блочный синтез БМК*»

```
1. → Укрупнить блоки по входам
2. → Синтез БМК по иерархии
```

Если же параметр укрупнения больше либо равен числу входов всего проекта, то после выполнения первой продукции проект будет собран в один блок и будет реализоваться стратегия глобального синтеза схемы в библиотечном базисе.

Заключение. Все рассмотренные стратегии обработки *SF*-описаний проектов комбинационной логики могут быть интерпретированы как эквивалентные преобразования двухуровневых и многоуровневых представлений систем полностью определенных булевых функций. Система *FLC* открыта для пополнения, в настоящее время в нее включено более пятидесяти стратегий. *FLC* является эффективным средством для экспериментального сравнения различных маршрутов оптимизации и синтеза логических схем. Она состыкована по входным и выходным данным с промышленным синтезатором логических схем *LeonardoSpectrum*. Широкие экспериментальные исследования подтвердили высокую эффективность совместного использования *LeonardoSpectrum* и *FLC*.

1. *Бибило П.Н.* Системы проектирования интегральных схем на основе языка *VHDL*. StateCAD, ModelSim, LeonardoSpectrum. – М.: СОЛОН-Пресс, 2005. – 384 с.
2. *Бибило П.Н.*, *Романов В.И.* Использование продукций для автоматизированного управления логическим проектированием дискретных устройств // Изв. РАН. Теория и системы управления. – 2007. – № 1. – С. 55–67.
3. Леончик П.В. Минимизация систем булевых функций в классе дизъюнктивных нормальных форм // Информатика. – 2006. – № 1. – С. 88–96.
4. *Торопов Н.Р.* Минимизация систем булевых функций в классе ДНФ // Логическое проектирование. – 1999. – **4**. – С. 4–19.
5. *Бибило П.Н.*, *Леончик П.В.* Алгоритм построения диаграммы двоичного выбора для системы полностью определенных булевых функций. // УСиМ. – 2009. – № 5. – С. 42–49.
6. *Черемисинова Л.Д.* Комплекс программ синтеза логических схем на базе программируемых логических интегральных схем // Информатика. – 2006. – № 3(11). – С. 112–121.
7. *Бибило П.Н.*, *Романов В.И.* Новые эксперименты повторного синтеза комбинационных схем // Микроэлектроника. – 2008. – **37**. – № 3. – С. 228–240.
8. *Бибило П.Н.*, *Романов В.И.* Оптимизация многоуровневых представлений систем булевых функций при перепроектировании логических схем // УСиМ. – 2006. – № 5. – С. 20–29.
9. *Кардаш С.Н.* Декомпозиция системы булевых функций методом тождественных отображений // Проектирование систем логического управления. – Сб. науч. тр. / Ин-т техн. кибернетики АН БССР. – Минск, 1986. – С. 22–31.
10. *Закревский А.Д.* Логический синтез каскадных схем. – М.: Наука, 1981. – 416 с.
11. *Бибило П.Н.*, *Кириенко Н.А.* Оптимизационные преобразования логической схемы на основе блочного разбиения // Информатика. – 2009. – № 3. – С. 5–15.
12. *Кириенко Н.А.* Подсистема обработки структурно-функциональных описаний схем в системе автоматизированного проектирования // Проблемы разработки перспективных микро- наноэлектронных систем–2008 (МЭС–2008): Сб. тр. 3-й Всерос. науч.-техн. конф., Подмосковье, 06–10 окт. 2008 г. – М.: ИППМ РАН, 2008. – С. 215–220.

Поступила 22.12.2010

Тел. для справок: + 375 (17) 284-2084 (Минск)
E-mail: *bibilo@newman.bas-net.by*

●