УДК 519.688
DOI: 10.15827/0236-235X.125.026-033
Дата подачи статьи: 26.10.18
2019. Т. 32. № 1. С. 026–033

Интеграция САПР для синтеза логических схем с использованием глобальной оптимизации

П.Н. Бибило [1], *д.т.н., профессор, зав. лабораторией, bibilo@newman.bas-net.by*
В.И. Романов [1], *к.т.н., доцент, ведущий научный сотрудник, rom@newman.bas-net.by*

[1] *Объединенный институт проблем информатики Национальной академии наук Беларуси, г. Минск, 220012, Беларусь*

Предлагается технология проектирования цифровых устройств, позволяющая выполнять логическое моделирование VHDL-описаний комбинационной логики, формировать соответствующие системы булевых функций, проводить их логическую оптимизацию и синтезировать логические схемы в различных технологических библиотеках логических элементов. Интеграция программных средств в рамках этой технологии основывается на использовании скриптов и BAT-файлов, которые поддерживаются современными САПР.

Исходные VHDL-описания могут задавать как алгоритмические, так и функциональные описания – таблицы истинности систем полностью либо неполностью определенных булевых функций, системы дизъюнктивных нормальных форм, описания многоуровневых (скобочных) логических уравнений. Как исходные VHDL-описания могут использоваться также структурные описания логических схем, синтезированных в различных целевых технологических библиотеках, в этом случае осуществляется их перепроектирование в другой базис логических элементов.

Переход от VHDL-описаний к системам булевых функций происходит на основе логического моделирования на всех возможных наборах (полных тестах) значений входных переменных.

Для логической оптимизации используются мощные программы совместной и раздельной минимизации систем булевых функций в классе дизъюнктивных нормальных форм, а также программы минимизации многоуровневых BDD-представлений систем булевых функций на основе разложения Шеннона.

Для проведения проектирования достаточно указать исходное VHDL-описание, способ логической оптимизации и целевую библиотеку логических элементов, используемую в синтезаторе LeonardoSpectrum. На основании полученных данных автоматически формируется BAT-файл, осуществляющий синтез с использованием глобальной логической оптимизации. Пользователь может оценить найденное решение, сравнив его с другим, получаемым синтезатором LeonardoSpectrum по исходному описанию без выполнения предварительной оптимизации.

Ключевые слова: *VHDL, логическое моделирование, синтез комбинационных логических схем, логическая оптимизация, разложение Шеннона.*

Современные *системы автоматизированного проектирования* (САПР) цифровых устройств на базе заказных *сверхбольших интегральных схем* (СБИС) [1] решают различные задачи на последовательно выполняемых этапах проектирования, начиная от алгоритмического и логического этапов и заканчивая этапом топологического и физического проектирования. На этапе алгоритмического проектирования важной является задача верификации исходных описаний проектов [2, 3]. Исходные описания проектов схем задаются в виде исходных спецификаций на языках VHDL и Verilog [4]. Решающую роль играют начальные этапы, от эффективности выполнения которых зависят сложность (площадь схемы, число транзисторов), быстродействие и энергопотребление логических схем. Именно два последних параметра приобретают все большее значение при проектировании [5]. Получение логической схемы (синтез) по исходному алгоритмическому либо функциональному описанию осуществляется системами синтеза – синтезаторами. Синтезаторы логических схем заменяют каждую конструкцию языка VHDL (либо Verilog) соответствующим функционально-структурным описанием [6], включающим логические функции и элементы памяти, после чего дальнейшей оптимизации подвергается комбинационная логика, представленная взаимосвязанными логическими выражениями. Такие выражения задают многоуровневые представления систем булевых функций, описывающих функциональные блоки, входящие в состав проекта цифровой схемы, синтезируемой в том или ином базисе логических элементов ASIC (application-specific integrated circuits – заказная СБИС) либо FPGA (field-programmable gate array). Изменение способов реализации логических элементов на транзисторном уровне [7] для субмикронных норм производства кристаллов влечет увеличение размерностей задач синтеза логических схем и требует совершенствования соответствующих алгоритмов и программных средств логической оптимизации. Используемая в промышленных синтезаторах логических схем оптимизация является по сути локальной, то есть оптимизации подвергаются части схемы – кластеры, выделяемые из оптимизируемого функционального описания проекта схемы. Глобальная оптимизация для достаточно больших проектов не выполняется, так как размерности оптимизационных задач огромны и достигают сотен входных и выходных переменных и сотен тысяч промежуточных логических переменных.

Правильное управление синтезатором позволяет получать лучшую реализацию VHDL-описаний проек-

тов. Как показывает опыт [8], использование соответствующих скриптов в синтезаторе Synopsys Design Compiler позволяет решать базовые задачи синтеза самосинхронных схем, хотя данный синтезатор ориентирован на реализацию синхронных схем. Как показано в [6], с помощью синтезатора LeonardoSpectrum можно даже решать задачи формальной верификации VHDL-описаний проектов цифровых схем.

От эффективности выполнения этапа синтеза логических схем в промышленных синтезаторах зависят основные характеристики реализуемого VHDL-проекта – быстродействие, площадь кристалла, энергопотребление. На практике в проектных организациях используются устоявшиеся маршруты проектирования, например, с использованием программных пакетов компании Mentor Graphics [9]. Использование программных пакетов Mentor Graphics для решения проектных задач на различных этапах проектирования СБИС представлено в [10].

В данной работе предлагается подход к созданию нового маршрута синтеза функциональных блоков ASIC, реализующих комбинационную логику, на основе объединения (интеграции) функциональных возможностей программных пакетов Mentor Graphics – системы моделирования ModelSim либо Questa, системы синтеза LeonardoSpectrum и свободно распространяемых (либо имеющихся в отечественных САПР) программ логической оптимизации. Реализация данного подхода разработчиками программных средств САПР не является трудоемкой, так как средства управления (скрипты) системами моделирования и системой синтеза представлены в данной работе, а программы конвертации, необходимые для преобразования проектных данных (систем булевых функций) в требуемые форматы, являются несложными. Эксперименты выявили области предпочтительного использования данного маршрута по сравнению с традиционным маршрутом синтеза, что может быть полезным для проектировщиков цифровых заказных СБИС.

Предлагаемый подход

Эксперименты показали, что для функциональных описаний комбинационных схем предварительная глобальная оптимизация, выполняемая с помощью программ совместной минимизации систем булевых функций в классе *дизъюнктивных нормальных форм* (ДНФ) либо с помощью программ минимизации многоуровневых BDD-представлений (BDD – Binary Decision Diagram) на основе разложения Шеннона, может давать значительные выигрыши по площади схем и быстродействию по сравнению с результатами синтеза от исходных неоптимизированных функциональных описаний [11, 12]. Однако для применения программ глобальной оптимизации требуется переход от алгоритмических представлений функций комбинационных блоков ASIC к представлению функций в виде таблицы истинности либо системы ДНФ, которая может быть подвергнута оптимизации с помощью мощных компьютерных программ логической оптимизации. Если же функции реализуемой системы являются неполностью определенными (частичными), то переход к двухуровневому (табличному) представлению позволяет проводить логическую минимизацию с учетом возможности доопределения частичных функций до полностью определенных булевых функций, что позволяет улучшать решения оптимизационных задач. Однако получение таких форм представления систем булевых функций не предусмотрено в синтезаторах, в частности, в синтезаторе LeonardoSpectrum, который ориентирован на синтез схем не только FPGA, но и ASIC. Имеющаяся в синтезаторе логических схем LeonardoSpectrum [6] управляющая опция установления размера кластера при логической оптимизации не позволяет получать представление проекта функционального блока в виде системы булевых функций, заданной в матричной форме (таблицы истинности, системы ДНФ), часто называемой в зарубежной литературе двухуровневым и/или представлением комбинационной логики.

Таким образом, предлагаемый подход к созданию маршрута синтеза логических схем основывается на выявленных ограничениях синтезатора LeonardoSpectrum и результатах широких экспериментальных исследований эффективности программ логической оптимизации при синтезе комбинационных схем. Естественно, этот подход имеет и свои ограничения, связанные с числом входных переменных функционального блока: число таких переменных не должно превышать 20.

Предлагаемый маршрут синтеза функциональных комбинационных блоков ASIC включает три этапа (рис. 1–3) и выполняется в пакетном режиме (выполняется BAT-файл), управляя вызываемыми подсистемами и программами с помощью соответствующих скриптов. Примеры скриптов будут приведены далее.

Этап 1. Получение системы булевых функций по алгоритмическому VHDL-описанию комбинационного блока на основе моделирования в системах ModelSim, Questa (Mentor Graphics).

Этап 2. Глобальная логическая оптимизация, выполняемая с помощью программ, имеющихся в системе CMOSLD [13] либо являющихся свободно распространяемыми.

Программа ESPRESSO [14] позволяет выполнять совместную и раздельную минимизацию систем булевых функций в классе ДНФ по различным критериям (число элементарных конъюнкций, число литералов) точным либо приближенным алгоритмом. Формат входных и выходных данных описан в [15]. Данная программа имеет мировую известность в среде разработчиков программных средств САПР цифровых схем и является свободно распространяемой [16].

Программа Tie_BDD [11] предназначена для минимизации многоуровневых BDD-представлений систем булевых функций, входными данными являются мат-

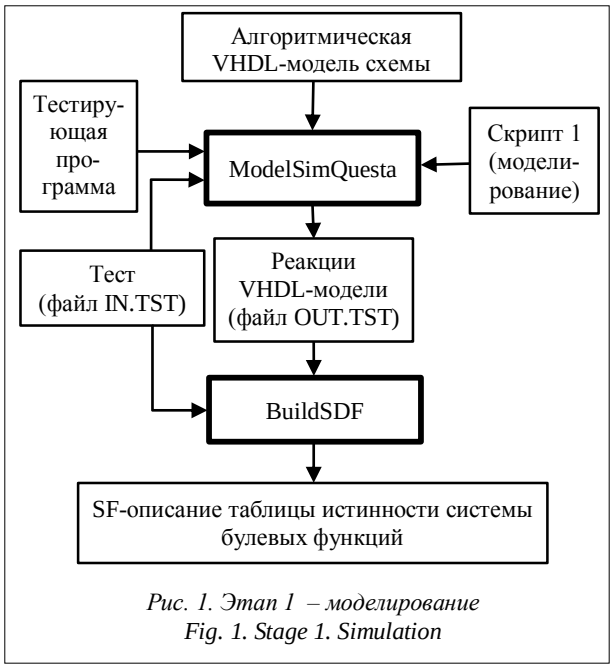

Рис. 1. Этап 1 – моделирование
Fig. 1. Stage 1. Simulation

ричные формы систем полностью определенных либо частичных булевых функций на языке SF – внутреннем языке системы CMOSLD [13] и системы FLC [11] логической оптимизации, выходными данными – логические уравнения на языке SF, задающие формулы разложений Шеннона, соответствующие оптимизированным BDD-представлениям систем булевых функций. Разложением Шеннона полностью определенной булевой функции $f(x_1, \ldots, x_n)$ по переменной x_i называется представление

$$f(x_1, \ldots, x_n) = \overline{x}_i\, f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) \vee \vee\, x_i f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n). \quad (1)$$

Рис. 2. Этап 2 – логическая оптимизация
Fig. 2. Stage 2. Logical optimization

Подфункции $f_0 = f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n)$ и $f_1 = f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$ в правой части (1) часто называются коэффициентами разложения по переменной x_i либо остаточными подфункциями. Они получаются из функции $f(x_1, \ldots, x_n)$ подстановкой вместо переменной x_i константы 0 и 1 соответственно. Каждая из подфункций f_0 и f_1 может быть разложена по одной из переменных из множества $\{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n\}$.

Для системы булевых функций разложение Шеннона каждой из функций ведется по одной и той же последовательности переменных разложения. Процесс разложения подфункций заканчивается, когда все n переменных будут использованы для разложения либо все подфункции выродятся до констант 0, 1. На каждом шаге разложения выполняется сравнение на равенство полученных подфункций и оставляется одна из нескольких попарно равных. Полученное многоуровневое представление системы булевых функций, задаваемое в виде графа, называется в [17, 18] BDD-представлением.

Программа BDD_Builder [12] реализует алгоритм минимизации многоуровневых BDD-представлений с нахождением на каждом шаге разложения взаимно инверсных подфункций. Форматы входных и выходных данных такие же, как у программы Tie_BDD.

Этап 3. Логический синтез по оптимизированному VHDL-описанию системы булевых функций, выполняемый в синтезаторе LeonardoSpectrum (Mentor Graphics).

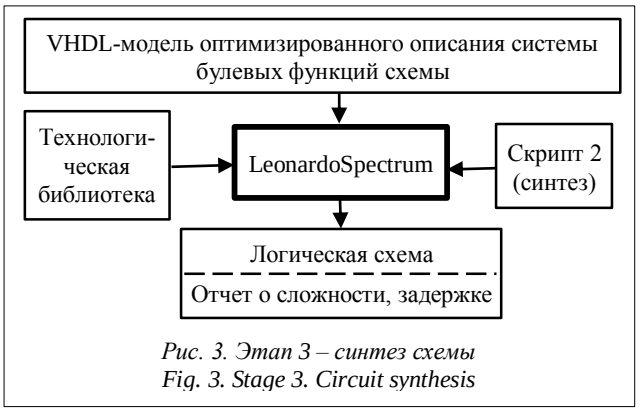

Рис. 3. Этап 3 – синтез схемы
Fig. 3. Stage 3. Circuit synthesis

Структура BAT-файла и выполнение проектных процедур

Этап 1.

1. По исходному VHDL-описанию комбинационного блока определяется число n входных полюсов данного блока.

2. Генерируется таблица (текстовый файл IN.TST) всех 2^n элементов булева пространства, построенного над переменными булева вектора входных сигналов.

3. Генерируется тестирующая программа для выполнения моделирования исходного алгоритмического описания на всех 2^n наборах значений входных сигналов.

4. Генерируется скрипт 1 для системы VHDL-моделирования. Пример скрипта 1:

```
# имя тестирующей VHDL-программы
Set PRJ tstb
# создание библиотеки
vlib ./vsim
# имя библиотеки
vmap work ./vsim
# компиляция файлов
vcom -f //files.f -source
# вызов моделирующей программы (моделирование без оптимизации)
vsim  -novopt work.${PRJ} +no_glitch_msg
# выполнение полного сеанса моделирования
run -a
# выход из системы моделирования
quit -f
```

При выполнении скрипта 1 моделируется исходное VHDL-описание комбинационного блока (список входящих в блок компонентов представлен в файле FILES.F) и строится текстовый файл OUT.TST, содержащий реакции комбинационного блока для каждого набора используемого при моделировании теста.

5. Вызывается программа BuildSDF для формирования (по файлу IN.TST входных наборов и файлу OUT.TST реакций) SF-описания функций комбинационного блока в виде таблицы истинности.

Этап 2.

6. Вызывается программа (ESPRESSO, BDD_Builder и др.) для выполнения глобальной логической оптимизации SF-описания (таблицы истинности), в результате получается минимизированное SF-описание системы булевых функций.

7. Вызывается программа конвертации минимизированного SF-описания в VHDL-описание.

8. Генерируется скрипт 2 для управления процессом синтеза. Пример скрипта 2 (жирным шрифтом выделены настраиваемые параметры – путь доступа и имя s3lib технологической библиотеки логических элементов) для синтезатора LeonardoSpectrum, где указываются опции синтеза и требуемая технологическая библиотека логических элементов (FPGA, ASIC):

```
# инициализация синтезатора
clean_all;
# Путь доступа к библиотеке логических элементов
set LIBpath z:/_Mon/libs/s3lib.syn;
# Имя библиотеки логических элементов, например
set LIBname s3lib;
# способ реализации арифметических операторов
set modgen_select Smallest;
# размер кластера для локальной оптимизации
set asic_auto_dissolve_limit 500;
set auto_dissolve_limit 500;
# чтение алгоритмического VHDL-описания example_1
read //example_1;
# чтение целевой библиотеки s3lib.syn
load_library LIBpath;
# установка опции синтеза по обработке иерархических описаний
set –hierarchy flatten
# установка режима синтеза
set effort standard
# выполнение синтеза с указанием установленных опций синтеза
optimize -target LIBname -macro -area -effort standard -hierarchy flatten
# выдача сообщения о числе и составе элементов синтезированной схемы
report_area -cell_usage
# выдача сообщения о величине задержки схемы
report_delay -num_paths 1 -critical_paths
# запись синтезированной схемы в формате структурного VHDL-описания
auto_write //result.vhd
```

Этап 3.

9. Вызывается синтезатор LeonardoSpectrum, который выполняет синтез по оптимизированному VHDL-описанию системы булевых функций под управлением скрипта 2.

Построение эквивалентных описаний для любого алгоритмического VHDL-описания комбинационной логики, оптимизация и последующий синтез по оптимизированным описаниям осуществляются с помощью разработанной программы, которая автоматически генерирует соответствующий BAT-файл. Пользователю нужно лишь указать исходное алгоритмическое VHDL-описание, способ оптимизации и технологическую библиотеку логических элементов, используемую в синтезаторе LeonardoSpectrum. В результате выполнения BAT-файла LeonardoSpectrum может выполнить синтез с использованием глобальной оптимизации, после чего можно сравнить решение (сложность, быстродействие, энергопотребление) полученной схемы с решением (схемой), полученным без использования BAT-файла. Эта схема может быть построена по исходному алгоритмическому VHDL-описанию с оптимизацией, имеющейся в синтезаторе. Предлагаемый подход к синтезу эффективен для цепочек (конвейеров) арифметических операций с ограниченным числом входных полюсов.

В качестве программ логической оптимизации могут быть программы, выполняющие оптимизацию различных классов систем булевых функций как полностью, так и неполностью определенных функций. При этом есть следующие ограничения.

1. Исходное VHDL-описание комбинационного блока должно содержать информационные входные и выходные порты только типов std_logic, std_logic_vector [4, 6]. Такой тип данных является стандартным и наиболее широко используется в VHDL-описаниях проектов цифровых схем.

2. Общее число n входных полюсов комбинационного блока не должно превышать 20. Это ограничение достаточно очевидно, так как число 1 048 576 всевоз-

можных двоичных наборов (число строк в таблице истинности) равно 2^n для системы булевых функций, зависящих от n переменных, и при $n = 20$ превышает миллион наборов.

Примеры

Пример 1. Пусть требуется синтезировать цифровое устройство example_1, выполняющее последовательность (цепочку) $y = (a + b)(a – b)$ арифметических операций, причем операнды (целые числа) a и b принимают значения в диапазоне $–6 < a < 5$; $–6 < b < 5$.

На языке VHDL каждый из операндов a и b представляется четырьмя битами, результат y операции представляется шестью битами. Естественно, что не для всех значений входных двоичных векторов нужно вычислять значение выходного вектора, это нужно делать только для чисел, попадающих в заданный диапазон. Алгоритмическое VHDL-описание устройства example_1 представляется следующим образом:

```
Library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity example_1 is
port (
a : in  std_logic_vector(3 downto 0);
b : in  std_logic_vector(3 downto 0);
y : out std_logic_vector(5 downto 0));
end;

architecture beh of example_1 is
signala_int, b_int : integer range -15 to
15;
signal e: integer range -144 to 144;
begin
a_int<= to_integer(signed(a));
b_int<= to_integer(signed(b));
 p0: process (a_int, b_int)
variable e: integer range -256 to 256;
begin
if  ((a_int<= -6) or (a_int>= 5) or (b_int<=
-6) or (b_int>= 5))
then y <= "------";
else
e   := ((a_int + b_int) * (a_int - b_int) );
y <= std_logic_vector(to_signed(e,6));
end if;
end process;
endbeh;
```

Результат синтеза по исходному VHDL-описанию example_1: площадь схемы (суммарное число транзисторов в элементах) – 720 транзисторов, задержка схемы – 6.87 наносекунды (нс). Проектирование осуществлялось в целевой технологической библиотеке [13] проектирования отечественных заказных КМОП-схем.

Сформируем и выполним BAT-файл, при этом для глобальной логической оптимизации полученной системы неполностью определенных функций (табл. 1) будет использоваться программа BDD_Builder [12]. В первой колонке этой таблицы представлено содержимое файлов IN.TST и OUT.TST, хранящих входные наборы и реакции VHDL-модели соответственно. Система неполностью определенных булевых функций, полученная в результате моделирования (этап 1) на всевозможных входных двоичных векторах, дана в таблице 1, отрицательные числа при синтезе задаются в дополнительном коде.

Таблица 1

Таблица истинности системы булевых функций, задающей функционирование цифрового устройства example_1

Table 1

The truth table of the system of Boolean functions that define digital device example_1

Система булевых функций			Арифметические операции $y = (a + b)(a– b)$
a	***b***	***y***	
0000	0000	000000	(0+0)*(0-0)=0
0000	0001	111111	(0+1)*(0-1)=-1
0000	0010	111100	(0+2)*(0-2)=-4
0000	0011	110111	(0+3)*(0-3)=-9
0000	0100	110000	(0+4)*(0-4)=-16
0000	0101	------	(0+5)*(0-5) не определено
0000	0110	------	(0+6)*(0-6) не определено
0000	0111	------	(0+7)*(0-7) не определено
0000	1000	------	(0+8)*(0-8) не определено
0000	1001	------	(0+(-7))*(0-(-7)) не определено
0000	1010	------	(0+(-6))*(0-(-6)) не определено
0000	1011	100111	(0+(-5))*(0-(-5))= -25
0000	1100	110000	(0+(-4))*(0-(-4))= -16
……			……
1111	1001	------	((-1) +(-7)) * ((-1)-(-7)) не определено
1111	1010	------	((-1) +(-6)) * ((-1)-(-6)) не определено
1111	1011	101000	((-1) +(-5)) * ((-1)-(-5)) =-24
1111	1100	110001	((-1) +(-4)) * (((-1)-(-4)) =-15
1111	1101	111000	((-1) +(-3)) * ((-1)-(-3)) =-8
1111	1110	111101	((-1) +(-2)) * ((-1)-(-2)) =-3
1111	1111	000000	((-1) +(-1)) * ((-1)-(-1)) =0

Результат синтеза в LeonardoSpectrum по оптимизированному представлению: сложность схемы – 474 транзистора, задержка – 2.34 нс.

Пример 2. При выполнении цепочки операций $y = (a + b)(ab)$ будем задавать операнды (целые числа) a и b пятиразрядными двоичными векторами, интерпретируемыми как целые положительные числа в диапазоне $8 < a < 16$, $8 < b < 16$. VHDL-код, описывающий поведение устройства example_2, выглядит следующим образом:

```
Library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity example_2 is
port (
a : in  std_logic_vector(4 downto 0);
b : in  std_logic_vector(4 downto 0);
y : out std_logic_vector(12 downto 0));
end;
```

```
architecture beh of example_2 is
signala_int, b_int : integer range 0 to 31;

begin
a_int<= to_integer(unsigned(a));
b_int<= to_integer(unsigned(b));

 p0: process (a_int, b_int)
variable e: integer range 0 to 8191;
begin
if ((a_int<=8) or (a_int>=16) or (b_int<=8)
or (b_int>=16))
then y <= "-------------";
else
e   := ((a_int + b_int) * (a_int * b_int) );
y <= std_logic_vector(to_unsigned(e,13));
end if;
end process;
end beh;
```

Результат синтеза по VHDL-описанию example_2: площадь схемы (суммарное число транзисторов в элементах) – 3 078 транзисторов, задержка схемы – 14.50 нс. Результат синтеза в LeonardoSpectrum по оптимизированному представлению: сложность схемы – 1 075 транзисторов, задержка – 2.96 нс.

Пример 3. Пусть требуется синтезировать цифровое устройство, выполняющее цепочку $y = (a + b)\times$ $\times(a - b)(a - 1)(b - 1)(a - 2)(b - 2)$ арифметических операций, причем операнды (целые положительные либо отрицательные числа) a и b представляются 5-разрядными двоичными векторами (тип signed пакета numeric_std [6]) для всего диапазона их возможных значений. Число входных портов устройства – 10, число выходных портов устройства – 25 старших разрядов двоичного представления числа y.

Результаты синтеза по исходному VHDL-описанию: площадь – 24 482 транзистора, задержка схемы – 32.40 нс. Результат синтеза по оптимизированному представлению: площадь – 19 131 транзистор, задержка – 5.72 нс.

Эксперимент 1

По аналогии с примером 2 были подготовлены VHDL-описания для вычисления арифметического выражения $y = (a + b)(ab)$, причем целые положительные числа a и b были представлены двоичными векторами с числом разрядов k = 6, 7, 8, 9. Как и в примере 2, каждое из этих чисел было ограничено соответствующим диапазоном, составляющим одну четверть возможных значений. Результаты эксперимента представлены в таблице 2, где S_{ASIC} – площадь схемы, выражаемая в суммарном числе транзисторов, входящих в логические элементы схемы; τ – задержка схемы (нс). Проектирование осуществлялось в той же библиотеке логических элементов, что и в примерах 1 и 2. Результаты всех экспериментов были верифицированы с помощью системы FormalPro [19] формальной верификации, которая смогла провести верификацию исходного проекта с частично определенным поведением (значения «–» в VHDL интерпретируются как значение don't care перечислимого типа std_logic) с VHDL-описанием логической схемы, реализующей систему полностью определенных функций, что является несомненным достоинством данной системы верификации.

Эксперимент 1 показывает, что предложенный подход позволяет получать для рассмотренных примеров алгоритмических VHDL-описаний значительный выигрыш по быстродействию схем, выигрыш по площади достигается только при разрядности чисел n = 5, 6. Применение предложенного подхода может быть эффективно, например, при реализации схем модулярной арифметики [20–22], когда значительное повышение скорости вычисления арифметических операций достигается за счет разбиения позиционного представления входных операндов на операнды меньшей разрядности с целью их обработки независимо друг от друга.

Эксперимент 2

В данном эксперименте были реализованы на FPGAVirtex-6 [23] те же VHDL-описания, что и в эксперименте 1. Результаты эксперимента представлены в таблице 3.

Эксперимент показал, что эффективными средствами выполнения цепочек арифметических операций являются схемы DSP (Digital Signal Processor), предназначенные для эффективной обработки цифровых сигналов. Замены арифметических операторов в VHDL-коде оптимизированными представлениями систем булевых функций нецелесообразны при использовании FPGA Virtex-6 для данного примера с

Таблица 2

Результаты эксперимента 1

Table 2

Results of the experiment 1

Число *(n)* входов схемы	**Разрядность (*k*) чисел *a, b***	**Диапазон значений *a, b***	**Синтез по исходной алгоритмической VHDL-модели**		**Предлагаемый подход**	
			S_{ASIC}	τ	S_{ASIC}	τ
10	5	$8 < a, b < 16$	3 078	14.50	***1 075***	***2.96***
12	6	$16 < a, b < 32$	4 534	17.31	***4 183***	***4.44***
14	7	$32 < a, b < 64$	6 338	20.60	16 007	***4.79***
16	8	$64 < a, b < 128$	8 328	23.54	57 779	***6.38***
18	9	$128 < a, b < 256$	10 674	26.63	193 503	***7.83***

Таблица 3

Результаты эксперимента 2

Table 3

Results of the experiment 2

Число *(n)* входов схемы	**Разрядность (*k*) чисел *a, b***	**Диапазон значений *a, b***	**Синтез по исходной алгоритмической VHDL-модели с использованием DSP**		**Синтез по исходной алгоритмической VHDL-модели без DSP**	**Предлагаемый подход**
			Число LUT	**Число блоков DSP48E1**	**Число LUT**	**Число LUT**
10	5	$8 < a, b < 16$	42	1	98	***22***
12	6	$16 < a, b < 32$	-	2	***130***	183
14	7	$32 < a, b < 64$	-	2	***180***	718
16	8	$64 < a, b < 128$	10	2	***231***	2915
18	9	$128 < a, b < 256$	11	2	***295***	14201

числом входов, большим 10, однако для более сложных цепочек арифметических операций ситуация может быть другой (см. пример 3).

Предложенный маршрут синтеза функциональных блоков заказных СБИС чаще всего предпочтителен при схемной реализации цепочек последовательно выполняемых арифметических операций для целочисленных операндов небольшой разрядности, принимающих значения в ограниченных диапазонах, что приводит к логической оптимизации систем неполностью определенных булевых функций.

Заключение

Замена (свертка) цепочек последовательно выполняемых арифметических операций соответствующими матричными моделями полностью либо частично определенных булевых функций и оптимизация этих моделей целесообразны при ограниченных размерностях задач синтеза, так как позволяет использовать программы глобальной логической оптимизации для систем булевых функций, являющихся математическими моделями поведения блоков комбинационной логики. В отличие от реализованного в промышленном синтезаторе LeonardoSpectrum (и других промышленных синтезаторах) подхода к синтезу, использующего локальную оптимизацию (что совершенно необходимо при решении задач синтеза большой размерности), предложенный в данной работе маршрут синтеза ориентирован на использование программ глобальной логической оптимизации систем булевых функций и может приводить к лучшим результатам синтеза для комбинационных функциональных блоков заказных СБИС с ограниченным числом входных переменных. На практике можно выполнить синтез схемы по различным маршрутам и выбрать маршрут, позволяющий получить схему с лучшими характеристиками по площади, быстродействию и энергопотреблению.

Литература

1. Рабаи Ж.М., Чандракасан А., Николич Б. Цифровые интегральные схемы; [пер. с англ.]. М.: Вильямс, 2007. 912 с.
2. Слинкин Д.И. Анализ современных методов тестирования и верификации проектов сверхбольших интегральных схем // Программные продукты и системы. 2017. Т. 30. № 3. С. 401–408.
3. Чэнь М., Цинь К., Ку Х.-М., Мишра П. Валидация на системном уровне. Высокоуровневое моделирование и управление тестированием. М.: Техносфера, 2014. 296 с.
4. Поляков А.К. Языки VHDL и VERILOG в проектировании цифровой аппаратуры. М.: СОЛОН-Пресс, 2003. 320 с.
5. Белоус А.И., Солодуха В.А., Шведов С.В. Космическая электроника. М.: Техносфера, 2015. Кн. 1. 696 с.
6. Бибило П.Н. Системы проектирования интегральных схем на основе языка VHDL. StateCAD, ModelSim, LeonardoSpectrum. М.: СОЛОН-Пресс, 2005. 384 с.
7. Amaru L.G. New data structures and algorithms for logic synthesis and verification. Springer, 2017, 156 p.
8. Сурков А.В. Использование Synopsys Design Compiler для синтеза самосинхронных схем // Программные продукты и системы. 2014. № 4. С. 24–30.
9. Турцевич А.С., Белоус А.И., Шведов С.В., Кутас М.А. Развитие САПР микроэлектроники. САПР ОАО «ИНТЕГРАЛ» // Современные информационные и электронные технологии: сб. тр. Междунар. научн.-практич. конф. Одесса, 2013. С. 17–20.
10. Быковский С.В., Горбачев Я.Г., Ключев А.О., Пенской А.В., Платунов А.Е. Сопряженное проектирование встраиваемых систем (hardware/software co-design). СПб, 2016. 106 с.
11. Бибило П.Н., Романов В.И. Логическое проектирование дискретных устройств с использованием продукционно-фреймовой модели представления знаний. Минск: Беларус. навука, 2011. 279 с.
12. Бибило П.Н., Ланкевич Ю.Ю. Использование полиномов Жегалкина при минимизации многоуровневых представлений систем булевых функций на основе разложения Шеннона // Программная инженерия. 2017. № 3. С. 369–384.
13. Бибило П.Н., Авдеев Н.А., Кардаш С.Н., Кириенко Н.А., Ланкевич Ю.Ю., Логинова И.П., Романов В.И., Черемисинов Д.И., Черемисинова Л.Д. Система логического проектирования функциональных блоков заказных КМОП СБИС с пониженным энергопотреблением // Микроэлектроника. 2018. Т. 46. № 1. С. 72–88.
14. Brayton K.R., Hactel G.D., McMullen C.T., Sangiovanni-Vincentelli A.L. Logic minimization algorithm for VLSI synthesis. Kluwer Acad. Publ., 1984, 193 p.
15. Espresso(5OCTTOOLS) URL:http://www.ecs.umass.edu/ece/labs/vlsicad/ece667/links/espresso.5.html (дата обращения: 23.10.2018).
16. Espresso(1OCTTOOLS) URL: http://www.ecs.umass.edu/ece/labs/vlsicad/ece667/links/espresso.1.html (дата обращения: 23.10.2018).
17. Akers S.B. Binary Decision Diagrams. IEEE Trans. on Computers. 1978, vol. C-27, no. 6, pp. 509–516.
18. Bryant R.E., Meinel C. Ordered Binary Decision Diagrams. Logic synthesis and verification. Kluwer Acad. Publ., 2002, pp. 285–307.
19. Лохов А. Функциональная верификация СБИС в свете решений Mentor Graphics // Электроника: наука, технология, бизнес. 2004. № 1. С. 58–62.
20. Червяков Н.И., Сахнюк П.А., Шапошников А.В., Рядов С.А. Модулярные параллельные вычислительные структуры нейропроцессорных систем. М.: Физматлит, 2003. 288 с.
21. Стемпковский А.Л., Корнилов А.И., Семенов М.Ю. Особенности реализации устройств с цифровой обработкой сигналов в интегральном исполнении с применением модулярной арифметики // Информационные технологии. 2004. № 2. С. 2–9.
22. Gorodecky D., Villa T. Efficient hardware realization of arith-

metic operations for the residue number system by Boolean minimization. Proc. 13th Intern. Workshop on Boolean Problems, Bremen, Germany, 2018, pp. 195–203.

23. Зотов В. Особенности архитектуры нового поколения высокопроизводительных ПЛИС FPGA фирмы Xilinx серии Virtex-6 // Компоненты и технологии. 2009. № 8. С. 78–85.

Software & Systems
DOI: 10.15827/0236-235X.125.026-033

Received 26.10.18
2019, vol. 32, no. 1, pp. 026–033

CAD integration for logic synthesis using global optimization

P.N. Bibilo [1], *Dr.Sc. (Engineering), Professor, Head of Laboratory, bibilo@newman.bas-net.by*
V.I. Romanov [1], *Ph.D. (Engineering), Associate Professor, Senior Researcher, rom@newman.bas-net.by*

[1] *United Institute of Informatics Problems of the National Academy of Sciences of Belarus (UIIP NASB), Minsk, 220012, Belarus*

Abstract. The paper proposes a technology for designing digital devices. This technology allows logical modeling of VHDL descriptions of combinational logic, forming the corresponding systems of Boolean functions, their logical optimizing and synthesizing logic circuits in various technological libraries of logic elements. The software integration within this technology is based using scripts and BAT files that are supported by modern CAD systems.

The source VHDL descriptions can set algorithmic and functional descriptions. They are truth tables of completely or noncompletely specified Boolean function systems, systems of partial Boolean functions, systems of disjunctive normal forms, descriptions of multilevel logical equations. In addition, structural descriptions of logic circuits synthesized in various target technological libraries might also be used as source VHDL descriptions. In this case, they are redesigned into another basis of logical elements.

The transition from VHDL descriptions to systems of Boolean functions is based on logical simulation for all possible sets of input variables.

Logical optimization includes using of powerful programs of joint and separate minimization of Boolean function systems in the class of disjunctive normal forms, as well as programs of minimization of multilevel BDD representations (BDD – Binary Decision Diagram) of Boolean function systems based on Shannon's expansion.

A user only needs to specify a VHDL source description, a logical optimization method and a target library of logic elements used in the LeonardoSpectrum synthesizer. The required BAT file is generated automatically. The file provides synthesis using global logic optimization. The user can assess the solution found by comparing with another one that the LeonardoSpectrum synthesizer obtained from the original description without prior optimization.

Keywords: VHDL, logical simulation, synthesis of combinational logic circuits, logical optimization, Shannon's expansion.

References

1. Rabaey J.M., Chandrakasan A., Nikolic B. *Digital Integrated Circuits*. 2nd ed., Pearson Publ., 2003, 800 p. (Russ. ed.: Moscow, Vilyams Publ., 2007, 912 p.).
2. Slinkin D.I. Analysis of modern methods of testing and verification of projects of very-large-scale integrated circuits. *Software & Systems*. 2017, no. 3, pp. 401–408.
3. Chen M, Tsin K., Ku Kh.-M., Mishra P. *Validation at the System Level. High-Level Simulation and Testing Management*. Moscow, Tekhnosfera Publ., 2014, 296 p.
4. Polyakov A.K. *The VHDL and VERILOG Languages in Digital Equipment Design*. Moscow, SOLON Press, 2003, 320 p.
5. Belous A.I., Solodukha V.A., Shvedov S.V. *Space Electronics*. Vol. 1. Moscow, Tekhnosfera Publ., 2015, 696 p.
6. Bibilo P.N. *Computer-Aided Design of Integrated Circuits Based on VHDL. StateCAD, ModelSim, LeonardoSpectrum*. Moscow, SOLON-Press, 2005, 384 p.
7. Amaru L. G. *New Data Structures and Algorithms for Logic Synthesis and Verification*. Springer Publ., 2017, 156 p.
8. Surkov A.V. Synthesis of self-timed circuits using Synopsys Design Compiler. *Software & Systems*. 2014, no. 4, pp. 24–30 (in Russ.).
9. Turtsevich A.S., Belous A.I., Shvedov S.V., Kutas M.A. The development of CAD microelectronics. CAD JSC "INTEGRAL". *Proc. Research and Practical Conf. "Modern Information and Electronic Technologies"*. Odessa, 2013, pp. 17–20 (in Russ.).
10. Bykovsky S.V., Gorbachev Ya.G., Klyuchev A.O., Penskoy A.V., Platunov A.E. *Interface design of embedded systems (hardware/software co-design)*. Part 2, St. Petersburg, 2016, 106 p.
11. Bibilo P.N., Romanov V.I. *Logical Design of Discrete Devices Using Production and Frame Model of Knowledge Representation*. Minsk, Belarus. navuka Publ., 2011, 279 p.
12. Bibilo P.N., Lankevich Yu. Zhegalkin polynomials in minimization of multilevel representations of Boolean functions based on Shannon's expansion. *Software Engineering*. 2017, no. 3, pp. 369–384 (in Russ.).
13. Bibilo P.N., Avdeev N.A., Kardash S.N., Kirienko N.A., Lankevich Yu.Yu., Loginova I.P., Romanov V.I., Cheremisinov D.I., Cheremisinova L.D. A system for logical design of custom CMOS VLSI functional blocks with reduced power consumption. *Microelectronics*. 2017, vol. 46, no. 1, pp. 72–88 (in Russ.).
14. Brayton K.R., Hactel G.D., McMullen C.T., Sangiovanni-Vincentelli A.L. *Logic Minimization Algorithm for VLSI Synthesis*. Boston, Kluwer Academic Publ., 1984, 193 p.
15. *Espresso(5OCTTOOLS)*. Available at: http://www.ecs.umass.edu/ece/labs/vlsicad/ece667/links/espresso.5.html (accessed October 23, 2019).
16. *Espresso(1OCTTOOLS)*. Available at: http://www.ecs.umass.edu/ece/labs/vlsicad/ece667/links/espresso.1.html (accessed October 23, 2019).
17. Akers S.B. Binary decision diagrams. *IEEE Trans. on Computers*. 1978, vol. C-27, no. 6, pp. 509–516.
18. Bryant R.E., Meinel C. *Ordered Binary Decision Diagrams. Logic Synthesis and Verification*. S. Hassoun, T. Sasao, R.K. Brayton (Eds.). Kluwer Academic Publ., 2002, pp. 285–307.
19. Lokhov A. VLSI functional verification of in the context of Mentor Graphics decisions. *Electronics: Science, Technology, Business*. 2004, no. 1, pp. 58–62 (in Russ.).
20. Chervyakov N.I., Sakhnyuk P.A., Shaposhnikov A.V., Ryadnov S.A. *Modular Parallel Computing Structures of Neuroprocessor Systems*. Moscow, Fizmatlit Publ., 2003, 288 p.
21. Stempkovsky A.L., Kornilov A.I., Semenov M.Yu. Implementation features of digital signal processing devices in the integrated design using modular arithmetic. *Information Tekhnologies*. 2004, no. 2, pp. 2–9 (in Russ.).
22. Gorodetsky D., Villa T. Efficient hardware realization of arithmetic operations for the residue number system by Boolean minimization. *Proc. 13th Intern. Workshop on Boolean Problems*. Bremen, Germany, 2018, pp. 195–203.
23. Zotov V. Architecture features of a new generation of high-performance Virtex-6 FPGA from Xilinx. *Components & Technologies*. 2009, no. 8, pp. 78–85 (in Russ.).