Оптимизация логических схем заказных СБИС в синтезаторе LeonardoSpectrum

Пётр Бибило (г. Минск, Белоруссия)

В статье исследованы различные способы оптимизации логических схем, описанных на языке VHDL, посредством синтезатора LeonardoSpectrum.

Правильный выбор и умелое использование системы синтеза (или синтезатора) является одним из решающих факторов эффективного проектирования заказных СБИС (ASIC – Application-Specific Integrated Circuits).

Во-первых, синтезатор должен поддерживать язык высокого уровня, на котором описан исходный проект (т.е. языки VHDL или Verilog). Во-вторых, синтезатор должен обеспечивать применение целевой библиотеки синтеза, разработанной пользователем, и – в-третьих – допускать гибкое управление процессом синтеза.

Одним из синтезаторов, удовлетворяющих основным требованиям, является LeonardoSpectrum фирмы Mentor Graphics (далее – Leonardo). Этот синтезатор позволяет описать собственную библиотеку логических элементов и проектировать логические схемы по исходным описаниям проектов на языках VHDL и Verilog с учётом различных технологических ограничений.

Синтез в Leonardo можно проводить несколькими способами [1].

Способ 1 – для начинающих пользователей. В окне графической оболочки указывается только исходное VHDL-описание, целевая библиотека синтеза и степень (глубина) оптимизации. Синтез может выполняться в двух режимах: *quick* (быстром) и *standard* (стандартном).

Способ 2 – управление синтезом из графической оболочки. Установки режимов синтеза и ограничений в многочисленных окнах графической оболочки интерпретируются как соответствующие консольные команды.

Способ 3 – управление синтезом с помощью скриптов (сценариев), т.е. наборов консольных команд, собранных в отдельные текстовые файлы.

Для сравнения эффективности способов оптимизации были проведены эксперименты с различными вариантами описания исходных проектов.

В качестве исходных описаний были взяты 25 проектов: 10 проектов описаний комбинационных схем ПЛМ (программируемых логических матриц) из известной библиотеки [2] примеров логических схем, 10 проектов многоуровневой комбинационной логики, два описания таблиц микрокоманд микроконтроллеров и три алгоритмических описания (см. табл. 1). Все описания были выпол-

Таблица 1. Результаты экспериментов

N	Вид описания	Название схемы	n (входы)	m (выходы)	Эксперимент 1 $S_{БМК}$	Эксперимент 2 $S_{БМК}$	Эксперимент 3 $S_{БМК}$	Эксперимент 4 $S_{БМК}$
1	ПЛМ	b2	16	17	3585	2474	**2034**	2163
2		bc0	26	11	5993	5993	3760	**3207**
3		chkn	29	7	1461	1319	1104	**1017**
4		dk48	15	17	389	**389**	395	398
5		ibm	48	17	630	630	538	**530**
6		in0	15	11	2107	2067	1539	**1381**
7		in2	19	10	2217	1845	1405	**1151**
8		prom2	9	21	10671	9852	9421	**7014**
9		signet	39	8	796	727	668	**658**
10		tial	14	8	5013	4505	4533	**4344**
11	Многоуровневая логика	c1355	41	32	1128	1128	1080	**1076**
12		c432	36	7	544	633	499	**401**
13		c880	28	18	270	271	269	**241**
14		dalu	75	16	1803	1803	1254	**503**
15		frg1	28	3	421	266	276	**210**
16		frg2	143	139	3491	3604	3490	**2533**
17		i3	132	6	248	248	244	**236**
18		i8	133	81	2898	2898	2761	**2128**
19		too_larg	38	3	9244	9244	7771	**1047**
20		x4	94	71	1334	1334	1249	**1240**
21	Не полностью определённые функции	verg1	17	61	16388	7571	**6674**	10899
22		verg2	18	63	16082	–	–	**15665**
23	Алгоритмы	uart	n+m=26		1896	1869	1633	**1632**
24		watchdog	n+m=20		461	446	443	**402**
25		timer2	n+m=31		3119	3119	**3089**	3089

нены на языке VHDL. Описания *verg1*, *verg2* (см. табл. 1) представляют собой таблицы микрокоманд отечественных микроконтроллеров, описываемых на языке VHDL в качестве систем не полностью определённых (частичных) логических функций. Последние три описания являются алгоритмическими (в основном используются операторы процессов языка VHDL, операторы назначения сигналов и последовательные операторы): uart – универсальный асинхронный приёмопередатчик информации, представленной последовательным кодом; *watchdog* – сторожевой таймер; *timer2* – таймер-счётчик.

В качестве примера приведём VHDL-описание системы частичных логических функций *sum_sc*. (В подобной форме задавались описания *verg1*, *verg2*, содержащие более двух тысяч строк.) Подсхема *sum_sc* представляет собой сумматор для сложения пары чисел, выбираемых из множества {0, 1, 2}, т.е. неполный сумматор. Данный сумматор описывается системой частичных функций, заданной в таблице 2. Если одно из слагаемых представляет собой число 3, то значения выходов сумматора не определены. Для кодирования каждого из слагаемых необходимы две булевы переменные. Первое число задаётся в виде вектора **a** = (a(1), a(0)), второе – в виде вектора **b** = (b(1), b(0)), старшие разряды – a(1), b(1); старший разряд суммы – s(2), для представления вектора суммы **s** = (s(2), s(1), s(0)) достаточно трёх булевых переменных.

Листинг 1. VHDL-модель схемы sum_sc

```
library ieee;
use ieee.std_logic_1164.all;
entity sum_sc is
 port (a, b : in std_logic_vec-
tor (1 downto 0);
s : out std_logic_vector (2
downto 0));
end;
architecture BEHAVIOR of sum_sc
is
begin
 s <=
"000" when a & b = "0000" else
"001" when a & b = "0001" else
"010" when a & b = "0010" else
"001" when a & b = "0100" else
"010" when a & b = "0101" else
"011" when a & b = "0110" else
"010" when a & b = "1000" else
"011" when a & b = "1001" else
"100" when a & b = "1010" else
"---";
end BEHAVIOR;
```

В качестве целевой библиотеки ASIC была выбрана библиотека проектирования базовых матричных кристаллов (БМК), состоящая из 35 элементов [3, стр.159].

Эксперимент 1 (Исследование оптимизации в режиме *quick*) и *эксперимент 2* (Исследование оптимизации в режиме *standard*) оставляем без комментариев.

Эксперимент 3. Итеративный синтез без смены целевой библиотеки. Синтез схемы от алгоритмических описаний на языке VHDL в Leonardo разбит на два этапа – высокоуровневый синтез, результатом которого является так называемое промежуточное RTL-описание (RTL – Register Transfer Level), и технологическое отображение (technology mapping) [2]. RTL-описание может быть получено повторно в Leonardo (этим он выгодно отличается от других синтезаторов) из результирующего структурного описания синтезированной логической схемы с помощью специальной команды *unmap*. Назовем это описание RTL0. Естественно, описание RTL0 функционально соответствует исходному алгоритмическому VHDL-описанию. По описанию RTL0 может быть проведён повторный синтез схемы FPGA как в синтезаторе Leonardo, так и в других синтезаторах, строящих логические схемы по VHDL-описаниям. На практике было замечено, что вновь построенная схема иногда обладает лучшими характеристиками, чем схема, построенная по исходному VHDL-описанию. Таким образом, синтезатор Leonardo позволяет уменьшить сложность схемы путём повторного (итеративного) синтеза. В эксперименте 3 для каждого исходного описания проводилось пять итераций повторного синтеза и отбирались лучшие результаты.

Эксперимент 4. Итеративный синтез с поочередной сменой целевой библиотеки синтеза. В этом эксперименте схема сначала синтезировалась в библиотеке FPGA (микросхема XC2S100 семейства SPARTAN II), затем выполнялась команда *unmap*, и полученное RTL-описание синтезировалось в целевой библиотеке БМК. Всего проведено пять итераций повторного синтеза со сменой библиотек, и затем отобраны лучшие результаты.

Результаты экспериментов представлены в таблице 1. Жирным шрифтом выделены лучшие – по занимаемой схемой площади – решения. Площадь схемы подсчитывалась как сумма площадей, входящих в данную схему элементов (Leonardo позволяет подсчитать суммарную площадь всех элементов схемы), а площадь элемента выражалась в числе элементарных ячеек БМК. Элементарная ячейка БМК соответствует одному транзистору, входящему в сеть транзисторов, из которых состоит логический элемент. При оценке сложности схемы, синтезированной в библиотеке БМК, в расчёт не принималась площадь, отводимая под межсоединения. Общую площадь схемы можно получить после выполнения трудоёмкого этапа топологического проектирования. На практике установлено, что минимизация суммарной площади всех элементов схемы в общем случае приводит к меньшей площади кристалла.

Для удобства поток примеров обрабатывался единообразно с помощью соответствующих скриптов. Пример скрипта, использованного в эксперименте 2, приведён в листинге 2.

Листинг 2. Скрипт для эксперимента 2

```
clean_all;
set encoding Gray;
set modgen_select Smallest;
set asic_auto_dissolve_limit 500;
set auto_dissolve_limit 500;
read b2.vhd;
load_library bmk.syn;
set -hierarchy flatten
```

Таблица 2. Система не полностью определённых булевых функций

a(1) a(0) b(1) b(0)	s(2) s(1) s(0)
0000	000
0001	001
0010	010
0011	---
0100	001
0101	010
0110	011
0111	---
1000	010
1001	011
1010	100
1011	---
1100	---
1101	---
1110	---
1111	---

```
set effort standard
optimize -target bmk -macro
-area -effort standard -hierar-
chy flatten
report_area -cell_usage
```

Если требовалось получить RTL-описание (эксперименты 3 и 4) для повторного синтеза, то в скрипт (листинг 2) добавлялись две команды: *unmap* и *auto_write b2.vhd*.

В листинге 2, кроме значения *standard*, жирным шрифтом выделены важные для синтеза значения *flatten* параметра hierarchy и значение 500 параметра *auto_dissolve_limit*. Установка значения *flatten* параметра hierarchy ориентирует программу на выполнение синтеза с учётом устранения иерархии описания, т.е. схема синтезируется в виде одного блока, поскольку в этом случае имеется больше возможностей для оптимизации. Параметр *auto_dissolve_limit* задаёт число «растворяемых» элементов при оптимизации. Под «растворением» понимается преобразование структурного описания подсхемы в функциональное описание. По умолчанию значение этого параметра для ASIC равно 30. При эксперименте задаётся значение 500, тогда результирующие схемы имеют меньшую площадь и оптимизируются подсхемы большей размерности. Однако если задать значение 1000, то время синтеза может значительно возрасти. Задание значения *Gray* параметра *encoding* важно при синтезе схем с памятью, – они получаются более экономичными.

В результате проведённых с использованием синтезатора Leonardo экспериментов можно сделать следующие выводы:

- Режим *standard* позволяет значительно лучше оптимизировать логические схемы, чем режим *quick*, что не является неожиданностью. Эксперименты 1 и 2 показывают, сколько можно выиграть по площади, если использовать режим *standard*. Для минимизации площади весьма важен правильный выбор значений параметров: *hierarchy*, *auto_dissolve_limit*;
- Итеративный синтез является эффективным для достаточно больших блоков комбинационной логики и особенно – для частичных функций. Для алгоритмических описаний общего вида эффективность повторного синтеза снижается, так как в таких схемах не выделяются блоки комбинационной логики. Поэтому их отдельная предварительная оптимизация вне синтезатора Leonardo может улучшить характеристики схем, например, значительно уменьшить задержку [4];
- Предварительная логическая И-ИЛИ минимизация матричных форм логических функций (описания ПЛМ) в классе дизъюнктивных нормальных форм значительно уменьшает сложность схем ASIC;
- Предварительный синтез в базисе FPGA по сути заменяет минимизацию в классе ДНФ минимизацией многоуровневых представлений функций, поэтому итеративный синтез со сменой библиотек позволяет получить лучшие результаты, чем простой итеративный синтез;
- Синтезатор Leonardo может быть использован для совместной работы с другими синтезаторами, работающими с языком VHDL, так как Leonardo позволяет сохранять промежуточные и результирующие VHDL-описания логических схем в виде RTL-описаний;
- Синтезатор Leonardo может быть использован при проектировании FPGA в системе WebPack ISE 8.1i (фирма Xilinx); синтез в этом случае осуществляется в режиме *quick*, поэтому площадь схем, синтезированных Leonardo, зачастую бывает больше площади схем, которые получает синтезатор XST, имеющийся в составе WebPack ISE.

Литература

1. *Бибило П.Н.* Системы проектирования интегральных схем на основе языка VHDL. StateCAD, ModelSim, Leonardo-Spectrum. СОЛОН-Пресс, 2005.
2. http://www1.cs.columbia.edu/~cs4861/sis/espresso-examples/ex/.
3. *Бибило П.Н.* Синтез логических схем с использованием языка VHDL. Солон-Р, 2002.
4. *Бибило П.Н., Кочанов Д.А.* Оптимизационные преобразования VHDL-моделей цифровых систем. Современная электроника. 2006. № 5. С. 64–66.

Новости мира News of the World Новости мира

Оптический сенсор уменьшит количество автомобильных аварий

Немецкая компания Continental AG, один из ведущих поставщиков комплектующих для автомобильной промышленности, объявила о завершении разработки нового оптического сенсора, призванного уменьшить количество автомобильных аварий. Встроенный в зеркало заднего вида сенсор с помощью лазерного луча измеряет скорость изменения скорости приближения ближайшего к автомобилю участнику дорожного движения и автоматически отдаёт команды тормозной системе последнего.

Представленное устройство, разработанное в рамках проекта Continental APIA (Active/Passive Integration Approach), постоянно «просматривает» пространство перед автомобилем на 10 м вперёд. Если расстояние до объекта в зоне видимости начнёт уменьшаться слишком быстро, система активирует тормозную систему. Более того, если столкновение неизбежно, система может выпустить подушки безопасности, не дожидаясь удара.

Согласно заверениям компании, представленная система обладает достаточной чувствительностью для обнаружения не только автомобилей, но и велосипедистов и пешеходов. Однако пока убедительные результаты было получены при скорости автомобиля не более 35 км/час. Ожидается, что массовое оснащение автомобилей подобными сенсорами начнётся в 2008 г.

eetimes.com

Intel продемонстрировала прототип 80-ядерного ПК

Чтобы продемонстрировать лидирующие позиции в сфере процессоростроения, Intel представила прототип компьютера с 80-ядерным процессором, получившего рабочее наименование Polaris PC, с пиковой вычислительной способностью порядка 2 терафлоп.

Процессор Polaris был разработан в рамках программы Tera-Scale Computing Research, а первую подложку с 80-ядерными процессорами генеральный директор Intel Пол Отеллини продемонстрировал в конце сентября 2006 г.

Прототип компьютера Polaris PC был разработан компанией Foxconn Electronics, присоединившейся к упомянутой программе около года назад.

digitimes.com