Проектирование комбинационных схем в базисе ПЛИС

Андрей СТРОГОНОВ,
д. т. н.
andreis@hotmail.ru

Цель данной статьи — приобретение начальных сведений по проектированию комбинационных устройств как с использованием ручных методов минимизации булевых функций, так и с использованием САПР ПЛИС Quartus II фирмы Altera, а также изучение некоторых возможностей по управлению логическим синтезом схем.

САПР ПЛИС Quartus II фирмы Altera относится к 4-му поколению, позволяет вести разработку систем на кристалле и демонстрирует высокие характеристики и производительность. В версии 7.2 Quartus II реализована поддержка современных ПЛИС FPGA (программируемые пользователем вентильные матрицы, ППВМ) Cyclone III и Stratix III. САПР Quartus II объединяет проектирование, синтез, размещение элементов, трассировку соединений и верификацию, связь с системами проектирования других разработчиков.

В САПР Quartus II доступна блочная методология проектирования, что помогает увеличить эффективность работы, снизить время проектирования и верификации. LogicLock позволяет проектировать и проверять каждый модуль проекта отдельно. NativeLink обеспечивает связь между САПР Quartus II и программным обеспечением сторонних разработчиков. NativeLink позволяет средствам синтеза сторонних разработчиков САПР преобразовывать свои примитивы напрямую в примитивы ПЛИС фирмы Altera.

Технология размещения элементов и трассировки соединений PowerFit в САПР Quartus II использует временные параметры, заданные разработчиком, для оптимального составления схемы и размещения логических элементов. Интеллектуальный алгоритм трассировки по временным параметрам в САПР Quartus II уделяет основное внимание соединениям, чувствительным к временным параметрам. Подобные соединения оптимизируются в первую очередь — для уменьшения задержек и достижения максимальной производительности (fMAX). Дальнейшее улучшение параметра fMAX достигается использованием новейшей архитектуры, такой как в семействе ПЛИС Stratix. Эта передовая технология размещения элементов и трассировки соединений помогает пользователям САПР Quartus II достичь максимальной производительности и обладает самым малым временем компиляции проекта среди подобных САПР.

В САПР Quartus II разработано отладочное средство SignalProbe и логический анализатор SignalTap. Технологии SignalTap и SignalProbe могут работать совместно со средствами синтеза сторонних разработчиков. Технология аппаратной отладки SignalProbe позволяет пользователям последовательно соединять внутренние точки устройства со свободными зарезервированными выводами для анализа с помощью осциллографа или логического анализатора.

Логический анализатор SignalTap позволяет разработчикам собрать данные с любых внутренних точек и входов/выходов ПЛИС в режиме реального времени при работе системы. САПР Quartus II позволяет использовать в проекте мегафункцию, содержащую логический анализатор. Данные собираются и сохраняются в блоках встроенной памяти ПЛИС и направляются в САПР Quartus II через загрузочный кабель. Кроме того, имеется возможность подавать внутренние сигналы на выводы ПЛИС для дальнейшего мониторинга. В САПР Quartus II используется технология PowerGauge — интегрированное средство анализа энергопотребления.

Теоретической базой при проектировании цифровых устройств являются булева алгебра, двоичная арифметика и теория конечных автоматов. Функцией алгебры логики (ФАЛ) называется функция, которая, как и ее аргументы, может принимать только два значения: 0 и 1. ФАЛ могут быть представлены различными способами: словесным описанием (назначением, определением); таблицей истинности; аналитическим выражением (использование базовых логических операций И, ИЛИ, НЕ, определяемых аксиомами булевой алгебры); картой Карно; переключательной схемой; диаграммой Венна; геометрическим способом (гиперкубами); диаграммой двоичного решения.

Привязка к конкретным архитектурам ПЛИС сильно ограничивает возможности использования формальных методов синтеза. Методы синтеза комбинационных схем при реализации их в базисе ПЛИС подразделяются на методы двухуровневого и многоуровневого синтеза. Методы двухуровневого синтеза используют для ПЛИС с двумя матрицами И-ИЛИ — так называемые программируемые логические матрицы (ПЛМ или Field Programmable Logic Array, FPLA). Методы многоуровневого синтеза используются для современных ПЛИС. Прохождение сигнала от входа к выходу представляется в виде логической сети, удовлетворяющей заданным ограничениям. Главными критериями оптимизации при многоуровневом синтезе являются длина критического пути (максимальная задержка сигнала), площадь (стоимость), занимаемая логической сетью, и т. д. При этом решается задача разделения проектируемой схемы на более простые части, которые могут быть реализованы на функциональных преобразователях ПЛИС. Функциональные преобразователи ПЛИС включают в себя настраиваемые средства реализации логических функций и триггер. Наиболее часто логические функции реализуются или в виде суммы логических произведений (sum of product), или на таблицах перекодировок (Look-up table, LUT). Таблицу перекодировок иногда называют таблицей истинности или логическим генератором. Четырехвходовая таблица истинности представляет собой однобитное ОЗУ на 16 ячейках (16x1). Такой блок памяти позволяет реализовывать логические функции, например, логическую функцию «Исключающее ИЛИ». Например, ПЛИС семейства Virtex-5 фирмы Xilinx имеет 6-входовую таблицу истинности и представляет блок памяти с организацией 64×1. В ПЛИС APEX20K таблица перекодировок — четырехвходовая.

ПЛИС с функциональными преобразователями на базе сумм произведений позволя-

ют проще реализовать сложные логические функции, а с преобразователями на базе таблиц перекодировок — создавать насыщенные триггерами цифровые устройства.

Рассмотрим примеры минимизации булевых функций с использованием САПР Quartus II и ПЛИС APEX20K. Архитектура APEX20K сочетает в себе как достоинства FPGA ПЛИС с таблицами перекодировок, входящими в состав логического элемента, так и термы произведений (Product Term), реализуемые с использованием макроячеек (macrocells) и характерные для ПЛИС CPLD. ПЛИС семейства APEX20K также содержат реконфигурируемые системные блоки памяти (ESB — embedded system block).

Булевы выражения являются описаниями комбинационных схем. Под комбинационной схемой понимается логическая схема, значения выходных сигналов которой в каждый момент времени полностью определяются значениями сигналов на ее входах. Поэтому комбинационная схема не обладает запоминающими свойствами. Иногда говорят, что комбинационная схема — это конечный автомат с одним состоянием или последовательностная схема без памяти.

По заданному алгоритму — булевому логическому уравнению, построим вариант комбинационного устройства в общем виде:

$$\overline{A}\overline{B}\overline{C}\overline{D}+\overline{A}BCD+A\overline{B}CD+AB\overline{C}D. \quad (1)$$

Пользуясь правилами алгебры логики Буля, проведем минимизацию исходного логического уравнения (расчетный метод минимизации):

$$\overline{A}\overline{B}\overline{C}\overline{D}+\overline{A}BCD+A\overline{B}CD+AB\overline{C}D =$$
$$= (\overline{A}\overline{B}+CD)(\overline{A}B+\overline{C}D)+$$
$$+(\overline{A}\overline{B}+CD)(A\overline{B}+\overline{C}\overline{D}) =$$
$$= \overline{A}\overline{B}\overline{C}\overline{D}+\overline{A}BCD+A\overline{B}CD+ABCD+AB\overline{C}\overline{D} = \quad (2)$$
$$= (AD+\overline{A}\overline{D})(BC+\overline{B}\overline{C}) = \quad (3)$$
$$= \overline{A\oplus D}\times\overline{B\oplus C} = \quad (4)$$
$$= \overline{A\oplus D+B\oplus C}. \quad (5)$$

Далее, по заданному логическому уравнению (1) построим таблицу истинности (табл. 1). По последнему правому столбцу таблицы f построим каноническую сумму минтермов или стандартную сумму произведений:

$$f = \overline{A}\overline{B}\overline{C}\overline{D}+ABCD+\overline{A}BC\overline{D}+A\overline{B}C\overline{D}. \quad (6)$$

Сравнивая логические уравнения (2) и (6), видим, что они идентичны. По таблице 1 построим карту Карно для функции четырех переменных. Карта Карно (табличный метод минимизации) является специальной компактной формой таблицы истинности (табл. 2).

Таблица 1. Таблица истинности для булевого выражения (1)

N п/п	A	B	C	D	$\overline{A}\overline{B}\overline{C}\overline{D}$	$\overline{A}B\overline{C}\overline{D}$	$A\overline{B}\overline{C}\overline{D}$	$\overline{A}\overline{B}C\overline{D}$	$\overline{A}\overline{B}\overline{C}\overline{D}+\overline{A}BCD$	$\overline{A}\overline{B}\overline{C}\overline{D}+A\overline{B}CD$	f
0	0	0	0	0	1	1	0	0	0	1	1
1	0	0	0	1	1	0	1	0	0	0	0
2	0	0	1	0	0	1	1	0	0	0	0
3	0	0	1	1	1	1	1	0	0	0	0
4	0	1	0	0	1	0	0	1	0	0	0
5	0	1	0	1	1	0	1	1	0	0	0
6	0	1	1	0	0	0	1	1	1	0	1
7	0	1	1	1	1	0	1	0	0	0	0
8	1	0	0	0	0	1	0	1	0	0	0
9	1	0	0	1	0	0	1	1	1	0	1
10	1	0	1	0	0	1	1	1	0	0	0
11	1	0	1	1	0	1	1	0	0	0	0
12	1	1	0	0	1	1	0	1	0	0	0
13	1	1	0	1	1	0	0	1	0	0	0
14	1	1	1	0	0	1	0	1	0	0	0
15	1	1	1	1	1	1	0	0	0	1	1

Таблица 2. Карта Карно для функции четырех переменных

AB \ CD	00	01	11	10
00	1	0	0	0
01	0	0	0	1
11	0	0	1	0
10	0	1	0	0

Кроме расчетного метода минимизации (метод непосредственных преобразований), наибольшее распространение получили: расчетно-табличный метод (метод Квайна-МакКласки); метод Петрика (развитие метода Квайна-МакКласки); метод гиперкубов; метод факторизации; метод функциональной декомпозиции и др.

По заданному логическому уравнению (1) и его минимизированным решениям (4) и (5) нарисуем электрические схемы с использованием схемного редактора САПР ПЛИС Quartus II (рис. 1). На рис. 1 обозначено: *F* — выход комбинационной схемы по уравнению (1); *QQQQ* — выход схемы по уравнению (4); *MMMM* — выход схемы по уравнению (5). Для размещения проектируемой комбинационной схемы выберем ПЛИС APEX20K.

Для этого необходимо зайти в окно настроек компилятора Processing/Compiler Settings, выбрать вкладку Chip&Devices и заказать семейство APEX20KE с автоматическим подбором ПЛИС из данного семейства (переключатель Auto device selected by the Compiler from the 'Available devices' list). Можно также выбрать нужную ПЛИС из списка доступных кристаллов Available device. На вкладке Mode (режим компиляции) зададим уровень Full compilation (полная компиляция) и скорость компиляции (Normal compilation/less disk space — нормальная компиляция с меньшими затратами дискового пространства). На вкладке Synthesis&Fitting задаются режимы оптимизации задержек — как внутренних, так и между входом и выходом.

В меню Project/Option&Parameter Settings откроем вкладку Options, выберем в настрой-

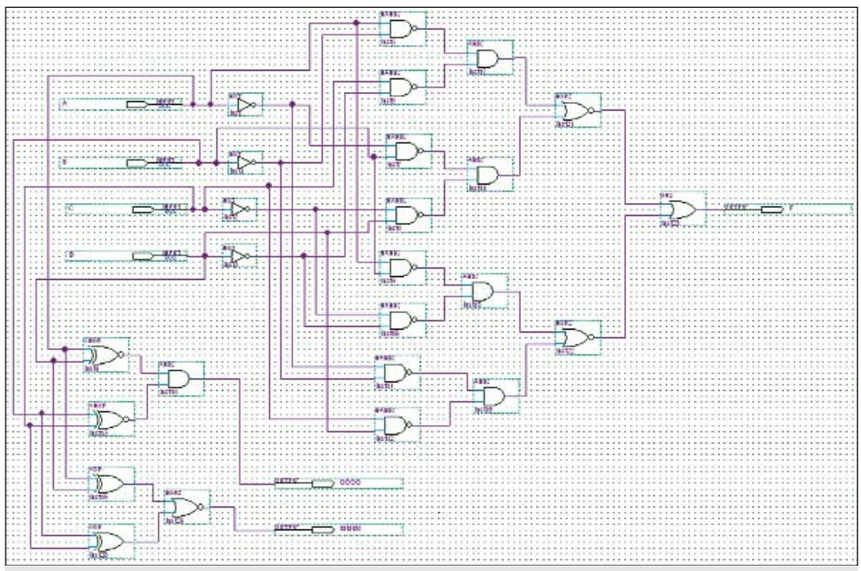

Рис. 1. Комбинационная схема по заданному логическому уравнению (1) и его минимизированным решениям (4) и (5) в схемотехническом редакторе САПР ПЛИС Quartus II

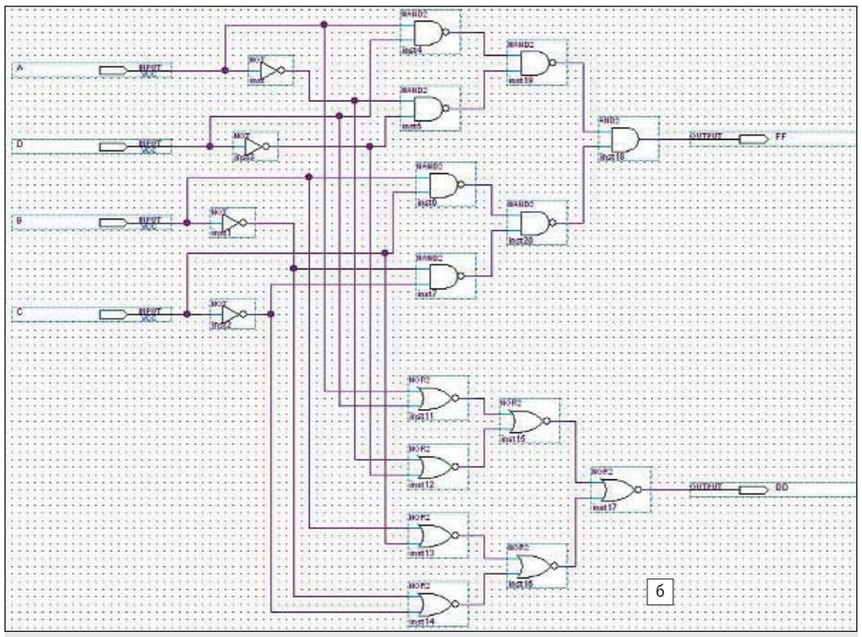

Рис. 2. а) Комбинационная схема по заданному логическому уравнению (7) в базисе И-НЕ;
б) по заданному логическому уравнению (8) в базисе ИЛИ-НЕ в схемотехническом редакторе САПР ПЛИС Quartus II

ках технологии маппирования (отображение проектируемой электрической схемы в ресурсы ПЛИС) Technology Mapper APEX — LUT.

В момент компиляции осуществляется минимизация комбинационных схем, построенных по уравнению (1) и по уравнениям (4) и (5). Файл отчета (его можно посмотреть в навигаторе проекта, отчет Equations) компилятора в САПР ПЛИС Quartus II после логического синтеза имеет вид:

```
A1L6 = A & D & (B $ !C) # !A & !D & (B $ !C);
A = INPUT();
D = INPUT();
B = INPUT();
C = INPUT();
F = OUTPUT(A1L6);
QQQQ = OUTPUT(A1L6);
MMMM = OUTPUT(A1L6);
```

Представим булево выражение (A1L6) в привычном для нас виде. Для этого к булевому логическому уравнению (3) применим законы и теоремы алгебры логики Буля:

$$(AD+\overline{A}\,\overline{D})(BC+\overline{B}\,\overline{C}) =$$
$$= (AD+\overline{A}\,\overline{D})(B\oplus\overline{C}) =$$
$$= AD(B\oplus\overline{C})+\overline{A}\,\overline{D}(B\oplus\overline{C}).$$

Видим, что булево уравнение A1L6, построенное компилятором, есть уравнение (3), построенное с использованием элементов «Исключающее ИЛИ». Компилятор автоматически подобрал ПЛИС EP20K30ETC144-1. Оказался задействованным 1 логический элемент из 1200 доступных, 0 макроячеек и 0 бит из 24576 ESB-бит.

Выберем другую технологию маппирования для ПЛИС APEX — термы произведений (установка Product Term) и выполним логический синтез. Файл отчета в САПР ПЛИС Quartus II после логического синтеза имеет вид:

```
A1P21_p1_out = A & D & !B & !C;
A1P21_p2_out = A & D & B & C;
A1P21 = A1P21_p1_out # A1P21_p2_out # A1P31;
A1P01_p1_out = A & D & !B & !C;
A1P01_p2_out = A & D & B & C;
A1P01 = A1P01_p1_out # A1P01_p2_out # A1P11;
A1P31_p2_out = !A & !D & B & C;
A1P31_p1_out = !A & !D & !B & !C;
A1P31 = A1P31_p2_out # A1P31_p1_out;
A1P11_p2_out = !A & !D & B & C;
A1P11_p1_out = !A & !D & !B & !C;
A1P11 = A1P11_p2_out # A1P11_p1_out;
A = INPUT();
D = INPUT();
B = INPUT();
C = INPUT();
F = OUTPUT(A1P21);
QQQQ = OUTPUT(A1P01);
MMMM = OUTPUT(A1P01);
```

Представим полученную информацию в привычном для нас виде:

$$A1P21 = F =$$
$$= AD\overline{B}\,\overline{C}+ADBC+\overline{A}\,\overline{D}BC+\overline{A}\,\overline{D}\,\overline{B}\,\overline{C},$$

$$A1P01 = QQQQ =$$
$$= \overline{A}\,\overline{D}BC+\overline{A}\,\overline{D}\,\overline{B}\,\overline{C}+AD\overline{B}\,\overline{C}+ADBC.$$

Таким образом, на этапе автоматической компиляции САПР Quartus выдал нам минимизированное уравнение A1P21, соответствующее уравнению (2). Компилятор автоматически подобрал ПЛИС EP20K30ETC144-1. Оказалось задействовано 0 логических элементов из 1200 доступных, 512 бит памяти для хранения термов произведений из 24576 ESB-бит (2%) и 4 макроячейки.

Анализируя две технологии маппирования, можно сделать вывод, что реализация логических функций на базе таблиц перекодировок более эффективна за счет использования операции «Исключающее ИЛИ». Однако САПР Quartus нашел компромиссное решение, разместив булевы выражения в неиспользуемых системных блоках памяти. Чтобы убедиться в том, что компилятор в случае выбора LUT-таблиц использует в булевых выражениях операцию «Исключающее ИЛИ», необходимо удалить комбинационные схемы, построенные по уравнениям (4) и (5), и скомпилировать проект заново, а затем в навигаторе проекта посмотреть отчет Eduations.

В случае, если использовать установку Technology Mapper APEX-Auto, то окажется задействованным 1 логический элемент из 1200 доступных и 0 бит из 24576 ESB-бит, то есть предпочтение будет отдано таблицам перекодировок.

Применив закон двойного отрицания, реализуем устройство в двух основных базисах (И-НЕ, ИЛИ-НЕ), что позволяет на практике в случае использования ТТЛ ИС средней степени интеграции КР1533, КР1531, К555, К155 получить существенный выигрыш в технико-экономических показателях. В начале реализуем устройство в базисе И-НЕ. Целесообразно воспользоваться уравнением (3):

$$\overline{\overline{(AD+\overline{A}\,\overline{D})(BC+\overline{B}\,\overline{C})}} =$$
$$= \overline{\overline{AD+\overline{A}\,\overline{D}}+\overline{BC+\overline{B}\,\overline{C}}} =$$
$$= \overline{\overline{AD}\,\overline{\overline{A}\,\overline{D}}+\overline{BC}\,\overline{\overline{B}\,\overline{C}}} =$$
$$= \overline{\overline{AD}\,\overline{\overline{A}\,\overline{D}}}\;\overline{\overline{BC}\,\overline{\overline{B}\,\overline{C}}}. \qquad (7)$$

Комбинационная схема по логическому уравнению (7) в базисе И-НЕ показана на рис. 2а. Продолжим дальнейшее применение теорем булевой алгебры и построим комбинационную схему в базисе ИЛИ-НЕ:

$$\overline{\overline{AD}\,\overline{\overline{A}\,\overline{D}}}\;\overline{\overline{BC}\,\overline{\overline{B}\,\overline{C}}} =$$
$$= \overline{(\overline{A}+\overline{D})(A+D)+(\overline{B}+\overline{C})(B+C)} =$$
$$= \overline{(\overline{A}+\overline{D})(A+D)}\;\overline{(\overline{B}+\overline{C})(B+C)} =$$
$$= \overline{\overline{(A+\overline{D}+\overline{A}+\overline{D})}\;\overline{(\overline{B}+\overline{C}+\overline{B}+\overline{C})}} =$$
$$= \overline{\overline{A}+\overline{D}+\overline{A}+\overline{D}}+\overline{\overline{B}+\overline{C}+\overline{B}+\overline{C}}. \qquad (8)$$

Комбинационная схема по логическому уравнению (8) в базисе ИЛИ-НЕ показана на рис. 2б. Комбинационная схема по логическому уравнению (7) или (8) требует 11 логических элементов, а по уравнению (1) — 19.

Проведем логический синтез схем, построенных по уравнениям (7) и (8) в САПР ПЛИС Quartus II. Файл отчета в Quartus II после логического синтеза имеет вид (булева функция реализуется на таблицах перекодировок):

```
A1L7 = C & B & (D $ !A) # !C & !B & (D $ !A);
C = INPUT(); B = INPUT(); D = INPUT(); A = INPUT();
FF = OUTPUT(A1L7);
DD = OUTPUT(A1L7);
```

Представим полученную информацию в привычном для нас виде:

$$FF = DD = CB(D \oplus \overline{A}) + \overline{C}\overline{B}(D \oplus \overline{A}).$$

Выражение A1L7 соответствует уравнению (3). Компилятор автоматически подобрал ПЛИС EP20K30ETC144-1. Оказался задействованным 1 логический элемент из 1200 доступных, 0 бит из 24576 ESB-бит и 0 макроячеек.

Выполним функциональное моделирование (без учета реальных задержек распространения сигналов в ПЛИС APEX) с использованием САПР Quartus II и проверим правильность построения таблицы истинности (табл. 1) и булевых уравнений. Для выполнения функционального моделирования необходимо в меню Processings/Simulator Settings открыть вкладку Mode, выбрать режим симулирования Functional. В противном случае, по умолчанию, будет выполнено временное моделирование с учетом задержек распространения сигналов (Timing).

Переберем все комбинации входных сигналов, когда на выходе функции *F* появляется 1. Результаты моделирования представлены на рис. 3. Сравнивая таблицу истинности (табл. 1) и результаты моделирования, представленные на рис. 3, видим, что проектируемая комбинационная схема работает правильно.

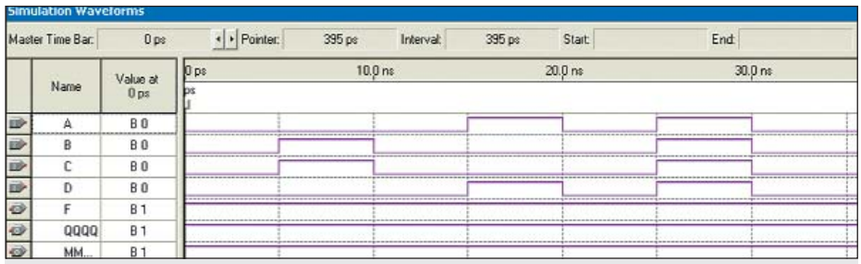

Рис. 3. Результаты функционального моделирования комбинационной схемы по заданному логическому уравнению (1)

В случае, если не требуется управлять логическим синтезом при компиляции проекта, рекомендуется в настройках технологии маппирования Technology Mapper использовать установку Auto, что позволит более эффективно использовать ресурсы ПЛИС. Пользователь может самостоятельно управлять ресурсами ПЛИС при компиляции проекта, однако для этого необходимо иметь представление об архитектуре используемой ПЛИС. Например, при проектировании комбинационных схем для реализации в базисе ПЛИС семейства APEX, если выбирается установка Product Term, то компилятор использует макроячейки и реконфигурируемые блоки памяти, а если LUT-таблицы, то логические элементы. Однако при этом может изменяться нагрузочная способность (коэффициент разветвления по входам и выходам).

Литература

1. Хуторной С. Система Excalibur — средство разработки SoC-решений фирмы Altera. Часть 3. САПР Quartuse // ChipNews. 2001. № 9.
2. Соловьев В., Климкович А. Введение в проектирование комбинационных схем на ПЛИС // ChipNews. 2003. № 5.
3. Тарасов И. Сравнительный анализ архитектуры основных семейств FPGA фирмы Xilinx // Компоненты и технологии. 2005. № 5.
4. Тарасов И. Обзор архитектуры ПЛИС семейства Virtex-5 // Компоненты и технологии. 2006. № 9.
5. Грушвицкий Р., Михайлов М. Проектирование в условиях временных ограничений: компиляция проектов // Компоненты и технологии. 2007. № 12.
6. Стешенко В. Б. ПЛИС фирмы Altera: элементная база, система проектирования и языки описания аппаратуры. М.: Издательский дом «Додэка-XXI», 2002.