Московский государственный технический университет им. Н.Э. Баумана Калужский филиал

Кафедра “САУ и Электротехники”

ЭИУ3-КФ

Расчётно-пояснительная записка к курсовой работе

тема:

“Разработка двоичного сумматора по модулю 13”

по дисциплине:

Микропроцессорные устройства систем

управления

Калуга 2009

Содержание

# Введение

# 1. Теоретическая часть

# 1.1 Логическое проектирование КЛС

# 1.2 Логические последовательности

# 1.3 Программное обеспечение

# 1.3.1 Decomposer

# 1.3.2 Пакет WebPACK ISE

# 1.3.2.1 Основные характеристики пакета WebPACK ISE

# 2. Практическая часть

# 2.1 Формирование логической последовательности

# 2.2. Синтез сумматора при помощи пакета Decomposer

# 2.3 Описание сумматора на языке VHDL

# 2.4 Сравнительный анализ используемых ресурсов для различных вариантов реализации схемы

# 2.5 Покрытие блоков

# Вывод

Список литературы

# Введение

В настоящее время известно множество методов синтеза комбинационных логических схем (КЛС). Практически все они включают следующие этапы:

1. Формальное описание поведения КЛС посредством таблицы истинности;

2. Получение совершенных дизъюнктивных нормальных форм (СДНФ) для всех собственных функций КЛС;

3. Минимизация СДНФ с использованием аналитических (Квайна-Мак-Класки) или графических (карт Карно, диаграмм Вейча) методов;

4. Переход с помощью формальной процедуры от системы минимизированных формул к графическому изображению схемы на элементах «И», «ИЛИ» и «НЕ».

Такой подход сложился исторически, поскольку цифровые устройства изначально реализовывались релейно-контактными схемами, а затем – схемами на бесконтактных логических элементах (ламповых, магнитных, магнитополупроводниковых и полупроводниковых). На этом этапе получили широкое распространение алгебраические методы в классе дизъюнктивных нормальных форм. Дело в том, что как контактные, так и первые бесконтактные электронные логические схемы реализовывали классический базис «И», «ИЛИ» и «НЕ», а дизъюнктивные нормальные формы представляют логические функции именно в этом базисе.

При большом числе входов (более пяти-шести) выполнение этих операций становится затруднительным даже для одной логической функции.

Позже на смену контактным и бесконтактным элементам «И», «ИЛИ» и «НЕ» пришли интегральные логические схемы, которые в одном кристалле полупроводника реализуют сложную логическую структуру. Базисом интегральных схем стали функции Шеффера («И-НЕ»), Пирса («ИЛИ-НЕ») и логическая функция «И-ИЛИ-НЕ». Некоторые из изменённых методов по-прежнему используют минимизацию дизъюнктивных нормальных форм с последующим преобразованием найденных минимальных формул в логические формулы в базисе «И-НЕ» либо «ИЛИ-НЕ». Другие методы используют представление собственных функций синтезируемой схемы в виде совершенных нормальных форм в указанных базисах и минимизацию в этих базисах. Но теперь уже не стало однозначного соответствия между числом вхождений букв в булевых формулах и числом логических элементов, поэтому минимизация формул не всегда приводит к упрощению логических схем.

Следует заметить, что разработка устройств с использованием программируемых БИС невозможна без применения средств и систем автоматизированного проектирования (САПР). «Ручная» разработка устройств, содержащих сотни и тысячи вентилей является весьма трудоёмкой и занимает достаточно длительное время. Особо значимыми становятся процедуры отладки и верификации проектных решений.

Бурное развитие современной интегральной микросхемотехники, особенно программируемых логических интегральных схем (ПЛИС), привело к тому, что алгебраическая методология логического проектирования перестала поспевать за технологическим прогрессом. Изменчивость базиса требует разработки всё новых алгебраических методов. В то же время привязка к конкретному логическому базису сильно ограничивает возможности использования формальных методов синтеза. Кроме того, в качестве конфигурируемых логических блоков (КЛБ) современных ПЛИС типа FPGA (Field Programmable Gate Arrays) используются логические модули на основе мультиплексоров или программируемых ПЗУ (LUT – Look-Up Tables). В этом случае при проектировании КЛС возникает задача разделения сложной схемы на более простые части, которые могут быть реализованы на указанных типах КЛБ.

## 1. Теоретическая часть

## 

## 1.1 Логическое проектирование КЛС

Комбинационной логической схемой называется устройство, изображённое на рис. 1.



Рис.1. Комбинационная логическая схема

Выходные функции называются собственными функциями комбинационных логических схем (КЛС). Изображённой на рис. 1 КЛС соответствует система из собственных функций от аргументов.



1.2 Логические последовательности

Традиционным является задание собственных функций в виде таблицы истинности или алгебраических выражений. Алгебраические формулы не являются инвариантными по отношению к базису, поэтому от такого способа следует отказаться. Таблицы истинности инвариантны по отношению к базису, но достаточно громоздки и неудобны. Если договориться, что наборы состояний входов всегда перечисляются в порядке возрастания их числовых эквивалентов, то их из таблицы можно исключить. Останется лишь выходной столбец, который удобнее записывать в виде строки. Полученная таким образом логическая (числовая) последовательность представляет собой компактный способ задания логических функций.

Процесс логического синтеза КЛС включает выполнение следующих этапов:

1) абстрактный синтез;

2) структурный синтез;

3) структурный анализ.

На этапе абстрактного синтеза решается задача формального описания функционирования проектируемой схемы. В качестве исходной информации, как правило, используется словесное описание алгоритма работы схемы. В результате решения этой задачи получаются собственные функции КЛС. Эту задачу легче всего решать с помощью таблицы истинности из которой затем получается логическая последовательность.

На этапе структурного синтеза заданы логические элементы, из которых строится схема, и система собственных функций (то есть задача абстрактного синтеза решена). Необходимо найти схему соединения логических элементов для реализации заданных собственных функций.

Решение задачи структурного синтеза разбивается на три этапа:

1) абстрактно-структурный синтез. На этом этапе сложная схема делится на более простые части, то есть производится декомпозиция. Критерий разделения – уменьшение общей сложности описания схемы.

2) детализация. На этом этапе производится деление схемы до блоков, сложность которых соизмерима со сложностью заданных для покрытия элементов. При этом не требуется уменьшения сложности схемы. Если сложность покрывающего элемента больше сложности покрываемого блока, то с помощью процедуры анализа объединяются несколько блоков в один. Покрытие производится лишь после выравнивания сложностей блока и логического элемента.

3) покрытие абстрактной схемы заданными логическими элементами. На данном этапе производится формальное замещение получившихся при детализации блоков логическими элементами заданного типа.

В результате выполнения этих этапов получается схема, состоящая из заданных логических элементов. Задача структурного синтеза всегда имеет множество решений (то есть схемы соединения элементов для реализации заданных собственных функций могут быть совершенно различными). Поэтому процедуру структурного синтеза следует проводить таким образом, чтобы получить схему, содержащую минимальное количество логических блоков (элементов).

Поскольку при формальном покрытии не принимаются во внимание никакие другие блоки кроме покрываемого, синтезированная схема, как правило, оказывается избыточной. Это вызывает необходимость проведения оптимизации полученной схемы.

## 

## 1.3 Программное обеспечение

1.3.1 Decomposer

Бурное развитие современной интегральной микросхемотехники, особенно программируемых логических интегральных схем (ПЛИС), привело к тому, что алгебраическая методология логического проектирования перестала поспевать за технологическим прогрессом. Изменчивость базиса требует разработки всё новых алгебраических методов. В то же время привязка к конкретному логическому базису сильно ограничивает возможности использования формальных методов синтеза. Кроме того, в качестве конфигурируемых логических блоков (КЛБ) современных ПЛИС типа FPGA (Field Programmable Gate Arrays) используются логические модули на основе мультиплексоров или программируемых ПЗУ (LUT – Look-Up Tables). В этом случае при проектировании КЛС возникает задача разделения сложной схемы на более простые части, которые могут быть реализованы на указанных типах КЛБ.

Разработанный в Калужском филиале МГТУ им. Н.Э. Баумана программный пакет Decomposer предназначен для решения задач автоматизированного логического синтеза цифровых схем с использованием методов многоуровневой декомпозиции и их реализации на микросхемах программируемой логики.

Возможности САПР:

- проведение параллельной и последовательной декомпозиции;

- детализация схемы до уровня двухвходовых блоков;

- анализ декомпозированной схемы.

Кроме того, пакет Decomposer позволяет получить описание синтезированной схемы на языке VHDL, что даёт возможность интегрироваться в специализированные пакеты программ (например, WebPACK) с целью получения файлов для «прошивки» микросхем программируемой логики и практической реализации спроектированных цифровых устройств.

Основные свойства VHDL:

Одно из главных свойств - это способность описывать аппаратуру и ее работу во времени. Поэтому основными в VHDL являются такие близкие разработчику понятия, как объект проекта, интерфейс, порт, архитектура, сигнал, атрибуты сигнала, операторы параллельного присвоения, процесс и др.

Сигнал в VHDL трактуется весьма широко и может быть скалярным (целым, вещественным, битовым и т. д.) или векторным (шинным), булевым или многозначным. Многозначная (реально 5-, 9- или 12-значная) логика дает возможность выявлять с помощью моделирования такие явления в схемах, как гонки, неопределенности на выходах схем (например, асинхронного RS-триггера при запрещенных значениях входных сигналов), решать задачи мультиплексирования шины.

### 1.3.2 Пакет WebPACK ISE

Программируемые логические интегральные схемы (ПЛИС) все более широко используются для создания цифровых систем различного назначения. Фирма Xilinx®, являясь ведущим мировым производителем ПЛИС, предоставляет разработчикам широкий спектр кристаллов с различной технологией производства, степенью интеграции, архитектурой, быстродействием, потребляемой мощностью и напряжением питания, выпускаемых в различных типах корпусов и в нескольких вариантах исполнения, включая промышленное, военное и радиационно-стойкое [1–5].

Кристаллы, выпускаемые фирмой Xilinx, в полной мере реализуют преимущества ПЛИС по сравнению с «жесткой логикой»:

* высокое быстродействие;
* возможность перепрограммирования непосредственно в системе;
* высокая степень интеграции, позволяющая разместить цифровое устройство в одном кристалле и тем самым снизить время и затраты на трассировку и производство печатных плат;
* сокращение времени цикла разработки и производства устройства;
* наличие мощных инструментов САПР, позволяющих устранить возможные ошибки в процессе проектирования устройства;
* сравнительно низкая стоимость (в пересчете на один логический вентиль);
* возможность последующей реализации проектов ПЛИС для серийного производства в виде заказных СБИС, что позволяет значительно снизить их себестоимость.

До недавнего времени, несмотря на все достоинства ПЛИС Xilinx, существовало обстоятельство сдерживающее их применение (особенно недорогих кристаллов при разработке несерийных устройств) — необходимость дополнительных затрат на приобретение пакета программных средств проектирования и программирования. Чтобы устранить это препятствие, фирма Xilinx предоставила разработчикам возможность использовать бесплатное программное обеспечение — пакет WebPACK™ ISE™ (Integrated Synthesis Environment). Цель настоящей публикации — познакомить разработчиков цифровых устройств с возможностями САПР WebPACK ISE и основами методики выполнения проектов в среде данного пакета.

### 1.3.2.1 Основные характеристики пакета WebPACK ISE

Программные средства WebPACK ISE представляют собой систему сквозного проектирования, которая реализует все этапы создания цифрового устройства на базе ПЛИС, включая программирование кристалла: разработка проекта, синтез, моделирование, трассировка и загрузка в кристалл. Версия 3.3WP8.0 САПР WebPACK ISE предназначена для проектирования цифровых устройств на базе ПЛИС производства Xilinx, относящихся как семействам CPLD: XC9500, XC9500XL, XC9500XV, XCR22V10, XCR3000 (XPLA1\_3, XPLA2), XCR3000XL (XPLA3), XCR5000 (XPLA1\_5), так и FPGA: Spartan™-II, Virtex™-E (только кристалл XCV300E), Virtex-II (кристаллы 2V40, 2V80 и 2V250).

Отличительные особенности пакета:

* поддержка различных методов описания проектируемых устройств (графических и текстовых);
* возможность использования проектов, подготовленных в других системах проектирования, в том числе в среде пакета Altera MAX+PlusII™;
* наличие схемотехнического редактора, укомплектованного набором обширных библиотек;
* интеллектуальные средства создания HDL (Hardware Description Language)-описаний, формирующие шаблоны на основании информации, предоставляемой пользователем, для языков описания аппаратуры VHDL, Verilog™ и ABEL™ HDL;
* высокоэффективные средства синтеза HDL-проектов, поддерживающие языки VHDL, Verilog и ABEL HDL, с возможностью оптимизации;
* развитые средства верификации проекта, позволяющие сократить полное время разработки устройства за счет обнаружения возможных ошибок на более ранних стадиях проектирования и сокращения длительности и количества возможных итераций;
* автоматические средства трассировки проекта в кристаллы различных семейств ПЛИС Xilinx с учетом оптимизации проекта по различным параметрам;
* средства программирования кристаллов семейств ПЛИС Xilinx, выполненных по различной технологии (CPLD и FPGA), поддерживающие несколько типов загрузочных кабелей JTAG-интерфейса;
* удобный для разработчика пользовательский интерфейс и наличие в каждом модуле пакета справочной системы, сокращающие время освоения САПР;
* наличие интегрированного с пакетом САПР набора инструментов и утилит других фирм, предоставляющих дополнительные удобства в процессе проектирования, включающего утилиту генерации тестовых сигналов HDL Bencher™, программу моделирования ModelSim XE Starter™ и редактор диаграмм состояний StateCAD™.

# 2. Практическая часть

## 2.1 Формирование логической последовательности

Десятичный сумматор с кодом 7-4-2-1. Этот код позиционный. Каждому разряду кодовой комбинации сопоставляется определенный весовой коэффициент, зависящий от позиции этого разряда. Если задана кодовая комбинация , где – двоичные символы, то десятичную цифру, соответствующую этой кодовой комбинации можно определить по формуле:



Таким образом, данный код имеет следующие обозначения цифр:

«0» – 0000; «4» – 0100; «8» – 1001; «12» - 1101.

«1» – 0001; «5» – 0101; «9» – 1010;

«2» – 0010; «6» – 0110; «10» - 1011;

«3» – 0011; «7» – 1000; «11» - 1100;

Для данного сумматора получена следующая логическая последовательность:

00 01 01 02 02 03 03 04 04 05 05 06 06 08 \*\* \*\*

08 09 09 0A 0A 0B 0B 0C 0C 0D 0D 10 \*\* \*\* \*\* \*\*

01 02 02 03 03 04 04 05 05 06 06 08 08 09 \*\* \*\*

09 0A 0A 0B 0B 0C 0C 0D 0D 10 10 11 \*\* \*\* \*\* \*\*

02 03 03 04 04 05 05 06 06 08 08 09 09 0A \*\* \*\*

0A 0B 0B 0C 0C 0D 0D 10 10 11 11 12 \*\* \*\* \*\* \*\*

03 04 04 05 05 06 06 08 08 09 09 0A 0A 0B \*\* \*\*

0B 0C 0C 0D 0D 10 10 11 11 12 12 13 \*\* \*\* \*\* \*\*

04 05 05 06 06 08 08 09 09 0A 0A 0B 0B 0C \*\* \*\*

0C 0D 0D 10 10 11 11 12 12 13 13 14 \*\* \*\* \*\* \*\*

05 06 06 08 08 09 09 0A 0A 0B 0B 0C 0C 0D \*\* \*\*

0D 10 10 11 11 12 12 13 13 14 14 15 \*\* \*\* \*\* \*\*

06 08 08 09 09 0A 0A 0B 0B 0C 0C 0D 0D 10 \*\* \*\*

10 11 11 12 12 13 13 14 14 15 15 16 \*\* \*\* \*\* \*\*

\*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\*

\*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\*

08 09 09 0A 0A 0B 0B 0C 0C 0D 0D 10 10 11 \*\* \*\*

11 12 12 13 13 14 14 15 15 16 16 18 \*\* \*\* \*\* \*\*

09 0A 0A 0B 0B 0C 0C 0D 0D 10 10 11 11 12 \*\* \*\*

12 13 13 14 14 15 15 16 16 18 18 19 \*\* \*\* \*\* \*\*

0A 0B 0B 0C 0C 0D 0D 10 10 11 11 12 12 13 \*\* \*\*

13 14 14 15 15 16 16 18 18 19 19 1A \*\* \*\* \*\* \*\*

0B 0C 0C 0D 0D 10 10 11 11 12 12 13 13 14 \*\* \*\*

14 15 15 16 16 18 18 19 19 1A 1A 1B \*\* \*\* \*\* \*\*

0C 0D 0D 10 10 11 11 12 12 13 13 14 14 15 \*\* \*\*

15 16 16 18 18 19 19 1A 1A 1B 1B 1C \*\* \*\* \*\* \*\*

0D 10 10 11 11 12 12 13 13 14 14 15 15 16 \*\* \*\*

16 18 18 19 19 1A 1A 1B 1B 1C 1C 1D \*\* \*\* \*\* \*\*

\*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\*

\*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\*

\*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\*

\*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\*

## 2.2 Синтез сумматора при помощи пакета Decomposer

Вариант 1

Вариант 1 состоит из 3 сумматоров и 5 блоков. Схема приведена на рисунке 2. Результаты трассировки для различных типов микросхем приведены в таблице 1.



Рис.2. Схема сумматора. Вариант 1.

Таблица 1.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Количество использованных ресурсов ПЛИС | | |
| макроячейки  (macrocells) | конъюнкт. термы  (product terms) | входы функц. блоков  (func. block inputs) |
| xc9500xl | 23 | 160 | 92 |
| CoolRunner | 15 | 78 | 29 |

Вариант 2

Эта схема состоит из 4-х сумматоров,2-х элементов «исключающее или» и 4-х блоков. Схема приведена на рис.3. Результаты трассировки для различных типов микросхем приведены в таблице 2.



Рис.3. Схема сумматора. Вариант 2.

Таблица 2.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Количество использованных ресурсов ПЛИС | | |
| макроячейки  (macrocells) | конъюнкт. термы  (product terms) | входы функц. блоков  (func. block inputs) |
| xc9500xl | 24 | 167 | 98 |
| CoolRunner | 18 | 91 | 39 |

Вариант 3

Проанализировав данную схему (рис.4.), можно сделать вывод о том, что она является наиболее сложной схемой, в плане реализации, а также по количеству использованных ресурсов ПЛИС (таблица 3).



Рис.4. Схема сумматора. Вариант 3.

Таблица 3.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Количество использованных ресурсов ПЛИС | | |
| макроячейки  (macrocells) | конъюнкт. термы  (product terms) | входы функц. блоков  (func. block inputs) |
| xc9500xl | 26 | 241 | 73 |
| CoolRunner | 20 | 67 | 29 |

## 2.3 Описание сумматора на языке VHDL

Данный способ реализации цифровых устройств пользуется наибольшей популярностью во всем мире. Ниже приведен текст на языке VHDL, описывающий поведение десятичного сумматора с кодом 7-4-2-1, и результаты трассировки данного варианта.

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity sum\_vhdl is

Port ( I : in std\_logic\_vector(8 downto 0);

O : out std\_logic\_vector(4 downto 0));

end sum\_vhdl;

architecture Behavioral of sum\_vhdl is

signal sum: std\_logic\_vector(4 downto 0);

signal sum\_prom: std\_logic\_vector(3 downto 0);

signal A: std\_logic\_vector(3 downto 0);

signal B: std\_logic\_vector(3 downto 0);

begin

A <= I(4 downto 1) when I(4 downto 1) < "0111" else (I(4 downto 1) - "0001");

B <= I(8 downto 5) when I(8 downto 5) < "0111" else (I(8 downto 5) - "0001");

sum <= ('0'&A) + ('0'&B) + I( 0 );

sum\_prom(3 downto 0) <= sum(3 downto 0) when sum < "01101"

else (sum(3 downto 0) + "0011");

O(3 downto 0) <= sum\_prom(3 downto 0) when sum\_prom(3 downto 0) < "0111"

else (sum\_prom(3 downto 0) + "0001");

O( 4 ) <= '0' when sum < "01101" else '1';

end Behavioral;

Таблица 4.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Количество использованных ресурсов ПЛИС | | |
| макроячейки  (macrocells) | конъюнкт. термы  (product terms) | входы функц. блоков  (func. block inputs) |
| xc9500xl | 18 | 241 | 74 |
| CoolRunner | 21 | 80 | 42 |

## 2.4 Сравнительный анализ используемых ресурсов для различных вариантов реализации схемы

Таблица 5. Результаты трассировки для xc9500xl.

|  |  |  |  |
| --- | --- | --- | --- |
| xc9500xl | Количество использованных ресурсов ПЛИС | | |
| macrocells | product terms | function block inputs |
| Вариант 1 | 23 | 160 | 92 |
| Вариант 2 | 24 | 167 | 98 |
| Вариант 3 | 26 | 241 | 73 |
| Вариант 4 (VHDL) | 18 | 241 | 74 |

Таблица 6. Результаты трассировки для CoolRunner.

|  |  |  |  |
| --- | --- | --- | --- |
| CoolRunner | Количество использованных ресурсов ПЛИС | | |
| macrocells | product terms | function block inputs |
| Вариант 1 | 15 | 78 | 29 |
| Вариант 2 | 18 | 91 | 39 |
| Вариант 3 | 20 | 67 | 29 |
| Вариант 4 (VHDL) | 21 | 80 | 42 |

## 2.5 Покрытие блоков

Проведя сравнительный анализ по таблицам 5, 6, можно сделать вывод о том, что оптимальным является первый вариант разложения. Проведем покрытие блоков при помощи мультиплексоров типа 4/1.

С помощью мультиплексора типа 4/1 можно реализовать любую логическую функцию 3-х аргументов, поскольку при разложении исключаются 2 аргумента, а на входы подаются функции одного аргумента. Это могут быть либо константы «0», «1», либо функция тождества, либо функция инверсии 3-го аргумента.



Логическая последовательность первого блока:

0012 345\* 6633 700\*



Матрицы разложения имеют вид:

1)



2)



3)



Используя данные разложения, представим схему данного блока (графическая работа “Схема электрическая принципиальная блока DC1”).

Логическая последовательность второго блока:

0112 2302 1203 0013 0222 0013 13\*3 1120



Матрицы разложения имеют вид:

1)



2)



Используя данные разложения, представим схему данного блока (графическая работа “Схема электрическая принципиальная блока DC2”).

Логическая последовательность третьего блока:

0123 0124 056\* 723\*



Матрицы разложения имеют вид:

1)



2)



3)



Используя данные разложения, представим схему данного блока (графическая работа “Схема электрическая принципиальная блока DC3”).

Логическая последовательность четвертого блока:

0123 4560 2347 \*476



Матрицы разложения имеют вид:

1)



2)



3)



Используя данные разложения, представим схему данного блока (графическая работа “Схема электрическая принципиальная блока DC4”).

Логическая последовательность пятого блока:

0123 4125 2345 6457



Матрицы разложения имеют вид:

1)



2)



3)



Используя данные разложения, представим схему данного блока (графическая работа “Схема электрическая принципиальная блока DC5”).

После покрытия блокa получили наглядную экономию ресурсов ПЛИС.

Таблица 7.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Количество использованных ресурсов ПЛИС | | |
| макроячейки  (macrocells) | конъюнкт. термы  (product terms) | входы функц. блоков  (func. block inputs) |
| xc9500xl | 20 | 160 | 91 |
| CoolRunner | 15 | 77 | 29 |

# Вывод

Для нетривиальных схем методы декомпозиции дают существенный выигрыш по отношению к используемым ресурсам. Как видно из таблиц 5-6, традиционный в наше время подход описания цифровых устройств (VHDL) для сумматора в коде 7-4-2-1 при трассировке в кристалл не дает существенного выигрыша, чем схема, полученная в результате декомпозиции.

Из таблиц 5-6, видно, что для разных типов ПЛИС оптимальный результат дают разные варианты реализации. Это связано с особенностью структуры тех или иных ПЛИС и требует отдельных исследований.

Поиск оптимального решения задачи синтеза складывается из рассмотрения и анализа нескольких вариантов реализации комбинационной схемы для различных типов ПЛИС. В некоторых случаях возникает необходимость оценивать не только используемые ресурсы ПЛИС, но и задержку прохождения сигнала через схему.

Как видно из таблиц 5-6, для реализации сумматора по модулю 13 с весовыми коэффициентами 7-4-2-1 наиболее оптимальными вариантами являются схемы №1 и №2. После покрытия схемы №1 мультиплексорами типа 4/1, заметен существенный выигрыш в использовании ресурсов ПЛИС, о чем свидетельствуют результаты таблицы 7.

Список литературы

* Коновалов В.Н., Белов А.А., Коновалов И.В., Нежельский П.Н.: Автоматизированный синтез комбинационных логических схем на основе многоуровневой декомпозиции. – КФ МГТУ им. Н.Э. Баумана, – 35с.: ил.
* Голубев Ю.П.: Автоматизация проектирования преобразователей дискретной информации. – Калуга: Издательство «Гриф», 2003. -652 с.
* Грэгори, Кейт: Использование Visual C++ 6. Специальное издание.: Пер. с англ. – М.; СПб.; К.: Издательский дом «Вильямс», 2003. – 864 с.: ил.
* Соловьев В.В.: Проектирование цифровых систем на основе программируемых логических интегральных схем. – М.: Горяцая линия-Телеком, 2001. – 636 с. ил.
* Кузелин М.О., Кнышев Д.А., Зотов В.Ю.: Современные семейства ПЛИС фирмы Xilinx. Справочное пособие. – М.: Горячая линия–Телеком, 2004. – 440 с.: ил.