Москва 2024

**АННОТАЦИЯ**

Данная работа включает в себя 9 рисунков, 3 листинга. Количество страниц в работе — 14.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc190464609)

[1 ПОСТАНОВКА ЗАДАЧИ 5](#_Toc190464610)

[2 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ 6](#_Toc190464611)

[2.1 Реализация заданного варианта 6](#_Toc190464612)

[2.2 Реализация заданного варианта с измененной разрядностью 8](#_Toc190464613)

[2.3 Реализация заданного варианта с добавлением signed 12](#_Toc190464614)

[ЗАКЛЮЧЕНИЕ 13](#_Toc190464615)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 14](#_Toc190464616)

ВВЕДЕНИЕ

FPGA (Field-programmable gate array) — архитектура ПЛИС, в основе которой лежит конфигурируемая матрица логических ячеек. К основным аппаратным ресурсам FPGA семейства «Artix» седьмой серии можно отнести: CLB, BRAM, DSP.

CLB (Configurable Logic Blocks) — базовый компонент для построения конфигурируемой матрицы. Каждый блок содержит в своём составе два подблока (slice), которые являются комплементарными.

В пределах каждого slice присутствуют четыре шестивходовые таблицы поиска (LUT – Lookup Table). Каждая шестивходовая LUT (6-LUT) состоит из двух пятивходовых LUT (5-LUT). ыКаждая 5-LUT представляет собой память, на основе которой может быть реализована любая логическая функция от пяти переменных. Наличие мультиплексора 2–1 на выходе 6-LUT, который управляется в свою очередь шестым входом 6-LUT, даёт возможность реализовывать логические функции от шести переменных, задействуя 5-LUT в количестве двух штук.

В дополнение к таблицам поиска для выполнения арифметических операций в рамках CLB присутствуют цепи переноса (fast carry chain). В пределах одного блока CLB две цепи переноса расположены изолированно друг от друга, и распространяют перенос к соответствующим slice следующего блока CLB.

1 ПОСТАНОВКА ЗАДАЧИ

Цель работы: знакомство студентов с возможными аппаратными ресурсами, которые могут быть задействованы при проектировании, а также связь описания устройства при помощи языков описания аппаратуры и полученными результатами синтеза.

Задание: согласно варианту рассмотреть различные варианты реализации заданной схемы, номер варианта представлен на Рисунке 1.1.



Рисунок 1.1 — Персональный вариант

# 2 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ

## Реализация заданного варианта

При помощи языка описания аппаратуры Verilog средствами САПР Vivado реализуем заданное уравнение. Реализация представлена в Листинге 2.1:

Листинг 2.1 — Модуль main.v

|  |
| --- |
| `timescale 1ns / 1ps  module main(  input a,b,c,d,e,  output [2:0] f  );  assign f=a/b\*c\*d+e;  endmodule |

Данный код реализует деление, умножение и сложение. Размерность выходной шины следует брать на единицу больше размерности операндов, поскольку оператор сложения в Verilog HDL подразумевает наличие единицы переноса.

Соответствующая комбинационная схема будет использовать аппаратный ресурс LUT. Поскольку входы, от которых зависят оба выхода, одинаковы, то для размещения будет достаточно одного аппаратного ресурса типа 6-LUT, что показано на Рисунке 2.1.

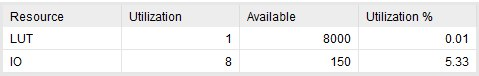


Рисунок 2.1 — Таблица задействованных ресурсов

Результат имплементации на Рисунке 2.2 показывает такое же количество задействованных LUT что и таблица с задействованными ресурсами.



Рисунок 2.2 — Задействованные ресурсы для реализации задания

## Реализация заданного варианта с измененной разрядностью

При написания данного кода, были изменены только разрядности входных и выходных сигналов, что продемонстрировано в Листинге 2.2.

Листинг 2.2 — Модуль main1.v

|  |
| --- |
| `timescale 1ns / 1ps  module main1(  input [7:0] a,b,c,d,e,  output [16:0] f  );  assign f=a/b\*c\*d+e;  endmodule |

С увеличением разрядности сумматоров растёт число задействованных аппаратных ресурсов 6-LUT для реализации. Из-за чего схема восьмиразрядного сумматора после имплементации будет использовать 139 аппаратных ресурсов 6-LUT.

С увеличением разрядности умножителя число аппаратных ресурсов, в частности LUT и Carry Chain, растёт. Уже для умножения 8-разрядного числа на 8-разрядное число потребуется большое количество аппаратных ресурсов Carry Chain и более чем в десять раз больше аппаратных ресурсов LUT.

Если основной операцией в рамках модуля является операция сложения и умножения, то для такой операции эффективнее всего также задействовать аппаратный ресурс DSP48, вследствие чего можно заметить наличие одного такого элемента на Рисунке 2.3 при реализации данного кода.

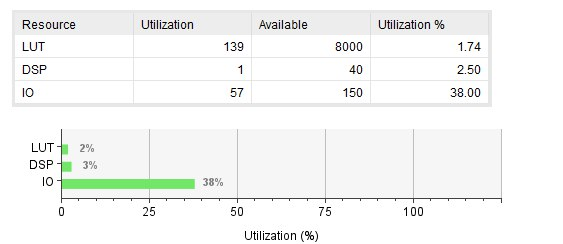


Рисунок 2.3 — Таблица ресурсов для выполнения большей разрядности

Более наглядные изображения DSP представлены на Рисунках 2.4–2.5.

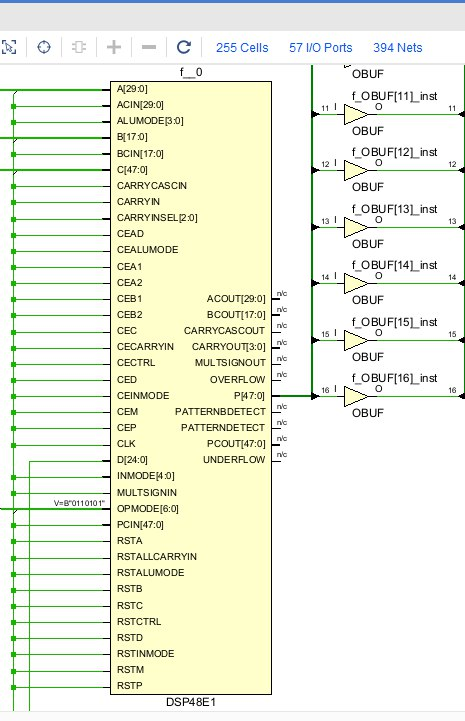


Рисунок 2.4 — Блок DSP из схематичного результата имплементации

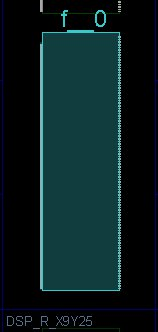


Рисунок 2.5 — Блок DSP из результата имплементации

Помимо использования DSP, из-за большего количества используемых LUT, не умещающихся в одном slice, начинают использоваться Carry chain которые помогают переносить с одного slice на другой, что можно отследить на Рисунке 2.6.

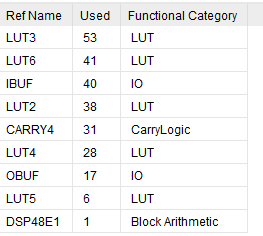


Рисунок 2.6 — Информация об элементах, задействованных для реализации модуля

На Рисунке 2.7 представлен полный результат имплементации данного модуля, на котором видно, что с увеличением разрядности таблицы растет напрямую увеличивается IO, становится больше задействованных LUT начинает использоваться одна DSP и используется несколько Carry chain для переносов на соседние slices.

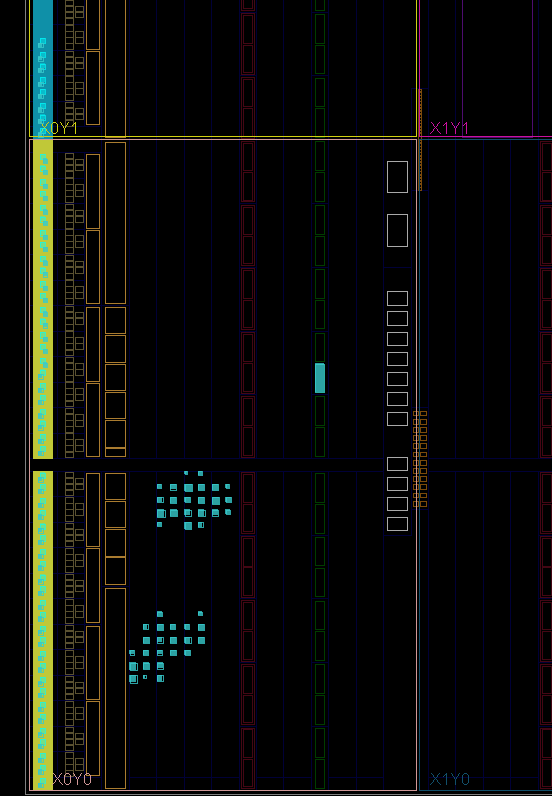


Рисунок 2.7 — Результат имплементации модуля

## Реализация заданного варианта с добавлением signed

Реализуем модуль с модификаторами «signed» для изначального решения поставленной задачи в Листинге 2.3.

Листинг 2.3 — Модуль main2.v

|  |
| --- |
| `timescale 1ns / 1ps  module main2(  input signed a,b,c,d,e,  output signed [2:0] f  );  assign f=a/b\*c\*d+e;  endmodule |

Несколько иначе будут реализовываться операции, в случае если операнды будут знаковые, т. е. изначально храниться в дополнительном коде. Для указания этого факта в языке Verilog используется ключевое слово «signed».

Исходя из вышесказанного модификатор «signed» помогает избежать ошибок при работе с отрицательными числами, однако в данном варианте отсутствуют операции вычитания, из чего можно сделать вывод что добавление данного модификатора в схему ничего не изменит, а следовательно помимо работы с дополнительным кодом добавление модификатора не изменить существенно итог имплементации, что показанно на Рисунке 2.8.

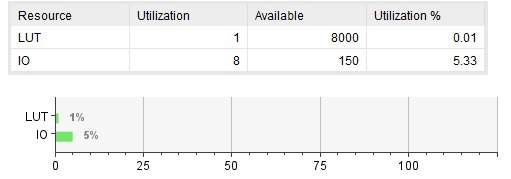


Рисунок 2.8 — Ресурсная таблица

ЗАКЛЮЧЕНИЕ

В результате выполнения практической работы были реализованы три модуля описывающие одинаковую функцию, но с разными изменениями, — при помощи Verilog HDL. Произведена имплементация полученных модулей средствами САПР Vivado. Были проанализированы результаты имплементации для каждого модуля и рассмотрены их различия, были изучены основные аппаратные ресурсы, доступные при проектировании устройств на базе ПЛИС «Artix-7» xс7a100tcsg324-1. Также получены базовые представления о реализации основных арифметических функций: описания на Verilog HDL, реализации на базе ПЛИС «Artix-7» xс7a100tcsg324-1..

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Тарасов И. Е. ПЛИС Xilinx. Языки описания аппаратуры VHDL и Verilog, САПР, приемы проектирования. М.: Издательство: Горячая линия - Телеком, 2019 г. ISBN: 978-5-9912-0802-4

2. Орлов С.А. Организация ЭВМ и систем: Учебник для вузов. 3-е изд. Стандарт третьего поколения / С.А. Орлов, Б.Я. Цилькер. – Санкт-Петербург: Питер, 2014. - 688 с. - ISBN 978-5-496-01145-7.

*3.* Паттерсон Д., Хеннесси Дж. Архитектура компьютера и проектирование компьютерных систем. 4-е изд. СПб.: Питер, 2012. – ISBN 978- 5-459-00291-1.

4. Рабан, Жан.М., Чандракасан, А., Николич, Б. Цифровые интегральные схемы. Методология проектирования. 2-е изд.: Пер. с англ. – М.: ООО «И.Д. Вильямс», 2016. – 912 с.: ил. – Паралтит. англ. ISBN 978-5-8459- 1116-2 (рус.).

5. Шафер Д., Фатрелл Р., Шафер Л. Управление программными проектами: достижение оптимального качества при минимуме затрат: Пер. с англ. – М.: Издательский дом «Вильямс», 2004. – 1136 с.: ил. – Парал.тит.англ.