**Руководство по выполнению курсовой работы**

**по основам вычислительной техники**

**СПбПУ**

Санкт-Петербург   
2020

Содержание курсовой работы

[1. Цели и задачи. Подготовка к курсовой работе. 3](#_Toc31722889)

[1.1. Цели и задачи. 3](#_Toc31722890)

[1.2. Знакомство с терминами и определениями. 3](#_Toc31722891)

[1.3 . Подготовка инструментальных средств. 3](#_Toc31722892)

[2. Исследование работы одноразрядного двоичного сумматора 5](#_Toc31722893)

[2.1. Синтез сумматора. 5](#_Toc31722894)

[2.2.Создание проектного логического файла на основе предварительно разработанной схемы 8](#_Toc31722895)

[2.3.Моделирование работы устройства 10](#_Toc31722896)

[3. Исследование суммирования и сравнения чисел без знака в формате с фиксированной запятой 12](#_Toc31722897)

[3.1. Исследование сложения чисел без знака на многоразрядном комбинационном двоичном сумматоре. 12](#_Toc31722898)

[3.2. Сравнение чисел без знака на многоразрядном комбинационном двоичном сумматоре. 17](#_Toc31722899)

[3.3. Умножение числа без знака на константу. 18](#_Toc31722900)

[4. Исследование сложения чисел в формате с фиксированной запятой в дополнительном коде на двоичном сумматоре. 20](#_Toc31722901)

[4.1. Сложение чисел со знаком в дополнительном модифицированном коде. 20](#_Toc31722902)

[4.2. Сравнение чисел со знаком в дополнительном модифицированном коде. 21](#_Toc31722903)

[4.3. Сложение чисел в дополнительном коде при использовании библиотечного модуля LPM\_ADD\_SUB, настроенного на сложение чисел со знаком. 22](#_Toc31722904)

[5. Исследование сложения чисел в формате с фиксированной запятой в обратном коде на двоичном сумматоре. 26](#_Toc31722905)

[5.1. Сложение чисел со знаком в обратном коде. 26](#_Toc31722906)

[6. Исследование сумматора чисел в формате с плавающей запятой 28](#_Toc31722907)

[Приложения 32](#_Toc31722908)

[П.1. Индивидуальные задания для исследования работы одноразрядного двоичного сумматора 33](#_Toc31722909)

[П.2. Индивидуальные задания для исследования арифметических операций с числами с фиксированной запятой. 34](#_Toc31722910)

[П.3. Индивидуальные задания для исследования сложения чисел с фиксированной запятой. 36](#_Toc31722911)

[Шаблон плана тестирования для сумматора модулей чисел 37](#_Toc31722912)

[Шаблон плана тестирования для сумматора в обратных/дополнительных кодах 38](#_Toc31722913)

[Шаблон плана тестирования для сумматора с плавающей запятой 39](#_Toc31722914)

# 1. Цели и задачи. Подготовка к курсовой работе.

## 1.1. Цели и задачи.

* Получение навыков работы с различными машинными форматами и кодами представления двоичных чисел; получение навыков по реализации сумматоров и компараторов на их основе.
* Исследование реализацию сумматора на логических элементах.
* Исследование работы двоичных сумматоров и компараторов чисел в формате с фиксированной запятой, выполняющих сложение в прямых, обратных и дополнительных кодах, реализованных с помощью библиотечного модуля.
* Исследование работы 32-х разрядного сумматора чисел в формате с плавающей запятой, реализованного с помощью библиотечного модуля.

## 1.2. Знакомство с терминами и определениями.

* 1. Записать определения основных понятий по теме «Представление чисел в ЭВМ. Сложение чисел в ЭВМ», включая следующие: двоичный сумматор, алгебраическое сложение, арифметическое сложение.
  2. Дополнительно: привести перевод основных терминов на английский язык.
  3. Коды отображения чисел: прямой, обратный, дополнительный. Способы получения обратного и дополнительного кодов числа. Представление нуля.
  4. Представление числа в формате с фиксированной запятой. Переполнение разрядной сетки.
  5. Представление числа в формате с плавающей запятой. Стандарт IEEE 754. Особые ситуации при сложении чисел в формате с плавающей запятой.

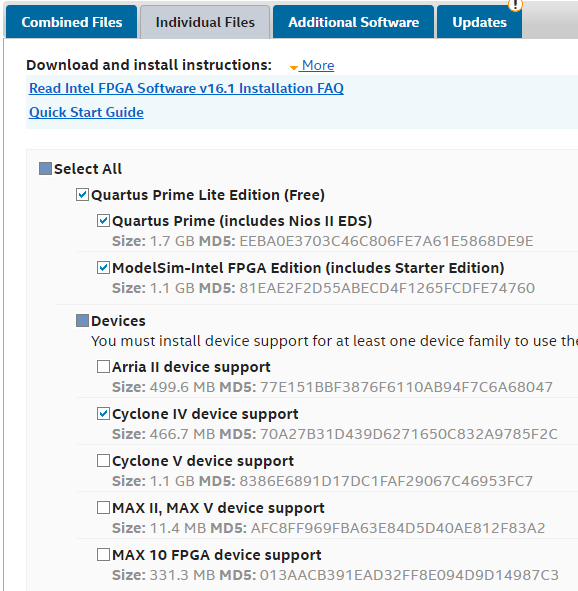
## 1.3 . Подготовка инструментальных средств.

Скачать дистрибутив программы Quartus Prime v 15.1 или выше с сайта Intel PSG. Для этого:

- перейти на страницу загрузки [Download Center](http://dl.altera.com/?edition=lite),

- выбрать предпочтительные настройки для скачивания (операционная система, способ загрузки),

- выделить файлы для скачивания, как показано ниже (в случае выбора менеджера загрузки). Будет загружен основной дистрибутив программы **QuartusLiteSetup-16….exe** (или **Quartus-lite-16…-linux.tar**), программа для моделирования и пакет поддержки ПЛИС семейств Cyclone IV **cyclone-16....qdz**.



- загрузить выбранные файлы, выполнив запрашиваемые требования.

2. Установить Quartus Prime и ModelSim, используя предлагаемые по умолчанию настройки и пути.

3. Скопировать архивы проектов **adder1**, **adder2,** **adder3** и **fpadder** в рабочую папку, путь к которой не содержит пробелов и русских символов.

4. Запустить **Quartus Prime**.

# 2. Исследование работы одноразрядного двоичного сумматора

## 2.1. Синтез сумматора.

Две логические функции, *cout* и *s*, зависят от трех переменных (cin, *a*, *b*) и заданы таблицей истинности.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cin | a | b | cout | s |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

Выполните минимизацию логических функций, ориентируясь на индивидуальное задание (см. П.1). Запишите для этих двух логических функций логические выражения в заданном в индивидуальном задании базисе.

Включите в пояснительную записку полученные выражения.

Разработайте схему, реализующую эти две логические функции в заданном базисе.

|  |  |  |  |
| --- | --- | --- | --- |
| Номер варианта | 1 | 2 | 3 |
| Базис | Шеффера | Пирса | Классический |
| Логические  элементы | 2И-НЕ | 2ИЛИ-НЕ двухвходовые | 2И, 2ИЛИ, НЕ |
| Элементы схемы |  |  |  |

Создайте и исследуйте модель устройства, реализующего разработанную схему, в среде Quartus. Последовательность шагов описана далее.

1. Создайте новый проект в среде Quartus. Для этого в меню **File** выберите **New Project Wizard…**. На экране появится окно введения **Introduction** (если оно не было отключено). Нажмите кнопку **Next**.
2. В появившемся окне **Directory, Name, Top-level Entity**(рис. 1) введите:

* расположение отдельной, новой папки проекта,

***Внимание:*** *Не используйте в путях и именах файлов русские буквы и пробелы, а также цифры в началах имен. Это позволит гарантированно избежать некоторых ошибок при работе с Quartus.*

* Имя проекта **(name of project):** schem1.

Нажмите кнопку **Next**. Подтвердите создание новой директории.

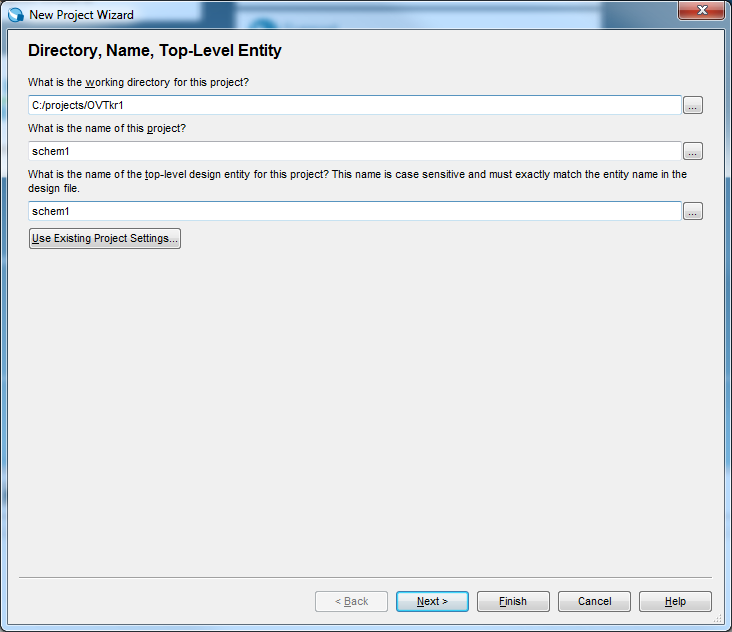


Рис. 2.1. Задание пути рабочей папки проекта, имени проекта и имени файла верхнего уровня

1. В следующем окне **Project Type** оставьте выбранную по умолчанию настройку **Empty Project** и нажмите кнопку **Next**.
2. Пропустите следующий этап, этап добавления к проекту логических файлов, нажав кнопку **Next** в окне **Add Files**.
3. В следующем окне **Family & Device** **Settings** (рис. 2.2) укажите семейство ПЛИС Cyclone IV E для реализации проекта. Выберите настройку **Auto device selected by the Fitter**. Нажмите кнопку **Next.**

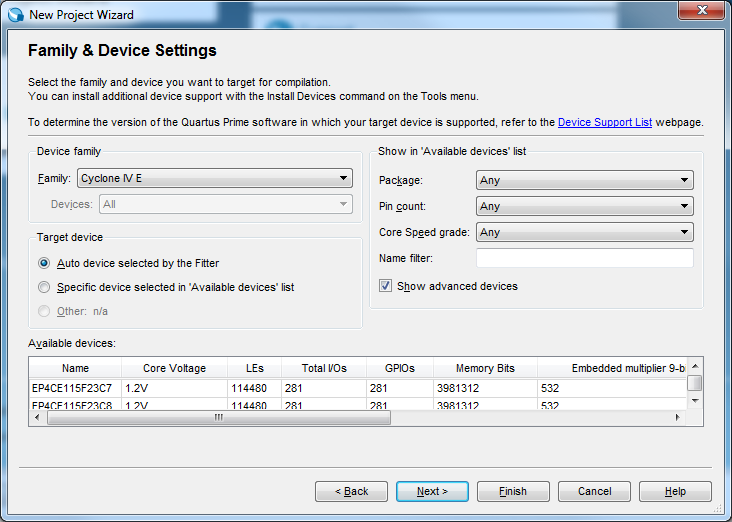


Рис. 2.2. Третий шаг создания нового проекта – выбор ПЛИС

1. Пропустите следующий этап, нажав кнопку **Next** в окне **EDA Tool Settings**.
2. В окне **Summary** (рис. 2.3) указаны установки, заданные Вами для создаваемого проекта. Проверьте их. Если все правильно, то нажмите кнопку **Finish**. В противном случае вернитесь назад, нажав (возможно несколько раз) кнопку **Back**.

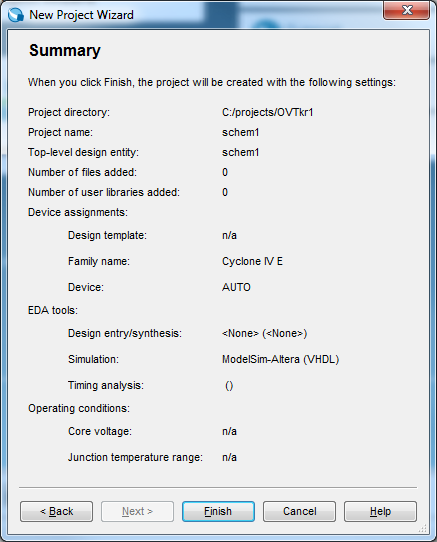


Рис. 2.3. Сводка информации при создании нового проекта

## 2.2.Создание проектного логического файла на основе предварительно разработанной схемы

1. Выполните команду: меню **File->New…** и в появившемся окне в разделе **Design Files** выберите **Block Diagram/Schematic File** – тип файла для схемного ввода**.** Нажмите **ОК.** Появится рабочее поле графического редактора для ввода схемы.
2. Сохраните созданный файл, выполнив команду меню **File->Save As…** и нажав кнопку **Сохранить**. По умолчанию файл сохраняется в папку проекта и имеет установленное ранее при создании проекта имя файла верхнего уровня. *Не используйте русские буквы и пробелы в именах файлов, не используйте цифры в качестве начального символа имени.*
3. Выполните графический ввод разработанной вами схемы. Для добавления элементов на схему воспользуйтесь интерфейсом ввода символов, который можно вызвать, дважды щелкнув левой кнопкой мыши в рабочем поле графического редактора так, чтобы появилось окно ввода символов – **Symbol** (рис. 2.4). В поле **Name** вводите названия искомых элементов («input», «output», «nand2»). Переименуйте добавленные элементы ввода/вывода так, чтобы они соответствовали переменным логических функций и именам этих логических функций. Дополните схему всеми требуемыми соединениями.

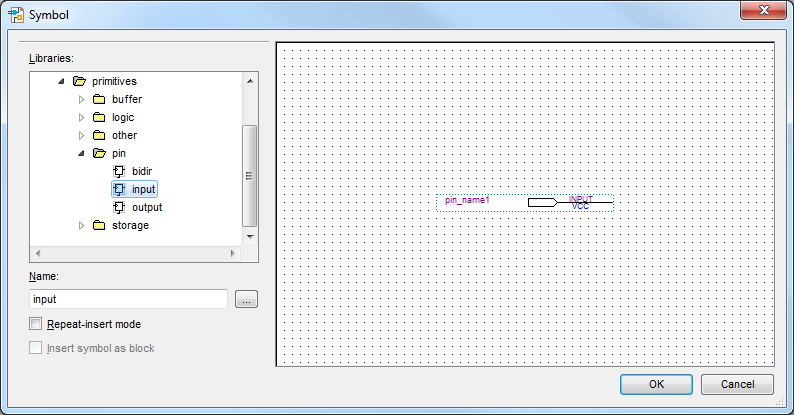
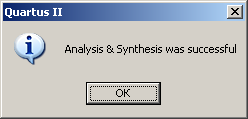


Рис. 2.4. Интерфейс выбора элементов схемы Symbol

1. Сохраните введенную схему, выбрав команду **File->Save** или нажав кнопку  на горизонтальной панели инструментов окна главного менеджера пакета.
2. Для проверки правильности введенной схемы (с точки зрения корректности использования символов и выполненных соединений) запустите процедуру анализа и синтеза, выполнив команду **Processing -> Start -> Start Analysis & Synthesis**.
3. При успешном завершении процедуры анализа и синтеза проекта появится окно с соответствующим сообщением (рис. 2.5). Если же компилятор обнаружит ошибки в схеме, то Вам следует исправить их и повторить анализ и синтез. Информация об ошибках расположена в поле Messages в нижней части окна главного менеджера пакета.

Одной из возможных причин возникновения ошибок является несоответствие схемы требованию к уникальности имен элементов схемы.

  
Рис. 2.5. Сообщение об успешном завершении анализа и синтеза проекта (по умолчанию отключено)

1. Нажмите кнопку **OK**. (В зависимости от настроек среды окно об успешном завершении может не появиться. В этом случае в строке состояния главного окна отобразиться прогресс компиляции 100 %)

Внесите снимок экрана с разработанной схемой в пояснительную записку.

## 2.3.Моделирование работы устройства

1. Создайте файл schem1.vwf с тестовыми воздействиями для анализа функциональности реализованного устройства. Для этого в меню **File** выберите **New**. В окне **New** выберите закладку **Verification/Debugging Files**. Укажите **University Program VWF** (см. рис. 2.6) и нажмите кнопку **OК**. Появится окно редактора временных диаграмм.

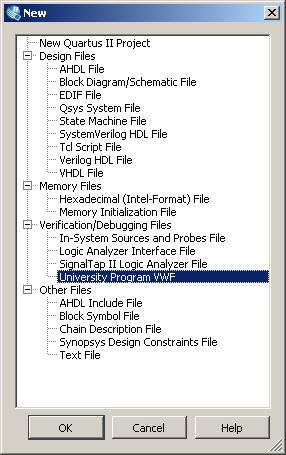


Рис. 2.6. Создание файла временных диаграмм

1. В меню **Edit** выберите **Insert->Insert Node or Bus…** (илидважды щелкните левой кнопкой в свободном поле **Name** окна редактора). В появившемся окне (рис. 2.7) нажмите кнопку **Node Finder**.

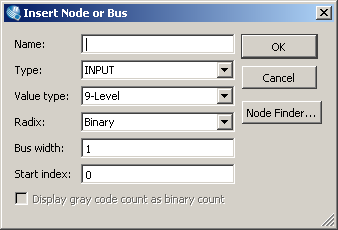


Рис. 2.7. Добавление цепей и шин на временную диаграмму

1. В окне **Node Finder** в поле **Filter** установите **Pins: all**. Нажмите кнопку **List**. В поле **Nodes Found** появятся все найденные в проекте сигналы типа ввода/вывод. Нажмите кнопку , чтобы скопировать все сигналы в раздел **Selected Nodes**.
2. Нажмите кнопку **OK**. В следующем окне еще раз нажмите кнопку **OK**.
3. Введите входные тестовые воздействия (временную диаграмму), которые позволят проверить работу логических функций на всех возможных входных наборах. Для этого выделяйте сигналы на требуемых временных интервалах и редактируйте их, используя вертикальную панель инструментов или меню окна **Edit** редактора временных диаграмм.

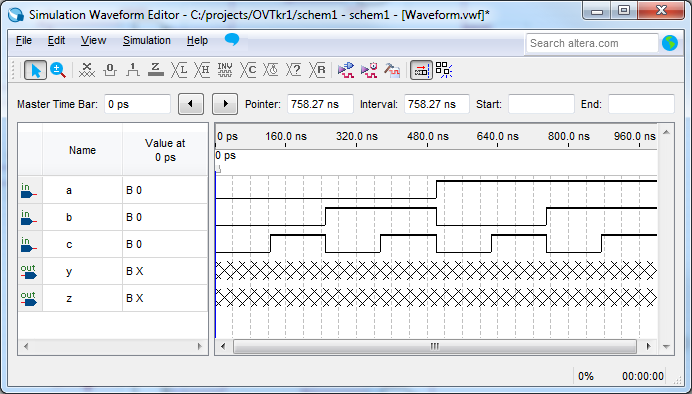


Рис. 2.8. Пример входных воздействий для моделирования работы устройства

1. Сохраните файл с введенным тестом под именем **schem1.vwf**. (Команда **File->Save As**. В появившемся окне введите имя файла **schem1.vwf**).

Запуск моделирования

1. Выполните команду **Run Functional Simulation** из меню **Simulation**.
2. Проанализируйте результаты моделирования, показанные в открывшемся окне. Сравните их со значениями из таблиц истинности заданных логических функций: они должны совпасть на всех наборах входных переменных.

В пояснительную записку включите снимок экрана с полученной временной диаграммой и результаты выполненного сравнения.

1. Завершите моделирование, закрыв окна с временными диаграммами.

Контрольное задание:

1. По указанию преподавателя постройте многоразрядный сумматор модулей чисел на базе созданного одноразрядного сумматора.

# 3. Исследование суммирования и сравнения чисел без знака в формате с фиксированной запятой

Исследование состоит в:

* создании модели заданного устройства. Средствами Quartus создается схема, которая при отсутствии ошибок ввода и выполненной компиляции определяет функциональность устройства.
* создании теста для проверки функциональности. Эта задача решается разработкой и последующей реализацией соответствующего плана тестирования (эксперимента) для выбранного объекта исследования. Совпадение результатов тестирования с ожидаемыми теоретически свидетельствует о корректности работы исследуемого устройства и о правильности наших теоретических представлений о его работе.

Для создания заданных арифметических устройств используется библиотечный модуль LPM\_ADD\_SUB (эталонная модель, безошибочно работающее устройство).

## 3.1. Исследование сложения чисел без знака на многоразрядном комбинационном двоичном сумматоре.

Создайте и исследуйте модель устройства, реализующего в среде Quartus схему восьмиразрядного комбинационного двоичного сумматора с использованием библиотечного модуля LPM\_ADD\_SUB. Схема представлена на рис.3.1.

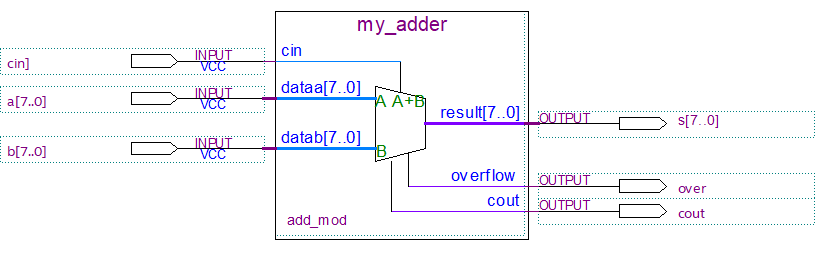
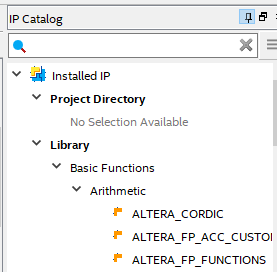


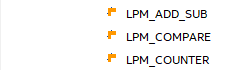
Рис.3.1.

Создайте новый проект **adder\_mod**, выполнив последовательность шагов, приведенную в П.2 "Исследование работы одноразрядного комбинационного двоичного сумматора".

Создайте логический файл по схеме рис.3.1, продолжив выполнение последовательности шагов, приведенную в разделе 2. Используйте мастер настроек (MegaWizard) для настройки библиотечного модуля LPM\_ADD\_SUB. Для этого:

* откройте IP-Catalog (например в меню Tools)и дважды щелкните LPM\_ADD\_SUB (см. рис.3.2).
* 

.............................................

Рис.3.2.

* в открывшемся окне задайте имя создаваемого модуля (см. рис.3.3) и щелкните ОК.

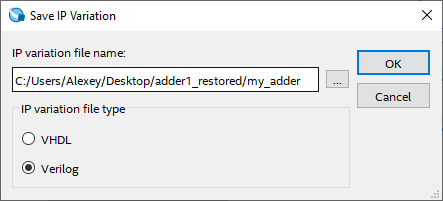
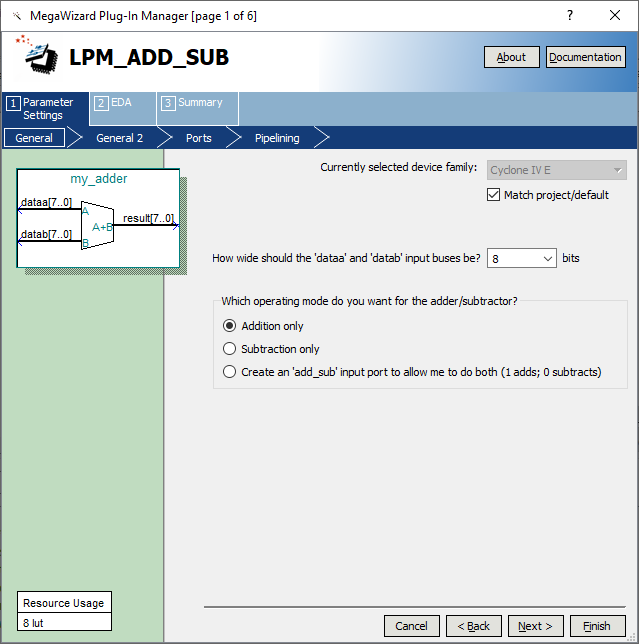
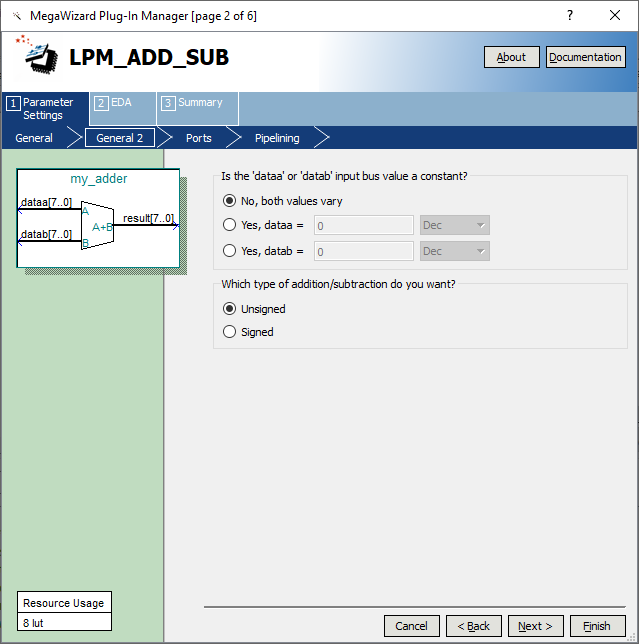


Рис.3.3.



На этой странице задается разрядность сумматора (по умолчанию 8) и режим работы (только сложение)

Рис.3.4.

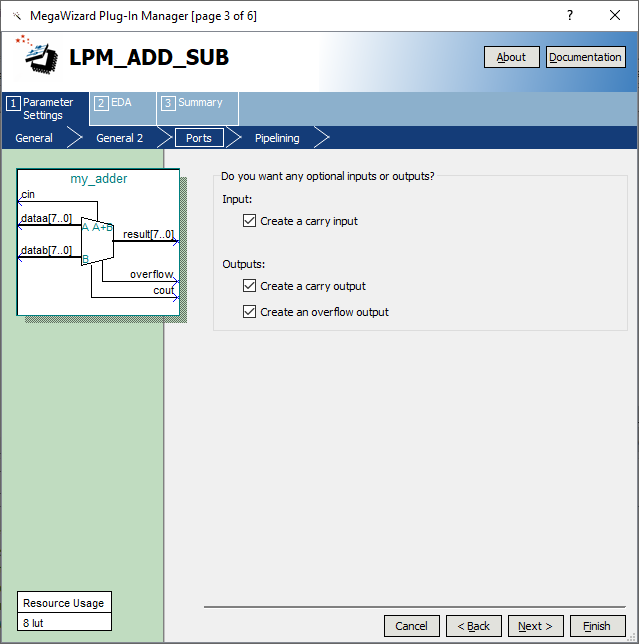


На этой странице определяются входные переменные. Тип входных переменных:

* Unsigned - числа без знака, модули;
* Signed - числа со знаком в дополнительном коде. Старший разряд - знак числа.

Задание типа входных переменных не влияет на логическую структуру сумматора, а только определяет логику формирования сигнала переполнения.

Рис.3.5



На этой странице задаются сигналы входного и выходного переносов и сигнал переполнения.

Рис.3.6

Окна 4 и 5 оставьте без изменений, а в окне 6 укажите создание Quartus Prime symbol file. Щелкните кнопку Finish.

Символ созданного сумматора записан в папке Project и может быть использован в схеме, как и любой символ из стандартных библиотек Quartus.

Закончите ввод проекта. Сохраните полученный графический файл под именем **adder\_mod.**

Скомпилируйте проект, выполнив команду **Start Analysis & Synthesis**.

**Моделирование работы многоразрядного комбинационного сумматора**

Рассмотрим ***пример тестирования*** сумматора модулей чисел.

Данные тестирования записываются в таблицу, как показано в Таблице 3.1. Данные размещены в строках, соответствующих типу теста по признакам наличия входного переноса cin и наличия выходного переноса cout. Для сумматора модулей чисел тестовые случаи включают суммирование без/с переполнением при отсутствии/наличии входного переноса. Последние столбцы будут заполнены после получения результатов тестирования.

**Таблица 3.1**

**Пример плана тестирования для сумматора модулей чисел**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| №  п/п | Тест | Входные воздействия | | | | |  | | Результат | | | |
| a[7..0] | | B[7..0] | | cin | S[7..0] | | | cout | over | Соот-  ветствует ожидаемому |
| двоичное | Десятич-ное | двоичное | Десятич-ное | Двоичное | Десятич-  ное | |
| 1 | Суммирование без переполнения | 00000101 | 5 | 00101000 | 40 | 0 |  |  | |  |  |  |
| 2 | 00101101 | 45 | 00001001 | 9 | 0 |  |  | |  |  |  |
| 3 | 00000101 | 5 | 00101000 | 40 | 1 |  |  | |  |  |  |
| 4 | 00101101 | 45 | 00001001 | 9 | 1 |  |  | |  |  |  |
| 5 | Суммирование с переполнением | 11111100 | 252 | 00101000 | 40 | 0 |  |  | |  |  |  |
| 6 | 11111111 | 255 | 00000001 | 1 | 0 |  |  | |  |  |  |
| 7 | 10000000 | 128 | 10000000 | 128 | 1 |  |  | |  |  |  |
| 8 | 10100000 | 160 | 10011010 | 154 | 1 |  |  | |  |  |  |

Выполните описанную далее процедуру тестирования для этого примера.

Скомпилируйте проект, выполнив команду **Start Analysis & Synthesis**.

По приведенному в табл.3.1 тестовому плану выполните моделирование работы сумматора (процедура создания тестового файла и запуск моделирования описаны в разделе 2).

В пояснительную записку включите снимок экрана с полученной временной диаграммой и результаты выполненного сравнения.

**Таблица 3.2**

**Пример заполнения таблицы с планом тестирования для сумматора модулей чисел**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| №  п/п | Тест | Входные воздействия | | | | |  | Результат | | | | |
| A\_iпр | | B\_iпр | | cin\_i | Y\_oпр | | | cout\_o | over | Соот-ветствует ожидаемому |
| двоичное | десятичное | двоичное | десятичное | двоичное | | десятичное |
| 1 | Суммирование без переполнения | 00000101 | 5 | 00101000 | 40 | 0 | 00101101 | | 45 |  | 0 | да |
| 2 | 00101101 | 45 | 00001001 | 9 | 0 | 00110001 | | 49 |  | 0 | да |
| 3 | 00000101 | 5 | 00101000 | 40 | 1 | 00101110 | | 46 |  | 0 | да |
| 4 | 00101101 | 45 | 00001001 | 9 | 1 | 00110111 | | 55 |  | 0 | да |
| 5 | Суммирование с переполнением | 11111100 | 252 | 00101000 | 40 | 0 | 00100100 | | 36 |  | 1 | да |
| 6 | 11111111 | 255 | 00000001 | 1 | 0 | 00000000 | | 0 |  | 1 | да |
| 7 | 10000000 | 128 | 10000000 | 128 | 1 | 00000001 | | 1 |  | 1 | да |
| 8 | 10100000 | 160 | 10011010 | 154 | 1 | 00111011 | | 59 |  | 1 | да |

Закройте окно с результатами моделирования.

Разработайте собственный план тестирования сумматора модулей чисел в соответствии с индивидуальным заданием (см. Таблицу П.2 в Приложении П.2).

Выполните редактирование теста, чтобы привести его в соответствие разработанному Вами плану. Для этого в окне редактора временных диаграмм **Simulation Waveform Editor** выделите сигналы, которые необходимо изменить, и назначьте для них новые значения, используя кнопки на панели инструментов **Waveform Editor** в верхней части окна. Например, чтобы изменить значение сигналов на входе a[7..0] на интервале от 120 нс до 200 нс, нужно выделить мышкой соответствующий участок временной диаграммы, как показано ниже:

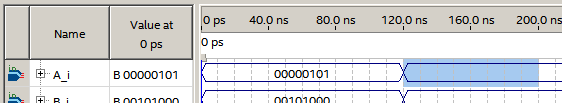


Рис. 3.7. Иллюстрация процесса редактирования временной диаграммы

через меню **Edit -> Value** выбрать команду **Arbitrary Value** и установить двоичное значение сигналов и нажать кнопку **ОК**.

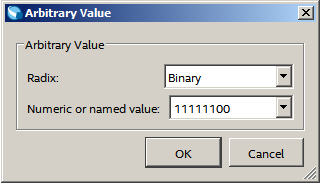


Рис. 3.8. Окно ввода произвольного двоичного значения для сигнала на временной диаграмме

Редактирование остальных сигналов на временной диаграмме выполняется аналогично.

8. Сохранив изменения, запустите функциональное моделирование и проверьте соответствие результатов моделирования ожидаемым. В том случае, если результаты моделирования не совпадают с ожидаемыми, определите и устраните причину несоответствия.

В пояснительную записку включите снимок с экрана со схемой исследуемого сумматора, полностью заполненный план тестирования и соответствующий ему снимок экрана с результатами моделирования сумматора.

Контрольное задание:

1. По указанию преподавателя модифицировать входные данные теста и объяснить полученные результаты.

**Программа исследования для сумматора модулей чисел**

1. Используя шаблон плана тестирования, в соответствии с индивидуальным заданием (см. Таблицу П.2 в Приложении П.2). выбрать и записать в таблицу двоичные значения операндов в столбцы a[7..0] и b[7..0], а также бит входного переноса cin. Выбранные значения должны соответствовать заданным тестовым случаям (например, суммирование двух положительных чисел без переполнения).

2. Полностью заполнить столбцы входных воздействий, записав десятичное представление для выбранных значений a[7..0] и b[7..0].

3. Реализовать тест в форме временной диаграммы в среде Quartus Prime, задавая входные воздействия во времени в том же порядке, что и в плане тестирования. Провести функциональное моделирование работы устройства.

4. Записать в таблицу двоичные значения s[7..0], а также cout и over, полученные в результате моделирования.

5. Дополнить результаты, записав для s[7..0]десятичные значения, равные полученным при моделировании двоичным.

6. Убедиться, что полученные результаты тестирования совпадают с ожидаемыми.

В том случае, если результаты моделирования не совпадают с ожидаемыми, определить и устранить причину несоответствия.

## 3.2. Сравнение чисел без знака на многоразрядном комбинационном двоичном сумматоре.

**Обоснование**

Арифметическое сравнение на неравенство модулей чисел выполняется на сумматоре выполнением операции вычитания и анализом знака результата. При сравнении чисел a[] ≥ b[] требуется из a[] вычесть b[] и проанализировать состояние знакового разряда. Для выполнения операции вычитания b[] следует перевести в дополнительный код. Таким образом, осуществляется сложение |a[]| и 2n - |b[]| (2n - число, до которого берется дополнение, в n - разрядном сумматоре это вес разряда cout).

Выполнение операции вычитания:

|a[]| + 2n - |b[]| = 2n +(|a[]| - |b[]|) для a[] ≥ b[] (число в скобках положительное и представлено в прямом коде, в разряде cout - единица весом 2n).

|a[]| + 2n - |b[]| = 2n +2n -(|b[]| - |а[]|)для a[] < b[] (число в скобках - модуль разности , но поскольку результат вычитания отрицательный, берется его дополнение до 2n, т.е. получаем 2n -(|b[]| - |а[]|). 2n +2n= 2n=1, значит в разряде cout с весом 2n записан ноль).

**Создание проекта**.

Создайте новый проект с именем **adder\_mod\_comp**, c такими же настройками, как у **adder\_mod**, задав имя файла верхнего уровня **adder\_mod\_comp.**

В проектную папку скопируйте файл **adder\_mod.bdf** и все файлы с именем **my\_adder** (для сравнения используется тот же сумматор что и для сложения модулей чисел).

Откройте файл , модифицируйте его в соответствии со схемой рис.3.9 и сохраните его под именем **adder\_mod\_comp.**

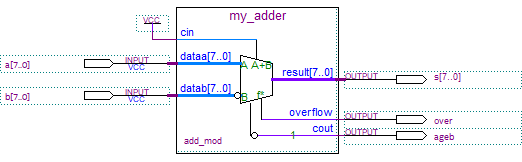


Рис.3.9

Дополнение кода b[7..0] получается инвертированием b[7..0] и подачей единицы на вход cin сумматора (единица младшего разряда).

Для инвертирования входа datab[7..0] сумматора щелкните по символу правой кнопкой во всплывшем окне выберите Properties , щелкните закладку Ports и для порта datab[7..0] в столбце Inversion поставьте All.

Выполните компиляцию проекта (анализ и синтез).

Создайте и заполните в соответствии с индивидуальным заданием (см. Таблицу П.2 в Приложении П.2). таблицу с планом тестирования компаратора.

Выполните исследование сравнения модулей чисел на двоичном сумматоре по программе исследования сумматора модулей чисел.

## 3.3. Умножение числа без знака на константу.

Обоснование.

Для умножения числа без знака на константу **с** используется сложение частичных произведений числа **a[]**х**ci** и, где **ci** - единицы в соответствующих разрядах константы. Получение частичного произведения сводится к получению сдвинутого значения числа **a[].** Если количество единиц в константе **с** не более двух, то для получения произведения достаточно не более одного двоичного сумматора.

**Например:** a[4..0] ; c =101.

Максимальное значение произведения s[] = 11111 x 101 = 31x5 = 155, т.е. разрядность s равна 8.

s[7..0] = a[4..0] x 100 + a[4..0] x 1 = a[4..0]00 + a[4..0]. Выровняв разрядность слагаемых получим:

a4 a3 a2 a1 a0 0 0

+

0 0 a4 a3 a2 a1 a0

Два младших разряда числа s[1..0] равны a[1..0] (складываются с нулями). Для получения разрядов s[7..2] необходим шестиразрядный сумматор (пятиразрядный с выходным переносом). Схема реализации умножения на константу для рассмотренного примера приведена на рис.3.10.

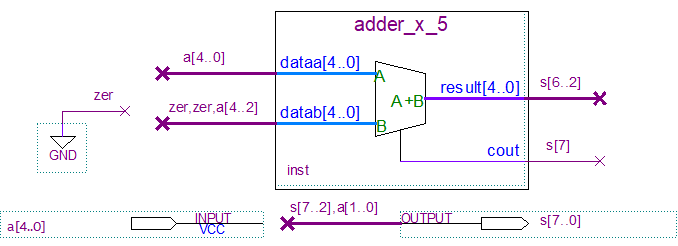


Рис.3.10.

Обратите внимание на именование цепей. В схеме получен сигнал **zer**, подключенный к "земле" и используемый как логический 0. Пятиразрядный сигнал, подаваемый на вход datab[4..0] сумматора содержит в двух старших разрядах нули и три разряда a[4..2]. Результирующий сигнал s[7..0] собирается из сигналов cout и result[4..0] сумматора и двух младших разрядов a[1..0].

**Программа исследования устройства умножения числа без знака на константу.**

1. Определите способ реализации устройства умножения на константу в соответствии с индивидуальным заданием (см. Таблицу П.3 в Приложении П.2)., обеспечивающим минимальную разрядность сумматора.

2. Используя мастер настроек (MegaWizard), настройте библиотечный модуль LPM\_ADD\_SUB для реализации устройства умножения. Создайте проект и схему устройства умножения в соответствии с индивидуальным заданием.

3. Создайте таблицу с планом тестирования. Полностью заполните столбец входных воздействий, записав двоичное и десятичное представление для выбранных значений a[].

4. Реализуйте тест в форме временной диаграммы в среде Quartus Prime, задавая входные воздействия во времени в том же порядке, что и в плане тестирования. Проведите функциональное моделирование работы устройства.

5. Запишите в таблицу двоичные и десятичные значения s[], полученные в результате моделирования.

6. Убедитесь, что полученные результаты тестирования совпадают с ожидаемыми.

В пояснительную записку включите снимок с экрана со схемой исследуемого устройства, полностью заполненный план тестирования и соответствующий ему снимок экрана с результатами моделирования сумматора.

В том случае, если результаты моделирования не совпадают с ожидаемыми, определите и устраните причину несоответствия.

# 4. Исследование сложения чисел в формате с фиксированной запятой в дополнительном коде на двоичном сумматоре.

Сложение чисел в формате с фиксированной запятой в дополнительном коде реализуется на том же двоичном сумматоре, на котором осуществляется сложение чисел без знака. В самом двоичном сумматоре не требуется каких-либо структурных изменений. Иначе только реализуется сигнал переполнения разрядной сетки.

## 4.1. Сложение чисел со знаком в дополнительном модифицированном коде.

Для исследования используйте модифицированный проект **adder\_mod**, созданный ранее в разделе 3.1.

Дополните схему сумматора схемой формирования сигнала переполнения (см. рис.4.1).

*(Для двоичного комбинационного сумматора чисел без знака сигнал overflow в библиотечном модуле LPM\_ADD\_SUB формируется как overflow = cout. Поскольку на этом сумматоре будет выполняться сложение чисел в дополнительных кодах, сигнал переполнения разрядной сетки следует формировать из наблюдаемых сигналов сумматора. На схеме рис.4.1 это Block Overflow).*

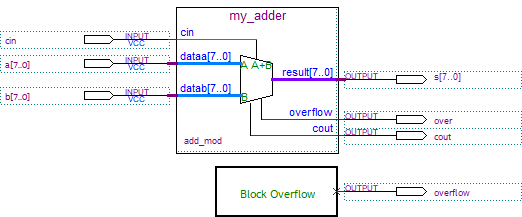


Рис.4.1.

Схема формирования сигнала переполнения **overflow** реализуется самостоятельно в соответствии с индивидуальным заданием(см. Приложение П.2, варианты представления чисел в дополнительном коде).

**Программа исследования устройства сложения чисел в дополнительных кодах.**

1.Откройте проект и проектный файл **adder\_mod.** Доработайте схему сумматора введя соответствующую индивидуальному заданию комбинационную схему выявления переполнения (см. рис.4.1). Выполните анализ и синтез (создается функциональная модель устройства).

2. Создайте таблицу с планом тестирования в соответствии с индивидуальным заданием (см. Таблицу П.2 в Приложении П.2). Полностью заполните столбец входных воздействий, записав двоичное и десятичное представление для выбранных значений a[] и b[].

3. Реализуйте тест в форме временной диаграммы в среде Quartus Prime, задавая входные воздействия во времени в том же порядке, что и в плане тестирования. Проведите функциональное моделирование работы устройства.

4. Запишите в таблицу двоичные и десятичные значения s[] и всех других выходных сигналов, полученные в результате моделирования.

5. Убедитесь, что полученные результаты тестирования совпадают с ожидаемыми.

В пояснительную записку включите снимок с экрана со схемой исследуемого сумматора, полностью заполненный план тестирования и соответствующий ему снимок экрана с результатами моделирования сумматора.

В том случае, если результаты моделирования не совпадают с ожидаемыми, определите и устраните причину несоответствия.

## 4.2. Сравнение чисел со знаком в дополнительном модифицированном коде.

Обоснование .

Для организации сравнения представленных в дополнительном коде чисел со знаком a[] и b[] следует рассмотреть возможные соотношения этих чисел.

* Знаки a[] и b[] различны. В этом случае, если знак b[] отрицательный, то a[] ≥ b[];
* Знаки a[] и b[] одинаковы, то следует сравнивать модули чисел, а для этого потребуется сумматор. Следует рассмотреть 2 варианта:

1. a[] ≥ 0 и b[] ≥ 0, числа положительные. Выполняется операция вычитания s[] = a[] - b[] и анализируется знак s[]. Приведем обоснование для правильных дробей:

s[] = 0 + |a| +1 +1 -|b| = 10 + (|a| - |b|), если |a| ≥ |b|, т.е. a[] ≥ b[], в разряде знака s[] - 0.

s[] = 0 + |a| +1 +1 -|b| = 1 + 1 - (|b| - |a|), если |a| < |b||, т.е. a[] < b[], в разряде знака s[] - 1.

2. a[] < 0 и b[] < 0, числа отрицательные. Выполняется операция вычитания s[] = a[] - b[] и анализируется знак s[]:

s[] = 1 + 1 - |a| + 0 + |b| = 10 + (|b| - |a|), если |b| ≥ |a|, т.е. a[] ≥ b[], в разряде знака s[] - 0.

s[] = 1 + 1 - |a| + 0 + |b| = 1 + 1 - (|a| - |b|), если |a| ≥ |b|, т.е. a[] < b[], в разряде знака s[] - 1.

Логическое выражение для выхода компаратора определяется как:

ageb = (a n-1 ˄ b n-1) & b n-1) | ~(a n-1 ˄ b n-1) & ~s n-1), где:

a n-1 , b n-1 и s n-1 - старшие разряды (знаковые) сравниваемых чисел и суммы;

˄, &, |, ~ - побитовые операторы языка Verilog (соответственно XOR, AND, OR, NOT).

Схема арифметического компаратора на неравенство для чисел в дополнительных кодах приведена на рис.4.2.

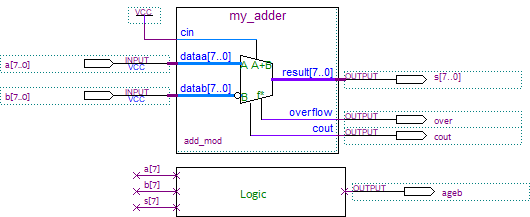


Рис.4.2.

Сигналы s[7..0], over и cout носят вспомогательный характер и для работы компаратора не требуются.

**Программа исследования компаратора чисел в дополнительных кодах.**

1.Откройте проект и проектный файл **adder\_mod.** Доработайте схему сумматора введя логическую схему формирования сигнала сравнения **ageb** (блок Logic на рис.4.2). Выполните анализ и синтез (создается функциональная модель устройства).

3. Создайте таблицу с планом тестирования. Полностью заполните столбец входных воздействий, записав двоичное и десятичное представление для выбранных значений a[] и b[]. Сигналы s[7..0], over и cout носят вспомогательный характер и могут понадобиться для объяснения работы устройства.

4. Реализуйте тест в форме временной диаграммы в среде Quartus Prime, задавая входные воздействия во времени в том же порядке, что и в плане тестирования. Проведите функциональное моделирование работы устройства.

5. Запишите в таблицу двоичные и десятичные значения s[] и всех других выходных сигналов, полученные в результате моделирования.

6. Убедитесь, что полученные результаты тестирования совпадают с ожидаемыми.

В пояснительную записку включите снимок с экрана со схемой исследуемого устройства, полностью заполненный план тестирования и соответствующий ему снимок экрана с результатами моделирования.

В том случае, если результаты моделирования не совпадают с ожидаемыми, определите и устраните причину несоответствия.

## 4.3. Сложение чисел в дополнительном коде при использовании библиотечного модуля LPM\_ADD\_SUB, настроенного на сложение чисел со знаком.

Создайте и исследуйте модель устройства, реализующего в среде Quartus схему восьмиразрядного комбинационного двоичного сумматора чисел со знаком с использованием библиотечного модуля LPM\_ADD\_SUB. Схема представлена на рис.4.3.

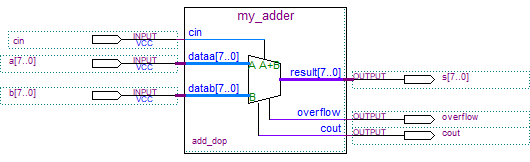


Рис.4.3.

Создайте новый проект **adder\_signed**, выполнив последовательность шагов, приведенную в П.2 "Исследование работы одноразрядного комбинационного двоичного сумматора".

Создайте логический файл по схеме рис.4.3, продолжив выполнение последовательности шагов, приведенную в разделе 2. Используйте мастер настроек (MegaWizard) для настройки библиотечного модуля LPM\_ADD\_SUB, как это делалось в П.3.1. Все настройки выполняются такими же, как в разделе 3.1, кроме стр.2 (см. рис.4.4). Здесь задайте тип входных переменных Signed.

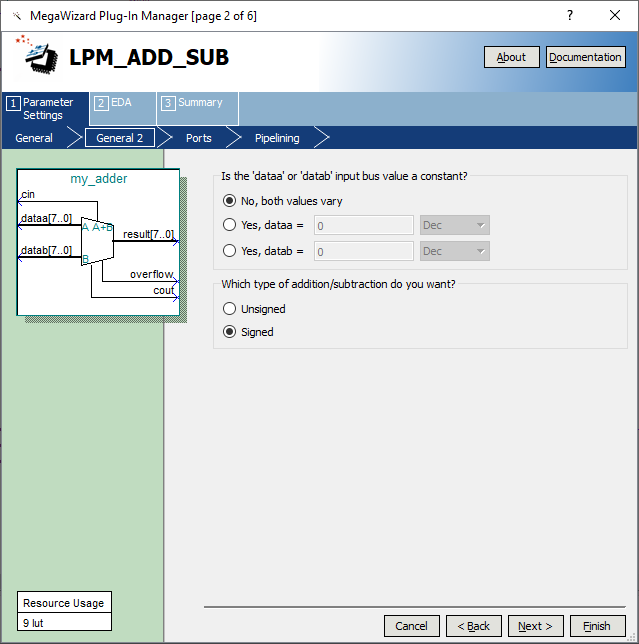


Рис.4.4

*На этой странице определяются типы входных переменных:*

* *Unsigned - числа без знака, модули;*
* *Signed - числа со знаком в дополнительном коде. Старший разряд - знак числа. Задание типа входных переменных не влияет на логическую структуру сумматора, а только определяет логику формирования сигнала переполнения.*

**Программа исследования устройства сложения чисел в дополнительных кодах.**

1. Создайте новый проект **adder\_signed** c логическим файлом рис.4.3. Выполните анализ и синтез (создается функциональная модель устройства).

2. Создайте таблицу с планом тестирования в соответствии с индивидуальным заданием (см. Таблицу П.2 в Приложении П.2). Полностью заполните столбец входных воздействий, записав двоичное и десятичное представление для выбранных значений a[] и b[].

3. Реализуйте тест в форме временной диаграммы в среде Quartus Prime, задавая входные воздействия во времени в том же порядке, что и в плане тестирования. Проведите функциональное моделирование работы устройства.

4. Запишите в таблицу двоичные и десятичные значения s[] и всех других выходных сигналов, полученные в результате моделирования. Пример заполнения таблицы с планом тестирования для сумматора в дополнительных кодах приведен в таблице 4.1.

5. Убедитесь, что полученные результаты тестирования совпадают с ожидаемыми.

В пояснительную записку включите снимок с экрана со схемой исследуемого сумматора, полностью заполненный план тестирования и соответствующий ему снимок экрана с результатами моделирования сумматора.

В том случае, если результаты моделирования не совпадают с ожидаемыми, определите и устраните причину несоответствия.

Контрольные задания:

1. По указанию преподавателя модифицировать входные данные теста и объяснить полученные результаты.

2. Объяснить значения, полученные в столбце cout. Почему они отличаются от значений overflow?

3. Пояснить величину веса бита переноса cin.

**Таблица 4.1**

**Пример заполнения таблицы с планом тестирования для сумматора в дополнительных кодах**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| №  п/п | Тест | | Входные воздействия | | | | Результат | | | | |
| A[7..0] | | B[7..0] | | S[7..0] | | overflow | cout | Соответствует ожидаемому |
| двоичное | десятичное | двоичное | десятичное | двоичное | десятичное |
| 1 | A > 0,  B > 0 | без переполнения | 0 0000 101 | 0,625 | 0 0101 000 | 5,0 | 0 0101 101 | 5,625 | 0 | 0 | да: 0,625 + 5,0 = 5,625 |
| 2 | 0 0101 101 | 5,625 | 0 0001 001 | 1,125 | 0 0110 110 | 6,75 | 0 | 0 | да: 5,625 + 1,125 =  = 6,75 |
| 3 | с переполнением | 0 1000 101 | 8,625 | 0 1001 011 | 9,375 | 1 0010 000 | -14,0 | 1 | 0 | да, из-за переполнения получен результат с противоположным знаком |
| 4 | 0 1001 111 | 9,875 | 0 1010 100 | 10,5 | 1 0100 011 | -11,625 | 1 | 0 | да, из-за переполнения получен результат с противоположным знаком |
| 5 | A < 0,  B < 0 | без переполнения | 1 1000 101 | -7,375 | 1 1001 011 | -6,625 | 1 0010 000 | -14,0 | 0 | 1 | да: -7,375 -6,625 = -14 |
| 6 | 1 1001 111 | -6,125 | 1 1010 100 | -5,5 | 1 0100 011 | -11,625 | 0 | 1 | да: -6,125 -5,5 =  = -11,625 |
| 7 | с переполнением | 1 0000 101 | -15,375 | 1 0101 000 | -11,0 | 0 0101 101 | 5,625 | 1 | 1 | да, из-за переполнения получен результат с противоположным знаком |
| 8 | 1 0101 101 | -10,375 | 1 0001 001 | -14,875 | 0 0110 110 | 6,75 | 1 | 1 | да, из-за переполнения получен результат с противоположным знаком |
| 9 | A > 0,  B < 0 | положительный результат | 0 1001 011 | 9,375 | 1 1000 101 | -7,375 | 0 0010 000 | 2,0 | 0 | 1 | да: 9,375 -7,375 = 2 |
| 10 | 0 1010 100 | 10,5 | 1 1001 111 | -6,125 | 0 0100 011 | 4,375 | 0 | 1 | да: 10,5 – 6,125 = 4,375 |
| 11 | отрицательный результат | 0 0101 000 | 5,0 | 1 0000 101 | -15,375 | 1 0101 101 | -10,375 | 0 | 0 | да: 5,0-15,375 = -10,375 |
| 12 | 0 0001 001 | 1,125 | 1 0101 101 | -10,375 | 1 0110 110 | -9,25 | 0 | 0 | да: 1,125-10,375 =  = -9,25 |

Для интерпретации двоичных значений используется следующий формат чисел: 1 знаковый разряд, 3 разряда дробной части

# 5. Исследование сложения чисел в формате с фиксированной запятой в обратном коде на двоичном сумматоре.

## 5.1. Сложение чисел со знаком в обратном коде.

Сумматор в обратных кодах, как и сумматор в дополнительных кодах, оперирует знаковыми числами. Однако для организации такого сумматора в схему двоичного сумматора необходимо ввести структурные изменения - "циклический перенос", соединив выход cout сумматора со входом cin, как показано на рис.5.1. Наличие этой обратной связи не позволяет в общем случае суммировать на таком сумматоре числа без знака и числа в дополнительном коде.

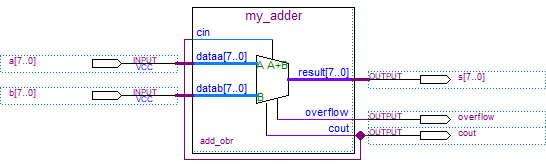


Рис. 5.1. Схема сумматора чисел в обратных кодах

**Программа исследования устройства сложения чисел в обратных кодах.**

1. Создайте новый проект **adder\_signed\_obr.** Создайте логический файл по схеме рис.5.1. Используйте мастер настроек (MegaWizard) для настройки библиотечного модуля LPM\_ADD\_SUB c логическим файлом рис.5.1. Настройки такие же, как в разделе 4.1. Выполните полную компиляцию (создается временная модель устройства). На предупреждения не обращайте внимания.

2. Создайте таблицу с планом тестирования в соответствии с индивидуальным заданием (см. Таблицу П.2 в Приложении П.2). Начните тест со сложения двух одинаковых по модулю чисел с разными знаками. Полностью заполните столбец входных воздействий, записав двоичное и десятичное представление для выбранных значений a[] и b[]. Продемонстрируйте получение положительного и отрицательного нуля.

3. Реализуйте тест в форме временной диаграммы в среде Quartus Prime, задавая входные воздействия во времени в том же порядке, что и в плане тестирования. Проведите функциональное моделирование работы устройства.

4. Запишите в таблицу двоичные и десятичные значения s[] и всех других выходных сигналов, полученные в результате моделирования. Пример заполнения таблицы с планом тестирования для сумматора в дополнительных кодах приведен в таблице 4.1.

5. Убедитесь, что полученные результаты тестирования совпадают с ожидаемыми.

В пояснительную записку включите снимок с экрана со схемой исследуемого сумматора, полностью заполненный план тестирования и соответствующий ему снимок экрана с результатами моделирования сумматора.

В том случае, если результаты моделирования не совпадают с ожидаемыми, определите и устраните причину несоответствия.

Контрольные задания:

1. По указанию преподавателя модифицировать входные данные теста и объяснить полученные результаты.

2. Объяснить возможность использования сигнала overflow, получаемого c выхода сумматора.

**Таблица 2.6**

# 6. Исследование сумматора чисел в формате с плавающей запятой

Сумматор чисел в формате с плавающей запятой с одинарной точностью (32 разряда) использует стандарт IEEE 754 и реализован в проекте **fp\_adder**. Задержка получения результата суммирования составляет 7 тактов, однако входные данные можно менять каждый такт частоты clk. Помимо выводов данных A\_i, B\_i и Y\_o, которые используются для задания значений операндов и получения результата, сумматор имеет дополнительные выводы, сигналы на которых отражают особые ситуации. Перечень особых ситуаций соответствует стандарту IEEE 754. Описание функций и выводов устройства приведено [в документации (раздел 5)](https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/ug/ug_altfp_mfug.pdf).

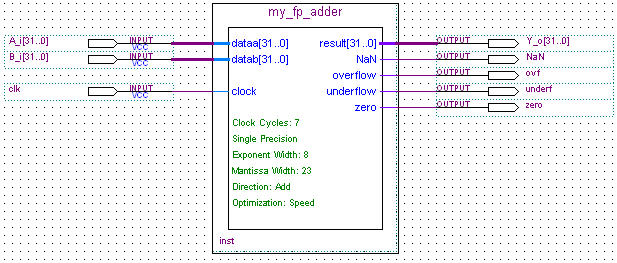
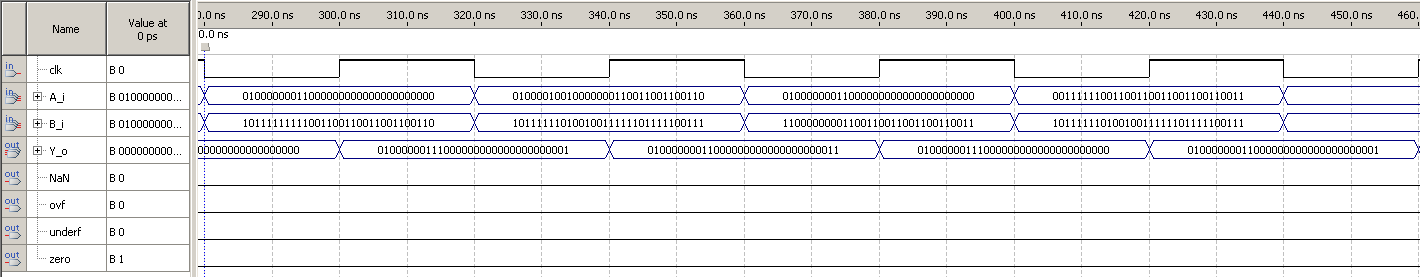


Рис. 6.1. Схема сумматора чисел в формате с плавающей запятой

Пример с фрагментом плана тестирования и временная диаграмма, иллюстрирующая его реализацию, представлены в Таблице 6.1 и на рисунке 6.2 ниже.



**Рис. 6.2. Временная диаграмма изменения входных данных для тестовых случаев 9-12**

В предлагаемом для исследования проекте **fp\_adder** используется библиотечный модуль ALTFP\_ADD\_SUB. Он настроен так, что результат на выходе сумматора появится только через 7 периодов тактового сигнала clk. Поэтому наблюдаемые на временной диаграмме выходные сигналы не отображают результаты складываемых операндов.

Для выполнения задания разархивируйтефайл **fp\_adder** в рабочей папке. Затем, в соответствии с индивидуальным заданием из Таблицы П.4, разработайте план тестирования сумматора с плавающей запятой и адаптируйте программу исследования.

 В пояснительную записку включите снимок с экрана со схемой исследуемого сумматора и полностью заполненный план тестирования. Обратите внимание на необходимость структурирования двоичных данных, представленных в таблице: в 32-хразрядном значении должны быть выделены разряды знака, порядка и мантиссы .

 В пояснительную записку включите снимки экрана с результатами моделирования сумматора. Для каждой временной диаграммы укажите соответствующие номера тестовых случаев для входных сигналов и номера тестовых случаев для выходных сигналов. Убедитесь, что на временных диаграммах показаны результаты моделирования для всех тестов.

 Включите в пояснительную записку объяснение перевода десятичного числа в двоичное в формате с плавающей запятой IEEE 754 и обратно для одного из чисел, использованных в ходе исследования сумматора,.

 Контрольные задания:

1. По указанию преподавателя модифицировать входные данные теста и объяснить полученные результаты.

2. На примере одного из тестов в плане тестирования объяснить закономерности в изменении величин мантиссы и порядка при выполнении суммирования.

**Таблица 6.1**

**Пример фрагмента плана тестирования для сумматора чисел в формате с плавающей запятой**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| №  п/п | Тест | | Входные воздействия | | | | Результат | | | | | |
| A\_ifp | | B\_ifp | | Y\_ofp | | NaN | overflow | underflow | zero |
| двоичное | десятичное  знаковое | двоичное | десятичное  знаковое | двоичное | десятичное  знаковое |
| 9 | A > 0,  B < 0 | положительный  десятичный  порядок результата | 0 10000000  11000000000000000000000 | 3,5 | 1 01111111  11001100110011001100110 | -1,8 | 0 01111111  10110011001100110011010 | 1,7 | 0 | 0 | 0 | 0 |
| 10 | 0 10000100  10000000110011001100110 | 48,1 | 1 01111110  10010011111101111100111 | -0,789 | 0 10000100  01111010011111001110110 | 47.310997 | 0 | 0 | 0 | 0 |
| 11 | отрицательный  десятичный  порядок результата | 0 10000000  11000000000000000000000 | 3,5 | 1 10000000  01100110011001100110011 | -2,8 | 0 01111110  01100110011001100110100 | 0.70000005 | 0 | 0 | 0 | 0 |
| 12 | 0 01111110  01100110011001100110011 | 0,7 | 1 01111110  10010011111101111100111 | -0,789 | 1 01111011  01101100100010110100000 | -0.0889999 | 0 | 0 | 0 | 0 |

6. Список использованных источников

1. ГОСТ 7.32-2001. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления

2. Зыков А.Г., Поляков В.И. Арифметические основы ЭВМ. – Санкт-Петербург: СПб: Университет ИТМО, 2016. – 140 с.

3. Камкин А.C., Чупилко М.M. Тестирование модулей арифметики с плавающей точкой микропроцессоров на соответствие стандарту IEEE 754. [Труды Института системного программирования РАН, 2008.](http://hardware.ispras.ru/files/publications/2008-fpu-testing.pdf)

4. Hennessy, Patterson: Computer Architecture: A Quantitative Approach, 5th Edition. – Morgan Kaufmann. 2011. – Appendix J: Computer Arithmetic by David Goldberg.

5. Goldberg D. What every computer scientist should know about floating-point arithmetic // ACM Computing Surveys. 1991. – Т. 23. – № 1– С. 5–48.

6. Floating-Point IP Cores User Guide. ALTFP\_ADD\_SUB IP Core [Электронный ресурс] // Intel FPGA. URL: https://www.altera.com/content/dam/altera-www/global/en\_US/pdfs/literature/  
ug/ug\_altfp\_mfug.pdf (дата обращения: 04.05.2018).

Заключение по полученным результатам исследования.

Заключение, выводы по работе, должны отражать цель работы и степень ее достижения, решаемые для достижения цели задачи и полноту их решения.

Дается краткая характеристика выполненных разработок и исследований (что и как сделано, с какими параметрами, какие получены результаты).

Описание и объяснение закономерностей, которые были выявлены в ходе исследования.

Описание проблем, возникших в ходе работы (по существу задания), и выбранные способы их решения.

Использовать слова: разработано, предложено, исследовано, выполнено, получено, доказано, подтверждено и т.д.

# Приложения

## П.1. Индивидуальные задания для исследования работы одноразрядного двоичного сумматора

**Таблица П.1**

|  |  |  |  |
| --- | --- | --- | --- |
| №п/п | Базис | №п/п | Базис |
| 1 | Шеффера | 17 | Пирса |
| 2 | Пирса | 18 | Классический |
| 3 | Классический | 19 | Шеффера |
| 4 | Шеффера | 20 | Пирса |
| 5 | Пирса | 21 | Классический |
| 6 | Классический | 22 | Шеффера |
| 7 | Шеффера | 23 | Пирса |
| 8 | Пирса | 24 | Классический |
| 9 | Классический | 25 | Шеффера |
| 10 | Шеффера | 26 | Пирса |
| 11 | Пирса | 27 | Классический |
| 12 | Классический | 28 | Шеффера |
| 13 | Шеффера | 29 | Пирса |
| 14 | Пирса | 30 | Классический |
| 15 | Классический | 31 | Шеффера |
| 16 | Шеффера | 32 | Пирса |

## П.2. Индивидуальные задания для исследования арифметических операций с числами с фиксированной запятой.

**Таблица П.2**

**Варианты заданий чисел для сложения и сравнения.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| №п/п | Ограничение  на значения a[] | Ограничение  на значения b[] | №п/п | Ограничение  на значения a[] | Ограничение  на значения b[] |
| 1 | xx00x001 | xxx0xxx1 | 17 | xx0xxx1,x | xx1x000,1 |
| 2 | xхx0x01,0 | xxx0xxx,1 | 18 | xx0xxx,1x | xхx0x0,10 |
| 3 | xх0xx0,11 | xxx0xx,x1 | 19 | xx0xxx1x | xх0x0x11 |
| 4 | xх01xx00 | xxx0xxx1 | 20 | xx0xxx1,x | xх01x0x,0 |
| 5 | xх01x0x,1 | xxx0xxx,0 | 21 | xx0xxx,0x | xх010x,1x |
| 6 | xх01x1,0x | xxx0xx,x0 | 22 | xx0xxx0x | xx10110x |
| 7 | xx00x111 | xxx0xxx0 | 23 | xx0xxx0,x | xxx1011,1 |
| 8 | xхx1x00,0 | xxx0xxx,0 | 24 | xx0xxx,0x | xхxx10,00 |
| 9 | xх1xx0,01 | xxx1xx,x0 | 25 | xx1xxx0x | xх1xx001 |
| 10 | xх10xx10 | xxx1xxx0 | 26 | xx1xxx0,x | xх10xx1,0 |
| 11 | xх10x1x,1 | xxx1xxx,0 | 27 | xx1xxx,0x | xх101x,x1 |
| 12 | xх11x0,0x | xxx1xx,x0 | 28 | xx1xxx0x | xх1100xx |
| 13 | xx01x101 | xxx1xxx1 | 29 | xx1xxx1,x | xx1x110,1 |
| 14 | xхx1x11,0 | xxx1xxx,1 | 30 | xx1xxx,1x | xхx1x1,10 |
| 15 | xх1xx1,11 | xxx1xx,x1 | 31 | xx1xxx1x | xх1x1x11 |
| 16 | xх00xx00 | xxx1xxx1 | 32 | xx1xxx1,x | xх00x0x,0 |

**Варианты представления чисел в дополнительном коде:**

* четные номера - дополнительный модифицированный код;
* нечетные номера - представление знака в одном разряде.

**Варианты заданий для умножения на константу.**

**Таблица П.3**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| №п/п | Разрядность  a[] | Значение  константы | №п/п | Разрядность  a[] | Значение  константы |
| 1 | 8 | 9 | 17 | 8 | 17 |
| 2 | 7 | 12 | 18 | 7 | 18 |
| 3 | 6 | 17 | 19 | 6 | 20 |
| 4 | 5 | 18 | 20 | 5 | 24 |
| 5 | 4 | 20 | 21 | 4 | 10 |
| 6 | 8 | 24 | 22 | 8 | 9 |
| 7 | 7 | 10 | 23 | 7 | 12 |
| 8 | 6 | 9 | 24 | 6 | 17 |
| 9 | 5 | 12 | 25 | 5 | 18 |
| 10 | 4 | 17 | 26 | 4 | 20 |
| 11 | 8 | 18 | 27 | 8 | 24 |
| 12 | 7 | 20 | 28 | 7 | 10 |
| 13 | 6 | 24 | 29 | 6 | 9 |
| 14 | 5 | 10 | 30 | 5 | 12 |
| 15 | 4 | 9 | 31 | 4 | 18 |
| 16 | 9 | 12 | 32 | 9 | 24 |

## П.3. Индивидуальные задания для исследования сложения чисел с фиксированной запятой.

**Варианты заданий исследования сумматора чисел в формате с плавающей запятой**

**Таблица П.4**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| №п/п | Ограничение  на мантиссу B | Ограничение  на порядок B | №п/п | Ограничение  на мантиссу B | Ограничение  на мантиссу B |
| 1 | x1xxxxxxxxxxxxxxxxxxxxx | xx1xx0xx | 17 | xxxxxxxxxxxxxxxxx1xxxxx | x1x0xxxx |
| 2 | xx1xxxxxxxxxxxxxxxxxxxx | xx1xx1xx | 18 | xxxxxxxxxxxxxxxxxx1xxxx | x1x1xxxx |
| 3 | xxx1xxxxxxxxxxxxxxxxxxx | xx0xx0xx | 19 | xxxxxxxxxxxxxxxxxxx1xxx | x0x0xxxx |
| 4 | xxxx1xxxxxxxxxxxxxxxxxx | xx0xx1xx | 20 | xxxxxxxxxxxxxxxxxxxx1xx | x0x1xxxx |
| 5 | xxxxx1xxxxxxxxxxxxxxxxx | x1xx0xxx | 21 | xxxxxxxxxxxxxxxxxxxxx1x | xxx1x0xx |
| 6 | xxxxxx1xxxxxxxxxxxxxxxx | x1xx1xxx | 22 | x1xxxxxxxxxxxxxxxxxxxxx | xxx1x1xx |
| 7 | xxxxxxx1xxxxxxxxxxxxxxx | x0xx0xxx | 23 | xx1xxxxxxxxxxxxxxxxxxxx | xxx0x0xx |
| 8 | xxxxxxxx1xxxxxxxxxxxxxx | x0xx1xxx | 24 | xxx1xxxxxxxxxxxxxxxxxxx | xxx0x1xx |
| 9 | xxxxxxxxx1xxxxxxxxxxxxx | xxx1xx0x | 25 | xxxx1xxxxxxxxxxxxxxxxxx | xxxx1x0x |
| 10 | xxxxxxxxxx1xxxxxxxxxxxx | xxx1xx1x | 26 | xxxxx1xxxxxxxxxxxxxxxxx | xxxx1x1x |
| 11 | xxxxxxxxxxx1xxxxxxxxxxx | xxx0xx0x | 27 | xxxxxx1xxxxxxxxxxxxxxxx | xxxx0x0x |
| 12 | xxxxxxxxxxxx1xxxxxxxxxx | xxx0xx1x | 28 | xxxxxxx1xxxxxxxxxxxxxxx | xxxx0x1x |
| 13 | xxxxxxxxxxxxx1xxxxxxxxx | xx1x0xxx | 29 | xxxxxxxx1xxxxxxxxxxxxxx | x1xxx0xx |
| 14 | xxxxxxxxxxxxxx1xxxxxxxx | xx1x1xxx | 30 | xxxxxxxxx1xxxxxxxxxxxxx | x1xxx1xx |
| 15 | xxxxxxxxxxxxxxx1xxxxxxx | xx0x0xxx | 31 | xxxxxxxxxx1xxxxxxxxxxxx | x0xxx0xx |
| 16 | xxxxxxxxxxxxxxxx1xxxxxx | xx0x1xxx | 32 | xxxxxxxxxxx1xxxxxxxxxxx | x0xxx1xx |

## Шаблон плана тестирования для сумматора модулей чисел

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| №  п/п | Тест | Входные воздействия | | | | | Результат | | | |
| a[7..0] | | b[7..0] | | cin | s[7..0] | | Cout/  over | Соот-  ветствует ожидаемому |
| двоичное | десятичное | двоичное | десятичное | двоичное | десятичное |
| 1 | Суммирование без переполнения |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  |
| 5 | Суммирование с переполнением |  |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |  |  |  |

## Шаблон плана тестирования для сумматора в обратных/дополнительных кодах

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| №  п/п | Тест | | Входные воздействия | | | | Результат | | | | |
| a[7..0] | | b[7..0] | | s[7..0] | | overflow | cout | Соответствует ожидаемому |
| двоичное | десятичное | двоичное | десятичное | двоичное | десятичное |
| 1 | A > 0,  B > 0 | без переполнения |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |  |
| 3 | с переполнением |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  |
| 5 | A < 0,  B < 0 | без переполнения |  |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |  |
| 7 | с переполнением |  |  |  |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |  |  |  |
| 9 | A > 0,  B < 0 | положительный результат |  |  |  |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |  |  |  |
| 11 | отрицательный результат |  |  |  |  |  |  |  |  |  |
| 12 |  |  |  |  |  |  |  |  |  |

## Шаблон плана тестирования для сумматора с плавающей запятой

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| №  п/п | Тест | | Входные воздействия | | | | Результат | | | | | |
| A\_ifp | | B\_ifp | | Y\_ofp | | NaN | overflow | underflow | zero |
| двоичное | десятичное  знаковое | двоичное | десятичное  знаковое | двоичное | десятичное  знаковое |
| 1 | A > 0,  B > 0 | положительный десятичный порядок результата | x xxxxxxxx  xxxxxxxxxxxxxxxxxxxxxxx |  | x xxxxxxxx  xxxxxxxxxxxxxxxxxxxxxxx |  | x xxxxxxxx  xxxxxxxxxxxxxxxxxxxxxxx |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |  |  |
| 3 | отрицательный десятичный порядок результата |  |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  |  |
| 5 | A < 0,  B < 0 | положительный десятичный порядок результата |  |  |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |  |  |
| 7 | отрицательный десятичный порядок результата |  |  |  |  |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |  |  |  |  |
| 9 | A > 0,  B < 0 | положительный десятичный порядок результата |  |  |  |  |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |  |  |  |  |
| 11 | отрицательный десятичный порядок результата |  |  |  |  |  |  |  |  |  |  |
| 12 |  |  |  |  |  |  |  |  |  |  |
| 13 | A = NaN | |  |  |  |  |  |  |  |  |  |  |
| 14 | A = +infinitiy | |  |  |  |  |  |  |  |  |  |  |
| 15 | A = -infinitiy | |  |  |  |  |  |  |  |  |  |  |
| 16 | A = -0 | |  |  |  |  |  |  |  |  |  |  |
| 17 | A = +0 | |  |  |  |  |  |  |  |  |  |  |
| 18 | денормализованное A | |  |  |  |  |  |  |  |  |  |  |
| 19 | Y = NaN | |  |  |  |  |  |  |  |  |  |  |
| 20 | Y = ovf | |  |  |  |  |  |  |  |  |  |  |
| 21 | Y = underf | |  |  |  |  |  |  |  |  |  |  |
| 22 | Y = zero | |  |  |  |  |  |  |  |  |  |  |