Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа компьютерных технологий и информационных систем

**Отчёт по лабораторной работе Lab\_MS\_SV\_3**

Дисциплина: Автоматизация проектирования дискретных устройств (на английском языке)

Выполнил студент гр. 5130901/10101 М.Т. Непомнящий

(подпись)

Руководитель А.А. Федотов

(подпись)

Санкт-Петербург

2024

**Оглавление**

[1. Задание 4](#_Toc161275562)

[1.1. Цель работы 4](#_Toc161275563)

[2. Ход решения 5](#_Toc161275564)

[2.1. Создание LFSR модуля 5](#_Toc161275565)

[2.2. Создание теста первого класса 6](#_Toc161275566)

[2.3. Создания модуля для тестирования на плате 9](#_Toc161275567)

[2.4. Настройка Signal Tap II 10](#_Toc161275568)

[2.5. Тестирование на плате средствами Signal Tap II 11](#_Toc161275569)

[3. Вывод 13](#_Toc161275570)

**Список иллюстраций**

[Рис. 1 – Пакет для типов данных 5](#_Toc161275571)

[Рис. 2 – Модуль АЛУ 5](#_Toc161275572)

[Рис. 3 – Структура модуля АЛУ в RTL Viewer 6](#_Toc161275573)

[Рис. 4 – Тестовый файл для модуля АЛУ (1) 6](#_Toc161275574)

[Рис. 5 – Тестовый файл для модуля АЛУ (2) 7](#_Toc161275575)

[Рис. 6 – Моделирование тестового файла средствами ModelSim (wave) 7](#_Toc161275576)

[Рис. 7 – Моделирование тестового файла средствами ModelSim (cmd) 7](#_Toc161275577)

[Рис. 8 – Модифицированный модуль для теста первого класса 8](#_Toc161275578)

[Рис. 9 – Общая вывод после симуляции модифицированного tb 8](#_Toc161275579)

[Рис. 10 – Результат SLU\_out > 127 9](#_Toc161275580)

[Рис. 11 – Обработка случая остатка от деления на 0 9](#_Toc161275581)

[Рис. 12 – Обработка случая деления на 0 9](#_Toc161275582)

[Рис. 13 – Модуль db для тестирования на плате 9](#_Toc161275583)

[Рис. 14 – Настройка окна Signal Tap II 10](#_Toc161275584)

[Рис. 15 – Настройка Signal Probe 10](#_Toc161275585)

[Рис. 16 – Мнемоническая таблица 10](#_Toc161275586)

[Рис. 17 – Симуляция в Signal Probe (op\_a = 10, op\_b = 3) 11](#_Toc161275587)

[Рис. 18 – Результат SP (ADD, op\_a = 10, op\_b = 3) 11](#_Toc161275588)

[Рис. 19 – Результат SP (SUB, op\_a = 10, op\_b = 3) 11](#_Toc161275589)

[Рис. 20 – Результат SP (MUL, op\_a = 10, op\_b = 3) 11](#_Toc161275590)

[Рис. 21 – Результат SP (DIV, op\_a = 10, op\_b = 3) 11](#_Toc161275591)

[Рис. 22 – Результат SP (VAR, op\_a = 10, op\_b = 3) 11](#_Toc161275592)

[Рис. 23 – Результат SP (ADD, op\_a = 127, op\_b = 127) 12](#_Toc161275593)

[Рис. 24 – Временные характеристики устройства 12](#_Toc161275594)

# Задание

## Цель работы

Создать АЛУ (арифметико-логическое устройство) с параметризированной разрядностью (width=8, по умолчанию). Реализуется как комбинационная схема. Выполняет знаковые операции:

* ADD – сложение.
* SUB – вычитание.
* MUL – умножение.
* DIV – деление.
* VAR – по варианту

Выводы устройства:

* ops – вход кода операции (**3** бита – в Вашем проекте, в примерах ниже – 2 бита ).
* op\_a – вход операнда А: знаковый (разрядность width).
* op\_b – вход операнда B: знаковый (разрядность width).
* ALU\_out – выход результата: знаковый (разрядность width).
* CLK – вход тактового сигнала, для синхронизации ISSPE и SignalTapII.
  1. **Программа работы:**

1. Разработать описание устройства АЛУ (модуль lab\_MS\_SV4) на SystemVerilog, используя пакет и структуру.

* Создать пакет (lab\_MS\_SV4\_pack.sv) с определением структуры и другими необходимыми конструкциями.
* В основном файле (lab\_MS\_SV4.sv) передать структуру в модуль.

1. Разработать тест (tb\_lab\_MS\_SV4.sv) для проверки АЛУ.

* Создать тест первого класса без автоматической проверки. Выбрать значения данных, демонстрирующие правильность работы всех операций, включая дополнительные.
* Представить результаты теста в отчете: временная диаграмма и вывод в консоль.

1. Разработать модуль верхнего уровня для отладки (db\_lab\_MS\_SV4), содержащий модуль lab\_MS\_SV4 и модуль SP\_unit.

* Модуль SP\_unit обеспечивает возможность задания входных сигналов, синхронизируемых тактовым сигналом CLK, без использования кнопок на плате, и отображения результата.

1. Используя ISSPE, провести анализ работы lab\_MS\_SV4 и зафиксировать работоспособность устройства АЛУ, демонстрируя выполнение всех операций, включая дополнительные.
2. Настроить SignalTapII для получения временной диаграммы, аналогичной примеру, и отобразить соответствующую дополнительную операцию.

# Ход решения

## Создание LFSR модуля

Создадим пакет для типов данных, которые будут использоваться в проекте:

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рис. 1 – Пакет для типов данных

В этом пакете мы храним коды операций (ADD='0, SUB, MUL, DIV, VAR) и общий тип данных для их входов и выходов. Эта информация помещена в один общий пакет.

Теперь, в соответствии с только что созданным модулем и вариантом задания создадим модуль АЛУ, где пропишем то, как работают операции с арифметической точки зрения:

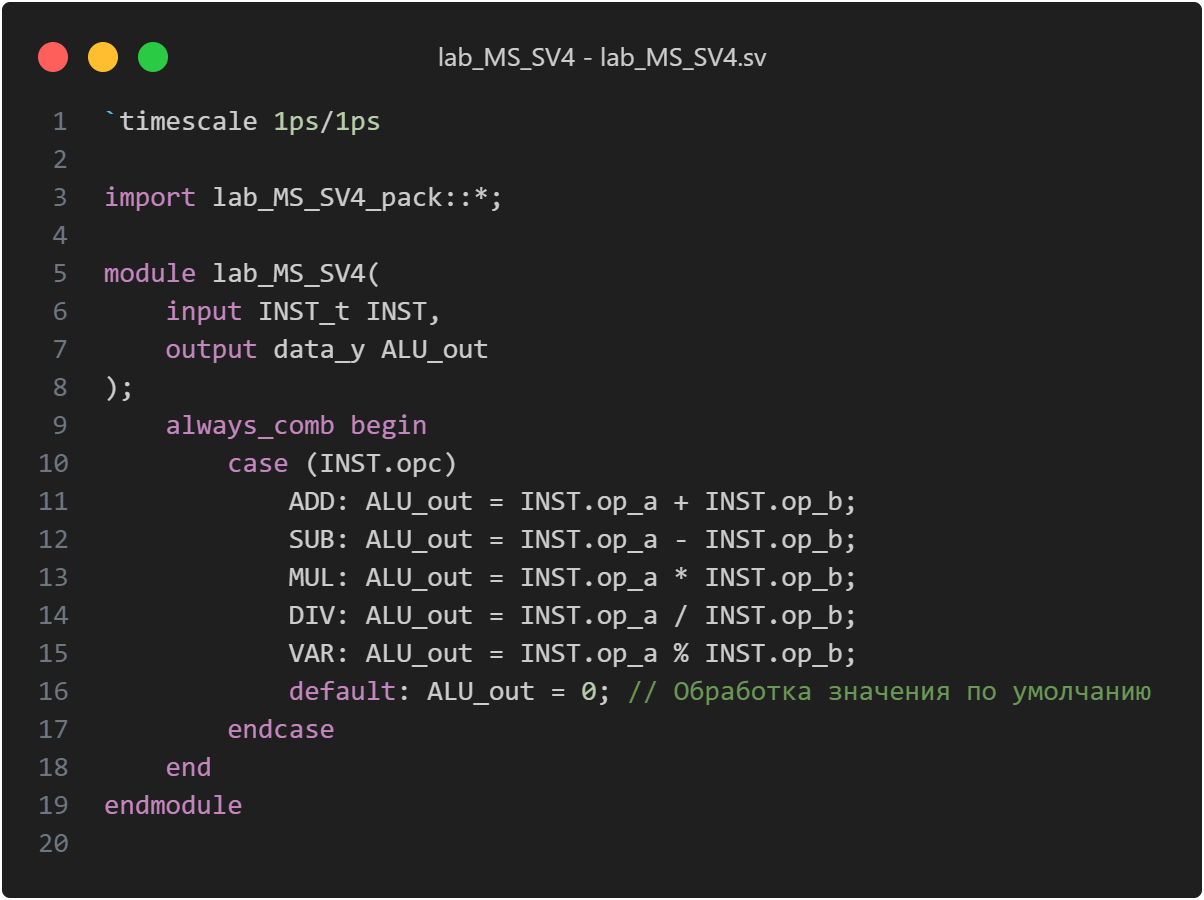


Рис. 2 – Модуль АЛУ

\*Здесь VAR – это операция нахождения остатка от деления нацело op\_a на op\_b, заданная по варианту.

После компиляции данного модуля можем увидеть, что его структура в RTL Viewer выглядит следующим образом:

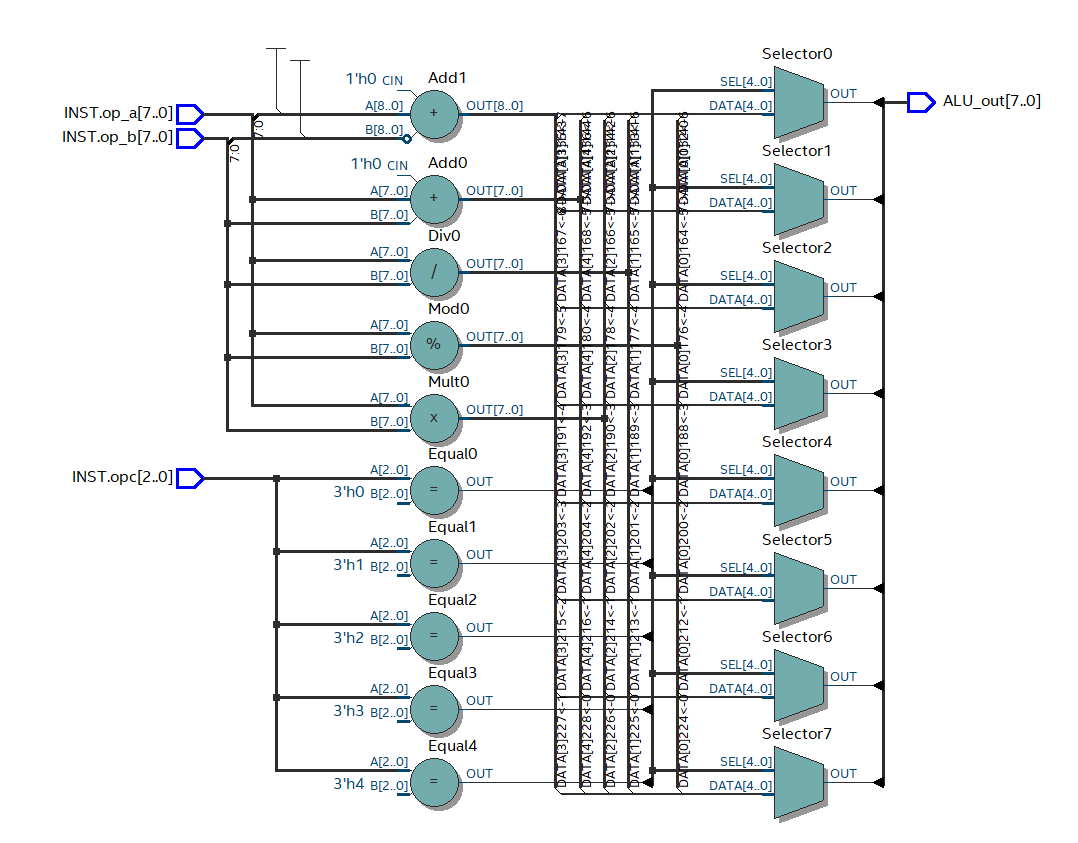


Рис. 3 – Структура модуля АЛУ в RTL Viewer

## Создание теста первого класса

Напишем тест первого класса для только что созданного модуля АЛУ:

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рис. 4 – Тестовый файл для модуля АЛУ (1)

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рис. 5 – Тестовый файл для модуля АЛУ (2)

Данный тест позволит пройтись по всем обрабатываемым в модуле АЛУ операциям и выведет их результат в консоль. Чтобы проверить коррекстность работы данного теста, укажем в качестве рассматриваемых значений op\_a = 10, op\_b = 3.

Изображение выглядит как Мультимедийное программное обеспечение, программное обеспечение, электроника, снимок экрана

Автоматически созданное описание

Рис. 6 – Моделирование тестового файла средствами ModelSim (wave)

Изображение выглядит как текст, чек, Шрифт, снимок экрана

Автоматически созданное описание

Рис. 7 – Моделирование тестового файла средствами ModelSim (cmd)

Заметим, что данный тест ограничен количеством одной парой значений. Чтобы провести полноценное тестирование модифицируем файл tb таким образом, чтобы он охватывал все возможные пары значений op\_a и op\_b (реализуем с помощью вложенного цикла for):

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рис. 8 – Модифицированный модуль для теста первого класса

Теперь смоделируем созданный файл средствами ModelSim и проанализируем полученные данные:

Изображение выглядит как снимок экрана, линия, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рис. 9 – Общая вывод после симуляции модифицированного tb

Рассмотрим отдельно краевые значения:

Изображение выглядит как снимок экрана, Мультимедийное программное обеспечение, линия, программное обеспечение

Автоматически созданное описание

Рис. 10 – Результат SLU\_out > 127

На рисунке выше можем увидеть, что выводится отрицательное число, однако, если раскрыть его и посмотреть на его бинарный код, то станет понятно, что это связано с тем, что мы выводим значения в дополнительном коде.

Также, в код добавлены обработки специальных случаев, которые выводятся в консоль. На рисунке ниже представлен пример такого вывода для обработки ошибочных значений при выполнении операции, заданной по варианту (остаток от деления на ноль):

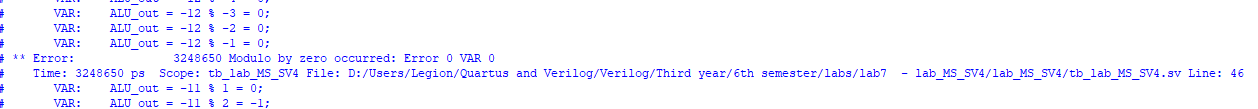


Рис. 11 – Обработка случая остатка от деления на 0

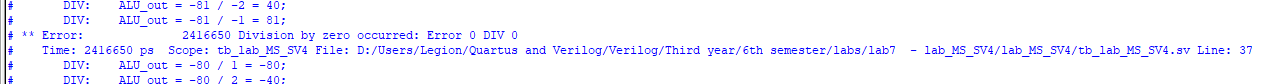


Рис. 12 – Обработка случая деления на 0

Остальные граничные случаи также обрабатываются корректно, однако в конкретном тесте сложно найти их выводы из-за обработки всех возможных значений (синтаксис обработки таких случаев схож с приставленным выше примером, поэтому можно просто посмотреть код, чтобы увидеть, что будет выводиться в конкретном случае).

Как видим, все операции работают корректно и выводят верный результат.

## Создания модуля для тестирования на плате

Теперь разработаем модуль для тестирования программы на плате:

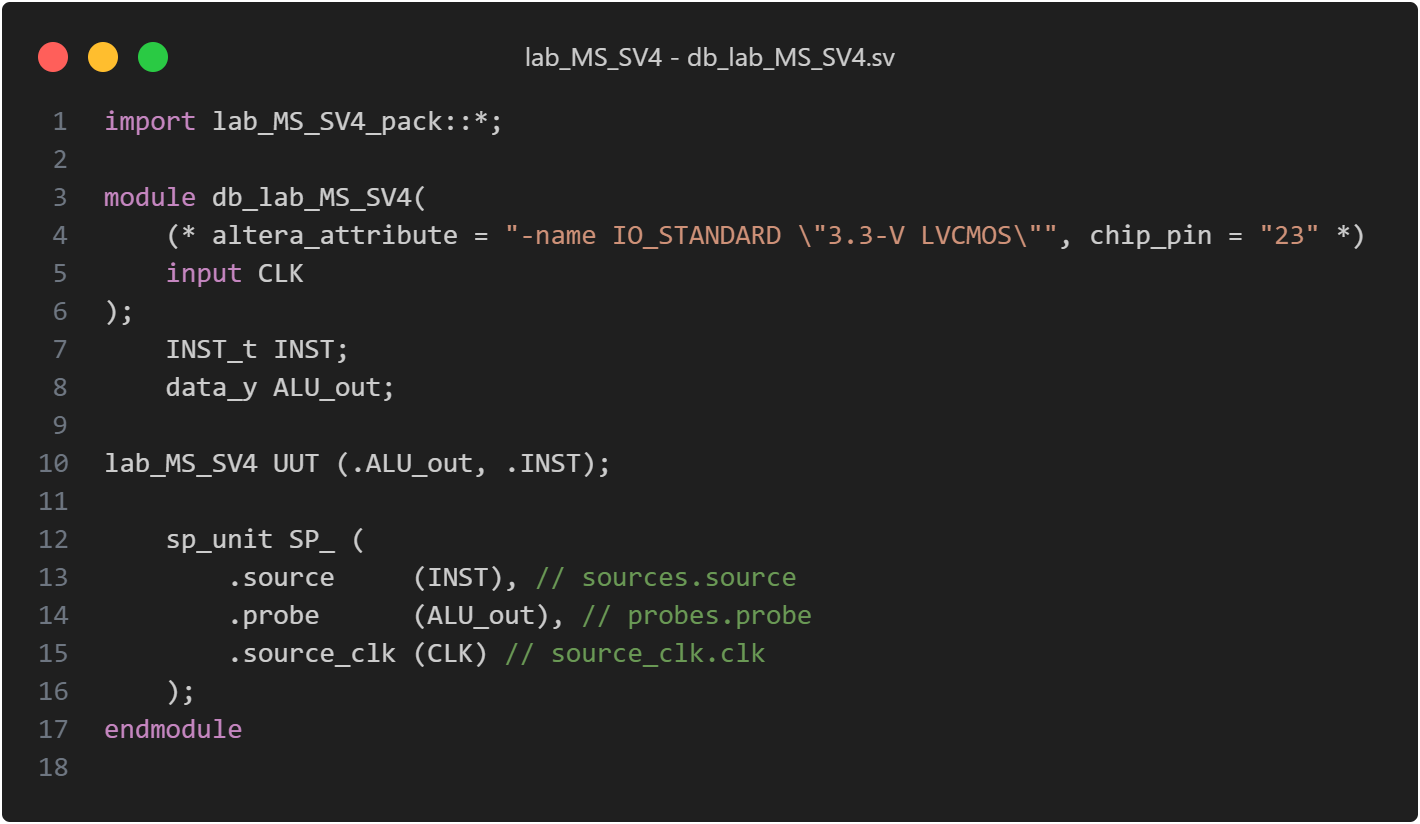


Рис. 13 – Модуль db для тестирования на плате

## Настройка Signal Tap II

Для ввода значений в модуль будем использовать ISSP, там же будем смотреть результат. Дополнительно добавим Signal Tap II, в котором будем получать значения по изменению типа операции и получать результат в виде сегментов:

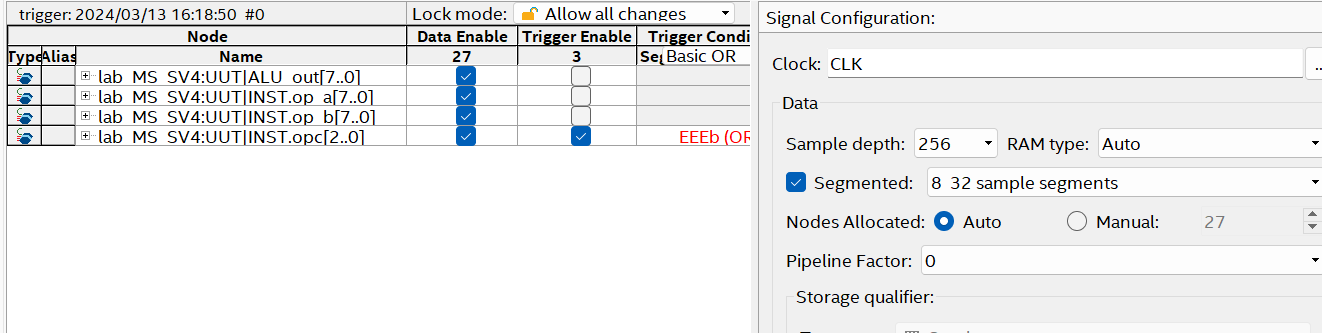


Рис. 14 – Настройка окна Signal Tap II

Перегруппируем S[18..0] сигналы в SP так, чтобы первая подгруппа (S[18..16]) отвечала за номер операции в пакете, вторая подгруппа (S[15..8]) отвечала бы за первое число op\_a, а третья подгруппа (S[7..0]) – за число op\_b:

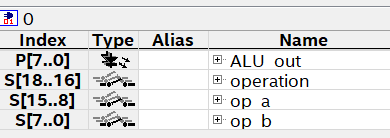


Рис. 15 – Настройка Signal Probe

Составим мнемоническую таблицу, чтобы при симуляции отображался не просто код операции, а её название:

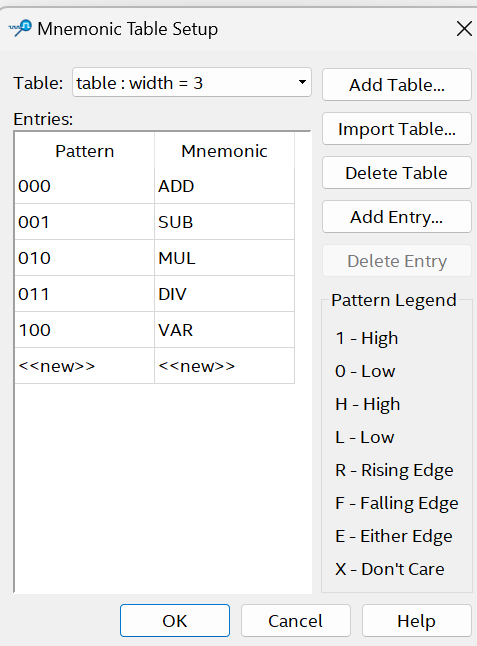


Рис. 16 – Мнемоническая таблица

## Тестирование на плате средствами Signal Tap II

Теперь запустим и выполним проверку корректности работы программы на плате. Выполним загрузку разработанного модуля на плату и запустим тестирование:

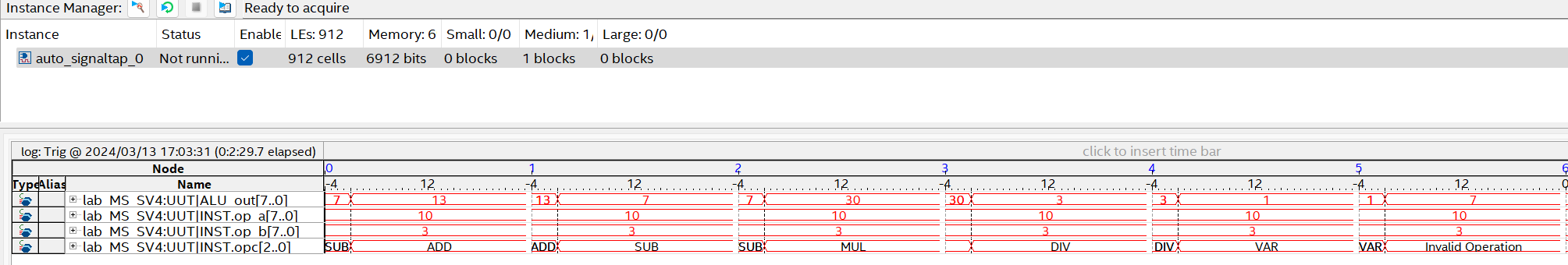


Рис. 17 – Симуляция в Signal Probe (op\_a = 10, op\_b = 3)

Как видим, значение было посчитано верно.

Теперь будем последовательно выполнять ввод номера операции, а также сами значения переменных непосредственно в редакторе SP, чтобы значение ALU\_out выводилось в режиме реального времени. Примеры таких тестов для одной пары чисел представлены ниже (op\_a = 10, op\_b = 3):

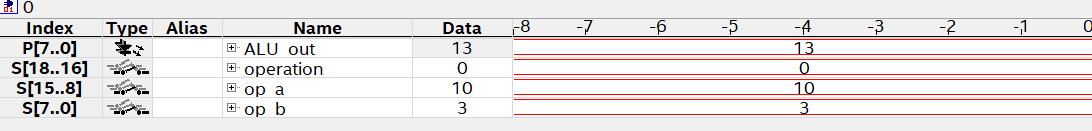


Рис. 18 – Результат SP (ADD, op\_a = 10, op\_b = 3)

Изображение выглядит как текст, снимок экрана, линия, Параллельный

Автоматически созданное описание

Рис. 19 – Результат SP (SUB, op\_a = 10, op\_b = 3)

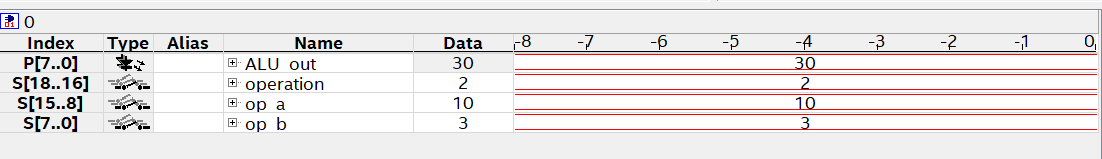


Рис. 20 – Результат SP (MUL, op\_a = 10, op\_b = 3)

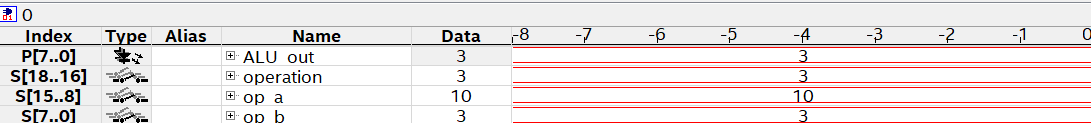


Рис. 21 – Результат SP (DIV, op\_a = 10, op\_b = 3)

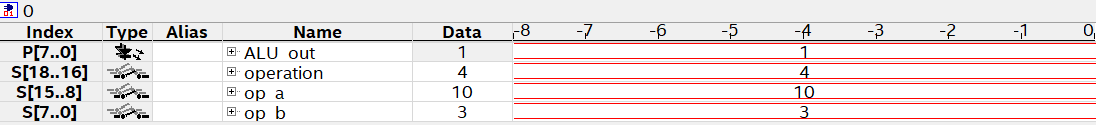


Рис. 22 – Результат SP (VAR, op\_a = 10, op\_b = 3)

Протестируем случай с переполнением:

Изображение выглядит как текст, снимок экрана, линия, Параллельный

Автоматически созданное описание

Рис. 23 – Результат SP (ADD, op\_a = 127, op\_b = 127)

Заметим, что результат выдаётся неверный. Это связано с тем, что происходит переполнение, поэтому при вводе есть ограничение на вводимые числа. Чтобы её вычислить надо рассмотреть операцию умножения, так как эта операция позволяет получить наибольшую разрядность выходного числа при одних и тех же входных переменных (размерности переменных складываются).

Выполним полную компиляцию. В отчете о компиляции видно, что устройство удовлетворяет временным параметрам.

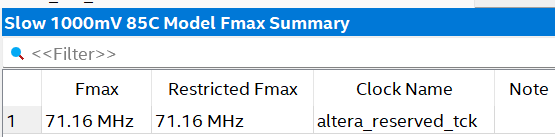


Рис. 24 – Временные характеристики устройства

# Вывод

В результате создания проекта на разработку АЛУ с параметризированной разрядностью и использованием пакетирования в SystemVerilog было достигнуто несколько значимых результатов.

Во-первых, использование пакета для хранения типов данных и определения кодов операций позволило сделать код более читаемым и модульным. Это упростило разработку и отладку проекта, так как изменения в типах данных или кодах операций можно внести в едином месте, и они автоматически отразятся на всех остальных частях проекта.

Во-вторых, модульность проекта была улучшена благодаря пакетированию. Каждый модуль (ALU, тестовое окружение, модуль для тестирования на плате) был разделен на отдельные компоненты, что сделало проект более гибким и масштабируемым. Это позволяет легко расширять функциональность проекта или переиспользовать его компоненты в других проектах.

Наконец, использование пакета способствует повышению производительности и уменьшению вероятности ошибок. За счет того, что типы данных и коды операций определены один раз и использованы во всех частях проекта, уменьшается вероятность ошибок при внесении изменений и обеспечивается единообразие данных в проекте.

Таким образом, применение пакетирования в проекте на разработку АЛУ существенно облегчило процесс разработки, повысило его модульность и гибкость, а также способствовало повышению производительности и уменьшению вероятности ошибок.