Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа компьютерных технологий и информационных систем

**Отчёт по лабораторной работе № 12**

Дисциплина: Языки описания аппаратных средств вычислительных систем.

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

(подпись)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. А. Федотов

(подпись)

Санкт-Петербург

2023

Оглавление

[1. Задание 2](#_Toc153743231)

[2. Описание на языке Verilog 3](#_Toc153743232)

[3. Вывод: 8](#_Toc153743233)

# Задание

На языке Verilog разработать:

* Параметризированный умножитель, реализующий умножение [W-1:0] dA на [W-1:0] dB по алгоритму «умножение старшими разрядами вперед со сдвигом суммы.
* Разработать тест 2 класса для всех возможных вариантов входных данных.

***Модуль mult\_SR\_V1:***

**Входы:**

* сlk – тактовый сигнал.
* [W-1:0] dA – вход данных A.
* [W-1:0] dB – вход данных B.

**Выходы:**

* [2\*W-1:0] a\_mult\_b – результат умножения.
* ready – сигнал готовности результата умножения.

На языке Verilog разработать:

* Модуль lab\_4s – умножитель двух 4-разрядных чисел, обеспечивающий отображение результата на 4-разрядном 7-сегментном индикаторе.
* Реализовать модуль на плате.
* Разработать тест первого класса.

***Модуль lab\_4s:***

**Входы:**

* сlk – тактовый сигнал.
* [3:0] dA – вход данных A.
* [3:0] dB – вход данных B.

**Выходы:**

* [6:0] ss – значение для 7-сегментного индикатора.
* [4:1] dig – значение для числа, куда выводить значение.

Изображение выглядит как текст, диаграмма, линия, Шрифт

Автоматически созданное описание

Рис. 1.1 – Структура устройства lab\_4s

**Состав устройства:**

* ss\_cntr – блок управления динамической индикацией на 7-сегментном индикаторе, используя модуль из lab\_2s.
* b2db\_ROM – блок преобразования 8-разрядного двоичного кода в двоично-десятичный, используя модуль из lab\_3s.
* mult\_SR\_V1 – модуль умножения.

# Описание на языке Verilog

Алгоритм умножителя старшими разрядами вперед выглядит следующим образом:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рис. 2.1 – Алгоритм умножения старшими разрядами вперед.

Также рассмотрим следующий пример:

Изображение выглядит как текст, число, Шрифт, Параллельный

Автоматически созданное описание

Как видно, данный алгоритм имеет два этапа вычислений, первый – сдвиг, второй – суммирование.

Как-то выполнить эти процессы параллельно невозможно, поэтому в один так будет необходимо выполнять одну операцию, во второй такт вторую, для этого реализуем конечный автомат.

Поскольку на входе умножителя нет сигнала загрузки, реализуем его самостоятельно, используя 2 последовательных регистра, что также добавит защиту от метастабильности:

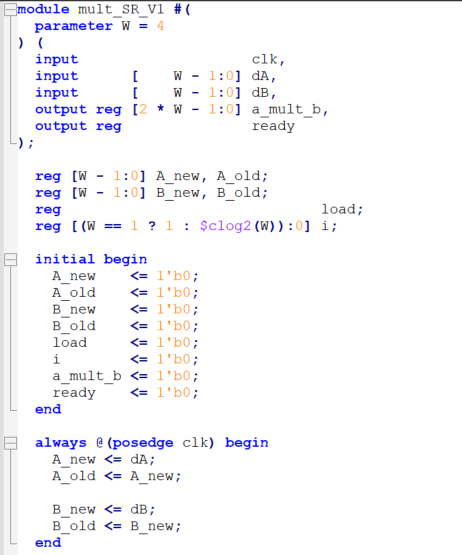


Рис. 2.2 – Задание параметров и начальных значений модуля mult\_SR\_V1

Программа на Verilog, представленная в изображении, реализует синхронный последовательностный умножитель с переносом (SR).

**Инициализация переменных**

В начальном состоянии все переменные инициализируются нулями.

initial begin

A\_new <= 1'b0;

A\_old <= 1'b0;

B\_new <= 1'b0;

B\_old <= 1'b0;

load <= 1'b0;

i <= 1'b0;

a\_mult\_b <= 1'b0;

end

**Циклический перенос**

На каждом такте цикла происходит следующее:

* Переменные A\_new и A\_old обновляются новыми значениями коэффициентов A и B.
* Переменные load и i обновляются в соответствии с состоянием переноса.

always @(posedge clk) begin

A\_new <= dA;

A\_old <= A\_new;

B\_new <= dB;

B\_old <= B\_new;

if (A\_new != A\_old | B\_new != B\_old) load <= 1'b1;

else load <= 1'b0;

end

**Запись результата**

Если на выходе регистра load установлен один, то происходит запись результата умножения в выходную переменную a\_mult\_b.

always @(posedge clk) begin

if (load) a\_mult\_b <= A\_new \* B\_new;

end

Далее реализуем сам конечный автомат. По сигналу load он должен сбрасывать все значения выходов в ноль, а также приводить в начальное состояние счетчик итераций.

Далее происходит сдвиг, после чего сложение результата.

В случае, если выполнено достаточно итераций для завершения формирования выхода, ответ должен зафиксироваться:

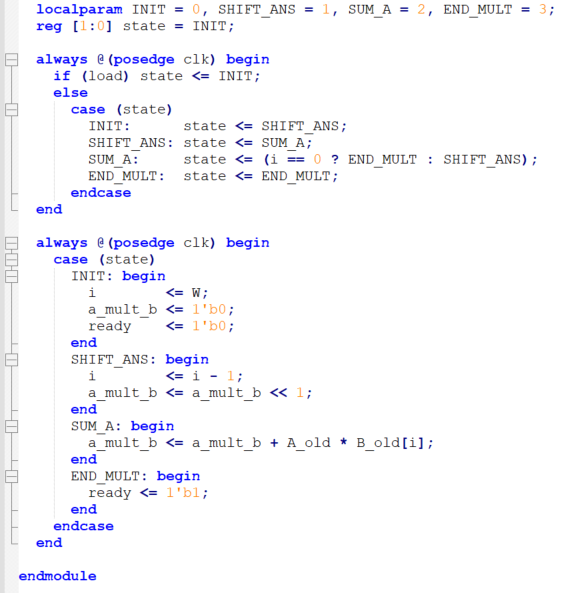


Рис. 2.3 – Реализация конечного автомата модуля mult\_SR\_V1

Циклический перенос выполняется в блоке always @(posedge clk). На каждом такте цикла переменная state обновляется в соответствии с состоянием алгоритма. Если переменная state имеет значение INIT, то происходит инициализация переменных a\_mult\_b и i. Если переменная state имеет значение SHIFT\_ANS, то происходит сдвиг результата умножения на один бит влево и добавление очередного коэффициента A. Если переменная state имеет значение SUM\_A, то проверяется, достигнут ли конец произведения. Если да, то переменная state устанавливается в значение END\_MULT. Если нет, то переменная state устанавливается в значение SHIFT\_ANS.

Запись результата выполняется в блоке always @(posedge clk). Если на выходе регистра ready установлен один, то производится запись результата умножения в выходную переменную a\_mult\_b. Результат умножения определяется как значение переменной a\_mult\_b.

Выполним тестирование разработанного модуля:

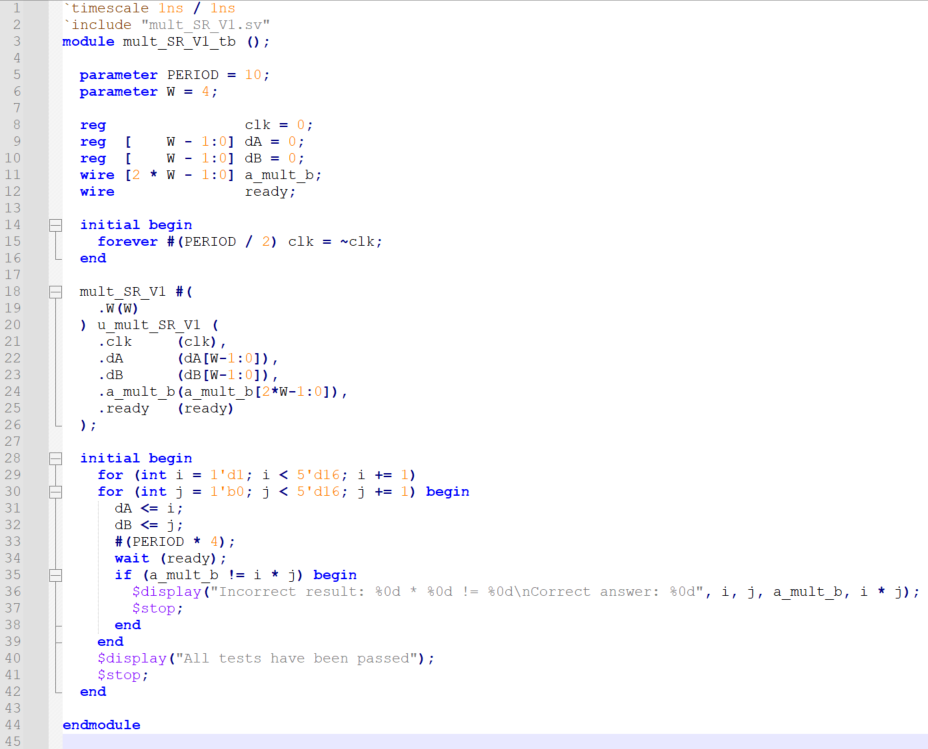


Рис. 2.4 – Тестовый модуль для mult\_SR\_V1

Циклический перенос выполняется в блоке always @(posedge clk). На каждом такте цикла переменная i увеличивается на единицу. Затем, производится вычисление очередного слагаемого произведения. Для этого, значение коэффициента A помещается в младший разряд выходной переменной a\_mult\_b. Затем, производится сдвиг выходной переменной a\_mult\_b на один бит влево. После этого, в старший разряд выходной переменной a\_mult\_b помещается произведение значений коэффициента A и коэффициента B.

Запись результата выполняется в блоке always @(posedge clk). Если на выходе регистра ready установлен один, то производится запись результата умножения в выходную переменную a\_mult\_b. Результат умножения определяется как значение переменной a\_mult\_b.

Этот тестовый модуль проверяет все возможные тестовые последовательности для i и j:

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

Автоматически созданное описание

Рис. 2.5 – Результат тестирования модуля mult\_SR\_V1

Все тестовые последовательности выдали корректный результат, что видно в консоли и на wave.

Далее реализуем модуль lab\_4s, используя модули из лабораторных lab\_3s и lab\_2s:

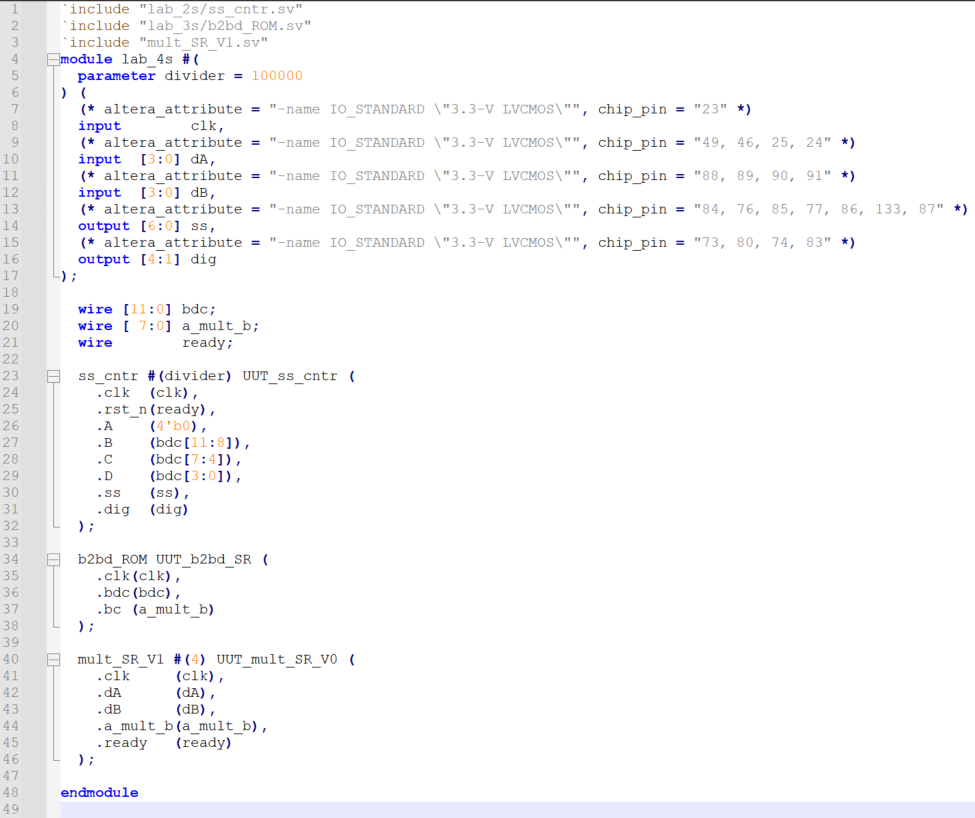


Рис. 2.6 – Модуль lab\_4s

Для реализации на плате divider устанавливается равным 100000 для того, чтоб на 7-сегментном индикаторе числа было хорошо видно, однако при тестировании, значение необходимо будет уменьшить, поэтому целесообразно использовать divider как параметр.

Выполним тестирование реализованного модуля, используя следующий тест:

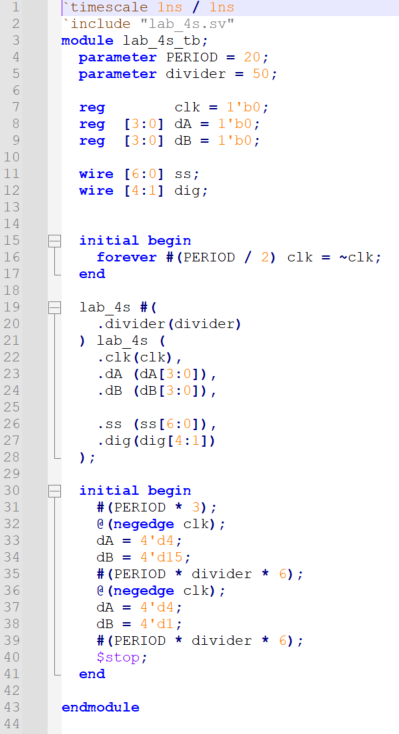


Рис. 2.7 – Модуль lab\_4s\_tb

Выполним моделирование реализованного теста:

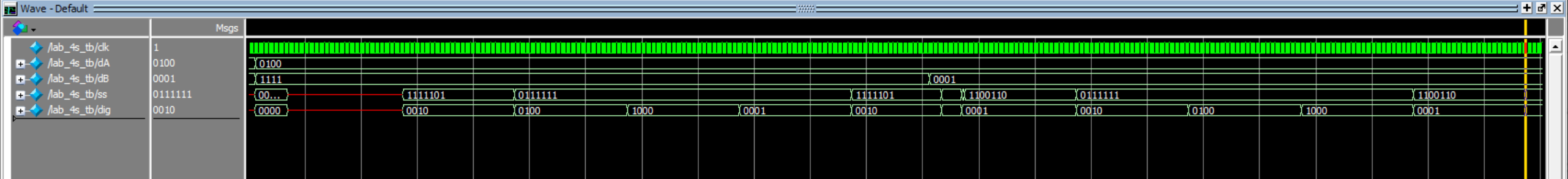


Рис. 2.8 – Результат моделирования модуля lab\_4s\_tb

Моделирование показало, что модуль работает правильно. При умножении 15 на 4 на индикатор выводится 0060, а при умножении 4 на 1–0004.

На рисунке видно, что в процессе пересчета на индикаторе появляются шумы, но они не заметны из-за высокой частоты.

Модуль был успешно протестирован на плате. Он корректно работал для всех тестовых значений.

# Вывод:

Проведя лабораторные исследования, было создано устройство для умножения чисел разной разрядности, использующее схему "старшие разряды вперед со сдвигом суммы". Это устройство подверглось тщательному тестированию в лаборатории на специальной стенде с применением тестовых модулей. Результаты тестов подтвердили правильную и эффективную работу данной схемы.

Применение языка программирования Verilog заметно упростило весь процесс разработки подобных устройств по сравнению с традиционным ручным вводом схем. Этот метод также обеспечил удобство интеграции уже готовых модулей, что значительно ускорило процесс разработки. Кроме того, использование программы ModelSim не только дало возможность провести более сложное тестирование устройства, но и повысило его надежность, что в конечном итоге способствует более эффективному внедрению разработанного устройства в практические задачи, такие как вычислительные операции в электронике или информационные технологии.