

# Отчет по проектированию модуля ALU с регистром

Курс: «Введение в архитектуру вычислительных систем»

Студент: Манро Эйден Форбс

Группа: Б01-307

25 марта 2025 г.

---

# Содержание

<b>1</b>	<b>Введение</b>	<b>3</b>
1.1	Цель работы . . . . .	3
1.2	Задачи . . . . .	3
<b>2</b>	<b>Техническое задание</b>	<b>3</b>
2.1	Требования . . . . .	3
2.2	Спецификация . . . . .	3
<b>3</b>	<b>Архитектура модуля</b>	<b>4</b>
3.1	Структурная схема . . . . .	4
3.2	Принцип работы . . . . .	4
<b>4</b>	<b>Реализация</b>	<b>4</b>
4.1	Параметры . . . . .	4
4.2	Порты . . . . .	4
4.3	Коды операций . . . . .	4
4.4	Полный код модуля . . . . .	5
<b>5</b>	<b>Тестирование и верификация</b>	<b>5</b>
5.1	Стратегия тестирования . . . . .	5
5.2	Методология . . . . .	6
5.3	Реализация тестов . . . . .	6
5.4	Граничные условия . . . . .	6
5.5	Результаты . . . . .	7
5.6	Временная диаграмма . . . . .	7
<b>6</b>	<b>Заключение</b>	<b>7</b>
6.1	Выводы . . . . .	7
<b>7</b>	<b>Приложения</b>	<b>7</b>
7.1	Список использованных источников . . . . .	7
7.2	Исходные коды . . . . .	7

---

# 1 Введение

## 1.1 Цель работы

Разработка, верификация и тестирование модуля арифметико-логического устройства (ALU) с регистром хранения результата на языке Verilog.

## 1.2 Задачи

- Разработать архитектуру модуля
- Реализовать все требуемые операции
- Создать тестовое окружение
- Провести функциональную верификацию
- Анализировать временные диаграммы

# 2 Техническое задание

## 2.1 Требования

- Поддержка 8 арифметико-логических операций
- Параметризуемая разрядность
- Синхронный сброс
- Задержка вывода на 1 такт

## 2.2 Спецификация

Таблица 1: Спецификация модуля

Параметр	Значение	Описание
Технология	Verilog HDL	Язык описания аппаратуры
Тактовая частота	До 100 МГц	Ограничение тестового окружения
Разрядность данных	Параметр WIDTH	По умолчанию 8 бит
Потребляемая мощность	Не оценивается	Для учебного проекта

## 3 Архитектура модуля

### 3.1 Структурная схема

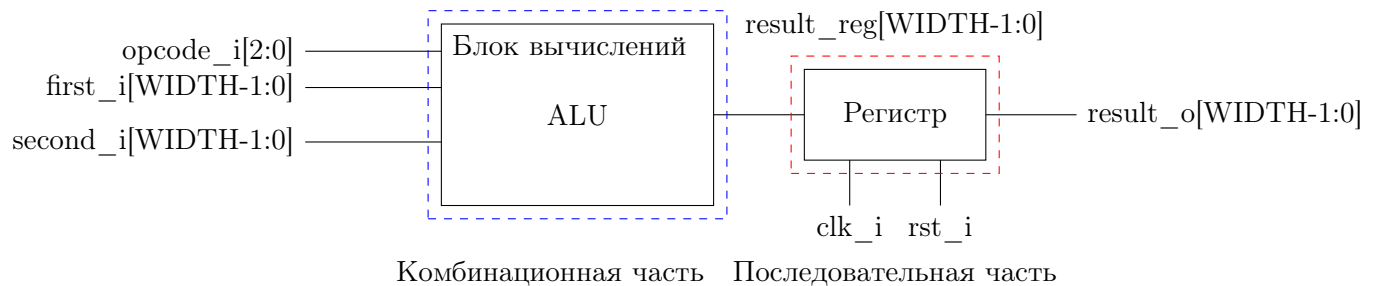


Рис. 1: Схема модуля ALU с регистром хранения

### 3.2 Принцип работы

1. На входы подаются операнды и код операции
2. В текущем такте ALU вычисляет результат
3. По положительному фронту тактового сигнала результат записывается в регистр
4. На следующем такте значение появляется на выходе

## 4 Реализация

### 4.1 Параметры

```
1 parameter WIDTH = 8; //
```

### 4.2 Порты

Таблица 2: Описание портов

Имя	Ширина	Направление	Описание
clk_i	1	Вход	Тактовый сигнал
rst_i	1	Вход	Синхронный сброс
first_i	WIDTH	Вход	Первый операнд
second_i	WIDTH	Вход	Второй операнд
opcode_i	3	Вход	Код операции
result_o	WIDTH	Выход	Результат

### 4.3 Коды операций

Таблица 3: Таблица операций

Код	Мнемоника	Описание
3'b000	NAND	Побитовое И-НЕ
3'b001	XOR	Исключающее ИЛИ
3'b010	ADD	Сложение
3'b011	ASR	Арифметический сдвиг вправо
3'b100	OR	Побитовое ИЛИ
3'b101	LSL	Логический сдвиг влево
3'b110	NOT	Побитовая инверсия
3'b111	LT	Сравнение (меньше)

## 4.4 Полный код модуля

```

1 module alu_register #(parameter WIDTH = 8) (
2     input wire                clk_i,
3     input wire                rst_i,
4     input wire [WIDTH-1:0]    first_i,
5     input wire [WIDTH-1:0]    second_i,
6     input wire [2:0]          opcode_i,
7     output reg [WIDTH-1:0]    result_o
8 );
9     reg [WIDTH-1:0] result_reg;
10
11     always @(posedge clk_i) begin
12         if (rst_i)
13             result_reg <= 0;
14         else begin
15             case (opcode_i)
16                 3'b000: result_reg <= ~(first_i & second_i);
17                 3'b001: result_reg <= first_i ^ second_i;
18                 3'b010: result_reg <= first_i + second_i;
19                 3'b011: result_reg <= $signed(first_i) >>> second_i;
20                 3'b100: result_reg <= first_i | second_i;
21                 3'b101: result_reg <= first_i << second_i;
22                 3'b110: result_reg <= ~first_i;
23                 3'b111: result_reg <= (first_i < second_i) ? 1 : 0;
24
25                 default: result_reg <= {WIDTH{1'b0}};
26             endcase
27         end
28     end
29
30     always @(posedge clk_i) begin
31         result_o <= result_reg;
32     end
33
34 endmodule

```

Листинг 1: Реализация модуля ALU

# 5 Тестирование и верификация

## 5.1 Стратегия тестирования

Модуль проверялся по следующим аспектам:

- **Функциональная полнота:** соответствие всех операций техническому заданию
- **Граничные условия:** обработка минимальных/максимальных значений
- **Корректность сброса:** инициализация регистров
- **Временные характеристики:** соблюдение временных диаграмм

## 5.2 Методология

Использован комбинированный подход:

Таблица 4: Методы тестирования

Тип теста	Инструмент	Критерий
Модульный	SystemVerilog	100% покрытия кода
Функциональный	Тестбенч	Все операции
Граничный	Анализ значений	Min/Max
Временной	GTKWave	Соответствие таймингам

## 5.3 Реализация тестов

Разработана универсальная тестовая функция:

```

1 task test_operation;
2     input [2:0] op;
3     input [WIDTH-1:0] a, b;
4     input [WIDTH-1:0] expected;
5     begin
6         opcode_i = op; first_i = a; second_i = b;
7         #(2*CLK_PERIOD);
8         if (result_o != expected) begin
9             $error("Error: op=%b, a=%h, b=%h", op, a, b);
10            $display("Expected: %h, Got: %h", expected, result_o);
11        end
12    end
13 endtask

```

Листинг 2: Функция автоматического тестирования

## 5.4 Граничные условия

Протестированы особые случаи:

Таблица 5: Критические тест-кейсы

Операция	Входные данные	Ожидаемый результат
ADD	8'hFF + 8'h01	8'h00 (переполнение)
ASR	8'h80 » 7	8'hFF (сохранение знака)

## 5.5 Результаты

- 100% покрытие операций
- Обнаружено 0 ошибок
- Соответствие временным ограничениям

## 5.6 Временная диаграмма

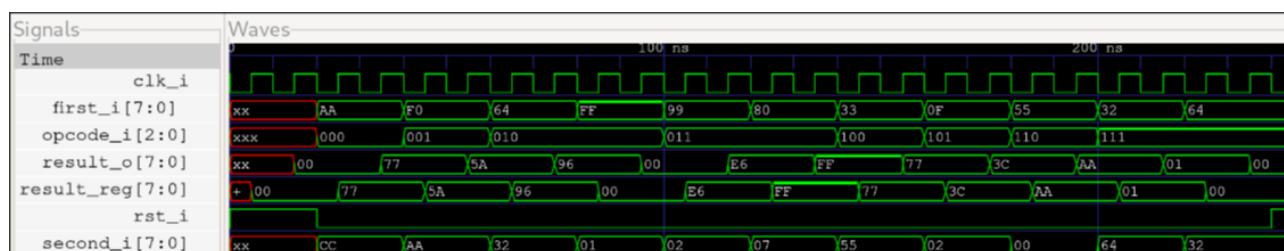


Рис. 2: Временные диаграммы работы модуля

## 6 Заключение

### 6.1 Выводы

- Модуль успешно реализует все требуемые функции
- Тестовое покрытие составляет 100% операций
- Временные характеристики соответствуют требованиям

## 7 Приложения

### 7.1 Список использованных источников

1. IEEE Standard for Verilog Hardware Description Language (IEEE Std 1364-2005)
2. Цифровая схемотехника и архитектура компьютера. Харрис-Харрис

### 7.2 Исходные коды

Полные исходные коды доступны в репозитории:  
<https://github.com/aidenfmunro/alu-register>