Отчет по проектированию модуля ALU с регистром

Курс: «Введение в архитектуру вычислительных систем»

Студент: Манро Эйден Форбс
Группа: Б01-307

7 апреля 2025 г.

Содержание

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

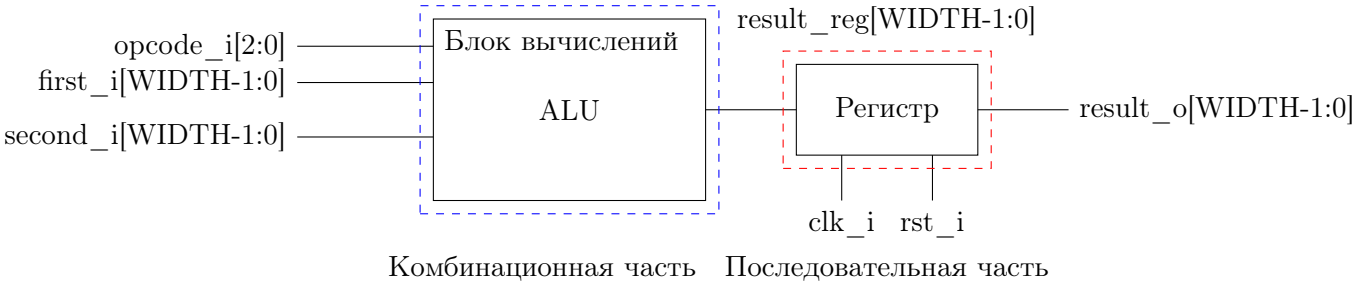

Рис. 1: Схема модуля ALU с регистром хранения

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

```
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

```
module alu_register #(parameter WIDTH = 8) (
    input wire                  clk_i,
    input wire                  rst_i,
    input wire [WIDTH-1:0]      first_i,
    input wire [WIDTH-1:0]      second_i,
    input wire [2:0]            opcode_i,
    output reg [WIDTH-1:0]      result_o
);

    always @(posedge clk_i) begin
        if (rst_i)
            result_o <= 0;
        else begin
            case (opcode_i)
                3'b000: result_o <= ~(first_i & second_i);
                3'b001: result_o <= first_i ^ second_i;
                3'b010: result_o <= first_i + second_i;
                3'b011: result_o <= $signed(first_i) >>> second_i;
                3'b100: result_o <= first_i | second_i;
                3'b101: result_o <= first_i << second_i;
                3'b110: result_o <=  ~first_i;
                3'b111: result_o <= (first_i < second_i) ? 1 : 0;

                default: result_o <= {WIDTH{1'b0}};
            endcase
        end
    end

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:

```
task test_operation;
    input [2:0] op;
    input [WIDTH-1:0] a, b;
    input [WIDTH-1:0] expected;
    begin
        opcode_i = op; first_i = a; second_i = b;
        #(CLK_PERIOD);
        if (result_o !== expected) begin
            $error("Error: op=%b, a=%h, b=%h", op, a, b);
            $display("Expected: %h, Got: %h", expected, result_o);
        end
    end
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

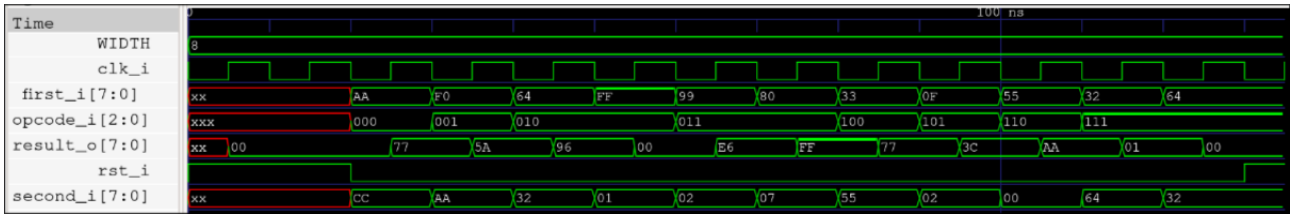

Рис. 2: Временные диаграммы работы модуля

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`