

# Тестовое задание на кафедру МЦСТ

Комиссаров Данил Андреевич

Март 2025

Сейчас мне требуется написать 2 проекта на Verilog: модель определения делимости на 3 и модуль памяти.

Из важных аспектов выделены:

1. Синтезируемость
2. Затраты аппаратных ресурсов
3. Читаемость

Первый видимо означает вообще работоспособность модуля, то есть компилируется ли код или же нет. Второе означает количество транзисторов или же размер на кристалле, который займет этот модуль, то есть следует писать как можно меньше переменных **reg** и использовать как можно меньше комбинационной логики, то есть сокращать количество операторов по возможности. Ну а третье, очевидно, говорит о понятности написанного кода и соответствии названий и форматов входных и выходных пинов.

Вроде более-менее понятно, перейдем к написанию модулей:

## 1 Определитель делимости на 3

Описать на Verilog параметризованный модуль, реализующий следующую функциональность: на выходной порт **divisibility** подаётся 1, если значение данных на входном порте **data** кратно числу 3 (в десятичной системе счисления); в остальных случаях подаётся 0.

Понятно, что придется в любом случае работать с двоичной формой представления числа, иначе было бы удобно просто сложить все цифры и определить делимость этой суммы.

Поэтому вспомним школьный факт, что число в бинарном представлении делится на 3, когда разность суммы битов на четных местах и на нечетных тоже делится на 3.

```

begin
    sum_even = 0; //Сумма битов на четных позициях
    sum_odd = 0; //Сумма битов на нечетных позициях

    even_count = 0;
    odd_count = 0;

    for (i = 0; i < DATA_W; i = i + 1)
    begin
        if (i % 2 == 0)
        begin
            sum_even = sum_even + data[i]; //Здесь вычисляем соответственно сумму на четных
        end
        else
        begin
            sum_odd = sum_odd + data[i]; //А здесь на нечетных
        end
    end
    //debag = ((even_count - sum_even) - (odd_count - sum_odd))%3;
    if ((sum_even - sum_odd)%3==0)
    begin
        divisibility <= 1'b1;
    end
    else
    begin
        divisibility <= 1'b0;
    end
end

```

Здесь реализована проверка на делимость положительных чисел. Предлагаю добавить еще и проверку для отрицательных: для этого нужно просто перевести число из отрицательного представления в положительное, то есть убрать минус в десятичном представлении. Для этого нужно просто инвертировать все биты и добавить единицу к числу и проверить точно так же, как и предыдущее.

```

for (i = 0; i < DATA_W; i = i + 1)
begin
    if (i % 2 == 0)
    begin
        sum_even = sum_even + data[i]; //Здесь вычисляем соответственно сумму на четных
        even_count = even_count + 1;
    end
    else
    begin
        sum_odd = sum_odd + data[i]; //А здесь на нечетных
        odd_count = odd_count + 1;
    end
end
//debag = ((even_count - sum_even) - (odd_count - sum_odd))%3;
if (data[DATA_W-1] == 0 && (sum_even - sum_odd)%3==0 || data[DATA_W-1] == 1 && (((even_count - sum_even) - (odd_count - sum_odd))%3==1))
begin
    divisibility <= 1'b1;
end
else
begin
    divisibility <= 1'b0;
end
end

```

Этим же и занимается вторая половина условия в условном операторе if.

Зачем так сложно инвертировать биты: то есть сначала подсчитывать все четные, а потом из них вычитать только те, в которых есть единица, а не, например, просто вычитать из **DATA\_W / 2**? Написано это с расчетом на то, что параметр **DATA\_W** может быть любым, в том числе и нечетным; в этом случае не будет работать первый метод.

Вроде все очевидно, проверим на тестбенче, в котором в том числе надо написать и проверку на отрицательные числа.