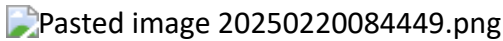
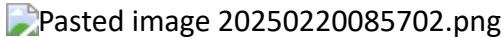


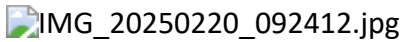
Семы таки Попов
Первая лаба - конвейерные устройства. Лаба на конвейризацию сумматора ради максимальной частоты работы, что бы это ни значило
Сегодня - подготовка к 1 лабе
Язык Verilog yippee
Работаем на Xilinx Virtex6, их 3 штуки на всех))


Imao
ну это лучше чем это V // По факту


Суть - описать некий сумматор, а потом будем дорабатывать его до максимальных скоростей

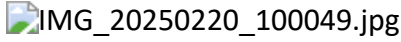
```
module n1#(parameter width = 8) //Width - ширина шины входа
    (input[width-1:0] a, b,
     output[width-1:0] out);
    assign out = a + b;
endmodule

/*
width <= 6:
    sumi = a[i] ^ b[i] ^ cin[i-1]
    cout = a[i]b[i] | a[i]cin[i-1] | b[i]c[i-1]
width > 6:
        cin    cout
1) 0 1    0    0    -> cout = cin;
    1 0    1    1
2) 1 1
    0 0
    -> cout = a;
*/
```


Один логический блок ПЛИС ^
Мы нарисовали на XOR, на других ПЛИС иногда это работает на LUT (LookUp Table) (слева снизу нарисован) - шестивходовый модуль, в который вшита таблица истинности, буквально как тот мужик с книгой иероглифов.
Дописываем код сумматора

```
module n1#(parameter width = 8) (
    input[width-1:0] a, b,
    input cin,
    output[width-1:0] out,
    output cout
);
wire 🗑; //Буквально мусорная переменная лмао
//assign out = a + b;
//assign {cout, out} = {0, a, 0} + {0, b};
assign {cout, out, 🗑} = {0, a, cin} + {0, b, cin};
/*
Суть cout - вынести один лишний бит при переполнении суммы
Принцип работы cout - добавить к обоим операндам
по нулю в начале и вынести cout в начало out,
так что если сумма результирует с битом переноса -
он попадёт в cout
*/
endmodule
```

"MAC адрес шиз"
©Попов 2025

Задача - наебашить стенд, который позволяет протестировать максимальную частоту

MMCM - генератор clk
Вход CLKFB - Click FeedBack - нужен для проверки сдвига по фазе
BUFG - буфер, дерево сигнала
G - модуль, выдающий 2 значения и эталонное значение
Дальше - сумматор и компаратор, проверяющий правильность вычисления
Если хоть раз отловит сигнал об ошибке - LED перманентно загорится (код ниже)

```
always @(posedge clk)
    if (error) led = 1;
```