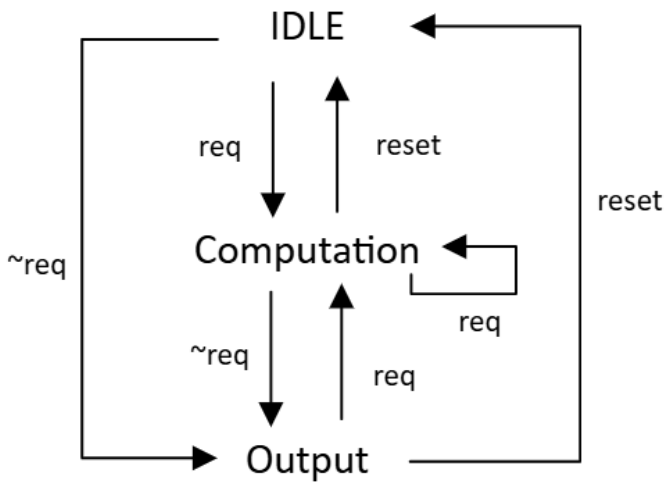
To build the diagram first thing first identify the states of the device, in this case there is 3 of them: IDLE, Computation and Output. Using a protocol identifiy how device is transiting between the states.



Writing a code
Introduce an internal registers. It is important to note that x_prime is present on the diagram, but not in the input variables, however it still will be used in further computations until the reset signal received according to the protocol. And as well we need to store a result, as during the process it will be assigned continuously, which is not possible to do with a wire in the input.

```
reg [3:0] res;

reg [3:0] x_prime;
```

Next step is to describe the start of the process, which will always be positive reset signal. On reset signal we are setting x_prime to user input x and we need to assign re to x as well, so if no computation happens and request signal haven't been sent, the device still shows an output. If you will look carefully this matches description of IDLE state.

```
always @(posedge reset)begin

    x_prime = x;

    res = x;

end
```

According to the protocol computation happens on the positive edge of clock.

```
always @(posedge clk) begin
    if(req)begin
        case (opcode)
            2'b00: res = x_prime & y;
            2'b01: res = ~(x_prime & y);
            2'b10: res = ~(x_prime | y);
            2'b11: res = x_prime^y;
        endcase
    end
end
```

As until reset signal received we keep using result of previous computation, therefore we need to reassign the x_prime to result, and assign to a output wire result as well.

```
always @(negedge req)begin
    x_prime = res;
end
assign result = res;
```