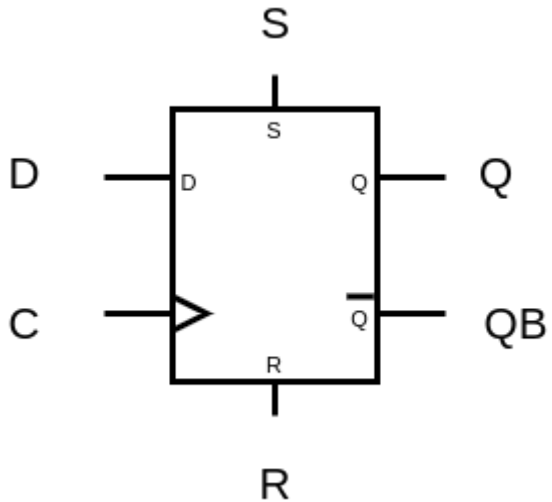
Простые последовательностые схемы

Дмитрий Смехов

HDL, RTL and FPGA: Part 2

https://bit.ly/2022-08-02-verilog-2-bishkek-yuri-panchul

D-триггер

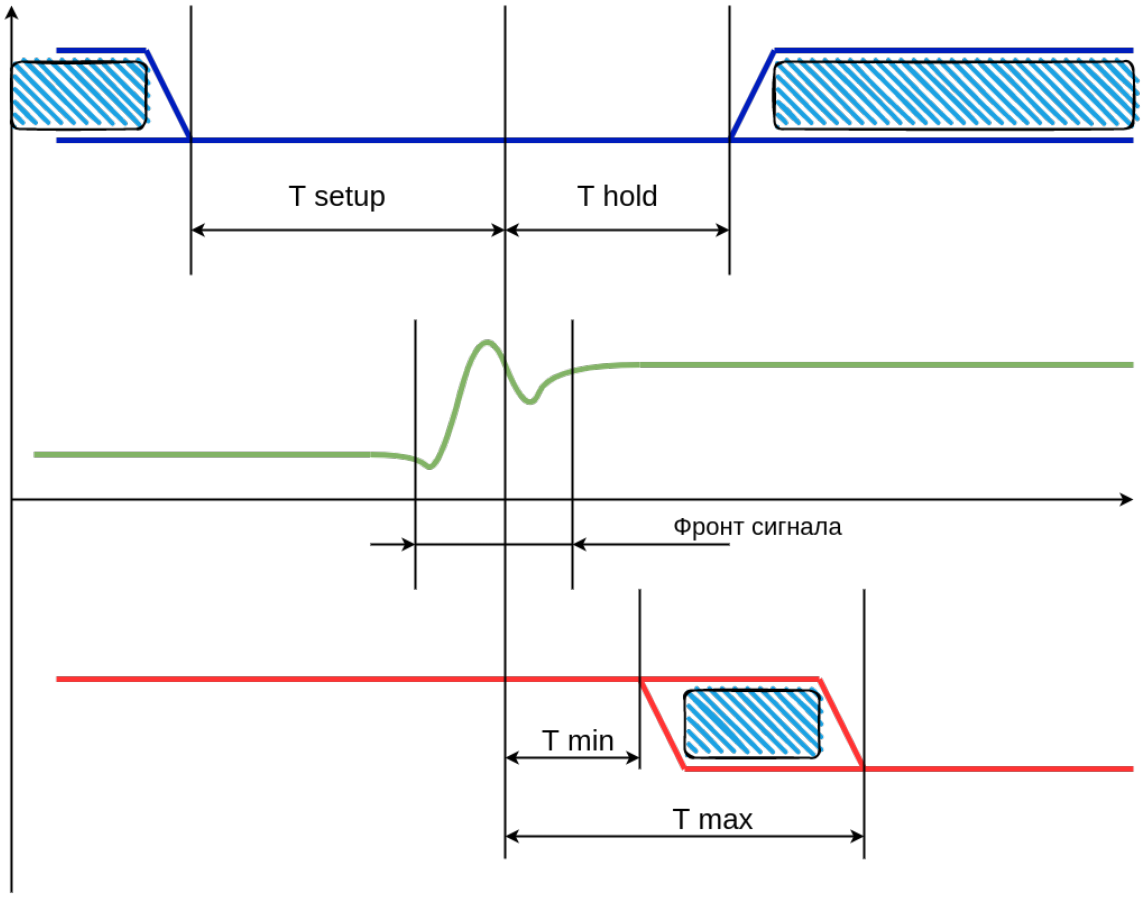


Логическое И

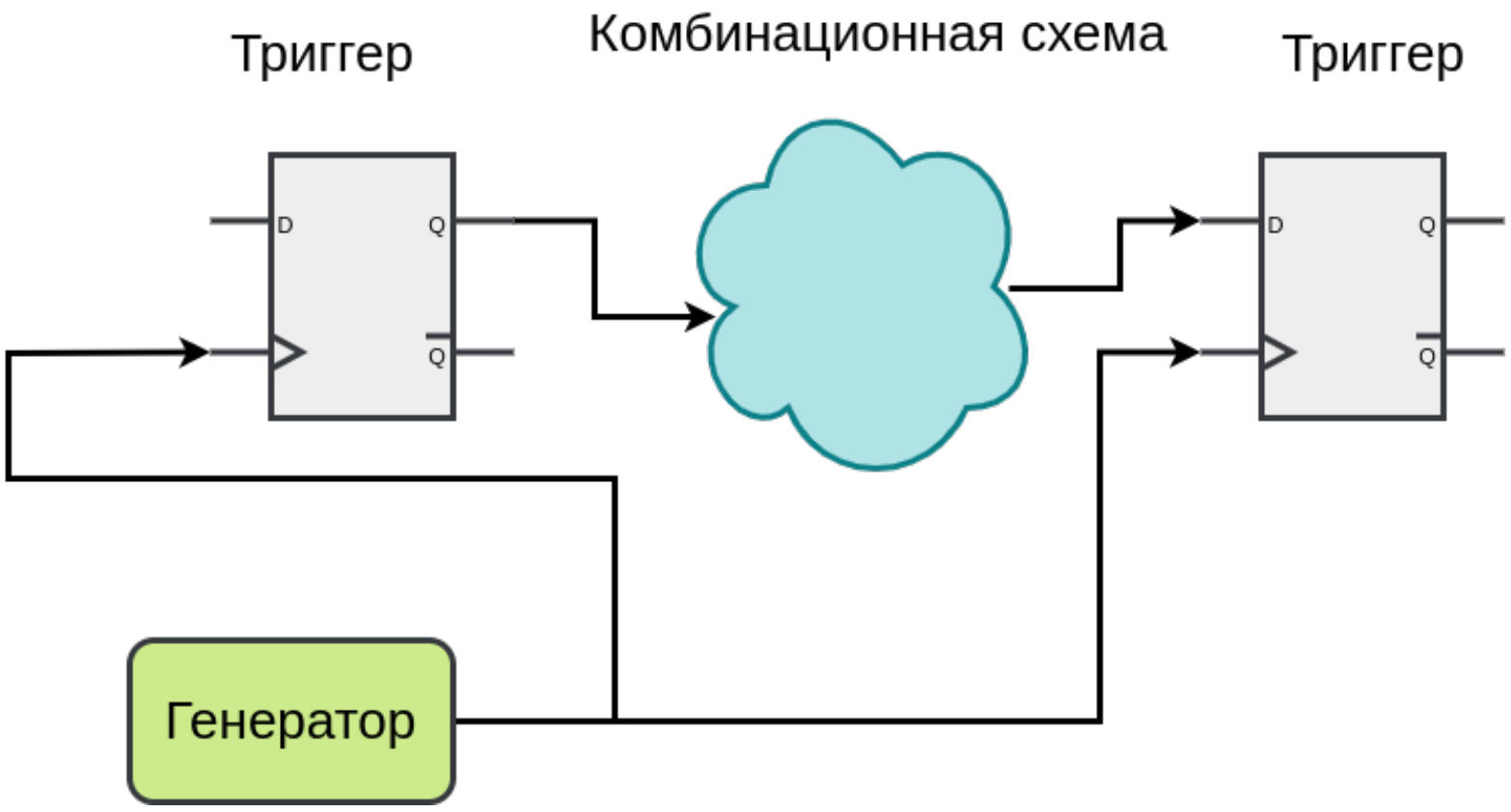
D	S	R	C	Q	QB
X	X	1	X	0	1
X	1	0	X	1	0
0	0	0	R	0	1
1	0	0	R	1	0

Фронт сигнала.
Переход из 0 в 1.

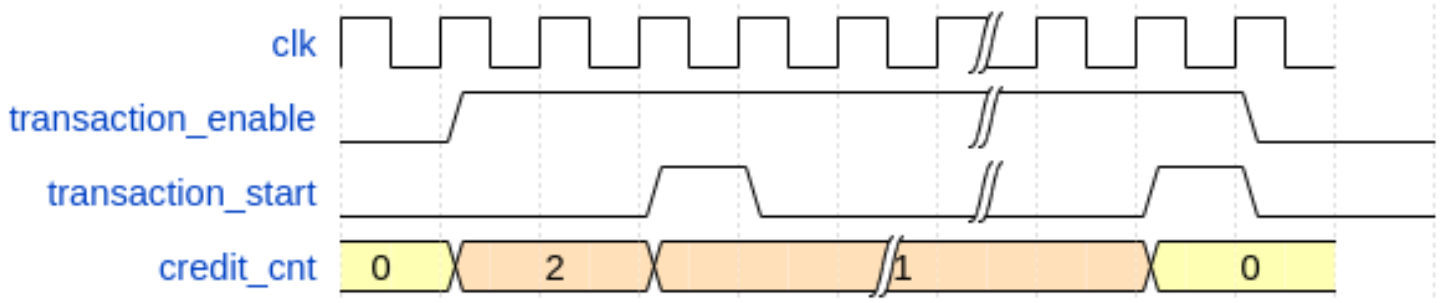
D-триггер — фиксация данных



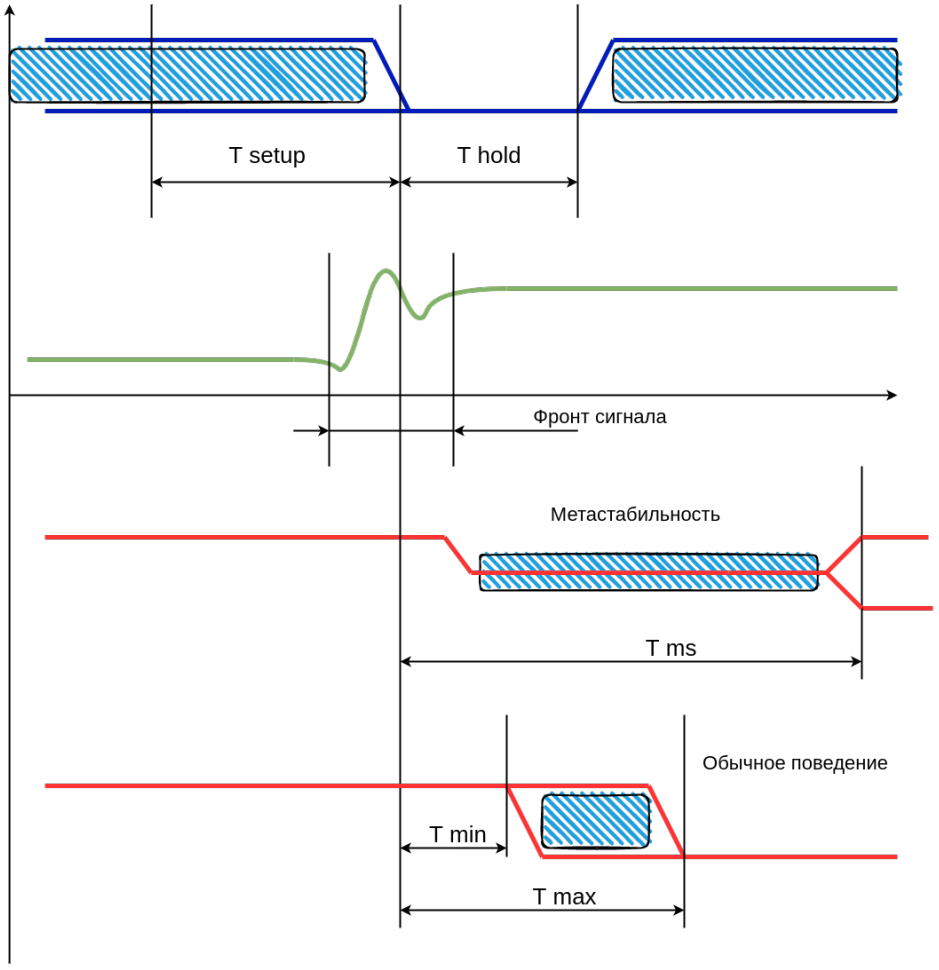
Синхронные схемы



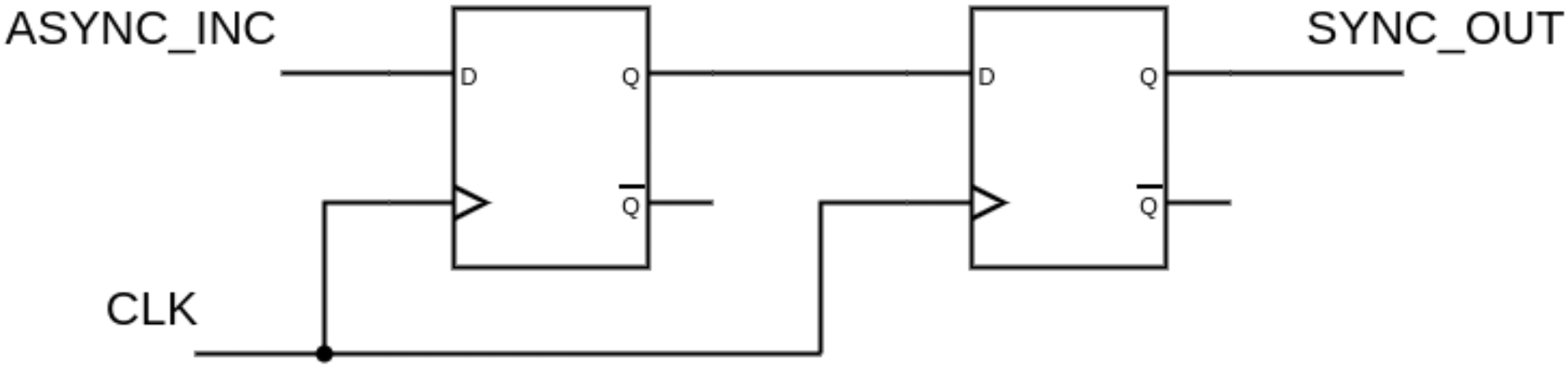
Временная диаграмма



Метастабильность



Синхронизация

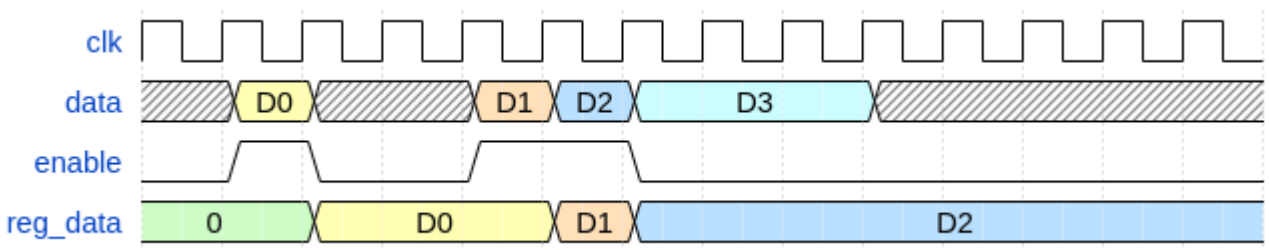


Некоторые элементы

- Регистр
- Сдвиговый регистр
- Счётчик
- Конечный автомат
- FIFO
- Кредитный счётчик

Параллельный регистр

```
module(
        input  wire                 clk,
        input  wire                 rstp,
        input  wire                 enable,
        input   wire [31:0]         data,
        output wire [31:0]          reg_data_o
);
logic  [31:0]           reg_data;

always_ff @(posedge clk) begin

        if( rstp )
                reg_data <= #1 32'h0;
        else if( enable )
                reg_data <= #1 data;
end

assign reg_data_o = reg_data;

endmodule
```



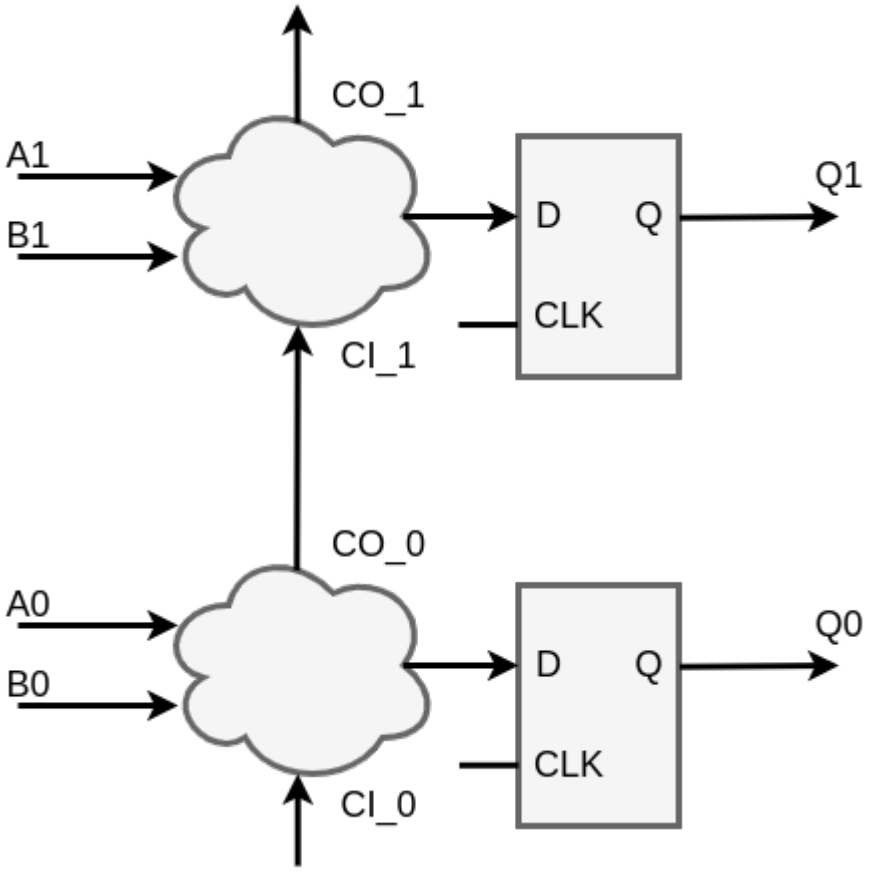
Счётчик

```
module(
        input  wire                 clk,
        input  wire                 rstp,
        input  wire                 cnt_en,
        output wire [31:0]          cnt_o
);

logic  [31:0]           cnt;

always_ff @(posedge clk) begin

        if( rstp )
                cnt <= #1 32'h0;
        else if( cnt_en )
                cnt <= #1 cnt+1;
end

assign cnt_o = cnt;

endmodule
```

Реализация счётчика

Сумматор

CI	A	B	CO	Q
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

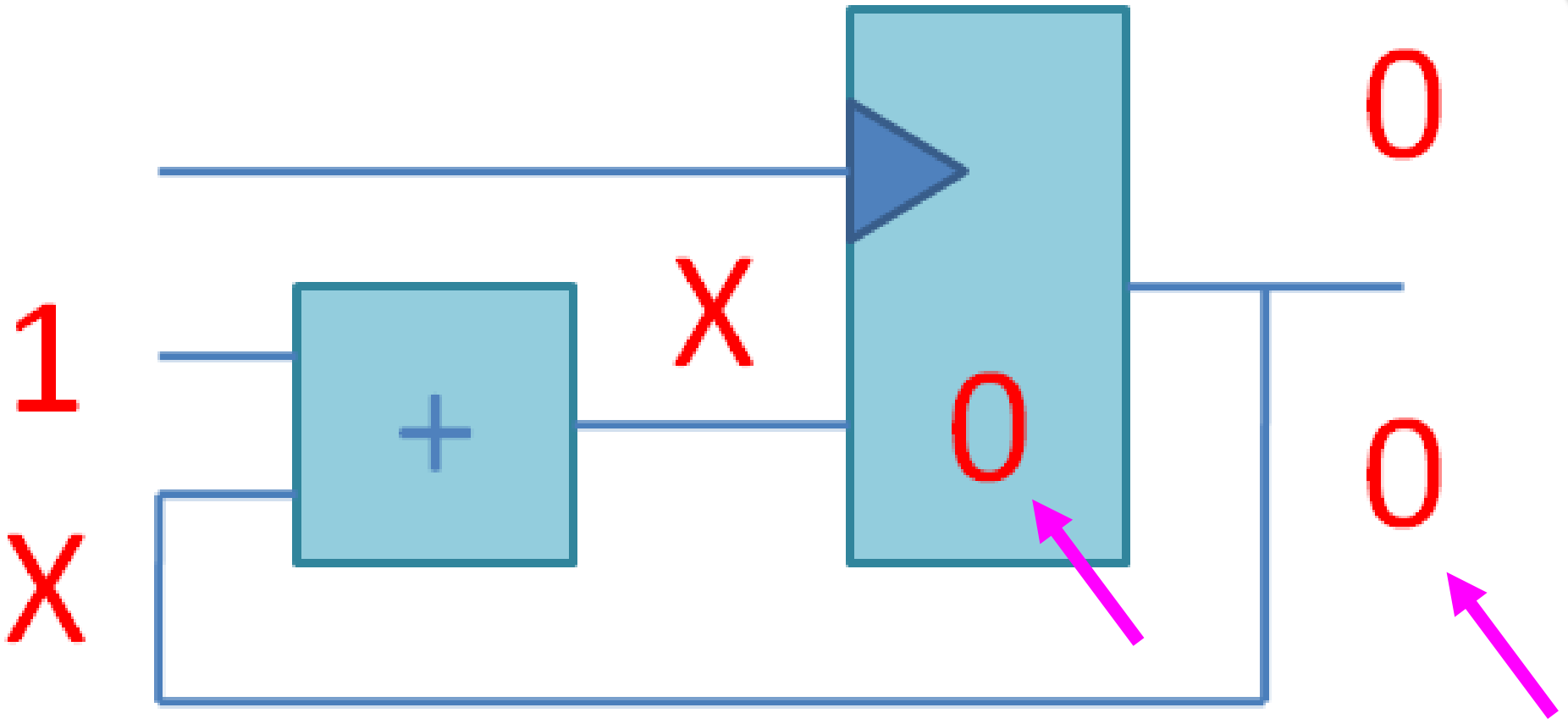


Adder + Register = Counter

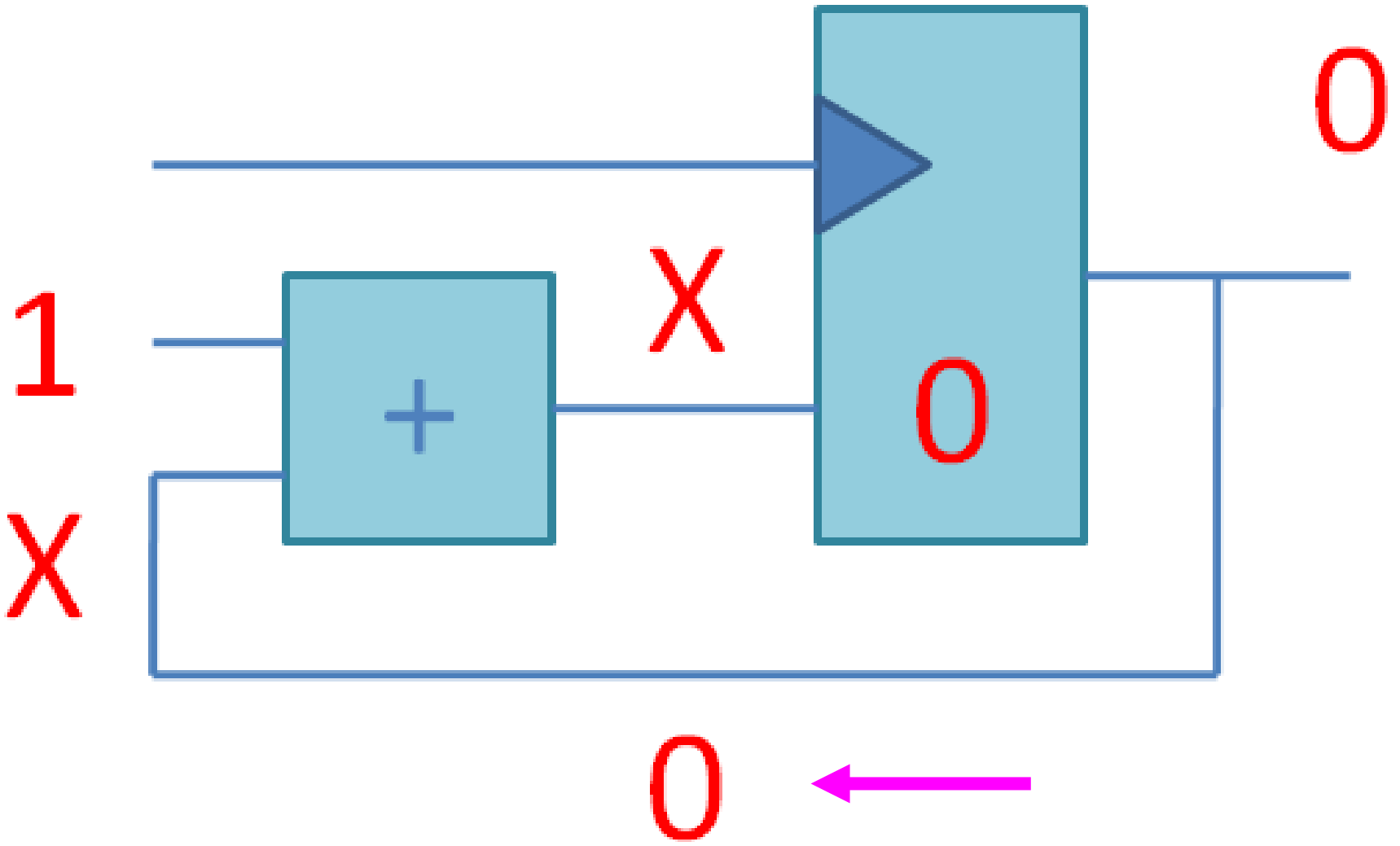
- A group of D-flip-flops is called a register.
- Do not confuse with Verilog reg or CPU registers in programming.
- In this code the result of addition is stored to use in the next clock cycle.

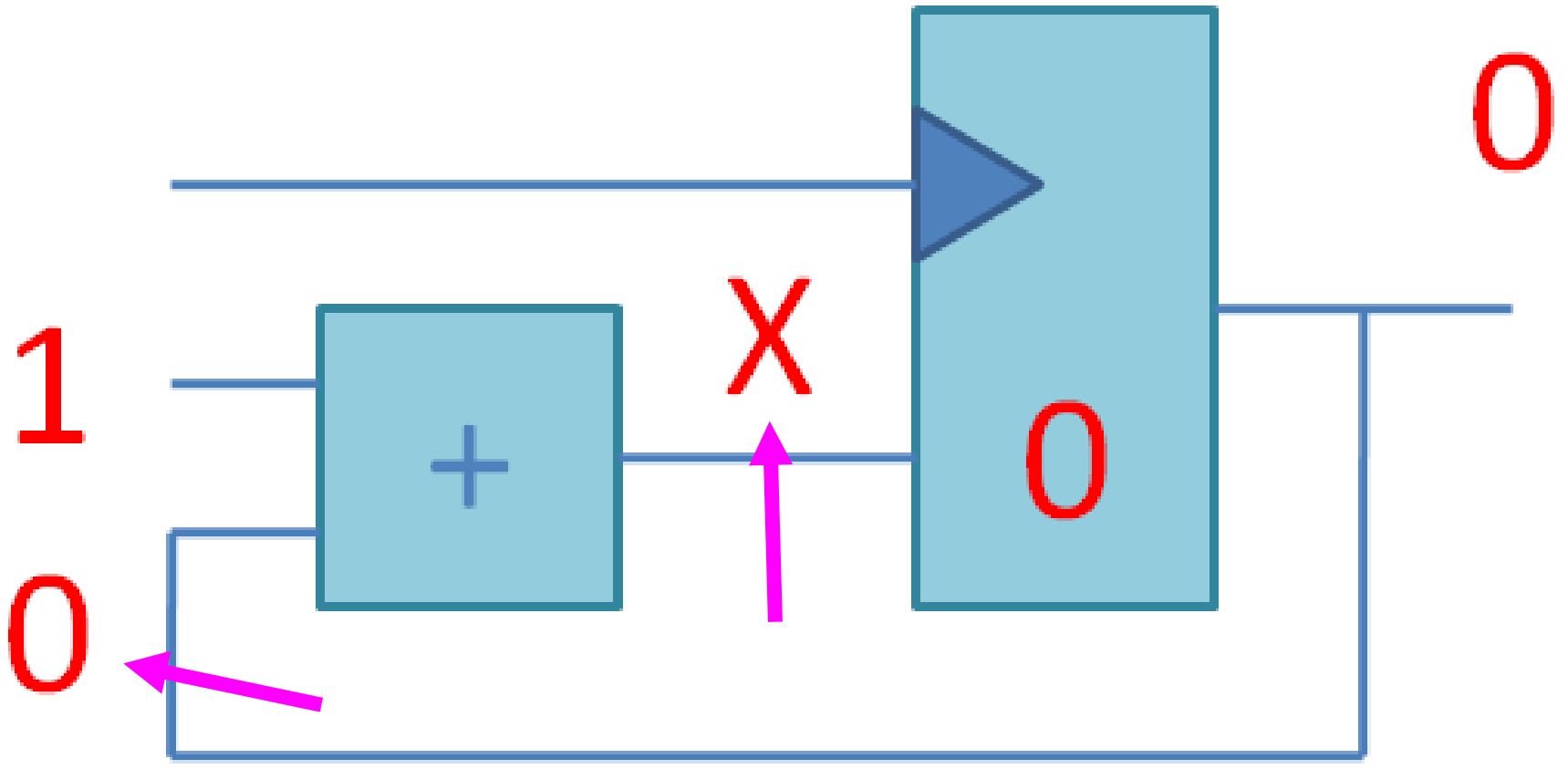
```
module counter
(
    input              clock,
    input              reset_n,
    output reg [31:0] count
);

    always @(posedge clock or negedge reset_n)
    begin
        if (!reset_n)
            count <= 32'b0;
        else
            count <= count + 32'b1;
    end
```

Source https://github.com/yuri-panchul/2017-tomsk-novosibirsk-astana/blob/master/pre_event_drafts/terasic_de10_lite/08_counter/top.v

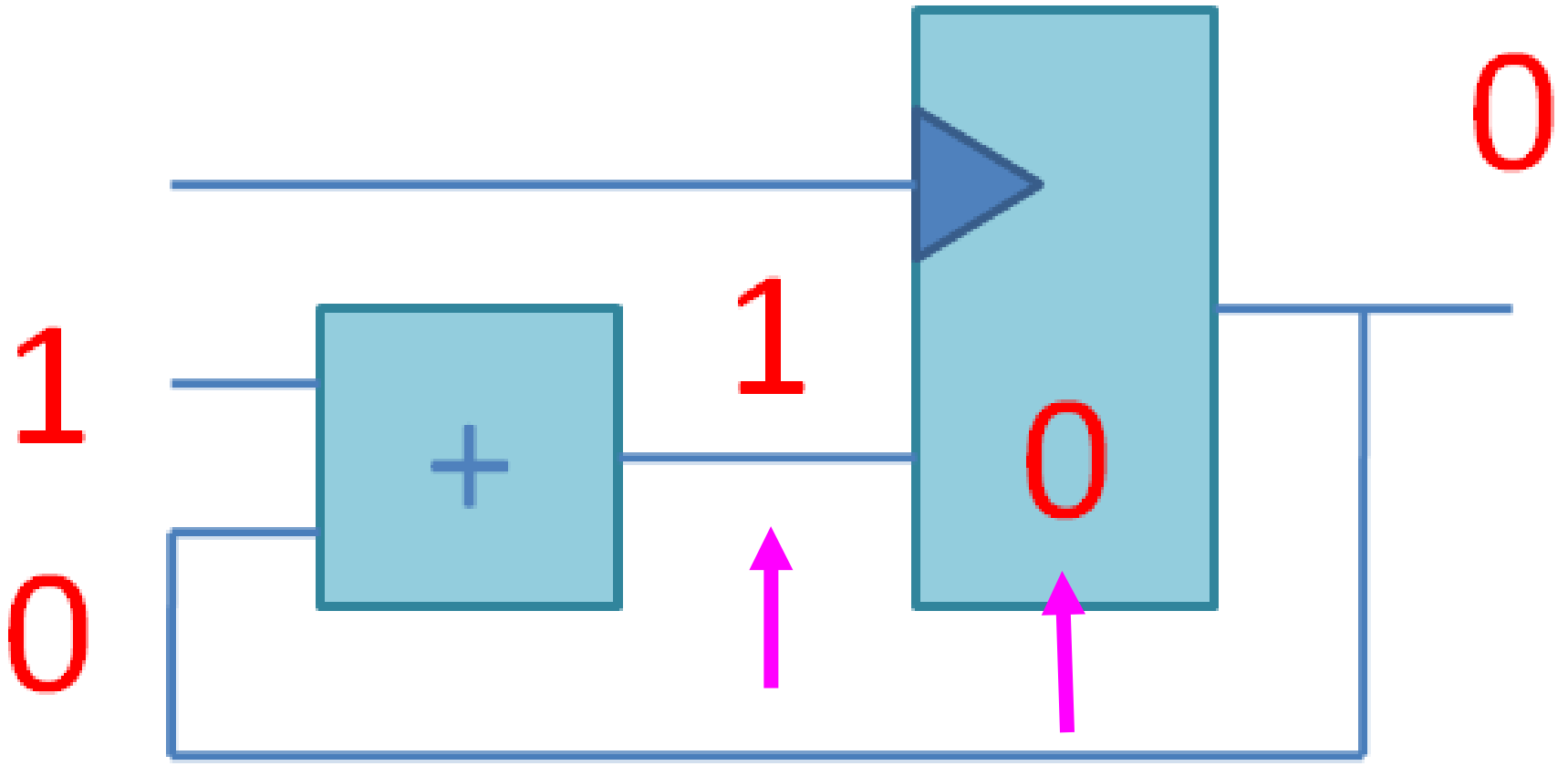


The register contains 0, it gets propagated to the adder.

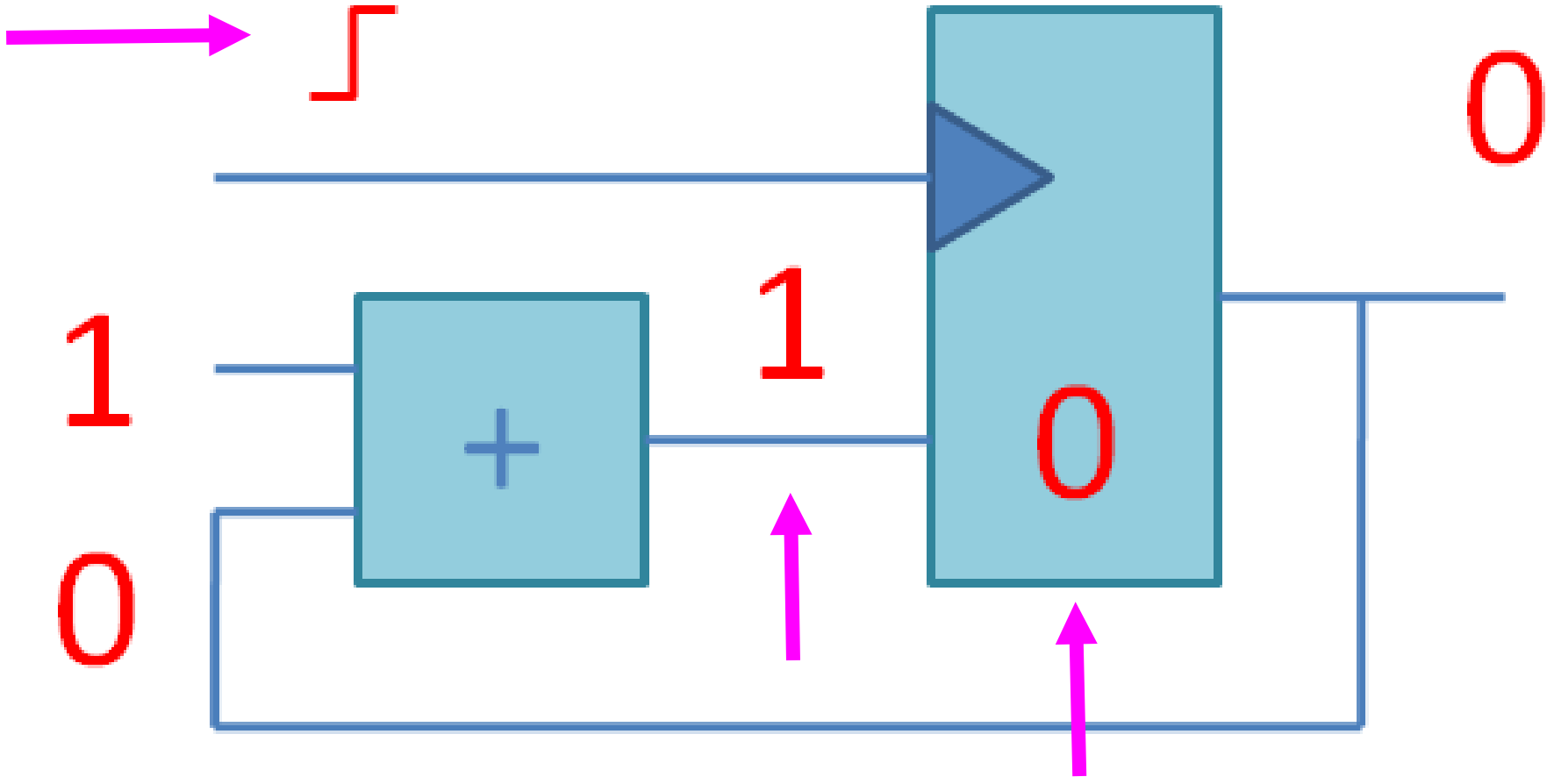

0
1
X
+
0
X
0



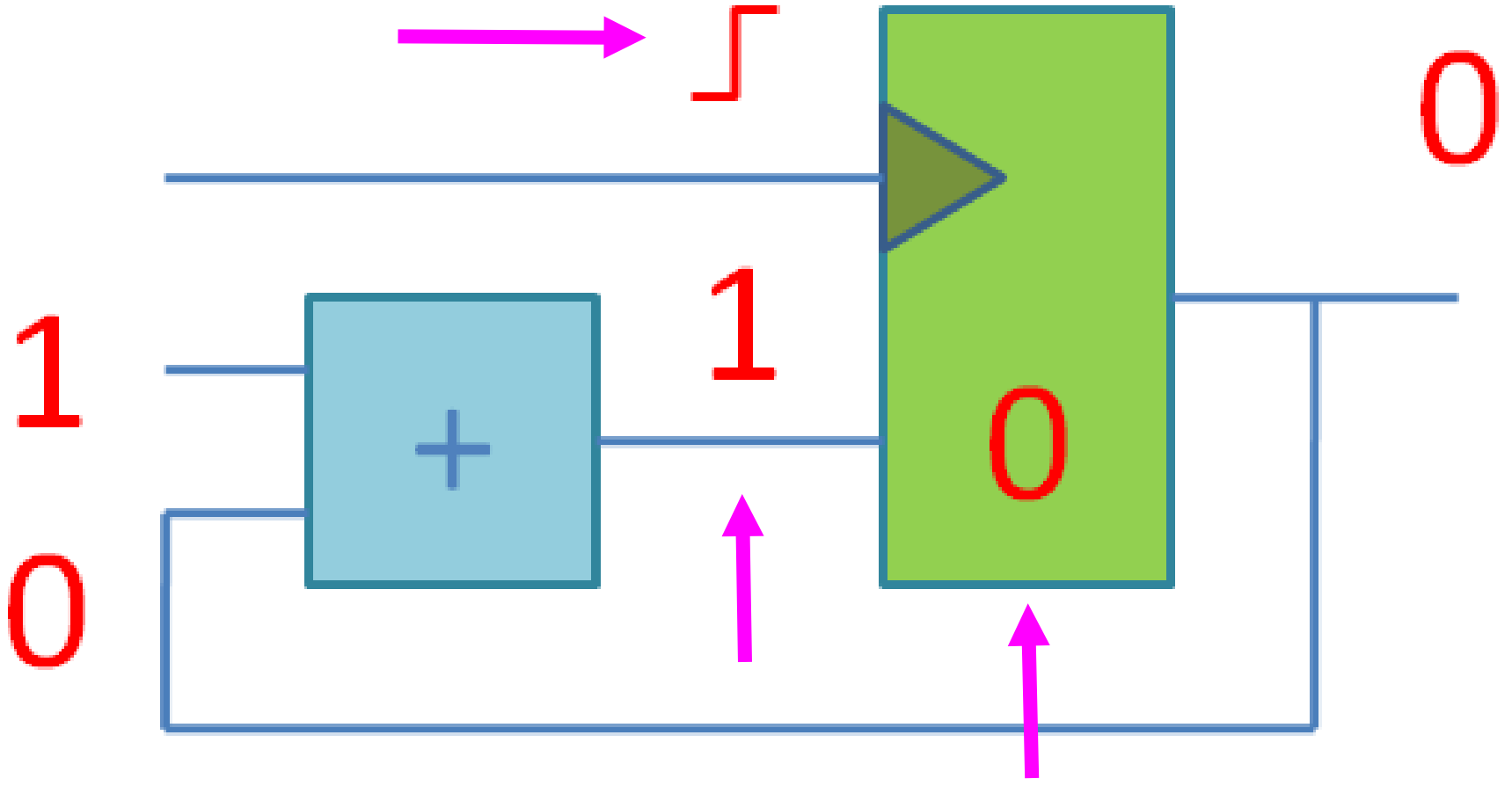
0 enters the adder. The adder’s output is not stable yet.



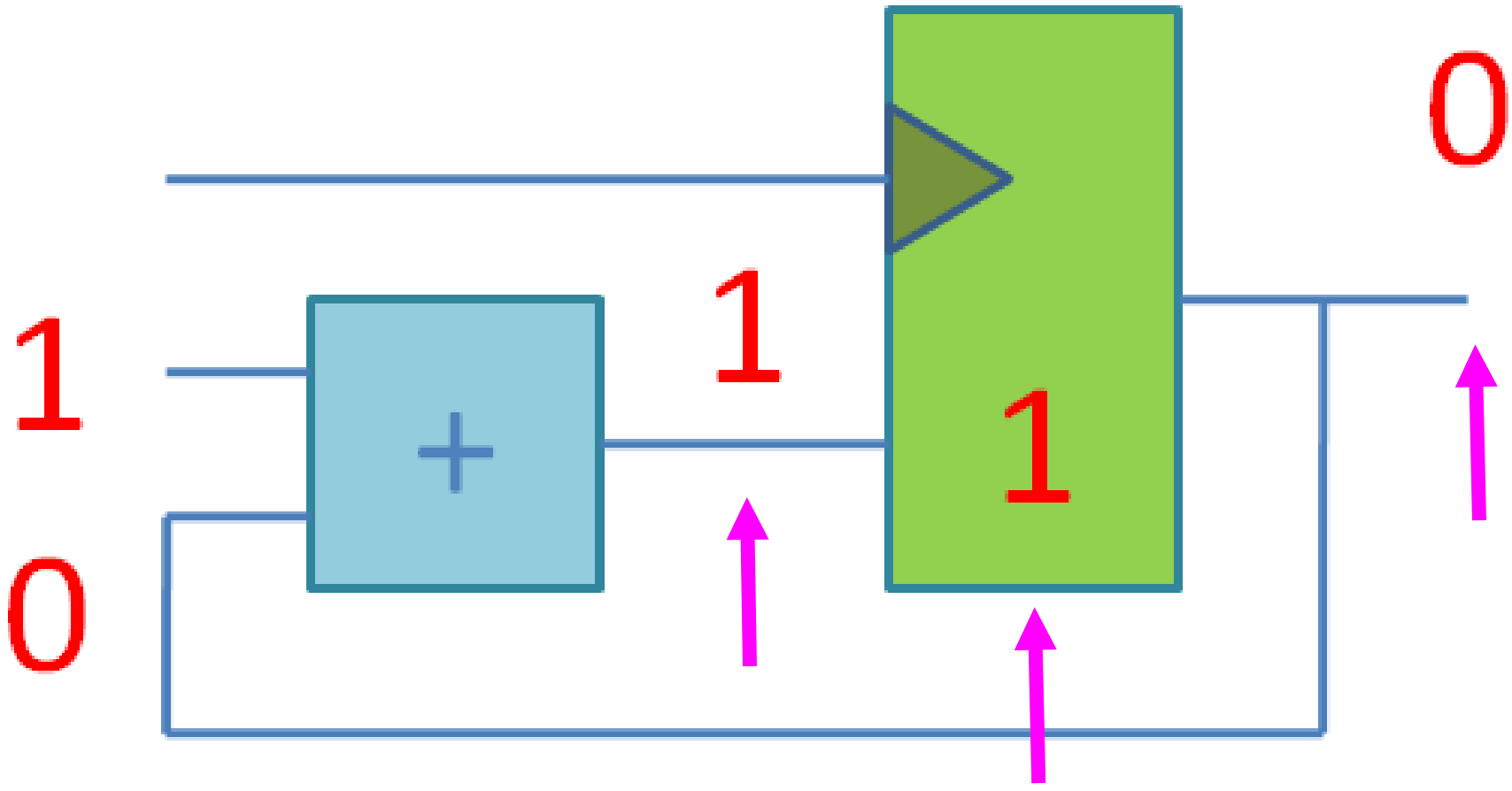
The adder computed 0 + 1 = 1. Register still contains 0.



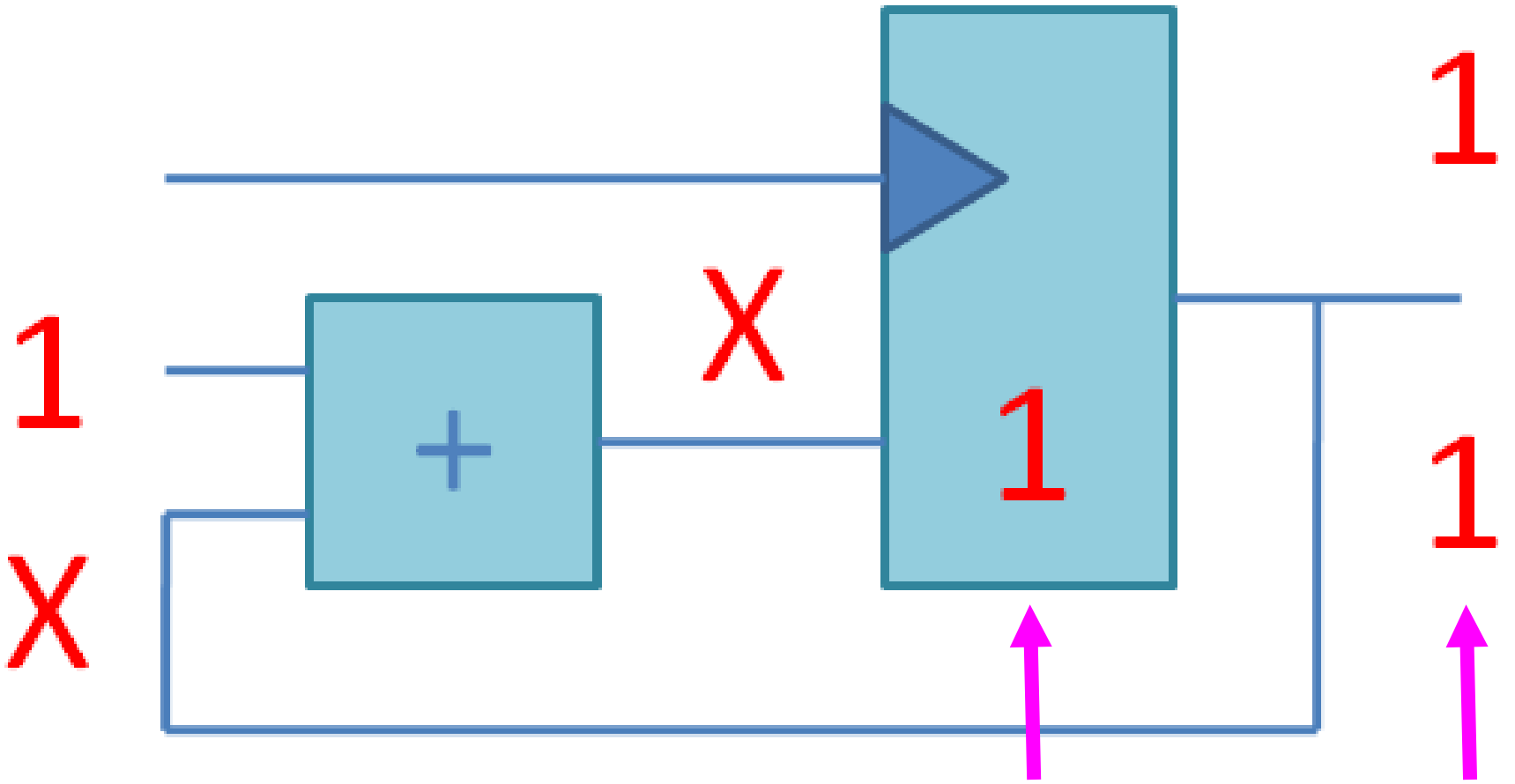
Positive edge of the clock is coming. Register is still 0.



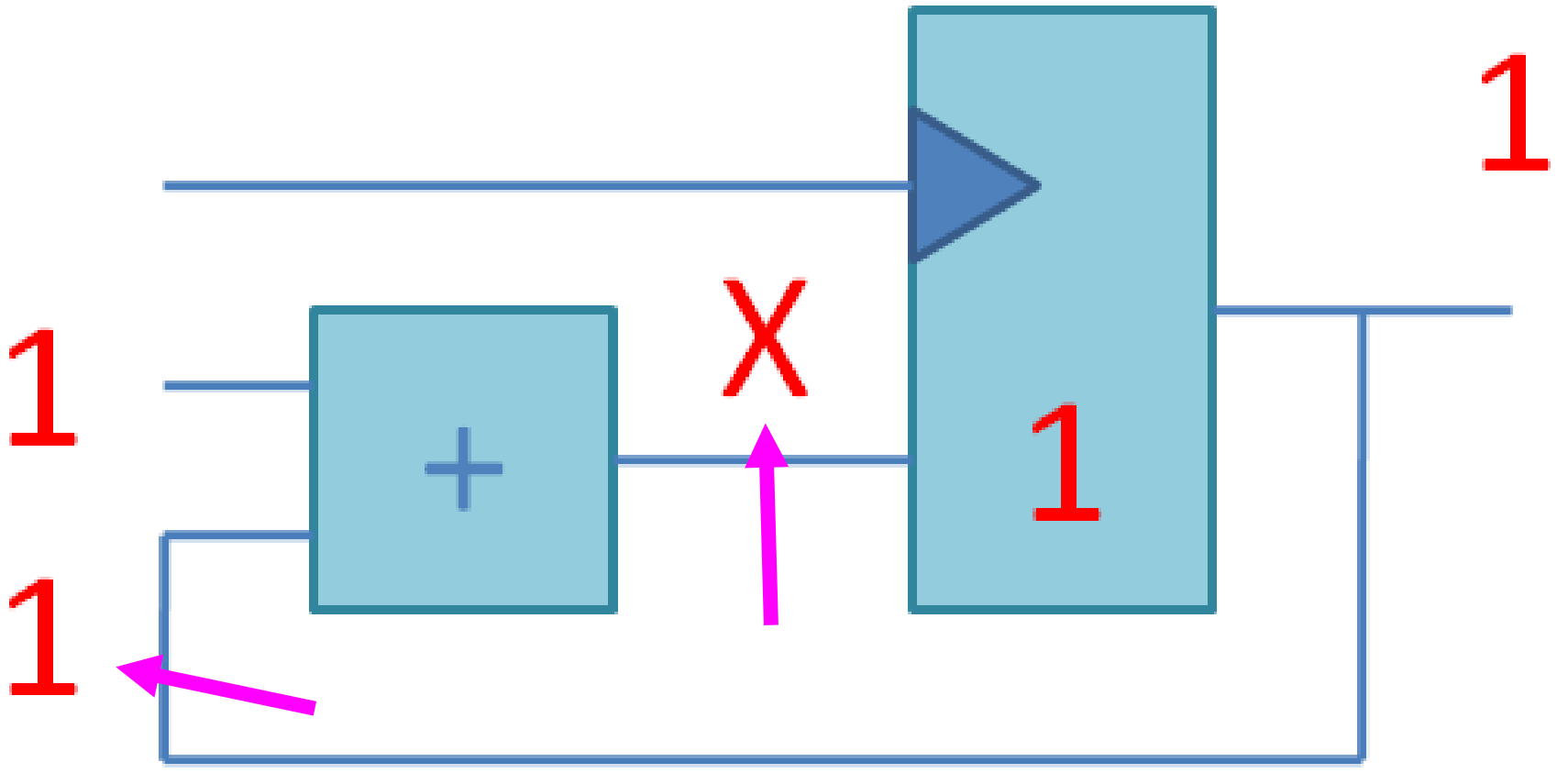
The aperture time. Register is about to store 1.



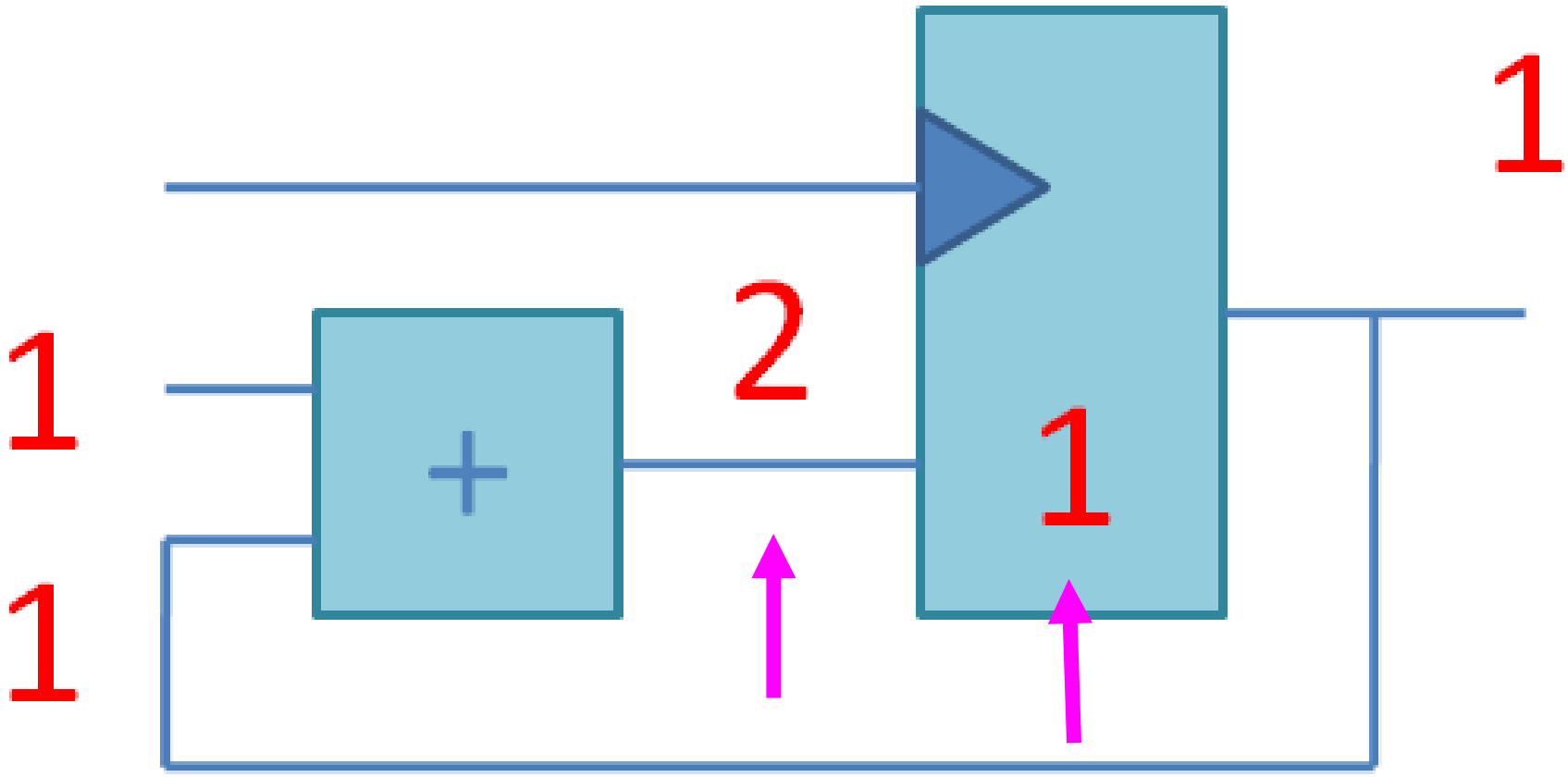
The register recorded 1 and is about to propagate it outside.



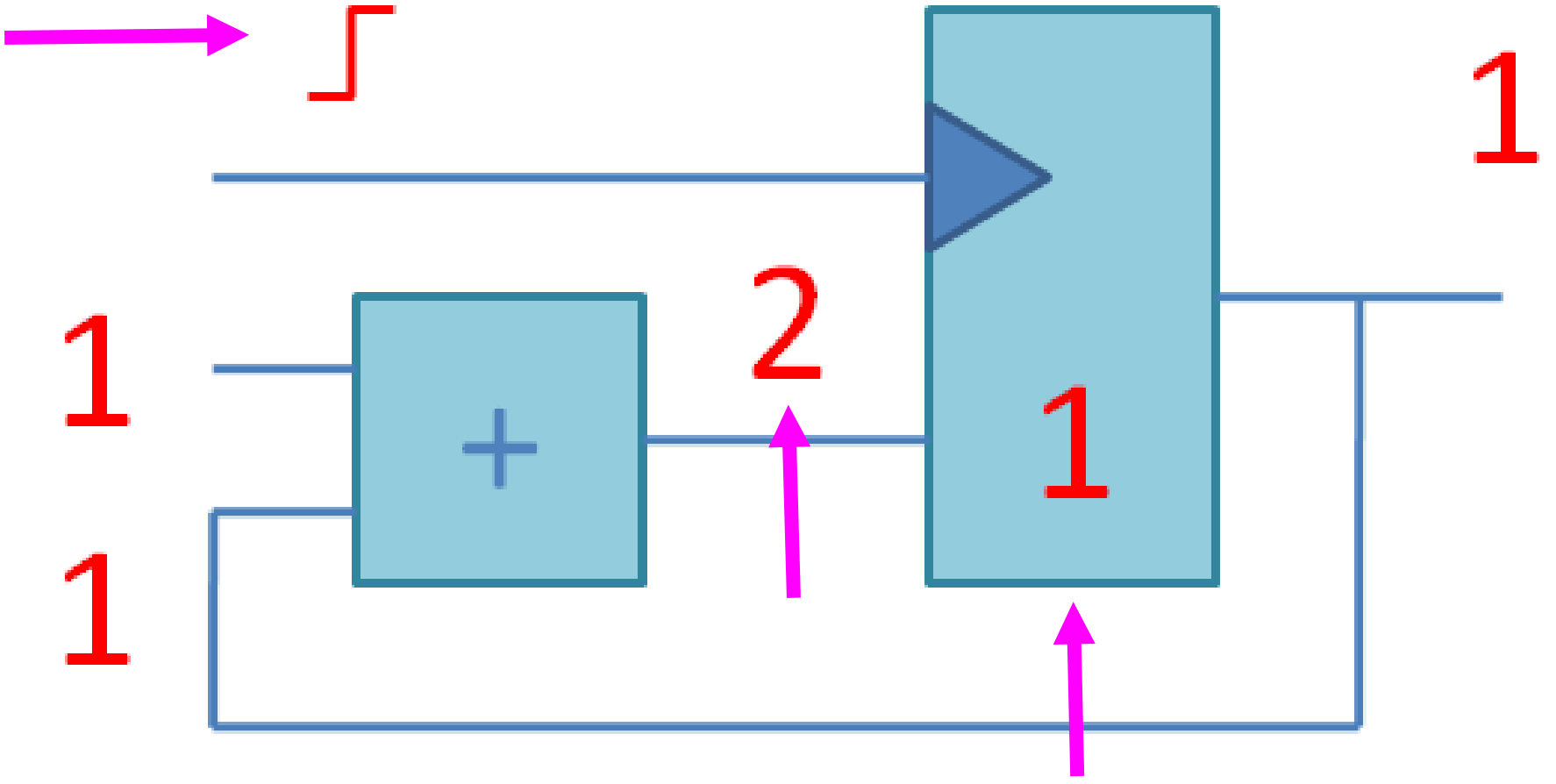
The current state is 1, it gets propagated to the adder.



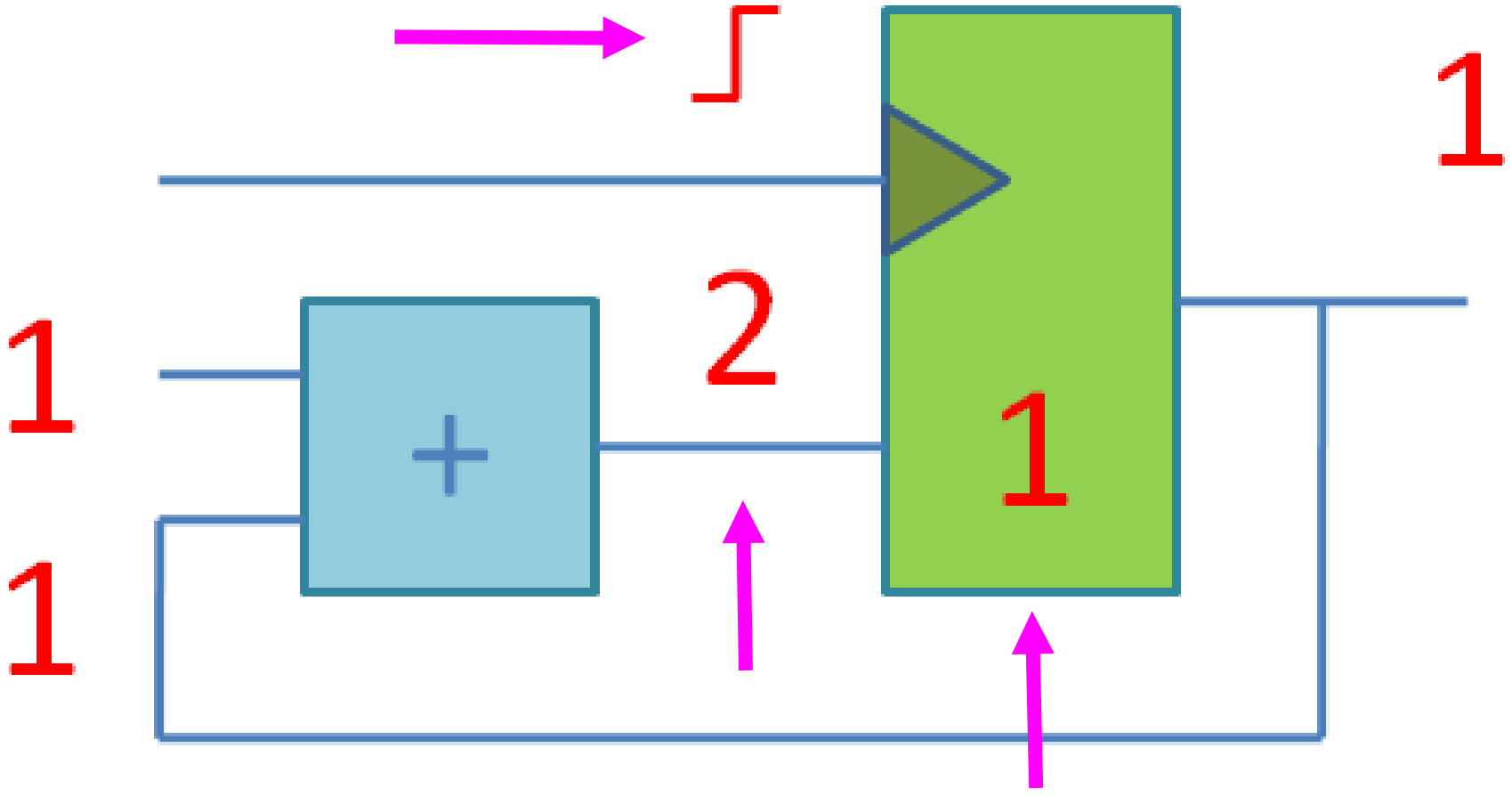
1 enters the adder. The adder’s output is not stable yet.



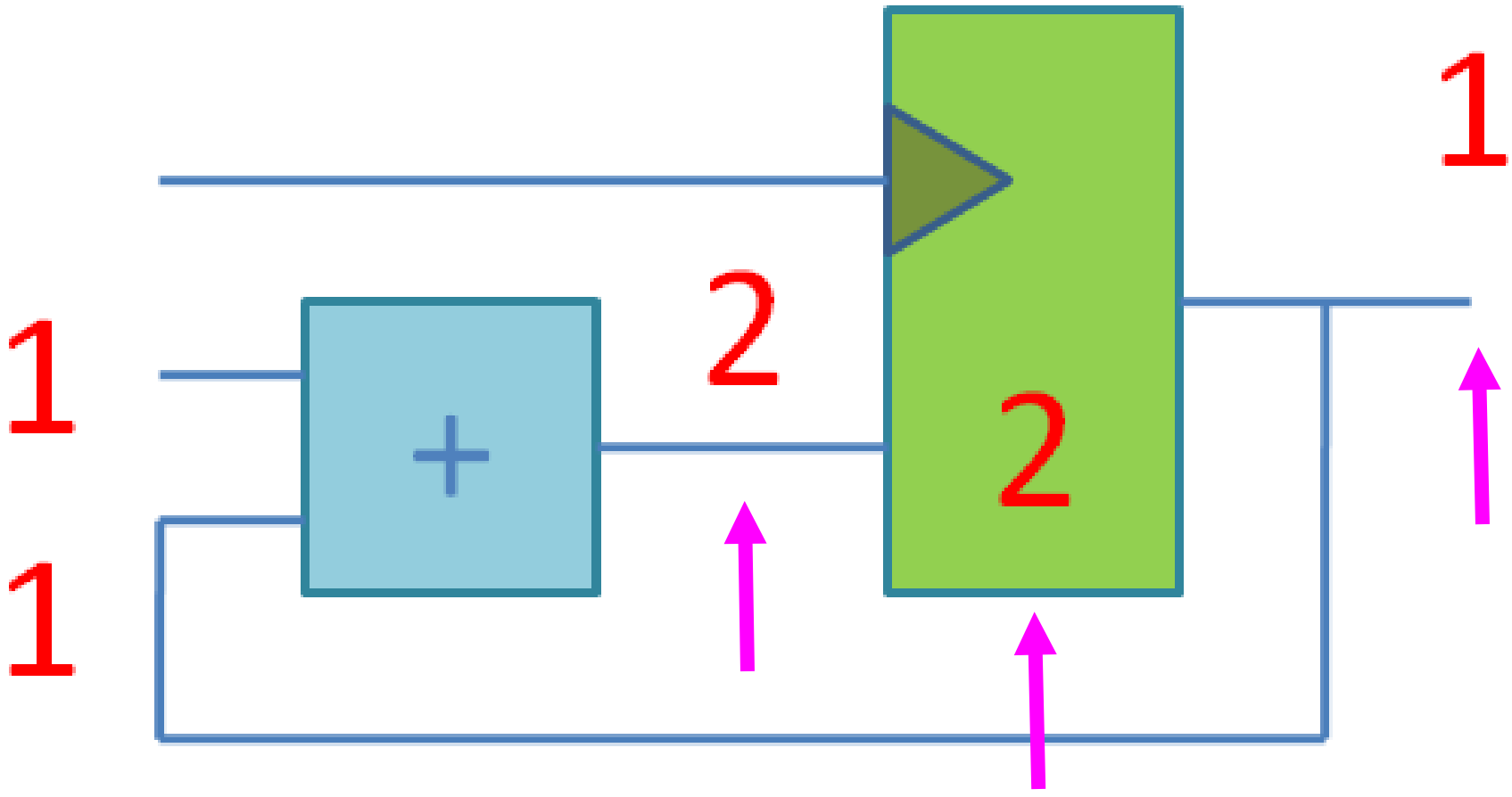
The adder computed 1 + 1 = 2. The register still contains 1.



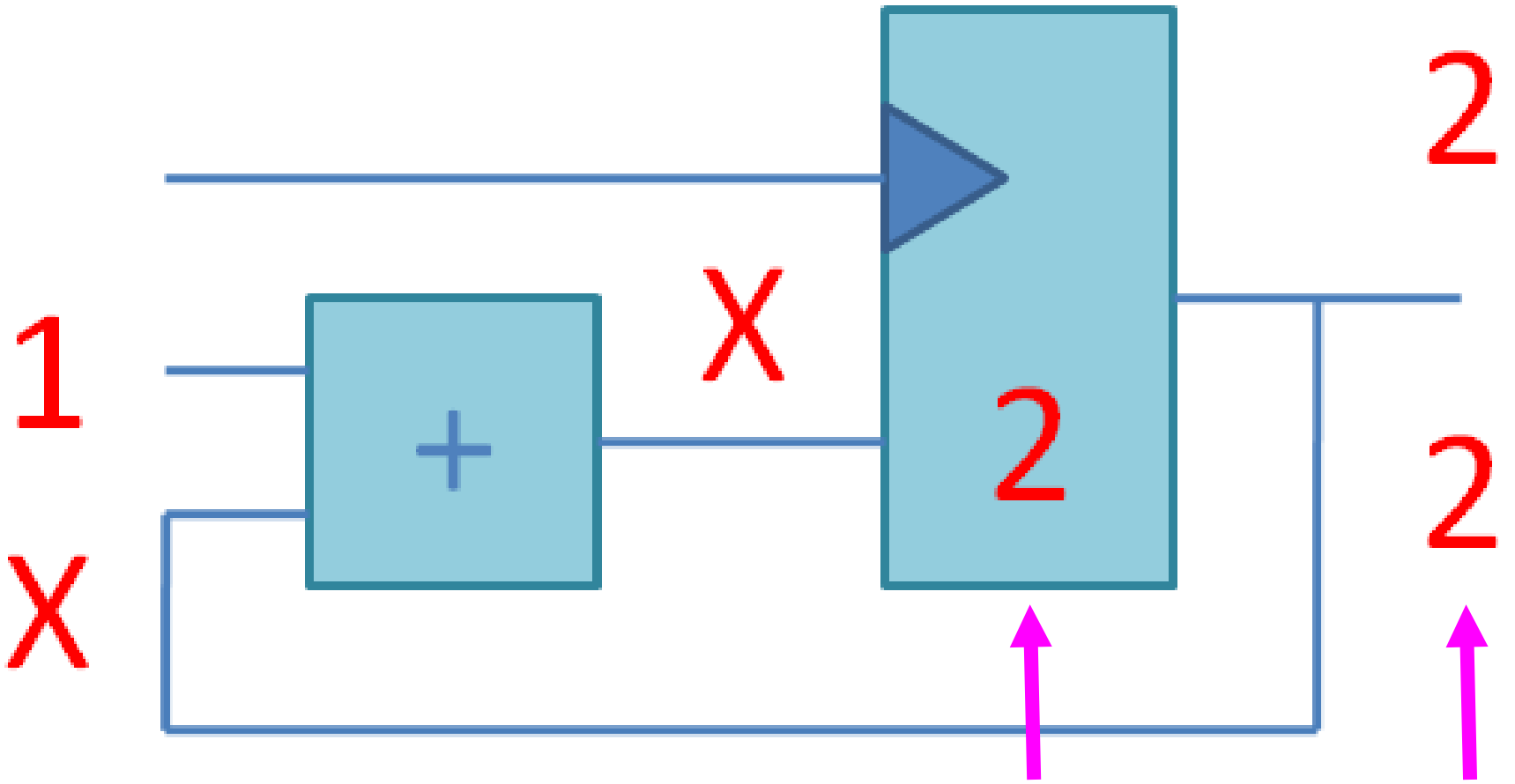
A positive edge of the clock is coming. The register is still 1.



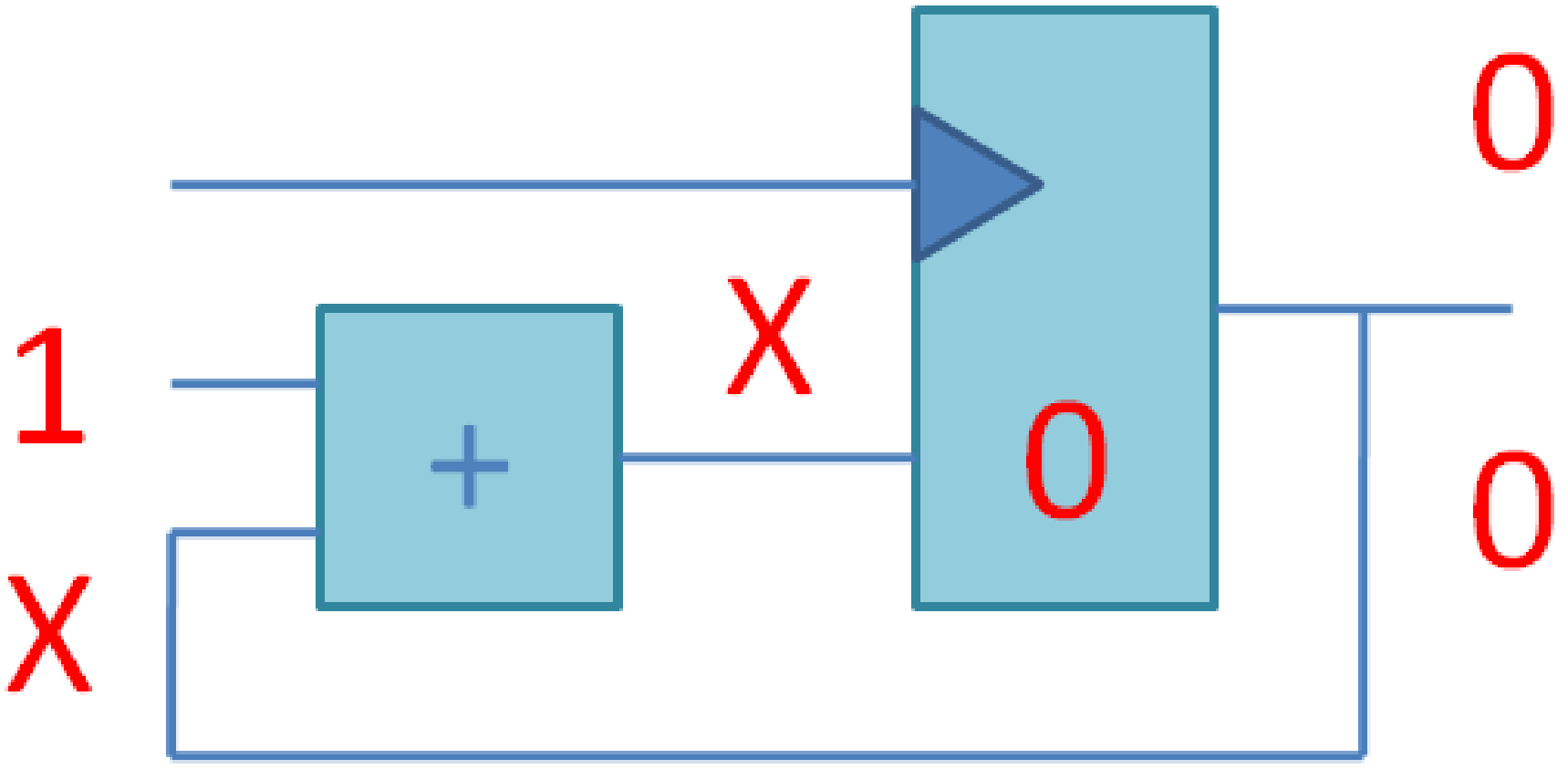
The aperture time. The register is about to store 2.



The register recorded 2 and is about to propagate it outside.



The current state is 2, it gets propagated to the adder.


0
1
X
+
0
0
X

Сдвиговый регистр

```
module(
        input  wire                 clk,
        input  wire                 rstp,
        input  wire                 shift_en,
        output wire [3:0]           reg_data_o
);
logic  [3:0]            reg_data;

always_ff @(posedge clk) begin

        if( rstp )
                reg_data <= #1 4'b0001;
        else if( shift_en ) begin
                reg_data[0]    <= #1 reg_data[3];
                reg_data[3:1] <= #1 reg_data[2:0];
        end
end

assign reg_data_o = reg_data;
endmodule
```