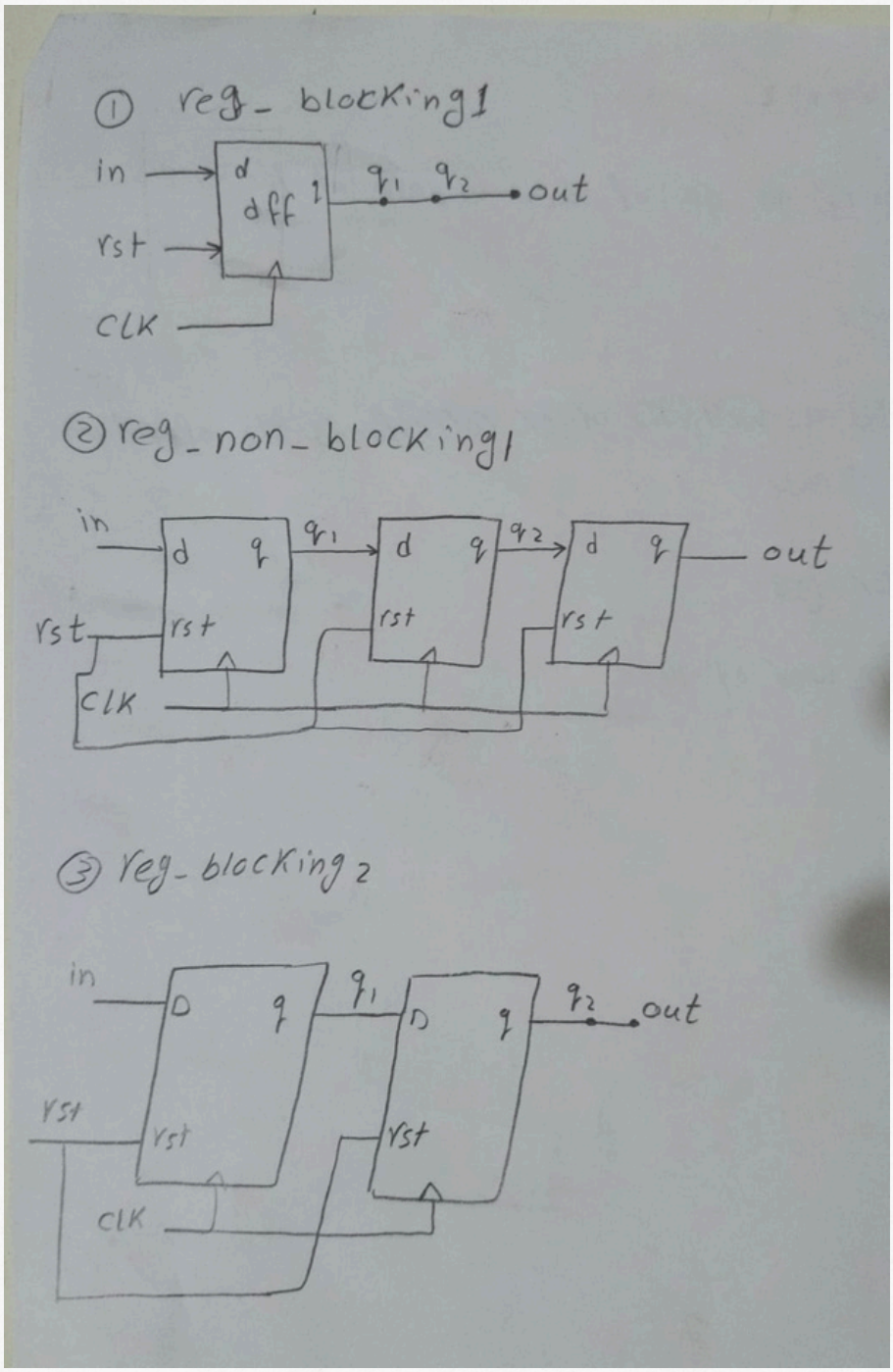
يوسف أحمد محمد ابراهيم

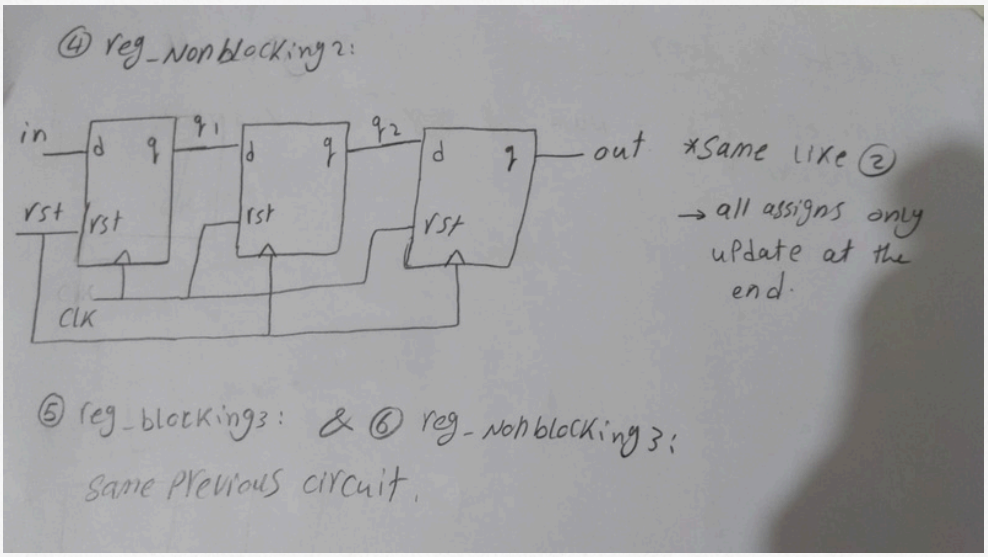
Assignment 3

Extra

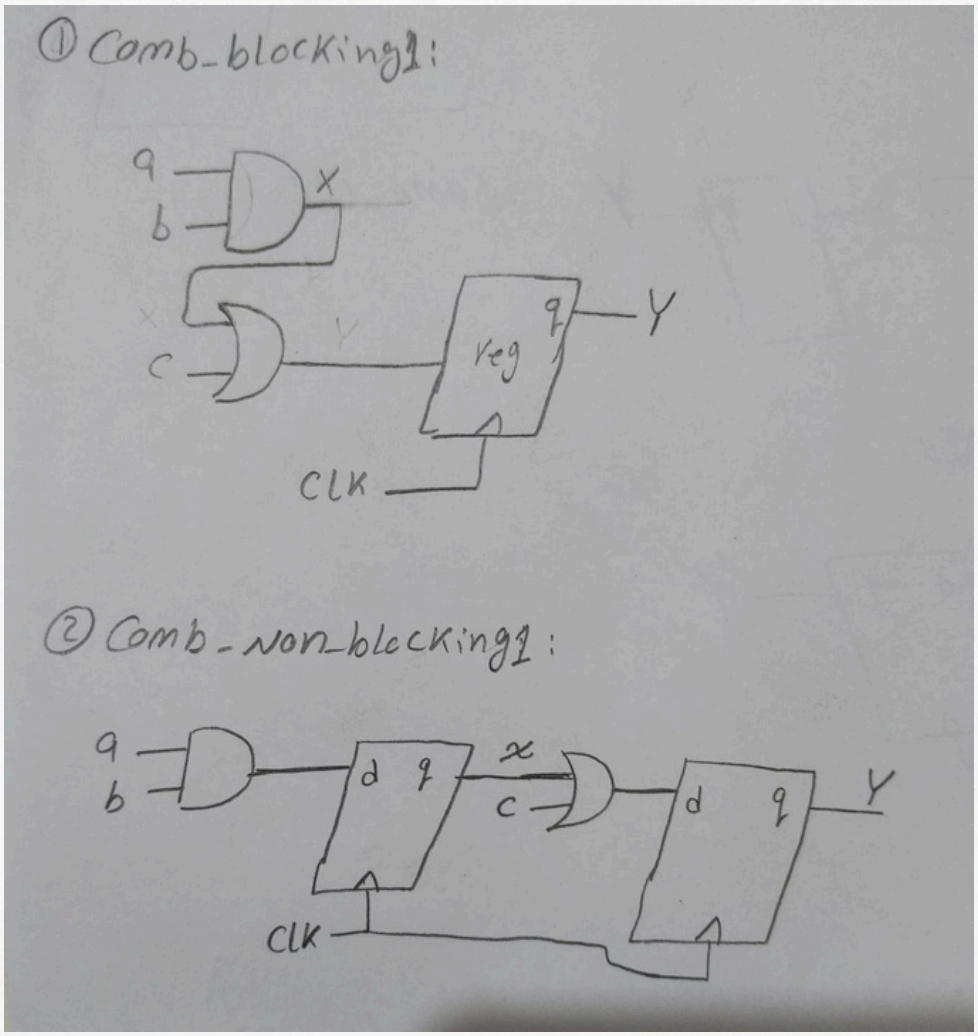
1:


① reg_blocking1
in
rst
clk
d
dff
q
q1
q2
out
② reg_non_blocking1
in
rst
clk
d
q
rst
q1
q2
out
③ reg_blocking2
in
rst
clk
D
q
q1
q2
out

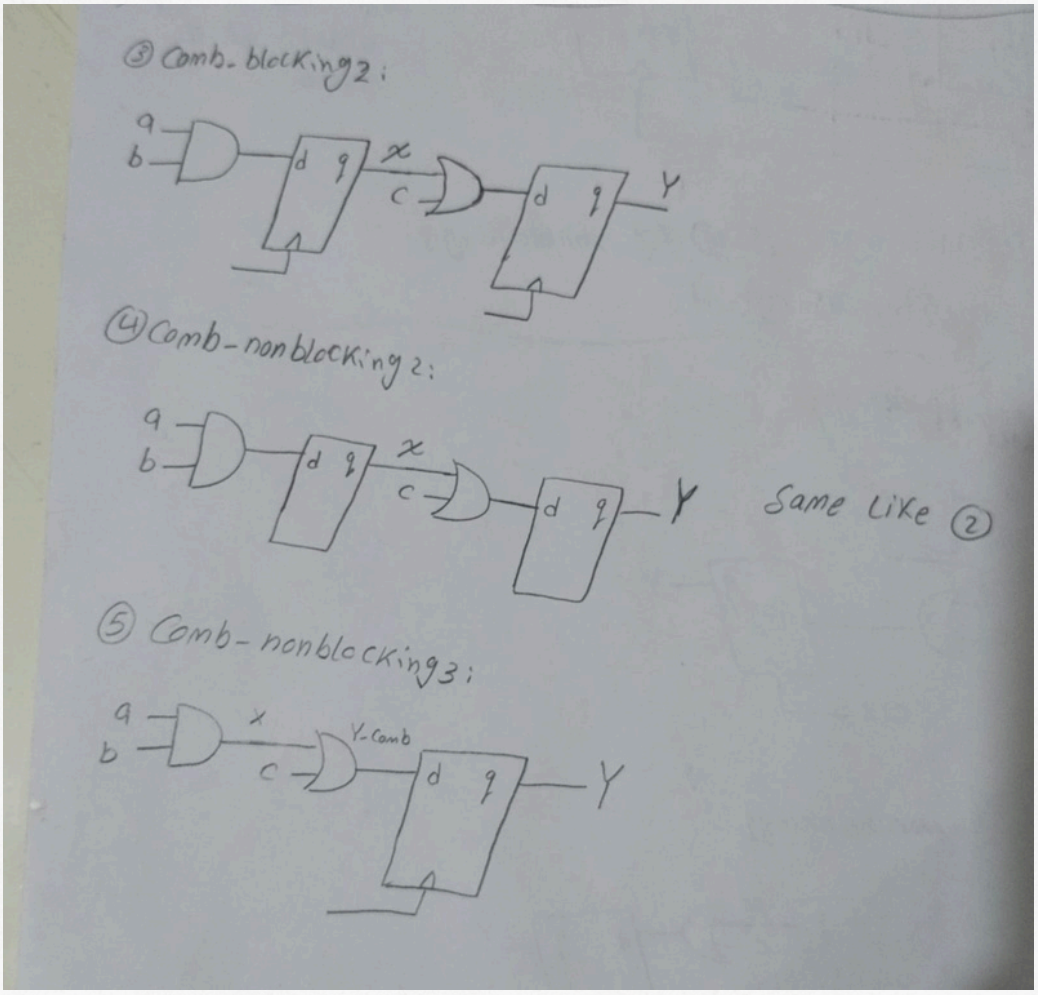
1:


④ reg_Nonblocking2:
in
d
q
q1
d
q
q2
d
q
out
rst
rst
rst
rst
clk
*Same like ②
→ all assigns only update at the end.
⑤ reg_blocking3: & ⑥ reg_nonblocking3:
Same Previous circuit.

2:


① Comb_blocking1:
a
b
X
c
Y
reg
q
Y
clk
② Comb_non_blocking1:
a
b
d
q
x
c
d
q
Y
clk

2:


③ Comb_blocking2:
a
b
d
q
x
c
d
q
Y
④ Comb-nonblocking2:
a
b
d
q
x
c
d
q
Y
Same like ②
⑤ Comb-nonblocking3:
a
b
x
c
Y_Comb
d
q
Y

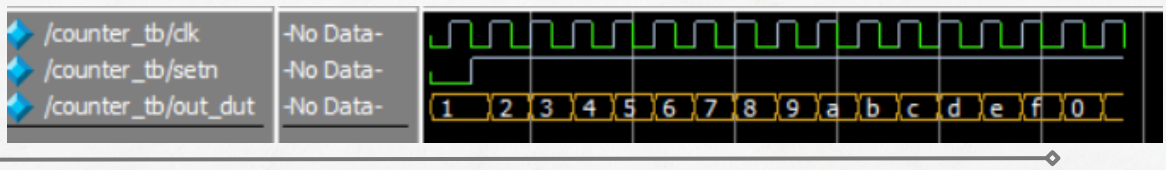
Circuit_2:

```
module counter(input clk, setn, output reg [3:0]out);

    always @(posedge clk, negedge setn) begin

        if(~setn)
            out <= 4'b0001;

        else begin
            out <= out - 1;
        end
    end
endmodule
```

- **testbench:**

```
module counter_tb();

    reg clk, setn;
    wire [3:0]out_dut;
    counter dut(clk, setn, out_dut);
    initial begin
        clk = 0;
        forever begin
            #1 clk = ~clk;
        end
    end
    initial begin
        setn = 0;
        @(negedge clk);
        setn = 1;
        repeat(16)
            @(negedge clk);
        $stop;
    end

endmodule
```

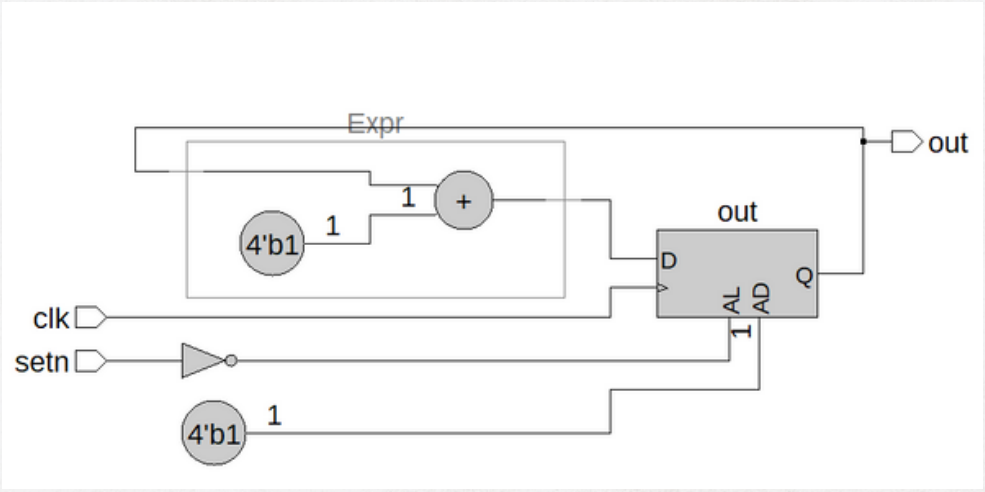
- **output:**



- **lint result:**

Severity	Status	Check
i	?	assigns_mixed
i	?	assigns_mixed_in_always_…
i	?	multi_ports_in_single_line

- **schematic:**



- *it was not applicable for my code to use the ripple ff as a golden reference as it was using reset signal and it was difficult to make both of them have same value at same conditions of control signals.*

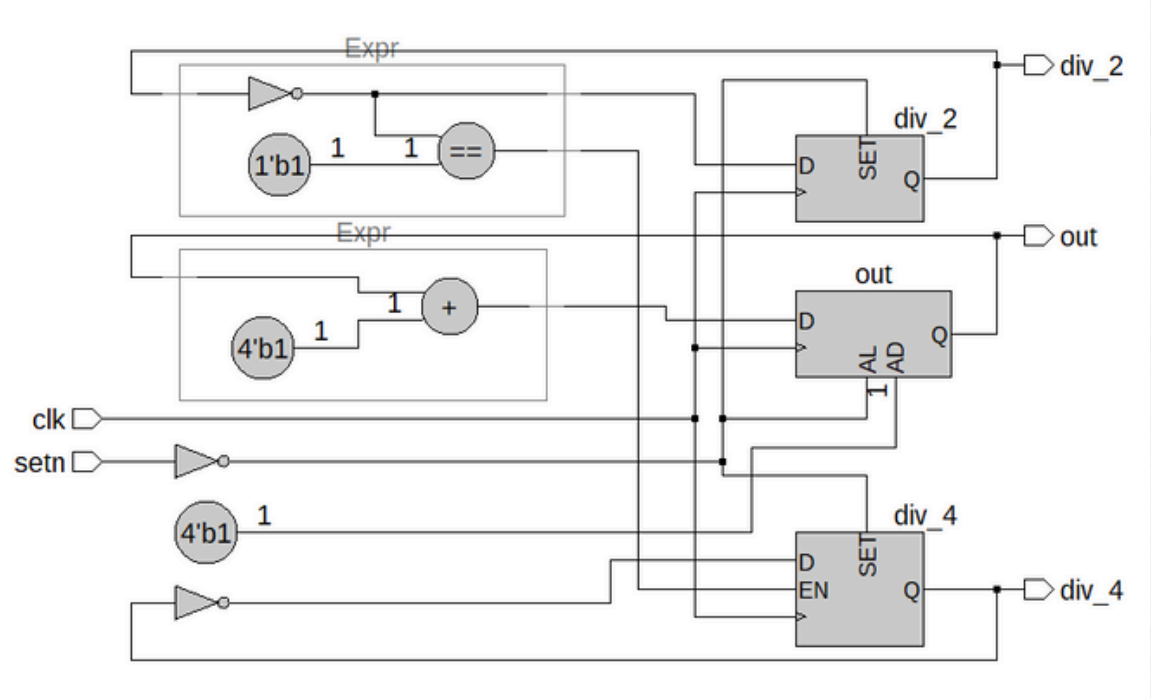
Circuit_3:

```
1   module counter_modified(input clk, setn,
2                   output reg [3:0]out,
3                   output reg div_2, div_4);
4
5       always @(posedge clk, negedge setn) begin
6
7           if(~setn) begin
8               out <= 4'b0001;
9               div_2 <= 1;
10              div_4 <= 1;
11          end
12
13          else begin
14              out <= out + 1;
15              div_2 = ~div_2;
16              if(div_2 == 1)
17                  div_4 <= ~div_4;
18          end
19      end
20  endmodule
```

- **lint result:**

Severity	Status	Check
	?	assigns_mixed
	?	assigns_mixed_in_always_...
	?	multi_ports_in_single_line

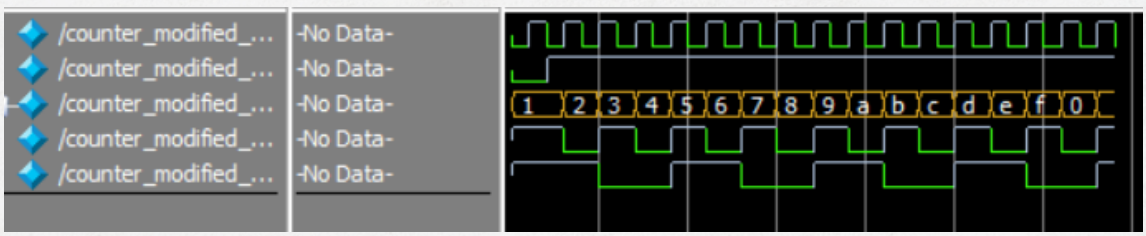
- **schematic:**


Expr
1'b1
1
1
==
Expr
4'b1
1
1
+
clk
setn
4'b1
1
div_2
D
SET
Q
div_2
out
out
D
AL
AD
Q
1
div_4
D
EN
SET
Q
div_4

- **testbench:**

```
module counter_modified_tb();
    reg clk, setn;
    wire [3:0] out_dut;
    wire div_2, div_4;

    counter_modified dut(clk, setn, out_dut, div_2, div_4);

    initial begin
        clk = 0;
        forever
            #1 clk = ~clk;
    end

    initial begin
        setn = 0;
        @(negedge clk);
        setn = 1;

        repeat(16)
            @(negedge clk);

        $stop;
    end
endmodule
```

- **output:**



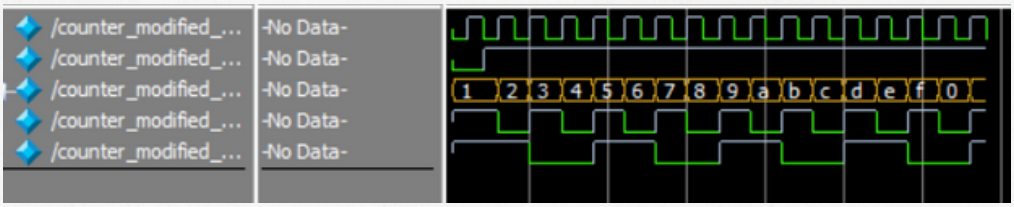
Circuit_4:

```
module gray_counter (input clk, rst,
                     output reg[1:0]gray_out);
    reg [1:0]binary_count;

    always @(posedge clk, posedge rst) begin
        if(rst)
            binary_count <= 0;
        else begin
            binary_count <= binary_count + 1;
            gray_out <= {binary_count[1], (^binary_count)};
        end
    end
endmodule
```

- **testbench**

```
module gray_counter_tb();
    reg clk, rst;
    wire [1:0]gray_out_dut;

    gray_counter dut(clk, rst, gray_out_dut);
    initial begin
        clk = 0;
        forever
            #1 clk = ~clk;
    end

    initial begin
        rst = 1;
        @(negedge clk);
        rst = 0;
        repeat(10)
            @(negedge clk);
        $stop;
    end
endmodule
```

- **output:**



- **lint result:**

Severity	Status	Check
i	?	async_reset_active_high
i	?	multi_ports_in_single_line

- **schematic:**