Embedded Systems

Week 7: Sequential Circuits

Dr. Vecdi Emre Levent

Instructors

Assist. Prof. Dr. Vecdi Emre Levent

Email : emre@levent.tc

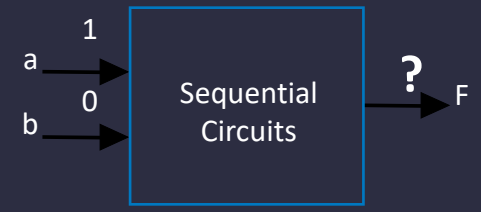
emre.levent@marmara.edu.tr

Web: www.levent.tc

Sequential Circuits

- Sequential Circuits
  - *In sequential circuit, the output depends not only on the current inputs, but also on previous inputs.*
  - Example : Counter counting up with a summation circuit

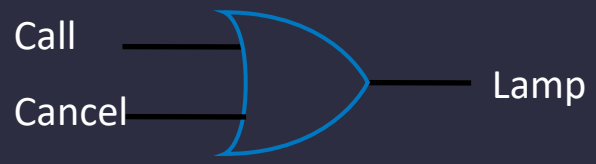
- Memory storage unit called flip-flop



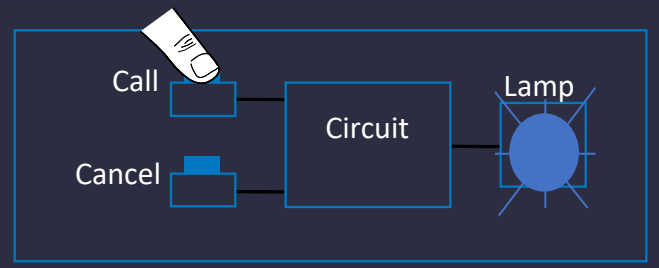
Sequential Circuits

- Flight attendant call button
  - When pressed, the lamp remains active until it is pressed again.
  - When pressed again, the lamp turns off

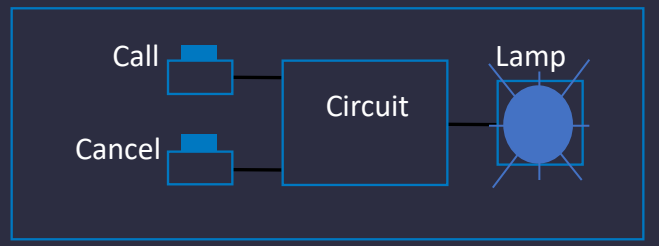
combinatorial circuits?



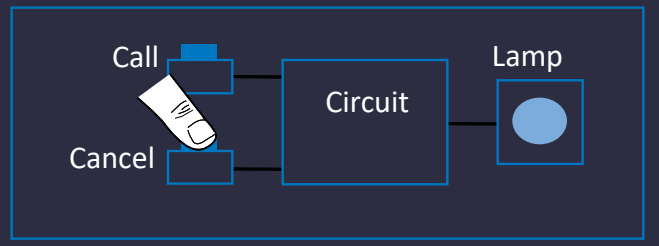
It will not work because the lamp will only be 1 when the call is pressed. The lamp will turn to 0 when the call button is not pressed. A register that holds the previous state of the circuit is needed for the desired behavior



1. Button pressed, light on

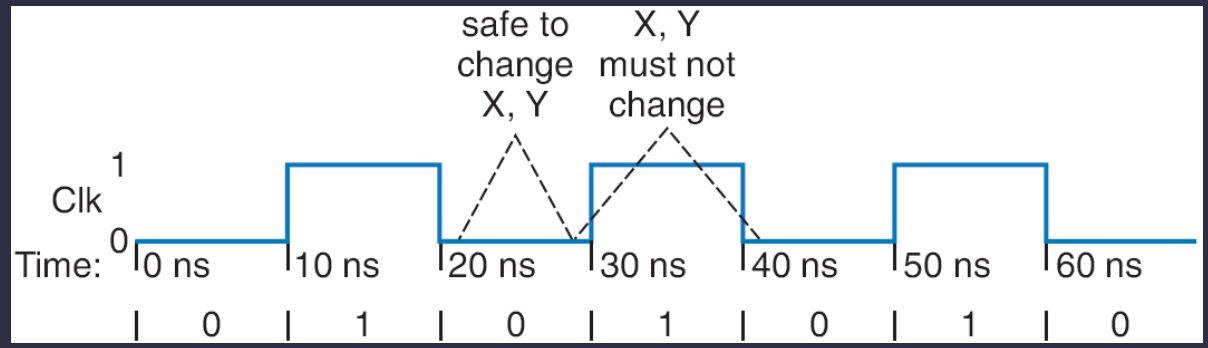


2. The button is released, the light remains on



3. Canceled, light off

Sequential Circuits

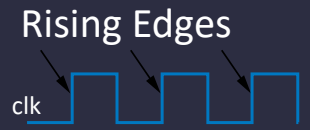


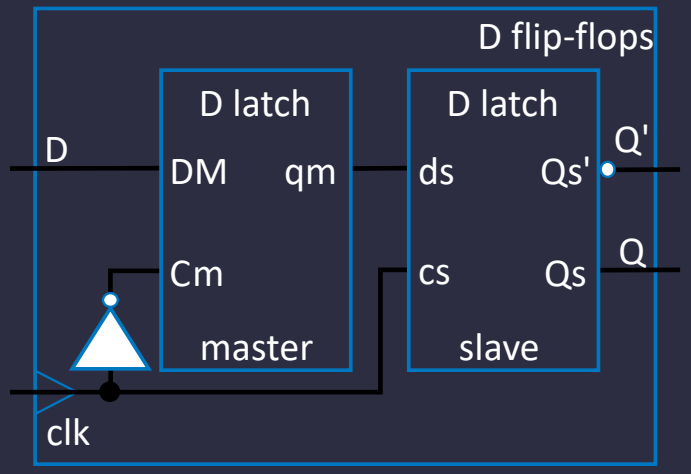
- ***Clock Period*** : The time between two rising edges
  - Period of the above signal : 20 ns
- ***Clock cycle*** :
  - Number of rising edge
- ***Clock Frequency*** : 1/ period
  - The period of the above signal = 1/20 ns = 50 MHZ

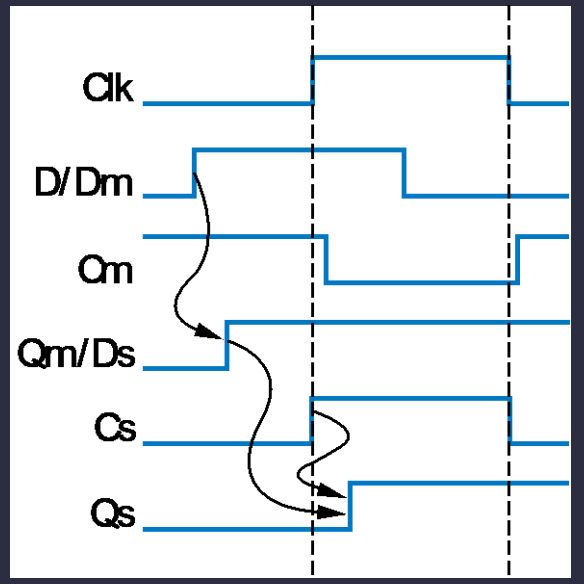
Freq	Period
100GHz	0.01 ns
10GHz	0.1 ns
1GHz	1 ns
100MHz	10 ns
10MHz	100 ns

Sequential Circuits

- ***Flip-flop*** : Samples the signal coming from outside on the rising edge of the incoming clock
- Design example – master - slave
  - 2 D latch is used.

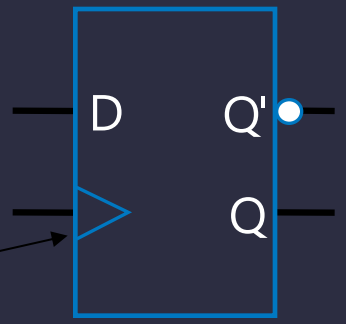




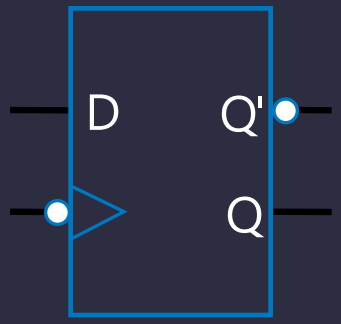


Sequential Circuits

The triangle represents the clock input.



Rising Edge active flip flop notation



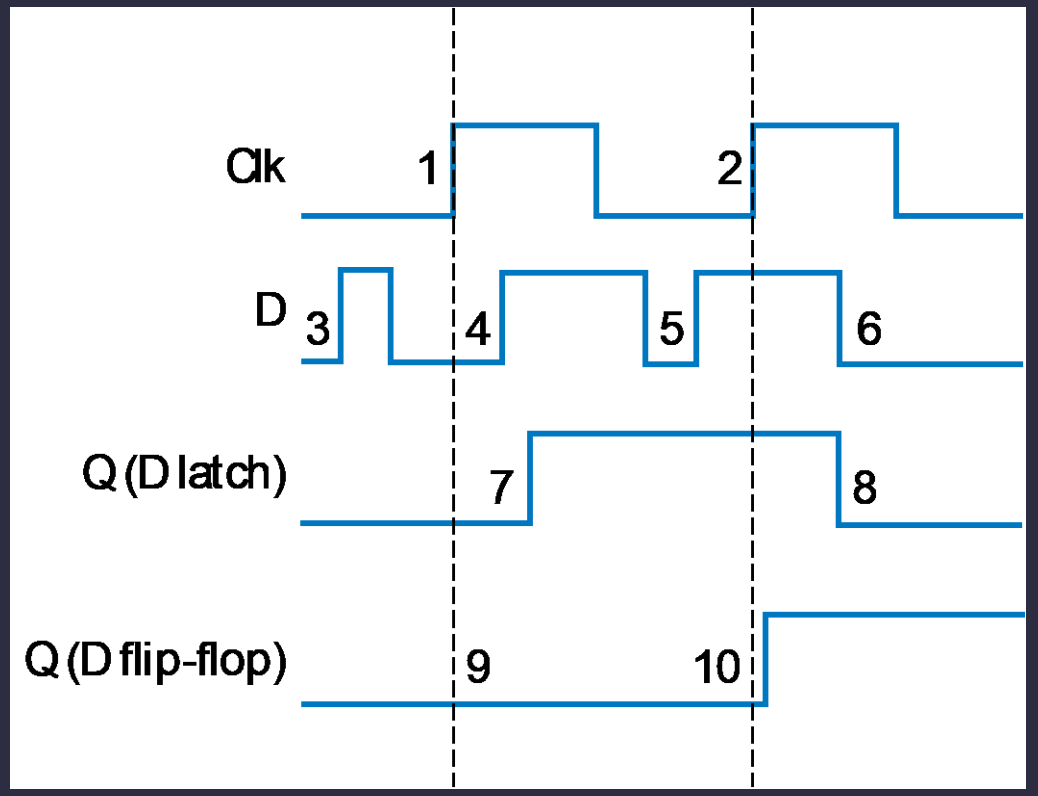
Falling Edge active flip flop notation

Rising Edges
(Positive edge)

Falling Edges
(negative edge)

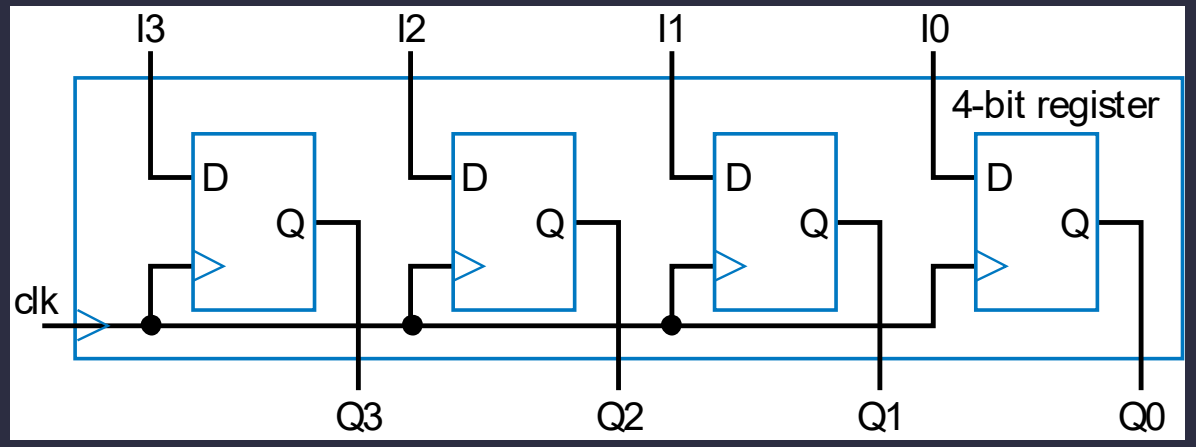
Sequential Circuits

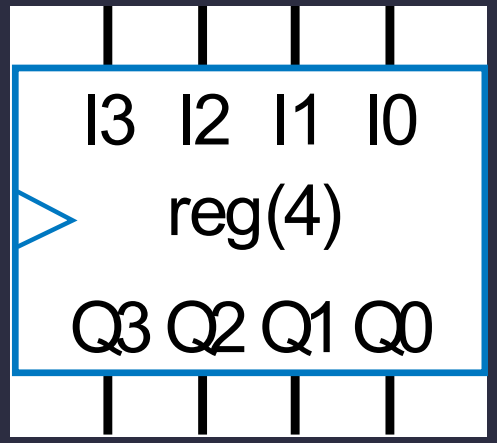
- D latch
- D -flip flop



Sequential Circuits

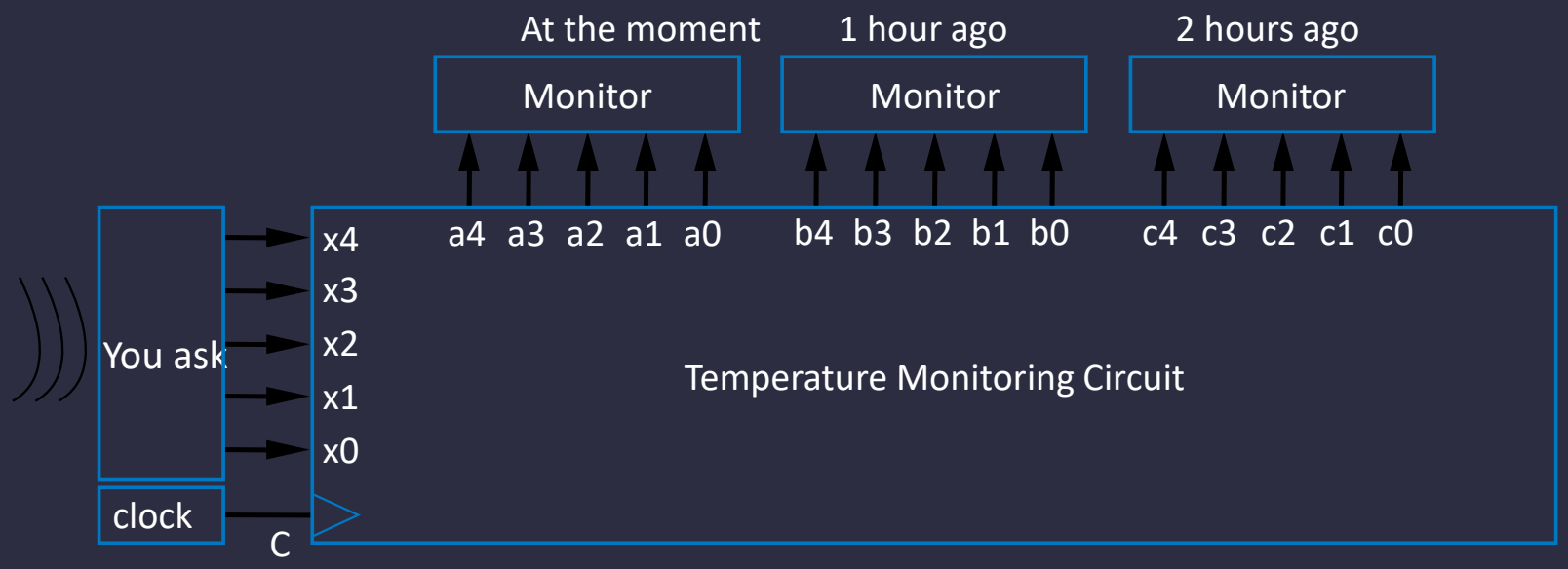
- When multiple bits need to be held, D flip flops are used together.
  - For example, when a 4-bit number needs to be kept
- Multi flip ***Register*** to the structure where the flops are held together *it is called.*





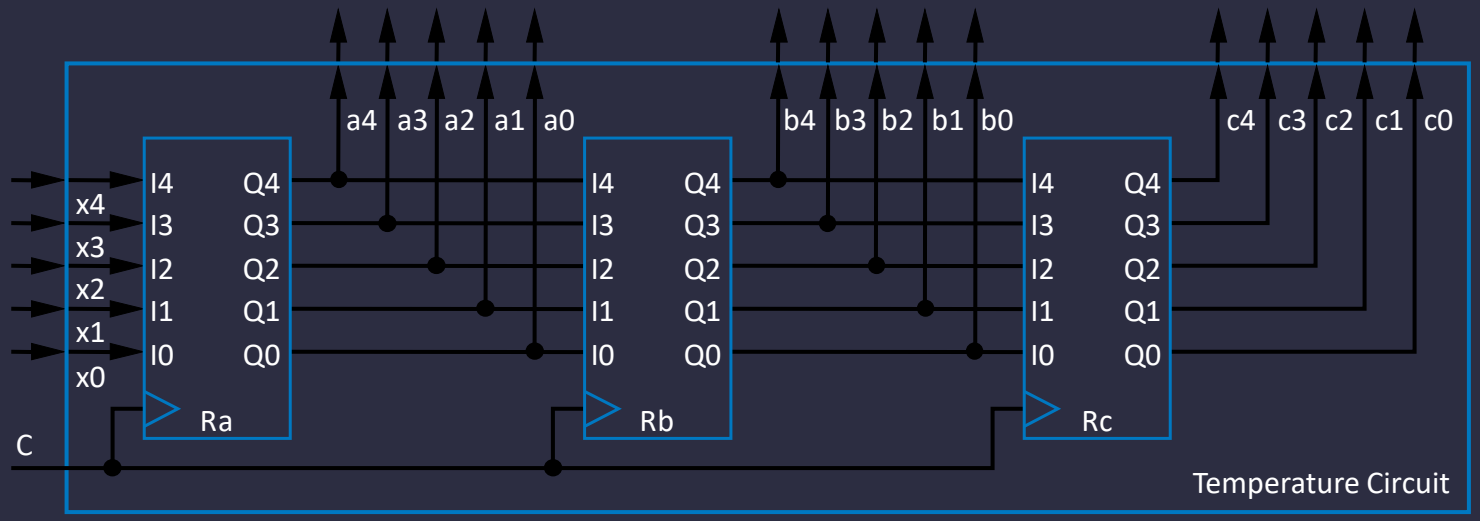
Sequential Circuits

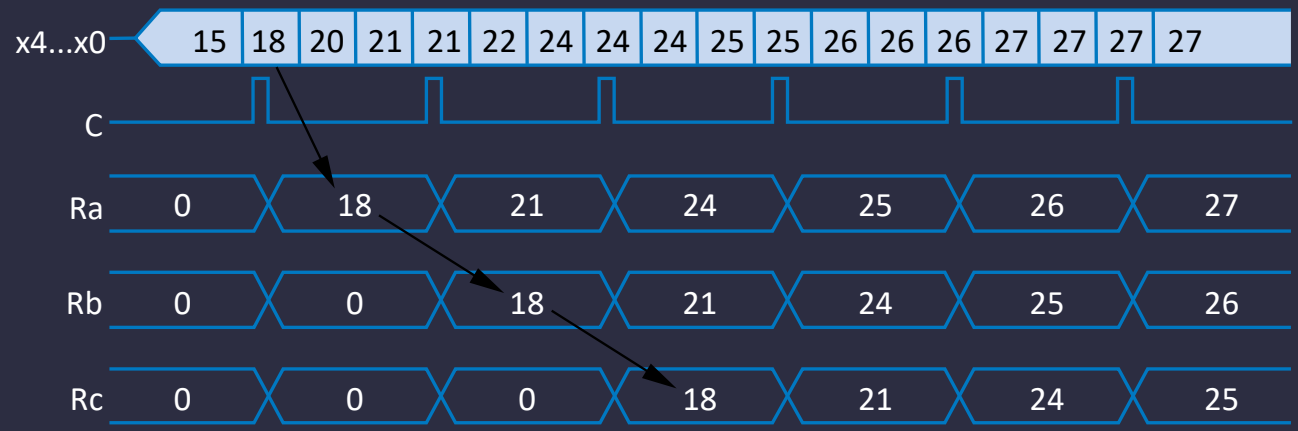
- Temperature Circuit
  - Temperature sensor gives a 5 bit output.
  - Clock's period is 1 hour. (It's a pretty slow clock)
  - Each clock on the rising edge and displays it to the monitor.



Sequential Circuits

- 5bit registers



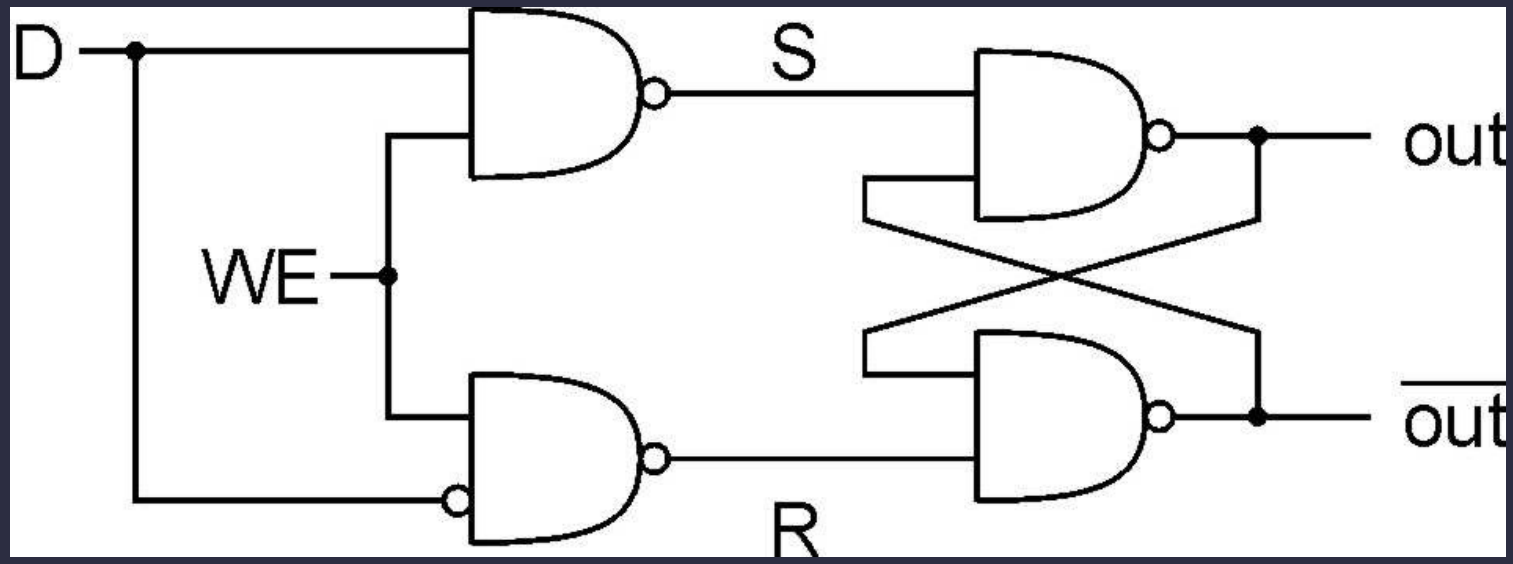


Course

- Sequential Circuits
  - Clock Crystal
  - Clock Cycle
  - Type D Storages

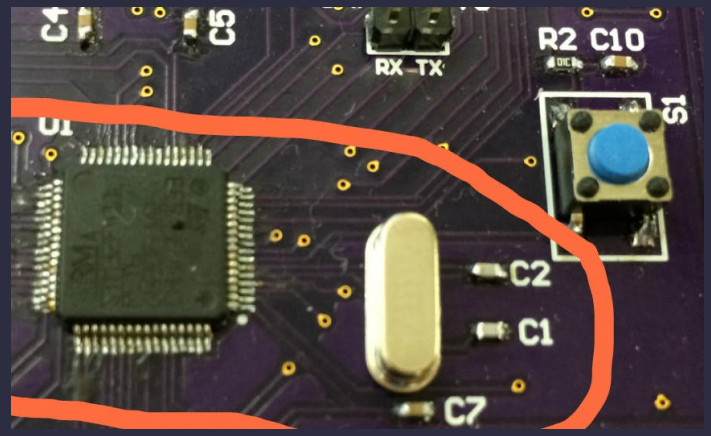
D Type Register (D Latch)

- It has two entrances. These; D (data) and WE (Write Access)
  - WE = 1 encloses the value at input D.
    - S = NOT(D), R = D
  - WE = 0 keeps its previous value.

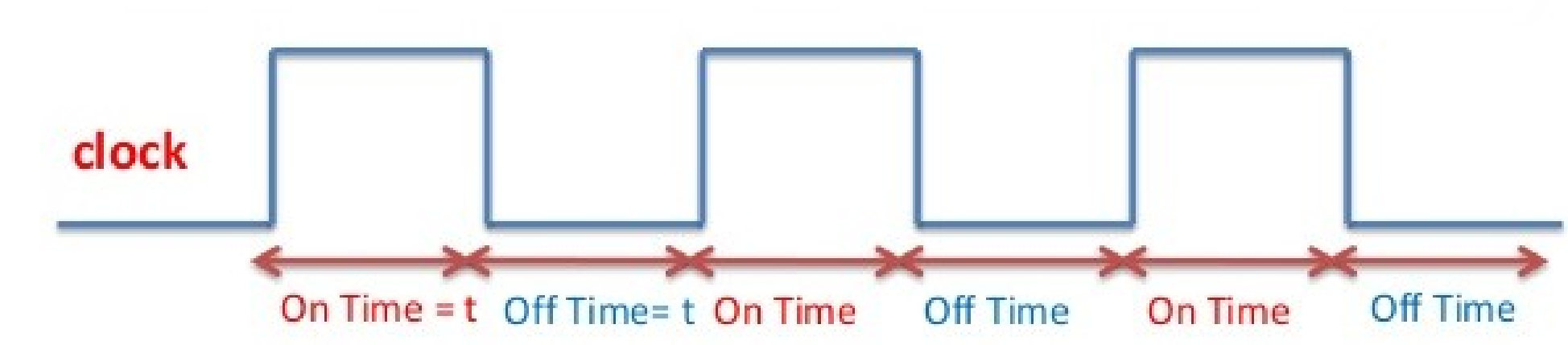


Clock Crystal

- It periodically generates a square-type signal.
- It's like someone is turning a switch on and off at regular intervals...



Clock

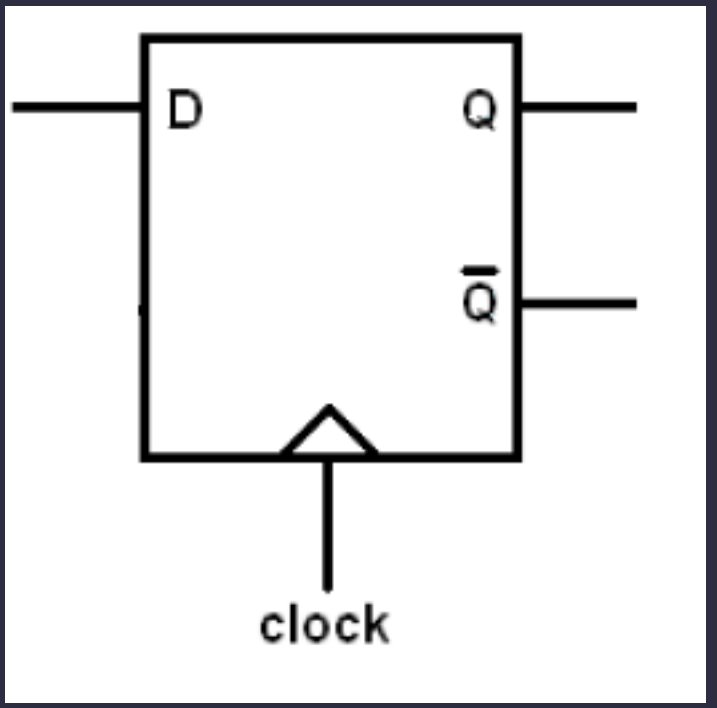


- Clock signal given in the figure repeats itself in 2t time/period.
- Each period of clock called clock cycle.

- It means Frequency = 1/Period.

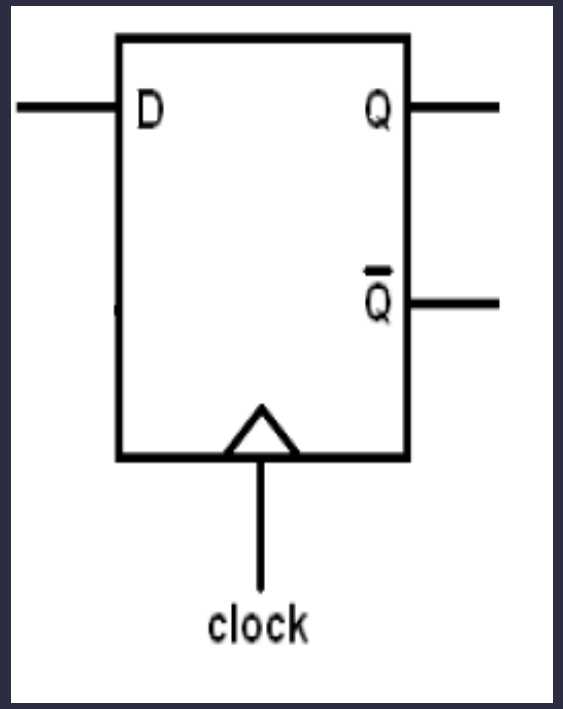
D Type Register (D Register , Flip-Flop)

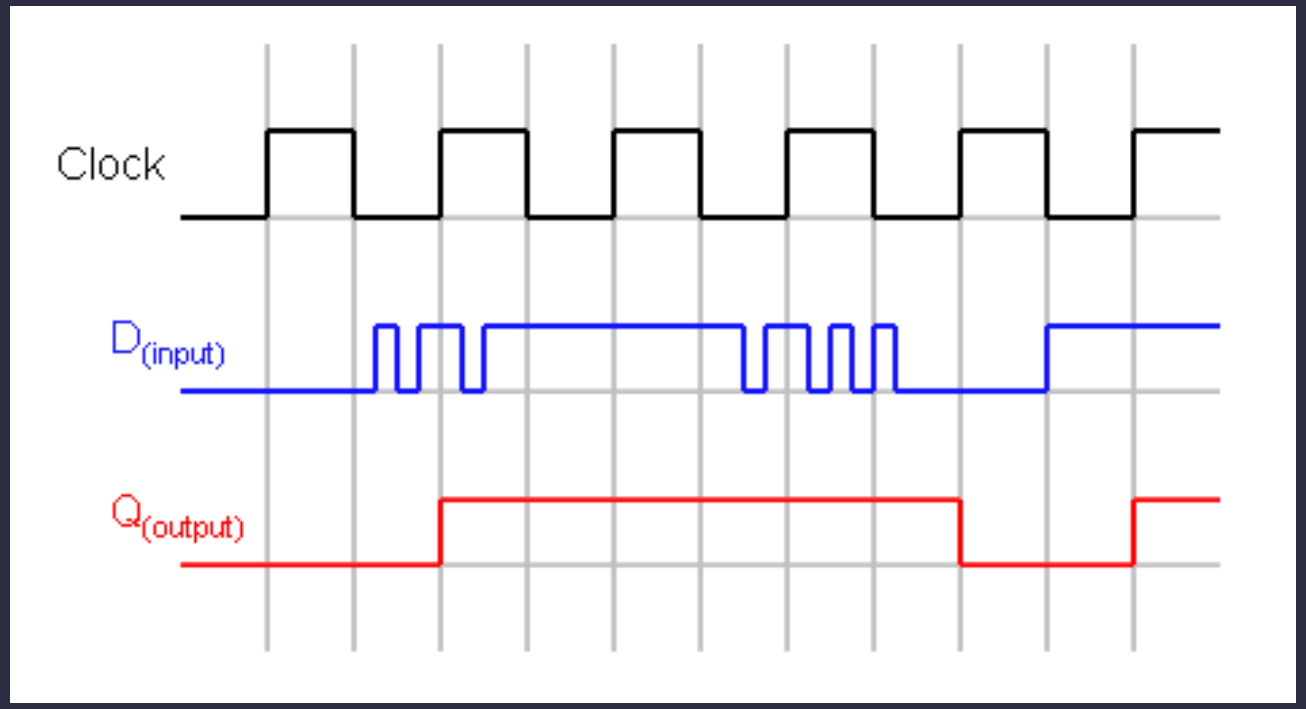
- When the rising or falling edge of the clock signal comes to itself, it transfers the value at the D input to the Q output.
- In other cases, the value at output Q does not change, even if input D changes.



D Type Register (D Register , Flip-Flop)

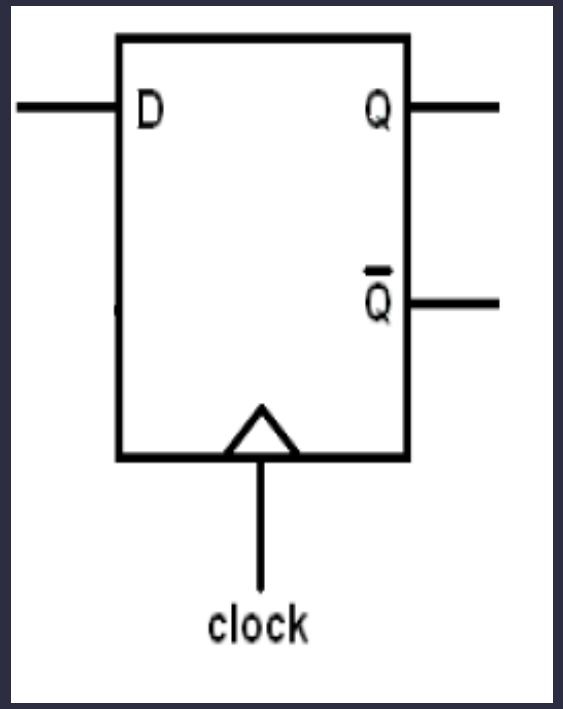
- Rising edge D-Type Storage Inputs and Outputs

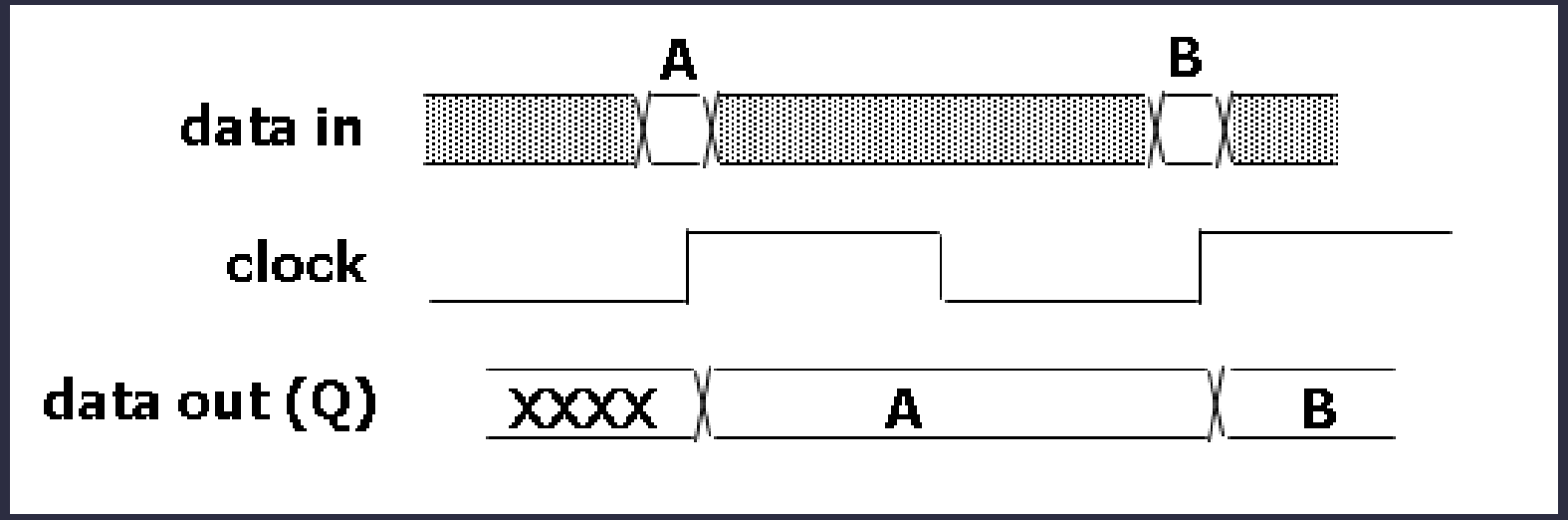




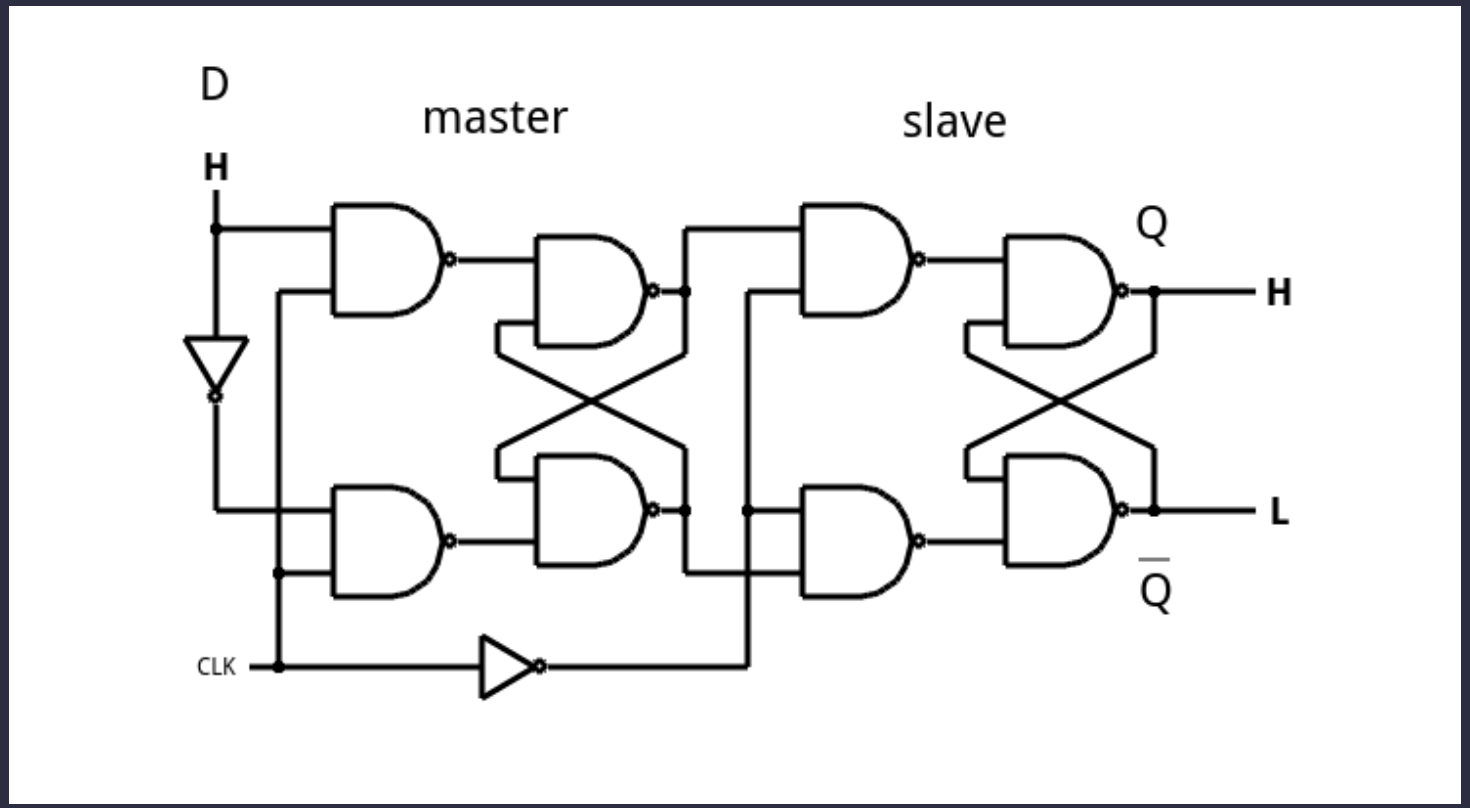
D Type Register (D Register , Flip-Flop)

- Rising edge D-Type Storage Inputs and Outputs





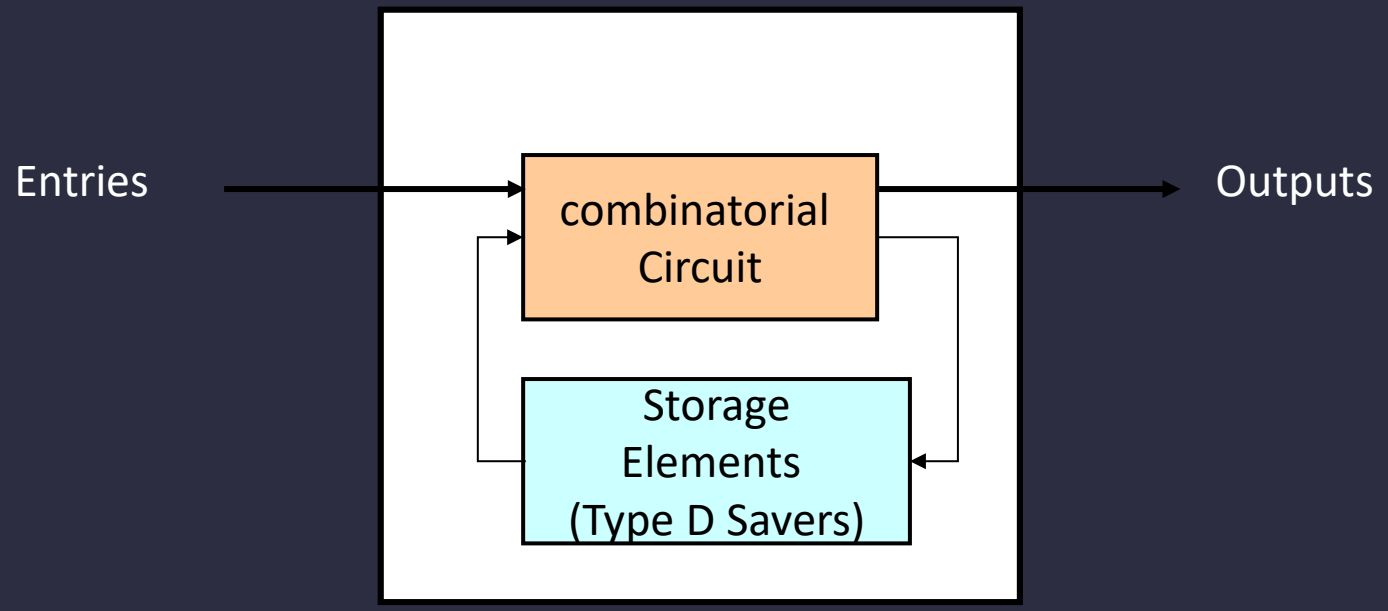
D Type Register (D Register , Flip-Flop)



D Type Storage

Sequential Circuits

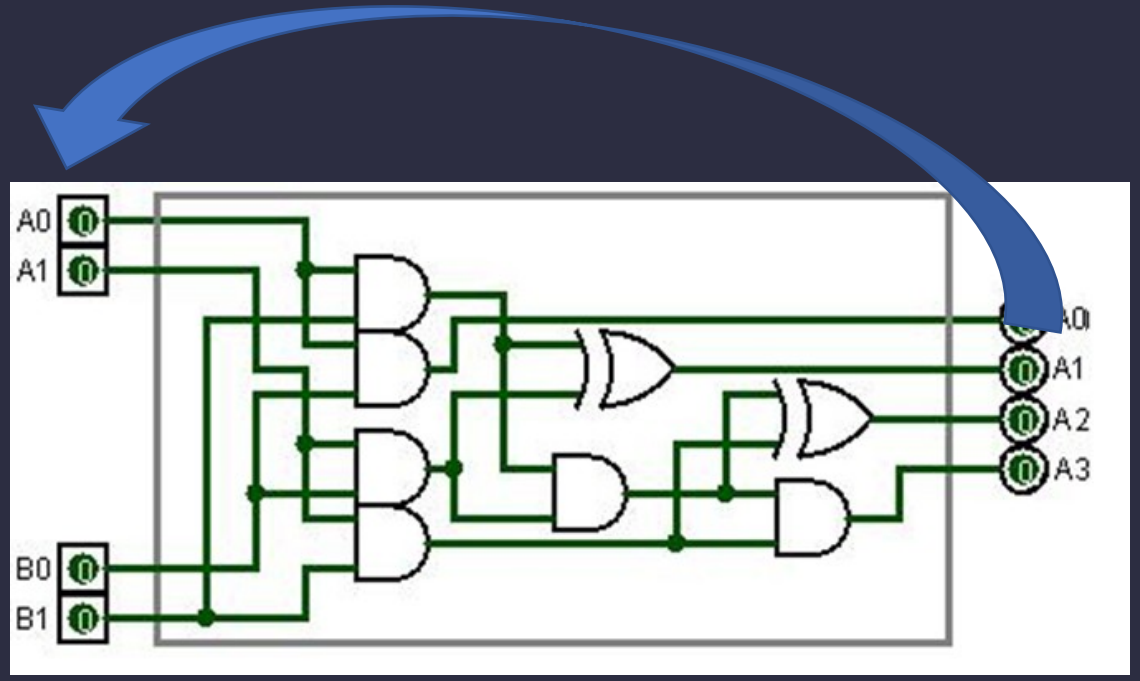
- Combinational circuits and storage elements.
- With the use of storage elements, the previous values produced by the circuit can also be used.



Sequential Circuits

- When is it necessary?

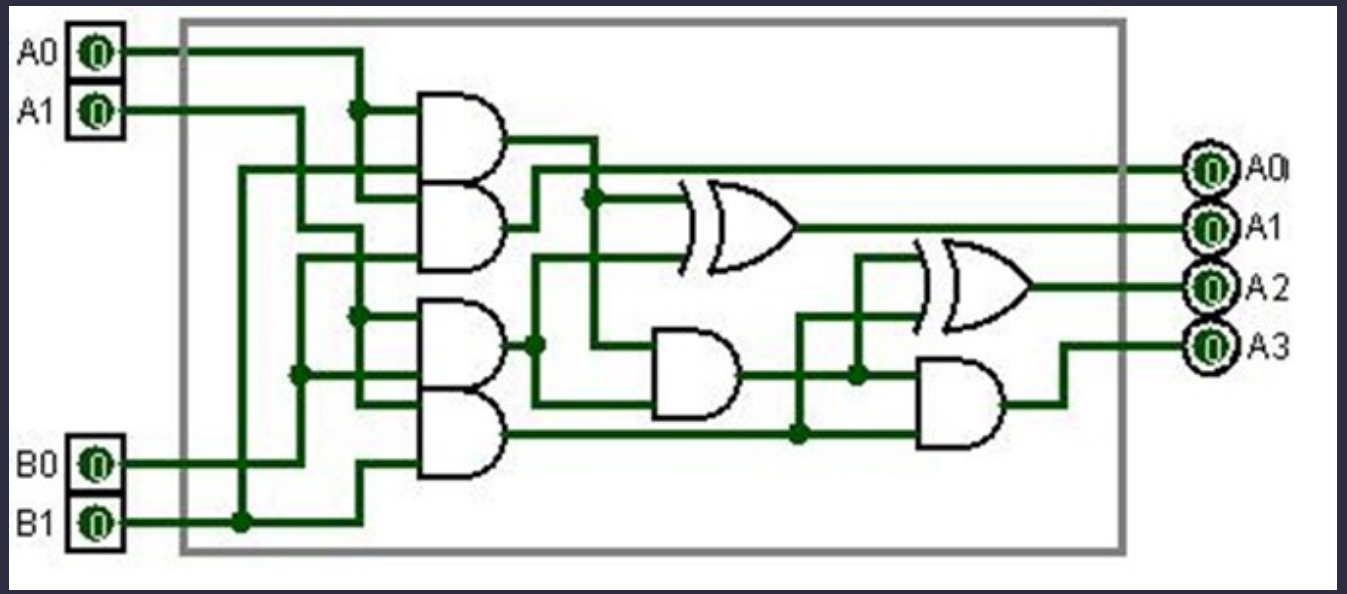
- If the result produced by a circuit will be fed as an input to the circuit;



Sequential Circuits

- When is it necessary?

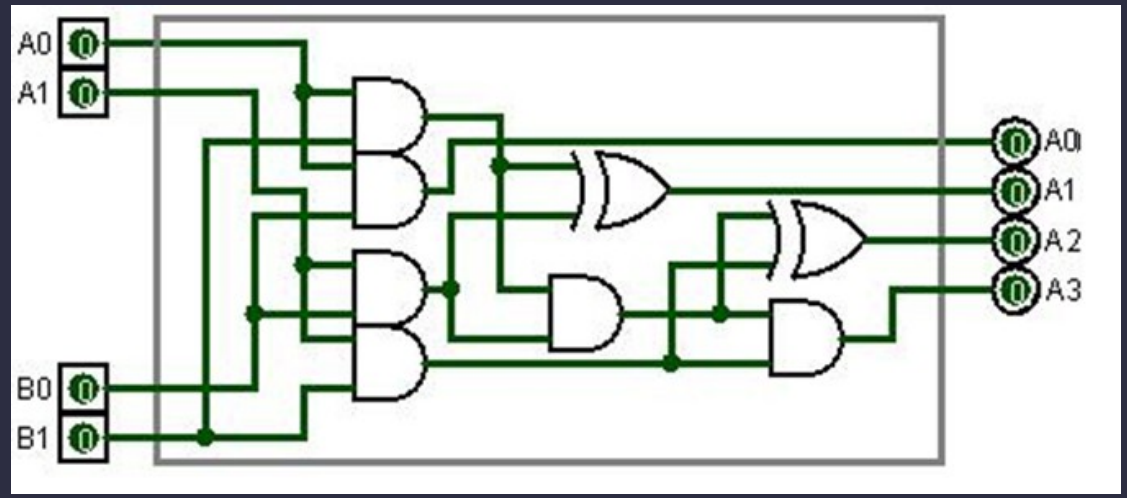
- In order for the circuit to produce correct results, all inputs must remain constant.



Sequential Circuits

- When is it necessary?

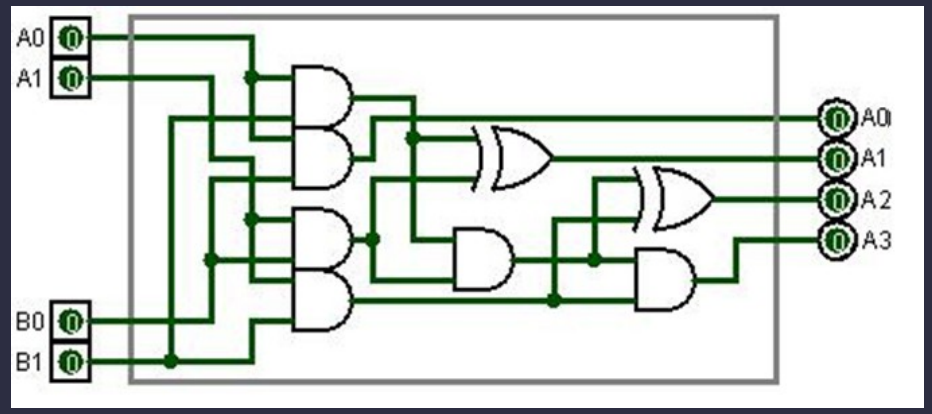
- Each output in this example comes from different logic gate paths.



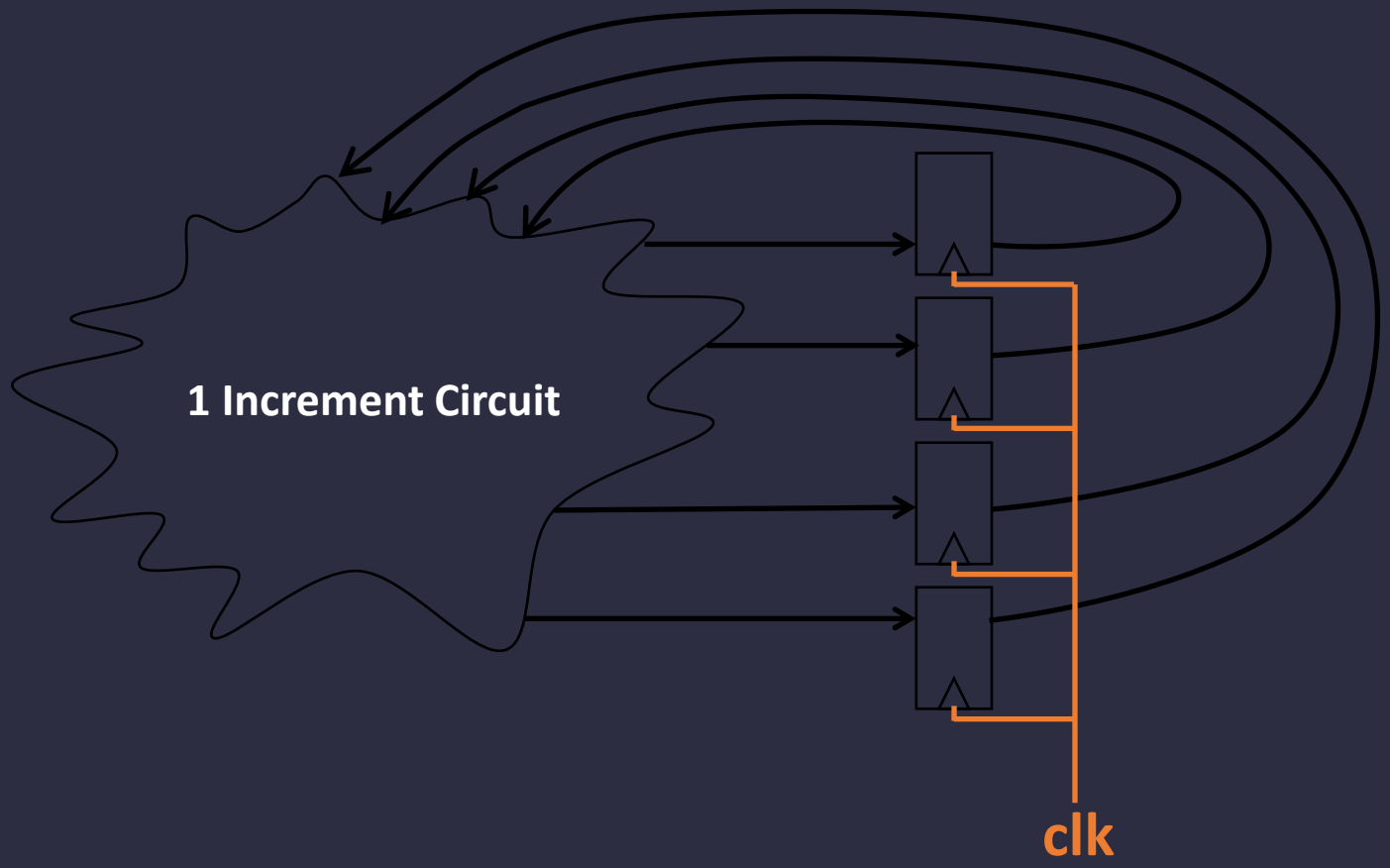
Sequential Circuits

- When is it necessary?

- Therefore, when we connect the output signals directly to the input, the input is changed before the circuit correct output values.
- Correct results can never be captured.



Sequential Circuits

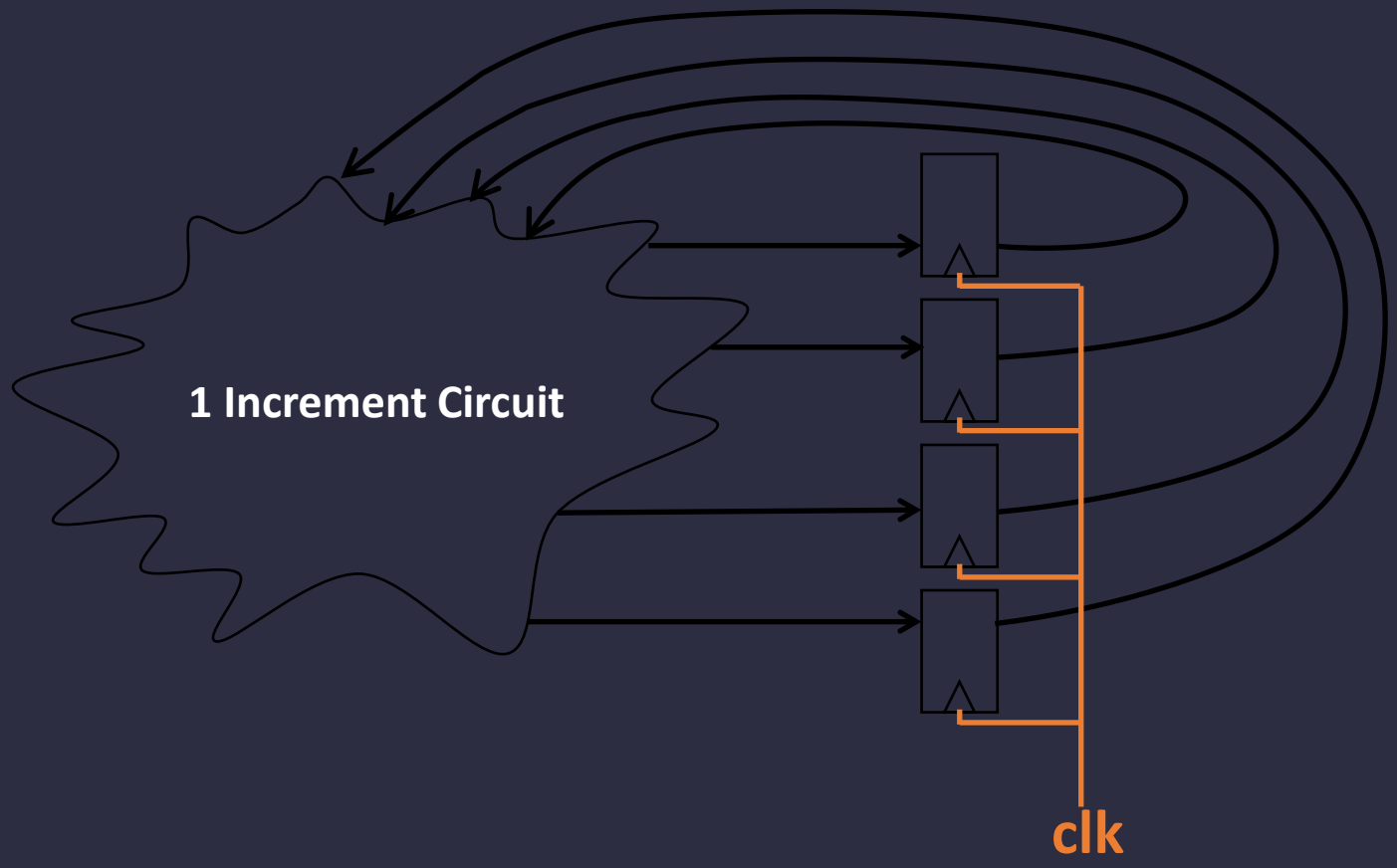


4 Bit 1 increment circuit.

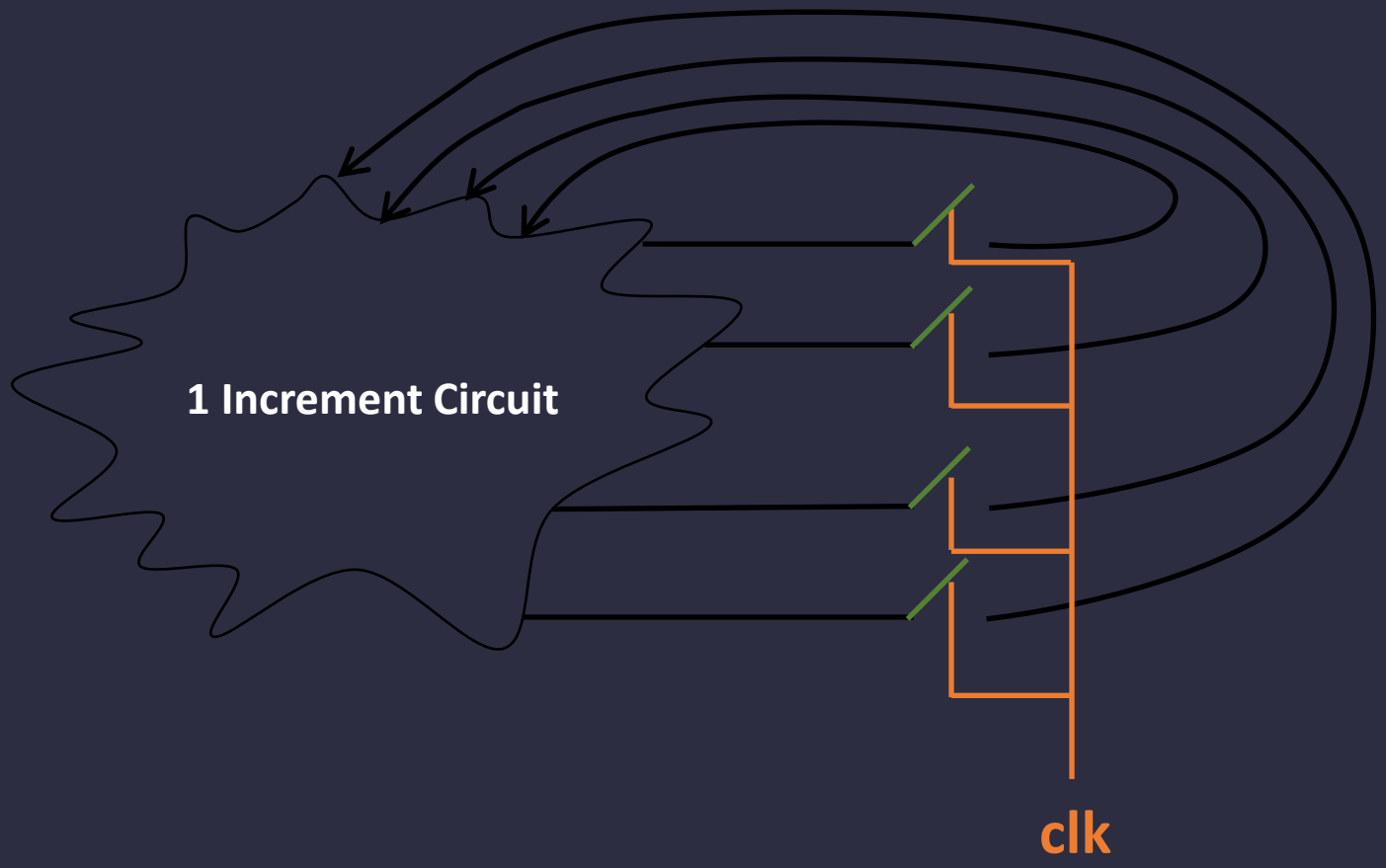
Sequential Circuits

Clock input is as much as the slowest output produced by the circuit

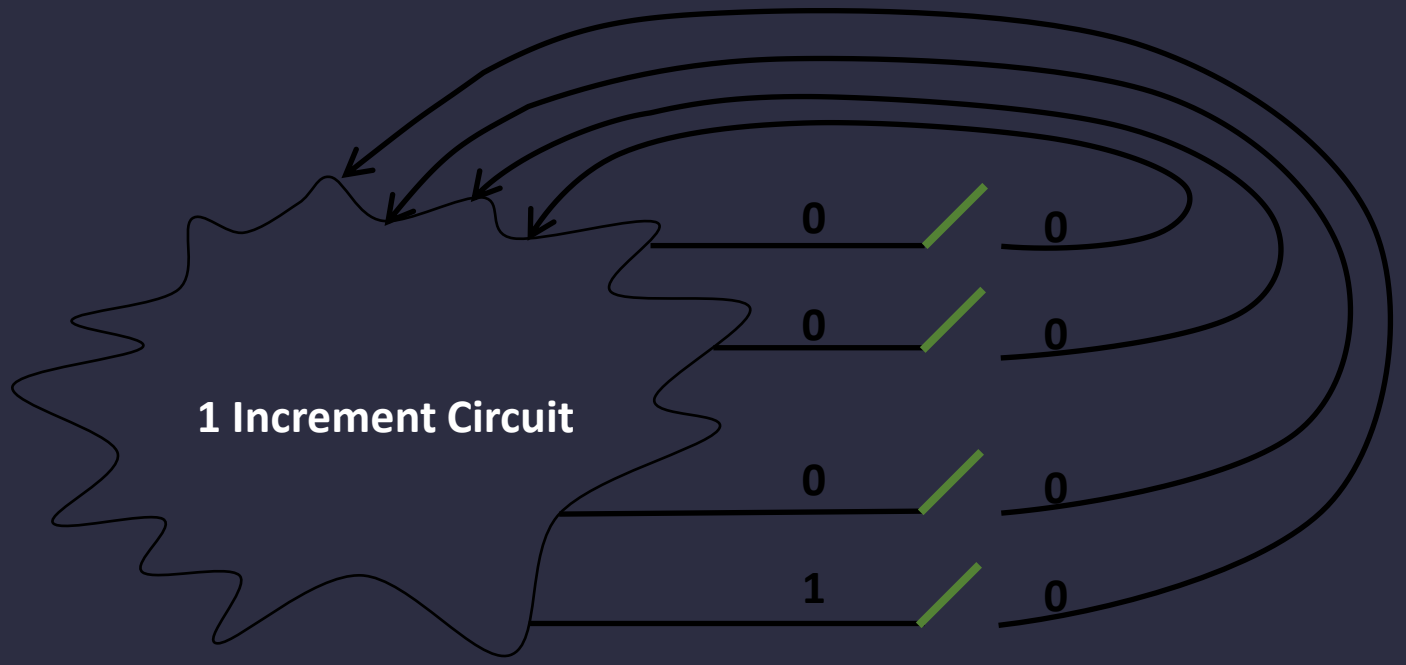
All outputs of the circuit will be fed back to the circuit by waiting until the slowest output output.



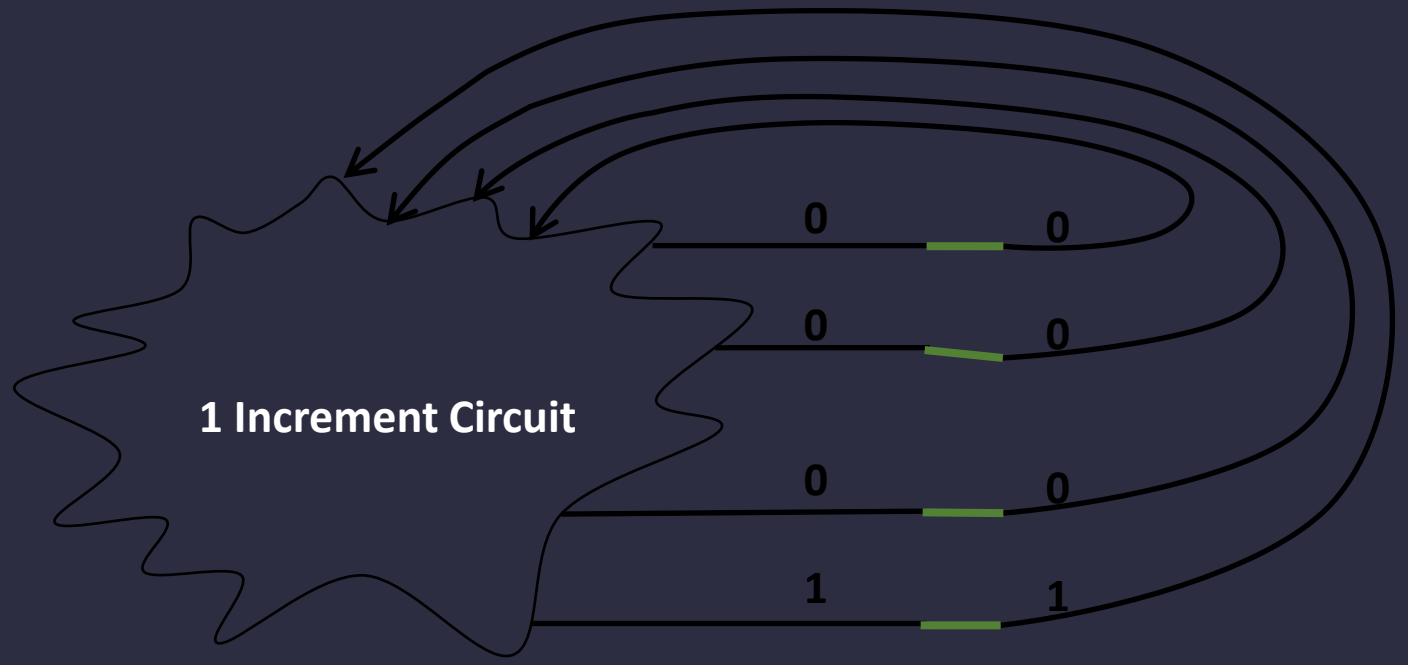
Sequential Circuits



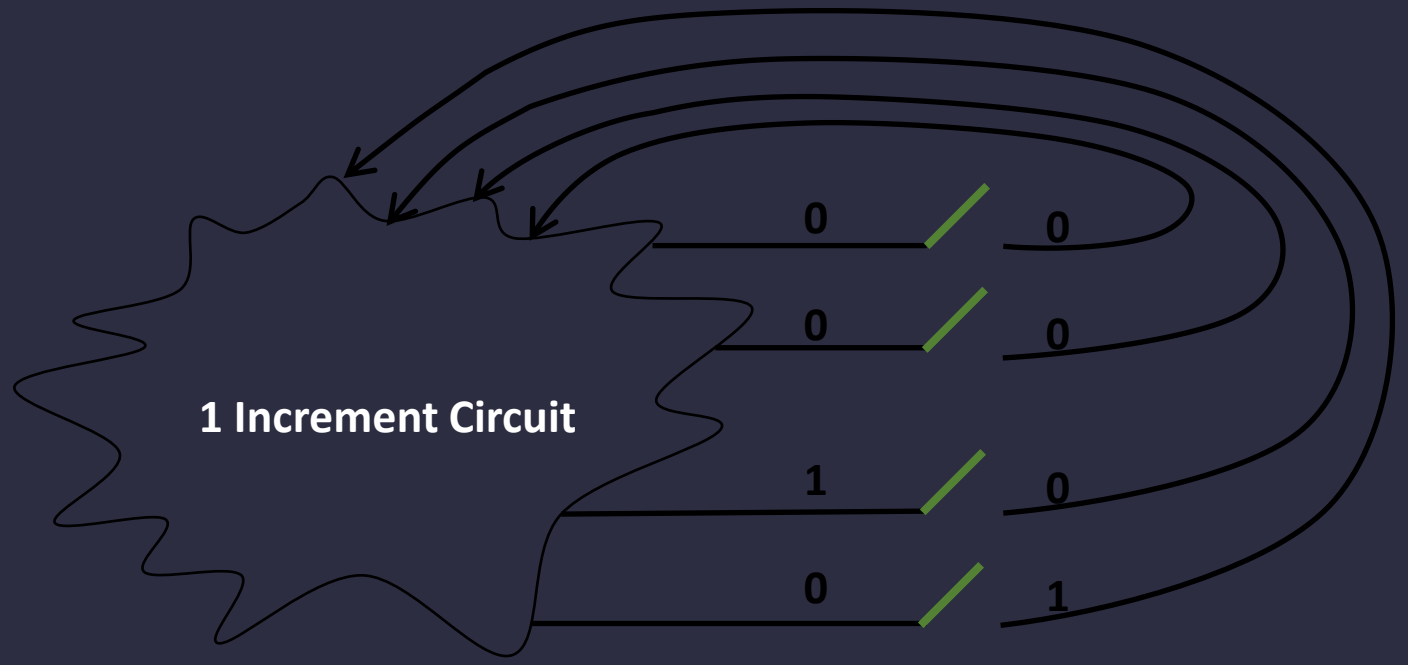
Sequential Circuits



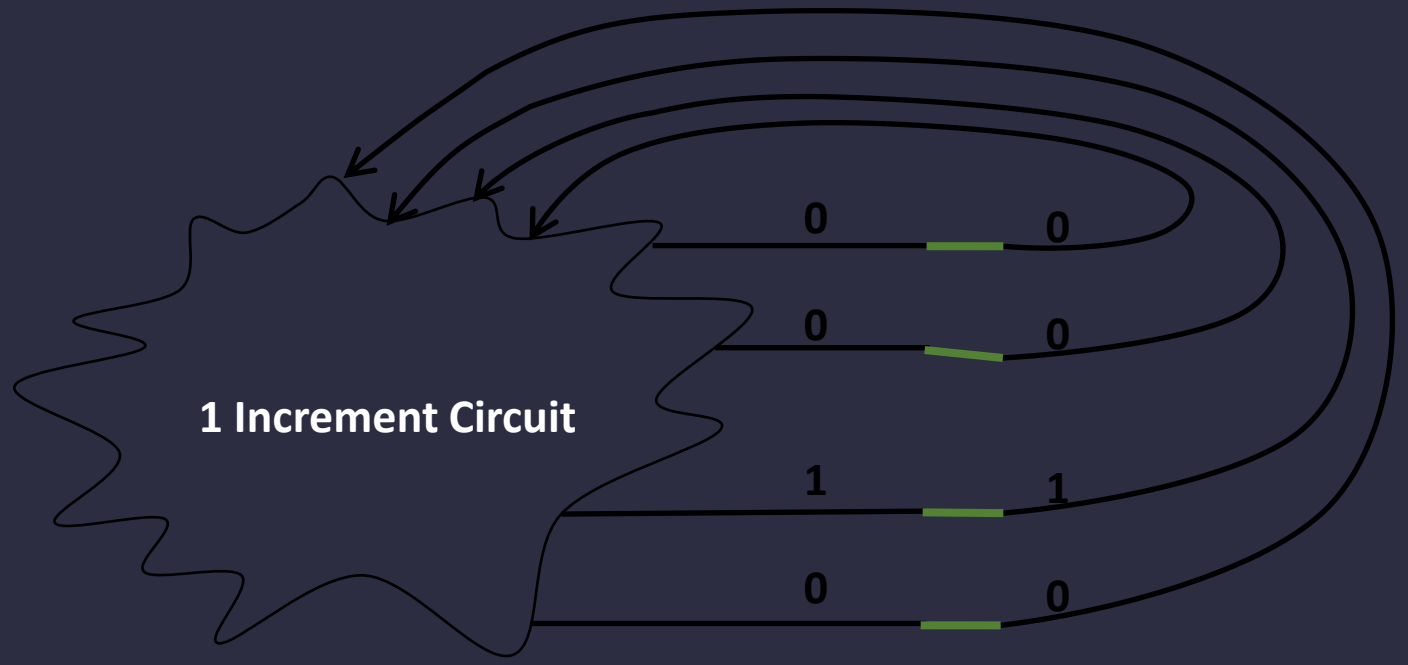
Sequential Circuits



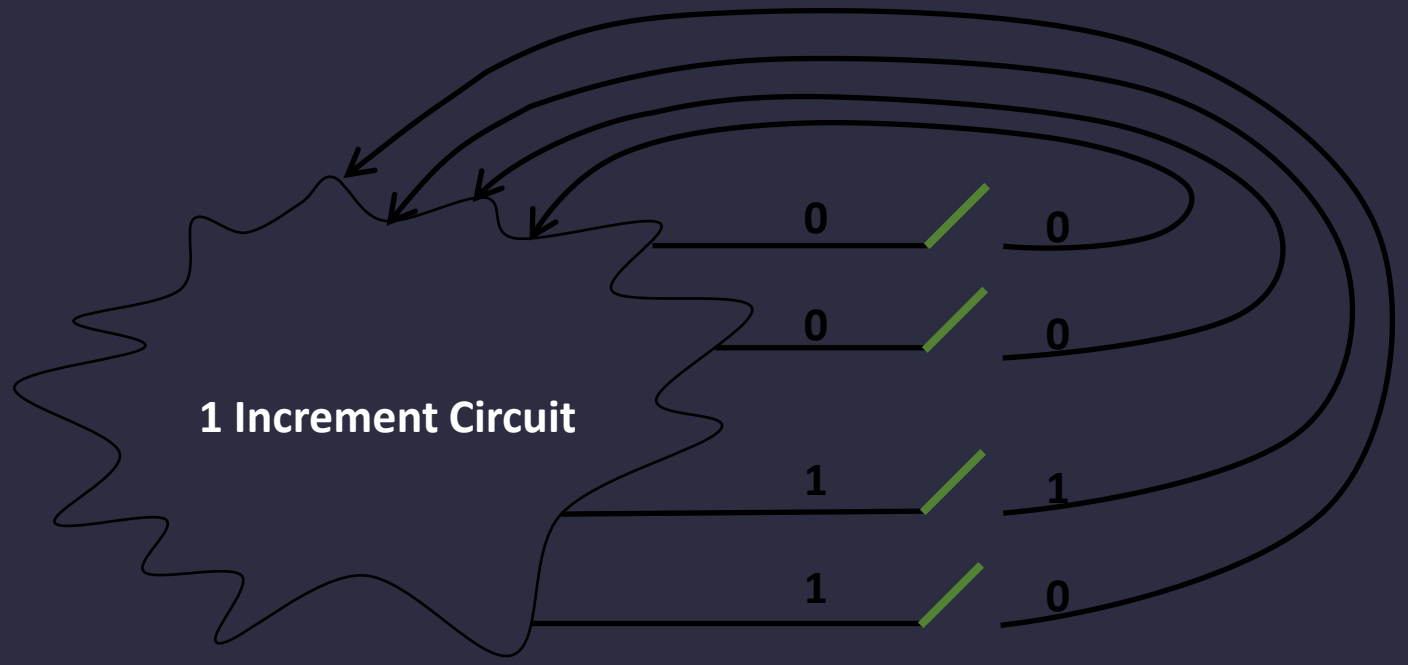
Sequential Circuits



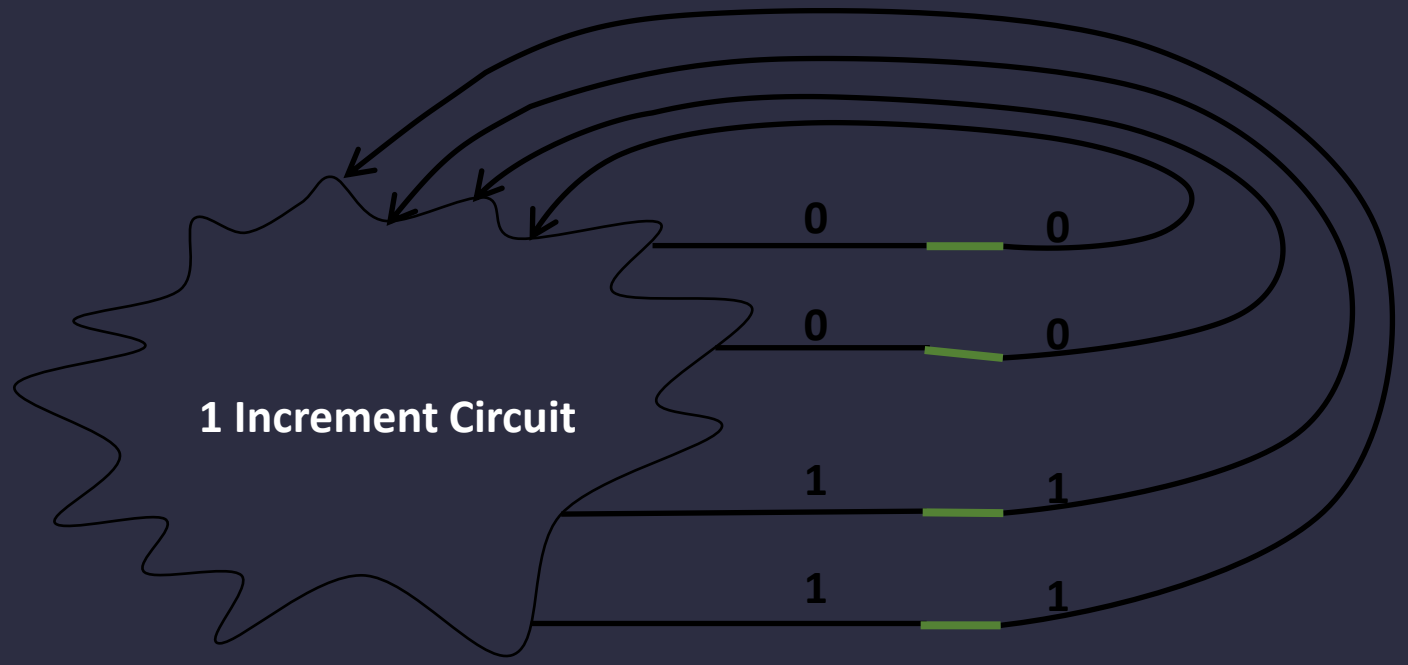
Sequential Circuits



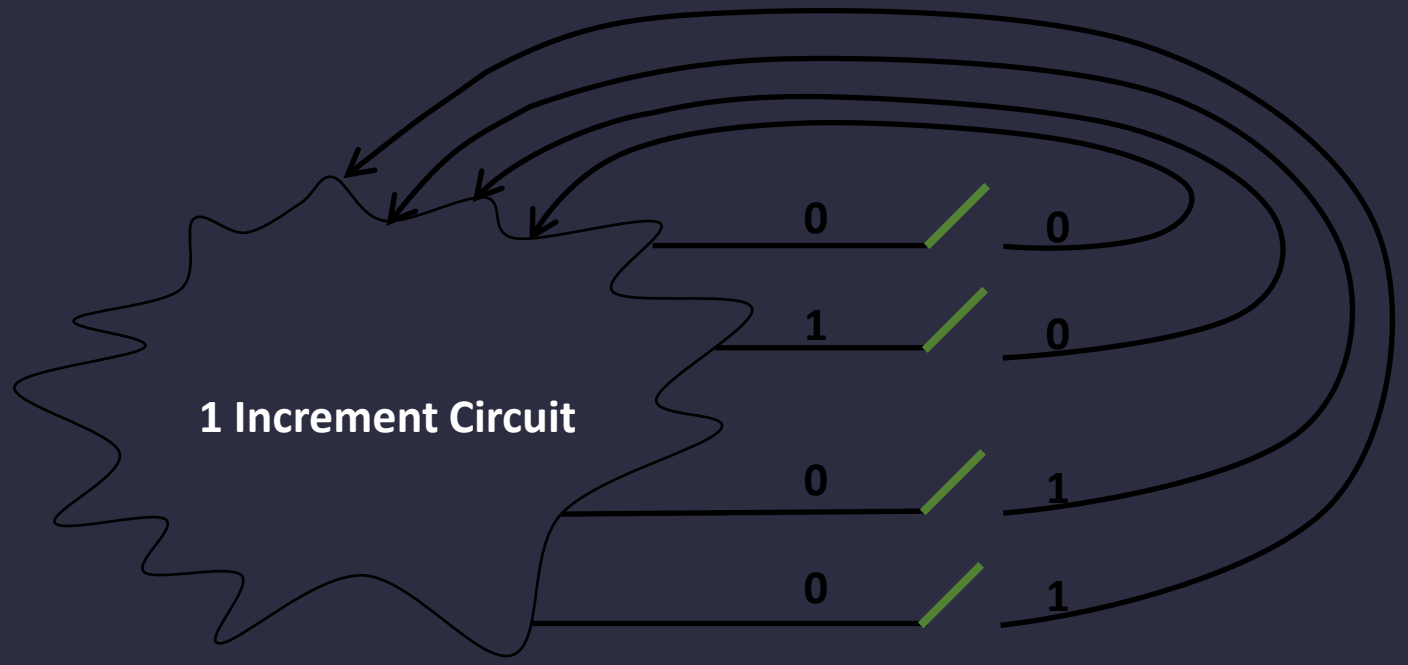
Sequential Circuits



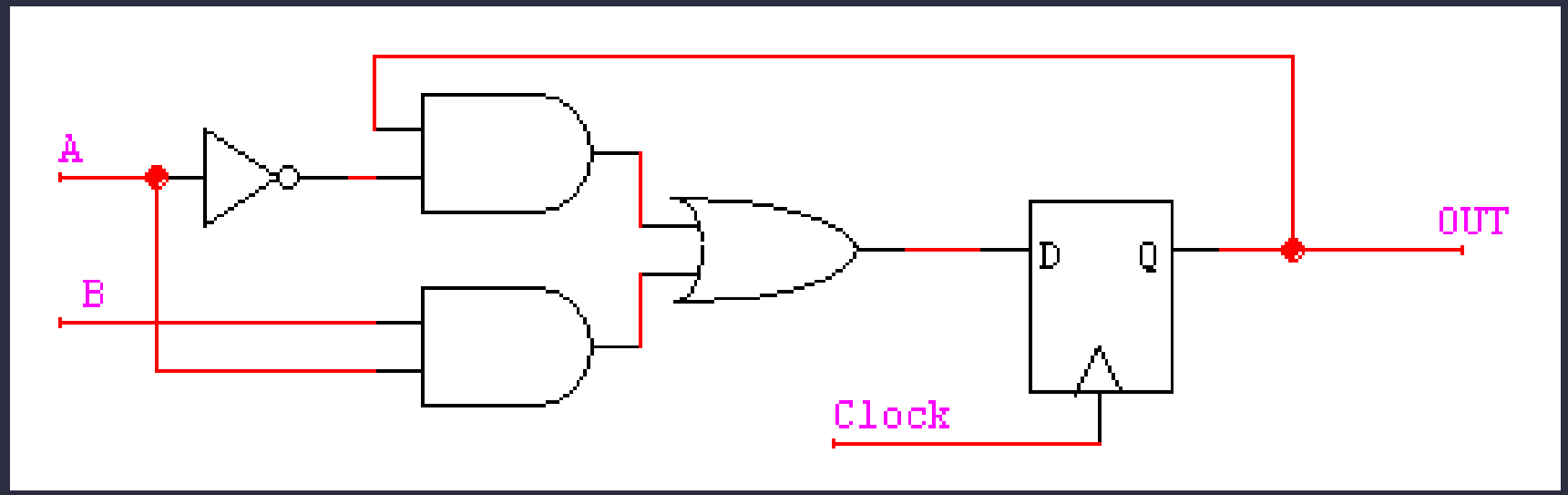
Sequential Circuits



Sequential Circuits

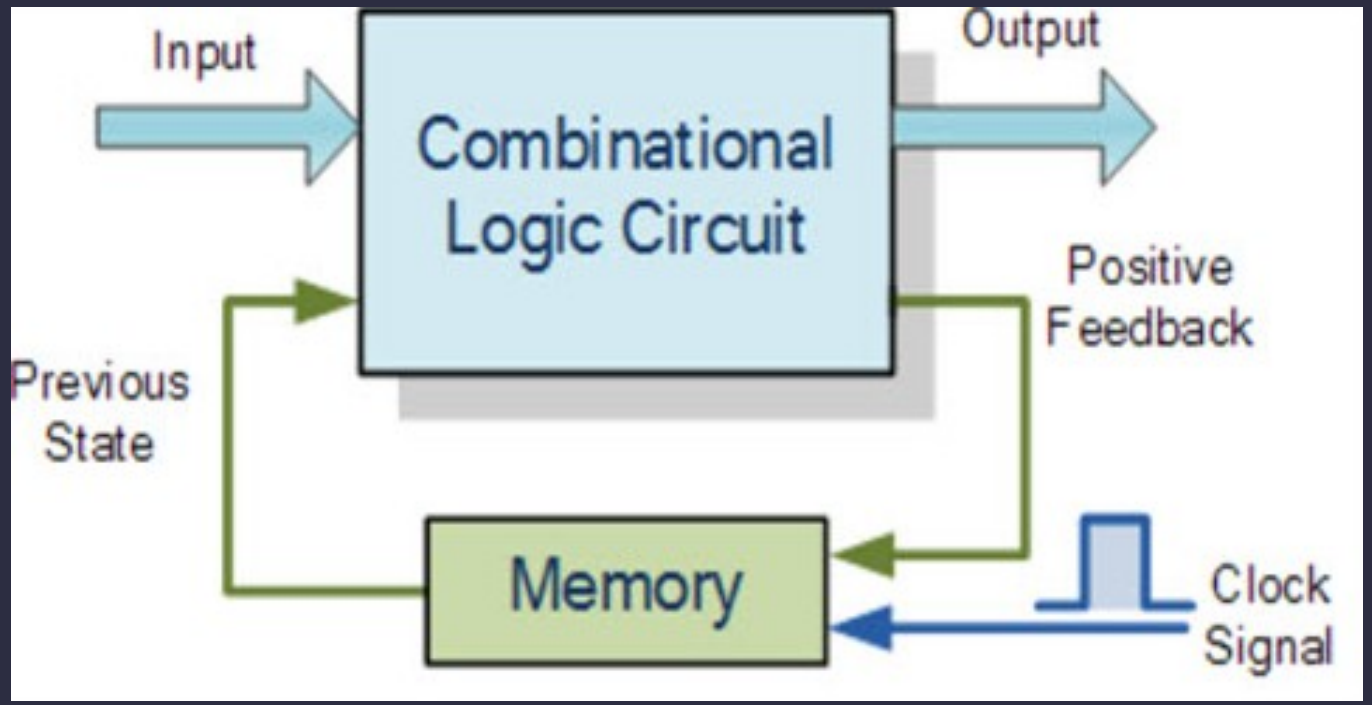


Sequential Circuits



Combinational Circuit and Sequential Circuit with Register

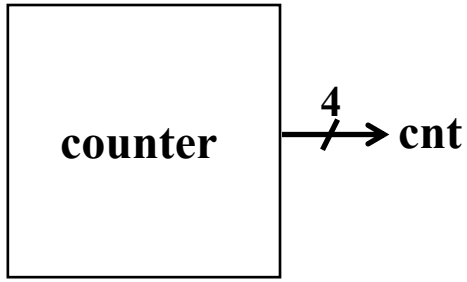
Sequential Circuits



Sequential Circuits

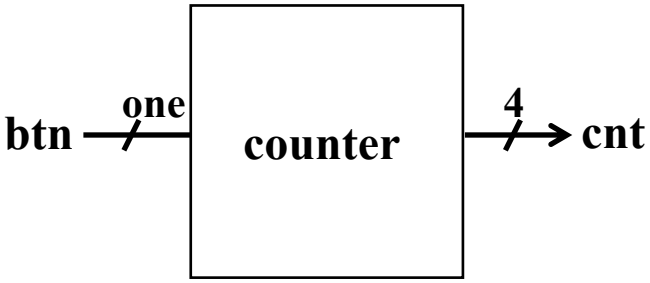

```
module counter();

endmodule
```

Sequential Circuits



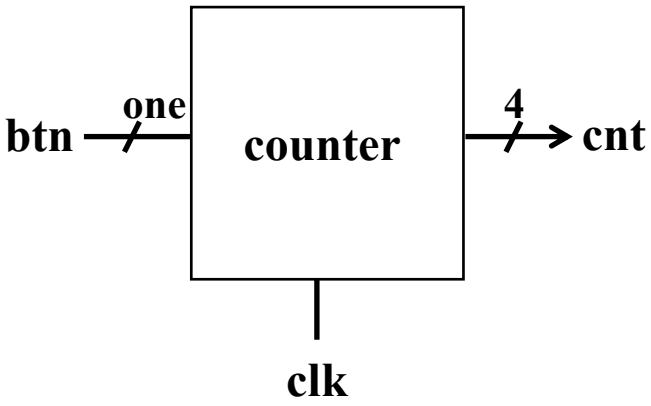

```
module counter(
cnt
);
output [3:0] cnt;

endmodule
```

Sequential Circuits



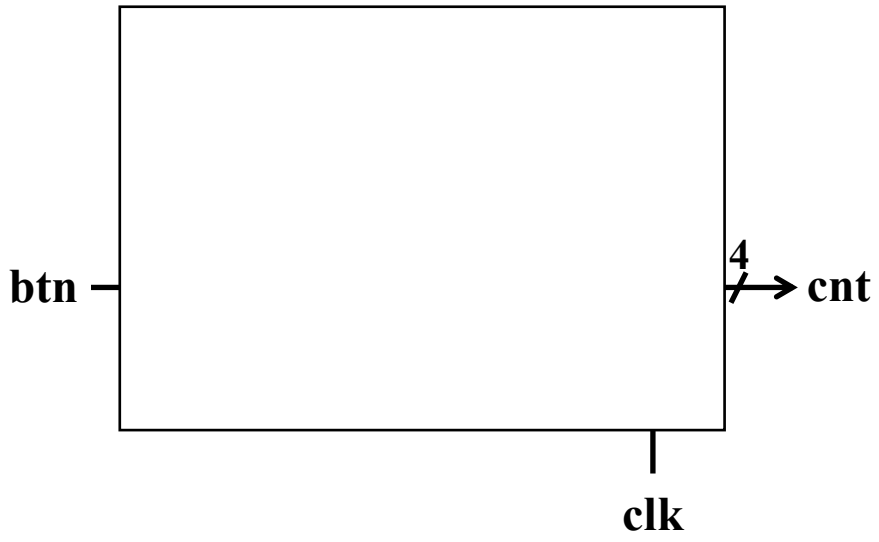

```
module counter(
cnt,
btn
);
output [3:0] cnt;
input btn;

endmodule
```

Sequential Circuits



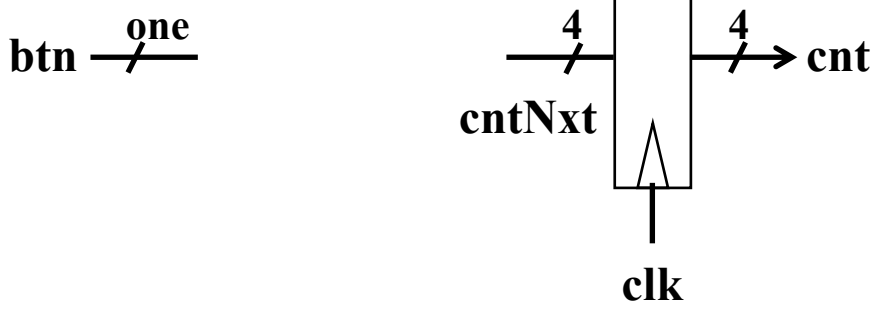

```
module counter(cnt, btn, clk);
output [3:0] cnt;
input btn, clk;

endmodule
```

Sequential Circuits



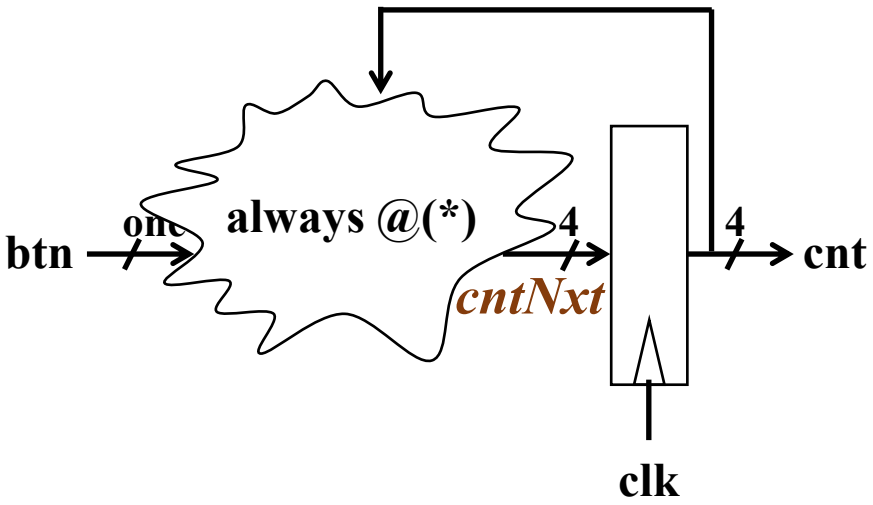

```
module counter(cnt, btn, clk);
output [3:0] cnt;
input btn, clk;

endmodule
```

Sequential Circuits



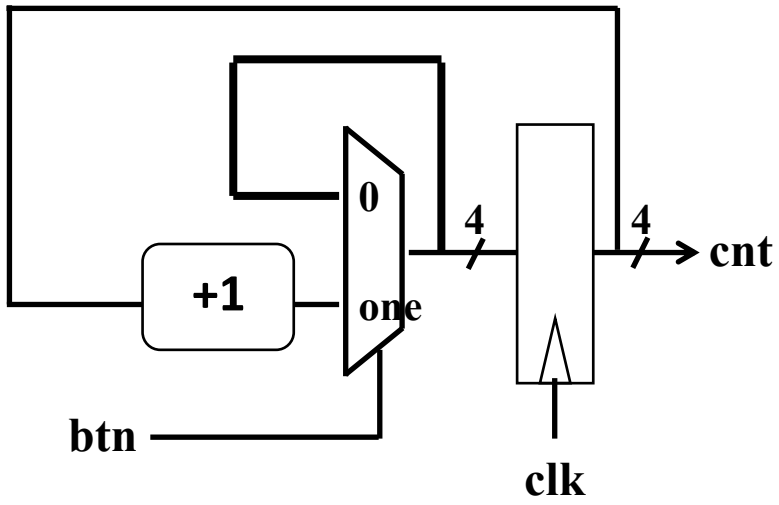

```
module counter(cnt, btn, clk);
output [3:0] cnt;
input btn, clk;

reg [3:0] cnt, cntNxt;

always @(posedge clk) begin
cnt <= #1 cntNxt;
end

endmodule
```

Sequential Circuits



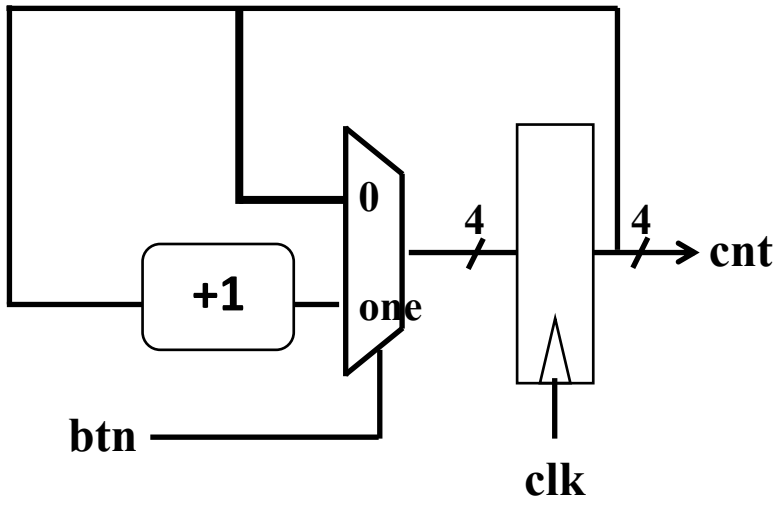

```
module counter(cnt, btn, clk);
output [3:0] cnt;
input btn, clk;

reg [3:0] cnt, cntNxt ;

always @(posedge clk) begin
cnt <= #1 cntNxt;
end

always @(*) begin
if(btn)
        cntNxt = cnt +1;
end

endmodule
```

Sequential Circuits




```
module counter ( cnt , btn , clk );
   output [3:0] cnt ;
   input btn , clk ;

   reg [3:0] cnt , cntNxt ;

    always @(posedge clk ) begin
       cnt <= #1 cntNxt ;
    end

    always @(*) begin
       if ( btn )
           cntNxt = cnt +1;
    end

endmodule
```

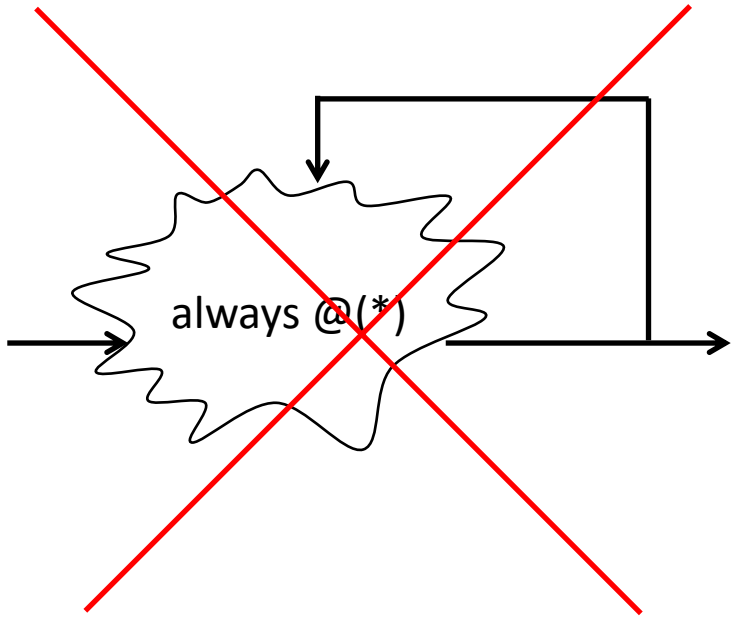
Sequential Circuits



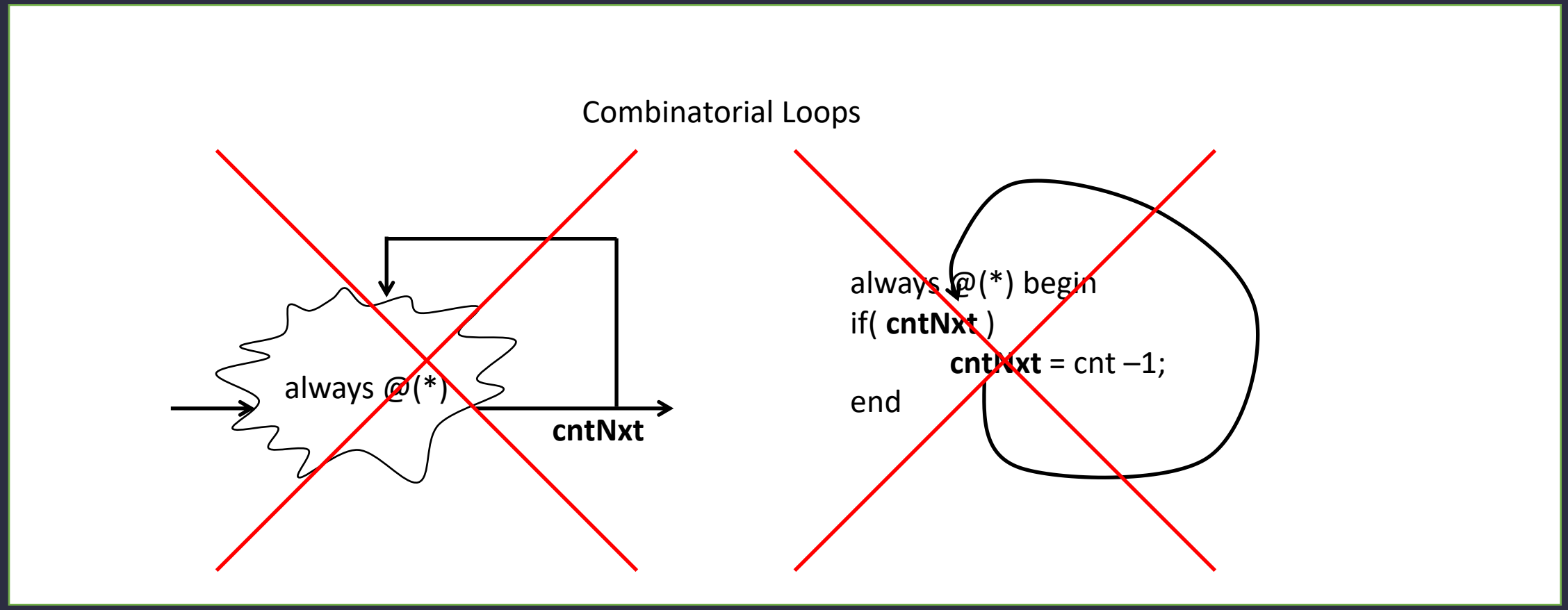

```
module counter(cnt, btn, clk);
output [3:0] cnt;
input btn, clk;

reg [3:0] cnt, cntNxt;

always @(posedge clk) begin
cnt <= #1 cntNxt;
end

always @(*) begin
cntNxt = cnt;
if(btn)
cntNxt = cnt +1;
end

endmodule
```

Sequential Circuits

Combinatorial Loops

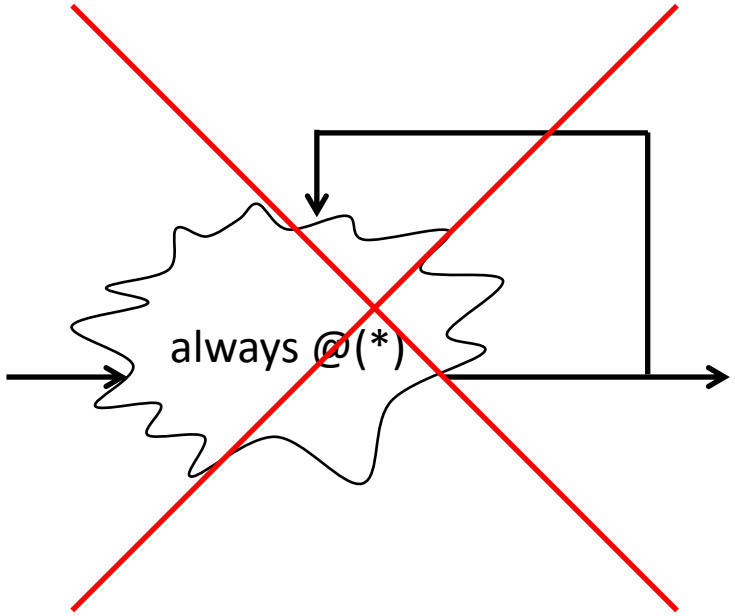


Sequential Circuits



Sequential Circuits

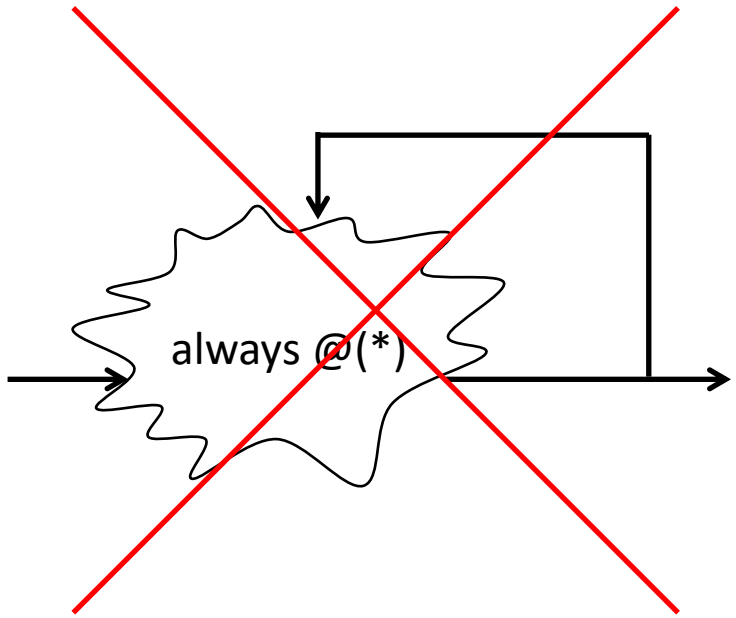
Combinatorial Loops

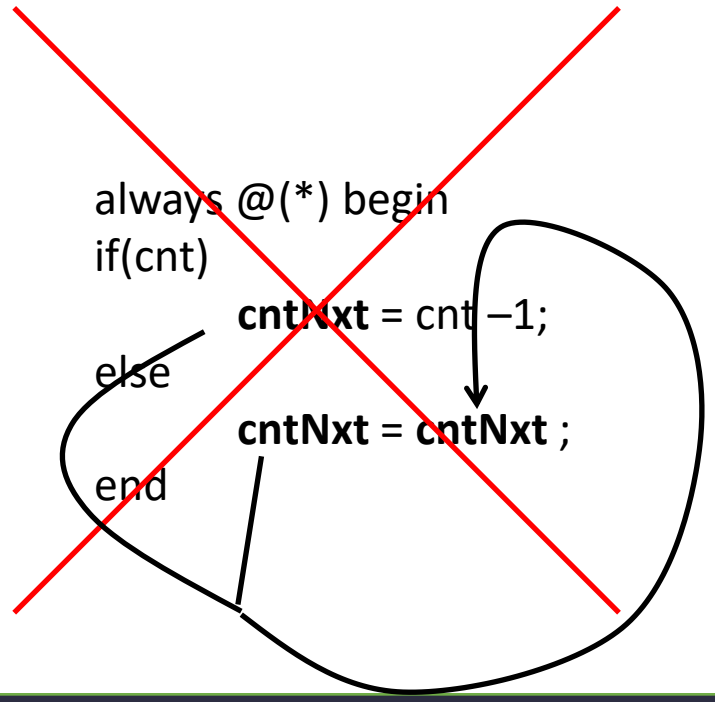



```
always @(*) begin
if( cnt )
        cntNxt = cnt –1;
end
```

Sequential Circuits

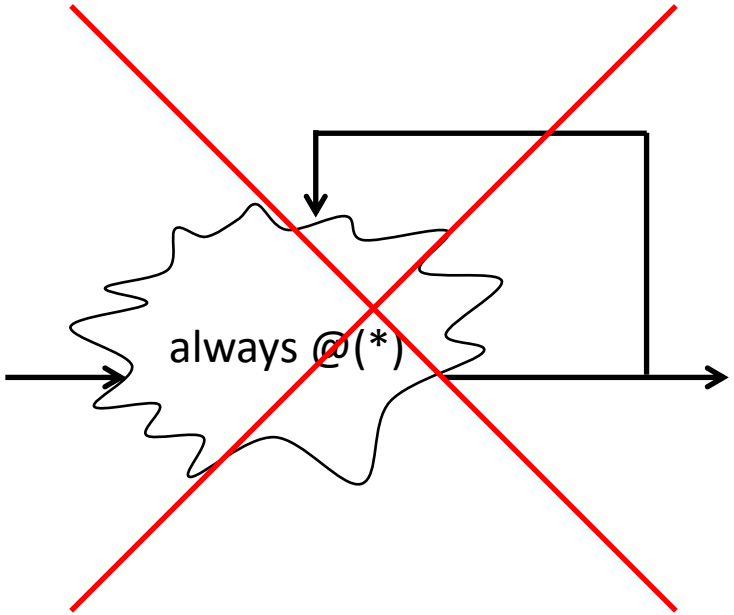
Combinatorial Loops





Sequential Circuits

Combinatorial Loops

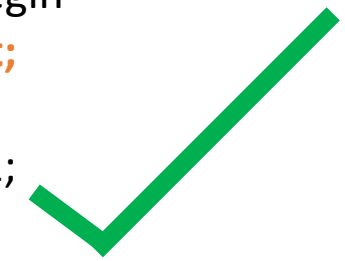



```
always @(*) begin
if(cnt)
cntNxt = cnt –1;
else
    cntNxt = cnt ;
end
```

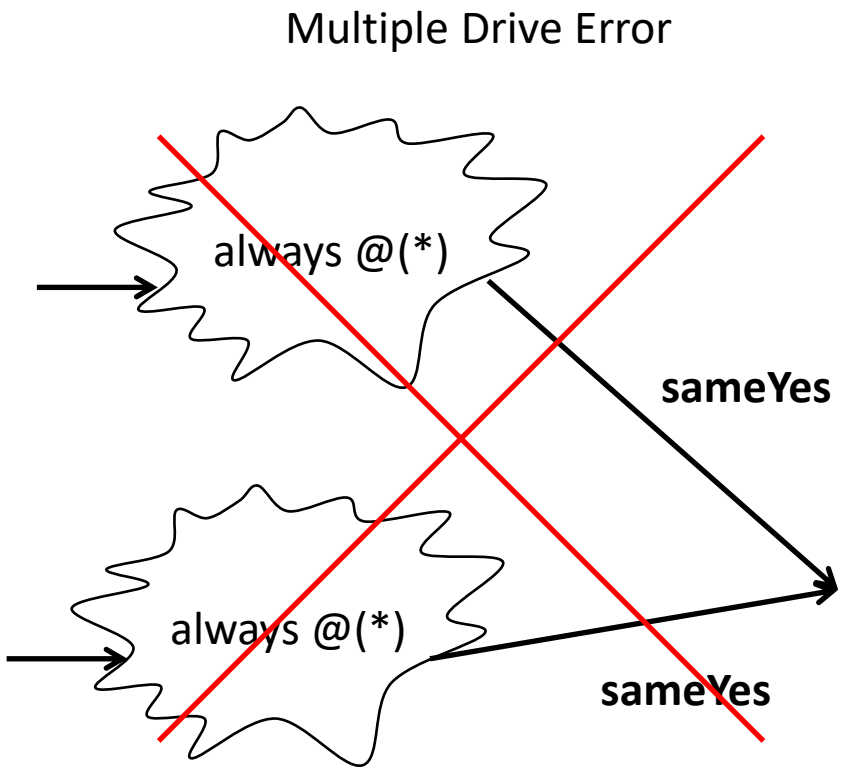
Sequential Circuits

Combinatorial Loops

```
always @(*) begin
    cntNxt = cnt;
if(cnt)
cntNxt = cnt –1;
end
```



Sequential Circuits



Sequential Circuits

Multiple Drive Error

```
always @(*) begin
    cntNxt = cnt ;
    if (btn1)
        cntNxt = cnt +1;
end

always @(*) begin
    cntNxt = cnt ;
    if (btn2)
        cntNxt = cnt –1;
end
```

Sequential Circuits

Multiple Drive Error

// Combined into a single always block

```
always @(*) begin
    cntNxt = cnt ;
    if (btn1)
        cntNxt = cnt +1;
    if (btn2)
        cntNxt = cnt –1;
end
```