Digital Circuit Design

Li Bai

Module 8

- Counter/Divider
  - Master clock with a frequency and convert it to another frequency.
- Ripper counter
- Counter IC

Clock stimulus in verilog

- Generate a clock
- Timescale (resolution)
- 1MHz clock

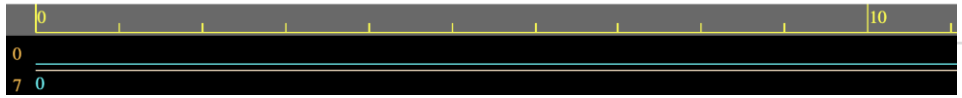
```
`timescale 1ns/1ns
```

```
// generate the clock stimulus
always begin
  clk = 0;
  #5;
  clk = 1;
  #5;
end
```

`timescale compiler directive

```
// or browse Examples
`timescale 1ns/1ns  // define the time unit resolution and
precision
```

Time resolution and precision



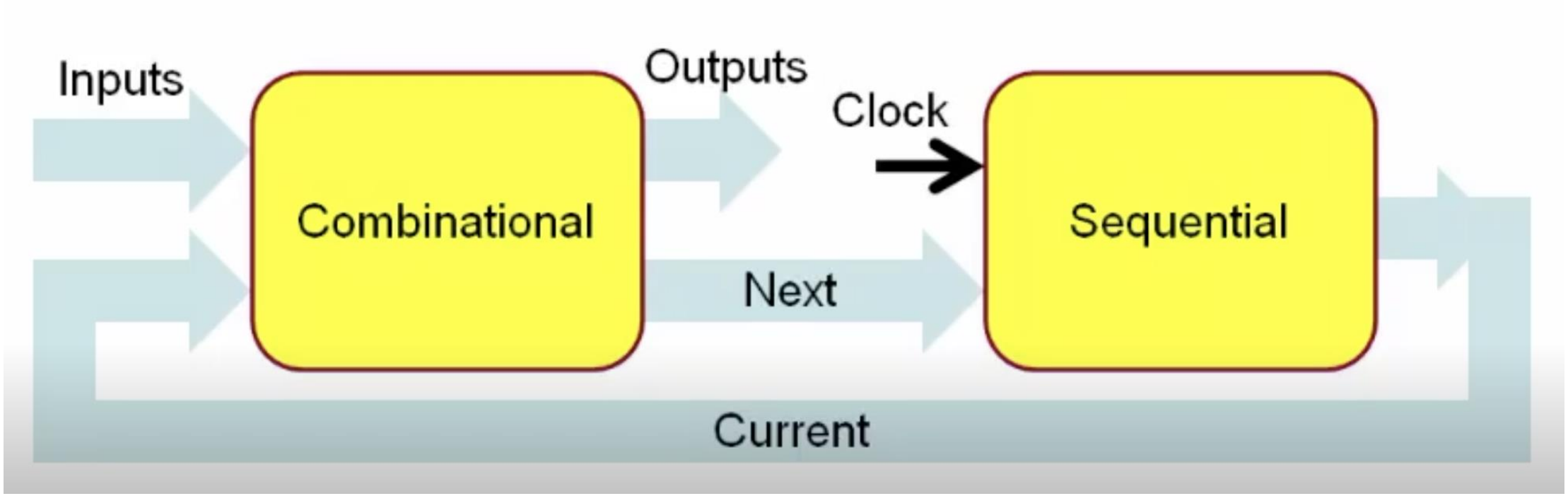
There are only **three valid magnitudes** for use in `time_unit` arguments. They are:

> `1`, `10`, and `100`.

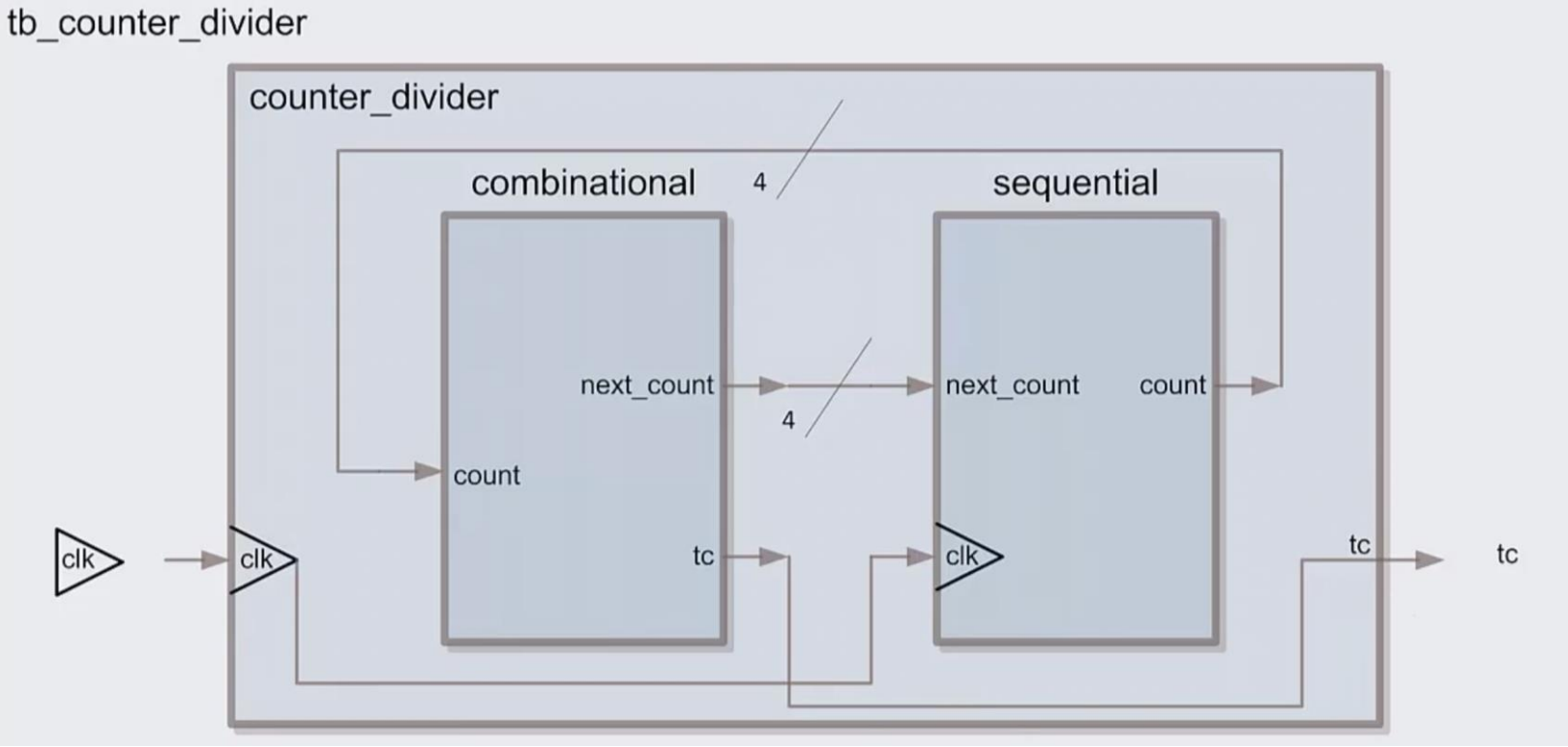
https://www.edaplayground.com/x/CJEF

Valid Time unit (in code)	Corresponding real-world time unit
`s`	seconds
`ms`	milliseconds
`us`	microseconds (μs)
`ns`	nanoseconds
`ps`	picoseconds
`fs`	femtoseconds

General sequential logic

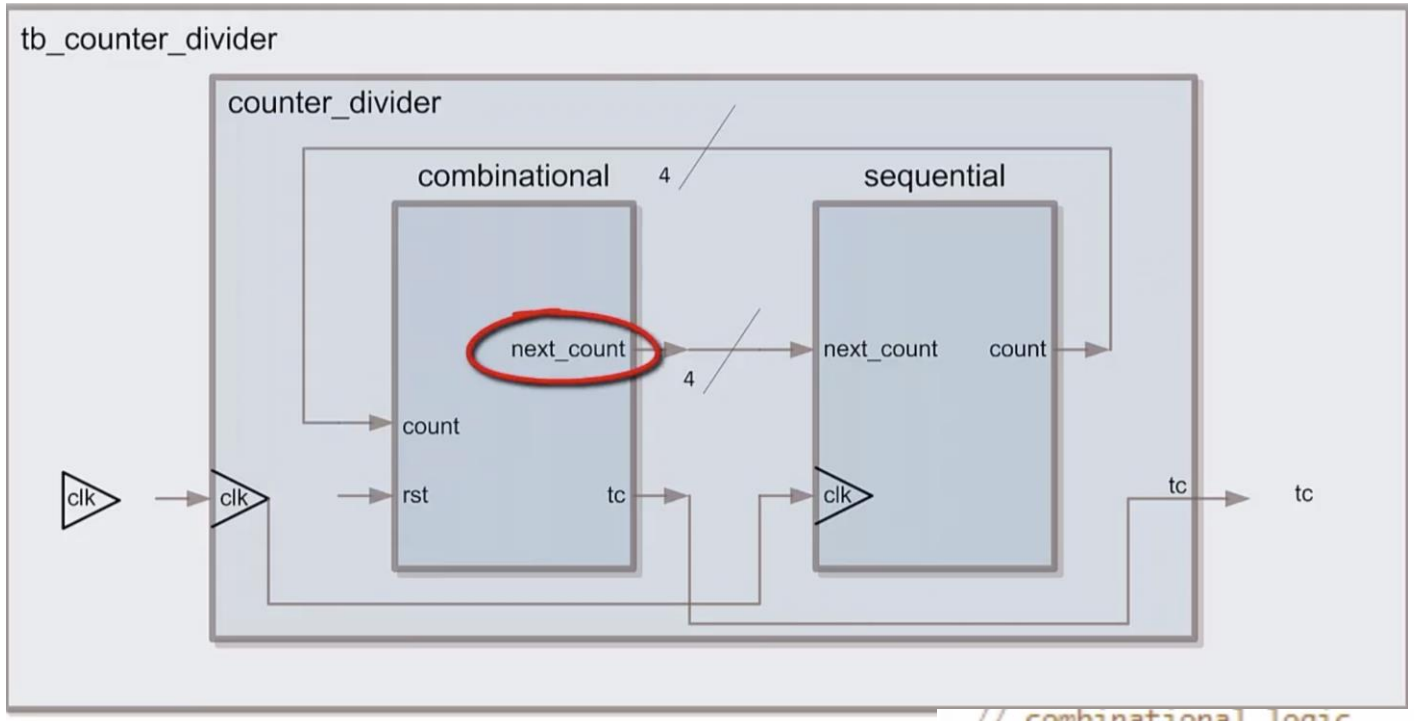


Tb_counter_divider



https://www.edaplayground.com/x/B9E

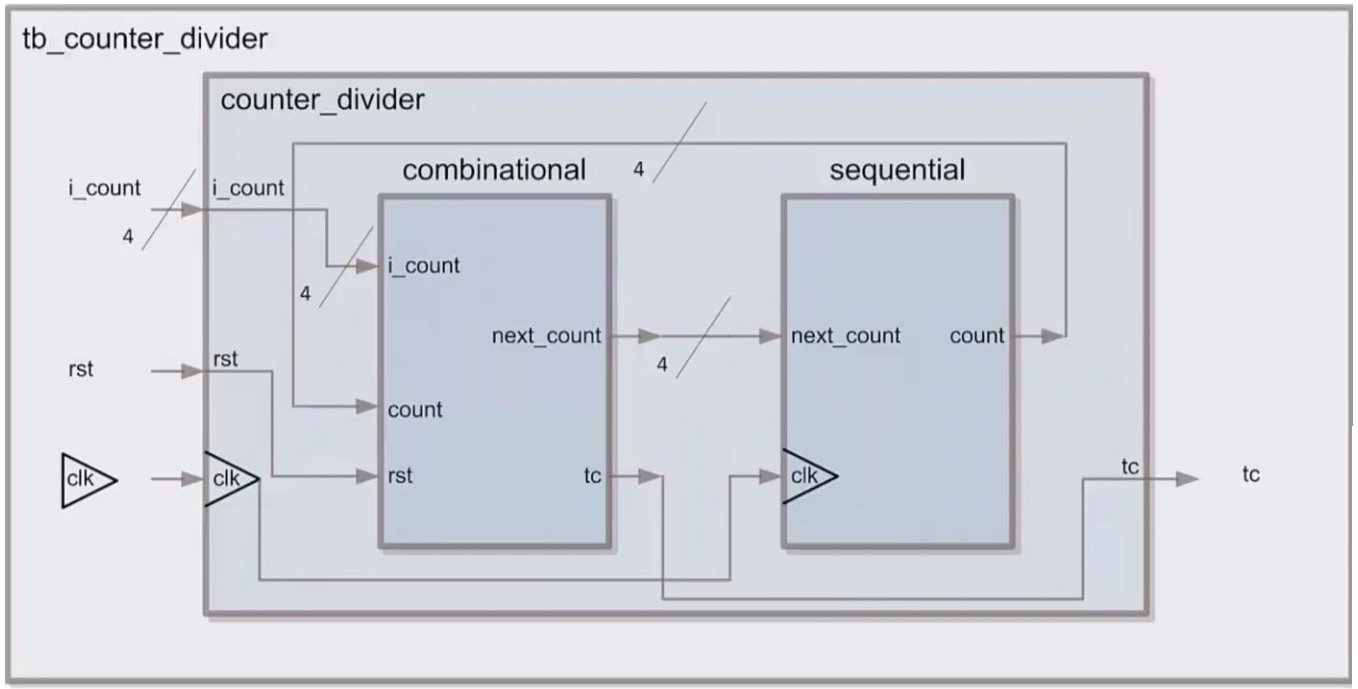
Counter_divider with reset (priority logic)




```
// combinational logic
always @* begin
  if (rst == 1) next_count = 0;
  // defaults
  tc = 0;
  // count down
  next_count = count - 1;
  if (count == 0) tc = 1; // set tc to 1 if count = 0
end
```

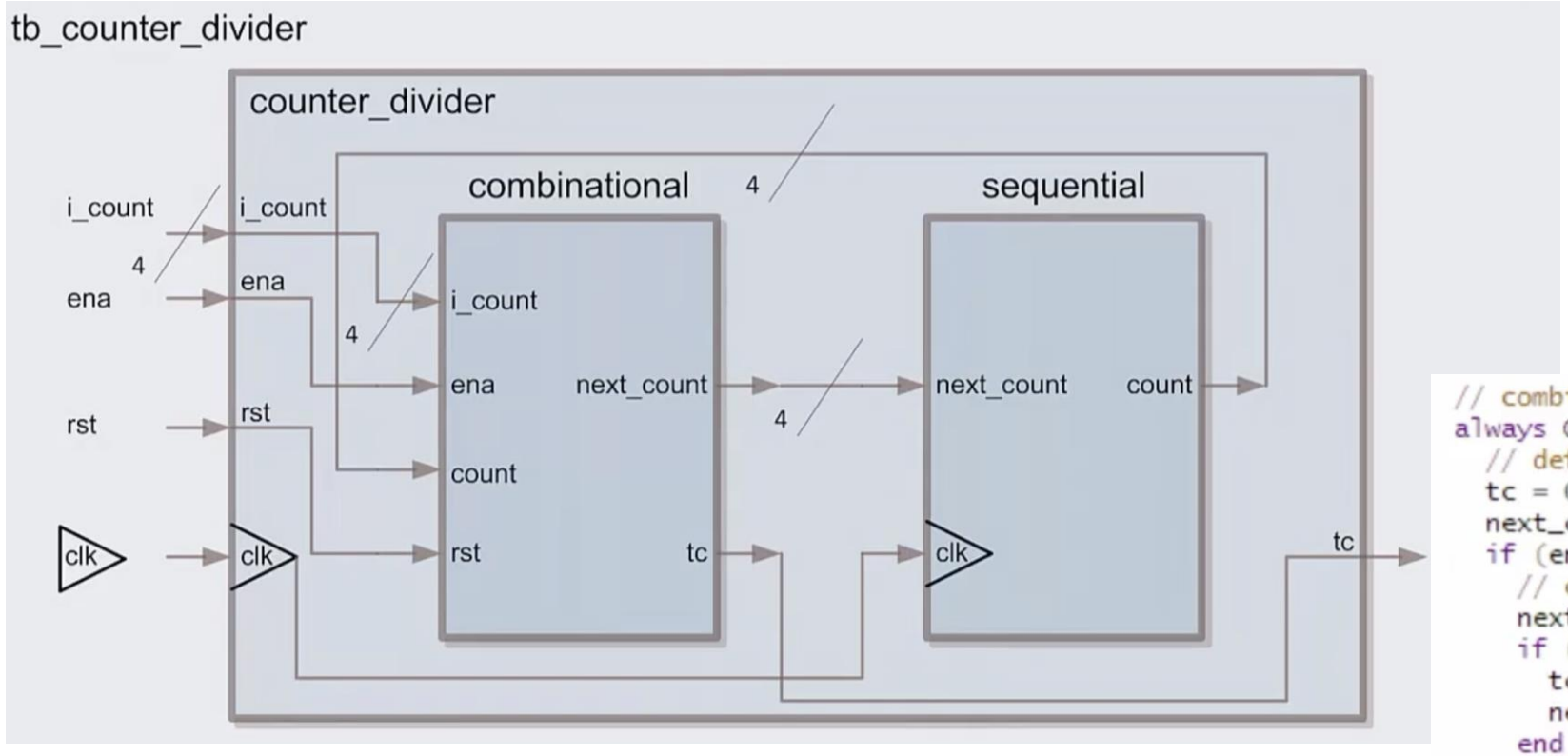
```
// combinational logic
always @* begin
  // defaults
  tc = 0;
  // count down
  next_count = count - 1;
  if (count == 0) tc = 1; // set tc to 1 if count = 0
  // priority logic
  if (rst == 1) next_count = 0;
end
```

Add input_counter to comb logic




```
always @* begin
  // defaults
  tc = 0;
  // count down
  next_count = count - 1;
  if (count == 0) begin
    tc = 1;    // set tc to 1 if count = 0
    next_count = i_count; // and initialize to i_count
  end
  // priority logic
  if (rst == 1) begin
    next_count = 0;
    next_count = i_count; // and initialize to i_count
  end
```

Counter_divider with ena



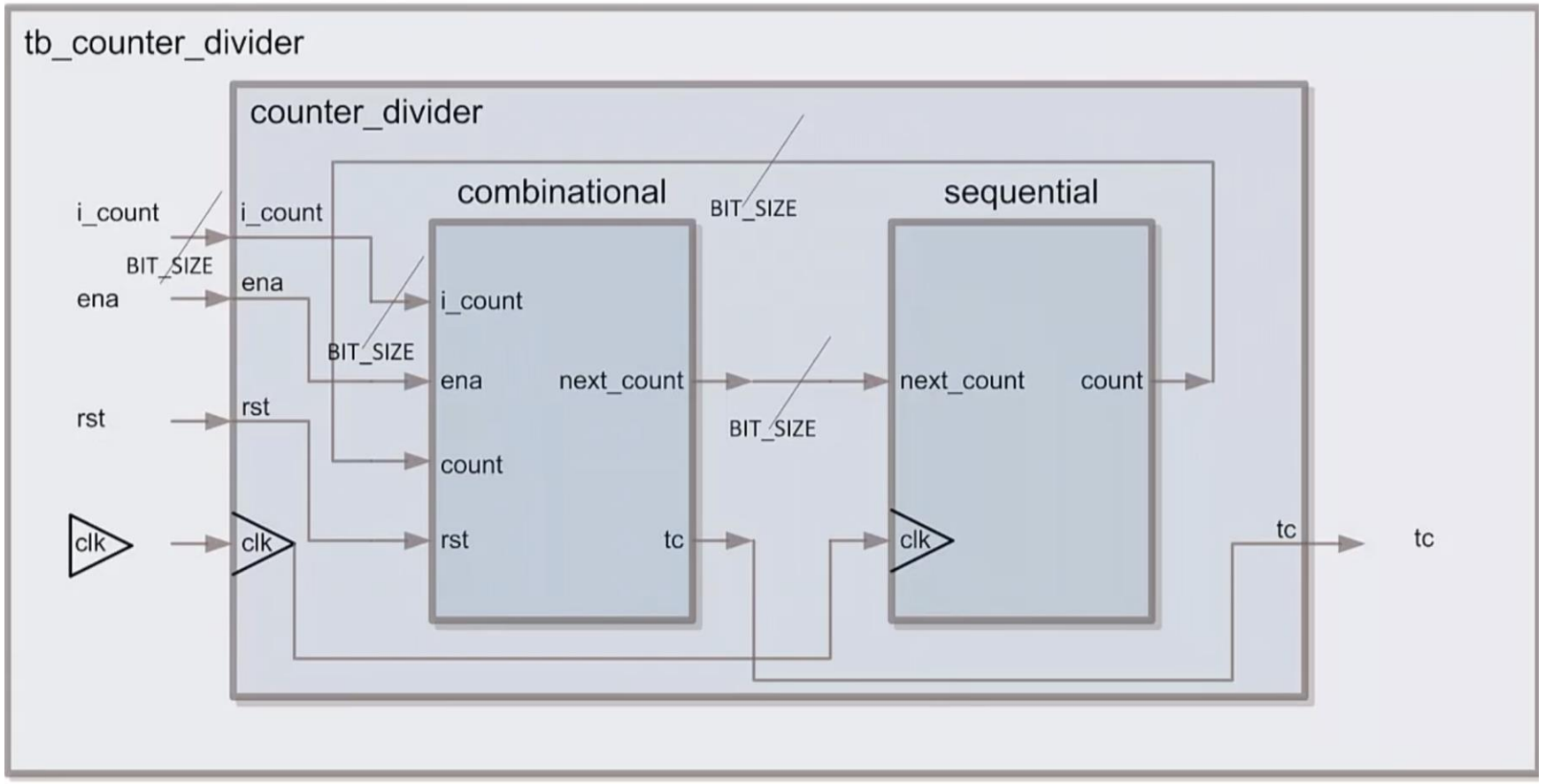

```
// combinational logic
always @* begin
  // defaults
  tc = 0;
  next_count = count; // hold the count
  if (ena == 1) begin
    // count down
    next_count = count - 1;
    if (count == 0) begin
      tc = 1; // set tc to 1 if count = 0
      next_count = i_count;    // and initialize to i_count
    end   // end of count = 0 if
  end // end of enable if
  // priority logic
  if (rst == 1) begin
    next_count = 0;
    next_count = i_count; // and initialize to i_count
  end
end
```

Parameter - BITSIZE

```
module counter_divider #(parameter BIT_SIZE=4)
  (output reg tc, input wire clk, input wire rst,
   input wire [BIT_SIZE-1:0] i_count, input wire ena);

  reg [BIT_SIZE-1:0] count, next_count;

  // synchronous logic
  always @(posedge clk) begin
    count <= next_count;
  end
```

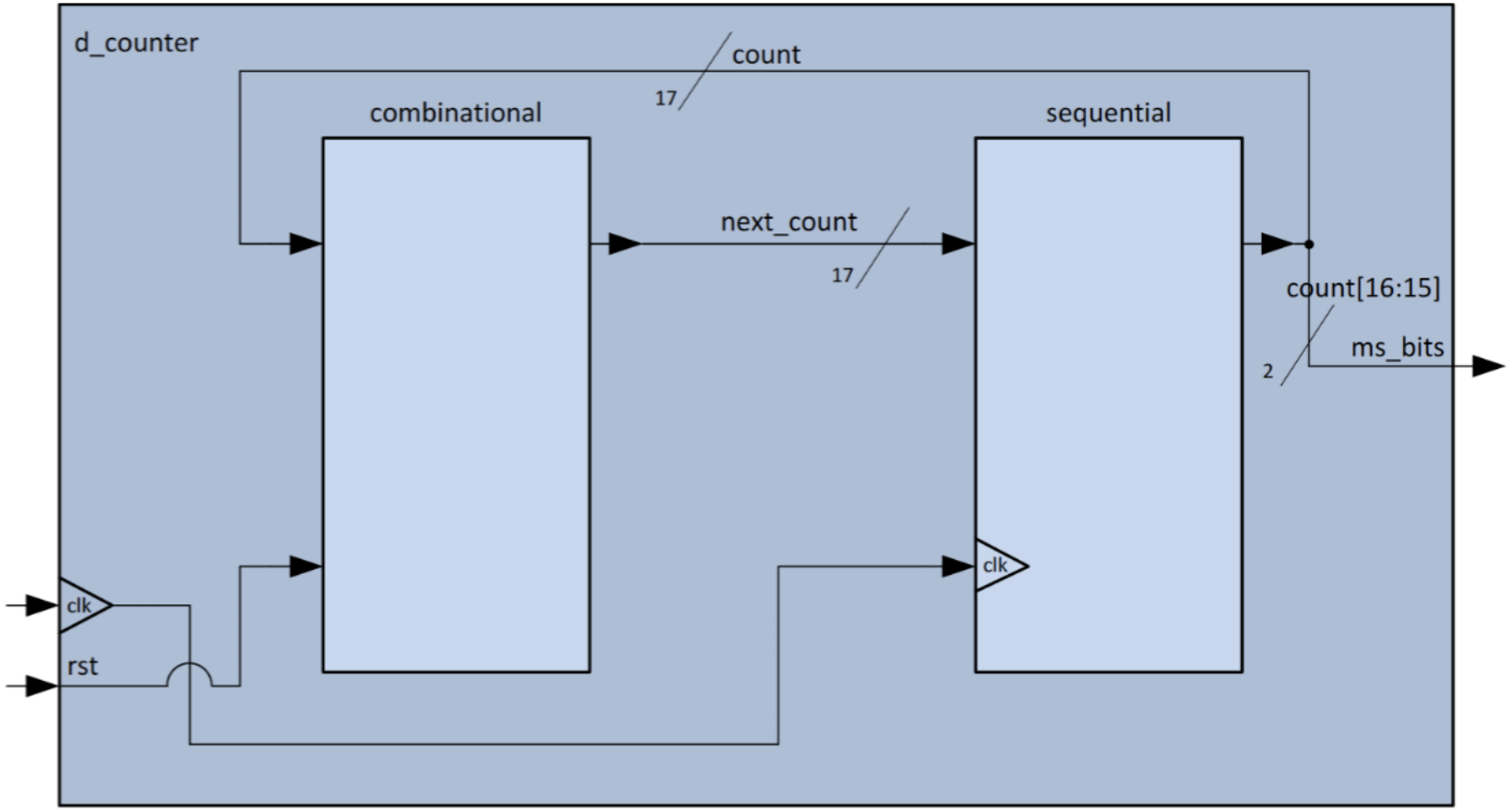
```
// instantiate the uut
counter_divider #(.BIT_SIZE(3)) uut
  (.tc(), .clk(clk), .rst(rst), .i_count(3'd4), .ena(ena));
```

A complete counter_divider



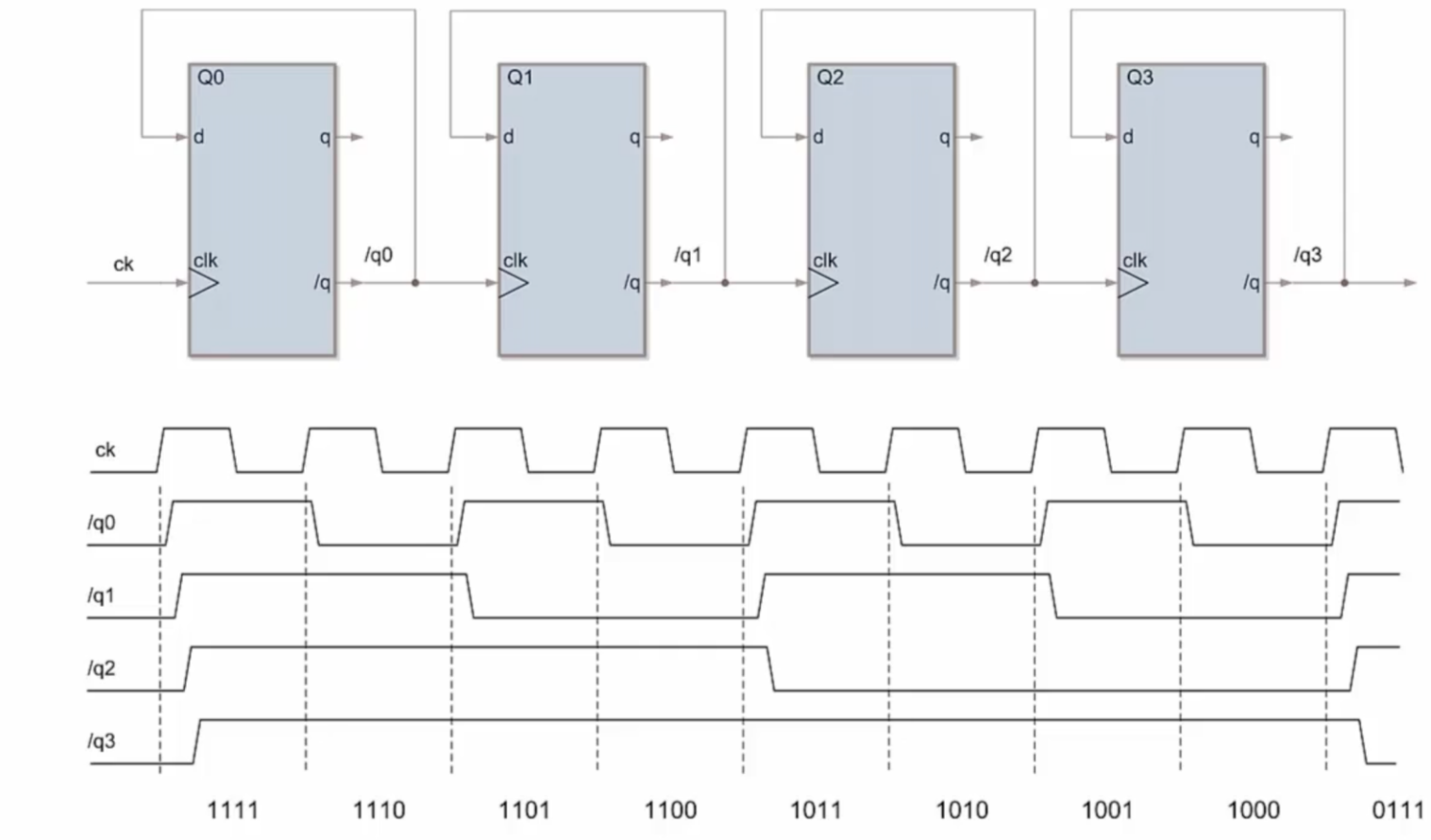
https://www.edaplayground.com/x/PxE

Counter Block Diagram

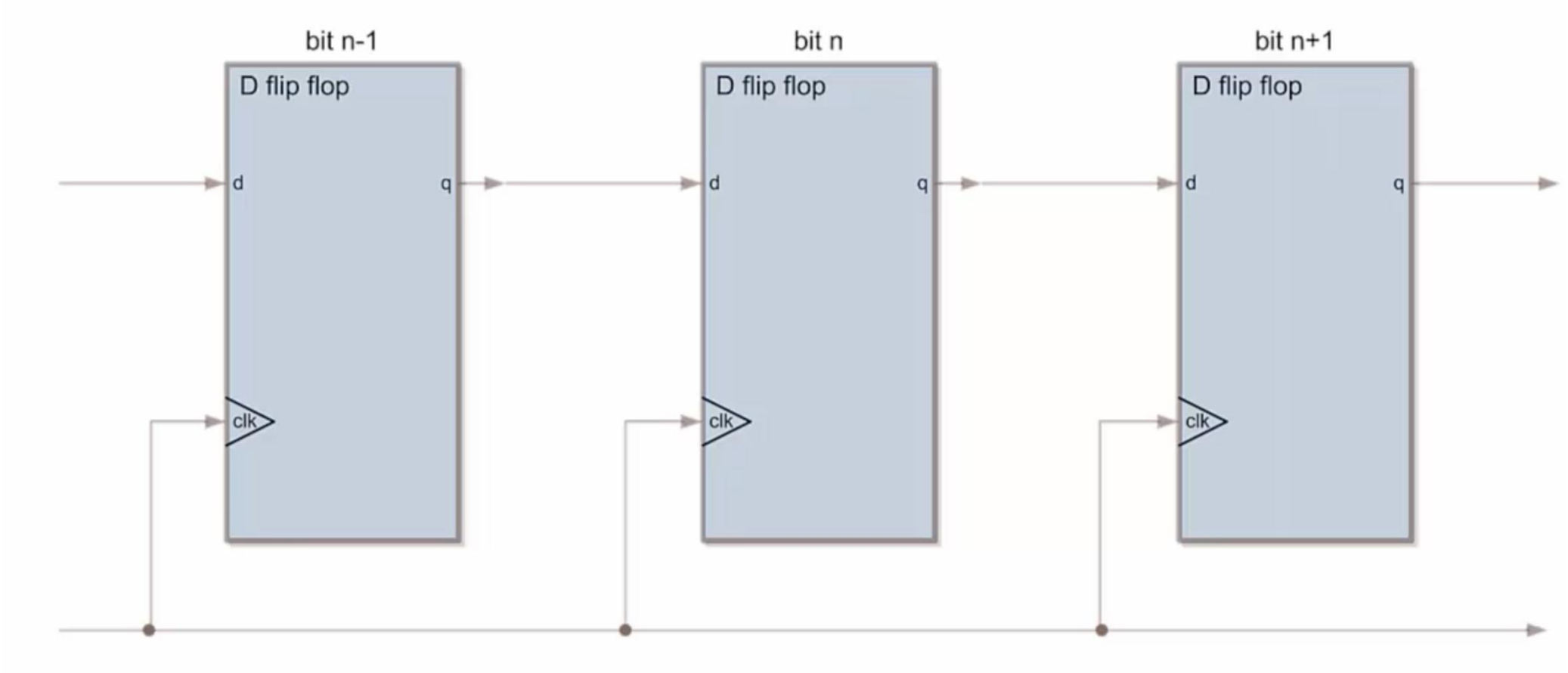



```
// Code your design here
`timescale 1ns/1ns
module cd_counter(output reg ms_bits, input wire clk, input wire rst);

 reg [16:0] count, next_count;

 // synchronous logic
 always @(posedge clk) begin
  count <= next_count;
 end

 // combinational logic
 always @* begin
  // defaults
 // count down
  next_count = count - 1;
 // priority logic
  if (rst == 1) next_count = 0;
 end

 assign ms_bits=count[16:15];
endmodule
```

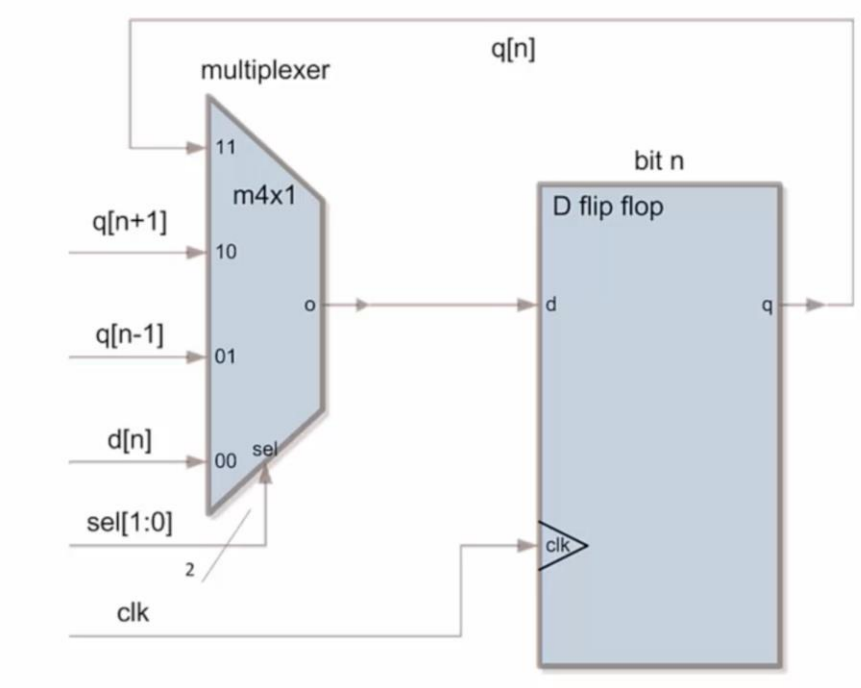
Ripper counter



Shift register



Circular shift using multiplexer




```
00_1000_1000     // load
01_0000_0100     // rotate down
01_0000_0010     // rotate down
01_0000_0001     // rotate down
01_0000_1000     // rotate down
```

```
// combinational logic
always @* begin
  // just a simple mux, since the select is fully deco
  case (sel)
    2'b00: next_q = d;     // load from the d port
    2'b01: next_q = {q[0],q[3:1]};     // rotate down
    2'b10: next_q = {q[2:0],q[3]};     // rotate up
    2'b11: next_q = q;     // hold
  endcase
end
```

```
01_0000_0011     // rotate down
01_0000_1001     // rotate down
```

Lookup an IC

- https://www.designfast.com/
- 4-bit counter

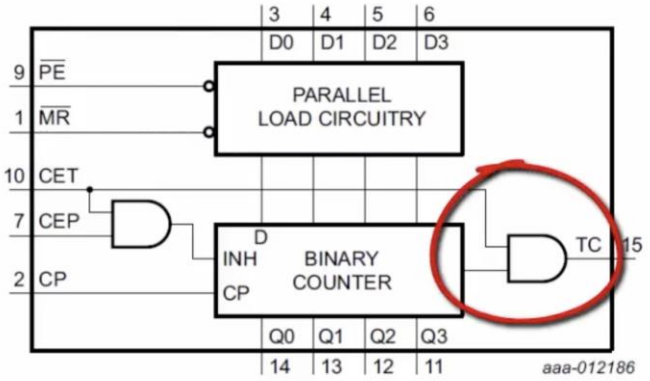


Fig 1. Functional diagram

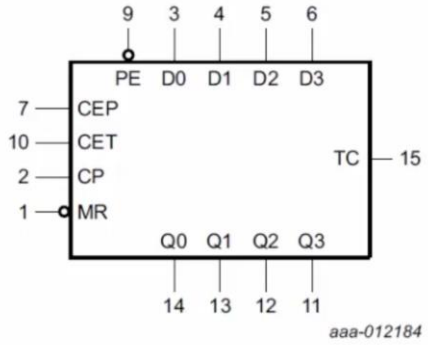
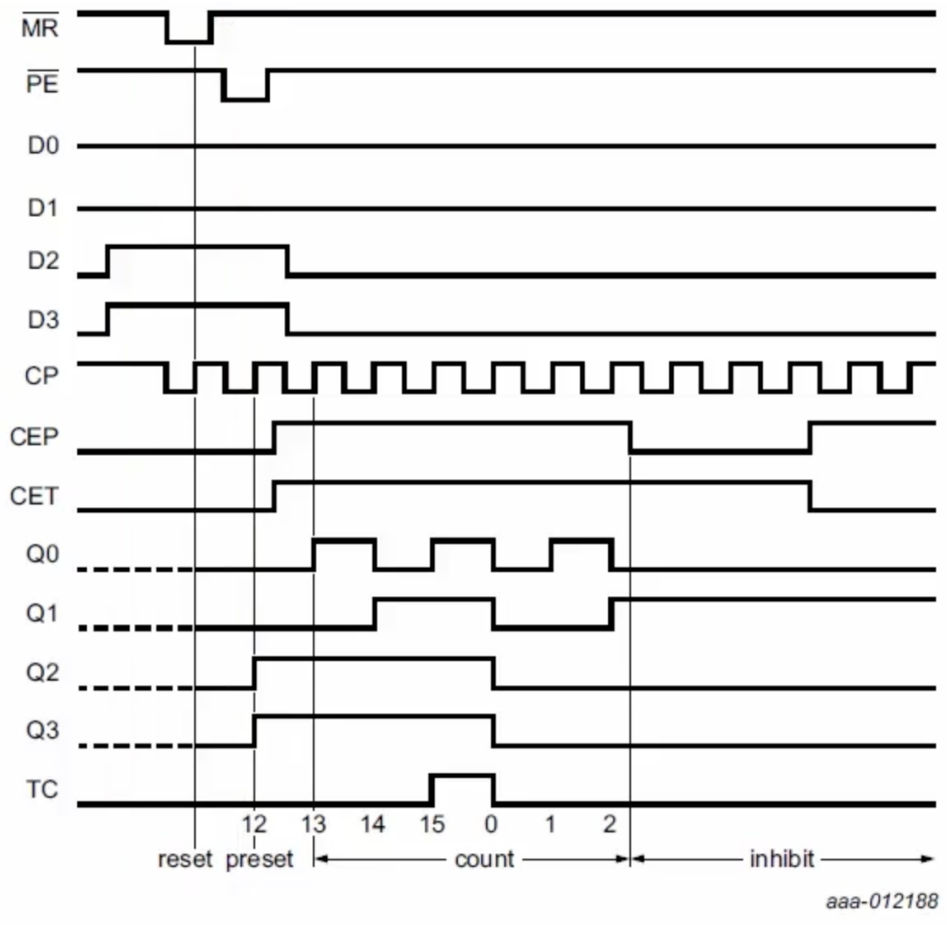


Fig 2. Logic symbol



How to use to module counter

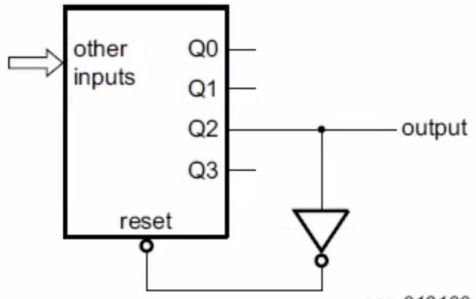


Fig 15. Modulo-5 counter

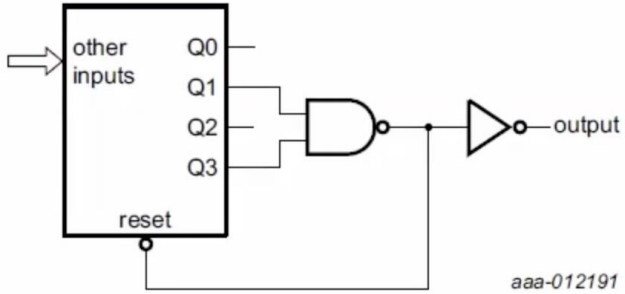


Fig 16. Modulo-11 counter