Digital System Design ELEC373/473

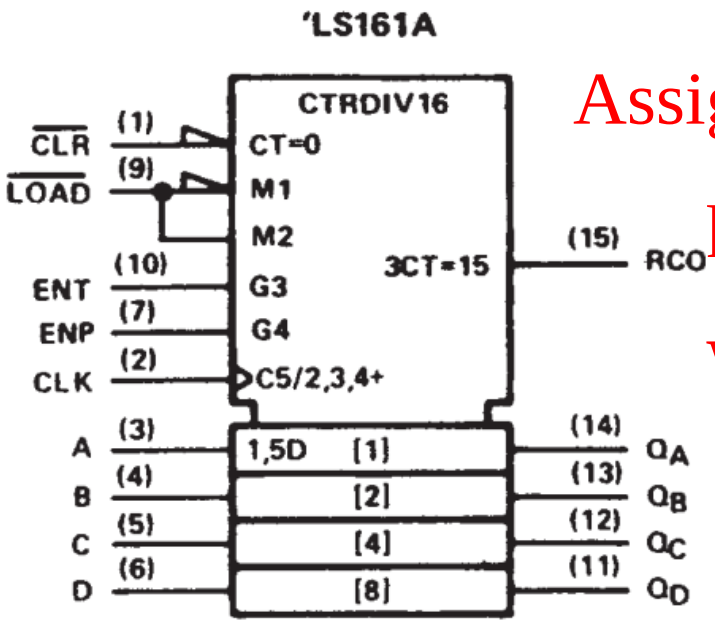
Counters in Verilog

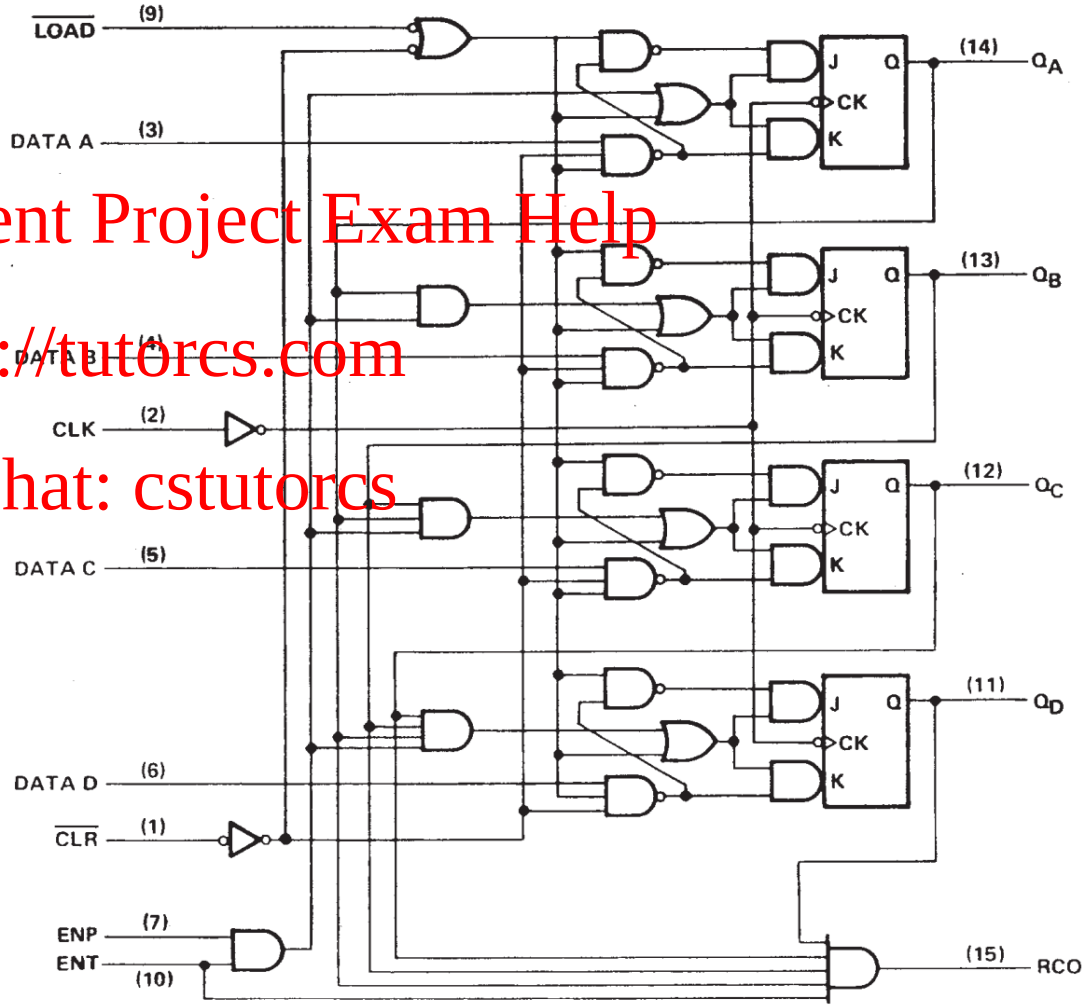
Prof J.S. Smith
Room A515;
E-mail: j.s.smith@liv.ac.uk

Counters

- Counters are probably one of the most heavily used components in digital systems.
- When circuits were built using the 74 series devices the common “synchronous” counters used were:
  - 74x160 - synchronous presettable 4-bit decade counter, asynchronous clear
  - 74x161 - synchronous presettable 4-bit binary counter, asynchronous clear
  - 74x162 - synchronous presettable 4-bit decade counter, synchronous clear
  - 74x163 - synchronous presettable 4-bit binary counter, synchronous clear
  - 74x190 - synchronous presettable up/down 4-bit decade counter
  - 74x191 - synchronous presettable up/down 4-bit binary counter

74x161- Schematic





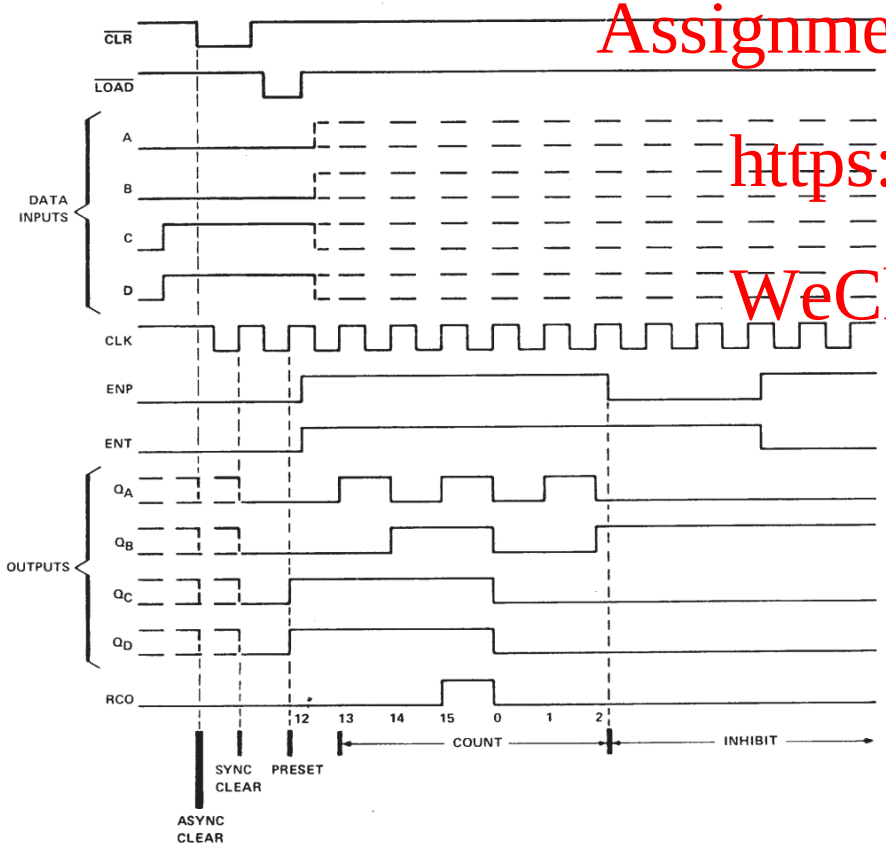
74x161- Timing Diagrams

'161, 'LS161A, '163, 'LS163A, 'S163 BINARY COUNTERS

typical clear, preset, count, and inhibit sequences

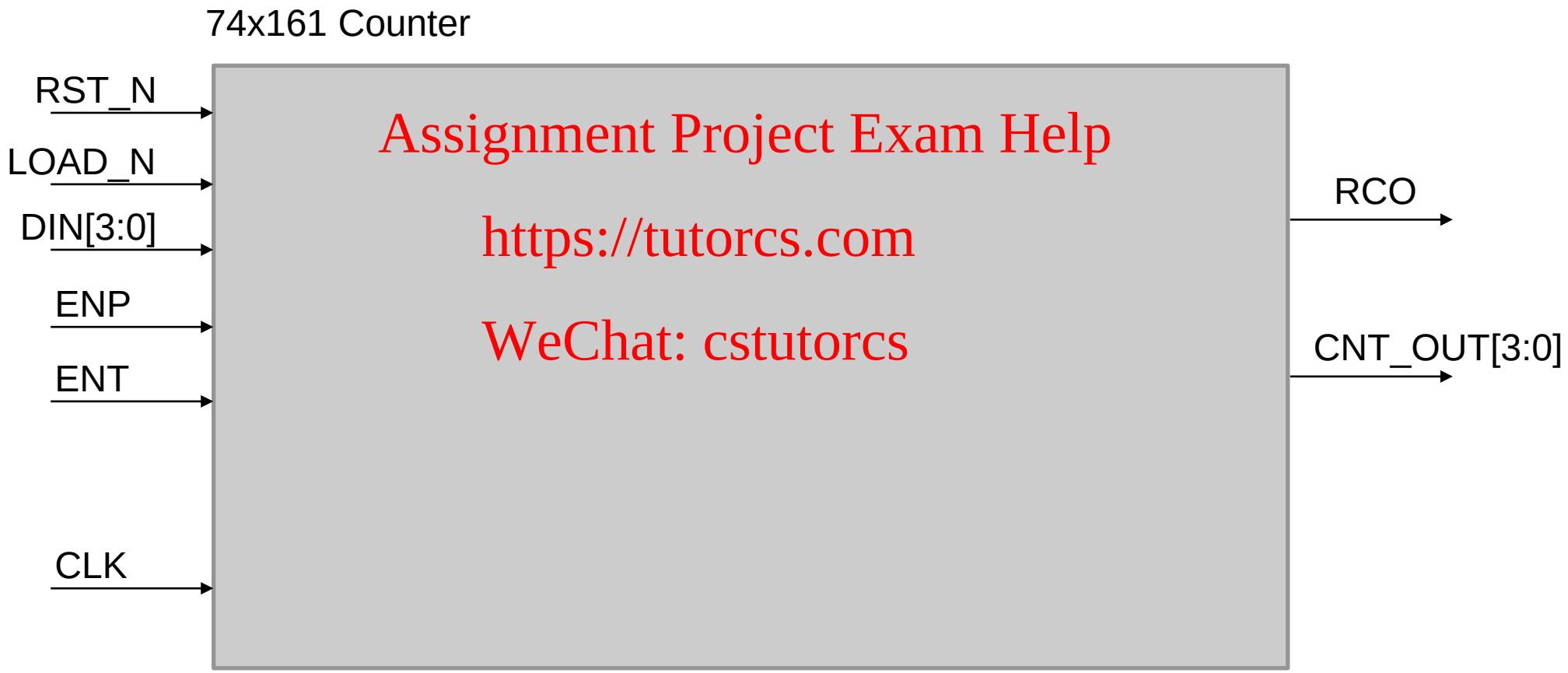
Illustrated below is the following sequence:

1. Clear outputs to zero ('161 and 'LS161A are asynchronous; '163, 'LS163A, and 'S163 are synchronous)
2. Preset to binary twelve
3. Count to thirteen, fourteen fifteen, zero, one, and two
4. Inhibit

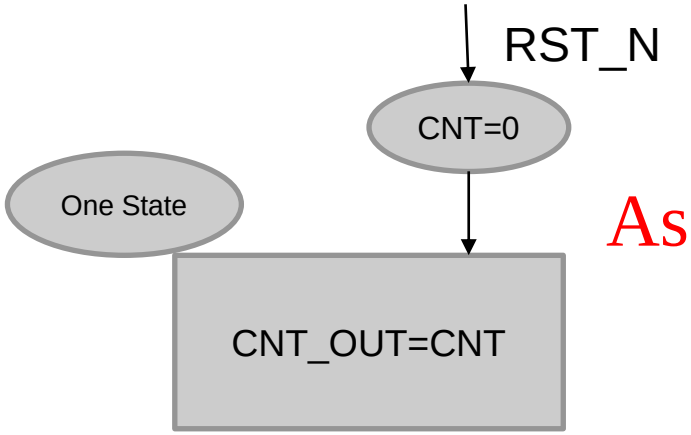


- 74x161 has asynchronous active low clear
- Active low synchronous Preset
- Synchronously counts when the Enables P & T are both true (ENT & ENP)
- Generates a Ripple Carry Out (RCO) when Count = 15 and ENT is TRUE.
- When cascaded to form larger counters i.e. 8 bit, 12 bit etc. the RCO of the lower bits is connected to the ENT of the next stage and ENP is treated as a global enable for all stages.

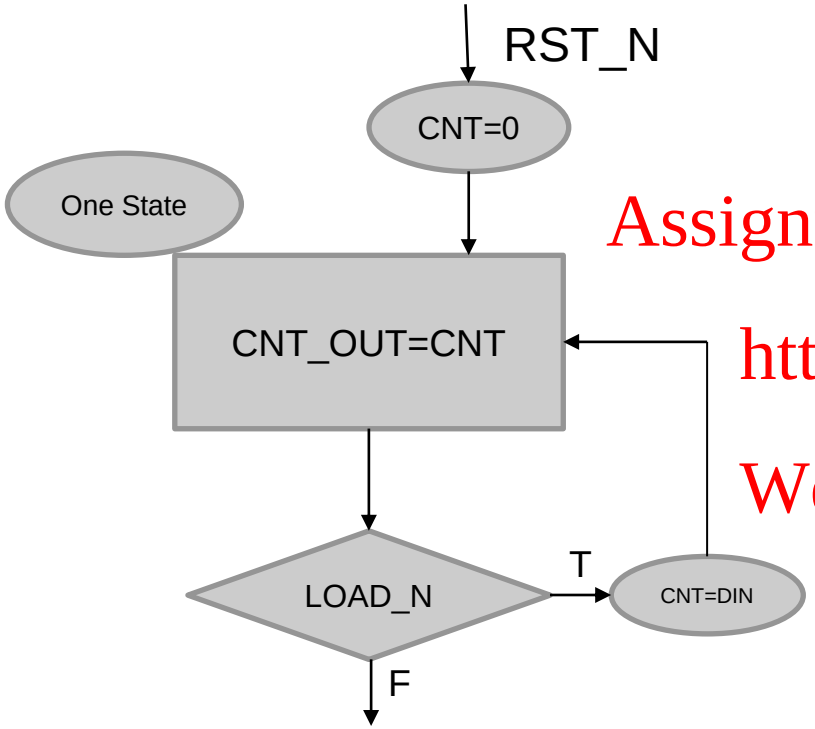
4 Bit Counter System Block Diagram



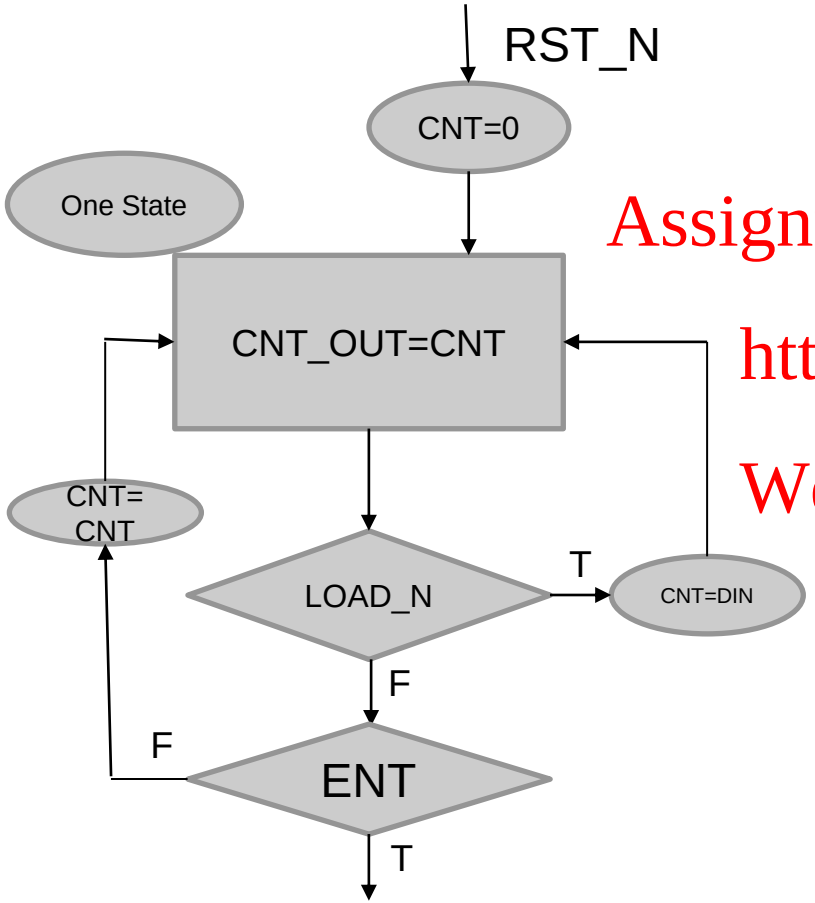
4 Bit counter ASM - 1



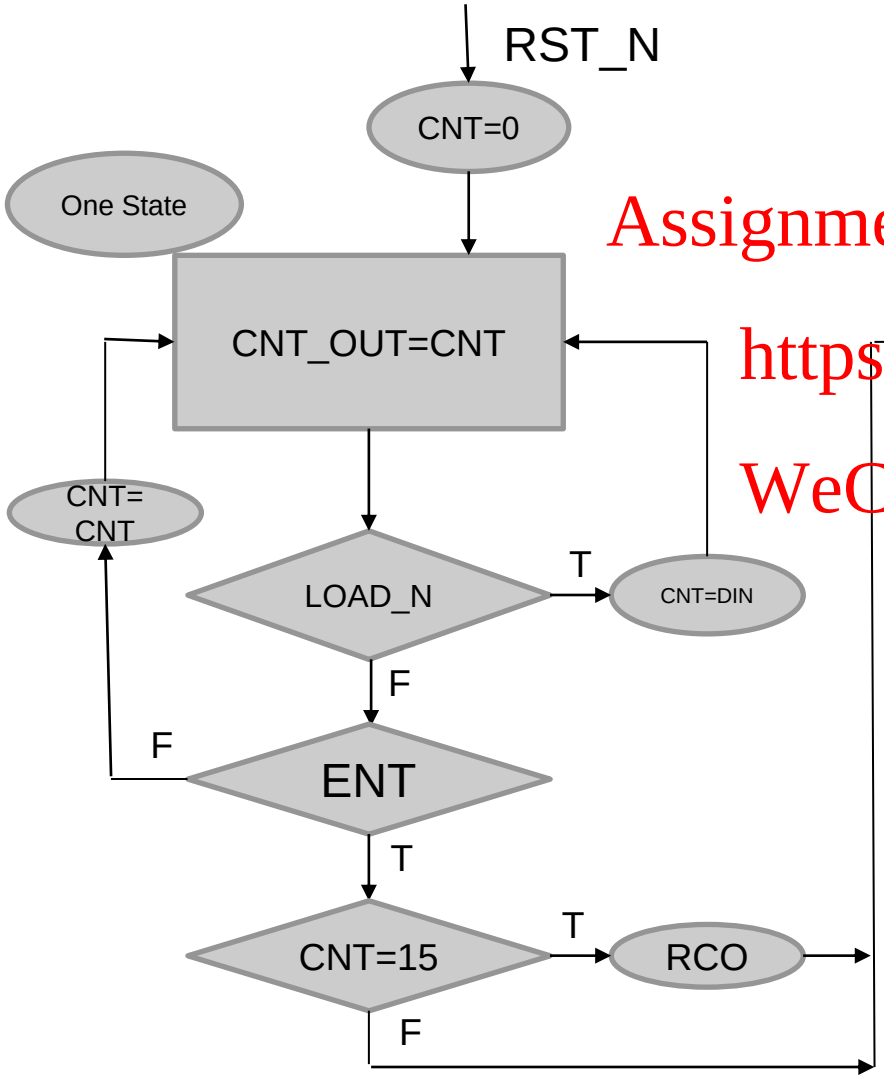
4 Bit counter ASM - 2



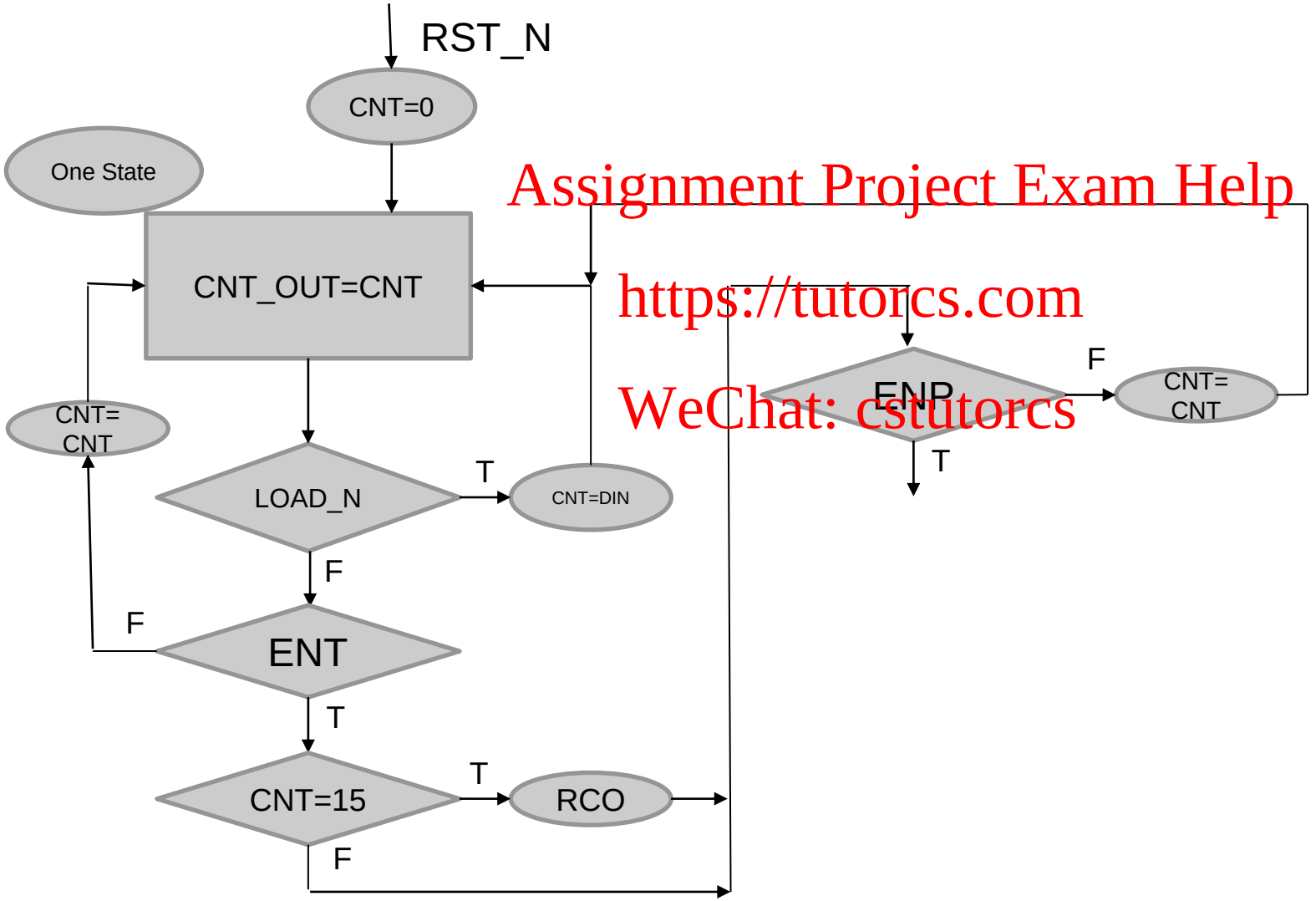
4 Bit counter ASM - 3



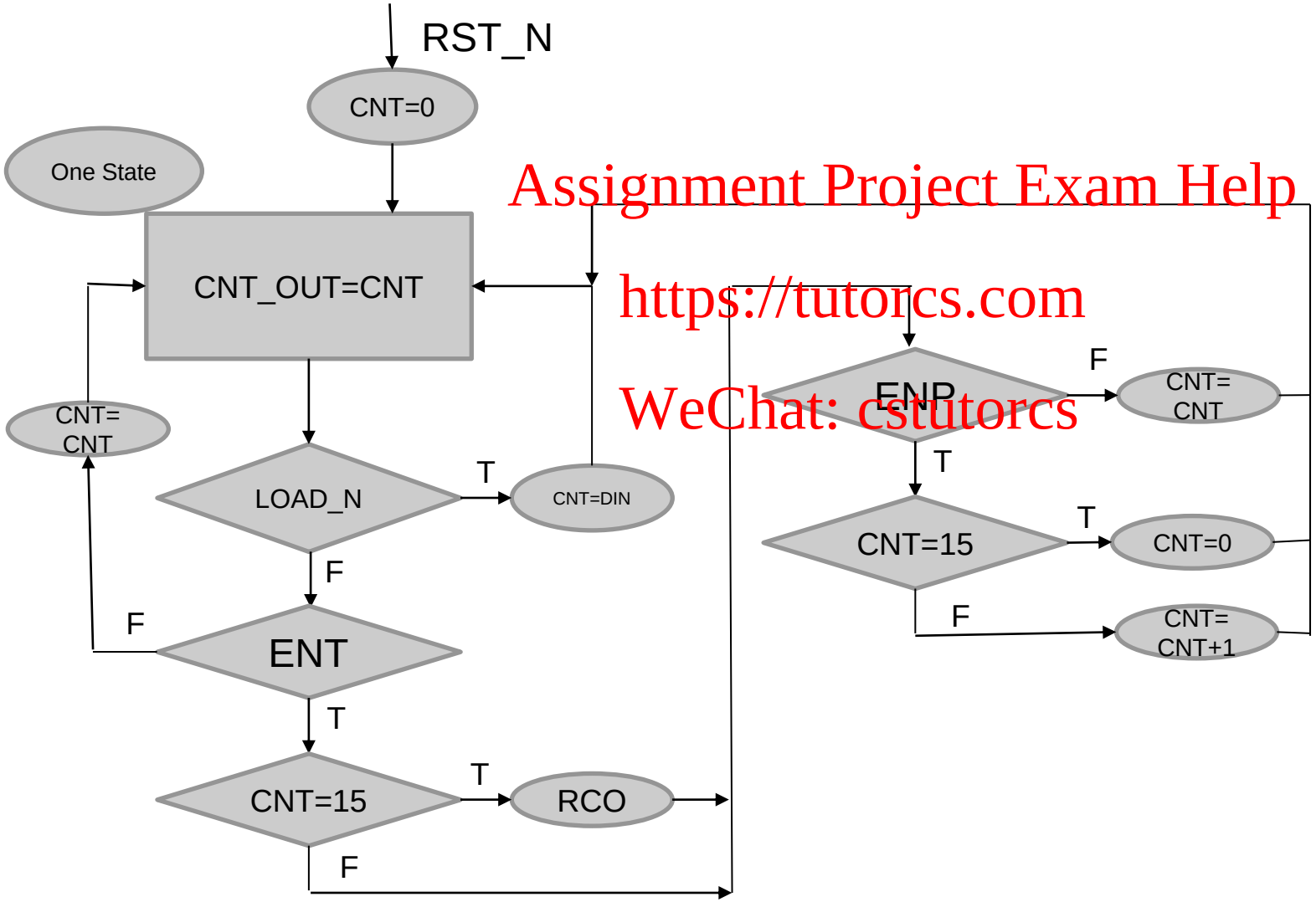
4 Bit counter ASM - 4



4 Bit counter ASM - 5



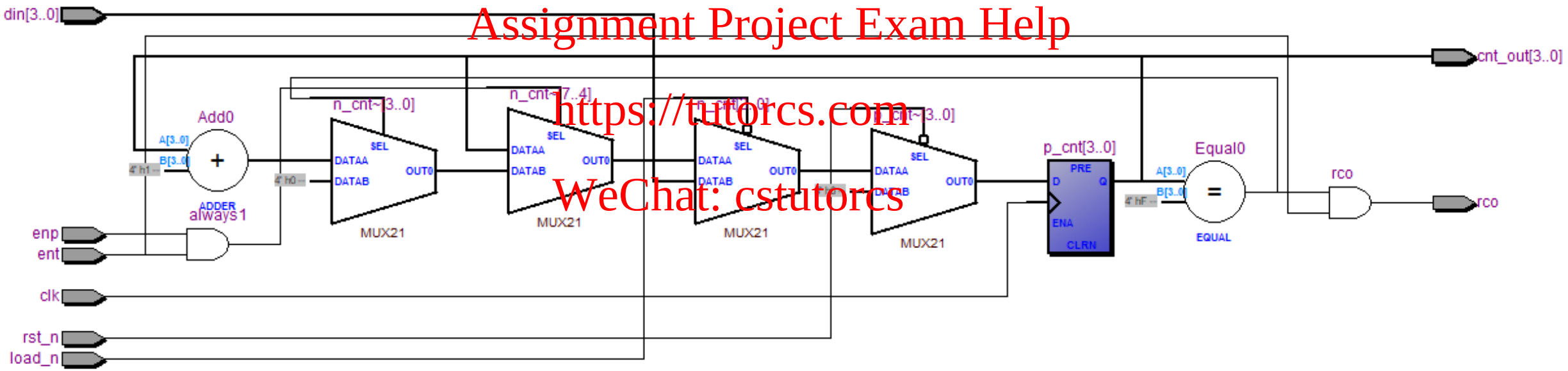
4 Bit counter ASM - 6



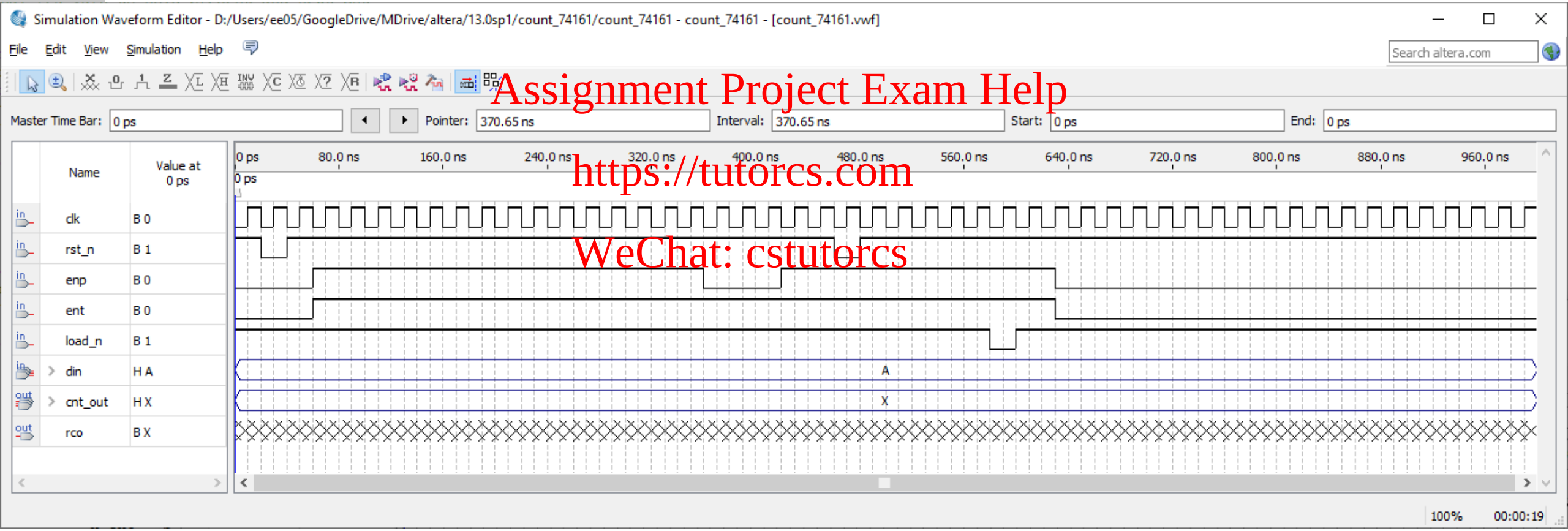
Verilog Code

```
1  module count_74161(output [3:0] cnt_out, output rco, input rst_n, load_n, clk, enp, ent, input [3:0] din);
2
3  // local reg vars to hold present and next cnt
4  reg [3:0] p_cnt, n_cnt;
5  // assign p_cnt to the output cnt_out
6  assign cnt_out = p_cnt;
7  // generate rco with assign when p_cnt = 15 & ent = 1
8  assign rco = (p_cnt == 4'd15) & ent;
9  //sequential block to synthesise flip flops
10 // note non-blocking assignments
11 always @ (posedge clk)
12    begin
13       //synchronous reset done here
14       if (rst_n == 1'b0)
15          p_cnt <= 4'b0;
16       else
17          p_cnt <= n_cnt;
18    end
19 // combinational block to generate n_cnt
20 // note using blocking assignments
21 always @ (load_n, enp, ent, din, p_cnt)
22    begin
23       // default to prevent latches
24       n_cnt = p_cnt;
25       if (load_n == 1'b0)
26          n_cnt = din;
27       else if (ent & enp)
28          begin
29             if (p_cnt == 4'd15)
30                n_cnt = 4'd0;
31             else
32                n_cnt = p_cnt + 1'b1;
33          end
34    end
35 endmodule
36
```

RTL View



Timing Simulation – input



Timing Simulation – output