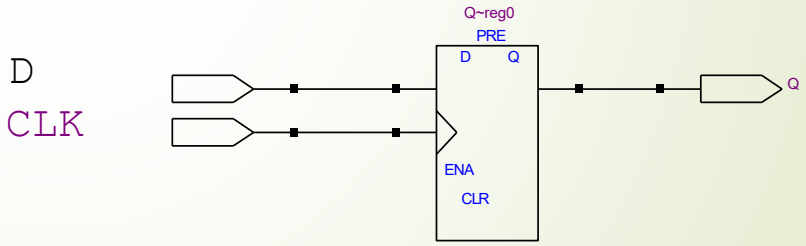
EGC221
Class Notes
4/26/2023

Baback Izadi
Division of Engineering Programs
bai@engr.newpaltz.edu

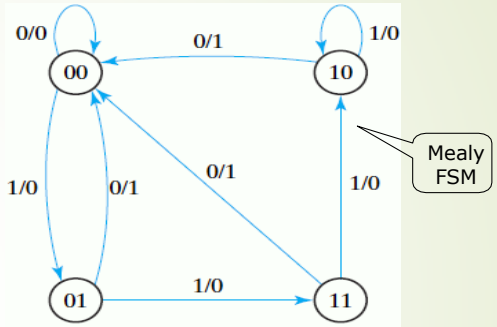
Verilog – D Flip-flop

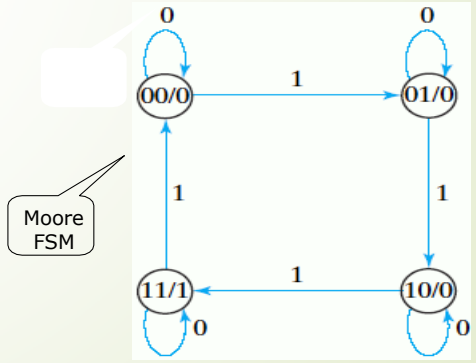
```
module D-flipflop (D, Clk, Q);

   input D, Clk;
   output Q;  reg Q;

   always @(posedge Clk)
      Q = D;

Endmodule
```



Finite State Machines (FSM)

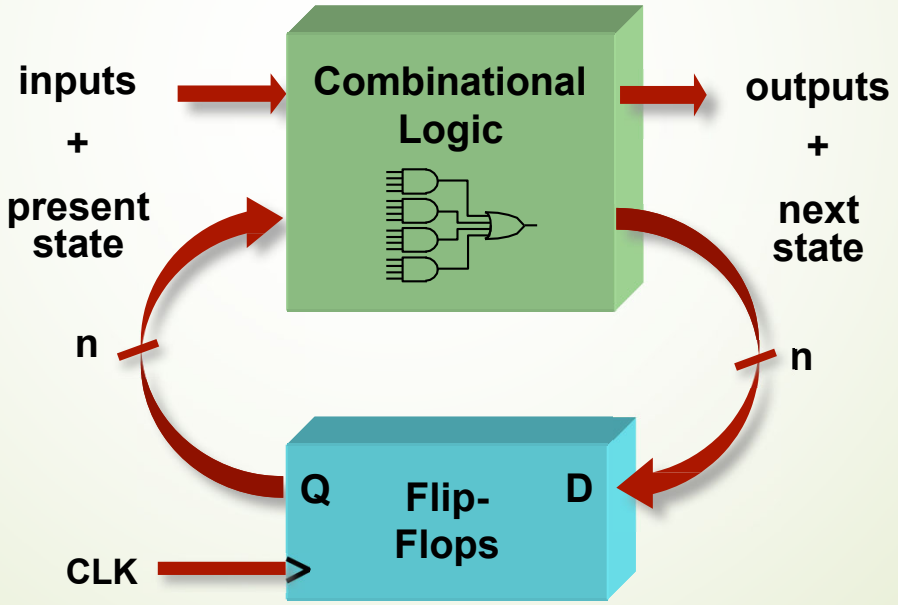
- State diagrams are representations of Finite State Machines (FSM)
- Mealy FSM
  - Output depends on input and state
  - Output is not synchronized with clock
  - can have temporarily unstable output
- Moore FSM
  - Output depends only on state





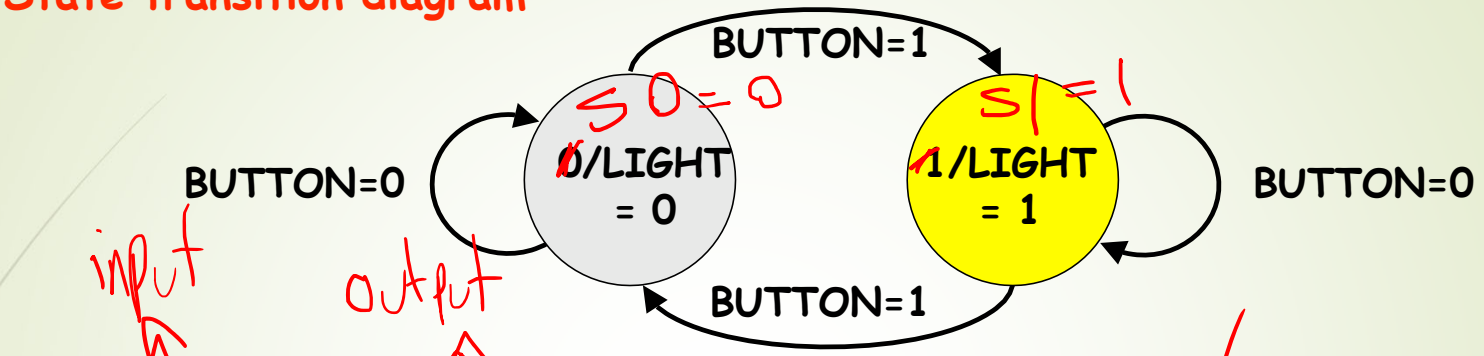
Finite State Machines

- **Finite State Machines (FSMs) are a useful abstraction for sequential circuits with centralized "states" of operation**
- **At each clock edge, combinational logic computes *outputs* and *next state* as a function of *inputs* and *present state***

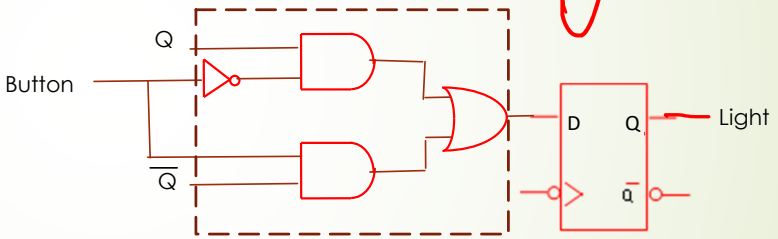


Example: Light Switch

- **State transition diagram**

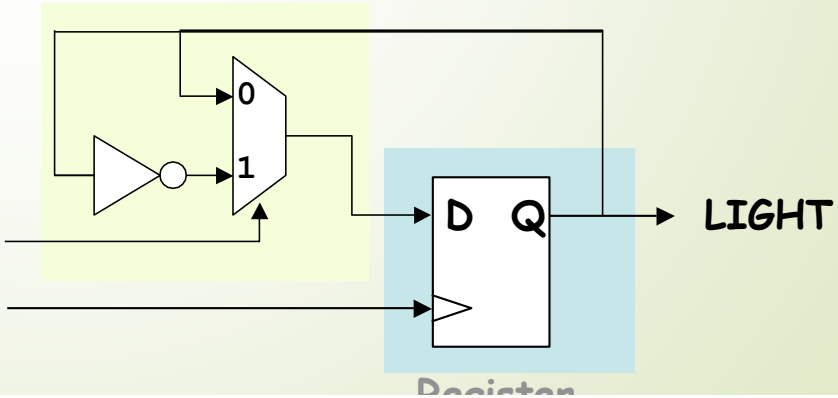


PS Q	Button	NS Q & D	Light
0	0	0	0
0	1	1	0
1	0	1	1
1	1	0	1



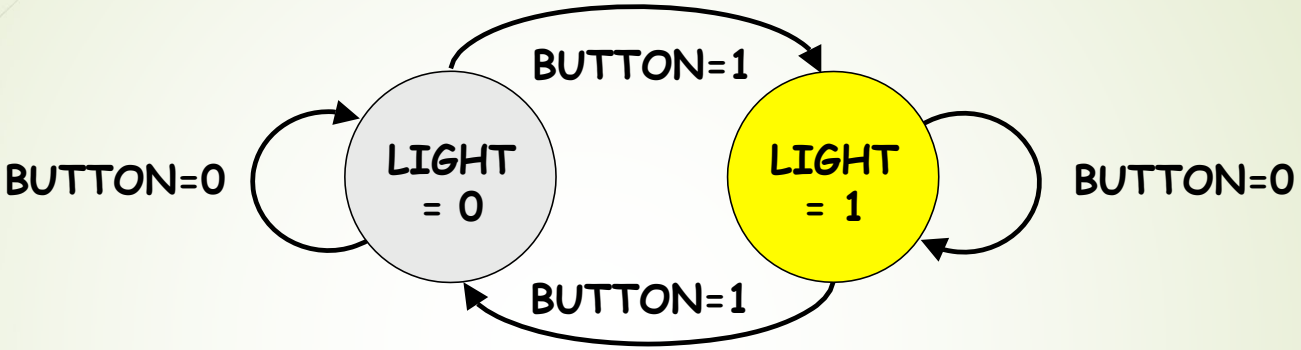
D = Q'B + QB'
Light = Q

Note: B = Button

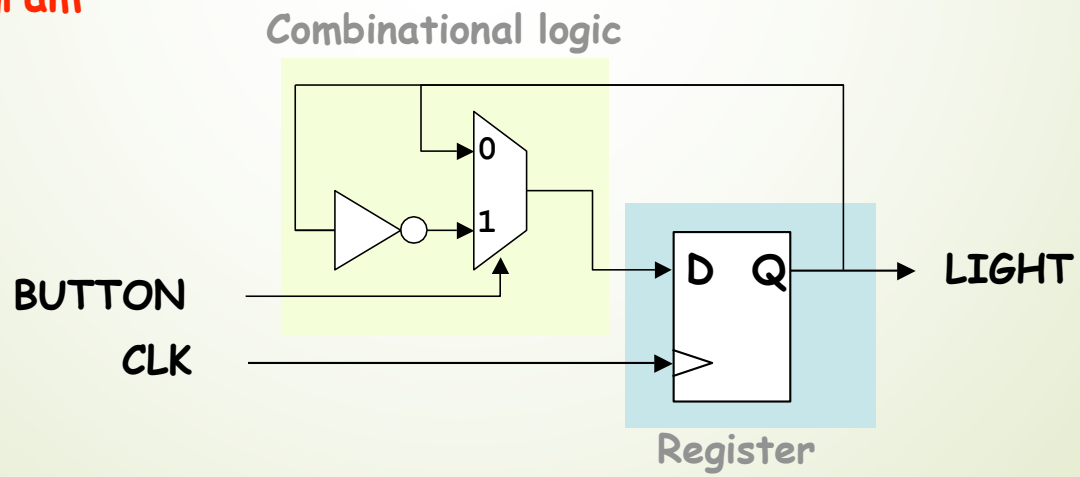


Example: Light Switch

- State transition diagram

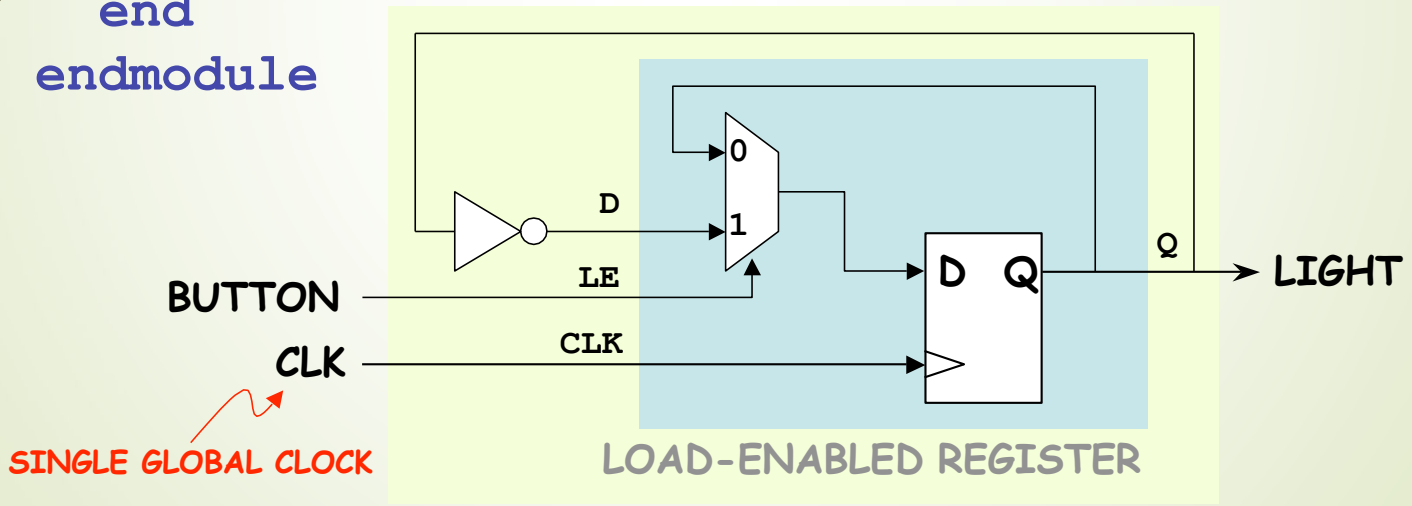


- Logic diagram



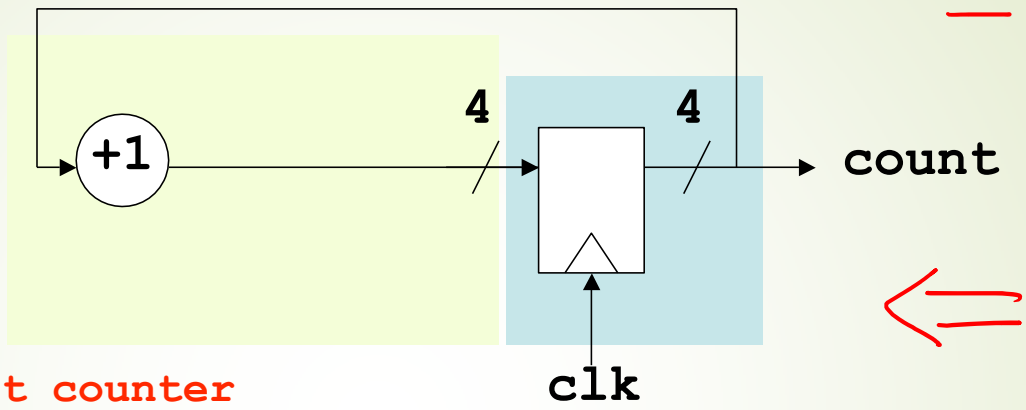
Clocked circuit for on/off button

```
module onoff(clk,button,light);
   input clk,button;
   output light;  reg light;
   always @ (posedge clk)  begin
    if (button) light <= ~light;
   end
endmodule
```



Example: 4-bit Counter

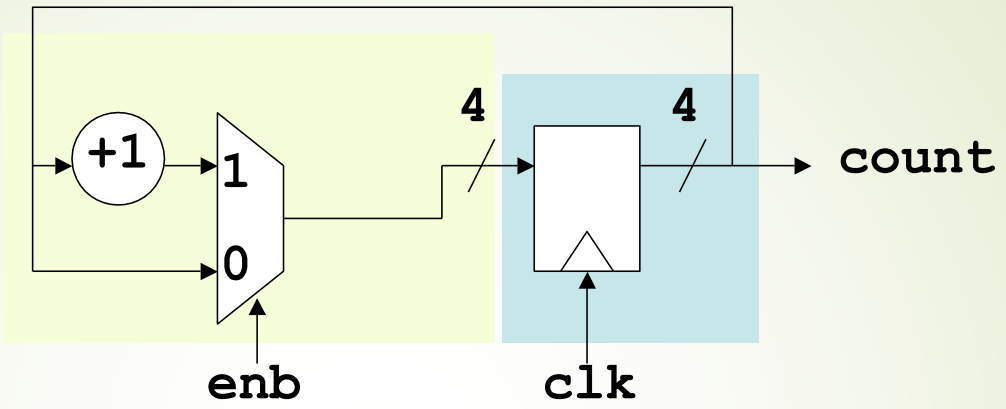
- Logic diagram



- Verilog

```
# 4-bit counter
module counter(clk, count);
  input clk;
  output [3:0] count;
  reg [3:0] count;

  always @ (posedge clk) begin
    count <= count+1;
  End
endmodule
```

- Logic diagram

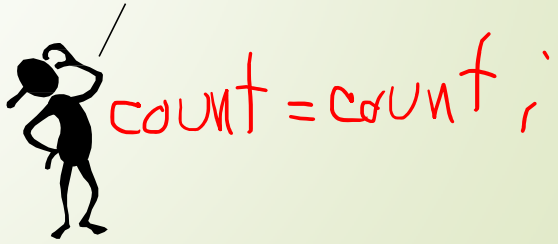


- Verilog

```
# 4-bit counter with enable
module counter(clk,enb,count);
  input clk,enb;
  output [3:0] count;
  reg [3:0] count;

  always @ (posedge clk) begin
    count <= enb ? count+1 : count;
  end
endmodule
```

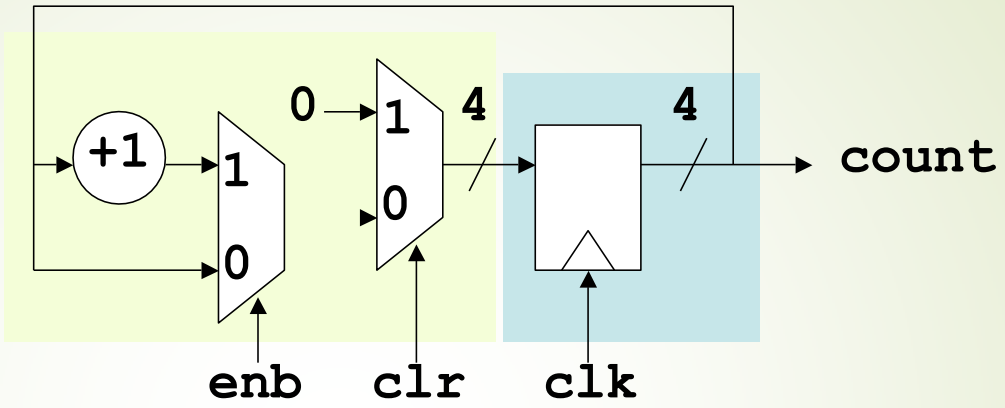
Could I use the following instead?

```
if (enb) count <= count+1;
```



Example: 4-bit Counter

- Logic diagram

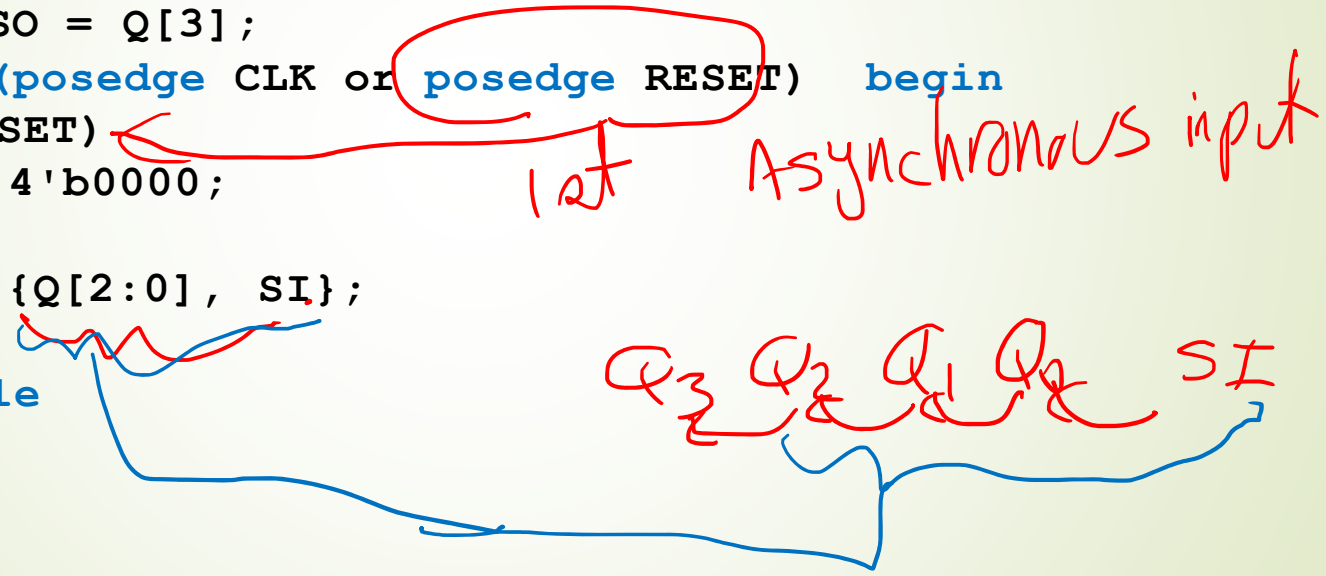


- Verilog

```
# 4-bit counter with enable and synchronous clear
module counter(clk,enb,clr,count);
  input clk,enb,clr;
  output [3:0] count;
  reg [3:0] count;

  always @ (posedge clk) begin
    count <= clr ? 4'b0 : (enb ? count+1 : count);
  end
endmodule
```

4-bit Shift Register with Reset

```
module srg_4_r_v (CLK, RESET, SI, Q,SO);
input CLK, RESET, SI;
output [3:0] Q;
output SO;
reg [3:0] Q;
assign SO = Q[3];
always@(posedge CLK or posedge RESET)  begin
  if (RESET)
    Q <= 4'b0000;
  else
    Q <= {Q[2:0], SI};
   end
endmodule
```

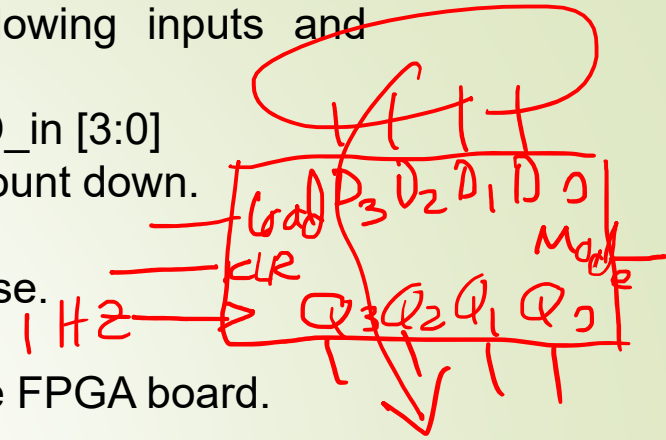


4-bit Binary Counter with Reset

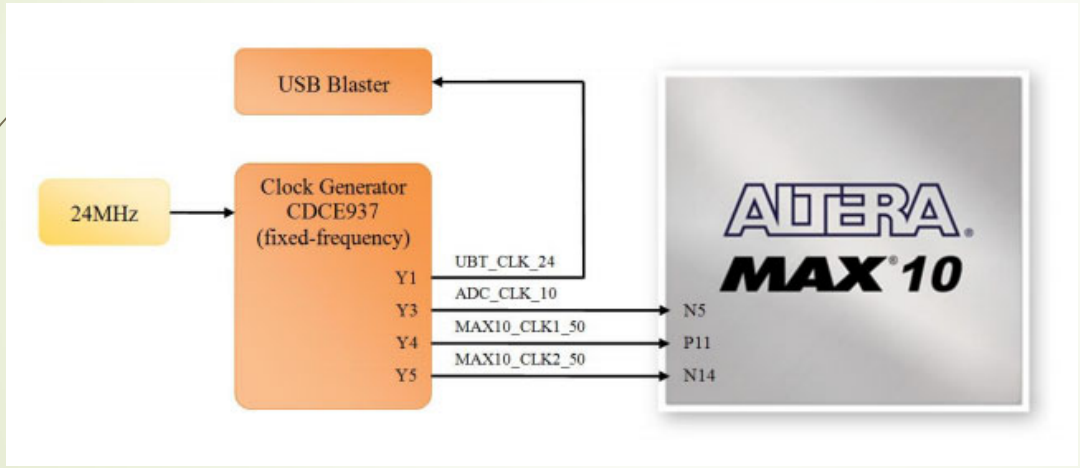
```
module count_4_r_v (CLK, RESET, EN, Q, CO);
input CLK, RESET, EN;
output [3:0] Q;
output CO;
reg [3:0] Q;
assign CO = (count == 4'b1111 && EN == 1'b1) ? 1 : 0;
always@(posedge CLK or posedge RESET)
  begin
  if (RESET)
   Q <= 4'b0000;
  else if (EN)
   Q <= Q + 4'b0001;
  end
endmodule
```

You are to design a 4-bit counter with the following inputs and functionality:

- Load (ld): if activated, count will be loaded from D_in [3:0]
- Mode: if 0, counter counts up. Otherwise, it will count down.
- Clear (clr): If activated, count will be 0
- Clock 1 Hz is generated from the previous exercise.



Complete the following code and implement it on the FPGA board.



Ld = 1

$Q_3 \cdots Q_0 \leftarrow D_3 \cdots D_0$

Signal Name	FPGA Pin No.	Description
~~ADC_CLk_50~~	~~PIN_N5~~	~~10 MHz clock input for ADC (Bank 3B)~~
MAX10_CLK1_50	PIN_P11	50 MHz clock input (Bank 3B)
MAX10_CLK2_50	PIN_N14	50 MHz clock input (Bank 3B)

Device family

Family: MAX 10 (DA/DF/DC/SA/SC)

Device: All

Target device

- Auto device selected by the Fitter
- Specific device selected in 'Available devices' list
- Other: n/a

Show in 'Available devices' list

Package: Any

Pin count: Any

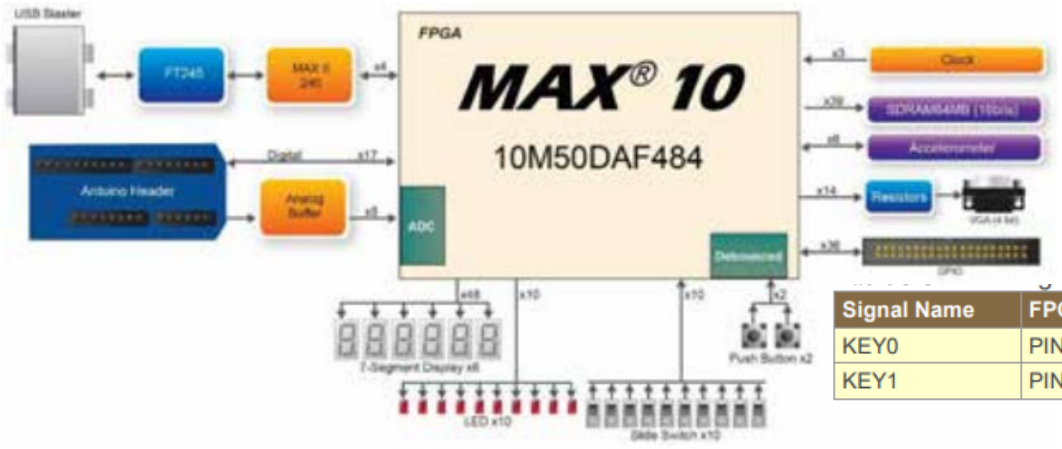
Core speed grade: Any

Name filter:

Show advanced devices

Available devices:

Name	Core Voltage	LEs	Total I/Os	GPIOs	Memory Bits	Embedded multiplier
10M50DAF484C7G	1.2V	49760	360	360	1677312	288



Signal Name	FPGA Pin No.	Description
KEY0	PIN_B8	Push-button[0]
KEY1	PIN_A7	Push-button[1]

Board	Device Name
DE10-Lite	MAX 10: 10M50DAF484C7G

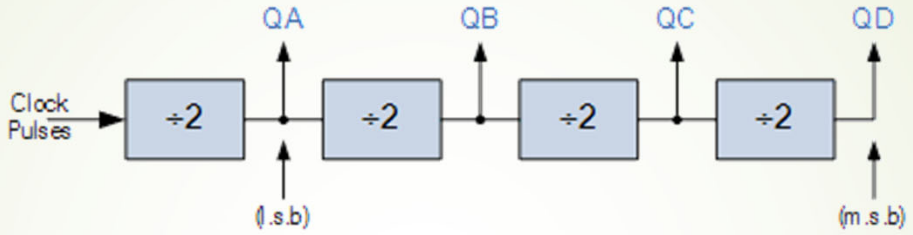
Pin Assignment

Signal Name	FPGA Pin No.	Description
SW0	PIN_C10	Slide Switch[0]
SW1	PIN_C11	Slide Switch[1]
SW2	PIN_D12	Slide Switch[2]
SW3	PIN_C12	Slide Switch[3]
SW4	PIN_A12	Slide Switch[4]
SW5	PIN_B12	Slide Switch[5]
SW6	PIN_A13	Slide Switch[6]
SW7	PIN_A14	Slide Switch[7]
SW8	PIN_B14	Slide Switch[8]
SW9	PIN_F15	Slide Switch[9]

Signal Name	FPGA Pin No.	Description
LEDR0	PIN_A8	LED [0]
LEDR1	PIN_A9	LED [1]
LEDR2	PIN_A10	LED [2]
LEDR3	PIN_B10	LED [3]
LEDR4	PIN_D13	LED [4]
LEDR5	PIN_C13	LED [5]
LEDR6	PIN_E14	LED [6]
LEDR7	PIN_D14	LED [7]
LEDR8	PIN_A11	LED [8]
LEDR9	PIN_B11	LED [9]

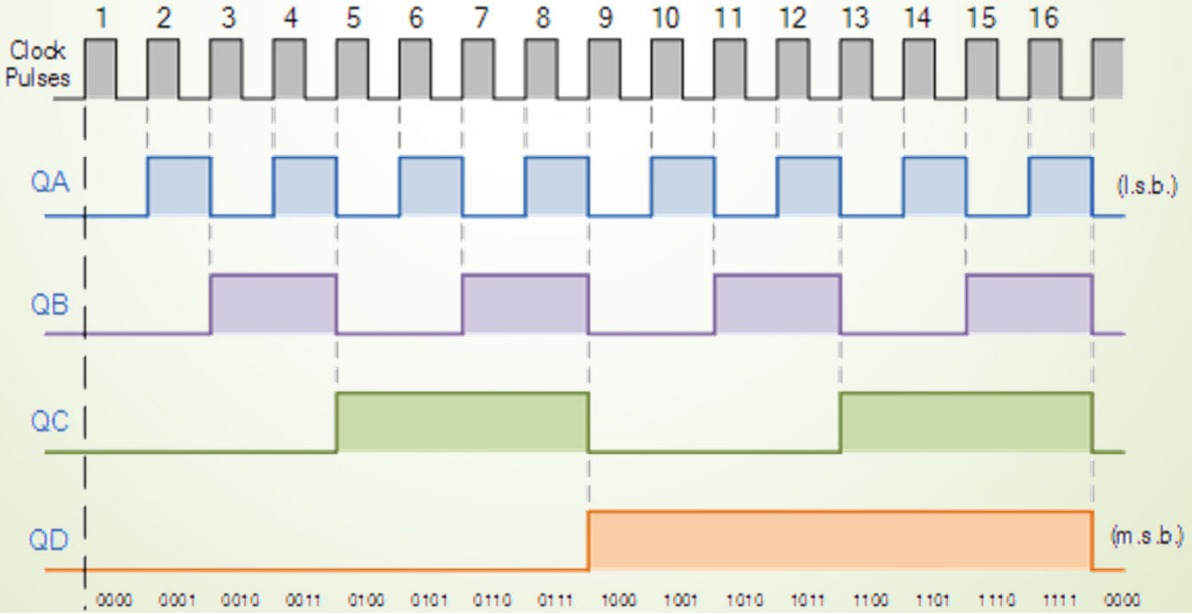
Exercise 1: Divide-by-N Circuit:

Design, simulate, and build a Divide-by-N circuit that will divide the on board clock from 50 MHz down to ~1 Hz. The basic principle is as follows:

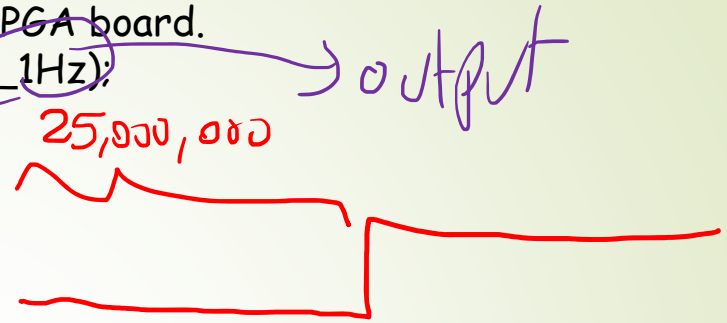


25×10^6

25×10^6



```
//The goal of this always procedural block is to generate 1Hz clock from a
//50MHz clock that is used in the Altera FPGA board.
module Divide_by_50M_counter(clr,clk,clk_1Hz);
input clr,clk;
output clk_1Hz;
reg clk_1Hz =1'b0;
integer counter_50M =0;
always @(posedge clk, posedge clr)
begin
  if (clr)
     counter_50M <=0;
  else if (counter_50M <25000000)
    begin
       counter_50M <= counter_50M + 1;
    end
  else if (counter_50M ==25000000)
     begin
       clk_1Hz <= !clk_1Hz;
       counter_50M <=0;
    end
end
endmodule
```



```
module up_down_counter(mode,clr,ld,D_in,clk,count,clk_1Hz);
input mode,clr,ld,clk;
input [3:0] D_in;
output clk_1Hz;
output [3:0] count;
reg [3:0] count;
reg clk_1Hz =1'b0;
integer counter_50M =0;
```

From Prev. Slide

```
//If "ld =1" we load the external data through D_in[3:0], if mode is active
// it will be counting up and if mode is inactive it will count down.

always @(posedge clk_1Hz, posedge clr)
begin
.
. if (clr)
     count <=0;
  else if (ld)
     count <= D_in[3:0];
.
endmodule
```

{0 up, 1 down} → Q=0 → 4 inputs → output

load

if(mode) count <= count - 1;