PWM Main Characteristics

1. Duty cycle
2. Resolution
3. Switching frequency

The minimum incremental step is given by the Number of bits R.

Resolution=1/2^R

System clock period =Ts

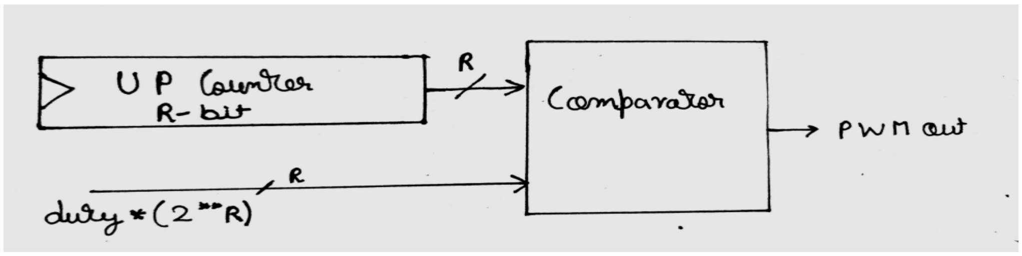
PWM period = (2^R)*Ts

Example-

If R= 8 bits.

The duty cycle can vary from (0,1/256,2/256……..255 /256)

Simple PWM

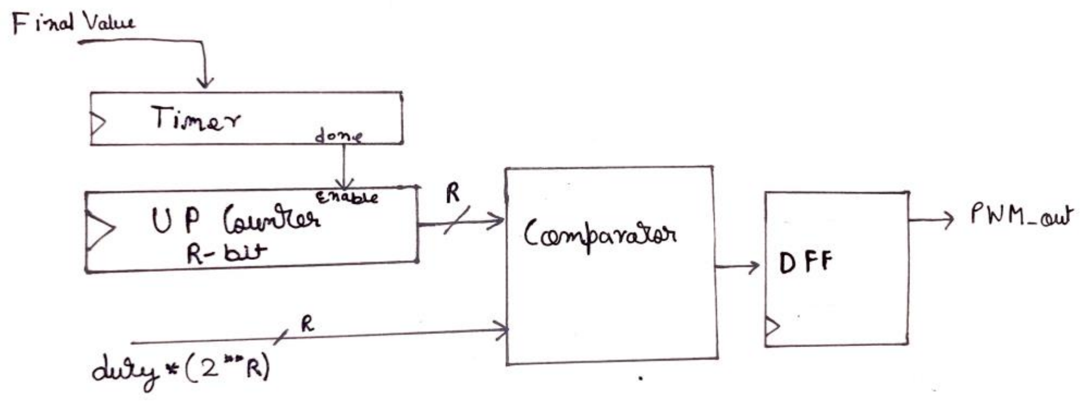


System clock period =Ts

PWM period = (2^R)*Ts

So, to change the frequency here we must change the Ts or R.

Improved PWM



System clock period =Ts

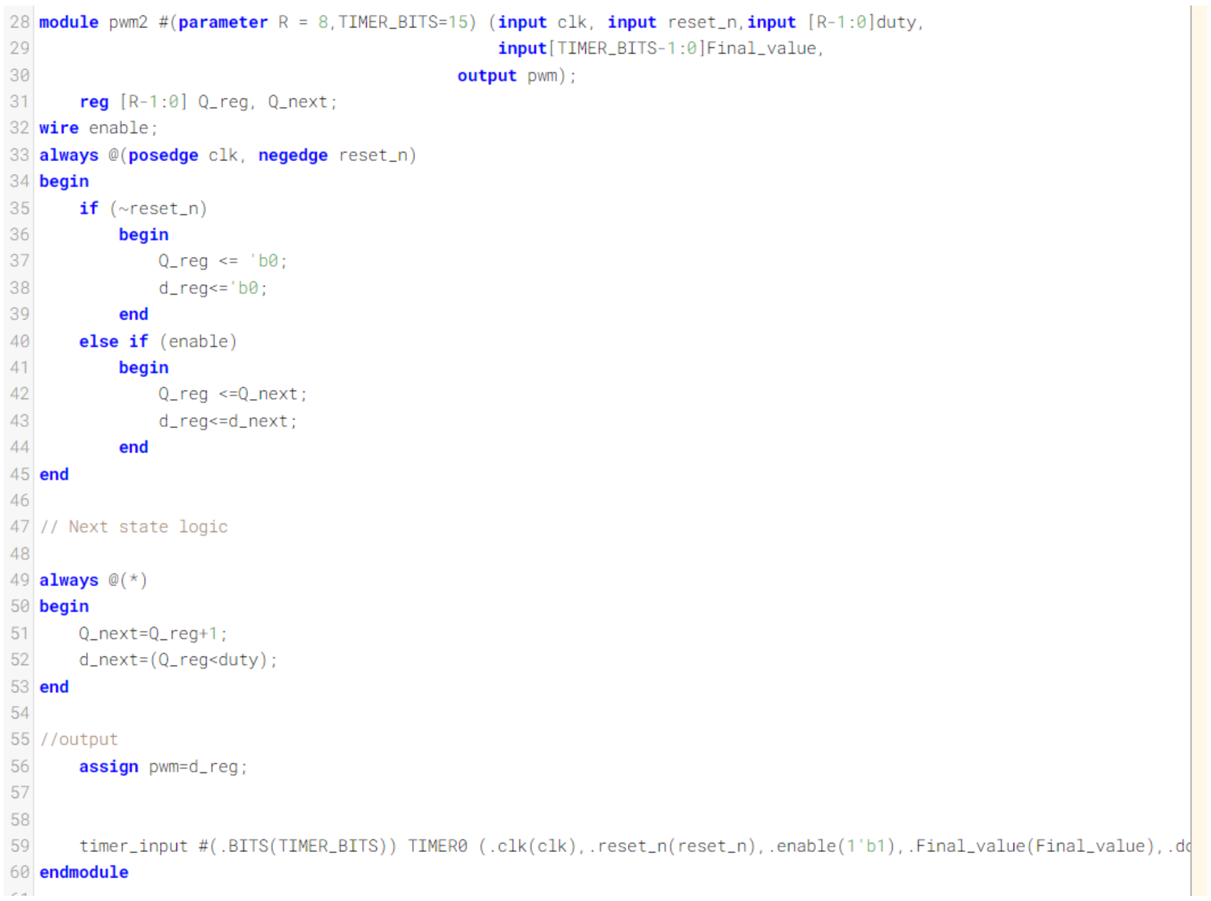
- PWM period = (2^R)*Ts*(Finalvalue+1)
  So, to change the frequency we adjust the final value keeping the resolution and system clock the same.

- Adding a D flip-flop will synchronize the output.
- http://hdlbits.01xz.net/wiki/Iverilog?load=GrHn4C
-

TIMER

```
module timer_input #(parameter BITS = 4) (input clk, input reset_n,input enable,input [BITS-1:0] FINAL_VALUE,
                                          output done);
    reg [BITS-1:0] Q_reg, Q_next;

always @(posedge clk, negedge reset_n)
begin
    if (~reset_n)
        Q_reg <= 'b0;
    else if (enable)
        Q_reg <=Q_next;
    else
        Q_reg <= Q_reg;

end

// Next state logic

always @(*)
begin
    Q_next = Q_reg + 1;
    if(FINAL_VALUE==Q_reg)
        done=1;
    else
        done=0;
end
endmodule

```

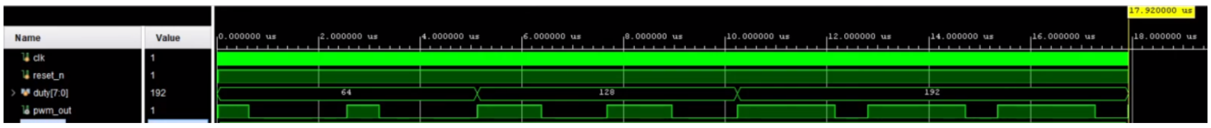
COUNTER /COMPARATOR



```verilog
module pwm2 #(parameter R = 8,TIMER_BITS=15) (input clk, input reset_n,input [R-1:0]duty,
                                              input[TIMER_BITS-1:0]Final_value,
                                          output pwm);
    reg [R-1:0] Q_reg, Q_next;
wire enable;
always @(posedge clk, negedge reset_n)
begin
    if (~reset_n)
        begin
            Q_reg <= 'b0;
            d_reg<='b0;
        end
    else if (enable)
        begin
            Q_reg <=Q_next;
            d_reg<=d_next;
        end
end

// Next state logic

always @(*)
begin
    Q_next=Q_reg+1;
    d_next=(Q_reg<duty);
end

//output
    assign pwm=d_reg;


    timer_input #(.BITS(TIMER_BITS)) TIMER0 (.clk(clk),.reset_n(reset_n),.enable(1'b1),.Final_value(Final_value),.d
endmodule
```

Output:

- Duty cycle –
  25%= 64/256
  50%= 128/126
  75%=192/256
- R=8, Finalvalue=2;



17.920000 us
Name
Value
clk
reset_n
duty[7:0]
pwm_out
1
1
192
1
0.000000 us
2.000000 us
4.000000 us
6.000000 us
8.000000 us
10.000000 us
12.000000 us
14.000000 us
16.000000 us
18.000000 us
64
128
192