Exp. No.: 03 Date:	DIGITAL DELAY TIMER (LS7212) IN VERILOG HDL

OBJECTIVES:

(i) To construct the Verilog code using HDL

(ii) To design a Verilog code for delay timer using HDL.

AIM:

(i) To design and construct the Verilog code for delay timer.

SOFTWARE REQUIRED:

Xilinx ISE

THEORY:

CODE DESCRIPTION:

The digital delay timer being implemented is CMOS IC LS7212 which is to generate programmable delays. The specification of the delay timer can be easily found here. Basically, the delay timer has 4 operating modes: one-shot (OS), Delayed Operate (DO), Delayed Release (DR), Dual Delay (DD). Those four modes will be selected by inputs mode_a and mode_b. The wb [7:0] input is to program the delays according to given equations in the specification of the delay timer.

FUCNTION TABLE:

Table:1 Function table

mode_a	mode_b	Mode
0	0	One-shot
0	1	Delayed operate
1	0	Delayed release
1	1	Dual delay

BLOCK DIAGRAM:

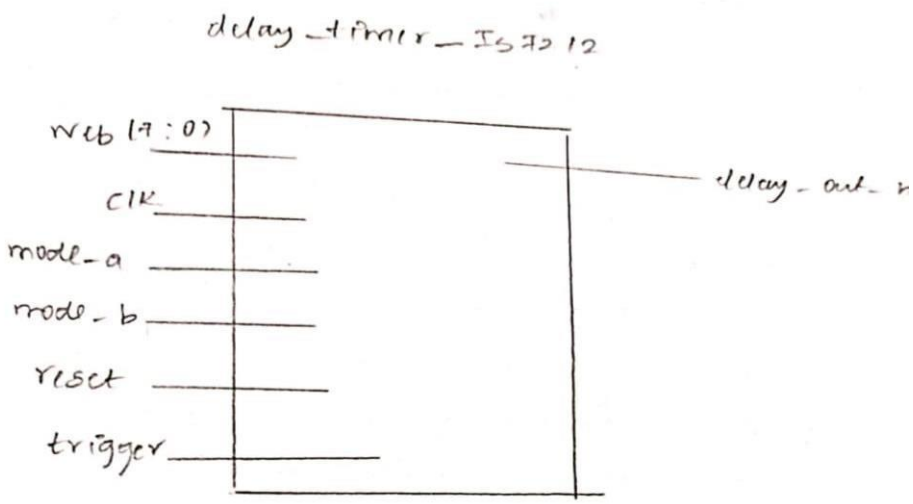

Figure:3.1 Block diagram of LS7212

VERILOG CODE FOR DELAY TIMER:

```
module delay_timer_ls7212
(
  input [7:0] wb,
input clk,       input reset,
input trigger,       input
mode_a, mode_b,
output reg delay_out_n
);
 reg[7:0] PULSE_WIDTH ;
reg [7:0] DELAY;
 reg [7:0] TIMER=0;
 reg trigger_sync_1=0,trigger_sync_2=0;
wire trigger_rising,trigger_falling;    reg
timer_start=0,out_low=0;    wire
timer_clear2,timer_clear3,timer_clear;    reg
[1:0] mode;
```

```
 reg reset_timer1=0,reset_timer2=0,reset_timer=0;
wire      reset_timer3,reset_det;                    reg
reset_det1=0,reset_det2=0;
always @(posedge clk)    begin
          trigger_sync_1 <= trigger;
trigger_sync_2 <= trigger_sync_1;
reset_timer1 <= reset_timer;
reset_timer2 <= reset_timer1;
reset_det1 <= reset;
   reset_det2 <= reset_det1;
 end
assign trigger_rising = trigger_sync_1 & (~trigger_sync_2);
assign trigger_falling = trigger_sync_2 & (~trigger_sync_1);
assign reset_timer3 = reset_timer1 & (~reset_timer2);
assign reset_det = reset_det2 & (~reset_det1);
always @(trigger_rising,trigger_falling,mode_a,mode_b,wb)    begin
     if(trigger_falling == 1 || trigger_rising == 1) begin
          PULSE_WIDTH = wb;
DELAY = (2*wb + 1)/2;              mode
= {mode_a,mode_b};
     end
end
always
@(mode,reset,trigger_falling,trigger_rising,TIMER,reset,trigger,PULSE_WIDTH,DELAY,re
set_det)    begin         case(mode)
              2'b00: // One-Shot Mode
begin
                          if(reset) begin
out_low <= 0;
timer_start <= 0;
reset_timer <= 1;                              end
```

```
                    else if(trigger_rising==1) begin
out_low <= 1;                               timer_start
<= 1;                                   reset_timer <= 1;
end
                    else if(TIMER>=PULSE_WIDTH) begin
                        out_low   <=   0;
timer_start               <=              0;
reset_timer <= 1;                       end
end
                begin
                    if(reset) begin
out_low <= 0;
timer_start <= 0;
reset_timer <= 1;
        end
else if(reset_det==1 && trigger==1) begin
 timer_start<=1;
reset_timer <= 0;
          end
  else if(trigger_rising==1) begin
timer_start <= 1;
reset_timer <= 0;                               end
 else     if(trigger_falling==1     ||     trigger     ==     0)     begin
out_low <= 0;
      reset_timer <= 1;
timer_start <= 0;                           end
    else if(TIMER >= DELAY) begin
out_low <= 1;
    timer_start <= 0;
     reset_timer <= 1;                          end
```

```
        reset_timer <= 0;
end
                    begin
if(reset) begin
out_low <= 0;
timer_start <= 0;
reset_timer <= 1;
end
 else if(trigger_rising==1 || trigger == 1) begin
out_low <= 1;                          end
  else if(trigger_falling==1 ) begin
timer_start <= 1;
reset_timer <= 0;
              end
else if(TIMER>=DELAY) begin
out_low <= 0;
      timer_start <= 0;
    reset_timer <= 1;                          end
end
begin                          if(reset)
begin                              out_low
<= 0;
timer_start <= 0;
reset_timer <= 1;
end
else if(reset_det==1 && trigger==1) begin
timer_start <= 1;
reset_timer <= 0;
 end
```

```
else      if(trigger_falling==1      ||      trigger_rising==1      )      begin
timer_start <= 1;
  reset_timer <= 0;
  end
 else
 if(TIMER>=DELAY)
 begin
       out_low                  <=                  trigger;
timer_start <= 0;                              reset_timer
<= 1;                          end
     end
endcase    end
 always @(posedge clk or posedge timer_clear)
begin    if(timer_clear)        TIMER <= 0;    else
if(timer_start)
     TIMER <= TIMER + 1;
end
 assign timer_clear = reset_timer3 | trigger_rising == 1 | timer_clear3 ;
assign timer_clear2 = (trigger_rising == 1)|(trigger_falling == 1);    assign
timer_clear3 = timer_clear2 & (mode == 2'b11);
if(out_low       ==       1)
delay_out_n      <=      0;
else
delay_out_n <= 1;    end
endmodule
```

TIME BENCH:

```
module tb_ls7212;
reg    [7:0]    wb;
reg clk;
```

```
    reg reset;        reg
trigger;        reg
mode_a;         reg
mode_b;
delay_out_n;
delay_timer_ls7212 uut (
        .wb(wb),
        .clk(clk),
        .reset(reset),
        .trigger(trigger),
        .mode_a(mode_a),
        .mode_b(mode_b),
        .delay_out_n(delay_out_n)
    ); initial begin
wb = 10;
mode_a = 0;
mode_b = 0;
reset = 0;
trigger = 0;
#500;             trigger =
1;            #15000;
trigger = 0;
#15000;      trigger = 1;
        #2000 ;
trigger = 0;
#2000;
trigger = 1;
#2000;
trigger = 0;
#20000;
```

```
trigger = 1;
#30000;
trigger = 0;
#2000;
trigger = 1;
#2000;
trigger = 0;
#4000;
trigger = 1;
#10000;              reset
= 1;              #10000;
reset = 0;
          end      initial begin
clk = 0;      forever #500
clk = ~clk;         end
 endmodule
```

TECHNICAL SPECIFICATION:

In this Verilog code, we have defined a module called Digital Delay Timer. It takes several inputs: clk(clock signal), reset (reset signal), enable (enable signal), and delay value (the desired delay value in number of clock cycles). It provides one output, timer done, which indicates when the delay has completed.

The module includes two always blocks. The first one is responsible for incrementing the count register when the timer is enabled and running. The second always block controls the state of the timer running register based on the enable signal and the count reaching the desired delay value.

OUTPUT ANALYSIS:

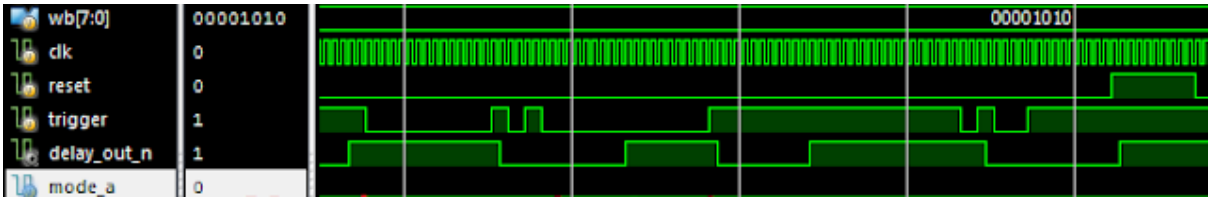

Figure: 3.2 Output of digital delay timer

PRE LABQUESTIONS:

1.What is Verilog HDL?

The Verilog Hardware Description Language (Verilog HDL) is a language that describes the behaviour of electronic circuits, most commonly digital circuits.

2.What is main use age of Verilog HDL software?

Verilog, standardized as IEEE 1364, is a hardware description language (HDL) used to model electronic systems. It is most commonly used in the design and verification of digital circuits at the register-transfer level of abstraction. Verilog HDL for designing hardware and for creating test entities to verify the behaviour of a piece of hardware. Verilog HDL is used as an entry format by a variety of EDA tools, including synthesis tools.

POST LAB QUESTIONS:

1.What is use of delay timer in Verilog HDL?

The digital delay timer being implemented is CMOS IC LS7212 which is to generate programmable delays. The specification of the delay timer can be easily found here. Basically, the delay timer has 4 operating modes: one-shot (OS), Delayed Operate (DO), Delayed Release (DR), Dual Delay (DD).

2.Advantage and disadvantage of Verilog HDL?

Advantages

- Verilog HDL allows different levels of abstraction to be mixed in the same model.
- Thus, a designer can define a hardware model in terms of switches, gates, RTL, or behavioral code.
- A designer needs to learn only one language for stimulus and hierarchical design. of choice for designers.

Disadvantages

- The control logic is designed with traditional techniques.
- The intent of design gets lost in complexity and details.
- The schematics are accompanied with documentation.
- Portability is an issue.
- Simulation environment for schematic capture design is not same.

RESULT:

Thus the digital delay timer (LS7212) in Verilog HDL has been stimulated and the output has been verified successfully.

MARK ALLOCATION:

S. No	Parameters	Mark Allotted	Mark Awarded
1	Circuit Design/ code developing and debugging / Trouble shooting	0-3	
2	Implementation and Demonstration	0-3	
3	Discussion	0-3	
4	Report writing & Presentation	0-3	
5	Contribution & Team Dynamics	0-3	
Total		15	

Signature of Lab In-charge