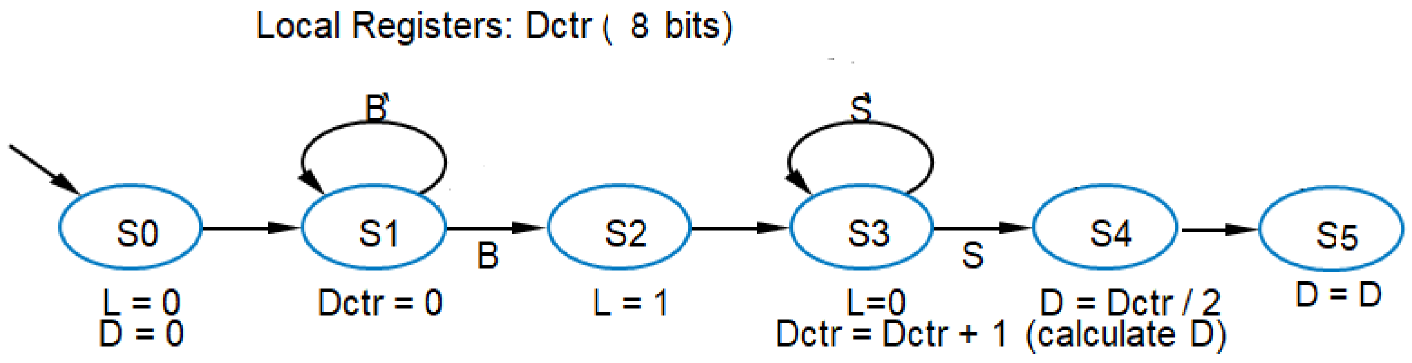
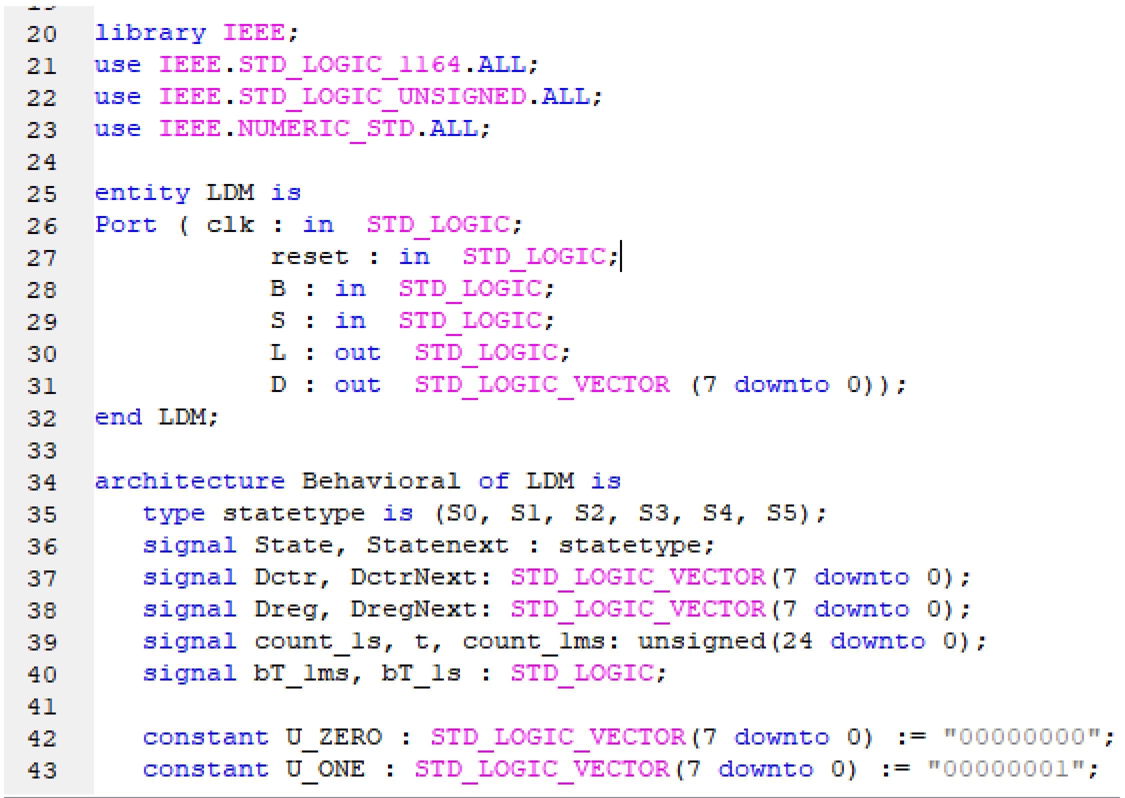
Laser Distance Measurer

Modified State Diagram for Elbert v2 board

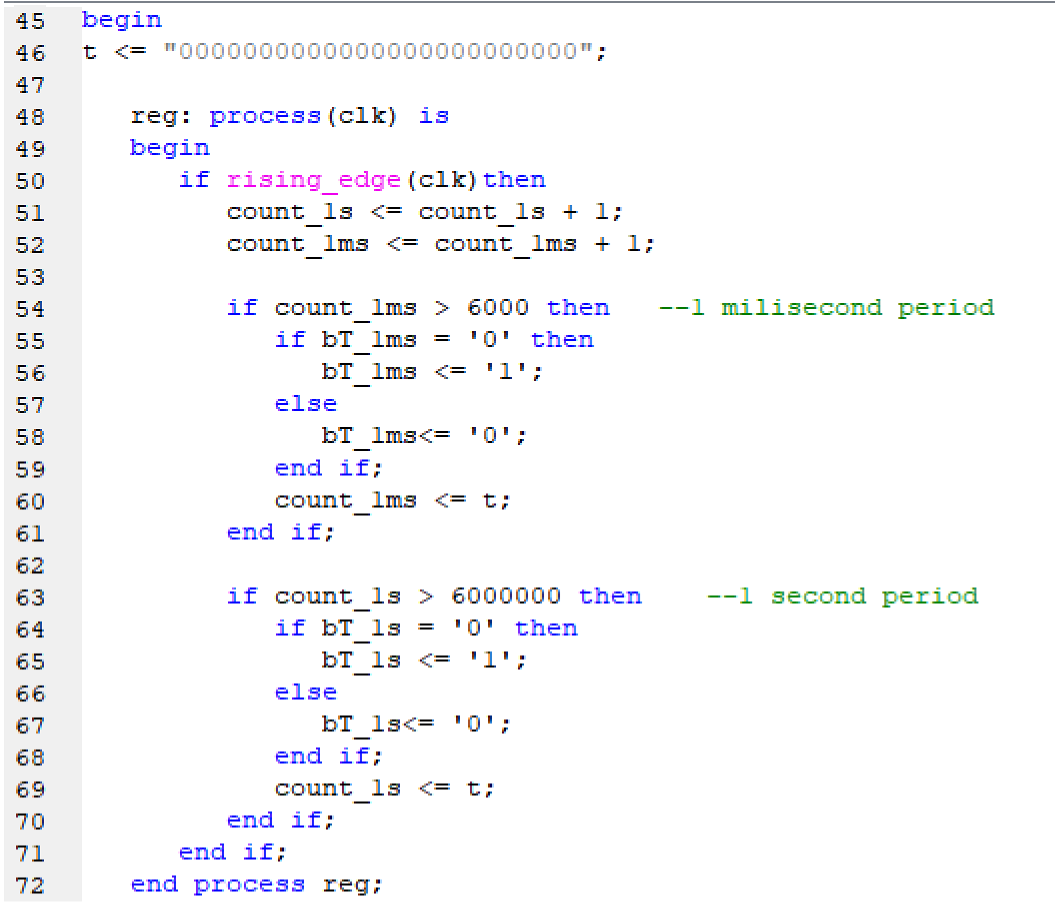


The clock cycle is slowed down to 1 second of a clock freqency =1 Hz.

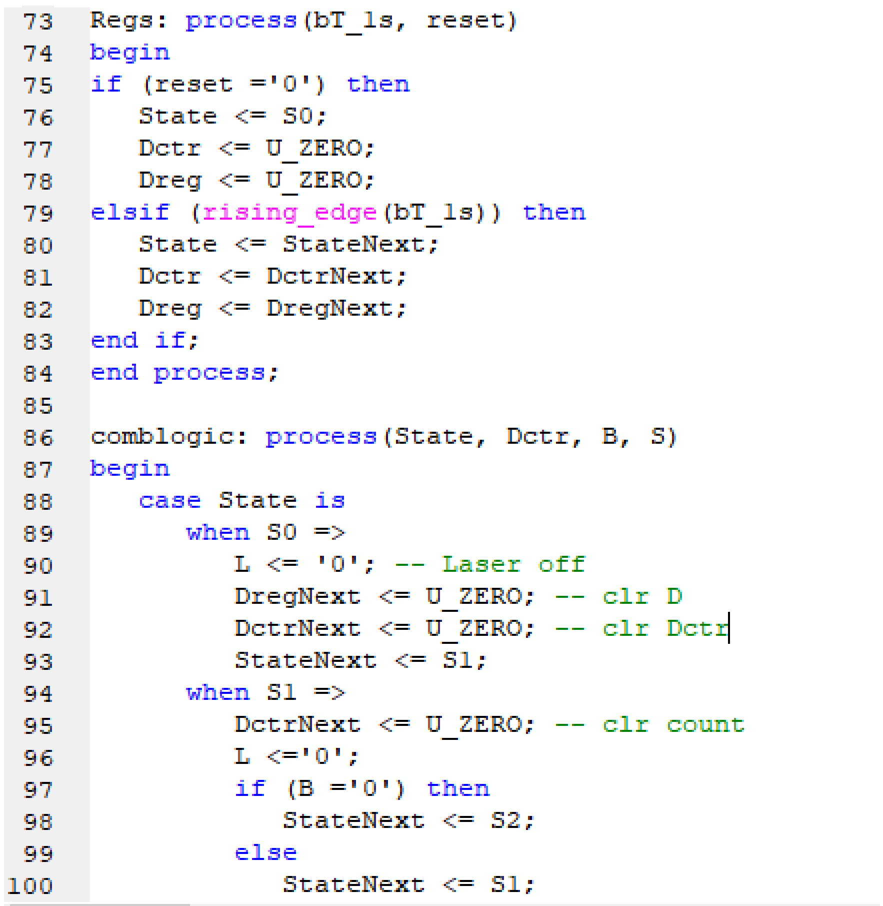
Behavioral VHDL Model



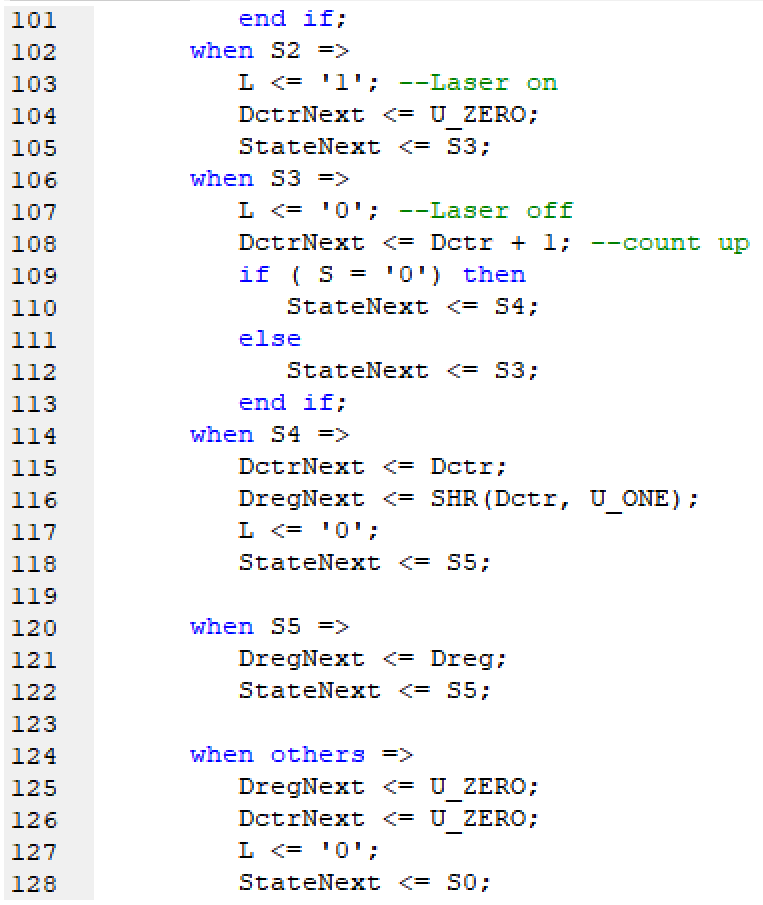
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;

entity LDM is
Port ( clk : in  STD_LOGIC;
           reset : in  STD_LOGIC;
           B : in  STD_LOGIC;
           S : in  STD_LOGIC;
           L : out  STD_LOGIC;
           D : out  STD_LOGIC_VECTOR (7 downto 0));
end LDM;

architecture Behavioral of LDM is
   type statetype is (S0, S1, S2, S3, S4, S5);
   signal State, Statenext : statetype;
   signal Dctr, DctrNext: STD_LOGIC_VECTOR(7 downto 0);
   signal Dreg, DregNext: STD_LOGIC_VECTOR(7 downto 0);
   signal count_ls, t, count_lms: unsigned(24 downto 0);
   signal bT_lms, bT_ls : STD_LOGIC;

   constant U_ZERO : STD_LOGIC_VECTOR(7 downto 0)  := "00000000";
   constant U_ONE : STD_LOGIC_VECTOR(7 downto 0)  := "00000001";
```



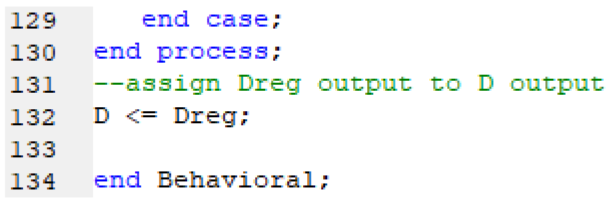
```
begin
t <= "0000000000000000000000000";

   reg: process(clk) is
   begin
      if rising_edge(clk)then
         count_ls <= count_ls + 1;
         count_lms <= count_lms + 1;

         if count_lms > 6000 then   --1 milisecond period
            if bT_lms = '0' then
               bT_lms <= '1';
            else
               bT_lms<= '0';
            end if;
            count_lms <= t;
         end if;

         if count_ls > 6000000 then    --1 second period
            if bT_ls = '0' then
               bT_ls <= '1';
            else
               bT_ls<= '0';
            end if;
            count_ls <= t;
         end if;
      end if;
   end process reg;
```



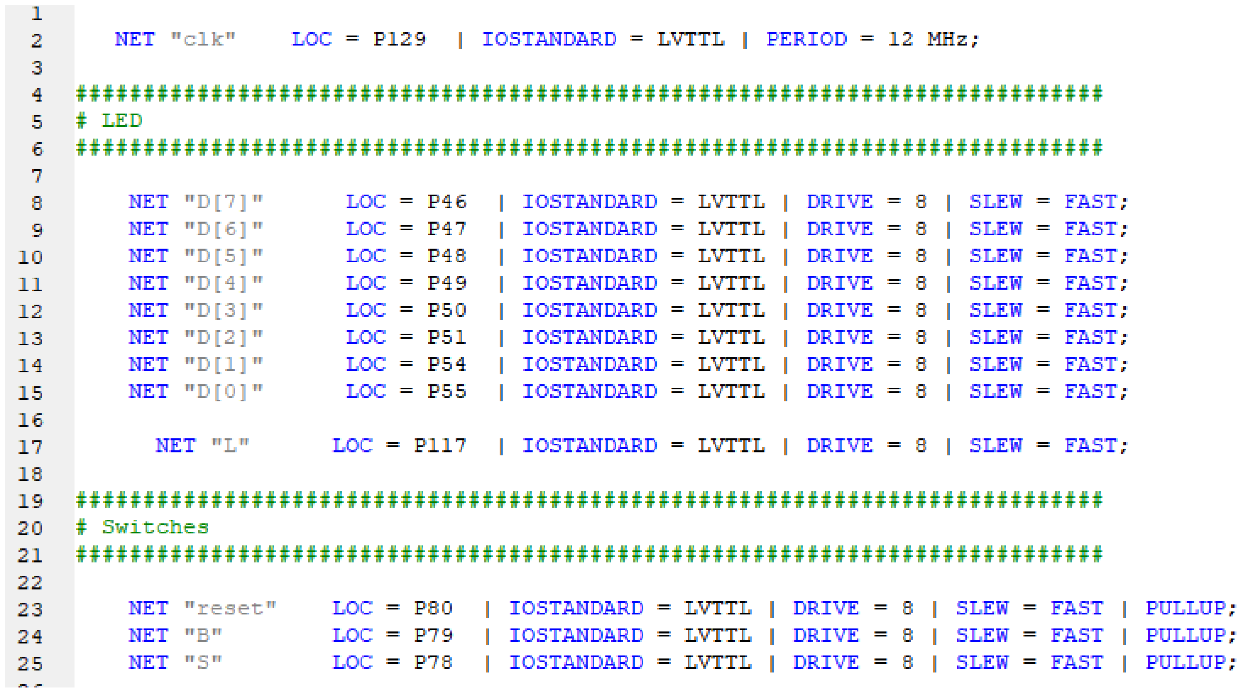
```vhdl
Regs: process(bT_ls, reset)
begin
if (reset ='0') then
   State <= S0;
   Dctr <= U_ZERO;
   Dreg <= U_ZERO;
elsif (rising_edge(bT_ls)) then
   State <= StateNext;
   Dctr <= DctrNext;
   Dreg <= DregNext;
end if;
end process;

comblogic: process(State, Dctr, B, S)
begin
   case State is
      when S0 =>
         L <= '0'; -- Laser off
         DregNext <= U_ZERO; -- clr D
         DctrNext <= U_ZERO; -- clr Dctr
         StateNext <= S1;
      when S1 =>
         DctrNext <= U_ZERO; -- clr count
         L <='0';
         if (B ='0') then
            StateNext <= S2;
         else
            StateNext <= S1;
```



```vhdl
         end if;
      when S2 =>
         L <= '1'; --Laser on
         DctrNext <= U_ZERO;
         StateNext <= S3;
      when S3 =>
         L <= '0'; --Laser off
         DctrNext <= Dctr + 1; --count up
         if ( S = '0') then
            StateNext <= S4;
         else
            StateNext <= S3;
         end if;
      when S4 =>
         DctrNext <= Dctr;
         DregNext <= SHR(Dctr, U_ONE);
         L <= '0';
         StateNext <= S5;

      when S5 =>
         DregNext <= Dreg;
         StateNext <= S5;

      when others =>
         DregNext <= U_ZERO;
         DctrNext <= U_ZERO;
         L <= '0';
         StateNext <= S0;
```

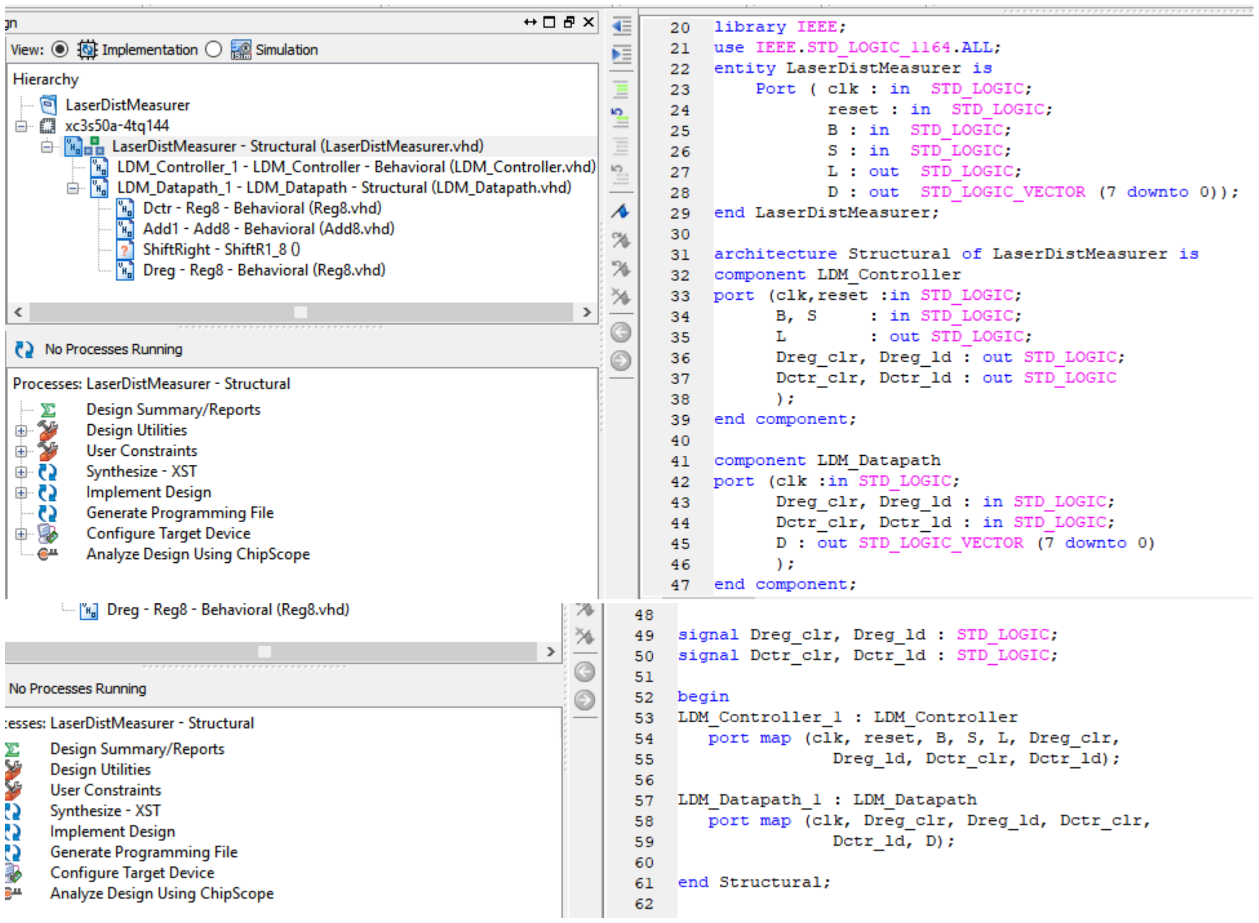


```vhdl
   end case;
end process;
--assign Dreg output to D output
D <= Dreg;

end Behavioral;
```



```
   NET "clk"    LOC = P129   | IOSTANDARD = LVTTL | PERIOD = 12 MHz;

##############################################################################
# LED
##############################################################################

    NET "D[7]"      LOC = P46   | IOSTANDARD = LVTTL | DRIVE = 8 | SLEW = FAST;
    NET "D[6]"      LOC = P47   | IOSTANDARD = LVTTL | DRIVE = 8 | SLEW = FAST;
    NET "D[5]"      LOC = P48   | IOSTANDARD = LVTTL | DRIVE = 8 | SLEW = FAST;
    NET "D[4]"      LOC = P49   | IOSTANDARD = LVTTL | DRIVE = 8 | SLEW = FAST;
    NET "D[3]"      LOC = P50   | IOSTANDARD = LVTTL | DRIVE = 8 | SLEW = FAST;
    NET "D[2]"      LOC = P51   | IOSTANDARD = LVTTL | DRIVE = 8 | SLEW = FAST;
    NET "D[1]"      LOC = P54   | IOSTANDARD = LVTTL | DRIVE = 8 | SLEW = FAST;
    NET "D[0]"      LOC = P55   | IOSTANDARD = LVTTL | DRIVE = 8 | SLEW = FAST;

      NET "L"      LOC = P117   | IOSTANDARD = LVTTL | DRIVE = 8 | SLEW = FAST;

##############################################################################
# Switches
##############################################################################

    NET "reset"    LOC = P80   | IOSTANDARD = LVTTL | DRIVE = 8 | SLEW = FAST | PULLUP;
    NET "B"        LOC = P79   | IOSTANDARD = LVTTL | DRIVE = 8 | SLEW = FAST | PULLUP;
    NET "S"        LOC = P78   | IOSTANDARD = LVTTL | DRIVE = 8 | SLEW = FAST | PULLUP;
```

Structural VHDL Model



View: ● Implementation ○ Simulation

Hierarchy
- LaserDistMeasurer
- xc3s50a-4tq144
  - LaserDistMeasurer - Structural (LaserDistMeasurer.vhd)
    - LDM_Controller_1 - LDM_Controller - Behavioral (LDM_Controller.vhd)
    - LDM_Datapath_1 - LDM_Datapath - Structural (LDM_Datapath.vhd)
      - Dctr - Reg8 - Behavioral (Reg8.vhd)
      - Add1 - Add8 - Behavioral (Add8.vhd)
      - ShiftRight - ShiftR1_8 ()
      - Dreg - Reg8 - Behavioral (Reg8.vhd)

No Processes Running

Processes: LaserDistMeasurer - Structural
- Design Summary/Reports
- Design Utilities
- User Constraints
- Synthesize - XST
- Implement Design
- Generate Programming File
- Configure Target Device
- Analyze Design Using ChipScope

Dreg - Reg8 - Behavioral (Reg8.vhd)

No Processes Running

esses: LaserDistMeasurer - Structural
- Design Summary/Reports
- Design Utilities
- User Constraints
- Synthesize - XST
- Implement Design
- Generate Programming File
- Configure Target Device
- Analyze Design Using ChipScope

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity LaserDistMeasurer is
    Port ( clk : in  STD_LOGIC;
           reset : in  STD_LOGIC;
           B : in  STD_LOGIC;
           S : in  STD_LOGIC;
           L : out  STD_LOGIC;
           D : out  STD_LOGIC_VECTOR (7 downto 0));
end LaserDistMeasurer;

architecture Structural of LaserDistMeasurer is
component LDM_Controller
port (clk,reset :in STD_LOGIC;
        B, S      : in STD_LOGIC;
        L         : out STD_LOGIC;
        Dreg_clr, Dreg_ld : out STD_LOGIC;
        Dctr_clr, Dctr_ld : out STD_LOGIC
        );
end component;

component LDM_Datapath
port (clk :in STD_LOGIC;
        Dreg_clr, Dreg_ld : in STD_LOGIC;
        Dctr_clr, Dctr_ld : in STD_LOGIC;
        D : out STD_LOGIC_VECTOR (7 downto 0)
        );
end component;

signal Dreg_clr, Dreg_ld : STD_LOGIC;
signal Dctr_clr, Dctr_ld : STD_LOGIC;

begin
LDM_Controller_1 : LDM_Controller
    port map (clk, reset, B, S, L, Dreg_clr,
                Dreg_ld, Dctr_clr, Dctr_ld);

LDM_Datapath_1 : LDM_Datapath
    port map (clk, Dreg_clr, Dreg_ld, Dctr_clr,
                Dctr_ld, D);

end Structural;
```