IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 9, September 2016

FPGA Based Digital Filter Module

Dr. Haresh Pandya[1], Mr. Mahesh Rangapariya[2], Mr. Kamaldeep Gosai[3], Mr. Jatin Savliya[4]

Professor & Head, Department of Electronics, Saurashtra University, Rajakot, Gujrat, India[1]

Ph.D. Student & Visiting Lecturer, Department of Electronics, Saurashtra University, Rajakot, Gujarat, India[2]

Junior Lecturer, Department of Electronics, Saurashtra University, Rajakot, Gujarat, India[3]

Visiting Lecture, Department of Electronics, Saurashtra University, Rajakot, Gujarat, India[4]

Abstract: Digital filtering algorithms are most commonly implemented using general purpose digital signal processing chips for audio applications, or special purpose digital filtering chips and application- specific integrated circuits (ASICs) for higher rates. This paper describes an approach to the implementation of digital filter algorithms based on field programmable gate arrays (FPGAs). The advantages of the FPGA approach to digital filter implementation include higher sampling rates than are available from traditional DSP chips, lower costs than an ASIC for moderate volume applications, and more flexibility than the alternate approaches. Since many current FPGA architectures are in-system programmable, the configuration of the device may be changed to implement different functionality if required. Our examples illustrate that the FPGA approach is both flexible and provides performance comparable or superior to traditional approaches.

Keyword: Digital Filter Module, VHDL (VHSIC Hardware Description Language), FPGA (Field Programmable Gate Arrays), Electronics Switch.

I. INTRODUCTION

Digital Signal Processing (DSP) affords greater flexibility, higher performance (in terms of attenuation and selectivity), better time and environment stability and lower equipment production costs than traditional analog techniques. Additionally, more and more microprocessor circuitry is being displaced with cost effective DSP techniques and products; an example of this is the emergence of DSP in cellular base stations. Components available today let DSP extend from baseband to intermediate frequencies (IFs). This makes DSP useful for tuning and signal selectivity, and frequency up and down conversion. The most common approaches to the implementation of digital filtering algorithms are general purpose digital signal processing chips for audio applications, or special purpose digital filtering chips and application-specific integrated circuits (ASICs) for higher rates. This paper describes an approach to the implementation of digital filter algorithms on field programmable gate arrays (FPGAs).

Recent advances in FPGA technology have enabled these devices to be applied to a variety of applications traditionally reserved for ASICs. FPGAs are well suited to data path designs, such as those encountered in digital filtering applications. The density of the new programmable devices is such that a nontrivial number of arithmetic operations such as those encountered in digital filtering may be implemented on a single device. The advantages of the FPGA approach to digital filter implementation include higher sampling rates than are available from traditional DSP chips, lower costs than an ASIC for moderate volume applications, and more flexibility than the alternate approaches. In particular, multiple multiply-accumulate (MAC) units may be implemented on a single FPGA, which provides comparable performance to general-purpose architectures which have a single MAC unit. Further, since many current FPGA architectures are In-system programmable, the configuration of the device may be changed to implement alternate filtering operations, such as lattice filters and gradient-based adaptive filters, or entirely different These new DSP applications result from advances in digital filtering. This Application Note will overview digital filtering by addressing concepts which can be extended to baseband processing on programmable digital signal processors.

Advantages of using digital filters

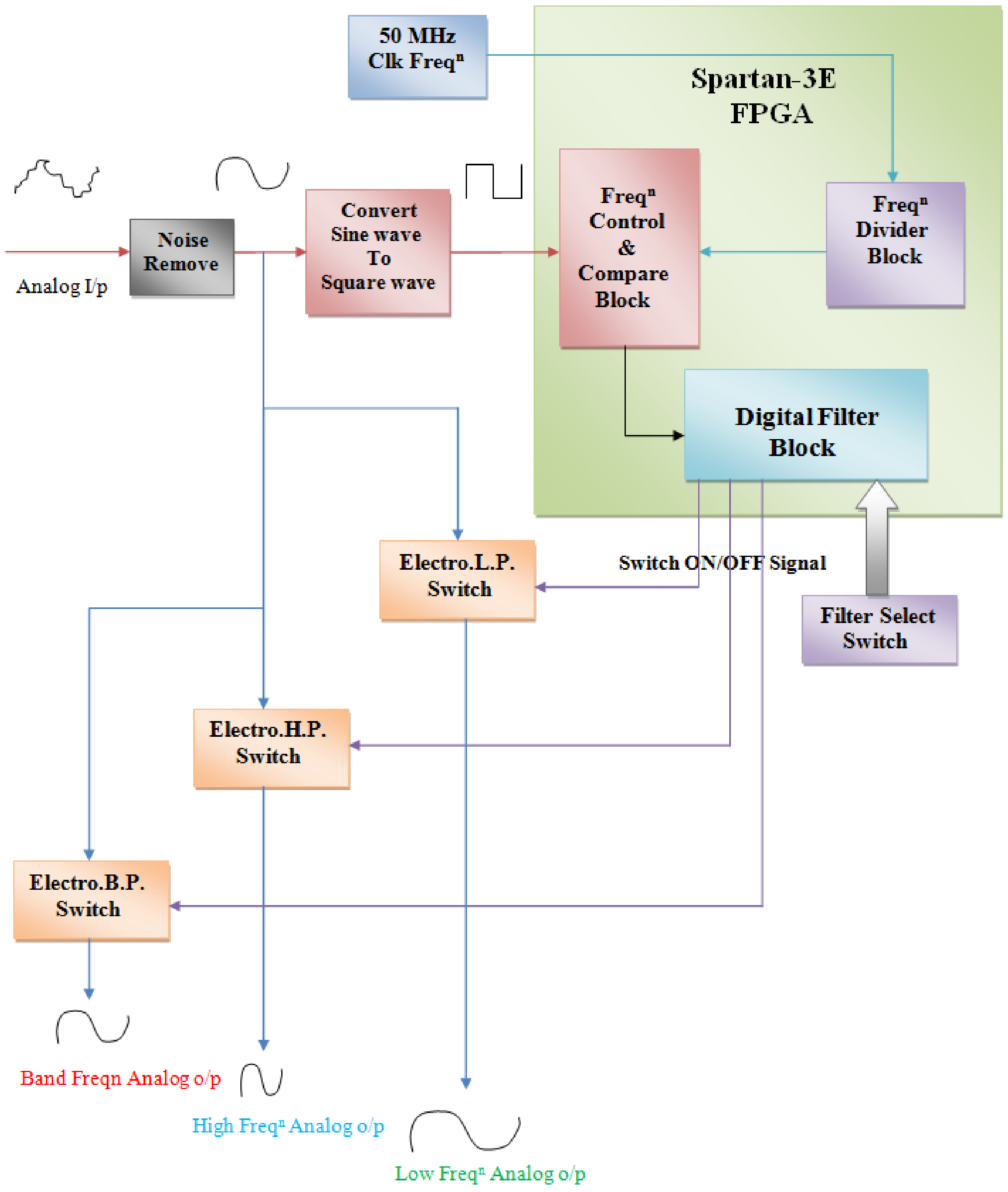
The following list gives some of the main advantages of digital over analog filters.

1. A digital filter is programmable, i.e. its operation is determined by a program stored in the processor's memory. This means the digital filter can easily be changed without affecting the circuitry (hardware). An analog filter can only be changed by redesigning the filter circuit.
2. Digital filters are easily designed, tested and implemented on a general-purpose computer or workstation.
3. The characteristics of analog filter circuits (particularly those containing active components) are subject to drift and are dependent on temperature. Digital filters do not

suffer from these problems, and so are extremely stable with respect both to time and temperature.

4. Unlike their analog counterparts, digital filters can handle low frequency signals accurately. As the speed of DSP technology continues to increase, digital filters are being applied to high frequency signals in the RF (radio frequency) domain, which in the past was the exclusive preserve of analog technology.
5. Digital filters are very much more versatile in their ability to process signals in a variety of ways; this includes the ability of some types of digital filter to adapt to changes in the characteristics of the signal.

II. FPGA BASE DIGITAL FILTER OVERALL SYSTEM



A. Contraction of digital filter

Shown as above figure digital filter module consist of Noise remove block, Sine wave to Square wave converter block, Frequency control & Compare block, Electronics switch and Digital filter block. Noise remove block, sine wave to square wave converter block and electronics switch create outside of FPGA. Reaming block create inside FPAG block using VHDL program.

Noise removes block consist of simple RC parallel network. Reactance of capacitor low for high frequency, so noise signal can't pass through RC network. Sine waves to square wave converter consist of BJT circuit. Here BJT operate in cut off and saturation region, therefore input base sine wave converts into collector square wave at the output of BJT. Another BJT and really use to making of electronics switch. Switch on/off control signal of digital filter output is high use for do on switch otherwise, switch off.

Frequency counter, frequency comparison, frequency divider and digital filter block prepare in side FPGA chip using VHDL program. FPGA chip is reconfigurable so, change functionality of all block as per over requirement. This is the main advantage of FPGA.

B. Working of digital filter

First fall analog signal with noise frequency apply to RC filter network. RC filter remove all high frequency unwanted noise signal pass only low frequency information signal. Output of RC filter is feed to two inputs, one is base of BJT and another is input of electronics switch. Electronics switch is controlled by digital output control signal. BJT operated in class c mode so, analog input convert into output pulse with same input frequency.

Converted pulse feed to input of frequency control block. This block consists of VHDL base frequency counter inside of FPGA. Frequency counter is totally controlled VHDL program, it is use to selection of cut off frequency. Parallel FPGA kit crystal produces 50MHz ref. clock frequency. This ref. frequency is applied to input of frequency divider block and they programmable by VHDL program, it is also use for selection of cut off frequency.

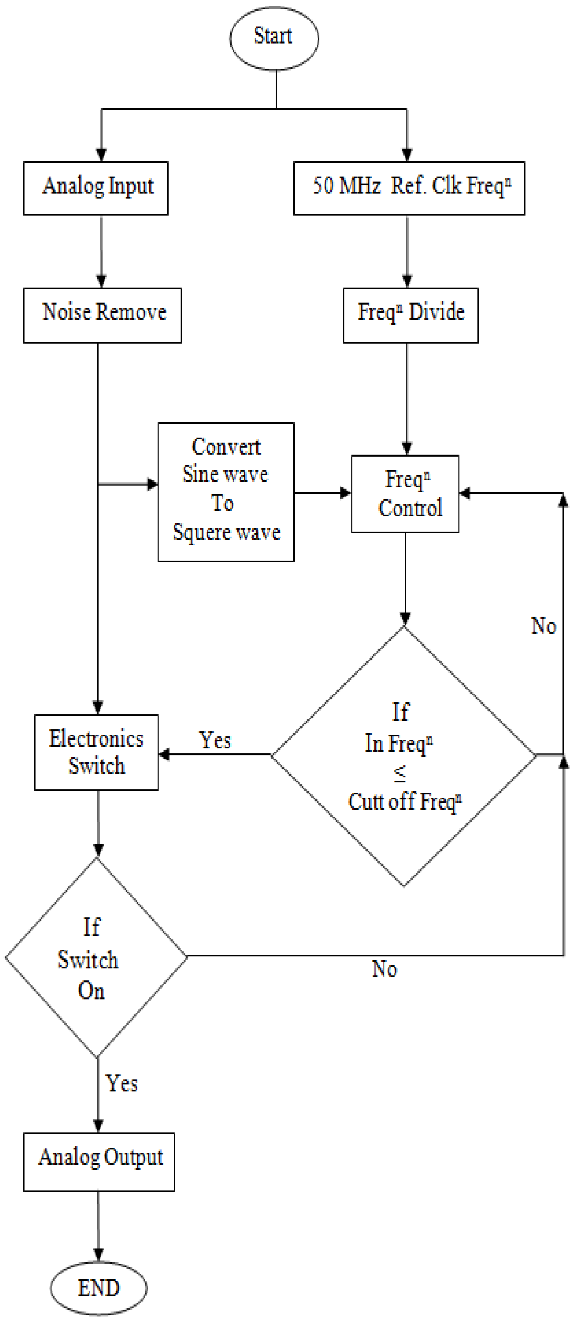
Frequency divider and frequency counter output feed to input of frequency compare block. Frequency compare make using VHDL program, like sequential statement object and if statement. After frequency comparisons block digital filter block divided into tree part like, low pass, high pass and band pass filter control block.

Low pass: In this section input signal feed into lower frequency cut off program. If input frequency is less than or equal to ref. clock frequency than low pass control signal is active. This low pass control signal is operate low pass electronics switch. Finally, switch is on and get selected low frequency analog signal at the output through switch.

High pass: In this section input signal feed into higher frequency cut off program. If input frequency is greater than or equal to ref. clock frequency than high pass control signal is active. This high pass control signal is operate high pass electronics switch. Finally, switch is on and get selected high frequency analog signal at the output through switch.

Band pass: In this section input signal feed into band frequency cut off program. If input frequency is greater than lower cut off and less than higher cut off of ref. clock frequency than band pass control signal is active.

LOW PASS



IJARCCE

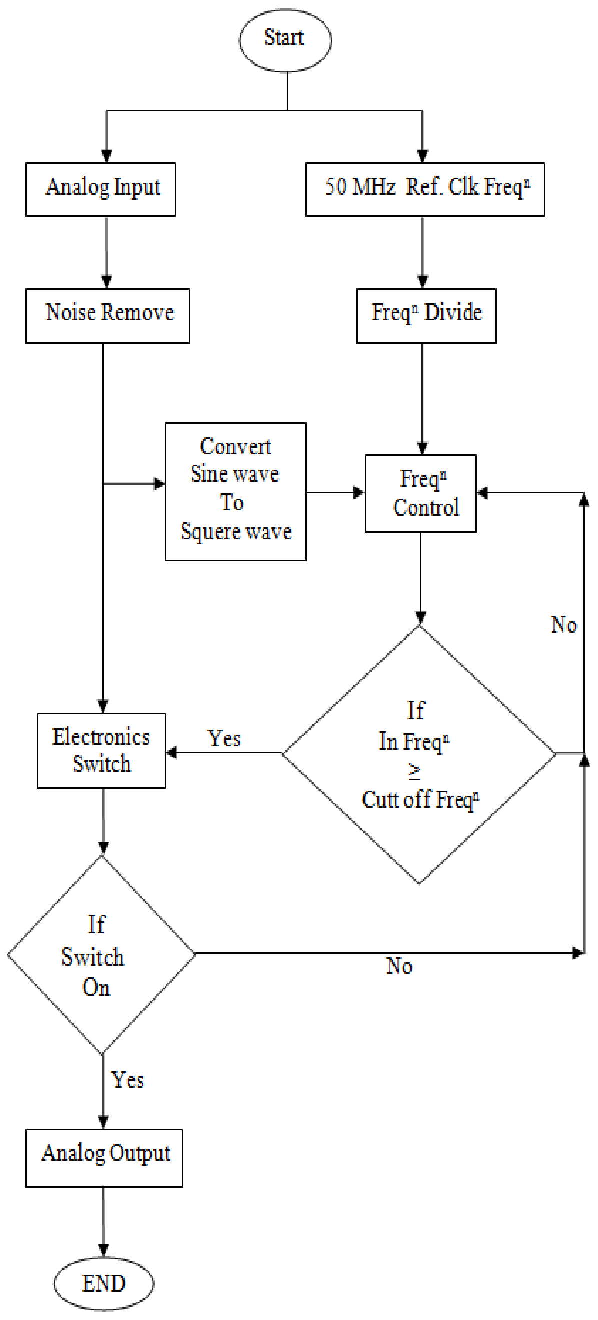
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 9, September 2016

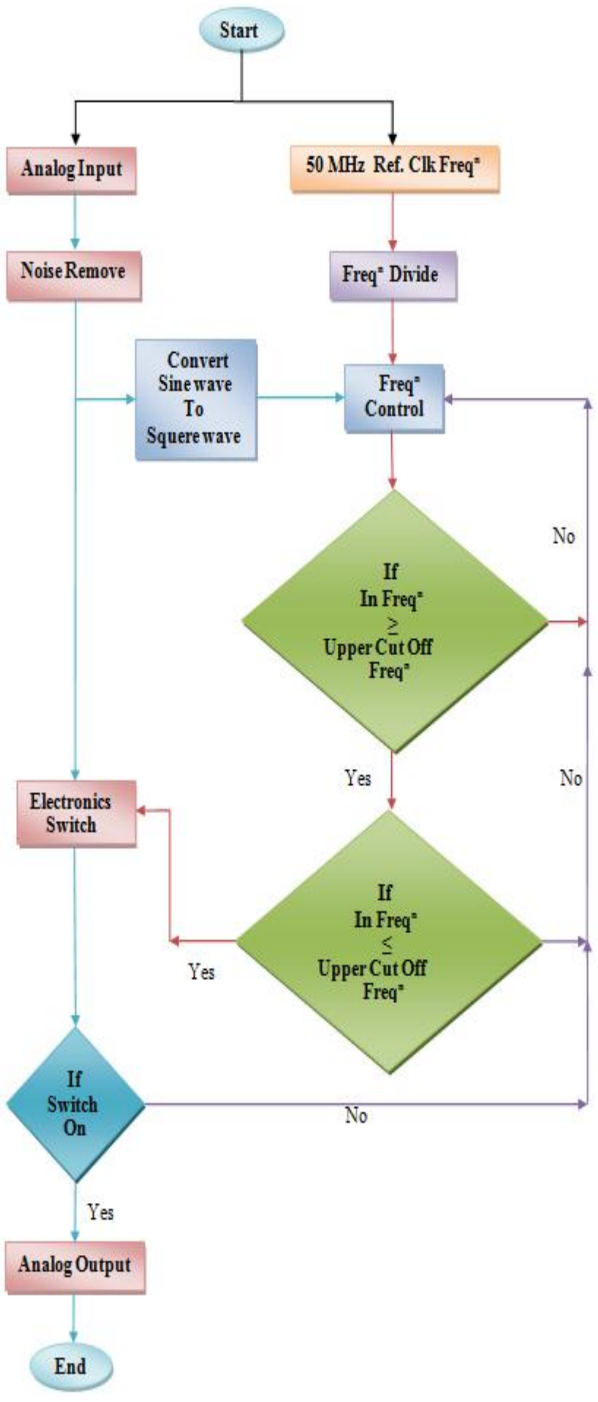
This band pass control signal is operate band pass electronics switch. Finally, switch is on and get selected center frequency analog signal at the output through switch.

Filter selection is use for selection of filter, using this can we select low pass, high pass, band pass and all filter. Use switch for on/off filter. All filter operation is independent to each other, so freely working. Characteristic of this filter is same to ideal filter, therefore it give very sharp cut off. It means roll off rate of this filter is very high. This is not possible for analog filter.

HIGH PASS



BAND PASS



III. VHDL PROGRAMM FOR DIGITAL FILTER

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
-- Create Date:      25/09/2016
-- Design Name:
-- Module Name:    dig - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
```

```
-- Dependencies:
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
------------------------------------------------------------------------
library IEEE;
Use IEEE.STD_LOGIC_1164.ALL;
Use IEEE.STD_LOGIC_ARITH.ALL;
Use IEEE.STD_LOGIC_UNSIGNED.ALL;
---- Uncomment the following library declaration if instantiating
---- Any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity fltr is
GENERIC (n : INTEGER := 7);
      Port ( rst,a,bclk,ls,hs,bs: in  STD_LOGIC;
           sc1,sc2,sc3,sc4,nxt,d1,m0,m1,m2,clk : inout  STD_LOGIC;
           bndpass,lps,hps,RS1,rs2,RW1,rw2,E1,e2: out  STD_LOGIC;
                    DB1 : out  STD_LOGIC_VECTOR(3 DOWNTO
0));
end fltr;

architecture Behavioral of fltr is
signal temp,q,q1,q2,q3,do,do1,do2,do3,dof :STD_LOGIC:='0' ;
signal lo,ho,E,RS,RW:STD_LOGIC:='0' ;
signal m: STD_LOGIC_VECTOR(0 to 1);
signal
DB,DB2,DB3,DB4,DB5,DB6,DB7,DB8,DB9:STD_LOGIC_VECTOR(
3 DOWNTO 0);
signal
FO,FO1,FO2,FO3,FO4,FO5,FO6,FO7,FO8:STD_LOGIC_VECTOR(3
DOWNTO 0);
signal temp1 : integer range 0 to 3;
signal temp2 : integer range 0 to 5;
signal temp3 : integer range 0 to 50;
signal temp5 : integer range 0 to 150;
signal temp4 : integer range 0 to 4;
signal btemp : integer range 0 to 4000000;
signal lcdcnt : integer range 0 to 164;
signal lcdcnt1 : integer range 0 to 144;
begin

-----board  clock -----50M--------
process(bclk)
      begin
            if(bclk'event and bclk = '0')then
                  btemp<= btemp+1;
                        if(btemp>=2000000)then
                              clk<='1';
                        else
                  clk<='0' ; end if; end if ;
end process;

---clock divider---
process(a)
      begin
            if(a'event and a = '0')then
                  temp4<= temp4+1;
                        if(temp4>=2)then
                              d1<='1';
                        else
            d1<='0' ; end if; end if ;
end process;

-------signal start------
process(rst)
     begin
  if(rst='1')then
            temp<='0';
     elsif(Clk'event and Clk = '1')then
             temp<= '1';
               end if;
end process;
   nxt<=temp ;

 ----low freqn senc -----
process(nxt,Clk,d1)
      begin
    if(nxt='1')then
        if(Clk='0')then
           if(d1'event and d1 = '0')then
                   temp1<= temp1+1;
                         if(temp1=3)then
                           temp1<=3; end if ; end if ;
                                else
                          temp1<=0; end if ; end if ;
    if(temp1=1)then
                         sc1<= '1';
                       else
                         sc1<= '0';
    end if ;
 end process;

---- medium freqn senc -----
 process(nxt,Clk,d1)
      begin
              if(nxt='1')then
                        if(Clk='0')then
                                 if(d1'event and d1 = '0')then
                   temp2<= temp2+1;
                                 if(temp2=3)then
                                  temp2<=3; end if ; end if ;
                                 else
                                    temp2<=0; end if ; end if ;
    if(temp2=2)then
           sc2<= '1';
                       else
              sc2<= '0';
     end if ;
end process;

----high freqn senc -----
process(nxt,Clk,d1)
        begin
              if(nxt='1')then
                       if(Clk='0')then
                        if(d1'event and d1 = '0')then
                             temp3<= temp3+1;
                             if(temp3=50)then
                                 temp3<=50; end if ; end if ;
                                         else
                                  temp3<=0; end if ; end if ;
    if(temp3=48)then
            sc3<= '1';
                      else
               sc3<= '0';
   end if ;
 end process;

----high high         freqn senc -----
process(nxt,Clk,d1)
             begin
              if(nxt='1')then
                      if(Clk='0')then
                          if(d1'event and d1 = '0')then
                            temp5<= temp5+1;
                                   if(temp5=100)then
                               temp5<=100; end if ; end if ;
                                          else
                                temp5<=0; end if ; end if ;

                      if(temp5=98)then
                          sc4<= '1';
                      else
                       sc4<= '0';
```

```
                         end if ;
end process;

-----lower cutoff-----
 q<= sc2 nor q1 ;q1<= sc3 nor q ;
 do<=  not q ; do1<= not q1;
 process(do,sc1)
             begin
              if(do='0')then
                              m0 <= '0';
                  elsif(sc1='1')then
                              m0 <= '1'; end if ;
end process;

-----higher cutoff------
  q2 <= sc3  nor q3 ;q3 <= sc4  nor q2 ;
  do2<= not q2 ; do3<= not q3 ;
 process(do2,sc1)
              begin
               if(do2='0')then
                              m1 <= '0';
                  elsif(sc1='1')then
                             m1 <= '1'; end if ;

 end process;
-----lower  cutoff ------
 process(do1,sc1)
          begin
    if(sc2='1')then
       if(do1='0')then
                m2 <= '0';
     else
        m2 <= '1'; end if ; end if ;
 end process;



 process(m0,m1,m2,ls,hs,bs)
      begin
-----low output-----
     if(ls='1')then
        lo<= (not m0) and (not m2)and temp ;
                   lps<= not lo and nxt ;
   end if ;
-----high output-----
     if(hs='1')then
         ho<= (not m0) and (not m2)and temp ;
                   hps<= ho and nxt ;
    end if ;
-----band output-----
      if(bs='1')then
          m(0)<= m0; m(1)<= m1;
                   end if ;
                 case m is
                   when "01" => dof <= '1';
                when others => dof <= '0';
       end case ;
     bndpass<= dof and nxt ;

end process;

process(Clk)
     begin
            if(ls='0')then
                if(hs='0')then
                   if(bs='0')then
                      if(Clk'event and Clk = '0')then
                            lcdcnt<= lcdcnt+1;
                                if(lcdcnt=164)then
                                    lcdcnt<=0;
                              end if ;
                          end if;
```

```
case lcdcnt is   ---------------higher first------then lower--------
when 16 => RS<= '0'; RW<= '0'; E<= '0'; DB<= "0000";--funct
when 17 => RS<= '0'; RW<= '0'; E<= '1'; DB<= "0000";
when 18 => RS<= '0'; RW<= '0'; E<= '0'; DB<= "0000";
when 19 => RS<= '0'; RW<= '0'; E<= '1'; DB<= "0000";
when 20 => RS<= '0'; RW<= '0'; E<= '0'; DB<= "0010";--funct
when 21 => RS<= '0'; RW<= '0'; E<= '1'; DB<= "0010";
when 22 => RS<= '0'; RW<= '0'; E<= '0'; DB<= "1000";
when 23 => RS<= '0'; RW<= '0'; E<= '1'; DB<= "1000";
when 24 => RS<= '0'; RW<= '0'; E<= '0'; DB<= "0010";--funct
when 25 => RS<= '0'; RW<= '0'; E<= '1'; DB<= "0010";
when 26 => RS<= '0'; RW<= '0'; E<= '0'; DB<= "1000";
when 27 => RS<= '0'; RW<= '0'; E<= '1'; DB<= "1000";
when 28 => RS<= '0'; RW<= '0'; E<= '0'; DB<= "0000";--entry mod
when 29 => RS<= '0'; RW<= '0'; E<= '1'; DB<= "0000";
when 30 => RS<= '0'; RW<= '0'; E<= '0'; DB<= "0110";
when 31 => RS<= '0'; RW<= '0'; E<= '1'; DB<= "0110";
when 32 => RS<= '0'; RW<= '0'; E<= '0'; DB<= "0000";--display on
 when 33 => RS<= '0'; RW<= '0'; E<= '1'; DB<= "0000";
when 34 => RS<= '0'; RW<= '0'; E<= '0'; DB<= "1100";
when 35 => RS<= '0'; RW<= '0'; E<= '1'; DB<= "1100";
when 36 => RS<= '0'; RW<= '0'; E<= '0'; DB<= "0000";--clear display
when 37 => RS<= '0'; RW<= '0'; E<= '1'; DB<= "0000";
when 38 => RS<= '0'; RW<= '0'; E<= '0'; DB<= "0001";
when 39 => RS<= '0'; RW<= '0'; E<= '1'; DB<= "0001";
when 40 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0101"; -- W
when 41 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0101";
when 42 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0111";
when 43 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0111";
when 44 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0100"; -- E
when 45 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0100";
when 46 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0101";
when 47 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0101";
when 48 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0100"; -- L
when 49 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0100";
when 50 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "1100";
when 51 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "1100";
when 52 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "1011"; -- -
when 53 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "1011";
when 54 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0000";
when 55 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0000";
when 56 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0100"; -- C
when 57 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0100";
when 58 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0011";
when 59 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0011";
when 60 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0100"; --
O
when 61 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0100";
when 62 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "1111";
when 63 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "1111";
when 64 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0100"; --
M
when 65 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0100";
when 66 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "1101";
when 67 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "1101";
when 68 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0100"; --
E
when 69 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0100";
when 70 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0101";
when 71 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0101";
when 72 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "1010"; -- Space
when 73 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "1010";
when 74 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0000";
when 75 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0000";
when 76 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0101"; -- T
when 77 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0101";
when 78 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0100";
when 79 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0100";
when 80 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0100"; -- O
when 81 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0100";
when 82 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "1111";
when 83 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "1111";
when 84 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "1010"; -- Space
when 85 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "1010";
when 86 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0000";
when 87 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0000";
when 88 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0101"; -- V
when 89 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0101";
when 90 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0110";
when 91 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0110";
when 92 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0100"; -- H
when 93 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0100";
when 94 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "1000";
when 95 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "1000";
when 96 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0100"; -- D
when 97 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0100";
when 98 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0100";
when 99 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0100";
when 100=> RS<= '1'; RW<= '0'; E<= '0'; DB<= "0100"; -- L
when 101=> RS<= '1'; RW<= '0'; E<= '1'; DB<= "0100";
when 102=> RS<= '1'; RW<= '0'; E<= '0'; DB<= "1100";
when 103=> RS<= '1'; RW<= '0'; E<= '1'; DB<= "1100";
when 104=> RS<= '0'; RW<= '0'; E<= '0'; DB<= "1100";-- give 2-line--
address
when 105=> RS<= '0'; RW<= '0'; E<= '1'; DB<= "1100";
when 106=> RS<= '0'; RW<= '0'; E<= '0'; DB<= "0011";
when 107=> RS<= '0'; RW<= '0'; E<= '1'; DB<= "0011";
when 108=> RS<= '1'; RW<= '0'; E<= '0'; DB<= "0100"; -

- D
when 109=> RS<= '1'; RW<= '0'; E<= '1'; DB<= "0100";
when 110=> RS<= '1'; RW<= '0'; E<= '0'; DB<= "0100";
when 111=> RS<= '1'; RW<= '0'; E<= '1'; DB<= "0100";
when 112=> RS<= '1'; RW<= '0'; E<= '0'; DB<= "0100"; -

- I
when 113=> RS<= '1'; RW<= '0'; E<= '1'; DB<= "0100";
when 114=> RS<= '1'; RW<= '0'; E<= '0'; DB<= "1001";
when 115=> RS<= '1'; RW<= '0'; E<= '1'; DB<= "1001";
when 116=> RS<= '1'; RW<= '0'; E<= '0'; DB<= "0100"; -

- G
when 117=> RS<= '1'; RW<= '0'; E<= '1'; DB<= "0100";
when 118=> RS<= '1'; RW<= '0'; E<= '0'; DB<= "0111";
when 119=> RS<= '1'; RW<= '0'; E<= '1'; DB<= "0111";
when 120=> RS<= '1'; RW<= '0'; E<= '0'; DB<= "0100"; -- when 121=>
RS<= '1'; RW<= '0'; E<= '1'; DB<= "0100";
when 122=> RS<= '1'; RW<= '0'; E<= '0'; DB<= "1001";
when 123=> RS<= '1'; RW<= '0'; E<= '1'; DB<= "1001";
when 124=> RS<= '1'; RW<= '0'; E<= '0'; DB<= "1010"; -- Space
when 125=> RS<= '1'; RW<= '0'; E<= '1'; DB<= "1010";
when 126=> RS<= '1'; RW<= '0'; E<= '0'; DB<= "0000";
when 127=> RS<= '1'; RW<= '0'; E<= '1'; DB<= "0000";
when 128=> RS<= '1'; RW<= '0'; E<= '0'; DB<= "0100"; -- F
when 129=> RS<= '1'; RW<= '0'; E<= '1'; DB<= "0100";
when 130=> RS<= '1'; RW<= '0'; E<= '0'; DB<= "0110";
when 131=> RS<= '1'; RW<= '0'; E<= '1'; DB<= "0110";
when 132=> RS<= '1'; RW<= '0'; E<= '0'; DB<= "0100"; -- I
when 133=> RS<= '1'; RW<= '0'; E<= '1'; DB<= "0100";
when 134=> RS<= '1'; RW<= '0'; E<= '0'; DB<= "1001";
when 135=> RS<= '1'; RW<= '0'; E<= '1'; DB<= "1001";
when 136=> RS<= '1'; RW<= '0'; E<= '0'; DB<= "0100"; -- L
when 137=> RS<= '1'; RW<= '0'; E<= '1'; DB<= "0100";
when 138=> RS<= '1'; RW<= '0'; E<= '0'; DB<= "1100";
when 139=> RS<= '1'; RW<= '0'; E<= '1'; DB<= "1100";
when 140=> RS<= '1'; RW<= '0'; E<= '0'; DB<= "0101"; -- T
when 141=> RS<= '1'; RW<= '0'; E<= '1'; DB<= "0101";
when 142=> RS<= '1'; RW<= '0'; E<= '0'; DB<= "0100";  when 143=>
RS<= '1'; RW<= '0'; E<= '1'; DB<= "0100";
when 144=> RS<= '1'; RW<= '0'; E<= '0'; DB<= "0100"; -- E
when 145=> RS<= '1'; RW<= '0'; E<= '1'; DB<= "0100";
when 146=> RS<= '1'; RW<= '0'; E<= '0'; DB<= "0101";
when 147=> RS<= '1'; RW<= '0'; E<= '1'; DB<= "0101";
when 148=> RS<= '1'; RW<= '0'; E<= '0'; DB<= "0101"; -- R
when 149=> RS<= '1'; RW<= '0'; E<= '1'; DB<= "0101";
when 150=> RS<= '1'; RW<= '0'; E<= '0'; DB<= "0010";
when 151=> RS<= '1'; RW<= '0'; E<= '1'; DB<= "0010";
when 152=> RS<= '1'; RW<= '0'; E<= '1'; DB<= "0010";
when 153=> RS<= '0'; RW<= '0'; E<= '0'; DB<= "0001";-- display shift
right
when 154=> RS<= '0'; RW<= '0'; E<= '1'; DB<= "0001";
when 155=> RS<= '0'; RW<= '0'; E<= '0'; DB<= "1100";
when 156=> RS<= '0'; RW<= '0'; E<= '1'; DB<= "1100";
when 157=> RS<= '0'; RW<= '0'; E<= '0'; DB<= "0001";-- display shift
left
when 158=> RS<= '0'; RW<= '0'; E<= '1'; DB<= "0001";
when 159=> RS<= '0'; RW<= '0'; E<= '0'; DB<= "1000";
when 160=> RS<= '0'; RW<= '0'; E<= '1'; DB<= "1000";
when 161=> RS<= '0'; RW<= '0'; E<= '0'; DB<= "0001";-- when 162=>
RS<= '0'; RW<= '0'; E<= '1'; DB<= "0001";
when 163=> RS<= '0'; RW<= '0'; E<= '0'; DB<= "1000";
when 164=> RS<= '0'; RW<= '0'; E<= '1'; DB<= "1000";
 when others =>null ;

end case;
  DB1<=DB ; E1<=E ; RS1<=RS ; RW1<= RW ;
     e2<=E ; rs2<=RS ; rw2<=  not RW ;
                  end if ;
            end if ;
end if ;
  if(ls='1' or hs = '1' or bs= '1')then
       if(Clk'event and Clk = '0')then
                lcdcnt1<= lcdcnt1+1;
                     if(lcdcnt1=144)then
                          lcdcnt1<=0;
                     end if ;
       end if;
                  case lcdcnt1 is   ---------------higher first------then lower-----
---
when 13 => RS<= '0'; RW<= '0'; E<= '0'; DB<= "0000";--clear display
when 14 => RS<= '0'; RW<= '0'; E<= '1'; DB<= "0000";
when 15 => RS<= '0'; RW<= '0'; E<= '0'; DB<= "0001";
when 16 => RS<= '0'; RW<= '0'; E<= '1'; DB<= "0001";
when 17 => RS<= '0'; RW<= '0'; E<= '0'; DB<= "0000";--entry mod
when 18 => RS<= '0'; RW<= '0'; E<= '1'; DB<= "0000";
when 19 => RS<= '0'; RW<= '0'; E<= '0'; DB<= "0110";
when 20 => RS<= '0'; RW<= '0'; E<= '1'; DB<= "0110";
when 21 => RS<= '0'; RW<= '0'; E<= '0'; DB<= "1000";--give 1-line--
address
when 22 => RS<= '0'; RW<= '0'; E<= '1'; DB<= "1000";
when 23 => RS<= '0'; RW<= '0'; E<= '0'; DB<= "0001";
when 24 => RS<= '0'; RW<= '0'; E<= '1'; DB<= "0001";
when 25 => RS<= '1'; RW<= '0'; E<= '0'; DB<= DB2; -- L  H B
when 26 => RS<= '1'; RW<= '0'; E<= '1'; DB<= DB2;
when 27 => RS<= '1'; RW<= '0'; E<= '0'; DB<= DB3;
when 28 => RS<= '1'; RW<= '0'; E<= '1'; DB<= DB3;
when 29 => RS<= '1'; RW<= '0'; E<= '0'; DB<= DB4; -- O  I  A
when 30 => RS<= '1'; RW<= '0'; E<= '1'; DB<= DB4;
when 31 => RS<= '1'; RW<= '0'; E<= '0'; DB<= DB5;
when 32 => RS<= '1'; RW<= '0'; E<= '1'; DB<= DB5;
when 33 => RS<= '1'; RW<= '0'; E<= '0'; DB<= DB6; -- W  G  N
when 34 => RS<= '1'; RW<= '0'; E<= '1'; DB<= DB6;
when 35 => RS<= '1'; RW<= '0'; E<= '0'; DB<= DB7;
when 36 => RS<= '1'; RW<= '0'; E<= '1'; DB<= DB7;
when 37 => RS<= '1'; RW<= '0'; E<= '0'; DB<= DB8; --

H   N
when 38 => RS<= '1'; RW<= '0'; E<= '1'; DB<= DB8;
when 39 => RS<= '1'; RW<= '0'; E<= '0'; DB<= DB9;
when 40 => RS<= '1'; RW<= '0'; E<= '1'; DB<= DB9;
when 41 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "1011"; -- -
when 42 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "1011";
when 43 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0000";
when 44 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0000";
when 45 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0101"; --

P
when 46 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0101";
when 47 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0000";
```

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

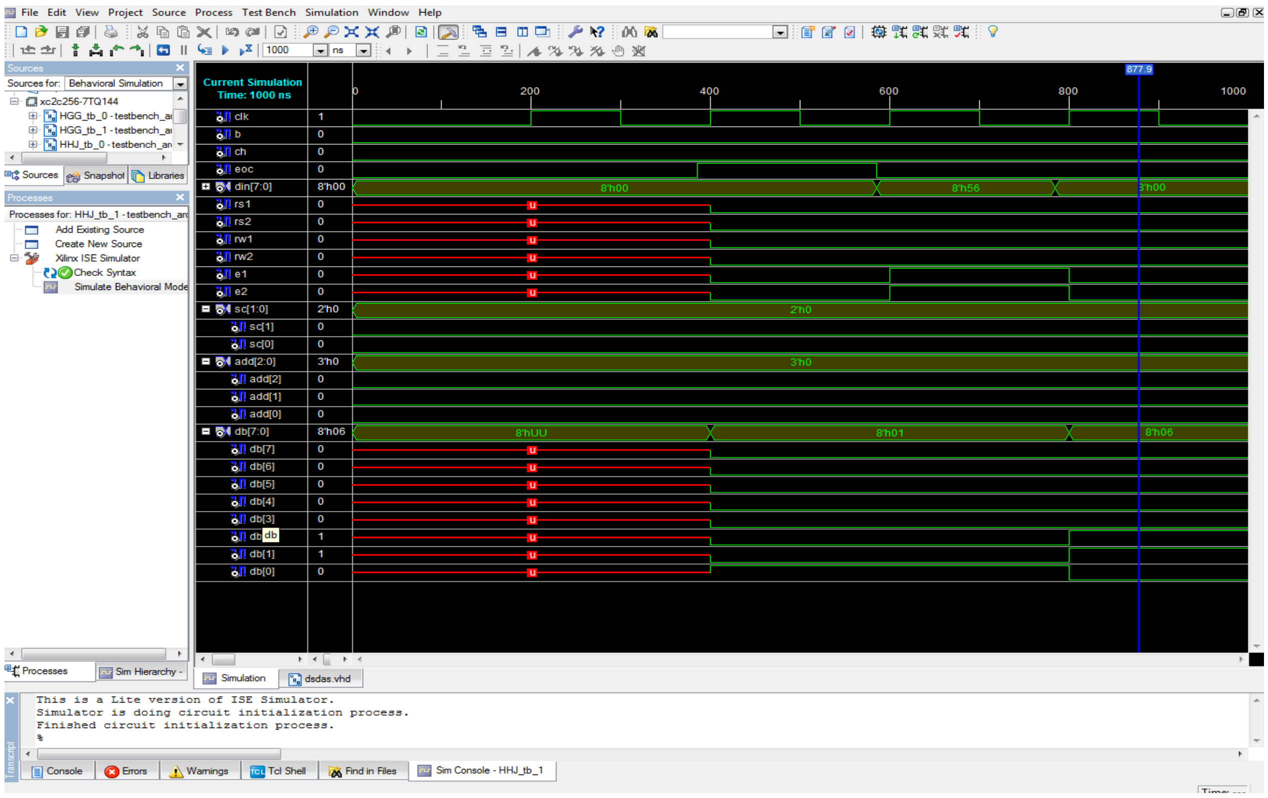
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 9, September 2016

```
when 48 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0000";
when 49 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0100"; -- A
when 50 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0100";
when 51 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0001";
when 52 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0001";
when 53 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0101"; -- S
when 54 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0101";
when 55 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0011";
when 56 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0011";
when 57 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0101"; -- S   when 58
=> RS<= '1'; RW<= '0'; E<= '1'; DB<= "0101";
when 59 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0011";
when 60 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0011";
when 61 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "1010"; -- Space
when 62 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "1010";
when 63 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0000";
when 64 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0000";
when 65 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0100"; -- F
when 66 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0100";
when 67 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0110";
when 68 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0110";
when 69 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0100"; -- I
when 70 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0100";
when 71 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "1001";
when 72 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "1001";
when 73 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0100"; -- L
when 74 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0100";
when 75 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "1100";
when 76 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "1100";
when 77 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0101"; -- T
when 78 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0101";
when 79 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0100";
when 80 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0100";
when 81 => RS<= '0'; RW<= '0'; E<= '0'; DB<= "1100";-- give 2-line—
address
when 82 => RS<= '0'; RW<= '0'; E<= '1'; DB<= "1100";
when 83 => RS<= '0'; RW<= '0'; E<= '0'; DB<= "0000";
when 84 => RS<= '0'; RW<= '0'; E<= '1'; DB<= "0000";
when 85 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0100"; -- C
when 86 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0100";
when 87 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0011";
when 88 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0011";
when 89 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0101"; -- U
when 90 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0101";
when 91 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0101";
when 92 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0101";
when 93 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0101"; -- T
when 94 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0101";
when 95 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0100";
when 96 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0100";
when 97 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0100"; -- O
when 98 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0100";
when 99 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "1111";
when 100 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "1111";
when 101 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0100"; -- F
when 102 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0100";
when 103 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0110";
when 104 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0110";
when 105 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0100"; -- F
when 106 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0100";
when 107 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0110";
when 108 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0110";
when 109 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "1010"; -- Space
when 110 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "1010";
when 111 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0000";
when 112 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0000";
when 113 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0100"; -- F
when 114 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0100";
when 115 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0110";
when 116 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0110";
when 117 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0101"; -- R
when 118 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0101";
when 119 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0010";
when 120 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0010";
when 121 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0100"; -- E
when 122 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0100";
when 123 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0101";
when 124 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0101";
when 125 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "1010"; -- Space
when 126 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "1010";
when 127 => RS<= '1'; RW<= '0'; E<= '0'; DB<= "0000";
when 128 => RS<= '1'; RW<= '0'; E<= '1'; DB<= "0000";
when 129 => RS<= '1'; RW<= '0'; E<= '0'; DB<= FO; -- P
when 130 => RS<= '1'; RW<= '0'; E<= '1'; DB<= FO;
when 131 => RS<= '1'; RW<= '0'; E<= '0'; DB<= FO1;
when 132 => RS<= '1'; RW<= '0'; E<= '1'; DB<= FO1;
when 133 => RS<= '1'; RW<= '0'; E<= '0'; DB<= FO2; -- U
when 134 => RS<= '1'; RW<= '0'; E<= '1'; DB<= FO2;
when 135 => RS<= '1'; RW<= '0'; E<= '0'; DB<= FO3;
when 136 => RS<= '1'; RW<= '0'; E<= '1'; DB<= FO3;
when 137 => RS<= '1'; RW<= '0'; E<= '0'; DB<= FO4; -- T
when 138 => RS<= '1'; RW<= '0'; E<= '1'; DB<= FO4;
when 139 => RS<= '1'; RW<= '0'; E<= '0'; DB<= FO5;
when 140 => RS<= '1'; RW<= '0'; E<= '1'; DB<= FO5;
when 141 => RS<= '1'; RW<= '0'; E<= '0'; DB<= FO6; -- T
when 142 => RS<= '1'; RW<= '0'; E<= '1'; DB<= FO6;
when 143 => RS<= '1'; RW<= '0'; E<= '0'; DB<= FO7;
when 144 => RS<= '1'; RW<= '0'; E<= '1'; DB<= FO7;
when others =>null ;
end case;

DB1<=DB ; E1<=E ; RS1<=RS ; RW1<= RW ;
e2<=E ; rs2<=RS ; rw2<=  not RW ;
    end if;
 end process;

 process(ls,hs,bs)

    begin
 if(ls='1')then
      DB2<= "0100"; DB3<= "1100"; DB4<= "0100";


       DB5<= "1111"; DB6<= "0101"; DB7<= "0111";
        DB8<= "1010";DB9<= "0000";
    elsif(hs='1')then
       DB2<= "0100"; DB3<= "1000"; DB4<= "0100";
         DB5<= "1001"; DB6<= "0100"; DB7<= "0111";
          DB8<= "0100";DB9<= "1000";


     elsif(bs='1')then
        DB2<= "0100"; DB3<= "0010"; DB4<= "0100";
         DB5<= "0001"; DB6<= "0100"; DB7<= "1110";
          DB8<= "0100";DB9<= "0100";
     end if ;
   end process;
 end Behavioral;
```

IV. SIMULATE BEHAVIORAL MODEL

You can now run a functional simulation on the display drive module. With display_drive.vhd highlighted in the **Source** window, the **Process** window will give all the available operations for the particular module. A VHDL file can be synthesized and then implemented through to a bit stream.

Normally, a design consists of several lower level modules wired together by a top-level file. In this instance, we are going to simulate only one lower-level module to introduce the functional simulation methodology.



```
This is a Lite version of ISE Simulator.
Simulator is doing circuit initialization process.
Finished circuit initialization process.
%
```

V. CONCLUSIONS

The proposed project-based approach encompasses whole engineering cycle, starting from specification, through design, modelling, simulation and verification, implementation, to performance measurement, and closing the cycle with the design improvements in order to maximize performance at minimal cost. Students have an opportunity to apply their theoretical knowledge of hardware description languages, digital design and computer architecture, and to gain real-world experience in developing IP cores. The work in small teams follows a real industry pattern, with one student designated as team leader, and instructor conducting design reviews every week. We feel that such projects are essential to educate future architects of complex systems-on-a-chip.

VI. REFERENCES

[1] Data Acquisition Linear devices Data Book - National Semiconductor.
[2] VHDL Programming by Example by Douglas L. Perry.
[3] Circuit Design with VHDL by Volnei A. Pedroni
[4] DIDITAL DESIGN principles & practices By John F. Wakerly
[5] Fundamentals of Digital Logic with VHDL Design by Stephen Brownand Zvonko Vranesic
[6] The Practical Xilinx Designer Lab Book- Prentice Hall by David Van den Bout
[7] The Practical Xilinx Designer Lab Book- Prentice Hall by David Van den Bout
[8] VHDL Starter's Guide.-Prentice Hall by Sudhakar Yalamanchili
[9] Digital Designing with Programmable Logic Devices.-Prentice Hall by John W. Carter
[10] International Journal of Advanced Research in Computer Science and Software Engineering (Volume 3, Issue 8, August 2013)
[11] Programmable Logic Design Quick Start Guide (UG500 (v1.0) May 8, 2008)

BIOGRAPHIES

Dr. Haresh Pandya Professor & Head, Department of Electronics, Saurashtra University, Rajkot, Gujrat, India. He obtained B.Sc, M.Sc. and Ph.D degrees from Saurashtra University-Rajkot. He specialized in the area of Electronic circuits and devices. Published large number of technical papers in deferent international journals.

Born in December 1986, **Mr. Mahesh Rangapariya** is currently working as a Ph.D. Student & Visiting Lecturer at Department of Electronics, Saurashtra University-Rajakot. Currently pursuing Master's Degree in ELECTRONISC from Department of Electronics Saradar Patel University-Vallabh Vidhaya Nagar. Attended Two-day "WOKRSHOPE ON EMBEDDED SYSTEMS" at Department of Electronics Saradar Patel University-Vallabh Vidhaya Nagar 2009. Attended One-day "WOKRSHOPE ON EMBEDDED SYSTEMS" at The Institution of Electronics and Telecommunication Engineers (IETE)-Rajkot 2010, he obtained B.Sc. degree in electronics from Gujarat University Ahmadabad in 2007.