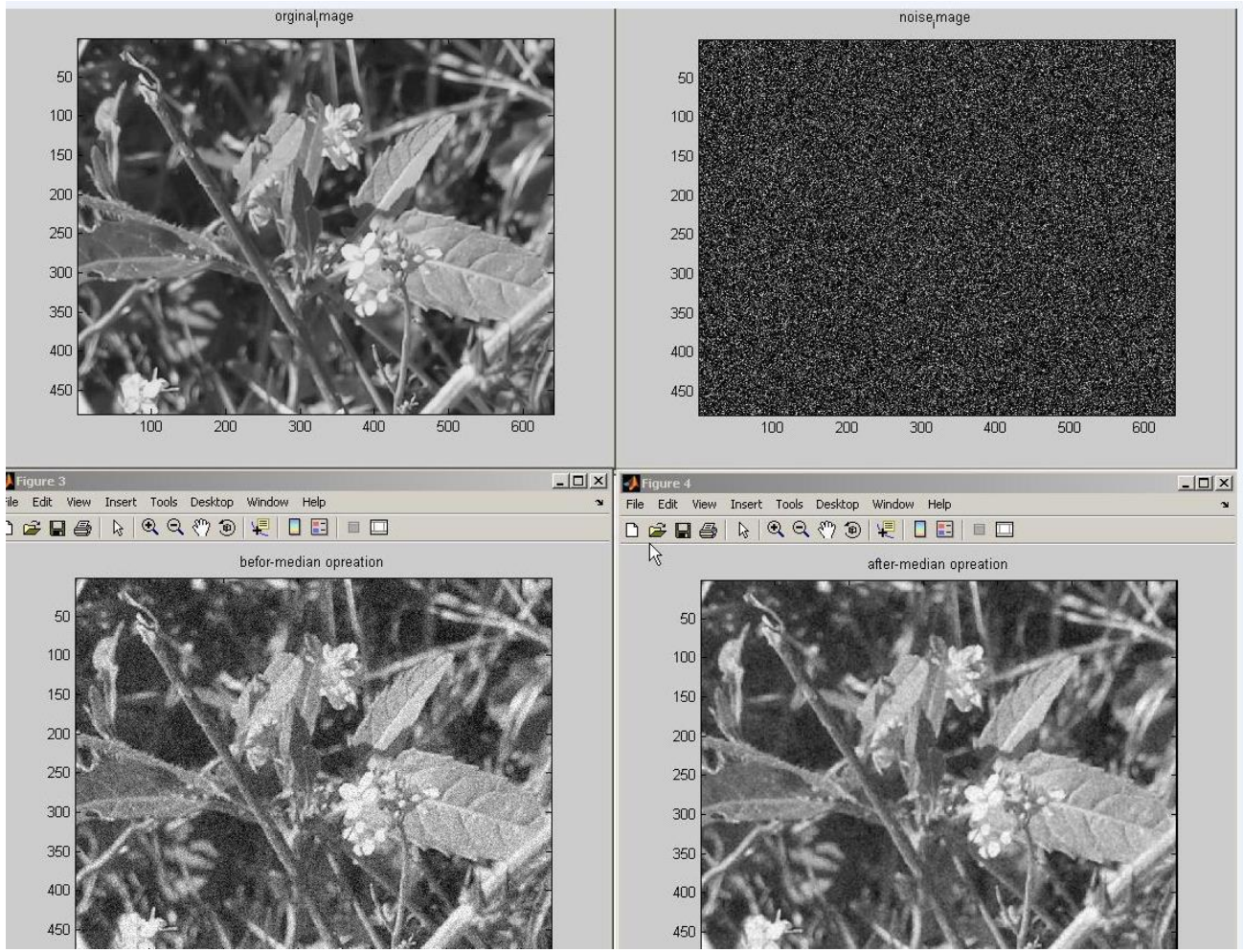
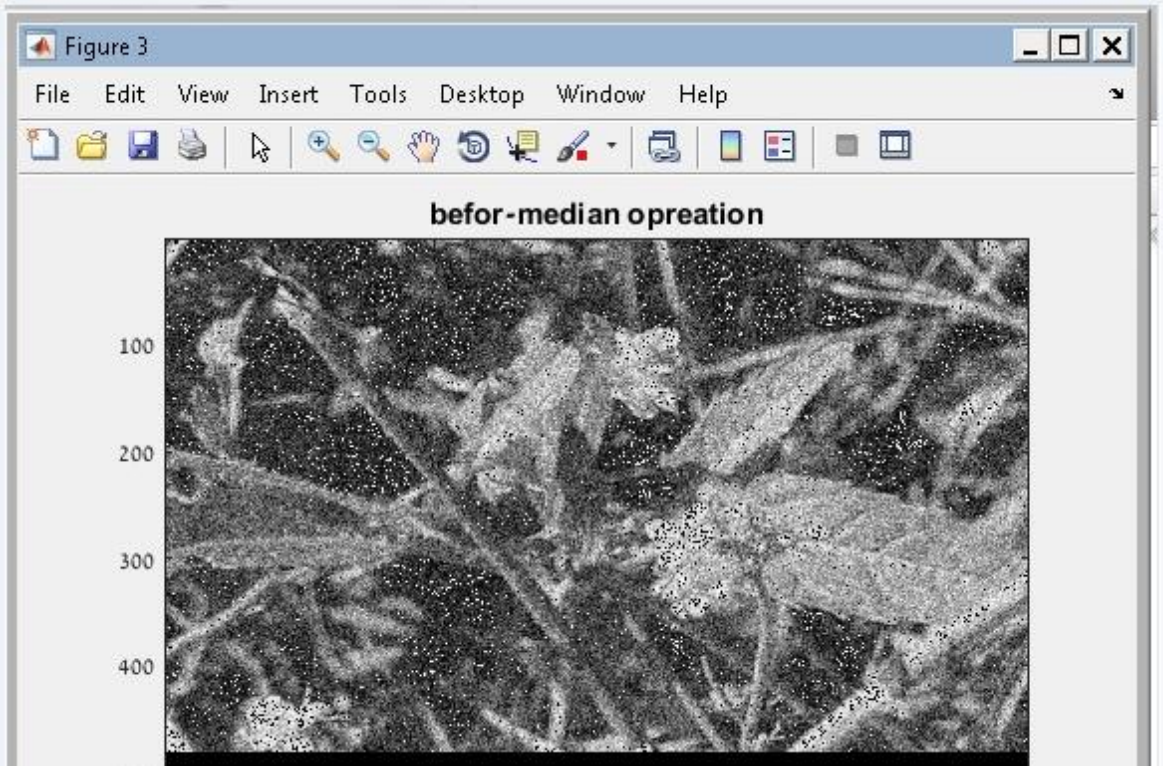
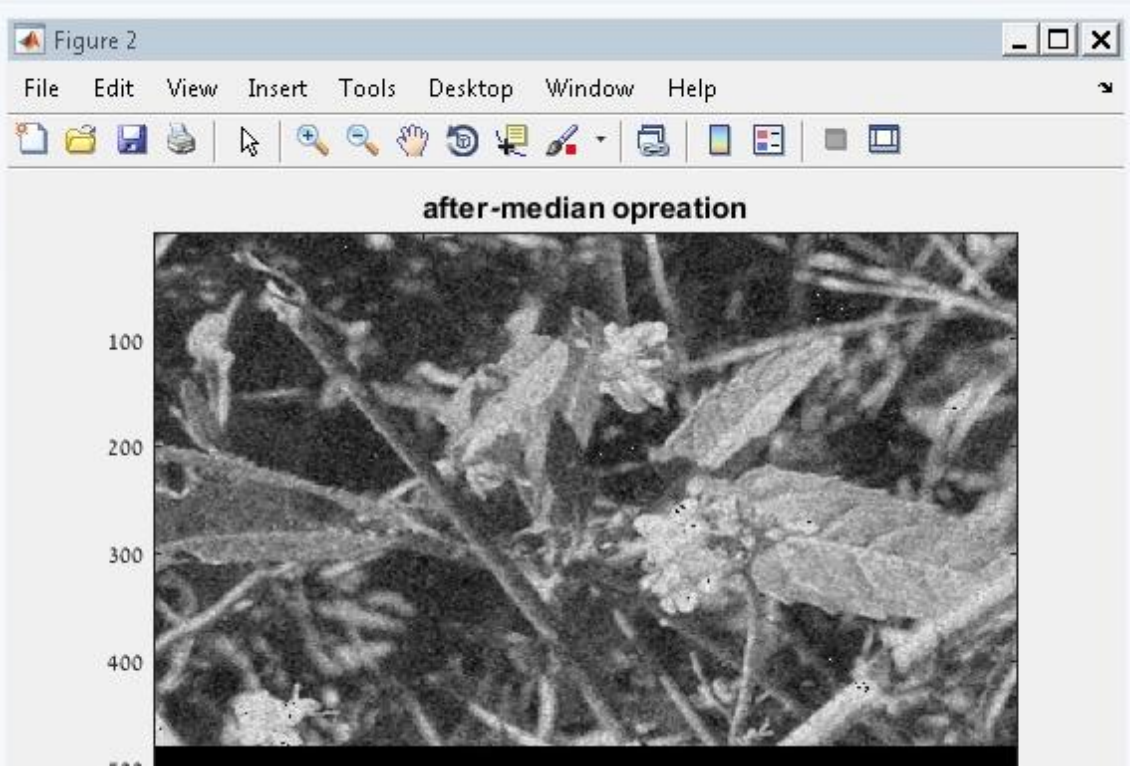
TEST VECTORS



TEXT I/O :VHDL TEST VECTORS





Text I/O :test bench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY test_bench IS
END test_bench;
```

Text I/O :test bench

```
ARCHITECTURE behavior OF test_bench IS

   -- Component Declaration befor the Unit Under Test (UUT)

   COMPONENT fpga_exp7
   PORT(
       clk : IN  std_logic;
       filed_enable : IN  std_logic;
       line_enable : IN  std_logic;
       line_sync : IN  std_logic;
       reset : IN  std_logic;
       PRE_cmos_data : OUT  std_logic_vector(7 downto 0)
      );
   END COMPONENT;
```

Text I/O :test bench

```
ARCHITECTURE behavior OF test_bench IS
------
-------

COMPONENT bh_write_file IS
  PORT(
    VIDEO_CLK                : IN     std_logic;
    VIDEO_OUT_FIELD_ENABLE    : IN     std_logic;
    VIDEO_OUT_LINE_ENABLE     : IN     std_logic;
    VIDEO_OUT_LINE_SYNC       : IN     std_logic;
    reset                     : IN     std_logic;
    EZOOM_DATA_OUTPUT         : IN     std_logic_vector(7 downto 0 )
  );


 END COMPONENT;
```

Text I/O :test bench

```
ARCHITECTURE behavior OF test_bench IS
------
-------
  --Inputs
   signal clk : std_logic := '0';
   signal filed_enable : std_logic := '0';
   signal line_enable : std_logic := '0';
   signal line_sync : std_logic := '0';
   signal reset : std_logic := '0';
--Outputs
   signal PRE_cmos_data : std_logic_vector(7 downto 0);
-- Clock period definitions
   constant clk_period : time := 36 ns;

BEGIN
```

Text I/O :test bench

```
ARCHITECTURE behavior OF test_bench IS
------
 -------
BEGIN
reset <= '1','0' after 200 ns ;
-- Instantiate the Unit Under Test (UUT)
  uut: fpga_exp7 PORT MAP (
        clk => clk,
        filed_enable => filed_enable,
        line_enable => line_enable,
        line_sync => line_sync,
        reset => reset,
        PRE_cmos_data => PRE_cmos_data
       );
```

Text I/O :test bench

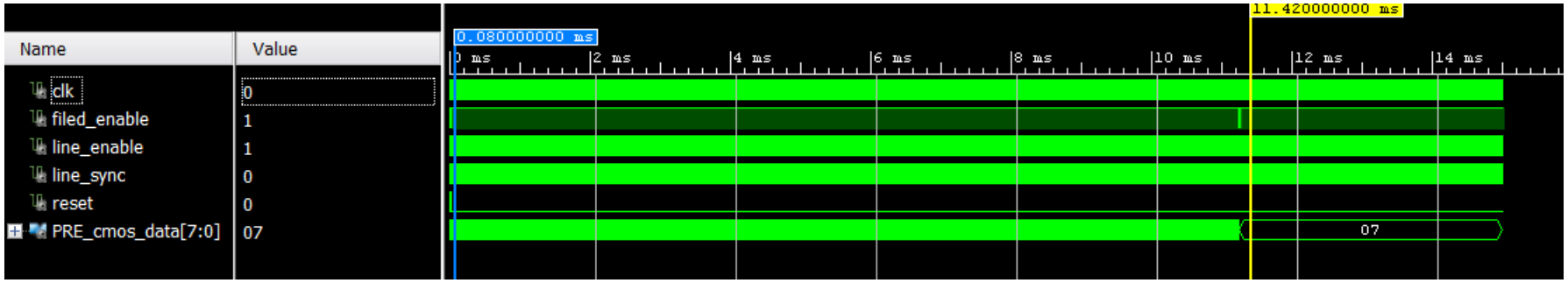
```
ARCHITECTURE behavior OF test_bench IS
------
 -------
BEGIN
reset <= '1','0' after 200 ns ;
-- Instantiate the Unit Under Test (UUT)
  uut: fpga_exp7 PORT MAP (
        clk => clk,
        filed_enable => filed_enable,
        line_enable => line_enable,
        line_sync => line_sync,
        reset => reset,
        PRE_cmos_data => PRE_cmos_data
       );
```

Text I/O :test bench

```
ARCHITECTURE behavior OF test_bench IS
 ------------
BEGIN
-----------------------------------------------
uut1:bh_write_file
  PORT map(
    VIDEO_CLK                => clk,
    VIDEO_OUT_FIELD_ENABLE    => filed_enable,
    VIDEO_OUT_LINE_ENABLE     => line_enable,
    VIDEO_OUT_LINE_SYNC       => line_sync,
    reset                    => reset,
    EZOOM_DATA_OUTPUT        => PRE_cmos_data
  );
```

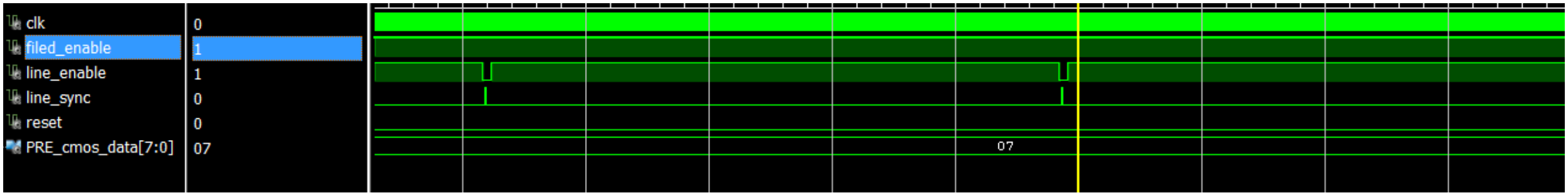
Text I/O :test bench

```
-- Clock process definitions
  clk_process :process
  begin
clk <= '0';
wait for clk_period/2;
clk <= '1';
wait for clk_period/2;
  end process;
```



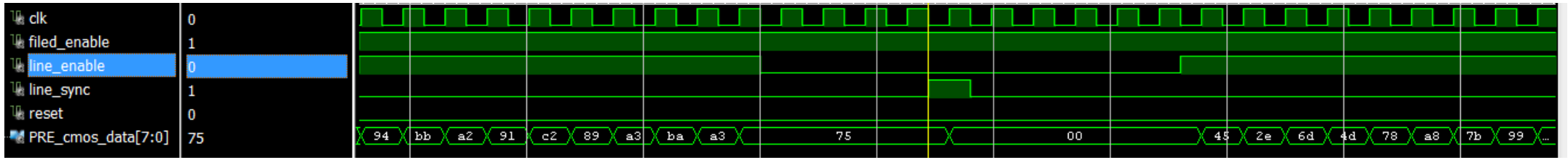
Text I/O :test bench

```
-----------------------------------------
line_enable_process :process
  begin
line_enable <= '0';
wait for clk_period*10;
line_enable <= '1';
wait for clk_period*640;
  end process;
```



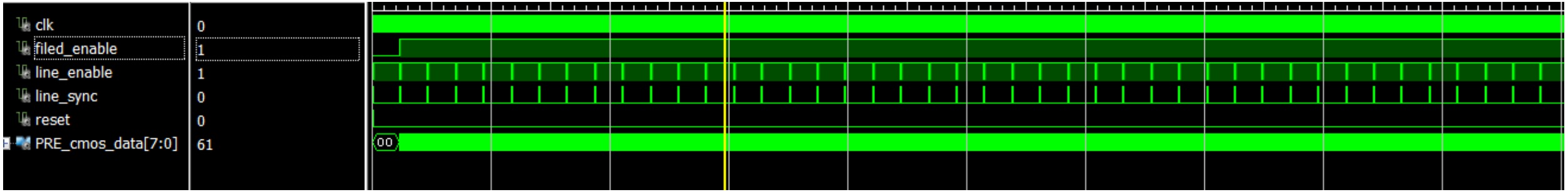
Text I/O :test bench

```
-----------------------------------
line_sync_process :process
  begin
line_sync <= '0';
wait for clk_period*4;
line_sync<= '1';
wait for clk_period*1;
line_sync<= '0';
wait for clk_period*645;
```



Text I/O :test bench

```
-----------------------------------------
filed_enable_process :process
  begin
filed_enable <= '0';
wait for clk_period*650;
filed_enable <= '1';
wait for clk_period*312000;
  end process;
```



Text I/O :read process

```
LIBRARY std;
USE std.textio.ALL;
USE ieee.std_logic_textio.ALL;
ENTITY fpga_exp7 IS
  PORT(
    clk              : IN    std_logic;
    filed_enable     : IN    std_logic;
    line_enable      : IN    std_logic;
    line_sync        : IN    std_logic;
    reset            : IN    std_logic;
    PRE_cmos_data    : OUT   std_logic_vector (7 DOWNTO 0)
  );
END fpga_exp7 ;
```

Text I/O :read process

```
ARCHITECTURE bh_fpga_exp7 OF fpga_exp7 IS

file input_file1 :text open read_mode is "C:\Yelement102_dirt.txt";
signal PRE_cmos_data1:std_logic_vector(7 DOWNTO 0);

BEGIN

process(clk, RESET)
variable l                    : line;
variable temp_input : integer;

begin
```

Text I/O :read process

```
if RESET = '1' then
                PRE_cmos_data1     <=(others=>'0') ;
 elsif clk = '1' and clk'event then
      if line_sync='1' and filed_enable='1' then
                    if not(endfile(input_file1)) then
                              readline(input_file1,l);
                              PRE_cmos_data1  <=(others=>'0') ;
                     end if;

      elsif  filed_enable='1' and line_enable='1' then
                               read(l,temp_input);
                               PRE_cmos_data1 <=conv_std_logic_vector(temp_input,8);
      end if;
     end if;
end process;
```

Text I/O :read process

```
PRE_cmos_data <=(PRE_cmos_data1(7))&PRE_cmos_data1(6 DOWNTO 0);

END ARCHITECTURE bh_fpga_exp7;
```

Text I/O : write process

```
LIBRARY std;
USE std.textio.ALL;
USE ieee.std_logic_textio.ALL;
ENTITY bh_write_file IS
  PORT(
    VIDEO_CLK                : IN     std_logic;
    VIDEO_OUT_FIELD_ENABLE    : IN     std_logic;
    VIDEO_OUT_LINE_ENABLE     : IN     std_logic;
    VIDEO_OUT_LINE_SYNC       : IN     std_logic;
    reset                     : IN     std_logic;
    EZOOM_DATA_OUTPUT         : IN     std_logic_vector(7 downto 0 )
  );
END ENTITY bh_write_file;
```

Text I/O : write process

```
ARCHITECTURE write_file OF bh_write_file IS

signal  EZOOM_DATA_OUTPUT1        :  std_logic_vector(7 downto 0 );
signal  VIDEO_OUT_LINE_SYNC1 :std_logic;
signal  bh_num :std_logic_vector(10 downto 0);
 file output_file1 :text open write_mode is "C:\median_out_nir.txt";

BEGIN
```

Text I/O : write process

```
process( VIDEO_CLK, RESET)
 begin
  if RESET = '1' then
                VIDEO_OUT_LINE_SYNC1 <='0';

  elsif rising_edge(VIDEO_CLK) then
              VIDEO_OUT_LINE_SYNC1 <= VIDEO_OUT_LINE_SYNC;
  end if ;
 end process;
```

Text I/O : write process

```
 process( VIDEO_CLK, RESET)

variable l: line;
variable temp_output : integer := 0;

Begin
  if RESET = '1' then
        temp_output                 := 0;
             bh_num <= (others=>'0');
  elsif VIDEO_CLK = '1' and VIDEO_CLK'event then
            EZOOM_DATA_OUTPUT1  <=        EZOOM_DATA_OUTPUT;
```

Text I/O : write process

```
if VIDEO_OUT_LINE_SYNC ='1'' and VIDEO_OUT_LINE_SYNC1 ='0' and VIDEO_OUT_FIELD_ENABLE ='1 then
                    writeline(output_file1,l);
                    bh_num <= (others=>'0');
elsif VIDEO_OUT_FIELD_ENABLE='1' and  VIDEO_OUT_LINE_ENABLE ='1' and   ( bh_num <"1010000000" )  then
        temp_output := conv_integer(unsigned( EZOOM_DATA_OUTPUT ));
         write(l,temp_output);
         write(l,' ');
         bh_num <=    bh_num+1;
  end if;     end if;
end process;
END ARCHITECTURE write_file;
```