References

[1] J. R. Armstrong and F.G. Gray, *Structured Logic Design with VHDL*. New Jersey: Prentice Hall,1993.

[2] P. J. Ashenden, *The Designer's Guide to VHDL*. San Francisco: Morgan Kaufmann Publishers, Inc.,1996.

[3] M. Abramovici, M. Breur, A. Friedman, *Digital Systems Testing and Testable Design*. New York: Computer Science Press,Inc.,1990.

[4] D. K. Pradhan, *Fault-Tolerant Computing Theory and techniques.* New Jersey: Prentice Hall, 1983

[5] J. R. Armstrong, G. A. Frank, S. Hrishikesh, P. Gowrisankaran, Z. Xu, "Test bench Development for RASSP DSP Models, "*Proceedings of the First Annual RASSP Conference*, Arlington, VA, August 15-18, 1994, pp. 91-96.

[6] H. William, T. Lee, I. Arthur, N. L. Hoft, *Web Page Design.* New York: John Wiley and Sons,Inc.,1996.

[7] J.E.P Miranda, J.S. Pinto, " Using Internet technology for course support," *SIGCSE Bulletin, vol. 28,* pp. 96-100, June 1996.

[8] J. Mayfield, S. K. Ali, "The Internet as an Educational tool, " *Computers and Industrial Engineering, vol. 31*, no. 1-2, pp. 21-4, October 1996.

[9] "IEEE Standard for Waveform and Vector Exchange (WAVES)," 1992. *IEEE Standard* [1029.1-1991].

[10] *IEEE Standard VHDL Language Reference Manual* [IEEE Std 1076-1987].

[11] V.E. Veraart and S.L Wright, " Supporting Software Engineer Education with a Local Web Site," *SIGCSE Bulletin*, pp. 275-279, February 96.

[12] P. Gowrishankaran, *Structural Testbench development for DSP models*, Master's thesis, March 1993.

[13] P. Narendran, J. Stillman, " Formal Verification of the Sobel Image processing Chip," *Proceedings of the 25th ACM/IEEE Design Automation Conference*, Anaheim, CA, June 1988, pp. 211-217.

[14] D.S. Suk, S.M. Reddy, "A March Test for Functional Faults in Semiconductor Random Access Memories," *IEEE Transactions on Computers*, vol. C-30, no. 12, pp. 982-985, December 1981.

[15] R. L. Baker, C. O. Scheper, "Image Processing Design using Silicon Compilation," *Generic Signal Processing Workshop*, Laurel, MD, July 1987.

[16] Weihong Song, *Development of Web-Based Educational modules for Designing VHDL Models of Digital systems*, Master's Thesis, August 1997.

[17] S. Downes, "Effective Interaction and Communication with web courses," *Presentation at the International University Consortium Computer-based Conference*, Fall 1996.

[18] R. Jain, R Kasturi, B. G. Schunk, *Machine Vision*. New York: McGraw Hill Inc.,1995, Chapter 5, page 147.

[19] V. P. Nelson, H.T. Nagle, B. D. Carroll, J. D. Irwin, *Digital Logic Circuit Analysis and Design*. New Jersey: Prentice Hall, 1995, Chapter 12, page 739.

[20] M. A. Breur, A. D. Friedman, *Diagnosis & Reliable Design of Digital Systems.* California: Computer Science Press,Inc.,1976, Chapter 1, pp. 15-18.

[21] V. D. Agrawal, K-T. Cheng, P. Agrawal, " A Concurrent Test Generator for Sequential Circuits," *Proceedings of the 25th Design Automation Conference*, Anaheim, CA, June 1988, pp. 84-89,

[22] H. Fujiwara, T. Shimono, " On the Acceleration of Test Generation Algorithms," *IEEE Transactions on Computers, vol. CAD-2*, pp. 1137-1144, December 1983.

[23] P. Goel, “ An Implicit Enumeration Algorithm to generate tests for Combinational Logic circuits,” *IEEE Transactions on Computers, vol. C-30*, pp. 215-222, March 1981.

[24] H-K. T. Ma, S. Devadas, A. R. Newton, A. S. Vincentelli, “ Test Generation for Sequential Circuits, “ *IEEE Transactions on Computer-Aided Design, vol. 7*, pp. 1081-1093, October 1988.

Appendix A: Test bench and configuration models

This appendix contains source codes of all test benches and configuration files for the sub-components of the Sobel Edge detector of Integer data type. Test bench models for the sub-components at all the other abstraction levels have been developed in a similar fashion.

TEST BENCH FOR ADDRESS GENERATOR - INTEGER DATA TYPE

```
-- file name: addrtb.vhd
-- purpose: TEST BENCH for the address generator - (file name:addr_gen.vhd) – TYPE INTEGER
-- LIBRARY vhdlwork
--------------------------LIBRARY DECLARATIONS-----------------------------
library IEEE;
use ieee.STD_LOGIC_1164.all;

library BEH_INT;
use BEH_INT.IMAGE_PROCESSING.all;
use BEH_INT.all;

library STRUC_INT;
use STRUC_INT.all;
--------------------------ENTITY TEST BENCH---------------------------------
entity TB is
end TB;
--------------------------ARCHITECTURE BODY---------------------------------
architecture TB of TB is

signal RUN: STD_LOGIC;
signal START: STD_LOGIC:='0';
signal CLOCK: STD_LOGIC:='0';
signal X_ADDR1 : NATURAL:=1;
signal X_ADDR2 : NATURAL:=1;
signal X_ADDR3 : NATURAL:=1;
signal Y_ADDR : NATURAL:=1;
signal RWB: STD_LOGIC:='0';
--------------------------COMPONENT INSTANTIATIONS-------------------------
component CLOCK_GENERATOR1
generic(HI_TIME,LO_TIME:TIME);
  port(  RUN : in STD_LOGIC;
        CLOCK: out STD_LOGIC);
end component;

component ADDR_GEN1
generic(ADDR_A_DELAY, RWB_A_DELAY:TIME;
       NUM_ROWS,NUM_COLS:NATURAL);
port(      START: in STD_LOGIC:='0';
           CLOCK: in STD_LOGIC:='0';
       X_ADDR1 : out NATURAL:=1;
           X_ADDR2 : out NATURAL:=1;
           X_ADDR3 : out NATURAL:=1;
           Y_ADDR : out NATURAL:=1;
           RWB: out STD_LOGIC:='0');
end component;
--------------------CONFIGURATION SPECIFICATIONS---------------------------
 for L1: CLOCK_GENERATOR1 use entity BEH_INT.CLOCK_GENERATOR(BEHAVIOR);
for L2: ADDR_GEN1 use entity STRUC_INT.ADDR_GEN(BEHAVIOR);
--------------------MAPPING-----------------------------------------------
```

```
begin
L1:CLOCK_GENERATOR1
generic map(HI_TIME=>75 ns,LO_TIME=>25 ns)
port map(RUN,CLOCK);

L2: ADDR_GEN1
generic map(ADDR_A_DELAY=> 2 ns, RWB_A_DELAY=>1 ns,
            NUM_ROWS => 5, NUM_COLS=> 6)
port map(START, CLOCK, X_ADDR1, X_ADDR2, X_ADDR3,
       Y_ADDR,RWB);

RUN<=transport '0' after 0 ns, '1' after 10 ns, '0' after  10000 ns;
START<='0' after 0 ns, '1' after 101 ns, '0' after 201 ns;

end TB;
```

Figure A-1 Source code for the test bench model of the Address generator in STRUC_INT library

TEST BENCH FOR MEMORY PROCESSOR - INTEGER DATA TYPE

```
-- file name: memsys_tb.vhd
-- purpose: TEST BENCH for the memory system : (file name - mem_sys2.vhd)
--              type integer
--library vhdlwork
---------------------------LIBRARY DECLARATIONS--------------------------------
library IEEE;
use ieee.STD_LOGIC_1164.all;
library BEH_INT;
use BEH_INT.IMAGE_PROCESSING.all;
library STRUC_INT;
use STRUC_INT.all;
---------------------------ENTITY TEST BENCH-----------------------------------
entity TB is
end TB;
---------------------------ARCHITECTURE BODY-----------------------------------
architecture MEMSYS_INT of TB is

signal RUN, CLOCK,START: STD_LOGIC:='0';
signal MEM_IN: PIXEL:=0;
signal MEM_OUT1, MEM_OUT2, MEM_OUT3:PIXEL:=0;
---------------------------COMPONENT INSTANTIATIONS----------------------------
component CLOCK_GENERATOR1
generic(HI_TIME,LO_TIME:TIME);
  port(  RUN : in STD_LOGIC;
       CLOCK: out STD_LOGIC);
end component;

component MEMORY_PROCESSOR1
 generic(NUM_ROWS, NUM_COLS:NATURAL; MEM_OUT_DELAY:TIME);
 port(CLOCK: in STD_LOGIC:='0';
```

```
    Start: in STD_LOGIC:='0';
    MEM_IN: in PIXEL:=0;
    MEM_OUT1, MEM_OUT2, MEM_OUT3: out PIXEL:=0);
end component;
-------------------CONFIGURATION SPECIFICATIONS--------------------------
for L1: CLOCK_GENERATOR1 use entity BEH_INT.CLOCK_GENERATOR(BEHAVIOR);
for L2: MEMORY_PROCESSOR1 use configuration STRUC_INT.MEMSYS_INT;
-------------------MAPPING-----------------------------------------------
begin
L1: CLOCK_GENERATOR1
generic map(HI_TIME=>75 ns,LO_TIME=>25 ns)
port map(RUN,CLOCK);

L2: MEMORY_PROCESSOR1
generic map(NUM_ROWS=>8, NUM_COLS=>3, MEM_OUT_DELAY=>2 ns)
port map(CLOCK,START,MEM_IN,MEM_OUT1, MEM_OUT2, MEM_OUT3);

RUN<=transport '1' after 0 ns,'0' after  4000 ns;
START <= '0' after 0 ns,'1' after 1 ns, '0' after 101 ns;
MEM_IN<=        1 after 110 ns,      4 after 210 ns,  7 after 310 ns,
                4  after 410 ns,3 after 510 ns, 6 after 610 ns,
                5 after 710 ns,  6 after 810 ns,1 after 910 ns,
                8 after 1010 ns,7 after 1110 ns, 30 after 1210 ns,
                9 after 1310 ns, 26 after 1410 ns, 11 after 1510 ns,
                3 after 1610 ns,10 after 1710 ns,9 after 1810 ns,
                14 after 1910 ns,14 after 2010 ns,1 after 2110 ns,
                16  after 2210 ns, 12 after 2310 ns,14 after 2410 ns;
end MEMSYS_INT;
```

Figure A-2 Source code for the test bench model of the Memory processor in STRUC_INT library

TEST BENCH FOR HORIZONTAL FILTER - INTEGER DATATYPE-STRUC_INT LIBRARY

```
-- file name: horizb_tb.vhd
-- purpose:   TEST BENCH for HORIZONTAL FILTER, file name :horiz_b.vhd  --type integer
-- library:   vhdlwork
--------------------------LIBRARY DECLARATIONS--------------------------------
library IEEE;
use ieee.STD_LOGIC_1164.all;

library BEH_INT;
use BEH_INT.all;
use BEH_INT.IMAGE_PROCESSING.all;

library STRUC_INT;
use STRUC_INT.all;
--------------------------ENTITY TEST BENCH-----------------------------------
entity TB is
end TB;
--------------------------ARCHITECTURE BODY-----------------------------------
```

```
architecture HORIZ_TB_INT of TB is

signal RUN: STD_LOGIC;
signal CLOCK:STD_LOGIC:='0';
signal P1,P3: PIXEL:=0;
signal H: FILTER_OUT:=0;
 -------------------------COMPONENT INSTANTIATIONS-----------------------------
component CLOCK_GENERATOR1
          generic(HI_TIME,LO_TIME:TIME);
          port(RUN : in STD_LOGIC;
         CLOCK: out STD_LOGIC);
end component;

component  HORIZONTAL_FILTER1
          generic(HORIZ_DELAY: TIME;
                    WAIT_TIME:TIME);
          port(CLOCK: in  STD_LOGIC:='0';
              P1,P3: in  PIXEL:=0;
              H:     inout FILTER_OUT:=0);
end component;
--------------------CONFIGURATION SPECIFICATIONS---------------------------
for L1: CLOCK_GENERATOR1 use entity BEH_INT.CLOCK_GENERATOR(BEHAVIOR);
for L2: HORIZONTAL_FILTER1 use entity STRUC_INT.HORIZONTAL_FILTER(BEHAVIOR);
--------------------MAPPING-----------------------------------------------
begin
L1: CLOCK_GENERATOR1
generic map(HI_TIME=>75 ns,LO_TIME=>25 ns)
port map(RUN,CLOCK);

L2: HORIZONTAL_FILTER1
generic map(HORIZ_DELAY=>1 ns,WAIT_TIME=>100 ns)
port map(CLOCK,P1,P3,H);

RUN<='0' after 0 ns,'1' after 5 ns,'0' after 1000 ns;
P1<=     11 after 10 ns,10 after 110 ns,9 after 210 ns,
          12 after 310 ns,13 after 410 ns,48 after 510 ns ;

  P3<=   8 after 10 ns,40 after 110 ns,7 after 210 ns,
          10 after 310 ns,5 after 410 ns,110 after 510 ns;
end HORIZ_TB_INT;
```

Figure A-3. Source code for the test bench model of the Horizontal filter in STRUC_INT library

TEST BENCH FOR VERTICAL FILTER - INTEGER DATATYPE-STRUC_INT LIBRARY

```
-- file name: vertb_tb.vhd
-- purpose:   TEST BENCH for Vertical filter: file name (vert_b.vhd)
--              type integer
--LIBRARY     vhdlwork
 ----------------------------LIBRARY DECLARATIONS-------------------------------
library IEEE;
use ieee.STD_LOGIC_1164.all;

library BEH_INT;
use BEH_INT.all;
use BEH_INT.IMAGE_PROCESSING.all;

library STRUC_INT;
use STRUC_INT.all;
--------------------------ENTITY TEST BENCH-----------------------------------
entity TB is
end TB;
--------------------------ARCHITECTURE BODY-----------------------------------
architecture VERT_TB_INT of TB is
signal RUN: STD_LOGIC;
signal CLOCK:  STD_LOGIC:='0';
signal P1, P2, P3: PIXEL:=0;
signal V: FILTER_OUT:=0;
 -------------------------COMPONENT INSTANTIATIONS----------------------------
component CLOCK_GENERATOR1
          generic(HI_TIME,LO_TIME:TIME);
          port(RUN : in STD_LOGIC;
         CLOCK: out STD_LOGIC);
end component;

component  VERTICAL_FILTER1
          generic(VERT_DELAY,WAIT_TIME:TIME);
          port(CLOCK: in  STD_LOGIC:='0';
              P1,P2,P3:in  PIXEL:=0;
              V:inout FILTER_OUT:=0);
end component;
------------------CONFIGURATION SPECIFICATIONS-----------------------------
for L1: CLOCK_GENERATOR1 use entity BEH_INT.CLOCK_GENERATOR(BEHAVIOR);
for L2: VERTICAL_FILTER1 use entity STRUC_INT.VERTICAL_FILTER(BEHAVIOR);
--------------------MAPPING--------------------------------------------------
begin
L1: CLOCK_GENERATOR1
generic map(HI_TIME=>75 ns,LO_TIME=>25 ns)
port map(RUN,CLOCK);

L2: VERTICAL_FILTER1
generic map(VERT_DELAY=>3 ns,WAIT_TIME=>0 ns)
port map(CLOCK,P1,P2,P3,V);
RUN <='0' after 0 ns,'1' after 5 ns,'0' after 1000 ns;
```

```
P1<=    11 after 10 ns,10 after 110 ns,9 after 210 ns,
        12 after 310 ns,13 after 410 ns,48 after 510 ns;

P2<=    6 after 10 ns,  5 after 110 ns,14 after 210 ns,
        13 after 310 ns,10 after 410 ns,19 after 510 ns;
P3<=
        8 after 10 ns,40 after 110 ns,7 after 210 ns,
        10 after 310 ns,5 after 410 ns,110 after 510 ns;
end VERT_TB_INT;
```

Figure A-4. Source code for the test bench model of the Vertical filter in STRUC_INT library

TEST BENCH FOR LEFT DIAGONAL FILTER - INTEGER DATATYPE-STRUC_INT LIBRARY

```
-- file name: leftb_tb.vhd
-- purpose:   TEST BENCH for LEFT DIAGONAL FILTER, file name :left_b.vhd
              -- type integer
-- library:   vhdlwork
----------------------------LIBRARY DECLARATIONS--------------------------------
library IEEE;
use ieee.STD_LOGIC_1164.all;

library BEH_INT;
use BEH_INT.all;
use BEH_INT.IMAGE_PROCESSING.all;

library STRUC_INT;
use STRUC_INT.all;
--------------------------ENTITY TEST BENCH-------------------------------------
entity TB is
end TB;
-------------------------ARCHITECTURE BODY-------------------------------------
architecture LEFT_TB_INT of TB is
signal RUN: STD_LOGIC;
signal CLOCK:  STD_LOGIC:='0';
signal P1, P2, P3: PIXEL:=0;
signal DL: FILTER_OUT:=0;
 -------------------------COMPONENT INSTANTIATIONS-----------------------------
component CLOCK_GENERATOR1
          generic(HI_TIME,LO_TIME:TIME);
          port(RUN : in STD_LOGIC;
         CLOCK: out STD_LOGIC);
end component;

component  LEFT_DIAG_FILTER1
          generic(LEFT_DIAG_DELAY,WAIT_TIME:TIME);
          port(CLOCK: in  STD_LOGIC:='0';
              P1,P2,P3:in  PIXEL:=0;
              DL:inout FILTER_OUT:=0);
```

```
end component;
-------------------CONFIGURATION SPECIFICATIONS--------------------------
for L1: CLOCK_GENERATOR1 use entity BEH_INT.CLOCK_GENERATOR(BEHAVIOR);
for L2: LEFT_DIAG_FILTER1 use entity STRUC_INT.LEFT_DIAG_FILTER(BEHAVIOR);
--------------------MAPPING---------------------------------------------
begin
L1: CLOCK_GENERATOR1
generic map(HI_TIME=>75 ns,LO_TIME=>25 ns)
port map(RUN,CLOCK);

L2: LEFT_DIAG_FILTER1
generic map(LEFT_DIAG_DELAY=>3 ns,WAIT_TIME=>0 ns)
port map(CLOCK,P1,P2,P3,DL);

RUN<='0' after 0 ns,'1' after 5 ns,'0' after 1000 ns;

 P1<=   11 after 10 ns,10 after 110 ns,9 after 210 ns,
        12 after 310 ns,13 after 410 ns,48 after 510 ns;

P2<=    6 after 10 ns,  5 after 110 ns,14 after 210 ns,
        13 after 310 ns,10 after 410 ns,19 after 510 ns;

P3<=    8 after 10 ns,40 after 110 ns,7 after 210 ns,
        10 after 310 ns,5 after 410 ns,110 after 510 ns;
end LEFT_TB_INT;
```

Figure A-5. Source code for the test bench model of the Left diagonal filter in STRUC_INT library

TEST BENCH FOR RIGHT DIAGONAL FILTER - INTEGER DATATYPE-STRUC_INT LIBRARY

```
-- file name: rightb_tb.vhd
-- purpose:   TEST BENCH for Right diagonal filter : file name (right_b.vhd)
--              type integer
-- library:   vhdlwork
 --------------------------LIBRARY DECLARATIONS-------------------------------
library IEEE;
use ieee.STD_LOGIC_1164.all;

library BEH_INT;
use BEH_INT.IMAGE_PROCESSING.all;

library STRUC_INT;
use STRUC_INT.all;
--------------------------ENTITY TEST BENCH----------------------------------
entity TB is
end TB;
--------------------------ARCHITECTURE BODY----------------------------------
architecture RIGHT_TB_INT of TB is
signal RUN: STD_LOGIC;
signal CLOCK:  STD_LOGIC:='0';
```

```
signal P1,P2,P3: PIXEL:=0;
signal DR: FILTER_OUT:=0;
 -------------------------COMPONENT INSTANTIATIONS-------------------------
component CLOCK_GENERATOR1
          generic(HI_TIME,LO_TIME:TIME);
          port(RUN : in STD_LOGIC;
         CLOCK: out STD_LOGIC);
end component;

component  RIGHT_DIAG_FILTER1
          generic(RIGHT_DIAG_DELAY,WAIT_TIME:TIME);
          port(CLOCK: in  STD_LOGIC:='0';
              P1,P2,P3:in  PIXEL:=0;
              DR:inout FILTER_OUT:=0);
end component;
--------------------CONFIGURATION SPECIFICATIONS---------------------------
for L1: CLOCK_GENERATOR1 use entity BEH_INT.CLOCK_GENERATOR(BEHAVIOR);
for L2: RIGHT_DIAG_FILTER1 use entity STRUC_INT.RIGHT_DIAG_FILTER(BEHAVIOR);
--------------------MAPPING-----------------------------------------------
begin
L1: CLOCK_GENERATOR1
generic map(HI_TIME=>75 ns,LO_TIME=>25 ns)
port map(RUN,CLOCK);

L2: RIGHT_DIAG_FILTER1
generic map(RIGHT_DIAG_DELAY=>3 ns,WAIT_TIME=>0 ns)
port map(CLOCK,P1,P2,P3,DR);

RUN<='0' after 0 ns,'1' after 5 ns,'0' after 1000 ns;

P1<=    11 after 10 ns,10 after 110 ns,9 after 210 ns,
        12 after 310 ns,13 after 410 ns,48 after 510 ns;

P2<=    6 after 10 ns,5 after 110 ns,14 after 210 ns,
        13 after 310 ns,10 after 410 ns,19 after 510 ns;

P3<=    8 after 10 ns,40 after 110 ns,7 after 210 ns,
        10 after 310 ns,5 after 410 ns,110 after 510 ns;
end RIGHT_TB_INT;
```

Figure A-6. Source code for the test bench model of the Right diagonal filter in STRUC_INT library

TEST BENCH FOR MAGNITUDE PROCESSOR - INTEGER DATATYPE-STRUC_INT LIBRARY

```
-- file name: mag_proctb.vhd
-- purpose:   TEST BENCH for magnitude processor : file name - mag_proc.vhd
--                type integer
-- library:   vhdlwork
---------------------------LIBRARY DECLARATIONS--------------------------------
library IEEE;
use ieee.STD_LOGIC_1164.all;

library BEH_INT;
use BEH_INT.IMAGE_PROCESSING.all;

library STRUC_INT;
use STRUC_INT.all;
--------------------------ENTITY TEST BENCH-------------------------------------
entity TB is
end TB;
-------------------------ARCHITECTURE BODY-------------------------------------
architecture MAGPROC_TB_INT of TB is

signal RUN:STD_LOGIC;
signal CLOCK : STD_LOGIC:='0';
signal M_H  : FILTER_OUT;
signal M_V  : FILTER_OUT;
signal M_DL  : FILTER_OUT;
signal M_DR  : FILTER_OUT;
signal THRESHOLD : FILTER_OUT;
signal MAG_OUT: PIXEL:=0;
signal DIR :DIRECTION:="000";
-------------------------COMPONENT INSTANTIATIONS-----------------------------

component CLOCK_GENERATOR1
generic(HI_TIME,LO_TIME:TIME);
  port(  RUN : in STD_LOGIC;
        CLOCK: out STD_LOGIC);
end component;

component MAG_PROCESSOR1
generic(MAG_DELAY: TIME);
   port(CLOCK: in STD_LOGIC:='0';
           M_H,M_V,M_DL,M_DR,THRESHOLD: inout FILTER_OUT;
       MAG_OUT: inout PIXEL:= 0;
       DIR: inout DIRECTION :="000");
end component;
-------------------CONFIGURATION SPECIFICATIONS--------------------------
 for L1: CLOCK_GENERATOR1 use entity BEH_INT.CLOCK_GENERATOR(BEHAVIOR);
for L2: MAG_PROCESSOR1 use entity STRUC_INT.MAG_PROCESSOR(BEHAVIOR);
--------------------MAPPING-------------------------------------------------
begin
L1: CLOCK_GENERATOR1
```

```
generic map(HI_TIME=>75 ns,LO_TIME=>25 ns)
port map(RUN,CLOCK);

L2: MAG_PROCESSOR1
generic map(MAG_DELAY=>10 ns)
port map(CLOCK,M_H,M_V,M_DL,M_DR,THRESHOLD,MAG_OUT,DIR);

RUN<='0' after 0 ns,'1' after 5 ns,'0' after 1000 ns;
M_H<=  -7 after 10 ns,16 after 110 ns,
          9 after 210 ns,12 after 310 ns,
          13 after 410 ns,14 after 510 ns;
M_V<= 6 after 10 ns,5 after 110 ns,
          4 after 210 ns,-20 after 310 ns,
         -20 after 410 ns,10 after 510 ns;
M_DL<=  -2  after 10 ns,24 after 110 ns,
          15 after 210 ns,30 after 310 ns,
          13 after 410 ns,-19 after 510 ns;

M_DR<=9 after 10 ns,8 after 110 ns,
          7 after 210 ns,14 after 310 ns,
          5 after 410 ns,50 after 510 ns;

THRESHOLD<=0 after 10 ns;
end MAGPROC_TB_INT;
```

Figure A-7. Source code for the test bench model of the Magnitude Processor in STRUC_INT library

TEST BENCH FOR WINDOW PROCESSOR - INTEGER DATATYPE-STRUC_INT LIBRARY

```
-- file name: window_s_tb.vhd
-- purpose:   TEST BENCH for Window Processor: file name (window_s2.vhd)
--              type integer
--LIBRARY     vhdlwork
----------------------------LIBRARY DECLARATIONS--------------------------------
library IEEE;
use ieee.STD_LOGIC_1164.all;

library BEH_INT;
use BEH_INT.all;
use BEH_INT.IMAGE_PROCESSING.all;

library STRUC_INT;
use STRUC_INT.all;
--------------------------ENTITY TEST BENCH-----------------------------------
entity TB is
end TB;
--------------------------ARCHITECTURE BODY-----------------------------------
architecture WINDOW_TB of TB is
signal P1,P2,P3: PIXEL:=0;
signal CLOCK:  STD_LOGIC:='0';
```

```
signal W_H: FILTER_OUT:=0;
signal W_V: FILTER_OUT:=0;
signal W_DL: FILTER_OUT:=0;
signal W_DR: FILTER_OUT:=0;
signal RUN: STD_LOGIC:='0';
 -------------------------COMPONENT INSTANTIATIONS-----------------------------
component CLOCK_GENERATOR1
          port(RUN : in STD_LOGIC;
         CLOCK: out STD_LOGIC);
end component;

component WINDOW_PROCESSOR1
   port (CLOCK: in STD_LOGIC:='0';
          P1,P2,P3: in  PIXEL:=0;
          W_H: inout FILTER_OUT:=0;
          W_V: inout FILTER_OUT:=0;
          W_DL: inout FILTER_OUT:=0;
           W_DR: inout FILTER_OUT:=0 );
end component;
--------------------MAPPING---------------------------------------------------
begin

L1: CLOCK_GENERATOR1
port map(RUN,CLOCK);

L2:WINDOW_PROCESSOR1
port map(clock,P1,P2,P3,W_H,W_V,W_DL,W_DR);

RUN<='0' after 0 ns,'1' after 5 ns,'0' after 1000 ns;

P1<=    11 after 10 ns,10 after 110 ns,9 after 210 ns,
        12 after 310 ns,13 after 410 ns,48 after 510 ns;

P2<=    6 after 10 ns,  5 after 110 ns,14 after 210 ns,
        13 after 310 ns,10 after 410 ns,19 after 510 ns;

P3<=    8 after 10 ns,40 after 110 ns,7 after 210 ns,
        10 after 310 ns,5 after 410 ns,110 after 510 ns;
end WINDOW_TB;
```

Figure A-8. Source code for the test bench model of the Window Processor in STRUC_INT library

CONFIGURATION FOR STRUCTURAL MODEL-EDGE DETECTOR(STD_LOGIC)

```
-- file name: edgedec_struc_std.vhd
-- purpose:   configuration file for Edge detector - (with custom memory)
              --type STD_LOGIC
-- library:   vhdlwork
-------------------------------------------------------------------------------
library IEEE;
use ieee.STD_LOGIC_1164.all;
```

```
entity TB_CONFIG is
end TB_CONFIG;

architecture TEST_BENCH of TB_CONFIG is
component TEST1
end component;

begin
con: TEST1;
end TEST_BENCH;

library BEH_INT;
use BEH_INT.CLOCK_GENERATOR;

library STRUC_STD;
use STRUC_STD.IMAGE_PROCESSING.all;
use STRUC_STD.all;
use WORK.all;

configuration CONFIG_S_STD of TB_CONFIG is

for TEST_BENCH

for CON:TEST1 use entity WORK.TEST(BENCH)
generic map("test_i1.dat","test_o1.dat","test_d1.dat",10,10,8);

        for BENCH

        for P1:CLOCK_GENERATOR1 use entity BEH_INT.CLOCK_GENERATOR(BEHAVIOR)
        generic map(HI_TIME=>75 ns,LO_TIME=>25 ns);
       end for;

       for P2:COMP_MAG1 use entity WORK.COMP_MAG(COMPARE)
       generic map("test_o1.dat","test_gm.dat",10,10);
       end for;

       for P3:COMP_DIR1 use entity WORK.COMP_DIR(COMPARE)
       generic map("test_d1.dat","test_gd.dat",10,10);
       end for;

       for P4:EDGE_DETECTOR1 use entity STRUC_STD.EDGE_DETECTOR(STRUCTURE)
       generic map(NUM_ROWS=>10,NUM_COLS=>10);

       for STRUCTURE

              for MEMP1:MEMORY_PROCESSOR1
              use entity STRUC_STD.MEMORY_PROCESSOR(STRUCTURE)
              generic map(NUM_ROWS=> NUM_ROWS,NUM_COLS=> NUM_COLS,
                     MEM_OUT_DELAY=> 2 ns, N=> 4)
              port map (CLOCK,EDGE_START,INPUT,MEM_OUT1,MEM_OUT2, MEM_OUT3);
```

```
for STRUCTURE

        for ADDR:ADDR_GEN1 use
        entity STRUC_STD.ADDR_GEN(BEHAVIOR)
        generic map(ADDR_A_DELAY=>2 ns, RWB_A_DELAY=>1 ns,
                    NUM_ROWS=>NUM_ROWS, NUM_COLS=>NUM_COLS, N=> N)
        port map(START,CLOCK, X_ADDR1, X_ADDR2, X_ADDR3,Y_ADDR, RWB);
        end for;

        for MEM:MEMORY1 use entity STRUC_STD.MEMORY(BEHAVIOR)
        generic map(MEM_OUT_DELAY=>MEM_OUT_DELAY,
                    NUM_COLS=> NUM_COLS, N=>N)
        port map(CLOCK,RWB,MEM_IN,X_ADDR1, X_ADDR2, X_ADDR3,
              Y_ADDR, MEM_OUT1, MEM_OUT2, MEM_OUT3);
        end for;
end for;
end for;

for WINP: WINDOW_PROCESSOR1 use entity
STRUC_STD.WINDOW_PROCESSOR(STRUCTURE)
generic map(HORIZ_DELAY=> 3 ns, VERT_DELAY=> 3 ns,LEFT_DIAG_DELAY => 3 ns,
            RIGHT_DIAG_DELAY=> 3 ns, WAIT_TIME => 0 ns)
port map(CLOCK,MEM_OUT1,MEM_OUT2,MEM_OUT3,E_H,E_V,E_DL,E_DR);

 for STRUCTURE

        for HP:HORIZONTAL_FILTER1 use entity
        STRUC_STD.HORIZONTAL_FILTER(BEHAVIOR)
        generic map(HORIZ_DELAY => HORIZ_DELAY,WAIT_TIME => WAIT_TIME)
        port map (CLOCK=>CLOCK,P1_H=>P1, P3_H=>P3, H=>W_H);
        end for;

        for VP:VERTICAL_FILTER1 use entity
        STRUC_STD.VERTICAL_FILTER(BEHAVIOR)
        generic map(VERT_DELAY =>  VERT_DELAY,WAIT_TIME => WAIT_TIME)
        port map (CLOCK=>CLOCK,P1_V=>P1,P2_V=>P2,P3_V=>P3,V=>W_V);
        end for;

        for LDP:LEFT_DIAG_FILTER1 use entity
        STRUC_STD.LEFT_DIAG_FILTER(BEHAVIOR)
        generic map(LEFT_DIAG_DELAY => LEFT_DIAG_DELAY,
                    WAIT_TIME => WAIT_TIME)
        port map (CLOCK=>CLOCK,P1_L=>P1,P2_L=>P2,P3_L=>P3,DL=>W_DL);
        end for;

        for RDP:RIGHT_DIAG_FILTER1 use entity
        STRUC_STD.RIGHT_DIAG_FILTER(BEHAVIOR)
        generic map(RIGHT_DIAG_DELAY => RIGHT_DIAG_DELAY,
                   WAIT_TIME => WAIT_TIME)
        port map (CLOCK=>CLOCK,P1_R=>P1,P2_R=>P2,P3_R=>P3,DR=>W_DR);
        end for;
end for;
```

```
                end for;

                for MAGP: MAG_PROCESSOR1 use entity
                STRUC_STD.MAG_PROCESSOR(BEHAVIOR)
                generic map(MAG_DELAY => 3 ns)
                port map (CLOCK,E_H,E_V,E_DL,E_DR,THRESHOLD,OUTPUT,DIR);
                end for;
        end for;
        end for;
        end for;
end for;
end for;
end ;
```

Figure A-9. Source code for configuration for Edge detector-Structural model (STD_LOGIC data type)

Test Results for the Test plan in Table 3.1

1. Test goals 1.3,1.4 - Left and right diagonal edges detected and given appropriate direction.

Input image **Output image**

Output image shows that the left and right diagonal edges have been detected correctly. The directions given to the left and right edges are “111” and “101” respectively.

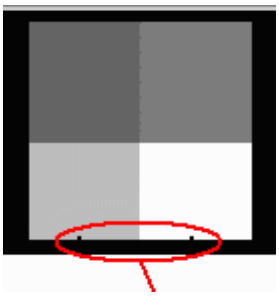
2. Test goal 1.5 - Corners detected and given appropriate direction

Input image **Output image**

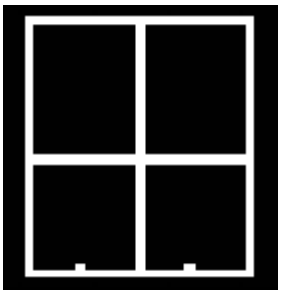
Output image shows that corners have been detected correctly. The directions given to the four corners are as follows:

1) Top left : “111” 2) Top right : “101” 3) Bottom left : “001” 4)Bottom right : “011”.

3. Test goal 1.6 - Single pixel “holes” in edges are filled.



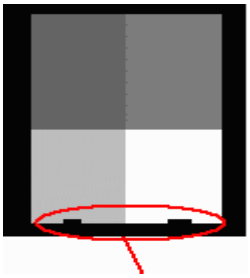
Input image



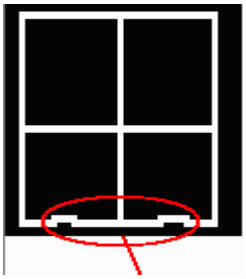
Output image

Output image shows that the single pixel holes circled in the input image have been filled.

4. Test goal 1.7 - Multi-pixel “holes” in edges remain unfilled.



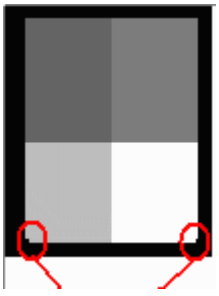
Input image



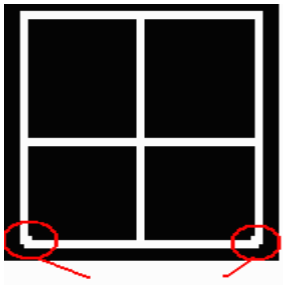
Output image

Output image shows that multi-pixel holes circled in the input image remain unfilled.

5. Test goal 1.8 - Single pixel corner “holes” are filled.



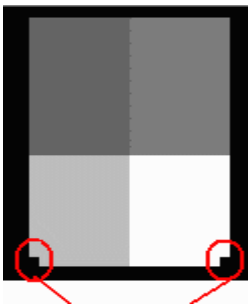
Input image



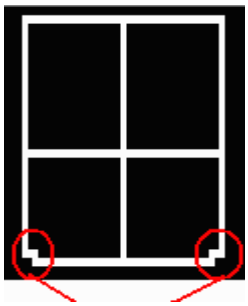
Output image

Output image shows that single pixel corner “holes” circled in the input image are filled.

6. **Test goal 1.9 - Multi-pixel corner “holes” remain unfilled.**



Input image



Output image

Output image shows that multi-pixel corner “holes” circled in the input image remain unfilled**.**

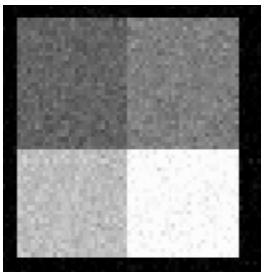
7. **Test goal 1.10 - Single pixel spots are ignored.**

Input image

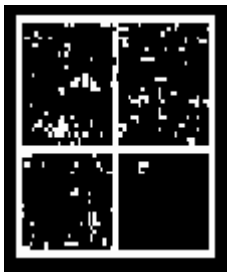
Output image

Output image shows that single pixel spots are ignored.

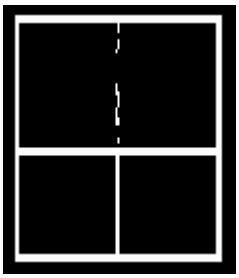
8. **Test goal 1.11 - Appropriate handling of rough edges**



Input image



Output image for for threshold=80



Output image for threshold=130

Output image shows that for threshold=80, all the edges have been detected, but the noise is not totally filtered, but at threshold=130, all the noise has been filtered, but also one of the edges is filtered.

9. Test goal 1.12 - Statement coverage test

Coverage value for the Sobel edge detector was verified with the Synopsys simulator which indicated a 100% coverage value.

Appendix B : FILES RELATED TO MEMORY and DIAGNOSTIC TESTING

This appendix contains the source code of the test bench model used for testing the memory model and faulty architectures of the models used for Diagnostic testing.

TEST BENCH MODEL USED FOR TESTING THE CUSTOM MEMORY PROCESSOR

```
-- file name: memsys_std_tb.vhd
-- purpose:   TEST BENCH for CUSTOM MEMORY TESTING, file name :mem_sys_std2.vhd
                --type STD_LOGIC
-- library:   vhdlwork
--------------------------------------------------------------------------------
library IEEE;
use ieee.STD_LOGIC_1164.all;

library STRUC_STD;
use STRUC_STD.IMAGE_PROCESSING.all;
use STD.TEXTIO.all;
use work.all;
------------------------entity declaration-------------------------------------
entity TEST is
generic(IN_FILE,GOLD_FILE,ERR_FILE:STRING(1 to 11);
      MEM_OUT_DELAY:TIME;
      NUM_ROWS,NUM_COLS,N:NATURAL);
end TEST;
----------------------architecture description--------------------------------
architecture BENCH of TEST is
type FRAME_IN_IMAGE is array(1 to NUM_ROWS,1 to NUM_COLS) of INTEGER;
type FRAME_OUT_IMAGE is array(1 to 3,1 to NUM_COLS) of INTEGER;
type ERROR_ARRAY is array(1 to 3,1 to NUM_COLS) of STD_LOGIC;
-------------------Declaration of signals-----------------------------------
signal RUN         : STD_LOGIC;
signal CLOCK,RWB: STD_LOGIC;
signal MEM_IN   :PIXEL:="00000000";
signal X_ADDR1,X_ADDR2,X_ADDR3: STD_LOGIC_VECTOR(1 downto 0):="01";
signal Y_ADDR:STD_LOGIC_VECTOR(N-1 downto 0):=INT_TO_STDLOGICN(1,N);
signal MEM_OUT1,MEM_OUT2,MEM_OUT3:PIXEL:="00000000";
signal INPUT:PIXEL:="00000000";
signal RD_START,START: STD_LOGIC:='0';
---------------Component Instantiations-----------------------------------------
component CLOCK_GENERATOR1
       port(  RUN : in STD_LOGIC;
       CLOCK: out STD_LOGIC);
 end component ;

component MEMORY_PROCESSOR1
  port(CLOCK: in STD_LOGIC:='0';           -- system CLOCK
    START: in STD_LOGIC:='0';    -- start signal
    MEM_IN: in PIXEL:="00000000";      -- INPUT PIXEL
    MEM_OUT1, MEM_OUT2, MEM_OUT3: out PIXEL:="00000000"); -- memory unit output
end component;

begin
----------------------Port Mapping-----------------------------------------------
P1 : CLOCK_GENERATOR1
port map(RUN,CLOCK);
```

```
P2 :MEMORY_PROCESSOR1
port map(CLOCK,RD_START,INPUT,MEM_OUT1,MEM_OUT2,MEM_OUT3);

START<='0' after 0 ns,'1' after 5 ns,'0' after 105 ns;
RUN<=transport '1' after 0 ns,'0' after 5000 ns;
----------Processes----------
MEM_1: process

variable VLINE1,VLINE2,OUTLINE:LINE;
variable BUSY1,BUSY2,BUSY3,BUSY4 :STD_LOGIC:='0';
variable INPUT_IMAGE,GOLD_MEM_IMAGE:FRAME_IN_IMAGE;
variable OUTPUT_IMAGE:FRAME_OUT_IMAGE;
variable ERR_ARRAY:ERROR_ARRAY;
variable Z,A,B     :INTEGER;
variable I,J   :INTEGER:=1;
variable T1,T2,T3,T4 : INTEGER;
variable X,Y:NATURAL:=1;
variable COUNT:INTEGER:=0;
variable COUNT2:INTEGER:=0;
variable TEMP1:STRING(1 to 13):="MEMORY CELL: ";
variable TEMP2:STRING(1 to 1):=",";
variable TEMP3:STRING(1 to 5):="ERROR";
variable TEMP4:STRING(1 to 4):="PASS";
variable ERR_Y:INTEGER:=1;
file IMAGEIN: TEXT is in IN_FILE;
file GOLD_MEMORY: TEXT is in GOLD_FILE;
file ERROR_FILE: TEXT is out ERR_FILE;

begin
        wait until rising_edge(CLOCK);
        -- set the internal BUSY signal
        if START = '1' then
                COUNT:=0;
                COUNT2:=0;
                X:=1;
                Y:=1;
                I:=1;
                J:=1;
                ERR_Y:=1;
                BUSY1 :='1';
        end if;
 if BUSY1='1' then
-- load the image data file to the memory array B
         for i in 1 to NUM_ROWS loop
        readline(IMAGEIN,VLINE1);
        readline(GOLD_MEMORY,VLINE2);
                for j in 1 to NUM_COLS loop
        read(VLINE1,Z);
        read(VLINE2,B);
                INPUT_IMAGE(i,j):= Z;
                GOLD_MEM_IMAGE(i,j):=B;
```

```
                    end loop;
            end loop;
            BUSY1:='0';
            BUSY2:='1';
-- assert message after loading all the data
        assert (false) report "array read in";
 end if;
if BUSY2='1' then
                    COUNT:=COUNT+1;
                    INPUT <= INT_TO_STDLOGIC8(INPUT_IMAGE(i,j));
                    if (I=1) and (J=1) then
                            RD_START<='1';
                    else
                            RD_START<='0';
                    end if;
                    if (I=NUM_ROWS) and (J=NUM_COLS) then
                     I:=I;
                     J:=J;
                    elsif J=NUM_COLS then
                            J:=1;
                            I:=I+1;
                    else
                            J:=J+1;
                    end if;
end if;
-- COMPARE THE MEMORY OUTPUTS WITH THE GOLD MODEL AND RECORD THE ERRORS--
      if COUNT>=(2*NUM_COLS + 3) and COUNT < ((NUM_ROWS*NUM_COLS)-1) then
             BUSY3:='1';
      end if;
if BUSY3='1' then
            OUTPUT_IMAGE(1,ERR_Y):= STDLOGIC_TO_INT(MEM_OUT1);
            OUTPUT_IMAGE(2,ERR_Y):= STDLOGIC_TO_INT(MEM_OUT2);
            OUTPUT_IMAGE(3,ERR_Y):= STDLOGIC_TO_INT(MEM_OUT3);

             if OUTPUT_IMAGE(1,ERR_Y)/=
            GOLD_MEM_IMAGE(1+ COUNT2 ,ERR_Y) then
              --report mem_cell( 1, ERR_Y) fail;
            ERR_ARRAY( COUNT2 mod 3 +1,ERR_Y):= '1';
             end if;


             if OUTPUT_IMAGE(2,ERR_Y)
             /= GOLD_MEM_IMAGE(2+COUNT2,ERR_Y) then
              --report mem_cell(COUNT2 mod 3 +1,ERR_Y) fail;
                    ERR_ARRAY( (COUNT2+1) mod 3 +1,ERR_Y):= '1';
            end if;
             if OUTPUT_IMAGE(3,ERR_Y)
             /= GOLD_MEM_IMAGE(3+COUNT2,ERR_Y) then
              --report mem_cell(1,ERR_Y) fail;
                    ERR_ARRAY((COUNT2+2) mod 3 +1,ERR_Y):= '1';
             end if;
          ERR_Y:=ERR_Y+1;
```

```
        if ERR_Y=NUM_COLS+1 then
                    COUNT2:=COUNT2+1;
                    ERR_Y:=1;
        end if;
 if COUNT2=NUM_ROWS-3 then
             BUSY3:='0';
        BUSY4:='1';
        end if;
end if;
--WRITE THE STATUS OF EACH CELL IN AN ERROR REPORT --
if BUSY4='1' then
            for i in 1 to 3 loop
             for j in 1 to NUM_COLS loop
                    if ERR_ARRAY(i,j)='1' then
                            T1:=i;
                            T2:=j;
                            write(OUTLINE,TEMP1);
                            write(OUTLINE,T1);
                            write(OUTLINE,TEMP2);
                            write(OUTLINE,T2);
                            write(OUTLINE,' ');
                            write(OUTLINE,TEMP3);
                            writeline(ERROR_FILE,OUTLINE);
                    else
                            T3:=i;
                            T4:=j;
                            write(OUTLINE,TEMP1);
                            write(OUTLINE,T3);
                            write(OUTLINE,TEMP2);
                            write(OUTLINE,T4);
                            write(OUTLINE,' ');
                            write(OUTLINE,TEMP4);
                            writeline(ERROR_FILE,OUTLINE);
                    end if;
         end loop;
        end loop;
 -- assert message after loading all the data
        assert (false) report "memory outputs compared with gold file";
             BUSY4:='0';
end if;
end process MEM_1;
end bench;
```

Figure B- 1 Test bench model for testing Memory Processor

FAULTY MEMORY MODULE

```
-- file name: MEM1.vhd ( WITH STUCK-AT-FAULTS EMBEDDED IN THE MODEL)
-- purpose: define the Memory of the edge detector system
-- Memory size: three times the number of columns (STRUC_INT library)
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

library STRUC_STD;
use STRUC_STD.IMAGE_PROCESSING.all;

---------- interface declaration ----------
entity MEMORY is
  generic(MEM_OUT_DELAY: TIME; -- Memory output delay
        NUM_COLS: NATURAL;   -- number of columns of the input image file
            N: INTEGER);         -- length of the Y_ADDRess vector
  port ( CLOCK: in STD_LOGIC:='0';  -- system CLOCK
            RWB: in STD_LOGIC:='0';     -- READ/(NOT)WRITE signal
       MEM_IN : in PIXEL:="00000000";  -- input PIXEL
       X_ADDR1: in STD_LOGIC_VECTOR(1 downto 0):="01";     -- X address 1
       X_ADDR2: in STD_LOGIC_VECTOR(1 downto 0):="01";     -- X address 2
       X_ADDR3: in STD_LOGIC_VECTOR(1 downto 0):="01";     -- X address 3
       Y_ADDR:  in STD_LOGIC_VECTOR(N-1 downto 0):=INT_TO_STDLOGICN(1,N);
                                                  -- Y address
       MEM_OUT1: out PIXEL:="00000000";           -- Memory unit output 1
       MEM_out2: out PIXEL:="00000000";           -- Memory unit output 2
       MEM_out3: out PIXEL:="00000000");          -- Memory unit output 3
end MEMORY;
---------- behavior description ----------
architecture STUCK_ZERO_ONE of MEMORY is
type MEMORY_ARRAY is array(1 to 3, 1 to NUM_COLS) of PIXEL;
begin
 process
   variable MEM:MEMORY_ARRAY;
   variable FAULT_CELL:PIXEL:="00000000";
begin
---------- WRITE operation ----------
           wait until rising_edge(CLOCK);

           if RWB='0' then
-------------------------------------------------------------------------
--INTRODUCING STUCK_AT_ZERO FAULT--
           if X_ADDR1="11" and Y_ADDR=INT_TO_STDLOGICN(2,N) then
                    FAULT_CELL:= MEM_IN;
           FAULT_CELL(1):='0';
                    MEM(STDLOGIC_TO_INT(X_ADDR1),STDLOGIC_TO_INT(Y_ADDR)):= FAULT_CELL;

      elsif
                X_ADDR1="01" and Y_ADDR=INT_TO_STDLOGICN(3,N) then
                    FAULT_CELL:= MEM_IN;
           FAULT_CELL(2):='0';
                    MEM(STDLOGIC_TO_INT(X_ADDR1),STDLOGIC_TO_INT(Y_ADDR)):= FAULT_CELL;
           elsif
                X_ADDR1="01" and Y_ADDR=INT_TO_STDLOGICN(14,N) then
                    FAULT_CELL:= MEM_IN;
           FAULT_CELL(2):='0';
```

```
                    MEM(STDLOGIC_TO_INT(X_ADDR1),STDLOGIC_TO_INT(Y_ADDR)):= FAULT_CELL;
-------------------------------------------------------------------------------
--INTRODUCING STUCK_AT_ONE FAULT--

      elsif
                    X_ADDR1="11" and Y_ADDR=INT_TO_STDLOGICN(3,N) then
                    FAULT_CELL:= MEM_IN;
          FAULT_CELL(2):='1';
                    MEM(STDLOGIC_TO_INT(X_ADDR1),STDLOGIC_TO_INT(Y_ADDR)):= FAULT_CELL;

      elsif
                    X_ADDR1="10" and Y_ADDR=INT_TO_STDLOGICN(4,N) then
                    FAULT_CELL:= MEM_IN;
          FAULT_CELL(0):='1';
                    MEM(STDLOGIC_TO_INT(X_ADDR1),STDLOGIC_TO_INT(Y_ADDR)):= FAULT_CELL;
          elsif
                    X_ADDR1="01" and Y_ADDR=INT_TO _STDLOGICN(15,N) then
                    FAULT_CELL:= MEM_IN;
          FAULT_CELL(0):='1';
                    MEM(STDLOGIC_TO_INT(X_ADDR1),STDLOGIC_TO_INT(Y_ADDR)):= FAULT_CELL;
-------------------------------------------------------------------------------
      else

                    MEM(STDLOGIC_TO_INT(X_ADDR1),STDLOGIC_TO_INT(Y_ADDR)) := MEM_IN;

          end if;
          end if;
---------- READ operation -------------------------------------------------
          wait until falling_edge(CLOCK);
          if RWB='1' then
          MEM_OUT1 <= MEM(STDLOGIC_TO_INT(X_ADDR1),STDLOGIC_TO_INT(Y_ADDR))
                              after MEM_OUT_DELAY;
          MEM_OUT2 <= MEM(STDLOGIC_TO_INT(X_ADDR2),STDLOGIC_TO_INT(Y_ADDR))
                              after MEM_OUT_DELAY;
          MEM_OUT3 <= MEM(STDLOGIC_TO_INT(X_ADDR3),STDLOGIC_TO_INT(Y_ADDR))
                              after MEM_OUT_DELAY;
     end if;
end process;
end STUCK_ZERO_ONE;
```

Figure B-2 Faulty memory module

CONFIGURATION FILE FOR FAULTY HORIZONTAL FILTER

```
-- file name: config_h_rtl_f.vhd (configuration for FAULTY Horizontal filter)
-- purpose:   horizontal filtering
-- library:   STRUC_RTL library

library STRUC_RTL;
use STRUC_RTL.all;
```

```
configuration H_RTL_F of HORIZONTAL_FILTER is
        for STRUCTURE
                ------------------ A <= "0000" & P1 ----------------
                for EXT8_12A: EXT8_12 use entity STRUC_RTL.EXT8_12(BEHAVIOR)
                        port map(P1,A);
                end for;
                ------------------ B <= "0000" & P3 ----------------
                for EXT8_12B: EXT8_12 use entity STRUC_RTL.EXT8_12(BEHAVIOR)
                        port map(P3,B);
                end for;
                ------------------ S1 <= A - B ----------------
                for Diff1:  SUM12_PM use entity STRUC_RTL.SUM12_PM(FAULT)
                        port map(A,B,S1);
                end for;
                ------------------ S2 <= S1 ----------------
                for DELAY1: REG12 use entity STRUC_RTL.REG12(FAULT)
                        generic map(HORIZ_DELAY)
                        port map( S1, S2, CLOCK );
                end for;
                ------------------ S3 <= S2 ----------------
                for DELAY2: REG12 use entity STRUC_RTL.REG12(FAULT)
                        generic map(HORIZ_DELAY)
                        port map( S2, S3, CLOCK );
                end for;
                ------------------ S4 <= S3 ----------------
                for DELAY3: REG12 use entity STRUC_RTL.REG12(FAULT)
                        generic map(HORIZ_DELAY)
                        port map( S3, S4, CLOCK );
                end for;
                ------------------ S5 <= S4 + S2 ----------------
                for SUM1:   SUM12_PP use entity STRUC_RTL.SUM12_PP(FAULT)
                        port map( S4, S2, S5 );
                end for;
                ------------------ S6 <= S5 ----------------
                for DELAY4: REG12 use entity STRUC_RTL.REG12(FAULT)
                        generic map(HORIZ_DELAY)
                        port map( S5, S6, CLOCK );
                end for;
                ------------------ C <= S4(10 downto 0) & '0' ----------------
                for MULT_2_12_12A: MULT_2_12_12
                        use entity STRUC_RTL.MULT_2_12_12(BEHAVIOR)
                        port map(S4,C);
                end for;
                ------------------ S7 <= C + S6 ----------------
                for SUM2:   SUM12_PP use entity STRUC_RTL.SUM12_PP(FAULT)
                        port map(C, S6, S7);
                end for;
                ------------------ H <= S7 ----------------
                for DELAY5: REG12 use entity STRUC_RTL.REG12(FAULT)
                        generic map(HORIZ_DELAY)
                        port map( S7, H, CLOCK );
                end for;
```

```
        end for;
        end;
```

Figure B- 3 Configuration file for Faulty Horizontal filter

CONFIGURATION FILE FOR FAULTY VERTICAL FILTER

```
-- file name: config_v_rtl_f.vhd (configuration for FAULTY Vertical filter)
-- purpose:   vertical filtering
-- library:   STRUC_RTL library
library STRUC_RTL;
use STRUC_RTL.all;
configuration V_RTL_F of VERTICAL_FILTER is
        for STRUCTURE
                ------------------- A <= "0000" & P1 ------------------
                for EXT8_12A: EXT8_12 use entity STRUC_RTL.EXT8_12(BEHAVIOR)
                        port map(P1,A);
                end for;
                ------------------- B <= "0000" & P3 ------------------
                for EXT8_12B: EXT8_12 use entity STRUC_RTL.EXT8_12(BEHAVIOR)
                        port map(P3,B);
                end for;
                ------------------- S1 <= A + B ------------------------
                for SUM1:   SUM12_PP use entity STRUC_RTL.SUM12_PP(PARTS)
                        port map( A, B, S1 );
                end for;
                ------------------ S2 <= S1 ---------------------------
                for HOLD_A: REG12 use entity STRUC_RTL.REG12(PARTS)
                        generic map(VERT_DELAY)
                        port map( S1, S2, CLOCK );
                end for;
                ------------------ HB <= P2 ---------------------------
                for HOLD_B: REG8 use entity STRUC_RTL.REG8(PARTS)
                        generic map(VERT_DELAY)
                        port map( P2, HB, CLOCK );
                end for;

                ------------- C <= ("000" & HB(7 downto 0) & '0'); -------
                for MULT_2_8_12A: MULT_2_8_12
                        use entity STRUC_RTL.MULT_2_8_12(BEHAVIOR)
                        port map(HB,C);
                end for;
                ------------- S3 <= S2 + C -----------------------------
                for SUM2:   SUM12_PP use entity STRUC_RTL.SUM12_PP(PARTS)
                        port map( S2, C, S3 );
                end for;
                ------------- S4 <= S3 ---------------------------
                for DELAY1: REG12 use entity STRUC_RTL.REG12(PARTS)
                        generic map(VERT_DELAY)
                        port map( S3, S4, CLOCK );
                end for;
```

```
        ------------ S5 <= S4 ----------------------------
        for DELAY2: REG12 use entity STRUC_RTL.REG12(PARTS)
                generic map(VERT_DELAY)
                port map( S4, S5, CLOCK );
        end for;
        ------------ S6 <= S5 ----------------------------
        for DELAY3: REG12 use entity STRUC_RTL.REG12(PARTS)
                generic map(VERT_DELAY)
                port map( S5, S6, CLOCK );
        end for;
        ------------ S7 <= S4 - S6 ----------------------------
        for DIFF1:  SUM12_PM use entity STRUC_RTL.SUM12_PM(PARTS)
                port map( S4, S6, S7 );
        end for;
        ------------ V <= S7 ----------------------------
        for DELAY4: REG12 use entity STRUC_RTL.REG12(PARTS)
                generic map(VERT_DELAY)
                port map( S7, V, CLOCK );
        end for;
end for;
end;
```

Figure B-4 Configuration file for Faulty Vertical filter

Appendix C : WAVES FILES

This appendix contains the source code for the following files :

- Waveform Generation Procedure (WGP) package file
- Waves Test bench file for the Sobel Edge detector - STD_LOGIC data type.
- Program to convert a two-dimensional image file into the external file suitable for WAVES.

Waveform Generation Procedure Package file

```
--
-- ******** This File Was Automatically Generated ********
-- ********      By The WAVES96-VHDL Tool Set     ********
-- ******** Generated for VHDL entity:            ********
-- ********      EDGE_DETECTOR                    ********
-- ******** Generation date and time:             ********
-- ********      Wed Jul 16 16:20:36 1997         ********
--
use STD.TEXTIO.all;
library IEEE;
use IEEE.WAVES_1164_Frames.all;
use IEEE.WAVES_1164_Declarations.all;
use IEEE.WAVES_Interface.all;
use WORK.WAVES_Objects.all;
use WORK.UUT_Test_Pins.all;

package WGP_EDGE_DETECTOR is
            procedure WAVEFORM( signal WPL : inout WAVES_PORT_LIST );
end WGP_EDGE_DETECTOR;
-------------------------------------------------------
package body WGP_EDGE_DETECTOR is
           procedure WAVEFORM( signal WPL : inout WAVES_PORT_LIST ) is
    -- Standard Waveform Generator
    --
    -- Declare external file
    --
    file VECTOR_FILE : text is in "test1.txt";
    --
    -- NOTE:  use following declaration for
    --        '93 compliant VHDL simulators
    --
    --  file VECTOR_FILE : text open READ_MODE is "vectors.txt";
    --
    -- Declare file slice variable
    --
    variable VECTOR : FILE_SLICE := NEW_FILE_SLICE;
    --
    -- Pinset declaration section
    --
    --
    -- Pinsets Generated from vectors
    --
    constant INPUT : PINSET := INPUT_7 + INPUT_6 + INPUT_5 +
        INPUT_4 + INPUT_3 + INPUT_2 + INPUT_1 + INPUT_0;

    constant THRESHOLD : PINSET := THRESHOLD_11 + THRESHOLD_10 +
        THRESHOLD_9 + THRESHOLD_8 + THRESHOLD_7 +
        THRESHOLD_6 + THRESHOLD_5 + THRESHOLD_4 +
        THRESHOLD_3 + THRESHOLD_2 + THRESHOLD_1 +
```

```
    THRESHOLD_0;

constant OUTPUT : PINSET := OUTPUT_7 + OUTPUT_6 + OUTPUT_5 +
    OUTPUT_4 + OUTPUT_3 + OUTPUT_2 + OUTPUT_1 +
    OUTPUT_0;

constant DIR : PINSET := DIR_2 + DIR_1 + DIR_0;
--
-- Input vectors pinset
--
constant INPUTS : PINSET := INPUT + THRESHOLD;
--
-- The following time set declaration(s) must be modified to
-- reflect the correct timing requirements for the test and
-- verification needs of your specific design.
--
-- Each time set contains one timing line for each pinset
-- and individual pin that was generated from the ports on
-- the VHDL entity.
--
-- This file is only a template to get you started.
--
--
-- Declare the frame sets
--
variable WTL : WAVE_TIMING_LIST( 1 to 2 ) := (
--
-- Frame Set 1
--
( PERIOD => 100 ns,
  FSA    => BUILD_FRAME_DATA(
            (
              (+CLOCK, PULSE_HIGH(25 ns,75 ns)),
              (+EDGE_START, NON_RETURN(0 ns )),
              (OUTPUT, NON_RETURN(0 ns )),
              (DIR, NON_RETURN(0 ns )),
              (INPUTS, NON_RETURN(0 ns))
            )
           )
),
--
-- Frame Set 2
--
( PERIOD => 100 ns,
  FSA    => BUILD_FRAME_DATA(
            (
              (+CLOCK, PULSE_HIGH(25 ns,75 ns)),
              (+EDGE_START, NON_RETURN(0 ns )),
              (OUTPUT, WINDOW(50 ns,100 ns )),
              (DIR, WINDOW(50 ns,100 ns )),
              (INPUTS, NON_RETURN(0 ns))
            )
```

```
            )
    ) );
 begin -- waveform generator procedure

       loop
        READ_FILE_SLICE( VECTOR_FILE, VECTOR );
         exit when VECTOR.END_OF_FILE;
         APPLY( WPL, VECTOR.CODES.all, WTL( VECTOR.FS_INTEGER ).FSA );
         DELAY( WTL( VECTOR.FS_INTEGER ).PERIOD );
       end loop;

end WAVEFORM;
END WGP_EDGE_DETECTOR;
```

Figure C-1. Waveform Generation Procedure package file

WAVES TEST BENCH FILE

```
-- ******** This File Was Automatically Generated  ********
-- ******** By The WAVES96-VHDL Tool Set      ********
-- ******** Generated for Entity: EDGE_DETECTOR
-- ******** This File Was Generated on: Mon May 19 11:19:28 1997
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.WAVES_1164_utilities.all;
USE ieee.WAVES_INTERface.all;

USE WORK.UUT_TEST_pins.all;
USE work.waves_objects.all;
USE work.WGP_EDGE_DETECTOR.all;
USE STD.TEXTIO.all;

library STRUC_STD;
use STRUC_STD.IMAGE_PROCESSING.all;
use STRUC_STD.all;

-- Include component libary references here
-- User Must Modify And ADD component libary references here
-- Include component libary references here

ENTITY TEST_bench IS
generic(OUT_FILE         :STRING(1 to 11);          -- OUTPUT file for storing magnitude Outputs
        DIR_FILE         :STRING(1 to 11);          -- OUTPUT file which stores Direction Outputs
        MAG_GOLD_FILE :STRING(1 to 11);             -- OUTPUT file which stores magnitude
                                                    -- Outputs of the GOLD model
        DIR_GOLD_FILE :STRING(1 to 11);             -- OUTPUT file which stores Direction
                                                    -- OUTPUT of the GOLD model
        NUM_ROWS      :NATURAL;                     -- number of rows in the INPUT IMAGE
        NUM_COLS      :NATURAL;                     -- number of columns in the INPUT IMAGE
        WAIT_CYCLES   :NATURAL);       -- time required before Outputs are
                                       -- written into frame buffers.
```

```
END TEST_bench;

ARCHITECTURE EDGE_DETECTOR_TEST OF TEST_bench IS
type FRAME_IMAGE is array(1 to NUM_ROWS,1 to NUM_COLS) of INTEGER;
type FRAME_DIRECTION is array(1 to NUM_ROWS,1 to NUM_COLS) of BIT_VECTOR(2 downto 0);
 --*********************************************************
 --***********CONFIGURATION SPECIFICATION ***************
 --*********************************************************
   COMPONENT EDGE_DETECTOR
   PORT ( CLOCK            : IN   std_logic;
        EDGE_START         : IN   std_logic;
        INPUT              : IN   std_logic_vector(  7 downto  0 );
        THRESHOLD          : IN   std_logic_vector( 11 downto  0 );
        OUTPUT           : INOUT std_logic_vector(  7 downto  0 );
        DIR              : INOUT std_logic_vector(  2 downto  0 ));
   END COMPONENT;
-- Modify entity use statement
 -- User Must Modify modify and declare correct
 --  .. Architecture, Library, Component ..
 -- Modify entity use statement
--FOR ALL:EDGE_DETECTOR USE ENTITY work.EDGE_DETECTOR(STRUCTURE);
 --************************************************************
 -- stimulus signals for the waveforms mapped into UUT INPUTS
 --************************************************************
   SIGNAL WAV_STIM_CLOCK              :std_logic;
   SIGNAL WAV_STIM_EDGE_START          :std_logic;
   SIGNAL WAV_STIM_INPUT              :std_logic_vector(  7 downto  0 );
   SIGNAL WAV_STIM_THRESHOLD          :std_logic_vector( 11 downto  0 );
   SIGNAL RD_START                    :STD_LOGIC:='0';
   SIGNAL INTER                       :STD_LOGIC_VECTOR(7 downto 0):="00000000";
    --*****************************************************
 -- Expected signals used in monitoring the UUT OUTPUTS
 --*****************************************************
   SIGNAL FAIL_SIGNAL          :STD_LOGIC;
   SIGNAL WAV_EXPECT_OUTPUT    :std_ulogic_VECTOR(7 downto 0);
   SIGNAL WAV_EXPECT_DIR       :std_ulogic_VECTOR(2 downto 0);
 --*****************************************************
 -- UUT Output signals used In Monitoring ACTUAL Values
 --*****************************************************
 --************************************************************
 -- bi-directional signals used  for stimulus signals mapped
 -- into UUT INPUTS and also monitoring the UUT OUTPUTS
 --************************************************************
   SIGNAL BI_DIREC_OUTPUT             :std_logic_vector(  7 downto  0 );
   SIGNAL BI_DIREC_DIR                 :std_logic_vector(  2 downto  0 );
--*******************************************************************
 -- WAVES signals Outputting each slice of the waves port list
--*******************************************************************
      SIGNAL wpl  : WAVES_port_list;
BEGIN
  --*****************************************************************
 -- process that generates the WAVES waveform
```

```
--*****************************************************************
   WAVES: waveform(wpl);
--*****************************************************************
-- processes that assign the WPL values to Testbench signals
--*****************************************************************
WAV_STIM_CLOCK          <= wpl.signals( 1 );
WAV_STIM_EDGE_START     <= wpl.signals( 2 );
WAV_STIM_INPUT          <= To_StdLogicVector(wpl.signals( 3 to 10 ));
WAV_STIM_THRESHOLD      <= To_StdLogicVector(wpl.signals( 11 to 22 ));
BI_DIREC_OUTPUT         <= To_StdLogicVector(wpl.signals( 23 to 30 ))
                    when wpl.direction( 23 ) = STIMULUS
                    else "ZZZZZZZZ";
WAV_EXPECT_OUTPUT       <= wpl.signals( 23 to 30 )
                    when wpl.direction( 23 ) = RESPONSE
                    else "--------";
BI_DIREC_DIR            <= To_StdLogicVector(wpl.signals( 31 to 33 ))
                    when wpl.direction( 31 ) = STIMULUS
                    else "ZZZ";
WAV_EXPECT_DIR          <= wpl.signals( 31 to 33 )
                    when wpl.direction( 31 ) = RESPONSE
                    else "---";
--*********************************************
-- UUT Port Map - Name Symantics Denote Usage
--*********************************************
u1: EDGE_DETECTOR
PORT MAP(       CLOCK               => WAV_STIM_CLOCK,
                EDGE_START          => RD_START,
                INPUT               => INTER,
                THRESHOLD           => WAV_STIM_THRESHOLD,
                OUTPUT              => BI_DIREC_OUTPUT,
                DIR                 => BI_DIREC_DIR);
 --*********************************************************
-- Monitor Processes To Verify The UUT Operational Response
--*********************************************************
MONITOR_OUTPUT:process
 variable VLINE1,VLINE2 :LINE;                -- for reading writing from the INPUT and OUTPUT text files
variable BUSY: INTEGER range 0 to 3;      --internal variable to trigger different parts of the TEST bench
variable OUTPUT_MAG_IMAGE : FRAME_IMAGE; -- frame buffer to store magnitude Outputs
variable GOLD_MAG_IMAGE :FRAME_IMAGE;-- frame buffer to store the magnitude values of the GOLD model
variable GOLD_DIR_IMAGE : FRAME_DIRECTION; -- frame buffer to store the Direction values of the GOLD
model
variable OUTPUT_DIR_IMAGE : FRAME_DIRECTION; -- frame buffer to store the Direction Outputs
variable A,B : INTEGER;  -- used for reading and writing the INPUT and OUTPUT text files.
variable FLAG1:STD_LOGIC:='0';         -- set high when there is an error in the magnitude Outputs
variable FLAG2:STD_LOGIC:='0'          -- set high when there is an error in the Direction Outputs
variable I,J:NATURAL:=1;              -- used for indexing the frame buffers
variable X,Y:NATURAL:=2;               -- used for indexing the frame buffers
variable COUNT:INTEGER:=0;            -- used to specify when the Outputs can be written into the buffers
-------------------------------------------------------------------------------
--FILES--
file IMAGEOUT  :TEXT is out OUT_FILE;               -- OUTPUT file for storing magnitude Outputs
file DIROUT:TEXT is out DIR_FILE;                   -- OUTPUT file which stores Direction Outputs
```

```
file GOLD_MAGNITUDE:TEXT is in MAG_GOLD_FILE;-- OUTPUT file which stores magnitude
                                                -- Outputs of the GOLD model
file GOLD_DIRECTION:TEXT is in DIR_GOLD_FILE;   -- OUTPUT file which stores Direction
                                                -- OUTPUT of the GOLD model-type INTEGER
 begin
 --write data out to a file
   wait until rising_edge(WAV_STIM_CLOCK);
     COUNT:=COUNT+1;
          INTER<=WAV_STIM_INPUT;
                    if (I=1) and (J=1) then
                              RD_START<='1';
                   else
                  RD_START<='0';
                    end if;
          if (I=NUM_ROWS) and (J=NUM_COLS) then
                              i:=1;
                              j:=1;
                    elsif j=NUM_COLS then
                              j:=1;
                              I:=I+1;
                    else
                              J:=J+1;
                    end if;

                    if COUNT>=(2*NUM_COLS+WAIT_CYCLES) then
                    if not(Y=1) and not(Y=NUM_COLS)  then
                    OUTPUT_MAG_IMAGE(X,Y):= STDLOGIC_TO_INT(BI_DIREC_OUTPUT);
                    OUTPUT_DIR_IMAGE(X,Y) := STDLOGIC_TO_BIT(BI_DIREC_DIR);
                    end if;
                    Y:=Y+1;
                    if Y=NUM_COLS+1 then
                    Y:=1;
                    X:=X+1;
                    end if;

          if (X=NUM_ROWS-1) and Y=NUM_COLS then
                    BUSY:=1;
                    end if;
                    end if;
case BUSY is

when 1=>
        for i in 1 to NUM_COLS loop
                    OUTPUT_MAG_IMAGE(1,i):=0;
                    OUTPUT_DIR_IMAGE(1,i):="000";
                    OUTPUT_MAG_IMAGE(NUM_ROWS,i):=0;
                    OUTPUT_DIR_IMAGE(NUM_ROWS,i):="000";
            end loop;

            for i in 2 to NUM_ROWS loop
                    OUTPUT_MAG_IMAGE(i,1):=0;
                    OUTPUT_DIR_IMAGE(i,1):="000";
```

```
                    OUTPUT_MAG_IMAGE(i,NUM_COLS):=0;
                    OUTPUT_DIR_IMAGE(i,NUM_COLS):=”000”;
            end loop;

            for i in 1 to NUM_ROWS loop
          for j in 1 to NUM_COLS loop
              write(VLINE1,OUTPUT_MAG_IMAGE(i,j));
             write(VLINE1,' ');
                    write(VLINE2,OUTPUT_DIR_IMAGE(i,j));
                    write(VLINE2,' ');
           end loop;
           writeline(IMAGEOUT,VLINE1);
          writeline(DIROUT,VLINE2);
        end loop;
        -- assert message after loading all the data
        assert (false) report "MAGNITUDE Outputs written to file";

           -- assert message after loading all the data
         assert (false) report "DIRECTION Outputs written to file";
            BUSY:=2;

       I:=1;
       J:=1;
       X:=2;
       Y:=2;
       COUNT:=0;

when 2=>

--Compare the magnitude and Direction Outputs
--with the Outputs of the "GOLD MODEL--
              for i in 1 to NUM_ROWS loop
           readline(GOLD_MAGNITUDE,VLINE1);
                readline(GOLD_DIRECTION,VLINE2);
             for j in 1 to NUM_COLS loop
                  read(VLINE1,A);
                 read(VLINE2,B);
             GOLD_MAG_IMAGE(i,j):=A;
             GOLD_DIR_IMAGE(i,j):=B;
             if FLAG1='0' then
               if OUTPUT_MAG_IMAGE(i,j) /= GOLD_MAG_IMAGE(i,j) then
                --set FLAG high--
               FLAG1:='1';
               end if;
             end if;
             if FLAG2 ='0' then
                if OUTPUT_DIR_IMAGE(i,j) /= GOLD_DIR_IMAGE(i,j) then
                FLAG2:='1';
                end if;
           end if;
  end loop;
end loop;
```

```
BUSY:=3;

if FLAG1='1' then
-- assert message after comparing all the data
        assert (false) report "MAGNITUDE VALUES DO NOT MATCH -- FAIL";
          else
        assert (false) report "MAGNITUDE VALUES MATCH -- PASS";
end if;
if FLAG2='1' then
 -- assert message after comparing all the data
        assert (false) report "DIRECTION VALUES DO NOT MATCH -- FAIL";
          else
        assert (false) report "DIRECTION VALUES MATCH -- PASS";
        end if;
when others => null;
end case;
END PROCESS;
END EDGE_DETECTOR_TEST;

library IEEE;
use ieee.STD_LOGIC_1164.all;
use work.all;

entity TB_CONFIG is
end TB_CONFIG;

architecture TEST_BENCH of TB_CONFIG is

component TEST1
end component;

begin
con: TEST1;
end TEST_BENCH;

library STRUC_STD;
use STRUC_STD.IMAGE_PROCESSING.all;
use STRUC_STD.all;

library BEH_INT;
use BEH_INT.all;

configuration CFG_TEST_BENCH_EDGE_DETECTOR_TEST of TB_CONFIG is
for TEST_BENCH

 for con:TEST1 use entity work.TEST_BENCH(EDGE_DETECTOR_TEST)
 generic map("test_o1.dat","test_d1.dat","test_gm.dat","test_gd.dat",10,10,8);

   for EDGE_DETECTOR_TEST

            for u1:EDGE_DETECTOR use entity STRUC_STD.EDGE_DETECTOR(STRUCTURE)
            generic map(NUM_ROWS=>10,NUM_COLS=>10);
```

```
for STRUCTURE

for MEMP1:MEMORY_PROCESSOR1
          use entity STRUC_STD.MEMORY_PROCESSOR(STRUCTURE)
 generic map(NUM_ROWS=> NUM_ROWS,NUM_COLS=> NUM_COLS,
                      MEM_OUT_DELAY=> 2 ns, N=> 4)
  port map (CLOCK,EDGE_START,INPUT,MEM_OUT1,MEM_OUT2, MEM_OUT3);

 for STRUCTURE

                    for ADDR:ADDR_GEN1 use
                      entity STRUC_STD.ADDR_GEN(BEHAVIOR)
                      generic map(ADDR_A_DELAY=>2 ns, RWB_A_DELAY=>1 ns,
                              NUM_ROWS=>NUM_ROWS,
                                  NUM_COLS=>NUM_COLS, N=> N)
                      port map(START,CLOCK, X_ADDR1, X_ADDR2, X_ADDR3,
                            Y_ADDR, RWB);
                    end for;

                    for MEM:MEMORY1 use entity STRUC_STD.MEMORY(BEHAVIOR)
                      generic map(MEM_OUT_DELAY=>MEM_OUT_DELAY,
                                  NUM_COLS=> NUM_COLS, N=>N)
                     port map(CLOCK,RWB,MEM_IN,X_ADDR1, X_ADDR2, X_ADDR3,
                            Y_ADDR, MEM_OUT1, MEM_OUT2, MEM_OUT3);
                    end for;
    end for;
 end for;
for WINP: WINDOW_PROCESSOR1 use entity
                      STRUC_STD.WINDOW_PROCESSOR(STRUCTURE)
                      generic map(HORIZ_DELAY=> 3 ns, VERT_DELAY=> 3 ns,
                                  LEFT_DIAG_DELAY => 3 ns,
                                  RIGHT_DIAG_DELAY=> 3 ns,
                                  WAIT_TIME => 0 ns)
            port map(CLOCK,MEM_OUT1,MEM_OUT2,MEM_OUT3,
                            E_H,E_V,E_DL,E_DR);

  for STRUCTURE

                    for HP  :  HORIZONTAL_FILTER1 use entity
                    STRUC_STD.HORIZONTAL_FILTER(BEHAVIOR)
                    generic map(HORIZ_DELAY => HORIZ_DELAY,WAIT_TIME => WAIT_TIME)
                    port map (CLOCK,P1, P3, W_H);
                    end for;

                    for VP: VERTICAL_FILTER1 use entity
                    STRUC_STD.VERTICAL_FILTER(BEHAVIOR)
                    generic map(VERT_DELAY =>  VERT_DELAY,WAIT_TIME => WAIT_TIME)
                    port map (CLOCK,P1,P2,P3,W_V);
                    end for;

                    for LDP: LEFT_DIAG_FILTER1 use entity
```

```
                        STRUC_STD.LEFT_DIAG_FILTER(BEHAVIOR)
                        generic map(LEFT_DIAG_DELAY => LEFT_DIAG_DELAY,
                         WAIT_TIME => WAIT_TIME)
                          port map (CLOCK,P1,P2,P3,W_DL);
                        end for;

                        for RDP: RIGHT_DIAG_FILTER1 use entity
                              STRUC_STD.RIGHT_DIAG_FILTER(BEHAVIOR)
                          generic map(RIGHT_DIAG_DELAY => RIGHT_DIAG_DELAY,
                                      WAIT_TIME => WAIT_TIME)
                          port map (CLOCK,P1,P2,P3,W_DR);
                        end for;
            end for;
           end for;

         for MAGP: MAG_PROCESSOR1 use entity STRUC_STD.MAG_PROCESSOR(BEHAVIOR)
         generic map(MAG_DELAY => 3 ns)
         port map (CLOCK,E_H,E_V,E_DL,E_DR,THRESHOLD,OUTPUT,DIR);
         end for;
        end for;
end for;
end for
end for;
end for;
end ;
```

Figure C- 2 Waves Test Bench file for Sobel Edge detector -Structural model -STD_LOGIC data type

PROGRAM FOR CONVERTING A TWO_DIMENSIONAL IMAGE INTO EXTERNAL FILE

```
library IEEE;
use ieee.STD_LOGIC_1164.all;
use STD.TEXTIO.all;

entity CONVERT is
generic(NUM_ROWS,NUM_COLS:NATURAL);
port(     CLOCK:in STD_LOGIC;
          START:in STD_LOGIC;
          I:in INTEGER;
          O:out BIT_VECTOR(7 downto 0));
end CONVERT;

architecture CON of CONVERT is
type IN_IMAGE is array(1 to NUM_ROWS,1 to NUM_COLS) of INTEGER;
type OUT_IMAGE is array(1 to NUM_ROWS,1 to NUM_COLS) of BIT_VECTOR(7 downto 0);
--*****************************************************************************
--INTEGER TO BIN( 7 downto 0) function
--*****************************************************************************
function INT_TO_BIN(A: INTEGER) return BIT_VECTOR is
          variable RESULT : BIT_VECTOR(0 to 7);
           variable TEMP_A : INTEGER:=0;
```

```
        variable TEMP_B : INTEGER:=0;
        begin
                TEMP_A := A;
                for i in 7 downto 0 loop
                TEMP_B:=TEMP_A/(2**i);
                TEMP_A:=TEMP_A rem(2**i);
                        if(TEMP_B = 1) then
                        RESULT(7-i):='1';
                        else
                        RESULT(7-i):='0';
                        end if;
                end loop;
                return RESULT;
end INT_TO_BIN;
begin

CONVERTION:process
variable VLINE1:LINE;
variable Z     : INTEGER;
variable INPUT_IMAGE:IN_IMAGE;
variable OUTPUT_IMAGE:OUT_IMAGE;
variable BUSY :INTEGER range 0 to 2;
variable TEMP1: STRING(1 to 3):="1 1";
variable TEMP2: STRING(1 to 12):="000000110010";
variable TEMP3: STRING(1 to 12):="-------- ---";
variable TEMP4:STRING(1 to 5):=" : 1;";
file IMAGEIN : TEXT is in "test_i1.dat";
file IMAGEOUT : TEXT is out "testg.dat";

begin
        wait until rising_edge(CLOCK);
         -- set the internal busy signal
        if START = '1' then
         busy :=1;
        end if;
 case BUSY is

when 1=>
         for i in 1 to NUM_ROWS loop
        readline(IMAGEIN,VLINE1);
                 for j in 1 to NUM_COLS loop
                 read(VLINE1,Z);
                INPUT_IMAGE(i,j):= Z;
                OUTPUT_IMAGE(i,j):=INT_TO_BIN(INPUT_IMAGE(i,j));
                 end loop;
        end loop;
        busy:=2;
 -- assert message after loading all the data
   assert (false) report "array read in";
when 2=>

for i in 1 to NUM_ROWS loop
```

```
            for j in 1 to NUM_COLS loop
            write(VLINE1,TEMP1);
            write(VLINE1,' ');
            write(VLINE1,OUTPUT_IMAGE(i,j));
            write(VLINE1,' ');
             write(VLINE1,TEMP2);
            write(VLINE1,' ');
            write(VLINE1,TEMP3);
            write(VLINE1,' ');
             write(VLINE1,TEMP4);
            writeline(IMAGEOUT,VLINE1);
            end loop;
end loop;
-- assert message after loading all the data
        assert (false) report "magnitude outputs written to file";
busy:=0;
when others=> null;
end case;
end process CONVERTION;
end CON;
```

Figure C-3. Source code for the program to convert a two-dimensional image to an external file suitable for WAVES

VITA

Sucharita Gopalakrishnan was born on February 2nd, 1972 in Coimbatore, India. She got her Bachelor's degree in Electronics and Communications engineering from Mangalore University, India. She joined graduate school at the Virginia Polytechnic Institute and State University in August, 1995 and received her Master's Degree in Electrical Engineering in August, 1997.