VHDL
File based Simulation

©Hanan Ribo

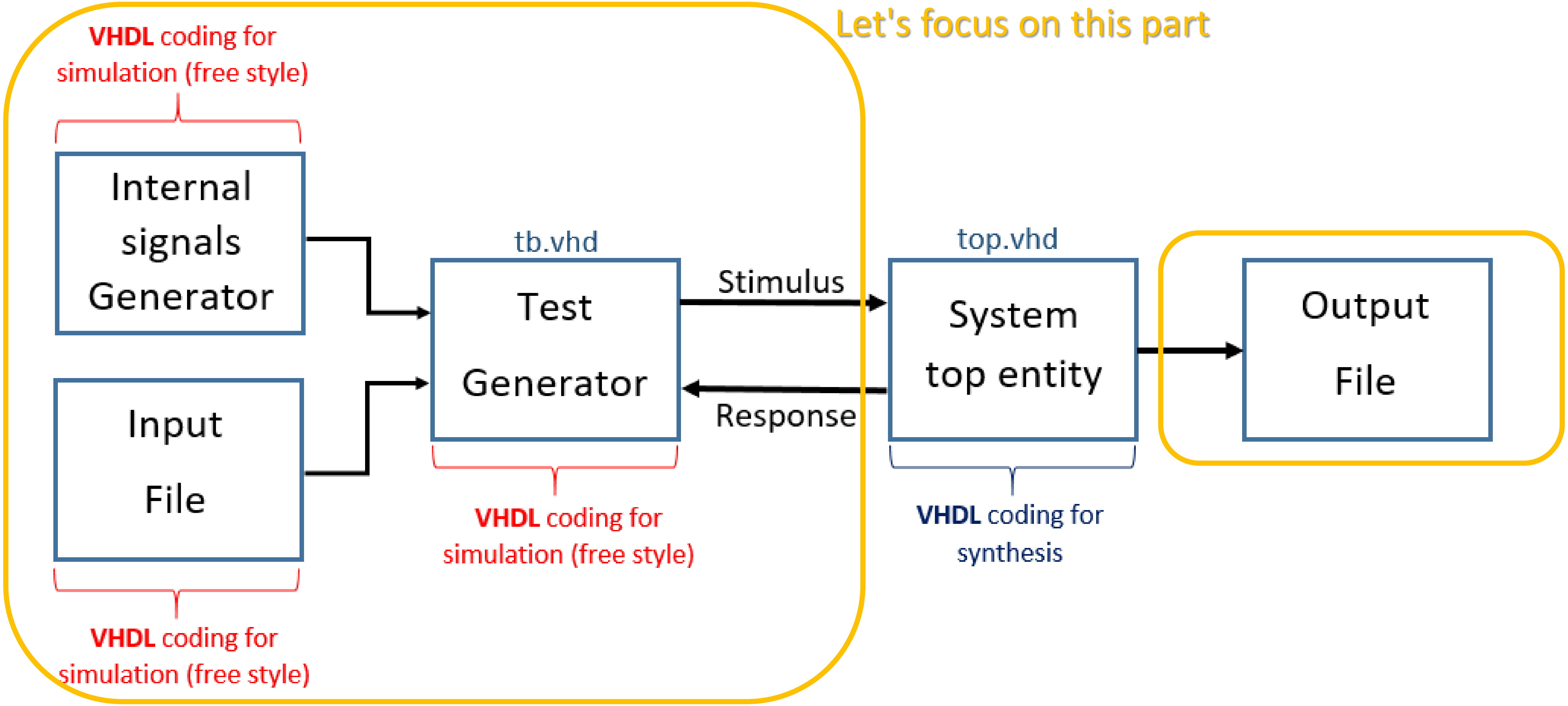
File based Simulation

- A file type provides access to objects containing a sequence of values of a given type.

- File types are typically used to access files in the host system environment for read, write and append operations, the value of a file object is the sequence of values contained in the physical file.

- Using files we can enhance the simulation and validation options and stimulate and examine our design in different ways.

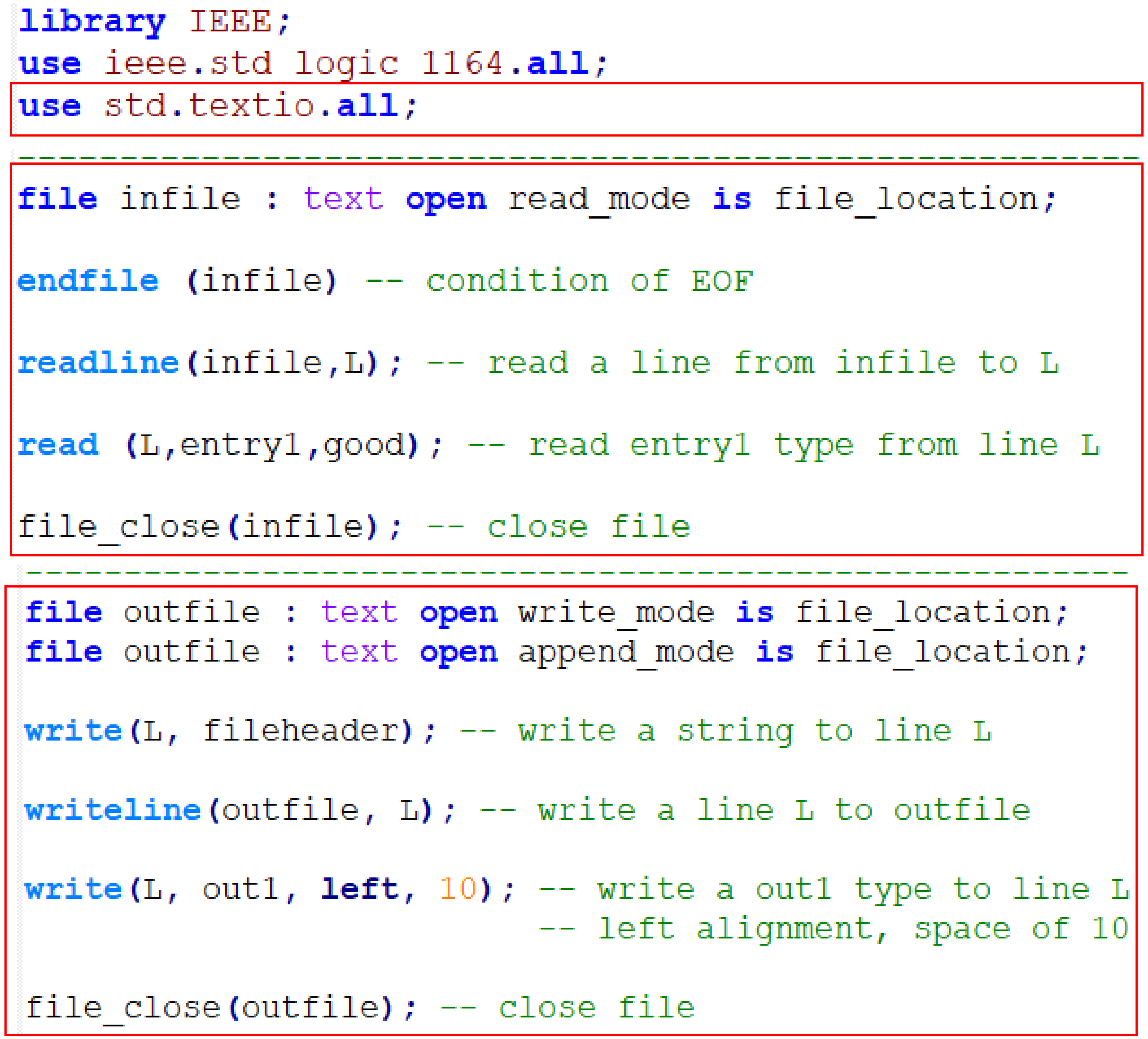
Read and **Write** File Test Bench Architecture



Let's focus on this part
VHDL coding for simulation (free style)
Internal signals Generator
Input File
VHDL coding for simulation (free style)
tb.vhd
Test Generator
VHDL coding for simulation (free style)
Stimulus
Response
top.vhd
System top entity
VHDL coding for synthesis
Output File

©Hanan Ribo

Read and **Write** using TextIO Library procedures



```
library IEEE;
use ieee.std_logic_1164.all;
use std.textio.all;
-------------------------------------------------------------------
```

Read

```
file infile : text open read_mode is file_location;

endfile (infile) -- condition of EOF

readline(infile,L); -- read a line from infile to L

read (L,entry1,good); -- read entry1 type from line L

file_close(infile); -- close file
-------------------------------------------------------------------
```

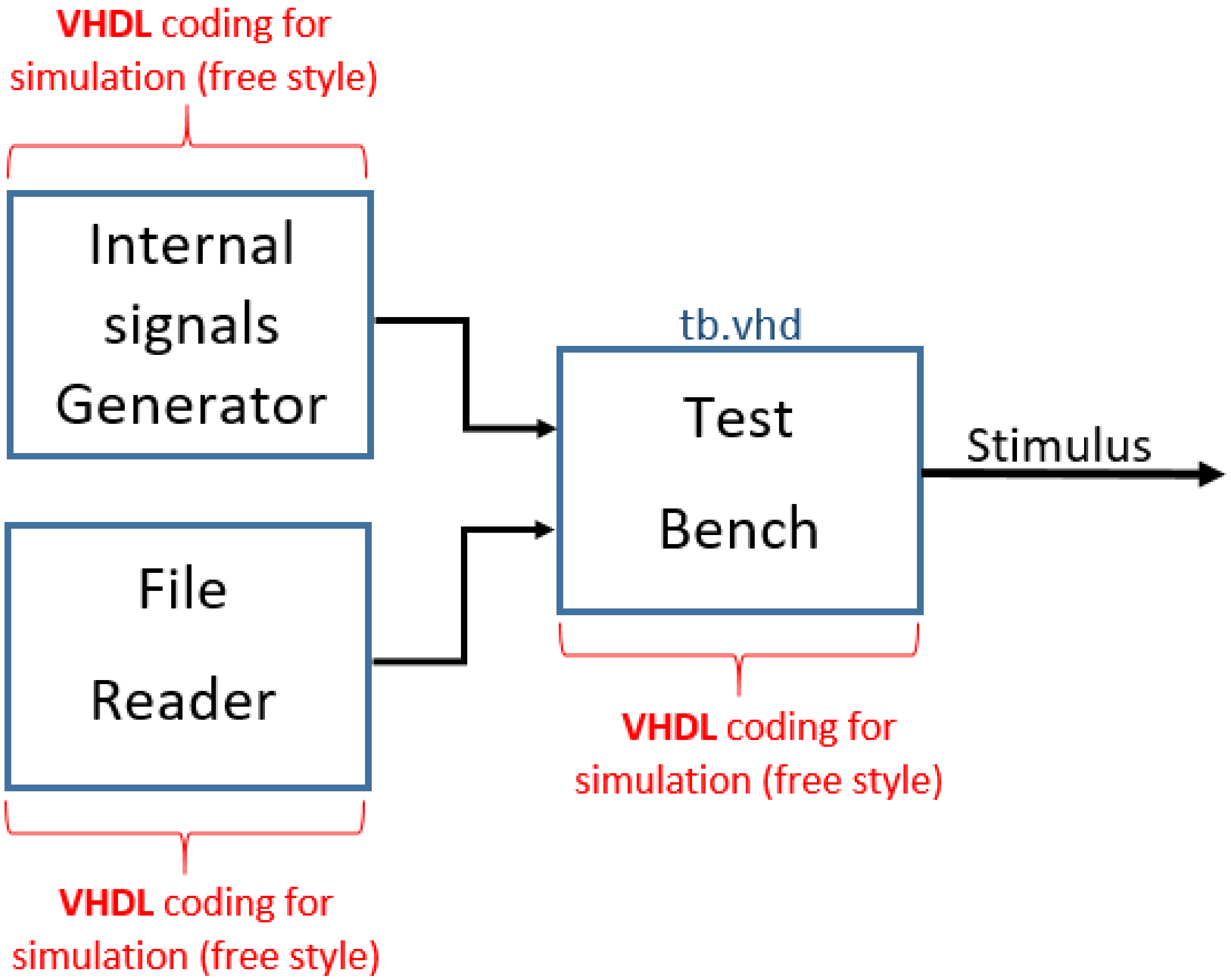
Write

```
file outfile : text open write_mode is file_location;
file outfile : text open append_mode is file_location;

write(L, fileheader); -- write a string to line L

writeline(outfile, L); -- write a line L to outfile

write(L, out1, left, 10); -- write a out1 type to line L
                          -- left alignment, space of 10

file_close(outfile); -- close file
```

Read and **Write** using TextIO Library procedures

- Read from File in VHDL using TextIO Library
- Write to File in VHDL using TextIO Library

Test bench stimulus using **file reading**


VHDL coding for simulation (free style)
Internal signals Generator
File Reader
VHDL coding for simulation (free style)
tb.vhd
Test Bench
Stimulus
VHDL coding for simulation (free style)

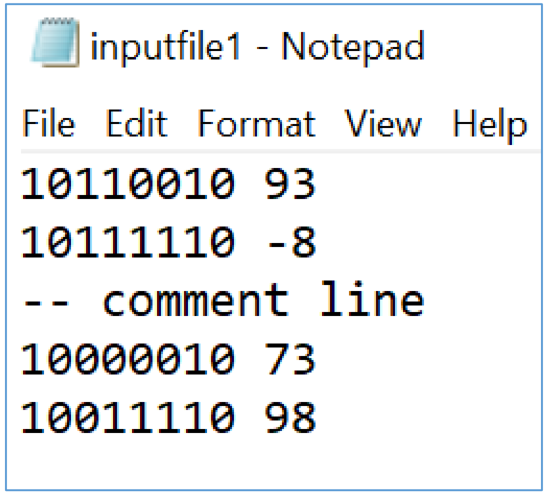
Test bench stimulus using **file reading** – Example1

```
library IEEE;
use ieee.std_logic_1164.all;
use std.textio.all;
---------------------------------------------
entity filegen is
    port(
        stimsig1: out bit_vector(7 downto 0);
        stimsig2: out integer
    );
end filegen;
---------------------------------------------
architecture rtl of filegen is
    signal gen : boolean := true;
    signal done : boolean := false;
    constant file_location : string(1 to 39) :=
    "C:\TestPrograms\ModelSim\inputfile1.txt";
begin
```

Trigger signals used by file read operations

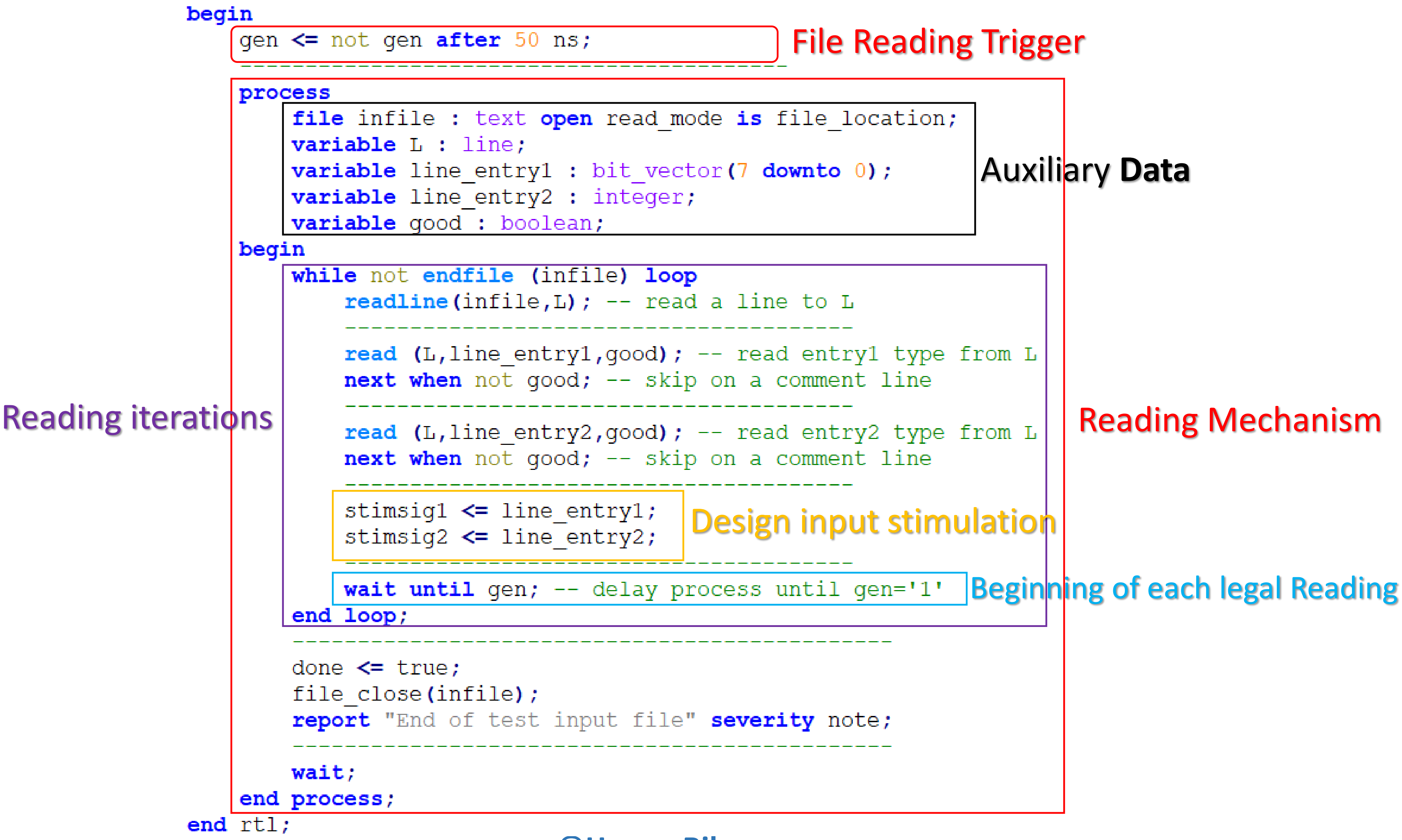
File location strings

External sampling time generator



```
10110010 93
10111110 -8
-- comment line
10000010 73
10011110 98
```

Test bench stimulus using **file reading** – Example1



```
begin
  gen <= not gen after 50 ns;
  ------------------------------------------
  process
      file infile : text open read_mode is file_location;
      variable L : line;
      variable line_entry1 : bit_vector(7 downto 0);
      variable line_entry2 : integer;
      variable good : boolean;
  begin
      while not endfile (infile) loop
          readline(infile,L); -- read a line to L
          ---------------------------------------
          read (L,line_entry1,good); -- read entry1 type from L
          next when not good; -- skip on a comment line
          ---------------------------------------
          read (L,line_entry2,good); -- read entry2 type from L
          next when not good; -- skip on a comment line
          ---------------------------------------
          stimsig1 <= line_entry1;
          stimsig2 <= line_entry2;
          ---------------------------------------
          wait until gen; -- delay process until gen='1'
      end loop;
      ----------------------------------------------
      done <= true;
      file_close(infile);
      report "End of test input file" severity note;
      ----------------------------------------------
      wait;
  end process;
end rtl;
```

File Reading Trigger

Auxiliary **Data**

Reading iterations

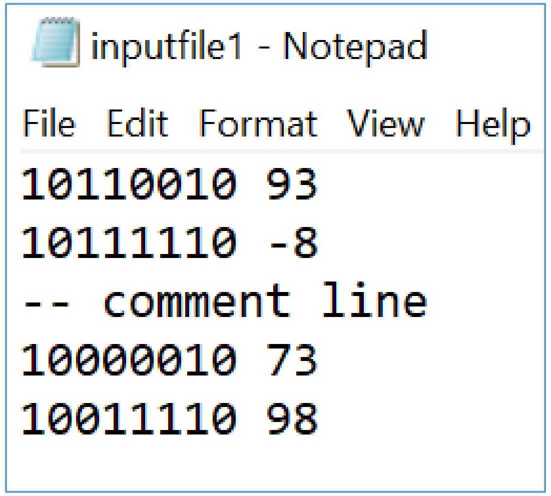
Reading Mechanism

Design input stimulation

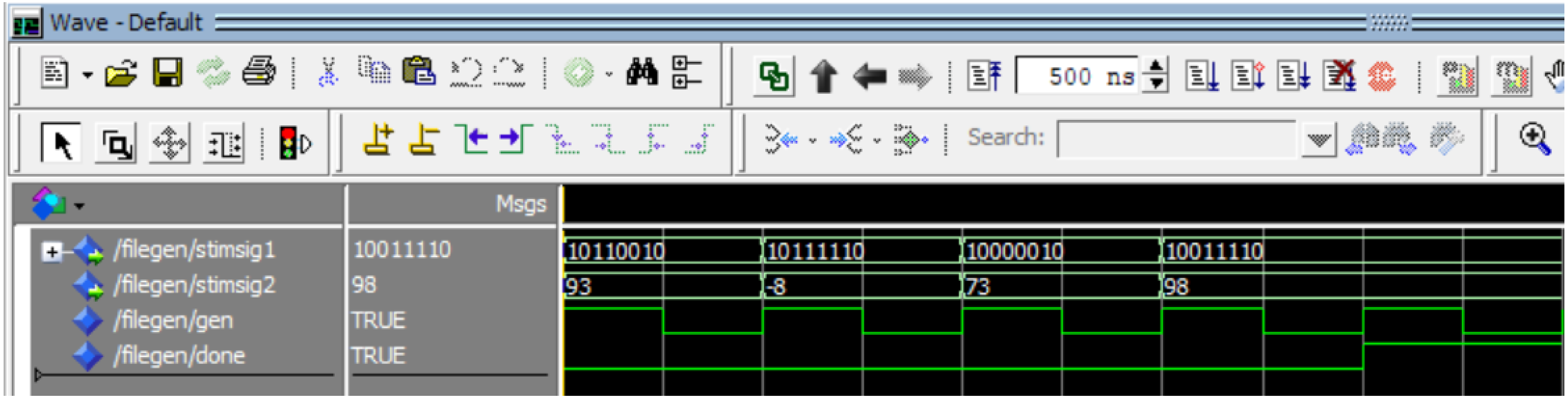
Beginning of each legal Reading

©Hanan Ribo

Test bench stimulus using **file reading** – Example1

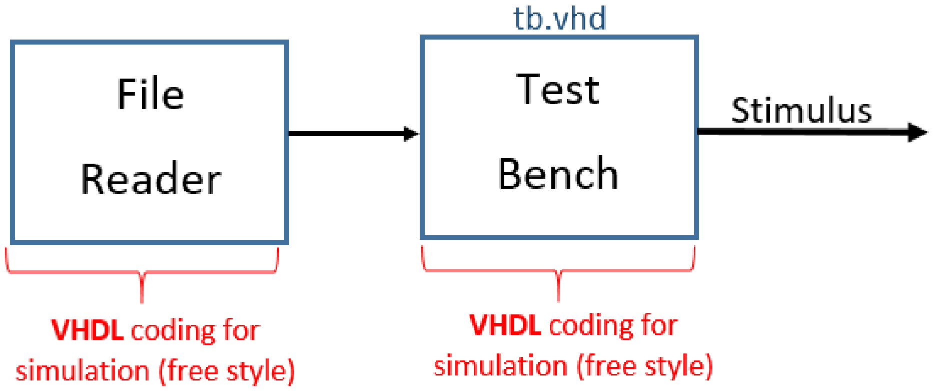


```
10110010 93
10111110 -8
-- comment line
10000010 73
10011110 98
```



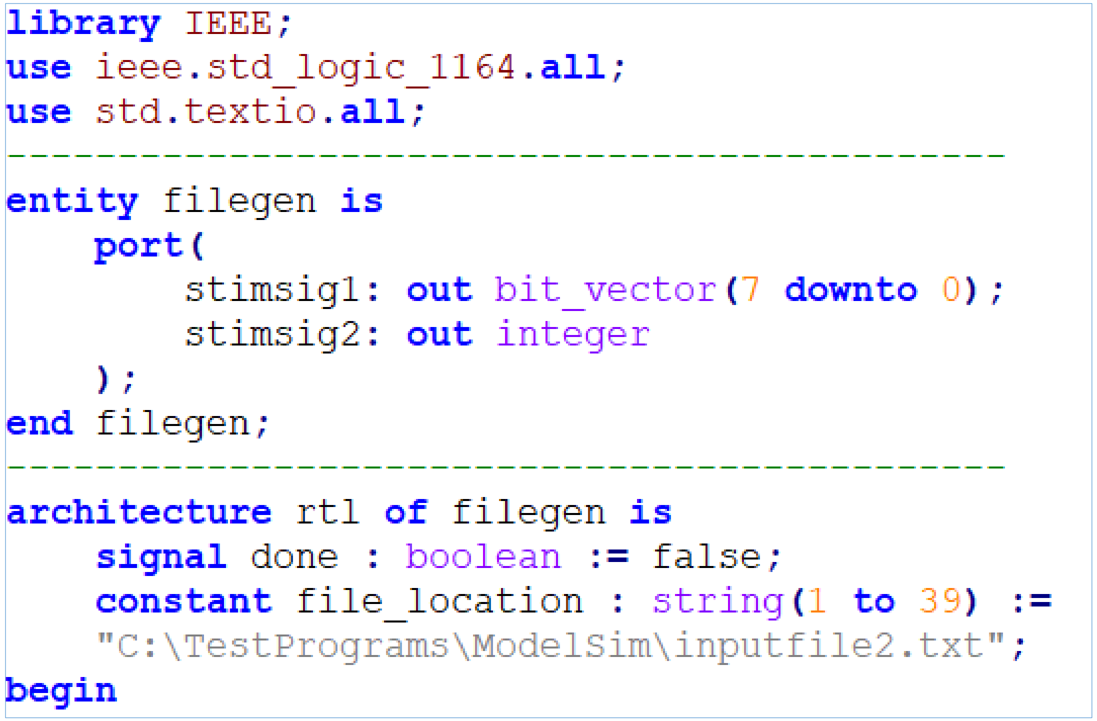
```
/filegen/stimsig1   10011110
/filegen/stimsig2   98
/filegen/gen        TRUE
/filegen/done       TRUE
```

Test bench stimulus using **file reading** – Example2



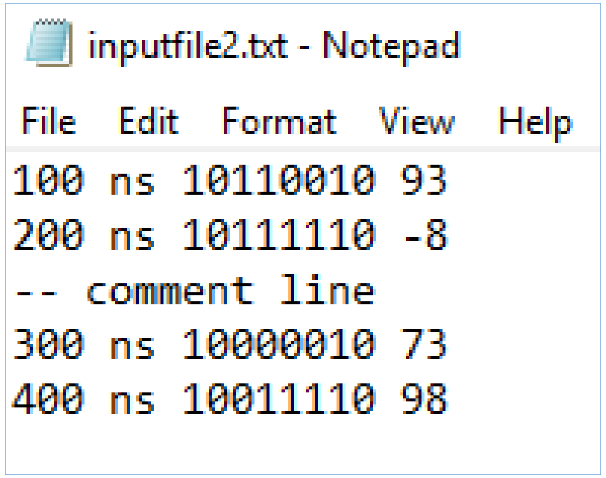
VHDL coding for simulation (free style)

VHDL coding for simulation (free style)



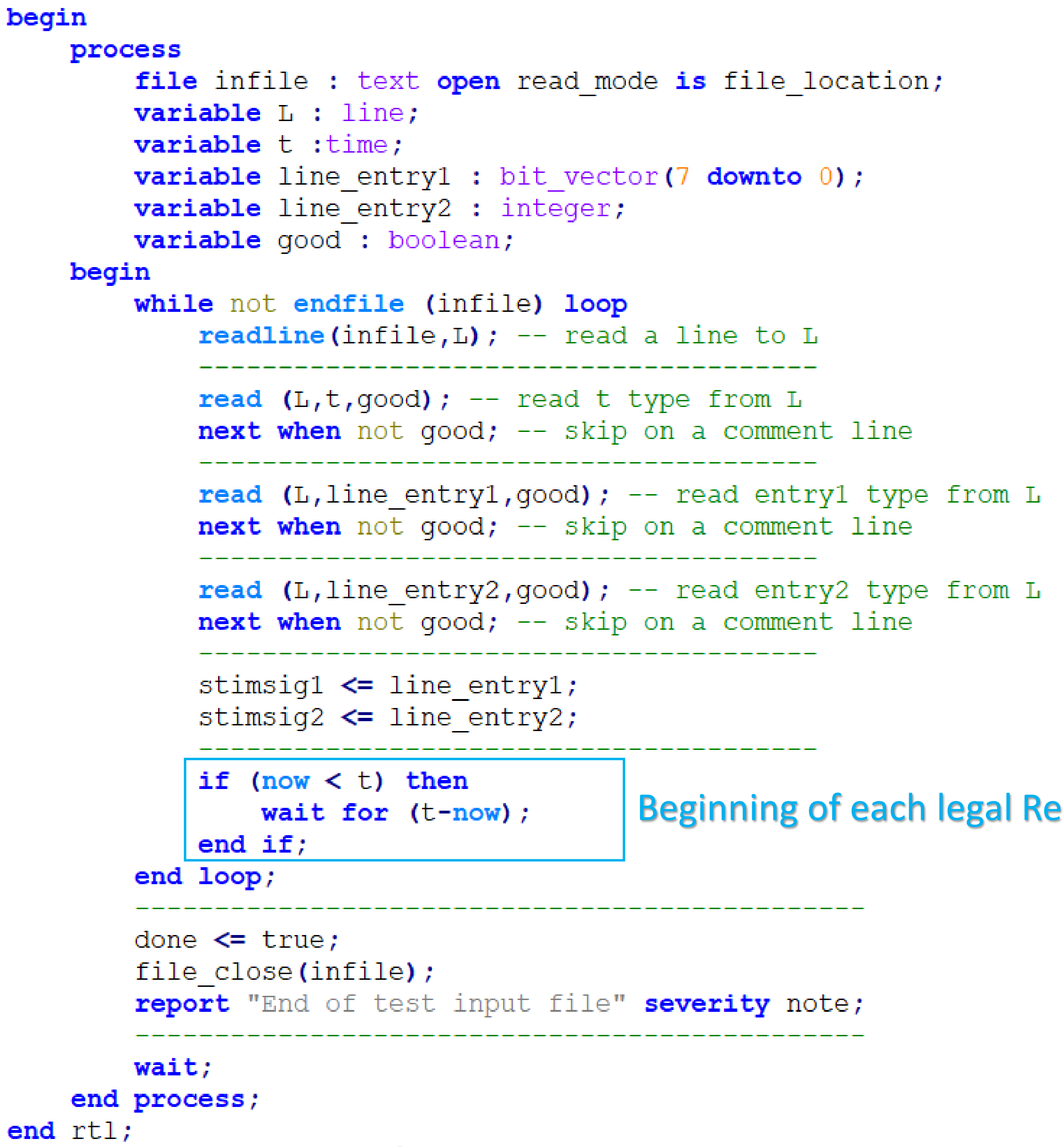
```vhdl
library IEEE;
use ieee.std_logic_1164.all;
use std.textio.all;
---------------------------------------------
entity filegen is
    port(
        stimsig1: out bit_vector(7 downto 0);
        stimsig2: out integer
    );
end filegen;
---------------------------------------------
architecture rtl of filegen is
    signal done : boolean := false;
    constant file_location : string(1 to 39) :=
    "C:\TestPrograms\ModelSim\inputfile2.txt";
begin
```

Internal absolute sampling time information



```
100 ns 10110010 93
200 ns 10111110 -8
-- comment line
300 ns 10000010 73
400 ns 10011110 98
```

Test bench stimulus using **file reading** – Example2

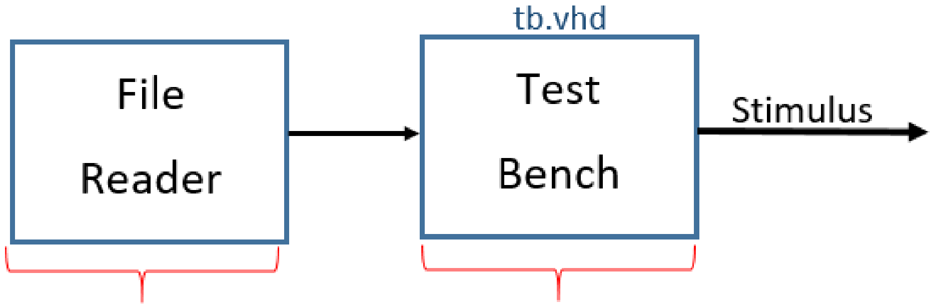


```vhdl
begin
    process
        file infile : text open read_mode is file_location;
        variable L : line;
        variable t :time;
        variable line_entry1 : bit_vector(7 downto 0);
        variable line_entry2 : integer;
        variable good : boolean;
    begin
        while not endfile (infile) loop
            readline(infile,L); -- read a line to L
            ---------------------------------------
            read (L,t,good); -- read t type from L
            next when not good; -- skip on a comment line
            ---------------------------------------
            read (L,line_entry1,good); -- read entry1 type from L
            next when not good; -- skip on a comment line
            ---------------------------------------
            read (L,line_entry2,good); -- read entry2 type from L
            next when not good; -- skip on a comment line
            ---------------------------------------
            stimsig1 <= line_entry1;
            stimsig2 <= line_entry2;
            ---------------------------------------
            if (now < t) then
                wait for (t-now);
            end if;
        end loop;
        ----------------------------------------------
        done <= true;
        file_close(infile);
        report "End of test input file" severity note;
        ----------------------------------------------
        wait;
    end process;
end rtl;
```

Beginning of each legal Reading

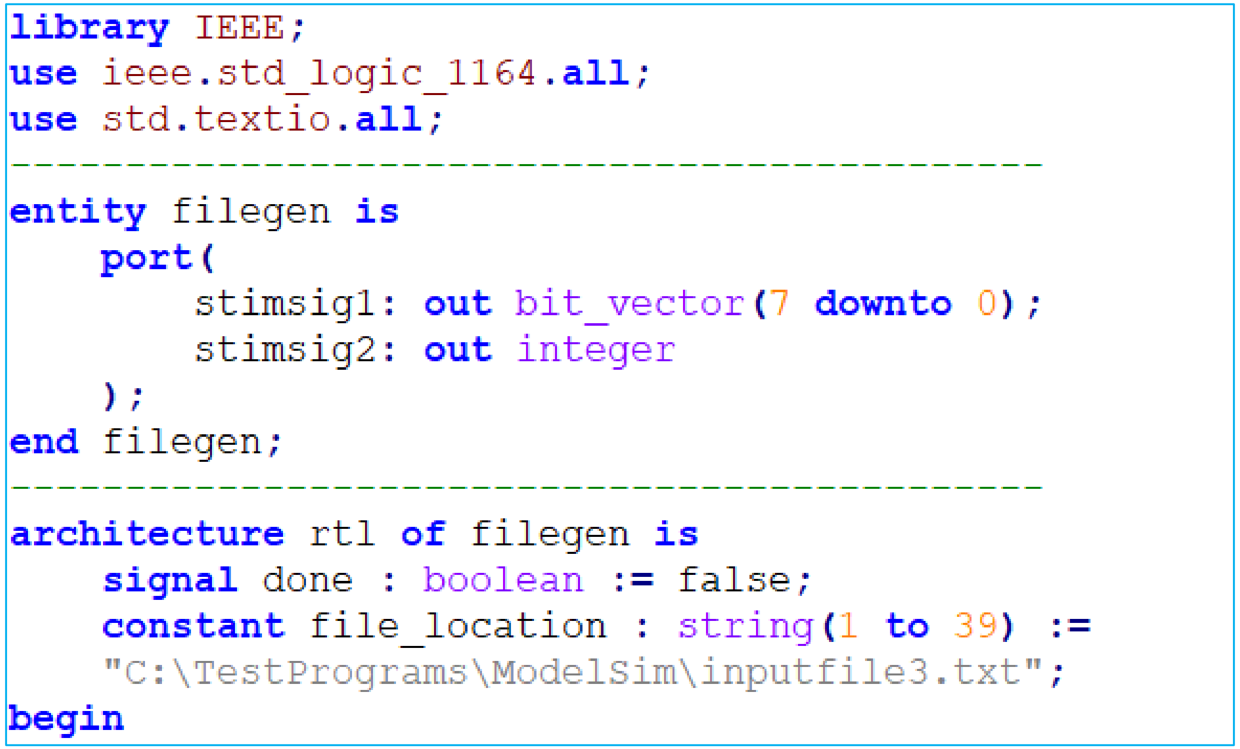
©Hanan Ribo

Test bench stimulus using **file reading** – Example3



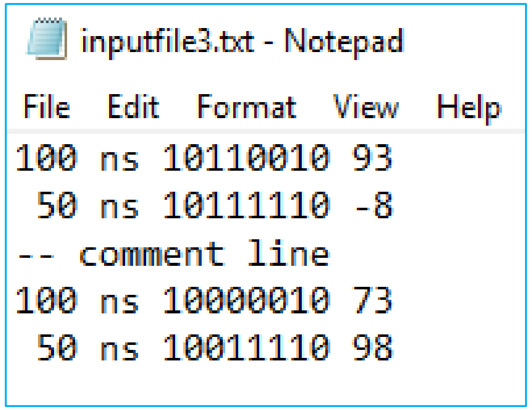
VHDL coding for simulation (free style)

VHDL coding for simulation (free style)



```
library IEEE;
use ieee.std_logic_1164.all;
use std.textio.all;
---------------------------------------------
entity filegen is
    port(
        stimsig1: out bit_vector(7 downto 0);
        stimsig2: out integer
    );
end filegen;
---------------------------------------------
architecture rtl of filegen is
    signal done : boolean := false;
    constant file_location : string(1 to 39) :=
    "C:\TestPrograms\ModelSim\inputfile3.txt";
begin
```

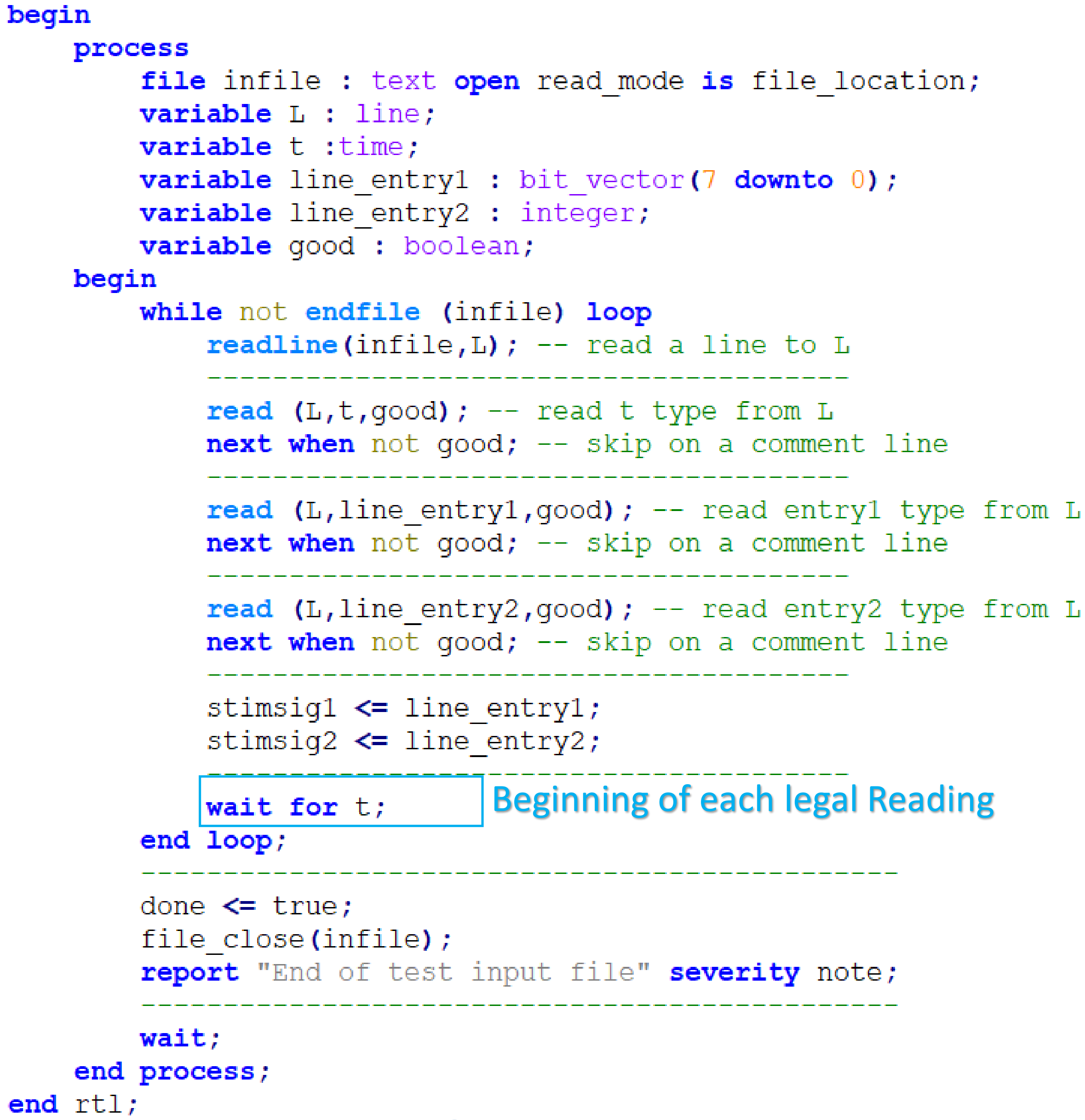
Internal relative sampling time information



```
100 ns 10110010 93
 50 ns 10111110 -8
-- comment line
100 ns 10000010 73
 50 ns 10011110 98
```

©Hanan Ribo

Test bench stimulus using **file reading** – Example3



```
begin
    process
        file infile : text open read_mode is file_location;
        variable L : line;
        variable t :time;
        variable line_entry1 : bit_vector(7 downto 0);
        variable line_entry2 : integer;
        variable good : boolean;
    begin
        while not endfile (infile) loop
            readline(infile,L); -- read a line to L
            ---------------------------------------
            read (L,t,good); -- read t type from L
            next when not good; -- skip on a comment line
            ---------------------------------------
            read (L,line_entry1,good); -- read entry1 type from L
            next when not good; -- skip on a comment line
            ---------------------------------------
            read (L,line_entry2,good); -- read entry2 type from L
            next when not good; -- skip on a comment line
            ---------------------------------------
            stimsig1 <= line_entry1;
            stimsig2 <= line_entry2;
            ---------------------------------------
            wait for t;
        end loop;
        ----------------------------------------------
        done <= true;
        file_close(infile);
        report "End of test input file" severity note;
        ----------------------------------------------
        wait;
    end process;
end rtl;
```

Beginning of each legal Reading

©Hanan Ribo

Test bench results retention using list **file exporting**

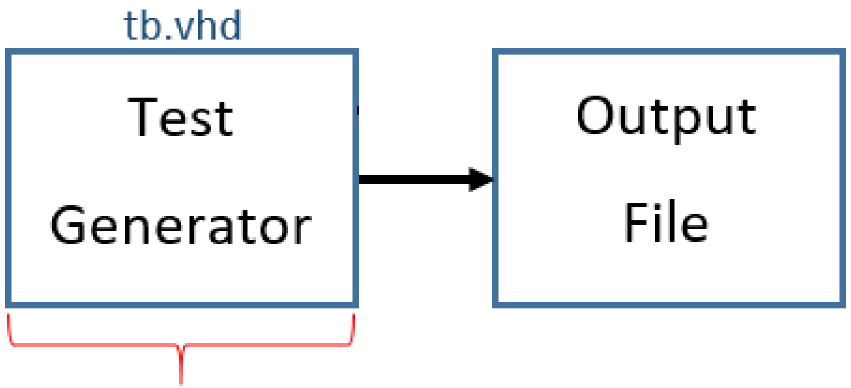
- As we saw, simulation results can be seen using **wave** and **list** visualization forms.
- After sim and run operations choose the list form view and choose the way you prefer to see the results (type in the Transcript window):
  - ✓ With delta expansion: `configure list -delta all`
  - ✓ Without delta expansion: `configure list -delta collapse`
- We can export the list visualization out to a txt file using the next command (the file location is to be the project folder – default location):

```
write  list  name.lst
```

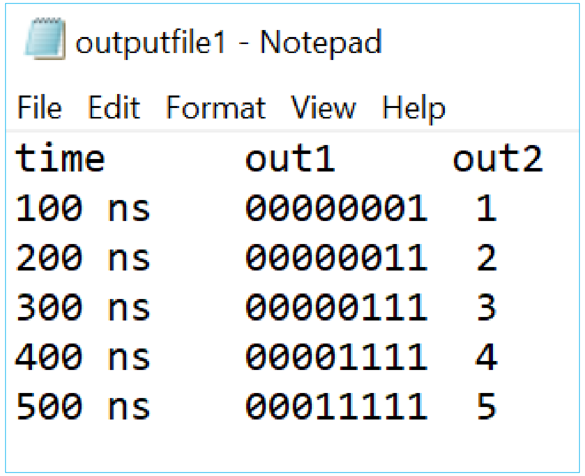
- We can export the list visualization out to a txt file using a specific path location:

```
write  list  C:/Test/ModelSim/Adder/name.lst
```

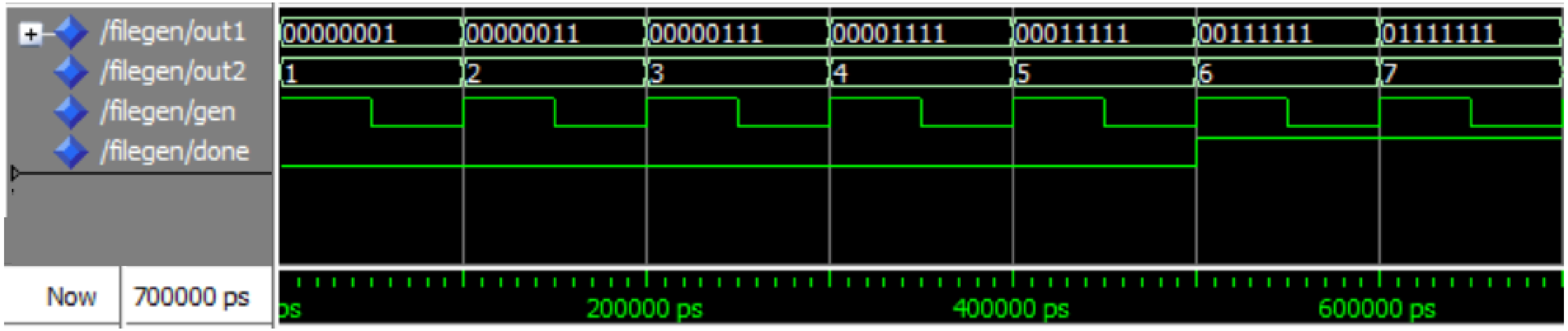
Test bench results retention using **file writing** – Example4



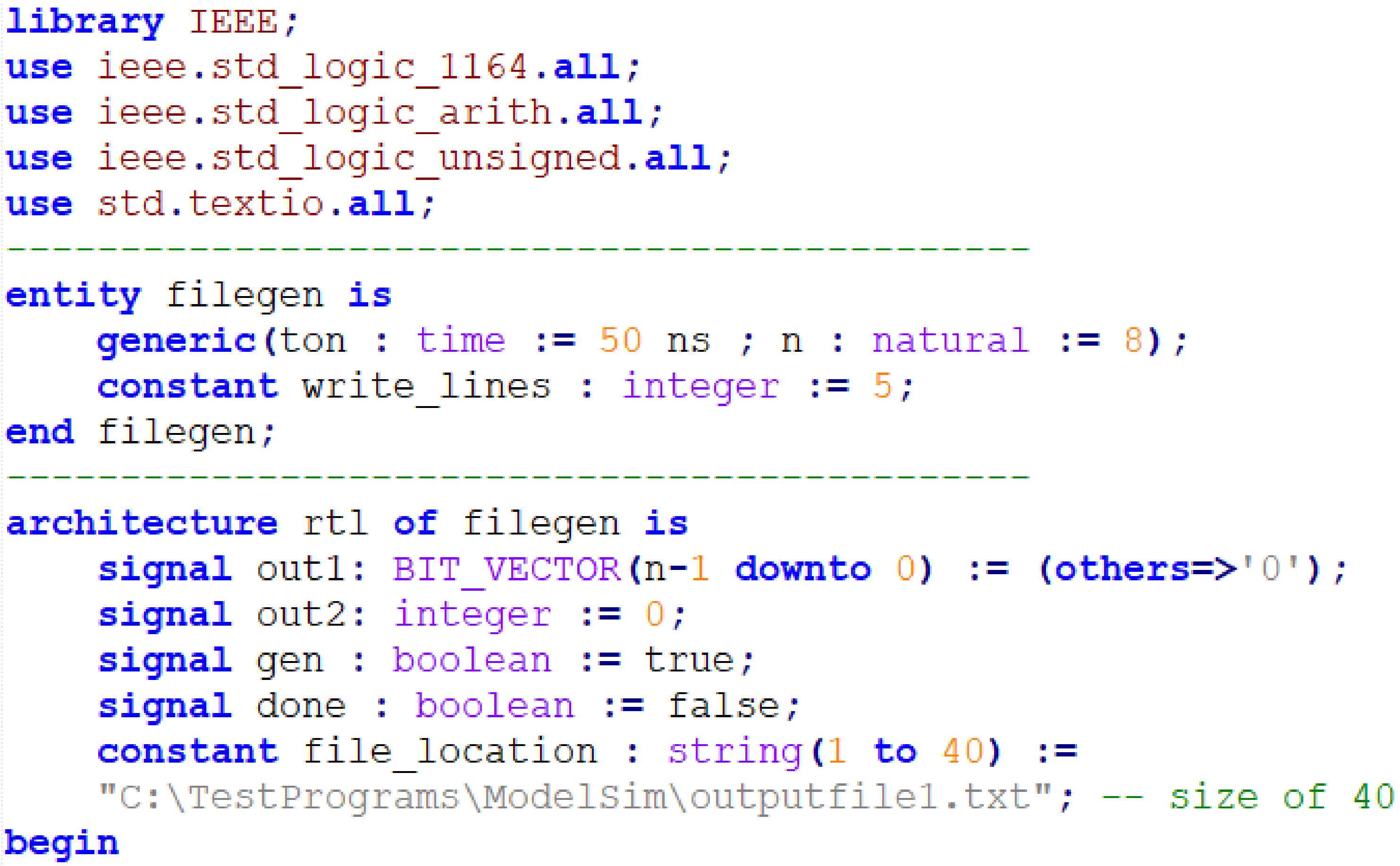
VHDL coding for mulation (free style)



```
time        out1      out2
100 ns      00000001   1
200 ns      00000011   2
300 ns      00000111   3
400 ns      00001111   4
500 ns      00011111   5
```

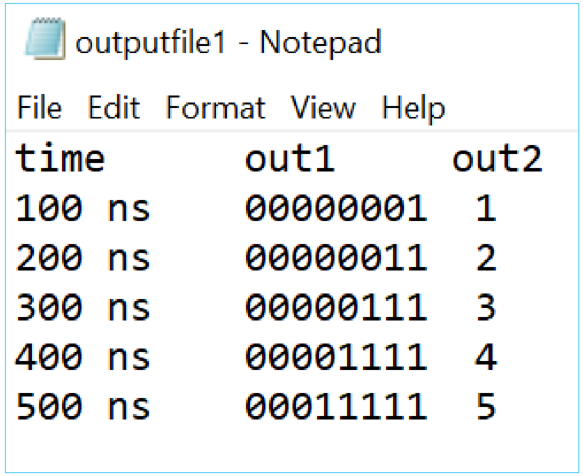


Test bench results retention using **file writing** – Example4

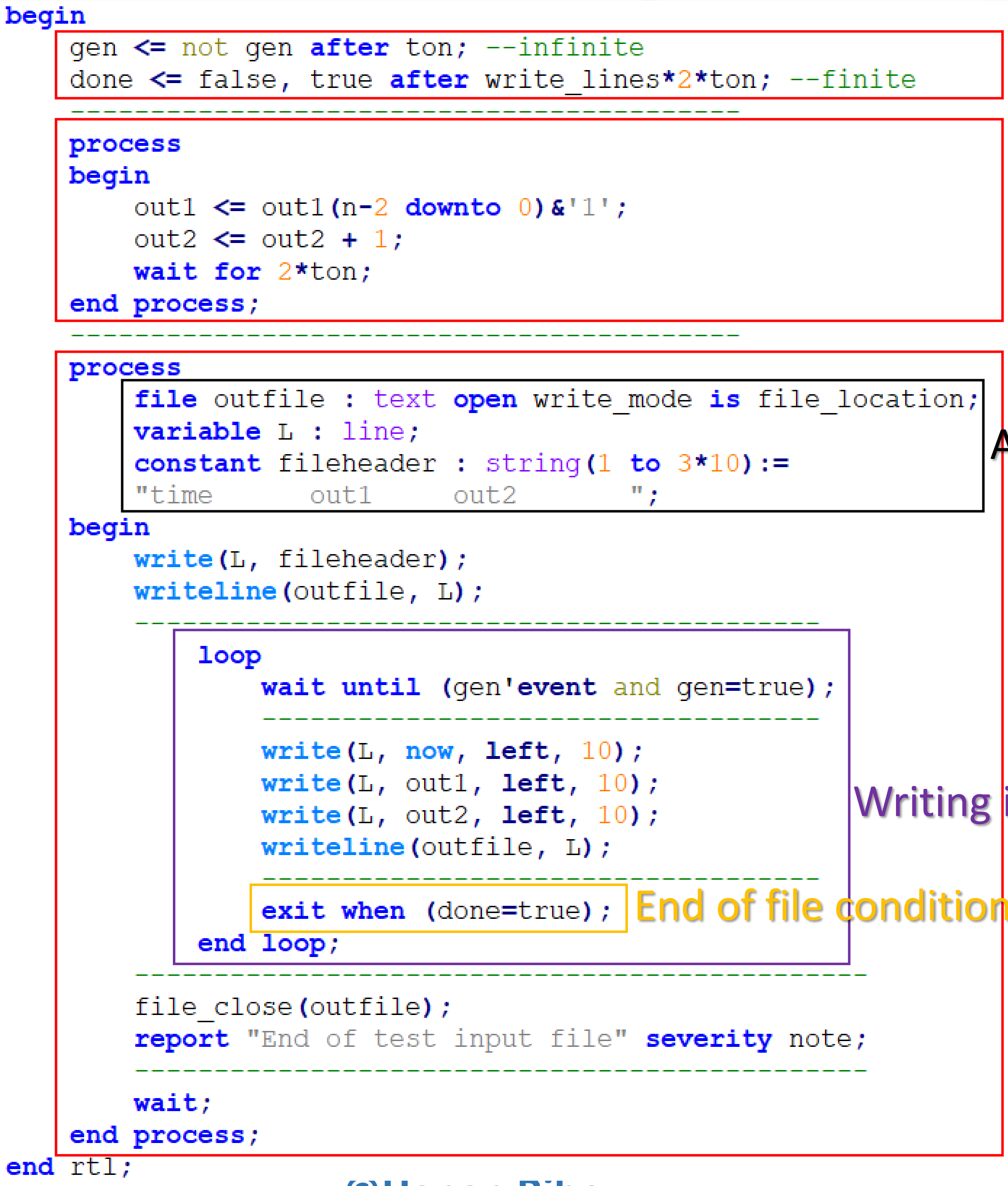


```
library IEEE;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
use std.textio.all;
---------------------------------------------
entity filegen is
    generic(ton : time := 50 ns ; n : natural := 8);
    constant write_lines : integer := 5;
end filegen;
---------------------------------------------
architecture rtl of filegen is
    signal out1: BIT_VECTOR(n-1 downto 0) := (others=>'0');
    signal out2: integer := 0;
    signal gen : boolean := true;
    signal done : boolean := false;
    constant file_location : string(1 to 40) :=
    "C:\TestPrograms\ModelSim\outputfile1.txt"; -- size of 40
begin
```

Test bench results retention using **file writing** – Example4



```
time      out1      out2
100 ns    00000001  1
200 ns    00000011  2
300 ns    00000111  3
400 ns    00001111  4
500 ns    00011111  5
```



```vhdl
begin
    gen <= not gen after ton; --infinite
    done <= false, true after write_lines*2*ton; --finite
    ------------------------------------------
    process
    begin
        out1 <= out1(n-2 downto 0)&'1';
        out2 <= out2 + 1;
        wait for 2*ton;
    end process;
    ------------------------------------------
    process
        file outfile : text open write_mode is file_location;
        variable L : line;
        constant fileheader : string(1 to 3*10):=
        "time      out1      out2        ";
    begin
        write(L, fileheader);
        writeline(outfile, L);
        ------------------------------------------
            loop
                wait until (gen'event and gen=true);
                ------------------------------------
                write(L, now, left, 10);
                write(L, out1, left, 10);
                write(L, out2, left, 10);
                writeline(outfile, L);
                ------------------------------------
                exit when (done=true);
            end loop;
        ----------------------------------------------
        file_close(outfile);
        report "End of test input file" severity note;
        ----------------------------------------------
        wait;
    end process;
end rtl;
```

Trigger signals

Information signals

Auxiliary **Data**

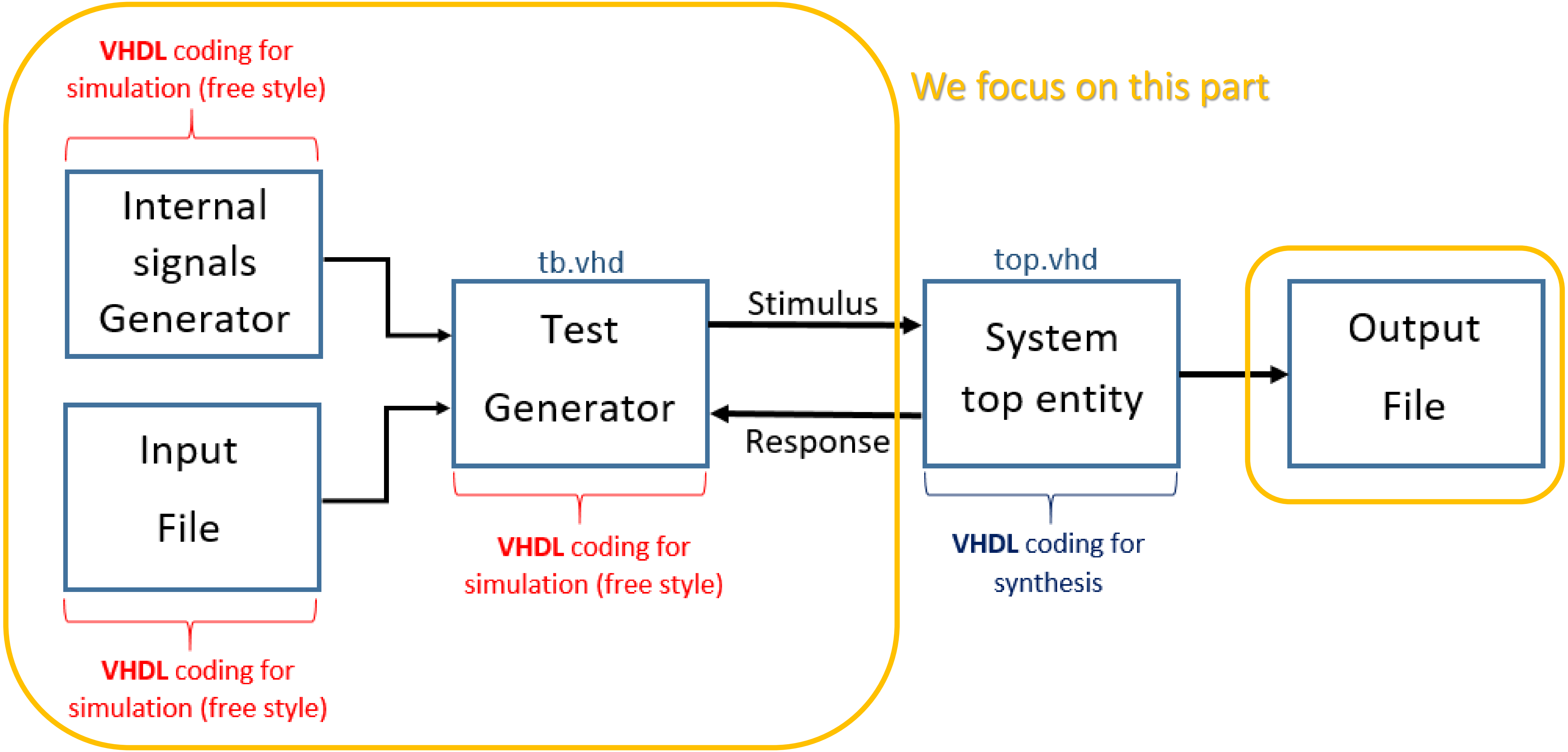
Writing Mechanism

Writing iterations

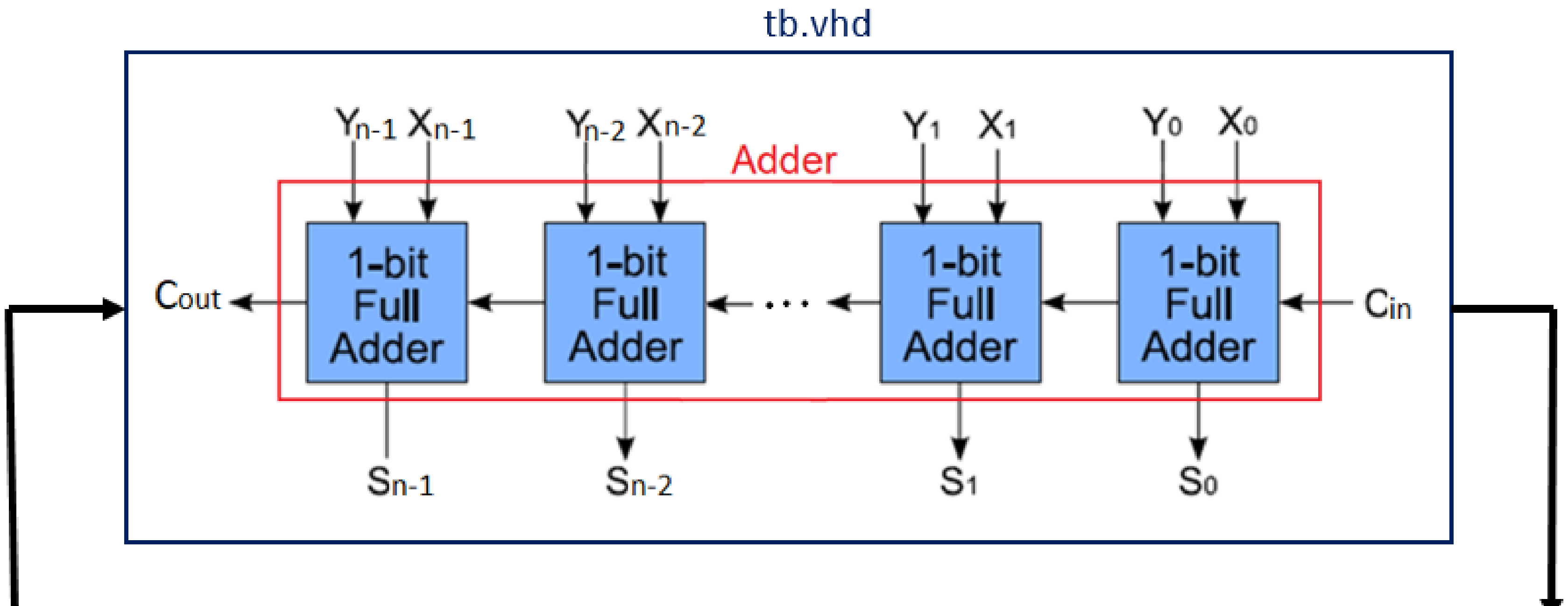
End of file condition

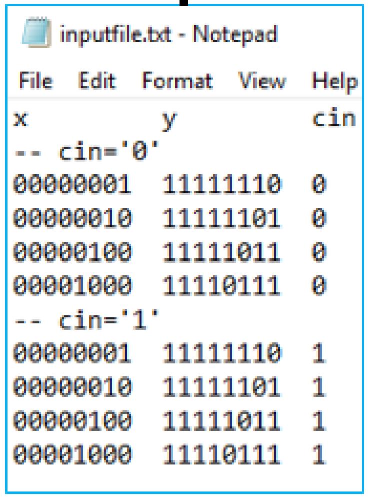
©Hanan Ribo

Advanced Simulation using **file reading** and **writing** – Example5


VHDL coding for simulation (free style)
Internal signals Generator
Input File
VHDL coding for simulation (free style)
tb.vhd
Test Generator
VHDL coding for simulation (free style)
Stimulus
Response
top.vhd
System top entity
VHDL coding for synthesis
Output File
We focus on this part

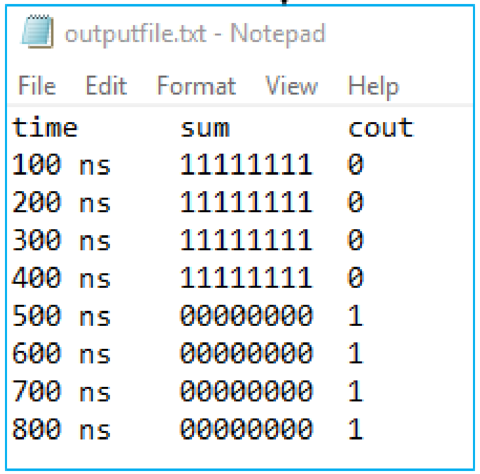
Advanced Simulation using file reading and writing – Example5





```
x          y          cin
-- cin='0'
00000001  11111110  0
00000010  11111101  0
00000100  11111011  0
00001000  11110111  0
-- cin='1'
00000001  11111110  1
00000010  11111101  1
00000100  11111011  1
00001000  11110111  1
```

Input file

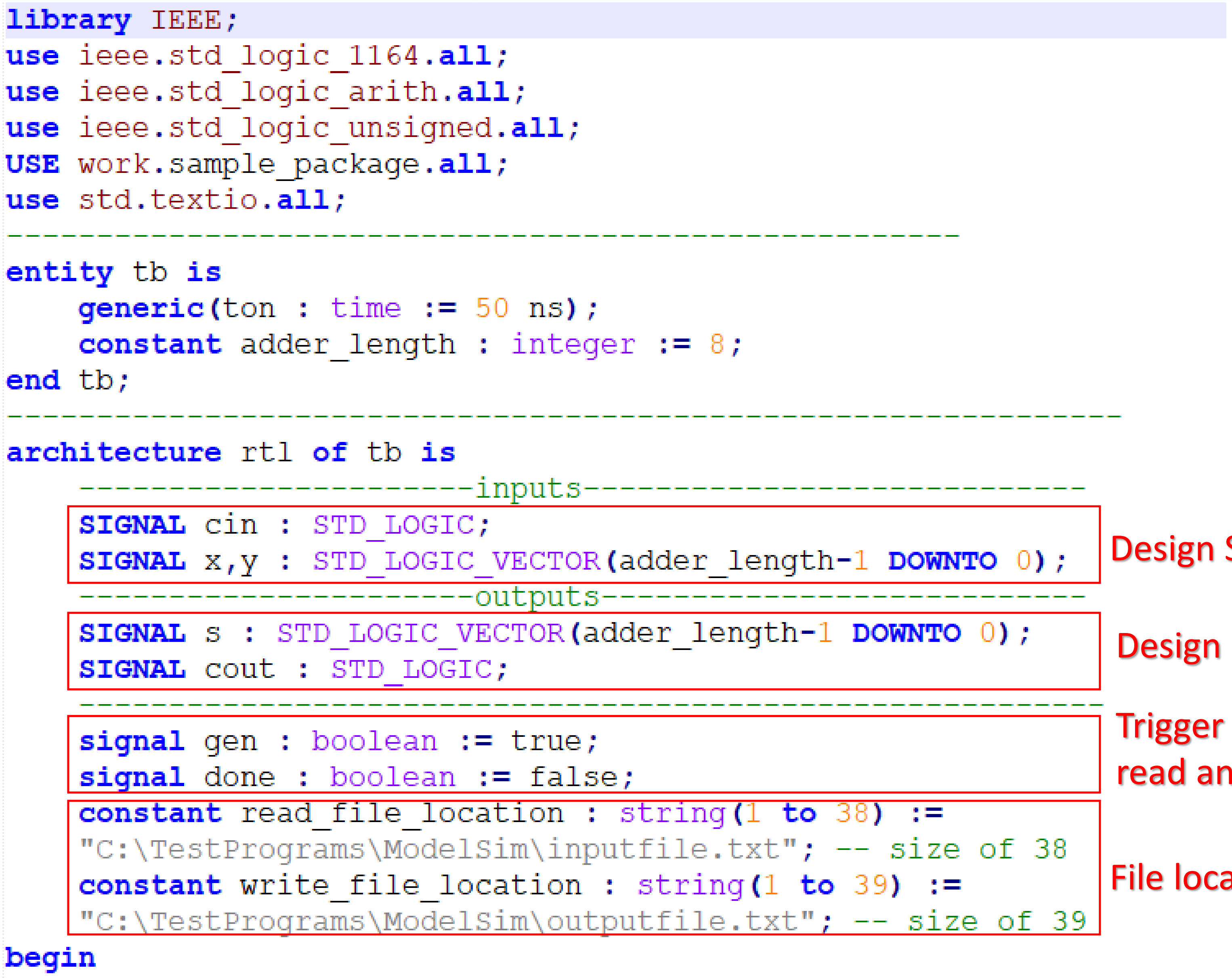


```
time        sum       cout
100 ns      11111111  0
200 ns      11111111  0
300 ns      11111111  0
400 ns      11111111  0
500 ns      00000000  1
600 ns      00000000  1
700 ns      00000000  1
800 ns      00000000  1
```

Output file

©Hanan Ribo

Advanced Simulation using **file reading** and **writing** – Example5



```
library IEEE;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
USE work.sample_package.all;
use std.textio.all;
-----------------------------------------------------
entity tb is
    generic(ton : time := 50 ns);
    constant adder_length : integer := 8;
end tb;
--------------------------------------------------------------
architecture rtl of tb is
    ----------------------inputs----------------------------
    SIGNAL cin : STD_LOGIC;
    SIGNAL x,y : STD_LOGIC_VECTOR(adder_length-1 DOWNTO 0);
    ----------------------outputs---------------------------
    SIGNAL s : STD_LOGIC_VECTOR(adder_length-1 DOWNTO 0);
    SIGNAL cout : STD_LOGIC;
    ---------------------------------------------------------
    signal gen : boolean := true;
    signal done : boolean := false;
    constant read_file_location : string(1 to 38)  :=
    "C:\TestPrograms\ModelSim\inputfile.txt"; -- size of 38
    constant write_file_location : string(1 to 39)  :=
    "C:\TestPrograms\ModelSim\outputfile.txt"; -- size of 39
begin
```

Design Stimulus signals

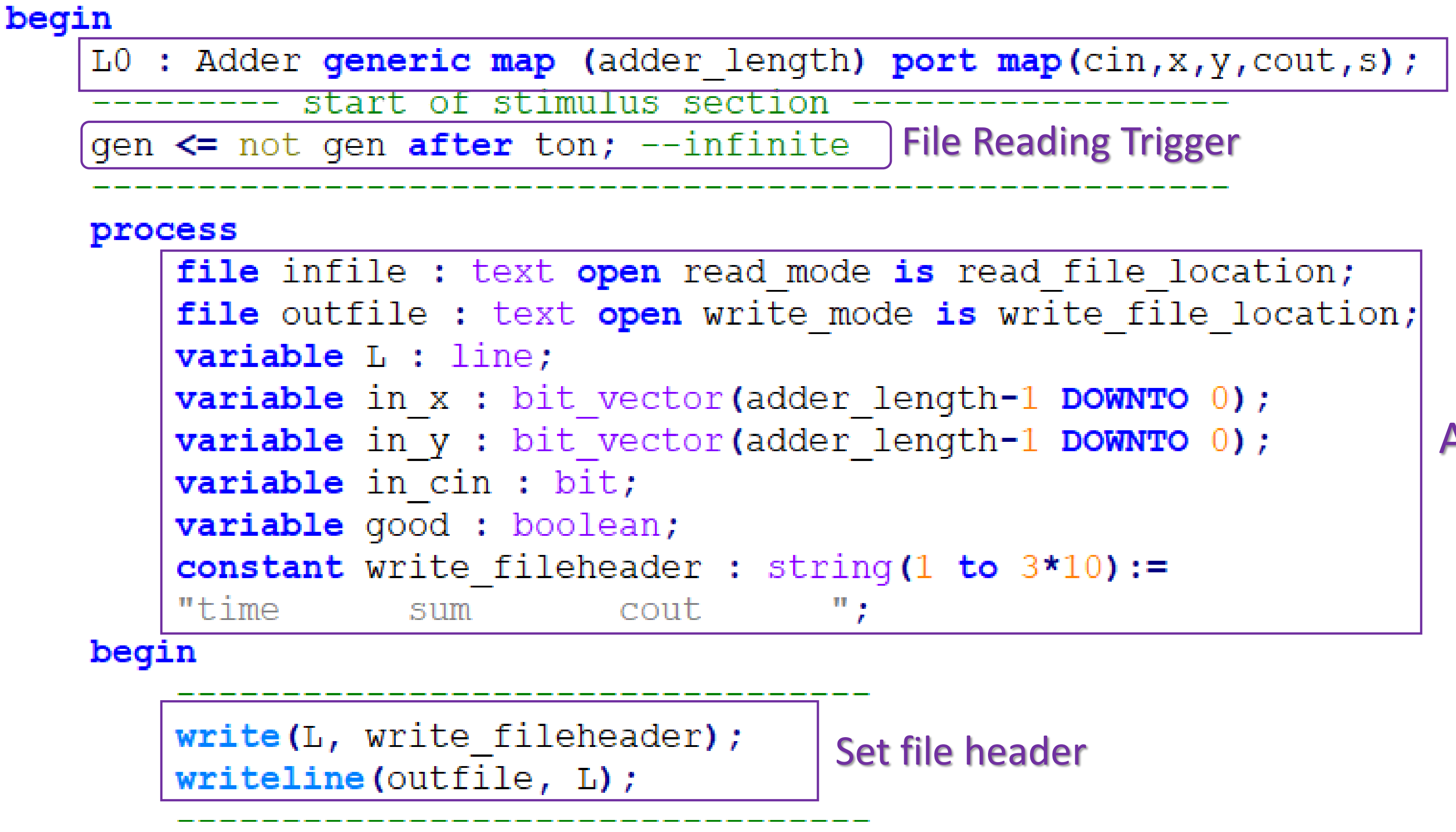
Design Response signals

Trigger signals used by file read and write operations

File location strings

©Hanan Ribo

Advanced Simulation using **file reading** and **writing** – Example5



```
begin
    L0 : Adder generic map (adder_length) port map(cin,x,y,cout,s);
    --------- start of stimulus section ------------------
    gen <= not gen after ton; --infinite
    ------------------------------------------------------
    process
        file infile : text open read_mode is read_file_location;
        file outfile : text open write_mode is write_file_location;
        variable L : line;
        variable in_x : bit_vector(adder_length-1 DOWNTO 0);
        variable in_y : bit_vector(adder_length-1 DOWNTO 0);
        variable in_cin : bit;
        variable good : boolean;
        constant write_fileheader : string(1 to 3*10):=
        "time      sum        cout      ";
    begin
        ---------------------------------
        write(L, write_fileheader);
        writeline(outfile, L);
        ---------------------------------
```

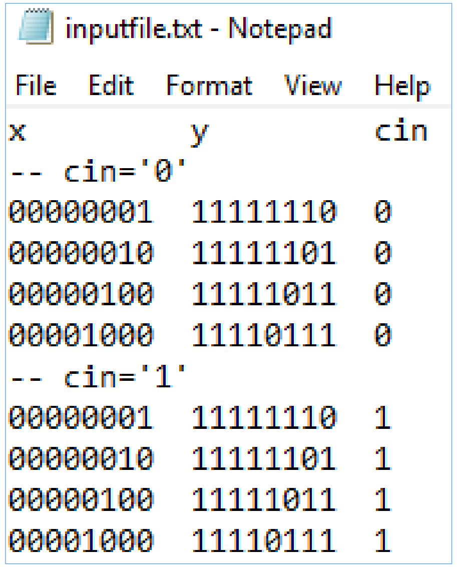
DUT = Design Under Test

File Reading Trigger

Auxiliary **Data**

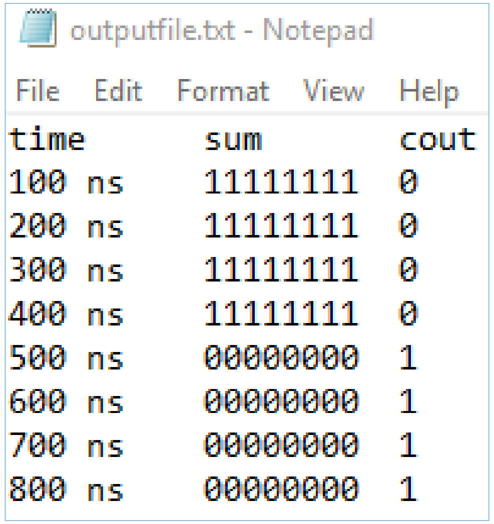
Set file header

Advanced Simulation using **file reading** and **writing** – Example5



```
x           y          cin
-- cin='0'
00000001   11111110   0
00000010   11111101   0
00000100   11111011   0
00001000   11110111   0
-- cin='1'
00000001   11111110   1
00000010   11111101   1
00000100   11111011   1
00001000   11110111   1
```

Input file



```
time       sum        cout
100 ns     11111111   0
200 ns     11111111   0
300 ns     11111111   0
400 ns     11111111   0
500 ns     00000000   1
600 ns     00000000   1
700 ns     00000000   1
800 ns     00000000   1
```

Output file

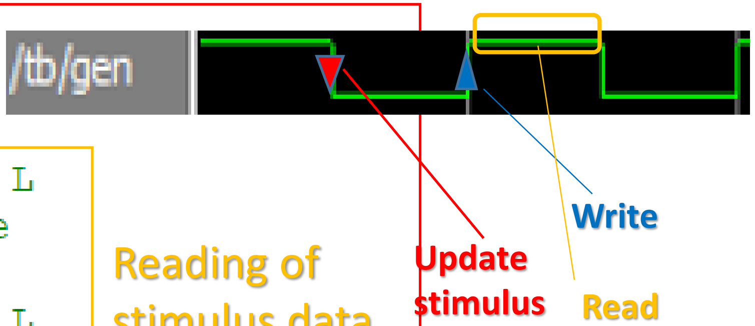
```
while not endfile (infile) loop
    readline(infile,L); -- read a line to L
    --------------------------------------
    read (L,in_x,good); -- read entry1 type from L
    next when not good; -- skip on a comment line
    --------------------------------------
    read (L,in_y,good); -- read entry2 type from L
    next when not good; -- skip on a comment line
    --------------------------------------
    read (L,in_cin,good); -- read entry3 type from L
    next when not good; -- skip on a comment line
    --------------------------------------
    wait until (gen'event and gen=false);
    x <= to_stdlogicvector(in_x);
    y <= to_stdlogicvector(in_y);
    cin <= to_stdulogic(in_cin);
    --------------------------------------
    wait until (gen'event and gen=true);
        write(L, now, left, 10);
        write(L, to_bitvector(s), left, 10);
        write(L, to_bit (cout), left, 10);
        writeline(outfile, L);
    --------------------------------------
    --wait until gen; -- delay process until gen='1'
end loop;
```

Reading of stimulus data from input file

DUT Stimulation

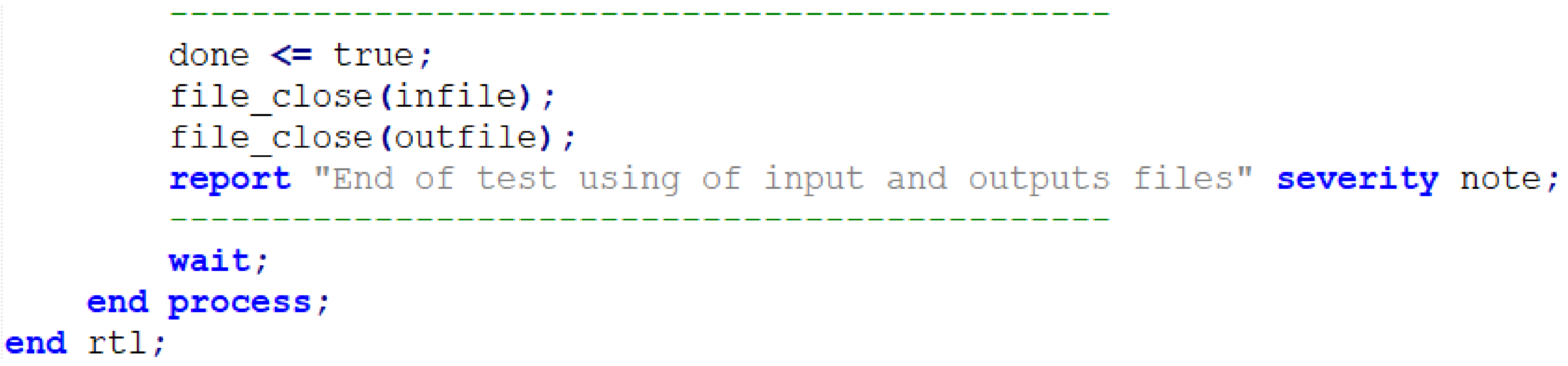
Write DUT response to file

Check this out as a DUT stimulation and discern What's happening

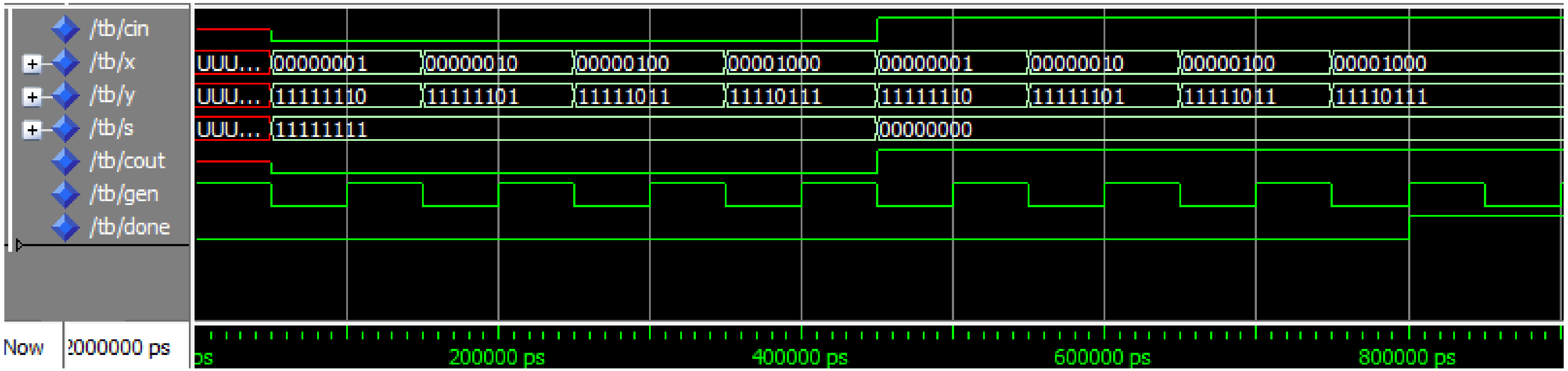


Write after Read iterations

Advanced Simulation using **file reading** and **writing** – Example5



```
        ----------------------------------------------
        done <= true;
        file_close(infile);
        file_close(outfile);
        report "End of test using of input and outputs files" severity note;
        ----------------------------------------------
        wait;
    end process;
end rtl;
```



/tb/cin
/tb/x
UUU... 00000001 00000010 00000100 00001000 00000001 00000010 00000100 00001000
/tb/y
UUU... 11111110 11111101 11111011 11110111 11111110 11111101 11111011 11110111
/tb/s
UUU... 11111111 00000000
/tb/cout
/tb/gen
/tb/done
Now
2000000 ps
ps
200000 ps
400000 ps
600000 ps
800000 ps

©Hanan Ribo