

Testbench File I/O

Last updated 7/21/23

Testbench File I/O

- File I/O
 - use package: *std.textio.all*
 - Includes types: *Text*, *File_Open_Status*, *Line*
 - Output Process
 - Create and open a file
 - Build up a line to output to the file
 - Output the line
 - ...
 - Build up a line to ...
 - Output the line
 - Repeat
 - ...
 - Close the file

Testbench File I/O

- File I/O
 - Create a FILE
 - A file can be any type, string, std_logic, ... `text`
 - Create the file object and status variable

```
file out_file: text;
```

```
variable file_status: File_Open_Status;
```

- Open the file

```
file_open(file_status, out_file, "my_file_name", write_mode;  
append_mode  
in_file read_mode
```

- File_Open_Status: OPEN_OK, STATUS_ERROR, NAME_ERROR ,
MODE_ERROR

Testbench File I/O

- File I/O

- Create a LINE to output

- Create the line object (buffer)

- ```
variable line_out: line;
```

- Create the line to output

- *write* concatenates until it is pushed out

- ```
write(line_out, stringvar1);
```

- ```
write(line_out, stringvar2);
```

- ```
write(line_out, stringvar3); write(line_out, stringvar4);
```

- Where “stringvar” matches the type stored in the file

Testbench File I/O

- File I/O

- Output the line

- *writeline* outputs the line

- ```
writeline(out_file, line_out);
```

- Close the file

- ```
file_close(out_file);
```

- NOTE: Ending the process in which the file was opened automatically closes the file ???

Testbench File I/O

- File I/O
 - use package: *std.textio.all*
 - Includes types: *Text*, *File_Open_Status*, *Line*
- Input Process
 - Open the desired file
 - Check for end of file
 - Read in a line from the file
 - Parse the line for specific inputs
 - ...
 - Read in a line from the file
 - Parse the line
 - Repeat
 - ...
 - Close the file

Testbench File I/O

- File I/O

- Open the FILE

- Open the file

```
file_open(file_status, out_file, "my_file_name", write_mode;  
append_mode  
in_file read_mode
```

- File_Open_Status: OPEN_OK, STATUS_ERROR, NAME_ERROR ,
MODE_ERROR

Testbench File I/O

- File I/O
 - Create a LINE for input
 - Create the line object (buffer)
variable line_in: line;
 - Read the line
 - *readline* reads the line
readline(in_file, line_in);

Testbench File I/O

- File I/O

- Parse the line

- `read(line_in, stringvar1);`

- `read(line_in, stringvar2);`

- `read(line_in, stringvar3); read(line_in, stringvar4);`

- Check for end of file

- `endfile(in_file)` returns a Boolean T/F

- Close the file

- `file_close(in_file);`

- NOTE: Ending the process in which the file was opened automatically closes the file ???

Testbench File I/O

- Testbench
 - Input generation – save to file
 - Results documentation

```
-----  
-- file_io_tb.vhdl  
-- by: johnsontimoj  
-- created: 7/20/18  
-- version: 0.0  
-----  
--  
-- File IO example  
-- inputs: None  
-- outputs: text file  
-- simulation run time 2000 ns  
-----  
  
library ieee;  
use ieee.std_logic_1164.all;  
use ieee.numeric_std.all;  
use std.textio.all;  
  
entity file_io_tb is  
  -- no I/O  
end entity;  
  
architecture testbench of file_io_tb is  
  constant per: time := 20 ns;  
  signal clk: std_logic := '0';  
  signal cnt_in: std_logic_vector(20 downto 0);  
  file out_file: text;  
  file in_file: text;
```

```
begin  
  writeIO: process  
    variable file_status: file_open_status;  
    variable out_line: line;  
  
    variable test_num: integer range 0 to 200;  
    constant str1: string(1 to 6) := "test: ";  
  
    variable t: time range 0 ns to 2000 ns;  
    constant str2: string(1 to 3) := "t= ";  
  
    variable clock: bit;  
    constant str3: string(1 to 5) := "clk= ";  
  
    variable cnt: integer := 0;  
    constant str4: string(1 to 5) := "cnt= ";  
  
    begin  
      file_open(file_status, out_file, "file_io_out", write_mode);  
      -- not checking status for now  
  
      for test_num in 0 to 200 loop  
        wait for per/2;  
        clk <= not clk;  
        clock := to_bit(clk); -- must output bit type  
        if (clk = '1') then  
          cnt := (5 + cnt);  
        end if;  
  
        write(out_line, str1); write(out_line, test_num); write(out_line, ht);  
        write(out_line, str2); write(out_line, now); write(out_line, ht);  
        write(out_line, str3); write(out_line, clock); write(out_line, ht);  
        write(out_line, str4); write(out_line, cnt);  
        writeline(out_file, out_line);  
      end loop;  
      file_close(out_file);  
    end process;
```

Testbench File I/O

- Testbench
 - Input generation – save to file
 - Results documentation

```
file_io_out - Notepad
File Edit Format View Help
test: 0 t= 10 ns      clk= 0 cnt= 0
test: 1 t= 20 ns      clk= 1 cnt= 5
test: 2 t= 30 ns      clk= 0 cnt= 5
test: 3 t= 40 ns      clk= 1 cnt= 10
test: 4 t= 50 ns      clk= 0 cnt= 10
test: 5 t= 60 ns      clk= 1 cnt= 15
test: 6 t= 70 ns      clk= 0 cnt= 15
test: 7 t= 80 ns      clk= 1 cnt= 20
test: 8 t= 90 ns      clk= 0 cnt= 20
test: 9 t= 100 ns     clk= 1 cnt= 25
test: 10          t= 110 ns      clk= 0 cnt= 25
test: 11          t= 120 ns      clk= 1 cnt= 30
test: 12          t= 130 ns      clk= 0 cnt= 30
test: 13          t= 140 ns      clk= 1 cnt= 35
test: 14          t= 150 ns      clk= 0 cnt= 35
test: 15          t= 160 ns      clk= 1 cnt= 40
test: 16          t= 170 ns      clk= 0 cnt= 40
test: 17          t= 180 ns      clk= 1 cnt= 45
test: 18          t= 190 ns      clk= 0 cnt= 45
test: 19          t= 200 ns      clk= 1 cnt= 50
```

Testbench File I/O

- Testbench
 - Input generation – read from file

```
-----  
-- file_io_tb.vhdl  
-- by: johnsontimoj  
-- created: 7/20/18  
-- version: 0.0  
-----  
--  
-- File IO example  
-- inputs: None  
-- outputs: text file  
-- simulation run time 2000 ns  
-----  
library ieee;  
use ieee.std_logic_1164.all;  
use ieee.numeric_std.all;  
use std.textio.all;  
entity file_io_tb is  
  -- no I/O  
end entity;  
architecture testbench of file_io_tb is  
  constant per: time := 20 ns;  
  signal clk: std_logic := '0';  
  signal cnt_in: std_logic_vector(20 downto 0);  
  file out_file: text;  
  file in_file: text;
```

```
begin  
--  
  readIO: process  
    variable file_status: file_open_status;  
    variable in_line: line;  
  
    variable test_num: integer range 0 to 200;  
    variable t_old: time := 0 ns;  
    variable t_new: time;  
    variable clock: integer;  
    variable cnt: integer;  
    variable str1: string(1 to 6) ;  
    variable str2: string(1 to 3) ;  
    variable str3: string(1 to 5) ;  
    variable str4: string(1 to 5) ;  
    variable tab: string(1 to 1);  
  
    begin  
      file_open(file_status, in_file, "file_io_out", read_mode);  
      -- not checking status for now  
  
      while not (endfile(in_file)) loop  
        readline(in_file, in_line);  
        read(in_line, str1); read(in_line, test_num); read(in_line, tab);  
        read(in_line, str2); read(in_line, t_new); read(in_line, tab);  
        read(in_line, str3); read(in_line, clock); read(in_line, tab);  
        read(in_line, str4); read(in_line, cnt);  
  
        wait for (t_new - t_old);  
        cnt_in <= std_logic_vector(to_unsigned(cnt, cnt_in'length));  
        clk <= std_logic(to_unsigned(clock,1)(0));  
        t_old := t_new;  
      end loop;  
      wait;  
    end process;  
end architecture;
```

Testbench File I/O

- Testbench
 - Input generation – read from file

```
file_io_out - Notepad
File Edit Format View Help
test: 0 t= 10 ns      clk= 0 cnt= 0
test: 1 t= 20 ns      clk= 1 cnt= 5
test: 2 t= 30 ns      clk= 0 cnt= 5
test: 3 t= 40 ns      clk= 1 cnt= 10
test: 4 t= 50 ns      clk= 0 cnt= 10
test: 5 t= 60 ns      clk= 1 cnt= 15
```

