VHDL File I/O

Zheng Peng
City College of New York

Courtesy of Erkay Savaş from Sabanci University

Basic I/O and applications

- Objects of `file` type
  - It is a special type that serve as an interface between the VHDL programs and the host environment.
- Motivation
  - How file objects are created, read, written and used within VHDL simulations

The Package TEXTIO

- A standard package supported by all VHDL simulators
- It provides a standard set of file types, data types, and I/O functions
- TEXTIO is in the library STD
- The library STD does not have to be explicitly declared
- However, the packages must be declared in order to use the package content
  - Syntax: **use** STD.textio.**all**;

Basic I/O Operations

- Type of a file object
  - Depends what sort of data is stored in them
  - Can be anything: integer, string, real number, std_logic_vector, etc.
- Three types of basic operations
  - Declaration of a file and its type
  - Opening and closing a file of a specified type
  - Reading and writing a file.

File Declarations

- File types
  - `TEXT` is file of string
    - Contain ASCII characters, that form human readable text
  - `INTF` is file of integer
    - Can store sequence of integers that are stored in <u>binary</u> form

Opening and Closing Files

- After declaration, files must be opened prior to use
- After use the files must be closed
- We have procedures for opening and closing files:
  - **procedure** `FILE_OPEN(`**file** `file_handle: FILE_TYPE; File_Name:` **in** `STRING; Open_Kind:` **in** `FILE_OPEN_KIND:=READ_MODE);`
  - **procedure** `FILE_CLOSE(`**file** `file_handle: FILE_TYPE);`

Opening and Closing Files

- `file file_handle: FILE_TYPE;`
  - pointer to the file
- `File_Name: in STRING`
  - name of the file
- `Open_Kind: in FILE_OPEN_KIND:=READ_MODE`
  - In which mode the file is to be opened.
- The opening modes for a file:
  - READ_MODE -- default mode
  - WRITE_MODE
  - APPEND_MODE

Reading and Writing Files (1)

- TEXTIO mechanism
  - **type** LINE **is access** STRING;

  -- A LINE is a pointer to a string
  - **type** TEXT **is file of** STRING;

  -- A file of variable-length ASCII records
  - **procedure** READLINE(**file** F: TEXT; L: **out** LINE);
  - **procedure** READ(L: **inout** LINE; value: **out** bit_vector);
  - **procedure** WRITELINE(**file** F: TEXT; L: **inout** LINE);
  - **procedure** WRITE(L: **inout** LINE; value: **out** bit_vector);

Reading and Writing Files (2)

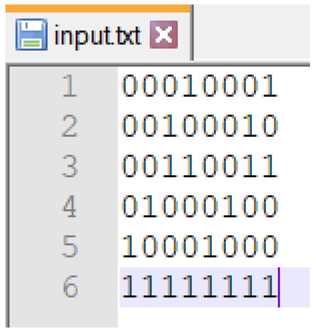
- `LINE` serve as a buffer area for reading and writing
  - `read()` and `write()` procedures access and operate on this buffer
  - They are overloaded and defined for `bit`, `bit_vector`, and `string`
  - `readline()` and `writeline()` procedures move the contents of this buffer to and from files
- `Access` types are similar to pointers in C language
- There are two special file handles called `input` and `output` that are defined in the package TEXTIO

Example (1)

- Read from a file and save the value in an array
  - Each line of the file contains 8 binary digits
  - Each binary digit is represented in the file as an ASCII code
  - We want to save each 8-digit binary number into an array
  - Each element of the array is a 8-digit `STD_LOGIC_VECTOR`
  - Each element of the `STD_LOGIC_VECTOR` is of the type `STD_LOGIC`

Example (2)

- Input file contents



```
00010001
00100010
00110011
01000100
10001000
11111111
```

- Type declaration of the array

```
type buf8x8 is array (0 to 8) of STD_LOGIC_VECTOR(7 downto 0);
```

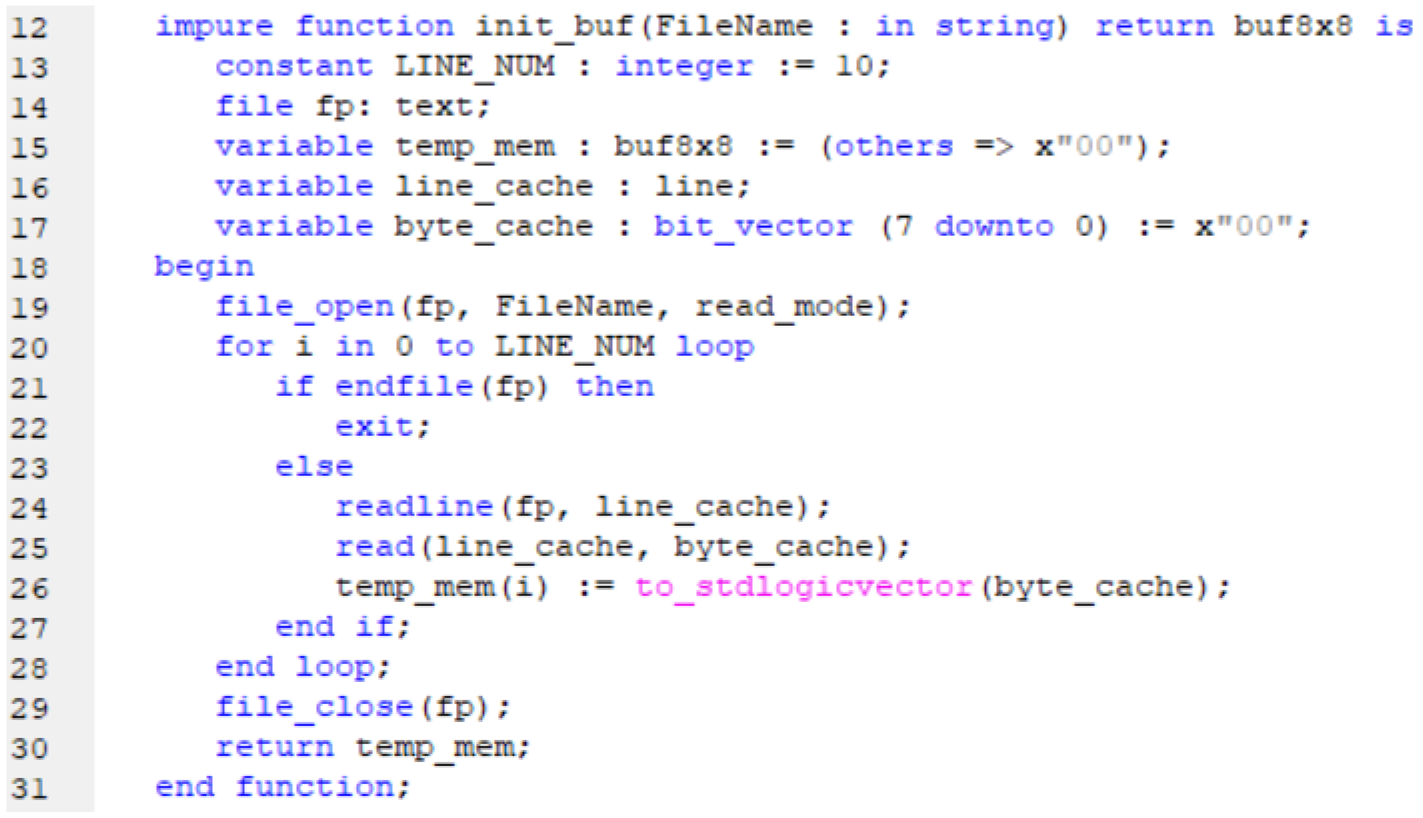
Example (3)

- Entity declaration and architecture definition

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use STD.textio.all;

entity file_io_ex2 is
     Port ( I_EN : in  STD_LOGIC);
end file_io_ex2;

architecture Behavioral of file_io_ex2 is
   type buf8x8 is array (0 to 8) of STD_LOGIC_VECTOR(7 downto 0);
   signal mem : buf8x8;

   ..................

begin
   process(I_EN)
   begin
      if I_EN = '1' then
         mem <= init_buf("input.txt");
      end if;
   end process;
end Behavioral;
```

Example (4)

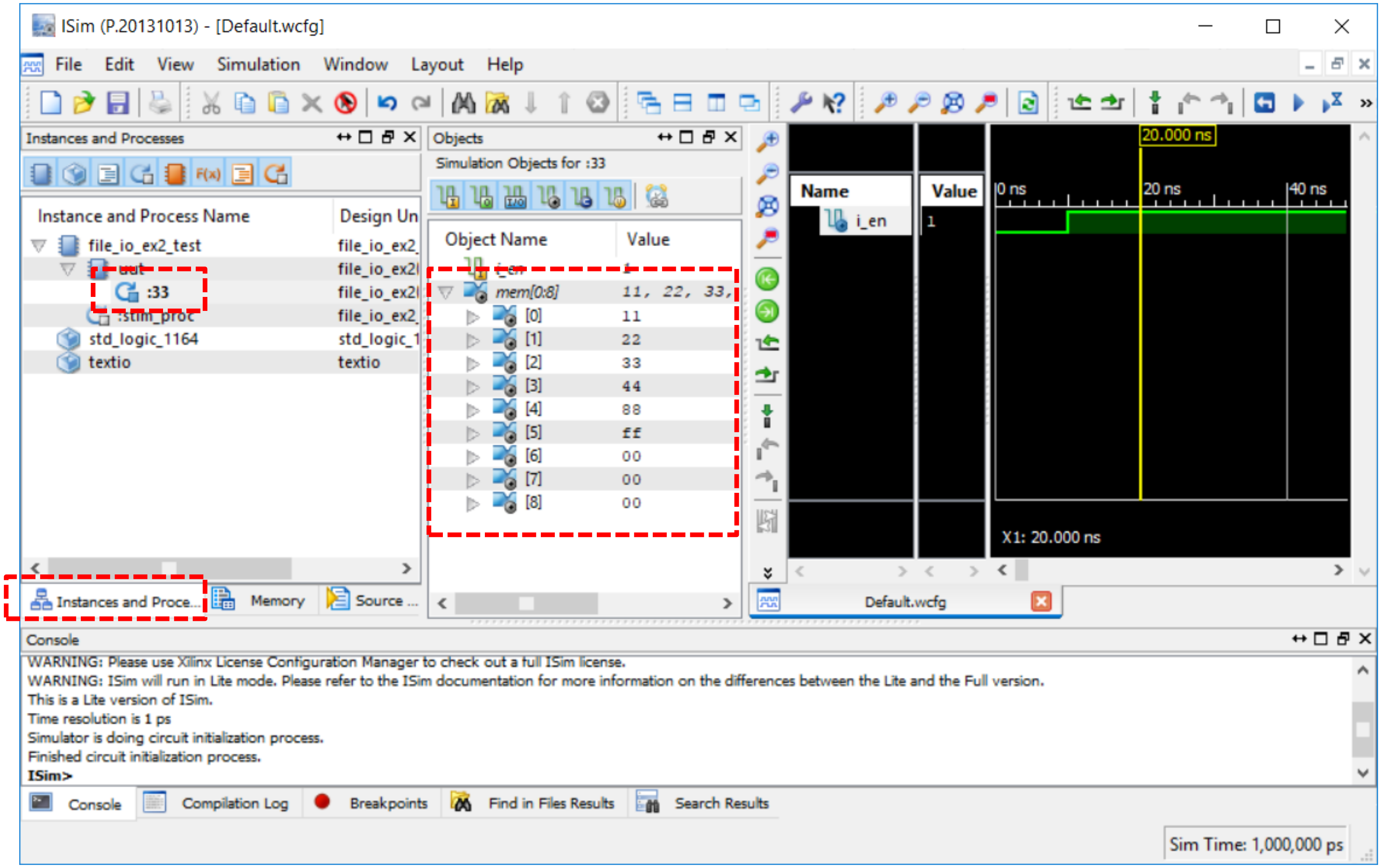
- File I/O function



```vhdl
impure function init_buf(FileName : in string) return buf8x8 is
   constant LINE_NUM : integer := 10;
   file fp: text;
   variable temp_mem : buf8x8 := (others => x"00");
   variable line_cache : line;
   variable byte_cache : bit_vector (7 downto 0) := x"00";
begin
   file_open(fp, FileName, read_mode);
   for i in 0 to LINE_NUM loop
      if endfile(fp) then
         exit;
      else
         readline(fp, line_cache);
         read(line_cache, byte_cache);
         temp_mem(i) := to_stdlogicvector(byte_cache);
      end if;
   end loop;
   file_close(fp);
   return temp_mem;
end function;
```

Example (5)

- Simulation results



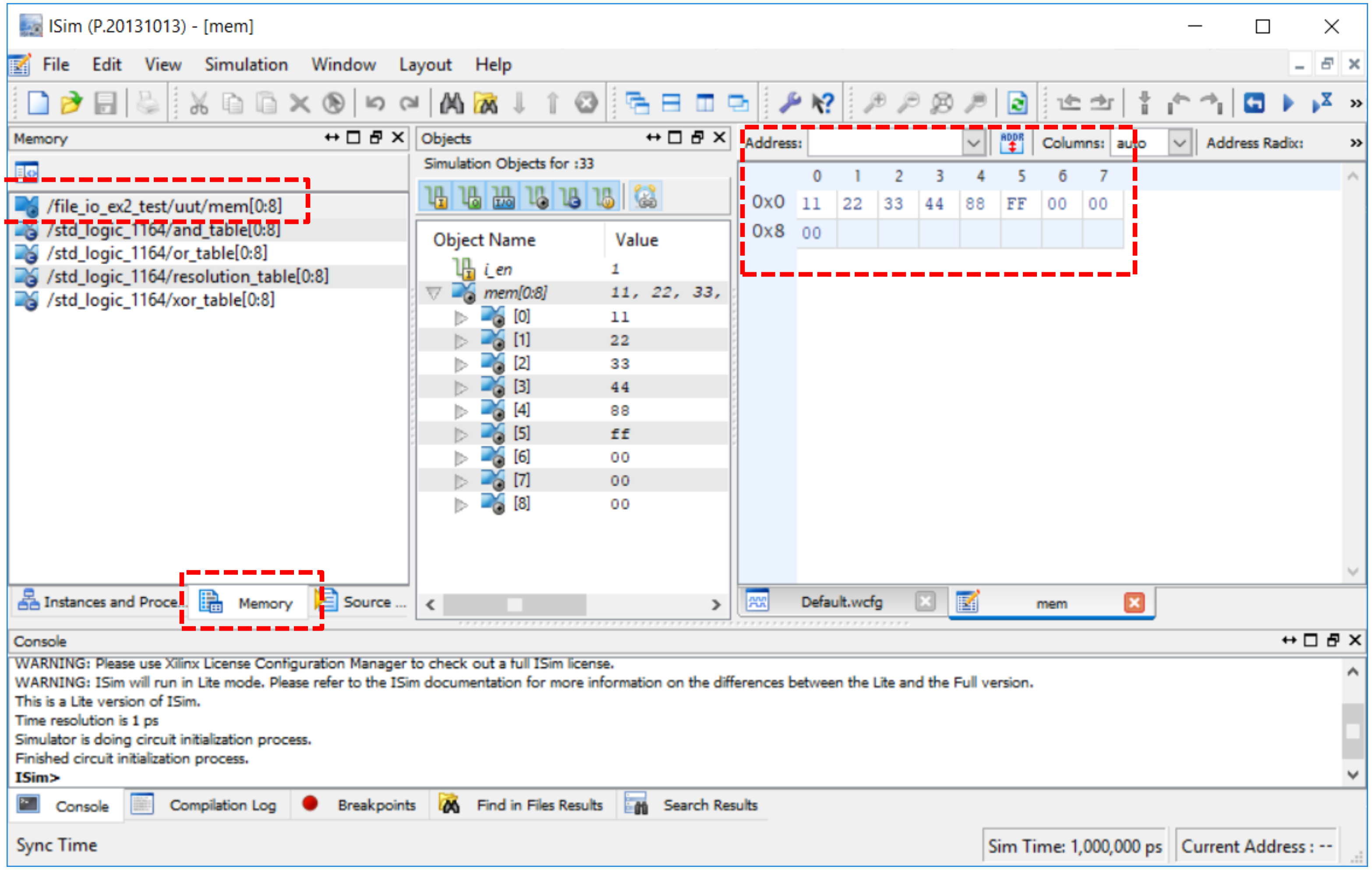
Instance and Process Name	Design Un
file_io_ex2_test	file_io_ex2_
uut	file_io_ex2
:33	file_io_ex2
:stim_proc	file_io_ex2_
std_logic_1164	std_logic_1
textio	textio

Object Name	Value
i_en	1
mem[0:8]	11, 22, 33,
[0]	11
[1]	22
[2]	33
[3]	44
[4]	88
[5]	ff
[6]	00
[7]	00
[8]	00

```
WARNING: Please use Xilinx License Configuration Manager to check out a full ISim license.
WARNING: ISim will run in Lite mode. Please refer to the ISim documentation for more information on the differences between the Lite and the Full version.
This is a Lite version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
```

Example (6)

- Simulation results



Memory:

- /file_io_ex2_test/uut/mem[0:8]
- /std_logic_1164/and_table[0:8]
- /std_logic_1164/or_table[0:8]
- /std_logic_1164/resolution_table[0:8]
- /std_logic_1164/xor_table[0:8]

Object Name	Value
i_en	1
mem[0:8]	11, 22, 33,
[0]	11
[1]	22
[2]	33
[3]	44
[4]	88
[5]	ff
[6]	00
[7]	00
[8]	00

	0	1	2	3	4	5	6	7
0x0	11	22	33	44	88	FF	00	00
0x8	00							

```
WARNING: Please use Xilinx License Configuration Manager to check out a full ISim license.
WARNING: ISim will run in Lite mode. Please refer to the ISim documentation for more information on the differences between the Lite and the Full version.
This is a Lite version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
```

Sync Time | Sim Time: 1,000,000 ps | Current Address : --