

Hardware Modeling [VU] (191.011)

– WS25 –

Advanced Testbenches

Florian Huemer & Sebastian Wiedemann & Dylan Baumann

WS 2025/26

Motivation

HWMod
WS25

Adv. TB
Motivation
File I/O
Random Testing
std.env

■ More powerful testbenches?

Motivation

HWMod
WS25

Adv. TB
Motivation
File I/O
Random Testing
std.env

■ More powerful testbenches?

Motivation

HWMod
WS25

Adv. TB
Motivation
File I/O
Random Testing
std.env

- More powerful testbenches?
 - Modern designs can be highly complex (e.g., hundreds of I/O pins)

Motivation

HWMod
WS25

Adv. TB
Motivation
File I/O
Random Testing
std.env

■ More powerful testbenches?

- Modern designs can be highly complex (e.g., hundreds of I/O pins)
- Manually generating and applying stimuli infeasible / impossible

Motivation

HWMod
WS25

Adv. TB
Motivation
File I/O
Random Testing
std.env

■ More powerful testbenches?

- Modern designs can be highly complex (e.g., hundreds of I/O pins)
- Manually generating and applying stimuli infeasible / impossible
- The per-transistor cost of testing is higher than that of designing

Motivation

HWMod
WS25

Adv. TB
Motivation
File I/O
Random Testing
std.env

- More powerful testbenches?
 - Modern designs can be highly complex (e.g., hundreds of I/O pins)
 - Manually generating and applying stimuli infeasible / impossible
 - The per-transistor cost of testing is higher than that of designing
- More powerful testbenches and automation!

Motivation

HWMod
WS25

Adv. TB
Motivation
File I/O
Random Testing
std.env

- More powerful testbenches?
 - Modern designs can be highly complex (e.g., hundreds of I/O pins)
 - Manually generating and applying stimuli infeasible / impossible
 - The per-transistor cost of testing is higher than that of designing
- More powerful testbenches and automation!
 - File I/O

Motivation

HWMod
WS25

Adv. TB
Motivation
File I/O
Random Testing
std.env

- More powerful testbenches?
 - Modern designs can be highly complex (e.g., hundreds of I/O pins)
 - Manually generating and applying stimuli infeasible / impossible
 - The per-transistor cost of testing is higher than that of designing
- More powerful testbenches and automation!
 - File I/O
 - Randomized testing

Motivation

HWMod
WS25

Adv. TB
Motivation
File I/O
Random Testing
std.env

- More powerful testbenches?
 - Modern designs can be highly complex (e.g., hundreds of I/O pins)
 - Manually generating and applying stimuli infeasible / impossible
 - The per-transistor cost of testing is higher than that of designing
- More powerful testbenches and automation!
 - File I/O
 - Randomized testing
 - Frameworks and packages

Motivation

HWMod
WS25

Adv. TB
Motivation
File I/O
Random Testing
std.env

- More powerful testbenches?
 - Modern designs can be highly complex (e.g., hundreds of I/O pins)
 - Manually generating and applying stimuli infeasible / impossible
 - The per-transistor cost of testing is higher than that of designing
- More powerful testbenches and automation!
 - File I/O
 - Randomized testing
 - Frameworks and packages

covered in other courses

- Recall `access` and `file` types

Access Types

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

- Recall `access` and `file` types
- Objects can be created *dynamically* during simulation
 - Using so-called *allocators*

Access Types

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

- Recall `access` and `file` types
- Objects can be created *dynamically* during simulation
 - Using so-called *allocators*
 - No identifier referring to them

Access Types

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

- Recall `access` and `file` types
- Objects can be created *dynamically* during simulation
 - Using so-called *allocators*
 - No identifier referring to them
- Access types provide access to objects of certain type

```
type TYPE_NAME is access DESIGNATED_TYPE;
```

Access Types

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

- Recall `access` and `file` types
- Objects can be created *dynamically* during simulation
 - Using so-called *allocators*
 - No identifier referring to them
- Access types provide access to objects of certain type

```
type TYPE_NAME is access DESIGNATED_TYPE;
```

- Can only be used for `variable`

Access Types

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

- Recall `access` and `file` types
- Objects can be created *dynamically* during simulation
 - Using so-called *allocators*
 - No identifier referring to them
- Access types provide access to objects of certain type

```
type TYPE_NAME is access DESIGNATED_TYPE;
```

- Can only be used for `variable`
- Default value `null`; assigned using allocators

```
int_ptr := new integer;
```

Access Types

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

- Recall `access` and `file` types
- Objects can be created *dynamically* during simulation
 - Using so-called *allocators*
 - No identifier referring to them
- Access types provide access to objects of certain type

```
type TYPE_NAME is access DESIGNATED_TYPE;
```

- Can only be used for `variable`
- Default value `null`; assigned using allocators

```
int_ptr := new integer;
```
- Access to value of designated type object via `all`

```
int_ptr.all := 42; print(to_string(int_ptr.all));
```

Access Types

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

- Recall `access` and `file` types
- Objects can be created *dynamically* during simulation
 - Using so-called *allocators*
 - No identifier referring to them
- Access types provide access to objects of certain type

```
type TYPE_NAME is access DESIGNATED_TYPE;
```

- Can only be used for `variable`
- Default value `null`; assigned using allocators

```
int_ptr := new integer;
```
- Access to value of designated type object via `all`

```
int_ptr.all := 42; print(to_string(int_ptr.all));
```
- Similar to object references in Java and the `new` operator

- File types define objects representing files on the host system

```
type FILETYPE is file of TYPE_MARK;
```

- File types define objects representing files on the host system

```
type FILETYPE is file of TYPE_MARK;
```

- Value of file type object is sequence of values in file

- File types define objects representing files on the host system

```
type FILETYPE is file of TYPE_MARK;
```

- Value of file type object is sequence of values in file
- TYPE_MARK
 - Defines types of values in file
 - (unconstrained) scalar types, 1D-array of constrained subtype, fully constrained record type

- File types define objects representing files on the host system

```
type FILETYPE is file of TYPE_MARK;
```

- Value of file type object is sequence of values in file
- TYPE_MARK
 - Defines types of values in file
 - (unconstrained) scalar types, 1D-array of constrained subtype, fully constrained record type
- Implicitly defined subprograms for each file type `ft of tm`

File Operations

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

```
1 procedure file_open (
2     status: out file_open_status;
3     file f: ft;
4     external_name: in string;
5     open_kind: in file_open_kind := READ_MODE);
6
7 procedure file_close (file f: ft);
8
9 procedure read (file f: ft; value: out tm);
10
11 procedure write (file f: ft; value: in tm);
12
13 procedure flush (file f: ft);
14
15 function endfile (file f: ft) return boolean;
```

File Operations

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

```
1 procedure file_open (
2     status: out file_open_status;
3     file f: ft;
4     external_name: in string;
5     open_kind: in file_open_kind := READ_MODE);
6
7 procedure file_close (file f: ft);
8
9 procedure read (file f: ft; value: out tm);
10
11 procedure write (file f: ft; value: in tm);
12
13 procedure flush (file f: ft);
14
15 function endfile (file f: ft) return boolean;
```

opens file on host

File Operations

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

```
1 procedure file_open (
2     status: out file_open_status;
3     file f: ft;
4     external_name: in string;
5     open_kind: in file_open_kind := READ_MODE);
6
7 procedure file_close (file f: ft);
8
9 procedure read (file f: ft; value: out tm);
10
11 procedure write (file f: ft; value: in tm);
12
13 procedure flush (file f: ft);
14
15 function endfile (file f: ft) return boolean;
```

indicates result, optional

File Operations

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

```
1 procedure file_open (
2     status: out file_open_status;
3     file f: ft;
4     external_name: in string;
5     open_kind: in file_open_kind := READ_MODE);
6
7 procedure file_close (file f: ft);
8
9 procedure read (file f: ft; value: out tm);
10
11 procedure write (file f: ft; value: in tm);
12
13 procedure flush (file f: ft);
14
15 function endfile (file f: ft) return boolean;
```

associated to open file

File Operations

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

```
1 procedure file_open (
2     status: out file_open_status;
3     file f: ft;
4     external_name: in string;
5     open_kind: in file_open_kind := READ_MODE);
6
7 procedure file_close (file f: ft);
8
9 procedure read (file f: ft; value: out tm);
10
11 procedure write (file f: ft; value: in tm);
12
13 procedure flush (file f: ft);
14
15 function endfile (file f: ft) return boolean;
```

name of the host file

File Operations

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

```
1 procedure file_open (
2     status: out file_open_status;
3     file f: ft;
4     external_name: in string;
5     open_kind: in file_open_kind := READ_MODE);
6
7 procedure file_close (file f: ft);
8
9 procedure read (file f: ft; value: out tm);
10
11 procedure write (file f: ft; value: in tm);
12
13 procedure flush (file f: ft);
14
15 function endfile (file f: ft) return boolean;
```

READ_MODE, WRITE_MODE
APPEND_MODE
2019: READ_WRITE_MODE

File Operations

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

```
1 procedure file_open (
2     status: out file_open_status;
3     file f: ft;
4     external_name: in string;
5     open_kind: in file_open_kind := READ_MODE);
6
7 procedure file_close (file f: ft);
8
9 procedure read (file f: ft; value: out tm);
10
11 procedure write (file f: ft; value: in tm);
12
13 procedure flush (file f: ft);
14
15 function endfile (file f: ft) return boolean;
```

closes opened file

File Operations

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

```
1 procedure file_open (
2     status: out file_open_status;
3     file f: ft;
4     external_name: in string;
5     open_kind: in file_open_kind := READ_MODE);
6
7 procedure file_close (file f: ft);
8
9 procedure read (file f: ft; value: out tm);
10
11 procedure write (file f: ft; value: in tm);
12
13 procedure flush (file f: ft);
14
15 function endfile (file f: ft) return boolean;
```

closes opened file

Always close opened files!

File Operations

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

```
1 procedure file_open (
2     status: out file_open_status;
3     file f: ft;
4     external_name: in string;
5     open_kind: in file_open_kind := READ_MODE);
6
7 procedure file_close (file f: ft);
8
9 procedure read (file f: ft; value: out tm);
10
11 procedure write (file f: ft; value: in tm);
12
13 procedure flush (file f: ft);
14
15 function endfile (file f: ft) return boolean;
```

reads next value

File Operations

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

```
1 procedure file_open (
2     status: out file_open_status;
3     file f: ft;
4     external_name: in string;
5     open_kind: in file_open_kind := READ_MODE);
6
7 procedure file_close (file f: ft);
8
9 procedure read (file f: ft; value: out tm);
10
11 procedure write (file f: ft; value: in tm);
12
13 procedure flush (file f: ft);
14
15 function endfile (file f: ft) return boolean;
```

appends value

File Operations

HWMod
WS25

Adv. TB

Motivation

File I/O

Access Types

File Types

TextIO

Examples

Random Testing

std.env

```
1 procedure file_open (
2     status: out file_open_status;
3     file f: ft;
4     external_name: in string;
5     open_kind: in file_open_kind := READ_MODE);
6
7 procedure file_close (file f: ft);
8
9 procedure read (file f: ft; value: out tm);
10
11 procedure write (file f: ft; value: in tm);
12
13 procedure flush (file f: ft);
14
15 function endfile (file f: ft) return boolean;
```

ensures buffered writes
are actually carried out

File Operations

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

```
1 procedure file_open (
2     status: out file_open_status;
3     file f: ft;
4     external_name: in string;
5     open_kind: in file_open_kind := READ_MODE);
6
7 procedure file_close (file f: ft);
8
9 procedure read (file f: ft; value: out tm);
10
11 procedure write (file f: ft; value: in tm);
12
13 procedure flush (file f: ft);
14
15 function endfile (file f: ft) return boolean;
```

returns true if read
can read another value

- Types and subprograms for formatted operations on text files

Text IO Package

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

- Types and subprograms for formatted operations on text files
- Revolves around two new types

```
type line is access string;  
type text is file of string;
```

IEEE SA
OPEN

Text IO Package

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

- Types and subprograms for formatted operations on text files
- Revolves around two new types

IEEE SA
OPEN

```
type line is access string; -- dynamically resizable buffer
type text is file of string;
```

Text IO Package

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

- Types and subprograms for formatted operations on text files
- Revolves around two new types

IEEE SA
OPEN

```
type line is access string; -- dynamically resizable buffer
type text is file of string; -- text-file type
```

Text IO Package

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

- Types and subprograms for formatted operations on **text files**
- Revolves around two new types

IEEE SA
OPEN

```
type line is access string; -- dynamically resizable buffer
type text is file of string; -- text-file type
```

Text IO Package

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

- Types and subprograms for formatted operations on text files
- Revolves around two new types IEEE SA
OPEN

```
type line is access string; -- dynamically resizable buffer
type text is file of string; -- text-file type
```

- Subprograms for formatted manipulation of line buffers

```
procedure read(l: inout line; value: out <type>; good: out boolean);
procedure write(l: inout line; value: in <type>;
                justified: in side:=right; field: in width:=0);
```

Text IO Package

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

- Types and subprograms for formatted operations on text files
- Revolves around two new types IEEE SA
OPEN

```
type line is access string; -- dynamically resizable buffer
type text is file of string; -- text-file type
```

- Subprograms for formatted manipulation of line buffers

```
procedure read(l: inout line; value: out <type>; good: out boolean);
procedure write(l: inout line; value: in <type>;
                justified: in side:=right; field: in width:=0);
```

Text IO Package

HWMOD
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

- Types and subprograms for formatted operations on text files
- Revolves around two new types IEEE SA
OPEN

```
type line is access string; -- dynamically resizable buffer
type text is file of string; -- text-file type
```

- Subprograms for formatted manipulation of line buffers

```
procedure read(l: inout line; value: out <type>; good: out boolean);
procedure write(l: inout line; value: in <type>;
                justified: in side:=right; field: in width:=0);
```

Text IO Package

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

- Types and subprograms for formatted operations on text files
- Revolves around two new types IEEE SA
OPEN

```
type line is access string; -- dynamically resizable buffer
type text is file of string; -- text-file type
```

- Subprograms for formatted manipulation of line buffers

```
procedure read(l: inout line; value: out <type>; good: out boolean);
procedure write(l: inout line; value: in <type>;
                justified: in side:=right; field: in width:=0);
```

Text IO Package

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

- Types and subprograms for formatted operations on text files
- Revolves around two new types IEEE SA
OPEN

```
type line is access string; -- dynamically resizable buffer
type text is file of string; -- text-file type
```

- Subprograms for formatted manipulation of line buffers

```
procedure read(l: inout line; value: out <type>; good: out boolean);
procedure write(l: inout line; value: in <type>;
                justified: in side:=right; field: in width:=0);
```

Text IO Package

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

- Types and subprograms for formatted operations on text files
- Revolves around two new types IEEE SA
OPEN

```
type line is access string; -- dynamically resizable buffer
type text is file of string; -- text-file type
```

- Subprograms for formatted manipulation of line buffers

```
procedure read(l: inout line; value: out <type>; good: out boolean);
procedure write(l: inout line; value: in <type>;
                justified: in side:=right; field: in width:=0);
```

Text IO Package

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

- Types and subprograms for formatted operations on text files
- Revolves around two new types IEEE SA
OPEN

```
type line is access string; -- dynamically resizable buffer
type text is file of string; -- text-file type
```

- Subprograms for formatted manipulation of line buffers

```
procedure read(l: inout line; value: out <type>; good: out boolean);
procedure write(l: inout line; value: in <type>;
                justified: in side:=right; field: in width:=0);
```

Text IO Package

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

- Types and subprograms for formatted operations on text files
- Revolves around two new types IEEE SA
OPEN

```
type line is access string; -- dynamically resizable buffer
type text is file of string; -- text-file type
```

- Subprograms for formatted manipulation of line buffers

```
procedure read(l: inout line; value: out <type>; good: out boolean);
procedure write(l: inout line; value: in <type>;
                justified: in side:=right; field: in width:=0);
```

- Subprograms for reading/writing line buffers to file

```
procedure readline(file f: text; l: inout line);
procedure writeline(file f: text; l: inout line);
```

Text IO Package

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

- Types and subprograms for formatted operations on text files
- Revolves around two new types IEEE SA
OPEN

```
type line is access string; -- dynamically resizable buffer
type text is file of string; -- text-file type
```

- Subprograms for formatted manipulation of line buffers

```
procedure read(l: inout line; value: out <type>; good: out boolean);
procedure write(l: inout line; value: in <type>;
                justified: in side:=right; field: in width:=0);
```

- Subprograms for reading/writing line buffers to file

```
procedure readline(file f: text; l: inout line);
procedure writeline(file f: text; l: inout line);
```

- Further procedures [BINARY|OCTAL|HEX]_[READ|WRITE] for multiple types (e.g., `bit_vector`, `std_[u]logic[_vector]`, `[un]signed`)

Read from file

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

```
1 [...]
2 use std.textio.all;
3 [...]
4 begin
5   main: process is
6     file f : text open READ_MODE is "data.txt";
7     variable l : line;
8     variable x : std_ulogic_vector(7 downto 0);
9   begin
10     while not endfile(f) loop
11       readline(f, l);
12       hex_read(l, x);
13       report to_string(x);
14     end loop;
15     file_close(f);
16     wait;
17   end process;
```

```
1 00
2 11
3 AA
```

Read from file

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

```
1 [...]
2 use std.textio.all;
3 [...]
4 begin
5   main: process is
6     file f : text open READ_MODE is "data.txt";
7     variable l : line;
8     variable x : std_ulogic_vector(7 downto 0);
9   begin
10     while not endfile(f) loop
11       readline(f, l);
12       hex_read(l, x);
13       report to_string(x);
14     end loop;
15     file_close(f);
16     wait;
17   end process;
```

```
1 00
2 11
3 AA
```

Read from file

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

```
1 [...]
2 use std.textio.all;
3 [...]
4 begin
5   main: process is
6     file f : text open READ_MODE is "data.txt";
7     variable l : line;
8     variable x : std_ulogic_vector(7 downto 0);
9   begin
10     while not endfile(f) loop
11       readline(f, l);
12       hex_read(l, x);
13       report to_string(x);
14     end loop;
15     file_close(f);
16     wait;
17   end process;
```

```
1 00
2 11
3 AA
```

Read from file

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

```
1 [...]
2 use std.textio.all;
3 [...]
4 begin
5   main: process is
6     file f : text open READ_MODE is "data.txt";
7     variable l : line;
8     variable x : std_ulogic_vector(7 downto 0);
9   begin
10     while not endfile(f) loop
11       readline(f, l);
12       hex_read(l, x);
13       report to_string(x);
14     end loop;
15     file_close(f);
16     wait;
17   end process;
```

```
1 00
2 11
3 AA
```

Read from file

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

```
1 [...]
2 use std.textio.all;
3 [...]
4 begin
5   main: process is
6     file f : text open READ_MODE is "data.txt";
7     variable l : line;
8     variable x : std_ulogic_vector(7 downto 0);
9   begin
10     while not endfile(f) loop
11       readline(f, l);
12       hex_read(l, x);
13       report to_string(x);
14     end loop;
15     file_close(f);
16     wait;
17   end process;
```

```
1 00
2 11
3 AA
```

Read from file

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

```
1 [...]
2 use std.textio.all;
3 [...]
4 begin
5   main: process is
6     file f : text open READ_MODE is "data.txt";
7     variable l : line;
8     variable x : std_ulogic_vector(7 downto 0);
9   begin
10     while not endfile(f) loop
11       readline(f, l);
12       hex_read(l, x);
13       report to_string(x);
14     end loop;
15     file_close(f);
16     wait;
17   end process;
```

```
1 00
2 11
3 AA
```

Read from file

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

```
1 [...]
2 use std.textio.all;
3 [...]
4 begin
5   main: process is
6     file f : text open READ_MODE is "data.txt";
7     variable l : line;
8     variable x : std_ulogic_vector(7 downto 0);
9   begin
10     while not endfile(f) loop
11       readline(f, l);
12       hex_read(l, x);
13       report to_string(x);
14     end loop;
15     file_close(f);
16     wait;
17   end process;
```

```
1 00
2 11
3 AA
```

```
[...]: 00000000
[...]: 00010001
[...]: 10101010
```

Example: VHDLDraw show

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

```
1 procedure show(filename : string) is
2     file f_img : text;
3     variable img_line : line;
4     [...] -- variables for color (r,g,b), width and height
5 begin
6     file_open(f_img, filename, WRITE_MODE);
7     swrite(img_line, "P3"); -- "string_write", c.f. standard
8     writeline(f_img, img_line);
9     [...] -- further writes for the image header
10    for y in 0 to height-1 loop
11        for x in 0 to width-1 loop
12            c := frame(y, x);
13            [...] -- set color variables r, g, b
14            if x /= 0 then
15                swrite(img_line, " ");
16            end if;
17            swrite(img_line, to_string(r) & " " & to_string(g) & " " & to_string(b));
18        end loop;
19        writeline(f_img, img_line);
20    end loop;
21    file_close(f_img);
22 end procedure;
```

Example: VHDLDraw show

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

```
1 procedure show(filename : string) is
2     file f_img : text;
3     variable img_line : line;
4     [...] -- variables for color (r,g,b), width and height
5 begin
6     file_open(f_img, filename, WRITE_MODE);
7     swrite(img_line, "P3"); -- "string_write", c.f. standard
8     writeline(f_img, img_line);
9     [...] -- further writes for the image header
10    for y in 0 to height-1 loop
11        for x in 0 to width-1 loop
12            c := frame(y, x);
13            [...] -- set color variables r, g, b
14            if x /= 0 then
15                swrite(img_line, " ");
16            end if;
17            swrite(img_line, to_string(r) & " " & to_string(g) & " " & to_string(b));
18        end loop;
19        writeline(f_img, img_line);
20    end loop;
21    file_close(f_img);
22 end procedure;
```

Example: VHDLDraw show

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

```
1 procedure show(filename : string) is
2     file f_img : text;
3     variable img_line : line;
4     [...] -- variables for color (r,g,b), width and height
5 begin
6     file_open(f_img, filename, WRITE_MODE);
7     swrite(img_line, "P3"); -- "string_write", c.f. standard
8     writeline(f_img, img_line);
9     [...] -- further writes for the image header
10    for y in 0 to height-1 loop
11        for x in 0 to width-1 loop
12            c := frame(y, x);
13            [...] -- set color variables r, g, b
14            if x /= 0 then
15                swrite(img_line, " ");
16            end if;
17            swrite(img_line, to_string(r) & " " & to_string(g) & " " & to_string(b));
18        end loop;
19        writeline(f_img, img_line);
20    end loop;
21    file_close(f_img);
22 end procedure;
```

Example: VHDLDraw show

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

```
1 procedure show(filename : string) is
2   file f_img : text;
3   variable img_line : line;
4   [...] -- variables for color (r,g,b), width and height
5 begin
6   file_open(f_img, filename, WRITE_MODE);
7   swrite(img_line, "P3"); -- "string_write", c.f. standard
8   writeline(f_img, img_line);
9   [...] -- further writes for the image header
10  for y in 0 to height-1 loop
11    for x in 0 to width-1 loop
12      c := frame(y, x);
13      [...] -- set color variables r, g, b
14      if x /= 0 then
15          swrite(img_line, " ");
16      end if;
17      swrite(img_line, to_string(r) & " " & to_string(g) & " " & to_string(b));
18    end loop;
19    writeline(f_img, img_line);
20  end loop;
21  file_close(f_img);
22 end procedure;
```

Example: VHDLDraw show

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

```
1 procedure show(filename : string) is
2     file f_img : text;
3     variable img_line : line;
4     [...] -- variables for color (r,g,b), width and height
5 begin
6     file_open(f_img, filename, WRITE_MODE);
7     swrite(img_line, "P3"); -- "string_write", c.f. standard
8     writeline(f_img, img_line);
9     [...] -- further writes for the image header
10    for y in 0 to height-1 loop
11        for x in 0 to width-1 loop
12            c := frame(y, x);
13            [...] -- set color variables r, g, b
14            if x /= 0 then
15                swrite(img_line, " ");
16            end if;
17            swrite(img_line, to_string(r) & " " & to_string(g) & " " & to_string(b));
18        end loop;
19        writeline(f_img, img_line);
20    end loop;
21    file_close(f_img);
22 end procedure;
```

Example: VHLDDraw show

HWMod
WS25

Adv. TB
Motivation
File I/O
Access Types
File Types
TextIO
Examples
Random Testing
std.env

```
1 procedure show(filename : string) is
2     file f_img : text;
3     variable img_line : line;
4     [...] -- variables for color (r,g,b), width and height
5 begin
6     file_open(f_img, filename, WRITE_MODE);
7     swrite(img_line, "P3"); -- "string_write", c.f. standard
8     writeline(f_img, img_line);
9     [...] -- further writes for the image header
10    for y in 0 to height-1 loop
11        for x in 0 to width-1 loop
12            c := frame(y, x);
13            [...] -- set color variables r, g, b
14            if x /= 0 then
15                swrite(img_line, " ");
16            end if;
17            swrite(img_line, to_string(r) & " " & to_string(g) & " " & to_string(b));
18        end loop;
19        writeline(f_img, img_line);
20    end loop;
21    file_close(f_img);
22 end procedure;
```

Random Testing - Introduction

HWMod
WS25

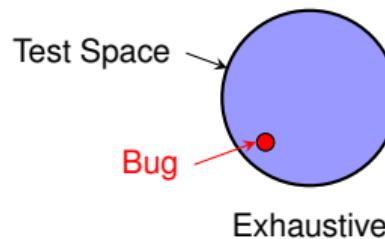
Adv. TB
Motivation
File I/O
Random Testing
std.env

Random Testing - Introduction

HWMod
WS25

Adv. TB
Motivation
File I/O
Random Testing
std.env

- Exhaustive testing infeasible (exponential in state and input space)

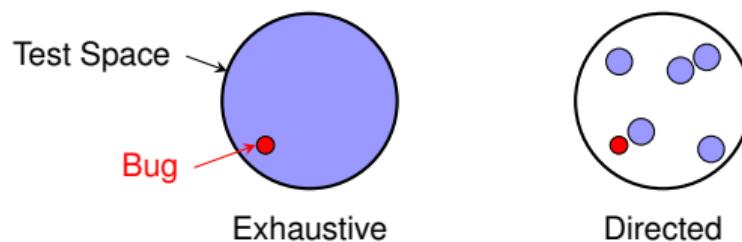


Random Testing - Introduction

HWMod
WS25

Adv. TB
Motivation
File I/O
Random Testing
std.env

- Exhaustive testing infeasible (exponential in state and input space)
- Directed Testing
 - Apply predefined stimuli
 - Requires solid knowledge of UUT
 - Only finds “anticipated” bugs

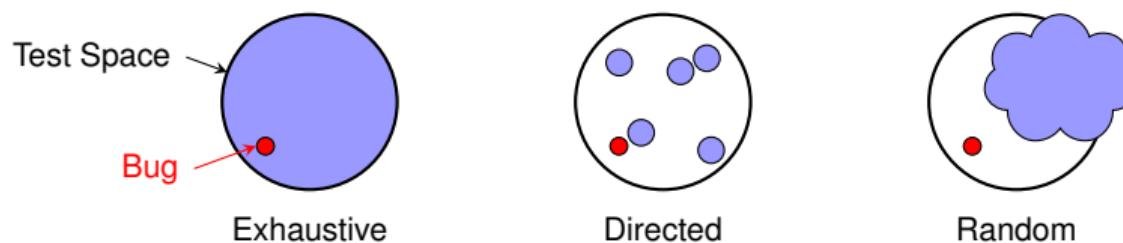


Random Testing - Introduction

HWMod
WS25

Adv. TB
Motivation
File I/O
Random Testing
std.env

- Exhaustive testing infeasible (exponential in state and input space)
- Directed Testing
 - Apply predefined stimuli
 - Requires solid knowledge of UUT
 - Only finds “anticipated” bugs
- Complementary technique: Random testing
 - Apply random stimuli to UUT
 - Finds non-anticipated bugs

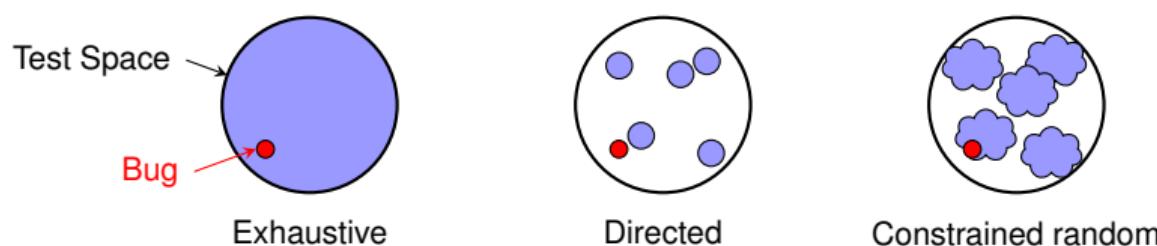


Random Testing - Introduction

HWMod
WS25

Adv. TB
Motivation
File I/O
Random Testing
std.env

- Exhaustive testing infeasible (exponential in state and input space)
- Directed Testing
 - Apply predefined stimuli
 - Requires solid knowledge of UUT
 - Only finds “anticipated” bugs
- Complementary technique: Random testing
 - Apply random stimuli to UUT
 - Finds non-anticipated bugs
 - Usually constrained



Random Testing in VHDL

HWMOD
WS25

Adv. TB
Motivation
File I/O
Random Testing
std.env

- uniform procedure of math_real package 

```
1 procedure uniform(variable seed1, seed2: inout positive;
2                      variable x : out real);
```

- Generates `real` in $0.0 < x < 1.0$

Random Testing in VHDL

HWMOD
WS25

Adv. TB
Motivation
File I/O
Random Testing
std.env

- **uniform procedure of math_real package** 

```
1 procedure uniform(variable seed1, seed2: inout positive;
2                     variable x : out real);
```

- Generates **real** in $0.0 < x < 1.0$
- Seeds allow repetition of generated sequence

Random Testing in VHDL

HWMod
WS25

Adv. TB
Motivation
File I/O
Random Testing
std.env

- **uniform procedure of math_real package** 

```
1 procedure uniform(variable seed1, seed2: inout positive;
2                      variable x : out real);
```

- Generates **real** in $0.0 < x < 1.0$
- Seeds allow repetition of generated sequence
- Seed values modified by the procedure!

Random Testing in VHDL

HWMOD
WS25

Adv. TB
Motivation
File I/O
Random Testing
std.env

- uniform procedure of math_real package 

```
1 procedure uniform(variable seed1, seed2: inout positive;
2                     variable x : out real);
```

- Generates `real` in $0.0 < x < 1.0$
- Seeds allow repetition of generated sequence
- Seed values modified by the procedure!
- Manual conversion to other types / ranges required

Random Testing in VHDL (cont'd)

HWMod
WS25

Adv. TB
Motivation
File I/O
Random Testing
std.env

■ Generation of random `std_uleogic` value

```
1 impure function rand_sul return std_uleogic is
2     variable rand : real;
3 begin
```

Random Testing in VHDL (cont'd)

HWMod
WS25

Adv. TB
Motivation
File I/O
Random Testing
std.env

■ Generation of random `std_uleogic` value

```
1 impure function rand_sul return std_uleogic is
2   variable rand : real;
3 begin
4   uniform(seed1, seed2, rand);
```

Random Testing in VHDL (cont'd)

HWMOD
WS25

Adv. TB
Motivation
File I/O
Random Testing
std.env

■ Generation of random `std_uleogic` value

```
1 impure function rand_sul return std_uleogic is
2   variable rand : real;
3 begin
4   uniform(seed1, seed2, rand);
5   if rand < 0.5 then
6     return '0';
7   end if;
8   return '1';
9 end function;
```

Random Testing in VHDL (cont'd)

HWMod
WS25

Adv. TB
Motivation
File I/O
Random Testing
std.env

■ Generation of random `std_uleogic` value

```
1 impure function rand_sul return std_uleogic is
2   variable rand : real;
3 begin
4   uniform(seed1, seed2, rand);
5   if rand < 0.5 then
6     return '0';
7   end if;
8   return '1';
9 end function;
```

■ Generation of `integer` range

```
1 impure function rand_int(start, stop : integer) return integer is
```

Random Testing in VHDL (cont'd)

HWMOD
WS25

Adv. TB
Motivation
File I/O
Random Testing
std.env

■ Generation of random `std_uleogic` value

```
1 impure function rand_sul return std_uleogic is
2   variable rand : real;
3 begin
4   uniform(seed1, seed2, rand);
5   if rand < 0.5 then
6     return '0';
7   end if;
8   return '1';
9 end function;
```

■ Generation of `integer` range

```
1 impure function rand_int(start, stop : integer) return integer is
2   variable rand : real;
3 begin
4   uniform(seed1, seed2, rand);
```

Random Testing in VHDL (cont'd)

HWMod
WS25

Adv. TB
Motivation
File I/O
Random Testing
std.env

■ Generation of random `std_uleogic` value

```
1 impure function rand_sul return std_uleogic is
2   variable rand : real;
3 begin
4   uniform(seed1, seed2, rand);
5   if rand < 0.5 then
6     return '0';
7   end if;
8   return '1';
9 end function;
```

■ Generation of `integer` range

```
1 impure function rand_int(start, stop : integer) return integer is
2   variable rand : real;
3 begin
4   uniform(seed1, seed2, rand);
5   return integer(rand * real(stop-start+1)+real(start)-0.5);
6 end function;
```

Random Testing in VHDL (cont'd)

HWMod
WS25

Adv. TB
Motivation
File I/O
Random Testing
std.env

■ Generation of random `std_uleogic` value

```
1 impure function rand_sul return std_uleogic is
2   variable rand : real;
3 begin
4   uniform(seed1, seed2, rand);
5   if rand < 0.5 then
6     return '0';
7   end if;
8   return '1';
9 end function;
```

■ Generation of `integer` range

```
1 impure function rand_int(start, stop : integer) return integer is
2   variable rand : real;
3 begin
4   uniform(seed1, seed2, rand);
5   return integer(rand * real(stop-start+1)+real(start)-0.5);
6 end function;
```

Random Testing in VHDL (cont'd)

HWMod
WS25

Adv. TB
Motivation
File I/O
Random Testing
std.env

■ Generation of random `std_uleogic` value

```
1 impure function rand_sul return std_uleogic is
2   variable rand : real;
3 begin
4   uniform(seed1, seed2, rand);
5   if rand < 0.5 then
6     return '0';
7   end if;
8   return '1';
9 end function;
```

■ Generation of `integer` range

```
1 impure function rand_int(start, stop : integer) return integer is
2   variable rand : real;
3 begin
4   uniform(seed1, seed2, rand);
5   return integer(rand * real(stop-start+1)+real(start)-0.5);
6 end function;
```

- VHDL-2008 defines `env` for interfacing between VHDL and host

Standard Environment Package

- VHDL-2008 defines `env` for interfacing between VHDL and host
- `stop` and `finish` procedures for simulation termination
 - `procedure stop;`

Standard Environment Package

- VHDL-2008 defines `env` for interfacing between VHDL and host
- `stop` and `finish` procedures for simulation termination
 - `procedure stop;`
- Can be used in main process to stop whole simulation

```
1 std.env.stop;           1 use std.env.all;
2 [...]                   2
3 stop;                  3
```

Standard Environment Package

- VHDL-2008 defines `env` for interfacing between VHDL and host
- `stop` and `finish` procedures for simulation termination
 - `procedure stop;`
- Can be used in main process to stop whole simulation

```
1 std.env.stop;           1 use std.env.all;
2 [...]                   2
3 stop;                  3
```

- Further functionality only introduced in VHDL-2019

Standard Environment Package

- VHDL-2008 defines `env` for interfacing between VHDL and host

- `stop` and `finish` procedures for simulation termination

```
procedure stop;
```

- Can be used in main process to stop whole simulation

```
1 std.env.stop;           1 use std.env.all;
2 [...]                   2
3 stop;                  3
```

- Further functionality only introduced in VHDL-2019

- Types and functions for getting and formatting real-world timestamps

Standard Environment Package

- VHDL-2008 defines `env` for interfacing between VHDL and host

- `stop` and `finish` procedures for simulation termination

```
procedure stop;
```

- Can be used in main process to stop whole simulation

```
1 std.env.stop;           1 use std.env.all;
2 [...]                  2
3 stop;                  3
```

- Further functionality only introduced in VHDL-2019

- Types and functions for getting and formatting real-world timestamps
- File system manipulations (e.g., create and delete directories)

Standard Environment Package

HWMod
WS25

Adv. TB
Motivation
File I/O
Random Testing
std.env

- VHDL-2008 defines `env` for interfacing between VHDL and host

- `stop` and `finish` procedures for simulation termination

```
procedure stop;
```

- Can be used in main process to stop whole simulation

```
1 std.env.stop;           1 use std.env.all;
2 [...]                   2
3 stop;                  3
```

- Further functionality only introduced in VHDL-2019

- Types and functions for getting and formatting real-world timestamps
- File system manipulations (e.g., create and delete directories)
- Simulation meta info (VHDL and tool version, tool name, name of file, etc.)

Lecture Complete!