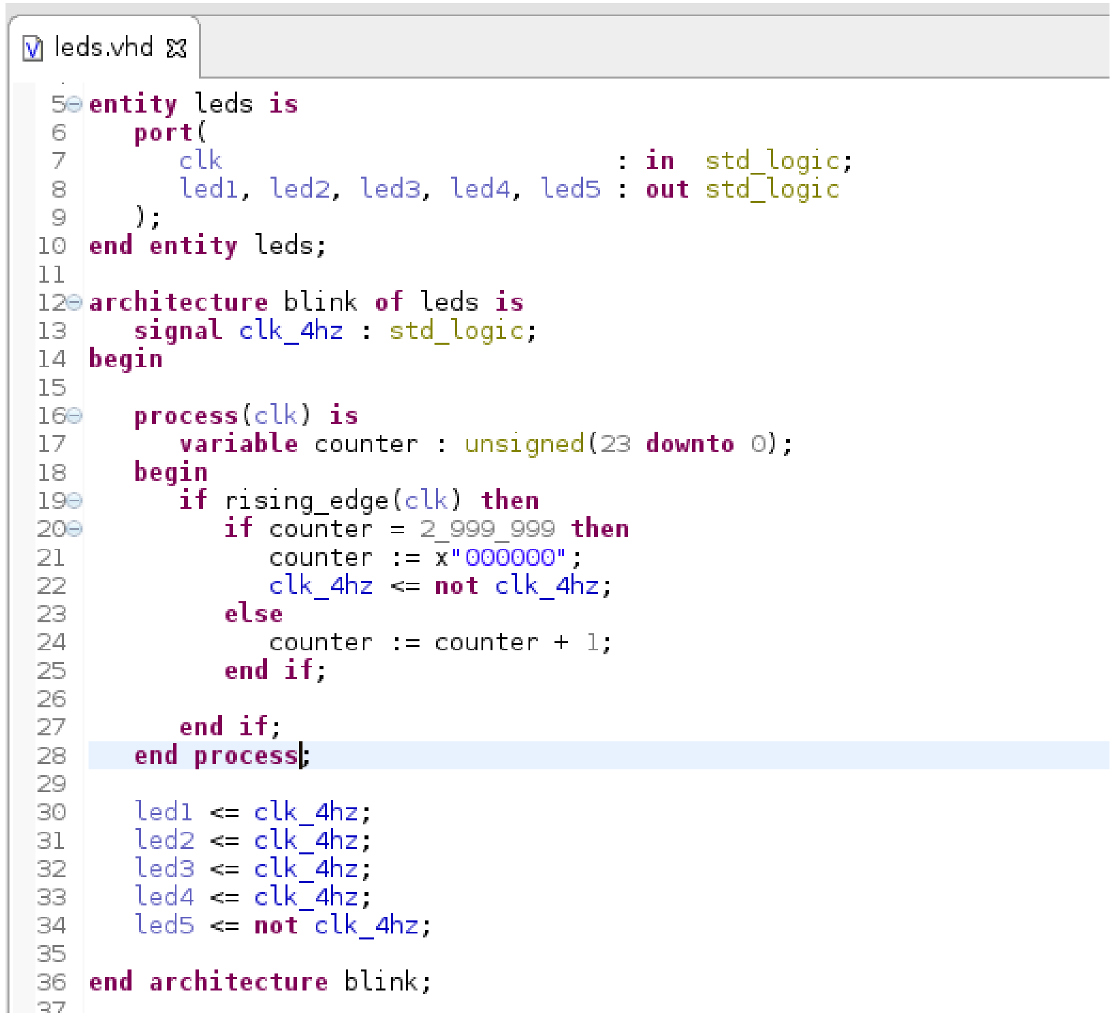
# VHDL 2018
# New and Noteworthy

Hendrik Eeckhaut (Sigasi)

Lieven Lemiengre (Sigasi)

# About me

- Chief R&D @ Sigasi
- Software guy
- Front-end compilers
  - IDEs: Editors, linting, documentation
  - VHDL and SystemVerilog
- VHDL Working Group


leds.vhd

```vhdl
entity leds is
   port(
      clk                          : in  std_logic;
      led1, led2, led3, led4, led5 : out std_logic
   );
end entity leds;

architecture blink of leds is
   signal clk_4hz : std_logic;
begin

   process(clk) is
      variable counter : unsigned(23 downto 0);
   begin
      if rising_edge(clk) then
         if counter = 2_999_999 then
            counter := x"000000";
            clk_4hz <= not clk_4hz;
         else
            counter := counter + 1;
         end if;

      end if;
   end process;

   led1 <= clk_4hz;
   led2 <= clk_4hz;
   led3 <= clk_4hz;
   led4 <= clk_4hz;
   led5 <= not clk_4hz;

end architecture blink;
```

## Overview

1. VHDL today
   - Short history
   - Advantages
2. VHDL 2018
   - Goals and design methodology
   - Key features: Interfaces and enhanced generics
   - Language ergonomics
   - Library development
3. Next steps

## 1. VHDL today

**History**

- 4 published standards
  - 1987: Initial standard
  - 1993: IEEE libraries (ieee.std_logic_1164, ieee.numeric_std, ...)
  - 2002: Protected types
  - 2008: Better generics, more libraries, PSL, ergonomics

- Current status?
  - Most popular in FPGA space
  - Some industries prefer VHDL

## 1. VHDL today

**Unique advantages (will add examples in final version)**

- Find bugs early
  - Strong typing
  - Easy for tools to analyze
- Library features > Language features
  - A growable language
- Portable designs
  - Deterministic simulation
  - Unambiguous, simple semantics

## 2. VHDL 2018

**Goals and design methodology**

- Need to evolve
  - Users: concise code, easy to use, safer
  - Library developers: better abstractions, less restrictions

- VHDL 20xx proposals survey
  1. Interfaces
  2. Language ergonomics
  3. Library features
  4. Bugfixes

## 2. VHDL 2018

- Evaluating > 100 proposals
  1. Backwards compatibility
  2. Feature complexity vs benefit
  3. Interaction with other features
  4. Implementation considerations

## 2. VHDL 2018

**Key features: Interfaces**

- How do you model an AXI, SPI, I2C, … interface?
  - Multiple elements with different directions or mode (in, out, …)
- Current approaches (examples will be added)
  - Repeat interface definition everywhere
  - `inout` record
  - Gaisler coding style: a record for each direction

# 2. VHDL 2018

## Key features: Interfaces

- Defining an interface

```
package p is
 type if_record is record
   data_in  : std_logic_vector;
   valid    : std_logic;
   data_out : std_logic_vector;
 end record if_record;

 view if_view of if_record is
   data_in          : in;
   valid, data_out : out;
 end view if_view;
end package p;
```

**Step 1: define a record with all interface elements**

**Step 2: define a mode view that assigns directions to each element of the record**

# 2. VHDL 2018

## Key features: Interfaces

- Using an interface

```
package p is
 type if_record is record
   data_in  : std_logic_vector;
   valid    : std_logic;
   data_out : std_logic_vector;
 end record if_record;

 view if_view of if_record is
   data_in          : in;
   valid, data_out  : out;
 end view if_view;
end package p;
```

**Step 3: use the interface in a port declaration**

```
use work.p.all;
entity e is
  port(
    clk : in                      std_logic;
    rst : in                      std_logic;
    if  : view if_view of if_record
  );
end entity e;
```

# 2. VHDL 2018

## Key features: Interfaces

- Using an interface

```
package p is
 type if_record is record
   data_in  : std_logic_vector;
   valid    : std_logic;
   data_out : std_logic_vector;
 end record if_record;

 view if_view of if_record is
   data_in          : in;
   valid, data_out  : out;
 end view if_view;
end package p;
```

**Step 3: use the interface in a port declaration**

```
use work.p.all;
entity e is
  port(
    clk : in                 std_logic;
    rst : in                 std_logic;
    if  : view if_view of if_record
  );
end entity e;
```

```
architecture a of top is
 signal clk, rst    : std_logic;
 signal if_inst     : if_record;
begin
 lbl : entity work.e port map(
   clk => clk;
   rst => rst;
   if  => if_inst;
 );
end architecture
```

**Step 4: Instantiation**

# 2. VHDL 2018

## Key features: Interfaces

- Refinements

```
package p is
 type if_record is record
   data_in  : std_logic_vector;
   valid    : std_logic;
   data_out : std_logic_vector;
 end record if_record;

 view if_view of if_record is
   data_in         : in;
   valid, data_out : out;
 end view if_view;
end package p;
```

```
use work.p.all;
entity e is
  port(
    clk : in                std_logic;
    rst : in                std_logic;
    if  : view if_view of if_record
  );
end entity e;
```

```
use work.p.all;
entity e is
  port(
    clk : in                std_logic;
    rst : in                std_logic;
    if  : view if_view
  );
end entity e;
```

The short form

## 2. VHDL 2018

### Key features: Interfaces

- Refinements

```
package p is
 type if_record is record
   data_in  : std_logic_vector;
   valid    : std_logic;
   data_out : std_logic_vector;
 end record if_record;

 view if_view of if_record is
   data_in         : in;
   valid, data_out : out;
 end view if_view;
end package p;
```

```
use work.p.all;
entity e is
  port(
    clk : in              std_logic;
    rst : in              std_logic;
    if  : view if_view of if_record
  );
end entity e;
```

**The converse**

```
out    -> in
in     -> out
inout  -> inout
buffer -> in
```

```
use work.p.all;
entity e is
  port(
    clk : in              std_logic;
    rst : in              std_logic;
    if  : view if_view'converse
  );
end entity e;
```

# 2. VHDL 2018

## Key features: Interfaces

- Refinements

```
package p is
 type if_record is record
   data_in   : std_logic_vector;
   valid     : std_logic;
   data_out  : std_logic_vector;
 end record if_record;

 view if_view of if_record is
   data_in         : in;
   valid, data_out : out;
 end view if_view;
end package p;
```

```
use work.p.all;
entity e is
  port(
    clk : in                   std_logic;
    rst : in                   std_logic;
    if  : view if_view of if_record
  );
end entity e;
```

**Constrained subtype**

```
entity e is
  generic(g : natural);
  port(
    clk : in                   std_logic;
    rst : in                   std_logic;
    if  : view if_view of if_record (
      data_in(0 to g), data_out(0 to g)
    )
  );
end entity e;
```

# 2. VHDL 2018

## Key features: Interfaces

- Refinements

```
package p is
 type if_record is record
   data_in  : std_logic_vector;
   valid    : std_logic;
   data_out : std_logic_vector;
 end record if_record;

 view if_view of if_record is
   data_in          : in;
   valid, data_out : out;
 end view if_view;
end package p;
```

```
use work.p.all;
entity e is
  port(
    clk : in                  std_logic;
    rst : in                  std_logic;
    if  : view if_view of if_record
  );
end entity e;
```

**Array of interfaces**

```
type if_record_vector is
  array (natural range <>) of if_record;

entity e is
  port(
    clk : in                      std_logic;
    rst : in                      std_logic;
    if  : view (if_view)
          of if_record_vector(1 to 10)
  );
end entity e;
```

# 2. VHDL 2018

## Key features: Interfaces

- Refinements

```
package p is
 type if_record is record
   data_in  : std_logic_vector;
   valid    : std_logic;
   data_out : std_logic_vector;
 end record if_record;

 view if_view of if_record is
   data_in         : in;
   valid, data_out : out;
 end view if_view;
end package p;
```

**Nested interfaces**

```
package p2 is
  type nested_if_record is record
    nested_field : if_record;
    valid        : std_logic;
  end record nested_record;


 view nested_if_view of nested_if_record is
   nested_field : view if_view;
   valid        : out;
 end view nested_if_view;
end package p2;
```

# 2. VHDL 2018

## Key features: Interfaces

- Refinements

```
package p is
 type if_record is record
   data_in  : std_logic_vector;
   valid    : std_logic;
   data_out : std_logic_vector;
 end record if_record;

 view if_view of if_record is
   data_in         : in;
   valid, data_out : out;
 end view if_view;
end package p;
```

**Subprograms**

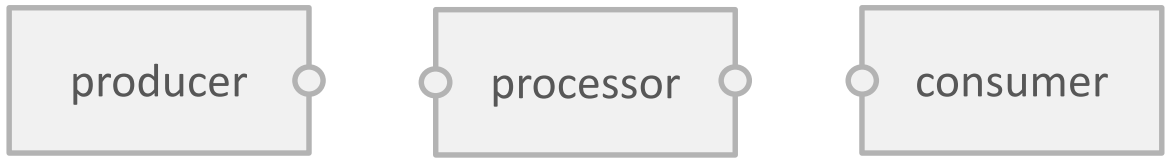
```
package subprograms is


  procedure p(signal b : view streaming_slave; ...);

  function f(signal b : view streaming_slave; ...)
    return std_logic;


end package subprograms;
```

## 2. VHDL 2018

### Key features: Interfaces

- Practical example

```
entity producer is
  port(clk, rst : in   std_logic;
       output   : view streaming_master);
end;
entity processor is
  port(clk, rst : in   std_logic;
       input    : view streaming_slave;
       output   : view streaming_master);
end;
entity consumer is
  port(clk, rst : in   std_logic;
       input    : view streaming_slave);
end;
```

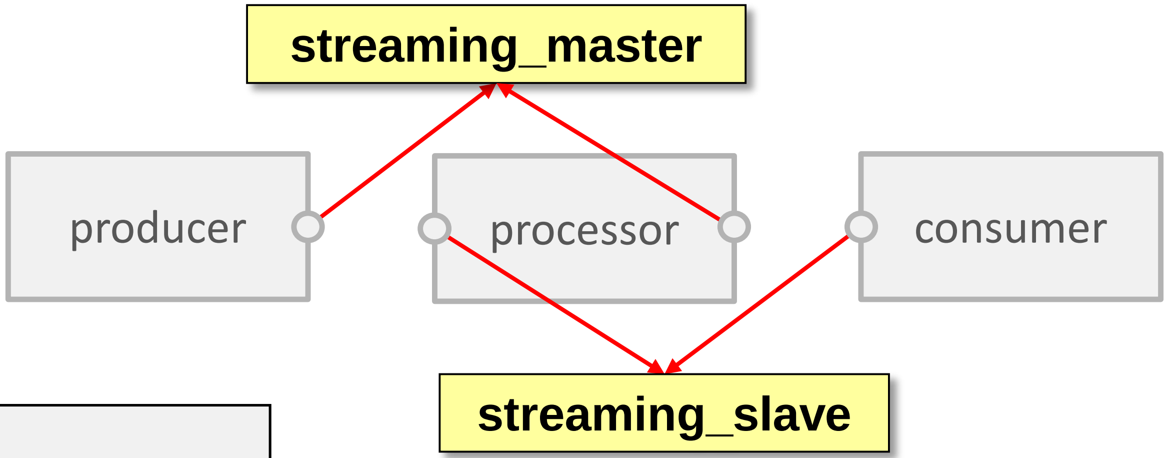


## VHDL 2018

### Key features: Interfaces

- Practical example

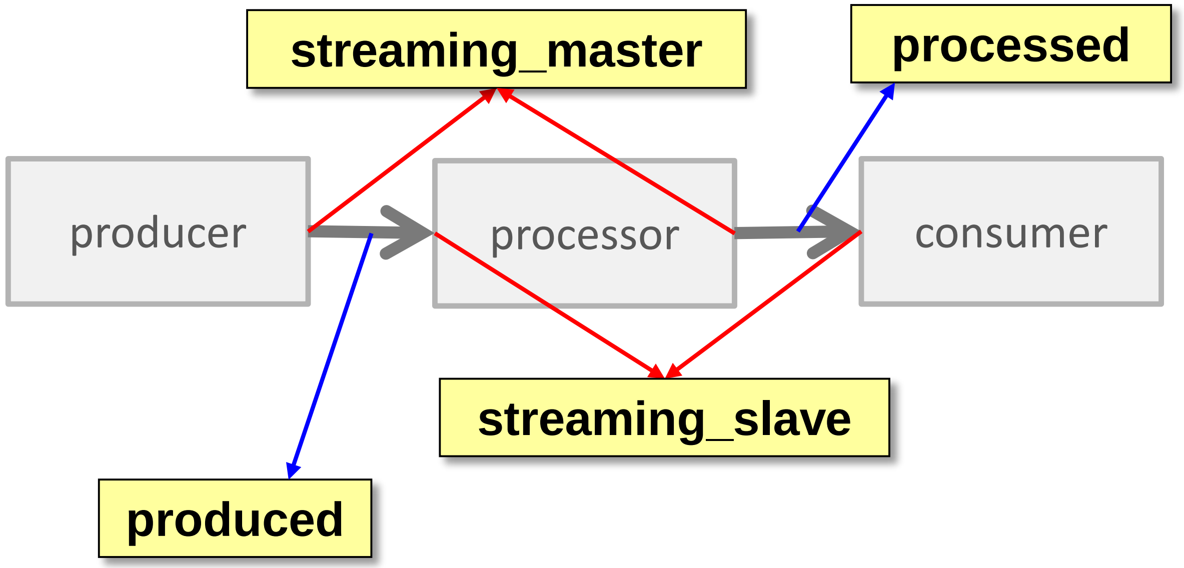
```
package interfaces is
  type streaming_bus is record
    valid : std_logic;
    data  : std_logic_vector(7 downto 0);
    ack   : std_logic;
  end record;

  view streaming_master of streaming_bus is
    valid, data : out;
    ack         : in;
  end view;

  alias streaming_slave is streaming_master’converse;
end package interfaces;
```



## 2. VHDL 2018

### Key features: Interfaces

- Practical example



```
architecture a of e is
  signal clk, rst : std_logic;
  signal produced, processed: streaming_bus;
begin
  prod : entity work.producer  port map(clk, rst, produced);
  proc : entity work.processor port map(clk, rst, produced, processed);
  cons : entity work.consumer  port map(clk, rst, processed);
end;
```

## 2. VHDL 2018

**Key features: Interfaces**

- Conclusion
  - Small change: Interface = Record + **Mode view** (+ Subtype)
  - Improved readability and maintainability
  - Very flexible
    - Nested interfaces
    - Arrays of interfaces
    - As a parameter on a subprogram
  - Concise syntax

# 2. VHDL 2018

## Key features: Enhanced generics

VHDL 2008

```
entity max is
  generic (
    type element_t;
    type array_t;
    function maximum(l : array_t)
      return element_t
  );
  port (
    clk    : in std_logic;
    input  : in array_t;
    output : out element_t
  );
end;
```

# 2. VHDL 2018

## Key features: Enhanced generics

VHDL 2008

```
entity max is
  generic (
    type element_t;
    type array_t;
    function maximum(l : array_t)
      return element_t
  );
  port (
    clk    : in std_logic;
    input  : in array_t;
    output : out element_t
  );
end;
```

```
architecture impl of max is
begin
  output <= maximum(input) when rising_edge(clk);
end
```

# 2. VHDL 2018

## Key features: Enhanced generics

VHDL 2008

```
entity max is
  generic (
    type element_t;
    type array_t;
    function maximum(l : array_t)
      return element_t
  );
  port (
    clk    : in std_logic;
    input  : in array_t;
    output : out element_t
  );
end;
```

VHDL 2018

```
entity max is
  generic (
    type element_t is <>;
    type index_t   is <>;
    type array_t   is array(index_t) of element_t
  );
  port (
    clk    : in  std_logic;
    input  : in  array_t;
    output : out element_t
  );
end;
```

# 2. VHDL 2018

## Key features: Enhanced generics

VHDL 2008

```
entity max is
  generic (
    type element_t;
    type array_t;
    function maximum(l : array_t)
      return element_t
  );
  port (
    clk    : in std_logic;
    input  : in array_t;
    output : out element_t
  );
end;
```

VHDL 2018

```
entity max is
  generic (


    type array_t is array(type is <>) of type is <>
  );
  port (
    clk    : in  std_logic;
    input  : in  array_t;
    output : out array_t’element_t
  );
end;
```

# 2. VHDL 2018

## Key features: Enhanced generics

**VHDL 2008**

```
entity max is
  generic (
    type element_t;
    type array_t;
    function maximum(l : array_t)
      return element_t
  );
  port (
    clk    : in std_logic;
    input  : in array_t;
    output : out element_t
  );
end;
```

**VHDL 2018**

```
entity max is






  port (
    clk    : in   std_logic;
    input  : in   type is
                  array(type is <>) of type is <>;
    output : out input’subtype’element_t
  );
end;
```

# VHDL 2018

## Key features: Enhanced generics

VHDL 2008

```
entity max is
  generic (
    type element_t;
    type array_t;
    function maximum(l : array_t)
      return element_t
  );
  port (
    clk    : in std_logic;
    input  : in array_t;
    output : out element_t
  );
end;
```

VHDL 2018

```
entity max is






  port (
    clk    : in  std_logic;
    input  : in  type is
                 array(type is <>) of type is <>;
    output : out input’subtype’element_t
  );
end;
```

```
maximizer : entity work.max port map(clk, input, output);
```

## 2. VHDL 2018

**Key features: Enhanced generics**

- Future libraries
  - Generic datastructures like List, Map, Set

- Conclusion
  - Aimed at creators of (verification) libraries
  - Reduced verbosity: declaration and instantiation
  - Improved type-safety

## 2. VHDL 2018

### Language ergonomics

- Ternary operator

```
y <= a xor b after (3 ns when FAST else 5 ns);

constant DELAY : time := 3 ns when FAST else 5 ns;

return a when condition else b;
```

- Conditional analysis

```
`if TOOL_VENDOR = "SIGASI" then
 attribute dont_touch of sig1 : signal is "true";
`endif
```

```
if env.tool_type = "SIMULATION" then ... endif;
```

# 2. VHDL 2018

## Language ergonomics

- Sequential declarative regions

```
p : process is
begin
  for i in some_vector’range then
    constant element : integer := some_vector(i);
  begin
    if element > CONST then
      variable result : integer;
    begin
      some_procedure(element, result);
      report result’image;
    end if;
  end for;
end process p;
```

# 2. VHDL 2018

## Language ergonomics

- Sequential declarative regions

```
p : process is
  variable combinatorial_or_register : unsigned(7 downto 0);
begin
  if rising_edge(clk) then
    variable only_combinatorial : unsigned(7 downto 0);
  begin
    ...
  end for;
end process p;
```

## 2. VHDL 2018

**Language ergonomics**

- New and improved APIs
  - Files
    - Read/write mode, random access, resize
  - Filesystem
    - Explore directories and files
  - Date and time
    - Query time
    - Minimal api to process time_record
    - Pretty printing

## 2. VHDL 2018

**Library development**

- Introspection
  - Convert any user defined data type in a known data type
  - Generic to_string function
- Asserts & PSL API
- Call path API
  - Get filename and location
  - Get stack trace
- Many removed restrictions

# 3. Next steps

- Finish the standardization
  - Finish editing the LRM
  - Balloting

- VHDL open source group
  - Create prototypes of future standardize libraries

- Next version
  - Continue work on interfaces
  - Finish what didn’t make the 2018 standard

## 3. Next steps

- Interested?
  - Follow the reflector!
  - Join the working group!
  - More information: http://www.eda-twiki.org/cgi-bin/view.cgi/P1076

- All 2018 proposals
  - http://www.eda-twiki.org/cgi-bin/view.cgi/P1076/VHDL2017

## Questions

?