SYSTEM MODELLING AND SYNTHESIS WITH HDL

DTEK0078

2023 Lecture 8

UNIVERSITY OF TURKU

Our research website: https://tiers.utu.fi/

Generics

UNIVERSITY OF TURKU

```
entity_declaration ::=
      entity identifier is
                [ generic ( generic_interface_list ) ; ]
                [ port ( port_interface_list ) ; ]
       end [ entity ] [ entity_simple_name ] ;

generic_interface_list ::=
[ constant ] identifier_list : [ in ] subtype_indication [ := static_expression ]
   {[ constant ] identifier_list : [ in ] subtype_indication [ := static_expression ]}
```

Generics

- Generics allow a design to be described so that its structure can be altered easily during the instantiation of the module

Up Counter

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity Counte3 is
port( clk   : in  std_logic;
      reset: in  std_logic;
      count: out std_logic_vector(3 downto 0));
end Counte3;

architecture behavioural of Counte3 is
  signal counting: std_logic_vector(3 downto 0);
begin
  process (clk,reset)
    begin
      if reset = '0' then
    counting <= "0000";
      elsif (rising_edge(clk)) then
    counting <= counting + 1;
      end if;
  end process;
  count <= counting;
end behavioural;
```

UNIVERSITY OF TURKU

Up Counter

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity Counte3 is
generic( w: natural := 4)
port( clk:   in std_logic;
      reset: in std_logic;
      count: out std_logic_vector(w-1 downto 0));
end Counte3;

architecture behavioural of Counte3 is
  signal counting: std_logic_vector(w-1 downto 0);
begin
  process (clk,reset)
    begin
      if reset = '0' then
    counting <= (others => 0);
      elsif (rising_edge(clk)) then
    counting <= counting + 1;
      end if;
    end process;
  count <= counting;
end behavioural;
```

UNIVERSITY OF TURKU

Port Mapping with Generics

```
entity Counte3 is
    generic( w: natural := 4)
    port( clk: in std_logic;
          reset: in std_logic;
          count: out std_logic_vector(w-1 downto 0));
    end Counte3;
```

```
c1:     entity work.counter (behavioural)
        generic map (16)
        port map (m_clk, m_reset, m_count1);
```

```
c2:     entity work.counter (behavioural)
        generic map (12)
        port map (m_clk, m_reset, m_count2);
```

```
c3:     entity work.counter (behavioural)
        generic map (w => 16)
        port map (m_clk, m_reset, m_count3);
```

UNIVERSITY OF TURKU

Port Mapping with Generics (VHDL-2008)

```
generic ( type T; constant init_val : T );
signal v : T := init_val;
```

```
entity Counte3 is
    generic( type data_type)
    port(   clk: in std_logic;
        reset: in std_logic;
      count: out data_type);
  end Counte3;
```

```
c1: entity work.counter (behavioural)
    generic map (data_type => std_logic_vector(3 downto 0))
    port map (m_clk, m_reset, m_count);
```

UNIVERSITY OF TURKU

Port Mapping with Generics (VHDL-2008)

NOTE

- To use types in generics requires one to be careful!
- For example, arithmetic operators are not defined for all the available types

```
entity counte3 is
    generic( type data_type )
    port( . . .
          Dout : out data_type);
    architecture
    . . .
    Dout <= Dout + 1;
    . . .
    end architecture;
```

UNIVERSITY OF TURKU


WE'RE GOING TO TRY SOMETHING CALLED AGILE PROGRAMMING.
THAT MEANS NO MORE PLANNING AND NO MORE DOCUMENTATION. JUST START WRITING CODE AND COMPLAINING.
I'M GLAD IT HAS A NAME.
THAT WAS YOUR TRAINING.
www.dilbert.com scottadams@aol.com
© Scott Adams, Inc./Dist. by UFS, Inc.

SYSTEM MODELLING AND SYNTHESIS WITH HDL

DTEK0078

2023 Lecture Coding Instructions

UNIVERSITY OF TURKU

Our research website: https://tiers.utu.fi/

Coding Instructions

UNIVERSITY OF TURKU

Purpose of VHDL Coding Instructions

- Prevent harmful or unpractical ways of coding
- Introduce a common, clear appearance for VHDL
- Increase readability for reviewing purposes
- Not to restrict creativity in any way
- Bad example:

```
A_37894 :process(xR,CK ,datai , DATAO )
BEGIN
if(XR =’1’)THEN DATAO<= "1010";end if;
if(CK’event) THEN if CK = ‘1’THEN
for ARGH in 0
to 3 Loop DATAO(ARGH) <=datai(ARGH);
end Loop;end if;
```

UNIVERSITY OF TURKU

Purpose of VHDL Coding Instructions

```
A_37894 :process(xR,CK ,datai , DATAO )
BEGIN
if(XR =’1’)THEN DATAO<= "1010";end if;
if(CK’event) THEN if CK = ‘1’THEN
for ARGH in 0
to 3 Loop DATAO(ARGH) <=datai(ARGH);
end Loop;end if;
```

```
A_37894 :process(xR, CK, datai, DATAO)
BEGIN
    if (XR =’1’) THEN DATAO <= "1010";
    end if;
    if (CK’event) THEN if CK = ‘1’THEN
        for ARGH in 0 to 3 Loop
                DATAO(ARGH) <= datai(ARGH);
        end loop;
    end if;
END process;
```

UNIVERSITY OF TURKU

File contents and naming

- One VHDL file should contain one entity and one architecture, file named as

  entityname.vhd

- Package name should be

  packagename_pkg.vhd

- Test bench name should be

  entityname_tb.vhd

A VHDL file and the entity it contains have the same name

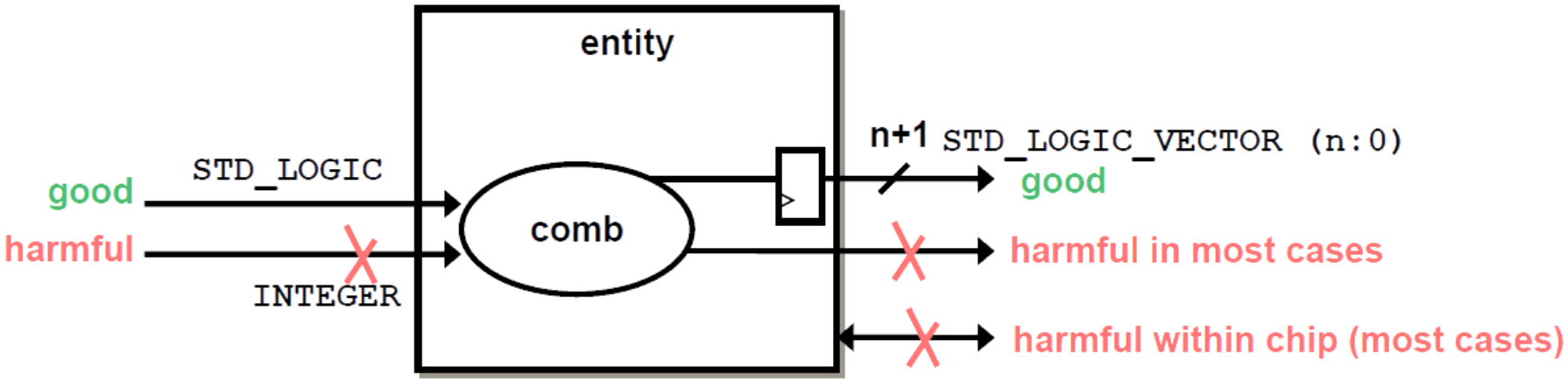
UNIVERSITY OF TURKU

Testbench

- Each entity requires at least one testbench
  - Design without a testbench is useless
- Prefer self-checking testbenches
  - Cannot assume that the verifier looks at the “magic spots” in waveform
  - (Occasionally, TB just generates few inputs to show the intended behaviour in common case)
- Informs whether the test was successful or not
- There can be several test benches for testing different aspects of the design

Entity ports

- Use only modes IN and OUT in the port
  - Signal names have corresponding post-fixes
- Use only signal types STD_LOGIC and STD_LOGIC_VECTOR in the ports
- Output of a block should always come directly from a register



UNIVERSITY OF TURKU

Sequential/synchronous process

- Sensitivity list of a synchronous process has always exactly two signals
  - Clock, rising edge used, named clk
  - Asynchronous reset, active low, named rst_n
- Signals that are assigned inside sync process, will become D-flip flops at synthesis
- Never give initial value to signal at declarative part
  - It is not supported by synthesis (but causes only a warning)

```
SIGNAL main_state_r : state_type := "11110000";
```

- **Assign values for control registers during reset**
- **Synchronous process is sensitive only to reset and clock**

Sequential/synchronous process

- Correct way of defining synchronous process:
  - Clock event is always to the rising edge
  - Assign values to control registers during reset

```
cmd_register : PROCESS (rst_n, clk)
BEGIN
   IF (rst_n = '0') THEN
      cmd_r <= (OTHERS => '0');
   ELSIF (clk'EVENT AND clk = '1') THEN
      cmd_r <= …;
   END IF;
END PROCESS cmd_register;
```

Combinatorial/asynchronous process

- An asynchronous process must have **all input signals in the sensitivity list**
  - If not, simulation is not correct
  - Top-3 mistake in VHDL
  - Input signals are on the right side of assignment or in conditions (if, for)
- If-clauses must be complete
  - Cover all cases, e.g. with else branch
  - All signals assigned in every branch
  - Otherwise, you'll get latches (which are evil)

Combinatorial/asynchronous process

- Same signal cannot be on both sides of assignment in combinatorial process
  - That would create combinatorial loop, i.e. malfunction

```
decode : PROCESS (cmd_r, bit_in, enable_in)
BEGIN
   IF (cmd_r = match_bits_c) THEN
      match_0       <= '1';
      IF (bit_in(1) = ‘1’and bit_in(0) = ‘0’) THEN
         match_both <= enable_in;
      ELSE
         match_both <= '0';
      END IF;
   ELSE --else branch needed to avoid latches
      match_0       <= '0';
      match_both    <= '0';
   END IF;
END PROCESS decode;
```

UNIVERSITY OF TURKU

Naming Conventions

- General register output `signalname_r`
- Combinatorial signal `signalname`
- Signal Between components `signalname_a_b`
- To multiple components `signalname_from_a`

- Input port `portname_in`
- Output port `portname_out`

- Constant `constantname_c`
- Generic `genericname_g`
- Variable `variablename_v`

Important is that the signal name clearly indicates its source and usage

UNIVERSITY OF TURKU

Clk and reset signals/inputs

- Active low reset is `rst_n`
  - Asynchronous set should not be used
- Clock signal `clk`
  - If there are more clocks the letters "clk“ appear in every one as a postfix

- When a signal ascends through hierarchy its name should remain the same. This is especially important for clock signal

Naming in general

- Descriptive, unambiguous names are very important
- Names are derived from English language
- Use only characters
  - alphabets ‘A’.. ‘Z’,‘a' .. ‘z',
  - numbers '0' .. '9‘ and underscore'_'.
  - First letter must be an alphabet
- Use enumeration for coding states in FSM
  - Do not use: s0, s1, a, b, state0, ...
  - Use: idle, wait_for_x, start_tx, read_mem, ...

- Average length of a good name is 8 to 16 characters

Signal types

- Direction of bits in
  `STD_LOGIC_VECTOR` **is always** `DOWNTO`

- Size of the vector should be parameterized
- Usually the least significant bit is numbered as zero (not one!):

```
SIGNAL data_r : STD_LOGIC_VECTOR(datawidth_g-1 DOWNTO 0);
```

- Use package numeric_std for arithmetic operations

Named signal mapping in instantiations

- Recommended to use named signal mapping, not ordered mapping

```
i_datamux : datamux
 PORT MAP (
    sel_in   => sel_ctrl_datamux,
    data_in  => data_ctrl_datamux,
    data_out => data_datamux_alu
    );
```

- This mapping works even if the declaration order of ports in entity changes

Use assertions

- Easier to find error location
- Checking always on, not just in testbench
- Assertions are not regarded by synthesis
- tools -> no extra logic

```
assert (we_in and re_in)=0
report ”Two enable signals must not active
at the same time”
severity warning;
```

- If condition is not true during simulation,
  - the report text, time stamp, component where it happened will be printed
- Ensure that your initial assumptions hold
  - e.g. data_width is multiple of 8 (bits)

Comment thoroughly

- Comment the intended function
  - Especially the **purpose** of signals
  - Not the VHDL syntax or semantics
  - Think of yourself reading the code after a decade
- A comment is indented like regular code
  - A comment is placed with the part of code to be commented.
- Be sure to update the comments if the code changes
  - Erroneous comment is more harmful than not having a comment at all

Code appearance: Aligning

- Align the colons and port types in the entity port:

```
ENTITY transmogrifier IS
   PORT (
      rst_n      : IN STD_LOGIC;
      clk        : IN STD_LOGIC;
      we_in      : IN  STD_LOGIC;
      cmd_0_in   : IN  STD_LOGIC_VECTOR(3-1 DOWNTO 0);
      data_in    : IN  STD_LOGIC_VECTOR(5-1 DOWNTO 0);
      valid_out  : OUT STD_LOGIC;
      result_out : OUT STD_LOGIC_VECTOR(6-1 DOWNTO 0)
);
END transmogrifier;
```

Code appearance: Aligning

- Align colons inside one signal declaration group:

```
 --control signals
 SIGNAL select     : STD_LOGIC_VECTOR (2-1 DOWNTO 0);
 SIGNAL cmd_r      : STD_LOGIC_VECTOR (32-1 DOWNTO 0);
 SIGNAL next_state : state_type;


--address and data signals
SIGNAL rd_addr  : STD_LOGIC_VECTOR (16-1 DOWNTO 0);
SIGNAL wr_addr  : STD_LOGIC_VECTOR (16-1 DOWNTO 0);
SIGNAL rd_data  : STD_LOGIC_VECTOR (32-1 DOWNTO 0);
```

Code appearance: Aligning

- Align the => in port maps:

```
i_pokerhand : pokerhand
PORT MAP (
   rst_n     => rst_n,
   clk       => clk,
   card_0_in => card (i),
   card_1_in => card (i),
   card_2_in => card (i),
   card_3_in => card (i),
   card_4_in => card (i),
   hand_out  => hand
   );
```

UNIVERSITY OF TURKU

Coding Instructions

1. Entity ports
   - Use only modes IN and OUT with names having suffixes _in or _out
2. Only types STD_LOGIC and STD_LOGIC_VECTOR
3. Use registered outputs
4. A VHDL file and the entity it contains have the same name
5. One entity+architecture per file
   - Every entity has a testbench
6. Synchronous process
   - always sensitive only to reset and clock
   - clock event is always to the rising edge
   - all control registers must be initialized in reset

Coding Instructions

7. Combinatorial process's sensitivity list includes all signals that are read in the process
   - Complete if-clauses must be used. Signals are assigned in every branch.
8. Use signal naming conventions
9. Indexes of STD_LOGIC_VECTOR are defined as DOWNTO
10. Use named signal mapping in component instantiation, never ordered mapping
11. Use assertions
12. Write enough comments

Coding Instructions

13. Every VHDL file starts with a header
14. Indent the code, keep lines shorter than 76 characters
15. Use descriptive names
16. Label every process and generate-clause
17. Clock is named clk and async. active-low reset rst_n
18. Intermediate signals define source and destination blocks
19. Instance is named according to entity name
20. Declare signals in consistent groups


UNIVERSITY
OF TURKU


WE'RE GOING TO TRY SOMETHING CALLED AGILE PROGRAMMING.
THAT MEANS NO MORE PLANNING AND NO MORE DOCUMENTATION. JUST START WRITING CODE AND COMPLAINING.
I'M GLAD IT HAS A NAME.
THAT WAS YOUR TRAINING.
www.dilbert.com scottadams@aol.com
© Scott Adams, Inc./Dist. by UFS, Inc.