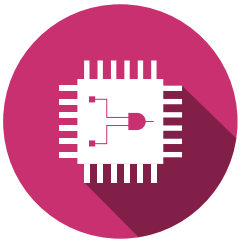


VHSIC Hardware Design Language (ex. VHDL)

Exercises embedded systems

1 Introduction

2 Structural description

3 Behavioral description

3.1 Assignments in a process

The following process contains two assignments. At the time when a signal of the sensitivity list changes, it is performed again.

```
1 process(in1, in2, in3)
2 begin
3   or12  <= in1 or in2 after 1 ns;
4   or123 <= or12 or in3 after 1 ns;
5 end process;
```

Listing 1: OR process

Draw the time evolution of the signals `or12` and `or123` in the time plot of the following figure 1.

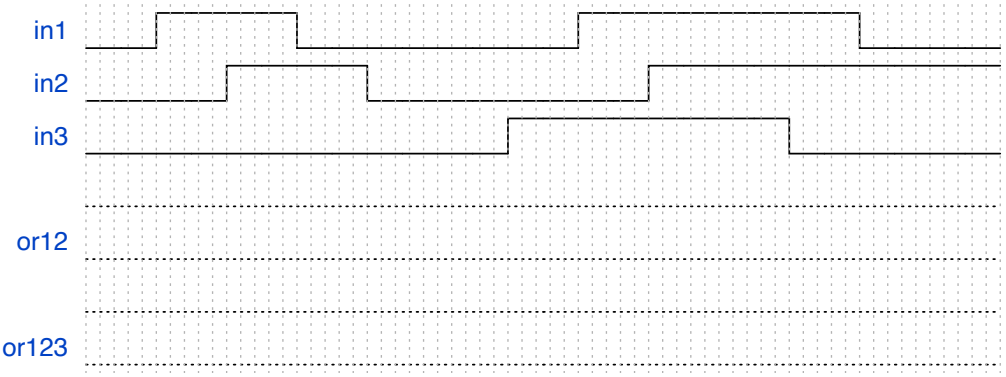
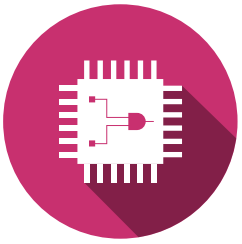


Figure 1: Chronogram

The vertical grid in the figure 1 shows a step of $1ns$.

3.2 Multiplexer

Write the architecture of the multiplexer whose entity is given hereafter.



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

entity multiplexer is
  port(
    sel    : in  std_ulogic;
    in0    : in  std_ulogic;
    in1    : in  std_ulogic;
    muxOut : out std_ulogic
  );
end multiplexer;
```

Listing 2: Multiplexer **entity**

The control signal `sel` can take 9 different values. Assign the value `'X'` to the output if the value of the output cannot be determined. Accept that if both inputs `in0` and `in1` are the same, then the output will take the same value even if the control signal has an undetermined value.

4 Common types and operations

4.1 Conversion from Unsigned to Signed

Write the architecture of the circuit that converts a **unsigned** to a **signed** whose **entity** is given hereafter.

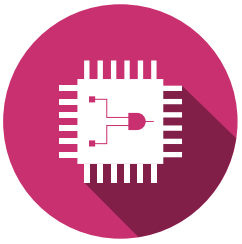
```vhdl
library ieee;
  use ieee.std_logic_1164.all;
  use ieee.numeric_std.all;

entity transformToSigned is
  generic(
    dataBitNb : positive := 8
  );
  port(
    signedIn  : in signed(dataBitNb-1 downto 0);
    unsignedOut : out unsigned(dataBitNb-1 downto 0)
  );
end transformToSigned;
```

Listing 3: Transform to signed **entity**

The range of the input signal is between 0 and $2^{dataBitNb} - 1$. Perform the transformation so that the value 0, the minimum value of the input signal, is converted to $-2^{dataBitNb-1}$, the minimum value of the output signal, and so that the maximum value $2^{dataBitNb} - 1$ of the input gives the maximum value $2^{dataBitNb-1} - 1$ of the output. In this way, the $2^{dataBitNb}$ value of the input will give 0.

4.2 Reduction of the range of a signal

Write the architecture of the circuit that reduces the range of a signal and whose entity is given hereafter.



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

entity rangeReducer is
  generic(
    signalBitNb     : positive := 16;
    saturationBitNb : positive := 8
  );
  port(
    inputSignal  : in  unsigned(signalBitNb-1 downto 0);
    outputSignal : out unsigned(signalBitNb-1 downto 0)
  );
end rangeReducer;
```

Listing 4: Range reducer `entity`

The operation reduces the maximum amplitude of the output signal to $2^{saturationBitNb} - 1$. This is created with saturation: if the input signal is greater than the maximum amplitude, the output is set to this maximum value.

Assume that `saturationBitNb` is always smaller than `signalBitNb`.

4.3 Counter with synchronous reset

Write the architecture of the counter with a synchonous reset signal, `signalBitNb`, whose `signalBitNb` is given hereafter.

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

entity counter is
  port(
    reset      : in  std_ulogic;
    clock      : in  std_ulogic;
    synchReset : in  std_ulogic;
    countOut   : out unsigned(7 downto 0)
  );
end counter;
```

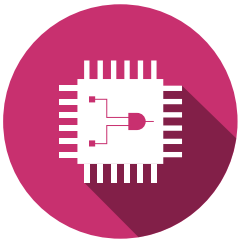
Listing 5: Counter `entity`

The VHDL language is very strict: it is only possible to assign an output, but not to re-read its value within the architecture.

4.4 Non-restartable delay

Write the architecture of the counter that generates a pulse of duration equal to one clock period, 100 clock periods after switching to `'1'` of the input signal. Its entity is given hereafter.

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

entity counter is
```



```
5    port(
6      reset    : in  std_ulogic;
7      clock    : in  std_ulogic;
8      trigger  : in  std_ulogic;
9      pulseOut : out std_ulogic
10   );
11 end counter;
```

Listing 6: Counter `entity`

If the input trigger switches to `'0'` and back to `'1'` before the output pulse has come, the circuit will ignore this extra pulse. This circuit is called a non-retriggerable one-shot.

Assume that the input signal has come back to `'0'` before the output pulse appears.

The VHDL language is very strict: it is only possible to assign an output, but not to re-read its value within the architecture.

5 Generic parameters

5.1 Restartable delay

Write entity and architecture of the counter that generates a pulse of duration equal to one clock period, `delay` clock periods after switching to `'1'` of the input signal. The delay `delay` is freely configurable between 1 and 100.

If the input `trigger` switches to `'0'` and back to `'1'` before the output pulse has come, the internal counter is restarted. This circuit is called a retriggerable one-shot.

Assume that the input signal has come back to `'0'` before the output pulse appears.

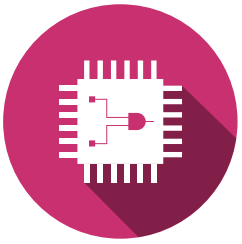
5.2 Adder

Write the architecture of the iterative adder whose entity is given hereafter:

```
1  library ieee;
2    use ieee.std_logic_1164.all;
3    use ieee.numeric_std.all;
4
5  entity iterativeAdder is
6    port(
7      Cin  : in  std_ulogic;
8      Ai   : in  std_ulogic;
9      Bi   : in  std_ulogic;
10     Si   : out std_ulogic;
11     Cout : out std_ulogic
12   );
13 end iterativeAdder;
```

Listing 7: iterativeAdder `entity`

If the input trigger switches to `'0'` and back to `'1'` before the output pulse has come, the circuit will ignore this extra pulse. This circuit is called a non-retriggerable one-shot.



Assume that the input signal has come back to `'0'` before the output pulse appears.

The VHDL language is very strict: it is only possible to assign an output, but not to re-read its value within the architecture.

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

entity adder is
  generic(
    bitNb : positive := 8
  );
  port(
    A        : in  unsigned (bitNb-1 downto 0);
    B        : in  unsigned (bitNb-1 downto 0);
    overflow : out std_ulogic;
    S        : out unsigned (bitNb-1 downto 0)
  );
end adder;
```

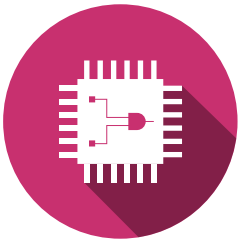
Listing 8: adder `entity`

6 Signals and variables

6.1 Counters

Enter the sequence of 3 counters of the following VHDL architecture.

```vhdl
architecture RTL of threeCounters is
  signal count1_int: unsigned(count1'range);
  signal count2_int: unsigned(count2'range);
begin

  cnt1: process(reset, clock)
  begin
    if reset = '1' then
      count1_int <= (others => '0');
    elsif rising_edge(clock) then
      if count1_int = 6 then
        count1_int <= (others => '0');
      else
        count1_int <= count1_int + 1;
      end if;
    end if;
  end process cnt1;

  count1 <= count1_int;

  cnt2: process(reset, clock)
  begin
    if reset = '1' then
      count2_int <= (others => '0');
    elsif rising_edge(clock) then
      count2_int <= count2_int + 1;
      if count2_int = 6 then
```



```
28          count2_int <= (others => '0');
29        end if;
30      end if;
31    end process cnt2;
32
33    count2 <= count2_int;
34
35    cnt3: process(reset, clock)
36      variable count3_int: unsigned(count3'range);
37    begin
38      if reset = '1' then
39        count3_int := (others => '0');
40      elsif rising_edge(clock) then
41        count3_int := count3_int + 1;
42        if count3_int = 6 then
43          count3_int := (others => '0');
44        end if;
45      end if;
46      count3 <= count3_int;
47    end process cnt3;
48
49  end RTL;
```

Listing 9: 3 little counters

6.2 Parity

Give the VHDL architecture of the block which calculates the even parity of a bit vector and whose **entity** is given hereafter.

```
1  library ieee;
2    use ieee.std_logic_1164.all;
3    use ieee.numeric_std.all;
4
5  entity parityCalc is
6    generic(
7      dataBitNb : positive := 8
8    );
9    port(
10     vectorIn  : in  std_ulogic_vector(dataBitNb-1 downto 0);
11     vectorOut : out std_ulogic_vector(dataBitNb-1 downto 0)
12   );
13 end parityCalc;
```

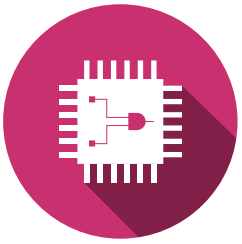
Listing 10: parityCalc **entity**

In the case of even parity, the total number of bits on '1', including the parity bit, is even.

6.3 Parity

Write the architecture of the circuit that determines the majority ('0' or '1') of the bits of a vector with an odd number en elements.

```
1  library ieee;
2    use ieee.std_logic_1164.all;
3    use ieee.numeric_std.all;
4
```



```vhdl
entity majorityFinder is
  port(
    poll     : in  std_uLogic_vector (1 to 7);
    majority : out std_ulogic
  );
end majorityFinder;
```

Listing 11: majorityFinder `entity`

As a general rule, the calculation of the majority is done by counting the votes.

Acronyms

VHDL **V**HSIC **H**ardware **D**esign **L**anguage. 1–7