


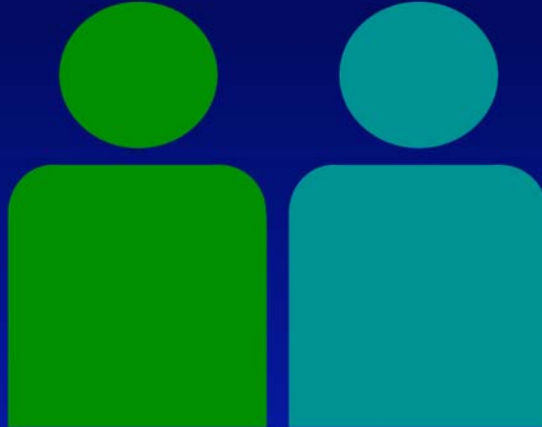


Exercices de VHDL

Eduardo Sanchez



Ecole Polytechnique Fédérale de Lausanne



Compteur 4 bits

```
library ieee;  
use ieee.std_logic_1164.all;  
use ieee.std_logic_arith.all;  
use ieee.std_logic_unsigned.all;  
  
entity counter is  
  port (Clk      : in std_logic;  
        Reset   : in std_logic;  
        Load    : in std_logic;  
        MaxCount : in std_logic_vector(3 downto 0);  
        Zero    : out std_logic;  
        Count   : out std_logic_vector(3 downto 0));  
end counter;
```

```
architecture behavioral of counter is
```

```
    signal counter_value, max_value : std_logic_vector(3 downto 0);
```

```
begin
```

```
MAX HOLDER : process (Clk, Reset, Load, MaxCount)
```

```
begin
```

```
    if (Reset = '1')
```

```
        then max_value <= (others => '0');
```

```
    elsif (Clk'event and Clk = '1')
```

```
        then if (Load = '1')
```

```
            then max_value <= MaxCount;
```

```
        end if;
```

```
    end if;
```

```
end process MAX HOLDER;
```



```
COUNTER : process (Clk, Reset, max_value)
```

```
begin
```

```
    if (Reset = '1')
```

```
        then
```

```
            counter_value <= (others => '0');
```

```
            Zero <= '1';
```

```
    elsif (Clk'event and Clk = '0')
```

```
        then if (counter_value >= max_value)
```

```
            then
```

```
                counter_value <= (others => '0');
```

```
                Zero <= '1';
```

```
            else
```

```
                counter_value <= counter_value + 1;
```

```
                Zero <= '0';
```

```
        end if;
```

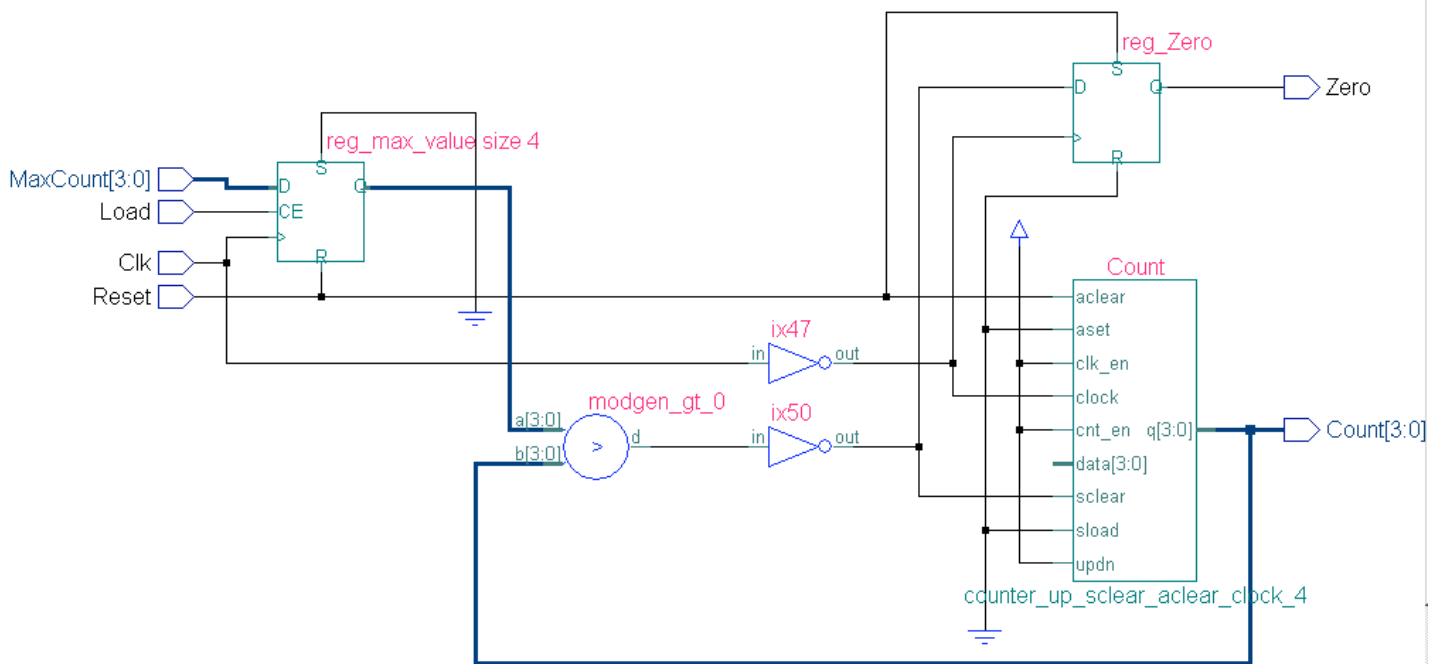
```
    end if;
```

```
end process COUNTER;
```

```
Count <= counter_value;
```

```
end behavioral;
```





Reconnaissance d'un motif 8 bits

```

library ieee;
use ieee.std_logic_1164.all;

entity pattern is
  port (Clk      : in std_logic;
        Reset   : in std_logic;
        Load    : in std_logic;
        Pattern  : in std_logic_vector(7 downto 0);
        DataIn  : in std_logic;
        Found   : out std_logic);
end pattern;

```

```
architecture behavioral of pattern is
```

```
    signal pattern_memory, window : std_logic_vector(7 downto 0);
```

```
begin
```

```
PATTERN HOLDER : process (Clk, Reset, Load, Pattern)
```

```
begin
```

```
    if (Reset = '1')
```

```
        then pattern_memory <= (others => '0');
```

```
    elsif (Clk'event and Clk = '1')
```

```
        then if (Load = '1')
```

```
            then pattern_memory <= Pattern;
```

```
            end if;
```

```
        end if;
```

```
    end process PATTERN HOLDER;
```



```
DATA_WINDOW : process (Clk, Reset, DataIn)
```

```
begin
```

```
    if (Reset = '1')
```

```
        then window <= (others => '0');
```

```
    elsif (Clk'event and Clk = '1')
```

```
        then window <= DataIn & window(7 downto 1);
```

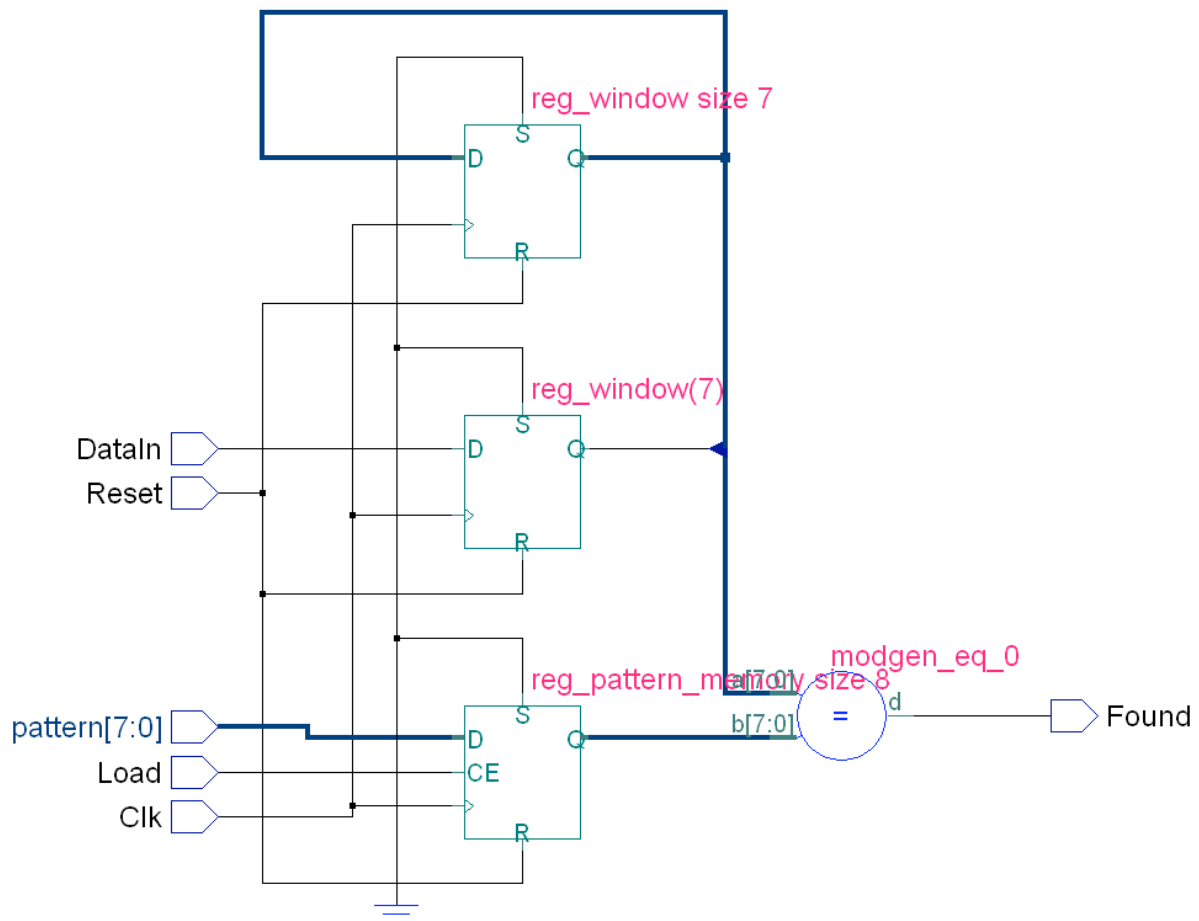
```
    end if;
```

```
end process DATA_WINDOW;
```

```
Found <= '1' when window = pattern_memory else '0';
```

```
end behavioral;
```





Convertisseur série-parallèle 8 bits

```

library ieee;
use ieee.std_logic_1164.all;

entity serial2para is
  port (Clk      : in std_logic;
        Reset   : in std_logic;
        Start   : in std_logic;
        DataIn  : in std_logic;
        Valid   : out std_logic;
        DataOut : out std_logic_vector(7 downto 0));
end serial2para;

```

```
architecture behavioral of serial2para is
```

```
    type ctrl_state_type is (state_init, state_got_1, state_got_2,  
                             state_got_3, state_got_4, state_got_5,  
                             state_got_6, state_got_7, state_valid);  
    signal ctrl_state, next_ctrl_state : ctrl_state_type;  
    signal memory_content : std_logic_vector(7 downto 0);  
    signal memory_write : std_logic;
```

```
begin
```

```
MEMORY_MGR : process (Clk, Reset)
```

```
    begin
```

```
        if (Reset = '1')
```

```
            then memory_content <= (others => '0');
```

```
            elsif (Clk'event and Clk = '1')
```

```
                then if (memory_write = '1')
```

```
                    then
```

```
                        memory_content <= DataIn & memory_content(7 downto 1);
```

```
                    end if;
```

```
                end if;
```

```
            end process MEMORY_MGR;
```



```
DataOut <= memory_content;
```

```
CTRL_SYNC : process (Clk, Reset)
```

```
    begin
```

```
        if (Reset = '1')
```

```
            then ctrl_state <= state_init;
```

```
            elsif (Clk'event and Clk = '1')
```

```
                then ctrl_state <= next_ctrl_state;
```

```
            end if;
```

```
        end process CTRL_SYNC;
```



```

CTRL_BEHAVIOR : process (ctrl_state, Start)
begin
    next_ctrl_state <= ctrl_state;
    Valid <= '0';
    memory_write <= '1';

```



```

case ctrl_state is
when state_init =>
    if (Start = '1')
        then next_ctrl_state <= state_got_1;
        else memory_write <= '0';
    end if;
when state_got_1 => next_ctrl_state <= state_got_2;
when state_got_2 => next_ctrl_state <= state_got_3;
when state_got_3 => next_ctrl_state <= state_got_4;
when state_got_4 => next_ctrl_state <= state_got_5;
when state_got_5 => next_ctrl_state <= state_got_6;
when state_got_6 => next_ctrl_state <= state_got_7;
when state_got_7 => next_ctrl_state <= state_valid;
when state_valid =>
    Valid <= '1';
    if (Start = '1')
        then next_ctrl_state <= state_got_1;
        else memory_write <= '0';
    end if;
end case;
end process CTRL_BEHAVIOR;
end behavioral;

```



