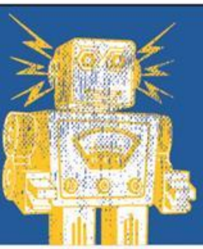
Elektronik



Introduction to VHDL III

Attributes – Flip-Flop Types - Examples

Roland Höller
Email: hoeller@technikum-wien.at

Attributes

- Attributes for VHDL types and signals ease VHDL coding

- TS'left, TS'right, TS'low and TS'high deliver the corresponding bounds of the type ranges.
- SIG'stable(T) is TRUE, if SIG did not change for the last T time units.
- SIG'event is TRUE, whenever SIG changes.

```
type a_byte is array (7 downto 0) of bit;

-- a_byte'left and a_byte'high are 7.
-- a_byte'right and a_byte'low are 0.
-----------------------------------------
```

Attributes

- Most often used attribute for RTL design:

```
if clk'event and (clk = 1) then
  s_q_o <= s_d_i;
end if;
```

jede steigende Flanke

s_q_o bekommt Wert von s_d_i bei der nächsten steigenden Flanke

- Using the LENGTH attribute:

```
for i in 0 to s_d_i'length
  v_parity := v_parity xor s_d_i(i);
end loop;
```

Attributes

- Using the HIGH attribute:

```
s_d_to_pci_o(s_status_reg'high downto 0) <= s_status_reg;

s_d_to_pci_o(31 downto s_status_reg'length) <= (others => '0‘);
```

Attributes

- Special form of attributes of VHDL signals for communication with EDA tools:

```
library IEEE;
use IEEE.std_logic_1164.all;

entity ff3 is
  port(data_i:  in  std_logic;
       clk_i:   in  std_logic;
       reset_i: in  std_logic;
       qout_o:  out std_logic);
  attribute sync_set_reset of reset_i : signal is "true";
end ff3;
```

Flip Flop 1 (edge sensitive)

```
library IEEE;
use IEEE.std_logic_1164.all;
entity ff1 is
  port(data_i:  in  std_logic;
       clk_i:   in  std_logic;
       qout_o:  out std_logic);
end ff1;
-- VHDL description of a D-Flip Flop
architecture rtl of ff1 is
begin
  p_ff: process(clk_i, data_i)
  begin
    if (clk_i'event and clk_i = '1') then
      qout_o <= data_i;
    end if;
  end process p_ff;
end rtl;
```

qout_o bekommt nur wenn sich clk_i ändert und eine steigende Flanke den Wert von data_i

Flip Flop 2 (edge sensitive)

```
library IEEE;
use IEEE.std_logic_1164.all;
entity ff2 is
  port(data_i:  in  std_logic;
       clk_i:   in  std_logic;
       qout_o:  out std_logic);
end ff2;
-- VHDL description of a D-Flip Flop
-- data_i is not in the sensitivity list
architecture rtl of ff2 is
begin
  p_ff: process(clk_i)
  begin
    if (clk_i'event and clk_i = '1') then
      qout_o <= data_i;
    end if;
  end process p_ff;
end rtl;
```

Flip Flop 3 (asynchronous reset)

```
library IEEE;
use IEEE.std_logic_1164.all;
entity ff3 is
  port(data_i:  in  std_logic;
       clk_i:   in  std_logic;
       reset_i: in  std_logic;
       qout_o:  out std_logic);
end ff3;
-- VHDL description of a D-Flip Flop
-- with asynchronous reset
architecture rtl of ff3 is
begin
  p_ff: process(clk_i, reset_i)
  begin
    if reset_i = '1' then
      qout_o <= '0';
    elsif (clk_i'event and clk_i = '1') then
      qout_o <= data_i;
    end if;
  end process p_ff;
end rtl;
```

Wenn reset_i eins ist, wird der Ausgang qout_o 0 gesetzt. Wenn Clock eine steigende Flanke ist bekommt der Ausgang qout_o den Wert vom Eingang data_i.

Flip Flop 4 (bad description)

```
library IEEE;
use IEEE.std_logic_1164.all;
entity ff4 is
  port(data_i:  in  std_logic;
       clk_i:   in  std_logic;
       reset_i: in  std_logic;
       qout_o:  out std_logic);
end ff4;
-- VHDL description of a Flip Flop
-- which is not synthesizeable
architecture rtl of ff4 is
begin
  p_ff: process(clk_i, reset_i)
  begin
    if (clk_i'event and clk_i = '1') then
      qout_o <= data_i;
    elsif reset_i = '1' then
      qout_o <= '0';
    end if;
  end process p_ff;
end rtl;
```

Flip Flop 5 (alternate description)

```
library IEEE;
use IEEE.std_logic_1164.all;
entity ff5 is
  port(data_i:  in  std_logic;
       clk_i:   in  std_logic;
       reset_i: in  std_logic;
       qout_o:  out std_logic);
end ff5;
-- VHDL description of a D-Flip Flop
architecture rtl of ff5 is
begin
  p_ff: process(clk_i)
  begin
    wait until (clk_i'event and clk_i = '1');
      qout_o <= data_i;
  end process p_ff;
end rtl;
```

Flip Flop 6 (falling edge sensitive)

```
library IEEE;
use IEEE.std_logic_1164.all;
entity ff6 is
  port(data_i:  in  std_logic;
       clk_i:   in  std_logic;
       reset_i: in  std_logic;
       qout_o:  out std_logic);
end ff6;
-- VHDL description of a Flip Flop
-- with asynchronous reset, negative edge
architecture rtl of ff6 is
begin
  p_ff: process(clk_i, reset_i)
  begin
    if reset_i = '1' then
      qout_o <= '0';
    elsif (clk_i'event and clk_i = '0') then
      qout_o <= data_i;
    end if;
  end process p_ff;
end rtl;
```

Flip Flop 7 (reset and enable)

```
library IEEE;
use IEEE.std_logic_1164.all;
entity ff7 is
  port(data_i:   in  std_logic;
       clk_i:    in  std_logic;
       reset_i:  in  std_logic;
       enable_i: in  std_logic;
       qout_o:   out std_logic);
end ff7;
-- VHDL description of a Flip Flop
-- with asynchronous reset and enable
architecture rtl of ff7 is
begin
  p_ff: process(clk_i, reset_i)
  begin
    if reset_i = '1' then
      qout_o <= '0';
    elsif (clk_i'event and clk_i = '1') then
      if enable_i = '1' then
        qout_o <= data_i;
      end if;
    end if;
  end process p_ff;
end rtl;
```

Flip Flop 8 (T-FF)

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;
entity ff8 is
  port(data_i:   in  std_logic;
       clk_i:    in  std_logic;
       reset_i:  in  std_logic;
       toggle_i: in  std_logic;
       qout_o:   out std_logic);
end ff8;
-- VHDL description of a
-- Toggle Flip Flop
-- with asynchronous reset
architecture rtl of ff8 is
  signal s_data_int : std_logic;
begin
```

```vhdl
p_ff: process(clk_i, reset_i)
  begin
    if reset_i = '1' then
      s_data_int <= '0';
    elsif (clk_i'event and clk_i = '1') then
      if toggle_i = '1' then
        s_data_int <= not(s_data_int);
      end if;
    end if;
  end process p_ff;
qout_o <= s_data_int;
end rtl;
```

Zuweisung außerhalb des Prozesses und nur einmal im Code

Flip Flop 9 (synchronous set)

```
library IEEE;
use IEEE.std_logic_1164.all;
entity ff9 is
  port(data_i:   in  std_logic;
       clk_i:    in  std_logic;
       set_i:    in  std_logic;
       qout_o:   out std_logic);
end ff9;
-- VHDL description of a D-Flip Flop
-- with synchronous set
architecture rtl of ff9 is
begin
  p_ff: process(clk_i)
  begin
    if (clk_i'event and clk_i = '1') then
      if set_i = '1' then
        qout_o <= '1';
      else
        qout_o <= data_i;
      end if;
    end if;
  end process p_ff;
end rtl;
```

Genau das Gegenteil von
dem asynchronen Flip-Flop.
Erst event Abfrage dann reset Abfrage

Summary

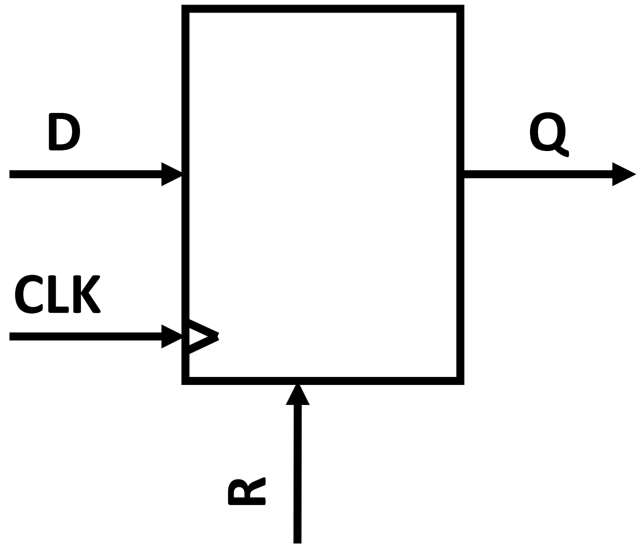
- VHDL Attributes
- Various Flip-Flop Descriptions

Questions

- 1) Which VHDL attribute is true if a signal changes its value?
  - A ‘length
  - B ‘high
  - C ‘low
  - D ‘event
- 2) Which VHDL attribute delivers the length of a vector?
  - A ‘low
  - B ‘high
  - C ‘event
  - D ‘length
- 3) Which of the following examples describe the falling edge of signal clk?
  - A if clk'event and clk='1' then
  - B if clk'stable then
  - C if clk'event and clk='0' then
  - D if clk='0' then

Class Example: D/JK-Flip Flop

1) Describe and simulate a D-Flip Flop in VHDL



2) Change the VHDL code to describe a JK Flip Flop

J	K	Q
0	0	hold
0	1	0
1	0	1
1	1	toggle