VHDL Flip-Flop Template

Last updated 1/8/25

VHDL Flip-Flop Template

- VHDL has no code-keyword or defined structure to create a flip-flop
  - Flip-Flop templates were developed by the synthesis tool developers
  - While most are essentially the same – it is not guaranteed

VHDL Flip-Flop Template

- Registers (Flip-Flops) are recognized by a pre-defined template
  - Provided by the synthesis/simulation tool developer

```
process (clock signal)
  begin
    if(clock edge detection) then
      actions
    end if;
  end process;
```

note:
- here the else is not required because the synthesizer recognizes the edge detection

- you can include an else for clarity

```
process (i_clk)
  begin
    if(clk'event and i_clk = 1) then
      q <= d;
    end if;
  end process;
```

```
process (i_clk)
  begin
    if(rising_edge(i_clk)) then
      q <= d;
    end if;
  end process;
```

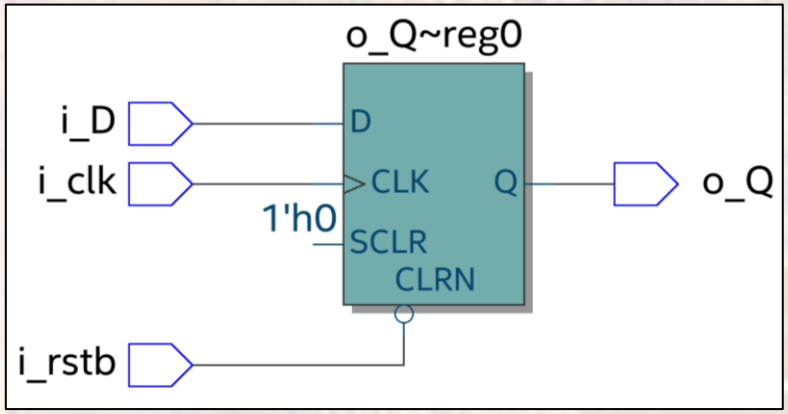
2008 release

VHDL Flip-Flop Template

- Base D-FF with reset

```
--------------------------------
--
--   ff_d.vhdl
--
--   by: johnsontimoj
--
--   created: 12/31/24
--
--   version: 0.0
--
--------------------------------
--
--   standard d-ff with rstb
--   inputs: clk, rstb, d
--
--   outputs: q
--
--------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity ff_d is
   port(
      i_clk:   in std_logic;
      i_rstb:  in std_logic;
      i_D :    in std_logic;

      o_Q :    out std_logic
   );
end entity;
```

```
architecture behavioral of ff_d is

begin

   process (i_clk, i_rstb)
   begin
      if (i_rstb = '0') then
         o_Q <= '0';
      elsif (rising_edge(i_clk)) then
         o_Q <= i_D;
      end if;
   end process;

end architecture;
```



Note – the template may include additional inputs

VHDL Flip-Flop

All asynchronous inputs + clk should be in the process sensitivity list

- Base D-FF with reset

```
--------------------------------
--
--  ff_d.vhdl
--
--  by: johnsontimoj
--
--  created: 12/31/24
--
--  version: 0.0
--
--------------------------------
--
--  standard d-ff with rstb
--  inputs: clk, rstb, d
--
--  outputs: q
--
--------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity ff_d is
   port(
      i_clk:   in std_logic;
      i_rstb:  in std_logic;
      i_D :    in std_logic;

      o_Q :    out std_logic
   );
end entity;
```

```
architecture behavioral

begin

   process (i_clk, i_rstb)
   begin
      if (i_rstb = '0') then
         o_Q <= '0';
      elsif (rising_edge(i_clk)) then
         o_Q <= i_D;
      end if;
   end process;

end architecture;
```

Asynchronous elements go before the rising_edge

Synchronous elements go inside the rising_edge

Only 1 rising_edge in a process

Most of our designs will use DFFs with an asynchronous reset BAR

VHDL Flip-Flop Template

- Warning – Warning – Warning
  - The FF template is an exception to the if/else and case rules for creating latches
  - Outside the FF construct:
    - If you do not complete an if-else with an else, a latch will be created
    - If you do not cover all cases in a case statement, a latch will be created
    - All paths/cases must be covered
    - The compiler will always warn you it created a latch

We do not want latches - EVER

I can see a latch in an RTL diagram from a mile away