Flip-Flop Construct

Last updated 2/8/24

Flip-Flop Construct

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

Flip-Flop Construct

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

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

2008 release

Flip-Flop Construct

- D-FF w/ asynchronous rstb

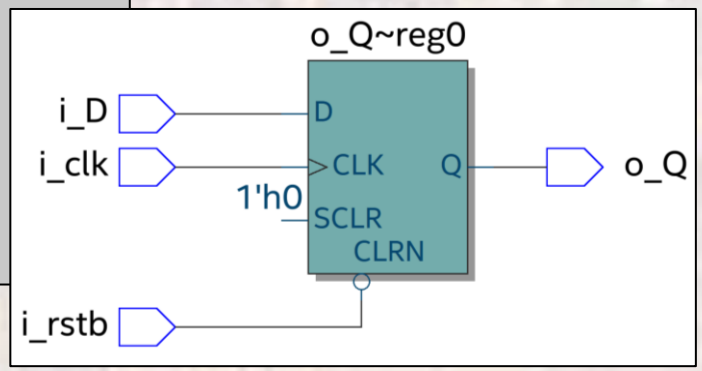
```
library ieee;
use ieee.std_logic_1164.all;

entity dff_a 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 dff_a 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 addition of rstb to the sensitivity list



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

Data Path designs will use DFFs with no reset

Flip-Flop Construct

- General rules

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

entity dff_a 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 dff_a 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;
```

Asynchronous elements go before the rising_edge

Synchronous elements go inside the rising_edge

Only 1 rising_edge in a process

Flip-Flop Construct

- D-FF w/ synchronous set, rstb, en

```
----------------------------------------
--
-- dff_s.vhdl
--
-- created 7/5/2018
-- tj
--
-- rev 0
----------------------------------------
--
-- dff example - syncronous inputs
--
----------------------------------------
--
-- Inputs: D, clk, rstb, set, enable
-- Outputs: Q
--
----------------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity dff_s is
   port(
      i_clk:   in std_logic;
      i_rstb:  in std_logic;
      i_set:   in std_logic;
      i_en:    in std_logic;
      i_D :    in std_logic;

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

Note rstb, set, and en are NOT in the sensitivity list

```
architecture behavioral of dff_s is
begin
   process(i_clk)
   begin
      if (rising_edge(i_clk)) then
         if (i_rstb = '0') then
            o_Q <= '0';
         elsif (i_set = '1') then
            o_Q <= '1';
         elsif (i_en = '1') then
            o_Q <= i_D;
         end if;
      end if;
   end process;

end architecture;
```

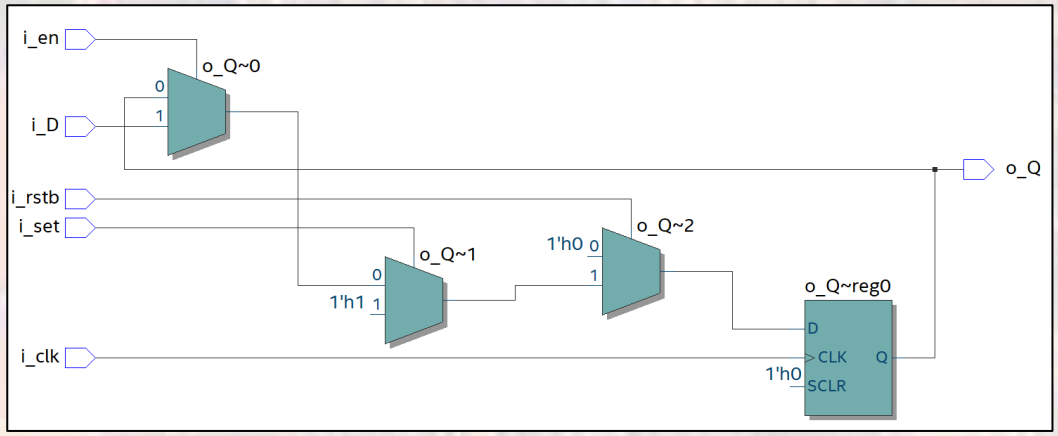
Note: there is an inherent priority in this design

Flip-Flop Construct

- D-FF w/ synchronous set, rstb, en

```
architecture behavioral of dff_s is
begin
   process(i_clk)
   begin
      if (rising_edge(i_clk)) then
         if (i_rstb = '0') then
            o_Q <= '0';
         elsif (i_set = '1') then
            o_Q <= '1';
         elsif (i_en = '1') then
            o_Q <= i_D;
         end if;
      end if;
   end process;

end architecture;
```



priority

rstb > set > en

Flip-Flop Construct

- Warning – Warning – Warning
  - The FF construct is an exception to the if/else rule 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