VHDL

Diego Trapero

Table of contents

1 VHDL syntax

1.1 when / else : concurrent signal assignment statement

A sequential signal assignment statement is also a concurrent signal assignment statement. Additional control is provided by the use of postponed and guarded.

```
[ label : ] sequential signal assignment statement

[ label : ] [ postponed ] conditional_signal_assignment_statement ;

[ label : ] [ postponed ] selected_signal_assignment_statement ;
```

The optional guarded causes the statement to be executed when the guarded signal changes from False to True. conditional signal assignment statement

A conditional assignment statement is also a concurrent signal assignment statement.

```
target <= waveform when choice; -- choice is a boolean expression
target <= waveform when choice else waveform;

sig <= a_sig when count>7;
sig2 <= not a_sig after 1 ns when ctl='1' else b_sig;

-- "waveform" for this statement seems to include [ delay_mechanism ]
-- See sequential signal assignment statement
```

2 Libraries

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

3 Simulation

3.1 Clocks

3.1.1 Default clock

Default clock used in automatically generated testbenchs:

```
constant <clock>_period : time := 10 ns;

   -- Clock process definitions
   <clock>_process :process
   begin
    <clock> <= '0';
    wait for <clock>_period/2;
    <clock> <= '1';
    wait for <clock>_period/2;
   end process;
```

3.1.2 My Clock

```
--proceso de reloj, útil para un testbech

-- inicializar la variable en la zona de señales
SIGNAL CLK  : STD_LOGIC := '0';


clock : process(clk)
begin
  clk <= not clk after 20ns;
end process;
```

4 Snippets

4.1 Rellenar un vector de ceros

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

4.2 Synchronous processes

Todos los procesos sincronos con resets

```
--todos los procesos sincronos con resets:

process (clk, reset)
   if reset = '1' then
      -- do RESET, other asyncronous taska ;
   elseif rising_edge(clk) then
      -- SINCRONOUS TASKS
      if enable = '1' then
         -- ENABLED TASKS

   endif;
end process;
```

5 VHDL component

5.1 Gates

NAND Gate

```
entity NANDGate is
   port (
      a, b : in bit;
      x : out bit
   );
end NANDGate;

architecture NANDArch of NANDGate is
begin
   x <= a NAND b;
end NANDArch;
```

5.2 Biestables

Latch Internal signal implementation

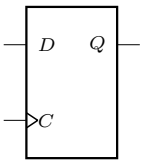
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity S_R_latch_top is
    Port ( S : in    STD_LOGIC;
           R : in    STD_LOGIC;
           Q : out   STD_LOGIC);
end S_R_latch_top;

architecture Behavioral of S_R_latch_top is
signal Q2   : STD_LOGIC;
signal notQ : STD_LOGIC;
begin

Q    <= Q2;
Q2   <= R nor notQ;
notQ <= S nor Q2;

end Behavioral;
```

Latch Inout ports implementation

```
entity SR_Latch is
  Port ( S,R : in STD_LOGIC;
         Q : inout STD_LOGIC;
         Q_n : inout STD_LOGIC);
 end SR_Latch;

 architecture SR_Latch_arch of SR_Latch is
 begin
 process (S,R,Q,Q_n)
       begin
             Q <= R NOR Q_n;
             Q_n <= S NOR Q;
       end process;

 end SR_Latch_arch;

-- http://vhdlbynaresh.blogspot.com.es/2013/07/design-of-sr-latch-using-behavior.html

library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity SR_Latch is
     port(
         enable : in STD_LOGIC;
         s : in STD_LOGIC;
         r : in STD_LOGIC;
         reset : in STD_LOGIC;
         q : out STD_LOGIC;
         qb : out STD_LOGIC
         );
end SR_Latch;

architecture SR_Latch_arc of SR_Latch is
begin

    latch : process (s,r,enable,reset) is
```

```
    begin
        if (reset='1') then
            q <= '0';
            qb <= '1';
        elsif (enable='1') then
            if (s/=r) then
                q <= s;
                qb <= r;
            elsif (s='1' and r='1') then
                q <= 'Z';
                qb <= 'Z';
            end if;
        end if;
    end process latch;

end SR_Latch_arc;
```

5.2.1 Basic D Flip Flop (FD)



CLK	D	Q_{next}
^	0	0
^	1	1
0	X	Q
1	X	Q

5.2.1.1 Code

```
-- Basic D Flip Flop


library IEEE;
use IEEE.std_logic_1164.all;

entity DFlipFlop is
  port(
    D : in std_logic;
    C : in std_logic;
    Q : out std_logic
  );
end DFlipFlop;

Architecture Behavioral of DFlipFlop is
begin
  process(C)
  begin
    if rising_edge(C) then
      Q <= D;
    end if;
  end process;
end Behavioral;
```

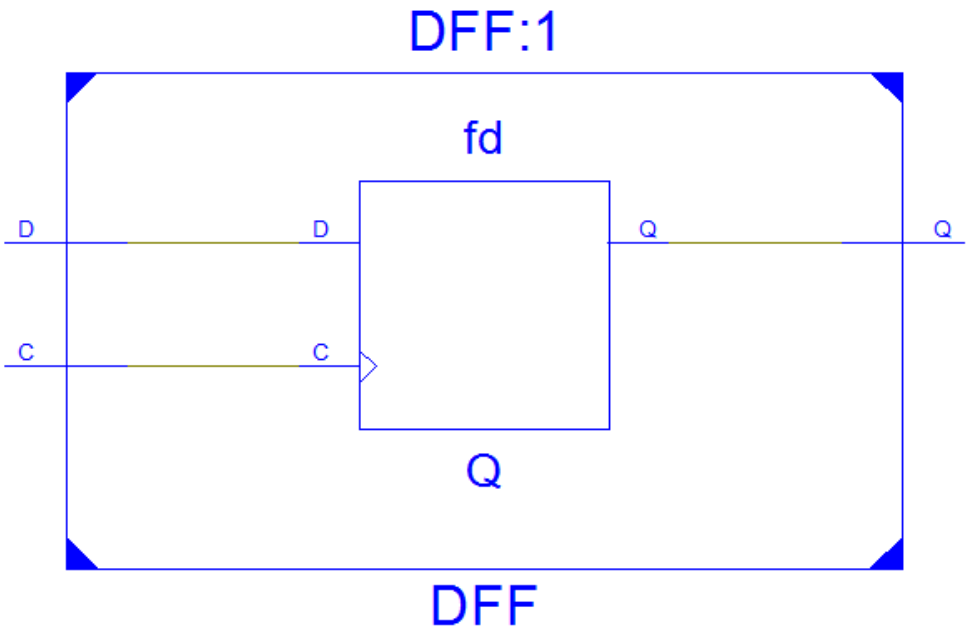


Figure 1: Synthesis Result

5.2.1.2 Synthesis results

- Target Device: xa3s500e-4cpg132
- Design Goal: Balanced

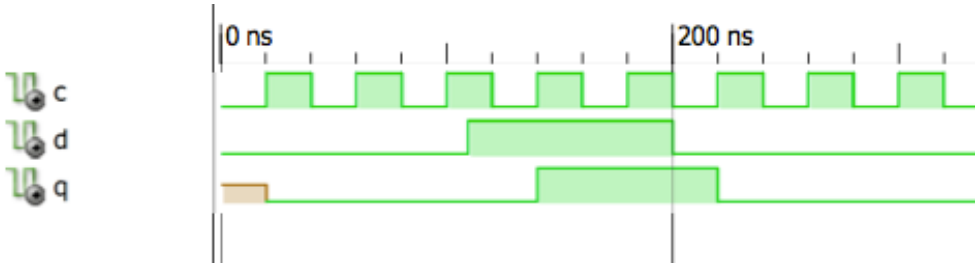
```
Logic Utilization     Used    Available   Utilization
-------------------------------------------------------------------
Number of bonded IOBs   3       210       1%
Number of BUFG/BUFGCTRLs  1       32        3%

=========================================================================
*                            Final Report                               *
=========================================================================
Final Results
RTL Top Level Output File Name     : DFlipFlop.ngr
Top Level Output File Name         : DFlipFlop
Output Format                      : NGC
Optimization Goal                  : Speed
Keep Hierarchy                     : No

Design Statistics
# IOs                              : 3

Cell Usage :
# FlipFlops/Latches                : 1
#      FD                          : 1
# Clock Buffers                    : 1
#      BUFGP                       : 1
# IO Buffers                       : 2
#      IBUF                        : 1
#      OBUF                        : 1
=========================================================================

Device utilization summary:
---------------------------

Selected Device : xa3s500ecpg132-4
```

```
 Number of Slices:                        0  out of   4656     0%
 Number of IOs:                           3
 Number of bonded IOBs:                   3  out of     92     3%
    IOB Flip Flops:                       1
 Number of GCLKs:                         1  out of     24     4%
```

5.2.1.3 Testbench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;

ENTITY tb IS
END tb;

ARCHITECTURE behavior OF tb IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT DFlipFlop
    PORT(
         D : IN  std_logic;
         C : IN  std_logic;
         Q : OUT  std_logic
        );
    END COMPONENT;


   --Inputs
  signal C : std_logic := '0';
   signal D : std_logic := '0';

  --Outputs
   signal Q : std_logic;

BEGIN

  -- Instantiate the Unit Under Test (UUT)
   uut: DFlipFlop PORT MAP (
          D => D,
          C => C,
          Q => Q
        );

  -- my clock process:
  clock : process(C)
  begin
    C <= not C after 20ns;
  end process;

   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
     wait for 110 ns;

      -- insert stimulus here
    D <= '1';
    wait for 90ns;
    D <= '0';
```

```
    wait for 90ns;

      wait;
   end process;

END;
```



5.2.1.4 Simulation

5.2.2 D Flip Flop with Enable and Asynchronous Reset

5.2.2.1 Code

```
-- D Flip Flop with Enable and Asynchronous Reset

library IEEE;
use IEEE.std_logic_1164.all;

entity DFlipFlop is
  port(
    D  : in std_logic;  -- Data input
    C  : in std_logic;  -- Clock signal
    EN : in std_logic;  -- Enable
    R  : in std_logic;  -- Asynchronous enable
    Q  : out std_logic  -- Data output
  );
end DFlipFlop;

Architecture Behavioral of DFlipFlop is
begin
  process(C, R) -- Only clock and asynchronous signals in the sensitivity list
  begin
    if R = '1' then
      Q <= '0';
    elsif rising_edge(C) then
      if EN = '1' then
        Q <= D;
      end if;
    end if;
  end process;
end Behavioral;
```

5.2.2.2 Synthesis XILINX primitive: FDCE, D Flip-Flop with Clock Enable and Asynchronous Clear

- Target Device: xa3s500e-4cpg132
- Design Goal: Balanced

```
Logic utilization     Used    Available   Utilization
-------------------------------------------------------------------
Number of Slices    0       4656      0%
```

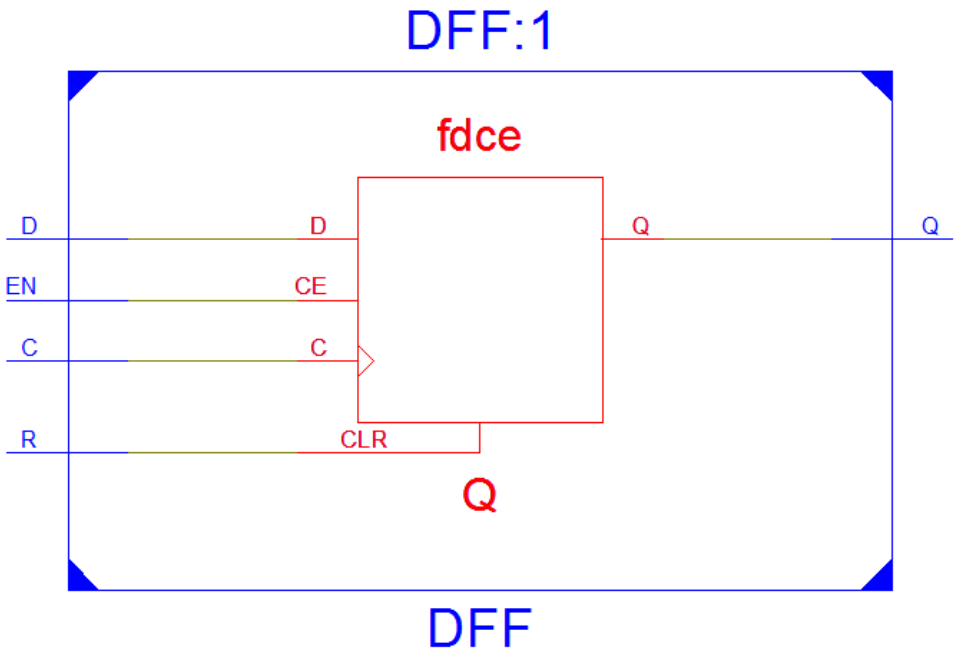


Figure 2: Synthesis Result

```
Number of bonded IOBs   5        92         5%
Number of GCLKs     1        24         4%

=========================================================================
*                            Final Report                               *
=========================================================================
Final Results
RTL Top Level Output File Name     : DFlipFlop.ngr
Top Level Output File Name         : DFlipFlop
Output Format                      : NGC
Optimization Goal                  : Speed
Keep Hierarchy                     : No

Design Statistics
# IOs                              : 5

Cell Usage :
# FlipFlops/Latches                : 1
#      FDCE                        : 1
# Clock Buffers                    : 1
#      BUFGP                       : 1
# IO Buffers                       : 4
#      IBUF                        : 3
#      OBUF                        : 1
=========================================================================

Device utilization summary:
---------------------------

Selected Device : xa3s500ecpg132-4

 Number of Slices:                        0  out of   4656     0%
 Number of IOs:                           5
 Number of bonded IOBs:                   5  out of     92     5%
    IOB Flip Flops:                       1
 Number of GCLKs:                         1  out of     24     4%
```

5.2.3 D Flip Flop with Enable, Reset and Preset

5.2.3.1 Code

```
-- Code from Wikibooks:
-- https://en.wikibooks.org/wiki/VHDL_for_FPGA_Design/D_Flip_Flop

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity DFF is
   port
   (
      clk : in std_logic;

      rst : in std_logic;
      pre : in std_logic;
      ce  : in std_logic;

      d : in std_logic;

      q : out std_logic
   );
end entity DFF;

architecture Behavioral of DFF is
begin
   process (clk) is
   begin
      if rising_edge(clk) then
         if (rst='1') then
            q <= '0';
         elsif (pre='1') then
            q <= '1';
         elsif (ce='1') then
            q <= d;
         end if;
      end if;
   end process;
end architecture Behavioral;
```

5.2.3.2 Synthesis Xilinx primitive: FDRSE

FDRSE is a single D-type flip-flop with synchronous reset (R), synchronous set (S), clock enable (CE) inputs. The reset (R) input, when High, overrides all other inputs and resets the Q output Low during the Low-to-High clock transition. (Reset has precedence over Set.) When the set (S) input is High and R is Low, the flip-flop is set, output High, during the Low-to-High clock (C) transition. Data on the D input is loaded into the flip-flop when R and S are Low and CE is High during the Low-to-High clock transition.

```
Inputs Outputs
---------------
R S CE D C Q
---------------
1 X X  X ? 0
0 1 X  X ? 1
0 0 0  X X No Change
0 0 1  1 ? 1
0 0 1  0 ? 0
```

5.3 FSM

FSM template

```
-- FSM template: from Circuit Design with VHDL

LIBRARY ieee;
USE ieee.std_logic_1164.all;
-------------------------------------------------
ENTITY <entity_name> IS
   PORT ( input: IN <data_type>;
          reset, clock: IN STD_LOGIC;
          output: OUT <data_type>);
END <entity_name>;
-------------------------------------------------
ARCHITECTURE <arch_name> OF <entity_name> IS
TYPE state IS (state0, state1, state2, state3, ...);
   SIGNAL pr_state, nx_state: state;
BEGIN
   ---------- Lower section: ------------------------
   PROCESS (reset, clock)
   BEGIN
      IF (reset='1') THEN
         pr_state <= state0;
      ELSIF (clock'EVENT AND clock='1') THEN
         pr_state <= nx_state;
      END IF;
   END PROCESS;
   ---------- Upper section: ------------------------
   PROCESS (input, pr_state)
   BEGIN
      CASE pr_state IS
         WHEN state0 =>
            IF (input = ...) THEN
               output <= <value>;
               nx_state <= state1;
ELSE ...
            END IF;
         WHEN state1 =>
            IF (input = ...) THEN
               output <= <value>;
               nx_state <= state2;
ELSE ...
            END IF;
         WHEN state2 =>
            IF (input = ...) THEN
               output <= <value>;
               nx_state <= state3;
ELSE ...
END IF; ...
      END CASE;
   END PROCESS;
END <arch_name>;
```

6 Examples

7 More