7.1 About this practical

The objective of this practical is to design and test three implementations of a D flip-flop as an asynchronous sequential circuit.

Contents

7.1 About this practical .......................................................... 1
7.2 A two-state-signals implementation of a D flip flop .................. 1
7.3 A two-state-signals implementation of a D flip flop .................. 2
7.4 A three-state-signals implementation of a D flip flop ................. 4
7.5 The report ................................................................. 5

7.2 A two-state-signals implementation of a D flip flop

The first implementation is based on the generic description of a D flip-flop in VHDL which is based on the process statement and the event attribute.

- Start the Design manager and create a new project:  
- Create a VHDL design view with the
  - Entity name:  DFF
  - Architecture name: e.g., ahdl
- In the VHDL editor window enter an appropriate entity specification for the D flip-flop and
- the architecture to be as follows (watch the single quotes!)

ARCHITECTURE ahdl OF DFF IS
BEGIN
  PROCESS (clk)
  BEGIN
    IF (clk'EVENT AND clk = '1') THEN
      Q <= D ;
    END IF ;
  END PROCESS ;
END ARCHITECTURE ahdl;
• Save the VHDL file.
• Compile the VHDL specification selecting in the VHDL editor window:
  \textbf{Tasks $\rightarrow$ ModelSim flow $\rightarrow$ Run single}
• At the conclusion of the successful compilation a window \textbf{Start ModelSim} appears. If you accept its default settings, it will open ModelSim simulation window.
• Write and execute an appropriate simulation script to generate waveforms conceptually similar to the following:

\begin{center}
\begin{tikzpicture}
\node (clk) at (0,0) {\text{clk}};
\node (D) at (1,0) {\text{D}};
\node (Q) at (2,0) {\text{Q}};
\node (s1) at (1,1) {s1};
\node (s2) at (2,1) {s2};
\node (s3) at (3,1) {s3};
\node (t1) at (1,-1) {t_1};
\node (t2) at (2,-1) {t_2};
\node (t3) at (3,-1) {t_3};
\draw[->] (clk) -- (D);
\draw[->] (D) -- (Q);
\draw[->] (Q) -- (s1);
\draw[->] (s1) -- (s2);
\draw[->] (s2) -- (s3);
\draw[->] (s3) -- (s1);
\end{tikzpicture}
\end{center}

7.3 \textbf{A two-state-signals implementation of a D flip flop}

• In the second implementation of the D flip-flop we start with the state diagram with two state signals $P$ and $Q$ as discussed in the lecture notes:

\begin{center}
\begin{tikzpicture}
\node (D) at (0,0) {D};
\node (Q) at (0,-1) {Q};
\node (C/L) at (0,-2) {C/L};
\node (P) at (0,-3) {P};
\node (clk) at (0,-0.5) {clk};
\node (clk0) at (0,-1.5) {clk=0};
\node (clk1) at (0,-2.5) {clk=1};
\node (P00) at (1,0) {P \ 0 \ 0};
\node (P01) at (1,1) {P \ 0 \ 1};
\node (P10) at (1,-1) {P \ 1 \ 0};
\node (P11) at (1,-2) {P \ 1 \ 1};
\node (Q00) at (2,0) {Q \ 0 \ 0};
\node (Q01) at (2,1) {Q \ 0 \ 1};
\node (Q10) at (2,-1) {Q \ 1 \ 0};
\node (Q11) at (2,-2) {Q \ 1 \ 1};
\draw[->] (D) -- (P00);
\draw[->] (D) -- (P01);
\draw[->] (D) -- (P10);
\draw[->] (D) -- (P11);
\draw[->] (Q) -- (P00);
\draw[->] (Q) -- (P01);
\draw[->] (Q) -- (P10);
\draw[->] (Q) -- (P11);
\draw[->] (C/L) -- (P00);
\draw[->] (C/L) -- (P01);
\draw[->] (C/L) -- (P10);
\draw[->] (C/L) -- (P11);
\draw[->] (P00) -- (Q10);
\draw[->] (P01) -- (Q11);
\draw[->] (P10) -- (Q10);
\draw[->] (P11) -- (Q11);
\draw[->] (P00) -- (P11);
\draw[->] (P01) -- (P10);
\draw[->] (P10) -- (P01);
\draw[->] (P11) -- (P00);
\end{tikzpicture}
\end{center}

• Subsequently, we convert the state diagram into the state transition table.
Create a new VHDL architecture flowing from the above state table similar to the following:

ARCHITECTURE ttbl OF DFFyou IS
  TYPE arr2d IS ARRAY (natural range <>,
                         natural range <>) OF std_logic;
  CONSTANT ttDff : arr2d(1 to 2, 0 to 15) := (  
    -- truth table for P+ Q+  
    --0123456789abcdef (P, Q, clk, D)  
    "0010110000111110" , -- 1 P  
    "0001111100001101" );  -- 2 Q  
  SIGNAL P, QQ : std_logic;  
  SIGNAL S : std_logic_vector (3 downto 0);  
  SIGNAL Si : integer range 0 to 15;
BEGIN
  S <= (P, QQ, clk, D);  
  Si <= conv_integer(unsigned(S));  
  -- reading from the truth table  
  P <= ttDff (1, Si);  
  QQ <= ttDff (2, Si);  
  Q <= QQ;  
END ttbl;

Simulate the VHDL design to obtain waveforms similar to the following:
7.4 A three-state-signals implementation of a D flip flop

- The third implementation of the D flip-flop is based on the following NAND-based logic diagram.

- Use the **graphical entry** to specify the D flip-flop given by the above logic diagram.

- Simulate the design using an appropriate simulation script. Show the internal \( R \) and \( S \) signals.
7.5 The report

In your report (due after prac 8) include the results in the form of:

- Relevant state diagrams, state tables, state equations, other logic equations,
- block/logic diagrams,
- VHDL programs,
- simulation scripts,
- simulation waveforms,
- short description of the above.

Wherever possible publish the results selecting in the Block Diagram window File → HTML Export …. Specify the export target directory to be ...\DigDes\Reports.