HACETTEPE UNIVERSITY
ELECTRICAL AND ELECTRONICS ENGINEERING
ELE227 FUNDAMENTALS OF DIGITAL SYSTEMS LABORATORY
FALL 2019

Experiment 4 – Implementation of Synchronous Sequential Circuits

Preliminaries:

1) Students who will attend this experiment are assumed to know:
    a. Basic number systems and Boolean algebra
    b. Basic combinational logic operations and Gates
    c. Basic VHDL operators and operands
2) Study related pages in the textbook.

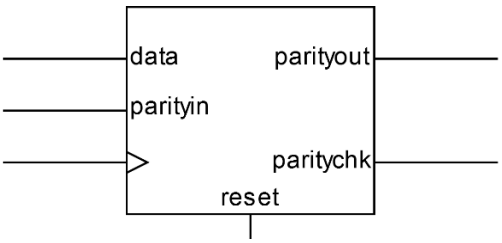
Work:

- Basic Logic Functions are "**and, or, not, nand, nor, xor, xnor**"
- **For each question use template in student database system!!!**
- Then write the VHDL code of each module and test bench of the overall circuit. Include the RTL schematics and test bench graphs.
- Insert comments in your code. Delete all unnecessary comments inserted by the software.
- Avoid too many pages. Present your work in the same sequence in this paper. The number of the answers of each question has to be included.

1) Design the below circuits. No test bench and its graph is required for this question. For each part use only the basic logic functions and the design of the previous part (except part d).
    a. SR-latch by using only basic logic gates/functions.
    b. D-latch using the SR-latch of part (a).
    c. Positive edge triggered D flip-flop by using the D-latch of part (b), do not use *if* command.
    d. Negative edge triggered T flip-flop by using JK flip-flop with process and if statements.
    e. (Optional) T flip-flop, D flip-flop and JK flip-flop from basic logic gates only.

2) Design a sequence detector that catches "1001" sequence with overlap by using Moore Model and JK flip-flops. The circuit has one-bit input and one-bit output. At each clock cycle a bit is read from the input. If the given sequence occurs for the last four clock cycles the output will be one, otherwise zero. Do not forget the reset. The VHDL code of JK flip-flop is given with process and if statements.

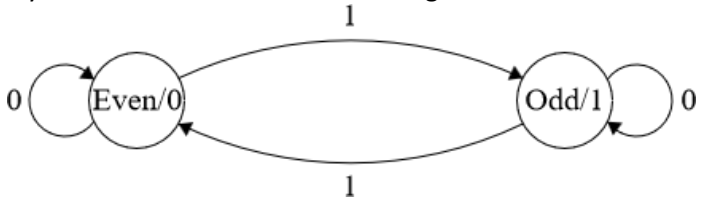
3) Design a sequential odd parity calculating and odd parity checking device as shown in the figure below. For this question you can use **any flip-flop** with process and if statements.



The input data is given to system with each clock rising edge. The parity that should be compared with calculated value is on "parityin" input. This system calculates the odd parity bit and outputs the result on "parityout" output. This value is compared with the value given

to the system originally and if the parityin and parityout are equal (i.e. parity checks out) then the paritychk is '1'.

FSM graph for parity calculator with Moore model is given below.



Test your design with following inputs separately:

Data1: 101101010110 Parityin1:1
Data2: 111100101101 Parityin2:1
Data3: 000000001010 Parityin3:0
Data4: 111111111111 Parityin4:0

4) Design a circuit that converts a D flip-flop to a JK flip-flop. Your design should use a D flip-flop as a component and implement remaining circuitry to get a black box that acts exactly as a JK flip-flop. Draw your excitation table and Karnaugh maps and get the equations for additional logic. Without excitation table, k-maps and equation your design will not get any point. For this question you can use the **D flip-flop** with process and if statements.

Important Note: As appendix major types of flip-flops are given with proper VHDL statements, **DO NOT USE** any inout ports in your design. Answers with inout usage will be ignored and get zero points.

JK Flip-Flop with process and if statements

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

entity JKFlipFlop is
    Port ( J : in  STD_LOGIC;
           K : in  STD_LOGIC;
           Reset : in  STD_LOGIC;
           Clock : in  STD_LOGIC;
           Q : out  STD_LOGIC;
           Qn : out  STD_LOGIC);
end JKFlipFlop;

architecture Behavioral of JKFlipFlop is
signal tempQ: std_logic;
begin
process(Reset,Clock)
begin
        if Clock'event and Clock = '1' then
                if Reset = '1' then
                        tempQ <= '0';
                elsif (J='0' and K='0') then
                        tempQ <= tempQ;
                elsif (J='0' and K='1') then
                        tempQ <= '0';
                elsif (J='1' and K='0') then
                        tempQ <= '1';
                elsif (J='1' and K='1') then
                        tempQ <= not (tempQ);
                end if;
        end if;
end process;
Q <= tempQ;
Qn <= not tempQ;
end Behavioral;
```

T Flip-Flop with process and if statements

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

entity TFlipFlop is
    Port ( T : in  STD_LOGIC;
           Reset : in STD_LOGIC;
           Clock : in  STD_LOGIC;
           Q : out  STD_LOGIC;
           Qn : out  STD_LOGIC);
end TFlipFlop;

architecture Behavioral of TFlipFlop is
signal tempQ: std_logic;
begin
process(Reset,Clock)
begin
        if Clock'event and Clock = '1' then
                if Reset = '1' then
                        tempQ <= '0';
                elsif T = '1' then
                        tempQ <= not tempQ;
                end if;
        end if;
end process;
Q <= tempQ;
Qn <= not tempQ;
end Behavioral;
```

D Flip-Flop with process and if statements

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

entity DFlipFlop is
    Port ( D : in  STD_LOGIC;
           Reset : in  STD_LOGIC;
           Clock : in  STD_LOGIC;
           Q : out  STD_LOGIC;
           Qn : out  STD_LOGIC);
end DFlipFlop;

architecture Behavioral of DFlipFlop is
signal tempQ: std_logic;
begin
process(Reset,Clock)
begin
        if Clock'event and Clock = '1' then
                if Reset = '1' then
                        tempQ <= '0';
                else
                        tempQ <= D;
                end if;
        end if;
end process;
Q <= tempQ;
Qn <= not tempQ;
end Behavioral;
```