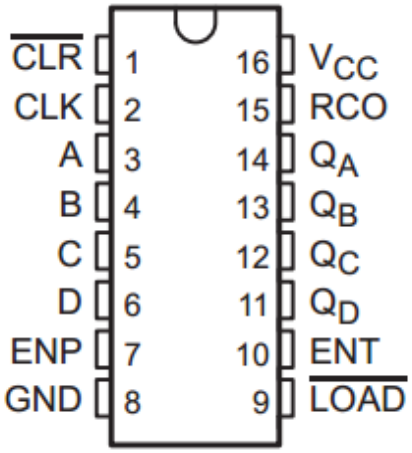
WESTERN MICHIGAN UNIVERSITY
ECE 5510 APPLICATION SPECIFIC INTEGRATED CIRCUIT DESIGN
Homework Assignment #2

Task 2:

Download the Data Sheets of the **SN74ALS163 Synchronous 4-Bit Binary Counter** from Texas Instruments' Web site. Use the available **CAD tools to design, simulate and compile a functionally** equivalent circuit on your **Xilinx Spartan-6 chip**. However, downloading the bit file to your Nexys 3 Board is **NOT** required.

Algorithm of SN74ALS163 Synchronous 4-Bit Binary Counter

The task elaborates the functionally equivalent circuit of SN74ALS163 Synchronous 4-Bit Binary Counter. To execute this task, we have some inputs and outputs. Look the figure below.



SN74ALS163

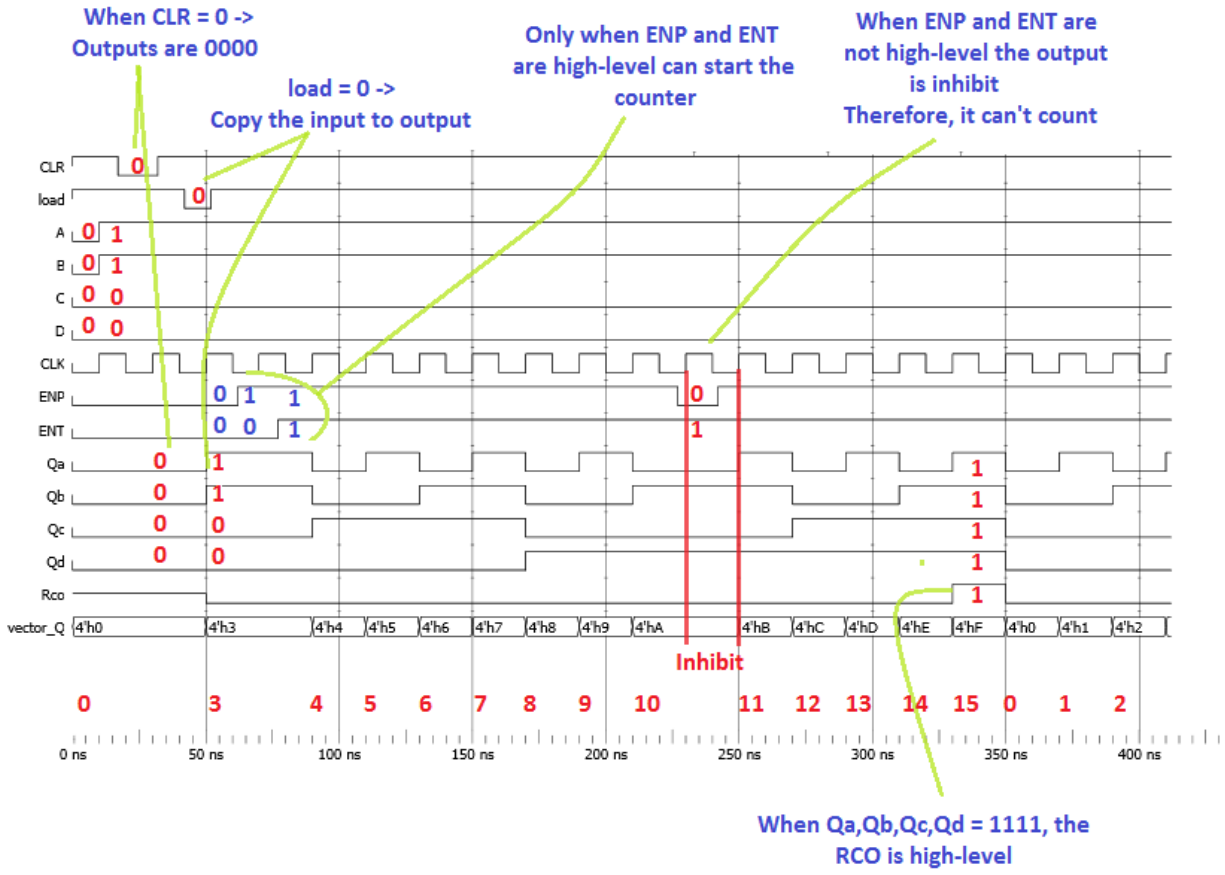
The counter can be present to any number between 0 and 15. Furthermore, it can start with any number between these numbers. It is possible because we can set up a low level at the load (LOAD). Consequently, the initial count will be the value in the input A, B, C, D. The CLR input when active in low can clear the count to 0000. Also, we have the inputs ENP and ENT that are conditional to count because both must be high to count and ENT is a conditional to enable RCO. RCO, always produces a high-level pulse while the count is 15.

I developed a finite state machine to reproduce the functionally equivalent circuit of SN74ALS163. To test the operation, I created a file .do to test the correct operation.

- I checked the CLR. When it is the low-level, the output must be 0000;
- I checked the LOAD. When the load is in the low-level, the input: A, B, C, and D is copied to the output: Qa, Qb, Qc, and Qd.
- I checked the RCO because it should be high-level when the output is 1111 = "15".
- I checked the inputs: ENP and ENT because the counter just can work when the ENP and ENT if high-level. In case it doesn't happen, the output cannot change.
- Finally, checked the all counter states.

Simulation - Wave

Look the wave below:



WESTERN MICHIGAN UNIVERSITY
ECE 5510 APPLICATION SPECIFIC INTEGRATED CIRCUIT DESIGN
Homework Assignment #2

File .vhd

Counter_vhdl.vhd

```
1
2   library IEEE;
3   use IEEE.STD_LOGIC_1164.ALL;
4
5   entity Counter_vhdl is
6      port( A, B, C, D  : in std_logic;
7            ENP         : in std_logic;
8            ENT         : in std_logic;
9            CLR         : in std_logic;
10           CLK         : in std_logic;
11           load        : in std_logic;
12           Rco         : out std_logic;
13           Qa, Qb, Qc, Qd : out std_logic);
14  end Counter_vhdl;
15
16  architecture Behavioral of Counter_vhdl is
17     type state_type is (start, load_in, clear, s0, s1, s2, s3, s4, s5, s6, s7, s8,
    s9, s10, s11, s12, s13, s14, s15);
18     signal present_state, next_state: state_type := start; --initial state idle
19     signal vector   : std_logic_vector( 3 downto 0 );
20     signal vector_Q : std_logic_vector( 3 downto 0 );
21  begin
22
23  Load_abcd: process(A, B, C, D)
24  begin
25     vector(0)<=A;
26     vector(1)<=B;
27     vector(2)<=C;
28     vector(3)<=D;
29  end process Load_abcd;
30
31  count_process: process(present_state, load, ENP, ENT, CLR, vector)
32  begin
33     case present_state is
34     when start =>
35        if (load = '0' and CLR = '1') then
36           if (vector = "0000") then     -- 0
37              next_state <= s0;
38           elsif (vector = "0001") then  -- 1
39              next_state <= s1;
40           elsif (vector = "0010") then  -- 2
41              next_state <= s2;
42           elsif (vector = "0011") then  -- 3
43              next_state <= s3;
44           elsif (vector = "0100") then  -- 4
45              next_state <= s4;
46           elsif (vector = "0101") then  -- 5
47              next_state <= s5;
48           elsif (vector = "0110") then  -- 6
49              next_state <= s6;
50           elsif (vector = "0111") then  -- 7
51              next_state <= s7;
52           elsif (vector = "1000") then  -- 8
53              next_state <= s8;
54           elsif (vector = "1001") then  -- 9
```

Counter_vhdl.vhd

```
55                next_state <= s9;
56             elsif (vector = "1010") then  -- 10
57                next_state <= s10;
58             elsif (vector = "1011") then  -- 11
59                next_state <= s11;
60             elsif (vector = "1100") then  -- 12
61                next_state <= s12;
62             elsif (vector = "1101") then  -- 13
63                next_state <= s13;
64             elsif (vector = "1110") then  -- 14
65                next_state <= s14;
66             elsif (vector = "1111") then  -- 15
67                next_state <= s15;
68             end if;
69          else
70             vector_Q <= "0000";
71          end if;
72       when load_in =>
73       when clear =>
74       when s0 =>
75          vector_Q <= "0000";
76          Rco <= '0';
77          if (ENP = '1' and ENT = '1' ) then
78             next_state <= s1;
79          else
80             next_state <= s0;
81          end if;
82       when s1 =>
83          vector_Q <= "0001";
84          Rco <= '0';
85          if (ENP = '1' and ENT = '1') then
86             next_state <= s2;
87          else
88             next_state <= s1;
89          end if;
90       when s2 =>
91          vector_Q <= "0010";
92          Rco <= '0';
93          if (ENP = '1' and ENT = '1') then
94             next_state <= s3;
95          else
96             next_state <= s2;
97          end if;
98       when s3 =>
99          vector_Q <= "0011";
100         Rco <= '0';
101         if (ENP = '1' and ENT = '1') then
102            next_state <= s4;
103         else
104            next_state <= s3;
105         end if;
106      when s4 =>
107         vector_Q <= "0100";
108         Rco <= '0';
109         if (ENP = '1' and ENT = '1') then
```

Counter_vhdl.vhd

```
110             next_state <= s5;
111          else
112             next_state <= s4;
113          end if;
114       when s5 =>
115          vector_Q <= "0101";
116          Rco <= '0';
117          if (ENP = '1' and ENT = '1') then
118             next_state <= s6;
119          else
120             next_state <= s5;
121          end if;
122       when s6 =>
123          vector_Q <= "0110";
124          Rco <= '0';
125          if (ENP = '1' and ENT = '1') then
126             next_state <= s7;
127          else
128             next_state <= s6;
129          end if;
130       when s7 =>
131          vector_Q <= "0111";
132          Rco <= '0';
133          if (ENP = '1' and ENT = '1') then
134             next_state <= s8;
135          else
136             next_state <= s7;
137          end if;
138       when s8 =>
139          vector_Q <= "1000";
140          Rco <= '0';
141          if (ENP = '1' and ENT = '1') then
142             next_state <= s9;
143          else
144             next_state <= s8;
145          end if;
146       when s9 =>
147          vector_Q <= "1001";
148          Rco <= '0';
149          if (ENP = '1' and ENT = '1') then
150             next_state <= s10;
151          else
152             next_state <= s9;
153          end if;
154       when s10 =>
155          vector_Q <= "1010";
156          Rco <= '0';
157          if (ENP = '1' and ENT = '1') then
158             next_state <= s11;
159          else
160             next_state <= s10;
161          end if;
162       when s11 =>
163          vector_Q <= "1011";
164          Rco <= '0';
```

Counter_vhdl.vhd

```
165         if (ENP = '1' and ENT = '1') then
166            next_state <= s12;
167         else
168            next_state <= s11;
169         end if;
170      when s12 =>
171         vector_Q <= "1100";
172         Rco <= '0';
173         if (ENP = '1' and ENT = '1') then
174            next_state <= s13;
175         else
176            next_state <= s12;
177         end if;
178      when s13 =>
179         vector_Q <= "1101";
180         Rco <= '0';
181         if (ENP = '1' and ENT = '1') then
182            next_state <= s14;
183         else
184            next_state <= s13;
185         end if;
186      when s14 =>
187         vector_Q <= "1110";
188         Rco <= '0';
189         if (ENP = '1' and ENT = '1') then
190            next_state <= s15;
191         else
192            next_state <= s14;
193         end if;
194      when s15 =>
195         vector_Q <= "1111";
196         Rco <= '1';
197         if (ENP = '1' and ENT = '1') then
198            next_state <= s0;
199         else
200            next_state <= s15;
201         end if;
202      end case;
203   end process count_process;
204
205   Q_process: process(vector_Q)
206   begin
207      Qa <= vector_Q(0);
208      Qb <= vector_Q(1);
209      Qc <= vector_Q(2);
210      Qd <= vector_Q(3);
211   end process Q_process;
212
213   clk_process: process(CLK)
214   begin
215      --wait until (CLK'event and CLK = '1'); --wait until the rising edge
216      if (CLK'event and CLK = '1') then
217         if (CLR = '0') then
218            present_state <= start;
219         else
```

Counter_vhdl.vhd

```
220             present_state <= next_state;
221           end if;
222         end if;
223     end process clk_process;
224
225     end Behavioral;
226
227
```

Simulation: File .do

```
#####################################
#          Macro for the SN7ALS163          #
#####################################
restart
# Generates de clock with T = 20ns
force CLK 0 0, 1 10ns -r 20ns

# Initial Information
force A 0
force B 0
force C 0
force D 0
force ENP 0
force ENT 0
force load 1
force CLR 1
# Initial condition for A, B, C, D
run 10ns
force A 1
force B 1
force C 0
force D 0
# Clear test
run 7ns
force CLR 0
run 15ns
force CLR 1

# Load of the initial condition
run 10ns
force load 0
run 10ns
force load 1
# Test of the inputs ENP and ENT
run 10ns
force ENP 1
run 15ns
force ENT 1
```

 # Maintain the count

run 150ns

Test the Inhibit

force ENP 0

run 15ns

force ENP 1

run 170ns

File .ucf

```
# PlanAhead Generated physical constraints

NET "A" LOC = T10;

# PlanAhead Generated IO constraints

NET "A" IOSTANDARD = LVCMOS33;

# PlanAhead Generated physical constraints

NET "B" LOC = T9;

# PlanAhead Generated IO constraints

NET "B" IOSTANDARD = LVCMOS33;

# PlanAhead Generated physical constraints

NET "C" LOC = V9;

# PlanAhead Generated IO constraints

NET "C" IOSTANDARD = LVCMOS33;
NET "CLK" IOSTANDARD = LVCMOS33;

# PlanAhead Generated physical constraints

NET "CLR" LOC = M8;
NET "D" LOC = N8;
NET "ENP" LOC = U8;
NET "ENT" LOC = V8;
```

```
NET "load" LOC = T5;

# PlanAhead Generated IO constraints

NET "load" IOSTANDARD = LVCMOS33;
NET "ENP" IOSTANDARD = LVCMOS33;
NET "ENT" IOSTANDARD = LVCMOS33;
NET "D" IOSTANDARD = LVCMOS33;
NET "CLR" IOSTANDARD = LVCMOS33;

# PlanAhead Generated physical constraints

NET "Qa" LOC = U16;
NET "Qb" LOC = V16;

# PlanAhead Generated IO constraints

NET "Qa" IOSTANDARD = LVCMOS33;
NET "Qb" IOSTANDARD = LVCMOS33;
NET "Qc" IOSTANDARD = LVCMOS33;
NET "Qd" IOSTANDARD = LVCMOS33;
NET "Rco" IOSTANDARD = LVCMOS33;

# PlanAhead Generated physical constraints

NET "Qc" LOC = U15;
NET "Qd" LOC = V15;
NET "Rco" LOC = M11;
NET "CLK" LOC = C9;
```