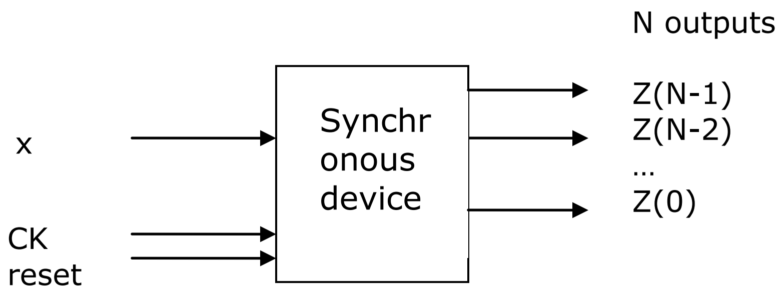
Electrical and Computer Engineering
ECE-C302

Quiz 2



```
Entity Sync_device is
Generic (N: natural := 2);
Port (X, CK, reset : in   std_logic;
              Z       : out std_logic_vector(N-1 downto 0));
End sync_device;
```

The device serially receives a string of bits at each clock rising edge. After a reset the device counts number of times a block of 2 or more consecutive ones had appeared.

Example: Let N=3, the string received after a reset is

```
  t: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 …
  X: 1 0 1 1 0 1 1 1 0 0  1  1  0  1  1  0  1  1  1  1 …
```

The output z are "000" at t=0, "001" at t=4, "010" at t=7, "011" at t=12, "100" at t=15, "101" at t=18, etc.

Note: The generic parameter N allows representation of 0 to 2^N-1 counts. Your design must be able to incorporate this generic parameter. Hint: You can add one (increment) to std_logic_vector type.

Design and show the correctness by running simulation for N=4.

```
Architecture behav of syn_device is

Signal Count : std_logic_vector(n-1 downto 0);
Type my_state is (s0, s1, s2, s3);
Signal n_s : my_state;
```

```
Begin
Z <= count; -- wire to output
Process(ck)
Variable temp : std_logic;
Begin
If ck='1' and ck'event then
    If reset = '1' then count <= (others => '0'); n_s <= s0; else
      Case n_s is
         When s0 => if x = '1' then n_s <= s1; end if;
         When s1 => if x = '1' then n_s <= s2; else n_s <= s0; end if;
         When s2 => if x = '0' then n_s <= s0; end if;
         -- with arith and unsigned package count <= count + 1;
               Temp := '1';
               For I in 0 to n-1 loop
                   Count(i) <= count(i) xor temp;
                    Temp := temp and count(i);
               End loop;
         When s2 => if x = '0' then n_s <= s0; end if;
       End case;
    End if;
End if;
End process;
```