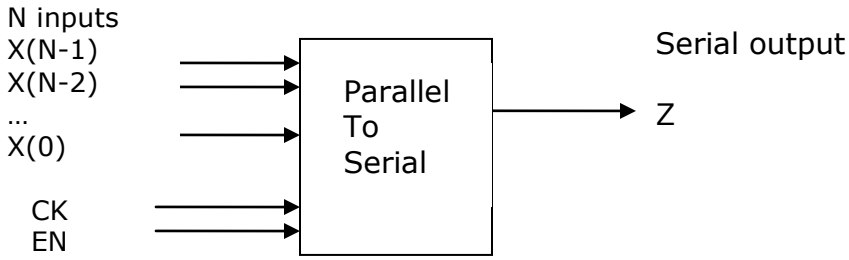
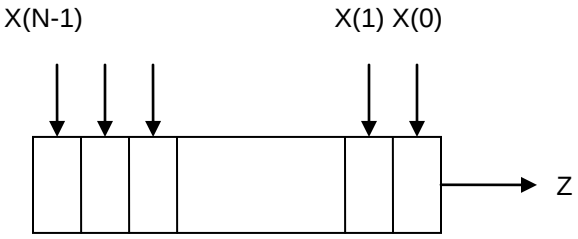
Electrical and Computer Engineering
ECE-C302

Quiz 2




```
Entity p2s is
Generic (N : natural := 4);
Port (     X : in   std_logic_vector(N-1 downto 0);
     CK, EN : in   std_logic;
          Z : out std_logic);
End p2s;
```

In converting parallel input data to serial data for transmission the device loads the input vector X into its internal register every N clock-cycles period. The device serially outputs X(0), X(1), …, X(N-1) at each clock rising edge. The figure below shows a plausible design based on internal shift-register with parallel load and shift enables.



X(0) appears at the output Z after EN changes from zero to one and X(1), …, X(N-1) on the following clock cycles. EN must remain a one at all time during the transmission. The new X(0) appears at the output Z the cycle after the old X(N-1). The output port Z is a zero when En is a zero.

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

Hint: For N=2 the internal register loads and shifts on the alternate clock cycles, and for N=3 the register loads followed by 2 shifts, periodically.

```
-- A Sketch
Architecture beh of p2s is
```

```
Type my_state is (idle, load, shift);
Signal n_s : my_state;
Signal temp: std_logic_vector(N-1 downto 0); -- place holder
Signal count : integer;
Begin
Z <= temp(0); -- wire to output
Process(ck)
Begin
If ck='1' and ck'event then
   If en = '0' then n_s <= idle; else
     When idle => if en = '1' then n_s <= load; end if;
                      Count <= 0; temp <= (others => '0'); -- clear register
     When load => temp <= x; count <= count + 1;
                       n_s <= shift;
     When shift => if count = N-1 then n_s <= load; end if;
            For I in N-1 downto 1 loop -- left shift
              Temp(i-1) <= temp(i);
            End loop;
     End case;
   End if;
End if;
End process;
End beh;
```