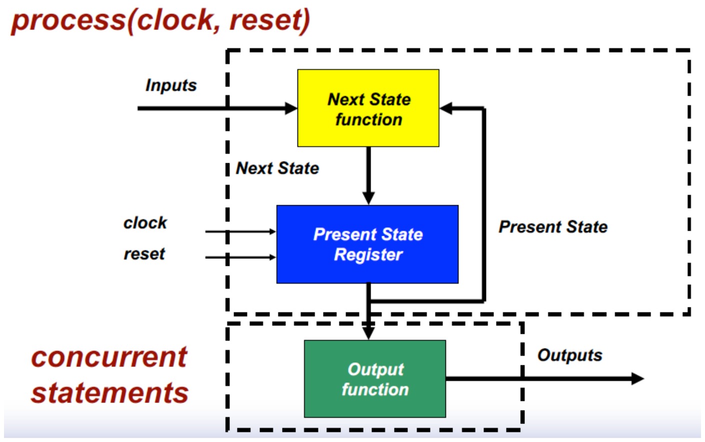
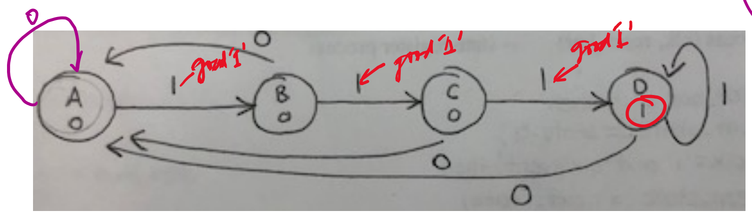
1. Let's construct the sequence detector for the sequence ***3 or more consecutive 1's*** using **Moore** state machine. The Output of the State machine depends only on present state. The output of state machine are only updated at the clock edge. ***The FSM uses an " asynchronous" positive reset which has a higher priority than the clock.***



The Moore state machine require **four states *a, b, c,* and *d*** to detect the ***3 or more consecutive 1's*** sequence.

a) Draw the state transition graph (STG).

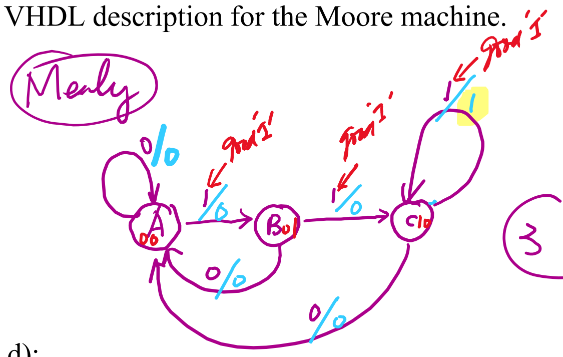


Moore

4 states

b) Complete the entity and architecture pair of VHDL description for the Moore machine.

```
library ieee;
use ieee.std_logic_1164.all;
entity seq_detect is
        port (clk, reset, x : in std_logic;
                y : out std_logic);
end;

architecture enum of seq_detect is
type state is (state_a, state_b, state_c, state_d);

signal present_state, next_state : state;

begin
        nx_state: process (present_state, x) -- next state process
        begin
        -- use CASE and then IF statement
        case present_state is
```



3 states

Flip-Flop A, B

A=(00), B(01), C(1,0)

	A_p	B_p	X	A_N	B_N	Y
A	0	0	0	0	0	0
	0	0	1	0	1	0
B	0	1	0	0	0	0
	0	1	1	1	0	0
C	1	0	0	0	0	0
	1	0	1	1	0	1
	1	1	0	X	X	X
	1	1	1	X	X	X

→ X don't care

A_N (B_p, A_p, X)

	0	0	1	0
A_p	0	1	X	X

$A_N = A_pX + B_pX$

B_N (B_p, A_p, X)

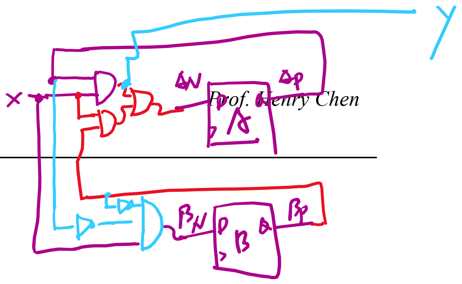
	0	1	0	0
A_p	0	0	X	X

$B_N = A_p' B_p' X$

Y (B_p, A_p, X)

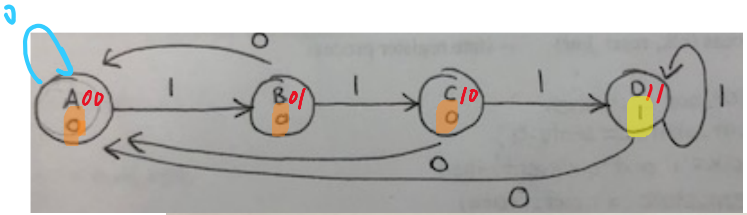
	0	0	0	0
A_p	0	1	X	X

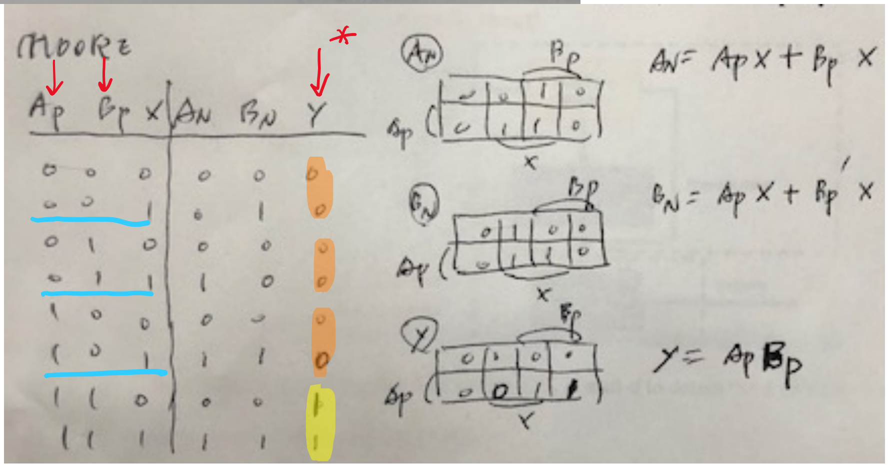
$Y = A_p X$



```
                        when state_a =>
                                if x = '1' then
                                        next_state <= state_b;
                                else
                                        next_state <= state_a;
                                end if;
                        when state_b =>
                                if x = '1' then
                                        next_state <= state_c;
                                else
                                        next_state <= state_a;
                                end if;
                        when state_c =>
                                if x = '1' then
                                        next_state <= state_d;
                                else
                                        next_state <= state_a;
                                end if;
                        when state_d =>
                                if x = '1' then
                                        next_state <= state_d;
                                else
                                        next_state <= state_a;
                                end if;
                end case;
        end process nx_state;

        state_reg: process (clk, reset_bar)      -- state register process
        begin
                if (reset = '1') then
                        present_state <= state_a;
                elsif  (clk'EVENT AND clk = '1') then
                        present_state <= next_state;
                end if;

        end process state_reg;

        -- Concurrent VHDL for output assignment

        y <= '1' WHEN (present_state = state_d) ELSE '0';



    end enum;
```

c) Construct **state table** for the Moore machine. There are 4 states, a, b, c, and d. You need 2 flip-flops, A and B for implementation. Use state assignment (A B) = (0 0) for state **a**, (0 1) for state **b**, (1 0) for state **c**, and (1 1) for state **d**. Note: In your state table, use <u>A_P for the present state A</u>, <u>B_P for the present state B</u>, <u>A_N for the next state A</u>, <u>B_N for the next state B</u>, <u>x for the input</u>, and <u>y for the output</u>. **Derive Boolean equation for the next state A_N, B_N, and the output y.**





MOORE

A_P	B_P	x	A_N	B_N	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

A_N

0	0	1	0
0	1	1	0

$A_N = A_P x + B_P x$

B_N

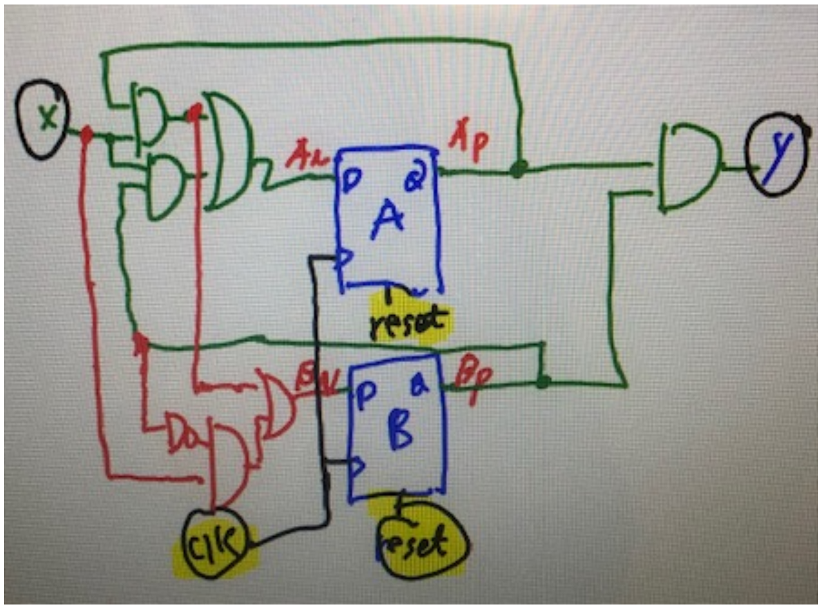
0	1	0	0
0	1	1	0

$B_N = A_P x + B_P' x$

Y

0	0	0	0
0	0	1	1

$Y = A_P B_P$


x
A_N
A_P
D
Q
A
reset
B_N
B_P
D
Q
B
clk
reset
y