Uygulama 6) State Machine

Bu deneyde, state machine diyagramı olarak verilen algoritma VHDL ile yazılacak ve **ALTERA-DE0** kartı üzerinde çalıştırılacaktır. (**tasarlanması istenilen algoritma deney sırasında tahtaya çizilecektir.**)

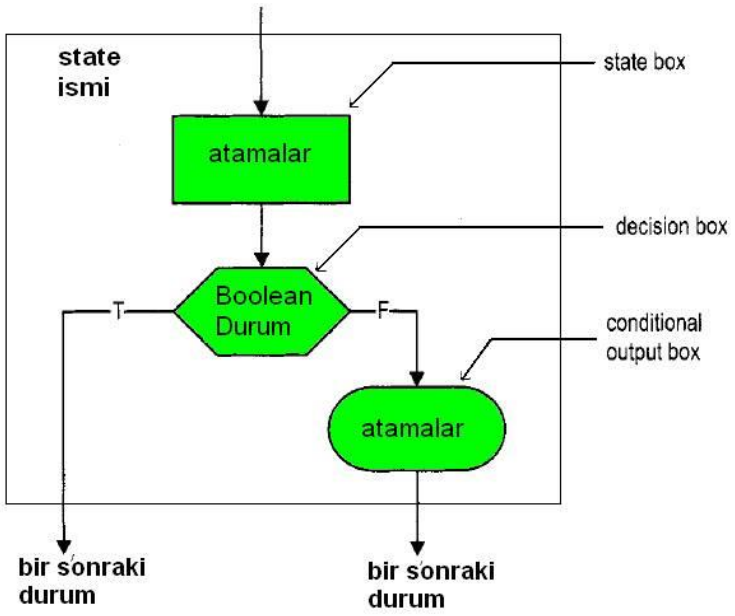
	Family:	Name:
ALTERA-DE0	Cyclone III	EP3C16F484C6

6a) State Machine

ASM chart, sistem akışını gösteren diyagramdır. State(durum) box, desicion box ve conditional output box larından oluşur.

Atama kutularında giriş ve çıkışlara göre yapılacak atama işlemleri veya değişkenler arasında matematiksel işlemler yapılabilir.

Karar kutusunda ise şart tanımlanır. İf kutusu gibi düşünülebilir.

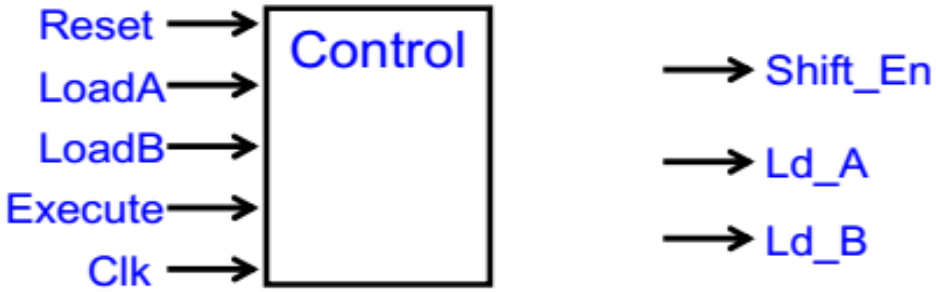


Aşağıda çizilen örnek state machine diyagramını VHDL kodu ile gerçekleştiriniz. 3 ana bölüm olarak düşününüz.;

1. Bölüm: başlangıç ayarları ve reset işlemini yürütür.
2. Bölüm: stateler arası geçiş ve sistem akışını kontrol eder.
3. Bölüm: statelerde yapılacak işlemleri ve atamaları kontrol eder.

State machine VHDL örnek kod;

Example: Control Unit




```
entity control is
  Port ( Reset, LoadA, LoadB, Execute : in std_logic;
         Clk : in std_logic;
         Shift_En, Ld_A, Ld_B : out std_logic);
end control;

architecture Behavioral of control is
--declare A, B, ..., F of type cntrl_state
--User defined type "cntrl_state" has 6 symbolic values.
type cntrl_state is (A, B, C, D, E, F);
--declare signals state and next_state of type cntrl_state
signal state, next_state : cntrl_state;

begin
        control_reg: process (Reset, Clk, LoadA, LoadB)
        begin
                if (Reset = '1') then
                        state <= A;
                elsif (rising_edge(Clk)) then
                        state <= next_state;
                end if;
        end process;

        --Assign 'next_state' based on 'state' and 'Execute'
        get_next_state: process (Execute, state)
        begin
                case state is
                        when A =>
                                if (Execute = '1') then
                                        next_state <= B;
                                  else
                                        next_state <= A;
                                end if;
```

```
                        when B =>
                                next_state <= C;
                        when C =>
                                next_state <= D;
                        when D =>
                                next_state <= E;
                        when E =>
                                next_state <= F;
                        --wait at state F until 'Execute' = 0
                        when F =>
                                if (Execute = '0') then
                                        next_state <= A;
                                        else
                                        next_state <= F;
                                end if;

-- "when others =>" default case is not needed here since there are
    -- only six values for "state" and we have exhausted them all.

                        end case;
        end process;

        --Assign outputs based on 'state'
        get_cntrl_out: process (LoadA, LoadB, state)
        begin
                case state is
                        when A =>
                                --Only load value in register(s) when in state A
                                Ld_A <= LoadA;
                                Ld_B <= LoadB'
                                Shift_En <= '0';

                        --No Load or Shift when in state F
                        when F =>
                                Ld_A <= '0';
                                Ld_B <= '0';
                                Shift_En <= '0';

                        when others =>      --This is used for states B, C, D, and E
                                Ld_A <= '0';
                                Ld_B <= '0';
                                Shift_En <= '1';
                end case;
        end process;
end Behavioral;
```