SORU: (MIT LOGIC I 2009 FİNAL SORUSU *bize uyarlanmış hali***)**

Bir teknoloji şirketinde çalışan bir mühendissiniz ve ürettiğiniz bir robotu Gazi Üniversitesi Maltepe Kampüsünde test edecek bir FSM tasarlamanız isteniyor. Şirketiniz, seyehat planlarını robotunuza kablosuz olarak gönderiyor ve robotunuz bu bilgilere göre hareket ediyor.

FSM'i tasarlarken öncelikle kampüsünüzden aşağıdaki lokasyonları seçiyorsunuz ve her bir lokasyonu 3 – bitlik ikili bir durumla eşliyorsunuz:

2. KAT EE DERSLİKLER: [000];

MUSTAFA AVCU KONFERANS SALONU: [001];

KUTDAŞ KANTİN: [010];

LİMON [011];

BAHÇE [100];

AKADEMİK BLOK [101];

LOGIC LABI [110];

-1. KAT MELİKE ABLA [111];

Testi basitleştirmek için şirketinize robotun FSM'ine ikili bir seyahat planı göndermelerini söylüyorsunuz (örneğin, "1-0-0-0-1", robotu beş kez hareket ettirir). Yani, robot her bir hareket için ya "0" ya da "1" alıyor ve aşağıdaki tabloya göre belirtilen bir sonraki hedefe gidiyor. Her test çalışmasında robot başlangıçta **2. Kat ee derslikler** başlıyor ve FSM, robotun mevcut konumunu çıktı olarak veriyor.

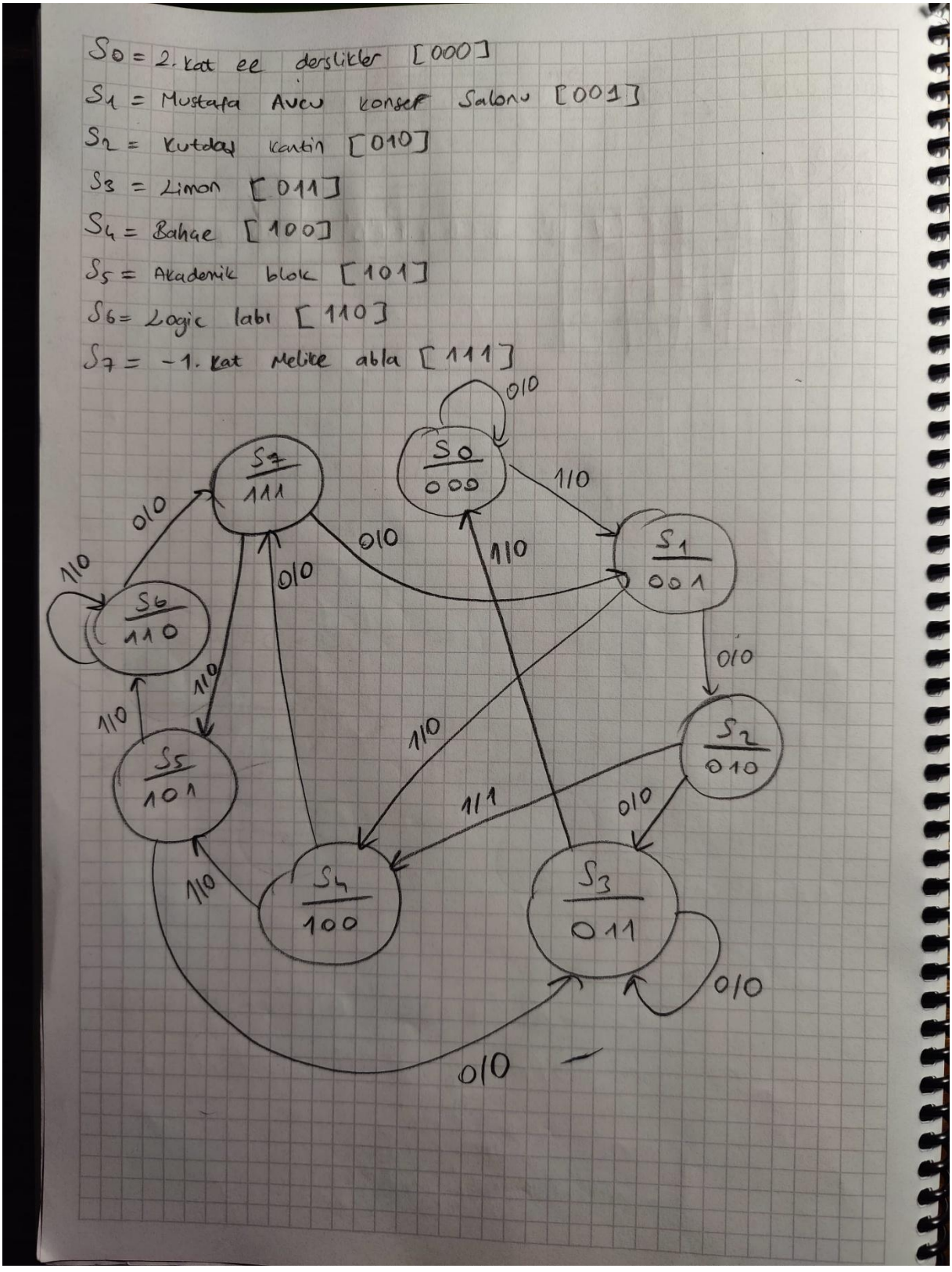
Durum Geçiş Kuralları:

- **2. Kat ee derslikler:**
  - Eğer "0" alırsa, dersliklerde kalır.
  - Eğer "1" alırsa, konferans salonuna gider.
- **Mustafa Avcu Konferans Salonu:**
  - Eğer "0" alırsa, Kutdaş'a gider.
  - Eğer "1" alırsa, bahçeye gider.
- **Kutdaş Kantin:**
  - Eğer "0" alırsa, Limon'a gider.
  - Eğer "1" alırsa, bahçeye gider.
- **Limon:**
  - Eğer "0" alırsa, Limon'da kalır.
  - Eğer "1" alırsa, ee dersliklere gider.
- **Bahçe:**
  - Eğer "0" alırsa, Melike ablaya gider.
  - Eğer "1" alırsa, akademik bloğa gider.
- **Akademik Blok:**
  - Eğer "0" alırsa, Limon'a gider.
  - Eğer "1" alırsa, 6.111 logic labına gider.
- **Logic labı:**
  - Eğer "0" alırsa, Melike ablaya gider.
  - Eğer "1" alırsa, logic labında kalır.
- **-1. Kat Melike Abla**
  - Eğer "0" alırsa, konferans salonuna gider.

- Eğer "1" alırsa, akademik bloğa gider.

a) Durum diyagramını çiziniz.
b) Design kodunu tasarlayınız.

a. Tablonun da yardımıyla state diagram şöyle görünmelidir:



Design kodunu yazacak olursak:

```
module kutdas (
    input logic clk,
    input logic rstn,
    input logic [2:0] din,
    output logic [2:0] dout
);

  typedef enum logic [2:0]
  {
    S0 = 3'b000, // 2. kat EE derslikler
    S1 = 3'b001, // Mustafa Avcı Konser Salonu
    S2 = 3'b010, // Kutdaş Kantin
    S3 = 3'b011, // Limon
    S4 = 3'b100, // Bahçe
    S5 = 3'b101, // Akademik Blok
    S6 = 3'b110, // Logic Lab
    S7 = 3'b111  // -1. Kat Melike Abla
  } state_t;

  state_t current_state, next_state;

  always_ff @(posedge clk or negedge rstn) begin
    if (!rstn) begin
      current_state <= S0;
    end else begin
      current_state <= next_state;  // Durum geçişi
    end
  end

  // FSM geçiş mantığı için "always_comb"
  always_comb begin
    next_state = current_state;  // Varsayılan olarak mevcut durumu koru
    dout = 3'b000;               // Varsayılan çıkış

    case (current_state)
      S0: begin
        if (din == 3'b001) begin
          next_state = S1;  // 1 geldi, Mustafa Avcı Konser Salonu'na geç
          dout = 3'b001;
        end
      end

      S1: begin
        if (din == 3'b010) begin
          next_state = S2;  // 0 geldi, Kutdaş Kantin'e geç
          dout = 3'b010;
        end else if (din == 3'b100) begin
```

```
          next_state = S4;  // Bahçe'ye geç
          dout = 3'b100;
        end
      end

      S2: begin
        if (din == 3'b011) begin
          next_state = S3;  // 0 geldi, Limon'a geç
          dout = 3'b011;
        end else if (din == 3'b100) begin
          next_state = S4;  // Bahçe'ye geç
          dout = 3'b100;
        end
      end

      S3: begin
        if (din == 3'b000) begin
          next_state = S0;  // Sıfır geldi, başa dön
          dout = 3'b000;
        end else if (din == 3'b011) begin
          next_state = S3;  // Kendine dön
          dout = 3'b011;
        end
      end

      S4: begin
        if (din == 3'b111) begin
          next_state = S7;  // -1. Kat Melike Abla'ya geç
          dout = 3'b111;
        end else if (din == 3'b101) begin
          next_state = S5;  // Akademik Blok'a geç
          dout = 3'b101;
        end
      end

      S5: begin
        if (din == 3'b011) begin
          next_state = S3;  // Limon'a dön
          dout = 3'b011;
        end else if (din == 3'b110) begin
          next_state = S6;  // Logic Lab'e geç
          dout = 3'b110;
        end
      end

      S6: begin
        if (din == 3'b111) begin
          next_state = S7;  // -1. Kat Melike Abla'ya geç
          dout = 3'b111;
```

```
        end
      end

      S7: begin
        if (din == 3'b001) begin
          next_state = S1;  // Mustafa Avcı Konser Salonu'na geç
          dout = 3'b001;
        end else if (din == 3'b101) begin
          next_state = S5;  // Akademik Blok'a geç
          dout = 3'b101;
        end
      end

      default: begin
        next_state = S0;  // Herhangi bir durum dışına çıkıldığında S0'a dön
        dout = 3'b000;
      end
    endcase
  end

endmodule

/*
Bu örnekte clock kısmını always_ff@()bloğunun içinde tanımlamışken state
geçişlerini
always_comb bloğunda tanımladık. İstersek always bloğunun içinde de
yazabilirdik, yanlış olmazdı. Fakat
always_comb bloğu kullanmak hem kodun daha düzgün ve anlaşılır olmasına hem de
sentezleme işlemi sırasında
flip flop kaynaklı oluşabilecek hataların önüne geçmek amacıyla katkı sağlar.
*/
```