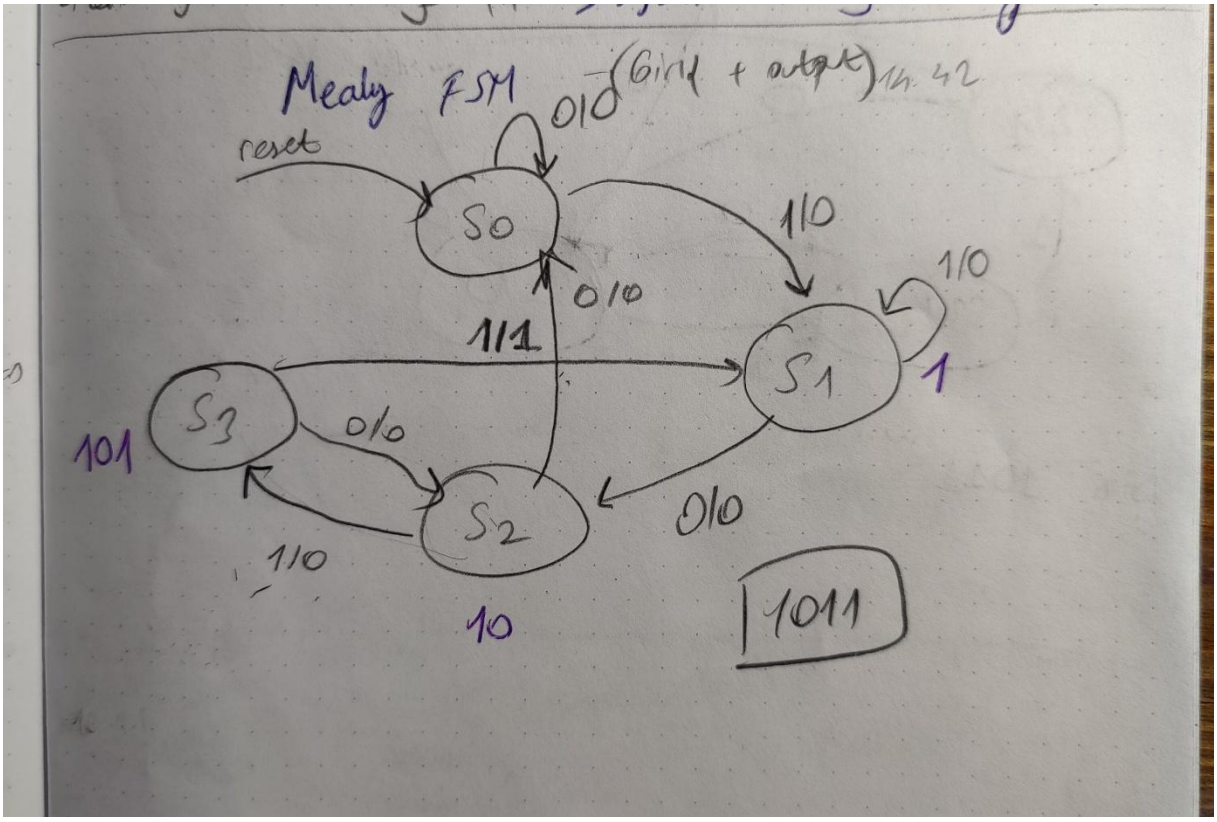
ÖRNEK 1: “1011” dizisi gelince “1” olan bir a) Mealy ve b) Moore tasarlayınız.

a) Önce state diagram’ımızı çizelim:



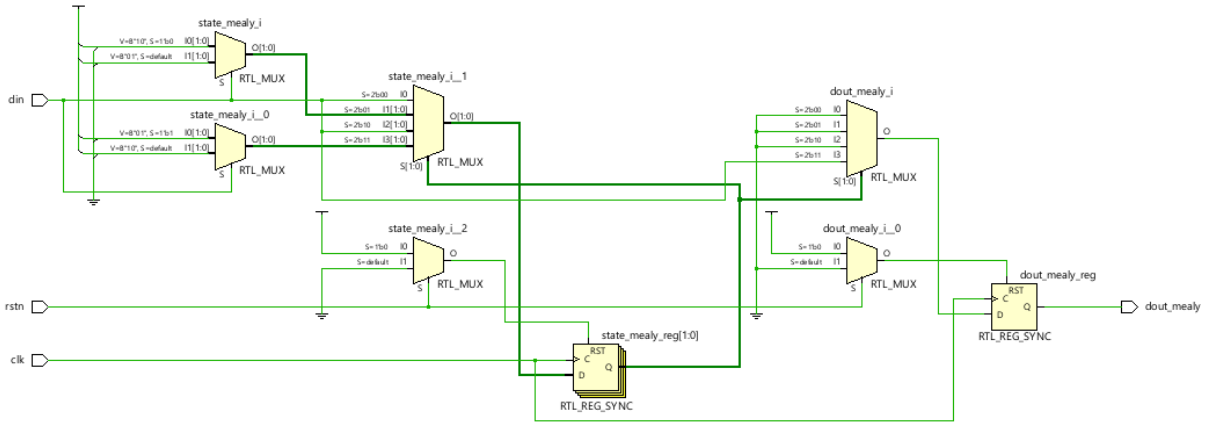
Artık design kodunu yazmaya başlayabiliriz:

```
module mealy_1011(
    input logic clk,
    input logic rstn,
    input logic din,
    output logic dout_mealy
  );

  integer counter;
  typedef enum logic [1:0]  //typedef enum, bir dizi sabit ismi bir araya
getirerek numaralandırılmış bir veri tipi oluşturur.
          {
            S0_mealy, //00
            S1_mealy, //01
            S2_mealy, //10
            S3_mealy //11
          } states_mealy;  //numerated tipin adı

  states_mealy state_mealy;

  always_ff @(posedge clk)
  begin
```

```
    if(!rstn)
    begin
      state_mealy <= S0_mealy;
      dout_mealy <= 1'b0;
    end

    else
    begin // din = 1 ; !din = 0
      case (state_mealy)
        S0_mealy :
        begin
          if (din) // 1
          begin
            state_mealy <= S1_mealy; // 1 gelirse S1 e geç
            dout_mealy <= 1'b0;
          end
          else
          begin
            state_mealy <= S0_mealy; //1 gelmezse kendisinde kalsın
            dout_mealy <= 1'b0;
          end
        end
        S1_mealy :
        begin
          if(!din) // 10 -> 0
          begin
            state_mealy <= S2_mealy; // 0 gelirse S2 ye geç
            dout_mealy <= 1'b0;
          end
          else
          begin
            state_mealy <= S1_mealy; // 0 gelmezse kendisinde kalsın
            dout_mealy <= 1'b0;
          end
        end

        S2_mealy:
        begin
          if(din) // 101 -> 1
          begin
            state_mealy <= S3_mealy; // 1 gelirse S3 e geç
            dout_mealy <= 1'b0;
          end
          else
          begin
            state_mealy <= S0_mealy; // 1 gelmezse S0 a geç --> "0" örüntünün
hiçbir şekilde başı olamayacağı için başa dönüş yapılır.
            dout_mealy <= 1'b0;
          end
```

```
        end

        S3_mealy:
        begin
          if(din) // 1011 -> 1
          begin
            state_mealy <= S1_mealy; // 1 gelirse S1 e geç --> yeni oluşacak
örüntünün başlangıç potansiyelini taşır.
            dout_mealy <= 1'b1;
          end
          else
          begin
            state_mealy <= S2_mealy; // 1 gelmezse S2 ye geç
            dout_mealy <= 1'b0;
          end
        end

        default :
        begin
          state_mealy <= S1_mealy;
          dout_mealy <= 1'b0;
        end
      endcase
    end
  end

    // S3 ün 1 gelmesi durumuna kadar olan bütün dout ların "0" olmasının
sebebi dizinin henüz tamamlanmamış olmasıdır.
    // Dizi tamamlanınca dout 1 i döndürür.

endmodule
```

Design çıktımız aşağıdaki gibi olacaktır:

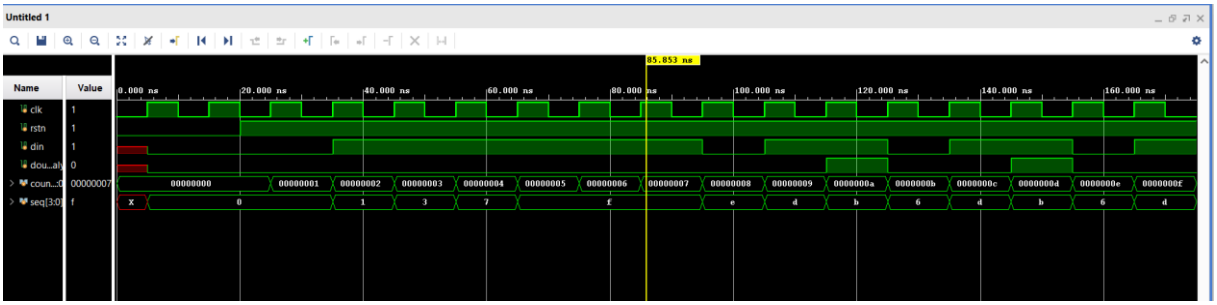


Testbanch kodunu yazmaya başlayabiliriz:

```
module tb_mealy_1011();
  logic clk = 0;
  logic rstn;
  logic din;
  logic dout_mealy;
  integer counter = 0;

  mealy_1011 mealy_1011_Inst(.*);

  always #5 clk <= ~clk;

  initial begin
    rstn <= 1'b0;
    @(negedge clk);
    @(negedge clk);
    rstn <= 1'b1;
  end

  always_ff @(posedge clk)
  begin
    if(!rstn)
    begin
      din <= 0;
    end
    else
    begin
      din = $random(); // rastgele 0 veya 1 üretir.
    end
  end

  logic [3:0] seq;  // giriş verilerinin (din sinyalinin) son 4 bitini
kaydetmek için kullanılan bir kaydırma (shift) kaydı gibi davranır.

  always_ff @(posedge clk)
  begin
    if (!rstn)
    begin
      seq <= 4'b0;
    end
    else  // kaydırma işlemiyle son 4 sinyali kaydeder.
    begin
      seq <= {seq[2:0], din};
      $display("din = %b, seq: %b, dout_mealy: %b", din, seq, dout_mealy);
      counter = counter + 1;

      if(counter == 16)
      begin
```

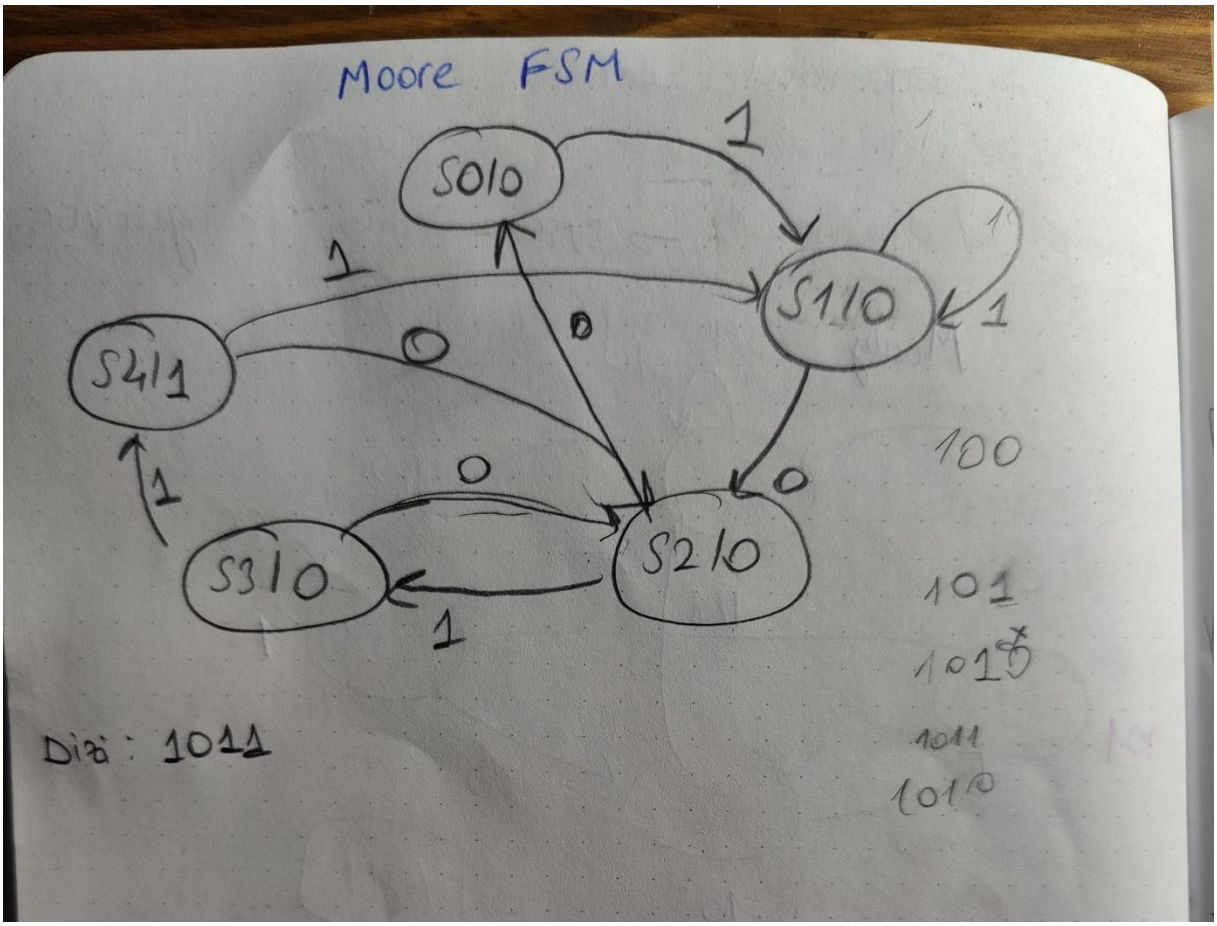
```
        $stop;  // 16 tane sinyal üretildikten sonra dursun.
      end
    end
  end
endmodule

/*
kaydırma işlemi için örnek:

İlk durumda: seq = 4'b0000 (sıfırlanmış)
Eğer din = 1 ise: seq = 4'b0001
Bir sonraki din = 0 ise: seq = 4'b0010
Bir sonraki din = 1 ise: seq = 4'b0101
*/
```

Simülasyon çıktılarımız aşağıdaki gibidir:




```
# run 1000ns
din = 0, seq: 0000, dout_mealy: 0
din = 1, seq: 0000, dout_mealy: 0
din = 1, seq: 0001, dout_mealy: 0
din = 1, seq: 0011, dout_mealy: 0
din = 1, seq: 0111, dout_mealy: 0
din = 1, seq: 1111, dout_mealy: 0
din = 1, seq: 1111, dout_mealy: 0
din = 0, seq: 1111, dout_mealy: 0
din = 1, seq: 1110, dout_mealy: 0
din = 1, seq: 1101, dout_mealy: 0
din = 0, seq: 1011, dout_mealy: 1
din = 1, seq: 0110, dout_mealy: 0
din = 1, seq: 1101, dout_mealy: 0
din = 0, seq: 1011, dout_mealy: 1
din = 1, seq: 0110, dout_mealy: 0
din = 0, seq: 1101, dout_mealy: 0
$stop called at time : 175 ns : File "C:/Users/zulal/OneDrive/Desktop/lc
INFO: [USF-XSim-96] XSim completed. Design snapshot 'tb_mealy_1011_behav
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:03 ; elapsed = 00:00:05 . Memor
```

b) State diagram’ımızı çizelim:



Moore’da çıktılar sadece durumun bir fonksiyonu olduğu için durum diyagramında outputlar yok ve mealy’e bir fark olarak diziyi bitirmek amacıyla “S4” durum baloncuğu koyulmuştur.

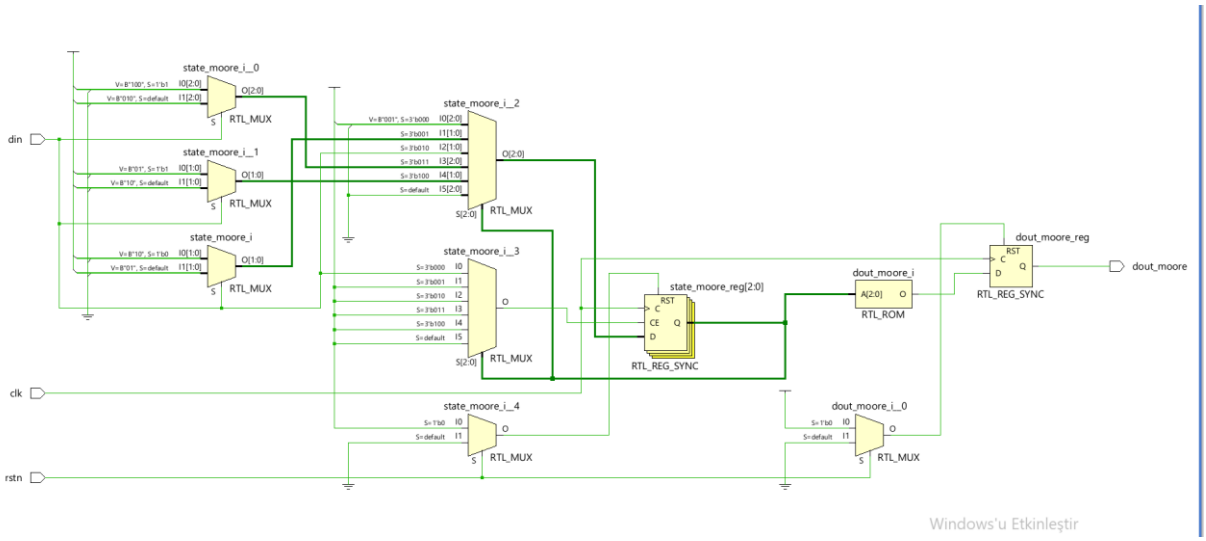
Artık design kodunu yazabiliriz:

```
module moore_1011(
    input logic clk,
    input logic rstn,
    input logic din,
    output logic dout_moore
    );

    typedef enum logic [2:0]
    {
        S0_moore, //000
        S1_moore, //001
        S2_moore, //010
        S3_moore, //011
        S4_moore //100
    } states_moore;

    states_moore state_moore;

    always_ff @(posedge clk)
    begin
        if(!rstn)
        begin
            state_moore <= S0_moore;
            dout_moore <= 1'b0;
        end

        else
        begin
            case(state_moore)
            S0_moore:
            begin
                dout_moore <= 1'b0;
                if(din)
                begin
                    state_moore <= S1_moore;
                end
            end

            S1_moore:
            begin
                dout_moore <= 1'b0;
                if(!din)
                begin
                    state_moore <= S2_moore;
                end

                else
```

```
            begin
                state_moore <= S1_moore;
            end
        end

        S2_moore:
        begin
            dout_moore <= 1'b0;
            if(din)
            begin
                state_moore <= S3_moore;
            end

            else
            begin
                state_moore <= S0_moore;
            end
        end

        S3_moore:
        begin
            dout_moore <= 1'b0;
            if(din)
            begin
                state_moore <= S4_moore;
            end

            else
            begin
                state_moore <= S2_moore;
            end
        end

        S4_moore:
        begin
            dout_moore <= 1'b1;
            if(din)
            begin
                state_moore <= S1_moore;
            end

            else
            begin
                state_moore <= S2_moore;
            end
        end

        default:
        begin
```

```
                state_moore <= S0_moore;
                dout_moore <= 1'b0;
            end
            endcase
        end
    end
endmodule
```

Design çıktımız aşağıdaki gibidir:

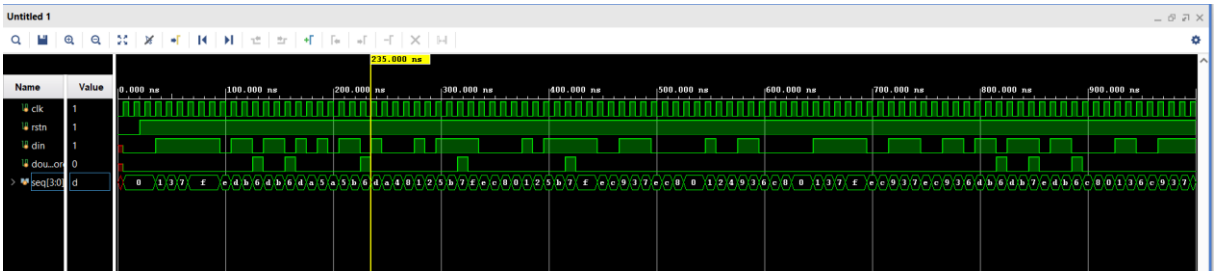


Testbanch kodu:

```
module tb_moore_1011();
  logic clk = 0;
  logic rstn;
  logic din;
  logic dout_moore;

  moore_1011 moore_1011_Inst(.*);

  always #5 clk <= ~clk; // non-blocking yapılarda küçüktürü unutma!

  initial
  begin
    rstn <= 1'b0;
    @(negedge clk);
    @(negedge clk);
    rstn <= 1'b1;
  end

  always_ff @(posedge clk)
  begin
    if(!rstn)
```

```
    begin
      din <= 0;
    end
    else
    begin
      din = $random(); // rastgele 0 veya 1 üretir.
    end
  end

  logic [3:0] seq;

  always_ff @(posedge clk)
  begin
    if (!rstn)
    begin
      seq <= 4'b0;
    end
    else
    begin
      seq <= {seq[2:0], din};
      $display("din = %b, seq: %b, dout_moore: %b", din, seq, dout_moore);
    end
  end
endmodule
```

Çıktılar şu şekilde olacaktır:




```
# run 1000ns
din = 0, seq: 0000, dout_moore: 0
din = 1, seq: 0000, dout_moore: 0
din = 1, seq: 0001, dout_moore: 0
din = 1, seq: 0011, dout_moore: 0
din = 1, seq: 0111, dout_moore: 0
din = 1, seq: 1111, dout_moore: 0
din = 1, seq: 1111, dout_moore: 0
din = 0, seq: 1111, dout_moore: 0
din = 1, seq: 1110, dout_moore: 0
din = 1, seq: 1101, dout_moore: 0
din = 0, seq: 1011, dout_moore: 0
din = 1, seq: 0110, dout_moore: 1
din = 1, seq: 1101, dout_moore: 0
din = 0, seq: 1011, dout_moore: 0
din = 1, seq: 0110, dout_moore: 1
```