Tahmini Ders İçeriği
(Tentative Couse Schedule – Syllabus)

TOBB ETÜ
Ekonomi ve Teknoloji Üniversitesi

1. Hafta: Sayısal Sinyaller/Sistemler, İkilik Tabanda Sayılar, Taban Aritmetiği, İşaretli/Eksi Sayıların Gösterimi, Sayısal Tasarım Tarihçesi

2. Hafta: İkili Mantık Aritmetiği ve Kapıları, Bool Cebiri Teorisi ve Tanımları, Bool Fonksiyonları, Kapı-Seviyesinde Yalınlaştırma, Karnough Haritası, Önemsenmeyen Durumlar, NAND, NOR, XOR

3-4. Hafta: FPGA, Birleşik (Combinational) Devreler, Aritmetik Modüller, Decoder, Encoder, Mux, Verilog HDL

5. Hafta: Ardışık (Sequential) Devreler, Mandal (Latch), Flip-Flop, Yazmaçlar (Registers)

Lab Sınavı (265/264L) **(19 Ekim)**

6. Hafta: **(17-21 Ekim)** Durum Makinaları, Örnek Tasarımlar, Sayaçlar (Counters)

7. Hafta: **(24-28 Ekim)** RTL (Register Transfer Level) ASMD (Algorithmic State Machine and Datapath) Tasarımları

8. Hafta: **(31 Ekim – 4 Kasım)** Bellekler, FPGA'da Block RAM, OpenRAM

Ara Sınav (265/264) **(15 Kasım)**

9. Hafta: **(7-11 Kasım)** Durağan Zaman Analizi (Static Timing Analysis)

10. Hafta: **(14-18 Kasım) ???**

11-12. Hafta: **(21-25 Kasım, 28 Kasım – 2 Aralık)** Boru hattı, FPGA ve ASIC Tasarım Akışları

Final **(Aralık)** – Proje Teslimleri **(18 Aralık)**

SONLU DURUM MAKİNELERİ (FINITE STATE MACHINES)

TOBB ETÜ
Ekonomi ve Teknoloji Üniversitesi

Neden “sonlu durum” → Çünkü n adet FF ile 2^n kadar durum meydana getirilebiliyor

Durum tablosu oluşturma → Giriş sinyalleri ve şu anki duruma (present state) göre sonraki durum (next state) ve çıkış sinyallerini hesaplamak

Her bir saat vuruşunda, FF’lar sonraki durum (next state) değerini alırlar ve buna göre de çıkış sinyalleri güncellenir

2 tür sonlu durum makinesi vardır: Moore – Mealy

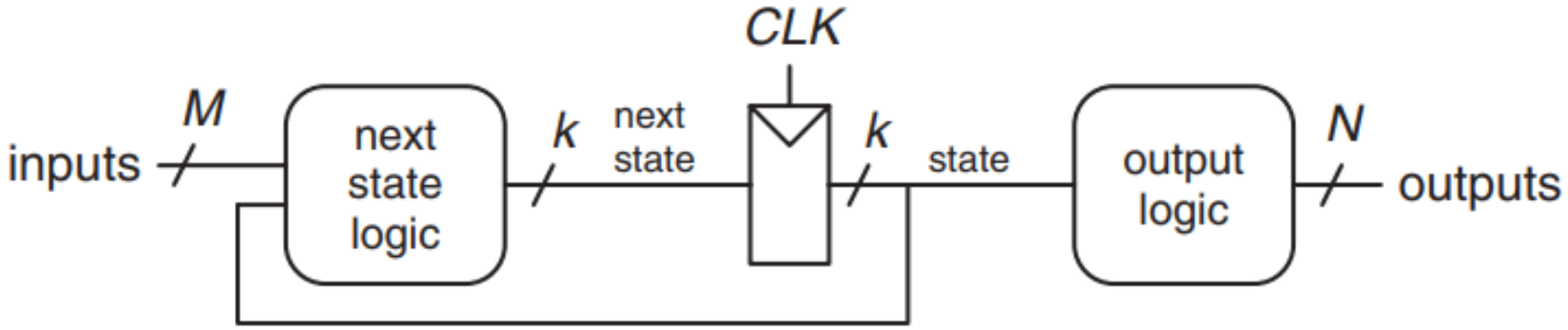
Moore makinelerinde, çıkış sinyalleri sadece devrenin o anki durumuna bağlıdır

Mealy makinelerinde, çıkış sinyalleri hem devrenin o anki durumuna, hem de giriş sinyallerine bağlıdır

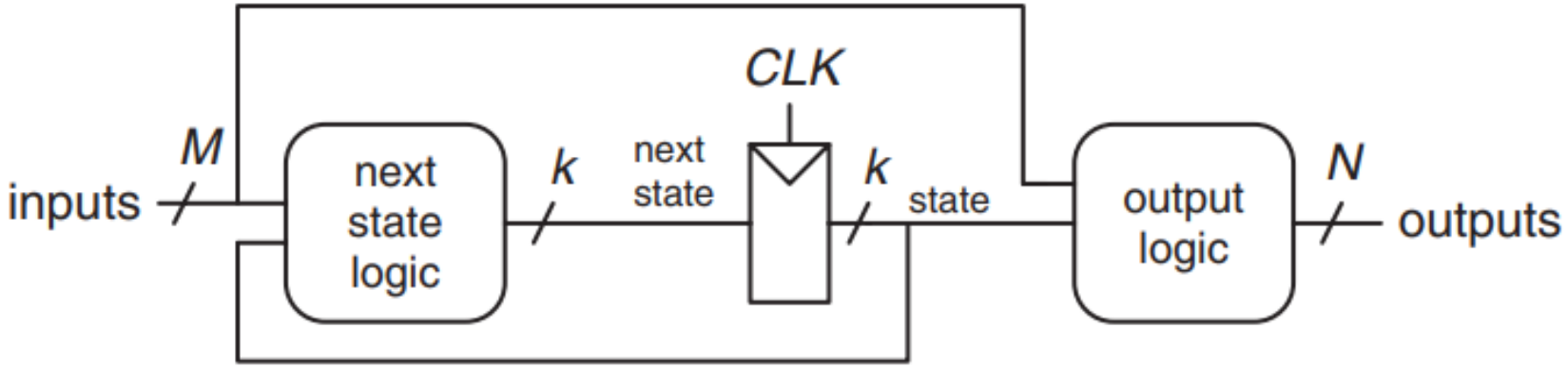
Moore ve Mealy bu yöntemleri 1955/56 yıllarında ortaya atmışlardır

Edward F. Moore (1925-2003) → Bell Lab, University of Wisconsin
George H. Mealy (1927-2010) → Bell Lab, Harvard University

SONLU DURUM MAKİNELERİ

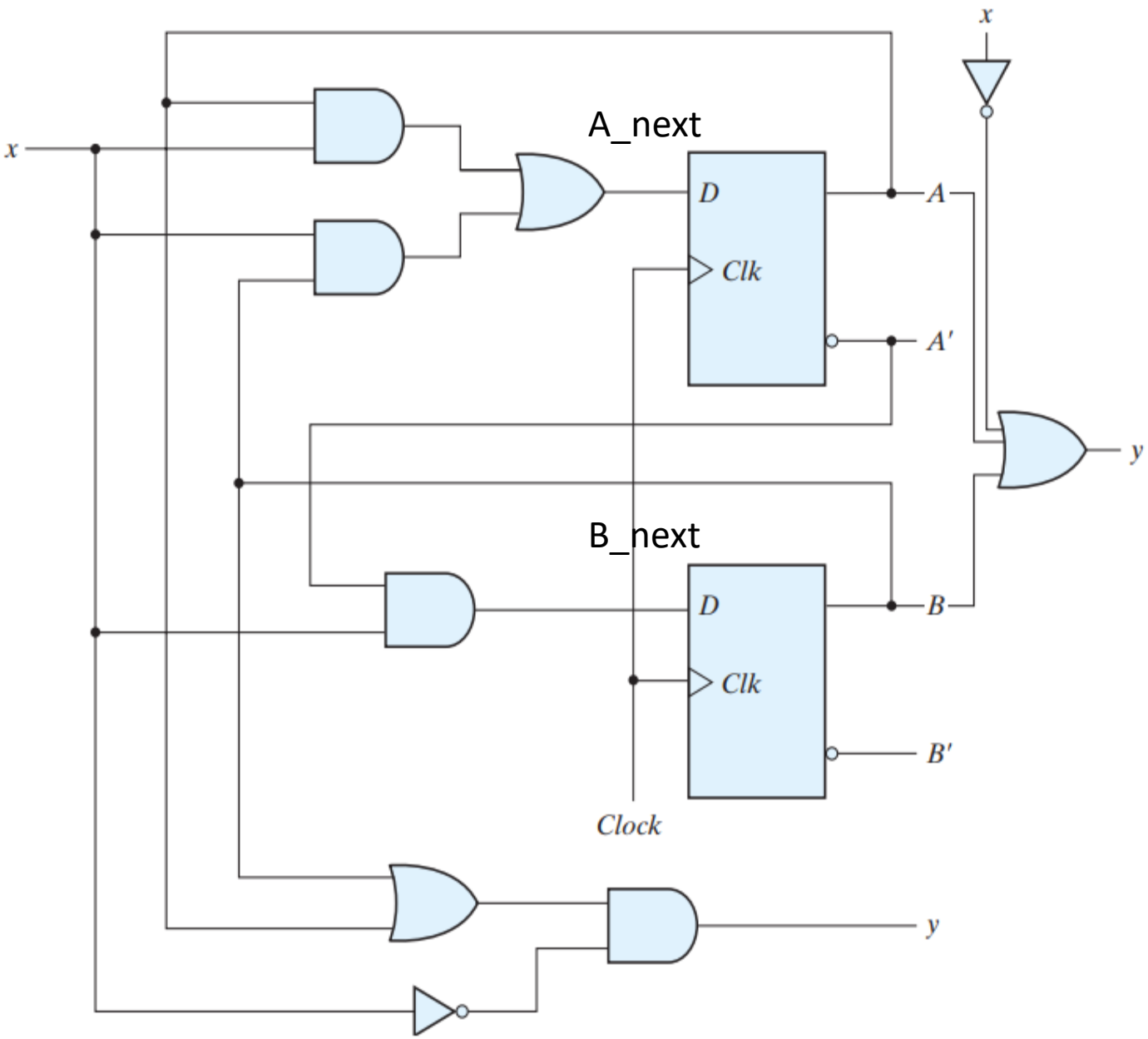


(a)



(b)

FSM DEVRE ANALİZİ

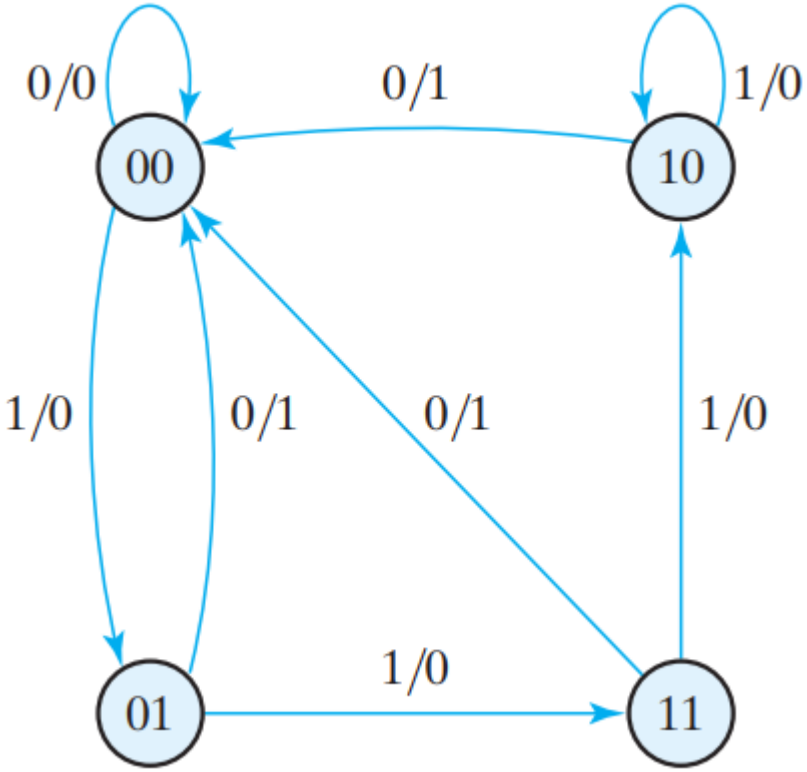


Giriş Sinyalleri : x
Durum FF'ları : A, B
Sonraki Durum : A_next, B_next
Çıkış Sinyalleri : y

A_next = Ax+Bx = x(A+B)
B_next = A'x
y = ~~A+B+x'~~ , (A+B)x' ???????!!!!

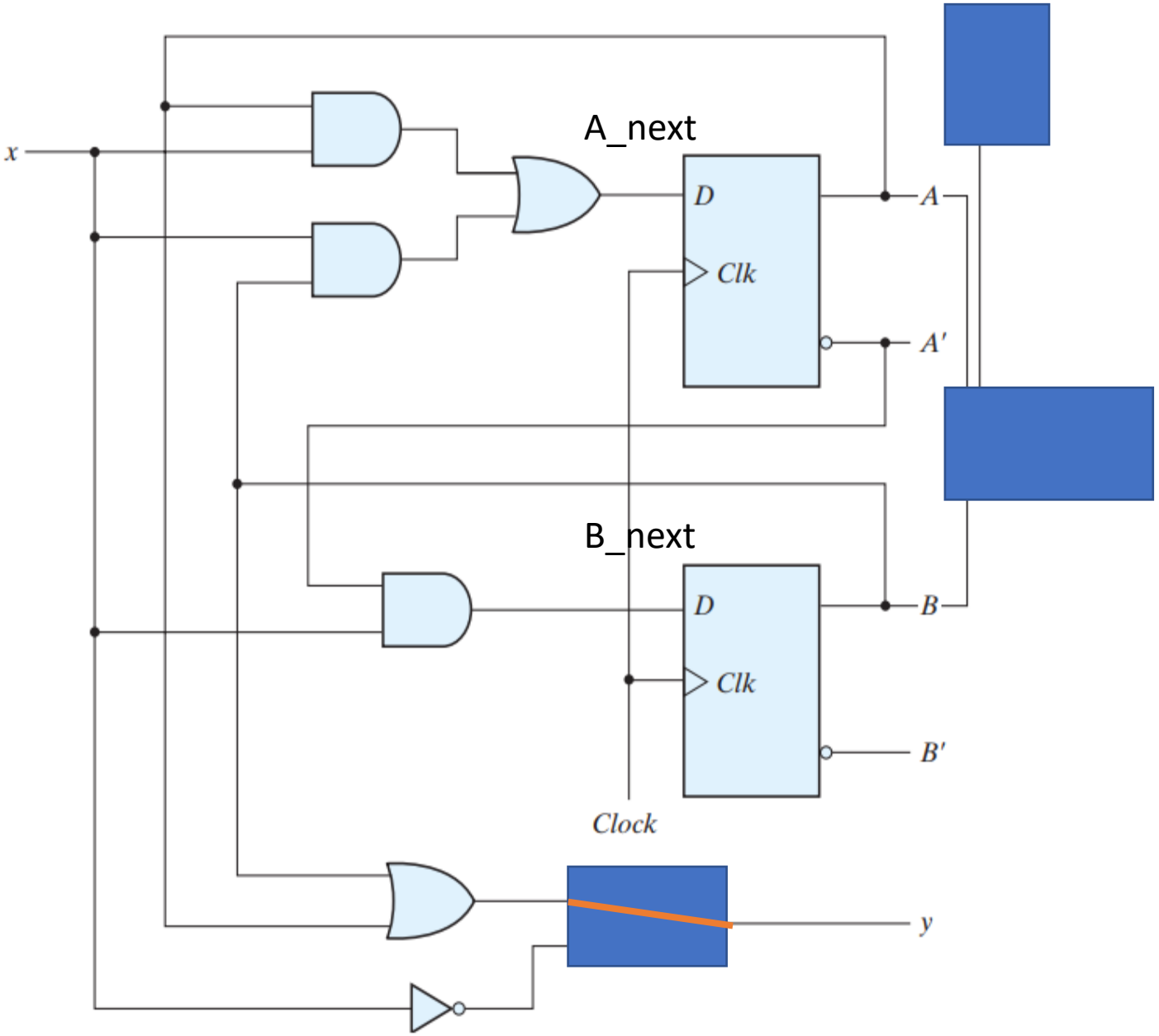
Giriş	Durum(t)		Çıkış	Durum(t+1)	
x	A	B	y	A_next	B_next
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

FSM DEVRE ANALİZİ



Present State		Next State				Output	
		$x = 0$		$x = 1$		$x = 0$	$x = 1$
A	B	A	B	A	B	y	y
0	0	0	0	0	1	0	0
0	1	0	0	1	1	1	0
1	0	0	0	1	0	1	0
1	1	0	0	1	0	1	0

FSM DEVRE ANALİZİ



A_next = Ax+Bx = x(A+B)

B_next = A'x

y = A+B → Sadece duruma bağlı (Moore)

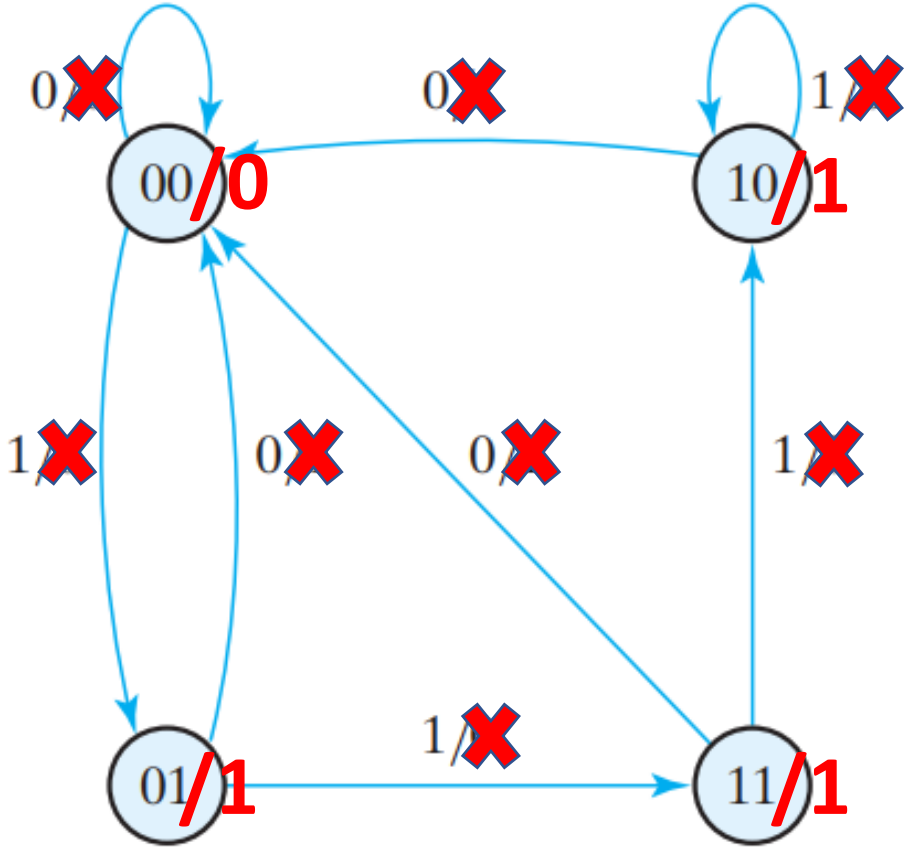
Giriş	Durum(t)		Çıkış	Durum(t+1)	
x	A	B	y	A_next	B_next
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	1	0	0
0	1	1	1	0	0
1	0	0	0	0	1
1	0	1	1	1	1
1	1	0	1	1	0
1	1	1	1	1	0

FSM DEVRE ANALİZİ

TOBB ETÜ
Ekonomi ve Teknoloji Üniversitesi

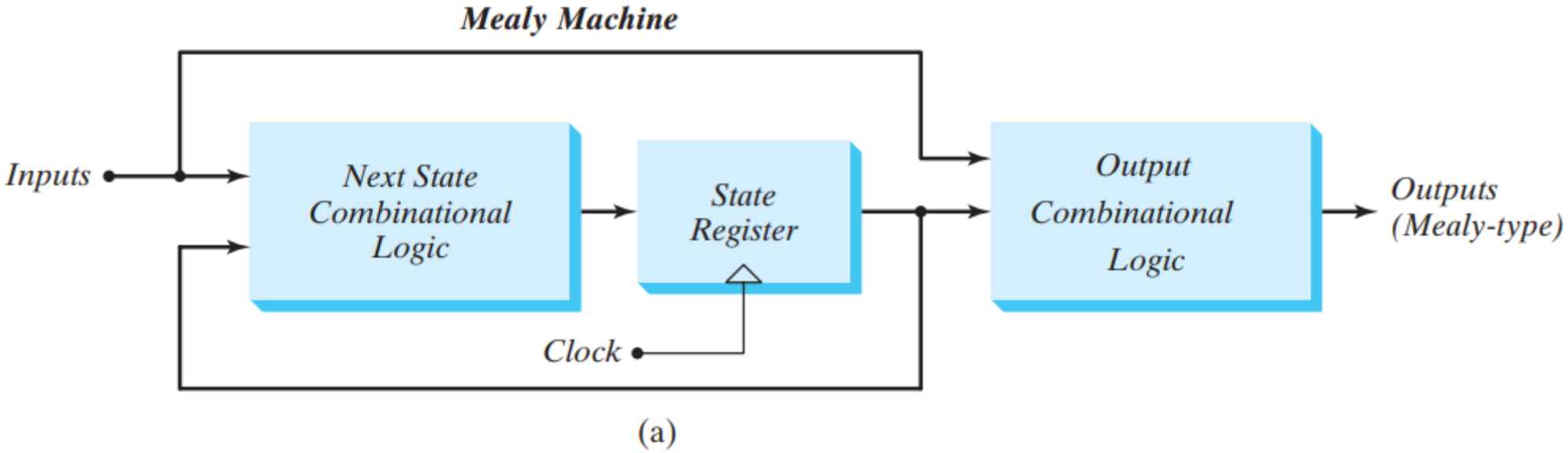
A_next = Ax+Bx = x(A+B)
B_next = A'x
y = A+B → Sadece duruma bağlı (Moore)

Giriş	Durum(t)		Çıkış	Durum(t+1)	
x	A	B	y	A_next	B_next
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	1	0	0
0	1	1	1	0	0
1	0	0	0	0	1
1	0	1	1	1	1
1	1	0	1	1	0
1	1	1	1	1	0

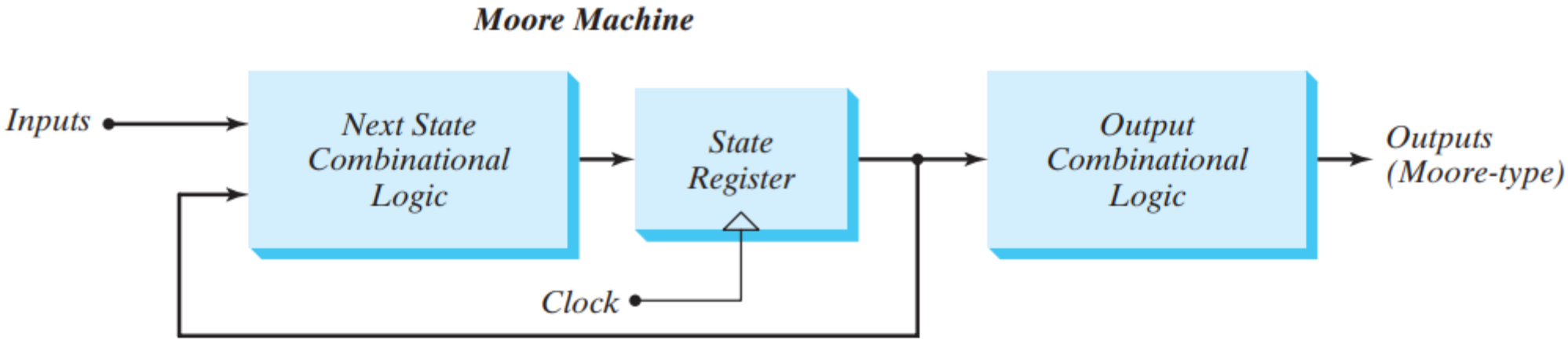


FSM → MEALY vs MOORE

TOBB ETÜ
Ekonomi ve Teknoloji Üniversitesi



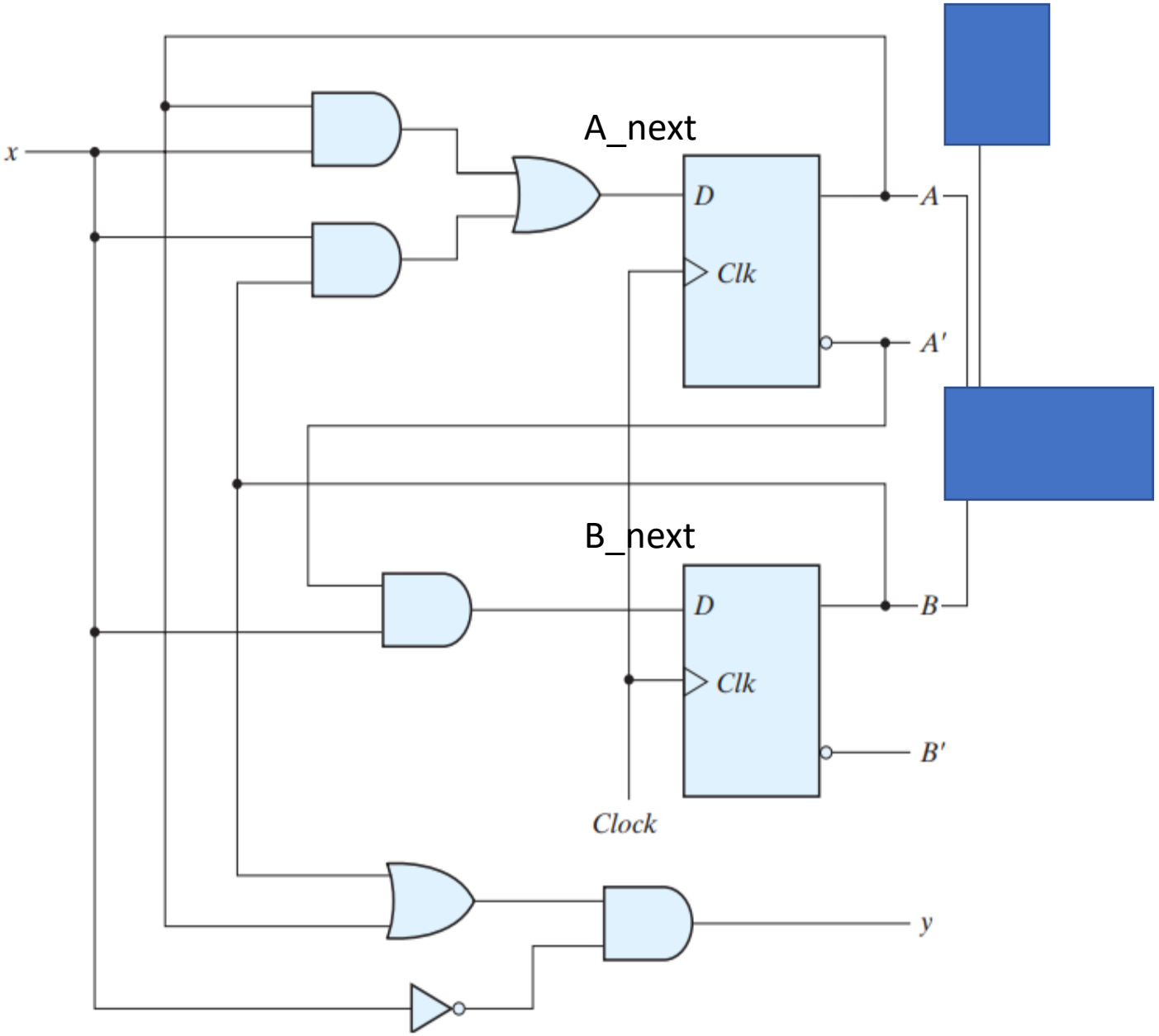
(a)



(b)

TOBB ETÜ
Ekonomi ve Teknoloji Üniversitesi

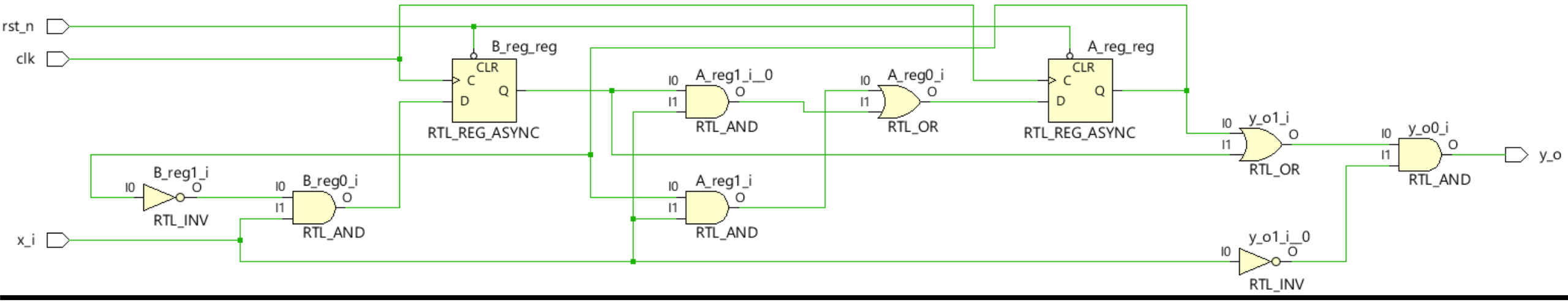
FSM → MEALY VERILOG

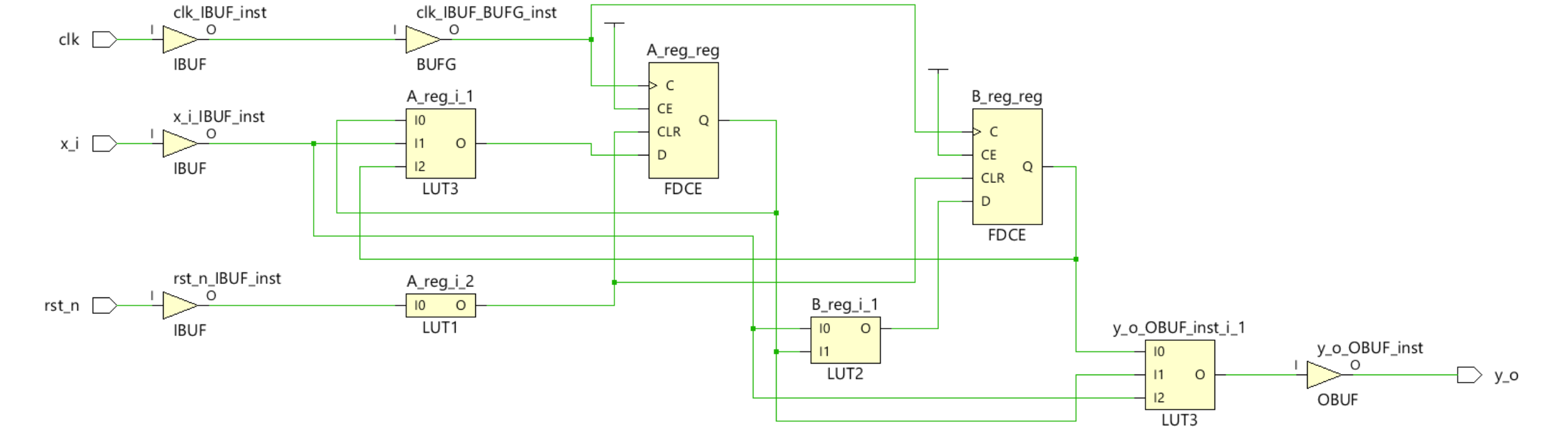



```
module fsm_mealy_3always
(
input x_i,
input rst_n,
input clk,
output y_o
);

reg y;
reg A_reg, B_reg, A_next, B_next;

// always block #1 for assigning D to Q
always @(posedge clk, negedge rst_n) begin
    if (rst_n == 1'b0) begin
        A_reg <= 1'b0;
        B_reg <= 1'b0;
    end
    else begin
        A_reg <= A_next;
        B_reg <= B_next;
    end
end

// always block #2 for evaluating next state
always @(*) begin
    A_next = (A_reg & x_i) | (B_reg & x_i);
    B_next = (!A_reg & x_i);
end

// always block #3 for evaluating output
always @(*) begin
    y = (A_reg | B_reg) & !x_i;
end

assign y_o = y;

endmodule
```

FSM → MEALY VERILOG

TOBB ETÜ
Ekonomi ve Teknoloji Üniversitesi





3 always vs 1 always

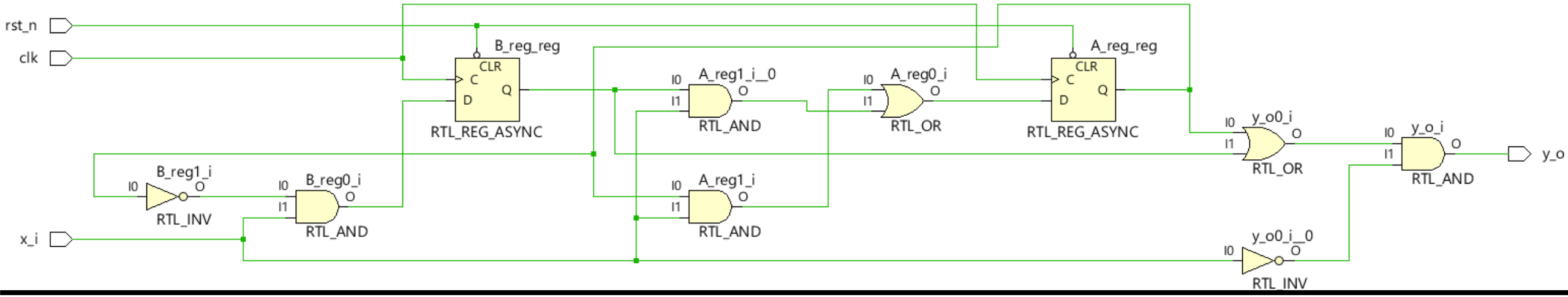
TOBB ETÜ
Ekonomi ve Teknoloji Üniversitesi

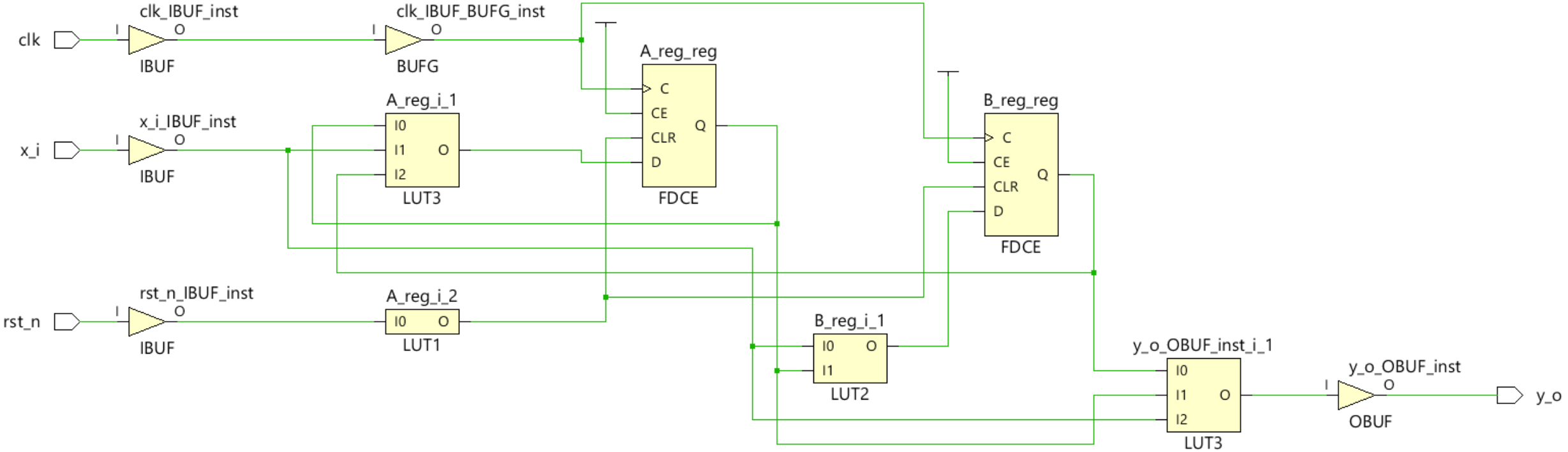
```
module fsm_mealy_3always
(
input x_i,
input rst_n,
input clk,
output y_o
);

reg y;
reg A_reg, B_reg, A_next, B_next;

// always block #1 for assigning D to Q
always @(posedge clk, negedge rst_n) begin
    if (rst_n == 1'b0) begin
        A_reg <= 1'b0;
        B_reg <= 1'b0;
    end
    else begin
        A_reg <= A_next;
        B_reg <= B_next;
    end
end

// always block #2 for evaluating next state
always @(*) begin
    A_next = (A_reg & x_i) | (B_reg & x_i);
    B_next = (!A_reg & x_i);
end

// always block #3 for evaluating output
always @(*) begin
    y = (A_reg | B_reg) & !x_i;
end

assign y_o = y;

endmodule
```

```
module fsm_mealy_1always
(
input x_i,
input rst_n,
input clk,
output y_o
);

reg A_reg, B_reg;

always @(posedge clk, negedge rst_n) begin
    if (rst_n == 1'b0) begin
        A_reg <= 1'b0;
        B_reg <= 1'b0;
    end
    else begin
        A_reg <= (A_reg & x_i) | (B_reg & x_i);
        B_reg <= (!A_reg & x_i);
    end
end

assign y_o = (A_reg | B_reg) & !x_i;


endmodule
```

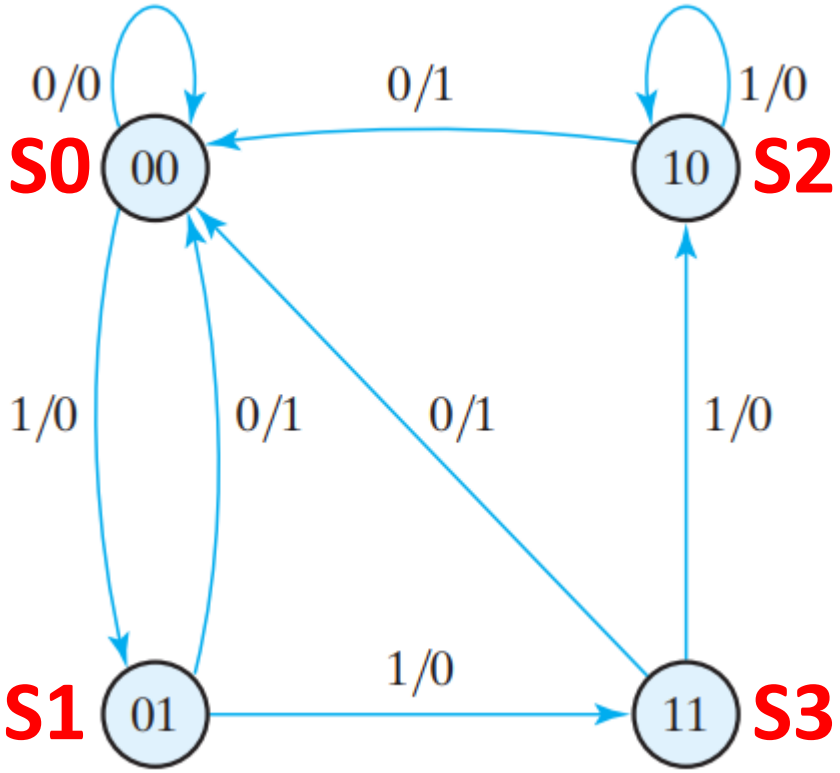
FSM → MEALY VERILOG

TOBB ETÜ
Ekonomi ve Teknoloji Üniversitesi





PARAMETRİK DURUM MAKİNASI - VERILOG



	Present State		Next State				Output	
			$x = 0$		$x = 1$		$x = 0$	$x = 1$
	A	B	A	B	A	B	y	y
S0	0	0	0	0	0	1	0	0
S1	0	1	0	0	1	1	1	0
S2	1	0	0	0	1	0	1	0
S3	1	1	0	0	1	0	1	0

PARAMETRİK DURUM MAKİNASI - VERILOG

TOBB ETÜ
Ekonomi ve Teknoloji Üniversitesi

```
module fsm_mealy_param
(
input x_i,
input rst_n,
input clk,
output y_o
);

localparam S0 = 2'b00;
localparam S1 = 2'b01;
localparam S2 = 2'b10;
localparam S3 = 2'b11;

reg y;
reg [1:0] state;
```

```
always @(posedge clk, negedge rst_n) begin
    if (rst_n == 1'b0) begin
        state <= S0;
    end
    else begin
        case (state)
            S0 : begin
                if (x_i == 1'b0) begin
                    state <= S0;
                end
                else begin
                    state <= S1;
                end
            end
            S1 : begin
                if (x_i == 1'b0) begin
                    state <= S0;
                end
                else begin
                    state <= S3;
                end
            end
            S2 : begin
                if (x_i == 1'b0) begin
                    state <= S0;
                end
                else begin
                    state <= S2;
                end
            end
```

```
            S3 : begin
                if (x_i == 1'b0) begin
                    state <= S0;
                end
                else begin
                    state <= S2;
                end
            end
            default : begin
                state <= S0;
            end
        endcase
    end
end
```

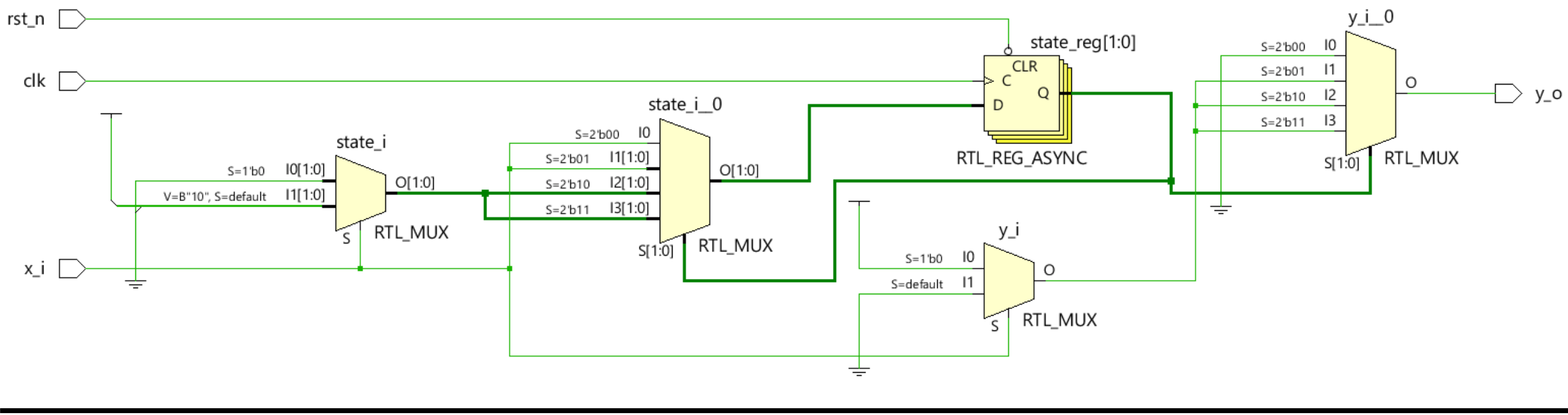
PARAMETRİK DURUM MAKİNASI - VERILOG

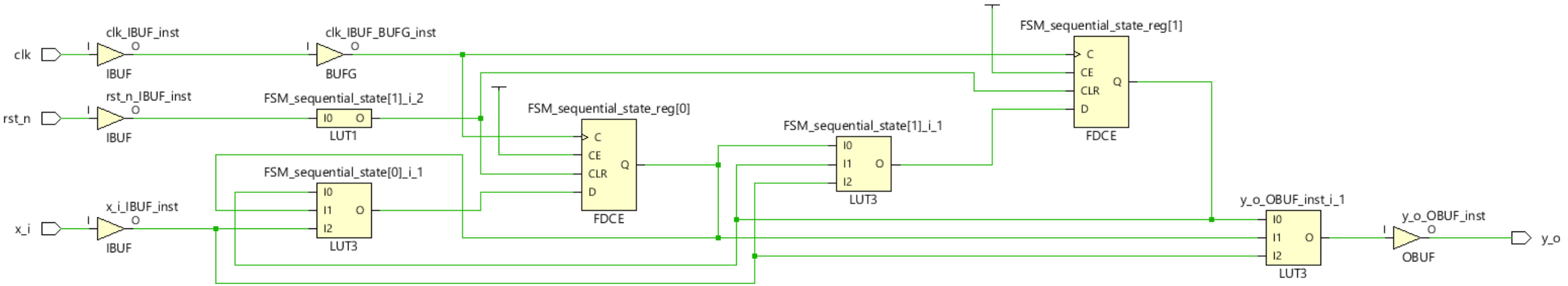
TOBB ETÜ
Ekonomi ve Teknoloji Üniversitesi

```
always @(*) begin
    case (state)
     S0 : begin
          if (x_i == 1'b0) begin
               y = 1'b0;
          end
          else begin
               y = 1'b0;
          end
     end
     S1 : begin
          if (x_i == 1'b0) begin
               y = 1'b1;
          end
          else begin
               y = 1'b0;
          end
     end
     S2 : begin
          if (x_i == 1'b0) begin
               y = 1'b1;
          end
          else begin
               y = 1'b0;
          end
     end
     S3 : begin
          if (x_i == 1'b0) begin
               y = 1'b1;
          end
          else begin
               y = 1'b0;
          end
     end
     default : begin

     end
    endcase
end

assign y_o = y;

endmodule
```

PARAMETRİK DURUM MAKİNASI - VERILOG

TOBB ETÜ
Ekonomi ve Teknoloji Üniversitesi



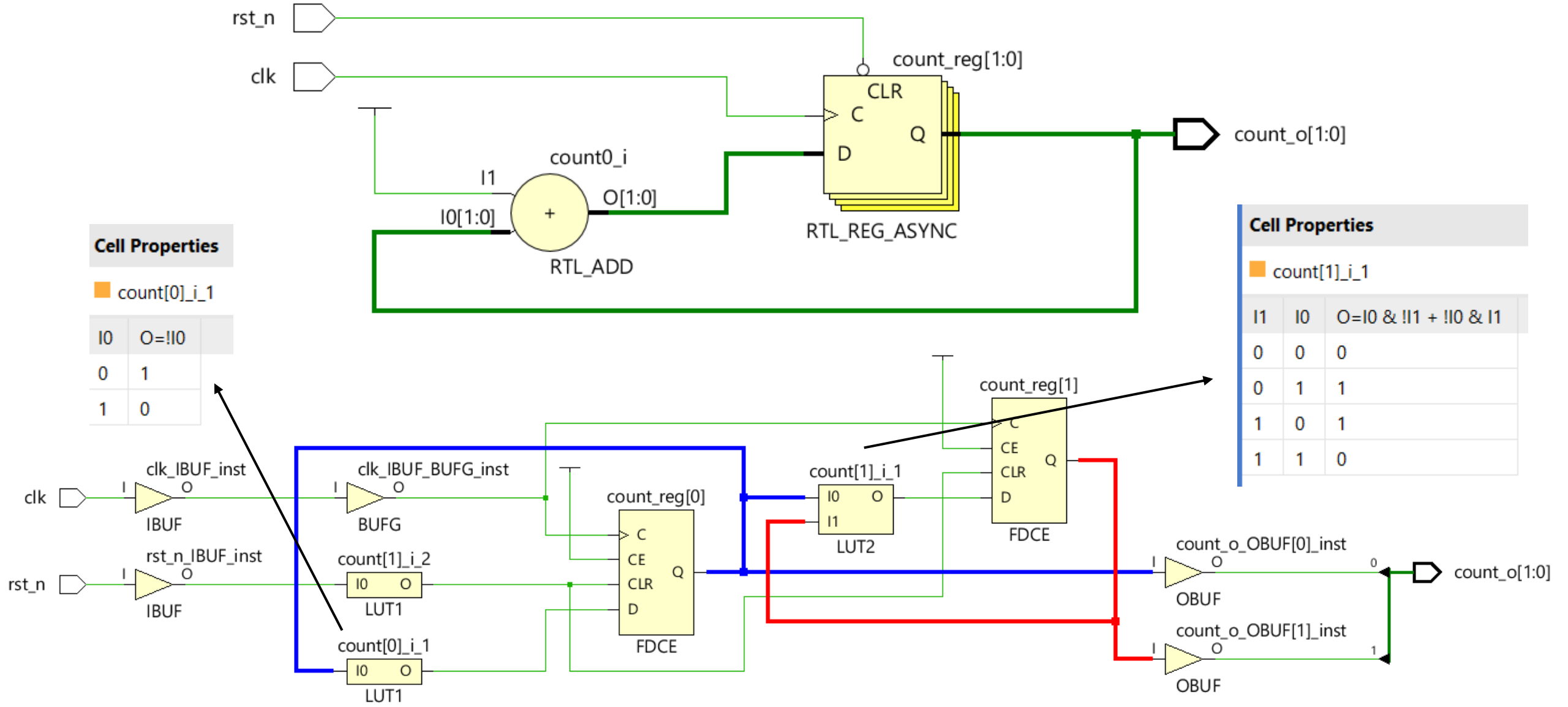


SAYAÇLAR (COUNTERS)

TOBB ETÜ
Ekonomi ve Teknoloji Üniversitesi

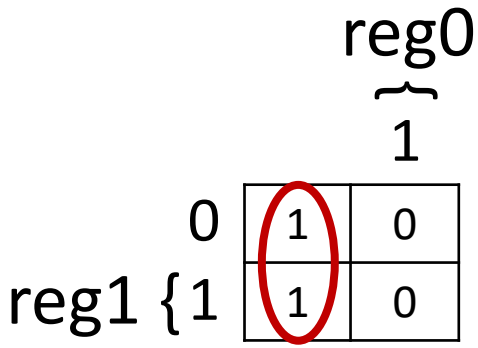
```
module counter_2bit
(
input clk,
input rst_n,
output [1:0] count_o
);

reg [1:0] count;

always @(posedge clk, negedge rst_n) begin
    if (rst_n == 1'b0) begin
        count <= 2'b00;
    end else begin
        count <= count + 1;
    end
end

assign count_o = count;

endmodule
```

SAYAÇLAR (COUNTERS)

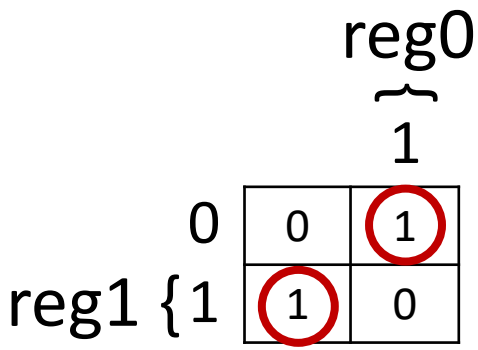


SAYAÇLAR (COUNTERS)

Durum(t)		Durum(t+1)	
reg1	reg0	reg1_next	reg0_next
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0



reg0_next = reg0’



reg1_next = reg0reg1’ + reg0’reg1

= reg0 ^ reg1

SAYAÇLAR (COUNTERS) – VERILOG PARAMETER

TOBB ETÜ
Ekonomi ve Teknoloji Üniversitesi

```
module reg_param
#(
parameter N=32
)
(
input [N-1:0] d_i,
input rst_n,
input clk,
output [N-1:0] q_o
);

reg [N-1:0] q;

always @(posedge clk or negedge rst_n) begin
    if (rst_n == 1'b0) begin
        q <= {N{1'b0}};
    end
    else begin
        q <= d_i;
    end
end

assign q_o = q;

endmodule
```

```
module counter_param
#(
parameter N = 8
)
(
input clk,
input rst_n,
output [N-1:0] count_o
);

reg [N-1:0] count;

always @(posedge clk, negedge rst_n) begin
    if (rst_n == 1'b0) begin
        count <= {N{1'b0}};
    end else begin
        count <= count + 1;
    end
end

assign count_o = count;

endmodule
```