GEBZE TEKNİK ÜNİVERSİTESİ

ELEKTRONİK MÜHENDİSLİĞİ

ELM - 234
ÖDEV 3

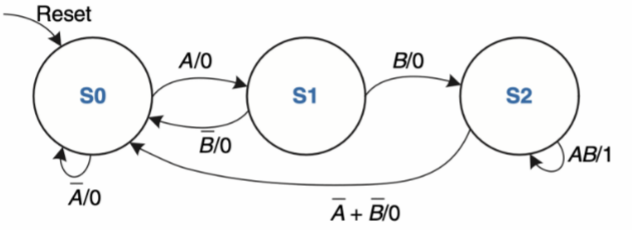
Finite State Machines

Hazırlayan: Alican Bayındır -200102002087

Problem 1. Mealy FSM tasarımı `[10 + 10 + 10 + 4 = 34 puan]`

Şekil 1 de state transition diagramı verilen devreyi

A. Binary state encoding kullanarak tasarlayın.
B. One-hot state encoding kullanarak tasarlayın.
C. 3-bit Johnson encoding kullanarak tasarlayın.
D. Bu üç encodingi kullanılan lojik kapı ve flip-floplara göre karşılaştırın.



Şekil 1

$S_0 = 00$
$S_1 = 01$
$S_2 = 10$

1.a) Binary state encoding için truth table

Current State Flags	States		Girdiler		Next states		Çıktı	
	f_1	f_0	A	B	f_1	f_0	Out	Next State Flags
S_0	0	0	0	0	0	0	0	S_0
	0	0	0	1	0	0	0	S_0
	0	0	1	1	0	1	0	S_1
	0	0	1	0	0	1	0	S_1
S_1	0	1	0	0	0	0	0	S_0
	0	1	0	1	1	0	0	S_2
	0	1	1	1	1	0	0	S_2
	0	1	1	0	0	0	0	S_0
S_2	1	0	0	0	0	0	0	S_0
	1	0	0	1	0	0	0	S_0
	1	0	1	1	1	0	1	S_2
	1	0	1	0	0	0	0	S_0

Binary state encoding $f_{0_{next}}$ için K-Map

fifo / AB	00	01	11	10
00	0	0	X	0
01	0	0	X	1
11	0	1	X	1
10	0	0	X	0

Boolean denklemi

$f_1\bar{B} + f_0AB$

Binary state encoding $f_{1_{next}}$ için K-Map

fifo / AB	00	01	11	10
00	0	0	X	0
01	0	0	X	0
11	1	0	X	0
10	1	0	X	0

Boolean denklemi

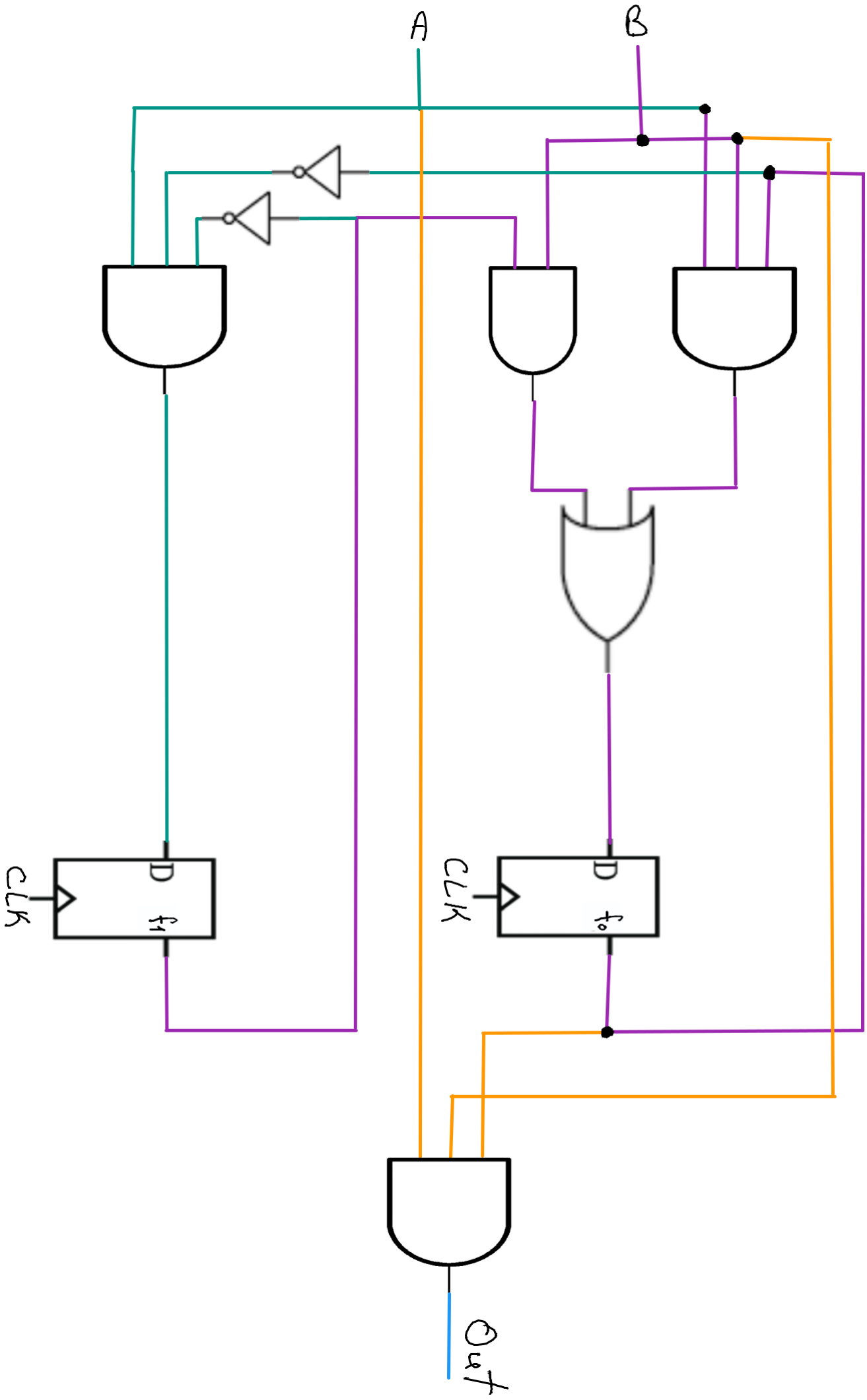
$\bar{f}_0\bar{f}_1A$

Binary state encoding Output için K-Map

fifo / AB	00	01	11	10
00	0	0	X	0
01	0	0	X	0
11	0	1	X	0
10	0	0	X	0

Boolean denklemi

f_0AB


A
B
CLK
D
f1
CLK
D
f0
Out

1.b) One-Hot encoding için truth table

($S_0 = 001$, $S_1 = 010$, $S_2 = 100$)

Current State flags	Current States			Girdiler		Next States			Output	Next State Flags
	f_0	f_1	f_2	A	B	f_0	f_1	f_2	Out	
S_0	0	0	1	0	0	0	0	1	0	S_0
	0	0	1	0	1	0	0	1	0	S_0
	0	0	1	1	1	0	1	0	0	S_1
	0	0	1	1	0	0	1	0	0	S_1
S_1	0	1	0	0	0	0	0	1	0	S_0
	0	1	0	0	1	1	0	0	0	S_2
	0	1	0	1	1	1	0	0	0	S_2
	0	1	0	1	0	0	0	1	0	S_0
S_2	1	0	0	0	0	0	0	1	0	S_0
	1	0	0	0	1	0	0	1	0	S_0
	1	0	0	1	1	1	0	0	1	S_2
	1	0	0	1	0	0	0	1	0	S_0

One-Hot encoding $f_{2_{next}}$ için K-Map

AB \ $f_2f_1f_0$	001	010	100
00	0	0	0
01	0	1	0
11	0	1	1
10	0	0	0

Boolean denklemi:

$$\bar{B}\bar{f}_2 f_1 \bar{f}_0 + AB\bar{f}_2$$

One-Hot encoding $f_{1_{next}}$ için K-Map

AB \ $f_2f_1f_0$	001	010	100
00	1	1	1
01	1	0	1
11	0	0	0
10	0	1	1

Boolean denklemi:

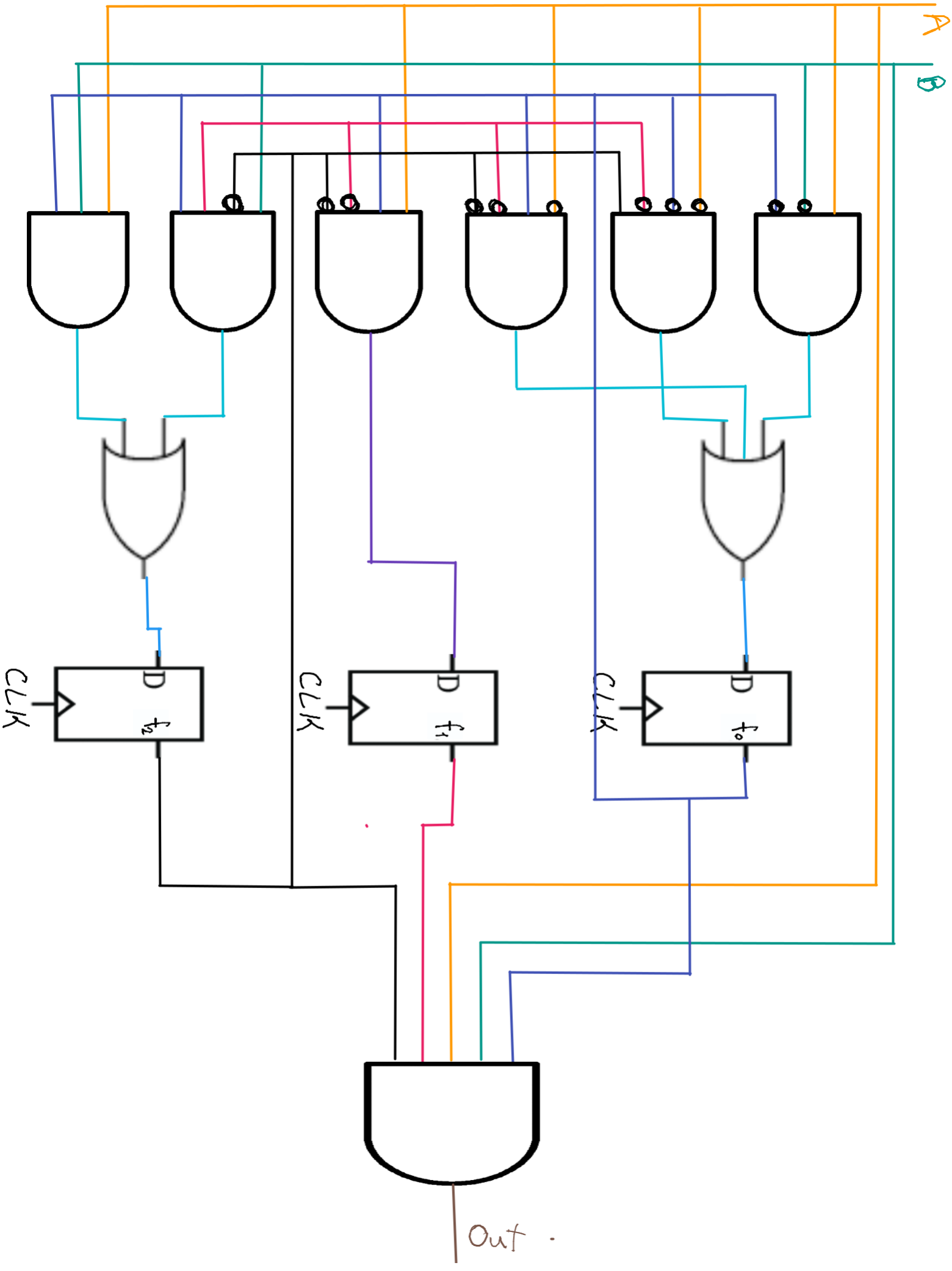
$$\bar{f}_2\bar{f}_1 f_0 \bar{A} + f_2\bar{f}_1\bar{f}_0\bar{A} + \bar{f}_0 A\bar{B}$$

One-Hot encoding $f_{0_{next}}$ için K-Map

AB \ $f_2f_1f_0$	001	010	100
00	0	0	0
01	0	0	0
11	0	0	1
10	0	0	0

Boolean denklemi:

$$f_2\bar{f}_1\bar{f}_0 AB$$


A
B
CLK
CLK
CLK
Out

1.c) Johnson encoding için truth table

$S_0 = 00$
$S_1 = 01$
$S_2 = 11$

Current State Flags	States		Girdiler		Next states		Çıktı	
	f_1	f_0	A	B	f_1	f_0	Out	Next State Flags
S_0	0	0	0	0	0	0	0	S_0
	0	0	0	1	0	0	0	S_0
	0	0	1	1	0	1	0	S_1
	0	0	1	0	0	1	0	S_1
S_1	0	1	0	0	0	0	0	S_0
	0	1	0	1	1	1	0	S_2
	0	1	1	1	1	1	0	S_2
	0	1	1	0	0	0	0	S_0
S_2	1	1	0	0	0	0	0	S_0
	1	1	0	1	0	0	0	S_0
	1	1	1	1	1	1	1	S_2
	1	1	1	0	0	0	0	S_0

Johnson encoding fo_{next} için K-Map

AB \ fifo	00	01	11	10
00	0	0	0	X
01	0	1	0	X
11	1	1	1	X
10	1	0	0	X

Boolean denklemi

$\overline{f_1}\,\overline{f_0}A + \overline{f_1}f_0B + AB$

Johnson encoding $f_{1_{next}}$ için K-Map

AB \ fifo	00	01	11	10
00	0	0	0	X
01	0	1	0	X
11	0	0	1	X
10	0	1	0	X

Boolean denklemi

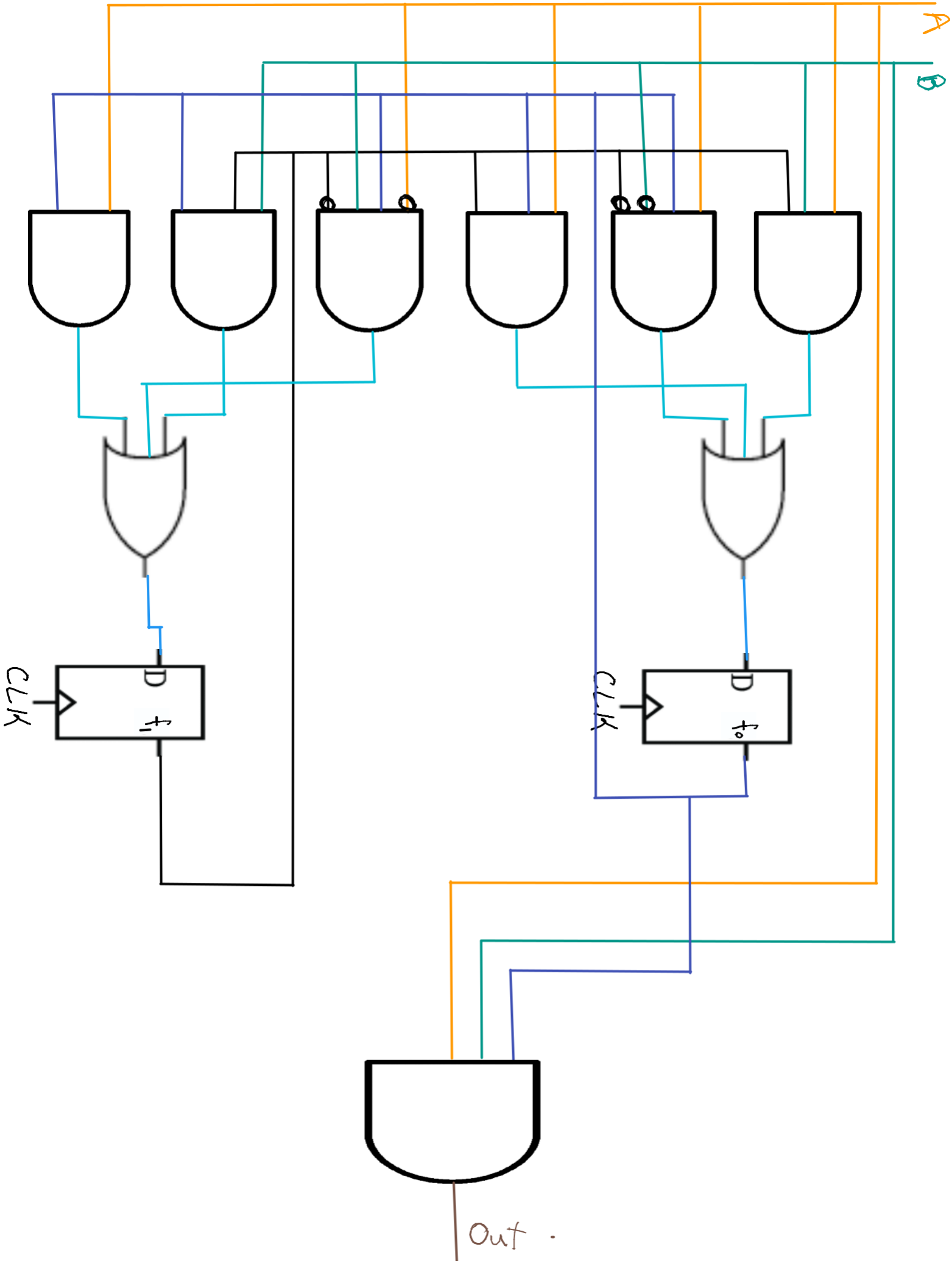
$\overline{f_0}f_1\bar{A}B + f_0AB + \overline{f_0}f_1A\bar{B}$

Johnson encoding Output için K-Map

AB \ fifo	00	01	11	10
00	0	0	0	X
01	0	0	0	X
11	0	0	1	X
10	0	0	0	X

Boolean denklemi

f_1AB


A
B
CLK
f_1
CLK
f_0
Out

1.D) Binary encoding en basit, sıradan encode etme yöntemidir. Sıralı bir şekilde 0'dan başlayarak sayı arttırılır. One-Hot encoding'de ise sadece 1 bit 'hot' yani 1 olacak. Johnson'da ise en MSB'deki bit invert edilerek LSB'ye yazılır. Binary'de 0'dan başlandığı için 2 flip flop ile devre kuruldu. One-Hot'ta her bir durumu ayırt etmek için 1 tane hot bit gerekli 3 state olduğu için 3 flip flop ile ifade edilebildi. Johnson's Encoding'te 3 bit ile yapıldığında 1 flip flop boşta kalıyor hep 0 göstermesi gerekiyordu. Bundan dolayı Johnson's Encoding 2 flip flop ile çizilmiştir. Her bir devre için critical pathler sırasıyla

Johnson > one-hot > binary

olacaktır.

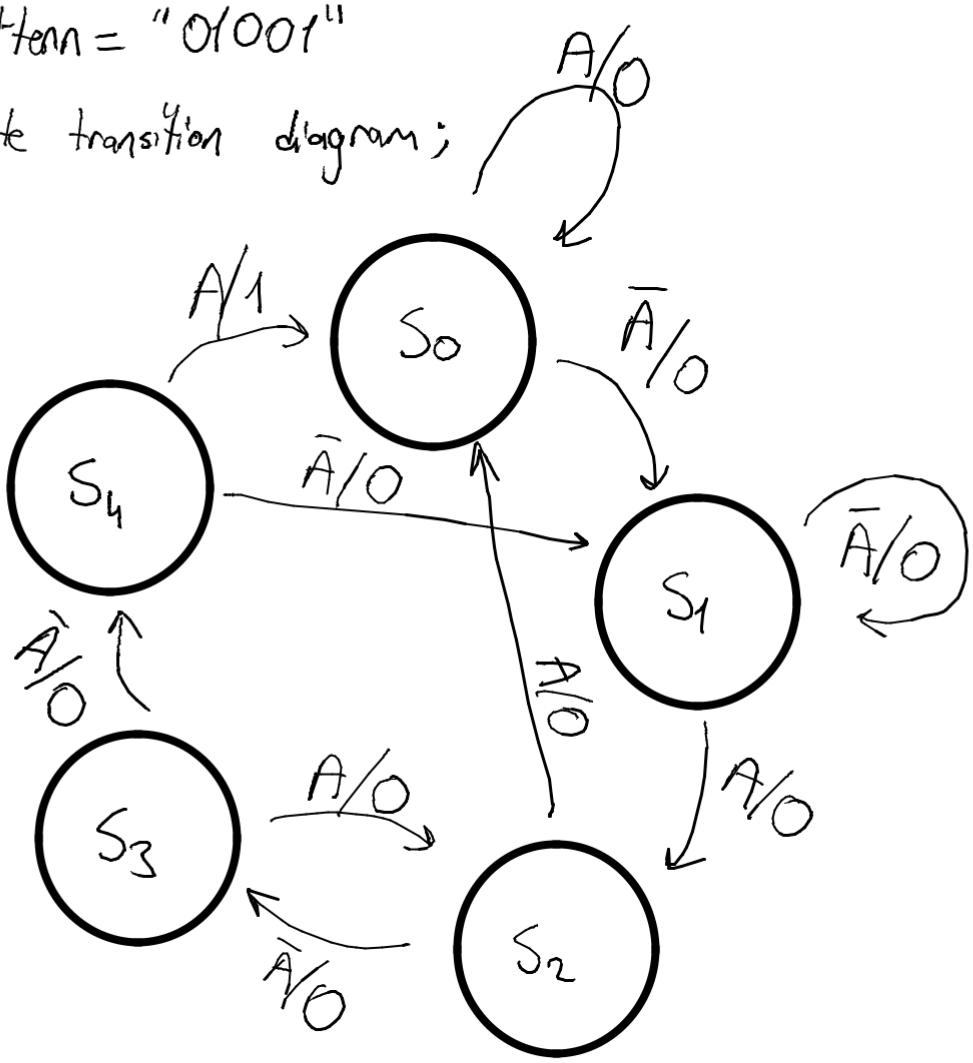
Problem 2. Pattern Detector [5 + 12 + 12 + 15 = 44 puan]

Sıralı olarak gelen bit dizisi içerisinde 01001 patterni geldiğinde unlock sinyalini yakan bir devre tasarlayın.

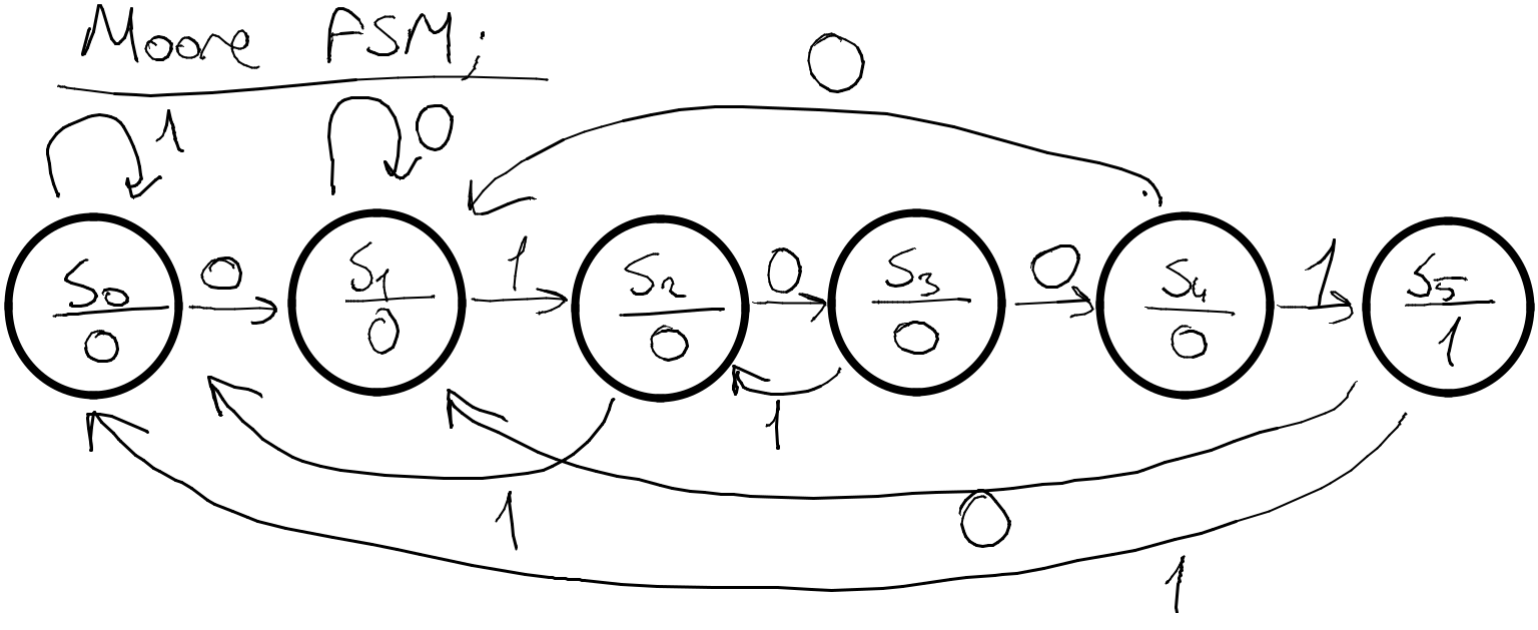
A. State Transition Diagramını çiziniz.
B. Moore FSM kullanarak tasarlayın.
C. Mealy FSM kullanarak tasarlayın.
D. İki FSM için de ayrı HDL kullanarak tasarlayın, bir testbench devresi oluşturun ve Modelsim de simüle edin. Uzun bir bit stream vererek iki devrenin de çıkışı gösterin.

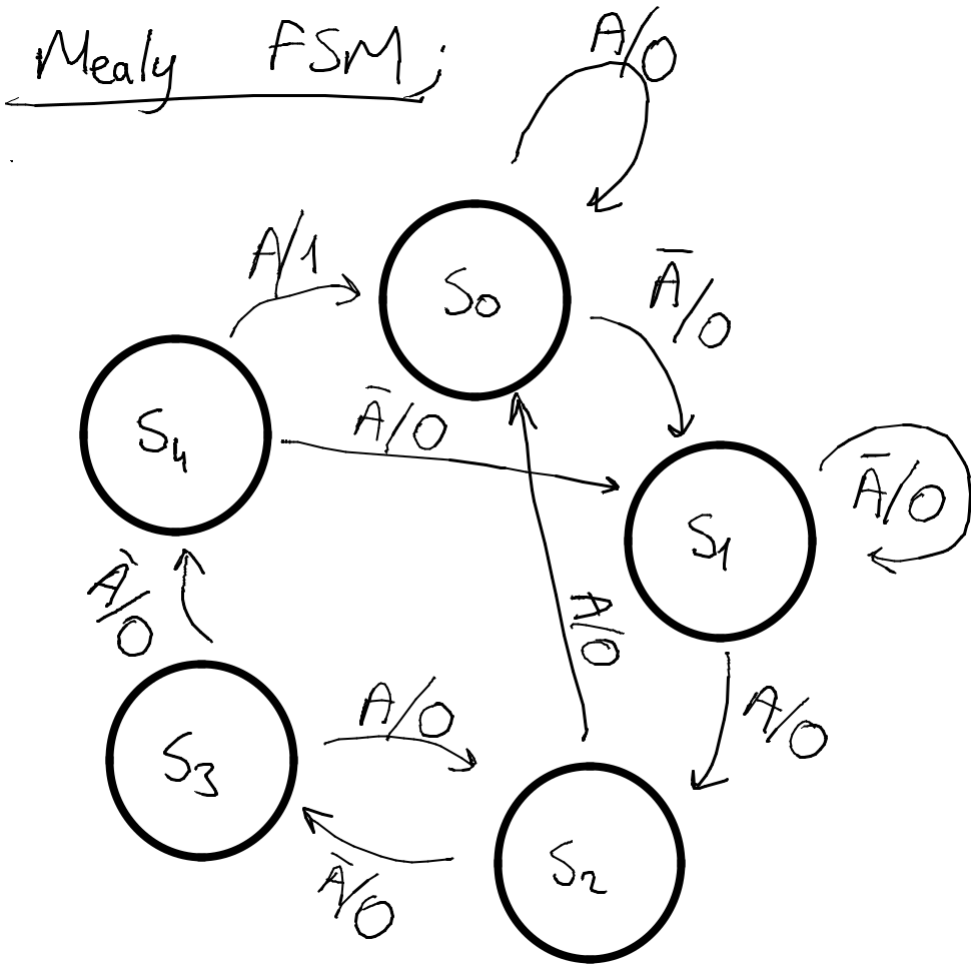
Pattern = "01001"

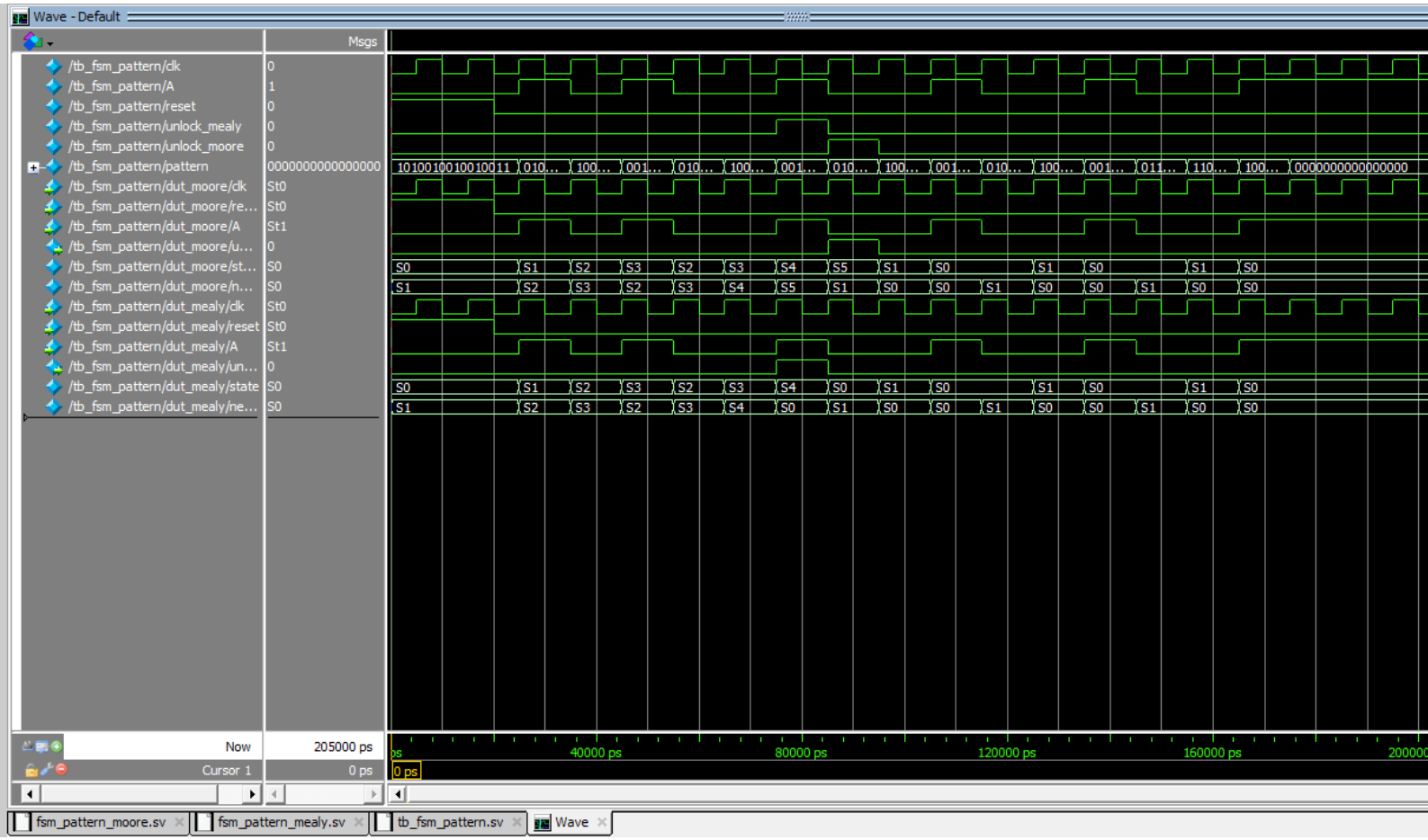
State transition diagram;



Moore FSM;



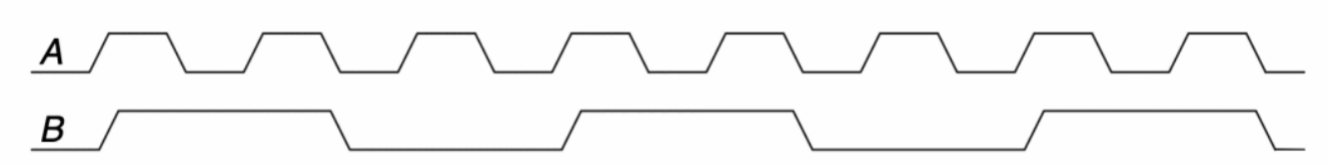




01001 Pattern çözen devrenin simülasyon sonucu

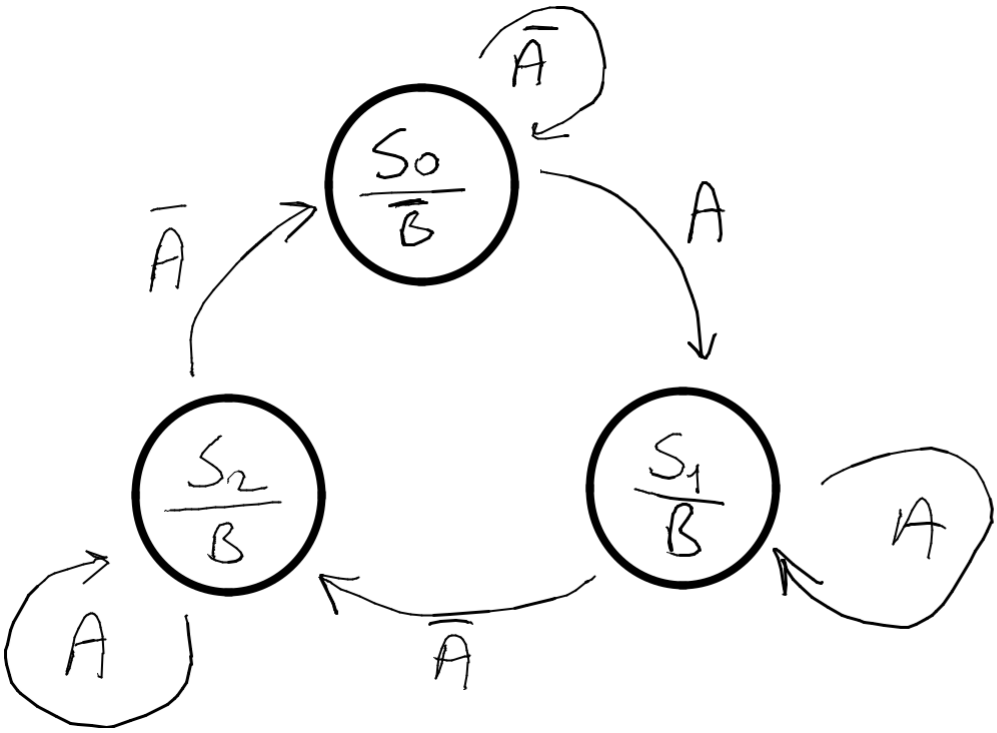
Problem 3. Devre tasarımı [10 + 12 = 22 puan]

A. Şekil 2 de verilen A girişi ile B sinyalini üreten bir FSM devre tasarlayınız. (state transition diagram + mealy or moore FSM)
B. HDL kullanarak bu devreyi tasarlayın, bir testbench devresi oluşturun ve Modelsim de simüle edin.



Şekil 2

State transition + Moore FSM;



Bu sorunun kodlarını yazdım ancak çalıştıramadım bir sonraki EKLER kısmına bütün kodlar eklenmişti

EKLER

```
/* 01001 pattern'i ile unlock sinyali veren devre kodu soru 2 - D
şıkkı MEALY */

module fsm_pattern_mealy (
    input  logic clk, reset, A,
    output logic unlock
);

typedef enum {S0, S1, S2, S3, S4, S5} statetype;
statetype state, nextstate;

always_ff @(posedge clk)
    if (reset) state <= S0;
    else       state <= nextstate;


always_comb
    case (state)
        S0:
            if (A) nextstate = S0;
            else   nextstate = S1;
        S1:
            if (A) nextstate = S2;
            else   nextstate = S0;
        S2:
            if (A) nextstate = S0;
            else   nextstate = S3;
        S3:
            if (A) nextstate = S2;
            else   nextstate = S4;
        S4:
            if (A) nextstate = S0;
            else   nextstate = S1;

        default: nextstate = S0;
    endcase

always_comb
    if (state == S4 && A) unlock = 1'b1;
    else unlock = 1'b0;

endmodule
```

```
/* 01001 pattern'i ile unlock sinyali veren devre kodu soru 2 - D
şıkkı MOORE*/

module fsm_pattern_moore (
    input  logic clk, reset, A,
    output logic unlock
);

typedef enum {S0, S1, S2, S3, S4, S5} statetype;
statetype state, nextstate;

always_ff @(posedge clk)
    if (reset) state <= S0;
    else       state <= nextstate;

always_comb
    case (state)
        S0:
            if (A) nextstate = S0;
            else   nextstate = S1;
        S1:
            if (A) nextstate = S2;
            else   nextstate = S0;
        S2:
            if (A) nextstate = S0;
            else   nextstate = S3;
        S3:
            if (A) nextstate = S2;
            else   nextstate = S4;
        S4:
            if (A) nextstate = S5;
            else   nextstate = S1;
        S5:
            if (A) nextstate = S0;
            else   nextstate = S1;

        default: nextstate = S0;
    endcase

assign unlock = (state == S5);

endmodule
```

```
/* 2. soru D şıkkı için testbench */

`timescale 1ns/1ps

module tb_fsm_pattern ();

    logic clk, A;
    logic reset;
    logic unlock_mealy, unlock_moore;
    logic [15:0] pattern = 16'b1010010010010011; // 2 adet 001
pattern mevcut

    fsm_pattern_moore dut_moore(.clk(clk), .reset(reset), .A(A),
.unlock(unlock_moore));
    fsm_pattern_mealy dut_mealy(.clk(clk), .reset(reset), .A(A),
.unlock(unlock_mealy));

    // clk sinyali olustur
    always
    begin
        clk = 0; #5; clk = 1; #5;
    end

    // active-high reset
    initial
    begin
        A = 0; reset = 1; #20;
        reset = 0; #5; // clock rising edgele ayni anda gonder
datayi
        for (int i=0; i<16; i++) begin
            A = pattern[15];
            pattern = pattern << 1'b1;
            #10;
        end
        #20;
        $stop;
    end

endmodule
```

```
module counter (
    input  logic clk, reset, A,
    output logic unlock
);

typedef enum {S0, S1, S2} statetype;
statetype state, nextstate;

always_ff @(posedge clk) begin
    if (reset) begin
        state <= S0;
    end
    else begin
        state <= nextstate;
    end
end

always_comb begin
    case (state)
        S0:
            if (A) begin
                nextstate = S1;
                unlock = 1;
            end
            else   nextstate = S0;
        S1:
            if (A) nextstate = S1;
          else nextstate = S2;
        S2:
      if(A)
            nextstate = S2;
      else begin
            nextstate = S0;
            unlock = 0;
      end
    endcase
end
endmodule
```

```
/* tb_fsm_counter.sv */

`timescale 1ns/1ps

module tb_fsm_counter ();

    logic clk, A;
    logic reset;
    logic unlock;

    counter_test          dut0(.clk(clk), .reset(reset), .A(A),
.unlock(unlock));
    // clk sinyali olustur
    always
    begin
        clk = 0; #5; clk = 1; #5;
    end

    // active-high reset
    initial
    begin
        A = 0; reset = 1; #15;
        reset = 0; A = 1; #15;
        A = 0; #15;
        A = 1; #15;
        A = 0; #15;
        A = 1; #15;
        $stop;
    end
endmodule
```