Synchronous Sequential Logic
Part II

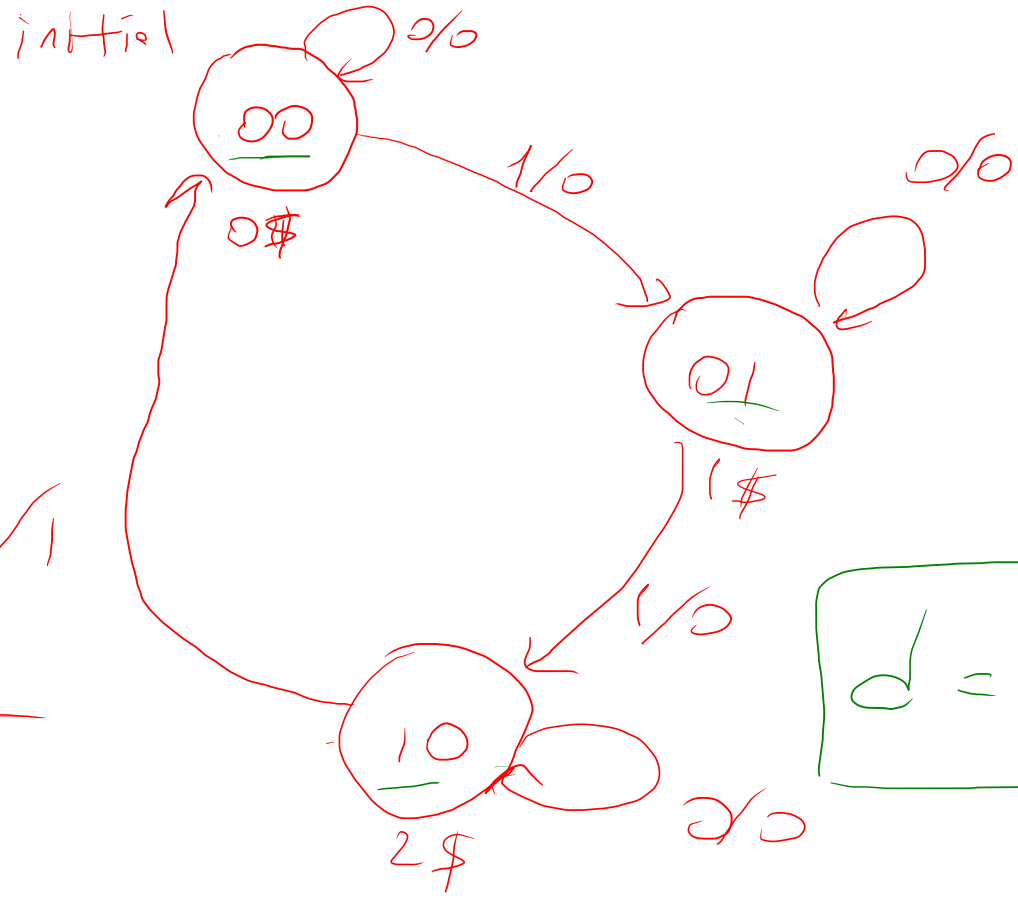
Logic and Digital System Design - CS 303

Sabancı University

Vending Machine

Current $q_1 q_0$	c	Next $q_1 q_0$	d
0 0	0	0 0	0
0 0	1	0 1	0
0 1	0	0 1	0
0 1	1	1 0	0
1 0	0	1 0	0
1 0	1	0 0	1
1 1	0	0 0	0
1 1	1	0 0	0

1/1

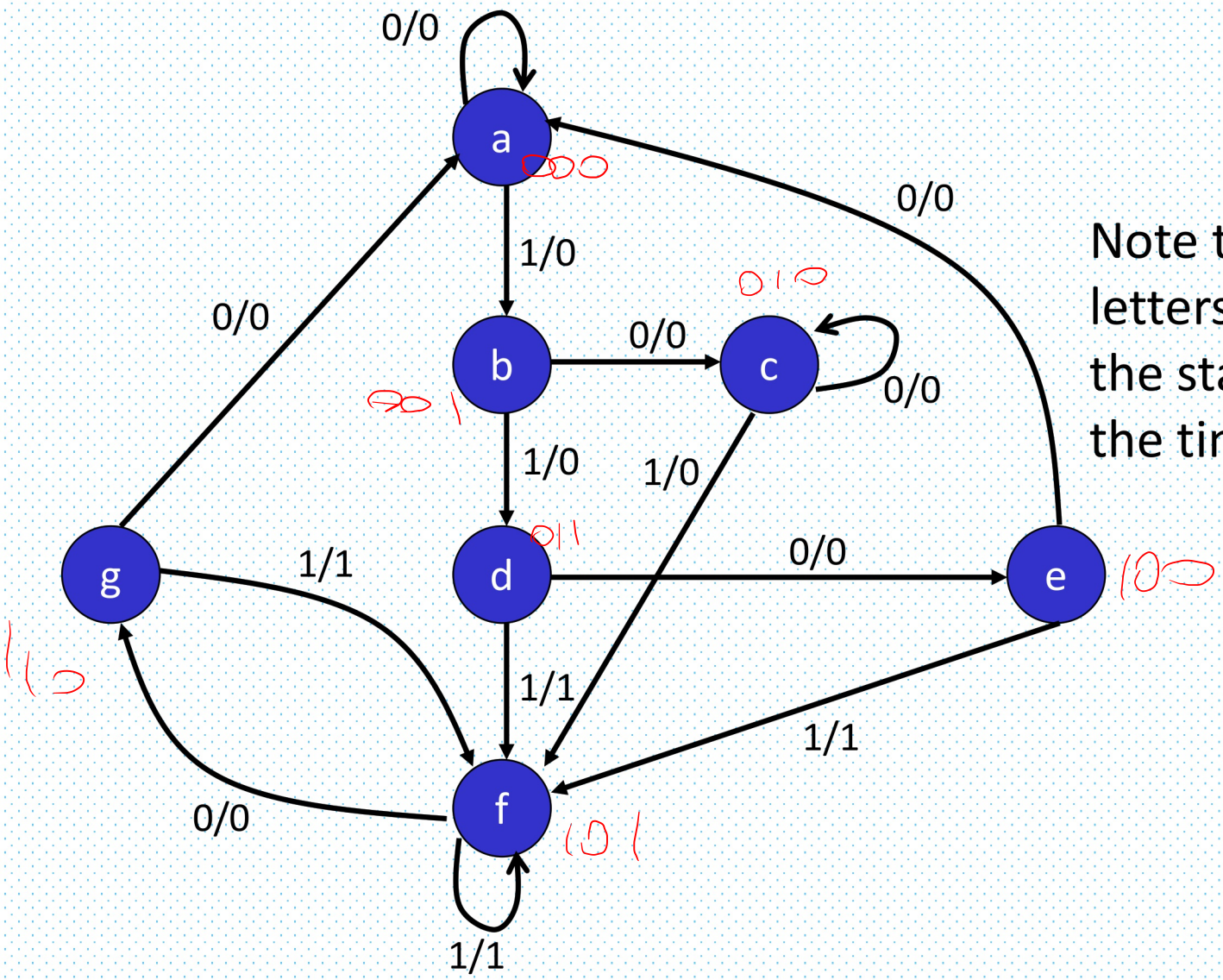


$$d = q_1 q_0 c$$

State Reduction and Assignment

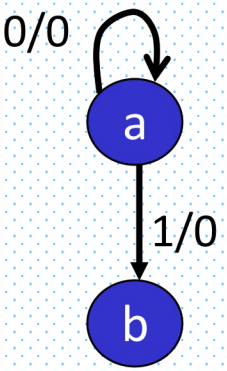
- In the design process of sequential circuits, certain techniques are useful in reducing the circuit complexity
  - state reduction
  - state assignment
- State reduction
  - Fewer states ➔ fewer number of flip-flops
  - m flip-flops ➔ 2^m states
  - Example: m = 5 ➔ $2^m = 32$
    - If we reduce the number of states to 21 do we reduce the number of flip-flops?

Example: State Reduction



Note that we use letters to designate the states for the time being

State Reduction Technique 1/7



- Step 1: get a state table

present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	c	d	0	0
c	c	f	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

State Reduction Technique 2/7

- Step 2: Inspect the state table for equivalent states
  - Equivalent states: Two states,
    1. that produce exactly the same output
    2. whose next states are identical
  - for each input combination

State Reduction Technique 3/7

present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	c	d	0	0
c	c	f	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

- States “e” and “g” are equivalent
- One of them can be removed

State Reduction Technique 4/7

present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	c	d	0	0
c	c	f	0	0
d	e	f	0	1
e	a	f	0	1
f	e	f	0	1

- We keep looking for equivalent states

State Reduction Technique 5/7

present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	c	d	0	0
c	c	d	0	0
d	e	d	0	1
e	a	d	0	1

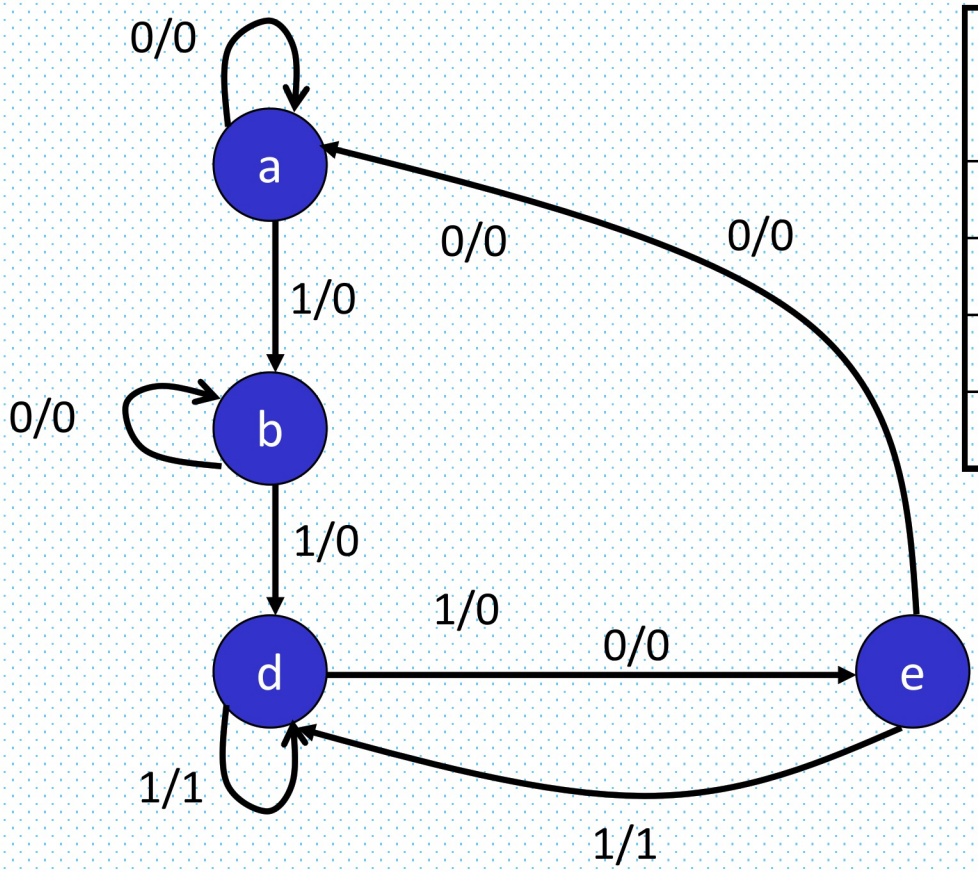
- We keep looking for equivalent states

State Reduction Technique 6/7

present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	b	d	0	0
d	e	d	0	1
e	a	d	0	1

- We stop when there are no equivalent states

State Reduction Technique 7/7



present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	b	d	0	0
d	e	d	0	1
e	a	d	0	1

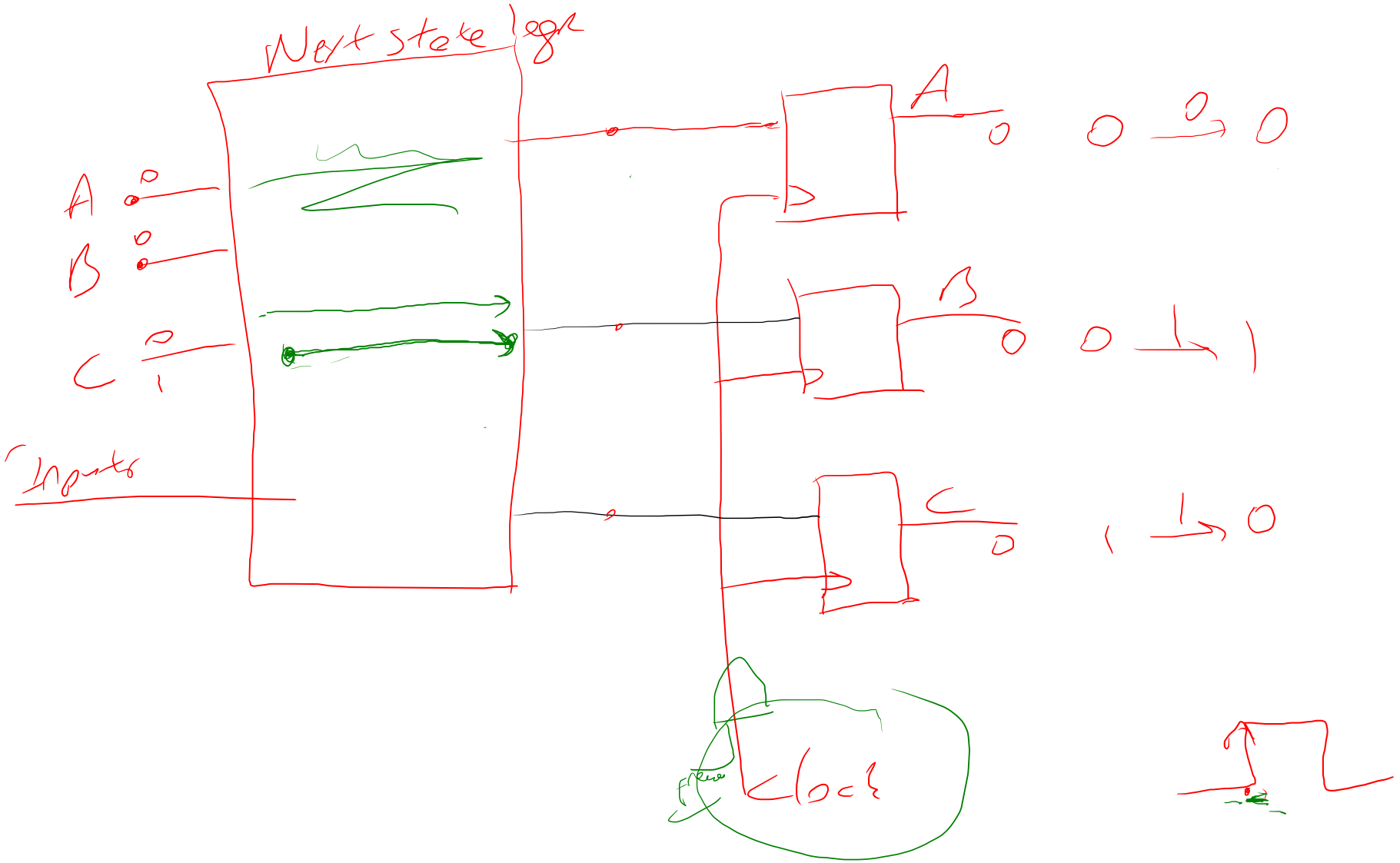
We need two flip-flops

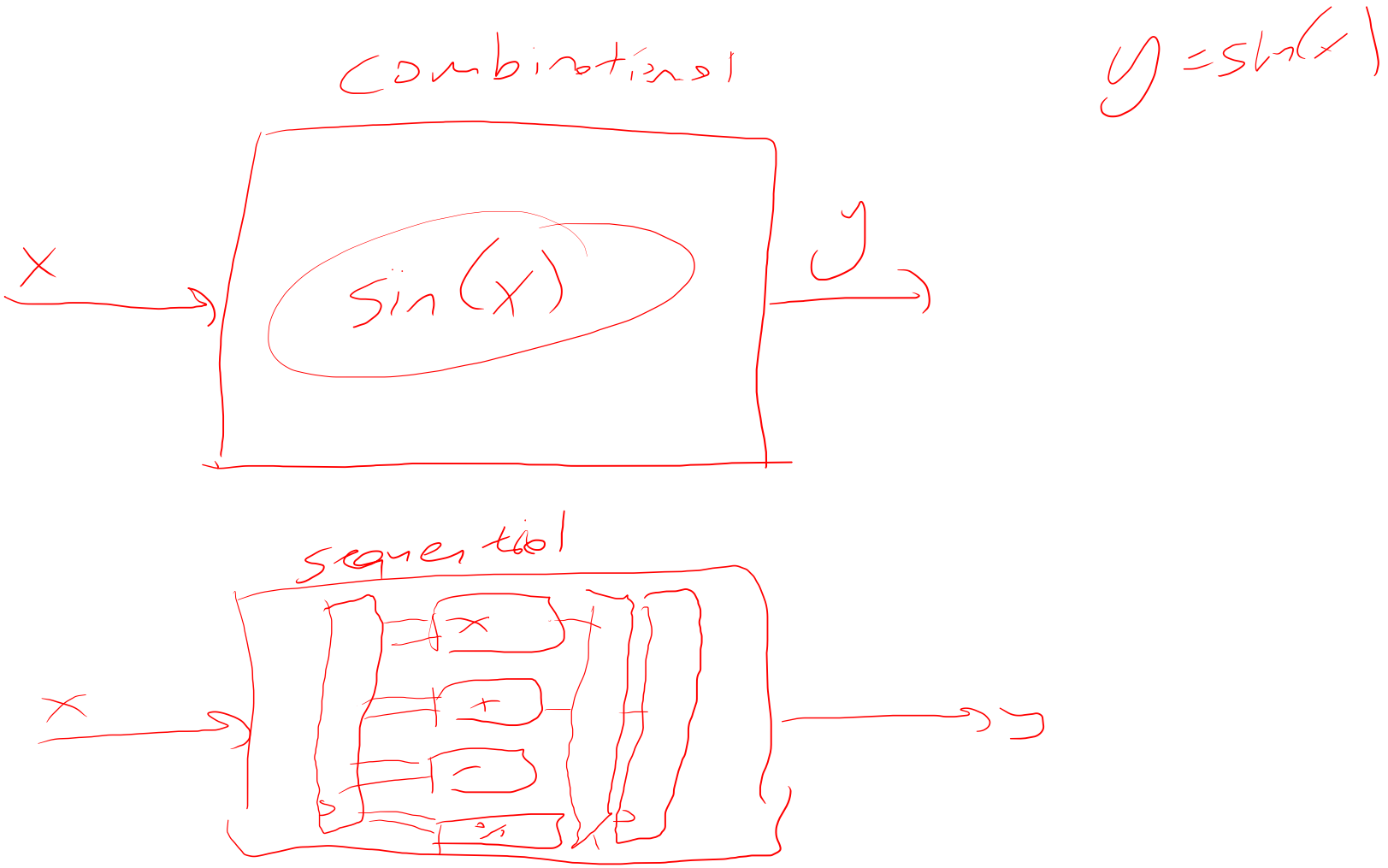
state	a	a	b	b	d	e	d	d	e	a	a		
input	0	1	0	1	0	1	1	0	0	0	0		
output	0	0	0	0	0	1	1	0	0	0			

State Assignments 1/4

- We have to assign binary values to each state
- If we have m states, then we need a code with minimum n bits, where $n = \lceil \log_2 m \rceil$
- There are different ways of encoding
- Example: Eight states: S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7

State	Binary	Gray	One-hot
S_0	000	000	00000001
S_1	001	001	00000010
S_2	010	011	00000100
S_3	011	010	00001000
S_4	100	110	00010000
S_5	101	111	00100000
S_6	110	101	01000000
S_7	111	100	10000000

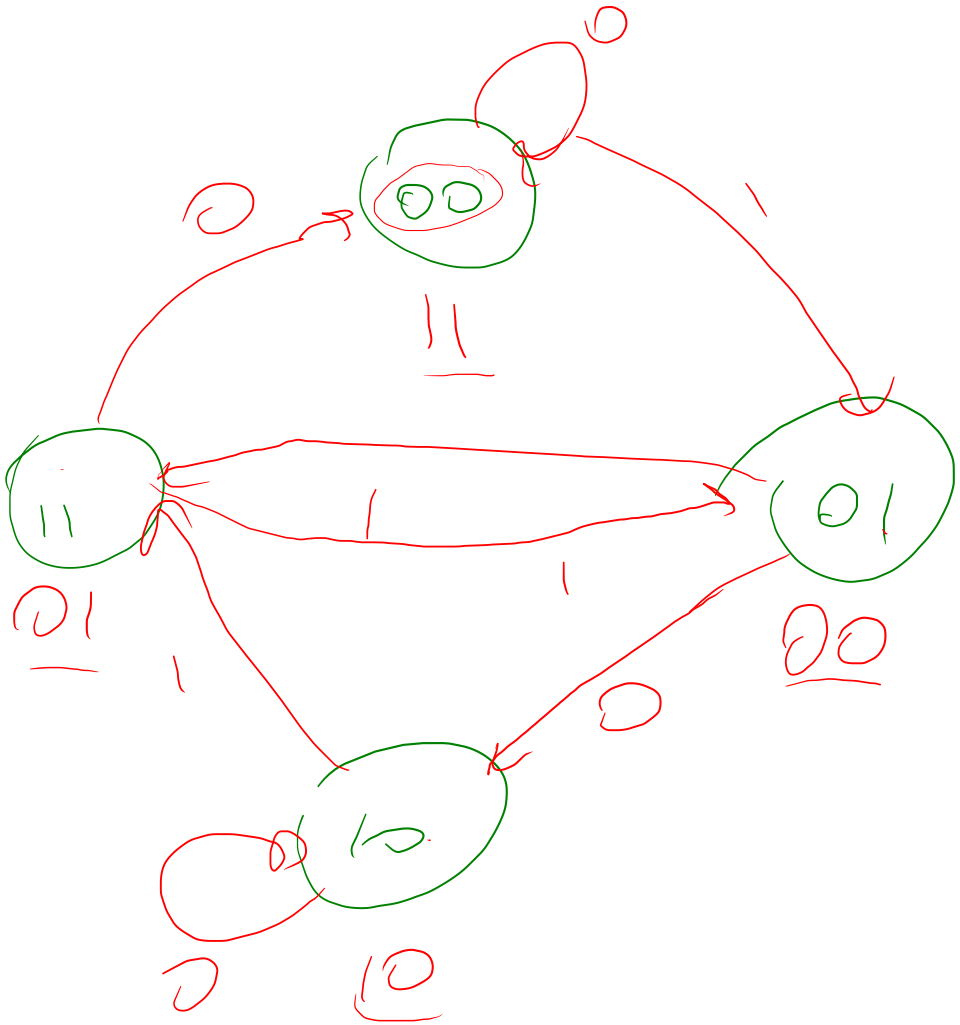

Next state logic
A
B
C
Inputs
A
0
0 → 0
B
0
0 → 1
C
0
1 → 0
Clock


combinational
y = sin(x)
x
sin(x)
y
sequential
x
x
y

State Assignments 2/4

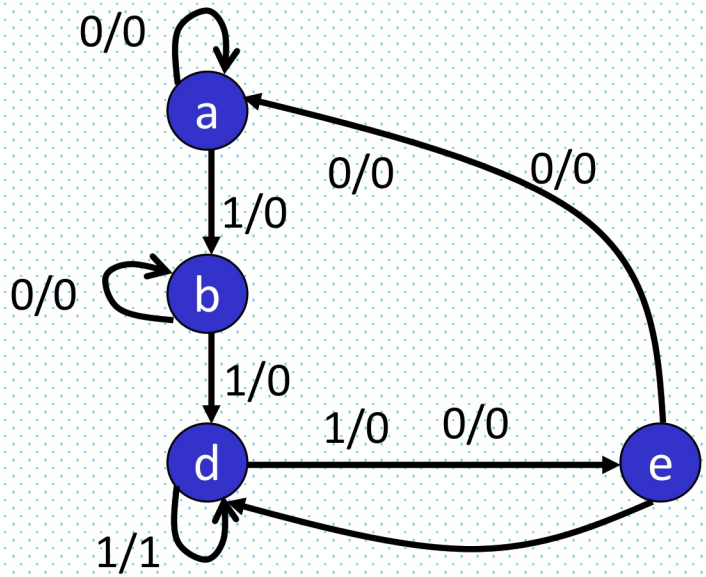
- The circuit complexity depends on the state encoding (assignment) scheme
- Previous example: binary state encoding

present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
(a) 00	00	01	0	0
(b) 01	01	10	0	0
(d) 10	11	10	0	1
(e) 11	00	10	0	1



State Assignments 3/4

- Gray encoding



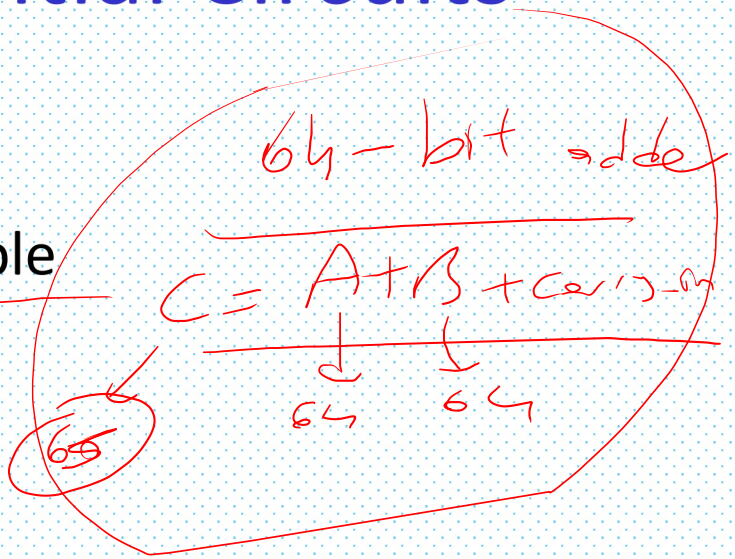
present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
(a) 00	00	01	0	0
(b) 01	01	11	0	0
(d) 11	10	11	0	1
(e) 10	00	11	0	1

State Assignments 4/4

- One-hot encoding

present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
(a) 0001	0001	0010	0	0
(b) 0010	0010	0100	0	0
(d) 0100	1000	0100	0	1
(e) 1000	0001	0100	0	1

Designing Sequential Circuits



- Combinational circuits
  - can be designed given a truth table
- Sequential circuits
  - We need,
    - state diagram or
    - state table
  - Two parts
    - flip-flops: number of flip-flops is determined by the number of states
    - combinational part:
      - output equations
      - flip-flop input equations

Design Process

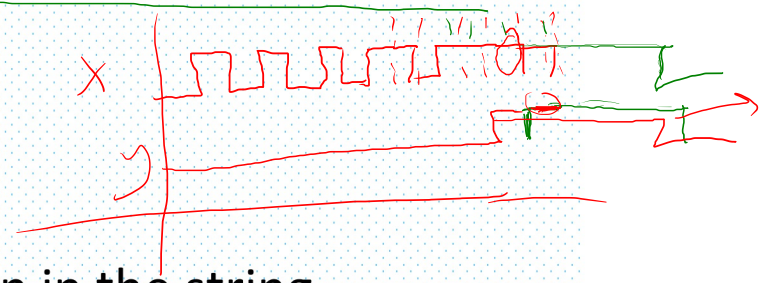
- Once we know the number of flip-flops, design process is reduced to design process of combinational circuits
- Therefore, we can apply the techniques of combinational circuit design
- The design steps

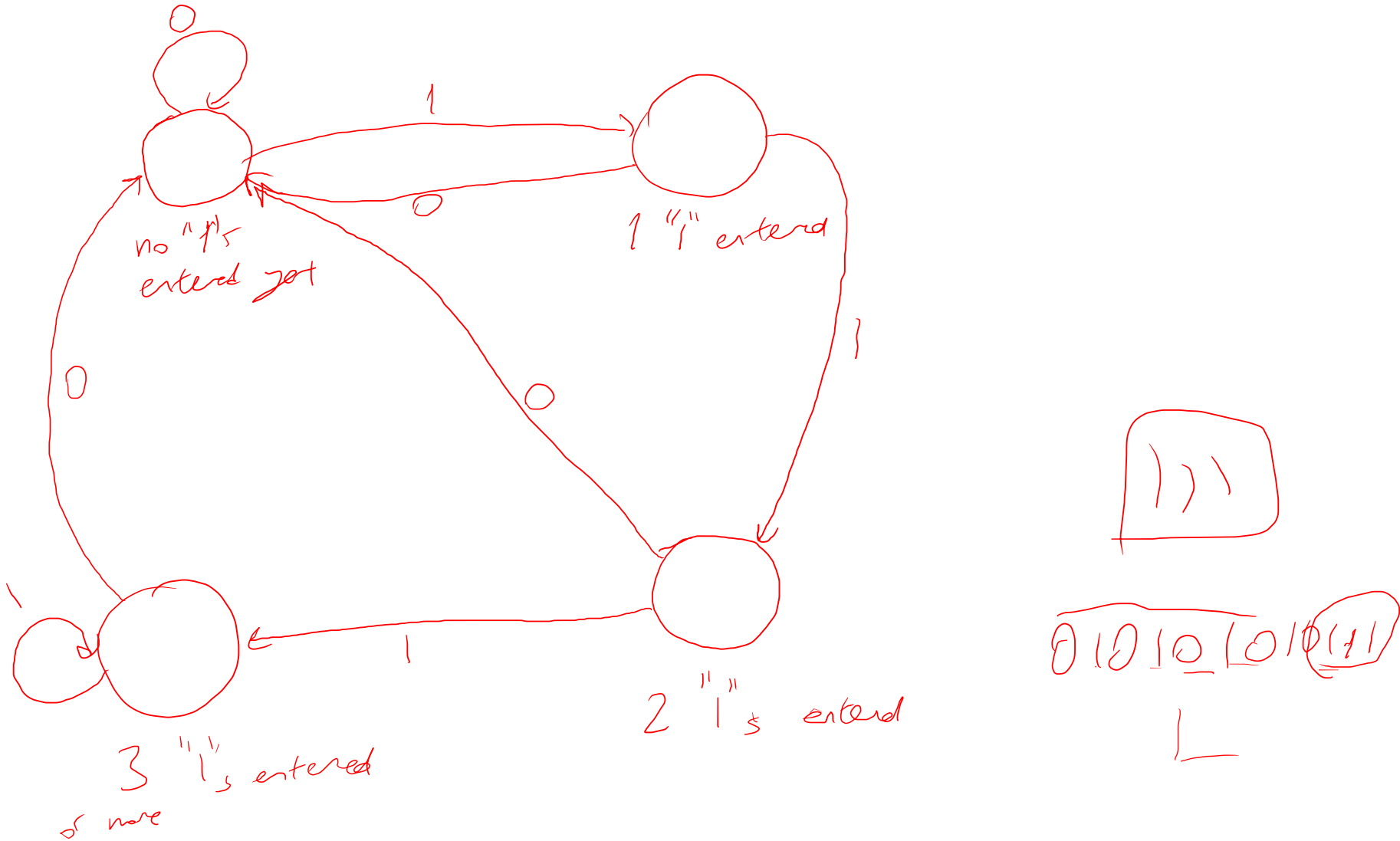
1. Given a verbal description of desired operation, derive state diagram
2. Reduce the number of states if necessary and possible
3. State assignment

Design Steps (cont.)

4. Obtain the encoded state table
5. Derive the simplified flip-flop input equations
6. Derive the simplified output equations
7. Draw the logic diagram

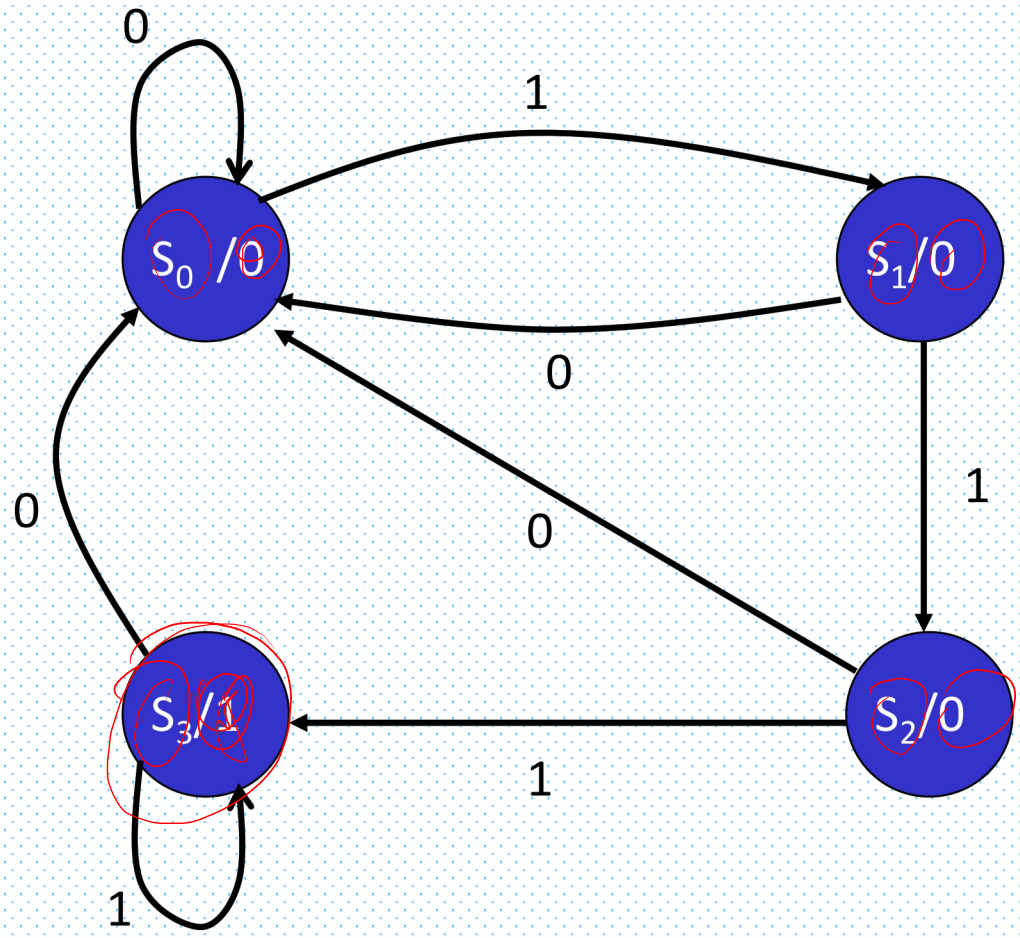
- Example: Verbal description
  - "we want a (sequential) circuit that detects three or more consecutive 1's in a string of bits"
  - Input: string of bits of any length
  - Output:
    - "1" if the circuit detects the pattern in the string
    - "0" otherwise




no "1"s entered yet
1 "1" entered
2 "1"s entered
3 "1"s entered or more
0
1
0
1
0
1
0
1
1
1
0 1 0 1 0 1 0 1 1

Example: State Diagram

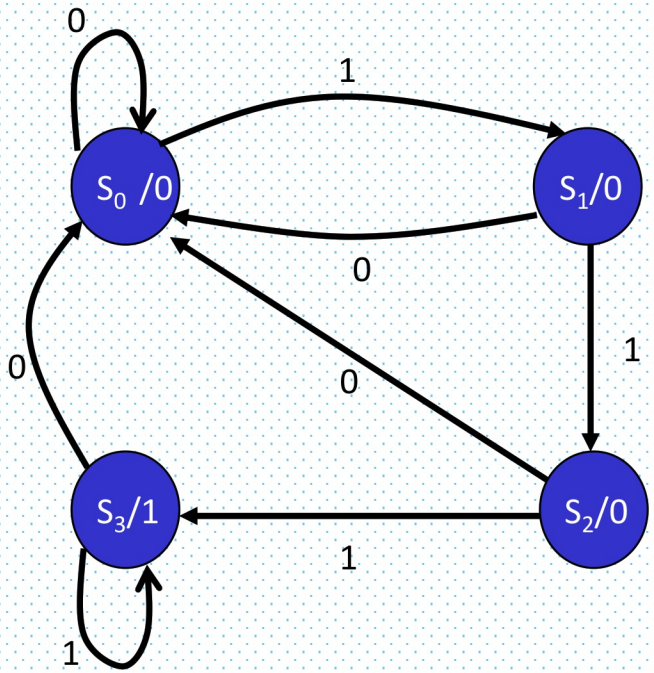
- Step 1: Derive the state diagram



Moore Machine

Synthesis with D Flip-Flops 1/5

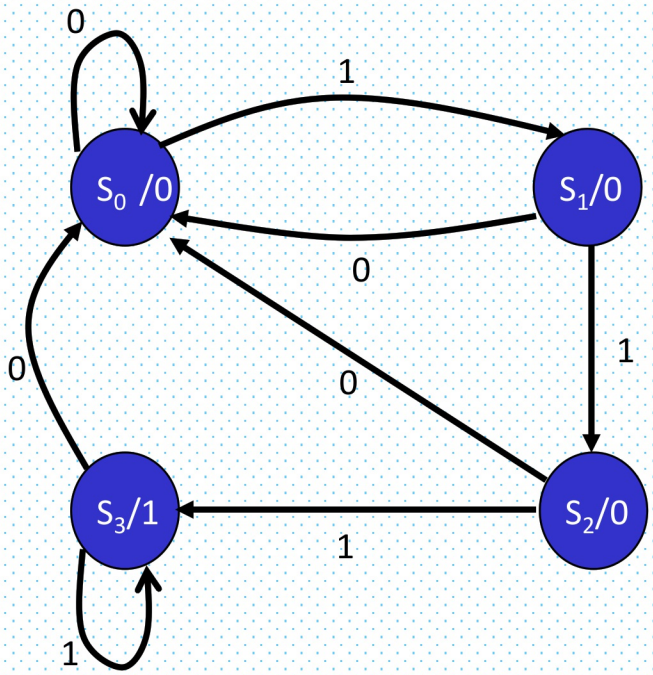
- The number of flip-flops
  - Four states
  - 2 or 4 flip-flops
- State reduction
  - not possible in this case
- State Assignment
  - Use binary encoding
    - S_0 → 00
    - S_1 → 01
    - S_2 → 10
    - S_3 → 11



Synthesis with D Flip-Flops 2/5

- Step 4: Obtain the state table

Present state		Input	Next state		Output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1



Synthesis with D Flip-Flops 3/5

- Step 5: Choose the flip-flops
  - D flip-flops
- Step 6: Derive the simplified flip-flop input equations
  - Boolean expressions for D_A and D_B

Present state		Input	Next state		Output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

A \ Bx	00	01	11	10
0	0	0	1	0
1	0	1	1	0

$$D_A = Ax + Bx$$

Synthesis with D Flip-Flops 3/5

Present state		Input	Next state		Output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

A \ Bx	00	01	11	10
0	0	1	0	0
1	0	1	1	0

$D_B = Ax + B'x$

- Step 7: Derive the simplified output equations
  - Boolean expressions for y.

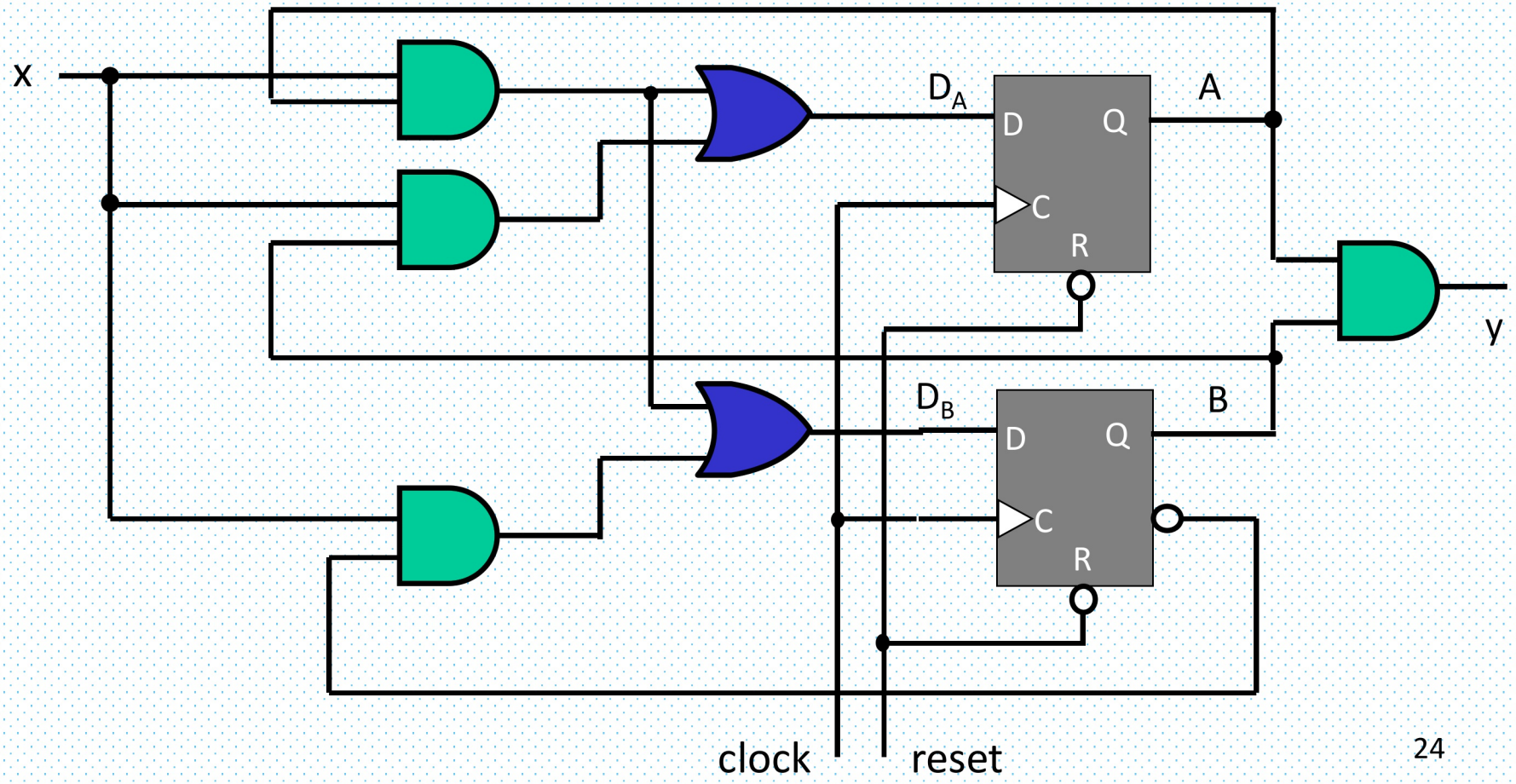
A \ Bx	00	01	11	10
0	0	0	0	0
1	0	0	1	1

$y = AB$

Synthesis with D Flip-Flops 5/5

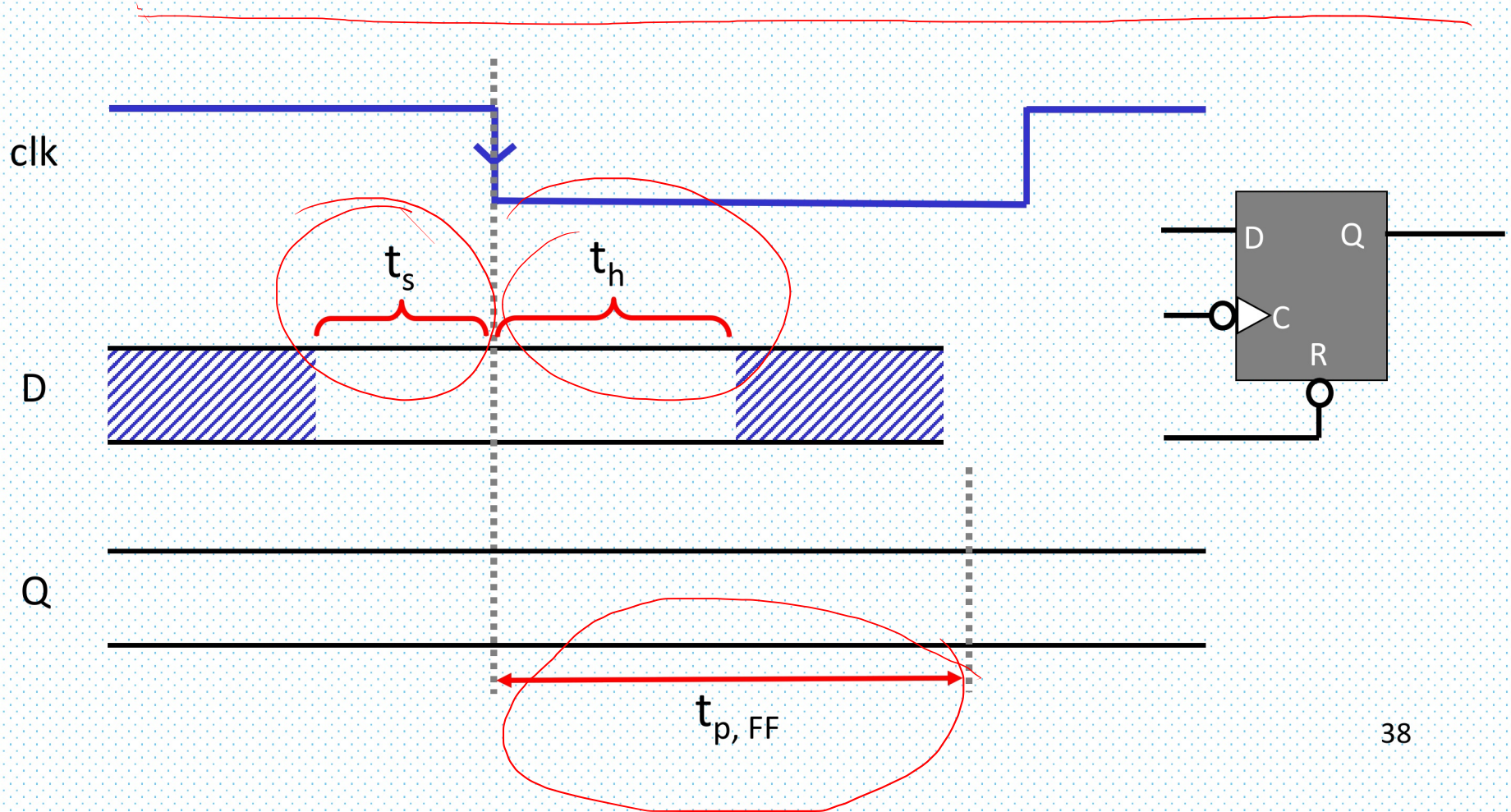
- Step 8: Draw the logic diagram

$D_A = Ax + Bx$ $D_B = Ax + B'x$ $y = AB$

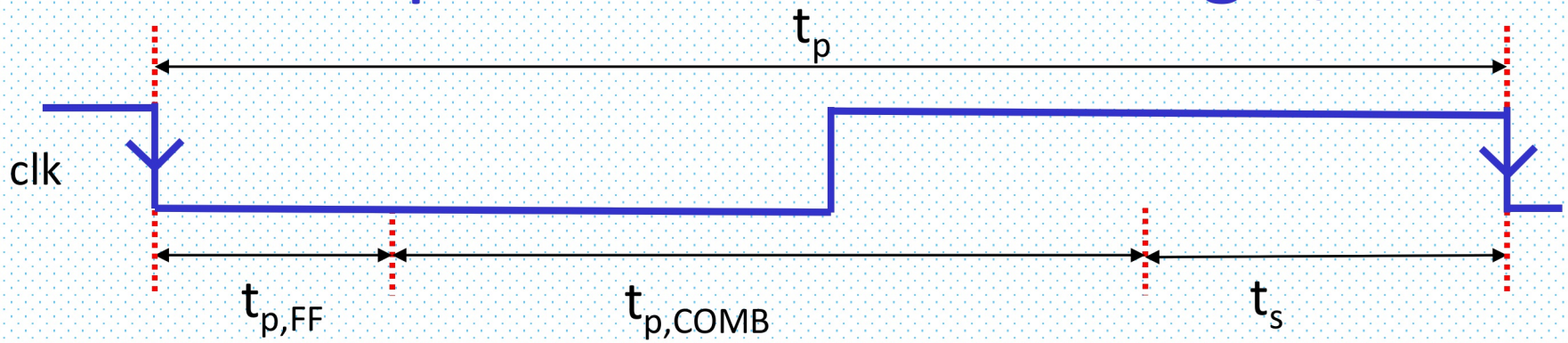


Sequential Circuit Timing 1/3

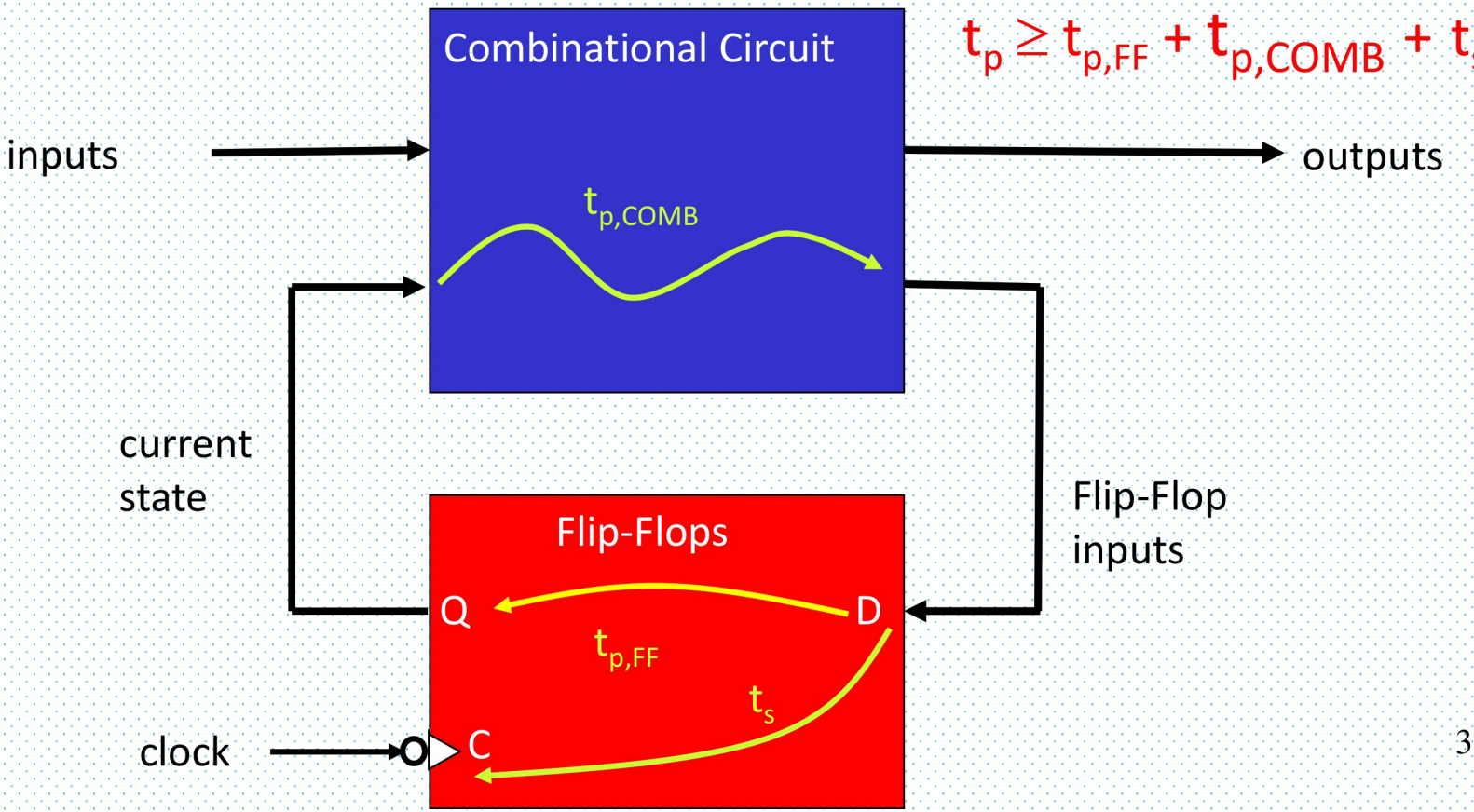
- It is important to analyze the timing behavior of a sequential circuit
  - Ultimate goal is to determine the maximum clock frequency



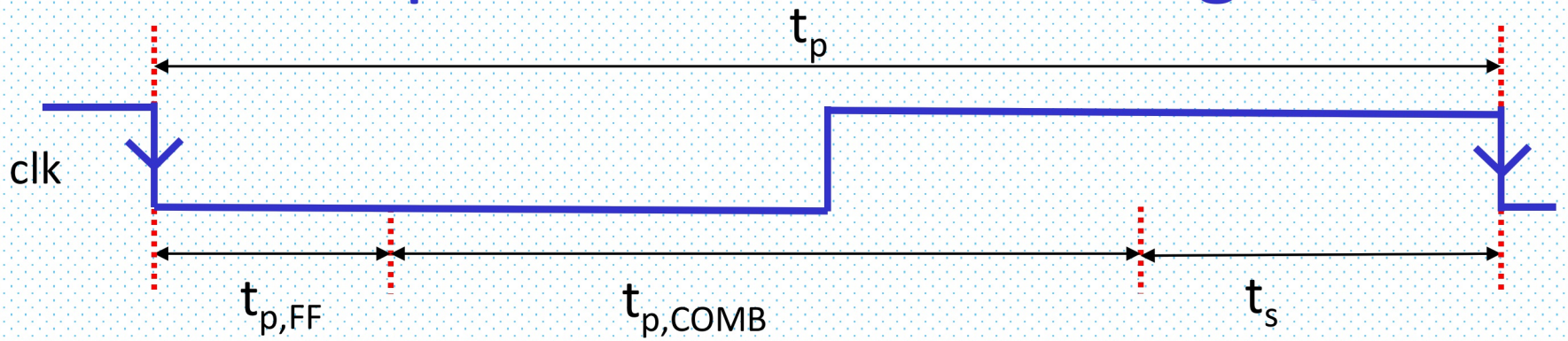
Sequential Circuit Timing 2/3



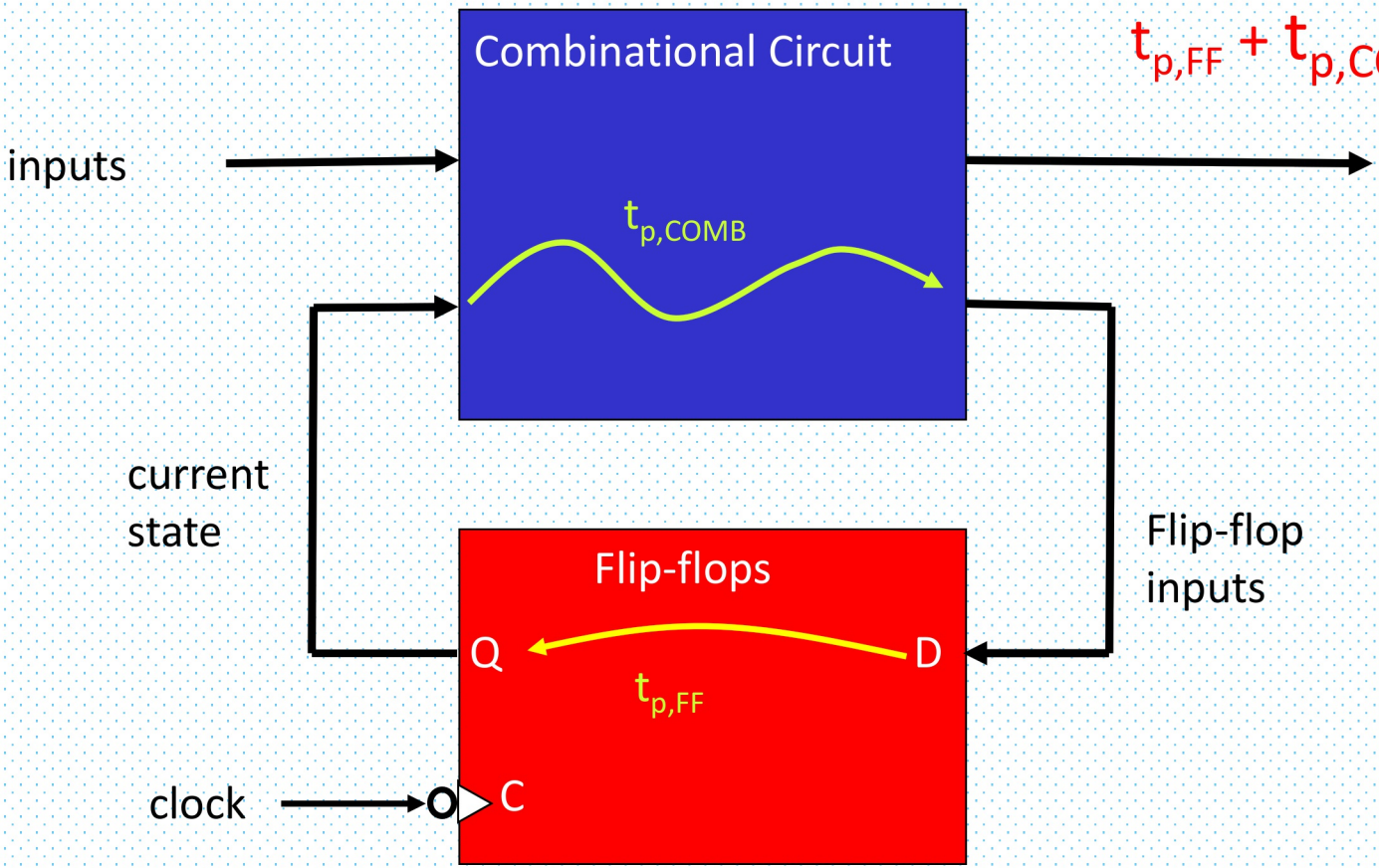
$$t_p \geq t_{p,FF} + t_{p,COMB} + t_s$$



Sequential Circuit Timing 2/3

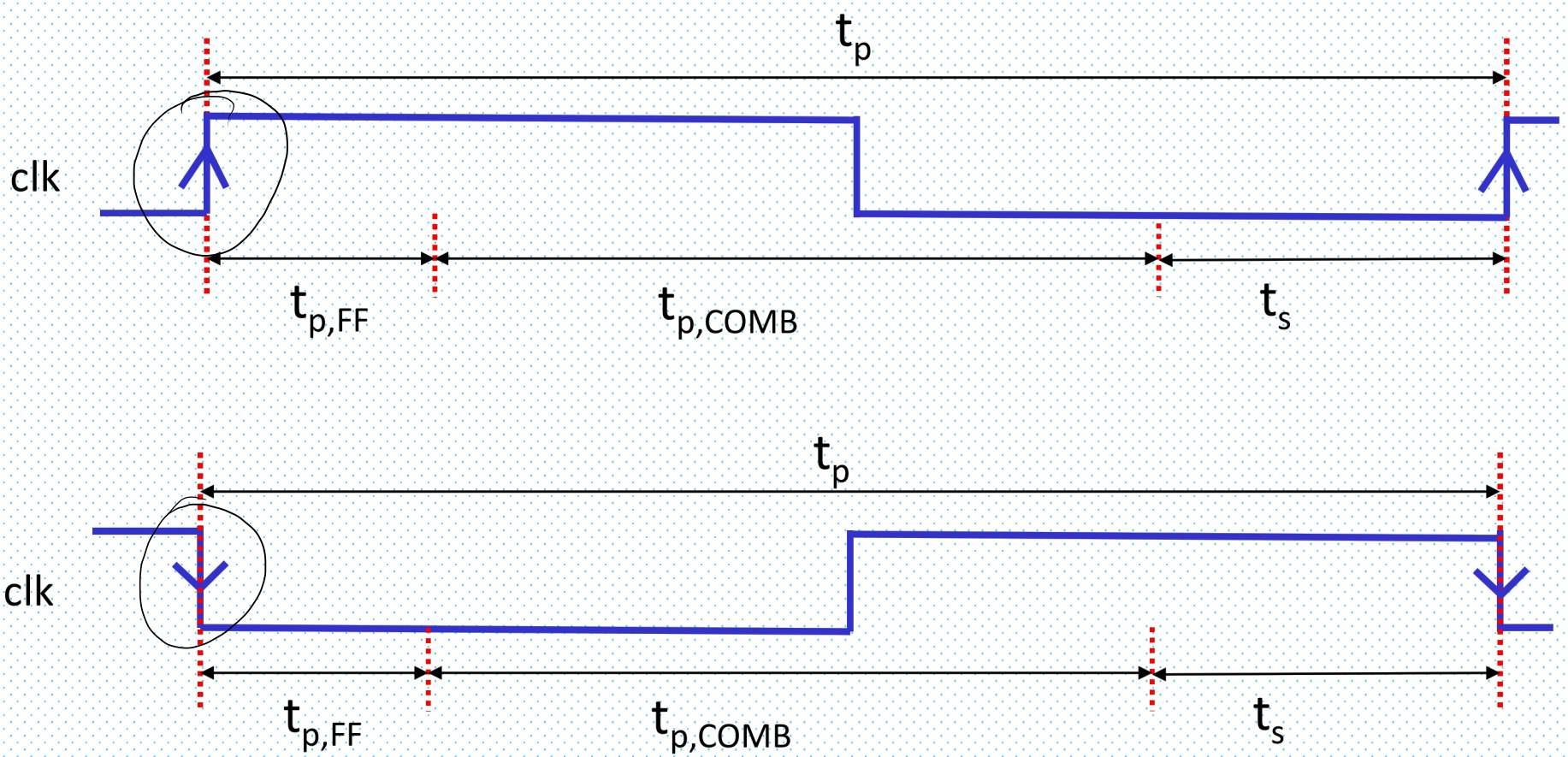


$t_{p,FF} + t_{p,COMB} >> t_h$



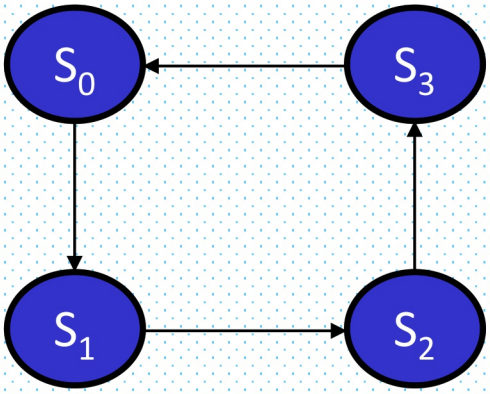
Sequential Circuit Timing 3/3

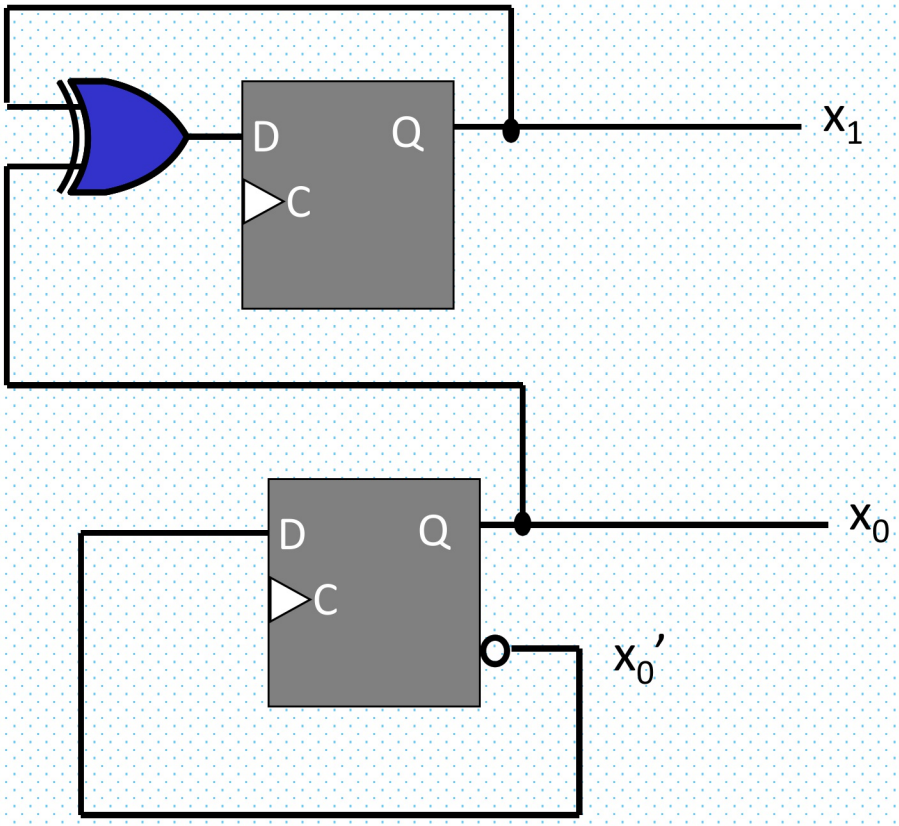
- Minimum clock period (or maximum clock frequency)



Example: Counter

Binary encoding





$t_{p,XOR} = 2.0$ ns

$t_{p,FF} = 2.0$ ns

$t_s = 1.0$ ns

$t_p = t_{p,FF} + t_{p,XOR} + t_s = 2.0 + 2.0 + 1.0 = 5.0$ ns

$f_{max} = 1/t_p = 1/(5.0 \times 10^{-9}) \approx 200$ MHz

Example: One-Hot-Encoding

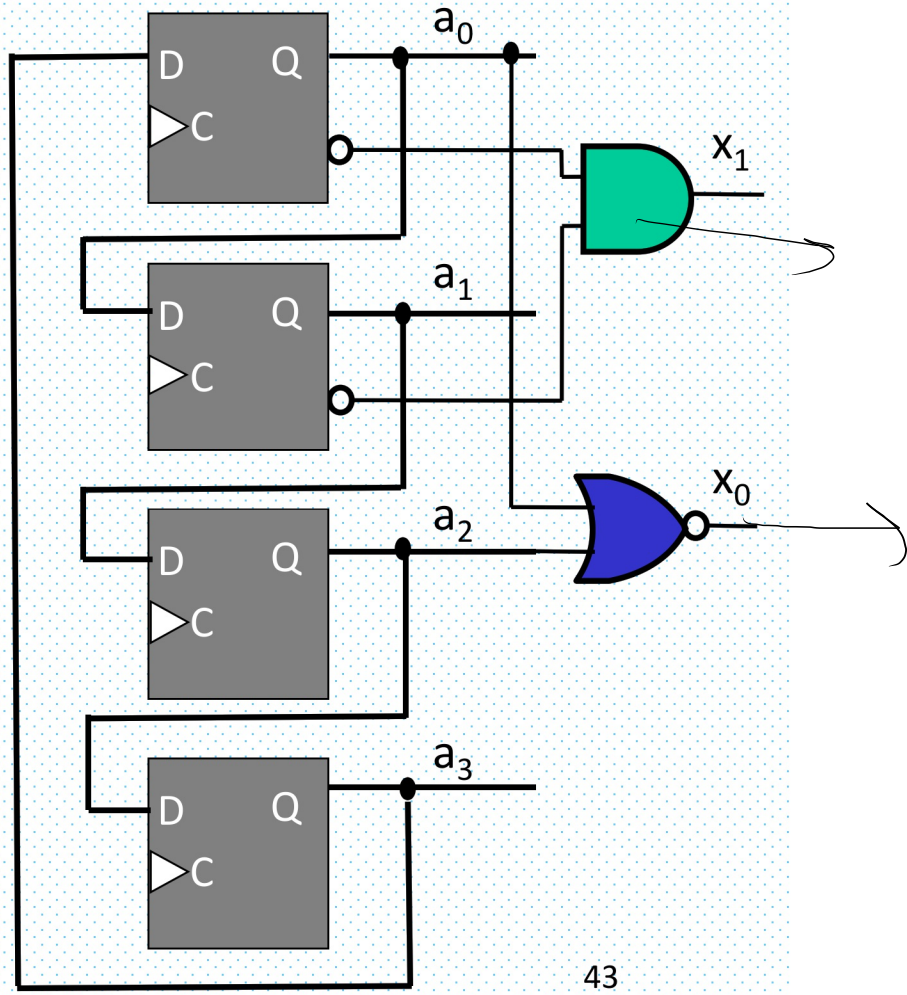
$S_0 \rightarrow 0001$
$S_1 \rightarrow 0010$
$S_2 \rightarrow 0100$
$S_3 \rightarrow 1000$

$t_{p,FF} = 2.0$ ns

$t_s = 1.0$ ns

$t_p = t_{p,FF} + t_s = 2.0 + 1.0 = 3.0$ ns

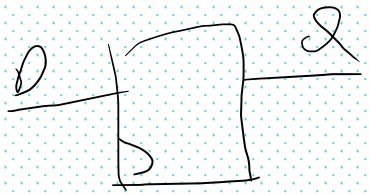
$f_{max} = 1/t_p = 1/(3.0 \times 10^{-9}) \approx 333$ MHz



State-Diagram Based HDL Models

- **always @** () statement
  - For clocked sequential circuits we use "**always @** ()" statement
  - Inside we have signals such as "clock" and "reset"
  - **always @** () statement may have a number of "edge events"
- Examples:

```
// D flip-flop
module DFF(output reg Q, input D, clk, reset);
   always @(posedge clk, posedge reset)
      if (reset == 1) Q <= 1'b0;
      else Q <= D;
endmodule
```



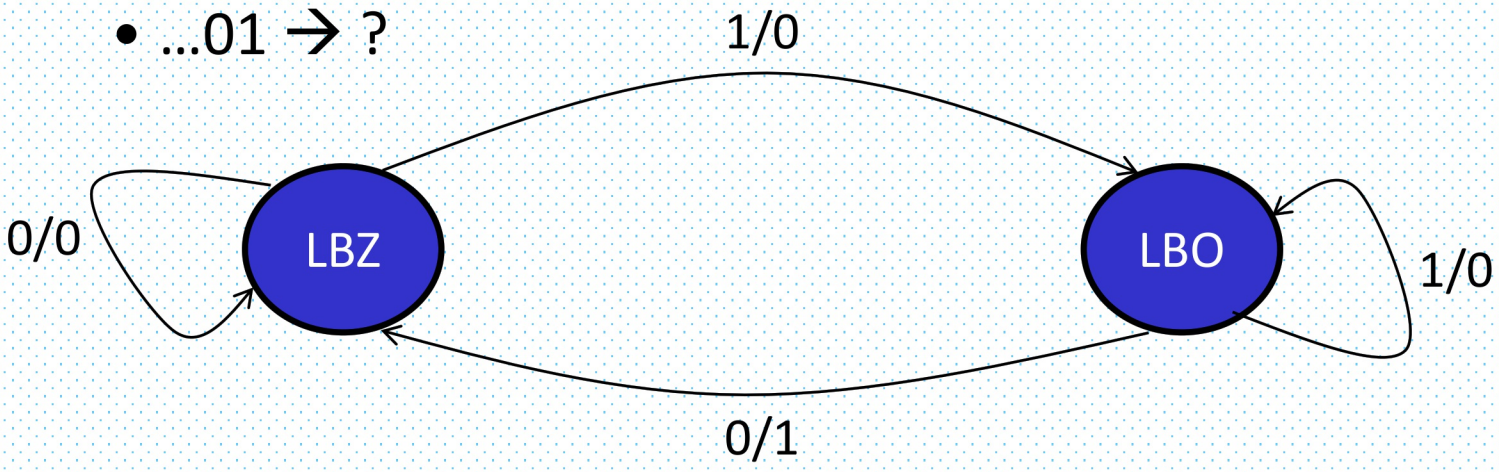
State-Diagram Based HDL Models

```
// T flip-flop
module TFF(output reg Q, input T, clk, reset);
   always @(posedge clk, negedge reset)
      if (reset == 0) Q <= 1'b0;
      else Q <= T ^ Q;
endmodule

module T (output reg Q, input T, input C);
   always @(C or T)
      if (C == 1)
         Q <= T ^ Q;
endmodule
```

Sequential Circuit Design with Verilog

- Zero detector circuit
  - The circuit detects a 0 following a sequence of 1s in a serial bit stream.
  - For instance,
    - …10 → 1,
    - …110 → 1,
    - …00 → 0
    - …01 → ?

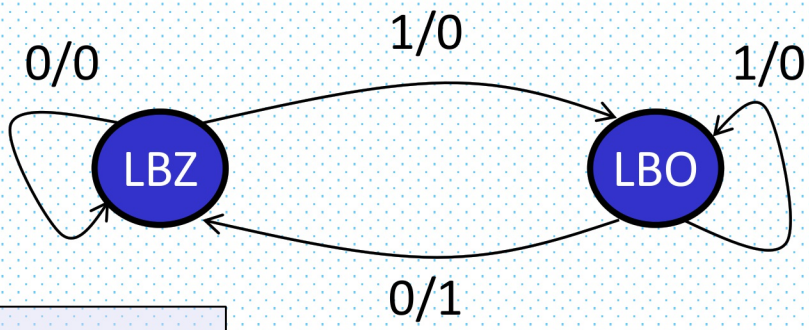


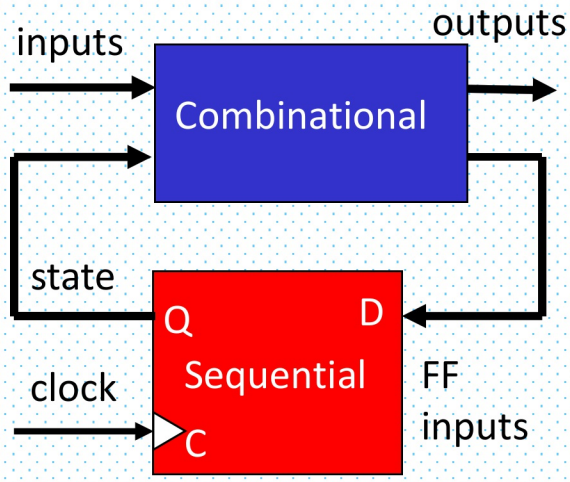
Zero Detector Circuit

```
module zero_detector(output reg y, input x, clk, reset);
   reg state, next_state;
   parameter LBZ = 1'b0, LBO = 1'b1;
   always @(posedge clk, negedge reset)
      if (reset == 0) state <= LBZ;
      else state <= next_state;
   always @(state, x)
      case(state)
         LBZ: if(~x) next_state = LBZ; else next_state = LBO;
         LBO: if(~x) next_state = LBZ; else next_state = LBO;
      endcase
   always @(state, x)
      case(state)
         LBZ: y = 0; LBO: y = ~x;
      endcase
endmodule
```

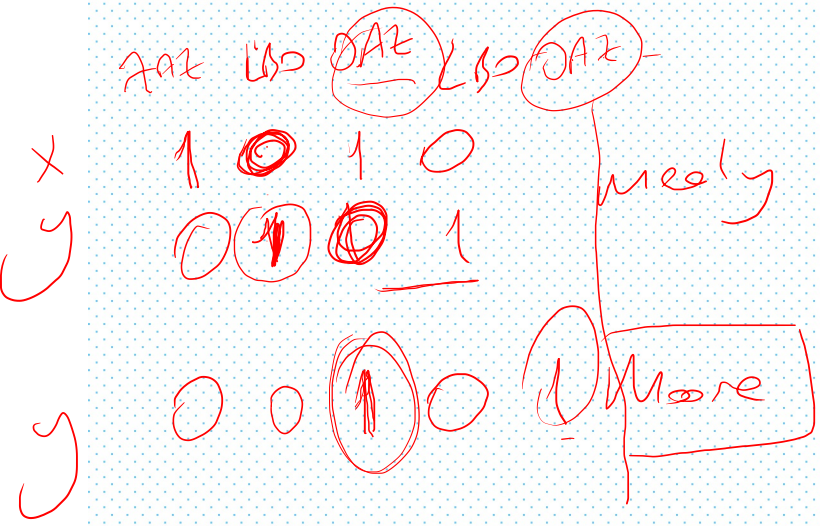
Sequential Part

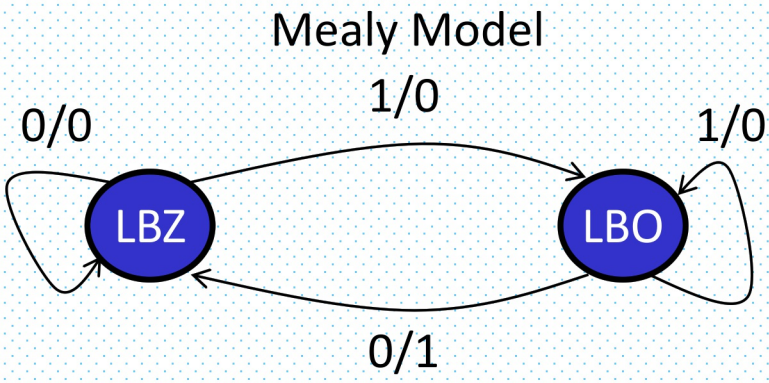
Combinational Part

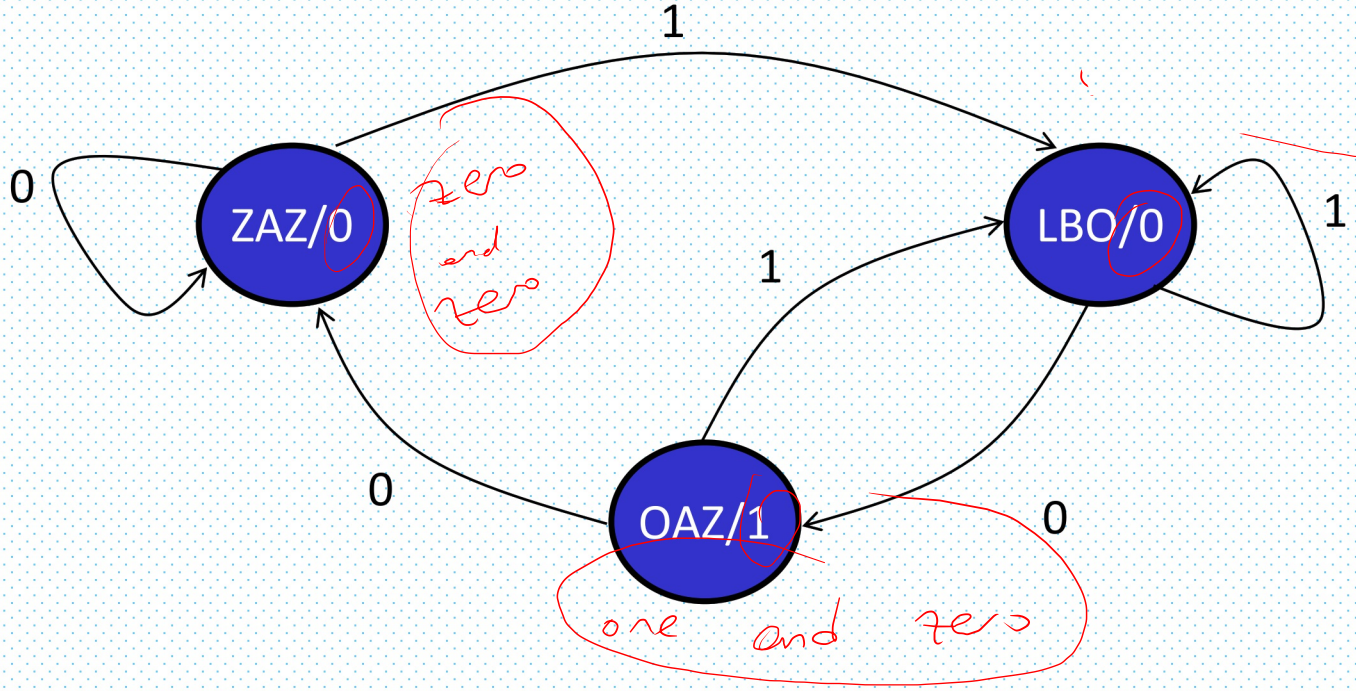




Zero Detector Circuit (Moore Model)







Zero Detector Circuit (Moore Model)

```
module zero_detector(output reg y, input x, clk, reset);
  reg[1:0] state;
  parameter ZAZ = 2'b00, LBO = 2'b01,  OAZ = 2'b10;
  always @(posedge clk, negedge reset)
    if (~reset) state <= ZAZ;
    else case(state)
      ZAZ: if(~x) state <= ZAZ; else state <= LBO;
      LBO: if(~x) state <= OAZ; else state <= LBO;
      OAZ: if(~x) state <= ZAZ; else state <= LBO;
    endcase
  always @(state)
    case(state)
      ZAZ, LBO: y = 0;
      OAZ: y = 1;
    endcase
endmodule
```