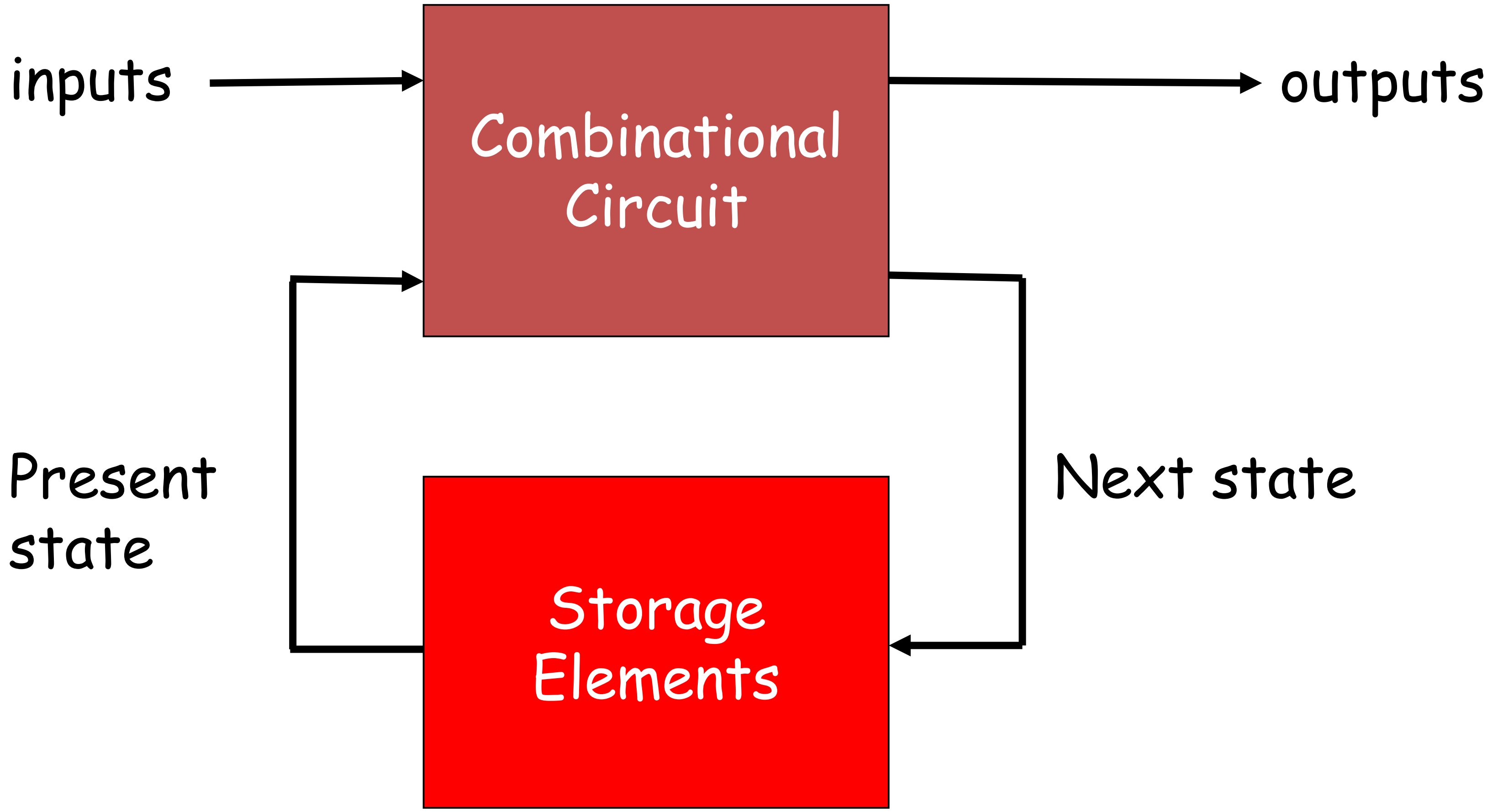
Sequential Circuit Model



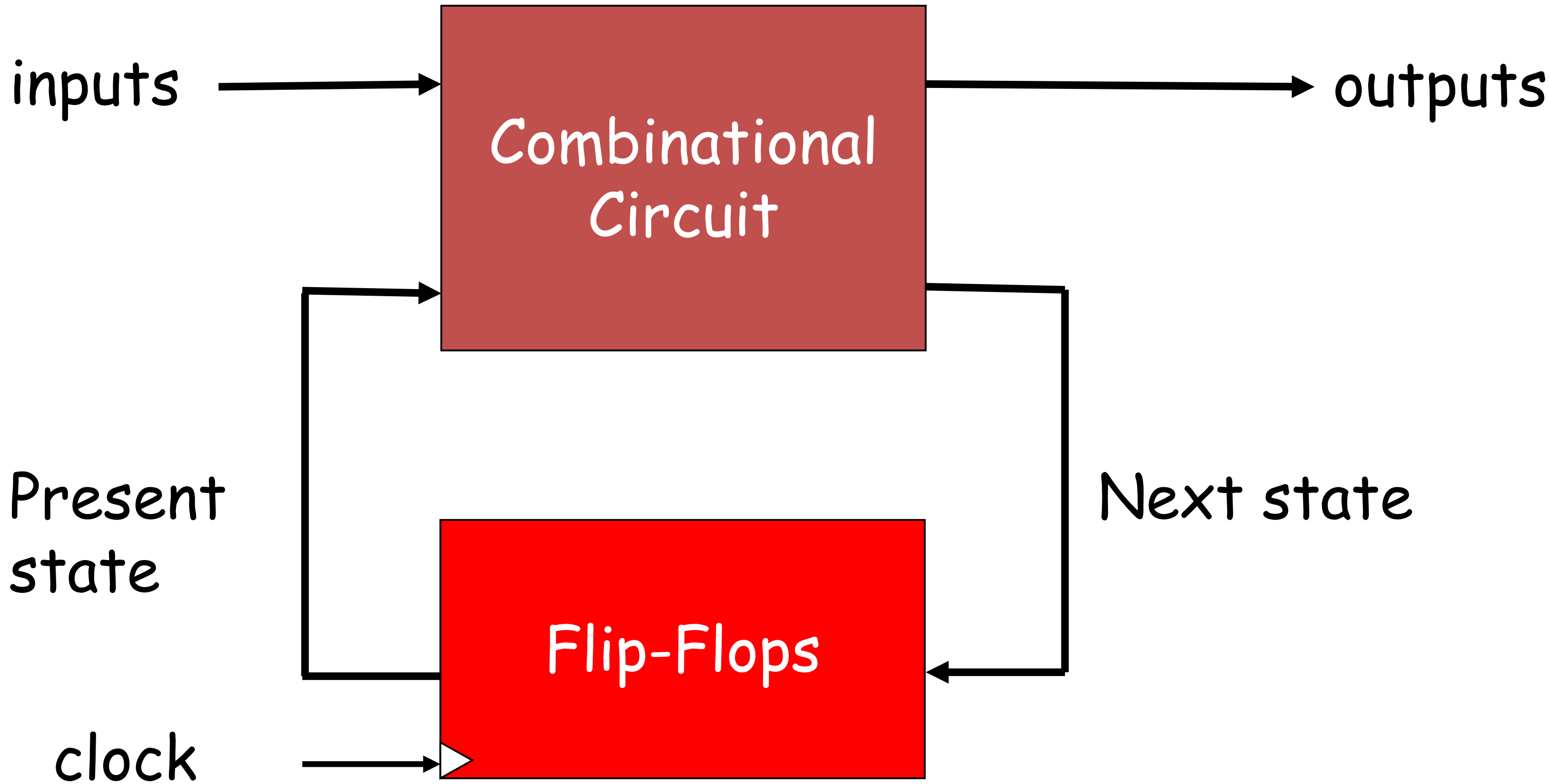
Present state depends on the previous inputs

Synchronous Sequential Circuits

- Behavior defined from knowledge of its signals at discrete instances of time.
- Discrete instances of time need synchronization.
- Synchronization is done by a common clock signal.
- Clock signal is a periodic square signal.
- Storage elements observe inputs and can change state only in relation to a timing signal (clock pulses from a clock)

Synchronous Sequential Circuits

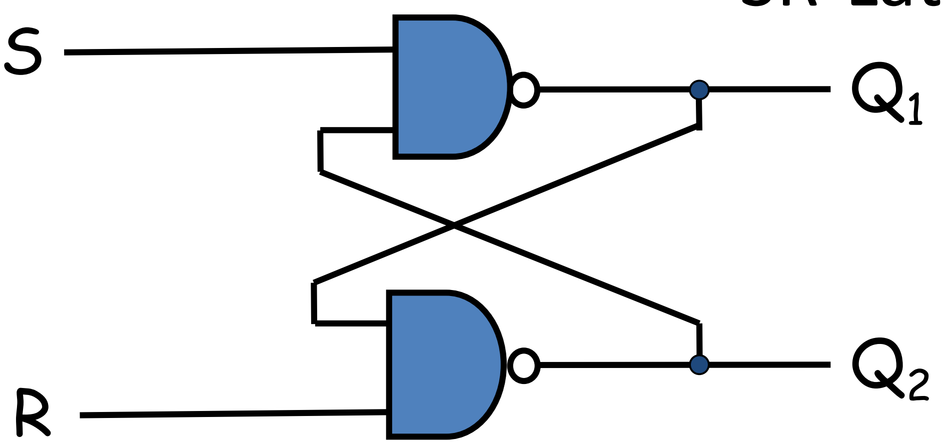
- The storage elements are flip-flops that can store one bit of information.



Latch

- Basic storage element
- A latch is a storage element that can store its content forever.
- Latches are asynchronous circuits and do not need a clock signal to operate.
- Hence they can not be used in synchronous circuits directly.
- They are used to construct flip-flops.

SR-Latch



$$Q_1 = (S\,Q_2)' = S' + q_2'$$
$$Q_2 = (R\,Q_1)' = R' + q_1'$$

S	R	q_1	q_2	Q_1	Q_2
0	0	0	0	1	1
0	0	0	1	1	1
0	0	1	0	1	1
0	0	1	1	1	1
0	1	0	0	1	1
0	1	0	1	1	1
0	1	1	0	1	0
0	1	1	1	1	0
1	0	0	0	1	1
1	0	0	1	0	1
1	0	1	0	1	1
1	0	1	1	0	1
1	1	0	0	1	1
1	1	0	1	0	1
1	1	1	0	1	0
1	1	1	1	0	0

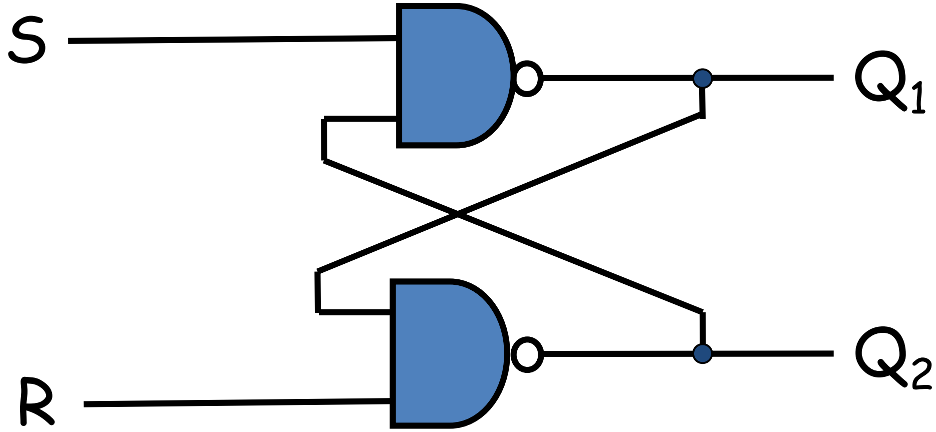
SR=00 $\Rightarrow$ 00-11-11-...
01-11-11-...
10-11-11-...
} Q_1Q_2=11

SR=01 $\Rightarrow$ 00-11-10-10-...
01-11-10-10-...
} Q_1Q_2=10

SR=10 $\Rightarrow$ 00-11-01-01-...
10-11-01-01-...
} Q_1Q_2=01

SR=11 $\Rightarrow$ 00-11-00-11-... } Q_1Q_2=osilates
01-01-...
10-10-...
} $Q_1Q_2=q_1q_2$

SR-Latch



S	R	Q_1	Q_2
0	0	x	x
0	1	1	0
1	0	0	1
1	1	q_1	q_2

Undefined

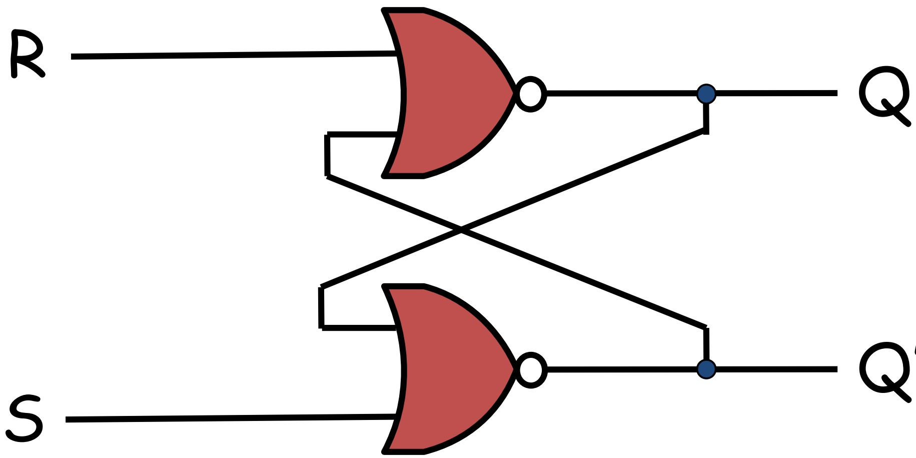
$Q_1 = Q_2'$

$Q_1 = Q$

$Q_2 = Q'$

6

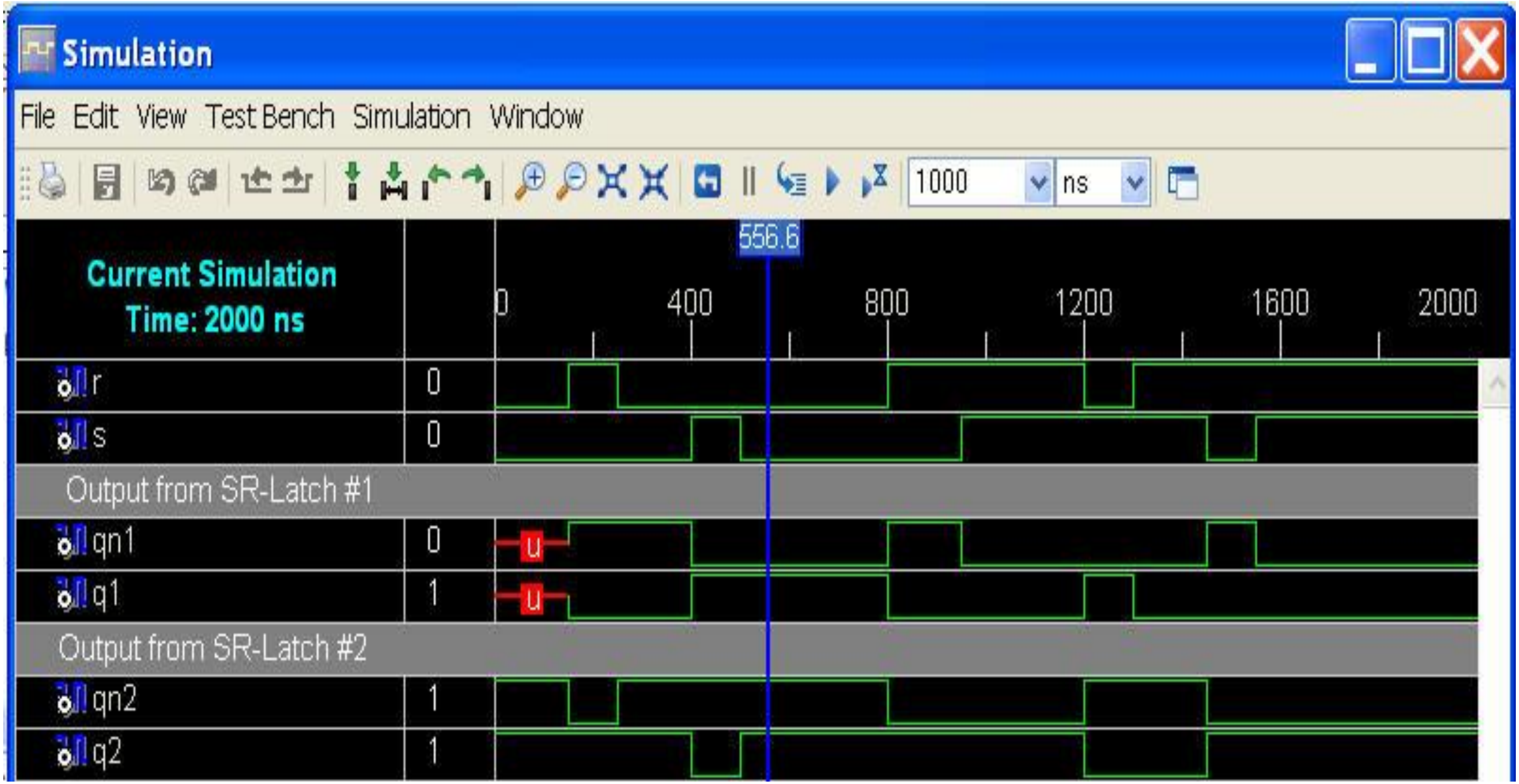
SR-Latch



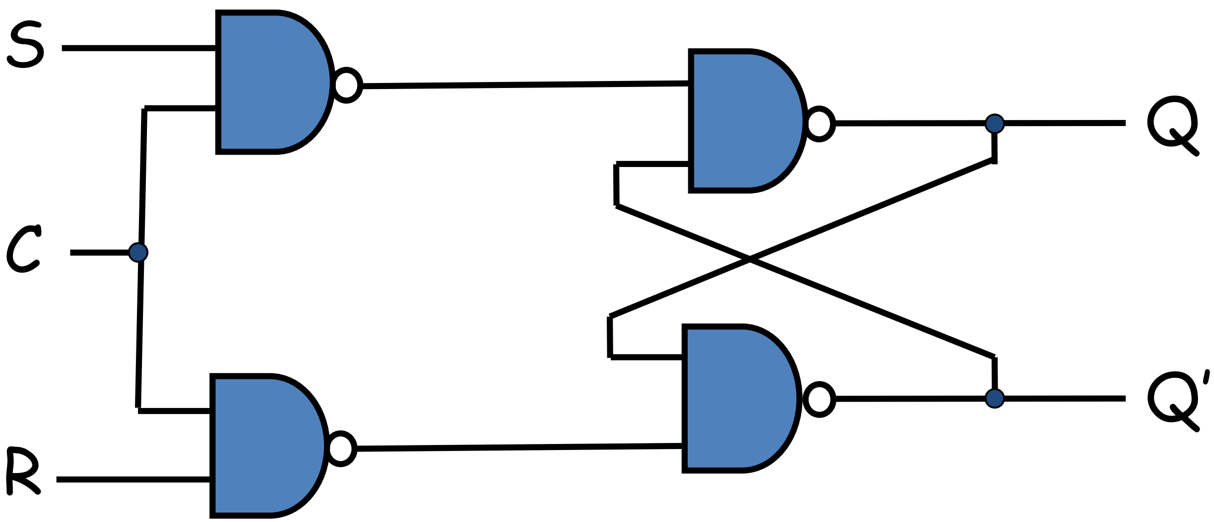
$Q = (R + Q')' = R'\,q$
$Q' = (S + Q)' = S'\,q'$

S	R	Q	Q'	
0	0	q	q'	
0	1	0	1	
1	0	1	0	
1	1	X	X	Undefined

Simulation of SR-Latch



SR-Latch with Control Input

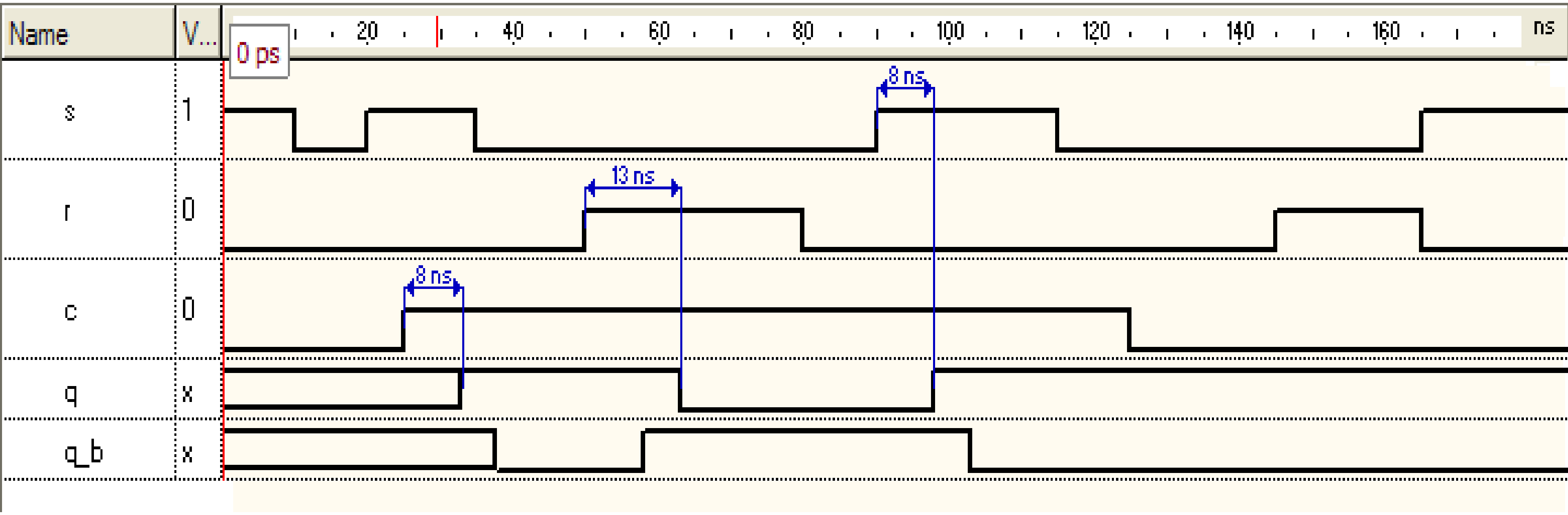


$Q = ((S\,C)'Q')' = SC + Q$

$Q' = ((R\,C)'Q)' = RC + Q'$

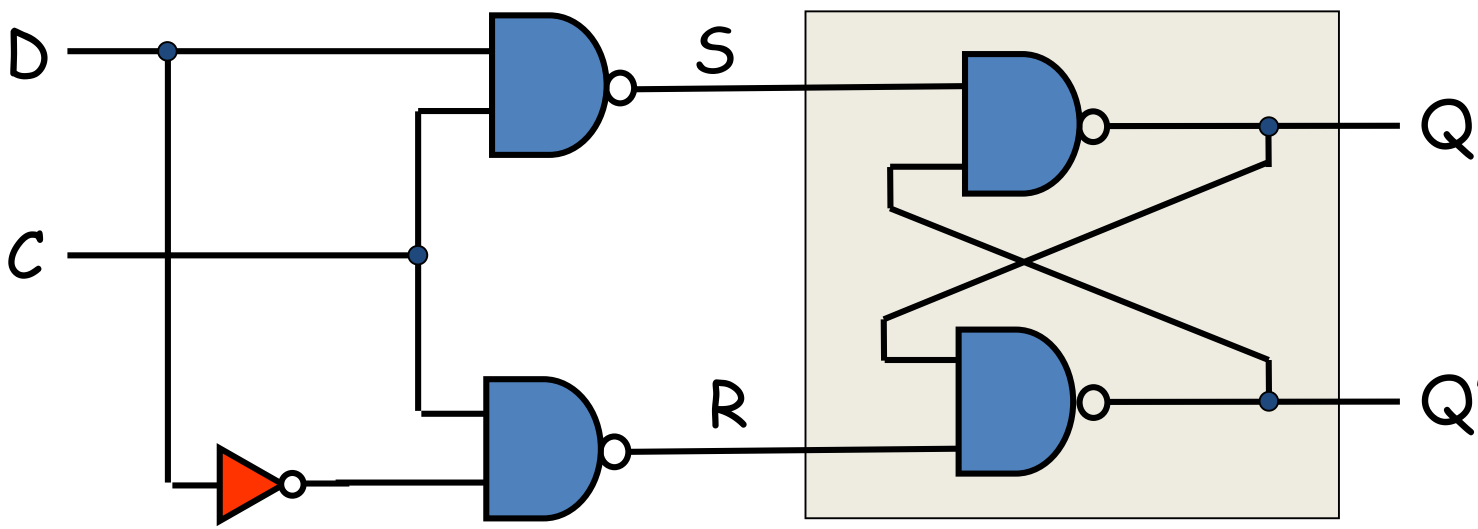
C	S	R	Q	Q′
0	X	X	No change	
1	0	0	No change	
1	0	1	Q = 0	Reset state
1	1	0	Q = 1	Set state
1	1	1	Undefined	

Simulation of SR-Latch with Control Input



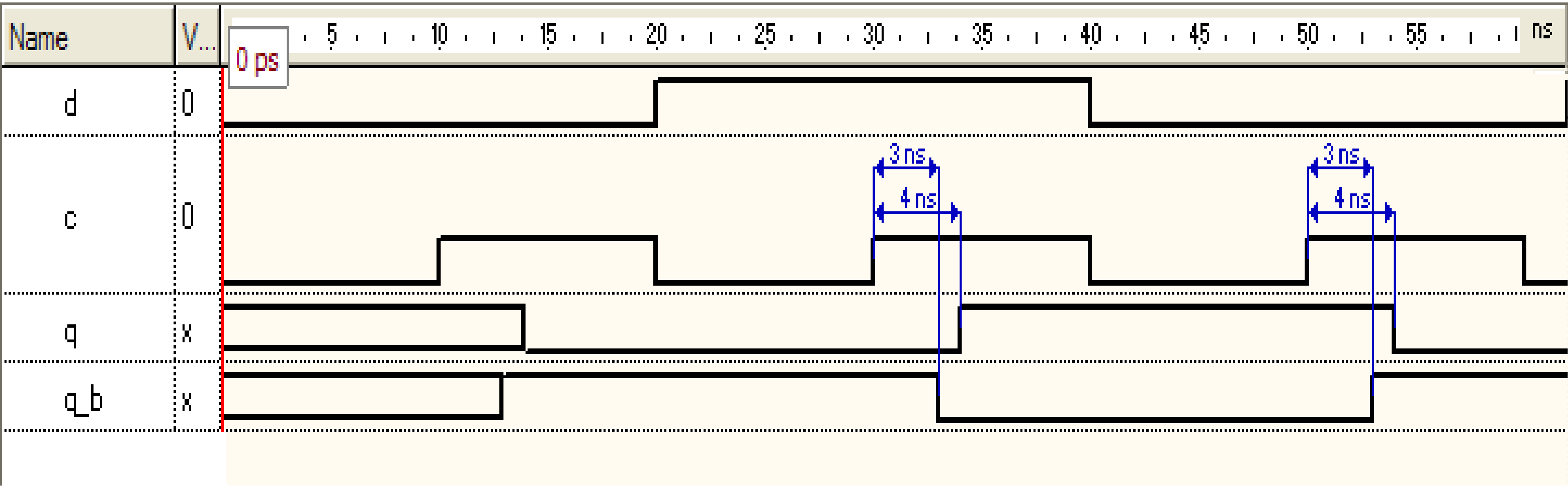
D-Latch

- Because the undefined situation can cause stability problems, SR latches are not used offen.
- <u>Solution</u>: D-latch



This circuit garanties that S and R inputs are always each other's complement.

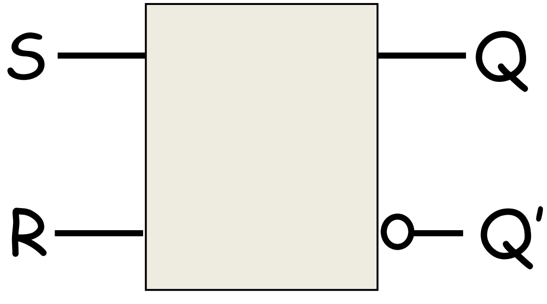
Simulation of D-Latch



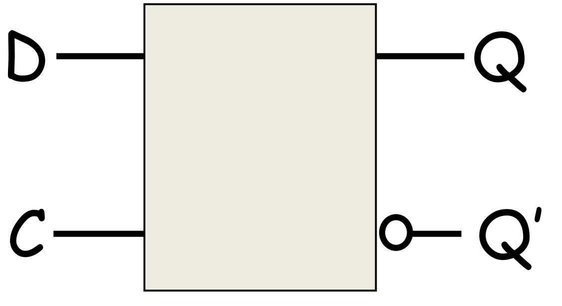
D-Latch

C	D	Next state of Q
0	X	No change
1	0	Q = 0; reset state
1	1	Q = 1; set state

- D input is sampled when C=1.



SR-latch



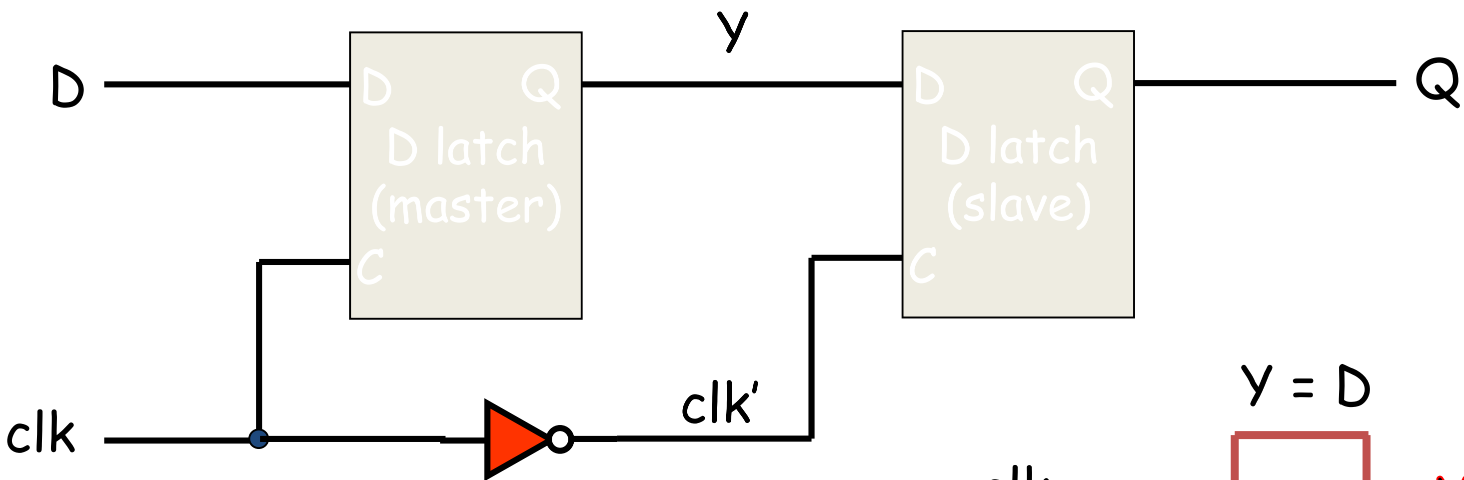
D-latch

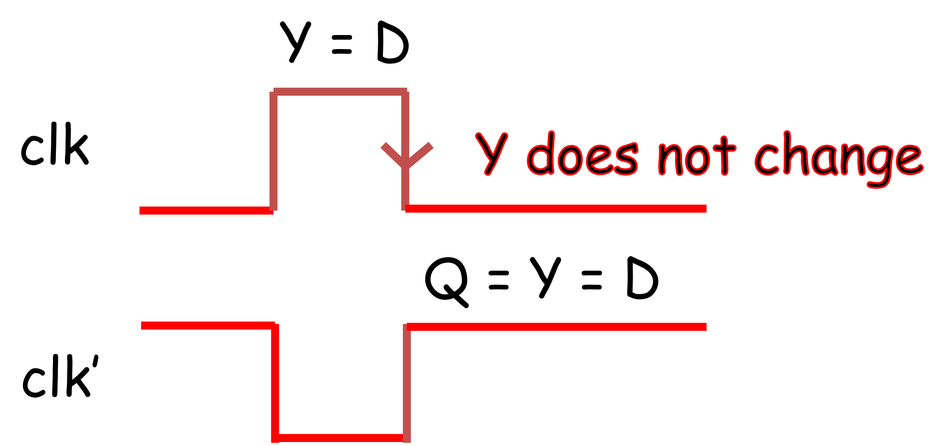
D Latch as a Storage Element

- When C = 1 D latch copies the input to the output.
- When C = 0 the information is kept unchanged.
- Latches are called level triggered.
  - While C is in logic-1 level, the changes at the input cause changes at the output.
- The states of the storage elements should change synchronously.
- We need a storage element which changes the state in a very short time spot.
- These storage elements are called edge triggered and specially flip-flops.

Edge Triggered D Flip-Flop

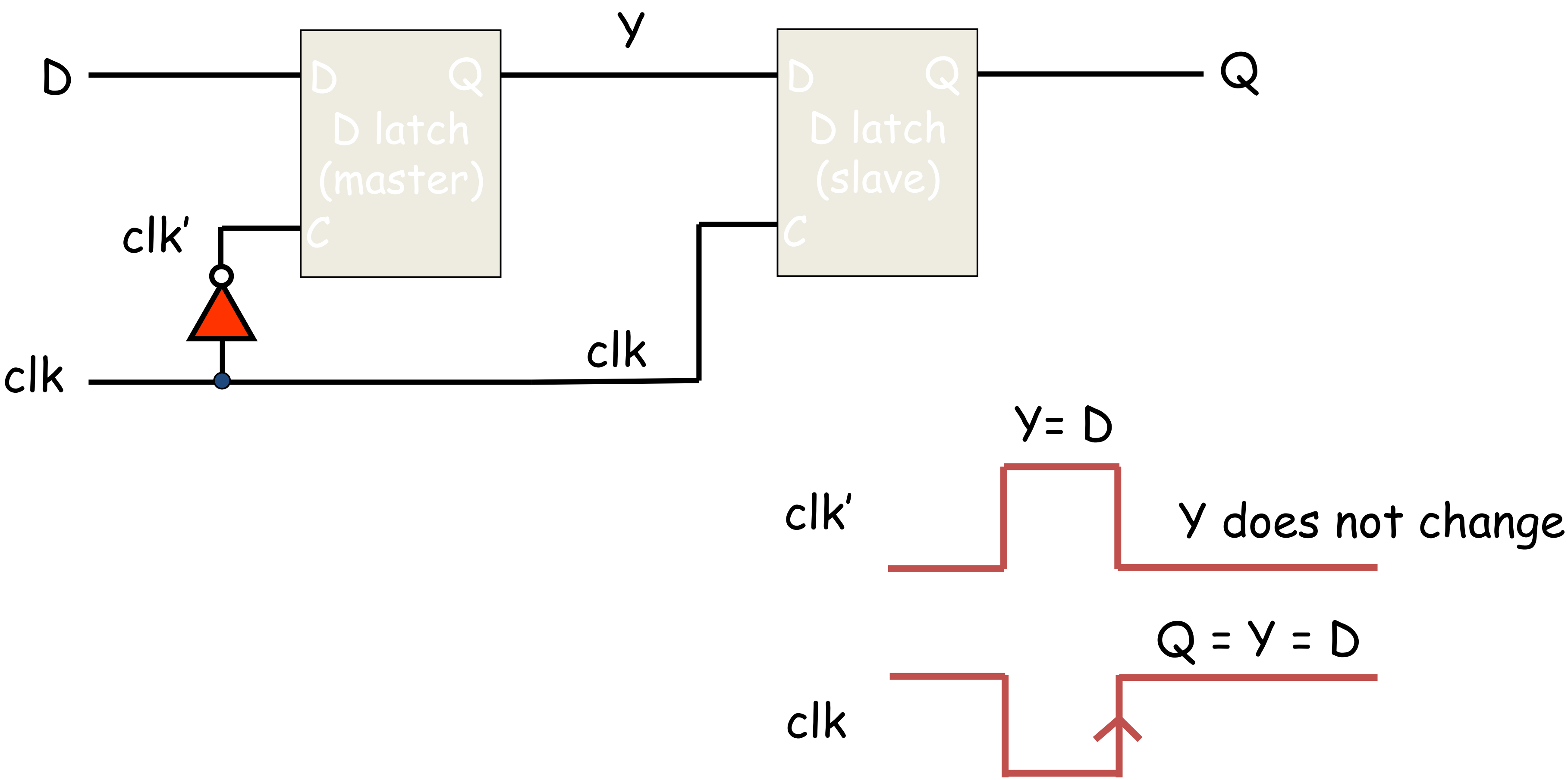
- Edge triggered D flip-flop can be constructed by using two D latches.



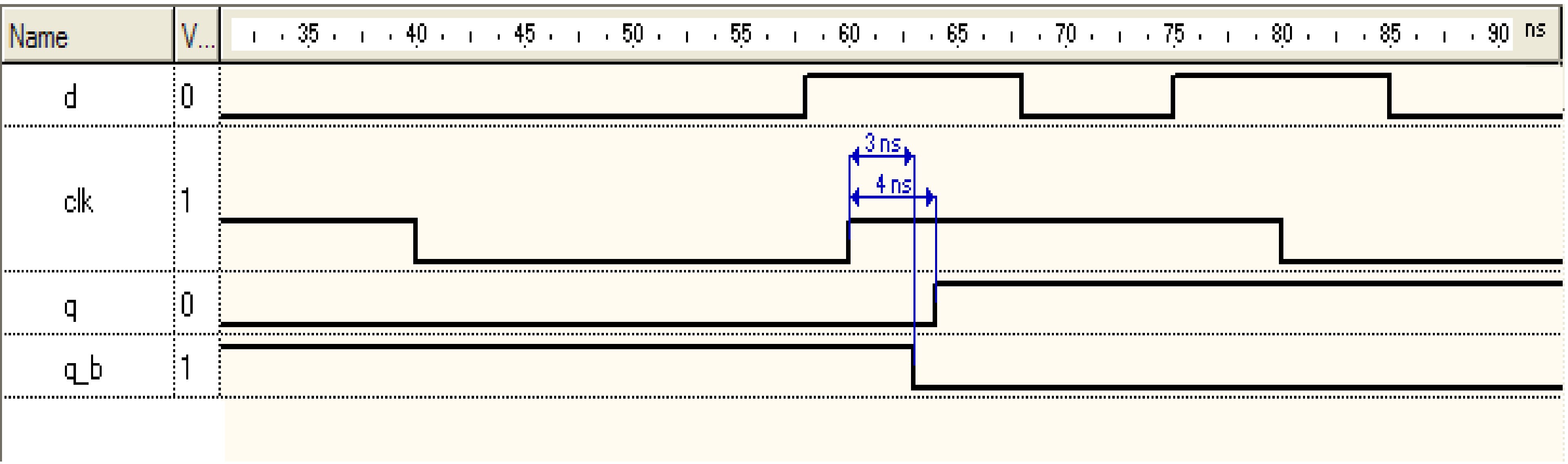


Falling edge triggered
D flip-flop

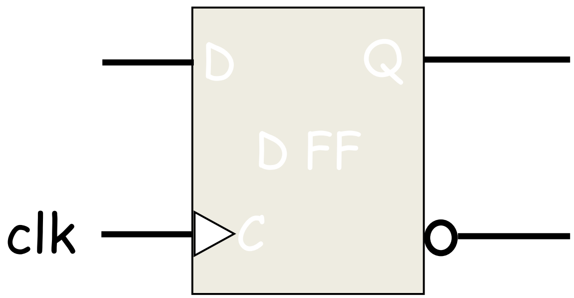
Rising Edge Triggered D Flip-Flop



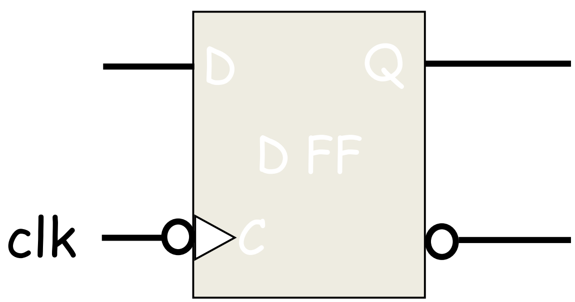
Simulation of Rising Edge Triggered D Flip-Flop



D Flip-Flop Symbols



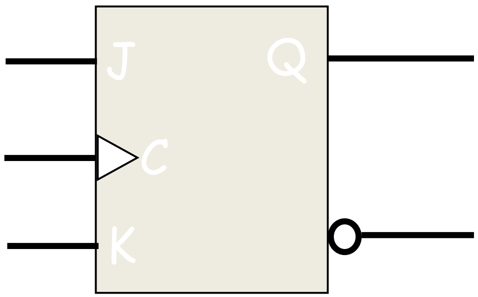
Rising edge triggered
D Flip-Flop



Falling edge triggered
D Flip-Flop

- Characteristic Equation
  - Q(t+1) = D
  - Y=D

JK Flip-Flop



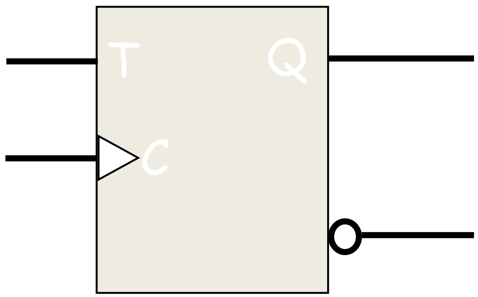
J	K	Q(t+1)	Next State
0	0	Q(t)	No change
0	1	0	Reset
1	0	1	Set
1	1	Q'(t)	Complement

- Characteristic Equation
  - $Q(t+1) = JQ'(t) + K'Q(t)$
  - $Y = Jy' + K'y$

J	K	Y	Next State
0	0	y	No change
0	1	0	Reset
1	0	1	Set
1	1	y'	Complement

Characteristic Table

T (Toggle) Flip-Flop



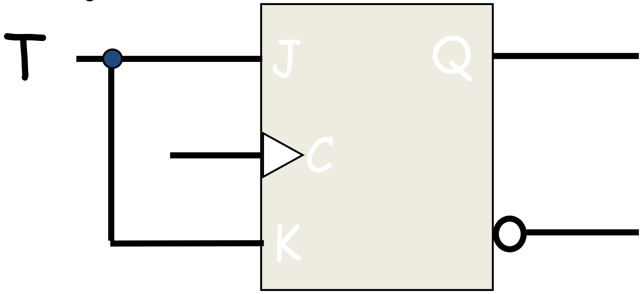
T	Q(t+1)	next state
0	Q(t)	no change
1	Q'(t)	Complement

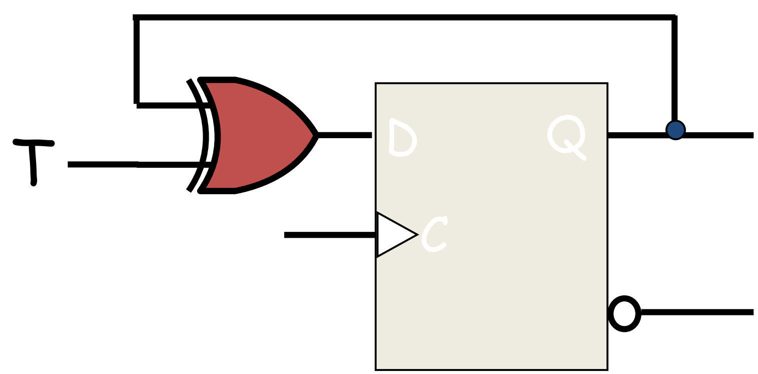
T	y	next state
0	y	no change
1	y'	Complement

Characteristic Equation

Characteristic Table

- $Q(t+1) = T \oplus Q(t) = TQ'(t) + T'Q(t)$
- $Y = T \oplus y = Ty' + T'y$
-



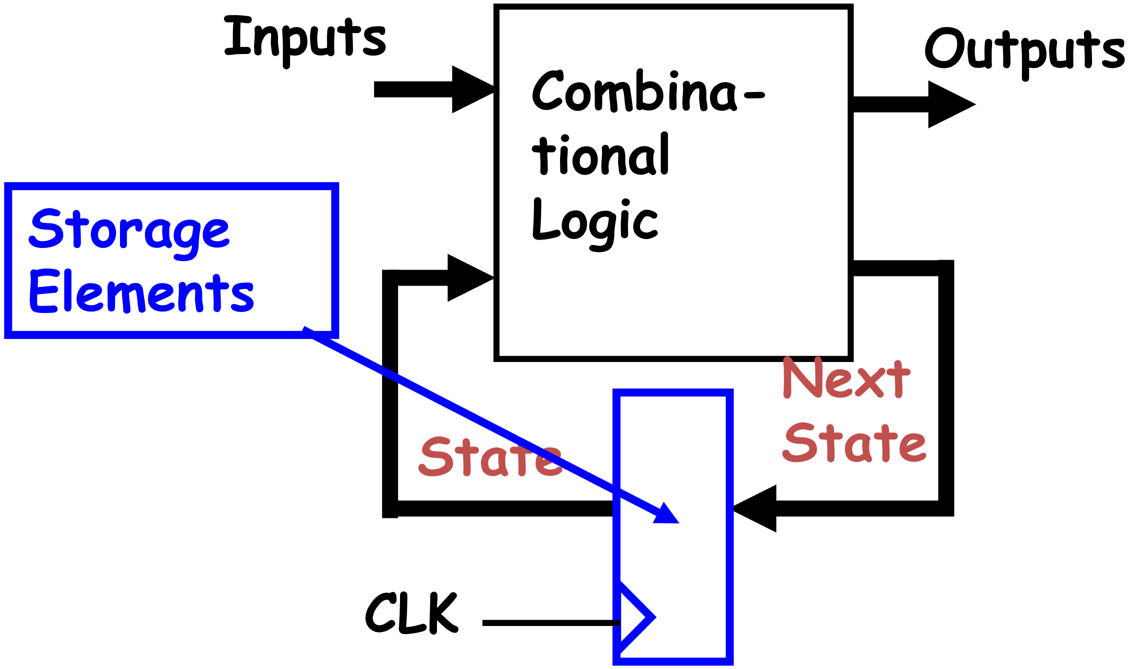


Analysis of Synchronous Sequential Circuits

- Aim:
  - Finding the behaviour of the synchronous sequential circuits
  - “Behaviour”
    - Inputs
    - Outputs
    - States of the flip-flops
  - Finding the Boolean functions of the outputs and the inputs of the flip-flops
      - Output and state equations
    - state table
    - state diagram

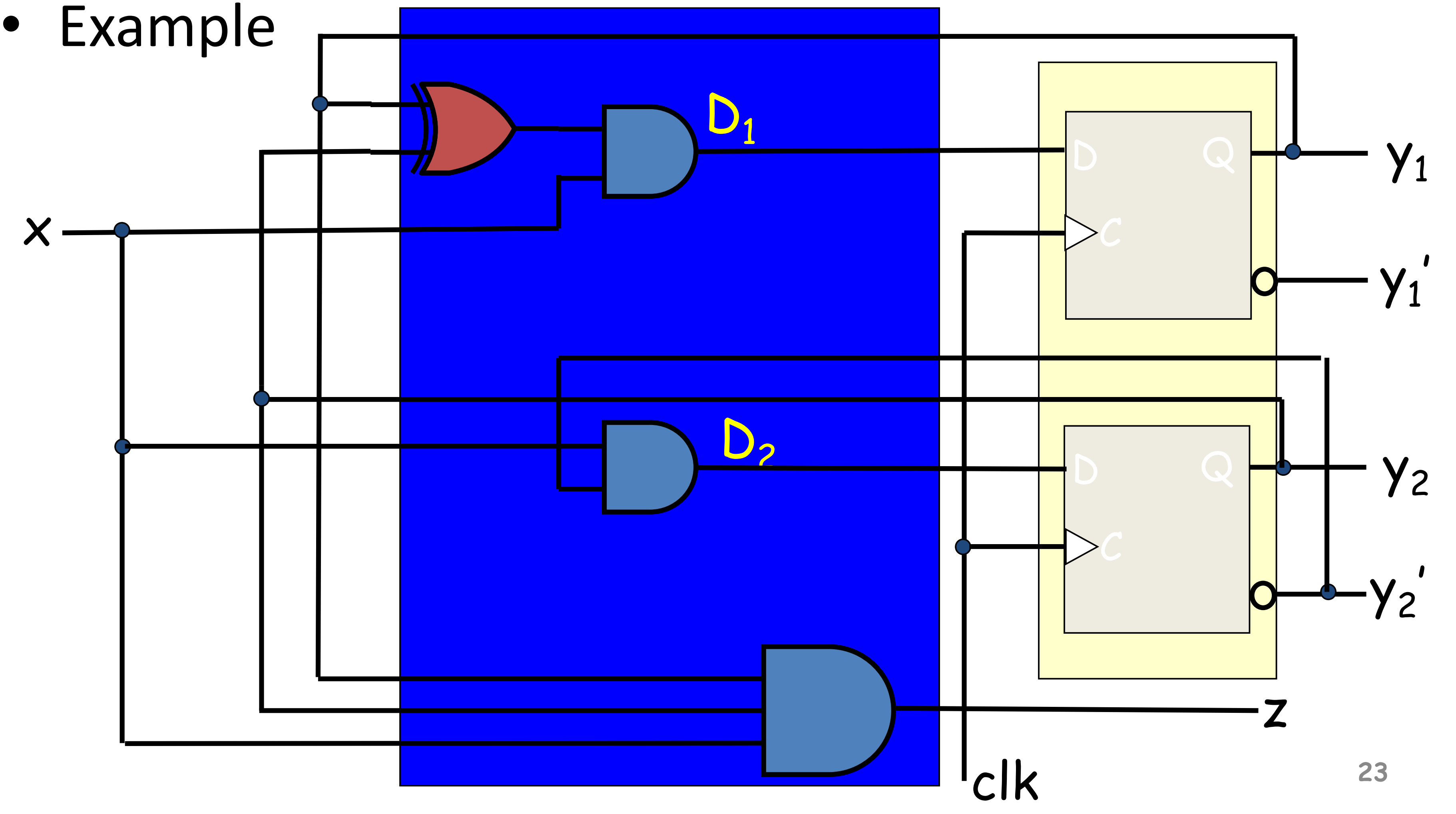
Analysis of Synchronous Sequential Circuits

- Current State at time t is stored in an array of flip-flops.
- Next State at time t+1 is a Boolean function of Current State and Inputs.
- Outputs at time t are a Boolean function of Current State and sometimes Inputs.



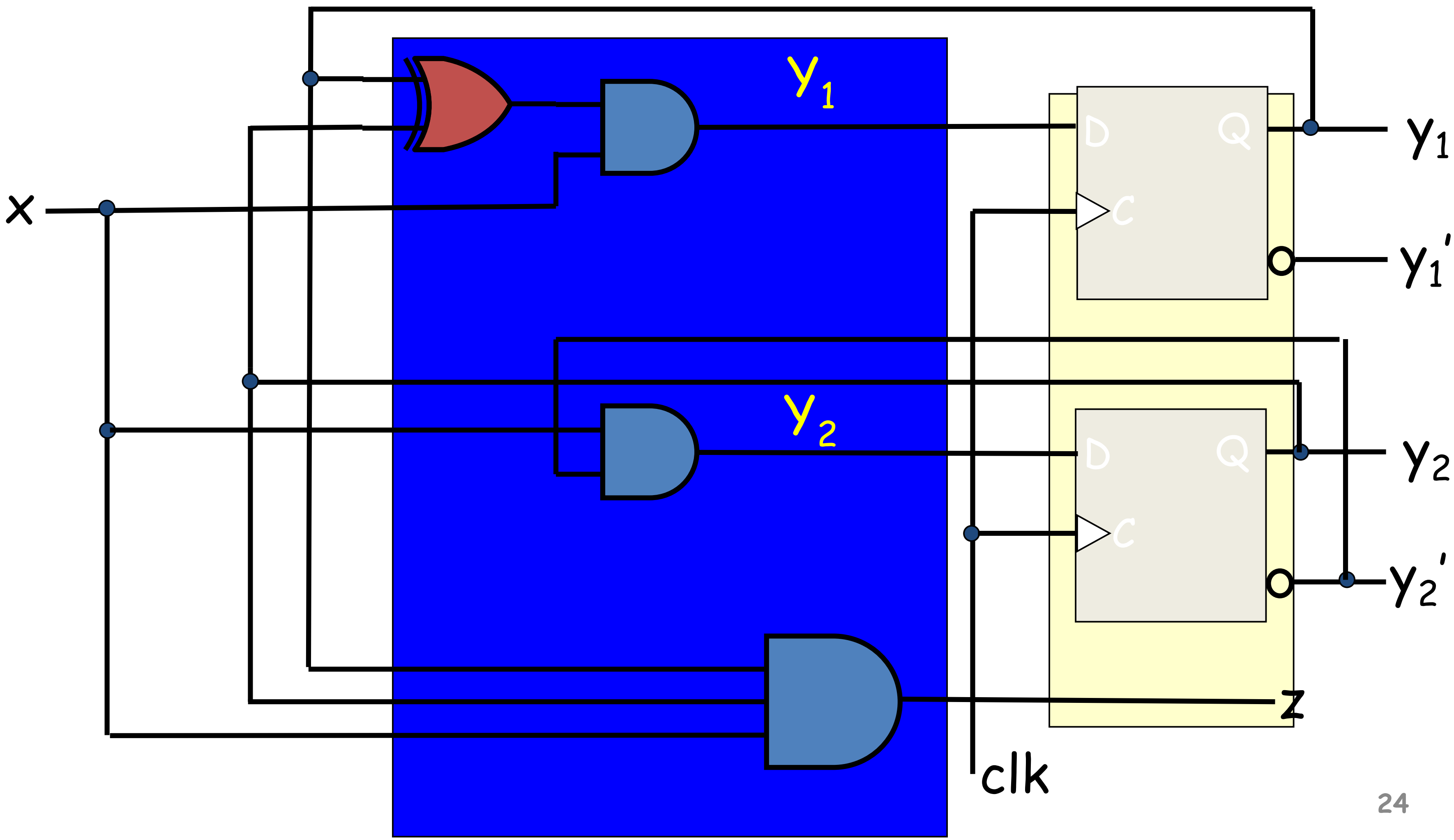
State and Output Equations

- They are also called transition equations.
  - They show the next state as a function of the present state and the inputs.
- Example



State and Output Equations

- $D_1 = (y_1 \oplus y_2)\,x = Y_1$
- $D_2 = x\,y_2' = Y_2$
- $z = y_1\,y_2\,x$



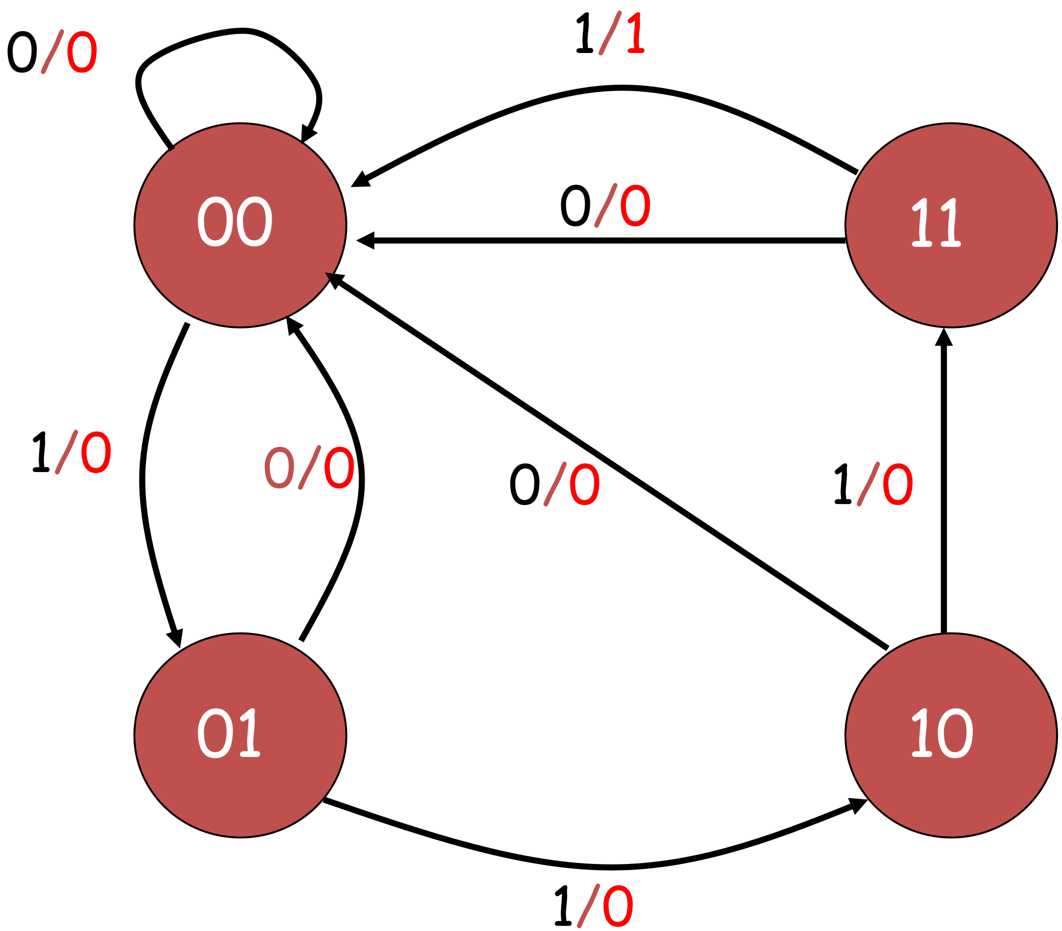
Example: State (Transition) Table

$Y_1 = ?$ $\quad$ $Y_2 = ?$ $\quad$ $z = ?$

Present State y_1	y_2	Input x	Next State Y_1	Y_2	Output z
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	0
1	1	1	0	0	1

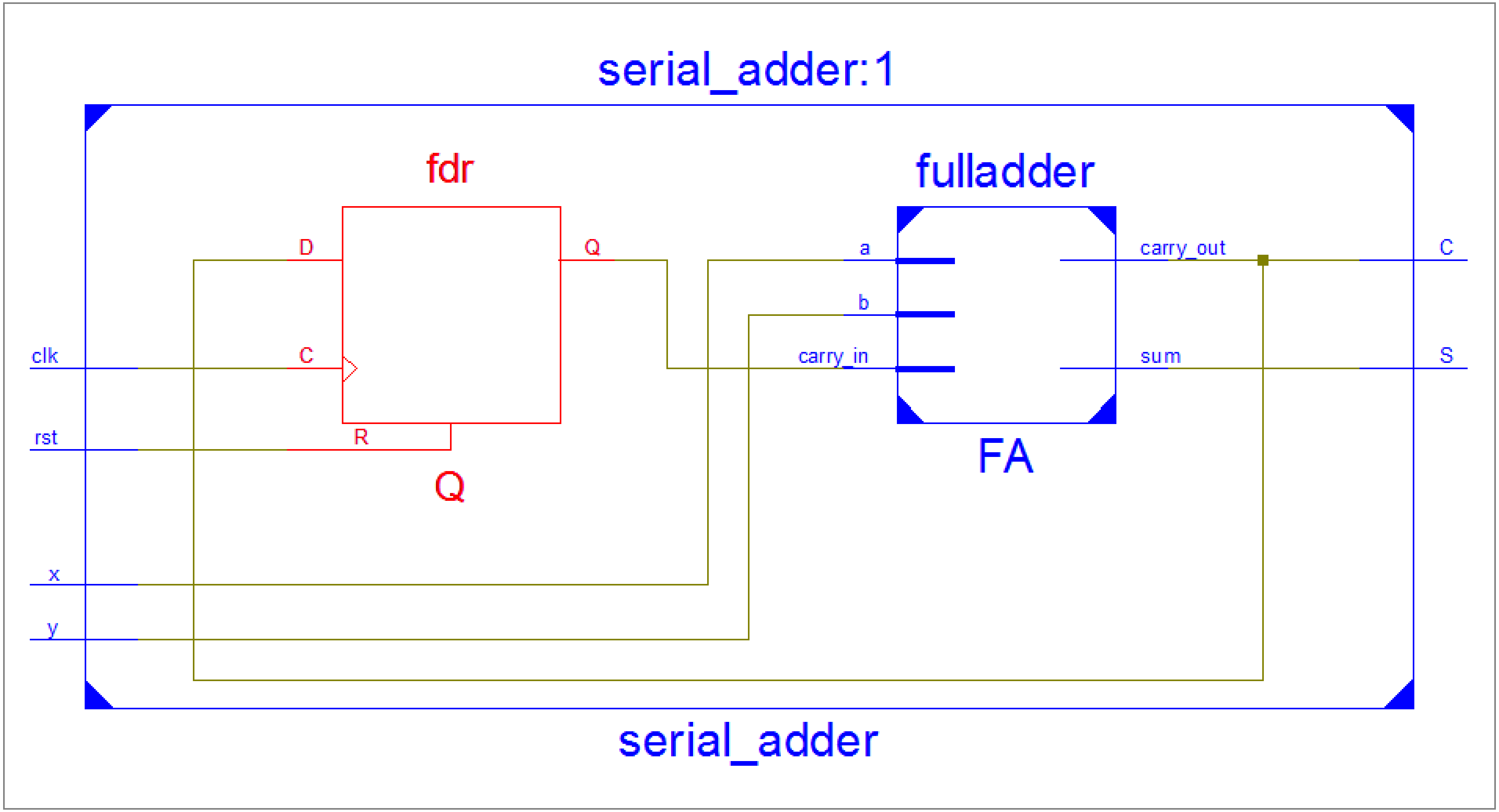
There are 2^{m+n} rows in the state table of a synchronous sequential circuit with m FFs and n inputs.

Example: State (Transition) Diagram



Current State		Input	Next State		Output
y_1	y_2	x	Y_1	Y_2	z
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	0
1	1	1	0	0	1

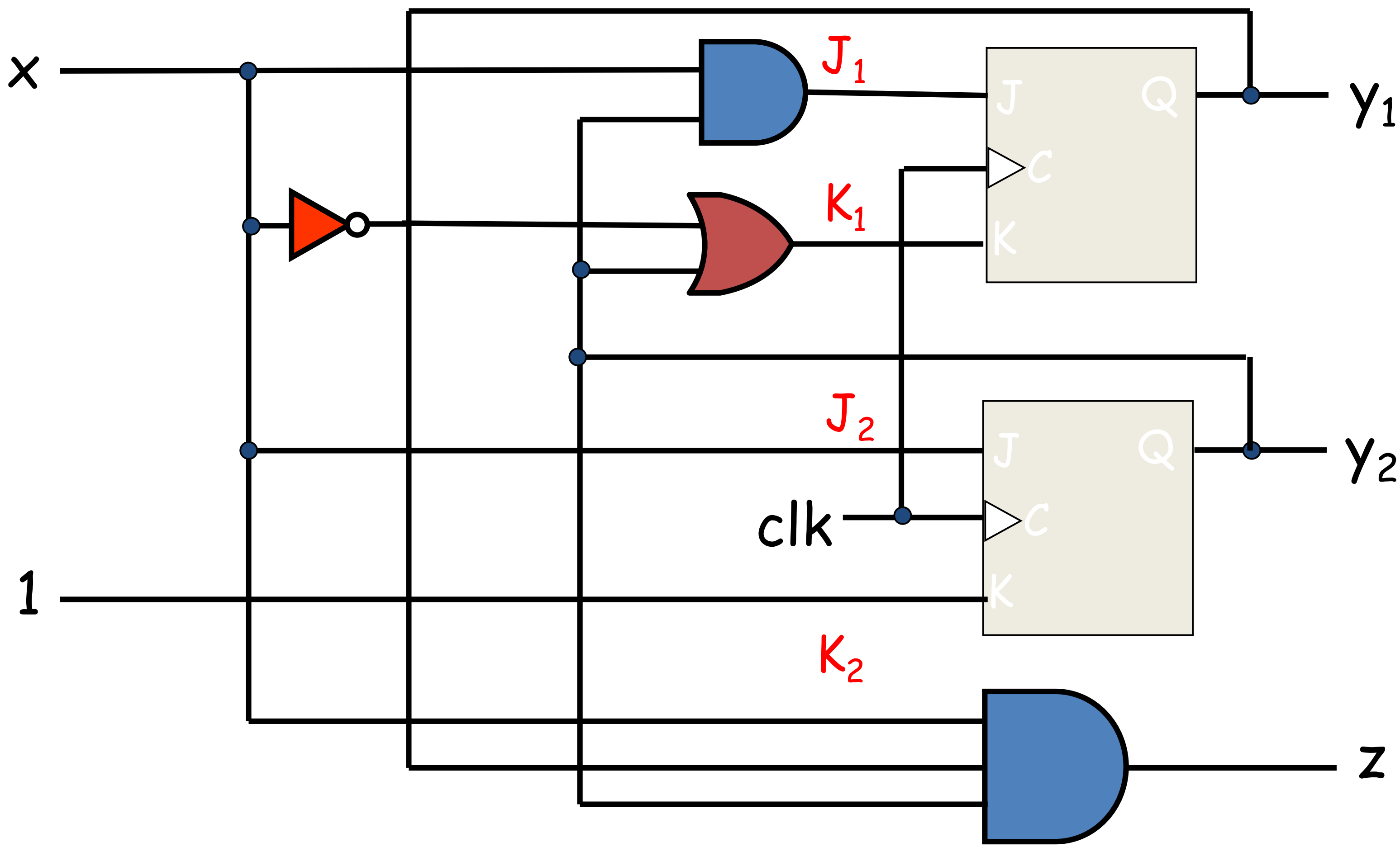
The state diagram and table give the same information


serial_adder:1
fdr
fulladder
D
Q
a
carry_out
C
b
clk
C
carry_in
sum
S
rst
R
FA
Q
x
y
serial_adder

Analysis of a Synchronous Sequential Circuit with JK Flip-Flops

- For a D flip-flop, the state equation is the same as the flip-flop input equation
  - $Q(t+1) = D$
- For JK flip-flops, situation is different
  - Goal is to find state equations
  - Method
    1. determine flip-flop input equations
    2. List the binary values of each input equation
    3. Use the corresponding flip-flop characteristic table to determine the next state values in the state table

Example: Analysis with JK FFs



- Flip-flop input equations
  - $J_1 = xy_2$ and $K_1 = x' + y_2$
  - $J_2 = x$ and $K_2 = 1$

Example: Analysis with JK FFs

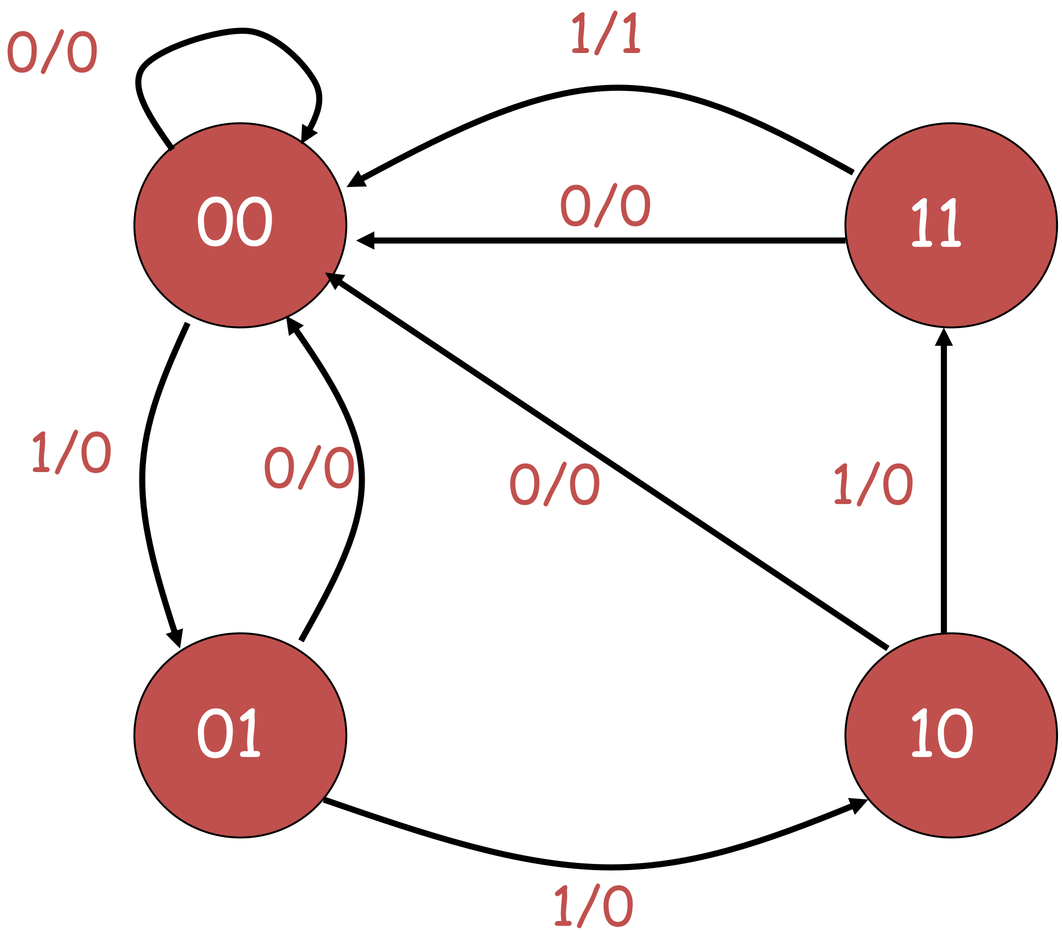
- $J_1 = xy_2$ and $K_1 = x' + y_2$
- $J_2 = x$ and $K_2 = 1$

present State		input	next state		FF inputs			
y_1	y_2	x	Y_1	Y_2	J_1	K_1	J_2	K_2
0	0	0	0	0	0	1	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	0	0	1	0	1
0	1	1	1	0	1	1	1	1
1	0	0	0	0	0	1	0	1
1	0	1	1	1	0	0	1	1
1	1	0	0	0	0	1	0	1
1	1	1	0	0	1	1	1	1

Example: Analysis with JK FFs

- Characteristic equations
  - $Y_1 = J_1y_1' + K_1'y_1$
  - $Y_2 = J_2y_2' + K_2'y_2$
- Flip-flop Input equations
  - $J_1 = xy_2$ ve $K_1 = x' + y_2$
  - $J_2 = x$ ve $K_2 = 1$
- State equations
  - $Y_1 = xy_2y_1' + (x' + y_2)'y_1 = xy_2y_1' + xy_2'y_1 = x(y_2 \oplus y_1)$
  - $Y_2 = xy_2' + 1'y_2 = xy_2'$

State Diagram



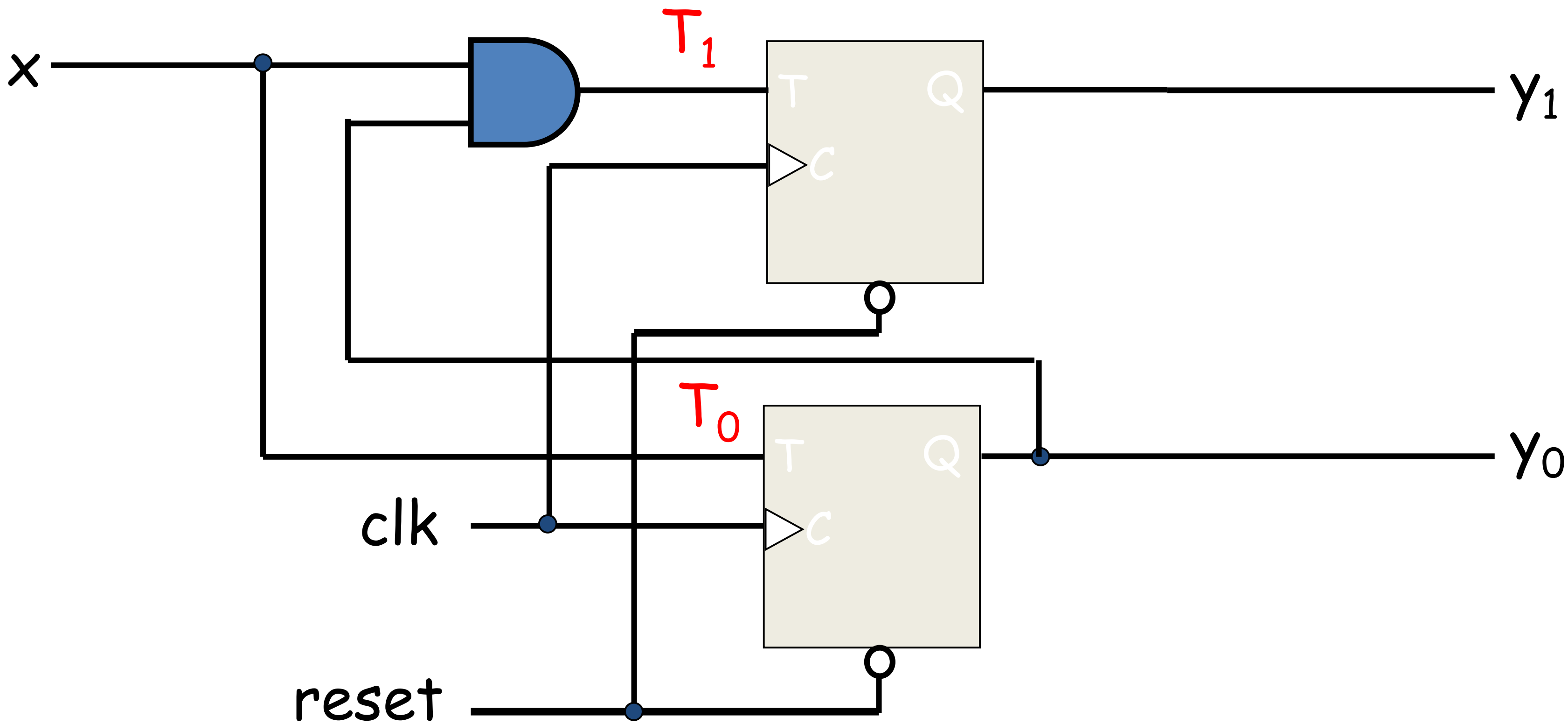
Present state		Input	Next State		Output
y_1	y_2	x	Y_1	Y_2	z
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	0
1	1	1	0	0	1

What is the circuit doing?

Analysis with T Flip-Flops

- Method is the same
- Example

$T_1 = xy_0$
$T_0 = x$



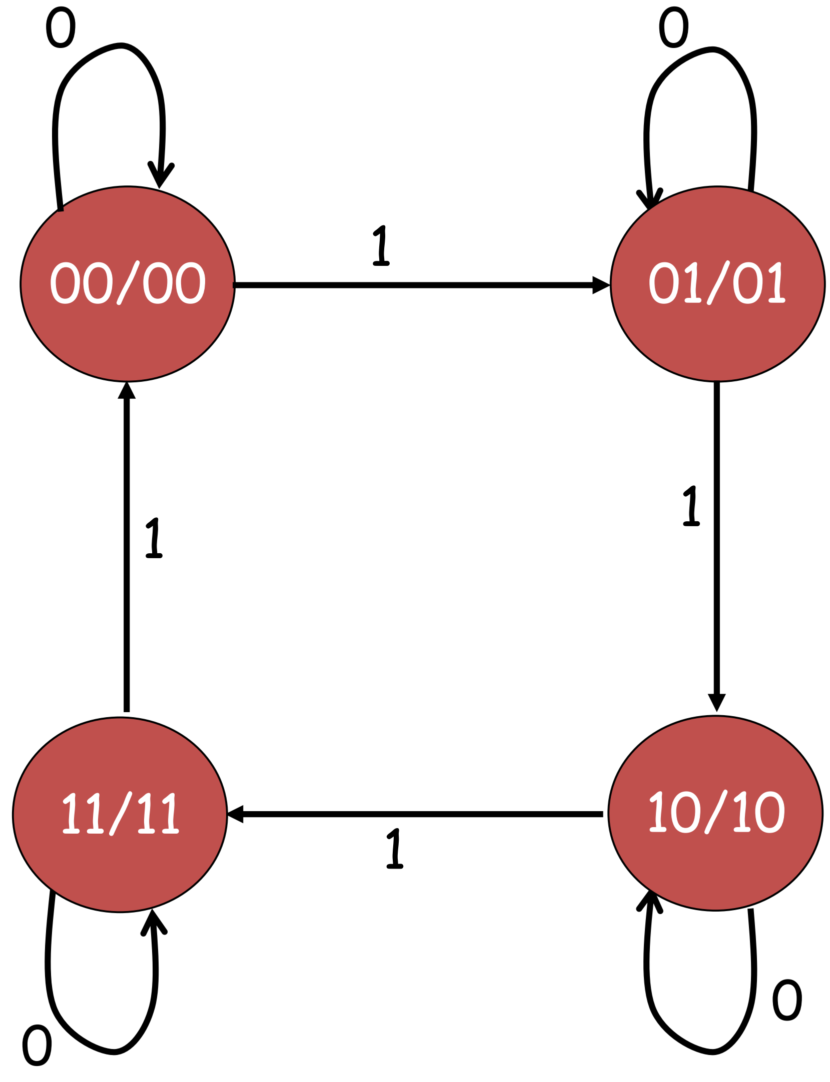
Example: Analysis with T Flip-Flops

- Characteristic equation
  - $Y_0 = T_0 \oplus y_0$
  - $Y_1 = T_1 \oplus y_1$
- Flip-flop Input equations
  - $T_1 = x\, y_0$
  - $T_0 = x$
- State equations
  - $Y_0 = x \oplus y_0$
  - $Y_1 = x\, y_0 \oplus y_1$

State Table & Diagram

- $Y_0 = x \oplus y_0$
- $Y_1 = x\, y_0 \oplus y_1$

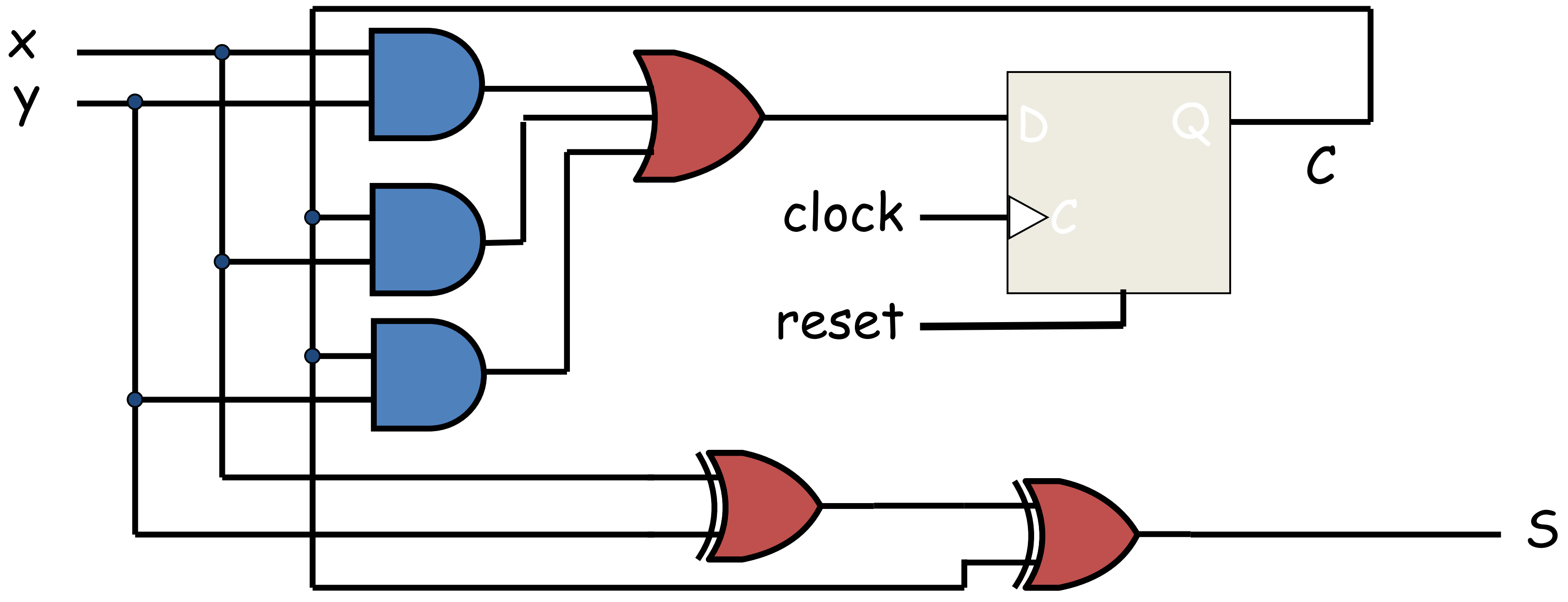
Present State y_1	Present State y_0	Input x	Next State Y_1	Next State Y_0	Output y_1	Output y_0
0	0	0	0	0	0	0
0	0	1	0	1	0	0
0	1	0	0	1	0	1
0	1	1	1	0	0	1
1	0	0	1	0	1	0
1	0	1	1	1	1	0
1	1	0	1	1	1	1
1	1	1	0	0	1	1



Mealy and Moore Models

- There are two models for sequential circuits
  - Mealy
  - Moore
- They differ in the way the outputs are generated
  - Mealy:
    - output is a function of both present states and inputs
  - Moore
    - output is a function of present state only

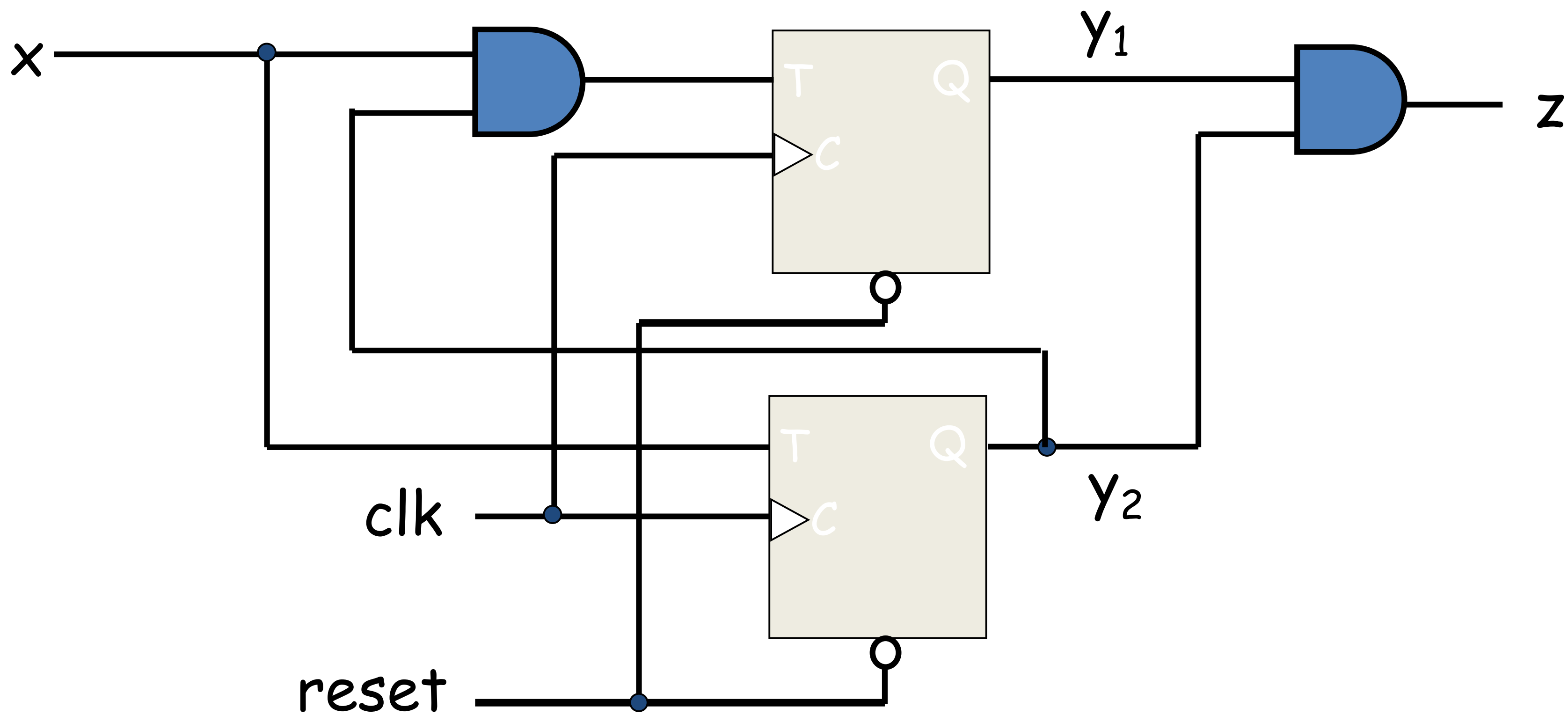
Example: Mealy and Moore Machines



Mealy Machine

- External inputs, x and y, are asynchronous
- Thus, outputs may have momentary (incorrect) values
- Inputs must be synchronized with clocks
- Outputs must be sampled only during clock edges.

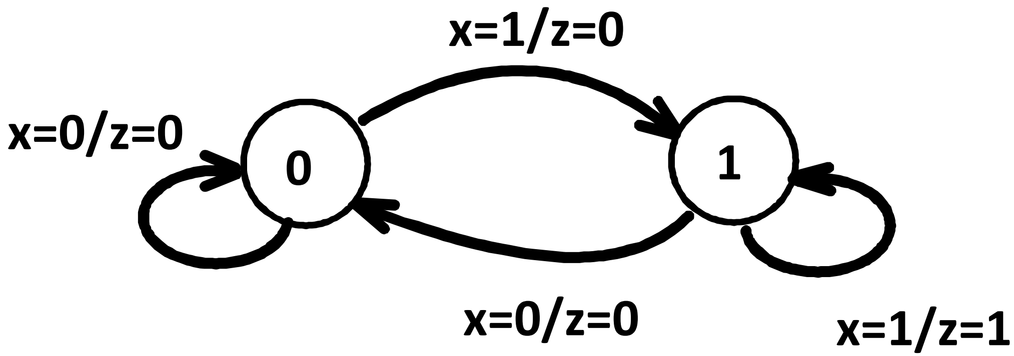
Example: Moore Machines



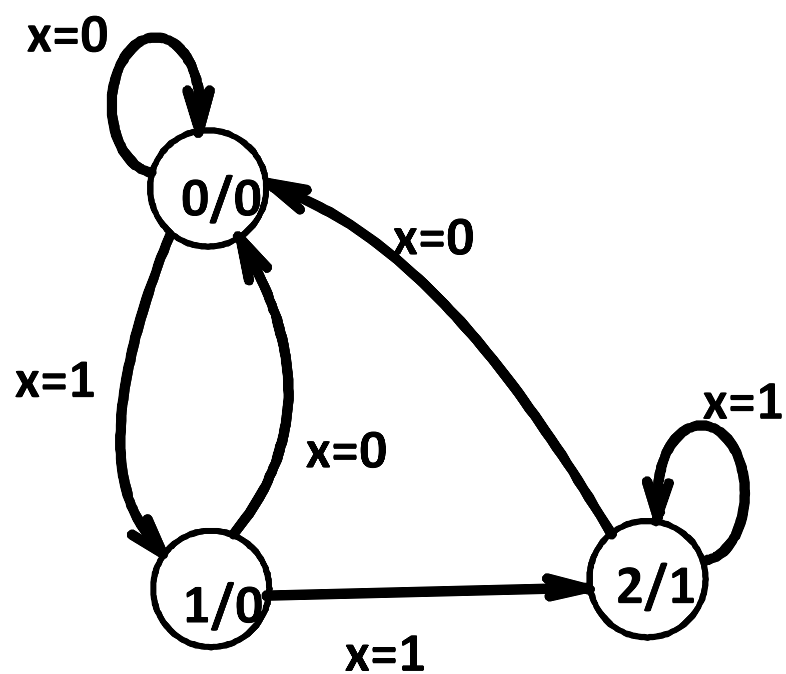
- Outputs are already synchronized with clock.
- They change synchronously with the clock edge.

Example State Diagrams for Moore and Mealy Machines

- State Diagram for Mealy Model



- State Diagram for Moore Model



Design Process

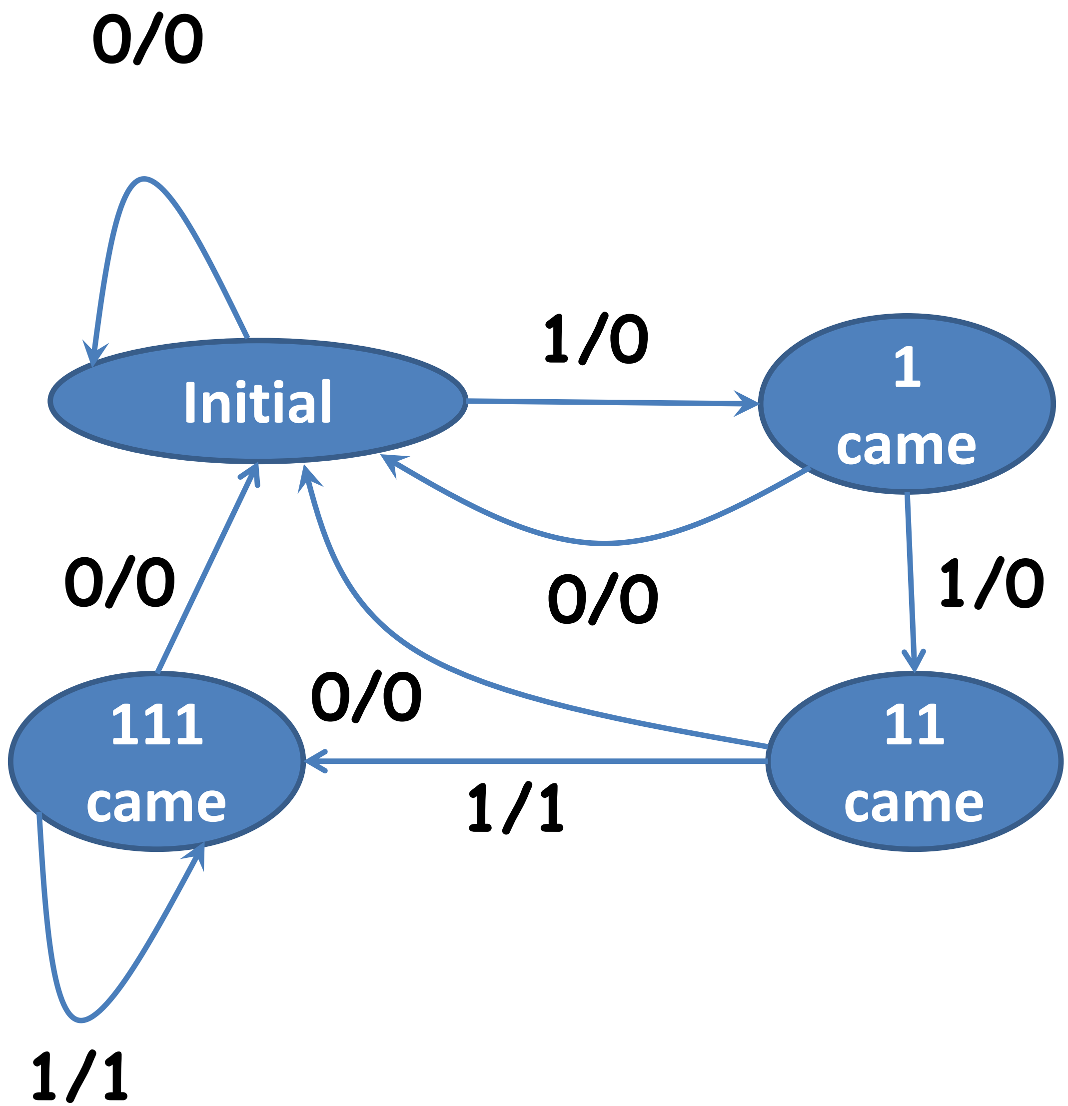
1. Verbal description of desired operation
2. Draw the state diagram
3. Reduce the number of states if necessary and possible: s = number of states
4. Determine the number of flip-flops: $n = \lceil \log_2 s \rceil$
5. State assignment: $\underbrace{00\ldots0}_{n-bits}, \underbrace{00\ldots1}_{n-bits}, \underbrace{00\ldots10}_{n-bits}, \ldots$
6. Obtaine the encoded state table
7. Choose the type of the flip-flops
8. Derive the simplified flip-flop input equations
9. Derive the simplified output equations
10. Draw the logic diagram

Example: Design of a Synchronous Sequential Circuit

- Verbal description
  - 1st Step: we want a 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 such a pattern in the string
      - "0" otherwise

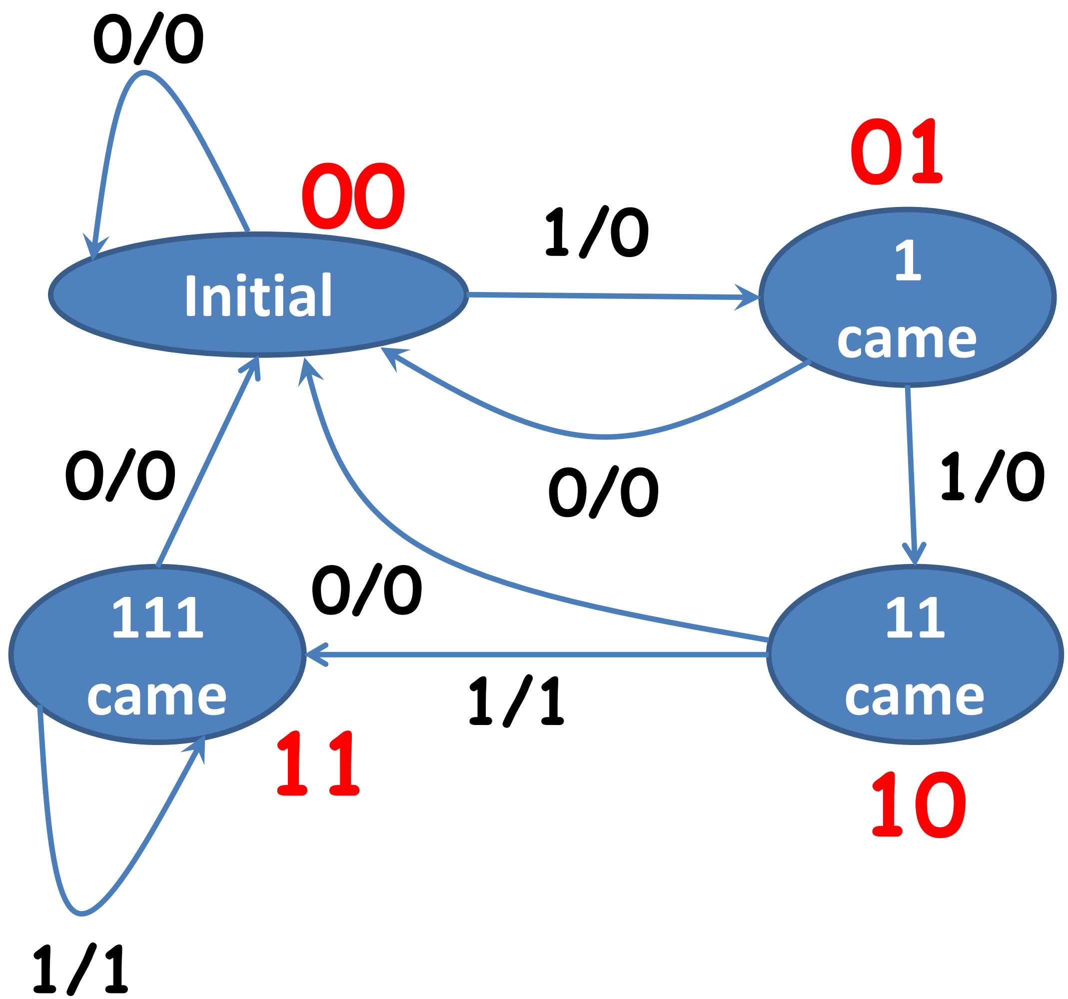
Example: State Diagram

2nd Step: **Draw the state diagram**



Synthesis with D Flip-Flops 1/5

- 3rd Step: State reduction
  - Not possible
- 4th Step: Number of flip-flops
  - 4 states
  - ? flip-flop
- 5th Step: State assignment



Synthesis with D Flip-Flops 2/5

- 6th Step: Obtain the state table

Present State		Input	Next State		Output
y_1	y_2	x	Y_1	Y_2	z
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	1
1	1	0	0	0	0
1	1	1	1	1	1

Synthesis with D Flip-Flops 3/5

- 7th Step: Choose the type of the flip-flops
  - D type flip-flops
- 8th Step: : Derive the simplified flip-flop input equations
  - Boolean expressions for D_1 and D_2

$y_1 \backslash y_2x$	00	01	11	10
0	0	0	1	0
1	0	1	1	0

$$D_1 = y_1x + y_2x$$

$y_1 \backslash y_2x$	00	01	11	10
0	0	1	0	0
1	0	1	1	0

$$D_2 = y_1x + y_2'x$$

Synthesis with D Flip-Flops 4/5

- 9th Step: : Derive the simplified output equations
  - Boolean expressions for z

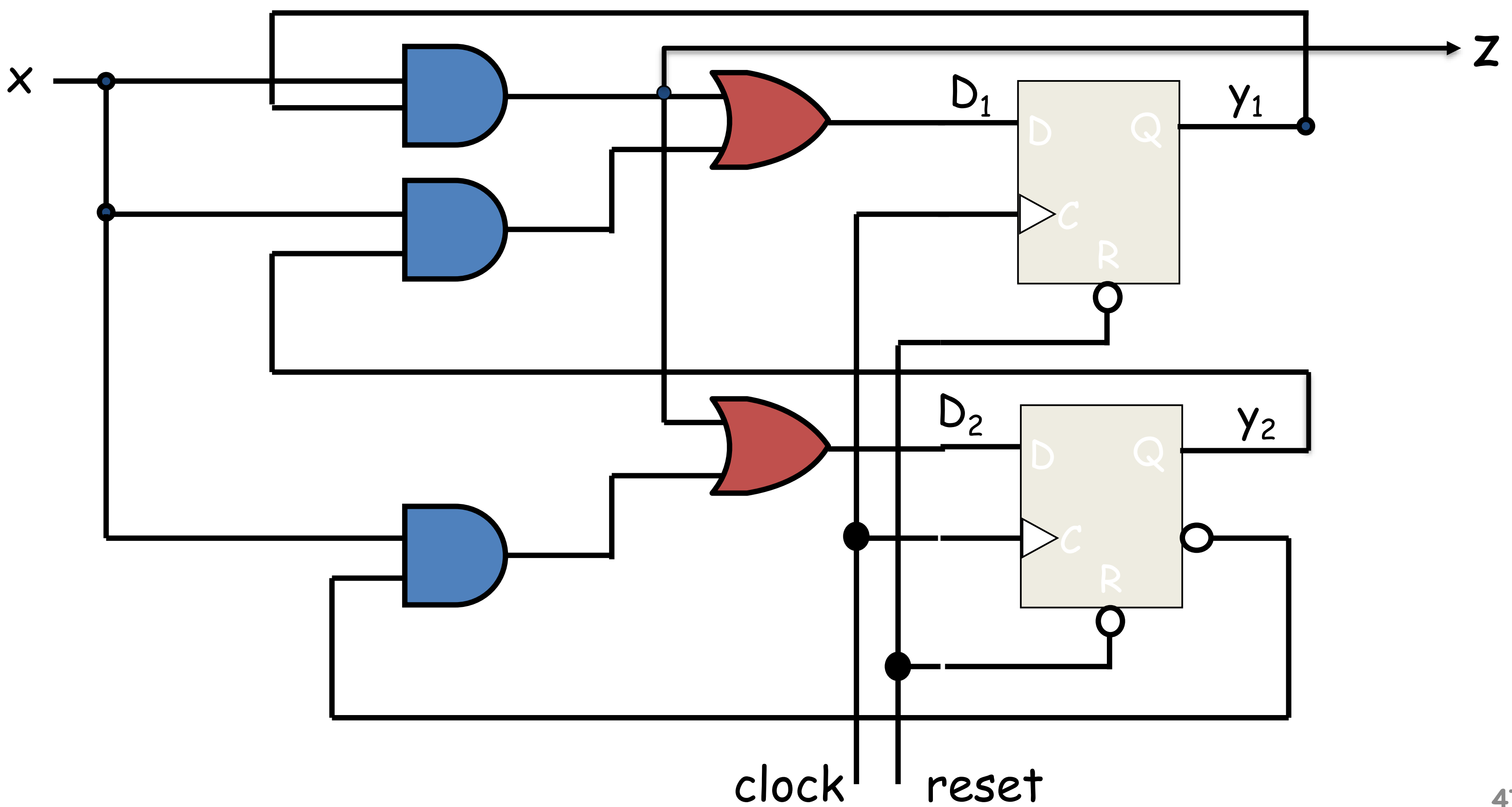
$y_1 \backslash y_2x$	00	01	11	10
0	0	0	0	0
1	0	1	1	0

$z = y_1x$

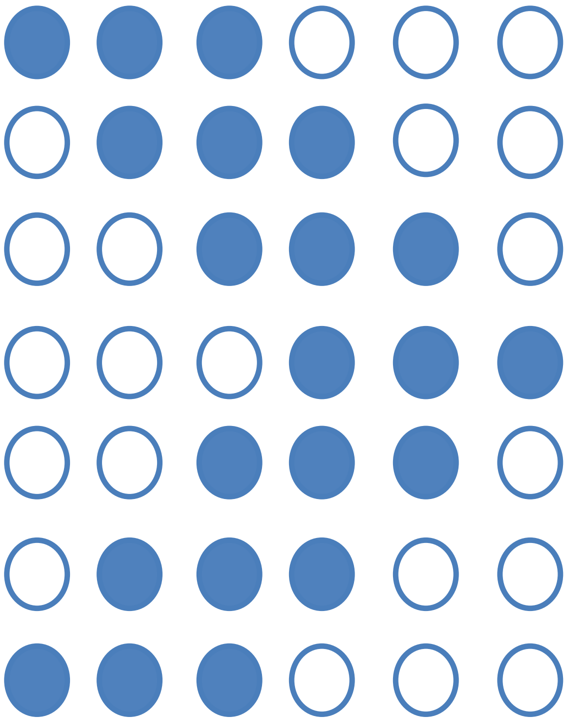
Synthesis with D Flip-Flops 5/5

- 10th Step: Draw the logic diagram

$D_1 = y_1x + y_2x$ $D_2 = y_1x + y_2'x$ $z = y_1x$



Synthesis with JK Flip-Flops and MUXs



Number of states= 6

Number of state variables= 3

Number of flip-flops= 3

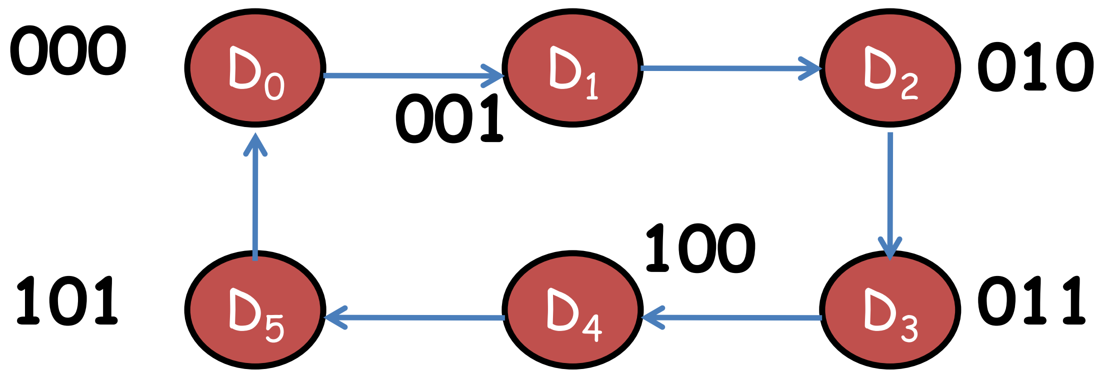
Number of Inputs= 0

Number of Outputs= 6

- 6 shifting lights
- ●= lojik-1
- O= lojik-0

$y_2 = J_2 y_2' + K_2' y_2$

State Diagram & Table



$$Y = Jy' + K'y$$

J	K	Y
0	0	y
0	1	0
1	0	1
1	1	Q'

Present State			Next State			Flip-flop inputs						Outputs					
y_2	y_1	y_0	Y_2	Y_1	Y_0	J_2	K_2	J_1	K_1	J_0	K_0	z_5	z_4	z_3	z_2	z_1	z_0
0	0	0	0	0	1	0	k	0	k	1	k	1	1	1	0	0	0
0	0	1	0	1	0	0	k	1	k	k	1	0	1	1	1	0	0
0	1	0	0	1	1	0	k	k	0	1	k	0	0	1	1	1	0
0	1	1	1	0	0	1	k	k	1	k	1	0	0	0	1	1	1
1	0	0	1	0	1	k	0	0	k	1	k	0	0	1	1	1	0
1	0	1	0	0	0	k	1	0	k	k	1	0	1	1	1	0	0

Inplementation of Flip-Flop Input Equations

y_2 \ y_1y_0	00	01	11	10
0	0	0	0	1
1	k	k	k	k

$J_2 = y_1y_0'$

y_2 \ y_1y_0	00	01	11	10
0	k	k	k	k
1	0	1	k	k

$K_2 = y_0$

y_2 \ y_1y_0	00	01	11	10
0	0	1	k	k
1	0	0	k	k

$J_1 = y_2'y_0$

y_2 \ y_1y_0	00	01	11	10
0	k	k	1	0
1	k	k	k	k

$K_1 = y_0$

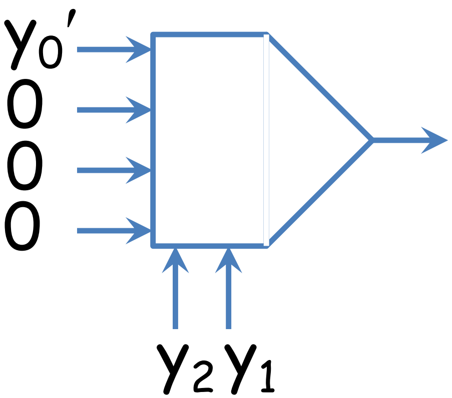
y_2 \ y_1y_0	00	01	11	10
0	1	k	k	1
1	1	k	k	k

$J_0 = 1$

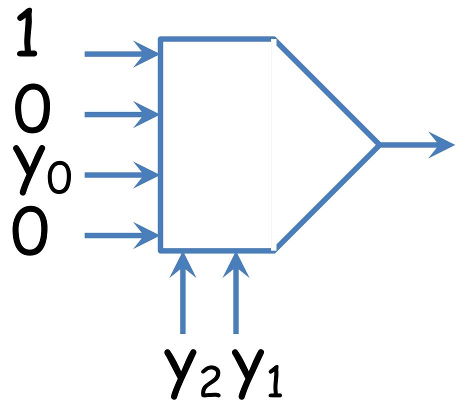
y_2 \ y_1y_0	00	01	11	10
0	k	1	1	k
1	k	1	k	k

$K_0 = 1$

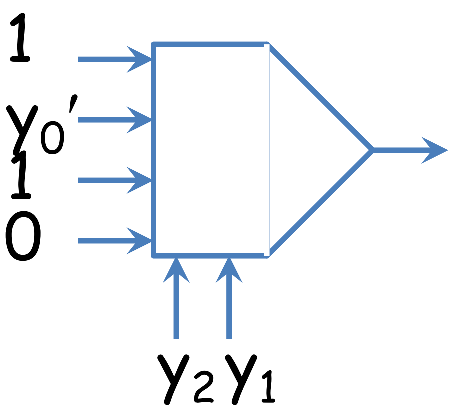
Inplementation of Output Equations



$z_5=y_2'y_1'y_0'+k(y_2y_1y_0'+y_2y_1y_0)$

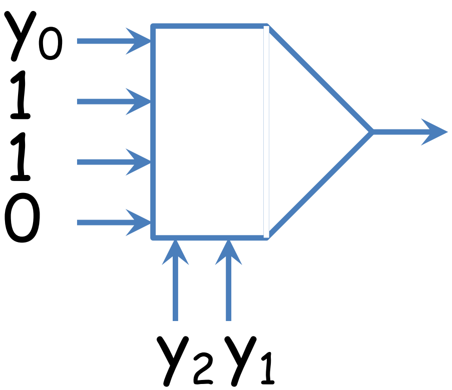


$z_4=y_2'y_1'y_0'+y_2'y_1'y_0+y_2y_1'y_0+k(y_2y_1y_0'+y_2y_1y_0)$

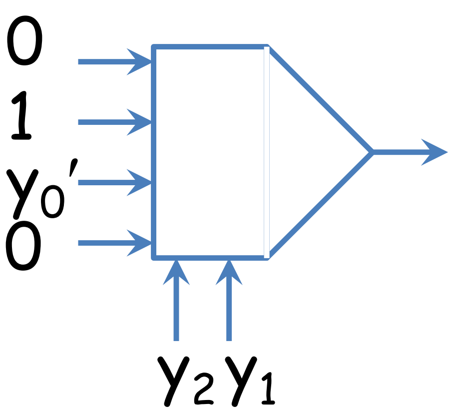


$z_3=y_2'y_1'y_0'+y_2'y_1'y_0+y_2'y_1y_0'+y_2y_1'y_0'+y_2y_1'y_0+k(y_2y_1y_0'+y_2y_1y_0)$

Inplementation of Output Equations

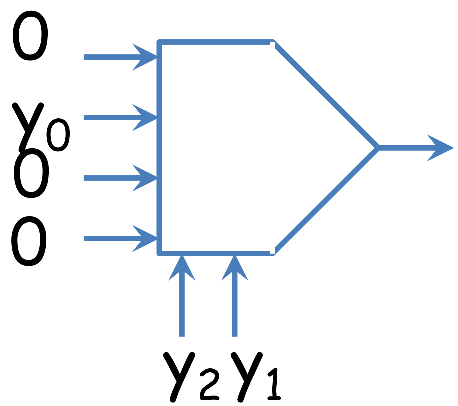


$z_2=y_2'y_1'y_0+y_2'y_1y_0'+y_2'y_1y_0+y_2y_1'y_0'+y_2y_1'y_0+k(y_2y_1y_0'+y_2y_1y_0)$



$z_1=y_2'y_1y_0'+y_2'y_1y_0+y_2y_1'y_0'+k(y_2y_1y_0'+y_2y_1y_0)$

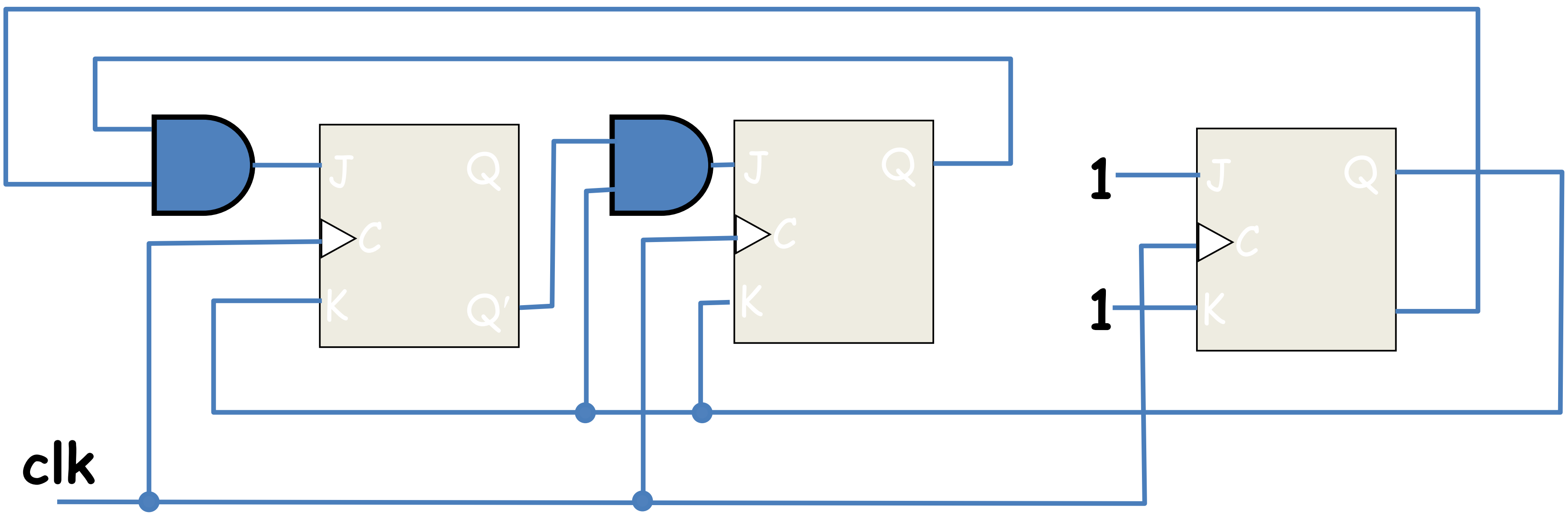
Inplementation of Output Equations



$z_0 = y_2'y_1y_0 + k(y_2y_1y_0' + y_2y_1y_0)$

Logic Diagram

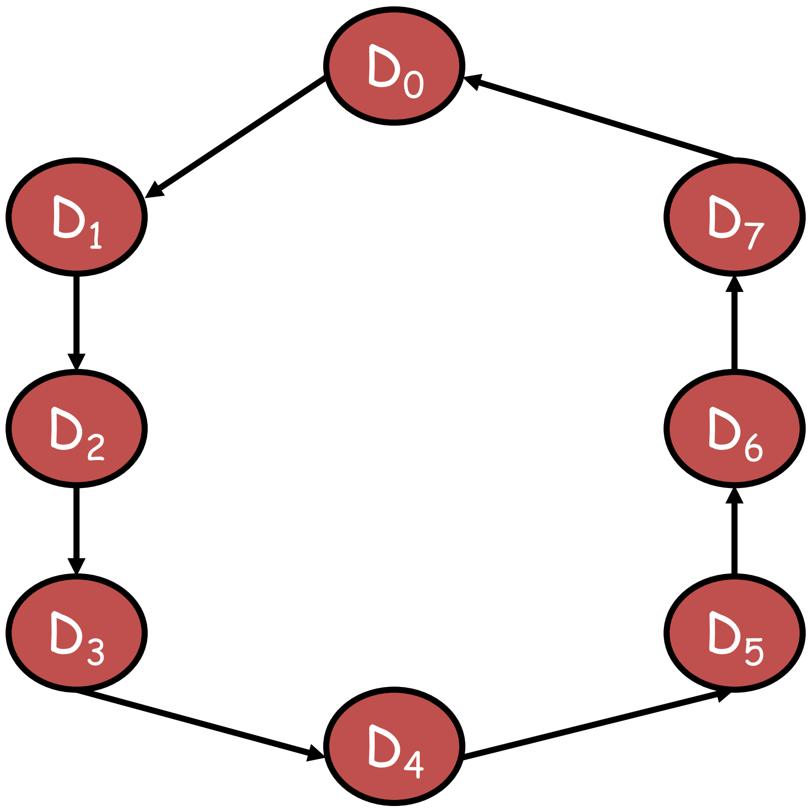
$J_2 = y_1y_0'$ $K_2 = y_0$ $J_1 = y_2'y_0$ $K_1 = y_0$ $J_0 = 1$ $K_1 = 1$



Synthesis with T Flip-Flops 1/4

- Example: 3-bit binary counter

0→1→2→ … → 7 → 0 → 1 → 2



State Diagram

How many flip-flops?

State assignments

- D_0 → 000
- D_1 → 001
- D_2 → 010
- . . .
- D_7 → 111

Synthesis with T Flip-Flops 2/4

- State Table

present state			next state			FF inputs		
y_2	y_1	y_0	Y_2	Y_1	Y_0	T_2	T_1	T_0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

Synthesis with T Flip-Flops 3/4

- Flip-Flop input equations

$y_2 \backslash y_1 y_0$	00	01	11	10
0	0	0	1	0
1	0	0	1	0

$T_2 = y_1 y_0$

$y_2 \backslash y_1 y_0$	00	01	11	10
0	0	1	1	0
1	0	1	1	0

$T_1 = y_0$

$T_0 = 1$

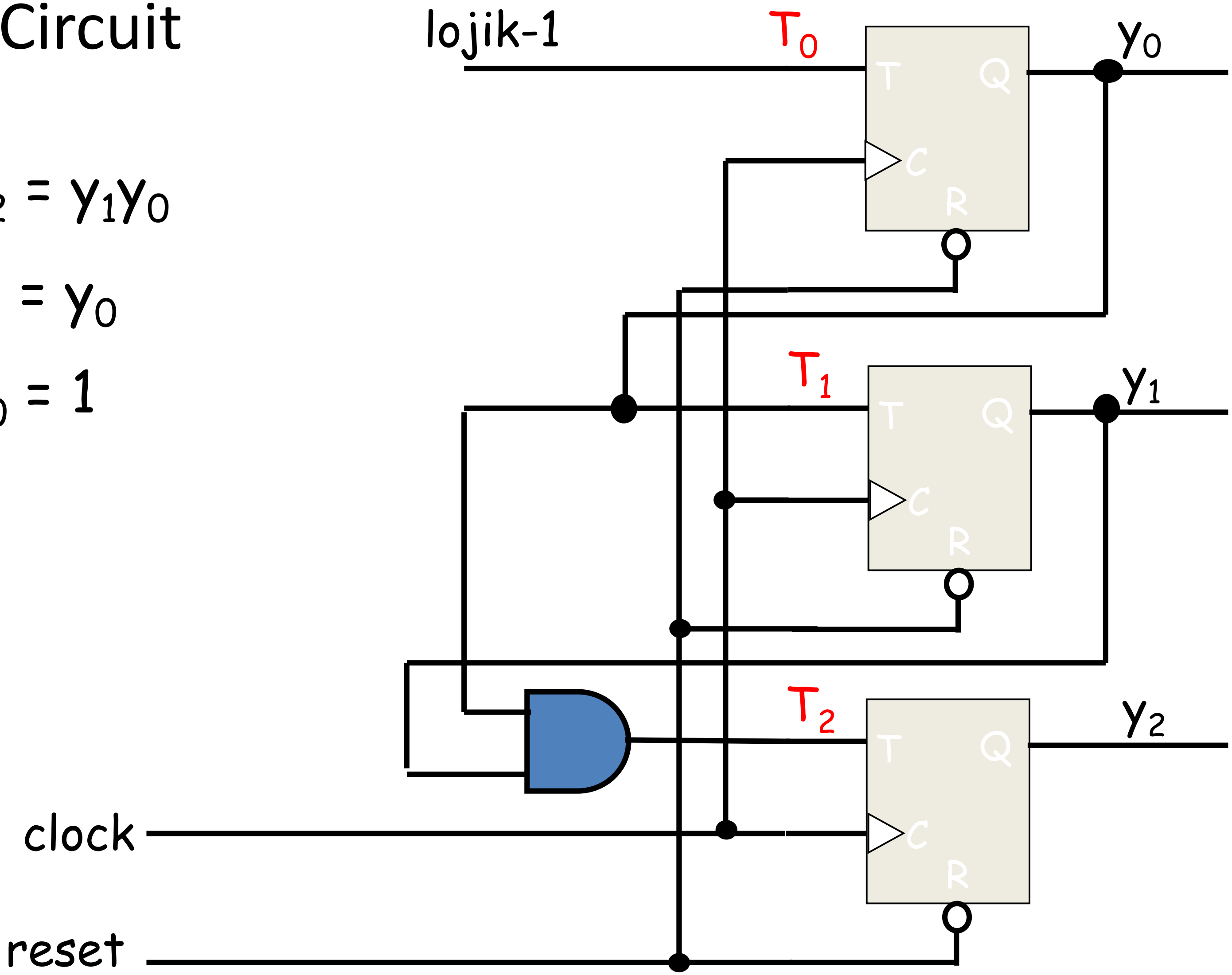
Synthesis with T Flip-Flops 4/4

- Circuit

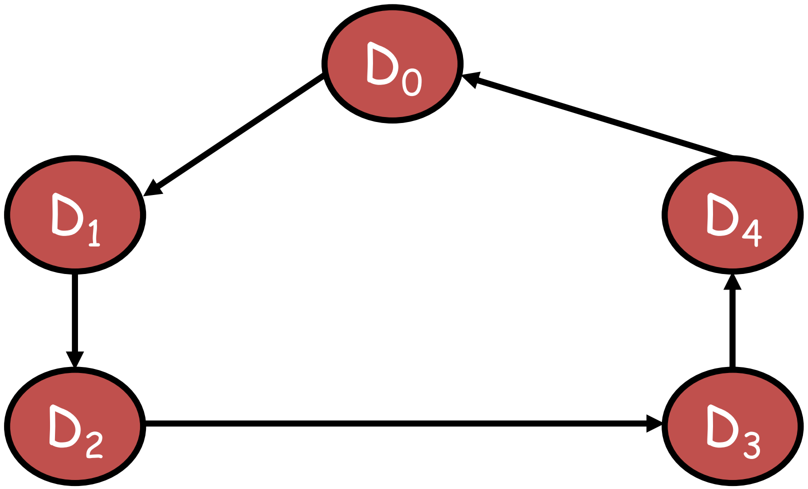
$T_2 = y_1y_0$

$T_1 = y_0$

$T_0 = 1$



Unused States



Modulo-5 counter

Present State			Next State		
y_2	y_1	y_0	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	0	0	0

Example: Unused States 1/4

Present State			Next State		
y_2	y_1	y_0	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	0	0	0

y_2 \ y_1y_0	00	01	11	10
0	0	0	1	0
1	0	X	X	X

$$Y_2 = y_1y_0$$

y_2 \ y_1y_0	00	01	11	10
0	0	1	0	1
1	0	X	X	X

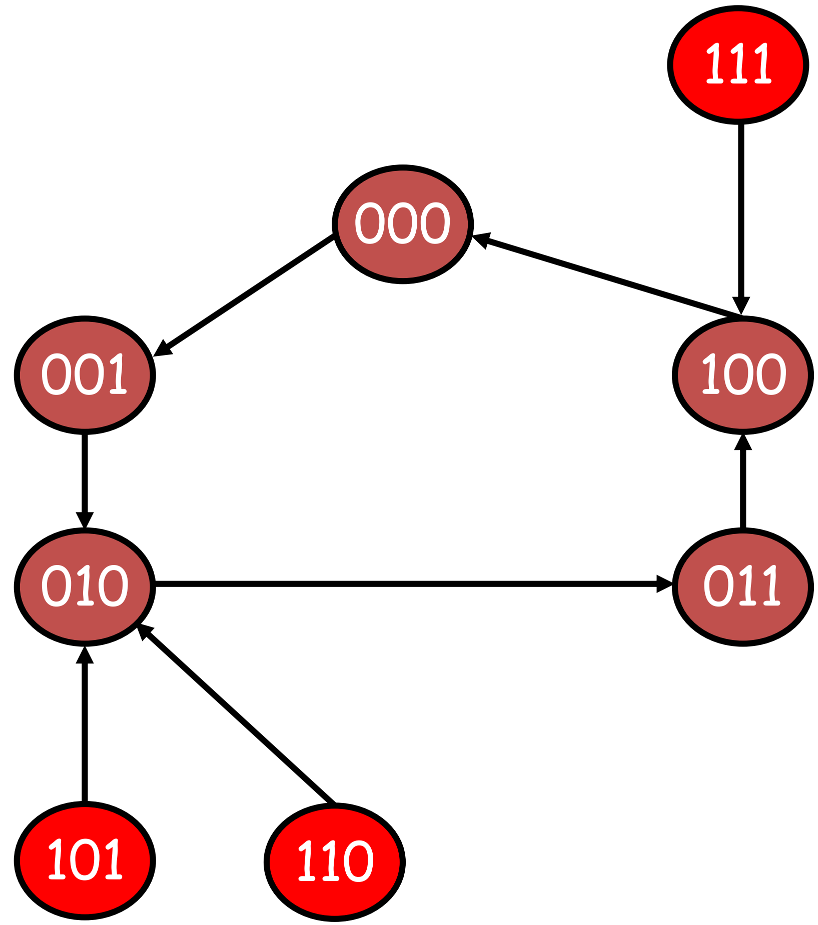
$$Y_1 = y_1' y_0 + y_1y_0' = y_1 \oplus y_0$$

y_2 \ y_1y_0	00	01	11	10
0	1	0	0	1
1	0	X	X	X

$$Y_0 = y_2' y_0'$$

Example: Unused States 2/4

Present State			Next State		
y_2	y_1	y_0	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	0	1	0
1	1	0	0	1	0
1	1	1	1	0	0



$Y_2 = y_1 y_0$

$Y_1 = y_1 \oplus y_0$

$Y_0 = y_2' y_0'$

The circuit is not locked type.

Example: Unused States 3/4

- Not using don't care conditions

Present State			Next State		
A	B	C	A	B	C
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	0	0	0

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

$A(t+1) = A'BC$

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

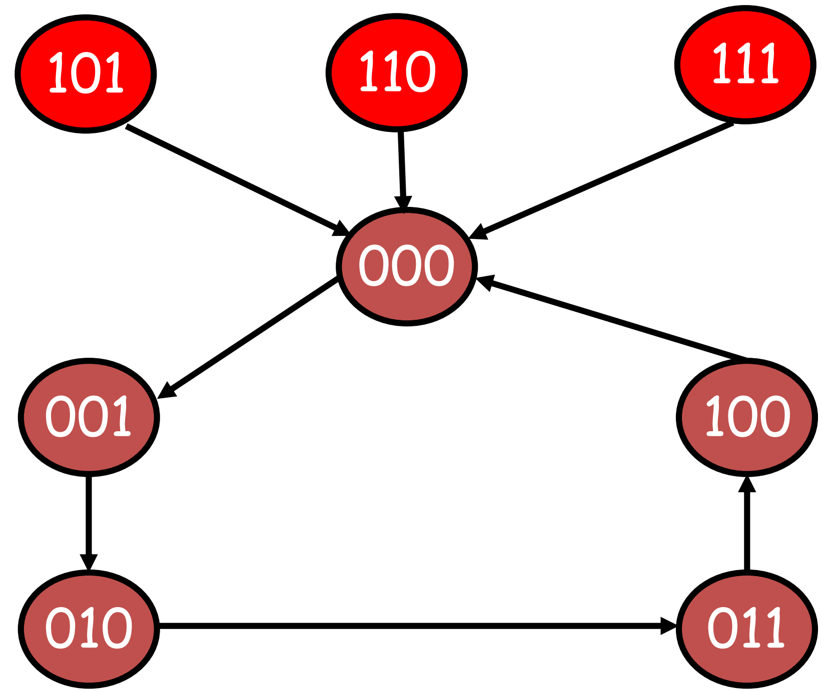
$$B(t+1) = A'B'C + A'BC' = A'(B \oplus C)$$

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

$C(t+1) = A'C'$

Example: Unused States 4/4

Present State			Next State		
A	B	C	A	B	C
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	0	0	0



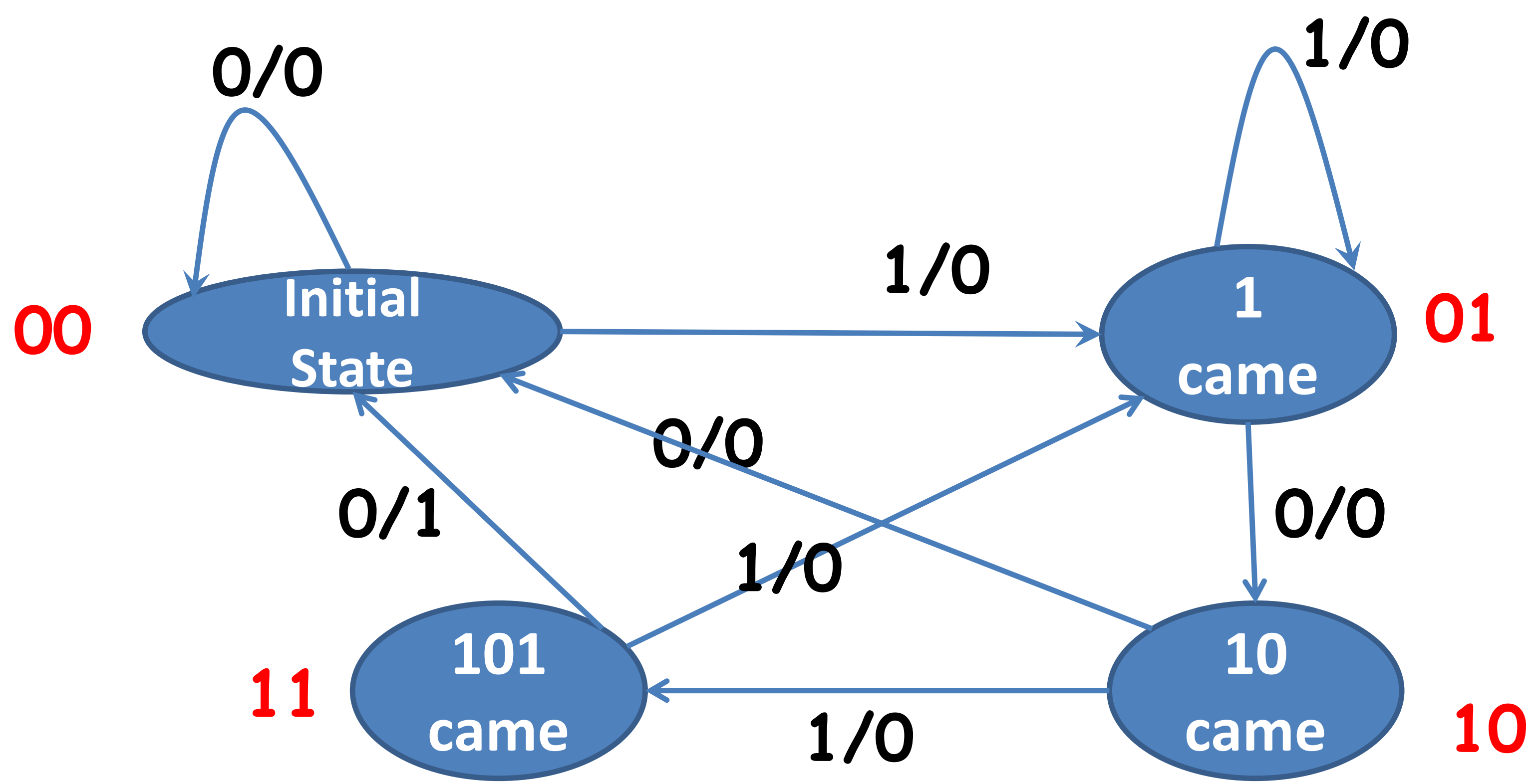
$A(t+1) = A'BC$

$B(t+1) = A'(B \oplus C)$

$C(t+1) = A'C'$

Design Example

- Design the synchronous sequential circuit which gives “1” as output when the last 4 values from the 1-bit input are 1010.
- Eaxmple: x= 1010 1011 ise z= 0001 0000



Mealy Machine

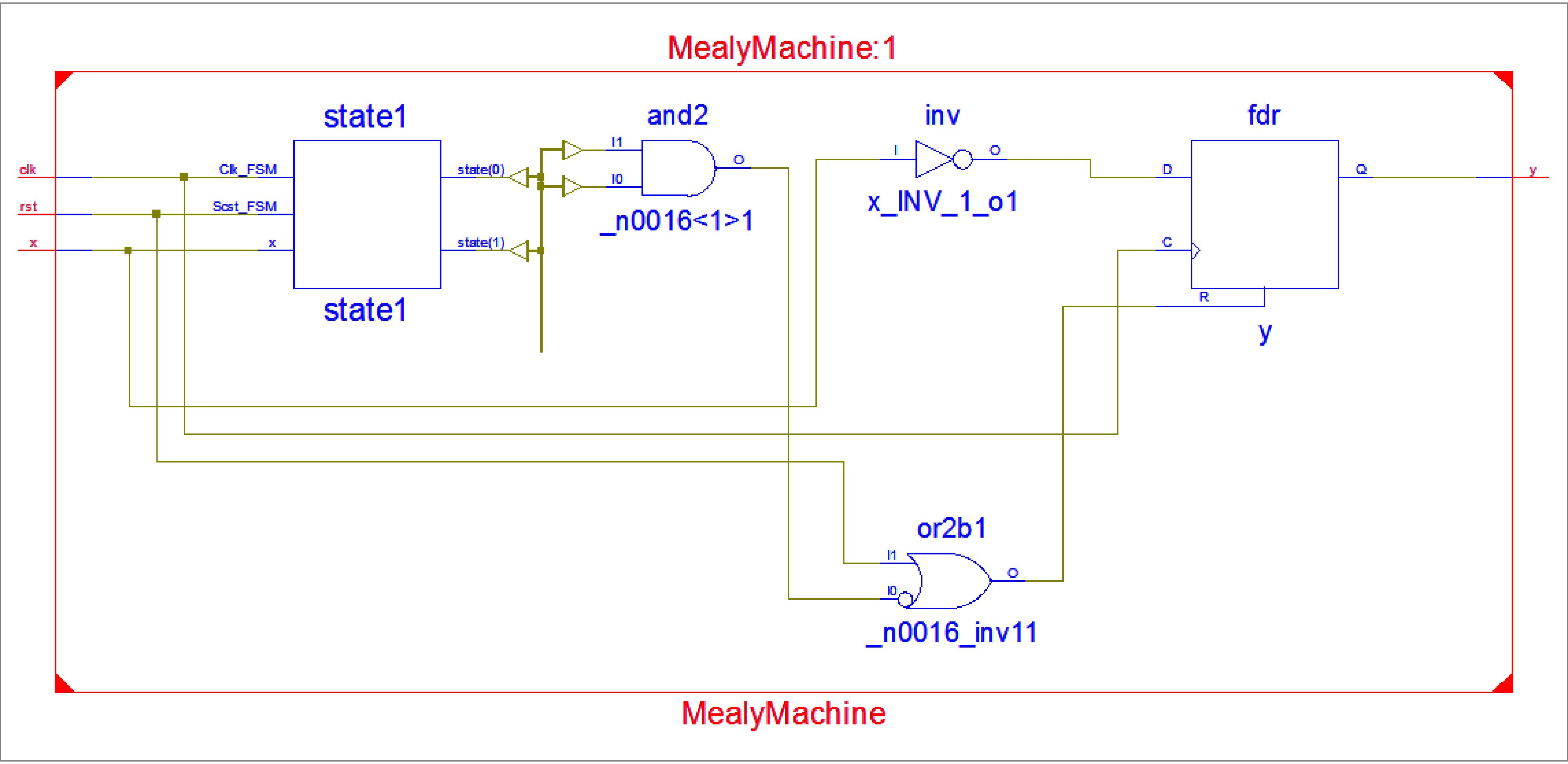
State Table

Present State		Input	Next State		Output
y_1	y_2	x	Y_1	Y_2	z
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	0	1	0

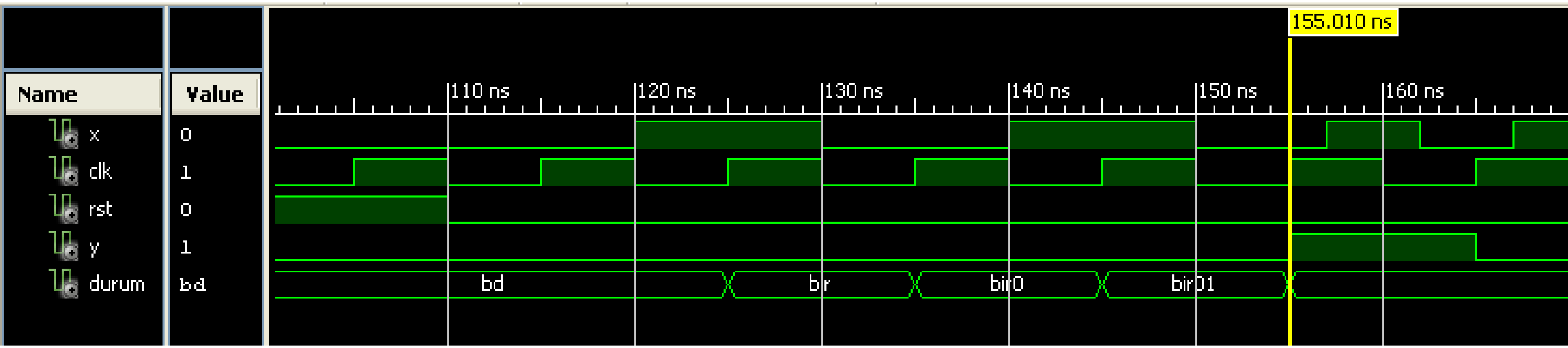
Verilog Code

```
module
    MealyMachine(x,y,clk
    ,rst);

input x,clk,rst;
output y;

parameter IS=0, S_1=1,
    S_10=2, S_101=3;

reg [1:0] state;
reg y;

always @(posedge clk)
 begin
  if (rst) begin state = IS; y = 0;
  end else
    case (state)
      IS: begin
        y = 0;
        if(x) state = S_1;
        else state = IS;
      end
     S_1: begin
        y = 0;
        if(x) state = S_1;
        else state = S_10;
    end
S_10: begin
        y = 0;
        if(x) state = S_101;
        else state = IS;
    end
      S_101: begin
        if(x) begin
          state = S_1; y =0;
        end else begin
          state = IS; y =1;
        end
      end
    endcase
  end
endmodule
```

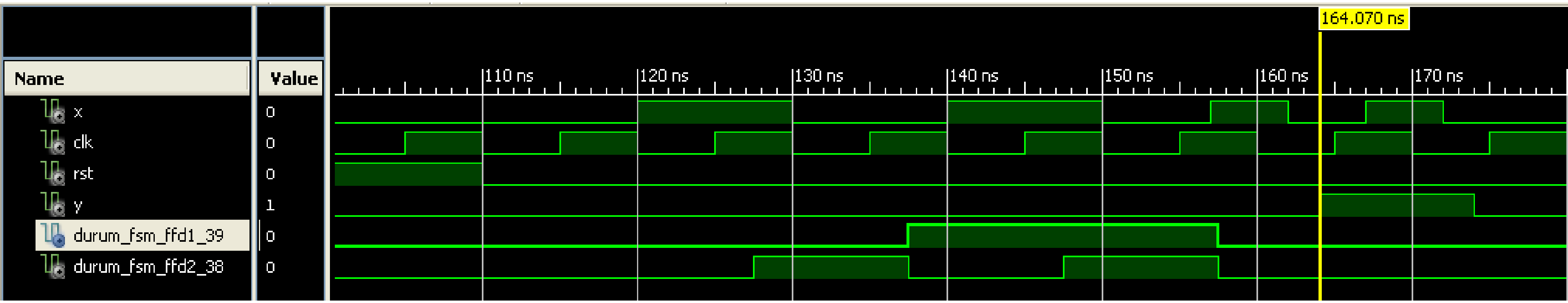
RTL Schematic



Timing Diagram

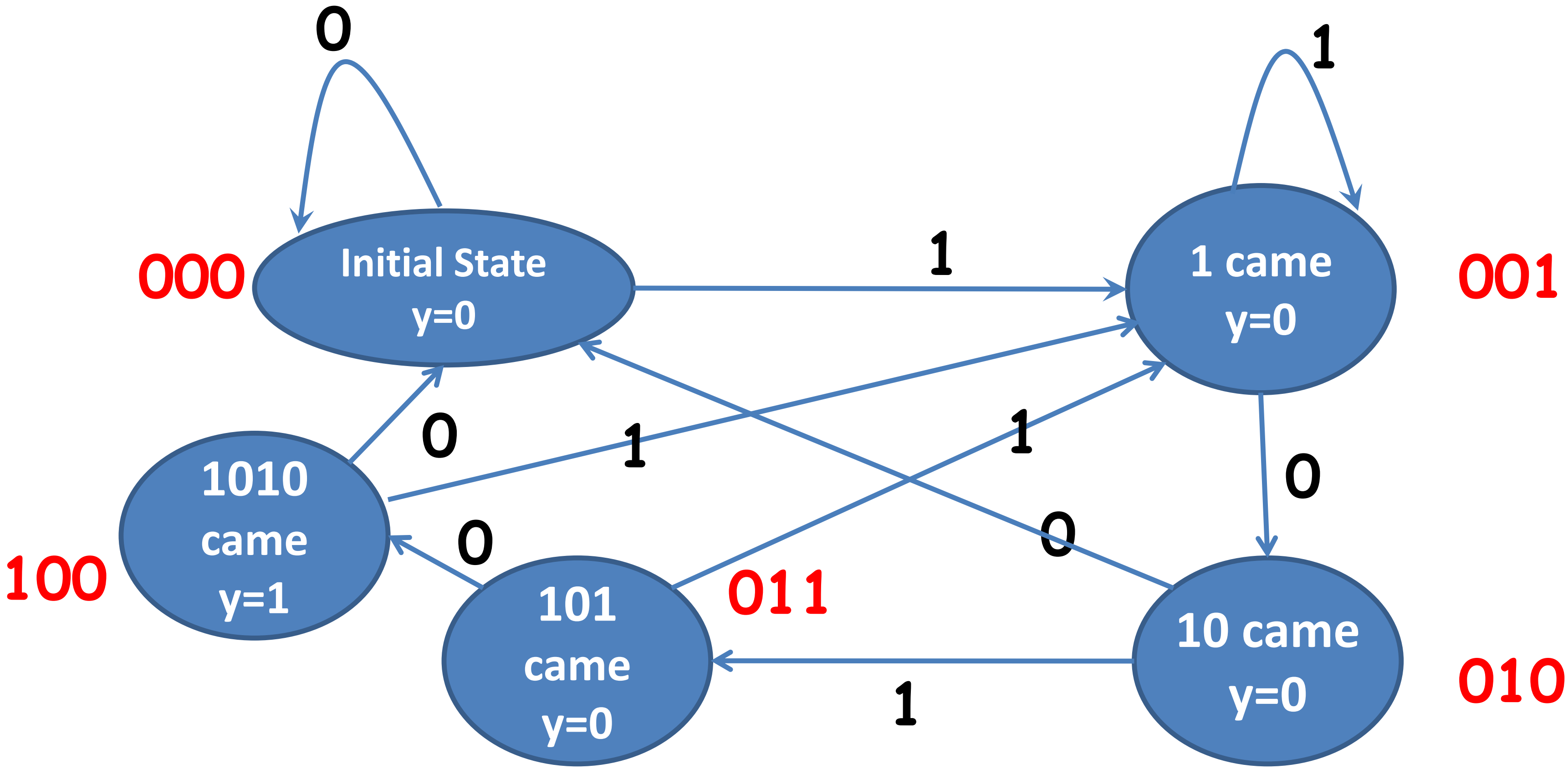


Ideal Case: Delay = 0



Not Ideal Case: Delay ≠ 0

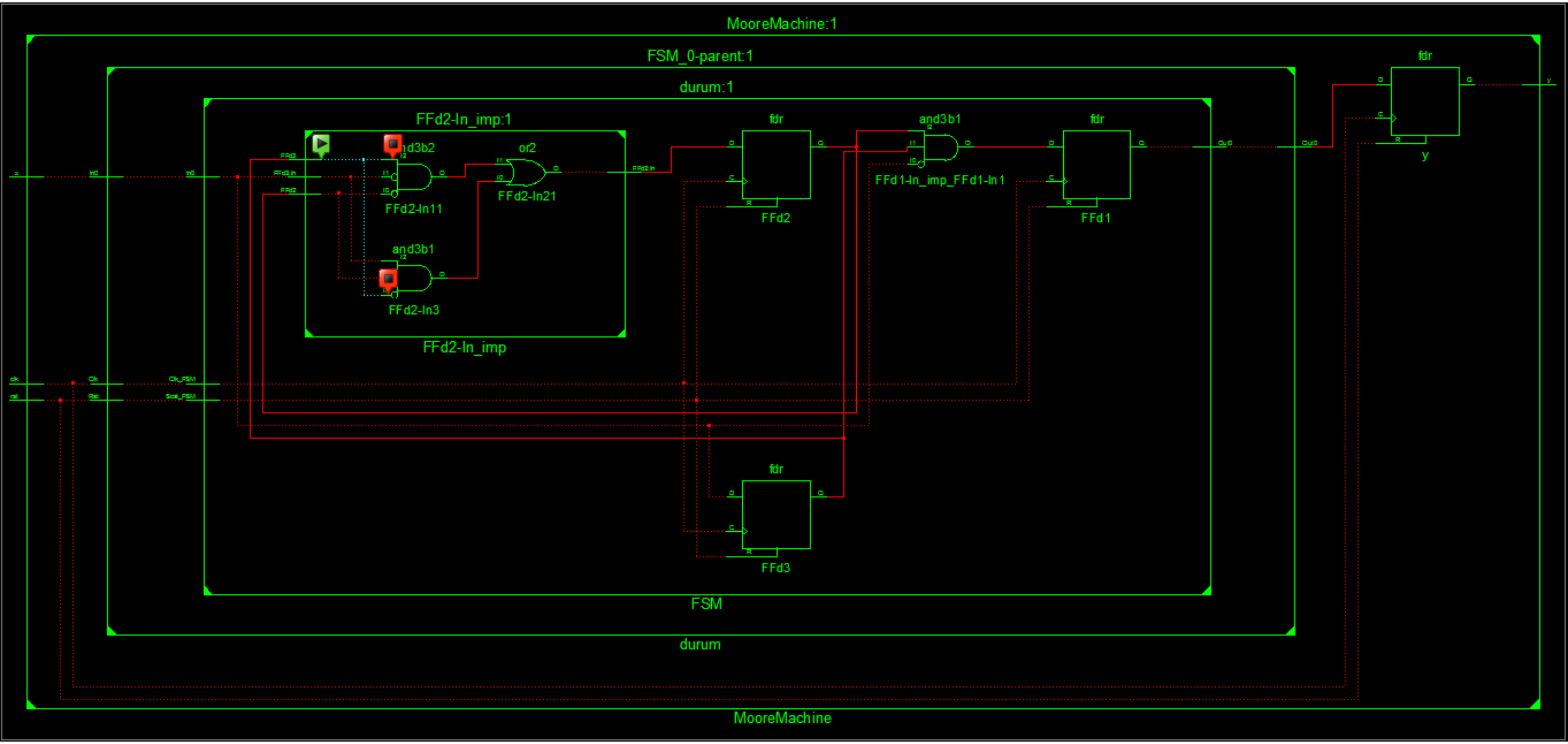
Moore Machine



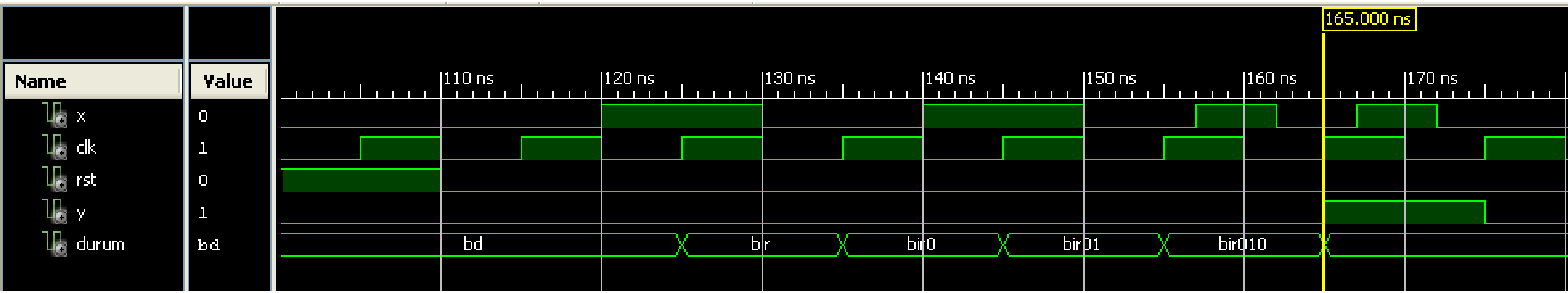
Verilog Code

```
module
    MealyMachine(x,y,clk
    ,rst);

İnput x,clk,rst;
output y;

Parameter IS=0, S_1=1,
    S_10=2, S_101=3,
    S_1010=4;

reg [2:0] state;
reg y;

always @(posedge clk)
 begin
  if (rst) state = IS; y = 0;
  else
    case (state)
      IS:
        y =0;
        if(x) state = S_1;
        else state = IS;
      S_1:
        y =0;
        if(x) state = S_1;
        else state = S_10;
      S_10:
        y =0;
        if(x) state = S_101;
        else state = IS;
      S_101:
        y =0;
        if(x) state = S_1;
        else state = S_1010;
      S_1010:
        y =1;
        if(x) state = S_1;
        else state = S_IS;
    endcase
  end
endmodule
```

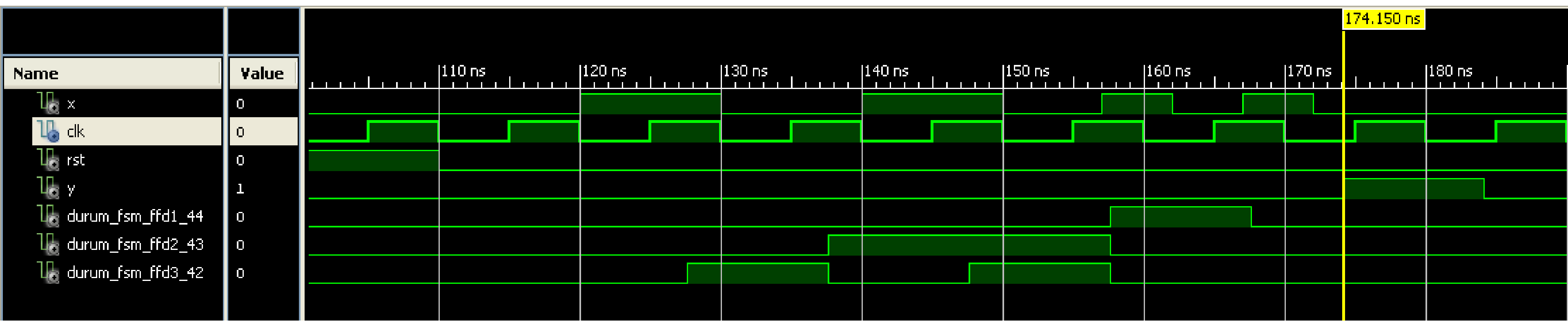
RTL Schematic



Timing Diagram



Ideal Case: Delay = 0



Not Ideal Case: Delay $\neq$ 0