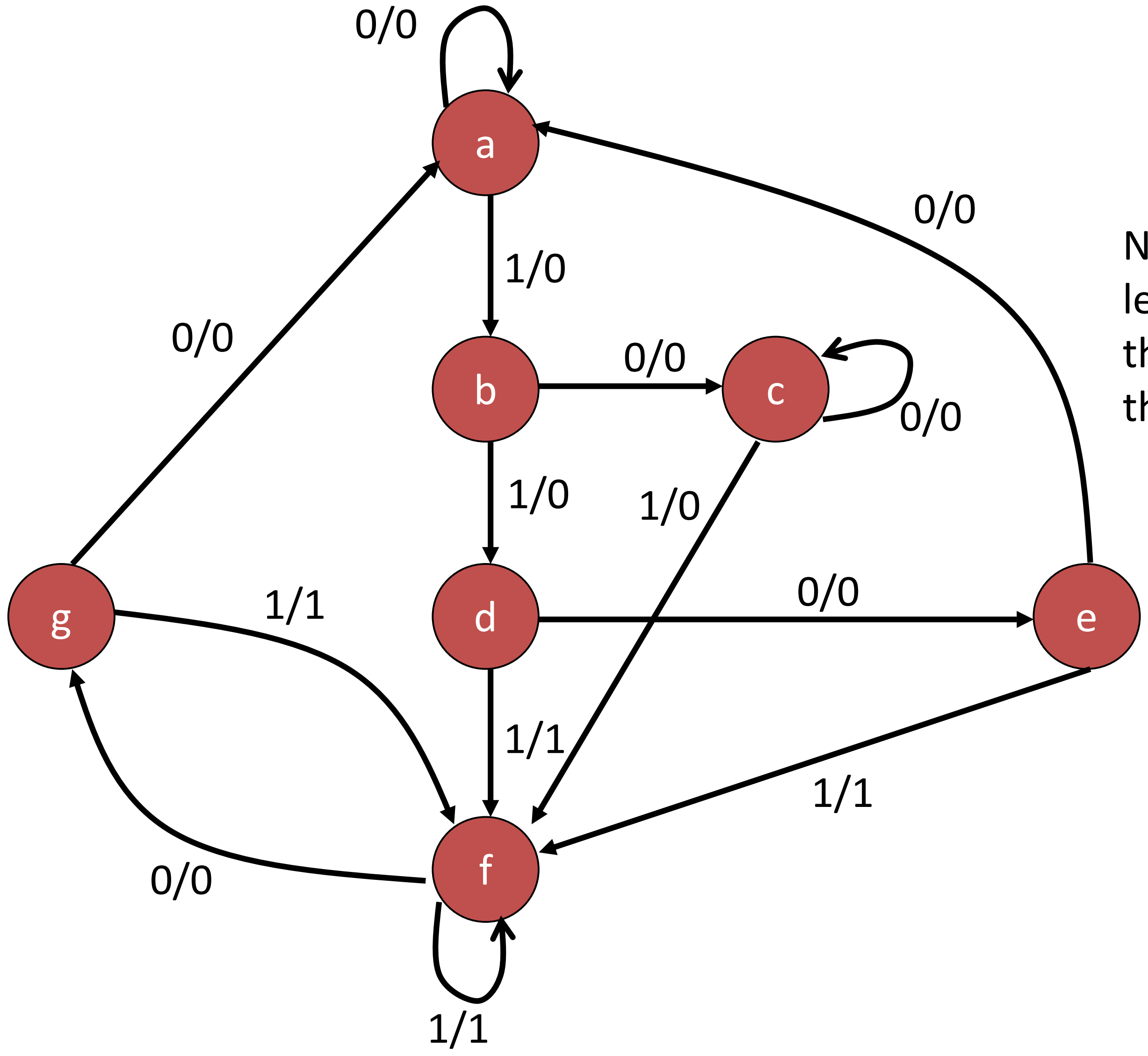
Synchronous Sequential Logic Part II

Mantıksal Tasarım – BBM231

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



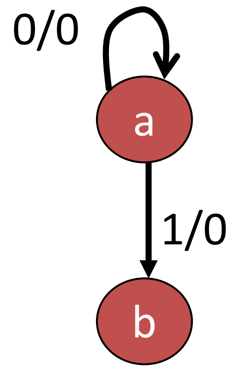
Note that we use letters to designate the states for the time being

Example: State Reduction

state	a	a	b	c	f	g	f	f	g	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			

- What is important?
  - not the states
  - but the output values the circuit generates
- Therefore, the problem is to find a circuit
  - **with fewer number of states,**
  - **but that produces the same output pattern for any given input pattern, starting with the same initial state**

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

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 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 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
- >> d & f are now equivalent

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 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
- >> b & c are now equivalent

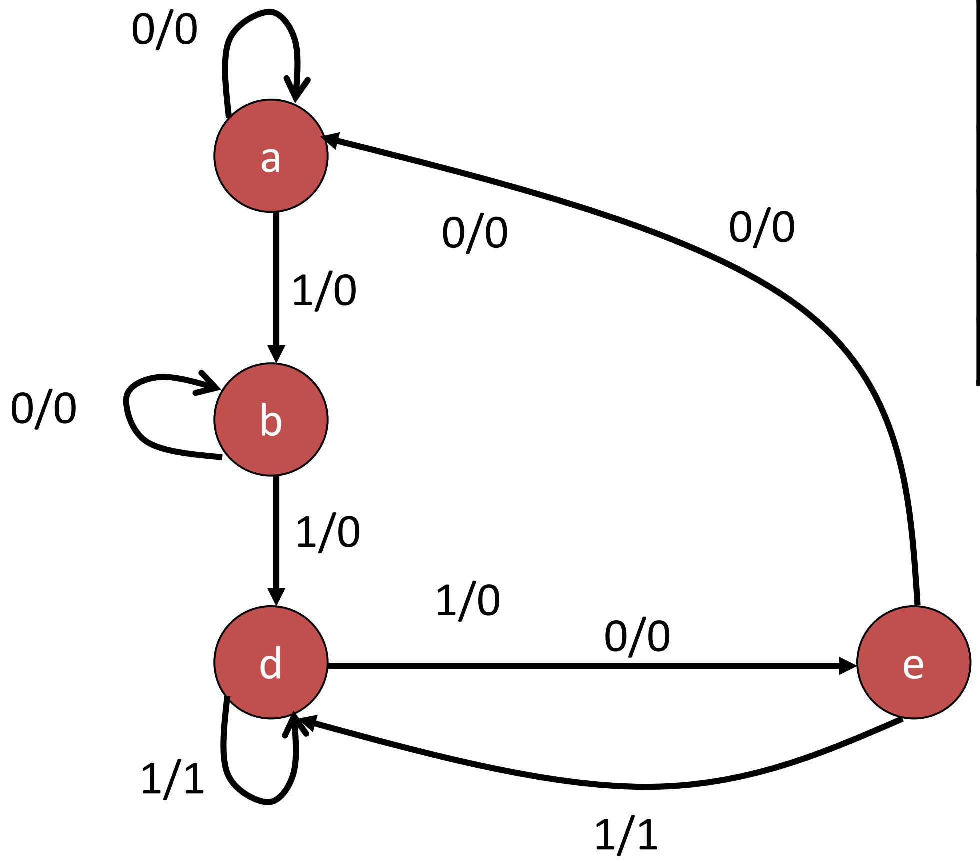
State Reduction Technique 6/7

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

- Any more?

- NO! We stop when there are no remaining equivalent states

State Reduction Technique 7/7

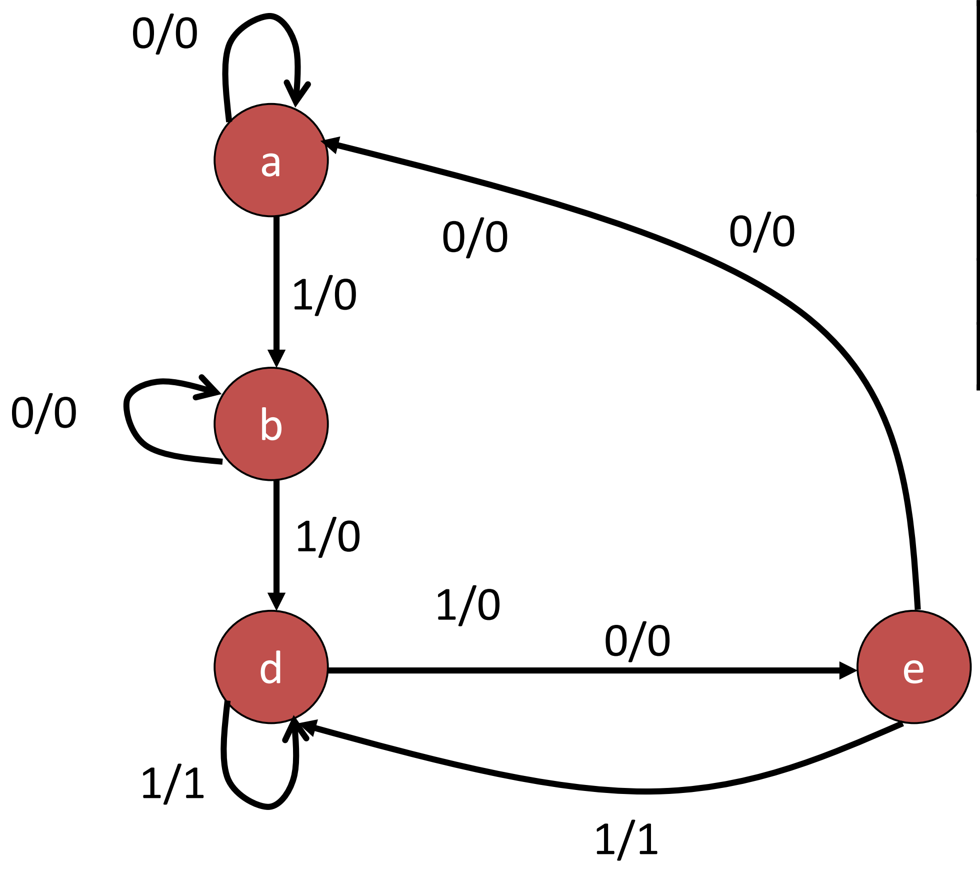


present state	next state x = 0	next state x = 1	Output x = 0	Output x = 1
a	a	b	0	0
b	b	d	0	0
d	e	d	0	1
e	a	d	0	1

How many flip flops do we need?

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 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 2 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
- <u>Example</u>: 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	000001
S_1	001	001	000010
S_2	010	011	000100
S_3	011	010	001000
S_4	100	110	010000
S_5	101	111	100000
S_6	111	101	100000
S_7	111	100	100000

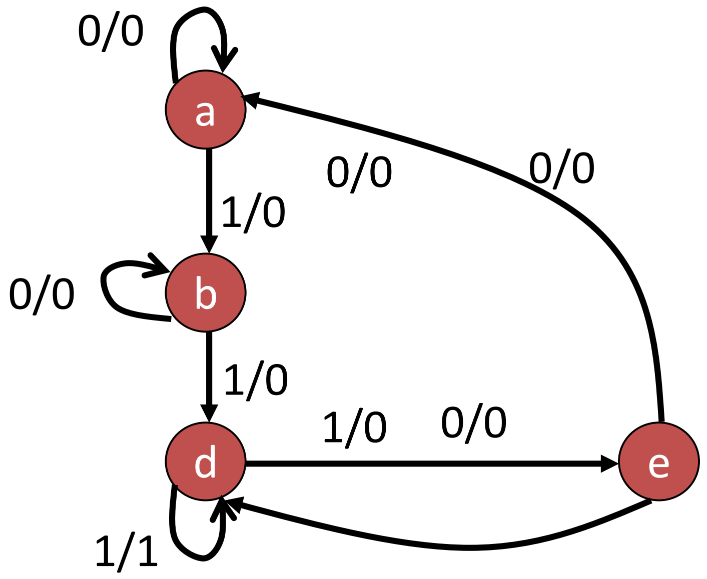
State Assignments 2/4

- 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
    - <u>flip-flops</u>: number of flip-flops is determined by the number of states
    - <u>combinational part</u>:
      - output equations
      - flip-flop input equations

Design Process

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

Design Steps (cont.)

- The design steps

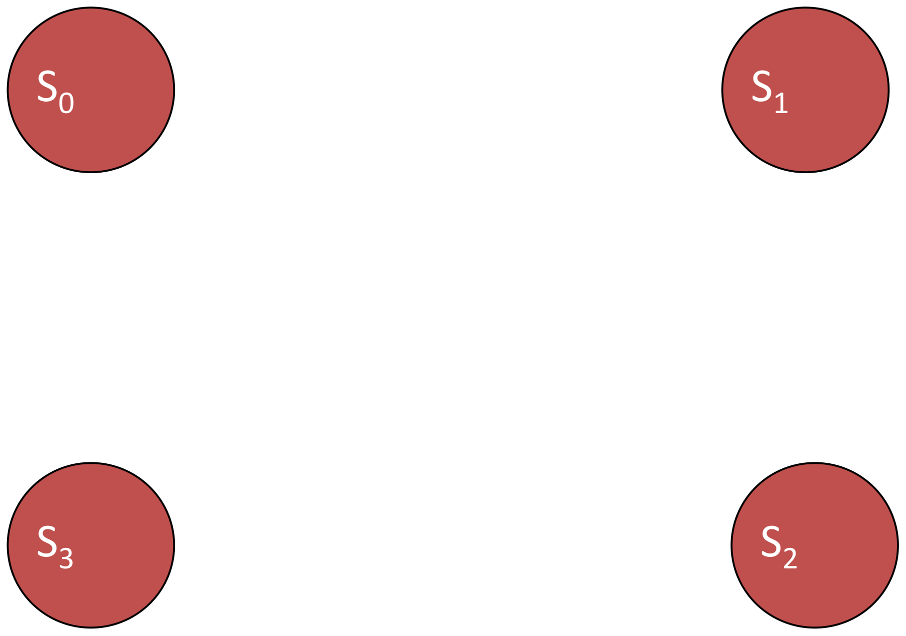
1. Given a verbal description of desired operation, derive the state diagram from that.
2. Reduce the number of states if necessary and possible
3. Do state assignment
4. Obtain the encoded state table
5. Derive the simplified flip-flop input equations
6. Derive the simplified output equations
7. Draw 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

Example: State Diagram

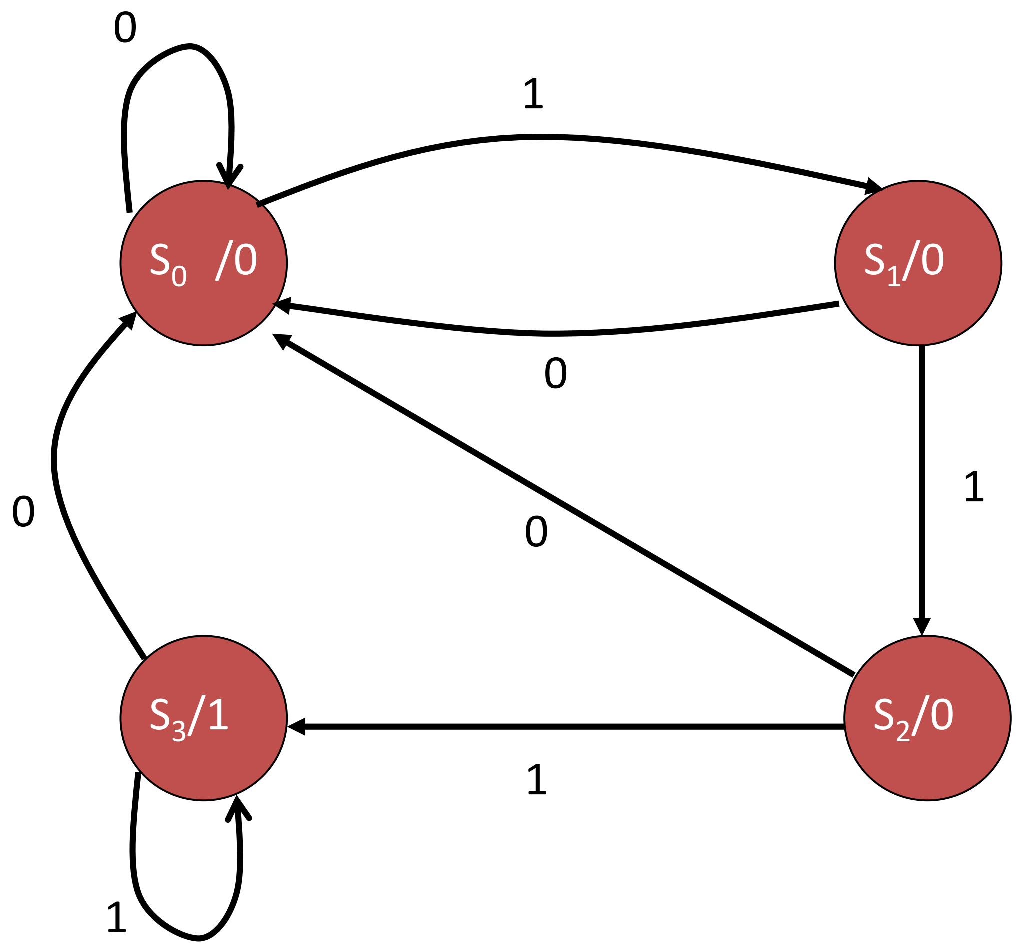
- Step 1: Derive the state diagram



What kind is this?
Mealy or Moore?

Example: State Diagram

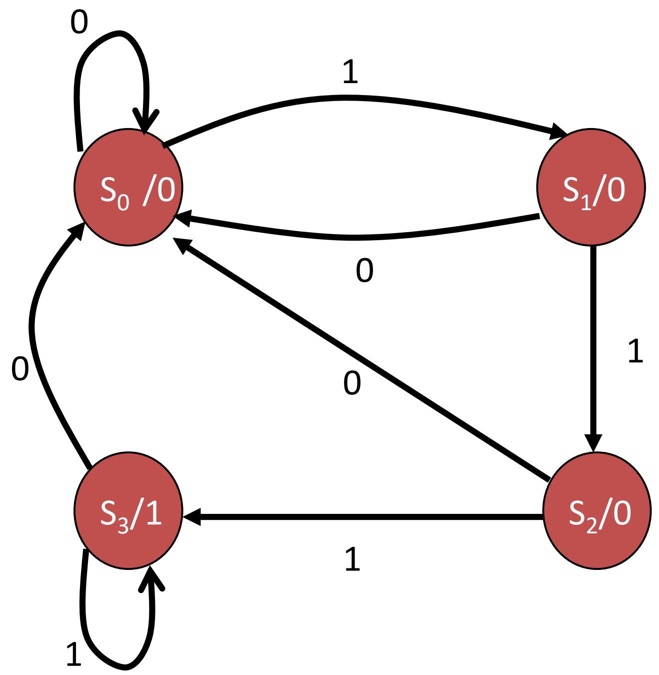
- Step 1: Derive the state diagram



Moore Machine

Synthesis with D Flip-Flops 1/5

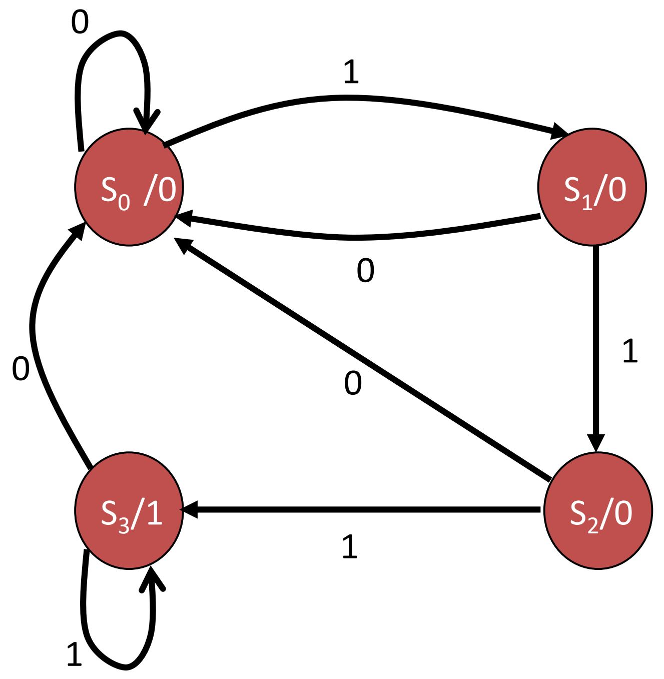
- The number of flip-flops
  - Four states
  - ? flip-flops
- State reduction
  - not possible in this case
- State Assignment
  - Use binary encoding
    - $S_0 \rightarrow$ **00**
    - $S_1 \rightarrow$ **01**
    - $S_2 \rightarrow$ **10**
    - $S_3 \rightarrow$ **11**



Synthesis with D Flip-Flops 2/5

- <u>Step 4</u>: Obtain the state table

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

D_A =

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

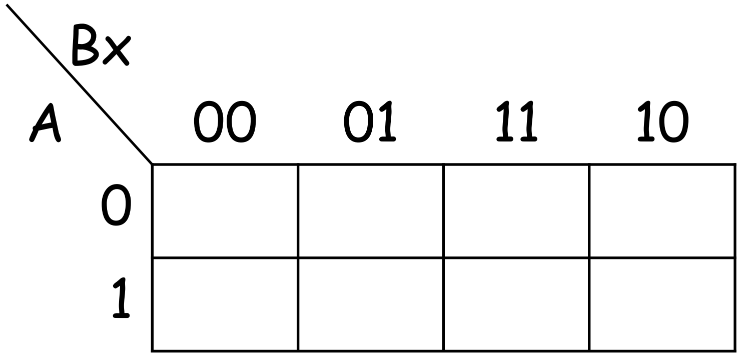
Present state		Input	Next state		Output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

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

$D_A = Ax + Bx$

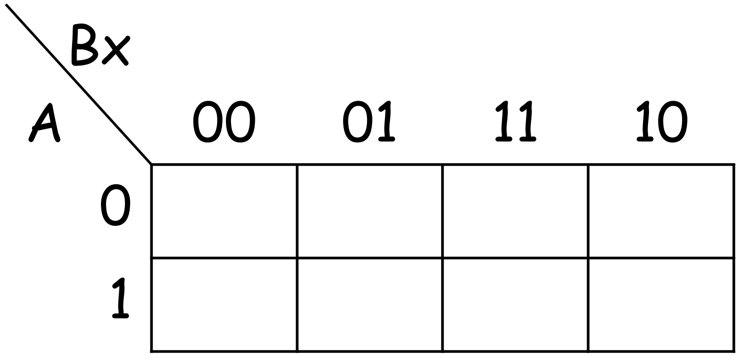
Synthesis with D Flip-Flops 3/5

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



$D_B =$

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



y =

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

$$\begin{array}{c|cccc} A \backslash Bx & 00 & 01 & 11 & 10 \\ \hline 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \end{array}$$

$D_B = Ax + B'x$

$$\begin{array}{c|cccc} A \backslash Bx & 00 & 01 & 11 & 10 \\ \hline 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \end{array}$$

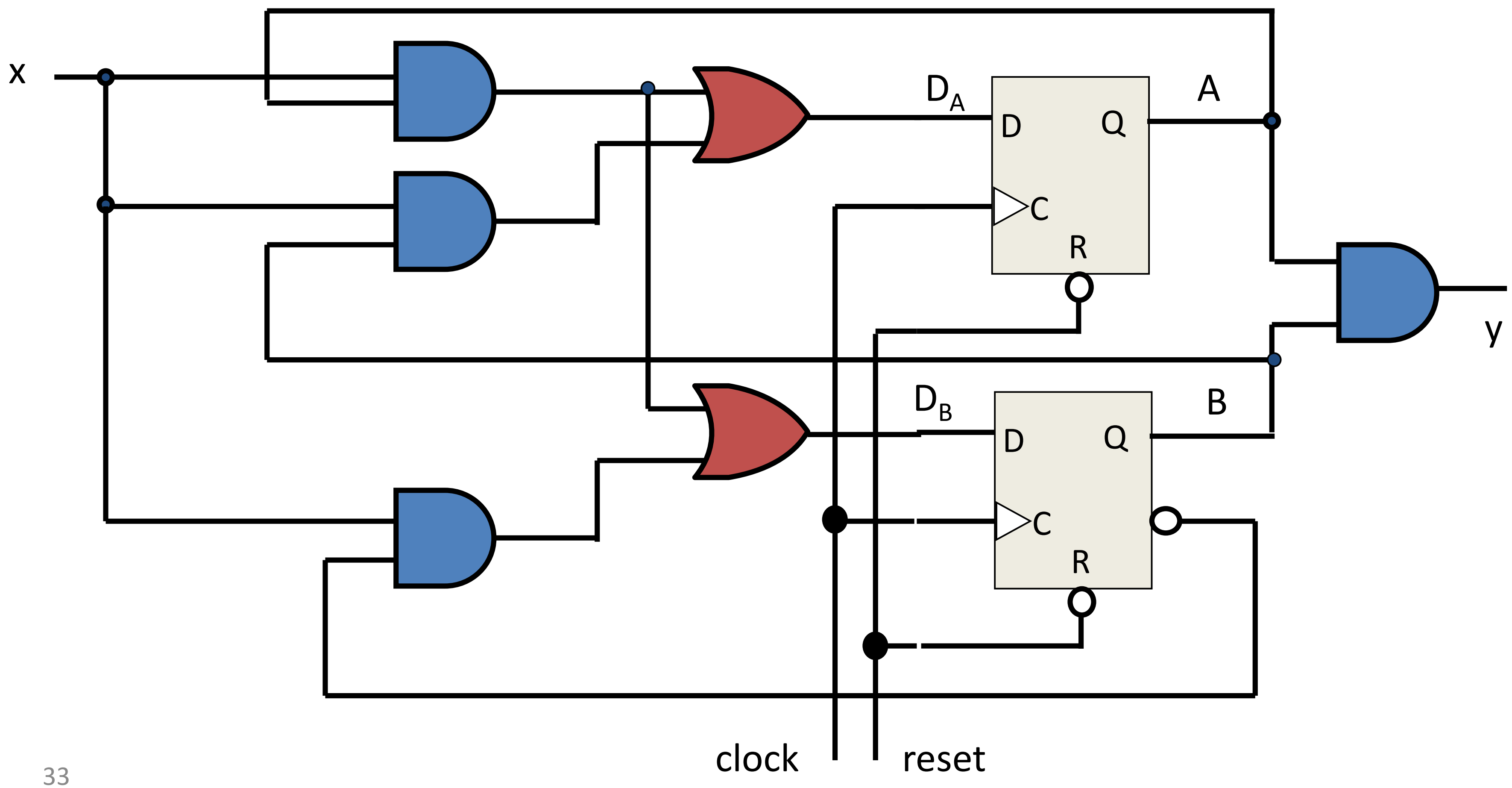
$y = AB$

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

Synthesis with D Flip-Flops 5/5

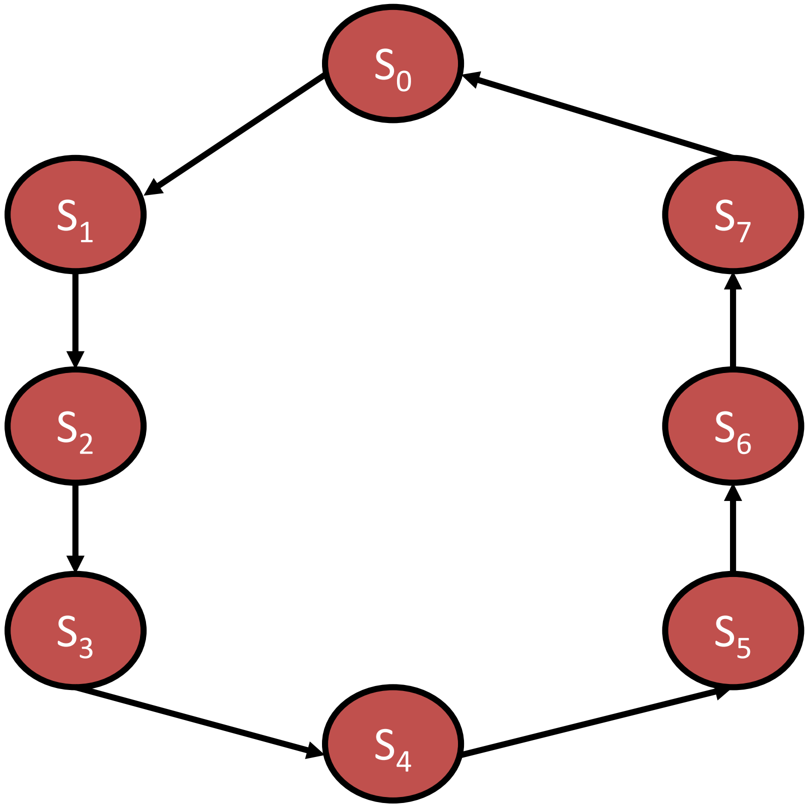
- Step 8: Draw the logic diagram

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



Synthesis with T Flip-Flops 1/4

- Example: 3-bit binary counter with T flip-flops



State Diagram

How many flip-flops?

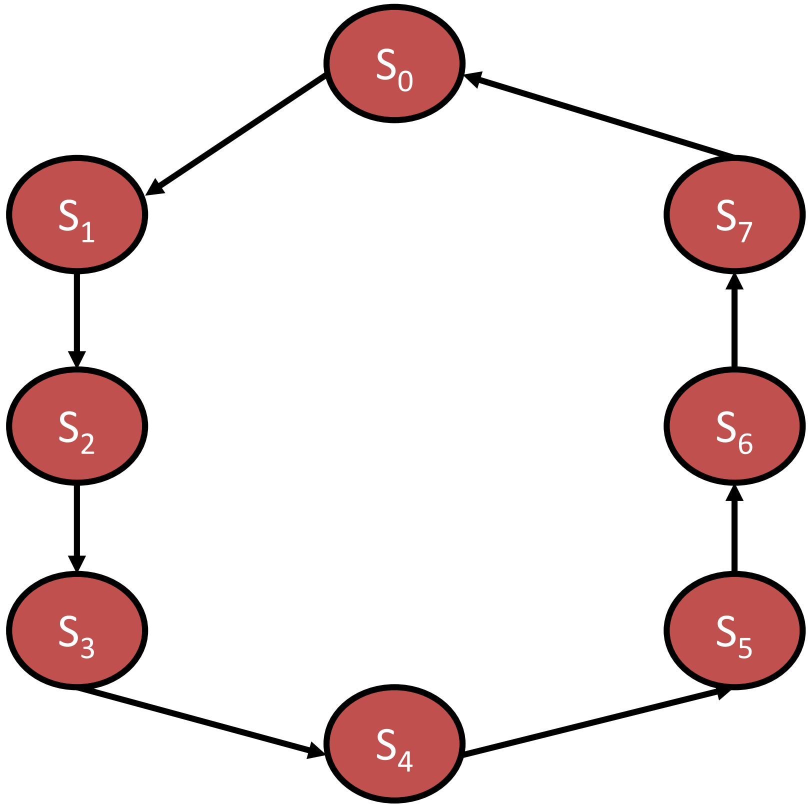
State assignments:

- $S_0 \rightarrow$ `000`
- $S_1 \rightarrow$ `001`
- $S_2 \rightarrow$ `010`
- ...
- $S_7 \rightarrow$ `111`

Synthesis with T Flip-Flops 1/4

- Example: 3-bit binary counter with T flip-flops

  0→1→2→ … → 7 → 0 → 1 → 2



State Diagram

How many flip-flops?

State assignments:

- S_0 → `000`
- S_1 → `001`
- S_2 → `010`
- ...
- S_7 → `111`

Synthesis with T Flip-Flops 2/4

- **State Table**

present state			next state			FF inputs		
A_2	A_1	A_0	A_2	A_1	A_0	T_2	T_1	T_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	1	0	1			
1	0	1	1	1	0			
1	1	0	1	1	1			
1	1	1	0	0	0			

Synthesis with T Flip-Flops 3/4

Present state			FF inputs		
A_2	A_1	A_0	T_2	T_1	T_0
0	0	0	0	0	1
0	0	1	0	1	1
0	1	0	0	0	1
0	1	1	1	1	1
1	0	0	0	0	1
1	0	1	0	1	1
1	1	0	0	0	1
1	1	1	1	1	1

- Flip-Flop input equations

A_2 \ $A_1 A_0$	00	01	11	10
0				
1				

T_2 =

A_2 \ $A_1 A_0$	00	01	11	10
0				
1				

T_1 =

T_0 = ?

Synthesis with T Flip-Flops 3/4

Present state			FF inputs		
A_2	A_1	A_0	T_2	T_1	T_0
0	0	0	0	0	1
0	0	1	0	1	1
0	1	0	0	0	1
0	1	1	1	1	1
1	0	0	0	0	1
1	0	1	0	1	1
1	1	0	0	0	1
1	1	1	1	1	1

- Flip-Flop input equations

$$
\begin{array}{c|cccc}
A_2 \backslash A_1 A_0 & 00 & 01 & 11 & 10 \\
\hline
0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0
\end{array}
$$

$T_2 = A_1A_0$

$$
\begin{array}{c|cccc}
A_2 \backslash A_1 A_0 & 00 & 01 & 11 & 10 \\
\hline
0 & 0 & 1 & 1 & 0 \\
1 & 0 & 1 & 1 & 0
\end{array}
$$

$T_0 = 1$

$T_1 = A_0$

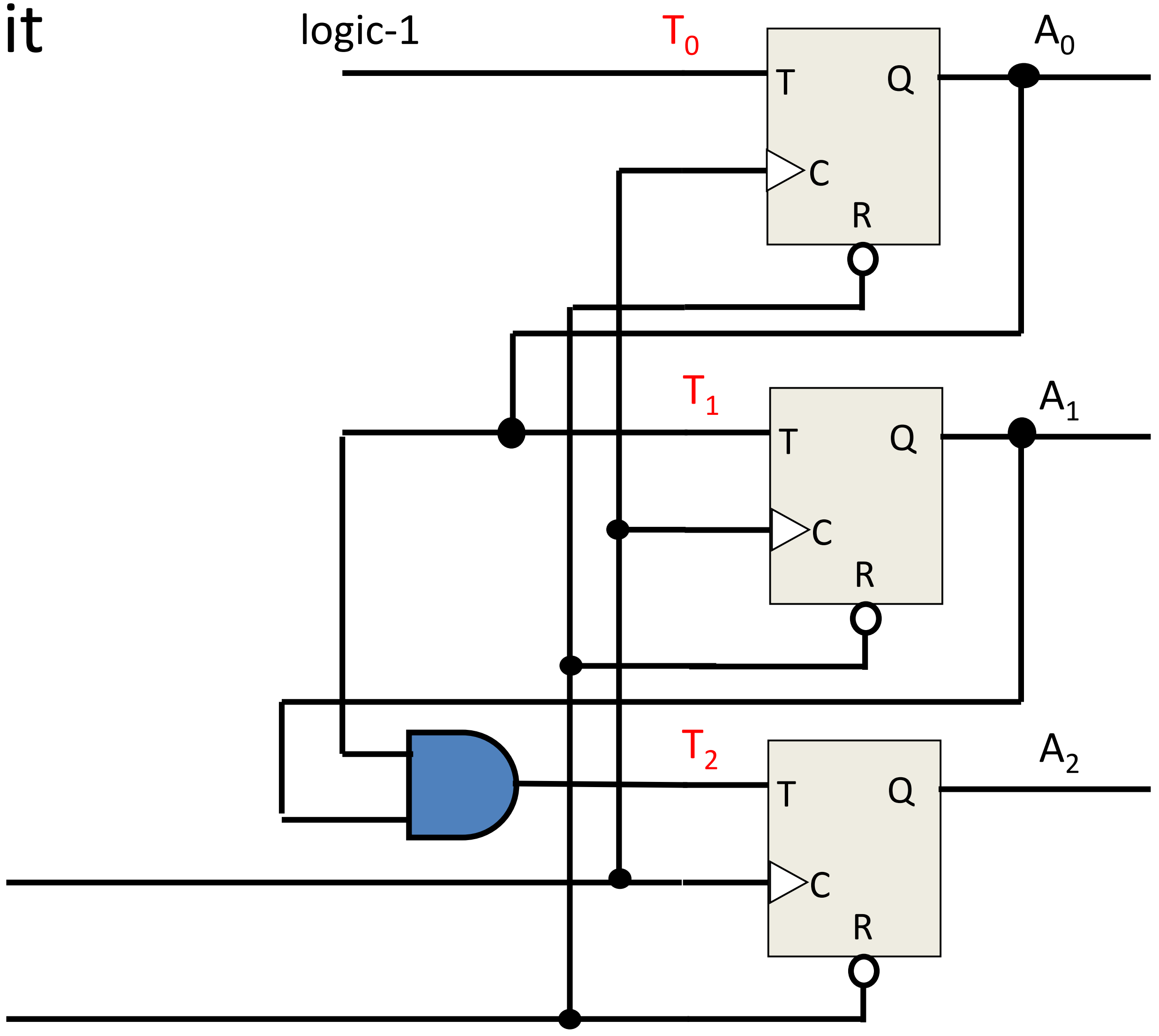
Synthesis with T Flip-Flops 4/4

- Circuit

$T_2 = A_1 A_0$

$T_1 = A_0$

$T_0 = 1$



Synthesis with JK Flip-Flops 1/4

$Q(t+1) = JQ' + K'Q$

J	K	Q(t+1)
0	0	Q
0	1	0
1	0	1
1	1	Q'

- State Table & JK FF Inputs

Present state		Input	next state		Flip-flop inputs			
A	B	x	A	B	J_A	K_A	J_B	K_B
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	1	0	1	X	X	1
0	1	1	0	1	0	X	X	0
1	0	0	1	0	X	0	0	X
1	0	1	1	1	X	0	1	X
1	1	0	1	1	X	0	X	0
1	1	1	0	0	X	1	X	1

Synthesis with JK Flip-Flops 2/4

- Optimize the flip-flop input equations

					Flip-flop inputs			
A	B	x	A(t+1)	B(t+1)	J_A	K_A	J_B	K_B
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	1	0	1	X	X	1
0	1	1	0	1	0	X	X	0
1	0	0	1	0	X	0	0	X
1	0	1	1	1	X	0	1	X
1	1	0	1	1	X	0	X	0
1	1	1	0	0	X	1	X	1

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

$J_A = Bx'$

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

$J_B = x$

Synthesis with JK Flip-Flops 3/4

					Flip-flop inputs			
A	B	x	A(t+1)	B(t+1)	J_A	K_A	J_B	K_B
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	1	0	1	X	X	1
0	1	1	0	1	0	X	X	0
1	0	0	1	0	X	0	0	X
1	0	1	1	1	X	0	1	X
1	1	0	1	1	X	0	X	0
1	1	1	0	0	X	1	X	1

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

$K_A = Bx$

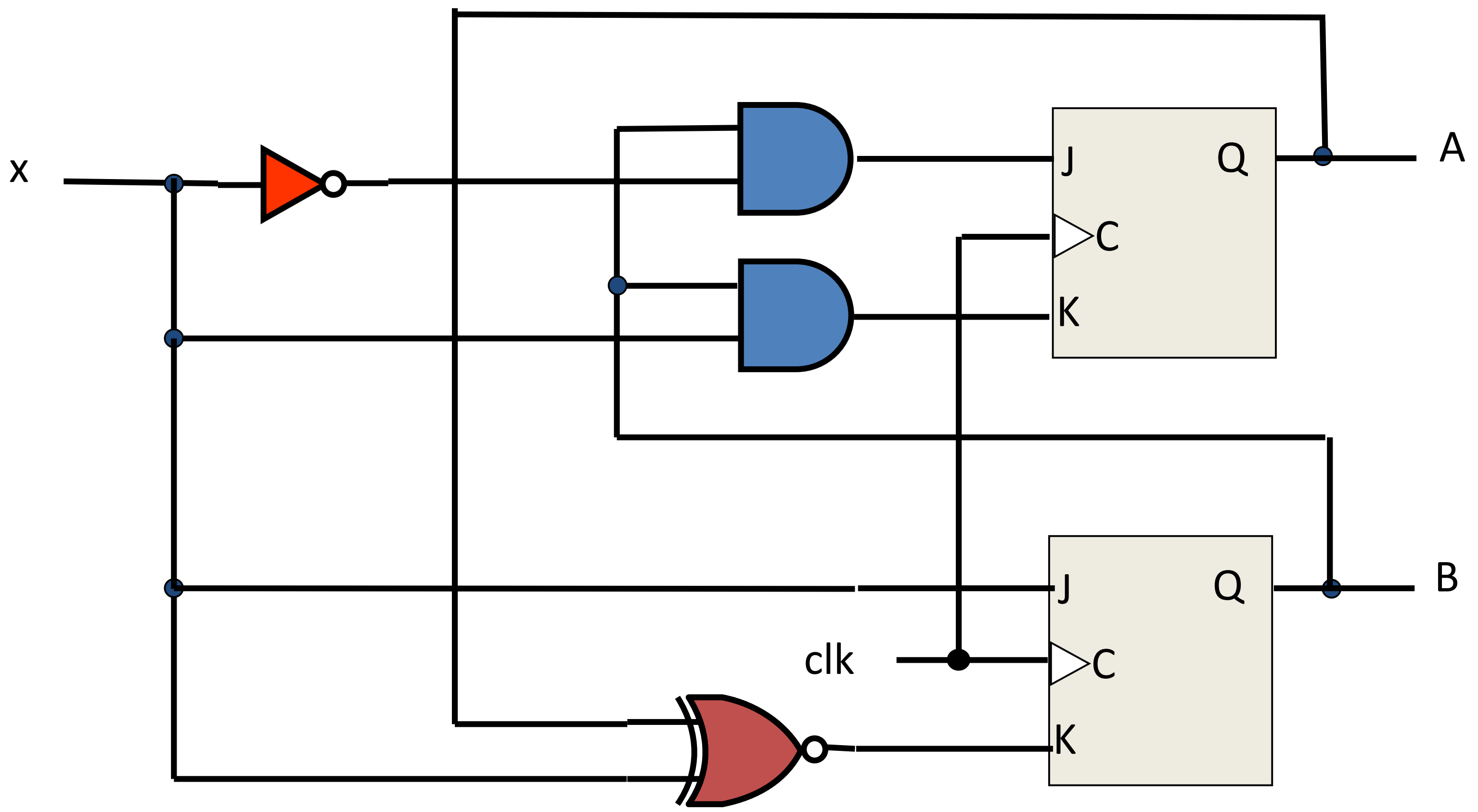
A \ Bx	00	01	11	10
0	X	X	0	1
1	X	X	1	0

$K_B = (A \oplus x)'$

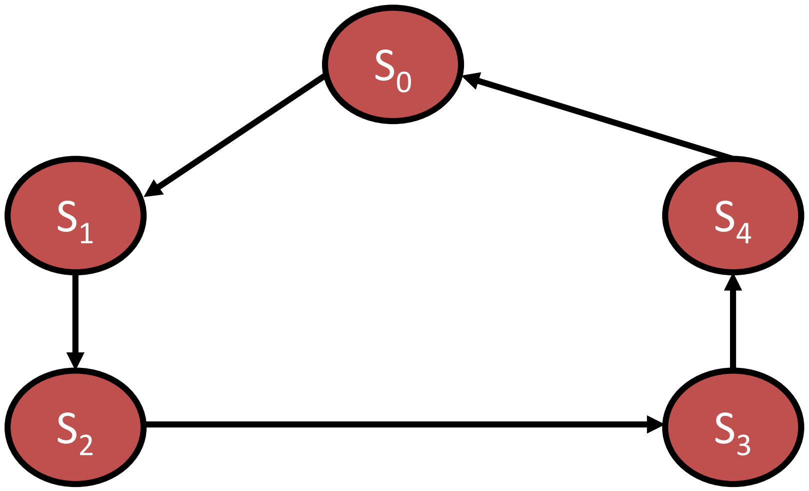
Synthesis with JK Flip-Flops 4/4

- Logic diagram

$J_A = Bx'$ $K_A = Bx$ $J_B = x$ $K_B = (A \oplus x)'$



Unused States


$$\begin{tikzcd} & S_0 \arrow[dl] & \\ S_1 \arrow[d] & & S_4 \arrow[ul] \\ S_2 \arrow[rr] & & S_3 \arrow[u] \end{tikzcd}$$

Modulo-5 counter

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

Example: Unused States 1/4

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 \ BC	00	01	11	10
0				
1				

A(t+1) =

A \ BC	00	01	11	10
0				
1				

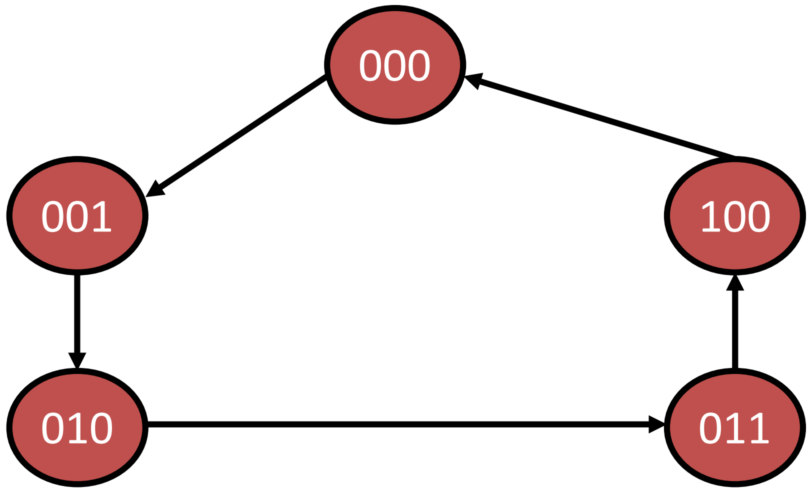
B(t+1) =

A \ BC	00	01	11	10
0				
1				

C(t+1) =

Example: Unused States 2/4

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
1	0	1			
1	1	0			
1	1	1			


$$\begin{tikzcd} & 000 \arrow[dl] & \\ 001 \arrow[d] & & 100 \arrow[ul] \\ 010 \arrow[rr] & & 011 \arrow[u] \end{tikzcd}$$

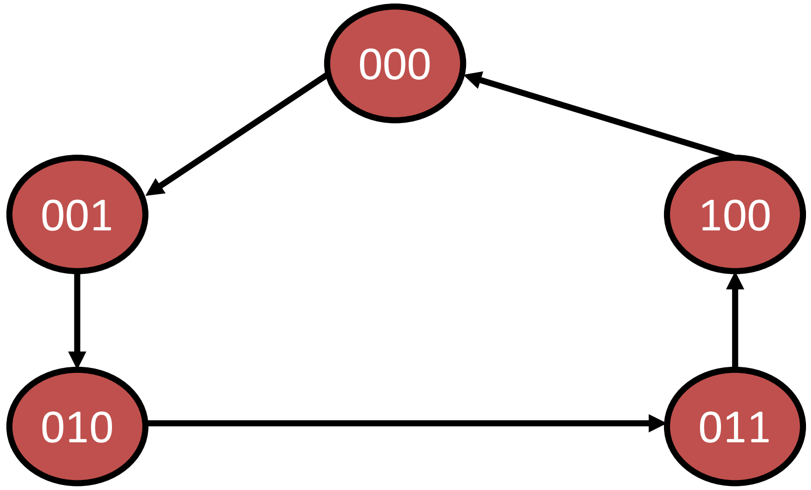
$A(t+1) = BC$

$B(t+1) = B \oplus C$

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

Example: Unused States 2/4

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
1	0	1	0	1	0
1	1	0	0	1	0
1	1	1	1	0	0


$$\begin{tikzcd} 000 \arrow[d] & 100 \arrow[l] \\ 001 \arrow[d] & \\ 010 \arrow[r] & 011 \arrow[uu] \end{tikzcd}$$

$A(t+1) = BC$

$B(t+1) = B \oplus C$

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

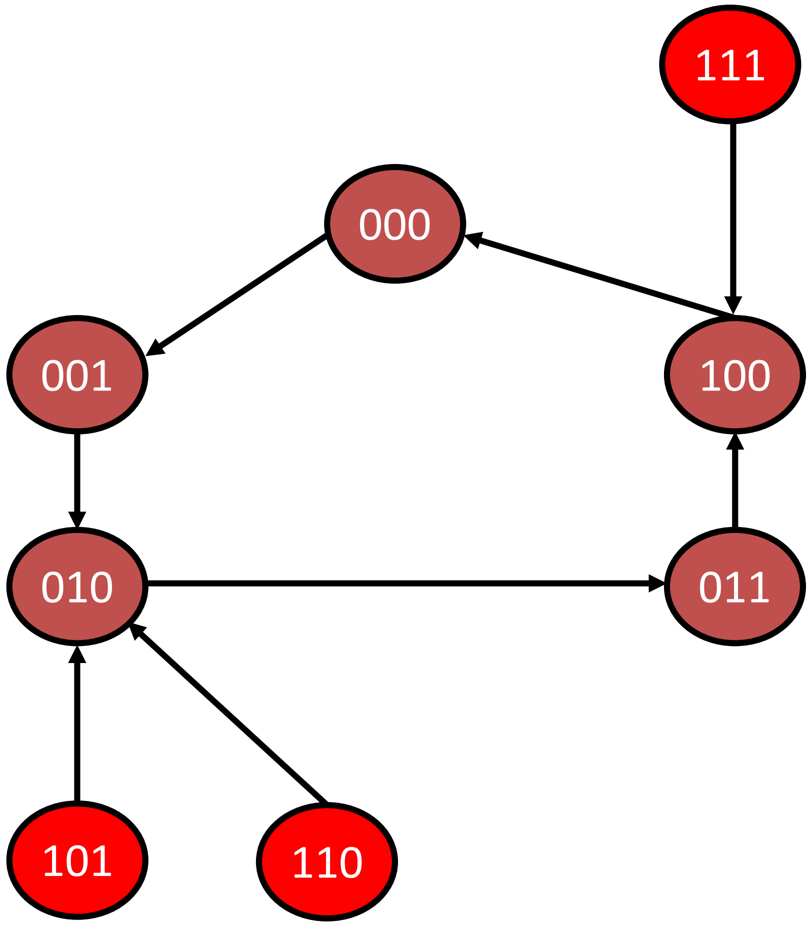
Example: Unused States 2/4

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
1	0	1	0	1	0
1	1	0	0	1	0
1	1	1	1	0	0

$A(t+1) = BC$

$B(t+1) = B \oplus C$

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



Example: Unused States 3/4

- This time not using don't care conditions, instead assigning them all 0s

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

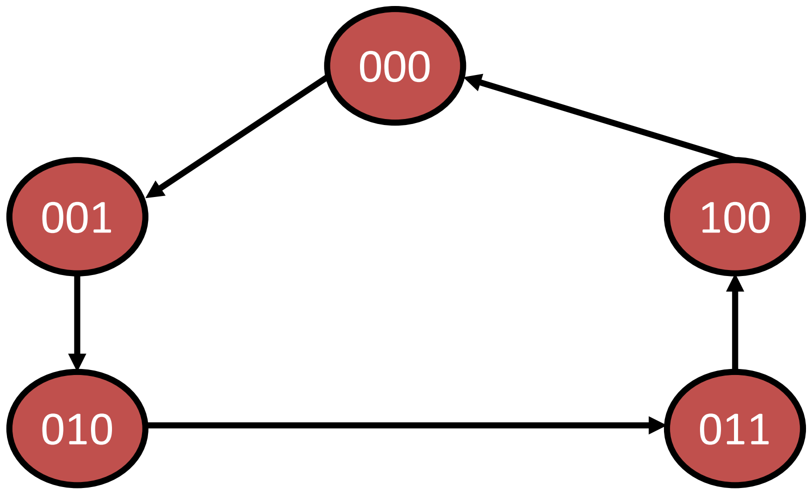
B(t+1) = A'B'C + A'BC'
= A'(B ⊕ C)

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
1	**0**	**1**			
1	**1**	**0**			
1	**1**	**1**			



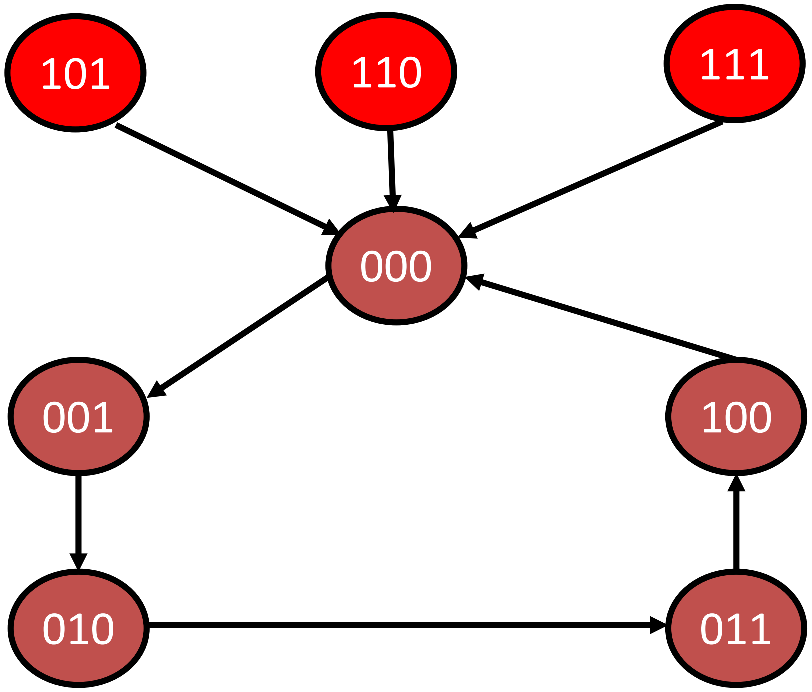
A(t+1) = A'BC

B(t+1) = A'(B ⊕ C)

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
1	0	1			
1	1	0			
1	1	1			



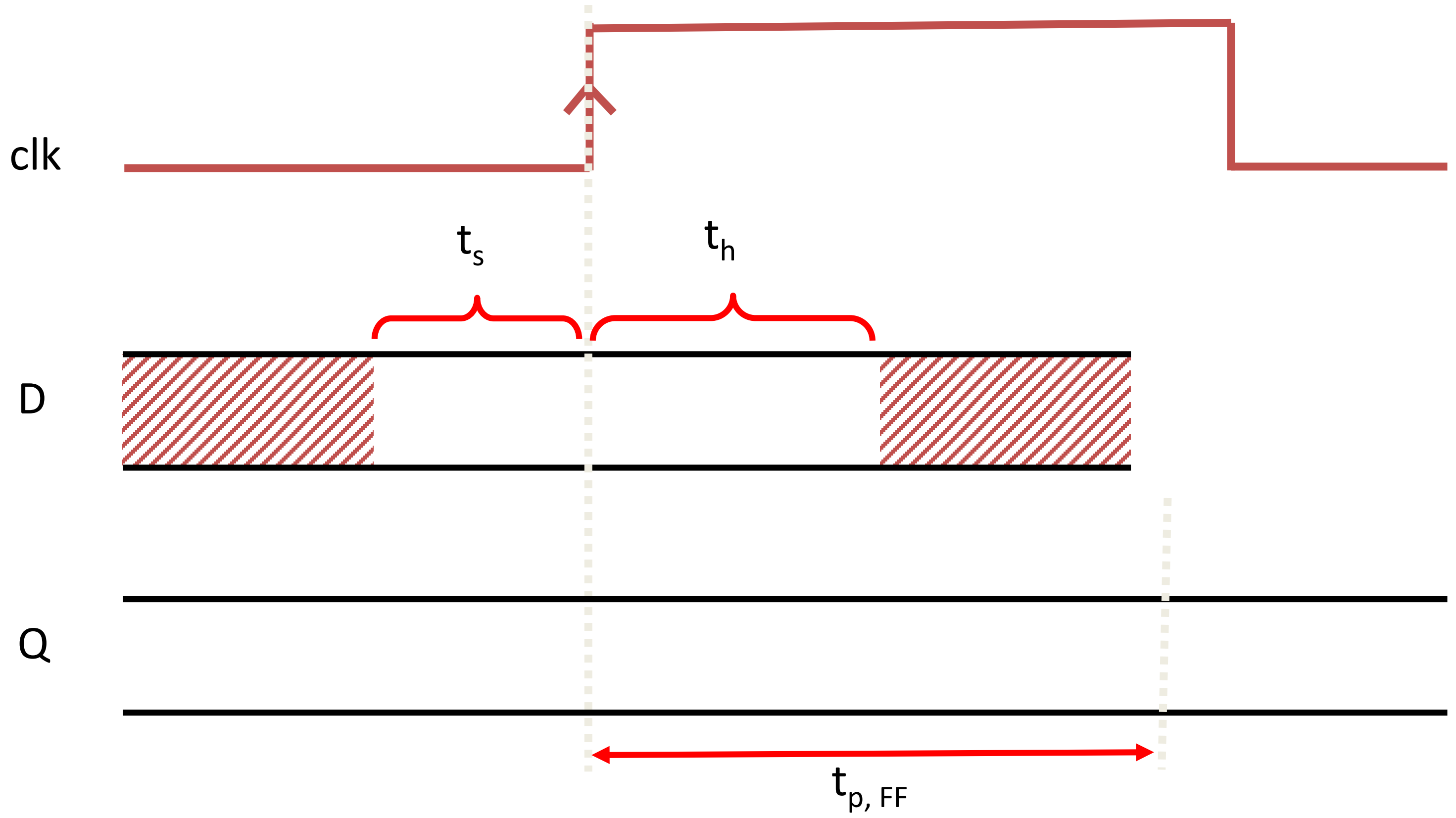
A(t+1) = A'BC

B(t+1) = A'(B ⊕ C)

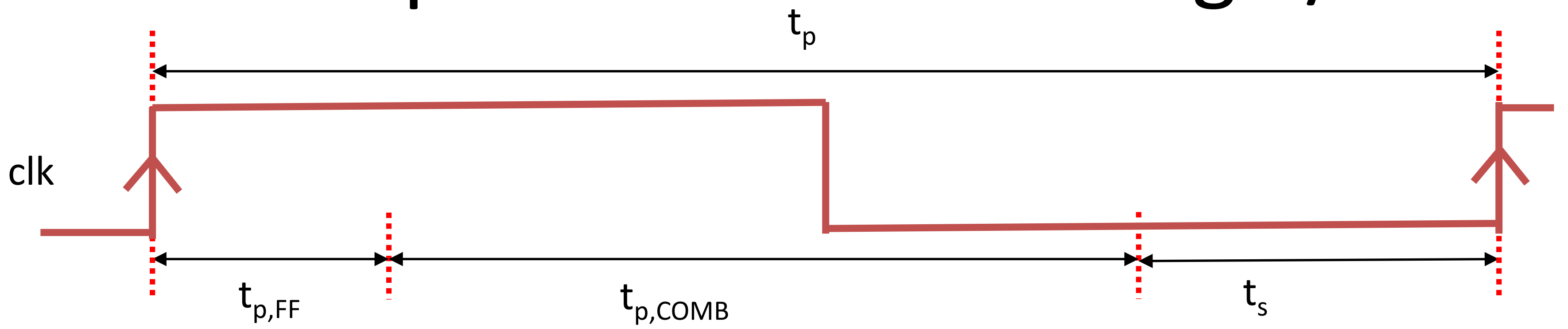
C(t+1) = A'C'

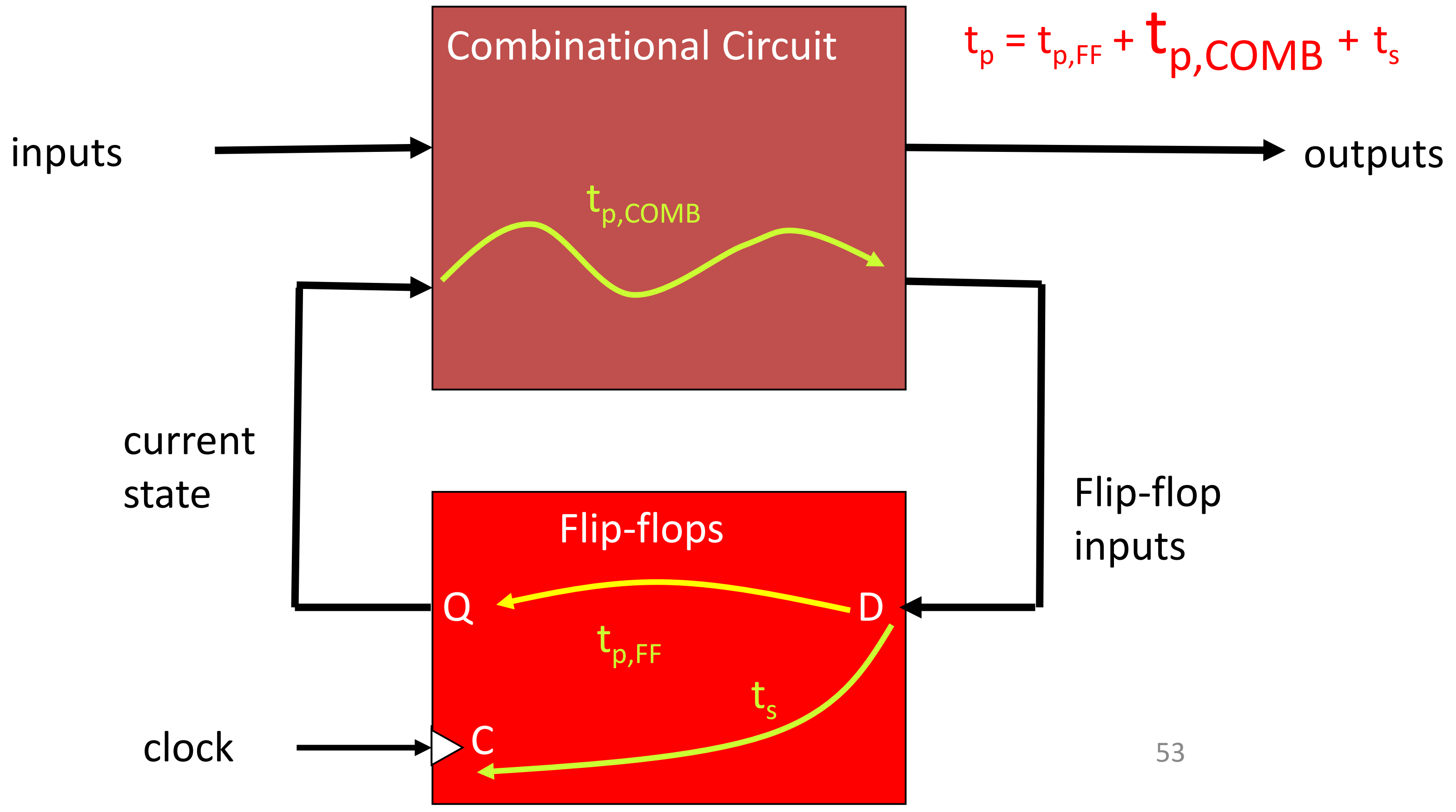
Sequential Circuit Timing 1/3

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



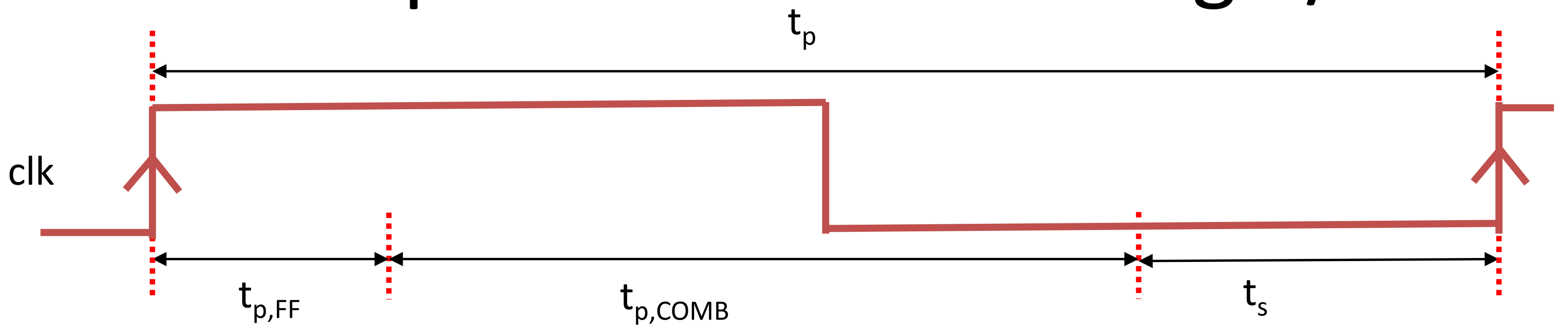
Sequential Circuit Timing 2/3



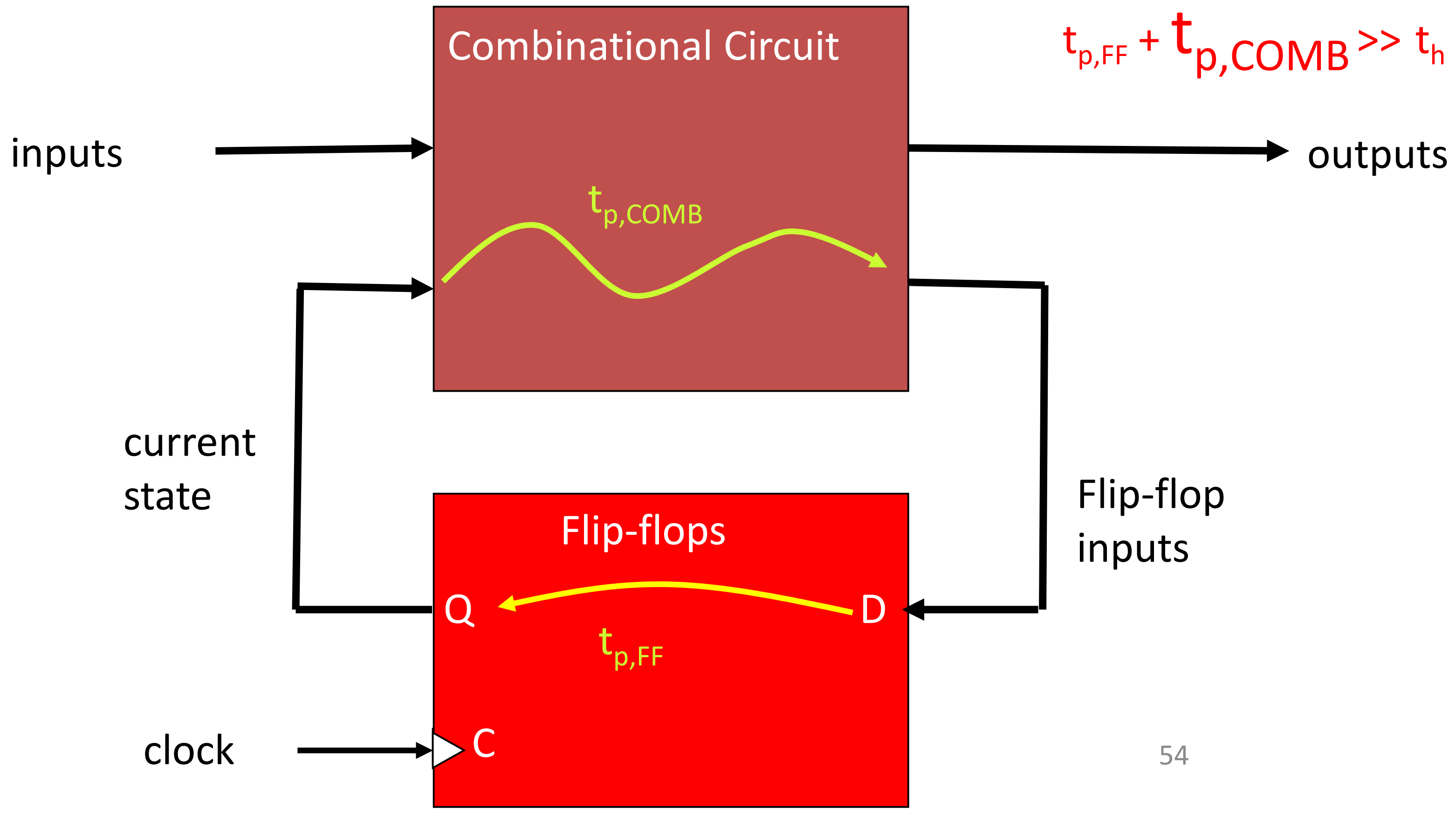


$t_p = t_{p,FF} + t_{p,COMB} + t_s$

Sequential Circuit Timing 2/3

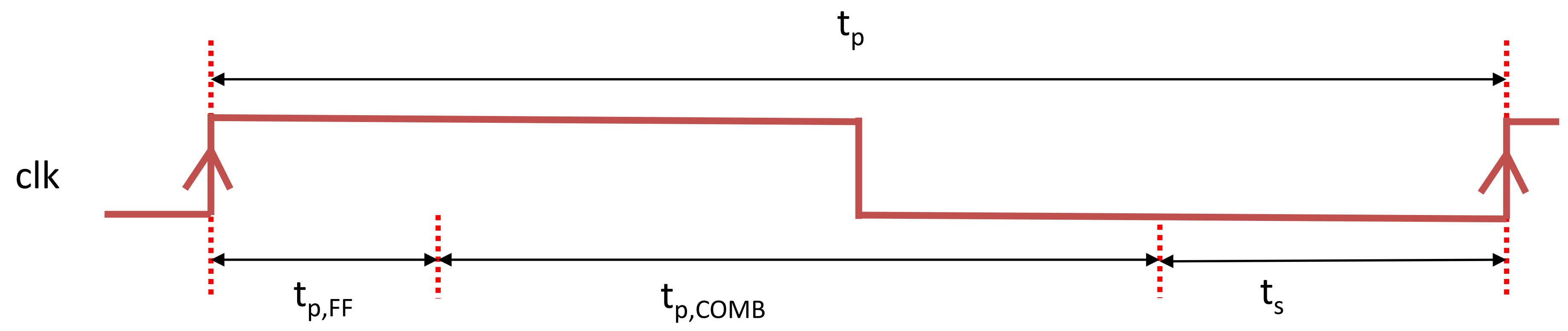


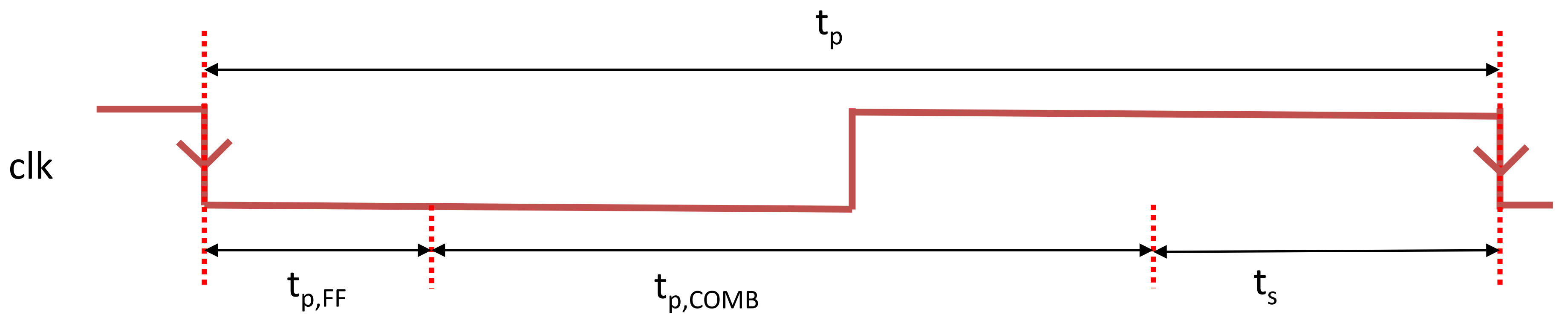
$t_{p,FF} + t_{p,COMB} >> t_h$



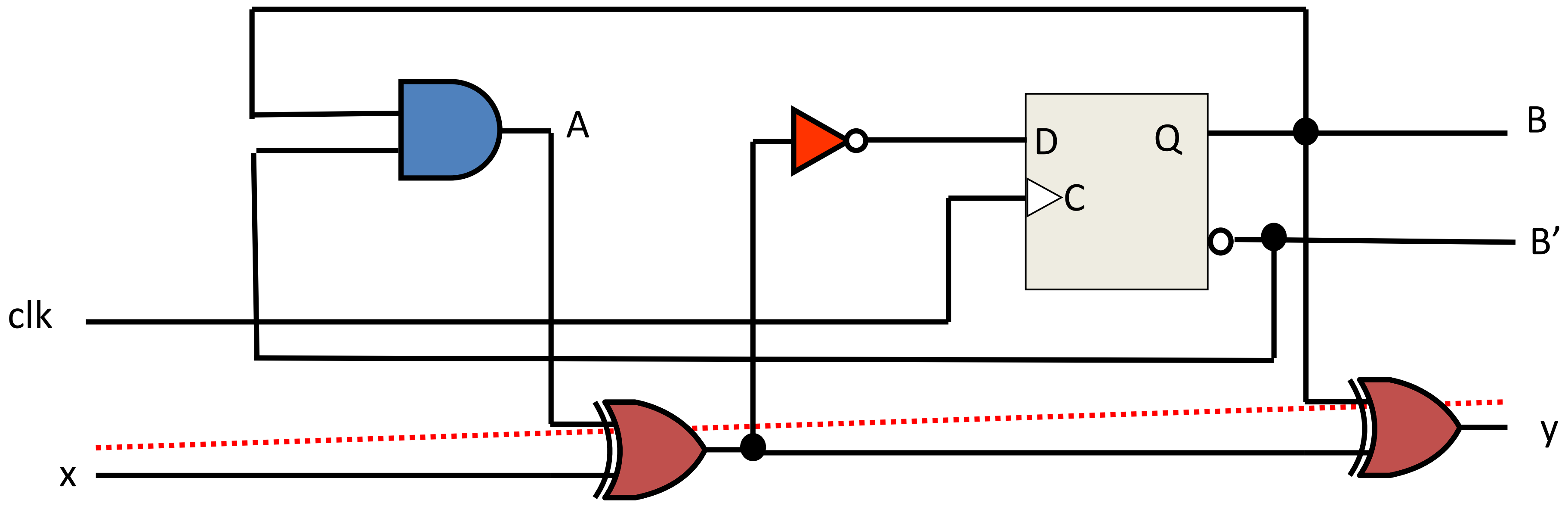
Sequential Circuit Timing 3/3

- Minimum clock period (or maximum clock frequency)





Example: Sequential Circuit Timing



$t_{p,NOT}$ = 0.5 ns

$t_{p,XOR}$ = 2.0 ns

$t_{p,FF}$ = 2.0 ns

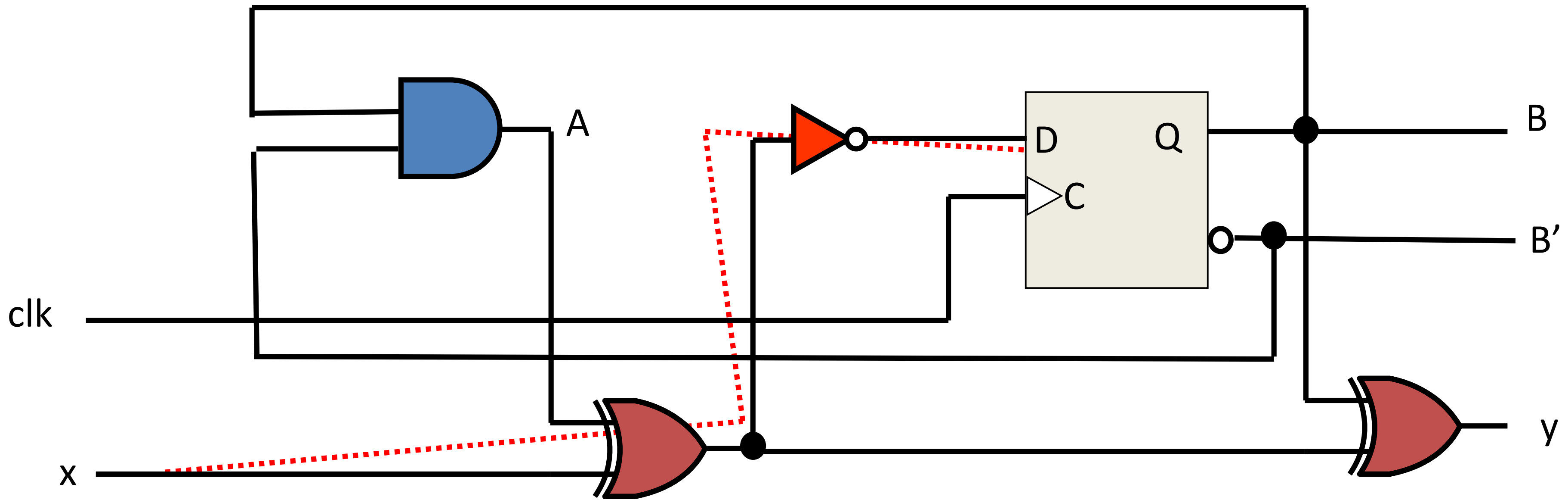
$t_{p,AND}$ = t_s = 1.0 ns

t_h = 0.25 ns

Find the longest path delay from external input to the output

$t_{p,XOR}$ + $t_{p,XOR}$ = 2.0 + 2.0 = 4.0 ns

Example: Sequential Circuit Timing



$t_{p,NOT}$ = 0.5 ns

$t_{p,XOR}$ = 2.0 ns

$t_{p,FF}$ = 2.0 ns

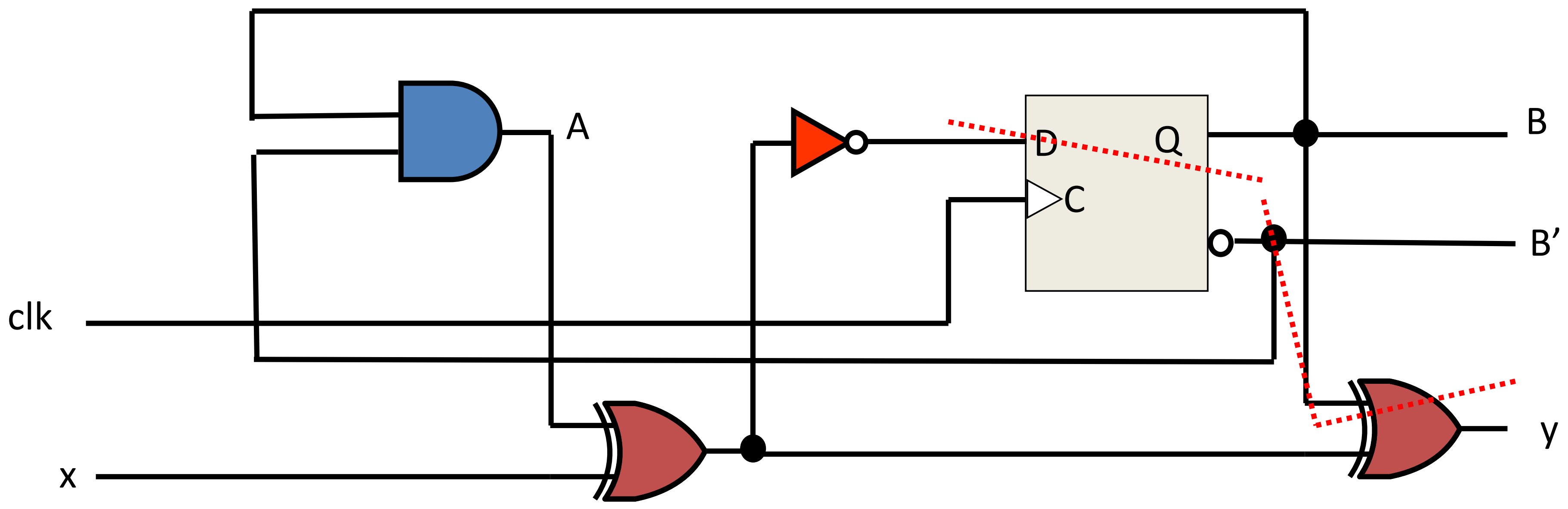
$t_{p,AND}$ = t_s = 1.0 ns

t_h = 0.25 ns

Find the longest path delay in the circuit from external input to positive clock edge

$t_{p,XOR} + t_{p,NOT}$ = 2.0 + 0.5 = 2.5 ns

Example: Sequential Circuit Timing



$t_{p,NOT}$ = 0.5 ns

$t_{p,XOR}$ = 2.0 ns

$t_{p,FF}$ = 2.0 ns

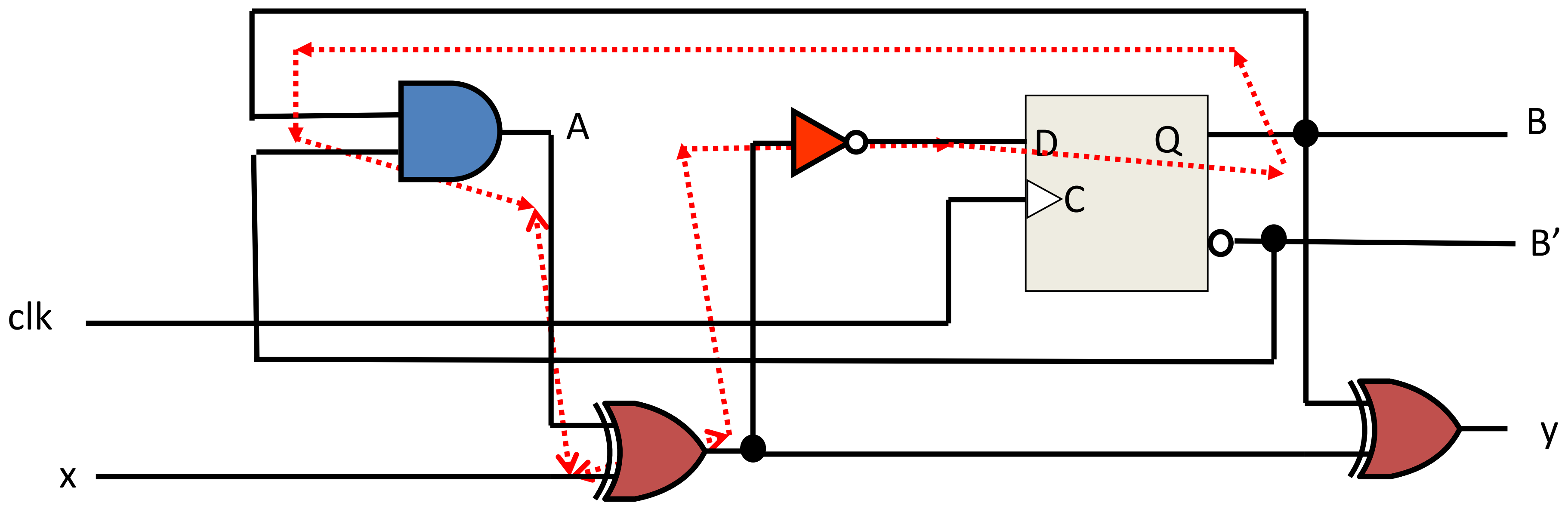
$t_{p,AND}$ = t_s = 1.0 ns

t_h = 0.25 ns

Find the longest path delay from positive clock edge to output

$t_{p,FF} + t_{p,XOR}$ = 2.0 + 2.0 = 4.0 ns

Example: Sequential Circuit Timing



$t_{p,NOT}$ = 0.5 ns

$t_{p,XOR}$ = 2.0 ns

$t_{p,FF}$ = 2.0 ns

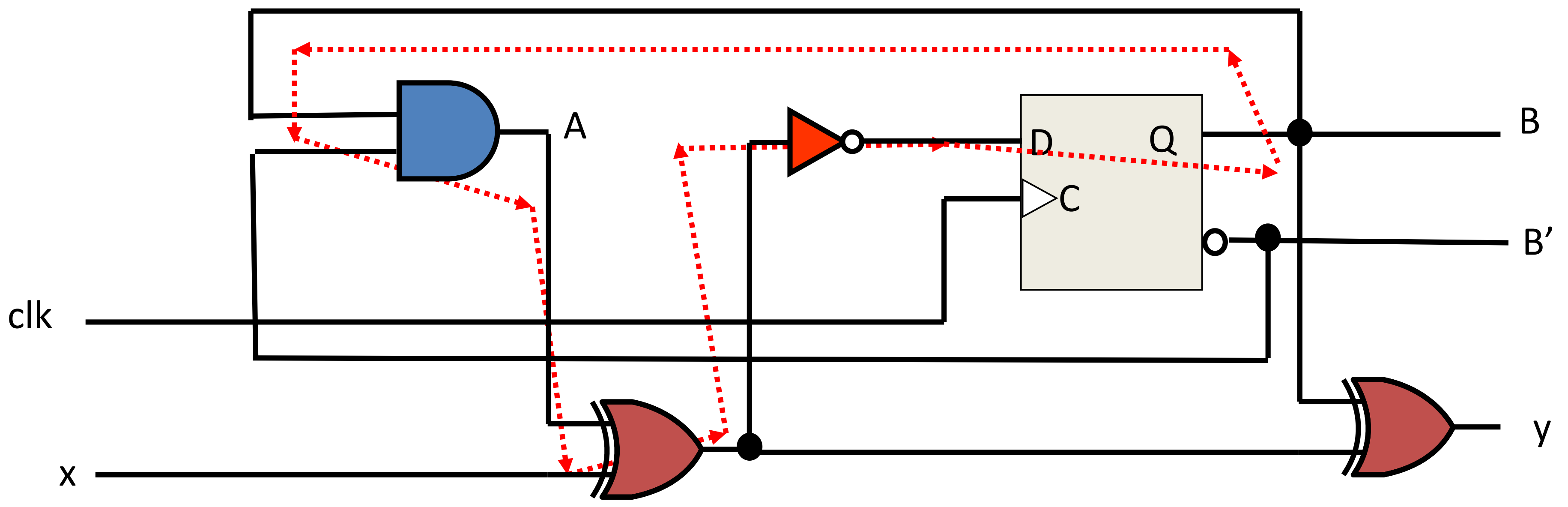
$t_{p,AND} = t_s$ = 1.0 ns

t_h = 0.25 ns

Find the longest path delay from positive clock edge to the flip-flop input

$t_{p,FF} + t_{p,AND} + t_{p,XOR} + t_{p,NOT}$
$= 2.0 + 1.0 + 2.0 + 0.5 = 5.5$ ns

Example: Sequential Circuit Timing



$t_{p,NOT}$ = 0.5 ns

$t_{p,XOR}$ = 2.0 ns

$t_{p,FF}$ = 2.0 ns

$t_{p,AND} = t_s$ = 1.0 ns

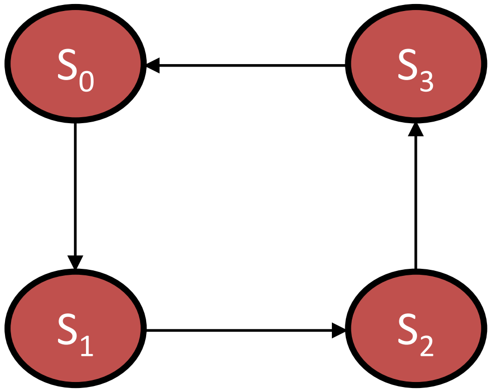
t_h = 0.25 ns

Determine the maximum frequency of operation of the circuit in megahertz

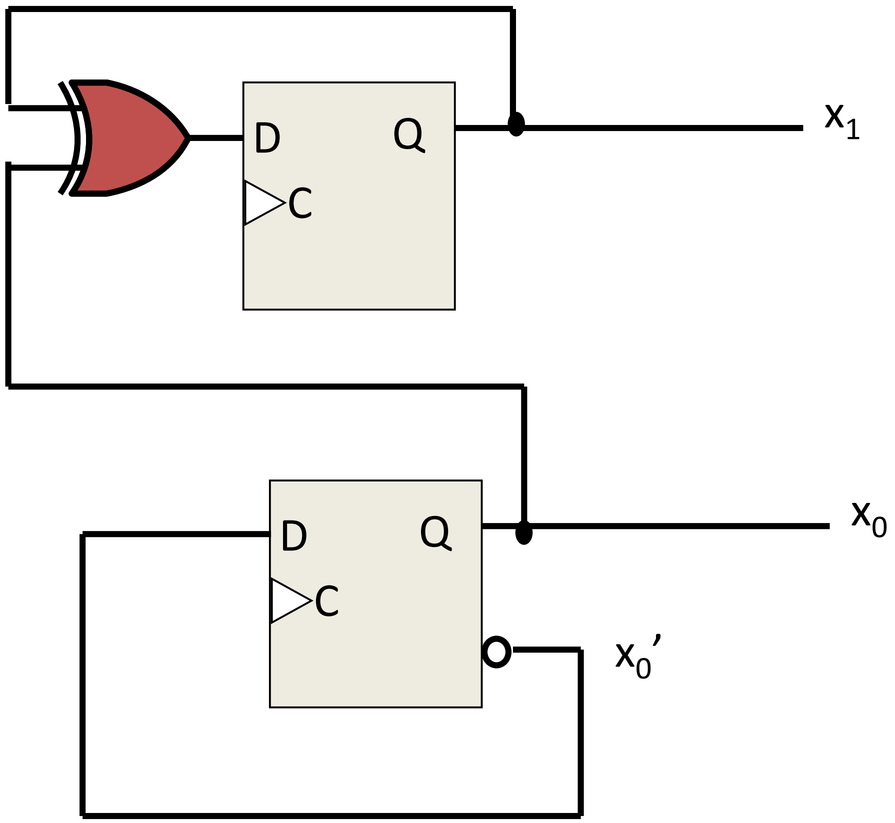
$$t_p = t_{p,FF} + t_{p,AND} + t_{p,XOR} + t_{p,NOT} + ?$$
$$= 2.0 + 1.0 + 2.0 + 0.5 + 1.0 = 6.5 \text{ ns}$$

$$f_{max} = 1/t_p = 1/(6.5\times10^{-9}) \approx 154 \text{ MHz}$$

Example



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

$f_{max} = 1/t_p = 1/(5.0\times10^{-9}) \approx 200$ MHz

Example: One-Hot-Encoding

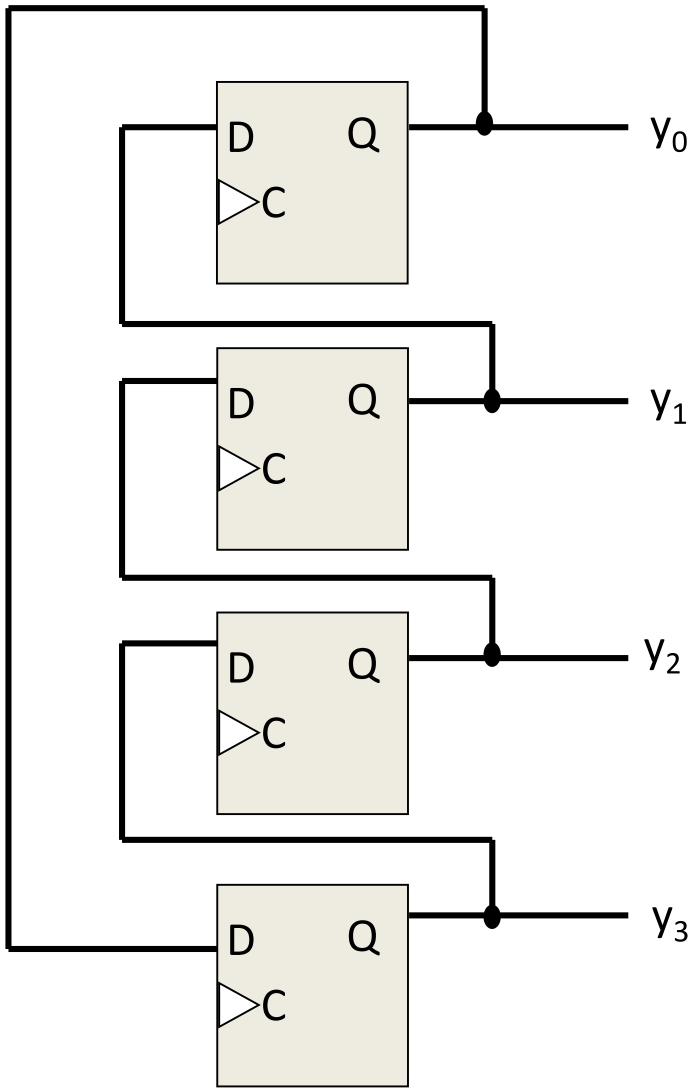
$S_0 \rightarrow$ `0001`
$S_1 \rightarrow$ `0010`
$S_2 \rightarrow$ `0100`
$S_3 \rightarrow$ `1000`

$t_{p,FF} = 2.0$ ns

$t_s = 1.0$ ns

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

$f_{max} = 1/t_p = 1/(3.0\times10^{-9}) \approx 333$ MHz


D
Q
C
y_0
D
Q
C
y_1
D
Q
C
y_2
D
Q
C
y_3