Sequential Circuit Design

State Optimization (Minimization)

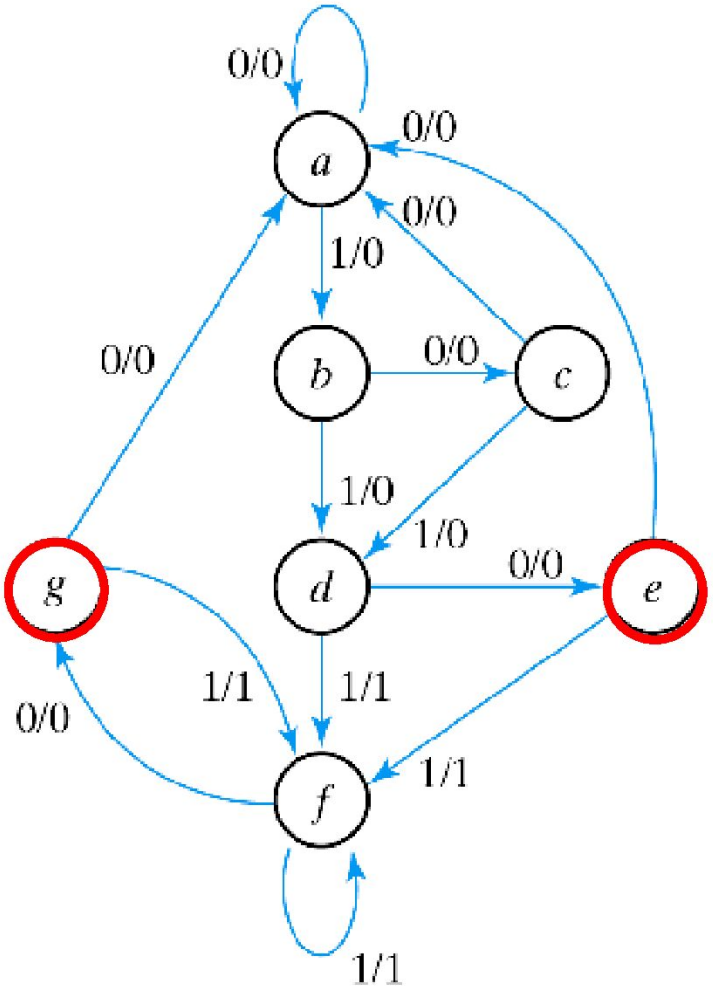
- **Equivalent States:**
  - Two states are equivalent if, for each member of the set of inputs,
    - they give exactly the same output and
    - send the circuit either to the same state or to an equivalent state
  - If two states are equivalent, one can be eliminated without affecting the behavior of the FSM.

State Optimization Algorithm: Row Matching

- Two states are equivalent if **any** of the conditions are true:
  1. If two states have the same output AND both transitions are to the same next state,
  2. If two states have the same output AND both transitions are to each other
  3. If two states have the same output AND both self-loop,

- Combine the equivalent states into a new renamed state
- Repeat until no more states can be combined

Row Matching by Example



State Transition Table

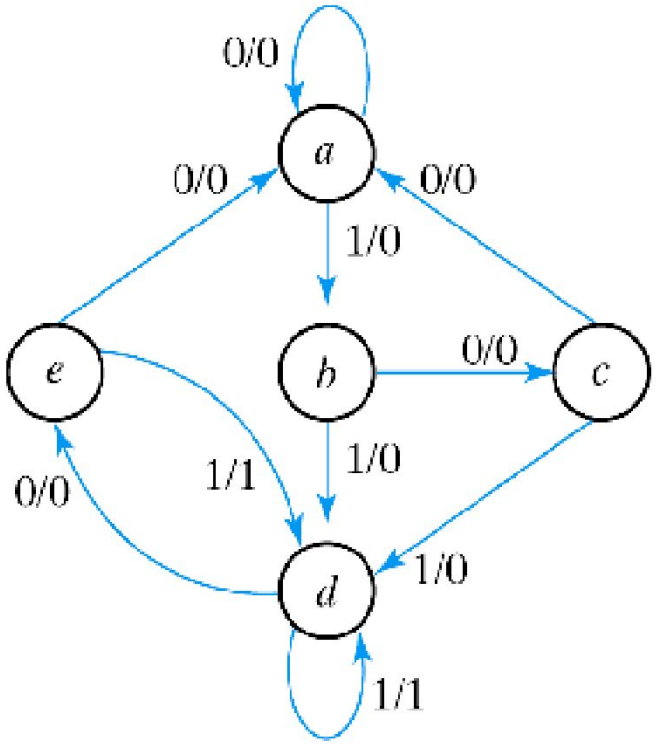
	NS		output	
PS	x=0	x=1	x=0	x=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

Row Matching Example (cont)

PS	NS x=0	NS x=1	output x=0	output x=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	e	f	0	1

PS	NS x=0	NS x=1	output x=0	output x=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1

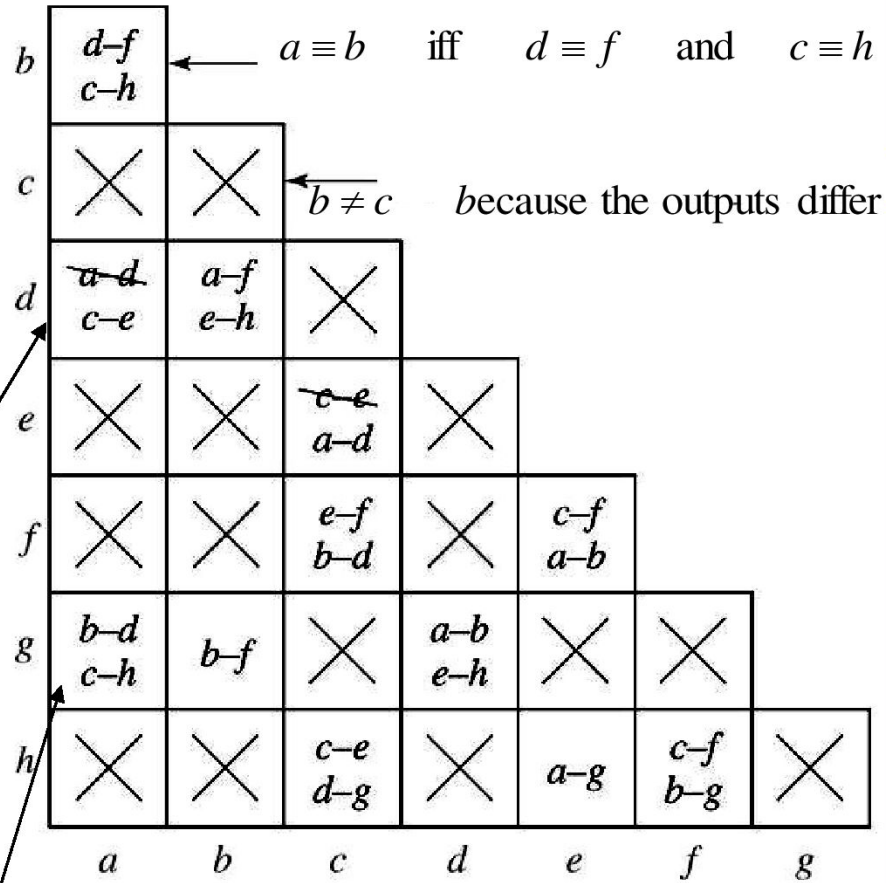
Reduced State Transition Diagram



State Optimization Algorithm

Implication Chart

Present State	Next State X=0	Next State 1	Output X=0
a	d	c	0
b	f	h	0
c	e	d	1
d	a	e	0
e	c	a	1
f	f	b	1
g	b	h	0
h	c	g	1

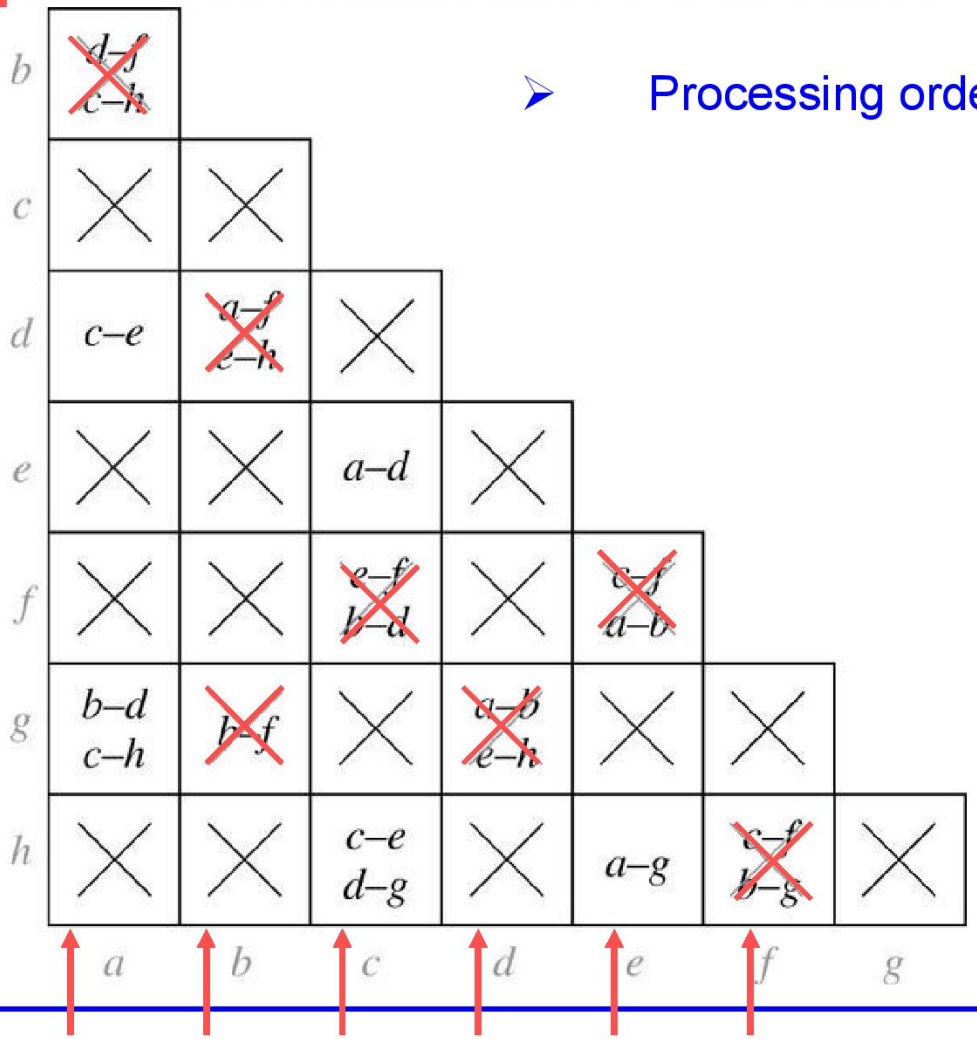


$a \equiv d$ iff $a \equiv d$ and $c \equiv e$

$a \equiv g$ iff $b \equiv d$ and $c \equiv h$

State Optimization Algorithm

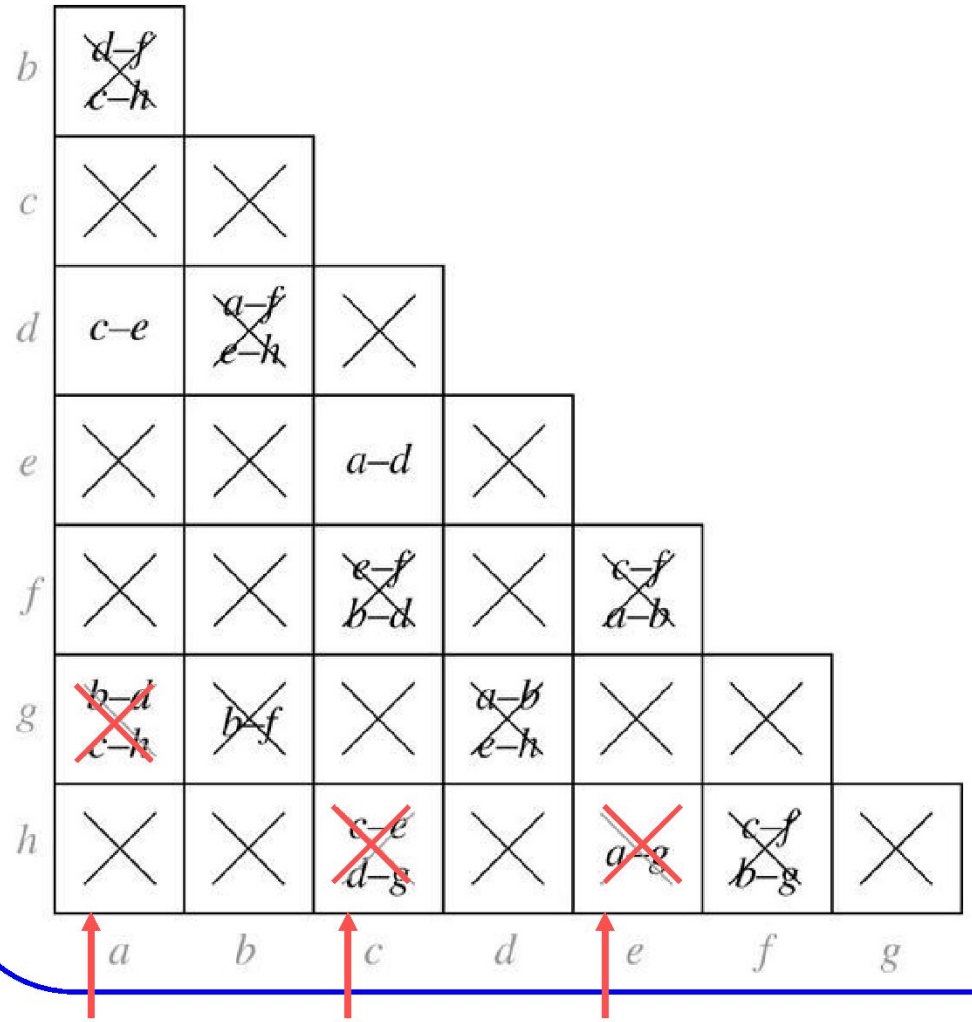
- **Implication Chart After First Pass**



- Processing order is important

State Optimization Algorithm

- **Implication Chart After Second Pass**



Present State	Next State X=0	Next State 1	Present Output X=0
a	d	c	0
b	f	h	0
c	e	d	1
d	a	e	0
e	c	a	1
f	f	b	1
g	b	h	0
h	c	g	1

Present State	Next State X=0	Next State 1	Output
a	a	c	0
b	f	h	0
c	c	a	1
f	f	b	1
g	b	h	0
h	c	g	1

Equivalent Sequential Circuits

- **Definition:**
  - Sequential circuit N1 is equivalent to sequential circuit N2 if
    - for each state p in N1, there is a state q in N2 such that $p \equiv q$, and
    - conversely, for each state s in N2, there is a state t in N1 such that $s \equiv t$.

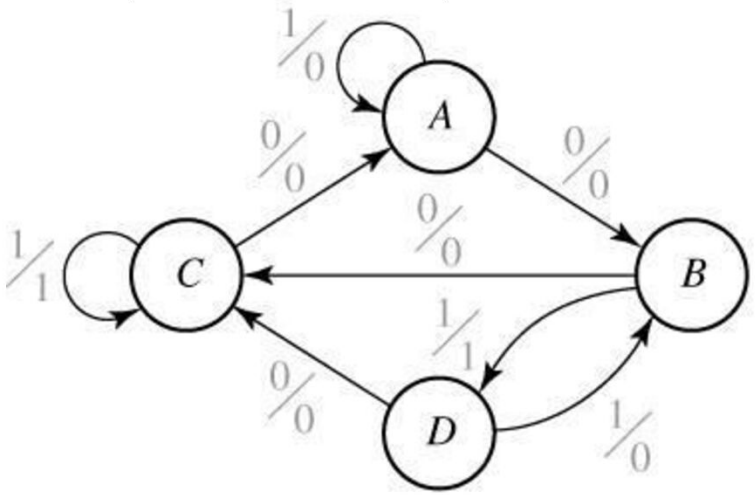
Example

N_1

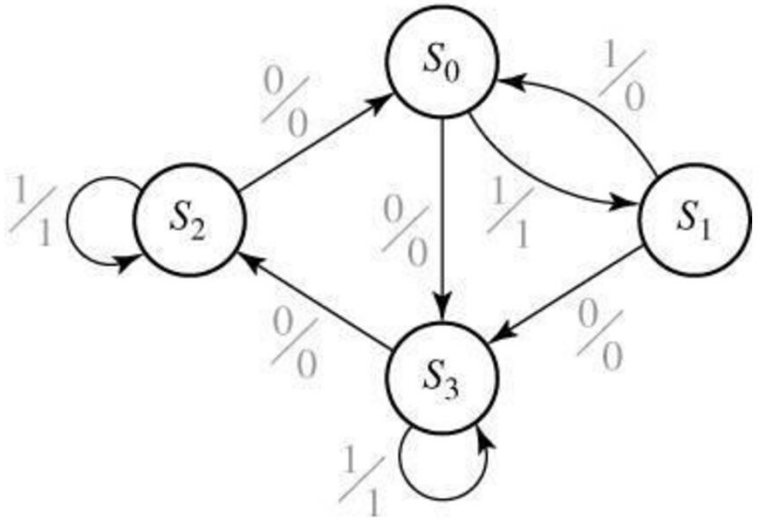
	X=0	X=1	X=0	X=1
A	B	A	0	0
B	C	D	0	1
C	A	C	0	1
D	C	B	0	0

N_2

	X=0	X=1	X=0	X=1
S_0	S_3	S_1	0	1
S_1	S_3	S_0	0	0
S_2	S_0	S_2	0	0
S_3	S_2	S_3	0	1



(a)



(b)

	X=0	X=1	X=0	X=1
A	B	A	0	0
B	C	D	0	1
C	A	C	0	1
D	C	B	0	0

	X=0	X=1	X=0	X=1
S_0	S_3	S_1	0	1
S_1	S_3	S_0	0	0
S_2	S_0	S_2	0	0
S_3	S_2	S_3	0	1

- **Implication Tables:**

	A	B	C	D
S_0	X	$C–S_3$ $D–S_1$	$A–S_3$ $C–S_1$	X
S_1	$B–S_3$ $A–S_0$	X	X	$C–S_3$ $B–S_0$
S_2	$B–S_0$ $A–S_2$	X	X	$C–S_0$ $B–S_2$
S_3	X	$C–S_2$ $D–S_3$	$A–S_2$ $C–S_3$	X

	A	B	C	D
S_0	X	$C–S_3$ $D–S_1$	~~$A–S_3$~~ ~~$C–S_1$~~	X
S_1	~~$B–S_3$~~ ~~$A–S_0$~~	X	X	$C–S_3$ $B–S_0$
S_2	$B–S_0$ ~~$A–S_2$~~	X	X	~~$C–S_0$~~ ~~$B–S_2$~~
S_3	X	~~$C–S_2$~~ ~~$D–S_3$~~	$A–S_2$ ~~$C–S_3$~~	X

$$A \equiv S_2 \quad B \equiv S_0 \quad C \equiv S_3 \quad D \equiv S_1$$

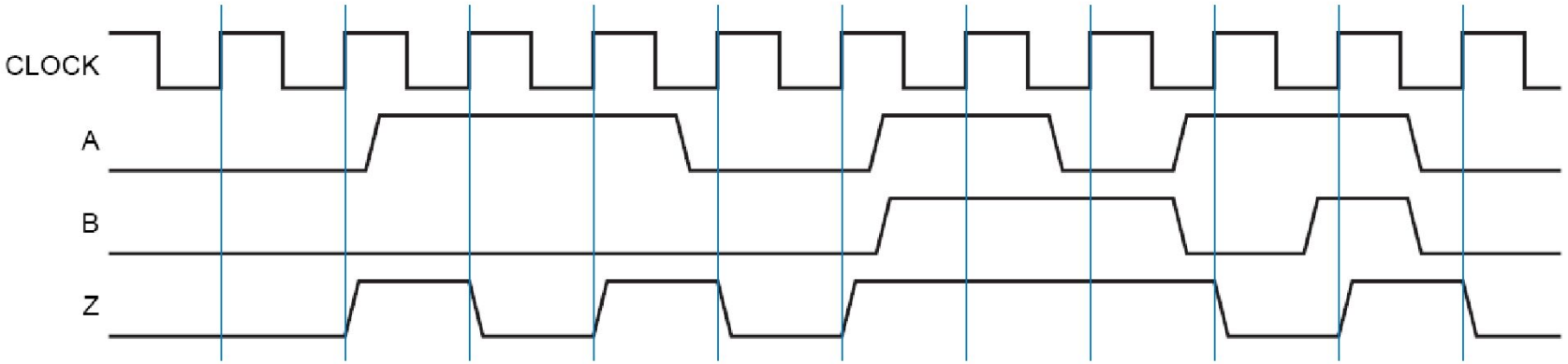
State Encoding or Assignment

- The simplest assignment of s coded states to 2^n possible states is to use the first s binary integers in binary counting order.
  - 000, 001, 010, 011, 100
- However, the simplest state assignment does not always lead to the simplest circuit.
  - In fact, the state assignment often has a major effect on circuit cost (as previously shown).
- In general, the only formal way to find the best assignment is to try all the assignments.
  - That's too much work!!
- Have to use some general guidelines (using the next example) …

Example

- Design a clocked synchronous state machine with two inputs, A and B, and a single output Z that is 1 if:
  - **A** had the same value at each of the two previous clock ticks, or
  - **B** has been 1 since the last time that the first condition was true.
  - Otherwise, the output should be 0.

Timing Diagram



Moore Machine

A0: got 0 on A

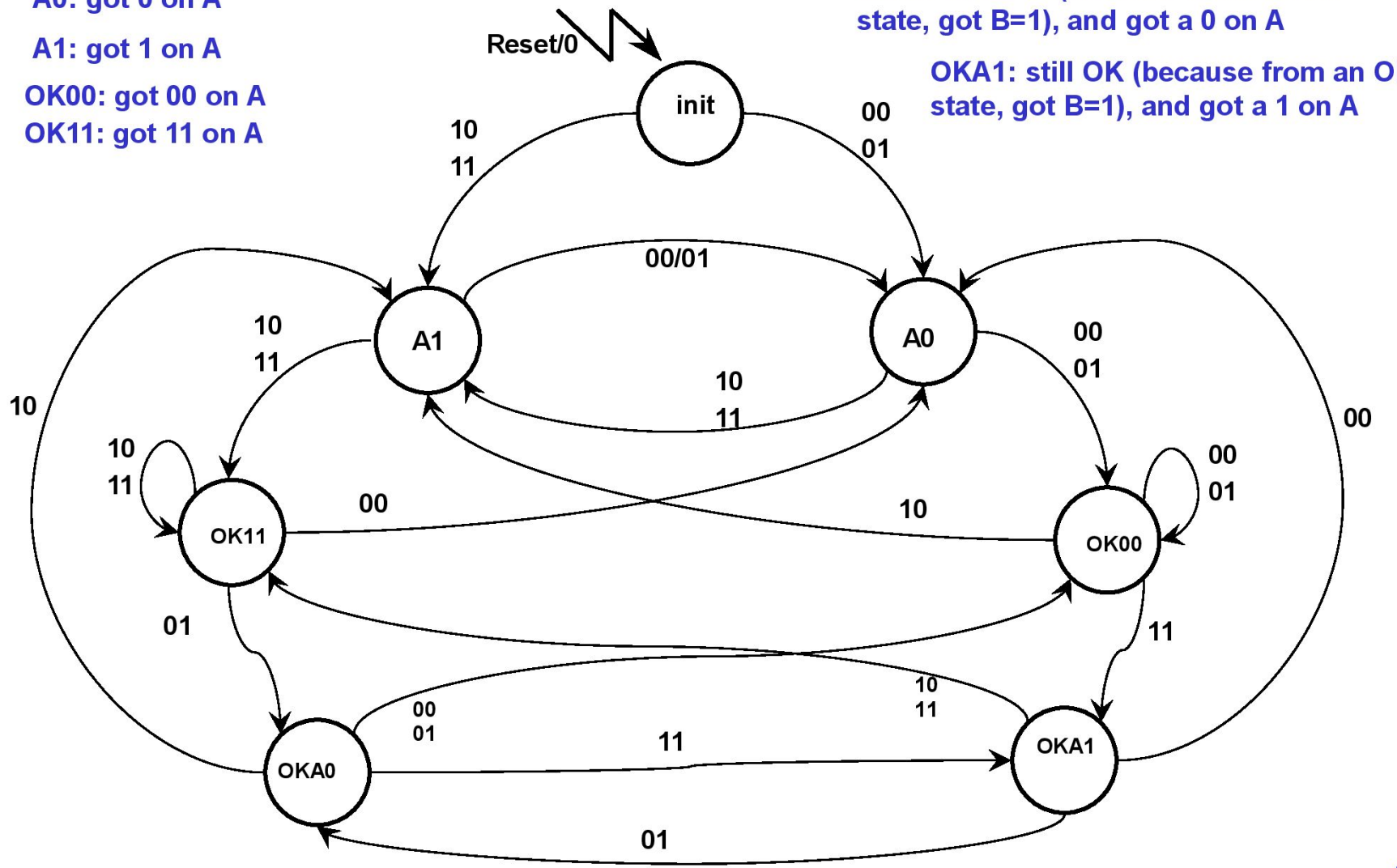
A1: got 1 on A

OK00: got 00 on A

OK11: got 11 on A

OKA0: still OK (because from an OK state, got B=1), and got a 0 on A

OKA1: still OK (because from an OK state, got B=1), and got a 1 on A



State Table

Meaning	S	A B: 00	01	11	10	Z
Initial state	INIT	A0	A0	A1	A1	0
Got a 0 on A	A0	OK00	OK00	A1	A1	0
Got a 1 on A	A1	A0	A0	OK11	OK11	0
Got 00 on A	OK00	OK00	OK00	OKA1	A1	1
Got 11 on A	OK11	A0	OKA0	OK11	OK11	1
OK, got a 0 on A	OKA0	OK00	OK00	OKA1	A1	1
OK, got a 1 on A	OKA1	A0	OKA0	OK11	OK11	1
				S*		

- OK00 and OKA0 are equivalent
- OK11 and OKA1 are equivalent

	A B				
S	***00***	***01***	***11***	***10***	***Z***
INIT	A0	A0	A1	A1	0
A0	OK0	OK0	A1	A1	0
A1	A0	A0	OK1	OK1	0
OK0	OK0	OK0	OK1	A1	1
OK1	A0	OK0	OK1	OK1	1
			S*		

Moore Machine

A0: got 0 on A

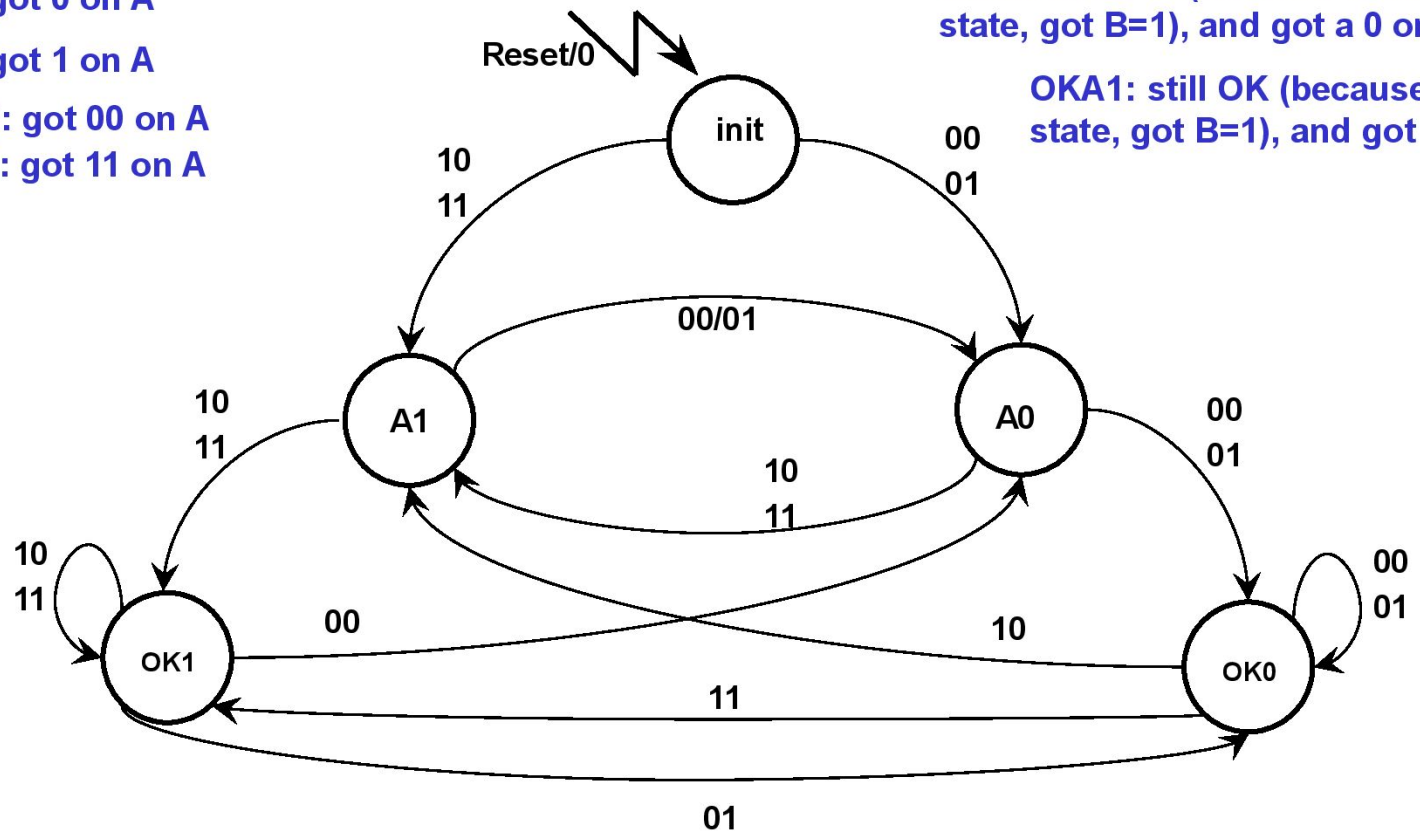
A1: got 1 on A

OK00: got 00 on A

OK11: got 11 on A

OKA0: still OK (because from an OK state, got B=1), and got a 0 on A

OKA1: still OK (because from an OK state, got B=1), and got a 1 on A



State Encoding

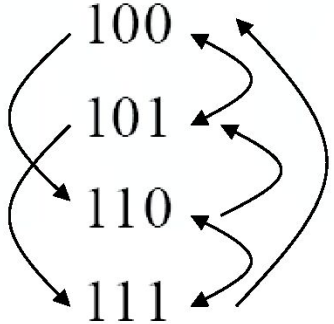
- **Guidelines:**
  - Choose an initial coded state into which the machine can easily be forced at reset
    - 00. . . 00 or 11. . . 11 in typical circuits.
  - Minimize the number of state variables that change on each transition.
  - Maximize the number of state variables that don't change in a group of related states
    - i.e., a group of states in which most of transitions stay within the group.

State Encoding

- **Example:**

	A B				
S	***00***	***01***	***11***	***10***	***Z***
INIT	A0	A0	A1	A1	0
A0	OK0	OK0	A1	A1	0
A1	A0	A0	OK1	OK1	0
OK0	OK0	OK0	OK1	A1	1
OK1	A0	OK0	OK1	OK1	1
			S*		

State name	***Simplest Q1–Q3***	***Decomposed Q1–Q3***
INIT	000	000
A0	001	100
A1	010	101
OK0	011	110
OK1	100	111



Synthesis Using D FF

Present State	Next State				Output
	A B				
Q1 Q2 Q3	*00*	*01*	*11*	*10*	*Z*
000	100	100	101	101	0
100	110	110	101	101	0
101	100	100	111	111	0
110	110	110	111	101	1
111	100	110	111	111	1
	Q1* Q2* Q3*				

A B			
00	*01*	*11*	*10*
100	100	101	101
110	110	101	101
100	100	111	111
110	110	111	101
100	110	111	111
D1 D2 D3			

State Assignment

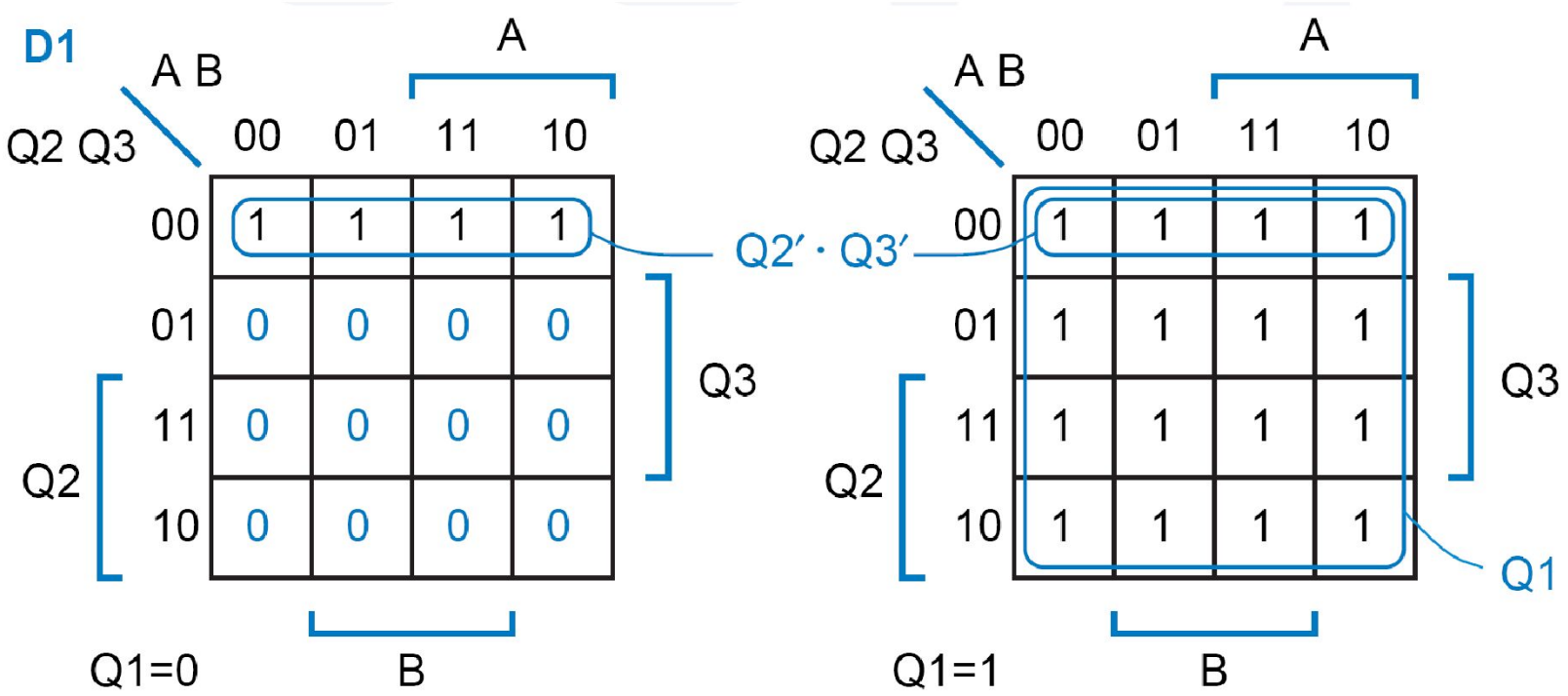
- **Unused States:**
  - Each of the *m* states must be assigned a unique code
  - Minimum number of bits required is *n* such that
    $n \geq \lceil \log_2 m \rceil$
    where $\lceil x \rceil$ is the smallest integer $\geq x$
  - There may be 2^n - m unused states

State Assignment

- **Unused States:**
  - Minimal risk:
    - Assumes that it is possible for the state machine to somehow get into one of the unused (or “illegal”) states,
      - (due to a hardware failure, an unexpected input).
    - Therefore, all unused states go to the “initial” state
  - Minimal cost:
    - Assumes that the machine will never enter an unused state.
    - Therefore, the next-state entries of the unused states are marked as “don’t-cares”.

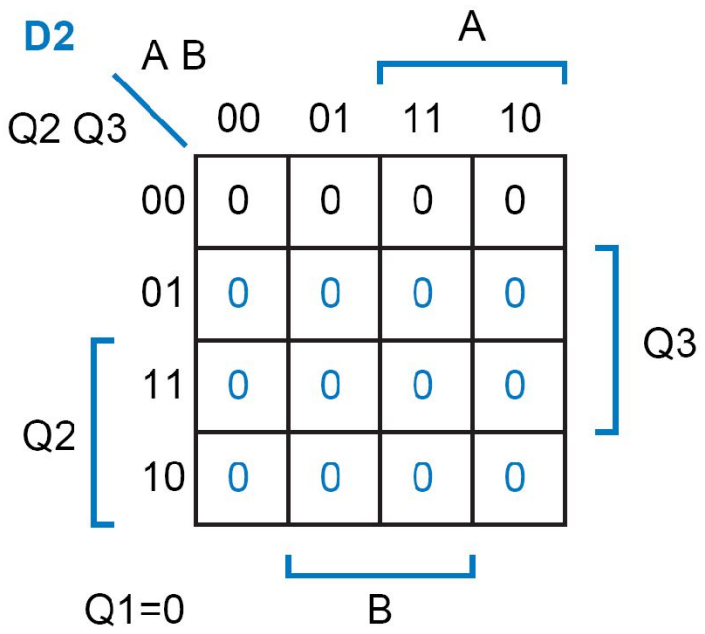
Case 1: Minimal Risk

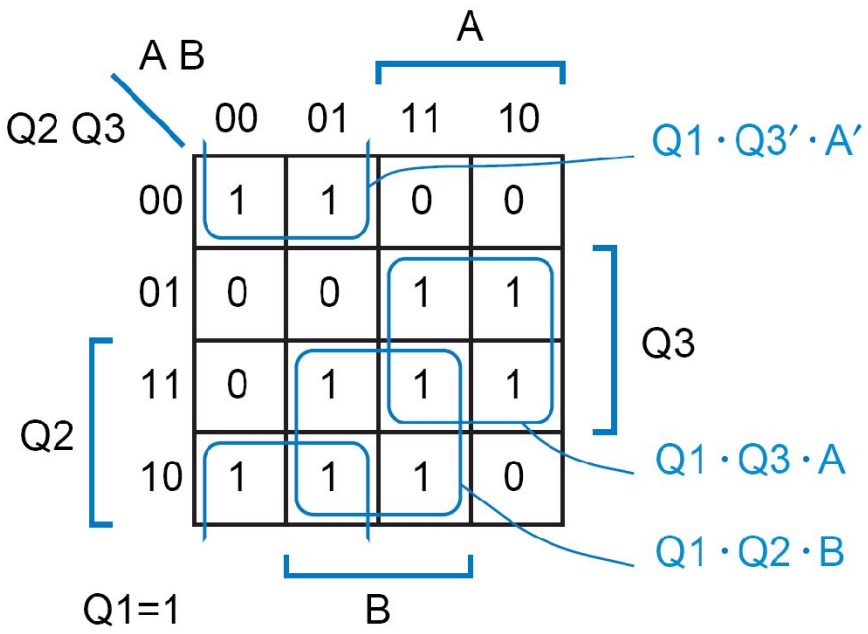
- **Assumption:**
  - Unused states go to 000 (minimal risk)



$$D1 = Q1 + Q2'.Q3'$$

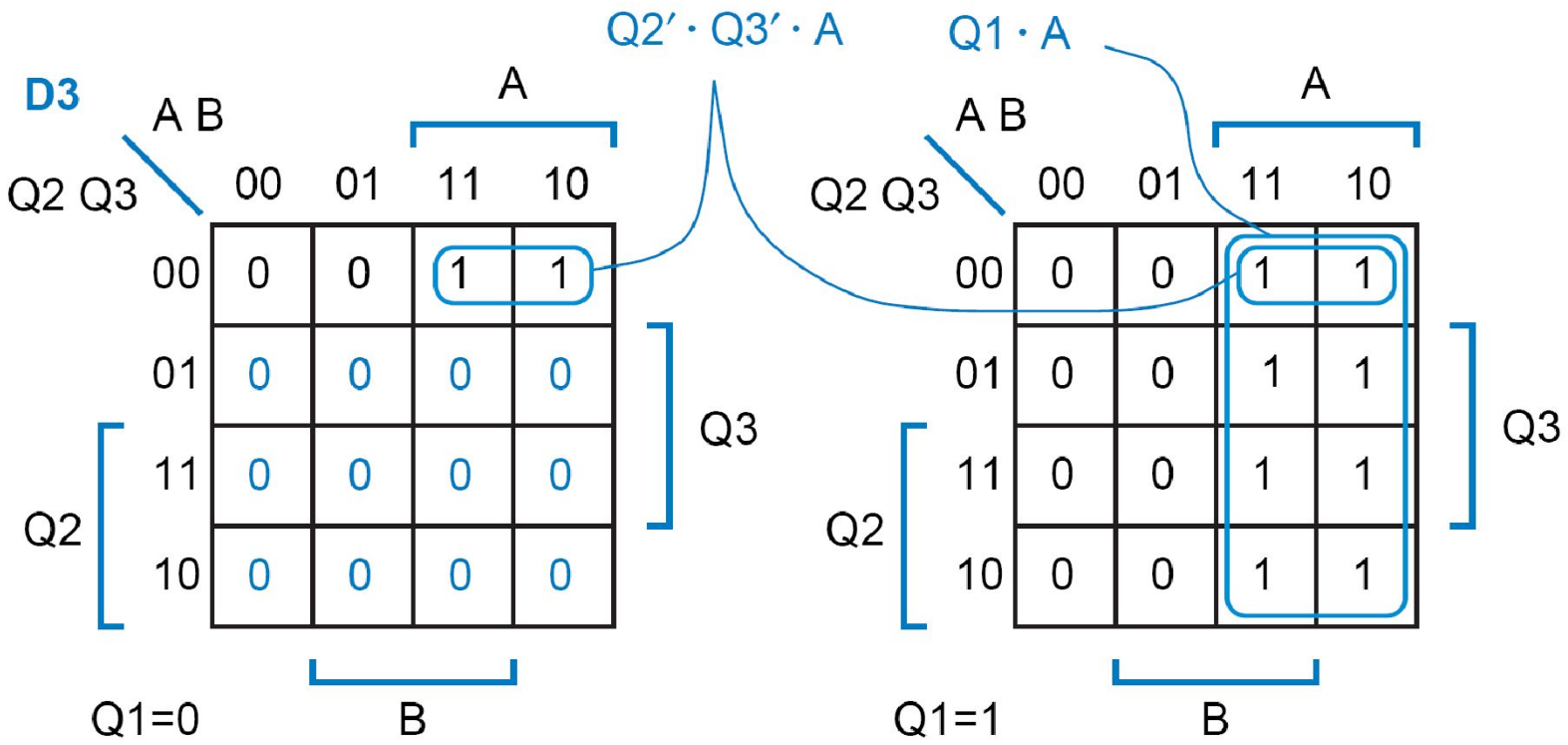
Case 1: D2 Equation





$$D2 = Q1.Q3'.A' + Q1.Q3.A + Q1.Q2.B$$

Case 1: D3 Equation



$$D3 = Q1.A + Q2'.Q3'.A$$

Case 1: Z Equation

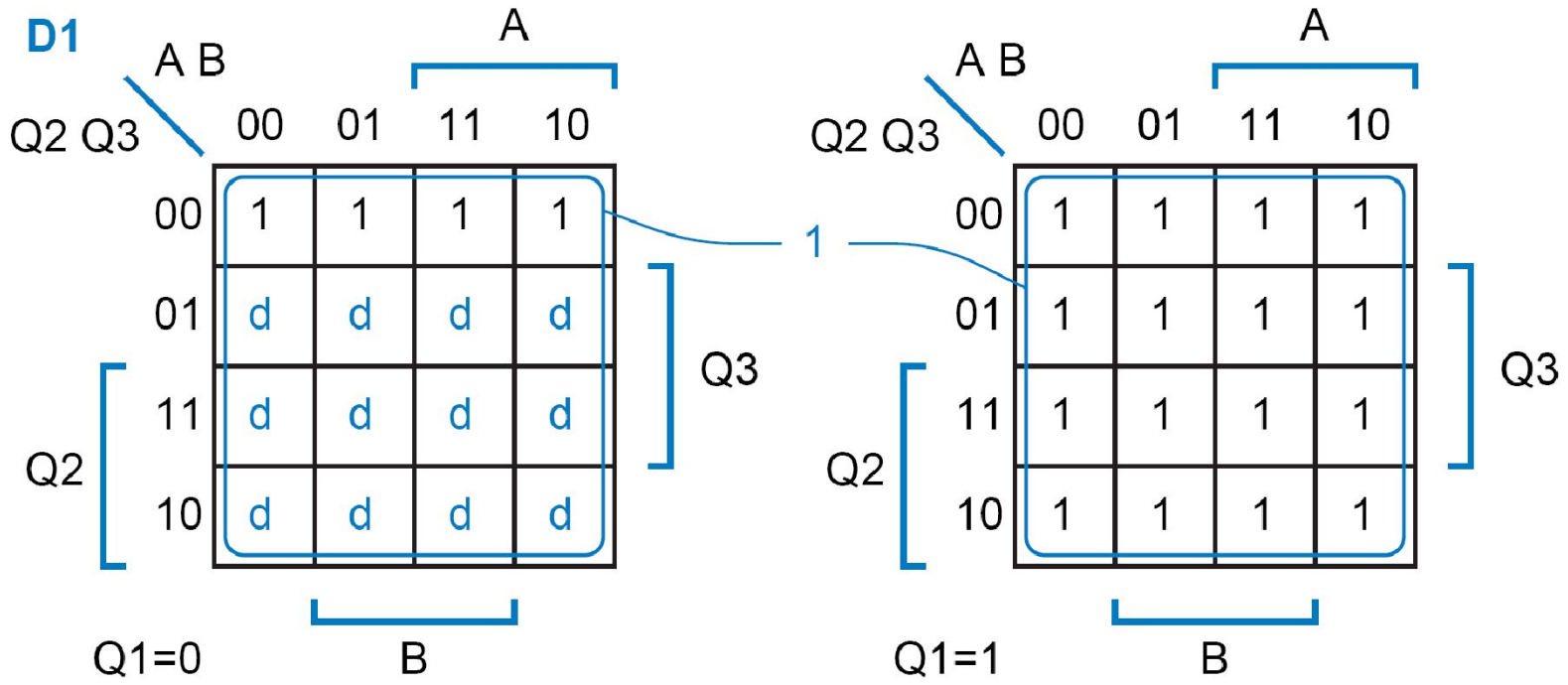
	AB				
Q1 Q2 Q3	00	01	11	10	Z
000	100	100	101	101	0
100	110	110	101	101	0
101	100	100	111	111	0
110	110	110	111	101	1
111	100	110	111	111	1
	Q1* Q2* Q3*				

$$Z = Q1.Q2.Q3' + Q1.Q2.Q3$$

$$= Q1.Q2$$

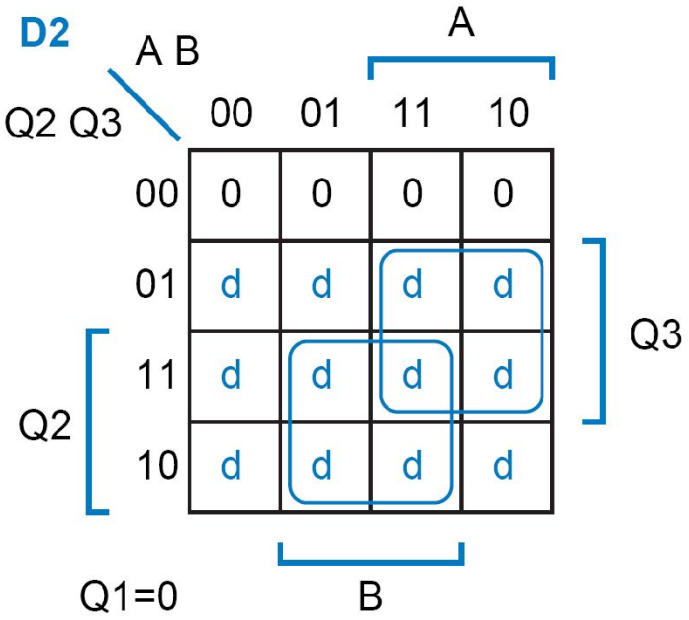
Case 2: Minimal Cost

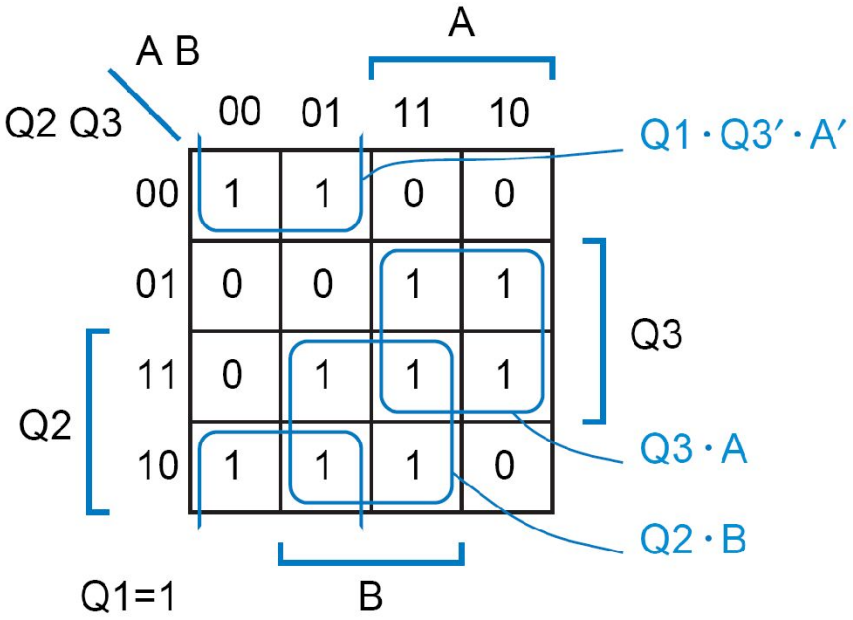
- **Assumption:**
  - Next states of unused states are “don’t-cares” (minimal cost)
  - → Z = don’t care for unused states



$$D1 = 1$$

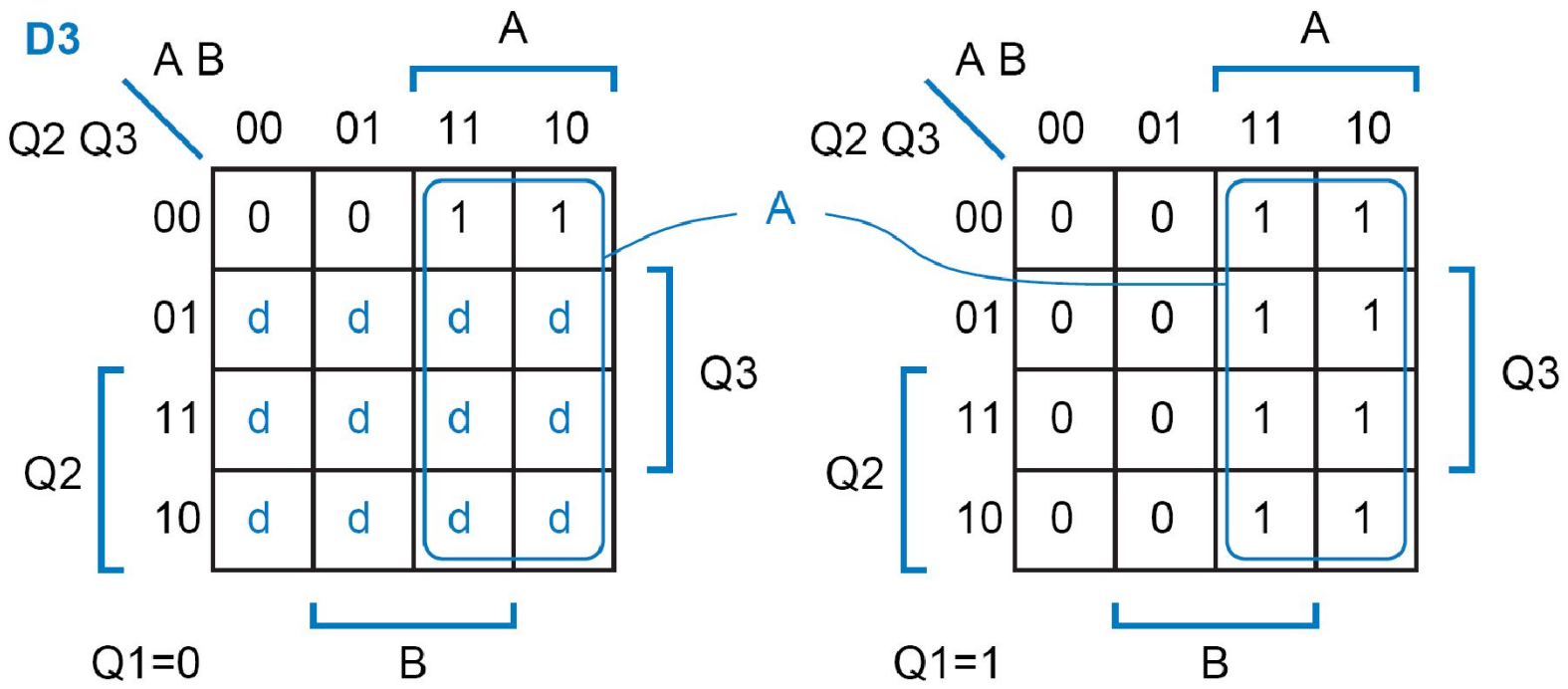
Case 2: D2 Equation





$$D2 = Q1.Q3'.A' + Q3.A + Q2.B$$

Case 2: D3 Equation



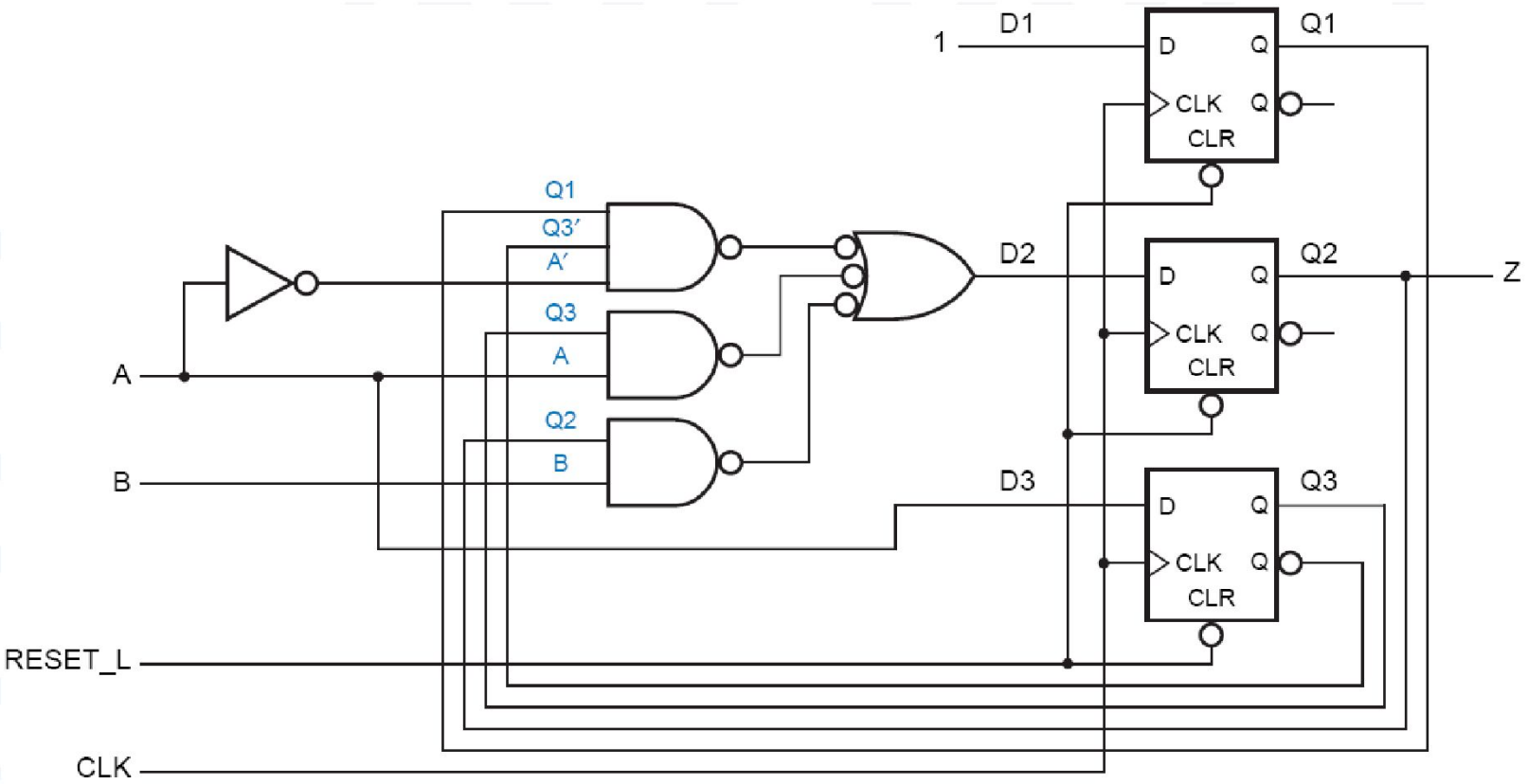
$$D3 = A$$

Case 2: Z Equation

	A B				
Q1 Q2 Q3	00	01	11	10	Z
000	100	100	101	101	0
100	110	110	101	101	0
101	100	100	111	111	0
110	110	110	111	101	1
111	100	110	111	111	1
	Q1* Q2* Q3*				

$$Z = Q2$$

Case 2: Logic Diagram



State Assignment Comparison

- **Logic used:**
  - Minimal risk:
    - 25 input-gates (excluding inverters)
  - Minimal cost:
    - 10 input-gates (excluding inverters)

Case 3: Synthesis Using JK FF

- **Characteristic Table:**

J	K	Q(t+1)	Operation
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$\overline{Q}(t)$	Complement

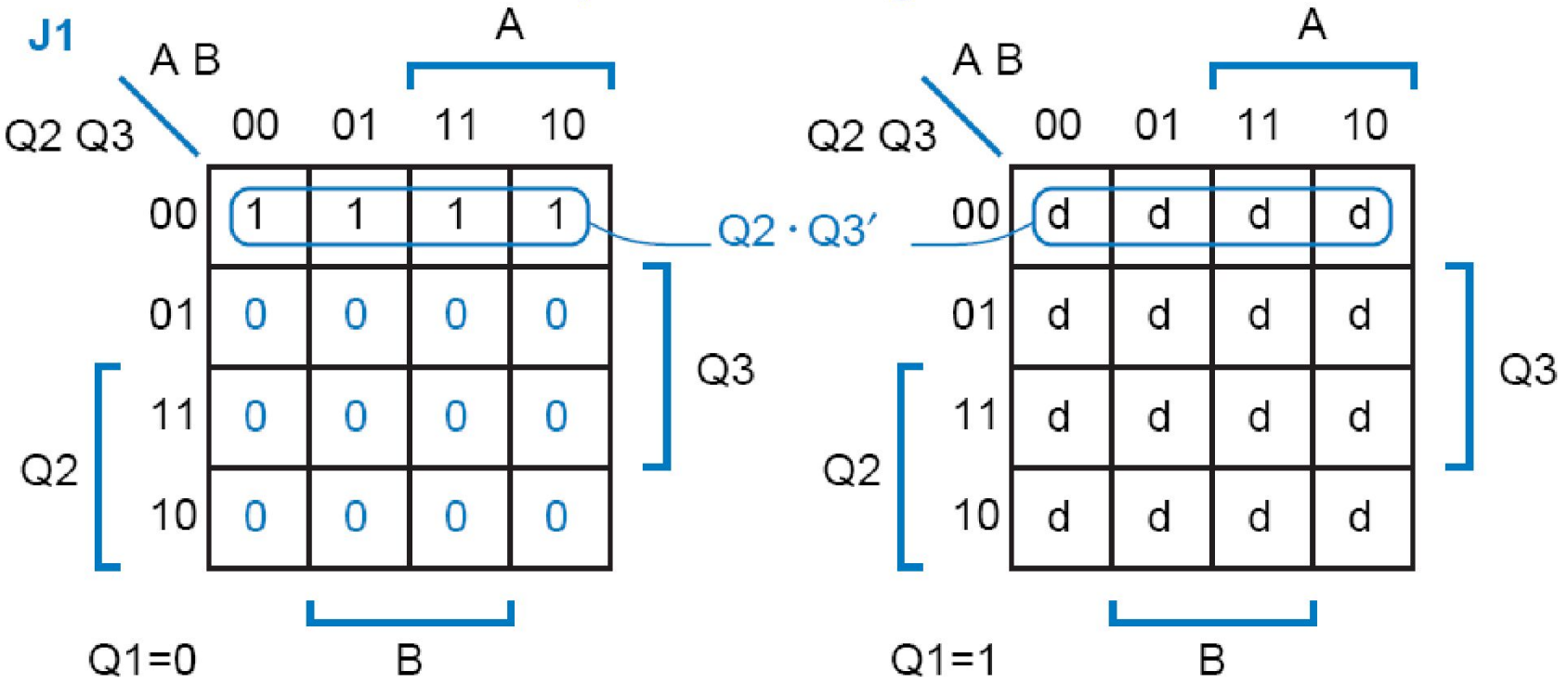
- **Excitation Table:**

Q(t)→Q(t+1)		J	K	Operation
0	0	0	X	No change
0	1	1	X	Set
1	0	X	1	Reset
1	1	X	0	No Change

	A B				
Q1 Q2 Q3	00	01	11	10	Z
000	1d, 0d, 0d	1d, 0d, 0d	1d, 0d, 1d	1d, 0d, 1d	0
100	d0, 1d, 0d	d0, 1d, 0d	d0, 0d, 1d	d0, 0d, 1d	0
101	d0, 0d, d1	d0, 0d, d1	d0, 1d, d0	d0, 1d, d0	0
110	d0, d0, 0d	d0, d0, 0d	d0, d0, 1d	d0, d1, 1d	1
111	d0, d1, d1	d0, d0, d1	d0, d0, d0	d0, d0, d0	1
	J1 K1, J2 K2, J3 K3				

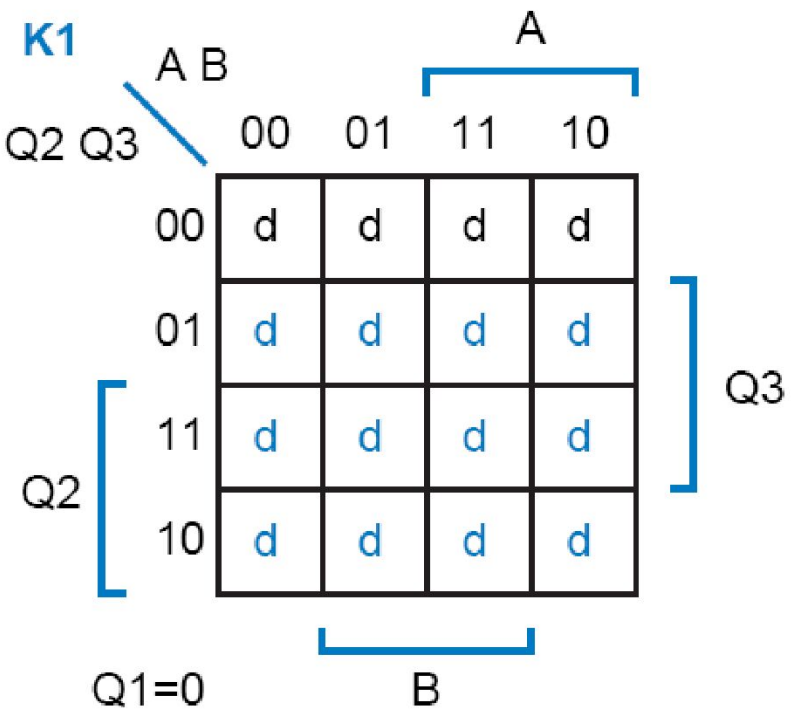
Case 3: Synthesis Using JK FF

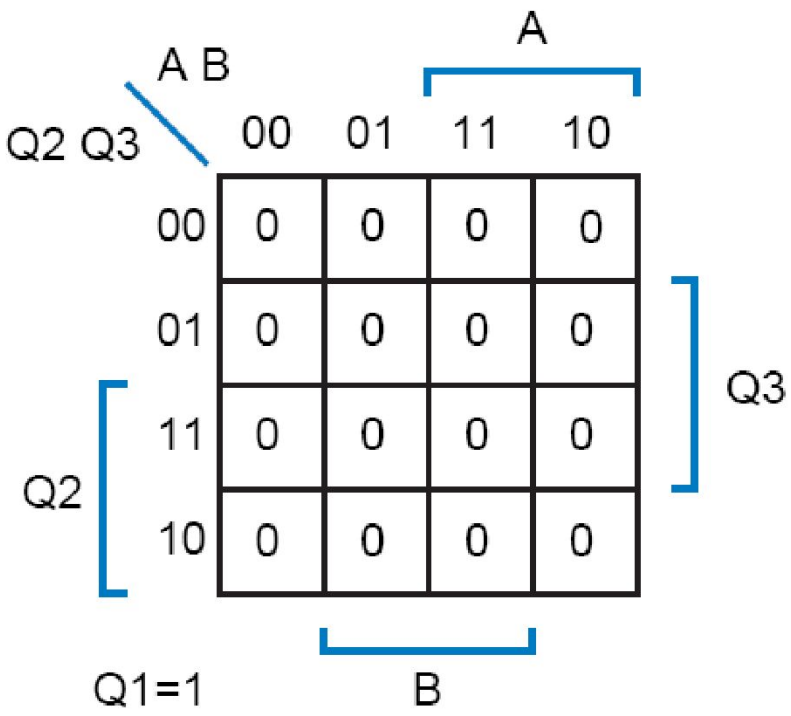
- Needs separate logic for J and K.



$$J1 = Q2'.Q3'$$

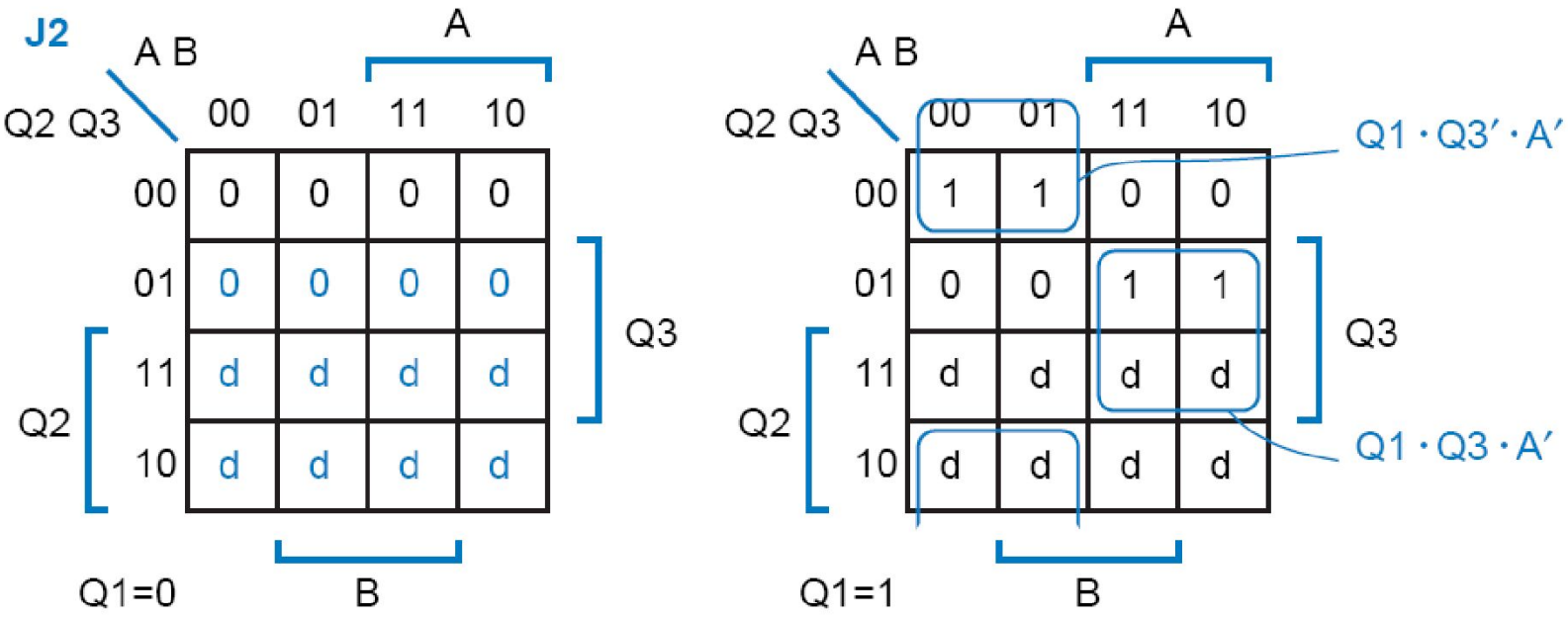
Case 3: Synthesis Using JK FF





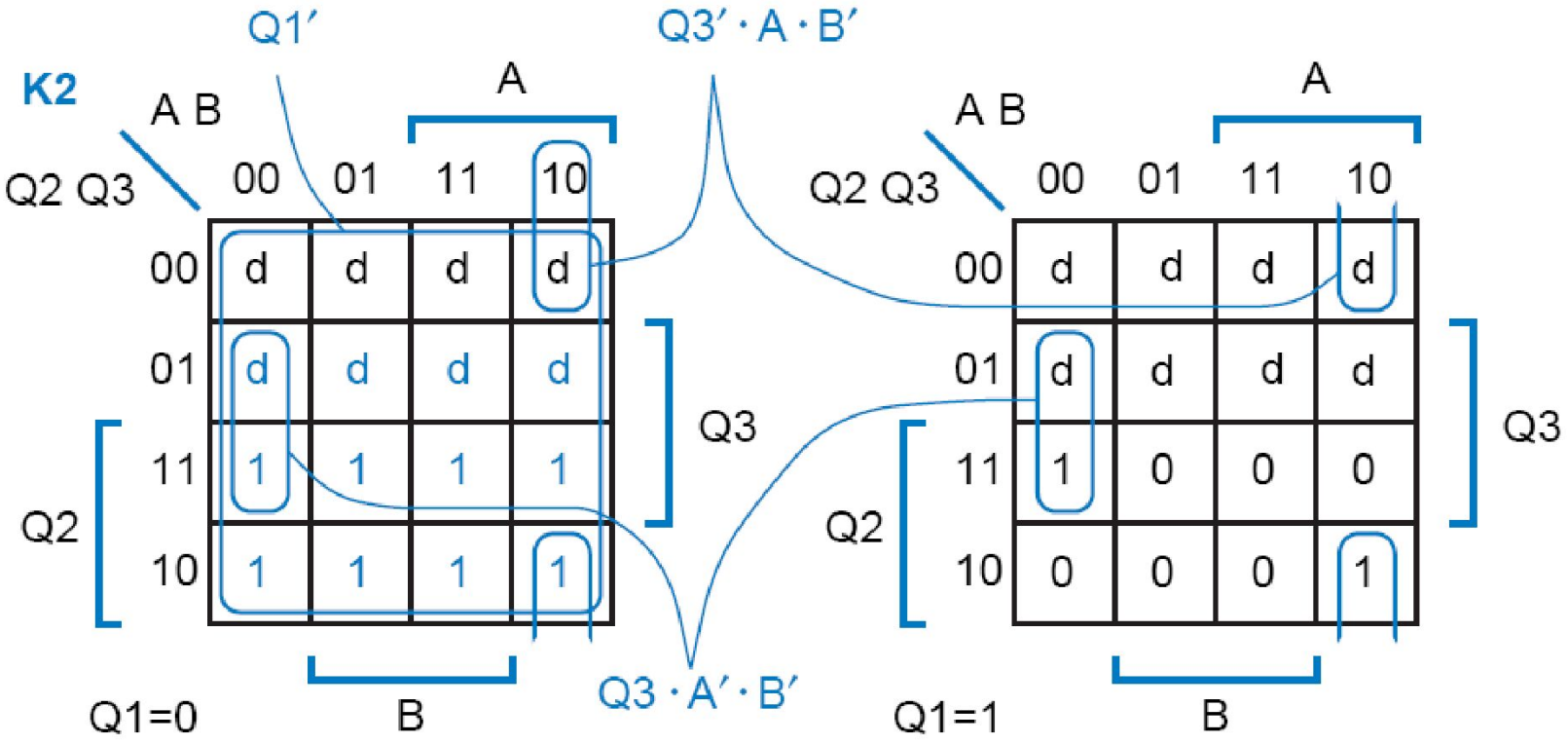
$$K1 = 0$$

Case 3: Synthesis Using JK FF



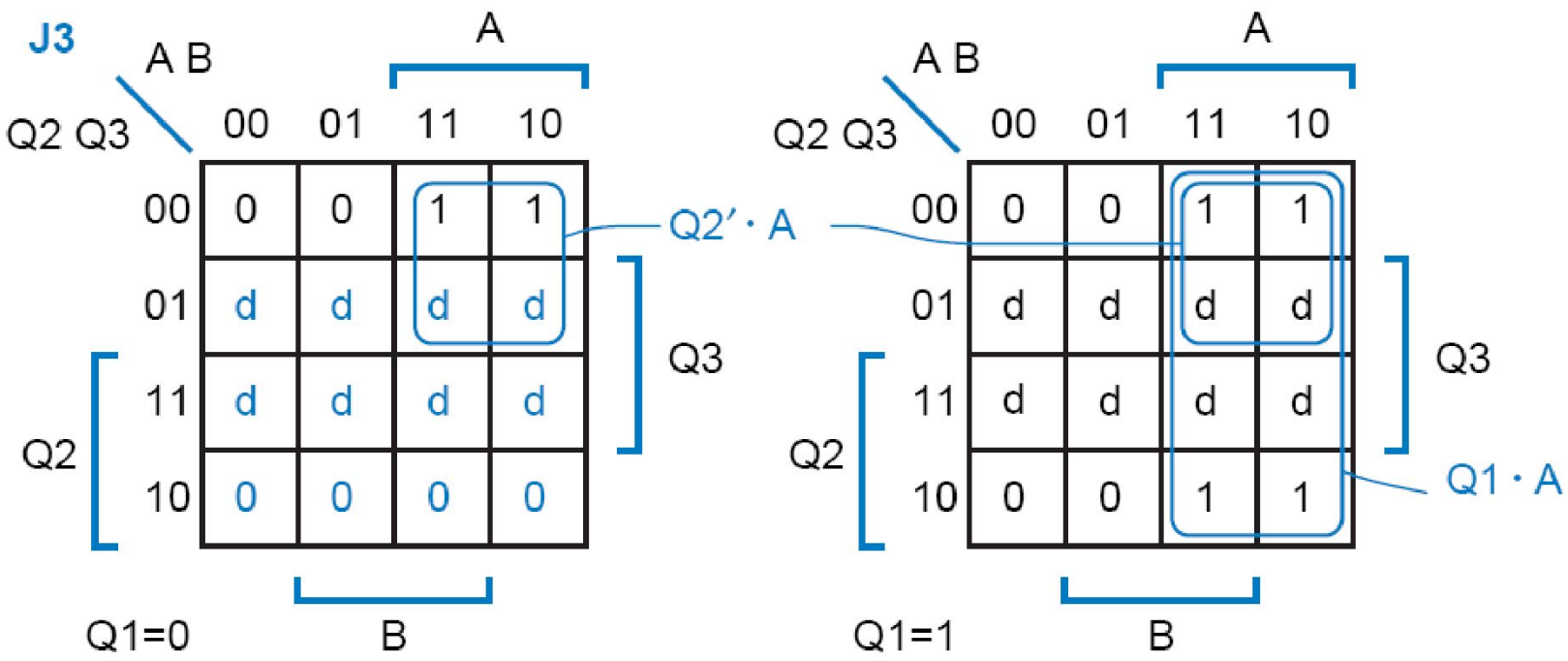
$$J2 = Q1.Q3'.A' + Q1.Q3.A$$

Case 3: Synthesis Using JK FF



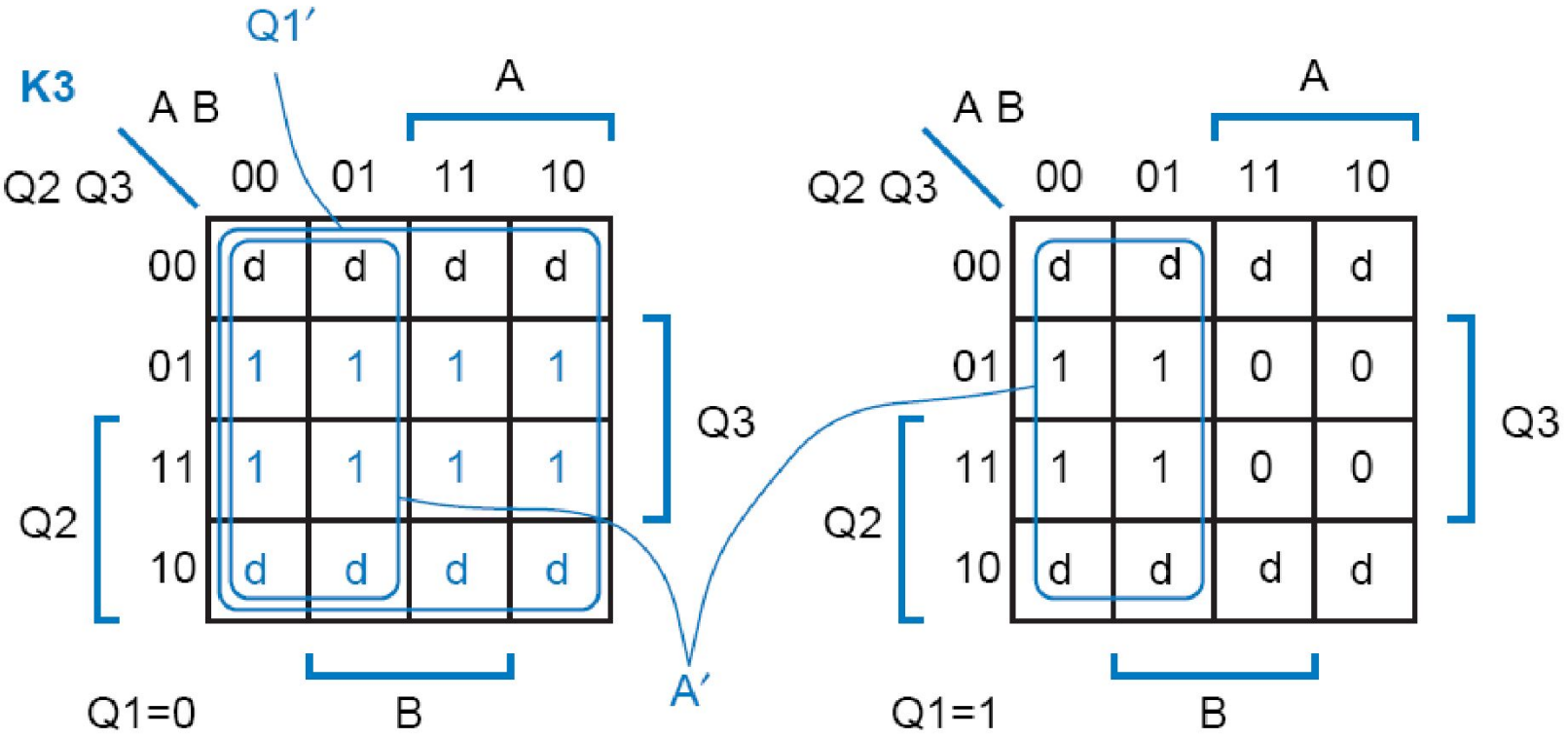
$$K2 = Q1' + Q3'.A.B' + Q3.A'.B'$$

Case 3: Synthesis Using JK FF



$$J3 = Q2'.A + Q1.A$$

Case 3: Synthesis Using JK FF



$$K3 = Q1' + A'$$

Case 3: Synthesis Using JK FF

- These equations take two more gates to realize than minimal-risk equations using D flip-flops
  - so J-K flip-flops didn't save us anything

Case 3: Synthesis Using JK FF

- Minimal cost using JK FF

$J1 = 1,$

$K1 = 0$

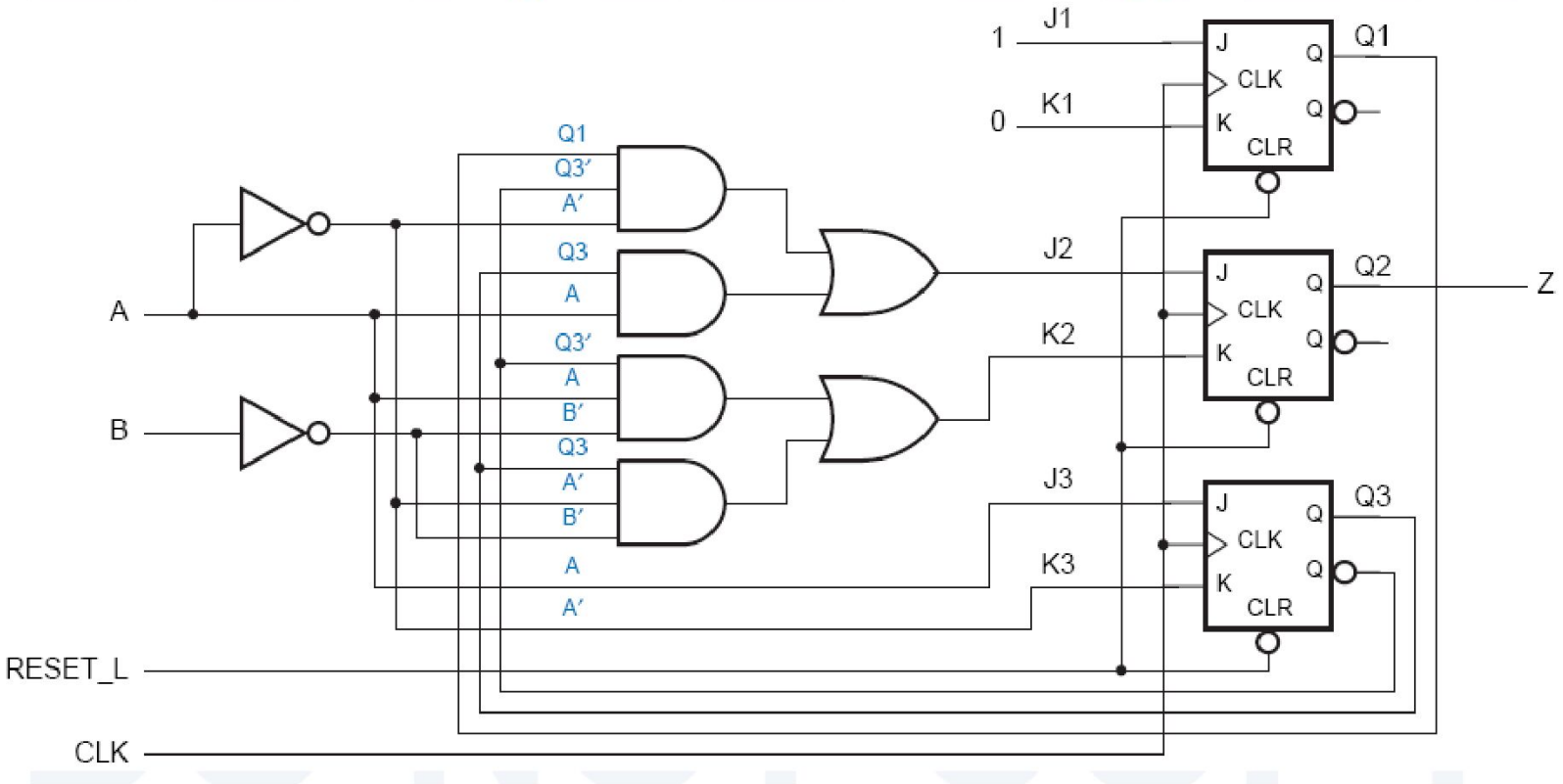
$J2 = Q1.Q3'.A' + Q3.A$

$K2 = Q3'.A.B' + Q3.A'.B'$

$J3 = A$

$K3 = A'$

Case 3: Synthesis Using JK FF



- This circuit has two more gates than the minimal-cost D circuit
  - J-K flip-flops still didn't save us anything.

Sequential Circuit Design Using ROM or PLA

	Q_1	Q_2	Q_3	X=0	X=1	Z X=0	Z X=1
A	0	0	0	001	010	1	0
B	0	0	1	011	100	1	0
C	0	1	0	100	100	0	1
D	0	1	1	101	101	0	1
E	1	0	0	101	110	1	0
H	1	0	1	000	000	0	1
K	1	1	0	000	-	1	-

$$D_1 = Q_1^+, \quad D_2 = Q_2^+ \quad and \quad D_3 = Q_3^+$$

Sequential Circuit Design Using ROM or PLA

X	Q_1	Q_2	Q_3	Z	D_1	D_2	D_3
0	0	0	0	1	0	0	1
0	0	0	1	1	0	1	1
0	0	1	0	0	1	0	0
0	0	1	1	0	1	0	1
0	1	0	0	1	1	0	1
0	1	0	1	0	0	0	0
0	1	1	0	1	0	0	0
0	1	1	1	x	x	x	x
1	0	0	0	0	0	1	0
1	0	0	1	0	1	0	0
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	1	0	0	0	1	1	0
1	1	0	1	1	0	0	0
1	1	1	0	x	x	x	x
1	1	1	1	x	x	x	x

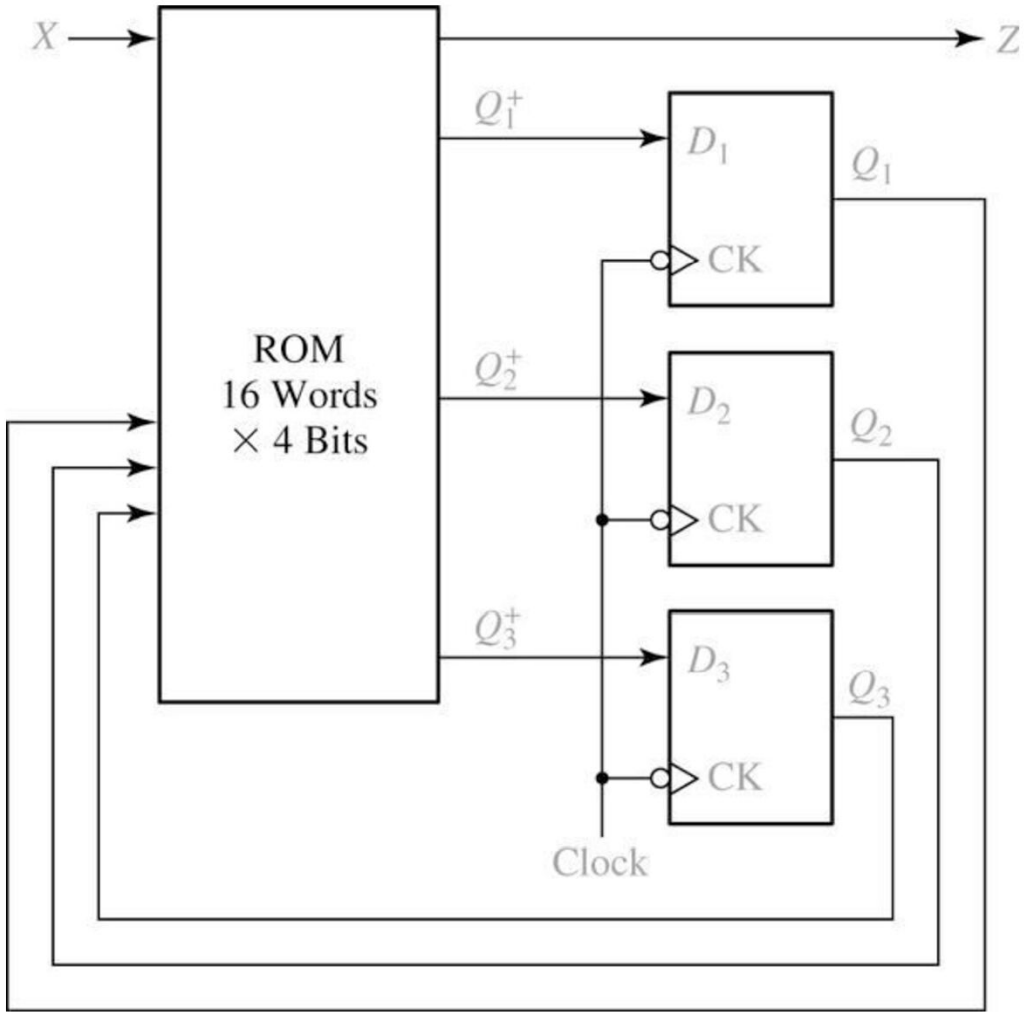
***ROM INPUTS**

(X,Q_1,Q_3 and Q_3)

***ROM OUTPUTS**

(Z,D_1,D_2 and D_3)

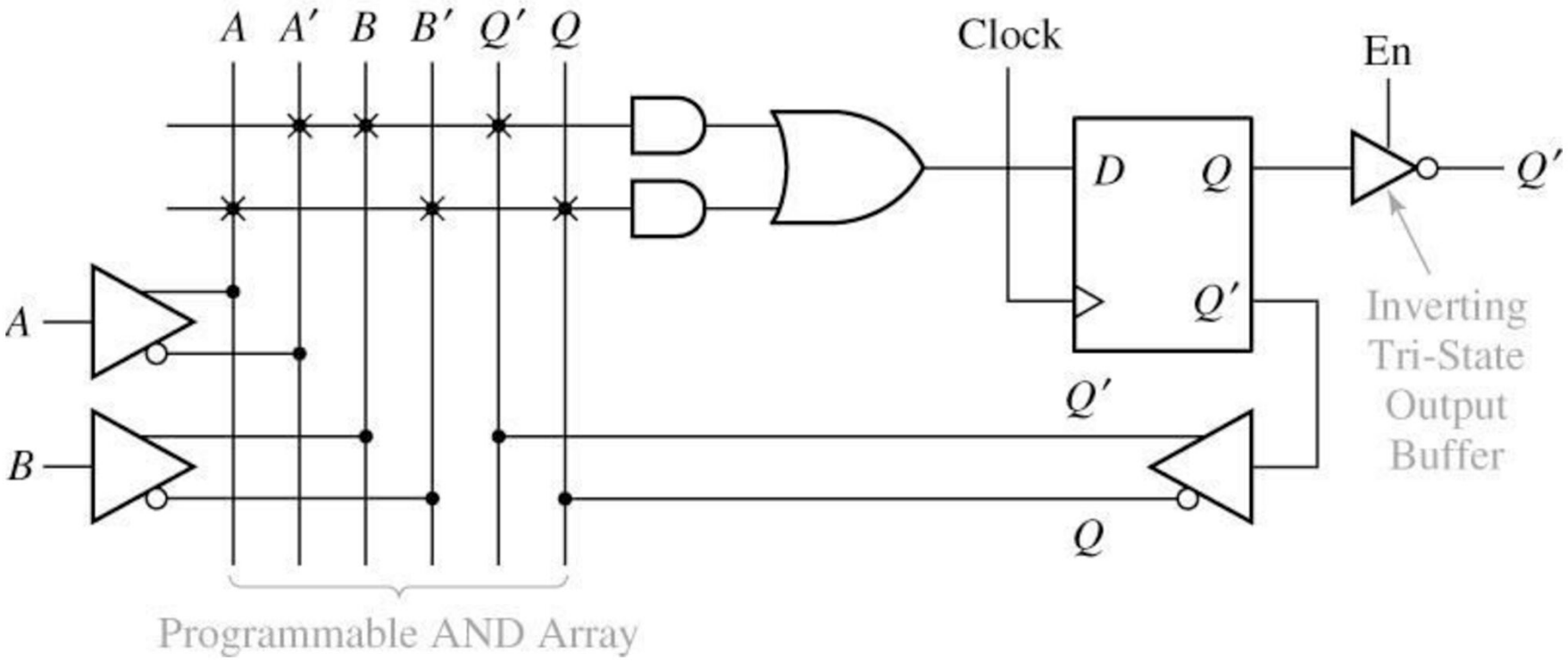
Sequential Circuit Design Using ROM



A ROM with four inputs (2^4 words) and four outputs is required.

Sequential Circuit Design Using PLA

Segment of Sequential PAL



$$Q^{+} = D = A'BQ' + AB'Q$$