



CprE 281: Digital Logic

Instructor: Alexander Stoytchev

<http://www.ece.iastate.edu/~alexs/classes/>

Review for the Final Exam

*CprE 281: Digital Logic
Iowa State University, Ames, IA
Copyright © Alexander Stoytchev*

Administrative Stuff

- **The FINAL exam is scheduled for**
- **Thursday Dec 19 @ 2:15 – 4:15 PM**
- **It will be in this room.**

Final Exam Format

- **The exam will cover: Chapter 1 to Chapter 6, and Sections 7.1-7.2**
- **Emphasis will be on Chapter 5, 6, and 7**
- **The exam will be closed book but open notes (you can bring up to 5 pages of handwritten/typed notes) plus your textbook.**

Final Exam Format

- The exam will be out of 130 points
- You need 95 points to get an A
- It will be great if you can score more than 100 points.
 - but you can't roll over your extra points 😞

Topics for the Final Exam

- **K-maps for 2, 3, and 4 variables**
- **Venn Diagrams**
- **Multiplexers (circuits and function)**
- **Synthesis of logic functions using multiplexers**
- **Shannon's Expansion Theorem**
- **1's complement and 2's complement representation**
- **Addition and subtraction of binary numbers**
- **Circuits for adding and subtracting**
- **Serial adder**
- **Latches (circuits, behavior, timing diagrams)**
- **Flip-Flops (circuits, behavior, timing diagrams)**
- **Counters (up, down, synchronous, asynchronous)**
- **Registers and Register Files**

Topics for the Final Exam

- **Synchronous Sequential Circuits**
- **FSMs**
- **Moore Machines**
- **Mealy Machines**
- **State diagrams, state tables, state-assigned tables**
- **State minimization**
- **Designing a counter**
- **Arbiter Circuits**
- **Reverse engineering a circuit**
- **ASM Charts**
- **Register Machines**
- **Bus structure and Simple Processors (i281 CPU)**
- **Assembly Language and Machine Language**
- **Something from Star Wars**

How to Study for the Final Exam

- **Form a study group**
- **Go over the slides for this class**
- **Go over the homeworks again**
- **Go over the problems at the end of Ch 5 & 6**
- **Exercise**
- **Get some sleep**

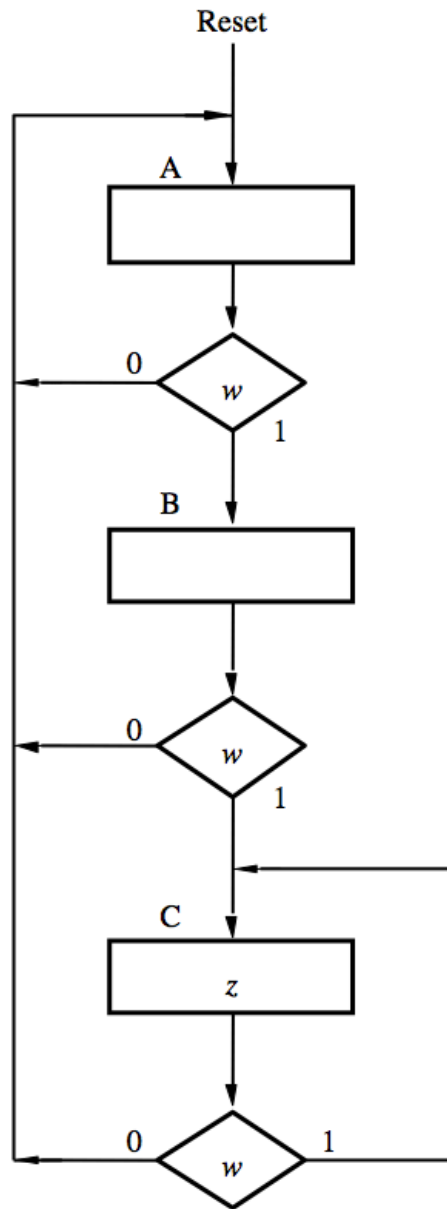
Administrative Stuff

- **Please check your grades on Canvas**
- **Let me know if something is wrong or missing**

Sample Problems

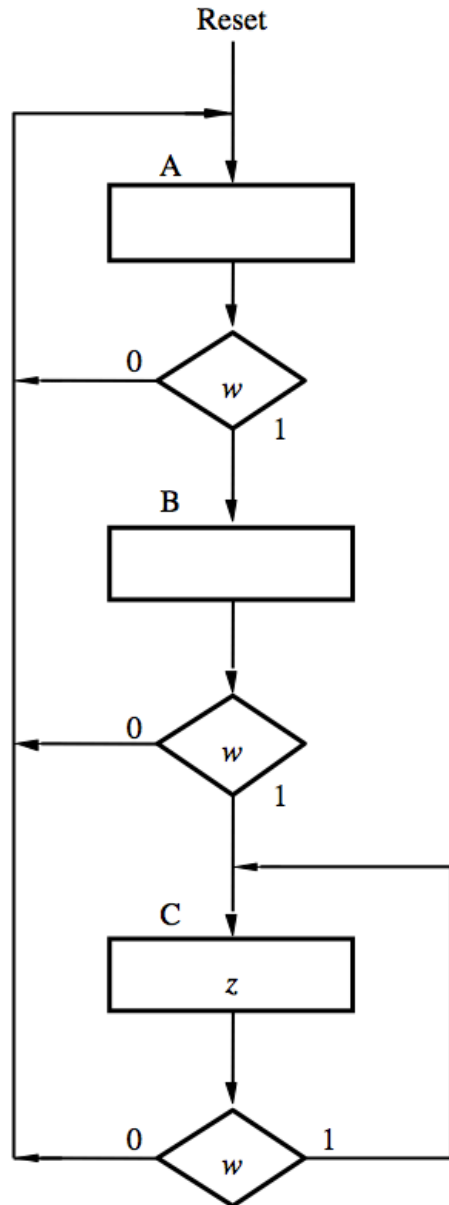
ASM Charts

Given an ASM chart draw the corresponding FSM

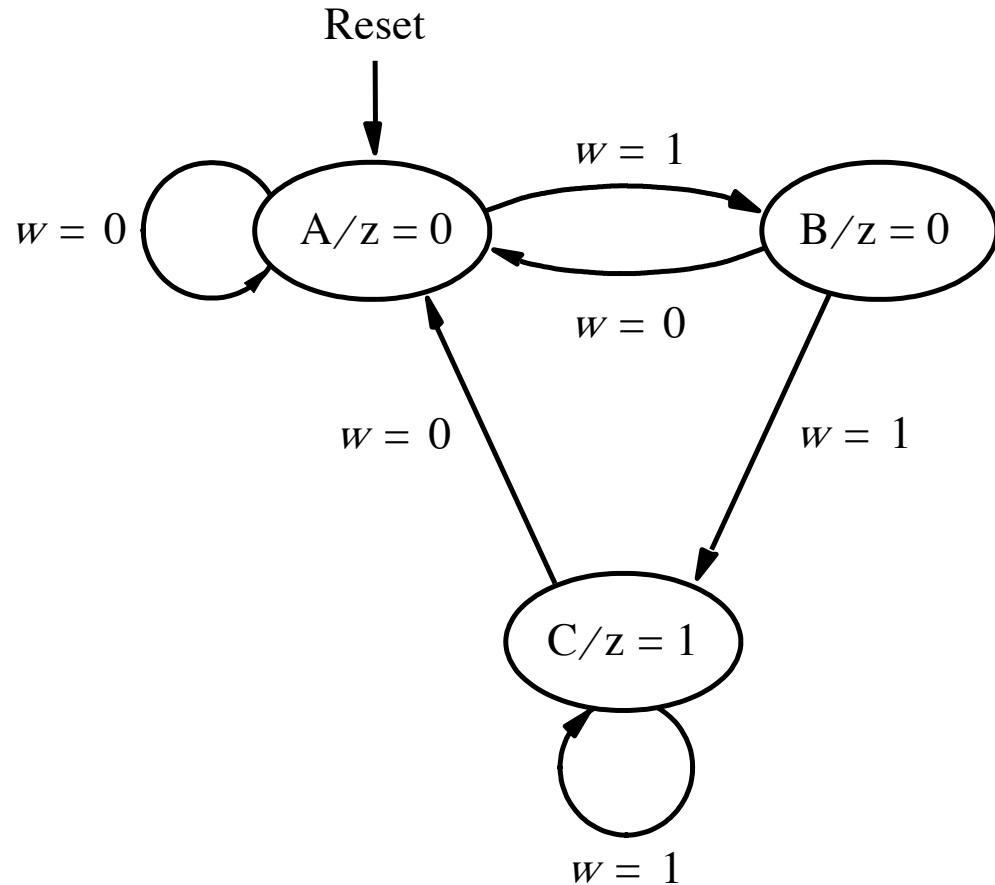


ASM Charts

Given an ASM chart draw the corresponding FSM



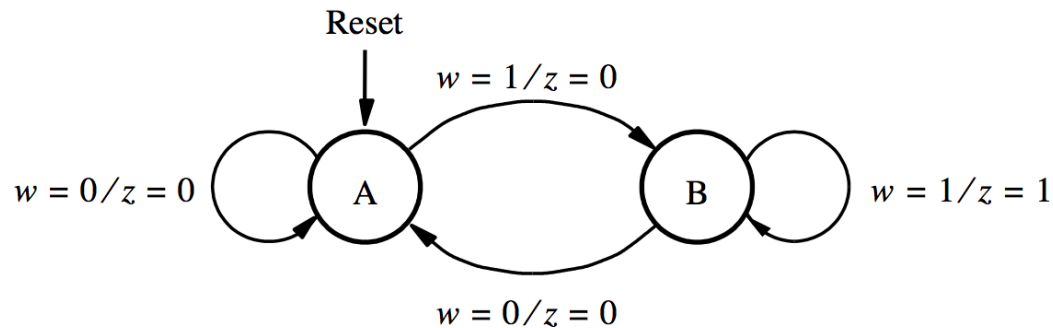
[Figure 6.82 from the textbook]



[Figure 6.3 from the textbook]

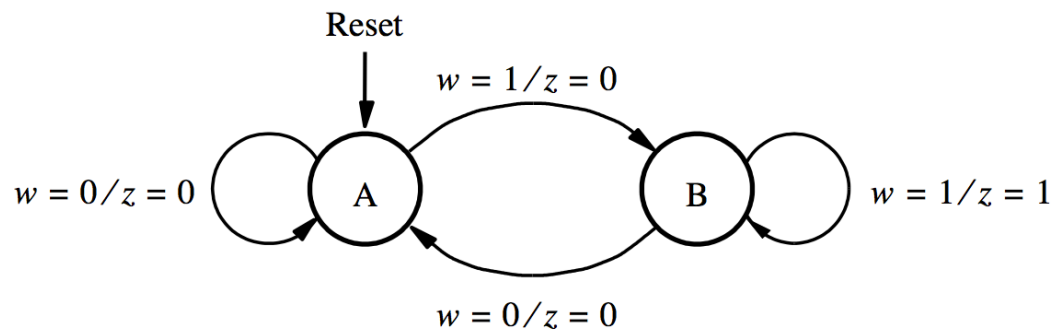
ASM Charts

Given an FSM draw the corresponding ASM Chart

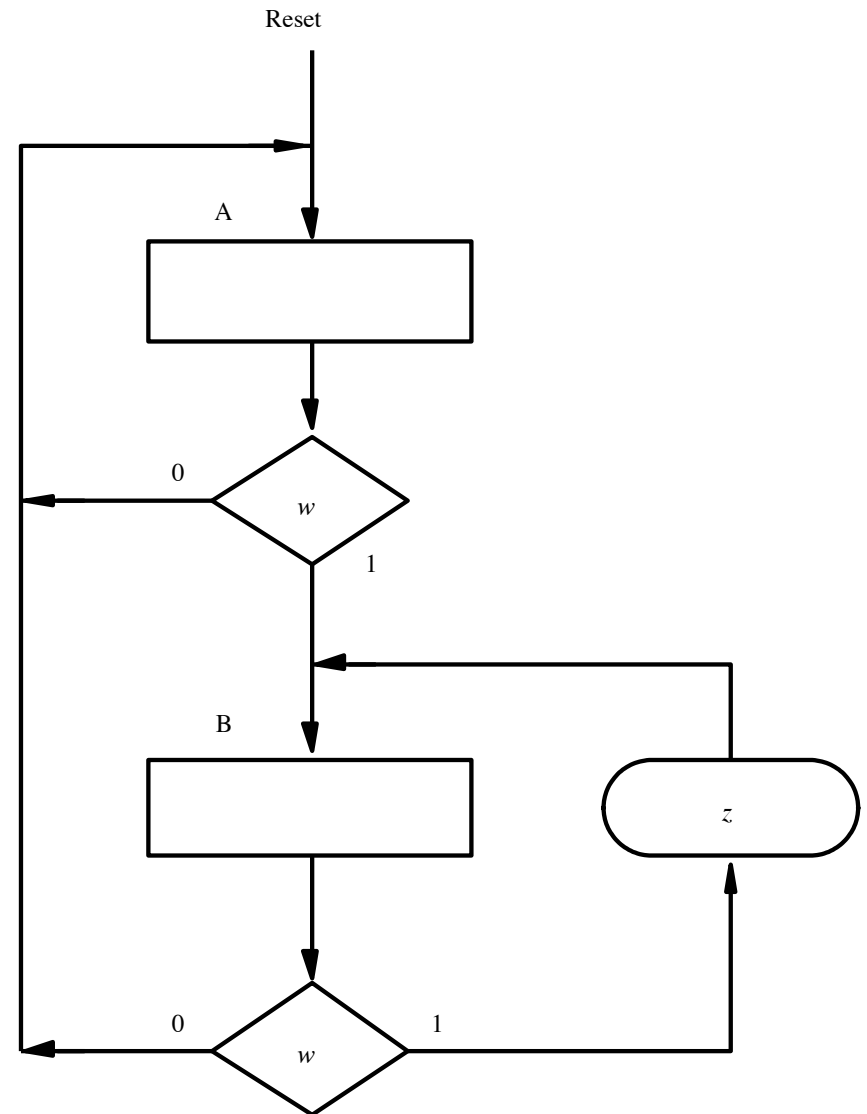


ASM Charts

Given an FSM draw the corresponding ASM Chart



[Figure 6.23 from the textbook]



[Figure 6.83 from the textbook]

Circuit Implementation of FSMs

Implement this state-assigned Table using JK flip-flops

	Present state $y_3y_2y_1$	Next state		Output z
		$w = 0$	$w = 1$	
		$Y_3Y_2Y_1$	$Y_3Y_2Y_1$	
A	000	100	110	0
B	100	101	110	0
C	101	101	110	1
D	110	100	111	0
E	111	100	111	1

Circuit Implementation of FSMs

Implement this state-assigned Table using JK flip-flops

$$J_1 = wy_2 + \bar{w}y_3\bar{y}_2$$

$$K_1 = \bar{w}y_2 + wy_1\bar{y}_2$$

$$J_2 = w$$

$$K_2 = \bar{w}$$

$$J_3 = 1$$

$$K_3 = 0$$

$$z = y_1$$

	Present state $y_3y_2y_1$	Next state		Output z
		$w = 0$	$w = 1$	
		$Y_3Y_2Y_1$	$Y_3Y_2Y_1$	
A	000	100	110	0
B	100	101	110	0
C	101	101	110	1
D	110	100	111	0
E	111	100	111	1

Circuit Implementation of FSMs

Implement this state-assigned Table using JK flip-flops

	Present state $y_3y_2y_1$	Flip-flop inputs								Output z
		$w = 0$				$w = 1$				
		$Y_3Y_2Y_1$	J_3K_3	J_2K_2	J_1K_1	$Y_3Y_2Y_1$	J_3K_3	J_2K_2	J_1K_1	
A	000	100	$1d$	$0d$	$0d$	110	$1d$	$1d$	$0d$	0
B	100	101	$d0$	$0d$	$1d$	110	$d0$	$1d$	$0d$	0
C	101	101	$d0$	$0d$	$d0$	110	$d0$	$1d$	$d1$	1
D	110	100	$d0$	$d1$	$0d$	111	$d0$	$d0$	$1d$	0
E	111	100	$d0$	$d1$	$d1$	111	$d0$	$d0$	$d0$	1

Excitation table with JK flip-flops

Register Machines:

What does this program do?

How many balls are left in each register at the end of the program?



Register 1



Register 2



Register 3

STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	3	1	2
2.	Deb	2	3	4
3.	Inc	3	2	
4.	End			

Register Machines:

Move the contents of register 2 to register 3



Register 1



Register 2



Register 3

STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	3	1	2
2.	Deb	2	3	4
3.	Inc	3	2	
4.	End			

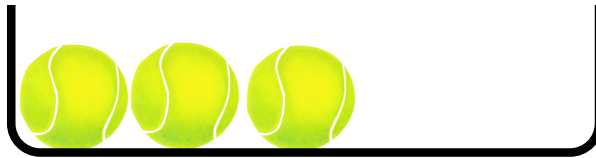
Register Machines:

What does this program do?

How many balls are left in each register at the end of the program?



Register 1



Register 2



Register 3

STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	3	1	2
2.	Deb	2	2	3
3.	Deb	1	4	6
4.	Inc	3	5	
5.	Inc	2	3	
6.	Deb	2	7	8
7.	Inc	1	6	
8.	End			

Register Machines:

Copy the contents of register 1 to register 3
using register 2 as a temporary storage



Register 1



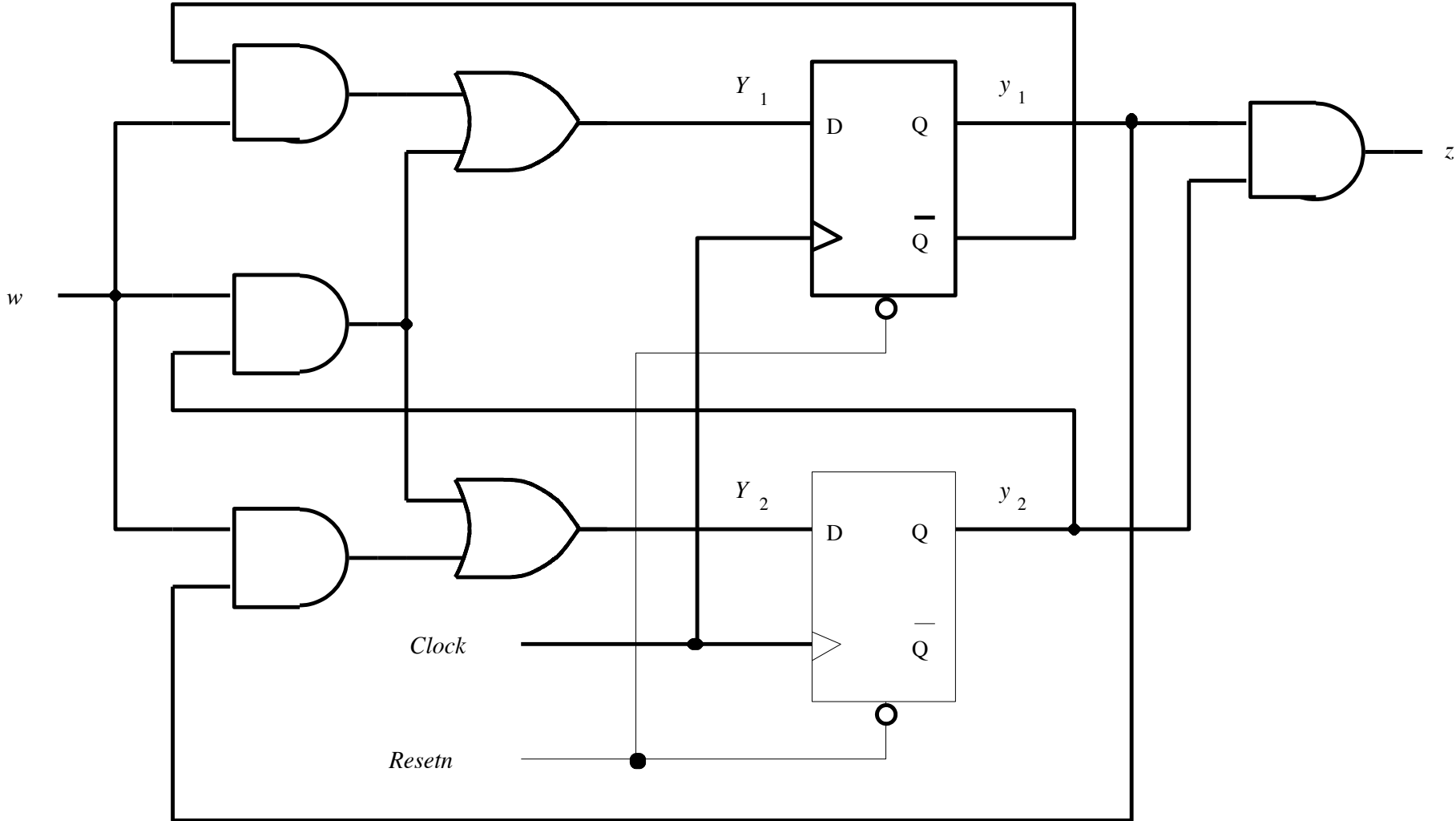
Register 2



Register 3

STEP	INSTRUCTION	REGISTER	GO TO STEP	[BRANCH TO STEP]
1.	Deb	3	1	2
2.	Deb	2	2	3
3.	Deb	1	4	6
4.	Inc	3	5	
5.	Inc	2	3	
6.	Deb	2	7	8
7.	Inc	1	6	
8.	End			

What does this circuit do?



[Figure 6.75 from the textbook]

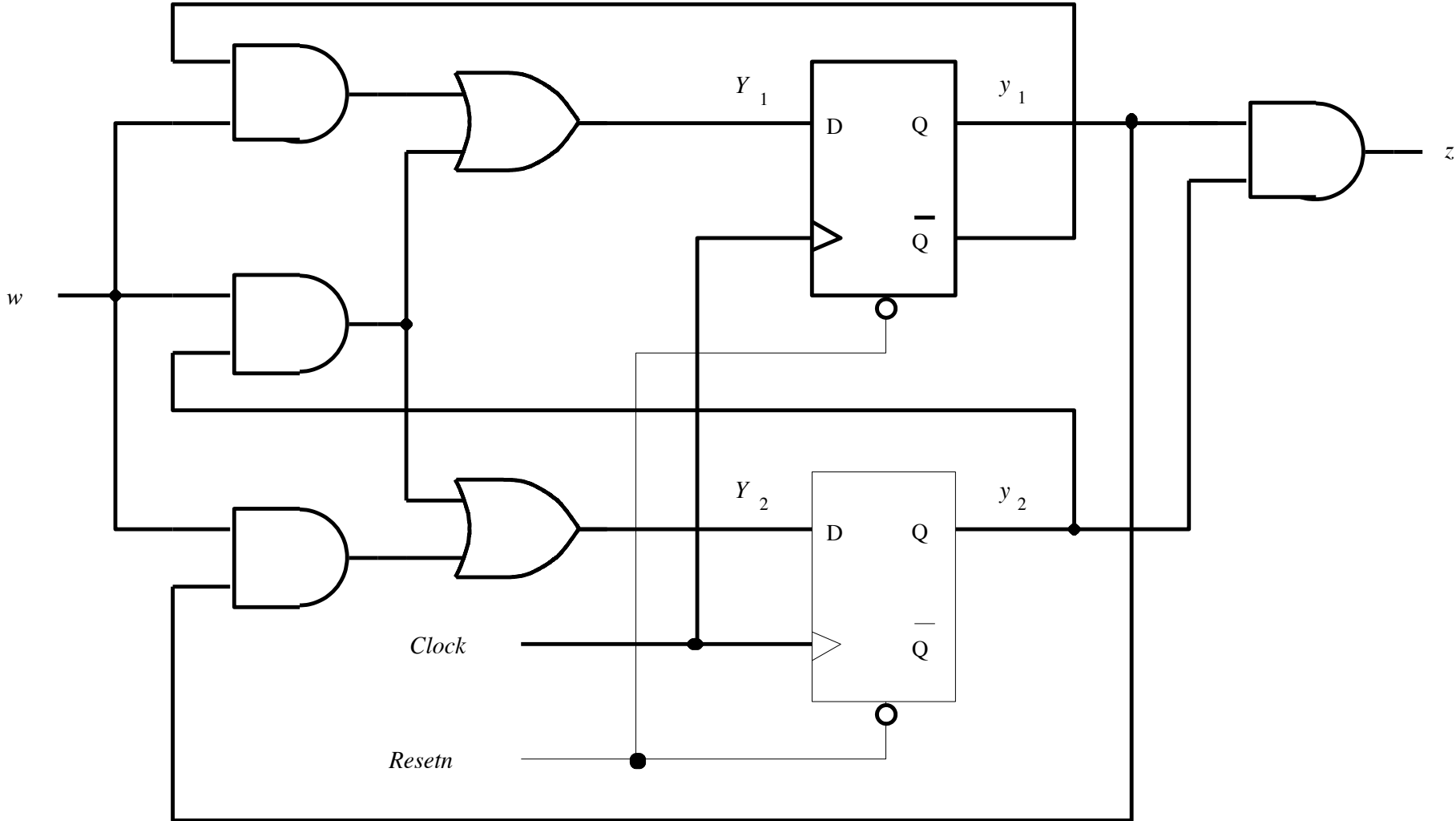
Approach

- **Find the flip-flops**
- **Outputs of the flip-flops = present state variables**
- **Inputs of the flip-flops determine the next state variables**
- **Determine the logical expressions for the outputs**
- **Given this info it is easy to do the state-assigned table**
- **Next do the state table**
- **Finally, draw the state diagram.**

Goal

- **Given a circuit diagram for a synchronous sequential circuit, the goal is to figure out the FSM**
- **Figure out the present state variables, the next state variables, the state-assigned table, the state table, and finally the state diagram.**
- **In other words, the goal is to reverse engineer the circuit.**

What does this circuit do?

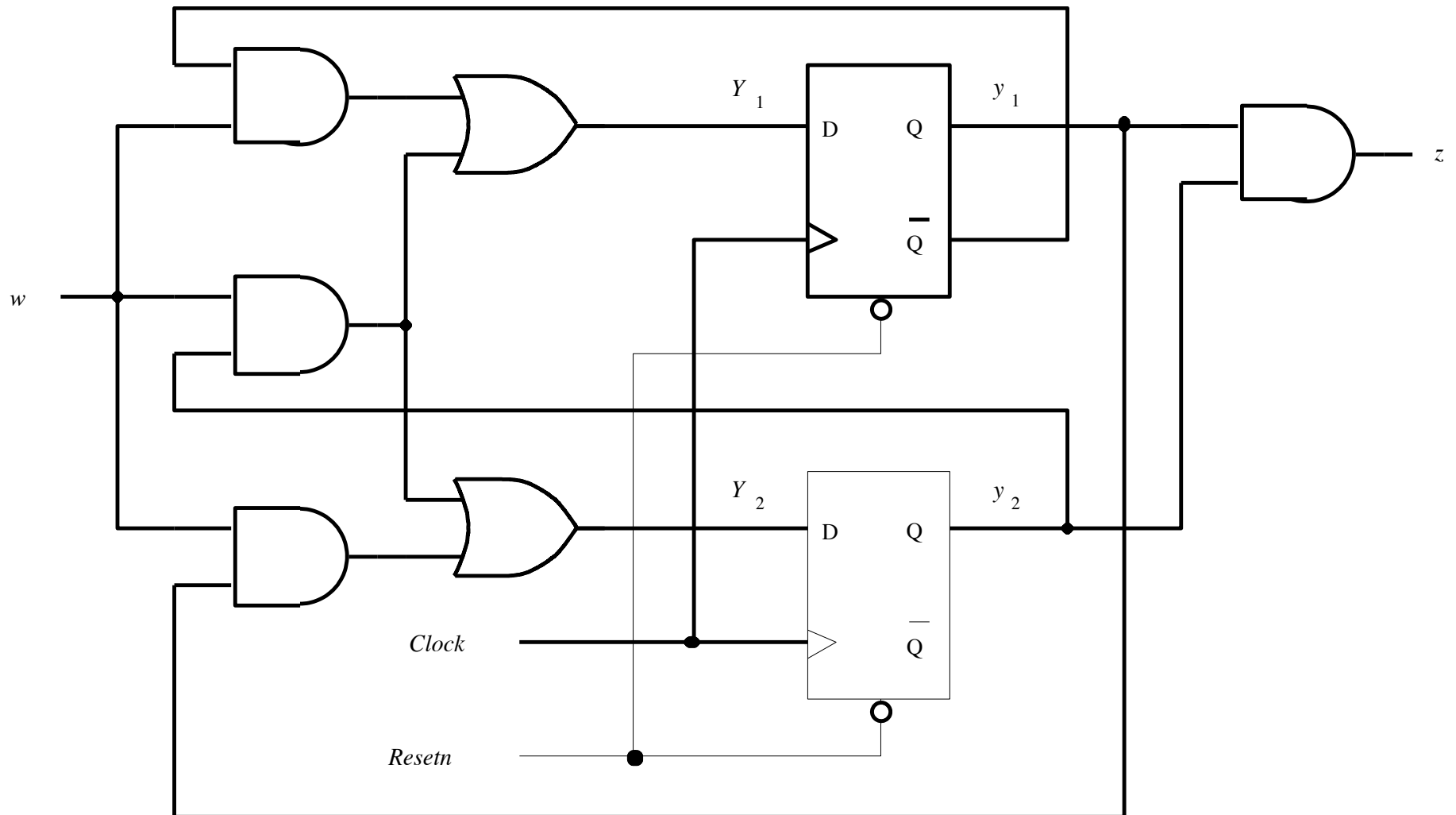


[Figure 6.75 from the textbook]

Approach

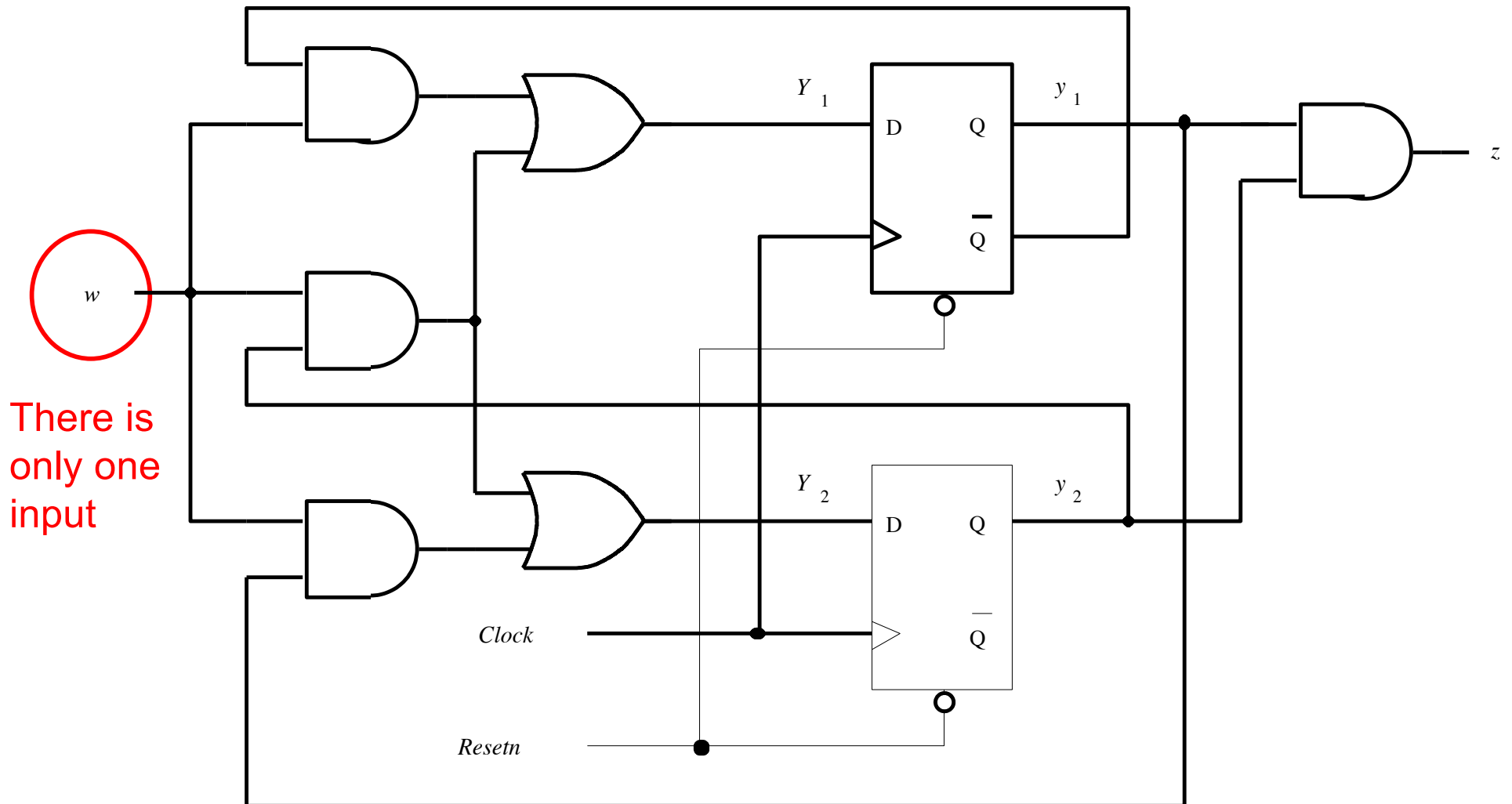
- **Find the flip-flops**
- **Outputs of the flip-flops = present state variables**
- **Inputs of the flip-flops determine the next state variables**
- **Determine the logical expressions for the outputs**
- **Given this info it is easy to do the state-assigned table**
- **Next do the state table**
- **Finally, draw the state diagram.**

Where are the inputs?



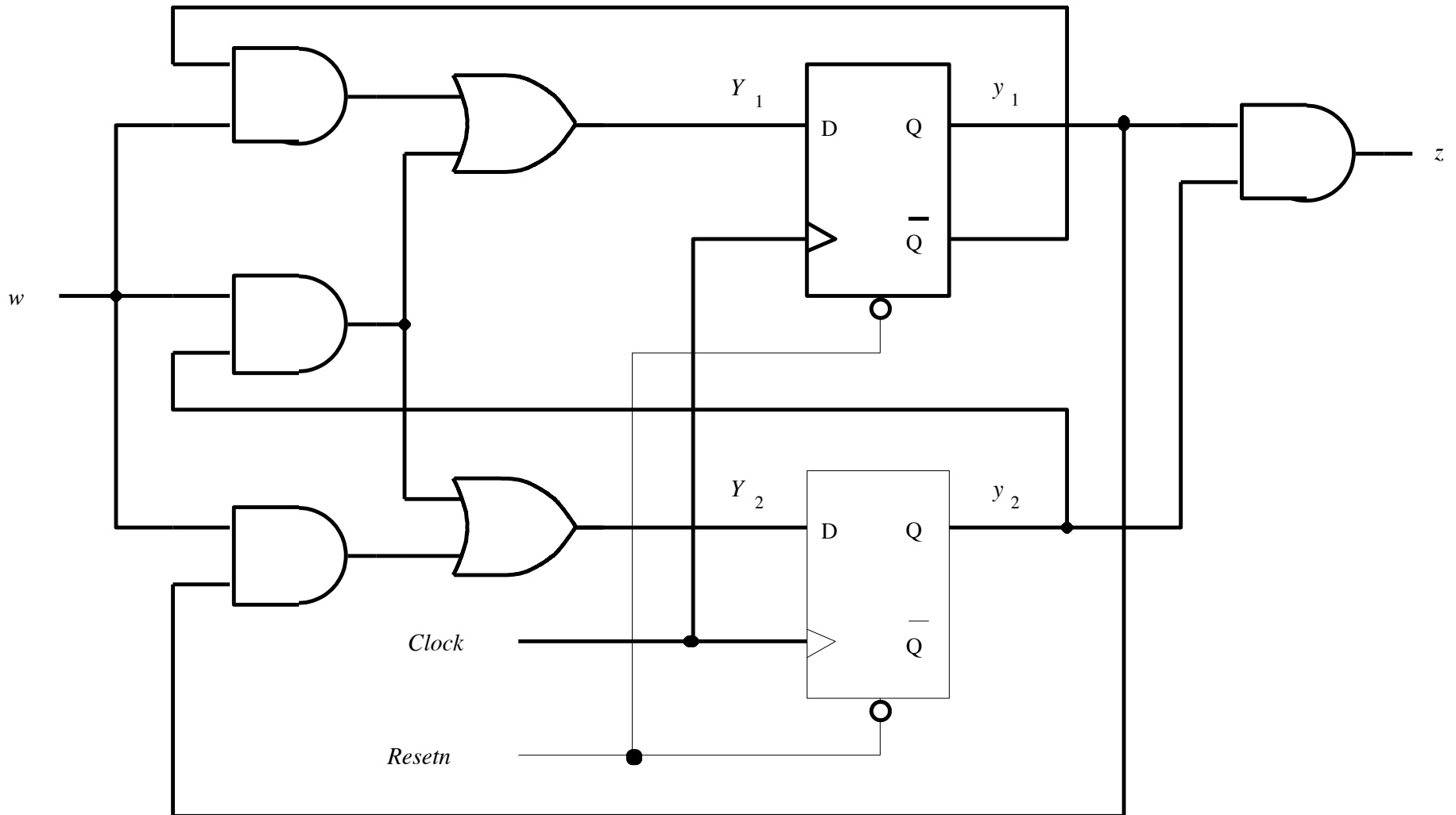
[Figure 6.75 from the textbook]

Where are the inputs?



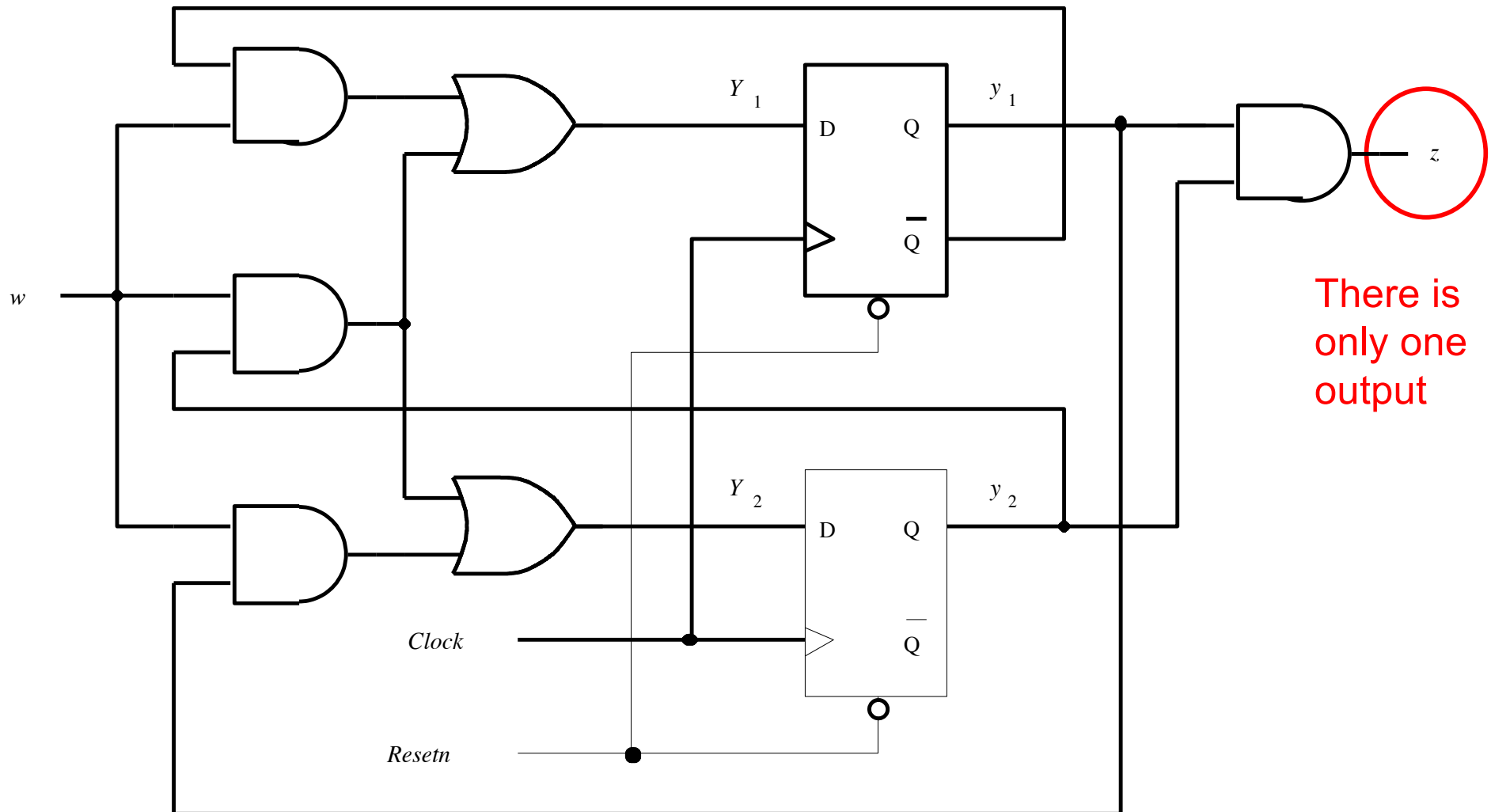
[Figure 6.75 from the textbook]

Where are the outputs?



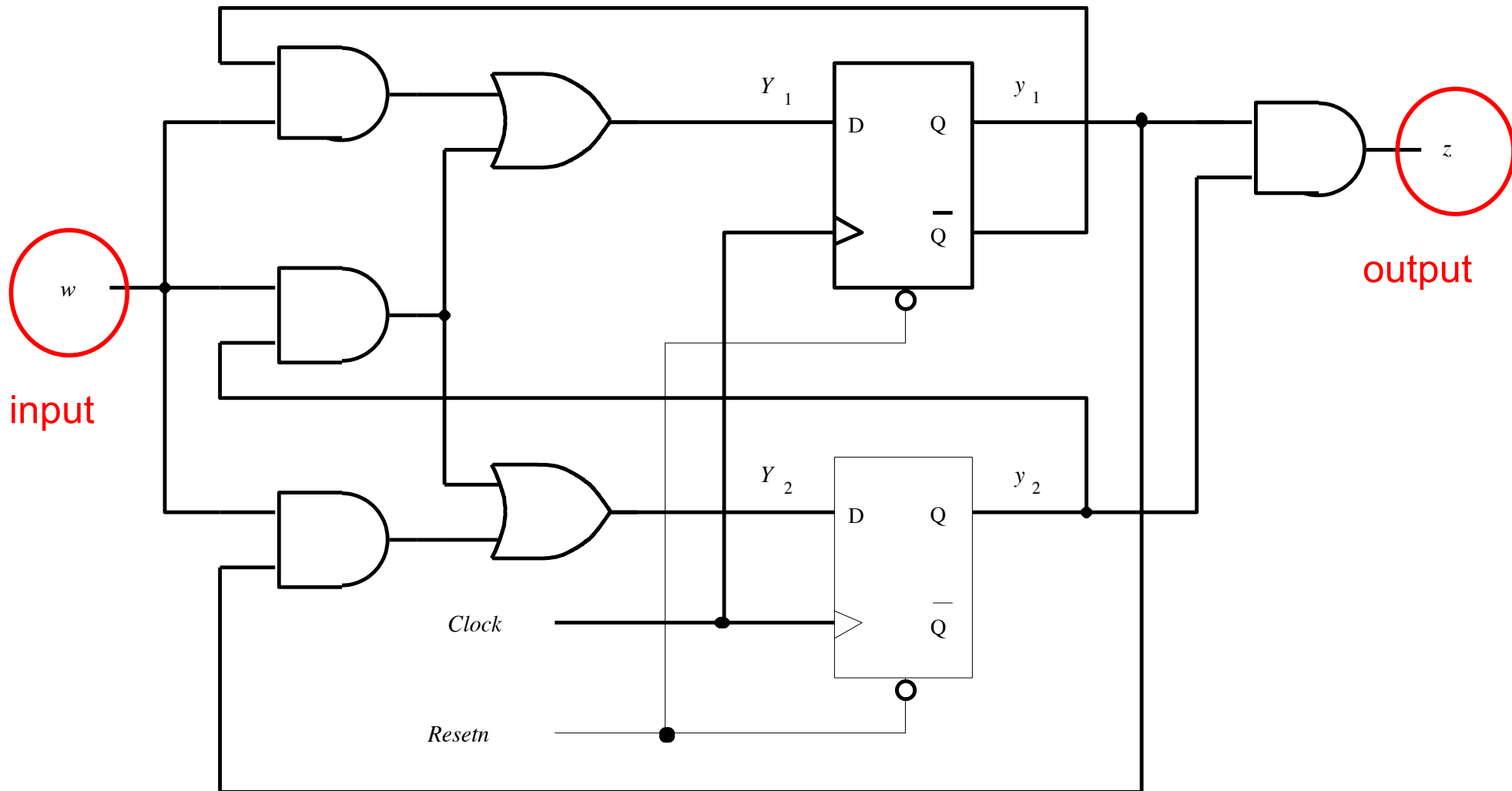
[Figure 6.75 from the textbook]

Where are the outputs?

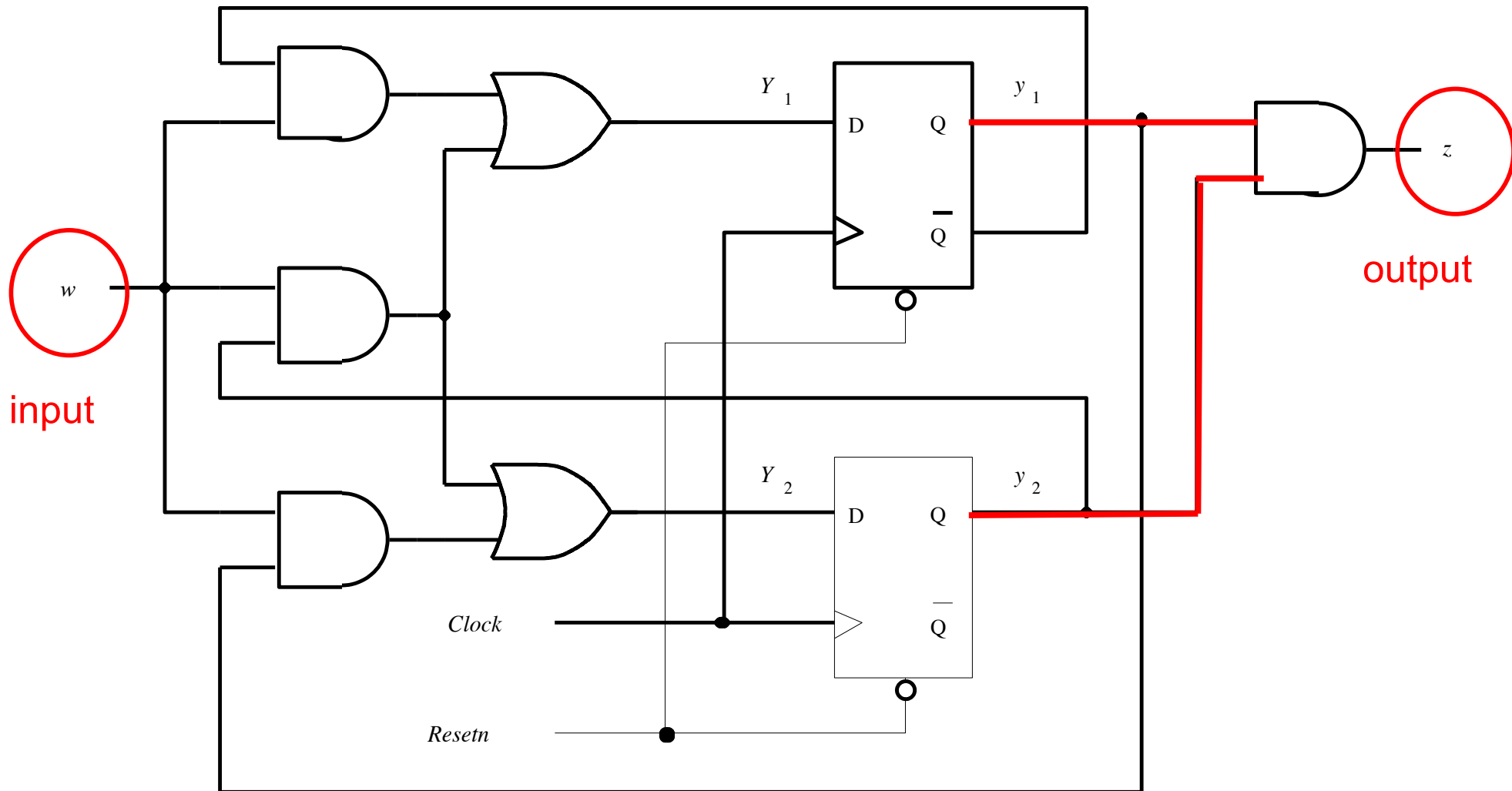


[Figure 6.75 from the textbook]

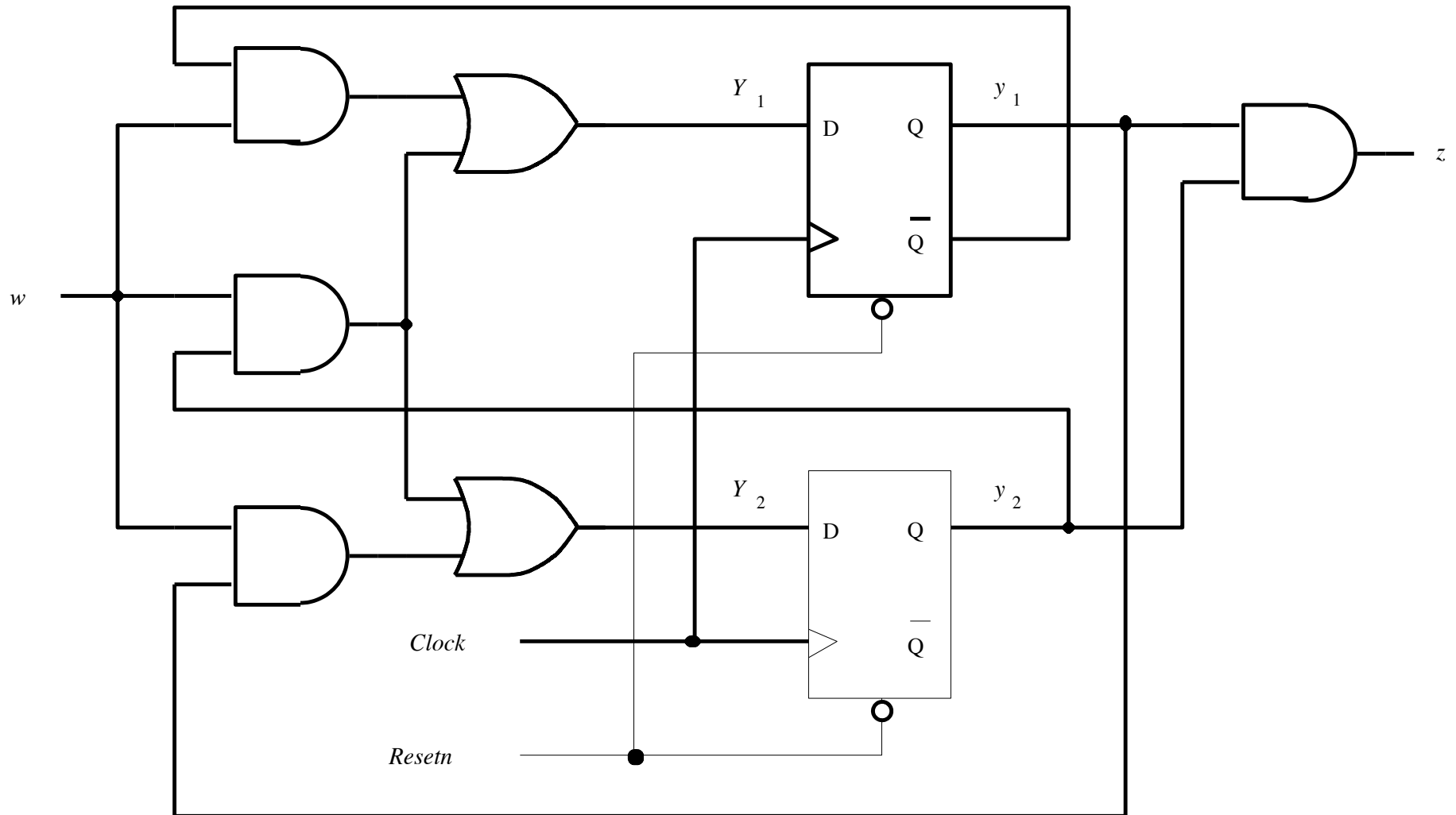
Where kind of machine is this? Moore or Mealy?



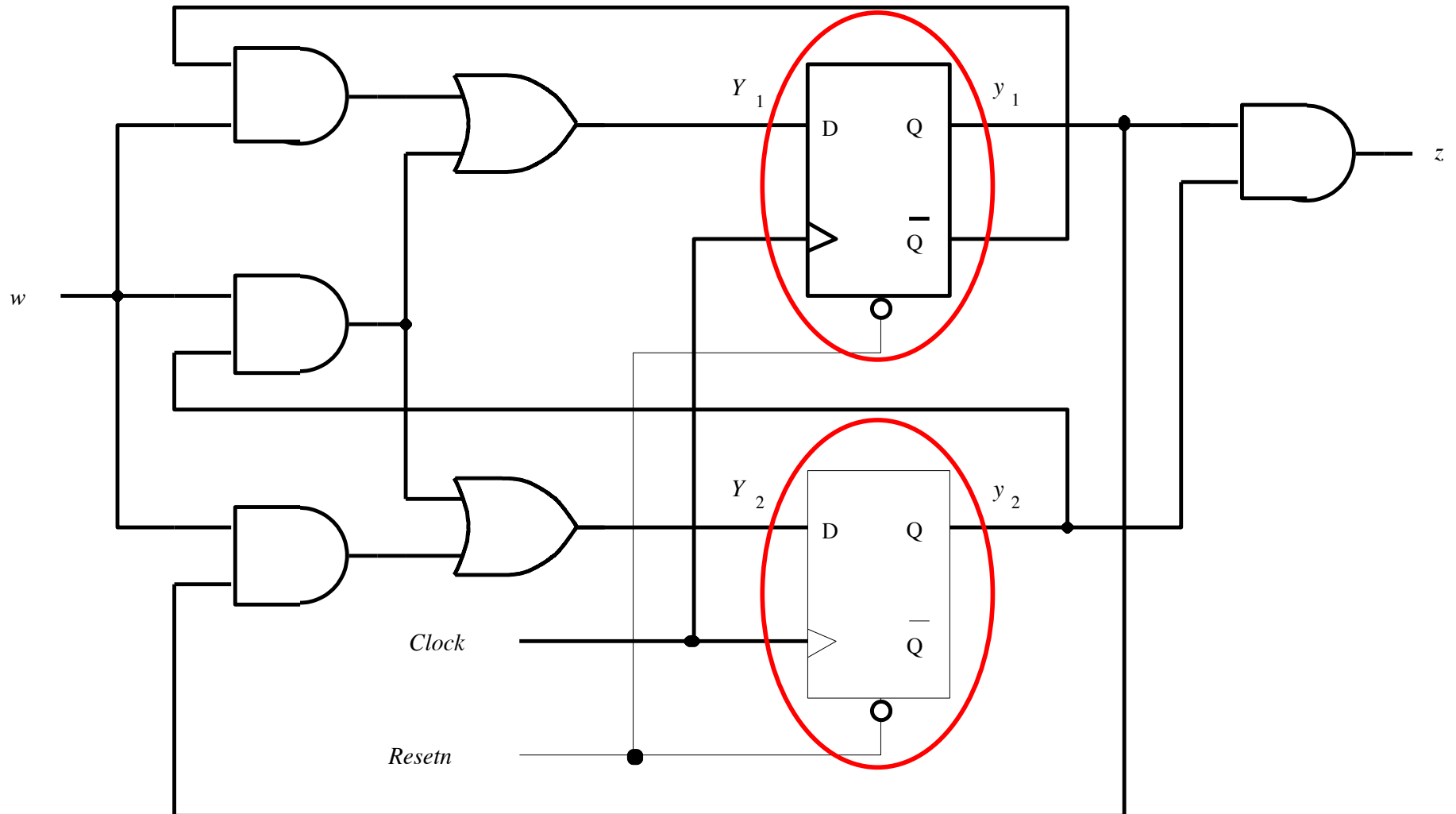
Moore: because the output does not depend directly on the primary input



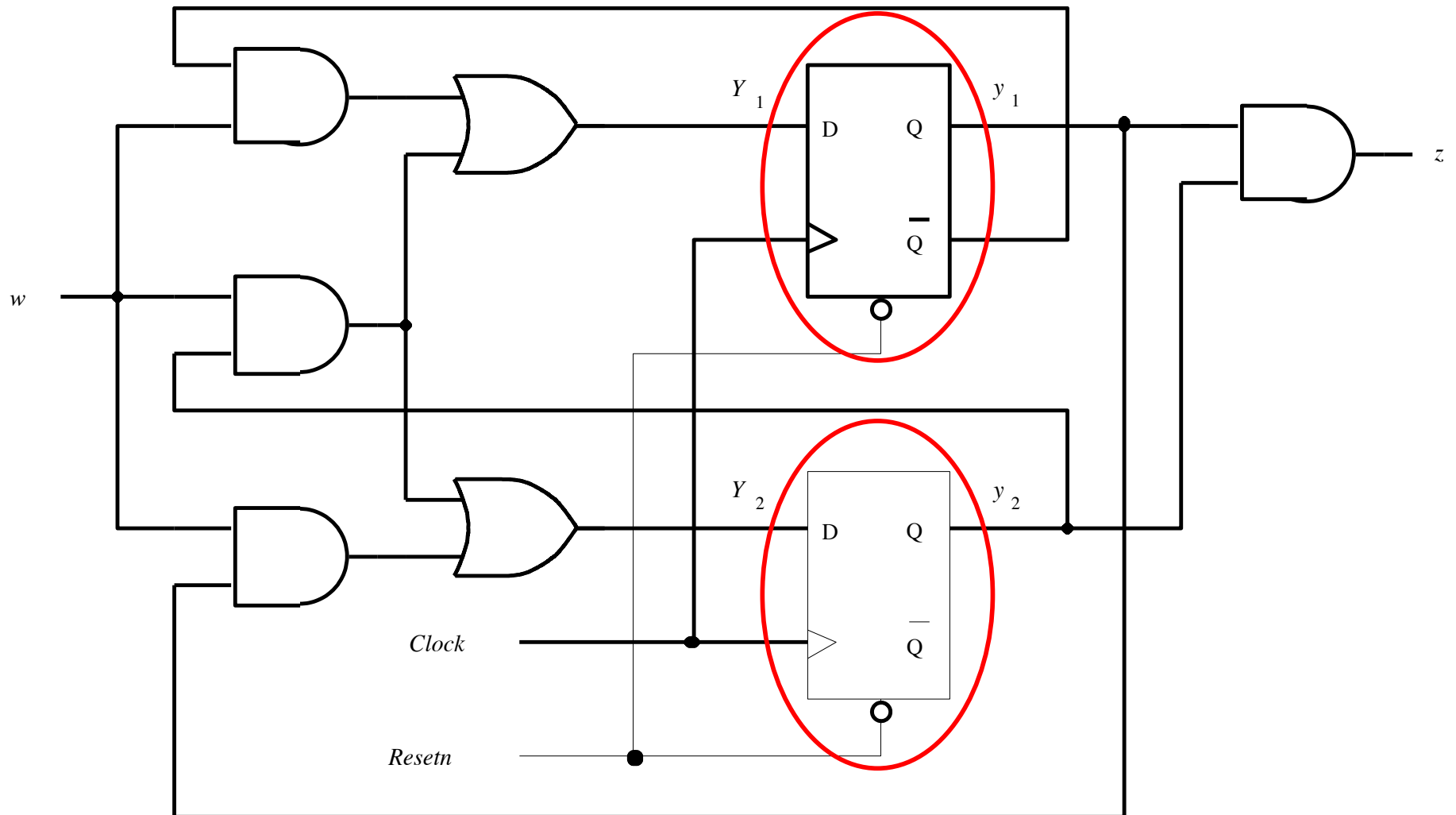
Where are the memory elements?



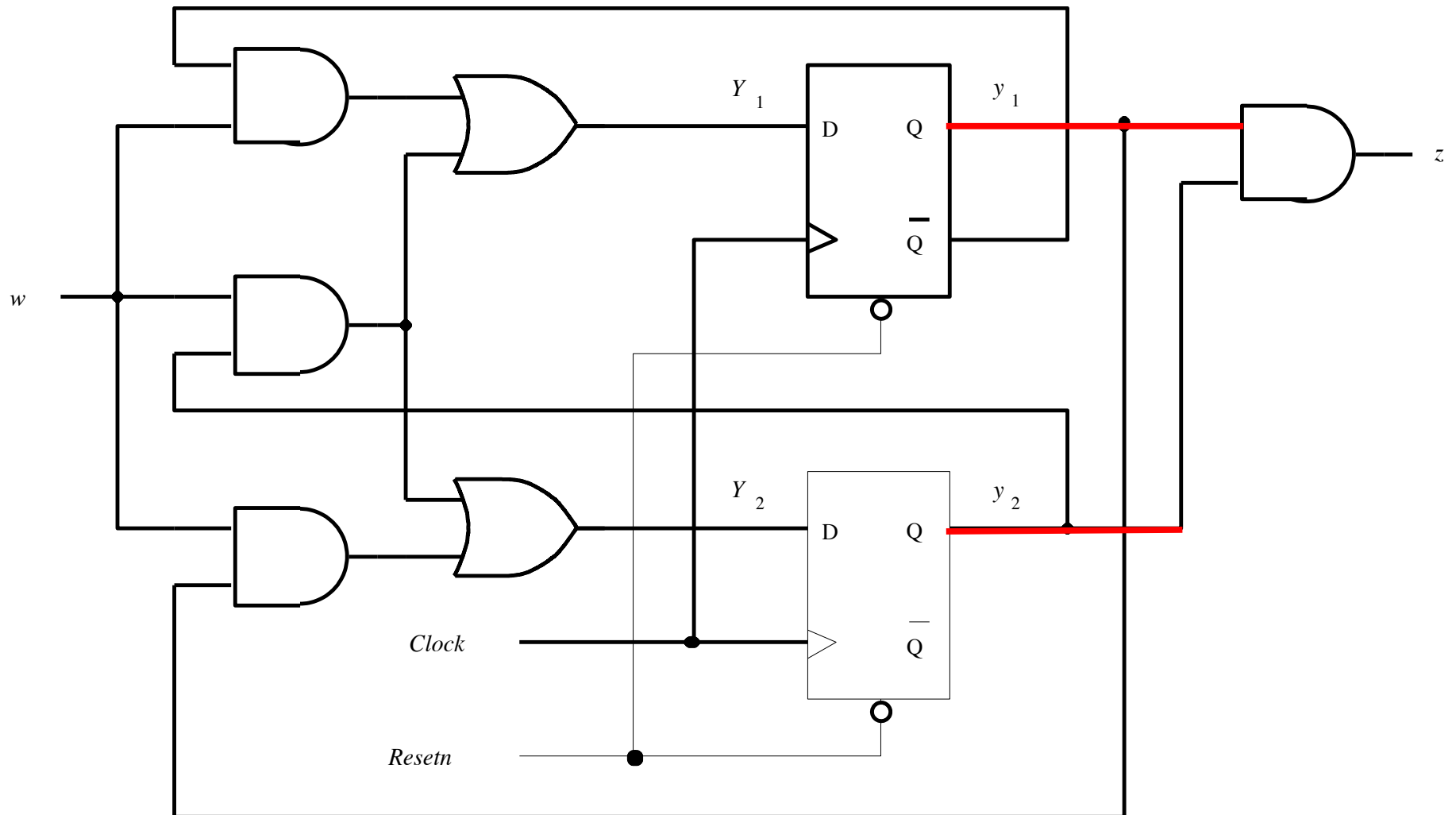
Where are the memory elements?



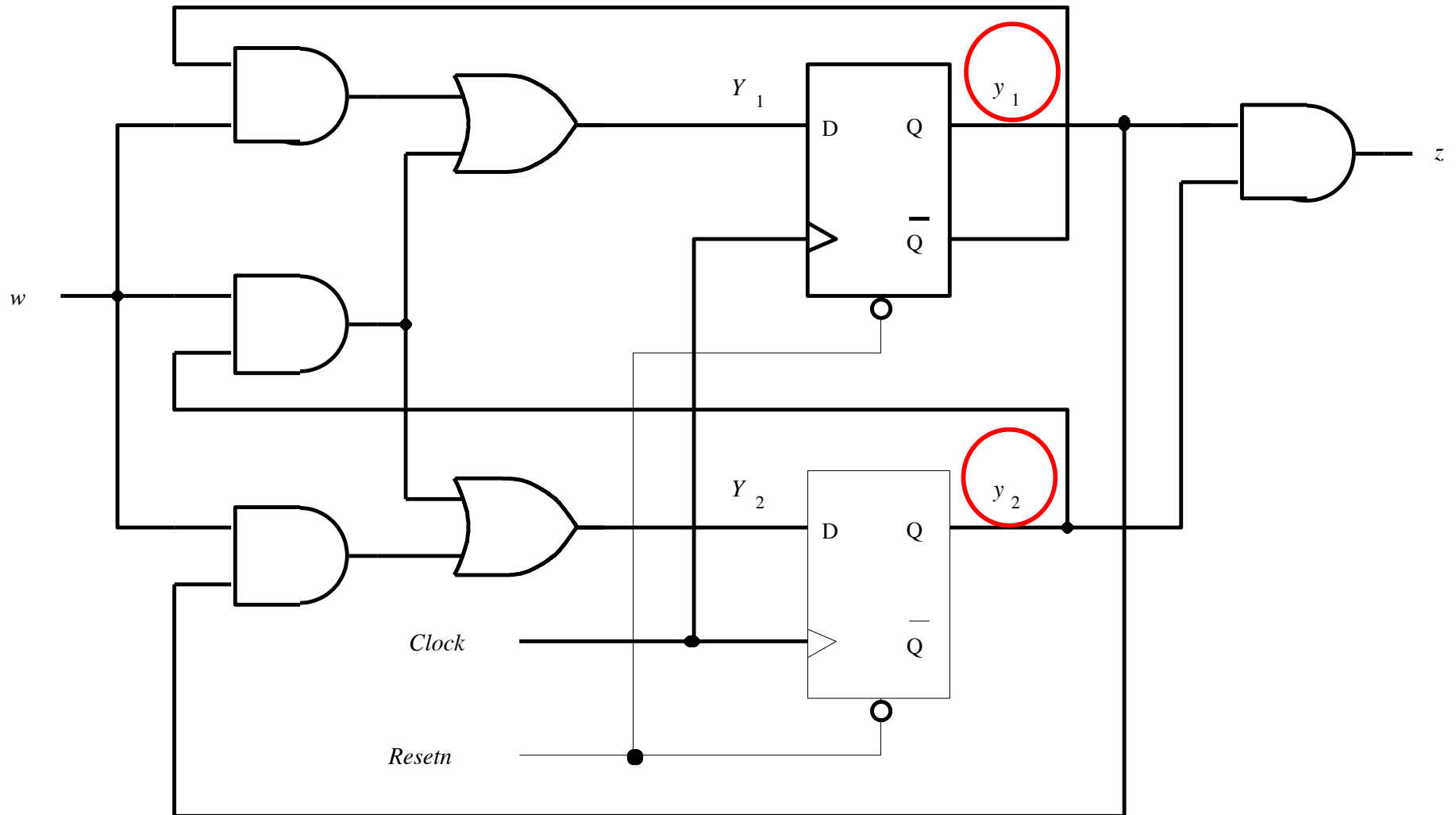
Where are the outputs of the flip-flops?



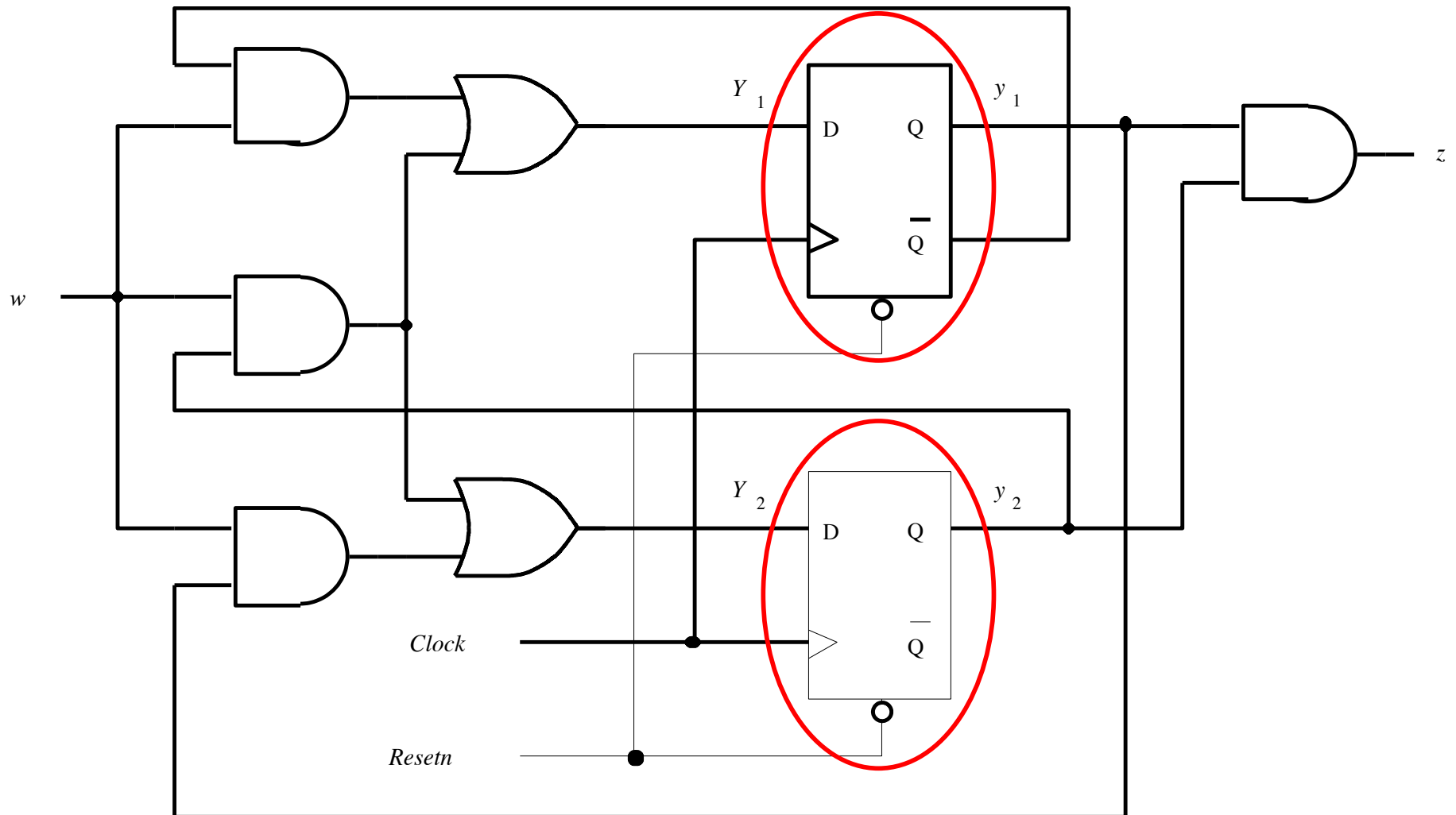
Where are the outputs of the flip-flops?



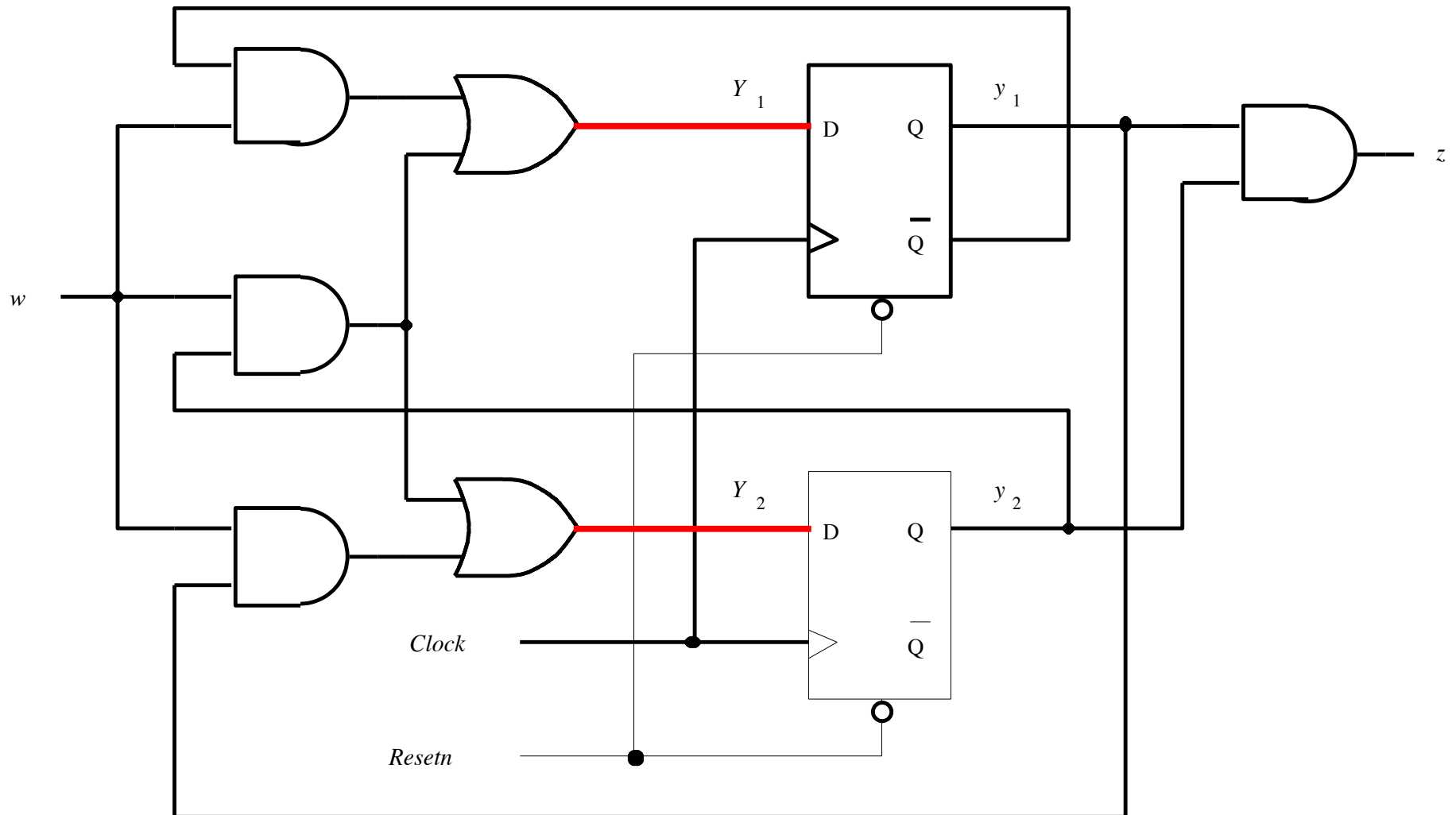
These are the present-state variables



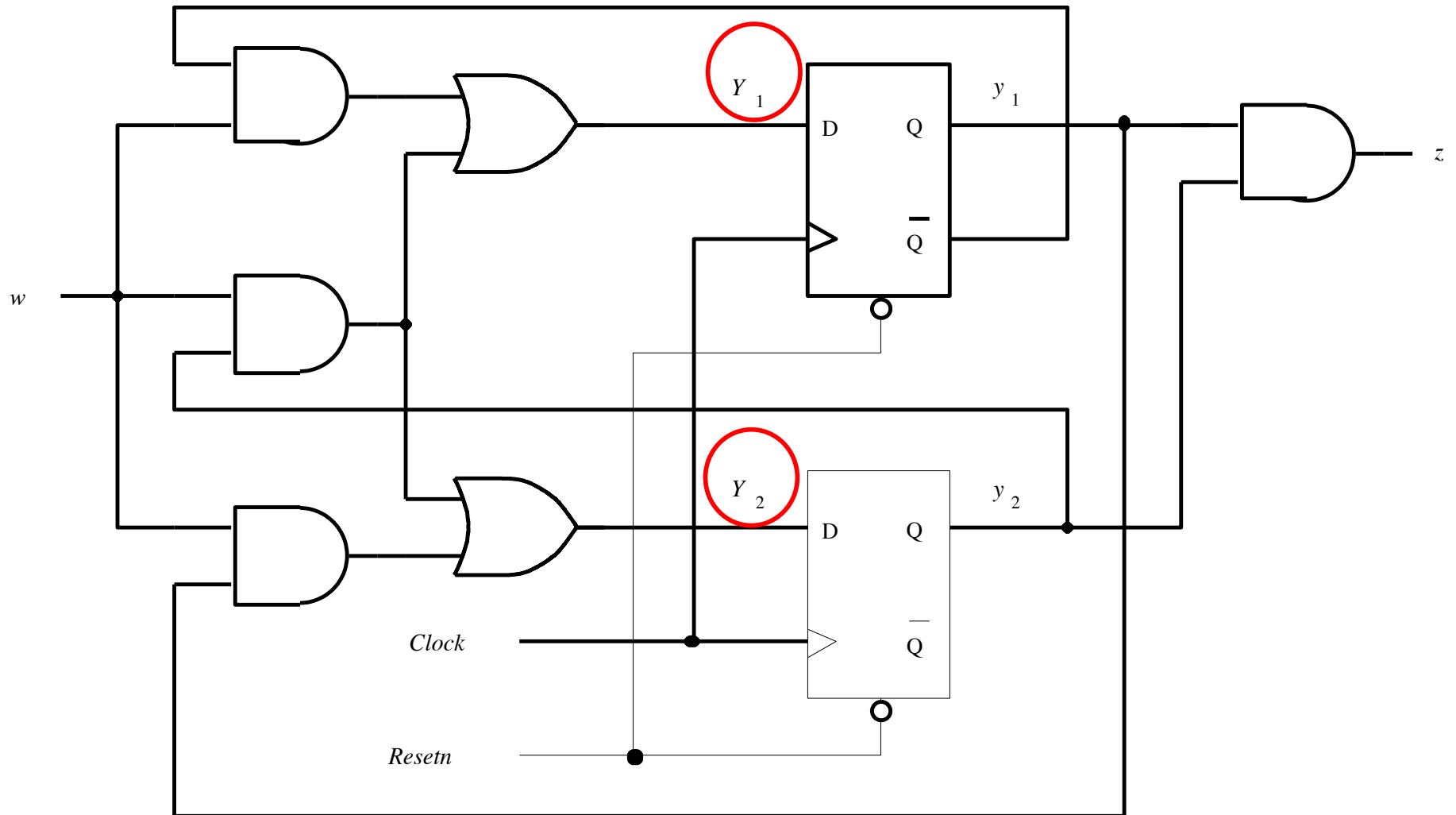
Where are the inputs of the flip-flops?



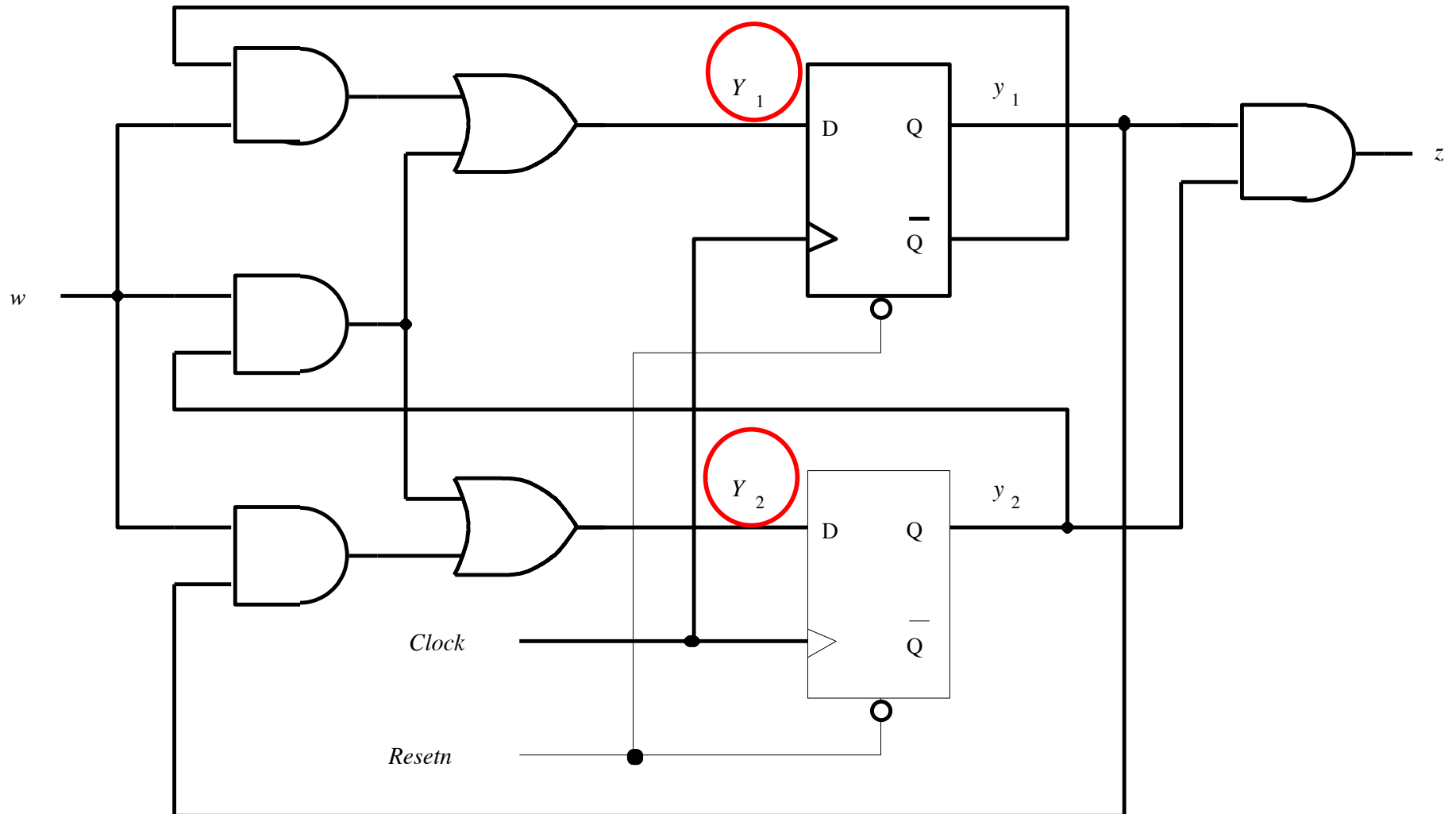
Where are the inputs of the flip-flops?



These are the next-state variables

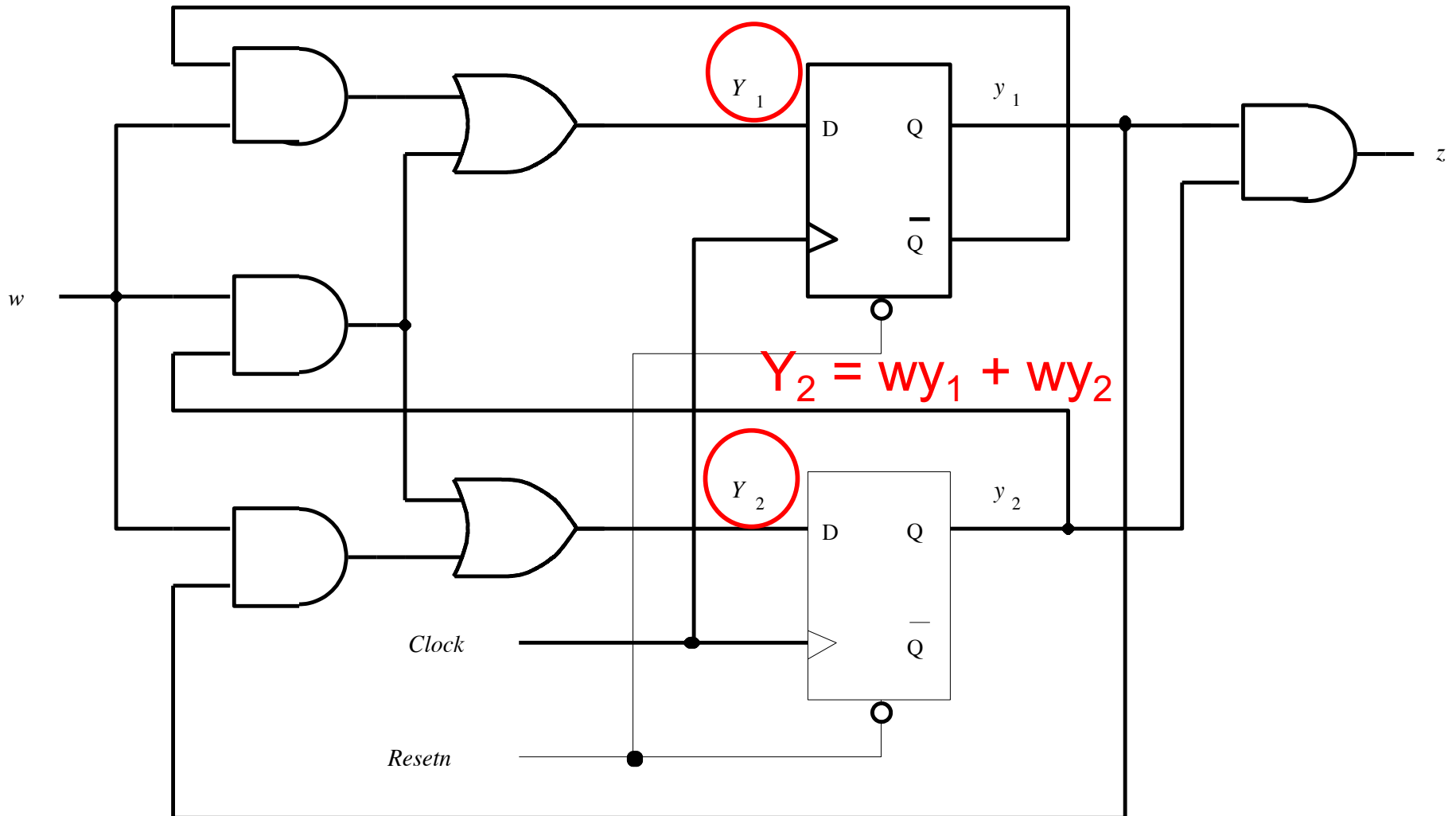


What are their logic expressions?



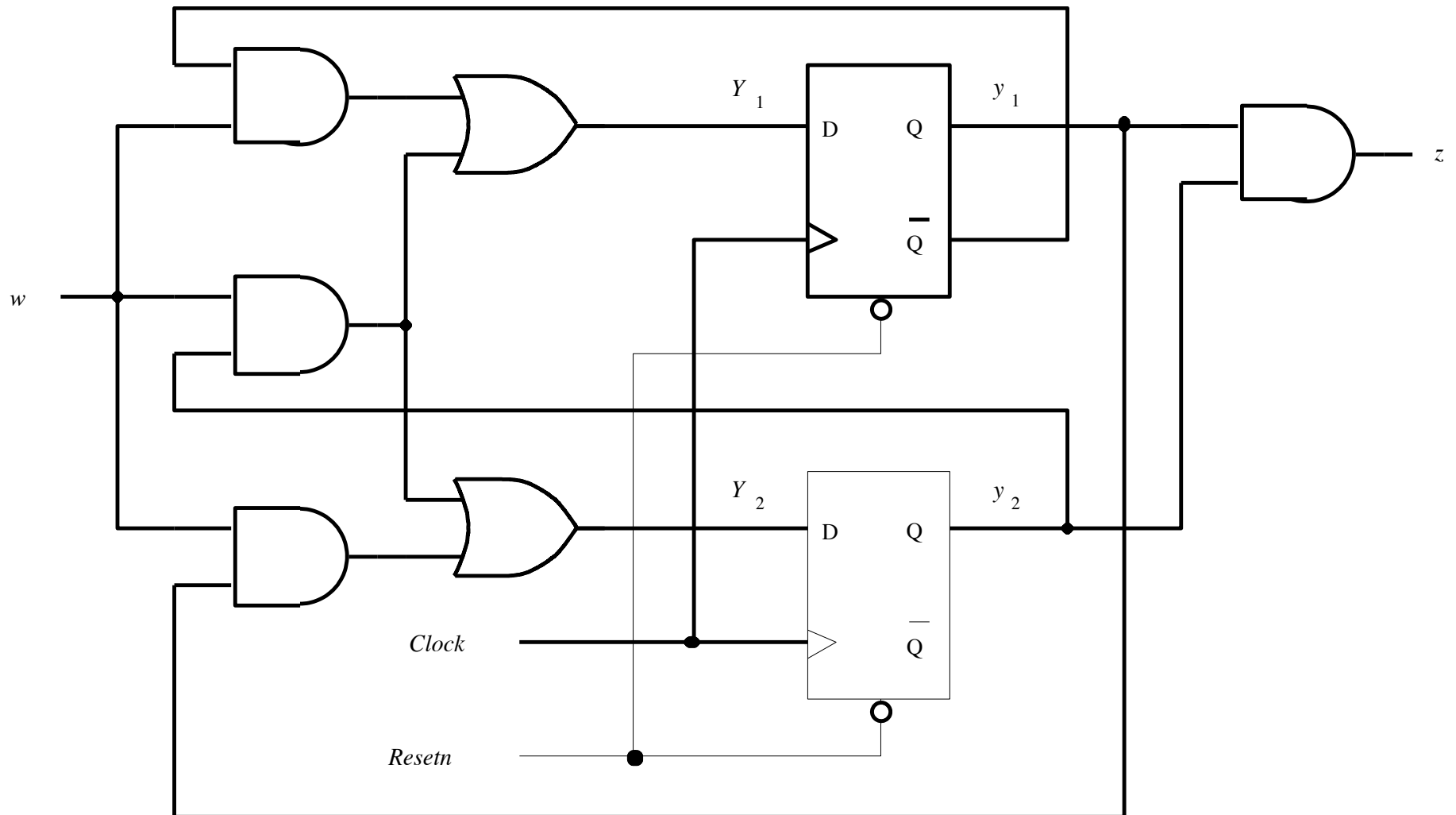
What are their logic expressions?

$$Y_1 = w\bar{y}_1 + wy_2$$

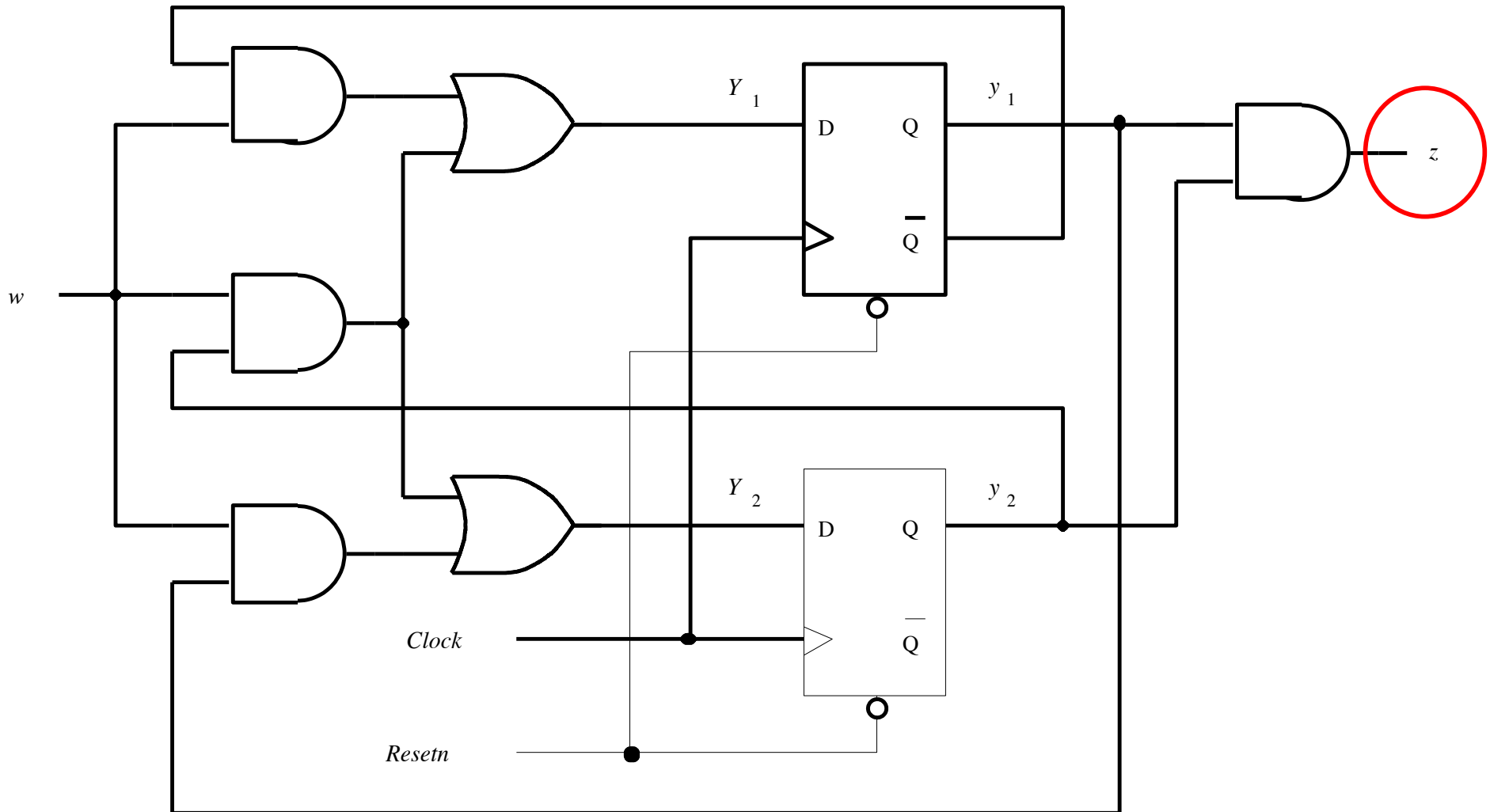


$$Y_2 = wy_1 + wy_2$$

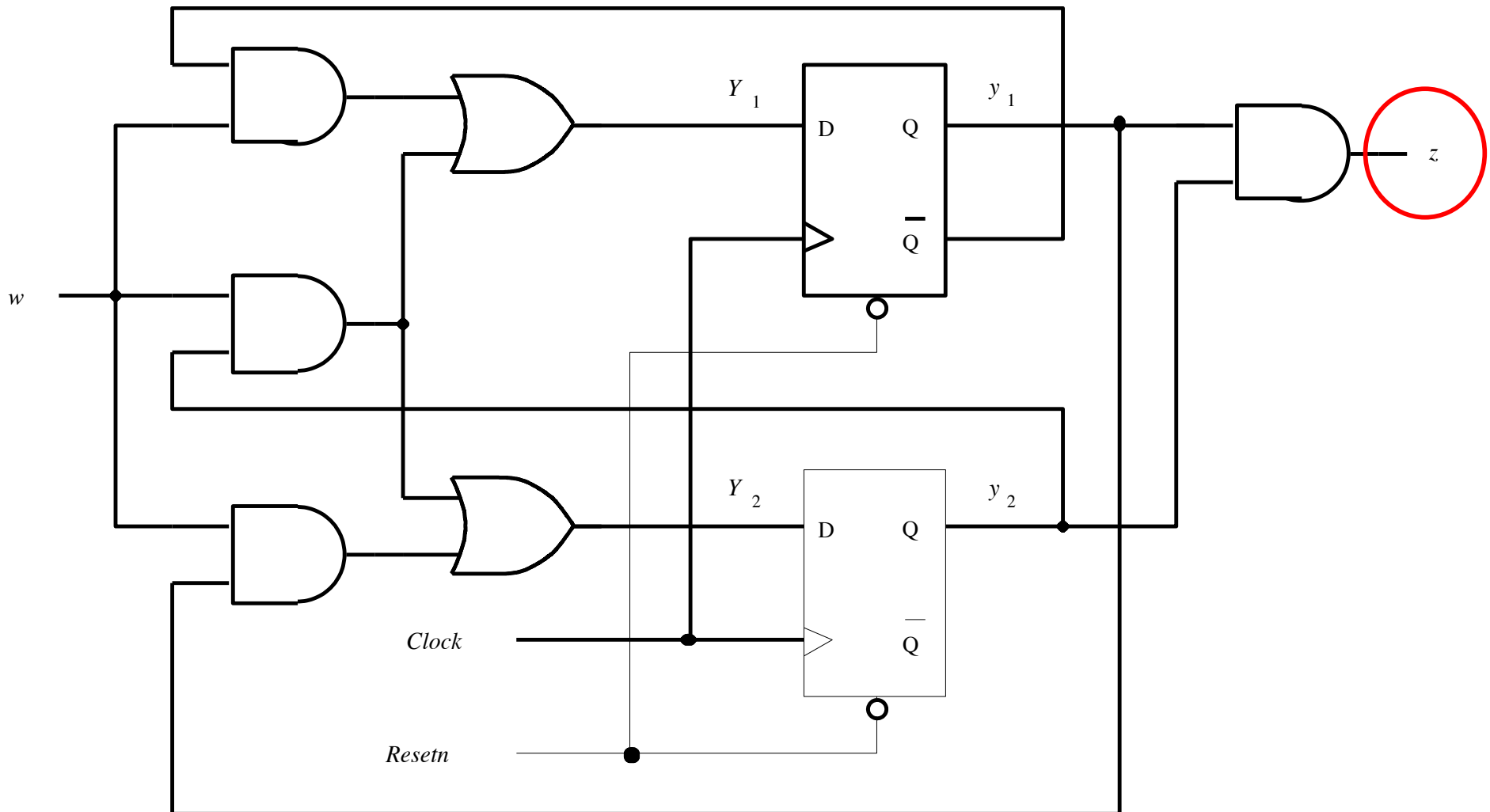
Where is the output, again?



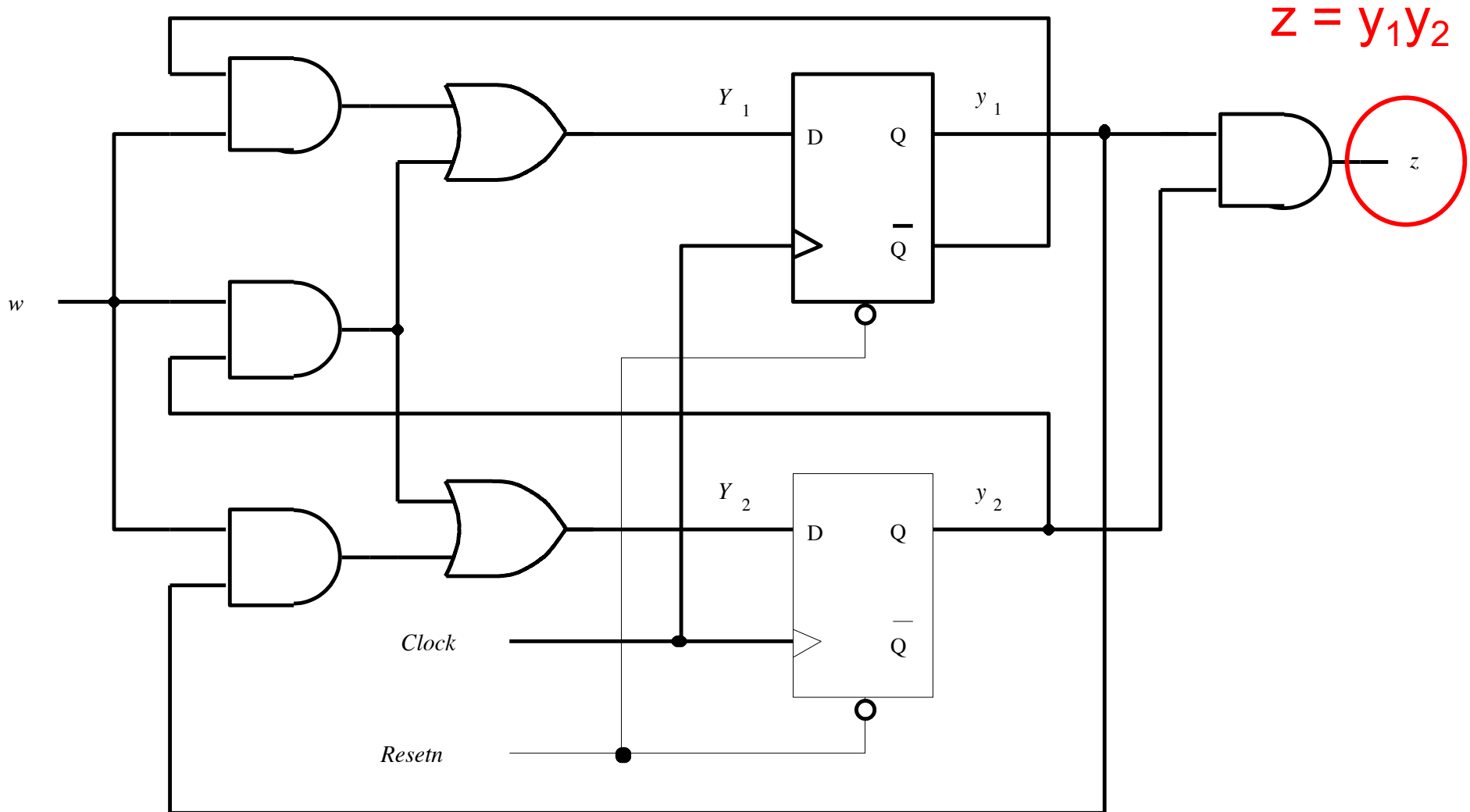
Where is the output, again?



What is its logic expression?



What is its logic expression?



**This is what we have to work with now
(we don't need the circuit anymore)**

$$Y_1 = w\bar{y}_1 + wy_2$$

$$Y_2 = wy_1 + wy_2$$

$$z = y_1y_2$$

Let's derive the state-assigned table

$$Y_1 = w\bar{y}_1 + wy_2$$

$$Y_2 = wy_1 + wy_2$$

$$z = y_1y_2$$

Present state y_2y_1	Next State		Output z
	$w = 0$	$w = 1$	
	Y_2Y_1	Y_2Y_1	
00			
01			
10			
11			

Let's derive the state-assigned table

$$Y_1 = w\bar{y}_1 + wy_2$$

$$Y_2 = wy_1 + wy_2$$

$$z = y_1y_2$$

Present state y_2y_1	Next State		Output z
	$w = 0$	$w = 1$	
00			
01			
10			
11			

Let's derive the state-assigned table

$$Y_1 = w\bar{y}_1 + wy_2$$

$$Y_2 = wy_1 + wy_2$$

$$z = y_1y_2$$

Present state y_2y_1	Next State		Output z
	$w = 0$	$w = 1$	
	Y_2Y_1	Y_2Y_1	
00			0
01			0
10			0
11			1

Let's derive the state-assigned table

$$Y_1 = w\bar{y}_1 + wy_2$$

$$Y_2 = wy_1 + wy_2$$

$$z = y_1y_2$$

Present state y_2y_1	Next State		Output z
	$w = 0$	$w = 1$	
		Y_2Y_1	Y_2Y_1
00			0
01			0
10			0
11			1

Let's derive the state-assigned table

$$Y_1 = w\bar{y}_1 + wy_2$$

$$Y_2 = wy_1 + wy_2$$

$$z = y_1y_2$$

Present state y_2y_1	Next State		Output z
	$w = 0$	$w = 1$	
	Y_2Y_1	Y_2Y_1	
00	0	1	0
01	0	0	0
10	0	1	0
11	0	1	1

Let's derive the state-assigned table

$$Y_1 = w\bar{y}_1 + wy_2$$

$$Y_2 = wy_1 + wy_2$$

$$z = y_1y_2$$

Present state	Next State		Output z
	w = 0	w = 1	
y ₂ y ₁	Y ₂ Y ₁	Y ₂ Y ₁	
00	0	1	0
01	0	0	0
10	0	1	0
11	0	1	1

Let's derive the state-assigned table

$$Y_1 = w\bar{y}_1 + wy_2$$

$$Y_2 = wy_1 + wy_2$$

$$z = y_1y_2$$

Present state y_2y_1	Next State		Output z
	$w = 0$	$w = 1$	
	Y_2Y_1	Y_2Y_1	
00	00	01	0
01	00	10	0
10	00	11	0
11	00	11	1

We don't need the logic expressions anymore

$$Y_1 = w\bar{y}_1 + wy_2$$

$$Y_2 = wy_1 + wy_2$$

$$z = y_1y_2$$

Present state y_2y_1	Next State		Output z
	$w = 0$	$w = 1$	
	Y_2Y_1	Y_2Y_1	
00	00	01	0
01	00	10	0
10	00	11	0
11	00	11	1

We don't need the logic expressions anymore

Present state y_2y_1	Next State		Output z
	$w = 0$	$w = 1$	
	Y_2Y_1	Y_2Y_1	
00	00	01	0
01	00	10	0
10	00	11	0
11	00	11	1

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	

State table

Present state y_2y_1	Next State		Output z
	w = 0	w = 1	
	Y_2Y_1	Y_2Y_1	
00	00	01	0
01	00	10	0
10	00	11	0
11	00	11	1

State-assigned table

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	

State table

Present state y_2y_1	Next State		Output z
	w = 0	w = 1	
	Y_2Y_1	Y_2Y_1	
00	00	01	0
01	00	10	0
10	00	11	0
11	00	11	1

State-assigned table

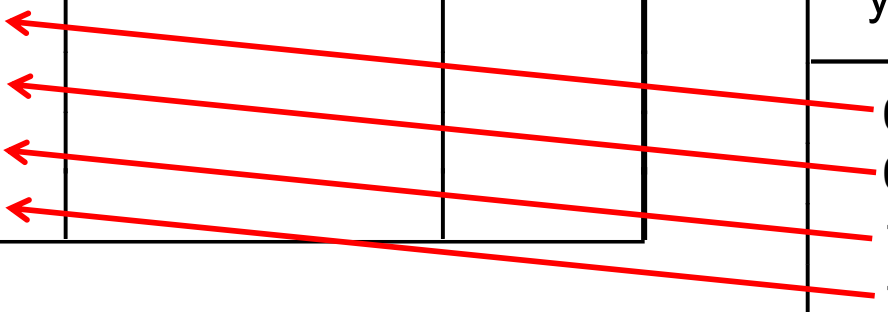
Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A			
B			
C			
D			

State table

Present state y_2y_1	Next State		Output z
	w = 0 Y_2Y_1	w = 1 Y_2Y_1	
00	00	01	0
01	00	10	0
10	00	11	0
11	00	11	1

State-assigned table



Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A			
B			
C			
D			

State table

Present state y_2y_1	Next State		Output z
	w = 0	w = 1	
00	00	01	0
01	00	10	0
10	00	11	0
11	00	11	1

State-assigned table

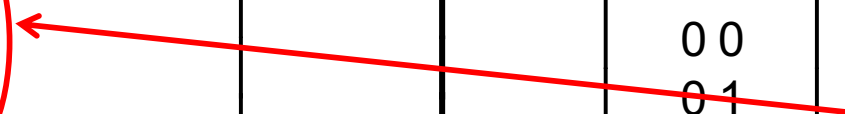
Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A	A		
B	A		
C	A		
D	A		

State table

Present state y_2y_1	Next State		Output z
	w = 0 Y_2Y_1	w = 1 Y_2Y_1	
00	00	01	0
01	00	10	0
10	00	11	0
11	00	11	1

State-assigned table



Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A	A		
B	A		
C	A		
D	A		

State table

Present state y_2y_1	Next State		Output z
	w = 0 Y_2Y_1	w = 1 Y_2Y_1	
00	00	01	0
01	00	10	0
10	00	11	0
11	00	11	1

State-assigned table

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A	A	B	
B	A	C	
C	A	D	
D	A	D	

State table

Present state	Next State		Output z
	w = 0	w = 1	
y_2y_1	Y_2Y_1	Y_2Y_1	
00	00	01	0
01	00	10	0
10	00	11	0
11	00	11	1

State-assigned table



Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A	A	B	
B	A	C	
C	A	D	
D	A	D	

State table

Present state y_2y_1	Next State		Output z
	w = 0	w = 1	
	Y_2Y_1	Y_2Y_1	
00	00	01	0
01	00	10	0
10	00	11	0
11	00	11	1

State-assigned table

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A	A	B	
B	A	C	
C	A	D	
D	A	D	

State table

Present state y_2y_1	Next State		Output z
	w = 0	w = 1	
	Y_2Y_1	Y_2Y_1	
00	00	01	0
01	00	10	0
10	00	11	0
11	00	11	1

State-assigned table

The output is the same in both tables

The two tables for the initial circuit

Present state	Next state		Output z
	w = 0	w = 1	
A	A	B	0
B	A	C	0
C	A	D	0
D	A	D	1

State table

Present state y_2y_1	Next State		Output z
	w = 0 Y_2Y_1	w = 1 Y_2Y_1	
00	00	01	0
01	00	10	0
10	00	11	0
11	00	11	1

State-assigned table

We don't need the state-assigned table anymore

Present state	Next state		Output z
	w = 0	w = 1	
A	A	B	0
B	A	C	0
C	A	D	0
D	A	D	1

State table

Present state y_2y_1	Next State		Output z
	w = 0 Y_2Y_1	w = 1 Y_2Y_1	
00	00	01	0
01	00	10	0
10	00	11	0
11	00	11	1

State-assigned table

We don't need the state-assigned table anymore

Present state	Next state		Output z
	w = 0	w = 1	
A	A	B	0
B	A	C	0
C	A	D	0
D	A	D	1

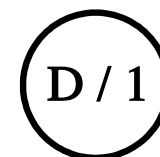
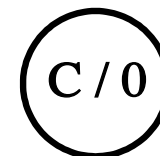
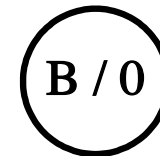
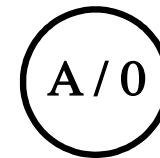
State table

Let's Draw the State Diagram

Present state	Next state		Output z
	w = 0	w = 1	
A	A	B	0
B	A	C	0
C	A	D	0
D	A	D	1

Let's Draw the State Diagram

Present state	Next state		Output z
	w = 0	w = 1	
A	A	B	0
B	A	C	0
C	A	D	0
D	A	D	1



Because this is a Moore machine
the output is tied to the state

Let's Draw the State Diagram

Present state	Next state		Output z
	w = 0	w = 1	
A	A	B	0
B	A	C	0
C	A	D	0
D	A	D	1

A / 0

B / 0

C / 0

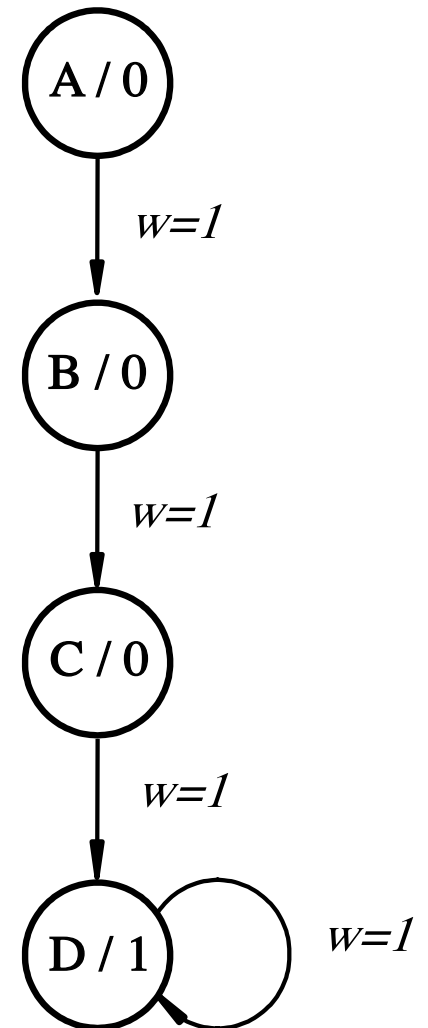
D / 1

All transitions when the input w is equal to 1

Let's Draw the State Diagram

Present state	Next state		Output z
	w = 0	w = 1	
A	A	B	0
B	A	C	0
C	A	D	0
D	A	D	1

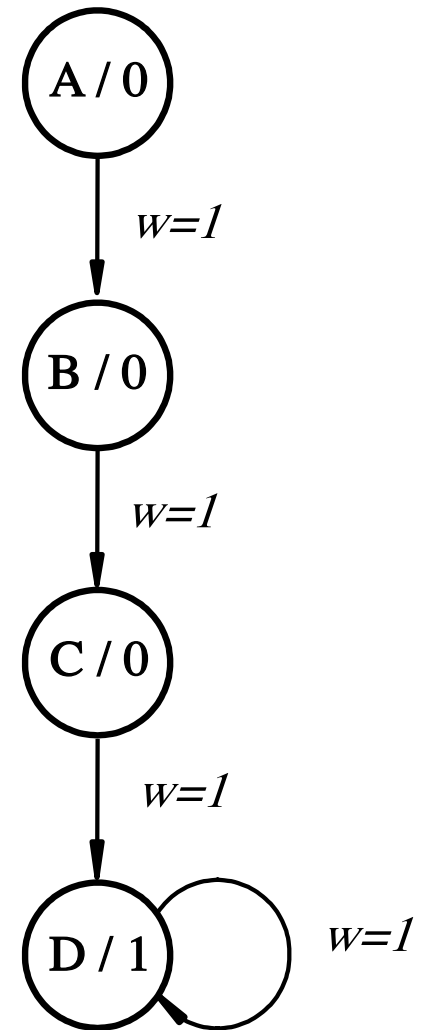
All transitions when the input w is equal to 1



Let's Draw the State Diagram

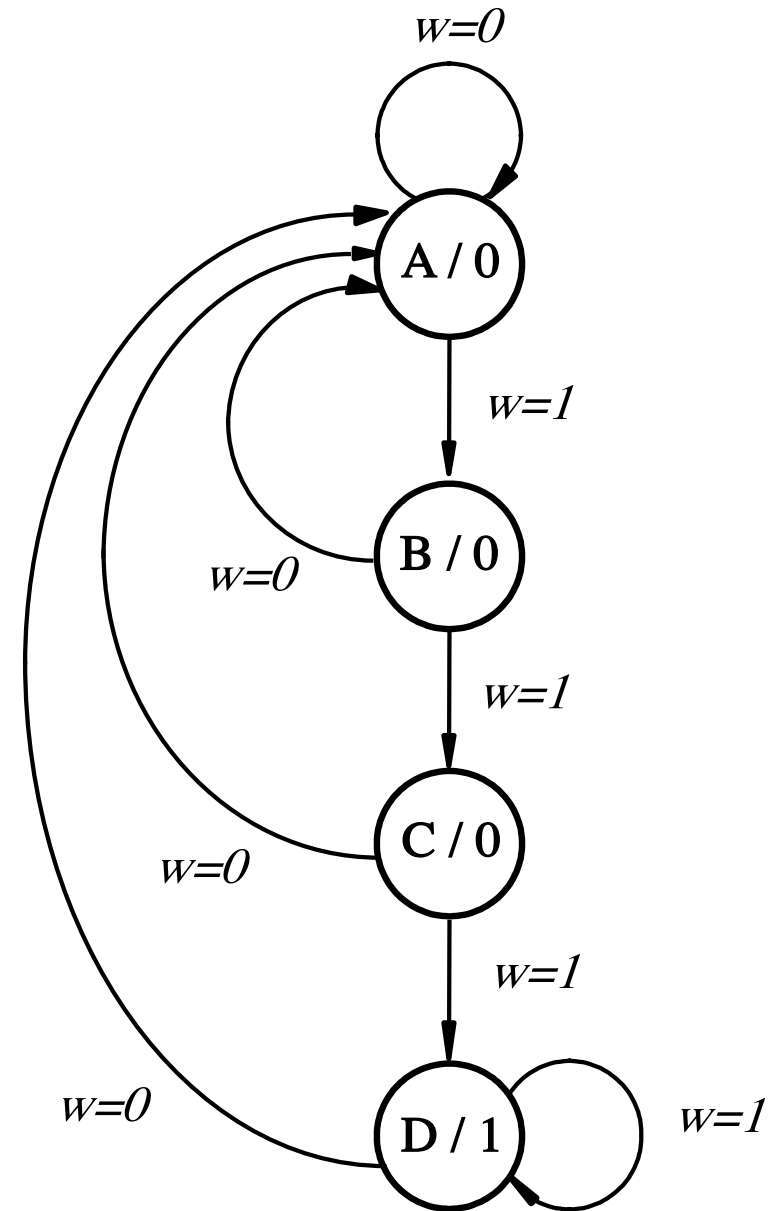
Present state	Next state		Output z
	w = 0	w = 1	
A	A	B	0
B	A	C	0
C	A	D	0
D	A	D	1

All transitions when the input w is equal to 0



Let's Draw the State Diagram

Present state	Next state		Output z
	w = 0	w = 1	
A	A	B	0
B	A	C	0
C	A	D	0
D	A	D	1

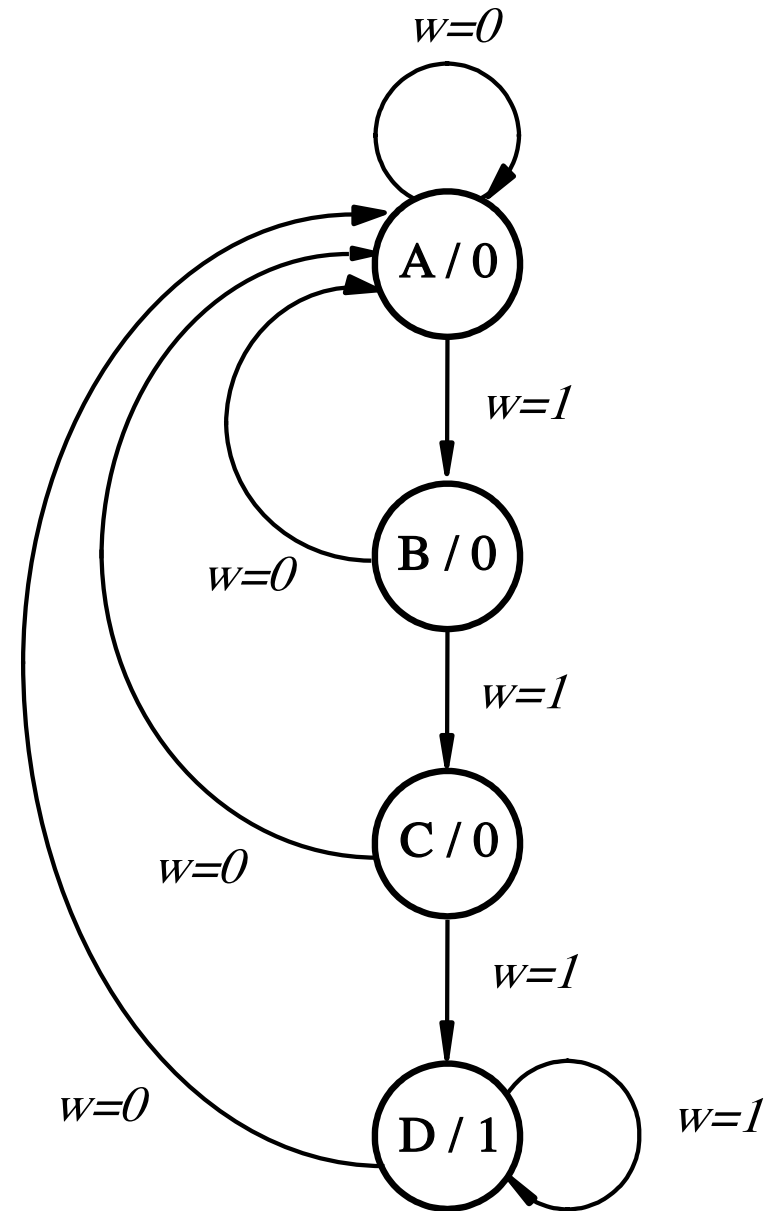


All transitions when the input w is equal to 0

We are done!

Present state	Next state		Output z
	w = 0	w = 1	
A	A	B	0
B	A	C	0
C	A	D	0
D	A	D	1

State table

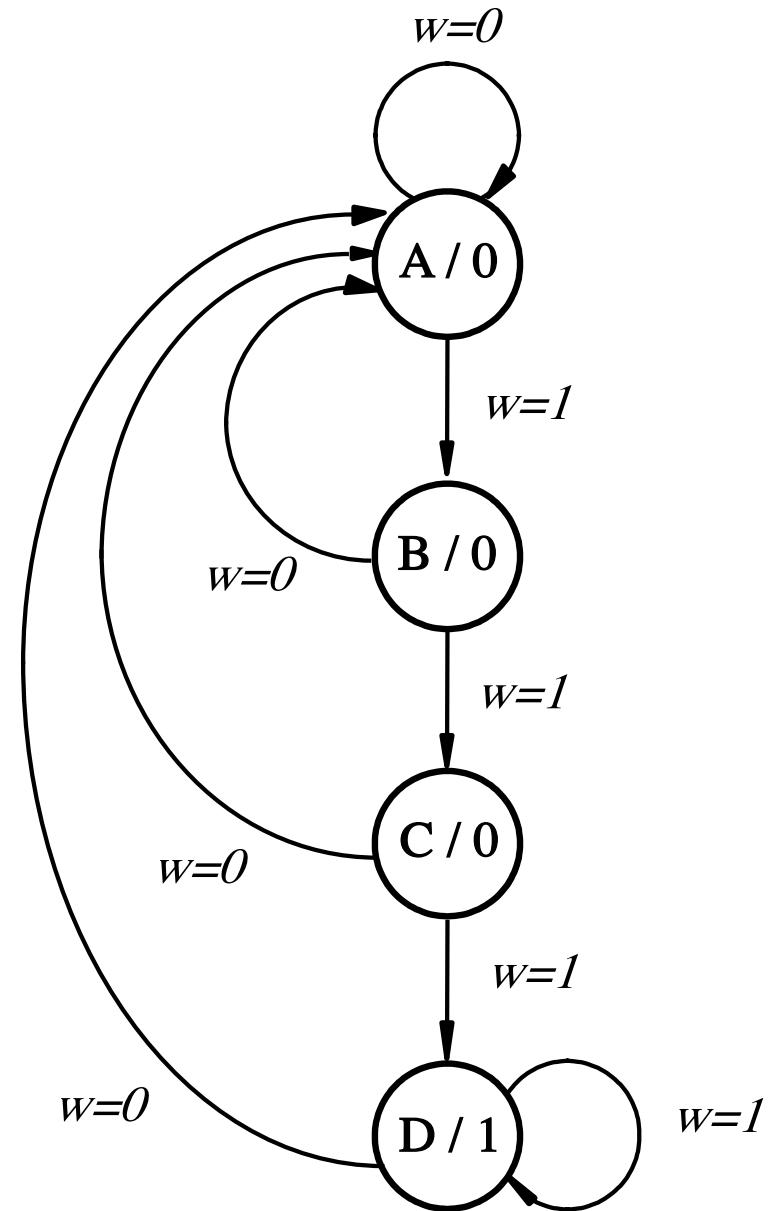


State diagram

Almost done. What does this FSM do?

Present state	Next state		Output z
	w = 0	w = 1	
A	A	B	0
B	A	C	0
C	A	D	0
D	A	D	1

State table



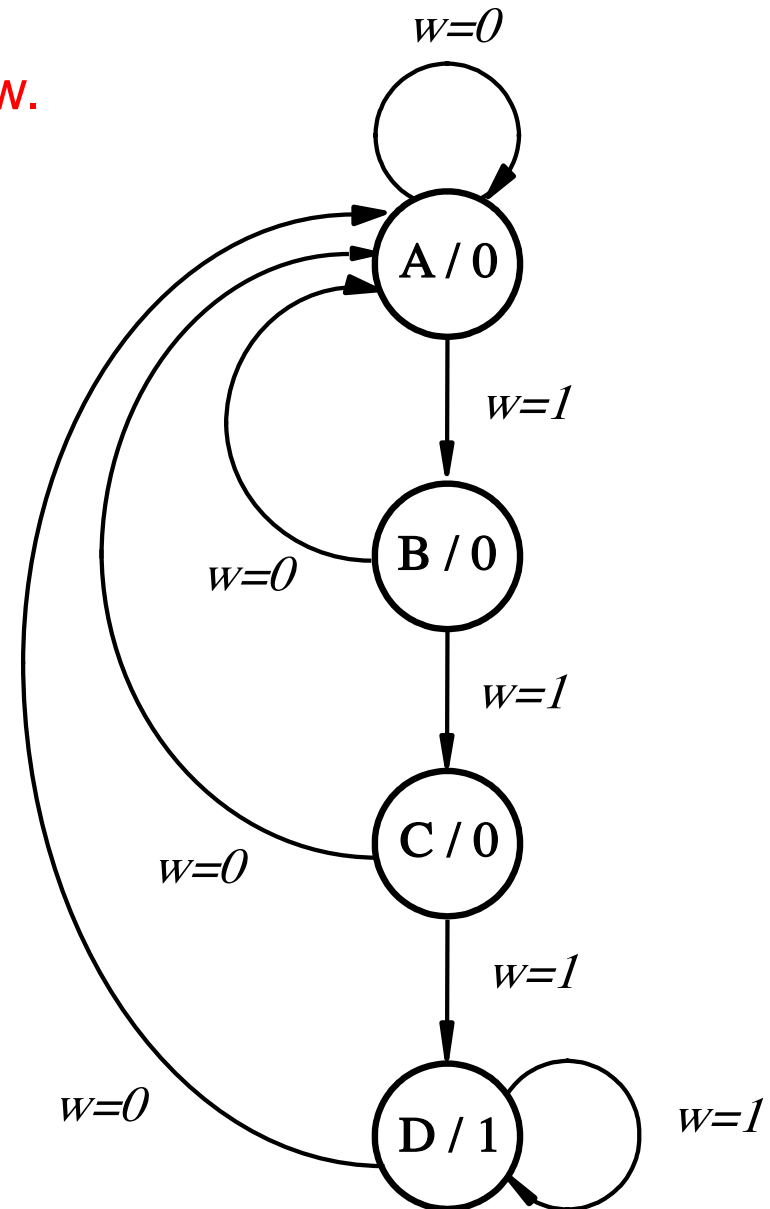
State diagram

Almost done. What does this FSM do?

It sets the output z to 1 when three consecutive 1's occur on the input w . In other words, it is a sequence detector for the input pattern 111.

Present state	Next state		Output z
	$w = 0$	$w = 1$	
A	A	B	0
B	A	C	0
C	A	D	0
D	A	D	1

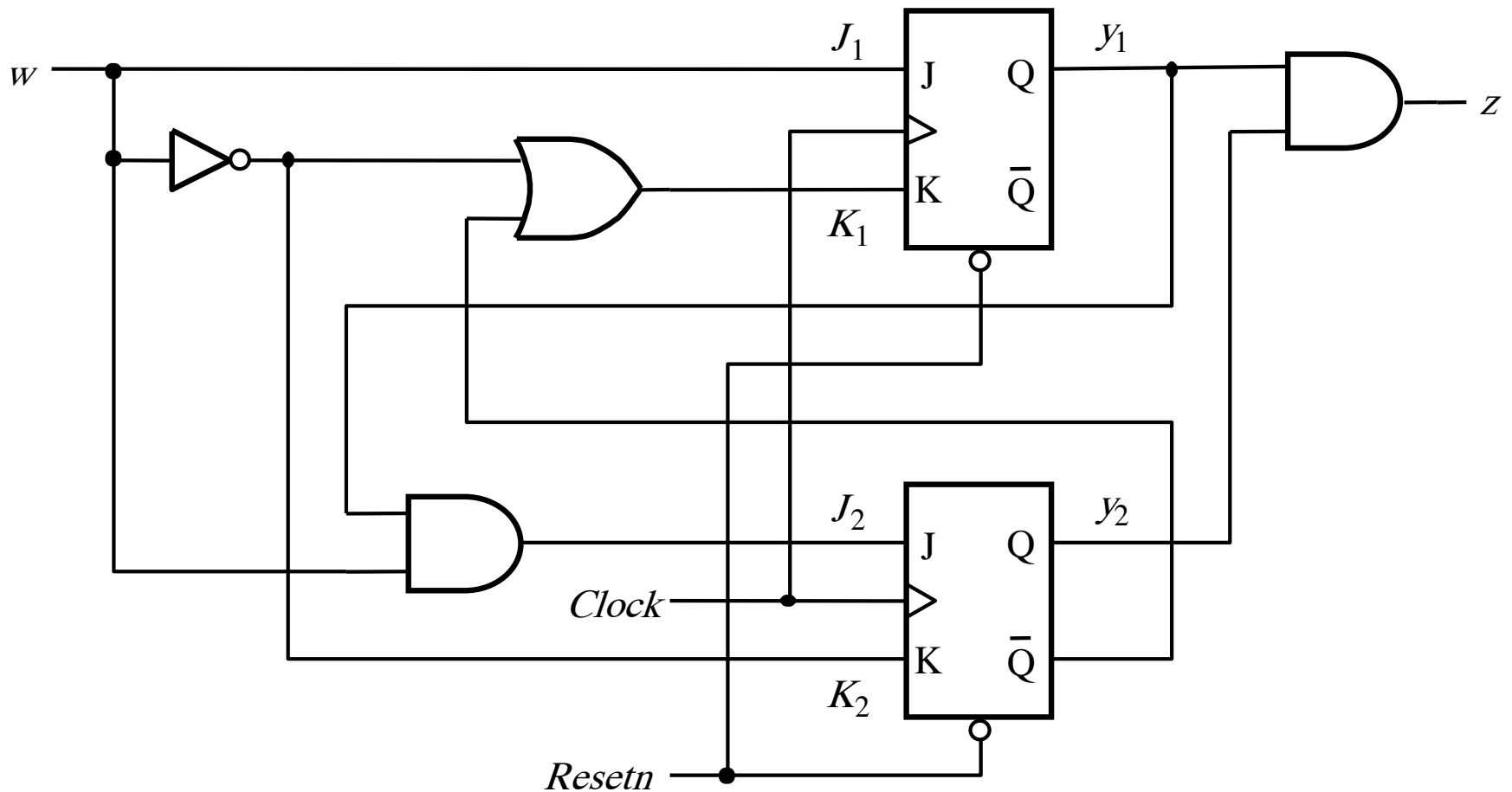
State table



State diagram

Another Example (with JK flip-flops)

What does this circuit do?

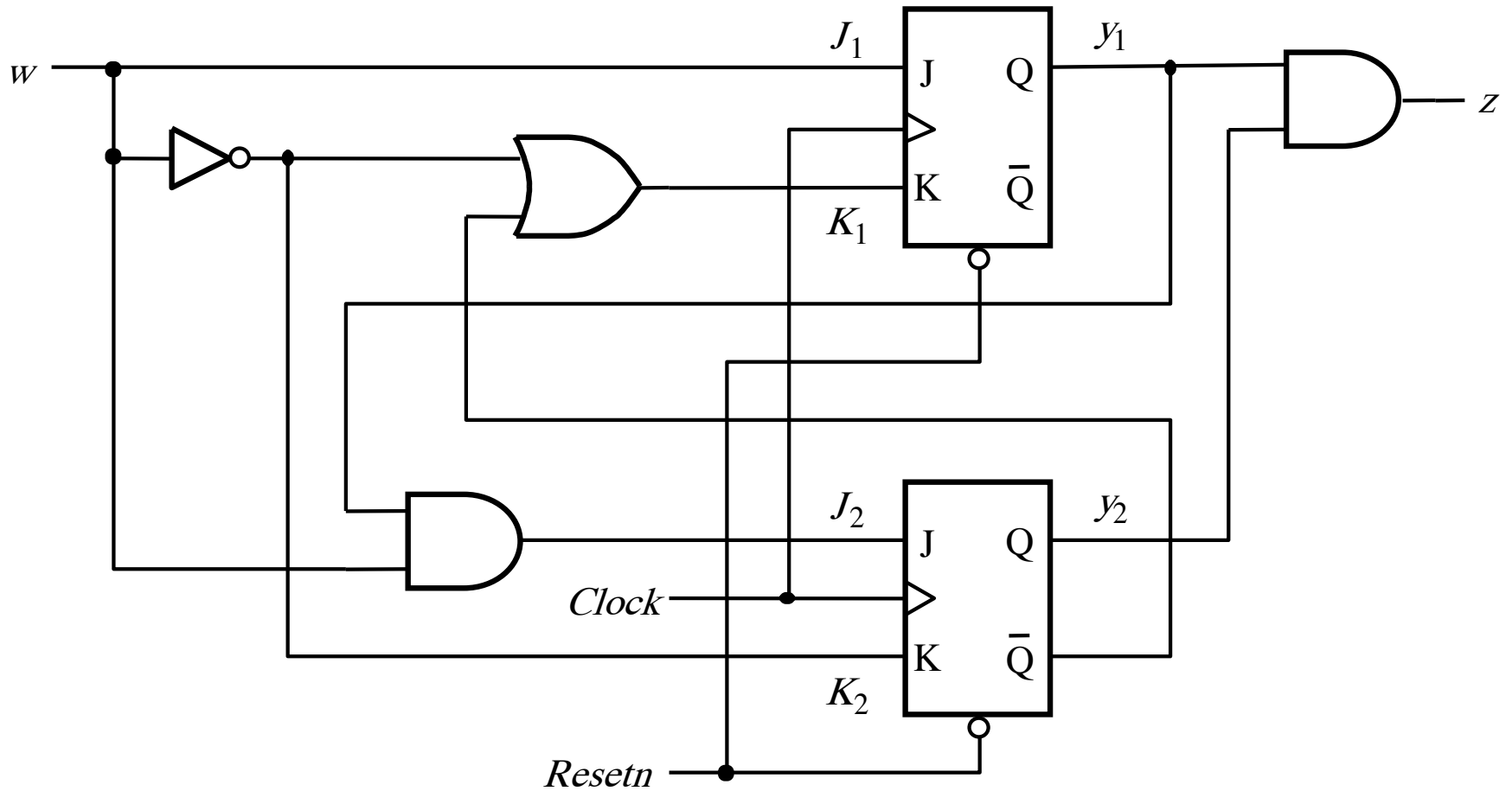


[Figure 6.77 from the textbook]

Approach

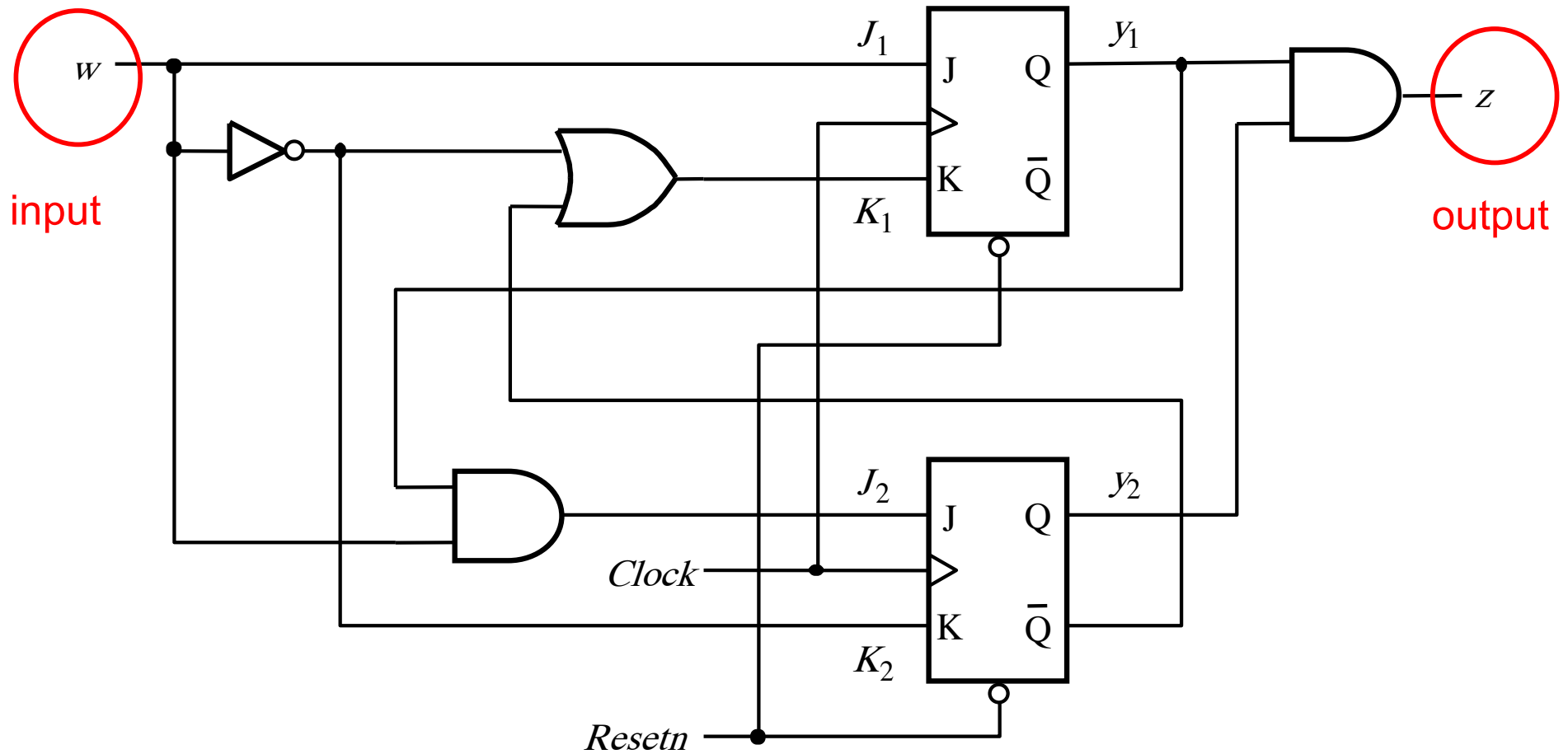
- **Find the flip-flops**
- **Outputs of the flip-flops = present state variables**
- **Inputs of the flip-flops determine the next state variables**
- **Determine the logical expressions for the outputs**
- **Given this info it is easy to do the state-assigned table**
- **Next do the state table**
- **Finally, draw the state diagram.**

Where are the inputs and outputs?

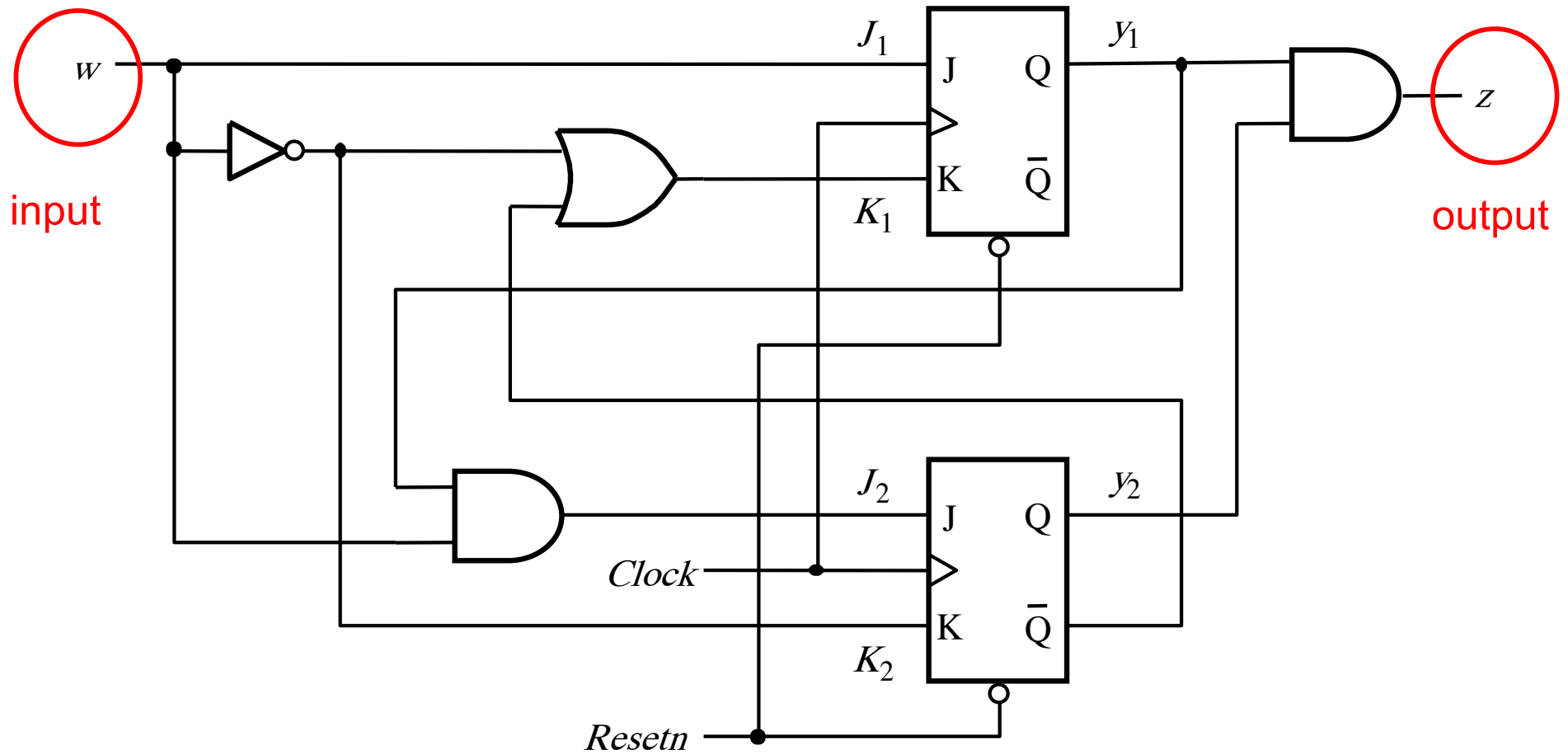


[Figure 6.77 from the textbook]

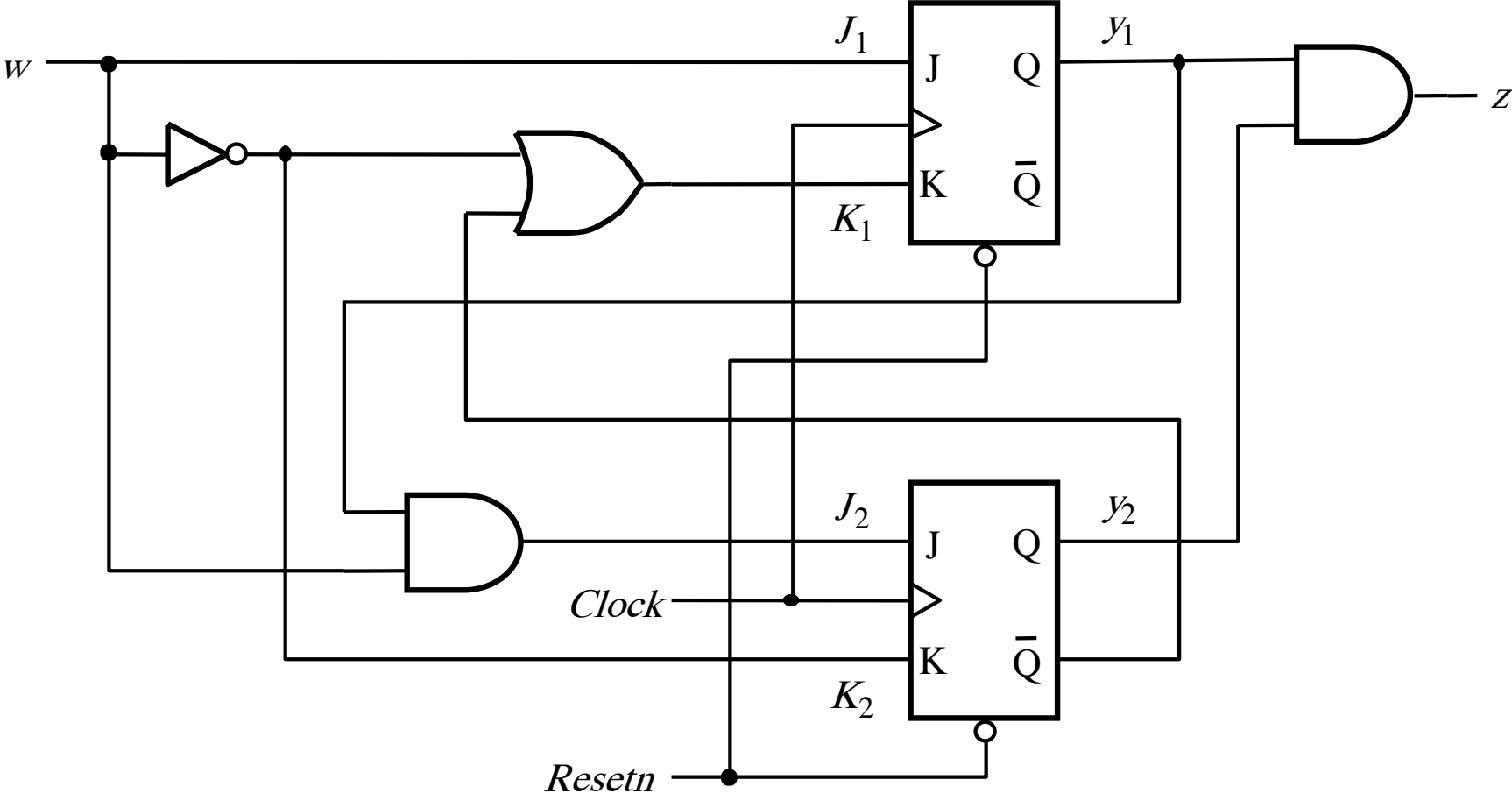
Where are the inputs and outputs?



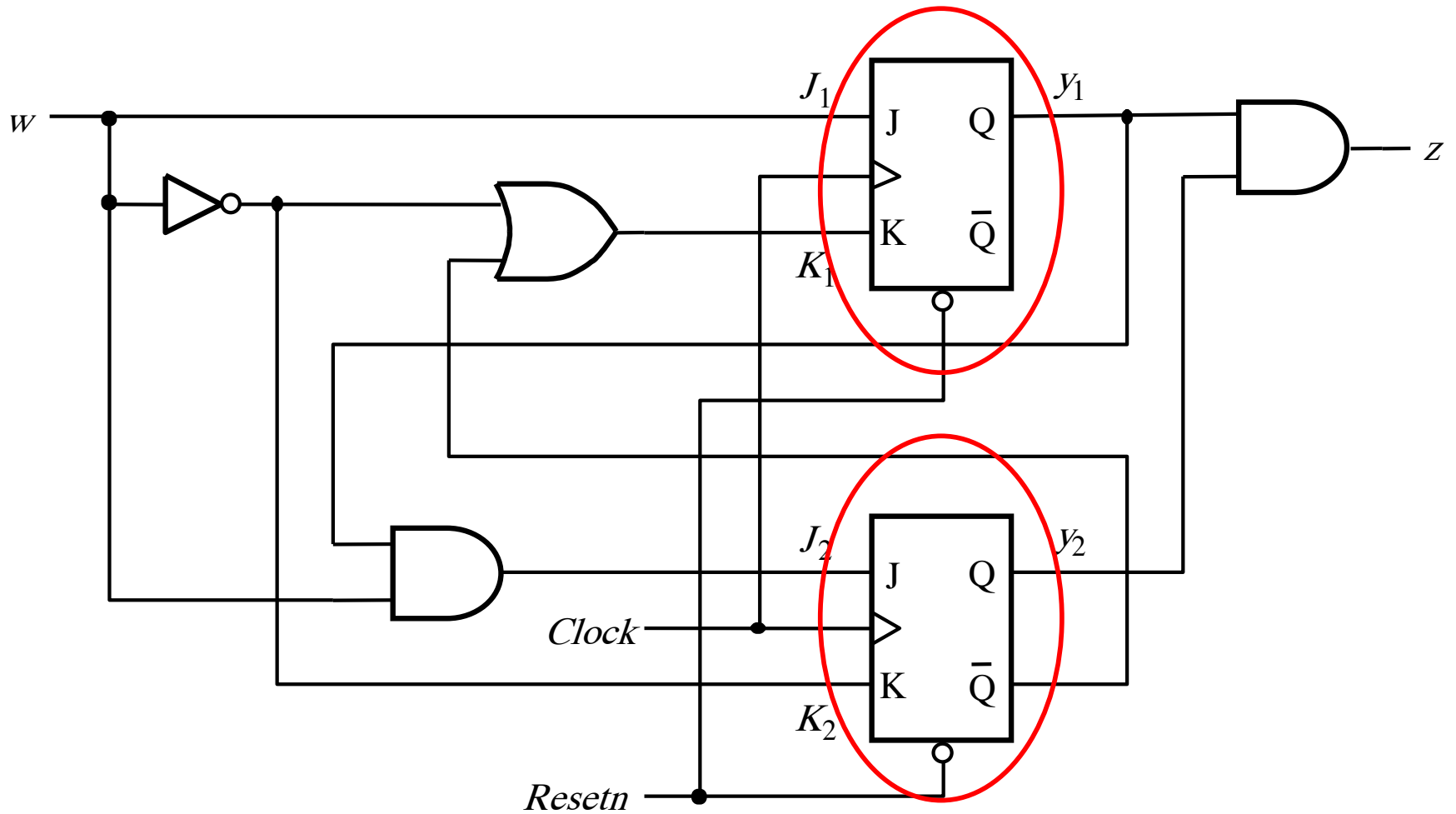
What kind of machine is this?



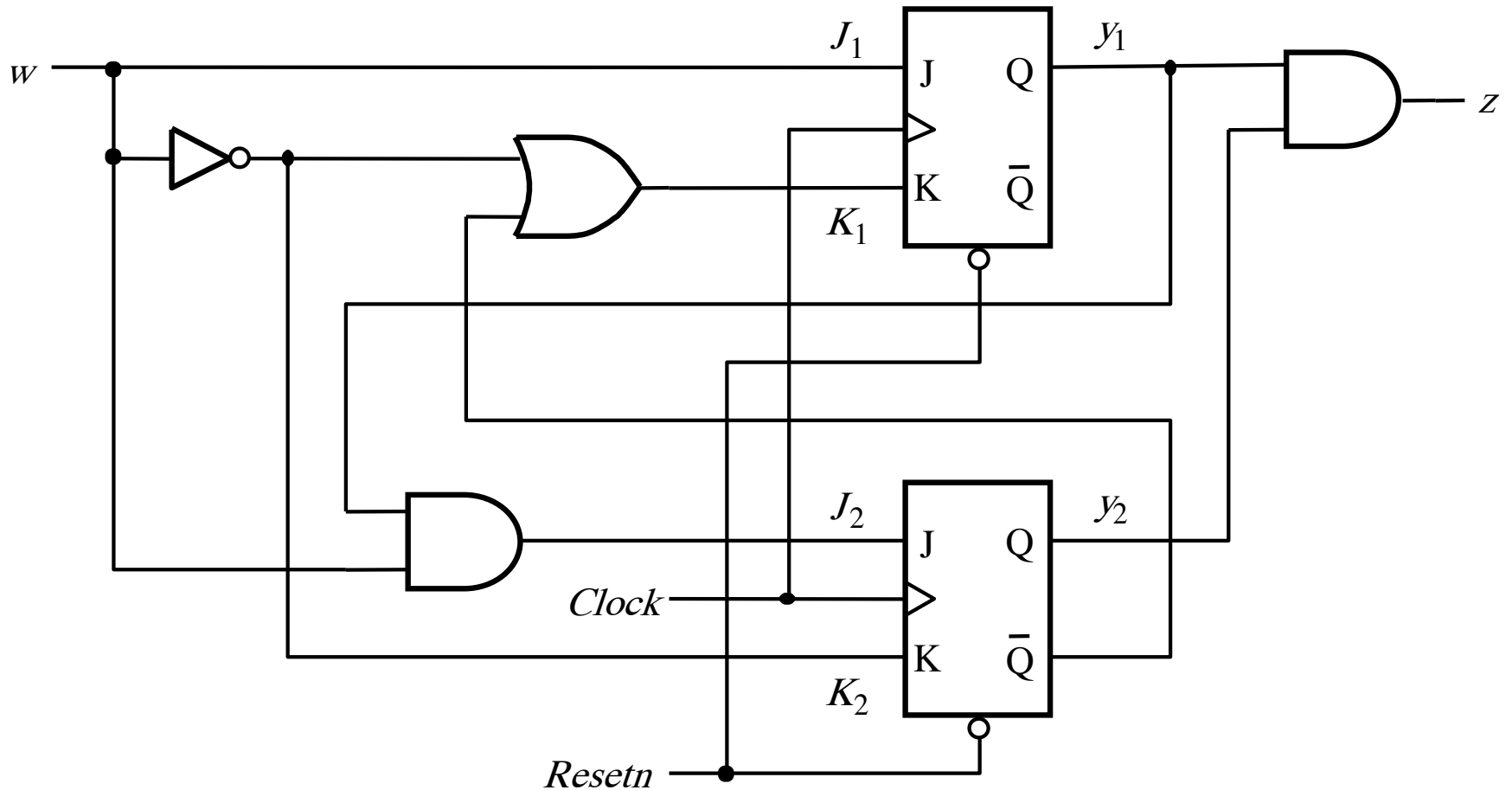
Where are the flip-flops?



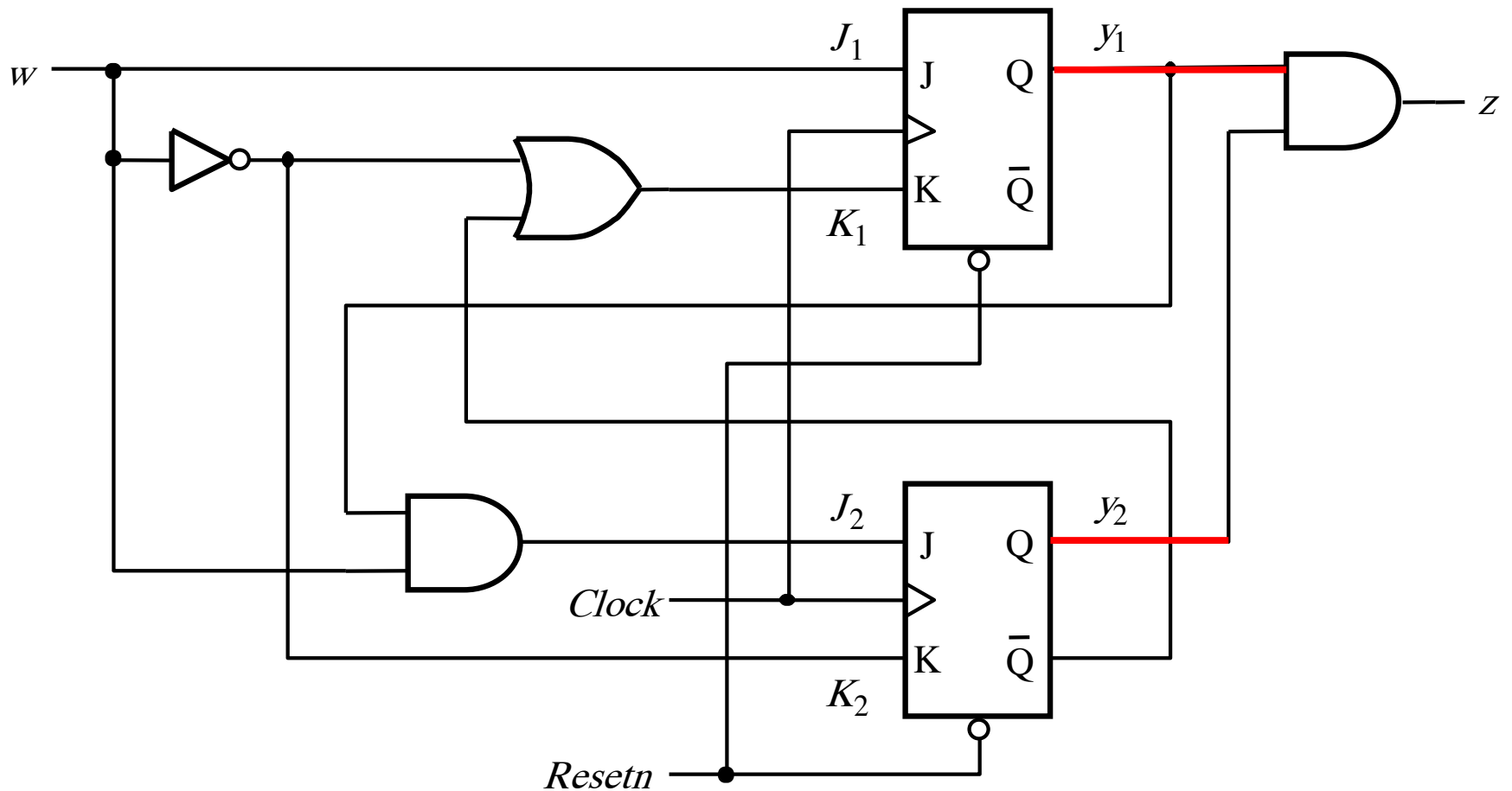
Where are the flip-flops?



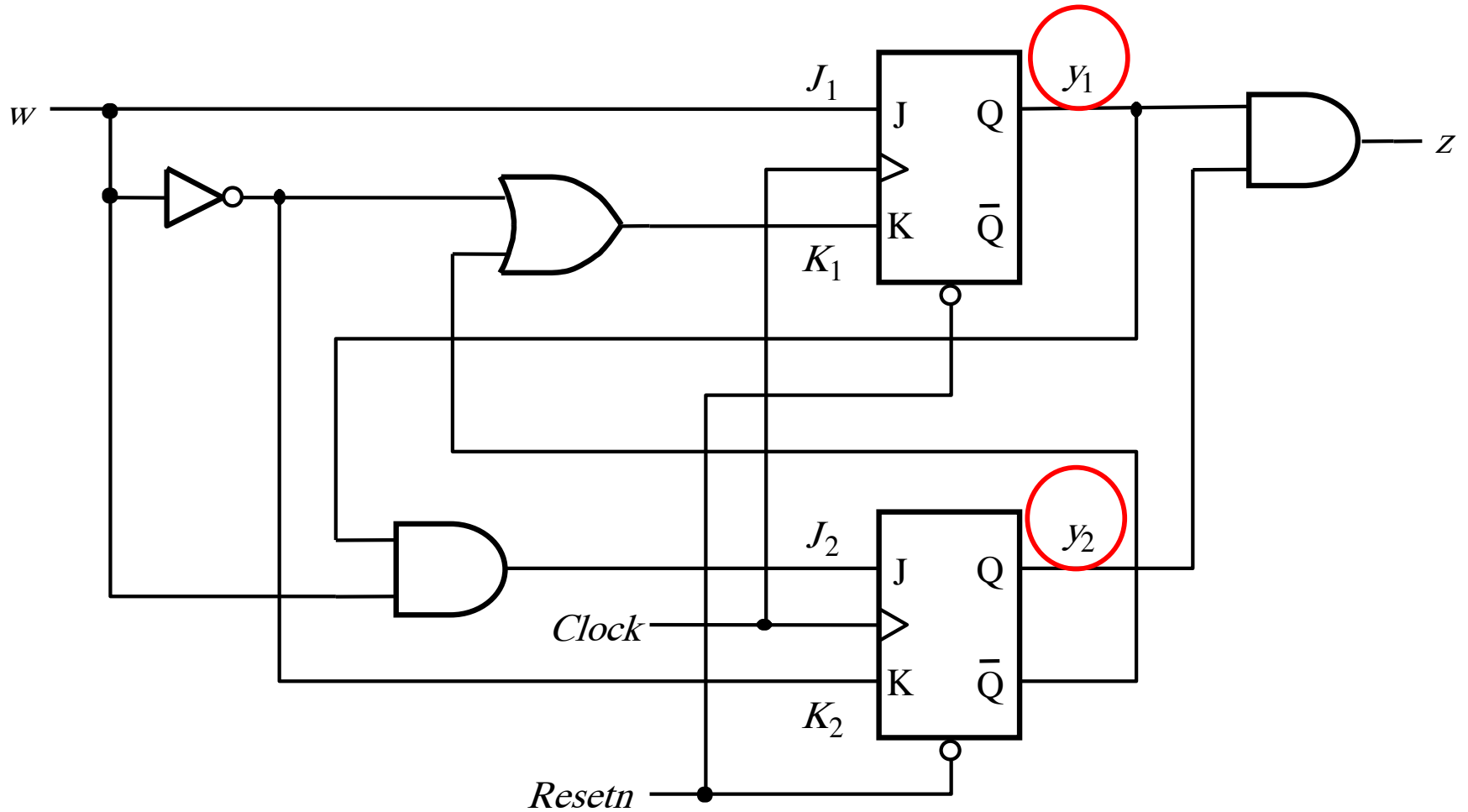
Where are the outputs of the flip-flops?



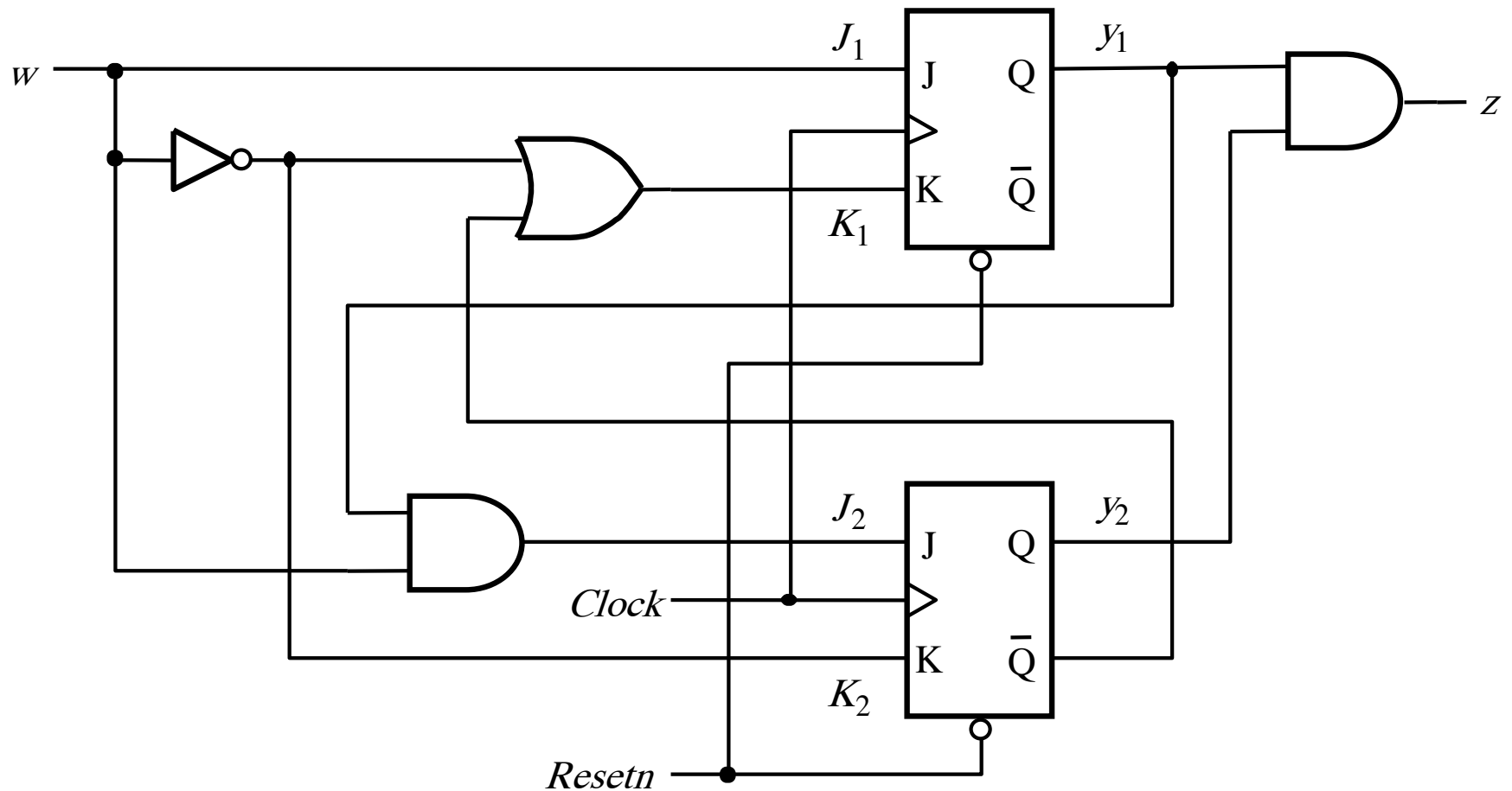
Where are the outputs of the flip-flops?



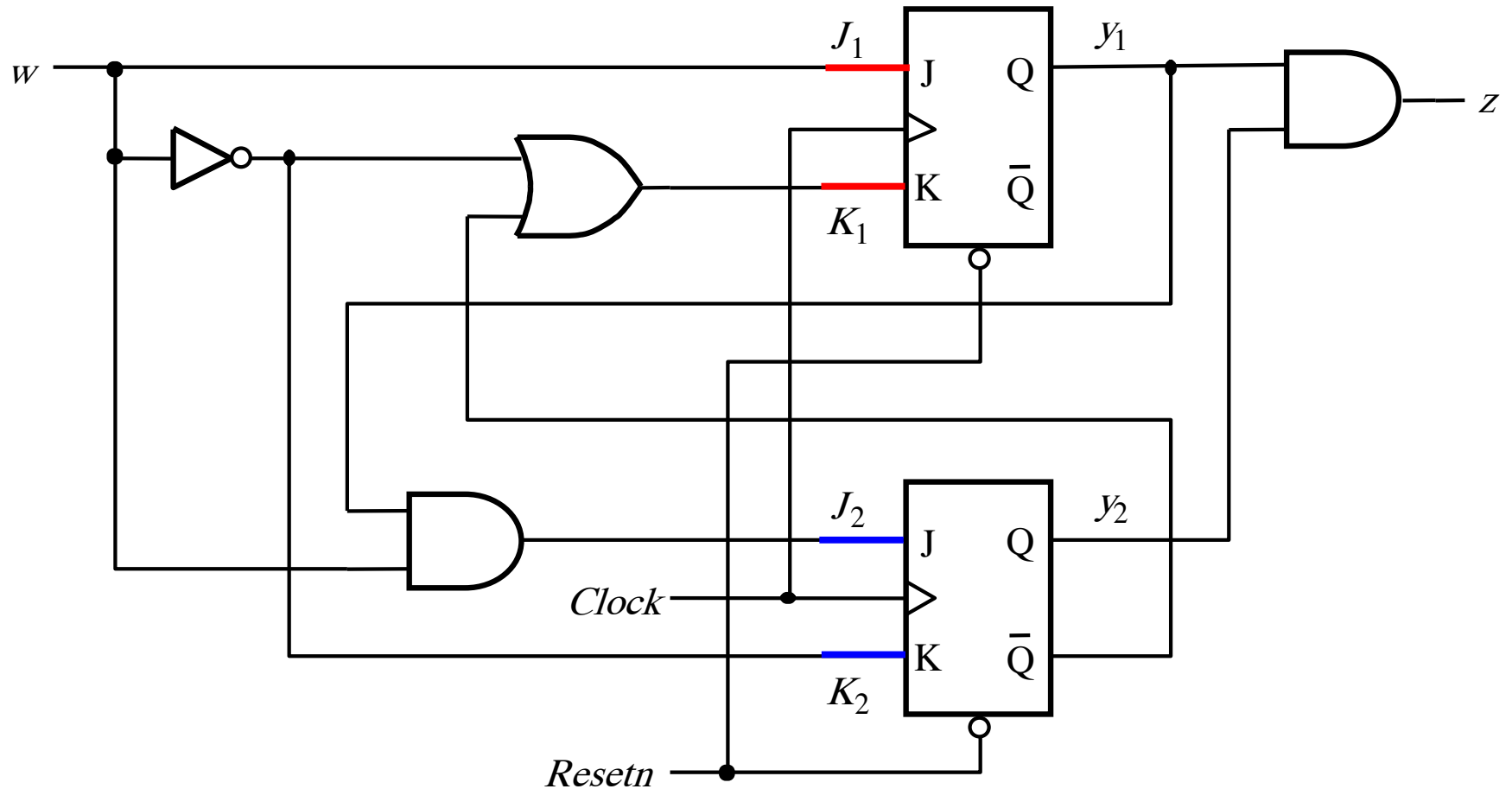
These are the next-state variables



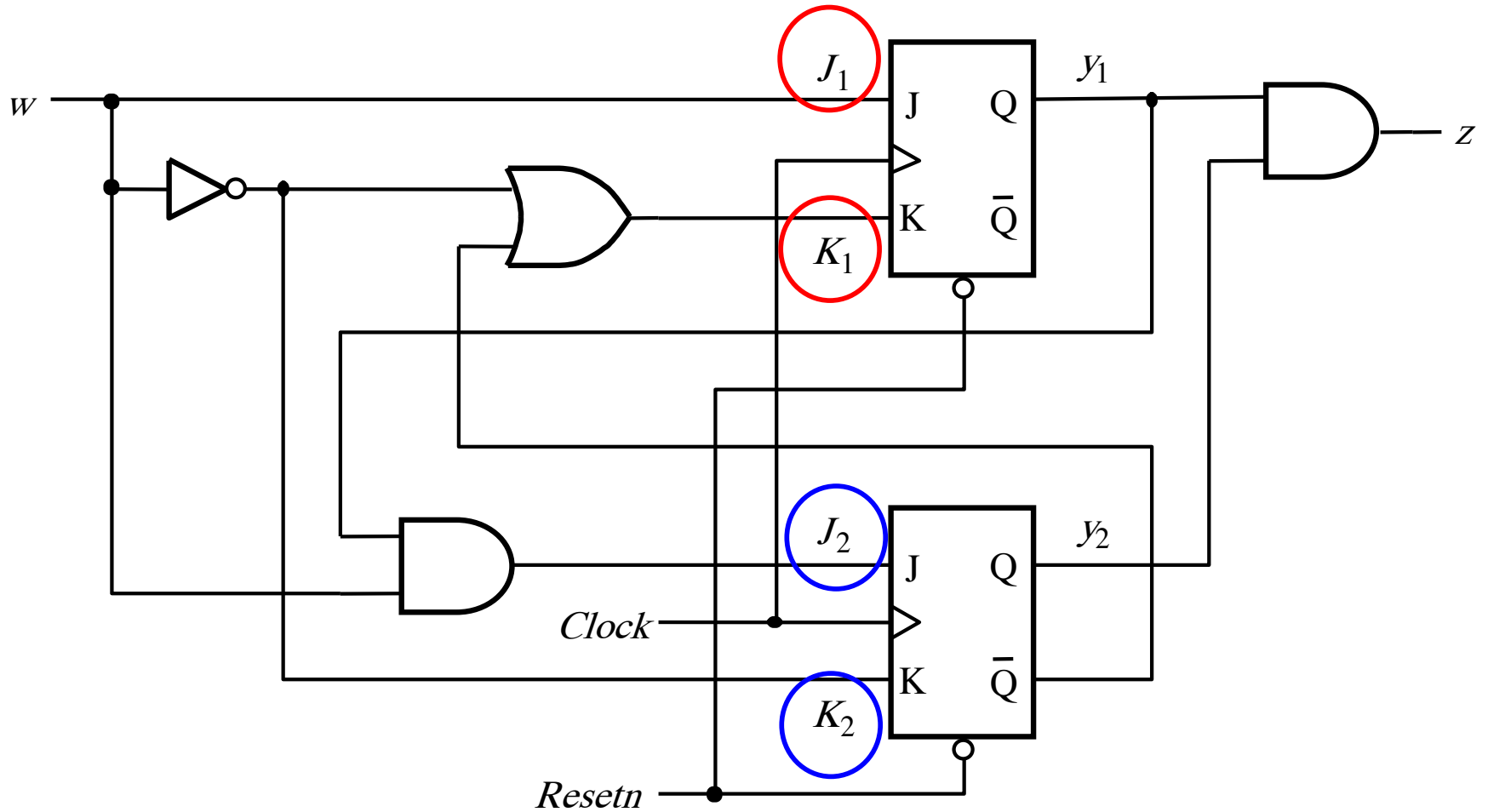
Where are the inputs of the flip-flops?



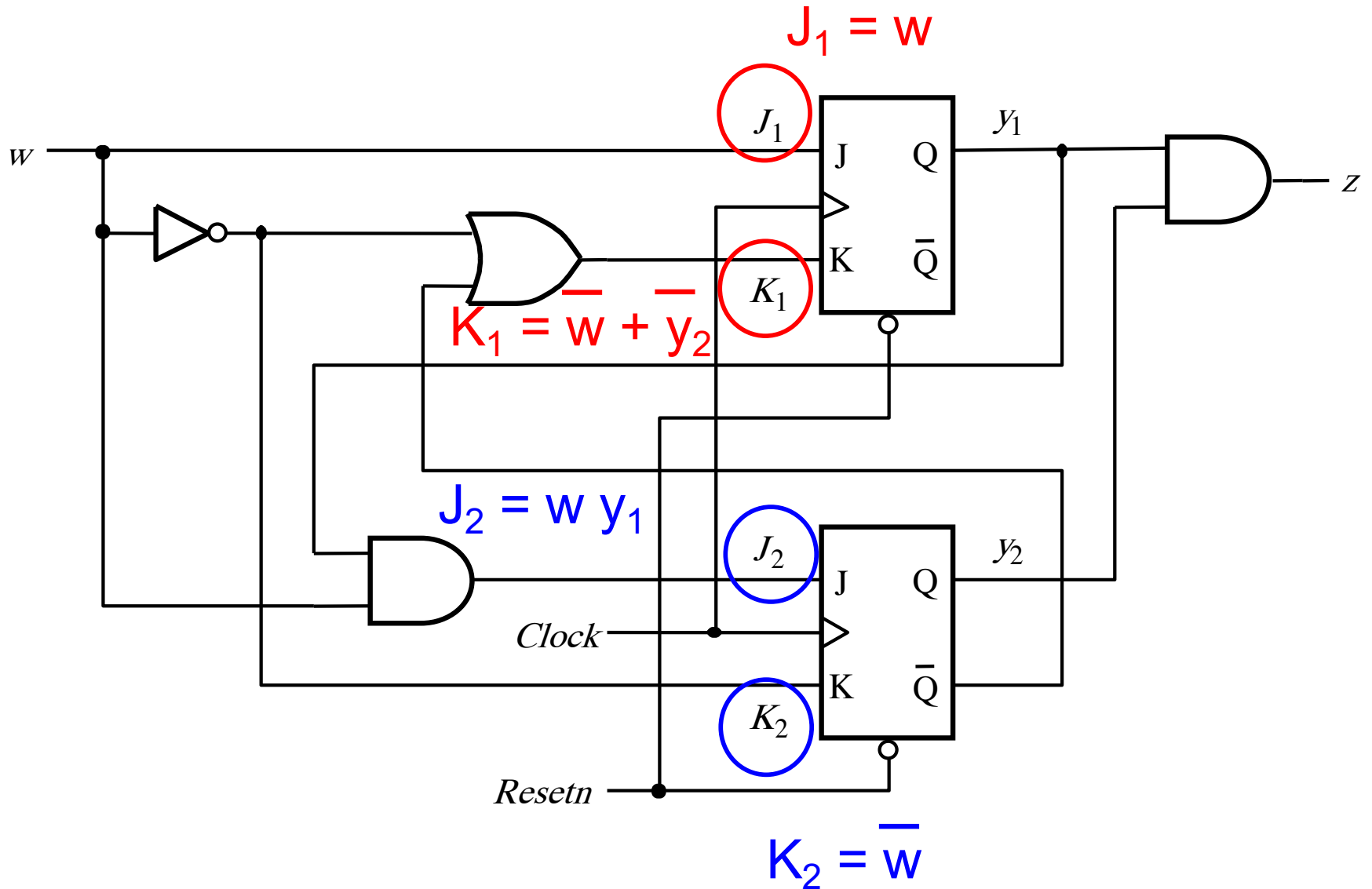
Where are the inputs of the flip-flops?



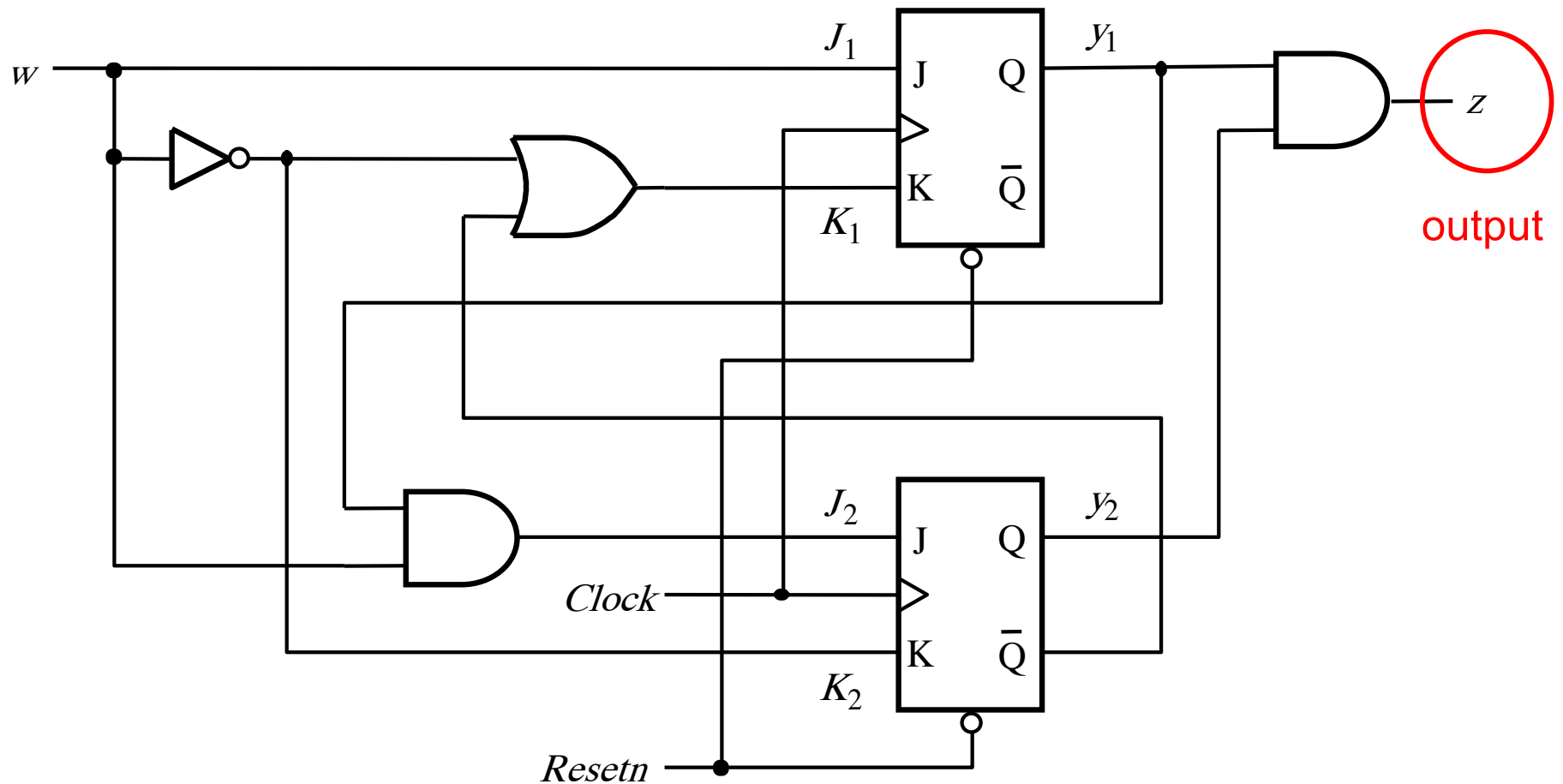
What are their logic expressions?



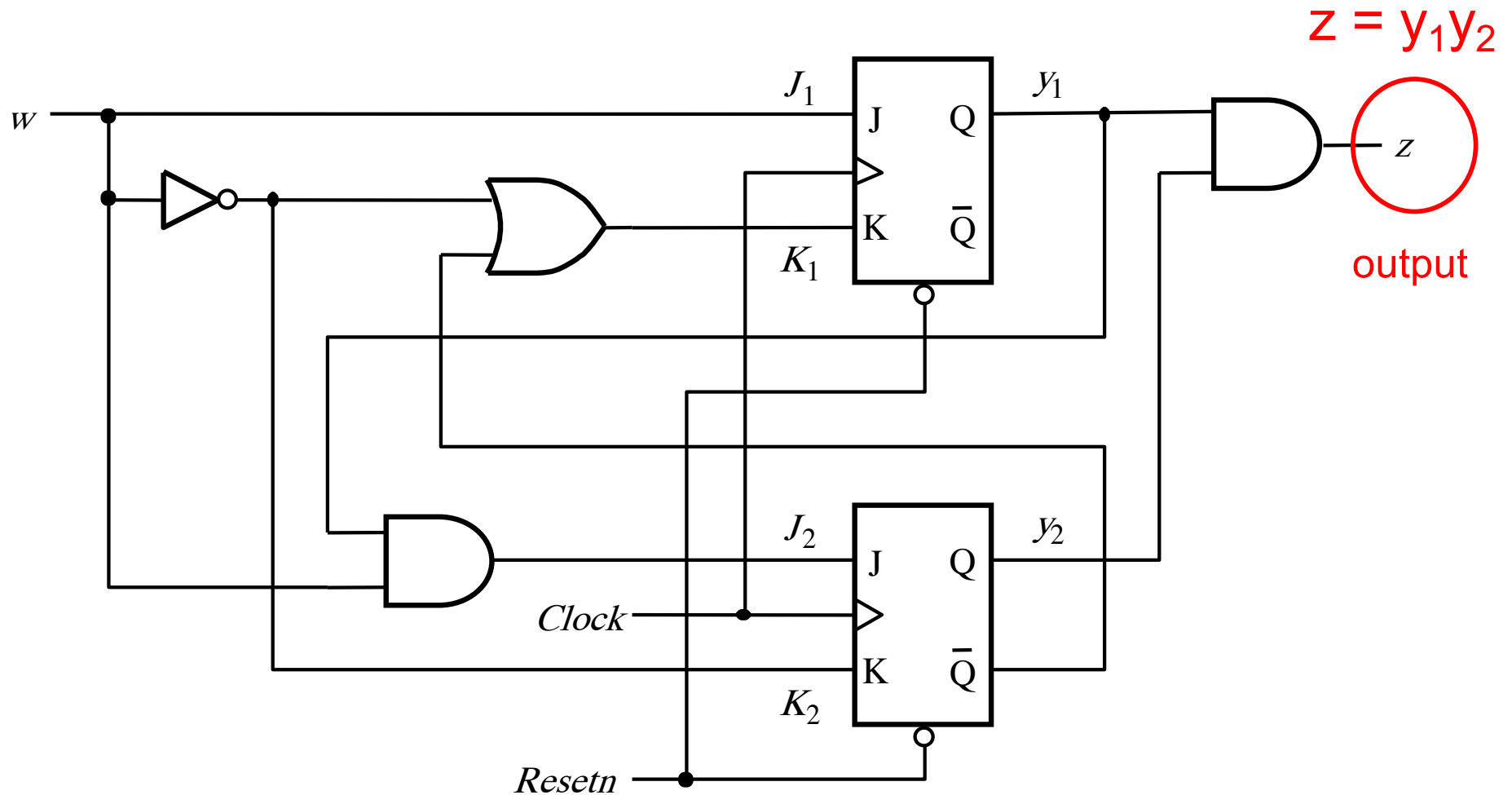
What are their logic expressions?



What is the logic expression of the output?



What is the logic expression of the output?



**This is what we have to work with now
(we don't need the circuit anymore)**

$$J_1 = w$$

$$K_1 = \bar{w} + \bar{y}_2$$

$$J_2 = w y_1$$

$$K_2 = \bar{w}$$

$$z = y_1 y_2$$

Let's derive the excitation table

$$J_1 = w$$

$$K_1 = \bar{w} + \bar{y}_2$$

$$J_2 = w y_1$$

$$K_2 = \bar{w}$$

$$z = y_1 y_2$$

Present state $y_2 y_1$	Flip-flop inputs				Output z
	$w = 0$		$w = 1$		
	$J_2 K_2$	$J_1 K_1$	$J_2 K_2$	$J_1 K_1$	
00					
01					
10					
11					

Let's derive the excitation table

$$J_1 = w$$

$$K_1 = \bar{w} + \bar{y}_2$$

$$J_2 = w y_1$$

$$K_2 = \bar{w}$$

Present state y_2y_1	Flip-flop inputs				Output z
	$w = 0$		$w = 1$		
	J_2K_2	J_1K_1	J_2K_2	J_1K_1	
00					
01					
10					
11					

$$z = y_1y_2$$

Let's derive the excitation table

$$J_1 = w$$

$$K_1 = \bar{w} + \bar{y}_2$$

$$J_2 = w y_1$$

$$K_2 = \bar{w}$$

Present state y_2y_1	Flip-flop inputs				Output z
	$w = 0$		$w = 1$		
	J_2K_2	J_1K_1	J_2K_2	J_1K_1	
00					0
01					0
10					0
11					1

$$z = y_1y_2$$

Let's derive the excitation table

$$J_1 = w$$

$$K_1 = \bar{w} + \bar{y}_2$$

$$J_2 = w y_1$$

$$K_2 = \bar{w}$$

$$z = y_1 y_2$$

Present state $y_2 y_1$	Flip-flop inputs				Output z
	$w = 0$		$w = 1$		
	$J_2 K_2$	$J_1 K_1$	$J_2 K_2$	$J_1 K_1$	
00					0
01					0
10					0
11					1

Let's derive the excitation table

$$J_1 = w$$

$$K_1 = \bar{w} + \bar{y}_2$$

$$J_2 = w y_1$$

$$K_2 = \bar{w}$$

$$z = y_1 y_2$$

Present state $y_2 y_1$	Flip-flop inputs				Output z
	$w = 0$		$w = 1$		
	$J_2 K_2$	$J_1 K_1$	$J_2 K_2$	$J_1 K_1$	
00		01		11	0
01		01		11	0
10		01		10	0
11		01		10	1

Let's derive the excitation table

$$J_1 = w$$

$$K_1 = \bar{w} + \bar{y}_2$$

$$J_2 = w y_1$$

$$K_2 = \bar{w}$$

Present state y_2y_1	Flip-flop inputs				Output z
	$w = 0$		$w = 1$		
	J_2K_2	J_1K_1	J_2K_2	J_1K_1	
00		01		11	0
01		01		11	0
10		01		10	0
11		01		10	1

$$z = y_1y_2$$

The excitation table

$$J_1 = w$$

$$K_1 = \bar{w} + \bar{y}_2$$

$$J_2 = w y_1$$

$$K_2 = \bar{w}$$

$$z = y_1 y_2$$

Present state $y_2 y_1$	Flip-flop inputs				Output z
	$w = 0$		$w = 1$		
	$J_2 K_2$	$J_1 K_1$	$J_2 K_2$	$J_1 K_1$	
00	01	01	00	11	0
01	01	01	10	11	0
10	01	01	00	10	0
11	01	01	10	10	1

We don't need the logic expressions anymore

$$J_1 = w$$

$$K_1 = \bar{w} + \bar{y}_2$$

$$J_2 = w y_1$$

$$K_2 = \bar{w}$$

$$z = y_1 y_2$$

Present state $y_2 y_1$	Flip-flop inputs				Output z
	$w = 0$		$w = 1$		
	$J_2 K_2$	$J_1 K_1$	$J_2 K_2$	$J_1 K_1$	
00	01	01	00	11	0
01	01	01	10	11	0
10	01	01	00	10	0
11	01	01	10	10	1

We don't need the logic expressions anymore

Present state y_2y_1	Flip-flop inputs				Output z
	$w = 0$		$w = 1$		
	J_2K_2	J_1K_1	J_2K_2	J_1K_1	
00	01	01	00	11	0
01	01	01	10	11	0
10	01	01	00	10	0
11	01	01	10	10	1

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	

State table

Present state y_2y_1	Flip-flop inputs				Output z
	w = 0		w = 1		
	J_2K_2	J_1K_1	J_2K_2	J_1K_1	
00	01	01	00	11	0
01	01	01	10	11	0
10	01	01	00	10	0
11	01	01	10	10	1

Excitation table

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A			
B			
C			
D			

Present state y_2y_1	Flip-flop inputs				Output z
	w = 0		w = 1		
	J_2K_2	J_1K_1	J_2K_2	J_1K_1	
00	01	01	00	11	0
01	01	01	10	11	0
10	01	01	00	10	0
11	01	01	10	10	1

State table

Excitation table

This step is easy
(map 2-bit numbers to 4 letters)

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A			0
B			0
C			0
D			1

State table

Present state y_2y_1	Flip-flop inputs				Output z
	w = 0		w = 1		
	J_2K_2	J_1K_1	J_2K_2	J_1K_1	
00	01	01	00	11	0
01	01	01	10	11	0
10	01	01	00	10	0
11	01	01	10	10	1

Excitation table

This step is easy too
(the outputs are the same in both tables)

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A	?		0
B			0
C			0
D			1

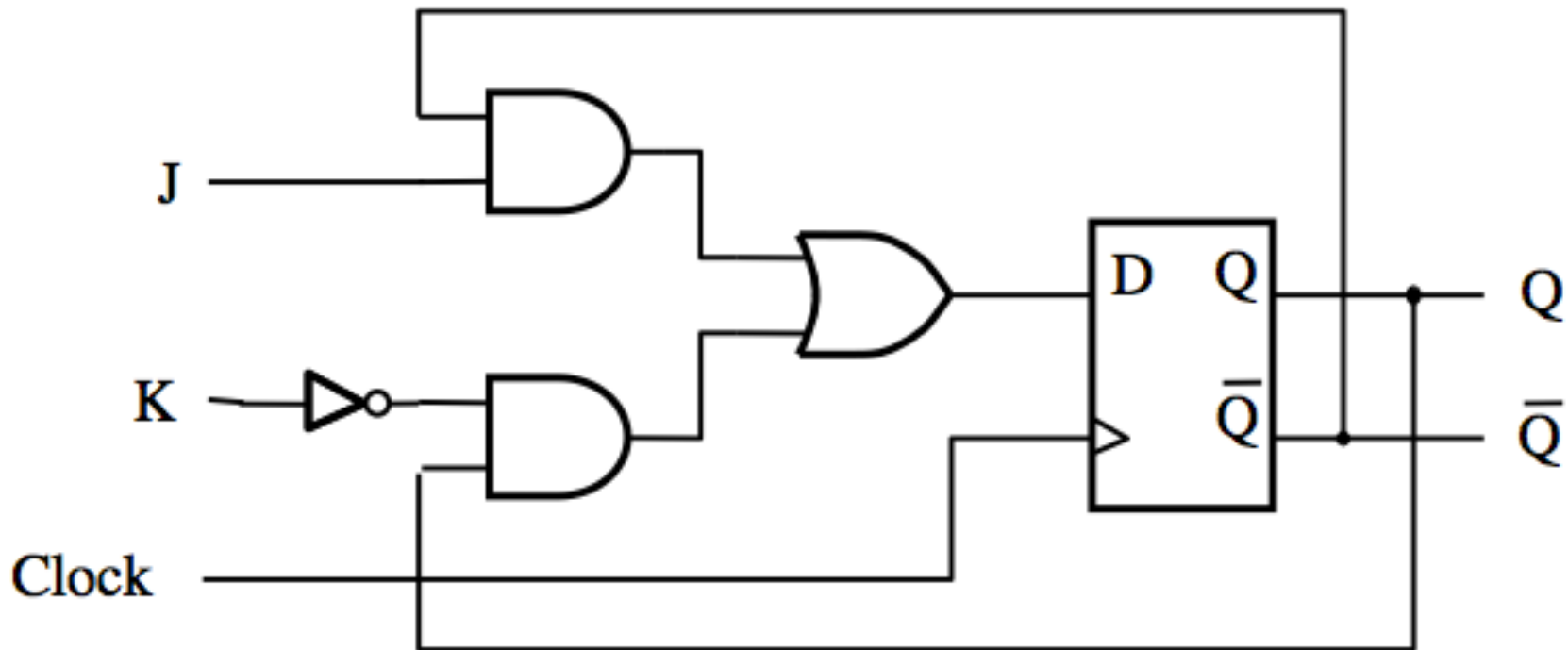
State table

Present state y_2y_1	Flip-flop inputs				Output z
	w = 0		w = 1		
	J_2K_2	J_1K_1	J_2K_2	J_1K_1	
00	01	01	00	11	0
01	01	01	10	11	0
10	01	01	00	10	0
11	01	01	10	10	1

Excitation table

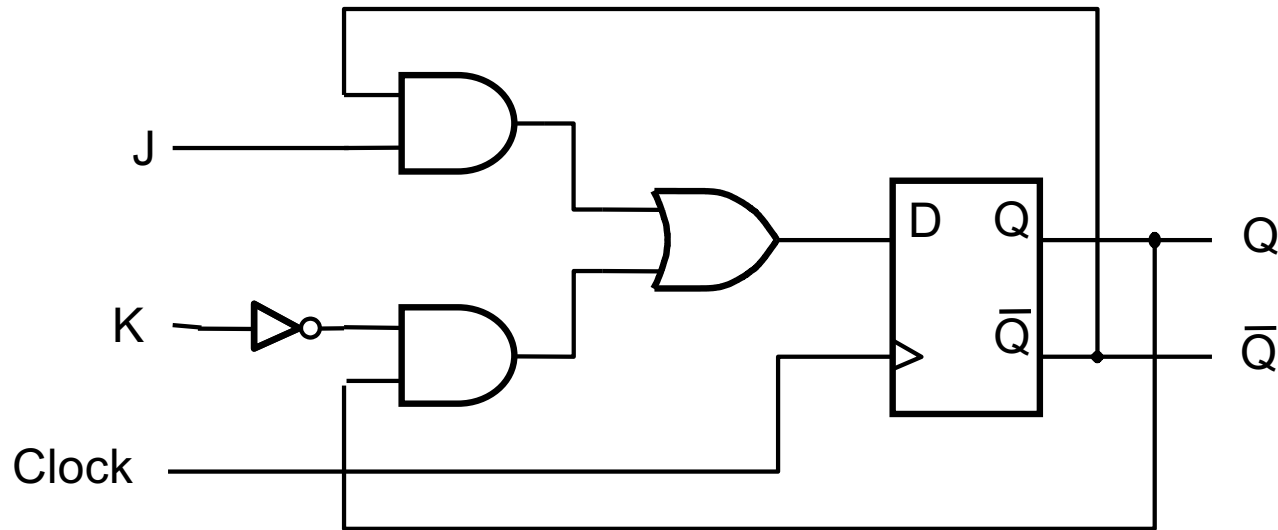
How should we do this?

JK Flip-Flop Refresher



$$D = \overline{J}Q + \overline{K}Q$$

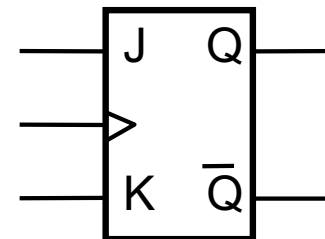
JK Flip-Flop Refresher



(a) Circuit

J	K	$Q(t+1)$
0	0	$Q(t)$
0	1	0
1	0	1
1	1	$\bar{Q}(t)$

(b) Truth table



(c) Graphical symbol

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A	?		0
B			0
C			0
D			1

State table

Present state y_2y_1	Flip-flop inputs				Output z
	w = 0		w = 1		
	J_2K_2	J_1K_1	J_2K_2	J_1K_1	
00	01	01	00	11	0
01	01	01	10	11	0
10	01	01	00	10	0
11	01	01	10	10	1

Excitation table

How should we do this?

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A			0
B			0
C			0
D			1

Present state y_2y_1	Flip-flop inputs				Output z
	w = 0		w = 1		
	J_2K_2	J_1K_1	J_2K_2	J_1K_1	
00	01	01	00	11	0
01	01	01	10	11	0
10	01	01	00	10	0
11	01	01	10	10	1

J K	Q(t+1)
0 0	Q(t)
0 1	0
1 0	1
1 1	$\bar{Q}(t)$

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A			0
B			0
C			0
D			1

Present state y_2y_1	Flip-flop inputs				Output z
	w = 0		w = 1		
	J_2K_2	J_1K_1	J_2K_2	J_1K_1	
00	01	01	00	11	0
01	01	01	10	11	0
10	01	01	00	10	0
11	01	01	10	10	1

J	K	Q(t+1)	J	K	Q(t+1)
0	0	Q(t)	0	0	Q(t)
0	1	0	0	1	0
1	0	1	1	0	1
1	1	$\bar{Q}(t)$	1	1	$\bar{Q}(t)$

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A	A		0
B			0
C			0
D			1

Present state y_2y_1	Flip-flop inputs				Output z
	w = 0		w = 1		
	J_2K_2	J_1K_1	J_2K_2	J_1K_1	
00	01	01	00	11	0
01	01	01	10	11	0
10	01	01	00	10	0
11	01	01	10	10	1

Note that A = 00

J	K	Q(t+1)	J	K	Q(t+1)
0	0	Q(t)	0	0	Q(t)
0	1	0	0	1	0
1	0	1	1	0	1
1	1	$\bar{Q}(t)$	1	1	$\bar{Q}(t)$

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A	A	?	0
B			0
C			0
D			1

Present state y_2y_1	Flip-flop inputs				Output z
	w = 0		w = 1		
	J_2K_2	J_1K_1	J_2K_2	J_1K_1	
00	01	01	00	11	0
01	01	01	10	11	0
10	01	01	00	10	0
11	01	01	10	10	1

J	K	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	$\bar{Q}(t)$

J	K	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	$\bar{Q}(t)$

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A	A		0
B	A		0
C	A		0
D	A		1

Present state y_2y_1	Flip-flop inputs				Output z
	w = 0		w = 1		
	J_2K_2	J_1K_1	J_2K_2	J_1K_1	
00	01	01	00	11	0
01	01	01	10	11	0
10	01	01	00	10	0
11	01	01	10	10	1

J	K	$Q(t+1)$	J	K	$Q(t+1)$
0	0	$Q(t)$	0	0	$Q(t)$
0	1	0	0	1	0
1	0	1	1	0	1
1	1	$\bar{Q}(t)$	1	1	$\bar{Q}(t)$

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A	A		0
B	A		0
C	A		0
D	A		1

Present state y_2y_1	Flip-flop inputs				Output z
	w = 0		w = 1		
	J_2K_2	J_1K_1	J_2K_2	J_1K_1	
00	01	01	00	11	0
01	01	01	10	11	0
10	01	01	00	10	0
11	01	01	10	10	1

J	K	$Q(t+1)$	J	K	$Q(t+1)$
0	0	$Q(t)$	0	0	$Q(t)$
0	1	0	0	1	0
1	0	1	1	0	1
1	1	$\bar{Q}(t)$	1	1	$\bar{Q}(t)$

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A	A		0
B	A		0
C	A		0
D	A		1

Present state y_2y_1	Flip-flop inputs				Output z
	w = 0		w = 1		
	J_2K_2	J_1K_1	J_2K_2	J_1K_1	
00	01	01	00	11	0
01	01	01	10	11	0
10	01	01	00	10	0
11	01	01	10	10	1

J	K	$Q(t+1)$	J	K	$Q(t+1)$
0	0	$Q(t)$	0	0	$Q(t)$
0	1	0	0	1	0
1	0	1	1	0	1
1	1	$\bar{Q}(t)$	1	1	$\bar{Q}(t)$

01

10

11

10

1

11

$\bar{Q}(t)$

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A	A		0
B	A		0
C	A		0
D	A		1

Present state y_2y_1	Flip-flop inputs				Output z
	w = 0		w = 1		
	J_2K_2	J_1K_1	J_2K_2	J_1K_1	
00	01	01	00	11	0
01	01	01	10	11	0
10	01	01	00	10	0
11	01	01	10	10	1

J	K	$Q(t+1)$	J	K	$Q(t+1)$
0	0	$Q(t)$	0	0	$Q(t)$
0	1	0	0	1	0
1	0	1	1	0	1
1	1	$\bar{Q}(t)$	1	1	$\bar{Q}(t)$ = $\overline{1} = 0$

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A	A		0
B		C	0
C			0
D			1

Present state y_2y_1	Flip-flop inputs				Output z
	w = 0		w = 1		
	J_2K_2	J_1K_1	J_2K_2	J_1K_1	
00	01	01	00	11	0
01	01	01	10	11	0
10	01	01	00	10	0
11	01	01	10	10	1

Note that C = 10

J	K	$Q(t+1)$	J	K	$Q(t+1)$
0	0	$Q(t)$	0	0	$Q(t)$
0	1	0	0	1	0
1	0	1	1	0	1
1	1	$\bar{Q}(t)$	1	1	$\bar{Q}(t)$ = 0

The two tables for the initial circuit

Present state	Next state		Output z
	w = 0	w = 1	
A	A	B	0
B	A	C	0
C	A	D	0
D	A	D	1

State table

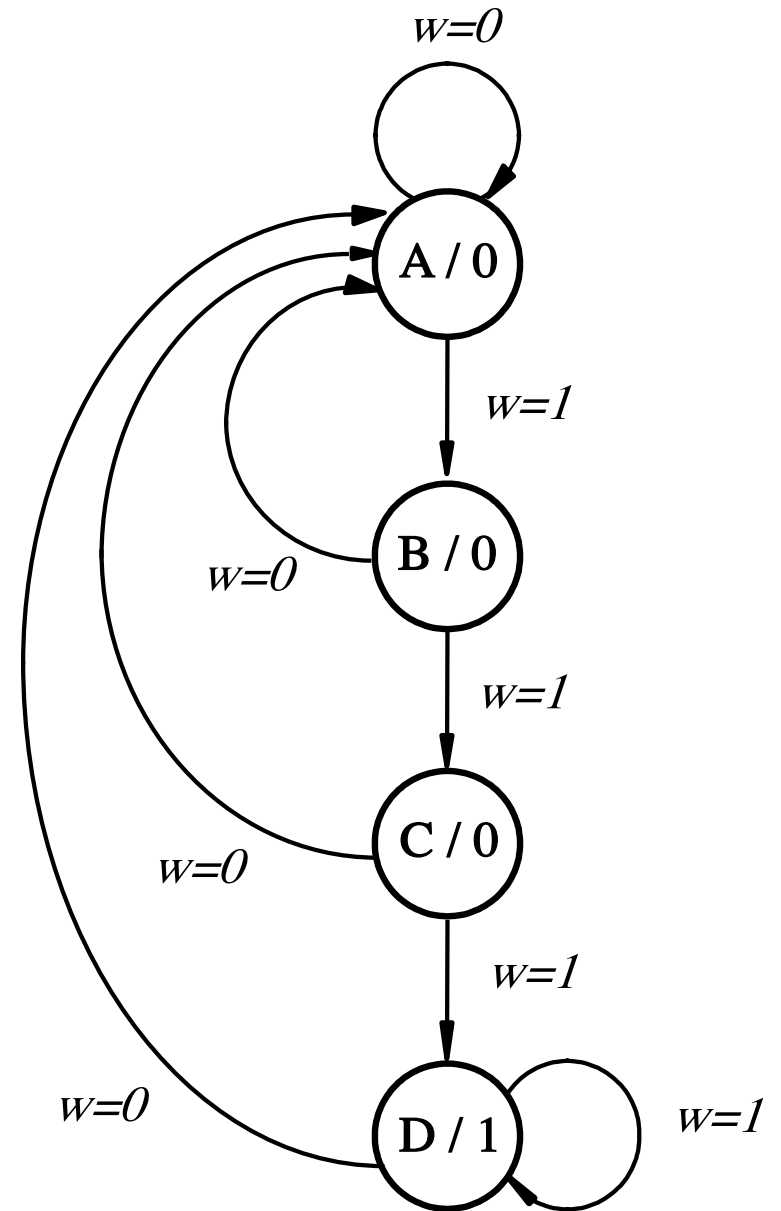
Present state y_2y_1	Flip-flop inputs				Output z
	w = 0		w = 1		
	J_2K_2	J_1K_1	J_2K_2	J_1K_1	
00	01	01	00	11	0
01	01	01	10	11	0
10	01	01	00	10	0
11	01	01	10	10	1

Excitation table

The state diagram

Present state	Next state		Output z
	w = 0	w = 1	
A	A	B	0
B	A	C	0
C	A	D	0
D	A	D	1

State table



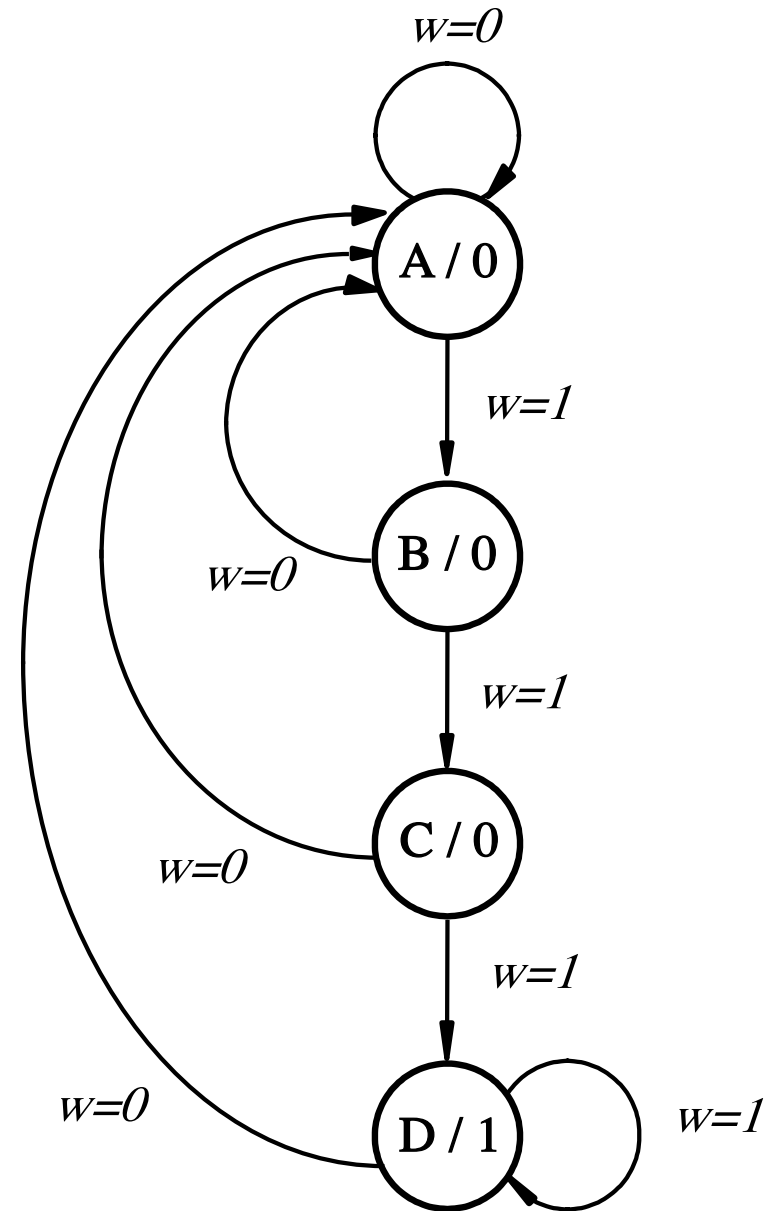
State diagram

The state diagram

Thus, this FSM is identical to the one in the previous example, even though the circuit uses JK flip-flops.

Present state	Next state		Output z
	w = 0	w = 1	
A	A	B	0
B	A	C	0
C	A	D	0
D	A	D	1

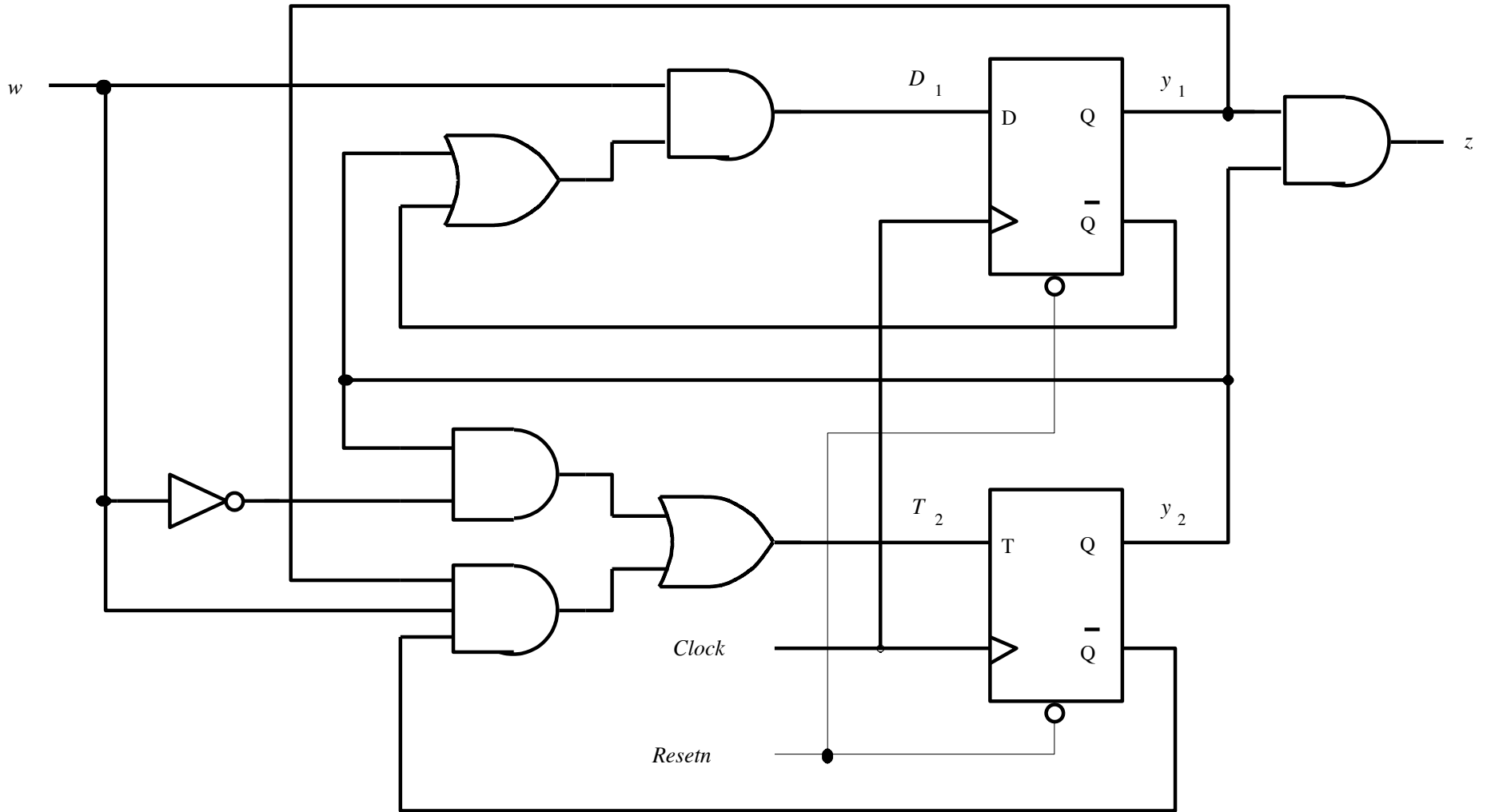
State table



State diagram

Yet Another Example (with mixed flip-flops)

What does this circuit do?

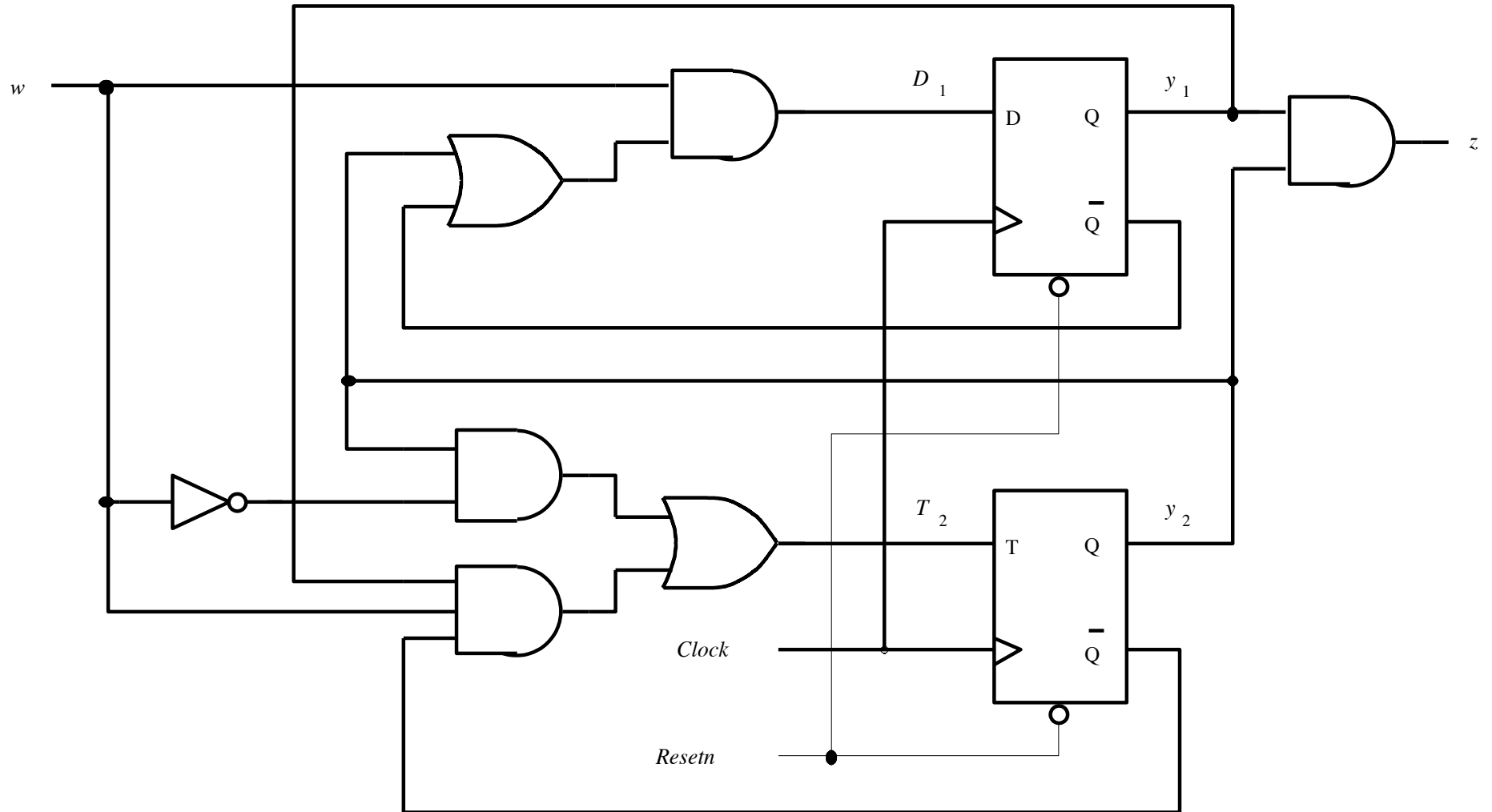


[Figure 6.79 from the textbook]

Approach

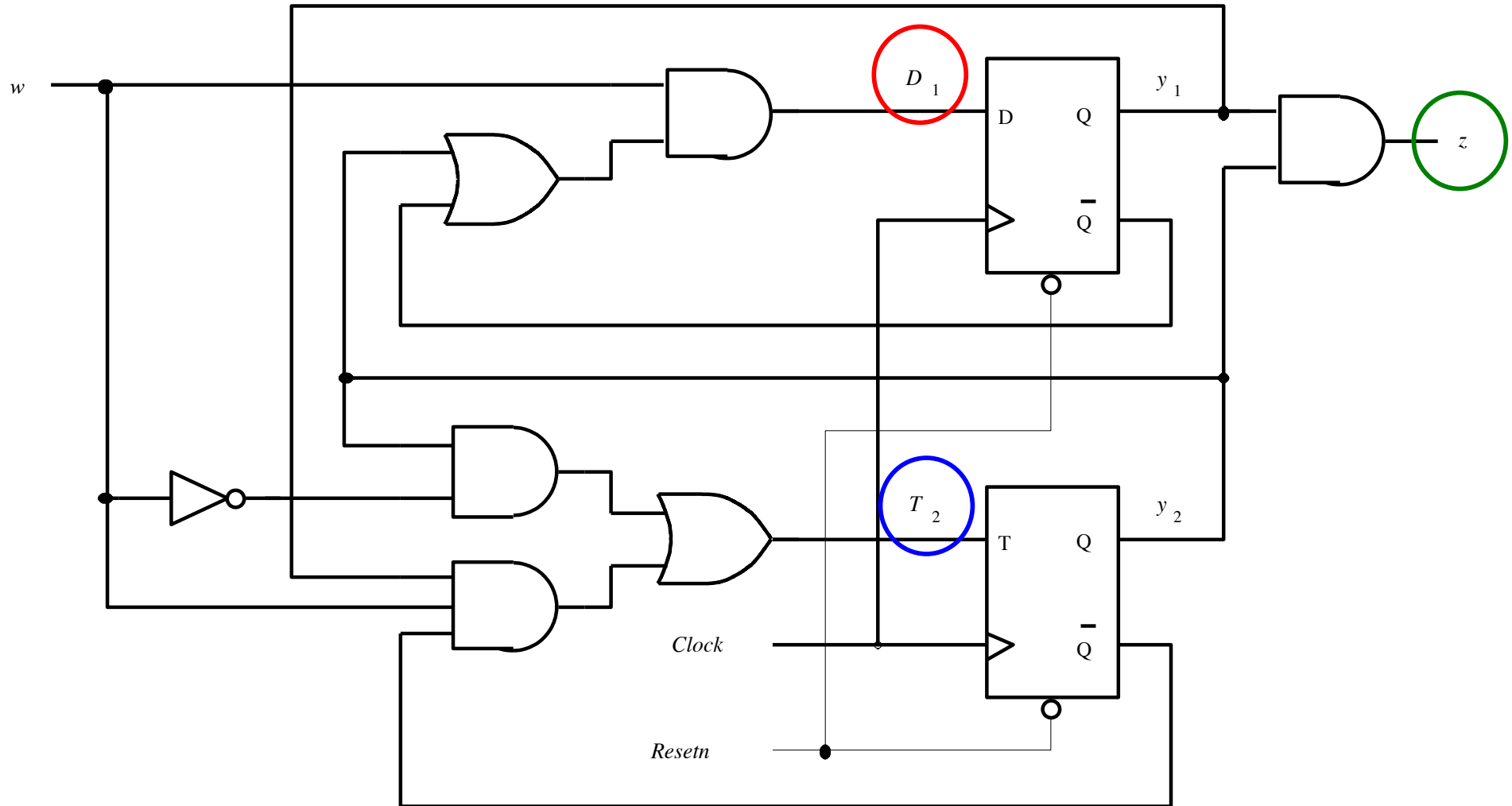
- **Find the flip-flops**
- **Outputs of the flip-flops = present state variables**
- **Inputs of the flip-flops determine the next state variables**
- **Determine the logical expressions for the outputs**
- **Given this info it is easy to do the state-assigned table**
- **Next do the state table**
- **Finally, draw the state diagram.**

What are the logic expressions?



[Figure 6.79 from the textbook]

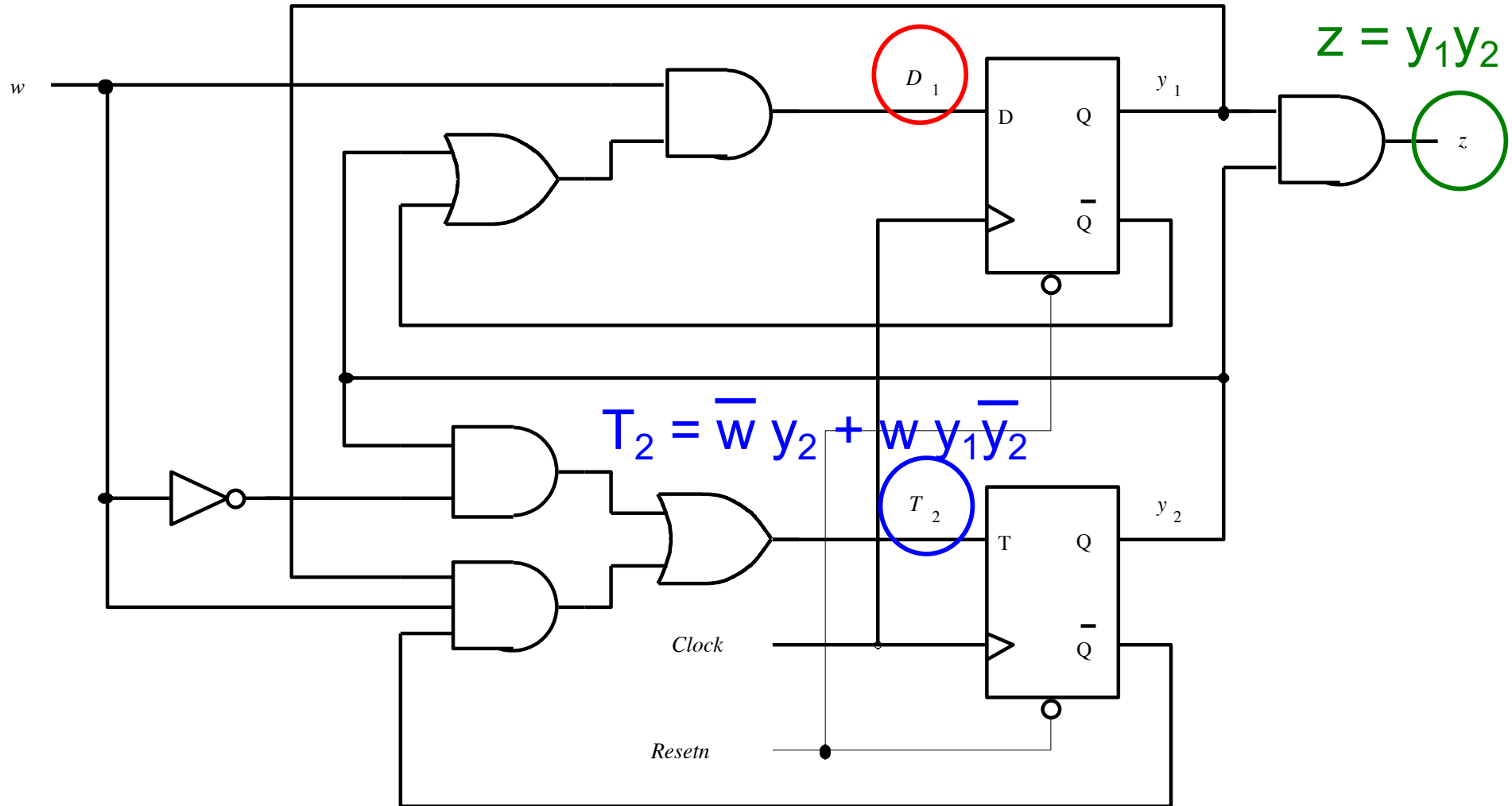
What are the logic expressions?



What are the logic expressions?

$$D_1 = w (\bar{y}_1 + y_2)$$

$$z = y_1 y_2$$



The Excitation Table

$$D_1 = w (\bar{y}_1 + y_2)$$

$$T_2 = \bar{w} y_2 + w y_1 \bar{y}_2$$

$$z = y_1 y_2$$

Present state $y_2 y_1$	Flip-flop inputs		Output z
	$w = 0$	$w = 1$	
	$T_2 D_1$	$T_2 D_1$	
00	00	01	0
01	00	10	0
10	10	01	0
11	10	01	1

Excitation table

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	

Present state y_2y_1	Flip-flop inputs		Output z
	w = 0	w = 1	
	T_2D_1	T_2D_1	
00	00	01	0
01	00	10	0
10	10	01	0
11	10	01	1

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A			
B			
C			
D			

Present state y_2y_1	Flip-flop inputs		Output z
	w = 0	w = 1	
	T_2D_1	T_2D_1	
00	00	01	0
01	00	10	0
10	10	01	0
11	10	01	1

This step is easy
(map 2-bit numbers to 4 letters)

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A			0
B			0
C			0
D			1

Present state y_2y_1	Flip-flop inputs		Output z
	w = 0	w = 1	
	T_2D_1	T_2D_1	
00	00	01	0
01	00	10	0
10	10	01	0
11	10	01	1

This step is easy too
(the outputs are the same in both tables)

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A	?		0
B			0
C			0
D			1

Present state y_2y_1	Flip-flop inputs		Output z
	w = 0	w = 1	
	T_2D_1	T_2D_1	
00	00	01	0
01	00	10	0
10	10	01	0
11	10	01	1

What should we do here?

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A	?		0
B			0
C			0
D			1

Present state y_2y_1	Flip-flop inputs		Output z
	w = 0	w = 1	
	T_2D_1	T_2D_1	
00	00	01	0
01	00	10	0
10	10	01	0
11	10	01	1

What should we do here?

T	$Q(t+1)$	D	$Q(t+1)$
0	$Q(t)$	0	0
1	$\overline{Q(t)}$	1	1

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A			0
B			0
C			0
D			1

Present state y_2y_1	Flip-flop inputs		Output z
	w = 0	w = 1	
	T_2D_1	T_2D_1	
00	00	01	0
01	00	10	0
10	10	01	0
11	10	01	1

T	$Q(t+1)$	D	$Q(t+1)$
0	$Q(t)$	0	0
1	$\overline{Q(t)}$	1	1

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A			0
B			0
C			0
D			1

Present state y_2y_1	Flip-flop inputs		Output z
	w = 0	w = 1	
	T_2D_1	T_2D_1	
00	00	01	0
01	00	10	0
10	10	01	0
11	10	01	1

T	$Q(t+1)$	D	$Q(t+1)$
0	$Q(t)$	0	0
1	$\overline{Q(t)}$	1	1

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A			0
B			0
C			0
D			1

Present state y_2y_1	Flip-flop inputs		Output z
	w = 0	w = 1	
	T_2D_1	T_2D_1	
00	00	01	0
01	00	10	0
10	10	01	0
11	10	01	1

T	$Q(t+1)$	D	$Q(t+1)$
0	$Q(t)$	0	0
1	$\overline{Q(t)}$	1	1

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A			0
B			0
C			0
D			1

Present state y_2y_1	Flip-flop inputs		Output z
	w = 0	w = 1	
	T_2D_1	T_2D_1	
00	00	01	0
01	00	10	0
10	10	01	0
11	10	01	1

T	$Q(t+1)$	D	$Q(t+1)$
0	0	0	0
1	$\overline{Q(t)}$	1	1

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A			0
B			0
C			0
D			1

Present state y_2y_1	Flip-flop inputs		Output z
	w = 0	w = 1	
	T_2D_1	T_2D_1	
00	00	01	0
01	00	10	0
10	10	01	0
11	10	01	1

T	$Q(t+1)$	D	$Q(t+1)$
0	0	0	0
1	$\overline{Q(t)}$	1	1

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A			0
B			0
C			0
D			1

Present state y_2y_1	Flip-flop inputs		Output z
	w = 0	w = 1	
	T_2D_1	T_2D_1	
00	00	01	0
01	00	10	0
10	10	01	0
11	10	01	1

T	$Q(t+1)$	D	$Q(t+1)$
0	0	0	0
1	$\overline{Q(t)}$	1	1

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A	A		0
B			0
C			0
D			1

Present state y_2y_1	Flip-flop inputs		Output z
	w = 0	w = 1	
	T_2D_1	T_2D_1	
00	00	01	0
01	00	10	0
10	10	01	0
11	10	01	1

Note that A = 00

T	$Q(t+1)$	D	$Q(t+1)$
0	0	0	0
1	$\overline{Q(t)}$	1	1

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A	A		0
B			0
C		?	0
D			1

Present state y_2y_1	Flip-flop inputs		Output z
	w = 0	w = 1	
	T_2D_1	T_2D_1	
00	00	01	0
01	00	10	0
10	10	01	0
11	10	01	1

What should we do here?

T	$Q(t+1)$	D	$Q(t+1)$
0	$Q(t)$	0	0
1	$\overline{Q(t)}$	1	1

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A	A		0
B			0
C			0
D			1

Present state y_2y_1	Flip-flop inputs		Output z
	w = 0	w = 1	
	T_2D_1	T_2D_1	
00	00	01	0
01	00	10	0
10	10	01	0
11	10	01	1

T	$Q(t+1)$	D	$Q(t+1)$
0	$Q(t)$	0	0
1	$\overline{Q(t)}$	1	1

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A	A		0
B			0
C			0
D			1

Present state y_2y_1	Flip-flop inputs		Output z
	w = 0	w = 1	
	T_2D_1	T_2D_1	
00	00	01	0
01	00	10	0
10	10	01	0
11	10	01	1

T	$Q(t+1)$	D	$Q(t+1)$
0	$Q(t)$	0	0
1	$\overline{Q(t)}$	1	1

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A	A		0
B	A		0
C	A		0
D	A		1

Present state y_2y_1	Flip-flop inputs		Output z
	w = 0	w = 1	
	T_2D_1	T_2D_1	
00	00	01	0
01	00	10	0
10	10	01	0
11	10	01	1

T	$Q(t+1)$	D	$Q(t+1)$
0	$Q(t)$	0	0
1	$\overline{Q(t)}$	1	1

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A	A		0
B	A		0
C	A		0
D	A		1

Present state y_2y_1	Flip-flop inputs		Output z
	w = 0	w = 1	
	T_2D_1	T_2D_1	
00	00	01	0
01	00	10	0
10	10	01	0
11	10	01	1

T	$Q(t+1)$	D	$Q(t+1)$
0	1	0	0
1	$\overline{Q(t)}$	1	1

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A	A		0
B			0
C			0
D			1

Present state y_2y_1	Flip-flop inputs		Output z
	w = 0	w = 1	
	T_2D_1	T_2D_1	
00	00	01	0
01	00	10	0
10	10	01	0
11	10	01	1

T	$Q(t+1)$	D	$Q(t+1)$
0	1	0	0
1	$\overline{Q(t)}$	1	1

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A	A		0
B	A		0
C	A		0
D	A		1

Present state y_2y_1	Flip-flop inputs		Output z
	w = 0	w = 1	
	T_2D_1	T_2D_1	
00	00	01	0
01	00	10	0
10	10	01	0
11	10	01	1

T	$Q(t+1)$	D	$Q(t+1)$
0	1	0	0
1	$\overline{Q}(t)$	1	1

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A	A		0
B			0
C		D	0
D			1

Present state y_2y_1	Flip-flop inputs		Output z
	w = 0	w = 1	
	T_2D_1	T_2D_1	
00	00	01	0
01	00	10	0
10	10	01	0
11	10	01	1

Note that D = 11

T	$Q(t+1)$	D	$Q(t+1)$
0	1	0	0
1	$\bar{Q}(t)$	1	1

Let's derive the state table

Present state	Next state		Output z
	w = 0	w = 1	
A	A	B	0
B	A	C	0
C	A	D	0
D	A	D	1

Present state y_2y_1	Flip-flop inputs		Output z
	w = 0	w = 1	
	T_2D_1	T_2D_1	
00	00	01	0
01	00	10	0
10	10	01	0
11	10	01	1

T	$Q(t+1)$	D	$Q(t+1)$
0	$Q(t)$	0	0
1	$\overline{Q(t)}$	1	1

The two tables for the initial circuit

Present state	Next state		Output z
	w = 0	w = 1	
A	A	B	0
B	A	C	0
C	A	D	0
D	A	D	1

State table

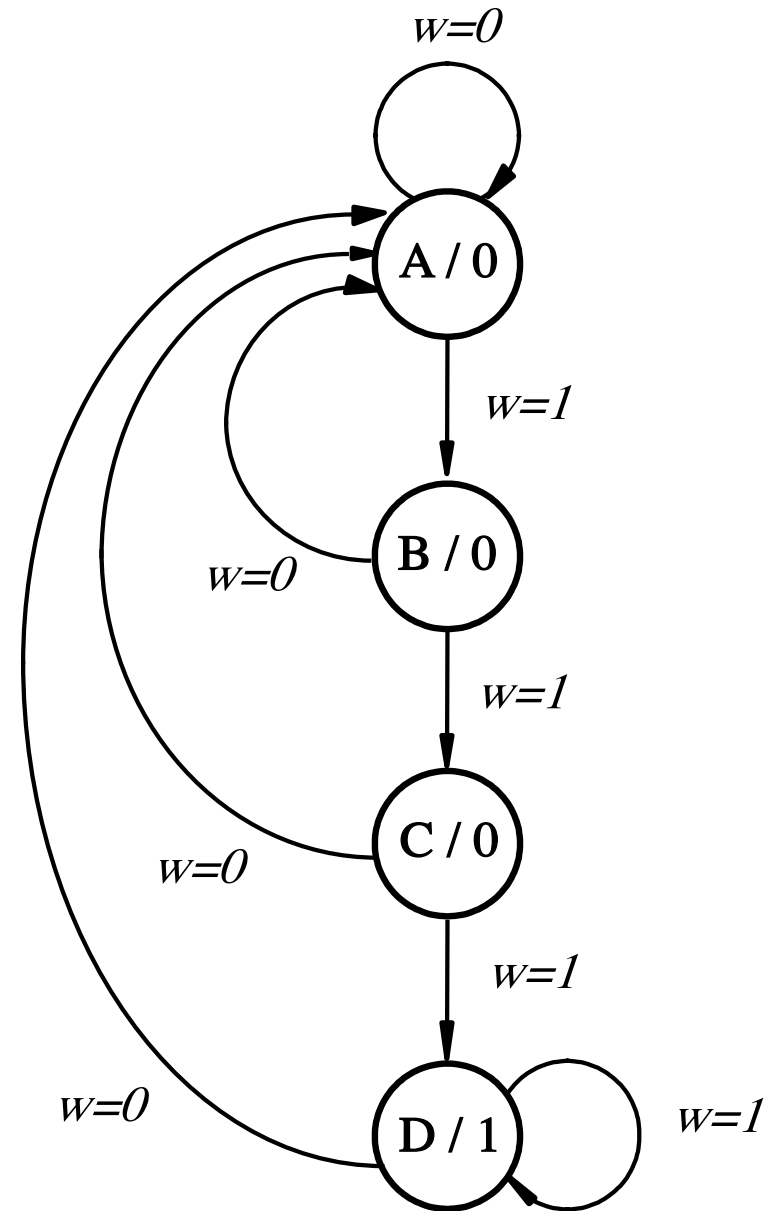
Present state y_2y_1	Flip-flop inputs		Output z
	w = 0	w = 1	
	T_2D_1	T_2D_1	
00	00	01	0
01	00	10	0
10	10	01	0
11	10	01	1

Excitation table

The state diagram

Present state	Next state		Output z
	w = 0	w = 1	
A	A	B	0
B	A	C	0
C	A	D	0
D	A	D	1

State table



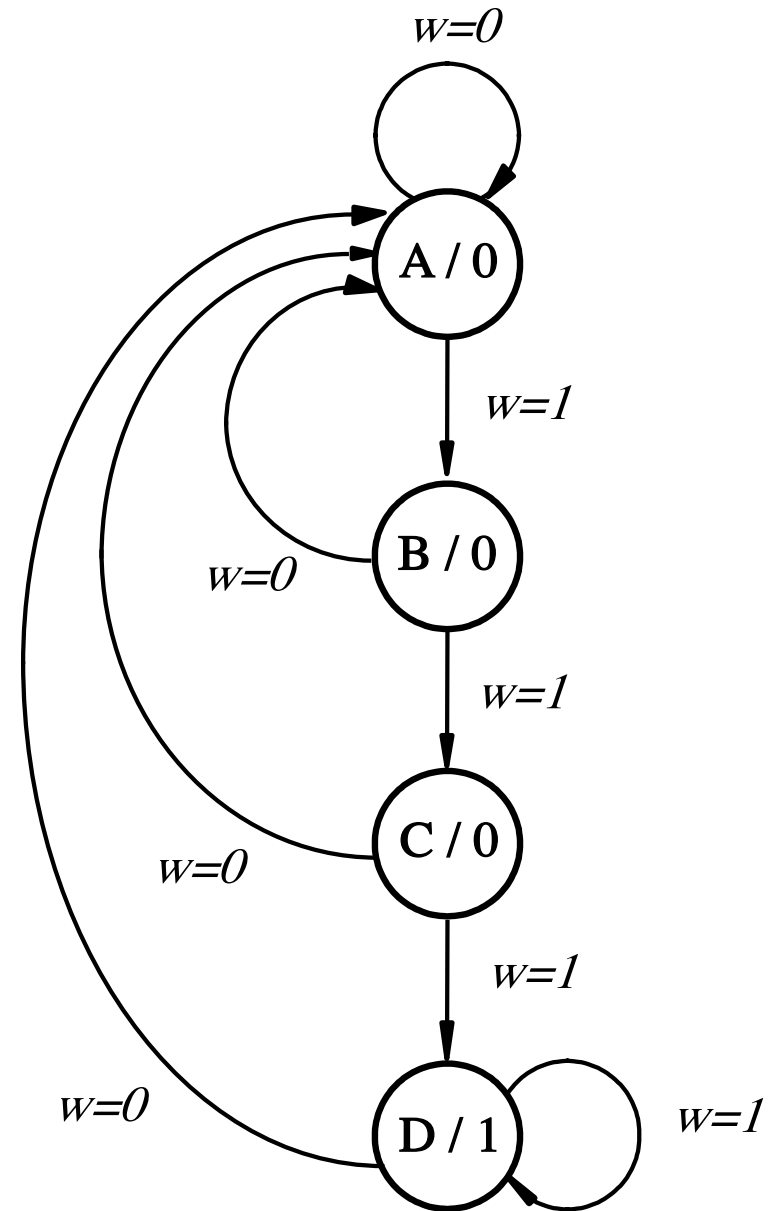
State diagram

The state diagram

Thus, this FSM is identical to the ones in the previous examples, even though the circuit uses JK flip-flops.

Present state	Next state		Output Z
	w = 0	w = 1	
A	A	B	0
B	A	C	0
C	A	D	0
D	A	D	1

State table



State diagram

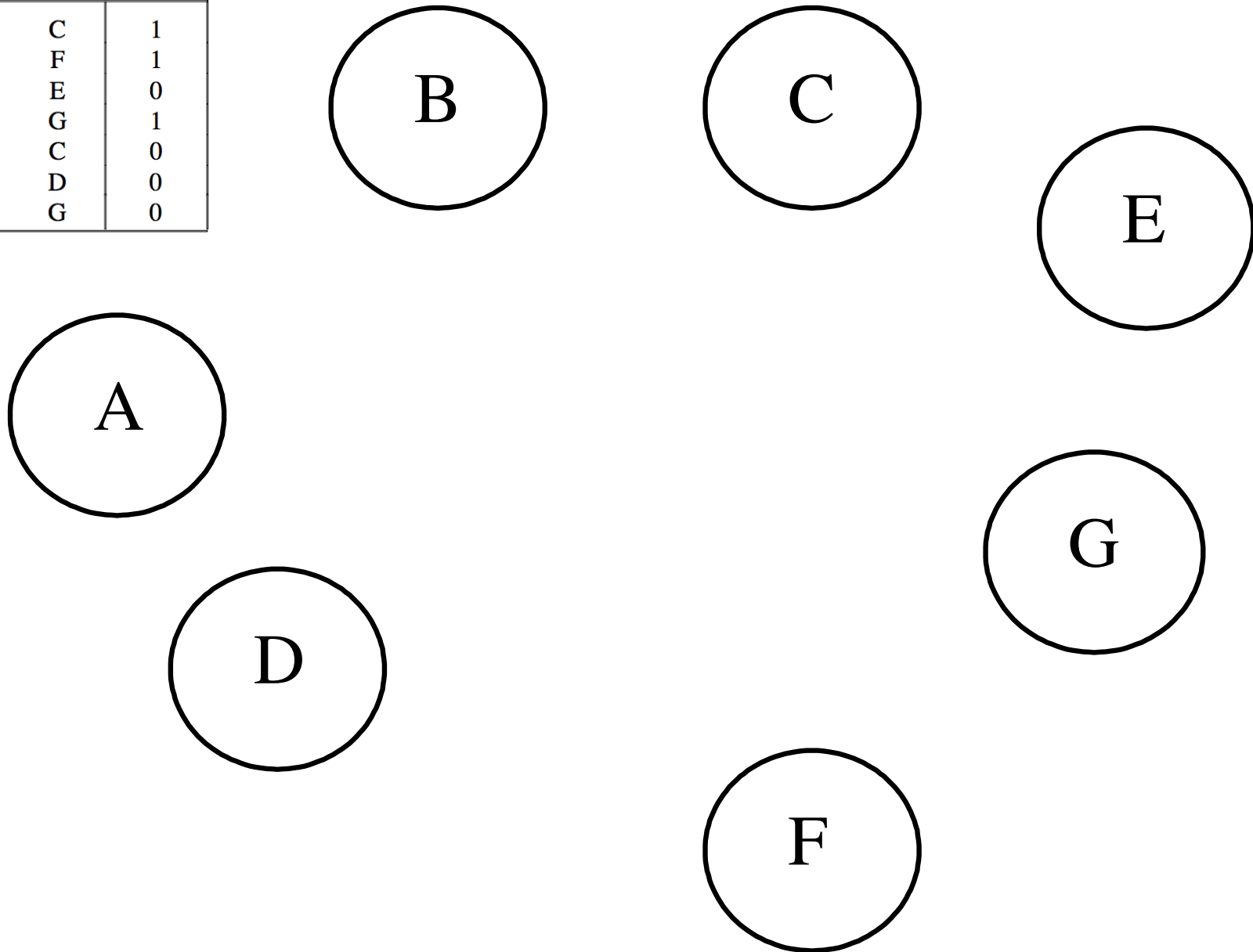
State Minimization

State Table for This Example

Present state	Next state		Output z
	$w = 0$	$w = 1$	
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

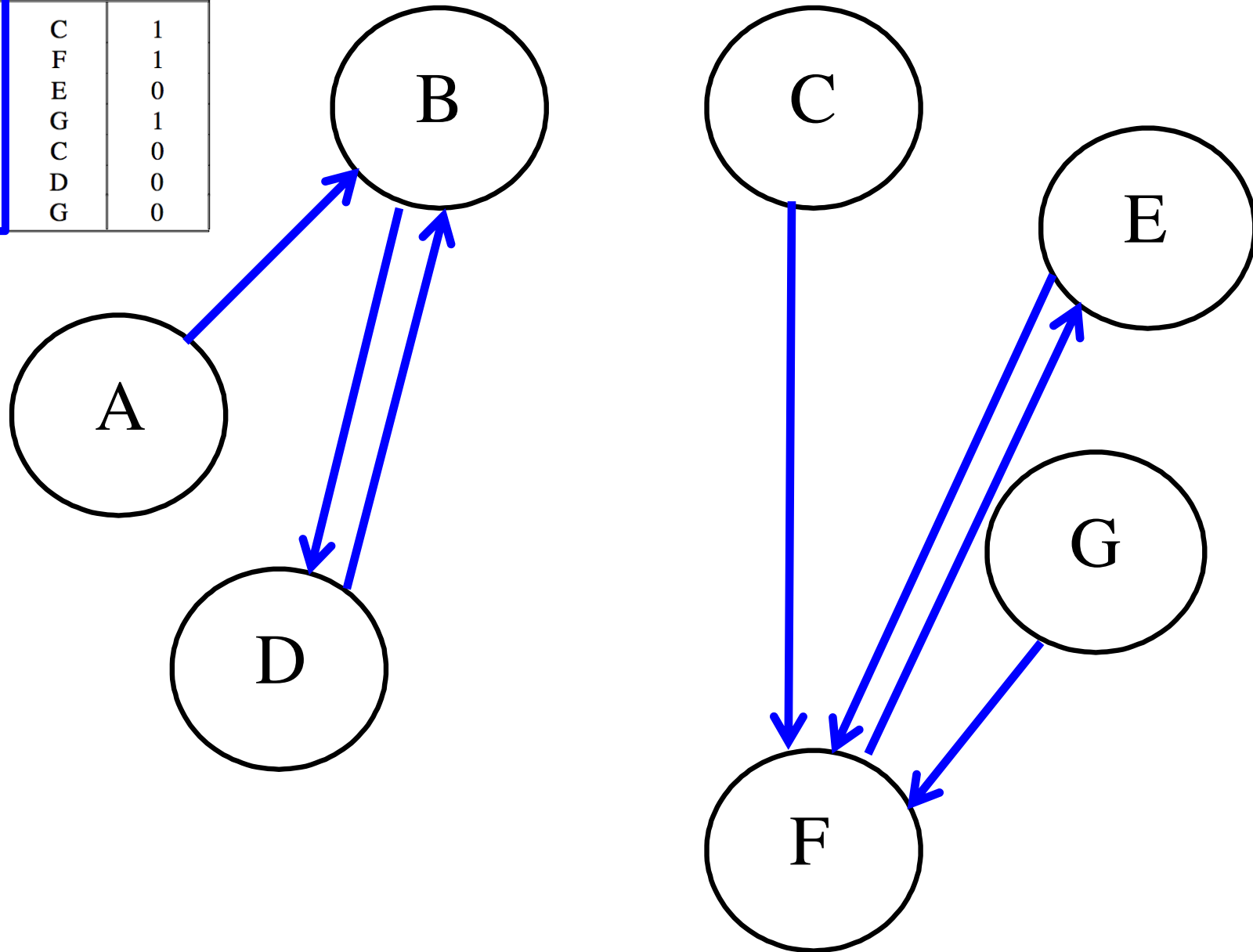
State Diagram (just the states)

Present state	Next state		Output z
	$w = 0$	$w = 1$	
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0



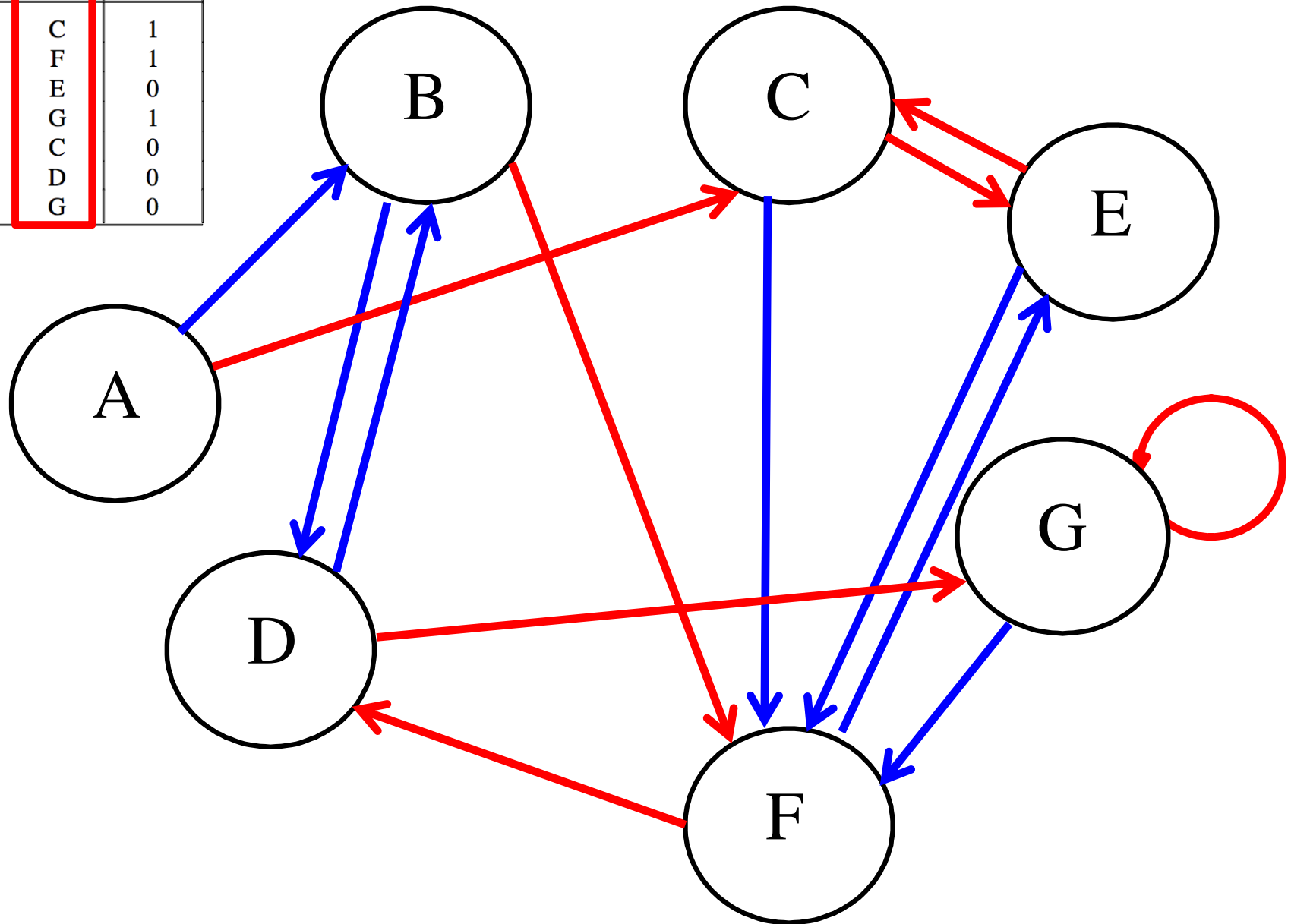
State Diagram (transitions when $w=0$)

Present state	Next state		Output z
	$w = 0$	$w = 1$	
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0



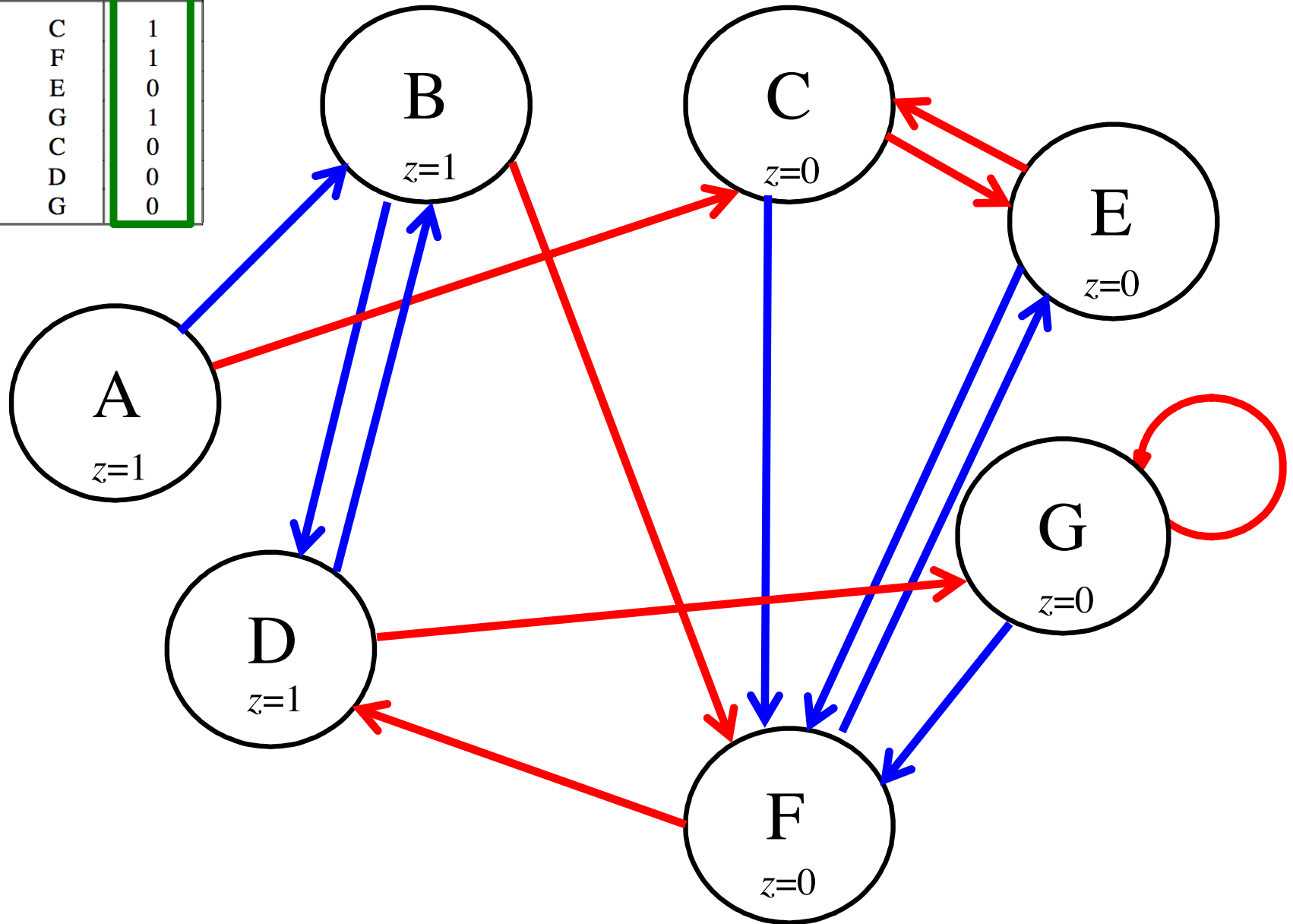
State Diagram (transitions when $w=1$)

Present state	Next state		Output z
	$w = 0$	$w = 1$	
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0



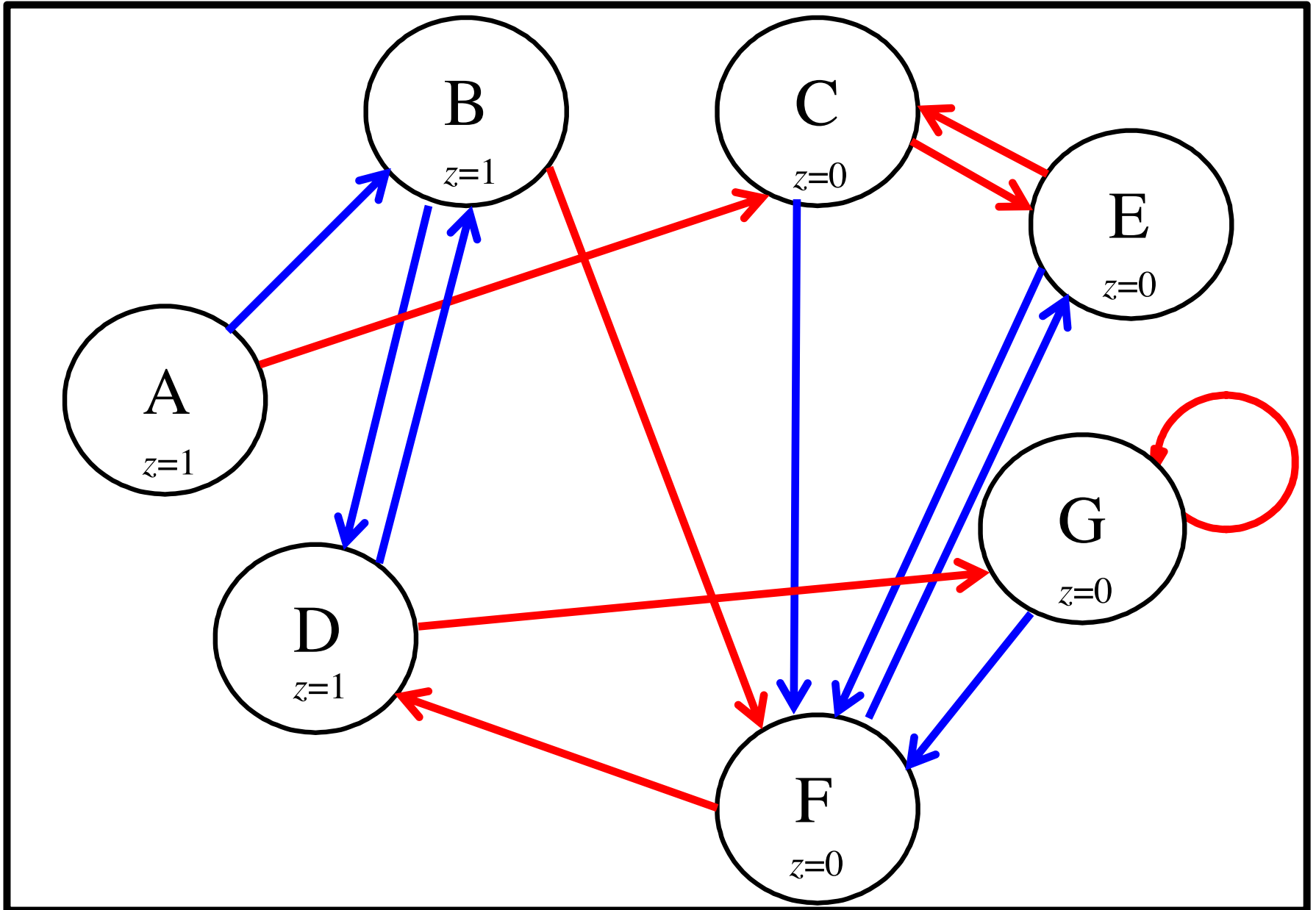
Outputs

Present state	Next state		Output
	$w = 0$	$w = 1$	z
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0



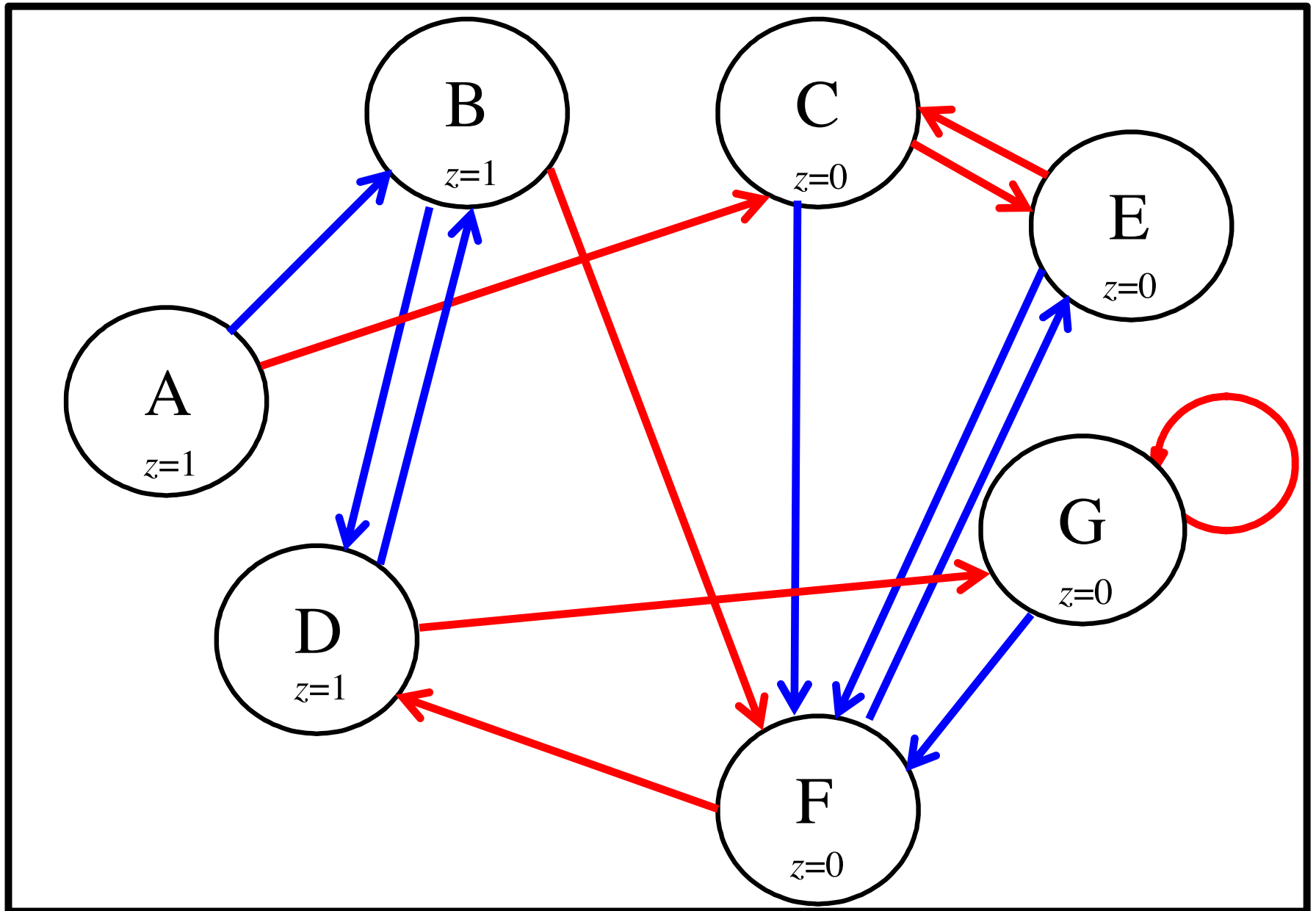
Partition #1

(All states in the same partition)



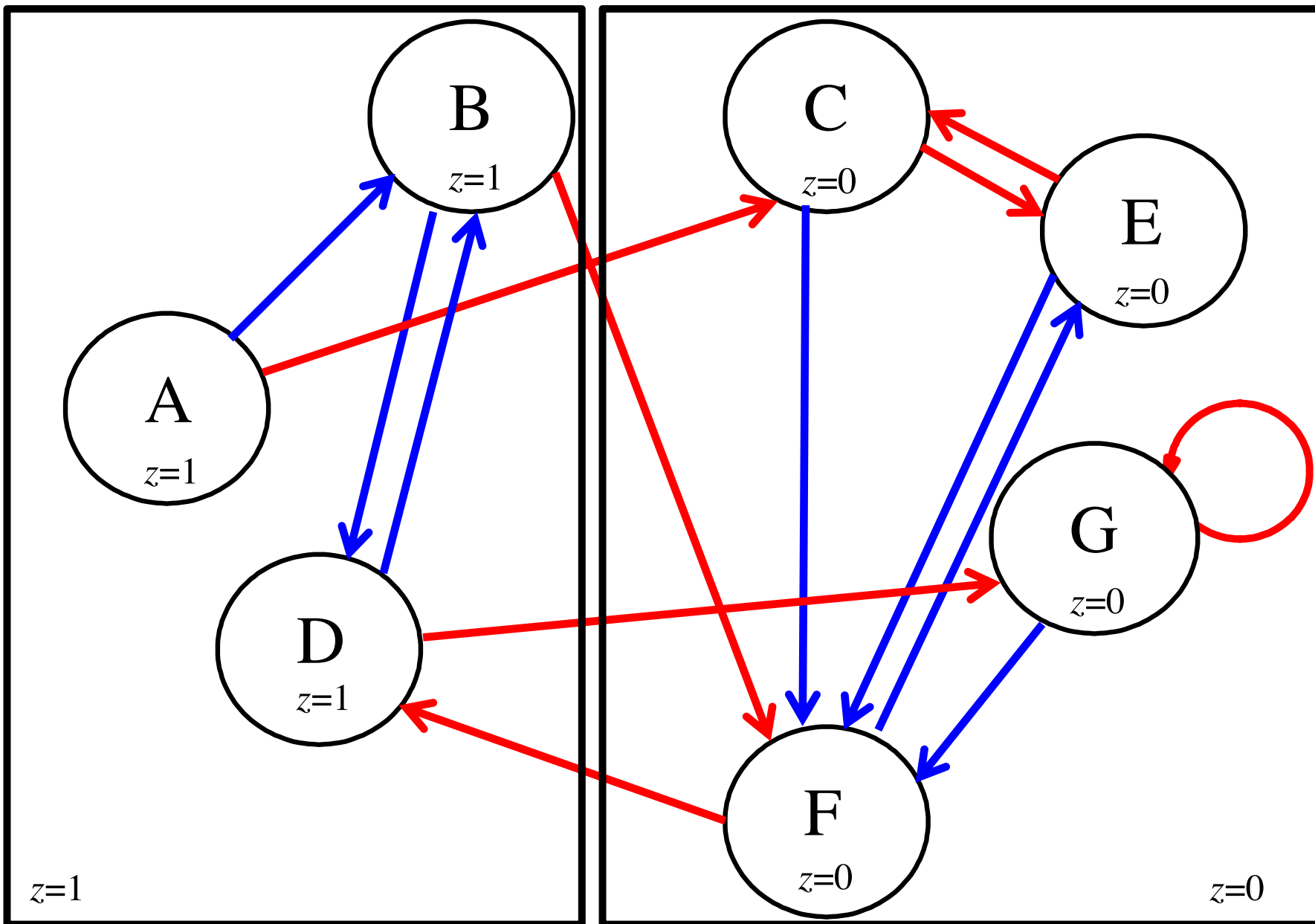
Partition #1

(ABCDEFGG)



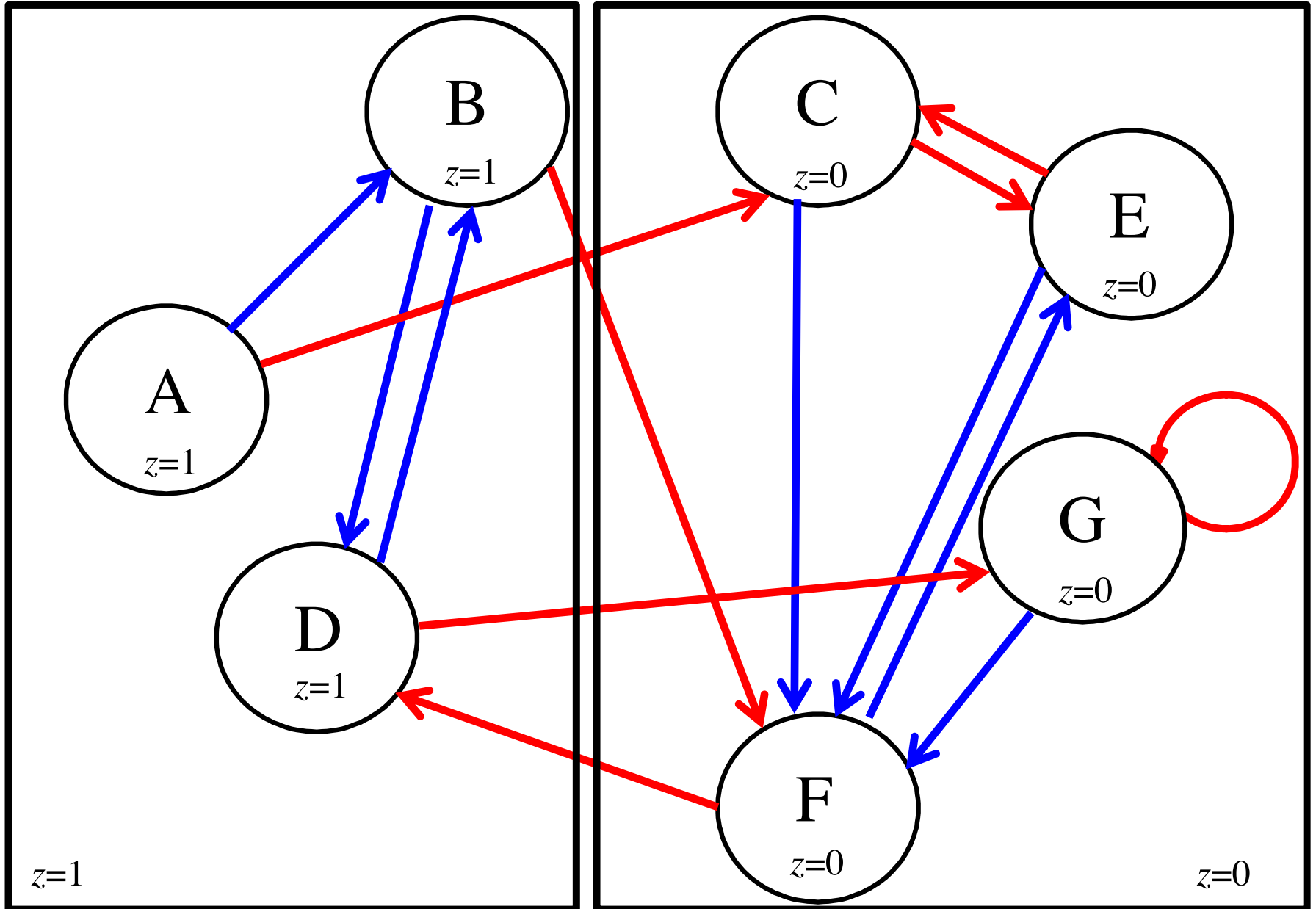
Partition #2

(based on outputs)



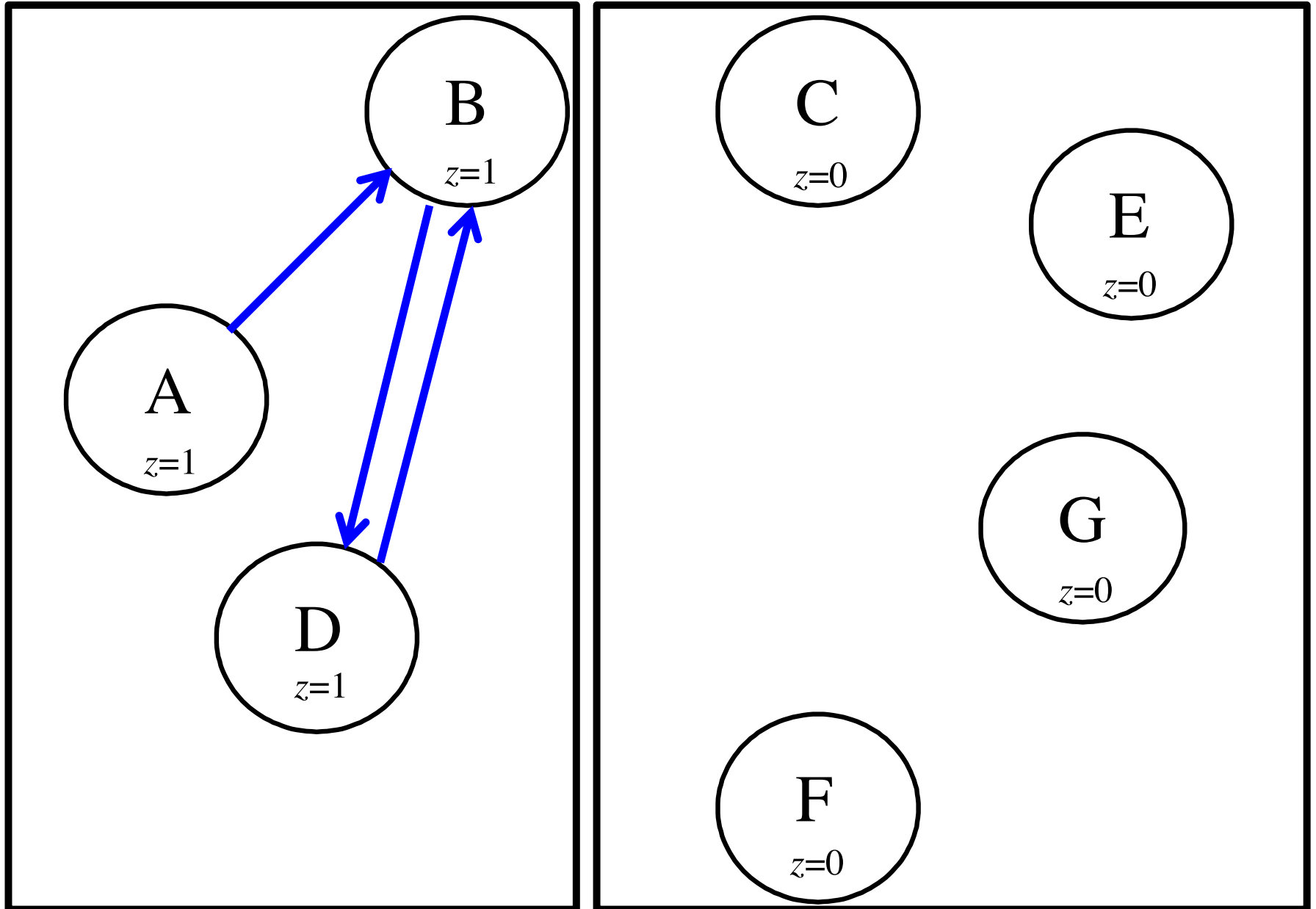
Partition #2

(ABD)(CEFG)



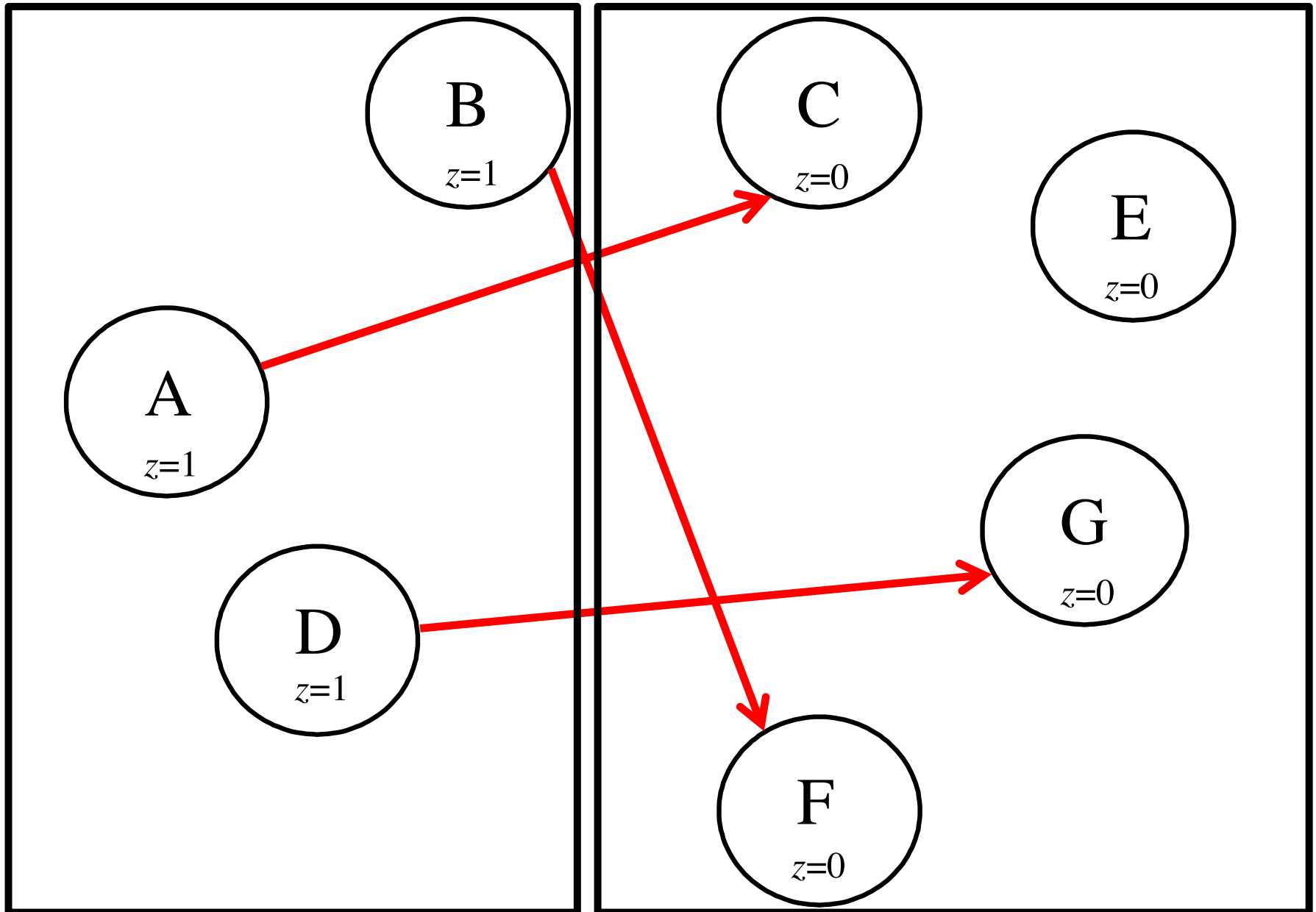
Partition #3.1

(Examine the 0-successors of ABD)



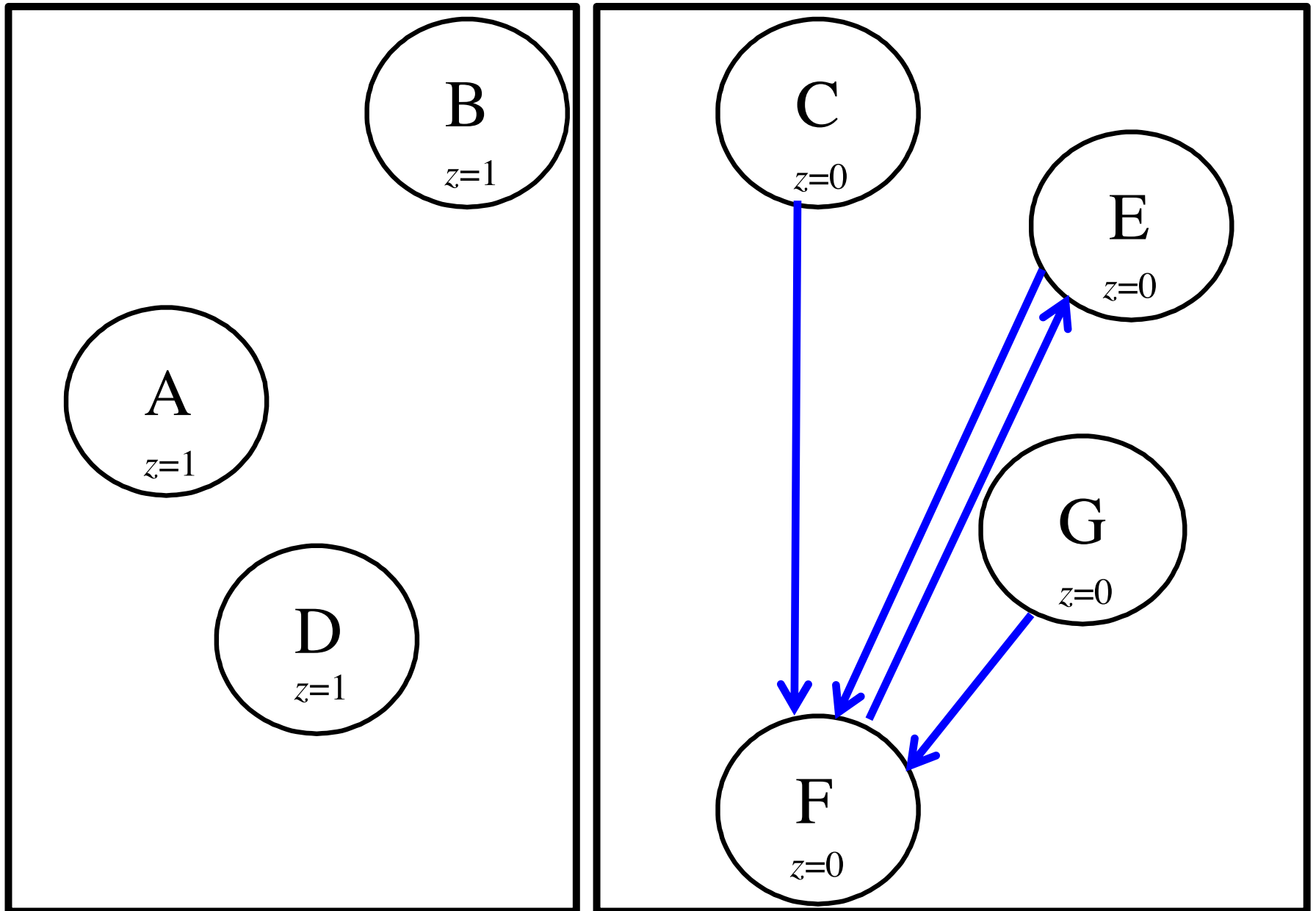
Partition #3.1

(Examine the 1-successors of ABD)



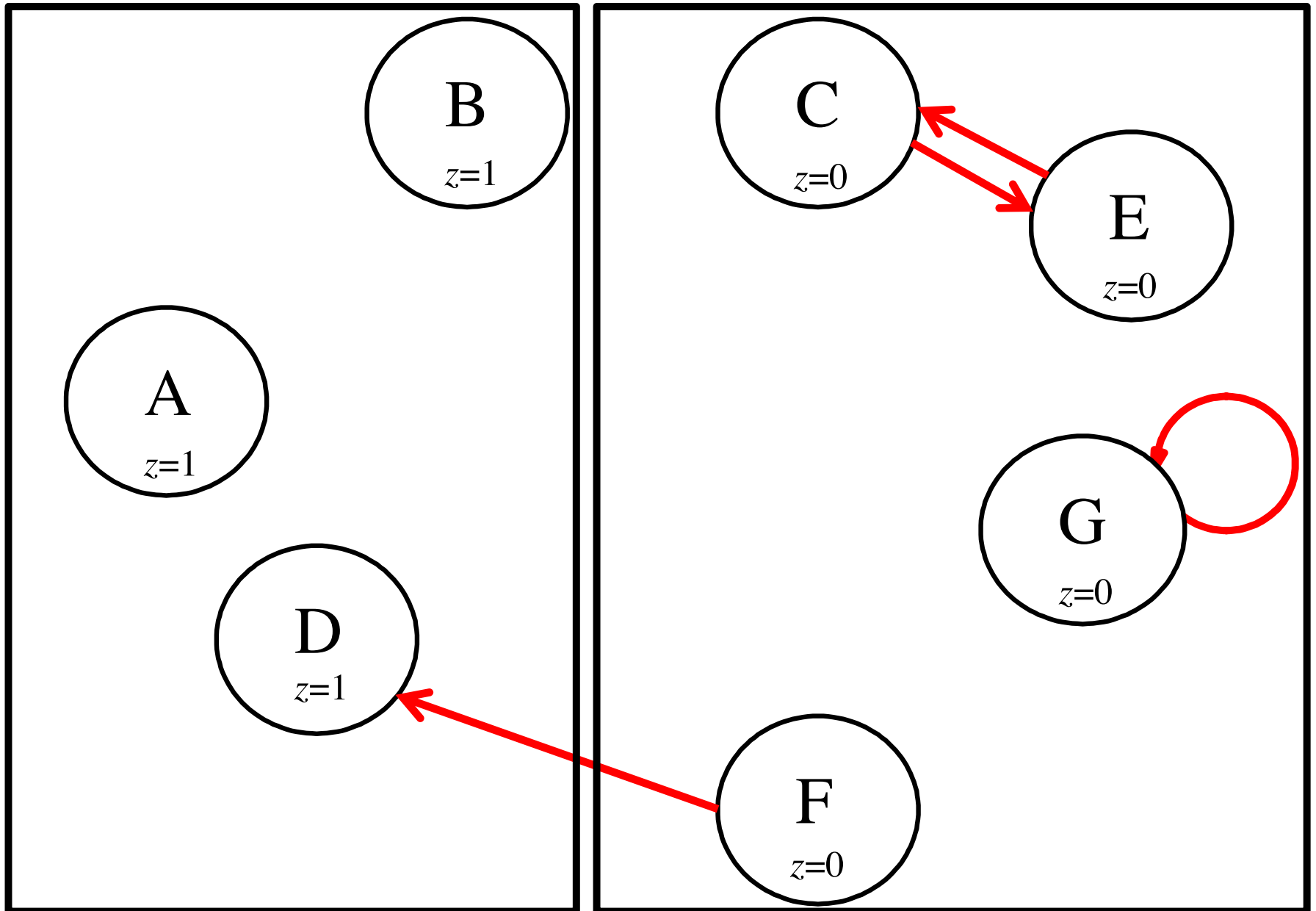
Partition #3.2

(Examine the 0-successors of CEFG)



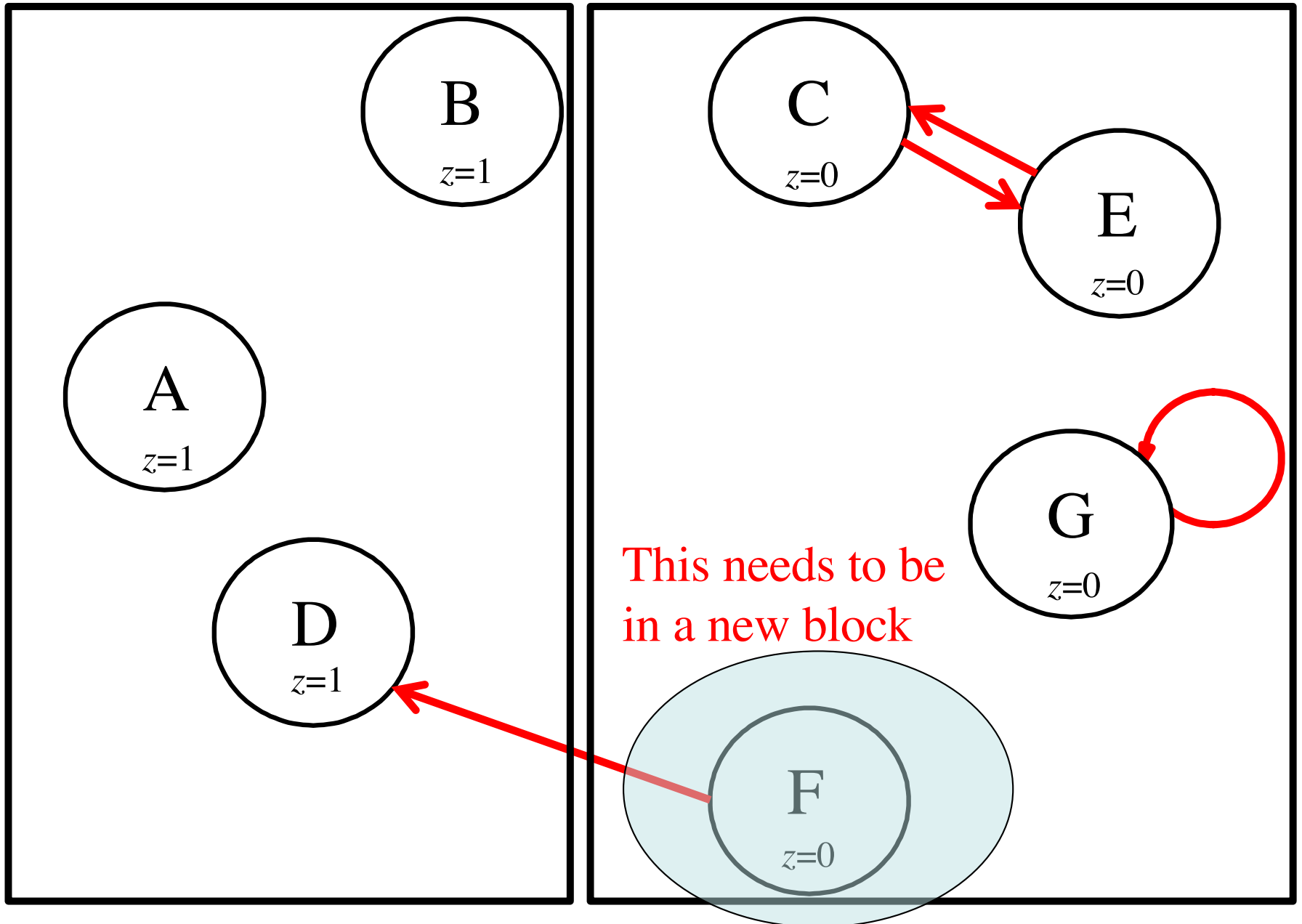
Partition #3.2

(Examine the 1-successors of CEFG)



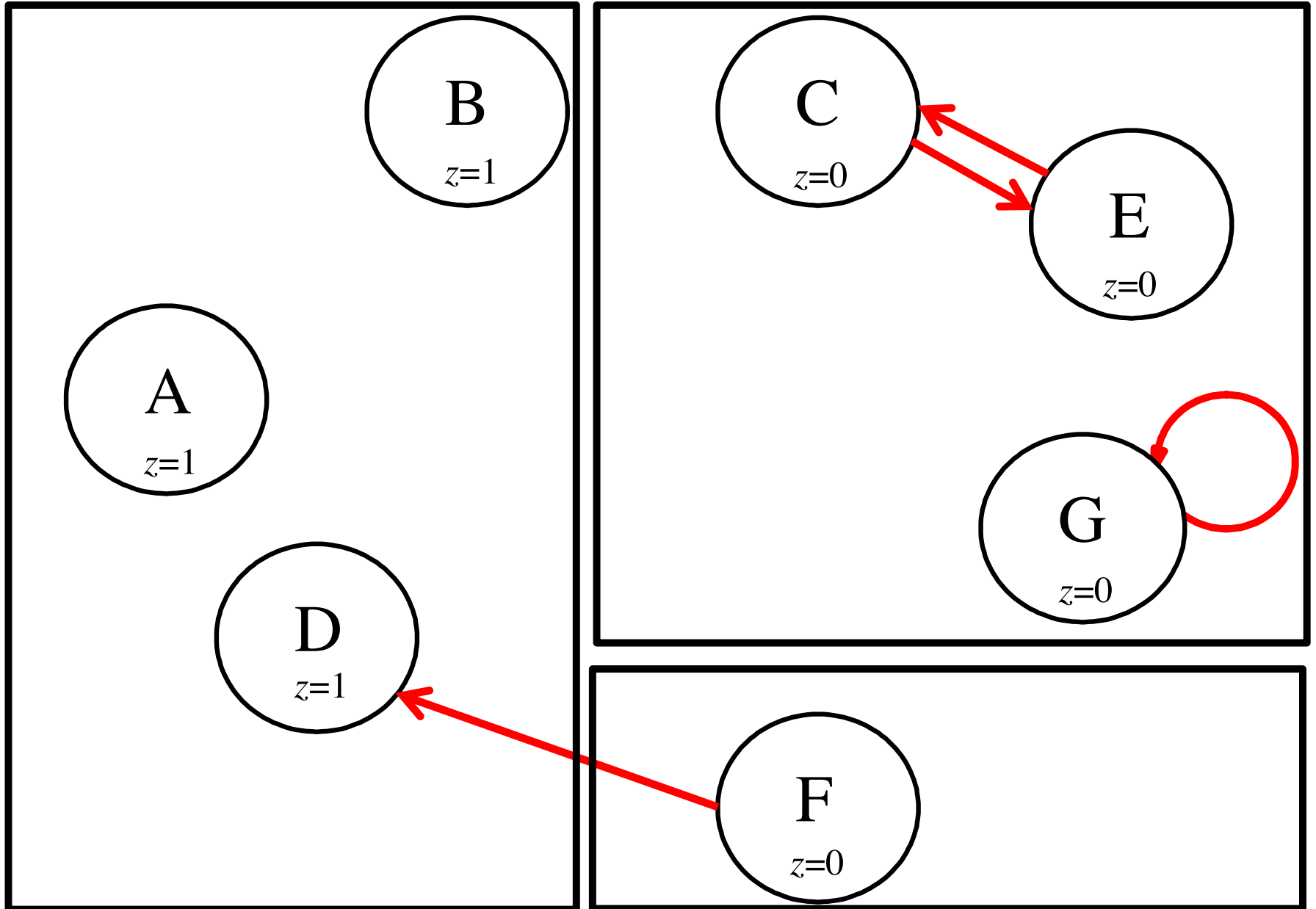
Partition #3.2

(Examine the 1-successors of CEFG)



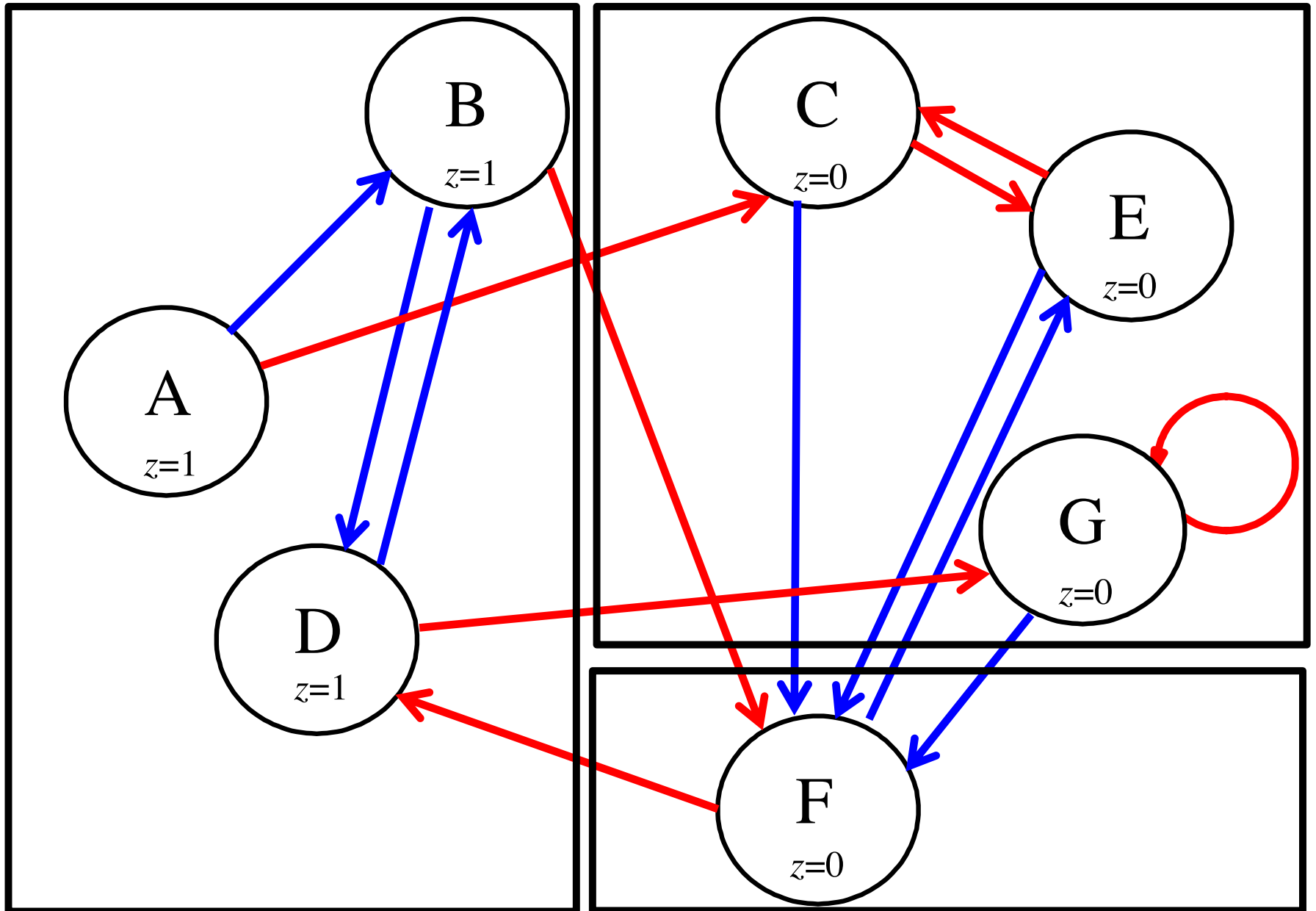
Partition #3

(ABD)(CEG)(F)



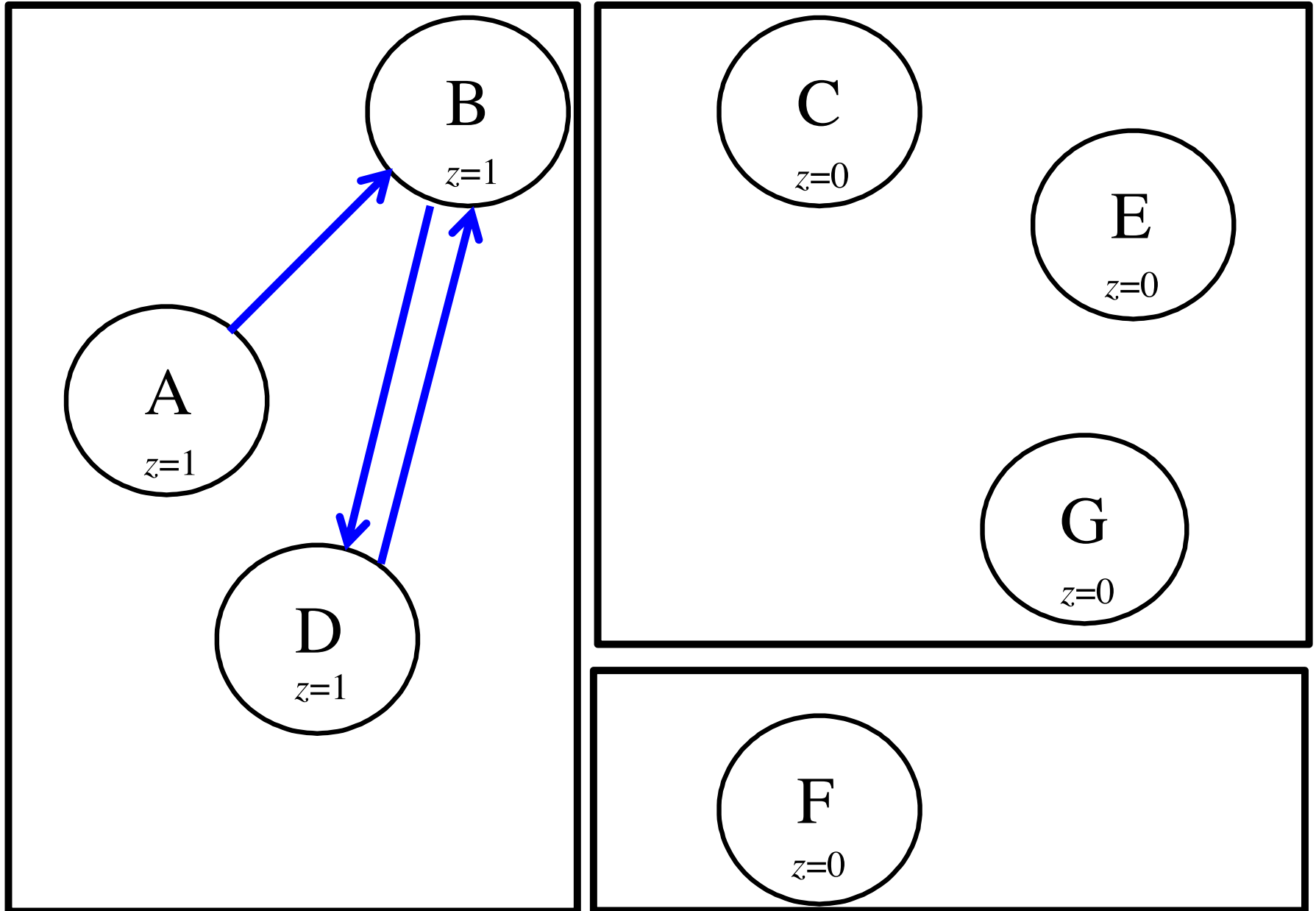
Partition #3

(ABD)(CEG)(F)



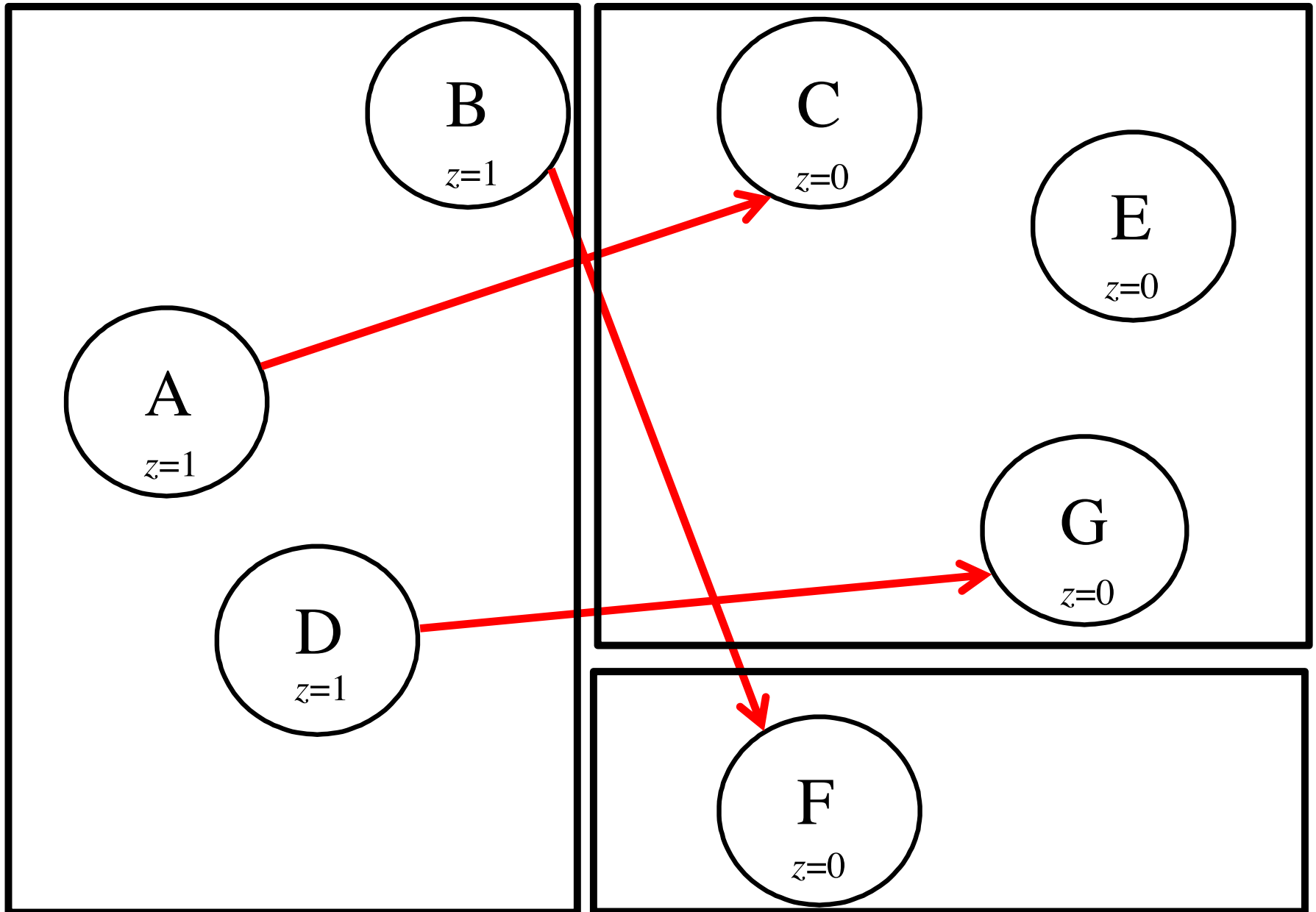
Partition #4.1

(Examine the 0-successors of ABD)



Partition #4.1

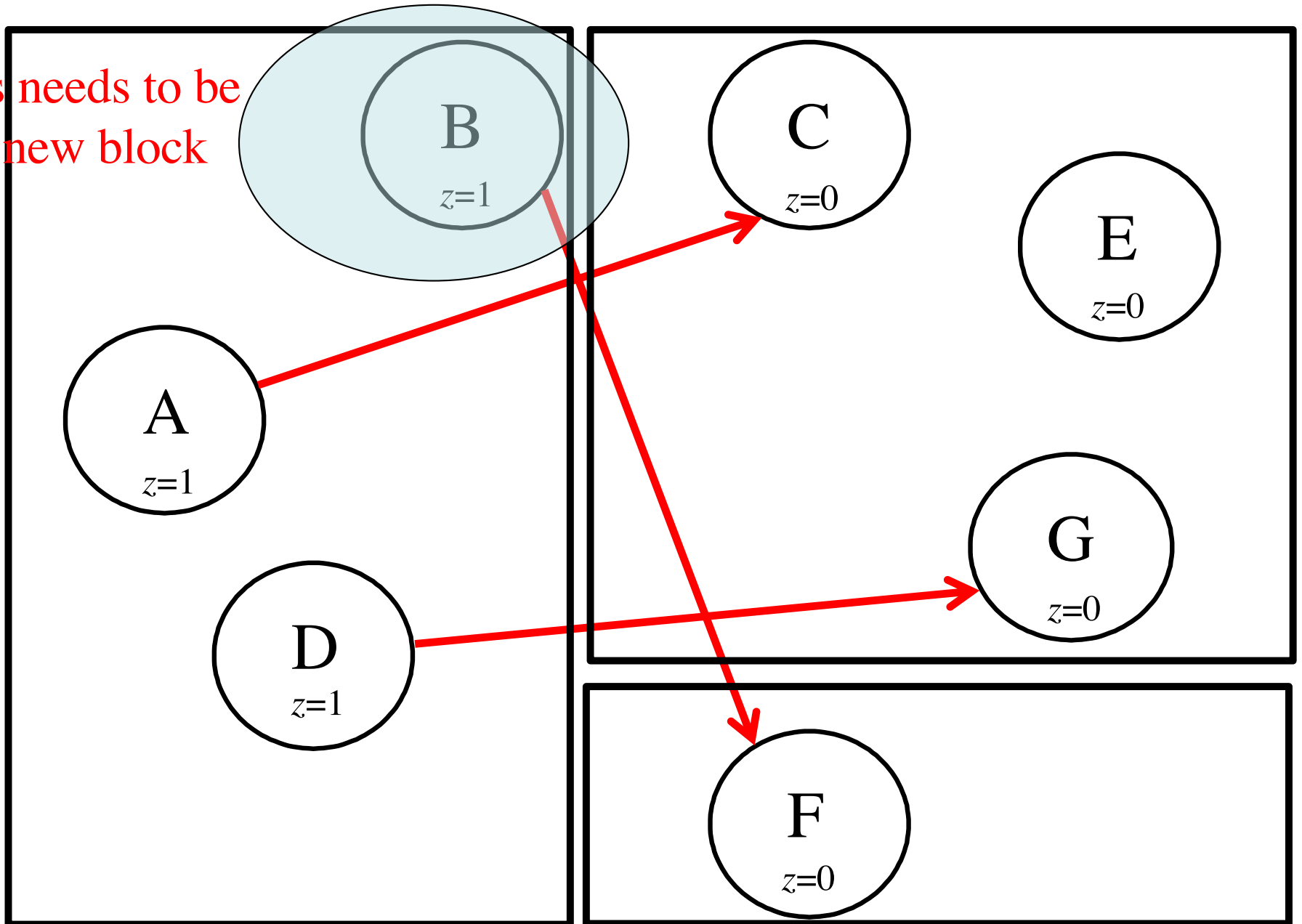
(Examine the 1-successors of ABD)



Partition #4.1

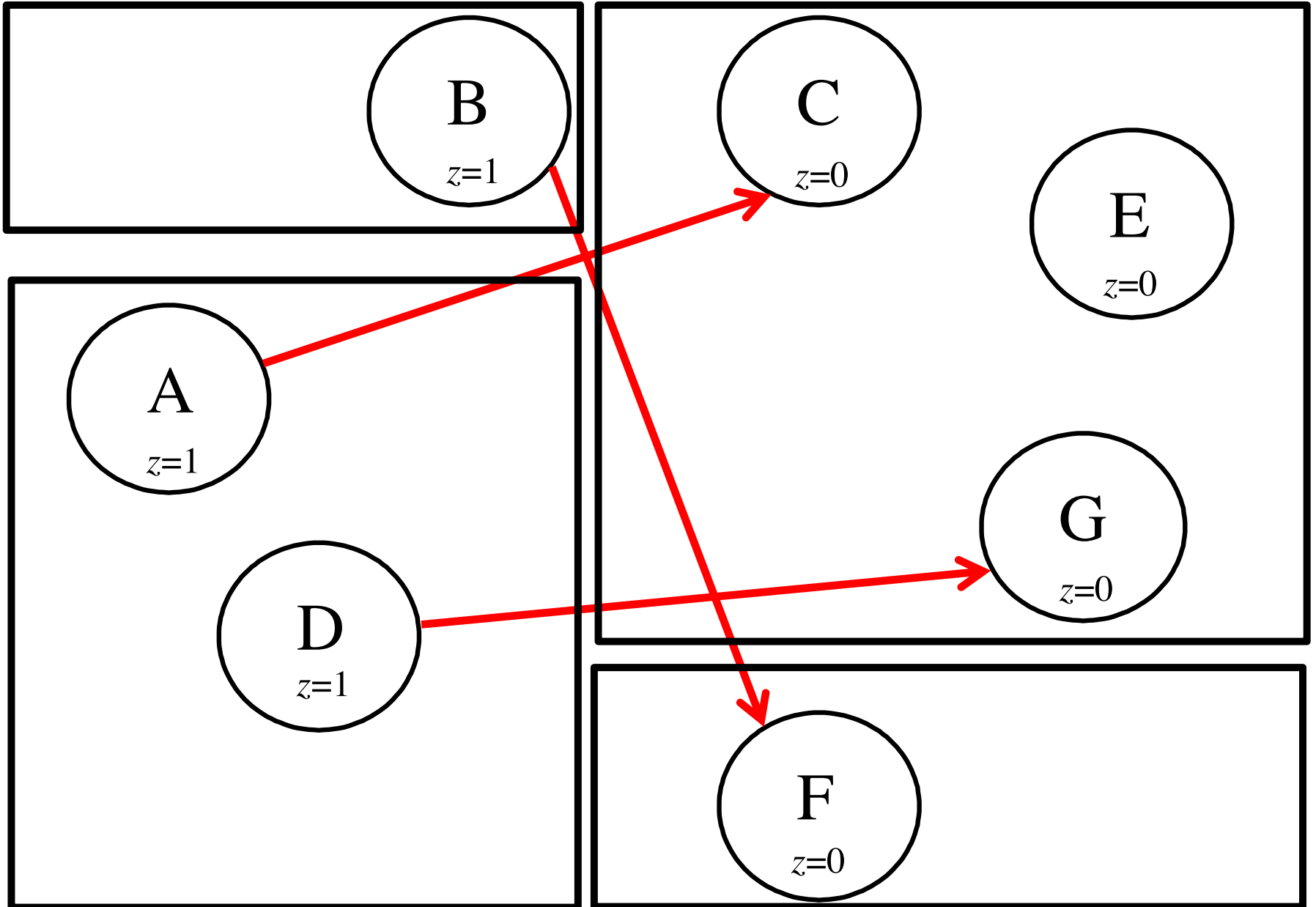
(Examine the 1-successors of ABD)

This needs to be
in a new block



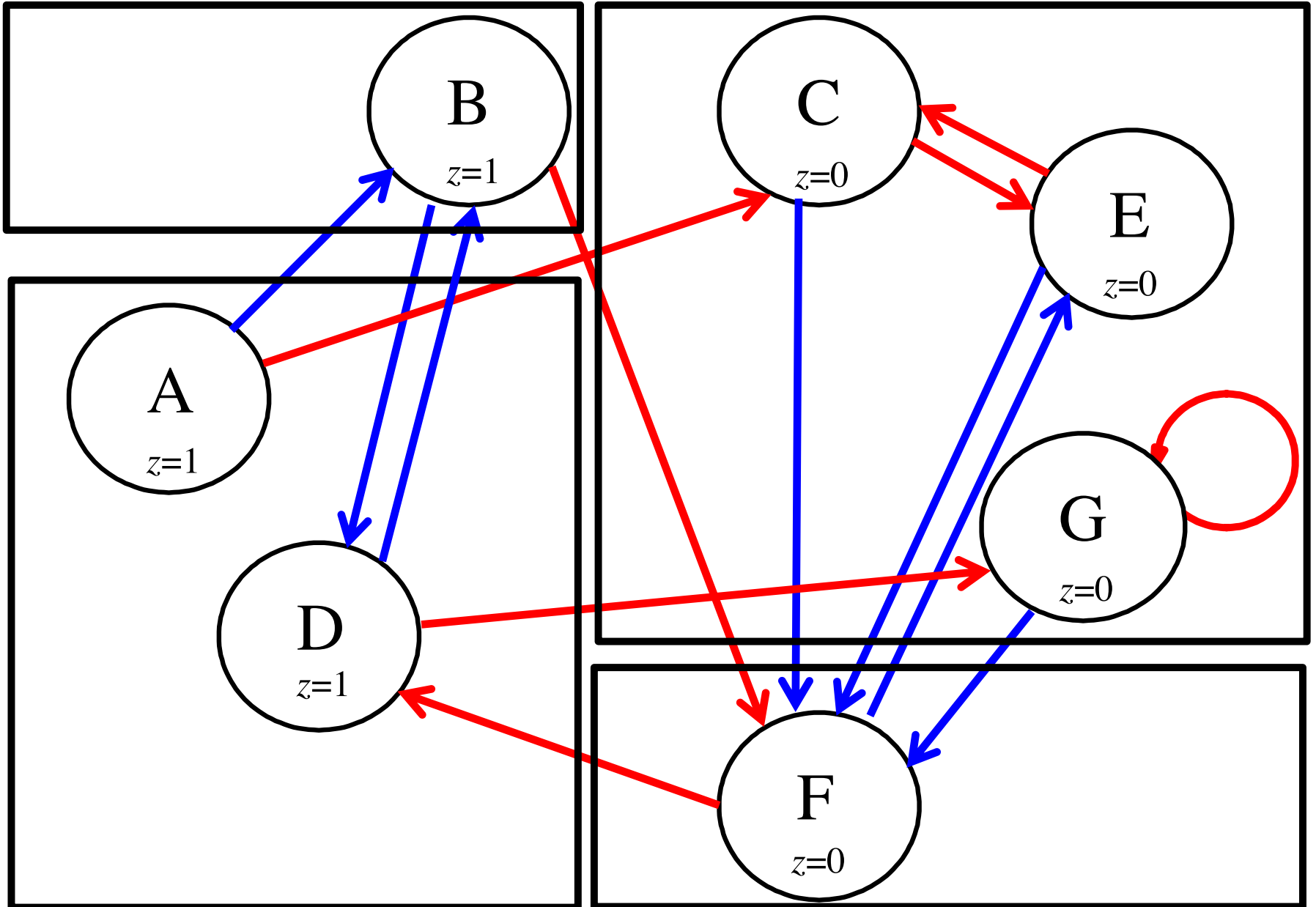
Partition #4

(AD)(B)(CEG)(F)



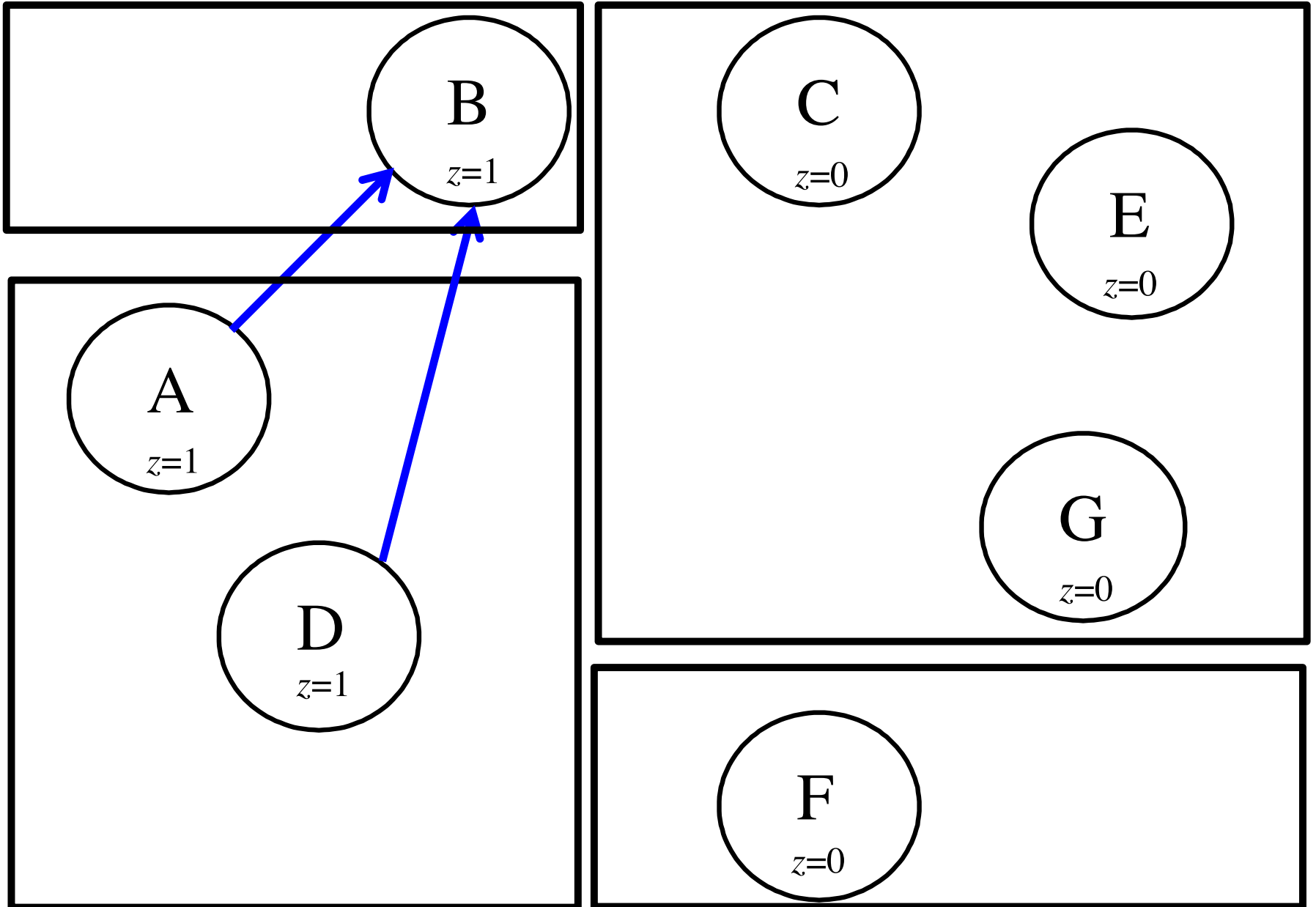
Partition #4

(AD)(B)(CEG)(F)



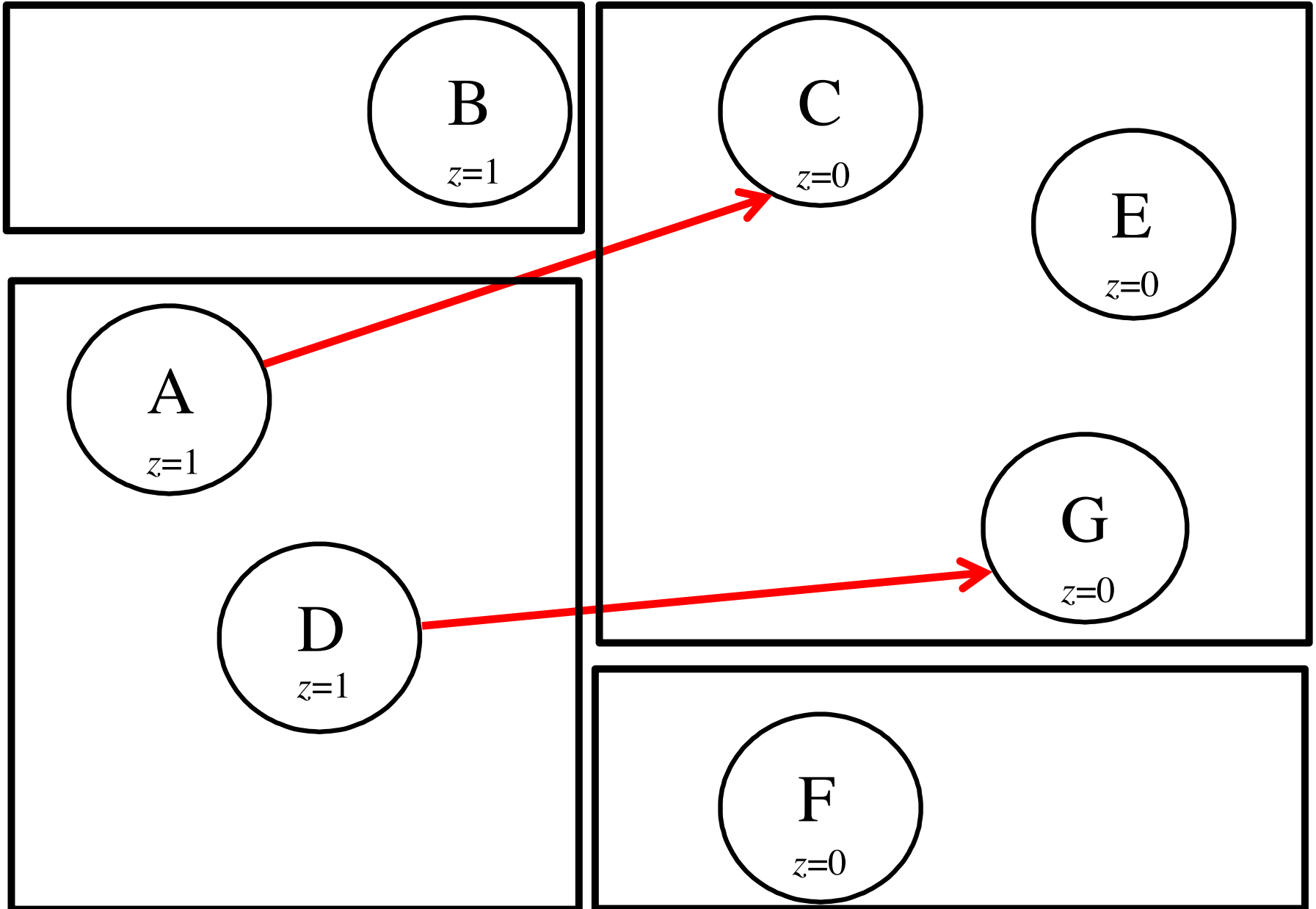
Partition #5.1

(Examine the 0-successors of AD)



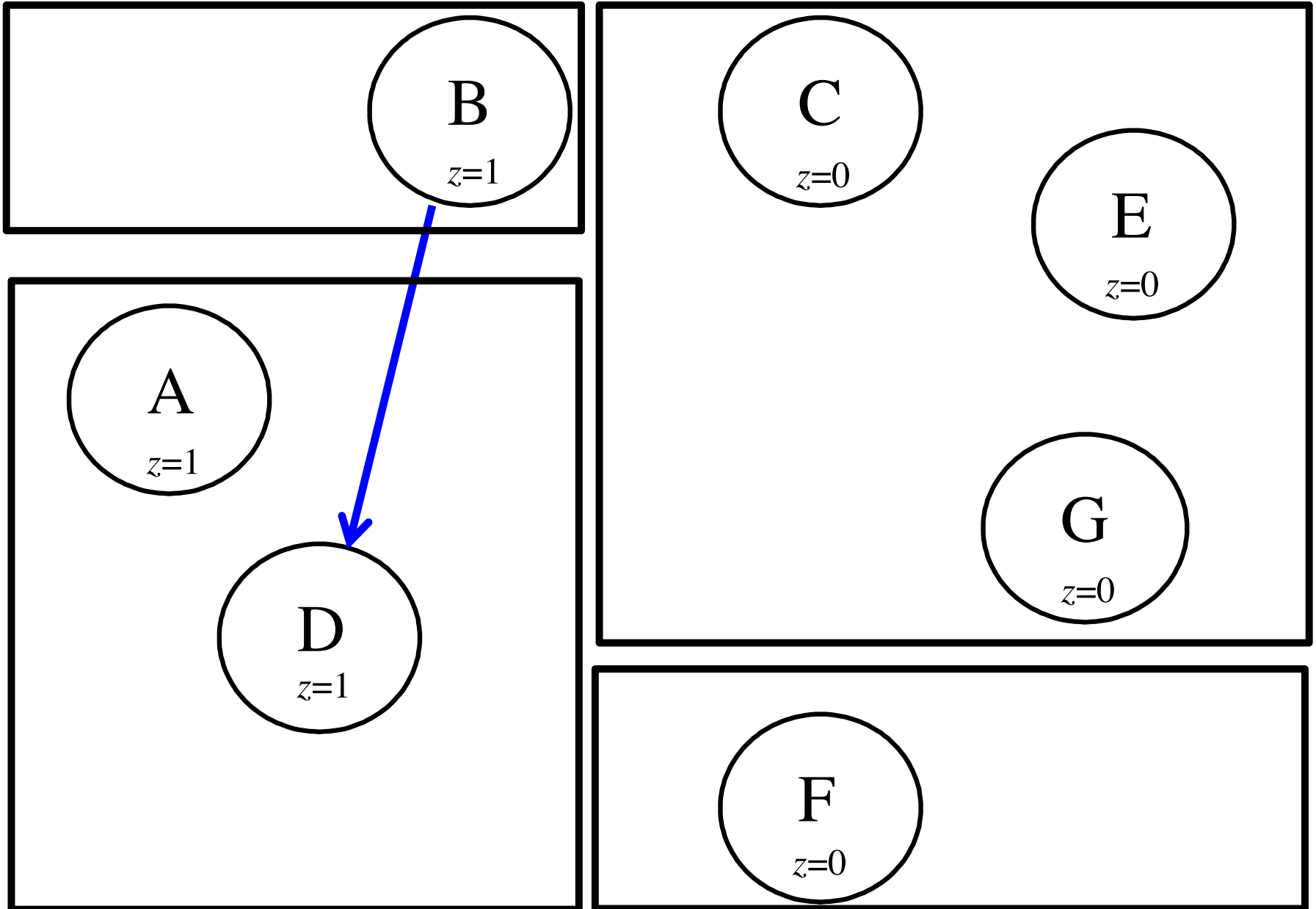
Partition #5.1

(Examine the 1-successors of AD)



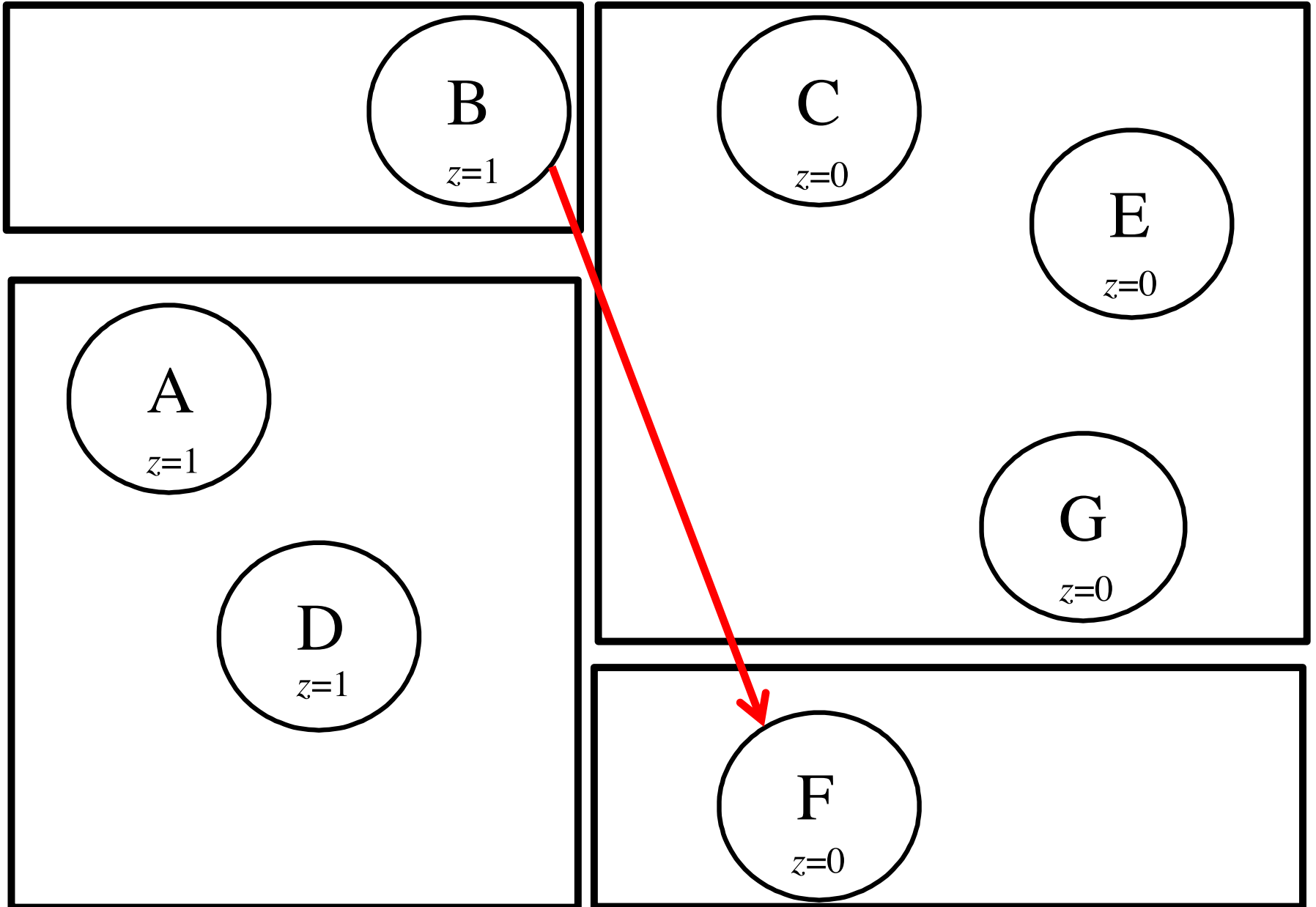
Partition #5.2

(Examine the 0-successors of B)



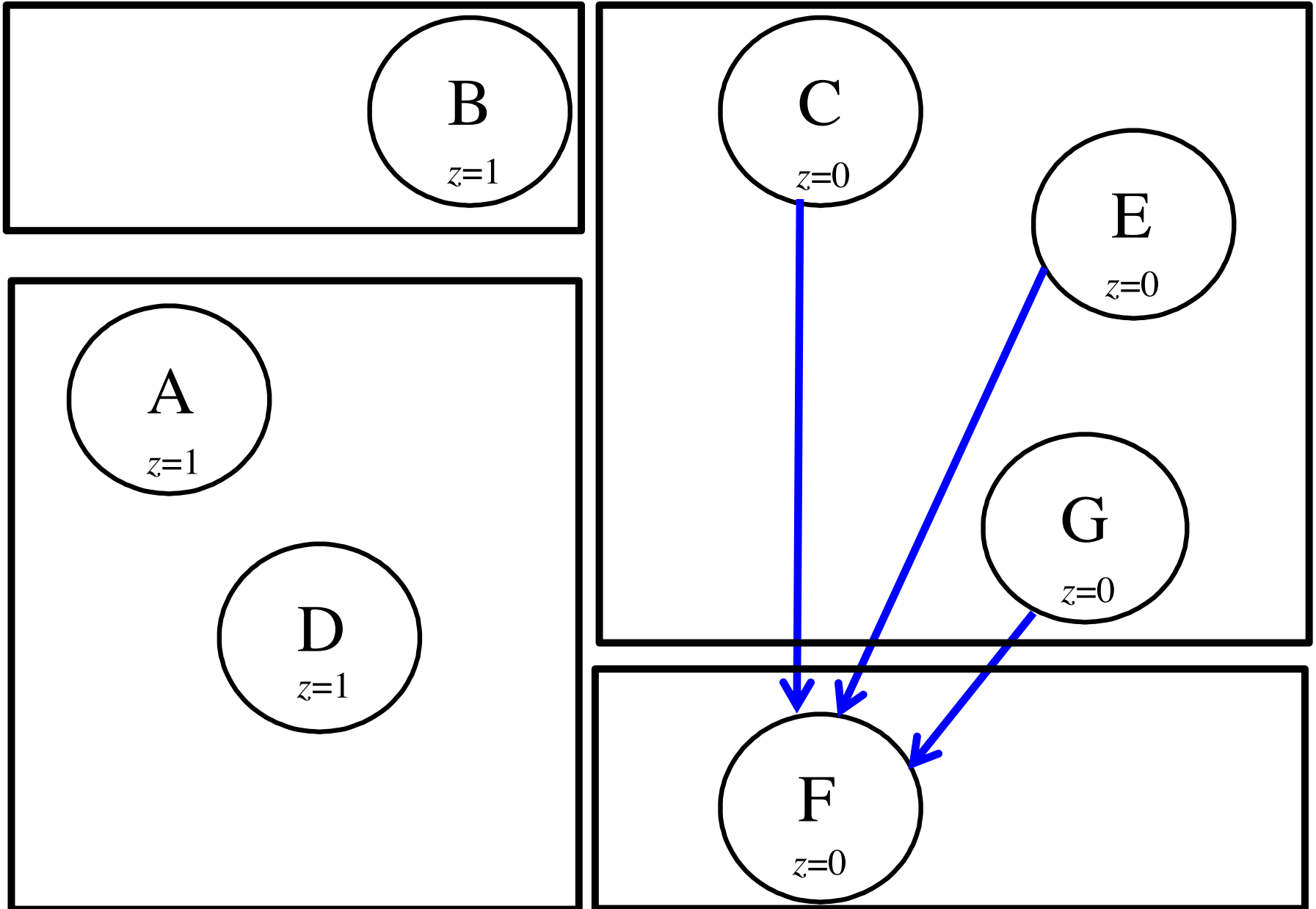
Partition #5.2

(Examine the 1-successors of B)



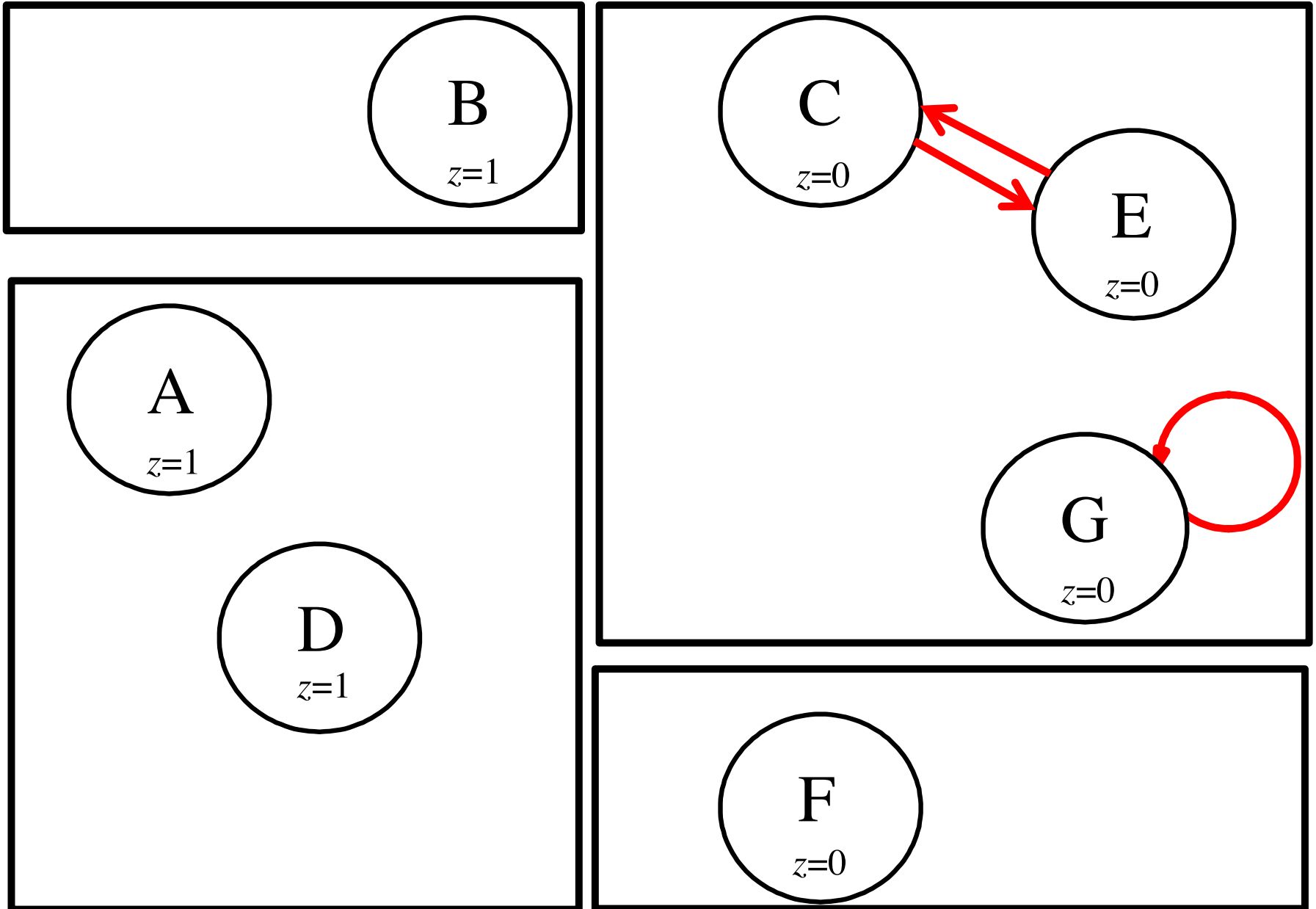
Partition #5.3

(Examine the 0-successors of CEG)



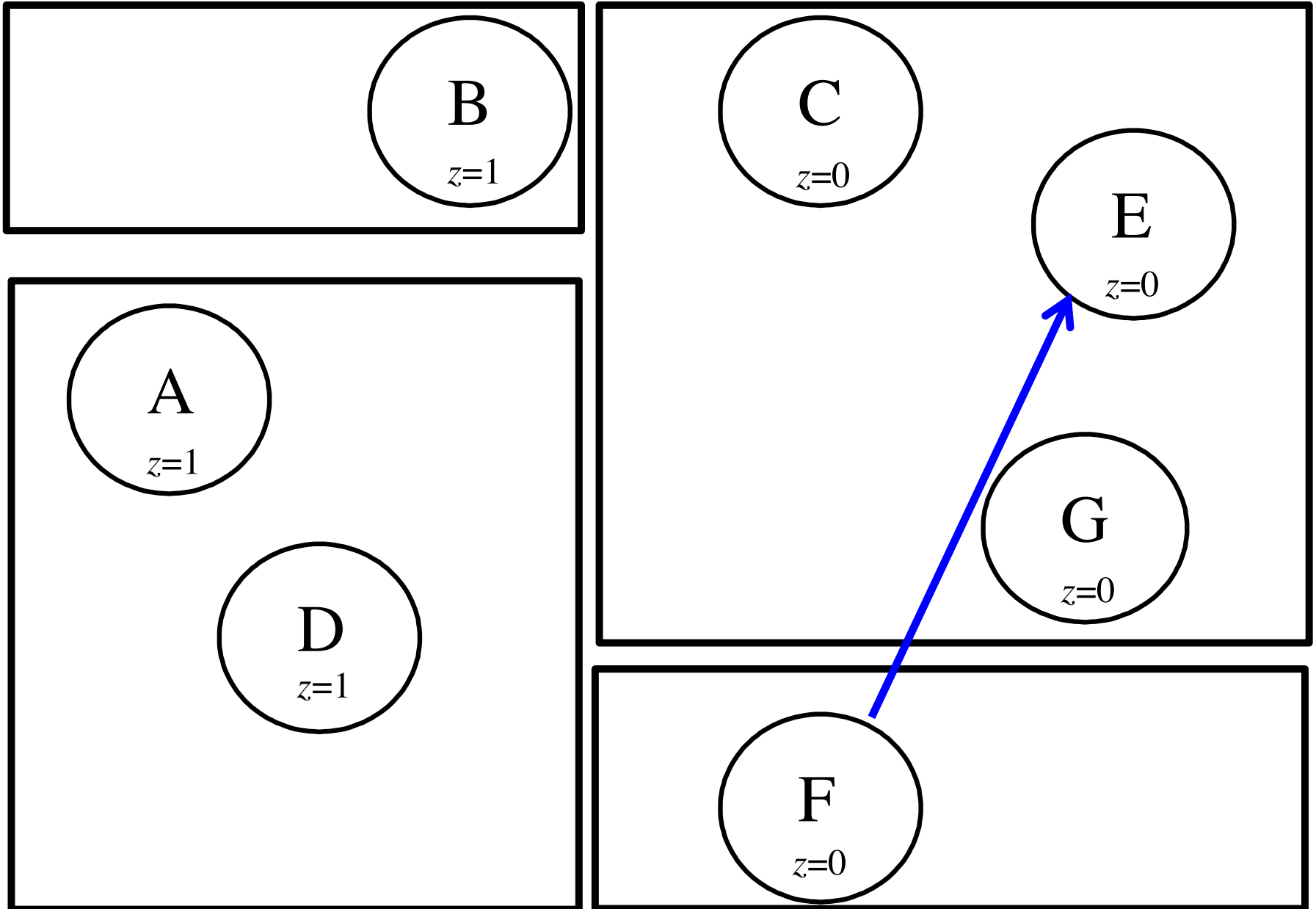
Partition #5.3

(Examine the 1-successors of CEG)



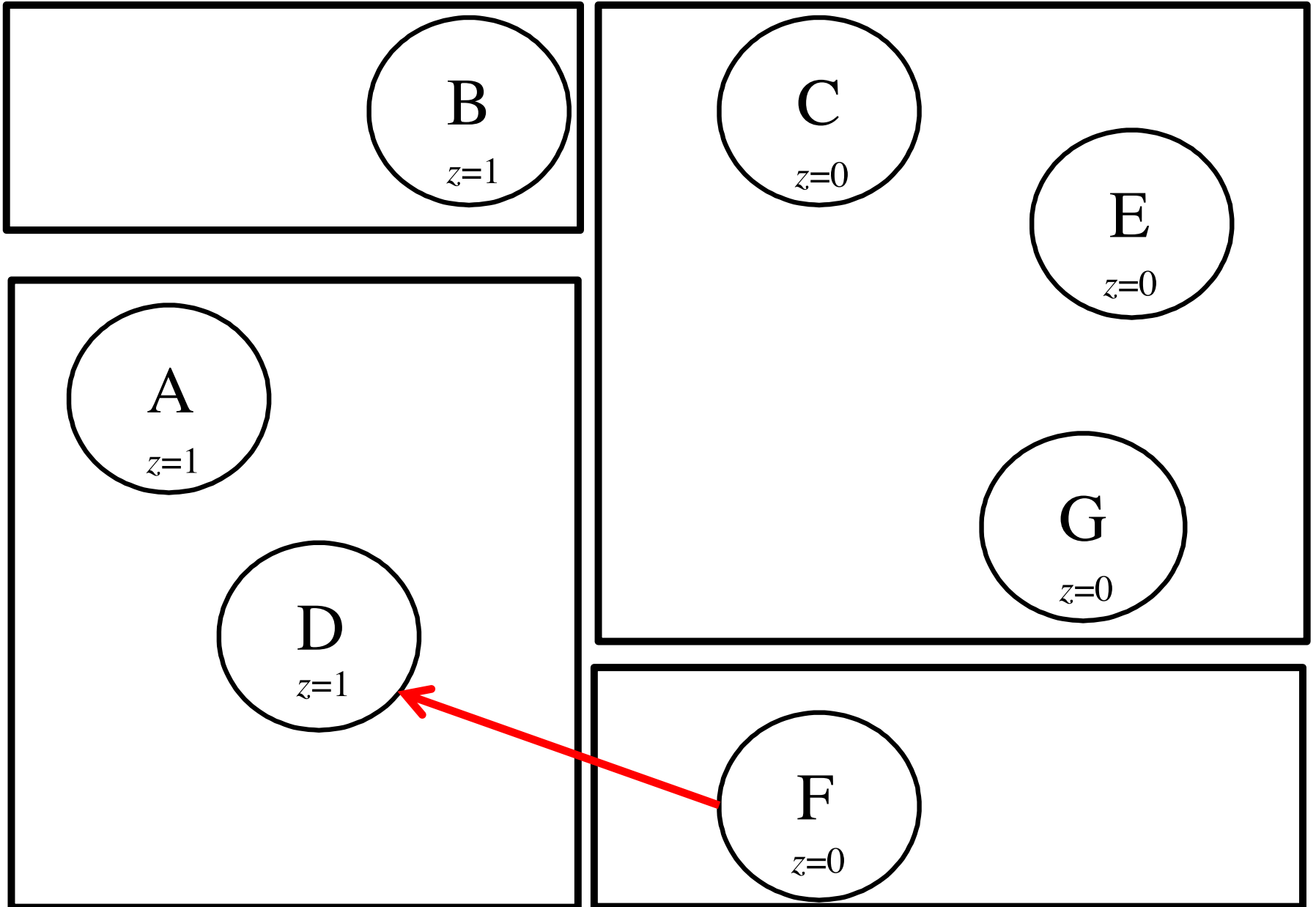
Partition #5.4

(Examine the 0-successors of F)



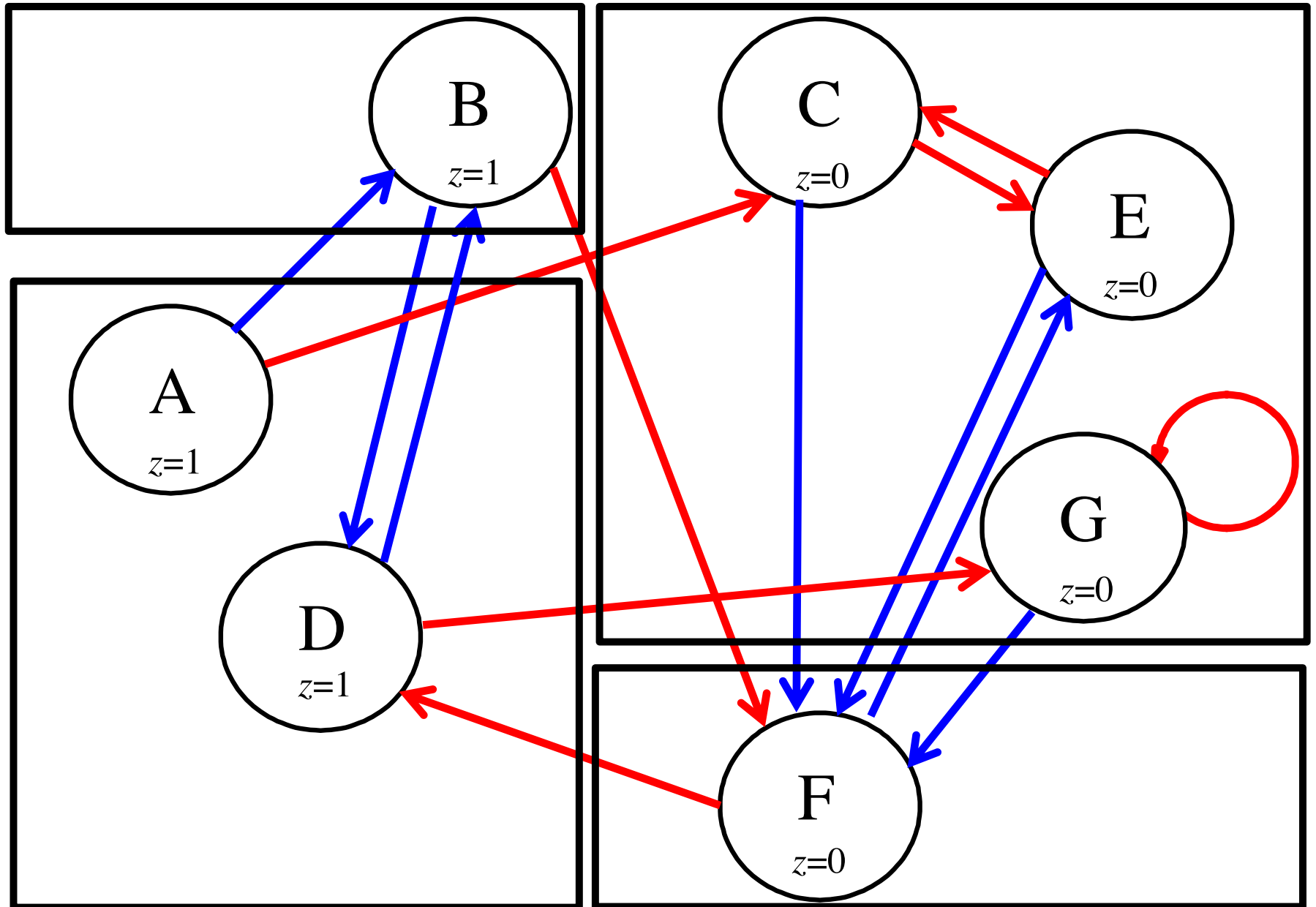
Partition #5.4

(Examine the 1-successors of F)



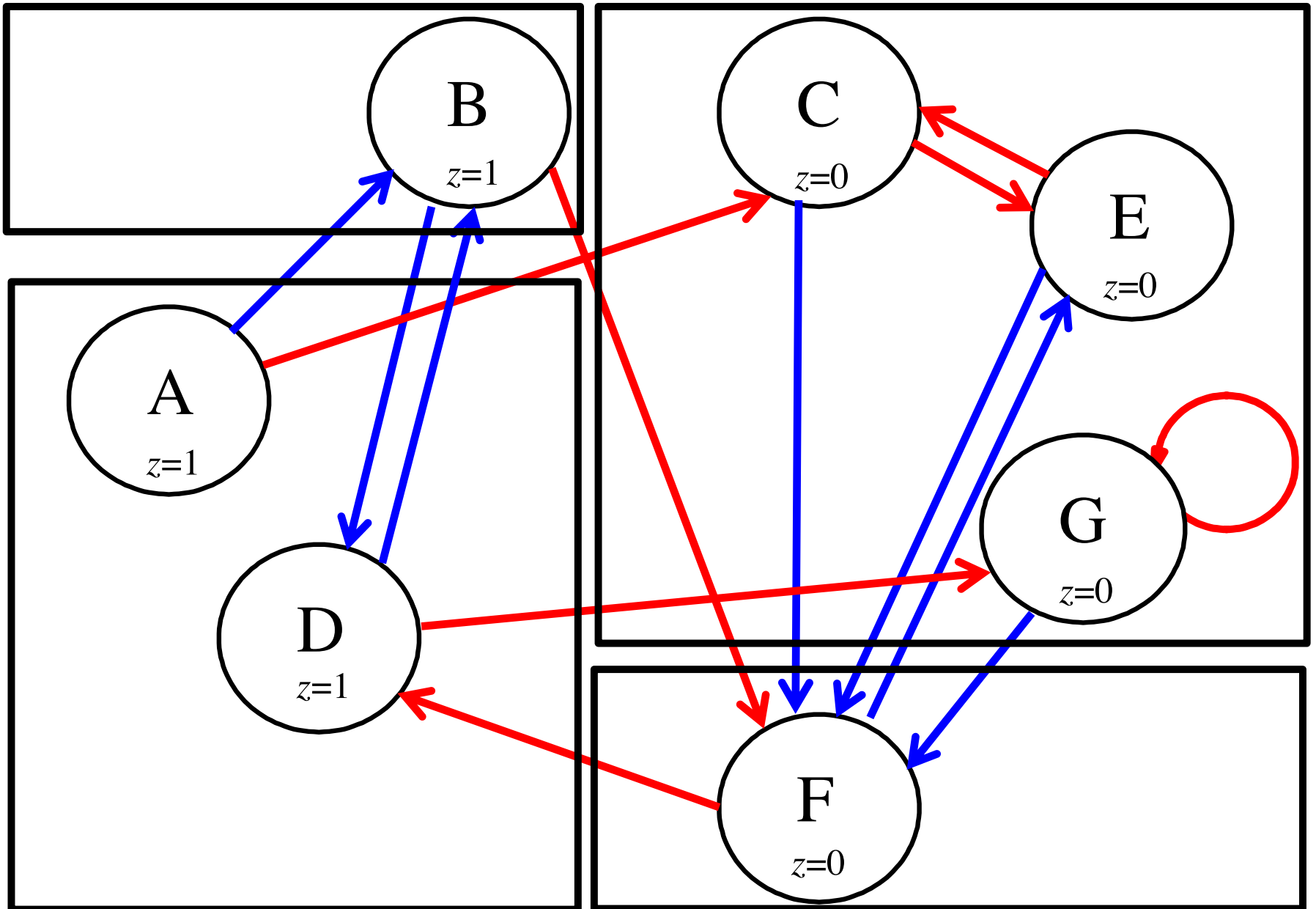
Partition #5

(AD)(B)(CEG)(F)



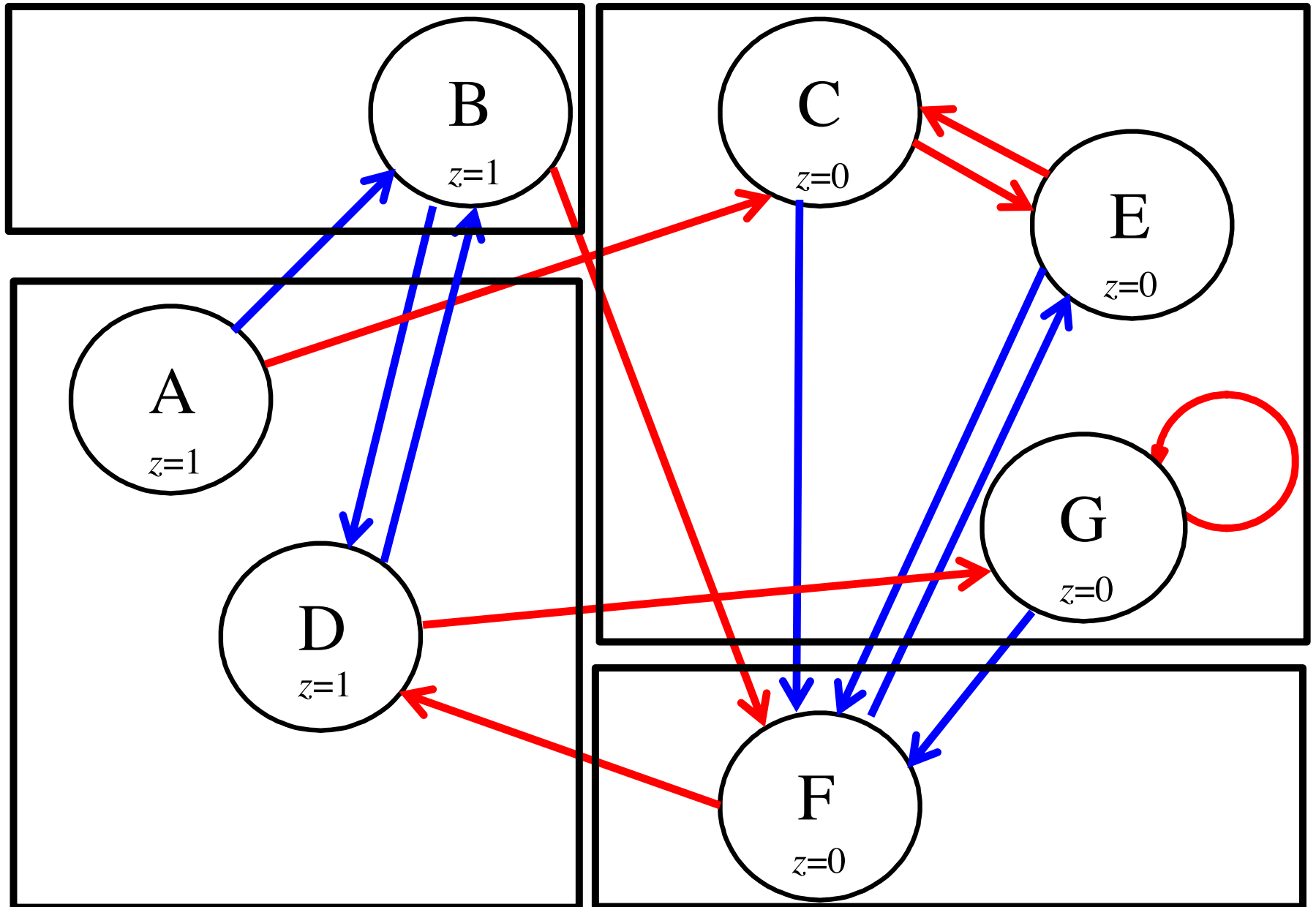
Partition #4

(AD)(B)(CEG)(F)



Partition #5

(This is the same as #4 so we can stop here)



Minimized state table

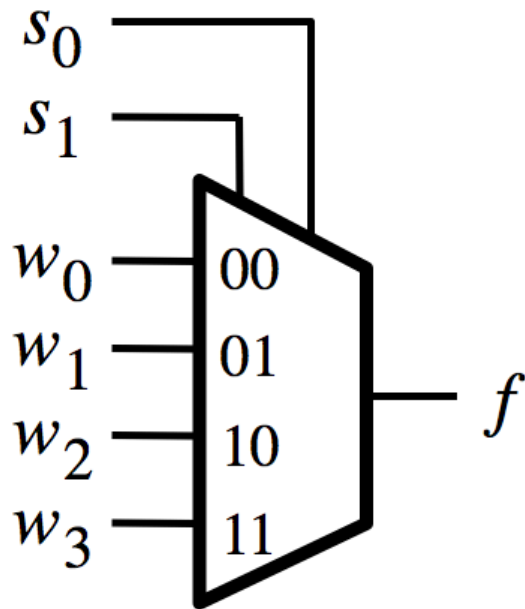
Present state	Nextstate		Output z
	w = 0	w = 1	
A	B	C	1
B	A	F	1
C	F	C	0
F	C	A	0

Multiplexers

4-1 Multiplexer (Definition)

- Has four inputs: w_0 , w_1 , w_2 , w_3
- Also has two select lines: s_1 and s_0
- If $s_1=0$ and $s_0=0$, then the output f is equal to w_0
- If $s_1=0$ and $s_0=1$, then the output f is equal to w_1
- If $s_1=1$ and $s_0=0$, then the output f is equal to w_2
- If $s_1=1$ and $s_0=1$, then the output f is equal to w_3

Graphical Symbol and Truth Table

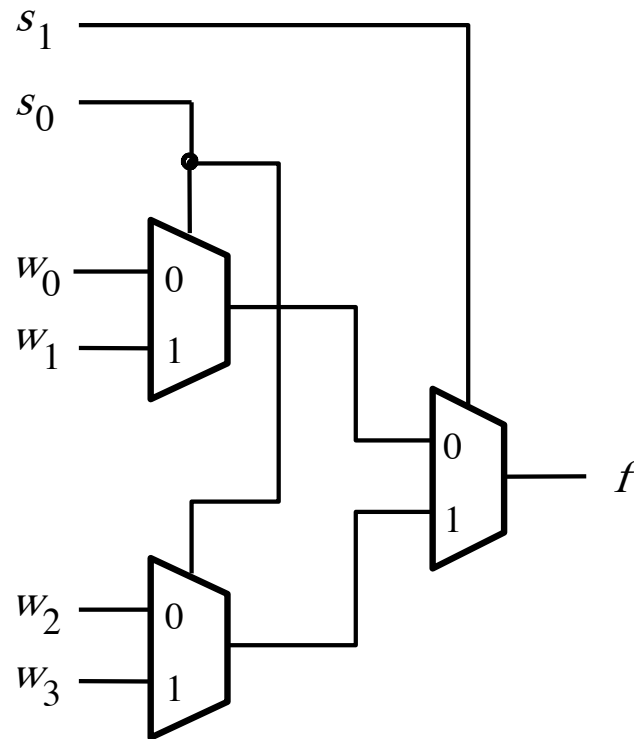


(a) Graphic symbol

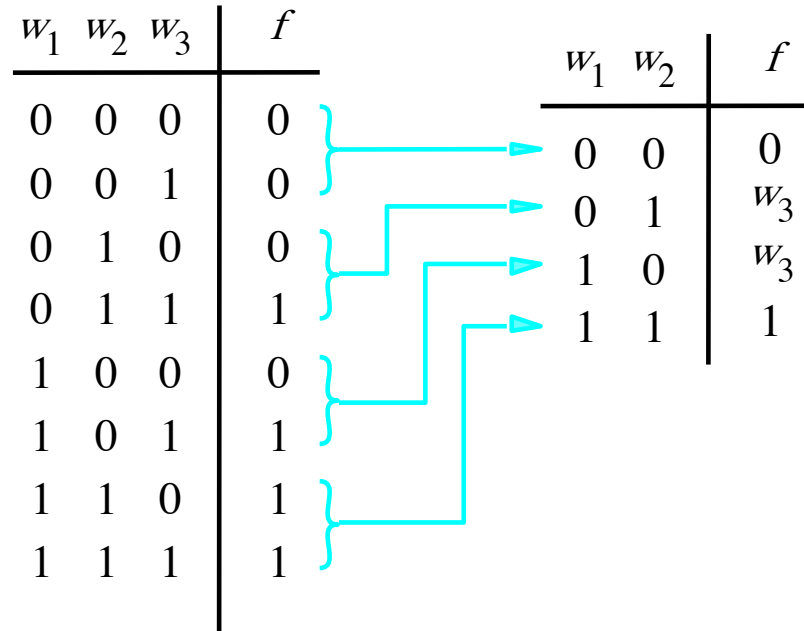
s_1	s_0	f
0	0	w_0
0	1	w_1
1	0	w_2
1	1	w_3

(b) Truth table

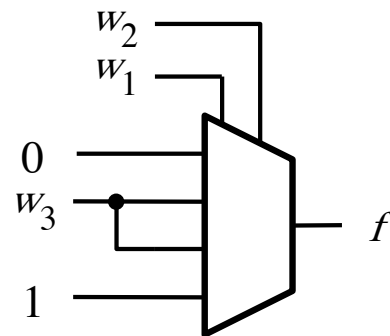
Using three 2-to-1 multiplexers to build one 4-to-1 multiplexer



Implementation of a logic function



(a) Modified truth table



(b) Circuit

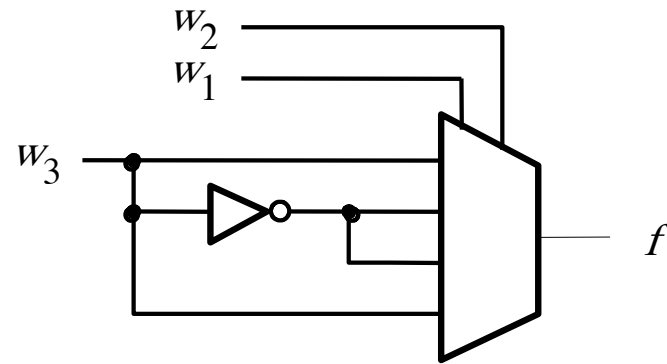
Implementation of 3-input XOR with a 4-to-1 Multiplexer

w_1	w_2	w_3	f	
0	0	0	0	} w_3
0	0	1	1	
0	1	0	1	} \bar{w}_3
0	1	1	0	
1	0	0	1	} \bar{w}_3
1	0	1	0	
1	1	0	0	} w_3
1	1	1	1	

Implementation of 3-input XOR with a 4-to-1 Multiplexer

w_1	w_2	w_3	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

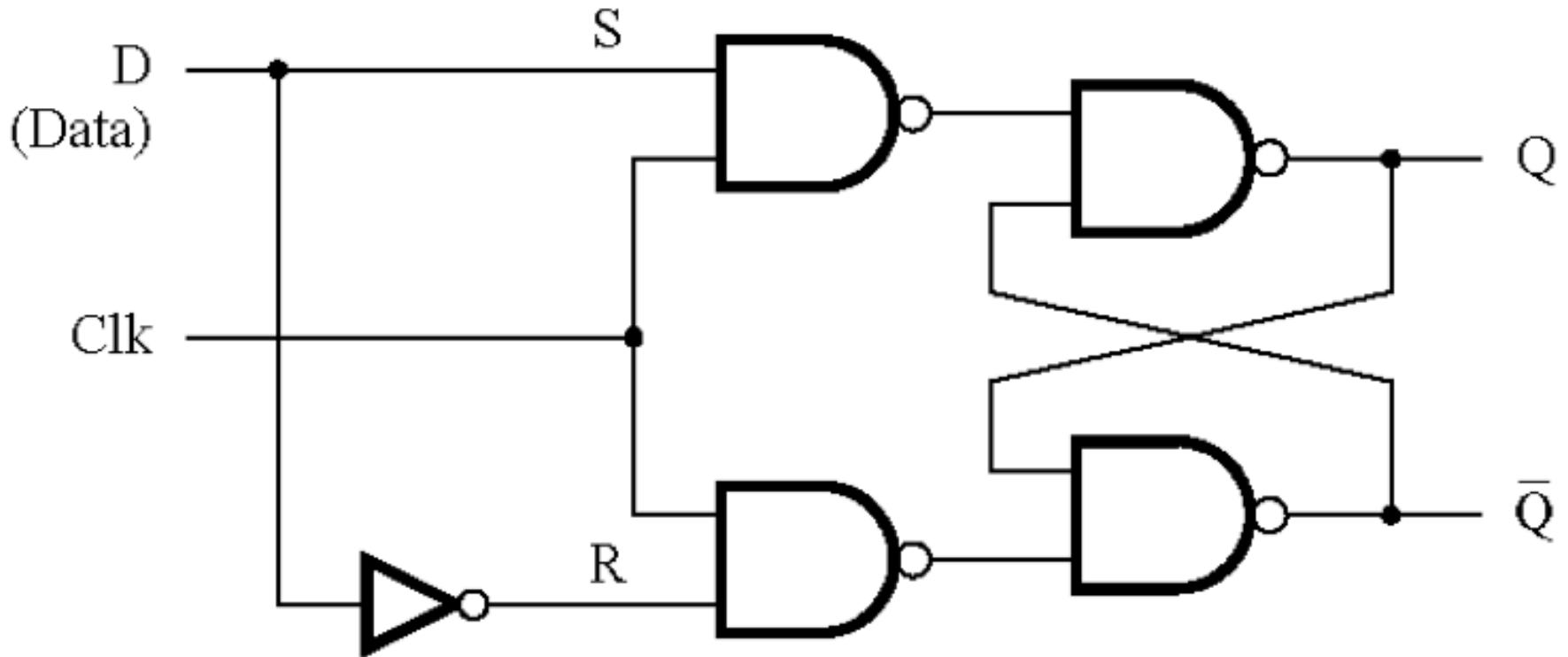
(a) Truth table



(b) Circuit

Gated D Latch

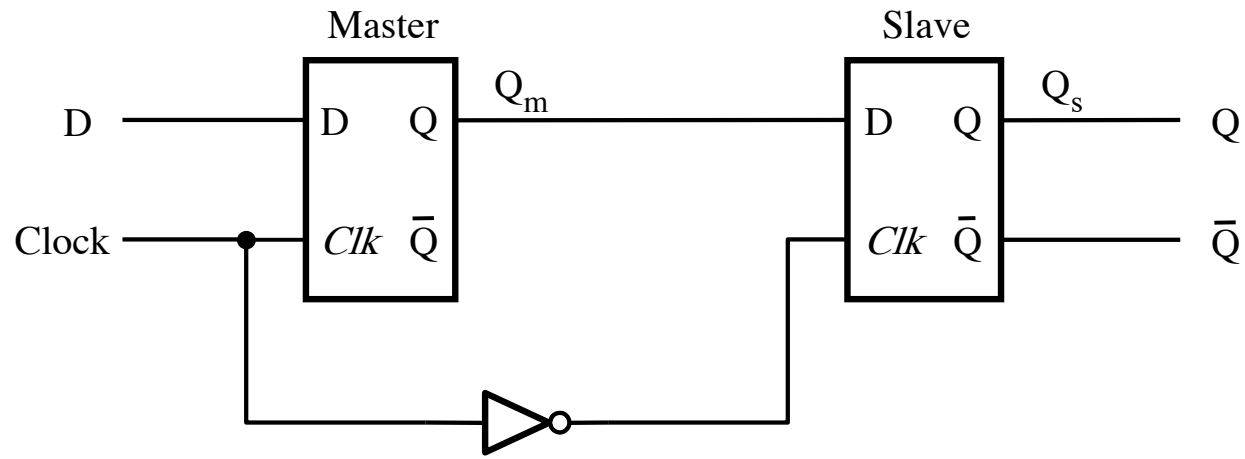
Circuit Diagram for the Gated D Latch



[Figure 5.7a from the textbook]

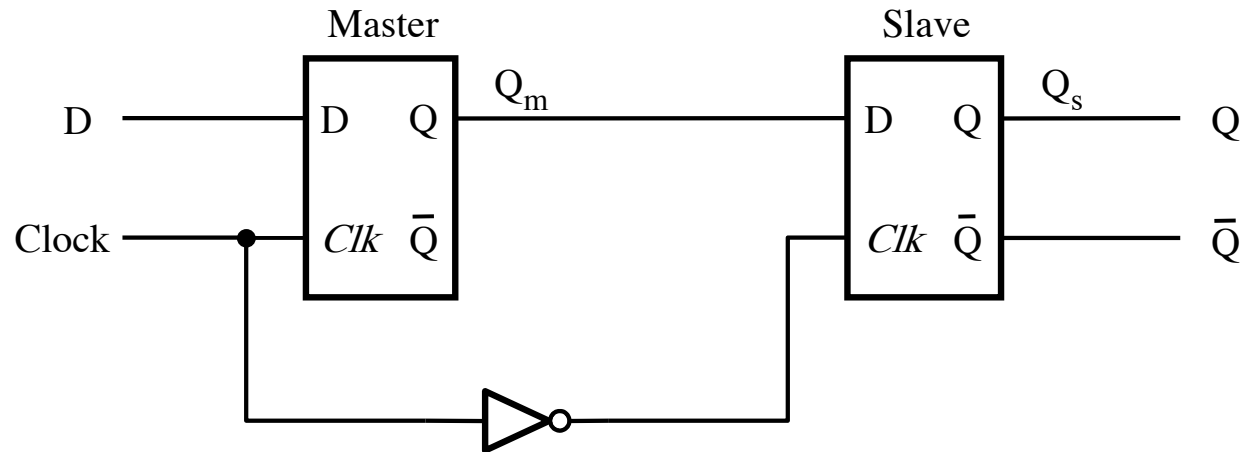
Edge-Triggered D Flip-Flops

Master-Slave D Flip-Flop

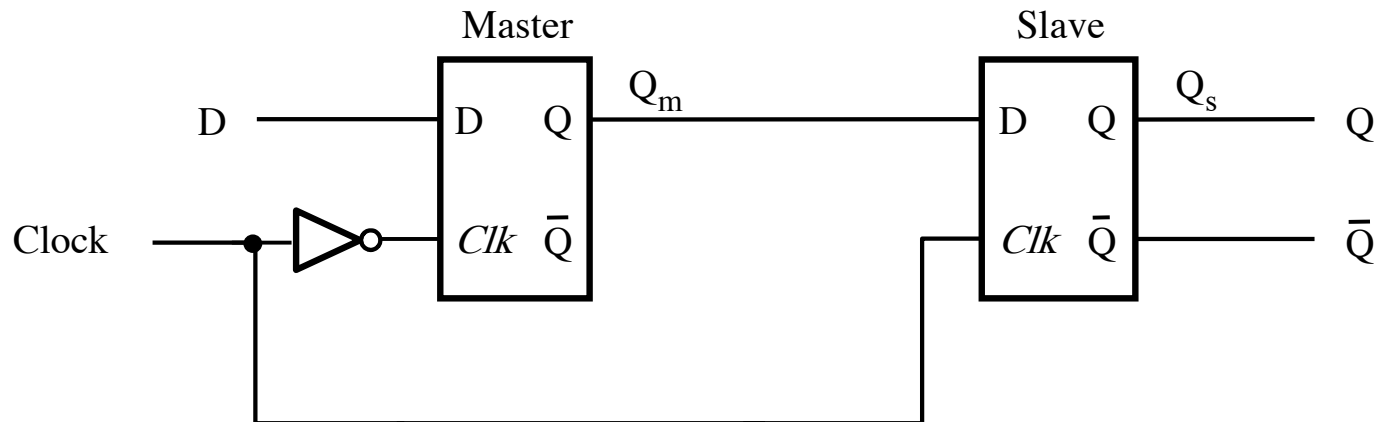


(a) Circuit

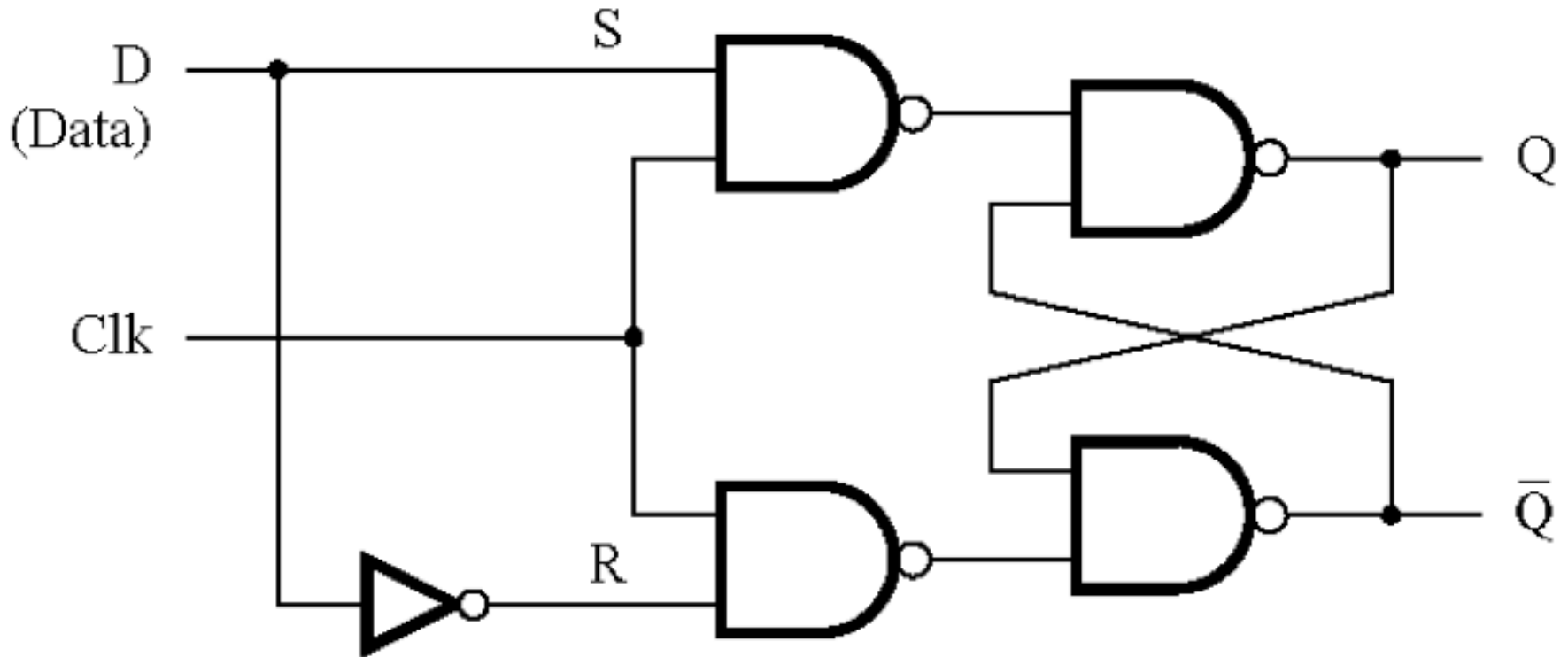
Negative-Edge-Triggered Master-Slave D Flip-Flop



Positive-Edge-Triggered Master-Slave D Flip-Flop

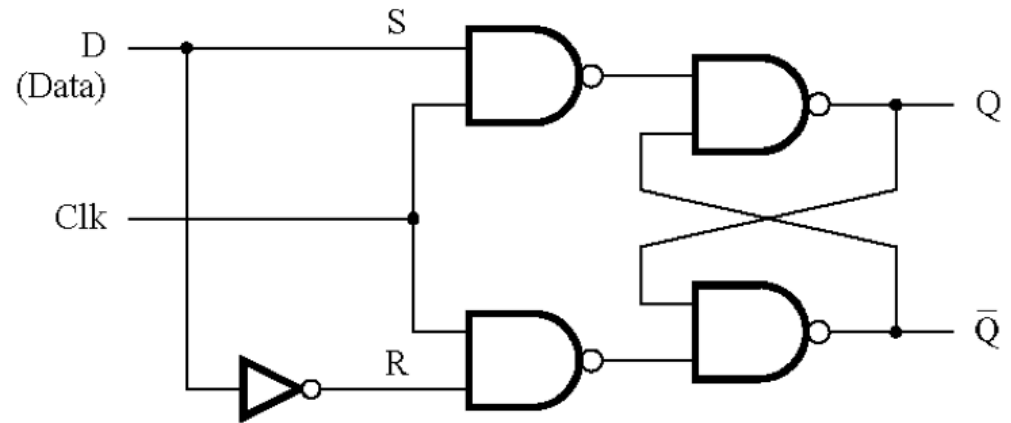
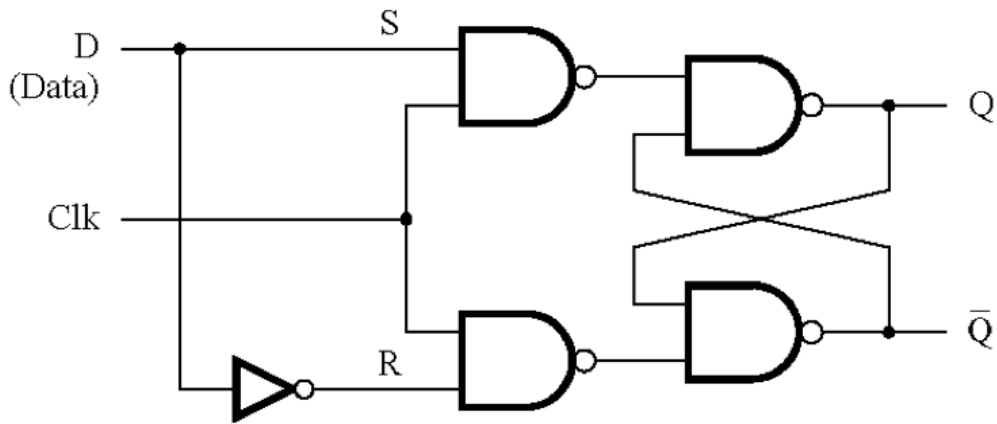


Circuit Diagram for the Gated D Latch

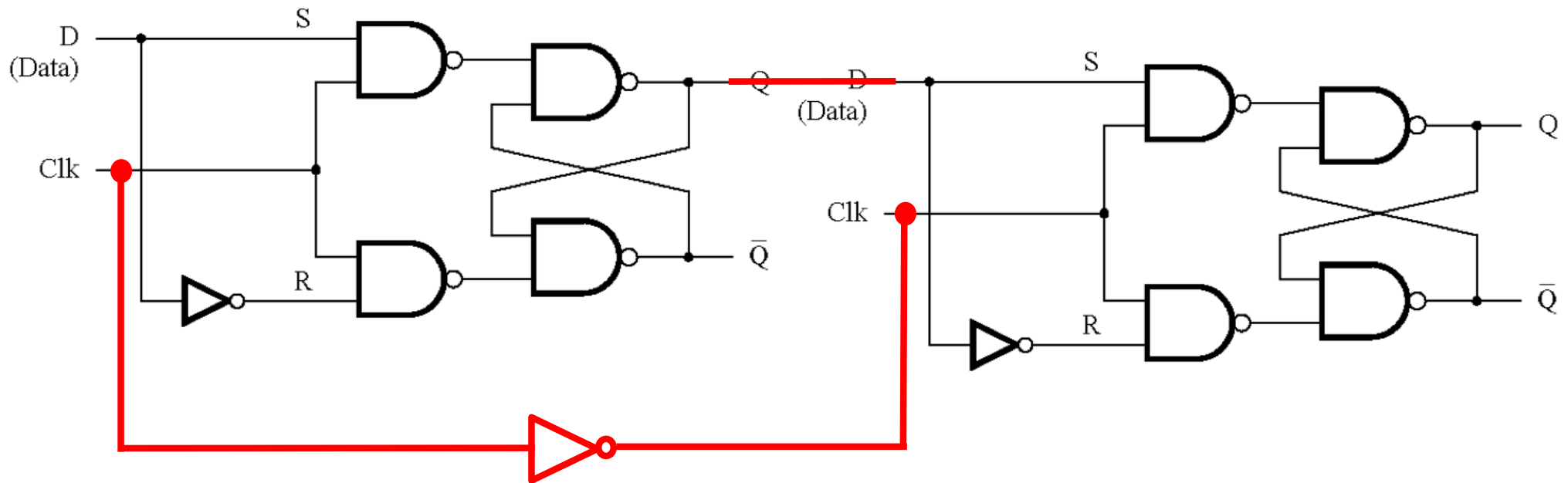


[Figure 5.7a from the textbook]

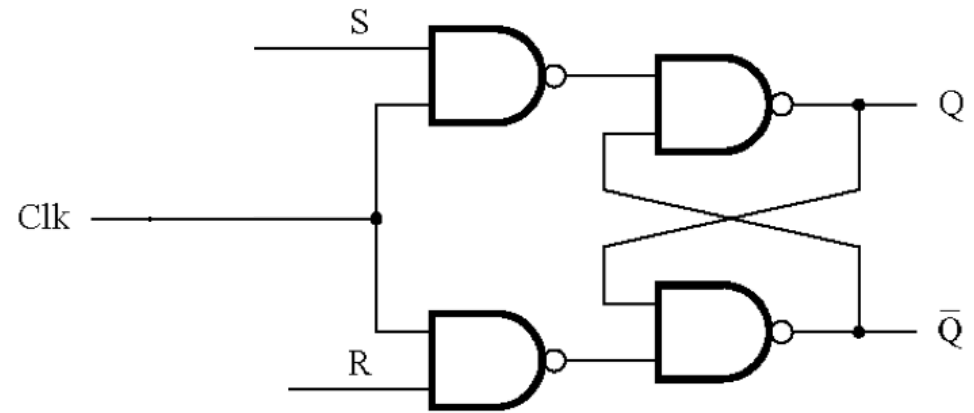
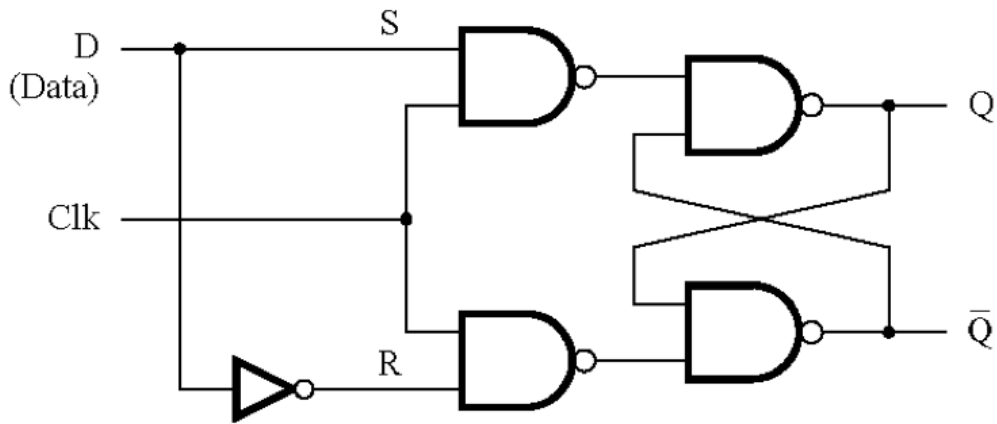
Constructing a D Flip-Flop



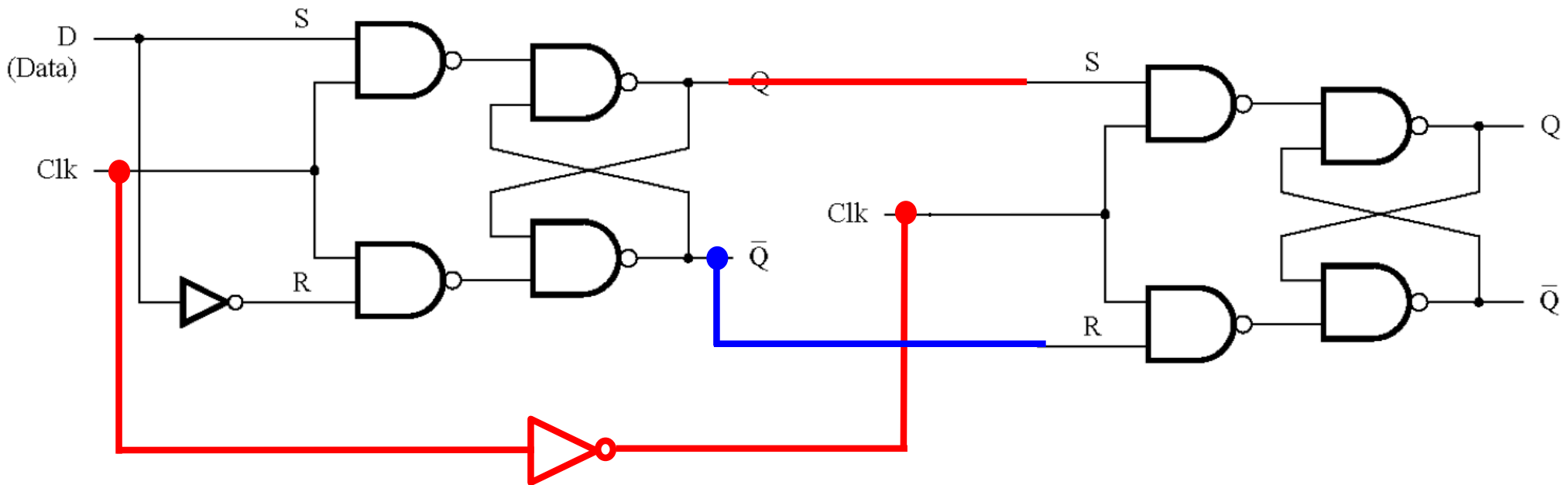
Constructing a D Flip-Flop



Constructing a D Flip-Flop (with one less NOT gate)

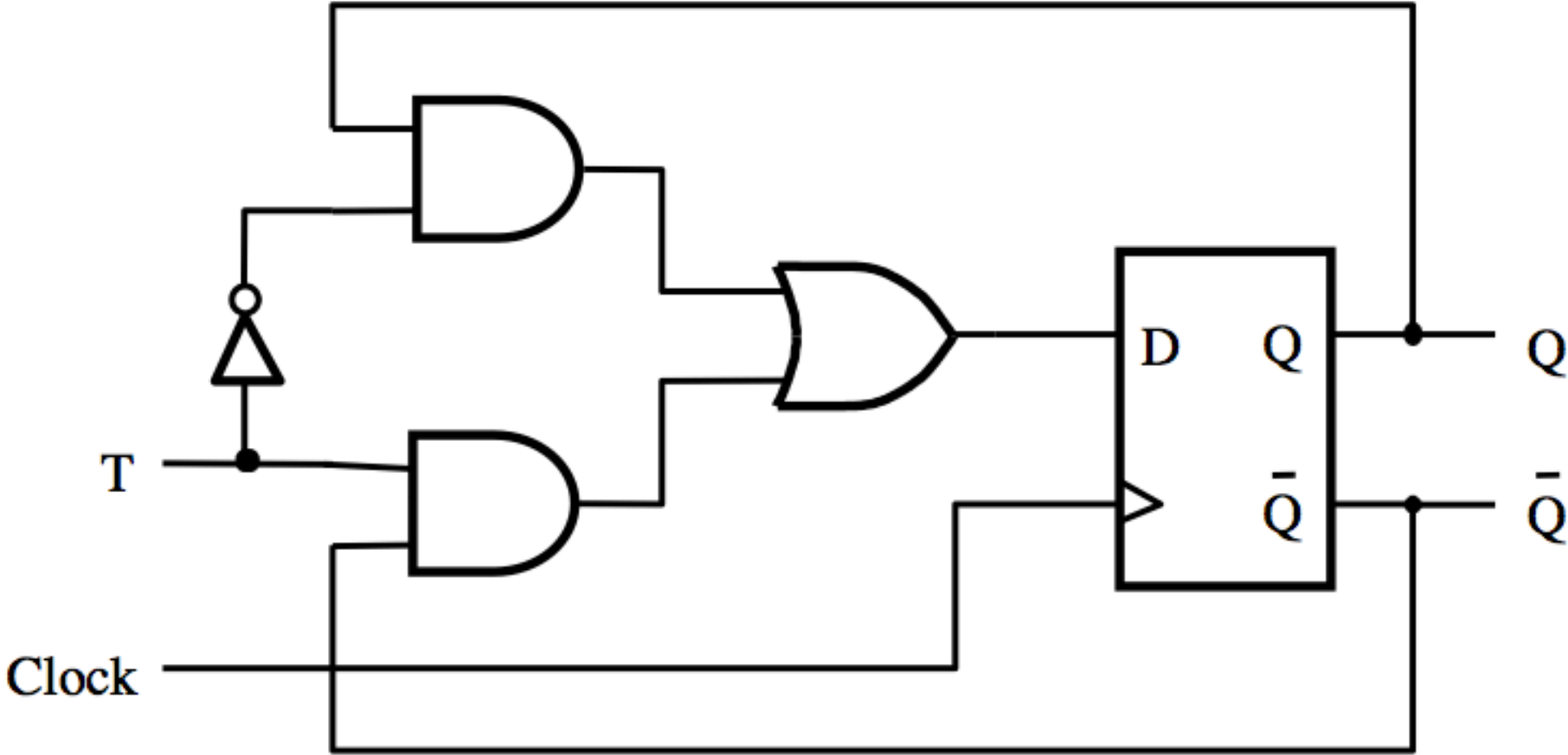


Constructing a D Flip-Flop (with one less NOT gate)



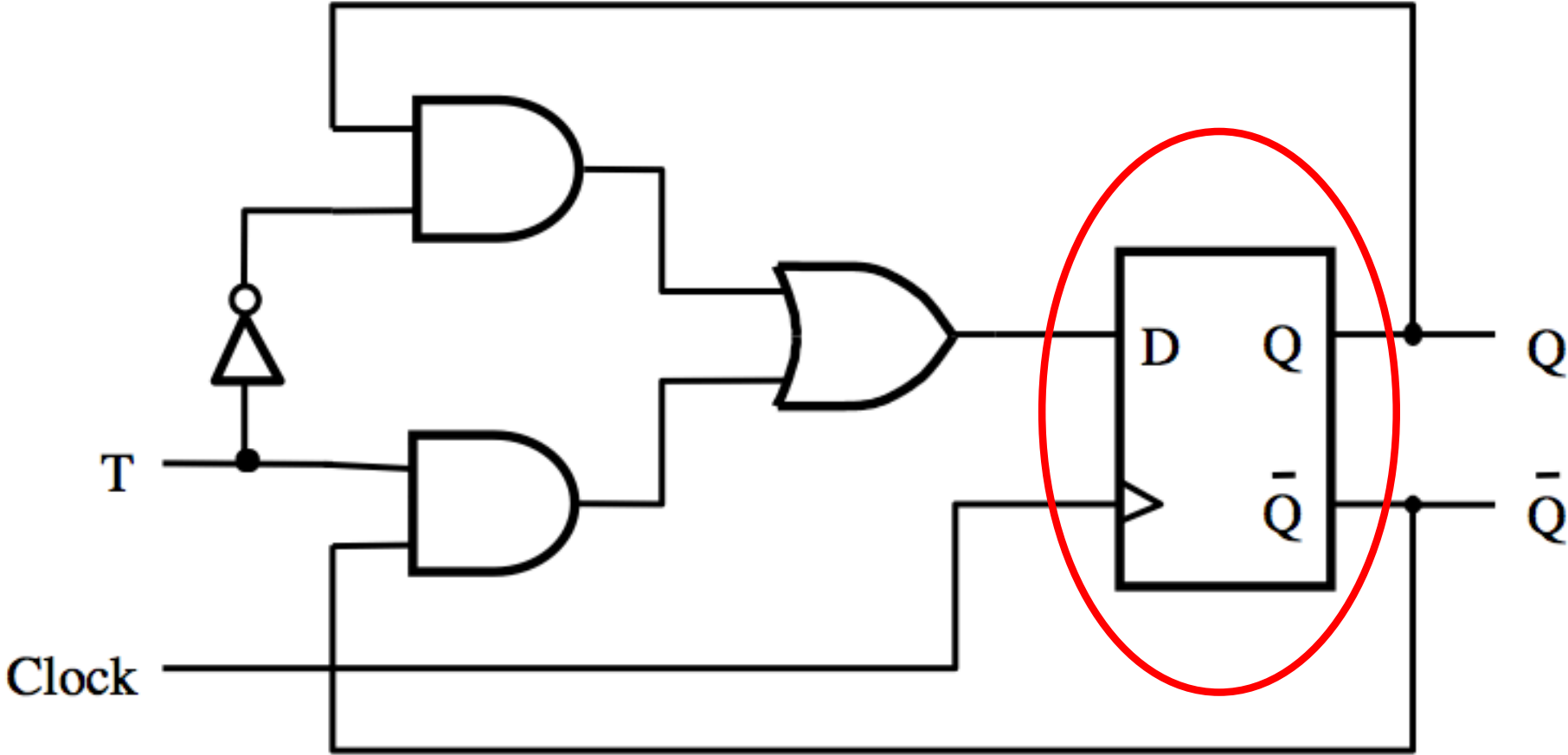
T Flip-Flop

T Flip-Flop



[Figure 5.15a from the textbook]

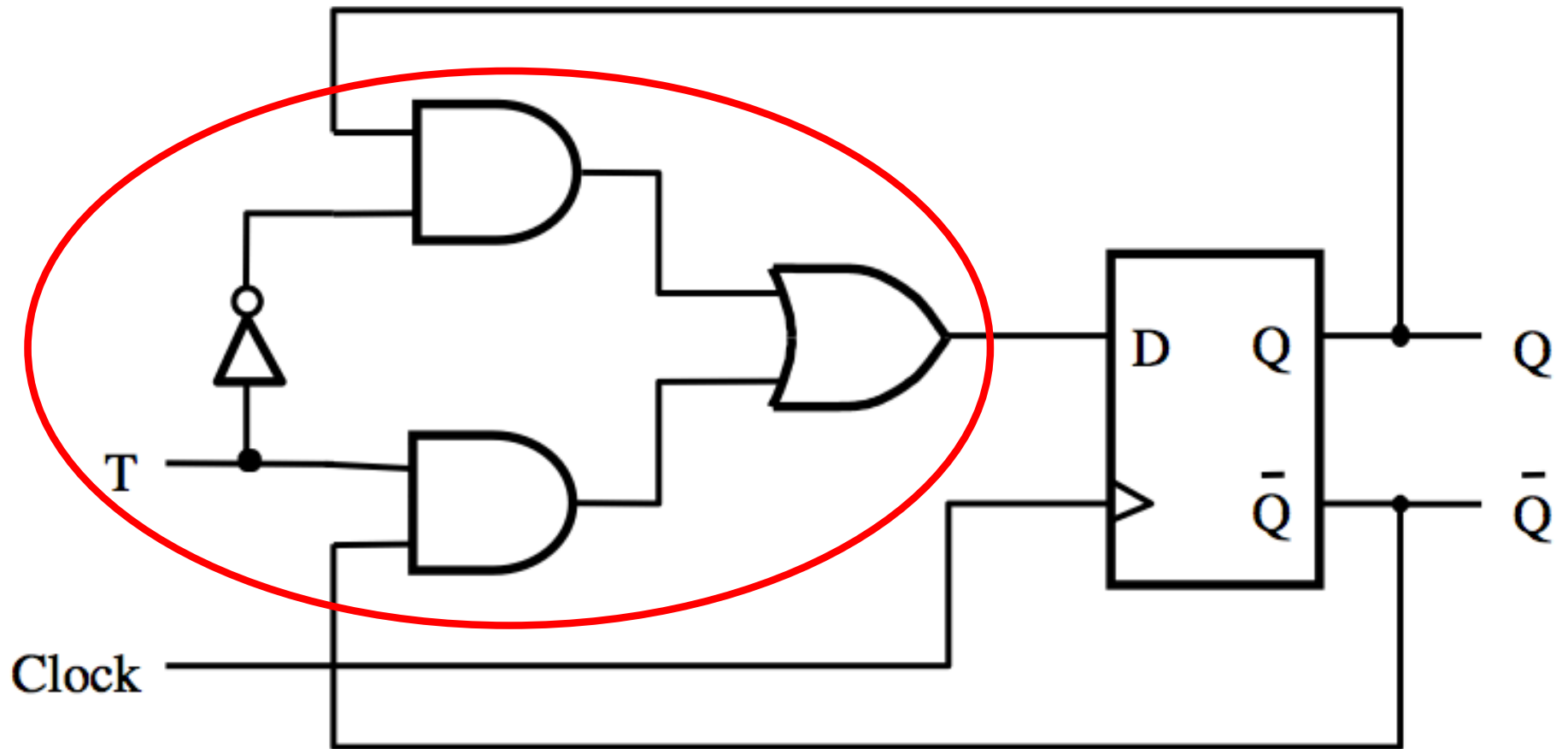
T Flip-Flop



Positive-edge-triggered
D Flip-Flop

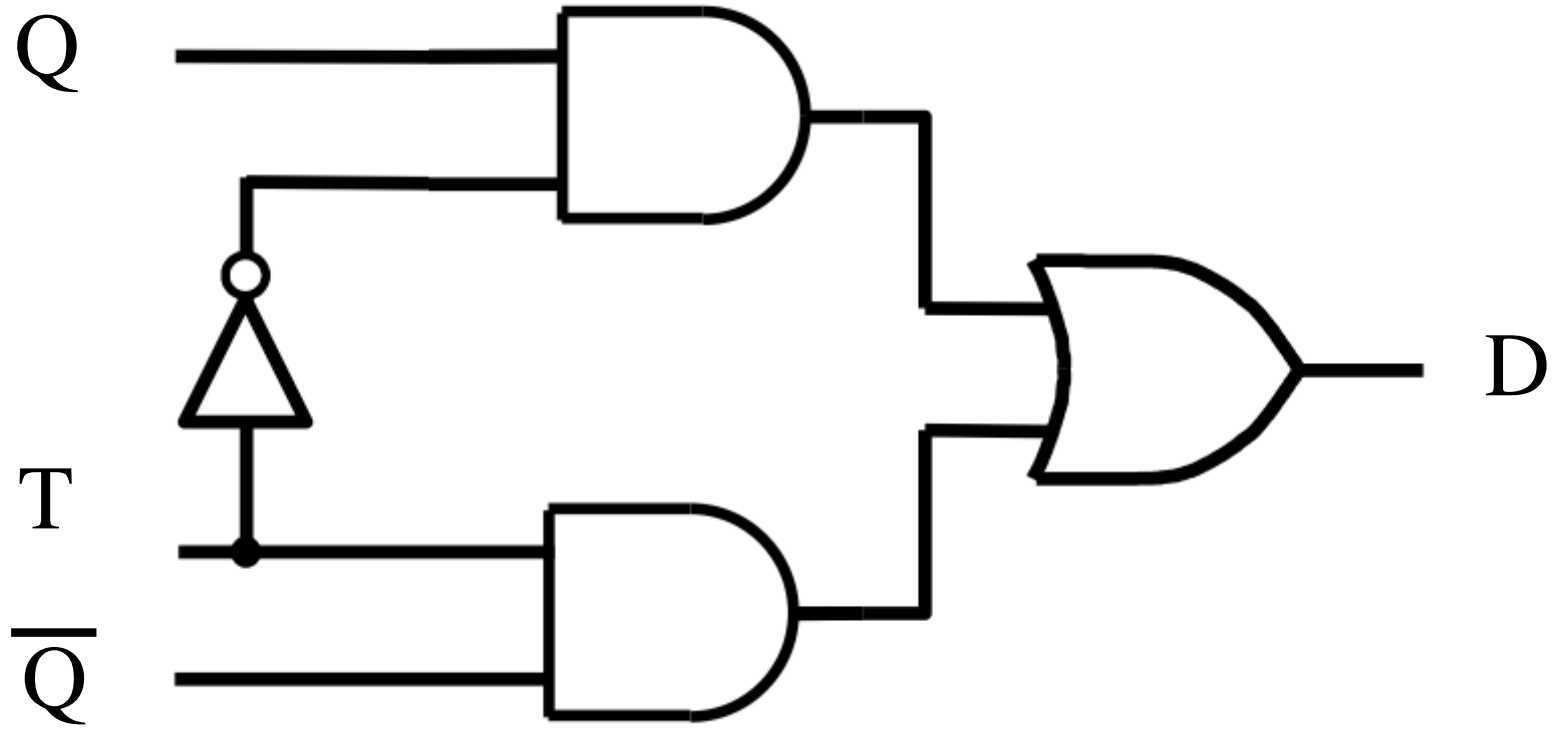
[Figure 5.15a from the textbook]

T Flip-Flop

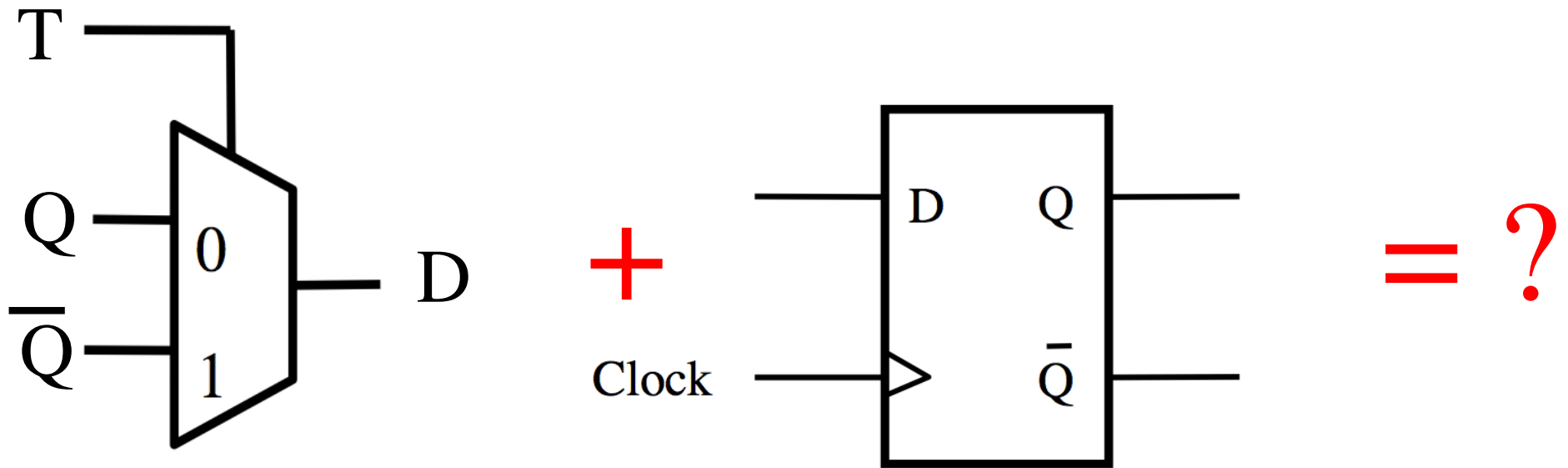


What is this?

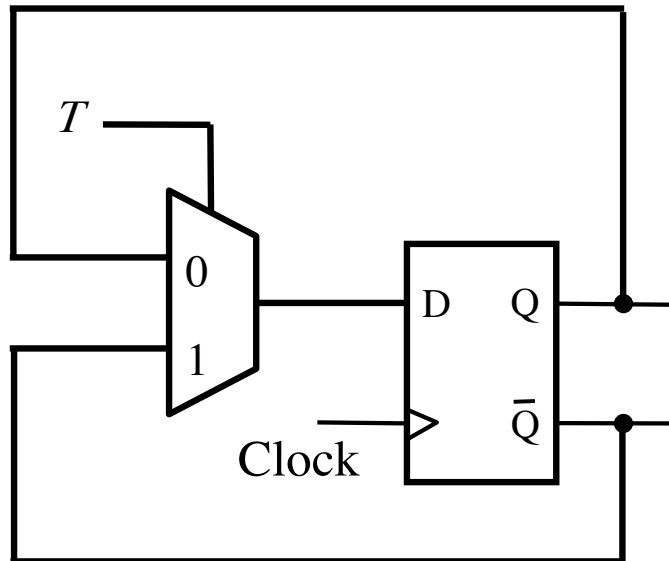
What is this?



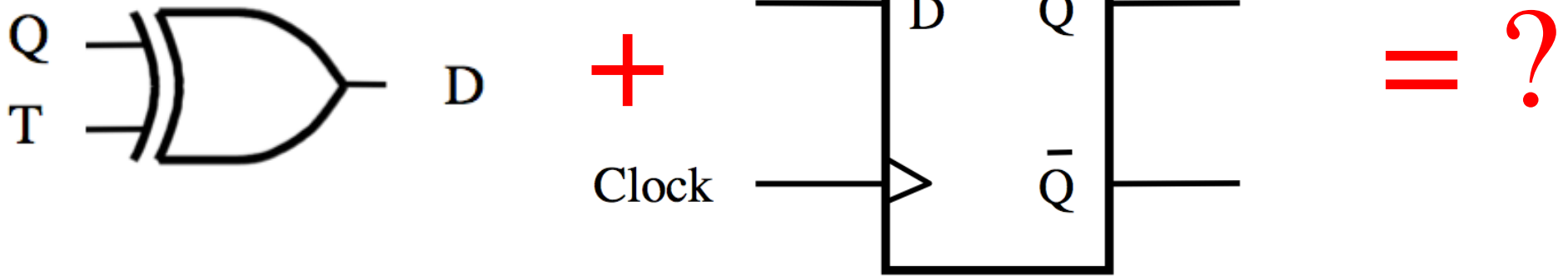
What is this?



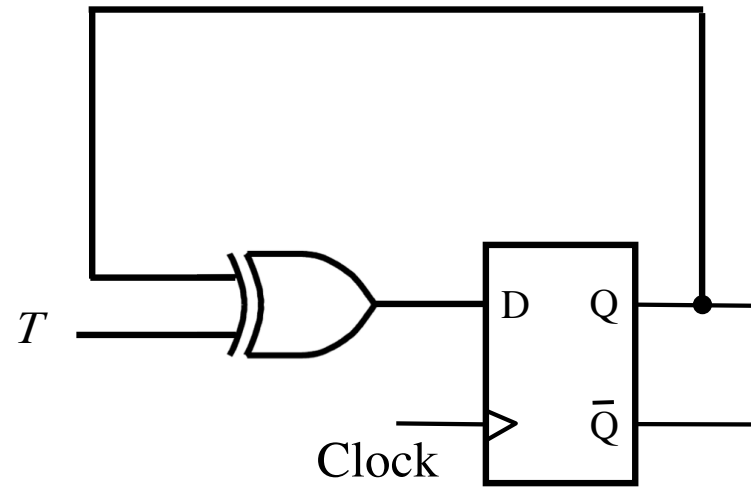
T Flip-Flop



What is this?

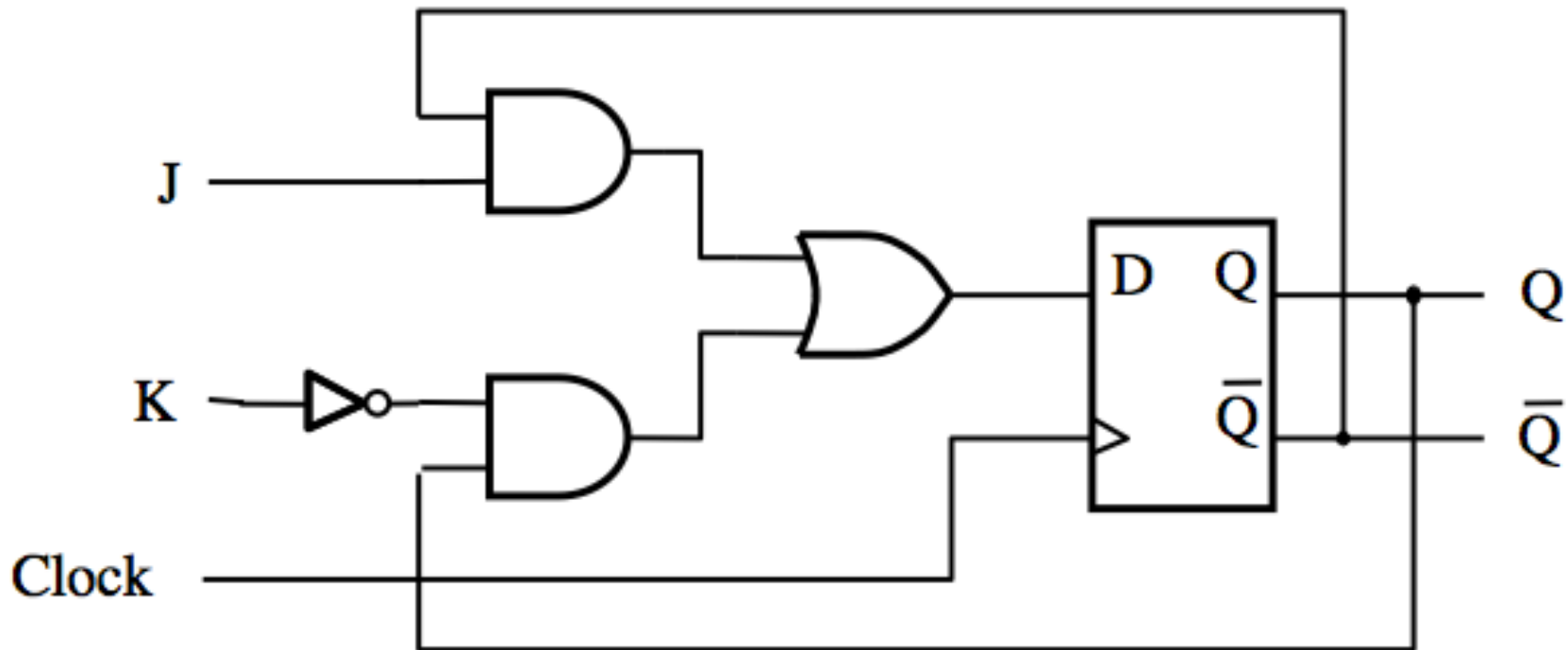


T Flip-Flop



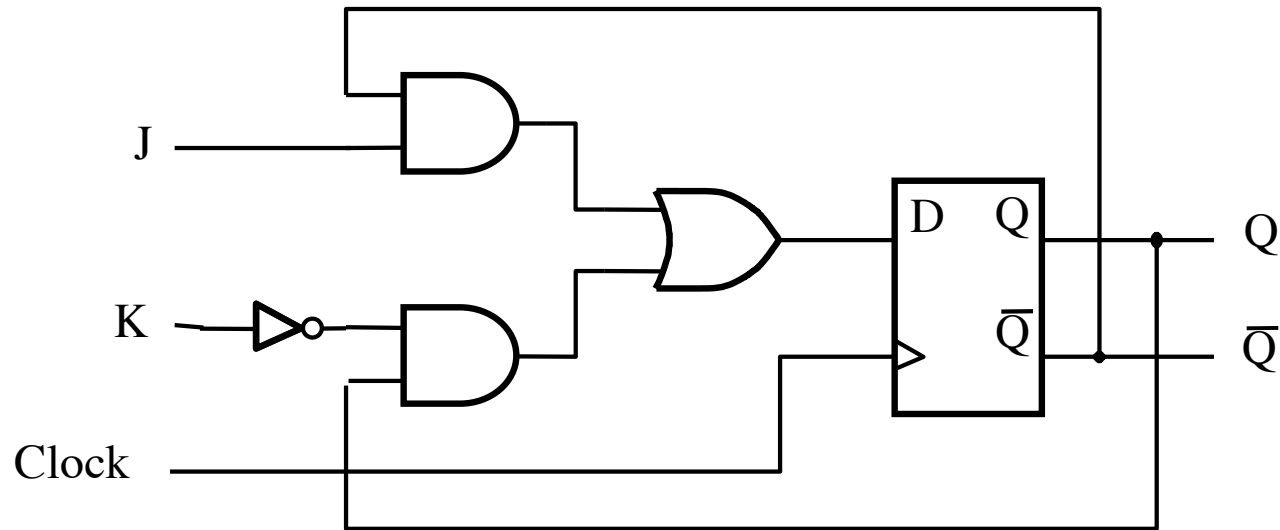
JK Flip-Flop

JK Flip-Flop



$$D = \overline{J}Q + \overline{K}Q$$

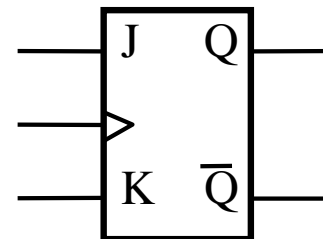
JK Flip-Flop



(a) Circuit

J	K	$Q(t+1)$
0	0	$Q(t)$
0	1	0
1	0	1
1	1	$\bar{Q}(t)$

(b) Truth table



(c) Graphical symbol

JK Flip-Flop (How it Works)

**A versatile circuit that can be used both as a
SR flip-flop and as a T flip flop**

If $J=0$ and $S =0$ it stays in the same state

Just like SR It can be set and reset

$J=S$ and $K=R$

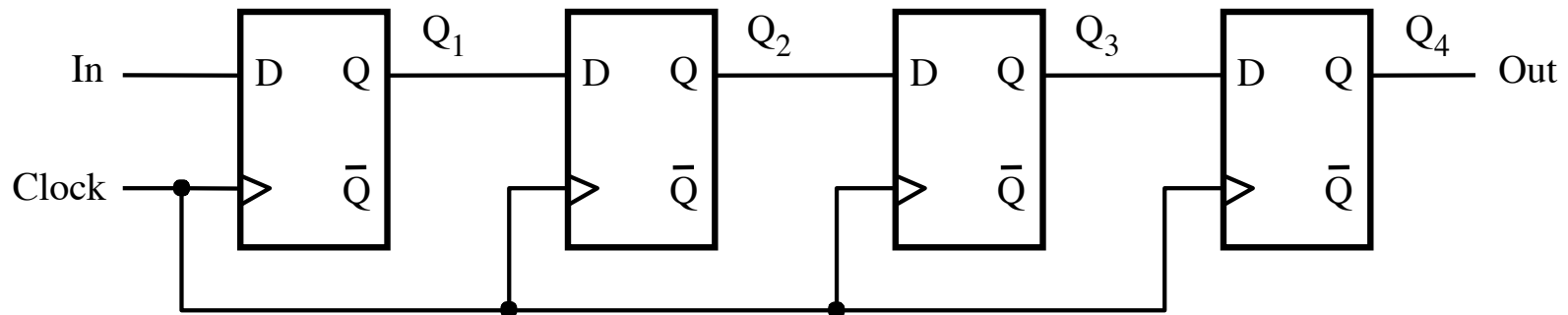
If $J=K$ then it behaves as a T flip-flop

Registers

Register (Definition)

An n-bit structure consisting of flip-flops

A simple shift register

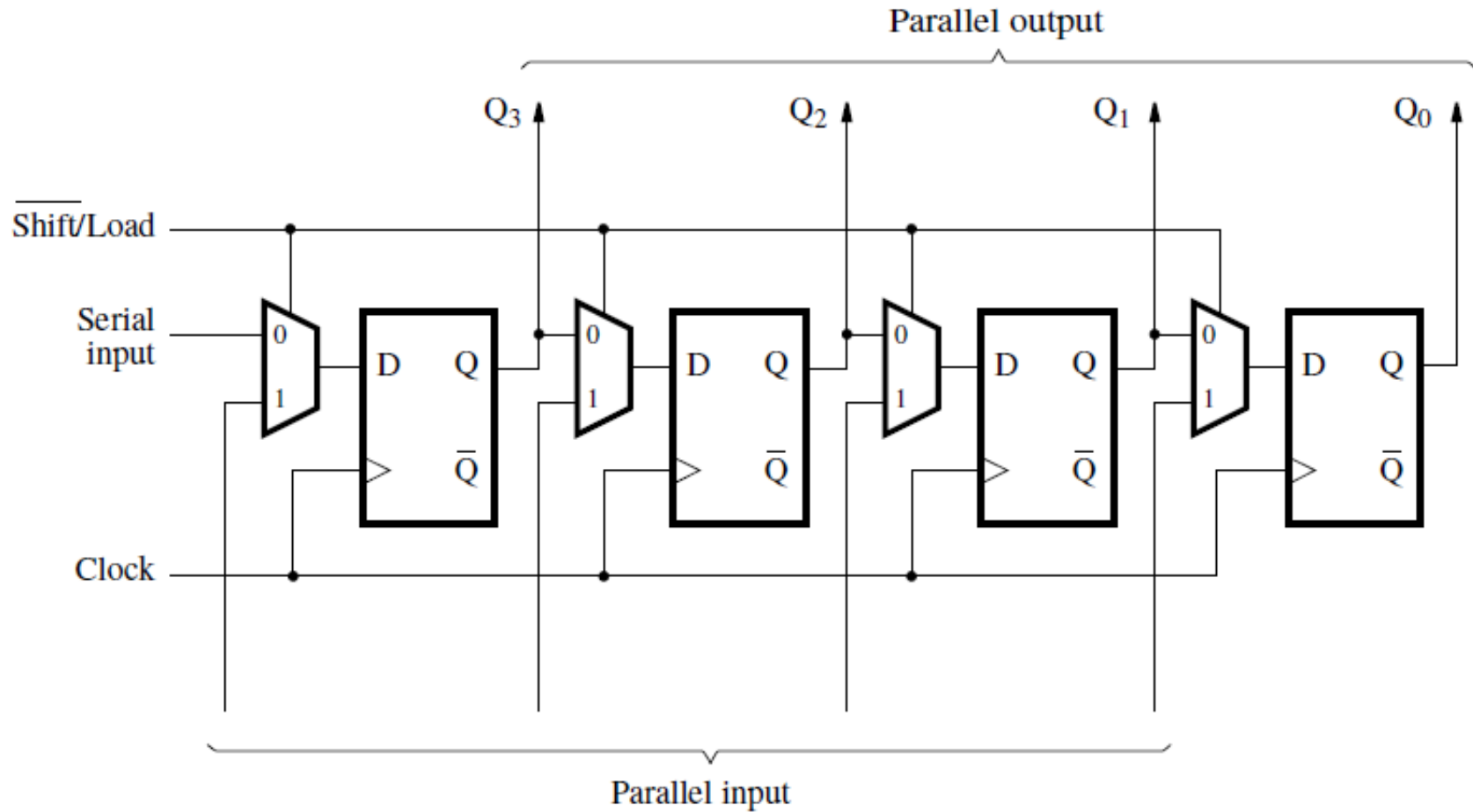


(a) Circuit

	In	Q_1	Q_2	Q_3	$Q_4 = \text{Out}$
t_0	1	0	0	0	0
t_1	0	1	0	0	0
t_2	1	0	1	0	0
t_3	1	1	0	1	0
t_4	1	1	1	0	1
t_5	0	1	1	1	0
t_6	0	0	1	1	1
t_7	0	0	0	1	1

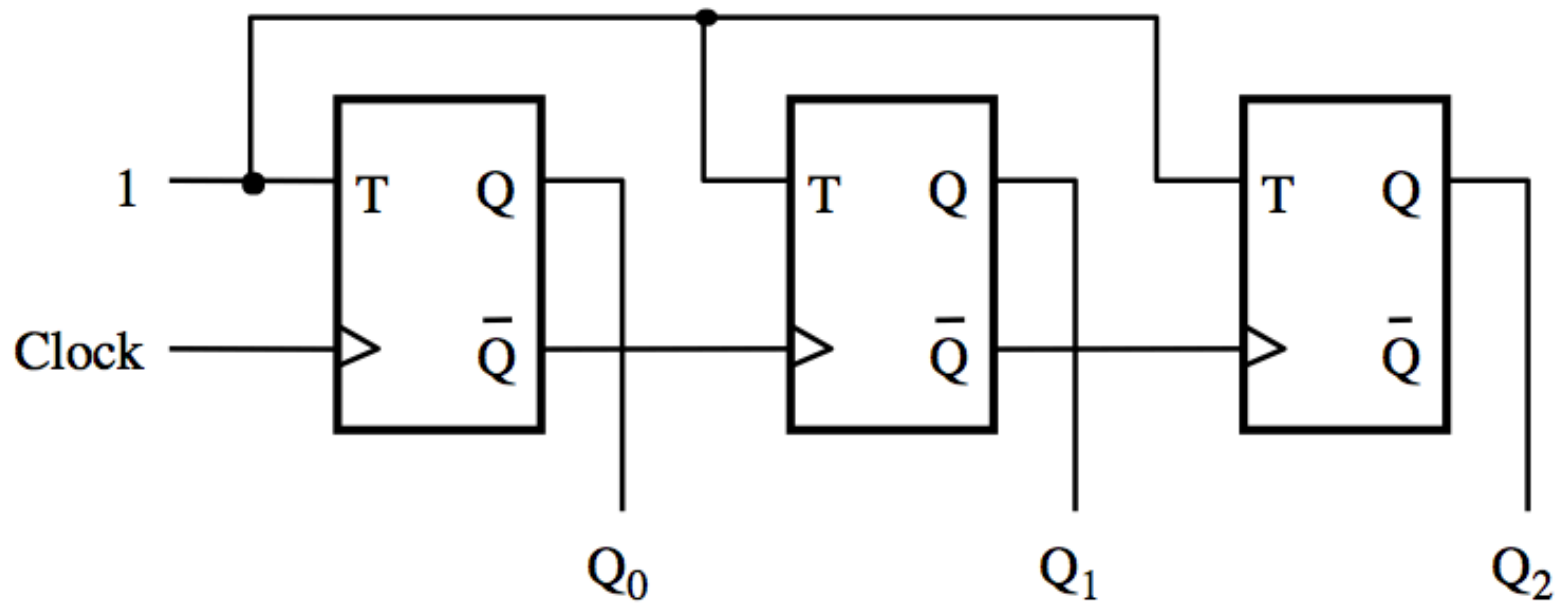
(b) A sample sequence

Parallel-access shift register

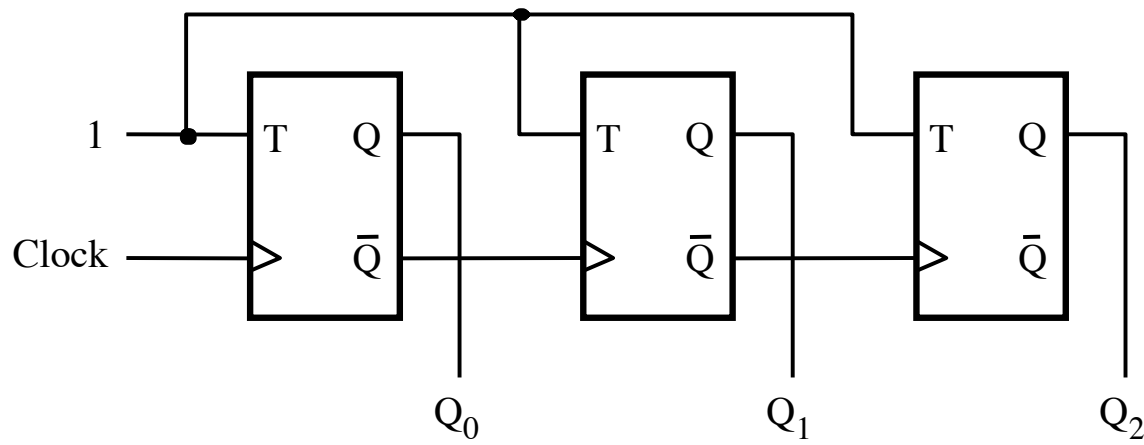


Counters

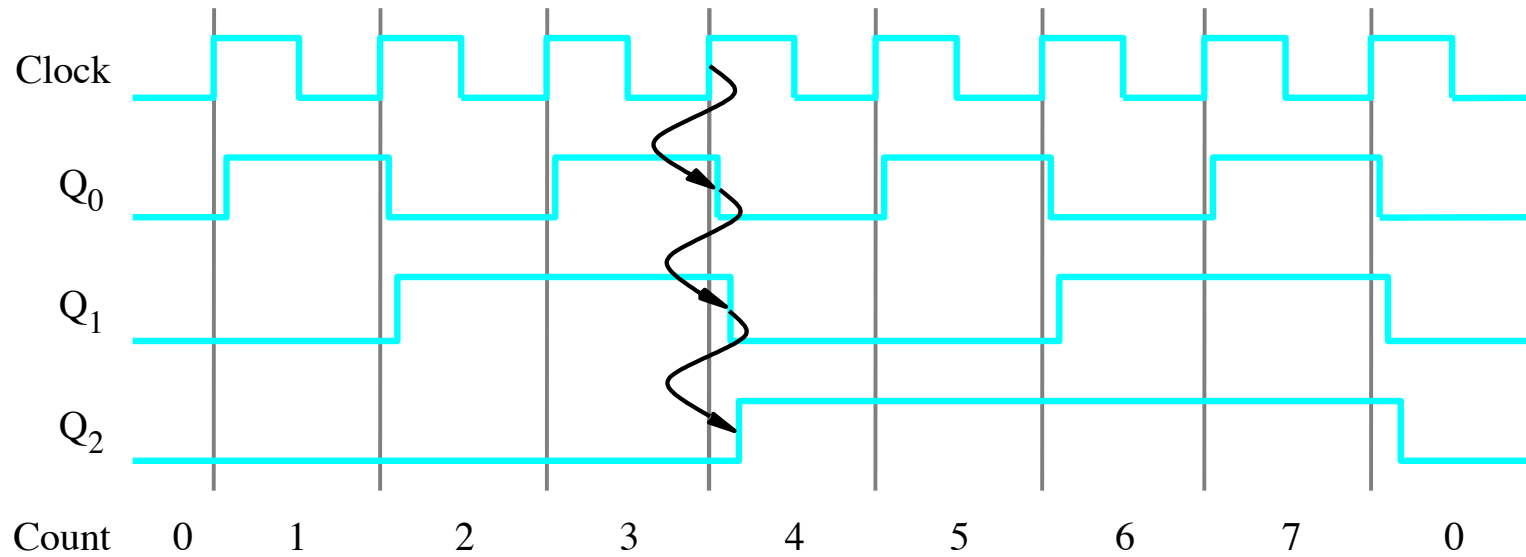
A three-bit up-counter



A three-bit up-counter

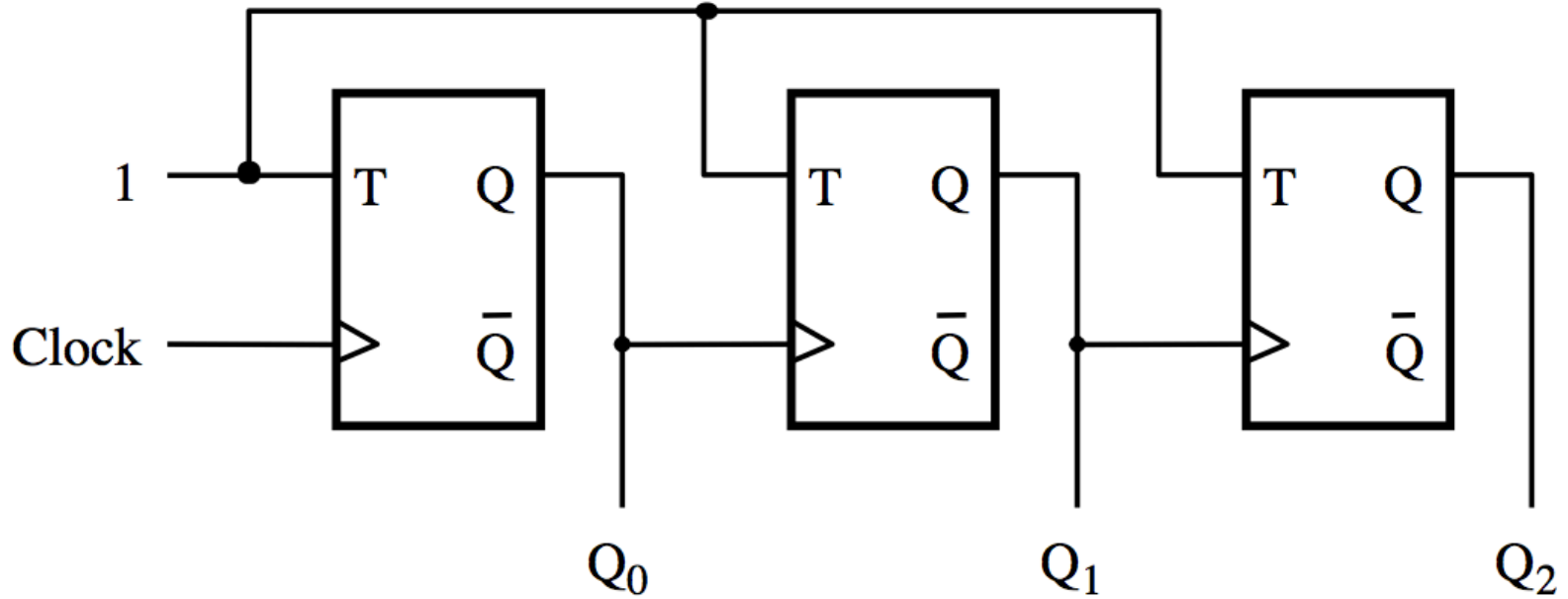


(a) Circuit

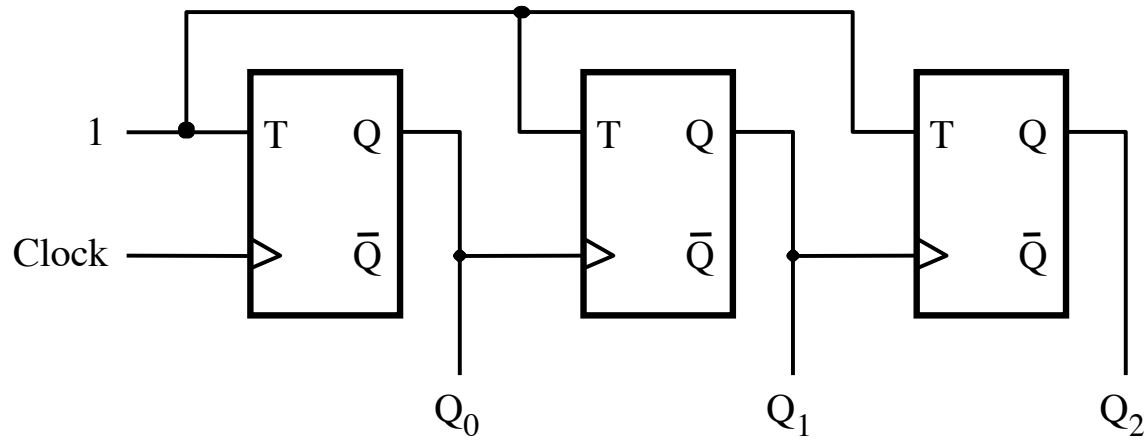


(b) Timing diagram

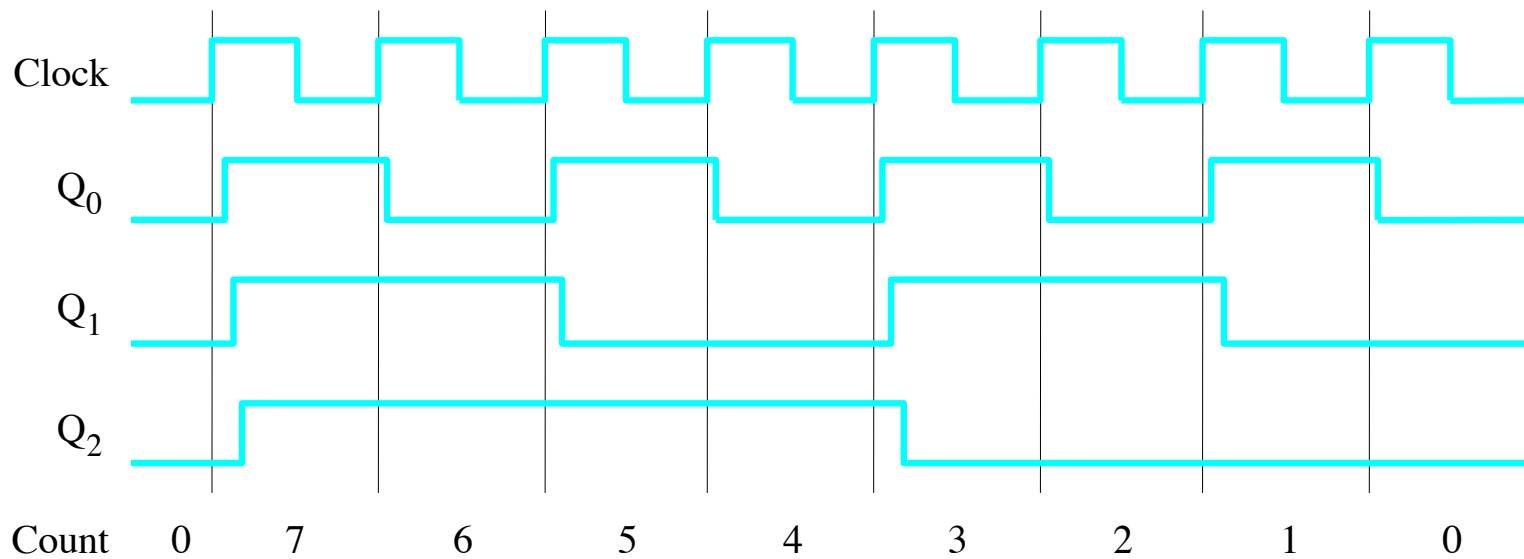
A three-bit down-counter



A three-bit down-counter



(a) Circuit

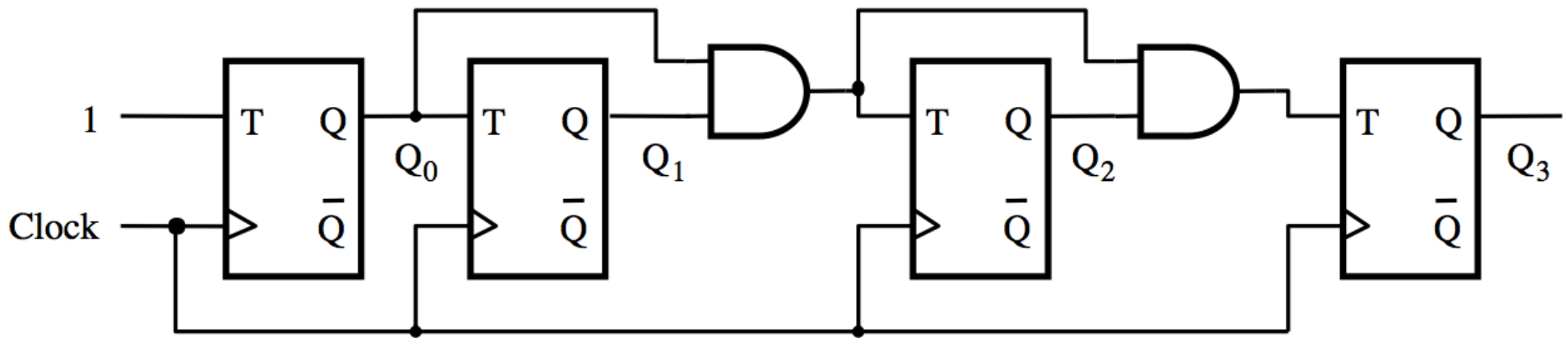


(b) Timing diagram

[Figure 5.20 from the textbook]

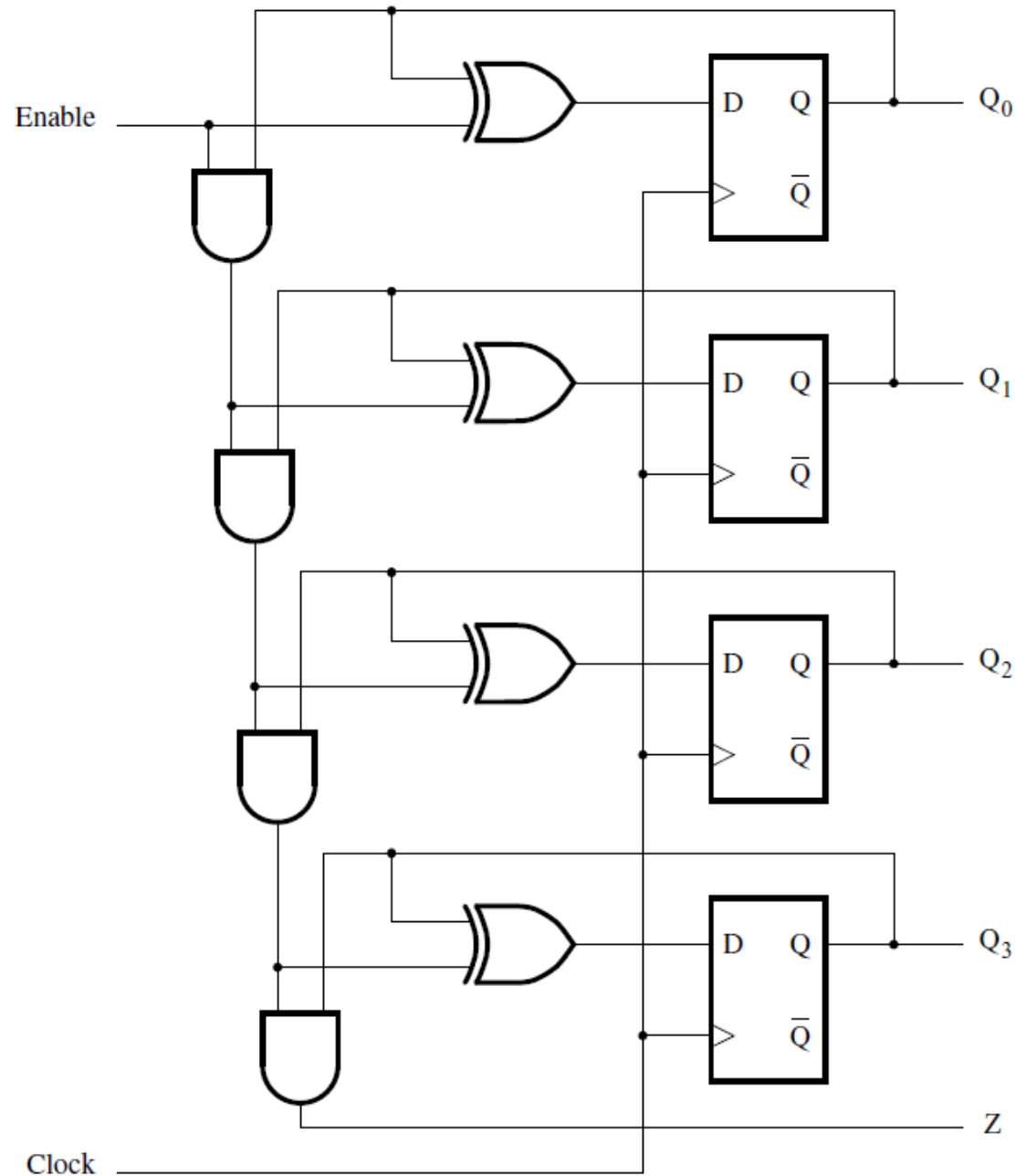
Synchronous Counters

A four-bit synchronous up-counter



Synchronous Counter with D Flip-Flops

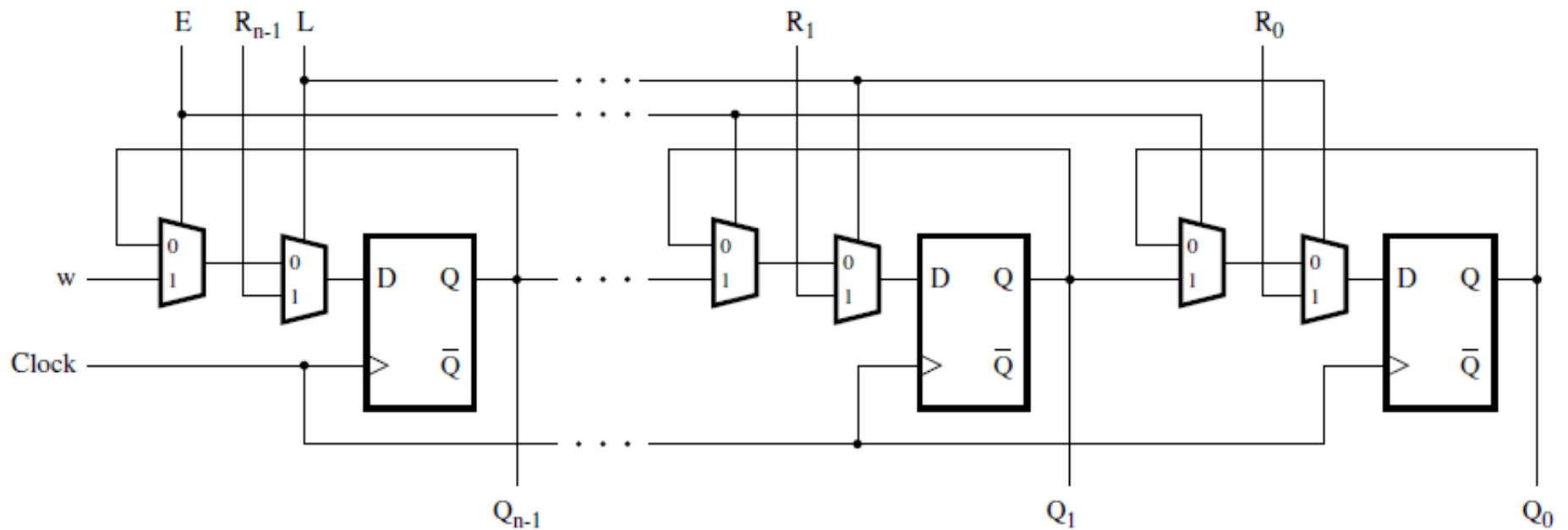
A four-bit counter with D flip-flops



[Figure 5.23 from the textbook]

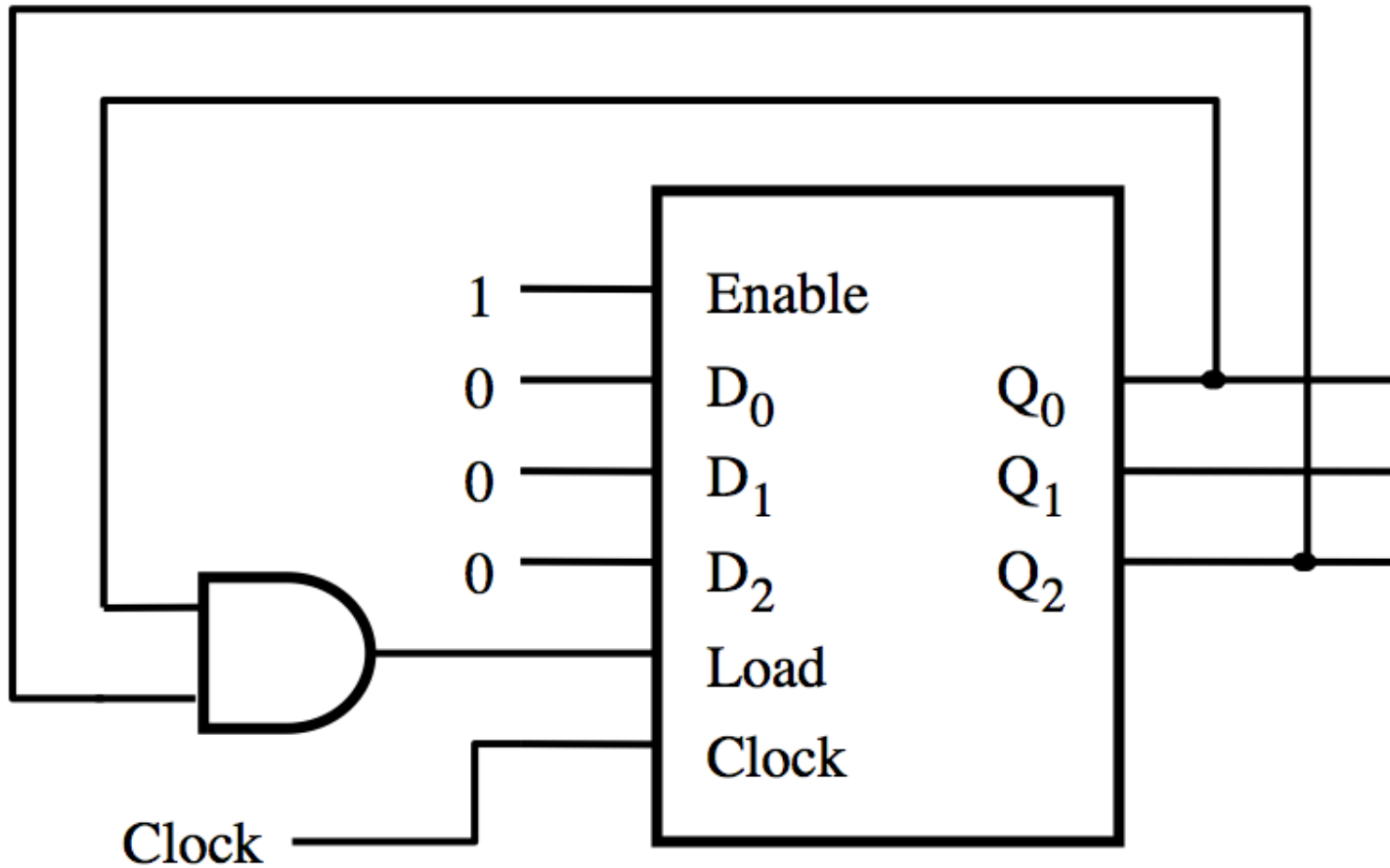
Counters with Parallel Load

A shift register with parallel load and enable control inputs

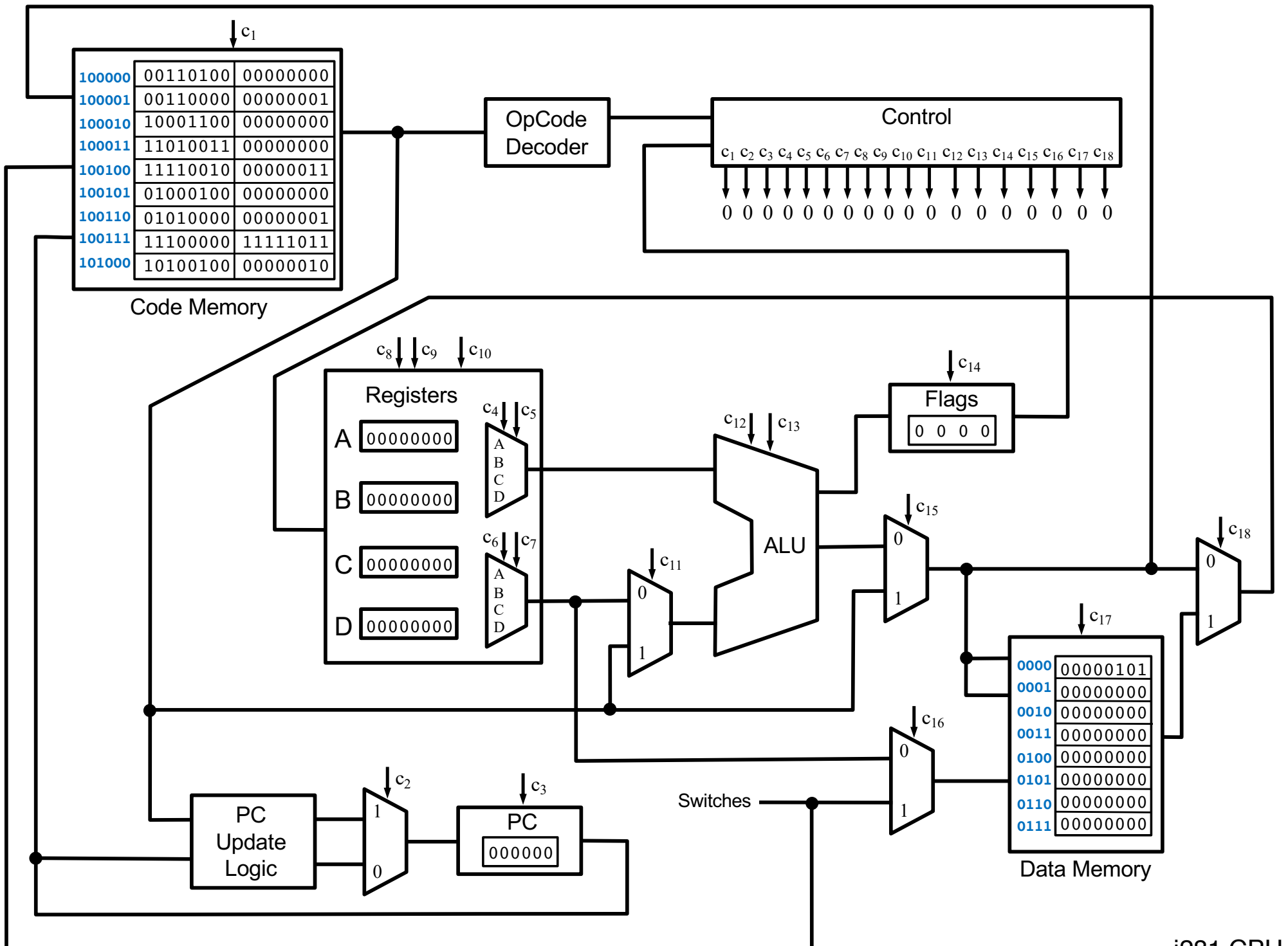


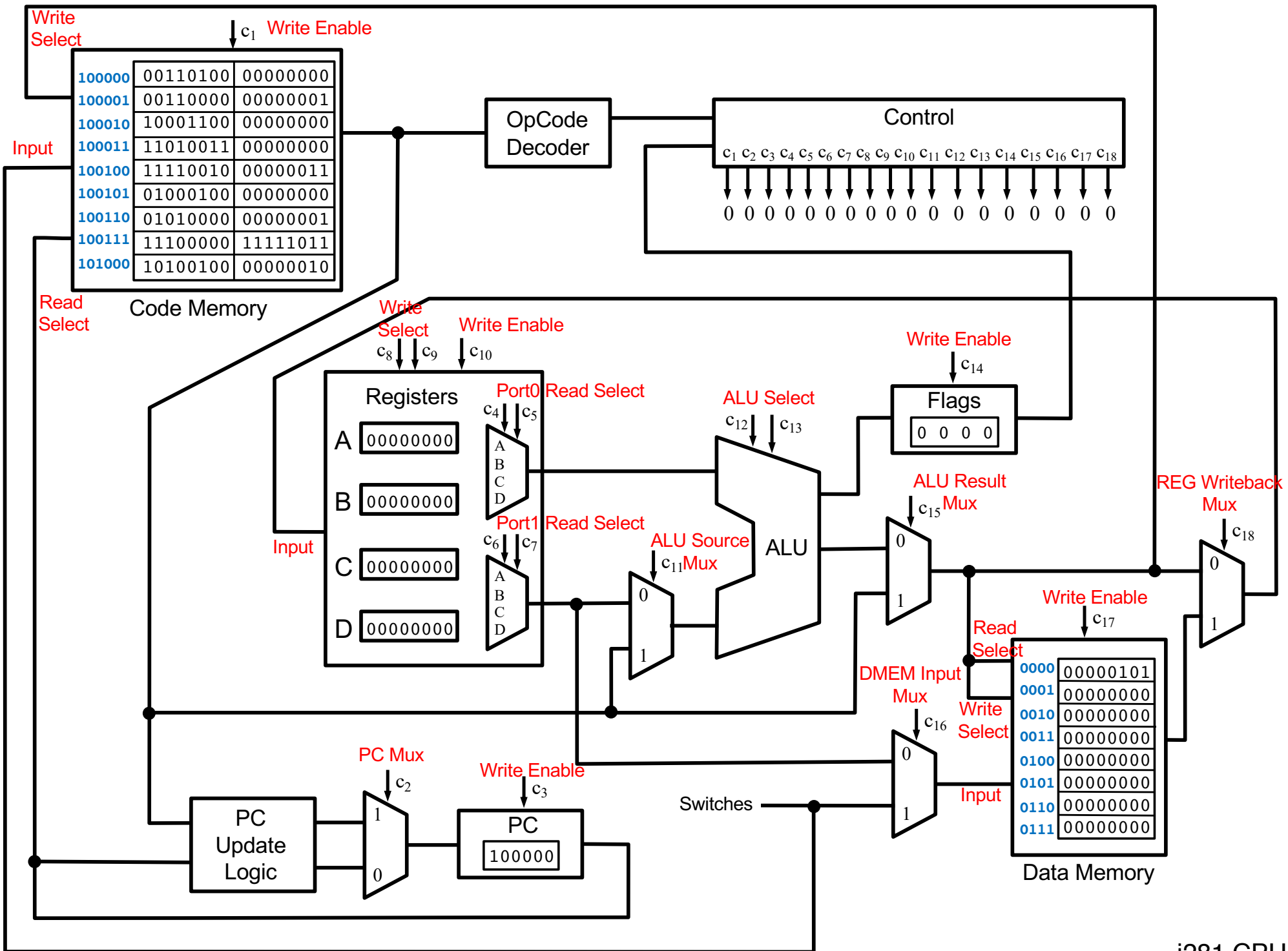
[Figure 5.59 from the textbook]

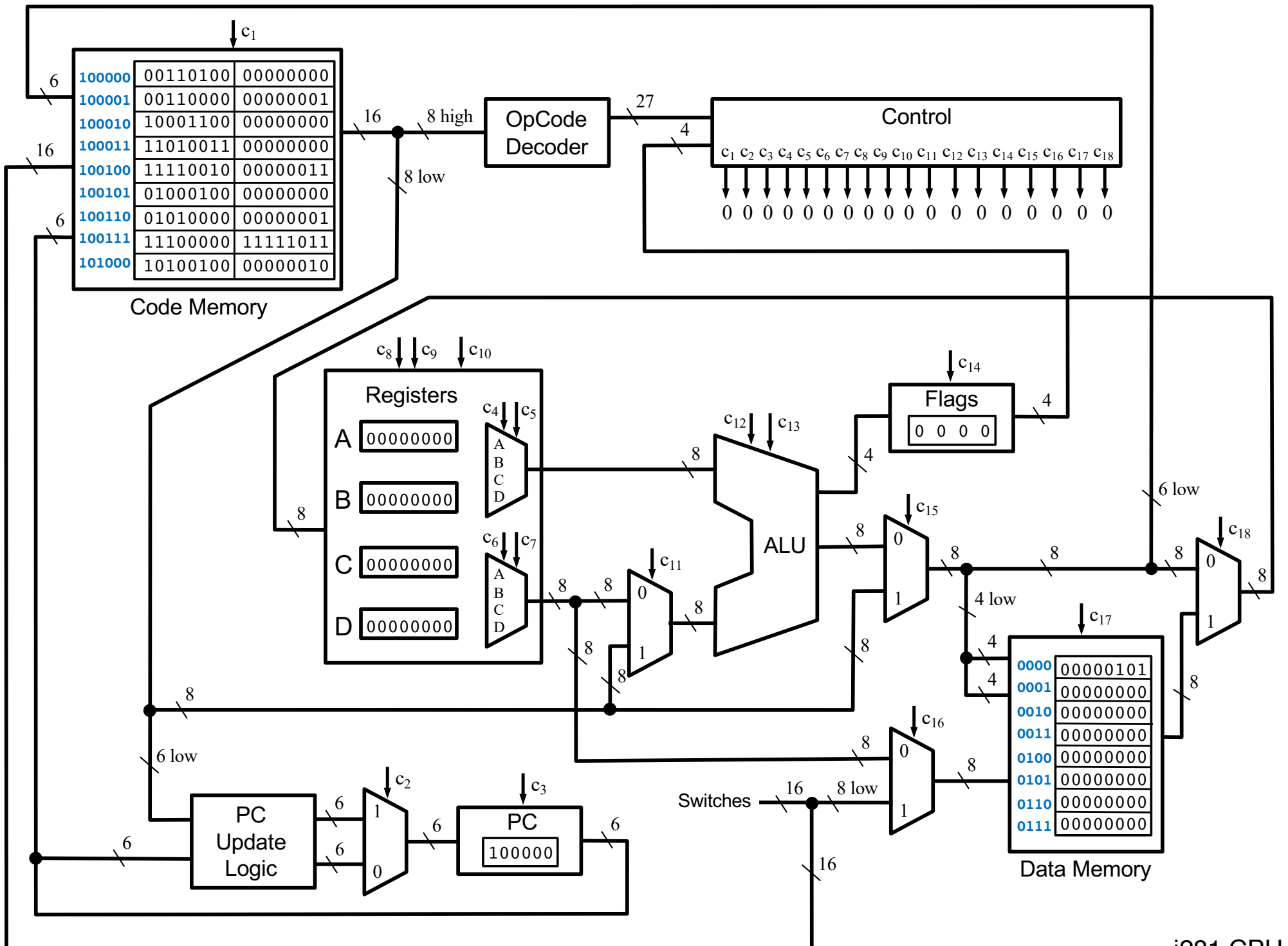
What does this circuit do?



i281 CPU







Add the numbers from 1 to 5

```
// C Version
// using a for loop

int main()
{
    int N=5;
    int i, sum;

    sum=0;
    for(i=1; i<=N; i++) {
        sum+=i;
    }

    // printf("%d\n", sum);
}
```

```
; Assembly Version

.data
N        BYTE    5
i        BYTE    ?
sum      BYTE    ?

.code

        LOADI   B, 0        ; sum=0
        LOADI   A, 1        ; i=1
        LOAD    D, [N]      ; register_D=N
Loop:   CMP     A, D        ; i<=N ?
        BRG     End        ; exit if i>N
Add:    ADD     B, A        ; sum+=i
        ADDI   A, 1        ; i++
        JUMP   Loop        ; next iteration
End:    STORE   [sum], B    ; write B to sum
```

Mapping Assembly to Machine Code

.data

N **BYTE** **5**
i **BYTE** **?**
sum **BYTE** **?**

Data Memory:

00000101
00000000
00000000

.code

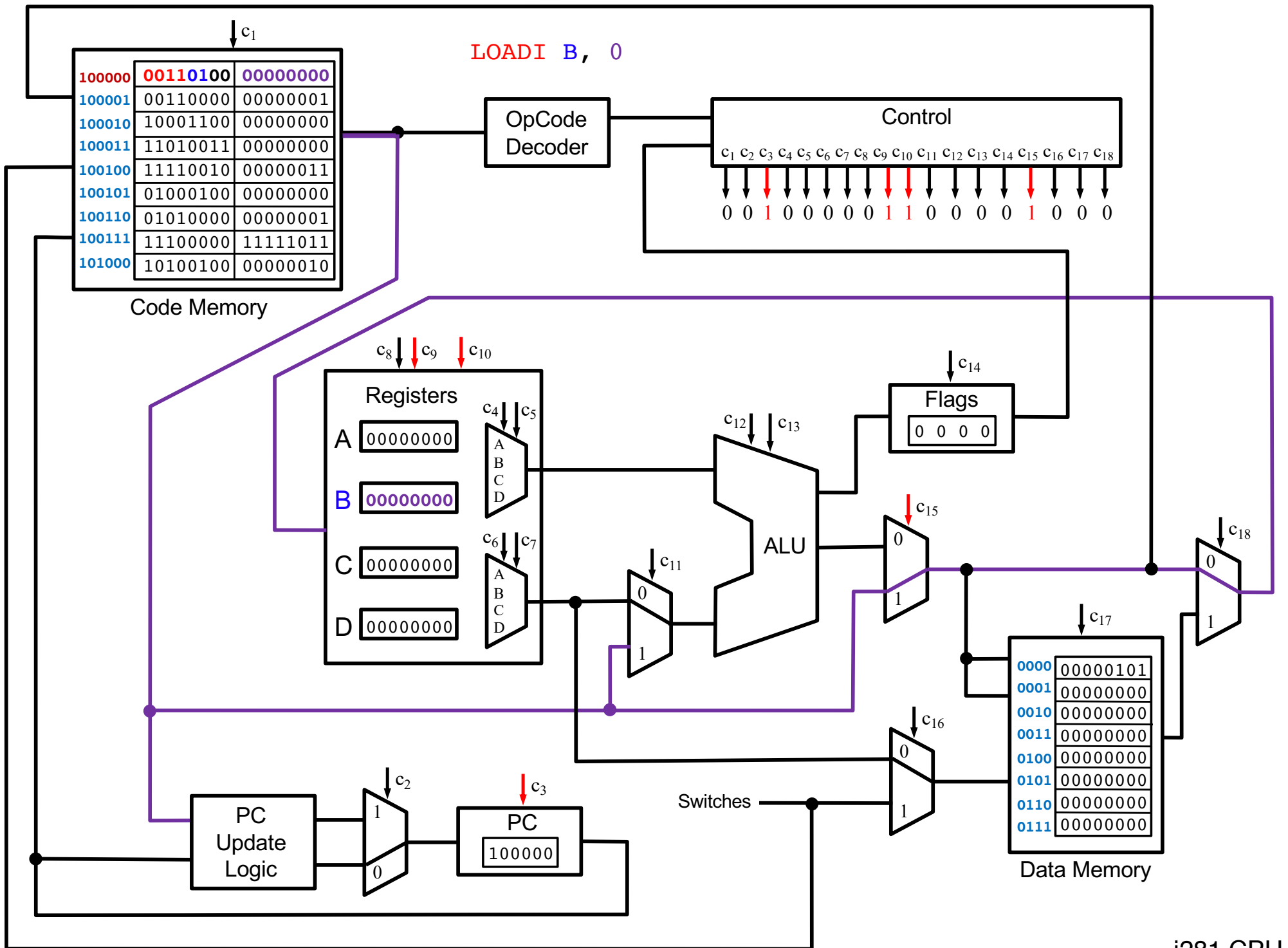
LOADI **B, 0**
 LOADI **A, 1**
 LOAD **D, [N]**
Loop: **CMP** **A, D**
 BRG **End**
Add: **ADD** **B, A**
 ADDI **A, 1**
 JUMP **Loop**
End: **STORE** **[sum], B**

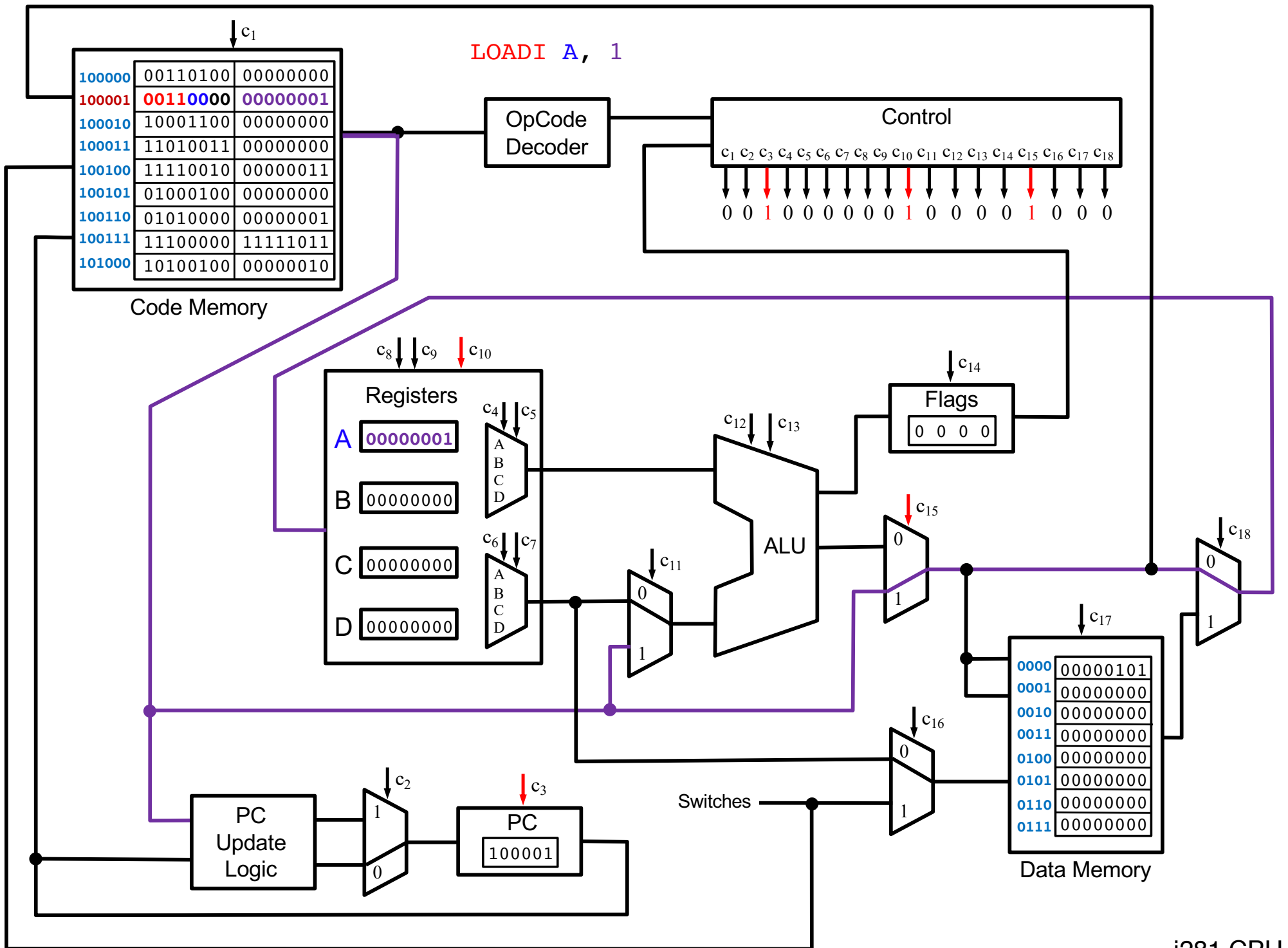
Code Memory:

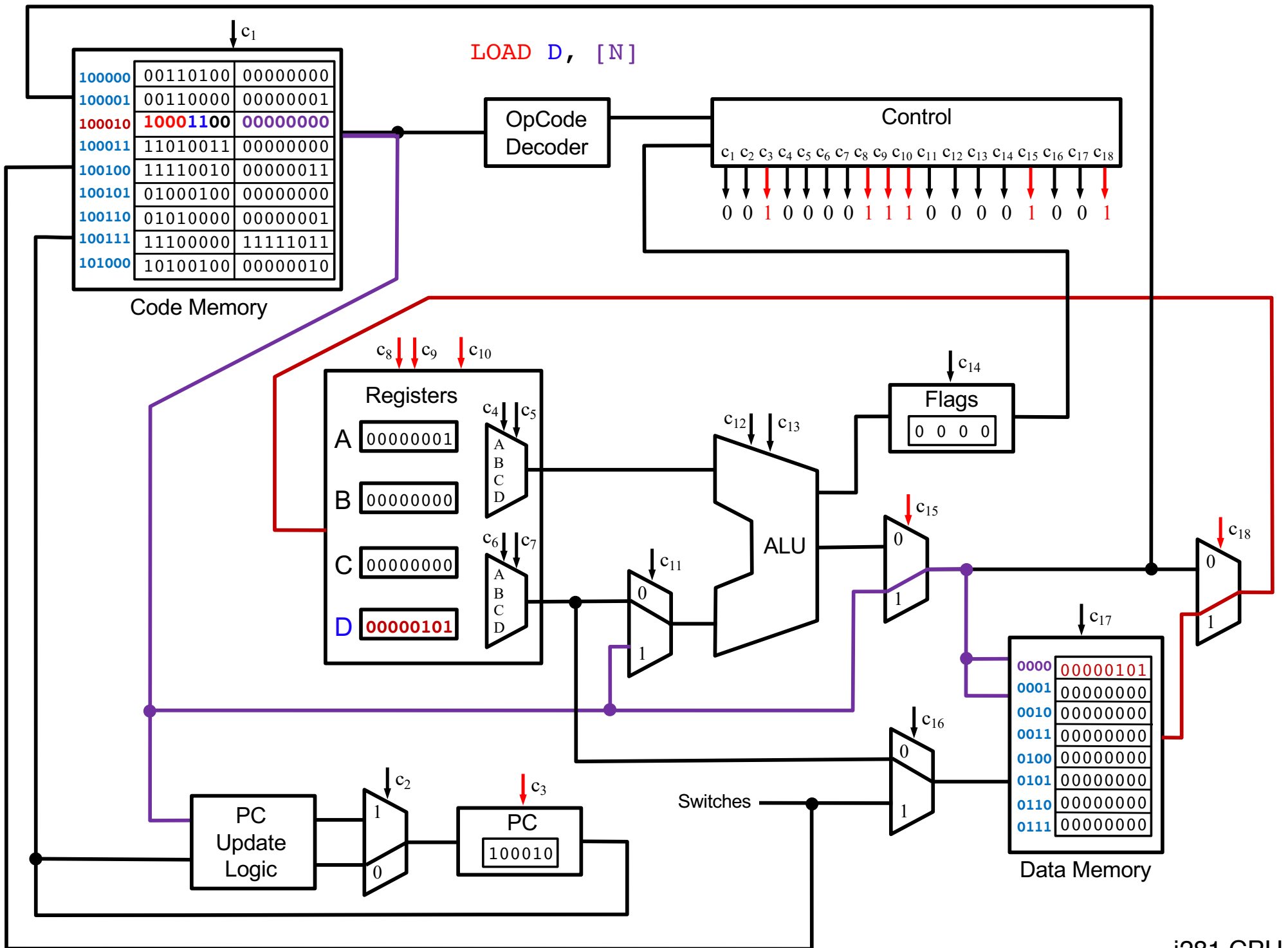
0011010000000000
0011000000000001
1000110000000000
1101001100000000
1111001000000011
0100010000000000
0101000000000001
1110000011111011
1010010000000010

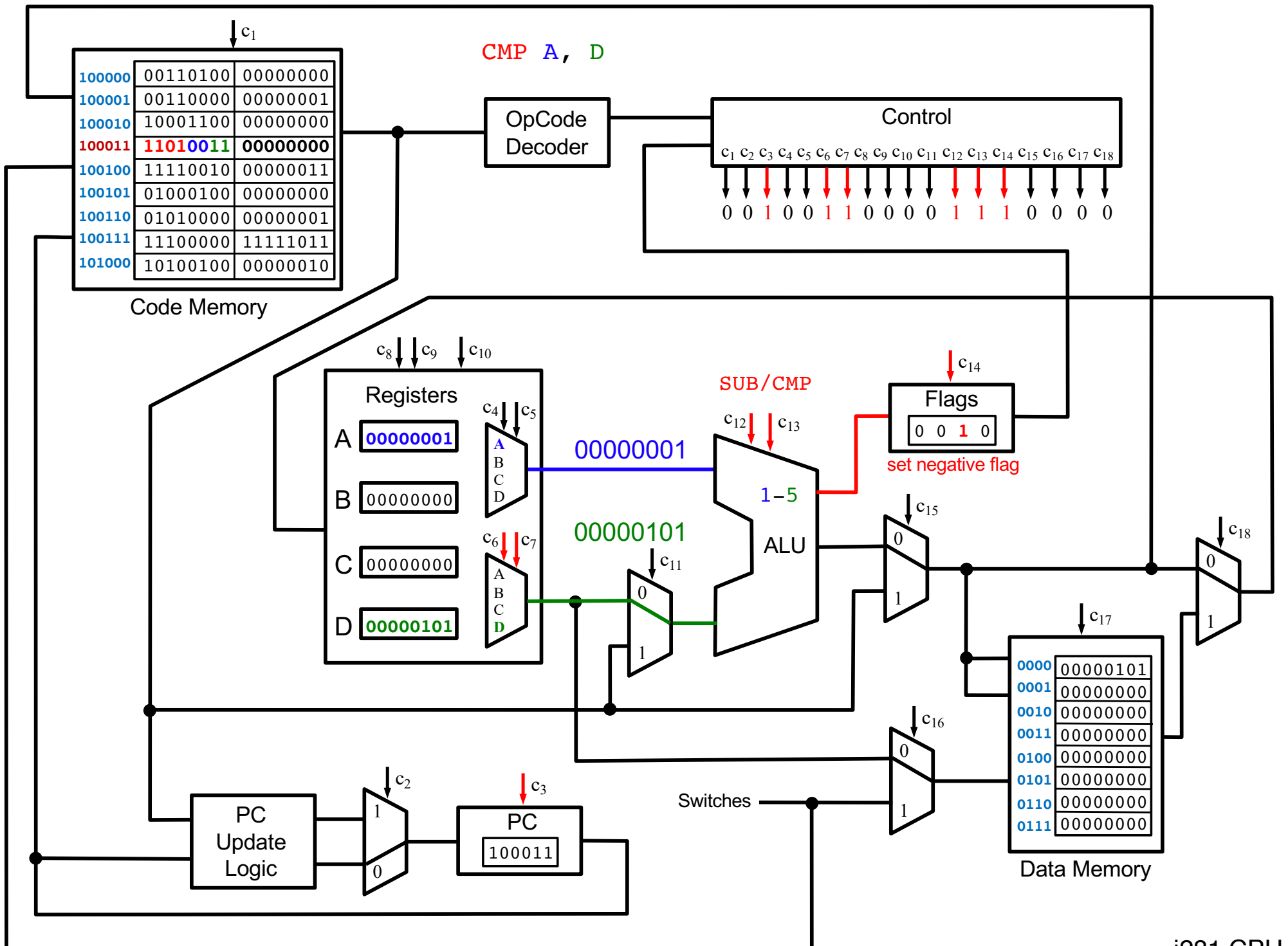
Assembly Language

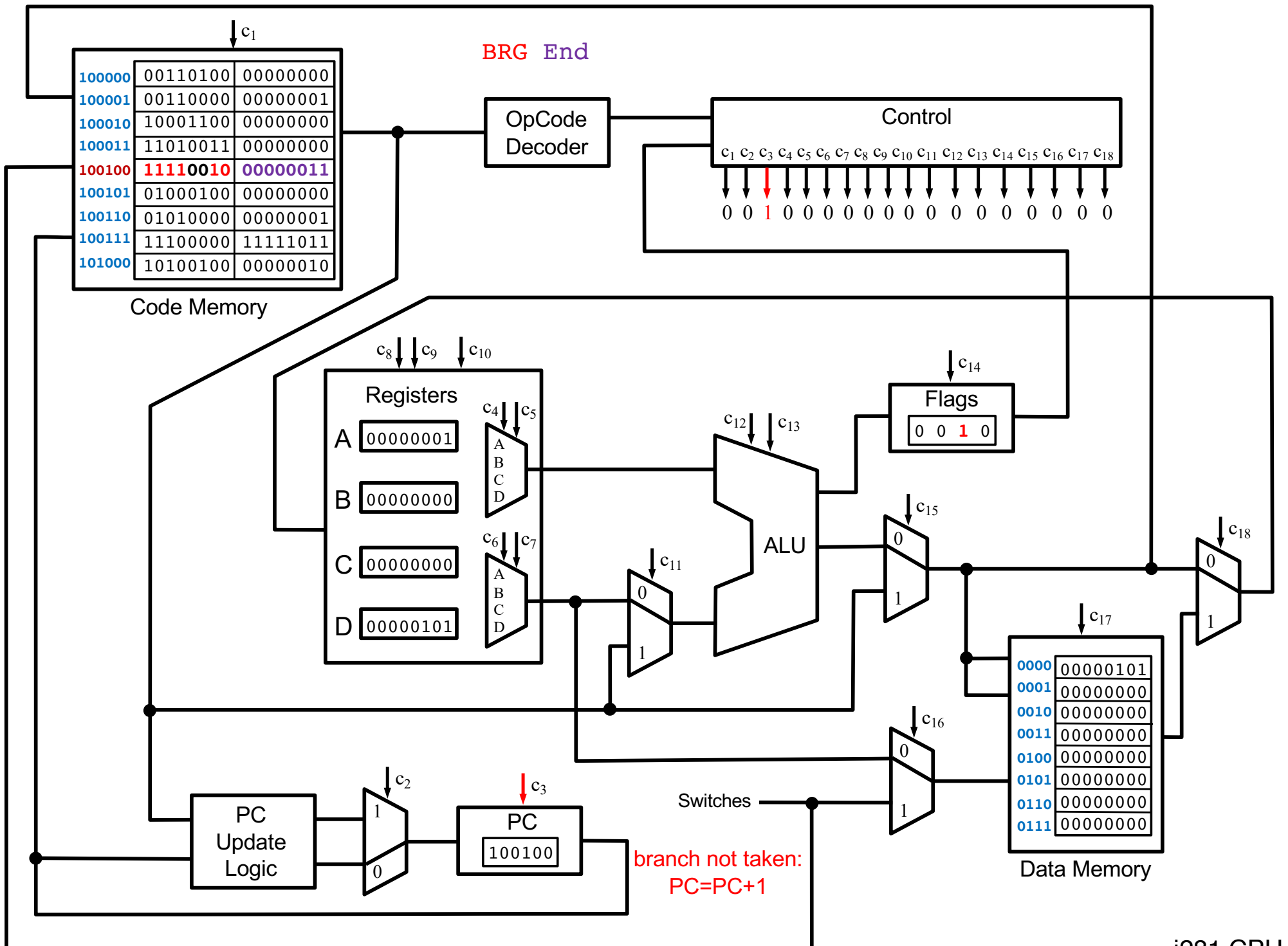
Machine Language

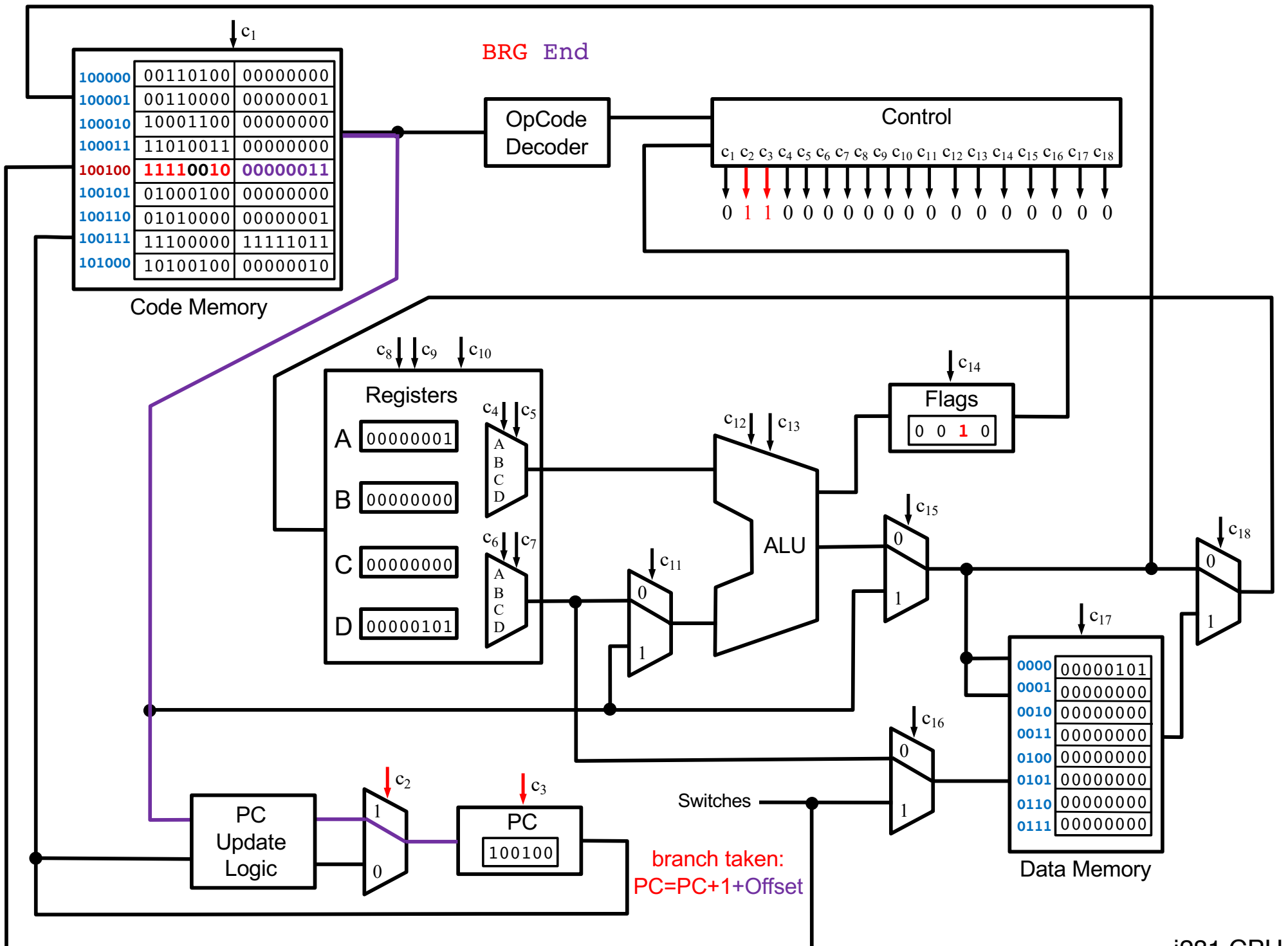












Bubble Sort

C Version

```
int array[] = {7, 3, 2, 1, 6, 4, 5, 8};
int last = 7; // last valid index in the array
int temp;
int i, j;

int main()
{
    for (i = 0; i < last; i++)
        for (j = 0; j < last-i; j++)
            if (array[j] > array[j+1]){
                temp = array[j];
                array[j] = array[j+1];
                array[j+1] = temp;
            }

    //for(i = 0; i < N; i++){
    //    printf("%d, ", array[i]);
    //}
}
```

Assembly Version

```
.data
array    BYTE 7, 3, 2, 1, 6, 4, 5, 8
last     BYTE 7
temp     BYTE ?

.code

Outer:   LOADI   A, 0                ; i = 0;
        LOAD    D, [last]           ; Load last into D
        LOADI   B, 0                ; j = 0;
        CMP     A, D                ; i < last
        BRGE    End                 ; If i >= last break out of the outer loop
Inner:   LOAD    D, [last]           ; Re-Load last into D (this register is shared)
        SUB     D, A                ; D = D - A (i.e., D = last - i)
        CMP     B, D                ; j < last - i
        BRGE    Iinc                ; If j >= last-i branch to Iinc
If:      LOADF   C, [array+B]        ; C = array[j]
        LOADF   D, [array+B+1]      ; D = array[j+1] (compiler adds 1 to addr. of array)
        CMP     D, C                ; if array[j+1] < array[j] (switched direction)
        BRGE    Jinc
Swap:    STOREF [array+B], D
        STOREF [array+B+1], C
Jinc:    ADDI    B, 1                ; j++
        JUMP    Inner
Iinc:    ADDI    A, 1                ; i++
        JUMP    Outer
End:     NOOP                        ; Do nothing

; Register allocation:
; A: i
; B: j
; C: array[j]
; D: last, array[j+1]

; Notes: i and j are optimized away. They exist only in registers, not in the main memory.
```


Machine Code Version

```
.data
array  BYTE 7, 3, 2, 1, 6, 4, 5, 8
last   BYTE 7
temp   BYTE ?
```

```
.code

        LOADI  A, 0
Outer:  LOAD   D, [last]
        LOADI  B, 0
        CMP    A, D
        BRGE   End
Inner:  LOAD   D, [last]
        SUB    D, A
        CMP    B, D
        BRGE   Iinc
If:     LOADF  C, [array+B]
        LOADF  D, [array+B+1]
        CMP    D, C
        BRGE   Jinc
Swap:   STOREF [array+B], D
        STOREF [array+B+1], C
Jinc:   ADDI   B, 1
        JUMP   Inner
Iinc:   ADDI   A, 1
        JUMP   Outer
End:    NOOP
```

Machine Code Version

		Data Memory:
.data		
array	BYTE 7, 3, 2, 1, 6, 4, 5, 8	00000111
last	BYTE 7	00000011
temp	BYTE ?	00000010
		00000001
.code		00000110
Outer:	LOADI A, 0	00000100
	LOAD D, [last]	00000101
	LOADI B, 0	00001000
	CMP A, D	00000111
	BRGE End	00000000
Inner:	LOAD D, [last]	
	SUB D, A	
	CMP B, D	
	BRGE Iinc	
If:	LOADF C, [array+B]	
	LOADF D, [array+B+1]	
	CMP D, C	
	BRGE Jinc	
Swap:	STOREF [array+B], D	
	STOREF [array+B+1], C	
Jinc:	ADDI B, 1	
	JUMP Inner	
Iinc:	ADDI A, 1	
	JUMP Outer	
End:	NOOP	

Machine Code Version

```
.data
array  BYTE 7, 3, 2, 1, 6, 4, 5, 8
last   BYTE 7
temp   BYTE ?

.code

Outer:  LOADI  A, 0
        LOAD  D, [last]
        LOADI B, 0
        CMP   A, D
        BRGE  End
Inner:  LOAD  D, [last]
        SUB   D, A
        CMP   B, D
        BRGE  Iinc
If:     LOADF C, [array+B]
        LOADF D, [array+B+1]
        CMP   D, C
        BRGE  Jinc
Swap:   STOREF [array+B], D
        STOREF [array+B+1], C
Jinc:   ADDI  B, 1
        JUMP  Inner
Iinc:   ADDI  A, 1
        JUMP  Outer
End:    NOOP
```

Data Memory:

```
00000111 //array[0]
00000011 //array[1]
00000010 //array[2]
00000001 //array[3]
00000110 //array[4]
00000100 //array[5]
00000101 //array[6]
00001000 //array[7]
00000111 //last
00000000 //temp
```

Machine Code Version

		Address	Data Memory:
.data			
array	BYTE 7, 3, 2, 1, 6, 4, 5, 8	0000	00000111 //array[0]
last	BYTE 7	0001	00000011 //array[1]
temp	BYTE ?	0010	00000010 //array[2]
		0011	00000001 //array[3]
.code		0100	00000110 //array[4]
Outer:	LOADI A, 0	0101	00000100 //array[5]
	LOAD D, [last]	0110	00000101 //array[6]
	LOADI B, 0	0111	00001000 //array[7]
	CMP A, D	1000	00000111 //last
	BRGE End	1001	00000000 //temp
Inner:	LOAD D, [last]		
	SUB D, A		
	CMP B, D		
	BRGE Iinc		
If:	LOADF C, [array+B]		
	LOADF D, [array+B+1]		
	CMP D, C		
	BRGE Jinc		
Swap:	STOREF [array+B], D		
	STOREF [array+B+1], C		
Jinc:	ADDI B, 1		
	JUMP Inner		
Iinc:	ADDI A, 1		
	JUMP Outer		
End:	NOOP		

Machine Code Version

		Address	Data Memory:
.data			
array	BYTE 7, 3, 2, 1, 6, 4, 5, 8	0000	00000111 //array[0]
last	BYTE 7	0001	00000011 //array[1]
temp	BYTE ?	0010	00000010 //array[2]
		0011	00000001 //array[3]
.code		0100	00000110 //array[4]
Outer:	LOADI A, 0	0101	00000100 //array[5]
	LOAD D, [last]	0110	00000101 //array[6]
	LOADI B, 0	0111	00001000 //array[7]
	CMP A, D	1000	00000111 //last
	BRGE End	1001	00000000 //temp
Inner:	LOAD D, [last]	1010	00000000
	SUB D, A	1011	00000000
	CMP B, D	1100	00000000
	BRGE Iinc	1101	00000000
If:	LOADF C, [array+B]	1110	00000000
	LOADF D, [array+B+1]	1111	00000000
	CMP D, C		
	BRGE Jinc		
Swap:	STOREF [array+B], D		
	STOREF [array+B+1], C		
Jinc:	ADDI B, 1		
	JUMP Inner		
Iinc:	ADDI A, 1		
	JUMP Outer		
End:	NOOP		

Machine Code Version

```
.data
array  BYTE 7, 3, 2, 1, 6, 4, 5, 8
last   BYTE 7
temp   BYTE ?
```

```
.code

        LOADI  A, 0
Outer:  LOAD    D, [last]
        LOADI  B, 0
        CMP    A, D
        BRGE  End
Inner:  LOAD    D, [last]
        SUB    D, A
        CMP    B, D
        BRGE  Iinc
If:     LOADF  C, [array+B]
        LOADF  D, [array+B+1]
        CMP    D, C
        BRGE  Jinc
Swap:   STOREF [array+B], D
        STOREF [array+B+1], C
Jinc:   ADDI   B, 1
        JUMP  Inner
Iinc:   ADDI   A, 1
        JUMP  Outer
End:    NOOP
```

Machine Code Version

```
.data
array  BYTE 7, 3, 2, 1, 6, 4, 5, 8
last   BYTE 7
temp   BYTE ?
```

```
.code

Outer:  LOADI  A, 0
        LOAD  D, [last]
        LOADI B, 0
        CMP   A, D
        BRGE  End
Inner:  LOAD  D, [last]
        SUB   D, A
        CMP   B, D
        BRGE  Iinc
If:     LOADF C, [array+B]
        LOADF D, [array+B+1]
        CMP   D, C
        BRGE  Jinc
Swap:   STOREF [array+B], D
        STOREF [array+B+1], C
Jinc:   ADDI  B, 1
        JUMP  Inner
Iinc:   ADDI  A, 1
        JUMP  Outer
End:    NOOP
```

Code Memory:

```
0011000000000000
1000110000001000
0011010000000000
1101001100000000
1111001100001110
1000110000001000
0110110000000000
1101011100000000
1111001100001000
1001100100000000
1001110100000001
1101111000000000
1111001100000010
1011110100000000
1011100100000001
0101010000000001
1110000011110100
0101000000000001
1110000011101110
0000000000000000
```

Machine Code Version

```
.data
array  BYTE 7, 3, 2, 1, 6, 4, 5, 8
last   BYTE 7
temp   BYTE ?
```

		Address	Code Memory:
	LOADI A, 0	100000	0011000000000000
Outer:	LOAD D, [last]	100001	1000110000001000
	LOADI B, 0	100010	0011010000000000
	CMP A, D	100011	1101001100000000
	BRGE End	100100	1111001100001110
Inner:	LOAD D, [last]	100101	1000110000001000
	SUB D, A	100110	0110110000000000
	CMP B, D	100111	1101011100000000
	BRGE Iinc	101000	1111001100001000
If:	LOADF C, [array+B]	101001	1001100100000000
	LOADF D, [array+B+1]	101010	1001110100000001
	CMP D, C	101011	1101111000000000
	BRGE Jinc	101100	1111001100000010
Swap:	STOREF [array+B], D	101101	1011110100000000
	STOREF [array+B+1], C	101110	1011100100000001
Jinc:	ADDI B, 1	101111	0101010000000001
	JUMP Inner	110000	1110000011110100
Iinc:	ADDI A, 1	110001	0101000000000001
	JUMP Outer	110010	1110000011101110
End:	NOOP	110011	0000000000000000

Questions?

THE END