CS221: Digital Design

FSM State Encoding

A. Sahu
Dept of Comp. Sc. & Engg.
Indian Institute of Technology Guwahati

Encoding

- What is State encoding?
- Random Encoding
- Sequential Encoding
- Gray Encoding
- One Hot Encoding
- Output Oriented Encoding
- Heuristics for Encoding

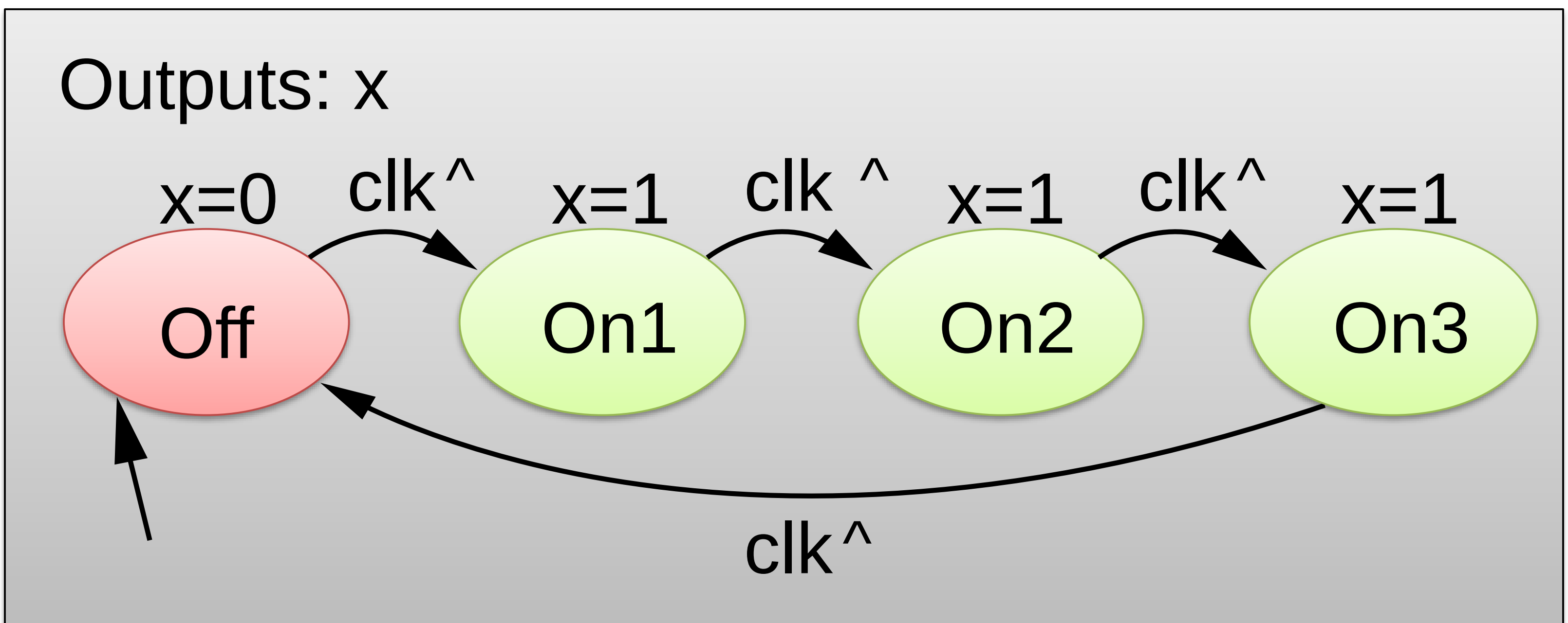
State Encoding/Assignment

State assignment

- State encoding:
  - Assigning unique binary value to each state
  - So that we can implement FSM
- Since we don't care about the actual flip-flop values for each state
- We can assign each state to any binary number we like **as long as**
  - Each state is assigned a unique binary number

Example of FSM: 3 cycle Laser

- Require 4 States
- Assigning unique binary value/code to each state
- Options available : 4!



State Encoding Example

- Four state FSM: S0, S1, S2, S3 : 4! options

SL	S0	S1	S2	S3
1	00	01	10	11
2	00	01	11	10
3	00	10	01	11
4	00	10	11	01
5	00	11	01	10
6	00	11	10	01
7	01	00	11	10
8	01	00	10	11
9	01	10	11	00
10	01	10	00	11
11	01	11	10	00
12	01	11	00	11

SL	S0	S1	S2	S3
13	10	01	00	11
14	10	01	11	00
15	10	00	01	11
16	10	00	11	01
17	10	11	01	00
18	10	11	00	01
19	11	00	01	10
20	11	00	10	01
21	11	10	01	00
22	11	10	00	01
23	11	01	10	00
24	11	01	00	11

State assignment

- Since we don’t care about the actual flip-flop values for each state we can assign each state to any binary number we like **as long as** each state is assigned a unique binary number
- Suppose a FSM have 6 states: Modulo 6 Counter
- If we use 3 bits to encode the 6 states, we have

$$\binom{8}{6} = \frac{8!}{6!(8-6)!}$$

possible choosing the 6 states from 8 total state

  – Than encodings

State Encoding

- The cost & delay of FSM implementation depends on encoding of symbolic states.
  - e.g., 4 states can be encoded in 4! = 24 different ways
- There are more than n! different encodings for n states.
  - **Exploration of all encodings is impossible**, therefore heuristics are used
- Heuristics Used
  - One-hot encoding, Minimum-bit change, Prioritized adjacency

State assignment

state	Encoding 1 (binary)	Encoding 2 (Gray)	Encoding 3
a	000	000	000
b	001	001	100
c	010	011	010
d	011	010	101
e	100	110	011

State Encoding

- Binary and Gray encoding use the minimum number of bits for state register
- Gray and Johnson code: Two adjacent codes differ by only one bit
  - Reduce simultaneous switching
  - Reduce crosstalk, Reduce glitch

One-hot encoding

- One flip-flop per state encoding
- Leads to greater number of flip-flops than binary encoding but possibly to simpler logic

State Assignment Problem

- Some state assignments are better than others.
- The state assignment influences the complexity of the state machine.
  - The combinational logic required in the state machine design is dependent on the state assignment.
- Types of state assignment
  - Binary encoding: 2^N states → N Flip-Flops
  - Gray-code encoding: 2^N states → N Flip-Flops
  - One-hot encoding: N states → N Flip-Flops

FSM: State Assignment

Example

Design a FSM that detects a sequence of two or more consecutive ones on an input bit stream.

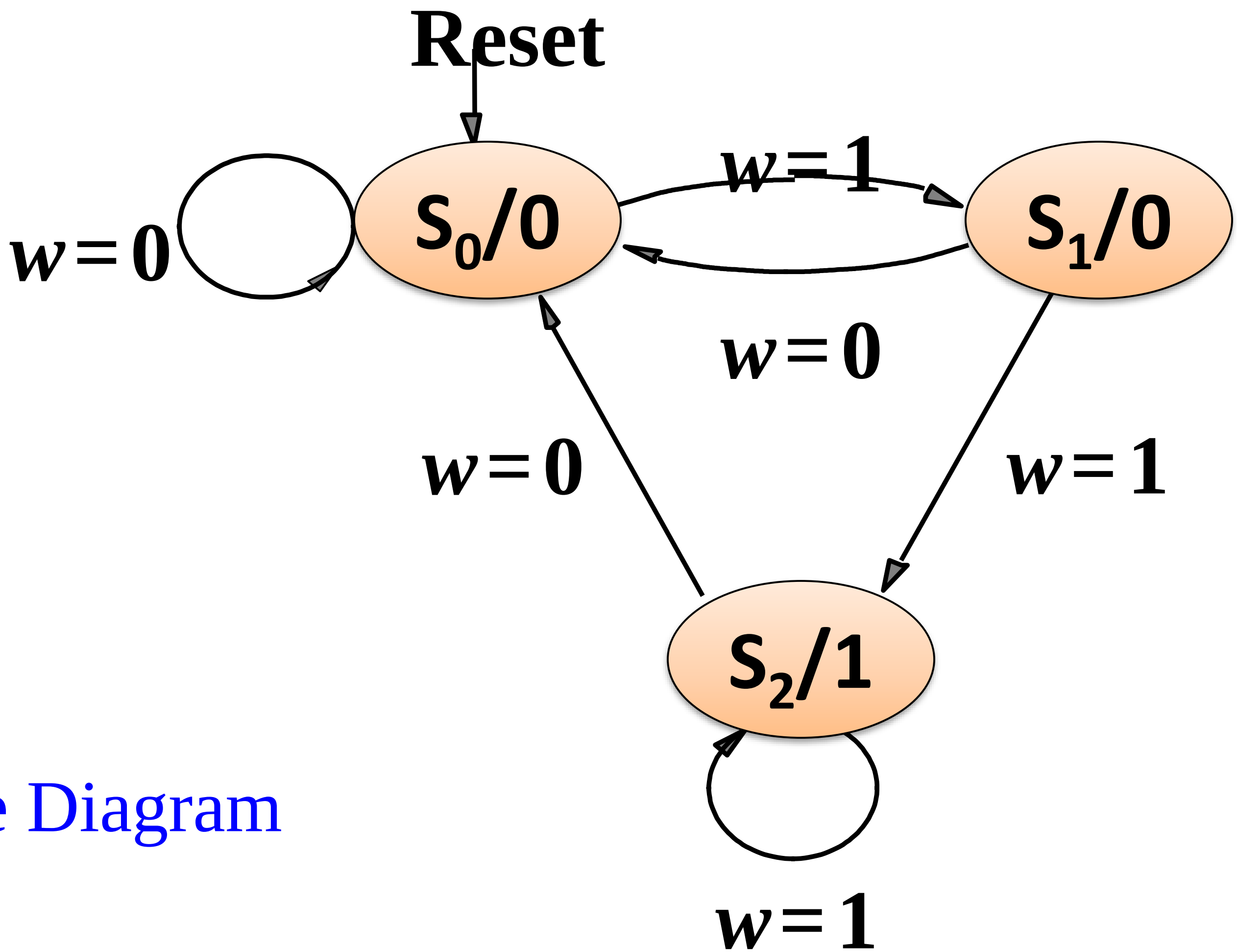
The FSM should output a 1 when the sequence is detected, and a 0 otherwise.

FSM: State Assignment

Input: 0 1 1 1 0 1 0 1 1 0 1 1 1 0 1 …

Output: 0 0 1 1 0 0 0 0 1 0 0 1 1 0 0 …

FSM: State Assignment



State Diagram

FSM: State Assignment

Present State	Next State		Output
	w = 0	w = 1	
S_0	S_0	S_1	0
S_1	S_0	S_2	0
S_2	S_0	S_2	1

State Table

FSM: State Assignment #1

State Assigned Table

Present State			Next State						Output
			w = 0			w = 1			
	Q_A	Q_B		Q_A^+	Q_B^+		Q_A^+	Q_B^+	z
S_0	0	0	S_0	0	0	S_1	0	1	0
S_1	0	1	S_0	0	0	S_2	1	0	0
S_2	1	0	S_0	0	0	S_2	1	0	1
	1	1		d	d		d	d	d

Using <u>Binary</u> Encoding for the State Assignment

$D_B = wQ'_A Q_B'$

$D_A = w(Q_A + Q_B)$

$Z = Q_A$

FSM: State Assignment #2

State Assigned Table

Present State			Next State						Output
			w = 0			w = 1			
	Q_A	Q_B		Q_A^+	Q_B^+		Q_A^+	Q_B^+	z
S_0	0	0	S_0	0	0	S_1	0	1	0
S_1	0	1	S_0	0	0	S_2	1	1	0
S_2	1	1	S_0	0	0	S_2	1	1	1
	1	0		d	d		d	d	d

Using <u>Gray-code</u> Encoding for the State Assignment

$D_A = w.Q_B$

$D_B = W$

$Z = Q_A$

FSM: State Assignment #3

State Assigned Table

Present State				Next State							
				w = 0				w = 1			
	Q_A	Q_B	Q_C		Q_A^+	Q_B^+	Q_C^+		Q_A^+	Q_B^+	Q_C^+
S_0	0	0	1	S_0	0	0	1	S_1	0	1	0
S_1	0	1	0	S_0	0	0	1	S_2	1	0	0
S_2	1	0	0	S_0	0	0	1	S_2	1	0	0

Using One-hot Encoding for the State Assignment

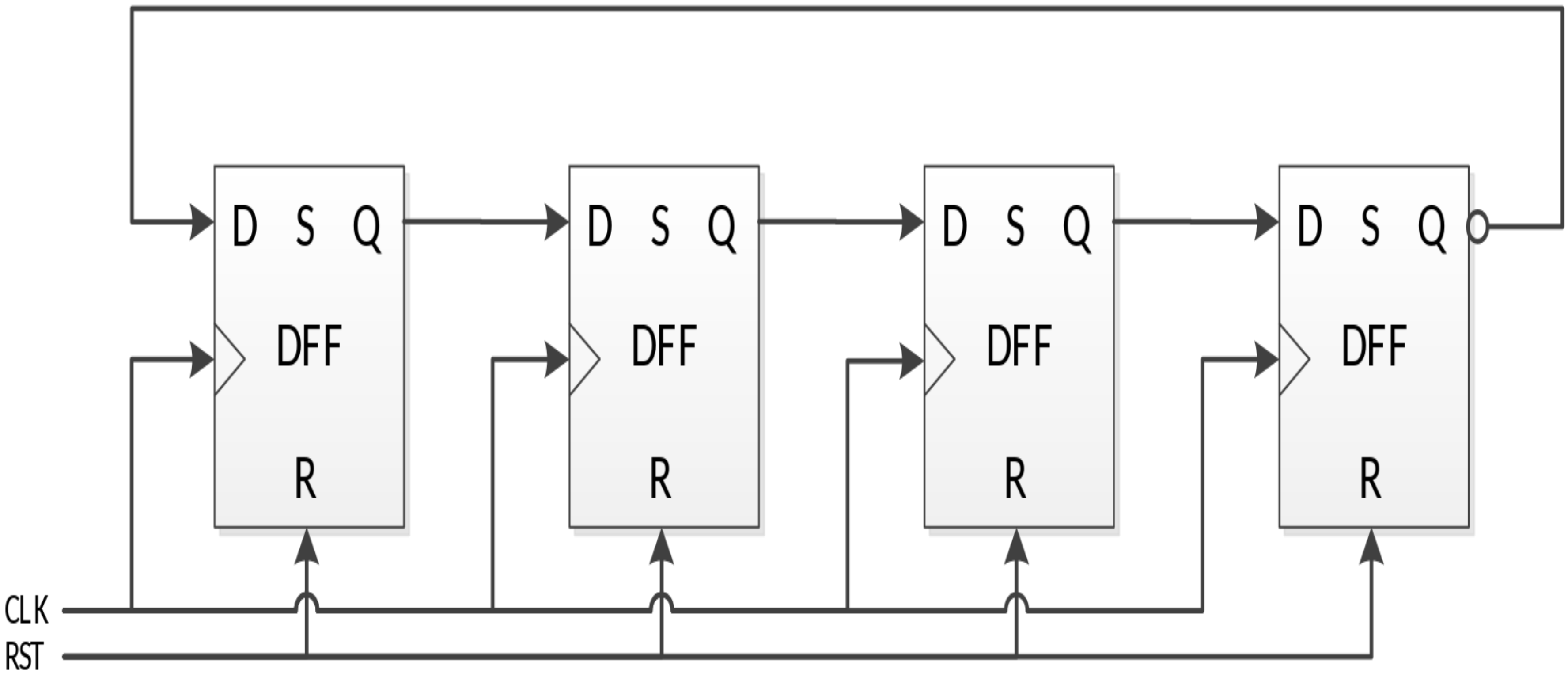
For each state only one flip-flop is set to 1. The remaining combination of state variables are not used.

$D_A = w.Q_C'$
$D_B = w.Q_C$
$D_C = w'$

State Encoding

#	Binary	Gray	Johnson	One-hot
0	000	000	0000	00000001
1	001	001	0001	00000010
2	010	011	0011	00000100
3	011	010	0111	00001000
4	100	110	1111	00010000
5	101	111	1110	00100000
6	110	101	1100	01000000
7	111	100	1000	10000000

Johnson Counter



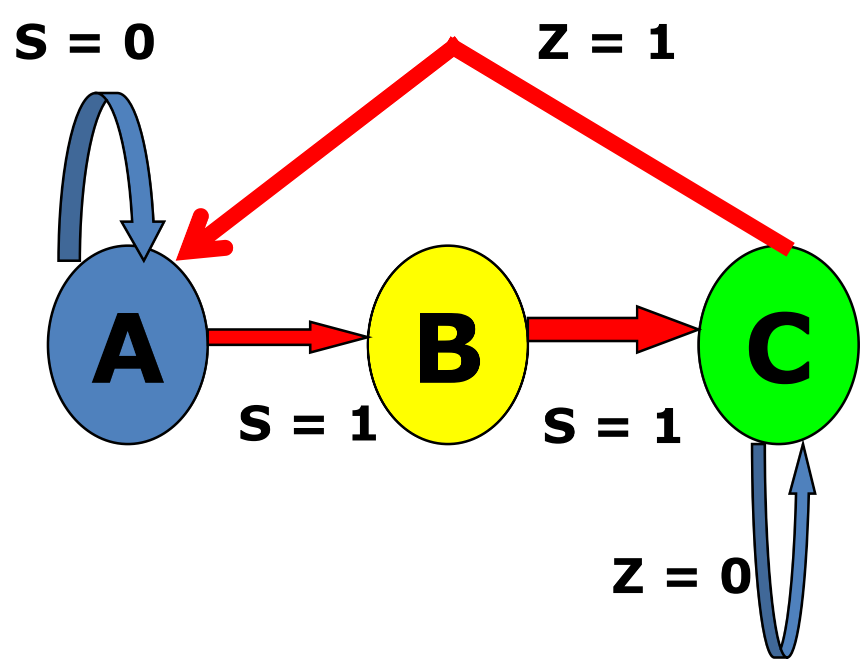
State Encoding

- The cost & delay of FSM implementation depends on encoding of symbolic states.
  - e.g., 4 states can be encoded in 4! = 24 different ways
- There are more than n! different encodings for n states.
  - **Exploration of all encodings is impossible**, therefore heuristics are used
- Heuristics Used
  - One-hot encoding, Minimum-bit change, Prioritized adjacency

One-hot Encoding

- Uses redundant encoding in which one flip-flop is assigned to each state.
- Each state is distinguishable by its own flip-flop having a value of 1 while all others have a value of 0.

A: 001
B: 010
C: 100

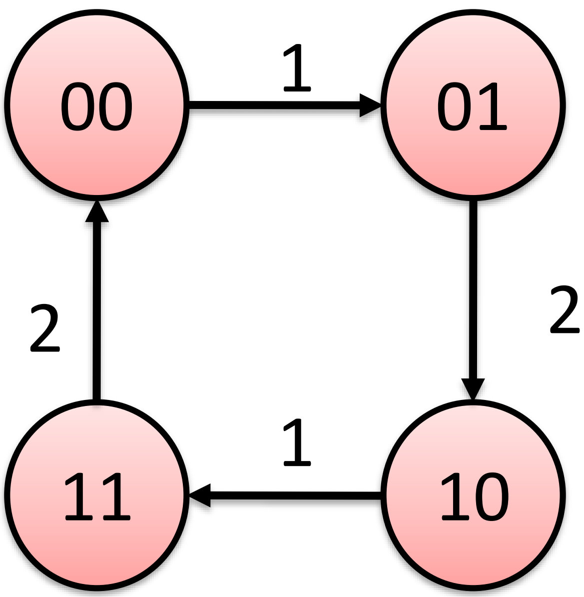


Minimum-bit Change

- Assigns codes to states so that
  - The total number of **bit changes** for all state transitions **is minimized.**
- In other words, if every arc in the state diagram has a **weight**
  - That is equal to the **number of bits** by which the source and destination encoding **differ**
  - This strategy would select the one that **minimizes the sum** of all these weights.

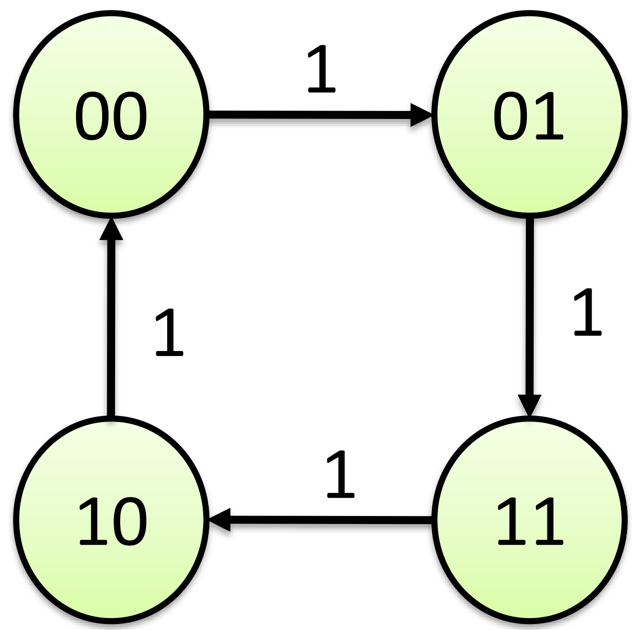
Minimum-bit Change

Encoding with
6 bit changes



Straight-forward
sequential Encoding

Encoding with
4 bit changes

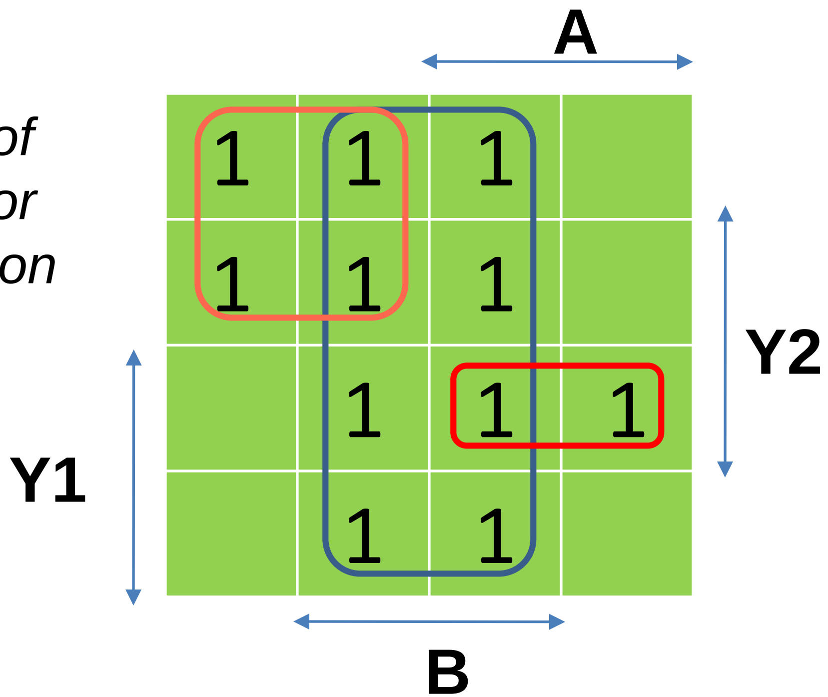


Minimum Bit
Change Encoding

The Idea of Adjacency

- Inputs are A and B
- State variables are Y1 and Y2
- An output is F(A, B, Y1, Y2) or F(Y1,Y2)
- A next state function is G(A, B, Y1, Y2)

Karnaugh map of output function or next state function



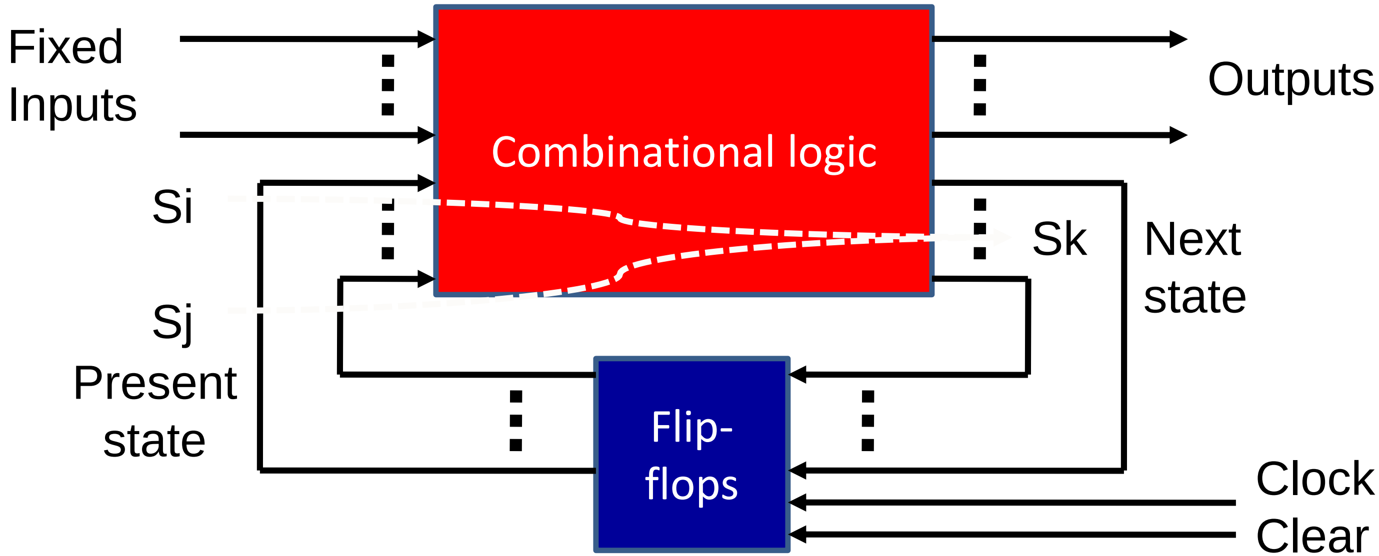
- ➢ Larger clusters produce smaller logic function.
- ➢ Clustered minterms differ in one variable.

Size of an Implementation

- Number of **product terms** determines number of **gates**.
- Number **of literals** in a product term determines number of gate **inputs**, which is proportional to number of transistors.
- Hardware area proportional to total number of literals
- Examples of four minterm functions:
  - F1 = ABCD +A'B'C'D' +A'B'CD +A'B' CD has 16 literals
  - F2 = ABC +A'C'D has 6 literals

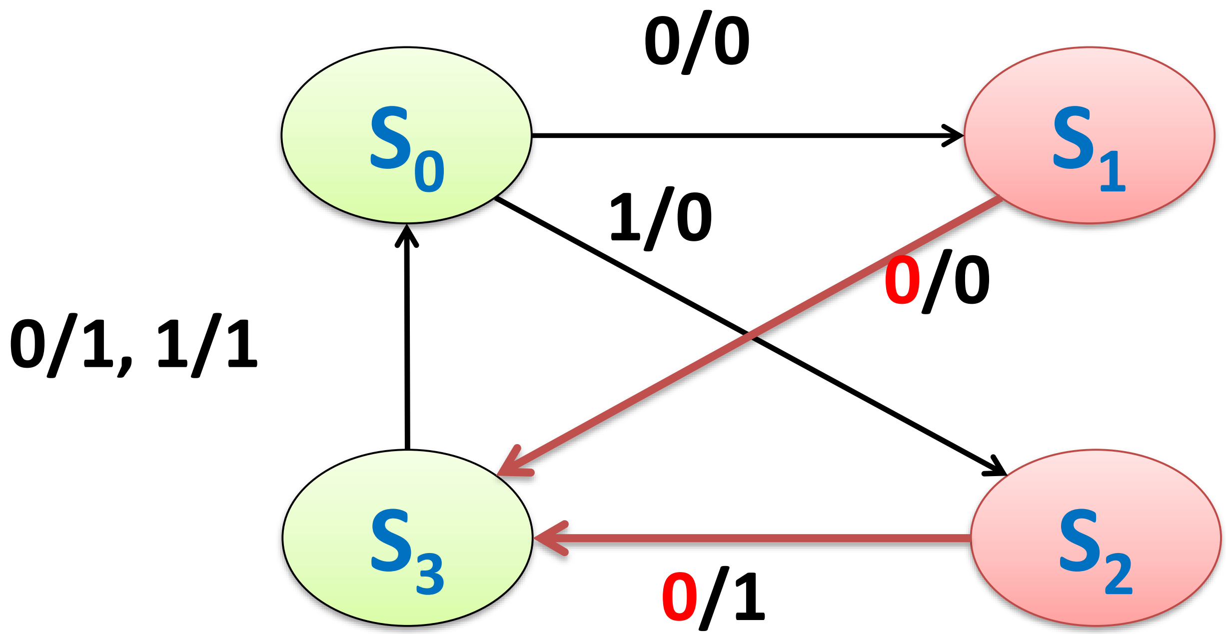
Rule 1 (Priority #1), Common Dest

States that have the same next state for some fixed input should be assigned logically adjacent codes.



Rule 1 (Priority #1), Common Dest

States that have the same next state for some fixed input should be assigned logically adjacent codes.

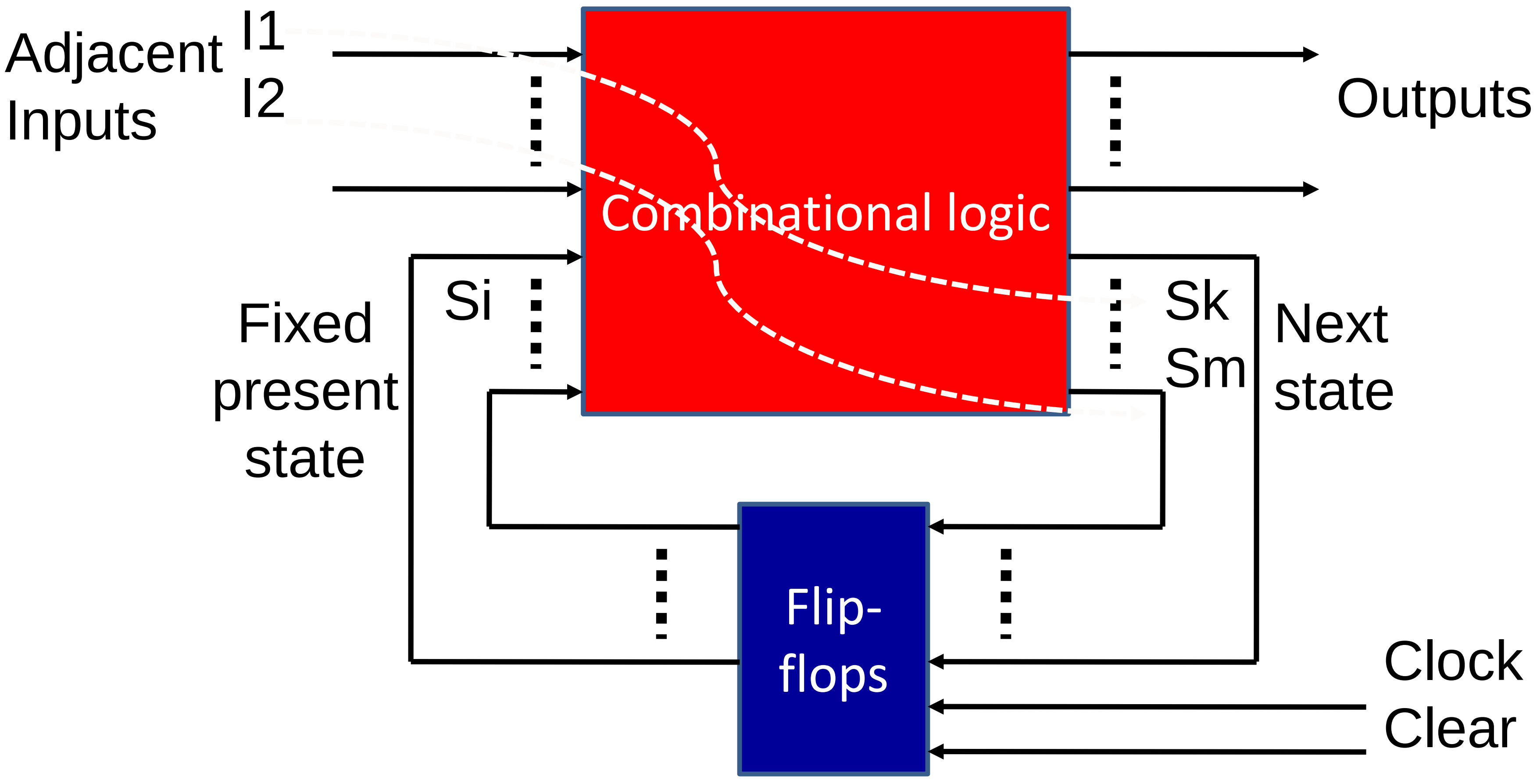


Rule #1: (S_1, S_2)

The input value of 0 will move both states into the same state S_3 → Adj (S_1, S_2)

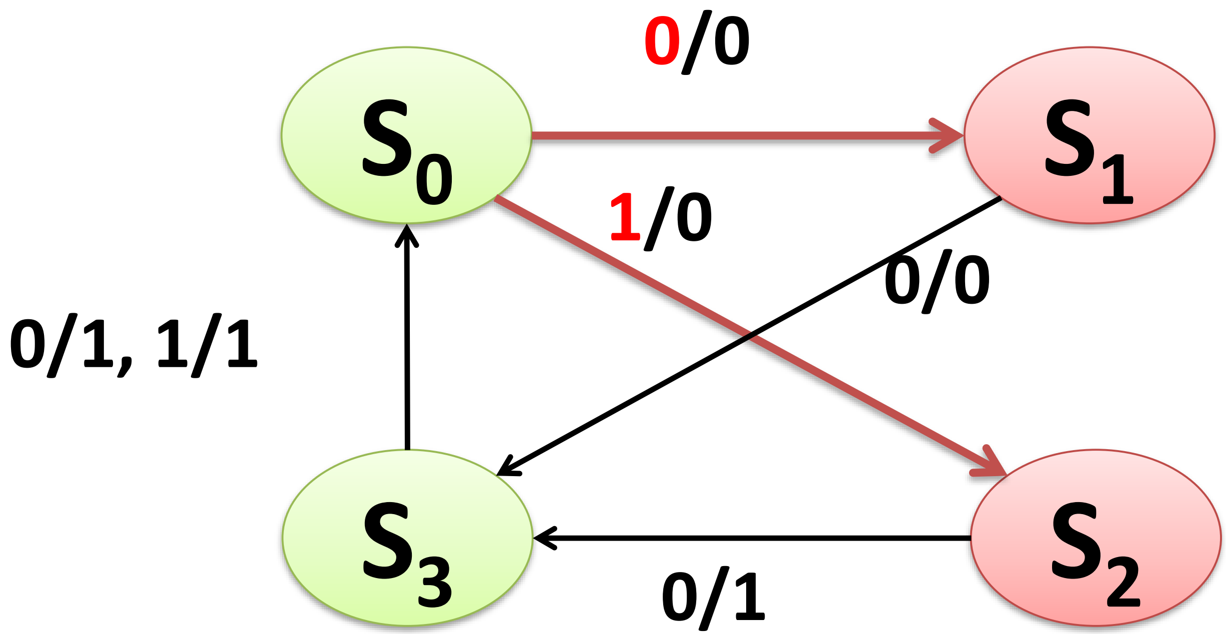
Rule 2 (Priority #2), A Common Source

States that are the next states of the same state under logically adjacent inputs, should be assigned logically adjacent codes.



Rule 2 (Priority #2), A Common Source

States that are the next states of the same state under logically adjacent inputs, should be assigned logically adjacent codes.

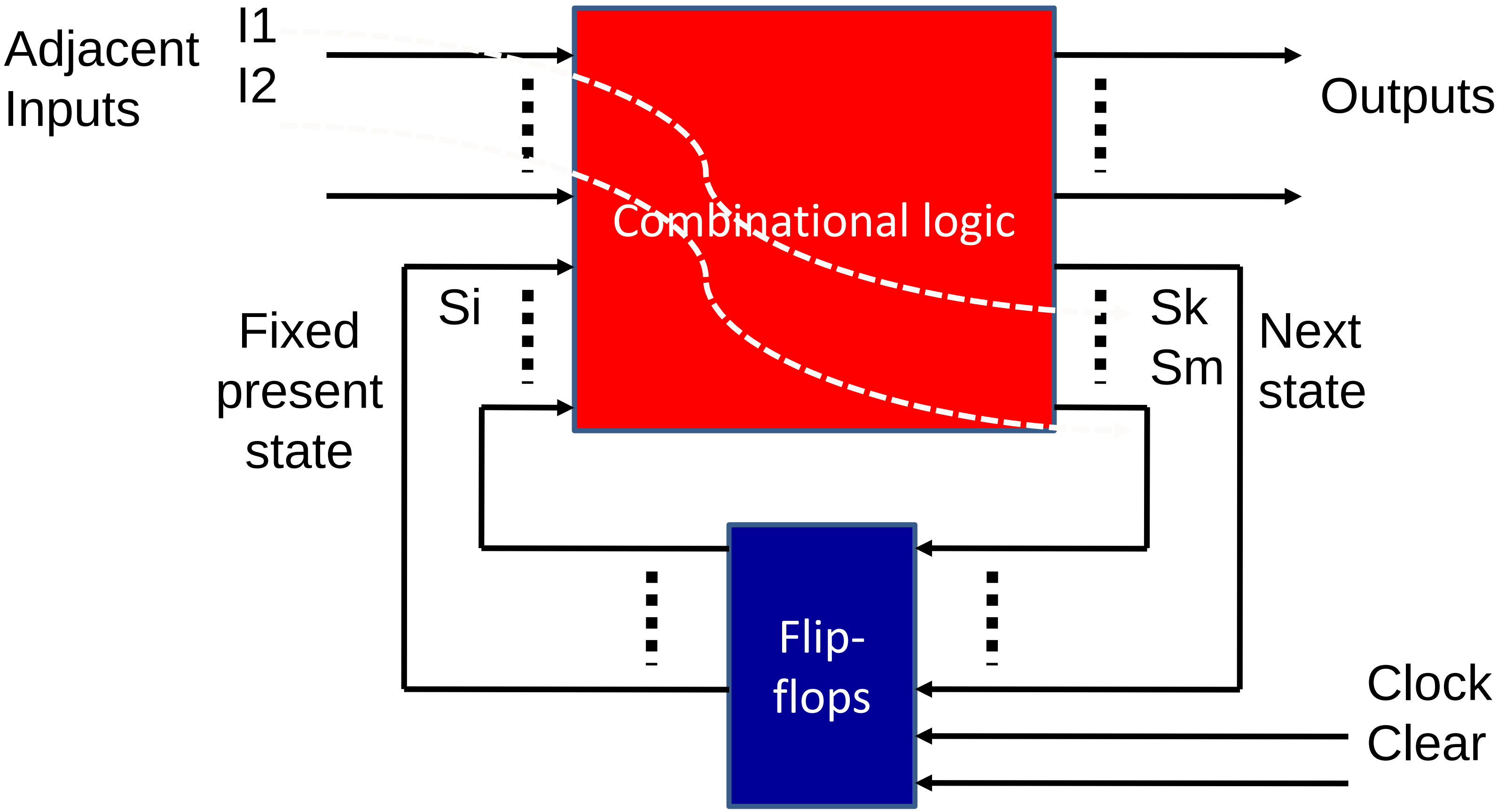


Rule #2: (S_1, S_2)

They are both next states of the state S_0 ➔ Adj (S_1, S_2)

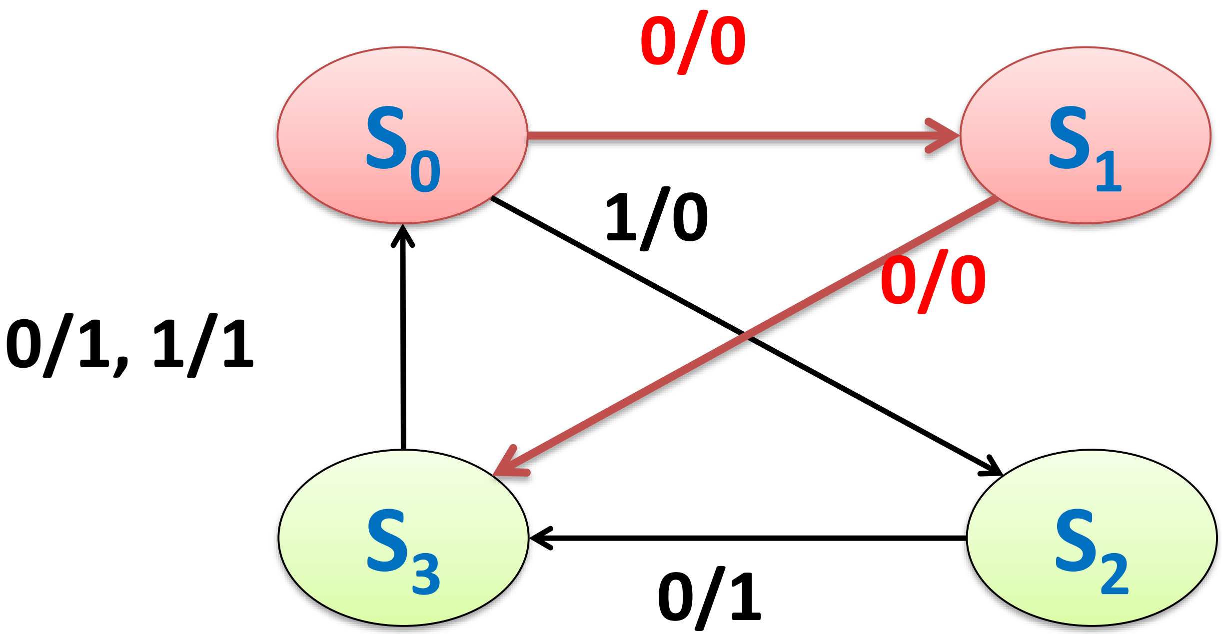
Rule 3 (Priority #3), A Common Output

States that have the same output value for the same input value, should be assigned logically adjacent codes.



Rule 3 (Priority #3), A Common Output

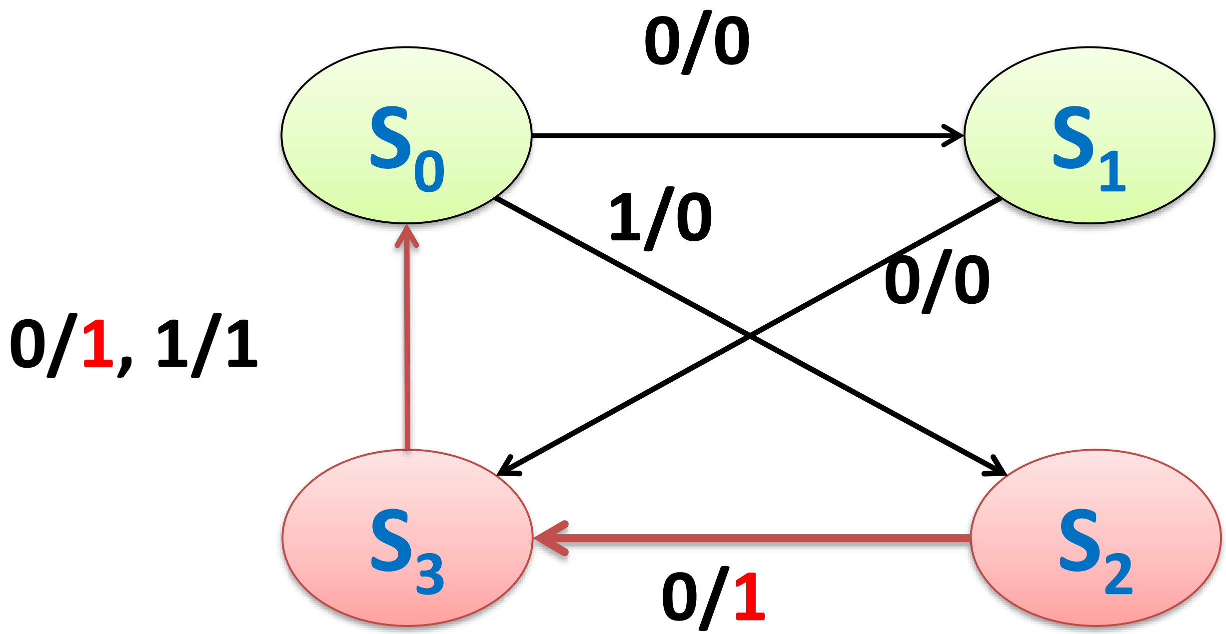
States that have the same output value for the same input value, should be assigned logically adjacent codes.



Rule #3: (S_0, S_1): states S_0 and S_1 have the same output value 0 for the same input value 0 ➔ Adj (S_0, S_1)

Rule 3 (Priority #3), A Common Output

States that have the same output value for the same input value, should be assigned logically adjacent codes.

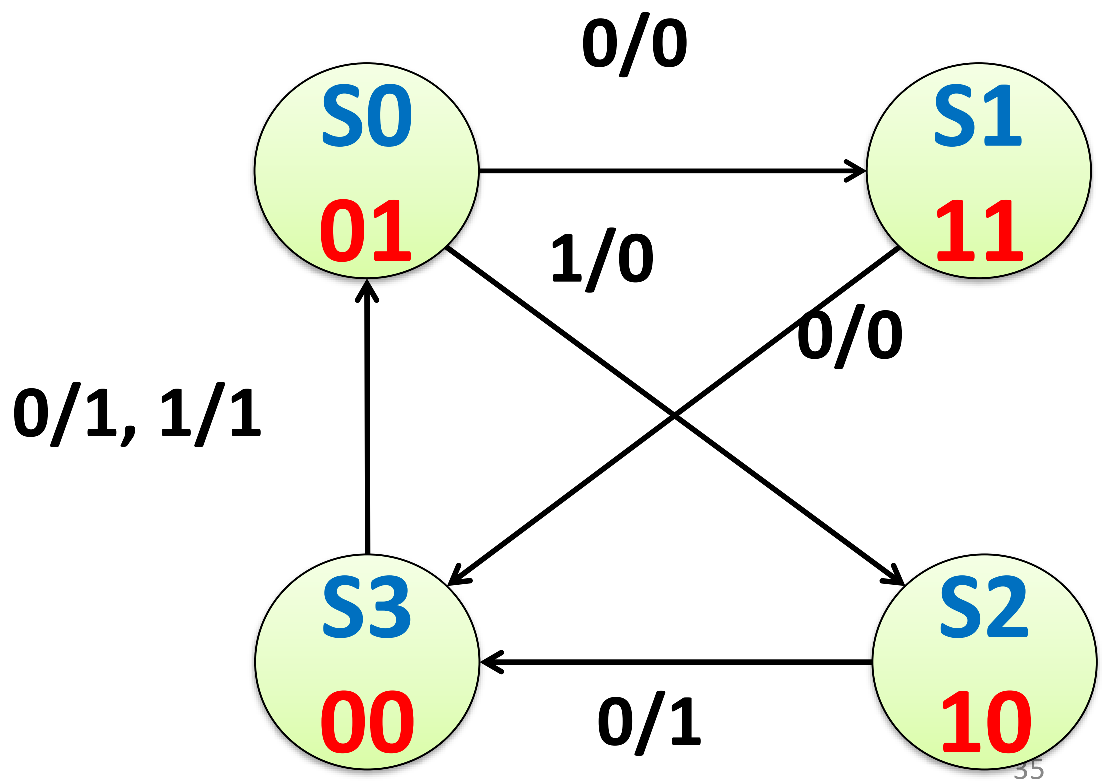


Rule #3: (S_2, S_3), states S_2 and $_3$ have the same output value 1 for the same input value 0 → Adj (S_2, S_3)

Applying Rules 1,2 and 3

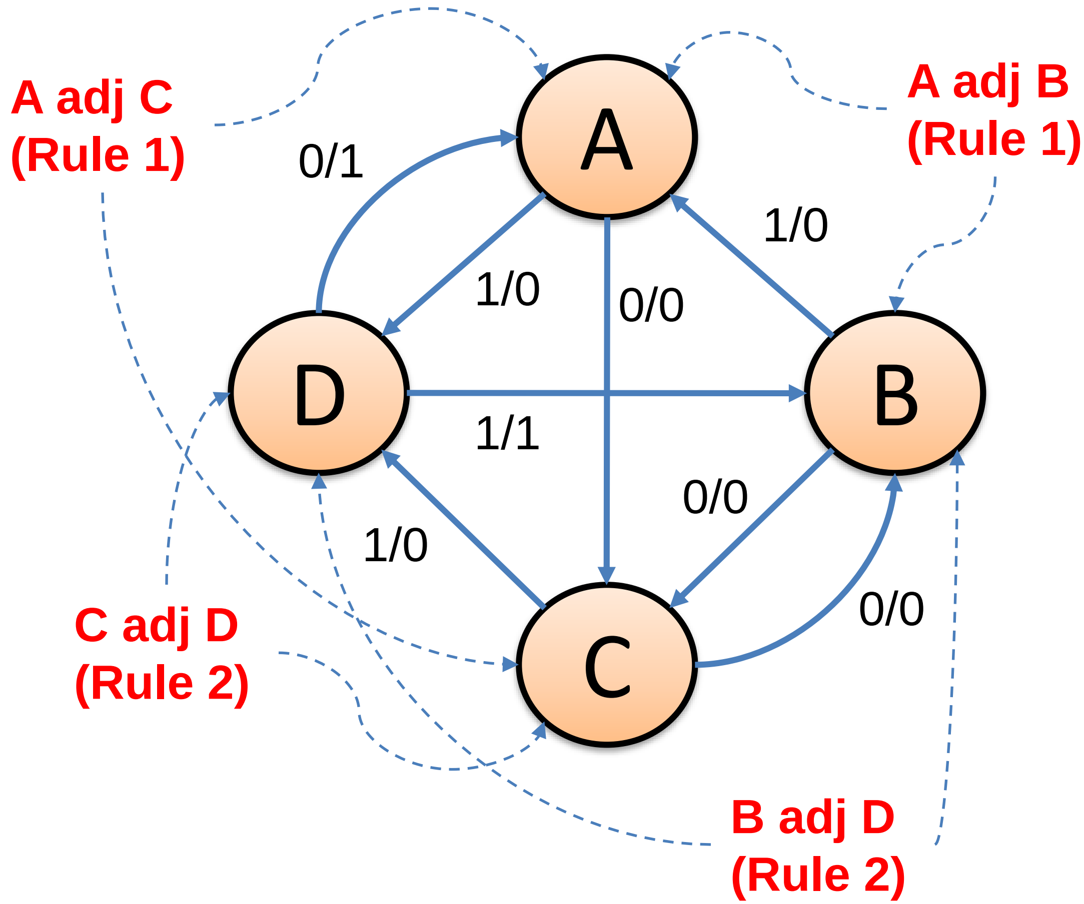
- Rule 1: S1, S2 ==> HD(11, 10)=1
- Rule 2 : S1, S2
- Rule 3: S0, S1 and S2, S3
  - HD(S0,S1)=1
  - HD(S2,S3)=1

- **HD(S0,S2)=2**
- **HD(S1,S3)=2**



Example of State Assignment

Present state	Next state, output (Z)	
	Input, X	
	0	1
A	C, 0	D, 0
B	C, 0	A, 0
C	B, 0	D, 0
D	A, 1	B, 1

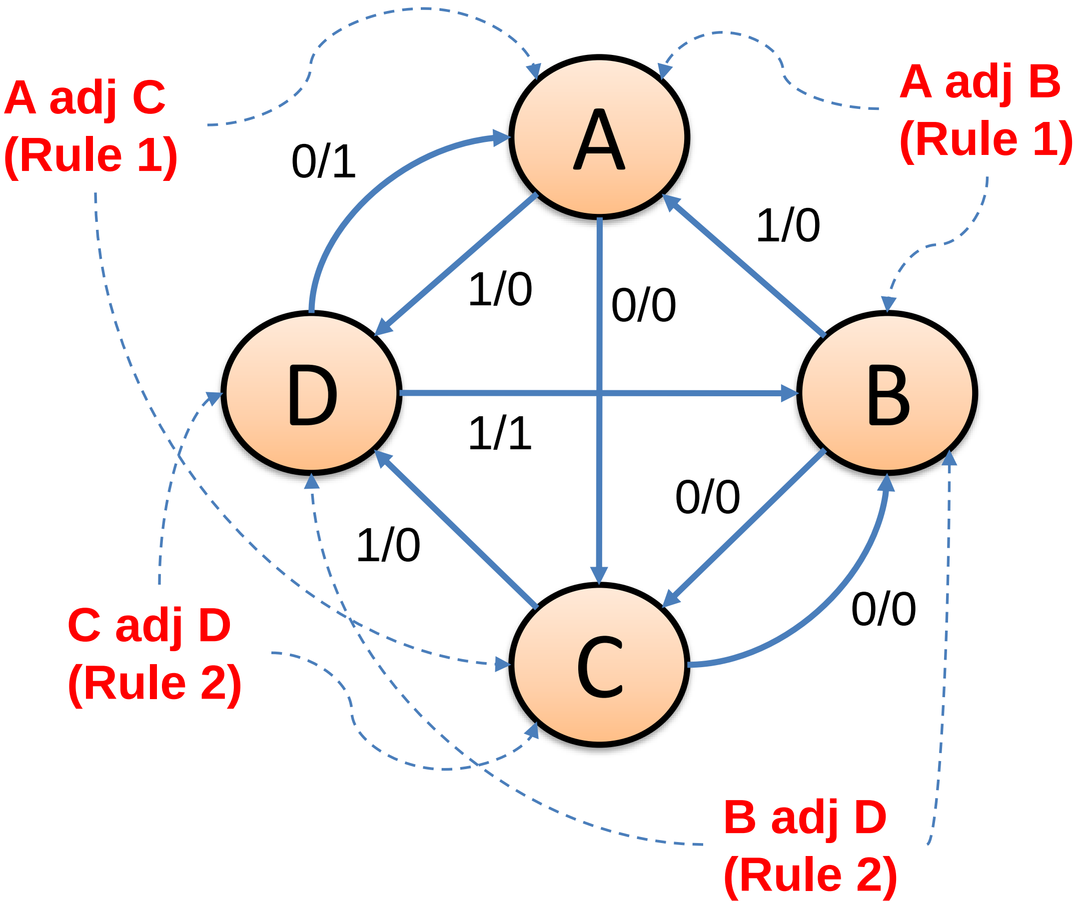


Rule 1: A and C goes to same next state D for same input 1

Rule 1: A and B goes to same next state C for same input 0

Example of State Assignment

Present state	Next state, output (Z)	
	Input, X	
	0	1
A	C, 0	D, 0
B	C, 0	A, 0
C	B, 0	D, 0
D	A, 1	B, 1

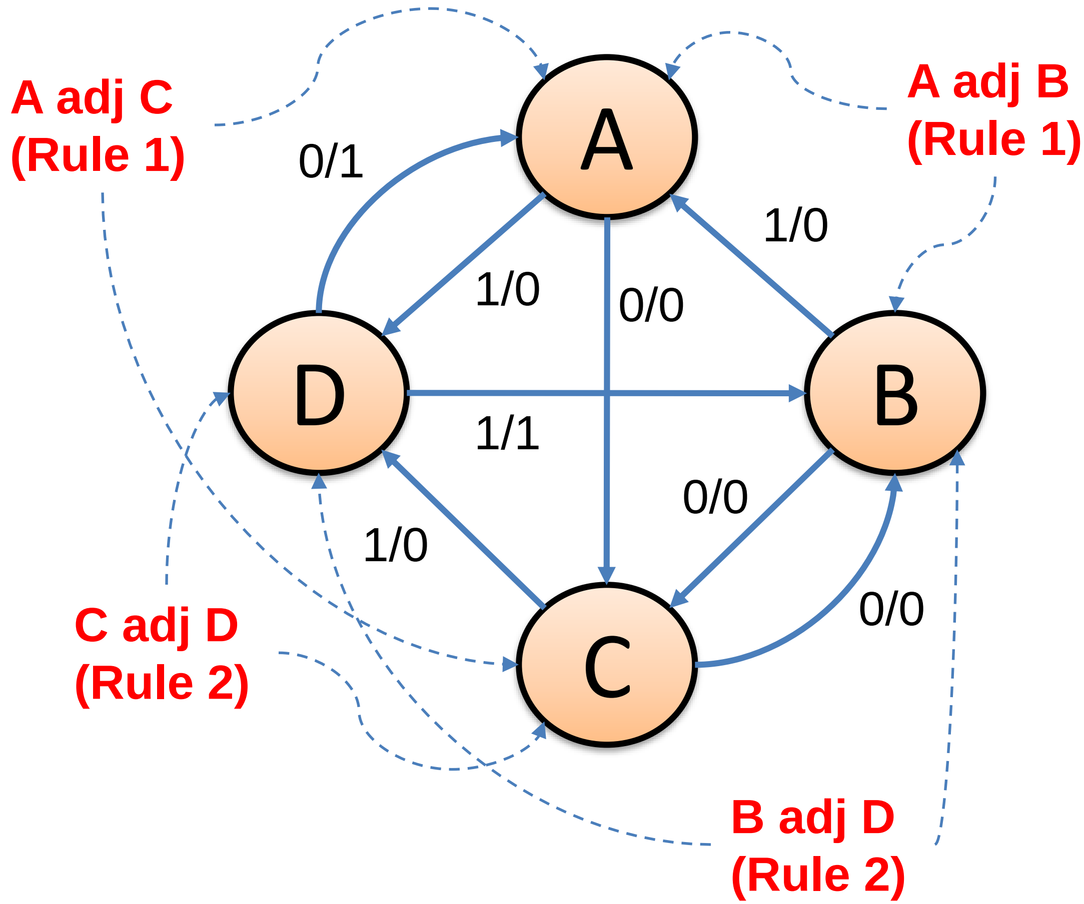


Rule 2: C and D are next state of A for input 0 and 1

Rule 2: B and D are next state of C for input o and 1

Example of State Assignment

Present state	Next state, output (Z)	
	Input, X	
	0	1
A	C, 0	D, 0
B	C, 0	A, 0
C	B, 0	D, 0
D	A, 1	B, 1



	0	1
0	A	B
1	C	D

Verify that BC and AD are not adjacent.

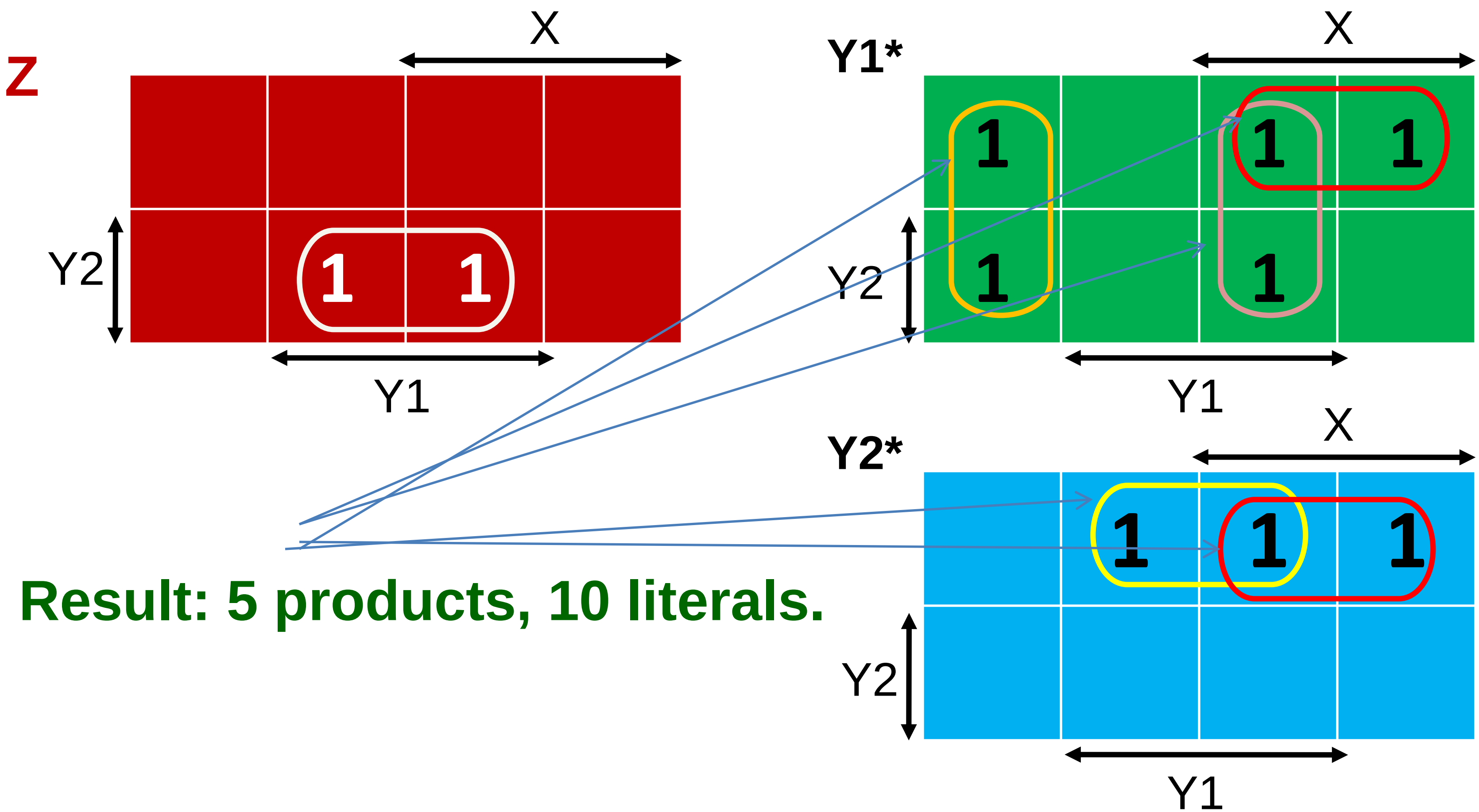
State Assignment #1

A = 00, B = 01, C = 10, D = 11

Present state Y1, Y2	Next state, output Y1*Y2*, Z Input, X 0	 1
A = 00	10 / 0	11 / 0
B = 01	10 / 0	00 / 0
C = 10	01 / 0	11 / 0
D = 11	00 / 1	01 / 1

Input	Present state		Output	Next state	
X	Y1	Y2	Z	Y1*	Y2*
0	0	0	0	1	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	0	0
1	0	0	0	1	1
1	0	1	0	0	0
1	1	0	0	1	1
1	1	1	1	1	0

Logic Minimization for Optimum State Assignment



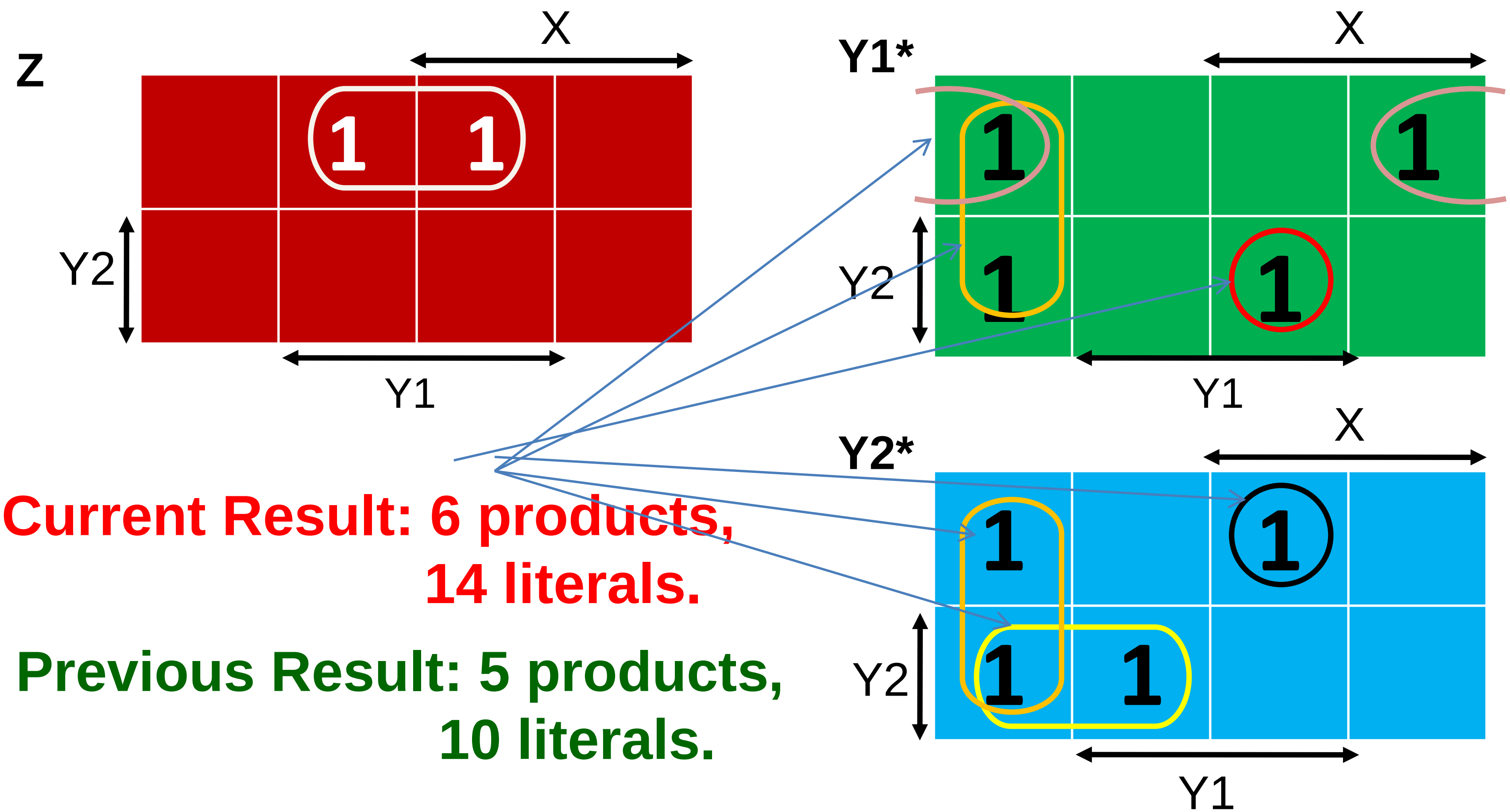
Result: 5 products, 10 literals.

Using an Arbitrary State Assignment:
A = 00, B = 01, C = 11, D = 10

Present state Y1, Y2	Next state, output Y1*Y2*, Z Input, X 0	 1
A = 00	11 / 0	10 / 0
B = 01	11 / 0	00 / 0
C = 11	01 / 0	10 / 0
D = 10	00 / 1	01 / 1

Input	Present state		Output	Next state	
X	Y1	Y2	Z	Y1*	Y2*
0	0	0	0	1	1
0	0	1	0	1	1
0	1	0	1	0	0
0	1	1	0	0	1
1	0	0	0	1	0
1	0	1	0	0	0
1	1	0	1	0	1
1	1	1	0	1	0

Logic Minimization for Arbitrary State Assignment



Current Result: 6 products, 14 literals.

Previous Result: 5 products, 10 literals.