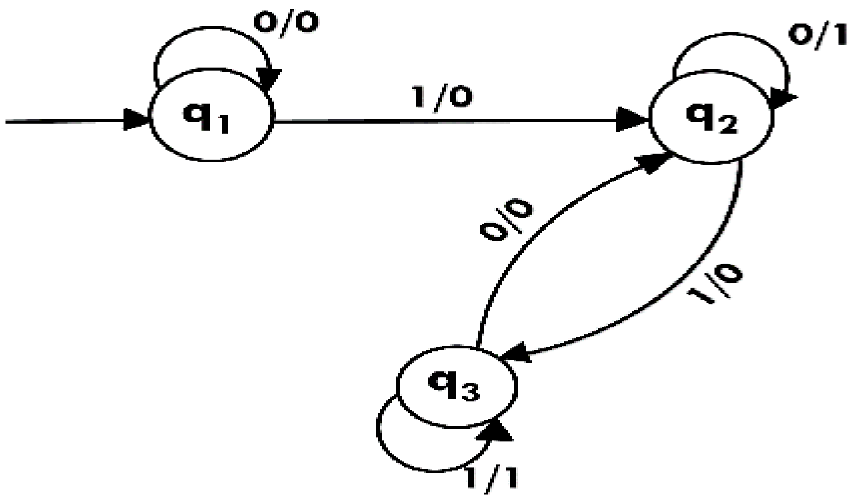
Course Name: Digital Hardware Design
Course Code: 17B1NEC741

Finite State Machine-7

Dr. Arti Noor
Dean, Academic Affairs
Electronics and Communication Engineering,
Jaypee Institute of Information Technology, Noida

Conversion From Mealy to Moore Machine

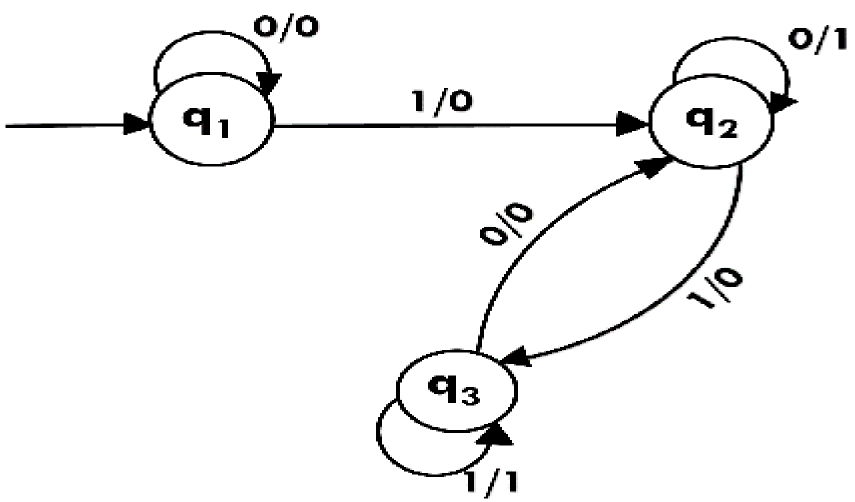
Example-1: Convert the following Mealy machine into an equivalent Moore machine



Present State	Next State 0		Next State 1	
	State	o/P	State	o/P
q_1	q_1	0	q_2	0
q_2	q_2	1	q_3	0
q_3	q_2	0	q_3	1

Conversion From Mealy to Moore Machine

Example-1: Convert the following Mealy machine into an equivalent Moore machine



1. State q1 has only one output. No Split
2. State q2 and q3 have both outputs 0 and 1. So create two states. For q2, two states will be q20(with output 0) and q21(with output 1).
3. Similarly, for q3 two states will be q30(with output 0) and q31(with output 1).

Conversion From Mealy to Moore Machine

Example-1 : Design the Moore Machine from Mealy Machine

Present State	Next State 0		Next State 1	
	State	o/P	State	o/P
q_1	q_1	0	q_2	0
q_2	q_2	1	q_3	0
q_3	q_2	0	q_3	1

Mealy Machine State table

Present State	Next State 0	Next State 1	o/P
q_1	q_1	q_{20}	0
q_{20}	q_{21}	q_{30}	0
q_{21}	q_{21}	q_{30}	1
q_{30}	q_{20}	q_{31}	0
q_{31}	q_{20}	q_{31}	1

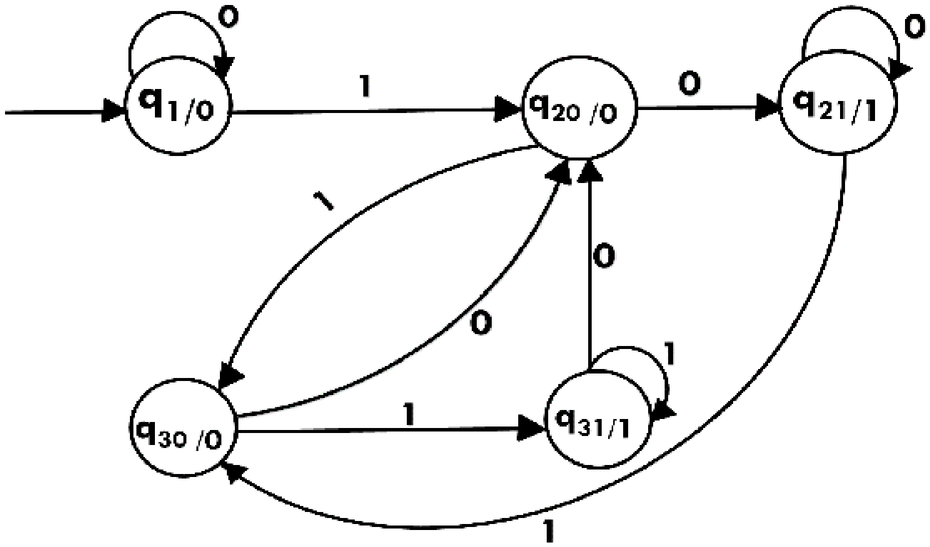
Moore Machine State table

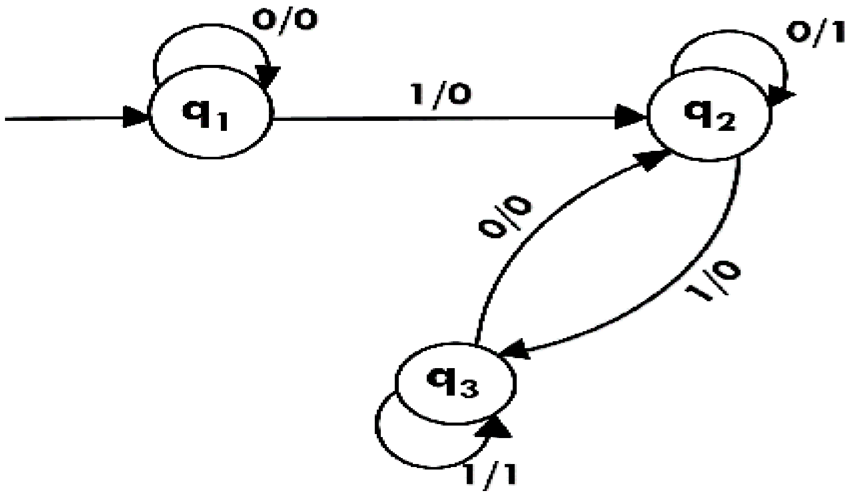
Conversion From Mealy to Moore Machine

Example-1 : Design the Moore Machine from Mealy Machine

Present State	Next State 0	Next State 1	0/P
q_1	q_1	q_{20}	0
q_{20}	q_{21}	q_{30}	0
q_{21}	q_{21}	q_{30}	1
q_{30}	q_{20}	q_{31}	0
q_{31}	q_{20}	q_{31}	1

Moore Machine State table





FSM Optimization

State minimization

- Fewer states require fewer state bits.
- Fewer bits require fewer logic equations;

Encodings: state, inputs, outputs

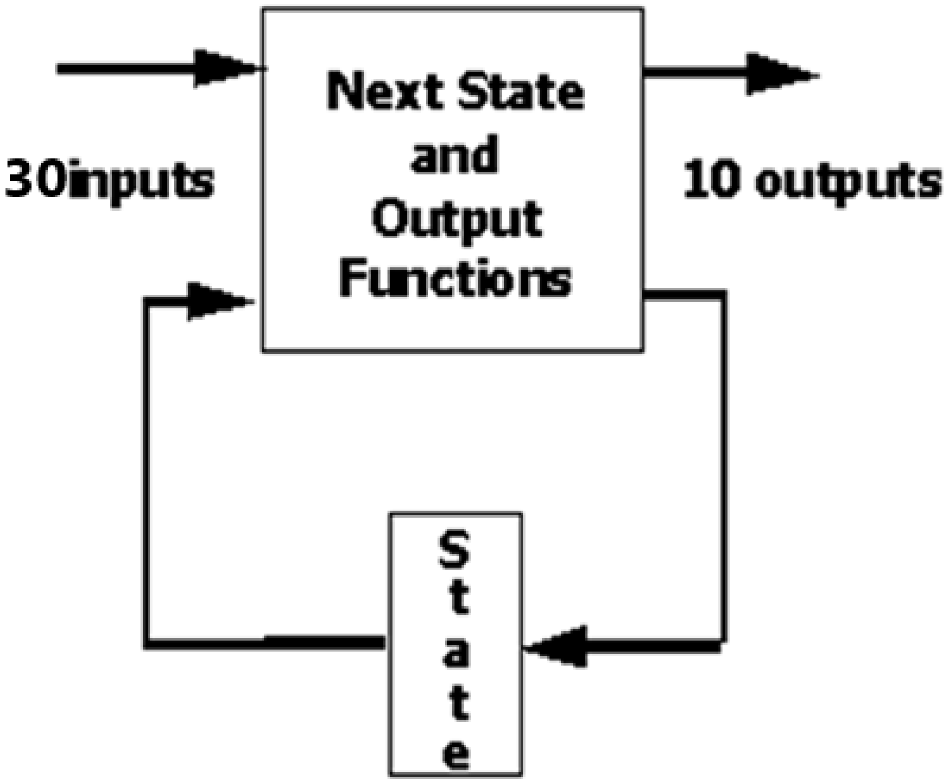
- State encoding with fewer bits has fewer equations to implement.
  - However, each may be more complex.
- State encoding with more bits has simpler equations.
  - Complexity directly related to the complexity of the state diagram,
- Input/output encoding may or may not be under designer control

FSM Optimization
by
FSM Partitioning

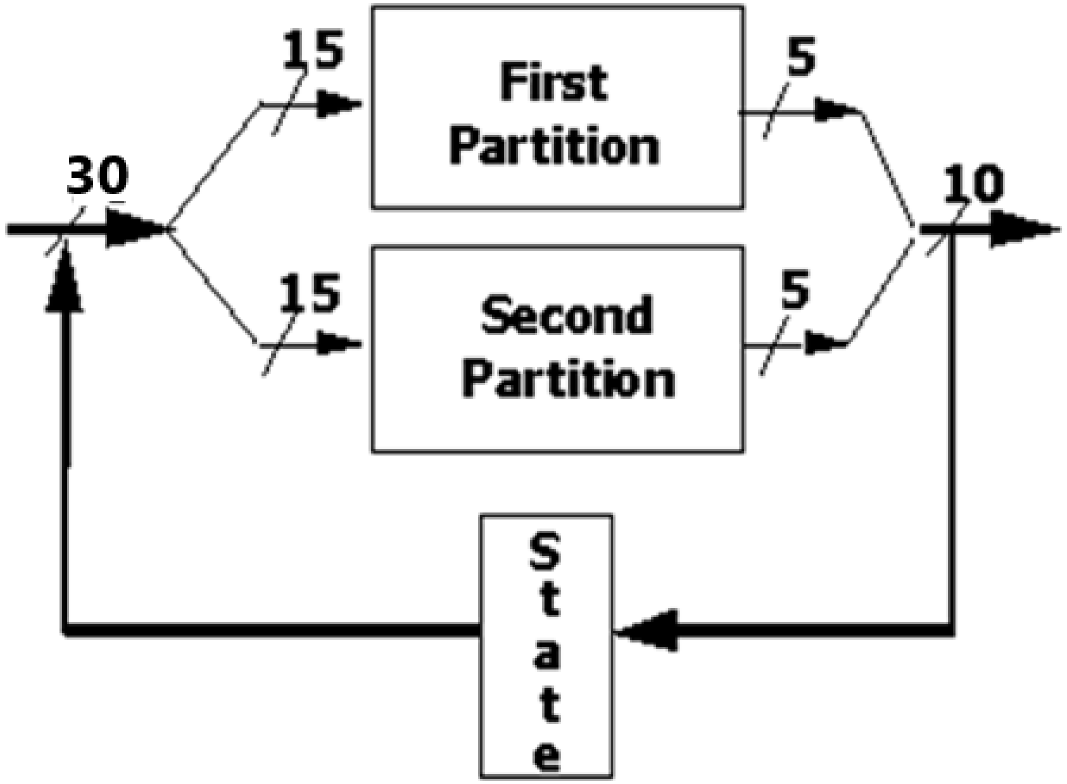
Idea: Break a large FSM into two or more smaller FSMs

- Less states in each partition
  - Simpler minimization and state assignment
  - Smaller combinational gates
  - Shorter critical path

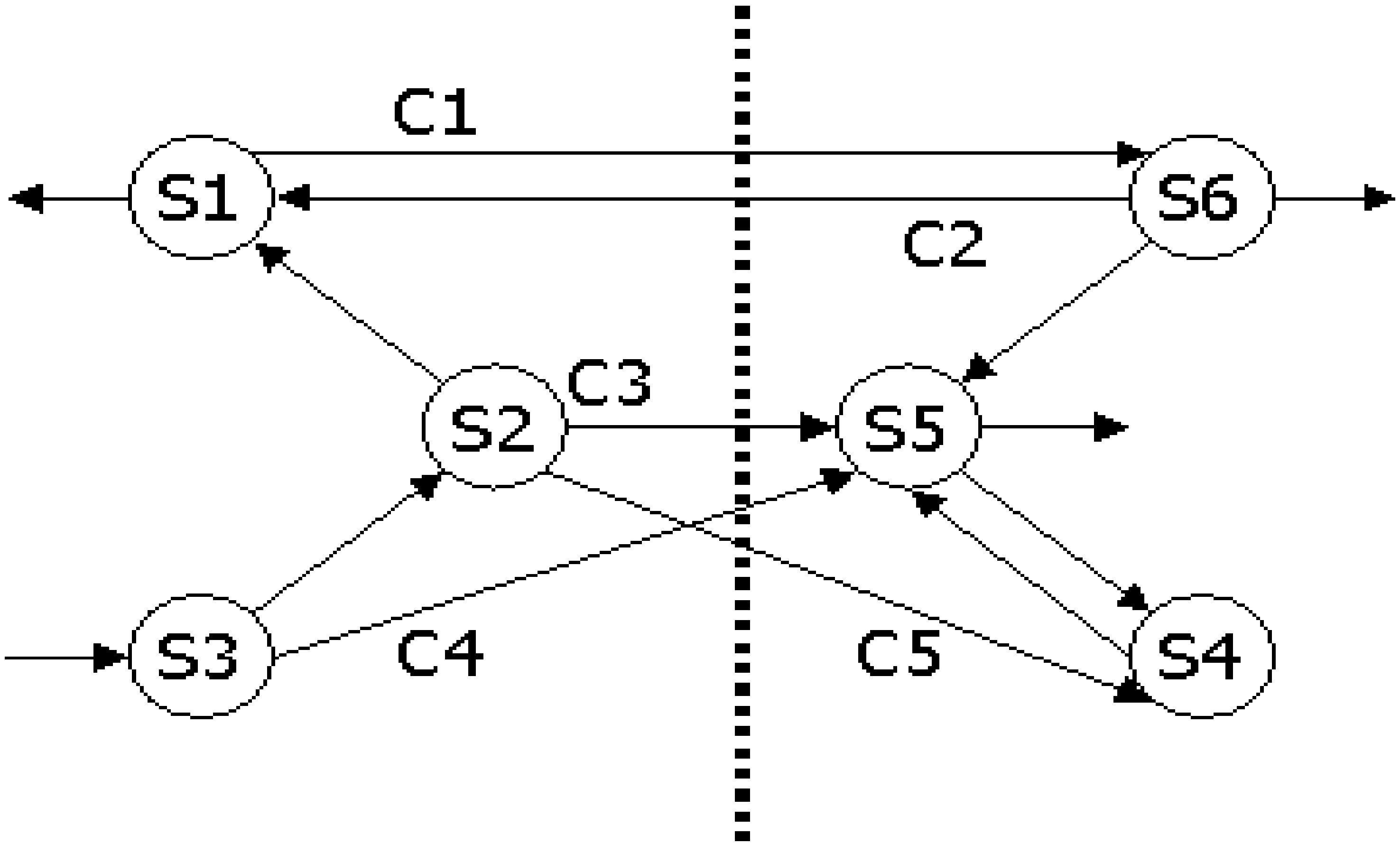
Before



After

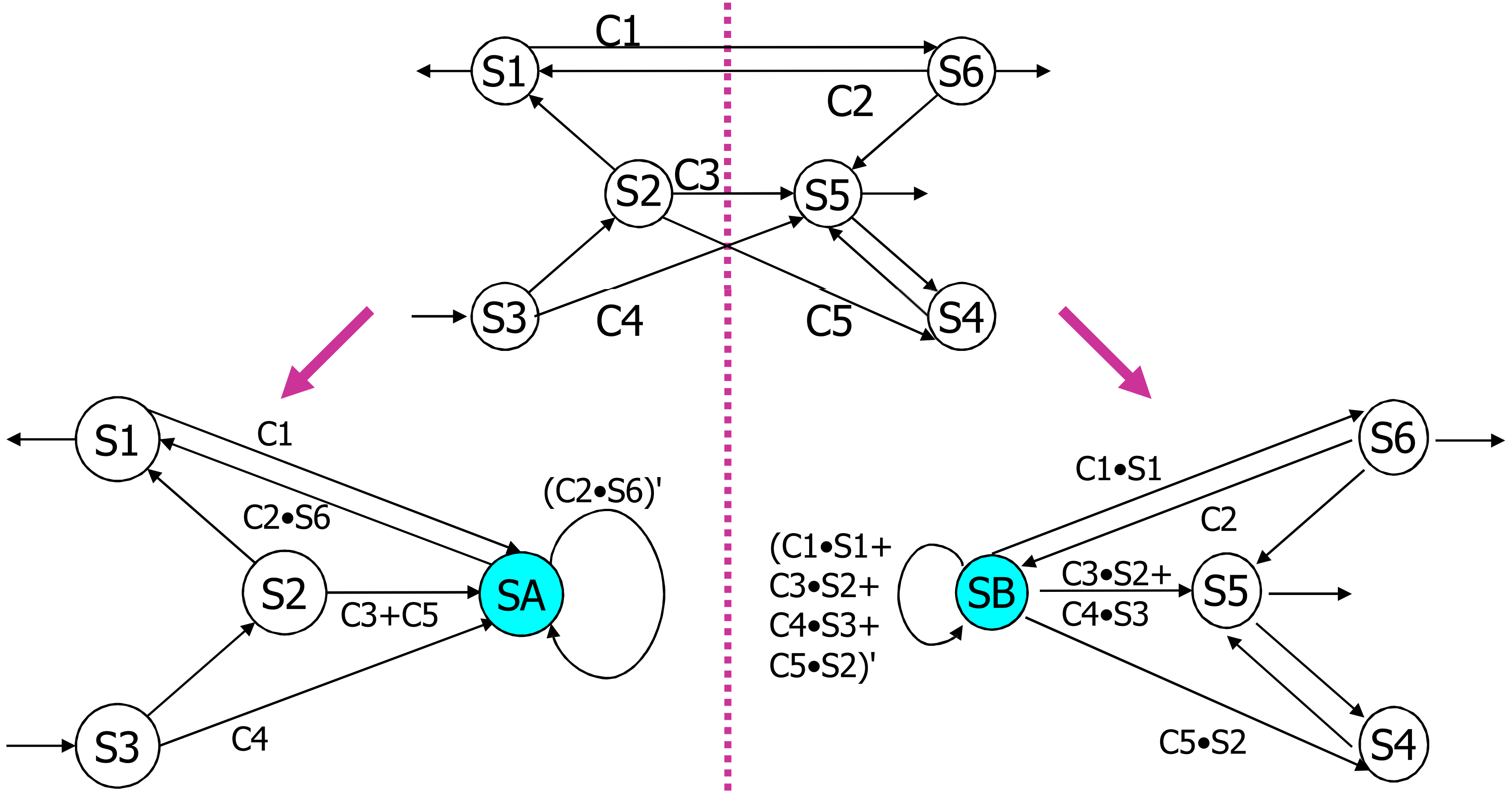


Example: Partition this machine into two halves



Introduce idle states

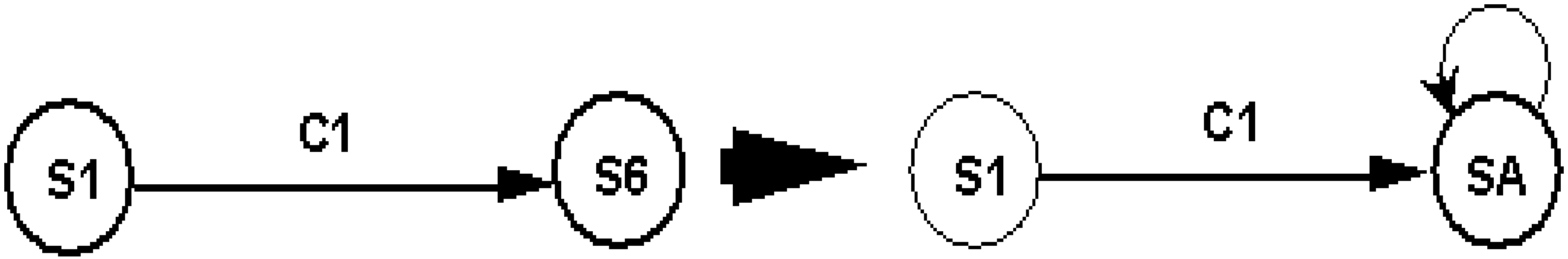
- SA and SB handoff control between machines



Partitioning rules

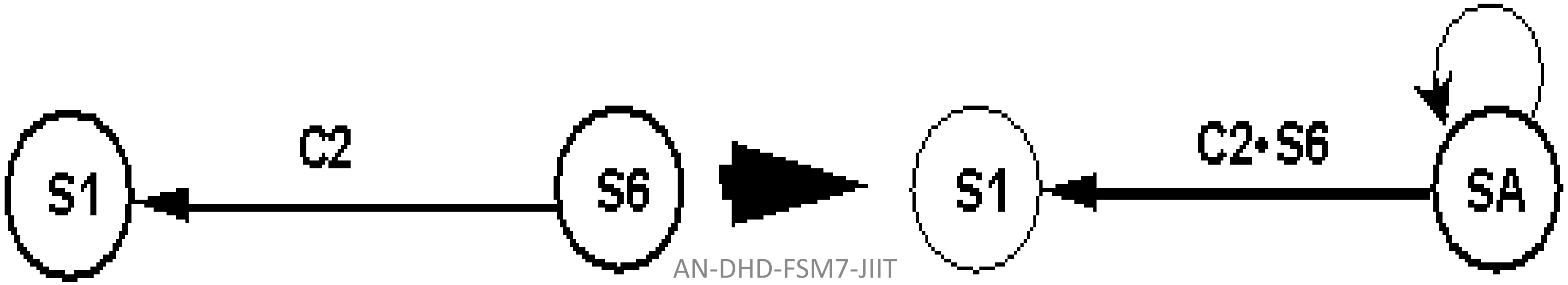
Rule #1: Source-state transformation

Replace by transition to idle state (SA)



Rule #2: Destination state transformation

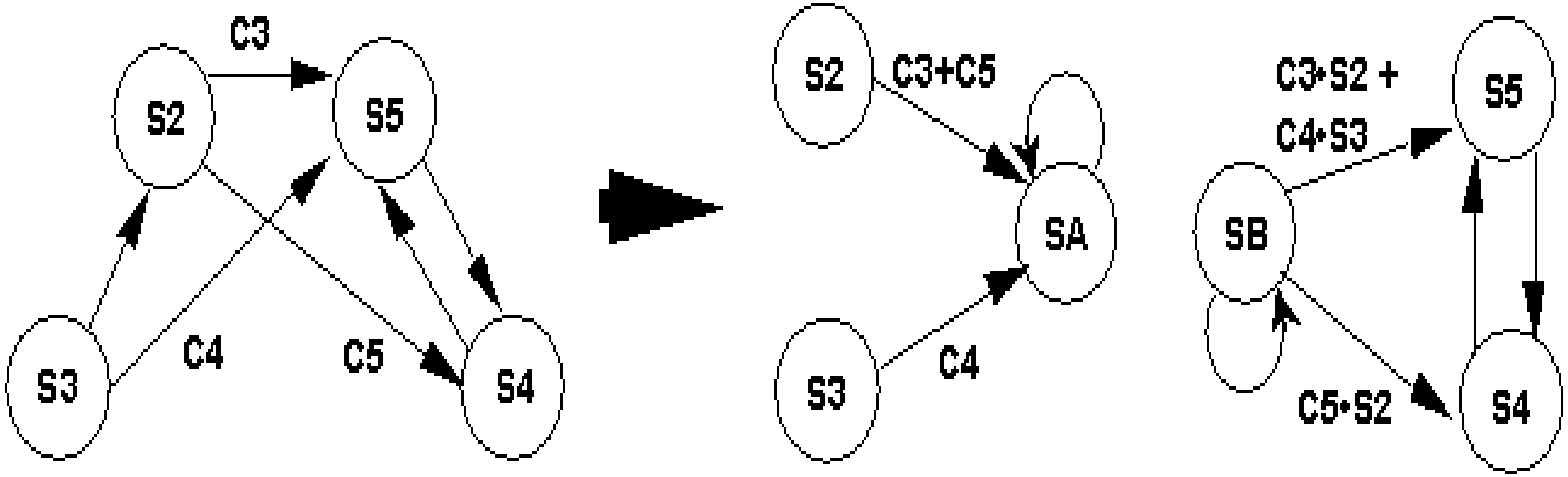
Replace with exit transition from idle state



Rule #3: Multiple transitions with same source or destination

Source $\Rightarrow$ Replace by transitions to idle state (SA)

Destination $\Rightarrow$ Replace with exit transitions from idle state

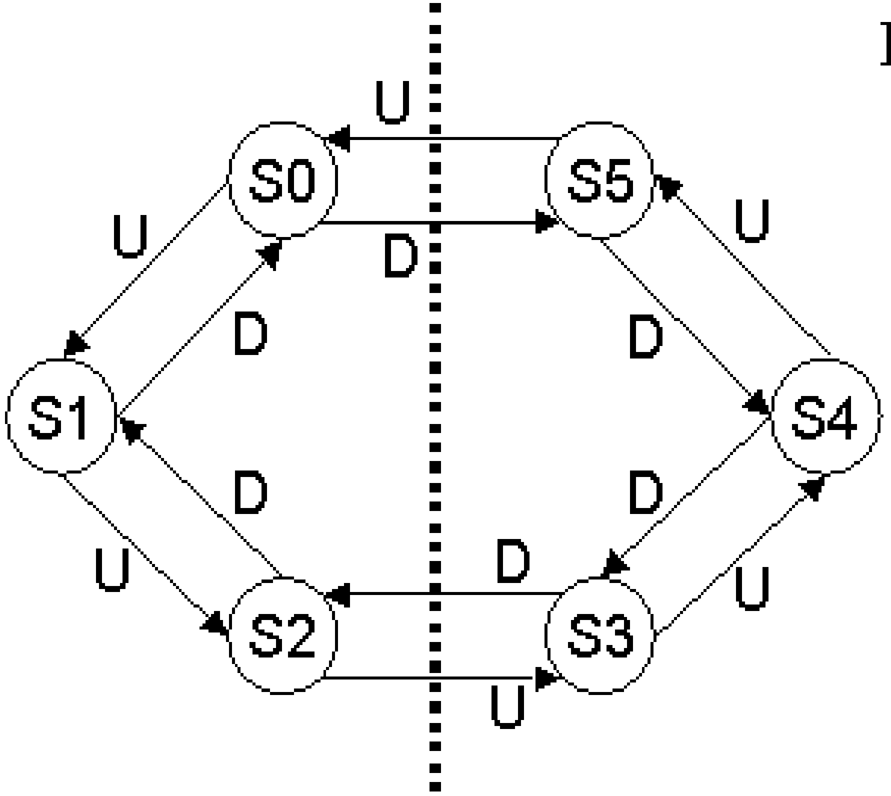


Rule #4: Hold condition for idle state

OR exit conditions and invert

Example: Six-state up/down counter

Break into 2 parts



U ≡ count up

D ≡ count down

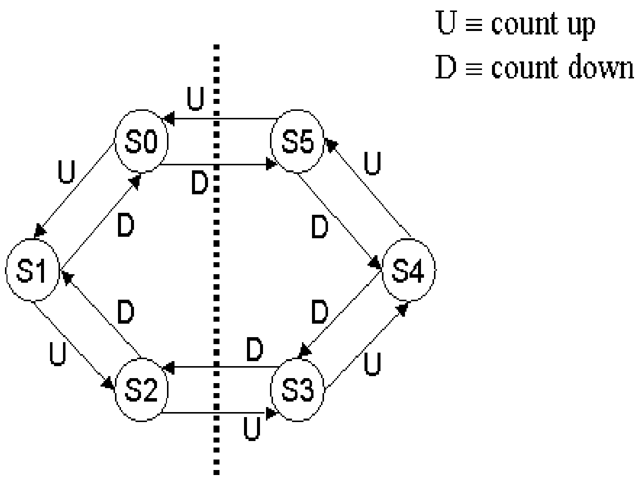
- Count sequence S_0, S_1, S_2, S_3, S_4, S_5

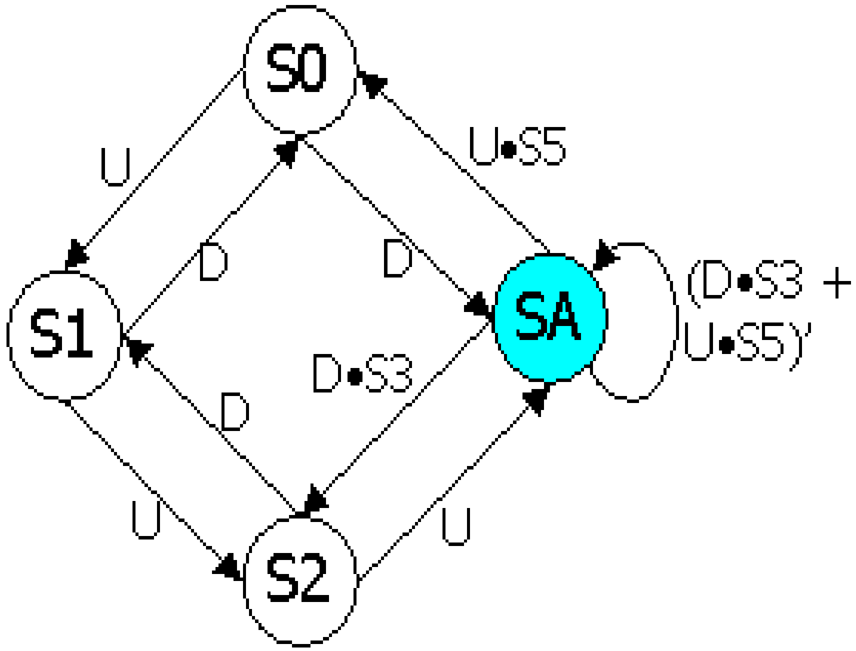
Up count

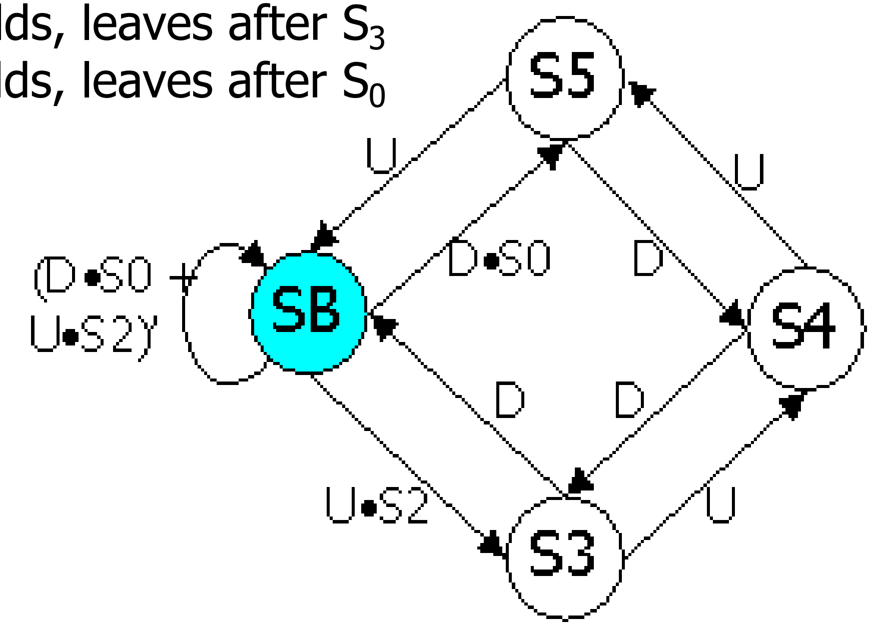
S_2 goes to S_A and holds, leaves after S_5
S_5 goes to S_B and holds, leaves after S_2

Down count

S_0 goes to S_A and holds, leaves after S_3
S_3 goes to S_B and holds, leaves after S_0

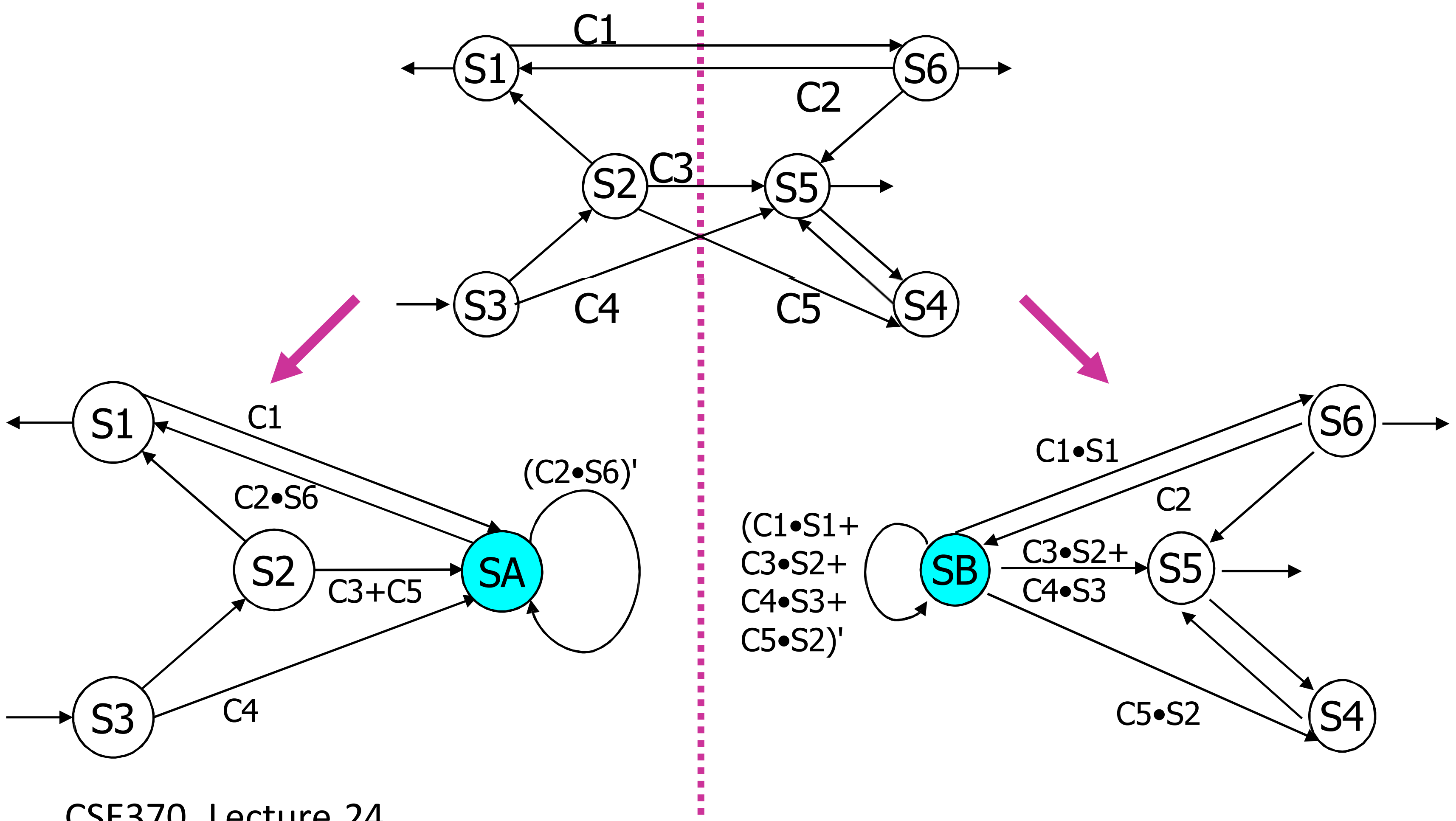




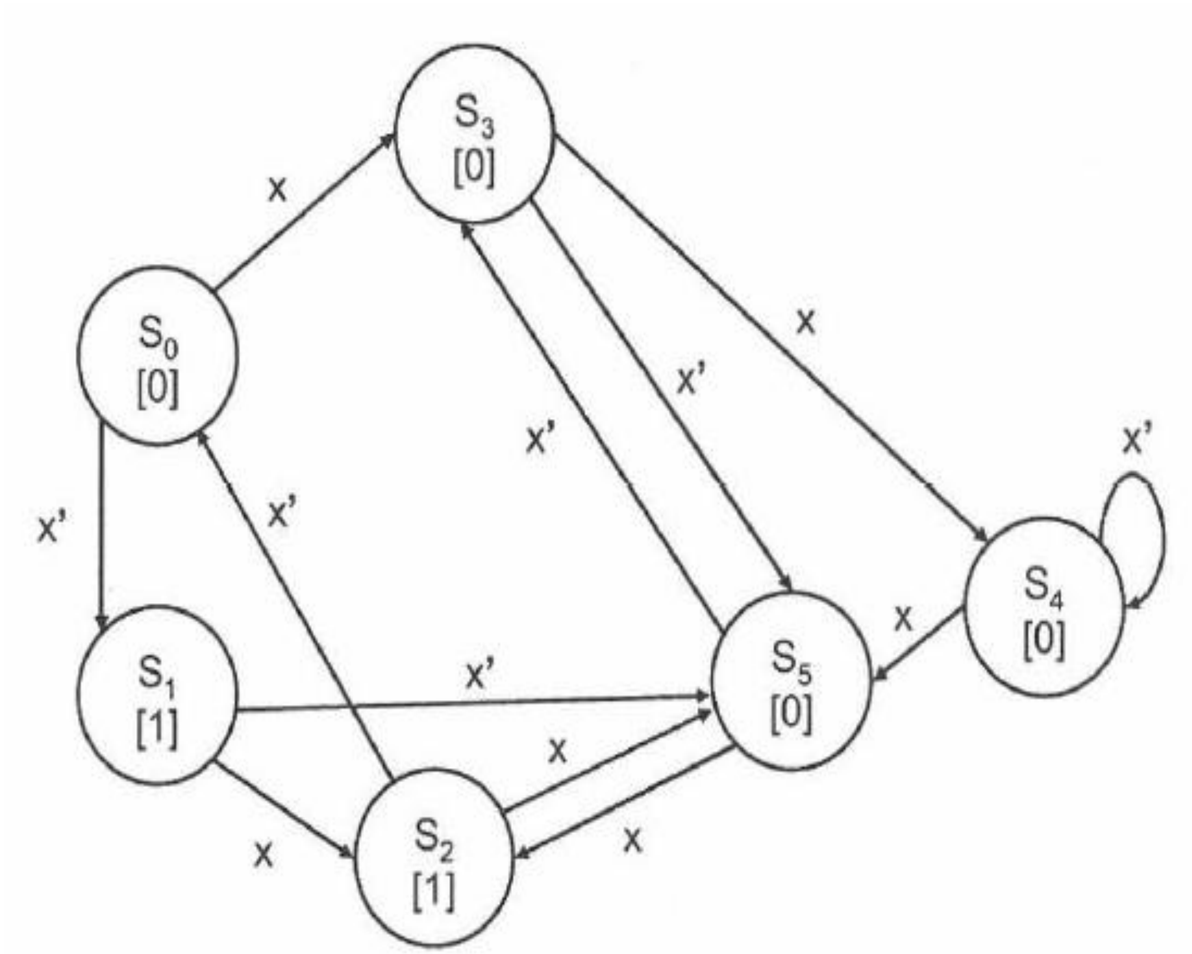


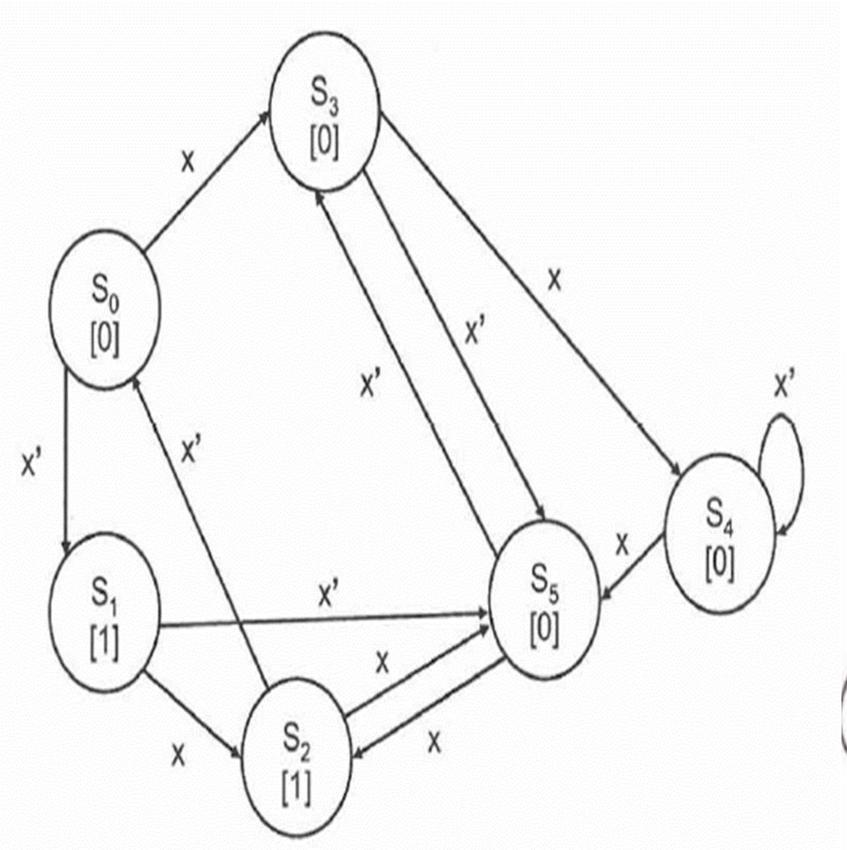
Introduce idle states

- SA and SB handoff control between machines

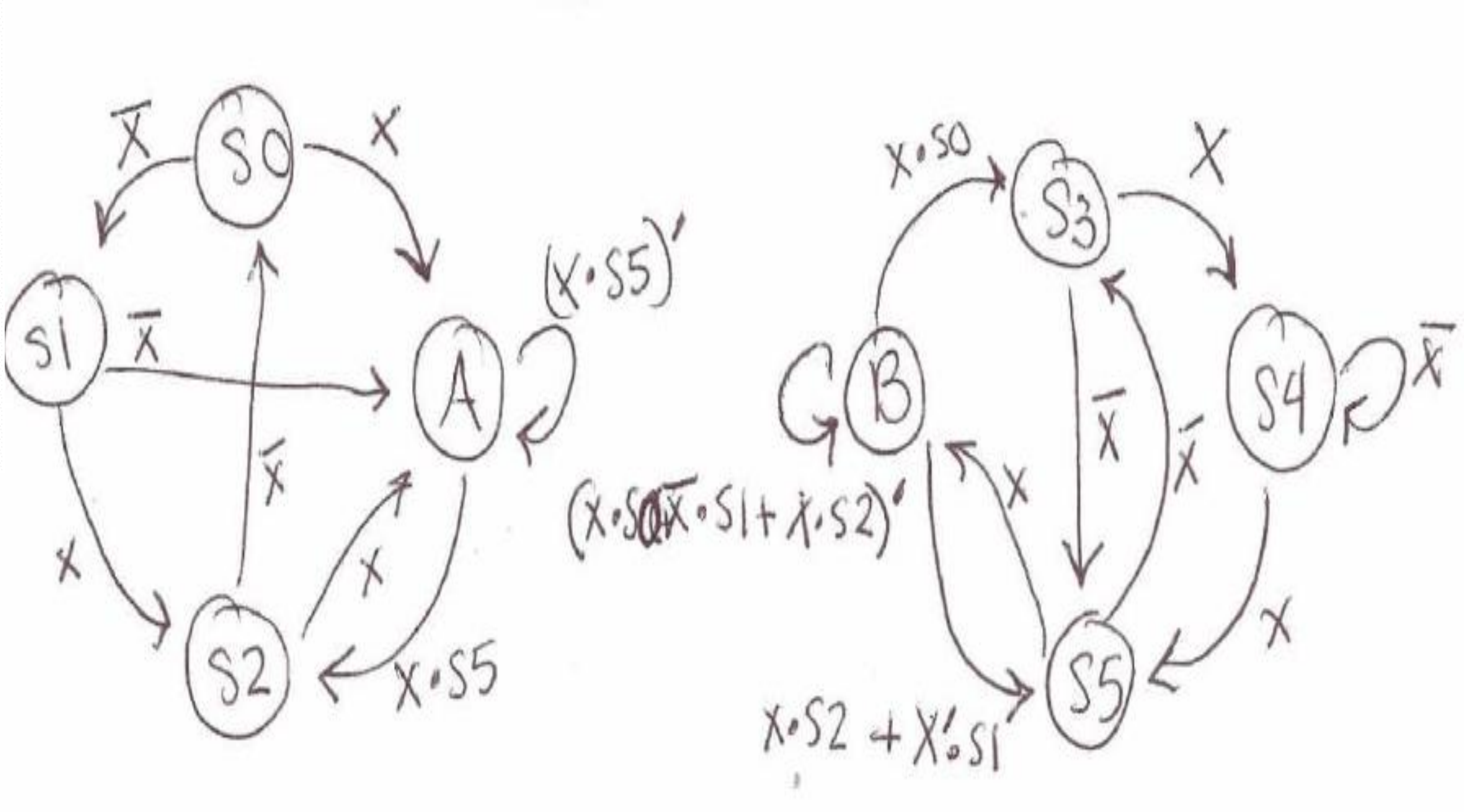


Example 1: Partition the FSMs, one covering states S_0,S_l,S_2 and the other covering states S_3,S_4,S_5. Show the labels of all transitions in your new machines.





S3 [0]
S0 [0]
S1 [1]
S2 [1]
S5 [0]
S4 [0]
x
x'



S0
S1
S2
A
B
S3
S4
S5
(X•S5)'
X•S0
X•S5
x•S2 + X'•S1