Spring 2015 Week 6 Module 31

Digital Circuits and Systems

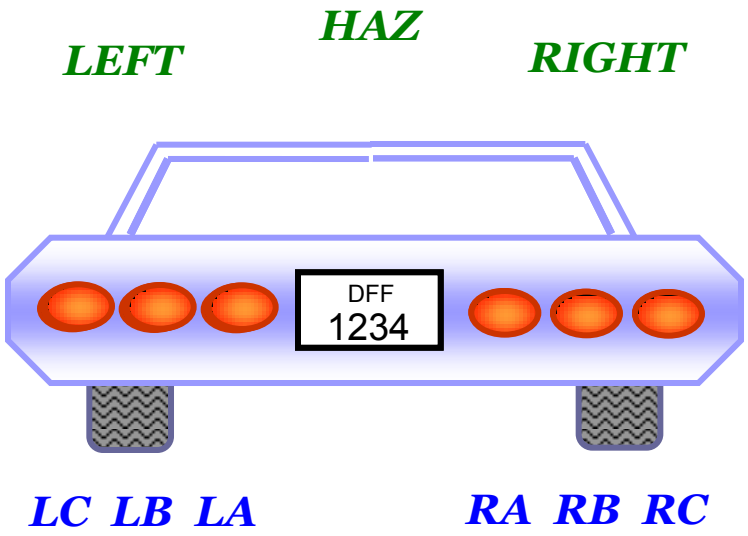
State Machines 3: State Minimization

*Shankar Balachandran**
Associate Professor, CSE Department
Indian Institute of Technology Madras

*Currently a Visiting Professor at IIT Bombay

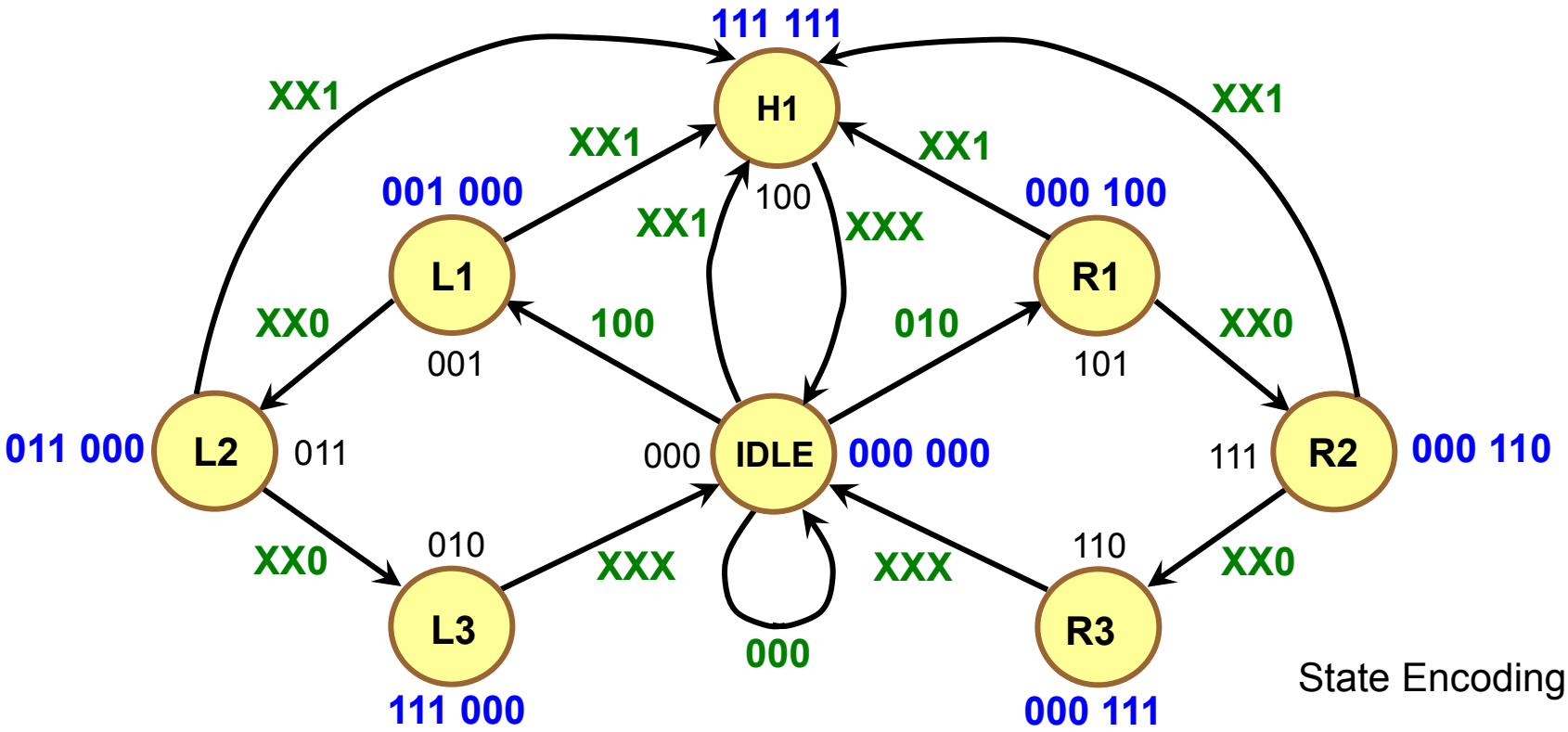
Design Example: *Tail Light Controller*

- Design a state machine to control tail lights of a car. On each side three lights are to be used for turn signaling and hazard. These are controlled by left, right and hazard signals.
  - Inputs: $LEFT\ (L),\ RIGHT\ (R),\ HAZ\ (H)$
  - Outputs: (LC, LB, LA) and (RA, RB, RC)
  - Operation:



Tail Light Controller: *State Diagram*

- Condition of each tail lamp defines a unique state
- Use Moore machine since outputs are solely determined by the state.
  - ***Input bit order***: *L R H*
  - ***Output bit order***: *LC LB LA RA RB RC*



Tail Light Controller: *State Table*

Inputs			Present State				FF Inputs			Next State				Outputs (Moore outputs)					
L	*R*	*H*	*Q2*	*Q1*	*Qo*		*D2*	*D1*	*Do*	*Q2**	*Q1**	*Qo**		*LC*	*LB*	*LA*	*RA*	*RB*	*RC*
0	0	0	0	0	0	(IDLE)	0	0	0	0	0	0	(IDLE)	0	0	0	0	0	0
1	0	0	0	0	0	(IDLE)	0	0	1	0	0	1	(L1)	0	0	0	0	0	0
0	1	0	0	0	0	(IDLE)	1	0	1	1	0	1	(R1)	0	0	0	0	0	0
X	X	1	0	0	0	(IDLE)	1	0	0	1	0	0	(H1)	0	0	0	0	0	0
X	X	0	0	0	1	(L1)	0	1	1	0	1	1	(L2)	0	0	1	0	0	0
X	X	1	0	0	1	(L1)	1	0	0	1	0	0	(H1)	0	0	1	0	0	0
X	X	0	0	1	1	(L2)	0	1	0	0	1	0	(L3)	0	1	1	0	0	0
X	X	1	0	1	1	(L2)	1	0	0	1	0	0	(H1)	0	1	1	0	0	0
X	X	X	0	1	0	(L3)	0	0	0	0	0	0	(IDLE)	1	1	1	0	0	0
X	X	0	1	0	1	(R1)	1	1	1	1	1	1	(R2)	0	0	0	1	0	0
X	X	1	1	0	1	(R1)	1	0	0	1	0	0	(H1)	0	0	0	1	0	0
X	X	0	1	1	1	(R2)	1	1	0	1	1	0	(R3)	0	0	0	1	1	0
X	X	1	1	1	1	(R2)	1	0	0	1	0	0	(H1)	0	0	0	1	1	0
X	X	X	1	1	0	(R3)	0	0	0	0	0	0	(IDLE)	0	0	0	1	1	1
X	X	X	1	0	0	(H1)	0	0	0	0	0	0	(IDLE)	1	1	1	1	1	1

Tail Light Controller: *Logic Equations*

Flip-Flop Equations:

$$D_2 = Q_2Q_0 + HQ_0 + H\overline{Q_2}\,\overline{Q_1} + \overline{L}\,R\overline{H}\,\overline{Q_2}\,\overline{Q_1}\,\overline{Q_0}$$

$$D_1 = \overline{H}\,Q_0$$

$$D_0 = \overline{H}\,\overline{Q_1}\,Q_0 + \overline{L}\,R\overline{H}\,\overline{Q_2}\,\overline{Q_1}\,\overline{Q_0} + L\overline{R}\,\overline{H}\,\overline{Q_2}\,\overline{Q_1}\,\overline{Q_0}$$

Output Equations:

$$LA = \overline{Q_2}\,Q_1 + \overline{Q_2}\,Q_0 + Q_2\overline{Q_1}\,\overline{Q_0}$$

$$LB = \overline{Q_2}\,Q_1 + Q_2\overline{Q_1}\,\overline{Q_0}$$

$$LC = \overline{Q_2}\,Q_1\overline{Q_0} + Q_2\overline{Q_1}\,\overline{Q_0}$$

$$RA = Q_2Q_1 + Q_2\,\overline{Q_0} + Q_2\overline{Q_1}\,Q_0$$

$$RB = Q_2Q_1 + Q_2\,\overline{Q_0}$$

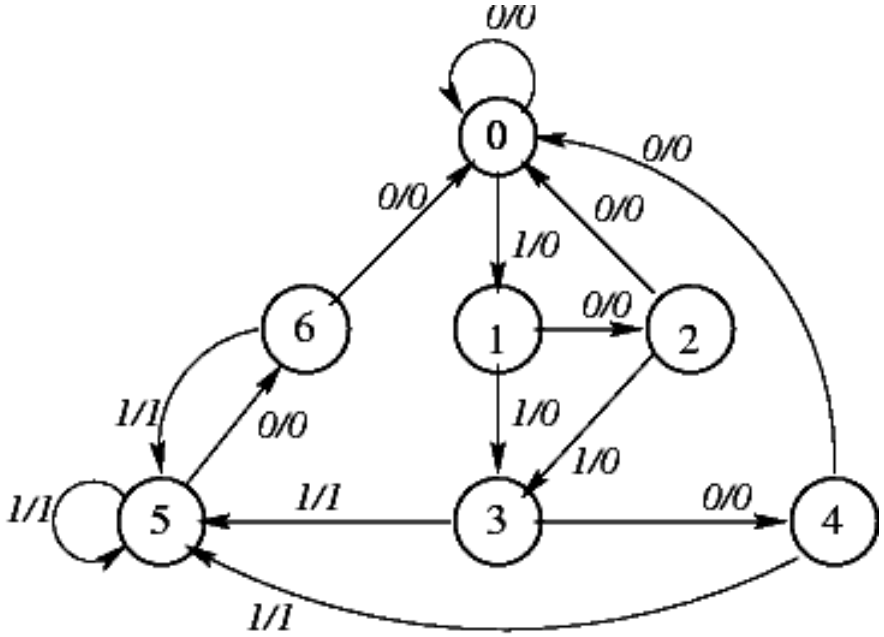
$$RC = Q_2\,\overline{Q_0}$$

State Minimization

- To reduce the cost of sequential machines, it is necessary to eliminate redundant (equivalent) states.
- State minimization is the removal of redundant states.
- Two states are said to be equivalent if for each member of the set of inputs, they:
  1. *give exactly the same output, and*
  2. *send the circuit either to the same state or to an equivalent state.*

Example

Consider the following state diagram for state minimization



Original state table

Present State	Next State		Output	
	$x = 1$	$x = 0$	$x = 1$	$x = 0$
0	1	0	0	0
1	3	2	0	0
2	3	0	0	0
3	~~5~~ → 3	4	1	0
4	~~5~~ → 3	0	1	0
~~5~~	~~5~~	~~6~~ → 4	~~1~~	~~0~~
~~6~~	~~5~~	~~0~~	~~1~~	~~0~~

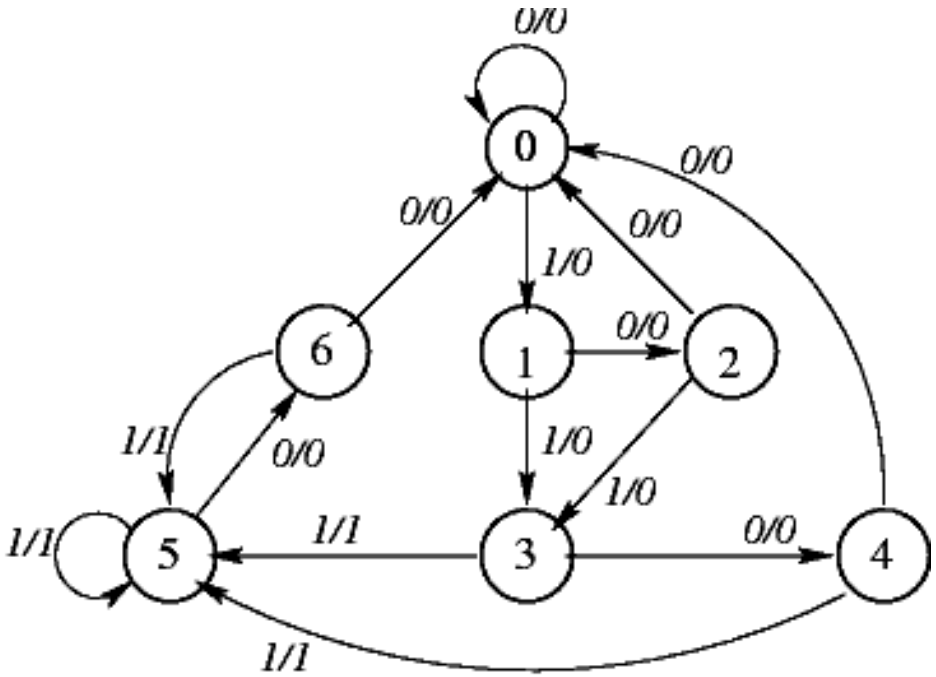
Present State	Next State		Output	
	$x = 1$	$x = 0$	$x = 1$	$x = 0$
0	1	0	0	0
1	3	2	0	0
2	3	0	0	0
3	3	4	1	0
4	3	0	1	0

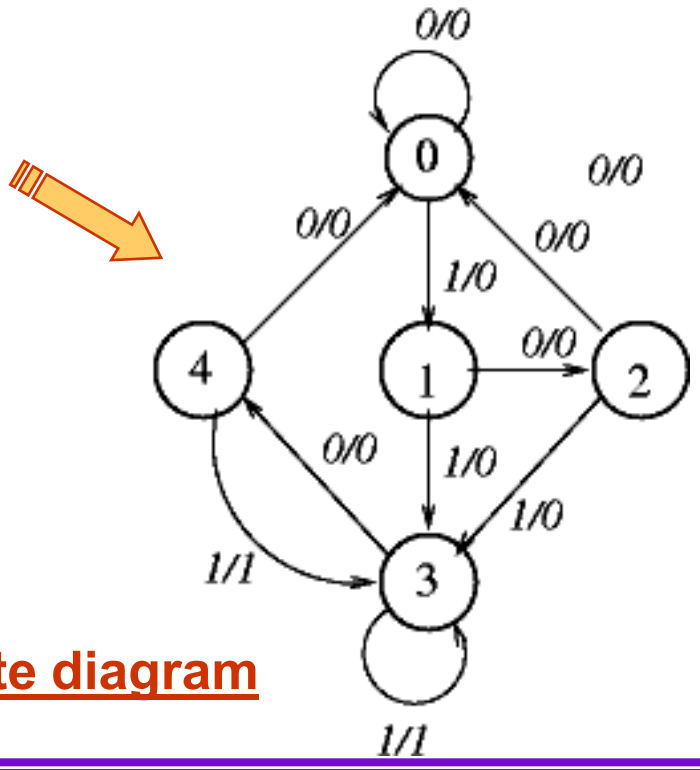
⇒ **Reduced state table**

1. Check for equivalent states.
2. States 4 and 6 are equivalent ⇨ replace state 6 by 4 everywhere.
3. Go to step 1. and check again.

Note: states 2 and 4 are not equivalent since the outputs are different.

Original state diagram





Reduced state diagram

Implication Table

- If there are 2^m states in a sequential machine we need m flip-flops
- Reduction in the number of states *may* or *may not* result in a reduction in the of flip-flops.
- Determination of equivalent states can be done using a tool called **Implication Table**. It is a more general technique compared to State Reduction by Inspection discussed earlier.

State Minimization using Implication Table

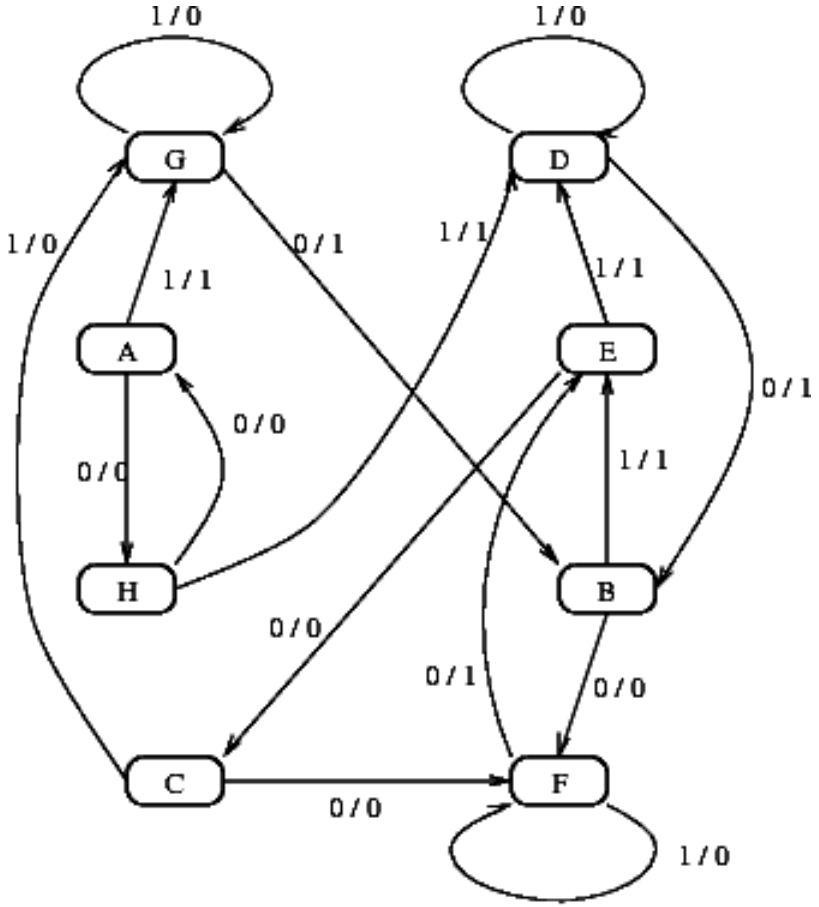
1. Using a table of present states, next states and outputs, construct an implication table as follows:

   Each state is associated with a column and a row, i.e., list all states except the first in rows and all except the last in columns. Each cell in this table corresponding to the intersection of a row and column represents two states being tested for equivalence.

2. Based on condition 1 for equivalent states place a cross in the cells corresponding to those state pairs whose outputs are not equal for every input.
3. In each remaining cell, place the pairs of next states whose equivalence is “implied” by the two states corresponding to the cell, i.e., states in each state pair must be equivalent in order for the states labeling the row and column to be equivalent.
4. Make successive passes through the entire table to determine if any more cells should be crossed off. Repeat this procedure until no additional cells can be crossed off.

Example:

Reduce the following state machine using an implication table.

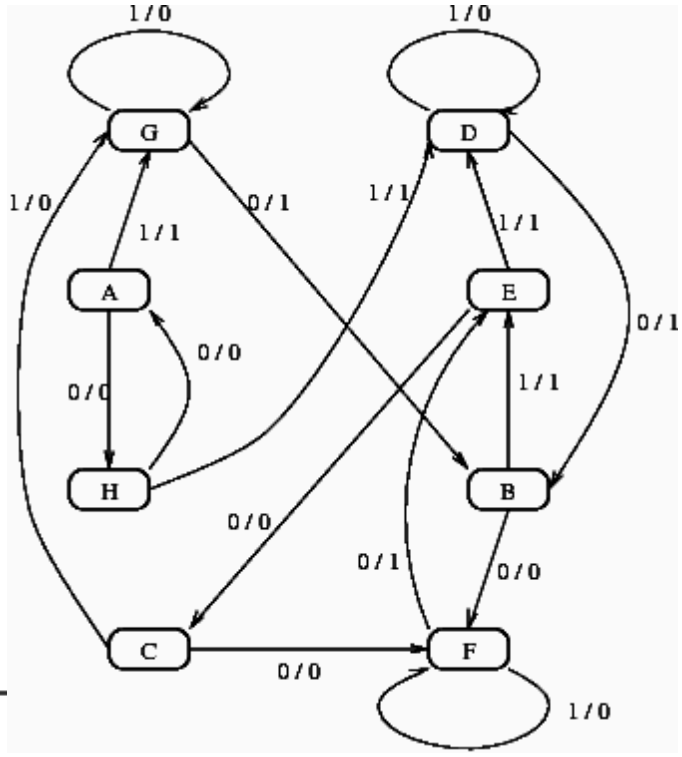


Implication Table

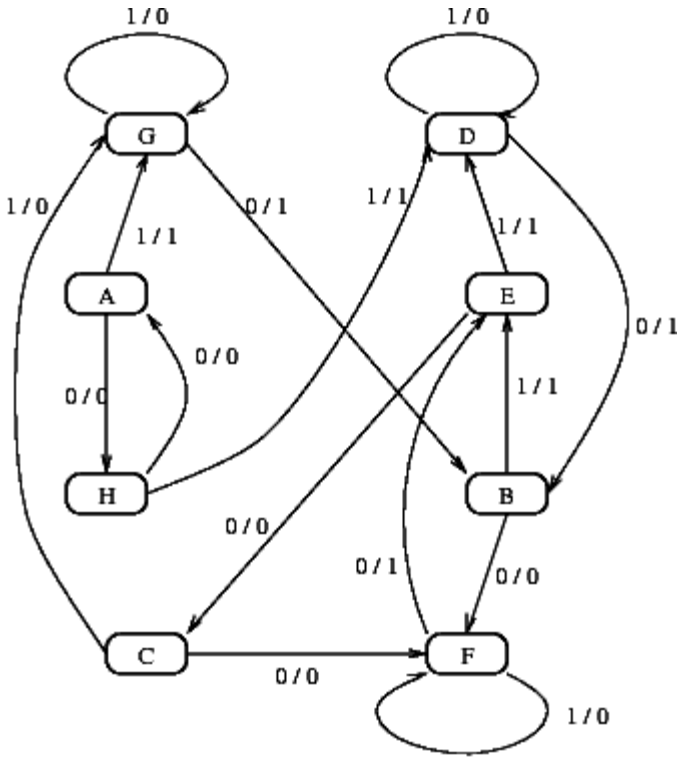
If 2 states are equivalent, their outputs must be the same and their next states must be equivalent.

	A	B	C	D	E	F	G
B	~~H–F~~ ~~G–E~~						
C	X	X					
D	X	X	X				
E	~~H–C~~ ~~G–D~~	~~F–C~~ ~~E–D~~	X	X			
F	X	X	X	~~B–E~~ **D-F**	X		
G	X	X	X	B–B D–G	X	~~E–B~~ ~~F–G~~	
H	A–H G–D	~~A–F~~ ~~E–D~~	X	X	~~A–C~~ ~~D–D~~	X	X

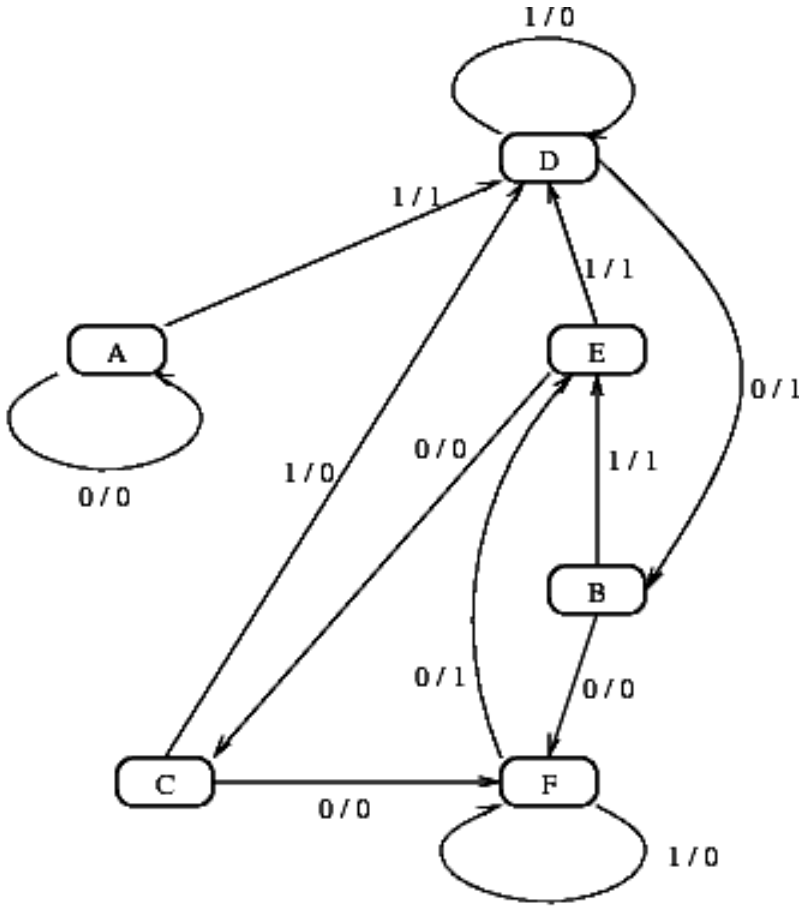
A ≡ H
D ≡ G



Reduce the state diagram removing equivalent states (A≡H and D≡G).



Original State Diagram



Reduced State Diagram

Example

Reduce the following state machine using an implication table.

Present State	Next State		Out
	$x = 1$	$x = 0$	
A	C	D	0
B	H	F	0
C	D	E	1
D	E	A	0
E	A	C	1
F	B	F	1
G	H	B	0
H	G	C	1

End of Week 6: Module 31

Thank You