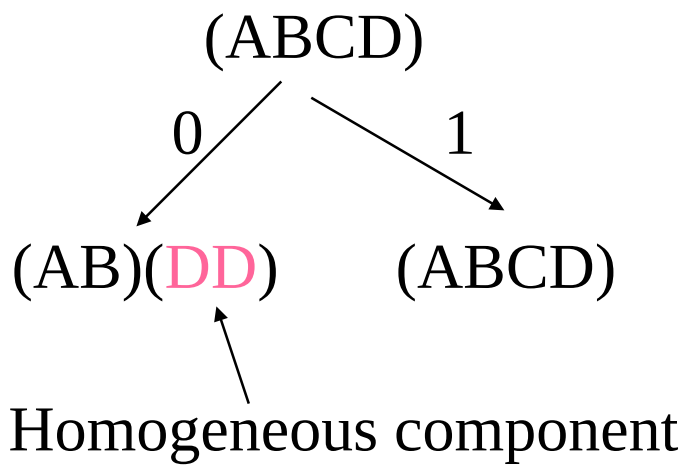
Algorithms to Generate Distinguishing, Homing and Synchronizing Sequences

State Table Verification for Sequential Circuit

Example : Consider FSM

State table

present state	input x=0	input x=1
A	B,0	D,0
B	A,0	B,0
C	D,1	A,0
D	D,1	C,0



No distinguishing sequence because DD means that two states go to D

Algorithm to Generate a Distinguishing Sequence

Distinguishing sequence - path from root to a trivial vector.

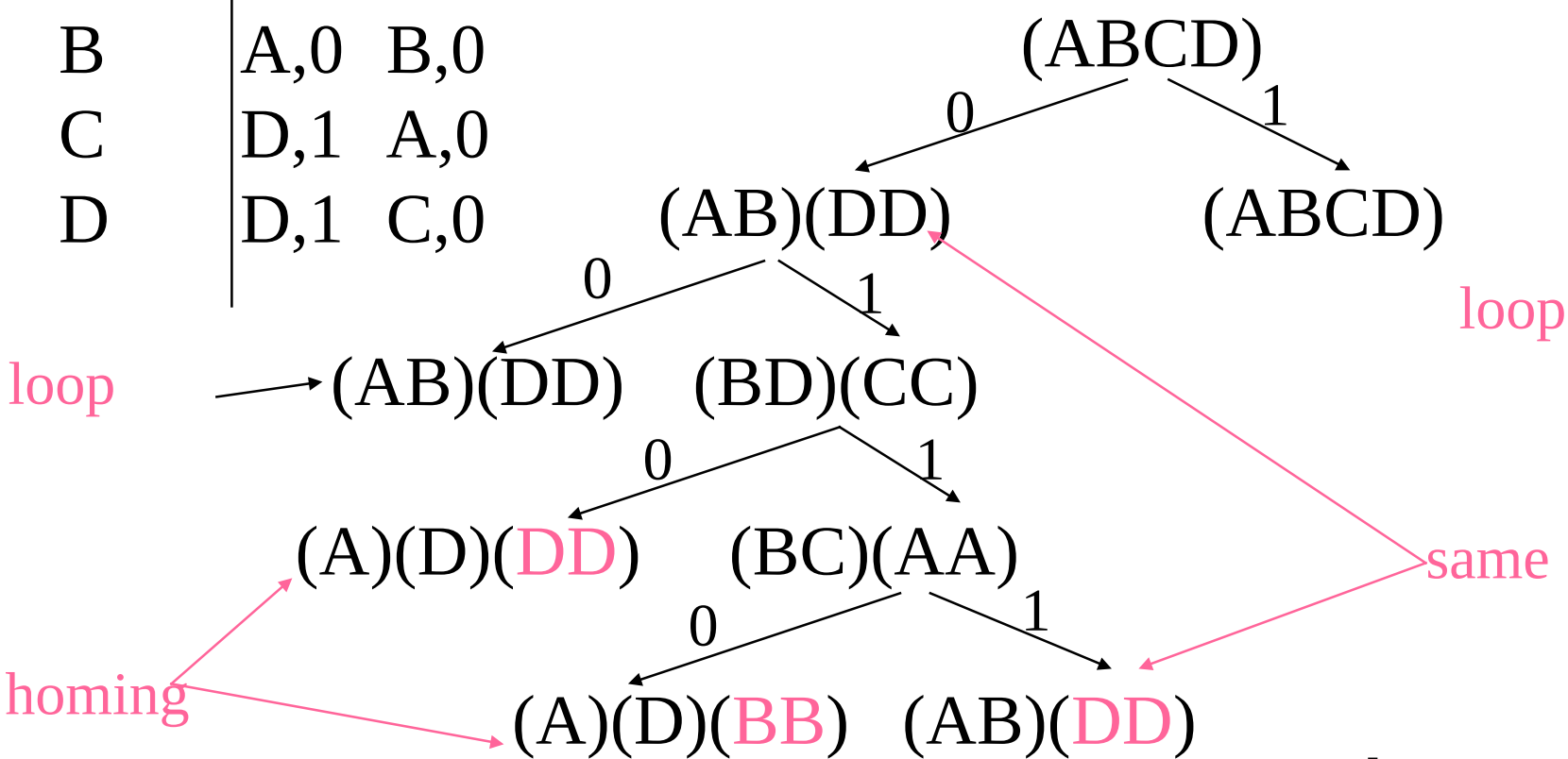
A distinguishing tree is a successor tree in which a node becomes terminal if

1. Non-homogenous components in an uncertainty vector are the same as on the previous level
2. Uncertainty vector contains a homogeneous non-trivial component (does not have to be a homogeneous vector)
3. Uncertainty vector is trivial

Example : continuation of the same FSM

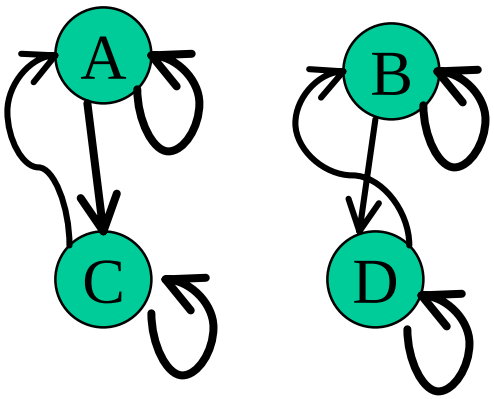
State table

present state	input x=0	input x=1
A	B,0	D,0
B	A,0	B,0
C	D,1	A,0
D	D,1	C,0



No distinguishing sequence

Example : Consider FSM, different output vectors for different initial state



State table

present state	input x=0	input x=1
A	A,0	C,1
B	B,0	D,1
C	A,1	C,0
D	D,0	B,0

Each input state responds to 10 with different output sequence

Input sequence X = 1,0

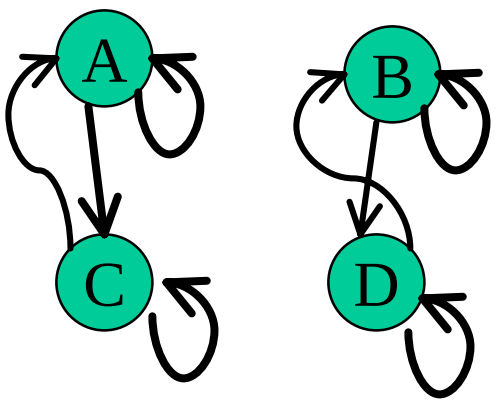
initial state	input x=1	input x=0
A	C,1	A,1
B	D,1	D,0
C	C,0	A,1
D	B,0	B,0

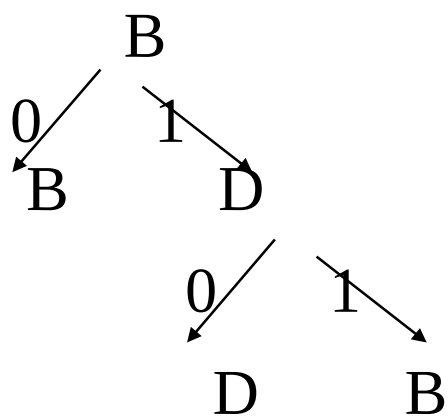
so X=1,0 distinguishing

State Table Verification for Sequential Circuit

Transfer sequence - takes machine from one state to another

Example : Consider previous FSM





We cannot reach A or C

Not strongly connected FSM

State Table Verification for Sequential Circuit

Example : Consider the following FSM

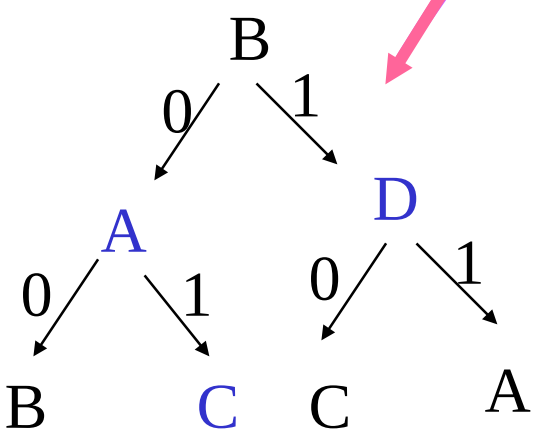
State table

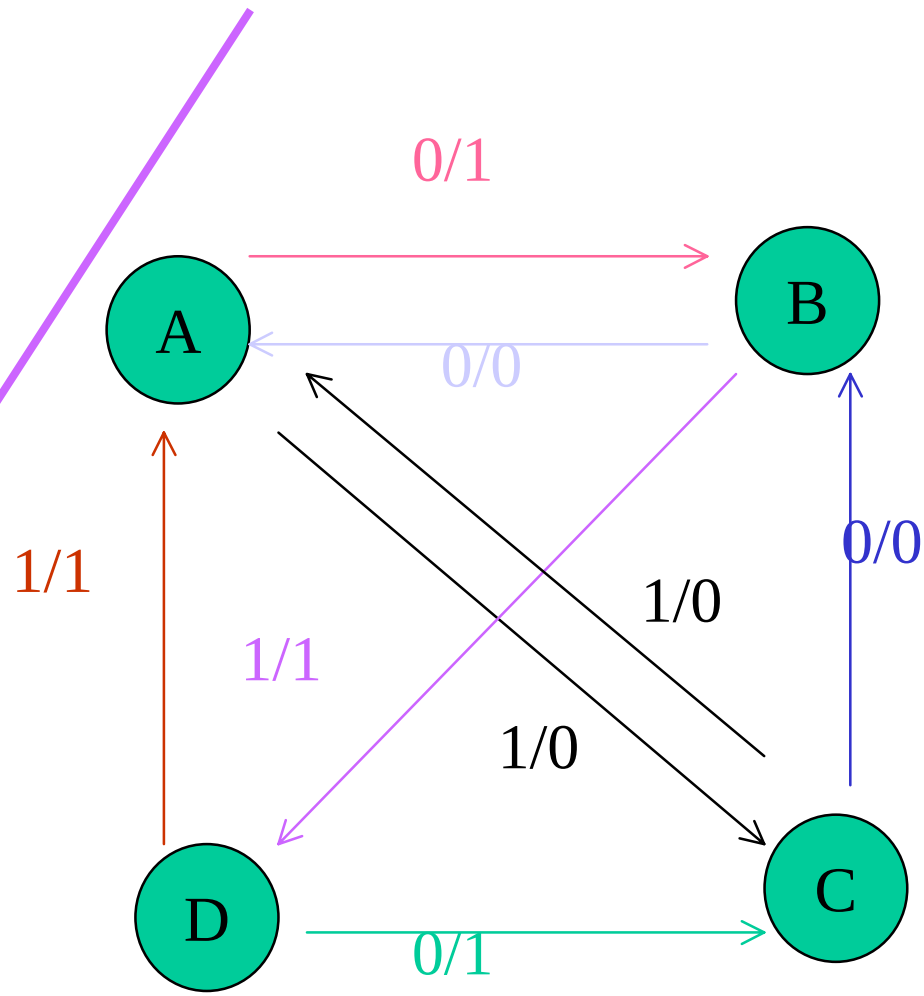
present state	input x=0	input x=1
A	B,1	C,0
B	A,0	D,1
C	B,0	A,0
D	C,1	A,1

Example : we get the transfer tree

State table

present state	input x=0	input x=1
A	B,1	C,0
B	A,0	D,1
C	B,0	A,0
D	C,1	A,1





To get to C we can select x = 1,0

State Table Verification for Sequential Circuit

Synchronizing sequence takes machine to the specific final state regardless of the output or initial state - does not always exists

Example :

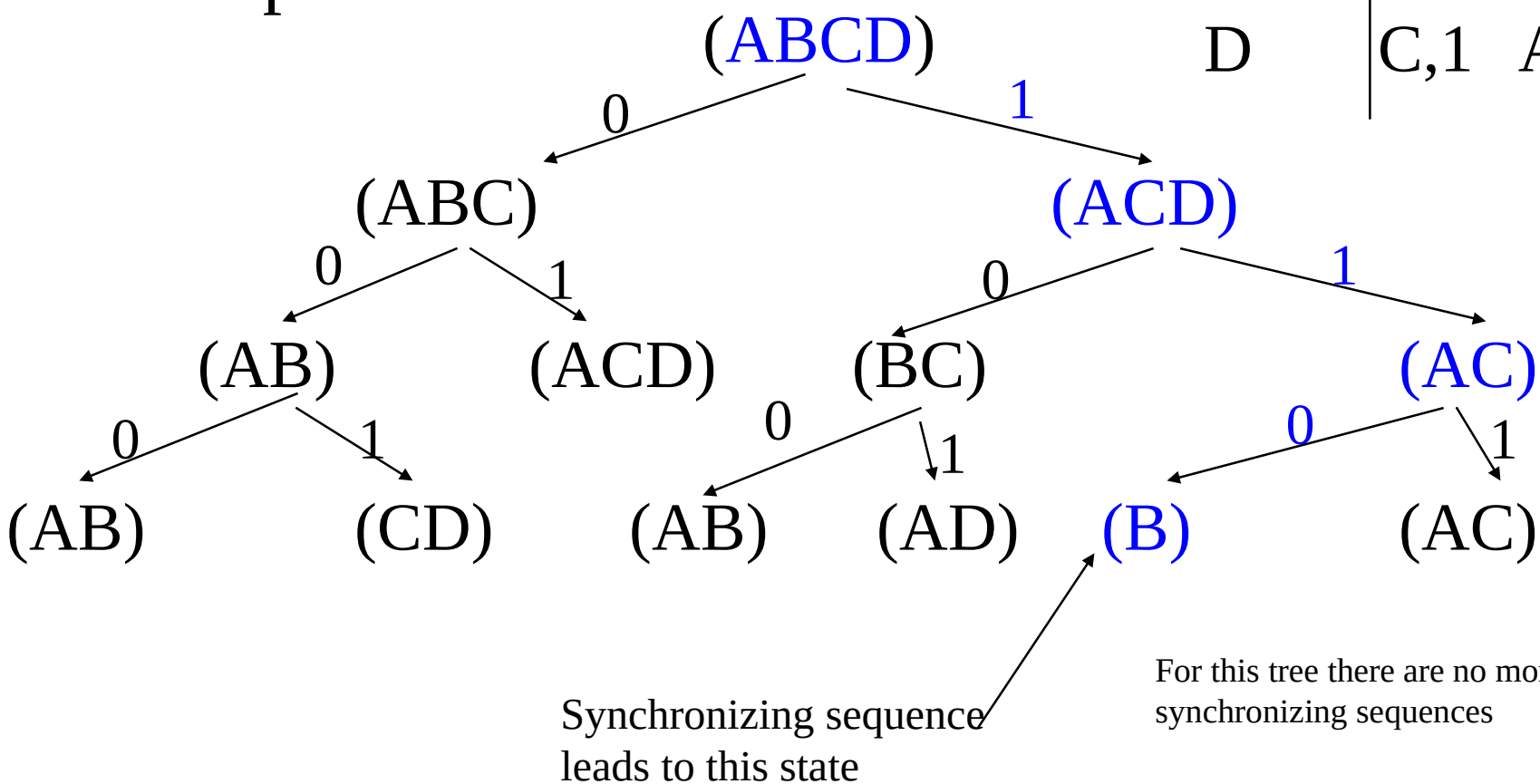
Algorithm to generate synchronizing sequence :
Consider the previous machine with synchronizing sequence X= 1,1,0

Synchronizing sequence

State table

present state	input x=0	x=1
A	B,1	C,0
B	A,0	D,1
C	B,0	A,0
D	C,1	A,1

Example :



Designing Checking Experiments

Machine must be strongly connected & diagnosable (i.e. *have a distinguishing sequence*)

1. Initialization (take it to a fixed state[s])
   a) Apply homing sequence & identify the current state
   b) Transfer current state to S
2. Identification (make machine to visit each state and display response)
3. Transition verification (make every state transition result checked by distinguishing sequence)

Designing Checking Experiments

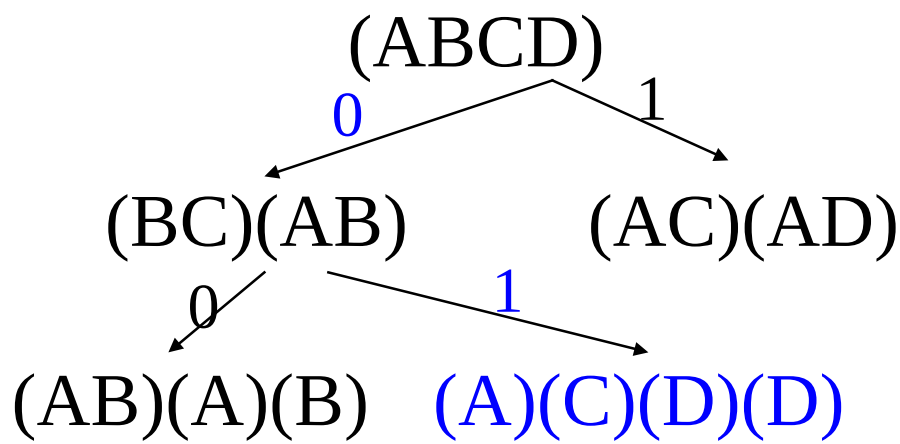
Example : Consider FSM

State table

present table	input x=0	x=1
A	B,1	C,0
B	A,0	D,1
C	B,0	A,0
D	C,1	A,1

1. Initialization:

Successor tree



Homing sequence x = 0,1

Designing Checking Experiments

Example (cont.) :

State table

present table	input x=0	x=1
A	B,1	C,0
B	A,0	D,1
C	B,0	A,0
D	C,1	A,1

Response Table

Initial states	Response to 0	1	final states
A	B,1	D,1	D
B	A,0	C,0	C
C	B,0	D,1	D
D	C,1	A,0	A

Designing Checking Experiments

take it to a fixed state

2. Identification:

recall

Analyze the results

D 01 A Generates 10 on output

time	1	2	3	4	5	6	7	8	9	10	11
input	0	1	0	1	0	0	1	0	1	0	1
state	A		D		A	B		C		D	A
output	1	1	1	0	1	0	0	0	1	1	0

Designing Checking Experiments

3. Transition verification:

Check transition from A to B with input 0, then apply distinguishing sequence 01

time	1	2	3
input	0	0	1
state	A	B	C
output	1	0	0

Designing Checking Experiments

Example : Check transition from C to B with input 0 and from C to A with input 1, and so on. The entire checking test

time	1	2	3	4	5	6	7	8	9	10	11
input	0	0	1	0	0	1	1	0	1	0	0
state	A → B			C → B			C → A			D → C	
output	1	0	0	0	0	0	0	1	1	1	0

Designing Checking Experiments (cont)

time	12	13	14	15	16	17	18	19	20	21
input	1	1	0	1	0	1	0	0	0	1
state	D → A			D			B → A			
output	1	1	1	1	1	0	1	0	1	1

time	22	23	24	25	26	27	28	29	30	31
input	1	1	0	1	0	1	0	1	0	1
state	D	A → C		D			B → D			A
output	1	0	0	1	1	0	1	1	1	0

DFT for Sequential Circuits

Critical testability problems

1. Noninitializable design - change design to have a synchronizing sequence
2. Effects of component delays - check for hazard & races in simulation
3. Nondetected logic redundant faults - do not use logic redundancy
4. Existence of illegal states – avoid; add transition to normal states
5. Oscillating circuit - add extra logic to control oscillations

DFT for Sequential Circuits

Checking experiments can not be applied for FSM without a distinguishing sequence

Modification procedure for such FSM:

I. Construct testing table

upper part contains states & input/output pairs,

lower part contains products of present & next states with the rule that (state)*(-) = (-)

II. Construct testing graph

DFT for Sequential Circuits

Example:

state table

present state	input x=0	input x=1
A	A,0	B,0
B	A,0	C,0
C	A,1	D,0
D	A,1	A,0

Testing table for machine

present state	input/output 0/0	0/1	1/0	1/1
A	A	-	B	-
B	A	-	C	-
C	-	A	D	-
D	-	A	A	-
AB	AA	-	BC	-
AC	-	-	BD	-
AD	-	-	AB	-
BC	-	-	CD	-
BD	-	-	AC	-
CD	-	AA	AD	-

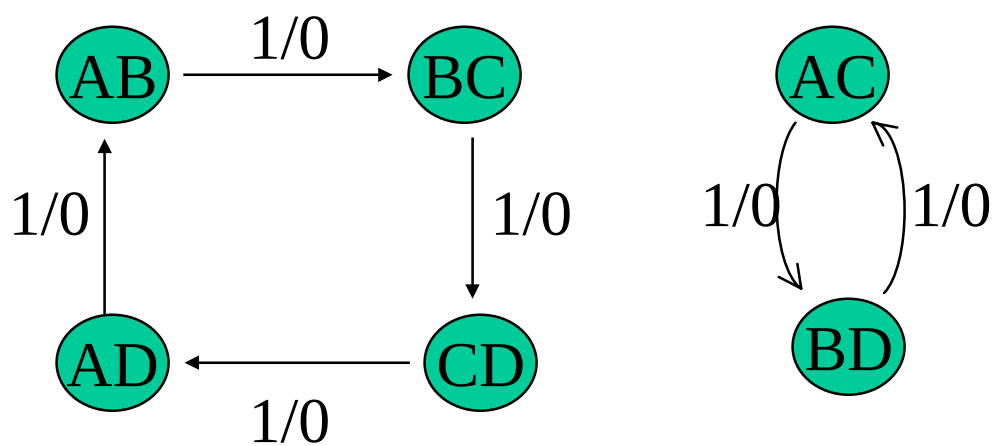
Example (continued)

present state	input x=0	x=1
A	A,0	B,0
B	A,0	C,0
C	A,1	D,0
D	A,1	A,0

II . Construct testing graph

An edge X_p/Z_p exists directed from present state S_iS_j to next states S_kS_l if S_kS_l $(k \neq l)$ is present in row S_iS_j under X_p/Z_p

Example:
for our machine
we have:



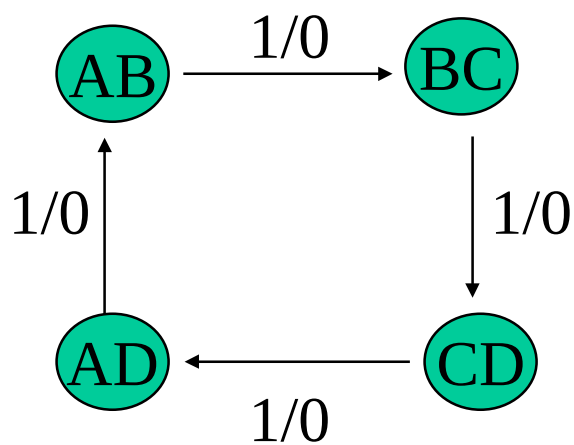
Example (continued)

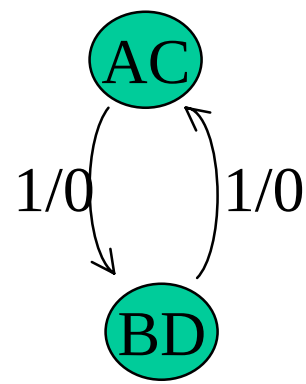
present state	input x=0	x=1
A	A,0	B,0
B	A,0	C,0
C	A,1	D,0
D	A,1	A,0

Now we can modify the graph by adding output(s)

First introduce new concept of definitely diagnosable

Example:
for our machine we have:





DFT for Sequential Circuits (cont)

A machine is definitely diagnosable if its testing graph has *no loops* and there are ***no repeated states*** (i.e. no circled states in testing table)

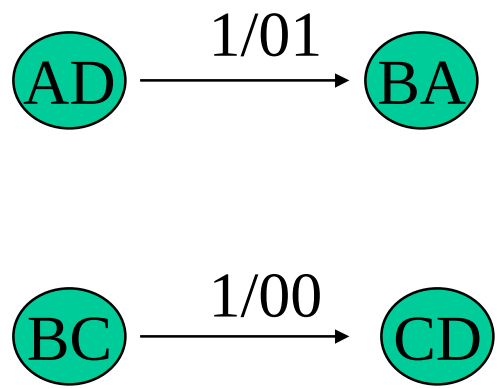
In order to make machine ***definitely diagnosable*** *additional outputs* (up to k = log (# states)) are ***required***

DFT for Sequential Circuits

Coming back to our Example: (with added output)

present state	input x=0	input x=1
A	A,00	B,01
B	A,01	C,00
C	A,10	D,00
D	A,11	A,01

Testing graph



Now the machine has a distinguishing sequence

After machine is ***modified to have distinguishing sequence*** apply checking experiment procedure to test it.

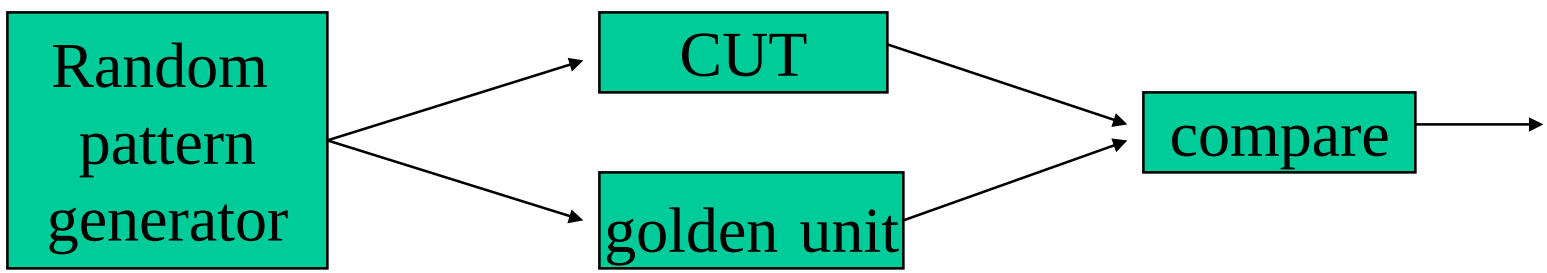
Random Testing

Reduce computation time

1. Random sequence is stored as a test when it can detect fault, then this fault is detected from the fault list and another random sequence is checked
2. Output of CUT (circuit under test) is compared with this “golden unit”

Is it good for FSMs?

Two approaches :



Sources

- Starzyk, Ohio University