COL215 Digital Logic and System Design

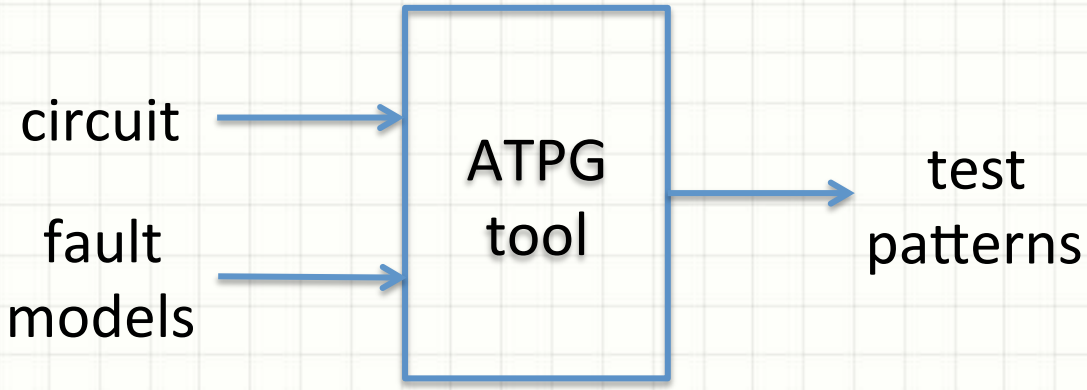
Testing of Digital Systems

15 November 2017

ATPG

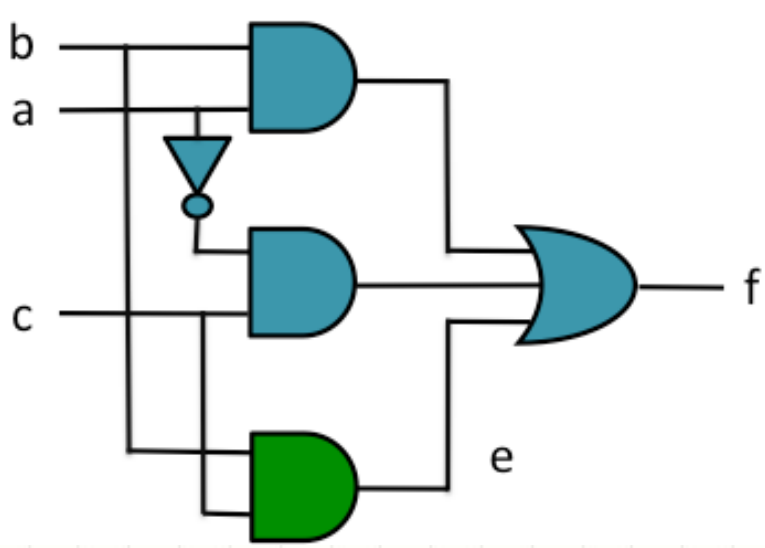
Automatic test pattern generation

Given a circuit and fault models, determine the test patterns



- Boolean differences
- D-algorithm

Applying Boolean Differences



Consider stuck-at fault at e

$f = Z(a,b,c,e) = a.b + a'.c + e$

$dZ/de = Z(a,b,c,e)|_{e=0} \oplus Z(a,b,c,e)|_{e=1}$

$= (a.b + a'.c + 0) \oplus (a.b + a'.c + 1)$

$= (a.b + a'.c)' = (a' + b') . (a + c')$

$= a.b' + a'.c' + b'.c'$

Test vector for e/0 satisfies –

$dZ/de . e = (a.b' + a'.c' + b'.c') . e$

$= (a.b' + a'.c' + b'.c') . (b.c) = 0$

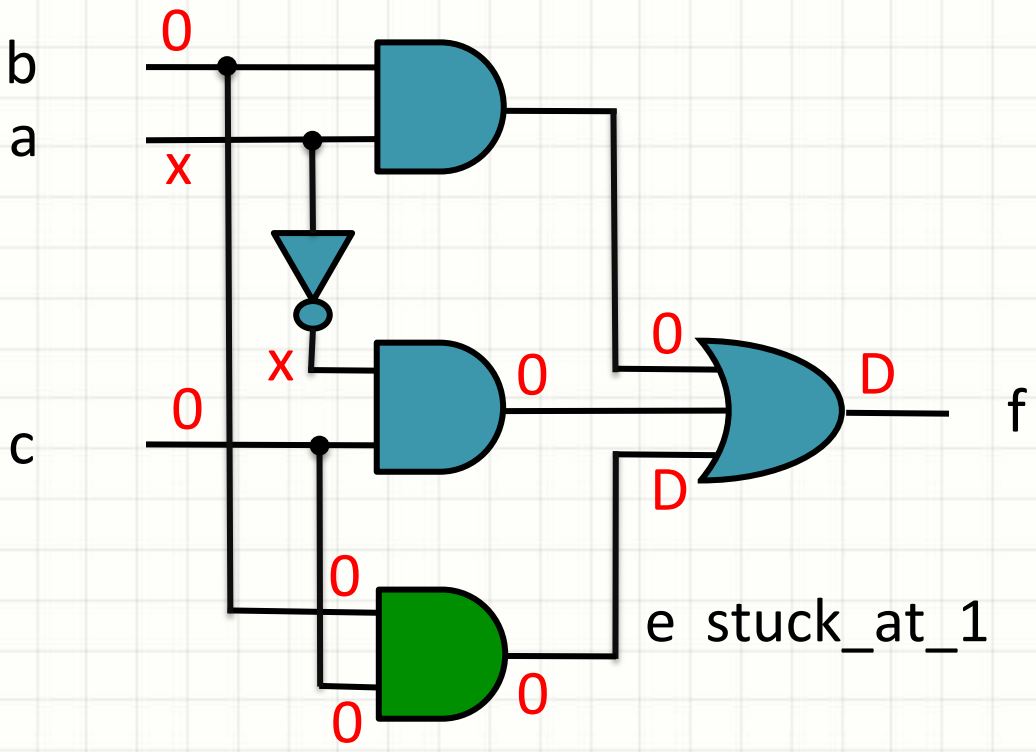
Test vector for e/1 satisfies –

$dZ/de . e' = (a.b' + a'.c' + b'.c') . e'$

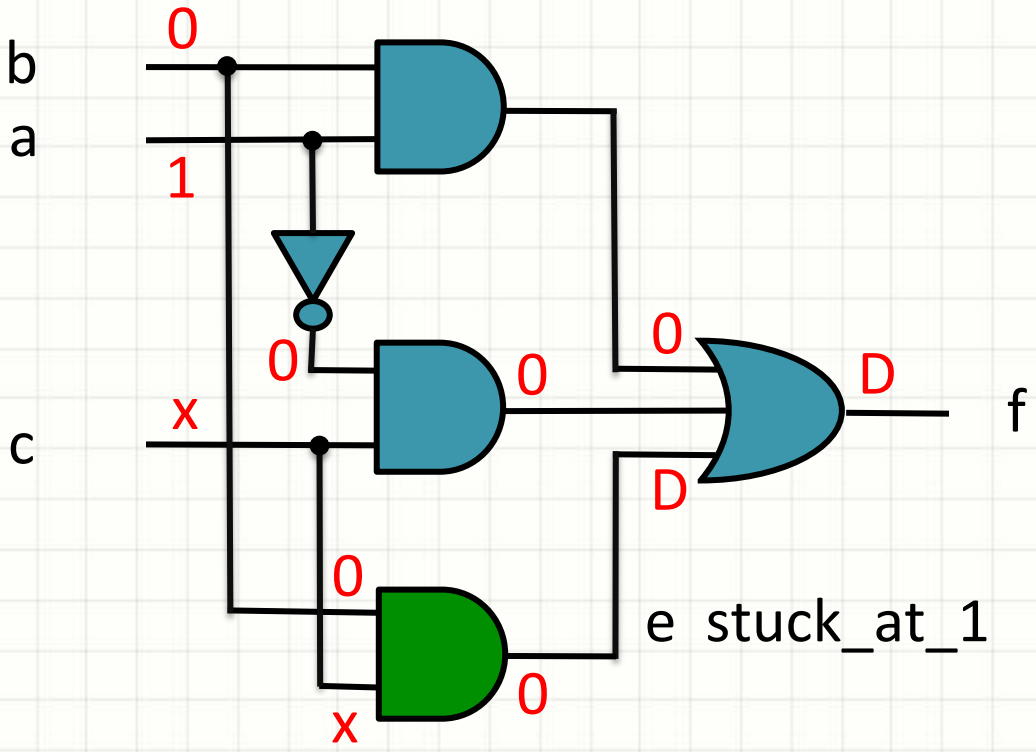
$= (a.b' + a'.c' + b'.c') . (b'+c')$

$= a.b' + a'.c' + b'.c'$

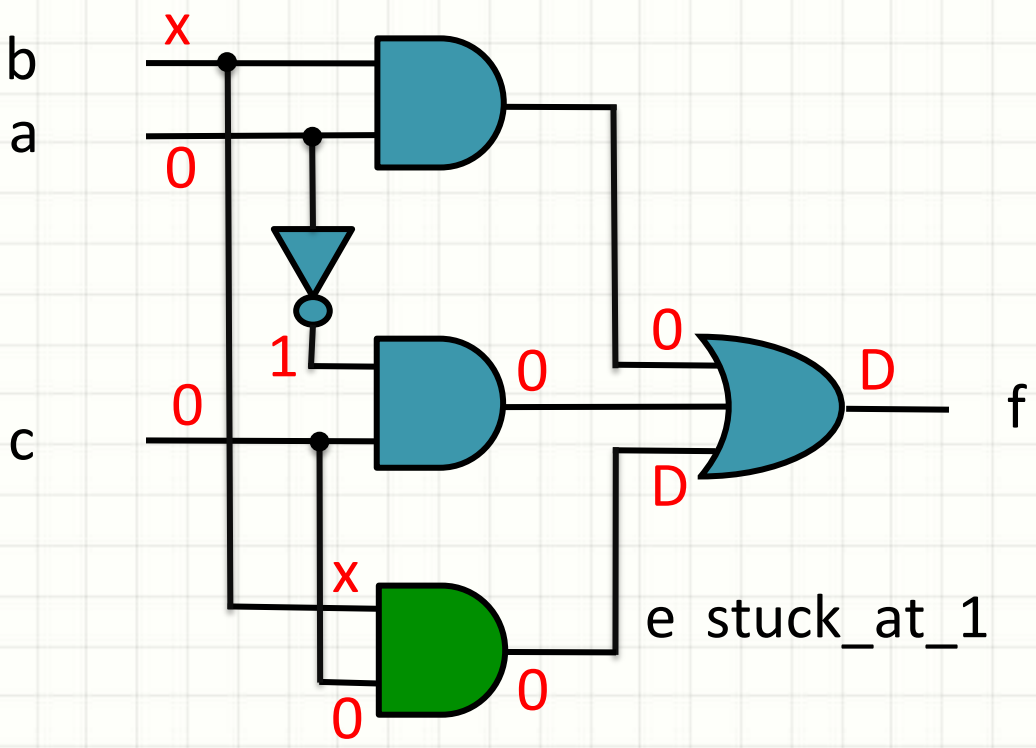
Applying D Algorithm



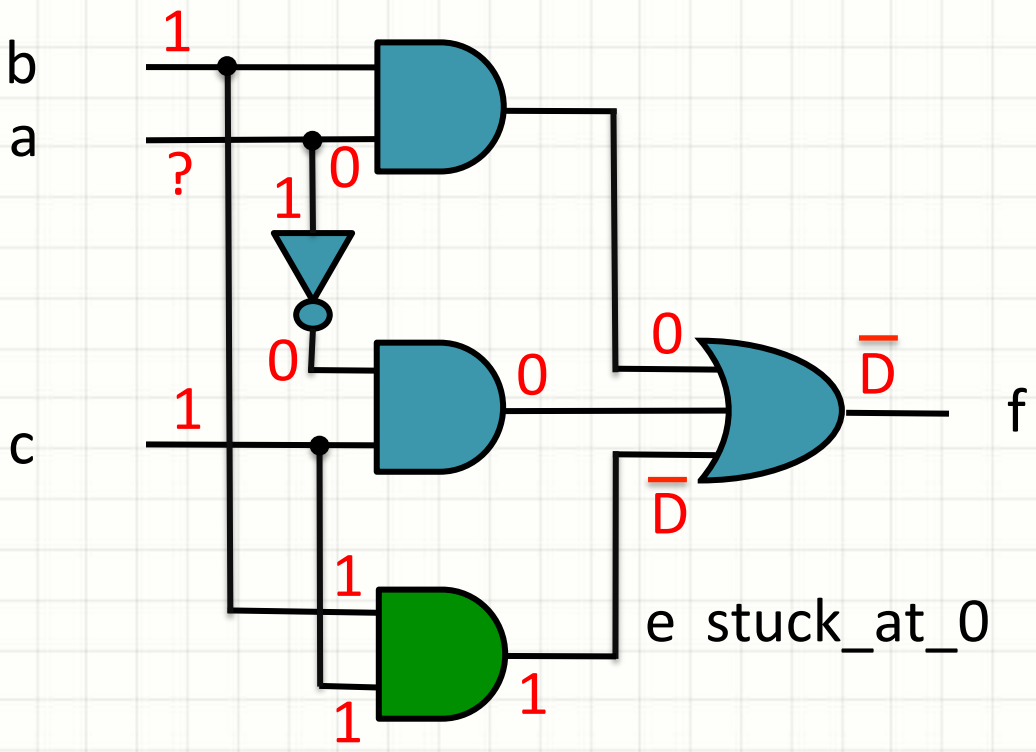
Applying D Algorithm



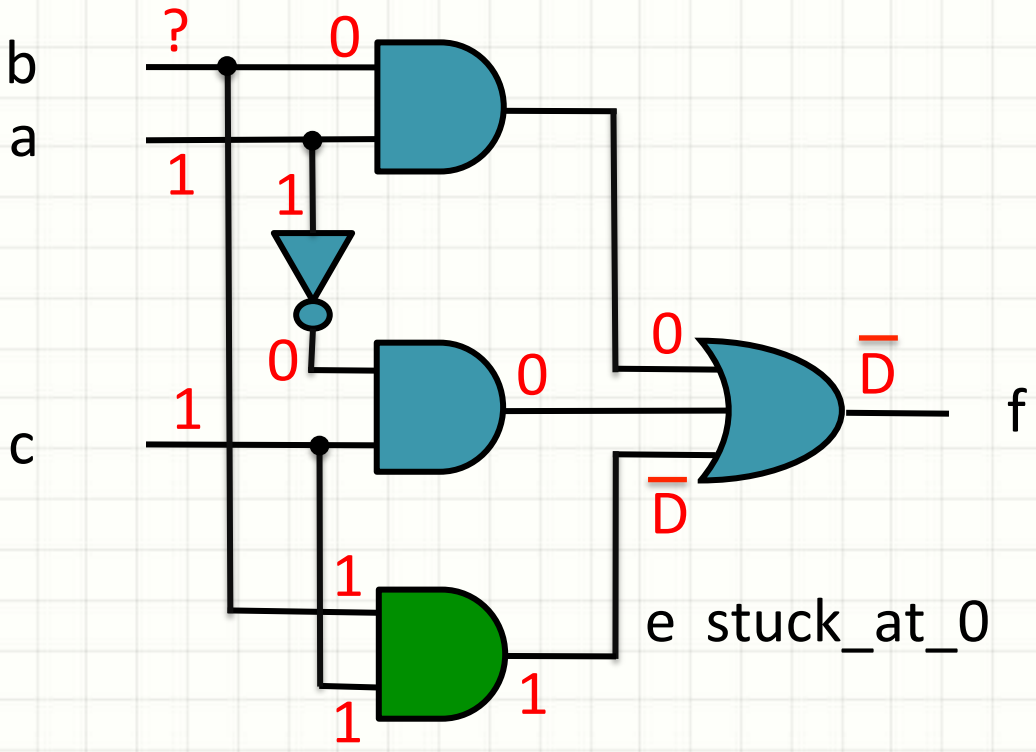
Applying D Algorithm



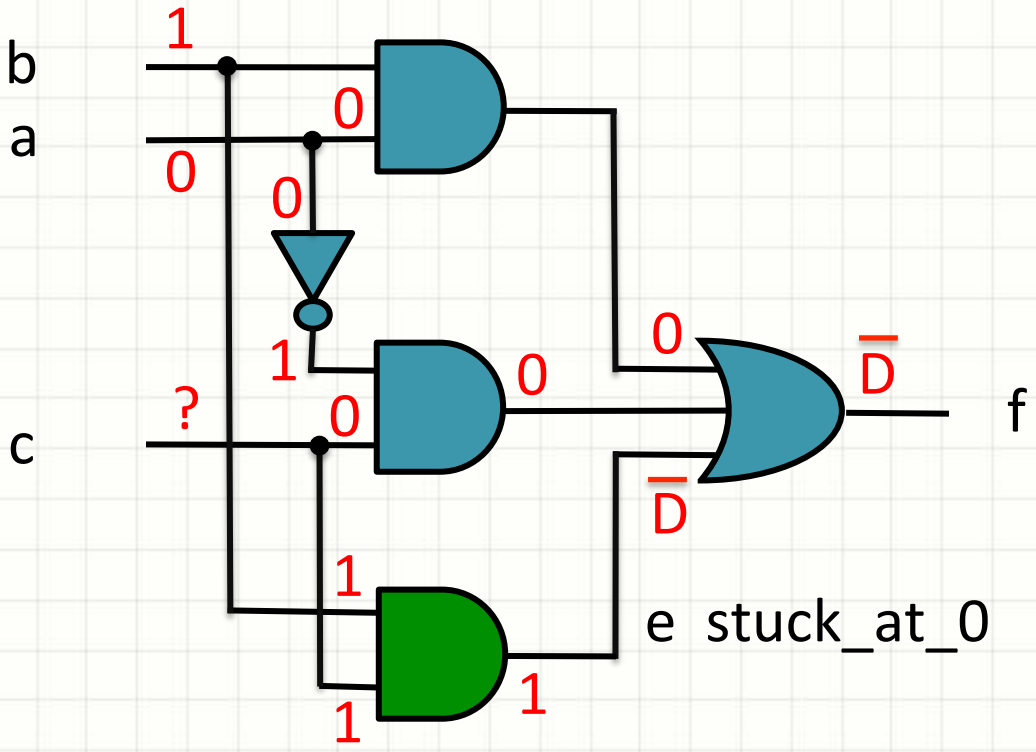
Applying D Algorithm



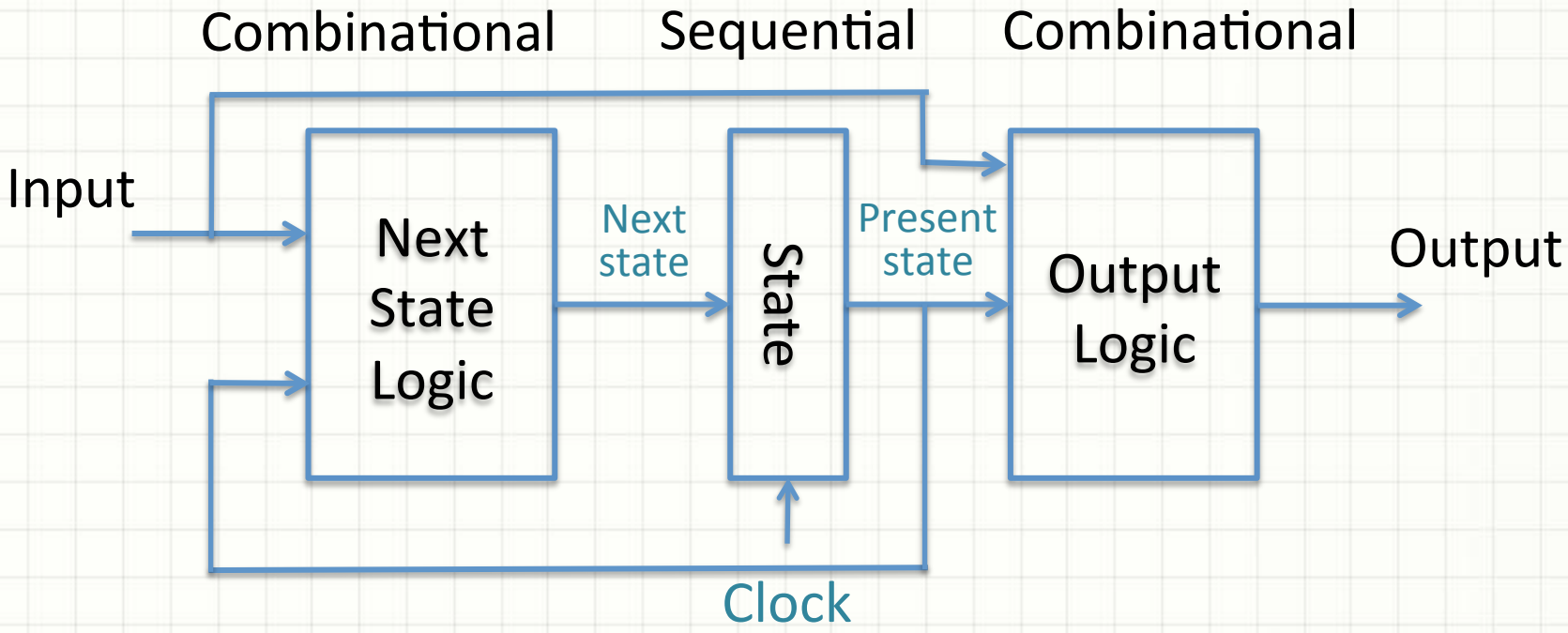
Applying D Algorithm



Applying D Algorithm

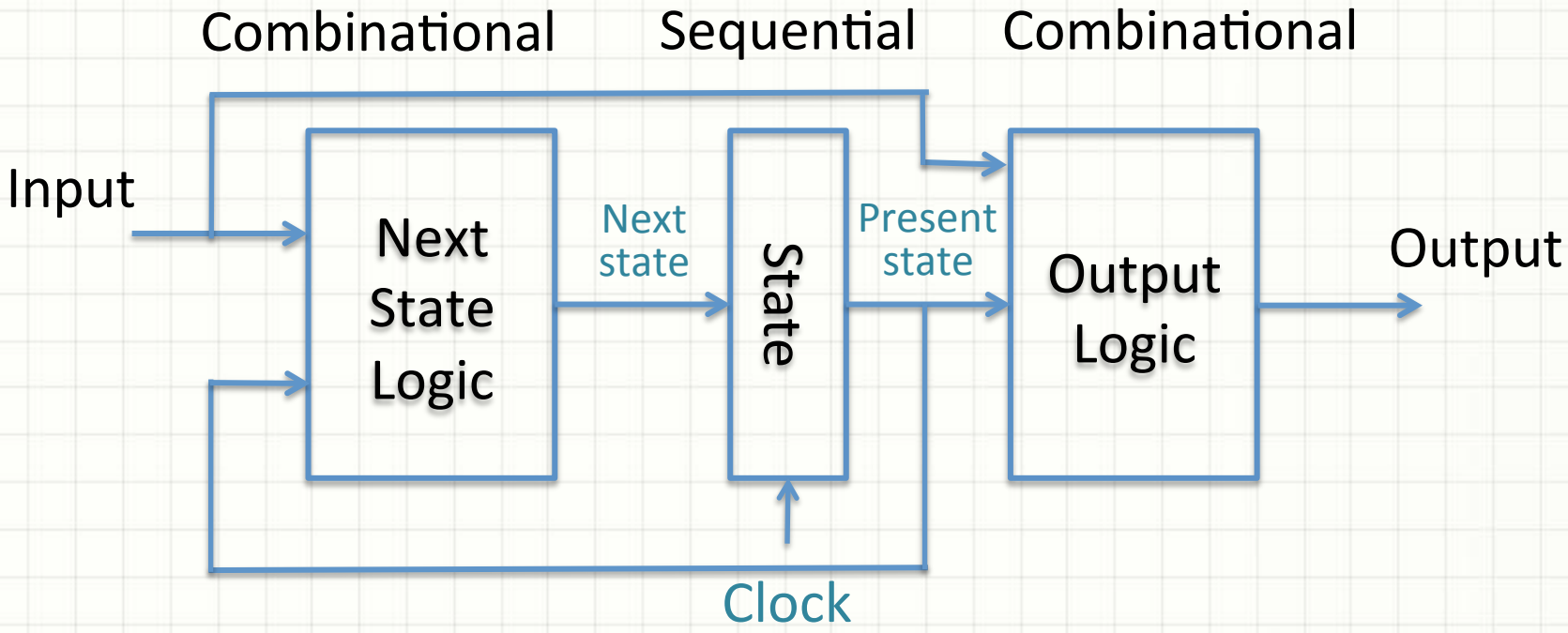


Testing of sequential circuits



- How do you know the present state?
- Can you test just by applying inputs and observing outputs?

Testing of sequential circuits



- Useful to have a reset input for the state register to take the FSM to a known initial state
- Test transitions from initial state to other states
- Test other state transitions and outputs

Testability

Defined in terms of

- controllability: ease /difficulty of setting a particular logic signal to a 0 or a 1
  - primary inputs are always controllable
  - other signals may be controlled through primary inputs
- observability: ease /difficulty of observing the value of a logic signal
  - primary outputs are always observable
  - other signals may be observed through primary outputs

Design for Testability (DFT)

Making a design more testable

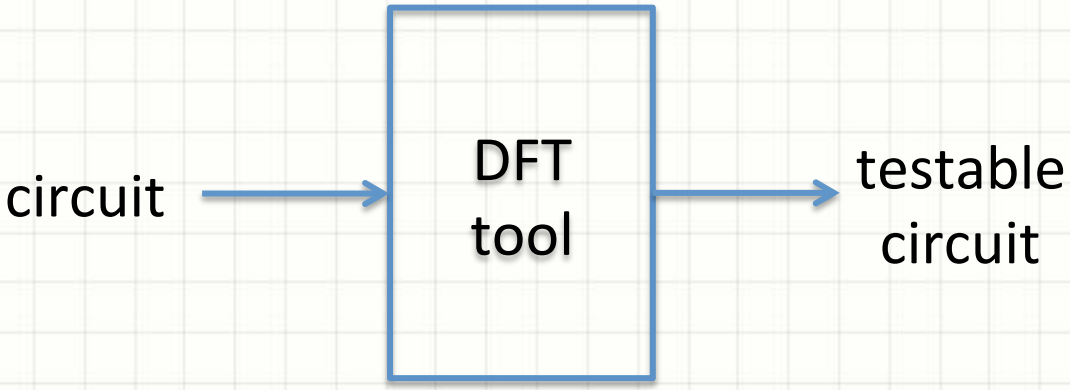
- Adhoc DFT
  - "good" design practices or guidelines
- Structured DFT
  - involves adding extra logic and signals dedicated for test according to some procedure

Adhoc DFT

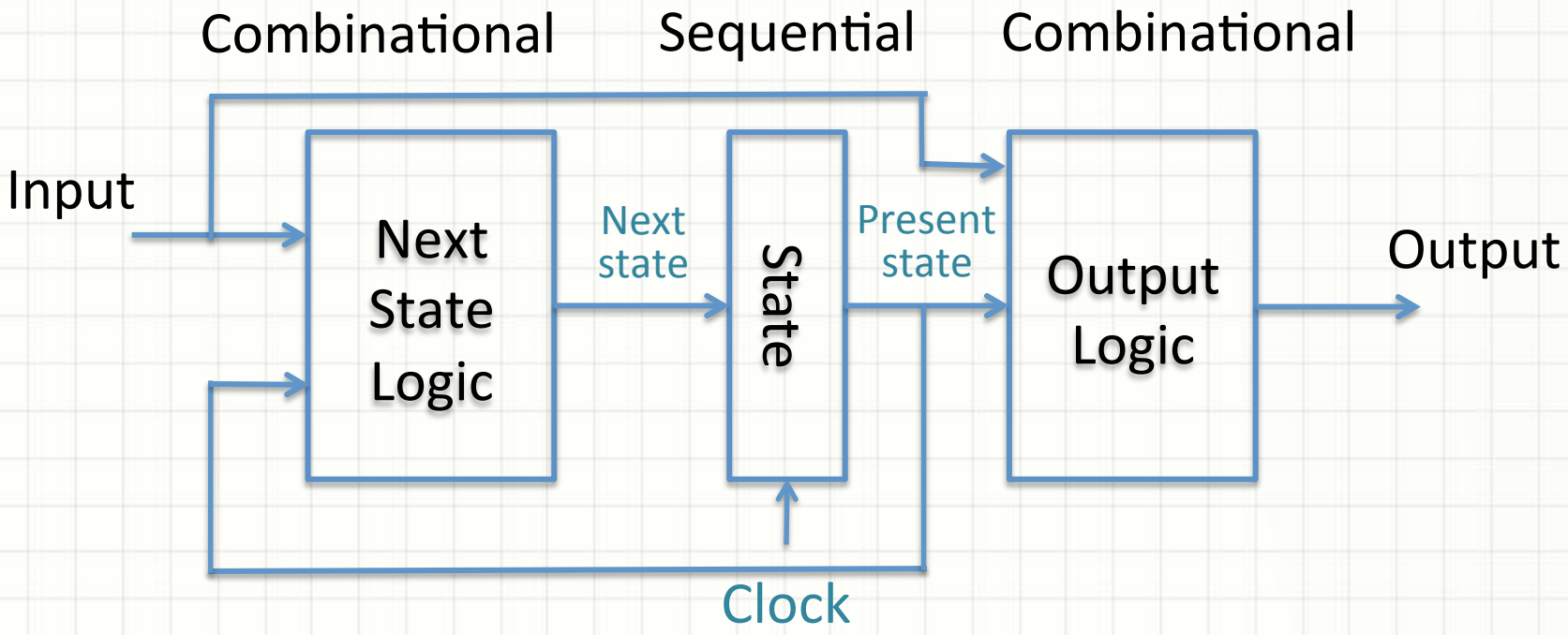
- Avoid *asynchronous* logic feedbacks.
  - Feedback can result in oscillation.
  - ATPG are designed to work on acyclic combinational logic.
- Make FFs initializable, i.e., provide clear and reset.
- Avoid gates with a large fan-in.
  - Large fan-in makes the inputs difficult to observe and the output difficult to control.
- Provide test control for difficult to control signals.
  - For example, signals produced by a long counter require many clock cycles to control.
  - This increases the length of the test sequence.

Structured DFT

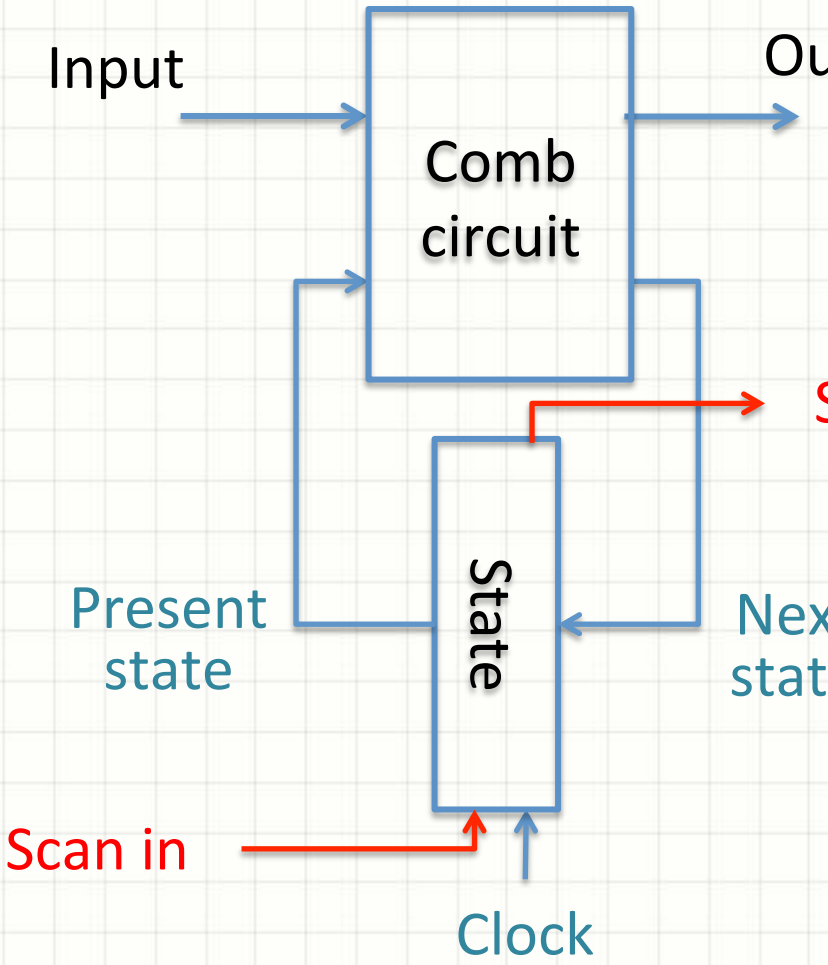
- Add extra logic and signals dedicated for test according to some procedure
- Scan based DFT is the most common approach
- Tools exist to carry out this modification of the designs



Design for testability - scan



Design for testability - scan



Improve controllability
and observability
of the state register

Two modes of operation:

- Normal mode
- Test mode

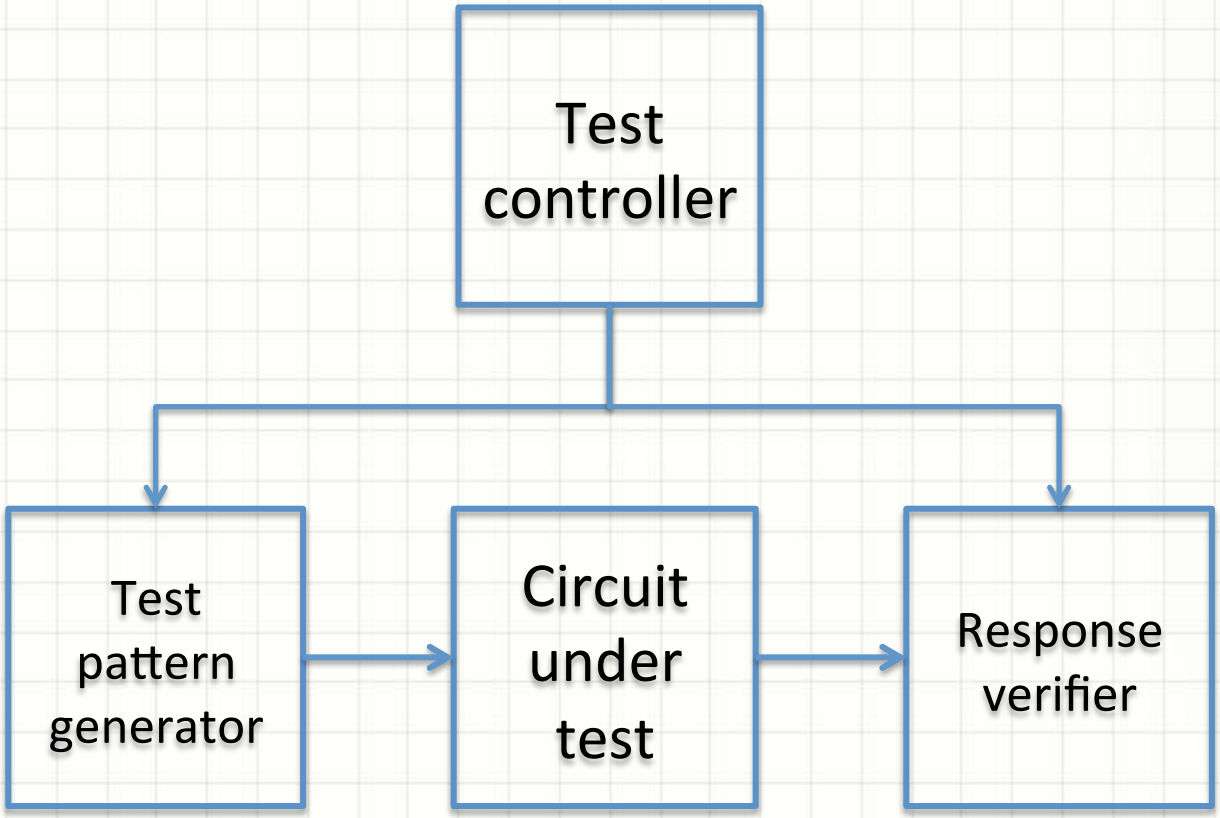
Scan variations

- Full scan | partial scan
- Single scan chain | multiple scan chains
  - more test time vs more input/output pins

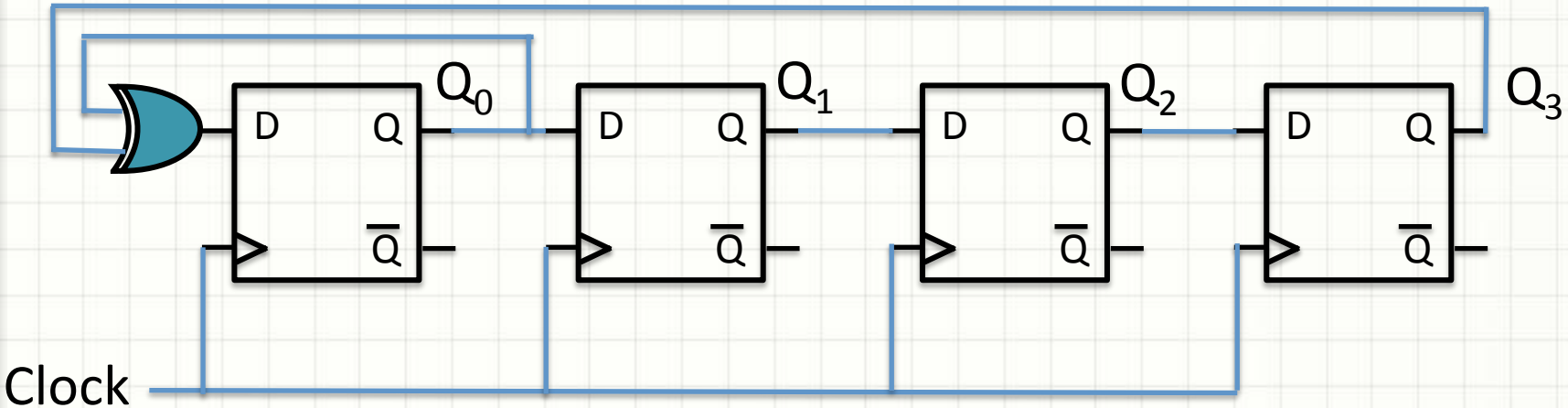
Built-in Self Test (BIST)

- Test patterns are generated internally
- No need to deal with large volumes of test data.
- It can be used to perform *at-speed* test.
- BIST entails three tasks:
  - TPG
  - Test application
  - Response verification
- Our focus on Logic BIST not Memory BIST

BIST


Test controller
Test pattern generator
Circuit under test
Response verifier

Pseudo random sequence generator (PRSG)

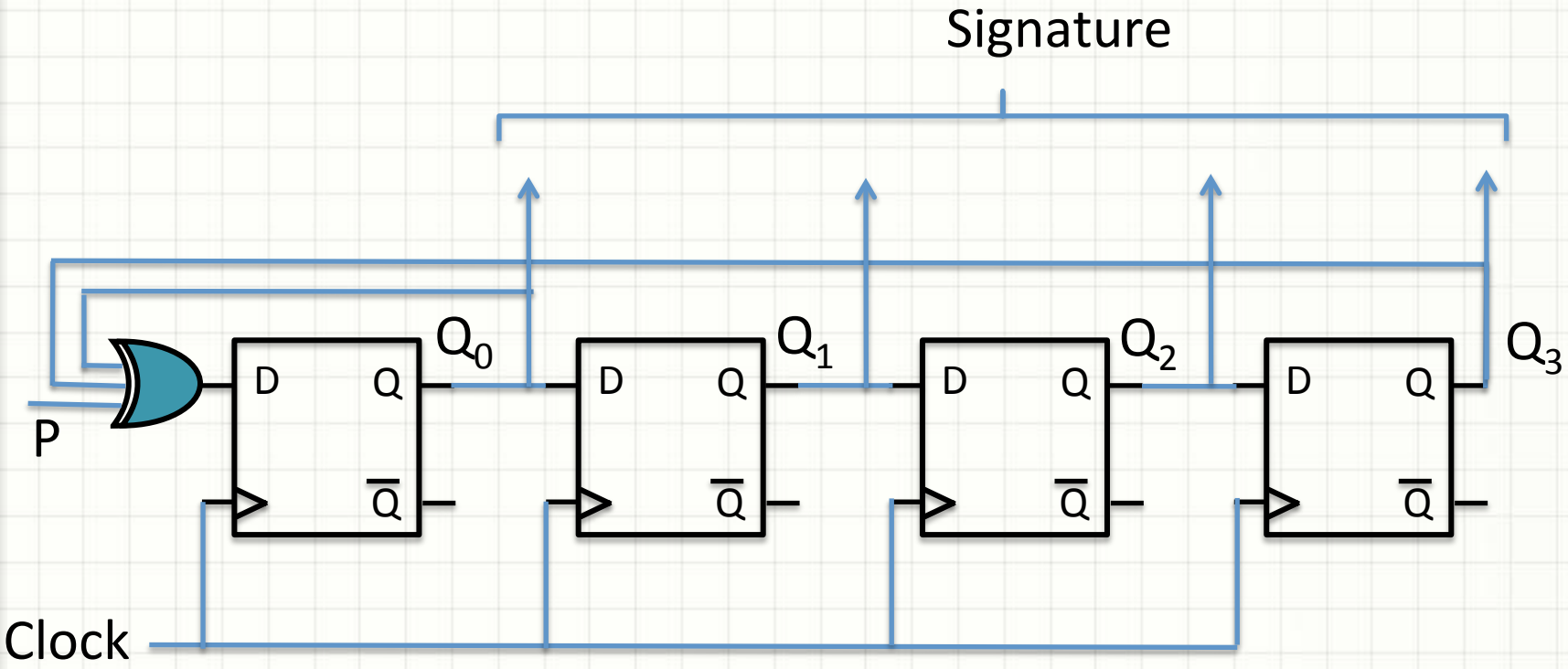


1 0 0 0
1 1 0 0
1 1 1 0
1 1 1 1
0 1 1 1
1 0 1 1
0 1 0 1
1 0 1 0

1 1 0 1
0 1 1 0
0 0 1 1
1 0 0 1
0 1 0 0
0 0 1 0
0 0 0 1
1 0 0 0

Linear Feedback Shift Register (LFSR)

Input compressor circuit/ Signature analyser



- Single input and multiple input compressors
- BILBO: Built-in logic block observer
  - combines scan, TPG, SA

What have we learnt?

- Logic design (combinational circuits)
  - truth tables, expressions, circuits, minimization
- Logic design (sequential circuits)
  - state transition tables, diagrams, state minimization
- Combinational & sequential modules
  - mux/demux, coders/decoders, FFs, registers, counters
- From logic to arithmetic
  - representations, conversions, operations and operators

What have we learnt?

- Technology
  - transistor to FPGA and things in between
- Design
  - VHDL: data flow, procedural, structural
  - from algorithmic description to circuits
  - control-data partition
- Testing
  - testing tools, design for testability

THANKS