Motivation and Introduction

• Ref: F.C. Hennie “Fault detection experiments for sequential circuits”, 5th annual symposium on switching and automata theory, 1964.
• Motivation
  – Test generation at higher level of abstraction in which only the function of the circuit is known but the implementation (structure) is not known.

An example

• Consider testing a 4-bit ALU
  – We need not know the structure – we can determine the number of inputs and outputs. If the number is small we can test the circuit exhaustively.
  – Can such a technique be used for sequential circuits, even if it is fairly small, such as a small finite state machine. Such FSMs exist often in practice (embedded controllers are good examples of such FSMs).
  • Derivation of tests for such circuits is of interest for the following two reasons
    – Need not worry about the realization and underlying technology
    – Such tests can also be used for validation and verification

Sequential circuit model

• Two ways to express a state machine
  – State table
  – State diagram
  – M = (Q, I, O, NS, OU)
    Q = set of states
    I = set of inputs from an input alphabet
    O = set of outputs from an output alphabet
    NS = next state function
    OU = output function

Fault model

• Two formulations of the test problem
  – Given the behavior of the circuit (such as state table), verify the behavior by applying the inputs and observing the outputs. Objective is to find a sequence of inputs that will verify the behavior.
  – Given a sequence of inputs and outputs, construct a state machine that will behave as specified by the input/output sequence
  • The above two problems have similarities but we will address the first of the two problems.
Fault model

- Assumptions about the faults
  - Number of states in the FSM are known or these are upperbounded
  - No fault causes an increase in the number of states or increase beyond the upperbound
- We will also limit our discussion to a class of FSMs that have some special properties. These properties are defined in the “theory” section of the discussion

Theory

- Strongly connected machine/circuit: every state is reachable from every other state
  - There are no “source” or “sink” states
- An example FSM – strongly connected?

<table>
<thead>
<tr>
<th>PS</th>
<th>x = 0</th>
<th>x = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>C/0</td>
<td>A/0</td>
</tr>
<tr>
<td>B</td>
<td>B/1</td>
<td>D/0</td>
</tr>
<tr>
<td>C</td>
<td>A/0</td>
<td>B/0</td>
</tr>
<tr>
<td>D</td>
<td>B/1</td>
<td>C/0</td>
</tr>
</tbody>
</table>

Theory (contd.)

- Synchronizing sequence
  - Application of this sequence takes the machine to a known state (final state), irrespective of the start state (initial state) of the circuit
- Synchronizing tree – see next slide

Theory (contd.)

- Synchronizing tree
- Ambiguity – states the circuit may be in
- Example:
  - Initial ambiguity (ABCD)
  - After an application of 0 the ambiguity is (ABC)
  - SS = 0 1 0 1 0 (Final state = B)

Theory (contd.)

- Homing sequence – application of this sequence and observation of outputs can determine the final state of the circuit
- Distinguishing sequence – application of this sequence and the observation of outputs and determine the initial (start) state of the circuit
  - Clearly this can also determine the final state of the circuit

Theory (contd.)

- Homing sequence
  - Construct a homing tree
  - 010 is a homing sequence
  - If output 000 – final state is C
  - If output 101 – final state is B
- Distinguishing sequence –
  - Construct a distinguishing tree
  - This machine does not have a DS
- Transfer sequence –
  - a sequence, Tij, that will take the machine from state i to j
Theory (contd.)

- Example:
  - SS ?
  - H S ?
  - DS = 100

<table>
<thead>
<tr>
<th>st</th>
<th>output</th>
<th>st</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1 0 0</td>
<td>C</td>
</tr>
<tr>
<td>B</td>
<td>1 0 1</td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>0 0 1</td>
<td>A</td>
</tr>
<tr>
<td>D</td>
<td>1 1 0</td>
<td>C</td>
</tr>
</tbody>
</table>

Checking experiment design

- Three part sequence
  - Part 1: Initialize the FSM to a known state
  - Part 1: verify that the FSM has n states
    - Check that there are n distinct states
  - Part 2: verify that all transitions from every state are correct
    - Apply one input at a time and check the output and the state of the circuit

Checking experiment design (contd.)

- Checking sequence construction
  - Apply SS and take the circuit to a known state
  - Repeat for each state
    - (known state) DS (transfer to another, different, state)
      - DS is used to verify the known state
    - Repeat of each state and every input
      - (known state) input DS (transfer to a known state)
        - verify output when input is applied
          - DS is used to verify that the transition was indeed correct

Checking experiment design (contd.)

- An example
  - SS = 01010
  - DS = 100

<table>
<thead>
<tr>
<th>PS</th>
<th>x = 0</th>
<th>x = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>C/0</td>
<td>D/1</td>
</tr>
<tr>
<td>B</td>
<td>C/0</td>
<td>A/1</td>
</tr>
<tr>
<td>C</td>
<td>A/1</td>
<td>B/0</td>
</tr>
<tr>
<td>D</td>
<td>B/0</td>
<td>C/1</td>
</tr>
</tbody>
</table>

| Phase 1: SS TCA DS TCB DS TAC DS TAD DS |
| C   | C    | A    | A    | C    |
| Phase 2: TCA 0 DS TAA 1 DS check output null check output |

Checking experiment design (contd.)

- Checking sequence – reducing sequence length
  - States need not be verified in the order we want them, they can be verified as they appear while designing the sequence
  - Phases 1 and 2 can be overlapped
  - Overlap parts of sequences where ever possible
  - If there is more than one DS, these can be integrated with in the design of sequence

Limitations of the method

- Assumptions are very restrictive and limit the application of the method
- Machine do not have SS, DS, etc. require more complex algorithms
- Length of the sequence can be very long
  - SS can be as long as O(\(n^3\))
    - The known best bound is \( n(n+1)(n-1)/6 \)
  - TS can be no longer than length n
  - DS – this can be very long in theory
    - \( (n-1)n^2 \)
    - Hence total sequence length can be \( O(2^n) \), where \( k \) is the number of flip-flops and \( n = 2^k \)
<table>
<thead>
<tr>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Need for functional testing methods for sequential circuits</td>
</tr>
<tr>
<td>• Described a fault model for functional faults in FSMs</td>
</tr>
<tr>
<td>• Developed theoretical foundation for FSM testing</td>
</tr>
<tr>
<td>• Design of test sequence</td>
</tr>
<tr>
<td>• Limitations of the method</td>
</tr>
</tbody>
</table>