Chapter 6 Test Generation for Sequential Circuits

Sequential Circuit Testing

- Exhaustive testing requires $> 2^n$ tests (sensitive to the order of tests).
- Testing requires a *sequence* of test vectors.
- Still has all the complexity of C/L ATPG.
- Requires initialization of the machine, which may be difficult:
  - Long initialization sequence.
  - Simulator limitations.
  - Invalid state justification.
- Faults may cause the number of internal states to grow.
- Have to cope with the harder-to-detect delay faults (timing errors).

General Case:

- Make no assumption about internal state.
- Apply test sequence T_α for a fault α and observe the response sequence R_α. If R_α is disjoint from the set of normal response sequences starting from any possible state, then T_α *strongly detects* α.
- NOT doable in practice!

Restricted Case (Static Testing):

① With the circuit output stable, apply the next input vector.

② Wait for new output to stablize, and then observe.

③ Do ① & ② for all $t \in T_\alpha$.

ATPG Approaches:

❶ *Time-frame expansion* (unrolling the feedback path): convert time domain into space domain, and use combinational ATPG—Extended D-algorithm, Sequential 9V, EBT, BACK, etc.

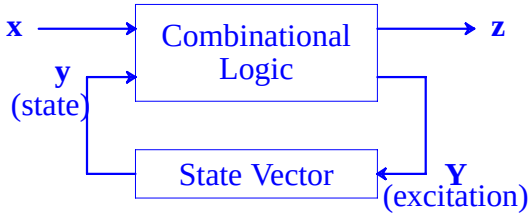
❷ *Simulation-based approach*: search for test vectors guided by cost functions and simulations—CONTEST.

❸ *Checking experiment*: formal approach to derive test sequence for FSMs (useful only for small circuits).

❹ *Scan*: modify the circuit to facilitate combinational ATPG.

❺ *Functional test*: testing that a (part of a) system operates correctly in a functional sense (i.e., each module is realization & fault independent—fault model is at functional level).

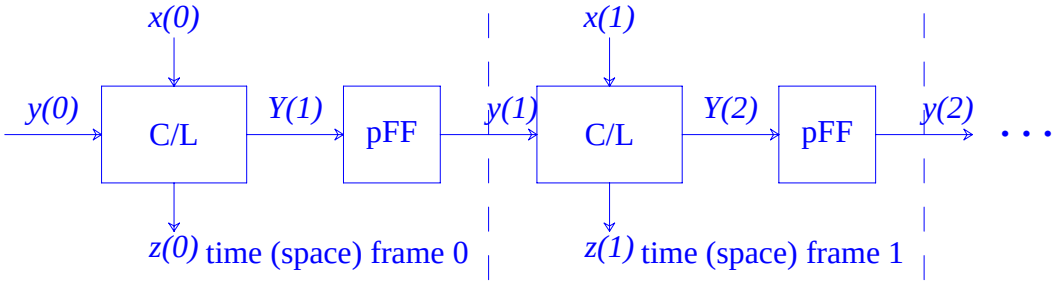
Time-Frame Expansion (Iterative Array Conversion)



Sequence of inputs in time: $x(0), x(1), \ldots, x(n)$.

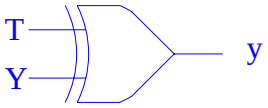
Sequence of outputs in time: $z(0), z(1), \ldots, z(n)$.

Sequence of internal states in time: $y(0), y(1), \ldots, y(n)$.



- For synchronous sequential circuits (FSMs).
- Transforming FSMs to iterative logic arrays by unrolling the Huffman model.
- Applying D-alg (or PODEM, path sensitizer, etc.) to the ILA.
- Pseudo flip-flop (pFF) is a combinational circuit mapping its excitation function onto its output, e.g., a pseudo T-FF can be constructed by an XOR gate as shown below.

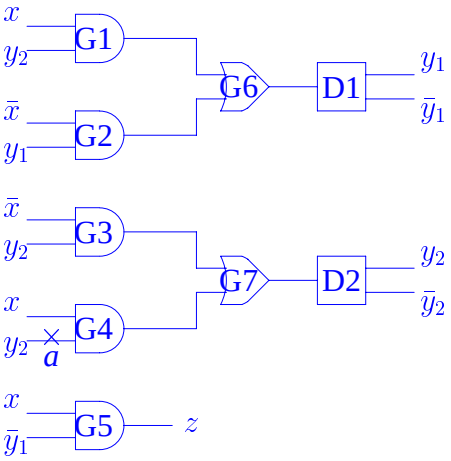
T	Y	y
0	0	0
0	1	1
1	0	1
1	1	0

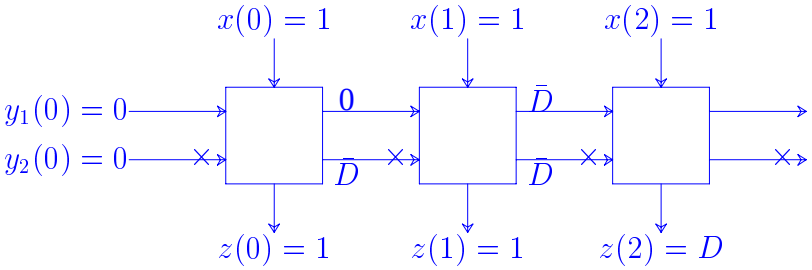


- When a single stuck fault is present in the real sequential circuit, it will appear as a multiple fault, existing in each unfolded iteration (time frame).

Example 1

We use the unrolling method to derive a test sequence for the fault $\alpha \equiv a/1$ of the following circuit. Assume that the initial state of the sequential circuit is $(y_1, y_2) = (0, 0)$.





t_0: Initial state $y_1 = y_2 = 0$.
$\because\ y_2 = 0$ and $a/1 \therefore\ a = \bar{D}$.
Set $x(0) = 1\ \Rightarrow\ y_2(1) = \bar{D}$ and $z(0) = 1$.
$\because\ z \neq D$ or $\bar{D}$. $\therefore$ Continue!

t_1: Cannot propagate D or $\bar{D}$ to z yet
$\therefore$ Set $x(1) = 1 \Rightarrow y_1(2) = y_2(2) = \bar{D}$ and $z(1)\ =\ 1$.

t_2: On $G5$, let $x(2) = 1$
$\because\ y_1(2) = \bar{D}. \therefore\ z(2) = D$.

$\therefore$ Test sequence $X\ =\ 111$. □

Termination rules

- ✯ If $z = D$ or $\bar{D}$ then a test sequence is found.
- ✯ If $k > 4^n$, where n is the number of FFs of the original circuit ($\neq$ number of time frames = k), then the circuit is redundant and no test sequence exists.
  ☞ $\because$ 4 possibilities (0, 1, D, & $\bar{D}$) for each y_i (FF).

Extended D-algorithm [Kubo, NEC R&D, 10/1968] [Putzolu & Roth, IEEE TC, 6/1971]

1. Select a target fault α.
2. Create a copy of C/L; time-frame = 0.
3. Generate a test for α by D-algorithm for time-frame 0.
4. If fault effect is propagated to FFs, continue propagation in the next time-frame.
5. If there are values required in FFs, continue justification in the previous time-frame.

☞ It is incomplete, because a fault effect may be propagated to a faulty line in a subsequent time-frame, and 5 values are insufficient for such a situation.

- It is complete if 9 values (instead of 5 values) is used [Muth, IEEE TC, 6/1976]: 0/0, 0/1, 0/u, 1/0, 1/1, 1/u, u/0, u/1, u/u (implementation is very complex, however).
- Going both forward and backward in time requires maintaining a large number of time-frames during test generation, and it is hard to identify *cycles*. ∴ Impractical.

Extended BackTrace (EBT) Algorithm [Marlett, DAC-78 & 86]

1. Select a path from the fault site to a PO. /* May involve several time-frames. */
2. Sensitize the path backwards from the PO to the fault site. /* Justification is done in the same direction. */
3. If it fails, select another path;
   if there is no path left, select another PO;
   go to 2.
4. Justify the required value at the fault site.

- Only 2 time-frames need to be maintained: present time-frame and previous time-frame.
- Easier to identify *cycles*.
- Only a single path is selected each time.
- Number of possible paths from fault site to POs can be large.

Exercise 1
Show that EBT is complete, or prove that it is not by giving a counter example.

Definition 1
Drivability is a measure associated with a signal that estimates the effort of propagating a D or $\bar{D}$ from the fault site to the signal.

Back Algorithm [Cheng, ICCD-88]

1. Calculate controllability measures for the fault-free circuit.
2. Pick a target fault;
   Calculate controllability measures for the faulty circuit.
3. Derive drivability measures.
4. Assign a $D/\bar{D}$ to a PO;
   time-frame = 0;
   if all POs with finite drivability have been considered, go to 2 (fault undetectable).
5. Justify all values—use drivability to guide the backward search (implication), i.e., select the gate input with the smallest drivability.
   - ☞ May involve several time-frames backwards.
   - ☞ May need backtracking on the decisions.
   - ☞ May need backtracking on the time-frames.
   - ☞ May need to back up to step 4.
   - ☞ Sensitized paths are created implicitly.

Other Time-Frame Expansion Algorithms

- ✯ ESSENTIAL [Schulz & Auth, ITC-89].
- ✯ FASTEST [Kelsey & Saluja, ICCAD-89].
- ✯ HITEC [Niermann & Patel, EDAC-91].
- ✯ Lee-Reddy [Lee & Reddy, ICCAD-91].

Simulation-Based Test Generation

CONTEST [Agrawal, Cheng, & Agrawal, IEEE TCAD, 2/1989]

- A concurrent test generator for sequential circuits—using a concurrent fault simulator which automatically analyzes races and oscillations.

```
Initialization:
{1. Start with an arbitrary vector and all FFs in
    unknown state;
 2. Generate new vectors to reduce cost by 1-bit changes
    in the present vector; /* only true-value sim */
    /* cost = number of FFs in unknown state */
 3. Stop when cost drops below desired value; }

Tests for concurrent targets:
{1. Start with initialization vectors;
 2. Fault simulate vectors and remove detected faults;
 3. Compute cost function for the last vector;
    /* cost(undetected f.)=min dist of its effect from a PO */
    /* cost(vector)=sum of costs of all undetected faults */
 4. New vectors = those 1-bit changes that reduce cost; }

Tests for remaining undetected single faults:
{1. Revise the cost function (a dynamic testability measure);
    /* cost = K x activation cost + propagation cost */
    /* activation cost = dynamic CY of faulty line */
    /* propagation cost = min dynamic OY of faulty line */
    /* K is a large weighting factor */
 2. Generate a test for a single fault; }
```

- Timing is considered, and asynchronous circuits can be handled.
- Can easily be implemented by modifying an existing fault simulator.
- Cannot identify untestable faults.

☞ Coverage for hard-to-detect faults can be low.

Checking Experiments [Hennie, 1964]

Given the state table of an FSM, find an I/O sequence pair (X, Z) $\ni$ the response of the machine to X will be Z iff the machine is operating correctly.

Definition 2
An *adaptive experiment* is an experiment in which the choice of the present input is based on the earlier outputs, while a *preset experiment* is one in which the entire input sequence X is specified in advance.

☞ A measure of efficiency: length of (i.e., # of symbols in) X.

Assumptions on the FSM:

- Fully specified and deterministic.
- Reduced, i.e., $q_i \neq q_j,\ \forall i \neq j$.
- Strongly connected, i.e., $\forall i, j,\ \exists X \ni q_i \rightarrow q_j$.
- No fault will increase the number of states.
- Has a distinguishing sequence.

Definition 3
A *homing sequence* (HS) for an FSM is an input sequence $\ni$ the corresponding output sequence uniquely determines its final state (regardless of the initial state).

Definition 4
A *distinguishing sequence* (DS) for an FSM is an input sequence $\ni$ given different initial states it produces different output sequences.

☞ Every DS is an HS.

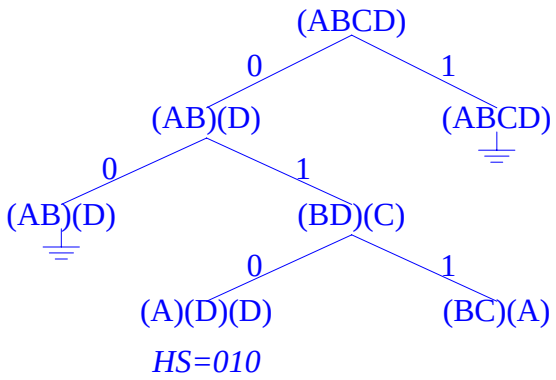
☞ The DS is a necessary tool for performing a checking experiment; it provides a way of identifying the state of a machine.

Example 2

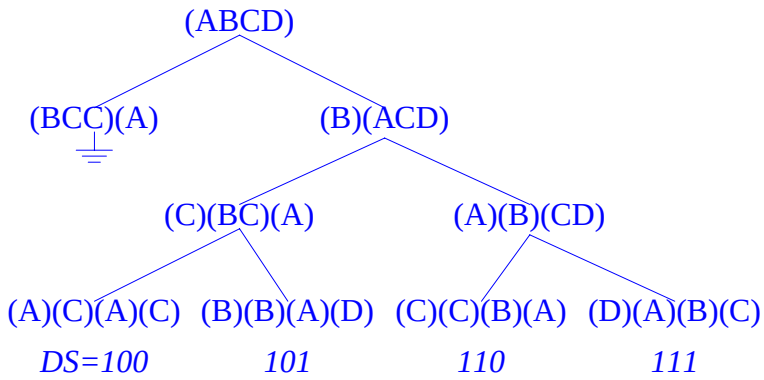
For the following two state transition tables representing two different FSMs, a homing sequence for the FSM shown on the left and a distinguishing sequence for the one shown on the right are derived below by using a *homing tree* and a *distinguishing tree*, respectively. □

PS	$x=0$	$x=1$
A	B,0	D,0
B	A,0	B,0
C	D,1	A,0
D	D,1	C,0

PS	$x=0$	$x=1$
A	C,0	D,1
B	C,0	A,1
C	A,1	B,0
D	B,0	C,1



homing tree



distinguishing tree

Definition 5

A *synchronizing sequence* (SS) for an FSM is an input sequence which takes the machine to a specified final state regardless of the output sequence or the initial state.

- ☞ Not every FSM has an SS (or DS).
- ☞ An HS with a length $\leq n(n-1)/2$ exists for every reduced FSM of n states.

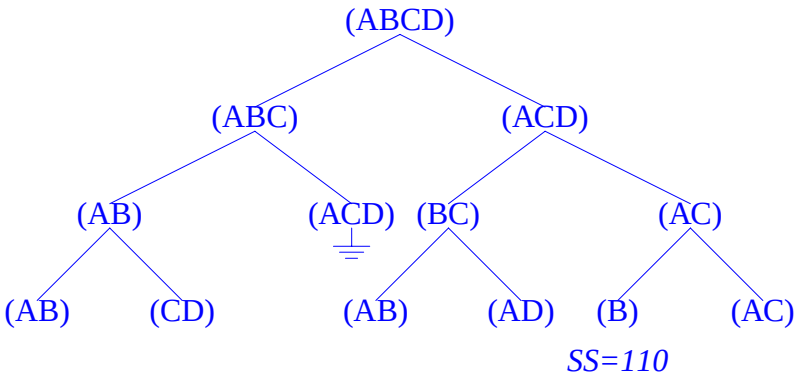
Exercise 2

Prove the above two statements.

Example 3

A synchronizing sequence for the following FSM can be derived by using a *synchronizing tree* as shown below. □

PS	$x=0$	$x=1$
A	B,1	C,0
B	A,0	D,1
C	B,0	A,0
D	C,1	A,1



synchronizing tree

Sketch of a Checking Experiment:

① (Initialization) Initialize the state, using an SS or HS.

② (State identification (A-seq)) Visit each state to assure its existence. DS must be applied twice to each state of the machine.

③ (Transition verification (B-seq)) Verify every state transition using the DS.

☞ The result for X takes the form: $X_1X_2X_3$, where X_1 is an SS or HS, X_2 the A-seq, and X_3 the B-seq, where A-seq & B-seq may be mixed up whenever appropriate to shorten $|X|$.

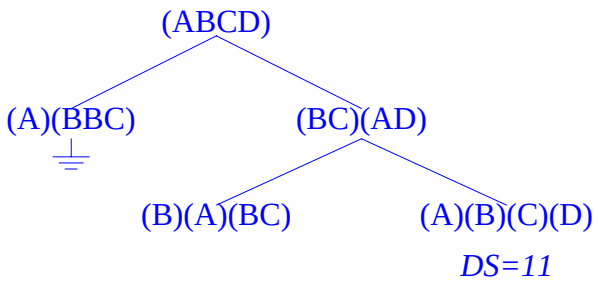
		DS		DS	
input:					
state:	q_i	$\longrightarrow$	q_j	$\longrightarrow$	q_k
output:		z_i		z_j	

- Observation of z_i identifies q_i and establishes q_j.
- The A-seq starts with a repetition of DS until a previously visited state occurs as q_i. The DS is applied once more to identify q_i. Further repetition of DS would continue looping through already visited states. Use a *transfer sequence*, i.e., a shortest input sequence $\ni\ q_i \rightarrow q_j$, to place the machine in a new state, and continue until all states have been verified.
- Not every FSM has a DS. If not, we must modify it to have one [Kohavi].

Example 4

We will find a DS for the following FSM, and use it to design a test sequence (checking experiment) for the machine. Assume that the machine is initialized to state A before the experiment.

PS	$x=0$	$x=1$
A	C,0	D,1
B	B,0	A,1
C	A,1	B,0
D	B,0	C,0



IS	*DS=11*
A	C,10
B	D,11
C	A,01
D	B,00

The result is shown in the next page. Note that in the A-seq, we have $A \xrightarrow{1} D \xrightarrow{1} C$. Therefore, if $A \xrightarrow{1} D$ is verified in the B-seq, then $D \xrightarrow{1} C$ is already verified in the A-seq. $\Rightarrow$ The experiment length can be reduced. $\square$

i/p:		$\frac{DS}{11}$			11			$\frac{TS}{1}$			11			11			11													
q:	A			C			A			D			B			D			B											
o/p:		10			01			1			00			11			00													end A-seq
i/p:		0			11			0			11			1			11			0			11							
q:	B	→	B			D	→	B			D	→	C			A	→	C			A									
o/p:		0			11			0			11			0			01			0			01							
i/p:		1			11			1			11			0			11			1			11							
q:	A	→	D			B	→	A			C	→	A			C	→	B			D									
o/p:		1			00			1			10			1			10			0			00							end B-seq

- ☞ Checking experiment can be considered as a form of functional test; it is realization independent and fault independent.
- ☞ Limit: it requires the generation of the state table, which is impractical for large circuits or for circuits obtained by high-level synthesis.

Definition 6

A *unique input-output sequence* (UIOs) for an FSM is an input-output (IO) sequence $\ni$ given different initial states it has different IO sequences [Sabnani & Dahbura, 1988].

Exercise 3

How do we use UIOs instead of DS for checking experiments? Discuss the advantages and/or disadvantages.

Functional Testing

Select tests which will validate the functional operation of the CUT.

- ☞ For example, the functional test for a FF may consists of
  1. Validating that the FF can be set and reset.
  2. Validating that the FF can hold its state.
- ☞ The testing of RAMs (and PLAs) are usually functional. Checking experiments are impossible for them.
- ☞ There is a growing need for functional-level ATPG due to increasing circuit complexity and synthesis.