Center for Reliable Computing

TECHNICAL REPORT

Using Checking Experiments To Test Two-State Latches

Samy R. Makar and Edward J. McCluskey

94-11

(CSL TR # 94-641)

November 1995

Center for Reliable Computing
ERL 460
Computer Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, California 94305-4055

Abstract:

A general technique to determine conditions for exhaustive functional testing (checking experiment) of two-state latches is derived. This technique is used to derive conditions for checking experiments of various two-state latches. Minimum-length checking experiments are also presented. One of the checking experiments for the D-latch is simulated using an HSpice implementation of the transmission gate latch. All detectable shorted interconnects, open interconnects, short-to-power, short-to-ground, stuck-open, and stuck-on faults are detected. A pin fault test set and a multiplexer-based test set are also simulated. These tests miss some faults detected by the checking experiment.

Funding:

This work was supported in part by the Ballistic Missile Defense Organization, Innovative Science and Technology (BMDO/IST) Directorate and administered through the Department of the Navy, Office of Naval Research under Grant No. N00014-92-J-1782, in part by the Advanced Research Projects Agency under Contract No. DABT63-94-C-0045, and in part by the National Science Foundation under Grant No. MIP-9107760. It was also funded in part by Cirrus Logic.

Imprimatur: Erin Kan and Jonathan Chang

Table of Contents

1. Introduction

Tests for stuck-at faults at latch inputs and outputs miss many internal faults [Reddy 86], [Lee 90] and [Al-Assadi 93]. Reddy, in [Reddy 86], derived tests for stuck-open faults in different latch implementations. Lee, in [Lee 90], analyzed bridging faults in scan registers, and combined the use of current and voltage tests. Al-Assadi, in [Al-Assadi 93], mapped many, but not all, of the internal faults to functional fault models. He also showed that some of the internal faults cannot be mapped to functional fault models. What is needed is a test set that detects all faults (both internal and I/O faults). This paper presents such test sets.

A *checking experiment* is an input-output sequence that distinguishes a given state machine from all other state machines with the same inputs and outputs, and the same number of or fewer states. Checking experiments were first defined by Hennie as follows, "Any circuit that responds to the *checking experiment* in accordance with a given state table and starting state either must be operating correctly or else must have suffered a malfunction not in the given class." [Hennie 64]. The important property of a checking experiment is that it contains enough information to derive the flow table.

Even though Hennie's work was for pulse-mode circuits (often called synchronous sequential circuits), checking experiments can be derived for fundamental-mode circuits (often called asynchronous circuits) by modifying the procedure. Friedman, in [Friedman 71], discusses the restrictions in fundamental-mode circuits. Since there is no inherent clock, the machine can change state after any input changes, and the same input cannot be repeated. Also, for deterministic behavior, only one input can be changed at a time. Some faults can cause critical races, making the behavior non-deterministic, and can thus not be guaranteed to be detected. We did not encounter such faults in our simulations.

In Section 2, we present various latches and minimum-length checking experiments for them. In Section 3, we present a general technique for deriving checking experiments for two-state latches. Separate sub-sections are devoted to the derivation of checking experiment requirements and minimum-length checking experiments for each latch type, with details given in the appendices. The checking experiment for the D-latch was simulated in HSpice using a transmission gate implementation. The results of this simulation are compared with those of a pin fault test set and a multiplexer-based test set in Section 4.

2. Latches and Their Minimum-Length Checking Experiments

Various latches are discussed in this paper (see Table 2-1). The simplest latch type is the SR-latch. An *SR-latch* is a sequential element that can be set or reset by activating the appropriate input. Even though the SR-latch is still occasionally used, the most commonly used latch today is the D-latch. A *D-latch* is a sequential element, in which the data input is propagated to the output when the clock is active, otherwise it holds the stored value. A *D-latch with Asynchronous Set/Reset* is a D-latch that can be set or reset when the clock is not active. Scan-paths require latches with two different data sources. These can be either Multiplexed-Data latches or Two-Port latches. A *Multiplexed-Data latch* (*MD-latch*) is a D-latch with multiplexed data inputs; a *Two-Port latch* has two control inputs with the data source determined by the active control input [McCluskey 86]. A *Load Enable latch* is a D-latch with a gated control input, and a *D-Enable latch* is a D-latch with gated data. An *XOR Input latch* performs an exclusive-or operation on its two data inputs. This latch is commonly used in an LFSR to generate pseudo-random vectors, and to compress results. Other latches commonly used for BIST are the *Built-In Logic Block Observer latch (BILBO latch),* and the *Concurrent Built-In Logic Block Observer latches (CBILBO latches).* The BILBO latch has two data inputs. It can

Table 2-1 Latches and Their Excitation Functions.

Latch Type	Excitation Function	Assumptions	M*
SR-Latch	$Q = S + \overline{R}q$	$SR = 0$	6
D-Latch	$Q = CD + \overline{C}q$		7
D-Latch with Asynchronous Set/Reset	$Q = \overline{R}(S + CD + \overline{C}q)$	$SR = 0$	14
MD-Latch	$Q = C(TS + \overline{T}D) + \overline{C}q$		26
Two-Port Latch	$Q = C_1D_1 + C_2D_2 + \overline{C_1}\overline{C_2}q$	$C_1C_2 = 0$	23
Load Enable Latch	$Q = CLD + \overline{(LC)}q$		15
D-Enable Latch	$Q = CDE + \overline{C}q$		16
XOR Input Latch	$Q = C(D \oplus S) + \overline{C}q$		13
BILBO Latch	$Q = C(B_1D \oplus \overline{B_2}S) + \overline{C}q$		58
CBILBO Latch	$Q_1 = C(\overline{B_1}D \oplus S) + \overline{C}q_1$ $Q_2 = C(B_2S + \overline{B_2}D) + \overline{C}q_2$		25 26

*M - minimum length of checking experiment.

be configured to load either of the two inputs (one a scan input, and the other for normal operation), load the exclusive-or of the two inputs (for signature analysis), or load 0. The CBILBO latches, an extension of the BILBO latch, are two latches that can operate simultaneously as a pseudo-random pattern generator and a signature analyzer. The two latches are treated separately, with outputs Q_1 and Q_2. Table 2-1 shows the excitation function of each of these latches and the minimum-length of a checking experiment. Minimum-length checking experiments for each of these latches are shown in Tables 2-2 through 2-11. Details for each latch type are presented in Section 3.

Table 2-2 A Minimum-Length (6) Checking Experiment for SR-Latch.

S	0	0	1	0	0	0
R	1	0	0	0	1	0
Q	0	0	1	1	0	0

Table 2-3 A Minimum-Length (7) Checking Experiment for D-Latch.

C	1	1	0	0	1	0	0
D	1	0	0	1	1	1	0
Q	1	0	0	0	1	1	1

Table 2-4 A Minimum-Length (14) Checking Experiment for Asynchronous Set/Reset Latch.

C	1	1	0	0	0	0	0	0	0	0	0	0	1	0
D	1	0	0	0	0	0	0	1	1	1	1	1	1	1
R	0	0	0	0	0	1	0	0	0	0	1	0	0	0
S	0	0	0	1	0	0	0	0	1	0	0	0	0	0
Q	1	0	0	1	1	0	0	0	1	1	0	0	1	1

Table 2-5 A Minimum-Length (26) Checking Experiment for MD-Latch.

D	1	1	1	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
S	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1	0	0	0
T	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
C	1	1	0	0	1	0	0	0	1	0	0	1	0	0	1	0	0	1	0	0	0	1	0	0	1	0
Q	0	1	1	1	0	0	0	0	1	1	1	0	0	0	1	1	1	0	0	0	0	1	1	1	0	0

Table 2-6 A Minimum Length (23) Checking Experiment for Two-Port Latch.

D_1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	1	1	1
C_1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1	0
D_2	0	0	0	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
C_2	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0
Q	1	0	0	0	1	1	1	0	0	0	1	1	0	0	0	1	1	1	0	0	0	1	1

Table 2-7 A Minimum-Length (15) Checking Experiment for Load Enable Latch.

L	1	1	0	0	1	1	0	0	1	0	0	1	1	0	0
D	1	0	0	0	0	1	1	1	1	1	1	1	0	0	0
C	1	1	1	0	0	0	0	1	1	1	0	0	0	0	1
Q	1	0	0	0	0	0	0	0	1	1	1	1	1	1	1

Table 2-8 A Minimum-Length (16) Checking Experiment for D-Enable Latch.

D	1	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0
E	1	1	1	1	1	0	0	1	1	1	0	0	1	0	0	0
C	1	1	0	0	1	1	0	0	1	0	0	0	0	0	1	0
Q	1	0	0	0	1	0	0	0	1	1	1	1	1	1	0	0

Table 2-9 A Minimum-Length (13) Checking Experiment for XOR Input Latch.

D	0	0	0	0	0	0	1	1	1	1	1	1	0
S	1	0	0	1	1	1	1	1	1	0	0	0	0
C	1	1	0	0	1	0	0	1	0	0	1	0	0
Q	1	0	0	0	1	1	1	0	0	0	1	1	1

Table 2-10 A Minimum-Length (58) Checking Experiment for BILBO Latch.

D	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1
S	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
B_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
B_1	0	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
C	1	1	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
Q	1	0	0	0	1	1	1	0	0	0	0	1	0	0	0	1	1	1	0	0	0	1	1	1	0	0	0	1	1
D	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1	1	0	0	0	0
S	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1
B_2	0	0	0	0	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
B_1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
C	0	1	0	0	1	0	0	1	0	0	1	1	1	0	0	1	0	0	1	0	0	1	0	0	0	0	0	1	0
Q	1	0	0	0	1	1	1	0	0	0	1	1	0	0	0	1	1	1	0	0	0	1	1	1	1	1	1	0	0

Table 2-11a A Minimum-Length (25) Checking Experiment for CBILBO Latch 1.

D	1	1	1	1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0
S	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0
B_1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
C	1	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	0	1	0	0	1	0
Q_1	0	1	1	1	0	0	0	1	1	1	0	0	0	1	1	1	0	0	0	0	1	1	1	0	0

Table 2-11b A Minimum-Length (26) Checking Experiment for CBILBO Latch 2.

D	1	1	1	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
S	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1	0	0	0
B_2	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
C	1	1	0	0	1	0	0	0	1	0	0	1	0	0	1	0	0	1	0	0	0	1	0	0	1	0
Q_2	0	1	1	1	0	0	0	0	1	1	1	0	0	0	1	1	1	0	0	0	0	1	1	1	0	0

3. Deriving Checking Experiments for Two-State Latches

All latches described in Table 2-1 are typically implemented as two-state latches; the flow tables of these latches have only two rows. In this section, special properties of two-state flow tables are analyzed and a method for generating checking experiments for them is developed. This method was used to generate the checking experiments in Tables 2-2 through 2-11. The details are presented in sub-sections of this section.

Each cell in a flow table, a *total state*, corresponds to an assignment of values to the circuit inputs and internal states. A total state is an *unstable state* if it causes a change in internal state of the machine. A total state is a *stable state* if the next internal state is the same as the current internal state. The notation used in this paper for stable total states is shown in Table 3-1. A total state is *unspecified* if it is not adjacent to a stable total state. Such states cannot be reached because of the single-input change restriction on fundamental mode circuits [McCluskey 86]. Unspecified states are shown with "–" in the flow tables. A sequence *visits* a total state when the sequence applies the input of the total state while the machine is in the internal state of the total state. A total state is *identified* by a sequence if the sequence provides enough information to reconstruct the corresponding entry in the flow table.

Table 3-1 Notation: Stable Total States.

Notation	Definition
2	Output = 0
$\boxed{3}$	Output = 1

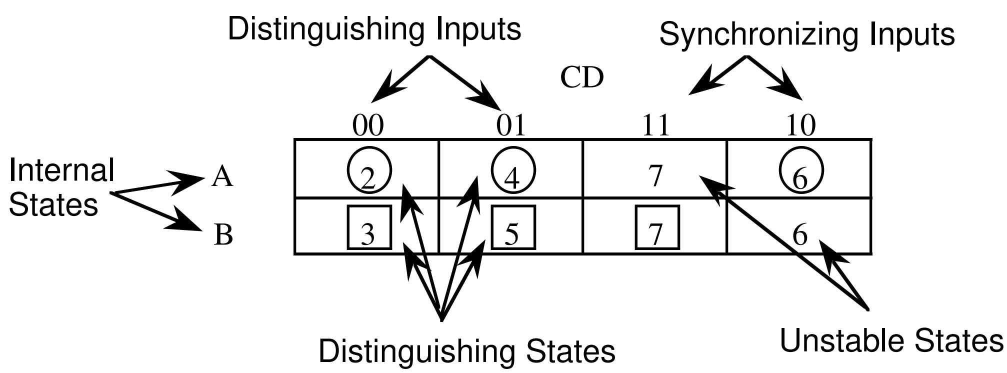

Figure 3-1 Definitions in Flow Table.

For a state machine to have sequential behavior, there must be at least one column in the flow table with different outputs. Otherwise, the machine acts as pure combinational logic. The total states in a column of a two-state flow table with differing outputs are called *distinguishing states*, and the input is called a *distinguishing input*. A state machine must also have two columns that change the internal state, so that both internal states are reachable. Inputs of such columns are called *synchronizing inputs* because they force the machine into a known state. These definitions are shown in Fig. 3-1.

As mentioned earlier, a checking experiment contains enough information to reconstruct the flow table. Therefore, it must identify all total states in the flow table. In this paper, we analyze flow tables that have distinguishing inputs and synchronizing inputs only (i.e. no columns in the table have two stable states with the same output), because all latches studied here have flow tables that fall into this class of flow tables. The requirement for reconstructing the flow table can be refined into three simpler requirements (proof of this will come later): all total states must be visited, all unstable states must be identified, and all distinguishing states must be identified.

We start with the first requirement, all total states must be visited. If a total state is not visited by the sequence, then the effect of applying the input of the total state when the machine is in the internal state of the total state is not known. This requirement is proven in Lemma 1.

Lemma 1: A checking experiment for a two-state flow table with only distinguishing and synchronizing inputs must visit all total states in the flow table.

Proof: Suppose that a sequence does not visit one of the stable total states. Create a second flow table by copying the original flow table and changing the output of the state not visited by the sequence. The output response of the sequence when applied to the second flow table would be the same as that of the original one because the sequence never enters the only total state that differs in the two flow tables. Since the two flow tables give the same response to the same input sequence, the sequence cannot be a checking experiment. Now suppose that the sequence does not visit one of the unstable states. In this case, the sequence would have the same response for a flow table that had the unstable total state replaced by a stable total state (the output does not matter). Since the input sequence has the same response for two flow tables, it cannot be a checking experiment. Therefore, a checking experiment must visit all total states (stable and unstable) in the flow table.

Even though visiting all the total states is a necessary requirement for a checking experiment, it is not a sufficient one. Consider the flow tables in Table 3-2. The sequence in Table 3-2c visits all the states of the two flow tables, Tables 3-2a and 3-2b, but the output response is the same for the two flow tables. Since the two flow tables are different, the sequence is not a checking experiment. Assuming that the flow table in Table 3-2a is the desired flow table, the sequence did not identify the unstable states 8 and 9. An unstable state is identified, when the sequence shows that the input caused a change in internal state. To show that an input causes a change in internal state, we need to show that the total states before and after the application of the input have different internal states. The internal state after the application of the input can be identified by following the input with a distinguishing input. The total state would then be a distinguishing state. The total state before the application of the input does not have to be a distinguishing state, but its internal state must be known. This analysis suggests the creation of state triples. A *state triple* is a set of three total states that contains the *setup state* (same internal state as unstable state and input unit distance from unstable state input), the total state associated with the unstable state, and a distinguishing state. The setup state and the distinguishing state must have different internal states. The last state of the triple is a distinguishing state. Visiting a distinguishing state does not change the internal state of the machine.

Table 3-2a Two-State Flow Table.

CD							
00	01	11	10	00	01	11	10
A=0				A=1			
2,0	4,0	7	6,0	9	8,0		
$\boxed{3}$,1	$\boxed{5}$,1	$\boxed{7}$,1	6	$\boxed{9}$,1	8		

Table 3-2b Another Flow Table That Produces Same Output Sequence When Table 3-2c Sequence Is Applied.

CD							
00	01	11	10	00	01	11	10
A=0				A=1			
2,0	4,0	7	6,0	$\boxed{9}$,1	8,0		
$\boxed{3}$,1	$\boxed{5}$,1	$\boxed{7}$,1	6	$\boxed{11}$,1	10,0		

Table 3-2c Sequence That Visits All Total States, But Is Not A Checking Experiment.

A	0	0	0	0	0	0	1	1
C	1	0	0	1	0	0	0	0
D	1	1	0	0	0	1	1	0
Q	1	1	1	0	0	0	0	1
State	7	5	3	6	2	4	8	9

Lemma 2: A checking experiment for a two-state flow table with only distinguishing and synchronizing inputs must identify all unstable states.

Proof: An unstable state is identified by a sequence if the sequence successively visits the three states of a triple corresponding to the unstable state. We have already shown that a checking experiment must visit all the total states, including the unstable states. Visiting an unstable state implies visiting the setup state and the unstable state of the triple. Therefore, a sequence that visits an unstable state but does not identify it is not visiting a distinguishing state after visiting the unstable state (i.e. it only visits the first two states of the triple). In this case, the input applied by the sequence after visiting the unstable state is a synchronizing input. Create a second flow table by copying the original flow table and changing the unidentified unstable state to a stable total state and give it the same output of the other total state in the same column. When the input corresponding to our unstable state is applied to either flow table, we get the same output. Since the next input is a synchronizing input, the next output and internal state will be the same for both flow tables. Therefore, a checking experiment must identify all unstable transitions.

Consider the flow tables in Table 3-3. Graphs for the state triples of Table 3-3a are shown in Fig. 3-2. The sequence shown in Table 3-3c visits all the total states, and identifies all the unstable states. However, both flow tables in Tables 3-3a and 3-3b, produce the same output response for the input sequence of Table 3-3c.

The sequence in Table 3-3c produces different outputs for both 00 and 01, indicating that they are distinguishing inputs. However, there are two possible permutations (barring isomorphism) for the distinguishing states in the flow table. These are shown in the first two columns of Tables 3-3a and 3-3b. To distinguish between the two flow tables, a sequence must have CD = 00,01 or CD = 01,00 as sub-sequences.

Table 3-3a Two-State Flow Table.

CD: 00	01	11	10	00	01	11	10
A=0				A=1			
2,0	4,0	7	6,0	9	8,0	–	–
$\boxed{3}$,1	$\boxed{5}$,1	$\boxed{7}$,1	6	$\boxed{9}$,1	8	–	–

Table 3-3b Another Flow Table That Produces Same Output Sequence When Table 3-3c Sequence Is Applied.

CD: 00	01	11	10	00	01	11	10
A=0				A=1			
2,0	$\boxed{5}$,1	$\boxed{7}$,1	6,0	9	8	–	–
$\boxed{3}$,1	4,0	7	6	$\boxed{9}$,1	8,0	–	–

Table 3-3c Sequence That Visits All Total States, Identifies Unstable States, But Is Not A Checking Experiment.

A	0	0	1	0	0	0	0	0	0	1	0
C	1	0	0	0	1	0	1	1	0	0	0
D	1	1	1	1	1	1	1	0	0	0	0
Q	1	1	0	0	1	1	1	0	0	1	1
State	7	5	8	4	7	5	7	6	2	9	3
Triples			A							D	
					B			C			

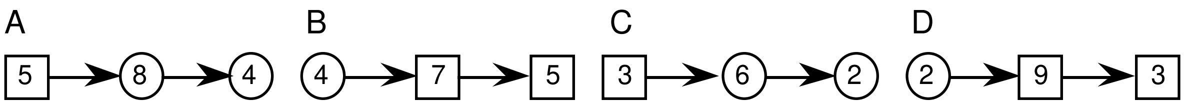

Figure 3-2 Graphs of State Triples for Flow Table in Table 3-3a.

This brings us to the third requirement: identifying the distinguishing states. As shown in this example, it is not enough to visit the distinguishing states. Two distinguishing inputs are said to be *linked* in a sequence, if the sequence provides enough information to determine the order of the distinguishing states in their two columns. If a distinguishing input follows another distinguishing input in the sequence, and the distinguishing states of both inputs are visited, then the two are linked. However, there can be many distinguishing inputs. Every distinguishing input can have one of two possible permutations of distinguishing states. Therefore, for n distinguishing inputs there are 2^n possible column permutations. Half of these permutations are nothing more than other permutations with the rows exchanged. Thus they do not need to be considered, and there are 2^{n-1} possible unique permutations. For a single distinguishing input, there is one unique permutation. Therefore, any distinguishing input is linked to itself, making the link relation reflexive. From the definition of link, link is a symmetric property. If a is linked to b, then b is linked to a. Now suppose that there are three distinguishing inputs a, b and c. If a is linked to b, then there is only one unique permutation for the distinguishing states in columns of a and b. Similarly, if a is linked to c, then there is only one unique permutation for the distinguishing states in columns of a and c. Therefore, there is one unique permutation for the all three columns, and so the link relationship is transitive. Since the link relationship is reflexive, symmetric, and transitive, it must be an equivalence relationship.

If a sequence links all pairs of distinguishing inputs then there can only be one permutation of the distinguishing states, and all distinguishing states are identified. Since the link relationship is an equivalence relationship, it suffices to show that the distinguishing inputs are all members of the same equivalence class. Therefore, distinguishing states are identified if they are all visited, and if they are all linked to each other.

Lemma 3: A checking experiment for a two-state flow table with only distinguishing and synchronizing inputs must identify all distinguishing states.

Proof: Distinguishing states are identified by a sequence if the sequence visits the states, and if all the distinguishing inputs are linked. We have already shown that a checking experiment must visit all the total states, including the distinguishing states. Suppose that two distinguishing inputs are not linked in a sequence. Since link is an equivalence relation between distinguishing inputs, the distinguishing inputs would fall into two equivalence classes. Within each of the classes, there is only one unique permutation of distinguishing states. Create a second flow table by copying the original flow table, and swapping the rows in the distinguishing input columns of one of the equivalence classes. Also, swap the rows of any synchronizing inputs that are a unit distance from any of the distinguishing inputs in that class. Applying the sequence to the new flow table would give the same response as when applied to the original flow table. Therefore, if the distinguishing inputs are not linked in a sequence, the sequence is not a checking experiment.

Now that we have shown that the three conditions (visiting all states, identifying all unstable states, and identifying all distinguishing states) are necessary for a sequence to be a checking experiment, we show that if all three conditions are satisfied that the sequence is a checking experiment. In other words, given a two-state flow table with distinguishing and synchronizing inputs, any sequence that satisfies all three conditions is guaranteed to be a checking experiment. These conditions are necessary and sufficient. The proof is given in Theorem 1. Lemma 2 and Lemma 3 showed that visiting a state was necessary to identify it. Thus visiting a state can be eliminated as an explicit condition for a sequence to be a checking experiment. However, the condition is retained here just for emphasis.

Theorem 1: A sequence for a two-state machine with distinguishing and synchronizing inputs is a checking experiment if and only if it satisfies the following properties:

1. Visits all the total states.
2. Identifies all the unstable states.
3. Identifies all the distinguishing states.

Proof: From Lemmas 1, 2 and 3, a checking experiment must satisfy all the above conditions. Now, we need to show that if a sequence satisfies the three conditions, then it is a checking experiment. If a sequence identifies all the distinguishing states, then there can only be one permutation of distinguishing states in the flow table. If the sequence also identifies all the unstable states, then all entries in the flow table are identified. Therefore only one flow table can be constructed from the response of the sequence, making it a checking experiment.

An important consequence of Theorem 1 is that a checking experiment does not have to ensure all possible transitions in a two-state flow table. For example, consider the flow table in Table 3-4. The graphs of the triples are shown in Fig. 3-3. A sequence formed by combining the triples is 6,7,5,3,6,2. Since state 3 follows state 5, the distinguishing inputs are linked. State 4 can be added to the end of the sequence to satisfy the first requirement of Theorem 1. Therefore, the state sequence becomes: 6,7,5,3,6,2,4. The transitions through this sequence are shown graphically in Table 3-4. Thick arrows are used to indicate the beginning and end of the sequence. The following six transitions are not included in this sequence: 4 -> 2, 2 -> 6, 4 -> 7, 3 -> 5, 5 -> 7, and 7 -> 6.

Table 3-4 Flow Table Marked With Checking Sequence.

	CD 00	01	11	10	Q
A	2	4	7	6	0
B	3	5	7	6	1

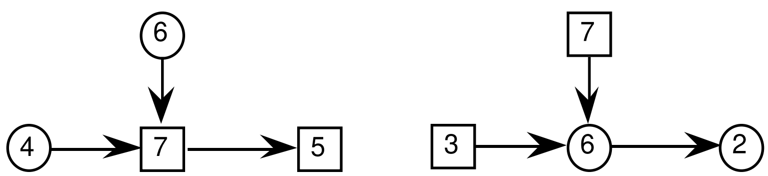

Figure 3-3 Graph of Triples for Flow Table in Table 3-4.

From Theorem 1, the following procedure can be used to generate a checking experiment for a two-state flow table with distinguishing and synchronizing inputs.

Procedure for deriving checking experiments from two-state flow tables.

1. Determine all the state triples for the unstable states.
2. Combine the triples of step 1.
3. If the sequence resulting from step 2 does not form a link among all the distinguishing inputs then modify the sequence so that it does, without destroying the triples.
4. Add any missing total states to the sequence.
5. Convert the state sequence into an input sequence, adding a synchronizing input, if necessary.

The first pattern in the sequence for a two-state flow table should force the machine into a known state. Therefore, in step 2 of the procedure, the setup state of the first triple should correspond to a synchronizing input whenever possible.

Combining triples would be most efficient if the distinguishing state of one triple is the setup state of another. Since triples cause a change in the internal state, the final state of one triple can be the setup state of another if the two triples cause internal state changes in the opposite directions (i.e. if the first triple causes the machine to change from internal state A to internal state B, then the second triple should cause the machine to change from internal state B back to internal state A). If there are more triples that cause state changes in one direction than in the other, then some of the state changes will need to be repeated in order to get to the setup states of all the triples.

Lower Bound on Sequence Length of Two-State Latches: The length of a checking experiment (L) is bound by the following equation.

$$L \geq S + 1 + \sum_{i=0}^{S} \max(n_i - 1, 0) \quad \text{if } v = 0$$

$$L \geq S + v + \sum_{i=0}^{S} \max(n_i - 1, 0) \quad \text{if } v > 0$$

where S = number of total States

n_i = number of times state i appears as the only distinguishing state of a triple

v = difference between the number of unstable states in the two rows

Proof: As seen from Lemma 1, a checking experiment must visit every total state. A machine can only be in one stable total state for every input pattern. Therefore, there must be at least as many patterns in the sequence as there are total states. There will always be at least one extra pattern for initialization. To be useful, the first pattern should force the machine into a known state (a synchronizing input). If a state s_i appears as the only distinguishing state of n_i triples, then s_i must appear at least n_i times. One of the occurrences of state s_i is accounted for in S (the total number of states). Therefore s_i must appear an additional n_i - 1 times. If s_i never appears as the only distinguishing state of a triple, then nothing should be added. Hence the term max(n_i - 1,0) is added for each total state. If the number of unstable states in one row differs from the number of unstable states in the other row by v, then at least v -1 extra internal state changes have to be applied. Each extra state change requires at least one more pattern. Adding v -1 to the initialization pattern gives v.

In many of the latch flow tables, half the inputs are distinguishing inputs, and the other half are synchronizing inputs. A single input variable determines whether an input is a distinguishing input or a synchronizing input. This class of state machines will be referred to as *single-input control state machines*, and the variable that determines the input type will be called a *control input*. For example, in Table 3-4, all inputs are distinguishing inputs when C = 0, and all inputs are synchronizing inputs when C = 1. Therefore, the flow table describes a single-input control state machine with C as the control input. An interesting property of such state machines is that the distinguishing state of one triple cannot be a setup state of another triple, because there are no unstable states adjacent to a distinguishing state of a triple. Therefore, if there are v more unstable states in one row than in the other, then $2(v -1)$ extra patterns are needed. This is used to derive a tighter lower bound on the length of the checking experiment.

Lower Bound on Sequence Length of Single-Input Control State Machine: The length of a checking experiment (L) of a Single-Input Control State Machine is bound by

$$L \geq S + 1 \qquad \text{if } v = 0$$

$$L \geq S + 2v - 1 \qquad \text{if } v > 0$$

where S = number of total States

v = difference between the number of unstable states in the two rows

Proof: The distinguishing state of one triple cannot be the setup state of another triple, because there are no unstable states adjacent to a distinguishing state of a triple. Therefore, the summation term in the original bound will always be 0. If v = 0, then the arguments for the previous bound apply. If v = 1, then two patterns are needed for each additional transition. Therefore 2(v -1) extra patterns are needed. Combining this with the initialization pattern gives 2v -1.

Another property of single-input control state machines is that a sequence that uses distinguishing states as setup states for all except the first triple, will identify the distinguishing states as well as the unstable states. This property is proved in Theorem 2.

Theorem 2: If a sequence is applied to a single-input control state machine, and the setup states of all but the first triple are distinguishing states, then the sequence links all the distinguishing inputs.

Proof: In a single-input control state machine each distinguishing input has a distinguishing state that is a setup state of a triple, and another distinguishing state that is a distinguishing state of the same triple. Therefore, there is a one-to-one correspondence between distinguishing inputs and triples. Now, if triple B is applied after triple A, the distinguishing input corresponding to triple A is linked to the one corresponding to triple B. Suppose triple C is applied after triple B, then the distinguishing input corresponding to triple B is linked to the one corresponding to triple C. Since link is an equivalence relation, the distinguishing input corresponding to triple A is linked to the one corresponding to triple C. Using the same arguments, it can be shown that the distinguishing input corresponding to triple A is linked to all the distinguishing inputs. Therefore, the sequence links all the distinguishing inputs.

3.1 Two-State SR-Latch

The equation for the SR-latch is $Q = S + \overline{R}q$. The latch is set when S = 1 and reset when R = 1. S and R should not be 1 at the same time. The flow table for the SR-latch is shown in Table 3.1-1. The graphs of the state triples are shown in Fig. 3.1-1.

Table 3.1-1 Flow Table for Two-State SR-Latch.

SR	00	01	11	10	Q
	2	4	-	5	0
	$\boxed{3}$	4	-	$\boxed{5}$	1

Figure 3.1-1 Graphs of State Triples for Two-State SR-Latch.

The setup state of one triple is the distinguishing state of the other triple. Therefore, the two triples can be combined without adding any states between them. Two possible state sequences are 2,5,3,4,2 and 3,4,2,5,3. Since there is only one distinguishing input, the distinguishing states are identified by simply visiting them. The two sequences contain all the total states in the flow table. The input sequence that would generate these state sequences is shown in Table 3.1-2. Since the sequences satisfy Theorem 1, they are checking experiments.

Table 3.1-2 Minimum-Length (6) Checking Experiments for Two-State SR-Latch.

<table>
<tr><td>S</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>S</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>R</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>R</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>Q</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>Q</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>State</td><td>4</td><td>2</td><td>5</td><td>3</td><td>4</td><td>2</td><td>State</td><td>5</td><td>3</td><td>4</td><td>2</td><td>5</td><td>3</td></tr>
<tr><td rowspan="2">Triples</td><td></td><td colspan="3">B</td><td colspan="2"></td><td rowspan="2">Triples</td><td></td><td colspan="3">A</td><td colspan="2"></td></tr>
<tr><td colspan="3"></td><td colspan="3">A</td><td colspan="3"></td><td colspan="3">B</td></tr>
</table>

Since the setup state in both triples is a distinguishing state, a synchronizing input needs to be added to the beginning of either sequence. The sequences in Table 3.1-2 are minimum length.

3.2 Two-State D-Latch

The equation for a D-latch is $Q = CD + \overline{C}q$, and the flow table for the D-latch is shown in Table 3.2-1. The graphs of the state triples are shown in Fig. 3.2-1. The two triples can be combined in any order. Suppose we start with the triple A. The first part of the state sequence is 6,7,5 (state 6 is picked because it is a synchronizing input). Now, looking at triple B, there are two choices: 7,6,2 or 3,6,2. State 7 has already been entered in the first sequence, thus entering it again has no benefit. Choosing state 3 as the setup state would make state 3 follow state 5, linking the two distinguishing inputs. Thus the sequence becomes 6,7,5,3,6,2. The sequence is missing state 4, which can be added to the end of the sequence. This makes the final sequence 6,7,5,3,6,2,4. The same approach can be used starting with triple B in Fig. 3.2-1. The sequence in that case would be 7,6,2,4,7,5,3. These two state sequences, and the checking experiments that would generate them, are shown in Table 3.2-2. Since the lengths of these sequences meet the lower bound, these sequences are minimum length.

Table 3.2-1 Flow Table for Two-State D-Latch.

CD: 00	01	11	10	Q
2	4	7	6	0
$\boxed{3}$	$\boxed{5}$	$\boxed{7}$	6	1

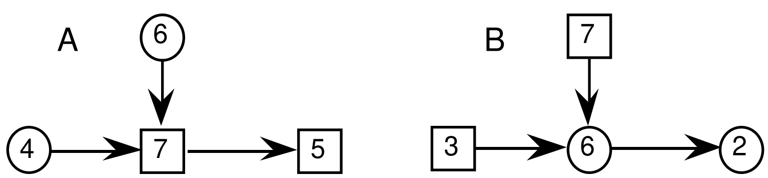

Figure 3.2-1 Graphs of State Triples for Two-State D-Latch.

Table 3.2-2 Minimum-Length (7) Checking Experiments for Two-State D-Latch.

C	1	1	0	0	1	0	0
D	1	0	0	1	1	1	0
Q	1	0	0	0	1	1	1
State	7	6	2	4	7	5	3
Triples	B			A			

C	1	1	0	0	1	0	0
D	0	1	1	0	0	0	1
Q	0	1	1	1	0	0	0
State	6	7	5	3	6	2	4
Triples	A			B			

3.3 Two-State D-Latch With Asynchronous Set/Reset

The equation for the D-latch with Asynchronous Set/Reset is $Q = \overline{R}(S + CD + \overline{C}q)$. The latch is set when S = 1 and reset when R = 1. S and R should not be 1 at the same time, and neither should be 1 when C = 1. When both R and S are 0, the latch behaves exactly like a D-latch. The flow table for this latch is shown in Table 3.3-1, and graphs of the state triples are shown in Fig. 3.3-1.

Table 3.3-1 Flow Table for Two-State Asynchronous Set/Reset Latch.

CD

00	01	11	10	00	01	11	10	00	01	11	10	00	01	11	10	
R = 0								R = 1								
S = 0				S = 1				S = 0				S = 1				Q
2	4	7	6	9	11	-	-	8	10	-	-	-	-	-	-	0
$\boxed{3}$	$\boxed{5}$	$\boxed{7}$	6	$\boxed{9}$	$\boxed{11}$	-	-	8	10	-	-	-	-	-	-	1

In Fig. 3.3-1 only the first two triples have synchronizing setup states. Therefore, a minimum-length sequence should start with one of these two triples. There are ten total states, and state 2 and state 5 appear twice as the only distinguishing inputs of state triples. From the first bound derived in Section 3, the minimum test length must be at least 13. State 4 and state 3 appear twice as setup states. However, in the first two triples there are alternate setup states. If these are used, then one of them could be the synchronizing input, but the other would appear twice in the sequence, raising the minimum length to 14. One such sequence is shown in Table 3.3-2. Since state 4 directly follows state 2, the distinguishing inputs are linked.

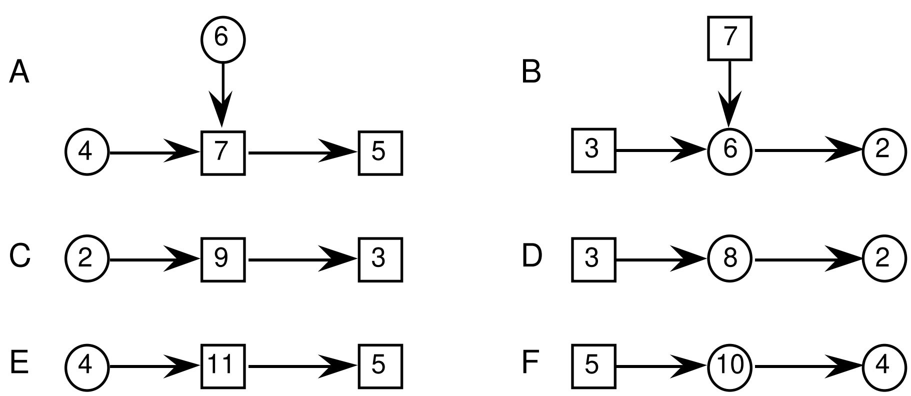

Figure 3.3-1 Graphs of State Triples for Two-State D-Latch With Asynchronous Set/Reset.

Table 3.3-2 A Minimum-Length (14) Checking Experiment for Asynchronous Set/Reset Latch.

C	1	1	0	0	0	0	0	0	0	0	0	0	1	0
D	1	0	0	0	0	0	0	1	1	1	1	1	1	1
R	0	0	0	0	0	1	0	0	0	0	1	0	0	0
S	0	0	0	1	0	0	0	0	1	0	0	0	0	0
Q	1	0	0	1	1	0	0	0	1	1	0	0	1	1
State	7	6	2	9	3	8	2	4	11	5	10	4	7	5
Triples	B				D					F				
			C					E				A		

3.4 Two-State MD-Latch

The equation for an MD-latch (Multiplexed-Data latch) is $Q = C(TS + \overline{T}D) + \overline{C}q$. When T = 0, the latch operates in normal mode (D is used as the input), and when T = 1 it uses S as input. The flow table for the MD-latch is given in Table 3.4-1. The graphs of the triples are shown in Fig. 3.4-1.

Looking at the flow table, all the distinguishing states occur when C = 0, and all the unstable states occur when C = 1. Therefore, the MD-latch is a single-input control state machine, and C is the control input. There are 24 total states, and the number of unstable states in each of the rows is 4. Therefore, the length of a checking experiment must be at least 25. Appendix A shows that the length must be at least 26. The details of combining the triples to derive minimum-length sequences also appear in Appendix A. One such sequence is shown in Table 3.4-2.

Table 3.4-1 Flow Table for Two-State MD-Latch.

DS	00	01	11	10	00	01	11	10	00	01	11	10	00	01	11	10	
	C = 0								C = 1								
	T = 0				T = 1				T = 0				T = 1				Q
	2	4	6	8	10	12	14	16	18	20	19	21	22	23	25	24	0
	$\boxed{3}$	$\boxed{5}$	$\boxed{7}$	$\boxed{9}$	$\boxed{11}$	$\boxed{13}$	$\boxed{15}$	$\boxed{17}$	18	20	$\boxed{19}$	$\boxed{21}$	22	$\boxed{23}$	$\boxed{25}$	24	1

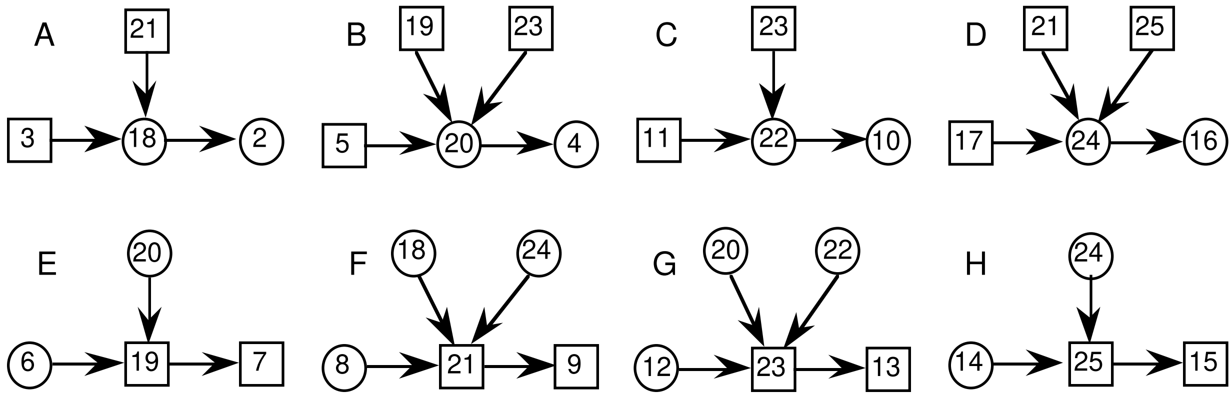

Figure 3.4-1 Graphs of State Triples for Two-State MD-Latch.

Table 3.4-2 A Minimum-Length (26) Checking Experiment for MD-Latch.

D	1	1	1	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
S	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1	0	0	0
T	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
C	1	1	0	0	1	0	0	0	1	0	0	1	0	0	1	0	0	1	0	0	0	1	0	0	1	0
Q	0	1	1	1	0	0	0	0	1	1	1	0	0	0	1	1	1	0	0	0	0	1	1	1	0	0
State	24	21	9	3	18	2	8	6	19	7	5	20	4	12	23	13	11	22	10	12	14	25	15	17	24	16
Triples	F			A				E			B			G			C				H			D		

3.5 Two-State Two-Port Latch

The equation for a Two-Port latch is $Q = C_1D_1 + C_2D_2 + \overline{C_1}\,\overline{C_2}q$. C_1 and C_2 should not be both active at the same time. The Two-Port latch loads the data input corresponding to the active control input. The flow table is shown in Table 3.5-1.

The last four columns are marked as don't cares because the operation of the latch is not defined when both control lines are active. Graphs of the triples are shown in Fig. 3.5-1. Since

Table 3.5-1 Flow Table for Two-State Two-Port Latch.

D_1D_2

00	01	11	10	00	01	11	10	00	01	11	10	00	01	11	10	
$C_2 = 0$								$C_2 = 1$								
$C_1 = 0$				$C_1 = 1$				$C_1 = 0$				$C_1 = 1$				Q
2	4	6	8	10	12	11	13	14	15	17	16	-	-	-	-	0
$\boxed{3}$	$\boxed{5}$	$\boxed{7}$	$\boxed{9}$	10	12	$\boxed{11}$	$\boxed{13}$	14	$\boxed{15}$	$\boxed{17}$	16	-	-	-	-	1

states 2 and 7 appear twice as distinguishing states of triples, and there are 16 total states, the lower bound on the test length is 19. Appendix B shows that the length must be at least 23. The details of combining the triples to derive minimum-length sequences also appear in Appendix B. One such sequence is shown in Table 3.5-2.

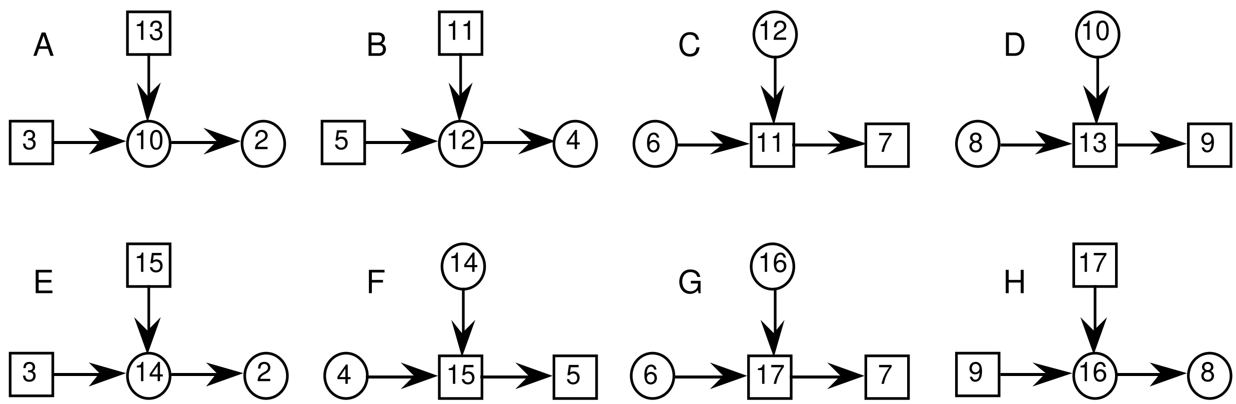

Figure 3.5-1 Graphs of State Triples for Two-State Two-Port Latch.

Table 3.5-2 A Minimum Length (23) Checking Experiment for Two-Port Latch.

D_1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	1	1	1
C_1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1	0
D_2	0	0	0	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
C_2	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0
Q	1	0	0	0	1	1	1	0	0	0	1	1	0	0	0	1	1	1	0	0	0	1	1
State	13	10	2	4	15	5	3	14	2	8	13	9	16	8	6	17	7	5	12	4	6	11	7
Triples	A			F			E			D													
												H			G			B			C		

3.6 Two-State Load Enable Latch

The equation for the Load Enable latch is $Q = CLD + \overline{(LC)}q$. When L = 0, the clock has no effect on the latch, and the latch retains the stored value. When L = 1, it behaves like a D-latch. The flow table for the Load Enable latch is given in Table 3.6-1. Graphs of the triples are shown in Fig. 3.6-1. Since there are 14 stable total states, a checking experiment must have at least 15 patterns. One such sequence is shown in Table 3.6-2. The details are in Appendix C.

Table 3.6-1 Flow Table for Two-State Load Enable Latch.

LD	00	01	11	10	00	01	11	10	
	C = 0				C = 1				Q
	2	4	6	8	10	12	15	14	0
	3	5	7	9	11	13	15	14	1

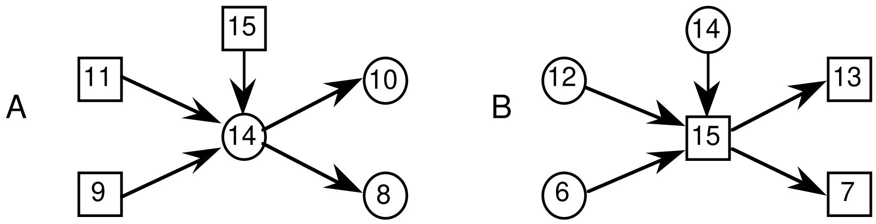

Figure 3.6-1 Graphs of State Triples of Two-State Load Enable Latch.

Table 3.6-2 A Minimum-Length (15) Checking Experiment for Load Enable Latch.

L	1	1	0	0	1	1	0	0	1	0	0	1	1	0	0
D	1	0	0	0	0	1	1	1	1	1	1	1	0	0	0
C	1	1	1	0	0	0	0	1	1	1	0	0	0	0	1
Q	1	0	0	0	0	0	0	0	1	1	1	1	1	1	1
State	15	14	10	2	8	6	4	12	15	13	5	7	9	3	11
Triples		A							B						

3.7 Two-State D-Enable Latch

The equation for the D-Enable latch is $Q = CDE + \overline{C}q$. A D-Enable latch operates as a D-latch when E = 1. When E = 0, the D-Enable latch will load 0. The flow table for the D-Enable latch is shown in Table 3.7-1. The graphs of the state triples are shown in Fig 3.7-1.

Table 3.7-1 Flow Table for Two-State D-Enable Latch.

DE	00	01	11	10	00	01	11	10	
	C = 0				C = 1				Q
	2	4	6	8	10	12	11	14	0
	3	5	7	9	10	12	11	14	1

Looking at the flow table, all the distinguishing states occur when C = 0, and all the unstable states occur when C = 1. Therefore, the D-Enable latch is a single-input control state machine, and C is the control input. There are 12 total states, one unstable state in the first row, and three in the second. Therefore, the length of a checking experiment must be at least 15. Appendix D shows that the length must be at least 16. One possible minimum-length sequence is shown in Table 3.7-2. Details of deriving a checking experiment are given in Appendix D.

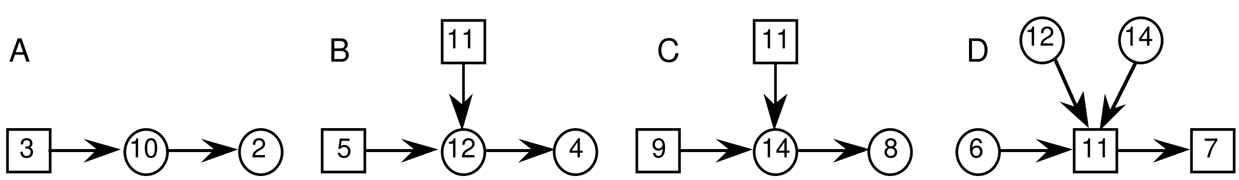

Figure 3.7-1 Graphs of State Triples for Two-State D-Enable Latch.

Table 3.7-2 A Minimum-Length (16) Checking Experiment for Two-State D-Enable Latch.

D	1	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0
E	1	1	1	1	1	0	0	1	1	1	0	0	1	0	0	0
C	1	1	0	0	1	1	0	0	1	0	0	0	0	0	1	0
Q	1	0	0	0	1	0	0	0	1	1	1	1	1	1	0	0
State	11	12	4	6	11	14	8	6	11	7	9	3	5	3	10	2
Triples		B				C			D						A	

3.8 Two-State XOR Input Latch

The equation for the XOR input latch is $Q = C(D \oplus S) + \overline{C}q$. The data loaded into the latch is $D \oplus S$. The flow table is given in Table 3.8-1. Since there are 12 stable states in the flow table, a checking experiment must have at least 13 patterns. Graphs of the state triples are shown in Fig. 3.8-1.

Table 3.8-1 Flow Table for Two-State XOR Latch.

DS	00	01	11	10	00	01	11	10	
	C = 0				C = 1				Q
	(2)	(4)	(6)	(8)	(10)	11	(12)	13	0
	[3]	[5]	[7]	[9]	10	[11]	12	[13]	1

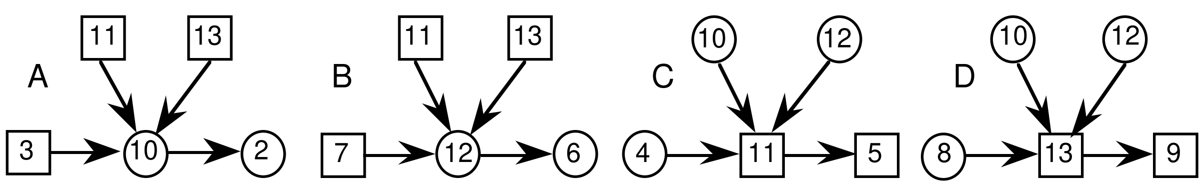

Figure 3.8-1 Graphs of State Triples for Two-State XOR latch.

Starting with any triple, there are two possible sequences. For example, starting with A, the two sequences are ACBD and ADBC. Since there are four triples, and any of them can be picked as the starting point, there are 8 sequences. Note that a sequence that connects the triples not only demonstrates all the unstable states, but also visits all the stable states. The transition from one triple to the other forms a link among all distinguishing inputs. The sequences formed by connecting the triples are all minimum-length checking experiments. One of the minimum-length checking experiments is given in Table 3.8-2.

Table 3.8-2 A Minimum-Length (13) Checking Experiment for XOR Input Latch.

D	0	0	0	0	0	0	1	1	1	1	1	1	0
S	1	0	0	1	1	1	1	1	1	0	0	0	0
C	1	1	0	0	1	0	0	1	0	0	1	0	0
Q	1	0	0	0	1	1	1	0	0	0	1	1	1
States	11	10	2	4	11	5	7	12	6	8	13	9	3
Triples	A			C			B			D			

3.9 Two-State BILBO Latch

The equation for a BILBO latch is $Q = C(B_1D \oplus \overline{B_2}S) + \overline{C}q$. Based on the setting of B_1 and B_2, the latch can be configured to load D (when B_1B_2 = 11), reset the latch (when B_1B_2 = 01), load S (when B_1B_2 = 00), and load $S \oplus D$ (when B_1B_2 = 10). The flow table for the BILBO latch is given in Table 3.9-1.

Table 3.9-1 Flow Table for Two-State BILBO Latch.

DS (C = 0)

00	01	11	10	00	01	11	10	00	01	11	10	00	01	11	10	
$B_2 = 0$								$B_2 = 1$								
$B_1 = 0$				$B_1 = 1$				$B_1 = 0$				$B_1 = 1$				Q
2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	0
$\boxed{3}$	$\boxed{5}$	$\boxed{7}$	$\boxed{9}$	$\boxed{11}$	$\boxed{13}$	$\boxed{15}$	$\boxed{17}$	$\boxed{19}$	$\boxed{21}$	$\boxed{23}$	$\boxed{25}$	$\boxed{27}$	$\boxed{29}$	$\boxed{31}$	$\boxed{33}$	1

DS (C = 1)

00	01	11	10	00	01	11	10	00	01	11	10	00	01	11	10	
$B_2 = 0$								$B_2 = 1$								
$B_1 = 0$				$B_1 = 1$				$B_1 = 0$				$B_1 = 1$				Q
35	34	36	37	38	40	42	44	39	46	41	48	50	52	43	45	0
$\boxed{35}$	34	36	$\boxed{37}$	38	40	42	44	$\boxed{39}$	46	$\boxed{41}$	48	50	52	$\boxed{43}$	$\boxed{45}$	1

Looking at the flow table, all the distinguishing states occur when C = 0, and all the unstable states occur when C = 1. Therefore, the BILBO latch is a single-input control state machine, and C is the control input. There are 48 total states, and four more unstable states in the second row than in the first. Therefore, the length of a checking experiment must be at least 55. Appendix E shows that the length must be at least 58. One possible minimum-length sequence is shown in Table 3.9-2. Details of deriving a checking experiment are given in Appendix E.

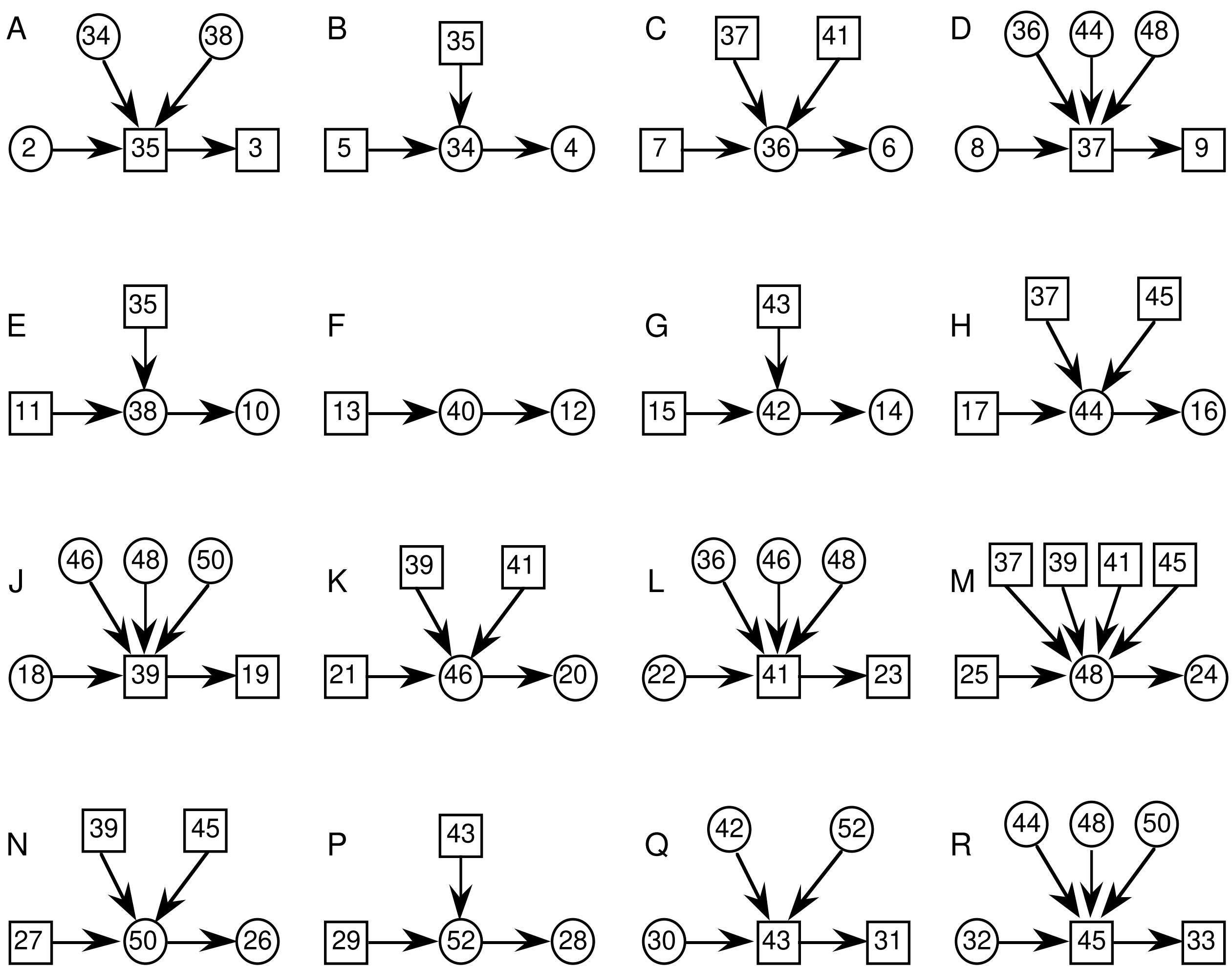

Figure 3.9-1 Graphs of State Triples for Two-State BILBO Latch.

Table 3.9-2 A Minimum-Length (58) Checking Experiments for Two-State BILBO Latch.

D	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1
S	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
B_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
B_1	0	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
C	1	1	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
Q	1	0	0	0	1	1	1	0	0	0	0	1	0	0	0	1	1	1	0	0	0	1	1	1	0	0	0	1	1
State	35	38	10	2	35	3	5	34	4	2	35	17	44	16	32	45	33	27	50	26	18	39	19	21	46	20	22	41	23
Triples	E			A			B					H			R			N			J			K			L		
D	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1	1	0	0	0	0
S	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1
B_2	0	0	0	0	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
B_1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
C	0	1	0	0	1	0	0	1	0	0	1	1	1	0	0	1	0	0	1	0	0	1	0	0	0	0	0	1	0
Q	1	0	0	0	1	1	1	0	0	0	1	1	0	0	0	1	1	1	0	0	0	1	1	1	1	1	1	0	0
State	7	36	6	8	37	9	25	48	24	32	45	43	42	14	30	43	31	29	52	28	30	43	31	15	17	11	13	40	12
Triples	C			D			M					G			Q			P									F		

3.10 Two-State CBILBO Latches

CBILBO latches are an extension of BILBO latch that can operate simultaneously as a pseudo-random pattern generator and a signature analyzer. Each of the two CBILBO latches has a different mode signal. The first latch loads S when $B_1 = 1$, and loads $\overline{S \oplus D}$ when $B_1 = 0$. The second latch is an MD-latch, with B_2 as the select input, S and D as the data inputs. Since the MD-latch has already been analyzed, only the first latch is considered in this section. The equation of this latch is $Q_1 = C(\overline{B_1}D \oplus S) + \overline{C}q_1$, and the flow table is given in Table 3.10-1. The graphs of the state triples are shown in Fig. 3.10-1.

Looking at the flow table, all the distinguishing states occur when C = 0, and all the unstable states occur when C = 1. Therefore, the CBILBO latch is a single-input control state machine, and C is the control input. Since there are 24 total states, the length of a checking experiment must be at least 25. An example of a minimum-length sequence is shown in Table 3.10-2. Details of deriving a checking experiment are given in Appendix F.

Table 3.10-1 Flow Table for Two-State CBILBO.

DS																
00	01	11	10	00	01	11	10	00	01	11	10	00	01	11	10	
C = 0								C = 1								
$B_1 = 0$				$B_1 = 1$				$B_1 = 0$				$B_1 = 1$				Q_1
2	4	6	8	10	12	14	16	18	19	20	21	22	23	25	24	0
3	5	7	9	11	13	15	17	18	19	20	21	22	23	25	24	1

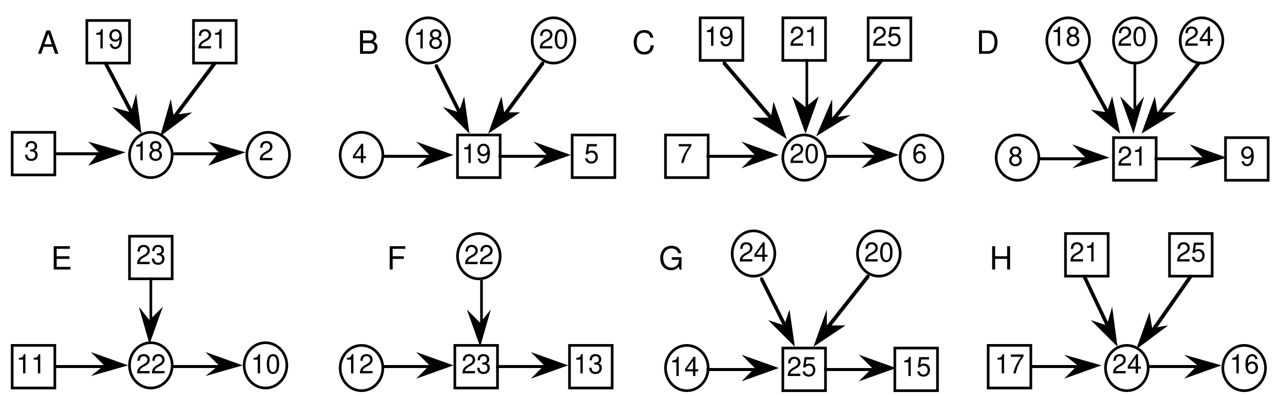

Figure 3.10-1 Graphs of State Triples for Two-State CBILBO Latch.

Table 3.10-2 A Minimum-Length (25) Checking Experiment for CBILBO Latch.

D	1	1	1	1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0
S	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0
B_1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
C	1	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	0	1	0	0	1	0
Q_1	0	1	1	1	0	0	0	1	1	1	0	0	0	1	1	1	0	0	0	0	1	1	1	0	0
State	24	25	15	7	20	6	4	19	5	3	18	2	8	21	9	17	24	16	14	12	23	13	11	22	10
Triples	G			C			B			A			D			H				F			E		

4. D-Latch Simulation

In this section, three different tests for the D-latch simulated using HSpice are compared. The first test is a pin fault test set, which targets the faults on the input and output of the D-latch. A D-latch can be viewed as a multiplexer that selects between D and Q, with C being the select signal. The second test is a multiplexer-based test. Patterns for testing multiplexers can be found in [Makar 88]. The third test is the checking experiment derived in Section 3.2. The three tests are shown in Fig. 4-1. The implementation used for the simulation is shown in Fig. 4-2.

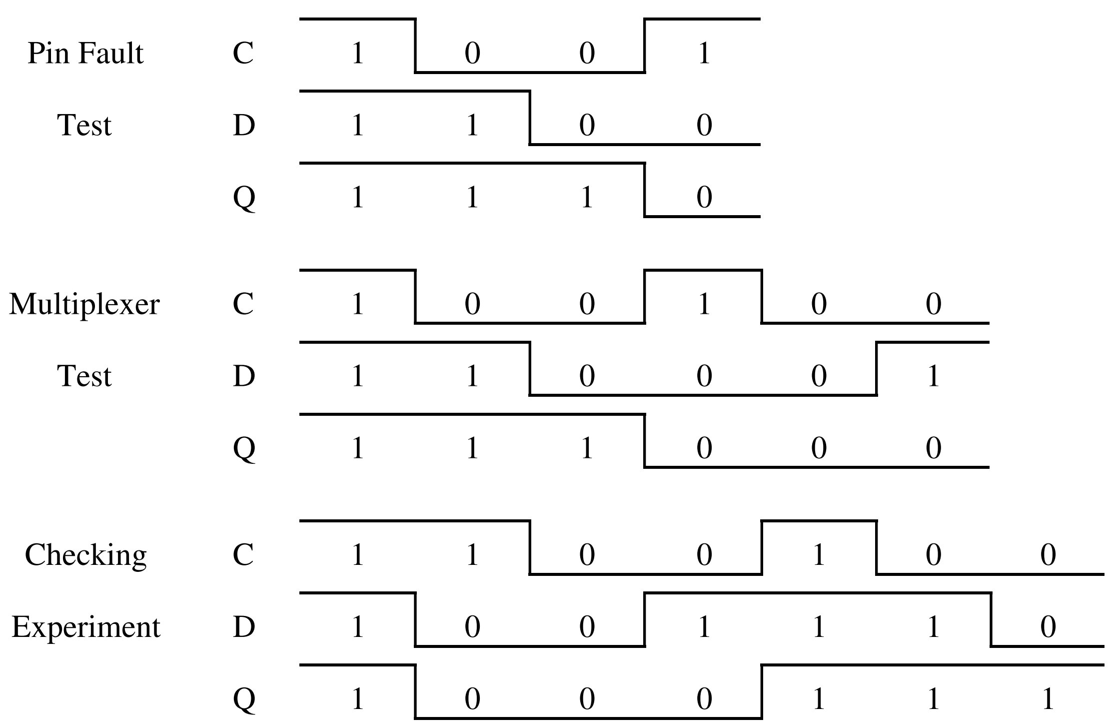

Figure 4-1 Test Sequences for Simulating D-Latch.

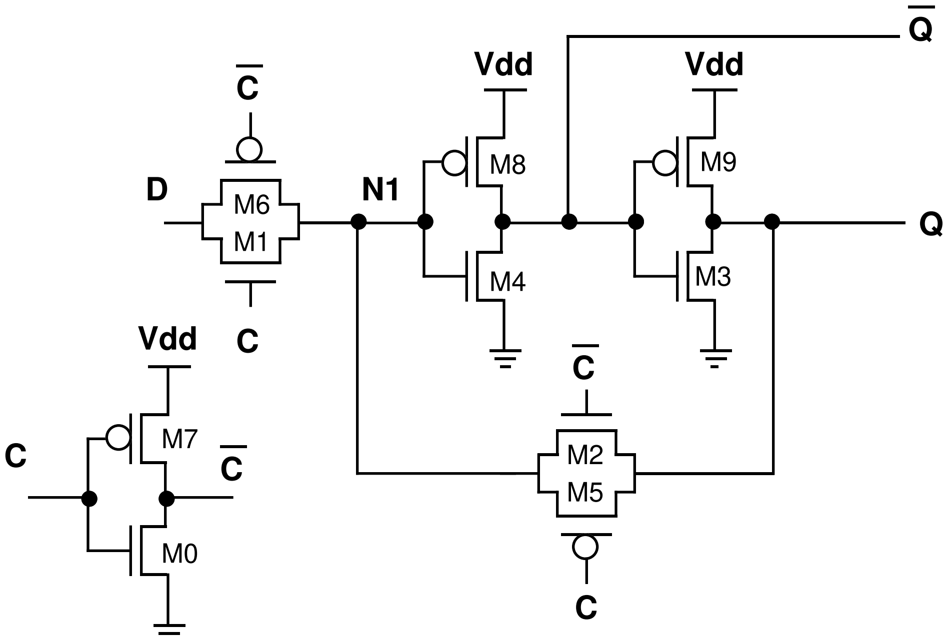

Figure 4-2 Transmission Gate D-Latch.

In the simulation, faults are injected by modifying the circuit description. The fault models used are based on the CrossCheck fault models [Sucar 89] and [Chandra 93]. The faults injected are shorted interconnects (STI), open interconnects (OPI), short-to-power (STP), short-to-ground (STG), transistor stuck-on (SON), and transistor stuck-open (SOP).

In CMOS, there are some faults whose presence does not change the functionality of the host circuit. Some of these cannot be detected (and thus are untestable or redundant). Others that cannot be detected by a Boolean voltage test (since the circuit functionality is correct) can, nevertheless, be discovered by a current test [Ma 95]. The simulations reported here record whether tests caused excessive supply current or incorrect outputs.

The current limit for IDDQ testing is often determined experimentally, by plotting the values of many good and bad die, and selecting an appropriate threshold that would detect as many faulty circuits as possible without discarding many good ones [Hawkins 89] and [Perry 92]. Fig. 4-3 shows the IDDQ distribution for circuits with the faults described above when the checking experiment was applied. Here IDDQf refers to the IDDQ value of a circuit with a fault present, and IDDQg refers to the IDDQ value of the fault-free circuit. The graph plots the ratio of IDDQf / IDDQg (i.e. the ratio of IDDQ increase), versus the faults. The graph shows that all but 7 of the faults cause an IDDQ that is over 100 thousand times that of the normal current. Using this graph, a threshold of 32 uA was selected. Maximum IDDQ for the fault-free circuit was about 320 pA.

The results of the simulations are shown in the graph of Fig. 4-4. For each test three numbers are reported: the number of faults detected by either a voltage or current test, the number of faults detected if the voltage measurement is used alone, and the number of faults detected if the current measurement is used alone. In spite of the fact that the checking experiments were generated to verify the functionality of the latches, the graph shows that, in addition to detecting functional faults, they are very useful in detecting faults that only cause excessive current. One drawback to the current tests is the large amount of time they require.

The graph also shows that the pin fault test and the multiplexer test miss several faults that are detected by the checking experiment. The faults missed by each of the tests are shown graphically in Fig. 4-5 through 4-7. In these figures white ovals indicate SOP or OPI faults, black ovals indicate SON faults, and thick black lines indicate STI faults. All STP and STG faults are detected by all three tests.

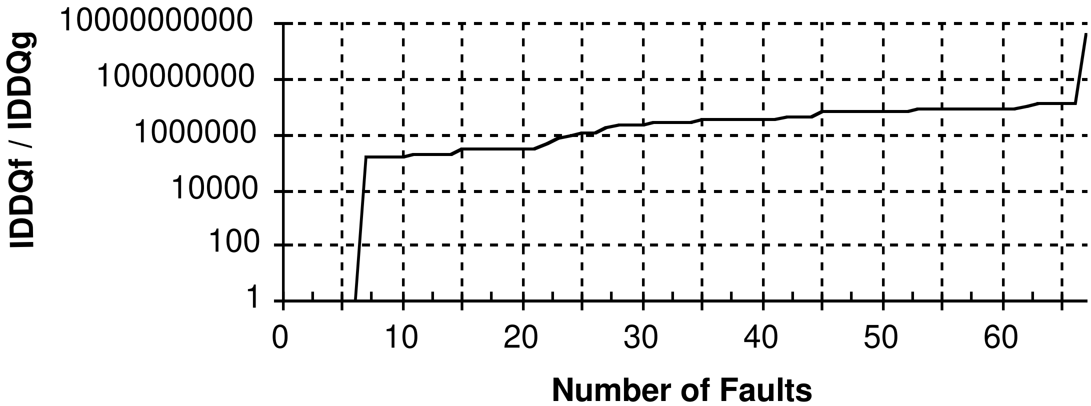

Figure 4-3 IDDQf / IDDQg Distribution.

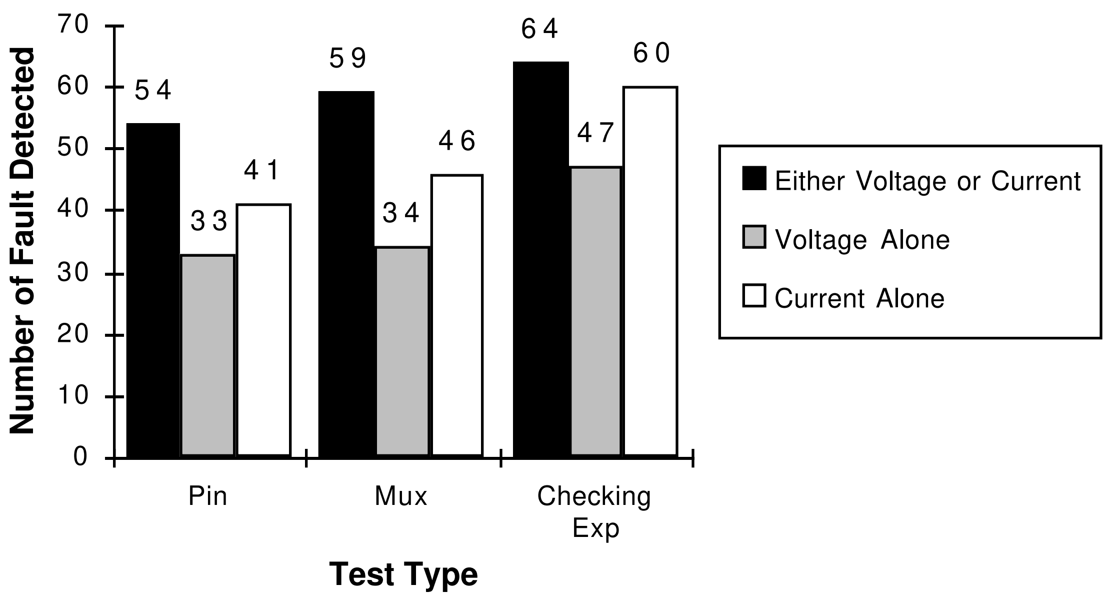

Figure 4-4 Fault Coverage of the Three Test Sets (Total Faults = 67).

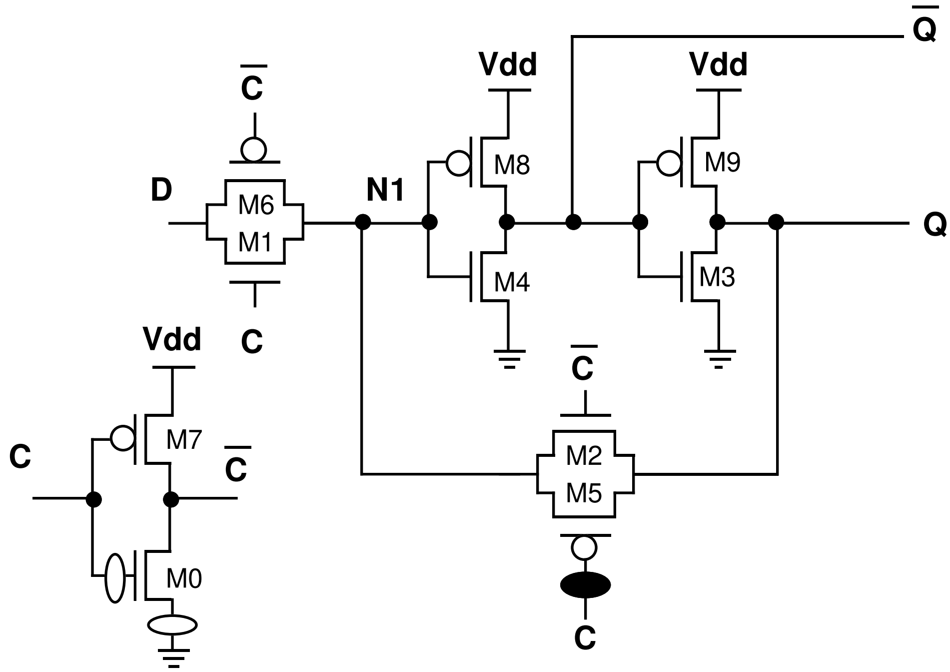

Figure 4-5 Faults Missed by Checking Experiment.

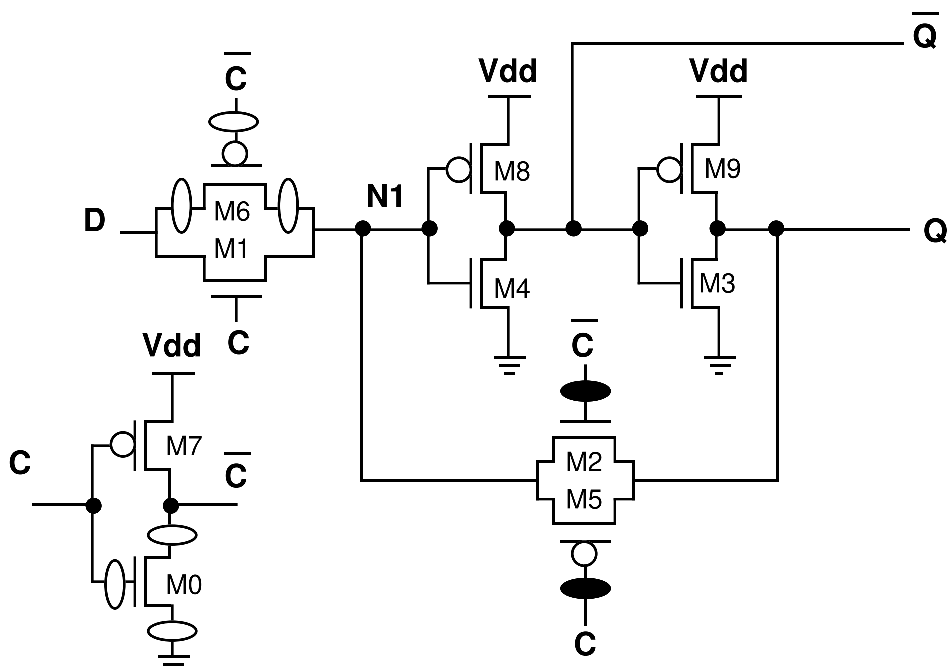

Figure 4-6 Faults Missed by Multiplexer Test.

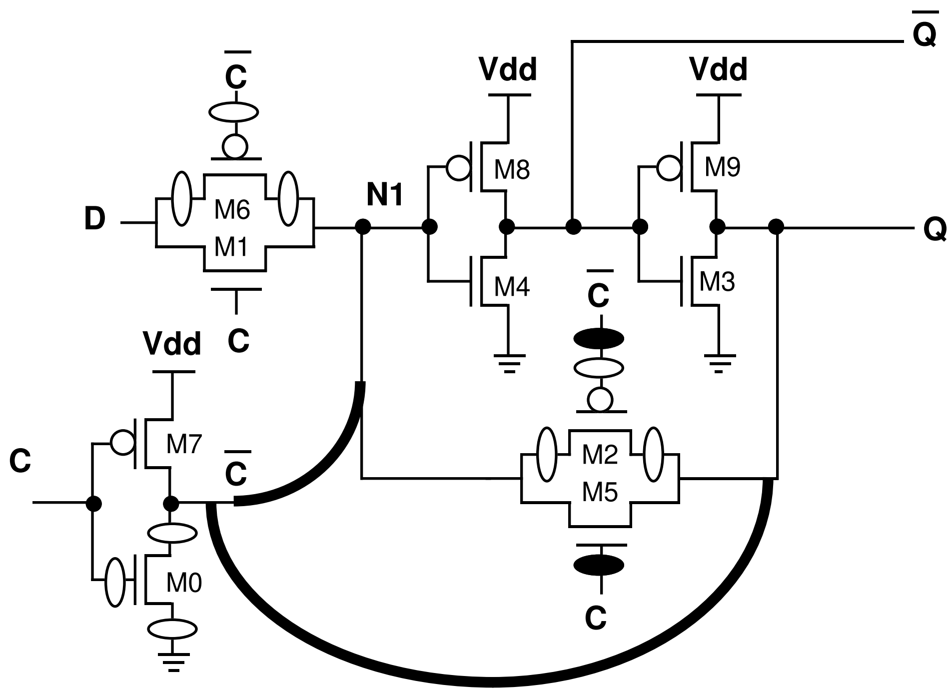

Figure 4-7 Faults Missed by Pin Fault Test.

The faults missed by the checking experiment fall into one of two categories. The faults on the M0 transistor cause the number of states to increase. With the presence of the fault, the application of CD = 10 when in state 7 (see Table 4-1), produces an output of 0. However, when the same input is applied to state 3, the output is 1. Therefore, states 7 and 3 cannot be in the same internal state, and there must be a third internal state. The stuck-on fault on M5 is not detected because it is on the feedback path. Even though not indicated in the figures, the NMOS transistors of the transmission gates are stronger than their PMOS counterparts. This makes it possible to detect M2 stuck-on, but not detect M5 stuck-on.

Table 4-1 Flow Table for Two-State D-Latch.

CD				Q
00	01	11	10	
2	4	7	6	0
$\boxed{3}$	$\boxed{5}$	$\boxed{7}$	6	1

As mentioned earlier, some faults are only detected by current measurement. In the D-latch, there are two kinds of faults that are detected by current and not by voltage. A stuck-on fault in M8 or M9 would cause a large IDDQ current when the corresponding NMOS transistor is turned on. However, due to the sizing ratio between the two transistors, the output voltage value would not be affected enough to be detected. In fact, the inverter would behave as an

NMOS inverter rather than a CMOS inverter. A plot of the relevant voltages and currents in the circuit with M8 stuck-on fault are shown in Fig. 4-8. In these graphs, $\overline{Q}$ has a fault-free value of 0 volts between 10 and 40 ms. With the presence of the fault, this voltage is around 1 volt, which is not high enough to be detected by a voltage test. In the same time frame, the IDDQ current reaches 2.5 mA. The other kind of faults that are only detected by current measurements are the stuck-open faults on M6 and M1, and faults on transmission gate in the feedback path. These faults cause voltage degradation on N1 which in turn causes M8 or M4 to turn on when they should not, raising the IDDQ current. A plot of the relevant voltages and currents in the circuit with M1 stuck-open fault are shown in Fig. 4-9. For the circuit with M5 stuck-open fault, the voltages and currents are shown in Fig. 4-10. In this case, the current reaches around 50 uA. Even though this value is much lower than that of M8 stuck-on fault, it is still above the threshold of 32 uA.

Some faults can only be detected by voltage measurements. Stuck-open faults on M3, M4, M8, and M9 do not substantially increase the IDDQ current. Also, a STI fault between D and N1 does not increase the IDDQ current.

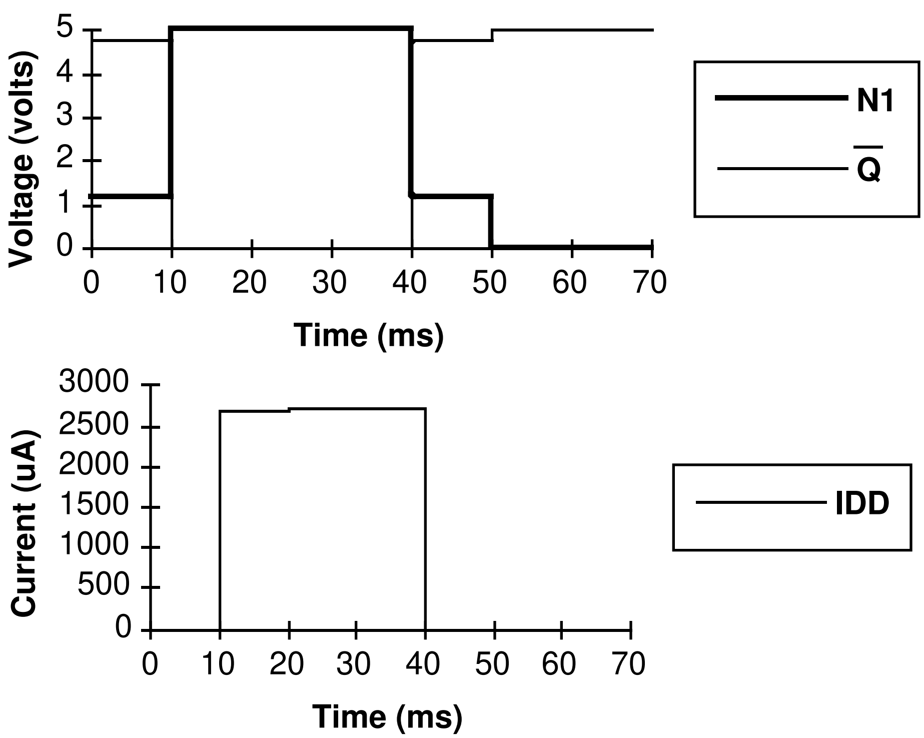

Figure 4-8 HSpice Output for M8 Stuck-On.

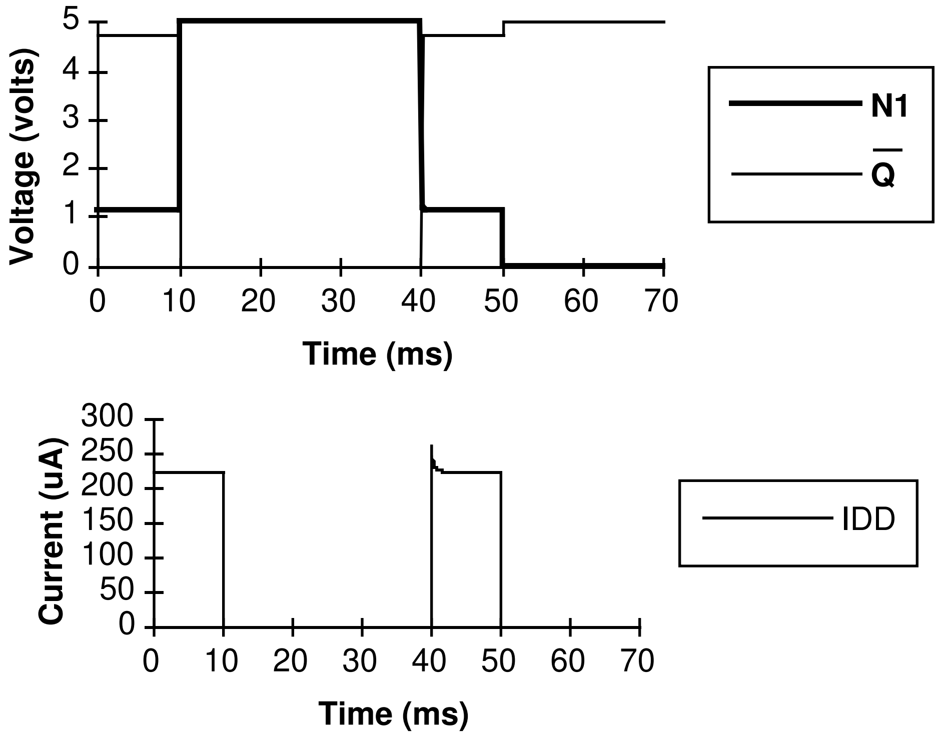

Figure 4-9 HSpice Output for M1 Stuck-Open.

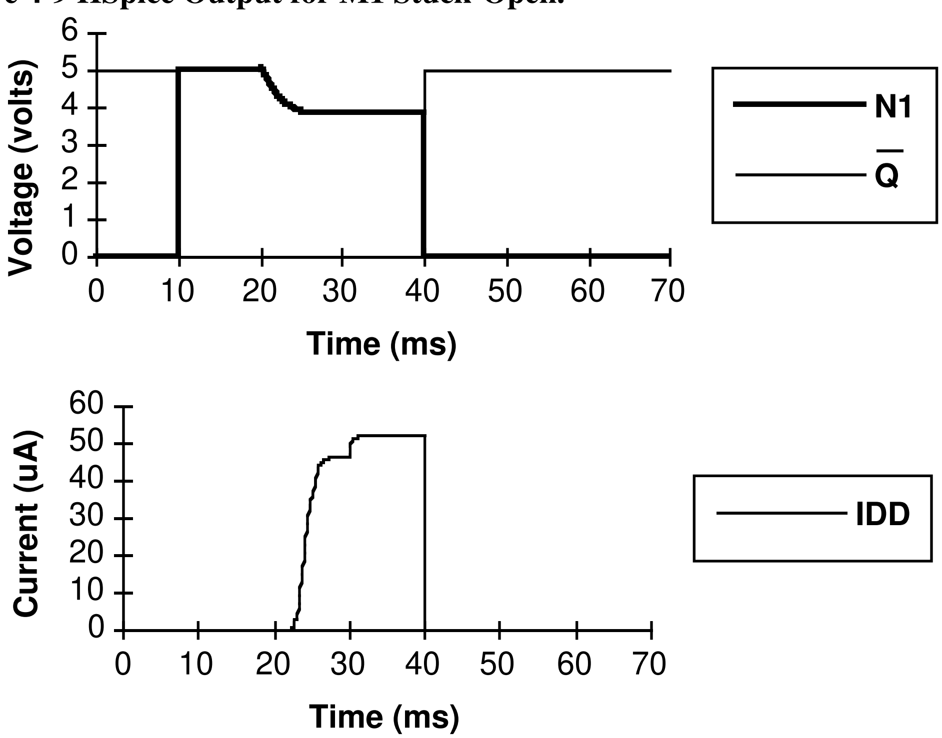

Figure 4-10 HSpice Output for M5 Stuck-Open.

5. Conclusions

Ten latch types were studied. For each, we derived requirements that can be used to verify whether or not a given sequence is a checking sequence. In addition, one example of a minimum-length checking sequence is derived for each latch type.

These checking sequences are guaranteed to detect any latch defects that do not increase the number of states and do not cause the latch operation to depend on other elements in the design (coupling or transition faults). To investigate whether the increase-in-state restriction could be a problem, one latch implementation was simulated using HSpice for several faults that could cause extra states. The checking sequence missed only two faults that cause an increase in state. A test set for the pin faults of the latch, and a test set for the multiplexer in the latch were also simulated. These two tests missed faults that were detected by the checking sequence.

These checking sequences are, in fact, shorter than some of the popular "rule of thumb" sequences such as "read 0 to 1, 1 to 0, 0 to 1, 1 to 1" and also shorter than sequences derived by ensuring that all possible transitions in the flow table are activated by the test sequence.

We feel that the checking sequences are a thorough, efficient technique for testing latches.

6. Acknowledgments

The authors would like to thank Jonathan Chang and Erin Kan for their valuable comments. This work was supported in part by the Ballistic Missile Defense Organization, Innovative Science and Technology (BMDO/IST) Directorate and administered through the Department of the Navy, Office of Naval Research under Grant No. N00014-92-J-1782, in part by the Advanced Research Projects Agency under Contract No. DABT63-94-C-0045, and in part by the National Science Foundation under Grant No. MIP-9107760. It was also funded in part by Cirrus Logic.

7. References

[Al-Assadi 93] Al-Assadi, W.K. "Faulty Behavior of Storage Elements and Its Effects on Sequential Circuits", *IEEE Transactions on VLSI*, Vol. 1, No. 4, December, 1993.

[Chandra 93] Chandra, S et. al. "CrossCheck: An Innovative Testability Solution," *IEEE Design and Test,* pp. 56-68, June, 1993.

[Friedman 71] Friedman, A.D. and P.R. Menon, Fault Detection in Digital Circuits, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971.

[Hawkins 89] Hawkins, CF. "Quiescent Power Supply Current Measurement for CMOS IC Defect Detection," *IEEE Trans. on Industrial Electronics*, pp. 211-218, May 89.

[Hennie 64] Hennie, F.C. "Fault Detecting Experiments for Sequential Circuits," *Proc. of the Fifth Annual Switching Theory and Logical Design Symposium*, S-164, Princeton, N.J., pp. 95-110, 1964.

[Lee 90] Lee, K.J. and M.A. Breuer, “A Universal Test Sequence for CMOS Scan Registers,” *IEEE Custom Integrated Circuits Conference*, pp. 28.5.1-28.5.4, 1990.

[Makar 88] Makar, S.R., and E.J. McCluskey, "On the Testing of Multiplexers," *Proc. Int. Test Conf.*, pp. 669-679, 1988.

[Ma 95] Ma, S., P. Franco and E.J. McCluskey, “An Experimental Chip to Evaluated Test Techniques, Experimental Results,” submitted to *ITC* 1995.

[McCluskey 86] McCluskey, E.J., *Logic Design Principles*, Prentice-Hall, New Jersey, 1986.

[Perry 92] Perry, R.. “IDDQ Testing in CMOS Digital ASICs - Putting It All Together,” *Proc. Int. Test Conf.*, pp. 151-157, 1992.

[Reddy 86] Reddy, M.K.and, S.M. Reddy, “Detecting FED Stuck-Open Faults in CMOS Latches and Flip-Flops,” *IEEE Design and Test*, pp. 17-26,1986.

[Saxena 93] Saxena, N. R. et. al., “Algorithmic Synthesis of High Level Tests for Data Path Designs,” *FTCS*, pp. 360-369, 1993.

[Sucar 89] Sucar, H., “High Performance Test Generation for Accurate Defect Models in CMOS Gate Array Technology,” *ICCAD*, pp. 166-169.

Appendix A Details of the Two-State MD-Latch

The graphs of the state triples for the MD-latch are shown in Fig. A-1. The flow table is given in Table A-1. The lower bound on the length of the checking experiment is 25, since there are 24 total states. Fig. A-2 shows the relationship between the triples. In this graph, each node represents a triple from Fig. A-1. An arrow indicates that the distinguishing states of the first triple is adjacent (differ in one input variable) to the setup state of the second triple. The graph consists of two disjoint parts which implies that at least one extra pattern is needed to connect all the triples. This implies that at least one additional pattern is needed, raising the minimum length to 26.

Fig. A-3 shows how the total states corresponding to the extra patterns connect the sub-graphs. Starting with A, C, E or H would mean only one transition between the two sub-graphs. Since the first triple starts with a synchronizing input, extra patterns would be needed to apply the distinguishing setup state of the first triple. For example, suppose we start with triple E. This would make the sequence end with triple A in state 2. Then adding state 6 to the end of the sequence would require two additional inputs, raising the length to 27. If the triple used to start the sequence has a distinguishing setup state that connects the two sub-graphs, then no additional inputs are needed. The triples B, D, F and G satisfy this criteria. For example, the distinguishing setup state of triple B is state 5. This happens to be the one of the states than can connect E to A. However, starting in state B, D, F, or G requires an extra input to go back to the original sub-graph making the length 26, the minimum length. Since there are four triples that can be split, and in each case there are two choices for states between the sub-graphs (note that there aren't four, because the original starting state of the starting triple must be one of the two states between the two sub-graphs), then there are 8 minimum-length checking experiments. Since a distinguishing state is used as the setup state for all the triples, then, from Theorem 2, any sequence formed by the graph corresponds to a checking experiment. One such checking experiment was shown in Table 3.4-2 in Section 3.4.

Table A-1 Flow Table for Two-State MD-Latch.

DS

00	01	11	10	00	01	11	10	00	01	11	10	00	01	11	10	
C = 0								C = 1								
T = 0				T = 1				T = 0				T = 1				Q
2	4	6	8	10	12	14	16	18	20	19	21	22	23	25	24	0
$\boxed{3}$	$\boxed{5}$	$\boxed{7}$	$\boxed{9}$	$\boxed{11}$	$\boxed{13}$	$\boxed{15}$	$\boxed{17}$	18	20	$\boxed{19}$	$\boxed{21}$	22	$\boxed{23}$	$\boxed{25}$	24	1

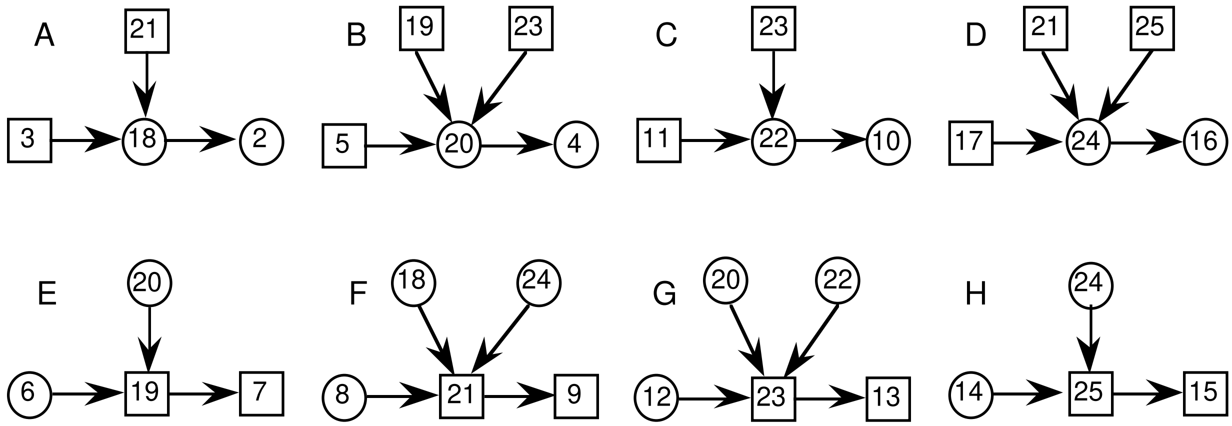

Figure A-1 Graphs of State Triples for Two-State MD-Latch.

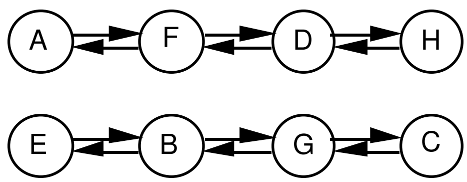

Figure A-2 Constraint Graph for State Triples.

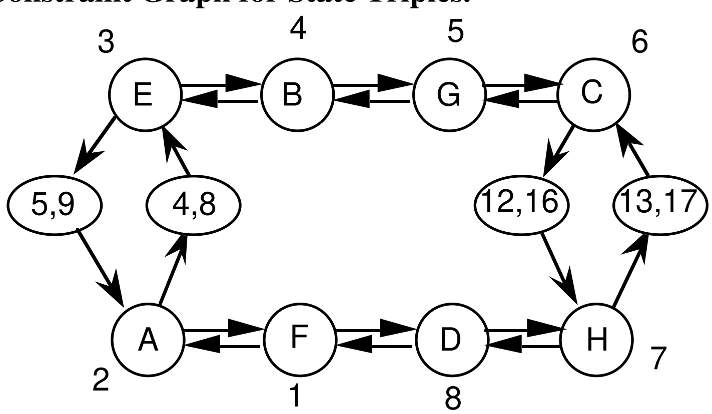

Figure A-3 Connecting the Disjoint Graph.

Appendix B Details of the Two-State Two-Port Latch

The graphs of the state triples for the Two-Port latch are shown in Fig. B-1. The flow table is shown in Table B-1. Fig. B-2 shows the relationship between the triples. A light arrow indicates that the distinguishing state of the first triple is adjacent to the setup state of the second triple. A dark arrow indicates that the distinguishing state of the first triple is the setup state of the second triple.

The lower bound on the length of the checking experiment is 19, since there are 16 total states, and states 2 and 7 appear twice as the only distinguishing states of triples. States 6 and 3 appear twice as starting states. Of course, an unstable state could be used instead as the starting state but that would still require an additional pattern. This raises the minimum length to 20. Looking at B and F, if B comes before F in the sequence, then state 5 must appear twice. If B comes after F in the sequence then state 4 must appear twice. The same argument can be applied to D and H. This raises the minimum length to 22. Now, looking at the graph in Fig. B-2, if B and F directly follow one another, and H and D directly follow one another, then it will not be possible to go through the whole graph without adding another pattern. But, if B and F do not directly follow each other, then both states 4 and 5 must appear twice. So in all cases, another pattern is needed, raising the minimum length to 23.

The connections between the nodes in Fig. B-2 form links between the distinguishing inputs. Only 3 links are needed to form a chain. The links are shown graphically in Fig. B-3. Any continuous traversal of this graph would form all links at least once. Therefore, the sequences generated are checking experiments. The order of one such sequence is marked on the graph of Fig. B-2. The actual checking experiment is shown in Table 3.5-2 in Section 3.5.

Table B-1 Flow Table for Two-State Two-Port Latch.

D_1D_2	00	01	11	10	00	01	11	10	00	01	11	10	00	01	11	10	
	$C_2 = 0$								$C_2 = 1$								
	$C_1 = 0$				$C_1 = 1$				$C_1 = 0$				$C_1 = 1$				Q
	2	4	6	8	10	12	11	13	14	15	17	16	-	-	-	-	0
	$\boxed{3}$	$\boxed{5}$	$\boxed{7}$	$\boxed{9}$	10	12	$\boxed{11}$	$\boxed{13}$	14	$\boxed{15}$	$\boxed{17}$	16	-	-	-	-	1

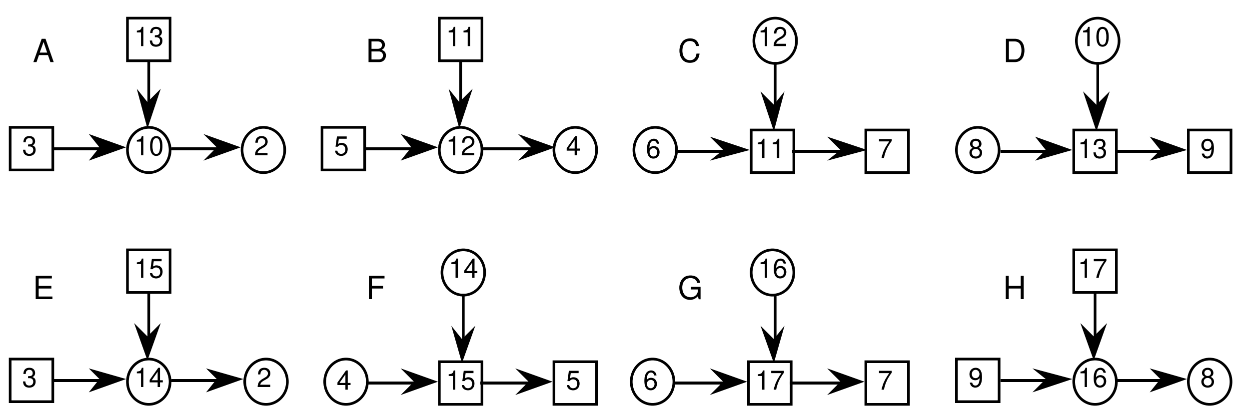

Figure B-1 Graphs of State Triples for Two-State Two-Port Latch.

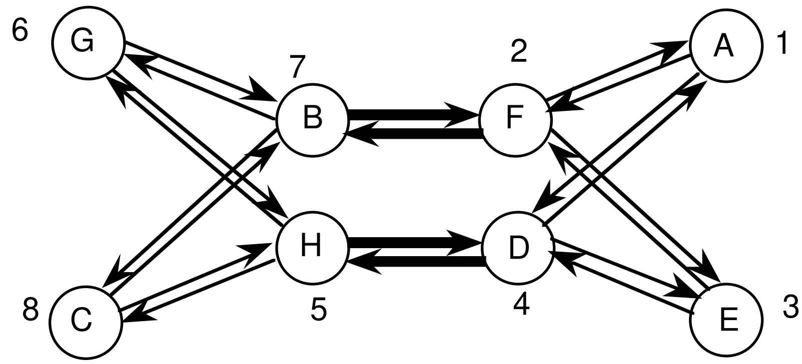

Figure B-2 Constraints on Order of Triples.

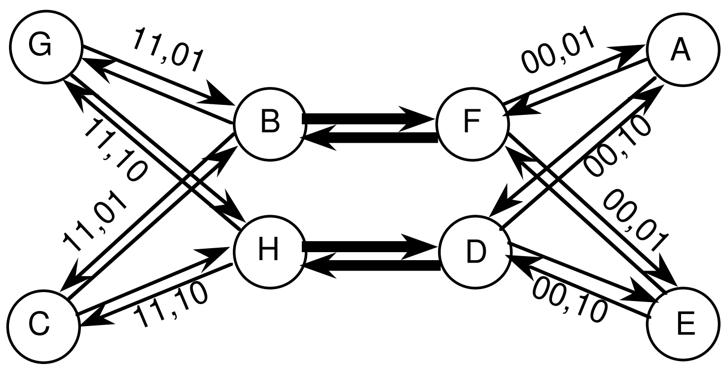

Figure B-3 Links in Triples.

Appendix C Details of the Two-State Load Enable Latch

The graphs of the state triples for the Load Enable latch are shown in Fig. C-1. The flow table is given in Table C-1. This latch is unique in that it has more distinguishing inputs than synchronizing inputs. The distinguishing states are shown graphically in Fig. C-2. In this graph, two states connected by an edge can follow each other in the sequence. The triples and the graphs of Fig. C-2 can be combined to produce minimum-length checking experiments. There are two ways to order the graphs to achieve a minimum-length checking experiment, ADBC and BCAD. Each of these orders contains two sequences. For example, either state 10 or state 8 could be used to go from Graph A to Graph D. If state 10 is selected then state 12 must be used to go from Graph D to Graph B. Similarly, there are two choices in going from Graph B to Graph C. This gives a total of four possible sequences for each graph order, and a total of eight minimum-length checking experiments. One of these checking experiments is shown in Table 3.6-2 in Section 3.6.

Since the states in Graph C are distinguishing states of all the distinguishing inputs, and since the sequences described go through all the states in Graph C before going into any other state, the sequences described above form a chain for all the distinguishing inputs. Hence, all the sequences form a chain among the distinguishing inputs.

Table C-1 Flow Table for Two-State Load Enable Latch.

LD								
00	01	11	10	00	01	11	10	
C = 0				C = 1				Q
2	4	6	8	10	12	15	14	0
$\boxed{3}$	$\boxed{5}$	$\boxed{7}$	$\boxed{9}$	$\boxed{11}$	$\boxed{13}$	$\boxed{15}$	14	1

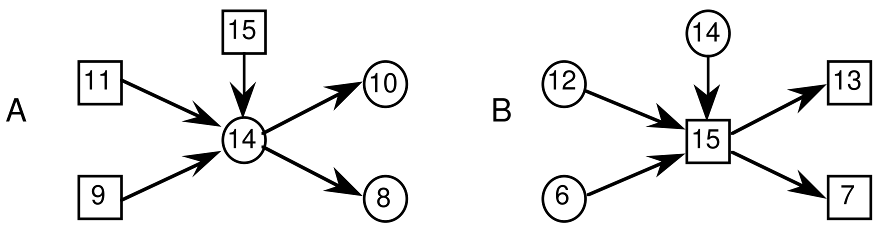

Figure C-1 Graphs of State Triples for Two-State Load Enable Latch.

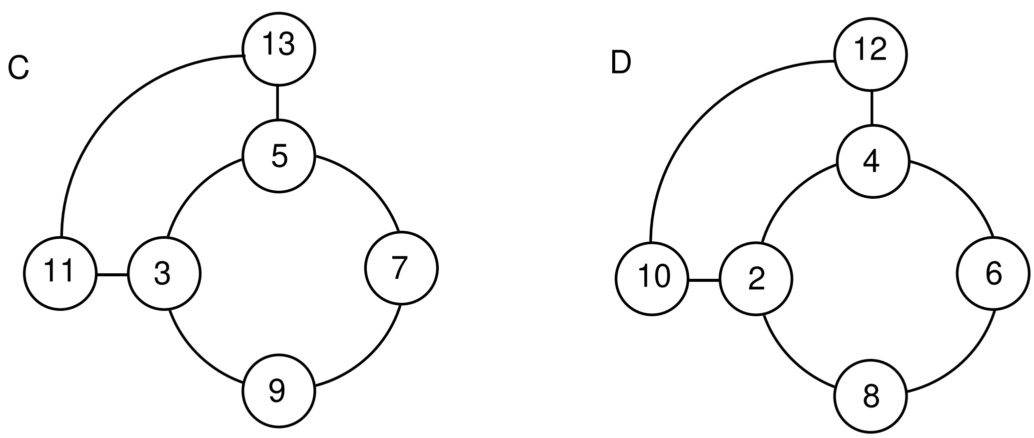

Figure C-2 Distinguishing States Adjacent States Can Follow Each Other in Sequence.

Appendix D Details of the Two-State D-Enable Latch

The graphs of the state triples for the D-Enable latch are shown in Fig. D-1. The flow table is given in Table D-1. The lower bound on the length of the checking experiment is 15, since there are 12 states, there are two more unstable states in one row than the other, and the latch is a single-input control state machine. The only way to state 3, the setup state of triple A, is to go through state 7 followed by either state 5 or state 9. The situation is shown in Fig. D-2. Suppose state 5 is picked. To enter state 9, either state 7 must be entered once again, or state 9 is followed by state 3, making state 3 be entered again. In either case one more pattern is needed raising the minimum length to 16.

Fig. D-3 shows the constraint between the state triples. One possible sequence of triples is BCDA. This sequence is shown in Fig. 3.7-2 of Section 3.7, and is of minimum length since it has 16 patterns.

Table D-1 Flow Table for D-Enable Latch.

DE	C = 0				C = 1				
	00	01	11	10	00	01	11	10	Q
	2	4	6	8	10	12	11	14	0
	3	5	7	9	10	12	11	14	1

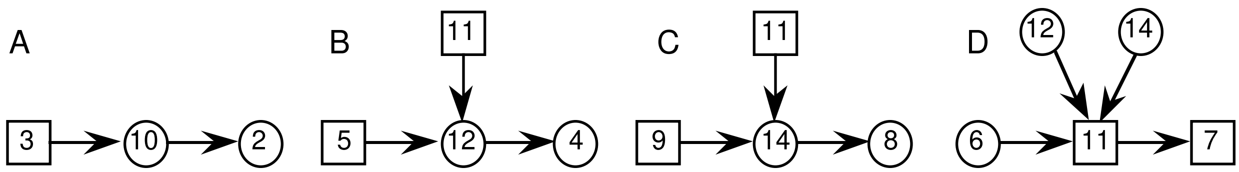

Figure D-1 Graphs of State Triples for D-Enable Latch.

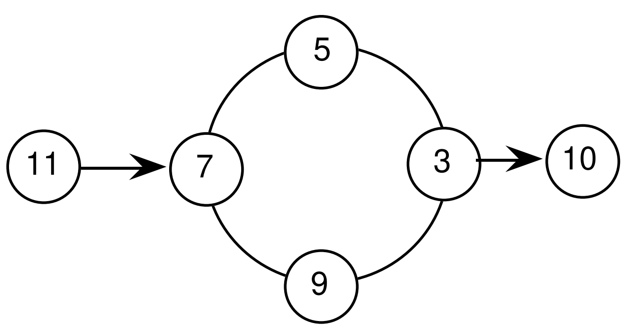

Figure D-2 Constraints on States.

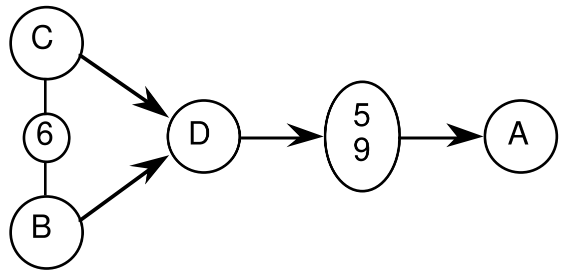

Figure D-3 Constraint Graph for State Triples.

Appendix E Details of the Two-State BILBO Latch

The graphs of the state triples for the BILBO latch are shown in Fig. E-1. The flow table is given in Table E-1. The lower bound on the length of the checking experiment is 55, since there are 48 total states and there are four more unstable states in the lower row of the flow table than in the upper one. Fig. E-2 show the relationship between the triples. The graph contains four disconnected subgraphs. The smallest subgraph contains only triple F. Fig. E-3 shows how the F subgraph can be joined to the other subgraphs. Since F connects to the middle of two other subgraphs, the subgraphs would need to be split. Each split would result in the addition of two patterns. To avoid splitting both subgraphs, which would result in 59 patterns, F should either be the first or last triple.

Table E-1 Flow Table for Two-State BILBO Latch.

DS (C = 0)

00	01	11	10	00	01	11	10	00	01	11	10	00	01	11	10	
$B_2 = 0$								$B_2 = 1$								
$B_1 = 0$				$B_1 = 1$				$B_1 = 0$				$B_1 = 1$				Q
2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	0
$\boxed{3}$	$\boxed{5}$	$\boxed{7}$	$\boxed{9}$	$\boxed{11}$	$\boxed{13}$	$\boxed{15}$	$\boxed{17}$	$\boxed{19}$	$\boxed{21}$	$\boxed{23}$	$\boxed{25}$	$\boxed{27}$	$\boxed{29}$	$\boxed{31}$	$\boxed{33}$	1

DS (C = 1)

00	01	11	10	00	01	11	10	00	01	11	10	00	01	11	10	
$B_2 = 0$								$B_2 = 1$								
$B_1 = 0$				$B_1 = 1$				$B_1 = 0$				$B_1 = 1$				Q
35	34	36	37	38	40	42	44	39	46	41	48	50	52	43	45	0
$\boxed{35}$	34	36	$\boxed{37}$	38	40	42	44	$\boxed{39}$	46	$\boxed{41}$	48	50	52	$\boxed{43}$	$\boxed{45}$	1

If F were the first triple, state 13 would need to be preceded by one of the following sequences: 35,3,5; 35,3,11; 43,31,15; 43,31,29. The first pattern in each of these sequences is accounted for in the bound by the initialization factor. The second two patterns in the sequence don't account for any of the other triples, so they will add to the minimum length. So the minimum length becomes 57. The distinguishing state of triple F is state 12. This state is not adjacent to the setup state of any triple. Therefore, an additional pattern is needed raising the minimum to 58. Now, suppose that triple F is in the last triple. Again 58 patterns are needed. Fig. E-4 shows the setup of one possible sequence. The actual sequence is shown in Table 3.9-2 in Section 3.9.

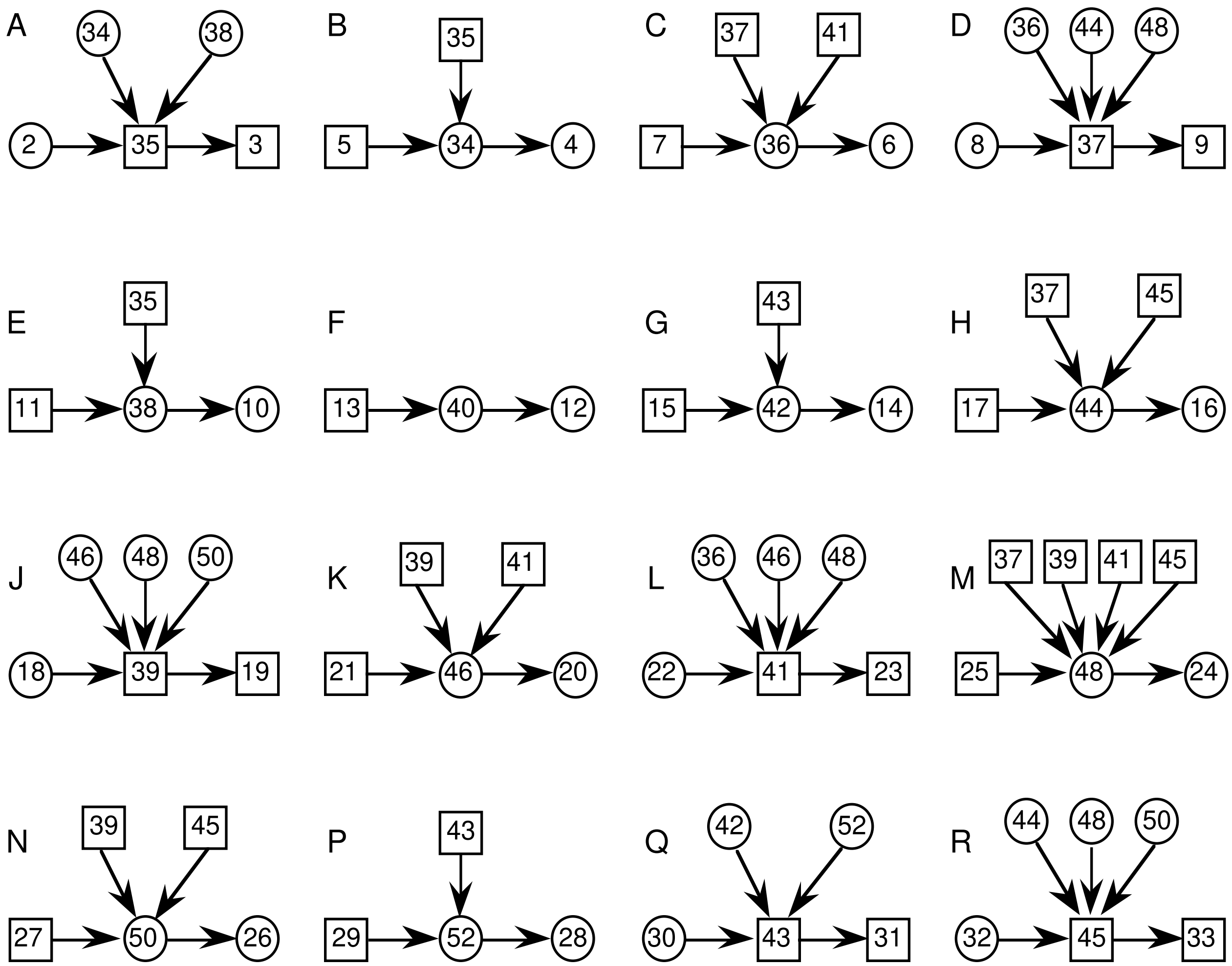

Figure E-1 Graphs of State Triples for Two-State BILBO Latch.

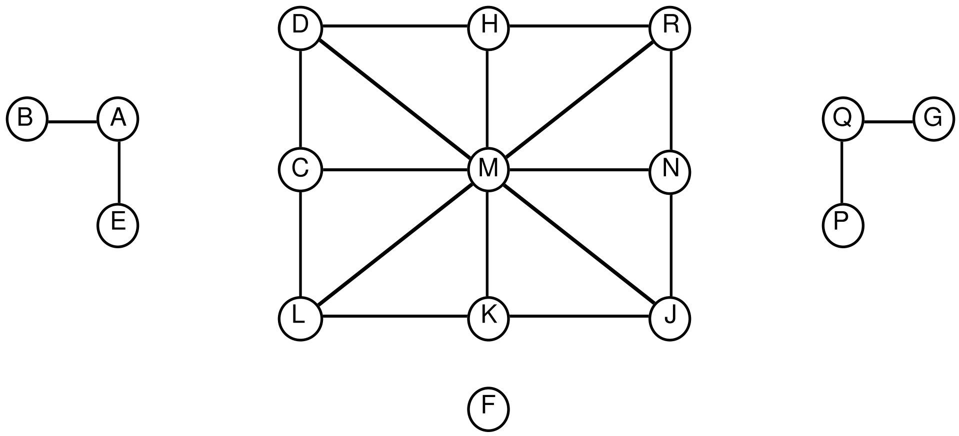

Figure E-2 Constraint Graph for State Triples.

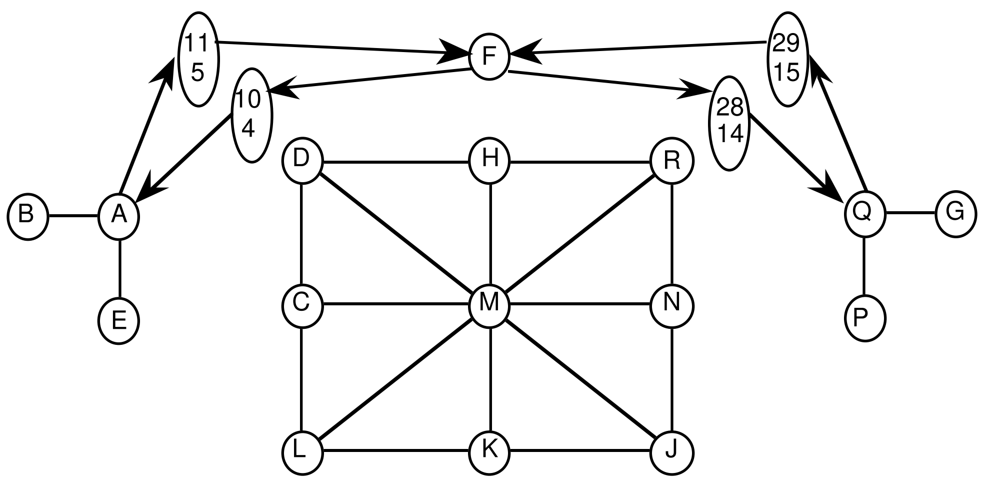

Figure E-3 Connection of Triple F to Other Triples.

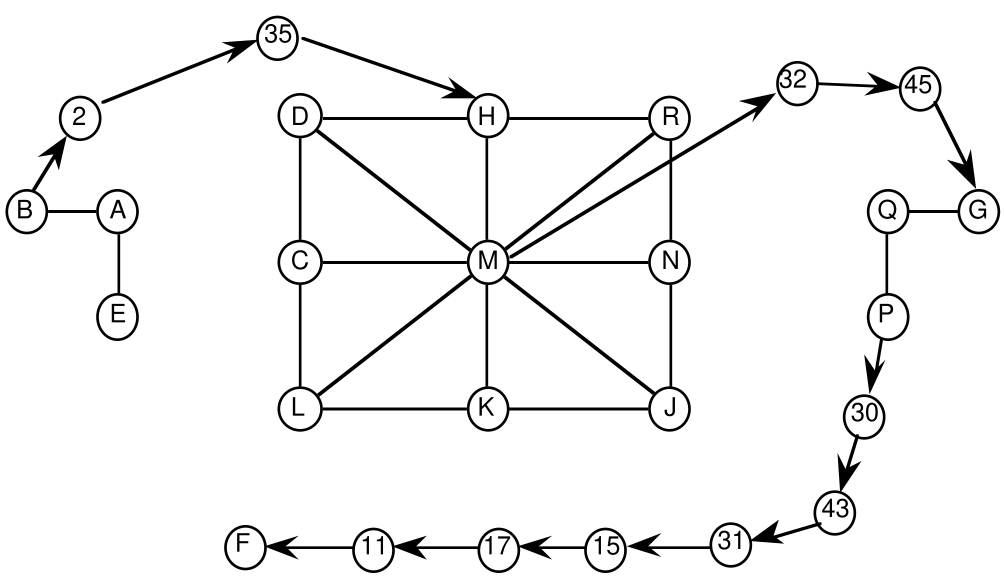

Figure E-4 Connecting the Disjoint Graph

Appendix F Details of the Two-State CBILBO Latch

The graphs of the state triples of the CBILBO latch are shown in Fig. F-1. The flow table is given in Table F-1. The lower bound on the length of the checking experiment is 25, since there are 24 states. Fig. F-2 shows the relationship between the triples. The graph is split into two disjoint sub-graphs. Fig. F-3 shows how the two sub-graphs can be connected with one additional state in the sequence. The minimum-length checking experiment can be achieved if the state used to connect the two disjoint graphs is a distinguishing setup state of the starting triple. This can be done if triple G (distinguishing setup state is 14), H (distinguishing setup state is 17), F (distinguishing setup state is 12) or E (distinguishing setup state is 11). The minimum-length checking experiment starting with triple G is shown in Table 3.10-2 of Section 3.10. From Theorem 2, a sequence derived from the graph in Fig. F-3 forms a chain among all the distinguishing inputs.

Table F-1 Flow Table for Two-State CBILBO Latch.

DS: 00	01	11	10	00	01	11	10	00	01	11	10	00	01	11	10	
C = 0								C = 1								
$B_1 = 0$				$B_1 = 1$				$B_1 = 0$				$B_1 = 1$				Q
2	4	6	8	10	12	14	16	18	19	20	21	22	23	25	24	0
3	5	7	9	11	13	15	17	18	19	20	21	22	23	25	24	1

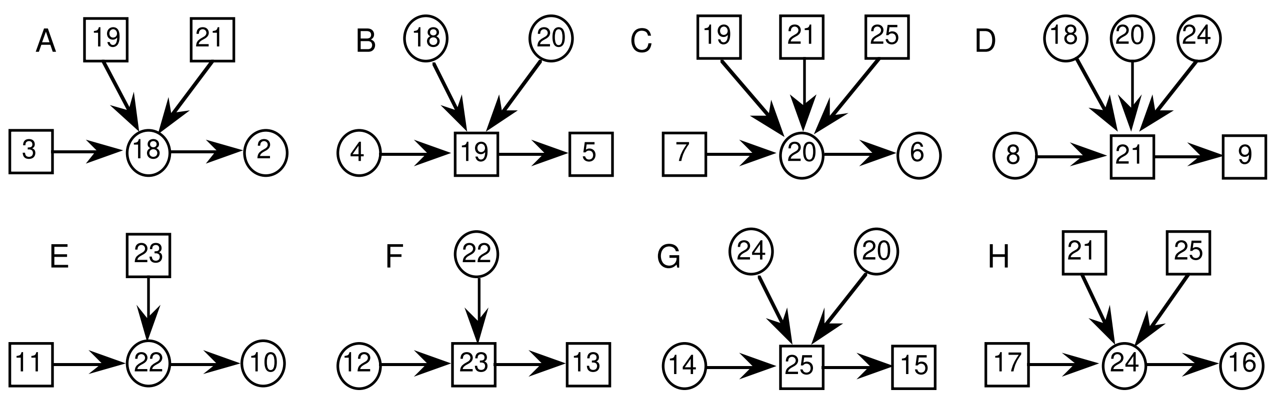

Figure F-1 Graphs of State Triples for Two-State CBILBO Latch.

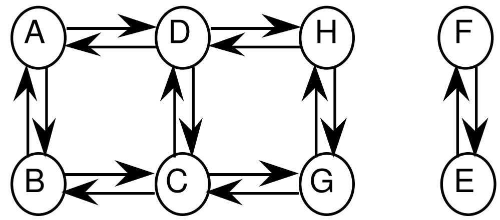

Figure F-2 Constraint Graph for State Triples.

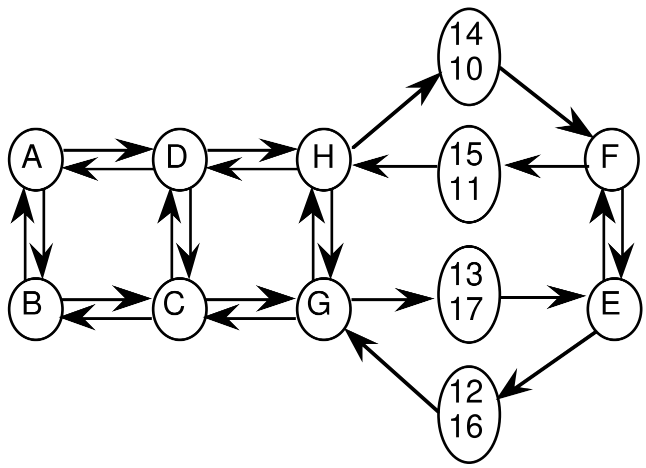

Figure F-3 Connecting the Disjoint Graph.