NAIST-IS-TR2006004
ISSN 0919-9527

Classification of Sequential Circuits Based on Acyclic Test Generation Complexity

Chia Yee Ooi and Hideo Fujiwara

July 2006

Graduate School of Information Science
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630-0192, Japan

Classification of Sequential Circuits Based on Acyclic Test Generation Complexity

Chia Yee OOI and Hideo FUJIWARA
Graduate School of Information Science
Nara Institute of Science and Technology
Kansai Science City 630-0192, Japan
{chiaye-o,fujiwara}@is.naist.jp

Abstract

This paper introduces a new class of sequential circuits called acyclically testable sequential circuits which is wider than the class of acyclic sequential circuits but whose test generation complexity is equivalent to that of the acyclic sequential circuits. Based on several circuit properties, the new class is further divided into three subclasses, namely acyclically testable sequential circuits class A, acyclically testable sequential circuits class B and acyclically testable sequential circuits class C. We also present a test generation procedure for acyclically testable sequential circuits and elaborate a design-for-test (DFT) method to augment an arbitrary sequential circuit into an acyclically testable sequential circuit. Since the class of acyclically testable sequential circuits is larger than the class of acyclic sequential circuits, the DFT method results in lower area overhead than partial scan method and still achieves complete fault efficiency. Besides, we show through experiment that the proposed method contributes to lower test application time compared to partial scan method. Moreover, the proposed method allows at-speed testing while the partial scan method does not.

1 Introduction

The test generation of acyclic sequential circuits has been shown to be τ^2-bounded [1,2] using time expansion model (TEM) [3]. In other words, the test generation complexity is at most the square of combinational test generation complexity, which is regarded as not difficult. This paper introduces a new class of sequential circuits with acyclic test generation complexity. The new class is called acyclically testable sequential circuits whose test generation complexity of the new class is bounded by a circuit property called thru function. [8] has introduced a class of circuits called partially strong testable circuits based on thru function but the target circuit is datapath only and test generation complexity was not discussed explicitly. [9] also considered existing thru functions in a scan technique but those thru functions are activated by primary inputs only. The new class that is defined in this paper covers some sequential circuits that are cyclic. In an acyclically testable sequential circuit, the signals that activate a thru function are either the signals at primary inputs or the signals at registers. Based on the properties of input dependency, thru tree dependency and the depth of thru trees, two subclasses of acyclically testable sequential circuits, namely acyclically testable sequential circuits type A and acyclically testable sequential circuits type B, are identified. By introducing stronger conditions, a subclass of acyclically testable sequential circuits type B is defined[4]. It is named acyclically testable sequential circuits type C. This paper also introduces a test generation procedure and an analysis of the test generation complexity of acyclically testable sequential circuits. This is followed by a design-for-test (DFT) method to augment an arbitrary sequential circuit into an acyclically testable sequential circuit. An experiment on benchmark circuits is conducted to show the effectiveness of the DFT method. Finally, the paper is concluded.

2 Acyclically Testable Sequential Circuits

This section defines a circuit representation called R-graph. Using R-graph, new concepts of circuit properties are introduced. These circuit properties include thru function, thru tree, thru tree dependency, input dependency and k-consistency. Based on these properties, the class of acyclically testable sequential circuits whose test generation is equivalent to the test generation of acyclic sequential circuits is defined. The relationship between the class of acyclically testable sequential circuits and acyclic sequential circuits are shown in Figure 1. Furthermore, three classes of sequential circuits are categorized as the subclasses of acyclically testable sequential circuits based on varying circuit properties.

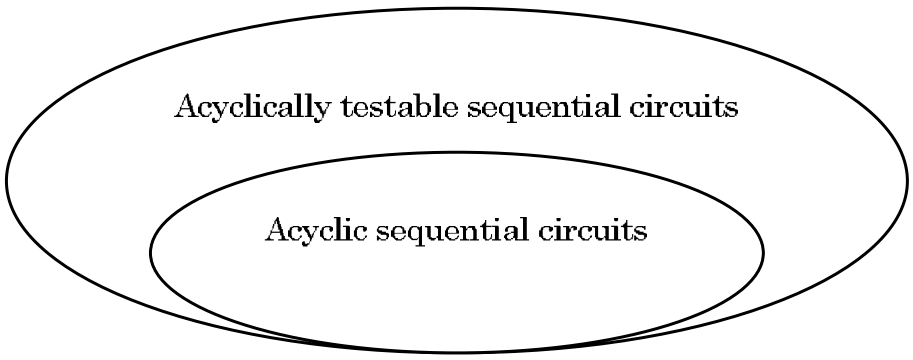

Acyclically testable sequential circuits
Acyclic sequential circuits

Figure 1: The relationship between acyclic sequential circuits and acyclically testable sequential circuits.

Definition 2.1. Let X, Y and Z be a set of boolean variables respectively in a circuit where $X \cap Y \cap Z = \emptyset$. A ***thru function*** $t_{X \to Y}$ is a boolean formula in conjunctive normal form such that

- the boolean connectives of the formula consist of $\wedge$ (AND), $\vee$ (OR) and $\neg$ (NOT);
- the boolean variables Z of the formula and X consist of register outputs and primary inputs while Y consists of register inputs and primary outputs;
- the signals at X transfer to Y if Z has an assignment that makes the thru function 'true' or active ($t_{X \to Y} = 1$);

Example 2.1. Figure 2(a) shows that without depending on the signals at the output X of feedback register r, Y can be justified by only U with thru function $t_{U\to Y}$ active. Figure 2(b) presents another example circuit with a multiplexer MUX. Signals at I transfer to L when $K = 0$.

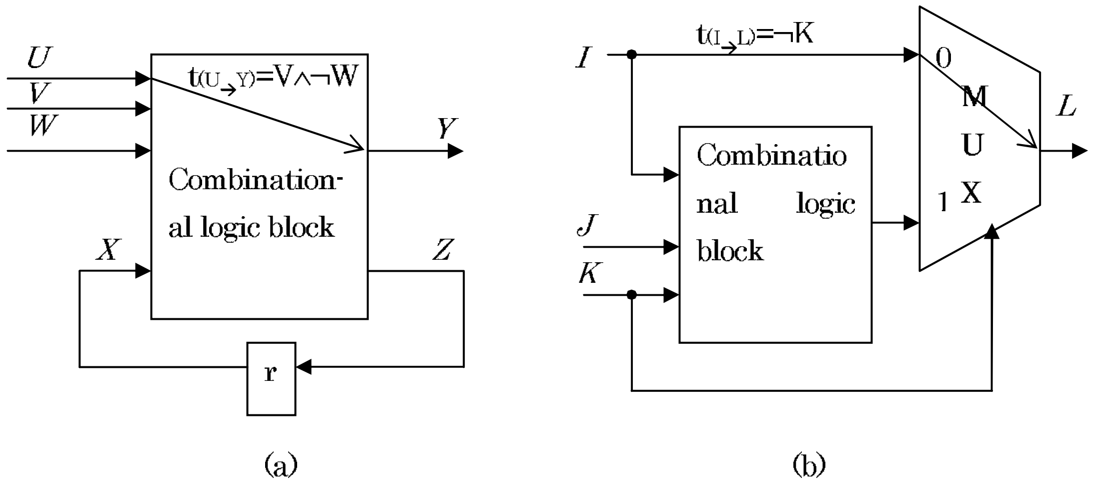


Figure 2: The use of thru functions

Definition 2.2. Two thru functions $t_{i\to j}$ and $t_{l\to m}$ are said to be ***dependent*** if they cannot be active at the same time.

Example 2.2. Figure 3 shows two functions $t_{I1\to O1}$ and $t_{I3\to O1}$ that are not dependent. In other words, thru functions t_1 can be active at the same time. Figure 4(a) shows two functions $t_{I1\to O1}$ and $t_{I3\to O1}$ that are dependent because signals at $I1$ and $I3$ do not transfer to $O1$ simultaneously. Figure 4(b) illustrates another example circuit that consists of three multiplexers. Thru function $t_{i\to o} = \neg p \wedge \neg q$ and thru funtion $t_{k\to o} = \neg p \wedge q$ are dependent as shown by the boolean formula in each thru function.

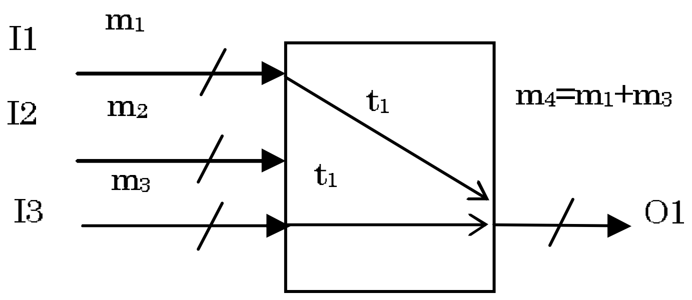


Figure 3: Not dependent thru functions.

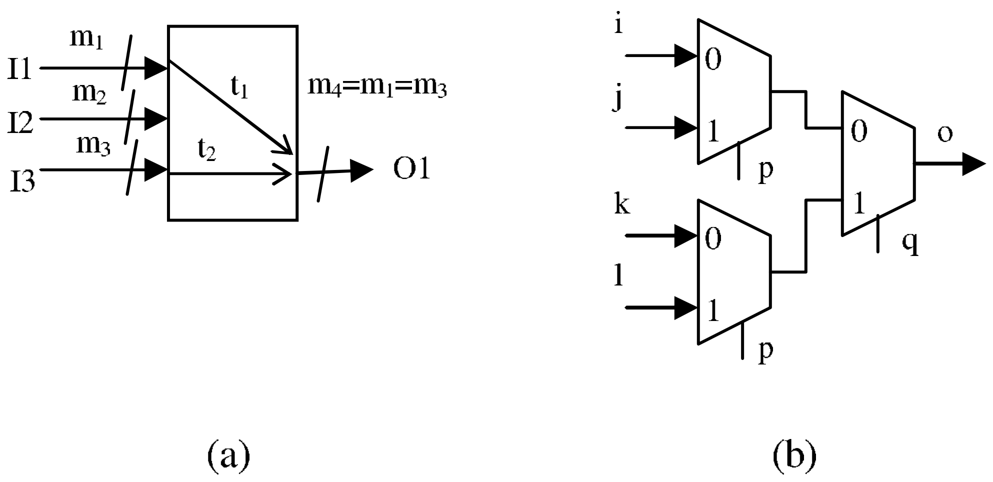


Figure 4: Dependent thru functions.

R-graph represents the topology of circuits by grouping flip-flops (FFs) into registers and including the information about the thru functions available in the logic. R-graph is used to introduce the new concepts of the circuit properties including thru function, thru tree, thru tree dependency and input dependency.

Definition 2.3. A circuit representation called ***R-graph*** is a directed graph $G = (V, A, w, r, t)$ that has the following properties.

1. Let FF_i denote a flip-flop. Let $pre(FF_i) = \{FF_j | FF_j \underset{\rightarrow}{c} FF_i\}$ (resp. $suc(FF_i) = \{FF_j | FF_i \underset{\rightarrow}{c} FF_j\}$) where c is a combinational path. $v \in V$ is a primary input or primary output or register that consists of a maximal set of flip-flops such that $pre(FF_p) = pre(FF_q)$ and $suc(FF_p) = suc(FF_q)$ for all FF_p, FF_q in the set of flip-flops;

2. $(v_i, v_j) \in A$ denotes an arc if there exists a combinational path from the register corresponding to v_i to the register corresponding to v_j;

3. $w : V \rightarrow Z^+$ (the set of positive integers) defines the number of flip-flops in each register corresponding to a vertex;

4. $r : V \rightarrow \{h, \emptyset\}$ defines type of a register where the register is a hold register v if $r(v) = h$. Else, it is a regular register. Note that $r(w) = \emptyset$ if w corresponds to a primary input or primary output;

5. $t : A \rightarrow T \bigcup \{\emptyset, 1\}$ (T is a set of thru functions) where $t(u, v) = \emptyset$ if there is no thru function for $(u, v) \in A$ and $t(u, v)$ is a thru function that transfer signals from the output of register u or primary input u to the input of register v or primary output v. If $t(u, v) = 1$ (also called identity thru function), the signal values are transferred from u to v through a wire logic (not a gate logic) directly. Note that identity thru function is always active.

Example 2.3. Figure 6 shows the R-graph of the sequential circuit S1 of Figure 5. The notation CLB in Figure 5 means combinational logic block that include the information of logic connection in the block. Black registers are registers with hold functions while others are regular registers. Register $R2$ is a hold register. The thru functions $t_{(I \to K)}$, $t_{(L \to N)}$, $t_{(O \to P)}$ and $t_{(Q \to S)}$ which are the thru functions extracted from the high level netlist of S1, are included in its R-graph. According to the R-graph, $R1$, $R2$ and $R3$ form a loop while $R5$ forms a self-loop.

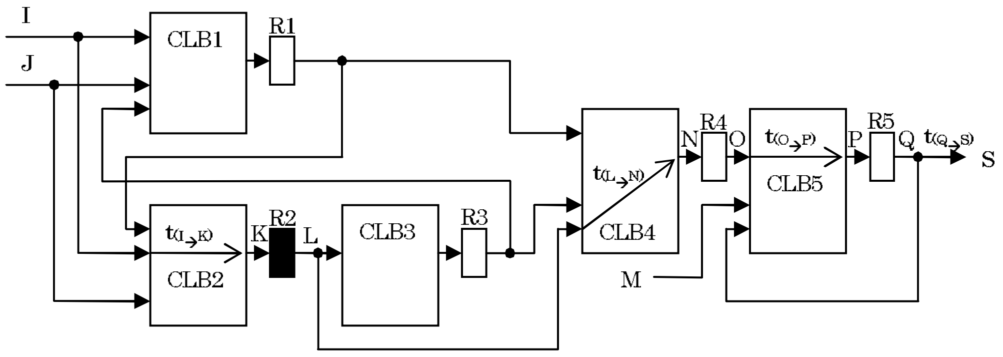


Figure 5: Sequential circuit $S1$.

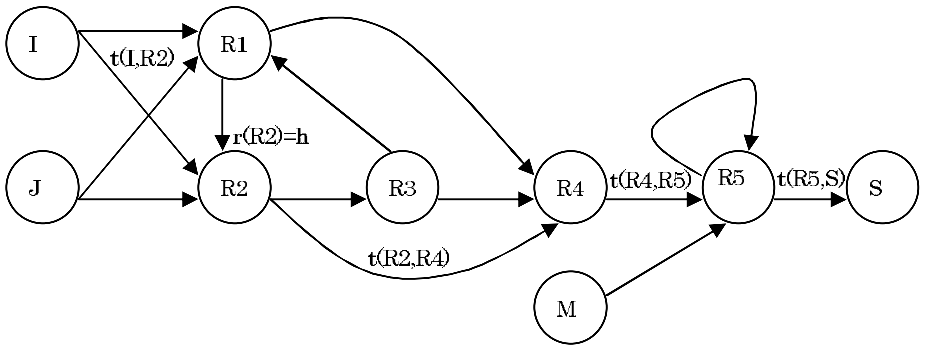


Figure 6: R-graph of $S1$.

If a thru function transfers signal from a register outside a loop or a primary input to a register inside the loop, the thru function can be used to justify the register in a loop with a signal from the register or the primary input outside the loop within one clock cycle but without depending on any signal in the loop. In other words, the loop is broken logically by the thru function. However, a thru function is not sufficient to guarantee that the register in the loop can be justified with any signal within the signal range of normal operation. The following shows an example where a thru function cannot justify a signal to a register in a loop.

Example 2.4. Let E (resp. G) denote a 4-bit variables consiting of bits e_3, e_2, e_1 and e_0(resp. g_3, g_2, g_1 and g_0). Figure 7 shows a sequential circuit that has two 4-bit adder

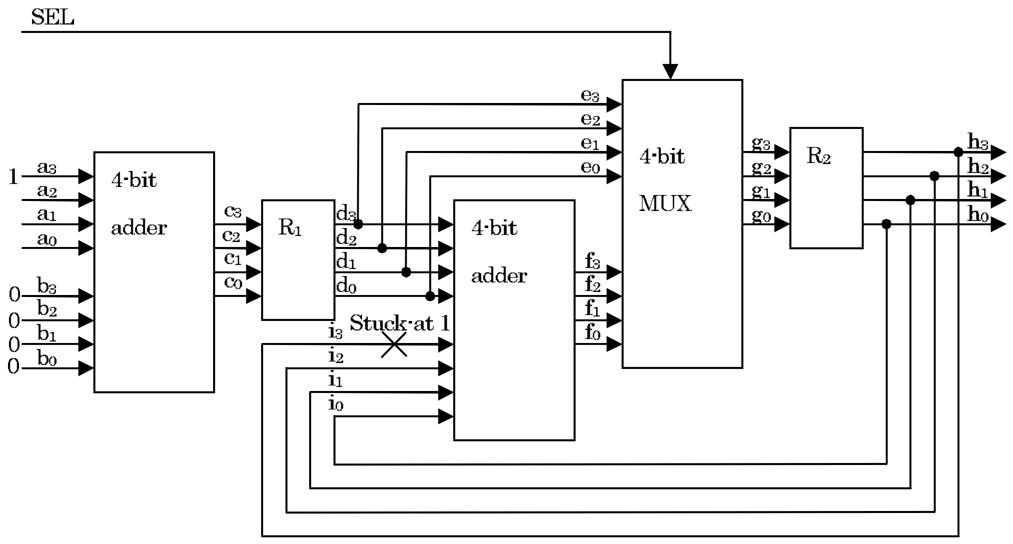


Figure 7: The limitation of thru function in justification.

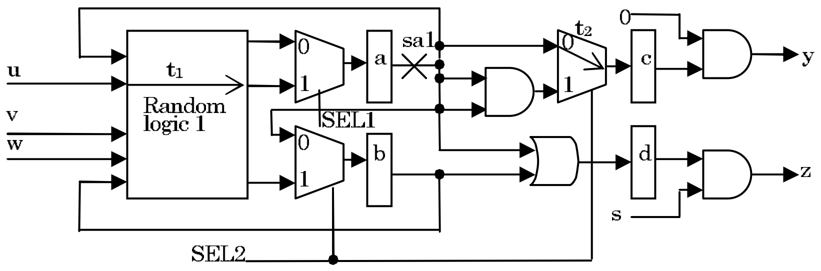


Figure 8: The limitation of thru function in propagation.

where $C = A + B$ and $F = D + I$ and a stuck-at-1 fault at i_3. There is a thru function $t_{E\rightarrow G} = \neg SEL$ that transfers signals from E to G when $SEL = 0$ within one clock cycle without depending on the signals at the output of feedback register R_2 at I. However, if the range of the signals that are justifiable by the thru function is studied carefully, it is obvious that the thru function cannot justify some signals that are in the range of the signals at G in normal operation. In this example, the signal range at E in normal operation is from 8 to 15. If the thru function is used to justify G, the justifiable signal range at G is from 8 to 15. Thus, the stuck-at-1 cannot be activated. However, if the following steps are done, the stuck-at-1 can be activated after four clock cycles.

- First, D is assigned 8 and then tranfered to G through the thru function with $SEL = 0$;
- At the next clock cycle, D is assigned 9 with $SEL = 1$. Note that the signal G is now 1;
- After one clock pulse at R_2, I is justified with 1 and the stuck-at fault is activated.

Thru function does not guarantee the justification of every signal within the range of normal operation. Similarly, thru function does not guarantee the propagation of every signal within the range of normal operation.

Example 2.5. Figure 8 shows a circuit where the stuck-at-1 fault (sa1) can be activated by assigning signal 0 from primary input u through thru function t_1 with $SEL1 = 1$. When the fault effect propagates through thru function t_2 to the output of flip-flop c after one clock cycle, the fault effect disappears as it is masked by the constraint 0 at the input of AND gate y. However, if the fault effect first propagates to the output of flip-flop b, the fault effect can be observed at z without depending on any thru function.

Therefore, a concept called thru tree is introduced in this paper. Thru tree consists of a set of thru functions connected in a form of rooted tree that starts from primary inputs and ends at a primary output.

Definition 2.4. Let R-graph $G_R = (V, A, w, r, t)$ represent a given sequential circuit S. A ***thru tree*** is a subgraph of the R-graph such that

1. it is a rooted tree;
2. there is only one sink (root), which is corresponding to a primary output;
3. the sources are vertices that correspond to primary inputs;
4. each arc is labeled with a thru function.

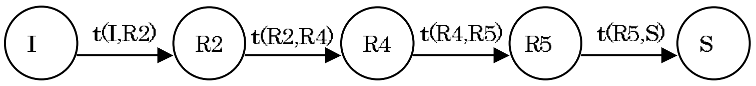
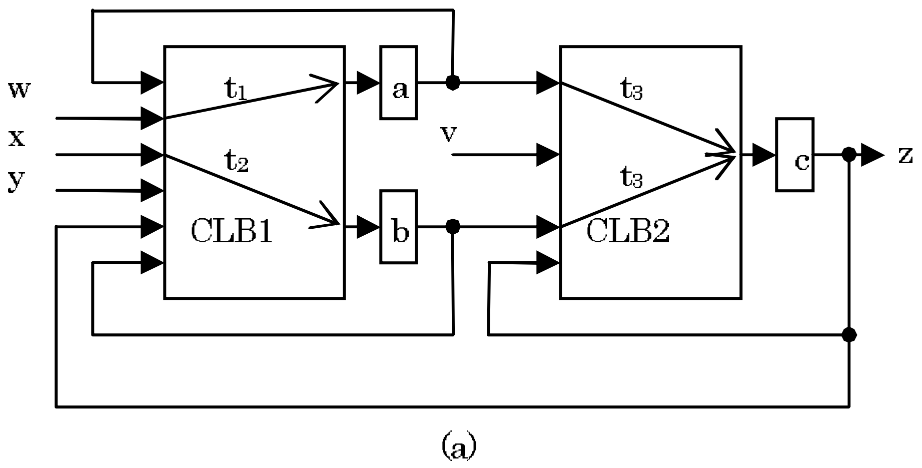


Figure 9: The only thru tree of S1.

Figure 9 shows the only thru tree of S1, which is also a path whose arcs are labelled with thru functions. In the thru tree, each register is justifiable from a primary input and is observable at a primary output through a series of thru functions if each thru function in the tree is activated by a signal whose corresponding vertex is not in the thru tree. If a thru function in a thru tree is activated by a signal of the vertex which is also in the same thru tree, the thru function does not guarantee the justification and propagation. The following shows an example where a contradiction takes place.

Example 2.6. Figure 10 shows another example circuit S2 and its R-graph. Figure 11 shows a thru tree of S2. Let $t_1 = y$ and $t_2 = c$ and $t_3 = \neg v$. In order to justify register c using the thru tree, all the thru functions t_1, t_2 and t_3 must be active. However, t_2 depends on c to become active while c is depending on thru function t_2 for justification from primary inputs x and w without depending on feedback a and c. The interdependency between justifying register c and activating thru function t_2 occurs. Therefore, justification of register c is not guaranteed.



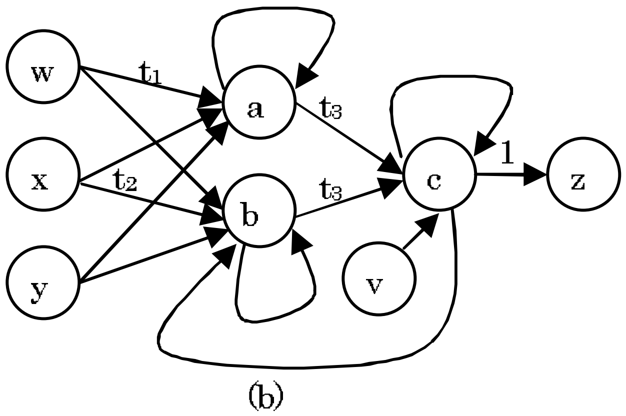


Figure 10: S2 (a) and its R-graph (b).

To avoid the interdependency, it is essential to define the concept of the dependency between a thru tree and the register that activates a thru function in the thru tree as well as the dependency between two thru trees. Then, the concept is based in the definition of acyclically testable sequential circuits.

Definition 2.5. If V_{ti} is a set of vertices that activate a thru function t_i in a thru tree T_j, T_j is said to be ***dependent*** on V_{ti}. Furthermore, if V_{ti} includes a vertex in a thru tree T_k, T_j is said to be ***dependent*** on T_k.

Example 2.7. Figure 12 are a sequential circuit S3, its R-graph and its thru trees $T1$ and $T2$. From its R-graph and thru trees, $T1$ is dependent on $I3$ while $T2$ is dependent on $R2$ and $I1$. Furthermore, $T2$ is dependent on $T1$ since $R2$ and $I1$ are included in $T2$.

Another issue to be discussed here is input dependency. While a thru function $t_{x \to y}$ is being used to justify y, x is fixed at the signals that are needed to justify y. If x is needed to justify y and another signal, for example z, at the same time, z may not be justified since x is fixed to justify y through the thru function. Again, justification is not guaranteed.

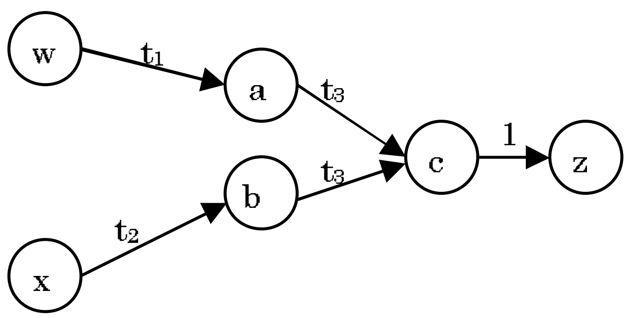


Figure 11: A thru tree of S2.

Definition 2.6. Let G_R be the R-graph of a sequential circuit S, and let B be a set of thru trees in G_R. Let (u, v) be a set of all paths starting at u and ending at v. Two distinct paths $p_1, p_2 \in (u, v)$ have ***input dependency*** if the following conditions are satisfied.

i. the first arc of one of the paths is different from the first arc of another path;

ii. the first arc of at least one of the paths is labeled with a thru function in a thru tree in B;

iii. each path contains at most one cycle that starts from and ends at v;

iv. if the first arc of a path p_1 (resp. p_2) does not have a thru function in a thru tree in B, all vertices except the first vertex and the last vertex are not included in any thru tree in B. Else if the first arc of a path p_1 (resp. p_2) has a thru function in a thru tree in B, all vertices except the first vertex, the second vertex and the last vertex are not included in any thru tree in B;

v. p_1 and p_2 have same length;

Example 2.8. Figure 13 shows an example circuit $S4$ with thru functions t_0, t_1, t_2 and t_3 and its R-graph. Figure 14 shows two of the paths from R-graph where the first arc of path $x \to v \to v$ is labeled with t_1 and both paths are of same length. Suppose v and w have to be justified 1 respectively in order to excite a fault in CLB2. To justify v, x has to be assigned 1 one clock cycle before. However, if x needs to be 0 in order to generate 1 at w using the whole logic in CLB1, a conflict takes place. Thus, the fault cannot be excited using thru functions t_0 and t_1 for justification of v.

Input dependency can be resolved by hold registers with certain conditions. The following shows one example how a hold register resolves an input dependency.

Example 2.9. Circuit $S5$ in Figure 15 has an input dependency between two paths, $R1 \to R4 \to PO$ and $R1 \to R5 \to PO$. The two paths with input dependency is shown in the aspects of graph and time expansion model in Figure 16. Circuit $S6$ in Figure 17 is same as $S5$ except register $R5$ of $S6$ is a hold register. By holding $R5$ at time $T2$, input dependency between paths $R1 \to R4 \to PO$ and $R1 \to R5 \to PO$ can be resolved. This is illustrated in Figure 18.

Thru tree dependency has to be resolved in test generation process. Thru tree dependency takes place when the signal of a register is used to activate a thru function and justify another register simultaneously.

Definition 2.7. Let G_R be the R-graph of a sequential circuit S, and let B be a set of thru trees in G_R and let $a_{ti} \in G_R$ be an arc with thru function t_i. For each pair of paths p_m and p_n, p_m and p_n have ***thru tree dependency*** if

i. p_m is a path that starts from the sink vertex u of arc a_{ti} and ends at a vertex v;

ii. p_n is a path that starts from a vertex w in V_{ti} and ends at v;

iii. p_m and p_n are the paths where each path is either a simple path or a path that contains a cycle starting from and ending at v, and each vertex, except the first vertex and the last vertex, is not included in any thru tree in B;

iv. $|p_m| < |p_n|$.

Example 2.10. Figure 19 is a sequential circuit with hold registers u, w, x and z. Note that signal at register z at time 4 is used to justify $CLB1$ and at the same time activate thru function t_0 at time 4 in Figure 20. This is because there is a thru tree dependency between paths $x \to x$ and $z \to u \to x$.

Thru tree dependency is also resolvable by hold registers.

The number of thru trees that depend on each other in an acyclically testable sequential circuit is one of the factors that bound its test generation complexity to τ^2-bounded. Therefore, a dependency graph is introduced to represent the property of the dependency and the number of thru trees that depend on each other.

Definition 2.8. Let G_R be the R-graph of a sequential circuit S, and let B be a set of thru trees in G_R. The ***dependency graph*** of B is a directed graph $G_D = (V_D, A_D)$ such that

i. vertex $v \in V_D$ is a thru tree in B;

ii. $(v_i, v_j) \in A_D$ denotes an arc if there exists a vertex (of G_R) in thru tree v_i that activates a thru function in thru tree v_j;

Example 2.11. Figure 21 shows the dependency graph of $T1$ and $T2$ of S3.

Based on the concepts of thru function, thru tree, thru tree dependency and input dependency, three classes of acyclically testable sequential circuits are detailed in the following subsections.

2.1 Acyclically Testable Sequential Circuits Type A

Acyclically testable sequential circuits type A is a class of acyclically testable sequential circuits that does not have input dependency as well as thru tree dependency.

Definition 2.9. Let R-graph $G_R = (V, A, w, r, t)$ represent a given sequential circuit S. S is called to be ***acyclically testable*** if G_R contains a set of disjoint thru trees such that the following conditions are satisfied.

1. The thru trees cover all the vertices of a feedback vertex set;
2. Let V_{ti} be a set of all vertices that activate a thru function t_i. For any thru function t_i in each thru tree T_j, the following conditions are satisfied.
   i. V_{ti} does not include any vertex of T_j;
   ii. V_{ti} does not include any register vertex that is not included in any thru tree. Note that V_{ti} can include an input vertex or output vertex that is not included in any thru tree;
   iii. All the register vertices in T_j and V_{ti} are hold registers;
3. Let T_i and T_j be two different trees.
   i. For each pair of thru function t_i in T_i and t_j in T_j, V_{ti} and V_{tj} are disjoint;
   ii. If T_i (resp. T_j) is dependent on T_j (resp. T_i), T_j (resp. T_i) is not dependent on T_i (resp. T_j), and
   iii. If T_i is dependent on T_j, $|p_i| \geq |p_j|$ for each pair of p_i and p_j such that p_i (resp. p_j) is a path starting from vertex u_i in T_i (resp. vertex u_j in T_j) and ending at vertex v with at most one cycle starting from and ending at v, where $|p_i|$ (resp. $|p_j|$) denotes the length of path p_i (resp. p_j);
4. For each pair of reconvergent paths p_1 and p_2, p_1 and p_2 does not have input dependency;
5. For each pair of paths p_1 and p_2, p_1 and p_2 does not have thru tree dependency.

2.2 Acyclically Testable Sequential Circuits Type B

Different from acyclically testable sequential circuits type A, acyclically testable sequential circuits type B are allowed to have input dependency and thru three dependency with conditions that there exists a set of hold registers that can resolve the input dependency and thru tree dependency. As a tradeoff between the input dependency (resp. thru tree dependency) and test generation complexity, the multiplication of the maximum length of paths in the dependency graph and the maximum depth of thru trees in the sequential circuit is to be bounded by a constant.

Definition 2.10. Let R-graph $G_R = (V, A, w, r, t)$ represent a given sequential circuit S. S is called to be ***acyclically testable*** if G_R contains a set of disjoint thru trees B such that the following conditions are satisfied.

1. B is a set of thru trees that satisfies the following conditions.
   i. The thru trees in B cover all the vertices of a feedback vertex set; and
   ii Let the maximum depth of thru trees in B be D_{max}. Let the maximum length of paths in the dependency graph of B be L_{max}. $D_{max} \times L_{max}$ is bounded by k;
2. Let V_{ti} be a set of all vertices that activate a thru function t_i. Let $a_{ti} \in A$ be an arc with thru function t_i. For any thru function t_i in each thru tree T_j in B, the following conditions are satisfied.

i. V_{ti} does not include any vertex of T_j;

ii. V_{ti} does not include any register vertex that is not included in any thru tree in B. Note that V_{ti} can include an input vertex or output vertex that is not included in any thru tree in B;

iii. The sink vertex of arc a_{ti} in T_j and the register vertices in V_{ti} are corresponding to hold registers;

iv. For each pair of paths p_m and p_n, if p_m and p_n have thru tree dependency then there exists a hold register vertex x ($r(x) = h$) that satisfies either Condition A or Condition B.

(A) i. x is on p_m but not p_n, and $x \neq u$; and

ii. Let p_k be a path that starts from x and ends at v. Let p_p be the subpath of p_m that starts from x and ends at v. $|p_p| \geq |p_k|$ for all p_k.

(B) i. x is on p_n but not p_m, and $x \neq w$; and

ii. Let p_k be a path that starts from x and ends at v. Let p_p denote the subpath of p_n that starts from x and ends at v. $|p_p| \geq |p_k|$ for all p_k.

$|p_m|$ (resp. $|p_n|$, $|p_k|$) denotes the length of path p_m (resp. p_n, p_k);

3. Let T_i and T_j be two different trees in B.

i. For each pair of thru function t_i in T_i and t_j in T_j, V_{ti} and V_{tj} are disjoint;

ii. If T_i (resp. T_j) is dependent on T_j (resp. T_i), T_j (resp. T_i) is not dependent on T_i (resp. T_j), and

4. For each pair of reconvergent paths p_1 and p_2 that start from u and end at v, there exists a hold register vertex w on p_1 such that the length of the subpath of p_1 that starts from w and ends at v is equal or longer than the length of p_k for all p_k if p_1 and p_2 have input dependency where p_k denotes a path that starts from w and ends at v.

2.3 Acyclically Testable Sequential Circuits Type C

In [4], a subclass of acyclically testable sequential circuits type B is introduced. This class is called acyclically testable sequential circuits type C, which has stronger conditions. A property called k-consistency is introduced. Then, the class of acyclically testable sequential circuits type C is defined based on k-consistency.

Definition 2.11. Let R-graph $G_R = (V, A, w, r, t)$ represent a given sequential circuit S. A set of thru tree B in G_R is said to be ***k-consistent*** with G_R if the following conditions are satisfied.

i. The dependency graph of B is acyclic;

ii. All thru trees in B are disjoint;

iii. Let the maximum depth of thru trees in B be D_{max}. Let the maximum length of paths in the dependency graph of B be L_{max}. $D_{max} \times L_{max}$ is bounded by k;

iv. Any vertex that activates a thru tree T_i in B is either an input vertex or a hold register vertex in B, and activates no other thru tree T_j in B;

v. For each pair of reconvergent paths p_1 and p_2 that start from u and end at v, there exists a hold register vertex w on p_1 but not on p_2 such that w is not the second vertex x of p_1 and the length of the subpath $w \rightarrow v$ of p_1 is equal to or longer than the length of any other path p_k that starts from w and ends at v if all vertices on p_1 and p_2 except u, v and x are not included in any thru tree in B and either of the following Conditions a and b is satisfied.

   a. p_1 and p_2 are of equal length and the first arc (u, x) on p_1 is labeled with a thru function of a thru tree in B; or

   b. p_1 is equal to or shorter than p_2 and the first arc (u, x) on p_1 activates the thru function coming to the vertex x.

Definition 2.12. A sequential circuit S is said to be ***k-acyclically testable*** if the R-graph G_R of S contains a set of thru trees B that is k-consistent with G_R and covers all the vertices of a feedback vertex set of G_R. A sequential circuit S is said to be ***acyclically testable*** if S is k-acyclically testable for some constant k.

Example 2.12. S3 is an acyclically testable sequential circuit. Its ATEG will be showed in the following subsection.

Since an acyclic sequential circuit is an acyclically testable sequential circuit with empty feedback vertex set according to definitions of acyclically testable sequential circuits, a sequential circuit is acyclically testable if it is acyclic but the converse is not correct. Therefore, the following theorem is concluded.

Theorem 2.13. *The class of acyclically testable sequential circuits is a proper superset of the class of acyclic sequential circuits. (Figure 22)*

3 Time Expansion Model

Time expansion model (TEM) has been introduced in [5] as a test generation model for acyclic sequential circuits based on time expansion graph (TEG). A topology graph is a directed graph of circuit representation where a vertex v denotes a combinational logic block while an arc (u, v) represents a connection from combinational logic block u to combinational logic block v. The authors defined time expansion graph (TEG) for the topology graph of a given acyclic sequential circuit. To facilitate the discussion of test generation model for acyclically testable sequential circuits, the time expansion graph (TEG) that is used to derive a time expansion model for a given acyclic sequential circuit represented by R-graph is redefined.

Definition 3.1. Let S be an acyclic sequential circuit and let $G_R = (V, A, w, h, t)$ be the R-graph of S. Let $G_T = (V_E, A_E, T, l)$ be a directed graph, where V_E is a set of vertices, A_E is a set of arcs, T is a mapping from V_E to a set of integer and l is a mapping from V_E to the set of vertices in R. If graph G_T satisfies the following five conditions, graph G_T is said to be a ***time-expansion graph (TEG)*** of G_R.

C1 (Input/Output and register preservation): The mapping l is a surjective, i.e., $\forall v \in V, \exists u \in V_E, s.t. v = l(u)$.

C2 (Logic preservation) Let u be a vertex in G_T. For any direct predecessor $v (\in pre(l(u)))$ of $l(u)$ in G_R where $v \neq l(u)$, there exists a vertex u' in G_T such that $l(u') = v$ and $u' \in pre(u)$. Here, $pre(v)$ denotes the set of direct predecessors of v.

C3 (Time consistency) For any arc (u, v) $(\in A_E)$, there exists an arc $(l(u), l(v))$ such that $T(v) - T(u) = 1$ if $l(u)$ corresponds to a register or a primary input and $l(v)$ corresponds to a register. $T(v) - T(u) = 0$ if $l(u)$ corresponds to a register and $l(v)$ corresponds to a primary output.

C4 (Time uniqueness) For any pair of vertices u, v $(\in V_E)$, if $T(u) = T(v)$ and if $l(u) = l(v)$, then the vertices u and v are identical, i.e., $u = v$.

C5 (Hold consistency): For any arc (u, v) in G_T, if $T(v) - T(u) = 1$ and $l(v) = l(u) = w$, w is a hold register $(r(w) = h)$ that is in hold mode at $T(u)$ and the number of predecessors of v is one.

Definition 3.2. Let S be an acyclic sequential circuit, let $G_R = (V, A, w, h, t)$ be the R-graph of S, and let $G_T = (V_E, A_E, T, l)$ be a TEG of G_R. The combinational equivalent $C_E(S)$ obtained by the following procedure is said to be the ***time expansion model (TEM)*** of S based on G_T.

1. For each time frame, replace each vertex with a connection without a register and replace each arc with the combinational logic block where the corresponding combinational path (represented by the arc) is located. Each combinational logic block appears at most once at each time frame.

2. A logic gate in each logic block is removed if it is not reachable to any input of other logic blocks.

Example 3.1. Figure 23(b) shows the R-graph of one of the acyclic sequential circuit S8 in Figure 23(a). Its time expansion graph (TEG) and its time expansion model (TEM) are derived in Figure 23(c) and 23(d).

4 Acyclically-Extended Time Expansion Model

This section introduces a test generation model called acyclically-extended time expansion model (ATEM) to perform the test generation on acyclically testable sequential circuits. The procedure of test generation is also described. In the following text, the vertex that corresponds to a primary input (resp. primary output) is called input vertex (resp. output vertex) while the vertex that corresponds to a register (resp. flip-flop) is called register vertex (resp. flip-flop vertex). First, acyclically-extended time expansion graph (ATEG) is defined. Some properties of ATEG are introduced. Based on ATEG and the properties, ATEM is redefined.

Definition 4.1. Let S be an acyclically testable sequential circuit with acyclic test thru trees B and let $G_R = (V, A, w, h, t)$ be the R-graph of S. The ***acyclically-extended time expansion graph (ATEG)*** $G_A = (V_A, A_A, T, l)$ with respect to B is a directed graph that satisfies the following conditions.

C1 (Input/Output and register preservation): The mapping l is a surjective, i.e., $\forall v \in V$, $\exists u \in V_A$, s.t. $v = l(u)$.

C2 (Logic preservation for fault excitation phase): Let u be a vertex in G_R. For any direct predecessor $v (\in pre(u))$ of u in G_R, there exists vertices w and x in G_A such that $l(w) = u$, $l(x) = v$, $x \in pre(w)$ and $|pre(w)| = |pre(u)|$. Here, $pre(w)$ (resp. $pre(u)$) denotes the set of direct predecessors of w (resp. u) and $|pre(w)|$ (resp. $|pre(u)|$) denotes the number of all direct predecessors of w (resp. u).

C3 (Thru tree for justification and propagation): Let u be a vertex in a thru tree T_i in B in G_R. Let $W \subset pre(u)$ be a set of all direct predecessors of u in T_i. Let t_j be a thru function on all incoming arcs of u in T_i and V_{tj} be a set of vertices that activate t_j. For each u in T_i in B in G_R, there exists a vertex v in G_A which satisfies the following conditions.

i $l(v) = u$;

ii For each vertex x in $pre(v)$, the following conditions are satisfied.

a. If there exists a vertex w' in W such that $l(x) = w'$ then $x \notin pre(z)$ for any z where $l(z)$ is a vertex included in any other thru tree T_k except T_i and $x \notin pre(y)$ such that $l(y) = l(x)$;

b. Let T_k be a thru tree that is activated by $l(x)$. If $l(x) = l(v)$, then $|pre(v)| = 1$ and $x \notin pre(z)$ for any z where $l(z) \neq l(v)$ and $l(z)$ is a vertex that is not included in thru tree T_k;

c. If $l(x) \in V_{tj}$, then $x \notin pre(z)$ for any z where $l(z) \neq l(x)$ and $l(z)$ is a vertex that is not included in thru tree T_i.

$|pre(v)|$ is the number of vertices in $pre(v)$.

C4 (Time consistency): For any arc (u, v) $(\in A_A)$, there exists an arc $(l(u), l(v))$ such that $T(v) - T(u) = 1$ if $l(u)$ corresponds to a register or a primary input and $l(v)$ corresponds to a register. $T(v) - T(u) = 0$ if $l(u)$ corresponds to a register and $l(v)$ corresponds to a primary output.

C5 (Time uniqueness): For any pair of vertices u, v $(\in V_A)$, if $T(u) = T(v)$ and if $l(u) = l(v)$, then the vertices u and v are identical, i.e., $u = v$.

C6 (Hold consistency): Let u be a vertex in G_A. Let v $(\in pre(u))$ be a predecessor of u. If $|pre(u)| < |pre(l(u))|$ and $l(u) = l(v) = w$, then $r(w) = h$ and $|pre(u)| = 1$.

C7 (Input Independency): Let u, v be two vertices in G_A. Let p_i and p_j be a pair of reconvergent paths that start from u and end at v. Let w be a vertex on p_i such that $u \in pre(w)$. Let x be a vertex on p_j such that $u \in pre(x)$. For each pair of paths p_i, p_j where $w \neq x$, $|pre(w)| = |pre(l(w))|$ and $|pre(x)| = |pre(l(x))|$.

The following three examples are used to explain condition C3.

Example 4.1. Figure 24(a) shows the time expansion models of S3 that does not satisfy condition C3(ii)(a). Condition C3(ii)(a) tells that if a vertex w' at time m is used to justify another vertex u at time $m+1$, then w' cannot be used simultaneously to activate a thru function at time m. In Figure 24(a), $I1$ at time 3 (corresponding to w' at time m) is used to justify $R2$ at time 4 (corresponding u at time $m+1$) but at the same time $I1$ is used to activate thru functions t_1 and t_4. This violates Condition C3(ii)(a).

Example 4.2. Condition C3(ii)(b) tells that if a vertex $r = l(x)$ is in HOLD mode at time m, r cannot be used to justify signal for any thru tree through a thru function at time m. But r can be used to activate a thru function in a thru tree at time m. In Figure 24(b), $R2$ is in hold mode at time 2 (corresponding to r at m). At time 2, $R2$ is used to activate thru functions t_1. At the same time, $R2$ is used to justify the value of $O2$ at 3. This violates Condition C3(ii)(b).

Example 4.3. Condition C3(ii)(c) tells that when a vertex $r = l(x)$ at time m is used to activate a thru function going to vertex u at time $m+1$, r cannot be used to justify signal for any thru tree through a thru function at time m. But r can be in HOLD mode. For example, Figure 24(c) shows that $R2$ (corresponding to r) is activating thru function t_1 at time 3 and at the same time $R2$ is justifying $R5$. But these two events are not allowed to happen at the same time.

As the first step of designing a test generation procedure for acyclically testable sequential circuits, the logic for hold function is assumed fault free. The tests for these faults can be generated separately and the test generation procedure for these faults will be considered in the future works. To guarantee the test generation for faults in thru functions, each register in the feedback vertex set are regarded as having reset function.

Definition 4.2. Let S be a given acyclically testable sequential circuit. The ***acyclically-extended time expansion model (ATEM)*** of S is the combinational equivalent obtained by the following procedure.

1. For each time frame, replace each vertex with a connection without a register and replace each arc with the combinational logic block where the corresponding combinational path (represented by the arc) is located. Each combinational logic block appears at most once at each time frame.

2. A logic gate in each logic block is removed if it is not reachable to any input of other logic blocks.

3. Each input that corresponds to an output of a register is assigned don't care value.

Example 4.4. Figure 23 shows the ATEM of S3 with respect to output O2.

5 Test Generation Procedure

For each stuck-at fault in a given acyclically testable sequential circuit, the test generation process is done as follows using ATEM test generation algorithm. Multiple fault modeling of [6] is considered.

Step 1 Generate an acyclically-extended time expansion model (ATEM) of the sequential circuit.

Step 2 Transform the combinational equivalent ATEM into its multiple fault model.

Step 3 Apply combinational ATPG on the multiple fault model.

Step 4 Derive the test sequence from the test pattern obtained from the test generation on the multiple fault model of the ATEM.

Theorem 5.1. *The ATEM test generation algorithm can identify redundancy and all testable faults.*

Theorem 5.2. *The test generation complexity of the acyclically testable sequential circuits is τ^2-bounded.*

6 Design-for-Test Method

In this section, a design-for-test(DFT) method to augment a given sequential circuit into an acyclically testable sequential circuit is introduced. The DFT method performs some operations on R-graph and it is designed to induce minimum area overhead. The procedure consists of the following three steps.

Step 1 Identify the vertices of minimum feedback vertex set (MFVS).

Step 2 Identify existing thru trees.

Step 3 Group the vertices of MFVS into two groups, G1, G2 and G3 as follows.

3.1 Group a vertex u into G1 if it corresponds to a register or input/output that activate a thru function. If the vertex is in an existing thru tree T_i, group all the vertices in T_i in G1. If G1 has only input/output, G1 is made empty.

3.2 Group the remaining register vertices in MFVS into G2.

3.3 Group the remaining input/output vertices into G3.

Step 4 For each group of G1 and G2, the following is done.

4.1 Check that at least one input vertex and one output vertex exist in the group. If the group does not have input vertex (resp. output vertex), one input vertex (resp. output vertex) is taken from G3. If G3 does not have one, a new vertex is added into the group.

4.2 Group each vertex (except output vertex) into a group called potential source if the vertex does not have an outgoing arc labeled with a thru function.

4.3 Group each vertex (except input vertex) into a group called potential destination if the register vertex does not have an incoming arc labeled with a thru function.

4.4 For each vertex u in the group of potential source, introduce a new outgoing arc labeled with a new thru function t_{new} to connect u to a vertex v in the group of potential destination. u and v are taken out from the groups of potential sources and potential destination, respectively.

4.5 Repeat 4.4 until the group of potential destination is empty or the group of potential desitination has only output vertices.

4.6 For each vertex u in the group of potential source, introduce a new outgoing arc labeled with a new thru function t_{new} to connect u to an output vertex v that does not have an incoming arc labeled with thru functions. If the group does not have one, an output vertex is taken from G3 to the group. If G3 does not have one, a new output vertex is introduced to the group.

Step 5 If G1 is not empty, each register in G1 and G2 is augmented into a hold register. For other register vertices in MFVS, each register is augmented into a register with reset function.

Step 1 is done by using an exact algorightm for selecting partial scan flip-flops introduced in [7]. All the new thru functions t_{new} introduced in the DFT method are same. For example $t_{new} = r$ means the new thru function is activated when $r = 1$ where r can be an existing primary input or a new primary input.

7 Case Studies

In the case studies, experiments are conducted on RTL benchmark circuits, which are datapaths of varying bit width. Our DFT method is applied on the datapaths of GCD, LWF, JWF, and MPEG and compare the area overhead of the augmented circuits with that of the full scanned circuits and the partial scanned circuits. Partial scanned circuits are the circuits whose minimum feedback set of flip-flops are scanned so that the augmented circuits are acyclic. Thus, the circuits modified with partial scan and with our DFT method have same test generation complexity. Table 1 presents the characteristics of the benchmark circuit. Table 2 shows the fault coverage and fault efficiency of each benchmark circuit. Each fault testable in the partial scan designed circuits is also testable in the corresponding circuit augmented by our DFT method, and vice versa. Table 3 shows the area overhead where one unit of area corresponds to the size of an inverter and pin overhead. It shows that the area overhead of the benchmark circuits augmented by our method is less than that of the full scanned circuits and the partial scanned circuits. The pin overhead in our method comes from the reset function and extra input to control the new thru functions. Table 4 tells that the test generation time for the original circuits is large while the test generation time for the partial scan designed circuits as well as the acyclically testable sequential circuits is small. Table 5 gives the information that the test application time of the circuits under our augmentation is more than the original circuits' but less than the partial scan.

Table 1: Characteristics

	Original			
B/mark	#Flip-flops	Area	#Primary inputs	#Primary outputs
GCD	48	1383	40	19
LWF	80	1763	39	32
JWF	224	5925	106	80
MPEG	1928	46772	499	128

Table 2: Number of faults, fault efficiency and fault coverage

	Original		Full scan		Partial scan		Our method	
B/mark	FC(%)	FE(%)	FC(%)	FE(%)	FC(%)	FE(%)	FC(%)	FE(%)
GCD	99.75	99.75	100	100	100	100	100	100
LWF	99.94	99.94	100	100	100	100	100	100
JWF	98.70	98.70	100	100	100	100	100	100
MPEG	84.80	84.80	100	100	100	100	100	100

8 Conclusion

A new class called acyclically testable sequential circuits has been introduced. The test generation complexity of the acyclically testable sequential circuits is τ^2-bounded. On the other hand, acyclically testable sequential circuits are at-speed testable. The DFT method to augment an arbitrary sequential circuit into an acyclically testable sequential circuit has been introduced. Experimental results showed that the area overhead of the resulting augmented circuits is less compared to the partial scan designed circuits. Complete fault efficiency is also achieved and the test generation time is low. Moreover, the test application time is less than the test application time of the full scanned circuits and partial scanned circuits.

Table 3: Area and pin overhead

	Full scan		Partial scan		Our method	
B/mark	Area(OH%)	Pin OH	Area(OH%)	Pin OH	Area(OH%)	Pin OH
GCD	1719(24.30)	3	1495(8.10)	3	1415(2.31)	1
LWF	2323(31.76)	3	1875(6.36)	3	1798(1.99)	2
JWF	7493(26.46)	3	6485(9.45)	3	5957(0.54)	2
MPEG	60268(28.85)	3	47612(1.80)	4	47556(1.68)	2

Table 4: Test generation time and test application time

B/mark	Test generation time(s)				Test application time (clock cycles)			
	Original	Full scan	Partial scan	Our method	Original	Full scan	Partial scan	Our method
GCD	87.19	0.02	0.19	0.43	159	6124	3334	815
LWF	49.02	0.02	0.06	0.40	59	4049	1444	196
JWF	1689.14	0.08	0.50	13.48	103	17100	12488	1648
MPEG	2646.42	0.18	12.05	33.91	114	162035	31822	9690

References

[1] C. Y. Ooi and H. Fujiwara, "Classification of sequential circuits based on τ^k-Notation," Proc. of ATS, pp. 348-353, Nov. 2004.

[2] C. Y. Ooi, T. Clouqueur and H. Fujiwara, "Classification of sequential circuits based on τ^k notation and its applications," Trans. of IEICE on Information and Systems, pp. 2738-2747, Dec. 2005.

[3] T. Inoue, T. Hosokawa, T. Mihara and H. Fujiwara, "An optimal time expansion model based on combinational ATPG for RTL circuits," Proc. 7th ATS, pp. 190-197, Dec. 1998.

[4] C. Y. Ooi and H. Fujiwara, "A new class of sequential circuits with acyclic test generation complexity," 24th IEEE ICCD, October 2006 (To appear).

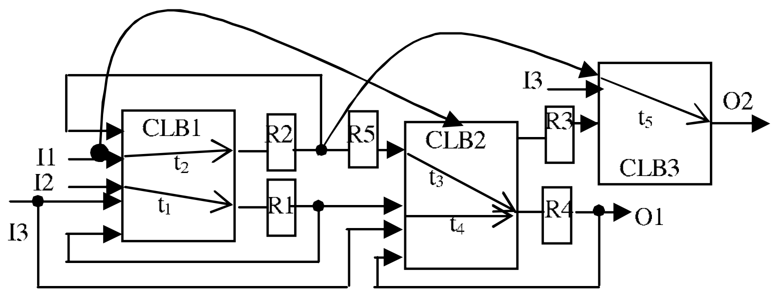
[5] T. Inoue, D. K. Das, C. Sano, T. Mihara, H. Fujiwara, "Test generation for acyclic sequential circuits with hold registers," Proc. 18th Int. Conf. on Computer Design, pp. 550-556, 2000.

[6] C. Y. Ooi, T. Clouqueur and H. Fujiwara, "Test generation complexity for stuck-at and path delay faults based on τ^k-notation," NAIST Technical Report, NAIST-IS-TR2005003, May 2005.

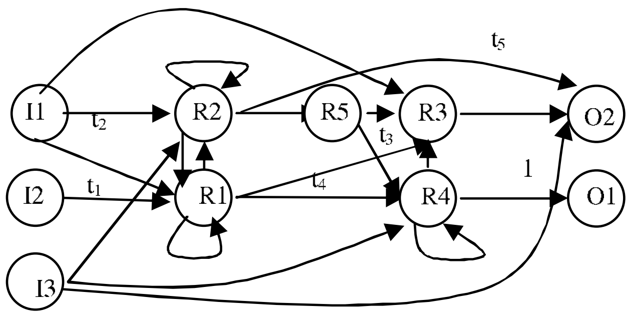
[7] S. T. Chakradhar, A. Balakrishnan, V. D. Agrawal, "An exact algorithm for selecting partial scan flip-flops," JETTA, pp. 83-93, 1995.

[8] H. Iwata, T. Yoneda, S. Ohtake and H. Fujiwara, "A DFT method for RTL data paths based on partially strong testability to guarantee complete fault efficiency," Proc. IEEE the 14th ATS, pp. 306-311, Dec. 2005.

[9] C. Lin, M. Marek-Sadowska, M.T. Lee and K. Chen, "Cost-free scan: a low-overhead scan path design," IEEE Trans. CAD Integra. Circuit and Sys., Vol. 17, No. 19, pp. 852-861, Sept. 1998.

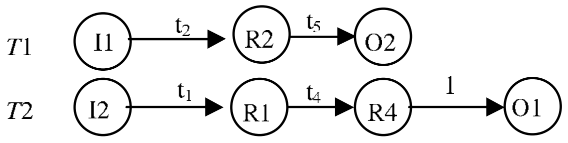


(a) Sequential circuit S3



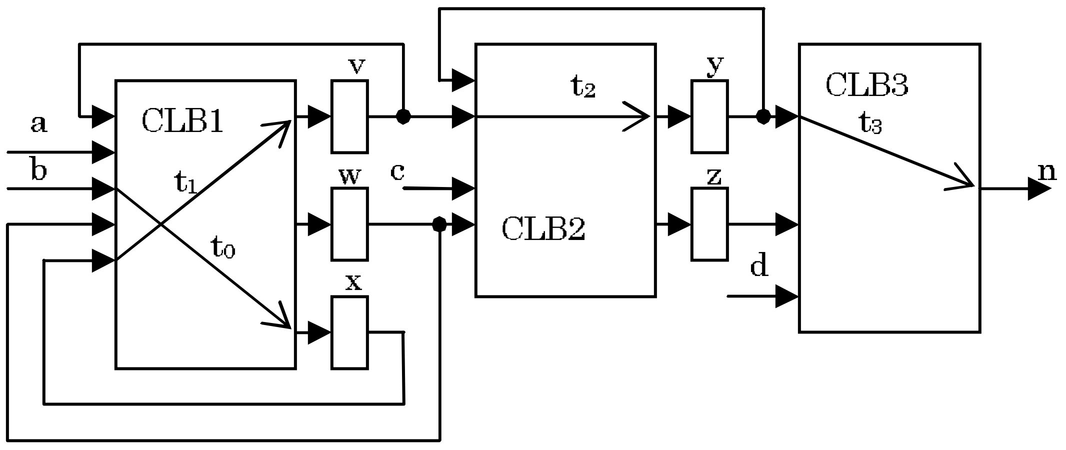
t_1= (R2) (I1); t_2= I3; t_3= I3; t_4= (R2) (I1); t_5= I3
r(R2)=h, r(R1)=h, r(R4)=h

(b) R-graph of S3

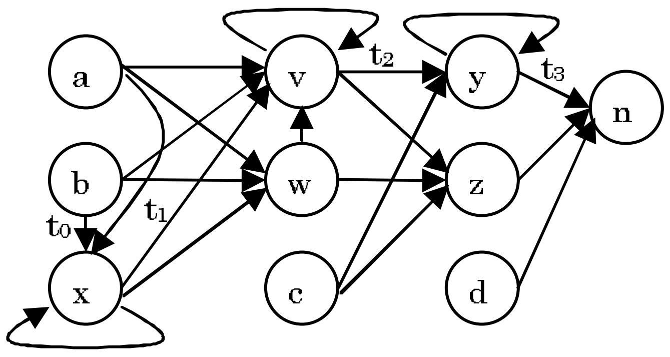


(c) Thru trees *T*1 and *T*2

Figure 12: Sequential circuit S3.



(a)



(b)

Figure 13: S4 (a) and its R-graph (b).

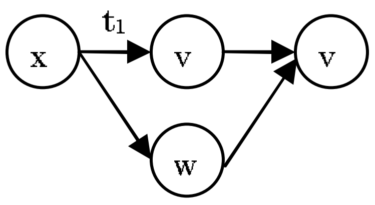
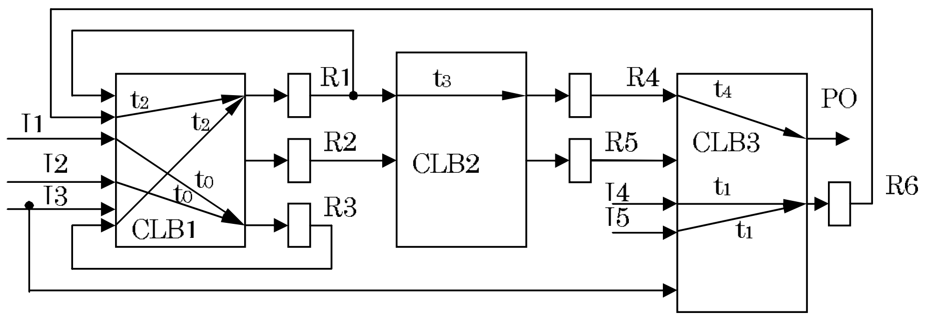
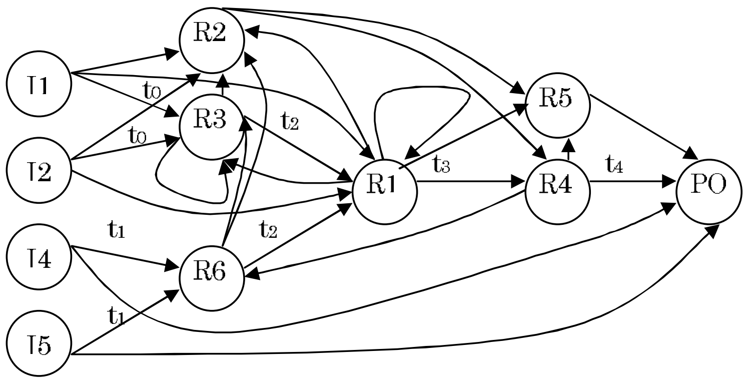


Figure 14: Two paths that have input dependency.

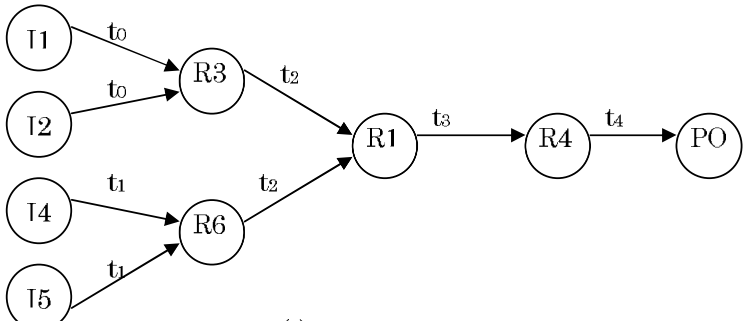
1



(a) S5

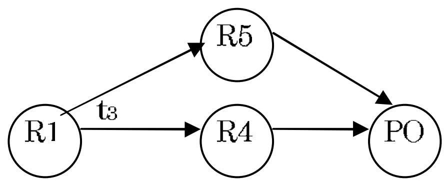


(b) R-graph for S5

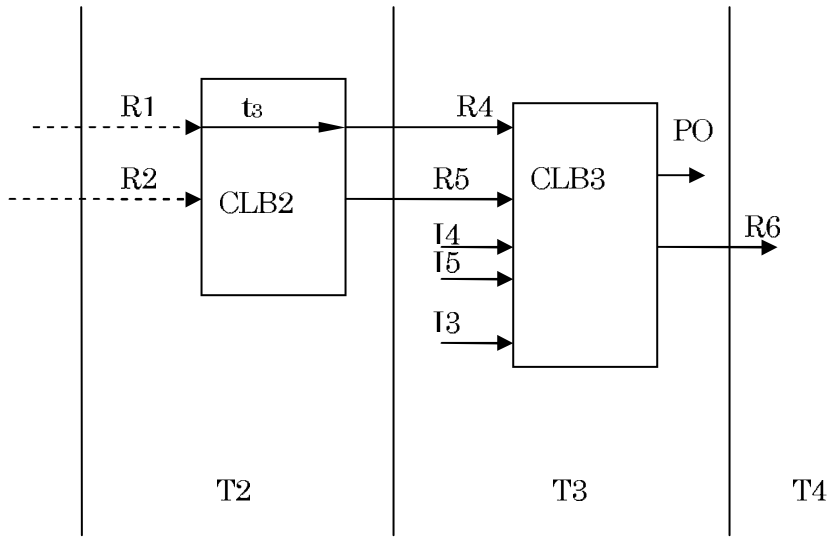


(c) Thru tree for S5

Figure 15: Sequential circuit S5.



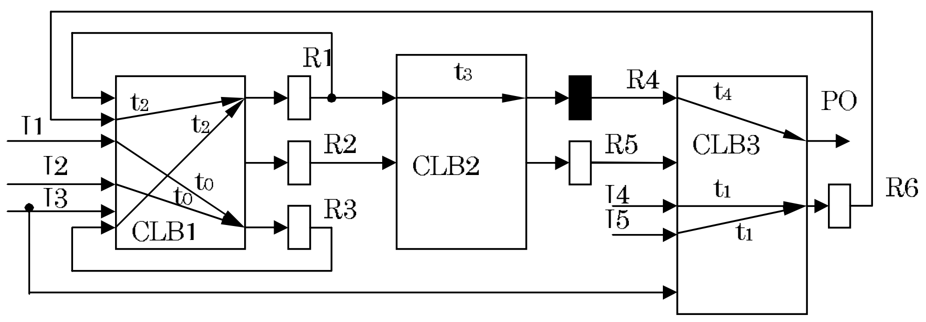
(a) Input dependency



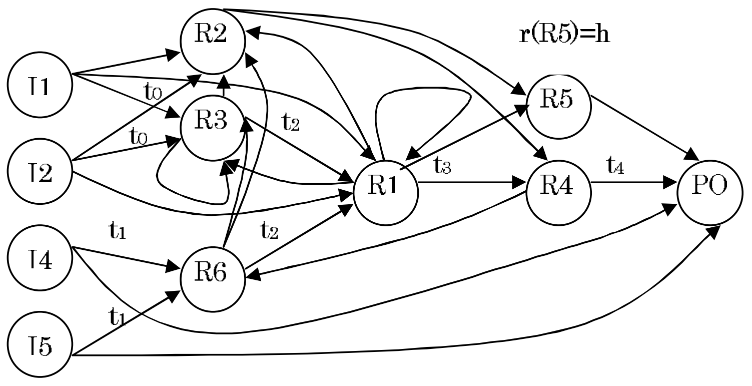
(b) Input dependency between path R1→R4→PO and path R1→R5→PO.

Figure 16: Input dependency in S5.

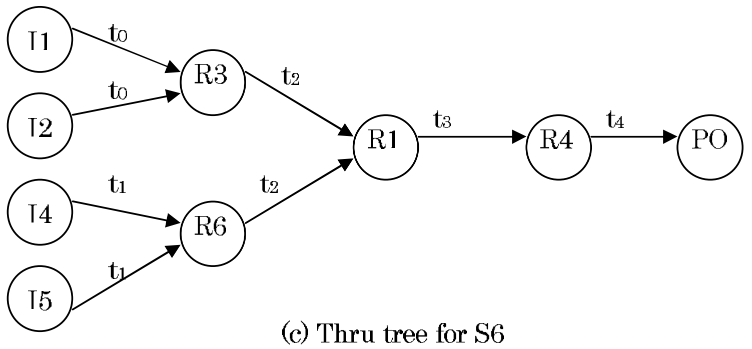
1



(a) S6

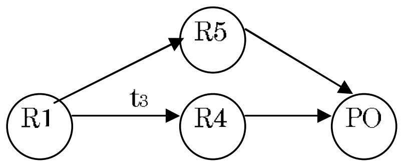


(b) R-graph for S6

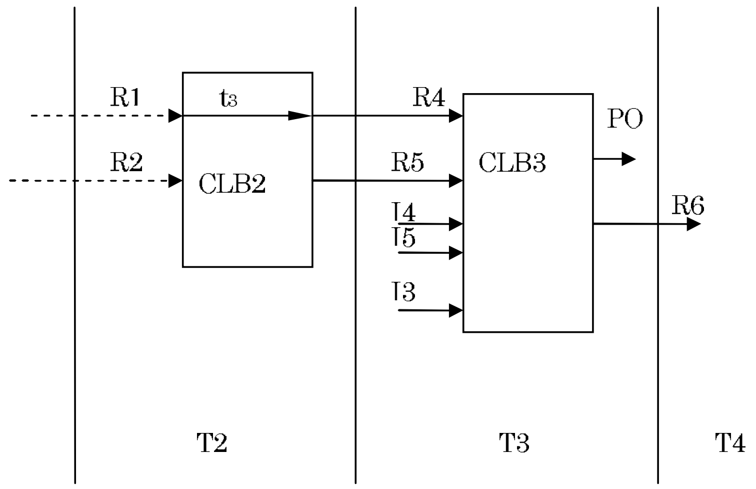


(c) Thru tree for S6

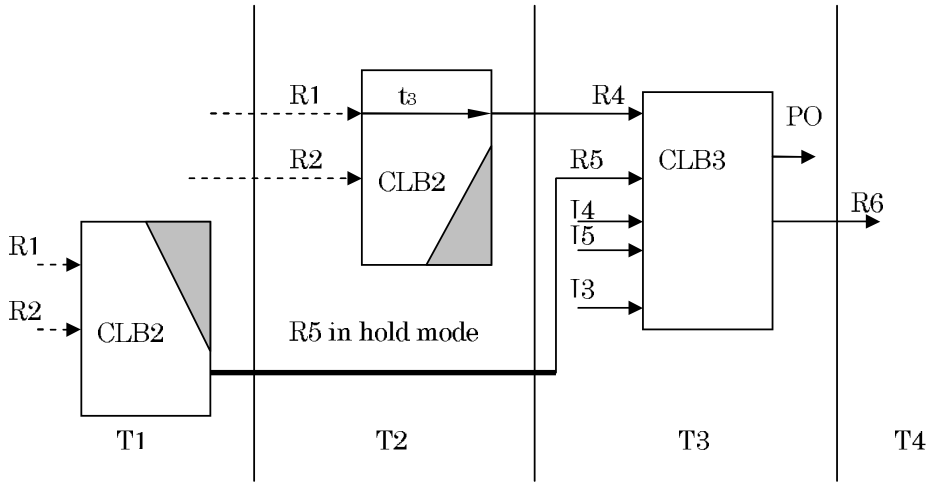
Figure 17: Sequential circuit S6 with hold register R5.



(a) Input dependency

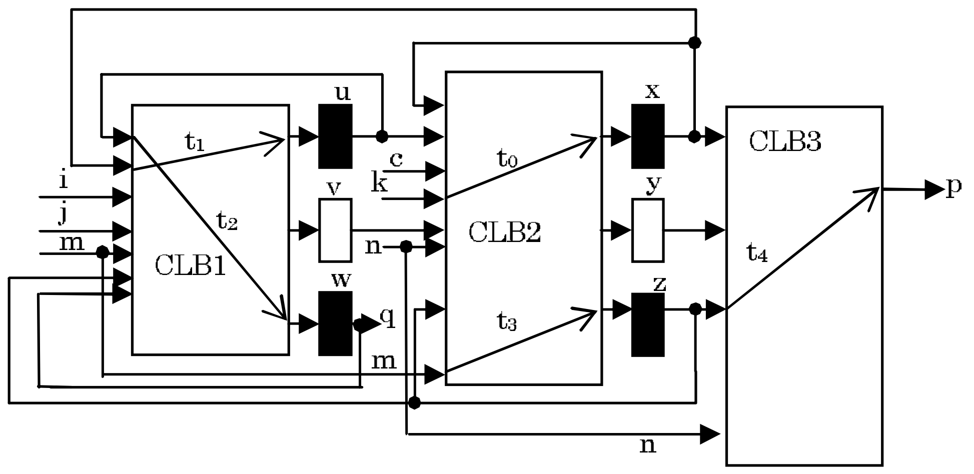


(b) Input dependency between path R1→R4→PO and path R1→R5→PO.

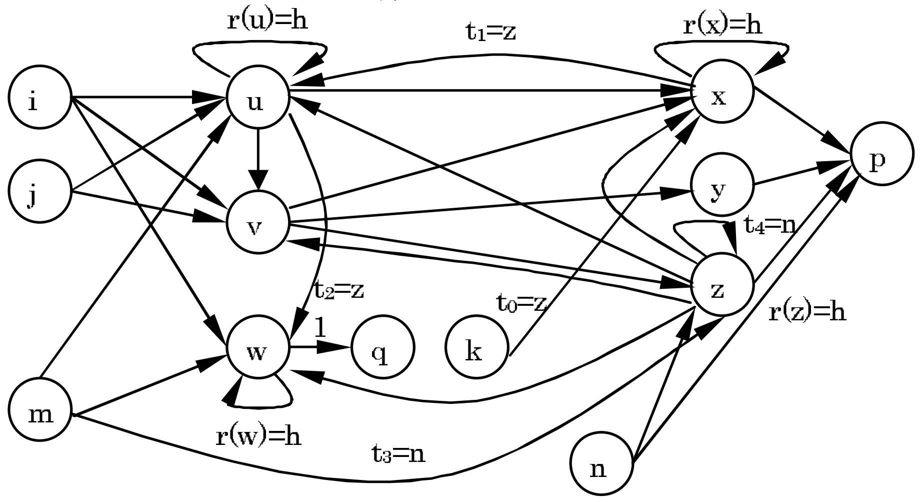


(c) Input dependency between path R1→R4→PO and path R1→R5→PO is resoved by holding R5.

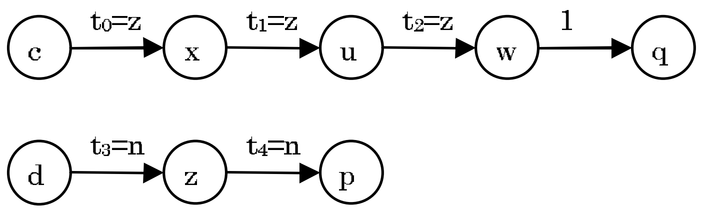
Figure 18: Resolution of the Input dependency in S6.



(a) S7



(b)R-graph for S7



(c) Thru trees for S7

Figure 19: S7 (a) and its R-graph (b).

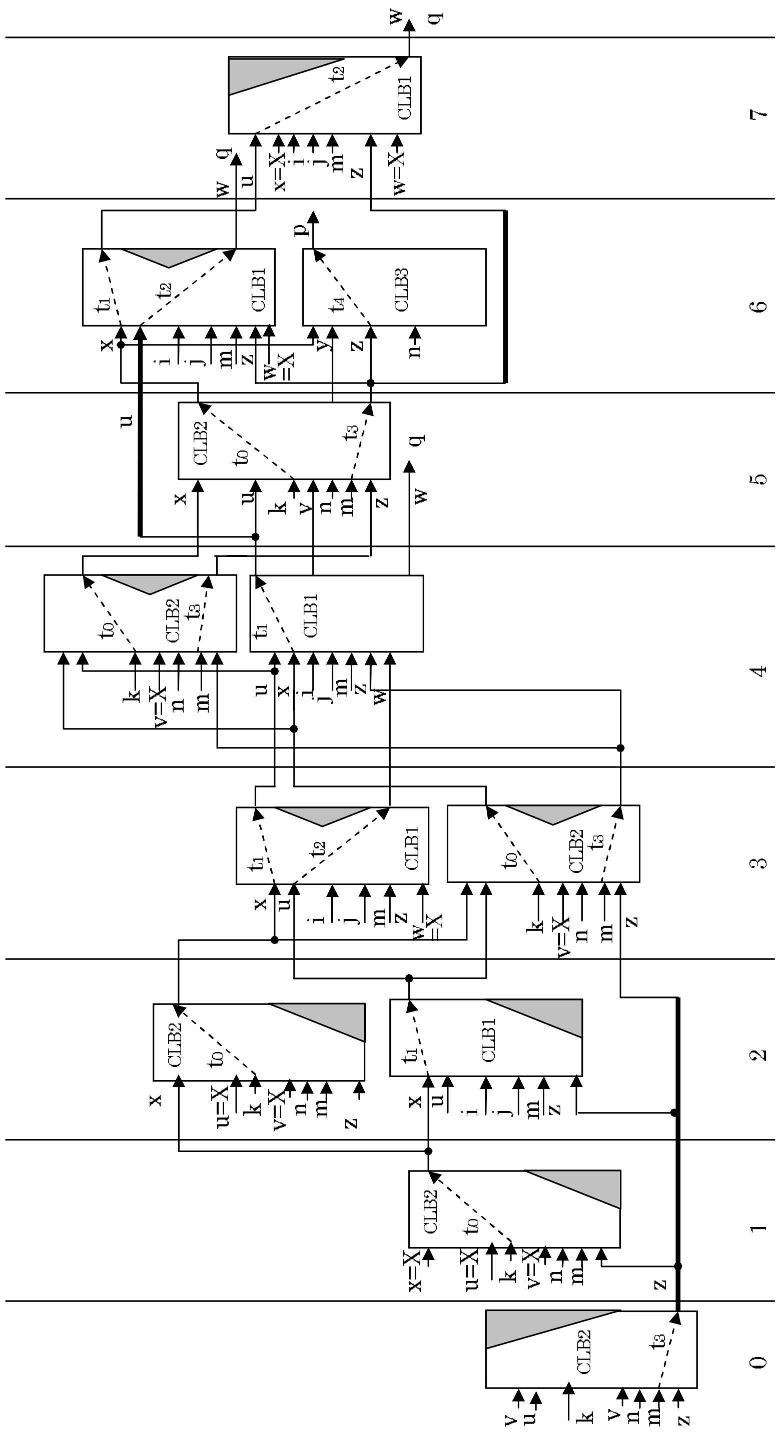
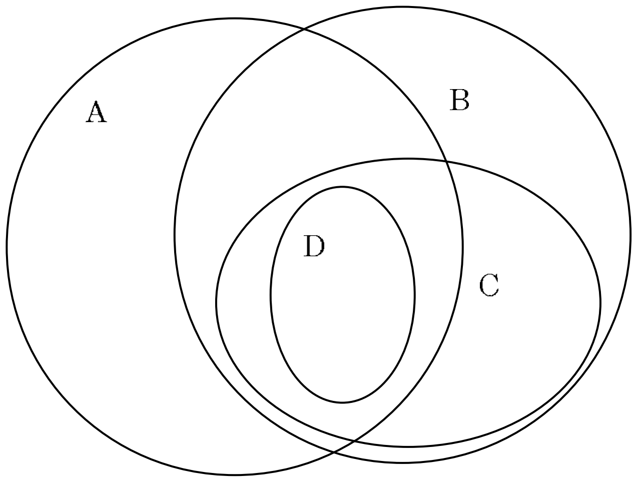


Figure 20: Time expansion model of S7.

$$\begin{tikzcd} T2 \arrow[r] & T1 \end{tikzcd}$$

Figure 21: Dependency graph of $T1$ and $T2$ of S3.



A: Acyclically Testable Sequential Circuits type A.

B: Acyclically Testable Sequential Circuits type B.

C: Acyclically Testable Sequential Circuits type C.

D: Acyclic Sequential Circuits

Figure 22: Relationship between acyclically testable sequential circuits and acyclic sequential circuits.

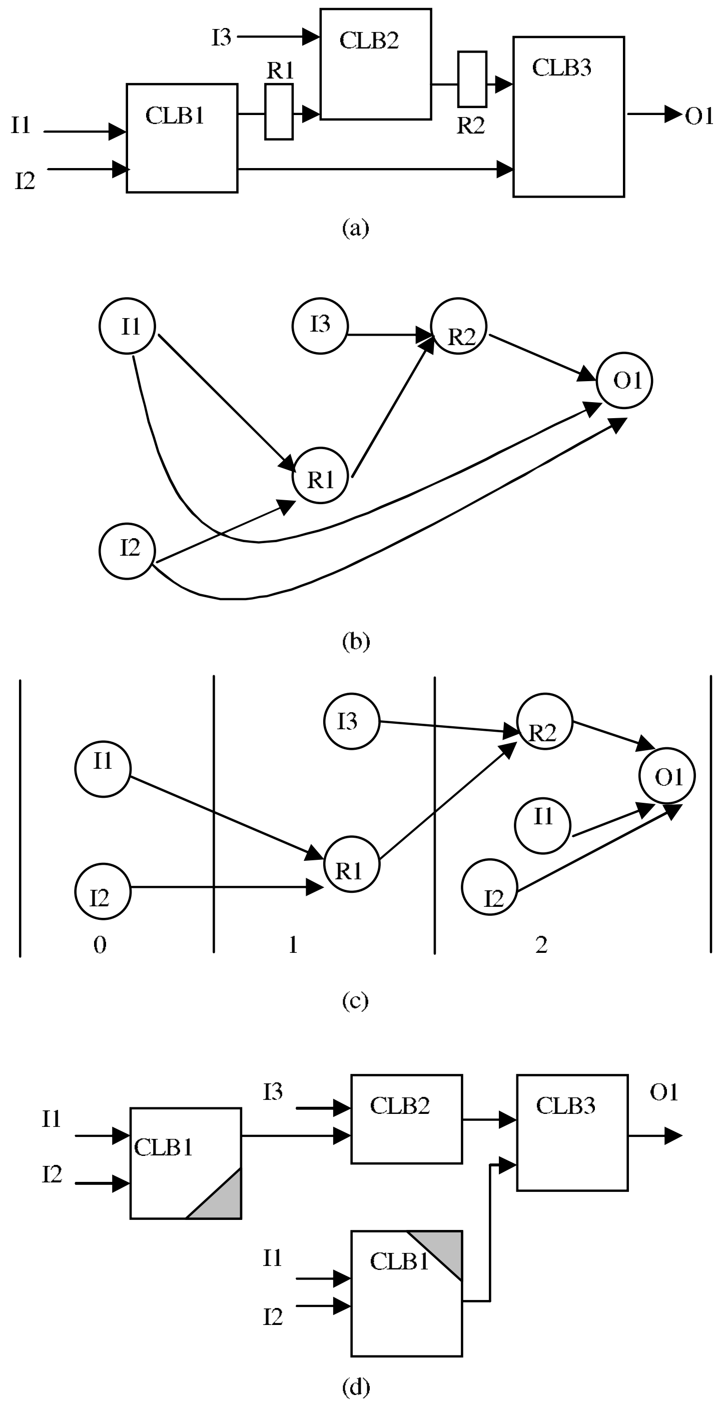
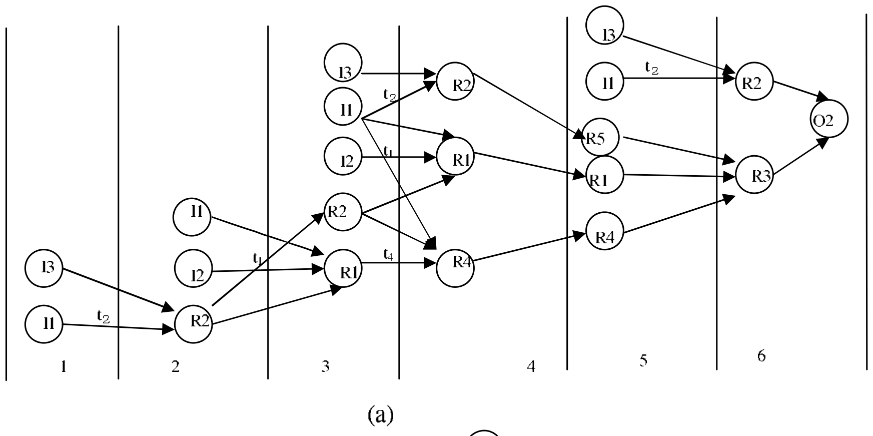
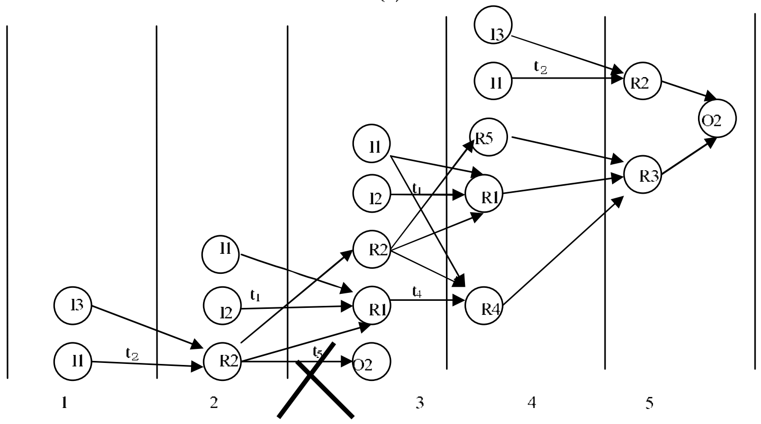


Figure 23: Example of time expansion model.(a) Acyclic sequential circuit S8. (b) R-graph of S8. (c) Time expansion graph of S8. (d) Time expansion model of S8.


(a)


(b)

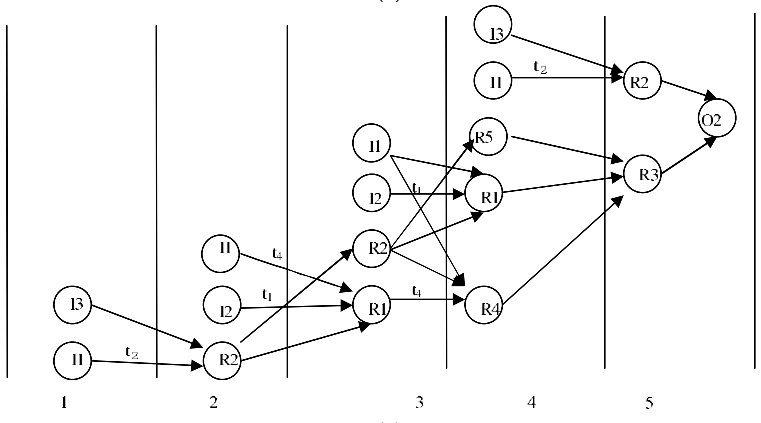

(c)

Figure 24: Time expansion models of S3 that violates C3.

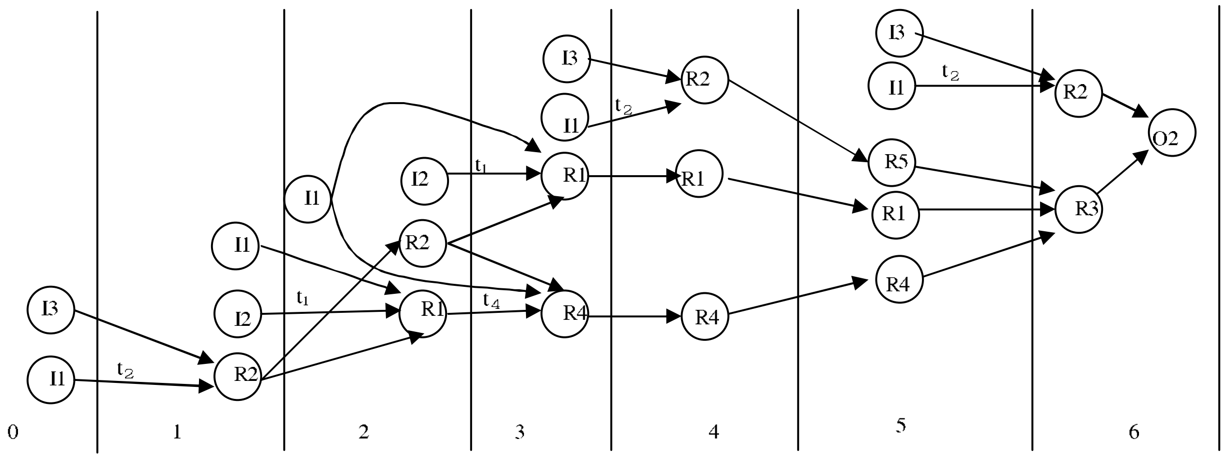


Figure 25: ATEM for S3.