Abstract

Performance driven synthesis of sequential circuits relies on techniques such as retiming and resynthesis [5] [6] [10]. These operations change the topological structure of the circuit, but should guarantee that the temporal behavior between the inputs and outputs of the circuit is preserved. In this paper, we provide an efficient procedure to check for temporal equivalence of sequential circuits.

1 Introduction

Circuit verification has received great attention because of its role in detecting design errors at an early stage. Functional (logic) verification of sequential circuits (checked by existing methods) and focus primarily in the functional elements (gates) and global topological structure of the circuit. They may not have identical temporal properties e.g., the clock cycle often depends on the circuit implementation.

The verification of temporal properties of a sequential circuit is still an open problem. Leiserson et al. [15] defined the behavior of a simple edge-triggered circuit. They related the behavior to the number of memory elements in each cycle in the circuit. They also proved that it was necessary to preserve this number in order to preserve the behavior. In Section 4, we extend this concept to include single phase level-sensitive circuits. Sakallah et al. used the notion of correct behavior of a circuit for a given clocking scheme implicitly (8), as the existence of early (and late) arrival and departure times at each latch and showed that these times must satisfy clocking constraints for correct operation.

In this paper, we extend the abstract notion of temporal behavior to compare arbitrary circuits with arbitrary multi-phase clocking schemes. We are concerned about the input/output behavior of circuits with respect to time. We assume the logical equivalence of two circuits (checked by existing methods) and focus primarily in the temporal behavior of the circuit. A multi-phase clocking scheme is [10] assumed. We define temporality to be the temporal behavior of the circuit. In the next section, we provide a formal definition of this term. We prove conditions under which this property is preserved. In Section 2, we introduce the basic terminology. Section 3 deals with multi-phase edge-triggered systems. This provides a means of extending the conditions to multi-phase mixed systems (Section 4). Results of the implementation on a few circuits are given in Section 5 and Section 6 provides the conclusions.

2 Definitions

A single clock multi-phase circuit can be considered to consist of two kinds of elements, functional elements (gates) and globally clocked memory elements. It is modeled as a finite vertex-weighted, edge-weighted, directed multi-graph \(G = (V, E, D, d, w)\). Every vertex represents a functional element (gate). The vertex-weights \(D_i, d_i\) are the maximum and minimum delays through the functional element at vertex \(v_i^1\). There is a directed edge \(e_{ij}\) from vertex \(v_i\) to vertex \(v_j\) if the output of the gate at \(v_i\) is an input the gate at \(v_j\). Vertex \(v_i\) is called the input node (out-node) of the edge, denoted by IN\((e_{ij})\) (OUT\((e_{ij})\)). The edge weight on edge \(e_{ij}\) is referred to as \(w_{ij}\). A path \(u \rightarrow v\) is a set of alternating vertices and edges \([v_0, e_{a_1}, v_1, e_{a_2}, \ldots, e_{a_{n-1}}, v_n]\), such that every pair of successive edges form an input-output pair to the vertex between. A path may have repeated edges.

Memory elements can be either edge-triggered or level-sensitive. The retardation [3] at a level-sensitive latch is the amount of time by which the data input is delayed since the beginning of the active period. For correct operation of both memory elements, the data signal should be stable at the input before the latching edge occurs by an amount called the set-up time and should remain stable after the latching edge has occurred by an amount called the hold time. We assume, without loss of generality, that edge-triggered elements sample input data on the falling transition and the level-sensitive latches use the time when the phase is high as the active period. Thus, the falling edge of each phase is the latching edge with respect to which set-up and hold constraints must be satisfied. Henceforth, we shall assume that all circuits satisfy the set-up and hold requirements and that the circuit operates in the fundamental mode. This ensures that there are no "wave-pipelining" like effects [12] in the circuit. A special vertex in \(G\), called the host vertex \(v_h\), represents the environment. The host vertex has edges into all primary inputs and has edges from all primary outputs. The graph \(G\) is cyclic, not only with cycles through the host \(v_h\), but also possibly internal cycles. The set of cycles internal to \(G\), due to the circuit structure, are called internal cycles. The set of cycles that contain the host vertex are called external cycles [see Figure 1]. We use the terms circuit and graph interchangeably.

![Figure 1: External and Internal Cycles](image)

The clock \(\phi\) has \(k\) phases i.e. \(\phi = (\phi_1, \phi_2, \ldots, \phi_k)(|\phi| = k)\).

The edge weight \(w_{ij}\) is an integer, \(w : \xrightarrow{} \{0, 1, \ldots, k\}\), which reflects the phase of the memory element (if any) on edge \(e_{ij}\).

\[
w_{ij} = \begin{cases} 
0 & \text{if there is no memory element on } e_{ij} \\
\text{phase of memory element on edge } e_{ij} & \text{otherwise}
\end{cases}
\]

We assume that wiring delay (delay through the edges) is insignificant or can be approximated as a function of the fanout of the vertex \(v_i\), and included in the vertex delay \(d_i\).
If the output of a memory element is an input to another memory element, we introduce a dummy vertex with zero delay between the two. This forces every edge to have only one memory element. Let $c$ denote the clock cycle. Associated with each phase of $c$ is an edge labeled $s_i$, $p_i$, $x_i$, $s_i$, $x_i$, $c$, the times at which the phase rises and falls respectively. We order the phases so that $x_1 \leq x_2 \cdots \leq x_n = c$.

All memory elements (henceforth referred to as a latch) are assumed to be edge-triggered to begin with. We later relax this restriction to include mixed circuits.

Define the temporality of a vertex $v_j$ along a path $p$ from the host vertex, to be as follows:

$$
\varphi(v_j, p) = \left\{ \begin{array}{ll}
\text{least number of clock phases after which signal at output of } v_j \text{ is dependent on an input along } p & \text{after the circuit has started computation} \\
0 & \text{if } v_j \text{ is a primary input} \\
\varphi(e_{i_j}, p) + \left| s_i - s_j \right| & \text{otherwise, } v_i = \text{IN}(e_{i_j}), v_j \in p.
\end{array} \right.
$$

We restrict the path to pass through the host vertex only once. Similarly define the temporality of an edge $e_{i_j}$ along a path $p$ to be the least number of clock phases after which the signal on the edge (and at the output of the latch, if $e_{i_j}$ has a latch) is dependent on an input along the path $p$. We can now recursively define $\varphi(x, p), x \in V \cup E$ as follows.

$$
\varphi(v_j, p) = \left\{ \begin{array}{ll}
\varphi(v_i, p) & \text{if } v_j \text{ is a primary input} \\
\varphi(v_i, p) + |s_i - s_j| & \text{otherwise, } v_i = \text{IN}(e_{i_j}), v_j \in p.
\end{array} \right.
$$

Let $\rho_i(p)$ denote the phase of the last latch encountered along path $p$, when we reach vertex $v_j$.

$$
\varphi(e_{i_j}, p) = \left\{ \begin{array}{ll}
\varphi(v_i, p) & \text{if } w_i = 0 \\
\varphi(v_i, p) + |s_i - s_j| & \text{if } \rho_i(p) < w_i \\
\varphi(v_i, p) + |s_i - s_j| - \rho_i(p) & \text{if } \rho_i(p) \geq w_i.
\end{array} \right.
$$

Intuitively, the temporality represents the latency of the latch or gate along the path $p$, assuming that all latches have been initialized correctly, $\varphi(x, p) \in V \cup E$ is the number of phases after which $x$ has a valid signal (dependent on the primary input in $p$). Consider the circuit in Figure 2. Let $p$ be the path \{c, e, e, e, f, e, g, e, p, e\}. Then

$$
\begin{align*}
\varphi(c, p) &= 0, \\
\varphi(e, p) &= 0, \\
\varphi(e, p) &= 2, \\
\varphi(f, p) &= 2, \\
\varphi(g, p) &= 2, \\
\varphi(p, p) &= 2.
\end{align*}
$$

Figure 2: Temporality along a path

In short a signal from $c$ takes 4 phases to reach the output. The temporality of a circuit $\varphi(G)$ is said to be preserved if the temporality of all outputs is preserved for all paths from primary inputs to primary outputs. Mathematically $\varphi(G)$ is a map from the set consisting of the product of primary outputs ($PO$) and valid paths to them ($P$) to integers. A path is valid to a primary output if the primary output lies on the path.

$$
\varphi(G) : PO \times P \rightarrow Z
$$

The output of $G$, at any time can be interpreted as the result of the causal interaction of several paths in $P$. The set $P$ can be infinite (it is finite when we have a cyclic pipelined circuit).

3 Multi-Phase Edge Triggered Systems

For the case of sequential circuits with edge-triggered and level-sensitive latches we assume that gates are characterized by the maximum and minimum delays through them. Correct operation requires the phases to separated appropriately.

For any path $p$ in $G$ define the number of tokens $\tau(p)$ to be the number of clock cycles needed for computation along $p$. $\tau(p)$ can be computed by scanning the path $p$ for successive latches, $l_m$ followed by $l_n$ with $\phi_{l_m} \leq \phi_{l_n}$. In other words, during the computation of $\varphi(v_i, p)$, $\tau(p)$ is the number of times we encounter the last case in the definition of $\varphi(v_i, p)$. The theorem given below provides the invariant that preserves temporality in the case of retiming. We use the subscript $r$ to denote the variables after retiming.

Theorem: Let $r : G \rightarrow G_r$ represent an operation of retiming. Assume all primary outputs are latched. The temporality of $G$ and $G_r$ is the same if and only if the number of tokens in each cycle is preserved and $r : G \rightarrow G_r$.

Assumptions: The clocking scheme does not change under this operation and the primary output latches do not move.

Proof: Consider any primary output in $G$ and the corresponding output in $G_r$, say $p_o$. Let $p_o$ be latched on phase $\phi_i$. Then we can write for any path $p$ from $v_i$ to $p_o$.

$$
\varphi(p_o, p) = \tau(p)\phi_i + j
$$

where $\phi_i$ is the total number of phases.

(\Rightarrow) (If part): Let the number of tokens be preserved along all cycles. Now any path from $v_i$ to any primary output $p_o$ can be seen as a part of a cycle in the graph. To obtain a cycle from such a path, we only need to add the edge from the primary output to $v_i$. Since the tokens along all cycles are conserved then clearly the value of $\varphi(p_o, p)$ remains the same for all primary outputs $p_o$ along all paths $p$.

(\Leftarrow) (Only if part): Assume there is a cycle $C$ in which the number of tokens is not preserved then the temporality is not preserved. There arise two cases.

- case 1: $C$ is an external cycle. Without loss of generality assume that the number of tokens after retiming in a cycle $C_r$ denoted by $\tau(C)$ increases. Let $p_o$ be the primary output on $C$. Since tokens along all cycles are conserved then clearly the value of $\varphi(p_o, p)$ remains the same for all primary outputs $p_o$ along all paths.

$$
\tau(C) < \tau_r(C)
$$

and the output of the latch has not changed during retiming, then

$$
\varphi(p_o, p) < \varphi_r(p_o, p).
$$

Hence temporality is not preserved.

- case 2: $C$ is an internal cycle. Since the presence of a clock implies periodicity, an internal cycle represents a set of infinite paths from at least one primary input to at least one primary output. Consider an internal cycle shown in Figure 3.

Let $c$ be the vertex where a path $(p_{o,a})$ from $v_i$ meets $C$. Let $b$ be the vertex in $C$ where a path $(p_{o,b})$ exists from $b$ to a primary output. In case 1 we showed that tokens in all external cycles must be preserved. Consider the set of extended cycles $C_m$ obtained by traversing $p_{o,a}, p_{o,b}, p_{o,c}$ and $m$ times through the cycle $C$.

$$
C_m' = p_{o,a} \cup C \cup p_{o,b} \cup p_{o,c}.
$$
This is impossible since
\[ r(C) < r_r(C). \]  

**Observation:** Let \( r: G \rightarrow G_r \) be a retiming of \( G \), where \( G \) is a single phase edge-triggered circuit. Leiserson's formulation of retiming is an integer-valued vertex labeling. A retiming is legal if after retiming no edge has a negative memory element count. Leiserson's formulation is equivalent to preserving the number of latches (the number of tokens is the same as the number of latches for single phase edge-triggered circuits) in each cycle.

The temporality defined in the previous section depends on the number of phases in the clock set \( \phi \). Let us define the \( r \)-temporality to be the temporality using the clock cycle \( c \) as a parameter i.e.
\[ \varphi(x, p) = c \left( \frac{\varphi(x, p)}{\phi} \right) + r_r(x, p) \pmod{\phi}, \quad x \in V \cup E. \]  

The tokens along a path from \( x \) to \( y \) can be related as
\[ r(p) = \varphi(x, p) + r_r(x, p) \pmod{\phi}, \quad \text{where } y \xrightarrow{p} x. \]  

Let \( r_r(x, p) \pmod{\phi} \) be called the remainder \( r(p) \). This is the full time of the phase for the last latch encountered. The \( r \)-temporality just gives the temporality in terms of time rather than the phase, i.e. a sum of a multiple of the clock cycle \( c \) and the phase of the last latch along the path.

We now introduce the concept of generalized retiming. In generalized retiming, we permit the clock set \( \phi \) to change. Look upon the \( r \)-temporality as a composition of the maps from \( (x, p) \) to \( \varphi(x, p) \), and from \( \varphi(x, p) \) to the pair \( (r(p), r(p)) \).

Let \( \varphi:\phi(G) : PO \times \mathcal{P} \rightarrow (r(p), r(p)), \quad p \in \mathcal{P}. \)  

Define \( R : (G, \phi) \rightarrow (G_R, \phi_R) \) to be an generalized retiming if,

- \( \forall P_x \in PO, w_{P_{x,y}} \in \phi \cap \phi_R. \)
- the corresponding outputs in \( G \) and \( G_R \) are latched by elements in the same phase,
- \( r \)-temporality is preserved along all paths from the host vertex to all primary-outputs.

Note, in an generalized retiming we cannot relate to the temporality defined earlier in this section since the number of phases of the clock set has changed. However we can still make use the concept of \( r \)-temporality. If the output latches remain fixed this implies that \( \varphi(x, p) \pmod{\phi_R} \) is the same. Note that this is nothing but \( r(p) \). In theorem 1 above there was no reference to the clocking scheme, except in relating the temporality of the circuit to the tokens (equation 6). If we replace \( \varphi(p, p) \) by \( \varphi_r(p, p) \) and carry out the argument, we see that tokens along all cycles need to be preserved for generalized retiming. We get the following lemma:

**Lemma:** Let \( R : G \rightarrow G_R \) represent an generalized retiming. Assume all primary outputs are latched. The \( r \)-temporality of \( G \) and \( G_R \) is the same if and only if the number of tokens in each cycle is preserved.  

Hence generalized retiming in the multi-phase scenario, implies not only shifting latches but also assigning phases to latches, possibly extending the clock set if necessary. However almost all designers fix the clocking scheme a priori to the design and rarely tolerate any deviations. The main reason for introducing the concept of generalized retiming is that it greatly facilitates the extension to mixed circuits.

### 4 Multi-Phase Mixed Systems

We now extend the algorithm to circuits that have both edge-triggered and level-sensitive memory elements. For every circuit \( G \) with mixed memory elements and a clock set \( \phi \), we shall show that there exists a temporally equivalent circuit \( G_{eq} \) with the same topology, and an extended clock set \( \phi_{eq} \). \( G_{eq} \) consists of edge-triggered elements only.

Let \( \text{retard}(l_i) \) denote the retardation at the latch on edge \( e_i \). If the signal arrives before the active phase, \( \text{retard}(l_i) \) is equal to zero. For all edge-triggered elements the retardation is zero. There are well known algorithms to compute the retardation at a latch (09 [8]).

We now describe the transformation \( \gamma \) to an edge-triggered system without changing the temporality. Replace every level-sensitive element in the circuit by edge-triggered ones. For each latch \( l_i \) with retardation greater than zero, add a phase \( \phi_{l_i} \) to the clock set and use it to latch element \( l_i \).

\[ \gamma = (\tau_{eq} + \text{retard}(l_i)) \pmod{c}. \]  

A phase which is not present in the original clock set but has been added to it due to the procedure above is called a pseudo-phase. We could potentially end up with a clock set with as many phases as the number of latches. In the above process the circuit behavior has not changed. This implies that the \( r \)-temporality is preserved.

![Figure 4: Equivalent Behavior](image-url)
and \( \phi_e \), but the counting of tokens is independent of the sets as long as \( w, \phi_e \in \phi_e \), \( \forall \phi_e \in PO \). We know \( G_{\phi_e} \) is an extended retiming of \( G_{\phi} \). As a result we need to conserve the tokens in all cycles of the circuit with mixed memory elements.

4.1 Temporal Verification

The problem posed is, given two circuits, \( G_1(V_1, E_1, d_1, w_1) \) and \( G_2(V_2, E_2, d_2, w_2) \) with the same clock set \( \phi \) verify if one is a legal retiming of the other. First we need to verify that the two circuits have the same graph structure (modulo the dummy vertices and edges introduced). We assume that the primary inputs and primary outputs have an existing correspondence, i.e., the same names. This is an example of directed graph isomorphism. We restrict the retiming algorithm to preserve the gate names. Once an isomorphism has been established, we proceed to check the \( \tau \)-temporality. Since the graphs are actually multi-graphs we need to take into account the isomorphism of edges when two vertices have more than one edge between them. In such a case the order of latches on the two edges will be different. After a retiming this difference must still be present. So when there is more than one edge connecting two vertices, we use the order of latches on the edges to distinguish them.

The second problem is to verify that \( \tau \)-temporality is maintained. The number of cycles in a circuit could be exponential in the size of the circuit. Hence we face the task of examining all cycles in a graph. However under a restriction on latch placement, we can reduce the problem to a linearly independent cycle set. Assume all primary outputs are latched on the last phase \( \phi_4 \) and all primary inputs are latched in the first phase \( \phi_1 \). Let \( p, \phi_1, \phi_2, \phi_3, \phi_4 \in V \) and \( \phi_1, \phi_2, \phi_3, \phi_4 \in V \) be any path in the graph \( G_1 \) (isomorphic to \( G_2 \)). The restriction requires that the phases of the latches (in the order of traversal along \( p \)) appear in the same order as a cyclic ordering of the clock phases. Under this assumption, if there are \( \tau \) tokens in a cycle, then there are \( \tau \) latches in that cycle. Some of the popular design styles fall in this category.

Proposition 1: Under the restriction described above, the number of tokens in any cycle is preserved if and only if the number of latches is preserved in all cycles.

The number of cycles in a cycle is a property dependent on the cycle edges. Let \( C_{\phi_e} \) represent the cycle edge incidence matrix, i.e., \( C_{\phi_e} \) = 1 if edge \( e_i \) is in cycle \( \phi_e \). Let \( C_{\phi_e} \) represent a linearly independent set of cycles. Any other cycle say \( C_\alpha \) with \( \alpha \) latches can be written as \( C_\alpha = \alpha^T C_{\phi_e} \alpha \in \mathbb{R}^\alpha \), where \( \alpha \) is the rank of \( C_{\phi_e} \) (rank of \( C_{\phi_e} \)). It is easy to see \( w_\alpha = \alpha^T w_\phi \), where \( w_\phi \) is a column vector whose \( \phi^{th} \) entry is the number of latches in the cycle \( C_{\phi_e} \). Hence if \( w_\phi \) is preserved across retiming, the value of \( w_\alpha \) will also be preserved. Intuitively we are finding the largest submatrix of non-zero rank in the matrix \( [C_{\phi_e} w_\phi] \), where \( w_\phi \) is the number of latches in \( C_{\phi_e} \).

5 Implementation

The algorithm has been implemented in SIS, the sequential synthesis system at Berkeley. This provides a means of formally verifying the results of optimization algorithms (like retiming) once they are designed and implemented. The pseudo-code is provided below.

\[\text{GL}, \text{G2} = \text{circuits to be verified.} \]
\[
\text{C} = \text{cycle set}
\]
\[
\text{timing_analyze} (\text{G1})
\]
\[
\text{timing_analyze} (\text{G2})
\]
\[
\text{check_graph_isomorphism} (\text{GL}, \text{G2})
\]
\[
\text{C} = \text{enumerate_cycles} (\text{GL}, \text{G2})
\]
\[
\text{foreach cycle in C}
\]
\[
\text{check tokens}
\]

Note that if we are enumerating all cycles, we check the token count as each cycle is found. So we never have to store all the cycles at any time. Often under restrictive retiming, the "all cycles" problem can be reduced to a set of linearly independent cycles called the "fundamental cycle" set. The latter can be enumerated in polynomial time (complexity of \( O(m^2 \log(n)) \)).

The algorithm is described (without proof) in the appendix. In this case, the temporal verification problem of a retimed circuit with the original circuit has the complexity of the timing verification problem. We present some results on checking if two circuits are equivalent in the table (1). Examples 1 and 2 are taken from [7], 3 and 4 are modified ISCAS benchmarks and 5 is a simple biquadratic filter. The third column specifies the nature of the clocking scheme. The L or E in brackets gives the type of latches (level-sensitive or edge-triggered). The time reported in column four is the total time taken by the algorithm on a DEC 5000, i.e., the time for the graph isomorphism check, the time for doing timing analysis on both circuits, and the time taken for the cycle enumeration. The last column gives the number of cycles in each graph circuit. The numbers in brackets in the fourth column are obtained by using the algorithm described in the appendix to generate a fundamental cycle set. The size of the fundamental cycle set is shown in brackets in the fifth column. The main advantage of the algorithm given in the appendix is only when we need to have the fundamental cycle set explicitly. For the verification problem there is no need to carry the set of cycles throughout the algorithm. Consequently the brute force enumeration is comparable with the algorithm in the appendix for moderately sized circuits.

6 Conclusion

To summarize, we have provided a model and a formal definition for the temporal behavior of a arbitrary multi-phase circuit. We have also described an algorithm to do formal verification of the temporal behavior of circuits.

The authors thank A. Wang for several useful discussions and an anonymous referee for several constructive comments. This research was supported by NSF under contract number EMC-8419744 and DARPA under contract number JPB-190-073. Their support is gratefully acknowledged.

References

We present here an algorithm to compute the fundamental cycle set in a strongly connected directed multi-graph. The circuit graph is represented by an incidence matrix $A$, be defined as

$$\begin{align*}
a_{ij} &= \begin{cases}
1 & \text{vertex } v_i \text{ is the source of edge } e_j \\
-1 & \text{vertex } v_i \text{ is the sink of edge } e_j \\
0 & \text{otherwise}
\end{cases}
\end{align*}$$

Each column of $A$ has only exactly one $+1$ entry and one $-1$ entry. The rank of $A$, denoted by $R(A)$ is $n - 1$ (adding all the rows to the last row we get a zero row vector).

Consider the matrix product $AC^T$. It is easy to show that this product is identically zero.

By Sylvester's Law we have $R(A) + R(C) - m \leq 0$. But $R(A) = n - 1$. Hence we conclude $R(C) \leq m - n + 1$. The rank of $C$ is the number of linearly independent cycles.

**Algorithm**

The algorithm uses a depth first search (DFS) on a directed graph. The host vertex as the starting vertex. The edges are classified as

1. **Tree edge**: Edge that is a part of the DFS tree.
2. **Back edge**: Edge that is directed from the current node during DFS to a node which has been already visited and this node is still on the DFS stack.
3. **Cross edge**: Edge that is directed from the current node during DFS to a node which has been already visited and this node is no longer on the DFS stack.

We proceed as follows. First we shall derive an upper bound on $n - 1 - 1$, where $V$ is the set of vertices and $E$ the set of edges. Let the number of vertices be $n (= |V|)$, and the number of edges be $m (= |E|)$. We use the terms fanin and fanout interchangeably. The terms fanin and fanout are used as in Section 2.

We proceed as follows. First we shall derive an upper bound on the cardinality of the fundamental cycle set. Then the algorithm will be outlined and we shall show that the upper bound is actually achieved. Finally we shall analyze the complexity of the algorithm.

Let $C$ represent the cycle-edge incidence matrix.

$$c_{ij} = \begin{cases}
1 & \text{if edge } e_j \text{ is in cycle } C_i \\
0 & \text{otherwise}
\end{cases}$$

We shall show that the rank of $C$ is $m - n + 1$. Readers familiar with network theory will recognize the correspondence of this proof with the technique for generating a linearly independent set of Kirchhoff Voltage Law equations in a circuit [1]. Let the node incidence matrix $A$, be defined as

$$a_{ij} = \begin{cases}
+1 & \text{vertex } v_i \text{ is the source of edge } e_j \\
-1 & \text{vertex } v_i \text{ is the sink of edge } e_j \\
0 & \text{otherwise}
\end{cases}$$

Each column of $A$ has only exactly one $+1$ entry and one $-1$ entry. The rank of $A$, denoted by $R(A)$ is $n - 1$ (adding all the rows to the last row we get a zero row vector).

Consider the matrix product $AC^T$. It is easy to show that this product is identically zero.

By Sylvester's Law we have $R(A) + R(C) - m \leq 0$. But $R(A) = n - 1$. Hence we conclude $R(C) \leq m - n + 1$. The rank of $C$ is the number of linearly independent cycles.

### Appendix: Computing the Fundamental Cycle Set

We present here an algorithm to compute the fundamental cycle set in a strongly connected directed multi-graph. The circuit graph described in the paper often satisfies this requirement. If it is not we can break it into strongly connected components and verify the behavior of each component. Let the graph be represented by $G(V,E)$, where $V$ is the set of vertices and $E$ the set of edges. Let the number of vertices be $n (= |V|)$, and the number of edges be $m (= |E|)$. We use the terms fanin and fanout interchangeably. The terms fanin and fanout are used as in Section 2.

We proceed as follows. First we shall derive an upper bound on the cardinality of the fundamental cycle set. Then the algorithm will be outlined and we shall show that the upper bound is actually achieved. Finally we shall analyze the complexity of the algorithm.

Let $C$ represent the cycle-edge incidence matrix.