Sequence Compaction to Preserve Transition Frequencies

Ali Pınar
Lawrence Berkeley National Laboratory
and
C. L. Liu
National Tsing Hua University

Simulation-based power estimation is commonly used for its high accuracy despite excessive computation times. Techniques have been proposed to speed it up by compacting an input sequence while preserving its power-consumption characteristics. We propose a novel method to compact a sequence that preserves transition frequencies. We prove the problem is NP-Complete, and propose a graph model to reduce it to that of finding a heaviest weighted trail on a directed graph, along with a heuristic utilizing this model. We also propose using multiple sequences for better accuracy with even shorter sequences. Experiments showed that power dissipation can be estimated with an error of only 2.3%, while simulation times are reduced by 10. Proposed methods effectively preserve transition frequencies and generated solutions that are very close to an optimal. Experiments also showed that multiple sequences granted more accurate results with even shorter sequences.

Categories and Subject Descriptors: B.7.2 [**Hardware**]: Integrated Circuits—*Design Aids*; F.2.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity—*Nonnumerical Algorithms and Problems*; G.2.2 [**Mathematics of Computing**]: Discrete Mathematics—*Graph Theory*; J.6 [**Computer Applications**]: Computer-Aided Engineering

General Terms: Algorithms, Performance

Additional Key Words and Phrases: Sequence Compaction, Power Estimation, Graph Algorithms

1. INTRODUCTION

Growing need for low-power systems requires accurate estimation of power dissipation, which has been the subject of numerous recent research efforts. The proposed techniques can be classified as *static* and *dynamic* [1]. Static techniques use statistical information about input sequences to estimate switching activity. These techniques are computationally efficient, but not accurate. Dynamic techniques explicitly simulate the circuit for a typical vector sequence. These methods can give very accurate results, especially when applied at the circuit level, but they require excessive computation time. Besides, results are highly dependent on the input vector sequence. To alleviate such dependency, lengths of the input sequences should be very long, leading to excessive simulation times. But still, these methods

This work is supported in part by the National Science Foundation (NSF) under grant 1-5-31333 NSF MIP 96 12184. The first author is als o supported by the Director, Office of Science, Division of Mathematical, Information, and Computational Sciences of the U.S. Department of Energy under contract DE-AC03-76SF00098.
Authors' addresses: Ali Pınar Lawrence Berkeley Lab, One Cyclotron Road MS 50F, Berkeley, CA 94720. e-mail:apinar@lbl.gov. C.L. Liu, Dept. of Computer Science, National Tsing Hua University, Hsinchu, Taiwan, ROC. e-mail: liucl@mx.nthu.edu.tw.

are used for their accuracy at least before taping out a chip [2]. For a review of power estimation literature, surveys by Pedram [1; 3] are valuable resources.

For a CMOS circuit, dominant source of power dissipation is dynamic transition current. In combinational circuits, power consumption corresponding to an input vector sequence depends only on the transitions between successive input vectors. Thus if a given vector sequence can be transformed to a shorter one while preserving the transition frequencies, the shorter sequence can be used to estimate power consumption for the original. Some recent works studied this problem for both sequential and combinational circuits. Methods proposed by Tsui *et al.* [2] and Marculescu *et al.* [4] have the disadvantage of generating vectors that are not in the original sequence. Huang *et al.* [5] used a two phase strategy: the first phase derives transition profile of internal signals by a fast power estimator, and the second phase generates a shorter sequence using this profile. Marculescu *et al.* [6] used a Markov model to generate a compact sequence and subsequently proposed a hierarchical model with macro and micro states to model the original sequence [7]. Finally, earlier results of our work were presented in [8]. Methods for sequential circuits have been studied in [9; 10].

This paper investigates combinatorial aspects of compacting a sequence with invariant transition frequencies. We call this problem the *Sequence Compaction* problem. We prove the problem is NP-Complete, thus a practical solution requires heuristics. We also propose a graph model to reduce the sequence compaction problem to that of finding a heaviest weighted trail on a directed graph, along with a heuristic to find such trails. In our model, each distinct vector in the input sequence defines a vertex, and each transition is represented by a number of parallel edges. The number of parallel edges and the weight of each edge is defined by how many times the transition appears in the original sequence. We also discuss generating multiple sequences with different compaction factors, as opposed to merely a single sequence, for better accuracy with even shorter sequences.

Proposed techniques have been applied to MCNC91 circuits [11], using SPICE for simulations. Experiments verify that simulation times can be significantly reduced with highly accurate results. Error in estimations is limited to only 2.3%, while the simulations are 10 times faster. We also investigated the *reliability* of compacted sequences. The objective of compaction is to avoid simulating long sequences, so accuracy of an estimation is not verifiable. A compacted sequence is reliable, if it preserves all transition frequencies. A compacted sequence might still give a good estimation, albeit unpreserved transition frequencies, but such a sequence is not reliable. Our experiments verify that generated sequences preserve the transition frequencies with great accuracy, thus are reliable. Experiments also show that solutions are very close to optimal, and scale well with higher compaction factors and longer sequences. Experiments also confirmed that multiple sequences grant better accuracy with even shorter simulation times.

The next section defines the problem, and proves its NP-Completeness. A graph model to reduce the problem to that of finding a heaviest weighted trail in a directed graph, and a heuristic based on this model are the subject of § 3. Using multiple sequences for power estimation is discussed in § 4. Experimental results are presented in § 5, followed by concluding remarks and future work in § 6.

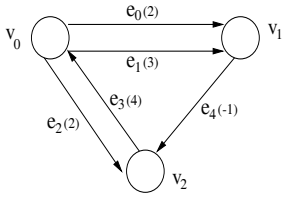


Fig. 1. Graph $G = (V, E)$, where $V = \{v_0, v_1, v_2\}$ and $E = \{e_0, e_1, e_2, e_3, e_4\}$. Numbers in parenthesis indicate edge weights.

2. SEQUENCE COMPACTION PROBLEM

This section provides background information on graph theory, presents a formal definition of the sequence compaction problem and analyzes its complexity.

2.1 Graph Theory Terminology

A graph $G = (V, E)$ is defined by two sets: a finite set V of elements called *vertices* and a finite set E of *edges*. Each edge is identified with a pair of vertices. If the edges of the graph are identified with ordered pairs, then the graph is called a *directed* graph, otherwise it is called an *undirected* graph. We will be working on directed graphs in this paper. Fig. 1 illustrates a graph. Edges that have the same end vertices and the same direction are called *parallel* edges (e.g., e_0 and e_1 in Fig. 1). A *trail* in a graph $G = (V, E)$ is a sequence of vertices $v_0, v_1, v_2, \ldots v_k$ such that $(v_i, v_{i+1}) \in E$ for $0 \leq i < k$, and each edge in E occurs at most once in a trail. In Fig. 1, v_0, v_1, v_2, v_0, v_1 is a trail (parallel edges can be used separately), however v_0, v_2, v_1 is not, because there are no edges from v_2 to v_1. Weights can be assigned to edges of a graph. Then, the *weight of a trail* is the sum of weights of its edges. In this example, the weight of trail v_0, v_1, v_2, v_0, v_1 is $2 - 1 + 4 + 3 = 8$. A *path* is a trail where each vertex occurs at most once. A *Hamiltonian* path in G is a path visiting all vertices. In Fig. 1, v_0, v_1, v_2, v_0 is not a path since v_0 is repeated. v_0, v_1, v_2 is a Hamiltonian path, since it is a path that visits all vertices.

2.2 Problem Formulation

Sequence compaction aims at transforming a sequence to a shorter one with similar characteristics. Here, we work on sequences of binary input vectors of a circuit and want to preserve power dissipation characteristics. A compacted sequence preserves power dissipation characteristics of the original sequence, if it "moves" all the internal states of the circuit with the same frequency as the original. The major source of power dissipation in a circuit is the switching activity, which is triggered by transitions in the input vectors. If a sequence can be compacted into a shorter one with the same vector transition frequencies, then the compacted sequence can be used to estimate power dissipation of the original sequence. Preserving vector-transition frequencies is not the only way to preserve power characteristics. Techniques have been proposed to preserve bit transitions instead of vector transitions, where compacted sequences might include new vectors and transitions that are not in the original sequence. Preserving vector transitions is better for sequences of fewer vectors with a lot of repetition (e.g., control-code driven circuits), whereas preserving bit-transition frequencies is better for sequences with a limited repetition of vectors. Alternatively, vectors can be decomposed into highly correlating bit groups, and transition frequencies for these groups can be preserved [4]. Correlating bits can be found by graph partitioning, which can be avoided with the Markov model [6].

Accuracy of compaction requires preserving the transition probabilities of all transitions. In [13], Marculescu *et al.* prove that satisfying the condition

$$|p(t) - p'(t)| < \epsilon \tag{1}$$

for all transitions, limits the error in estimation to $O(\epsilon)$, where ϵ is an infinitesimal quantity and $p(t)$ and $p'(t)$ denote the transition probabilities of transition t in the original and compacted sequences, respectively. Here we use a slightly different objective function. Formally, an input sequence $S = \langle s_1, s_2, \ldots, s_i, s_j, \ldots s_m \rangle$ is a sequence of binary n-vectors. A *transition* $t = (s_i, s_j)$ is an ordered pair of distinct n-vectors. We will use $S(t)$ to denote the number of transitions t, and $T(S)$ to denote the set of transitions in sequence S. Based on these definitions, we define the sequence compaction problem as follows.

Given a compaction factor c, and an input sequence $S = \langle s_1, s_2, \ldots, s_m \rangle$; construct a new sequence S' to minimize cost $\mathcal{C}(S, S', c)$, where

$$\mathcal{C}(S, S', c) = \sum_{t \in T(S)} \frac{|S(t) - c * S'(t)|}{S(t)}. \tag{2}$$

This formula aims at preserving frequencies of all transitions individually, as advised in Eq. 1, and gives a more tractable function for a combinatorial problem. Ideally, the cost of the compacted sequence will be zero (i.e., if a transition t appears $S(t)$ times in the original sequence it will appear $S'(t) = S(t)/c$ times in the compacted sequence), however this is not always possible, and the cost function punishes any deviation in preserving the frequency of a transition. We normalize the difference in estimation to preserve frequency of each transition, since the power dissipation for transitions can be quite diverse. It is worth noting that techniques to be described can be used with alternative cost functions, as discussed in § 3.3.

Accuracy $\mathcal{A}(S, S', c)$ of a solution S' can be defined as $\mathcal{A}(S, S', c) = |T(S)| - \mathcal{C}(S, S', c)$, where $|T(S)|$ denotes the cardinality of set $T(S)$. We will use the cost metric for most of our discussions, and refer to accuracy occasionally for simplicity of presentation. Notice that minimizing cost is equivalent to maximizing accuracy, since $|T(S)|$ is a constant.

Our formulation does not involve an explicit constraint on the length of the compacted sequence. Compaction factor provides an inherent constraint on length, and we would expect the compacted sequence to be c times shorter than the original. This formulation gives flexibility on the length of the compacted sequence for more accurate solutions.

Example 1: Let $S = \langle ABCABCABCABC \rangle$ and $c = 4$. Here, A, B and C represent binary primary input vectors of the circuit. (A, B), (B, C) and (C, A) are transitions in this sequence. $S((A, B)) = 4$, because there are four (A, B) transitions in S. Similarly, $S((B, C)) = 4$, and $S((C, A)) = 3$. Let $S' = \langle ABCA \rangle$ be a compacted sequence. The cost $\mathcal{C}(S, S', c)$ of this sequence is: $\frac{|S((A,B)) - c*S'((A,B))|}{S((A,B))} + \frac{|S((B,C)) - c*S'((B,C))|}{S((B,C))} + \frac{|S((C,A)) - c*S'((C,A))|}{S((C,A))} = \frac{|4-4*1|}{4} + \frac{|4-4*1|}{4} + \frac{|3-4*1|}{3} = \frac{1}{3}$.

One way to estimate power for a sequence is to measure the power dissipation for each distinct transition, which is impractical due to the enormous number of measurements. If the simulator reports cycle-by-cycle power dissipation however, all

transitions can be concatenated to a single sequence, so that power for a transition is the change in total power dissipation in the corresponding cycle. Straight-forward concatenation of transitions generates a sequence of length twice the number of distinct transitions, however a clever ordering of the transitions yields a shorter sequence. For instance, for transitions (A, B) and (C, A) we can use sequence CAB instead of $ABCA$. A shortest sequence that covers all transitions at least once is required, which corresponds to finding a (variant of) Chinese Postman tour [12] in the graph where there is a vertex for each vector, and an edge for each transition. In this work, we work on generating one compact sequence whose total power dissipation will be used to estimate the power dissipation of the original, which has also been the subject of related work [2; 4; 5; 7; 9; 10; 13; 14].

A similar problem has been studied to generate representative instruction traces for architecture-level performance analysis [15; 16]. Iyengar *et al.* try to generate shorter instruction sequences that preserve cache utilization and branch prediction performance. Instructions are grouped to *fully-qualified basic blocks* (instruction blocks that give the same performance whenever executed), and preserve the frequencies of these blocks. The two problems are related in the broad sense (both try to generate shorter sequences preserving some characteristic), but differ in combinatorics. In Iyengar *et al.* preserve the frequencies of the entities, whereas we preserve the transition frequencies between the entities.

2.3 Complexity of the Sequence Compaction Problem

This section proves that the sequence compaction problem is NP-Complete. Before the proof, we restate the problem as a decision problem, which we will refer to in this section: *Given a compaction factor c, an input sequence $S = \langle s_1, s_2, \ldots, s_m \rangle$, and a bound B; decide if there exists a sequence S' so that $\mathcal{C}(S, S', c) \leq B$.*

The Hamiltonian path problem [17] will be reduced to the sequence compaction problem for the proof. First, we will construct a sequence S for a given directed graph, where vertices and edges will be represented by subsequences, then try to compact this sequence. A compacted sequence with a small enough cost will indicate the existence of a Hamiltonian path in the graph.

Theorem 1: *The Sequence Compaction problem is NP-Complete.*

Proof: Let $G = (V, E)$ be a directed graph for which a Hamiltonian path is being sought. Each vertex and edge will be represented by a unique subsequence, which is repeated to make room for compaction. These subsequences will be concatenated into a single sequence by putting separators between every vertex and edge subsequence. The path described by a compacted sequence can be identified by replacing a vertex subsequence with the corresponding vertex. Here is the construction: Let $M, N \geq 4$ be arbitrarily large and even integers.

(1) Corresponding to each vertex $v_i \in V$, construct subsequence $v_{i1}, v_{i2}, \ldots, v_{iN}$, repeat the subsequence for M times, and add x_i at the end. Thus, sequence for vertex v_i will be $v_{i1}, v_{i2}, \ldots, v_{iN}, v_{i1}, \ldots v_{iN}, \ldots v_{iN}, x_i$.

(2) Corresponding to each edge $(v_i, v_j) \in E$, construct subsequence v_{iN}, v_{j1}, e_{ijk}. Repeat the subsequence for $M/2$ times, each time with a different k value. Thus sequence for edge (v_i, v_j) will be $v_{iN}, v_{j1}, e_{ij1}, v_{iN}, v_{j1}, e_{ij2}, \ldots, v_{iN}, v_{j1}, e_{ijM/2}$.

(3) S is a concatenation of all these subsequences followed by a sentinel y.

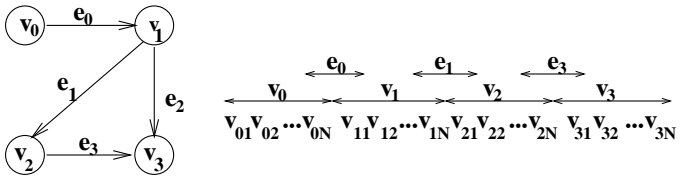


Fig. 2. Compacted sequence describing a Hamiltonian path. Graph has a Hamiltonian path $v_0 \stackrel{e_0}{\rightarrow} v_1 \stackrel{e_1}{\rightarrow} v_2 \stackrel{e_3}{\rightarrow} v_3$. Corresponding compacted sequence is presented on the right.

In this construction, dummy variables: x_i, e_{ijk}, separate sequences corresponding to vertex and edge transitions, enabling encoding a set as a sequence. A subsequence like $v_{i1}, v_{i2}, \ldots, v_{iN}$ corresponds to vertex v_i in the path, and a transition v_{iN}, v_{j1} corresponds to edge (v_i, v_j) . We will try to compact this sequence by a factor of M. Proof is based on the following remarks:
(1) Any transition that is not in the original sequence cannot be in the compacted sequence, since the cost of such a solution as described in Eq. 2 is ∞. This ensures that v_{i1} can only be followed by v_{i2}, or generally, an input v_{ij} for $1 \leq j < N$ and $v_i \in V$ can only be followed by v_{ij+1}. In other words, subsequence representing a vertex either appears as a whole, or does not appear at all.
(2) If a vertex transition $v_{i1}, v_{i2}, \ldots, v_{iN}$ occurs just once, penalty will be zero. Each deviation from one will add a cost of $N-1$. Absence of $v_{iN}v_{i1}$ transitions will cost 1. Minimum cost due to vertex subsequences is $|V|$, achieved when each vertex sequence occurs exactly once.
(3) For each edge transition v_{iN}, v_{j1}, zero or one appearance cost the same: 1. After the first appearance, each extra appearance costs 2. Thus minimum total cost for edges is $|E|$, achieved when each edge transition occurs at most once.
(4) Dummy variables e_{ijk} and x_i appear in two transitions, and if these variables do not appear in the output, cost due to these transitions will be 2. If these variables occur in the output, associated costs will be $2 * (M-1)$. Minimum total cost for these variables will be $M * |E|$ and $2 * |V|$, respectively.

These remarks are valid albeit the order vertex and edge transitions are concatenated in sequence S, thanks to the separators at the ends of vertex and edge subsequences. Remark 2 implies that the cost of S' is minimum when it includes all vertex subsequences exactly once. Moreover these subsequences must be connected by edge transitions, because by Remark 1, the compacted sequence does not have any transitions that are not in the original sequence, and a transition (v_{iN}, v_{j1}) occur in the sequence only if there is an edge from v_i to v_j in the graph. Also by the first remark, v_{j1} is followed by the vertex subsequence $v_{j2}, \ldots v_{jN}$, that is there are only vertex subsequences connected by edge transitions in the compacted sequence. Thus S' will consist of all the vertex subsequences with one edge transition in between. This is an ordering of vertices with edges connecting them, thus a Hamiltonian path. A compact sequence describing a Hamiltonian path is illustrated in Fig. 2. Minimum total cost is $3 * |V| + (M+1) * |E|$, which is achieved when solution describes a Hamiltonian path. Any other solution has a higher cost, thus we can conclude that graph G has a Hamiltonian path if and only if cost of the compacted sequence is equal to $3 * |V| + (M+1) * |E|$.

Notice that the cost of a solution for the compaction problem can be verified in polynomial time, thus the sequence compaction problem is NP-Complete. □

3. SOLVING THE SEQUENCE COMPACTION PROBLEM

3.1 Graph Model

This section describes our graph model for an instance of a sequence compaction problem, where a heaviest weighted trail describes an optimal solution for the compaction problem. In this model, each distinct vector in the input sequence is represented by a vertex, and each transition is represented by multiple weighted directed edges. Equivalence between a heaviest weighted trail and an optimally compacted sequence is established by the way edge weights are assigned. Remember that edges of the graph correspond to transitions of the sequence, and if an edge occurs once more on the trail, corresponding transition will occur once more in the compacted sequence. Thus, edge weights can be computed as the decrease in $\mathcal{C}(S, S', c)$ when the corresponding transition appears in the compacted sequence. In this case, increase in the weight of a trail when an edge is added will be equal to the decrease in cost of the compacted sequence when the corresponding transition is added to the sequence. Greater the sum of edge weights of edges is, less the cost of the compacted sequence will be. More specifically, each transition t_i in S, will be represented by several edges $e_{i1}, e_{i2}, \ldots e_{ij}, \ldots$ with $w(e_{i1}) \geq w(e_{i2}) \geq \ldots \geq w(e_{ij}) \geq \ldots$. If an optimal solution covers j copies of edge e_i, then there is an optimal solution that uses the first j edges: $e_{i1}, e_{i2}, \ldots, e_{ij}$, because edge weights are nonincreasing and a heaviest weighted trail is sought. Thus, the weight of the jth edge can be set as the change in cost, if transition t_i is added once more to S', which already has $j-1$ copies of transition t_i.

$$w(e_{ij}) = \frac{|S(t_i) - c*(j-1)| - |S(t_i) - c*j|}{S(t_i)} = \begin{cases} \frac{c}{S(t_i)} & \text{if } S(t_i) \geq c*j \\ \frac{-c}{S(t_i)} & \text{if } S(t_i) \leq c*(j-1) \\ \frac{2(S(t)\%c)-c}{S(t_i)} & \text{otherwise} \end{cases}$$

where % represents the modulo operation.

A graph $G = (V, E)$ that represents the sequence compaction problem for input sequence $S = \langle s_1, s_2, \ldots, s_i, \ldots, s_m \rangle$ and compaction factor c, has a vertex v_i for each input vector s_i in S. Each transition $t = (s_i, s_j)$ in S is represented by multiple copies of edge (v_i, v_j):

(1) $\lfloor \frac{S(t)}{c} \rfloor$ copies with weight $\frac{c}{S(t)}$

(2) one copy with weight $\frac{2(S(t)\%c)-c}{S(t)}$, if $\lfloor \frac{S(t)}{c} \rfloor \neq \frac{S(t)}{c}$

(3) sufficiently many copies with weight $\frac{-c}{S(t)}$.[1]

Edges generated by the first rule correspond to transitions that will always reduce $\mathcal{C}(S, S', c)$, whereas edges generated by the last rule always increase the cost, to penalize overestimation. The second rule corresponds to approximating $\frac{S(t)}{c}$ by $\lfloor \frac{S(t)}{c} \rfloor$ or $\lceil \frac{S(t)}{c} \rceil$. This formulation might produce excessive number of edges, disabling a

[1] An optimal solution might employ negative weight edges for subsequent positive weight edges. The number of negative weight edges between two vertices should be less than that of positive weight edges in the graph, since a sequence of nonrepeated negative weight edges should precede at least one positive weight edge. Maximum number of positive weight edges in the graph is $\frac{2*|S|}{c+1}$, which occurs when each transition occurs $\frac{c+1}{2}$ times.

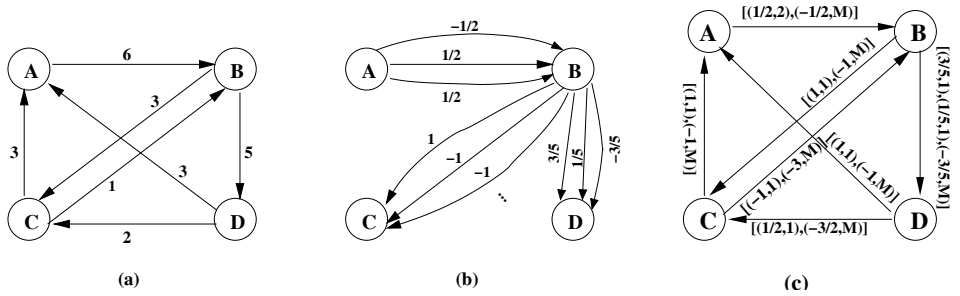


Fig. 3. (a) Number of transitions for the sequence in Example 2. (b) A subset of the edges. (c) graph after edges are packed, where edges are labeled as [(weight of the edge, multiplicity), ...]. e.g., From B to D, there is 1 edge with weight $3/5$, 1 edge with weight $1/5$, and M edges with weight $-3/5$, where $M=9$ is total number of positive weight edges.

practical solution. However, parallel edges with identical weight can be packed into one edge with *multiplicity* to reduce the problem size. Notice that there are at most three different weights for each edge. Thus, the size of the graph is determined by the number of distinct input vectors, which determines the number of vertices, and the number of distinct transitions, which determines the number of edges.

Theorem 2: *A heaviest weighted trail in the graph for a sequence S corresponds to an optimal solution, and weight of the trail is equal to accuracy of the solution for the sequence compaction problem.*
Proof: Proof comes from the way weights are assigned to edges. □

Example 2: Let $S = \langle BDABCABDABDCABDABCBDCABC\rangle$ and $c = 3$. Fig. 3 (a) illustrates number of transitions in this sequence. There are 5 transitions from B to D. Weight of the first $B \rightarrow D$ edge can be calculated as the difference in the cost of zero and one appearance, thus $w(e_{BD1}) = \frac{|5-3*0|-|5-3*1|}{5} = \frac{3}{5}$. Weight of the second edge is difference between underestimating and overestimating transition BD, and can be computed as $w(e_{BD2}) = \frac{|5-3*1|-|5-3*2|}{5} = \frac{1}{5}$. Any other edge will cost $-3/5$. Fig. 3 (b) illustrates some of these edges with their weights. Graph after packing parallel edges with the same weight into edges with multiplicity is depicted in Fig. 3(c). In this graph, maximum weighted trail is $B \rightarrow D \rightarrow A \rightarrow B \rightarrow C \rightarrow A \rightarrow B \rightarrow D \rightarrow C$. Weight of this trail is $\frac{3}{5}+\frac{3}{3}+\frac{1}{2}+\frac{3}{3}+\frac{3}{3}+\frac{1}{2}+\frac{1}{5}+\frac{1}{2} = \frac{159}{30}$. Corresponding compact sequence is $S' = \langle BDABCABDC\rangle$. Cost of this solution is $\mathcal{C}(S,S',3) = \frac{|S(A,B)-3*S'(A,B)|}{S(A,B)} + \frac{|S(B,C)-3*S'(B,C)|}{S(B,C)} + \ldots = \frac{0}{6}+\frac{0}{3}+\frac{1}{5}+\frac{0}{3}+\frac{1}{1}+\frac{0}{3}+\frac{1}{2} = \frac{17}{10}$. Note that accuracy of the solution $\mathcal{A}(S,S') = |T(S)| - \mathcal{C}(S,S') = 7 - \frac{51}{30} = \frac{159}{30}$ is equal to the weight of the trail.

3.2 Finding a Heaviest Weighted Trail

Although finding a heaviest weighted trail is NP-Complete, there are polynomial-time algorithms for special cases. Our heuristic removes positive weight cycles to achieve a graph, free of positive weight cycles where a heaviest weighted trail is easy to find. The heuristic has three steps: (1) detecting and removing positive weight cycles (2) finding a heaviest weighted trail in the reduced graph (3) improving solution quality by adding the cycles back to this trail.

Positive weight cycles in a graph can be detected by applying Bellman-Ford algorithm [17] repeatedly. There is not a unique solution for removing positive

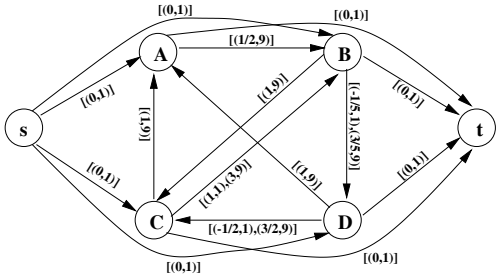


Fig. 4. Augmented graph for Example 3. Edge weights are labeled as [(weight of edge, multiplicity),...]. e.g., there is one edge with weight $\frac{1}{5}$ and 9 edges with weight $\frac{-3}{5}$ from B to D.

weight cycles of a graph. Different algorithms as well as different implementations of the same algorithm will give different solutions. This might have a minor effect on the quality of solutions produced by our heuristic. Once positive weight cycles are removed, a heaviest weighted trail in the reduced graph becomes a path, which can be identified using the Bellman-Ford algorithm, after the graph is augmented with a source vertex s, a terminal vertex t and edges from s to every other vertex and from every other vertex to t. The costs for these edges are zero, and the multiplicities are one. A heaviest weighted path from s to t in this augmented graph will be a heaviest weighted trail in the original graph. This trail can be improved by inserting positive weight cycles removed at the first step. If a vertex v in the cycle is on the trail, the cycle can be inserted to the trail by opening it at vertex v to make a trail with v at both ends, and replacing vertex v of the original trail with this trail.

Example 3: Let $S = \langle BDABCABDABDCABDABCBDCABC\rangle$, and $c=3$ as in Example 2. Graph for this sequence is depicted in Fig. 3(c). Positive weight cycles in this graph are $B \rightarrow D \rightarrow A \rightarrow B$ and $B \rightarrow C \rightarrow A \rightarrow B$. Graph after removing these cycles and negating edge weights is presented in Fig. 4. Shortest path from s to t follows $s \rightarrow B \rightarrow D \rightarrow C \rightarrow t$. Cycles and the trail have B in common. Removing source and sink vertices and inserting cycle $B \rightarrow C \rightarrow A \rightarrow B$, we get trail $B \rightarrow C \rightarrow A \rightarrow B \rightarrow D \rightarrow C$. Inserting the other cycle $B \rightarrow D \rightarrow A \rightarrow B$ on the first B, we get the trail $B \rightarrow D \rightarrow A \rightarrow B \rightarrow C \rightarrow A \rightarrow B \rightarrow D \rightarrow C$ and the corresponding compact sequence is $S' = \langle BDABCABDC\rangle$.

3.3 Extensions of the Model

The proposed graph model introduces a novel method to construct compacted sequences. The model has flexibility to support different heuristics to give better performance, or to handle sequences with different characteristics, or to be used in conjunction with other methods in the literature such as hierarchical models. Different heuristics can be employed to find a trail after reducing the compaction problem with the proposed graph model. The heuristic described in the previous section is designed for and is effective on unimodal sequences, however it may not be as effective on multi-modal sequences [14]. For this type of sequences, we can design new heuristics. One possible problem with the current heuristic is that it may not be possible to add cycles from a different mode of operation to the trail, which is not likely for unimodal sequences. To avoid leaving cycles out, we can add

a path to the end of the trail to reach one of the vertices in the cycle to add this cycle to the trail to cover all modes of operation.

The techniques proposed can also be used in a hierarchical way. Marculescu *et al.* [7] proposed modeling the input sequence with *macro* and *micro* states to handle multi-modal sequences. Macro states correspond to different modes of operation of the circuit, and micro states correspond to operation of the circuit within a certain mode. The compacted sequence is generated at two steps. First a sequence of macro states is generated, and then each macro state in the sequence is replaced with a compacted sequence for the corresponding operation mode. Our techniques can easily fit into the hierarchical model by being employed to generate macro- and micro-state sequences.

Our methods can easily support alternative cost functions as well. One might want to work on slight variations of our cost function, or transitions might be weighted depending on some estimations on their power consumption. For any cost function, edge weights will still be computed as the difference in the cost function if one more edge appears on the trail, and an optimal solution is still defined by a heaviest weighted trail.

4. CONSTRUCTING MULTIPLE SEQUENCES

Discussions so far focused on generating a single sequence to preserve the transition frequencies of the original sequence. Alternatively, multiple sequences can be generated to collectively preserve transition frequencies, which might help both for better accuracy and shorter sequences. Compacted sequence does not always preserve frequencies of all transitions, either because of the imperfectness of the heuristics or difficulty of the problem itself. Accuracy can be increased by generating a second sequence to cover the underestimated transitions in the first sequence. Note that the second sequence is influenced by the first one, it tries to make up for the shortcomings of the first sequence, thus the two sequences collectively preserve the transition frequencies. With the robustness that the sequence provides, we can use the first sequence more aggressively to generate sequences with higher compaction factors. Consider the sequence $S = \langle ABCDABCDABCDABCDACBEACBEA \rangle$, where the $ABCDA$ subsequence is repeated 4 times, whereas the $ACBEA$ subsequence is repeated only twice. If we generate a single sequence for $c = 2$, then the compact sequence will be $S' = \langle ABCDABCDACBEA \rangle$. However, we can generate two sequences $\langle ABCDA \rangle$ and $\langle ACBEA \rangle$ with respective compactions factors of 4 and 2, to achieve the same accuracy with a shorter total sequence length.

Here is the formal definition of this problem: *Given an input sequence S, and compaction factors $c_1 \geq c_2 \geq \ldots \geq c_i \geq c_k$, construct sequences $S'_1, S'_2, \ldots S'_i, , \ldots, S'_k$ to minimize*

$$\sum_{t \in T(S)} \frac{|S(t) - \sum_{i=1}^{k} c_i * S'_i(t)|}{S(t)}.$$

Given the compacted sequences, power consumption of the original sequence $P(S)$ can be estimated as the weighted average of power consumptions of compacted sequences, i.e., $P(S) = |S| \sum_{i=1}^{k} P(S_i) * c_i / \sum_{i=1}^{k} c_i * |S_i|$. Note that compacted sequences collectively preserve transition frequencies as opposed to all of them independently trying to achieve the same goal.

Table I. Circuit properties

Circuit	Gates	Inputs	Outputs	Circuit	Gates	Inputs	Outputs
i3	90	132	6	cordic	102	23	2
C432	160	36	7	C880	202	41	32
C1355	546	41	32	C1908	880	33	25
C3540	1669	50	22	C6288	2406	32	32

Our graph model and heuristic can be employed for this problem. First, sequence S'_1 for c_1 is constructed, and then before constructing S'_2, edge weights are recomputed considering the transitions already contained in S'_1. While constructing the graph, in which the trail for the ith sequence S_i will be sought, weight of an edge is computed as the change in the objective function, when one more copy of the corresponding transition occurs in S_i. Transitions already covered by preceding sequences $S'_1, S'_2, \ldots, S'_{i-1}$, are also taken into account. Let $P_i(t)$ denote the number of times transition t is covered in the first $i-1$ compacted sequences; $P_i(t) = \sum_{j=1}^{i-1} S_j(t) * c_j$. Remember that $S_j(t)$ denotes how many times transition t appears in the sequence S_j. Weight of the jth edge $w(e^i_j)$ corresponding to transition t for constructing the ith sequence is computed as $w(e^i_j) = \frac{|S(t)-P_i(t)-c_i*(j-1)|-|S(t)-P_i(t)-c_i*j|}{S(t)}$. This scheme assumes subsequent compacted sequences $S_{i+1}, \ldots S_k$, to be empty while computing edge weights, which might cause sequences with large compaction factors to greedily overestimate some transitions. For instance, assume we have 8 copies of a transition in the original sequence, and compaction factors for the two sequences to be generated are 5 and 3. It seems to be a better choice for the first sequence to include 2 copies of this transition with cost of 2/8 than just one copy with cost of 3/8. However, the second sequence can cover the remaining copies of the transition resulting in zero cost, if the first sequence includes only one copy. Another problem arises when the first sequences use negative weight edges to add more positive weight edges to the trail. But again these positive weight edges might be covered by subsequent sequences. To hinder the aggressiveness of early sequences, it is helpful to construct sequences with high compaction factors using only positive weight edges. More specifically, satisfying condition $S(t) \geq \sum_{i=1}^{j} c_i * S'_i(t)$ (always underestimating a transition) for the first few sequences is more effective.

5. EXPERIMENTAL RESULTS

We start with describing our experimental setup and proceed with thorough analysis of proposed techniques for single- and multiple-sequence generation. Three metrics are employed to evaluate the proposed techniques: (*i*) accuracy of estimations for circuit simulations (*ii*) reliability: preservation of transition frequencies (*iii*) closeness to optimal solutions. For circuit simulations, we worked with SPICE for maximum accuracy on 8 MCNC91 circuits [11] in Table I, and 3 compaction factors: 3, 5, and 10. We measured power consumption of circuits for six biased sequences of length 2000. Each sequence mimics to cover multiple macro-states of the circuit, i.e., sequence starts with a group of frequently repeated vectors, then moves to a different state with a different group of vectors (not necessarily distinct), and proceeds. This provides enough repetition to enable compaction, yet yields nontrivial test instances. In terms of bit-level switching activity, sequences can be grouped into three with two in each group.

The sequences were compacted using simple random sampling (RS) [14] using each transition as a unit, Markov model (MM) [6] and proposed heaviest weighted trail method (HWT). Table II presents average accuracies, calculated

Table II. Misspredict ion percentages for SPICE simulations

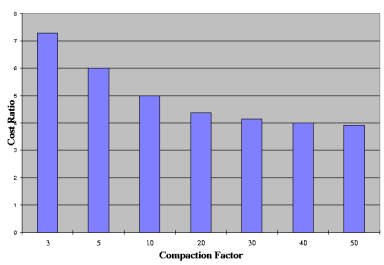
Circuit	$c = 3$			$c = 5$			$c = 10$		
Name	RS	MM	HWT	RS	MM	HWT	RS	MM	HWT
i3	3.3	2.4	0.6	4.7	2.3	0.7	6.2	1.7	1.2
cordic	4.4	4.3	1.3	5.4	4.9	3.0	7.1	4.8	3.2
C432	2.9	3.0	0.8	4.1	2.7	0.6	5.9	3.1	1.1
C880	5.2	5.1	0.8	6.7	5.3	1.9	9.2	5.8	2.2
C1355	6.1	8.6	1.5	6.9	9.4	2.3	8.2	9.4	2.7
C1908	5.2	3.3	1.6	6.6	4.2	2.5	8.5	5.3	2.9
C3540	4.8	3.8	1.1	6.7	4.6	1.7	8.4	5.1	2.0
C6288	5.9	6.5	1.4	7.2	8.4	2.6	9.5	8.6	3.2
Average	4.7	4.6	1.1	6.0	5.2	1.9	8.9	5.5	2.3

Table III. Misspredict ion percentages for sequences with different bit-level switching activity

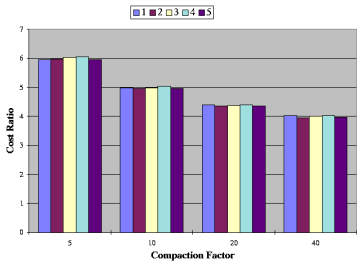
Circuit	Low		Medium		High	
Name	MM	HWT	MM	HWT	MM	HWT
i3	2.1	0.7	2.2	0.7	2.5	0.8
cordic	4.6	2.8	4.8	3.1	5.2	3.1
C432	2.7	0.5	2.6	0.7	2.9	0.7
C880	5.1	1.8	5.2	1.9	5.6	2.1
C1355	8.2	2.1	9.3	2.4	10.7	2.5
C1908	3.8	2.5	4.2	2.5	4.7	2.6
C3540	4.2	1.6	4.6	1.6	5.0	1.8
C6288	8.2	2.5	8.3	2.6	8.8	2.8
Averages	4.9	1.8	5.2	1.9	5.7	2.1

as: $\frac{|AP(S)-AP(S')|}{AP(S)} * 100$, where $AP(S)$ is the average power dissipation for sequence S. The difference between RS and other two columns verify the importance of tracing transition frequencies. We also see that HWT can predict original power consumption very accurately. The ratio of costs of the solutions generated by MM and HWT in these experiments are 3.6, 3.1 and 2.8, for $c = 3, 5$ and 10, respectively, on average. This shows how improvement in preserving transition frequencies translates to accuracy in estimations. Differences between original and estimated values for HWT are negligible. Table III presents missaccuracies in estimations for $c = 5$ on three sequence types with different bit-switching activities. Tracing transition frequencies becomes more crucial as the variance in power dissipation among transitions increases. As expected, performances of both methods MM and HWT degrade as switching-activity variance increases, but performance of HWT is more steady due to better preserved transition frequencies.

Compaction methods are proposed to avoid simulating long sequences, thus accuracy of an estimation cannot be verified. In a compacted sequence, some transitions may be overestimated while some others are underestimated, and errors can cancel each other to give an accurate estimation. Such a compacted sequence is definitely not reliable. A reliable solution estimates each transition accurately, which directly implies accuracy in estimations. We compared the reliabilities of HWT and MM solutions for 750 sequences of length 20000 and 7 compaction factors: 3, 5, 10, 20, 30, 40 and 50. The sequences can be grouped into five according to the number of transitions that dominate the sequence. In group 1, the most frequent 10% transitions in the transition set constitute 80% of the sequence. In groups 2, 3, 4 and 5, respectively, the most frequent 20%, 30%, 40% and 50% transitions constitute 80% of the sequence. As seen in Fig. 5, there is a drastic difference in terms of cost values, thus preserving transition frequencies between the two methods. For

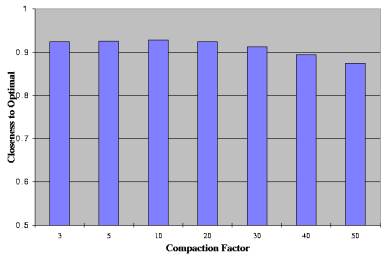


(a) Average

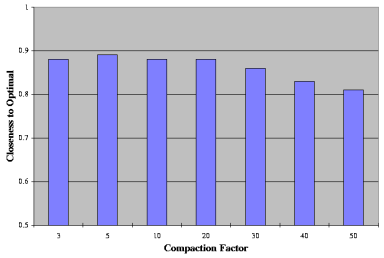


(b) Breakdown for 5 sequence types

sol1



(a) Average

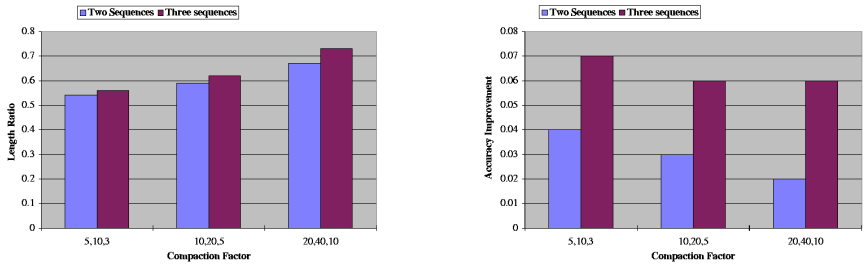


(b) Worst

Fig. 6. Closeness to an upper bound on an optimal solution

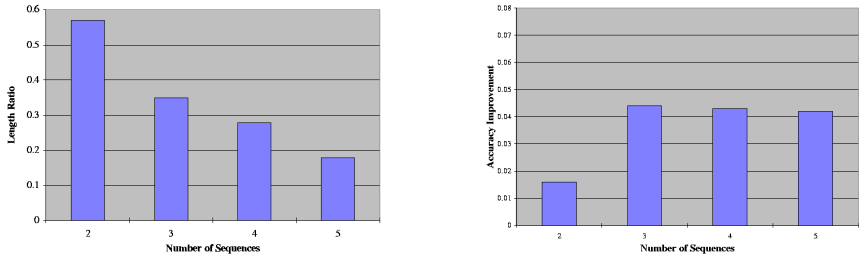
$c = 3$, cost of an MM solution is more than 7 times bigger than that of HWT (cost ratios for sequences in SPICE simulations are smaller, due to shorter length). The ratio decreases as compaction factor increases, and performances virtually get closer. However, it should be noted that cost of an optimal solution increases with increasing compaction factor, as well. We can split the cost of a heuristic solution into two as cost of an optimal solution and cost due to imperfectness of the heuristic. In this case, the rapid increase in the first component overshadows the second. Thus the closing gap should be attributed to increasing cost of an optimal solution. But still, cost ratio is 4 for $c = 40$, showing that HWT solutions are remarkably better than MM solutions in preserving transition frequencies. We did not observe any sensitivity on the sequence type of relative performances of two methods.

Next set of experiments investigates closeness to optimality. Since the value of an optimal solution is not known, an upper bound on the accuracy of an optimal solution, which we call accuracy $\mathcal{A}^*$ of an *ideal solution*, is used. In an ideal solution each transition is estimated in the most accurate way, i.e., a transition t that appears $S(t)$ times in original sequence S, should appear $round(\frac{S(t)}{c})$ times in S', where *round* maps its input to the nearest integer. This is only a bound on the optimal value, because such a sequence does not necessarily exist. Notice that accuracy of an ideal solution $\mathcal{A}^*$ is equal to the total positive weight edges in the associated graph. Fig. 6 presents results for compacting 750 sequences of length 20000. Numbers in these figures are calculated as: $\frac{\mathcal{A}'}{\mathcal{A}^*}$, where $\mathcal{A}'$ denotes accuracy of the generated solution. Fig. 6 (a) presents the average, and Fig. 6 (b) presents the worst of 750 runs. We see that accuracy of HWT solutions are not only close to ideal solutions on average, but also consistently good.



(a) Length (b) Accuracy

Fig. 7. Solution qualities for multiple sequences



(a) Length (b) Accuracy

Fig. 8. Effect of number of sequences

We also investigated the scalability of our methods by keeping the compacted sequence length constant at 1000, and scaling the total sequence length and compaction factor. Sequence lengths are 5000 for $c = 5$, 10000 for $c = 10$, 20000 for $c = 20$, and so on. Compacted sequence qualities were within 93%, 93%, 93%, 93%, and 92% for $c = 5$, 10, 20, 40, and 80, on average, and within 89%, 89%, 88%, 89%, and 87% for $c = 5$, 10, 20, 40, and 80, in the worst of 100 sequences. These results show that HWT can effectively compact very long sequences with large compaction factors without any loss in performance.

Fig. 7 compares solution qualities of generating 2 and 3 sequences with a single sequence using four different compaction factor sets. The first (second) column in each set corresponds to generating two (three) sequences with the first two (three) compaction factors that are displayed under columns of that set. Single sequence solutions use the first number as the compaction factor. Ratios of total sequence length of multiple sequence solutions to length of a single sequence solution are presented in Fig. 7 (a). Fig. 7 (b) shows improvement in accuracies for multiple sequence solutions over a single sequence solution. Numbers are computed as $\frac{\mathcal{A}_k - \mathcal{A}_1}{\mathcal{A}_1}$, where $\mathcal{A}_1$ and $\mathcal{A}_k$ correspond accuracies of a single-sequence solution and multiple-sequence solution with $k > 1$ sequences, respectively. The first set shows that it is possible to get a more accurate solution with almost half the total sequence length. Generating a third sequence improves accuracy but slightly increases total sequence length. Improvement in total sequence length decreases with higher compaction factors, mostly due to the first compaction factor being too high, leaving very few edges for the first trail to cover. In another experiment, solution qualities (both in terms of accuracy and sequence length) are compared for generating $k = 1, 2, 3, 4$,

Table IV. SPICE simulations for MS and HWT

Circuit	$c = 3, 5, 2$		$c = 5, 8, 3$		$c = 10, 15, 7$	
Name	HWT	MS	HWT	MS	HWT	MS
i3	0.6	0.4	0.7	0.5	1.2	0.5
cordic	1.3	1.0	3.0	1.6	3.2	2.1
C432	0.8	0.6	0.6	0.5	1.1	0.7
C880	0.8	0.5	1.9	1.2	2.2	1.1
C1355	1.5	1.1	2.3	1.6	2.7	1.7
C1908	1.6	0.9	2.5	1.4	2.9	1.7
C3540	1.1	0.8	1.7	1.1	2.0	1.2
C6288	1.4	1.0	2.6	1.8	3.2	2.3
Averages	1.1	0.8	1.9	1.2	2.3	1.4

and 5 sequences, where compaction factors are chosen as the first k of $3, 5, 10, 20$, and 40. The first $k-1$ sequences use edges produced only by the first rule as was discussed in § 4. Fig. 8 (a) shows the ratio of total length of sequences to length of a single sequence, and Fig. 8 (b) shows improvement in accuracy, numbers are computed as $\frac{\mathcal{A}_k - \mathcal{A}_1}{\mathcal{A}_1}$, where $\mathcal{A}_1$ and $\mathcal{A}_k$ correspond to accuracy of a single sequence and a multiple sequence solution with $k > 1$ sequences, respectively. As seen in Fig. 8 (a), length of sequences can be significantly decreased by generating more sequences. Accuracy decreases slightly as we increase the number of sequences. This is probably because the first sequences with high compaction factors aggressively cover most of the positive weight edges, leaving little room for the last sequences to operate.

Finally, Table IV presents average error in SPICE simulations for 6 sequences of length 2000. MS corresponds to generating three sequences with compaction factors listed, and HWT corresponds to generating one sequence with the first compaction factor at the top of the column. The table shows that accuracies can be improved by generating multiple sequences. Error in estimations reduces to 0.8, 1.2 and 1.4 from 1.1, 1.9 and 2.3, respectively. The improvement is minor, because HWT estimations are already accurate, but the numbers justify generating multiple sequences as a valuable alternative for "harder" sequences.

6. CONCLUSION

We addressed the sequence compaction problem for efficient and accurate power estimation. We proved that transforming a sequence into a smaller one with minimum deviation in transition frequencies is NP-Complete. We also proposed a novel graph model to reduce the compaction problem to that of finding a heaviest weighted trail in a directed graph along with a decent heuristic to find such trails. Generating multiple compact sequences with different compaction factors is also discussed. Proposed methods have been applied to MCNC 91 benchmark circuits, using SPICE for simulations. Results showed that our methods can significantly reduce simulation times with very high accuracy. Moreover, transition frequencies are preserved to make results very reliable. Experiments also showed that multiple sequences yield more accurate estimations with even shorter sequences.

One possible extension of this work is to combine the proposed combinatorial framework with other compaction methods in the literature. Another extension is enhancing the method for sequential circuits, where switching activity is determined not only by the input vectors but also by the current state of the finite state machine (FSM) of the circuit. Although techniques for combinational circuits are

not directly applicable, we can simulate the FSM and mark each input vector with the state of the FSM. This translates the input sequence to a sequence of tuples of the input vector and the FSM state. Preserving transition frequencies on this tuple sequence will preserve the power characteristics. This translation reduces the problem to the sequence compaction problem studied in this paper, however it will suffer from enlarged memory requirement. The number of distinct vectors and transitions in the tuple sequence can be much larger than those of the original input sequence. Hybrid approaches might be employed to tradeoff between the accuracy of deterministic methods and the memory efficiency of probabilistic methods.

References

[1] Pedram, M., Advanced power estimation techniques, *Low Power Design in Deep Submicron Technology*, Edit. J. Mermet and W. Nebel. Kluwer Academic Publishers, 1997.

[2] Tsui C., Marculescu R., Marculescu D., and Pedram M., Improving efficiency of power simulators by input vector compaction, *Proc. 33rd ACM/IEEE Design Auto. Conf.*, pp. 165–168, 1996.

[3] Pedram M., Power simulation and estimation in VLSI circuits, *The VLSI Handbook*, Edit. W-K. Chen. The CRC Press and the IEEE Press, 1999.

[4] Marculescu D., Marculescu R., and Pedram M., Stochastic sequential machine synthesis targeting constrained sequence generation, *Proc. 33rd ACM/IEEE Design Auto. Conf.*, pp. 696–701, 1996.

[5] Huang S.H., Chen K.C., Cheng K.T., and Lee T.C., Compact vector generation for accurate power simulation, *Proc. 33rd ACM/IEEE Design Auto. Conf.*, pp. 161–164, 1996.

[6] Marculescu R., Marculescu D., and Pedram M., Vector compaction using dynamic markov models, *IEICE Trans. Fundamentals of Electronics Communications and Computational Sciences*, Vol. E80-A, No.10, pp. 1924–1933, 1997.

[7] Marculescu R., Marculescu D., and Pedram M., Hierarchical sequence compaction for power estimation, *Proc. 34th ACM/IEEE Design Auto. Conf.*, pp. 570–575, 1997.

[8] Pınar A., and Liu C.L., Power invariant vector sequence compaction, *Proc. 1998 IEEE/ACM Int. Conf. Computer-Aided Design*, pp. 473–476, 1998.

[9] Marculescu R., Marculescu D., and Pedram M., Block Entropy and High-Order Temporal Effects in Composite Sequence Compaction for Finite-State Machines, *Proc. ACM Int. Symp. Logic Programmable Electronic Devices*, 1997.

[10] Marculescu D., Marculescu R., and Pedram M., Sequence Compaction for Probabilistic Analysis of Finite-State Machines, *Proc. 34th ACM/IEEE Design Auto. Conf.*, pp. 12–15, 1997.

[11] Yang S., Logic Synthesis and Optimization Benchmarks User Guide V3.0, distributed as a part IWLS91 benchmark distribution, 1991.

[12] Papadimitriou C.H., and Steiglitz K., *Combinatorial Optimization*, Prentice Hall, 1982.

[13] Marculescu R., Marculescu D., and Pedram M., Sequence Compaction for Power Estimation: Theory and Practice, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol. 18, No. 7, pp.973–993, 1999.

[14] Ding C-S., Wu Q., Hsieh C-T., and Pedram M., Stratified random sampling for power evaluation, *IEEE Trans. Computer Aided Design*, Vol: 17(6), pp. 465–471, 1998.

[15] Iyengar V.S., Trevillyan L.H., and Bose P., Representative traces for processor models with infinite cache, *Proc. 2nd Int. Symp. High Perf. Comp. Architecture*, pp. 62–72, 1996.

[16] Iyengar V.S., and Trevillyan L.H., Evaluation and generation of reduced traces for benchmarks, *IBM Research Report*, RC 20610, 1996.

[17] Cormen T.H., Leiserson C.E., and Rivest R.L., *Introduction to Algorithms*, MIT Press, 1990.