Power Invariant Vector Sequence Compaction*

Ali Pınar
Department of Computer Science
University of Illinois at Urbana-Champaign

C. L. Liu
Department of Computer Science
National Tsing Hua University

Abstract

Simulation-based power estimation is commonly used for its high accuracy, despite excessive computation times. Techniques have been proposed to speed it up by transforming a given sequence into a shorter one while preserving the power consumption characteristics of the original sequence. This work proposes a novel method to compact a given input vector sequence to improve on the existing techniques. We propose a graph model to transform the problem to the problem of finding a heaviest weighted trail in a directed graph. We also propose a heuristic based on min-cost flow algorithms, using the graph model. Furthermore, we show that generating multiple input sequences yields better solutions in terms of both accuracy and simulation time. Experiments showed that significant reduction in simulation times can be achieved with extremely accurate results. Experiments also showed that the generation of multiple sequences improved the results further both in terms of accuracy and simulation time.

1 Introduction

The growing need for low-power systems raises two major issues: design optimization for low power and accurate estimation of power consumption. Both issues have been studied extensively in recent years. This work proposes a method to speed up simulation-based power estimation, which often suffers from excessive running times. For a CMOS circuit, the dominant source of power dissipation is the dynamic transition current. Other sources are much smaller and can be neglected. For a combinational circuit, power consumption corresponding to an input vector sequence depends only on the transitions between successive input vectors. So, 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 the power consumption for the original sequence.

Some recent works studied the problem of transforming a given vector sequence to a shorter one preserving the power characteristics. Tsui *et al.* [9] and Marculescu *et al.* [4] proposed methods for combinational circuits. These methods have the disadvantage of generating vectors that are not in the original sequence. Huang *et al.* [3] used a two phase strategy: they derived the transition profile of internal signals by a fast power estimator in the first phase, and then generated a new and shorter sequence using this profile in the second phase. Marculescu *et al.* [5] used a Markov model to generate a compact sequence, and subsequently they proposed a hierarchical model with macro and micro states to model the original sequence [6].

This paper proposes a novel method to transform a sequence to a shorter one preserving the transition frequencies. We will call this problem the *Sequence Compaction* problem. In this paper, we propose a graph model to transform the sequence compaction problem to the problem of finding a heaviest weighted trail in a directed graph. We propose a heuristic based on the min-cost flow problem [2] in graphs. We also discuss the problem of generating several input sequences with different compaction factors as opposed to generating merely a single input sequence. In this case, the power consumption for the original sequence can be computed as the weighted average of all sequences. This method can generate accurate results with shorter sequences.

The proposed techniques have been applied to MCNC 91 circuits [10]. The circuit behaviors were simulated for different input sequences using SPICE. Experiments show that significant reductions in simulation times can be achieved with highly accurate results. The error in estimations is limited to only 2%, where as the simulation time is reduced by a factor of 5. Moreover, the generated sequences are reliable for estimation, since they preserve the transition frequencies. We also showed that by generating multiple sequences the results can be improved further both in terms of accuracy and simulation time.

The rest of this paper is organized as follows. In Section 2 we will discuss the problem, propose a graph model, and describe a heuristic. Section 3 discusses generating multiple sequences. Experimental results are given in Section 4, and finally we conclude with Section 5.

2 The Sequence Compaction Problem

The sequence compaction problem aims at transforming a given vector sequence to a shorter one while preserving its characteristics. 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. The sequence compaction problem can formally be stated as:

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

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

The accuracy $\mathcal{A}(S, S')$ of a solution S' can be defined as $\mathcal{A}(S, S') = |T(S)| - \mathcal{C}(S, S')$, $|T(S)|$ being the cardinality of the set $T(S)$. So minimizing the cost is equivalent to maximizing the accuracy. The problem is NP-Complete [8], but we can not present the proof here due to space restrictions.

*Supported in part by the National Science Foundation (NSF) under grant 1-5-31333 NSF MIP 96 12184.

2.1 A Graph Model

In this section, we will describe a graph model for the representation of the problem. In this model, each distinct input vector in S will be represented by a vertex, and each transition will be represented by several weighted directed edges in the graph. We will assign the weights such that finding a heaviest weighted trail in this graph will be equivalent to finding an optimal compact sequence for S. Replacing each vertex on the trail by its corresponding input vector defines the corresponding sequence. The weight of an edge will be equal to the change in the cost of solution, if the corresponding transition of the edge appears in S'. More specifically, for a transition t_i in S, there will be several edges $e_{i1}, e_{i2}, \ldots$ with $w(e_{i1}) \geq w(e_{i2}) \geq \ldots$. If an optimal solution has j copies of the edge e_i, then there exists an optimal solution which uses the first j edges: $e_{i1}, e_{i2}, \ldots e_{ij}$, because the weights of the edges are nonincreasing. Exploiting this fact, the weight of the jth edge is set to be equal to the change in the cost, when transition t_i is added once more to S', which already has $j-1$ copies of transition t_i,

$$w(e_{ij}) = \frac{|S(t_i) - c * (j-1)| - |S(t_i) - c * j|}{S(t_i)}$$

A formal description for the construction of the graph follows. Let $S = \langle s_1, s_2, \ldots, s_i, \ldots, s_m \rangle$ be the input sequence and c be the compaction factor. In the graph $G = (V, E)$ representing this instance of the sequence compaction problem, each distinct input vector s_i in S is represented by a vertex v_i in V, and each vertex in V correspond to an input vector in S, and for each transition $t = (s_i, s_j)$ in S,

(*i*) The edge set E contains $\lfloor \frac{S(t)}{c} \rfloor$ copies of edge (v_i, v_j) with weight $\frac{c}{S(t)}$.

(*ii*) If $\lfloor \frac{S(t)}{c} \rfloor \neq \frac{S(t)}{c}$ then E has one more (v_i, v_j) edge with weight $\frac{2(S(t)\%c)-c}{S(t)}$, where $\%$ represents the modulo operation.

(*iii*) E contains M more copies of the edge (v_i, v_j) with weight $\frac{-c}{S(t)}$, where M is the total number of positive weight edges in the graph. Note that in the worst case each transition occurs $\frac{c+1}{2}$ times, and the number of positive weight edges can be bounded as $M \leq \frac{2*|S|}{c+1}$.

Edges generated by the first rule correspond to transitions that will always reduce the error in S'. On the contrary, edges generated by the last rule correspond to transitions that will always increase the error in S'. The edge produced by the second rule corresponds to approximating the $\frac{S(t)}{c}$ by either $\lfloor \frac{S(t)}{c} \rfloor$ or $\lceil \frac{S(t)}{c} \rceil$. There are only three possible weights for an edge between two vertices. Based on this observation, we can add capacities to edges in order to avoid too many edges in the graph.

A trail in this graph clearly describes a sequence for the compaction problem. The greater is the sum of weights of the edges this trail covers, the less will the cost be in the corresponding sequence. Negative weight edges can be included in the trail for the sake of forthcoming positive weight edges. A heaviest weighted trail in this graph gives the optimal solution for the sequence compaction problem.

Figure 1 illustrates an example. There are 5 transitions from B to D. The cost of the first $B \rightarrow D$ edge can be calculated as the difference in cost of zero appearances and one appearance, i.e., $w(e_{BD1}) = \frac{|5-3*0|-|5-3*1|}{5} = \frac{3}{5}$. The weight of the second edge is the difference between underestimating and overestimating the transition BD, and can be computed as $w(e_{BD2}) = \frac{|5-3*1|-|5-3*2|}{5} = \frac{1}{5}$. Other edges will cost $-3/5$. In this graph,

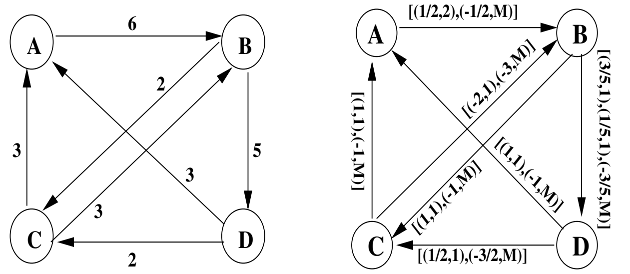

Figure 1: The graph representation for the sequence $S = \langle BDABCABDABDCABDABCBDCABC \rangle$ and the compaction factor $c = 3$. The figure on the left presents the number of transitions, and the one on the right presents the graph for S. Edge weights are written as [(weight of the edge, number of edges with this weight), ...]. 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 the total number of positive weight edges.

the maximum weighted trail, can be constructed as $B \rightarrow D \rightarrow A \rightarrow B \rightarrow C \rightarrow A \rightarrow B \rightarrow D \rightarrow C$. The weight of this trail is $\frac{3}{5} + \frac{3}{3} + \frac{3}{6} + \frac{3}{3} + \frac{3}{3} + \frac{3}{6} + \frac{1}{5} + \frac{1}{2} = \frac{159}{30}$. The corresponding compact sequence is $S' = \langle BDABCABDC \rangle$. The cost of this solution can be computed as $\mathcal{C}(S, S') = \frac{17}{10}$. Note that the 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 a heaviest weighted trail.

2.2 Finding a Heaviest Weighted Trail

The heuristic has three basic steps: (1) removing the positive weight cycles (2) finding a heaviest weighted trail on the reduced graph (3) improving the solution by adding back the cycles, if possible. Positive weight cycles on a graph can be detected by repeatedly applying the Bellman-Ford algorithm [1]. After removing the positive weight cycles, a heaviest weighted trail in the reduced graph can be found by using a minimum-cost flow algorithm. We will augment the reduced graph, and by finding the minimum cost flow in this augmented graph, we will identify a heaviest weighted trail in the reduced graph.

Let $G_1 = (V_1, E_1)$ be the graph induced by the removal of positive weight cycles. The flow graph $G_2 = (V_2, E_2)$ satisfies the following conditions. The set of vertices V_2 is equal to V_1 with a source vertex s and a terminal vertex t added, i.e., $V_2 = V_1 \cup \{s, t\}$. The node flows, nf, are: $nf(s) = 1$; $nf(t) = -1$; and 0 for all other vertices. The source vertex s is connected to each vertex in V_1, and each vertex in V_1 is connected to the terminal vertex t. The cost of these edges are all zero, and capacities are all one. If there exists an edge from v_i to v_j in E_1, E_2 contains a distinct edge from v_i to v_j for each distinct cost value for v_i to v_j edges. The cost of this edge is equal to the negative of the determining cost value, and the capacity is equal to the number of edges with this cost value in E_1.

The min-cost flow problem will identify a trail from s to t with the minimum cost. This trail corresponds to a heaviest weighted trail in the reduced graph. We have used Goldberg's algorithm and implementation [2], the complexity of which is $O(V^2 E \log(VC))$, where C is equal to the largest edge cost for finding teh min-cost flow solution. The trail found at the second step can be further improved by inserting the positive weight cycles removed at the first step.

Consider the example in Figure 1. The positive weight cycles in this graph can be detected as $B \rightarrow D \rightarrow A \rightarrow B$ and $B \rightarrow C \rightarrow A \rightarrow B$. The flow graph after removing these cycles and negating the weight of each edge is presented in Figure 2. The min-cost flow in this graph follows the path $s \rightarrow B \rightarrow D \rightarrow C \rightarrow t$. Removing the source and sink vertices and inserting the cycles removed before,

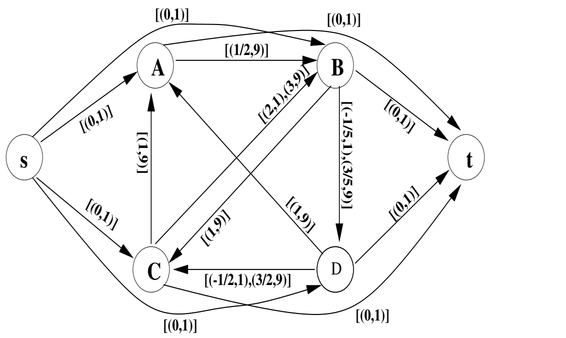

Figure 2: The flow graph for Example 2. Edge weights are written as [(weight of the edge, number of edges with this weight), ...]. e.g., For example, there is one edge with weight $1/5$ and 9 edges with weight $-3/5$ from B to D.

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 Constructing Multiple Sequences

So far, we discuss how an input sequence can be compacted as a single sequence with a given compaction factor. However, several sequences with different compaction factors can be generated to represent the original sequence, and the power consumption can be estimated by the weighted average of the power consumptions of these sequences. That is, a given input sequence S is represented by k subsequences $S_1, S_2, \ldots, S_k$ with compaction factors $c_1, c_2, \ldots c_k$, respectively, then the average power consumption $AP(S)$ can be estimated as

$$AP(S) = \frac{\sum_{i=1}^{k} P(S_i) * c_i}{\sum_{i=1}^{k} c_i * |S_i|}$$

where $P(S)$ is the total power dissipation for S. This approach can be helpful both in decreasing the total length of the input sequences and in achieving more accurate estimations. We will work on constructing a specified number of sequences for an input sequence S. The problem can be stated as:

Given an input sequence S, and compaction factors $c_1 \geq c_2 \geq \ldots \geq c_k$, construct sequences $S'_1, S'_2, \ldots, S'_k$ to minimize

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

Effective solutions to this problem can be found by using the graph model and the heuristic for finding a heaviest weighted trail already described. First, the sequence S'_1 for c_1 is constructed, and then the edge weights are recomputed considering S'_1, before constructing S'_2. Generally, during the construction of the graph for the ith sequence, the weight of an edge is computed to be equal to the change in the objective function, with $S'_1, S'_2, \ldots S'_{i-1}$ already constructed and $S'_{i+1}, S'_{i+2}, \ldots S'_k$ assumed to be empty sequences.

In this scheme, sequences with large compaction factors can greedily use negative weight edges to add more positive weight edges to the trail. However, these positive weight edges might be covered by another sequence in the upcoming compactions. So, it might be helpful to construct sequences with high compaction factors using only positive weight edges. More specifically, satisfying the condition, $S(t) \geq \sum_{i=1}^{j} c_i * S'_i(t)$ for the first few sequences usually helps, and this can be achieved by allowing the sequences to be constructed by only those edges produced by the first rule in the graph definition in Section 2.1.

4 Experimental Results

The methods proposed were implemented in C and applied to the MCNC91 benchmark circuits [10]. Input sequences were generated randomly but biased. The power consumption of the circuits for input sequences were measured with SPICE for maximum accuracy. We worked on 6 circuits and 3 compaction factors: $3, 5, 10$. We measured the power consumption of the circuits for 4 different sequences of length 1000. Then the sequences were compacted using the proposed heaviest weighted trail method (HWT) and the Markov model (MM) described in [5]. Table 1 presents the average accuracies for the two methods. In this table, the first three columns present the name, number of inputs and actual power dissipation of the circuit, respectively. The other columns present the accuracy in estimations, calculated as: $\frac{AP(S)-AP(S')}{AP(S)} * 100$, where $AP(S)$ denotes the average power dissipation for sequence S. The results show that HWT can predict the original power consumption very accurately, with negligible differences from the original values.

Table 1: SPICE Simulations

Name	#inp	Power μW	$c = 3$		$c = 5$		$c = 10$	
			HWT	MM	HWT	MM	HWT	MM
C432	36	1878.5	0.7	2.8	0.7	2.4	1.1	2.8
C880	60	3788.7	0.9	5.0	1.8	5.0	2.0	5.5
C1355	41	3956.2	1.8	8.8	2.5	9.3	2.6	9.2
C1908	33	6454.9	1.5	3.3	2.8	4.1	3.1	5.4
cordic	23	1641.5	1.4	4.4	3.1	4.9	3.3	4.6
i3	133	3856.8	0.5	2.2	0.7	2.0	1.7	1.9
Averages			1.1	4.1	1.9	4.6	2.3	4.9

Another important issue is the reliability of the compacted sequences. In a compacted sequence, some transitions may be overestimated while some others maybe underestimated, and these errors can cancel each other to give an accurate estimation. Such a compacted sequence is definitely not reliable. Since these compaction methods are proposed to avoid simulating long sequences, the user cannot determine the accuracy of the estimation. So reliability is a major concern. A reliable solution should estimate each transition accurately. We compared the solution qualities of HWT and MM in terms of reliability, for 200 input sequences of length 4000. We also wanted to see how far the solutions are from an optimal solution. Since the value of an optimal solution is not known, we used an upper bound on the accuracy of an optimal solution, which we call the accuracy $\mathcal{A}^*$ of an *ideal solution*. In an ideal solution, each transition is estimated in the most accurate way, i.e., a transition t, which appears $S(t)$ times in the original sequence S, should appear $round(\frac{S(t)}{c})$ times in S', where $round$ maps the number to the nearest integer. The accuracy computed this way is only a bound on the optimal value, because there does not necessarily exist a sequence to realize this. Note that the accuracy of an ideal solution $\mathcal{A}^*$ is equal to the sum of weights of positive weight edges in the associated graph. Table 2 presents the results (the numbers in parentheses display the std. deviation). The second

Table 2: The reliability for HWT and MM

c	$\lvert S' \rvert$	$\frac{\mathcal{C}(S,S'_{MM})}{\mathcal{C}(S,S'_{HWT})}$	Acc. HWT
3	852 (87)	3.92 (0.41)	82.0(2.69)
5	592 (68)	3.61 (0.42)	85.3(5.22)
10	329 (33)	3.36 (0.38)	88.2(2.96)

column in Table 2 presents the average length of the sequences generated by the HWT method and shows that HWT produces much shorter sequences than MM. The third column displays the average of the ratio of costs of the solutions of the two methods. The numbers show that the costs of solutions of HWT are more than 3 times smaller than those of MM, meaning that the solutions by the HWT are much more reliable. The ratio becomes smaller with increasing compaction factor. The reason for this increase should be attributed to the increase in the cost of an optimal solution for large compaction factors. Since, the cost of an optimal solution becomes higher, the cost due to the imperfectness of the solution becomes less effective in this ratio. The last column presents the comparison with the ideal solution. The numbers were calculated as: $\frac{\mathcal{A}' * 100}{\mathcal{A}^*}$, where $\mathcal{A}'$ denotes the accuracy of the produced solution. It can be seen that the accuracy of solutions of HWT are within 18% of the ideal solutions for $c = 3$, going down to 14.7% and 11.8% for $c = 5$ and 10, respectively. This decrease is most likely because the bound on the value of an optimal solution becomes tighter as the compaction factor increases. The results show that the HWT is not only more reliable than MM, but also generates solutions that are very close to optimal. The compaction times are negligible compared to the simulation times.

The next set of experiments observes the performance of generating several input sequences. We worked on 200 input vectors of length 4000. Table 3 presents the results for these experiments. The table displays the total length of sequences for generating $k = 1, 2, 3$ sequences. Each column displays the results for using the first k compaction factors indicated in that row. The accuracy for $k = 1$, and 2 is computed as $\frac{\mathcal{A}' * 100}{\mathcal{A}^*}$, where $\mathcal{A}^*$ is the accuracy of an ideal solution for c equal to the first number in that row. For $k = 3$ we computed the improvement in accuracy as $\frac{\mathcal{A}_2 - \mathcal{A}_3 * 100}{\mathcal{A}_2}$, where $\mathcal{A}_2$ and $\mathcal{A}_3$ stand for the the accuracy of the solutions for $k = 2$ and 3, respectively (Note that $\mathcal{A}^*$ is no longer an upper bound, because the last compaction factor is smaller than the compaction factor $\mathcal{A}^*$ is computed for). The results show that

Table 3: Reliability of generating multiple sequences

c	$k = 1$		$k = 2$		$k = 3$	
	$\|S'\|$	Acc	$\sum_i \|S'_i\|$	Acc	$\sum_i \|S'_i\|$	Imp
3,5,2	852	82.0	808	89.7	1024	5.1
5,8,3	592	85.3	567	89.6	715	13.4
10,15,7	329	88.2	318	89.7	383	4.3

by using compaction factors greater than the intended compaction factor (the compaction factor if only one sequence would be generated, which is the first number in each row in this case) more accurate solutions can be achieved with shorter sequences. The accuracy is increased by 7.7%, 4.3% and 1.5%, whereas the lengths of the sequences are around 5% shorter. By using a compaction factor lower than the intended compaction factor, the accuracy can be improved with the cost of longer sequences. As the last set of experiments, we measured the power consumption for these sequences on circuits. In Table 4, MS correspond to generating three sequences with compaction actors listed, and HWT correspond to generating one sequence with the first compaction factor at the top of the column. The table shows that the accuracies can be improved by generating multiple sequences. The error in estimation is only 0.9%.

5 Conclusion

This paper addressed the input sequence compaction problem for efficient power estimation. We described a novel graph model which reduces the sequence compaction problem to the problem of finding a heaviest weighted trail in a directed graph. Then a decent heuristic for constructing a heaviest weighted trail was proposed. The paper also proposed generating multiple compact sequences with different compaction factors as opposed to constructing merely a single sequence. The experimental results showed that simulation times can be significantly reduced with a negligible difference in accuracy. Furthermore, generating multiple sequences helped to reduce the simulation times and to increase accuracy.

Table 4: SPICE Simulations for MS and HWT

Name	$c = 3, 5, 2$		$c = 5, 8, 3$		$c = 10, 15, 7$	
	HWT	MS	HWT	MS	HWT	MS
C432	0.7	0.6	0.7	0.6	1.1	1.0
C880	0.9	0.7	1.8	1.2	2.0	1.5
C1355	1.8	1.5	2.5	2.0	2.6	2.2
C1908	1.5	1.2	2.8	1.7	3.1	2.2
cordic	1.4	1.1	3.1	2.4	3.3	2.7
i3	0.5	0.5	1.3	0.5	1.7	0.7
Avg.	1.1	0.9	1.9	1.4	2.3	1.7

References

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

[2] A.V. Goldberg, An efficient implementation of a scaling minimum-cost flow algorithm, J. Alg., Vol: 22, pp:1–29, 1997.

[3] S.H. Huang, K.C. Chen, K.T. Cheng and T.C. Lee, Compact vector generation for accurate power simulation, *Proc. 33rd DAC.*, pp. 161–164, 1996.

[4] D. Marculescu, R. Marculescu, M. Pedram, Stochastic sequential machine synthesis targeting constrained sequence generation, *Proc. 34th DAC.*, pp. 696–701, 1996.

[5] R. Marculescu, D. Marculescu, M. Pedram, Vector compaction using dynamic markov models, *IEICE T. Fund. of Electronics Comm. and Comp. Sci.*,E80-A(10), 1997.

[6] R. Marculescu, D. Marculescu, M. Pedram, Hierarchical sequence compaction for power estimation, *Proc. 34th DAC.*, pp. 570–575, 1997.

[7] F.N. Najm, A survey of power estimation techniques in VLSI circuits, *IEEE Trans. on VLSI Systems* 2(4), 1994.

[8] A. Pınar, C.L. Liu, Power invariant vector sequence compaction, *to be submitted to IEEE Trans. CAD*

[9] C. Tsui, R. Marculescu, D. Marculescu, M. Pedram, Improving efficiency of power simulators by input vector compaction, *Proc. 33rd DAC*, pp. 165–168, 1996.

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