PowerDrive: A fast, canonical POWER estimator for DRIVing synthEsis

Sumit Roy Harm Arts Prithviraj Banerjee†
Ambit Design Systems
2500 Augustine Drive,
Santa Clara, CA 95054, USA
sroy, harm@ambit.com

Electrical and Comp. Engineering
Northwestern University
Evanston, IL-60208, USA
banerjee@ece.nwu.edu

Abstract

The computational complexity of a probability-based combinational power metric lies in the creation of a BDD for each node in the circuit. In this paper, we formalize the problem of finding intermediate support-set which controls the size of BDD. We propose an exact algorithm to solve it. We also propose an heuristic solution, **PowerDrive***, for estimating power of large circuits. Apart from being more accurate and* **several times faster** *than [2, 7], PowerDrive possess the unique quality of being* **canonical** *and of* **constant** *complexity, a very desirable quality for a power metric guiding a synthesis tool. Finally, the proposed power metric was able to guide the synthesis tool [11] to optimize large circuits which could not be synthesized by POSE [6], thus proving the effectiveness of our power metric.*

1 Introduction

Accurate and fast power estimation during the design phase is required to guide power optimization techniques employed to meet the stringent power specifications. The probabilistic techniques [4, 8, 9] are examples of such estimation techniques. They use symbolic simulation in order to produce a set of boolean functions representing conditions for switching at each gate in the circuit. These techniques allow the user to cover a large set of possible input patterns very quickly. Hence probabilistic techniques are used for driving power-driven logic synthesis tools [6].

The complexity of any probability-based combinational power estimator lies in the creation of BDD for each node in the circuit. Global BDDs cannot be constructed for large circuits. Hence local BDDs are created with an intermediate support-set. In this paper, we propose algorithms to identify the intermediate support-set necessary for computing power metrics guiding a logic synthesis tool. We formulate it as a problem of finding an optimally-correlated L-feasible frontier. We propose an exact solution and experimentally show that solving this problem provides good estimate of the signal probability. We present a heuristic algorithm that is computationally bounded, considers local convergence and is canonical (explained later). We provide experimental results on a wide range of benchmarks. Our approach is general enough to use any of the power estimation models reported in [8, 9]. Finally the scope of this work is not to perform final power estimation of a design but to be used for computing the power metric required by a low-power driven synthesis tool.

† This research was supported in part by the National Science Foundation under grant MIP-9320854 and the Semiconductor Research Corporation under contract SRC 96-DP-109.

Table 1: Moves evaluated during an optimization step in [1].

Circuit	gates	moves	time/gate	estTime
C432	136	21554	2.03	43810
C499	239	36756	2.17	79903
C880	221	25628	1.45	37121
C1355	237	39930	1.41	56107
C1908	288	34654	1.73	59907
C3540	692	34654	4.77	513132
C6288	2341	151937	1.94	294063

The rest of the paper is organized as follows: We present the related work in Section 2. Section 3 provides some definitions. In Section 4, we solve the problem of finding support-set exactly. We provide a heuristic solution in Section 5. We provide experimental results in Section 6 and conclude in Section 7.

2 Related Work

In [8], the authors proposed a solution to consider spatial correlation without creating global BDDs. They modeled the correlations using *lag-zero Markov chain* and propagated them from the inputs to the outputs of the circuit. In a feasibility experiment, we calculated the number of candidate *moves* which are considered during a typical optimization step of an industrial synthesis tool, BuildGates [1], for various MCNC benchmarks and tabulated in Table 1. The columns *gates*, *time/gate* and *estTime* report the number of gates in the circuit, the average time (in seconds/gate) and the total estimated time (in seconds) to compute average power (projected from the reported results in [8]). This table indicates that the technique is computationally expensive for large MCNC circuits. Moreover, the complexity of the proposed technique [8] is superlinear in the size of the circuit. Hence it may be impractical for guiding optimizations of large industrial circuits.

In [7], Kapoor proposed heuristics to partition the circuit and build local BDDs to calculate the switching activity of nodes. A drawback of the method is that the support-set contains either immediate fanins of the node or the primary inputs. It ignores support-sets with internal nodes. In Figure 1, the bold line shows the desirable support-set since the signals a and b are independent. But the approach in [7] finds the dashed line as the support-set when the limit on the support-set is set to 2.

In [2], a new approach is proposed that tries to cover as much reconvergent nodes as possible in order to account for the spatial correlation. The basic problem is that the heuristic stops at the shallowest support-set with size larger than the specified maximum value as illustrated in Figure 1.

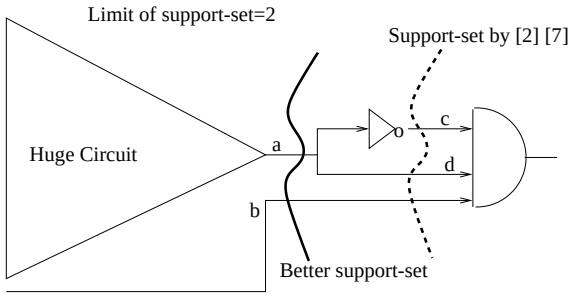

Figure 1: Drawback of the method [2] and [7].

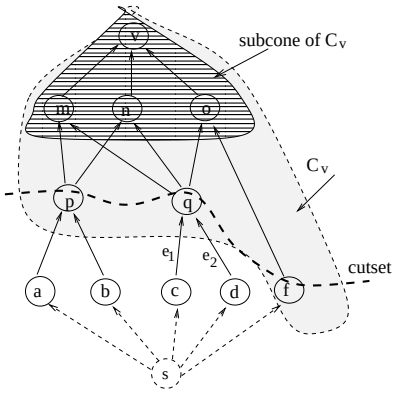

Figure 2: Illustrates graph terminologies.

3 Definitions

In this section, we will describe the various terminologies used and the power model assumed in the paper.

The *signal probability* $p_s(x)$ at a signal x is defined as the average fraction of clock cycles in which the steady state of x is a logic high. If a zero-delay model and temporal independence is assumed, the average power consumed by a signal is given by:

$$P_{av} = \frac{1}{2.T_c}.V_{dd}^2.\sum_{i}^{n} 2.C_i.p_s(x_i).(1-p_s(x_i)) \qquad (1)$$

where T_c is the clock period, C_i is the total capacitance at signal x_i, V_{dd} is the supply voltage and n is the total number of signals in the circuit. In [8] and [9], the authors have reported power measures which accounts for temporal correlation and non-zero delay model, repsectively. In this paper, we will use Equation 1 for measuring the average power, but our work is also applicable to the other power models [8, 9]. The focus of this paper is to compute p_s efficiently and accurately.

3.1 Definitions from Graph Theory

A combinational Boolean network $\mathcal{N}(V, E)$ can be represented as a directed acyclic graph (DAG) where each vertex in V represents a logic gate and a directed edge(i, j) in E exists if the output of gate i is an input of gate j. We define u to be a *predecessor* of v if there is a directed path from u to v. If H is a subgraph of $\mathcal{N}$, its support, $sup(H)$, is the set of *distinct* vertices in H which are either primary inputs or have fanins outside H.

A $cone$ of a signal v of the network $\mathcal{N}$, $C_v(V', E')$, is a subgraph of $\mathcal{N}$ such that 1) if $u \in V'$, u is a predecessor of v and 2) if $u \in V'$, any path from u to v lies entirely in C_v. A $subcone$ of a cone C_v is a cone that is also a subgraph of C_v. A cone C_v is *L-feasible* if $|sup(C_v)| \leq L$. The *volume* of a cone $C_v(V, E)$, $vol(C_v)$ is $|V|$. A $fanin\ cone$ of a vertex v in a network, FC_v, is the cone with all the vertices in $sup(FC_v)$ being primary inputs.

A vertex u is *reconvergent* if there are more than one disjoint path from u to a given vertex $v \in V$. Ideally, we would like to find the amount of spatial correlation of u captured within a cone C_v. Since we are not aware of an efficient metric for measuring spatial correlation and it is caused by reconvergent paths, we define a metric based on the latter to measure the spatial correlation.

Definition 1 Degree of reconvergence*: The degree of reconvergence of a vertex u in a cone C_v, denoted $\eta(u, C_v)$, is one less than the number of fanouts of u in C_v.*

$$\eta(u, C_v) = |\{w : w \in (fanout(u) \cap C_v)\}| - 1 \qquad (2)$$

In Figure 2, $fanin(p) = \{a, b\}$, $fanout(p) = \{m, n\}$ and d is a predecessor of o. Let C_v be the cone marked by the shaded region. The subcone of C_v is shown in dashed region. $sup(C_v) = \{p, q, f\}$. C_v is 3-feasible but not 2-feasible. $vol(C_v) = 7$. $FC_q = (\{q, c, d\}, \{e_1, e_2\})$. p and q are reconvergent vertices in C_v. The degree of reconvergence of q, $\eta(q, C_v) = 2$.

Given a network $\mathcal{N}$ with a source s and a sink t, a $cut(X, \overline{X})$ [3] is a partition of the vertices in V such that $s \in \overline{X}$ and $t \in X$. The *vertex cut-set* of $(X, \overline{X})$, denoted $cutset(X, \overline{X})$, is the set of vertices in $\overline{X}$ that are adjacent to some vertex in X. The *vertex cut-size*, $n(X, \overline{X})$, is the sum of weight of the vertices in a cutset. A cut $(X, \overline{X})$ is an *L-cut* if $n(X, \overline{X}) \leq L$.

In Figure 2, s is the source and v is the sink of the network. The cut corresponding to the dashed line is given by $cut(X, \overline{X}) = (\{v, m, n, o\}, \{p, q, a, b, c, d, f, s\})$. The vertex cut-set is given by $cutset(X, \overline{X}) = \{p, q, f\}$ and the vertex cut-size of the cut is $n(X, \overline{X}) = 3$, assuming all vertices have weight of 1. $vol(X, \overline{X}) = 4$.

4 Problem Formulation

The complexity of a power metric considering spatial correlation is proportional to the BDD size. If we assume that restricting the support-set of a BDD controls its size, our objective is to identify an L-feasible support-set of a given signal, where L is an user defined constant. We would like to find an *L-feasible* cone of v that calculates the signal probability of v as accurately as possible. Since the accuracy of $p_s(v)$ is very difficult to characterize based on the underlying Boolean network and it is dependent on the set of reconvergence paths in FC_v, we will use the degree of reconvergence (Equation 2) as a measure of accuracy. Larger the total degree of reconvergence, more the correlation it will contain and better will be the power estimation. Hence we would like to find that L-feasible cone that maximizes the summation of the degree of reconvergences and hence captures maximum spatial correlations.

Problem Specification: An *optimally correlated L-feasible cone* of a vertex v, denoted $C_v^*(V^*, E^*)$, is an L-feasible cone, such that there is no lesser volume L-feasible cone that has higher degree of reconvergence than C_v^* has. Thus, if $C_v(V, E)$ is an L-feasible cone of v, $\sum_{u \in V} \eta(u, C_v) \leq \sum_{w \in V^*} \eta(w, C_v^*)$. In case of equality, $vol(C_v^*) \leq vol(C_v)$. The $sup(C_v^*)$ is called

the *optimal frontier*. In Figure 2, C_v is an optimally correlated 3-feasible cone of v and $\{p, q, f\}$ is an optimal frontier. In the following section we will describe a divide and conquer algorithm to find the optimally correlated L-feasible cone.

4.1 Exact Algorithm

Since we will be using network flow algorithms to obtain the C_v^*, we convert the given DAG into a network with v as the sink and connect all the primary inputs of the circuit to a dummy *source* vertex s (shown by the dashed arrows in Figure 2).

A **max-vol, min-cut** $(X, \overline{X})$, **(MVMC)**, is a cut of a network such that the *vertex cut-size* is minimum and the $vol(X)$ is maximum. Since there can be several min-cuts in a network, the MVMC identifies the one that has the largest number of vertices on the sink side. In Figure 2 there are two min-cuts, namely $cut(X1, \overline{X1}) = (\{v\}, \{m, n, o, p, q, a, b, c, d, f, s\})$ and $cut(X2, \overline{X2}) = (\{v, m, n, o\}, \{p, q, a, b, c, d, f, s\})$. with *vertex cut-size* of 3. But $cut(X2, \overline{X2})$ is the MVMC of the network. [3] proved that such cut is unique for a network and described a polynomial algorithm to obtain it. We prove the following relation between MVMC and optimally correlated cone.

Theorem 1 *If C_v^* is an optimally correlated L-feasible cone with vertex set Y and $(X, \overline{X})$ is a MVMC with $n(X, \overline{X}) \leq L$, then $X \subseteq Y$.*

Proof: Excluded due to shortage of space. Refer to [10] for the proof of all the theorems and lemmas in this paper. □

From the above theorem, it follows that the cone corresponding to the MVMC $(X, \overline{X})$ is a subgraph of the optimally correlated cone. Hence, the optimal frontier is confined in $\overline{X}$. So we can remove the MVMC from the original network without losing the optimal frontier. Given a network G we will define a transformation that removes the sink partition of MVMC $(X, \overline{X})$ of G and connects the sink vertex v to $cutset(X, \overline{X})$.

Definition 2 Sink-chopping: *Given G, if $(X, \overline{X})$ is the MVMC of G, the sink-chopped graph, denoted $\rho(G)$, contains vertices $V = \overline{X} \cup \{v\}$ and edges E consists of all the edges in G connecting the vertices $\overline{X}$ and extra edges from $cutset(X, \overline{X})$ to v.*

Figure 3 illustrates sink-chopping. Given the network, the MVMC is shown by the shaded region in Figure 3(a). This region is chopped out and the node v is added to get the sink-chopped network as shown in Figure 3(b).

We will now describe the basic principle of the exact algorithm. We start with the original network, G_0, perform *sink-chopping* to get G_1. If the min-cut size in G_1 is less than L, we continue the process until the cut-size exceeds L. At this point, we may assume that the iterative application of MVMC algorithm can provide us the optimal frontier. A closer look reveals that the MVMC of $\rho(G)$ is always the immediate fanins of v, since it was the deepest min-cut in G. Figure 3(a) shows the original network with the cutset of the MVMC in dashed bold line. The sink-chopped network is shown in Figure 3(b). As mentioned above, the min-cutset of the network in Figure 3(b) is the fanin(v) as shown by the dashed line. We overcome this problem by artificially increasing the weight of one of the vertices (shaded vertex in Figure 3(b)) in the fanin of v to infinity. This forces the cut-size of $fanin(v)$ to infinity and leads to a new cutset (bold line in Figure 3(b)). If $u \in fanin(v)$ in G, then G^u represents the graph G with $wt(u) = \infty$.

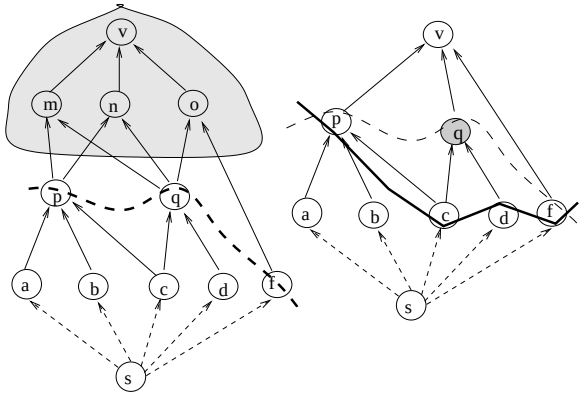

Figure 3: Diagram showing sink-chopping of a network.

```
OPTIMALLYCORRFRONTIER(G)
 1  ▷ Divide and conquer
 2  BestFront = fanin(v);
 3  n ← |BestFront|
 4  if (n = L)
 5      return BestFront;
 6  if (n > L)
 7      return NULL;
 8  for j ← 1 to n
 9      u_j ← fanin(v)[j];
10      G' ← ρ(G^{u_j});
11      f^j ← OptimallyCorrFrontier(G');
12      if (IsBetterFrontier(f^j, BestFront))
13          BestFront ← f^j;
14  return BestFront;
```

Figure 4: An exact algorithm to find the optimal frontier.

Theorem 2 *The optimal frontier of G is either $fanin(v)$ or one of the optimal frontier of $\rho(G^{u_j})$, where $u_j \in fanin(v)$.*

Figure 4 describes the exact algorithm to find the optimal frontier of a given graph. It is based on a divide-and-conquer technique. Based on Theorem 2, the optimal frontier is either the $fanin(v)$ or one of the optimal frontier of $\rho(G^{u_j})$, that is found recursively. *IsBetterFrontier* computes the number of reconvergent paths in the cone cut by f^j and if it is same as that of the previous *BestFront*, it compares the volume of the two cones. If the new frontier is better, it is assigned *BestFront*. BestFront contains the best amongst the optimal frontiers of $\rho(G^{u_j})$, for all $u_j \in fanin(v)$. We return *BestFront* to the previous level of recursion. When the recursion ends, we obtain the optimal frontier of the original solution. If the original graph had min-cutsize larger than L, $fanin(v)$ is returned.

Lemma 1 *The algorithm described in Figure 4 finds the optimal frontier.*

The complexity of the above algorithm is $L!.k|E|$ for a k-feasible graph G with E edges. Although it is polynomial in problem size (size of circuit), the constant is large for large values of L. We use the above algorithm to validate our problem formulation in Section 4. We show in Section 6 that solving the formulated problem, leads to very accurate estimate of the power. In the next section, we describe a heuristic solution to the above problem.

```
HEURISTICCORRFRONTIER(FC_v, d, L)
1  ▷ A d-deep BFS is performed
2  LabelVertsTillDepth(d);
3  for i ← 1; i < d; i++
4      C_v^i ← FindDdeepCone(i);
5      f ← FindCheapestCFF(C_v^i);
6      if (|f| ≤ L)
7          f* = f;
8  return f*.
```

Figure 5: A heuristic algorithm to find the optimal frontier.

5 A Heuristic Solution

In this section, we will describe a constant complexity algorithm of the problem defined in Section 4. The algorithm to find support-set is local, canonical and bounded in complexity. We will gradually explain the importance of each of these properties for a power estimation algorithm guiding a power optimization tool. To our knowledge, we are the first to report such an estimation algorithm. The algorithm is based on the breadth first search and the algorithm described in [5] to find the cheapest correlation free frontier. A *correlation free frontier* (**CFF**) of a DAG is a cutset such that the signals in the cutset are not spatially-correlated. The cheapest CFF corresponds to that CFF that has the least volume in the sink side of the corresponding cut. The dashed line in the Figure 2 is the cheapest CFF whereas the cutset $\{p, c, d, f\}$ is not.

5.1 Locality of Solution

A $depth$, denoted d(u,v), for a predecessor u of v is the length of the longest path from u to v. In Figure 3(a), d(q, v)=2 and d(b, v)=3. The depth of all the nodes in the fanin cone of v, FC_v can be found by a breadth-first traversal of the fanin cone.

Definition 3 m-deep cone*: An m-deep cone, denoted C_v^m, of a node v in a graph G, is a cone of v in G with every node in the support of C_v^m having depth greater than d from v.*

The shaded portion in Figure 3(a) is a 1-deep cone of v whereas the cone of v above and including the dashed line is C_v^2. We will confine our search of the optimal frontier within a d-deep cone of v, thereby capturing all the local spatial correlation without compromising on the complexity.

Figure 5 describes the heuristic algorithm to find the optimal frontier. We first find the depth of all the nodes till d-level deep by performing a d level breadth first traversal. Next, we find the i-deep cone for each i and assume that the support of it is independent. We find the cheapest CFF for the i-deep cone and if its size is smaller than L, it is assigned to f^*. We continue till we find the deepest CFF within the d-deep cone with size less than L.

We will apply the above algorithm to the graph in Figure 2 for finding a 3-feasible optimal frontier with the maximum depth, $d = 3$. In the first iteration of the algorithm, we compute C_v^1 (dashed area). We find the cheapest CFF of C_v^1 that is $\{m, n, o\}$. In the next iteration, we find C_v^2 (shaded area). The cheapest CFF for C_v^2 is shown by the dashed line. Although, C_v^3 is the entire graph, the final solution is the same dashed line since the circuit below it is spatially independent. In the next section, we will describe an important property of the above algorithm.

5.2 Canonical

During optimization, it is desirable that if two candidates for a transformation are identical but are represented by different input orders, the power metric should be same for both of them. Let us consider two n-input AND gate, one connected to signals $x_1, \ldots, x_n$ and the other to $x_n, \ldots, x_1$. If the power estimation is non-canonical, it is possible that it could potentially identify two different frontiers for the two AND gates. In fact, for the $n!$ combinations of the input, it can potentially produce $n!$ different power metric value resulting in the synthesis tool to waste time proportional to $n!$. Hence it is desirable that the power metric is invariant of the input ordering of the circuit.

More importantly, if the power metric is not canonical, the synthesis results may not be reproducible. In an industrial flow, while synthesizing large circuits, the synthesis is interrupted periodically for various reasons and restarted. The input ordering of the circuit in the internal database may be different before and after the interruption, although the circuit is logically identical. Hence a non-canonical power metric may drive the synthesis to take two different optimization paths for the uninterrupted and interrupted flow and the synthesis process becomes irreproducible. This is very undesirable since it affects the design convergence cycle adversely and also makes debugging impossible. The importance of a canonical power metric for driving optimization are:

1. It avoids unnecessary optimization moves that does not change the circuit.
2. It allows the designer to interrupt the optimization at various stages and obtain the same final result.
3. It enables designers to debug designs from a previous checkpoint rather than from the beginning thus saving several hours to days of synthesis effort.

Definition 4 Structural canonicity*: A power metric is structurally-canonical if its value for any signal of two circuits, with isomorphic underlying DAGs, are identical, assuming identical input probabilities on the primary inputs of both circuits.*

Thus if we compute the power metric of any node in a circuit and then permute its primary inputs, a structurally-canonical power metric would produce the same value. In simple terms, the power metric does not depend on the input ordering. One of the biggest drawback of the previous techniques [7, 2] was that their power metric were not structurally-canonical. The following lemmas and theorem prove that our power metric is structurally-canonical.

Lemma 2 *A* m-deep cone *of a node v in a graph G is unique.*

Lemma 3 *The cheapest CFF for a fanin cone of v is unique.*

Theorem 3 *If two graphs are isomorphic, the algorithm described in Figure 5 will produce frontiers equivalent to isomorphism.*

5.3 Constant Complexity

Lemma 4 *The complexity of the algorithm described in Figure 5 is $k.m.|V'|^2$, where V' is the set of vertices in the C_v^m cone and k is the maximum fanin size of any vertex in C_v^m.*

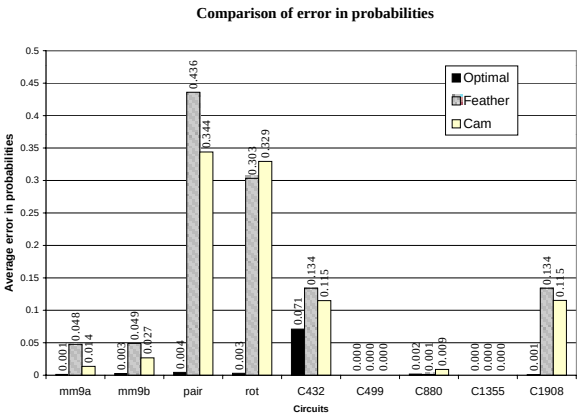

Figure 6: Figure to validate our problem formulation.

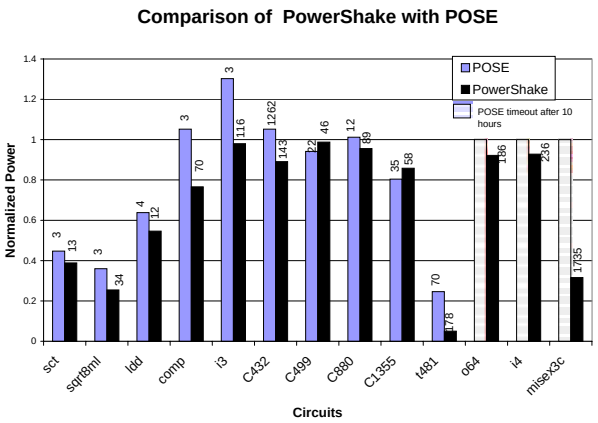

Figure 7: Efficacy of PowerDrive in driving optimization. (Numbers in top of bar are run time in seconds.)

The value of k is equal to the number of input pins of the largest cell in a technology library and is fixed for a given library. As $|V'|$ is bounded by k^d and d is a user-defined constant, $|V'|$ does not depend on the circuit size. Hence our algorithm has constant complexity (in size of the circuit) for a given technology library.

6 Experimental Results

In this section, we will describe the results of the several experiments we performed. We implemented the exact and the heuristic algorithms described in Section 4 and Section 5 respectively into a tool called **PowerDrive**. In the experimental results, we will use mean error to measure the accuracy. **Mean error** of an algorithm for a circuit corresponds to the *average absolute error* of the signal probability of all its primary outputs with respect to the accurate probabilities(P_{acc}) obtained using global BDDs. For example, let $P_{opt}(x)$ be the signal probability of x based on the support-set obtained by Figure 4 and PO be the primary outputs of a circuit. Then the mean error of the algorithm in Figure 4 is given by: $mean\ \ error = \frac{1}{|PO|} \sum_{x \in PO} abs(P_{opt}(x) - P_{acc}(x))$ where abs computes the absolute difference. Since primary outputs are expected to have the maximum error, the results reported are pessimistic. In all our experiments the size of the support-set was limited to 12.

The first experiment was conducted to validate the problem formulation in Section 4. The results are shown in Figure 6. We implemented the exact algorithm described in Figure 4 to find the intermediate support-set of all the primary outputs of the various MCNC circuits and then computed the probability(P_{opt}) based on it. The bars labeled *Optimal*, *Feath* and *Cam* represent the mean error computed from the support-set obtained from the algorithms described in Figure 4, reported in [2] and [7] respectively. The mean error obtained using our optimal algorithm is less than 0.005 for all circuits but C432. This shows that our problem formulation was appropriate since solving it restrains the mean error of node specific power within 0.005. C432 contains couple of nodes close to the primary inputs where at least 30 paths converge. Being deep inside the circuit, it could not be included and hence it leads to the inaccuracy in the probability value. The number of BDDs used by all the three methods were comparable and were less than 3000 nodes.

In the second experiment, we establish the efficacy of our algorithm (*PowerDrive*) described in Figure 5. We computed the probabilities of all the primary outputs of **50** MCNC benchmarks using PowerDrive (*PowDr*), Feather (*Feath*) [2] and Cam (*Cam*) [7]. Table 2 tabulates the mean error (scaled by 10^3), the bdd size to compute it and the runtime of each of the above algorithms for 15 of the circuits. The complete table is excluded due to shortage of space and is reported in [10]. *Global* tabulates statistics corresponding to the global bdds. The table shows that the mean error of Feather [2] and Cam [7] are *several times* more than the mean error of PowerDrive. On an average over **50** circuits, the mean errors of Feather and Cam are **85%** and **28%** more than that of PowerDrive. $C432$ produces the maximum error among all the 50 circuits. PowerDrive is **19** times faster than Feather and **twice** as fast as Cam on an average over the 50 circuits. The runtime of PowerDrive is lesser because it uses a constant complexity algorithm compared to the linear complexity algorithms used by the other two techniques. Also, the BDD size for calculating the power metric is slightly smaller than that of the previous approaches. The above experiment demonstrates that PowerDrive is a superior technique compared to Feather and Cam both in accuracy and speed.

In the final experiment, we integrated PowerDrive into the low power driven synthesis tool PowerShake [12, 11]. We used PowerShake driven by PowerDrive and POSE [6] to optimize the MCNC benchmark circuits, mapped to mcnc.genlib using the technology mapper present in SIS [13] and used the power estimator in SIS to estimate the power(f=20MHz, $V_{dd} = 5V$). The bar chart in Figure 7 reports the power consumed by circuits optimized by POSE and PowerShake normalized with respect to the power of the original circuit. The objective of the experiment is not to compare between PowerShake and POSE (comparison reported in [11, 12]) but to show the effectiveness of PowerDrive

Table 2: Comparison of PowerDrive with previous approaches.

Circuit	Mean Error (scaled by 10^3)			Bdd size				Time (seconds)			
	PowDr	Feath	Cam	PowDr	Feath	Cam	Global	PowDr	Feath	Cam	Global
cm150a	0.00	29.80	39.10	3	3	5	34	0.0018	0.0578	0.0025	0.0115
pcler8	0.03	7.49	6.61	158	159	123	240	0.0235	0.0532	0.0198	0.0383
C432	115.66	134.54	115.30	41	170	103	62,681	0.0237	1.6006	0.0525	34.7953
C499	0.00	1.98	0.00	195	1080	242	186,864	0.1261	4.4875	0.2960	19.7204
C880	1.38	14.00	9.31	349	525	419	525,001	0.0495	0.6850	0.0753	36.9677
C1355	0.00	0.00	0.00	269	915	469	4,296,288	0.1452	11.5992	0.4450	706.2347
C1908	0.77	16.99	0.92	80	685	326	98,566	0.0659	4.0065	0.2604	18.0691
my_adder	0.00	60.06	21.27	34	34	181	476	0.0148	0.9659	0.0892	0.2822
mult16a	0.00	54.15	12.31	34	34	236	2,911	0.0181	1.2371	0.0977	1.0970
sbc	3.00	10.48	4.12	176	176	936	2,577	0.0758	0.4223	0.1686	0.3579
term1	6.38	19.89	15.12	47	94	119	403	0.0187	0.9299	0.0663	0.2191
misex3c	13.76	32.17	14.06	221	288	297	704	0.0892	0.5597	0.1017	0.3191
cps	23.72	111.03	189.37	957	1470	1306	2,871	0.3077	1.9195	0.4537	1.0273
i9	44.27	65.11	65.43	1918	1785	4663	15,575	0.3250	5.9688	1.0391	1.7016
i10	5.01	24.84	23.53	883	1517	813	8,166,267	0.2977	8.2351	0.5137	1,267.4235
Total	213.98	582.53	516.45	5365	8935	10238	13,361,458	1.5827	42.7281	3.6815	2,088.2647

in guiding PowerShake. First, we notice that PowerDrive has successfully guided PowerShake to reduce the power of **t481** by 95%. Second PowerShake was able to synthesize all the circuits where POSE (which uses global BDD to estimate power of newly created nodes during synthesis) could not optimize **o64, i4, misex3c** after 10 hours. Since PowerDrive uses local BDDs based on the intermediate support-set, it could estimate power of those nodes while optimization in PowerShake. Finally, we can see, certain circuits, (**i3, C432, comp**), optimized by POSE consume more power than the initial circuit. This is due to the inaccurate power estimate based on immediate fanins of nodes in POSE. PowerDrive, which considers certain amount of local correlation, does not suffer from the problem. Hence we can conclude that PowerDrive is an efficient and accurate power estimation tool to guide power optimization.

7 Summary

In this paper, we proposed a probability-based power metric for guiding a low power driven synthesis tool. We formally defined the problem of finding the optimally-correlated L-feasible frontier. We also showed experimentally that solving this problem exactly provides good estimate of the probability. We also provided a heuristic solution, **PowerDrive** to the above problem with **constant** complexity. PowerDrive is **canonical**, a property very essential for a power metric used by a power optimization tool. We showed experimentally on 50 benchmark circuits that our approach is at least **two times** faster and **28%** more accurate than the best known approach [7]. PowerDrive was able to guide our low-power driven logic synthesis tool [11] to optimize large circuits which could not be synthesized by previous reported approaches [6], thus proving the effectiveness of our power metric.

References

[1] $BuildGates$. Ambit Design Systems, Santa Clara, CA, 1998.

[2] H. Choi and S. H. Hwang. Improving the accuracy of support-set finding method for power estimation of combinational circuits. In *Proceedings of the International Symposium on Circuits and Systems*, Hongkong, June 1997.

[3] J. Cong and Y. Ding. FlowMap: An optimal technology mapping algorithm for delay optimization in lookup-table based FPGA designs. *IEEE Transactions on Computer Aided Design*, 13(1):1–12, Jan. 1994.

[4] A. Ghosh, S. Devadas, K. Keutzer, and J. White. Estimation of average switching activity in combinational and sequential circuits. In *Proceedings of the Design Automation Conference*, pages 253–259, Anaheim, June 1992.

[5] G. Hasteer, A. Mathur, and P. Banerjee. An efficient assertion checker for combinational properties. In *Proceedings of the Design Automation Conference*, pages 734–739, Anaheim, CA, June 1997.

[6] S. Imam and M. Pedram. Logic extraction and factorization for low power. In *Proceedings of the Design Automation Conference*, pages 248–253, San Francisco, CA, June 1995.

[7] B. Kapoor. Improving the accuracy of circuit activity measurement. In *Proceedings of the Design Automation Conference*, pages 734–739, San Diego, June 1994.

[8] R. Marculescu, D. Marculescu, and M. Pedram. Logic level power estimation considering spatiotemporal correlations. In *Proceedings of the International Conference on Computer-Aided Design*, pages 294–299, Nov. 1994.

[9] F. Najm. Transition density:a new measure of activity in digital circuits. *IEEE Transactions on Computer Aided Design*, 12(2):310–323, Feb. 1993.

[10] S. Roy. *Low-Power-Driven Synthesis Algorithms for Sequential and Combinational Circuits*. PhD thesis, University of Illinois, Urbana-Champaign, IL, 1998.

[11] S. Roy, H. Arts, and P. Banerjee. A low-power logic optimization methodology based on a fast power-driven mapping. In *Proceedings of the International Conference on Computer Design*, Austin, Texas, Oct. 1998.

[12] S. Roy, H. Arts, and P. Banerjee. PowerShake: A low power driven clustering and factoring methodology for boolean expressions. In *Proceedings of the Design Automation and Test in Europe*, pages 967–968, Paris, France, Feb. 1998.

[13] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. Sangiovanni-Vincentelli. SIS: A system for sequential circuit synthesis. Technical Report UCB/ERL M92/41, Electronics Research Laboratory, University of California, Berkeley, CA, May 1992.