Implicit Pseudo Boolean Enumeration Algorithms for Input Vector Control

Kaviraj Chopra, Sarma B. K. Vrudhula
NSF Center for Low Power Electronics
ECE Department, University of Arizona
(chopra, sarma)@ece.arizona.edu

ABSTRACT

In a CMOS combinational logic circuit, the subthreshold leakage current in the standby state depends on the state of the inputs. In this paper we present a new approach to identify the minimum leakage set of input vectors (MLS). Applying a vector in the MLS is known as Input Vector Control (IVC), and has proven to be very useful in reducing gate oxide leakage and sub-threshold leakage in standby mode of operation. The approach presented here is based on *Implicit Enumeration* of integer-valued decision diagrams. Since the search space for minimum leakage vector increases exponentially with the number of primary inputs, the enumeration is done with respect to the minimum balanced cut of the digraph representation of the circuit. To reduce the switching power dissipated when the inputs are driven to a given state (during entry into and exit from the standby state), we extend the MLS algorithm to compute a bounded leakage set (BLS). Given a bound of standby leakage, we present an algorithm for computing minimal switching cost partial input vectors such that the leakage of the circuit is always less than the upper bound.

Categories and Subject Descriptors: B.7.1 VLSI

General Terms: Algorithms, Design, Theory, Performance.

Keywords: CMOS, Leakage, Power, Binary Decision Diagrams,Symbolic Methods, SAT.

1. INTRODUCTION

The reduction of threhold voltage, gate length and gate oxide thickness leads to an exponential increase in the leakage (subthreshold leakage, gate tunnelling leakage and band-to-band junction leakage). This is a significant problem in portable battery powered systems as they are often in the standby mode for long periods of time, and the leakage currents drain the battery charge even when no useful work is being performed. The subthreshold and gate oxide leakage of a CMOS gate vary by more than an order of magnitude over the set of input states. One method to reduce leakage during standy mode is to apply an input vector for which the leakage is minimum. This technique is referred to as input vector control (IVC) [12].

Implementing IVC requires efficient algorithms to compute one or more of the minimum leakage vectors. However, using IVC to minimize standby leakage results in power consumption due to switching that is proportional to the number of inputs being driven while entering or exiting the standby state. Switching power consumption can be reduced by decreasing the number of inputs being driven during IVC (i.e., the others are don't cares). If the set of all minimum leakage vectors is known, then the minimal switching cost cube contained in this set gives a *partial vector*. In addition, for a small increase in standby leakage, a larger set of input states, or equivalently a smaller partial vector, can be found. This further minimizes the switching power consumption at the cost of a increase in standby leakage. Thus, there exists tradeoff between the optimality of the minimum leakage state and the switching cost of entering and exiting this state. However evaluating this tradeoff, again requires algorithms to compute the set of leakage vectors.

The subject of leakage analysis and reduction has received significant attention over the past few years First, the problem of finding the exact minimum leakage and minimum leakage set (MLS) for an arbitrary circuit is NP-complete [4]. A a result, a large number of heuristics have been proposed. A random simulation based approach was proposed in [12]. In [4], the authors presented a constraint graph based approach. But exact solutions for circuits with only a small number of inputs was possible. An explicit branch and bound enumeration technique is described in [13]. For large circuits, bounds on the minimum and maximum leakage values were obtained using heuristics. In [10] an interesting ATPG based approach was introduced to compute the minimum and maximum leakage. Again, only bounds were possible for larger circuits. Recent works ([1, 9]) on leakage analysis attempt to leverage the advances in the solutions to SAT problems. In [1] a SAT solver is used to perform an iterative search. Leakage values were quantized into 32 and 64 levels to tradeoff accuracy for runtime. Similarly, a pseudo Boolean SAT solver is used in [9] to find the minimum leakage.

In this paper we present algorithms to perform accurate leakage analysis. The proposed approach is based on pseudo boolean enumeration using integer valued decision diagrams ([15], [7], [3]). The first algorithm the INPSPENUM, implicitly enumerates the circuit leakage with respect to its inputs. Similar to all previous approaches INPSPENUM ceases to be practical for circuits having large number of inputs. We present hypergraph partitioning based recursive procedure called MINCUTENUM for larger circuits. It performs the enumeration in min-cut space of the circuit hypergraph without compromising the optimality. This is in contrast to the existing methods ([1, 4, 10, 13]) which apply approximations to deal with larger circuits. The method presented here successfully computed the exact minimum and maximum of nearly all the benchmark circuits, most of which were not solved by the previous accurate method [9]. Moreover, with the goal to minimize switching cost of IVC, we present the algorithm FINDBLS to compute a *Bounded Leakage Set* (BLS) - a set of input vectors that result in a leakage $\leq \beta\times$ the minimum leakage, where β is the given normalized upper bound of standby leakage. Once BLS is computed, the *maximal cube* contained in BLS gives the *minimal switching cost* partial vector.

2. NOTATION & TERMINOLOGY

2.1 Pseudo Boolean Functions & Relations

1. $B = \{0,1\}$ is the Boolean space of two elements; X, Y are two sets of Boolean vectors, and Z is the set of integers.
2. A function $f : B^{|X|} \rightarrow B^{|Y|}$ is a Boolean function. A relation $\mathcal{F} \subseteq B^{|X|} \times B^{|Y|}$ is a Boolean relation.
3. The characteristic function of a Boolean relation $\mathcal{F}$ is a Boolean function $\chi_{\mathcal{F}} : B^{|X|} \times B^{|Y|} \rightarrow B$ defined as
$$\chi_{\mathcal{F}}(x,y) = \begin{cases} 1 & \text{if } (x,y) \in \mathcal{F} \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$
4. A function $p : B^{|X|} \rightarrow Z$ is called a pseudo Boolean function (PBF). A pseudo Boolean relation (PBR) and its characteristic function are similarly defined.
5. Let $X = (x_1, \ldots, x_n)$. The *positive cofactor* of a (pseudo) boolean function $p(x_1,\ldots,x_n)$ w.r.t. literal x_i is a (pseudo) boolean function $p_{x_i} = p(x_1,\ldots,x_{i-1},\boldsymbol{x_i} = \mathbf{1}, x_{i+1},\ldots,x_n)$. The *negative cofactor*, denoted by $p_{\overline{x_i}}$, is similarly defined.

Boolean functions and relations can be efficiently represented and manipulated using Reduced Ordered Binary Decision Diagrams ROBDDs [6, 5]. Similarly, pseudo Boolean functions and relations can be efficiently represented and manipulated using one of several integer valued decision diagrams (e.g. ADD [3], EVBDD [15], MTBDD [7]). In this paper, we will often use the terms function and its relation interchangeably, with the understanding any relation is represented by the appropriate decision diagram of its characteristic function. **Note:** We assume that the reader is familiar with the theory and operation of decision diagrams [19].

2.2 Leakage Relation

The leakage of a subcircuit, which we assume is non zero, depends on the state of its inputs. Thus, a non zero integer leakage value can be associated with each input minterm of a subcircuit. Hence the leakage of a subcircuit with inputs U is a PBF $\lambda : B^{|U|} \rightarrow Z$. Figure 1 shows an example circuit and the leakage function $\lambda_{C_3}(u_1,u_2)$ of gate 3 as a function of its local inputs. Algebraically, $\lambda_{C_3}(u_1,u_2) = 9 + 26u_1 + 41u_2 + 260u_1u_2$.

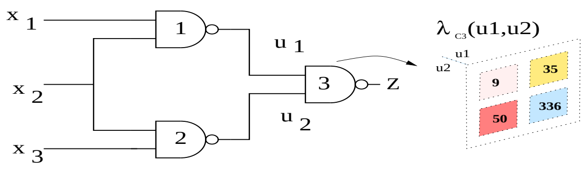

Figure 1: Example Circuit Leakage Relation of C_3

2.3 The Operator MINABSTRACT

DEFINITION 1. *Let $X = (x_1,\ldots,x_n)$, and $\lambda : B^{|X|} \rightarrow Z$ be a PBF. The operator* MINABSTRACT *w.r.t the variable x_i is denoted by $\exists^{min}(\lambda, x_i)$, computes a PBF and is defined as*
$$\exists^{min}(\lambda, x_i) = min(\lambda_{x_i}, \lambda_{\overline{x_i}}). \quad (2)$$

Note: In a leakage relation, any minterm for which the leakage is zero means that the minterm is in the offset. Therefore, prior to performing operations in the arithmetic domain, the *logic zero* must be replaced by an *integer zero* or $+\infty$, as appropriate.

Figure 2 shows an example of MINABSTRACT applied to a three variable PBF. Given a PBF,MINABSTRACT w.r.t. x minimizes the function in x direction. Given a characteristic function of a pseudo boolean relation MINABSTRACT computes the minimum projection on subspace $B^{|X-x|}$ orthogonal to sub-space $B^{|x|}$. The operator MINABSTRACT is a generalization of the existential quantification operator [8] applied to the pseudo Boolean domain. Consequently, it is no surprise that MINABSTRACT can be computed based on the Shannon decomposition of a Boolean function and using a procedure similar to the ITE procedure [5]. The operator MAXABSTRACT can be defined similarly.

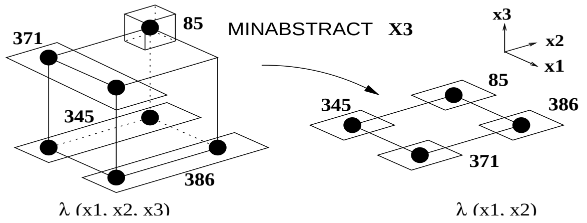

Figure 2: MINABSTRACT($\lambda(x_1,x_2,x_3),x_3$) $\rightarrow$ $\lambda(x_1,x_2)$

3. INPUT SPACE ENUMERATION

Let C denote a combinational circuit with primary inputs $X = (x_1, x_2, \ldots x_n)$. C is a netlist of primitive subcircuits (gates or cells). Let C_i be such a subcircuit with local inputs $U = (u_1,\ldots,u_j)$. We assume that the leakage of each primitive subcircuit C_i for each of its input states is given. This is represented as a leakage relation $\lambda_{C_i}(U)$ in variables U. Here we describe a procedure called INPSPENUM that computes the leakage relation $\lambda_C(X)$ of C, given the leakage relations of its subcircuits. The steps of this procedure are shown in Algorithm 1. The method is based on two basic operations: (1) leakage space transformation and (2) leakage addition. **Note:** In the algorithms to be presented, the decision diagrams are shown with arguments which are the set of variables in the diagram.

Algorithm 1: INPSPENUM(C)

```
1: Input: Circuit $C$ with subcircuits $\{C_1, C_2 \ldots\}$. Inputs $C_i$ is $U_i$
2: $\lambda_C(X) \leftarrow 0$
3: for all $C_i \in C$ do
4:     $\lambda_{C_i}(X,U_i) \leftarrow \lambda_{C_i}(U_i) \times \chi_C(X,U_i)$;
5:     $\lambda_{C_i}(X) \leftarrow \exists^{min}_{U_i}\lambda_{C_i}(X,U_i)$;
6:     $\lambda_C(X) \leftarrow \lambda_C(X) + \lambda_{C_i}(X)$;
7: end for
8: Return($\lambda_C(X)$)
```

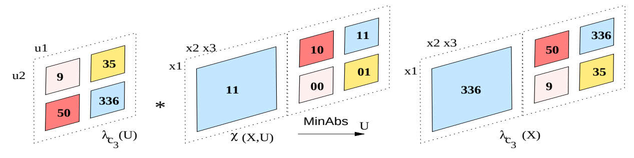

Figure 3: Leakage Space Transformation $\lambda_{C_3}(U) \rightarrow \lambda_{C_3}(X)$

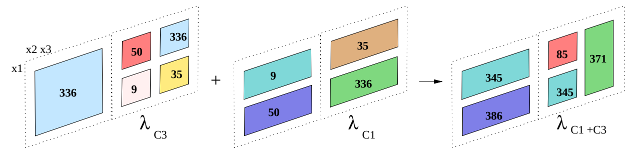

Figure 4: Leakage Addition by Partition Refinement

3.1 Leakage Space Transformation

Consider the Boolean function $f : B^{|X|} \rightarrow B^{|U|}$ between circuit's primary input space $B^{|X|}$ and local input space $B^{|U|}$ of some subcircuit C_i. The leakage relation $\lambda_{C_i}(U)$ is given in local input space $B^{|U|}$ and we want to compute its leakage relation $\lambda_{C_i}(X)$ in the primary input space $B^{|X|}$. Each point of $B^{|U|}$ corresponds to a set of points in $B^{|X|}$. Therefore the local input space of a subcircuit with $|U|$ inputs, partitions the primary input space into at most $2^{|U|}$ disjoint sets. Since a leakage value is associated with every point in the local input space, the leakage of sub-circuit C_i also partitions the primary input space into at most $2^{|U|}$ disjoint sets. Therefore by mapping the leakage of every point in local input space $B^{|U|}$ to the corresponding set in primary input space $B^{|X|}$, the leakage of C_i in primary input space $B^{|X|}$ can be computed. The relation between the primary input subsets in $B^{|X|}$ and every point in $B^{|U|}$ is represented by the Boolean relation $\chi(X, U)$. The product $\chi(X, U) \times \lambda_{C_i}(U)$ results in a leakage relation $\lambda_{C_i}(X, U)$ (see step 4 of Algorithm 1). It represents the leakage relation of C_i in terms of the primary inputs X and local inputs U. Now to obtain the leakage relation $\lambda_{C_i}(X)$, the variables in U are *abstracted away* by computing the projection $\lambda_{C_i}(X, U_i) \rightarrow \lambda_{C_i}(X)$ using MINABSTRACT. This is shown in step 5 of Algorithm 1). Figure 3 illustrates the leakage space transform operation applied to subcircuit C_3 of the circuit in Figure 1. **Note:** Although the leakage space transform operation is explained here w.r.t to the input space, it can be carried out w.r.t any set of signals in the circuit.

3.2 Leakage Addition

We use a partition refinement technique to add two leakage relations λ_{C_1} and λ_{C_2}. Recall that a leakage relation is a set of disjoint leakage partitions. First, a cross-product of two sets (the two leakage relations) is performed and new refined partitions are formed by intersecting every partition pair in λ_{C_1} and λ_{C_2}. The leakage of the resulting refined partitions is obtained by adding the leakages of the two intersecting partitions. This partition refinement (+ operation in step 6) is performed implicitly by using the Shannon decomposition based APPLY procedure with algebraic addition as the terminal case. Figure 4 illustrates the partition refinement approach for the example circuit of Figure 1.

By repeating these two steps for all gates in C, the leakage of C for every input is computed without explicitly computing leakage for every input vector. While adding two leakage relations, the number of leakage partitions increases after every partition refinement step (i.e. + operation). The total number of partitions possible in the worst case is bounded by $2^{|X|}$. Since the addition operation is performed once for every gate, the runtime complexity of the above algorithm is $O(|C|.2^{|X|})$, where $|C|$ is the number of gates or sub-circuits in C. Actual runtimes are much better than indicted by this order due to the efficient representation of integer valued decision diagrams. Nevertheless, this approach ceases to be practical for circuits with a large number of inputs.

4. MIN-CUT SPACE ENUMERATION

For larger circuits, we use a divide and conquer approach. Each of the smaller sub-problems is solved independently and the resulting solutions are merged without compromising accuracy. In [8, 20] this is referred to as a *partitioned transition relation* method. Several heuristics have been proposed for the above method. In [11] a separator-set induced partitioning approach for symbolic state space exploration and reachability analysis is described. We use a similar hypergraph partitioning approach for leakage enumeration. The given circuit net-list is represented by a hypergraph where the vertices of the hypergraph are gates and the hyperedges are nets. A Min-Cut bi-partitioning of this hypergraph is performed so that both number of circuit inputs and the number of gates are balanced. If the input space (i.e. the number of input variables) and/or the circuit size (i.e. the number of gates) of the partitioned sub-circuits is more than a specified threshold size, the partitioning is repeated recursively. As a result, a binary tree called *Partition Tree* is formed. The root of the tree is C and child node represents one of the two sub-circuits that resulted from partitioning its parent.

Algorithm MINCUTENUM successively performs the *minimum leakage enumeration* and *bounded leakage enumeration* in a bottom up manner at every node of the partition tree. This is followed by Algorithm FINDBLS which traverses the partition tree in a top down manner to compute the BLS in terms of the primary inputs of the circuit. Finally, the minimal switching cost cube contained in BLS is computed.

4.1 Minimum Leakage Enumeration

The goal of minimum leakage enumeration (Algorithm 2) to enumerate the leakage of the circuit in the min-cut space. Since the input space of leaf node sub-circuits is small, their input leakage relation can be efficiently computed using procedure INPSPENUM. The goal here is to find the global minimum leakage of the complete circuit, which is achieved by adding the leakage of all leaf node sub-circuits recursively. With every node P of the partition tree, we can identify four sets of nets(hyperedges).

1. E_{LR} are the nets in P that were *cut* at an ancestor of P.

2. E_C are the nets in P that were not cut at an ancestor of P but were cut at P, i.e. as a result of bi-partitioning P.

3. $E_L \subseteq E_{LR}$ are nets in P that are also included in P_L, the left child of P.

4. $E_R \subseteq E_{LR}$ are nets in P that are also included in P_R, the right child of P.

Consider the case of a leaf node P. Recall that every leaf node sub-circuit P is separated from the rest of the circuit by nets E_{LR} - the subset of nets that were cut at an ancestor of P. Therefore we need to map its input leakage relation to the ancestor cut space of variables E_{LR}. The dependencies between the variables in X_P and those in E_{LR} are described by $\chi_P(X_P, E_{LR})$. This is conjoined with the leakage relation $\lambda_P(X_P)$. Then the local variables X_P are abstracted way, resulting in the leakage relation of P being expressed in its parent subspace, i.e. $\lambda_{P+}(E_{LR})$ (step 3 of Algorithm 2). This is returned to the parent node.

Now consider the case where P is a non-leaf node. P is the union of left sub-circuit P_L and right child sub-circuit P_R. The left child

Algorithm 2: MINCUTENUM(P,β)

```
1: ZERO = 0
2: if $P$ is a leaf node then
3:     $\lambda_P(X_P) \leftarrow$ INPSPENUM($P$)
4:     $\lambda_P(X_P, E_{LR}) \leftarrow \lambda_P(X_P) \times \chi_P(X_P, E_{LR})$
5:     ZERO = $+\infty$
6:     $\lambda_{P^+}(E_{LR}) \leftarrow \exists^{min}_{X_P - E_{LR}}(\lambda_P(X_P, E_{LR}))$
7:     $\chi^e_P(X_P, E_{LR}) \leftarrow (\lambda_P(X_P, E_{LR}) \leq \lambda_{P^+}(E_{LR}) \times \beta)$
8:     ZERO = 0
9:     $\lambda^b_{P^+}(E_{LR}) \leftarrow \exists^{max}_{X_P - E_{LR}}(\lambda_P(X_P) \times \chi^e_P(X_P, E_{LR}))$
10: else
11:     $\lambda_{P_L}(E_L, E_C), \lambda^b_{P_L}(E_L, E_C) \leftarrow$ MINCUTENUM($P_L, \beta$)
12:     $\lambda_{P_R}(E_R, E_C), \lambda^b_{P_R}(E_R, E_C) \leftarrow$ MINCUTENUM($P_R, \beta$)
13:     $\lambda_P(E_{LR}, E_C) \leftarrow \lambda_{P_L}(E_L, E_C) + \lambda_{P_R}(E_R, E_C)$
14:     ZERO = $+\infty$
15:     $\lambda_{P^+}(E_{LR}) \leftarrow \exists^{min}_{E_C}(\lambda_P(E_{LR}, E_C) \times \chi(E_{LR}, E_C))$
16:     $\chi^e_P(E_{LR}, E_C) \leftarrow (\lambda_P(E_{LR}, E_C) \leq \lambda_{P^+} E_{LR}) \times \beta)$
17:     $\lambda^b_P(E_{LR}, E_C) \leftarrow \lambda^b_{P_L}(E_L, E_C) + \lambda^b_{P_R}(E_R, E_C)$
18:     ZERO = 0
19:     ${\lambda_{P^+}}^b(E_{LR}) \leftarrow \exists^{max}_{E_C - E_{LR}}(\lambda_P(E_C, E_{LR}) \times \chi^e_P(E_C, E_{LR}))$
20: end if
21: Return($\lambda_{P^+}(E_{LR}), \lambda^b_{P^+}(E_{LR})$)
```

returns the minimum leakage relation $\lambda_{P_L}(E_L, E_C)$. The variables involved are the nets cut at P (i.e. E_C) and the subset of nets that were cut at an ancestor of P (i.e. E_{LR}), that were passed to the left child of P (i.e. E_L). Similarly the right child of P returns the relation $\lambda_{P_R}(E_R, E_C)$. The addition of the two child relations using $+$ operation gives the minimum leakage relation $\lambda_P(E_{LR}, E_C)$ of circuit P. **Note:** The child leakage relations $\lambda_{P_L}(E_L, E_C)$ and $\lambda_{P_R}(E_R, E_C)$ were both computed ignoring the logic dependencies among them due to the common nets in $E_C \cup (E_L \cap E_R)$. Therefore the sum $\lambda_P(E_{LR}, E_C)$ is conjoined with Boolean relation $\chi_P(E_{LR}, E_C)$ (step 10 of Algorithm 2). Now the non-leaf node P is connected to the rest of the circuit only through nets E_{LR}. Consequently, the internal nets E_C, (the ones cut at P,) are *abstracted away* by MINABSTRACT, resulting in the minimum leakage relation $\lambda_{P^+}(E_{LR})$. This is returned to the parent node and the enumeration procedure is successively repeated at every non-leaf node. Finally, at the root node, the leakage of the complete circuit is obtained and since $E_{LR} = \phi$, the minimum leakage relation λ_{P^+} is a constant i.e., the minimum leakage value of the circuit. Similarly by replacing the MINABSTRACT with MAXABSTRACT the maximum leakage of the circuit can also be computed.

4.2 Bounded Leakage Enumeration

Algorithm 2 also performs bounded leakage enumeration while traversing the partition tree. This is used to identify the set of input vectors that will result in a leakage that is at most a given percentage above the minimum leakage. Let β denote the upper bound on the leakage normalized w.r.t to the minimum leakage.

Consider a leaf node P in the partition tree. Let $x \in B^{|X_P|}$, $v \in B^{|E_{LR}|}$. In step 4, we compute $\lambda_P(x, v)$, which is the leakage of P at x, but assigned to every satisfiable point (x, v). Note that each point $v \in B^{|E_{LR}|}$ corresponds to a set of points $X \subseteq B^{|X_P|}$. As we proceed up the partition tree, the variables in X_p and not in E_{LR} are abstracted away. The result (step 6) is $\lambda_{P^+}(E_{LR})$. At each point $v \in B^{|E_{LR}|}$, this relation represents the minimum leakage of P over the corresponding set X. Therefore, at each such point v, $\lambda_{P^+}(E_{LR}) \times \beta$ is the maximum allowable leakage value beyond the minimum. We now need to identify those points (x, v) in the space $B^{|X_P \times E_{LR}|}$ where the leakage value is less than or equal to the $\lambda_{P^+}(v) \times \beta$. These points are identified by the *enumeration relation* $(\lambda_P(X_P, E_{LR}) \leq \lambda_{P^+}(E_{LR}) \times B)$, which is denoted by $\chi^e_P(X_P, E_{LR})$ (step 7). The operation $\leq$ is again computed using the Shannon decomposition based APPLY procedure.

Now for computing the enumeration relation at the non-leaf node, we must know the maximum leakage relations of sub-circuits P_L and P_R amongst all vectors belonging to their enumeration relation. Hence the last step involves abstracting away the variables in X_P and keeping only the variables in E_{LR}, i.e. those the parent subspace. This gives the *bounded leakage relation* ${\lambda_P}^b(E_{LR})$. The bounded leakage relation ${\lambda_P}^b(E_{LR})$ is the maximum projection of the subset of input leakage relation having leakage less than or equal to the upper bound relation $\lambda_P(E_{LR}) \times B$.

At the non-leaf node, similar to the minimum leakage relation computation, first the bounded leakage relation ${\lambda_P}^b(E_C, E_{LR})$ of P is computed by adding the left child bounded leakage relation ${\lambda_{P_L}}^b(E_C, E_L)$ and the right child bounded leakage relation ${\lambda_{P_R}}^b(E_C, E_R)$. Next, the *enumeration relation* $\chi^e_P(E_C, E_{LR})$ is computed by evaluating the inequality $\lambda^b_P(X_P, E_{LR}) \leq \lambda_P(E_{LR}) \times B$. Note that the bounded leakage relation λ^b_P is compared with upper bound leakage relation $\lambda_P(E_{LR}) \times B$. Similar to the case of a leaf node, the new bounded leakage relation $\lambda^b_{P^+}(E_{LR})$ for the parent node enumeration is again obtained by conjoining the bounded leakage relation $\lambda^b_P(X_P, E_{LR})$ with the enumeration relation $\chi^e_P(E_C, E_{LR})$ and then abstracting away internal nets E_C from the resulting relation using MAXABSTRACT operation. The bounded leakage relation $\lambda^b_{P^+}(E_{LR})$ is returned to the parent node for further enumeration.

The runtime of the algorithm is dominated by the APPLY procedure that is used to compute $+$, $\leq$ and $\times$ operations. Because it is difficult to bound the size of decision diagram at every partition node for an arbitrary circuit graph, a tight upper bound on runtime of the algorithm is generally not possible. While the worst case runtime of the algorithm is exponential in the size of largest support set over all nodes in partition-tree (i.e. $max_P(E_C, E_{LR})$), in practice, the average runtime is actually much smaller. In fact, $|E_C|$ for many benchmark circuits with a large number of inputs is very small (e.g. for C7552 $|X| = 207$, $\max_P(E_C, E_{LR}) = 34$).

4.3 Bounded Leakage Set Computation

We now describe the algorithm FINDBLS to compute the BLS. Note if $\beta = 1$ then the algorithm computes the minimum leakage set. During bottom up traversal of MINCUTENUM, the enumeration relation χ^e_P - boolean relation between λ_P and λ_{P^+}, was computed at every node P. Now, at the root node the enumeration relation $\chi^e_P(E_C)$ is equal to the BLS in E_C variables. However, for implementing input vector control the goal is to find BLS in circuits primary input space. The algorithm FINDBLS first recursively computes the child space BLS from BLS given in the root node min-cut space in order to obtain the BLS in terms of primary inputs for each subcircuit. Thereafter the child space BLS are conjoined together in a bottom up manner, similar to the MINCUTENUM procedure, to compute the BLS of the complete circuit in terms of the primary inputs. The pseudo code of FINDBLS can be found in Algorithm. 3. Figures.5(a) and (b) illustrate the action of FINDBLS at the non leaf node for both the top down and bottom up traversals respectively. The parent space BLS and cut space BLS are denoted by the characteristic functions $\chi^v_{P^+}$ and χ^v_P respectively. As $E_{LR} = \phi$ for the root node, algorithm FINDBLS is called with $\chi^v_{P^+} = 1$, (initialized as the universal set).

Consider the top down recursive case when P is a non-leaf node. The parent BLS $\chi^v_{P^+}(E_{LR})$ is conjoined with the enumeration relation $\chi_P^e(E_{LR}, E_C)$ (step 4 of Algorithm. 3). At this point the BLS we seek is expressed in terms of the cut-variables at or above P. Ultimately the goal is to express this in terms of the input variables of the circuit. The cut-variables of the left child of P are E_L, E_C. We abstract away all the variables in E_{LR} except those in E_L . This results in $\chi_{P_L}^v(E_L, E_C)$, the BLS expressed in left child cut-variables (step 5 of Algorithm. 3). The same is repeated for the right child of P (step 6 of Algorithm. 3). The resulting BLS $\chi_{P_L}^v(E_L, E_C)$ and $\chi_{P_R}^v(E_R, E_C)$ are recursively passed to the left and right child nodes respectively. Similarly at leaf node P, the parent BLS $\chi_{P+}^v(E_{LR})$ is conjoined with enumeration relation $\chi_P^e(E_{LR}, X_P)$ (step 2 of Algorithm. 3). The resulting primary input variable BLS $\chi_P^v(E_{LR}, X_P)$ of the leaf node sub-circuit is returned to its parent node.

Algorithm 3: FINDBLS $(P, \chi_{P+}^v(E_{LR}))$

```
1:  if $P$ is leaf Node then
2:      $\chi_P^v(E_{LR}, X_P) \leftarrow \chi_P^e(E_{LR}, X_P) \times \chi_{P+}^v(E_{LR})$
3:  else
4:      $\chi_P^v(E_{LR}, E_C) \leftarrow \chi_P^e(E_{LR}, E_C) \times \chi_{P+}^v(E_{LR})$
5:      $\chi_{P_L}^v(E_L, E_C) \leftarrow \exists_{E_{LR}-E_L}(\chi_P^v(E_{LR}, E_C))$
6:      $\chi_{P_R}^v(E_R, E_C) \leftarrow \exists_{E_{LR}-E_R}(\chi_P^v(E_{LR}, E_C))$
7:      $\chi_{P_L}^v(E_L, E_C, X_{P_L}) \leftarrow$ FINDBLS$(P_L, \chi_{P_L}^v(E_L, E_C))$
8:      $\chi_{P_R}^v(E_R, E_C, X_{P_R}) \leftarrow$ FINDBLS$(P_R, \chi_{P_R}^v(E_R, E_C))$
9:      $\chi_P^v(E_{LR}, E_C, X_P) \leftarrow \chi_{P_L}^v(E_L, E_C, X_{P_L}) \times \chi_{P_R}^v(E_L, E_C, X_{P_L})$
10:     $\chi_P^v(E_{LR}, X_P) \leftarrow \exists_{E_C - X}(\chi_P^v(E_{LR}, E_C, X_P) \times \chi_P^v(E_{LR}, E_C))$
11: end if
12: Return$(\chi_P^v(E_{LR}, X_P))$
```

After computing the BLS of every leaf node subcircuit we need to conjoin these BLSs to compute the BLS of the complete circuit. At every non leaf node the two child nodes P_L and P_R return the two BLSs $\chi_{P_L}^v(E_L, E_C, X_{P_L})$ and $\chi_{P_R}^v(E_R, E_C, X_{P_R})$ respectively (step 7 and 8 of Algorithm. 3). The variables X_{P_L} and X_{P_R} are the primary inputs contained in left and right child subcircuits. These two child node BLSs $\chi_{P_L}^v(E_L, E_C, X_{P_L})$ and $\chi_{P_R}^v(E_R, E_C, X_{P_R})$ are conjoined together to compute BLS $\chi_P^v(E_{LR}, E_C, X_P)$ at P (step 9 of Algorithm. 3). Now recall that during the top down traversal, existential quantification was performed at every non-leaf node for computing the two child space BLS (step 7 and 8). Consequently there may exist some vectors in conjunction BLS $\chi_P^v(E_{LR}, E_C, X_P)$ which originally didn't belong to the cut space BLS $\chi_P^v(E_{LR}, E_C)$. Therefore, the resulting BLS $\chi_P^v(E_{LR}, E_C, X_P)$ is further conjoined with the previously computed BLS $\chi_P^v(E_{LR}, E_C)$ and thereafter the internal cut variables are abstracted away. We abstract away all the variables in E_C except those in X, the set of primary input variables (step 10 of Algorithm. 3). This results in $\chi_P^v(E_{LR}, X_P)$, the BLS expressed in primary input variables contained in P and ancestor nets E_{LR}. The BLS $\chi_P^v(E_{LR}, X_P)$ of the sub-circuit at P is returned to its parent node. This is repeated recursively and finally, at the root node, the BLS $\chi_P^v(X)$ of the complete circuit is obtained ($\because E_{LR} = \phi$).

After finding the BLS we wish to find the partial vector contained in the set $\chi_P^v(X)$ such that the switching cost is minimized. Now every vector in $\chi_P^v(X)$ is a input minterm of the Boolean function $\chi_P^v(X)$. Therefore, by finding the minimum switching cost cube contained in the function one can find the optimal subvector. This problem can be formulated as a binate covering problem. We use the method presented in [21] to find close to optimal partial vectors by finding shortest one path of the BDD digraph of $\chi_P^v(X)$, where the cost of each variable in $\chi_P^v(X)$ is the switching cost associated with the input.

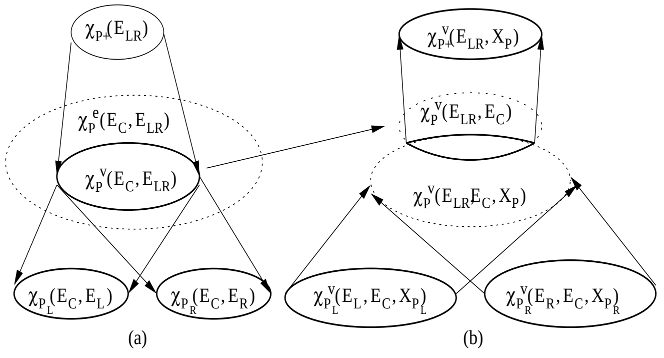

Figure 5: Bounded Leakage Set Computation

5. EXPERIMENTAL RESULTS

The algorithms presented here were tested on the MCNC91 benchmark circuits. All circuits were synthesized using SIS and were mapped to a 0.18μ library of 25 cells including inverters, two and three input nand and nor gates. This gate library was precharacterized for V_T=0.3V , $V_{dd} = 1.8V$ and $25'$C temperature. Algorithms were implemented using the Colorado University Decision Diagram CUDD package [18] and hypergraph partitioning tool HMETIS [14]. Hypergraph partitioner HMETIS was run multiple times, by varying two parameters: (1) Weights of primary input vertices relative to weights of internal vertices, and (2) Imbalance factor for bi-partitioning[2]. The maximum support size over all partition nodes was computed for each partition tree. The best candidate partition tree with the minimum number of partition nodes and minimum of the maximum support size was selected. These experiments were performed on a 1.8 GHz Pentium PC with 512 MB RAM.

Similar to [9], experiments for every benchmark circuit were conducted with a runtime upper limit of 5000s. 61 out of 76 benchmark circuits completed successfully within 5000s. Results for 15 hard benchmark examples for which runtime exceeded the limit and 39 trivial examples having inputs less than 30 are omitted due to space limitations. Leakage analysis results for the remaining 22 non trivial circuits which were successfully solved are listed in Table 1. This is a significant improvement over [9], where only 24 out of the listed 29 circuits completed the simulation. Moreover, the runtime for the proposed approach are on average an order of magnitude smaller than [9] for listed examples. On circuits like C6288 where BDD based methods are inherently limited, our approach was also found to fail. For each benchmark circuit, the total number of leaf node partitions and maximum min-cut is reported in columns 4 and 5. Columns 6 and 7 give the minimum (${I_{lk}}^{min}$) and maximum leakage current (${I_{lk}}^{max}$). The runtime for computing the minimum and maximum leakage are reported in the columns 8 and 9, which includes run time for computing the partition tree.

Results for the partial vector computation using FINDBLS algorithm are shown in Table 2. Notice that the difference between the maximum and minimum leakage varies considerably. Therefore for fair evaluation, we compare the results for bound b specified w.r.t to the leakage range. Thus for a given b, the upper bound w.r.t. the minimum leakage was computed as:

$$\beta = 1 + b\frac{({I_{lk}}^{max} - {I_{lk}}^{min})}{{I_{lk}}^{min}}$$

For ease of implementation, we assumed that all inputs have uniform switching power consumption, however the presented approach can be applied to any arbitrary values of switching power

Table 1: Results for Minimum and Maximum Leakage

Circ.	| X |	| C |	| P |	max-cut	$I_{lk}{}^{min}$(pA)	$I_{lk}{}^{max}$ (pA)	T^{min} (s)	T^{max} (s)
count	35	138	4	4	22,624	37,016	0.96	0.66
unreg	36	117	4	4	16,926	32,181	0.48	0.55
cht	47	136	4	3	13,150	28,786	0.79	0.71
i3	132	128	8	1	26,418	48,607	6.52	3.52
i6	138	491	14	6	84,202	140,923	6.39	7.71
b9	41	111	4	9	13,632	28,188	26.73	66.94
i2	201	215	16	2	9,080	101,081	17.06	12.41
apex7	49	233	5	7	30,403	55,022	10.47	11.77
i5	133	198	9	1	15,232	48,034	10.85	10.07
i4	192	168	16	1	14,339	36,875	7.88	8.46
x4	94	324	8	8	38,589	82,516	19.29	103.44
C432	36	227	5	13	30,525	55,077	62.64	29.22
term1	34	209	5	15	27,920	46,052	108.46	146.34
example2	85	300	10	15	34,212	64,497	48.44	39.44
i7	199	546	18	7	79,740	147,793	31.48	38.95
frg2	143	819	19	15	129,172	195,828	338.58	414.37
apex6	135	713	21	19	107,013	168,871	278.06	423.68
C499	41	515	7	16	88,276	105,885	429.13	521.29
C1355	41	515	7	16	88,276	105,885	520.04	549.88
C1908	33	500	8	24	79,653	107,458	1324.53	2001.1
C880	60	388	8	15	53,692	98,195	2679.98	3226.68
C7552	207	2288	31	13	383,544	480,829	972.11	1098.73

Table 2: Results for Bounded Leakage Set

	b =5 %			b =10 %			b =15 %			b =20 %		
Circ.	$\|X\|$	$\triangle$%	T(s)	$\|X\|$	$\triangle$%	T(s)	$\|X\|$	$\triangle$%	T(s)	$\|X\|$	$\triangle$%	T(s)
count	19	45.7	0.27	19	45.7	0.33	19	45.7	0.42	18	48.6	0.41
unreg	34	5.6	0.41	31	13.9	0.4	29	19.4	0.52	27	25.0	0.46
cht	43	8.5	0.48	41	12.8	0.45	38	19.1	0.6	36	23.4	0.62
i3	97	26.5	11.6	82	37.9	4.8	72	45.5	73.66	53	59.8	14.4
i6	136	1.4	1.86	127	8.0	1.59	108	21.7	3.05	98	29.0	4.34
b9	35	14.6	1.15	32	22.0	0.97	30	26.8	0.95	27	34.1	1.65
i2	108	46.3	11.14	99	50.7	9.24	98	51.2	17.07	96	52.2	25.81
apex7	43	12.2	0.85	38	22.4	1.09	33	32.7	1.36	29	40.8	1.24
i5	132	0.8	9.15	126	5.3	7.95	121	9.0	19.82	118	11.3	24.16
i4	173	9.9	3.17	172	10.4	4.01	168	12.5	7.76	156	18.8	10.31
x4	85	9.6	3.24	77	18.1	3.38	70	25.5	5.43	63	33.0	4.51
C432	31	13.9	3.79	30	16.7	3.46	27	25.0	5.33	27	25.0	5.31
term1	30	11.8	13.05	28	17.6	13.34	25	26.5	79.69	23	32.4	19.85
exmp2	80	5.9	8.77	74	12.9	8.29	67	21.2	12.19	61	28.2	11.16
i7	192	3.5	3.36	177	11.1	7.38	157	21.1	9.37	135	32.2	15.41
frg2	117	18.2	37.23	95	33.6	51.07	87	39.2	104.88	61	57.3	55.86
apex6	111	17.8	848.45	92	31.9	1322.56	82	39.3	4411.49	64	52.6	4899.98
C499	41	0.0	129.76	40	2.4	278.29	40	2.4	646.41	39	4.9	272.63
C1355	41	0.0	170.41	40	2.4	376.94	40	2.4	234.79	38	7.3	157.93
C1908	30	9.1	1085.15	30	9.1	1702.66	28	15.2	1966.37	28	15.2	4317.45
C880	52	13.3	3895.83	49	18.3	4587.86	48	20.0	4861.64	42	30.0	4923.86
C7552	191	7.7	2520.1	191	7.7	2619.42	191	7.7	3257.09	178	14.0	3326.77

consumption. Because of this simplification, the switching power cost for implementing IVC is directly proportional to the cardinality of the partial vector. Cardinality of partial input vector $|X|$, for b = 5%, 10%, 15% and 20%, are listed in column 2, 5, 8 and 11 respectively and corresponding switching power minimization $\triangle$%, is given in columns 3, 6, 9 and 12, respectively. Runtime for each case is reported in column 4, 7, 10 and 13 respectively. Notice how $\triangle$% increases considerably w.r.t b for circuits *frg2*, *apex6*, *i3* and *C1908* while, it hardly changes for *count*, *C499* and *C1355*. The, proposed algorithms can hence provide designers with a mechanism to evaluate and choose right tradeoff for implementing IVC.

6. CONCLUSION

In this paper we have introduced a new operator called MINABSTRACT to perform minimum existential quantification for a pseudo Boolean functions. Using this operator and integer valued decision diagrams, we have developed new implicit enumeration technique for pseudoBoolean functions. We apply this implicit enumeration approach to perform accurate leakage analysis. Experimental results demonstrate significant improvements over previously published pseudo Boolean SAT based approach.[9] We also present algorithms for computing the bounded leakage set of a circuit. This bounded leakage set is then used to compute the minimal switching cost partial vectors by solving the covering problem. Experimental results indicate that cardinality of partial vector, and hence the switching power can be reduced up to 60 % for an increase in standby leakage of about 20%.

ACKNOWLEDGEMENTS This work was supported by the NSF Center for Low Power Electronics Grant #EEC-9523338, and an NSF ITR Grant #CCR-0205227.

7. REFERENCES

[1] A. Abdollahi, et al., "Runtime mechanisms for leakage current reduction in CMOS VLSI circuits," in *Proc. of ISLPED.*, 2002.

[2] C. J. Alpert, et al., "Recent directions in netlist partitioning: A survey," in *.VLSI Journal*, 1995.

[3] R. I. Bahar, et al., "Algebraic decision diagrams and their applications," in *Proc. of IEEE ICCAD.*, 1993.

[4] S. Bobba , et al., "Maximum Leakage Power Estimation for CMOS Circuits," in *Proc. IEEE AVWLPD.*, 1999.

[5] K. S. Brace, et al., "Efficient implementation of a BDD package," in *Proc. ACM/IEEE DAC.*, 1990.

[6] R. Bryant, "Graph-based algorithms for boolean function manipulation," in *IEEE Trans. on Computers.*, 1986.

[7] E. Clarke, et al., "Multi-terminal BDDs: An efficient data structure for matrix representation," in *Proc. of (IWLS).*, 1993.

[8] O. Coudert, et al., "A Unified Framework for the Formal Verification of Sequential Circuits," in *Proc. of IEEE ICCAD.*, 1990

[9] A. Fadi, et al., "Robust SAT-Based Search Algorithm for Leakage Power Reduction," in *Proc. of PATMOS.*, 2002.

[10] A. Ferr, et al., "Leakage Power Bounds in CMOS Digital Technology," in *IEEE Trans of CAD.*, 2002

[11] A. Gupta, et al., "Partition Based Decision Heuristics for Image Computation Using SAT and BDDs," in *Proc. of IEEE ICCAD.*, 2001

[12] J. Halter, et al., "A Gate-Level Leakage Power Reduction Method for Ultra-Low-Power CMOS Circuits," in *Proc. of IEEE CICC.*, 1997

[13] M. Johnson, et al., "Models and Algorithms for Bounds on Leakage in CMOS Circuits," in *IEEE Trans of CAD.*, 1999.

[14] G. Karypis, et al., "Multilevel Hypergraph Partitioning: Applications in VLSI Design," in *ACM,IEEE Proc. of DAC*, 1997.

[15] Y.-T. Lai et al., "Edge-valued binary decision diagrams for multi-level hierarchical verification," in *ACM,IEEE Proc of DAC*, 1992.

[16] D. Lee, et al., "Analysis and Minimization Techniques for Total Leakage Considering Gate Oxide Leakage," in *ACM,IEEE Proc. DAC*, 2003.

[17] K. Roy, et al., "Leakage Current Mech. and Leakage Reduction Techniques in Deep-Submicron CMOS Circuits ," in *IEEE Trans of CAD.*, 2003.

[18] F. Somenzi,"CUDD: CU Decision Diagram Package, 2.3.1.," *ECE Dept., Univ. of Colorado, Boulder.*

[19] "International Journal of Software Tools for Technology Transfer," in *Special issue on Decision Diagrams. Springer-Verlag.*, 2001.

[20] H. Touati et al., " Implicit state enumeration of finite state machines using BDDs," in *Proc. of IEEE ICCAD.*, 1990.

[21] T. Villa, et al., "Explicit and Implicit Algorithms for Binate Covering Problems . in *IEEE Trans of CAD.*, 1997.