The Approximation Scheme For Peak Power Driven Voltage Partitioning

Jia Wang, Xiaodao Chen, Chen Liao, Shiyan Hu
Department of Electrical and Computer Engineering
Michigan Technological University
Houghton, Michigan 49931
Email: {jiaw, cxiaodao, cliao, shiyan}@mtu.edu.

***Abstract*— With advancing technology, large dynamic power consumption has significantly limited circuit miniaturization. Minimizing peak power consumption, which is defined as the maximum power consumption among all voltage partitions, is important since it enables energy saving from the voltage island shutdown mechanism. In this paper, we prove that the peak power driven voltage partitioning problem is NP-complete and propose an efficient provably good fully polynomial time approximation scheme for it. The new algorithm can approximate the optimal peak power driven voltage partitioning solution in $O(m^2(\frac{mn}{\epsilon^4})^m)$ time within a factor of $(1+\epsilon)$ for sufficiently small positive ϵ, where n is the number of circuit blocks and m is the number of partitions which is a small constant in practice. Our experimental results demonstrate that the dynamic programming cannot finish for even 20 blocks while our new approximation algorithm runs fast. In particular, varying ϵ, orders of magnitude speedup can be obtained with only 0.6% power increase. The tradeoff between the peak power minimization and the total power minimization is also investigated. We demonstrate that the total power minimization algorithm obtains good results in total power but with quite large peak power, while our peak power optimization algorithm can achieve on average 26.5% reduction in peak power with only 0.46% increase in total power. Moreover, our peak power driven voltage partitioning algorithm is integrated into a simulated annealing based floorplanning technique. Experimental results demonstrate that compared to total power driven floorplanning, the peak power driven floorplanning can significantly reduce peak power with only little impact in total power, HPWL, estimated power ground routing cost, level shifter cost and runtime. Further, when the voltage island shutdown is performed, peak power driven voltage partitioning can lead to over 10% more energy saving than a greedy frequency based voltage partitioning when multiple idle block sequences are considered.**

Keywords: Voltage Partitioning, Fully Polynomial Time Approximation Scheme, NP-Complete, Peak Power Minimization, Voltage Island Shutdown.

I. Introduction

With fast technology scaling, power consumption has become a bottleneck for circuit miniaturization. In practice, the dynamic power consumption due to charging and discharging the capacitance during transistor switching is still a very important factor in the total power consumption. It has been indicated in [1] that in a recent Intel high performance microprocessor, 60% of the total power is due to dynamic power consumption. In this paper, the "power" refers to the "dynamic power" which is proportional to $CVdd^2$ where C and Vdd refer to the effective capacitance and the supply voltage, respectively. As a highly effective technique to mitigate the dynamic power consumption, multiple supply voltages technique [2] has been widely used in practice.

In physical design domain, multiple supply voltages can be implemented through on-chip voltage island [9], [3], [4]. Basically, voltage island technique groups the functional units (i.e., blocks/modules) with different supply voltages into a small number of voltage islands, each of which has a single supply voltage. Note that in this paper, "block" and "functional unit" are used interchangeably. A critical step in voltage island is voltage partitioning. It clusters functional units into different voltage levels to maximize power reduction. Later, voltage islands can be generated using the voltage partitioning result during floorplanning. Refer to [3] for an example of such a flow.

Since voltage partitioning technique is an important enabling technique for the subsequent voltage island generation, it has received significant research attention [3], [4], [5], [10]. Previous works [3], [4], [5] consider the total power driven voltage partitioning problem which is to minimize the total power consumption over all voltage partitions. The peak power consumption which is the maximum power over all partitions is also quite important. For example, an important mechanism in the voltage island technique is to perform the runtime voltage island shutdown for energy saving. That is, when all the blocks in a voltage island are not in use during a period of time, the whole voltage island can be shut down to save energy during that time. Such a shutdown mechanism has been explored in various previous works including [6], [7], [8], [9]. Most of them need the assumption that the shutdown frequency of each voltage island is known in advance. Based on this assumption, they are able to design voltage partitioning particularly good for those shutdown frequencies, and thus these techniques can be viewed as input-specific voltage partitioning algorithms in some sense. The above assumption could be reasonable for some specific applications, however, it would be quite difficult to estimate the shutdown frequency for general applications. In fact, even for the same chip, it can be very difficult to estimate the shutdown frequency since it essentially asks to model the user behavior. For example, whether a user would surf the internet or watch the movie more often can lead to quite different shutdown frequencies on logic blocks and thus on voltage islands. In contrast, our peak power driven voltage partitioning minimizes the maximum power and tends to implicitly balance power distribution over all partitions. Having similar power in each voltage island could produce significant energy saving since each voltage island should have similar shutdown frequency when multiple user behaviors are considered. In contrast, focusing on only one user behavior could result in inferior energy saving for the others.

In this paper, the peak power driven voltage partitioning problem is proven to be NP-complete and is solved with approximation guarantee. Precisely, a fully polynomial time approximation scheme (FPTAS) is proposed. An FPTAS on our problem can always compute a solution with peak power bounded by $(1+\epsilon)P_{opt}$ for any small positive ϵ, with P_{opt} referring to the peak power of the optimal peak power driven voltage partitioning solution. In addition, it runs in time polynomially in terms of the number of functional units and $1/\epsilon$. An FPTAS is regarded as the best possible algorithm one can design for an NP-hard problem in theory unless P=NP [11]. The main contribution of this paper is summarized as follows.

- The problem of voltage partitioning for peak power minimization is proven to be NP-complete.
- An FPTAS algorithm is proposed to approximate the optimal solution by a factor of $1+\epsilon$ in $O(m^2(\frac{mn}{\epsilon^4})^m)$ time for sufficiently small positive ϵ, where n is the number of functional units and m is the number of partitions which is a constant in practice.
- The new FPTAS algorithm is practical. Our experimental results demonstrate that the dynamic programming algorithm cannot finish for even 20 functional units while our new provably good approximation algorithm runs fast. In particular, varying ϵ, orders of magnitude speedup can be obtained while the power increase can be as small as 0.6%.
- The tradeoff between the peak power minimization and the total power minimization is investigated. Our experimental results demonstrate that the total power minimization algorithm obtains good results in total power but with large peak power. In contrast, our peak power driven voltage partitioning algorithm can achieve on average 26.5% reduction in peak power with only 0.46% increase in total power.
- The peak power driven voltage partitioning algorithm is integrated into a simulated annealing based floorplanning technique. Experimental results demonstrate that compared to total power driven floorplanning, the peak power driven floorplanning can significantly reduce peak power with only little impact in total power, HPWL, estimated power ground routing cost, level shifter

978-1-4577-1400-9/11/$26.00 ©2011 IEEE

TABLE I
AN EXAMPLE FOR PEAK POWER DRIVEN VOLTAGE PARTITIONING.

Functional Units (Blocks)	t_1	t_2	t_3	t_4	t_5	t_6
Capacitance (c)	1	3	5	2	2	2
Minimum Mapped Voltage (v)	0.8	1.5	1.2	0.9	1.2	1.0

cost and runtime. Further, when the voltage island shutdown is performed, peak power driven voltage partitioning can lead to over 10% more energy saving than a greedy frequency based voltage partitioning when multiple idle block sequences are considered.

II. PROBLEM FORMULATION

Voltage partitioning is an enabling technique for forming voltage islands [4], [3], [10]. Recall that functional units are blocks/modules which will be placed in floorplanning. Voltage partitioning classifies functional units into a set of available discrete voltage levels. The input to the voltage partitioning problem is a minimum mapped voltage for each functional unit together with a capacitance for each functional unit which can be obtained through e.g., [3]. For a functional unit t, denote by $v(t)$ its initial minimum mapped voltage from the m available voltage levels and by $c(t)$ the effective capacitance. Its dynamic power $P(t)$ is computed as $c(t)v(t)^2$. Consider a set T of n functional units $T = \{t_1, t_2, \ldots, t_n\}$. The power of T is computed as the sum of the product of each capacitance and the square of maximum voltage, i.e., $P(T) = \sum_{i=1}^{n} c(t_i)v(T)^2$, where $v(T) = \max_i v(t_i)$. Given m voltage levels, the problem is to classify a set T of functional units into m pair-wise disjoint partitions such that the peak power over all m partitions, $\max_i\{P(T_i)\}$, is minimized.

It is helpful to look at an example to illustrate the above concepts. Refer to Table I which shows the functional units and their minimum mapped voltages and capacitances. Suppose that $m = 2$ and the functional units are clustered into two partitions, namely, $T_1 = \{t_1, t_2, t_3\}$ and $T_2 = \{t_4, t_5, t_6\}$. For T_1, $\max_i\{v(t_i)\} = 1.5, \sum_{t_i \in T_1} c(t_i) = 9$, and $P(T_1) = 9 \cdot 1.5^2 = 20.25$. For T_2, $\max_i\{v(t_i)\} = 1.2, \sum_{t_i \in T_2} c(t_i) = 6$, and $P(T_2) = 6 \cdot 1.2^2 = 8.64$. Thus, $\max_i\{P(T_i)\} = P(T_1) = 20.25$. Our problem asks to compute a partitioning solution to minimize $\max_i\{P(T_i)\}$.

Note that the difference in problem formulation between this work and [10] is that roughly speaking they minimize $\max_j[\max_{t_i \in T_j} c(t_i) \cdot \max_{t_i} v(t_i)^2]$ while we minimize $\max_j[\sum_{t_i \in T_j} c(t_i) \cdot \max_{t_i} v(t_i)^2]$ (precisely, their $c(t_i)$ refers to capacitance per unit area). The algorithm in [10] heavily relies on the fact that the peak power density of each partition is determined by only two functional units (i.e., $\max c$ and $\max v$), while in our case the peak power of a partition is determined by all functional units in the partition (due to $\sum c$). This critical difference makes their optimal algorithm not applicable in our case. As regard to total power driven voltage partitioning [3], [4], [5], their algorithms heavily rely on the fact that in any optimal solution there, each partition has to consist of functional units with consecutive voltages (if we sort all voltages of functional units). However, it is easy to design a counterexample in our peak power driven voltage partitioning problem such that the optimal solution can be only achieved through grouping functional units with non-consecutive voltages. Such an example is omitted due to space limitation. Our problem is as follows.

Peak Power Driven Voltage Partitioning Problem: Given a set T of functional units, the capacitance of each functional unit, and a set of discrete voltage levels, to compute a voltage partitioning solution with m partitions, i.e., $T = T_1 \bigcup T_2 \bigcup \ldots \bigcup T_m$, where $T_i \bigcap T_j = \emptyset, i \neq j$, such that the peak power over all partitions, $\max_i\{P(T_i)\}$, is minimized.

III. NP-COMPLETE PROOF

The problem is shown to be NP-complete. It is sufficient to prove that the problem is NP-complete even when $m = 2$. The corresponding decision problem is: given a value p, whether there is a voltage partitioning solution $\{T_1, T_2\}$ satisfying:

$$\max_i P(T_i) \leq p.$$

Given any voltage partitioning solution, it is easy to verify the above condition in polynomial time. Thus, the problem is in NP.

To prove the NP-hardness, we reduce from the perfect bipartition problem which is known to be NP-complete [12]. The problem asks to decide with a given integer w, whether n integers $\{a_1, a_2, \ldots, a_n\}$ can be partitioned into two groups/partitions $\{A_1, A_2\}$, satisfying:

$$|\sum A_1 - \sum A_2| \leq w.$$

Without loss of generality, assume that all the numbers a_i and w are even integers. This is valid since one can scale up every number by a factor of 2 which will not impact the answer of the decision problem.

Given an instance of perfect bipartition problem, an instance of voltage partitioning is constructed as follows. Set $c(t_i) = a_i, \forall i$ and $p = M/2 + w/2$ where $M = \sum_i c(t_i)$. Set $v(t_i)$ to be any positive value within the range $(\sqrt{1 - 1/(p+1)}, 1), \forall i$. Since a_i and w are even integers, p is an integer. Note that the number of distinct $v(t_i)$ can be bounded by either n or m, which will not impact the NP-hardness proof. We claim that there is a solution for the given perfect number partitioning instance if and only if there is a solution for the constructed voltage partitioning instance.

We begin with "only if" direction. Given a solution satisfying $\max_i P(T_i) \leq p$, it means that

$$\max(\sum_{t_i \in T_1} c(t_i), \sum_{t_i \in T_2} c(t_i)) \leq M/2 + w/2. \quad (1)$$

This is the case due to the following.

$$P(T_j) = \sum_{t_i \in T_j} c(t_i)v(T_j)^2 = v(T_j)^2 \sum_{t_i \in T_j} c(t_i) \leq p. \quad (2)$$

Thus, $\sum_{t_i \in T_j} c(t_i) \leq p/v(T_j)^2$ for $j = 1, 2$. Since $1 - 1/(p+1) < v(T_j)^2$, $p/v(T_j)^2 < p + 1$. Subsequently, $\sum_{t_i \in T_j} c(t_i) < p + 1$. It means that $\sum_{t_i \in T_j} c(t_i) \leq p$ since all $c(t_i) = a_i$ are integers. Thus, $\sum_{t_i \in T_j} c(t_i) \leq p$.

On the other hand, since $M = \sum_i c(t_i)$, from Eqn. (1) we have

$$\min(\sum_{t_i \in T_1} c(t_i), \sum_{t_i \in T_2} c(t_i)) \geq M - (M/2 + w/2) = M/2 - w/2. \quad (3)$$

Combining the Eqn. (1) and Eqn. (3), one has

$$|\sum_{t_i \in T_1} c(t_i) - \sum_{t_i \in T_2} c(t_i)| \leq w. \quad (4)$$

Subsequently, a_i can be assigned according to $c(t_i)$, that is, a_i is assigned to A_1 if and only if $c(t_i)$ is in T_1. This gives a solution for the perfect bipartition problem since $c(t_i) = a_i, \forall i$.

We next prove "if" direction. Given a solution for the perfect bipartition problem instance, one just needs to accordingly form the voltage partition in the voltage partitioning instance, namely, $c(t_i)$ is assigned to T_1 if and only if a_i is in A_1. $|\sum_{a_i \in A_1} a_i - \sum_{a_i \in A_2} a_i| \leq w$ means that

$$\max(\sum_{t_i \in T_1} c(t_i), \sum_{t_i \in T_2} c(t_i)) \leq M/2 + w/2 = p, \quad (5)$$

because otherwise the absolute difference between the two sum will be greater than w. Since

$$\begin{aligned} &\max(\textstyle\sum_{t_i \in T_1} P(t_i), \sum_{t_i \in T_2} P(t_i)) \leq \\ &\max_{t_i} v(t_i)^2 \cdot \max(\textstyle\sum_{t_i \in T_1} c(t_i), \sum_{t_i \in T_2} c(t_i)), \end{aligned} \quad (6)$$

and $\max_{t_i} v(t_i)^2 < 1$, we have

$$\max(\sum_{t_i \in T_1} P(t_i), \sum_{t_i \in T_2} P(t_i)) \leq p. \quad (7)$$

This gives a solution for the voltage partitioning problem. We reach the following theorem.

Theorem 1: The voltage partitioning for peak power minimization problem is NP-complete.

TABLE II
FUNCTIONAL UNITS AFTER TRANSFORM.

Functional Unit	t_1	t_2	t_3	t_4	t_5	t_6
Capacitance (c')	0.2	0.6	1	0.4	0.4	0.4
Voltage (v')	10	18.75	15	11.25	15	12.5

IV. THE ALGORITHMIC FLOW

As the best possible solution for an NP-complete problem in theory, a fully polynomial time approximation scheme is proposed for the peak power voltage partitioning problem in this paper. At a high level, the voltage partitioning problem instance is first scaled and rounded by the double transform which consists of exact transform and approximate transform. As a result, the number of non-redundant voltage partitioning solutions is limited to be polynomial in the total number of functional units and $1/\epsilon$ which is related to the approximation guarantee. Varying ϵ which is a user input, our algorithm can achieve different tradeoff between solution quality and runtime. Moreover, the tradeoff is provably good. After scale-round procedure, a dynamic programming is proposed to solve the scaled voltage problem instance. Without performing scale-round transform, the dynamic programming runs in exponential time. A critical pruning technique is integrated in the dynamic programming which can greatly speedup the algorithm without solution quality degradation. Since the total number of non-redundant solutions is bounded, the dynamic programming runs in polynomial time, i.e., $O(m^2(\frac{mn}{\epsilon^4})^m)$ time.

V. THE ALGORITHM

A. Exact Transform

The objective of the transform is to achieve $c(t_i) \cdot v(t_i)^2 \geq m$ and $0 < c(t_i) \leq 1$ for every t_i. The process of the transform consists of two steps.

Step 1: scale capacitance $c(t_i)$ by $\max c(t_i)$ for all t_i. Denote the resulting c by c'. It is clear that $c'(t_i) \in (0, 1]$.

Step 2: scale voltage $v(t_i)$ to $\frac{m \cdot v(t_i)}{\min c'(t_i) \cdot \min v(t_i)}$ for all t_i. Denote the resulting v by v'.

After the transform, $v'(t_i) > 1$ and $c'(t_i) \cdot v'(t_i) > m$. Thus, the scaled power $c'(t_i) \cdot v'(t_i)^2 > m$. In any voltage partitioning solution, there exists one partition with at least $\lceil n/m \rceil$ functional units since there are n functional units. For this partition, the power is at least $m \cdot \lceil n/m \rceil > n$. Denote by OPT the optimal solution. This means that $OPT > n$ after exact transform (since there exists a partition with peak power $> n$). To apply the exact transform to the example shown in Table I, one obtains the result as shown in Table II. For example, $c'(t_1)$ is obtained by scaling $c(t_1)$ with the factor of $\max c(t_i) = 5$, and $v'(t_1)$ is obtained by scaling $v(t_1)$ to $\frac{m \cdot v(t_1)}{\min c'(t_i) \cdot \min v(t_i)} = 10$ where $\min c'(t_i) = 0.2$, $\min v(t_i) = 0.8$ and $m = 2$. Note that the exact transform will not impact the optimality of the solution since one can always scale back the optimal solution for the scaled voltage partitioning problem to obtain the optimal solution for the original voltage partitioning problem.

B. Approximate Transform

Through the above transform, c' is in the range of $(0, 1]$. One then divides the interval into small intervals at a step of ϵ where $0 < \epsilon < 1$. In each group round the c' to the lower bound of this group. For example if $7\epsilon \leq c'(t_i) \leq 8\epsilon$, after round-down, $c'(t_i) = 7\epsilon$. After performing this approximate transform, there are $O(1/\epsilon)$ number of distinct c'. This key property will be used to achieve the provable tradeoff between approximation bound and runtime. For convenience, c and v are used to denote c' and v' in the following subsections.

C. Dynamic Programming

After above scale-round procedure, we are ready to present the key dynamic programming approach for the voltage partitioning problem. At a high level, the functional units will be processed one by one and the partial voltage partitioning solutions will be propagated. When processing a functional unit, it is assigned to each of m possible partitions. Thus, given a partial solution where the first $i-1$ functional units have been processed, m new partial solutions can be generated due to processing i-th functional unit since there are totally m partitions. After all the partial solutions are processed on i-th functional unit (i.e., all new partial solutions are generated), the algorithm proceeds to the next functional unit. The process is iterated until the last functional unit is processed. Subsequently, the solution with minimum peak power will be returned. Without any speedup technique, this algorithm runs in exponential time.

Before describing the speedup technique, each partial solution is first characterized. Denote a partial solution by s. s consists of a set of m partitions $T_1(s), T_2(s), \ldots, T_m(s)$. For a partition $T_j(s)$, denote its maximum voltage $\max_{t_j \in T_j(s)} v(t_j)$ by $maxv_j(s)$, and its total capacitance $\sum_{t_j \in T_j(s)} c(t_j)$ by $sumc_j(s)$, respectively. Each solution s is characterized by a $2m$-unit $(maxv_1(s), sumc_1(s), \ldots, \ldots, maxv_m(s), sumc_m(s))$.

Given a partial solution s_{i-1} after processing the first $i-1$ functional units, m new partial solutions $s_{i,1}, s_{i,2}, \ldots, s_{i,m}$ are generated as follows.

$$\begin{aligned} T_j(s_{i,j}) &= T_j(s_{i-1}) \bigcup \{t_i\}, \\ T_k(s_{i,j}) &= T_k(s_{i-1}), \forall k \neq j. \end{aligned} \tag{8}$$

The solution characterization is updated as follows.

$$\begin{aligned} maxv_j(s_{i,j}) &= \max\{maxv_j(s_{i-1}), v(t_i)\}, \\ maxv_k(s_{i,j}) &= maxv_k(s_{i-1}), \forall k \neq j, \end{aligned} \tag{9}$$

and

$$\begin{aligned} sumc_j(s_{i,j}) &= sumc_j(s_{i-1}) + c(t_i), \\ sumc_k(s_{i,j}) &= sumc_k(s_{i-1}), \forall k \neq j. \end{aligned} \tag{10}$$

As mentioned above, if one just propagates the solutions in this way, the number of solutions will increase exponentially. For speedup, *redundant solution* in our voltage partitioning algorithm is defined as follows. Given two solutions s_1 and s_2 formed by processing the same set of functional units, s_1 is redundant with respect to s_2 if $maxv_i(s_1) = maxv_i(s_2)$ and $sumc_i(s_1) = sumc_i(s_2)$ for all i. In other words, solution s_1 is redundant with respect to s_2 if it has the same maximum voltage and total capacitance for each partition. When a solution is redundant, it is eliminated from the solution set for further solution propagation. This is valid since the power is uniquely determined by $maxv$ and $sumc$ for each group.

One may wonder how much speedup we may get by performing the above pruning procedure. The speedup is actually quite significant. The following lemma says that there are only polynomial number of non-redundant solutions which need to be maintained through performing the dynamic programming, a large improvement over the worst-case exponential-time algorithm.

Lemma 1: For any set that has k functional units are partitioned into m partitions, there are at most $O((\frac{mk}{\epsilon})^m)$ possible non-redundant solutions.

Proof: Consider a single partition i in a solution s. For a set of functional units, there are at most m distinct $maxv_i$. On the other hand, after scale-round transform, all the capacitances will be multiples of ϵ. Note that $c(t_i) \leq 1$ and $\sum c(t_i) \leq k$. Thus, there are at most $\frac{k}{\epsilon}$ distinct $sumc$ for partition i. For all m partitions in a solution s, the number of distinct characterizations is bounded by $O((\frac{mk}{\epsilon})^m)$. This means that there are only $O((\frac{mk}{\epsilon})^m)$ non-redundant solutions during dynamic programming. Thus, after performing solution pruning, there are at most $O((\frac{mk}{\epsilon})^m)$ solutions.

The solution pruning can be implemented in a way similar to radix sort as follows. After all new solutions are generated, they are sorted according to $maxv_1$. For those solutions with the same $maxv_1$, they are sorted according to $sumc_1$. For the solutions with the same $maxv_1$ and $sumc_1$, they are sorted according to $maxv_2, sumc_2$. This process is repeated until solutions are sorted according to $maxv_m, sumc_m$. According to the definition of redundant solutions, given multiple solutions with the same $maxv_i$ and $sumc_i$ for all i, only one of them is maintained. All the redundant solutions can be found by a single scan over sorted solutions. After processing i functional units, there are only at most i distinct $maxv$, and i/ϵ distinct $sumc$. Thus, one scan is sufficient to move each solution in the solution set to the one of i corresponding $maxv$ values for $maxv_1$. Similarly, one scan is sufficient to move each solution with same $maxv_1$ to the one of i/ϵ corresponding $sumc$ values. Thus, this solution pruning process takes $O(m^2 \cdot (\frac{m(i-1)}{\epsilon})^m) = O(m^2 \cdot (\frac{mi}{\epsilon})^m)$ time. This is the case since (1) there are m scans for $maxv$ and m scans for $sumc$ over all solutions, and (2) there are at most

$O((\frac{m(i-1)}{\epsilon})^m)$ non-dominated solutions after processing the previous functional unit and each partial solution can generate m new partial solutions.

The algorithm is now clear. After processing each functional unit, the solution pruning is performed. Suppose that all of the first $i-1$ functional units have been processed. There are at most $O((\frac{m(i-1)}{\epsilon})^m)$ non-redundant solutions. After processing the i-th functional unit, there are at most $O(m(\frac{m(i-1)}{\epsilon})^m)$ solutions since each partial solution can generate m new partial solutions. After solution pruning, there are at most $O((\frac{mi}{\epsilon})^m)$ solutions. This process is repeated until the last functional unit is processed. Subsequently, the solution with minimum peak power ($\max_{T_i} P(T_i)$) will be returned. Since each capacitance is scaled/rounded as in Section V-B with rounding errors, the unrounded capacitance will be used in reporting the peak power (but voltage assignment is not changed). Note that the scaling due to the exact transform in Section V-A introduces no error and thus rounding it back will not impact the solution optimality.

D. Approximation Guarantee

Note that the approximation error is introduced due to the approximate rounding on capacitance in the approximate transform in Section V-B. Approximate rounding is necessary since otherwise the number of distinct capacitance sum is unbounded. Consequently, the optimal solution, denoted by $OPT_{round-down}$, of the round-down problem forms a lower bound on the optimal solution, denoted by OPT, in the unrounded problem. It is clear that our algorithm computes the optimal solution for the round-down problem, i.e., $OPT_{round-down}$. The problem is how much error there is between OPT and $OPT_{round-down}$.

Given a capacitance $c(t_i)$ in the interval of $[r\epsilon, (r+1)\epsilon)$ for an integer r, it is rounded to $r\epsilon$ in approximate transform. The multiplicative rounding error is at most $\frac{r+1}{r}$. The problem is that when r is small, this analysis gives a large approximation ratio. Consider the extreme case when $r=0$, such an analysis would give an unbounded approximation ratio. Our analysis tackles this difficulty through noting that when r is small, the additive rounding error is quite limited. Our analysis is as follows.

We set a parameter r such that the error due to rounding back $c(t)$ with $c(t) \geq r$ is measured by the multiplicative error and the error due to rounding back $c(t)$ with $c(t) < r$ is measured by the additive error. For multiplicative rounding error, the error on capacitance and power is at most $\frac{r+1}{r}$. For additive rounding error, the error on capacitance is at most $nr\epsilon$ and the error on power is at most $(\max_i\{v(t_i)\})^2 \cdot nr\epsilon$ since there are n functional units. Denote by ALG the solution returned by our algorithm. Thus,

$$OPT \leq ALG \leq \frac{r+1}{r} OPT_{round-down} + n(\max_i\{v(t_i)\})^2 r\epsilon.$$

Recall that $OPT \geq n$ due to the exact transform and that $OPT_{round-down} \leq OPT$. We have $ALG \leq \frac{r+1}{r} OPT + OPT(\max_i\{v(t_i)\})^2 r\epsilon$. Suppose that ϵ is sufficiently small, i.e., $\epsilon \leq \frac{1}{(\max_i\{t_i\})^8}$. This is desired since in practice, one always wishes to compute the solution close to the optimum. Set $r = \frac{1}{\sqrt{\epsilon}}$. We have $(\max_i\{v(t_i)\})^2 r\epsilon \leq \epsilon^{3/4} r \leq \epsilon^{1/4}$, and $ALG \leq (1 + \sqrt{\epsilon} + \epsilon^{1/4}) OPT$. Since $\sqrt{\epsilon} < \epsilon^{1/4}$ when $\epsilon < 1$ which is desired, $ALG \leq (1 + 2\epsilon^{1/4}) OPT$. Setting $\epsilon' = 2\epsilon^{1/4}$, a $(1+\epsilon')$-approximation algorithm is obtained for the voltage partitioning problem.

E. Time Complexity

Exact transform, approximate transform, and rounding back take linear time. Most time is spent on performing the dynamic programming. For each functional unit, the pruning process needs $O(m^2 \cdot (\frac{mn}{\epsilon})^m)$ time as mentioned above. Summing over all functional units, the whole algorithm is bounded by $O(m^2 \cdot (\frac{mn}{\epsilon})^m)$ time. Plugging ϵ' in, the time complexity becomes $O(m^2 \cdot (\frac{mn}{\epsilon'^4})^m)$. We reach Theorem 2.

Theorem 2: A $(1+\epsilon)$ approximation to the voltage partitioning for peak power minimization problem can be computed in $O(m^2(\frac{mn}{\epsilon^4})^m)$ time for a sufficiently small $\epsilon > 0$, where n is the number of functional units and m is the number of partitions.

F. Extension to Peak Power Driven Floorplanning

The proposed algorithm for peak power driven voltage partitioning is also integrated with a floorplanning technique and we compare it to a classical total power driven floorplanning. Following many existing floorplanning techniques, simulated annealing is used. In our simulated annealing implementation, the optimization objective consists of three parts: the half perimeter wire length (HPWL), the estimated power ground routing cost and the level shifter cost. Denote HPWL of the floorplanning solution by W_{hpwl} which is defined as the summation of HPWL of all block-level nets. Since different floorplans will lead to different power ground routing topologies, it is necessary to estimate this routing cost. In this paper, the power ground routing estimation cost used in [13] is adopted. It is denoted by W_{pg} and is computed as the sum of HPWL of bounding box enclosing each group of blocks assigned with the same voltage, i.e., the sum of the HPWL of the bounding box of each voltage island. In fact, this metric has been widely used to estimate power ground routing cost [13], [14]. As is well known, a voltage level-shifter is needed when a lower voltage block drives a higher voltage block. The level shifter cost, denoted by W_{ls}, is estimated through the number of voltage level-shifters, which is also the same as [13]. Our optimization objective for simulated annealing is as follows.

$$\min \alpha \cdot W_{hpwl} + \beta \cdot W_{pg} + \gamma \cdot W_{ls}, \tag{11}$$

where α, β and γ are the user defined weighting factors.

Note that before floorplanning, the voltage of each block is assigned to be the one computed from our power density driven voltage partitioning algorithm. Similarly, in the total power driven floorplanning, the voltage of each block is assigned to be the one computed from total power driven voltage partitioning. Due to the space limitation, the details of our simulated annealing implementation are omitted.

VI. Experimental Results

The proposed provably good approximation algorithm for the voltage partitioning problem is implemented in C++ and tested on an Intel machine with $1.8GHz$ CPU and $2GB$ memory. The new algorithm is compared to the optimal dynamic programming approach. We will also compare to total power minimization algorithm [4]. Due to lack of industrial testcases, the experiments are performed to a set of 10 small randomly generated testcases. For each testcase, the capacitances are randomly generated with means of $20pF$. Five voltage levels are used in the technology library and the initial mapped minimum voltage of each functional unit (block) is randomly set to be one of them. Note that previous voltage partitioning works such as [3], [4], [10] also use the randomly generated testcases in their experiments and our experiment setup is similar to theirs.

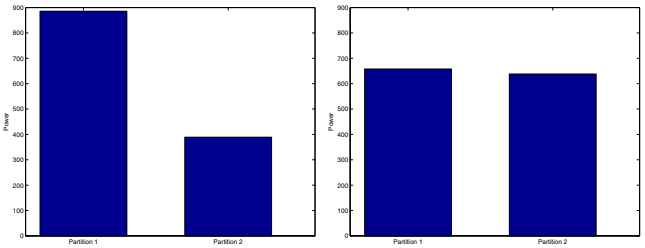

Fig. 1. Power distribution among all three partitions for the testcase with 50 functional units. (a) total power driven voltage partitioning (b) peak power driven voltage partitioning.

We would like to demonstrate the performance of the new FPTAS algorithm through comparing to the optimal solution. However, the dynamic programming technique which generates the optimal solution cannot finish on the testcases even with 20 functional units (see below). Thus, *solution number restriction* technique is applied where only the top 10 solutions with the minimum peak power are maintained during dynamic programming. The results on 10 testcases are summarized in Table III. Although our theoretical analysis of the approximation scheme uses small ϵ, the proposed FPTAS also works well for large ϵ as demonstrated in Table III. We make the following observations.

TABLE III

COMPARISON OF THE DYNAMIC PROGRAMMING WITH SOLUTION NUMBER RESTRICTION AND THE NEW ALGORITHM WITHOUT SOLUTION NUMBER RESTRICTION ON 10 TESTCASES. # UNITS IS THE NUMBER OF FUNCTIONAL UNITS. P. POWER IS PEAK POWER. CPU IS RUNTIME IN SECONDS.

Testcases		Dynamic Programming w/ Sol. Num. Restriction		New FPTAS (with $\epsilon = 5\%$)		New FPTAS (with $\epsilon = 10\%$)		New FPTAS (with $\epsilon = 50\%$)				New FPTAS (with $\epsilon = 150\%$)	
Index	# units	P. Power	CPU(s)	P. Power	CPU(s)	P. Power	CPU(s)	P. Power	CPU(s)	Ratio	Speedup	P. Power	CPU(s)
1	50	717.4	0.7	648.9	48.4	648.9	7.1	658.3	0.8	1.5%	60.5×	885.7	0.5
2	55	774.6	0.7	708.0	83.4	708.0	10.7	711.2	1.0	0.5%	83.4×	938.7	0.6
3	60	892.1	0.8	789.3	128.0	789.3	16.0	794.7	1.2	0.7%	106.7×	1050.0	0.7
4	70	1041.3	0.9	932.7	208.0	933.0	44.7	939.3	1.2	0.7%	173.3×	1236.7	0.8
5	75	1104.1	1.0	1003.4	283.4	1003.3	66.8	1007.8	1.3	0.4%	218.0×	1305.1	0.8
6	80	1168.6	1.1	1068.3	397.7	1068.4	90.1	1068.8	1.5	0%	265.1×	1359.4	0.9
7	85	1237.3	1.1	1136.6	600.5	1136.4	134.1	1142.6	1.7	0.5%	353.2×	1495.8	0.9
8	90	1312.3	1.2	1205.5	798.7	1205.4	182.0	1210.8	1.9	0.4%	420.4×	1624.3	1.0
9	95	1385.0	1.3	1264.3	1009.5	1263.9	238.9	1273.9	2.1	0.8%	480.7×	1708.7	1.0
10	100	1451.4	1.4	1325.3	1325.0	1325.3	319.1	1335.8	2.2	0.8%	602.3×	1820.9	1.1
Average										0.6%	276.4×		

TABLE IV

COMPARISON OF THE RESTRICTED DYNAMIC PROGRAMMING APPROACH AND THE RESTRICTED NEW FPTAS ALGORITHM ON A SET OF 5 LARGE TESTCASES. CPU REFERS TO RUNTIME IN SECONDS. NOTE THAT BOTH ALGORITHMS ARE WITH SOLUTION RESTRICTION SET TO 10.

Test cases		Restricted Dynamic Programming		Restricted New FPTAS (with $\epsilon = 50\%$)	
Index	# units	Peak Power	CPU(s)	Peak Power	CPU(s)
1	500	7419.5	9.9	6809.0	9.5
2	1000	14891.5	27.4	13859.3	26.5
3	2000	29869.8	87.5	28564.6	84.8
4	4000	59644.4	320.2	57416.5	301.6
5	5000	74467.4	490.8	71551.1	465.0

TABLE V

COMPARISON OF THE TOTAL POWER MINIMIZATION ALGORITHM AND THE NEW FPTAS ALGORITHM (FOR PEAK POWER MINIMIZATION) ON A SET OF 5 LARGE TESTCASES USED IN TABLE IV. THE RESULTS ARE EVALUATED USING TOTAL POWER AND PEAK POWER.

Test cases		Total Power Minimization Algorithm		New FPTAS (with $\epsilon = 50\%$)			
Index	# units	Total Power	Peak Power	Total Power	Peak Power	Total Power Increase	Peak Power Reduction
1	500	13553.0	9669.7	13614.5	6809.0	0.45%	29.6%
2	1000	27123.2	19238.9	27434.0	13859.3	1.15%	28.0%
3	2000	54384.3	38355.0	54480.1	28564.6	0.17%	25.5%
4	4000	108365.7	75961.4	108669.5	57416.5	0.28%	24.4%
5	5000	135391.1	95113.2	135707.4	71551.1	0.23%	24.8%
Average						0.46%	26.5%

TABLE VI

COMPARISON OF FLOORPLANNING RESULTS ON LARGE GSRC BENCHMARK CIRCUITS. HPWL REFERS TO THE HALF PERIMETER WIRE LENGTH, P/G COST REFERS TO THE ESTIMATED POWER GROUND ROUTING COST AS IN [13], L.S. COST REFERS TO THE LEVEL SHIFTER COST, CPU REFERS TO THE RUNTIME IN SECONDS, T.P. REFERS TO THE TOTAL POWER, AND P.P. REFERS TO THE PEAK POWER.

Test cases		T.P.-Driven Floorplanning						P.P.-Driven Floorplanning							
Name	# units	HPWL	P/G Cost	L.S. Cost	CPU(s)	T.P.	P.P.	HPWL	P/G Cost	L.S. Cost	CPU(s)	T.P.	P.P.	T.P. Inc.	P.P. Red.
$n50$	50	198152	1891	235	3.8	1341	913	198152	1891	217	4.3	1369	685	2.0%	25.1%
$n100$	100	249364	2353	497	5.0	2587	1364	254036	2318	469	5.9	2619	952	1.2%	30.2%
$n200$	200	447921	1649	743	6.7	4548	2999	447900	1639	733	7.7	4609	2609	1.3%	13.0%
$n300$	300	657132	2094	861	8.2	8082	5469	657265	2075	830	11.2	8147	4507	0.8%	17.6%
Average														1.3%	21.5%

- The runtime of dynamic programming approach can be treated as infinity since without solution number restriction, dynamic programming can not finish within 2 hours even for a testcase at size of 20 functional units. With solution number restriction, the solution quality by dynamic programming may be significantly degraded, and is worse than the results by FPTAS for computing approximations as is clear from Table III.
- Our new FPTAS algorithm can compute the solution efficiently without any solution number restriction in these 10 testcases due to the powerful solution set reduction resulting from the scale-round approximate transform.
- Our new FPTAS always returns the solution within the target ϵ. This is theoretically guaranteed by the nature of our FPTAS algorithm, although we cannot empirically demonstrate this in large testcases since the optimal solution cannot be computed. Note that in some cases, the solution quality does not change (e.g., comparing $\epsilon = 5\%$ and $\epsilon = 10\%$ on testcase 1). This is possible as long as they are within the approximation guarantee.
- Comparing the results by FPTAS with different target approximation ratios, one can clearly see the solution quality and runtime tradeoff, which is guaranteed theoretically. In particular, the setting $\epsilon = 50\%$ gives the best performance. For testcase 10, compared to the case when $\epsilon = 5\%$, it achieves the speedup of about $602.3\times$ with only $1335.8/1325.3-1 = 0.8\%$ peak power increase. Our FPTAS algorithm provides significant flexility to designers for various algorithmic optimization strength which would benefit a large multitude of application scenarios. For example, in some fast estimation scenarios, large ϵ could be used while in some high performance design stage, small ϵ should be used.

Using solution number restriction, our FPTAS can be easily extended to handle larger testcases. For this, we perform the experiments on five large randomly generated testcases whose sizes range between 500 and 5000. The results are summarized in Table IV. Due to space limitation, only the results with $\epsilon = 50\%$ are shown. Both dynamic programming and our new FPTAS algorithm are with the same solution number restriction (i.e., 10). It is clear that our new FPTAS (with $\epsilon = 50\%$) always outperforms the dynamic programming in terms of solution quality, with up to $7419.5/6809.0 - 1 = 8.9\%$ reduction in peak power. The speedup is less significant since both algorithms use solution number restriction.

The tradeoff between the peak power minimization and the total power minimization is also investigated since both optimization objectives can be important in practice. Refer to Table V which compares peak power and total power between the total power minimization algorithm [3], [4], [5] and the proposed peak power minimization FPTAS algorithm with $\epsilon = 50\%$. It is clear that the total power minimization algorithm obtains good results in total power but with large peak power. In contrast, our peak power optimization algorithm can achieve on average 26.5% reduction in peak power

with only 0.46% increase in total power. That is, our peak power minimization algorithm can *significantly reduce the peak power at almost no cost of total power increase*.

A closer look at our FPTAS algorithm would reveal the reason. Our algorithm essentially tries to *balance the power of each individual partition during the process of minimizing the largest power over all partitions*. This means that our algorithm tends to implicitly minimize the sum of power over all voltage partitions as well, which is the total power. In contrast, the total power minimization algorithm [4] ignores the power of each individual partition and only targets to minimize the sum of power over all partitions. In addition to the analysis, the above phenomenon has been observed experimentally. As an example, refer to Figure 1(a) for the power distribution of total power driven voltage partitioning solution and Figure 1(b) for the power distribution of peak power driven voltage partitioning solution on the testcase with 50 functional units. It is clear that peak power driven voltage partitioning leads to much more balanced power distribution among the partitions than the total power driven voltage partitioning. Therefore, the total power minimization algorithm would not generate good results in terms of peak power, while our FPTAS algorithm can well approximate the optimal total power with much lower peak power.

The proposed peak power driven voltage partitioning is integrated with a simulated annealing based floorplanning technique as described in Section V-F. We compare Total Power (T.P.)-driven floorplanning and Peak Power (P.P.)-driven floorplanning. Before floorplanning, the voltage of each block is assigned to be the one computed from total power driven voltage partitioning algorithm (for T.P.-driven floorplanning) or the peak power driven voltage partitioning algorithm (for P.P.-driven floorplanning), respectively. We perform the experiments on the largest four GSRC floorplanning benchmark circuits. Due to the lack of the information of the voltage partition in GSRC benchmark circuit, some additional information is randomly generated. Since the timing information is not known, as before capacitance for each block is randomly generated and the mapped minimum voltage of each functional unit (block) is randomly set to be one of five voltage levels in the technology library. Table VI summarizes the results. Our P.P.-driven floorplanning can always achieve better peak power with little sacrifice in total power. Table VI shows that P.P.-driven floorplanning can save on average 21.5% peak power compared to T.P.-driven partition floorplanning, with only 1.3% increase in total power. One may wonder whether changing the power target from total power to peak power may significantly impact other objectives in optimization. As one can see, HPWL cost, P/G cost and L.S. cost of our P.P.-driven floorplanning are all quite similar to those of T.P.-driven floorplanning.

Further,, we perform voltage island shutdown technique to our P.P.-driven voltage partitioning solution and compare to a natural greedy frequency based voltage partitioning. Since voltage island shutdown is an important mechanism in the voltage island technique, such experiments can help investigate the impact of the proposed P.P.-driven voltage partitioning to the energy saving. The results on GSRC benchmark $n100$ are shown in Table VII. In the experiments, to mimic multiple user behaviors, we randomly generate 5 different sequences of sets of idle blocks. In each sequence, there are 100 sets corresponding to 100 timestamps, where each set consists of some randomly selected blocks that are assumed to be idle at the corresponding timestamp. Recall that when all the blocks in a partition are idle at a timestamp, the voltage island can be shut down at that timestamp. Given a sequence of sets of idle blocks, define the idle frequency of a block to be the number of occurrence of the block in this sequence. In the greedy frequency based voltage partitioning, we compute the idle frequency of each block and sort the blocks in the decreasing order according to the obtained frequencies. To compute the idle frequencies, one needs to know which one out of 5 sequences should be used, which is difficult to decide. Thus, we compare to the greedy algorithm with the idle frequency computation using every sequence and also using the merging of the 5 sequences, resulting in six comparisons. Refer to Table VII for the results where the energy reduction due to voltage island shutdown is shown. We make the following observations.

- In most cases, the greedy algorithm partitioning give the best result for the sequence it uses to compute the idle frequency, which is expected. For example, Seq1 based greedy algorithm produces the largest energy saving over all other algorithms for sequence 1. In fact, using the merging of the 5 sequences in the idle frequency computation still leads to 10.6% worse results compared to our algorithm. However, the greedy algorithm gives significantly worse results in many of the rest sequences than P.P.-driven voltage partitioning solution. For example, Seq1 based greedy algorithm produces less energy saving in sequences 2, 3 and 4. This makes sense since the greedy algorithm is only optimized for one sequence and thus has little control on the energy saving from other sequences.
- Comparing the total energy saving from 5 sequences, P.P.-driven voltage partitioning always produces the largest energy saving and the energy saving improvement can be from 11.5% to 13.2%. This is not surprising since P.P.-driven voltage partitioning solution, which minimizes the maximum power, tends to implicitly balance power distribution over all partitions as demonstrated before. Having similar power in each voltage island could produce significant energy saving since each voltage island should have similar shutdown frequency when there are multiple different sequences of idle blocks.

TABLE VII
ENERGY SAVING RESULTS THROUGH PERFORMING VOLTAGE ISLAND SHUTDOWN ON P.P.-DRIVEN PARTITIONING, T.P.-DRIVEN PARTITIONING AND GREEDY FREQUENCY BASED PARTITIONING RESULTS ON GSRC $n100$. SEQ DENOTES THE SEQUENCE, TOTAL DENOTES THE TOTAL SAVED ENERGY AND RATIO IS CALCULATED THROUGH COMPARING TO TOTAL OF P.P.-DRIVEN VOLTAGE PARTITIONING.

Algorithms	Seq1	Seq2	Seq3	Seq4	Seq5	Total	Ratio
Seq1 based Greedy	14396	11133	13455	12361	10405	61750	88.2%
Seq2 based Greedy	12366	16015	11534	11534	9325	60774	86.8%
Seq3 based Greedy	11302	11871	16030	12820	9944	61967	88.5%
Seq4 based Greedy	11398	12158	11894	14793	11250	61493	87.8%
Seq5 based Greedy	9947	11733	14296	12007	13120	61103	87.3%
Merge based Greedy	12979	12099	12407	13628	11483	62596	89.4%
T.P.-driven partition	10724	13378	10136	16609	10510	61357	87.6%
P.P.-driven partition	12873	16178	14506	18109	8353	70019	100%

REFERENCES

[1] R. Shelar and M. Patyra, "Impact of Local Interconnects on Timing and Power in a High Performance Microprocessor," in *Proceedings of ACM International Symposium on Physical Design*, 2010.

[2] J.-M. Chang and M. Pedram, "Energy Minimization Using Multiple Supply Voltages," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 5, no. 4, pp. 436 – 443, 1997.

[3] Z. Gu, Y. Yang, J. Wang, R. Dick, and L. Shang, "Taphs: Thermal-Aware Unified Physical-Level and High-Level Synthesis," in *Proceedings of IEEE Asia and South Pacific Design Automation Conference*, pp. 879 – 885, 2006.

[4] H.-Y. Liu, W.-P. Lee, and Y.-W. Chang, "A Provably Good Approximation Algorithm for Power Optimization Using Multiple Supply Voltages," in *Proceedings of IEEE/ACM Design Automation Conference*, pp. 887 – 890, 2007.

[5] T. Lin, S. Dong, B. Yu, S. Chen and S. Goto, "A revisit to voltage partitioning problem", in *Proceedings of the IEEE/ACM Symposium on Great Lakes Symposium on VLSI*, pp. 115 – 118, 2010.

[6] C. Seiculescu, S. Murali and G. De Micheli, "NoC Topology Synthesis for Supporting Shutdown of Voltage Islands in SoCs", in *Proceedings of IEEE/ACM Design Automation Conference*, 2009.

[7] A.V. Sathanur, L. Benini, A. Macii, E. Macii and M. Poncino, "Multiple power-gating domain (multi-VGND) architecture for improved leakage power reduction", in *Proceedings of the International Symposium on Low Power Electronics and Design*, 2008.

[8] Q. Ma and E.F.Y. Young, "Voltage Island-Driven Floorplanning", in *Proccedings of IEEE/ACM International Conference on Computer-Aided Design*, 2007.

[9] D.E. Lackey, P.S. Zuchowski, T.R. Bednar, D.W. Stout, S.W. Gould and J.M. Cohn, "Managing power and performance for System-on-Chip designs using Voltage Islands", in *Proccedings of IEEE/ACM International Conference on Computer-Aided Design*, 2002.

[10] J. Wang and S. Hu, "The Fast Optimal Voltage Partitioning Algorithm For Peak Power Density Minimization", in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, 2010.

[11] V.V. Vazirani, *Approximation Algorithms*, Springer, 2001.

[12] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, 1979.

[13] W.-P. Lee, H.-Y. Liu and Y.-W. Chang, "Voltage-Island partitioning and floorplanning under timing constraints." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 5, pp. 690 – 702, 2009.

[14] S. Kulkarni and D. Sylvester, "Power distribution techniques for dual VDD circuits," in *Proceedings of IEEE Asia South Pacific Design Automation Conference*, pp. 838 – 843, 2006.