行政院國家科學委員會專題研究計畫　成果報告

子計畫七：積極低功率最佳化下之平面規劃(3/3)

計畫類別：整合型計畫
計畫編號：NSC94-2220-E-009-020-
執行期間：94 年 08 月 01 日至 95 年 07 月 31 日
執行單位：國立交通大學電子工程學系及電子研究所

計畫主持人：陳宏明
共同主持人：周景揚

報告類型：完整報告
報告附件：出席國際會議研究心得報告及發表論文
處理方式：本計畫可公開查詢

中　華　民　國 95 年 10 月 30 日

National Science Council Funded Research Report Design and Automation for Low Power Systems

Subproject 7: Floorplanning with Aggressive Power Optimization

NSC 94-2220-E-009-020-
November 1, 2003 - July 31, 2006
Principal Investigator: Prof. Hung-Ming Chen
Co-Principal Investigator: Prof. Jing-Yang Jou
Department of Electronics Engineering
National Chiao Tung University, Hsinchu, Taiwan
Email:hmchen@mail.nctu.edu.tw

Abstract — This report is to present some results of the project funded by National Science Council from 2003 to 2006. Our objectives are mainly to explore and find new techniques in physical design, including floorplanning/placement and clock tree physical synthesis, for low power demand. In this report, we will demostrate the effectiveness of our execution in three years [42], [34], [9], including additional publications [10], [11], [23], [54], [53]. In the first part, we show some results in improving clustered voltage scaling by better power-timing slack sensitivity strategy [42]. We further generate voltage islands for low power designs with performance constraints consideration in second part [34]. As for the third part, we present a methodology for low power clock tree synthesis by transition time manipulation from library study [9].

I. A More Effective Power-Timing Slack Sensitivity Metric in Dual VDD Assignment of Low Power VLSI Arithmetrics

Power consumption problem has been critical for a long time. It increases the design difficulty for battery powered applications, and also affects ordinary designs in terms of time to market, cost, and reliability. Clustered Voltage Scaling (CVS) is an effective way to reduce IC power consumption. CVS utilizes the excess time slacks inside circuits and trade them for power reduction. Methods based on CVS for saving power have been studied for years. In this work, we study the previous approaches and propose an improved CVS method called Bilateral CVS (BCVS). BCVS is a general Clustered Voltage Scaling method which subsumes both CVS and ECVS (also GECVS), and also includes a more effective priority criterion metric in power-timing slack sensitivity. The experimental results show that previous CVS approaches, especially GECVS [30], save substantial power in some arithmetic cell-based designs. Among all approaches, BCVS outperforms GECVS 13% power saving in a 32-bit multiplier under our experimental setup with level converter insertion consideration.

A. Background and Objective

Power dissipation is an important design parameter in the design of microelectronic circuits nowadays, especially in portable computing devices and personal communication applications, also in battery powered applications. A design might be considered not valuable because it consumes too much power. Thus the low power skills become more significant than before [14]. Since dynamic power dissipation in CMOS circuits is directly proportional to the square of the supply voltage (VDD), reduction in VDD can lower the power dissipation considerably. Voltage scaling, sometimes with dual threshold voltage [5], [2], [29], [28], is one of the most effective techniques in reducing the power consumption of CMOS circuits. However, decreasing VDD leads to increase in circuit delay. In the designs of most microprocessors or ASIC chips, the operating frequency is set by the design specification according to the target market. The timing constraints in chips are in turn set by the operating frequency. Designers need to optimize designs to reduce power consumption within the specified timing constraints. If the supply voltage is reduced while V_{TH} remains constant, the critical-path delay will not meet the timing constraints (Fig.1).

Clustered voltage scaling (CVS) is a technique which partially reduces the supply voltage. It utilizes the excess time slack within circuits and then trades the time slack for power saving. Methods based on CVS for saving power have been studied for years, including [46], [48], [30], [47], [8]. [46] was one of the original papers discussing CVS. Due to some limitation in inserting level converter [22], [48] improved CVS in inserting non-flip-flop-type level converters. Not greedily enough, [30] found a way to assign power-timing slack sensitivity (propagation priority) to further improve power saving. However, due to mixed information in the sensitivity assigned for dual VDD assignment, it will not obtain tremendous power saving at all times.

In this work, we propose an improved power-timing slack sensitivity strategy for dual VDD assignment. Our main contributions are as follows. First, we have studied previous

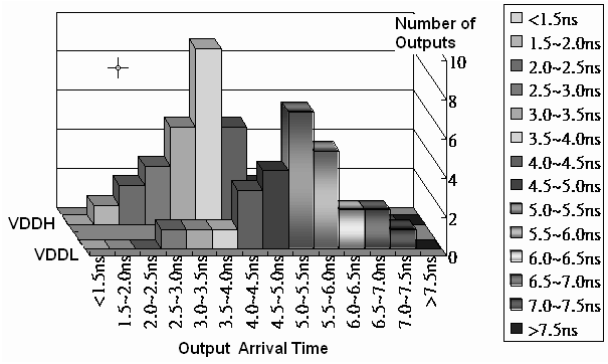

Fig. 1. Output timing distribution of some design in our experimental platform. It shows the migration of output arrival time and this causes the difficulty in designing dual VDD systems.

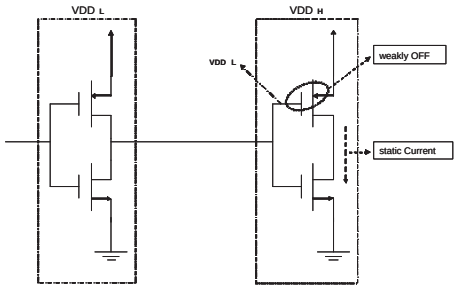

Fig. 2. There exists static weakly-on leakage current in CVS technique [48]. This makes necessary to insert level converters/shifters [22].

CVS techniques and experimented on some arithmetic circuits under our experimental setup. We further demostrate the effectiveness of those approaches, like in [30]. Second, due to our improved propagation priority from [30], we obtain more power saving with our greedy based bilateral CVS (BCVS). Among all approaches, BCVS outperforms GECVS 13% power saving in a 32-bit multiplier under our experimental setup with level converter insertion consideration.

B. Clustered Voltage Scaling Techniques

In the following subsections, we briefly describe three earlier VDD assignment methodologies for low power design and give our problem formulation.

1) Cluster Voltage Scaling (CVS): As shown in Fig. 1, the output arrival time of a circuit usually distributes over a range. After lowering down the supply voltage for low power operation, the output arrival time migrates to a slower range. If the required timing constraint lies on an interval, such as 5.5$\sim$6ns in the shown case, we might fail to find a low power solution in total supply scaling down (voltage island [31] for a subcircuit).

Clustered Voltage Scaling (CVS), firstly proposed by Usami et al. [46], is a simple and practical technique for low power design. The essence of CVS is based on the utilization of excess timing slack in synchronous circuits. It relies on the inner excess time slack inside circuit blocks. Since most circuits have a critical path and other non-critical paths, we usually have the opportunity to minimize power consumption by virtue of CVS.

However we can not make a gate supplied by VDD_L directly fan out to another gate which is supplied by VDD_H. As shown in Fig. 2, the sub-threshold current, or even worse, a static turn-on current, would nullify the efforts done to power saving. We need level converters [22] to shift up signal voltage level so as to drive the succeeding logic gates. Unfortunately, such circuits are relatively large and power consumptive. They form the main overhead of clustered-type multiple-supply-voltage (MSV) low power designs when we try to drive VDD_H gates with VDD_L gates for possibly more power saving.

Usami et al. used a kind of specially designed flip-flop with built-in level conversion (LCFF) in their CVS technique [46] . To save the overhead induced by level converters, the original CVS paper proposed an algorithm that performs Depth-First-Search (DFS) from each output pins backward toward the input pins to achieve a converter-free solution.

2) Extended Clustered-Voltage-Scaling (ECVS): Usami et al. had proposed two ways to improve CVS in [48] due to the limitation of using level converters between flip-flops. First, they allowed the insertion of level converter. ECVS algorithm extends CVS algorithm with a hill-climbing possibility. The VDD_L assignment to some cell on the path from one flip-flop to another flip-flop was performed if it was feasible (considering the cost of level converter insertion, if necessary) and the total power consumption increment was within a margin, apply it. Second, they applied the concept of the stage level of gates, instead of original DFS operation, as a new way to decide the order of VDD_L assignments. They labeled gates and sorted the labeled number as the priority of VDD_L assignment.

3) Greedy-ECVS (GECVS): More recently, Kulkarni et al. proposed a way to further improve ECVS in [30]. They put emphasis on the priority of VDD_L assignment. They introduced a concept of power-timing slack sensitivity measurement for further power minimization.

They defined the sensitivity of a gate 'x' as:

$$Sensitivity_x = \frac{\Delta Power \times slack\ at\ gate\ output}{\Delta Delay} \quad (1)$$

where

ΔPower = Change in total power due to move, and

ΔDelay = Change in arrival time at gate output due to move

They pointed out a concept that we can exploit the movements (from VDD_H to VDD_L) according to the best power savings per unit delay penalty. This is a good idea which directly targets at the primitive goal of CVS: trade

the excess delay for power saving. Intuitively, this sensitivity measurement seems to give a perfect and non-improvable guideline. In the next section, we provide a better approach to further lowering power consumption based on sensitivity measurement in cell-based design.

4) Problem Formulation: We formulate our problem as follows. We want to find the best power saving without violating the timing requirements. The objective is to trade the excess time slacks for most power . We have set up the timing requirements by the Back-roll ratio. The Back-roll ratio (backoff in [30]) means the percentage of increment of the critical path delay. For example, if the Back-roll ratio is 10%, that means the timing requirement is set to 1.1 times the critical path delay. The default value of Back-roll ratio is 0, meaing the timing requirement is equal to the critical path delay.

C. Low Power Design via an Improved CVS and More Effective Power-Timing Slack Sensitivity Metric

Original CVS does not require any insertion of stand-alone level converters. Therefore, it is a more practical approach than ECVS, especially when the overheads of level converters were still high. As the research and improvement in level converter design, the overheads of level converters are lowered. We can then utilize more excess slacks by ECVS if the circuit structure and the timing specification allow. Furthermore, GECVS gives a guideline on how to trade slacks for power in an efficient way. In this section, we show an improved approach to implementing Clustered Voltage Scaling, called BCVS.

The term ”bilateral” means that we push our clusters both from the output side and input side. The motivation is that we want to try to push the clusters from both sides alternatively for more possibility to reach the optimal solution. Originally, we try to push both of the wave fronts just n-levels in each step. But the experimental data show that if n is small, the resulting quality is deteriorated. Therefore, we let n be very large so that the optimality for each wave front is not sacrificed by the action of push of other ones.

The BCVS algorithm and flow are shown in Fig. 3 and Fig. 4. We start our optimization procedure firstly from the output side and grow the cluster as large as possible if slacks allow. During the wave front traversing on circuit, we mark the best movement sequence of power reduction. As it is finished, we push the other wave front from the input side in the same way. After one such iteration is completed, we compare the results. If the solution is better than the previous optimal results, we re-apply the sequence of movement to the marked position and then go on the next iteration.

We utilize a wave front propagator as the engine of our optimizer. As shown in Fig. 5, the wave front starts from the output pins, propagates to the fan-in cells if the timing slacks allow. We also implement a reverse wave front which behaves symmetrically to the ordinary wave front. It starts from the input pins, propagates to the fan-out cells, and automatically includes level converters if necessary. Another characteristic

```
put all output nets into the wave front;
put all input nets into the reverse-wave front;
ini_wave_front = wave front;
ini_reverse_wave_front= reverse_wave_front;
MIN_POWER=total_power;
do{
    wave front= ini_wave_front; power=MIN_POWER;
    while(propagation is possible){
        propagate from output side to input side;
        if(update_total_power( )< power){
            mark the movement in the moving sequence;
            power=total_power;
        }
    }
    undo all movements;
   reverse_wave_front= ini_reverse_wave_front;
    reverse_power=MIN_POWER;
    while(reverse_propagation is possible){
        reverse_propagate from input side to output side;
        if(update_total_power ( )< reverse_power){
            mark the movement in the moving sequence;
            reverse_power=total_power;
        }
    }
    undo all reverse_movements;

    if(power or reverse_power < MIN_POWER){
        update MIN_POWER;
        redo corresponding movement to marked position;
        update both wave fronts (and initial wave fronts );
    }
    else break;
}while(better solution is found);
```

Fig. 3. Proposed BCVS algorithm. This algorithm tries both sides greedily, more aggresive than ECVS and GECVS. The priority assignment for finding better solutions is more effective than previous CVS approaches as well.

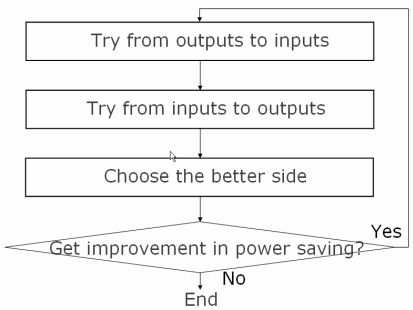

Fig. 4. Methodology flow for lower power consumption using BCVS.

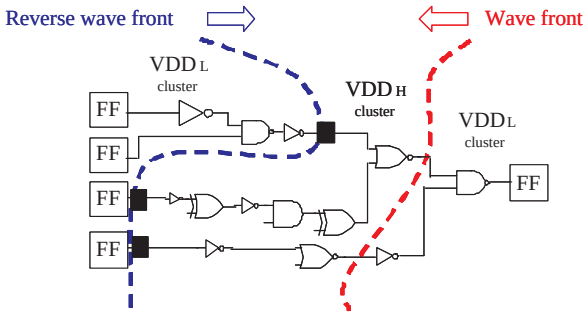

Fig. 5. Illustration of bilateral wave fronts. It saves power by more aggresive VDD_L assignment from both sides alternatively.

of this algorithm is to use different propagation priority. In [30], it uses single key: $slack*\frac{\Delta Power}{\Delta Delay}$, with decreasing order. The one we use is double key: ($slack$, $\frac{\Delta Power}{\Delta Delay}$), both with increasing order.

The criterion proposed by GECVS multiplies $\frac{\Delta Power}{\Delta Delay}$ with slack, therefore the information of slack is blurred. It can not determine whether the cell has a large slack or a large $\frac{\Delta Power}{\Delta Delay}$. As we know, the slacks carry information about the circuit topology, so we can use it as an observer of topology/timing behavior of the circuit. However, our target is the most power saving rather than the largest slack utilization. That is why we need two keys, one observe the topology and timing, the other measure the location of most power saving. In general, if a function α can be a good measurement of the topology/timing information for the propagation algorithm, while our final target is to get the most change in function β, we should use α as the primary key and $d\beta/d\alpha$ as the secondary key. That is the reason that we propose slack as the first key and $\frac{\Delta Power}{\Delta Delay}$ as the secondary key in BCVS. This can be verified by our experimental results.

D. Experimental Results

We set our target to find the most power-saving solution on the condition that the maximal input to output arrival time between all the I/O pins remains the same. As described in the problem formulation, the program automatically gives timing constraint according to the result of the initial Static Timing Analysis (STA). Then it starts to trade the excess timing slack inside the circuit for best power saving and make sure the timing constraint is still satisfied after each movement. For simplicity, we do not aim to the uphill climbing ability about the timing constraint but set our focus on the strategy to exploit all feasible movement without timing violation, and then mark the most power saving solution we have reached. If the uphill movement support is demanded, we can implement it with special care to the evaluation of timing requirements.

To simplify the timing analysis, we set up all gates with the same timing and power parameters. We omit the information about rise/fall transition time at the I/O pin of each gate so as to focus on the slack/power relation to the wave propagation inside the circuitry. The reason is that our primary goal is to exploit all the feasible movements without timing violation, and secondary mark the best sequence with most power saving. We also want to examine the sensitivity criterion proposed by GECVS. We use UMC 0.18 standard cell library and set different leakage power to each type of cells according to this cell library. We set the delay/power of level converters to be multiples of unit gate delay/power, respectively.

We use some real arithmetic designs as our test bench. They are listed in Table I. We have tried three types of propagation priority, then compared the results with CVS approach. The first one is single key: $slack*\frac{\Delta Power}{\Delta Delay}$, with decreasing order, which stands for the GECVS algorithm. The second is double key: ($slack$, $\frac{\Delta Power}{\Delta Delay}$), both with increasing order, which is the one we proposed. The last one is double key: (slack, $fan_{in/out}number$), both with increasing order. The reason for choosing minimal fan number is that we want to do least perturbation to the slack distribution of the whole circuit after each VDD_L assignment.

TABLE I
DESCRIPTIONS OF TESTING CIRCUITS

Circuit name	Circuit function	# standard cells
cla128	128-bit carry look-ahead adder	1911
csm128	128-bit conditional sum adder	1701
add_bk128	128-bit BK adder	1942
mult32	32-bit Booth multiplier	3418

TABLE II
COMPARISON BETWEEN FOUR CVS TECHNIQUES APPLIED ON CIRCUITS IN TABLE I. THE EXPERIMENTAL SETUP IS: VDD_H=1.8,VDD_L=1.2,V_{TH}=0.5, LEVEL CONVERTER DELAY/POWER COST: 1.0/1.0, BACK-ROLL=0%. THE RESULTS SHOW THAT OUR APPROACH CAN OBTAIN MORE POWER SAVING UNDER MODERATE COST OF LEVEL CONVERTER.

Circuit name		original	CVS	GECVS	(slack, $\frac{\Delta P}{\Delta D}$)	(slack,fan)
cla128 cpd=2ns	#H cells	1911	1210	1210	1210	1210
	#L cells	0	701	701	701	701
	#LCs	0	0	0	0	0
	AOAT (ns)	1.51	1.66	1.66	1.66	1.66
	Power (%)	1.0	0.84	0.84	0.84	0.84
csm128 cpd=1.5ns	#H cells	1701	872	872	241	231
	#L cells	0	829	829	1460	1470
	#LCs	0	0	0	203	196
	AOAT(ns)	1.20	1.30	1.30	1.42	1.42
	Power (%)	1.0	0.74	0.74	0.65	0.64
add_bk128 cpd=1.4ns	#H cells	1942	721	930	721	721
	#L cells	0	1221	1012	1221	1221
	#LCs	0	0	0	0	0
	AOAT (ns)	1.09	1.27	1.24	1.27	1.27
	Power (%)	1.0	0.71	0.73	0.71	0.71
mult32 cpd=3.1ns	#H cells	3418	3085	2829	1712	1588
	#L cells	0	333	589	1706	1830
	#LCs	0	0	344	550	499
	AOAT (ns)	2.73	2.89	2.94	2.98	2.98
	Power (%)	1.0	0.97	0.90	0.82	0.81

In Table II-IV, cpd means critical path delay, LC means level converter, $AOAT$ stands for average output arrival time, and $\frac{\Delta P}{\Delta D}$ is $\frac{\Delta Power}{\Delta Delay}$ in Section B.3. The effectiveness of all CVS approaches is shown on those tables, based on different setups in VDD_H, VDD_L, V_{TH}, LC cost, and back-roll ratio. We have observed that there are two phenomena worth mentioning. First, in Table III, the performance of GECVS in add_bk128 seems to be too bad. The reason is that GECVS mixed up the information of timing slack with $\frac{\Delta Power}{\Delta Delay}$ and

TABLE III

COMPARISON BETWEEN FOUR CVS TECHNIQUES APPLIED ON CIRCUITS IN TABLE I. THE ORIGINAL CIRCUIT PARAMETERS, CRITICAL PATH DELAY, AND THE EXPERIMENTAL SETUP ARE THE SAME AS IN TABLE II EXCEPT FOR LEVEL CONVERTER DELAY/POWER COST IS 4.0/4.0. THE RESULTS SHOW THAT ALL CVS TECHNIQUES CAN NOT GAIN ANY POWER SAVING FROM LEVEL CONVERTER INSERTION DUE TO HIGH COST OF LEVEL CONVERTER.

Circuit name		CVS	GECVS	(slack, $\frac{\Delta P}{\Delta D}$)	(slack,fan)
cla128	#H cells	1210	1210	1210	1210
	#L cells	701	701	701	701
	#LCs	0	0	0	0
	AOAT (ns)	1.66	1.66	1.66	1.66
	Power (%)	0.84	0.84	0.84	0.84
csm128	#H cells	872	872	872	872
	#L cells	829	829	829	829
	#LCs	0	0	0	0
	AOAT (ns)	1.30	1.30	1.30	1.30
	Power (%)	0.74	0.74	0.74	0.74
add_bk128	#H cells	721	930	721	721
	#L cells	1221	1012	1221	1221
	#LCs	0	0	0	0
	AOAT (ns)	1.27	1.24	1.27	1.27
	Power (%)	0.71	0.73	0.71	0.71
mult32	#H cells	3085	3085	3085	3085
	#L cells	333	333	333	333
	#LCs	0	0	0	0
	AOAT (ns)	2.89	2.89	2.89	2.89
	Power (%)	0.97	0.97	0.97	0.97

TABLE IV

COMPARISON OF DIFFERENT PARAMETER SETUP ON $mult32$ APPLYING FOUR CVS TECHNIQUES. THE ORIGINAL NUMBERS AND CRITICAL PATH DELAY FOR $mult32$ ARE ALL THE SAME AS IN TABLE II. THE RESULTS SHOW THAT OUR APPROACH OUTPERFORMS OTHER CVS TECHNIQUES EVEN MORE UNDER RELAXED TIMING MARGIN. BCVS OBTAINS 13% POWER SAVING IMPROVEMENT OVER GECVS IN THE SETUP OF BACK-ROLL RATIO = 20%.

Setup		CVS	GECVS	(slack, $\frac{\Delta P}{\Delta D}$)	(slack,fan)
Vdd_h=1.8	#H cells	3085	2307	1410	1214
Vdd_l=1.2	#L cells	333	1110	2008	2204
V_{TH}=0.5	#LCs	0	411	447	463
LCD/PC:0/0	AOAT (ns)	2.89	2.92	2.97	2.96
back-roll=0%	Power (%)	0.97	0.72	0.63	0.65
Vdd_h=1.8	#H cells	3231	3229	2096	2033
Vdd_l=0.9	#L cells	187	189	1322	1385
V_{TH}=0.4	#LCs	0	119	568	561
LCD/PC:1/1	AOAT (ns)	2.94	2.97	3.05	3.04
back-roll=0%	Power (%)	0.97	0.95	0.82	0.83
Vdd_h=1.8	#H cells	2314	1940	1115	1148
Vdd_l=1.2	#L cells	1104	1478	2303	2270
V_{TH}=0.5	#LCs	0	542	350	361
LCD/PC:1/1	AOAT (ns)	3.16	3.26	3.24	3.23
back-roll=10%	Power (%)	0.88	0.77	0.65	0.67
Vdd_h=1.8	#H cells	1003	1577	695	778
Vdd_l=1.2	#L cells	2415	1841	2723	2640
V_{TH}=0.5	#LCs	0	406	206	181
LCD/PC:1/1	AOAT (ns)	3.39	3.51	3.43	3.56
back-roll=20%	Power (%)	0.63	0.61	0.54	0.55

the key of selection criterion. It can not make the right decision that the cells with larger timing slack should have higher priority to VDD_L assignment. Therefore GECVS had detected a larger $\frac{\Delta Power}{\Delta Delay}$ while the actual delay remains constant. This is the reason for the unexpected results. Second, in Table IV, the (slack,fan) priority key obtains much more number of VDD_L cells than the (slack, $\frac{\Delta Power}{\Delta Delay}$) key. But the final power ratio seems to be inconsistent. The reason is that the (slack,fan) criterion can not detect the difference in power saving between cells.

E. Conclusion

We have studied various CVS techniques and successfully improved Clustered Voltage Scaling technologies by assigning better priority/sensitivity. Through well-defined cost function, we have shown that our priority criterion embedded in BCVS outperforms the one defined in GECVS in real VLSI arithmetics.

Our future works include trying to upgrade our optimizer so that we can perform STA with more practical precision. The short circuit power contributes a large portion of total power consumption. However, to analyze this effect, we need more precise timing analysis to evaluate transition time and its sensitivity. Such a work requires much more efforts, especially if we want to merge it into our algorithms in an efficient way. Also we will apply our techniques to other types of circuits to take advantage of saving more power.

II. PERFORMANCE CONSTRAINTS AWARE VOLTAGE ISLANDS GENERATION IN SOC FLOORPLAN DESIGN

Using voltage island methodology to reduce power consumption for System-on-a-Chip (SoC) designs has become more and more popular recently. Currently this approach has been considered either in system-level architecture or post-placement stage. Since hierarchical design and reusable intellectual property (IP) are widely used, it is necessary to optimize floorplanning/placement methodology considering voltage islands generation to solve power and critical path delay problems. In this work, we propose a floorplanning methodology considering voltage islands generation and performance constraints. Our method is flexible and can be extended to hierarchical design. The experimental results on some MCNC benchmarks show that our method is effective in meeting performance constraints and simultaneously considers the tradeoff between power routing cost and the assignment of supply voltage in modules.

A. Background and Objective

To cope with the increasing System-on-a-Chip (SoC) design complexity, hierarchical design and reusable IP (Intellectual Property) modules are widely used [13], [51]. Meanwhile, increased circuit density and performance compel the need to reduce power consumption that increases significantly as designers strive to utilize the advancing silicon capabilities [26], [36]. Since the early stage of design will determine the overall chip performance, an efficient and effective power-aware floorplanning/placement approach is needed to improve the quality and reduce the design cycle.

One of the techniques to reduce power consumption is Voltage Island methodology, proposed from IBM [32]. A voltage island is a group of on-chip cores powered by the same voltage source, independently from the chip-level voltage supply. This concept in use of voltage islands permits

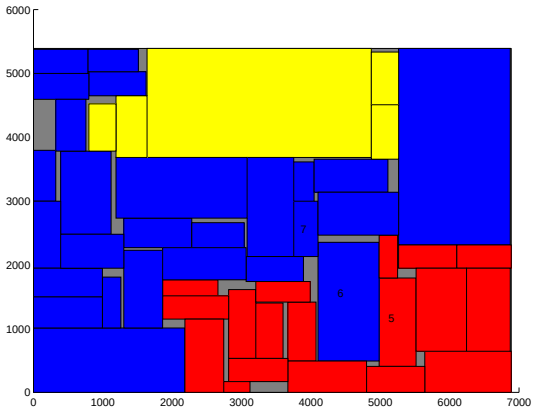

Fig. 6. A resultant floorplan $ami49$ from our approach which generates voltage islands with performance constraint consideration. Blocks 5, 6 and 7 are under performance constraints and they are placed on different voltage islands.

operating different portions of the design at different supply voltage levels.

Voltage island architecture can achieve power saving and has become more and more popular [4], [24], [32], [55], [25]. In [24], [25], iterative voltage island partitioning and floorplanning approach is used, but the exploration of solution space is somewhat restricted. In [55], a post-placement approach to generating voltage islands is proposed. However, chip floorplanning level has more flexibility. Moreover, since timing convergence is an important issue in deep submicron (DSM) design, the critical delay should be bounded. Therefore floorplanning with performance constraints is a necessity [43].

In this work, we propose a methodology to preserve good voltage islands property, which can be viewed as the clustering of modules with same operating supply voltage in achieving lower power consumption. We adopt B*-tree [6] as our floorplan representation and underlying implementation since B*-tree has provided very good quality of non-slicing floorplans in area and wirelength costs, plus some properties for voltage islands generation. Our methodology can save power consumption and routing cost by location constraint [6], and to solve the critical delay problems by performance constraint consideration [56]. Our main contributions include:

- Generate voltage islands in chip floorplanning stage to have more flexibility in design.
- Simultaneously consider voltage islands generation and performance constraints, illustrated in Fig. 6.
- Use one-stage floorplan packing methodology, which can explore more solution space.
- Meet the performance requirements while reducing the cost of power routing complexity.

B. Voltage Islands Architecture and Performance Constraints in Chip Level Floorplanning

In this section, we briefly review the B*-tree representation, concepts of voltage islands, and performance constraints in floorplanning. The problem is then formulated.

1) Review of B-tree Representation:* A B*-tree [6] is an ordered binary tree for modeling a nonslicing floorplan. Given a B*-tree, we can also obtain an admissible placement by packing the blocks in linear time with a contour structure [21]. We adopt B*-tree [6] as our floorplan representation and underlying implementation due to its good quality of non-slicing floorplans in area and wirelength costs, plus some properties for voltage islands generation.

2) Voltage Islands Methodology: The combination of increasing active power density and leakage currents has created a power management problem in the semiconductor industry. Mostly performance-critical element of the design requires the highest voltage level to maximize performance, while other coexisting functional cores may not need this voltage level, hence they can be run at lower voltages to save significant active power. This idea enables the concept of voltage island architecture[32].

Introducing voltage islands concept makes the chip design process even more complicated with respect to static timing and power routing. The cores powered by the same voltage source should be grouped together without violating design metrics such as timing and wire congestion. Meanwhile, the number of voltage islands should be appropriate (not too many) considering signal translation and communication between different islands, which requires level converters. We also need to consider power routing complexity [24] for design cost. Hence the overhead for applying voltage islands methodology with respect to area and delay is inevitable.

3) Performance Constraints Consideration in Floorplanning: Performance is a concern since the interconnect delay dominates the circuit performance for DSM VLSI design. Minimizing total wire length, as traditional floorplanners/placers did, can not guarantee bounded delay for critical nets. It is desirable to minimize the critical net delay by binding them together to optimize performance or to meet the delay constraints by placing them close enough to each other. The constraint requires designated nets (blocks/cores) to be placed within a pre-defined bounding box nets. In [43], the maximum delay of performance constraint blocks is bounded by the summation of its height and width of the bounding box enclosing those blocks. However it is not trival to bound the maximum delay for those performance constraint blocks in voltage island architecture, especially for those which are not in the same voltage island.

4) Problem Formulation: For voltage island planning, we use a simplified model for modules/IPs, based on the setup in [24]. Since the power consumption of an IP varies with different supply voltage, we use a power table, which is a list of matching pairs, *(supply voltage, power dissipation)*, specifying the legal voltage levels to work functionally and the corresponding average power dissipation values, for every

IP. We set this power dissipation based on IP's timing constraint and circuit size.

The problem concerned is as follows. Let B=$\{b_1$,b_2,...,$b_n\}$ be a set of n rectangular modules whose width, height, and area are denoted by W_i, H_i, and A_i, $1 \leqq i \leqq n$. Let (x_i,y_i) denote the coordinates of the bottom-left corner of module b_i, $1 \leqq i \leqq n$, on a chip. Each module is associated with a power table. A floorplan/placement P considering the performance constraint and voltage islands generation is an assignment of (x_i,y_i) for each b_i, $1 \leqq i \leqq n$, such that cores are clustered using the same voltage to form appropriate number of islands and achieving low power consumption, while no two modules overlap and the given performance constraints are satisfied. The goal is to simultaneously minimize the packing area, power routing cost and total power dissipation, while meeting performance constraints.

C. Performance Constraints Aware Voltage Islands Generation in Floorplan Design

In this section, we propose the heuristics for voltage islands generation with B*-tree representation, then discuss the strategy to consider performance constrained blocks during floorplanning under voltage island architecture.

1) Floorplanning with Voltage Islands Generation: We first give an example to show the setup in creating voltage islands in SoCs using B*-tree and one intuitive strategy. In Fig.7, each core is followed by a number which identifies the number of its usable voltages, then associated with a power table. For instance, the block b_3 can operate at 1.0, 1.1 or 1.2V, and its corresponding power consumption are 1.3mW, 1.8mW and 2.6mW. One obvious way to maximize power saving in floorplanning is to operate each block at its lowest voltage, which means that we need at least 3 voltage islands: one for $\{b_0, b_4, b_7\}$, one for $\{b_1, b_2, b_9, b_{10}\}$, and one for $\{b_3, b_5, b_6, b_8\}$. This arrangement is obviously not optimal since the exploration of solution space is limited and the price of area/wirelength overhead may be very high. Sometimes we may be forced to use more islands, or higher legal voltages to alleviate the problems.

One key observation to create the voltage islands is to constrain the nodes relationship between each pair of nodes which exist the parent-child relationship in the B*-tree representation, which means to cluster the blocks with the same supply voltage (say $compatible$), grouping them to be a subtree in corresponding B*-tree. However, the condition that two nodes do not abut in the tree does not always mean that the corresponding two blocks abut. Similarly, the condition that nodes are not in the same subtree does not mean they do not abut physically. We give an example in Fig.8. We believe that it is more practical to increase the probability of those nodes to be clustered together, then apply a simple checking method to inspect if they really form a favorable island.

From above observation, we know that the area cost is getting lower and the dead space of the total area is becoming smaller due to B*-tree module packing, there will be a visible mapping relationship that if n_j is the left child (or right

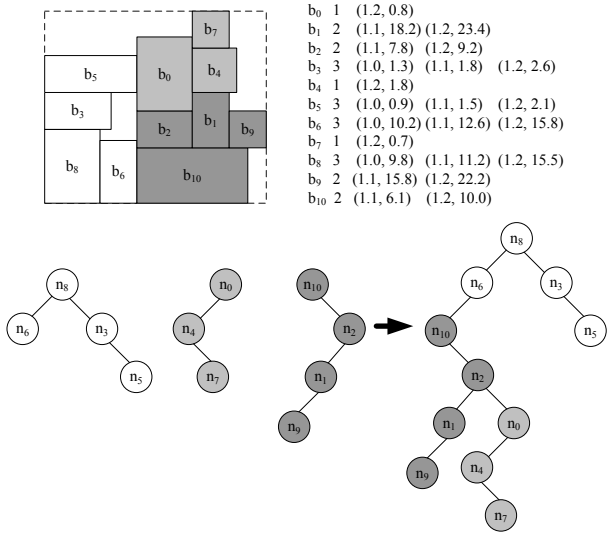

Fig. 7. An illustration of an intuitive approach to generate voltage islands in chip-level design. We partition the blocks by their lowest supply voltage, construct the subtrees of those compatible blocks, then build the B*-tree and the corresponding floorplan. This approach will seriously limit the exploration of the solution space.

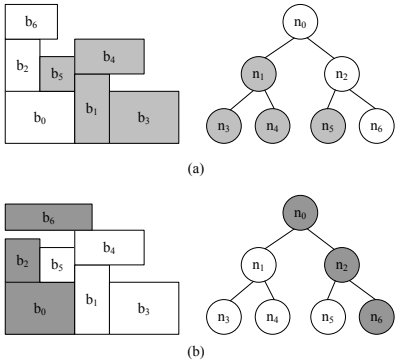

Fig. 8. In (a), node n_5 is not in the samesubtree with $\{n_1, n_3, n_4\}$; but in the floorplan, block b_1, b_4 and b_5 are connected. In (b), nodes $\{n_0, n_2, n_5\}$ form a subtree in the B*-tree; but in the floorplan, block b_2 is not connected with block $c6$. There are extra area overhead in this voltage island.

child) of n_i in the B*-tree representation, then the block b_j right (or left) abuts to block b_i. The probability a node adds to a compatible subtree and the subtree grows and maps to a favorable voltage island shape will be increased. To implement this idea, we first randomly choose two nodes n, p in the tree ($V(n)$ and $V(p)$ denote the adopted voltages of node n and node p), if the following conditions appear, we change the positions of these two nodes.

- $V(p) = V(n)$: Node p and node n are compatible, No good voltage island property will be ravaged.
- $V(p.parent) = V(n)$: Node p's parent and node n are compatible, let n be the leaf of the subtree or connect two compatible subtrees to a larger subtree.
- $V(p.leftchild) = V(n)$ and $V(p.rightchild) = V(n)$:

Node p's left child and right child both have the same voltage with node n, let n be the root of the subtree or connect two compatible subtrees as well.

Except these conditions are considered for perturbation during simulated annealing, we modify two following operations in the B*-tree algorithm.

- Delete_Node: If we want to delete node n, we adjust the supply voltage of the child node ($n.leftchild$ or $n.rightchild$) so that it is compatible with node n's parent. If the children are both compatible or both not compatible, we randomly choose one of them.
- Insert_Node: If node n is to be inserted into the subtree which exists compatible nodes, it will be placed to join the cluster of the compatible nodes. If there does not exist any node compatible, we randomly choose one place to insert.

Since the subtree construction is just a method to increase the possibility in forming a good voltage island property, we need a property checking function to check if there exists a favorable voltage island shape. We do it after the contour updated to make sure the voltage island property is acceptable.

2) Floorplanning with Performance Constraints Blocks in Voltage Island Architecture: Traditional floorplanners/placers minimize total wirelength but they can not guarantee critical nets to meet bounded delay. This problem becomes more important because timing convergence is a big issue in DSM design. In order to meet critical delay constraint, there are methods proposed in [43], [56] during floorplanning.

Since actual interconnect delay after appropriate buffer insertions will be close to linear in terms of distance, linear function in terms of distance to estimate delay is used. Assume there are a source at (x_s,y_s) and a sink at (x_t,y_t), their locations are the corner points as far as possible, and the delay of the net $D_{s,t} = \delta(\mid x_t - x_s \mid + \mid y_t - y_s \mid)$, where δ is a constant to scale the distance to timing, $D_{s,t}$ is the maximum distance between source and sink, equal to the half perimeter of the bounding box of the two blocks. In [56], it uses the delay model to do sub-placement (to place a set of feasible sub-placements for the performance blocks) and they can get some rectilinear super blocks that the width W_{perf} and height H_{perf} satisfy the performance constraint: $W_{perf} + H_{perf} = B \leq B_{max}$, where B_{max} is the maximum bounded distance. Among the placements (rectilinear super blocks) meeting the performance constraint, they pick the one with the minimum dead space $S_{perf} = W_{perf} * W_{perf} - \sum A_i$ and fix the rectilinear block (and thus fix the delay) for further processing with other blocks. By using the pre-clustered shape-fixed appropriate rectilinear block, they guarantee that the performance constraint will be satisfied throughout the remaining processing.

There is a major problem in this performance model using voltage island architecture. The performance of each block does not vary with supply voltage. The legal supply voltage has big impact on the driving strength, thus the bounding box size. If signals are communicated by high supply voltage, the bounding box for performance constraint blocks will be the largest. In addition, allowable box size should be a function of the supply voltage. In this work we use the conservative modeling by using largest bounding box size for those performance blocks.

Based on the above discussion in the setup of enclosing bounding box of performance constraint blocks, our approach combines the advantages of the two methods in [56], [43], keeping the flexibility of the sub-placement for the performance constraint. We do not pick the minimum dead space sub-placement and fix the shape (or the relational position) of the performance blocks before processing with other blocks at the beginning. Instead, we let the performance blocks process with other blocks as if they are not under restriction, the total area and wirelength can be better optimized. This is further verified in the condition that supply voltages of the performance constraint blocks are possibly different. If we tighten the shape of performance constraint blocks at the beginning, we may be forced to raise the supply voltages of some of them to the higher one to meet voltage island property; or we will get a disorder B*-tree structure that the voltage property is withered. When the temperature becomes lower in annealing process, we do not allow a solution that violates performance constraints even if it has a better cost, the best solution will be kept until next feasible solution with better cost.

3) The Algorithm: Our floorplanning/placement design algorithm is based on the simulated annealing method and we only consider hard modules in this work. We perturb a B*-tree to another by the following operations:

- $Op1$: Change the supply voltage of a block. (Except that only one supply voltage is available.)
- $Op2$: Rotate a block.
- $Op3$: Flip a block.
- $Op4$: Swap two blocks. (The situations we discussed increase probability to be allowed to do swap, while other situations that wither the subtree property have lower probability.)
- $Op5$: Move a block to another place. (new Delete_Node and Insert_Node)

The first three operations are trivial and almost the same with the original B*-tree. $Op4$ and $Op5$ change the relations of blocks to get a different placement and B*-tree structure based on our heuristics.

D. Experimental Results

We implemented our algorithm in C++ on a PC with P4-2.4GHz cpu and 440MB memory. Our method can handle circuits that have two or three kinds of supply voltages, circuits with more than three supply voltages are applicable as well.

For testing our observation in voltage island generation on large number of blocks, we apply our approach on some of the MCNC benchmarks with more blocks, and compare with [6]. For adopting voltage island architecture, power routing cost and level converter issues should be addressed. We simplified the cost of power routing/overhead area by

using the scaled boundary length of voltage islands except for the boundary side of the chip. The scaling is based on the amount of level converters necessary in the chip [1].

Table V shows the comparison between [6] and our approach on power consumption and power routing cost, where the power consumption in column 5 is lowest since we use the lowest available voltages for every block in it. From Table V, we can see that our power consumption is a little more than the lowest power listed in column 5, but our routing/level converters cost is about 16.4% - 55.2% less when compared with [6]. Fig.9 shows the comparison between two floorplans of $ami33$, with and without voltage islands generation heuristic.

In order to compare our results with [6] and [56] in similar number of voltage islands and power routing cost, we apply an intuitive heuristic that adjusts supply voltage of the blocks from original B*-tree results. In Table VI, we can see that, with almost the same number of voltage islands, at least 10% - 20% power consumption can be saved by our method, not to mention the good shape of the generated voltage islands.

TABLE VI
THE COMPARISON BETWEEN POWER AMOUNT THAT NEED TO BE RAISED TO FORM A FLOORPLAN WITH VOLTAGE ISLANDS. THIS SHOWS OUR APPROACH CAN OBTAIN LOWER POWER VIA VOLTAGE ISLAND METHODOLOGY.

Circuit	Table	Lowest	Ours		Original B*-tree [6]	
			Power	P Inc(%)	Power	P Inc(%)
hp	pt2	83.7	86.4	3.2%	97.9	16.7%
	pt3	73.4	78.3	6.7%	91.6	24.8%
ami33	pt3	113.6	123.2	8.5%	136.8	20.4%
	pt3-1	131.1	136.3	4%	161.7	23.3%
ami49	pt2	147.1	151.5	3%	171.4	16.5%
	pt3	142	156.2	10%	169.6	19.4%
	pt3-1	183.1	196.4	9.7%	239.6	30.1%
	pt3-2	208	222.9	7.2%	254.3	22.3%

Table VII shows the comparison of our results with [56] which considers only alignment and performance constraints. Both methods meet performance constraints but our approach could get much lower cost of level converters with slightly more power consumption. Fig.6 illustrates final floorplanning result of $ami49$ with performance constraints blocks 5, 6, and 7, and they are not on the same voltage island.

TABLE VII
THE COMPARISON BETWEEN [56] AND OUR APPROACH ON POWER CONSUMPTION AND POWER ROUTING COST. WITH BOTH MEETING PERFORMANCE CONSTRAINTS, OUR APPROACH OBTAINS MUCH LOWER POWER ROUTING COST WITH SLIGHTLY MORE POWER CONSUMPTION.

Circuit	Table	Perf.	Perf. Const. Only[56]				Ours		
			Area	Dead	P(mw)	C	Area	Dead	P(mw)
ami33	pt3	3	1.181	2.2%	113.6	4.34	1.18	2.02%	121
	pt3-1				131.1	4.93	1.181	2.2%	145.1
ami49-2	pt2	3	36.56	3.1%	147.1	4.5	36.78	3.64%	156
	pt3				142	6.33	36.89	3.93%	154.5
	pt3-1				183.1	6.89	36.87	3.86%	200.9
	pt3-2				208	6.7	36.89	3.93%	221.9
ami49-3	pt2	6	36.64	3.3%	147.1	4.48	36.8	3.68%	156.8
	pt3				142	6.43	36.98	4.14%	149.7
	pt3-1				183.1	6.6	37.1	4.46%	215.9
	pt3-2				208	6.25	37.07	4.38%	223.3

[1]Level converters are only needed when the signals are transmitted from low supply island to high supply island

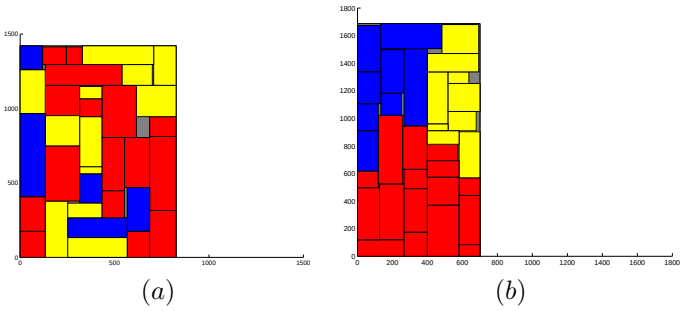

Fig. 9. Two floorplans of circuit $ami33$ with 3 usable supply voltage. (a) Floorplan with much higher power routing complexity since the number of voltage islands is large. (b) Floorplan with nice voltage island property (slightly more power dissipation).

E. Conclusion

We have presented an effective algorithm to deal with the floorplanning with voltage islands consideration and performance constraints. The algorithm is based on the B*-tree representation and the simulated annealing framework. According to the circuit power table information and the idea of location constraint (LC relation), we can group a set of cores using the same supply voltage, obtain appropriate number of voltage islands, and form good shapes of voltage islands. We also take performance constraints into consideration while generating voltage islands.

III. ON ACHIEVING LOW-POWER SOC CLOCK TREE SYNTHESIS BY TRANSITION TIME PLANNING VIA BUFFER LIBRARY STUDY

Clock power dissipation has become a significant issue since it occupies around half of the total system power. Due to high working frequency in modern system designs, the transition time of the clock signal is extremely short. In order to keep up with this trend and to use less wire area, a large number of buffers have to be inserted in the network. As a consequence, short-circuit power of the clock buffers is no longer negligible. In this work, we introduce a methodology which can be applied in global clock tree synthesis to achieve low short-circuit power. It is based on the analysis of any given buffer library in manipulating buffer transition time and hierarchical clustering of loads during buffer insertion. The experimental results are encouraging. Since there are very few works on gate/buffer sizing or buffer library analysis to overcome clocking power problem, we compare our approach with a greedy buffer sizing approach and obtain 13.7% clock power saving for a 10,000 flip-flop design under user-specified clock skew constraints.

A. Background and Objective

Clock designs play an important role in modern VLSI designs. As technology advances, a chip may have millions of gates with a very complex structure. The synchronization of clocks on a chip is critical to the performance and reliability

TABLE V

THE COMPARISON BETWEEN [6] AND OUR APPROACH ON POWER CONSUMPTION AND POWER ROUTING COST. C IS LEVEL CONVERTERS AREA AND ROUTING COST, NORMALIZED TO OUR APPROACH. WITH SLIGHTLY MORE POWER CONSUMPTION, WE CAN OBTAIN MUCH LOWER POWER ROUTING COST IN VOLTAGE ISLANDS GENERATION.

Circuit	Table	Original B*-tree [6]					Ours				
		Area(mm^2)	Dead	P(mw)	C	CPU(sec)	Area(mm^2)	Dead	P(mw)	C	CPU(sec)
hp	pt2	8.95	1.4%	83.7	1.81	4	9.11	3.10%	86.4		15
	pt3			73.4	2.38		9.10	2.98%	78.3		18
ami33	pt3	1.174	1.47%	113.6	4.52	26	1.181	2.07%	123.2		89
	pt3-1			131.1	4.76		1.183	2.23%	136.3	1	89
ami49	pt2	36.8	3.68%	147.1	4.18	53	36.67	3.34%	151.5		243
	pt3			142	5.43		36.68	3.38%	156.2		234
	pt3-1			183.1	6.11		36.75	3.52%	196.4		234
	pt3-2			208	5.97		36.78	3.64%	222.9		240

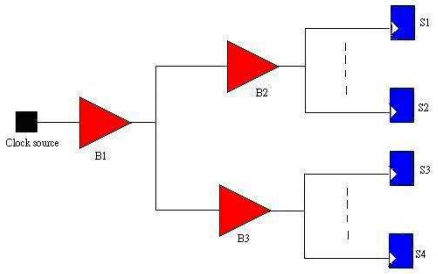

Fig. 10. A buffered clock tree with synchronizing elements {S1,S2,S3,S4} and buffers{B1,B2,B3}.

of the chip. In a synchronous digital system, the clock signal defines the time reference for the movement of data within the system. In fact, clock network can consume 15%-45% of the total system power [40], [15]. Moreover, systems are operating at very high frequencies due to technology advances, and this leads to shorter signal transition time. The transition time has effects on power consumption. Therefore the clock distribution needs more careful design planning methodology in low power for modern VLSI.

Due to a large amount of fan-outs that distribute over long routing distances in clock tree, among clock network designs, the buffered clock tree structure (shown in Fig. 10) is one of the most popular clock network designs adopted in modern VLSI designs. The buffered clock tree is a clock tree which has some buffers inserted between the source and all the registers driven by the clock signals. The major advantages of buffering clock tree is as follows. First, the clock buffers can maintain the quality of the clock waveform. Second, the inserted buffer in a clock path helps reduce the interconnect resistance between clock source and driven register so that the wire RC delay is decreased. Third, buffering can help avoid DRC violation during physical verification. In order to meet timing constraint and to achieve short transition time, we need to insert a lot of buffers along the clock paths. Since short-circuit power is due to simultaneous conduction of the PMOS and NMOS transistors during input transitions [49], [37], this kind of power should be analyzed and accounted for [38].

There are some previous works on low power buffered clock tree construction. [41] and [50] tried to insert buffers, in order to minimize the clock power. Tellez et al. in [44] investigated the problem of computing a lower bound on the number of buffers required in the clock tree, given a maximum transition time constraint. More recently, [38] emphasized that the transition time has become the key factor in low power clock design, and the tradeoff between the power and transition time when optimizing the clock tree was discussed. However, none of them can effectively use buffer library to help reduce power dissipation, especially on short-circuit power, while inserting buffers in clock network.

In this work, we observe that buffer transition time is critical in saving short-circuit power, similar to the conclusion of [38], when inserting clock buffers. We develop a methodology to reduce the power consumption of buffered clock tree by finding the tradeoff between buffer transition time and power via buffer library study. We actually obtain up to 13.7% power saving based on some industrial benchmarks, compared with a greedy buffer sizing approach. The goal is to insert appropriate number of buffers while reducing the total clock power dissipation by bounded transition time optimization strategy.

B. Preliminaries

In this section, we describe some previous works, introduce our power estimation model in clock tree generation, and formulate our clock tree synthesis problem.

1) Previous Works on Clock Tree Synthesis: Previous works on clock tree construction focused on zero or near zero-skew routing, such as symmetric H-tree [1], MMM (method of means and medians) algorithm [27], zero skew routing [45], [17], Deferred-Merge Embedding(DME) algorithm [20], [7], [3], load balancing [35], and simultaneous routing, wire sizing and buffer insertion [33]. Until recently, it is getting more important to reduce the power consumed by the clock network due to higher frequency operation in modern digital systems, including clock gating [19], [18], [15], [16], buffered clock network [41], [50], [39], [52], [44], [38], among which [52] proposes to use sequential linear programming (SLP) to size buffers under general skew constraints for clock power reduction. However, none of them can effectively use buffer library to help reduce power dissipation, especially on short-circuit power, while inserting buffers in clock network. Below we describe clock

power estimation used in this work, followed by our problem formulation.

2) Power Dissipation Estimation of Clock Trees: In a clock tree, wires and cells contribute to the power consumption. For wires, the power $P_{sw}(wire)$ is dissipated by charging and discharging the wire capacitance. We use the formula (1) to estimate the net capacitance for calculating the delay of clock buffers. The equation to estimate the net load C of a driver pin is

$$C_{wire} = \sum_{all\_fanout} wire\_length * \phi \tag{2}$$

where ϕ is the weighting parameter from industrial benchmarks. We can obtain the wirelength by summing up the Manhattan distance of any two connecting cells, which is the net length from driving cell to the driven cell. The power consumption for cells consists of two components: switching power consumption $P_{sw}(cells)$, which corresponds to charging and discharging of the capacitance in cells, and internal (short-circuit) power of cells. We use lookup table based nonlinear power model library in this work to find accurate values of short-circuit power for cells. The estimation of total power is then from the switching power $P_{sw}(cells + wires)$ and internal power $P_{int}(cells)$ (short-circuit power) of cells, where V is supply voltage, f is operating frequency, and C_i is for capacitance in cells and wires (i_{th} element):

$$P_{sw}(cells + wires) = (\sum_{i=1}^{all\_net} C_i) * V^2 * f \tag{3}$$

$$P_{total} = P_{sw}(cells + wires) + P_{int}(cells) \tag{4}$$

3) Problem Formulation:

Problem 1: **Low Power Buffered Clock Tree Construction:** Given a set of sinks (flip-flops) of the circuit S=$\{s_1$,s_2,.....,$s_n\}$ and buffer library, construct a buffered clock tree topology in reducing the power consumption on cells (flip-flops and clock buffers) and wires (wirelength) under specified clock skew constraint.

C. Achieving Low Power Clock Planning by Buffer Library Study on Transition Time Manipulation and Skew Minimization

Due to skew and timing constraints (both delay and transition time), buffer insertion becomes a necessity in clock tree synthesis. However, buffering technique may lead to more power penalty. The following subsections describe our methodology in guiding buffer insertion for further clock power reduction.

1) Buffered Clock Tree Construction Methodology: Here we depict our methodology to construct clock tree to reduce total clock power. First we automatically analyze buffer library to obtain the characteristics of each buffer and find the best transition time for each buffer. We also consider inverter as repeater and take care of phase assignment problem. We decide the number of clusters which is based on the best loading in the circuit via buffer library study. Then we use the clustering algorithm in order to obtain each cluster of approximately equal capacitance loading and further insert buffers to drive identical loading at the same level of clock tree. We also check if overlap occurs at the same time.

Our buffer insertion follows bottom-up fashion, like in [39]. Since transition time calculation follows top-down fashion, we have the following reasonable observations supporting us to do clock tree synthesis in bottom-up way. First, from some experiments, we find that we can minimize the power dissipation waste by setting buffer's input transition time and output transition time the same. In this way, we can also ensure that the transition time is not degraded during propagation. Second, we find that there is not much change in transition time and power after two levels of buffers in hierarchy, starting from clock root buffer, which means that most of clock power is decided in bottom levels (close to sinks). Our algorithm follows these steps:

1) Study given clock buffer library, along with target benchmark, and find best buffer transition time for each buffer and corresponding loading in power minimization
2) Partition given cell-based design based on buffer library study
3) Insert the identical type of buffers at the same level, also check for overlap
4) Go back to second step to bottom-up partition a set of buffers for lower level, and perform the third step to insert higher level buffers, until the root buffer is reached

2) Clustering Clock Tree Sinks for Skew Constraints: In order to avoid clock skews when constructing clock trees, we create clusters for all the sinks and let each cluster has approximate same loading for buffers to drive. The goal is to partition a given set of clock pins/buffers so that each cluster can be driven by appropriate size of buffers. The same type of buffers will be inserted for the same level to maintain the skew minimization in clock tree synthesis. We have implemented a clustering algorithm in [35], which can be used to create clusters for clock tree load balancing. This clustering can be applied hierarchically at different levels of a clock tree. At bottom level, the algorithm clusters flip-flops, while in middle levels of the clock tree clock buffers are clustered for upper levels. As for the number of clusters in a level, it is based on buffer library study presented in Section C.3. The tradeoff between best buffer transition time and skew for clock power optimization is considered as well.

The total load of each cluster can be measured by a cost function:

$$C(each\_cluster) = \sum_{i=1}^{all\_cells} C_i + \beta * D$$

where C_i is the input capacitance of cells, β is the weight term, and D is the diameter of the input set which is defined as the Manhattan distance. Interconnect delays within clusters are concurrently balanced as well, thereby generating a low-

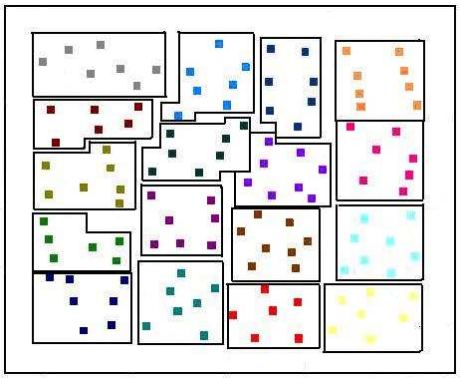

Fig. 11. Flip-Flops are partitioned into clusters of approximately identical loading. We utilize the buffer library analysis result and partition this 123 flip-flop circuit.

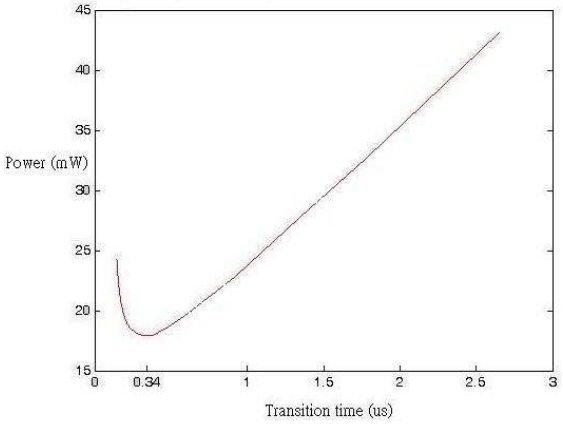

Fig. 12. Different power dissipation for a given load/circuit can be obtained due to different transition time between buffers and flip-flops. We analyze the buffer library and get the best transition time to save power from a convergence of iterative process. When we obtain the best transition time, the number of clusters for the given load is obtained as well.

skew buffered clock tree design. Fig. 11 shows an example of clustering for a 123 flip-flops circuit.

3) Manipulating Inserted Clock Buffer Transition Time by Buffer Library Analysis: [38] found that it is necessary to have a transition time bound to assure the performance of the system. This bound can affect the power optimization result significantly. It is also known that cells consume power differently under different transition time of driving buffers. Starting from those two points, we analyze any given buffer library and try to find the correspondance in short-circuit power and buffer transition time for driving output loading. Our key contribution in this methodology is to find appropriate buffer transition time for saving total clock power.

In order not to insert many buffer to drive loads, we intend to use less number of buffers in clock tree construction, which means the number of load clusters will be less. However, buffers will provide longer transition time for flip-flops, introducing more power dissipated in flip-flop. Conversely, if we have more clusters of loads, which means there are larger amount of buffers in the clock network. This leads to more power consumption in inserting more buffers. Based on those observations, we use the following technique to find appropriate buffer transition time for each buffer. Experimental result is shown in Fig. 12. With a given load and buffer type, we assume an initial input transition time and enumerate all possible number of clusters to computer corresponding power, we can find best transition time for lowest total clock power for the first iteration. Then we use this resultant transition time as input transition time and calculate again, until this process converges (input and output transition times are the same). The cluster size can be obtained for the given load as well. Among all buffers we choose the one with lowest power for these two levels. The time complexity for this library analysis depends on the size of the library and the number of iteration for finding best transition time. Usually it can be found in several iterations.

D. Experimental Results

We have applied this methodology shown in previous section to optimize the power consumption of the clock tree while conforming to the skew constraint and have performed experiments on some real cell-based cases from industry. The buffer library and related parameters are based on 0.18um technology. We also implement another combined approach ([12] and [50]) in order to show the effectiveness of our approach [2].

First, we compared the gate sizing algorithm alone with the clock skew constraint on five benchmarks, number of flip-flops ranging from 100 to 10000. We change the implementation in [12] so that it can handle clock skew constraints. The results show that our approach can achieve low power clock design and it is scalable to larger circuits. The results are shown in Table 1. Furthermore, the approach in [12] may change the size of inserted buffers and may cause the increase of clock skew. Our approach, however, inserts identical size of the buffers in the same level of clock tree, which effectively avoids the skew increase.

Second, we apply the greedy based algorithm [50] in gate sizing approach as much fairly comparison platform. It will generate the initial topology in the clock tree and insert buffer based on greedy approach in the clock tree. However this approach does not consider transition time issue for low power. In Table 2, the results of power consumption have been compared with those two approaches. It is shown that we have obtained averagely 3.14% total power saving.

[2]The approach in [12] is a general gate/buffer sizing algorithm for low power circuit, not specifically for clock tree design. The combined approach from gate sizing and greedy based optimizations is to create more fairly comparison platform, compared with gate sizing alone. The reason we use those approaches for comparison is that there are very few works discussed about gate sizing or buffer library analysis in short-circuit power reduction in clock tree. [52] actually used SLP to solve buffer sizing problem under general skew constriant for clock power minimization, which will take longer time to find solutions.

TABLE VIII
COMPARISON BETWEEN GATE SIZING APPROACH WITH OUR APPROACH IN POWER CONSUMPTION.

		GS	Ours	
Benchmark	Skew constraint(ns)	power(mW)		reduction(%)
Clk_100	0.05	4.48512	4.13609	8.4%
Clk_500	0.08	19.29948	17.51394	10.1%
Clk_1000	0.1	46.80549	42.30692	10.4%
Clk_5000	0.5	188.39265	169.13925	11.38%
Clk_10000	1	377.51768	331.92524	13.73%
Average				10.8%

TABLE IX
COMPARISON BETWEEN OUR APPROACH AND GREEDY BASED ALGORITHM PLUS GATE SIZING ALGORITHM IN POWER CONSUMPTION.

		Greedy+GS	Ours	
Benchmark	Skew constraint(ns)	power(mW)		reduction(%)
Clk_100	0.05	4.19938	4.13609	1.5%
Clk_500	0.08	17.96822	17.51394	2.5%
Clk_1000	0.1	43.62621	42.30692	3.0%
Clk_5000	0.5	176.7925	169.13925	4.3%
Clk_10000	1	347.2925	331.92524	4.4%
Average				3.14%

We further analyze four approaches for power consumption in Fig. 13, where $N * N + GS$ approach is a combined approach with naive geometric $N * N$ grid clusters and aggressive gate sizing. This approach can achieve better low power results than buffer sizing alone algorithm [12]. We find that our approach can save more power consumption and less clock skew is achieved when the number of flip-flops is larger.

E. Conclusion

We further verify that the transition time is one of the key factors for low power clock design. We find that the transition time is important in low short-circuit power design because it affects the power consumption of the cells. We propose to analyze buffer library and insert appropriate number of buffers with transition time manipulation in buffered clock tree synthesis. According to our buffer library study, we attain the smaller short-circuit power between the buffers and the flip-flops, and we ensure the power consumption of each cluster is the best solution by utilizing buffers more effectively. Due to the use of equally-sized buffer at the same level of clock tree, we can generate a nearly zero-skew clock tree. The future works include the application of higher order delay model and practical clock tree routing to further verify the effectiveness of this low power clock tree synthesis methodology.

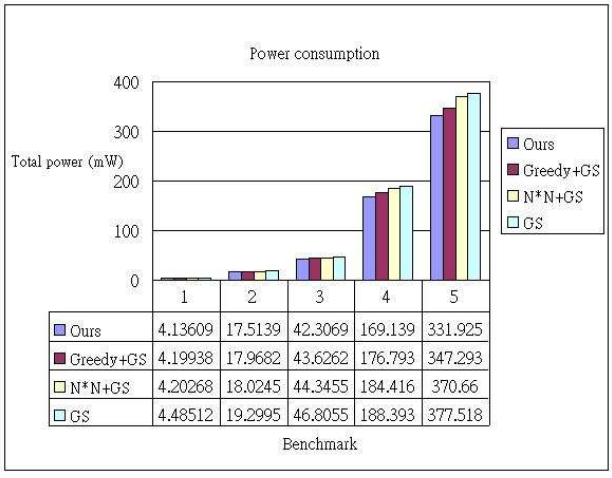

Fig. 13. Clock power reduction comparison between approaches. It shows our approach can obtain lowest power dissipation among natural extensions of buffer sizing and greedy based low power clock generation algorithms.

IV. ENDING NOTES

We really appreciate NSC to fund us for researches on low power in physical design area. We have produced 8 publications as shown in Abstract, including one journal paper. We will continue our efforts in further working on low power methodologies for VLSI and SoC designs, and hope the NSC can still give us supports and comments/concerns.

REFERENCES

[1] H. Bakoglu, J.T. Walker, and J.D. Meindl. "A symmetric clock distribution tree and optimized high-speed interconnections for reduced clock skew in VLSI and WSI circuits,". In *Proceedings IEEE International Conference on Computer Design*, pages 118–122, 1986.

[2] Anirban Basu, Sheng-Chih Lin, Vineet Wason, Amit Mehrotra, and Kaustav Banerjee. "Simultaneous Optimization of Supply and Threshold Voltages for Low-Power and High-Performance Circuits in the Leakage Dominant Era". In *Proceedings IEEE/ACM Design Automation Conference*, pages 884–887, 2004.

[3] K.D. Boese and A.B. Kahng. "Zero-skew clock routing trees with minimum wirelength". In *IEEE International Conference on ASIC*, pages 1.1.1–1.1.5, 1992.

[4] J.-A. Carballo, J.L. Burns, S.-M. Yoo, I. Vo, and V.R. Norman. "A semi-custom voltage-island technique and its application to high-speed serial links". In *Proceedings ACM International Symposium on Low Power Electronics and Design*, pages 60–65, 2003.

[5] Chandrakasan, A.P., S. Sheng, Brodersen, and R.W. "Low power CMOS digital design". *IEEE Journal of Solid-State Circuits*, 27(4):473–484, April 1992.

[6] Y.-C. Chang, Y.-W. Chang, G.-M. Wu, and S.-W. Wu. "B*-trees: A new representation for non-slicing floorplans". In *Proceedings IEEE/ACM Design Automation Conference*, pages 458–463, 2000.

[7] T.H. Chao, Y.C. Hsu, and J.M. Ho. "Zero skew clock net routing". In *Proceedings IEEE/ACM Design Automation Conference*, pages 518–523, 1992.

[8] Chunhong Chen, Ankur Srivastava, and Majid Sarrafzadeh. "On Gate Level Power Optimization Using Dual-Supply Voltages". *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 9(5):616–629, October 2001.

[9] H.-L. Chen and H.-M. Chen. "On Achieving Low-Power SoC Clock Tree Synthesis by Transition Time Planning via Buffer Library Study". In *Proceedings IEEE Systems on Chip Conference*, 2006.

[10] H.-M. Chen, I-M. Liu, and M.D.F. Wong. "I/O Clustering in Design Cost and Performance Optimization for Flip-Chip Design". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(11), November 2006.

[11] H.-M. Chen, I-M. Liu, M.D.F. Wong, M. Shao, and L.-D. Huang. "I/O Clustering in Design Cost and Performance Optimization for Flip-Chip Design". In *Proceedings IEEE International Conference on Computer Design*, pages 562–567, 2004.

[12] O. Coudert. "Gate Sizing for Constrained Delay/Power/Area Optimization". *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 20:465–472, 1997.

[13] P. Coussy, A. Baganne, and E. Martin. "A design methodology for integrating IP into SoC systems". In *Proceedings of the IEEE*, pages 307–310, 2002.

[14] Srinivas Devadas and Sharad Malik. "A Survey of Optimization Techniques Targeting Low Power VLSI Circuits". In *Proceedings IEEE/ACM Design Automation Conference*, pages 242–247, 1995.

[15] M. Donno, A. Ivaldi, L. Benini, and E. Macii. "Clock-tree power optimization based on RTL clock-gating". In *Proceedings IEEE/ACM Design Automation Conference*, pages 622–627, 2003.

[16] M. Donno, E. Macii, and L. Mazzoni. "Power-aware clock tree planning". In *Proceedings International Symposium on Physical Design*, pages 138–147, 2004.

[17] M. Edahiro. "A Clustering-Based Optimization Algorithm in Zero-Skew Routings". In *Proceedings IEEE/ACM Design Automation Conference*, pages 612–616, 1993.

[18] A.H. Farrahi et al. "Activity-driven clock design". In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 705–714, June 2001.

[19] Chunhong Chen et al. "Activity-sensitive clock tree construction for low power". In *IEEE International Symp. on Low Power Electronics and Design*, pages 279–282, 2002.

[20] Ting-Hai Chao et al. "Zero skew clock routing with minimum wirelength". In *IEEE Transactions on Circuits and Systems: Analog and Digital Signal Processing*, pages 799–814, 1992.

[21] P.-N. Guo, C.-K. Cheng, and T. Yoshimura. "An O-tree representation of non-slicing floorplan and its applications". In *Proceedings IEEE/ACM Design Automation Conference*, pages 268–273, 1999.

[22] K. Joe Hass and David F. Cox. "Level Shifting Interfaces for Low Voltage Logic". In *Proceedings NASA Symposium on VLSI Design*, pages 3.1.1–3.1.7, 2000.

[23] L.-C. Hsu and H.-M. Chen. "On Optimizing Scan Testing Power and Routing Cost in Scan Chain Design". In *Proceedings International Symposium on Quaility Electronic Design*, 2006.

[24] J. Hu, Y. Shin, N. Dhanwada, and R. Marculescu. "Architecting voltage islands in core-based system-on-a-chip designs". In *Proceedings ACM International Symposium on Low Power Electronics and Design*, pages 180–185, 2004.

[25] W.-L. Hung, G. M. Link, Y. Xie, N. Vijaykrishnan, N. Dhanwada, and J. Conner. "Temperature-Aware Voltage Islands Architecting in System-on-Chip Design". In *Proceedings IEEE International Conference on Computer Design*, pages 107–112, 2005.

[26] W. Hwang. "New trends in low power SoC design technologies". In *IEEE International SOC Conference*, page 422, 2003.

[27] M.A.B. Jackson, A. Srivasan, and E.S. Kuh. "Clock routing for high-performance ICs". In *Proceedings IEEE/ACM Design Automation Conference*, pages 573–579, 1990.

[28] James T. Kao and Anantha P. Chandrakasan. "Dual-Threshold Voltage Techniques for Low-Power Digital Circuits". *IEEE Journal of Solid-State Circuits*, 35(7):1009–1018, July 2000.

[29] Tanay Karnik, James Tschanz Yibin Ye, Liqiong Wei, Steven Burns, Venkatesh Govindarajulu, Vivek De, and Shekhar Borkar. "Total Power Optimization By Simultaneous Dual-Vt Allocation and Device Sizing in High Performance Microprocessor". In *Proceedings IEEE/ACM Design Automation Conference*, pages 486–491, 2002.

[30] Sarvesh H. Kulkarni, Ashish N. Srivastava, and Dennis Sylvester. "A New Algorithm for Improved VDD Assignment in Low Power Dual VDD Systems". In *Proceedings ACM International Symposium on Low Power Electronics and Design*, pages 200–205, 2004.

[31] David E. Lackey, Paul S. Zuchowski, Thomas R. Bednar, Scott W. Gould Douglas W. Stout, and John M. Cohn. "Managing Power and Performance for System-on-Chip Designs using Voltage Islands". In *Proceedings IEEE/ACM International Conference on Computer-Aided Design*, pages 195–202, 2002.

[32] 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 *Proceedings IEEE/ACM International Conference on Computer-Aided Design*, pages 195–202, 2002.

[33] I-M. Liu, T.-L. Chou, A. Aziz, and D.F. Wong. "Zero-skew clock tree construction by simultaneous routing, wire sizing and buffer insertion". In *Proceedings International Symposium on Physical Design*, pages 33–38, 2000.

[34] M.-C. Lu, M.-C. Wu, H.-M. Chen, and H.-R. Jiang. "Performance Constraints Aware Voltage Island Generation in SoC Floorplan Design". In *Proceedings IEEE Systems on Chip Conference*, 2006.

[35] Ashish D. Mehta, Yao-Ping Chen, Noel Menezes, D.F. Wong, and Lawrence T. Pileggi. "Clustering and load balancing for buffered clock tree synthesis". In *Proceedings IEEE International Conference on Computer Design*, pages 217–223, 1997.

[36] J.D. Meindl. "Low power microelectronics: retrospect and prospect". In *Proceedings of the IEEE*, pages 619–635, 1995.

[37] K. Nose and T. Sakurai. "Analysis and Future Trend of Short Circuit Power". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(9):1023–1030, Sept. 2000.

[38] Min Pan, Chris Chong-Nuen Chu, and J. Morris Chang. "Transition time bounded low-power clock tree construction". In *Proceedings Internationl Symposium on Circuits and Systems*, 2005.

[39] J. Pangjun and S.S. Sapatnekar. "Low-Power Clock Distribution Using Multiple Voltages and Reduced Swings". *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 10(3):309–318, June 2002.

[40] M. Pedram. "Power minimization in IC design: principles and applications". In *ACM Transactions on Design Automation of Electronic Systems*, volume 1, pages 3–56, Jan 1996.

[41] S. Pullela, N. Menezes, and L.T. Pillage. "Low power IC clock tree design". In *Custom Integrated Circuits Concerence*, pages 263–266, 1995.

[42] S.-Y. Tan and H.-M. Chen. "Improved Clustered Voltage Scaling Technique via Better Power-Timing Slack Sensitivity Strategy". In *Workshop on Synthesis And System Integration of Mixed Information technologies*, 2006.

[43] X. Tang and D.F. Wong. "Floorplanning with alignment and performance constraints". In *Proceedings IEEE/ACM Design Automation Conference*, pages 848–853, 2002.

[44] G.E. Tellez and M. Sarrafzadeh. "Minimal buffer insertion in clock trees with skew and slew rate constraints". In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, volume 16, pages 333–342, April 1997.

[45] R.-S. Tsay. "Exact zero skew". In *Proceedings IEEE/ACM International Conference on Computer-Aided Design*, pages 336–339, 1991.

[46] Kimiyoshi Usami and Mark Horowitz. "Clustered Voltage Scaling Technique for Low-Power Design". In *Proceedings ACM International Symposium on Low Power Design*, pages 3–8, 1995.

[47] Kimiyoshi Usami, Mutsunori Igarashi, Takashi Ishikawa, Masahiro Kanazawa, Masafumi Takahashi, Mototsugu Hamada, Hideho Arakida, Toshihiro Terazawa, and Tadahiro Kuroda. "Design Methodology of Ultra Low-power MPEG4 Codec Core Exploiting Voltage Scaling Techniques". In *Proceedings IEEE/ACM Design Automation Conference*, pages 483–488, 1998.

[48] Kimiyoshi Usami, Mutsunori Igarashi, Fumihiro Minami, Takashi Ishikawa, Masahiro Kanazawa, Makoto Ichida, and Kazutaka Nogami. "Automated Low-Power Technique Exploiting Multiple Supply Voltages Applied to a Media Processor". *IEEE Journal of Solid-State Circuits*, 33(3):463–472, March 1998.

[49] H.J.M. Veendrick. "Short-Circuit Dissipation of Static CMOS Circuitry and Its Impact on the Design of Buffer Circuits". *IEEE Journal of Solid-State Circuits*, SC-19(4):468–473, August 1984.

[50] M. Vittal and M. Marek-Sadowska. "Low-power buffered clock tree design". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 16(9):965–975, Sept. 1997.

[51] A. Vorg, M. Radetzki, and W. Rosenstiel. "Measurement of IP qualification costs and benefits". In *Proceedings Design, Automation and Test in Europe*, pages 996–1001, 2004.

[52] K. Wang and M. Marek-Sadowska. "Buffer sizing for clock power minimization subject to general skew constraints". In *Proceedings IEEE/ACM Design Automation Conference*, pages 159–164, 2004.

[53] K.-C. Wang and H.-M. Chen. "Improved Neighborhood Exchange in Multilevel Large-Scale Modules Floorplanning/Placement". In *Workshop on Synthesis And System Integration of Mixed Information technologies*, 2006.

[54] K.-C. Wang and H.-M. Chen. "Multilevel Large-Scale Modules Placement with Refined Neighborhood Exchange". In *Proceedings IEEE International Symposium on VLSI Design, Automation, and Test*, 2006.

[55] H. Wu, I-M. Liu, M.D.F. Wong, and Y. Wang. "Post-Placement Voltage Island Generation under Performance Requirement". In *Proceedings IEEE/ACM International Conference on Computer-Aided Design*, pages 309–316, 2005.

[56] M.-C. Wu and Y.-W. Chang. "Placement with alignment and performance constraints using the B*-tree representation". In *Proceedings IEEE International Conference on Computer Design*, pages 568–571, 2004.