Reliability-Aware SOC Voltage Islands Partition and Floorplan

†Shengqi Yang, †Wayne Wolf, ‡N. Vijaykrishnan and ‡Yuan Xie
†Department of Electrical Engineering, Princeton University, Princeton, NJ, 08544
‡Microsystems Design Lab, The Penn State University, University Park, PA, 16802
{shengqiy|wolf}@princeton.edu {vijay|yuanxie}@cse.psu.edu

***Abstract*— Based on the proposed reliability characterization model, reliability-bounded low-power design as a methodology to balance reliability enhancement and power reduction in chip design, for the first time, is illustrated. Voltage island partitioning and floorplanning for System-On-a-Chip (SOC) design is used as a case study for this reliability aware methodology. The proposed methodology partitions all SOC components into different voltage islands with power reduction and guaranteed system reliability. Experiments show that for a typical SOC the algorithm execution time is within several minutes while achieving 12% to 23% power reduction. Extended SOC algorithm partitions and floorplanns the voltage islands within 2.5 to 29.7 minutes and achieved 9.74% to 18.50% power reduction.**

I. Introduction

Continuous decrease in transistor feature size enables more and more devices to be fabricated in a single chip and various functional modules to be realized as a Systems-On-a-Chip (SOC). This scaling trend poses two critical issues for chip design, i.e., power consumption and system reliability.

Due to the big increase in power density and the wide use of portable systems, power consumption, including dynamic power and leakage power, is becoming a critical design metric. Voltage islands architecting [1]–[3] has emerged as a new technique for core-based low-power SOC design and can effectively cut down all kinds of power sources. A voltage island is a group of on-chip cores powered by the same voltage source, independently from the chip-level voltage supply. The use of voltage islands permits operating different portions of the SOC at different voltage levels in order to optimize the overall chip power consumption at core-level.

Accompanying the feature size scaling down, chip reliability, here it means immunity to soft errors, is becoming another important issue. Soft errors or transient errors are circuit errors caused due to excess charge carriers induced primarily by external radiations [4], [5]. Radiation directly or indirectly induces a localized ionization which upsets internal data states [6]–[8]. Soft errors are particularly troublesome for pipeline latches, and memory-based elements, such as caches, register files, branch target buffer, etc., as the stored values of the bits may be changed. Recent experiments [9] showed that soft error rate of combinational circuits is comparable to or more than those of SRAMs with similar sizes. Soft errors can have much serious impact and lead not only to corrupt data, but also to a loss of functionality and system critical failures. As the technology scales down, both the supply voltage and the node capacitance scale down accordingly. The above two factors make a single device node more sensitive to soft error.

Acknowledgments: This work was supported by National Science Foundation under Grant numbers CCR-0324869 and CCR-0329810.

Supply voltage is lowered, as a most effective way to reduce system power consumption, at some specific location (voltage island) of a SOC. However, lowered supply voltage makes the device nodes more sensitive to soft errors because less charge is required to flip a bit at the node. Systems become less reliable. Design methodology, which can not only reduce the power consumption but insure the Mean Time To Failure (MTTF) constrain without violation of the application deadline time, are desired to help tradeoff the requirements from both reliability and power reduction in chip designs. In the following section, for the first time we will illustrate this reliability-bounded low-power design methodology by using SOC voltage island partitioning as a case study and implementing the corresponding algorithm.

The remainder of this paper is organized as follows. Section II describes the problem. Section III details the algorithm implementation. Section IV discusses the experimental results and Section V concludes this paper.

II. Problem Definition

In this section, reliability enhancement and power reduction tradeoff problem is defined for the SOC voltage island partitioning.

Figure 1 shows an example of core-based SOC design. There are six cores for this SOC. Each core is associated with a list of voltages, that can be used to operate the core, and a list of reliability levels corresponding to each voltage. We will discuss how to get the reliability levels in the following section. For instance, the core C_3 can operate at v_2, v_3, and v_4. If it operates at v_3, the reliability level under this voltage is r_{32}. The chip-level voltage is assumed to be v_1 which is the highest voltage compared with v_2, v_3 and v_4. For core C_6, a distinct voltage island is not created for it because it only can operate at the chip-level supply voltage. In order to minimize the power consumption, one obvious way is to operate each core at its lowest voltage. This means that at least two voltage islands should be created: one for C_1 and C_2 which operate at v_2; one for C_3, C_4 and C_5 which operate at v_4. For the above answer, the power reduction is maximized. However, system reliability is totally ignored in the design procedure. A bad scenario is: C_3, for example an SRAM-based component in civilian aviation avionics system, is very sensitive to soft errors, that occasional bit flips will cause a failure of the application execution, i.e., a flight tragedy. In this case, reliability is so important for C_3 that we cannot neglect it just in order to reduce power. Instead a little power

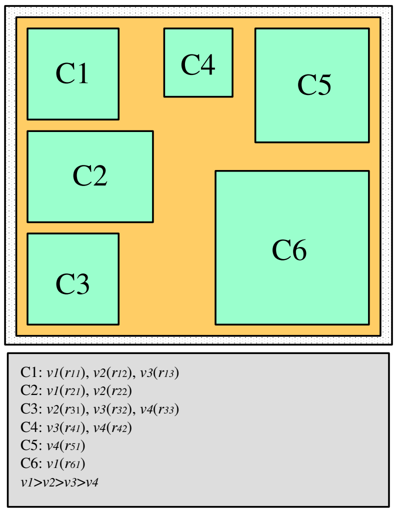

Fig. 1. A Core-based SOC.

reduction will be sacrificed to guarantee the core's reliability.

Now, Let us generalize the reliability and power consumption tradeoff problem for the SOC voltage islands partitioning. We assume given an SOC consisting of a set of cores C_i. Each core is associated with a table, named as PR-table, which specifies the legal voltage levels Vi, the corresponding average power consumption values Pi, and the corresponding average reliability levels Ri. The legal voltage levels of a core can be characterized by the core designers. For example, the designers may keep changing the core supply voltage, as long as the core-level timing assertions are satisfied. And this gives out a list or a range of voltages that can be used to operate the core. Once the legal voltages are selected, the average power consumption and the average reliability level corresponding to each voltage can also be characterized either through experimentation, simulation or analytical estimation. For a voltage island Λ_i, it consists of a few cores which are operated at the unique voltage, denoted as $\Lambda_i(v)$. $\Lambda_i(v)$ is selected from a set, denoted as ΛV_i, containing a list of supply voltage levels. ΛV_i is equal to the intersection of the legal voltage levels of all cores in this island. As an example, if we create an island for C_1 and C_2 in Figure 1, $\Lambda V_i = \{v_1, v_2\}$, while $\Lambda i(v)$ is equal to one of them, either v_1 or v_2, depending on the final tradeoff of power consumption and system reliability. For the whole SOC system, it can be represented as $SOC = \sum_i \Lambda_i + \sum_i C_i$, where $\sum_i C_i$ only denotes those cores not assigned to any voltage island and therefore operated at chip-level power supply voltage.

We use the PR-tables as the algorithm inputs and assume some cores in the SOC, which have a same voltage level in their V_is, can possibly be grouped together to form a voltage island. The constraints or rules for this grouping are power consumption, system reliability and application deadline time. For the convenience of discussion, all other constraints, like chip area, will be considered in the extended algorithm. The problem of power and reliability tradeoff island planning for core-based SOC design can be summarized as partitioning SOC cores into a set of voltage islands and under

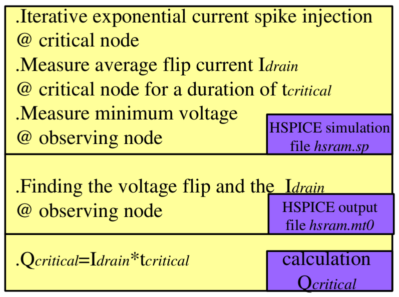
.Iterative exponential current spike injection
@ critical node
.Measure average flip current Idrain
@ critical node for a duration of tcritical
.Measure minimum voltage
@ observing node
HSPICE simulation file hsram.sp
.Finding the voltage flip and the Idrain
@ observing node
HSPICE output file hsram.mt0
.Qcritical=Idrain*tcritical
calculation Qcritical

Fig. 2. $Q_{critical}$ calculation procedure.

this partition reducing the total system power consumption as much as possible without violation of system reliability level and application deadline time.

III. Modeling and Algorithm

In this section, we first illustrate how to characterize component reliability level, and then elaborate the algorithm for reliability-aware SOC voltage island partitioning and floorplanning. Note that the focus of this paper is on how to incorporate the reliability issue into low power design, specifically into the voltage island partitioning and floorplanning.

A. Characterization of Component Reliability Level

For a soft error to occur at a specific node in a SRAM-based element, the collected charge Q at that particular node should be more than a critical charge $Q_{critical}$. If this happens, a pulse is generated and latched on by the feedback mechanism of the inverter, which results in a bit flip at that node. Similar phenomenon happens in combinational logic. The rate at which soft errors occur is given as Soft Error Rate (SER). In order to estimate the Component Reliability Level (CRL), we first select the most sensitive node to soft error in either a SRAM-based component or a logic component, and calculate the node $Q_{critical}$. In general, the sensitivity of a circuit to soft error depends not only on the most sensitive node, but also on the average sensitivity of all circuit nodes. For simplicity,we only consider the most sensitive node. In order to measure $Q_{critical}$ for a particular node, we define it as:

$$Q_{critical} = \int_0^{t_{critical}} I_{drain}(t)dt \tag{1}$$

where $I_{drain}(t)$ is the drain current induced by the charged particle, and $t_{critical}$ is the flipping time. In HSPICE experiment, we model the particle striking current $I_{drain}(t)$ as an exponential current waveform to account for funneling and diffusion charge collection. The current was injected at the most sensitive node and measured up to a point where the circuit commits a bit flip. Finally the current pulse was integrated to get the critical charge of that node. The $Q_{critical}$ calculation procedure for an SRAM is shown in Figure 2. By using this procedure, the $Q_{critical}$ value for a 6-T standard SRAM (65nm Berkeley Predictive Technology (BPT)) is $15.1fC$ and $51.2fC$ for a ripple-carry adder, for example.

Then, $Q_{critical}$ is used to estimate the SER which is expressed in Equation (2) where N_{flux} is the intensity of the Neutron Flux, CS is the area of the cross section of the node and Q_s is the charge collection efficiency. We reasonably assume that N_{flux} and CS are same for different components

TABLE I
CRL FOR ADDER, MULTIPLIER, AND SRAM.

Component	CRL	Area (Unit)
Adder	1.0000	1
SRAM	0.9877	2
Mult	0.9625	2

under the same technology. With this assumption, Q_s can also be set same for different components. We select a component, say C_N, and define its SER to be one, i.e., $SER_N = 1$. For other components, the SERs can be expressed as Equation (3), where $Q_{criticalN}$ and $Q_{criticali}$ mean the critical charges for C_N and C_i.

$$SER \propto N_{flux} \times CS \times exp(\frac{-Q_{critical}}{Q_s}) \quad (2)$$

$$SER_i = exp(\frac{Q_{criticalN} - Q_{criticali}}{Q_s}) \quad (3)$$

The next step is to relate the SER of each component to its reliability level (CRL). Mathematically, component reliability is defined as the probability with which this component can perform the intended function successfully for a period of time, namely [t1, t2], given that it worked properly at time t1. To calculate the reliability of a component, we should first determine its failure rate which is the probability with which it will fail in the next time unit. Although SER does not directly mean failure rate because sometimes soft error will not result an application failure, it can be modified and used to calculate the CRL as Equation (4), where α is a parameter within [0, 1] representing the probability a soft error finally results a failure. As discussed in [10], some masking, such as logical masking, temporal masking and electrical masking, can prevent soft error from resulting an actual failure. α reflects this fact and calculation of α is complicated. For simplicity, here we assume α is equal to one.

$$CRL_i = exp(-\alpha \cdot SER_i \cdot t) \quad (4)$$

In order to build the components library with reliability characterizations, Cadence Virtuoso was used to construct the components layout and HSPICE was for getting the $Q_{critical}$s. Table I shows the normalized CRL for adder, SRAM and carry-save multipler as examples under 65nm BPT. The CRL and area of the adder is set to one. For larger components which contain a lot of basic units, each basic unit like an adder in the netlist can be characterized individually. After this, by analyzing the interconnection of gates in the netlist, the overall soft error susceptibility of the whole component can be determined. Specifically, we use the following formula (5) to describe CRL:

$$CRL_i = \prod_{j=1}^{n} R_j \quad (5)$$

where n means there are n basic units inside the components. This equation reflects, to have a successful execution of an entire component, all the basic units must succeed.

Finally, for a SOC, the system Reliability Level ($RL(SOC)$) can be defined as Equation (6), which is the product of the reliability values for different voltage islands. While the reliability value for a voltage island $RL(\Lambda_i)$ is defined in Equation (7), where component C_{ij} belongs to Λ_i. For CAR, it reflects the application program properties. If a component or some part of the component is never accessed by the application program, soft errors inside this component or inside the specific part of that component cannot result in a program failure; while if a component is accessed very frequently, not only do soft errors which already exist in the component have much possibility to result in a program failure, but also more soft errors can happen in this component during the access time. We use Equation (8) to calculate the Component Access Rate (CAR). Of course, there are some other factors, for example whether a component is protected or not, which affect the final reliability level. Here the system reliability level calculation is a simplified case.

$$RL(SOC) = \prod_{i=1}^{m} RL(\Lambda_i) \quad (6)$$

$$RL(\Lambda_i) = \prod_{j=1}^{n} CAR_{ij} * CRL_{ij} \quad (7)$$

$$CAR_i = \frac{i_{th}\ component\ access\ time}{application\ execution\ time} \quad (8)$$

B. Reliability-aware SOC Voltage Island Partitioning Algorithm

For core-based SOC voltage island partitioning, let us first consider a simplified reliability-bounded low-power design algorithm which generates solutions meeting the requirements from system reliability and application deadline time with reasonable power reduction. There are several other important factors, for example area overhead, by adding which in the following extended algorithm the final optimal SOC voltage island partition and floorplan can be completed. Here, those additional factors are not considered to make the simplified algorithm clearly illustrate the idea of reliability-bounded low-power design. The algorithm is show in Figure 3.

In the algorithm for SOC island partitioning, the inputs are the lower bound of system reliability level $RL(SOC)^*$, the application execution time DT in the SOC system with each core operated at its highest voltage, the upper limit of application deadline time DT^*, the net information, and the characteristic parameters for each SOC component including the applicable voltage levels V_i, the power consumption value P_i, the component reliability level R_i, and CAR_i. Among the component parameters, V_i is a vector; while P_i, R_i, and CAR_i are scalars achieved with respect to the highest component supply voltage. The list of voltages, V_i, for a specific component can be characterized by the designer. Once a prototype of the SOC system is constructed, the targeted application programs can be tested in this prototype at some supply voltage. Then P_i and CAR_i for each component can be estimated and calculated on average. For example, if the component C_i is tested under the supply voltage v_{test} and the average power consumption is $P_i(v_{test})$, the $P_i(v)$ can be calculated according to Equation (9), where v and v_{test} belong to the set V_i. If C_1 works at v_1 as illustrated in Figure 1, C_2 at v_1, C_3 at v_2, C_4 at v_3, C_5 at v_4 and C_6 at v_1, the average access time of each component, AT_i, can be achieved by

```
Input   :  Ci(Vi, Ri, Pi, CARi), RL(SOC)*, DT*
Output  :  All possible solutions which meet the criterion

Algorithm :

1:  For each SOC component Ci Do
2:        calculate CRLi and CPi
3:  End For
4:  construct a link list (VILL) for possible voltage island partitions
5:  For each partition in VILL Do
6:        update the CARi for each component
7:        calculate the deadline time DT for this partition
8:        If DT > DT* Then
9:              delete this partition from VILL
10:       Else
11:             calculate the reliability level RL(SOC)
12:             If RL(SOC) < RL(SOC)* Then
13:                   delete this partion from VILL
14:             Else
15:                   calculate the power reduction
16:             End If
17:       End If
18:       output the head of VILL
19: End For
20: sort VILL by the value of power reduction
21: output all the possible solutions
```

Fig. 3. Reliability and power consumption tradeoff algorithm for core-based SOC voltage island partitioning

running the application program in the SOC under the above specified voltage level for each component. The equation for calculation of CAR_i can be extended from Equation (8) to Equation (10). Later in the final implementation of the SOC system, if core C_1 operates at v_2 instead and the supply voltages for other cores do not change, the CAR_1 can be calculated as Equation (11) which scales the access time according to the frequency change caused by the voltage change. Here v_{th} means the threshold voltage. As a result, the voltage level at which CAR_i is achieved must be specified and input to the algorithm. For R_i, it is the reliability level for the core operated at the highest voltage.

$$P_i(v) = \frac{v^2}{v_{test}^2} \times P_i(v_{test}) \tag{9}$$

$$CAR_i = \frac{AT_i}{\sum_{j=1}^{6} AT_j} \tag{10}$$

$$CAR_1 = \frac{AT_1(\frac{v_1 - v_{th}}{v_2 - v_{th}})^{1.3} \times \frac{v_2}{v_1}}{AT_1(\frac{v_1 - v_{th}}{v_2 - v_{th}})^{1.3} \times \frac{v_2}{v_1} + \sum_{j=2}^{6} AT_j} \tag{11}$$

$$CRL_i(v) = \frac{v}{v_{highest}} \times R_i \tag{12}$$

First, the algorithm scales the R_i with respect to each voltage in V_i and keeps this number in a vector, CRL_i. The calculation is shown in Equation (12), where v and $v_{highest}$ are in the set V_i with the latter to be the highest supply voltage for core C_i. Here a linear dependence between reliability and operation voltage is assumed according to the idea of $Q_{critical}$. More complex analytical models can be applied for CRL_i calculation without significant change in the algorithm complexity. Similarly, the algorithm calculates the component power consumption CP_i at each voltage level according to Equation (9), by substituting the CP_i for P_i. Both CRL_i and CP_i are vectors. Second, the algorithm constructs a link list $VILL$ for all possible voltage island partitions. For each possible partitioning, the CAR_i should be refreshed for each component according to Equation (11). In the following steps, the application deadline time is first checked to see whether it exceeds the upper limit DT^* under this partitioning. If it does, then delete this partition from the link list. Otherwise the second checking is made for the system reliability level according to Equations (6). If the system reliability level under this configuration is less than $RL(SOC)^*$, then delete this partition from the list. The partitions remaining in the link list after both checks must meet the requirements from application deadline time and system reliability. Finally, the algorithm calculates the power reduction and the number of voltage islands for each possible voltage island partition.

The above algorithm is only for partitioning the SOC voltage islands by using three metrics, system reliability level, application deadline time, and power reduction. After this procedure which clearly illustrated the idea of reliability-bounded low-power design, all the remaining partitions guarantee the SOC system to be fast enough, reliable and power-saving. However, there are a few other significant factors which should be considered in order to find the final optimal solution for the SOC system. Here we consider application deadline time overhead, further reliability degradation, power consumption overhead and area overhead. The first three factors are caused by off-island interconnections between components in different voltage islands. While the fourth factor is caused by dead space inside a voltage island. In order to consider these factors, a floorplan algorithm must be applied subsequently in order to get the desired interconnection information.

Once the SOC system and its application are determined, the critical path and the corresponding components within this path can be identified. If floorplanning algorithm is applied after the partitioning, the locations of all components are set. In the case that the components in the critical path, for example C_1 and C_2, are partitioned into two different voltage islands, a lot of off-island communications between C_1 and C_2 are created. For this case, the application execution time increases compared with the case with two components in the same island. In order to reflect this performance overhead, a penalty is added to the deadline time. This penalty is proportional to the wire length between voltage islands in the critical path. As a result, the application deadline time under a specific partition is expressed as Equation (13), where δ is the penalty factor. If all the components in the critical path are located in the same island, δ is equal to zero. Otherwise, a positive fractional number which is less than one is added with respect to the number of islands which contain the critical components.

Furthermore, to make the communications between two voltage islands feasible, voltage converters are used between them. Soft errors and power consumption can also happen inside these converters. Correspondingly after the floorplanner applied (without floorplanner, the desired information is not available), the system reliability and system power consumption can be updated and expressed in Equations (14) and (15). Samely, ϵ and η are penalty factors for system reliability and power consumption, respectively. Both of them are positive fractional numbers and proportional to the islands number. More voltage islands mean more power consumption in the converters and less reliable system. For the area overhead,

```
Input    :  component information, net information
            all kinds of constraints
Output   :  optimal voltage island partition
Algorithm :

1:  construct a link list for all possible voltage island partitions VILL
2:  For each node in VILL Do
3:    floorplan all voltage islands
4:    calculate deadline time
5:    If DT <= DT* Then
6:      calculate system reliability level
7:      If RL(SOC) >= RL(SOC)* Then
8:        calculate area overhead
9:        If (area overhead meets the requirement) Then
10:         calculate the number of VIs
11:         If (number of VIs meets the requirement) Then
12:           calculate the power reduction
13:         Else
14:           delete this node
15:       Else
16:         delete this node
17:     Else
18:       delete this node
19:   Else
20:     delete this node
21: End For
22: find the node in VILL with maximal power reduction
23: output this node
```

Fig. 4. Extension of power consumption and reliability tradeoff algorithm for core-based SOC voltage island partitioning

a voltage island bringing C_1 and C_6 together would have a lot of dead space inside this island. This kind of area overhead should be minimized if it cannot be avoided. The information for calculating the above factors can be obtained after the floorplanning of the partitioned voltage islands. A simple floorplanner based on sequence pair representation and evaluation [2] is realized in the extended algorithm. By adding these factors, the algorithm is augmented and shown in Figure 4.

$$DT = (1+\delta) \times DT \tag{13}$$

$$RL(SOC) = (1-\epsilon) \times RL(SOC) \tag{14}$$

$$P(SOC) = (1+\eta) \times P(SOC) \tag{15}$$

In the extended algorithm, all possible voltage island partitions are constructed and saved in a link list. For each possible partition, the algorithm first floorplans all partitioned voltage islands bounded by the specified constraints. For example, a voltage island may not be allowed for a few components with one located on the periphery of the SOC chip because this may violate the constraint of proximity to the power pins. Furthermore, a clock generator/distributor is better to be put in the center of the chip in order to evenly distribute the clock signal to all components. After the floorplanning, the algorithm checks the deadline time and system reliability level according to Equations (13) and (14). Further, area overhead (dead space) and number of voltage island for this partition are also calculated. If the limit of either of the above four factors is not met, this partition node is deleted. Finally, the power consumption for the nodes meeting all the four requirements is calculated according to Equation (15). And the node or partition with the maximal power reduction is output.

IV. Experimental Results

The reliability-bounded low-power design algorithms are realized in C programs. The platform for the experiments is an Intel Celeron 2.0Ghz CPU with Red Hat Linux 9.0 as the operating system. To the best of our knowledge, this is the first work on reliability-bounded low-power design. We used two MCNC benchmarks [11] and constructed a few synthetic benchmarks $s10$, $s15$, $s30$ and $s50$ as summarized in Table II. For voltage setting, there are totally seven voltage levels available with range from $0.99V$ to $1.49V$. The second column shows the number of cores in each benchmark. Each core in the SOC system has a list of applicable voltage levels. The third column gives the average number of voltage levels for each core in the benchmark. For these benchmarks, we generated the power consumption values assuming that they are proportional to the corresponding core areas, and then scaled them appropriately according to Equation (9). The initial reliability level for each core is generated according to Equation (5). Assignments of the lower bound of system reliability level $RL(SOC)^*$ and the upper limit of system deadline time DT^* affect the number of final optimal solutions, but have little effect on the algorithm execution time.

In Table II, the fourth and the fifth columns show the results for a particular optimal partition in a benchmark with the maximal power reduction and the corresponding number of voltage islands under this partition. The sixth and the seventh columns show the results for another specific optimal partition with the minimal power reduction and the corresponding number of voltage islands. This is the extreme case with guaranteeing system reliability level with highest level and sacrificing power reduction. All the power reduction numbers are achieved by comparing the power under this specific partition and the power consumed by the SOC with all cores operated at their individual highest voltage levels. The eighth column shows the algorithm execution time. And the final column illustrates the reliability bound used to generate the optimal solutions. 95% means that the system reliability under one optimal partition should not be lower than 95 percent of the system reliability under the specific partition with all chip components operated at their highest voltage.

The results in Table II show that our algorithm is fast. It typically finishes in minutes with the actual running time depending on the size of a design and the constraints imposed on it, including number of cores and number of voltage levels for each core. Since in this work, the focus is on how to introduce the reliability issue in the partition and floorplan, the algorithm is not optimized by using advanced techniques, instead here we used heuristic algorithm. Further improvement in algorithm execution time is expected in the follow-up work on algorithm optimization.

Bounded by the system reliability and performance (application deadline time), the extended SOC algorithm searches for a final optimal solution whose area overhead and number of voltage island are within the tolerance range with maximal power reduction. Table III shows the results. Obviously, the extended algorithm is still very fast. Columns fourth and fifth compare the power consumption with reliability consideration and without consideration. Notice that the time unit is minute.

V. Conclusions

For the first time, we have developed a reliability characterization methodology and implemented a algorithm to illustrate the idea of reliability-bounded low-power design by using voltage island partitioning in SOC as a case study.

TABLE II
RESULTS FOR THE RELIABILITY-BOUNDED LOW-POWER SOC VOLTAGE ISLAND PARTITION ALGORITHM.

Benchmark	Number of cores	Average number of voltage levels	Max power reduction	Number of VIs	Min power reduction	Number of VIs	Execution time(sec)	Reliability bound
s10	10	2.0	13.87%	3	5.84%	2	1	95%
s15	15	3.7	12.01%	3	5.21%	3	58	95%
s30	30	2.3	23.10%	4	6.40%	3	431	95%
s50	50	1.9	15.68%	6	2.84%	4	480	95%
ami33	33	2.2	12.91%	4	7.8%	4	456	95%
ami49	49	1.7	15.20%	5	5.70%	4	479	95%

TABLE III
RESULTS FOR THE SOC EXTENDED ALGORITHM.

Benchmark	Number of cores	Execution time (min)	Power reduction w reliability	Power reduction wo Reliability	Number of VIs	Reliability bound
s10	10	2.5	10.77%	16.69%	3	95%
s30	30	19.8	18.50%	20.85%	4	95%
s50	50	29.7	9.74%	13.10%	5	95%
ami33	33	22.7	12.93%	16.37%	4	95%
ami49	49	25.8	8.98%	9.85%	4	95%

Three metrics, i.e. system reliability level, power reduction and application deadline time, were used to determine the quality of candidate voltage island partitions. Experiments show that for a SOC the algorithm execution time is within a few minutes while achieving 12% to 23% power reduction. For the extended SOC algorithm, it partitioned and floorplanned the voltage islands within 2.5 to 29.7 minutes and achieved 9.74% to 18.50% power reduction.

REFERENCES

[1] D. E. Lackey, *et al.*, "Managing Power and Performance for System-on-chip Designs Using Voltage Islands," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 2002, pp. 195 – 202.

[2] J. Hu, Y. Shin, N. Dhanwada, and R. Marculescu, "Architecting Voltage Islands in Core-based System-on-a-chip Designs," in *Proc. Int. Symp. Low Power Electronics and Design*, Aug. 2004, pp. 180–185.

[3] J. N. Kozhaya and L. A. Bakir, "An Electrically Robust Method for Placing Power Gating Switches in Voltage Islands," in *Proc. Conf. Custom Integrated Circuits*, Oct. 2004, pp. 321 – 324.

[4] N. Seifert, D. Moyer, N. Leland, and R. Hokinson, "Historical Trend in Alpha-particle Induced Soft Error Rates of the AlphaTM Microprocessor," in *Proc. IEEE Int. Symp. Reliability Physics*, May 2001, pp. 259–265.

[5] N. Seifert, *et al.*, "Frequency Dependence of Soft Error Rates for Sub-micron CMOS Technologies," in *International Electron Devices Meeting, Technical Digest*, Dec. 2001, pp. 14.4.1 –14.4.4.

[6] V. Degalahal, N. Vijaykrishnan, and M. J. Irwin, "Analyzing Soft Errors in Leakage Optimized SRAM Design," in *Proc. Int. Conf. VLSI Design*, Jan. 2003, pp. 227–233.

[7] V. Degalahal, R. Rajaram, N. Vijaykrishnan, Y. Xie, and M. J. Irwin, "The Effect of Threshold Voltages on Soft Error Rate," in *Proc. Int. Conf. Quality Electronic Design*, Mar. 2003.

[8] R. Ramanarayaman, V. Degalahal, N. Vijaykrishnan, M. J.Irwin, and D. Duarte, "Analysis of Soft Error Rate in Flip-flops and Scannable Latches," in *Proc. IEEE Int. Conf. SOC*, Sept. 2003, pp. 231–234.

[9] M. Zhang and N. R. Shanbhag, "A Soft Error Rate Analysis (SERA) Methodology," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 2004, pp. 111–118.

[10] C. Weaver, J. Emer, S. S. Mukherjee, and S. K. Reinhardt, "Techniques to Reduce the Soft Error Rate of a High-Performance Microprocessor," in *Proc. Int. Symp. Computer Architecture*, June 2004, pp. 264–275.

[11] "Mcnc floorplan benchmarks." [Online]. Available: http://www.cse.ucsc.edu/research/surf/GSRC/MCNCbench.html