

# Simultaneous PVT-Tolerant Voltage-Island Formation and Core Placement for Thousand-Core Platforms

Sohaib Majzoub, Resve Saleh, Steven J.E. Wilton, and Rabab Ward  
Department of Electrical and Computer Engineering  
University of British Columbia  
Vancouver, BC, Canada

**Abstract**— In this paper, we propose a novel approach to voltage island formation and core placement for energy optimization in manycore architectures under parameter variations at pre-fabrication stage. We group the cores into irregular "cloud-shaped" voltage islands. The islands are created by balancing the desire to limit the spatial extent of each island, to reduce PVT impact, with the communication patterns between islands. Compared to using rectangular islands, our approach leads to power improvements between 10 and 12%.

## I. INTRODUCTION

Manycore platforms are gaining momentum within the processor design spectrum. Chips with thousands of cores promise high computing power when multiple applications are running simultaneously, each with multiple tasks and multiple threads. There are a number of design challenges for these architectures including high power consumption, memory and cache coherence, network architectures, and the impact of Process Voltage and Temperature (PVT) variation [1]. In this paper, we are tackling the problem of power reduction under PVT impact.

One common way of reducing power in integrated circuits is to divide the chip into voltage islands. A voltage island is a region of a chip that is powered by a single voltage level. Regions of the chip that are timing critical can be mapped to voltage islands powered by relatively high voltages, while regions of the chip that are not timing critical can be mapped to voltage islands powered by relatively low voltages. The regions powered by high voltages consume more power but run faster; the regions powered by low voltages consume less dynamic and static power, but run slower [2][3][4][5][6]. In a manycore chip, each voltage island will typically contain many adjacent cores.

Typically, these voltage islands are rectangular [2][4][5]. In this paper, we show that this is a particularly bad choice in the presence of Process, Voltage, and Temperature variations for manycore architectures.

The 2005 International Technology Roadmap for Semiconductors (ITRS) projected that these variations will be the major obstacle to high chip yields. On-chip variations include process variation due to manufacturing deficiencies at nano-scale technologies, and voltage and temperature variations due to activity and workload distribution. Process variation can be die-to-die (D2D) or within-die (WID). We focus on WID variation in this work [2][5][7].

Within-die process variation will cause some cores to run slower than others. We can mitigate this effect by increasing the supply voltage of the slower cores. However, if a chip is designed using voltage islands, increasing the voltage of a single core requires increasing the voltage (and hence power) of all cores within the island. Thus, the voltage level for each island is dictated by the slowest core in that island [2][4][5].

In this paper, we show that, due to the spatial variation inherent in within-die variation (WID), circular, or cloud-shaped islands result in better power characteristics than rectangular cores. We then present a novel algorithm that uses this insight to create voltage islands that are optimized for both PVT variations and the inter-island communication required by the application, and places individual cores within these islands. We evaluate our algorithm in the context of a power-aware mapping algorithm involving voltage island creation, voltage selection, and an optimization flow.

## II. RELATED WORK

In [5], the authors statistically modeled transistor parameters to construct a process variation model. Then, they attempted to reduce power consumption using voltage/delay dependency. Humenay et al. provided a model for systematic variation and then showed process variation and thermal throttling implications on the cores' frequency [8]. Herbert and Marculescu compared the throughput of chip-level multiprocessors (CMP) constructed using frequency islands across a range of core counts and sizes under process variation [9]. They showed that designs with smaller cores are more tolerant to variability than designs with fewer larger cores. Huang et al. showed that an architecture with many simple cores rather than complex cores reduces the thermal power density (TPD)[10]. They proposed an on-chip adaptive regulation to the voltage supply. However, on-chip regulation can lead into significant increase in energy. Wang et al. presented a variation-aware task scheduling on a Multiprocessor System-on-Chip (MPSoC) [11]. They introduced performance yield as a metric to assist the task allocation as a statistical timing task graph. Coskun et al. provided an exploration of thermal-aware task allocation on an MPSoC [12]. They presented an OS-level dynamic task allocation algorithm for different policies with negligible performance overhead. Li et al. described a tool for NoC

architecture space exploration under process and temperature variation [13].

This previous research dealt with simple multicore platforms with less than 100 cores. They did not combine the effects of process, voltage and temperature variation together in their energy optimization techniques for manycore designs. There is no prior study on the relationship between the voltage island shape and PVT impact.

### III. PVT-VARIATION MODELS

Before describing our voltage island technique, we review the PVT models assumed in this work. To capture the WID process variation for circuit parameters, such as  $V_t$  and  $L_{eff}$ , we used the methodology described in [2][7]. The systematic process variation is created using a multivariate normal distribution with a spherical spatial correlation structure. The chip is divided into small rectangular-shaped fragments. Each fragment is given a normal distribution of  $V_t$  and  $L_{eff}$ , the main cause of the systematic variations, with 0 mean and a standard deviation given by  $\sigma_{sys}$ . The correlation factor,  $\rho(r)$ , is a function of the distance  $r$  between fragment X and Y. The derived correlation function:

$$\rho(r) = \begin{cases} 1 - \frac{3r}{2\phi} + \frac{r^3}{2\phi^3}, & r \leq \phi \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

The random variation is represented by  $\sigma_{rand}$ . The random segment is due to uncorrelated random doping effects. Both the random and systematic variations are considered equal. For  $L_{eff}$ , the systematic variation  $\sigma_{total}/\mu$  is assumed to be half of that of  $V_t$ . We assumed the total standard deviation over the mean,  $\sigma_{total}/\mu$ , to be 0.12, which is somewhat pessimistic [2].

In order to compute the temperature variation in the system, the HotSpot tool of [14] is used to calculate the temperature of each core during execution of a segment of code. On the other hand, the power grid model used in [2] was utilized to estimate the voltage variation. The power grid was assumed to be a resistive network, with each processor a current source. Each voltage level was assumed to have its own power grid.

As in [2], the model is extended to a multiprocessor platform by dividing the chip into separate core regions. The process variation within the region has a direct impact on the speed of the core. Based on the variation across a core and its location on the chip, the frequency can be computed for each core using the delay along the critical path. Figure 1(a) shows an example of manycore platform with process variation. Indeed, visual inspection reveals the potential advantage of cloud-shaped islands.

### IV. VOLTAGE ISLAND FORMATION

In this section, we present our voltage island formation algorithm and core placement algorithm.

#### A. Motivation

Intuitively, our algorithm should create voltage islands and position the cores such that:

- the distance among cores in the same island is minimized to reduce PVT impact.

- islands with heavy communication are placed close to each other.

- islands that will be supplied the same voltage level should be placed next to each other to allow these islands to be merged.

The first goal suggests that islands should be circular, or cloud-shaped, rather than very oblong rectangles. Intuitively, these shapes would have a smaller average distance between cores. Due to the spatial correlation inherent in within-die process variations, cores that are close to each other share similar characteristics, but as the distance between the two cores increases, such similarities will start to fade [2][7][8]. In the case of a thousand-core platform, the wide spread of cores makes frequency differences among cores significant [1]. Figure 2 shows this graphically. The rectangular voltage island  $VI_1$  is much more affected by the systematic variations due to its span across the chip, whereas the circular voltage island  $VI_2$  is more PVT-tolerant. A PVT-tolerant solution is more power efficient, since the voltage supplied to each island is dictated by the slowest core in the island; large variations within an island means that, on average, more cores have to be powered at a higher voltage than would otherwise be necessary. The second goal ensures that communication paths are short.

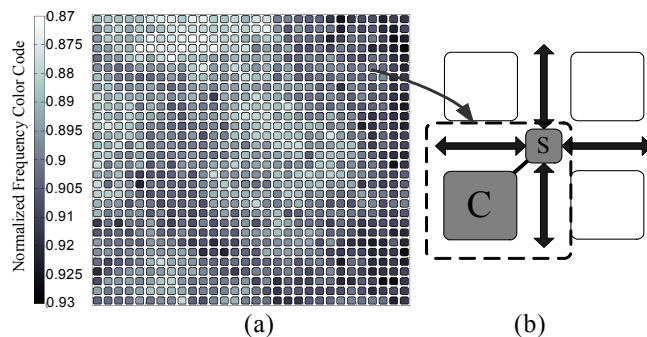


Figure 1. Manycore Architecture (a) core's normalized frequency due to process variation, (b) core, its switch and links

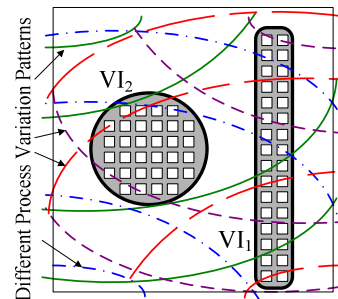


Figure 2. Circular vs. rectangular shapes Voltage Islands

A good solution will balance these goals. Circular voltage islands would minimize the maximum distance between cores within each island. However, it is difficult to closely-pack circular voltage islands together which will have a negative impact on communication between different islands. Our algorithm "grows" islands in such a way as to balance these optimization goals.

### B. Cloud-Shaped Voltage Island Formation Algorithm

We assume that the cores have already been assigned islands and that the islands have been partitioned into  $d$  sets, where each island in a set will be eventually assigned the same voltage level (and hence merged into a single island). The number of islands in set  $j$  will be denoted  $n_j$ .

Cores are assigned locations governed by three seeds. Each seed represents a physical location on the chip. The highest level seed, called the *placement seed*, is used to anchor all sets on the chip. For each set, there is one intermediate-level seed, called a *global seed*. This seed anchors the voltage islands within the set. Finally, there is a *local seed* for each voltage island that is used to anchor individual cores within an island.

#### Voltage Island Cloud Placement

Inputs: Voltage Islands to be placed:

$$\left\{ \{VIC_1, \dots, VIC_{n_1}\}, \{VIC_1, \dots, VIC_{n_2}\}, \dots, \{VIC_1, \dots, VIC_{n_d}\} \right\}$$

Outputs: Placed Voltage Island Clouds      VICs to be merged together

```

begin
pick placement seed PS
for j = 1 to d do
  pick {VIC1..VICnj} with highest inter-island communication with the
  already placed islands
  pick Global Seed GSj, such that distance |PS-GSj| is minimum
  for k = 1 to nj, all VICs to be merged together do
    pick VICk with highest inter-island communication with the
    already placed islands
    pick Local Seed LSk, such that distance |GSj-LSk| is minimum
    /* start placing VICk cores */
    for i = 1 to P do
      /* start placing cores closer to the seed LSk */
      pick core Ci with highest intra-island communication
      place core Ci, such that its distance to LSk is minimum
    end for
  end for
end for
end for

```

Figure 3. Cloud-shaped voltage island placement (pseudo-code)

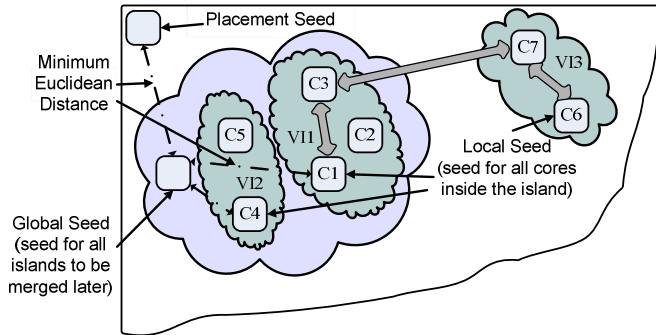


Figure 4. Seeding hierarchy with inter- and intra- island traffic.

Figure 3 shows our algorithm. We first set the placement seed to be one corner of the chip (other strategies are possible). We then cycle through each set, each time considering the set that has the highest inter-island communication with the islands already placed. For this set, we determine the closest unused location to the placement seed, and assign this to the set's global seed. We then cycle through all voltage islands within the set, each time considering the voltage island with the highest inter-island

communication with the islands already placed. For each voltage island, we then determine the closest unused location to the global seed, and assign this to the island's local seed. Finally, we cycle through all cores within the island, and greedily position each core at the closest unused location to the island's local seed.

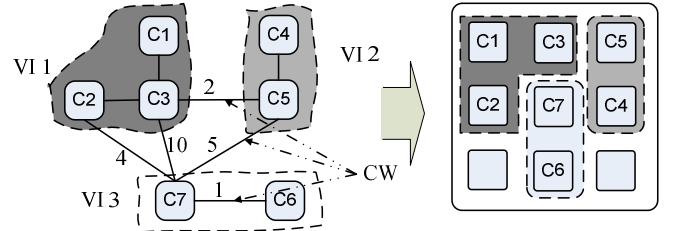


Figure 5. Example showing VIC Island Placement.

An example is shown in Figure 4. VI<sub>1</sub> and VI<sub>3</sub> communicate heavily and therefore they should be placed close to one another. Core C<sub>1</sub> does not communicate with the outside world so it should be closer to the seed of VI<sub>1</sub>. Since VI<sub>1</sub> and VI<sub>2</sub> will be merged, they should be placed adjacent to one another so that they can be assigned the same voltage level later. Figure 5 shows the actual implementation of the example shown in Figure 4. In the example of Figure 5, Islands 1 and 3 have higher communication so they should be placed close to each other. Similarly, core C<sub>3</sub> has high inter-island communication, thus it should be placed close to the border of the island.

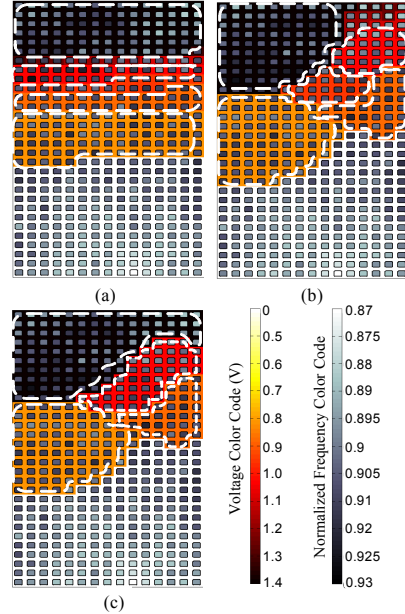


Figure 6. Voltage Islands; (a) rectangular-shaped and (b) cloud-shaped, and (c) after performing voltage selection on (b)

Figure 6(a) and (b) show real example of islands with rectangular and VIC shapes. The islands are circled with a white dotted line. The voltage level of the island is shown in the shading outside the core square, and the frequency value is shown by the shading inside the core square. Figure 6(c) shows how islands assigned the same voltage level can be merged.

## V. EXPERIMENTAL RESULTS

We evaluate our algorithm in the context of a larger power-aware mapping flow. Figure 7 shows the overall mapping algorithm. First, different applications are allocated to the available resources. Tasks are then scheduled and assigned to cores using an As-Soon-As-Possible (ASAP) scheduling algorithm.

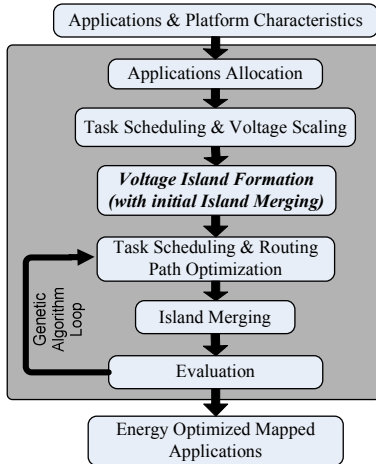


Figure 7. Optimization Flow graph.

The slack of each task is computed, and used to scale the voltage of each core as low as possible such that all tasks can meet their deadlines. Voltage islands are then formed by grouping cores with the same voltage. Voltage islands are merged using the algorithm from [6], and cloud-shaped voltage islands are then formed as described in this paper.

The routing traffic is optimized using a genetic algorithm. Each potential route is represented by a chromosome, and the outer loop of the algorithm optimizes each route. Further details can be found in [2] and [4].

In our experiments, we used 30 test cases. In every test case, a randomly generated task graph, with 950 tasks representing the task graphs were used [15]. The number of cores considered in the simulation varies from 400 to 2025 cores. Up to four voltage levels between 0.6V and 1.4V are available for the optimization algorithm. Any unused cores are switched off completely.

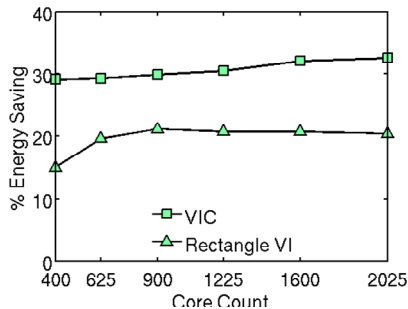


Figure 8. Cloud- vs. Rectangular-shaped Voltage Islands

A 65nm with nominal  $V_t=400\text{mV}$  and maximum  $V_{dd}=1.4\text{V}$  was assumed. The processor speed was 50MHz, and the network speed was 100MHz. The dynamic power was estimated to be 0.4mW/MHz [2]. The optimization and

placement operations were carried out on a representative process variation (PV) profile for calibration purposes; then, the resulting mapping was enforced on 8 other arbitrary profiles [2]. The voltage levels of the islands in the manycore platform were scaled up until the slowest core meets the timing requirements. The energy savings results of all the test cases are grouped and averaged.

Figure 8 compares the energy savings obtained by our optimization algorithm for two cases: cloud-shaped voltage islands and rectangular voltage islands for various numbers of cores. Using the cloud-shaped cores, power is reduced by 10% to 12% beyond that with rectangular cores.

## VI. CONCLUSIONS

In this paper, we proposed a novel approach for voltage island formation in manycore designs under the influence of within-die process, temperature, and voltage variations. Because of the locality of these effects, cloud-shaped voltage islands are much more PVT-tolerant than rectangular-shaped voltage islands. The proposed voltage island shape change will improve the energy savings up to 12%.

## REFERENCES

- [1] S. Borkar, "Thousand core chips: a technology perspective" DAC, 2007.
- [2] S. Majzoub, R. Saleh, and R. Ward, "PVT Variation Impact on Voltage Island Formation in MPSoC Design" ISQED 2009. San Jose, CA, USA March 2009.
- [3] S. Majzoub, R. Saleh, S. Wilton and R. Ward, "Removal-Cost Method: An Efficient Voltage Selection Algorithm for Multi-Core Platforms under PVT" SoCC 2009.
- [4] L.-F. Leung, C.-Y. Tsui, "Energy-Aware Synthesis of Networks-on-Chip Implemented with Voltage Islands." DAC 2007.
- [5] A. Das, S. Ozdemir, G. Memik and A. Choudhary. "Evaluating Voltage Islands in CMPs under Process Variations." ICCD, 2007.
- [6] K. Agarwal, K. Nowka, "Dynamic Power Management by Combination of Dual Static Supply Voltages," ISQED, 2007.
- [7] S. Sarangi, B. Greskamp, R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas. "VARIUS: A Model of Process Variation and Resulting Timing Errors for Microarchitects," IEEE TSM, Feb. 2008.
- [8] E. Humenay, D. Tarjan, and K. Skadron, "Impact of process variations on multicore performance symmetry." DATE 2007.
- [9] S. Herbert, and D. Marculescu "Characterizing Chip-Multiprocessor Variability-Tolerance" DAC 2008.
- [10] W. Huang, M. R. Stan, K. Sankaranarayanan, R. Ribando, and K. Skadron. "Many-Core Design from a Thermal Perspective." DAC, June 2008.
- [11] F. Wang, C. Nicopoulos, X. Wu, Y. Xie, N. Vijaykrishnan, "Variation-aware task allocation and scheduling for MPSoC." ICCAD 2007.
- [12] A. Coskun, T. Rosing, and K. Whisnant, "Temperature aware task scheduling in MPSoCs." DATE 2007.
- [13] B. Li, L.-S. Peh, P. Patra. "Impact of Process and Temperature Variations on Network-on-Chip Design Exploration". IEEE International Symposium on Networks-on-Chip, April, 2008.
- [14] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, M. R. Stan, "HotSpot: A Compact Thermal Modeling Methodology for Early-Stage VLSI Design." IEEE TVLSI 2006.
- [15] T. Tobita, H. Kasahara, "A standard task graph set for fair evaluation of multiprocessor scheduling algorithms." ICS'99 Workshop. (1999).
- [16] A. Zjajo, S. Krishnan, and J. de Gyvez. "Efficient Estimation of Die-Level Process Parameter Variations via the EM-Algorithm." DDECS'08 Bratislava, Slovakia April 16-April 18.