PARM: Power Supply Noise Aware Resource Management for NoC based Multicore Systems in the Dark Silicon Era

Venkata Yaswanth Raparti, Sudeep Pasricha
Department of Electrical and Computer Engineering
Colorado State University, Fort Collins, CO, U.S.A.
yaswanth@rams.colostate.edu, sudeep@colostate.edu

ABSTRACT

Reliability is a major concern in chip multi-processors (CMPs) due to shrinking technology and low operating voltages. Today's processors designed at sub-10nm technology nodes have high device densities and fast switching frequencies that cause fluctuations in supply voltage (V_{dd}) and ground networks, which can adversely affect the execution of applications running on them. In this paper, we propose a novel runtime framework to reduce the power supply noise (PSN) in cores and routers at runtime. Experimental results for 7nm FinFET process nodes show that our framework not only achieves up to 4.5× reduction in PSN, and up to 34.3% improvement in application performance, but also manages to map up to 38% more applications when the CMP is oversubscribed, compared to the state-of-the-art.

Categories and Subject Descriptors: [EDA] System level design methodology; [Reliability] Reliability for Power Supply Noise; [Resource Management] Task mapping to meet given constraints
General Terms – Reliability, Performance, Application mapping
Keywords – Power Supply Noise, Dark Silicon

1. INTRODUCTION

The advent of sub-10nm technology has led to chip multi-processors (CMPs) that are densely packed with transistors. In these CMPs, multiple parallel applications frequently execute at the same time on multiple cores, creating variations in workload activity at runtime. In cases where several cores switch at the same time, noise is introduced in the power supply due to the resistive drop of chip parasitics or the inductive droop between circuits [1]. Studies have shown that power supply noise (PSN) increases critical path latency and degrades chip performance. Moreover, PSN above a certain threshold can also lead to timing errors, called *voltage emergencies* (VE), which if left uncorrected can result in incorrect outputs from application execution.

Recent studies have shown that at any given time, large sections of a chip remain inactive, to not exceed thermal design budgets [2]. This phenomenon is called *dark silicon*. Studies have shown that the percent of the chip that remains inactive (dark) is increasing with technology scaling [3]. Researchers have proposed operating cores at near threshold voltages, dubbed as *near threshold computing* (NTC), to reduce the chip power footprint and minimize the amount of dark silicon [5]. But this results in a shrinking headroom between supply voltage V_{dd}, and threshold voltage V_{th}, with a low margin of error for noise in V_{dd}. Fig. 1 shows peak PSN in the power delivery wires on the chip due to inter-core interference [4], *which is increasing alarmingly with technology scaling by going beyond the permissible noise margin, making the PSN threat a serious concern for chip designers.*

Simultaneously, applications are becoming increasingly parallel to utilize the abundance of compute resources on a chip. This has led to increasing communication between cores on a chip. Modern CMPs are embracing network-on-chips (NoCs) as their de-facto communication fabric to cope with increasing on-chip traffic. But NoCs in today's chips can consume a significant amount of chip power (~ 30% in the 80-core TeraFLOPS chip [6]). The varying workloads and NoC traffic profiles which are typical of most parallel applications also contribute heavily to the supply noise. *Thus, it is important to address voltage variations in both processor cores and the NoC, to comprehensively mitigate the negative effects of PSN in CMPs.*

Traditional approaches have addressed the PSN issue at the circuit and micro-architectural levels. Although PSN is most readily observed at the circuit level, the compute intensity and distribution of the workload on the cores decides the magnitude of PSN observed at each cycle. *Hence it is important to address the issue of PSN at a higher level of abstraction than the circuit level.*

In this paper, for the first time, we propose a novel PSN-aware runtime resource management framework (*PARM*) that employs dynamic voltage scaling (DVS), adaptable application degrees of parallelism (DoP), and an intelligent mapping scheme for a NoC-based CMP that operates at near threshold voltages within dark silicon constraints. *PARM* selects the mapping region, V_{dd}, and DoP for every application that arrives at runtime in such a way that the peak PSN in the mapping region and its vicinity is kept below a threshold, minimizing the number of voltage emergencies. Our key contributions in this paper are:

- We study the correlation of PSN with application activity, proximity of concurrently executing threads, and core supply voltages;
- We utilize DVS and adaptable application DoP, to reduce the peak PSN and optimally utilize the available dark silicon power slack while maximizing the number of applications serviced at runtime;
- We propose a novel PSN aware application mapping heuristic for emerging sub-10nm fabricated CMPs, to reduce the PSN in a region and minimize communication latency between tasks;
- We devise a novel PSN-aware routing scheme that balances router activity near highly switching cores, and reduces the impact of the NoC traffic on PSN without incurring any additional latency.

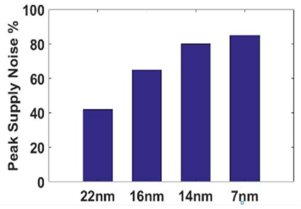

Figure 1: Peak supply noise percentage, relative to the nominal near threshold supply voltage, across fabrication process technology nodes.

2. RELATED WORK

Several techniques have been proposed to cope with PSN at the circuit-level. In [7], conservative noise margins are used to ensure safe operation even when worst-case PSN is observed. In [8], decoupling

DAC '18, June 24–29, 2018, San Francisco, CA, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5700-5/18/06 $15.00
https://doi.org/10.1145/3195970.3196090

capacitors are used to reduce core-to-core voltage interference. In [17]-[18] mechanisms were presented to predict PSN and the occurrence of voltage emergencies. Other efforts have proposed micro-architectural solutions, to reduce inter-core interference, e.g., pipeline throttling, instruction rescheduling, relaxed entry/exit at synchronization barriers, etc. [9]-[15]. But these solutions are primarily designed for single-core or few-core systems. The use of on-die digital sensors was proposed in [16] for the runtime measurement of PSN and to take reactive (corrective) measures post-detection. However, these solutions are not very beneficial in preventing PSN-induced voltage emergencies. Also, the penalty for error correction is expensive when the system is over-subscribed. In [19], Hu et al. proposed a thread mapping and migration scheme for Single Program Multiple Data (SPMD) applications to minimize large voltage fluctuations in CMPs. In [33]-[34], reliability aware task mapping and NoC routing schemes have been proposed to minimize the effects of aging and soft errors. In [35]-[36], Power Delivery Network aware resource management frameworks for voltage islands based 2D and 3D CMPs have been proposed. In [20]-[21], PSN-aware workload assignment schemes are proposed for 2D and 3D CMPs. The schemes map highly active threads at longer Manhattan distances from each other to minimize PSN in a region. However, these schemes do not consider the impact of activity in NoC routers on PSN. In [22], PSN-aware routing and flow control schemes are proposed, to reduce NoC router activity. However, application mapping plays a crucial role in overall NoC activity as tasks separated by longer distances cause more NoC routers to switch. Also, none of these works consider the low voltage margins imposed by NTC to meet dark silicon constraints.

To the best of our knowledge, this is the first work that addresses PSN due to both core and NoC switching activity in the presence of dark silicon power constraints for CMPs executing multi-application workloads and designed at sub-10nm technology.

3. BACKGROUND: MODELS AND ASSUMPTIONS

3.1 Processor Model

We assume a CMP with *N* tiles $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2 \ldots \mathcal{T}_N\}$. Each tile $\mathcal{T}_i$, has a processing core, a NoC router, and L1 instruction and data caches as shown in Fig. 2. The tiles also have a shared global L2 cache, organized in banks. The tiles are connected by a NoC fabric. The CMP is constrained by a dark silicon power budget (DsPB) which is the thermally safe power limit that the cooling system of the chip can operate effectively within. The chip supports dynamic voltage scaling and can operate at different supply voltages $V = \{V_1, V_2 \ldots V_s\}$.

3.2 Application Model

We assume the applications $A = \{A_1, A_2, \ldots A_M\}$ that execute on the CMP to be multithreaded, with each application A_j able to spawn up to *K* threads $\{T_1, T_2 \ldots T_K\}$. Each thread executes on a dedicated core $C_i \in \mathcal{T}_i$. An application A_j can execute with different thread counts hence allowing for variable degree of parallelism (DoP). Each application has a performance deadline constraint.

Application communication requirements are represented by an application graph APG = $G(V, E)$ which is a directed acyclic graph, where each vertex $v_i \in V$ represents a thread and each edge $e_{i,j} \in E$ represents communication volume between thread *i* and thread *j*. All cores that execute the threads of an application are supplied with the same V_{dd}. The applications are stored in a service queue upon arrival at runtime and are considered for mapping to the CMP on a first-come-first-serve (FCFS) basis. Applications are mapped on non-overlapping regions on the CMP for inter-application isolation. In the rest of the paper, the terms *thread* and *task* are used interchangeably.

3.3 Power Delivery Network (PDN) in CMPs

We assume a baseline PDN with multiple independent domains as shown in Fig 2. A domain is a group of four tiles that has its own voltage regulator module (VRM). These domains are physically separated so that there is no interference between tiles from different domains. Each domain is powered by an independent source, which enables efficient monitoring of the power consumption on the chip as the core count scales up. All of the tiles in a domain are supplied with the same V_{dd}, although the actual voltage received at the tile varies due to PSN-induced variation. We assume the presence of digital sensors [16] to monitor the runtime PSN levels at cores and NoC routers. We also assume that tasks of different applications are not mapped into a single domain, which is ensured by limiting the DoP values of each application to be multiples of 4.

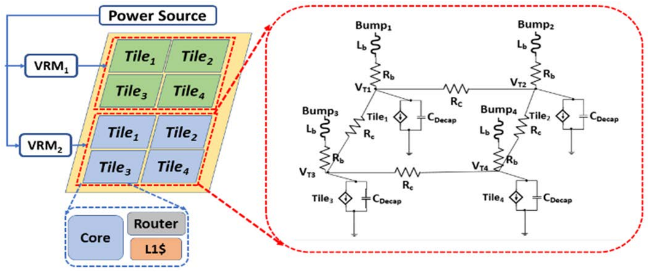

Figure 2: Baseline CMP with power supply domain of four tiles; each tile is powered by a voltage regulator (VRM) connected to a power source.

3.4 Power Supply Noise (PSN) Modeling and Estimation

PSN is caused by *(i)* resistive drop of power delivery wires (*IR*), and *(ii)* inductive droop due to wire inductance ($L\Delta i/\Delta t$). While resistive drop is proportional to current flowing in the wires, inductive droop is proportional to the switching activity of the wires carrying current. We model the PDN as in previous works as shown in Fig 2, where L_b and R_b are inductance and resistance at a bump, R_c is resistance of the PDN wires, and C_{decap} is the decapacitance between cores. The workload on a tile is modeled as a current source, similar to [19]-[21], based on power consumption of the core and NoC router in a tile. The dynamic values of PSN observed at each tile is given by:

$$\Delta V = V_{bump} - V_{Ti} \tag{1}$$

where V_{bump} is the voltage supplied by the source and V_{Ti} is the voltage observed at tile *i* after on-chip parasitic drop. As in [12], we consider a PSN of 5% as a margin for a VE in near threshold voltages that leads to faulty outcomes for a thread executing on the PSN-affected tile.

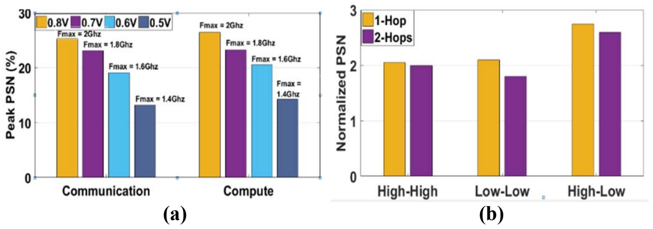

Figure 3: (a) Peak PSN (as % of supply voltage) observed in a domain for communication- and compute-intensive workloads; (b) Normalized PSN due to interference between pairs of tasks of different switching activity (High or Low) and separated by Manhattan distances of 1 and 2 hops.

3.5 Impact of Mapping Decisions and DVS on PSN

PSN is significantly impacted by variations in switching activity of transistors that leads to interference between the current flows in wires. Moreover, as shown in Fig. 3(a), the peak PSN observed in a domain is also directly proportional to its operating voltage (V_{dd}), which decides the maximum operating frequency (F_{max}) of cores and routers in that domain. The trend exists for both communication-intensive and computation-intensive applications. To reduce PSN, one solution is to reduce V_{dd}. However, this also reduces F_{max}, which diminishes application performance. Dynamic adaptation of application DoP is one way to improve performance while running at a low V_{dd} [25].

The switching characteristics of tasks executing in close proximity to each other on a chip also have a considerable impact on PSN. Fig.

3(b) shows interference effects of different combinations of switching activities for two tasks executing on adjacent cores. We categorize application tasks into two bins, "High" and "Low" active, based on extensive analysis of their switching activity. The PSN observed due to interference between tasks with High-Low switching activity is up to 35% higher than tasks with High-High and Low-Low switching activity. Cores running low switching activity tasks get affected by the resistive and inductive interference from the power drawn by high switching tasks running on the neighboring cores in the same domain. This behavior is also observed in [24]. Interestingly, Fig. 3(b) indicates that highly interfering tasks mapped at a distance of 2-hops away interfere up to 10% less than tasks mapped at a distance of 1-hop away. None of the prior works have exploited this observation to reduce the negative impacts of PSN.

Given application performance deadlines and the dark silicon power budget (DsPB) for a CMP, our objective is to use the above observations to opportunistically select a combination of V_{dd}, DoP, and mapping region for each application that arrives for execution at runtime, to minimize the PSN observed in power supply domains. The next section discusses our proposed framework to meet this objective.

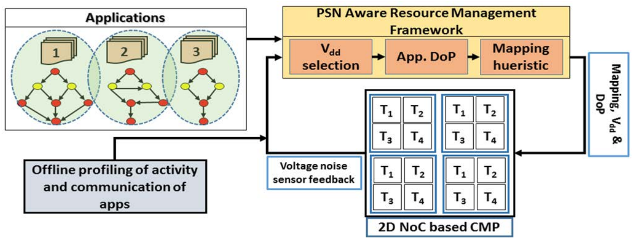
Applications
1
2
3
PSN Aware Resource Management Framework
V_{dd} selection
App. DoP
Mapping hueristic
Mapping, V_{dd} & DoP
T_1 T_2 T_3 T_4
Offline profiling of activity and communication of apps
Voltage noise sensor feedback
2D NoC based CMP

Figure 4: Overview of the proposed PARM framework

4. PSN AWARE RESOURCE MANAGEMENT (*PARM*)

Fig. 4 gives an overview of our proposed *PARM* framework. Offline profiling information about applications, and online voltage noise feedback from on-chip voltage noise sensors are inputs to the framework. The application profiling collects statistics on switching activity, power consumption, and NoC communication characteristics for all of the tasks of an application at different V_{dd}'s and DoPs. The output of the framework is a task to core mapping, V_{dd} assignment, and DoP selection for each application that arrives for execution on the NoC-based CMP with independent power domains. Our *PARM* framework first selects an appropriate V_{dd} and DoP for an application to be mapped, based on the application performance and chip-level DsPB constraints. After V_{dd} and DoP selection, *PARM* uses a PSN aware mapping heuristic, to find a mapping in such a way that the total PSN in PDN domains is minimized; and communication distance between tasks is minimized (to improve performance). The following subsections discuss the components of the *PARM* framework.

4.1 V_{dd} and DoP Selection

Algorithm 1 presents our V_{dd} and DoP selection method. The inputs are a set of permissible voltages sorted in *increasing order* $V = \{V_1, V_2, \dots V_S\}$, and the set of applications waiting in the service queue $A = \{A_1, A_2, \dots A_M\}$. To consume low power (and generate low peak PSN) while ensuring that application deadlines are met, the algorithm starts with the lowest V_{dd} and the highest DoP combination. This is because peak PSN is always low at lower V_{dd} values (Fig. 3(a)). Also, to meet application deadlines, it is intuitive to utilize the available tiles to spawn more number of threads (i.e., use a higher application DoP) without violating the DsPB constraint. To ensure this, the permitted DoP values of an application are sorted and considered in a decreasing order (line 1). The minimum DoP considered in our work is 4.

Our selection algorithm then iteratively searches for a suitable (V_{dd}, DoP) combination. First, the *Worst Case Execution Time* (WCET) of an application for a selected (V_{dd}, DoP) combination is estimated using the *offline application profile data* (line 5) and checked to see if the application deadline constraint will be met (line 6). If the deadline constraint is met, this (V_{dd}, DoP) combination is sent as an input to the *PSN-aware mapping heuristic* (line 7; this heuristic is discussed in the next subsection). If the mapping is successful, the next application in the service queue is processed (line 8). If not, the algorithm waits till the CMP completes a currently executing application (which would free up tiles for mapping), and tries again to find a mapping region (lines 9-11). If the algorithm fails to find a mapping region, it selects the next DoP (lower value) from the list *D*, and performs the same operations as above (line 12). This can be useful to map an application when there are a lack of sufficient number of tiles, or a limited power budget. Selecting a lower DoP would resolve both of these concerns. However, if the estimated WCET (from line 5) does not meet the application deadline constraint, the algorithm skips iterating through DoPs, as the lower values of DoP cannot satisfy the deadline constraint, and continues searching for a new (V_{dd}, DoP) combination with the next V_{dd} from the list *V*, that is higher than the current V_{dd} value (line 13). If the deadline constraint is not met or if the mapping region is not found on the CMP after exploring all V_{dd} and DoP combinations, the current application is dropped and the next waiting application is processed, to avoid stagnation in the service queue due to the stalled application.

Algorithm 1: V_{dd}, DoP selection

Inputs: V_{dd} values sorted in increasing order $V = \{V_1, \dots V_S\}$, applications $A = \{A_1, \dots A_M\}$

```
1:  for all $A_j$ in $A$ do
2:      $D \leftarrow$ Sort($A_{i}$.{ $D_1, \dots D_T$ }) //sorted in descending order
3:      for all $V_i$ in $V$ do
4:          for all $D_k$ in $D$ do
5:              WCET $\leftarrow$ EstimateExecutionTime($V_i, D_k, A_j$)
6:              if WCET < Deadline ($A_j$) then
7:                  if PSNAwareMapping ($V_i, D_k, A_j$) is successful then
8:                      goto line 1 (continue to next app in A)
9:                  else stall till an app exit event on CMP
10:                     if PSNAwareMapping ($V_i, D_k, A_j$) is successful then
11:                         goto line 1 (map the next app in A)
12:                     else goto line 3 (Try with lower $D_k$,)
13:             else goto line 2 (Try with next $V_i$)
```

Outputs: V_{dd}, DoP, and a valid region to map the application

4.2 PSN Aware Mapping Heuristic

Given a V_{dd} and DoP that satisfy the deadline constraint for the application to be mapped, we next attempt to find a mapping that fulfils the CMP dark silicon (DsPB) constraint and minimizes PSN in CMP power domains, as well as minimizing the total Manhattan distance of communication between tasks. This can be formulated as multi-objective optimization problem which has been shown to be NP-Hard. Traditional multi-objective optimization methods (e.g., integer programming, genetic algorithms) to solve the problem are too slow for decision making at runtime. Hence, we propose a fast runtime heuristic to select a suitable mapping region and meet all constraints.

Algorithm 2: PSN Aware Mapping

Inputs: V_{dd}, DoP, application A, Sorted APG edges $A(E) = \{e_{12}, \dots, e_{nn-1}\}$

```
1:  if EstimatedPowerConsumption(A) > DsPB then
2:      return False  //unable to find viable mapping
3:  H $\leftarrow$ {Ø} L $\leftarrow$ {Ø}  // Set of clusters
4:  for all $e_i$ in A(E) do
5:      for each tasks $T_j$, connected to $e_i$ do
6:      if $T_j$. $\notin$ H or $T_j$. $\notin$ L then
7:          if $T_j$ .High then push_back($T_j$,H)
8:          else push_back($T_j$,L)
9:  num_cluster $\leftarrow$ create_clusters(H, L)
10: if num_available_domains < num_cluster then
11:     return False  //unable to find viable mapping
12: else
13:     task-cluster-to-domain-mapping()
14:     return True
```

Output: Successful mapping or indication of failed mapping

Algorithm 2 shows our PSN-aware mapping approach. Given V_{dd} and DoP for the application A as inputs, the mapping heuristic aims to map all of the tasks of application A on to the CMP without violating the DsPB constraint, while minimizing the observed PSN. The algorithm first checks if the power consumption estimated from offline profiling is more than the available DsPB and returns false if the condition is not met without going further (lines 1-2). The tasks are labeled as high switching active or low switching active based on offline profile data. The heuristic utilizes the application graph *APG* of the application to be mapped to extract a sorted list of edges in the decreasing order of edge weights (communication volumes). To reduce PSN due to inter-task interference, the heuristic maps as many tasks with similar switching activity into the same power supply domain as possible. To reduce the NoC traffic, the heuristic also tries to map tasks with the highest communication volumes in the same domain. To achieve this, the heuristic iterates through the sorted edge list and creates clusters of 4 tasks, corresponding to the power supply domains of 4 cores. As the tasks are categorized into two types (high and low switching), we create two lists corresponding to the two task types (line 3). The algorithm checks if tasks connected to the edge being evaluated are already assigned to a list (line 4); if not, they are pushed to one of the two lists (lines 5-8).

The two lists end up with tasks arranged in the decreasing order of communication volumes, as the edges have been evaluated in the decreasing order of their weights. Each list is then divided into clusters of four tasks in the order in which they are stored in the list (line 9). Any remaining un-clustered tasks from each list (< four; if the list size is not a multiple of four) are grouped into a single cluster. Clustering is done to ensure that (1) all but one of the created clusters will have tasks with similar switching activity, to be mapped in the same domain, (2) tasks with high communication volume between them are not mapped far from each other. If the available domains are less than the number of clusters (line 10) the algorithm returns false (i.e., no mapping found). If there are sufficient number of domains, each task cluster is mapped on to domains (line 13), in a manner that minimizes the hop distance between inter-domain mapped tasks. Further details of this step are omitted due to lack of space.

Fig. 5 presents an example of the mapping heuristic for an APG and a sorted edge list. When mapping a task cluster on to a domain, if a cluster has two tasks of each switching activity level, tasks of the same level are mapped adjacent to each other, as shown in Fig. 5, to reduce PSN due to inter-task interference (Section 3.5). On successful mapping, the heuristic returns true (line 11), indicating a successful mapping). After mapping, the tasks of the mapped application are scheduled using the fast and efficient earliest deadline first (EDF) scheduling scheme. For EDF, each task is assigned a deadline (priority) based on the deadline of the entire application, using a technique proposed in our prior work on task-graph scheduling [23].

4.3 Time Complexity Analysis of *PARM*

The V_{dd} and DoP selection step runs in linear time complexity with respect to the permissible V_{dd} and DoP levels ($|V| = 5$, $|D| = 4$ in this work). In the mapping step, task clustering runs in linear complexity with respect to the number of edges in the APG. The total number of possible edges in an APG is $T\times(T+1)/2$, where T is the total number of tasks of an application. So, the clustering step has $O(T^2)$ complexity. *Task-cluster-to-domain-mapping()* has linear complexity with respect to number of tiles, hence the mapping step takes O(Ƭ), where Ƭ is the total number of tiles in the CMP. The runtime complexity of EDF scheduling scheme, given by $O(T\times\log T)$, where T is the total number of tasks of an application, is masked by running in parallel with the mapping scheme that takes longer. So, *PARM* runs with a complexity of $O(V\times D\times\{max(Ƭ, T^2)\})$ which depends on the number of CMP tiles, or number of tasks of an application, while V and D are small integers.

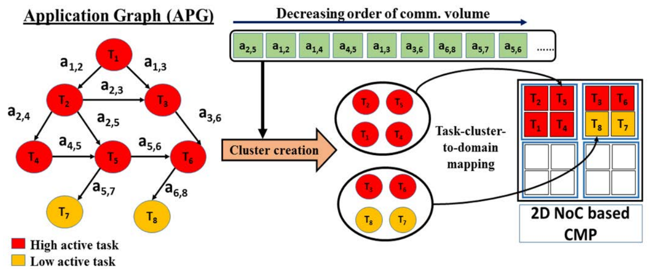
Application Graph (APG)
Decreasing order of comm. volume
a2,5 a1,2 a1,4 a4,5 a1,3 a3,6 a6,8 a5,7 a5,6 ......
T1 T2 T3 T4 T5 T6 T7 T8
a1,2 a1,3 a2,3 a2,4 a2,5 a3,6 a4,5 a5,6 a5,7 a6,8
Cluster creation
Task-cluster-to-domain mapping
2D NoC based CMP
High active task
Low active task

Figure 5: Overview of PSN aware mapping heuristic

4.4 PSN and Congestion Aware NoC Routing (*PANR*)

To complement our PSN-aware mapping framework (*PARM*), we propose a *PSN- and congestion-aware NoC routing* scheme (*PANR*). *PANR* builds on and enhances a deadlock-free turn model based routing scheme called west-first routing [32]. Algorithm 3 shows the decisions made at each hop in a route. For each header flit in the input channel buffers of a NoC router, the routing scheme first computes the permitted destination hop directions (lines 1-3). It then selects a hop direction, from a set of permitted directions, by considering the voltage noise sensor data and incoming data rate (flits/cycle) from the routers in the tiles that are adjacent to the current tile. If the buffer occupancy of the input channel is beyond a threshold B, the output direction with the least incoming data rate is chosen to minimize the congestion (line 5). In the case of lower buffer occupancy than B, the output direction with the least observed PSN is chosen (line 6), to reduce the activity in the router in that direction, which in turn minimizes the overall PSN observed in the domain. Once a direction is decided for a header flit, the remaining flits in a packet follow the header flit.

Algorithm 3: PSN and Congestion Aware NoC Routing

Inputs: destination tile coordinates, PSN activity of adjacent tiles, traffic load in adjacent NoC routers

```
1:  for each channel in Input_channels do
2:     flit ← channel.packet.header_flit
3:     {permissible directions} ← WestFirstRouting (flit.src, flit.dest)
4:     if channel.buffer_occupancy > B then
5          hop ← min_data_rate{permissible directions}
6:     else hop ← min_PSN{permissible directions}
7:  return hop
```

Output: Next hop direction for packets in input channels

Overhead computation: The overhead of our routing scheme involves registers to store the values of noise and router traffic levels of the adjacent tiles, and additional wires to transmit those values between tiles. In addition, two 64-bit comparators are used per router to find the minimum values of PSN and incoming data rates of adjacent routers. The additional circuitry at each router consumes ~1 mW (3%) power and ~115 μm^2 (0.5%) area overhead over the baseline NoC router, at the 7nm node. Hop selection takes 1 cycle in a router, at 1 GHz. This latency is masked by executing the selection step in parallel with the route computation step. The overhead of the network of digital PSN sensors used to sense the voltage noise [16], is around 413μm^2 which is negligible compared to the core area which is ~4 mm^2, and the router area of ~71300 μm^2, at a 7nm FinFET node.

4.5 Fault Detection and Correction

Our proposed PSN aware mapping and routing minimizes voltage fluctuations due to PSN. However, there may still be some cases when inter-core interferences lead to PSN above a certain threshold which leads to voltage emergencies (VE). VEs have the potential to cause errors in the functionality of logic devices and faulty application execution. To prevent such scenarios, applications are checkpointed at periodic intervals [26]. When a VE is detected using on-chip sensors in a tile that runs an application, it is rolled back to its last saved checkpoint and begins execution from there

5. EXPERIMENTS

5.1 Simulation Setup

We conducted experiments on 13 different parallel applications from the SPLASH-2 [28] and PARSEC [29] benchmark suites. We used the GEM5 [30] multicore simulator to generate the offline profile data for applications. We modeled the PDN as discussed in section 3.3 using the SPICE simulator. We estimated the power consumption of different applications at various V_{dd}, clock frequency values, and app-DoP, at a 7nm FinFET technology node, using data from McPAT [31] and ITRS [4]. The DoP values used range from 4 to 32 (in multiples of 4, beyond which most of the applications were observed to have lower performance due to communication (synchronization) overheads. The switching activity of the core and NoC router in a tile was observed to be proportional to its power consumption. We also sampled PSN values of the tiles at periodic intervals, and when a new application begins or a current one ends execution on the CMP, using the PDN SPICE model. The buffer occupancy threshold *B* for hop selection in *PANR* is set to 50% after analyzing the effects of different occupancy levels on router throughput, with a cycle-accurate NoC simulator.

We categorized 13 benchmarks into two groups: (i) communication-intensive benchmarks: *{cholesky, fft, radix, raytrace, dedup, canneal, vips}*; and (ii) compute-intensive benchmarks: *{swaptions, fluid-animate, streamcluster, blackscholes, radix, bodytrack, radiosity}*. As *radix* has properties of both, we use it in both groups. We employed three sequences of application with up to 20 applications picked randomly from each of the groups mentioned above. The sequences are categorized as compute-intensive, communication-intensive, and mixed, according to the type of applications used in each sequence. We also experimented with three different inter-application arrival rates of 0.2s, 0.1s, and 0.05s to test the efficiency of our framework for different CMP utilization (workload subscription) scenarios.

We consider a 60 core 2D NoC-based CMP designed at 7nm FinFET technology node for our studies. The tiles are arranged in a 10 ×6 mesh layout. Each tile has an ARM Cortex A-73 low power mobile core, a NoC router, and a private L1 cache. We assume that *PARM* is part of the OS or middleware that assigns V_{dd}, DoP, and task-to-core mapping regions to each application according to the availability of DsPB and idle cores at runtime. The V_{dd} values supported by each tile (core + router) are between 0.4V (NTC) to 0.8V in steps of 0.1V. We assume that all of the cores on which an application is mapped to run at the same V_{dd}. We assume a dark silicon power budget (DsPB) of 65W. We treat PSN above 5% as a voltage emergency, similar to prior work [12]. We assume an overhead of ~256 cycles for periodic checkpointing with a 1ms checkpoint period, and ~10000 cycles of overhead for rolling back (restart) to an earlier state, after an error.

5.2 Simulation Results

We compare our *PARM* framework against a prior work [21] that tries to minimize PSN using a harmonic mapping scheme, where tasks with high activity are mapped far away from each other. We refer to the scheme as *HM*. To show the impact of NoC routing on PSN, we compare our proposed *PANR* routing scheme with an *XY* routing scheme, and a scheme from prior work *ICON* [22] that minimizes PSN in NoCs, but is agnostic of application mapping. Overall, we evaluate six combinations of comparison works, *HM+XY, HM+ICON, HM+PANR, PARM+XY, PARM+ICON, PARM+PANR.* The prior works *HM* and *ICON* do not consider fault detection and correction in the event of failures due to VEs. For a fair comparison, we assume the presence of fault detection and correction mechanisms in these works, as in our proposed work. Not having such mechanisms would result in unpredictable outcomes for applications, which is not desirable.

We simulated and analyzed total execution time (Fig. 6), peak and average PSN (Fig. 7), and total number of applications executed successfully (Fig. 8), for the six frameworks mentioned above, across different types of workloads.

<u>Compute intensive workload:</u> As shown in Fig. 6 and Fig.7, *PARM+PANR* shows up to 25.4% improvement in execution time and 4.15× improvement in peak PSN observed compared to *HM+XY*, whereas *PARM+ICON* and *PARM+XY* show 23.3% (3.82×) and 20.3% (3.81×) improvements in execution time (peak PSN) respectively. The *HM* framework does not consider the negative effect of mapping tasks with varying switching intensities in close proximity on PSN, as shown in Fig. 3(b) in Section 3. This leads to high PSN and hence poor performance of applications with *HM*. *PANR* routes flits through less congested and low switching paths to minimize PSN in the tiles that execute tasks with high switching activity, leading to lower PSN (Fig. 7). *ICON* and *XY* routing schemes are unaware of the core activity and create a higher PSN (> 5%) (Fig. 7) and hence have (a few) more number of VEs than *PANR*. Hence, *HM+PANR* has lower execution time over *HM+XY* and *HM+ICON*, due to its PSN aware routing scheme that minimizes overall PSN of cores and routers, and avoids VEs. Thus *for compute intensive workloads, intelligent task mapping and packet routing is crucial to minimize PSN.*

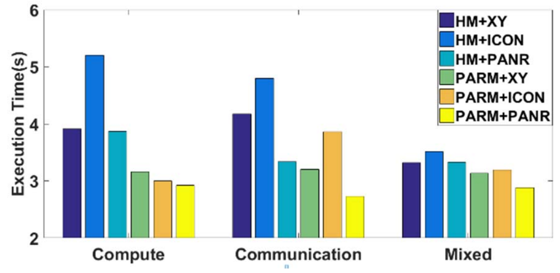

Figure 6: Total time taken to execute 20 applications with different frameworks across different types of workloads.

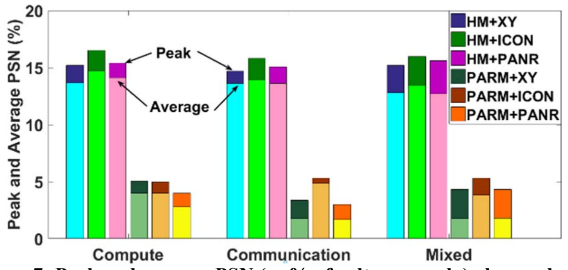

Figure 7: Peak and average PSN (as % of voltage supply) observed with different frameworks across different types of workloads

<u>Communication intensive workload:</u> NoC components consume 18-20% of the chip power, and are in the critical path of the application performance, when running communication intensive workloads. Hence it is important to consider the effects of NoC routers on PSN, along with cores. Fig. 6 and Fig.7 show that *PARM+PANR* gives 34.3% improvement in execution time and 4.5× improvement in observed PSN compared to *HM+XY*. This is because in communication intensive workloads, PSN-aware mapping and PSN-aware NoC routing work in tandem to reduce overall packet latency while keeping the router activity low around highly active cores. *ICON* only considers router activity and ignores the activity in the cores while making decisions. Hence PSN is higher in *ICON* compared to *XY* and *PANR* in both *HM* and *PARM* mapping schemes, as shown in Fig. 7. This increases the tile activity and results in more VEs and worse application execution time in both *HM+ICON* and *PARM+ICON*. *PARM+XY* shows 23.3% performance improvement compared to *HM+XY* due to the PSN-aware mapping heuristic. Even though PSN observed with *PARM+XY* is much lower than *HM+XY* (Fig. 7), there is an increased NoC traffic and higher packet latency which slows down the application when the application DoP is high with *PARM*. This case is handled better by the *PANR* routing scheme when compared to *XY*. *HM+PANR* is aware of the tile PSN and hence

balances the activity with its intelligent routing scheme. This leads to lower PSN than *HM+ICON* and *HM+XY* (Fig. 7), as well as 20% improvement in execution time compared to *HM+XY* (Fig. 6). This proves that *PSN aware NoC routing schemes provide better results when they are combined with a PSN aware mapping scheme.*

Mixed workload: When both types of applications are executed in a sequence, *PARM+PANR* give improvements of ~13.1%, which is better than *PARM+XY* (5.7%), *PARM+ICON (6.5%)* compared to *HM+XY*.

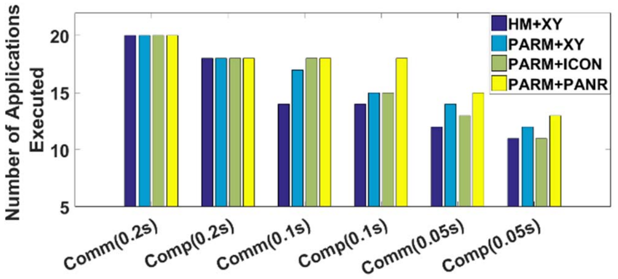

Figure 8: Total number of application successfully completed across different workload types and arrival rates for different frameworks.

Lastly, we analyze the efficiency of various mapping frameworks for different application arrival rates at runtime. We compare *HM+XY* with *PARM+XY, PARM+ICON,* and *PARM+PANR* across three different inter-application arrival rates and two workload types (with 20 applications each, as in earlier experiments). From Fig. 8, it can be observed that all of the frameworks perform similarly at the 0.2s arrival rate since the applications arrive at a relatively slower rate. This slow arrival rate allows more applications in the queue to be mapped within the given DsPB and deadline constraints. For communication intensive workloads, *PARM+PANR* maps 38(25)% more applications than *HM+XY,* and 17(9)% more application when the arrival rate is 0.1s(0.05s), compared to *PARM+XY*. For compute intensive workload *PARM+PANR* maps 38(18)% more applications than *HM+XY,* and 29(11)% more application when the arrival rate is 0.1s(0.05s) compared to *PARM+XY* and *PARM+ICON*. *HM* maps tasks in non-contiguous regions unlike *PARM* and scatters them across the CMP. However, *HM* fails to do so without violating the DsPB constraint because of its increased power consumption (due to high V_{dd}). As *PARM* adaptively reduces V_{dd} and increases DoP to reduce PSN, it fits more number of applications within a given DsPB.

6. CONCLUSION

In this paper, we proposed a novel runtime framework called *PARM* that minimizes PSN in near threshold voltages while meeting application performance deadlines without violating the DsPB constraint. We also proposed a PSN aware NoC routing scheme called *PANR* which routes flits through tiles with low switching activity to reduce the overall PSN in the CMP. Our experiments show that *PARM* enables up to 4.5× reduction in peak and average PSN observed, and up to 34.3% improvement in application execution times compared to the state-of-the art in PSN-aware runtime application mapping. *PARM* can be used to minimize the hardware overhead due to costly guardbanding techniques and decapacitance circuits to reduce the effect of interferences between cores, and minimize the software overhead due to schemes such as thread migration employed to keep the tile switching activity in check. Our experimental results also show that reducing PSN in both cores and NoC routers is crucial for improvement in application performance.

REFERENCES

[1] M. Gupta, et al. "Understanding voltage variations in chip multiprocessors using a distributed power-delivery network." Proc. DATE, 2007.
[2] H. Esmaeilzadeh et al. "Dark silicon and the end of multicore scaling," Proc. ISCA '11.
[3] J. Henkel, et al. "New trends in dark silicon." Proc. ACM DAC, 2015.
[4] International Tech Roadmap for Semiconductors, http://www.itrs2.net
[5] N. Pinckney, et al. "Near-threshold computing in FinFET technologies: Opportunities for improved voltage scalability." Proc. DAC, 2016.
[6] Y. Hoskote, et al. "A 5-GHz mesh interconnect for a teraflops processor." Proc. IEEE Micro, 2007.
[7] N. James, et al. "Comparison of split-versus connected-core supplies in the power6 microprocessor," Proc. ISSCC, 2007
[8] J. Gu, et al. "On-chip supply noise regulation using a low-power digital switched decoupling capacitor circuit." Proc. IEEE JSSC, 2009.
[9] M. Powell, et al. "Pipeline damping: a microarchitectural technique to reduce inductive noise in supply voltage." ACM Comp. Arch., 2003.
[10] T. Miller, et al. "VRSync: Characterizing and eliminating synchronization -induced voltage emergencies in many-core processors." SIGARCH 2012
[11] A. Paul, et al. "Staggered Core Activation: A Circuit/architectural approach for mitigating resonant supply noise issues in multi-core multi-power domain processors." Proc. CICC, 2012.
[12] V. J. Reddi, et al. "Eliminating voltage emergencies via software-guided code transformations." ACM TACO, Vol. 7, No. 2, 2012.
[13] M. S. Gupta., et al. "An event-guided approach to reducing voltage noise in processors." Proc. DATE, 2009.
[14] M. Healy, et al. "A unified methodology for power supply noise reduction in modern microarchitecture design." IEEE ASPDAC 2008.
[15] L .Zheng, et al. "Full-chip power supply noise time-domain numerical modeling and analysis for single and stacked ICs." IEEE TED, 2016.
[16] M. Sadi, et al. "Design of a network of digital sensor macros for extracting power supply noise profile in SoCs." TVLSI 24(5), 2016.
[17] Y. Chen, et al. "Pattern based runtime voltage emergency prediction: An instruction-aware block sparse compressed sensing approach." IEEE ASP-DAC, 2017.
[18] V. J. Reddi, et al. "Voltage emergency prediction: Using signatures to reduce operating margins" Proc. IEEE HPCA, 2009
[19] X. Hu, et al. "Orchestrator: a low-cost solution to reduce voltage emergencies for multi-threaded applications." Proc. IEEE DATE, 2013.
[20] Y. Cheng, et al. "Power supply noise-aware workload assignments for homogeneous 3D MPSoCs with thermal consideration." ASPDAC, 2014.
[21] N. Dahir, et al. "Minimizing power supply noise through harmonic mappings in networks-on-chip." Proc. IEEE/ACM CODES+ISSS, 2012.
[22] P. Basu, et al. "IcoNoClast: Tackling Voltage Noise in` the NoC Power Supply Through Flow-Control and Routing Algorithms." TVLSI, 2017.
[23] Y. Xiang, S. Pasricha, "Soft and Hard Reliability-Aware Scheduling for Multicore Embedded Systems with Energy Harvesting", IEEE *TMSCS*, vol. 1, no. 4, pp. 220-235, Oct-Dec. 2015.
[24] A. Todri, et al. "Power supply noise aware workload assignment for multi-core systems." Proc. IEEE/ACM ICCAD, 2008.
[25] N. Kapadia, S. Pasricha, "VARSHA: Variation and reliability-aware application scheduling with adaptive parallelism in the dark-silicon era." Proc. DATE, 2015.
[26] Q. Han et al. "Energy minimization for fault tolerant scheduling of periodic fixed-priority applications on multiprocessors" DATE, 2015.
[27] R. Teodorescu, "SWICH: A prototype for efficient cache-level checkpointing and rollback." Proc. IEEE Micro, 2006.
[28] S.V. Woo et al., "The SPLASH-2 programs: characterization and methodological characterization" Proc. ISCA, 1995.
[29] C. Bienia et al., "The PARSEC benchmark suite: characterization and architectural implications," PACT, 2008.
[30] N. Binkert, et al. "The gem5 simulator." ACM Comp. Arch. News 2011.
[31] L. Sheng et al. "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures" MICRO, 2009
[32] J. C. Glass et al. "The turn model for adaptive routing." ACM SIGARCH Computer Architecture News 20.2, 1992.
[33] V. Y. Raparti, N. Kapadia, S. Pasricha, "ARTEMIS: An Aging-Aware Runtime Application Mapping Framework for 3D NoC-based Chip Multiprocessors", IEEE TMSCS, Vol. 3, No. 2, pp. 72-85, Apr-Jun 2017.
[34] V. Y. Raparti, S. Pasricha, "CHARM: A Checkpoint-based Resource Management Framework for Reliable Multicore Computing in the Dark Silicon Era," IEEE ICCD, Oct 2016.
[35] N. Kapadia, S. Pasricha, "VISION: A Framework for Voltage Island Aware Synthesis of Interconnection Networks-on-Chip", Proc. ACM GLSVLSI, May 2011.
[36] N. Kapadia, S. Pasricha, "A Power Delivery Network Aware Framework for Synthesis of 3D Networks-on-Chip with Multiple Voltage Islands", Proc. IEEE VLSID, Jan. 2012.