Shift Sprinting: Fine-Grained Temperature-Aware NoC-based MCSoC Architecture in Dark Silicon Age

Amin Rezaei[1], Dan Zhao[1], Masoud Daneshtalab[2], and Hongyi Wu[1]

[1] University of Louisiana at Lafayette (ULL), Lafayette, USA
(me@aminrezaei.com, dzhao@cacs.louisiana.edu, wu@cacs.louisiana.edu)
[2] Royal Institute of Technology (KTH), Stockholm, Sweden
(masdan@kth.se)

ABSTRACT
Reliability is a critical feature of chip integration and unreliability can lead to performance, cost, and time-to-market penalties. Moreover, upcoming Many-Core System-on-Chips (MCSoCs), notably future generations of mobile devices, will suffer from high power densities due to the dark silicon problem. Thus, in this paper, a novel NoC-based MCSoC architecture, called Shift Sprinting, is introduced in order to reliably utilize dark silicon under the power budget constraint. By employing the concept of distributional sprinting, our proposed architecture provides Quality of Service (QoS) to efficiently run real-time streaming applications in mobile devices. Simulation results show meaningful gain in performance and reliability of the system compared to state-of-the-art works.

Keywords
MCSoC; NoC; Dark Silicon; Sprinting; Reliability; Temperature

1. INTRODUCTION
As mobile devices, such as tablets and smart phones, get ever more sophisticated in their functionality, more internal heat-producing components are added to them. Keeping components cool is one of the most important challenges in mobile device industry and becomes even more severe by semiconductor technology scaling. Moreover, overheating causes significant reductions in the operating life of a device and uncertainties in reliability can lead to performance, cost, and time-to-market penalties [1]. By the same token, for upcoming Many-Core System-on-Chips (MCSoCs), dark silicon is anticipated to dominate most of the chip area since Dennard scaling [2] fails because of the voltage scaling limitations. Notably in mobile devices, which have limited cooling options, voltage scaling problem leads to a utilization wall [3] in which sustained chip performance is limited mainly by power rather than area. It means most of the cores will lie inactive (i.e. dark) in upcoming technologies.

In this paper, a fine-grained NoC-based MCSoC architecture, called *Shift Sprinting*, is introduced in order to reliably utilize dark silicon under the power budget constraint. The key contributions are twofold:

• Proposing a novel NoC architecture to gain high-performance by utilizing the intervals between cool-down periods of the cores

• Presenting an innovative temperature-aware application running scheme to gain high-reliability by using simultaneous techniques of core sprinting, application migration, and power-gating

DAC '16, June 05-09, 2016, Austin, TX, USA

DOI: http://dx.doi.org/10.1145/2897937.2898090

The rest of the paper is organized as follows. Section 2 reviews backgrounds and related works. Preliminaries and motivations of the proposed architecture are presented in Section 3. Shift sprinting architecture including core behavior model, system topology, application migration scheme, and controlling mechanism is proposed in Section 4. Experimental results and comparison with state-of-the arts architectures are presented in Section 5. Finally, conclusion and dedication are given in Section 6 and Section 7 respectively.

2. BACKGROUNDS AND RELATED WORKS
Due to the dark silicon problem, the threshold voltage cannot be scaled without exponentially increasing leakage, and as a result, the operating voltage should be kept roughly constant. This is an exponentially worsening problem that accumulates with each process generation [4]. Recent studies [5] have predicted that, on average, 52% of a chip's area will stay dark for the 8nm technology node. The author in [4] has discussed four key solutions emerged as top contenders for thriving in the dark silicon age. Each class requires a careful understanding of the underlying tradeoffs and benefits.

In addition, Computational Sprinting [6] is presented by using of phase-change materials to allow chips to exceed their sustainable thermal budget for sub-second durations, providing a short but substantial computational boost. Furthermore, NoC-Sprinting [7] is proposed, in which the chip selectively sprints to any intermediate stages instead of directly activates all the cores in response to short-burst computations. Moreover, a fine-grained voltage scaling [8] is proposed in order to allow on-chip voltage regulation.

The mode-switching in computational sprinting lacks adaptability and only considers two states of single-core operation or all-core sprinting. However, based on the behaviors of running applications, some in-between numbers of active cores may be sufficient for reaching the optimal performance speedup with less power consumption. On the other hand, NoC-sprinting lacks reliability and does not clearly consider the cool-down period requires for each core after sprinting phase. By assigning sprinting periods to all or some fixed cores of the system, both of the mentioned techniques are categorized in periodical sprinting class. However to address periodical sprinting problems, we introduce the concept of distributional sprinting by utilizing cool-down period to provide high-performance for media streaming while using the dark silicon to become with a reliable temperature-aware application running scheme.

3. PRELIMINARIES AND MOTIVATIONS
According to [9], smart-phone penetration is increased to 78% among people. Moreover, around 95% of the devices sold recently are smart-phones. Nowadays, users are using their smart-phones not only for daily communication but also increasingly for media streaming. Thus, the Quality of Service (QoS) of real-time streaming applications will become more and more important for future generations of mobile devices. Furthermore, reliability issues are highly challengeable in the future mobile devices due to the limited cooling options. In short, designing reliable mobile devices to provide QoS-aware real-time streaming for the users is crucially important in dark silicon age.

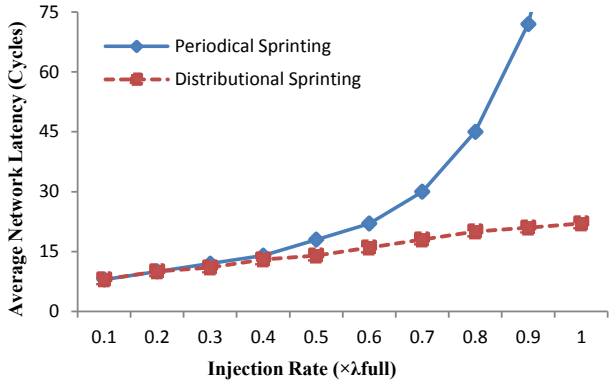


Figure 1. Average network latency for 2×2 NoC-based MCSoC with periodical sprinting and distributional sprinting

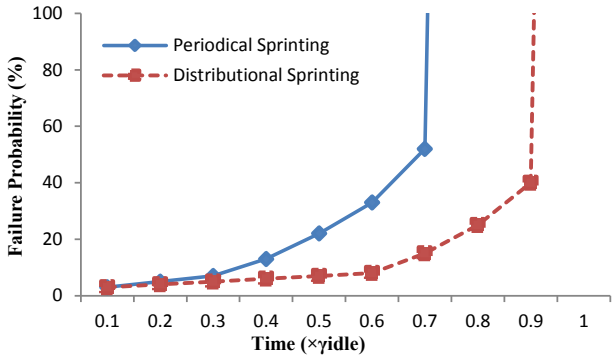


Figure 2. Failure probability for 2×2 NoC-based MCSoC with periodical sprinting and distributional sprinting

As a preliminary study, a 2×2 NoC-based MCSoC is simulated under uniform streaming traffic to show the necessity of high-performance and high-reliability demands of real-time streaming applications. By an optimistic assumption each sprinting period is considered to be equal to the cool-down period. Moreover each core in sprinting status is supposed to gain performance by a factor of 4 and to lose life-span by a factor of 2 compared to the core in nominal status. Maximum traffic injection rate and the average life-time of the system running in idle status are called λ_{full} and γ_{idle} respectively.

3.1 High-Performance Demands

It is shown in [6] and [7] that for serving short burst computations, interleaving the status of the cores between nominal and sprinting along with cool-down intervals can be beneficial. Since in sprinting status the core is operating on higher than the Thermal Design Power (TDP) constraint, phase change of the core internal materials can be used to tolerate such situation. It is assumed that the core temperature stays constant for a specified time during the melting phase of the materials.

However, applying short burst computations is not enough to fully support real time streaming applications (e.g. watching a live football match) because these applications require continues high-computation demands in order to provide QoS to the users. Figure 1 shows the average network latency for both periodical sprinting and distributional sprinting in a 2×2 NoC-based MCSoC. As can be seen, applying periodical sprinting to some fixed cores does not solve the problem neither since it still requires full cool-down intervals. On the other hand, migrating the running application to the dark cores in distributional sprinting utilizes cool-down periods and can provide appropriate QoS to the users. Moreover, By increasing the traffic injection rate, the weakness of periodical sprinting over distributional sprinting is fully observed.

3.2 High-Reliability Demands

Allowing periodical sprinting to some specific cores while let others stay at dark greatly increases the permanent failure probability of those highly active cores. Hence, distributing the sprinting periods through the whole system can reduce the core malfunction possibility. Figure 2 demonstrates the failure probability of the system over time in both periodical sprinting and distributional sprinting in a 2×2 NoC-based MCSoC under the injection rate of $0.5\lambda_{full}$. It is supposed that failure of a single core leads to the whole system failure. As can be seen, the failure probability of periodical sprinting is almost 1.5x as likely as distributional sprinting.

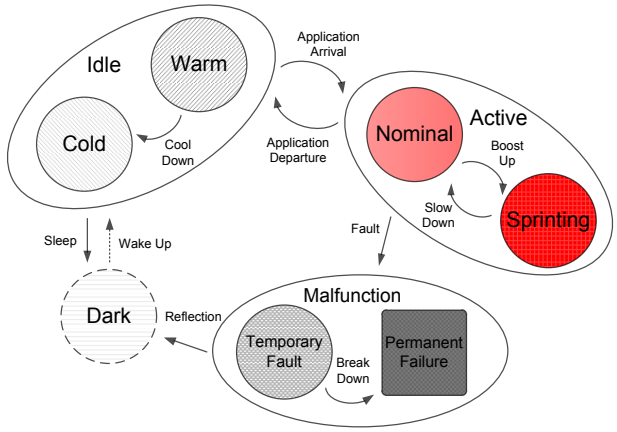


Figure 3. Core behavior model of shift sprinting

4. SHIFT SPRINTING ARCHITECTURE

In this section, by employing the concept of distributional sprinting, we propose a novel fine-grained temperature-aware NoC-based MCSoC architecture named *Shift Sprinting*, targeted at increasing both performance and reliability of the system in the upcoming age of dark silicon.

4.1 Core Behavior Model

The core behavior model of shift sprinting is depicted in Figure 3. Each core has four main states including *Dark*, *Idle*, *Active*, and *Malfunction*.

Dark: In the future generations of MCSoCs, the common state of the core is dark. In this state the core is power-gated.

Idle: After waking up the core, it goes to idle state. (i.e. the core is powered on but still no application is assigned to it.) Moreover, after the application departure, the core goes to warm status until it cools down and reaches the cold status. Then it goes back to dark state.

Active: When an application arrives to the core, it goes to active state. In active state the status is interchangeable between nominal and sprinting. In nominal status, the core is operating under the TDP constraint. On the other hand, in sprinting status, the core is operating on higher than the TDP for a temporary period in order to accelerate the process.

Malfunction: In the case of fault happening, the core state is changed to malfunction. If the fault is temporary, after resolving the problem, the core can goes back into the normal cycle; otherwise, it comes to permanent failure.

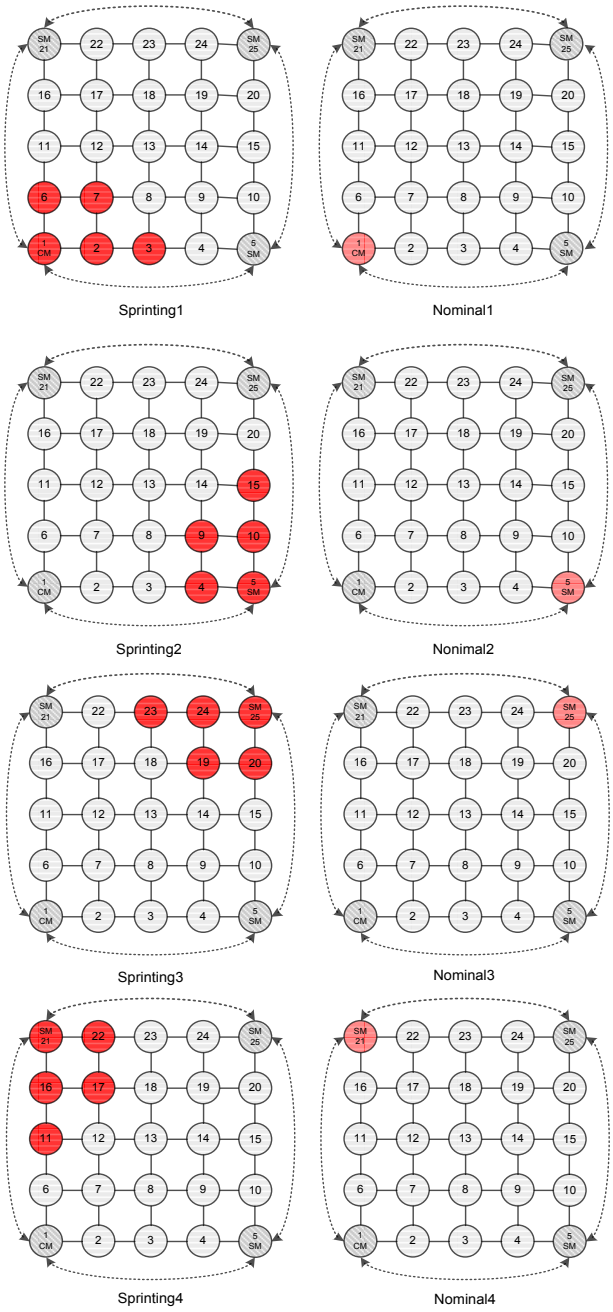


Figure 4. A 5×5 shift sprinting architecture

4.2 System Topology

Rather than abandon the benefits of transistor density scaling, some cores are transiently allowed to operate on higher than the TDP. The mobile platform trend shows that the future MCSoCs with the same die area as current mobile chips will have enough dark cores (i.e. on average 52% [5]) to support additional cores during sprinting [6]. Without loss of generality, the topology of shift sprinting is considered as the 2-D mesh NoC. As an example we have 8 different phases (i.e. 4 sprinting and 4 nominal) in shift sprinting shown in Figure 4. The shift can happen between different phases whenever necessary. In nominal phases, a single core is operating under the TDP constraint. On the contrary, in sprinting phases, thermal capacitance of chosen cores is increased over short timescales to boost-up the process. (i.e. the sprinted cores operate on higher than the TDP.) Based on each application characteristics, the number of sprinted cores required to provide maximal performance speedup varies.

In most of the available dynamic application mapping techniques in the literature, one core is already dedicated to the Central Manager (CM) [10]. In shift sprinting, CM is also used to globally control application migration process. In order to accelerate the application migration process, in addition to CM, there are three Sub-Managers (SMs) that are responsible for collecting information from the other cores. Each core sends and receives information from its nearest manager. For controlling the application migration process efficiently, the managers have created a virtual ring network [11]. As can be seen from Figure 4, CM is resided in one corner (i.e. core #1) and three SMs are resided in the other corners (i.e. core #5, core #21, and core#25); They formed a virtual ring network all together.

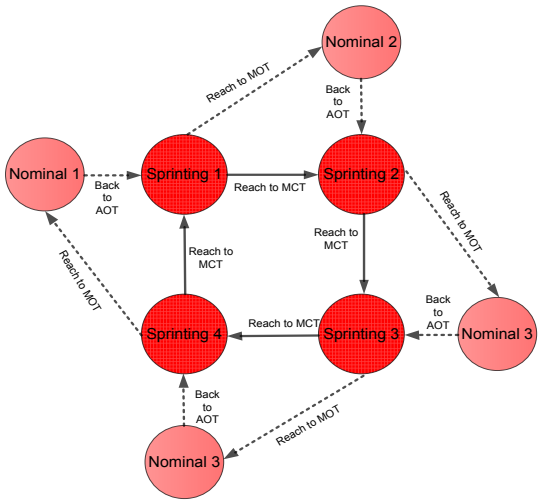


Figure 5. Shift sprinting state diagram

4.3 Application Migration Scheme

Based on the behaviors of running applications, if the temperature reaches a certain threshold, the application migration (i.e. shifting between different phases) will start. Two different upper-bound thresholds are defined: One called Maximum Core Temperature (MCT) and the other called Maximum Overall Temperature (MOT). When the temperature of each core reaches the MCT, the shift happens from the current sprinting to the next sprinting phase. On the other hand, when the overall temperature of the system reaches the MOT, the shift happens from the current sprinting to the next nominal phase. For a lower-bound threshold, Average Overall Temperature (AOT) is defined. When the overall temperature of the system goes back to the AOT, the shift happens from the current nominal to the sprinting phase. Shift sprinting state diagram is shown in Figure 5. In dark silicon age, the required free cores are guaranteed to be available for application migration. With this scheme, instead of waiting for the cores to cool-down after each sprinting phase, the intervals between cool-down periods of the cores are utilized by migrating the running applications to the dark cores and maximizing the sprinting timelines.

In order to lower the overhead of application migration process, shift sprinting is implemented based on a message passing interface called MMPI [12], in which application mapping is independent of application re-mapping. By changing the application mapping table, the application is remapped to another core. Then application state

information is transferred; hence, the migrated application can restore execution on a different core. In this case, the application migration contributes less communication overhead because application state information excludes application code.

Algorithm 1. Application migration algorithm

$\boldsymbol{MCT}$: maximum core temperature
$\boldsymbol{MOT}$: maximum overall temperature
$\boldsymbol{AOT}$: average overall temperature
$\boldsymbol{C}$: set of all the cores
$\boldsymbol{j}$: phase of the system (i. e. phases 1 to 4)
$\boldsymbol{D_j}$: set of the dark cores in j_{th} phase
$\boldsymbol{m_j}$: , manager core in j_{th} phase
$\boldsymbol{t_i}$: temperature of the core $c_i \in C$
$\boldsymbol{T}$: overall temperature of the system

set the value of MCT
initiate phase j
while true **do**
  set the values of MOT and AOT
  if j is in $sprinting$ phase **then**
    if $T < MOT$ **then**
      for $\forall c_i \in j$ **do**
        if $t_i > MCT$ **then**
          choose $c_d \in D_{j+1}$ with min$[t_d]$
          migrate the running application from c_i to c_d
        end
      end
      start $sprinting$ phase in $j+1$
    else // i.e. $T \geq MOT$
      for $\forall c_i \in j$ **do**
        migrate the running application from c_i to m_{j+1}
      end
      start $nominal$ phase in $j+1$
    end
  else // i.e. j is in $nominal$ phase
    if $T < AOT$ **then**
      while $\exists$ running application in m_j **do**
        choose multiple c_ds $\in D_j$ with min$[t_d]$s
        migrate the running application from m_j to chosen c_ds
      end
      start $sprinting$ phase in j
    end
  end
  set the next phase as j
end

4.4 Controlling Mechanism

A cost function is needed for CM in order to determine the best destination core for application migration. When the threshold in one sprinting phase reaches MCT, the running application will migrate to the next sprinting phase. The destination cores in the next phase are chosen based on the least current temperature order. The system stays in this loop (i.e. inner loop of Figure 5) until it reaches MOT threshold. For MOT threshold, all the running applications will migrate to the next phase manager until the overall temperature of the system goes back to AOT. In this case the running applications in the manager spread through the dark cores of that phase based on the optimal number of sprinted cores required for each application. Then the sprinting phase is started again.

Algorithm 1 shows shift sprinting application migration scheme.. It is assumed that only one application can be executed in a sprinted core at each time and the application itself knows the optimal number of required sprinted cores to provide maximal performance speedup. MCT is a static threshold relies on the core materials while both MOT and AOT are dynamic thresholds that depend highly on application behaviors. Note that obtaining the optimal values for these thresholds are beyond the scope of this paper.

5. EXPERIMENTAL RESULTS

Experiments are performed on a cycle-accurate many-core platform implemented in SystemC. A pruned version of an open source simulator for mesh-based NoCs called Noxim [13] is utilized as its communication architecture. For power and temperature simulations, power and thermal models taken from [14], [15] are integrated as libraries into the simulator. Some multi-threaded applications from the PARSEC [16] benchmark suite are used in the experiments. Maximum traffic injection rate and the average life-time of the system running in idle status are called λ_{full} and γ_{idle} respectively. Three different network sizes of 16 (no dark silicon), 36 (55% dark silicon), and 64 (75% dark silicon) cores are considered in the simulations. Comparisons are also made between shift sprinting (SS) and two state-of-the-arts architectures: computational sprinting (CS) [6] and NoC-sprinting (NS) [7].

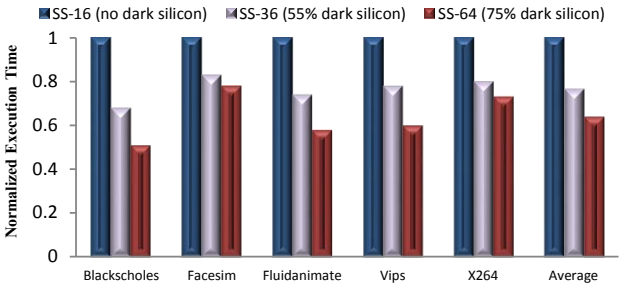


Figure 6. Execution time comparison between different sizes of SS

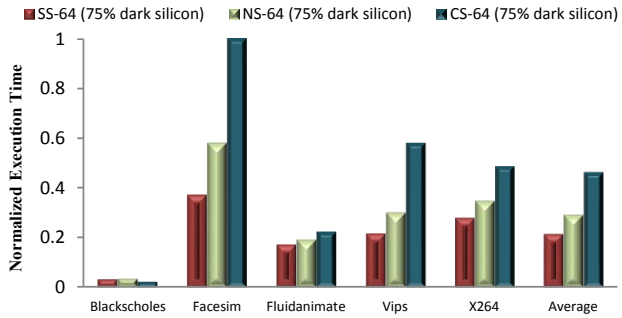


Figure 7. Execution time comparison between different architectures

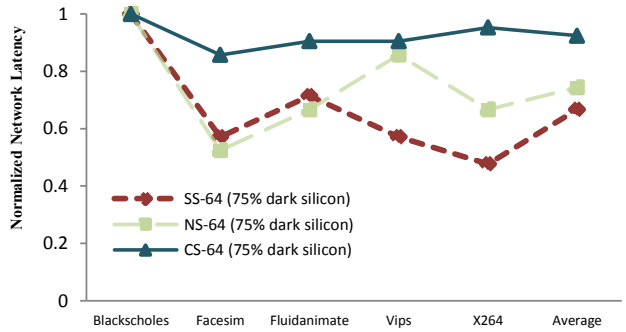


Figure 8. Network latency comparison between different architectures

5.1 Performance Evaluation

Figure 6 shows the normalized execution time of different workloads in SS. In comparison with 16-core NoC (no dark silicon), the average performance improvement of 64-core NoC (75% dark silicon) is 36% and that of 36-core NoC (55% dark silicon) is 24%. As a result, even with increasing of dark silicon in upcoming MCSoCs, the performance

of SS will still improve. This happens because SS utilize cool-down periods by activating dark cores.

Figure 7 demonstrates the normalized executing time of different workloads with different architectures for 64-core NoC (75% dark silicon). The results show that SS considerably reduces the execution time compared to other approaches. It achieves 55% and 25% average performance improvement compared to CS and NS respectively. The performance gain is based on the higher overall sprinting periods achieved by utilizing cool-down periods. As can be seen from Figure 7, all the architectures performed quite the same under "Blackscholes" workload that is a non-streaming financial analysis application. This is because "Blackscholes" achieves the optimal performance speedup in CS and hence leave no space for power-gating in NS and neither power-gating nor application migration in SS.

In addition, Figure 8 shows the normalized network latency of different workloads with different architectures for 64-core NoC (75% dark silicon) under the injection rate of $0.75\lambda_{full}$. It can be seen that SS reduces the communication latency for all the applications (28% in average) in comparison with CS and for most of the applications (11% in average) in comparison with NS. Overall, SS performs quite well in media processing applications (e.g. Vips and X264). On the other hand, performance degradation of SS in the animation workloads (e.g. Facesim and Fluidanimate) compared with NS is due to application migration overhead. Although by applying MMPI [12], the application migration contributes less communication overhead, still more attempts are required to minimize this overhead by finding optimal thresholds and presenting low-overhead migration mechanisms.

5.2 Power Consumption Measurement

Figure 9 displays the normalized network power consumption of different workloads with different architectures for 64-core NoC (75% dark silicon) under the injection rate of $0.75\lambda_{full}$. On average, SS saves 58% power compared to CS while consuming almost the same power as NS. This is due to the fact that both SS and NS can adopt network topology based on an intermediate number of sprinted cores and benefit from power-gating in dark cores (i.e. 75% of the cores) while CS fully actives the network and loses power-gating opportunities.

As another evaluation parameter, Figure 10 depicts the normalized Energy Delay Product (EDP) of different workloads with different architectures for 64-core NoC (75% dark silicon). It can be seen that even with the overhead of application migration approach, the average EDP of SS in the media processing applications (e.g. Vips and X264) is less than the other architectures that make it a promising architecture for future MCSoC mobile devices.

5.3 Thermal Analysis

Figure 11 demonstrates the thermal analysis of SS under X264 workload for 36-core NoC (55% dark silicon). The peak temperature is 322.8K and the average temperature of the system is 298.9K, 304.3K, and 312.5 after one, two, and three consecutive sprinting respectively. Since SS uses simultaneous techniques of core sprinting, application migration, and power-gating to distribute the heat across the chip, it can efficiently avoid hot-spots in the system.

Furthermore, Figure 12 displays the thermal analysis of different architectures under X264 workload for 36-core NoC (55% dark silicon) after four consecutive sprinting. As shown in Figure 12a, CS results in a hot-spot in the center of the chip. Moreover, since thermal-aware floor-planning of NS, tries to physically separate logical connected cores, heat is distributed to the corners of the chip as depicted in Figure 12b. Such floor-planning proposal has three disadvantages: First, it requires additional overheads at design stage; Second, it is highly application-specific and is not suitable for dynamic workloads; Third, it leads to performance degradation due to long-distance communications between physically separated cores. On the other hand, as shown in Figure 12c, SS outperforms the other two architectures to efficiently distribute the heat across the chip. First, it does not add any temperature-aware floor-planning overhead to the system; Second, it does not rely on specific running applications to avoid hot-sports; Third, there is no need to change the physical positions of the cores.

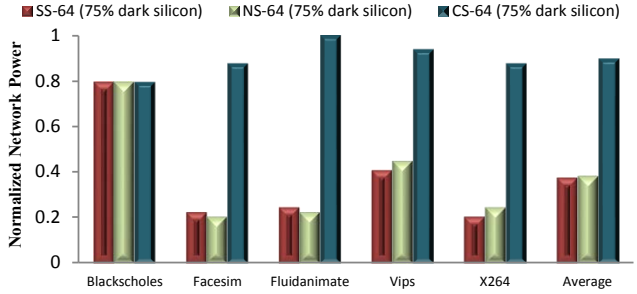


Figure 9. Network power comparison between different architectures

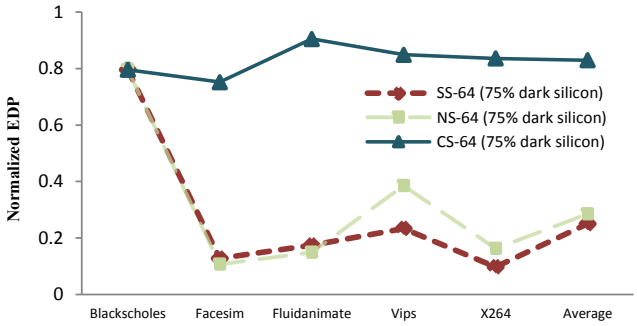


Figure 10. EDP comparison between different architectures

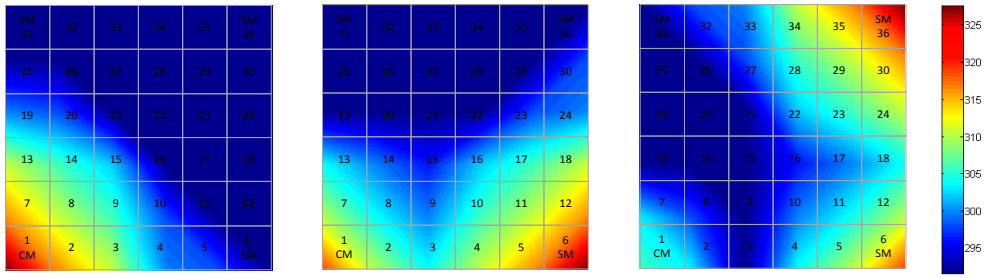
5.4 Reliability Assessment

The central hot-spot in CS and the angular hot-spots in NS greatly increase the permanent failure probability of those highly active cores due to frequent phase-change of the core internal materials. Figure 13 demonstrates the failure probability of different architectures over time under X264 workload for 36-core NoC (55% dark silicon) under the injection rate of $0.5\lambda_{full}$. It is assumed that failure of a single core leads to the whole system failure. It can be seen that fair core unitization in SS results in not only efficiently distributing the heat across the chip, but also increasing the reliability of the system. In other words, in SS the cores are aging almost evenly. On the contrary, there are some central aged cores in CS as well as some cornered aged ones in NS through time while the others are still young. The aged cores are increasingly subjected to failure than the young ones. This fact makes the requirement of fault-tolerant mechanisms inevitable in CS and NS.

6. CONCLUSION

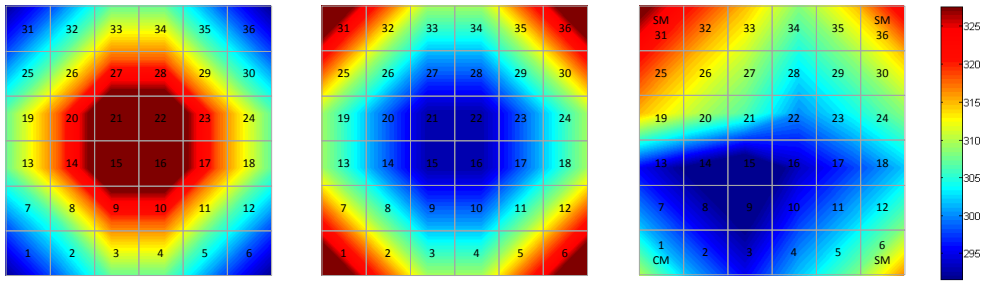
Among all the challenges the mobile device industry faces, keeping components cool is the most important, since overheating causes significant reductions in the operating life of a device. Moreover, QoS of real-time streaming applications will become more and more important for future generations of mobile devices.

In this paper, a novel fine-grained temperature-aware NoC-based MCSoC architecture, called *shift sprinting* was introduced in order to handle high-performance QoS-aware mobile demands by reliably utilizing dark silicon. Simulation results reported meaningful gain in performance and reliability of the system compared to state-of-the-art works.



(a) (b) (c)

Figure 11. Thermal distribution in SS-36 (55% dark silicon) under X264 workload after (a) one (b) two (c) three consecutive sprinting



(a) (b) (c)

Figure 12. Thermal distribution comparison between different architectures under X264 workload (a) CS-36 (b) NS-36 (c) SS-36 (55% dark silicon)

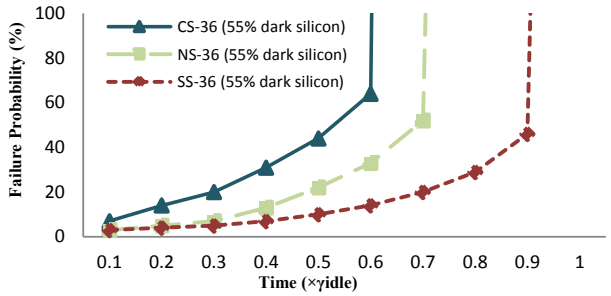


Figure 13. Reliability comparison between different architectures

7. DEDICATION

This work is dedicated to my wonderful grandfather, Mr. Akbar Hemami, who passed away recently. I did not get a chance to see him again before his death nor to participate in his funeral; Because I was more than 7000 miles away working on this paper...

8. REFERENCES

[1] ITRS. International Technology Roadmap for Semiconductors, 2013 edition.

[2] R. H. Dennard, F. H. Gaensslen, H. N. Yu, V. L. Rideout, E. Bassous, and A. R. Leblanc, "Design of ion-implanted MOSFET's with very small physical dimensions," In *IEEE Journal of Solid-State Circuits*, Vol. 9, pp. 256-268, 1974.

[3] N. Goulding-Hotta, J. Sampson, Q. Zheng, V. Bhatt, J. Auricchio, S. Swanson, and M. B. Taylor, "GreenDroid: an architecture for the dark silicon age," In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 100-105, 2012.

[4] M. B. Taylor, "Is dark silicon useful? harnessing the four horsemen of the coming dark silicon apocalypse," In *ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1131-1136, 2012.

[5] J. Henkel, H. Khdr, S. Pagani, and M. Shafique, "New trends in dark silicon," In *ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp 1-6, 2015.

[6] A. Raghavan, Y. Luo, A. Chandawalla, M. Papaefthymiou, K. P. Pipe, T. F. Wenisch, and M. M. K. Martin, "Computational sprinting," In *IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 1-12, 2012.

[7] J. Zhan, Y. Xie, and G. Sun,"NoC-sprinting: interconnect for fine-grained sprinting in the dark silicon era," In *ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1-6, 2014.

[8] W. Godycki, C. Torng, I. Bukreyev, A. Apsel, and C. Batten, C, "Enabling realistic fine-grain voltage scaling with reconfigurable power distribution networks," In *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 381-393, 2014.

[9] Chetan Sharma Consulting. [Online]. Availble: http://chetansharma.com/usmarketupdateq22015.htm

[10] A. Rezaei, M. Daneshtalab, D. Zhao, F. Safaei, X. Wang, and M. Ebrahimi, "Dynamic application mapping algorithm for wireless network-on-chip," In *Euromicro International Conference on Parallel, Distributed and Network-Based Computing (PDP)*, pp. 421-424, 2015.

[11] B. Goodarzi and H. Sarbazi-Azad, "Task migration in mesh NoCs over virtual point-to-point connections," In *Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pp. 463-469, 2011.

[12] F. Fu, S. Sun, X. Hu, J. Song, J. Wang, and M. Yu,"MMPI: a flexible and efficient multiprocessor message passing interface for NoC-based MPSoC," In *IEEE International SoC Conference (SoCC)*, pp. 359-362, 2010.

[13] "Noxim: network-on-chip simulator," [Online]. Available: http:// www.noxim.org/.

[14] L. Wang and K. Skadron "Dark vs. dim silicon and near-threshold computing extended results," Technical Report (UVA-CS-2013-01), Department of Computer Science, University of Virginia, 2013.

[15] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. Stan, "HotSpot: a compact thermal modeling methodology for early-stage VLSI design," In *IEEE Transaction on Very Large Scale Integration (VLSI) Systems*, Vol.14, Issue 5, pp. 501-513, 2006.

[16] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC benchmark suite: characterization and architectural implications," In *International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pp. 72-81, 2008.