On Trade-off Between Static and Dynamic Power Consumption in NoC Power Gating

Di Zhu†, Yunfan Li*, Lizhong Chen*
†Ming Hsieh Department of Electrical and Computer Engineering, University of Southern California, United States
*School of Electrical Engineering and Computer Science, Oregon State University, United States
Email: †dizhu@usc.edu, *{liyunf, chenliz}@oregonstate.edu

***Abstract*—Recent research has proposed to minimize network-on-chip (NoC) static power by proactively power-gating selected routers when not all the cores are active. However, as more routers are powered off, on-chip packets are forced to take detours more frequently, resulting in a higher hop count and increased dynamic power that may potentially offset the static power savings. This paper investigates such a trade-off between static and dynamic power in detail, and explores how overall NoC power consumption can be minimized through proactive power-gating. Three efficient and effective algorithms are proposed to reduce NoC static power, dynamic power, and overall power consumption, respectively. Evaluation results based on PARSEC benchmarks demonstrate the importance of the trade-off and show a substantial improvement in total NoC power savings of the proposed algorithms, compared with previous work that did not give full consideration to both static and dynamic power.**

I. INTRODUCTION

To meet the communication demands of parallel computing among the many cores in a processor, networks-on-chip (NoCs) have been proposed as the key communication component in the system architecture. However, compared with traditional buses, the relatively complex NoC architecture with routers and links can draw a substantial percentage of the chip's power (e.g., 28% in Intel Teraflop [8] and 19% in Scorpio [6]). Meanwhile, many-core processors often exhibit low core utilization (typically 15% to 55% [1]), with some cores sent to deep sleep states periodically, making the NoC power percentage even higher if it is not powered off proportionally to the core usage. In particular, a considerable percentage of the NoC power is contributed by the static power part, and this contribution will continue to grow as manufacturing technology scales further.

Recent research has been proposed which applies power gating techniques to NoC routers to reduce static power. A popular method to apply power gating is to turn on and off the NoC routers *reactively* ([4][5][11]), i.e., router states are determined by incoming traffic, and idle routers that have no forwarded or injected traffic are powered off. In these reactive, traffic-oriented strategies, router idle periods are inherently limited to what traffic patterns allow, and the lengths of the idle periods tend to be fragmented by intermittent packet arrivals, with only tens to hundreds of cycles in practice.

Another type of NoC power gating methods takes advantage of under-utilized cores and *proactively* puts selected routers into sleep, detouring any traffic that might need to go through these routers [13][15]. This approach allows idle and under-utilized routers to be powered off for longer periods of time, thus offering more savings in static power. However, detoured packets may increase the overall hop count and router activities. This leads to higher dynamic power that may potentially offset the static power savings. Merely turning off as many routers as possible does not necessarily leads to the maximum total power savings. Therefore, it is imperative to take into account the penalty of dynamic power consumption during power gating in order to achieve the goal of power minimization.

In this work, we explore the above power reduction opportunity in proactive NoC power gating, and present effective algorithms to improve total NoC power. First and foremost, we identify that there is a trade-off between static power and dynamic power in proactive power gating of on-chip routers, and conclude that the optimal solution with minimum overall power should be found at a balance point between static and dynamic power. We then present three effective power gating algorithms with the awareness of core-state and communication status: the first one to achieve the maximum static power savings, the second one to minimize dynamic power penalty, and the last one to locate the minimum overall NoC power solution. Evaluation based on PARSEC benchmarks demonstrates the importance of the trade-off as well as the advantage of the proposed algorithms over prior art. In particular, for the proposed algorithm that optimizes the overall NoC power consumption, 33.4% NoC power savings is achieved with only 3.5% latency overhead.

II. BACKGROUND

A. Power Consumption of On-Chip Networks

The power consumption of on-chip components includes dynamic power and static power. Figure 1 plots the power of a NoC router at 3GHz under different technologies and injection rates (flits per cycle), obtained by the DSENT NoC power model [14]. It can be seen that dynamic power increases linearly with the injection rates of routers whereas the static power is independent of router activity and strongly relates to manufacturing technology. As shown in Figure 1(b), the average static power percentage of NoC routers increases considerably from 45.37% in 45nm to 56.14% in 32nm when r=0.1. This urges researchers to find effective techniques to reduce not only the dynamic power but also the static power consumption of on-chip networks.

B. Power Gating Techniques of On-Chip Networks

Recent research has started to apply power gating to on-chip routers, which cuts off the supply voltage to a router when it is idle to save static power. Matsutani *et al.* propose a look-ahead technique for NoC power gating to reduce the runtime latency penalty [11]. Chen *et al.* equip a mesh-based NoC architecture with a ring-based bypass channel, which achieves high power saving with small latency penalty [4]. Das *et al.* use multiple

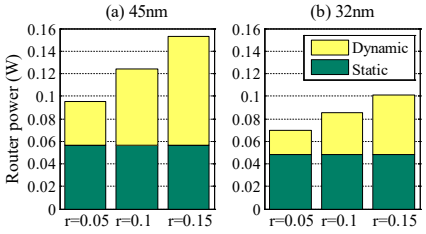

Figure 1. Dynamic and static power consumption in a NoC router.

narrow networks to achieve different levels of NoC power gating [5]. These works are all traffic-oriented power gating and do not exploit the long idle period of sleeping cores.

Proactive power gating utilizes core idleness by using a central control unit to collect traffic information and make decisions on which routers are power-gated. Core status usually changes at the time scale of milliseconds [10], which gives sufficient time for calculation and exchange of router status and routing information. Along this line, Samih *et al.* propose *Router Parking* on mesh topology, with implementation to support proactive power gating [13]. Yue *et al*. develop proactive power gating algorithms on the flattened butterfly with the awareness of core status [15]. These works, however, mainly target at minimizing static power with no or limited consideration of the dynamic power penalties brought by packet detours.

III. MOTIVATION

The NoC static power P_S is determined by the number of active routers,

$$P_S = \gamma \cdot |\boldsymbol{R}| \quad (1)$$

where γ is the static power consumption of a single router, and $\boldsymbol{R}$ is the set of active routers. Note that $\boldsymbol{R}$ includes routers that are directly connected to active cores (referred to as *anchor routers* hereinafter, e.g., Routers 2,4,9,11 in Figure 2), as well as additional powered-on routers to main the connectivity among all the active cores (e.g., Routers 3,6,10 in Figure 2). In order to achieve maximum static power savings, we can power off as many routers as possible with the condition that full connectivity of anchor routers is maintained. Unfortunately, doing this may potentially increase dynamic power consumption.

As mentioned in Section II.A, dynamic power consumption of router i is proportional to the router load l_i, which can be characterized as the number of packets processed by router i per unit time. With only the minimal set of $\boldsymbol{R}$ to keep connectivity, the paths between the anchor routers are limited, forcing packets between some router pairs to detour and be forwarded through more routers. This increase in hop count means that some routers have to process increased packets during the same time period, i.e., a higher load l_i, leading to the penalty in dynamic power.

Specifically, the total NoC dynamic power P_D is proportional to the summation of all router load $\sum_i l_i$. Notice that the load sum $\sum_i l_i$ is the total number of packets processed by all the routers per unit time. This is equivalent to the total number of hops traveled by all the packets during the unit time (e.g., a packet traveling 3 hops is essentially processed three times). Thus, if H denotes the overall packet hop count per unit time (referred to as *overall hop count* hereinafter), we have

$$\sum_i l_i = H = \sum_i \sum_j h_{ij} \cdot r_{ij} \quad (2)$$

where h_{ij} denotes the number of hops a packet needs to travel from anchor router i to j, which is determined by the currently active routers. r_{ij} denotes the communication rate, i.e., number of packets sent from i to j per unit time, collected by the aforesaid central controller in each execution epoch. The total NoC dynamic power P_D is therefore determined by

$$P_D = \rho \cdot H \quad (3)$$

where ρ is the dynamic power consumption of a single packet going through one router hop. Once a NoC design is given (i.e., router radix, link width, frequency, etc.), ρ can be considered as a fixed value at the granularity targeted in this paper.

In conclusion, there exists a trade-off between static and dynamic power consumption in proactive power gating, and the optimal solution to minimize the overall power must take into account both factors.

IV. PROBLEM STATEMENT

Based on the above analysis, we can formulate the proactive power gating problem as follows.

Given:

1) A mesh on-chip network consisting of $N = n^2$ routers (each router is connected to a core);
2) A set of anchor routers $\boldsymbol{R}_0$, each of which connects directly to an active processing core;
3) Communication rates between the active cores r_{ij}, where the i-th and j-th routers are both the anchor routers in $\boldsymbol{R}_0$;
4) Static power consumption per router γ, and the coefficient ρ for dynamic power consumption per packet per hop.

Find:

1) The minimum active router set $\boldsymbol{R}_S \supseteq \boldsymbol{R}_0$ that maintains the connectivity between the anchor routers.

As mentioned above, $\boldsymbol{R}_S$ provides the minimum static power consumption because it turns on the least possible number of routers, i.e., $|\boldsymbol{R}_S|$ is minimum, but with potentially high overall hop count and high dynamic power consumption.

2) The minimum active router set $\boldsymbol{R}_D \supseteq \boldsymbol{R}_0$ that minimizes hop count.

$\boldsymbol{R}_D$ denotes the active router set with a minimum number of routers, while ensuring all the packets travel through the shortest path (i.e., the smallest number of hops) between any two anchor routers to minimize the dynamic power consumption. In other words, we aim to find the minimum set of routers that provide shortest paths for all the packets. Turning on more routers than $|\boldsymbol{R}_D|$ does not further reduces the overall hop count H.

3) The active router set $\boldsymbol{R}_P \supseteq \boldsymbol{R}_0$ that minimizes the overall NoC power consumption.

Since the NoC static power is minimized when $|\boldsymbol{R}_S|$ routers are powered on and the dynamic power is minimized when $|\boldsymbol{R}_D|$ routers are powered on, the size of $\boldsymbol{R}_P$ for the minimum overall NoC power consumption satisfies $|\boldsymbol{R}_S| \leq |\boldsymbol{R}_P| \leq |\boldsymbol{R}_D|$.

Note that, in the selection of active routers for $\boldsymbol{R}_P$, it is important (and challenging) to take into account the communication rates r_{ij} between the active cores, as r_{ij} may greatly affect the overall hop count and thus dynamic power consumption.

V. PROPOSED SOLUTIONS

A. Minimal Active Router Set to Maintain Connectivity

The problem of finding $\boldsymbol{R}_S$ is equivalent to the well-known Rectilinear Steiner Minimum Tree (RSMT) problem. Recall that a Steiner minimum tree is a tree of minimum weight that contains all the terminal nodes (may include additional nodes), and RSMT is a special case where the nodes are connected only by vertical and horizontal lines. RSMT has been proved to be NP-complete [12]. In the NoC power gating problem, we need to find a RSMT, formed by all the routers in $\boldsymbol{R}_S$, to connect all the terminal (anchor) routers $\boldsymbol{R}_0$.

According to the Hanan's theorem [7], there exists a RSMT by starting from a minimum spanning tree (MST) of terminal

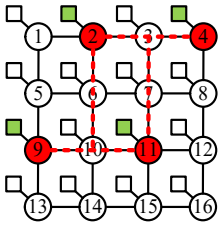

Figure 2. Mesh example.

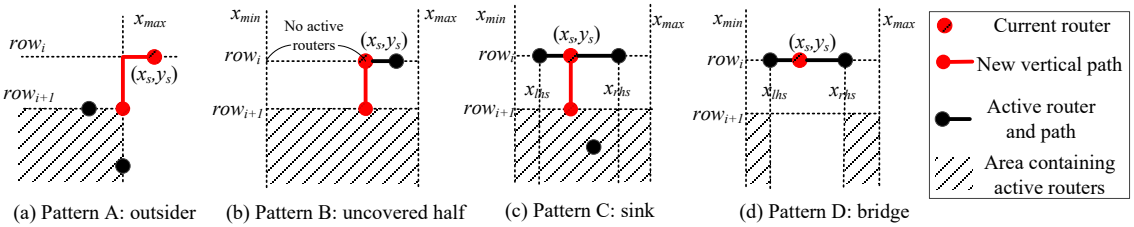

Figure 3. Four basic patterns for vertical path decisions from row_i to row_{i+1} in CAID.

nodes, and gradually adding Steiner points chosen from a candidate set $\boldsymbol{S}_{Cand}$ until the total length of the spanning tree cannot be reduced further. The $\boldsymbol{S}_{Cand}$ is the intersection points of all the horizontal and vertical lines drawn through the terminal points. In our case, $\boldsymbol{S}_{Cand}$ is the set of *intersection points* of all the horizontal and vertical lines drawn through the anchor routers in $\boldsymbol{R}_0$. For the example in Figure 2, $\boldsymbol{R}_0 = \{2,4,9,11\}$ and $\boldsymbol{S}_{Cand} = \{1,3,10,12\}$, e.g., "1" is the intersection point of the horizontal line drawn through "2" and "4" and the vertical line drawn through "9".

With the above analysis, we propose an algorithm named *Communication-Aware Iterative 1-Steiner* (CAIS). As shown in **Alg. 1** below, the basic procedure is as follows: CAIS starts with a minimum spanning tree (MST) of $\boldsymbol{R}_0$ denoted as $MST(\boldsymbol{R}_0)$, and adds the Steiner points one at a time from $\boldsymbol{S}_{Cand}$ until no Steiner points can further reduce the length of the current MST $L\big(MST(\boldsymbol{R})\big)$. An important aspect is how to choose the next Steiner point. In the proposed algorithm, at each iteration, CAIS selects a Steiner point that maximizes the reduction of the length of the current MST. The MST length reduction $\Delta L(s, \boldsymbol{R})$ of any Steiner point s is calculated by

$$\Delta L(s, \boldsymbol{R}) = L\big(MST(\boldsymbol{R})\big) - L\big(MST(\boldsymbol{R} \cup \{s\})\big). \quad (4)$$

In order to take the communication rates into consideration to minimize the overall hop count of $\boldsymbol{R}_S$, CAIS uses the overall hop count H defined in (2) as the tiebreaker. More precisely, whenever there is a tie in choosing the next Steiner point, CAIS selects the one that results in a smaller H.

Alg. 1 Communication-Aware Iterative 1-Steiner (CAIS)

Initialize $\boldsymbol{S}$, $MST(\boldsymbol{R}_0)$ //$\boldsymbol{S}$ is the Steiner point set
while $\boldsymbol{S}_{Cand} \neq \emptyset$ and $\Delta L(s, \boldsymbol{S} \cup \boldsymbol{R}_0) > 0$
  find all $s \in \boldsymbol{S}_{Cand}$ which maximizes $\Delta L(s, \boldsymbol{S} \cup \boldsymbol{R}_0)$
  if there are multiple solutions of s
    find s with minimum overall hop count H
  if $\Delta L(s, \boldsymbol{S} \cup \boldsymbol{R}_0) > 0$ **then** $\boldsymbol{S} = \boldsymbol{S} \cup \{s\}$ //add s to set $\boldsymbol{S}$
  Remove s from $\boldsymbol{S}_{Cand}$
return $\boldsymbol{R}_S$ is the set of all the routers on $MST(\boldsymbol{S} \cup \boldsymbol{R}_0)$

The overall complexity of the CAIS is $O(N^3)$, as there are $O(N)$ iterations in total, each taking $O(N^2)$ to locate the next Steiner point and update H. The final step of building $\boldsymbol{R}_S$ takes $O(N^2)$ to complete. Based on [9][12], it can be proven that CAIS has a worst-case 3/2 approximation ratio.

B. Minimum Active Router Set to Minimize Hop Count

Different from the above $\boldsymbol{R}_S$, the router set $\boldsymbol{R}_D$ allows all the packets to use the shortest paths (i.e., minimize the overall hop count) with as few active routers as possible. While keeping all the routers powered on apparently achieves the minimal hop count, there might be some unnecessarily powered-on routers. As shown in the example in Figure 2 where $\boldsymbol{R}_0 = \{2,4,9,11\}$, some routers are not receiving or forwarding any packets (e.g., $\{13,14,15,16\}$), and some other routers can be powered off but still allow packets to be re-routed through the minimum path (e.g., if we power off Routers 1 and 5, the packets from Router 2 to Router 9 can be routed through Routers 6 and 10). Therefore, the minimum router set $\boldsymbol{R}_D$ should exclude the unnecessary routers and maximize the "sharing" of routers among multiple minimum paths.

The problem of finding $\boldsymbol{R}_D$ has an exponential search space similar to the $\boldsymbol{R}_S$ problem. In order to develop an efficient algorithm for $\boldsymbol{R}_D$, we identify one crucial property, which is the memoryless property of the $\boldsymbol{R}_D$ solution on the mesh topology. To be specific, assume s is a router within the rectangle defined by router i and j (e.g., Router 6 in the rectangle defined by Router 4 and 9 in Figure 2). A shortest path $p_{ij}^{\min}$ can be derived by combining the shortest path $p_{is}^{\min}$ between i and s with $p_{sj}^{\min}$ between s and j, i.e., $p_{ij}^{\min} = p_{is}^{\min} \cup p_{sj}^{\min}$. For example, a minimum path between Routers 4 and 6 is $\{4,3,2,6\}$, and the one between Routers 6 and 9 is $\{6,10,9\}$. Combining these two paths gets a minimum path $\{4,3,2,6,10,9\}$ between Router 4 and 9.

Inspired by this memoryless property, we propose an algorithm *Constructing Along sIngle Dimension* (CAID), which progressively constructs the shortest paths between the anchor routers along one dimension (e.g., row-by-row from top to bottom). The basic idea is that, at iteration i to process row_i, all the shortest paths among the anchor routers from row_1 to row_{i-1} have already been constructed in previous iterations and remain unchanged. Iteration i only needs to construct the horizontal paths within row_i and the vertical paths from row_i to row_{i+1} based on the relative positions of the active routers on row_i to the bottom (explained next). Here the construction of a path means to turn on all the routers on that path. In this way, we reduce the solution search space to one row in each iteration.

Specifically, the CAID algorithm first makes decisions on the paths from the current row to the next row (vertical paths) and then connects all the routers in the current row (horizontal paths) in each iteration. In order to ensure minimum paths between any pair of routers, the CAID algorithm constructs the vertical paths for each router based on four basic patterns shown in Figure 3, where (x_s, y_s) is the current active router ($y_s = row_i$). x_{min} and x_{max} indicate the smallest and largest x coordinates among all the anchor routers located in the *remaining* rows from row_{i+1} to the last row. We elaborate the four patterns and their corresponding decisions in the following.

(Pattern A Outsider) When s is outside the left or right boundaries of the remaining routers (i.e., $x_{min} > x_s$ or $x_s > x_{max}$), CAID constructs an XY path from s to the corner router (x_{min}, row_{i+1}) or (x_{max}, row_{i+1}).

(Pattern B Uncovered Half) When $x_{min} < x_s < x_{max}$, and no routers in the current row are active on one side of s (i.e.,

$[x_{min}, x_s)$ or $(x_s, x_{max}]$), CAID constructs the path from s to (x_s, row_{i+1}), i.e., the router right below s.

(Pattern C Sink) When $x_{min} < x_s < x_{max}$ and there are active routers in the current row on both sides of s, if there exists an remaining anchor router from column x_{lhs} to x_{rhs} where $x_{lhs} = \max_x\{x < x_s\}$ and $x_{rhs} = \min_x\{x > x_s\}$ (x is the x coordinates of active routers on row_i), CAID constructs the path from s to (x_s, row_{i+1}).

(Pattern D Bridge) When $x_{min} < x_s < x_{max}$, and there are no remaining anchor routers from column x_{lhs} to x_{rhs}, CAID constructs no new vertical paths.

The pseudo code of the above CAID is shown below:

Alg. 2 Constructing Along sIngle Dimension (CAID)

Initialize $\boldsymbol{R}_D = \boldsymbol{R}_0$
sort the routers in $\boldsymbol{R}_0$ based on their row number $\{row_i\}$
foreach row_i in $\{row_i\}$
  if row_i is the last row with anchor routers
  //construct horizontal paths
    $\boldsymbol{R}_D = \boldsymbol{R}_D \cup \{s | x_{min}^{row_i} \le x_s \le x_{max}^{row_i}, y_s = row_i\}$
    break
  endif
  find x_{min} and x_{max} of the remaining rows
  foreach active router s in row_i
    if (Pattern A==true)
    //turn on all the routers from s to the corner router
    $\boldsymbol{R}_D = \boldsymbol{R}_D \cup XYpath(s, (x_m, row_{i+1}))$ [a]
    else if (Pattern B or C==true)
    // turn on all the routers from s to row_{i+1}
    $\boldsymbol{R}_D = \boldsymbol{R}_D \cup XYpath(s, (x_s, row_{i+1}))$
    else if (Pattern D==true)
    //do nothing
  endfor
  //construct horizontal paths
  $\boldsymbol{R}_D = \boldsymbol{R}_D \cup \{s | x_s \in [x_{min}^{row_i}, x_{max}^{row_i}], y_s = row_i\}$
return the router set $\boldsymbol{R}_D$

[a] Function $XYpath(a, b)$ returns the set of routers on the path following X-Y routing from a to b.

CAID has n iterations (n rows), each having n routers. For each router, x_{min}, x_{max}, and whether there are any routers from column x_{lhs} to x_{rhs} are calculated in $O(\log N)$ by maintaining a binary search tree. The step of adding horizontal paths for each row takes $O(n)$. Overall, CAID has a complexity of $O(n) \cdot \big(O(n) \cdot O(\log N) + O(n)\big) = O(N \log N)$.

C. Active Router Set to Minimize Overall NoC Power

With $\boldsymbol{R}_S$ achieves the minimal static power and $\boldsymbol{R}_D$ achieves the minimal dynamic power, the active router set $\boldsymbol{R}_P$ for achieving the minimal overall NoC power can be found by powering on additional routers on top of $\boldsymbol{R}_S$. We adopt a similar basic flow of CAIS to find $\boldsymbol{R}_P$, i.e., iteratively turn on routers until a satisfying result is achieved. However, choosing one router to power on in each iteration may make no difference in the hop count, as a new path often requires more than one router. Therefore, instead of adding one router a time, we propose the *Communication-Aware Iterative 1-Path* (CAIP) algorithm that chooses one path at each iteration and turns on all the routers on that path. In this way, we can explore the full range of static-dynamic power trade-off, from the starting point of $\boldsymbol{R}_S$ up to $\boldsymbol{R}_D$, as the overall hop count H can always be reduced if $\boldsymbol{R}_D$ is not reached (i.e., there exists two routers with no minimum path between them).

Similar to the CAIS algorithm, a criterion of selecting a new path is needed for the CAIP algorithm. This criterion should reflect not only the cost of turning on a path but also the reward of doing so. The cost equals the static power increase, whereas the reward should comprise of two parts, namely (i) the dynamic power savings due to the hop count decrease brought by the new path, and (ii) the potential of the newly powered-on routers on future new paths to reduce dynamic power in later iterations as part of other minimum paths. More precisely, some powered-off routers may belong to minimum paths of more than one router pairs, and the future reward of turning on other paths of these "shared" routers should also be considered. For example, using the numbering in Figure 2, assume no minimum path is available between Routers 2 and 11 or between Routers 9 and 11. For adding a minimum path {9,10,11} between Routers 9 and 11, its reward calculation should include its own contribution in hop count reduction as well as the potential to further reduce the hop count between Routers 2 and 11.

Based on the above analysis, we give the criterion of selecting the paths as follows. We first define the weighted hop count reduction ΔH_{ij} for each router pair, which equals the hop count decrease in the overall hop count H if we connect router i and j with a minimum path,

$$\Delta H_{ij} = \Big(|p_{ij}| - M(i,j)\Big) \cdot r_{ij} \tag{5}$$

where p_{ij} is the current path from router i to j, and $M(i,j)$ is the Manhattan distance (i.e., the length of a minimum path in mesh networks). The difference between p_{ij} and $M(i,j)$ is weighted by r_{ij} to reflect the actual decrease on H.

With ΔH_{ij} calculated, we define the reward function $g(s)$ of router s, i.e., the potential power savings of turning on a powered-off router s as

$$g(s) = \rho \cdot \sum\nolimits_{p_{ij}^{\min} \ni s} \Delta H_{ij} - \gamma, \forall s \notin \boldsymbol{R}_S \tag{6}$$

where γ is subtracted from the $g(s)$, indicating the cost of static power consumption incurred by powering on the router. The reward part uses the summation of ΔH_{ij} over all the router pairs i and j that has a minimum path via router s ($p_{ij}^{\min} \ni s$).

Finally, the gain function of turning on a minimum path $p_{ij}^{\min}(k)$ between router i and j is defined by

$$g\left(p_{ij}^{\min}(k)\right) = \sum\nolimits_{s \in p_{ij}^{\min}(k)} g(s) \tag{7}$$

$p_{ij}^{\min}(k)$ denotes the k-th minimum path between i and j as there could be more than one minimum path.

As shown in the pseudo code below, the CAIP algorithm first sorts all the anchor router pairs in $\boldsymbol{R}_S$ based on ΔH_{ij} in descending order. When processing each router pair i and j, a minimum path with the maximum gain $g\left(p_{ij}^{\min}(k)\right)$ based on (7) is selected. CAIP then powers on the path, and updates $g(s)$ by subtracting ΔH_{ij} from the gain functions of all the other powered-off routers on the minimum paths between i and j (because this gain ΔH_{ij} has been already utilized in this round). The algorithm continues for the rest of the router pairs until (i) $\forall \Delta H_{ij} = 0$, i.e., every anchor router pair (but not necessarily all the router pairs) has a minimum path between them, or (ii) $P(\boldsymbol{R}_P) \ge$

$P(\boldsymbol{R}_D)$, i.e., $\boldsymbol{R}_P$ has reached $\boldsymbol{R}_D$ where no more active router would reduce H further (all router pairs have minimum paths).

Alg. 3 Communication-Aware Iterative 1-Path (CAIP)

Initialize $\boldsymbol{R}_P = \boldsymbol{R}_S$
Sort the router pairs based on ΔH_{ij}
foreach ΔH_{ij} in the sorted list $\{\Delta H_{ij}\}$
  if $P(\boldsymbol{R}_P) \geq P(\boldsymbol{R}_D)$ (reaches the end of $\boldsymbol{R}_S$ to $\boldsymbol{R}_D$ range)
    return the router set $\boldsymbol{R}_P = \boldsymbol{R}_D$
  find $p_{ij}^{\min}(k)$ s.t. $g\left(p_{ij}^{\min}(k)\right) \geq g\left(p_{ij}^{\min}(l)\right), \forall l$
  $\boldsymbol{R}_P = \boldsymbol{R}_P \cup p_{ij}^{\min}(k)$ // turn on all the routers on the path
  calculate overall power $P(\boldsymbol{R}_P) = P^{sta}(\boldsymbol{R}_P) + P^{dyn}(\boldsymbol{R}_P)$
  if $P(\boldsymbol{R}_P) < P_{\min}$ // better than the current one so far
    $P_{\min} = P(\boldsymbol{R}_P)$, $\boldsymbol{R}_{P,\min} = \boldsymbol{R}_P$
  foreach router s on a minimum path between i and j
    $g(s) = g(s) - \Delta H_{ij}, \forall s \notin \boldsymbol{R}_P$
return the router set $\boldsymbol{R}_{P,\min}$

The complexity of this algorithm is $O(N^3)$, explained in the following. Assume R_i routers are powered on in the i-th iteration. Finding the path for each router pair takes $O(n^2)$, adding each router to $\boldsymbol{R}_P$ and updating power consumption (equivalent to H) takes $O(N^2)$, and updating $g(s)$ takes $O(n^2)$. Therefore, the overall complexity is $\sum_i R_i \cdot \left(O(N^2) + O(n^2)\right) = O(N^3)$, as there are N routers in total and each router cannot be powered on more than once.

VI. Evaluation

A. Evaluation Setup

We evaluate the proposed power gating schemes on an 8x8 mesh-based NoC, with the multi-threaded PARSEC benchmarks [2] running on the cycle-accurate full-system gem5 simulator [3]. We experiment with 8, 16, and 32 active cores, and these active cores are randomly selected from the 64 cores on the 8x8 mesh. The following results are the average of ten randomly generated active core sets unless otherwise specified. The NoC power consumption data are from the post-synthesized router circuits under the 15 nm technology node by North Carolina State University [17]. Table I lists the key parameters.

Deadlock-avoidance: Given the light communication rates in real applications and thus the low likelihood of deadlock, we adopt a deadlock recovery approach. If packet latency exceeds a timeout threshold (indicating possible deadlock), all routers are powered back on to re-engage the original deadlock-free NoC. As true deadlock is extremely rare, the threshold can be very large (e.g., 10k cycles) to a point that has negligible actual performance impact. We compare the following six schemes:

1) Baseline: no power gating applied (NoPG);
2) Two *Router Parking* schemes proposed in [13]: aggressive (RPA) and conservative (RPC). RPA powers off as many routers as possible to minimize static power, which has the similar goal as our CAIS; whereas RPC powers off a subset of routers to minimize hop count increase[1], which has the similar goal as our CAID.
3) The three proposed schemes: CAIS, CAID, and CAIP to find $\boldsymbol{R}_S$, $\boldsymbol{R}_D$, and $\boldsymbol{R}_P$ respectively (to increase clarity, we name the algorithms in a way that the last letter also matches the objectives of minimizing static, dynamic, and overall power).

TABLE I. KEY PARAMETERS IN EVALUATION.

Network topology	8x8 mesh
Link bandwidth	128 bits/cycle
Router	3-stage, 3GHz
Virtual channel	2 VCs/VN, 3 VNs, 5-flit for data, 1-flit for control
Private I/D L1$	32KB, 2-way, LRU, 1-cycle latency
Shared L2 per bank	256KB, 16-way, LRU, 6-cycle latency
Coherence protocol	Two-level MESI
Memory controllers	4, located one at each corner
Memory latency	128 cycles

B. Comparison of the Power Gating Schemes

Figure 4 shows the results of NoC power consumption and hop count of the six schemes under different numbers of active cores, averaged over the PARSEC benchmarks. For better readability, the hop count in the figure is averaged to each packet (i.e., $H/\sum_i \sum_j r_{ij}$) instead of H directly which would otherwise be a less intuitive value that depends on traffic intensity.

Among the five power gating algorithms, both RPA and CAIS aim at minimizing static power while maintaining full connectivity. Compared to RPA, the proposed CAIS achieves an average of 21.7% less static power consumption with 5.4% less dynamic power consumption on average, indicating CAIS not only reduces the number of active routers, but also considers the hop count between active cores to reduce dynamic power. In addition, the proposed CAID guarantees the same minimal hop count as NoPG, but achieves an average of 18% power savings compared to NoPG. In contrast, RPC solutions require 14.5% more routers to be powered on than CAID on average while incurring 3.4% hop count increase compared to NoPG. RPC also has 31.7% higher static power consumption compared to CAID.

Finally, compared to NoPG, the proposed CAIP achieves 33.4% overall NoC power consumption savings under 8 active cores, and 24.0%, 17.4% under 16 and 32 active cores (all out of 64 cores), respectively. As can be seen, CAIP may not have the minimal hop count or the minimum number of active routers, but it achieves the minimal overall NoC power among all the schemes. This verifies that the minimal overall power may be different from the minimal static power or the minimal dynamic power configurations.

C. Trade-off between Static Power and Dynamic Power

To further study the trade-off between static and dynamic power, we look at the results of one of the benchmarks, x264, in detail. Other PARSEC benchmarks exhibit similar trends, although the specific values are not exactly the same. Figure 5 plots the changes of static power, dynamic power, and overall NoC power consumption as the number of active router increases, for a randomly generated set of 16 active cores shown in Figure 6. The static power is linearly proportional to the number of active routers. The solution generated by CAIS corresponds to the leftmost case, in which the dynamic power accounts for as high as 79.2% of the overall NoC power due to the large hop count caused by detour. As new minimum paths are added, the dynamic power consumption reduces because of the smaller hop count between the active cores. The dynamic power gradually gets close to a minimum as the active router set gets close to CAID. The overall NoC power consumption is minimized at 36 active routers with 0.45W, of which 57.7% comes from dynamic power and 42.3% comes from static power.

[1] The paper [13] claims RPC minimizes latency overhead, which is equivalent to minimizing overall hop count per unit time in their paper setup.

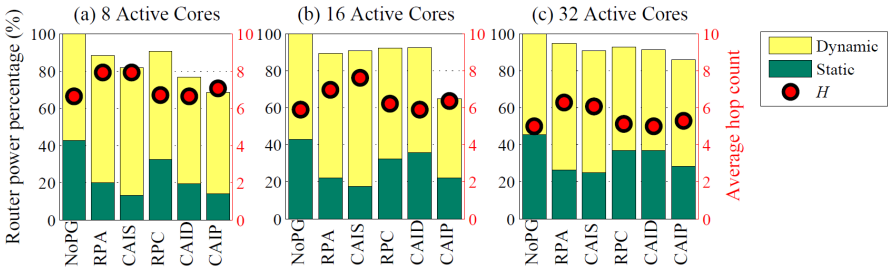

Figure 4. Comparison of different algorithms for PARSEC benchmarks.

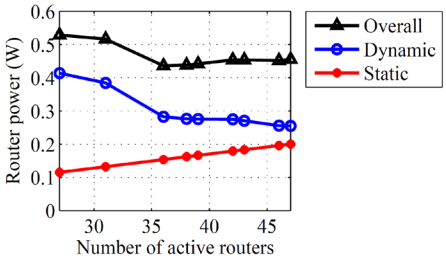

Figure 5. Static-dynamic power trade-off curve.

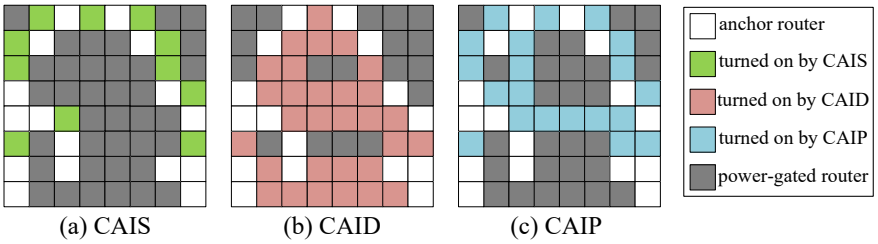

Figure 6. x264 result with 16 active cores.

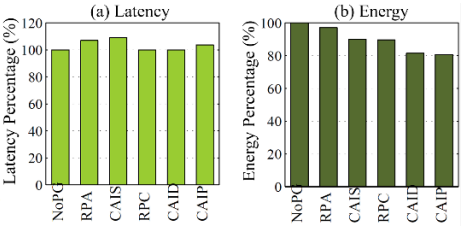

Figure 7. Latency and energy comparisons.

Figure 6 depicts the solutions generated by the three proposed algorithms for the above example. The anchor routers are shown in white blocks. The CAIS solution in Figure 6(a) needs to power on 11 routers (in green blocks) in addition to the anchor routers to ensure full connectivity; whereas CAID turns on 31 routers (in red blocks) to minimize hop count. Finally, the CAIP algorithm achieves the lowest power by turning on 20 routers (in blue blocks), or 9 more routers on top of CAIS. The newly added paths on top of CAIS are highlighted in red lines.

D. Impact on Latency and NoC Energy

We also investigate the impact of the proposed power gating schemes on performance and energy. Figure 7(a) and (b) compare the latency and NoC energy averaged over the PARSEC benchmarks under the six schemes. RPA and CAIS take longer with 6.9% and 8.0% latency overhead, respectively, but CAIS is able to save 9.9% NoC energy compared to NoPG, which is 7.0% more than that of RPA. For RPC and CAID, they have negligible performance overhead because of their minimal impact on hop count, and they save 10.4% and 19.5% NoC energy compared to NoPG, respectively. This indicates that CAID can be adopted to save NoC energy when high performance NoC is required during runtime. Lastly, among these six schemes, the minimal power solution CAIP achieves 25.8% NoC energy saving with only 3.5% latency overhead compared to NoPG.

E. Discussion on Implementation

Similar to RPA and RPC [13], the proposed scheme can be implemented in either hardware or software. For the hardware approach, a dedicated component is implemented to execute the algorithm once every epoch. For the software approach, the algorithm is executed on a processing core once every epoch. Sensitivity study shows that an epoch length of 50k cycles can reap most of the power-saving benefits while incurring negligible power overhead for both hardware and software implementations. However, software avoids the complexity of additional components, thus being a better implementation overall. We adopt the software approach for all the evaluated schemes, and the power overhead of running the algorithms has been accounted for in the total power.

VII. Conclusion

This paper identifies and explores the important trade-off between static and dynamic power in proactive NoC power gating schemes. We present three efficient algorithms to find the active router set that minimize the static, dynamic, and overall NoC power, respectively. Simulation results demonstrate the advantage of the proposed algorithms over prior art.

Acknowledgements: This research was supported, in part, by NSF grants #1619456, #1619472 and #1750047.

References

[1] L. A. Barroso, and U. Hölzle, "*The case for energy-proportional computing,*" Computer, no. 12, pp. 33-37, 2007.
[2] C. Bienia, S. Kumar, J. Singh, and K. Li, "*PARSEC 2.0: A new benchmark suite for chip-multiprocessors.*" Workshop on Modeling, Benchmarking and Simulation, 2009.
[3] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A Saidi, A. Basu, J Hestness, D. R. Hower, T. Krishna, S. Sardashti, and R. Sen. "*The gem5 simulator,*", in ACM SIGARCH Computer Architecture News, 2011.
[4] L. Chen, and T. M. Pinkston, "*NoRD: Node-router de-coupling for effective power-gating of on-chip routers,*" in MICRO, 2012.
[5] R. Das, S. Narayanasamy, S. K. Satpathy, and R. G. Dreslinski, "*Catnap: Energy Proportional Multiple Network-on-Chip,*" in ACM SIGARCH Computer Architecture News, 2013.
[6] B. Daya, C.-H. Chen, S. Subramanian, W.-C. Kwon, S. Park, *et al.*, "*SCORPIO: A 36-core research chip demonstrating snoopy coherence on a scalable mesh NoC with in-network ordering,*" in ISCA, 2014.
[7] M. Hanan. "*On Steiner's problem with rectilinear distance,*" SIAM Journal on Applied Mathematics, vol. 14, no. 2, pp. 255-265, 1966.
[8] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, "*A 5-GHz mesh interconnect for a Teraflops processor,*" in MICRO 2007,
[9] A. B. Kahng, and G. Robins, "*A new class of iterative Steiner tree heuristics with good performance,*" in IEEE TCAD, 1992.
[10] N. Madan, A. Buyuktosunoglu, P. Bose, and M. Annavaram, "*A case for guarded power gating for multi-core processors,*" in HPCA 2011.
[11] H. Matsutani, M. Koibuchi, D. Ikebuchi, *et al.*, "*Ultra fine-grained run-time power gating of on-chip routers for CMPs,*" in NOCS, 2010.
[12] G. Robins, and A. Zelikovsky. "*Minimum steiner tree construction,*" The Handbook of Algorithms for VLSI Phys. Design Automation, 2009.
[13] A. Samih, R. Wang, A. Krishna, C. Maciocco, C.-M. Tai, and Y. Solihin, "*Energy-efficient interconnect via router parking,*" in HPCA, 2013.
[14] C. Sun, *et al.*, "*DSENT-A tool connecting emerging photonics with electronics for opto-electronic network-on-chip modeling,*" NOCS 2012.
[15] S. Yue, L. Chen, D. Zhu, T. M. Pinkston, and M. Pedram, "*Smart butterfly: reducing static power dissipation of network-on-chip with core-state-awareness,*" in ISLPED, 2014.
[16] "*FreePDK15: Contents,*" www.eda.ncsu.edu/wiki/FreePDK15.