Smart Butterfly: Reducing Static Power Dissipation of Network-on-Chip with Core-State-Awareness

Siyu Yue, Lizhong Chen, Di Zhu, Timothy M. Pinkston, and Massoud Pedram
Ming Hsieh Department of Electrical Engineering
University of Southern California
Email: {siyuyue, lizhongc, dizhu, tpink, pedram}@usc.edu

ABSTRACT

While power gating is a promising technique to reduce the static power consumption of network-on-chip (NoC), its effectiveness is often hindered by the requirement of maintaining network connectivity and the limited knowledge of traffic behaviors. In this paper, we present *Smart Butterfly*, a core-state-aware NoC power-gating scheme based on flattened butterfly that utilizes the active/sleep state information of processing cores to improve power-gating effectiveness. Smart Butterfly exploits the rich connectivity of the flattened butterfly topology to allow more on-chip routers to be power-gated when their attached cores are asleep. We present two heuristic algorithms to determine the set of routers to be turned on to maintain connectivity and allow tradeoff between power consumption and average packet latency. Simulation results show an average of 42.85% and 60.48% power reduction of Smart Butterfly over prior art on 4x4 and 8x8 networks, respectively.

Keywords

Network-on-chip, power-gating, flattened butterfly.

1. INTRODUCTION

Networks-on-chip (NoCs) have been proposed as a key component in many-core systems such as chip-multiprocessors (CMPs) and multiprocessor system-on-chips (MPSoCs). Compared with traditional bus structures, the relatively complex NoCs with routers and links can draw a substantial percentage of chip power [2][3][4][10]. This is particularly true as cores are often underutilized and therefore put into low-power sleep states (10–50% average utilization [8]).

An effective approach to reduce NoC power consumption is to apply power gating techniques. Most state-of-the-art NoC power gating schemes are traffic-oriented, in which routers are power gated when there is no traffic that needs to go through the routers [9]. However, the traffic-oriented power gating strategies use only traffic information and are unable to take full advantage of inactive cores. For example, even if a core is in sleep state and, thus, has no incoming or outgoing packets, its attached router cannot stay power-gated for long. This is because the router must be awoken intermittently to forward the passing packets to support communication of other active cores. In typical applications, the length of router idle periods is in the order of tens to hundreds of cycles – short enough to cause frequent wakeups and the associated energy overhead.

A new approach, which we refer to as core-state-aware power gating, aims to save more static power by enhancing NoCs with information of core states to make better power gating decisions. To be specific, some routers that are attached to the sleeping cores stay power-gated until the cores become active. Any traffic that would go through the sleeping routers is detoured. As the sleep periods of cores are in the order of several milliseconds [8], this approach allows routers to be turned off for a much longer time compared to the traffic-oriented approach. Note that not all of the routers attached to sleeping cores can be turned off as the network must maintain full connectivity of the active cores. In some cases, it may even need to turn on additional routers than the minimally required in order to reduce detoured traffic, and therefore reduce packet latency.

The closest example of core-state-aware power gating to-date is the Router Parking technique, which provides core power state information in power-gating mesh networks [10]. However, conventional mesh-based NoC topologies have limited ability in utilizing core status to reduce NoC power, as many routers that are attached to the sleeping cores must be turned on to provide full connectivity and reduce packet latency to an acceptable range.

The inherent topological limitation of mesh networks prompts us to look at other topologies that have richer connectivity. One such topology is high-radix networks that have express channels added to tile-based NoCs [5][6]. In particular, the flattened butterfly topology [5], uses express channels as shortcuts to connect directly the non-neighboring tiles on the same row or column, thus bypassing intermediate routers and accelerating packet transfer. Figure 1(b) shows an example of a 4x4 flattened butterfly network, in comparison to a 4x4 mesh network in Figure 1(a). Due to its superior connectivity, flattened butterfly is very suitable for core-state-aware power gating, as the number of routers that need to be powered on to guarantee connectivity is much smaller than in meshes (more details in Section 2.2).

To this end, we propose *Smart Butterfly*, a novel NoC power gating scheme that exploits the potential of power gating in flattened butterfly NoCs, and utilizes core-state-awareness to increase power-saving effectiveness. Specifically, we first prove the minimal number of routers that need to be powered on in flattened butterflies to ensure full connectivity of a given set of active cores. We then reduce packet latency without increasing much power overhead by selectively turning on additional routers. Two heuristic algorithms are proposed to solve the problem. We show that the two algorithms are able to achieve near-optimal results. With these algorithms, Smart Butterfly is able to achieve a wide range of power-latency trade-offs by varying the number of *ON* routers.

2. PRELIMINARIES

2.1 NoC Power Gating Techniques

Power gating is an effective technique to reduce static power consumption, especially for components with sufficiently long idle periods. Recent research has started to apply power gating to on-chip routers. Matsutani, *et al.*, propose look-ahead technique to

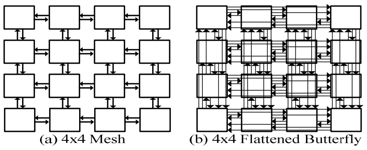

Figure 1. On-chip network topologies.

ISLPED'14, August 11–13, 2014, La Jolla, CA, USA.

http://dx.doi.org/10.1145/2627369.2627663

reduce run-time power consumption and wake-up latency [9]. Chen, *et al.*, present a mesh-based NoC architecture with a bypass channel which allows more sleeping routers and smaller latency penalty [2]. In addition, a Clos NoC based power-gating scheme [3] and a multiple network based scheme [4] have been proposed to increase power-gating opportunity. However, these works belong to traffic-oriented power gating approaches and cannot exploit the long idle period of sleeping cores. Samih, *et al.*, propose a novel NoC power gating technique, namely *Router Parking*, which shuts off selected routers attached to sleeping cores to achieve low NoC power consumption [10]. However, as discussed in detail later, the power efficiency of Router Parking deployment is largely restrained by the underlying mesh topology, and the adopted algorithms suffer unstable results due to randomness and lack the flexibility to trade-off between power and performance.

2.2 Connectivity Analysis

A key challenge to enable core-state-aware power gating is to ensure the connectivity of all active cores while minimizing the set of routers that needs to be powered on. Although the Router Parking work shows that the mesh is a viable candidate to allow some routers to be powered-off without disconnecting cores, flattened butterfly networks require much fewer powered-on routers due to their rich connectivity.

We first identify the minimal number of routers that need to be powered on additionally in flattened butterfly to ensure full connectivity of a given set of active cores.

Theorem: In a flattened butterfly network, in which the set of routers attached to the active cores form K connected components, full connectivity of all K components can be maintained only if a minimum of $(K-1)$ additional routers are powered on.

Proof sketch: The proof consists of the following two parts:

1) Full connectivity => At least $(K-1)$ additional routers.

In a flattened butterfly, all active routers in a row (or in a column) are already connected and are in the same component. For a sleeping router on the i-th row and j-th column, it can merge at most two components if turned on, one comprised of all the active routers on the i-th row and the other formed by all the active routers on the j-th column. This means that turning on one sleeping router can reduce the number of components by no more than one. Therefore, by induction, at least $(K-1)$ additional routers exist in powered-on state to connect K components and hence maintain full connectivity of all active cores.

2) The boundary case of $(K-1)$ additional routers is achievable.

We start from connecting two components. Assume A and B are two components. There exists at least one sleeping router which will connect A and B if turned on. We can turn on this router to merge A and B. Repeat this step $(K-1)$ times and we can connect all K components with $(K-1)$ additional routers. ■

While this theorem states that a selective set of $(K-1)$ additional routers is needed, there are different ways of choosing these routers, leading to different paths and varying packet latency among cores. Moreover, it may be beneficial to turn on extra routers besides the $(K-1)$ routers to further reduce average packet latency. Therefore, we focus on this more complex but more important problem and present two efficient algorithms to solve it.

3. PROBLEM STATEMENT

Before presenting the problem formulation, we first describe the network topology as well as the packet latency model used.

An n by m flattened butterfly network is defined as a grid with n rows and m columns. There is a core with a router attached to it on each grid point, making $N \triangleq n \times m$ cores (and routers) in total. All the routers on the same row or column are directly connected by a physical link between them.

Each core in the network can be either active or in sleep state. Let $c_i = 1$ denote core i is active and $c_i = 0$ otherwise. The set $C \triangleq \{i | c_i = 1\}$ is the set of all active cores in the network. Similarly, let $s_i = 1$ denote router i (which is attached to core i) is active and $s_i = 0$ otherwise, and $S \triangleq \{i | s_i = 1\}$ is the set of all active routers in the network. As a router cannot be put to sleep state if the corresponding core is in the active state, the following constraint exists,

$$s_i \geq c_i, \forall i \in \{1, \ldots, N\} \tag{1}$$

For each pair of active cores (i, j), let $r_{i,j}$ denote the communication load rate, and $d_{i,j}$ denote the communication latency between them. Assuming minimal routing which forwards packets on the shortest paths, we can compute the communication latency $d_{i,j}$ by

$$d_{i,j}(S) = \left(H_{i,j}(S) + 1\right) \cdot (T_R + t_c) + L_{i,j}(S) \cdot T_L + T_S \tag{2}$$

where $H_{i,j}(S)$ and $L_{i,j}(S)$ are the number of hops and the link length (in terms of how many unit lengths), respectively, on the shortest path between router i and j with given active router set S. T_R is the router pipeline latency, typically 2-4 cycles. t_c is the per hop contention latency. T_L is the unit length link latency, typically 1 cycle. T_S is the serialization latency, which is the quotient of packet size and link bandwidth.

The goal of the proposed Smart Butterfly scheme is to determine the best active router set S based on the knowledge of current active cores C and communication rates $r_{i,j}$. The objective is to minimize the average packet latency APL, calculated by

$$APL = \frac{\sum_{i,j \in C} d_{i,j}(S) \cdot r_{i,j}}{\sum_{i,j \in C} r_{i,j}} \tag{3}$$

subject to the maximum number of active routers S_{max}, i.e.,

$$|S| \leq S_{max} \tag{4}$$

It is impractical to enumerate all the possible solutions of the above problem when N is large. Therefore we propose two efficient heuristic algorithms, both having near-optimal performance and polynomial-time complexity.

4. PROPOSED ALGORITHMS

4.1 Exact Cost-Based Approach

The first algorithm is an exact cost-based approach, which starts from the state that only the routers connected to active cores are ON, and then turns on other routers one by one as needed. At each step when we determine which router to turn on, and choose the one that minimizes APL. The pseudo code is given below:

```
Algorithm Input: $n, m, C, r_{i,j}, S_{max}$
Initialize $activeS = C,\ sleepS = \{1, \ldots, N\} - activeS$
For $k$ from $|C| + 1$ to $S_{max}$        // Turn on routers one by one
  $minAPL = \infty$
  For $s$ in $sleepS$
    $activeS = activeS \cup \{s\}$
    Compute $APL$ using equation (3)
    If $APL < minAPL$
      $minAPL = APL,\ mins = s$
    $activeS = activeS / \{s\}$
  $activeS = activeS \cup \{mins\}$
  $sleepS = sleepS - \{mins\}$
Return
```

We assume t_c is a small fixed value to compute APL in designing the heuristics. It is also worth mentioning that we define $d_{i,j}$ to be a finite large number (e.g., 10^4) instead of ∞ in the algorithm if router i and j are not connected to each other yet.

Computing APL in the algorithm involves solving an all-pairs shortest path problem. Our implementation has a runtime of

$O(N^4)$ by bookkeeping $d_{i,j}(activeS)$. When router s is added to $currentS$, we can update $d_{i,j}(activeS)$ in $O(N^2)$ time.

4.2 Merit Value-Based Approach

In case $O(N^4)$ time complexity is still not fast enough for online implementation, we propose a faster algorithm, namely the merit value-based algorithm with $O(N^2)$ time complexity. Similar to the exact cost-based approach, the merit value-based approach turns on routers one by one. At each step of deciding which router to turn on, we first consider the routers that can connect two components, with the help of a disjoint set. If there is a tie (i.e., either multiple or no routers that can connect two components), we use a pre-computed merit value associated with each router as tie-breakers. The merit value of a router serves as a rough approximation of the reward of turning on that router. The merit value of router s is computed as the sum of communication rate $r_{i,j}$ where router i and router j are not directly connected to each other but are both directly connected to router s (so that they become connected if router s is turned on). When a sleeping router is turned on, the merit values of other routers are updated accordingly.

The pseudo code is given below:

```
Algorithm Input: $n, m, C, r_{i,j}, S_{max}$
Initialize $activeS = C,\ sleepS = \{1, \dots, N\} - activeS$
Initialize $merit = \mathbf{0}$
Initialize disjoint-set $DS = \{Row_1, \dots, Row_n\} \cup \{Col_1, \dots, Col_m\}$
For $c_1$ in $C$                         // Compute merit values
  For $c_2$ in $C$
    If $c_1$ and $c_2$ are not on the same row or column
      $s_1$ =router at $c_1.row$ and $c_2.column$
      $s_2$ =router at $c_2.row$ and $c_1.column$
      $merit(s_1) += r_{c1,c2}$, $merit(s_2) += r_{c1,c2}$
For $s$ in $activeS$                     // Update disjoint-set
  $DS.union(s.row, s.column)$
For $k$ from $|C| + 1$ to $S_{max}$      // Turn on routers one by one
  $maxs = -1$, $maxMerit = -\infty$, $connected = true$
  For $s$ in $sleepS$
    If $DS.find(s.row, s.column) = false$
      // Router $s$ connects two components
      If $connected$
        $maxs = s$,$maxMerit = merit(s)$,$connected = false$
      Else If $merit(s) > maxMerit$
        $maxs = s$, $maxMerit = merit(s)$
    Else If $connected$ And $merit(s) > maxMerit$
      $maxs = s$, $maxMerit = merit(s)$
  $DS.union(maxs.row, maxs.col)$
  $activeS = activeS \cup \{maxs\}$
  $sleepS = sleepS - \{maxs\}$
  For $s$ in $sleepS$                    // Update merit values
    If $s$ and $maxs$ are not on the same row or column
      $c_1$ =core at $s.row$ and $maxs.column$
      $c_2$ =core at $maxs.row$ and $s.column$
      $merit(s) -= r_{c1,c2}$
Return
```

We use an array-based disjoint-set implementation whose *find* operation has $O(1)$ time complexity and *union* operation has $O(n+m)$ time complexity. As computing and updating merit values take $O(N^2)$ time, the overall time complexity is $O(N^2)$, which is much smaller than that of the exact cost-based approach.

5. SIMULATION RESULTS

5.1 Simulation Setup

In the simulation, the proposed Smart Butterfly is evaluated on both 4x4 and 8x8 flattened butterfly (FB) networks with real application traces including four MPSoC traces (namely mms2, mpeg4, toybox, and vopd_t) and eight CMP traces (referred to as spec1~4 for 4x4 network and spec5~8 for 8x8 network). The MPSoC traces are collected from 12 to 16-core real applications. For 4x4 networks, the application traces are concentrated onto 3 to 4-core traces to form the active core set. The CMP traces are synthesized based on the memory and cache access traffic from a subset of SPEC benchmarks. Cores of all the test traces are randomly mapped to the NoC tiles.

We compare the following eight schemes, including both aggressive and conservative algorithms proposed in the Router Parking work [10] (these algorithms can be applied to FB as well):

1. Mesh_BB: A branch and bound algorithm on mesh network that minimizes APL at given maximum number of ON routers
2. Mesh_RPA: Router Parking – Aggressive on mesh network
3. Mesh_RPC: Router Parking – Conservative on mesh network
4. FB_BB: A branch and bound algorithm on FB that minimizes APL at given maximum number of ON routers
5. FB_RPA: Router Parking – Aggressive on FB
6. FB_RPC: Router Parking – Conservative on FB
7. FB_EC: The proposed exact cost-based approach on FB
8. FB_MV: The proposed merit value-based approach on FB

The network configurations in the simulation are listed in Table 1. Based on a previous study [7], the on-chip traffic is composed of approximately 80% short packets and 20% long packets. For fair comparison, both mesh and flattened butterfly networks have the same total buffer size in number of bits and the same total bisection bandwidth (so the individual link width of FB is narrower than that of mesh).

The APLs are computed based on Equation (3). For each of the test case, the per-hop contention latency t_c is acquired by feeding the trace into Garnet, a cycle-accurate NoC simulator [1]. NoC power (comprised of router power and link power) is calculated by the NoC power model DSENT [11] with 32nm technology.

5.2 Simulation Results

5.2.1 Power-Latency Trade-offs

Figure 2 shows the simulation results of the five algorithms on flattened butterfly (aforesaid Schemes 4-8) in the form of trade-off curves between overall NoC power and average APL. Only two 4x4 and two 8x8 test cases are shown due to lack of space.

As shown in the figure, the trade-off curves of the two proposed heuristic algorithms are close to the optimal curve of branch and bound-based algorithm on 4x4 network. The branch and bound result is not shown for 8x8 network because it did not finish in a reasonable time period. Compared with Router Parking, we can see that at the same level of power consumption, FB_EC and FB_MV achieve 13% and 12% lower APLs on average, respectively, compared to FB_RPA. At the same level of APL, FB_EC and FB_MV save 28% and 27% of NoC power consumption on average, respectively, compared to FB_RPC.

It is worth mentioning that, the proposed FB_EC and FB_MV can produce a range of power-latency trade-off points that can be used by system operators under different constraints and scenarios.

5.2.2 Comparison of Eight Schemes

Figure 3 and Figure 4 compare the minimal NoC power consumption (left y-axis and the bars) that can be achieved by each of the

Table 1. Simulated network configurations.

Traffic	64-bit (80%) and 512-bit (20%) packets			
Network Size	4x4		8x8	
Network Type	Mesh	FB	Mesh	FB
Link Width (bit)	512	128	512	32
Average T_S	1	1.6	1	4.8
T_R	3	3	3	3
T_L	1	1	1	1
Router Radix	5	7	5	15

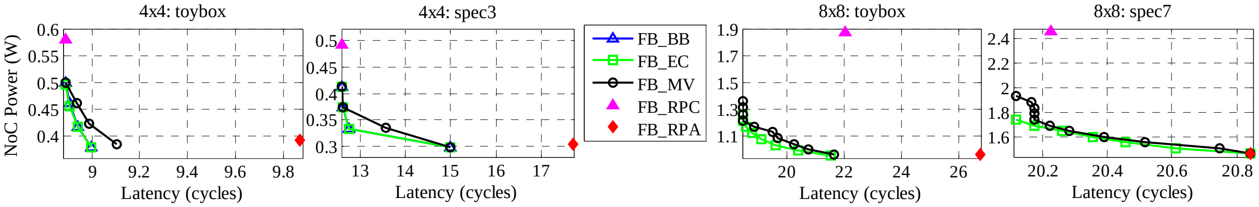

Figure 2. Tradeoff curves between overall NoC power and average packet latency.

eight schemes (x-axis), and the corresponding average packet latency (right y-axis and the curves) for different test cases. As can be seen, the power and latency results for FB_EC and FB_MV are very similar to those of FB_BB, demonstrating the effectiveness of the two proposed heuristic algorithms.

In addition, the proposed FB_EC and FB_MV schemes are considerably better than the mesh-based schemes in both power and latency. For example, FB_EC achieves on average 42.85% and 60.48% less NoC power consumption compared to the best results on 4x4 and 8x8 mesh network, respectively. This advantage mainly comes from two aspects. First, each router in the flattened butterfly (higher radix but narrower width) consumes less power compared to mesh routers (around 16% and 29% on 4x4 and 8x8 networks at the same injection rate, respectively). Second, the proposed algorithms can utilize the express channels in the flattened butterfly to maintain connectivity of active cores while allowing more routers to be powered off, thus having more power savings and less detours than the mesh. Note that although the serialization latency in flattened butterfly is slightly higher than in mesh, the reduced detours and the use of express channels in FB lead to much lower average packet latency than mesh.

5.2.3 Dynamic Power vs. Static Power

Figure 3 and Figure 4 also show the relative percentages between dynamic power and static power, which varies among different workloads. Overall, the static power percentage in flattened butterfly networks is slightly higher than that in mesh. This is because flattened butterflies have a smaller average hop count than meshes. In other words, packets in the flattened butterfly are forwarded through fewer routers, resulting in lower dynamic power and lower average injection rate per router. Table 2 shows the toybox example. As can be seen, when the same workload is executed on mesh and flattened butterfly, the static power percentage can be different due to the change in hop count and average injection rate. However, even if this relative static power percentage is higher in flattened butterfly, the absolute value of static power consumption of FB is still much lower than that of the mesh.

Table 2. Avg. hop count, inj. rate and static power percentage.

Test Case	Mesh			FB		
	Hop	Inj. Rate	Static %	Hop	Inj. Rate	Static %
toybox(4x4)	2.93	0.28	55	2.22	0.23	69
toybox(8x8)	7.73	0.24	59	3.51	0.18	76

6. CONCLUSION

In this paper, we propose *Smart Butterfly*, an effective NoC power-gating scheme that applies core-state-awareness to flattened butterfly networks. Smart Butterfly exploits the rich connectivity of flattened butterfly networks, and selectively powers off routers attached to sleeping cores to save more power. Furthermore, it achieves a wide range of power-latency trade-offs by adjusting the number of ON routers. We propose two heuristic algorithms to implement Smart Butterfly with different complexity and performance. Simulation results show that the two heuristic algorithms are able to achieve near-optimal solutions with low complexity, resulting in 42.85% and 60.48% less power consumption, on average, on 4x4 and 8x8 networks compared to a recently proposed mesh-based technique, respectively.

7. ACKNOWLEDGEMENT

This research is supported, in part, by the National Science Foundation (NSF) grant CCF-1321131 and the Software and Hardware Foundations program of the NSF.

8. REFERENCES

[1] Agarwal, N., Krishna, T., Peh, L. S., & Jha, N. K., GARNET: A detailed on-chip network model inside a full-system simulator. In *IEEE ISPASS*, pp. 33-42, 2009.

[2] Chen, L., & Pinkston, T. M., NoRD: Node-router decoupling for effective power-gating of on-chip routers. In *MICRO*, pp. 270-281, 2012.

[3] Chen, L., Zhao, L., Wang R., & Pinkston, T. M. (2014). MP3: Minimizing Performance Penalty for Power-gating of Clos Network-on-Chip", In *HPCA*, 2014.

[4] Das, R., *et al.*, Catnap: Energy Proportional Multiple Network-on-Chip," In *ISCA*, 2013.

[5] Kim, J., Balfour, J., & Dally, W., Flattened butterfly topology for on-chip networks. In *MICRO*, pp. 172-182, 2007.

[6] Kumar, A., Peh, L. S., Kundu, P., & Jha, N. K., Express virtual channels: towards the ideal interconnection fabric. In *ACM SIGARCH Comp. Architecture News*, *35*(2), pp. 150-161, 2007.

[7] Ma, S., Jerger, N. E., & Wang, Z., Whole packet forwarding: Efficient design of fully adaptive routing algorithms for networks-on-chip. In *HPCA*, pp. 1-12, 2012.

[8] Madan, N., Buyuktosunoglu, A., Bose, P., & Annavaram, M., A case for guarded power gating for multi-core processors. In *HPCA*, pp. 291-300, 2011.

[9] Matsutani, H., Koibuchi, M., Amano, H., & Wang, D., Run-time power gating of on-chip routers using look-ahead routing. In *ASP-DAC*, pp. 55-60, 2008.

[10] Samih, A., *et al.*, Energy-efficient interconnect via router parking. In *HPCA*, pp. 508-519, 2013.

[11] Sun, C., *et al.*, DSENT-a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling. In *IEEE/ACM NOCS*, pp. 201-210, 2012.

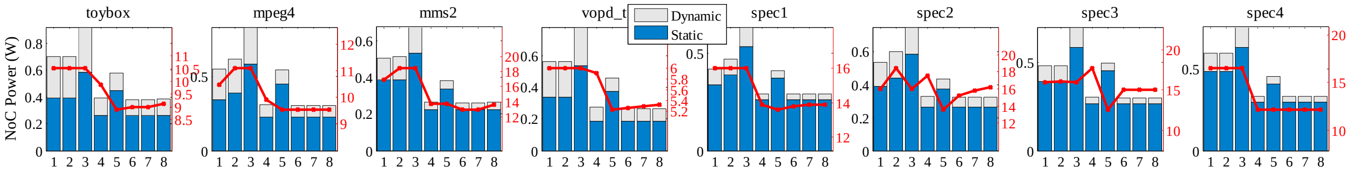

Figure 3. 4x4 results: (1-3) Mesh_BB, Mesh_RPA, Mesh_RPC, (4-5) FB_RPA, FB_RPC, (6-8) FB_BB, FB_EC, FB_MV

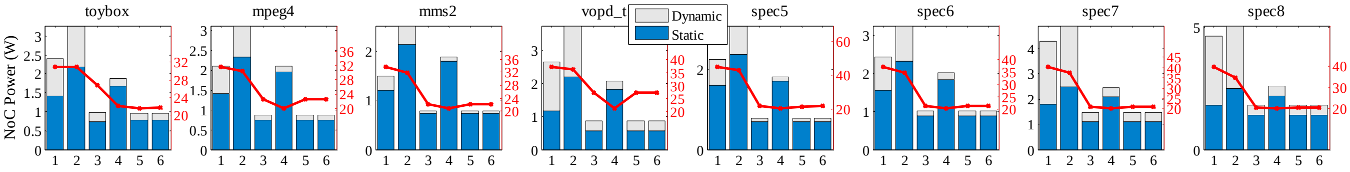

Figure 4. 8x8 results: (1-2) Mesh_RPA, Mesh_RPC, (3-4) FB_RPA, FB_RPC, (5-6) FB_EC, FB_MV