Towards Connectivity-Guaranteed Power-gating Large-scale On-chip Networks

Pengju Ren[†], Michel A. Kinsy[‡], Mengjiao Zhu[†] and Nanning Zheng[†]
[†]Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Xi'an, China, 710049
[‡]Department of Electrical and Computer Engineering, Boston University, MA, USA, 02215
Email: {pengjuren, nnzheng}@mail.xjtu.edu.cn mkinsy@bu.edu

***Abstract*—In this work, we present an efficient and practical algorithm, named COPAL (Connectivity preserving algorithm), to identify routers, that will cause network disconnection in an *off* position, based on a distributed *depth-first search* (DFS) through "neighbor-to-neighbor" communication. The algorithm allows the runtime system to make effective power-gating decisions in on-chip network based systems. For an N-node network, the time complexity of COPAL is $O(N)$ and the total number of messages sent in classifying nodes' criticality in optimizing network connectivity is of the order of $2Nlog_2(N)$. FPGA implementation shows that the algorithm is scalable and the required hardware resource overhead is minimal.**

***Keywords*—Power-gating, On-chip-Network Connectivity.**

I. INTRODUCTION

The power wall problem has forced multicore and many-core systems into the "dark silicon" era [1] [2] where large portions of silicon are effectively "dark"– either idle for long periods of time or significantly underclocked at the nominal operating voltage – to stay within the power budget. Esmaeilzadeh et al. [1] predict that over 50% of chips will not be used with the ITRS scaling at 8nm technology node. Taylor [2] estimates that in few years, designs may be 93.75% dark. The exponential growth of core count per chip coupled with the "dark silicon" problem has intensified the need for robust and efficient dynamic power management schemes. Strategic chip-wide power supply management is now required to narrowing the gap between available resources and sustainable power dissipation.

Power-gating has emerged as an important technique to mitigate the standby power consumption. Power-gating allows components or sub-systems to be switched *on* or *off* based on compute or communication workload variations during the application execution [3]. It can be applied at the logic block level, e.g., function units [4], or at the core or router level [5]. Although it has been shown that as the number of cores increases, a large portion of chip's power is consumed by the on-chip network (OCN). In current many-core systems, when processing units or cores are *turn-off* and not generating any network traffic, routers and links remain active in order to maintain on-chip network connectivity. This has prompted researchers and designers to investigate power-gating schemes for the on-chip routers [6].

Efficient and effective power-gating at the network-on-chip level remains an important research problem [7]. In particular, how to judiciously select router nodes to power *off* without disconnecting the network or causing deadlock is a major challenge. The difficulty stems from the fact that in a node, processing unit(s) or core(s) and local router have two different activity levels. A core may be idle or *power-off*, while the router is active and being used to transport packets for other nodes.

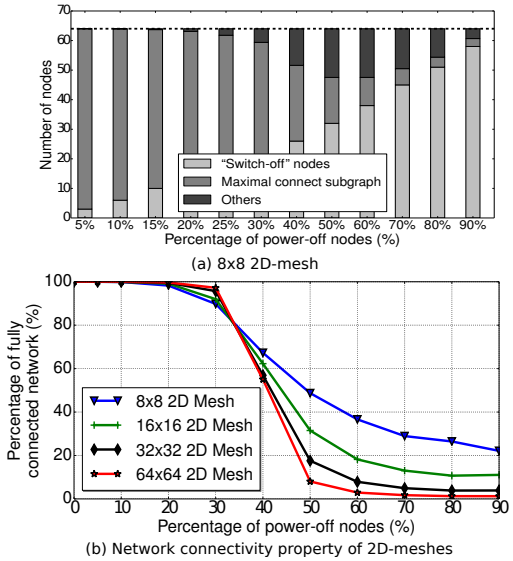

Fig. 1. Relationship between percentage of *inaccessible/power-off* components (a) dis- and connected components, and (b) probability of completely connected network for different netowrk size.

If, for power saving purposes, an essential router is *power-off*, the on-chip network will become disconnected and node pairs belonging to different subnets will not be able to communicate. This will severely impact the correctness and the deadlock freedom of the on-chip network routing protocol since packets and credits may be lost. Fig.1 shows network connectivity results induced by fine-grained dynamic power management, which assumes per-tile power-gating capability. Taking an 8×8 2D-mesh with uniform-random *power-off* node distribution as an example, the size of the maximal connected subgraph is 15.59 out of 32 active nodes, when 50% of nodes are switched off, see Fig.1.(a). While for a medium sized 2D-mesh network with 256 nodes, approximately 68.49% of the network loses connectivity when 50% of the routers are *powered-off*, see Fig.1.(b). Results from Fig.1 suggest that for large-scale networks, nodes should be turned off in cluster fashion when a large portion of the chip is inactive to preserve maximum network connectivity. Simply assumeing a uniform distribution of power-off nodes throughout the network is unrealistic, nevertheless, data in Fig.1 theoretically highlight

978-1-5090-5117-5/16/$31.00

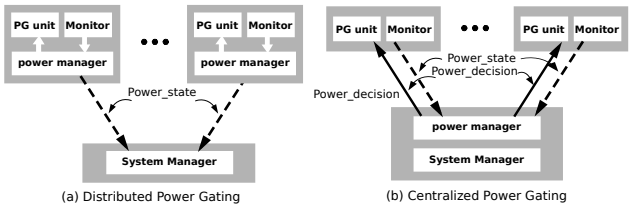

Fig. 2. (a) and (b) are our two baseline fine-grained power-gating algorithms.

the significance of maining network connectivity for power-gating. Further, it is important to detect the *cut elements* in a very efficient manner, and guide the runtime system in judiciously powering routers down while maintaining network connectivity, or to avoid assigning processes with continual communications to nodes belonging to different *subgraphs*.

A more formal approach to determine at runtime when a router can be safely *power-down* in power-gating mode would represent a significant contribution. In this work, we present an efficient and practical algorithm, named COPAL (**Co**nnectivity **p**reserving **al**gorithm), to identify routers, that will cause network disconnection in an *off* position, based on a distributed *depth-first search* (DFS) through "neighbor-to-neighbor" communication. Our algorithm allows the runtime system to make effective power-gating decisions.

This brief makes three major contributions to on-chip network routing in multicore/many-core systems: (1) an online algorithm to determine the criticality of a network node for maintaining network connectivity for power-gating, and (2) highlighting the potential inefficiencies associated with centralized power-gating management under network connectivity guarantees, and (3) propose and design a practical distributed algorithm can be implemented directly in hardware. To the best of our knowledge, it is the first distributed and scalable algorithm addressing the connectivity property for guiding power-gating decision for on-chip networks.

II. POWER-GATING IN MULTICORE SYSTEMS

A. Power-gating schemes

In multicore/many-core systems, power-gating is done either through a centralized scheme or in a distributed fashion [8]. In general, the centralized or global approach is more software-centric and uses the operating system or hypervisor to make power-gating decisions. With distributed techniques, power-gating decisions are often made in hardware via a simple state machine controller in each tile. Fig.2 depicts the two approaches. In both cases, a programmable power-gating (PG) unit is placed between the functional unit and its power rails. For the distributed approach (Fig.2.a), each processor tile has its own *PG unit*, *Monitor* and *Power Manager*. The *Monitor* tracks the idle time and the thermal profile of the node or a functional unit. If the *Power Manager* detects that the accumulative idle time recorded by the *Monitor* exceeds some programmed number of consecutive cycles or the temperature profile is approaching its warning value, PG unit is *switched-off* cutting the power supply to the functional unit. In this configuration where *Power Manager* is embedded in each tile, the *PG unit* switches between *powered-on* and *off* states independently using local information, e.g., the interval of time where internal datapaths stay empty or idle, and only sends its state information to the *System Manager*.

For the centralized management scheme, tiles do not have *in situ Power Manager* module, monitoring data are sent to the global power management module as part of the *System Manager* (Fig.2.b). After gathering all the state information, the *Power Manager* makes its power-gating decisions and distributes them to the computing tiles. Based on the decision received from the centralized *Power Manager*, the PG unit can then switch on or off the power supply locally.

In either case, to systemically and effectively manage on-chip resources, the system must be aware of the *on*/*off* states of each tile. Our algorithm applies to both distributed and centralized schemes of power-gating in on-chip network based systems.

B. Multiple applications

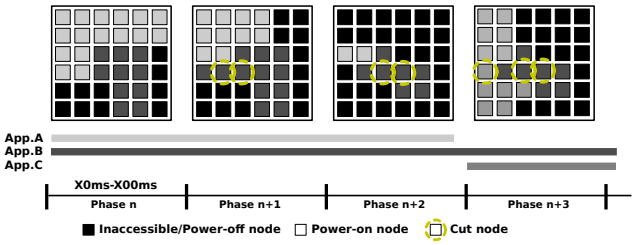

Fig. 3. Example of active nodes that vary with changing workloads or performance/power constraints of multiple applications on a manycore platform.

Fig. 3 demonstrates an example of simultaneously executed multiple applications. The set of active nodes for each application varies with the time-phase changes seen in the executing workload to dynamically adapt to some degree of parallelism. The critical nodes, surrounded by circle, even their associated processing units are idle, they need to keep *power-on* state to transport messages among other nodes that execute the same application. Therefore, a quick response method for connectivity identification is important to enable groups of nodes with connectivity requirements to make more judicious power-gating decisions.

III. COPAL: CONNECTIVITY PRESERVING ALGORITHM

A. Definitions and Framework

We begin the presentation of the COPAL algorithm by giving the standard definitions for the key concepts used in it formulation. DEFINITION *Given an NoC characterization graph $G = G(R,L)$, where the routers and links in the network are given by the sets R and L, $r_i \in R$ represents the router associated with processor element i, while each arc $l_{i,j} \in L$ represents a link from r_i to r_j.* For a given G, two vertices i and j are called **connected elements** if G contains a path from i to j; graph G is said to be **connected** if every pair of vertices in G is connected.

The connectivity of G dictates path diversity and routing choices in the NoC. Node pairs can only be able to communicate with each other belonging to the same **connected subgraph**. Supposing the vertex set $V^{cg} = \{G_1, G_2, ..., G_k\}$ contains all the *connected subgraph* G_i of G, the **maximal connected subgraph** G^{max} is the one with the maximal number of vertices. The V^{cg} is totally determined by the topology of the G.

DEFINITION A **rooted acyclic graph** is a graph in which one of the vertices is distinguished from the others. This

particular vertex is called the **root** of the graph. A rooted graph G with root r_{root}, any node j on the unique path from r_{root} to a node i is called an **ancestor** of i, and i is called a **descendant** of j. If the last edge on the path from the root r_{root} of the graph G to a node i is (j,i), the j is the **parent** of i, and i is called a **child** of j, denoted $i = j_{child}$ and $j = i_{parent}$. If two nodes have the same parent, they are **siblings**; a node with no children is a **leaf**.

The key insight of the COPAL algorithm is to identify the minimal set of network nodes, i.e., *cut elements* of subgraphs, to keep active during each power-gating time interval to maintain necessary network connectivity. If temporarily *switch-off* router happens to be a *cut element*, it will cause network disconnections where node pairs belonging to different subgraphs will not be able to communicate and breaking down the routing protocol.

B. Network Node Classification Algorithm

Depth-first searching (DFS) algorithm can be used to detect *connected elements* and *cut elements*. The *root* of the graph corresponds to the *System Manager* and is always active. The following rule serve as detection criteria [9]:

- **RULE**: A *root* vertex is a *cut vertex iff* it has at least two *children*. A non-root vertex u is a cut vertex *if and only if* it has a *child* v, with no back edge from v or any *descendant* of v to an *ancestor* of u.

Cut vertex classification
Given a vertex i, $((i_{root} == true)\ \&\ (|i_{children}| \geq 2))$ or
$((i_{root} == false)\ \&\ (\exists j \in i_{children}, i_{depth} \leq j_{low}))$
$\Rightarrow$ i is a *cut element*.

Proof: For a *root* node r, with $|r_{children}| \geq 2$, let us assume $r \neq cut\ element$, that is, $\exists (i\ and\ j) \in r_{children}$ such that when r is *off*, i and j can communicate with each other through another path.
Without loss of generality, let i be traversed before j. In a depth first traversal, $(|node_{parent}| = 1) \Rightarrow (node \neq root)$. $\forall k$ connected to i through $\{path|\ r \notin path\}$, $k_{parent} \neq r$. Since $(j \in k_{children}) \Rightarrow (j_{parent}$ must not be $r)$. This is a contraction because j_{parent} is in fact r. Therefore, there cannot be such a path. And we can conclude that $((i_{root} == true)\ \&\ (|i_{children}| \geq 2)) \Rightarrow i$ is a *cut element*.

For a *non-root* node i, $i_{depth} \leq j_{low}$ and $j \in i_{children}$, if $j_{descendants}$=Ø and the only edge connected to j is (i,j), then j will be disconnected when i is turned *off*. This implies that i is a *cut element*. If $j_{descendants} \neq \emptyset$ and i is not a cut element, a communication path between j and an ancestor k of i can still be established even when i is *power-off*. It follows that $\exists (k,j) \in \{(m,n)|\ m \in j, j_{descendants}, n \in i_{ancestors}\}$. And $\exists j \in i_{children}\ with\ i_{depth} > j_{low}$. However, we assumed $i_{depth} \leq j_{low}$, therefore, there exists no such non-cut element. It implies that i is a *cut element*. ■

C. Algorithm Description

COPAL is a four-phase algorithm that must be executed at startup in order to detect the *maximal connected subgraph* and mark all disconnected components caused by faults in the initialization phase of each time interval.

Anchoring Phase: In this phase, the "System Manager" node acts as the *root* and starts the execution to build the *depth-first tree* by transitioning into the *Forward Phase*;

Forward Phase: This phase explores undiscovered neighbors. Particularly, it discovers every vertex connected to the current node and going as *deep* as possible until it reaches a *leaf node*. In the process it updates $depth$ and low values, and sets up *parent* and *child* relationship between nodes along the paths. There is a node *counter* that keeps count of nodes during the forward searching. It is incremented by 1 when a new UNVISITED node is discovered;

Backward Phase: It performs the tree traversal from a node back to its *ancestor*. During a *backward searching*, at intermediate *parent* nodes, if an UNVISITED neighbor is discovered, a *forward searching* is initiated. Otherwise, it continues searching back until it reaches the *root* node. It updates the *low* values of nodes along *backward* paths. The node *counter* remains unchanged during backward searches;

Classification Phase: This phase finally classifies nodes into *cut elements* and *non-cut component*. The *cut elements* detection is executed based on local information.

The algorithm is triggered from and terminated at both the *root* node, *counter* records number of *connected components* after the execution. Node relationships in the *depth first tree* are updated during the *forward searching*. Each router node is marked whether it is a *cut element* or not after the execution. This allows power-gating scheme to selectively switch *on* or *off* routers to guarantee maximum network connectivity while minimizing the number of active routers associated with idling processing elements. The detailed algorithm is presented in **Algorithm 1**.

COPAL can be easily applied to identify critical elements inside a group of nodes that the application is executing on, as in Fig. 3. In this case, one selected node in a group act as the *root*, while the *forward* and *backward* phases only traverse its neighbors with an identical label that indicates they belong to the same group. Different application groups can execute COPAL simultaneously. In this way, applications of a resource-aware programming paradigm [10] can efficiently control the power-states of their resources.

D. Complexity Analysis

With COPAL, each edge in the network is traversed at most twice. The upper bound of the algorithm is $2L$ steps, where L is the number of edges in the network. The COPAL framework applies to both distributed and centralized *System Manager* schemes. For a 2D-mesh network with N nodes, the massage size is about $log_2(N)$. Messages contain node *id* and local power utilization and node activity data. The average transmission distance (hop count) from a node to the *System Manager* is $\sqrt{N}/2$, if $\sqrt{N}$ is odd, otherwise, it is $N^{3/2}/2(N-1)$, assuming the *System Manager* is sitting in the middle of the network. Thus, the total message volume of "all-to-one" transmission is around $N^{3/2}log_2(N)$. At *System Manager*, at least an $N(log_2(N)+2)$ sized memory is required to store all these information. The computational complexity

Algorithm 1: Pseudocode of COPAL Algorithm

Initialization
  i is the unique id of *System Manager* node ;
  i_{parent} = NULL; i_{child} = NULL ;
  $i_{low} = i_{depth} = \infty$, i_{cut} = False, $counter$ = 0 ;
for $j \in i_{neighbors}$ **:**
  $(i,j)_{cutedge}$ = False ;
Anchoring phase at node i
if *i in graph & i is* UNVISITED **:**
  i_{root} = True ; /* *i is the root* */
  $i_{depth} = i_{low} = 0$, $counter = 1$;
if $\exists j \in i_{neighbors}$ *& j is* UNVISITED **:**
  $i_{child} = j$;
  Send Forward inputs $(i, i_{depth}, counter)$ to j;
Forward Phase $(i, i_{depth}, counter)$ **at** j
$j_{depth} = j_{low} = i_{depth} + 1$;
$j_{parent} = i$, $counter$++;
if $\exists k \in j_{neighbors}$ *& k is* UNVISITED **:**
  $j_{child} = k$;
  Send Forward inputs (j, j_{depth}) to k;
else:
  Send Backward inputs $(j, j_{low}, counter)$ to j_{parent};
Backward Phase $(j_{low}, counter)$ **at** i
if $\exists k \in i_{neighbors}$, *k is* UNVISITED **:**
  Send forward inputs $(i, i_{depth}, counter)$ to k;
else:
  $i_{low} = min(i_{depth}, min(m_{depth}, \forall m \in i_{neighbors} \setminus i_{parent}), min(n_{low}, \forall n \in i_{children}))$;
  execute classification algorithms at node i;
  if i_{root} == *true* **:**
    Finish *Depth-first Searching*;
    Return $counter$;
  else:
    Send Backward inputs $(i, i_{low}, counter)$ to i_{parent};

Classification at node i
if i_{root} == *True* **:**
  if *more then 2 nodes in* $i_{neighbors}$ **:**
    i_{cut} = True ; /* *i is a cut element* */
  else:
    $i.cut$ = False;

elif $\exists n \in i_{children}, i_{depth} \leq n_{low}$ **:**
  i_{cut} = True ; /* *node i is a cut element* */
  $(i,j)_{cut-edge}$ = True;

of calculating the *cut components* and *connected components* is $O(N)$ [9].

For centralized power management schemes, another around of "one-to-all" transmission containing power-gating decisions with total message volume of $N^{3/2} log_2(N)$ is sent from *System Manager* to all the tiles. The information gathering and decision distribution phases would typically require tens of thousands of cycles for a typical wormhole based medium sized network, even for a multiple-NoC with dedicated transmission resources for the system power management. Our experiment results show averages of 2871 cycles for the "all-to-one" communication and 2809 cycles for the "one-to-all" decision transmissions on a dedicated NoC, 256-node, 2D-mesh network. More details are presented in the evaluation section (cf., V).

For the *connected components* classification, the average system latency is 68482 cycles which represents a significant portion of execution time and network bandwidth utilization. The main reason is because "all-to-one" and "one-to-all" traffic patterns are self-congested and lead to longer transmission latency. In [8], Niti stated that for the distributed scheme, the optimal idleness threshold is 2ms, while for the centralized scheme, the ideal time interval is 100ms. The centralized method is limited in its scalability. As the size of network grows, the monitor-and-actuate power-gating control loop degrades [11]. Bandwidth requirements and time delays tend to limit the associated "all-to-one" and "one-to-all" transmissions. There is also a drastic increased in memory requirement in centralized schemes. These drawbacks make centralized power-gating approaches difficult to implement on large NoCs under tight power and area budgets. Therefore, implementation of COPAL in a distributed power-gating framework is more effective and efficient.

E. Illustrative Example

In this illustrative case, we consider a network consisting of 10 nodes. There are *ids* associated with each node. *Cut elements* are highlighted using dashed lines. **Fig.4.(a)** shows node 8 as switched *off*. At node 0 - "root", there are two UNVISITED neighbors 2 and 3. The algorithm randomly chooses node 2 as a *child* and includes 2 in $0_{children}$). It then sends a *Forward (0,* 0_{depth}=0*, counter*=2*)* message to node 2. It updates 2_{depth}=2_{low}=1, increments *counter* by 1, and forwards message *Forward (2,* 2_{depth}=1*, counter*=3*)* to UNVISITED node 3, shown in **Fig.4.(d)**. The searching process continues until node 5 is reached (**Fig.4.(f)**). Node 5 has no UNVISITED neighbor, so it sends a *Backward (*5_{low}=4*, counter*=6*)* to its *parent* node 4. The algorithm continues by exploring UNVISITED neighbors 6 and 7 of node 4.

Node 5 is 4's child, and $5_{low} = 4 > 4_{depth} = 3$, so according to *cut element* detection **Rule**, node 4 is labeled as a *cut element* as shown in **Fig.4.(g)**. At node 7, the algorithm discovers that there is a *back edge* connecting node 3, and 7_{low} is updated to $min(7_{depth}$=5, $min(6_{depth}$=4, 3_{depth}=2))=2 (**Fig.4.(j)**). Backward searching continues at node 3 and updates 6_{low}=2 and 4_{low}=2 along the path, shown in **Fig.4.(l)**. At node 3, there is a *back edge* connect it with *root* 0. *Backward* messages are sent to *root* 0 through node 2. Parameters 3_{low} and 2_{low} are updated to 0, and *non-root* node 3 is classified as a *cut element* since 3_{depth}=4_{low}=2, where node 4 is 3's *child* (**Fig.4.(m)** and **(n)**). *Backward searching* then moves to the *root* node (0) and finds its *neighbors*. Node 0 is labeled as a *cut element* since its neighbors have already been visited and it has two *children* 1 and 2.

The algorithm finishes constructing the current *depth first tree* with *root* node 0 and records the size of *connected subgraph* is 8 (*counter*=8), shown in **Fig.4.(o)**. There are still UNVISITED nodes 8 and 9 in the network. If they are not active and added to the tree construction, they will be disconnected. In which case, the number of *maximal connected subgraph* will be 8.

IV. RELATED WORK

Finding *cut-components* has been studied in graph theory and wireless network. Conventional centralized *cut element* detection methods [12] [13] require knowledge of the complete network topology. They are implemented either "off-line" or using a "supervisor node" to maintain the network information. Samih proposed an algorithm which implements a centralized fabric manager that gathers nodes information and reconfigures routing decisions to maintain the network connectivity [6]. In [14], Lizhong presented a power-gating bypass techniques,

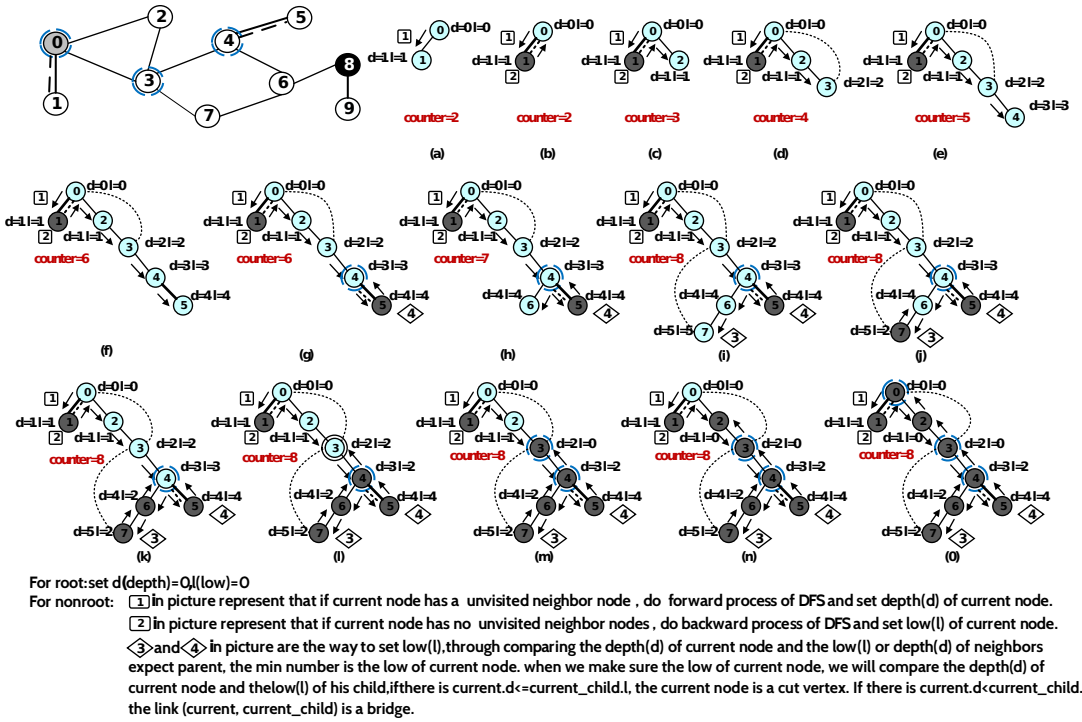

Fig. 4. Illustrative example of COPAL. The power of node 8 is cut-off by power-gating, which seperates the network to two subgraphs. Arrows from lower *ids* to higher *ids* are for the **Forward Phase**, otherwise are part of the **Backward Phase**. *Cut elements* 0, 3 and 4 are surrounded with dot line, *cut edges* (0,1) and (4,5) are highlighted with dashed lines.

named NoRD. Lizhong also proposed a "power punch" algorithm [15] that sends power control signals ahead of packets. This allows packets to be transported without suffering any router wake-up or packet detour latency. Das in [16] uses a multiple-network, where each subnetwork can be power gated without compromising the connectivity of the network.

These works have not fully investigated the effect of power-off nodes on network connectivity and communication overhead for gathering power-states information from each tile. In order to assist power gating modules to make better decisions, *connected elements* detection must be fast enough to accommodate frequent communication or resource changes. In addition, as mentioned by Annavaram [8], power management policies should scale well as the number of cores per chip is continuously increasing from technology generation to generation.

V. Evaluation and Discussion

In this section, we present the simulation results of the COPAL algorithm. For the experimental setup, we implement 8×8, 16×16 and 32×32 2D-meshes/tori with different number of *power-off* nodes. We assume an uniform-random distribution of *power-off* nodes over the network and repeat the simulation 10,000 times for each case. HORNET [17] is the simulator used for all the experiments.

A. Execution Latency

For this part of the analysis, we implement the baseline distributed and centralized power management scheme with and without the COPAL using the *SMT32F103RB*, an ARM Cortex-M3 core with 20KB RAM and 128kB ROM, as the hypervisor. For the on-chip network, we use a 5-stages 2 virtual channels wormhole router using DOR routing algorithm. The execution times are summarized in Table I. COPAL used in a distributed fashion shows 98.63% and 63.4% execution time saving compared with the centralized and distributed approaches, respectively.

Network size	Executation time in clock cycles				*Copal*
	"all-to-1"	Comput.	"1-to-all"	Total	
8×8	576	17090	571	18237	251
16×16	2871	68482	2809	74108	1019
32×32	13314	274178	13304	300796	4091

TABLE I. Comparison in terms of execution cycles.

As mentioned above, optimal power gating intervals for distribued and centralized schemes are around 2ms and 100ms, respectively. For a 2GHz clock frequency system, the centralized detection execution time is around 9.1ms, 37ms and 150.4ms for 64- 256- and 1024-nodes networks. These execution time either exceed or make up a large portion of the power gating period. COPAL execution time is very low; it only takes around 0.13ms, 0.5ms and 2ms (251, 1019 and 4091 clock cycles) for different network scales. The performance impact in the form of longer execution latency can be further reduced through power-gating during system wake-up. It leaves the system enough time for further optimize its operations (e.g., making optimal power-gating decision through multiple iterations or workload rebalancing among turn-on nodes).

B. Hardware implementation of COPAL

All the messages are transmitted through "neighbor-to-neighbor" transmission, a $log_2(N)$-bits register is applied to record *counter* and two one-bit sized registers are required to record whether current node is a *root* and *cut-node* or not. Another two $log_2(N)$-bits registers are needed to record the *depth* and *low* values, besides, a 3×D-bits look-up-table(LUT) is required to store *parent* and *child* relationship with its directly connected neighbors, *D* is the degree of nodes.

Module	8x8 2D mesh			16x16 2D mesh		
	LUTs	Regs	PCT	LUTs	Regs	PCT
Router	6252	2068	-	6614	2108	-
COPAL	127	124	3.017%	141	148	3.313%

TABLE II. FPGA IMPLEMENTATION RESOURCE UTILIZATION

Network Size	4×4	8×8	16×16	32×32	64×64
LUTs	98	127	141	157	168
Regs	95	124	148	169	182

TABLE III. THE RESOURCE OVERHEAD OF COPAL ON FPGA, RANGING FROM 16 TO 4096 [11] 2D-MESH.

Moreover, the combinational circuits to implement the required function are also quite simple. The design is implemented on a Xilinx Virtex7-2000T FPGA device. The baseline router with 4VCs and each VC contains 32 *flits*, message width is 64-bits. COPAL uses only a small fraction (around 3.017∼3.313%) of hardware resource allocated to the router for 8×8 and 16×16 2D-meshes, see Table II. Table III shows the increase of resource overhead alongside the increasing of number of total nodes as much as 4096, which approximates the logarithmic curve. Because there are two $log_2(N)$-bits sized $depth$ and low registers which grows logarithmically with network size. The hardware implementation shows that COPAL scales well with network size.

C. Deadlock Avoidance

If, for power-gating purposes, an essential router is power-off, the on-chip network will become disconnected and node pairs belonging to different subgraphs will not be able to communicate. Messages sent between those node pairs will create back pressure leading to routing deadlocks even for pairs in the same subgraph, if no time-out mechanism for in-network packets is built into the system.

It is precisely to avoid or significantly reduce the occurrence of an essential router being turned off during a power-gating action that Copal is proposed. In cases where disconnected subgraphs are unavoidable due to faults and power constraints, messages can only be transmitted within subgraphs following well prescribed deadlock freedom rules. The support for in-subgraph-only messaging, therefore, should be determined and provisioned for off-line during system configuration like at routing tables programming stage.

VI. CONCLUSIONS

Fine-grained dynamic power management has become an essential part of energy-aware chip design. Efficient and effective power-gating at the network-on-chip level remains an important research problem. In this work we examine online algorithms to determine the criticality of a network node for maintaining network connectivity for power-gating. We develop a distributed and scalable algorithm, named COPAL, for identifying router nodes that will cause network disconnection in an *off* state to allow the runtime system to make more judicious power-gating decisions. The algorithm places no restriction on topology. The work has also shown that distributed control strategy is preferred over centralized holistic power management for large-scale manycore systems.

In current work, we mainly concentrate on maintaining the network connectiivty for power-gating. it also worth and necessary exploring the intergation of COPAL with task mapping strategy as well as routing solutions to give a comprehensive power management policy in the future works.

VII. ACKNOWLEDGMENTS

This research is partially funded by NSFC grant No.61303036, China Postdoctoral Science Foundation No.2016M590949, National Key Research and Development Program No.2016YFB0200202 and Nationl High Technology Research and Development Program of China No.2014AA01A301.

REFERENCES

[1] H. Esmaeilzadeh, E. Blem, R. St Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *Computer Architecture (ISCA), 2011 38th Annual International Symposium on*. IEEE, pp. 365–376.

[2] M. B. Taylor, "A landscape of the new dark silicon design regime," *Micro, IEEE*, vol. 33, no. 5, pp. 8–19, 2013.

[3] A. Lungu, P. Bose, A. Buyuktosunoglu, and D. J. Sorin, "Dynamic power gating with quality guarantees," in *Proceedings of the 2009 ACM/IEEE international symposium on Low power electronics and design*. ACM, pp. 377–382.

[4] H. Jiang, M. Marek-Sadowska, and S. R. Nassif, "Benefits and costs of power-gating technique," in *Proceedings of the 2005 International Conference on Computer Design*, ser. ICCD '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 559–566.

[5] L. Chen and T. Pinkston, "Nord: Node-router decoupling for effective power-gating of on-chip routers," in *Microarchitecture (MICRO), 2012*, pp. 270–281.

[6] A. Samih, R. Wang, A. Krishna, C. Maciocco, C. Tai, and Y. Solihin, "Energy-efficient interconnect via router parking," in *High Performance Computer Architecture, 2013 IEEE 19th International Symposium on*. IEEE, pp. 508–519.

[7] M. Shafique, S. Garg, J. Henkel, and D. Marculescu, "The eda challenges in the dark silicon era," in *Design Automation Conference (DAC)*, 2014, pp. 1–6.

[8] M. Annavaram, "A case for guarded power gating for multi-core processors," in *High Performance Computer Architecture, 2011 IEEE 17th International Symposium on*, pp. 291–300.

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Introduction to algorithms, 2009."

[10] L. Moreau and C. Queinnec, "Resource aware programming," *Acm Transactions on Programming Languages & Systems*, vol. 27, no. 3, pp. 441–476, 2005.

[11] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[12] S. Shamshiri, A.-A. Ghofrani, and K.-T. Cheng, "End-to-end error correction and online diagnosis for on-chip networks," in *Test Conference (ITC), 2011 IEEE International*. IEEE, pp. 1–10.

[13] K. Akkaya, F. Senel, A. Thimmapuram, and S. Uludag, "Distributed recovery from network partitioning in movable sensor/actor networks via controlled mobility," *Computers, IEEE Transactions on*, vol. 59, no. 2, pp. 258–271, 2010.

[14] L. Chen and T. M. Pinkston, "Nord: Node-router decoupling for effective power-gating of on-chip routers," in *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 270–281.

[15] L. Chen, D. Zhu, M. Pedram, and T. M. Pinkston, "Power punch: Towards non-blocking power-gating of noc routers," in *IEEE International Symposium on High Performance Computer Architecture*, 2015, pp. 378–389.

[16] R. Das, S. Narayanasamy, S. K. Satpathy, and R. G. Slinski, "Catnap: energy proportional multiple network-on-chip," *Acm Sigarch Computer Architecture News*, vol. 41, no. 3, pp. 320–331, 2013.

[17] P. Ren, M. Lis, M. H. Cho, K. S. Shim, C. W. Fletcher, O. Khan, N. Zheng, and S. Devadas, "Hornet: A cycle-level multicore simulator," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 31, no. 6, pp. 890–903, 2012.