A Reconfigurable NoC Topology for the Dark Silicon Era

Mehdi Modarressi¹, Hamid Sarbazi-Azad²,³

¹Department of Electrical and Computer Engineering, College of Engineering, University of Tehran, Tehran, Iran.
² Department of Computer Engineering, Sharif University of Technology, Tehran, Iran.
³ School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran.
modarressi@ut.ac.ir, azad@{ipm.ir,sharif.edu}

ABSTRACT
Future chips are expected to feature hundreds of on-die resources, but a considerable portion of silicon area in these chips will be dark that cannot be powered and provided with required bandwidth. As a result, only a limited number of cores of future processors can be powered on simultaneously. The most promising solution to this challenge is to trade off the cheaper silicon area with energy-efficiency by integrating a selection of many diverse application-specific cores into a single multi-core processor and only activate those cores that best match the processing requirements of each running application. Activated cores act as an application-specific CMP for target applications. In this paper, we propose a reconfigurable network-on-chip that leverages the routers of the dark portion of the chip to customize the NoC topology for the active cores at any time. In this design, routers of the dark parts of the chip can be downgraded to simple FPGA-like switches to directly connect distant active nodes in the network. Our experimental results reveal considerable reduction in energy consumption and latency of on-chip communication when compared to state-of-the art NoCs.

Categories and Subject Descriptors
C.1.2 [Computer Systems Organization]: Multiprocessors; Interconnection architectures

General Terms
Design, Experimentation, Performance.

Keywords
NoC, Performance, Circuit-switching, Dark silicon, Topology.

1. INTRODUCTION
The exponential increase in transistor count, coupled with the ever increasing demand for higher performance in embedded, desktop, and server computers, have moved the semiconductor industry toward many-core chip multiprocessors (CMPs) and systems-on-chip (SoCs). Major semiconductor manufacturers already ship chips with few tens to hundreds of cores and chips with several hundreds to thousands of cores are likely to appear in the future [1].

However, power constraints at current and future technology nodes limit the achievable performance improvement. Due to this practical constraint, increasing core counts can no longer directly result in performance improvements, because we do not have enough power budget to power on all cores simultaneously. Moreover, chips are also physically limited by the off-chip bandwidth to work with the memory, because I/O pads cannot be scaled well with Moore's Law [1]. Consequently, chips will not satisfy the excessive memory bandwidth requirement of the rising core and thread counts. To stay within the limited power and bandwidth budget, therefore, only a limited fraction of cores can be utilized at the same time, leaving the remaining fraction, referred to as dark silicon, powered off [1][2].

According to different projections, without a technological improvement to overcome the power and bandwidth constraints, the dark silicon is expected to cover from 50% to 90% of the chip area in near future [2][3].

Since silicon area in the dark silicon era is a cheaper resource than energy, core specialization, which trades area with power efficiency, is the most promising solution to the dark silicon problem [1][3]. With specialized cores, a selection of many diverse application-specific cores might be integrated into a single multicore chip. Specialized cores are designed by characterizing the target workloads and identifying the functional units that execute computation-intensive parts of the codes and then implementing those parts in a performance- and power-efficient manner. Such specialized cores may include ASIC accelerators, GPUs, DSPs, and FPGAs and can offer orders-of-magnitude better energy-efficiency than a general-purpose core. For an input application, then, those cores that better match its processing requirements are activated to run the application, while the unused cores are left power-gated to stick to power limit.

The network-on-a-chip (NoC) communication paradigm has become more and more prevalent in modern multi- and many-core systems. These networks must provide low latency and high-
bandwidth for on-chip communication while satisfying the limitations imposed by tight area and power budgets.

The performance of a NoC is extremely sensitive to its topology because topology determines the system cost, as well as performance bounds for the network by setting the average message hop count and network bisection bandwidth. Finding a proper topology for an application (as well as task to node mapping for the selected topology), in which the number of intermediate routers between communicating cores with heavy communication demand is minimized, is an important target for many NoC optimization methods. Note that when the number of intermediate routers between two communicating cores is reduced, the power consumption and latency of the communication between them decreases proportionally.

Several previous works have explored potential benefits of topology reconfiguration to optimize NoC power/performance [4][5][6][7]. Authors in [5] introduce a polymorphic NoC which is constructed by a configurable set of building blocks including buffers, crossbars, and links. The network can be configured to offer the same performance as a fixed function network while incurring 40% area overhead, on average. However, authors have not analyzed the power consumption of their NoC architecture, nor do they propose algorithms to customize NoC for an application. In [7] a reconfigurable NoC is presented that can dynamically match its topology to the communication pattern of a given application. The reconfiguration of this NoC is achieved by inserting a set of simple reconfiguration switches in the network to dynamically change the NoC inter-node connections. The reconfiguration switches can establish a shortcut path between nodes with high communication demand to reduce their topological distance and carry their packets with lower latency and power consumption. By dynamically tailoring the topology to on-chip traffic, this NoC can offer up to 30% reduction in power consumption and 34% reduction in network latency, but at the price of up to 45% area overhead imposed by extra links and configuration switches that add reconfiguration capability to the NoC.

In a many-core chip, with a vast majority of on-chip cores left powered off (or dark), it is very likely that the active core areas be non-contiguous and spread out across the network. A conventional NoC still necessitates all packets to go through the router pipeline at all intermediate nodes (both active and inactive nodes) along their path on a hop by hop basis. In this case, packets may suffer from long latencies if active nodes are located at a far topological distance.

In this work, we show how the routers of dark regions of a chip can be used as bypass paths, a logic like switch boxes in FPGAs, to reconfigure the topology in favor of the active subset of cores. The proposed NoC aims at providing virtual inter-core connections among active cores by bypassing dark cores to enable NoC to operate in the same way as the case when the cores are placed nearby.

Since our proposal realizes application-specific topologies over structured and regular components, it stands between these two extreme points of topology design and benefits from both worlds; it is designed and fabricated like a regular NoC, but can be dynamically configured to a topology that best matches the traffic pattern of the active cores.

The rest of the paper is organized as follow. Section 2 presents the proposed NoC architecture. The topology reconfiguration algorithm is introduced in Sections 3. Section 4 presents experimental results and finally, Section 5 concludes the paper.

2. Dark Silicon-aware NoC Reconfiguration

By adding some extra logic to implement FPGA-like switches in a conventional NoC router [8], we can get the unused routers (connected to dark cores) of a many-core CMP to act as bypass paths in order to customize the network connectivity among the active fraction of cores. Figure 1 shows a router in this design. Every input port has an internal shortcut path, hence the new router combines an FPGA switch box with a conventional packet-switched router. It that allows packets skip the router pipeline stages (e.g. buffering, routing, switch and VC allocation) and directly head for the crossbar. REG in Figure 1 is a register that allows flits to get buffered when they arrive at the router and then continue their journey in the next cycle in a pipelined manner. In this case, a virtual long link is formed to directly connect two remote active nodes. The other modified component is the output arbiter; it is augmented by a register which can establish a fixed crossbar connection for a given duration.

![Figure 1. The architecture of a dark silicon-aware router](image)
pipeline stages and just pass through the crossbars and links which form the physical path between the two endpoint nodes of the virtual long links.

Bypassing intermediate router can lead to considerable performance improvement since the latency and power consumption of routers make significant contributions to the total NoC latency and power consumption. For example, in Intel’s 80-core TeraFlops, more than 80% of the on-chip communication power budget is consumed by routers [9].

Figure 2.a shows the case where a workload is running on a selected set of specialized cores, while the remaining cores are left dark (inactive). In this figure, the active nodes are connected by a mesh topology implemented on the dark routers which are now downgraded to bypass switches. In this case, the active cores are connected by a mesh topology that excludes many unnecessary routers, hence all network functions (such as routing algorithm, deadlock avoidance, load balancing,...) that are designed for the original mesh can be applied for this new network. In Figure 2.b an irregular customized topology is made for an application whose communication task graph is depicted in Figure 2.c.

![Figure 2. Implementing a mesh (the links are bidirectional) (a) and an application-specific topology (b) for active cores of a dark silicon-aware reconfigurable NoC. The application-specific NoC is designed for H.263 application [10] (c).](image)

Assuming a heterogeneous CMP where each task is allowed to (or gives the best performance when) run on a specific node, the topology is changed in such a way that the frequently communicating nodes have the minimum topological distance. Ideally, all communicating cores will be provided by a direct virtual long link. In practice, however, this is sometimes impossible: the application may have many communication flows (in particular if a large portion of cores are active) and the limited bypass switches (inactive routes) cannot afford to provide direct connection for all of them. In this case, some communication flows may be directed through a multi-hop path (e.g. communication between nodes 1 and 5 is done through node 6 in Figure 2.b). Please note that the NoC is still a packet-switched network, but its topology is customized for the on-chip traffic.

3. Topology Reconfiguration Procedure

In this section, an algorithm to establish a customized topology for the current active cores is presented. The Proposed NoC and its design flow can support both CMP and multicore SoC workloads.

Multicore SoCs are often used in embedded systems where applications and their traffic characteristics are often known at design time. Each input application is spatially partitioned into several tasks, each of which is assigned to a processing unit. The inter-core communication pattern remains relatively static as each core performs a fixed task. Communication Task Graph (CTG) of each application then provides estimates of bandwidth demands between any two nodes and can be obtained through program profiling. In such applications, inter-core communication often occurs across a certain limited number of connections, i.e. each node generally communicates with a few other cores.

On the other hand, most applications targeting homogeneous CMPs composed of multiple similar tasks that all execute the same code, but on different data. As these applications often adapt the shared-memory paradigm, the source of on-chip traffic in such applications is data block transfers initiated by reading from and writing to a shared data. Therefore, all source-destination pairs will potentially exchange packets and there must be at least one path between all node pairs when running these workloads. Consequently, the communication task graph of these applications is in the form of a fully connected graph.

**Problem formulation.** The CTG of input applications is a directed graph $G(V,E)$, where each $v \in V$ represents a task, and a directed edge $e_{ij} \in E$ represents the communication flow from $v_i$ to $v_j$ with the communication volume (bits per second) of $l(e_{ij})$. The subset of NoC cores that will execute the tasks of an incoming application (and associated routers) is represented by $ActiveC$, while the remaining cores form the $InactiveC$ set.

The algorithm takes the CTG and $ActiveC$ of an input application as input and constructs virtual long links over the cores of $InactiveC$ (through the internal shortcut paths of these routers) such that

$$\text{Min}\{\sum_{e_{ij} \in E} l(e_{ij}) \times \text{Hop}(v_i,v_j)\}$$

Where $\text{Hop}(v_i,v_j)$ is the shortest path (in terms of the number intermediate routers) between nodes $v_i$ and $v_j$ in the new topology.

**Topology reconfiguration algorithm.** Initially all routers of $InactiveC$ are unconfigured. Algorithm starts by selecting the CTG edges in the decreasing order of bandwidth demand and then, Dijkstra’s shortest path algorithm is used to find a path with minimum weight between the endpoint nodes of the edge. In this procedure, the network (including both $InactiveC$ and $ActiveC$ nodes) is considered as a
directed weighted graph and the cost of a path is calculated as the cumulative cost of the routers and bypass switches (InactiveC routers) that it contains. We assign a cost of 1 to a link ending to a bypass switch and a cost of 3 to a link ending to a router. These costs reflect the power/latency ratio of the switches and routers and bias the algorithm to find a path through the bypass switches, hence facilitates creating longer virtual links for communication flows. Virtual long links are not allowed to overlap; so, if a bypass switch is already configured in previous iterations of the algorithm for the flows with higher traffic rates (e.g., by connecting its E input port to S output port), the algorithm is not allowed to use conflicting turns on that switch (for example, connecting E input port to W input port). This is done by setting the cost of such turns to infinity.

To restrict Dijkstra to return topological shortest paths, it only considers the links that progress towards the destination node, i.e. the links that reduce the distance to the destination by one hop at each step. The algorithm is not also allowed to violate the maximum bandwidth of the NoC links.

If the current configuration of the bypass switches blocks all paths between the endpoint nodes of an edge, the algorithm (1) selects one of shortest paths of the edge that has minimum number of blocking bypass switches (switches whose configuration prevent the path advancing toward the destination) and (2) selects blocking switches along that path one by one and makes them revert to a router again, and then (3) reverts path finding, until a path is found. When a bypass switch reverts to a router, all virtual long links passing through this node are broken into two parts. The routing tables of the router are configured appropriately to bridge the two virtual link parts via the router.

After a path is found, all bypass switches and routing tables involved in the path construction are configured accordingly and the algorithm continues with the next flow. Bypass switch configuration is done by setting the value of the select lines of the multiplexer of the input units and the output allocator registers at the crossbar. As the number of communication flows is often limited in such systems, the routing table logic size is comparable to the size of an FSM-based routing logic (e.g. the X-Y routing logic).

A path may contain several routers and virtual long links. For example, the path between nodes 7 and 8 in Figure 2.b is composed of two long links (7→4 and 4→8) and an intermediate router. Packets traveling from 7 to 8 bypass the routers located between 7 and 4, but enter the router pipeline stages in node 4. After route computation in node 4 (done by routing tables filled during path selection), packets take the next long link to bypass the routers located between 4 and 8 and directly reach the destination.

**Topology setup.** Topology selection can be done offline to find an appropriate topology. The topology is then stored with the application and will be loaded onto the network once the application starts. Alternatively, topology calculation can also be done online before starting the application. The online scheme is especially beneficial when the chip executes multiple applications simultaneously and new applications can be inserted incrementally. In this case, the algorithm should also take the current on-chip topology into account when optimizing the topology for a new application.

**CMP workload considerations.** As mentioned before, we should guarantee the existence of at least one path between all active source-destination pairs when running CMP workloads. In addition, these applications distribute the traffic load evenly across the NoC, so the bandwidth demand between all source-destination pairs is roughly the same. In this case, we can take advantage of reusing already found paths to reduce the computation load of the path finding algorithm. For example in Figure 3, assume a CMP workload activates the cores shown in dark red. Starting the algorithm by node X, some virtual long links are established (solid blue arrows) to connect X to all other nodes. In this figure, node D reverts to a router because by considering it as a bypass switch, node X cannot connect to all other active nodes along a shortest path. When node X is selected in the second iteration of the algorithm, it reuses the existing paths to A, B, F, and G that are constructed in the previous iteration for X. Y can also establish a long link to X and then access the nodes D, E, and H through the paths already established for X.
pipelined wormhole routers [8] with 8-flit buffers and two virtual channels per port. Those experiments that involve random parameters are repeated ten times and averaged out to produce final results.

We evaluate our scheme using several CMP applications consisting of Splash [13] and commercial workloads. The commercial benchmarks include Apache [1] and Software Testing (ST) program from the CloudSuite benchmarks [14]. We have extracted the traffic traces of Apache and ST for a 16-node CMP using Flexus [15] and feed them to the network simulator as the input traffic. We also use the traffic traces of a range of Splash programs, each with 49 parallel tasks. Apache and ST are tested on an 8×8 CMP, where the 16 cores that are assumed to be customized for their tasks are randomly distributed throughout the network. Similarly, Splash programs run on a 12×12 network. In these experiments, a customized topology is constructed for active cores. Figure 4 shows the average energy per flit and packet latency for the reconfigurable NoC presented in [7] (RecNoC, hereinafter) a conventional NoC (introduced earlier in this section), a conventional NoC that fit the application size, and the proposed reconfigurable NoC.

The proposed reconfigurable NoC outperforms the baseline across all benchmarks, reducing latency by as much as 16%, and energy per flit by up to 12%. It also operates at latencies very close to that of the RecNoC, while imposing no area overhead. For the selected NoC parameters, the area overhead of the RecNoC over a baseline NoC is 31%. The energy consumption of the proposed NoC is less than RecNoC, mainly because the static power of the additional hardware of RecNoC is removed.

Figure 4 also compares the energy and latency of the different applications on the proposed NoC with the energy and latency that would be obtained on a NoC with the same size as the application. As results show, the proposed NoC can bridge the performance gap between the large NoCs in dark silicon era and NoCs that fit the application size.

**Sensitivity to ratio of active to dark cores.** Figure 5 studies the effect of the ratio of active to dark cores on the achievable performance. The figure shows latency improvement of the proposed NoC over the baseline for different number of active cores in a 10×10 NoC. The active cores are selected randomly with a uniform distribution and each active core generates packets with a poison distribution and sends them to random active destination cores. The packet injection rate is set in such a way that the networks operate close to (and below of) the saturation point. As the figure shows, performance gains reduce with increase in active core count. The reason is that a small fraction of dark cores leave less unused routers to be used for topology customization. Nonetheless, even for the case of 80 active cores, the proposed NoC shows 4% improvement in latency over the baseline.

**5. Conclusion**

Managing on-chip communication resources in a many-core system is a fundamental challenge. Dark silicon-aware reconfigurable NoC, proposed in this paper, could leverage the routers of the dark portion of the chip to customize the NoC topology for the cores that are simultaneously powered up, primarily as a way to improve communication latency and energy consumption. The new topology, which was obtained by bypassing the routers associated with dark cores to directly connect remote active routers, could reduce the NoC average message hop count and hence, reduce communication latency and power consumption.
For a given workload, we reconfigure the inter-core connectivity among the active cores to obtain an application-specific irregular topology, optimized for the workload traffic characteristics. An evaluation of the proposed reconfigurable NoC, using both CMP and SoC workloads, shows that the proposed reconfiguration scheme improves network resource usage, energy, and delay as compared with state-of-the-art NoCs, while requiring only minor changes to the router and network architecture.

6. References