Energy Efficient On-Chip Communications Implementation Based on Power Slacks

Xiao-Yu Xia, Wen-Ming Pan, and Jia-Chong Kan

Abstract—The quest for energy efficiency has growing importance in high performance many-core systems. However, in current practices, the power slacks, which are the differences observed between the input power budget and the actual power consumed in the many-core systems, are typically ignored, thus leading to poor energy efficiency. In this paper, we propose a scheme to effectively power the on-chip communications by exploiting the available power slack that is totally wasted in current many-core systems. As so, the demand for extra energy from external power sources (e.g., batteries) is minimized, which helps improve the overall energy efficiency. In essence, the power slack is stored at each node and the proposed routing algorithm uses a dynamic programming network to find the globally optimal path, along which the total energy stored on the nodes is the maximum. Experimental results have confirmed that the proposed scheme, with low hardware overhead, can reduce latency and extra energy consumption by 44% and 48%, respectively, compared with the two competing routing methods.

Index Terms—Adaptive routing, dynamic programming network, networks-on-chip, power slack.

1. Introduction

Energy efficiency has become a major consideration in high-performance many-core systems. A large number of many-core systems, whether they are used in datacenters or in embedded systems, can be powered by energy harvested from the environment. To maximize the energy efficiency, numerous dynamic power management approaches have been proposed to allocate the power budget adaptively to the cores, by means of voltage/frequency scaling or power gating. It has been observed that these management-enabled systems tend to create many power slacks which are defined as the difference between the input power budget (either scavenged from the environment or from the power supply) and the actual power consumption of cores. Fig. 1 shows the power budget and actual consumption of a solar-powered many-core system that employs the dynamic power management methods: dynamic power partitioning and control (DPCC) used in [3]. The solid line is the power budget scavenged from the solar energy and the dashed line is the actual power consumption of the system. In this system, as shown in Fig. 1, the power slacks are totally discarded leading to lower energy efficiency.

Actually, the wasted energy from the unused power slacks can be recycled to power the on-chip communications. Power slacks can be stored into miniaturized on-chip energy storage devices, like supercapacitors or micro-batteries. This stored energy can be used to power the communications; by doing so, the energy drain from external power sources like batteries can be minimized, which brings the benefits of energy saving and battery longevity. One possible approach to store and exploit the power slack will be illustrated in the following example.

Energy from power slacks will be distributed and stored at the tiles beforehand, and data packets/flits can be routed following a power slack aware routing scheme. In the example shown in Fig. 2, energy recovered from power slacks are stored in the supercapacitors at each node. The color map represents the relative quantity of the stored energy. When a data transmission session is initiated between a source and a destination node, the path along which the nodes have the highest accumulated energy recovered from the power slacks and less congestion will be selected for routing the data packets. In this way, it can be seen that little, at the best case, even zero energy will be drawn from the power rail. In a simple term, energy spent on communications can be substantially reduced by exploiting the power slacks.

This novel technique of exploiting power slacks to power on-chip communications is referred as exploiting power slack (EPS) in this paper. EPS is a vast departure from the existing popular low-power design techniques,
especially in the category of dynamic power management, where power saving is achieved by squeezing out unnecessary consumption. In other words, EPS recycles the wasted energy in current many-core systems to improve the energy efficiency, and as a result, it helps reduce the energy needed from the external power sources. EPS is complementary to modern power management approaches.

2. Related work

In this section, various works concerning 1) on-chip low power techniques, 2) on-chip energy storage techniques, and 3) the adaptive routing algorithms are reviewed.

2.1 Low Power Techniques

There is a growing attention on how to improve the energy efficiency under a given power budget in a many-core system. Traditional run time power management approaches using frequency/voltage scaling\(^3\), power gating\(^7\), and so on, are still applicable at core and/or system levels. These techniques are particularly appreciated when they are applied to certain on-chip modules/resources that consume disproportionately large amount of chip power. One good example is the on-chip networked interconnect, often referred as network-on-chip (NoC), whose power consumption alone contributes a large portion of the total system power consumption. Many efforts have been dedicated to reduce the power consumption of NoC, by shutting down the under-utilized sub-networks\(^8\), or by passing routing pipeline stages\(^9\), etc. All these low-power techniques, however, do not directly deal with power slacks found in many-core systems, so our proposed method is complementary to these approaches.

2.2 On-Chip Energy Storage

Numerous on-chip micro-supercapacitors have been proposed\(^5\) and even commercialized. Micro-supercapacitors can store energy based upon either the electrochemical double layer (EDL) effect, or the fast surface redox reactions (pseudo capacitance) that take place at the interface of electrodes and electrolytes. These micro-supercapacitors exhibit fast charge and discharge rates (compared with rechargeable batteries) and long cycle lives (millions of cycles), and their energy density is getting close to that of batteries with the advance of using advanced nanostructured materials. In this paper, we assume that the supercapacitors are used to store energy from power slacks, and the energy stored in supercapacitors is sufficient to power the on-chip communications.

2.3 On-Chip Routing Algorithms

Routing algorithms can be categorized from different angles. The first angle is from the cost functions which can be the congestion level\(^6\), temperature\(^10\),\(11\), and fault information\(^12\),\(13\), etc. A different kind of classification could be based on adaptiveness. And the routing algorithms are divided to be either adaptive or deterministic. In adaptive routing algorithms\(^14\), the output channels with the max/min cost are selected at each router. DPN\(^11\) can be used to find globally optimal paths with respect to the cost functions. Following the same philosophy, the local stored energy recovered from power slacks can also be used as the cost function in finding the paths with the maximum stored energy.
3. EPS Approach

3.1 Overview

Fig. 3 shows the overview of the EPS approach whereas each tile has an energy storage element. The EPS framework consists of two major aspects: 1) power slack distribution and storage, and 2) power slack aware routing. Excessive energy from the power slack needs to be stored at the supercapacitor attached to each tile. During the data transmission phase, each participating router will choose the output to ensure that the path has the maximum energy stored in supercapacitors.

![Figure 3: Overview of the EPS approach. Each tile is augmented with a supercapacitor-based on-chip energy storage element](image)

3.2 Power Slack Distribution and Storage

Energy that is collected from power slacks is distributed, through a power grid, in each tile’s supercapacitor and stored there. In the following, we will first introduce the power grid and the power/energy models, then we will present the supercapacitor model.

A. Power Grid and Power/Energy Models

Each tile or router is equipped with an energy storage unit and realized by using an on-chip supercapacitor. Fig. 4 shows the equivalent circuit of one tile in the power delivery network together with an energy storage unit, where \( C_{SC} \), \( C_{ij} \), \( L_{ij} \), and \( R_{ij} \) are the lumped equivalent capacitor, inductor, and resistor between node \( i \) and its adjacent node \( j \).

![Figure 4: Equivalent circuit of one node in the power delivery network together with the energy storage element](image)

The current demand of load \( i \) in Fig. 4 at each node can be modeled as

\[
I_{\text{load}}(k) = I_{\text{atk}}(k) + I_{r}(k) \tag{1}
\]

where \( I_{\text{atk}}(k,t) \) and \( I_{r}(k,t) \) are the equivalent current demands of the link and router at cycle \( k \).

Let \( SW \) be a vector of size \( n \) (the number of link wires) with each element \( sw \), taking values of 0 and 1, \( i=1,2,\ldots,n \). The current demand of a link can be expressed as

\[
I_{\text{atk}}(k) = \sum_{k=1}^{n} \sum_{i=1}^{n} M_{k,i}^Tsw_{l,i,k} \tag{2}
\]

where \( I_{\text{atk}} \) is the current of wire \( i \in \{1,2,\ldots,n\} \) for the link at cycle \( k \), \( M \) is the decoupling transformation matrix, and \( sw_{l,i,k} \) is the switching direction of wire \( i \) at cycle \( k \).

To determine the current need of a router, Let \( \Psi = \{\text{RECEIVE, ROUTE, FORWARD, STANDBY}\} \) be the set of micro-architectural-level processes that can be executed by a router. These processes deal with receiving a flit, computing a route, forwarding a flit, and idling (leakage power), and their energy consumption is determined as follows, respectively.

The energy of the RECEIVE process is the energy required for storing the flits to the input buffer.

During a ROUTE process, energy is consumed for route computation, which is only performed for header flits (in the wormhole switching).

The total energy consumption of a FORWARD process is the energy spent on data retrieval from the input buffer, switch allocation, switch traversal, and link traversal.

The total energy consumed by router \( r \) at cycle \( k \), \( E_r(k) \), can be expressed as

\[
E_r(k) = \sum_{\psi \in \Psi} E_\psi(\alpha_\psi(r,k) + P^r_\psi \Delta t) \tag{3}
\]

where \( E_\psi \) is the energy required by the router circuit to execute process \( \psi \), and \( \alpha_\psi(r,k) \) is the number of instantiations of process \( \psi \) in router \( r \) at cycle \( k \). The term \( P^r_\psi \Delta t \) accounts for energy contributed by leakage power, where \( P^r_\psi \) is the leakage power and \( \Delta t \) is the cycle time.

Thus, the amount of current needed by the router \( r \) is

\[
I_r(k) = E_r(k)/V_{DD} \tag{4}
\]

The current demand of load \( i \) can be derived by tools in [14] which combines the router micro-architectural level power model and an NoC event driven simulator.

B. Energy Storage Device Model

The energy storage unit in Fig. 4 could be implemented by on-chip supercapacitors which can be essentially modeled as a capacitor \( C_{SC} \),

\[
C_{SC} = \varepsilon_0 \varepsilon_A / d \tag{5}
\]

where \( \varepsilon_0 \) is the permittivity of the vacuum, \( \varepsilon_a \) is the relative permittivity of the electrolyte, \( d \) is the double-layer.
thickness and it is an electrolyte-dependent parameter, and $A$ is the overall surface area of the electrodes.

The remaining energy stored in the supercapacitor at time point $t$ is

$$E_{PS}(t) = V_{DHO}(t)/2$$  \hspace{1cm} (6)

where $Q(t)$ is the charge remaining in the supercapacitor.

The power delivered to each node can be derived by (1) to (5). When the stored energy is insufficient to forward a flit, the router might either hold the flit in its buffer, or still forward it, but with the energy drain from an external power source (e.g., external batteries), depending on the task criticality. Thus, to reduce the leakage power when holding a flit in a buffer, two voltage levels, $V_{DHO}$ and $V_{DDL}$, are assumed here: the former with a higher voltage level is for normal operations and the latter for the low-power mode to reduce the leakage. The low-power model can be used to hold flits in buffers when the stored energy is not sufficient to forward a flit. Different voltage islands can still talk to each other using circuits like level shifters\[15\]; their energy consumption and delay are denoted as $E_{LS}$ and $t_{LS}$.

C. Power Slack Aware Routing

Once the energy recovered from the power slack is stored at each tile, the routing algorithm can exploit the stored energy to forward packets to minimize extra energy consumption. To find a globally optimal path in that sense, DPN[6] is used.

D. DPN

DPN is a parallel implementation of Bellman’s equation with optimization. It has a graph of states $G(V, E)$, where each vertex $v \in V$ is a state and each edge $e = (u, v)$ is the transition between the states, where $u$ is a vertex in the graph, too. Each transition is associated with a cost-to-go function $C_{u,v}$ and each vertex has an expected reward $C_{MAX}(v, D)$ to reach a destination $D$. The optimization or decision-making problem can be expressed in a recursive form at each state as follows:

$$C_{MAX}(v, D) = \max_{\forall u, \text{an edge } e \text{ exists between } (u, v)} \{C_{u,v} + C_{MAX}(u, D)\}. \hspace{1cm} (7)$$

The cost of a path $PATH_{3,5} = s_1 \cdots s_7 \cdots s_9 \cdots s_2$ from a source node $S$ to a destination node $D$ is the sum of the costs of its constituent edges, i.e.,

$$C_{MAX}(PATH_{3,5}, D) = \sum_{i=1}^{r-1} C_{s_i, s_{i+1}}.$$ 

The optimal cost from the state $S$ to $D$ can be represented as any paths with the maximum cost:

$$C'_{MAX}(S, D) = \max_{\text{PATHs } S \rightarrow D} \{\sum_{i=1}^{r} C_{s_i, s_{i+1}}\}. \hspace{1cm} (8)$$

where $PS(D)$ is the set of all paths from the source node $S$ to the destination node $D$.

E. Finding Maximum Reward Paths Using DPN

To employ DPN to find the paths with the maximum stored energy, the NoC can be denoted as a connected graph $G(V, E)$ where each vertex $v \in V$ is a tile and each edge $e = (u, v)$ is a link between the two tiles, where $u$ denotes a tile in the graph, too. Each edge is associated with a cost-to-go function $C_{u,v}$, which can be defined as

$$C_{u,v} = E_{PS} - \gamma BL_{u,v}, \hspace{1cm} (9)$$

where $E_{PS}$ is the energy converted from the power slack and stored in tile $u$, $BL_{u,v}$ is the buffer level of the link between tiles $u$ and $v$ measuring the congestion level, and $\gamma$ is a user defined weight that is employed to balance between the energy efficiency and low latency.

To find a path with the maximum stored energy and less congestion (i.e., the maximum cost), the dynamic programming equation (7) can be calculated within $N$ iterations\[6\]. For each vertex $v$, calculate the $C_{MAX}(v, D)$ value in (7) with the $C_{u,v}$ defined in (9).

The calculation of (7) at each vertex leads to finding the maximum cost path (optimal solution) under the power constraint in $N$ steps, given as (8). So, the optimal decision at each vertex $v$ along the maximum cost path to reach the destination $D$ can be obtained as

$$u^* = \max_{\forall u, \text{an edge } e \text{ exists between } (u, v)} \{C_{u,v} + C''_{MAX}(u, D)\}. \hspace{1cm} (10)$$

Fig. 5 presents the algorithm operations required for updating the routing directions at each tile. Each vertex $v$ receives the different maximum costs to reach the destination $D$ from the adjacent vertices. In the main loop, the optimal cost will be computed for reach destination $D$. The corresponding optimal output direction is selected to update the routing table. When the computation of $C_{MAX}(v, D)$ at vertex $v$ is done, it will be sent to all the adjacent vertices. The two for-loops are implemented in hardware with a parallel architecture and the computational delay complexity can be reduced to be linear.

Once an output direction with the maximum cost is found at each router, the flit can be forwarded to that output channel. If the stored energy is not sufficient to forward a packet, the flit is held in the buffer operating at the low-power mode. Each buffer is associated with a timer $T$, which could be set to different values, e.g., 10, 50, and 100 cycles. Once the timer is overflow, the flits will be forwarded using extra energy from the power supply system (e.g., external battery). Proper setting of $T$ strikes a balance between performance and extra energy consumption needed, i.e., larger $T$ indicates longer latency but a lower extra energy requirement.

The dynamic programming unit at each router is the same as that detailed in [6] and [11] with the total area and power overheads less than 2%\([11]\) to a router. To avoid deadlocks, the physical network is separated into two virtual networks, $VN_0$ and $VN_1$, and each of these virtual
networks does not include any cyclic path by restricting some turns. VNₙ does not allow packets to turn to the north, while VNₙ does not allow packets to turn to the south. In addition, any packet only can traverse one virtual network. These routing policies will ensure that no cycle will be formed at the network level.

In this routing scheme, packets need to be distinguished by the virtual network that they shall pass through. One bit is thus added in the packet header to identify the virtual network designated for a packet. At the source router, this bit is set for the packets. If the destination is in the north to the source node, the packet follows VNₙ, while the packet follows VNₙ if the destinations are in the south to the source node. For the destinations that have the same X-coordinate as the source, the random virtual networks can be chosen. All the intermediate nodes sitting in a routing path should never flip the virtual network identification bit; they just simply forward the packet to the next node on the same virtual network.

There are various on-chip micro supercapacitors made of different materials. The capacitance of the micro supercapacitors ranges from 80 mF/cm² to 90 mF/cm² up to a few hundreds of microfarads per square centimetre. Since the energy density is directly proportional to the capacitance, we select the capacitance value to be 90 mF/cm² in the experiments. The area of the supercapacitors is assumed to be 1×1 mm². The power slacks are represented as time series traces, generated from a uniform distributed random process with different expectations.

### 4.2 Performance Evaluation

In the experiments, we will first study the impact of the weight in the cost function (see (9)) and timer T (Sub-Section 3.2 E). Then two competing routing algorithms, XY routing and odd-even (OE) routing will be compared against EPS.

#### A. Impact of Weight Coefficient in Cost Function

Fig. 6 (a) and Fig. 6 (b) show the latency and extra energy consumption (from the external power rail) of EPS with different weight values included in the cost function (the γ parameter in (9)). The timer T is set to be 0 and the injection rate is 1 flit/cycle. As mentioned in Sub-Section 3.2 E, the weight coefficient γ is a parameter that helps balance between latency and extra energy consumption. In general, a larger value of γ provides lower latency and a smaller γ implies lower energy consumption. This has been confirmed by the results shown in Fig. 6 (a) and Fig. 6 (b). In following experiments, we will use a weight γ = 5.

### 4. Experimental Results

#### 4.1 Experimental Setup

Table 1 tabulates the simulation configuration. Noxim⁶⁶, a cycle accurate NoC simulator, is extended to simulate an 8×8 NoC system. The power grid model is integrated with the NoC simulator. We measure the average packet latency and extra energy consumption in the experiments. The dynamic and static power of the 5×5 router is estimated using Synopsis Design Vision 2009.06 with Taiwan Semiconductor Manufacturing Company (TSMC) 65 nm CMOS technology library.

<table>
<thead>
<tr>
<th>Items</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network size</td>
<td>8×8</td>
</tr>
<tr>
<td>Packet length (flits)</td>
<td>8</td>
</tr>
<tr>
<td>Flit size (bits)</td>
<td>64</td>
</tr>
<tr>
<td>VC depth</td>
<td>8</td>
</tr>
<tr>
<td>VC number</td>
<td>2</td>
</tr>
<tr>
<td>Distribution of the destinations</td>
<td>Uniform</td>
</tr>
<tr>
<td>Distribution of the power slack</td>
<td>Uniform</td>
</tr>
</tbody>
</table>

Fig. 5. Algorithm for updating the routing directions at each tile.

Fig. 6. Effect of buffer level weight coefficient γ on (a) packet latency and (b) extra energy consumption from external power sources.

#### B. Impact of Timer

The timer T also balances between the latency and energy consumption: a larger T put an emphasis on low energy consumption while a smaller value of T brings shorter latency. Fig. 7 shows the latency and energy consumption under different T values, where EPS-T indicates the timer with a value T. As shown in Fig. 7 (a), when T increases, the latency increases accordingly, i.e., the latency of EPS-100 (holding flits for at most 100 cycles in
case of insufficient stored energy) is about 2 times over that of EPS-0 (forwarding the flits immediately with extra energy) at the packet injection rate of 1 flit/cycle. On the other hand, Fig. 7 (b) confirms that larger $T$ will tend to help reduce the energy consumption. For example, the extra energy request of EPS-0 is about 1.7 times over that of EPS-100 at the injection rate 1.5 flit/cycle.

![Fig. 7. Latency and energy consumption under different $T'$ values: (a) latency and (b) extra energy consumption with different timers.](image)

**C. Performance Comparison of EPS against Other Routing Algorithms**

In this subsection, the performance and extra energy consumption of EPS are compared against two competing routing algorithms, XY routing and OE routing algorithms in terms of latency and extra energy consumption. The values of weight coefficient $\gamma$ and timer $T$ are set to be 5 and 0, respectively. The average power slack is set to be 10 W and 40 W, respectively, in Fig. 8. From Fig. 8, it can be seen that the EPS algorithm has the lowest latency among the three routing algorithms. For example, in Fig. 8 (a), the latency to XY and OE routing algorithms are 1.8 times and 1.5 times over that of EPS. EPS outperforms OE because EPS uses DPN which can find the globally optimal (less congested) path, while OE selects output channels only with the local knowledge of congestion from adjacent routers.

![Fig. 8. Packet latency to different average power slacks: (a) 10 W and (b) 40 W.](image)

**D. Summary**

From the above experiments, some conclusions can be drawn.

1) The buffer level weight coefficient $\gamma$ (refer to (9)) and timer $T$ relate to a balance between the latency and energy consumption, i.e., a larger weight or smaller $T$ corresponding to lower latency, while a smaller weight or larger $T$ indicating lower energy consumption.

2) Comparing with XY and OE routing, the proposed algorithm EPS has the lowest latency due to its capability of finding globally optimal paths. When the power slack is high, the proposed algorithm EPS can significantly reduce

![Fig. 9. Extra energy consumption when the power slack is (a) 10 W, and (b) 40 W.](image)

**Fig. 9. Extra energy consumption when the power slack is (a) 10 W, and (b) 40 W.**

![Fig. 10. Latency and extra energy request with the traces of four benchmarks from PARSEC.](image)

**Fig. 10. Latency and extra energy request with the traces of four benchmarks from PARSEC. (a) latency reduction of EPS over OE and XY, and (b) extra energy of EPS, OE and XY with traces from three benchmarks.**
extra energy requests over the other two compared routing algorithms.

5. Conclusions

The power slack has been consistently ignored in most modern many-core systems. Exploiting or recycling the wasted energy available from the power slack can certainly improve the energy efficiency and reduce the extra energy request from the external power rail. In this paper, we have proposed an approach, named EPS, to convert and store the energy recycled from the power slack to power the on-chip communications. Essentially, EPS uses on-chip energy storage devices to store the power slacks at each node. Paths with the highest stored energy can be found by a DPN based routing algorithm. Experiments have confirmed that EPS can reduce latency and extra energy consumption over two existing routing algorithms, XY routing and OE routing algorithms, by up to 48% and 44%, respectively.

References


Xiao-Yu Xia was born in Hunan, China in 1987. He received the B.S. and M.S. degree from Wuhan University, Wuhan in 2010 and 2012, respectively, both in electrical engineering. He is currently the research assistant at Guangzhou Institute of Advanced Technology, Chinese Academy of Science. His research interests include networks-on-chip and parallel computing.

Wen-Ming Pan was born in Guangdong, China in 1982. He received the B.S. and M.S. degrees in electronic engineering from Jinan University, Guangzhou in 2004 and 2007, respectively. He works as a research assistant with Guangzhou Institute of Advanced Technology, Chinese Academy of Sciences. He has been engaged in the FPGA design research for years. His research interests include networks-on-chip and parallel computing.

Jia-Chong Kan was born in Liaoning, China in 1990. He received his M.S. degree in circuit and system from South China Normal University, Guangzhou in 2014. He works as an intern with the Guangzhou Institute of Advanced Technology, Chinese Academy of Sciences. His research interests include image processing and parallel computing.