Int. J. Appl. Math. Comput. Sci., 2011, Vol. 21, No. 2, 385–399
DOI: 10.2478/v10006-011-0029-7

ENERGY CHARACTERISTIC OF A PROCESSOR ALLOCATOR AND A NETWORK–ON–CHIP

DAWID ZYDEK *, HENRY SELVARAJ *, GRZEGORZ BOROWIK **, TADEUSZ ŁUBA **

* Department of Electrical and Computer Engineering
University of Nevada, Las Vegas, 4505 S. Maryland Parkway, Las Vegas, NV 89154-4026, USA
e-mail: {zydekd,selvaraj}@unlv.nevada.edu

**Institute of Telecommunications
Warsaw University of Technology, ul. Nowowiejska 15/19, 00–665 Warsaw, Poland
e-mail: {gborowik,luba}@tele.pw.edu.pl

Energy consumption in a Chip MultiProcessor (CMP) is one of the most important costs. It is related to design aspects such as thermal and power constrains. Besides efficient on-chip processing elements, a well-designed Processor Allocator (PA) and a Network-on-Chip (NoC) are also important factors in the energy budget of novel CMPs. In this paper, the authors propose an energy model for NoCs with 2D-mesh and 2D-torus topologies. All important NoC architectures are described and discussed. Energy estimation is presented for PAs. The estimation is based on synthesis results for PAs targeting FPGA. The PAs are driven by allocation algorithms that are studied as well. The proposed energy model is employed in a simulation environment, where exhaustive experiments are performed. Simulation results show that a PA with an IFF allocation algorithm for mesh systems and a torus-based NoC with express-virtual-channel flow control are very energy efficient. Combination of these two solutions is a clear choice for modern CMPs.

Keywords: CMP, PA, energy model, processor allocation.

1. Introduction

The tiled Chip Multi-Processor (CMP) architecture, consisting of many parallel Processing Elements (PEs) integrated on one die, is the dominant trend in parallel processing systems. Jobs created to run on these systems are parallel programs containing many tasks that communicate with each other. Each of the tasks runs on a separate PE (core). Therefore, the incoming job specifies the number of PEs needed to execute the job. The selection of a subset of PEs required for a given job is called a processor allocation problem (Chmaj *et al.*, 2004; Rose *et al.*, 2007; Zydek and Selvaraj, 2009) and it is done by the Processor Allocator (PA) (Zydek and Selvaraj, 2010). Efficient processor allocation should be characterized by

- high utilization: processor allocation must provide maximal resource utilization;
- low overhead: allocation techniques must be fast and deliver low overhead;
- scalability: algorithms must support systems with thousands of nodes without any bottleneck;
- recognition completeness: it means the ability to find free subgrids for incoming jobs, if such a free subgrid exists.

The jobs are allocated in such a manner that they do not overlap with each other and, if allocated, they run until completion.

The resource utilization parameter in a CMP system can be improved by a better job scheduling process (Krueger *et al.*, 1994; Mohapatra *et al.*, 1993). Job scheduling is done by a Job Scheduler (JS) that deals with the selection of the job to be executed next. In this paper, the authors focus on a PA and its energy consumption. The FCFS (First Come First Served) strategy is assumed as a job scheduling policy.

Besides a good processor allocation strategy, efficient communication among PEs is also an important factor in high performance multi-core processors. Moreover, technology scaling enables integration of billions of transistors on a chip that keeps increasing the number of PEs

in a CMP. With the increase in the number of cores, the system performance begins to be affected by the on-chip interconnect—currently common bus structures become a limiting factor for performance, space and energy consumption (Dally and Towles, 2001). In order to overcome the disadvantages of bus structures, the idea of a shared-medium approach with a Network-on-Chip (NoC) is proposed.

Among several NoC topologies developed for CMPs, low-dimensional networks like 2D-mesh (k-ary 2-mesh) and 2D-torus (k-ary 2-cube) represent better performance (higher throughput and lower latency) in comparison with high-dimensional networks (like, e.g., high dimensional k-ary n-cubes and meshes) (Dally, 1990). The topologies match also very closely with the physical layout of the die (Jayasimha *et al.*, 2006), which makes them the basic topologies for current and future CMPs. 2D-torus and 2D-mesh networks have uniformly short wires that allow high-speed communication and the use of locality between nodes. Both topologies (especially torus) contain many redundant paths between nodes and ensure good load balance. The wrap-around channels in a torus network may cause problems—they are long (require repeaters) and slower than other channels. This problem can be avoided by folding a 2D-torus (Zydek and Selvaraj, 2009). In this paper, the k-ary 2-mesh and k-ary 2-cube topologies of an NoC are considered.

For 2D-mesh and 2D-torus topologies, the number of PEs requested by an incoming job is the size of the requested subgrid. There are two major approaches to processor allocation: contiguous and non-contiguous. In contiguous processor allocation, the processors allocated to a job are physically adjacent and have the same topology as the NoC. In the non-contiguous allocation strategy, the job can be executed on multiple disjoint smaller subgrids that allows dividing a job rather than waiting until a single subgrid of the requested size becomes available. But it can generate global traffic, which we would like to avoid in the NoC. Thus, the authors have chosen to focus on the contiguous strategy.

A lot of research has been done to increase the efficiency of processor allocation algorithms for 2D-mesh topology (Chmaj *et al.*, 2004; Yoo and Das, 2002; Zydek and Selvaraj, 2010). Similarly, 2D-torus networks are found in many works (Zydek and Selvaraj, 2011; Zydek *et al.*, 2010).

Zydek and Selvaraj (2009) presented the idea of hardware implementation of the JS and the PA, and their integration on one die together with the processing elements. In another work (Zydek and Selvaraj, 2010), new allocation algorithms for a 2D-mesh NoC were presented. Additionally, the hardware implementation and synthesis results of the PA for a mesh-based CMP were shown. A torus topology can be found in a recent paper by Zydek and Selvaraj (2011), where allocation techniques and their comparison to the mesh schemes are discussed. Finally, in the research by Zydek *et al.* (2010), the synthesis results of the PA driven by the schemes for a torus network are presented together with their mesh counterparts. In this paper, we study the energy aspects of an NoC-based CMP with an integrated PA. The novel NoC architectures are reviewed, together with the most efficient PAs. In both cases, two-dimensional torus and mesh topologies are considered. An energy model is proposed for the NoC under discussion. Energy consumption of PAs is estimated using the synthesis results presented by Zydek and Selvaraj (2010) as well as Zydek *et al.* (2010). The performance of the CMP is evaluated using a simulation environment developed by the authors that includes the energy model analyzed in the paper.

The reminder of this paper is organized as follows: Section 2 contains a short overview of the NoC-based CMP and novel NoC architectures. Information about the PA and processor allocation techniques is in a Section 3. The energy model for NoCs and the energy characteristic of PAs are described in Section 4. Experimental results are presented in Section 5, while final remarks in Section 6.

2. NoC-based chip multiprocessor

CMP architectures with an NoC are currently the most advanced multiprocessor architectures. They differ from standard CMPs with few cores on the same die, where regular buses are used as interconnects. They also differ from older multiprocessor systems, where PEs were placed on one main board and connected by buses (or a network) on the board, not on the chip. The structure of the chip, as is in CMPs, has interconnects and processing elements significantly closer than in off-chip networks. This implies better latency and bandwidth performance, but properties such as power, area and cost restrictions become crucial. Also, the complexity of designing efficient and scalable on-chip communication solutions increases together with the number of PEs integrated on a single die. To provide scalability and effective use of resources available in ULSI technology, a tiled architecture is proposed (Taylor *et al.*, 2002), cf. Fig. 1. The PEs are connected by an NoC that replaces the traditional on-chip buses. The chip area is divided into square tiles. Each tile contains networking elements (router, PE network interface, network channel) and PEs (processor, cache memory, etc.). Each tile is connected to a network Router (R) by a PE network interface. Communication among tiles is done by sending messages over the network using tiles' network interfaces (routers). In order to achieve high performance and efficiency, two components of the processor management system, i.e., the PA and the JS, are proposed to be implemented in hardware and placed as a tile on the same die as the PEs in the CMP (Zydek and Selvaraj, 2009), cf. Fig. 1.

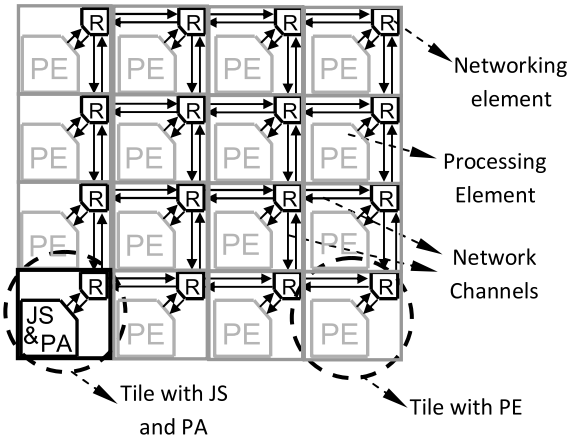


Fig. 1. Tiled CMP (4×4 2D-mesh) with an integrated job scheduler and processor allocator.

The JS receives a request and places the job in the queue, which is controlled in a particular way (in this paper, in an FCFS fashion). The scheduled job is moved to a PA, which assigns the job to available PEs according to the allocation algorithm. As decided by the PA, PEs are reserved and operands are sent to PEs to process them. After execution, PEs return the results and a "release" message is sent to the PA, which updates the status of processors. Operands and results are sent through I/O ports, which for simplicity are not shown in Fig. 1. All data (control messages, operands and results) are sent by the implemented NoC. The PA and the JS are implemented using the same networking elements (R) like PEs that give communication possibility.

2.1. Network topology. Topology is one of the main properties that characterizes the NoC. It describes the layout and structure of the nodes and links on the chip (Dally and Towles, 2004). The degree of a node decides on the number of ports in the router. Thus, a node's degree has impact on the complexity (power, energy and area consumption) of the router. The smaller the value of the degree, the lesser the cost, while its homogeneity leads to uniform routers. It implies a desire of a small and fixed degree. The diameter of a network is the largest, minimal hop count over all pairs of nodes in the network. The diameter has direct impact on the latency of the network. Lower latency gives shorter distances, which imply a need for a smaller network diameter. Topology has significant influence on flow control and routing algorithms. Simple and regular topology (like, e.g., 2D-mesh and 2D-torus) reduces the complexity of the routing algorithm. An optimal topology is also characterized by its path diversity. Multiple minimal paths among nodes reduce the impact of defects in the manufacturing process. Path diversity also allows balancing the load across channels and makes the network more tolerant to faulty channels and nodes.

Whatever the topology, the network needs to be laid out in a die. Due to poor progress of "3D stacking" that is still under research, a 2D die is considered. This implies that, for networks of dimension higher than 2D, topological adjacency does not lead to spatial adjacency. Thus higher dimensional topologies have negative impact on wire delay and wiring density. These are possible reasons for the necessity for long wires. However, implementation of long wires can affect the operating frequency and power consumption. Furthermore, tiles can have more channels crossing at least one of their edges that can force the channel width to be less than required by the architecture. Besides on-chip embedding issues, low dimensional networks represent also better performance in comparison with their higher dimensional counterparts. At the same bisection, low dimensional topologies provide lower latency and higher throughput. Moreover, topologies of many dimensions require more and longer wires (Dally, 1990; Jayasimha *et al.*, 2006; Dally and Towles, 2004).

The 2D-mesh (k-ary 2-mesh) topology (Dally and Towles, 2004; Duato *et al.*, 2003) meets all properties described above. It is also an obvious and natural choice for a tiled architecture due to its close match with the physical layout of the die. A k-ary 2-mesh consists of $N = k^2$ nodes arranged in a 2-dimensional mesh with k nodes along the two dimensions. Every node in the middle of the mesh is connected to four neighboring nodes, while the nodes on the edges of the mesh are connected to two or three neighbors, which makes the mesh network degree slightly irregular (Fig. 2(a)). The nodes are addressed by a 2-digit radix-k address $\{a_1, a_0\}$, where a_1 and a_0 represent the node position in the first and second dimension, respectively.

The diameter of 2D-meshes is $2 \cdot (k-1)$ hops. The bisection is $2N/k$, which can offer wider channels and higher channel bandwidth for given bisection density. 2D-meshes are characterized by uniformly short wires that allow high speed operation without repeaters. Minimal paths between nodes from a logical point of view are also physically minimal, allows exploiting local traffic. Path diversity is good as it ensures better reliability and load balance. One of the main drawbacks of the k-ary 2-mesh is the lack of the same available bandwidth for every node—the bandwidth available to nodes at corners and edges is smaller while these nodes have a higher average distance from other nodes.

The 2D-torus (k-ary 2-cube) topology (Dally and Towles, 2004; Duato *et al.*, 2003) is achieved by enriching a 2D-mesh with additional channels that connect the external nodes in each row and column (Fig. 2(b)). Similar to the 2D-mesh, a k-ary 2-cube has regular physical arrangements that make it well suited for an on-chip layout.

A k-ary 2-cube consists of as many as $N =$

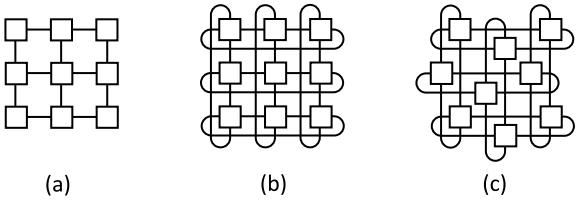


Fig. 2. 2D-mesh and 2D-torus networks: 3-ary 2-mesh (a), 3-ary 2-cube (b), folded 3-ary 2-cube (c).

k^2 nodes arranged in a 2-dimensional torus with k nodes along the two dimensions. Every node is connected to four neighboring nodes—a torus network is degree regular. Wraparound channels added in 2D-toruses double their bisection to $4N/k$ and decrease their diameter to $2 \cdot \lfloor k/2 \rfloor$ hops. They also provide edge-symmetry that helps to improve the load balance and reliability of the network. Wraparound channels are reasons for asymmetric channel lengths in a direct physical mapping of the 2D-torus. The wraparound channel has to be long enough to span the length of all k nodes. A long wraparound link can increase the propagation delay, which brings negative impact on latency. Also, it can require repeaters and decrease the operating frequency of all links. All these problems can be avoided by folding the torus as shown in Fig. 2(c). The folding keeps the topology untouched but eliminates the wraparound channels at the expense of doubling the length of the other channels. However, such a doubled channel possesses an acceptable latency characteristic and does not require repeaters.

2.2. Flow control. Flow control describes the allocation of NoC's resources (channel bandwidth, buffer capacity and control state) for packets traversing the network. The flow control policy decides if a packet should be dropped, blocked in place, buffered or rerouted. A well designed flow control allocates these resources effectively in order to get good bandwidth and low latency. There are two approaches to flow control:

- Problem of resource allocation: resources have to be assigned to each packet that traverses the network (e.g., routing algorithm determines which resources are allocated to packets).
- Problem of resolving contention: when an outgoing channel is requested by packets arriving on different inputs, the flow control mechanism has to allocate this channel to one packet and do something else with the others, e.g., block or drop.

Buffering data is more efficient than waiting to get network resources, like in circuit switching (Dally and Towles, 2004). It can be done in units of packets (store-and-forward and cut-through flow control) or flits (wormhole, virtual-channel or express-virtual-channel flow control). Flits are fixed sized, smallest units of information recognized by the flow control method, and grouped together to create a variable-length packet. The usage of flits significantly reduces the amount of storage in each node, which saves crucial space on the chip (Zydek *et al.*, 2008).

As can be deduced, in wormhole flow control (Dally and Towles, 2004; Dally, 1992), a packet is divided into a number of flits. A header flit (or flits) manages a route and the other flits (body and tail flits) follow the header in a pipeline fashion according to a path determined by the header flit. If the head flit meets a channel which is already busy, it is blocked together with the following flits (blocked flits can block many nodes along the established route). This blocking may occur because only the flits of one packet can use a channel (like actually we would send a packet through the channel), and only buffers are allocated on a flit-by-flit basis.

This disadvantage of wormhole flow control is eliminated in virtual-channel flow control that logically separates channels which share the same physical channel. It leads to the possibility of the existence of flits of many packets in the channel. Practically, Virtual Channels (VCs) are flit buffers associated with a single physical channel. By introducing VCs, packets are forwarded in the network over them, which separates the allocation of buffers from that of channels. A blocked packet blocks only the VC of a physical channel, but the other VCs can still use the physical channel (Fig. 3).

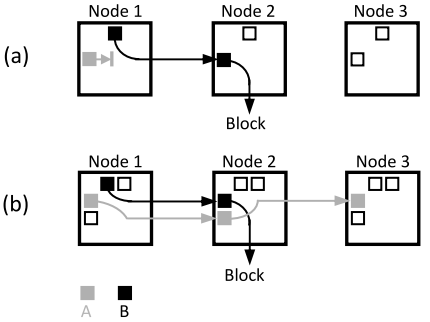


Fig. 3. Packet A is blocked behind B—wormhole flow control (a), packet A can pass packet B—virtual channel flow control (b).

The physical channel is blocked only when all its VCs are blocked. However, the probability of such a situation is lower than that of blocking with a wormhole technique. In this way, the utilization of physical channels and network throughput is higher. The way of implementing VCs requires careful analysis. In high traffic rates under uniform traffic, the increasing number of virtual channels per physical channel can raise performance, but, e.g., in the case of hotspot traffic, assigning deeper buffers to a smaller number of virtual channels gives better results

(Rezazad and Sarbazi-Azad, 2005).

Another, novel approach to flow control is the express-virtual-channel (Kumar *et al.*, 2007). The key idea of the technique is to provide virtual express lanes in the network, which allow bypassing intermediate routers by skipping the router pipeline. It is realized by introducing express buffers that define Express Virtual Channels (EVCs). By implementing EVCs, packets can be virtually bypassed through intermediate nodes. The virtual bypassing in a router forwards EVC flits as soon as they reach the router without any buffering and arbitration, which significantly reduces packet latency and router energy consumption. Express-virtual-channel flow control consists of two kinds of virtual channels at each port of a router: 1-hop regular VCs and k-hop EVCs that carry flits k-hops at a time. The flit is allowed to choose either a VC or an EVC depending on their availability and the path of the flit. When transmission is over a k-hop EVC, the flit is allowed to bypass the router pipeline at the next intermediate $k-1$ nodes.

In Fig. 4(b), 2-hop EVCs in 5-ary 2-mesh topology are shown and compared with regular VCs solution. In the VCs case (Fig. 4(a)) seven nodes are fully involved in the transmission, while in the structure with EVCs (Fig. 4(b)) the number of fully involved nodes is reduced to five (for two nodes the router pipeline is bypassed). EVCs are designed statistically at each router, and they are prioritized over normal virtual channels. Beside a lower latency (up to 84%) and a better throughput (up to 23%), EVCs reduce the router switching activity (a packet is going through a number of routers), limit the number of buffers, and reduce contention, which makes this solution energy (up to 38%) and area efficient (Krishna *et al.*, 2008).

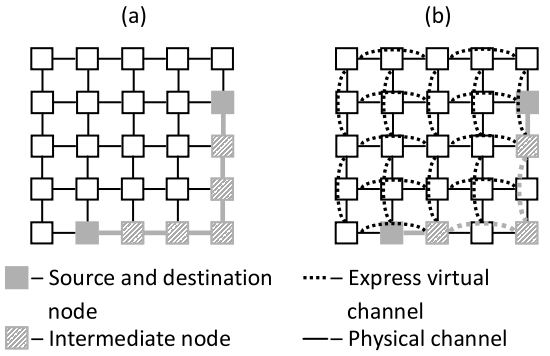


Fig. 4. Regular structure and example of transmission (a), the same transmission in a structure with express virtual channels (b).

2.3. Routing. Routing is the procedure of selecting a path from a source node to a destination node in a particular topology. Besides topology and flow control, a routing algorithm is also an important factor in the performance of the NoC. A good routing algorithm balances the load in the network channels, routes paths as fast as possible and is still able to work in the presence of faults. It should also be easily implemented in hardware. There are three main classifications of routing algorithms:

- Deterministic: always choose the same path between a pair of nodes. The balance of the load is very poor in this case, but they are commonly used due to easy implementation.
- Oblivious: route packets without considering the network's state (deterministic algorithms are a subset of oblivious).
- Adaptive: use information about the network's state (e.g., channel load information, length of queues for resources, etc.) to make routing decisions.

If the path selected by a routing algorithm is the shortest path between the source and the destination, the algorithm is said to be minimal. Using a minimal routing algorithm, every traversed channel brings the packet closer to the destination. In non-minimal routing, the chosen path can be longer, which allows reacting to current network conditions.

Another important property of routing algorithms is freedom from deadlocks and livelocks. A deadlock occurs when packets are waiting for each other in the cycle. A livelock is caused by a packet which proceeds through the network indefinitely and never arrives at its destination. A livelock is possible only for adaptive and non-minimal routing. If there is no limit of the maximum number of times, a packet can choose a non-minimal path, and remain in the network indefinitely. One approach to avoiding livelocks is to implement in the packet a field indicating its progress. It can be the number of times a packet has been routed through the non-minimal path. Once the specified value of non-minimal progress reaches a threshold (often called misroute value), only a minimal path can be chosen. Another approach to livelock avoidance is age-based priority filed in the packet. When a conflict between packets occurs, a packet with a higher priority (older) is privileged (Dally and Towles, 2004).

Deadlocks can be avoided by eliminating cycles in the resource dependence graph (Dally and Seitz, 1987). We can do it by providing some restrictions on routing. An example of such restriction is the well known deterministic algorithm—Dimension Order Routing (DOR), or the xy algorithm (Dally and Towles, 2004). DOR sends every packet from a source to a destination over exactly the same path. A path diversity offered by topology is ignored. Similarly, load balancing and reliability are very weak. These issues were addressed by Valiant and Brebner (1981), who proposed Valiant's oblivious algorithm. In the Valiant scheme, a packet sent from a source to a destination is first

sent from the source to a randomly chosen intermediate node and then from that node to the destination. It reduces the load of any traffic pattern (spatial distribution of messages in interconnection networks). However, the good performance given by randomization provides decreased locality. Better load balance can be also achieved by randomizing the order of dimensions in which the packet is traversed (Dally and Towles, 2004). At each node, either for DOR or Valiant routings, an x-first or a y-first direction is randomly chosen. Such enhanced algorithms are named here DOR Load Balanced (DOR-LB) and Valiant Load Balanced (Valiant-LB). DOR-LB provides a minimal, load balanced oblivious routing and preserves the locality.

Another approach to create routing algorithms that are deadlock free exploits VCs, which allows the design of highly adaptive algorithms. Actually, designing deadlock-free, fully adaptive routing algorithms without virtual channels is not possible (Dally and Towles, 2004). Very popular are hybrid solutions, which combine splitting network resources (virtual channels) with restricting the paths for packets. Algorithms such as 3P (Su and Shin, 1993), mesh_route (Boura and Das, 1994) and PFNF (Upadhyay *et al.*, 1997) are highly adaptive and need only two VCs per physical channel to ensure deadlock freedom.

2.4. NoC summary. In this paper, we investigate NoC architectures for CMPs with the following parameters:

- topology: k-ary 2-mesh and k-ary 2-cube;
- flow control: virtual-channel and express-virtual-channel;
- routing: DOR, Valiant, DOR-LB, Valiant-LB and adaptive.

The adaptive routing technique considered in the paper uses historical data of network resource usage in order to route packets. It is a hybrid, deadlock avoidance technique. In order to avoid livelocks, the misroute value is implemented.

3. Processor allocator

3.1. Preliminaries. The internal architecture of the PA may vary with the implemented allocation algorithm that leads also to different I/O structures. A detailed description of the PA structure that is considered in this paper can be found in the work of Zydek and Selvaraj (2010). The allocation technique used by a PA is the most important part of the PA. We consider algorithms for 2D-torus and 2D-mesh systems. The 2D-torus topology, denoted by $T(w,h)$, consists of $w \times h$ nodes arranged in a $w \times h$ 2D grid. Each node in the torus refers to an on-chip processor. The node in column c and row r is identified by address $\langle c,r\rangle$, where $0 \leq c < w$ and $0 \leq r < h$. A node $\langle c,r\rangle$ is connected by a direct communication channel to its neighboring nodes $\langle c \pm 1, r\rangle$ and $\langle c, r \pm 1\rangle$, where

- $c = -1, c \leftarrow w-1$;
- $r = -1, r \leftarrow h-1$;
- $c = w, c \leftarrow 0$;
- $r = h, r \leftarrow 0$.

Thus each node has four neighboring nodes.

2D subtorus. It is a subgrid $T(p,q)$ in the torus $T(w,h)$ such that $1 \leq p \leq w$ and $1 \leq q \leq h$. A job requesting a subtorus $p \times q$ is denoted by $J(p,q)$. A subtorus S is identified by its *base* (lower left node) and *end* (upper right node), and is denoted as $S[\langle x_b, y_b\rangle\langle x_e, y_e\rangle]$.

Busy subtorus. A busy subtorus β is a subtorus where all of its nodes have been allocated to jobs. Similarly, a subtorus is free when all of its nodes are free.

Busy array. A busy array of a torus $T(w,h)$ is a bit map $B[w,h]$, in which element $B[c,r]$ has a value 1 or 0 if node $\langle c,r\rangle$ is busy or free, respectively.

Busy list. A busy list is a set of all busy subtoruses in the system. Similarly, a free list is a set of all free subtoruses in the system.

Base. It is a node that can be used as a base to allocate an incoming job. A base block is a subtorus whose nodes can be used as base for free subtoruses to allocate a job. A set of disjoint base blocks is called the base set.

Coverage. The coverage of a busy subtorus β with respect to a job J is denoted by $\xi_{\beta,J}$ and it is a set of processors such that the use of any node in $\xi_{\beta,J}$ as the base of a free subtorus for the allocation of J will cause the job J to be overlapped with β. The coverage set with respect to J is denoted by C_J, and it is the set of the coverages of all busy subtoruses.

For the 2D-mesh topology $M(w,h)$, the definitions are equivalent. Additional terms used only in meshes include the following:

Reject area. The reject area with respect to a job J, denoted by R_J, is a set of processors such that the use of any node in R_J as the base of free submesh for the allocation of J will cause the job J cross the boundary of the mesh.

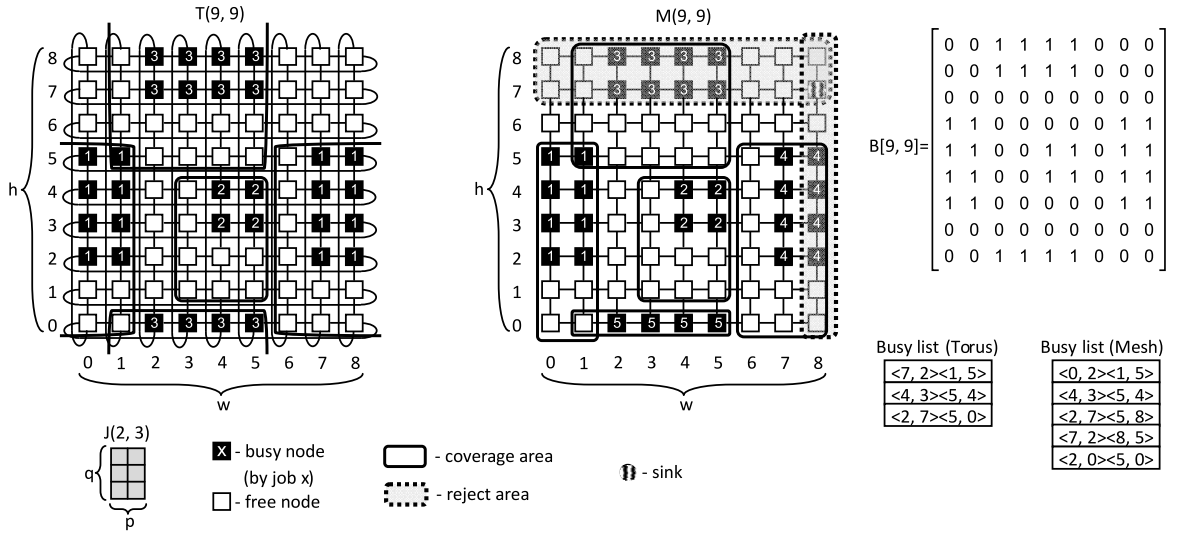


Fig. 5. Torus $T(9,9)$ and mesh $M(9,9)$, busy and free nodes, coverage areas, busy array, busy lists.

Sink. The sink of the reject area is the processor with coordinates $\langle w-p+1, h-q+1\rangle$.

As an illustration of the above definitions, torus $T(9,9)$ and mesh $M(9,9)$ with busy and free nodes, coverage areas, busy array and busy lists with respect to $J(2,3)$ are presented in Fig. 5. The busy nodes are marked black while the free are marked white. The reject area is presented by the shaded region with doted edges. For the torus $T(9,9)$, we get

- the coverages of busy submeshes
  $\beta_1 = [\langle 7,2\rangle\langle 1,5\rangle]$,
  $\beta_2 = [\langle 4,3\rangle\langle 5,4\rangle]$,
  $\beta_3 = [\langle 2,7\rangle\langle 5,0\rangle]$
  are
  $\xi_{\beta_1,J} = [\langle 6,0\rangle\langle 1,5\rangle]$,
  $\xi_{\beta_2,J} = [\langle 3,1\rangle\langle 5,4\rangle]$,
  $\xi_{\beta_3,J} = [\langle 1,5\rangle\langle 5,0\rangle]$;
- the coverage set $C_J = \xi_{\beta_1,J} \cup \xi_{\beta_2,J} \cup \xi_{\beta_3,J}$;
- the base set is $[\langle 6,6\rangle\langle 0,8\rangle] \cup [\langle 2,1\rangle\langle 2,4\rangle]$.

For the mesh $M(9,9)$,

- the coverages of busy submeshes
  $\beta_1 = [\langle 0,2\rangle\langle 1,5\rangle]$,
  $\beta_2 = [\langle 4,3\rangle\langle 5,4\rangle]$,
  $\beta_3 = [\langle 2,7\rangle\langle 5,8\rangle]$,
  $\beta_4 = [\langle 7,2\rangle\langle 8,5\rangle]$,
  $\beta_5 = [\langle 2,0\rangle\langle 5,0\rangle]$
  are
  $\xi_{\beta_1,J} = [\langle 0,0\rangle\langle 1,5\rangle]$,
  $\xi_{\beta_2,J} = [\langle 3,1\rangle\langle 5,4\rangle]$,
  $\xi_{\beta_3,J} = [\langle 1,5\rangle\langle 5,8\rangle]$,
  $\xi_{\beta_4,J} = [\langle 6,0\rangle\langle 8,5\rangle]$,
  $\xi_{\beta_5,J} = [\langle 1,0\rangle\langle 5,0\rangle]$;
- the coverage set $C_J = \xi_{\beta_1,J} \cup \xi_{\beta_2,J} \cup \xi_{\beta_3,J} \cup \xi_{\beta_4,J} \cup \xi_{\beta_5,J}$;
- the base set is $[\langle 0,6\rangle\langle 0,6\rangle] \cup [\langle 6,6\rangle\langle 7,6\rangle] \cup [\langle 2,1\rangle\langle 2,4\rangle]$;
- the reject area $R_J = [\langle 0,7\rangle\langle 8,8\rangle] \cup [\langle 8,0\rangle\langle 8,8\rangle]$;
- the sink has coordinates $\langle 8,7\rangle$.

For a given job J, C_J represents the set of processors which cannot be the base of the free subtoruses (for meshes it is $C_J \cup R_J$). Thus the base set for J is $Z - C_J$ (for a mesh it is $Z - C_J - R_J$), where Z refers to the set of all processors in the system. It is important to note that the base set with respect to $J(p,q)$ is different from that with respect to $J(q,p)$, when $p \neq q$.

It is assumed that jobs are always placed in the torus network from left to right in the horizontal direction and from bottom to top in the vertical direction. This is illustrated in Fig. 5 (torus case). For example, the base of the job J_1 is $\langle 7,2\rangle$ and its end is $\langle 1,5\rangle$. The beginning of J_1 is on the right-hand side of the torus, but its end is on the left-hand side of the network, while the job is placed in the left-right fashion. Similarly to job J_3, the base and the end are $\langle 2,7\rangle$ and $\langle 5,0\rangle$, respectively. The beginning of J_2 is on the top of the system, but its end is on the bottom of the torus, although the job is allocated in the bottom-up way.

The topological characteristic of 2D-torus topology are described in Section 2.1. In the case of processor allocation, 2D-toruses give a better flexibility and a possibility of finding a free processor for the job. This can be noticed in Fig. 5. For the requested allocation of the job $J(2,3)$,

32 nodes are already allocated. It is important to notice that in the torus $T(9,9)$ these 32 nodes are allocated to three jobs J_1 to J_3, but in the mesh $M(9,9)$ it becomes five jobs, J_1 to J_5. Because of wraparound channels in the torus topology, pairs of jobs J_1 with J_4 and J_3 with J_5 in the mesh can be treated in the torus as single jobs J_1 and J_3, respectively. Moreover, the base set in the torus network in Fig. 5 contains 16 nodes, while in the mesh there are only 7.

3.2. Allocation algorithms for 2D-mesh.

3.2.1. Improved First Fit (IFF). The IFF algorithm (Zydek and Selvaraj, 2010) is the approach with a bit map representing the allocation status of processors in the mesh. The bit map idea has also been used by Zhu (1992), who presented the First Fit (FF) and the Best Fit (BF) techniques. In the IFF, with respect to an incoming job $J(p,q)$, the busy array B (without considering the reject area R_J) is examined to create a coverage array C_T, which is a bit map representing the coverage set. In order to form C_T in an efficient way, each coverage $\xi_{\beta,J}$ is divided into three disjoint regions: job coverage, left coverage and bottom coverage. Then, two scans are necessary:

- all rows from right to left (determining a job and left coverages);
- all columns (left to right) from top to down (creating a bottom coverage).

The first available node that does not belong to the coverage set is returned and can serve as a base for the job J. The IFF strategy is recognition complete by considering two job orientations: If allocation of $J(p,q)$ fails and $p \neq q$, then $J(q,p)$ possibility is checked.

With IFF allocation, the achieved time complexity is $O(wh)$. Deallocation of processors in the systems with a busy array reduces to clearing all the elements in bit map B. This is done also in $O(wh)$.

3.2.2. Improved Adaptive Scan (IAS). The IAS algorithm (Zydek and Selvaraj, 2010) replaces a bit map with a list with busy subgrids. A similar approach is also proposed by Chmaj *et al.* (2004), Yoo and Das (2002) as well as Ding and Bhuyan (1993). As in the IFF case, the IAS is based on the fact that for any job J none of the nodes inside $C_J \cup R_J$ can serve as the base. Thus creating C_J and R_J is a first step in the algorithm. Because of the fact that the allocation status of processors is maintained using a busy list, C_J is also in a list form and it is constructed by scanning the busy list, and, for each β in the list, $\xi_{\beta,J}$ is constructed. For a given $\beta = [\langle x_b, y_b\rangle\langle x_e, y_e\rangle]$, its coverage with respect to $J(p,q)$ is $\xi_{\beta,J} = [\langle x_c, y_c\rangle\langle x_e, y_e\rangle]$, where $x_c = \max(0, x_b - p + 1)$ and $y_c = \max(0, y_b - q + 1)$. R_J is found by calculating the sink.

In order to find a node that is not in $C_J \cup R_J$, the IAS for the given job $J(p,q)$ maintains a frame of size $p \times q$, which is identified by the lower left corner. The frame slides through the mesh, starting from the lowest left available node. When nodes in the currently examined frame are not available, the frame slides over the mesh, node by node, taking horizontal and vertical steps, respectively. Each node belonging to $C_J \cup R_J$ is tested by going through the whole coverage list. Membership to $C_J \cup R_J$ causes the algorithm to check another node, otherwise the node can be a base for a given J. In much the same way as for the IFF, if the allocation of $J(p,q)$ fails and $p \neq q$, then the possibility $J(q,p)$ of is checked. The IAS is also recognition complete.

With the IAS solution, the achieved complexity is $O(whB)$, where B is the length of the busy list. Deallocation of a job requires removing an element from the busy list. This can be done in constant time by implementing pointers from allocated jobs to corresponding elements in the busy list. Thus, the time complexity of deallocation is $O(1)$.

3.2.3. Other allocation schemes for 2D-meshes. The other important processor allocation algorithms based on busy lists are ISBA (Chmaj *et al.*, 2004; Zydek and Selvaraj, 2010) and IQA (Zydek and Selvaraj, 2010). However, as reported by Zydek and Selvaraj (2010), the algorithms are hardly synthesizable and they are not considered in this paper.

Another approach to the processor allocation problem is not keeping in memory information about subgrids that are busy, but maintaining a list with free subgrids (keep the list of processors that can be allocated for a requested job). Such an approach is taken, e.g., in the FSL and CFL algorithms (Zydek and Selvaraj, 2010; Ababneh, 2006). However, a synthesis of free list techniques is even more difficult than that of busy list schemes (Zydek and Selvaraj, 2010). Moreover, based on computer simulations (Yoo and Das, 2002; Ababneh, 2006), they do not perform better than other techniques. Thus, free list algorithms are not considered in this paper.

3.3. Allocation algorithms for a 2D-torus.

3.3.1. Bit Map Allocation for Torus (BMAT). The BMAT technique (Zydek and Selvaraj, 2011) is based on the bit map approach used in the IFF algorithm. With respect to an incoming job, the busy array B is scanned to create a coverage array C_T in the form of a bit map. The methodology of creating a coverage array C_T is similar to the IFF scheme. However, due to the lack of the reject area R_J in the torus networks, all nodes have to be considered.

Also similarly to the IFF, each coverage $\xi_{\beta,J}$ is divided into three regions: job coverage, left coverage and bottom coverage. However, instead of two scans required by the IFF, BMAT needs four passes through C_T:

- all rows from right to left, two times each (determining a job and left coverages);
- all columns (left to right) from top to down, two times each (creating a bottom coverage).

These additional two scans are due to wraparound channels in the torus and they are obligatory.

The BMAT technique is recognition complete by manipulating the job orientation. If for a given $J(p,q)$ the allocation fails and $p \neq q$, the scheme will change the orientation of the job and then the $J(q,p)$ possibility is checked. When both attempts fail, the allocation of the job fails.

The time complexity of BMAT for allocation and deallocation is $O(wh)$.

3.3.2. Busy List Allocation for Torus (BLAT). The BLAT algorithm (Zydek and Selvaraj, 2011) is based on the busy list strategy and the approach presented in the IAS scheme. For an incoming job $J(p,q)$, BLAT scans a busy list and creates a coverage set C_J, which is also in a list form. Coverages $\xi_{\beta,J}$ depend on the addresses of base $\langle x_b, y_b\rangle$ and end $\langle x_e, y_e\rangle$ of the busy subgrid β. For the 2D-mesh topology (IAS technique), we had only one possible case of addresses, where $x_b \leq x_e$ and $y_b \leq y_e$. In the 2D-torus topology (BLAT technique), we have four possible cases:

- $x_b \leq x_e$ and $y_b \leq y_e$,
- $x_b > x_e$ and $y_b \leq y_e$,
- $x_b \leq x_e$ and $y_b > y_e$,
- $x_b > x_e$ and $y_b > y_e$.

For each case, coverages need to be determined in a different way. For a given $\beta = [\langle x_b, y_b\rangle\langle x_e, y_e\rangle]$, its coverage with respect to $J(p,q)$ is $\xi_{\beta,J} = [\langle x_1, y_1\rangle\langle x_2, y_2\rangle]$, where $x_1 = x_b - p + 1, y_1 = y_b - q + 1, x_2 = x_e, y_2 = y_e$ and

if $x_b \leq x_e$ and $y_b \leq y_e$ then
  if $x_1 < 0$ then $x_1 = w + x_1$ and
    if $x_1 \leq x_e + 1$ then $x_1 = 0$; $x_2 = w - 1$;
  if $y_1 < 0$ then $y_1 = h + y_1$ and
    if $y_1 \leq y_e + 1$ then $y_1 = 0$; $y_2 = h - 1$;
else if $x_b > x_e$ and $y_b \leq y_e$ then
  if $x_1 \leq x_e + 1$ then $x_1 = 0$; $x_2 = w - 1$;
  if $y_1 < 0$ then $y_1 = h + y_1$ and
    if $y_1 \leq y_e + 1$ then $y_1 = 0$; $y_2 = h - 1$;
else if $x_b \leq x_e$ and $y_b > y_e$ then
  if $x_1 < 0$ then $x_1 = w + x_1$ and
    if $x_1 \leq x_e + 1$ then $x_1 = 0$; $x_2 = w - 1$;
  if $y_1 < y_e + 1$ then $y_1 = 0$; $y_2 = h - 1$;
else if $x_b > x_e$ and $y_b > y_e$ then
  if $x_1 \leq 0$ then $x_1 = 0$; $x_2 = w - 1$;
  if $y_1 \leq 0$ then $y_1 = 0$; $y_2 = h - 1$.

When C_J is created, each node is tested for belonging to C_J, which is done by inspecting the whole C_J for every node in a torus. The node which is not in the C_J can be a base for the given J, otherwise, the algorithm checks another node. The BLAT scheme is recognition complete.

The time complexity for allocation in BLAT is $O(whB)$, where B is the size of the busy list—the number of busy subtoruses. Deallocation is done in $O(1)$.

3.3.3. Other allocation schemes for 2D-toruses. Other processor allocation algorithms for k-ary 2-cubes were proposed by Zydek and Selvaraj (2011). The remaining techniques, namely, SAT and SBAT, use the busy list to keep the status of nodes. However, investigations conducted by Zydek and Selvaraj (2011) showed that the performance of these algorithms is not so bright. Moreover, the results presented by Zydek *et al.* (2010) revealed that the synthesis of the SAT and SBAT techniques not possible. Thus in this paper only the BMAT and BLAT allocation algorithms for a 2D-torus NoC are discussed.

4. Energy model

4.1. Energy of a network-on-chip. A model of power consumption of network routers was proposed by Ye *et al.* (2002). The bit energy in router ($E_{R_{bit}}$) is defined as the dynamic energy consumed while traversing one bit of data through the router:

$$E_{R_{\text{bit}}} = E_{S_{\text{bit}}} + E_{B_{\text{bit}}} + E_{W_{\text{bit}}}, \tag{1}$$

where $E_{S_{\text{bit}}}$ is energy consumed by switch arbitration, $E_{B_{\text{bit}}}$ is buffering energy (buffer write and read energy) and $E_{W_{\text{bit}}}$ is energy consumed by interconnection wires inside the switching fabric (energy required to traverse the crossbar switch). Besides the energy consumed by the router, we also have to consider the energy consumed on the physical channels between tiles ($E_{L_{\text{bit}}}$). Thus, the average energy consumed while sending one bit of data from a tile to its neighboring tile can be calculated as

$$E_{\text{bit}} = E_{R_{\text{bit}}} + E_{L_{\text{bit}}}. \tag{2}$$

Consequently, the average energy consumption while sending one bit of data from tile t_i to tile t_j is

$$E_{\text{bit}}^{t_i,t_j} = E_{R_{\text{bit}}}(N_{\text{hops}} + 1) + E_{L_{\text{bit}}} N_{\text{hops}}, \tag{3}$$

where N_{hops} is the number of channels traversed by a packet between tile t_i and t_j.

In Eqn. (3), $E_{R_{\mathrm{bit}}}$ and $E_{L_{\mathrm{bit}}}$ are constants for a given design. Wolkotte *et al.* (2005a) presented a performance analysis of wires. The estimated $E_{L_{\mathrm{bit}}}$ in [pJ/bit] for the NoC is

$$E_{L_{\mathrm{bit}}} = 0.39 + 0.12 l_{\mathrm{wire}}, \tag{4}$$

where l_{wire} is the length of a physical channel in [mm].

The energy model is proposed for two topologies discussed earlier: a k-ary 2-mesh and a k-ary 2-cube. In the 2D-mesh, the assumed length of the physical channel equals length of the PE edge. That length reported by Vangal *et al.* (2007) is 1.5 mm, so for the 2D-mesh we can assume l_{wire} = 1.5 mm. For the 2D-torus, a folded version is considered that doubles the length of channels in comparison to 2D-mesh, thus for a k-ary 2-cube l_{wire} = 3 mm. Finally, the energy consumed on the physical channels between tiles in [pJ/bit] is $E_{\mathrm{L_{bit}}}^{\mathrm{2D-mesh}} = 0.57$ for k-ary 2-mesh and $E_{\mathrm{L_{bit}}}^{\mathrm{2D-torus}} = 0.75$ for a k-ary 2-cube topology.

The research by Wolkotte *et al.* (2005a) is extended in another work (Wolkotte *et al.*, 2005b), where gate level power simulation of a VC router is performed. The router routes packets according to the DOR scheme. The amount of energy required for a single bit to pass the router is $E_{R_{\mathrm{bit}}} = 0.98$ pJ/bit.

Kim *et al.* (2005) present implementation of a router for routing algorithms based on VCs: DOR and a fully adaptive hybrid algorithm. The presented results show that the average energy per packet for both DOR and adaptive algorithms under uniform and transpose traffic patterns is similar. Thus, for both DOR and adaptive routing techniques router energy $E_{R_{\mathrm{bit}}} = 0.98$ pJ/bit.

For NoCs based on virtual-channel flow control, $E_{R_{\mathrm{bit}}}$ can be expressed as (Kumar *et al.*, 2007)

$$E_{R_{\mathrm{bit}}} = E_{S_{\mathrm{bit}}} + E_{B_{\mathrm{bit}}} + E_{W_{\mathrm{bit}}} + E_{VC_{\mathrm{bit}}}, \tag{5}$$

where $E_{\mathrm{VC_{bit}}}$ is the energy consumed by VC arbitration. In express-virtual-channel flow control, packet traveling EVC is able to bypass the router pipeline of intermediate nodes without buffering, VC and switch arbitration. Thus, it saves $E_{B_{\mathrm{bit}}}$, $E_{\mathrm{VC_{bit}}}$ and $E_{\mathrm{S_{bit}}}$, which reduces $E_{\mathrm{R_{bit}}}$.

The synthesis results presented by Kavaldjiev *et al.* (2004) show that energy required to traverse the crossbar switch $E_{\mathrm{W_{bit}}}$ is 24% of all energy consumed by the router. Thus, the energy of bit traversing through EVC is 24% of the energy of the bit traversing through regular VC:

$$E_{R_{\mathrm{bit}}}^{\mathrm{EVC}} = 0.24 E_{R_{\mathrm{bit}}}^{\mathrm{VC}} = 0.23\text{pJ/bit}. \tag{6}$$

Thus, finally, for all routing algorithms considered (DOR, DOR-LB, Valiant, Valiant-LB and adaptive), the average energy consumption while sending one bit of data from tile t_i to tile t_j can be expressed as follows:

- for a k-ary 2-mesh:

$$E_{\mathrm{bit}}^{t_i,t_j} = 0.98(N_{\mathrm{hops}}^{\mathrm{VC}}+1)+0.23N_{\mathrm{hops}}^{\mathrm{EVC}}+0.57N_{\mathrm{hops}}, \tag{7}$$

- for a k-ary 2-cube:

$$E_{\mathrm{bit}}^{t_i,t_j} = 0.98(N_{\mathrm{hops}}^{\mathrm{VC}}+1)+0.23N_{\mathrm{hops}}^{\mathrm{EVC}}+0.75N_{\mathrm{hops}}, \tag{8}$$

where $N_{\mathrm{hops}}^{\mathrm{VC}}$ is the number of regular VCs traversed by a packet between tiles t_i and t_j, $N_{\mathrm{hops}}^{\mathrm{EVC}}$ is the number of EVCs traversed by a packet between tile t_i and t_j, and N_{hops} is the number of physical channels (number of EVCs + number of VCs − 1) traversed by a packet between tiles t_i and t_j.

4.2. Processor allocator: Energy estimation. A synthesis of PAs based on allocation algorithms for 2D-meshes and 2D-toruses is presented by Zydek and Selvaraj (2010) as well as Zydek *et al.* (2010), respectively. The synthesis of the PA is done for Altera's Stratix III family device EP3SL150F780C2. The presented results contain such parameters as the maximum frequency $f_{\max}$, logic utilization, the number of registers used and the number of combinational ALUTs needed. These parameters allow estimating the total power used by the PA.

Since Stratix III is the target device, we have to deal with Stratix-specific features in power estimation. Altera provides the PowerPlay Early Estimator (Altera Corporation, 2009) based on a spreadsheet. The tool allows users to specify switching activities, $f_{\max}$, usages of various components and other related information to estimate the total power in early design stages. Because, as for the NoC, we would like to get energy estimation for a PA, the average power dissipation P from the spreadsheet is converted into energy consumed in a cycle E_c (Cardarilli *et al.*, 2002), according to the formula

$$E_c = P\frac{1}{F_{\max}}\ [\mu\mathrm{J}], \tag{9}$$

where $F_{\max}$ is the average maximum frequency of $f_{\max}$ at 0 °C and 85 °C in [MHz], obtained as results of synthesis.

The final values are presented in Table 1. Positions marked by the dash "–" mean that the synthesis for these instances was not possible (Zydek and Selvaraj, 2010; Zydek *et al.*, 2010). The results show a significant advantage of busy array techniques, especially the IFF technique. The busy list schemes use between two and five times more energy in comparison with bit map solutions, which makes PAs driven by the IAS and BLAT not efficient. Based on outcomes presented by Zydek and Selvaraj (2010) as well as Zydek *et al.* (2010) and Table 1, the amount of logic and dedicated registers used by these algorithms has huge impact on the maximum frequency and energy consumption.

5. Experimental energy results

5.1. Description of the simulation environment. The proposed energy model for NoCs and PAs has been em-

Table 1. Power P and energy consumed in a cycle E_c for PAs with the discussed allocation algorithms.

Scheme	IFF		IAS		BMAT		BLAT	
Grid Size	P [W]	E_c [J]	P [W]	E_c [J]	P [W]	E_c [J]	P [W]	E_c [J]
2×2	0.6116136	2.7310275	0.6109082	11.299514	0.6116136	9.7351947	0.6114341	18.692574
3×2	0.6114821	2.8569259	0.6113026	17.268436	0.6118108	16.713859	0.6120258	26.890412
4×2	0.6116136	3.2569019	0.6115656	25.766404	0.6119423	21.696234	0.6126175	37.150849
3×3	0.6116136	4.751135	0.6118285	21.949006	0.6120738	30.74975	0.6129462	47.059212
4×3	0.6116136	5.277308	0.6124202	29.837771	0.6123367	37.892125	0.6140641	64.165524
4×4	0.6116793	8.6799964	0.6136038	34.774937	0.6127969	58.809687	0.6155766	86.945855
5×4	0.6117451	9.9228723	0.6148532	46.246951	0.6133229	74.207249	0.6176815	125.80072
5×5	0.6118108	14.274634	0.6168264	46.996295	0.6139804	102.1598	0.6210373	180.53409
6×5	0.6118108	15.829517	0.6186684	62.777108	0.6147695	126.10657	0.6238676	193.14787
6×6	0.6119423	23.113968	0.6213663	69.040705	0.6158875	161.22709	0.6278843	263.26387
7×6	0.612008	25.357698	0.6247235	84.88091	0.6168083	188.62639	0.6328913	349.66373
8×6	0.612008	31.272766	0.6284113	95.286013	0.6175976	219.00625	0.6379012	414.22157
8×7	0.6121395	35.797632	0.633814	100.6054	0.6192423	273.39617	0.6446295	477.50335
8×8	0.612271	42.981467	0.6383628	102.87878	0.6222693	317.48436	–	–
10×8	0.6124025	56.914728	–	–	0.6272728	415.41245	–	–
10×9	0.6125997	65.55374	–	–	0.6289852	435.28386	–	–
10×10	0.6127312	86.482878	–	–	0.6310932	551.17307	–	–
15×10	0.6134544	128.74174	–	–	0.6479754	766.83481	–	–
16×10	0.6135859	136.20109	–	–	0.6517388	835.56256	–	–
20×10	0.6140462	173.21472	–	–	–	–	–	–
20×15	0.6155587	296.65478	–	–	–	–	–	–
20×20	0.6173345	414.31847	–	–	–	–	–	–

ployed in an experimentation system developed by the authors of the present paper. The system allows examining the NoC-based CMP with an integrated PA. The testing environment provides the possibility to test all NoC configurations considered as well as the PA driven by all allocation techniques discussed in this work.

The PA takes jobs from the queue generated by the system, job by job, and tries to find free PEs in order to allocate the job. If such free PEs exist, the PEs are allocated to the job (the job is sent to the PE). If there are no free PEs, the PA waits until another job releases some PEs—jobs are processed in the FCFS fashion.

An allocation process employs the NoC of the CMP. In order to allocate PEs, the allocation message has to be sent from a PA to each PE assigned to a job. It is assumed that this message takes one flit, e.g., if a job requires six PEs, six flits have to be sent from a PA to all PEs assigned to the job. Similarly, if a job is done, a deallocation message (that also takes one flit) has to be sent from each involved PE to the PA. Thus, the PA is updated and just released PEs can accommodate another job. The objective of the experimentation system is to analyze a PA and traffic generated by that PA, so only allocation and deallocation messages are considered. In a real system, a lot of different packets are sent between nodes, e.g., just after allocation, messages with commands and operands have to be sent to the PEs involved. Similarly, while processing, PEs can exchange control messages and data with themselves. When processing is done, messages with results have to be sent as well. However, all messages different from the allocation and deallocation message are not considered.

5.2. Analysis of results. The CMP simulator was run on an Intel Pentium 4 machine (2×3 GHz processor) with 2 GB of RAM. Due to the energy and performance analysis discussed in Section 4.2 and in the work of Zydek and Selvaraj (2011), the final experiments were performed for allocation techniques based on a bit map, i.e., the IFF and BMAT algorithms. The rest of allocation schemes were omitted because of their worse characteristics in comparison with busy array solutions.

The investigations were conducted for a mesh/torus NoC with 100 nodes (10×10). In the experiments, a queue of 1000 jobs for each tested instance was generated. The size of the jobs in the queue was generated using random numbers normally distributed between 1 and 4 for vertical and horizontal size separately. The execution time of the job was generated similarly between 1 and 500 [ms]. The results of the experiments are shown in Table 2.

Jobs from the queue were allocated by the PA configured with the IFF or BMAT algorithms. For the IFF algorithm, the free PEs allocated to a job are always adjacent like in a mesh topology. However, such adjacent PEs can also communicate with each other and a PA using torus topology. Thus, for the PA driven by the IFF algorithm,

Table 2. Energy consumption of the PA, the NoC and the whole CMP with the routing algorithms considered, based on the flow control used.

	PA Energy [μJ]		NoC Energy [μJ]			Total CMP Energy [μJ]		
EVC's Length	BMAT	IFF	BMAT	IFF-Mesh	IFF-Torus	BMAT	IFF-Mesh	IFF-Torus
	DOR							
0	1633.1258	256.8541	4.223335	6.280482	4.25179	1637.349141	263.13463	261.105938
1	1633.1258	256.8541	3.794839	5.059122	3.821422	1636.920645	261.91327	260.67557
2	1633.1258	256.8541	3.730471	4.82637	3.759502	1636.856277	261.680518	260.61365
3	1633.1258	256.8541	3.856711	4.825938	3.88747	1636.982517	261.680086	260.741618
	Valiant							
0	1633.1258	256.8541	7.53419	10.122796	7.584019	1640.660002	266.976944	264.438167
1	1633.1258	256.8541	6.72101	8.301033	6.725983	1639.846818	265.155181	263.580132
2	1633.1258	256.8541	6.748088	8.064732	6.755974	1639.873894	264.91888	263.610122
3	1633.1258	256.8541	7.031965	8.182994	6.400887	1640.157772	265.037142	263.255034
	DOR-LB							
0	1633.1258	256.8541	4.223335	6.280482	4.25179	1637.349141	263.13463	261.105938
1	1633.1258	256.8541	3.794839	5.059122	3.821422	1636.920645	261.91327	260.67557
2	1633.1258	256.8541	3.730471	4.82637	3.759502	1636.856277	261.680518	260.61365
3	1633.1258	256.8541	3.856711	4.825938	3.88747	1636.982517	261.680086	260.741618
	Valiant-LB							
0	1633.1258	256.8541	7.453812	9.961001	7.677578	1640.579619	266.815149	264.531725
1	1633.1258	256.8541	6.699823	8.299948	6.704807	1639.825629	265.154096	263.558954
2	1633.1258	256.8541	6.764379	8.072758	6.667442	1639.890185	264.926905	263.52159
3	1633.1258	256.8541	7.083686	8.269919	7.025202	1640.209492	265.124067	263.87935
	Adaptive							
0	1633.1258	256.8541	6.530851	8.600572	6.567831	1639.656657	265.4547198	263.421979
1	1633.1258	256.8541	6.028247	7.147596	6.060985	1639.154054	264.0017438	262.915133
2	1633.1258	256.8541	5.910089	6.852687	5.914527	1639.035895	263.706835	262.768674
3	1633.1258	256.8541	6.111203	6.765978	6.126948	1639.237009	263.6201262	262.981096

we consider two cases:

- IFF-Mesh, where the PEs are adjacent like in the mesh communicate with each other using a mesh-based NoC,
- IFF-Torus, where the PEs are adjacent like in the mesh communicate with each other using a torus-based NoC.

The BMAT technique finds free PEs for the requested job based on a torus topology, and thus the neighboring PEs allocated to a job can be adjacent using wraparound channels. In order to exchange messages, the PEs could use a mesh-based NoC. However, in such a case we could destroy the locality of the PEs allocated to one job. Additionally, jobs allocated to PEs using wraparound channels would not be contiguous if a mesh-based NoC were used. Thus, we consider only torus-based NoCs as the communication medium for a PA with BMAT.

In the experiments, the jobs were allocated to PEs using an NoC with virtual-channel or express-virtual-channel flow control. In Table 2, the flow control used is marked in the column "EVC's length". For virtual-channel flow control, the length of EVCs is 0. In the express-virtual-channel approach, the EVC length is a parameter (in the experiments it varies from 1 to 3).

The routing techniques considered are DOR, Valiant, DOR-LB, Valiant-LB and adaptive. The results presented in Table 2 are for the adaptive algorithm, where two misroutes are allowed.

NoCs with each described flow control and routing scheme were tested for the same queue with jobs. The presented energy results are calculated according to the formulas (7) and (8), and based on the results of the synthesis presented in Table 1. During calculations it was assumed that the width of one flit is 32 bits, which is the most popular width used in research.

As can be noticed, the PA with the BMAT allocation strategy consumes significantly (six times) more energy than with the IFF scheme. It is the price for wraparound channel recognition offered by BMAT. Among the NoCs considered in the experiment, the NoC with the DOR routing technique achieves the lowest energy usage. It is not surprising because the DOR is the easiest possible algorithm and it routes packets through minimal paths. The Valiant algorithm requires a high amount of energy to route the traffic. Even the most advanced adaptive routing algorithm achieves better energy performance. The largest amount of energy is consumed in a mesh-based NoC (IFF-Mesh case in Table 2). The lack of wraparound channels in a mesh topology makes longer routes, which increases

the number of routers involved in transmission together with energy needed to send a message.

The experiments reveal enormous advantages of the IFF-based PA with a torus-based NoC. The IFF-Torus approach was characterized by a very good energy performance. Moreover, the torus topology of the NoC ensures better traffic balance than the mesh. This solution, in connection with the DOR-LB routing technique, yields very good energy and load balance characteristics. The BMAT-based PA demonstrates very high energy consumption, which makes this solution less attractive.

Implementation of EVCs for all routing techniques considered decreases the amount of energy used. Especially for a mesh-based NoC, savings offered by express-virtual-channel flow control are significant. The best results are achieved in the case where the number of VC buffers used is reduced by employing more express buffers. Thus, if more express buffers are used, the energy saving gained is higher. The length of EVCs also has impact on the amount of energy saved and its choice has to be made individually for each system. In the described experiments, the optimal length of EVCs is 2—one express channel bypasses virtually two nodes.

6. Conclusions

In this paper, we investigated the energy characteristic of two components of novel CMPs: the processor allocator and the network-on-chip. We explored the most important NoC architectures and processor allocation algorithms for k-ary 2-meshes and k-ary 2-cubes topologies. To perform the energy analysis, we proposed an energy model for NoCs based on virtual-channel and express-virtual-channel flow control. Energy estimation for a PA was done based on results of hardware synthesis of the PA. The proposed energy model of the NoC-based CMP was implemented in an experimentation environment, where the intensive simulations of the CMP were conducted.

The presented energy results reveal a huge advantage of the PA driven by bit map algorithms, especially by the IFF algorithm for mesh systems. However, as an efficient NoC solution, the torus network proved to be very energy efficient. This led to the idea of implementing the PA driven by the IFF with the torus topology of the NoC (IFF-Torus). This solution turned to be the best one. The IFF-based PA and the torus-based NoC driven by DOR-LB routing with express-virtual-channel flow control deliver very good energy characteristics and ensure good load balance. If higher reliability and load balance are needed, the adaptive routing technique with carefully chosen parameters can also be a very good solution.

Acknowledgment

This work has been supported by the European Union in the framework of the European Social Fund through the Warsaw University of Technology Development Programme, and by the Ministry of Science and Higher Education under grant no. N517 003 32/0583.

References

Ababneh, I. (2006). An efficient free-list submesh allocation scheme for two-dimensional mesh-connected multicomputers, *Journal of Systems and Software* **79**(8): 1168–1179, DOI: 10.1016/j.jss.2006.01.019.

Altera Corporation (2009). *Quatrus II 9.1 Handbook*, Vol. 3, Altera, San Jose, CA.

Boura, Y. and Das, C. R. (1994). Efficient fully adaptive wormhole routing in n-dimensional meshes, *14th International Conference on Distributed Computing Systems, Poznań, Poland*, pp. 589–596, DOI: 10.1109/ICDCS.1994.302473.

Cardarilli, G., Re, A. D., Nannarelli, A. and Re, M. (2002). Power characterization of digital filters implemented on FPGA, *IEEE International Symposium on Circuits and Systems (ISCAS 2002)*, Vol. 5, pp. 801–804, DOI: 10.1109/ISCAS.2002.1010825.

Chmaj, G., Zydek, D. and Koszalka, L. (2004). Comparison of task allocation algorithms for mesh-structured systems, *Computer Systems Engineering, Theory & Applications, 4th Polish-British Workshop, Szklarska Poręba, Poland*, pp. 39–50.

Dally, W. (1990). Performance analysis of k-ary n-cube interconnection networks, *IEEE Transactions on Computers* **39**(6): 775–785, DOI: 10.1109/12.53599.

Dally, W. (1992). Virtual-channel flow control, *IEEE Transactions on Parallel and Distributed Systems* **3**(2): 194–205.

Dally, W. and Seitz, C.L. (1987). Deadlock-free message routing in multiprocessor interconnection networks, *IEEE Transactions on Computers* **36**(5): 547–553, DOI: 10.1109/TC.1987.1676939.

Dally, W. and Towles, B. (2001). Route packets, not wires: On-chip interconnection networks, *38th Annual Design Automation Conference*, pp. 684–689, DOI: 10.1109/DAC.2001.156225.

Dally, W. and Towles, B. (2004). *Principles and Practices of Interconnection Networks*, Morgan Kaufmann, San Francisco, CA.

Ding, J. and Bhuyan, L. N. (1993). An adaptive submesh allocation strategy for two-dimensional mesh connected systems, *International Conference on Parallel Processing, Syracuse, NY, USA,* Vol. 2, pp. 193–200, DOI: 10.1109/ICPP.1993.39.

Duato, J., Yalamanchili, S. and Ni, L. (2003). *Interconnection Networks*, Morgan Kaufmann, San Francisco, CA.

Jayasimha, D., Zafar, B. and Hoskote, Y. (2006). On-chip interconnection networks: Why they are different and how to compare them, *Technical report*, Intel Corp, Oration ,Santa Clara, CA.

Kavaldjiev, N., Smit, G.J.M. and Jansen, P.G. (2004). A virtual channel router for on-chip networks, *IEEE International System-on-Chip Conference*, pp. 289–293, DOI: 10.1109/SOCC.2004.1362438..

Kim, J., Park, D., Theocharides, T., Vijaykrishnan, N. and Das, C.R. (2005). A low latency router supporting adaptivity for on-chip interconnects, *42nd Annual Design Automation Conference*, pp. 559–564, DOI: 10.1145/1065579.1065726..

Krishna, T., Kumarand, A., Chiang, P., Erez, M. and Peh, L.S. (2008). NoC with near-ideal express virtual channels using global-line communication, *16th IEEE Symposium on High Performance Interconnects*, pp. 11–20, DOI: 10.1109/HOTI.2008.22.

Krueger, P., Lai, T. H. and Dixit-Radiya, V. A. (1994). Job scheduling is more important than processor allocation for hypercube computers, *IEEE Transactions on Parallel and Distributed Systems* **5**(5): 488–497, DOI: 10.1109/71.282559.

Kumar, A., Peh, L.S., Kundu, P. and Jha, N.K. (2007). Express virtual channels: Towards the ideal interconnection fabric, *ACM SIGARCH Computer Architecture News* **35**(2): 150–161, DOI: 10.1145/1273440.1250681.

Mohapatra, P., Yu, C., Das, C.R. and Kim, J. (1993). A lazy scheduling for improving hypercube performance, *The 1993 International Conference on Parallel Processing (ICPP '93)*, Vol. 1, pp. 110–117, DOI: 10.1109/ICPP.1993.26.

Rezazad, M. and Sarbazi-Azad, H. (2005). The effect of virtual channel organization on the performance of interconnection networks, *19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)*, Vol. 15, DOI: 10.1109/IPDPS.2005.427.

Rose, C., Heiss, H.U. and Linnert, B. (2007). Distributed dynamic processor allocation for multicomputers, *Parallel Computing* **33**(3): 145–158, DOI: 10.1016/j.parco.2006.11.010.

Su, C. and Shin, K.G. (1993). Adaptive deadlock-free routing in multicomputers using only one extra virtual channel, *1993 International Conference on Parallel Processing*, Vol. 1, pp. 227–231, DOI: 10.1109/ICPP.1993.37.

Taylor, M., Kim, J., Miller, J., Wentzlaff, D., Ghodrat, F., Greenwald, B., Hoffman, H., Johnson, P., Lee, J.W., Lee, W., Ma, A., Saraf, A., Seneski, M., Shnidman, N., Strumpen, V., Frank, M., Amarasinghe, S. and Agarwal, A. (2002). The raw microprocessor: A computational fabric for software circuits and general-purpose programs, *IEEE Micro* **22**(2): 25–35, DOI: 10.1109/MM.2002.997877.

Upadhyay, J., Varavithya, V. and Mohapatra, P. (1997). A traffic-balanced adaptive wormhole routing scheme for two-dimensional meshes, *IEEE Transactions on Computers* **46**(2): 190–197, DOI: 10.1109/12.565594.

Valiant, L. and Brebner, G.J. (1981). Universal schemes for parallel communication, *13th Annual ACM Symposium on Theory of Computing*, pp. 263–277, DOI: 10.1145/800076.802479.

Vangal, S., Howard, J., Ruhl, G., Dighe, S., Wilson, H., Tschanz, J., Finan, D., Iyer, P., Singh, A., Jacob, T., Jain, S., Venkataraman, S., Hoskote, Y. and Borkar, N. (2007). An 80-tile 1.28TFLOPS network-on-chip in 65nm CMOS, *IEEE International Solid-State Circuits Conference (ISSCC 2007), San Francisco, CA, USA,* pp. 98–589, DOI: 10.1109/ISSCC.2007.373606.

Wolkotte, P., Smit, G. and Becker, J. (2005a). Energy efficient NoC for best effort communication, *15th International Conference on Field Programmable Logic and Applications*, pp. 197–202, DOI: 10.1109/FPL.2005.1515722.

Wolkotte, P., Smit, G.J.M., Kavaldjiev, N., Becker, J.E. and Becker, J. (2005b). Energy model of networks-on-chip and a bus, *2005 International Symposium on System-on-Chip, Tampere, Finland,* pp. 82–85, DOI: 10.1109/ISSOC.2005.1595650.

Ye, T., Benini, L. and Micheli, G.D. (2002). Analysis of power consumption on switch fabrics in network routers, *39th Annual Design Automation Conference*, pp. 524–529, DOI: 10.1145/513918.514051.

Yoo, B. and Das, C.R. (2002). A fast and efficient processor allocation scheme for mesh-connected multicomputers, *IEEE Transactions on Computers* **51**(1): 46–60, DOI: 10.1109/12.980016.

Zhu, Y. (1992). Efficient processor allocation strategies for mesh-connected parallel computers, *Journal of Parallel and Distributed Computing* **16**(4): 328–337, DOI: 10.1016/0743-7315(92)90016-G.

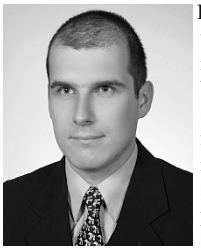
Zydek, D. and Selvaraj, H. (2009). Processor allocation problem for NoC-based chip multiprocessors, *6th International Conference on Information Technology: New Generations (ITNG 2009), Las Vegas, NV, USA,* pp. 96–101, DOI: 10.1109/ITNG.2009.182.

Zydek, D. and Selvaraj, H. (2011). Fast and efficient processor allocation algorithm for torus-based chip multiprocessors. *Journal of Computers & Electrical Engineering* **37**(1): 91–105, DOI: 10.1016/j.compeleceng.2010.10.001..

Zydek, D. and Selvaraj, H. (2010). Hardware implementation of processor allocation schemes for mesh-based chip multiprocessors, *Microprocessors and Microsystems* **34**(1): 39–48, DOI: 10.1016/j.micpro.2009.11.003.

Zydek, D., Selvaraj, H. and Gewali, L. (2010). Synthesis of processor allocator for torus-based chip multiprocessors, *7th International Conference on Information Technology: New Generations (ITNG 2010), Las Vegas, NV, USA,* pp. 13–18, DOI: 10.1109/ITNG.2010.145.

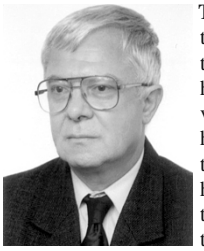
Zydek, D., Shlayan, N., Regentova, E. and Selvaraj, H. (2008). Review of packet switching technologies for future NoC, *19th International Conference on Systems Engineering (ICSEng 2008), Las Vegas, NV, USA,* pp. 306–311, DOI: 10.1109/ICSEng.2008.47.



Dawid Zydek earned his Pg.Cert. degree in systems and control from Coventry University, UK, in 2005, an M.Sc. in computer science from the Wrocław University of Technology, Poland, also in 2005, and a Ph.D. in computer engineering from the University of Nevada, Las Vegas, USA, in 2010. Since 2007 he has been a part-time instructor in the Department of Electrical and Computer Engineering, University of Nevada. His research interests include design, optimization and job scheduling for computer systems, parallel processors architectures, NoC structures, computer networks and computer experimentation systems. He has published more than ten scientific papers in many international publications.

Henry Selvaraj earned his Master's and Ph.D. from the Warsaw University of Technology in 1986 and 1994, respectively. In the years 1994–1998, he was a member of the Faculty of Computing and Information Technology, Monash University, Melbourne, Australia. Since 1998 he has been a faculty member of the Department of Electrical and Computer Engineering, University of Nevada, Las Vegas, USA. Currently, he is a professor and the chair of the Department. His research interests include logic synthesis, programmable logic devices, digital signal processing and systems engineering. He has published more than 110 scientific papers in refereed journals and proceedings.

Grzegorz Borowik earned his M.Sc. degree in mathematics from the Warsaw University of Technology (WUT), Poland, in 2002. He earned his Ph.D. degree in telecommunications from the same university in 2007. He joined the Institute of Telecommunications of the WUT in 2003, where he has been an assistant professor since 2007. Since 2008 he has been a technical editor of the *Electronics and Telecommunications Quarterly* of the Polish Academy of Sciences, and, since 2010, of the *International Journal of Electronics and Telecommunications*. His research interests include logic synthesis, optimization and testability for programmable logic devices, digital signal processing, cryptography, and numerical algorithms. Dr. Borowik has authored or co-authored about 30 scientific papers published in Polish and international scientific journals as well as conference proceedings.



Tadeusz Łuba earned the M.Sc. degree in electronics engineering from the Warsaw University of Technology, Poland, in 1971. He earned his Ph.D. and D.Sc. degrees from the same university in 1980 and 1988, respectively. In 1972 he joined the Institute of Telecommunications of the Warsaw University of Technology, where he has been a professor since 1993. Currently he is the head of the Telecommunications Fundamentals Department with the Institute of Telecommunications. His research interests include switching theory, CAD tools for logic synthesis and optimization, compilers for programmable logic devices, and digital signal processing. Prof. Łuba has authored or co-authored almost 200 scientific papers published in prestigious Polish and foreign scientific journals as well as numerous conference contributions. He is a member of the programme committees of several international and national conferences and a board of directors member of EUROMICRO. Since 2007 he has been a member of the Committee on Electronics and Telecommunications of the Polish Academy of Sciences and the editor-in-chief of the *International Journal of Electronics and Telecommunications*.

Received: 12 May 2010
Revised: 10 September 2010