PC-Mesh: A Dynamic Parallel Concentrated Mesh

J. Camacho, J. Flich, A. Roca, J. Duato  
Parallel Architectures Group  
Universitat Politècnica de València  
Spain

Abstract—We present a novel network on-chip topology, PC-Mesh (Parallel Concentrated Mesh), suitable for tiled CMP systems. The topology is built using four concentrated mesh (C-Mesh) networks and a new network interface able to inject packets through different networks. The goal of the new combined topology is to minimize the power consumption of the network when running applications exhibiting low traffic rates and maximize throughput when applications require high traffic rates. Thus, the topology is dynamically adjusted (switching on and off network components) with a proper injection algorithm, adapting itself to the network on-chip traffic requirements.

The PC-Mesh network performs as a C-Mesh network (using one subnetwork) when the traffic is low obtaining large savings in power consumption. When the load network increases, new subnetworks are opened and thus higher traffic rates are supported, thus providing comparable results as the mesh network. Additional benefits of the PC-Mesh network is its fault tolerance degree and the lower latency in terms of hops. An alternative PC-Mesh version is provided to optimize the fault-tolerance degree.

Comparative results with detailed evaluations (in area, power, and delay) are provided both for the network interface and switches. Results demonstrate PC-Mesh is able to dynamically adapt to the current traffic situations. Experimental results with a system-level simulation platform (including the application being run and the operating system) are provided. Results show how the PC-Mesh network achieves the same results as the C-Mesh topology reducing execution time of applications by 20% as well as energy consumption by also 20%, when compared with the 2D-Mesh network topology. However, when challenged with higher traffic demands, PC-Mesh outperforms the C-Mesh network by achieving much lower execution time of applications and lower energy consumption. In some scenarios, execution time is reduced by a factor of 2 and power consumption by 50%.

I. INTRODUCTION

As technology scales, the number of transistors that can be integrated on a chip increases. This allows designers to add more functionality on current microprocessors. The current trend, however, is to replicate basic simple processor components (together with cache memories) and thus, increase the number of processing elements on the same chip. These chips are known as Chip MultiProcessors (CMPs). This design style is preferred over a design with complex processor cores. Power consumption is becoming the limiting factor and smaller processor cores have a better performance-power consumption ratio. Currently, there are chip prototypes and real products with tens of processors. Examples include the Intel Polaris chip [20] with 80 simple cores, and the Single-chip Cloud Computer [18] with 48 x86 compatible processors, each able to run an operating system. Also, Tilera has shipped its new 100-core chip [21].

With advances in technology, we can expect chips with hundreds of cores in the near future so the way these cores are connected becomes an important and challenging issue. Although buses, rings, and crossbar topologies were used in initial systems (e.g. Cell Broadband Engine processor [3]), these structures do not scale well and therefore achieve low performance when the number of cores is high. Indeed, current CMP systems mostly rely on a 2D-Mesh topology. In such a topology, the switch is connected to its neighbors in the north, east, west and south directions. The mesh network is appealing since it matches the planar surface of the chip. Indeed, the tile-based design (a tile is designed and the chip is built by replicating the same tile design) promotes the use of a 2D structure. One negative aspect of the 2D-Mesh topology, however, is the high latency of messages due to the number of hops needed when communicating with distant nodes. This becomes a problem as system size increases.

An alternative network on-chip topology is the concentrated mesh (C-Mesh) [1]. In this topology, the hop count decreases by reducing the number of switches (75% reduction when compared with the 2D-Mesh with the same number of end nodes). In the C-Mesh topology sets of four neighboring nodes are connected to the same switch. It scales better than the 2D-Mesh topology because of its lower hop count. In addition, despite the larger number of switch ports, having a quarter of switches than in 2D-Mesh leads to large savings in power consumption. Unfortunately, the reduced bisection bandwidth of the C-Mesh topology leads latency to exponentially increase and, potentially, high energy consumption due to congestion.

Therefore, the C-Mesh topology is a good candidate topology when traffic requirements are low. With low injection loads, packets will experience low latency values and the network power consumption will be low. However, with high traffic requirements (bursty traffic, barrier synchronization, hot-spots) the low bisection bandwidth of the network leads to congestion, thus high packet latencies and large power consumption values. Figure 1 shows both the 2D-Mesh and C-Mesh topologies using 16 nodes. The main difference is the number of end nodes connected to each switch.

There are different solutions that address the capacity problem of the C-Mesh network for high traffic requirements. One possible solution is the use of express links between non-neighboring switches. Other solutions rely on higher dimensional topologies. However, most of these solutions require higher radix switches, which has proven to severely impact the operating frequency of switches [27].

In this paper we take as a reference design the C-Mesh network due to its good performance with low traffic loads (low latency). Moreover, we address the C-Mesh topology capacity limitations in high traffic loads with an alternative
approach. We extend the C-Mesh topology in order to obtain savings in power with low traffic loads, as in the C-Mesh topology, and to avoid congestion with high traffic loads, achieving similar bisection bandwidth as of the 2D-Mesh, but still exhibiting low end-to-end latency. The proposed network-on-chip topology is the parallel concentrated mesh (PC-Mesh). The new topology, as its own name indicates, is composed by different concentrated meshes. Specifically, we have four different concentrated meshes (or subnetworks). Each subnetwork is used only when necessary, otherwise the subnetwork is in off state and every switch is powered off. The key differentiating point is the fact the end node is connected to different subnetworks, thus, having more opportunities for power savings and performance optimization.

In order to achieve large power saving values, the PC-Mesh topology is enriched by a injection (selection) algorithm at the network interface of each node. The algorithm manages the parallel networks in order to maximize power savings without compromising performance. This algorithm works in parallel and closely with a distributed on/off power saving mechanism at every switch. The complementarity of both mechanisms is the key to achieve large power saving values. In this paper we provide a detailed implementation and evaluation (in terms of area, power, and delay) of the injection algorithm.

One important design point in a NoC system is also the routing algorithm. The routing algorithm, highly coupled with the topology, determines on every switch which output port every message must take in order to reach its destination. In both topologies, the most efficient routing algorithm (in terms of implementation complexity and power consumption) is dimension-order routing (DOR). DOR is implemented on every switch and requires small logic blocks. The message first moves in the X dimension and once it reaches the same column of the destination, then it moves through the Y dimension, always following minimal paths. The low complexity of DOR makes it very appealing for network-on-chip designers. Indeed, it prevents the need of using routing tables which are expensive in terms of area, delay, and power. However, DOR only provides a single path for every source-destination pair. This occurs in both 2D-Mesh and C-Mesh topologies. This fact leads to not exhibiting any alternative path, thus not being, for instance, able to tolerate a single link or switch failure. In this paper we assume the use of DOR algorithm by default, but with the new topology, every source-destination pair will have more than one alternative X-Y path, thus providing higher fault-tolerance rates than Mesh and C-Mesh topologies. Indeed, in order to maximize fault-tolerance, and still using the DOR algorithm, two different configuration layouts will be provided for the PC-Mesh topology, each one achieving similar characteristics. However, because of the different connectivity pattern, they will show different fault tolerance properties.

As a summary of contributions, we propose (1) a new topology for NoCs that offers reduced hop latencies as the C-Mesh network and enables the use of alternate paths; (2) an alternative design able to increase the fault-tolerance properties of the PC-Mesh topology; (3) an injection algorithm that allows every end node to decide the best subnetwork to use in order to achieve low end-to-end latencies and minimized power consumption values; and (4) a detailed implementation analysis in terms of power, area, and delay of the network interface with the implementation of the injection algorithm.

Experimental results with an execution-driven simulation model demonstrate significant power consumption reduction for the new topologies. 20% power is saved, on average, when compared to the 2D-Mesh topology. Also, the execution time of real applications is decreased, on average, by 20%. These results match C-Mesh behavior. However, when challenged with higher traffic demands the PC-Mesh topology enables much more intrinsic bandwidth, reducing execution time by a factor of 2 and power consumption by 50%. It is worth mentioning these results are achieved with the PC-Mesh network having the same switch radix than the C-Mesh topology. Thus, benefits when using higher connectivity, e.g. express links are applicable to both topologies.

The rest of the paper is organized as follows. In Section II, the new PC-Mesh topology is described. Section III describes the proposed power-aware routing algorithm used at each end node interface. Next, Section IV describes and evaluates the fault tolerance degree of the PC-Mesh topology. In Section V, every topology is evaluated, taking into account both synthetic and real applications traffic. Then, in Section VI related work is discussed. Finally, in Section VII, the paper is concluded and future research directions are provided.

II. PC-MESH

Figures 2(a) shows the PC-Mesh topology. End nodes are represented as circles and switches as squares. When focusing only on switches, they build four disjoint subnetworks, thus, every switch belongs only to a single subnetwork. In our example, four $2 \times 2$ networks are build (each with a different link color and pattern). Notice that switches of different subnetworks are not connected between them. Although it is possible, short links would be used, it would increase the radix of the switch quite significantly, with the known problems this carries on [27]. Also, as we plan to switch off entire subnetworks, they should be isolated from the others in order to get an efficient power management strategy.

When focusing on end nodes, we can see that they have different connectivity patterns. Nine nodes are connected to four switches and the remaining ones to fewer switches. In particular, the top left-most node is connected to a single switch and the remaining boundary nodes are connected to
two switches. As end nodes are connected to some switches, additional logic is required at the network interface. The logic will implement an algorithm that selects the switch (network) to use on a per packet basis (described in the next section). As we will see in the evaluation, the algorithm and associated logic has negligible impact.

One downside of the PC-Mesh topology shown in Figure 2(a) is that some end nodes have lower connectivity than others. To homogenize the network, we add a row and a column of extra switches, leading to the topology shown in Figure 2(b). Now, the four C-Mesh topologies are different in size and number of switches, however, every end node is connected to the four parallel subnetworks. This configuration will be referred to as Fault-Tolerant PC-Mesh (FTPC-Mesh).

It is important to note that the FTPC-Mesh configuration uses more switches than end nodes, and at first sight this will render more power consumption. However, it is necessary to note that these extra switches will be also dynamically powered on and off, thus, used only when really needed.

A. Tile-Based Design

One important aspect in CMP systems is to use a tile-based design as it significantly reduces the design effort. Thus, the proposed PC-Mesh topology must be adapted to a tile-based structure. Indeed, the PC-Mesh topology is similar to the C-Mesh topology. The key differentiating points are the connection of each end node to more than one subnetwork and the existence of a switch on every tile (as the 2D-Mesh). Figure 2(c) shows a possible tile design for the PC-Mesh topology. The differences with the C-Mesh topology lie on the extra links connecting end nodes to switches and the higher number of switches. Overlapped links can effectively be routed over the logic through higher metal layers.

B. Injection Algorithm

In this section we describe the injection algorithm assuming a $4 \times 4$ PC-Mesh topology. The goal of the algorithm is to adapt the use of subnetworks to the current injection load of the end node. The four subnetworks are used in an ordered way and the algorithm is used on a per packet basis. The logic have been modeled in Verilog and results in terms of power, area and delay are provided in Section V-A.

As XY routing is assumed and an end node now can be reached through four subnetworks, now we need to compute the final switch destination through each possible subnetwork. The labels of end nodes and switches is depicted in Figure 2(a). End nodes are labeled from 0 to 15 and switches belonging to the same subnetwork from 0 to 3. As can be seen, end node 10 can be reached through switch 3 in subnetwork 0 (green), 2 in subnetwork 1 (blue), 1 in subnetwork 2 (red), and 0 in subnetwork 3 (black).

Upon reception of a packet to be injected into the network, the algorithm proceeds with three stages computed in parallel. In the first stage, to compute the switch destination reaching the end node, we derive the following equations (assuming destination end node is coded in four bits $b_3 b_2 b_1 b_0$, the network to use is represented by four signals $N_x$, and $x$ is the subnetwork id):

$$\begin{align*}
X_x &= b_1 + (N_1 \lor N_3) \times b_0 \\
Y_x &= b_3 + (N_2 \lor N_3) \times b_2
\end{align*}$$

This equation is compatible with the lower connectivity of the end nodes along the first row and column of switches. A similar equation can be deduced for the FTPC-Mesh topology.

In parallel, at the second stage, the number of enqueued flits $F_x$ for each subnetwork are checked. A threshold register is used ($T_h$) to compare with. Several thresholds have been tested, and the chosen values appear in the evaluation section. For each subnetwork $x$, a $T_x$ signal is computed representing whether the threshold has been exceeded or not ($T_x = F_x > T_h$). This threshold is used for low injection rates of the node (when injection is below 20%). For high injection rates (superior to 20%), the $T_h$ threshold is considered to be zero (enabling all the networks to be used).

As different path lengths are achieved from the same source-destination pair, depending on the subnetwork used, in the third stage the end-to-end node distances are computed. The Manhattan distance ($MD_x$) for each subnetwork is obtained by

1As an example, end nodes 5 and 10 are one hop away through subnetwork 3, two hops away through subnetworks 1 and 2, and three hops away through subnetwork 0.
computing the distances along X and Y dimensions: $MD_x = abs(X_x - XC_x) + abs(Y_x - YC_x)$ where $XC_x$ and $YC_x$ are the coordinates of the switch attached to the injection node for the particular $x$ subnetwork (these values are constant and, thus, there is no need to compute at every packet injection).

After the three parallel stages, at the fourth stage, the different manhattan distances are compared in order to prepare the selection of the final subnetwork. To do this, three $CMP_i$ signals are computed, where manhattan distances are compared (the $CMP_i$ signal just checks if the subnetwork $i$ provides equal or shorter path for the destination end node):

- $CMP_1 = MD_1 <= MD_0$
- $CMP_2 = MD_2 <= min(MD_1, MD_0)$
- $CMP_3 = MD_3 <= min(MD_2, MD_1, MD_0)$

In the final stage, the subnetwork to use is computed, based on the $CMP_x$, $T_x$ and $S_x$ signals. The $S_x$ signals indicate whether the attached switch in the $x$ subnetwork is powered on or not. $SEL_x$ signals are computed appropriately as follows:

- $SEL_0 = (T_0 \times S_0) + (SEL_1 \times SEL_2 \times SEL_3)$
- $SEL_1 = SEL_0 \times T_1 \times CMP_1 \times S_1$
- $SEL_2 = SEL_1 \times SEL_0 \times T_2 \times CMP_2 \times S_2$
- $SEL_3 = SEL_2 \times SEL_0 \times SEL_1 \times T_3 \times CMP_3 \times S_3$

Networks are prioritized based on the previous equations. Notice that, in order to optimize the most frequent case (low injection rates), if $T_0$ is set (meaning enqueued flits for the first subnetwork is below the threshold), then the remaining logic can be switched off.

With the designed algorithm, the PC-Mesh network will allow most of the subnetworks to be switched off as traffic will be directed to only a subnetwork, thus resembling a C-Mesh network for low traffic. For high traffic rates, the PC-Mesh network will use an increasing number of subnetworks, thus, maximizing performance.

III. POWER MANAGEMENT ALGORITHM

In this section we combine the injection algorithm with a power management algorithm. The goal is to switch off complete switches (not only the ports of the switch) while they detect no traffic activity. Switches, however, will cooperate with neighboring switches and attached end nodes. The basic goal of the algorithm is to maximize the time switches and links are powered down, thus enabling large savings in static power. The designed injection algorithm will avoid, whenever possible, the use of powered down switches. In this paper we focus exclusively on the switches, considering both, power and clock gating.

We elaborate the power management mechanism assuming the new logic. For switching off, $PML$ measures the number of cycles the switch is empty of flits (buffer use of every input port). Upon reaching a threshold ($T_{swoff}$) the switch is powered down as well as the clock feeding the switch.

The process to power down a block may consume some power, however this power consumption is compensated with the number of cycles the switch is powered off. Then to save power, we need the logic is not powered on again until a certain number of cycles, more than 9 cycles specifically. However, there is no power penalty waking up the switch earlier. See [8] for more details.

The $PML$ logic is waken upon arrival of a control signal from the connected devices (neighbor switches or attached end nodes). In order to overlap the 3-cycles wake up time required by the switch, we have augmented the VA_SA stage of the switch to trigger the $switch_{on}$ signal, reaching the next switch in one cycle.

IV. FAULT TOLERANCE

One of the key properties of the PC-Mesh network is its fault-tolerance degree. Providing parallel networks increases significantly the reliability of the entire network. In this section we analyze this characteristic by computing the probability that one or more failed component (switches and the attached links) leave the end nodes unconnected. We provide the analysis for both PC-Mesh and FTPC-Mesh network topologies and assume the use of the DOR routing algorithm.

The hypergeometric distribution has been used to obtain the fault tolerance for the PC-Mesh and FTPC-Mesh topologies. Figure 4 shows the obtained results for both versions of the topology: PC-Mesh and FTPC-Mesh, when using 16 and 32 cores. As can be seen, FTPC-Mesh is able to tolerate 3 failures as has four complete parallel networks. However, the PC-Mesh version obtains an 75% of fault-tolerance for a single switch failure. Indeed, having one end node connected to a single switch prevents achieving 100% fault tolerance coverage for the one-switch failure case. Even though, when compared to the C-Mesh and 2D-Mesh topologies results are good, since these topologies are not able to tolerate, assuming the DOR routing algorithm, a single switch failure in any case.

The figure also shows the graceful degradation of the fault tolerance for the FTPC-Mesh network. For 6 switch failures, the network is still able to achieve a 60% fault tolerance.
V. PERFORMANCE EVALUATION

In this section we evaluate the new topology and the injection algorithm at the network interface. We compare the topology with the 2D-Mesh and C-Mesh topologies. DOR is used in all the topologies and with the same number of virtual channels (VCs). VCs are used to avoid protocol-level request-reply deadlocks induced by the cache coherency protocols.

In all the topologies switches (including clock signal) are powered on and off, following the algorithm provided previously. The number of cycles for powering switches on and off has been obtained from [8]. The delay to power on the switch again (including the clock signal) is fixed to 3 cycles. The threshold to decide a switch needs to be powered off will depend on the current traffic and will be applied on a per switch basis (distributed and local off actions). In a first analysis we evaluate the robustness of the thresholds used in the injection algorithm ($T_{th}$) and the threshold used in the switches ($T_{swoff}$). This analysis will allow us to fix those thresholds. Once fixed, we analyze the different topologies under a wide range of traffic, with synthetic traffic patterns, and real applications. However, before analyzing performance and power consumption results we provide implementation results.

A. Implementation Results

The network interface and the described switch have been designed and synthesized with 45nm Nangate opensource library. Switches with different number of ports have been designed, all of them with the same target frequency (1GHz). Total power consumption has been obtained using the Power Compiler tool from Synopsis after place and route using Encounter Place&Route tool from Cadence. Results are shown in Table I. Dynamic power has been measured for different traffic loads (from no load to one flit/cycle/input port). Results did not vary significantly as the static power is the main contributor. Furthermore, we have included the average power consumption in the results shown in the table. We use this model when analyzing the power requirements of the different topologies (including the link power). These results have been compared with the Orion-2 power model [10].

As can be deduced from Table I, additional power consumption at the network interface (due to the injection algorithm) is negligible when compared with the switch power consumption. Power consumption of an 8-port switch (required for the C-Mesh and PC-Mesh topologies) almost doubles when compared to a 5-port switch (2D-Mesh topology). This is mainly due to the increment of the buffer resources.

Table II shows the latency and area overheads of the injection algorithm at the network interface as well the switch designs. As noticed, the delay of the injection algorithm is quite far from the bounds set by the switches. Thus, the injection algorithm does not set a bottleneck along the packet’s critical path.

B. Simulation Model

We provide two sets of results. First, we provide results for synthetic traffic patterns. With this set we provide the range of benefits of the new topology for different injection rates and scenarios. We have developed an in-house network simulator (gNoCsim simulator). gNoCsim is an event-driven cycle-accurate network simulator that models the pipelined structure of the gNoC switch (designed in Verilog) and the network interface developed for the topology. We evaluate uniform, bit-complement and bit-reversal traffic patterns.

Besides, using synthetic traffic we have used SIMICS [13] and GEMS [12] to model a complete system. GEMS (General Execution-driven Multiprocessor Simulator) is a subset of Virtutech Simics modules that enables detailed simulation of multiprocessor systems, including Chip-Multiprocessors (CMPs). Simics is a full system simulation platform, capable of simulating high-end systems with sufficient fidelity and speed to boot and run operating systems and real applications. One of the most important modules of GEMS is Ruby. The GEMS Ruby module provides a detailed memory system simulator. In our case, we have replaced the network simulator inside Ruby by gNoCsim.

In GEMS/SIMICS, each end node includes an in-order processor core, an L1 data and instruction cache, an L2 cache bank, and a directory/memory controller. All the elements are connected to the network interface that connects to the NoC. In the PC-Mesh topology case, there are four internal ports (instead of one as in the 2D-Mesh), each connecting to a
different switch/subnetwork (all of them have been modeled). Delay of links connecting end nodes are modeled with one cycle in the 2D-Mesh case and with 2 cycles in the rest of topologies. Previously, in Section II we have described the logic required by the PC-Mesh at the end node to inject a flit. Table III shows the main end node parameters and cache coherency protocol parameters (obtained based on the Sun Niagara Multiprocessor). A MESI directory-based protocol is used where five virtual networks are required (to avoid protocol-level deadlock). Table IV shows the network parameters. Flit size has been set to 8 bytes. For internal links (links connecting end nodes with switches) and external links (links connecting switches) one cycle delay is assumed for the 2D-Mesh topology and 2 cycles for the rest of the studied topologies. Buffer size at switches is set to 10 flits and one cycle is assumed for each switch stage. Several Splash-2 [25] applications have been run, whereby only the parallel section has been measured: FFT, RAYTRACE, BARNES, LU, and RADIX.

### TABLE III
**END NODE AND CACHE COHERENCY PROTOCOL PARAMETERS.**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1 instructions size</td>
<td>128 KB private</td>
<td>L1 hit latency</td>
<td>3 cycles</td>
</tr>
<tr>
<td>L1 data size</td>
<td>128 KB private</td>
<td>L2 hit latency</td>
<td>6 cycles</td>
</tr>
<tr>
<td>L2 size</td>
<td>8 MB shared</td>
<td>Virtual Networks</td>
<td></td>
</tr>
<tr>
<td>Coherence protocol</td>
<td>MESI Directory-based</td>
<td>Processors</td>
<td>16 and 32</td>
</tr>
</tbody>
</table>

### TABLE IV
**NETWORK PARAMETERS.**

<table>
<thead>
<tr>
<th>General network parameters</th>
<th>Value</th>
<th>Switch parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flit size</td>
<td>8 bytes</td>
<td>Buffer size</td>
<td>10 flits</td>
</tr>
<tr>
<td>Externals links</td>
<td>1c (2d-mesh) / 2c (rest) delay</td>
<td>VCs</td>
<td>1</td>
</tr>
<tr>
<td>Internal links</td>
<td>1c (2d-mesh) / 2c (rest) delay</td>
<td>Routing</td>
<td>1c delay</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Crossbar</td>
<td>1c delay</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transmit</td>
<td>1c delay</td>
</tr>
</tbody>
</table>

C. Threshold Analysis

The PC-Mesh network has two key threshold parameters. $T_h$ is used on every end node to decide which subnetwork use to inject a message. Whenever a queue reaches that threshold the next subnetwork is set as a candidate to be used. This threshold is analyzed under four different values: 1, 50, 100, and 200. The threshold is measured as flit occupancy of the injection queue. The second parameter ($T_{swoff}$) relates to the switches being off. It indicates the number of cycles the switch needs to have all its queues empty in order to be switched off. We evaluate three values for this threshold: 1, 5, and 10 cycles. Figure 5 shows the performance and power consumption when using uniform distribution of messages as the traffic pattern in a $4 \times 4$ and a $8 \times 4$ network. Twelve combinations of threshold values have been explored. Each case is labelled with the $T_h$ value followed by the $T_{swoff}$ value.

As can be seen, performance (latency and network throughput) is largely insensitive to the threshold values. Only marginal differences are seen for very high traffic loads. In any case, the network is always ready to accept messages (until saturation is reached) and the mechanism to switch on the network components is efficient. Thus, no delays are incurred. However, differences are much greater in terms of power consumption. Indeed, for very low traffic conditions the threshold selection impacts power consumption heavily. The $T_{swoff}$ parameter is the one that impacts power consumption in very low traffic conditions. Indeed, a threshold value of 1 allows to save the maximum power from the switches. A threshold of 1 achieves 100% power reduction when compared with a threshold of 10. The value of 5 for $T_{swoff}$ lies in between, as expected. For the $T_h$ value we can see that for the same $T_{swoff}$ value all perform the same both in performance and in power consumption, No entiendo bien lo siguiente en
Energy consumption

The previous results clearly indicate the PC-Mesh topology behaves like the C-Mesh network. This is due to the low injection rates of the tested applications. In order to test the network in a much stressed scenario, and to identify the potentials of the proposed networks, we evaluate every topology with a background synthetic traffic in addition to the application's traffic. This traffic model represents a more stressed system with the exception of the value of 1. In this case there is an increase in the power consumption around an injection traffic of 0.29 flits/cycle/tile. Therefore, this threshold (used to select the network to inject) needs to be increased. Starting with a value of 50 the power saving is maximum and constant, regardless of the $T_h$ value.

One special case is the 0.29 injection rate point where power consumption per flit slightly increases. At this point switches are toggling between on and off modes thus incurring in a small penalty. Prior to this injection point only the first sub-network is working and the rest of sub-networks are down. After this critical injection point all the sub-networks are stable and working because of the higher traffic demand.

From these results we could conclude the best threshold values would be 50 no seria 200? for $T_h$ and 1 for $T_{swoff}$. However, notice that the injected traffic is steady and does not have burstiness. In order to better assess the correct threshold values we also run an application (FFT) with different threshold values, and compare the results with the C-Mesh network also with varying $T_{swoff}$ values (notice C-Mesh has only one injection link thus there is no $T_h$). Figure 6 shows the results for the FFT application, both the execution time and the power consumption. As can be seen, the results with $T_{swoff}$ set to 1 are the ones that achieve the largest amount of power saving. However, for $T_h$ we get differing results. Indeed, FFT consumes the same power in the network with values of 50 and 100, which are more than a value of 200. This is due to the burstiness of the application that leads to opening many subnetworks thus having larger consumption levels. From these results, and in combination with synthetic traffic results we conclude that a safe value for $T_h$ is 200.

**D. Synthetic Traffic Results**

Figure 7 shows the performance and power results for a 16-tile system ($4 \times 4$ topology) under synthetic traffic patterns (uniform, bit-reversal, and bit-complement). Figures a, b, and c show accepted traffic and figures d, e, and f show the power consumption per flit unit (namely, overall power consumed divided by accepted traffic).

First thing to note is the higher throughput of both PC-Mesh and FTPC-Mesh, in the three scenarios. The PC-Mesh and FTPC-Mesh topologies exhibit shorter paths and provide parallel paths not available in the 2D-Mesh network, thus lowering contention in high traffic conditions. On the other hand, the limited bisection bandwidth of the C-Mesh topology limits its maximum throughput, low below the 2D-Mesh network (except for the bit-reversal traffic where is almost equal).

Now, looking into the power consumption figures, we can see how PC-Mesh, FTPC-Mesh and C-Mesh topologies exhibit a similar power consumption rate per accepted traffic unit. The Mesh topology achieves worse results due to its higher number of switches traversed. Also, at high injection rates, both PC-Mesh and FTPC-Mesh obtain in some cases a small reduction in power when compared with the 2D-Mesh and C-Mesh topologies. Anyway, we can conclude that PC-Mesh and FTPC-Mesh exhibit the same degree of power efficiency (per delivered flit) as C-Mesh. It is worth mentioning that this does not mean all the topologies deliver the same amount of traffic (as seen in the previous figures). Similar results for 32-tile systems ($4 \times 8$ topology) have been achieved (not shown).

**E. Application Execution Time and Power Results**

In this section we evaluate application’s performance, and power savings, when real applications are run. Figure 8 shows normalized execution time and network energy consumption of different applications when using different topologies for the 16- and 32-tile configuration. The execution time is reduced when using the C-Mesh, PC-Mesh, FTPC-Mesh. On average, a reduction of 20% is achieved when compared with the 2D Mesh topology. This is mainly due to the reduction in the path length. The take away message here is that performance of the application is not sacrificed in the new topologies. The benefit comes also with the network energy consumption values. These results demonstrate the on/off mechanism can be used effectively in the new topologies, together with the injection algorithm at the network interfaces. Average results indicate savings of 20% when compared with the 2D-Mesh topology.

**F. Results with Overloaded Systems**

The previous results clearly indicate the PC-Mesh topology behaves like the C-Mesh network. This is due to the low injection rates of the tested applications. In order to test the network in a much stressed scenario, and to identify the potentials of the proposed networks, we evaluate every topology with a background synthetic traffic in addition to the application’s traffic. This traffic model represents a more stressed system.
Fig. 7. 16-node synthetic traffic comparison. Accepted traffic measured as flits/cycle/tile and power consumption as $W$.

Fig. 8. Application execution time and network energy consumption.

where multiple applications will be running at the same time. In such scenario the network will need to facilitate much higher bandwidth. Figure 9 shows the performance and power consumption values for both 16-node and 32-node systems. Background traffic of 12% (for 16-node systems) and 6% (for 32-node systems) is injected. As can be seen, the C-Mesh network is not able to run efficiently the application’s traffic. Application’s execution time is severely impacted and in most cases, more than doubled. On average, execution time is tripllicated. For power consumption values, on average, the C-Mesh network now achieves a 50% higher consumption values than the other topologies. The Mesh network, on the contrary achieves the same performance values, as it has much larger bisection bandwidth values. When compared with the PC-Mesh, and with the FTPC-Mesh, however, it achieves higher execution time and larger power consumption values. Therefore, the PC-Mesh topology is the one that achieves the best results.

Another interesting point is when different traffic classes can be mapped on different networks in PC-Mesh. Indeed, this is a key property that is not available in Mesh and C-Mesh topologies. To demonstrate the potential, we have analyzed the impact of mapping a congested traffic on a PC-Mesh subnetwork while injecting the full range of traffic through other networks. Figure 10 shows the performance and the energy consumption in hot-spot scenarios for a 16-node configuration using uniform traffic. As can be seen, the mapping of the hotspot traffic on a particular PC-Mesh subnetwork allows to attain much larger performance levels, even with similar energy consumption values (per accepted flit). In particular,
the Mesh network reduces maximum accepted traffic by 160% and C-Mesh by 40%, compared with the figure 7(a) (uniform traffic without a hot-spot scenario). The PC-Mesh topology does not get impacted by the hotspot scenario. Similar results have been obtained also for the 32-node configuration. These results open the door to a smart use of the four subnetworks available in PC-Mesh.

VI. RELATED WORK

Different topologies for CMPs have been proposed in the literature during the last years. Initially, designs and proposals relied on rings [16] and 2D meshes [24], [22], [23]. Efforts to reduce hop count have been performed with concentrated meshes [1] (end nodes connected to the same switch) and flattened butterfly networks [11]. Other works [6] reduce hop count by relying on the concept of express channels where a switch is connected to several switches along each direction in the 2D mesh [5]. A complete analysis and comparison of several topologies can be found in [6]. One special case is the Diagonal Mesh (DMesh) topology proposed in [9]. In such topology diagonal links are added between switches. However, all these topologies (including DMesh) rely on the fact that every end node is connected only to one switch. In the PC-Mesh topology, the end node is connected to four different switches, thus providing larger benefits.

In [1] the authors also hint the use of two concentrated meshes for performance issues. Therefore, that contribution can resemble ours. However, there are major differences. The first one relies on the fact that in [1] no details are given to the network interface connected to different switches. This is of major importance as it may lead to excessive performance bottlenecks and power consumption issues. Secondly, in [1] only two concentrated meshes are used, instead of four in the current proposal. In this paper not all the end nodes are connected to all the concentrated meshes (in the first layout proposal). As a major difference, in this paper we propose a power-aware selection algorithm together with an on/off switch mechanism both working cooperatively, as the target is power consumption savings without excessive end-to-end latency penalty. In [26] an end node is connected to two switches in a 2D mesh configuration. The main goal of such approach is for fault-tolerance and not for power saving. In our case, our goal is to reduce power consumption without degrading performance. In addition, in our fault-tolerant proposal the end node is connected to four neighboring switches each belonging to a different independent network. Besides, latency reduction is higher in PC-Mesh as links bypass intermediate switches.

Power gating (gated-Vdd) is a well-known technique to reduce static power consumption. In [17] a circuit technique was proposed to disconnect (by using a gating transistor) the power supply. Power gating can be applied at different levels, from complete execution units [8] down to single SRAM cells [4]. For on-chip networks different works applied the power gating technique. In [4] buffers are power gated and different policies are proposed. In [14], [15] power gating is applied to virtual channels. In [19] powering down links is proposed. In [7] static power consumption is reduced by using the concept of on/off links [19] with power-aware buffers [4], [14], [15]. The proposed power management algorithm in this paper uses also the concept of on/off links. The different proposals for power gating usually rely on a standard 2D-Mesh topology, and potentially can be applied to the proposed topology in this paper.

VII. CONCLUSIONS

In this paper we have proposed two alternative topologies to address the increasing network power consumption in CMP systems. The proposed topologies rely on extending the connectivity of the nodes to different subnetworks. Parallel networks allow an efficient on/off mechanism to cooperatively
work with an injection algorithm at the network interface. Switches are powered down in a distributed way and subnetworks are used based on the traffic injection requirements. When using the parallel? concentrated mesh as a subnetwork, network latencies are kept low, when compared with the 2D-Mesh topology. Furthermore, a tile design has been proposed to allow the implementation of four parallel concentrated meshes, forming the derived PC-Mesh network. Also, fault-tolerance is extended by enhancing the PC-Mesh network with an additional row and column of switches. Experimental results both with synthetic traffic patterns (exhibiting the full range of benefits) and with real applications, both in 16- and 32-tile configurations demonstrated the high benefits of the new topologies, achieving large savings in network power consumption without increasing path lengths and execution time of applications.

As future work we plan to extend the PC-Mesh concept to different and complementary subnetworks that can provide additional features, like increased path redundancy, different path lengths, and quality of service. Also, we will compare the PC-Mesh topology with other topologies like rings, torus, etc.

ACKNOWLEDGMENT

This work was supported by the Spanish Ministerio de Ciencia e Innovación (MICINN), and jointly financed with Plan E funds, under Grant TIN2009-14475-C04-01.

REFERENCES

[10] Andrew Kahng, Bin Li, Li-Shiuan Peh and Kambiz Samadi, "ORION 2.0: A Fast and Accurate NoC Power and Area Model for Early-Stage Design Space Exploration," in Design Automation and Test in Europe (DATE), Nice, France, April 2009.