Tiny – optimised 3D mesh NoC for area and latency minimisation

C. Marcon, T. Webber, R. Fernandes, R. Cataldo, F. Grando, L. Poehls and A. Benso

Tiny is a scalable and efficient three-dimensional (3D) network-on-chip (NoC) designed to reduce latency and area. A theoretical analysis demonstrates its efficiency when compared with a basic 3D mesh NoC. Mapping independent traffics with different injection rates makes the trade-offs analysis of Tiny possible. Results highlight that Tiny always reduces area and for several cases minimises latency.

Introduction: Multiprocessor system-on-chip (MPSoC) is composed of processing elements (PEs) containing a communication architecture that allows their joint operation [1]. Network-on-chip (NoC) is a scalable architecture that provides efficient parallel communication [2]. NoC-based MPSoC is a powerful target architecture that offers the high performance required by some recent applications. Three-dimensional (3D) NoC connects the PEs arranged in more than one silicon plane through routers and links. It reduces the communication distance, but increases throughput, minimising both the latency and the execution time of applications [3]. This Letter presents a novel topology exploration for the 3D mesh NoCs, increasing the number of PEs on a basic 3D structure. The approach enables a significant reduction in the NoC area and the latency by using a new addressing model based on straightforward deterministic *XYZ* routing.

Tiny architecture: Tiny extends a straightforward 3D direct mesh NoC called Lasio [4], reducing the area and the latency with very low extra costs. The main idea is to keep the topology and the layout as simple and regular as possible, achieving a highly scalable network. Tiny provides more PE communications than a basic direct mesh topology [5]. The proposed NoC is able to connect multiple PEs per router by using ports of routers placed on the NoC surface, implying different PE placement according to the router's physical location. Fig. 1 shows that the capacity of the PEs jumps from 8 to 32 when comparing a $2 \times 2 \times 2$ Tiny with a $2 \times 2 \times 2$ Lasio.

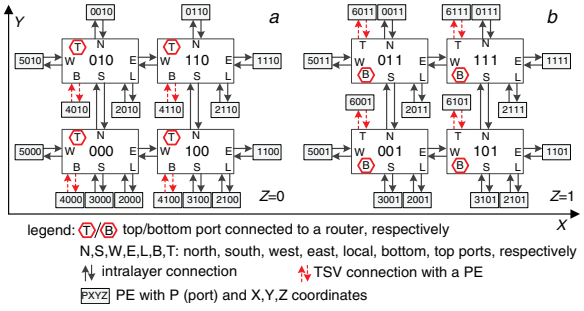

Fig. 1 *2 × 2 × 2 Tiny with layers*
a Bottom
b Top

The address of a router's port comprises the router's *XYZ* coordinates and an extra ID field (i.e. 0 – North, 1 – East, 2 – Local, 3 – South, 4 – Bottom, 5 – West and 6 – Top). Each PE assumes the address of the port to which is connected. Fig. 1 depicts the PEs addressing in a $2 \times 2 \times 2$ Tiny, e.g. address 6101 is the PE's place on the Top port of the router 101. Tiny extends the Lasio routing adding only the port code field. Thus, the internal routers of Lasio and Tiny are the same, which allows us to implement a heterogeneous Tiny (i.e. composed of the Lasio and the Tiny routers), further reducing its area.

Theoretical comparison of Lasio and Tiny regarding PE quantity: The Lasio topology has a single PE connected to each router. Thus, (1a) states that the quantity of Lasio's PEs (qPE_L) is directly proportional to the *X*, *Y* and *Z* dimensions. Compared with Lasio, Tiny supports more PEs (qPE_T) connected to the unused links of the border routers. Hence, (1b) shows that the differential increase in the quantity of the PEs is proportional to the NoC surface.

$$qPE_L = X \times Y \times Z \tag{1a}$$

$$qPE_T = 2 \times \big((X \times Y) + (X \times Z) + (Y \times Z)\big) + qPE_L \tag{1b}$$

Fig. 2 demonstrates the high impact of Tiny for small cubic NoCs (i.e. *X*, *Y* and *Z* dimensions equal to *N*) regarding the percentage and its reduction with increasing NoC size, while its absolute quantity increases. For instance, $N = 2$ implies a PE gain of 300%, since Tiny allows us to connect 32 PEs, whereas Lasio allows us to connect 8 PEs only. Furthermore, $N = 10$ implies that the gain is reduced to 60%, which means 600 additional PEs.

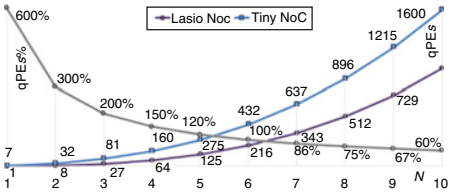

Fig. 2 *Quantity of PEs of Tiny and Lasio with N variation*

Theoretical comparison of Lasio and Tiny regarding latency: The latencies comparison uses the average number of hops of all communications ($\bar{\eta}$) allowed by each topology, representing these quantities by η_L and η_T, respectively. Equation (2) computes η_L by performing six nested sums. The first three sums represent the source PE address (x, y, z), whereas the last three sums represent the target address (i, j, k)

$$\eta_L = \sum_{x=0}^{X-1}\sum_{y=0}^{Y-1}\sum_{z=0}^{Z-1}\sum_{i=0}^{X-1}\sum_{j=0}^{Y-1}\sum_{k=0}^{Z-1} \big(|x-i| + |y-j| + |z-k| + 1\big) \quad \forall\,(x,\ y,\ z) \neq (i,\ j,\ k) \tag{2}$$

Similarly, (3) computes η_T including (4) to capture the optimisation of the hops caused by clustering the PEs into the routers. Equation (4) employs (5a) to compute the effect of a communication among the PEs connected in the ports of the same router and (6) employs (5b) to compute the quantity of the PEs linked to each border router, respectively

$$\eta_T = \sum_{x=0}^{X-1}\sum_{y=0}^{Y-1}\sum_{z=0}^{Z-1}\sum_{i=0}^{X-1}\sum_{j=0}^{Y-1}\sum_{k=0}^{Z-1}\big(|x-i| + |y-j| + |z-k| + 1\big) \times \mathrm{oE}(x,\ y,\ z,\ i,\ j,\ k) \tag{3}$$

$$\mathrm{oE}(x,\ y,\ z,\ i,\ j,\ k) = \mathrm{qP}(x,\ y,\ z) \times (\mathrm{qP}(i,\ j,\ k) - \mathrm{eql}(x,\ y,\ z,\ i,\ j,\ k)) \tag{4}$$

$$\mathrm{eql}(x,\ y,\ z,\ i,\ j,\ k) = \begin{cases} 1, & (x,\ y,\ z) = (i,\ j,\ k) \\ 0, & \text{else} \end{cases} \tag{5a}$$

$$\mathrm{brd}(n,\ l) = \begin{cases} 1, & n = 0 \text{ or } n = l-1 \\ 0, & \text{else} \end{cases} \tag{5b}$$

$$\mathrm{qP}(a,\ b,\ c) = \mathrm{brd}(a,\ X) + \mathrm{brd}(b,\ Y) + \mathrm{brd}(c,\ Z) + 1 \tag{6}$$

Equation (7a) computes $\bar{\eta}$ for both the NoC topologies dividing η by the quantity of all the possible communications among all PEs. Furthermore, η is η_L or η_T depending on the NoC topology. Hence, for both the NoCs, (7b) represents the number of the clock cycles spent in a transmission of a flit that is delayed by $\bar{\eta}$ routers (t_r), the $\bar{\eta} - 1$ inter-router links (t_{lr}) and the two local links (t_{lp}) (i.e. PE-router link). It is remarked that (7b) is valid, even if a communication occurs between two PEs placed in the same router

$$\bar{\eta} = \frac{\eta}{qPE \times (qPE - 1)} \tag{7a}$$

$$\overline{n_{CK}} = \bar{\eta} \times t_r + (\bar{\eta} - 1) \times t_{lr} + 2 \times t_{lp} \tag{7b}$$

Fig. 3 shows the average of the clock cycles ($\overline{n_{CK}}$) of both the NoCs according to the quantity of the PEs (i.e. some NoC sizes) by using (7b) and having the following assumptions supported by the NoCs implementation: (i) critical path delay implies the same clock frequency

operation; (ii) credit-based flow control transmitting a flit per clock cycle in each link (i.e. t_{lr} and t_{lp}); and (iii) routing taking on average four clock cycles (i.e. t_r). This experiment applies the fully populated NoCs; thus, the same quantity of the PEs does not match for some NoC sizes. In such cases, we use the nearest quantity of the PEs (e.g. 270 PEs for Lasio and 275 for Tiny).

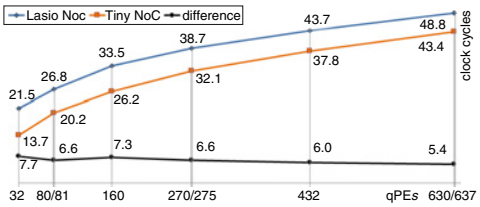

Fig. 3 *Comparison of theoretical NoC latencies with respect to amount of PEs*

Fig. 3 depicts that Tiny compared with Lasio always minimises the theoretical model's latency. Note that this work does not consider traffic contention, task/PE mapping and traffic injection on analysis. The latency gains tend to be less significant as the NoC size increases, which is reasonable due to the proportional reduction of the border routers compared with the internal routers.

Lasio has more links and more buffers compared with Tiny, while Tiny has shorter average communication paths. These features make Lasio more appropriate to minimise the packets contention. On the contrary, Tiny provides efficient communication for low concurrent traffic. In addition, a crossbar circuit implements port switching allowing full parallel communication. The PEs of Tiny connected directly to the same router may communicate without delaying other communications. This feature highlights the importance of PE/task mapping when applying Tiny.

Experimental results: Packet latency is highly dependent on mapping. Hence, the experiments explore two variations of all-to-all uniform traffic [5], whose symmetry enables evaluation of the latency independent of the mapping: (i) the quantity of the PEs that are simultaneously injecting packets and (ii) the injection rate of these packets. Fig. 4 summarises the simulation results considering three different sizes of both the NoCs with a 16-bit size flit and an 8-bit depth buffer: (i) qPE = 32 referring to the 32 PEs of $2 \times 2 \times 2$ Tiny or $4 \times 4 \times 2$ Lasio; (ii) qPE = 80/81 denoting the 80 PEs of $5 \times 4 \times 4$ Lasio and the 81 PEs of $3 \times 3 \times 3$ Tiny; and (iii) qPE = 160 referring to the 160 PEs of $4 \times 4 \times 4$ Tiny or $8 \times 5 \times 4$ Lasio.

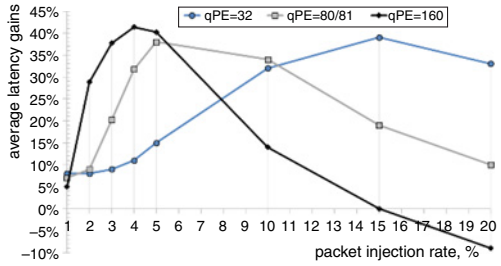

Fig. 4 *Average packet latency gains of Tiny compared with Lasio*

Finally, Fig. 4 shows that for the very low injection rates, implying few packet concurrences, the results are near the theoretical ones. When the packet injection rate increases, the reduction of the latency becomes evident for Tiny. For instance, with 4% of the packet injection rate Tiny reduces more than 40% on the average packet latency. The latency gains against the injection rate vary according to the NoC size. Although the experiment qPE = 32 shows a maximum gain of 15% regarding the injection rate, the experiment qPE = 160 reaches the maximum gain of 4%.

We remark that for all experiments the increase of the packet injection rate reduces the latency gains because augmenting the quantity of the packets results in more concurrence between resources (i.e. Tiny contains less links and buffers than Lasio). However, application task/PE mapping may eliminate, or at least, minimise this shortcoming.

Conclusion: By comparing Tiny with a straightforward 3D mesh NoC such as Lasio, it is possible to conclude that: (i) Tiny always reduces the overall area, even with a small area increase of border routers and (ii) Tiny reduces the average packets latency when there are few concurrent communications and/or a low packet injection rate. In contrast, when the packet injection rate increases and there are huge concurrent communications, Lasio provides a lower average packet latency due to the larger quantity of the buffered paths.

1 August 2013
doi: 10.1049/el.2013.2557
One or more of the Figures in this Letter are available in colour online.

C. Marcon, T. Webber, R. Fernandes, R. Cataldo, F. Grando, L. Poehls and A. Benso (*PUCRS University, Ipiranga, 6681 Porto Alegre, Brazil*)

E-mail: cesar.marcon@pucrs.br

References

1 Wolf, W.: 'The future of multiprocessor systems-on-chips'. Design Automation Conf., San Diego, CA, USA, July 2004, pp. 681–685
2 Bjerregaard, T., and Mahadevan, S.: 'A survey of research and practices of network-on-chip', *ACM Comput. Surv.*, 2006, **38**, (1), pp. 1–51
3 Feero, B., and Pande, P.: 'Networks-on-chip in a three-dimensional environment: a performance evaluation', *IEEE Trans. Comput.*, 2009, **58**, (1), pp. 32–45
4 Ghidini, Y., *et al.*: 'Buffer depth and traffic influence on 3D NoCs performance'. IEEE Int. Symp. Rapid Syst. Prot., Tampere, Finland, October 2012, pp. 9–15
5 Duato, J., Yalamanchili, S., and Ni, L.: 'Interconnection networks: An engineering approach' (Morgan-Kaufmann Press, San Francisco, CA, USA, 2003)