Traffic Configuration for Evaluating Networks on Chips

Zhonghai Lu and Axel Jantsch
Laboratory of Electronics and Computer Systems
Royal Institute of Technology, Sweden
{zhonghai,axel}@imit.kth.se

Abstract

Network-on-Chip (NoC) provides a network as a global communication platform for future SoC designs. Evaluating network architectures requires both synthetic workloads and application-oriented traffic. We present our traffic configuration methods that can be used to configure uniform and locality traffic as synthetic workloads, and to configure channel-based traffic for specific application(s). We also illustrate the significance of applying these methods to configure traffic for network evaluation and system simulation. These traffic configuration methods have been integrated into our Nostrum NoC simulation environment.

1 Introduction

One crucial aspect of Network-on-Chip design is to determine its network architecture [1]. It is challenging mainly due to two facts. One is that there exists a huge network design space with respect to topology, routing and flow control schemes etc. The other is that the network design should be customized for applications or a class of applications under current consideration and even for future upgrades or extensions. It is therefore very important to evaluate the network extensively in order to make right decisions on the network architecture.

Network evaluation commonly employs two kinds of traffic [2]. One is application-driven traffic, and the other synthetic traffic. Application-driven traffic models the network and its clients simultaneously. This is based on full-system simulation and communication traces. Full-system simulation requires building the client models. Besides, the feedback from the network influences the workload. Alternatively, execution traces may be recorded in advance and then replay this sequence for the network simulation. Application-driven traffic can be too cumbersome to develop and control. Synthetic traffic captures the salient aspects of the application-driven workload but can also be more easily designed and manipulated. Because of this, synthetic traffic is widely used for network evaluation.

In this paper, we present our traffic configuration schemes for the evaluation of networks on chips. Our contributions are (1) a unified representation for describing synthetic traffic. This representation is used to construct both uniform and locality traffic. The latter is essential to capture the traffic characteristics that explore communication locality for performance enhancement and energy saving; (2) a method to configure *application-oriented* traffic. Application-oriented traffic can be viewed as a traffic type between application-driven traffic and synthetic traffic. The spatial pattern of the application-oriented traffic reflects the communication distribution of the application(s). The temporal and message size specification may be synthetic or characterized from execution traces.

The rest of the paper is structured as follows. Section 2 briefs related work. In Section 3, we first describe the traffic configuration tree by which we introduce traffic characteristic parameters, then we detail the unified synthetic traffic representation and application-oriented traffic configuration. The experiments of applying the traffic configuration methods are reported in Section 4. Finally we conclude the paper in Section 5.

2 Related Work

In the communication community, traffic traces from physical networks are usually collected and analyzed to detect, identify, and quantify pertinent characteristics. For example, scale-invariant burstiness or self-similarity is an ubiquitous phenomenon found in diverse context, from LANs and WANs to IP and ATM protocol stacks [4].

For domain-specific applications, researchers use analysis or characterization methods to model the traffic of applications. In [3], a method is proposed to create abstract instruction-level workload models from source code for simulating application domain-specific multi-processor systems. It is shown in [8] that the traffic of multi-media applications has self-similarity characteristics.

There exist a variety of sources for application-driven workloads. Parallel computing benchmarks such as SPLASH [9] or database benchmarks are possible tests for processor interconnection networks. The construction of synthetic workloads for network simulation can be found in [2].

3 Traffic Configuration

3.1 The traffic configuration tree

Network messages (traffic) can typically be characterized and constructed by considering their distributions along the three dimensions: spatial distribution, temporal characteristics, and message size specification. The spatial distribution gives the communication partnership between sources and destinations. The temporal characteristics describe the message generation probability over time. The size specification defines the length of communicated messages. We use a traffic configuration tree to express the elements and their attributes of traffic in Figure 1.

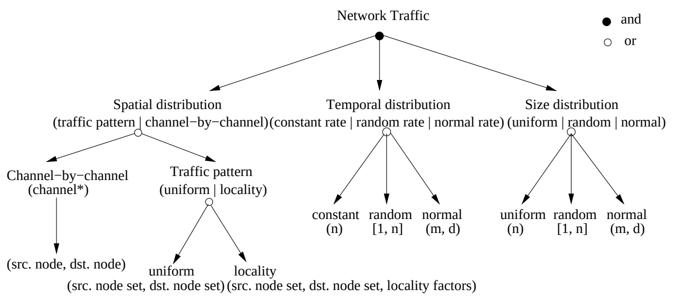

Figure 1. Traffic configuration tree

By the spatial distribution, traffic is classified into two categories: *traffic pattern* and *channel-by-channel* traffic. Traffic patterns consist of uniform and locality traffic. Each simulation cycle, the destinations of a traffic pattern may vary. That is to say, the same source node may send messages to a different channel. [1] With a traffic pattern, all the channels share the same temporal and size parameters. In contrast, channel-by-channel traffic consists of a set of channels with each channel taking its own temporal and size parameters. In addition, channel-by-channel traffic statically defines communication channels before simulation starts, implying that the source and destination nodes are fixed during the whole network simulation.

The temporal distribution has a list of items such as constant rate (periodic), random rate, and normal rate etc. The size distribution has a list of items such as uniform, random, and normal. As can be observed, these lists are just examples of possible distributions. Other interested distributions can be integrated into the tree with their associated parameters. By the tree, each traffic configuration can be set with a set of parameters.

Please note that, although we have divided the traffic configuration into three independent axes, it also allows one to configure traffic by jointly considering two axes. For example, the configuration of burstiness traffic may involve both the time and size axis.

[1] A *channel* in this paper refers to a logical path from a source node to a destination node.

3.2 Traffic patterns

3.2.1 Communication distribution probability DP

In the tree, two classes of traffic patterns are considered, namely, uniform and locality traffic. In order to build a unified expression for both traffic, we define *communication distribution probability* in relation to *communication probability* as follows:

- Communication probability of node i P_i: the probability of sending messages to the network from node i.
- Communication probability from node i to node j $P_{i>j}$: the probability of sending messages to node j from node i. For any source node i with N destination nodes numbering from 1 to N, $\sum_{j=1}^{N} P_{i>j} = P_i$.
- Communication distribution probability from node i to node j $DP_{i>j}$: the probability of distributing messages to node j from node i while node i sends messages to the network. For any source node i with its N destination nodes, we have Equation 1, meaning that all the messages from node i are aimed to all the N destination nodes in the network.

$$\sum_{j=1}^{N} DP_{i>j} = 1 \qquad (1)$$

By the definitions, we also have $P_{i>j} = P_i \cdot DP_{i>j}$.

We consider on-chip networks with regular topologies such as 2D meshes/tori, rings, trees etc. The benefit of the topological regularity is that the network nodes can be identified more structurally and with less bits. In the rest of the section, we use two-dimension topology to illustrate our representation for synthetic traffic patterns.

With two dimensions, the network topology directly maps to the Cartesian coordinate. Each network node can be identified and denoted as (x, y). For a source node (x_s, y_s), we define its *communication distribution matrix* $M_{(x_s,y_s)}$, which represents the spatial communication distribution of the node. Each item v in position (x_d, y_d) of the matrix expresses the communication distribution probability $DP_{(x_s,y_s)>(x_d,y_d)}$, i.e., from the given source node (x_s,

y_s) to the destination node (x_d, y_d). For example, the following gives the communication distribution matrix $M_{(0,1)}$ of node $(0,1)$ in a 4x3 network.

$$M_{(0,1)} = \begin{bmatrix} 0 & 0 & 0 & 0.3 \\ 0 & 0 & 0.2 & 0 \\ 0 & 0.5 & 0 & 0 \end{bmatrix}$$

From the matrix, we can see that $DP_{(0,1)>(1,0)}$=0.5, $DP_{(0,1)>(2,1)}$=0.2, $DP_{(0,1)>(3,2)}$=0.3. If an item in the matrix is zero, it means that no traffic is distributed/sent to the node from the source node.

3.2.2 Distribution coefficient *coef*

Suppose the distance between a source node (x_s, y_s) and a destination node (x_d, y_d) is d, we define communication distribution probability $DP_{(x_s,y_s)>(x_d,y_d)}$ as a relative probability to a common probability factor P_c ($0 \leq P_c \leq 1$) in Equation 2 and 3:

$$DP_{(x_s,y_s)>(x_d,y_d)} = coef \cdot P_c \quad (2)$$

$$coef = 1 + \frac{\alpha}{d+1} \quad (3)$$

where *coef* is the *distribution coefficient*; α is called *locality factor*. Since $DP_{(x_s,y_s)>(x_d,y_d)} \geq 0$, $\alpha \geq -(d+1)$. This suggests that the valid region of α depends on distance d. Particularly when $\alpha = -(d+1)$, $DP_{(x_s,y_s)>(x_d,y_d)} = 0$; when $\alpha = 0$, $DP_{(x_s,y_s)>(x_d,y_d)} = P_c$. Besides, when $-(d+1) < \alpha < 0$, $DP_{(x_s,y_s)>(x_d,y_d)}$ is proportional to the distance d; When $\alpha > 0$, $DP_{(x_s,y_s)>(x_d,y_d)}$ is inversely proportional to the distance d. Intuitively, we would have defined the distribution coefficient as $coef = \alpha/d$. We use $d+1$ instead of d in order to allow $d = 0$; We use $coef = 1+\alpha/(d+1)$ in order to incorporate the case when $\alpha = 0$. In this case, the distribution coefficient *coef* becomes independent of d, thus the traffic is uniformly distributed to nodes with a different distance.

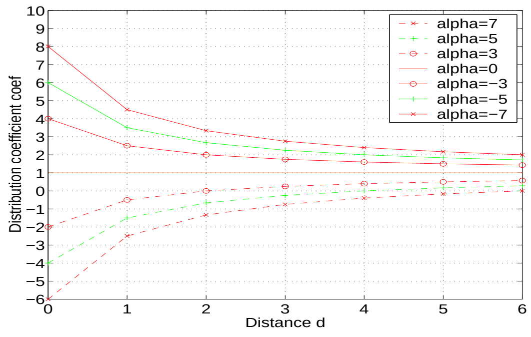

Figure 2. The distribution coefficient

By Equation 3, we depict a set of curves between *coef* and d when $\alpha = 7, 5, 3, 0, -3, -5, -7$ in Figure 2. Note that the region when $coef < 0$ is invalid region. As can be seen, the value range of *coef* differs when $\alpha > 0$ from that when $\alpha < 0$. In order for coefficient *coef* to have a symmetric range when $\alpha > 0$ and $\alpha < 0$, we constrain the distribution coefficient $coef(d)$ to be not greater than 2, then the locality factor α falls into the region $[-(d+1), (d+1)]$ and $0 \leq P_c \leq 0.5$. The distribution coefficient *coef* in relation to locality factor α is shown in Figure 3.

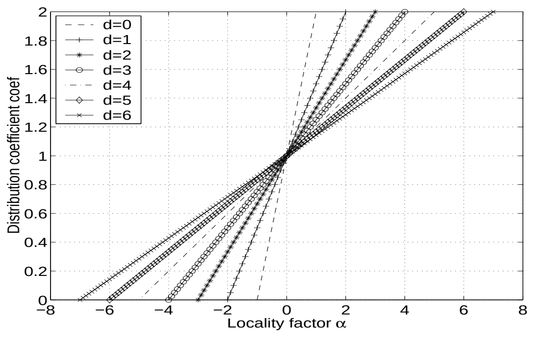

Figure 3. The locality factor and distribution coefficient

Applying the formula $\sum_{j=1}^{N} DP_{i>j} = 1$ on Equation 2, we have

$$\sum_{k=0}^{D} N_k coef_k \cdot P_c = 1 \quad (4)$$

where N_k is the number of nodes with distance k and $\sum_{k=0}^{D} N_k = N$; D is the maximum distance between the source node and the destination nodes; $coef_k$ is the distribution coefficient for a destination node with distance k.

3.2.3 Common probability factor P_c

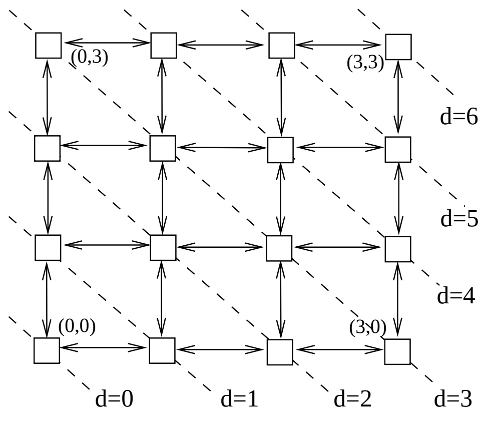

Figure 4. A distance graph

The common probability factor P_c is calculated after $\alpha(d)$ is given. This allows us to give the value(s) of α freely, ignoring the details about the exact number of destination nodes with a different distance. As this detail is dependent on the source node position in the network topology, it may vary from node to node. We use an example to illustrate how P_c is determined. Figure 4 shows a 4×4 mesh topology. We define on this mesh that $\alpha = 1$, which means α is a constant, i.e., irrespective of distance d. To determine P_c for node $(0, 0)$, we draw the dashed lines to indicate all the destination nodes with a different distance. For its distance array $[0, 1, 2, 3, 4, 5, 6]$, it has the array of the number of destination nodes $[1, 2, 3, 4, 3, 2, 1]$. By Equation 3, we obtain the coefficient array $coef(d)$ of node $(0, 0)$ as $[2, 1.5, 1.3333, 1.25, 1.2, 1.1667, 1.1429]$.

Using Equation 4, we have

$$(1{\cdot}2+2{\cdot}1.5+3{\cdot}1.3333+4{\cdot}1.25+3{\cdot}1.2+2{\cdot}1.1667+1{\cdot}1.1429){\cdot}P_c = 1$$

Solving the equation, we receive $P_c = 0.0474$. If $\alpha = 0$, $P_c = 1/16 = 0.0625$. After obtaining P_c, we can calculate the distribution probability $DP(d)$ from node $(0, 0)$ to destination node(s) with distance d by Equation 2 as an array $[0.0948, 0.0711, 0.0632, 0.0592, 0.0569, 0.0553, 0.0542]$. Clearly, for a different source node, P_c may be different even if α is the same, since its distance array and the number of destination nodes with a certain distance may be different.

Since locality factor α may be set individually with d, we can control the amount of traffic distributed to a certain distance. Again with the example in Figure 4, if we define the node $(0, 0)$'s locality factor array $\alpha(d)$ as $[-1, 0, -1.2, -2.4, -4.0, -5.4, -6.3]$, its distribution coefficient array $coef(d)$ is $[0, 1, 0.6, 0.4, 0.2, 0.1, 0.1]$. Then its $P_c = 0.1587$. As a result, the communication distribution array $DP(d)$ of node $(0, 0)$ is $[0, 0.1587, 0.0952, 0.0635, 0.0317, 0.0159, 0.0159]$.

If all the source nodes' locality factors α are zero, their distribution coefficients $coef(d)$ to all destinations are one, i.e., independent of distance d. In this case, the traffic is uniformly distributed.

3.3 Application-oriented Workloads

Channel-by-channel traffic differs from the traffic patterns in that the traffic's spatial pattern is built on per-channel basis and static. This type of traffic is used to construct application-oriented workloads for specific applications. The temporal characteristics and message size specification may be approximated using analysis or communication traces. The set of traffic parameters of a channel is $\{s\_node, d\_node, \mathcal{T}, \mathcal{S}\}$, where s_node represents the source node, d_node the destination node, $\mathcal{T}$ its temporal characteristics, and $\mathcal{S}$ is its message size specification. In the following, we use a Motion JPEG (M-JPEG) encoder to illustrate how to approximately construct its application workload, supposing that its functional blocks are mapped to network nodes in a one-to-one manner. For simplicity, we assume a single synchronous clock.

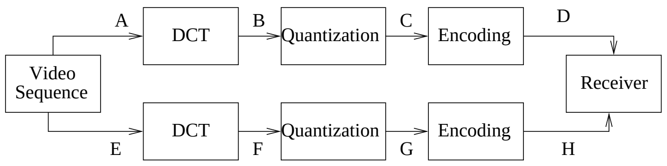

Figure 5. An M-JPEG encoder

Figure 5 shows the functional blocks of an M-JPEG encoder where each frame is separately compressed into a JPEG image [5]. The characters A, B, C, D, E, F, G, H indicate eight communication channels between modules. The M-JPEG codec is decomposed into three independent pipelined stages, namely, DCT (Discrete Cosine Transform), quantization and an encoding module. The figure consists of two parallel pipelines. Computations are performed on an 8x8 pixel block. Suppose one pixel contains 8 bits, one block has 64 bytes. The encoding module processes four blocks of data before outputting results. Due to data compression, the output from this module has a variable size. From recorded communication traces, we found it falls in the region $[16, 56]$ bytes.

Channels	Period (Cycles)	Size (bytes)
A, E; B, F; C, G;	160	64
D, H	640	[16, 56]

Table 1. Channel parameters

The DCT module is the performance bottleneck. To process one block of data, it consumes 135 cycles [6]. Assuming the maximum latency for processing one block of data over the communication channels is 25 cycles, we could consider that the critical path passes from the video sequence sender through the channel A/E and DCT. Following the worst-case style for synchronous design, we can design the period of the pipelined stages to be 160 cycles. Consequently, we may configure the traffic to model the application as follows: The channels A, B, C, and E, F, G are periodic channels with a constant period of 160 cycles and a uniform size of 64 bytes. The encoding modules may not generate output periodically. But, with the support of traffic shaping, the channel D and H can be assumed to transmit messages with a period of 640 cycles and a random size in region $[16, 56]$ bytes. We summarize the traffic parameters in Table 1.

4 Experiments

4.1 Use of the synthetic traffi c patterns

We have integrated the traffic configuration methods into our Nostrum Network-on-chip Simulation Environment (NNSE) [7]. Based on a SystemC NoC simulation kernel, this tool allows one to construct a network and traffic by using the network and traffic characteristic parameters, and then evaluate the network with the traffic.

TRAFFIC	*d*	0	1	2	3	4	5	6
Locality	α	-1	0	-1.2	-2.4	-4.0	-5.4	-6.3
	coef	0	1	0.6	0.4	0.2	0.1	0.1
Uniform	α	-1	0	0	0	0	0	0
	coef	0	1	1	1	1	1	1
Non-locality	α	-1	-1.8	-2.7	-3.2	-3	-2.4	0
	coef	0	0.1	0.1	0.2	0.4	0.6	1

Table 2. Traffic specifications

In order to understand the network behavior for locality traffic, we can create synthetic traffic using the method described in Section 3.2. We first construct a 4×4 mesh network operating synchronously. The network employs wormhole-based virtual-channel (VC) flow control with dimension-ordered XY routing, which is deterministic and deadlock free on meshes. The switch model is a single-cycle model. The number of VCs per physical channel of a switch is 4 and the depth of a VC is 2. The network diameter is 6. Then we inject one of the three classes of traffic into the network: *locality traffic*, *uniform traffic*, and *non-locality traffic*. With the uniform traffic, a network node sends packets to other nodes with the same probability. With the locality traffic, a network node sends packets to its *nearer* nodes with higher probability. With the non-locality traffic, a network node sends packets to its *further* nodes with higher probability. We list the traffic's locality factors $\alpha(d)$ and calculated distribution coefficients $coef(d)$ in Table 2. All the network nodes are both packet sources and sinks. The source nodes inject packets into the network via bounded FIFOs with a constant rate. The flits of packets are ejected from the network immediately once they reach destinations. One message contains only one packet and each packet is 4-flit long. By our configuration tree, all the traffic classes belong to periodic traffic with a uniform message size.

The left one of Figure 6 illustrates the average latency of packets. This figure demonstrates that, the more locality the traffic has, the lower average latency the network achieves. Note that, with the same injection rate, the offered load is different for the three types of traffic since the average packet distance is different. The non-locality traffic results in the largest offered load since it has the largest average packet distance. The uniform traffic is the second,

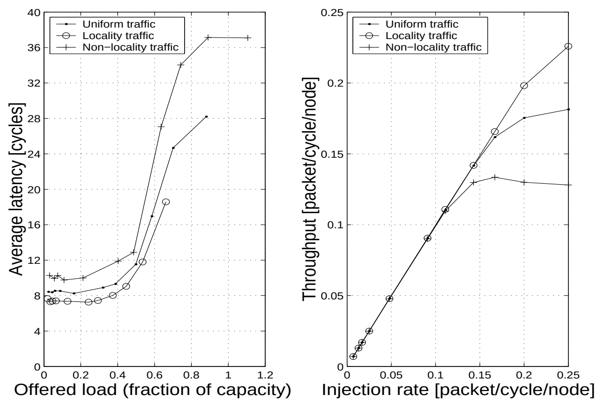

Figure 6. Performance with traffic classes

followed by the locality traffic. This also implies that the locality traffic allows higher packet injection rate before saturation. Besides, due to buffer overflow, the offered load does not increase proportionally when the injection rate is high. The right one of Figure 6 shows the throughput in relation to the packet injection rate. As can be observed, under the same amount of workload, the network can reach a higher throughput if the locality of traffic is higher.

Clearly the network shows significant performance improvement if the traffic is more locally distributed. We can conclude that, with the dimension-order routing, the wormhole network can efficiently benefit from traffic locality. Our traffic configuration method can flexibly enable to explore this by changing the traffic's locality factors.

4.2 Use of the application-oriented traffi c

In NNSE, we manually map the functional modules of the M-JPEG model in Figure 5 onto the 4×4 mesh network described above. The mapping is done in a one-to-one fashion. Messages from these modules are encapsulated into packets. One packet has a payload length of 96 bits. This means a message with size s bytes will be decomposed into $\lceil s/12 \rceil$ packets. The traffic is created and injected into the network according to Table 1. Simulation results show that the average latency of the packets is 9.68 cycles, the link utilization is 14.7%. Designers can use the simulated results to optimize the mapping or adjust the pipeline period to achieve design goals. Since the network is under-utilized, the designers may decide to shrink the network size to 3×3, or to implement more concurrent pipelines in the network.

5 Conclusions

We have proposed and demonstrated our traffic configuration schemes for evaluating networks on chips. The unified expression for configuring regular traffic patterns allows designers to adjust the locality of traffic so as to analyze the network behavior under locality traffic. The configuration of application-oriented traffic enables to take into account the spatial communication patterns of the application while generating traffic, and it maintains the flexibility of synthetic traffic. The configured synthetic and application-oriented traffic can help designers to evaluate and thus make right decisions on the network architecture. In addition, the configuration of application-oriented traffic is beneficial to obtain performance data at an early design phase, thus useful for fast design space exploration.

Future work will fledge the traffic configuration tree to include more traffic distributions. Another direction is to integrate Quality-of-Service (QoS) traffic into the framework in order to evaluate NoC architectures supporting QoS with different guarantees on throughput and delay bounds.

References

[1] L. Benini and G. D. Micheli. Networks on chips: A new SoC paradigm. *IEEE COMPUTER*, (1):70–78, 2002.

[2] W. J. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufman Publishers, 2004.

[3] J. Kreku, T. Kauppi, and J.-P. Soininen. Evaluation of platform architecture performance using abstract instruction-level workload models. In *Proceedings of International Symposium on System-on-Chip*, November 2004.

[4] K. Park, G. Kim, and M. Crovella. The effect of traffic self-similarity on network performance. In *Proceedings of SPIE International Conference on Performance and Control of Network Systems*, November 1997.

[5] S.-H. Sun and S.-J. Lee. A JPEG chip for image compression and decompression. *The Journal of VLSI Signal Processing*, 35(1):43–60, August 2003.

[6] The CS6100 M-JPEG core. www.amphion.com.

[7] The University Booth Program at the Design Automation and Test in Europe 2005. http://www.date-conference.com/conference/universitybooth.htm.

[8] G. Varatkar and R. Marculescu. Traffic analysis for on-chip networks design of multimedia applications. In *Proceedings of Design Automation Conference*, June 2002.

[9] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. The SPLASH-2 programs: Characterization and methodological considerations. In *Proceedings of the International Symposium on Computer Architecture*, June 1995.