A Set of Traffic Models for Network-on-Chip Benchmarking

Esko Pekkarinen, Lasse Lehtonen, Erno Salminen, Timo D. Hämäläinen
Tampere University of Technology, Department of Computer Systems

***Abstract*— This paper presents a set of 9 application traffic models for benchmarking Networks-on-Chip designs. Common benchmarks allow fair comparison, reproduction of research results, and accelerate NoC development. The set is based on real applications found in literature and executable on freely available benchmarking tool called Transaction Generator (TG). It was found that traffic target distribution is far from uniform and bandwidth requirements vary very much between tasks and between applications. TG and the model source codes are freely available. Models are stored in XML format and are based on task graphs with on average 15 processing tasks. The average communication load is about 2 GByte/s.**

***Index Terms*— Multiprocessor System-on-Chip, Network-on-Chip, benchmarking**

I. Introduction

A modern Multiprocessor System-on-Chip (MP-SoC) integrates many Processing Elements (PEs) into a single chip to execute complex applications, such as multimedia players or wireless communication. System design has shifted towards combining existing Intellectual Properties (IPs) from different vendors to fulfill the desired functionality. However, parallel processing increases the requirements for the Network-on-Chip (NoC) [1], [2] which provides interconnections between PEs, memories and off-chip interfaces.

A lot of research and design effort has been invested in NoCs in order to fulfill the stringent performance requirements. Optimizing the resources and application broadens the already large design space and the need for efficient tools is obvious. This applies to NoCs as well, yet they still lack common benchmarks for fair comparison of designs [3]. The problem is currently addressed by the OCP-IP Network-on-chip benchmarking workgroup which seeks to define common benchmarks and benchmarking methods for NoCs [4].

This work presents a set of traffic models and is part of the OCP-IP standardization. The overall flow is depicted in Figure 1. We analyze the traffic patterns of parallel applications based on literature and create executable models for them. The models can be used in a SystemC tool called Transaction Generator (TG) that creates traffic load to the benchmarked NoC, and for collecting statistics [5], [6].

This paper concentrates on model creation. The main contribution is a set of 9 application models for TG tool. This paper is organized as follows. In Section II, related work is presented and Section III introduces the Transaction Generator tool. Details regarding the proposed benchmarking set are given in Section V. Finally, conclusions are drawn.

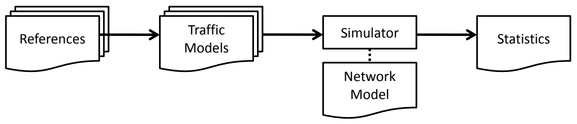

Fig. 1. Overview of traffic model flow.

II. Related work

NoC benchmarks has been divided into four categories. A *synthetic* benchmark creates traffic to NoC attempting to mimic the behavior of target application. *Algorithm-based kernels* focus on the key algorithm isolated from the application. Running the *actual application* is naturally the best possible benchmark, but most often it is not an option. Varying *combinations* of all three are also possible. So far, most NoC research has been based on proprietary test cases which complicates direct comparison and reproduction of findings [1].

Two aspects of network traffic must be considered: *spatial* (where to send) and *temporal* (when to send). Spatial target distribution can be purely synthetic, such as random, bit rotation or tornado traffic; or based on statistics from real application. Temporal properties include data rate, burstiness, and dependencies. Another approach is presented in [7] where three statistical components are used to cover the spatio-temporal traffic properties of 25 applications: *Hurst exponent H* for burstiness, p for hop distance and σ for distribution ratio of injected traffic.

Three applications from E3S benchmark suite [8] have been used earlier by Hu *et al.* for NoC study [9]. Noxim [10] is a simulator tools that allows evaluation of throughput, delay and power consumption for customised NoCs. SDF^3 [11] generates Synchronous Dataflow Graphs (SDFGs) which can be used to mimic e.g. multimedia applications with cyclic behavior. This work is based on [9], [12], [13], [14], [15], [16], [17] and more details about them are given in section V.

All works cited above are synthetic and based on real applications (except the generic cases). We use the same approach but intercommunicating tasks are used to create traffic to NoC instead of independent statistical components. Furthermore, all presented tools and traffic models are freely available for download.

III. Transaction Generator

We utilize Transaction Generator which is a freely available MP-SoC evaluation tool written in SystemC [14]. Abstract

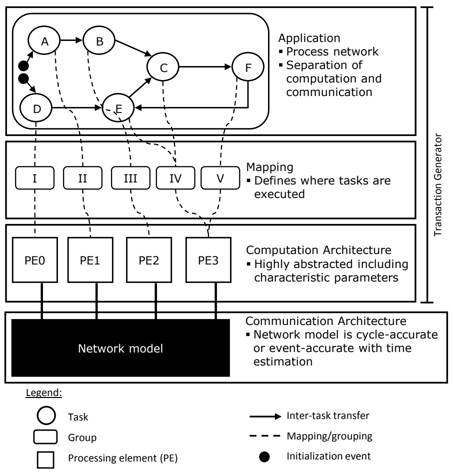
Application
▪ Process network
▪ Separation of computation and communication
Mapping
▪ Defines where tasks are executed
Computation Architecture
▪ Highly abstracted including characteristic parameters
Communication Architecture
▪ Network model is cycle-accurate or event-accurate with time estimation
Network model
Transaction Generator
A
B
C
D
E
F
I
II
III
IV
V
PE0
PE1
PE2
PE3
Legend:
Task
Group
Processing element (PE)
Inter-task transfer
Mapping/grouping
Initialization event

Fig. 2. Transaction Generator overview. Task network models application mapped on multiple PEs. An abstract network model provides the interconnection.

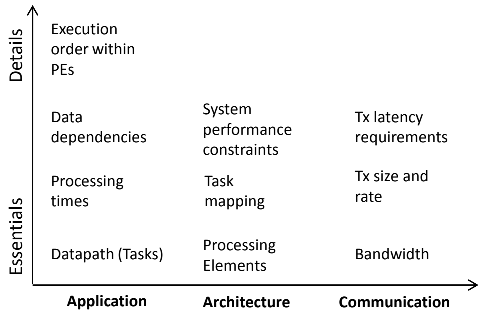

Fig. 3. Modeling parameters addressed by TG.

high level application workload model is based on task graphs and described in eXtensible Markup Language (XML) format [5]. TG creates traffic to NoC according to the description file and measures various performance metrics during simulation. The concept is shown in Figure 2.

A task is activated when it receives a data token. Then it “executes” a certain amount of computation (practically waits), and initializes data transfers to other tasks. Most transfers are simple write operations and reads are supported with memories. Tasks can perform processing and transfers simultaneously. In addition to tasks, there are timers to create stimulus to model the environment. TG automatically collects statistics regarding data amounts, latencies, PE utilization and task execution counts.

Designer can easily modify the mapping of the application tasks to the PEs which are modeled in abstract manner. Note that TG kernel is decoupled from the NoC. NoCs are interfaced with Open Core Protocol (OCP) and they can be descibed at Register Transfer Level (RTL) and Transaction Level, the latter obtaining up to 80x speedup in simulation [6]. Current implementation takes about 5000 lines of SystemC for TG and about 15000 lines of Java for optional graphical user interface.

IV. Traffic modeling

Models always impose certain inaccuracy that must be accepted,since otherwise model creation would be too cumbersome. Our earlier experiments with TG suggest that error remains mostly below 10% compared to the FPGA prototype which is an acceptable level in NoC benchmarking. The modeling parameters are depicted in Figure 3 and explained next.

A. Parameters

Based on our experience companies are not willing to share details regarding their applications or details are not gathered. Hence one must rely on literature. However, many details of test application are often omitted because they are not necessarily the main focus of the paper, for example when discussing design methodology. Modeling is possible with rather little information, at minimum only the data rates i.e. bandwidth between PEs is sufficient. However, minimal models do not allow other evaluation than basic NoC benchmarking, for example CPU utilization cannot be measured.

Computation can be refined into a set of tasks, resulting in at least one task per PE and perhaps much more. The processing time of a task on the given PE can be measured either in clock cycles or seconds. Dependencies force tasks to wait until the operands are present which makes the model more accurate. For example, two transmissions from separate tasks are required before computation can be started. TG initializes transfers at the end of the computation. In real applications this may not always be the case and data can be sent at several stages during computation. At the most accurate level the execution order of tasks within one PE is also known.

Architecture defines the set of resources, for example PEs and memories, that is absolutely mandatory information in case that task-level information is not available. Task mapping defines on which PE each task is executed. Optimal mapping is rarely intuitive and it is one of the main problems of design space exploration. Although mapping isn’t an essential part of a traffic model, it contributes to the measured performance. Some systems impose constraints to PEs, such as maximum frequency cannot exceed 200 MHz.

Bandwidth is a minimum requiement and it can be derived from other parameters if available. Exact transfer (Tx) sizes refine the communication and affect the packetization at network interfaces. Transfer sizes can be derived from bandwidth by dividing the overall bandwidth by transfer rate. The rate is based on an estimate or the requirements of the target application. Transfers may have constraints, such as maximum latency, similarly to system level constaints.

B. Model capture for TG

Figure 4 shows an example of a task graph for audio video (AV) benchmark. It has 40 tasks (nodes) and 57 unidirectional connections (edges). Bandwidth in MB/s is marked next to each edge. In [9] the application was mapped to 16 PEs and hence not all bandwidths are known. The processing times and transfer sizes are also unknown.

The XML model description is rather easily generated. The previous example is less than 2000 lines of XML and

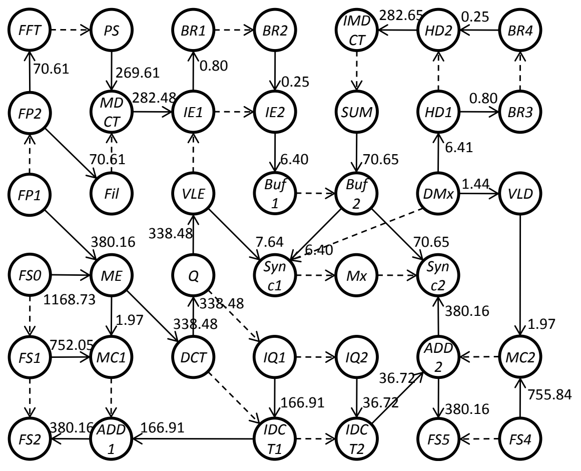

Fig. 4. Example of system [9] traffic graph. Nodes present tasks and edges unidirectional data channels. Dashed edges present transfers within a PE and therefore their bandwidths are omitted. Channel bandwidths are given in MB/s.

```
<task name="MDCT" id="39" class="general">
  <in_port id="3936"/>
  <in_port id="3938"/>
  <out_port id="3913"/>
  <trigger dependence_type="and">
    <in_port id="3936"/>
    <in_port id="3938"/>
    <exec_count>
      <op_count>
        <int_ops>
          <polynomial>
            <param value="0" exp="0"/>
          </polynomial>
        </int_ops>
      </op_count>
      <send out_id="3913" prob="1">
        <byte_amount>
          <polynomial>
            <param value="8828" exp="0"/>
          </polynomial>
        </byte_amount>
      </send>
      <next_state value="READY"/>
    </exec_count>
  </trigger>
</task>
```

Fig. 5. XML description of task MDCT. Data transfer to task IE1 (port 3913) is initialized only after data from tasks PS and Fil has been received. The task executes no computation.

creation took less than 4 hours once we had got familiar with the used notations in the original article. Description of task MDCT near top-left corner is given in Figure 5. It demonstrates the XML usage and how the total number of lines is somewhat expanded by repetitive structures. The task has two input ports and is triggered when data arrives in both of them (dependence_type=and). In this case there is no computation (int_ops value=$0 \cdot x^0 = 0$) and the task only sends 8828 bytes when activated. This model assumes that each task is executed 32000 times per second. Hence $32\ kHz \cdot 8828\ B = 282\ MB/s$.

V. Proposed Benchmarking Set

Our proposed set contains 9 traffic models that focus on multimedia and telecommunication applications derived from literature. SoC is a natural platform in both application domains. Many telecommunication applications process several parallel signal streams and both require heavy computation capacity. Both domains have real-time requirements regarding minimum bandwidth or maximum latency.

This set serves as a starting point for standardized evaluation and will undergo updating and refinement by the OCP-IP workgroup. However, it is already ready-to-use with TG as it is. Evaluator may choose to modify the platform (e.g. NoC type or frequency) but the applications should remain unchanged for benchmarking purposes.

A. *Telecommunication benchmarks*

We have selected four applications having different communication loads. Channel Equalizer presents the smallest communication of 19 MB/s in total [13]. UMTS and OFDM receivers both have modest total bandwidths, 94 and 196 MB/s respectively [15], [16]. The largest by far is an application by Ericsson Radio Systems with 4488 MB/s [12]. It is heavy in communication but exact functionality of the application is unknown.

Channel Equalizers are needed in radio receivers to correct the distortion, caused e.g. by signal reflection from buildings and vehicles. In portable systems adaptive equalization which immediately reacts to changes in channel characteristics is one of the key features.

UMTS receivers are used mostly in mobile phones. Incoming serial data is branced into parallel streams. In our models one task processes all parallel streams. Orthogonal Frequency Division Modulation (OFDM) is used for radio and television transmissions. OFDM data is transferred over several channels, each carrying different data, and receiver selects the desired channel by filtering it. Both methods enable the efficient use of the overall wireless bandwith for many simultaneous transfers.

B. *Multimedia*

Data-parallel MPEG-4 encoder is based on our earlier work [14] and the only one whose computation is known in detail and addressed by the model. One master processor assigns the computation to several identical slave processors and merges the results. The number of slaves is configurable and XML file is automatically generated. Here, values are for QCIF and 9 slave CPUs whereas having 16CIF and with 36 slaves has a total bandwidth requirement of about 170 MB/s. It is modest due to local data memories at PEs.

In other cases PEs and data rates have been reported, but their exact functionality is unknown. The MPEG-4 decoder, Multiwindow Display (MWD) and Video Object Plane Decoder (VOPD) have been used earlier for benchmarking e.g. in [17]. In the first one, communication is concentrated to/from memories whereas the two others have more pipeline style structure.

Audio video benchmark model is derived from the E3S suite [8]. The presented values in this paper are based on earlier NoC study by Hu *et al.* [9]. The structure of the task graph is one of the most complicated. Note that tasks have at most two targets which favors hierachical NoCs that can exploit locality.

TABLE I
TRAFFIC CHARACTERISTICS

#	Application	Structure				PE data rate [MB/s]				
#	Application	#tasks	#edges	avg #dst	max #dst	max eject	max emit	avg emit	Total emit	XML lines
1	Ericsson Radio System [12]	16	26	1.6	6	576	3 072	280	4 488	1 108
2	Channel Equalizer [13]	12	20	1.7	4	10	13	5	19	818
3	UMTS receiver [15], [16]	7	11	1.6	6	61	61	24	94	592
4	OFDM receiver [15], [16]	7	12	1.7	4	64	64	49	196	466
5	MPEG-4 encoder [14]	229	505	2.2	19	<1	1	<1	4	13 063
6	MPEG-4 decoder [17]	12	26	2.2	7	897	897	206	2 466	943
7	MWD [17]	12	22	2.4	4	480	480	132	1 184	682
8	VOPD [17]	12	14	1.2	2	800	500	282	3 378	712
9	AV benchmark [9]	40	57	1.4	2	1549	1169	423	6 772	1 635
	min	7	11	1.2	2	10	13	5	19	466
	avg	39	77	1.7	6	555	782	178	2 325	2 094
	max	229	505	2.2	19	1 549	3 072	482	6 772	13 063

With almost 7000 MB/s it's highly demanding application for the NoC.

C. Summary

Traffic characteristics of the released models are shown in table I. We noticed that average size of a task graph is around 15 tasks, excluding the MPEG-4 encoder that is generated with scripts. On average each task has only two targets and in 6 graphs there are tasks that have at least 4 targets. In 4 cases there is one very active source, i.e. a task sending large fraction of the whole traffic, for example 68% for the first graph. Similarly, sinks that eject large fraction of traffic, can be identified in 4 cases. This shows that traffic is far from uniform and great variation can be seen within a single application and between applications.

For complete application descriptions some general assumptions were made: Internal transfers are either minimal messages or copies of outgoing transfers. If separate tasks aren't reported, we assume one task per PE. Tasks with more than one incoming channels are triggered when data is received in all inputs. Bidirectional edges (in original article) are split into two unidirectional channels with half the bandwidth. If no apparent system input exists, tasks with no incoming channels are triggered by timers simultanuosly at short interval to run the application.

TG allows combining multiple task graphs to model heterogenous behavior. For example, one part of large MP-SoC computes telecommunication function and the other multimedia processing. Another example is that telecommunication stack changes dynamically between UMTS and WLAN which can be achieved with timers and few additional scheduling tasks.

VI. CONCLUSION

We have presented a set of traffic models for NoC benchmarking purposes. It is freely available along with the necessary tools and serves as a starting point for the OCP-IP standardization. Transaction Generator and the presented traffic models are available under Lesser General Purpose License (LGPL) at http://www.tkt.cs.tut.fi/research/nocbench/

ACKNOWLEDGEMENT

This work has been supported by the Academy of Finland.

REFERENCES

[1] E. Salminen, On Design and Comparison of On-Chip Networks, PhD Thesis, Tampere University of Technology, Publication 872, 2010, 230 pages.

[2] T. Bjerregaard and S. Mahadevan. A survey of research and practices of Network-on-chip. ACM Computing Surveys. 38 (1). pp. 1-51. 2006.

[3] R. Marculescu. U. Ogras. P. Li-Shiuan. N.E. Jerger. Y. Hoskote. Outstanding Research Problems in NoC Design: System, Microarchitecture, and Circuit Perspectives. IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol. 28. no. 1. pp. 3-21. Jan. 2009

[4] E. Salminen. K. Srinivasan and Z. Lu. OCP-IP Network-on-chip benchmarking workgroup. White paper, OCP-IP, 2010.

[5] E. Salminen, C. Grecu, T. D. Hämäläinen, A. Ivanov, Application modeling and hardware description for Network-on-chip benchmarking, IET Computers & Digital Techniques, Sep. 2009, vol. 3, no. 5, Special issue on Network-on-chip, pp. 539-550.

[6] L. Lehtonen, E. Salminen, T. D. Hämäläinen, Analysis of Modeling Styles on Network-on-Chip Simulation, in Norchip, Nov. 2010, 4 pages

[7] V. Soteriou. H. Wang. L.-S. Peh. A statistical traffic model for on-chip interconnection network. in MASCOTS. Sep. 2006. pp. 104-166.

[8] R. Dick. Embedded System Synthesis Benchmark Suites (e3s). [Online]. Available: http://ziyang.eecs.umich.edu/ dickrp/e3s/

[9] J. Hu. U. Ogras. and R. Marculescu. System-level buffer allocation for application-specific networks-on-chip router design. IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.. vol. 25. no. 12. Dec. 2006. pp. 2919-2933

[10] M. Palesi. D. Patti. F. Fazzino. Noxim the NoC simulator. [Online]. Available: http://noxim.sourceforge.net/

[11] S. Stuijk. M. Geilen. T. Basten. SDF3: SDF for Free, in ACSD, June 2006, pp. 276-278.

[12] Z. Lu. A. Jantsch. TDM virtual-circuit configuration for NoC. IEEE Trans. VLSI systems. vol. 16. no. 8. Aug. 2008. pp. 1021-1034

[13] A. Moonen. M. Bekooij. R. van den Berg. J. van Meerbergen. Evaluation of the throughput computed with a dataflow model - A case study. ES Reports. ESR-2007-01. Mar. 2007

[14] T. Kangas. K. Kuusilinna. T. Hämäläinen. Scalable Architecture for SoC Video Encoders. J. VLSI Signal Processing-Systems for Signal, Image, and Video Technology. Mar. 2006. Vol.44. pp. 79-95.

[15] P. Wolkotte. Exploration withing the Network-On-Chip Paradigm. PhD Thesis. University of Twente. CTIT Ph.D.-thesis series No. 09-133. 2008.

[16] G. Rauwerda. Multi-standard adaptive wireless communication receivers. PhD Thesis. University of Twente. CTIT Ph.D.-thesis series No. 08-109. 2007. 241 p.

[17] D. Bertozzi, A. Jalabert, S. Murali, R. Tamhankar, L. Benini and G. de Micheli. Noc synthesis flow for customized domain spesific multiprocessor systems-on-chip. IEEE Trans. Parallel Distrib. Syst. vol. 16. no 2. pp. 113-129. Feb. 2005.