VENTTI: a Vertically Integrated Framework for Simulation and Optimization of Networks-On-Chip

Young Jin Yoon, Nicola Concer, and Luca Carloni

Department of Computer Science, Columbia University
{youngjin, concer, luca}@cs.columbia.edu

***Abstract*—Networks-on-chip have been proposed to enable the integration of heterogeneous SoCs in a power-efficient and scalable way. We present VENTTI, a design and simulation environment that combines a virtualized platform, a NoC synthesis tool, and a library of NoC building blocks characterized at different abstraction levels. By efficiently simulating complex application scenarios running on top of Linux, VENTTI enables the evaluation of alternative NoC architectures and the derivation of a final RTL implementation that meets the communication requirements while optimizing power dissipation.**

Experimental results demonstrate the features of VENTTI and illustrate the trade-offs in terms of simulation time and power-estimation accuracy.

I. INTRODUCTION

The computing platforms for embedded systems are increasingly based on Systems-on-Chip (SoC), which are composed by many general-purpose processors, memories, and a growing variety of accelerators subsystems. The general-purpose cores run an operating system as well as the software of multiple applications for which the SoC does not provide specific hardware support. The accelerators execute critical computational tasks with better performance and energy efficiency than the software. The on-chip memory supports the execution of multiple parallel applications and the data exchanges among cores and accelerators [1], [2].

For many SoCs, power dissipation is increasingly the main factor that drives the whole design process. The design of power-efficient SoCs is made more challenging by the growing number of IP blocks being integrated on the chip. Meanwhile, the interconnect naturally comes to play a central role because it is in charge of supporting the communications between the multiple cores, the memory, and the accelerator subsystems.

Networks-on-Chip (NoCs) are now considered the most promising solution to overcome the limitations of traditional bus-based architectures and satisfy the communication requirements of future SoCs while guaranteeing both scalability and power-efficiency [3], [4]. As NoCs depend on the physical placement and computational properties of each IP block, their implementation is usually *tailored* to meet the specific requirements of the target SoC. This peculiar characteristic makes the NoC a particularly critical component to design and test. Additionally, the NoC implementation is open to a large set of possible design choices including: network topology, routing algorithm, flow-control protocol, end-to-end communication protocol, presence of virtual channels, as well as the specifics of the router micro-architecture (with such properties as the size of the buffering queues and the port parallelism, or flit width). Each one of these design points has an impact on the final performance and power dissipation of the NoC.

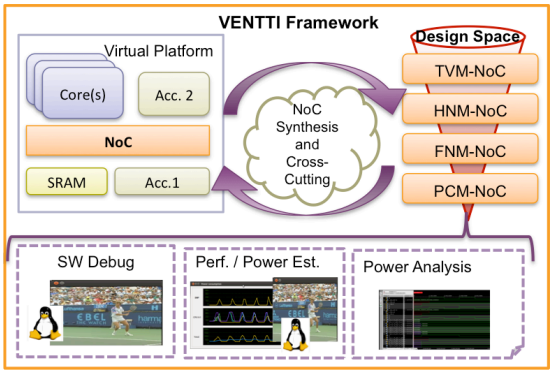


Fig. 1. The proposed VENTTI framework.

To assist engineers in the exploration of this rich and complex design space we introduce VENTTI, a vertically integrated framework for simulation and validation of NoCs. Our goal is to provide the research community with a design environment for NoCs that supports fast performance analysis, early power estimation, and efficient NoC synthesis based on the execution of real software application scenarios. In particular, VENTTI: *i)* facilitates the decision-making process regarding the design of the interconnect for a SoC by offering early estimation of the NoC power dissipation; *ii)* simplifies the specification of the communication requirements for NoC synthesis; and *iii)* enables the validation of the NoC design through simulations with realistic applications. Fig. I shows the main components of the proposed framework: VENTTI includes a virtual platform that models the target SoC and enables the simulations of various user scenarios with the actual applications software running on top of a complex operating system such as Linux; VENTTI also integrates an open-source synthesis tool to automatically derive the main properties of an optimal NoC. Then, it uses a library of pre-designed and pre-characterized NoC building blocks to build an NoC implementation that is represented at three, increasingly detailed but consistent, levels of abstraction. The NoC implementation and the applications running on the SoC can be simulated and validated at these different levels of abstractions by trading off accuracy for simulation speed, all within the same virtual environment.

II. THE VENTTI INTEGRATED FRAMEWORK

VENTTI tackles the complexity in the design and analysis of NoC by adopting a layered approach where at each layer different design points can be efficiently analyzed and tested. As shown in Tab. I, each model provides a different accuracy degree in terms of delay and power estimation, thus enabling the analysis of the NoC properties by trading-off accuracy vs. simulation time.

978-1-4673-1295-0/12/$31.00 ©2012 IEEE

TABLE I
TYPE OF ANALYSIS PERFORMED AT EACH LEVEL OF ABSTRACTION.

	Type of Analysis		*Models*			*Relative*
	SW	*HW*	*Latency*	*Bandwidth*	*Power*	*Sim. Time*
TVM	debug	–	constant	infinite	–	$1\times$
HNM	debug	clock frequency, topology and routing algorithm selection	zero-load	ideal	no contentions	$\sim 2.5\times$
FNM	validation	queue sizing, num VCs selection, flit width	zero-load+ contentions	saturation	with contentions	$\sim 4.3\times$
PCM	–	synthesis	–	–	pin-accurate	$\sim 350\times$

The *Transaction-Verification Model (TVM)* is the most abstract model of the SoC interconnect: the cores, memory and accelerators are modeled in a virtual platform and linked with "ideal" point-to-point channels offering a constant traversal delay per packet. This enables extremely fast simulations and is optimal for the testing and debugging of the embedded software. It offers, however, limited details regarding the performance of the interconnects.

The *Hop-count-based Network Model (HNM)* is a fast network model that maximizes the simulation speed by abstracting away most of the micro-architectural details of a NoC. In particular, HNM uses a look-up table storing just topological information (e.g. path length between any source and destination) and channel flit-width of the NoC to quickly estimate the delay taken by a packet to reach its destination. This layer allows the testing of the applications running on the virtual platform with a more realistic communication delay, a first estimation of the power dissipation (see Section III), as well as a rapid evaluation of basic NoC design choices such as topology, flit width and achievable clock frequency.

The *Flit-based Network Model (FNM)* adds more details to the router design, including the finite size of the buffering queues and the actual routing algorithm. Data transfers are simulated with multiple flits injected into the NoC where they can experience contentions that generate delays. Hence, the correctness and performance of the hardware and software can be validated with an SoC model that is much closer to the final implementation. With FNM, the power and communication latency estimation can be very accurate and allows designers to perform a detailed analysis of the SoC.

Finally, the *Pin-and-Cycle-accurate Model (PCM)* is a register-transfer level (RTL) model that can be used as the input for a logic synthesis tool to derive the final gate-level implementation of that NoC which has emerged as the best choice from the previous steps of the design process. At this level designers can also run further simulations to derive accurate power-estimations. Specifically, we support SYSTEMC-RTL co-simulations by recording the communication activity traces from the FNM model. These traces can be used to drive the RTL simulation. They accurately reproduce the communication activity generated by the SoC modules while avoiding the additional complexity of running the software applications and the operating system.

In VENTTI we guarantee the consistency among the four different levels of abstractions by using a single tool to generate the network models. As shown in Fig. I, given a virtual platform implementing an SoC and the applications running on top of it, VENTTI synthesizes an optimal NoC by either extracting or taking a *Communication-Task Graph (CTG)* as input parameter[1]. In case the CTG is not provided, VENTTI derives it automatically by analyzing the communication traces obtained with the TVM simulation of the applications and the SoC. VENTTI uses the maximum point-to-point communication bandwidth to annotate each CTG edge. We leave as future work the further refinement of such technique (e.g. the automatic differentiation of the message classes).

VENTTI leverages two open-source tools: COSI, a NoC synthesis tool [5], [6] and RABBITS, an SoC virtual platform [7]. We chose COSI as it is a public-domain tool that is based on the principles of *platform-based design* [8] and is capable of synthesizing an optimal NoC given a CTG and a library of components (such as routers and network interfaces) and a desired synthesis algorithm from a library of available ones [5], [9]. As discussed in Section III, in VENTTI we extended the library of components to generate three of the proposed models, namely: HNM, FNM and PCM, and implemented a set of back-end code generators to produce a representation of the synthesized network.

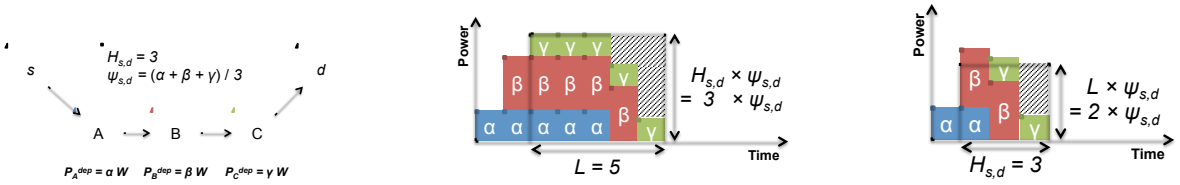
RABBITS is an open-source virtual platform for SoC simulations based on two main libraries: QEMU [10] and SYSTEMC [7]. QEMU is a fast instruction-set simulator capable of executing code compiled for a guest architecture (e.g. ARM) on top of a host machine (e.g. Intel x86). In RABBITS we can have multiple instances of QEMU within a *time-approximated* SYSTEMC model [11] of a multi-core SoC. This allows testing of embedded applications into a virtual environment that closely resembles the real chip being developed. We chose RABBITS because it uses a fast time-approximate model where time is represented by estimating the delays taken for each computation to happen following a given event. In particular, to estimate the duration of a sequence of instructions in a emulated processor, RABBITS adds time-approximation constants also to the micro-operations used by QEMU. In VENTTI we modified the original source-code by implementing a common interface between COSI and RABBITS and added the interconnect component described in the following section.

III. VENTTI NETWORK ABSTRACTION LAYERS

We present here a more detailed description of the modeling properties of each of the four abstraction layers of VENTTI.

Transaction Verification Model. This executable model is built on top of the event-driven SYSTEMC simulation engine and leverages the RABBITS virtual platform. The TVM allows very fast simulation performance because it abstracts away all the details of the actual transmission of a packet across an interconnection network between two SoC modules. If

[1]A CTG is a direct graph $G(V, E)$ where each vertex $v \in V$ represents an IP element while each direct edge $(v_i, v_j) \in E$ represents a point-to-point communication link and is annotated with performance requirements such as the minimum required bandwidth [5].



(a) Example (b) When $H_{s,d} < L$ (c) When $H_{s,d} > L$

Fig. 2. The energy estimation method implemented in the HNM for a path of three (heterogeneous) routers: A,B and C.

expressed, these details would add a large number of SystemC simulation events, which would slow down the overall simulation. With TVM, we can simulate the booting of the Linux OS on the target SoC in a matter of seconds. We can then execute application software on the virtualized processing cores and simulate SystemC models of hardware accelerators. Further, we can analyze the communication traces among the various SoC modules and build a CTG expressing the point-to-point communication requirements among them in terms of either average or worst-case bandwidth depending on the targeted embedded-system characteristics (e.g. soft or hard real-time). The CTG can be given as an input to COSI for the synthesis of three different views of the NoC implementation, each corresponding to one of the three other models.

Power Model: Due to the abstract nature of the interconnect model, no power analysis is performed in TVM.

Hop-count-based Network Model. The HNM captures important system-level properties of an NoC while maximizing the simulation speed by abstracting away most of its micro-architectural details. During an HNM simulation, a look-up table with the topological information generated by COSI is used to quickly estimate the transmission latency $T_{s,d}$ of a packet between a source s and a destination d as:

$$T_{s,d} = H_{s,d} \cdot t_r + L \tag{1}$$

where $H_{s,d}$ is the number of hops stored in the look-up table, t_r is the router processing latency in clock cycles and L is the length of the packet expressed in flits [12]. Since the instantiation of the NoC routers is not considered in HNM, the actual values of latency and power consumption that may be caused by packet congestion cannot be estimated. Instead, for any given packet, the value of Eq. 1 represents a lower bound of the actual delay that the NoC can deliver. Note, however, that results obtained with HNM remain strictly valid from a functional viewpoint because the NoC generated by COSI is guaranteed to sustain the bandwidth requirements for all end-to-end communications between pairs of SoC modules [5].

Power Model: A look-up table is also used to estimate the power consumption of the NoC. In particular, we define two parameters to model the power dissipation of a router: *i)* the *traffic-independent power* P^{ind} is dissipated constantly regardless of the input activity (e.g. static power and clock); *ii)* the *traffic-dependent power* P^{dep} is the extra power consumed during a flit traversal. Since P^{ind} does not depend on the traffic activity generated by the application, we account for it only at the end of the simulation where it is multiplied by the SoC execution time. To populate the HNM look-up table, users of VENTTI can choose between a power-estimation tool like ORION [13] or back-annotated power values derived from power analysis at the PCM layer. In particular, we developed a COSI back-end module that derives the table by first identifying the routing path $\Pi_{s,d}$ between each source-destination pair and then associating to the path the "*average per-hop traffic-dependent power*":

$$\Psi_{s,d} = \frac{\sum_{\forall r \in \Pi_{s,d}} P_r^{dep}}{H_{s,d}} \tag{2}$$

where r is a router in the path. The energy consumed to transfer a packet of size L along $\Pi_{s,d}$ is estimated as:

$$E_{s,d} = \Psi_{s,d} \cdot L \cdot H_{s,d} \cdot t_r \cdot T_{clk} \tag{3}$$

where T_{clk} is the clock period of the synchronous routers.

The example of Fig. 2(a) illustrates how in HNM we estimate the energy spent to transfer a packet of L flits across three routers A, B, and C from source s to destination d. The traffic-dependent power of the three routers is set as $P_A^{dep} = \alpha W$, $P_B^{dep} = \beta W$, and $P_C^{dep} = \gamma W$, and $H_{s,d} = 3$. With these values, we can derive the average per-hop traffic dependent power $\Psi_{s,d} = (\alpha+\beta+\gamma)/3$. Notice that Eq. 3 holds regardless of the actual packet length L. Fig. 2(b) and (c) show the case of packet with length $L = 5$ and $L = 3$, respectively[2]. The total energy consumption $E_{s,d}$ is represented by the area of the box. Notice how the shaded portion in the upper-right corner corresponds exactly to the sum of the energy consumed by the flits remaining outside the box.

Flit-based Network Model. This executable model is based on the SYSTEMC TLM1.0 library. Specifically, each message transfer in the NoC (e.g. a load/store operation towards the DRAM) leads to the injection of a sequence of flits into the NoC. Also, the SYSTEMC `SC_THREAD` process shown as Algorithm 1 is instantiated for each input port of a NoC router. The FNM obtains a good balance of accuracy and speed because the NoC is modeled with an approximated-time TLM such that: *i)* no component in the NoC uses a synchronous clock; *ii)* each router remains idle (does not add any event to the SYSTEMC kernel) until it receives a flit; and *iii)* when a flit is received a processing delay is modeled before the forwarding of the flit. In case of contention, a router can generate *back-pressure* signals and delay the delivery of the flits until the channel becomes available. FNM leverages the blocking properties of the `tlm_fifo` queues of the TLM library and a `sc_mutex` SYSTEMC class that regulates the access to a shared resource (e.g. an output port requested by multiple input ports). In summary, FNM accurately reproduces

[2]Notice that the different height of the α, β and γ boxes reflect the possible heterogeneous router design (e.g. different number of input/output ports) and hence a different power dissipation.

Algorithm 1 Modeling process of router input port in FNM.

```
loop
    F ← the flit at the head of input queue Q
    Pop Q
    O ← the output port of the destination in F
    Lock the mutex of O
    Wait for a clock period
    Send F to O
    while F is not a tail flit do
        F ← the flit at the head of Q
        Pop Q
        Wait for a clock period
        Send F to O
    end while
    Release the mutex of O
end loop
```

a cycle-accurate behavior of the NoC while minimizing the number of events in the SystemC simulation engine.

Power Model: the model of a router r contains traffic-independent power P_r^{ind} and traffic-dependent power P_r^{dep} values as defined in Section III; it also uses a energy-consumption counter E_r^{dep} that is updated upon the arrival and departure of a flit according to the following equation:

$$E_r^{dep} = E_r^{dep} + t \cdot f \cdot P_r^{dep} \quad (4)$$

where t is the time elapsed since the last update of E_r^{dep} and f is the number of flits that traversed r during t.

Pin-and-Cycle-accurate Model. The PCM is a complete synthesizable RTL model of the NoC, including routers, repeaters, and network interfaces. With PCM, the application performance can be obtained by plugging PCM directly into Rabbits to perform RTL-SystemC co-simulation. This, however, should be used with care because it may result in a very long simulation time.

Power Model: At this level we leverage standard RTL power-estimation tools like Synopsys Primetime PX combined with the traces generated with FNM simulation that capture the switching activity of each router.

IV. Case Study

In this section we demonstrate the capabilities of the Ventti environment by comparing the different network modeling layers for a complex application scenario.

Experimental Setup. The SoC case study, shown in Fig. 3, is similar to the one presented by Gligor *et al.* [14]: it consists of four symmetric-multi-processor ARM11MP with private L1 caches, an on-chip SRAM, a frame buffer handling the display, and an accelerator subsystems called *DBF*. The ARM processors support dynamic-frequency scaling that is automatically adjusted to the application load. The valid frequencies reach a maximum clock of 300MHz. Each core has L1 instruction and data caches of 8 KB. The SoC contains also an 8 MB SRAM and a memory controller that is connected to an 256MB off-chip DRAM.

The SoC runs a full 2.6.x Linux OS and a multi-threaded H264 soft real-time decoder application whose execution is distributed among all four cores. Additionally, the application exploits the DBF accelerator to execute a specific computationally-intensive decoding task.

All SoC modules are directly connected to the NoC that is synthesized by using Cosi with the following input setting:

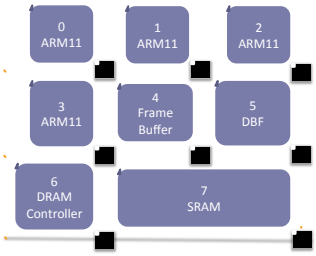


Fig. 3. The H264 SoC considered in the test-case.

a 500MHz clock frequency and 64 bits flit width. In this experiment, we use two alternative algorithms that are available in Cosi to synthesize the NoC: *H2* and *Mesh* [5]. *H2* in a heuristic that generates an *ad-hoc* NoC starting from an initial solution and iterating until it reaches a fixed point: specifically, for each point-to-point communication requirement the algorithm checks if there are bandwidth violations and solves them by adding new routers or re-routing the source-destination paths that cause the violations. *Mesh* is an algorithm that maps an application onto a given 2D mesh topology [9]: it iteratively improves an initial mapping by placing cores that communicate rapidly into mesh nodes that are topologically close to one another; at each iteration, two cores are swapped in the mapping and new paths are selected in the mesh such that the number of hops between sources and destinations is minimized.

While Cosi generates networks that are deadlock-free by construction, it is still necessary to address the message-dependent deadlock that can occur in the network interfaces [15]. In Ventti we address this issue by providing sufficiently large queues in the interfaces in order to guarantee the *consumption assumption* of all packets injected into the NoC. A similar solution is adopted by the Tilera Tile64 and IBM Cell processors [16][3].

For the HNM and FNM models, we use a power-library generated by synthesizing our PCM routers with Synopsys Design Compiler using a $45nm$ standard-cell library. In particular, the traffic-independent power P^{ind} is estimated with Synopsys Primetime PX by setting the static probability and toggle rate to 0 for all input ports. For the traffic-dependent power P^{dep}, we first estimate the total power dissipation P^T (dependent plus independent) of a router by setting the static probability to 0.5 and a toggle rate of 1 and then we subtract P^{ind} from P^T. Note that this approach is based on the conservative assumption that at every clock cycle all pins in the active input port always switch from 0 to 1 or 1 to 0.

For the PCM power estimation, we synthesize the VHDL network using a $45nm$ standard-cell library and Synopsys Primetime PX.

Experimental Results. For each of the proposed Ventti abstraction layers, Fig. 4 reports the *host time*, i.e. the wall-clock time necessary to simulate the H264 application with the two different NoC implementations: *H2* and *Mesh*. For the case of PCM, due to the large amount of time and storage

[3]As future work, we plan to enrich the library of components in Cosi with new network interface designs that are capable of solving this kind of deadlock by using virtual channels or an end-to-end flow-control protocol.

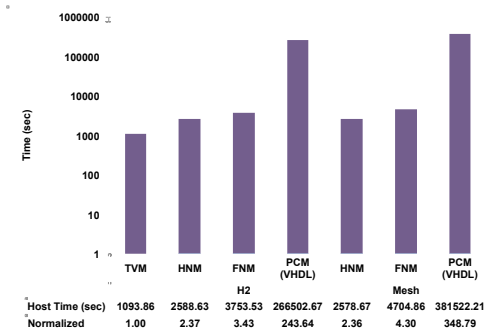


Fig. 4. Host time to simulate the H264 application with the *H2* and *Mesh* networks at the four different NoC abstraction levels.

required for the entire H264 simulation, we only consider a subset of the traces lasting for 0.5 seconds of the *guest time* (i.e. the time as seen by the emulated SoC). We then report the *host time* by linearly projecting this processing period for the entire duration of the H264 application. Note that the reported value only includes the PCM simulation time, without considering the time to generate the traces and to synthesize the VHDL code for the PCM. While the fastest simulation (TVM) takes 1093 seconds (∼18.2 minutes) to run the full H264 application, the SYSTEMC-RTL co-simulation with PCM takes 381522 seconds (∼106 hours). The values reported in the second row of the table at the bottom of Fig. 4 are normalized with respect to the TVM host time.

The normalized *host time* of the approximated network model for *Mesh* is much slower than for *H2*. The difference is due to the total number of routers in the NoCs. Since FNM and PCM simulate contentions in each router, more time is necessary to simulate the behavior of an NoC with more routers. In HNM, instead, the simulation is simply based on the calculation of hop count, which can be done in constant time regardless of the number of routers in the NoC.

To verify the accuracy of all network models, in Fig. 5 we report the *guest time*, *average latency*, and *average injection rate per node* while executing the H264 application. Since FNM models the cycle-accurate behavior of the NoCs, the values of this simulations can be considered as baseline values. The HNM and FNM simulations return similar performance results across all three metrics. For both *Mesh* and *H2*, they give similar average latencies, which means that both networks are not congested. Still as expected, the average latency for FNM is higher than HNM. HNM in fact does not capture the additional delays caused by packet contentions.

For the injection rate, TVM also shows similar injection rates as the two other network models. The maximum differences of injection rate comes from the comparison between HNM and FNM of *M*esh, which is about 16.38%. This difference is an expected behavior and is due to the variability introduced by having multiple threads and an operating system running on the emulated SoC [17].

Fig. 6 reports the power estimations with the four different NoC models for the two NoC configurations. We consider the power estimations of PCM as the baseline values. By comparing HNM and/or FNM to PCM, we find noticeable differences for both traffic-independent and -dependent power estimations. For traffic-independent power, the main difference comes from the optimization done by the synthesis tools: for HNM and FNM we simply aggregate the power values of all instantiated routers in the NoC, while for PCM the logic synthesis of the NoC uses the *flatten* and *uniquify* options. These options give the synthesis tools more scope for optimizations across the NoC submodules by flattening the hierarchy of the VHDL design. Thus, since we have more routers in a mesh, those traffic-independent power dissipation values differ significantly between PCM and the other two network models.

For traffic-dependent power estimations, *Mesh* dissipates more power than *H2* for all NoC abstraction layers. The power values increase as we go from HNM to PCM given the more detailed power analysis. Specifically, as we move from HNM to FNM, we account for the power dissipation caused by contentions. As we move from FNM to PCM, we additionally consider the power dissipation caused by the all flow-control signals and data transfer.

The difference between HNM and FNM is noticeable for *H2*, while it is minimal for *Mesh*. The reason is that the routers in *H2* experience more contentions than those in *Mesh*. Since more hardware resources are used for the *Mesh* implementation that the *H2* one, congestion is comparatively less likely to occur.

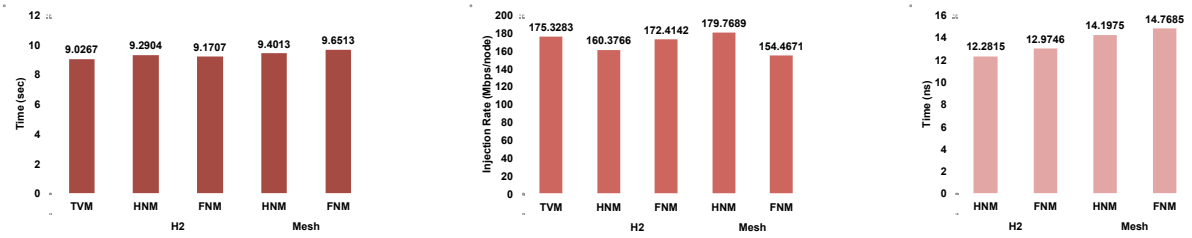
Not only does VENTTI provide performance and power estimation with different levels of modeling abstraction, but it also gives NoC designers the opportunity to identify the relationship between performance and power estimation at early stages of the design process. For example, the average packet latency of *Mesh* is longer than that of *H*2, which means that packets stay in the *Mesh* longer than in the *H*2. The more a packet stays in a NoC, the more it contributes to the overall power dissipation. Such relationship cannot be captured by a static power analysis tool, while it requires a significant amount of *host time* to be identified with a cycle-accurate-based virtual platforms.

V. RELATED WORK

While the literature offers a large number SoC Instruction-Set-Simulators (ISS), virtual platforms, and NoC synthesis tools, to the best of our knowledge no public-domain tool combines and integrates them to obtain an optimized design of the interconnect that can be both directly tested with a virtual platform and synthesized with commercial logic-synthesis tools.

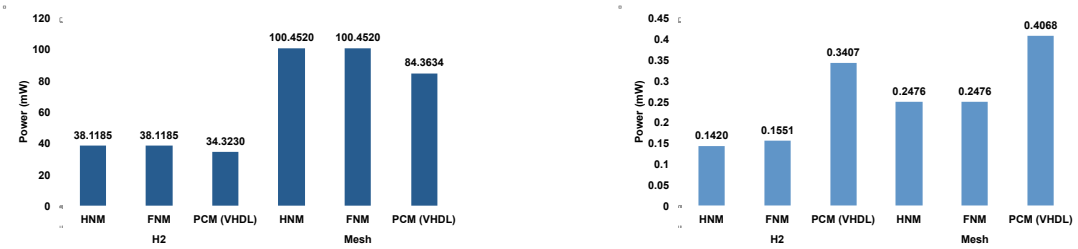
SOCLIB is a virtual platform with a very large library of components targeted to the development of both SoC-based embedded systems and the software running on top of them. Nevertheless, SOCLIB lacks a network-synthesis tool capable of synthesizing a power-efficient interconnect from a set of communication requirements obtained from the execution of complex application scenarios [18].

MPARM [19] is a multi-processor cycle-accurate architectural simulator. Its main purpose is the system-level analysis of the design trade-offs and component selection. MPARM is based on SYSTEMC and relies on SWARM, an open-source cycle-accurate ISS of the ARM processor and bus-based AMBA AHB protocol [20]. Instead, VENTTI focuses on the



(a) Guest Time (b) Injection Rate (c) Average Latency

Fig. 5. Performance of H264 application with *H2* and *Mesh* network and different NoC abstraction levels.



(a) Traffic-Independent Power (b) Traffic-Dependent Power

Fig. 6. Power estimation of H264 application with *H2* and *Mesh* network and different NoC abstraction levels.

design-space exploration of the NoC while relying on a time-approximate virtual-platform that allows us to quickly and efficiently simulate the entire system at a higher abstraction layer than cycle-accurate models.

The GEM5 simulation infrastructure provides diverse CPU models with multiple ISAs and a detailed and flexible memory system with multiple cache coherence protocols and interconnect models. GEM5 provides two interconnect models, called *Simple* and GARNET, which are similar to HNM and FNM, respectively. While GARNET uses ORION [13] to estimate the NoC power dissipation, there is no power model for *Simple*, and GEM5 does not provide a synthesizable PCM-equivalent network model.

VI. CONCLUSIONS

We presented VENTTI, an integrated environment for the efficient simulation and validation of NoCs. VENTTI leverages three network models that enable the analysis of the NoC at different levels of abstraction all within the same virtual platform. With fast performance and power analyses, VENTTI provides the information necessary to guide the design-space exploration towards an optimal NoC implementation and to evaluate its properties. By means of a case study, we showed that the NoC abstraction layers provided in VENTTI offer interesting trade-off points between simulation time and simulation accuracy. Further, VENTTI can be used to compare the performance and power dissipation across alternative NoC implementations and to understand the relationships between performance and power at early stages of the design process.

Acknowledgements. This research is partially supported by the National Science Foundation under Award #: 1147406, an ONR Young Investigator Award, and the Gigascale Systems Research Center, one of six research centers funded under the Focus Center Research Program (FCRP), a Semiconductor Research Corporation entity.

REFERENCES

[1] C. H. van Berkel, "Multi-core for mobile phones," in *Conf. on Design, Automation and Test in Europe*, Apr. 2009, pp. 20–24.
[2] J. Brown, S. Woodward, B. Bass, and C. Johnson, "IBM power edge of network processor: A wire-speed system on a chip," *IEEE Micro*, vol. 31, no. 2, pp. 76 –85, Mar. 2011.
[3] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *Proc. of the Design Autom. Conf.*, Jun. 2001, pp. 684–689.
[4] L. Benini and G. D. Micheli, "Networks on chip: A new SoC paradigm," *IEEE Computer*, vol. 49, no. 2/3, pp. 70–71, Jan. 2002.
[5] A. Pinto, L. Carloni, and A. Sangiovanni-Vincentelli, "COSI: A framework for the design of interconnection networks," *IEEE Design & Test of Computers*, vol. 25, no. 5, pp. 402–415, sep-oct 2008.
[6] "COSI," http://code.google.com/p/commsynth.
[7] "TIMA research lab," http://tima-sls.imag.fr/www/research/rabbits.
[8] A. Sangiovanni-Vincentelli, "Quo vadis SLD: Reasoning about trends and challenges of system-level design," *Proceedings of the IEEE*, vol. 95, no. 3, pp. 467–506, March 2007.
[9] S. Murali and G. De Micheli, "Sunmap: a tool for automatic topology selection and generation for nocs," in *Proc. of the Design Automation Conference*, ser. DAC '04, 2004, pp. 914–919.
[10] F. Bellard, "QEMU, a fast and portable dynamic translator," in *Proc. of the USENIX Annual Technical Conference*, Apr. 2005, pp. 41–46.
[11] "SystemC Website," http://www.systemc.org/.
[12] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, 2004.
[13] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik, "Orion: A power-performance simulator for interconnection networks," in *IEEE/ACM Intl. Symp. on Microarchitecture (MICRO-35)*, Nov. 2002.
[14] M. Gligor *et al.*, "Practical design space exploration of an h264 decoder for handheld devices using a virtual platform," in *Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation*, 2010, vol. 5953, pp. 206–215.
[15] Y. H. Song and T. M. Pinkston, "A progressive approach to handling message-dependent deadlock in parallel computer systems," *IEEE Trans. on Par. and Dist. Systems*, vol. 14, no. 3, pp. 259–275, Mar. 2003.
[16] N. E. Jergerand and L.-S. Peh, *On-Chip Networks:Synthesis Lectures on Computer Architecture*. Mark Hill, 2009.
[17] A. R. Alameldeen and D. A. Wood, "Addressing workload variability in architectural simulations," *IEEE Micro*, vol. 23, no. 6, pp. 94–98, nov.–dec. 2003.
[18] "SocLiB," http://www.soclib.fr/.
[19] L. Benini, D. Bertozzi, A. Bogliolo, F. Menichelli, and M. Olivieri, "MPARM: Exploring the multi-processor SoC design space with SystemC," *J. VLSI Signal Process. Syst.*, vol. 41, pp. 169–182, Sep. 2005.
[20] M. Dales *et al.*, "SWARM - Software ARM," http://www.swarm.org, 2008, [Online; accessed 7-December-2011].