# Performance Evaluation of Network-on-Chip-Based H.264 Video Decoders via Full System Simulation

Milad Ghorbani Moghaddam and Cristinel Ababei, *Senior Member, IEEE*

***Abstract*—We present a full system simulation framework for a network-on-chip (NoC)-based H.264 video decoder. By combining both the communication, i.e., the NoC and the processing, i.e., H.264 modules, components into the same simulation framework, we present for the first time the capability of simulating NoCs exercised with truly real traffic. Such a simulator can be utilized to evaluate performance metrics of systems-on-chip where the communication is done via NoCs, which are predicted to be the future communication paradigm of such systems. Because the NoC is exercised with real traffic instead of synthetic traffic such evaluations and optimizations can be more accurate and can lead to better design solutions.**

***Index Terms*—Full system simulation, H.264 video decoder, network-on-chip (NoC).**

## I. Introduction

NETWORKS-ON-CHIP (NoCs) represent a new communication paradigm for integrated circuits with increasingly large number of cores, such as chip multiprocessors (CMPs) and multiprocessor systems-on-chip (MPSoC). The NoC concept replaces design-specific global on-chip wires with a generic on-chip interconnection network implemented by specialized routers that connect processing elements (PEs)—such as processors, application-specified integrated circuits, field-programmable gate arrays, memories, etc.—to the network and facilitate communications or links between them. The benefits of the NoC-based SoC design approach include scalability, predictability, and higher bandwidth to support concurrent communications [1].

The most popular approach in studying NoCs is to use NoC simulators. In fact, in many cases, especially when the number of PEs is very large and hardware implementations are not yet available, simulators are the only way to investigate the performance characteristics of design ideas related to NoCs. As such, several NoC simulators have been implemented and released to the public domain including BookSim, Noxim, Garnet, and VNOC [2]–[5]. However, one of the main limitations of such simulation tools is that they usually simulate synthetic traffic including uniform random, transpose, and hotspot. In such cases, the NoC is not exercised with realistic workload/traffic and thus optimization techniques may be misled to suboptimal solutions. The closest approaches to simulating realistic workloads include self-similar traffic generators and application specific trace files. However, self-similar traffic is still synthetic while trace files are cumbersome to work with, do not include actual payload data inside the trace files but only injection times and number of injected packets, and cannot be used in set-ups where dynamic voltage and frequency scaling (DVFS) is utilized because packet injection times change under DVFS conditions. Yet another approach to simulate realistic NoC workloads is to use full-system simulators, such as Gem5, which has integrated the Garnet NoC model [6]. The limitation of this approach is that one is limited to only simulating CMPs, where all the PEs are usually Alpha or X86 processor architectures. It cannot be used to simulate specific MPSoCs, such as multimedia applications where the PEs are not processors but rather heterogeneous modules with specific functionality.

Therefore, in this letter we present a full system simulation framework for NoC-based MPSoCs and demonstrate it for the specific case of an H.264 video decoder. The full system simulator is capable of simulating both the basic modules of the video decoder and the NoC as the communication infrastructure. The simulator is capable of estimating NoC performance as average network latency and the power consumption on real world video streams. This approach is the closest to an actual real implementation that one can get via emulation or simulation-based approaches.

Manuscript received September 2, 2016; revised January 24, 2017 and March 8, 2017; accepted March 27, 2017. Date of publication March 29, 2017; date of current version May 25, 2017. This manuscript was recommended for publication by P. R. Panda. *(Corresponding author: Cristinel Ababei.)*

The authors are with the Department of Electrical and Computer Engineering, Marquette University, Milwaukee, WI 53233 USA (e-mail: milad.ghorbanimoghaddam@marquette.edu; cristinel.ababei@marquette.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LES.2017.2689036

## II. Simulation of NoC-Based H.264 Video Decoders

### A. Background on the H.264 Video Decoding Standard

Video coding is an integral part of many visual information-based applications including digital TV, HDTV, mobile TV, and online video streaming. The first part of video *coding* compresses the digital video to reduce its file size to be stored on disk or streamed online. The second part is the *decoding* phase, which decompresses the encoded data in a frame by frame manner. H.264 is an advanced video compression/decompression standard that offers a better compression efficiency and less quality loss compared to previous methods, such as MPEG2 and MPEG4 [7].

The H.264 video decoder takes as input an encoded video file and produces a video stream that can be directly displayed. A video stream is a sequence of frames, where each frame is described as one or several slices and each slice is decomposed into several 16×16 pixels microblocks. Each microblock consists of 16×16 luma and 8×8 chroma samples. The H.264 video decoder generates these microblocks one by one using the information received from the entropy decoding, inverse

quantization, and transformation procedures applied on the encoded video stream. In order to generate an entire frame, the decoder uses two different prediction techniques to decode each microblock. Based on the slice type that the microblock belongs to, one of the *intra* or *inter prediction* methods is called. The *intra prediction* method focuses on the current frame and predicts the current microblock considering the previously generated microblocks in the same frame. The *inter prediction* method is also known as motion compensation. In this method, a microblock similar to the current microblock has been present in previous frames of the video stream but it could have been at a different location, $(x, y)$, within the frame. Providing the reference frame number and the displacement vectors in $x$ and $y$ directions, the inter prediction method can predict the current microblock. Furthermore, the residual data that contains the differences between the real frame and the predicted frame is received, entropy decoded, inverse transformed and quantized, and finally added to each predicted microblock to bring the quality of the video to fairly that of the original frame used before the encoding process.

### B. Proposed NoC-Based H.264 Video Decoder Framework

VNOC is our in-house and already publicly available NoC simulator [5]. It is an *event-based* simulator, meaning that whenever data needs to be injected into the network an event is created and added to the simulation queue. These events are then processed in a first-input first-output manner. When an event is processed, packets are created and injected into the network accordingly. Besides the data payload, packets also carry information about the *source* and *destination* addresses. The VNOC simulator was limited until now to only simulating synthetic traffic including random uniform, transpose, hotspot, self-similar, and trace files. In this letter, we use the VNOC simulator to develop the proposed VNOC+H.264 simulation framework. Specifically, the synthetic traffic *packet injectors* are replaced with the real H.264 video decoder related modules, which effectively utilize the NoC as the communication medium to talk to each other and to exchange information now organized as packets. The H.264 video decoder modules that replace the synthetic packet injectors are adapted from an existing implementation that used the traditional memory-based communication approach between different functions of the decoder [8].

One of the first tasks in implementing an NoC-based MPSoC is the partitioning of the application at hand into modules that represent the PEs, which will be attached to the NoC routers. Usually, such partitioning may be natural due to the internal structure of the application. This is the case of the H.264 video decoder, which we partition as shown in the block diagram from Fig. 1, which also shows the interactions between the modules. These interactions represent communications that will be facilitated by the NoC component inside the simulation tool.

In our implementation, we have split the functionality of the H.264 decoder into seven modules: 1) *get network adaptive layer* (NAL); 2) *decode header*; 3) *decode microblock*; 4) *inter prediction*; 5) *intra prediction*; 6) *frame buffer*; and 7) *display*. *Get NAL* module retrieves the NAL units from the encoded video stream and passes them to the *decode header* module. This module decodes the slice header received in the

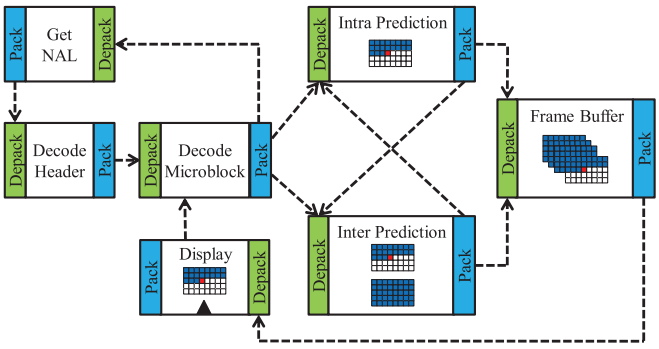


Fig. 1. Block diagram of the H.264 video decoder partitioned into seven modules, which communicate as shown by the dashed lines. All these communications between modules are implemented in the NoC.

NAL data and triggers the *decode microblock* module. The *decode microblock* module is responsible for entropy decoding and for the inverse transformation and quantization processes for each microblock. Based on the information produced during these processes, data would be sent to either the *intra prediction* module or the *inter prediction* module to be used in predicting the corresponding microblock. Because the *intra* and *inter prediction* modules need previously predicted data for their future predictions, the microblock generated by each of these two modules would be sent to one another as well as to the *frame buffer* module. The *frame buffer* module sends the microblocks to the *display* module, which is responsible with driving the display, which in our case is the screen. The *frame buffer* module can be substituted by any filter, such as *deblocking filter* to generate smoother edges in the frame, which is done in some H.264 implementations. Such a filter is not included here because it was missing in the original source code of the H.264 implementation too. However, we kept the *frame buffer* as a separate module in order to mimic the presence of such a filter and have data transfer behavior inside the NoC. Once all the above steps are done for a frame the entire process, beginning with the *get NAL* module, is started again, and repeated until the end of the video stream.

Because the data is transmitted through the NoC in the form of packets, each module from Fig. 1 is fitted with a network interface (NI), which is responsible with packetizing and depacketizing the information. When a module sends data, an event is created and added to the simulation queue. When the event is removed from the queue and processed, the NI generates a series of packets, which will carry the data to be sent. The number of packets in this series depends on the amount of data that must be sent and the number of flits per packet. Basically, the data from the module is split into packets of a specified number of flits and sent over the NoC to the destination. In addition to the actual payload data, packets include information about the sender and receiver addresses as well as the send time.

Another important task in implementing an NoC-based MPSoC is the placement or mapping of the application modules to the routers of the regular mesh NoC. This is a very important step because the performance of the NoC depends on such placement, which must be done such that modules that communicate more data are placed closer to each other. In this letter, we assume that this is a step that has been already done

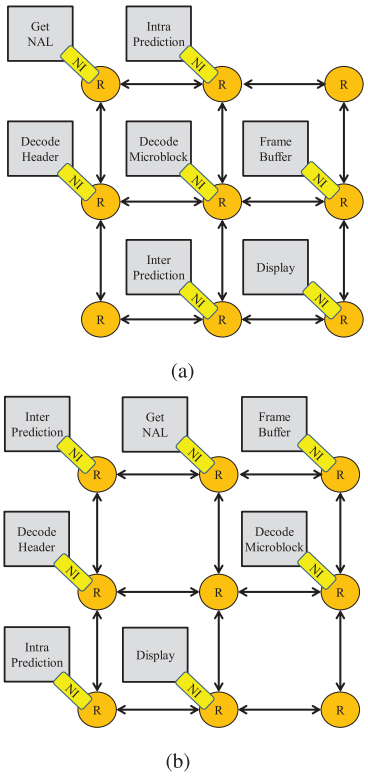


Fig. 2. Two different manual placements of the H.264 video decoder modules from Fig. 1 to a 3×3 regular mesh NoC. Each of the modules is mapped to its own router of the NoC to which is connected via the NI. (a) Placement 1. (b) Placement 2.

and module placement information is directly provided to the proposed simulation tool. For example, Fig. 2 shows two different placements (done manually) of the seven modules from Fig. 1. These placements *map* the seven modules to seven of the nine routers of the 3×3 regular mesh NoC.

### C. Simulation Framework Architecture

While the focus of this letter is on the H.264 video decoder, the proposed simulation framework is developed with flexibility in mind. The source code is object oriented and its architecture is modular, which makes the framework easily extendable to implement full system simulators for other applications. The block diagram from Fig. 3 illustrates the main objects that form the code architecture of our current implementation.

Each of the shown objects correspond to separate C++ classes, which can be utilized in a plug-and-play manner. The NoC related objects represent instances of classes from the VNOC simulator while the H.264 video decoder objects represent instances of classes adapted from a separate C++ implementation where communication of data used to be done using the memory-mapped communication programming model [8]. Here, we took that implementation and replaced the memory-mapped communication with the NoC-based communication, effectively combining the source codes of VNOC and the H.264 video decoder. Therefore, when the proposed framework is used to implement a full system simulation for a different application, all that is needed is to replace the objects that implement the application's functionality with the correct ones. The NoC and NI objects remain unchanged.

TABLE I
SUMMARY DESCRIPTION OF THE FOUR BENCHMARK VIDEO STREAMS

| Benchmark | Resolution (width x height) | Number of frames | Duration |
|---|---|---|---|
| B1 (Girl) | 704x576 | 174 | 6"96 |
| B2 (Freeway) | 704x576 | 232 | 9"28 |
| B3 (Plane) | 704x576 | 298 | 11"92 |
| B4 (Golf) | 704x576 | 311 | 12"44 |

TABLE II
SUMMARY OF THE SIMULATION RESULTS FOR THE NoC COMPONENT

| Placement + Benchmark | Power consumption (W) | Average latency (cycles) | Power Delay Product, PDP |
|---|---|---|---|
| Placement 1, B1 | 13.4628 | 0.1329 | 1.7892 |
| Placement 1, B2 | 13.5219 | 0.1335 | 1.8052 |
| Placement 1, B3 | 13.4579 | 0.1331 | 1.7912 |
| Placement 1, B4 | 13.6682 | 0.1334 | 1.8233 |
| Placement 2, B1 | 21.0177 | 0.1390 | 2.9215 |
| Placement 2, B2 | 21.5775 | 0.1405 | 3.0316 |
| Placement 2, B3 | 21.5129 | 0.1401 | 3.0139 |
| Placement 2, B4 | 21.5252 | 0.1399 | 3.0114 |

## III. SIMULATION RESULTS

In this section, we report simulation results obtained with the proposed simulation tool. We used four diverse H.264 encoded video streams as our benchmarks, which we downloaded from [9]. The characteristics of these videos are listed in Table I.

By feeding these video streams as input into our simulator, we effectively exercise the NoC with truly real traffic. The nature of this traffic is determined by the functionality of the application (in this case H.264 decoder) and the content of the video stream supplied as input. By combining both the NoC component with actual implementations of the H.264 component, we are able to emulate the operation of the entire NoC-based H.264 video decoder system as a whole as closely as possible to a real hardware implementation.

The simulation results for the NoC component using the four different video stream benchmarks for each of the two different placements from Fig. 2 are summarized in Table II. The power values include the power consumed by the NoC routers and links. The VNOC simulator [5], which we used to develop the proposed simulation framework, has integrated the Orion 2.0 power calculator [10] for 65 nm technology node. This power calculator is the most popular power estimation tool in the NoC community and has been validated with real data from the Intel's 80 core chip [11]. In addition, it has the advantage that the power estimation is done by directly considering the switching activities inside each router as resulted from the manipulation of real data carried as flits through the network. The latency numbers are directly reported by the VNOC simulator, which is an event-based simulation approach, where each router has a pipeline with four stages, with each stage taking place in one cycle.

By simulating both placements, we demonstrate the usefulness of the proposed simulation tool in evaluating different mappings of the application modules to routers of the NoC. These results show that the placement from Fig. 2(a) is better than that from Fig. 2(b). This is something that we expected because we created placement 2 by moving modules that share large communication volume far from each other—particularly, the *decode microblock* module. This in

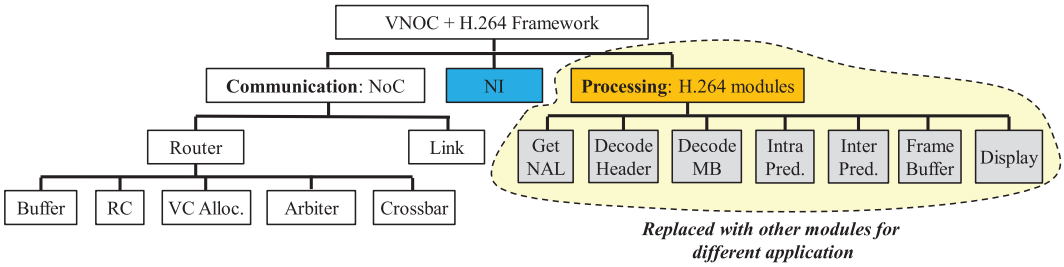


Fig. 3. Hierarchy of objects that make up the code architecture of the proposed simulation tool.

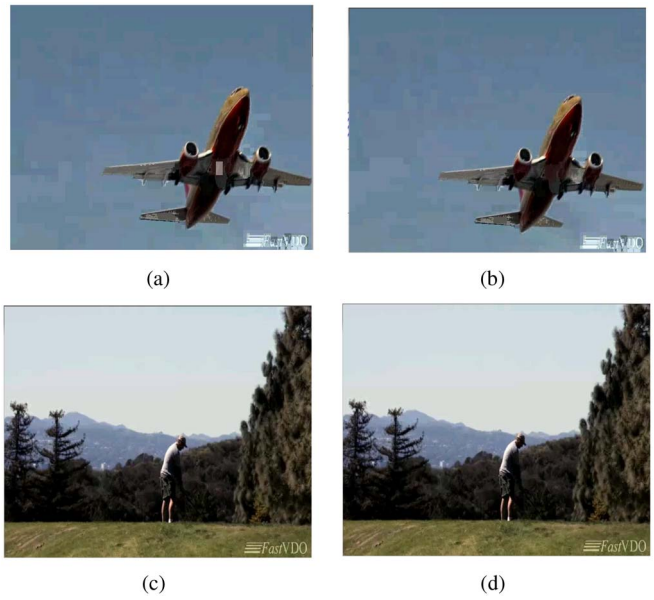


Fig. 4. Qualitative comparison of random frames processed with the original memory-mapped-based communication H.264 decoder and with the new NoC-based communication H.264 decoder. (a) Frame of benchmark B3 decoded by the reference memory-mapped implementation. (b) Same frame decoded by the NoC-based implementation. (c) Frame of benchmark B4 decoded by the reference memory-mapped implementation. (d) Same frame decoded by the NoC-based implementation.

turn increased the power consumption because flits need to travel through more routers and links. Also, average network latency increased because of the longer routing paths between those modules.

To validate correctness of the proposed NoC communication-based implementation of the H.264 video decoder we compared frame by frame its decoded video stream with that produced by the traditional implementation where the communication was done using the traditional memory-mapped programming model. Fig. 4 shows qualitatively a side by side comparison of randomly selected frames from the output video streams of benchmarks 3 and 4. The decoded video streams are virtually the same confirming the correctness of the NoC-based implementation.

## IV. Discussion

The proposed simulation tool has the following characteristics. It has the ability to accurately simulate the entire video decoder implemented using an NoC as the communication infrastructure. It can be used to evaluate different decoder application mappings to regular mesh NoCs. As such, it could be integrated into optimization frameworks/loops whose objective is application mapping for latency and power minimization. In addition, the proposed framework has integrated support for DVFS. Therefore, it could potentially be used to construct simulation frameworks aimed at power reductions via DVFS algorithms. We will investigate this in a different paper. The implementation is object-oriented and modular. It was developed with flexibility in mind to allow easy extension to implement full system simulation of other NoC-based systems. The entire source code and documentation is made publicly available [5].

## V. Conclusion

To address the lack of NoC simulation tools with real traffic workloads, we combined an NoC simulator with an H.264 video decoder implementation to create a combined simulation framework for an NoC-based H.264 video decoder. In this way, the NoC is simulated and its design can be optimized with truly real traffic determined by the actual video stream being decoded and by the mapping of the H.264 modules to the NoC routers.

## References

[1] R. Marculescu, U. Y. Ogras, L.-S. Peh, N. E. Jerger, and Y. Hoskote, "Outstanding research problems in NoC design: System, microarchitecture, and circuit perspectives," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 1, pp. 3–21, Jan. 2009.

[2] *BookSim Interconnection Network Simulator*, Stanford Univ., Stanford, CA, USA, 2016. [Online]. Available: http://nocs.stanford.edu/cgi-bin/trac.cgi/wiki/Resources/BookSim

[3] *Noxim—The NoC Simulator*, Univ. at Catania, Catania, Italy, 2016. [Online]. Available: https://github.com/davidepatti/noxim

[4] *GARNET: A Detailed on-Chip Network Model Inside a Full-System Simulator*, MIT, Cambridge, MA, USA, 2016. [Online]. Available: http://projects.csail.mit.edu/cgi-bin/wiki/view/LSPgroup/GarnetPage

[5] *Software Downloads at MESS Lab*, Marquette Univ., Milwaukee, WI, USA, 2016. [Online]. Available: http://dejazzer.com/software.html

[6] N. Binkert *et al.*, "The gem5 simulator," *ACM SIGARCH Comput. Archit. News Archive*, vol. 39, no. 2, pp. 1–7, 2011.

[7] "Advanced video coding for generic audiovisual services," ITU-T H.264, Geneva, Switzerland, 2012. [Online]. Available: http://www.itu.int/ITU-T/recommendations/rec.aspx?rec=11466

[8] M. Fiedler. (2016). *Implementation of a Basic H.264/AVC Decoder*. [Online]. Available: http://keyj.emphy.de/files/projects/h264-src.tar.gz

[9] (2016). *FastVDO: H.264 Video Streams*. [Online]. Available: http://www.fastvdo.com/H.264.html

[10] A. B. Kahng, B. Li, L.-S. Peh, and K. Samadi, "ORION 2.0: A power-area simulator for interconnection networks," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20. no. 1, pp. 191–196, Jan. 2012.

[11] S. R. Vangal *et al.*, "An 80-tile sub-100-W teraFLOPS processor in 65-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 43, no. 1, pp. 29–41, Jan. 2008.