Investigation of DVFS for Network-on-Chip Based H.264 Video Decoders with Truly Real Workload

Milad Ghorbani Moghaddam and Cristinel Ababei
Dept. of Electrical and Computer Engineering
Marquette University, Milwaukee WI, USA
Email: {milad.ghorbanimoghaddam,cristinel.ababei}@marquette.edu

***Abstract*—We investigate dynamic voltage and frequency scaling (DVFS) for a network-on-chip (NoC) based H.264 video decoder. The investigation is done using a simulation framework that combines both the communication, i.e. the NoC, and the processing, i.e., H.264 modules, components into the same simulation. This approach allows for the NoC to be exercised with truly real traffic instead of synthetic traffic because the H.264 modules process real data provided by the actual video streams supplied as input into the decoder. Therefore, the NoC design and optimization can be done by directly considering the workload under which the NoC will operate later on. Because truly real rather than synthetic traffic is utilized, evaluation of different NoC design choices is more accurate. Our investigation demonstrates that we are able to evaluate different NoC mapping solutions to asses the impact of a given DVFS strategy toward identifying optimal NoC design solutions for the specific case of an H.264 video decoder.**

***Index Terms*—network-on-chip; H.264 video decoder; DVFS; hybrid simulation;**

I. Introduction

Networks-on-Chip (NoCs) replace design-specific global on-chip wires with a generic on-chip interconnection network implemented by specialized routers that connect generic processing elements (PEs) − such as processors, ASICs, FPGAs, memories, etc. − to the network and facilitate communications or links between them. NoCs are predicted to become the primary communication paradigm for integrated circuits with increasingly large number of cores, such as chip multiprocessors (CMPs) and multiprocessor systems-on-chip (MPSoCs) [1], [2]. The benefits of the NoC based SoC design include scalability, predictability, and higher bandwidth with support for concurrent communications. Since the idea of routing packets instead of wires was proposed in the early 2000's, the NoC concept has grown into a rich research topic, with numerous papers published on this topic [3].

The most popular approach in studying Networks-on-Chip (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 pubic domain including BookSim, Noxim, Garnet, and VNOC [4]–[7]. Moreover, simulators are used to develop design and optimization methods for NoCs and to evaluate various design decisions. For example, NoC simulators can used to evaluate specific dynamic voltage and frequency scaling (DVFS) algorithms for NoCs [10].

978-1-5090-5117-5/16/$31.00

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 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 [8]. The limitation of this approach is that one is limited to only simulating chip multiprocessors, where all the PE are usually Alpha or X86 processor architectures. It cannot be used to simulate specific MPSoCs such as multimedia applications where the processing elements are not processors but rather heterogeneous modules with specific functionality.

To address this limitation of current NoC simulators in the context of an NoC based H.264 video decoder, in our previous work [9] we presented a new simulation framework that combines both the NoC with the H.264 video decoder modules. By simulating concurrently both the NoC and the functionality of the H.264 application, we effectively operate the NoC under truly real traffic. In this paper, we utilize this simulation framework to investigate the DVFS algorithm proposed in [10] applied to the NoC based H.264 video decoder.

II. Combined NoC and H.264 Video Decoder Simulation Framework

In this section, we present a short description of the VNOC+H.264 video decoder full system simulation framework which we reported initially in our previous work and which we employ here as a platform to investigate the DVFS algorithm. Further details on the architecture of this simulation framework can be found in [9].

The VNOC+H.264 video decoder full system simulation framework combines essentially two different simulators into a single simulation tool. It effectively combines the functionality of the H.264 modules with that of the NoC (see Fig.1), which is now responsible with carrying information concurrently between modules. To simulate the NoC, the tool uses the publicly available VNOC simulator [7]. The H.264 video decoder modules are adapted from an existing implementation that used the traditional memory-based communication approach between different functions of the decoder [11].

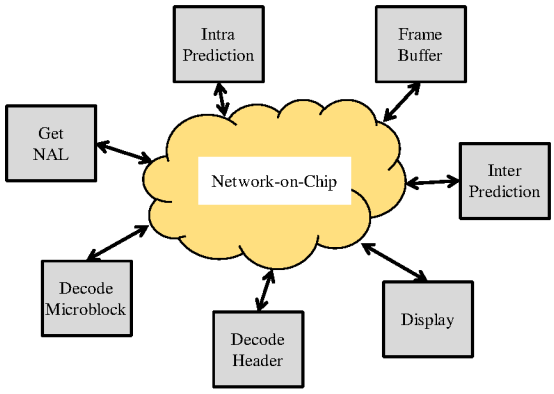


Fig. 1. The simulation framework combines H.264 video decoder modules with an NoC, which facilitates communications among modules.

The NoC simulator 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 FIFO manner. When an even 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. Before, the VNOC simulator was limited to only simulating synthetic traffic including random uniform, transpose, hotspot, self-similar, and trace files. By combining the VNOC simulator with the H.264 video decoder modules within the same simulation tool, the NoC is exercised with truly real traffic. Specifically, the initial 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.

Because the data is transmitted through the NoC in the form of packets, each module of the video decoder 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.

Thus, the VNOC+H.264 video decoder full system simulation framework has the ability to accurately simulate the entire video decoder implemented using an NoC as the communication infrastructure. The source code is object oriented and its architecture is modular, which makes the framework easily extendable to implement full system simulators for other MPSoC applications. The block diagram from Fig.2 illustrates the main objects that form the code architecture of the VNOC+H.264 video decoder implementation.

The blocks shown Fig.2 correspond to separate C++ classes, which can be utilized in a plug-an-play manner inside the implementation of the simulator. 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 [11]. As such, when this framework is used to implement a full system simulation for a different MPSoC 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.

III. Dynamic Voltage and Frequency Scaling Algorithm for NoCs

In this section, we describe briefly the DVFS algorithm from [10], which we apply here to the NoC simulated by the simulation framework described in the previous section. That is, the NoC that is used as a communication mechanism for the H.264 video decoder is investigated in terms of performance and power consumption with DVFS performed dynamically. Note that this is the only simulation approach that can evaluate design decisions for the NoC targeted as communication enabler for the video decoder. Without the combined simulation infrastructure from the previous section, we would be limited to simulating the NoC with synthetic traffic, which would not accurately reflect the application at hand, which in this case is the H.264 video decoder.

Being developed on top of the VNOC simulator, the simulation framework described in the previous section has integrated support for dynamic voltage and frequency scaling (DVFS). Therefore, we can use it to construct simulation frameworks aimed at power reductions via DVFS algorithms as well. That is precisely what we do in this paper. We investigate an DVFS algorithm for different placements of the H.264 video decoder modules onto a regular mesh NoCs. A placement is obtained by solving the problem of NoC application mapping, which is illustrated in Fig.3, where the application consists of seven modules. In this paper, we assume that mapping has been already done and module placement information is directly provided to the proposed VNOC+H.264 video decoder simulation framework, which we use to investigate the DVFS algorithm briefly described next.

The pseudocode of the distributed DVFS algorithm is shown in Fig. 4. It is distributed in the sense that is implemented

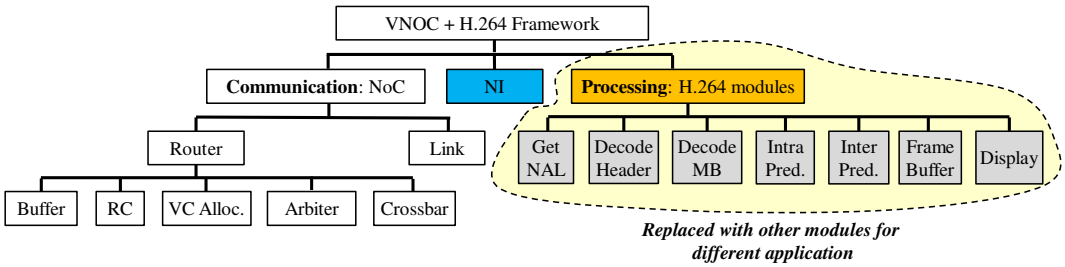


Fig. 2. Hierarchy of objects that make up the code architecture of the VNOC+H.264 video decoder simulation framework.

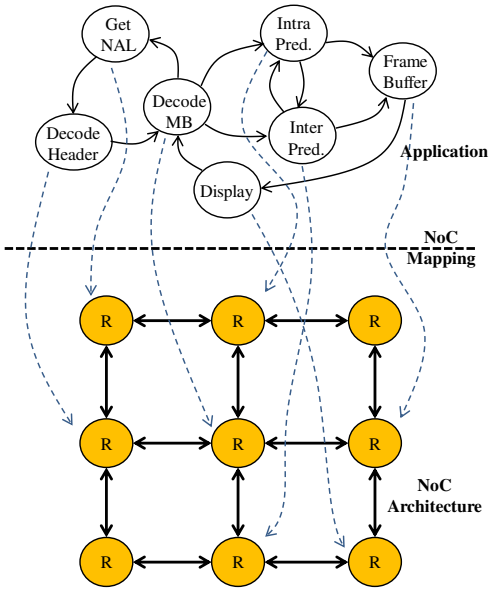


Fig. 3. Illustration of the mapping of the application modules to the routers of the NoC architecture.

Algorithm: Distributed DVFS for Congestion and Power Reduction

Start with each router set at f_{base} and VDD_{base}
At end of each control period, calculate predicted BU and LU
for all input buffers of each router and the links that drive them
for $i \leftarrow 1$ to n **do** // n: number of routers
  $counter_{switch-down} = 0$, $counter_{switch-up} = 0$
  for $j \leftarrow 1$ to 4 **do** // 4: number of output ports
    $BU^j_{pred} = (W * BU^j_{curr} + BU^j_{last})/(W+1)$
    $BU^j_{last} = BU^j_{pred}$
    $LU^j_{pred} = (W * LU^j_{curr} + LU^j_{last})/(W+1)$
    $LU^j_{last} = LU^j_{pred}$
    if $BU^j_{pred} < BU_{congested}$ **then** // $BU_{congested} = 0.5$
      $T_{low} = TL_{low}$, $T_{high} = TL_{high}$ // 0.3, 0.4
    else
      $T_{low} = TH_{low}$, $T_{high} = TH_{high}$ // 0.6, 0.7
    end if
    if $LU^j_{pred} < T_{low}$ **then**
      Frequency of this link to be switched down
      $counter_{switch-down} = counter_{switch-down} + 1$
    else if $LU^j_{pred} > T_{high}$ **then**
      Frequency of this link to be switched up
      $counter_{switch-up} = counter_{switch-up} + 1$
    end if
  end for
  if $counter_{switch-up} > 0$ **then**
    Increase frequency of this router
  else if $counter_{switch-down} > 0$ **then**
    Decrease/throttle frequency of this router
  else
    Keep the same frequency for this router
  end if
end for

Fig. 4. Pseudocode of the distributed DVFS algorithm that we investigate in this paper using the VNOC+H.264 video decoder full system simulation framework discussed in section II.

inside each router, which operates as a voltage frequency island (VFI) and is primarily composed of two steps that are executed at the end of each control period. First, it uses history based predictors to predict link and buffer utilizations denoted as LU and BU, respectively. These predictions are used to forecast the future network load. History based prediction works with a predefined history window (as a number of control periods), during which the variable of interest, x, is sampled and then averaged at the end of the window. To predict the average value of the variable of interest, x_{pred}, for the next history window, the following equation is used:

$$x_{pred} = \frac{W \times x_{curr} + x_{past}}{W+1} \quad (1)$$

where, x_{curr} is the computed average value of the variable of interest in the current history window, x_{past} is the previous prediction made during the past history window, and W is a user set parameter.

In the second step, the predictions are used to decide whether to throttle or boost the router's frequency in response to the forecast congestion in the neighboring routers. In other words, frequency is tuned proactively, thereby addressing potential congestion issues and reducing power consumption. This DVFS algorithm was studied in [10] for NoCs exercised with synthetic traffic only. It was reported that when frequency throttle was used only, power consumption could be reduced by up to 50% while the network latency was only slightly degraded. When frequency boost was also used, in addition to significant power reductions, network latency was also improved for high packet injection rates only. In the next section, we investigate this DVFS algorithm for an NoC where the traffic is truly real because it is generated during the combined simulation of both the NoC and the H.264 video decoder modules.

IV. Simulation Results

In this section, we report simulation results obtained with the proposed VNOC+H.264 video decoder simulation framework. We used ten diverse H.264 encoded video streams as our benchmarks, which we downloaded from [12], [13]. These video streams are real encoded videos that we feed as input into the video decoder, which decodes and displays them. All these video streams contain 100 frames and each frame is 352x288 pixels. The default NoC architectural configuration parameters utilized in our simulations are shown in Table I. To estimate the power consumption, the simulator is integrated with the Orion 2.0 power model for NoCs for a 65nm technology node [14], validated with real data from the Intel's 80 core chip [15].

TABLE I
NoC and DVFS algorithm configuration parameters.

Parameter	Value
NoC topology	Mesh
Size	3x3
Input buffer size	16
Output buffer size	16
Packet size	6
Flit size	16
Link length [mm]	2
Virtual channel number	2
Routing algorithm	XY
Base frequency and voltage	[2GHz, 1.2V]
Throttle frequencies and voltages	[1.8GHz, 1.1V], [1.6GHz, 1V]
Boost frequency and voltage	[2.5GHz, 1.2V]

A. Impact of NoC Mapping

In our investigation we selected four different placements corresponding to four different mappings of the seven H.264 video modules to seven of the nine routers of the 3x3 regular mesh NoC. These placements are shown in Fig.5.

Each of these placements are simulated using the ten different video stream benchmarks, which are provided as actual input into the video decoder. By feeding these video streams as input into the full system simulator, we effectively exercise the network-on-chip with truly real traffic. The nature of this traffic is determined by the functionality of the H.264 decoder application and the content of the video stream supplied as input. Thus, 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 objective of these simulations is to see the impact of the NoC mapping on power consumption, average latency, and power delay product (PDP). The results are summarized in Fig.6. These results show that the *placement1* from Fig.5.a is the best one, which is what we expected because it was manually done such that highly communicating video modules were placed as close as possible.

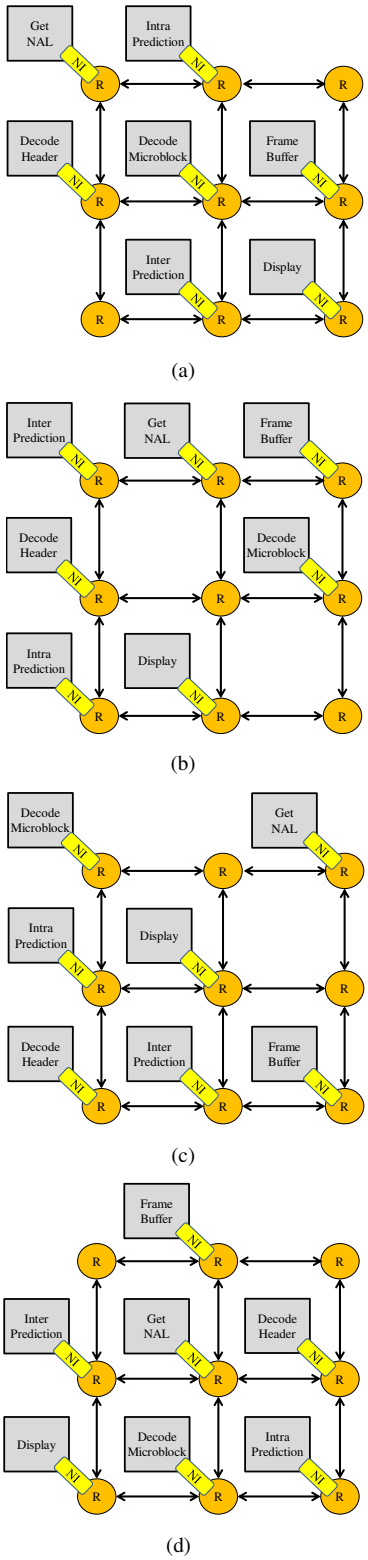
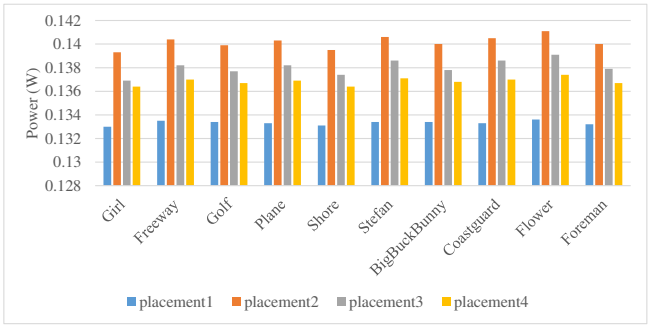


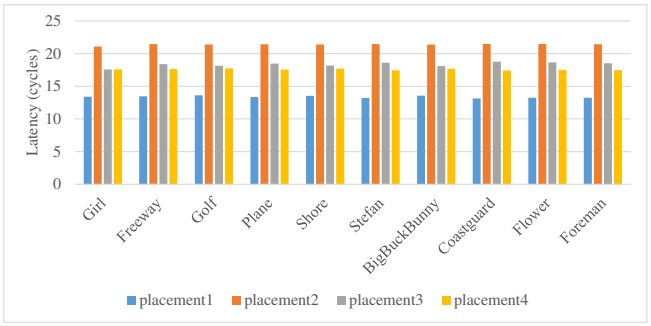
Fig. 5. Four different manual placements of the H.264 video decoder modules to a 3x3 regular mesh NoC. Each of the modules is mapped to its own router of the NoC to which is connected via the network interface.

B. Impact of DVFS Algorithm

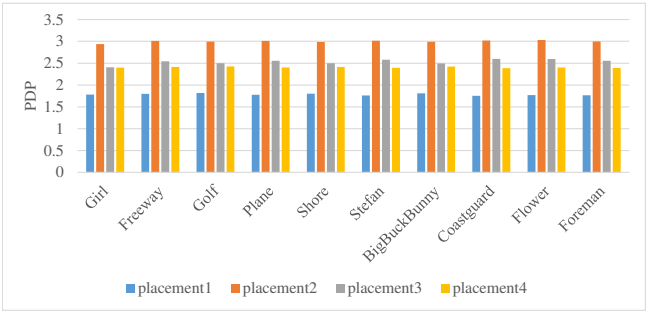
In the second set of simulations, we investigate the impact of the dynamic voltage and frequency scaling (DVFS) algorithm studied in [10] using the *placement1* that was found the best in the previous section. Each of these simulations is done with and without the DVFS algorithm being enabled. The



(a)

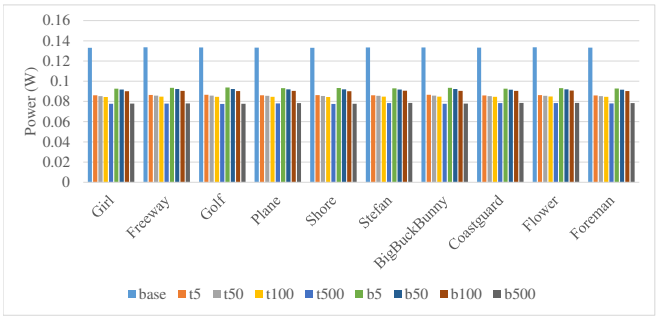


(b)

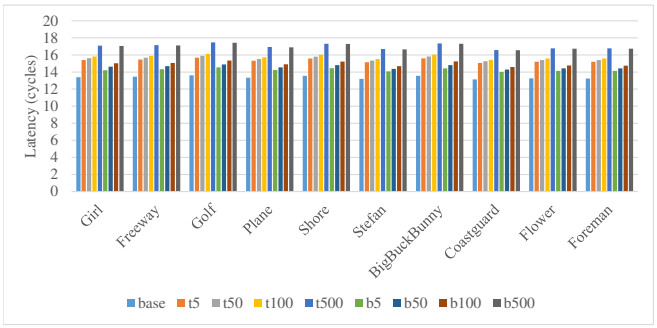


(c)

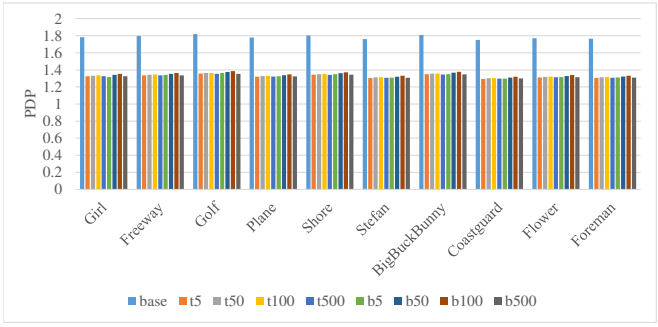
Fig. 6. Investigation of the impact of video decoder module placements on power, latency, and power delay product of the NoC. The horizontal axis lists the name of the video stream benchmarks. (a) Power consumption, (b) Average network latency, (c) Power delay product (PDP).



(a)



(b)



(c)

Fig. 7. Investigation of the impact of DVFS on power, latency, and power delay product. The horizontal axis lists the name of the video stream benchmarks. (a) Power consumption, (b) Average network latency, (c) Power delay product (PDP).

reference or base case is when DVFS is turned off. When the DVFS algorithm is turned on, all the routers are set initially to the base frequency and base voltage. Then, as the simulation progresses, frequencies and voltages are changed dynamically as described in Fig. 4. Following the approach in [10], we have ran our simulations for different lengths of the history window (HW) that characterizes the link and buffer utilization predictors.

With DVFS turned on, we investigate two different modes. The first is when only frequency throttling is used. The second mode is when we also allow frequency boost, which is 25% higher than the base frequency. This is similar to the study in [16] and is often employed in real designs with overclocking strategies [17]. The simulation results are summarized in Fig.7, where for each of the ten video stream benchmarks (shown on the x axis) we show nine data points (i.e., a cluster of nine bars), which include the *base* case (i.e, no DVFS being used), the cases where frequency is only throttled *t5*, *t50*, *t100*, *t500* for history windows of length 5, 50, 100, 500 control periods, and finally the cases where frequency can be both throttled and boosted *b5*, *b50*, *b100*, *b500* again for history windows of length 5, 50, 100, 500 control periods.

We observe that as the history window is increased the impact of the DVFS in terms of reducing power consumption is more significant as shown by the smaller power consumption for each of the ten benchmarks in Fig.7.a. In addition, as expected, once frequency boosting is enabled, the power consumption increases compared to when only frequency throttling is used (see taller four bars on the right hand side for each benchmark in Fig.7.a). Frequency boost does help to

reduce the average network latency compared to when only frequency throttling is used as seen in Fig.7.b. However, the latency is larger than in the base case when no DVFS is used. This is because when any of the video stream benchmarks are processed by the VNOC+H.264 simulator, the NoC is not operated at a packet injection rate that is high enough to trigger a lot of frequency boosting. Because at low and moderate packet injection rates mostly frequency throttling is triggered by the DVFS algorithm, the latency is worse than the base case. Nevertheless, from a power delay product (PDP) perspective the DVFS algorithm has a positive impact in that the PDP values are lower across the board compared to the base case as seen in Fig.7.c.

C. Qualitative Validation of Decoded Video Quality

Finally, to validate the VNOC+H.264 video decoder working under DVFS, 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.8 shows qualitatively a side by side comparison of selected frames from the decoded output of one of the video stream benchmarks. The decoded video streams are the same confirming the correctness of the NoC based implementation.

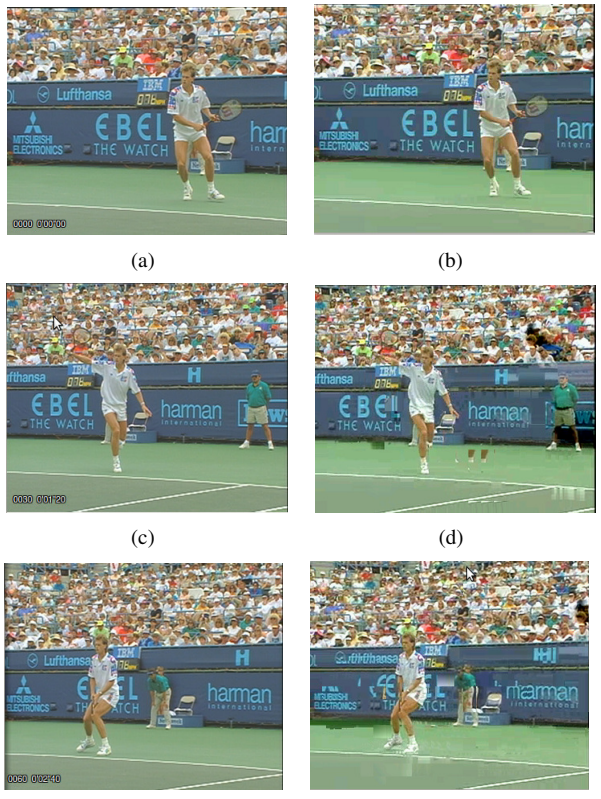


Fig. 8. Qualitative comparison of three different frames processed with the original memory-mapped based communication H.264 decoder and with the new NoC based communication H.264 decoder. a), c), e) Frames index 1, 30, and 60 of video stream benchmark *Stefan* decoded by the reference memory-mapped implementation. b), d), f) The same frames decoded by the NoC based implementation.

V. Conclusion

As a departure from traditional NoC simulation tools that use mostly synthetic traffic workloads, we described the use of an NoC simulation framework, that is able to exercise the NoC with truly real traffic in order to investigate dynamic voltage and frequency scaling for a network-on-chip based H.264 video decoder. The simulation framework combines both the NoC and the H.264 video decoder modules. Without such a simulation capability, the NoC would be designed using simulations based on synthetic traffic, which may not be accurate and could lead to sub-optimal solutions.

References

[1] P. Guerrier and A. Grenier, "A generic architecture for on-chip packet-switched interconnections," *ACM/IEEE Design Automation and Test in Europe Conference (DATE)*, pp. 250-256, 2000.

[2] W.J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," *ACM/IEEE Design Automation Conference (DAC)*, pp. 684-689, June 2001.

[3] 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. on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 28, no. 1, pp. 3-21, 2009.

[4] BookSim Interconnection Network Simulator, Stanford University, 2016. [Online]. Available: http://nocs.stanford.edu/cgi-bin/trac.cgi/wiki/Resources/BookSim.

[5] Noxim - the NoC Simulator, University of Catania, 2016. [Online]. Available: https://github.com/davidepatti/noxim.

[6] GARNET: a detailed on-chip network model inside a full-system simulator, MIT, 2016. [Online]. Available: http://projects.csail.mit.edu/cgi-bin/wiki/view/LSPgroup/GarnetPage.

[7] Software downloads at MESS Lab, Marquette University, 2016. [Online]. Available: http://dejazzer.com/software.html.

[8] N. Binkert, B. Beckmann, G. Black, S.K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D.R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewall, M. Shoaib, N. Vaish, M. D. Hill, D.A. Wood, "The gem5 simulator," *ACM SIGARCH Computer Architecture News Archive*, 2011.

[9] M.G. Moghaddam and C. Ababei, "Performance evaluation of network-on-chip based H.264 video decoders via full system simulation," *IEEE Embedded Systems Letters*, Under Review, 2016.

[10] C. Ababei and N. Mastronarde, "Benefits and costs of prediction based DVFS for NoCs at router level," *IEEE Int. SoC Conference (SOCC)*, 2014.

[11] Martin Fiedler, Implementation of a basic H.264/AVC Decoder, 2016. [Online]. Available: http://keyj.emphy.de/files/projects/h264-src.tar.gz

[12] FastVDO: H.264 Video streams, 2016. [Online]. Available: http://www.fastvdo.com/H.264.html

[13] YUV Video Sequences, 2016. [Online]. Available: http://trace.eas.asu.edu/yuv

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

[15] S.R. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, V. Erraguntla, C. Roberts, Y. Hoskote, N. Borkar, and S. Borkar, "An 80-tile sub-100-W TeraFLOPS processor in 65-nm CMOS," *IEEE Journal of Solid-state Circuits*, vol. 43, no. 1, pp. 29-41, Feb. 2008.

[16] A.K. Mishra, A. Yanamandra, R. Das, S. Eachempati, R.R. Iyer, N. Vijaykrishnan, and C.R. Das, "RAFT: a router architecture with frequency tuning for on-chip networks," *J. Parallel Distrib. Comput.*, vol. 71, no. 5, pp. 625-640, 2011.

[17] D. Lo and C. Kozyrakis, "Dynamic management of TurboMode in modern multi-core chips," *IEEE Int. Symp. on High Performance Computer Architecture (HPCA)*, 2014.