Embedded debugging for NoCs

Marta Stępniewska, Olgierd Stankiewicz, Adam Łuczak, Jakub Siast
Chair of Multimedia Telecommunications and Microelectronics
Poznan University of Technology
Poznań, Poland
{mstep, ostank, aluczak, jsiast}@multimedia.edu.pl

Abstract—**This paper presents a proposal of debugging enhancements embedded in Network-on-Chip. Such approach makes debugging functionality hardly attainable in classical hardware methodology. We introduce real-time packet sniffing and monitoring with the use of multicast transmission. Each packet in the debug mode is additionally sent to an off-chip control device, called Off-chip Debugging Host (ODH), as a multicast stream. This enables packet sniffing similar to the one that is available in computer networks. Protocol enhancements also include mechanisms for fast and simple configuration. Thanks to introduced control packets, Off-chip Debugging Host is able to remotely read NoC's settings as well. The proposal has been assessed basing on hardware implementation.**

Index Terms—**NoC; debugging; H.264; AVC; design-for-debug;**

I. INTRODUCTION

Network on Chip (NoC) is a relatively new approach that provides a methodology to design Systems on Chip (SoC) interconnections [1]-[4]. It allows for separating the design of communication infrastructure from processing elements (PEs) implementation. Such approach provides possibility of reusing network and processing modules. Application of NoC as a communication infrastructure leads to a significant reduction of design time of the whole system. These advantages are important since current designs became more and more complex and include tens or even hundreds of PEs.

By these modules however, some errors may arise during designing phase. Moreover, many system failures result from extensive PEs interaction that is hardly predictable before connecting them together. Debuging of current NoC-based Systems on Chips requires communication-oriented approach that focuses on correctness of messages that are exchanged between PEs. In order to apply this method it is necessary to embed some debugging functionality in NoC architecture i.e. design-for-debug, that helps to understand possible SoC's failures.

The well known debugging technique which employs standard for in-circuit test is JTAG [5] protocol, which originally was intended for system management tasks. Such approach requires two physical components: the first is test access port (TAP) which interprets JTAG protocol, and the second is boundary scan register (BSR). Implementation of these modules in each PE consumes large amount of chip resources regarding scale of current designs. Tang and Xu in [6] propose to use JTAG as a debug interface in conjunction with NoC. A chip includes a debugging core, called "debug agent", that contains TAP. "Debug agent" exchanges debugging data with off-chip controller through Test Access Port. Off-chip controller is a hardware that transfers data gathered from the chip to debug software placed on PC. Moreover, in [6], authors suggest to attach debug probes to the core under debug (CUD) i.e. each tested PE. These probes are connected to the NoC via separate network interface. Such solution is not cost efficient (in terms of chip area) due to significant rise of NoC elements.

In turn, Yi et al. proposed in [7] and [8] so called "core debug supporters" which are small distributed debugging cores. These are attached to the system cores and to test&debug interface unit for the sake of management of transactions. Test&debug control data are sent via data control bus IEEE 1149.1 but data application and observation are transported through parallel NoC data path. Despite its advantages, such solution doesn't allow for watching messages that are sent in NoC.

In [9] and [10] authors propose transaction-based NoC monitoring. In this proposal, debugging cores, called "monitors" are attached to network elements. These monitors make measuring link performance and analysing of protocol specific data possible. Although the cost of such monitors (in terms of occupied chip resources) may be controlled by reducing their implemented functionality, it still may be significant for hardware-cost-sensitive designs. Moreover, such solutions significantly impact performance of the network or negatively influence chip resources, because results of traffic analysis are transported over NoC [9] or through dedicated links [10] respectively.

In this paper, we try to tackle abovementioned problems. We propose to add some low-cost (in terms of chip resources) enhancements to NoC infrastructure, that allow to observe and analyse data exchanged between PEs, outside the chip. The main idea of our novel NoC-based communication-oriented approach is presented in Section II. It comprises NoC enhancements described in Section III, which are used in order to provide devised debugging functionality (Section IV). Because our work is focused on FPGAs, assesment of the idea has been performed with the use of exemplary FPGA implementation of AVC/H.264 video decoder. Sections V and VI present results regarding use-cases and chip size.

II. IDEA OF THE PAPER

This paper presents a proposal of debugging enhancements to support communication-oriented debug of a system being designed. The proposed novel set of debugging mechanisms consists of:

- Control device that analyzes NoC packets, described below, called Off-chip Debugging Host (ODH),
- Set of control packets for management of debugging process,
- Programmable communication channels, that enable packet sniffing, novel for NoC networks,
- Novel NoC multicast transmission mode to reduce required bandwidth (debugging overhead).

The main element of the proposed debugging approach (Fig. 1) is Off-chip Debugging Host (ODH). ODH is an off-chip software entity run entirely on a personal computer. It is connected to the system through a physical interface. Such off-chip ODH placement allows advanced debugging tools to be implemented without affecting system hardware resource consumption. Moreover, new functionalities can be added to ODH without time-consuming design resynthesis process.

The role of ODH is to provide user interface to the debugging functionality, like: NoC monitoring, packet sniffing, applying test vectors, gathering debugging data, handling exceptions or emulating hardware PEs in software.

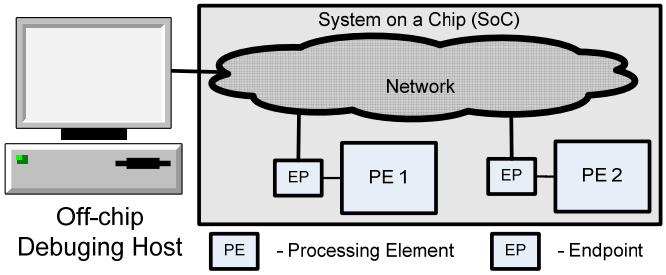


Figure 1. Idea of NoC-debugged system with ODH.

The proposed enhancements are not network-specific in principle. They can be embedded into an already designed and implemented Network-on-Chip without having serious impact on hardware complexity. Moreover, some of them, like multicast transmission mode, can be successfully exploited in regular NoC operations.

The devised ideas can be applied to variety of network topologies, like mesh, tree, ring, etc. Also, routing algorithm need not to be altered significantly – it can be geographic (coordinate) -based like in 2D-mesh) as well as any other, as long as it can support multicast mode.

In our approach, network is an abstract object which allows communication between PEs (Fig. 1) with the use of endpoints (EP). These endpoints implement simple networking functions and provide network interface for the processing elements.

All of these compatibilities together with the variety of network architectures increase applicability of our proposal.

This work was supported by the public funds as a research project.

III. PROPOSED DEBUGGING ENHANCEMENTS

In order to introduce a novel debugging functionality we propose some network enhancements. Although, in principal, our proposal abstracts from exact network specifics, the following tools may impose modifications to the existing NoC implementations. Results presented in Section VI, show that the hardware cost is not significant.

A. *Off-chip Debugging Host*

Off-chip Debugging Host constitute user interface for debugging functionality. It is composed of software applications that run on a personal computer (Fig. 2).

ODH require only minimal unavoidable hardware: a physical connection to the NoC communication infrastructure. This implements NoC protocol and provides a gateway to the NoC. In this way, with use of appriopriate API (Application Programming Interface), debugging applications are able to send and receive packets and thus are logically part of the NoC.

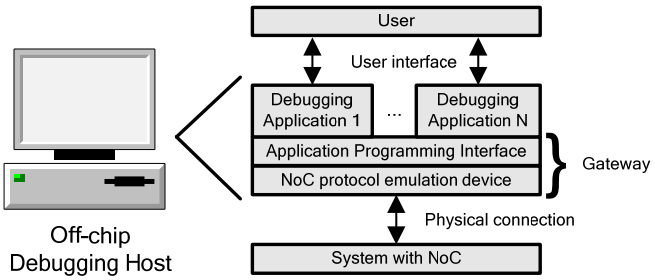


Figure 2. Architecture of Off-Chip Debugging Host, connected to system with Network-on-Chip.

ODH software comprises variety of debugging applications which provide advanced debugging tools and graphical interface for the user. These include: packet sniffers and parsers, NoC management and setup toolkits, NoC monitors and test vector senders.

B. *Control Packets*

NoC protocol is enhanced with support for control packets classes, which functionality is presented in Table I.

These control packets offer functionality that is necessary for efficient debugging control by ODH. Control packets described in Table I are introduced to read and write EP and PE configuration. However, GET_STATUS packet can be applied to other network elements, such as switches, and allows for reading rooting tables.

TABLE I. CONTROL PACKETS IN NOC

Control Packet	Description
PE_ENABLE	Enables or disables a PE
SET_DST_ADDR	Sets new destination address in endpoint
GET_DST_ADDR	Request for current destination address
GET_STATUS	Request for status in PE or network element e.g. switch, EP
GET_PE_DESC_WD	Request for PE build-in word specifying e.g. type, functionality, version etc.

Control Packet	Description
PE_ENABLE	Enables or disables a PE
SET_DST_ADDR	Sets new destination address in endpoint
PE_RST	Provides soft PE reset

C. *Programmable communication channels*

Communication channel is a path in network between a source endpoint and a destination endpoint. In typical NoC approach, these are staticly set at design time and cannot be altered dynamicly. We propose a novel approach in which communication channels are programmable: communication channel can be arranged between any two PEs at run-time. To achive that, processing element endpoints are extended with programmability of destination addresses (please note that EP may have more than one destination address). For example, destination address list of EP (originally requested by the PE) can be extended by ODH with an additional address for debugging. Moreover, originally set addresses can be substituted with another ones.

This constitutes an additional virtual abstraction layer over communication in NoC. Such abstraction, gives an opportunity to compare work results between resources (for example different versions of designed PE – Fig.3 a) versus b) – PEs 2 and 3). Apart from debugging function, programmable destination addresses can also be used to implement network-based resource management. Such management allows to distribute tasks between redundant system resources dynamically or to attach some restricted resources to many clients alternately.

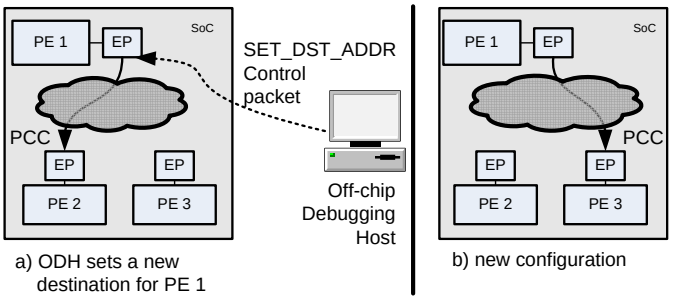


Figure 3. Programmable communication channels (PCC) reconfiguration performed by ODH in real-time.

D. *Multicast*

Classic NoC networks support only unicast transmission that allows for delivering a single packet between single source and single destination [11]. Such approach leads to significant traffic overhead in case of use of debugging mechanisms (like packet sniffing) because traffic has to be duplicated at the beginning of packet routes (Fig. 4a).

To overcame this problem, in this work we propose to enhance NoC with multicast transmission and use it for debugging. Multicast is a packet delivery strategy which allows for simultaneous transmission of packet over network to multiple destinations only once over each link on the path. The packet is duplicated only in case of splitting routes. Implementation of multicast on the top of existing unicast architecture is rather straighforward: instead of single destination address (Fig. 4a), a list of multiple addresses is assigned to each multicast packet (Fig. 4b). Of course, at some switching point of the network, the packet is duplicated, but in case of multicast transmission it is done as far from the source as possible. Finally, packet is transmitted to every destination from the address list. Such functionality, allows for sending a copy of the packet to a network monitoring element like ODH without much bandwith loss.

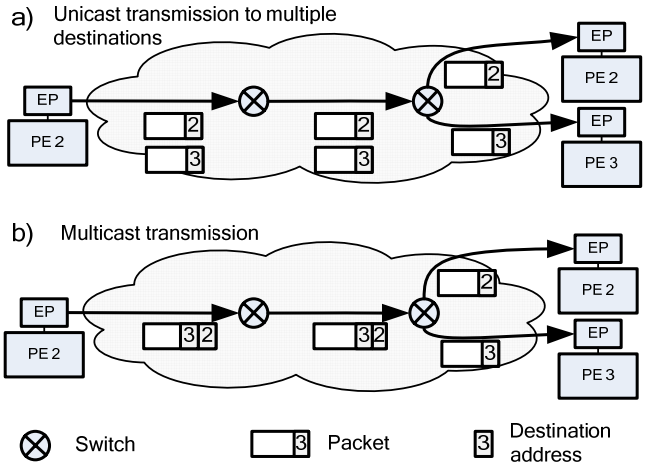


Figure 4. Packet propagation diagram for unicast transmission to multiple destinations (a) and for proposed multicast communication (b).

IV. PROPOSED FUNCTIONALITY.

The abovementioned proposed network enhancements extend functionality of NoC beyond pure communications. These enhancements are a foundation for desired debugging tools. Debugging functionalities proposed in this paper, of which some examples are mentioned below, heavily depend on adjustment of Off-chip Debugging Host software for the specific application. We propose the following debugging tools:

A. *Packet sniffing in endpoint debug-mode*

Packet sniffing is a technique of exhaustive packet monitoring commonly known in computer networks. In this paper, we propose to enhance NoC network with such debugging functionality. In our proposal, packet sniffing is enabled in so called debug mode which is initiated by ODH. When in debug-mode, endpoints use multicast transmissions and add ODH address to each packet address list. ODH receives copies of the packets and uses special packet parsing software application to recognize packets and then present to the user. Further, correctness of packets and data format are verified. Debug-mode can be switched on and off for each PE endpoint individually.

B. *Reading configuration and status*

Off-chip Debugging Host can read configuration and status of any network element. Such functionality is useful when configuration of SoC PEs has to be verified. With the proposed

tools, network configuration can be retrieved dynamically. Also the types of attached PEs can be recognized. This is a faster way than manual configuration and can help evading human mistakes in managing network configuration.

C. Turning the PEs on and off

User can turn PEs on and off by sending control packets (PE_ENABLE packet). It is usedl when processing element declines to work as a potential result of an error. Knocked PE (Fig. 5 – PE 2) can be then replaced with an alternative implementation (e.g. other revision of design – Fig. 5- PE 3).

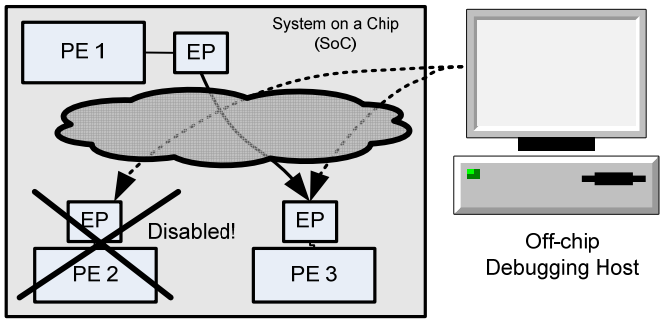


Figure 5. Turning exemplary PE 2 off and turning exemplary PE 3 on.

D. PE emulation

When only a sub-part of the design is ready to be placed on a chip and the rest is still in simulation phase, it may be useful to split the design into hardware and software. Namely, hardware is a part of design that is already implemented, and software emulates missing or broken part of a design. Partial design emulation may easily be done using devised Off-chip Debugging Host. On Fig. 6, exemplary PE 4 emulates hardware PE 2, which may be broken or simply not implemented in hardware completly.

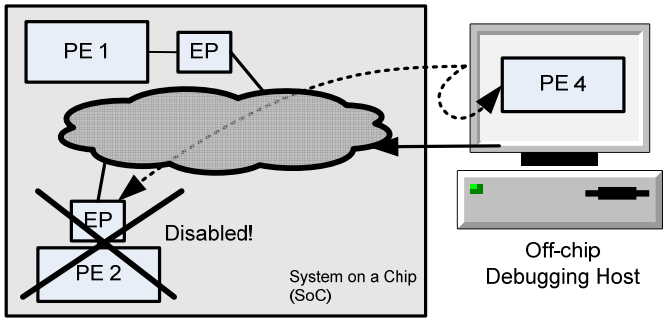


Figure 6. PE emulation on Off-chip Debuging Host.

V. Assesment.

Assessment of a debugging functionality is not a straight-forward task. There are at least two factors that have to be concerned: savings of development time during the design process and costs of implementation in silicon. The results for the latter factor are difficult to be measured precisely, due to automatic optimization during synthesis process, but can be estimated (Section VI).

As for the first factor, unfortunately it is difficult to objectively measure development time spent on debugging with and without debugging amenities, in order to compare them. In the real world, repeting the design process is not possible, because it would have to be completed by the same persons with the same initial knowledge of the problem. The other option is to get by the design process many times, under random human conditions, to determine the design time statistically. This was unobtainable for us, because design of typical system takes about few person-years, and there would have to be tens or even hundreds of experiments done simultaneously.

Instead of this, we have assessed the proposed ideas during designing, implementing and testing of AVC/H.264 video decoder. At that time, we have observed many examples of debugging functionality usage, that otherwise would be very difficult to attain. An example scenario is presented below.

The whole AVC/H.264 video decoder project was implemented on two separate circuit boards (Fig. 7). This was motivated by the size of the project (which has an impact on the synthesis time) and also by a variety of external components exploited by the system (that were difficult to connect to a single FPGA). The boards were developed separately in order to hasten the design.

At the time of linking the boards we encountered a problem – the boards definitly declined to work together. After launching the decoding task, both boards just hung-up. Since the simulation worked perfectly, the problem must have laid in the hardware part.

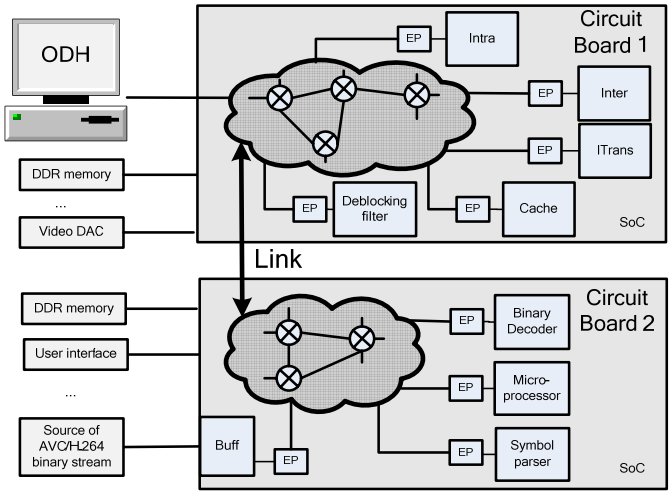


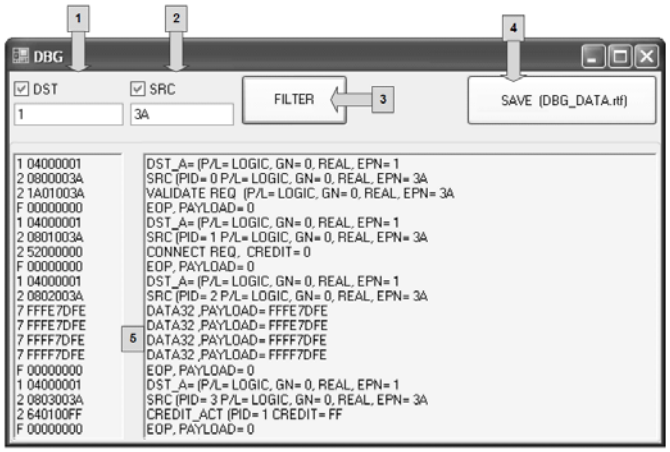
Figure 7. Assesed AVC/H.264 decoder system on two circuit boards.

Without debugging functionality, we would be forced to resynthesize the whole project with additional testing benches in order to conclude what was wrong. The transmission through the link was corrupted, there was some kind of hazard situation somewhere or simply there was a synthesis error.

In spite of doing extra work, we have used the proposed functionality of reading configuration and status (GET_STATUS and GET_PE_DESC_WD packets) of network elements on the Circuit Board 2. The whole process

ran correctly – the packets were able to propagate from ODH to destination EPs and get back again. The verdict was that the link is operating correctly as well as network infrastructure on both boards.

The next step was to run decoding task and try to find the reason of the crash. In this case, we have used packet sniffing tool in a debug-mode (Fig. 8) with the use of SET_DST_ADDR packets.



1 – destination address filtering function enable
2 – source address filtering function enable
3 – filter execution button
4 – filtered data save button
5 – filtered data preview with protocol analyzer

Figure 8. Examplary packet sinffing with Packet parser application for "Intra" PE.

Analysis of the packets delivered to ODH, which were sent from/to Circuit Board 1, revealed that there were errors in "Intra" (Circuit Board 1 – Fig. 7) output formatting sub-module. In order to find the exact source of the problem we have switched "Intra" PE off (PE_ENABLE), plugged "Intra" processing element emulator into ODH, and redirected other PEs' requests accordingly (Fig. 9).

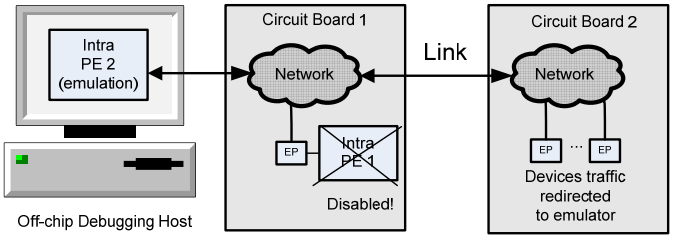


Figure 9. '"Intra" PE emulation in AVC/H.264 decoder.

Once we have done it, the whole project run well, which has narrowed down list of potential error sources to mentioned "Intra" processing element. Detailed analysis of captured packets led to creation of additional test vectors which finally resulted in bug eradication from the "Intra" module at the simulation level.

The whole error identification process took few hours instead of few days, which is typical in such situations. Although it is just an example of benefits that result from proposed debugging functionality, it is worth to mention, ast scenarios like the one described above are very likely to happen.

VI. Results

The exemplary AVC/H.264 project has been modeled in Verilog HDL and synthesized for Xilinx Virtex 5 FPGA devices to assess size costs of the proposed tools.

As mentioned before, the estimation of real chip occupation by network infrastructure is a very difficult task. Synthesis of the whole system without any communication network is not representative, because such synthesis would be affected by optimization.

To estimate growth of the costs of costs of the proposed tools, first we have measured the whole NoC contribution to the whole project size. Network implemented on two boards includes three 3-port switches (Switch 3p) and one 4-port switch (Switch 4p). Morover, 4 bidirectional (receiver and transceiver) endpoints (EP BD) were used that support both receiving and transmitting data. Additionaly, 8 endpoints used as receivers (EP RX). Implemented routing protocol is static, i.e. the switches use build-in static routing tables. Links between network primitives are 32 bits wide with 4 bit description word. The endpoints (both BD and RX) have 2-port multiplexer build-in. This way, they can be connected serially without using switches between them and save chip resources, since they are smaller than ordinary switches. As presented in Table II, the cost of used NoC network is relatively small - 16% in LUTs and 13% in registers cost of whole AVC/H.264 decoder.

TABLE II. NoC contribution in AVC/H.264 decoder

Unit	Unit Size	
	LUTs	***Registers***
Whole project	34354	29723
Network infrastructure	5344	3952
Network contribution	16%	13%

TABLE III. contribution of proposed tools' cost in NoC modules

Network primitive	With tools		Without tools		Tool contribution	
	LUTs	***Regs***	***LUTs***	***Regs***	***LUTs***	***Regs***
Switch 3p	658	570	615	523	6,5%	8,2%
Switch 4p	1068	759	1008	699	5,6%	7,9%
EP BD	428	335	376	292	12%	13%
EP RX	328	280	280	237	15%	15%

What is more important, proposed debug tools have very small impact on size of the whole NoC. Table III sums the hardware costs of implementation of NoC's components - with and without the proposed tools. Tool contribution in case of 3

and 4 port switches is at most 8.2% for register number. For endpoints (marked as EP BD for bidirectional interface and EP RX for receiver-only interface) it can be noticed that the worst case is 15% growth of the registers number.

Taking results of Table II and Table III into consideration, it can be concluded that resultant impact of having proposed debugging functionality is relatively small.

We have also assessed the efficiency of the proposed multicast transmission mode. This efficiency depends on network topology and ODH placement in the network i.e. how source–destination path differs from source–ODH route. Table IV compares size of traffic when using multicast transmission to ODH in debug mode with the unicast approach. Both results are compared with traffic in the application in case of the debug mode switched off in all IP cores. Moreover our implementation allows for choosing single PE for debug, which can improve traffic statistics. Table IV considers 3 cases: debugging inverse transform module, deblocking filter only, and debugging all PEs in AVC/H.264 decoder. First two cores are responsible for about 50% of whole traffic in application, so the ODH needed to be placed in the vicinity of both. Comparing results for the third case i.e. debugging all cores in AVC/H.264 application, the gain achieved by mulicast transmission is about 50%.

TABLE IV. MULTICAST EFFICIENCY

Debugged PEs	**WITHOUT transmission to ODH**	**WITH MULTICAST transmission to ODH**	**WITH UNICAST transmission to ODH**
Inverse transform module	100%	108%	118%
Deblocking filter	100%	113%	128%
Whole AVC/H.264 decoder	100%	150%	200%

VII. CONCLUSIONS

Debugging is an integral part of the design process but so far it has been omitted and treated like a standalone problem. Experiences in that matter lead to conclusion, that design of a SoC is an iterative process, in which debugging and evolving occur alternately.

In this paper, we have introduced a new debugging approach in which the design is debugged with the use of Off-chip Debugging Host. ODH provides user interface to debugging functionalities, like applying test vectors, gathering debugging data and handling exceptions. Because such off-chip host runs entirely on personal computer, significant amount of hardware chip resources can be saved. Another advantage of such approach is that new functionalities can be added to ODH without time-consuming design resynthesis process.

Our novel debugging approach is based on network enhancements that can easily be embedded into variety of NoC networks. We have described these enhancements. They include proposals that are new to Networks-on-Chip, like programmable communications channels. We have discussed functionality benefits resulting from those enhancements, showing possible usage like: network packet sniffing and monitoring, reading of configuration and status of attached PEs and PE emulation on Off-chip Debugging Host.

Finally, we have made an assessment of the proposed debug system on an exemplary real debugging scenario. As a conclusion we point out that the proposed enhancements provide high level debugging methodology and can significantly speed up implementation process. Moreover, synthesis results in Section VI show that cost of implementing the required network enhancements is slight. Tool contribution in case of the endpoint is not more than 15% and in case of switches it is less than 8,2%. Costs to benefits ratio incline towards approach in which debug functionality is incorporated in the SoC even in the final product, to enable future in-the-field debugging, modifications and enhancements.

REFERENCES

[1] C. Chou, R, Marculescu, “Run-Time Task Allocation Considering User Behavior in Embedded Multiprocessor Networks-on-Chip”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 29, no. 1, pp. 78-91, 2010.

[2] Q. Yu, P. Ampadu, “A Flexible Parallel Simulator for Networks-on-Chip With Error Control” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 29, no. 1, pp. 103-116 2010.

[3] A. Ejlali, B.M. Al-Hashimi, P. Rosinger, S.G. Miremadi, L. Benini, “Performability/Energy Tradeoff in Error-Control Schemes for On-Chip Networks”. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 18, no. 1, pp. 1-14, 2010.

[4] C. Hilton, B. Nelson, “PNoC : a flexible circuit-switched NoC for FPGA-based systems”, IEE Proceedings on Computers and Digital Techniques, vol.153, no.3, pp. 181-188, 2006.

[5] IEEE Std 1149.1-1990 IEEE Standard Test Access Port and Boundary-Scan Architecture –Description.

[6] S. Tang, Qiang Xu; , "A Multi-Core Debug Platform for NoC-Based Systems," Design, Automation & Test in Europe Conference & Exhibition, pp.1-6, 16-20 April 2007.

[7] H. Yi, S. Park and S. Kundu, "On-Chip Support for NoC-Based SoC Debugging," IEEE Transactions on Circuits and Systems I: Regular Papers, no.99.

[8] H. Yi, S.Park and S. Kundu, "A Design-for-Debug (DfD) for NoC-Based SoC Debugging via NoC," 17th Asian Test Symposium, pp.289-294, 24-27 Nov. 2008.

[9] C. Ciordas, K. Goossens, T. Basten, A. Radulescu and A. Boon, "Transaction Monitoring in Networks on Chip: The On-Chip Run-Time Perspective," International Symposium on Industrial Embedded Systems, pp.1-10, 18-20 Oct. 2006.

[10] B. Vermeulen and K. Goossens, "A Network-on-Chip monitoring infrastructure for communication-centric debug of embedded multi-processor SoCs," International Symposium on VLSI Design, Automation and Test, pp.183-186, 28-30 April 2009.

[11] Erno Salminen, Ari Kulmala, Timo D. Hämäläinen, Survey of Network-on-chip Proposals, White Paper, OCP-IP, March 2008.