Fast Design Space Exploration Environment Applied on NoC's for 3D-Stacked MPSoC's.

Alienor Richard, Dragomir Milojevic, Frederic Robert
Bio Electro and Mechanical Systems, Université Libre de Bruxelles, Belgium

Alexandros Bartzas, Antonis Papanikolaou, Kostas Siozios, Dimitrios Soudris
Microprocessors and Digital Systems Lab, National Technical University of Athens, Greece

Abstract

In this paper we present a design methodology and associated tool chain for efficient design of complex MPSoC architectures implemented using 3D-Stacked Integrated Circuits (3D-SIC). The proposed framework is based on a three step methodology that combines relatively accurate high-level, and two more accurate low-level prototyping tools. The high-level exploration tool, *Nessie*, developped at the ULB, allows designers to quickly simulate several system architecture and application scenarios. Using Nessie, the designer can easily explore many different system level options before deciding on the design space points he would like to explore in more details. In this paper, the high-level estimations are subsequently validated using an existing C++, transaction-level, 3D-SIC aware NoC simulator. Then, floorplaning and global routing of the system are performed using a novel tool for 3D physical prototyping. The low-level performance metrics of the system are derived from the resulting physical prototype and can be compared to the results predicted by *Nessie*. We demonstrate our approach using the example of a fairly complex MPSoC platform dedicated to advanced high-performance and low power video coding applications (AVC/H.264 encoder). The MPSoC platform is prototyped as traditional 2D-IC and then as 3D-SIC design using various 3D stack configurations and stack assignment schemes.

1 Introduction

The design of modern embedded Systems-on-Chip (SoC) is a difficult process of finding a satisfying solution obeying to functional (more and more complex applications) and non functional (energy, area, cost, time-to-market) constraints, further aggravated by the increasing number of design options. This is particularly the case with multimedia applications and their stringent energy dissipation requirements for handheld consumer devices. MPSoCs are prevailing as the target platform architecture for this type of applications because they offer a good trade-off between cost development and time-to-market constraints. However, the design of such a system still relies on a huge search space where both the application and the architecture choices affect final performances. If Networks-on-Chip (NoC) are the natural choice for on-chip communication for these systems, due to their scalability and the fact they do not interfere with programming models, their design has also to be correctly fixed for guaranteeing proper exchange.

Moreover, to keep Moore's law momentum alive, novel physical implementation can be explored. An example is 3D chip stacking [5]. Indeed, instead of increasing transistor density on the plane via geometrical shrinking, density in space increases by stacking multiple dies on top of each other for a given footprint. A 3D stacked architecture can be obtained by the superposition of different classical CMOS 2D Integrated Circuits, leading to better performance and higher functionality density. Thanks to already tested fabrication technology based on Through-Silicon Vias (TSVs) [13], these different layers can communicate together at any location on the layout and with very short vertical links when thinned wafers are used [12]. This technology can clearly offer benefits to the implementation of complex MPSoCs, albeit at the cost of increasing the design and implementation options drastically.

Conventional simulation based CAD tools are not time-efficient enough to handle all these choices at once, hence options are de-coupled and explored sequentially, which lead to global sub-optimalities and a lot of iterations. High-level exploration tools that can take into account all these options, at the cost of reduced accuracy, are thus becoming necessary to take global decisions at a high abstraction level. The example of such tool, Nessie, has been outlined in [15]. In this paper we intend to illustrate the ability of this tool to model complex systems and perform quick simulations of several solutions efficiently in order to limit further iterations in the design flow. We will apply the proposed tool chain on the design of a AVC/H2.264 video encoding application running on a fairly complex MPSoC using 3D stacking technology and show that the results that are produced at a high abstraction level are relatively accurate when compared with physical implementations of

the resulting MPSoC platforms. This increases the credibility and usability of the tool, and proves that high-level exploration tools can produce relevant results for taking strategic decisions for the design of complex systems at system level.

The paper is structured as follows. Section 2 presents the high and low-level exploration tools used for the simulations and design. Section 3 provides more details about the case study. Results for 2D and 3D architectures are given and commented in Section 4. Finally, we conclude this work in Section 5.

2 Design exploration environment

2.1 High-level exploration tool

The high-level tools we can find in the literature are often dedicated to a kind of architecture (processor e.g.), considering one criteria (most likely timing) and without automatic exploration capabilities, making them rather pure simulators than exploration tools. As only a small part of the design space is explored based on few criteria, it still leads to costly iterations and sub-optimal solutions. We believe improvement can be achieved if a high-level tool can consider a joint multi-criteria exploration of both the functionality and the architecture based on a flexible mapping core able to trade-off accuracy and simulation time. Our high-level exploration tool, Nessie, is born to face state-of-the-art tools that do not provide such solutions to nowadays hardly constrained SoC design problems.

Amongst these tools, we briefly present some interesting ones. The SESAME [10] system-level modelling environment takes the functionality and the architecture description (with SystemC) at a transaction level and enables its refinement but is devoted to MPSoC. Furthermore, the mapping does not perform automatic exploration of design possibilities, focuses on timing aspects and does not support synthesis phase. Koski [4] and Design Trotter [9] both enable automatic exploration of MP-like platforms without refinement or synthesis possibilities. The former uniformises the description of the entire system with UML but restricts this to highest level of abstraction. The latter represents the functionality thanks to Hierarchical Control and Data Flow Graph (HCDFG) and the platform with XML files and takes technological parameters into account enabling multi-objective modelling. On the contrary, Metropolis [2], using meta-models to describe functionality, platform and multi-objective mapping policy, enables refinement through different abstraction levels and offers synthesis capability without automatic exploration of the design space. Synthesis is also possible in the Cofluent [1] environment and the Chinook [3] tool, both limited to one abstraction level. The former uses SystemC to describe functionality/platform and automatically explores different multiprocessor/multicore/multitask architectures by estimating timing criteria. The latter targets FPGA and enables also compilation for processors. It is based on Verilog description of the system.

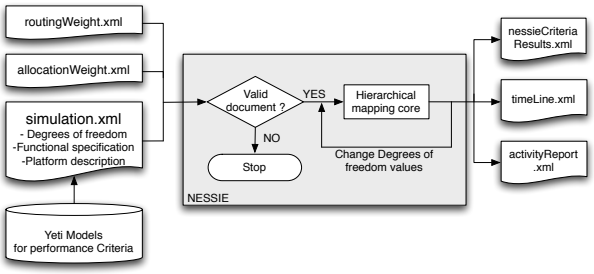

Figure 1: High-level exploration framework.

The major contribution of Nessie [14] is to propose a performance centric environment filling the gaps pointed above. As shown in Fig. 1, the tool is based on a hierarchical mapping core built in C++, interfaced with XML files. In the simulation file, the user describes the different functionality and architecture structures he wants to explore at the abstractions levels he is interested in. Petri Nets are used as Model of Computation (MoC), enabling to describe data dependencies of the functionality. Places are the basic tasks at the considered abstraction level. Transitions controls the data flow from a task to an other depending on the number of token (associated to the incoming edges) needed to fire it. The architecture is modeled thanks to three kind of nodes: computational nodes (able to execute one or more tasks defined in the Petri Nets), memory and interconnection. The user has to define costs for these nodes, depending on the criteria he wants the tool to estimate. If several abstraction levels have been defined by the user, the tool will explore hierarchically all the structures thanks to a recursive algorithm. There is no limitation on the number and kind of criteria the user can define. Nessie calculates the criteria for the global execution of the petri net on the architecture based on a dynamically-built timeline. Each time an event is triggered on an architecture block (eg: the execution of a task A), Nessie estimates the associated criteria and combines them according to rules defines by the user in the simulation file (eg: if one criterium is number of cycles, Nessie will add the cycles for each event on the block, locally, and add the local cycles of all the blocks to obtain the total number of cycles for the entire system until the end of the timeline). The mapping itself is performed in three steps, as shown in the figure 2. First, tasks are scheduled in a "first-in first-out" order. Then, Nessie allocates tasks dynamically on the less costly block according to a user-defined weight (in *allocationWeight.xml*) and routes data thanks to a Dijkstra algorithm. The mapping core communicates with the models library, *Yeti* [15], built to offer users model flexibility for easy evaluation, input sensitivity analysis and

easy model-building capability. Evaluated criteria are detailed for each functionality/architecture couple in a unique xml file. In addition, an activity report gives the relative activity of each architectural node (executing, idle, sleeping,transmitting modes) to enable an easy analysis in terms of resource needs.

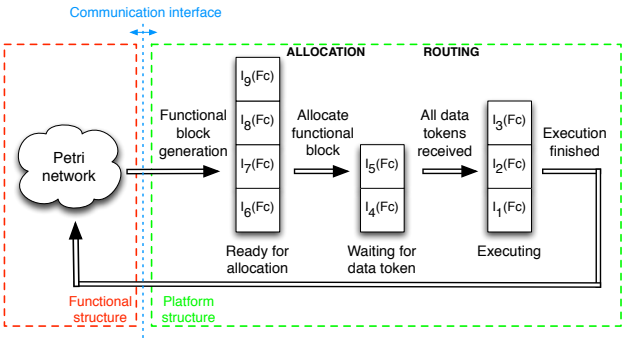

Figure 2: Dynamic mapping process in Nessie.

The originality and major advantage of Nessie lies in its ability to evaluate several solutions in one run and to offer a lot of flexibility to simulate any kind of system. It becomes particularly relevant in the design problem introduced in the first section where the huge design space is combined with multiple interesting criteria. In this paper however, we will focus on one sole criterium, the energy consumption of the NoC in a 3D systems, in order to validate our mehodology.

2.2 Low-level design flow

In order to evaluate the accuracy of the high-level exploration framework we employ a set of tools forming a low-level design flow. This flow consists in a C++ transaction level NoC simulator, namely an extended version of the WormSim NoC simulator [6], and a tool that performs physical prototyping based on floorplanning and routing of the MPSoC with NoC.

The former simulates the traffic of the interconnection network and generates the transfer traces between the various IP blocks and routers. At a lower level, after the placement and routing, it is also possible to calculate more accurate performance metrics like NoC clock frequency and power dissipation, based on the physical prototype. Furthermore, the NoC simulator can be linked to the SoC encounter (see below) and use the wire-length information in order to perform a more accurate simulation and power dissipation estimation.

In the latter, the synthesis is performed using the Cadence SoC Encounter tool suite. Similar procedure to the one we employ in this work has been used in [11]. This step of the design flow enables a more realistic (i.e., physical design aware) parameters estimation thanks to the floorplan and global routing of the entire 3D system. The area of the various system components, i.e. processor cores, memories, on-chip routers, is extracted from the gate-level description of the system and their dimensions are calculated based on the area estimations coming from the actual logic synthesis. These area calculations are independent of the implementation of the system in terms of 2D or 3D integration and the different application mapping scenarios. The connectivity of these components is extracted from the architectural diagram of the system and all required ports for the interfacing between components is extracted from the gate-level netlist. As a result, a block-level netlist of the entire system is obtained. It is used to floorplan the design, perform global routing and report the detailed wire-lengths.

While this tool flow does not perform exploration of the application or the platform, the user has to pre-define the system before going through the NoC simulator and synthesis tool in order to have power dissipation information. This can take several hours for one case and limit the number of possibilities tested by the designer. This is where our high-level tool arises as a complementary solution to the flow by enabling a quick exploration of several solutions in one shot what enables to guide the designer for the following steps.

3 Case study: MPSoC for video encoding applications

The proposed case study is based on an existing MPSoC platform designed for high-performance, low-power and flexible video encoding applications (multiple encoding standards, resolutions, frame rates, qualities etc.). An example of results from which we have based our case study is explained for a 2D layout in the paper [8]. The figures 3 and 4 resulting from the floorplan shows the architecture of the system, a 64 mm^2 square die, where the layout of the nodes and the NoC have been chosen arbitrarly.

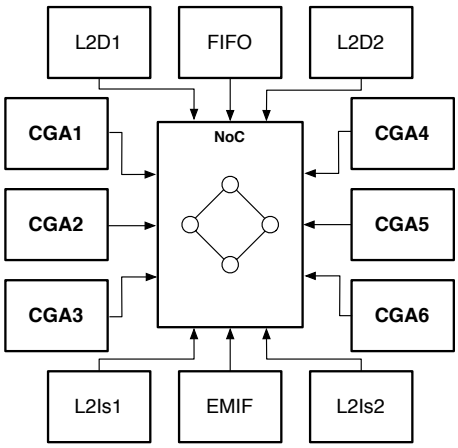

Figure 3: Block diagram of the MPSoC platform

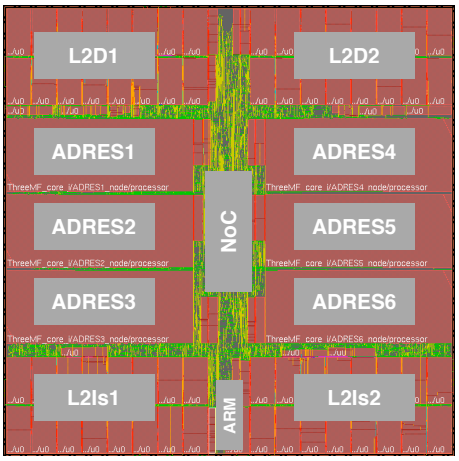

Figure 4: Layout of the MPSoC platform

Architecture - The system, described in [8], is composed of 6 Coarse Grain Array (CGA) cores, ADRES processors [7], 1 ARM processor for system configuration and control purposes, one off-chip memory interface (EMIF) and 1 dedicated video stream input/output engine (FIFO). The system uses 4×256KB on-chip memory nodes: 2 memories are reserved for data (L2D1 and L2D2) and 2 for instructions (L2Is1 and L2Is1). The communication infrastructure is based on Arteris NoC library, providing necessary components: network interfaces, routers and links. Two separate NoCs have been implemented: 1) one NoC for the datapath (using a regular, fully connected mesh of 2×2 routers) and 2) one NoC for instructions (using only 1 router with 6 input and 2 output ports, allowing simultaneous access to the instruction memories for 2 CGA cores).

Application - AVC/H.264 encoder is a good example of computationally intensive application requiring MPSoCs, especially when targeting high-resolution video streams (HDTV) and high frame rates (30 fps). Having in mind different functional blocks, the whole application can be mapped on the MPSoC platform using following implementation scenarios: a) *Data split*: all processing cores execute the same instruction flow on 6 different video substreams derived from the same input, b) *Functional split*: different functionalities are executed by different cores, everything being organised in a pipe-line fashion and c) *Hybrid split*: a combined version of the previous two: basically the most computationally intensive kernel, Motion Estimation, is implemented using *Data split*, while the remaining functionality is distributed as in the functional split. Based on the C implementation of the encoder, the information about the bandwidths between different functional blocks can be captured.

Design space - The design space is formed by the combination of three different application scenarios (*Data Split*, *Functional Split*, *Hybrid Split*), three image resolutions (CIF, 4CIF, HDTV) and ten 3D stacked architecture variants (on three layers, L1, L2, L3). These are the following (components that are not mentioned are assumed to be floorplanned on layer1):

v1 L2I1, L2I2 on layer2

v2 L2I1 on layer2 ; L2I2 on layer3

v3 L2I1, L2I2, L2D1, L2D2, EMIF on layer2

v4 A1, A2, A3, FIFO on layer2 ; L2I1, L2I2 on layer3

v5 L2I1, L2D1 on layer2 ; L2I2, L2D2 on layer3

v6 L2I1, A1, A2, A3 on layer2 ; L2I2, A4, A5, A6 on layer3

v7 A1, A4, L2D1 on layer2 ; A3, A6, L2D2 on layer3

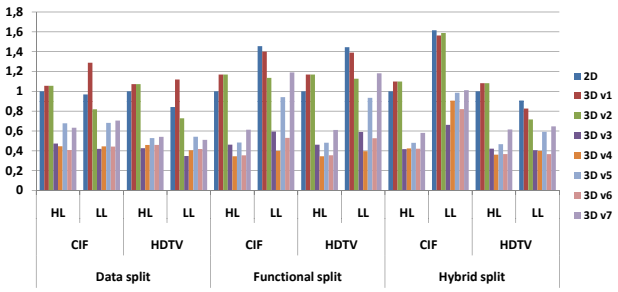

Figure 5: Comparison of the high-level estimated and final normalised NoC link power dissipation for different application mapping scenarios, resolutions, 2D and different 3D-SIC

The components on the platform have been placed such to minimize the area on the layers.

Modeling using Nessie - In this paper, we focus especially on the power dissipation estimation of the wires. For a given resolution and bandwidth, it depends on the wire length. In the simulation file, we thus have defined different wires primitives (transmitting blocks) depending on the architectural variants. We have extracted wires value approximatively based on the figure 4 and typical 3D designs (of course, these approximations will explain some of the differences in the results of section 4). For each functionality scenario and resolution, a Petri Net has been built (9 structures), where primitives vary respectively with tasks and data token size (bus traffics). An example of petri net is shown in the figure 6 for the data split, where we clearly see the possible parallelization of the tasks. The simulation is done in one shot by Nessie which performs the mapping for each functionality/architecture couple and stores all the results in a unique file (*nessieCriteriaResults.xml*). The tool performs the routing by choosing the shorter route between two computational nodes and calculates the corresponding power dissipation dynamically via Yeti, based on user-defined parameters (NoC frequency, capacitance, wire length, traffics, average toggle probability of a wire,

supply voltage).

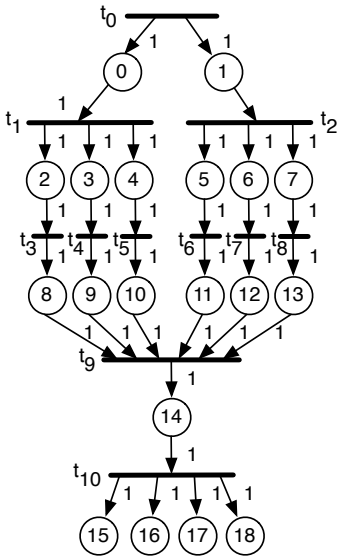

Figure 6: Data Flow Graph of the data split scenario.

Low level design - To feed the NoC simulator, we had to extract the inter-layers links, hence those that require to go through TSVs. Based on this information we had to construct one netlist per 3D-Stack in order to do the floorplanning and global routing. For the physical prototyping part, we have assumed TSMC 90nm GHP technology library. Floorplan and global routing of an entire 3D system is performed sequentially, layer per layer. Consider MPSoC platform components placed in the periphery of the die and the space in the middle occupied by NoC (fig. 4) suits very well for 3D systems. The size of the different layers is usually not the same, but this strategy allows the tools to be able to align the TSV blocks in such a way that the resulting die stacks are mechanically stable. The die with the largest size was floorplanned first and the TSV blocks were always placed somewhere in the middle of the die. Then the relative positions of all the relevant TSV blocks were propagated to the other layer and the system components were placed around the pre-placed TSV blocks. This strategy has resulted in reasonable die sizes without compromising the wire-length reduction that is promised by 3D integration.

We have grouped together the TSVs of different links that end up in the same NoC switches. For instance, in variant v3 two data memories and the EMIF are on the second layer and are connected to the same router that is assigned to the first layer. The links that connect these components to the router all pass through one block that includes 96 TSVs (3 times 32). This allows the TSV blocks to grow in size and leads to fewer very small blocks without a noticeable increase in the total wire-length of the horizontal wiring. As a result, it is easier for the floor-planning engine to come up with a more efficient result. In the case of the other links that need to use TSVs, we have assigned separate blocks of 32 TSVs per link in order to keep the horizontal wire-length under control.

4 Results and discussion

To demonstrate the usefulness of the high-level exploration framework described in Section 2, we will present some of the results that have been generated. Figure 5 illustrates the power dissipation of the NoC links estimated at two different abstraction levels. The HL label in the figure refers to the power estimated by *Nessie*, while the LL label refers to the power estimated after physical prototyping. This Figure 5 shows that the power trends of both estimations are very close for a variety of very different design options ranging from application mapping and input resolution, to the technology for the system integration defined in the previous section. Note that all values in the figure are normalised to the power dissipation of the 2D implementation for each mapping scenario and resolution combination.

It is clear that the magnitude of power dissipation estimated by Nessie can differ and this is captured in Figure 7 showing the differences between the corresponding bars in Figure 5. It gives us an estimation of the high-level model accuracy for the case study. In worst case scenario, the high-level exploration tool underestimates the NoC link power dissipation by about 60%. However, the relative trend for each solution being comparable to the low-level simulations, it shows the ability of the high-level tool to predict design choices for the lower tools and limit thus the upcoming number of iterations.

Concluding we can see that even though the application scenarios and the input resolution are changing, leading also to significantly different traffics on the NoC, the estimations of the high-level exploration are reliable enough to guide a process of exploring huge search space of many alternative options and narrowing down the promising ones. Moreover the high-level tool, Nessie, produces all results in a matter of minutes. Creating even a physical prototype, let alone a detailed layout, can take up to hours depending on the complexity of the architecture and the netlist. The high-level exploration framework presented in this paper can clearly offer significant speed-up in the exploration of large multi-parameter search spaces and pinpoint solutions that are globally promising.

Another conclusion that can be made is that 3D integration offers significant opportunities for reducing the communication power in large MPSoC architectures. The figure 5 above illustrates that independently of the mapping scenario or the resolution of the video sequence, power gains of more than a factor 2 can be consistently expected on the NoC links. This is mainly due to the increased locality between architecture components that a 3D floorplan can offer compared to a planar floorplan.

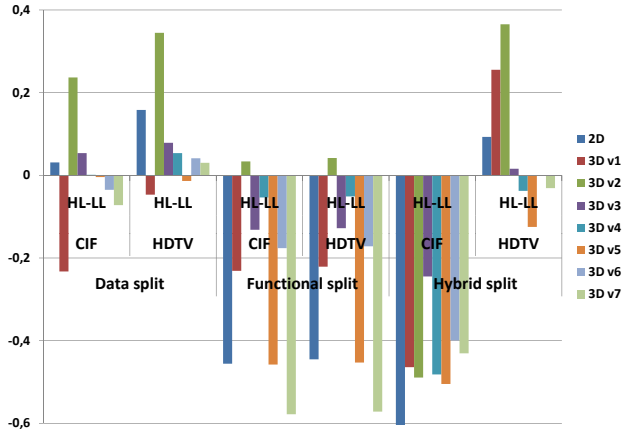

Figure 7: Evaluation of differences between the high-level power estimations and low-level validation results.

5 Conclusions

This paper presents a design methodology to help the exploration of huge design spaces like those industrial face when designing embedded SoC. In particular, we have applied this methodology on a typical MPSoC platform and an existing practical tool chain. Via this case study, we have shown the ability to model a complex system and to integrate our tool, Nessie, in an existing design flow. The resulting framework combines a high-level exploration phase, which quickly and reliably identifies the promising solutions, and a more conventional simulation and physical prototyping phase, which serve to find the optimal solution among the pre-selected ones. Such approach enables exploration of fundamental design options at the level of system architecture definition, thus focusing the effort of the entire design of the system down the path of the globally optimal solution.
Further work could include a global estimation of the energy consumption (including the computational node consumption) and the estimation of other criteria, like the total execution time for each platform. The related effort is linked to the informations extraction (modelisation effort) in order to include costs models (for the criteria estimation) into Nessie.

References

[1] Cofluent Studio, 2007. http://www.cofluentdesign.com/.

[2] F. Balarin, Y. Watanabe, H. Hsieh, L. Lavagno, C. Passerone, and A. Sangiovanni-Vincentelli. Metropolis: An Integrated Electronic System Design Environment. *IEEE Computer Society*, 36(4):45–52, April 2003.

[3] P. H. Chou, R. B. Ortega, and G. Borriello. The Chinook hardware/software co-synthesis system. In *ISSS '95: Proceedings of the 8th international symposium on System synthesis*, pages 22–27, New York, NY, USA, 1995. ACM Press.

[4] T. Kangas, P. Kukkala, H. Orsila, E. Salminen, M. Hännikäinen, T. D. Hämäläinen, J. Riihimäki, and K. Kuusilinna. UML-based multiprocessor SoC design framework. *Trans. on Embedded Computing Sys.*, 5(2):281–320, 2006.

[5] G. H. Loh, Y. Xie, and B. Black. Processor Design in 3D Die-Stacking Technologies. *Micro, IEEE*, 27(3):31–48, May 2007.

[6] R. Marculescu, U. Y. Ogras, and N. H. Zamora. Computation and communication refinement for multiprocessor SoC design: A system-level perspective. pages 564–592, 2004.

[7] B. Mei, S. Vernalde, D. Verkest, and H. D. M. R. Lauwereins. ADRES:An Architecture with Tightly Coupled VLIW Processor and Coarse-Grained Reconfigurable Matrix. In *FPL*, pages 61–70, 2003.

[8] D. Milojevic, L. Montperrus, and D. Verkest. Power Dissipation of the Network-on-Chip in Multi-Processor System-on-Chip Dedicated for Video Coding Applications. *Journal of Signal Processing Systems*, 57(2):139–156, Nov 2009.

[9] Y. Moullec, J.-P. Diguet, N. Ben Amor, T. Gourdeaux, and J.-L. Philippe. Algorithmic-level Specification and Characterization of Embedded Multimedia Applications with Design Trotter. *J. VLSI Signal Process. Syst.*, 42(2):185–208, 2006.

[10] A. Pimentel. The Artemis Workbench for System-level Performance Evaluation of Embedded System Architectures at Multiple Abstraction Levels. *International Journal of Embedded Systems*, 1(7), 2005.

[11] K. Siozios, A. Papanikolaou, and D. Soudris. A method and tool for early design/technology search-space exploration for 3D ICs. *VLSI-SoC08*, 2008.

[12] e. a. Swinnen B. *Wafer Level 3-D ICs Process Technology*. Springer, 2009.

[13] K. Takahashi and M. Sekiguchi. Through Silicon Via and 3-D Wafer/Chip Stacking Technology. *VLSI Circuits, 2006. Digest of Technical Papers. 2006 Symposium on*, pages 89–92, 2006.

[14] A. Vander Biest, A. Richard, D. Milojevic, and F. Robert. A framework introducing model reversibility in SoC design space exploration. *LNCS*, 4599:211–221, 2007.

[15] A. Vander Biest, A. Richard, D. Milojevic, and F. Robert. A multi-objective and hierarchical exploration tool for SoC performance estimation. *LNCS*, 5114:85–95, 2008.