

Cost Modeling and Cycle-Accurate Co-Simulation of Heterogeneous Multiprocessor Systems

Sven van Haastregt, Eyal Halm and Bart Kienhuis
LIACS, Leiden University
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
{svhaastr,ehalm,kienhuis}@liacs.nl

Abstract—In this paper, we present a method to analyze different implementations of stream-based applications on heterogeneous multiprocessor systems. We take both resource usage and performance constraints into account. For the first aspect we use an empirical cost model. For the second aspect we build a network of cycle-accurate processor simulators. The simulation and resource cost estimation have been integrated in an existing framework, allowing one to generate fast exploration simulations, cycle-accurate simulations and FPGA implementations from a single system level specification. We show that with our methodology cycle-accurate performance numbers of candidate systems can be obtained. In our experiments with the QR and MJPEG applications, we found that the error of our resource cost model is below two percent.

I. INTRODUCTION

The computing power demands of embedded systems are increasing continuously. To meet these demands, multiprocessor systems have become attractive solutions as only they can deliver the ever increasing throughput requirements for stream-based applications. Unfortunately, designing such systems is not straightforward, as it is not trivial to find the best tradeoff between resource usage and performance. Therefore, multiple candidate systems have to be evaluated before a designer finds a design which satisfies the set of performance and cost constraints, a process known as *Design Space Exploration (DSE)*. In this paper, we focus on design point evaluation using cycle-accurate simulations and cost modeling.

To assess the quality of a design point, an evaluation is done which provides metrics such as throughput and resource cost. Based on these metrics, a designer makes decisions about the final implementation. Obtaining reliable and correct metrics as fast as possible is crucial in DSE to obtain a design satisfying performance and resource cost constraints. In Fig. 1, we show that evaluation of a system can be done at different levels, making different tradeoffs between speed and accuracy of the evaluation. The fast exploration methods in the upper left allow one to evaluate many different systems in a short time, but such methods cannot provide the designer with guarantees on the achievable throughput of the final system. Shown at the bottom right of the figure, a prototype implementation for an FPGA can be synthesized to obtain fully accurate performance and resource utilization numbers. However, such an implementation requires a fully functional implementation of *all* system components, which are typically not yet available during a hardware-software co-design process. Moreover, for

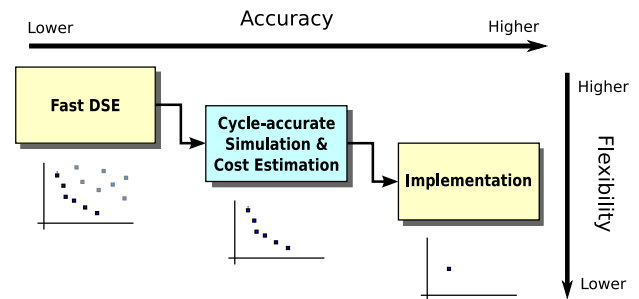


Fig. 1. Different levels of design point evaluation.

complex designs the time taken by synthesis tools can be a limiting factor if many designs need to be evaluated.

Therefore, we present another evaluation method between the fast exploration and implementation methods, namely a cycle-accurate simulation environment together with a method to estimate resource cost. This is depicted by the middle part of Fig. 1. It allows simulation of the most promising designs found by the fast exploration methods, e.g., those on the Pareto front, providing the designer with reliable metrics. This should reduce the time needed to find a suitable design.

This paper is organized as follows. In Section II we discuss related work and in Section III we outline our solution approach. In Section IV we present our resource cost model and in Section V we present our cycle-accurate simulation environment. In Section VI we show results of experiments with our cost model and simulation environment. Finally, we conclude in Section VII.

II. RELATED WORK

During the years, a lot of research in the field of MPSoC design space exploration and simulation has been conducted. The SoClib project [1] uses SystemC to model multiprocessor systems. However, unlike ours, their MicroBlaze simulation is not cycle-accurate. The SystemCoDesigner ESL tool [2] offers automated fast DSE and implementation flows from behavioral SystemC models, but no cycle-accurate simulation flow.

Various people have constructed models to estimate FPGA resource costs. In [3], LUT and flipflop usage of SystemC designs is estimated. In [4], a cost model for DFT IP cores is presented, which considers slices, BRAMs and multipliers. We also consider slices and BRAMs, but instead of focussing on

specific IP cores, we estimate cost for multiprocessor systems. The method of [5] requires a netlist and then does extensive analysis to obtain resource cost numbers. Consequently, this method is more expensive compared to the other approaches.

III. SOLUTION APPROACH

The Daedalus tool set [6] aids the designer in multiprocessor system-on-chip (MPSoC) design. As part of Daedalus, the PNGEN tool [7] allows one to partition a static affine nested loop program written in the C language into a *Kahn Process Network (KPN)*. A KPN [8] consists of processes that communicate using FIFO channels. A KPN consisting of three processes and two FIFOs is shown in the upper half of Fig. 2.

The ESPAM tool [9] allows one to implement a KPN on a given platform, that is, a system consisting of different processors. The user specifies how the processes of the KPN are mapped onto the processors. Using the Sesame [10] backend of ESPAM, fast exploration can be performed. Using the FPGA backend, a prototype implementation on an FPGA can be made. Several processor types are supported by ESPAM, including the MicroBlaze software processor and custom hardware accelerators, consisting of an IP core wrapped using the LAURA processor model [11]. Processors are interconnected using Fast Simplex Link (FSL) components, which implement unidirectional FIFO channels. In the lower half of Fig. 2, an example mapping of the KPN is shown: processes ND_0 and ND_2 are mapped onto MicroBlaze processors, and process ND_1 is mapped onto a hardware accelerator.

To realize cycle-accurate simulation and resource cost modeling of KPNs, we have extended ESPAM with a backend that estimates resource cost and produces a SystemC-based co-simulation environment. By extending ESPAM with a cycle-accurate simulation backend, we realize a flow to evaluate a multiprocessor system at the three levels shown in Fig. 1. The Sesame backend can be used for fast DSE, the SystemC backend for cycle-accurate simulation, and the XPS backend for obtaining specific FPGA implementations. This allows a designer to perform fast exploration, accurately analyze the most promising designs, and finally generate an FPGA implementation from a single system level specification. In the next sections, we describe our resource cost model and simulation framework in more detail.

IV. RESOURCE COST MODEL

An FPGA offers different types of resources that are used for implementation of a design. For example, the basic primitive for logic synthesis in Xilinx FPGAs is the slice, which

contains a set of *LookUp Tables (LUTs)* and flipflops. For limited amounts of on-chip storage, block memories (BRAMs) are available. Depending on the device, other resources such as Digital Signal Processing (DSP) blocks can be included in an FPGA. However, for our cost model, we focus on the two most generic resources, namely slices and BRAMs.

The designs we consider all exhibit a common structure. They consist of processor components, which are interconnected using FIFO components. Each component type has a different resource utilization “footprint”, which depends on its specific configuration parameters, e.g., the number of FSL ports for a MicroBlaze processor. To obtain a realistic cost model, we need to assess for each component type the influence of the different configuration parameters on resource utilization. To calibrate a new model for a specific architecture, we use a fixed suite of component configurations.

A. Slice Utilization

We express the total amount of slices σ utilized by a system consisting of M MicroBlazes and H hardware accelerators, interconnected using F FSLs, as follows:

$$\sigma = \sum_{i \in M} \sigma_{\mu}(i) + \sum_{i \in H} \sigma_h(i) + \sum_{i \in F} \sigma_f(i) \quad (1)$$

Equation 1 can be extended with additional terms to take other component types into account as well. We model slice utilization statistics reported after synthesis of the individual components. In general, these numbers differ only little from the actual cost reported after full implementation of the entire system, provided no unused logic is introduced in the design.

To illustrate the concept we take the $\sigma_{\mu}(i)$ function as an example. This function estimates the amount of slices needed to implement a MicroBlaze v7.00a processor with a 3-stage pipeline on the Virtex-II Pro device family. In our networks, an important MicroBlaze configuration parameter is the number of available FSL ports p . Therefore, we express the cost $\sigma_{\mu}(i)$ for MicroBlaze i of the network as a function of p_i :

$$\sigma_{\mu}(i) = \begin{cases} 556 & \text{if } p_i = 0 \\ 570 + 10 \cdot 2^{\lceil \log_2(p_i) \rceil} + a_i & \text{if } 1 \leq p_i \leq 16 \end{cases} \quad (2)$$

For $1 \leq p_i \leq 16$, we assume a fixed base cost of 570 slices. Next to this, for each FSL port connected, an additional cost is added. Since the MicroBlaze is highly configurable, e.g., the presence of a hardware multiplier or FPU can be controlled by the user, we also allow such additional costs to be added as a factor a . Each MicroBlaze also requires instantiation of other components, like memory controllers, but the costs of those are typically small and (almost) constant. Hence we do not model these separately, but amortize them in the base cost of 570 slices in Equation 2.

Estimating the cost of a hardware accelerator requires a more complex function, as more parameters are involved. Still, creating a cost function is feasible, as each hardware accelerator is again composed of parameterized components, namely a read, execute, write and controller unit [11]. For the read and write units, important parameters are the number of

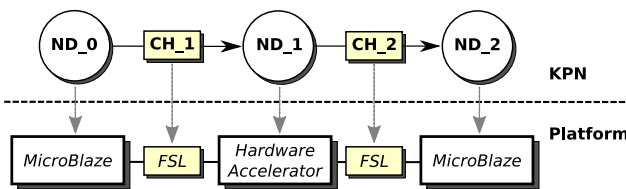


Fig. 2. A KPN mapped onto a platform.

incoming and outgoing FSL ports, respectively. The amount and complexity of control logic has a significant impact on the slice utilization. We estimate the cost of the control logic by taking into account the iteration domain dimension and the amount of terms in the guard and loop bound expressions. The cost of the execute unit greatly depends on the amount of slices utilized by the wrapped IP core. However, we do not consider analyzing the IP core itself in this work, so this number should be provided by the user.

For the FSL components in our networks, the buffer size is the most important parameter. An FSL component can be implemented in three different ways, namely using shift registers, LUT-based memory and BRAM-based memory. Since slice utilization differs with each implementation style, we also consider the implementation style in our FSL cost model.

B. BRAM Utilization

In the same way as slice utilization, we express BRAM utilization β of a whole system as a summation of the BRAM utilization of its different components:

$$\beta = \sum_{i \in M} \beta_{\mu}(i) + \sum_{i \in H} \beta_h(i) + \sum_{i \in F} \beta_f(i) \quad (3)$$

BRAM utilization of a MicroBlaze processor is directly related to the sizes of its program and data memories, which are specified in the platform description that is given to ESPAM. Because of the straightforward relationship between total memory size and number of instantiated BRAMs and because synthesis tools generally do not optimize or merge instantiated BRAMs, the BRAM estimate is fully accurate.

For hardware accelerators, the IP core wrapper itself does not utilize any BRAMs. Therefore, $\beta_h(i)$ only depends on the amount of BRAMs utilized by the IP core. Like the slice cost of a hardware accelerator, we do not analyze BRAM utilization of the wrapped IP core itself.

The BRAM utilization of an FSL component is also calculated accurately. Obviously, only a BRAM implementation of the FSL component leads to BRAM utilization. For those implementations, BRAM utilization is directly related to the FSL buffer size.

V. SIMULATION MODEL

To obtain a multiprocessor simulation environment from a system level specification, we have developed a backend to ESPAM. This backend generates C++ code for a SystemC scheduler and C++ code that has to be run on each MicroBlaze processor. The MicroBlaze processor itself is simulated using the cycle-accurate MicroBlaze ISS provided by Xilinx. However, this ISS was not designed to operate as a multiprocessor simulator, therefore allowing multiple instances to interact and run in parallel is not trivial. The simulator code was not reentrant, preventing straightforward integration using e.g. SystemC threads.

In Fig. 3, we show an implementation of the KPN shown in Fig. 2 using our multiprocessor simulation model. The SystemC top level module implements a controller module

which drives each MicroBlaze ISS instance. This controller module communicates with the simulator instance running as a separate process in the operating system. A program running on a MicroBlaze simulator instance communicates data to the other processors of the network via its FSL ports, using the `get` and `put` instructions. Since the original MicroBlaze ISS does not implement these instructions, we have extended the ISS to send and receive data to and from the other simulated processors in the network. When a blocking write or read occurs, the simulator is stalled on these instructions, such that we simulate stall cycles exactly as they would occur in a final implementation. Furthermore, we have modified the ISS in such a way that before each instruction is executed, it waits for a clock signal from the corresponding SystemC controller module. This way, all simulated MicroBlaze processors advance at the same pace, resulting in a cycle-accurate simulation of the system.

In our multiprocessor simulation environment it is also possible to simulate a heterogeneous system. This means we can integrate hardware accelerators into the simulation. Instead of attaching a MicroBlaze ISS and its corresponding SystemC controller module to the top level module, we attach a cycle-accurate RTL simulation model of a hardware accelerator.

VI. EXPERIMENTS & RESULTS

We now show results for the QR decomposition and Motion JPEG applications in terms of cycle counts and resource costs. In this section, we assume that the KPN for QR consists of five processes. For the MJPEG application, the DCT and Quantization processes are unrolled by a factor of eight, resulting in a KPN consisting of twenty processes.

To assess the accuracy of our cost model, we have taken different applications and mappings and synthesized these into a netlist for implementation on an FPGA. The synthesis tool (XST) reports resource utilization for the different components of the system, which we compare with the resource utilization numbers predicted by our cost model. The results of this comparison are given in Table I. For example, consider design QR-3 μ B1H, in which the five processes of the QR application are mapped onto a platform consisting of three MicroBlazes

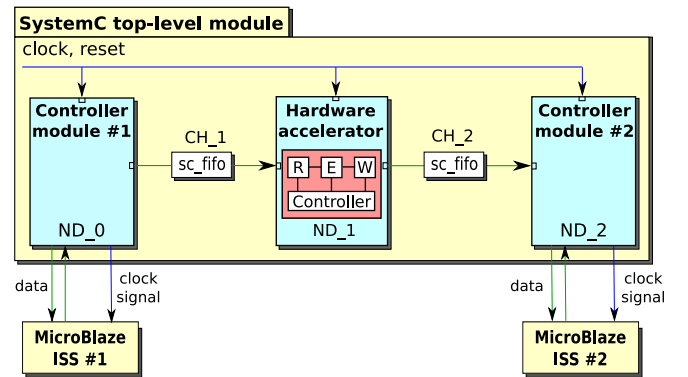


Fig. 3. The multiprocessor simulation model.

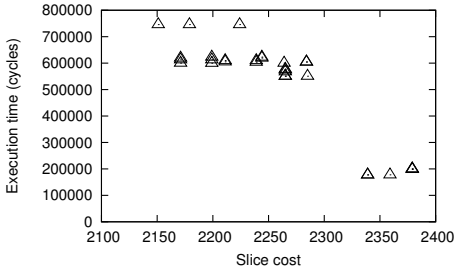


Fig. 4. Different mappings of QR.

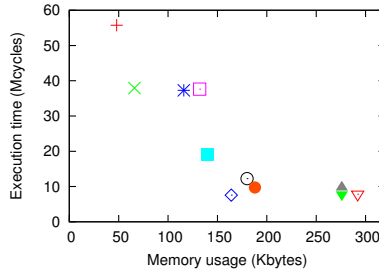


Fig. 5. MJPEG and memory usage.

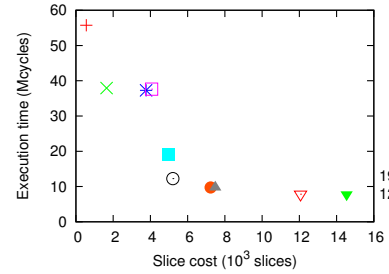


Fig. 6. MJPEG and slice cost.

(3 μ B) and one hardware accelerator (1H). As the number of processes is one higher than the number of processors, this means that one of the processors gets two processes assigned. Our model predicts a slice cost σ of 2477, while the actual cost reported by the synthesis tool is 2503 slices, meaning our model underestimates the real cost by 1.05%. We see that for the configurations in Table I, the error in slice utilization estimation is well below 2%. We believe this error is acceptable, since the estimates can be obtained very quickly, without having to wait for a time-consuming synthesis. We see that the error in BRAM utilization is zero; this confirms our expectations in Section IV. The results for QR-3 μ B1H and QR-3 μ B2H include the IP core wrapper costs, but not the additional costs of the wrapped IP core itself.

We have performed experiments with the QR and MJPEG applications on different platforms and using different mappings. In Fig. 4, we show the results of implementing QR onto a platform consisting of one hardware accelerator and three MicroBlaze processors. From the 30 different mappings that are plotted on the graph, one can see that the specific mapping choice can heavily influence the performance and to a lesser extent the cost of the design.

In Fig. 5 and 6, we show the results obtained by applying our cost and simulation model to the MJPEG algorithm, for memory and slice costs, respectively. We mapped the application onto various platforms, ranging from a single-MicroBlaze system (1 μ B) to a system consisting of nineteen MicroBlazes and a DCT hardware accelerator (19 μ B1H). We assume a DCT hardware accelerator IP core costs 2823 slices [12], which results in a cost estimate of over 30000 slices for the 12 μ B8H design. This is much higher than the cost of any other design, while the 2.6% increase in throughput compared to 19 μ B1H is only very small. Due to its high slice cost, the 12 μ B8H

design lies beyond the right edge of Fig. 6, whereas it seems to be a very attractive solution according to Fig. 5. This shows the importance of considering multiple cost aspects, like slice and memory usage.

VII. CONCLUSIONS

In this paper, we have presented an FPGA resource cost model and a cycle-accurate simulation environment for stream-based multiprocessor systems. The need for this was driven by the observation that an evaluation method was missing between fast explorations on one hand and prototype implementations on the other hand. Using experiments we have shown that our resource cost model is accurate within two percent. Therefore, it successfully fills the gap between the two already existing evaluation methods.

REFERENCES

- [1] SoClib, “The SoClib Project,” <https://www.soclib.fr/trac/dev/wiki>, last accessed: 2009-08-13.
- [2] J. Keinert et al., “SystemCoDesigner – An Automatic ESL Synthesis Approach by Design Space Exploration and Behavioral Synthesis for Streaming Applications,” *ACM Transactions on Design Automation of Electronic Systems*, vol. 14, no. 1, 2009.
- [3] C. Brandolese, W. Fornaciari, and F. Salice, “An Area Estimation Methodology for FPGA Based Designs at SystemC-Level,” in *Proc. of the Design Automation Conference (DAC’04)*, 2004, pp. 129–132.
- [4] P. A. Milder, M. Ahmad, J. C. Hoe, and M. Püschel, “Fast and Accurate Resource Estimation of Automatically Generated Custom DFT IP cores,” in *Field Programmable Gate Arrays (FPGA’06)*, 2006, pp. 211–220.
- [5] M. Xu and F. Kurdahi, “Area and Timing Estimation for Lookup Table Based FPGAs,” in *Proc. of the Eur. Conference on Design and Test (EDTC’96)*, 1996, p. 151.
- [6] H. Nikolov et al., “Daedalus: Toward Composable Multimedia MP-SoC Design,” in *Proc. of the Design Automation Conference (DAC’08)*, June 2008.
- [7] S. Verdoolaege, H. Nikolov, and T. Stefanov, “PN: a Tool for Improved Derivation of Process Networks,” *EURASIP Journal on Embedded Systems*, 2007.
- [8] G. Kahn, “The Semantics of a Simple Language for Parallel Programming,” in *Proc. of the IFIP Congress 74*. North-Holland Publishing Co., 1974.
- [9] H. Nikolov, T. Stefanov, and E. Deprettere, “Multi-processor System Design with ESPAM,” in *Proc. of the Conference on HW/SW Codesign and System Synthesis (CODES-ISSS’06)*, October 2006, pp. 211–216.
- [10] A. Pimentel, C. Erbas, and S. Polstra, “A Systematic Approach to Exploring Embedded System Architectures at Multiple Abstraction Levels,” *IEEE Trans. on Computers*, vol. 55, no. 11, February 2006.
- [11] H. Nikolov, T. Stefanov, and E. Deprettere, “Automated Integration of Dedicated Hardwired IP Cores in Heterogeneous MPSoCs Designed with ESPAM,” *EURASIP Journal on Embedded Systems*, 2008.
- [12] A. Tumeo, M. Monchiero, G. Palermo, F. Ferrandi, and D. Sciuto, “A Pipelined Fast 2D-DCT Accelerator for FPGA-based SoCs,” in *Proc. of the IEEE Comp. Soc. Annual Symposium on VLSI*, 2007, pp. 331–336.

Design setup	Estimate		Actual		Error (%)	
	σ	β	σ	β	σ	β
QR-5 μ B	3335	80	3327	80	0.24	0
QR-3 μ B1H	2477	48	2503	48	-1.05	0
QR-3 μ B2H	2676	48	2658	48	0.68	0
MJPEG-1 μ B	556	24	556	24	0	0
MJPEG-5 μ B	3816	62	3768	62	1.33	0
MJPEG-6 μ B	4066	74	4040	74	0.64	0
MJPEG-20 μ B	12026	150	12066	150	-0.33	0

TABLE I

ACCURACY OF RESOURCE COST MODEL FOR DIFFERENT DESIGNS.