Integration of Correct-by-Construction BIP Models into the MetroII Design Space Exploration Flow

Alena Simalatsar[1], Liangpeng Guo[2], Marius Bozga[3], Roberto Passerone[4]
[1]EPFL, Lausanne, Switzerland – [2]University of California, Berkeley, USA –
[3]VERIMAG, Grenoble, France – [4]University of Trento, Italy

***Abstract*—Design correctness and performance are major issues which are usually considered separately, and with different emphasis, by traditional system design flows. In this paper we show that one can meaningfully connect and benefit from the advantages of two design frameworks, with different design goals. We consider BIP for high-level rigorous design and correct-by-construction implementation, and METROII, for low-level platform-based design and performance evaluation.**

I. INTRODUCTION

Several different methodologies based on the concept of *component* and *platform* address system level analysis and design problems [4], [6], [8], [10], [11], in order to achieve the desired functionality within some given constraint. However, most of the present industrial approaches have the drawback of primarily addressing either the platform or the software components, but not both. One way of solving this problem is to take advantage of the respective strengths of complementary analysis tools, by bridging their semantic gap through model transformations and integration.

We present the coupling of two design and analysis frameworks, BIP [3] and METROII [5]. BIP is primarily focused on correct-by-construction *functional* models. Conversely, a key feature of METROII is the ability to specify both the functional and a separate *architectural* model. These are *mapped* together to produce a system model with performance metrics [7], [9]. Our objective is to reuse a verified correct functional model developed in BIP within an architectural model developed in METROII for design space exploration. Our main contribution consists in structure and semantic preserving transformations that replicate the BIP interaction mechanism in METROII, using a combination of schedulers and constraint solvers. The model can be formally verified in BIP. Then, the performance of the instantiated functional model can be evaluated over different architectures and platforms using the flexible METROII mechanism, without the need to change the BIP components.

II. TOOL INTEGRATION

Structurally, BIP [3] and METROII [5] share the same component-based approach to modeling, making a structure preserving transformation relatively simple. However, the two frameworks are operationally significantly different. We take advantage of the ability of BIP to generate C++ code, which can be then imported in METROII by defining wrappers. Therefore, an atomic *behavior* of the BIP model can be preserved by being embedded into a METROII component.

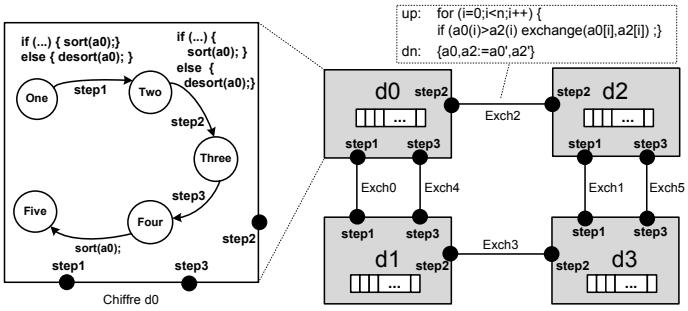

Fig. 1. An example of BIP functional model

The communication mechanism of METROII is different from the one implemented in BIP. Each BIP connector contains an arbitrary computation on the port variables, which is mediated in METROII by a *communication channel*. The introduction of an additional component for communication modeling is similar to the S/R BIP model [3], therefore, the latter is used for the connection with METROII. We have implemented an automatic BIP to METROII ANTLR [1]-based converter which takes as input the BIP functional model and produces a corresponding, structurally and behaviorally equivalent, METROII functional model.

a) Case-study Functional Model: Figure 1 shows a parallelized version of the bitonic sorting algorithm [2] as a composition of components in BIP, previously introduced in [3]. The algorithm splits a sequence to be sorted into several subsequences (*d0*, *d1*, *d2*, and *d3*) handled in parallel by an equal number of components. One of the components, *Chiffre d0*, is presented in details in Figure 1. The top (A) part of Figure 2 shows a fragment of the S/R-BIP model corresponding to the example of Figure 1. The METROII translation of the same example is presented in the middle (B) part of Figure 2. The figure shows the detail of how the interaction between *Chiffre d0* and *Chiffre d2* has been realized with a dedicated communication component. Port *step2* of the model of Figure 1 is replaced with a pair of ports *S_step2* and *step2* for *send* and *receive* communication, while the connector itself is replaced by a scheduler component called *Engine2*. Other ports and connectors of the model components are transformed in a similar way. The states of the components are split each into two states, one that sends and the other that receives the data to/from the scheduler. The *Engine2* scheduler is described as a Petri Net (PN) that manages explicitly the interaction of the BIP connector.

978-1-4673-3052-7/12/$31.00 ©2012 IEEE

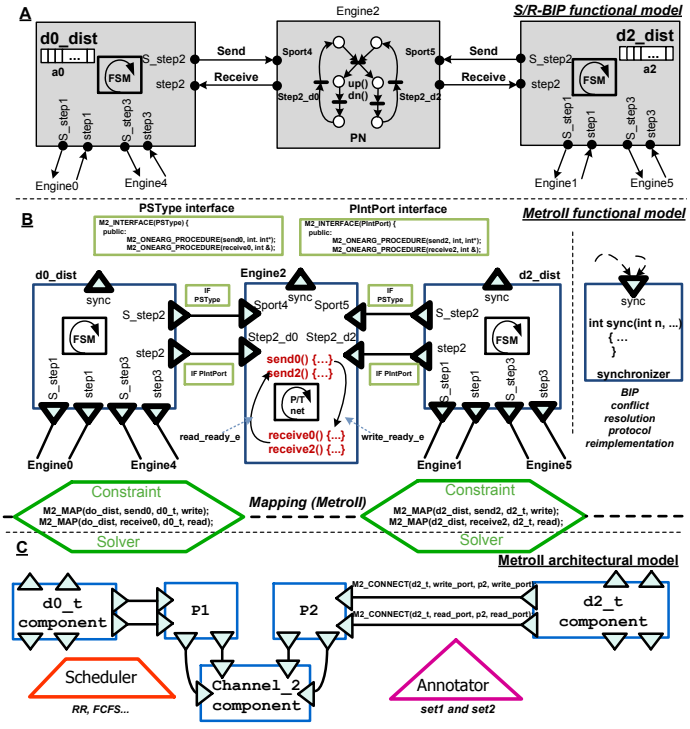

Fig. 2. S/R-BIP and METROII models of the bitonic sorting algorithm

In BIP the interactions between components are concretely realized by using a *synchronization* function call managed by the BIP kernel. We have reimplemented this mechanism in a separate METROII component called *synchronizer* shown in the middle right part (B) of Figure 2.

III. DESIGN SPACE EXPLORATION

Once the BIP functional model is imported in METROII, it is easy to explore the performance of the system under different mappings. Part (C) of Figure 2 shows a simple model of a platform. Each *d(n)_dist* component, with $n \in \{0, 1, 2, 3\}$, is mapped using mapping constraints solvers to a dedicated task component *d(n)_t* of the architectural model. We have evaluated two different mappings. In the first mapping (m1), each of the tasks is executed on a separate processing element which communicate through dedicated channels. In the second mapping (m2), pairs of components (e.g., *d0_dist/d1_dist* and *d2_dist/d3_dist*) share a processing element. Each task of the architectural model is annotated and scheduled using METROII annotators and schedulers. We use a simple characterization of tasks where performance numbers are selected arbitrarily to illustrate the mapping capabilities. However, METROII provides strong support for both off-line and run-time architectural components characterization [7].

We run our experiments with two sets of annotation data (set1 and set2) while the tasks are scheduled either using a pure Round Robin (RR) or a First-come First-served (FCFS) scheduler, or their mix. The information about the mappings, annotation, scheduling policies and simulation results are summarized in Table I. Each column name denotes the type of mapping and scheduling policy. For instance, *m1FCFS*

	m1RR	m1FCFS	m1mix	m2RR	m2FCFS	m2mix
set1	d0(de)sort = 100; write/read memory = 10 d1(de)sort = 200; write/read memory = 20 d2(de)sort = 300; write/read memory = 30 d3(de)sort = 400; write/read memory = 40					
Exe_t	1480	2050	1980	2590	2640	2670
Util %	**p0:** 20.27 **p1:** 40.54 **p2:** 60.81 **p3:** 81.08	14.63 29.27 43.90 58.54	15.15 30.30 45.45 60.60	34.75 81.08	34.09 79.55	33.70 78.65
set2	d0(de)sort = d1(de)sort = d2(de)sort = d3(de)sort = 200 write/read memory = 20					
Exe_t	1040	1040	1040	1460	1460	1460
Util %	57.69	57.69	57.69	82.19	82.19	82.19

TABLE I
EXPERIMENTAL RESULTS

stands for mapping (m1) using the FCFS scheduler. We have studied two main parameters of the system, the total execution time of the distributed algorithm (Exe_t) and the processors utilization (Util) for both sets of annotation data. We calculate the utilization as the percentage of time when the processing element is actually performing some job. The rest of the time it is idle either due to interprocessor communication or due to the data from other components being delayed.

For the annotation data we assume that the execution time for ascending and descending sorting as well as write/read to/from the memory running on the same processing element to be the same. The time for these operations is marked in the table as *d(n)_(de)sort* and *write/read memory* respectively.

REFERENCES

[1] http://www.antlr.org.

[2] K. E. Batcher. Sorting networks and their applications. In *Proceedings of the April 30–May 2, 1968, spring joint computer conference*, AFIPS '68 (Spring), pages 307–314, New York, NY, USA, 1968. ACM.

[3] B. Bonakdarpour, M. Bozga, M. Jaber, J. Quilbeuf, and J. Sifakis. Automated Conflict-free Distributed Implementation of Component-Based Models. In *SIES'10 Proceedings*. IEEE, 2010.

[4] A. Cimatti, L. Palopoli, and Y. Ramadian. Symbolic computation of schedulability regions using parametric timed automata. In *Proceedings of the 2008 Real-Time Systems Symposium*, pages 80–89, Washington, DC, USA, 2008. IEEE Computer Society.

[5] A. Davare, D. Densmore, T. Meyerowitz, A. Pinto, A. Sangiovanni-Vincentelli, G. Yang, H. Zeng, and Q. Zhu. A next-generation design framework for platform-based design. In *Conference on Using Hardware Design and Verification Languages (DVCon)*, February 2007.

[6] D. Densmore, R. Passerone, and A. L. Sangiovanni-Vincentelli. A platform-based taxonomy for ESL design. *IEEE Design & Test of Computers*, 23(5):359–374, May 2006.

[7] D. Densmore, A. Simalatsar, A. Davare, R. Passerone, and A. Sangiovanni-Vincentelli. UMTS MPSoC design evaluation using a system level design framework. In *Proceedings of DATE'09*, pages 478–483, 2009.

[8] K. Keutzer, S. Malik, A. R. Newton, J. Rabaey, and A. Sangiovanni-Vincentelli. System level design: Orthogonolization of concerns and platform-based design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(12), December 2000.

[9] A. Simalatsar, D. Densmore, and R. Passerone. A methodology for architecture exploration and performance analysis using system level design languages and rapid architecture profiling. In *Proceedings of SIES'08*, La Grande Motte, France, June 11–13, 2008.

[10] A. Simalatsar, K. Lampka, Y. Ramadian, S. Perathoner, R. Passerone, and L. Thiele. Enabling parametric feasibility analysis in real-time calculus driven performance evaluation. In *Proceedings of International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES)*, October 2011.

[11] L. Thiele, S. Chakraborty, and M. Naedele. Real-time Calculus for Scheduling Hard Real-time Systems. In *Proceedings of ISCAS 2000*, pages 101–104. IEEE, March 4 2002.