LavA: Model-Driven Development of Configurable MPSoC Hardware Structures for Robots

Matthias Meier and Olaf Spinczyk

Technische Universität Dortmund
Embedded System Software — Computer Science 12
Dortmund, Germany

{matthias2.meier, olaf.spinczyk}@tu-dortmund.de

Abstract: Deploying multicore or multiprocessor hardware for robotics applications is highly beneficial. Parallel hardware structures can be utilized to improve the performance, real-time characteristics, or fault tolerance. Special accelerator components can boost the performance and energy efficiency even more. However, the optimal hardware design is application-specific. This is a dilemma especially for modular general purpose robots, because the application scenario is unknown at design time. Therefore, more and more robots are being equipped with configurable hardware such as FPGAs. In this paper we describe the LavA framework, which facilitates the development of *application-specific* MPSoC hardware structures. Our prototype can interact with Lego Mindstorms NXT sensors and actuators. A DSL is used to describe the hardware structure. Syntactic and semantic checks are performed on the high-level hardware model and a resource model quickly provides an estimate of the required FPGA resources. The hardware synthesis itself is fully automated and requires no special know how. Optionally, the framework can even statically analyze the C/C++ application code. Based on the hardware access patterns found in the code, a suitable hardware description is derived automatically.

1 Introduction

Nowadays, the use of multiprocessor systems is a common practice in many areas, for instance servers or personal computers, but also in the area of embedded systems, multiprocessor systems are an emerging trend. Especially, robotic applications can benefit from the parallel processing of tasks. These applications are almost always depending on real-time requirements and have to handle unexpected events very rapidly, for instance a six-legged robot with environment detection via camera or other sensors. In this case the hardware structures, or more precisely multiprocessor system in our case, have a major impact on the quality of how the robot acts. Does the movement of the six legs looks natural? How quickly and efficiently can the robot spot obstacles or detect objects? Tailored hardware structures can help to achieve the performance goals. The most simple option could be to provide two processors, to separate the environment detection from the motion controll. More sophisticated solutions may result in better performance of the robot, but

erschienen im Tagungsband der INFORMATIK 2011
Lecture Notes in Informatics, Band P192
ISBN 978-3-88579-286-4

end up in a time-consuming process when the hardware structures have to be designed manually.

In this paper we describe our solution to tailor hardware structures to application-specific requirements. Our LavA framework provides a domain specific language to describe heterogeneous multiprocessor systems on a high abstraction level. This high abstraction level enables a straightforward and less error-prone interface to the hardware design. The output of the LavA toolchain is an entirely configured multiprocessor system implemented in VHDL (Very High Speed Integrated Circuit Hardware Description Language). Finally, the synthesized VHDL code can be used to program an FPGA (Field Programmable Gate Array) to test the configured hardware structure with real world applications. In order to evaluate our approach we further need a flexible platform that allows a wide variety of robotic applications. Thus, we have decided to use the sensors and actuators from the Lego Mindstorms NXT [Leg] platform and connect them to the FPGA.

The following sections describe the LavA approach in greater detail. Section 2 will introduce our domain specific language that describes hardware structures. In Section 3 we present details about the resulting hardware structures of the LavA approach and the hardware interface to the Lego Mindstorms NXT components. Our current work on the automated generation of hardware structures by code analysis is presented in Section 4. After discussing related work in Section 5, the final Section will present our ideas for future work and summarize this paper.

2 Domain Specific Language

The development of hardware structures is a time-consuming and complex challenge. Even in the case of hardware description languages such as VHDL or Verilog the time and effort to design, implement, and test the hardware structures is still very high. LavA hides these low-level hardware details by the use of a domain specific language developed with the Eclipse Modeling Framework[1] [Ecl]. Our domain specific language is based on a model-driven approach, which in particular allows us to check the desired hardware structure for correctness and to estimate the consumption of resources in an early stage of the design process. First of all this faciliates the specification of hardware structures, because software developers don't need to have an in-depth knowledge about the design of hardware.

2.1 Hardware Meta Model

The basic structure of the multiprocessor system and the various options of configuration are defined in the LavA meta model (Figure 1). On the top level the MPSoC (multiprocessor system-on-chip) is composed from SoCs (systems-on-chip), connections, shared

[1] formerly known as openArchitectureWare

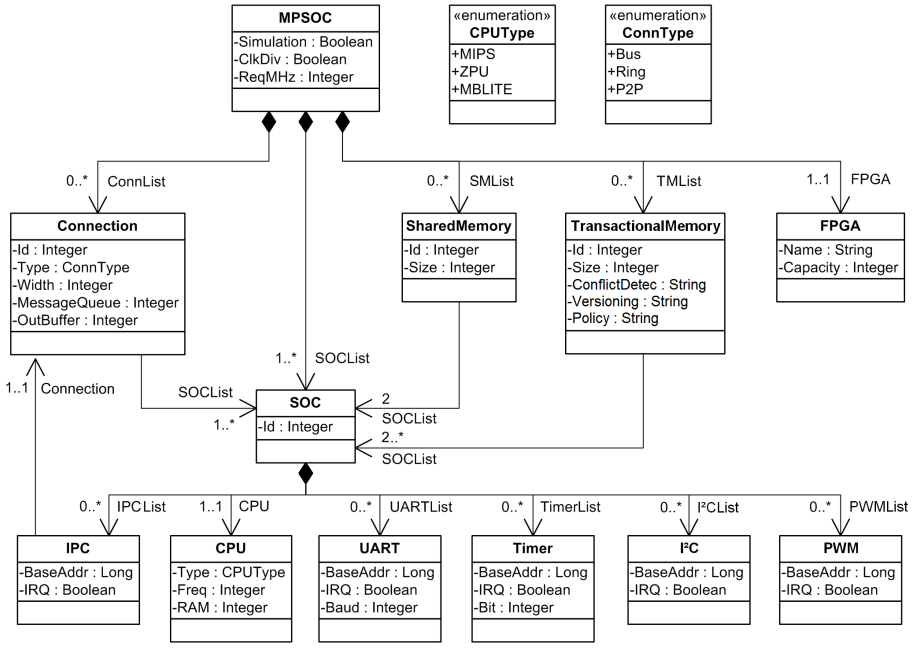

Figure 1: Excerpt from the LavA meta model

memory, and transactional memory [MASS11]. In this context a SoC signifies one processor of the MPSoC with its local peripherals and communication interfaces. The SoCs, or rather the processors, can be associated with connections in order to communicate by a message-based mechanism with any number of processors. LavA offers three connection types for the message-based communication: bus, ring, and point-to-point connection. The shared memory can be used to exchange large amounts of data between exactly two processors, whereas the transactional memory supports any number of processors and lots of parameters for adjustment. Additionally, all hardware components provide a minimal required set of attributes, for instance the size of the message queue for a message-based connection, or the width of a timer register.

By means of the meta model we can easily configure a model representing a specific hardware structure. Therefore, the Eclipse Modeling Framework provides a comfortable graphical user interface to select the desired components and to set their attributes. The configured multiprocessor system, presented by the model, can then be transformed by the text manipulation language *Xpand* of the Eclipse Modeling Framework to the final VHDL code.

2.2 Plausibility Check

Not all checks for correctness of the specified multiprocessor system can directly be derived from LavA's meta model. These checks, however, can be performed using the *Check* language of the Eclipse Modeling Framework, which is provided to specify constraints that have to be fulfilled by the created model. These checks reduce the number of possible errors in the configuration process and identify misconfigurations of the multiprocessor system in an early stage of the design process, reducing the necessity of time consuming hardware synthesis and debugging.

In the example below, our constraints force an error during validation of the model, when no interrupt controller is configured for the associated processor, but interrupts are required for one or more UART[2] devices.

```
context mmMPSoC::SoC if (UARTList.size > 0) ERROR
  "Interrupt-Controller missing in SoC"  + Id :
  (UARTList.forAll(e|e.IRQ==false)) || (InterruptCtrl!=null);
```

Initially, the context of a constraint has to be described. In this case, the constraint only has to be checked for *SoC* components. The *if* statement in line 1 of the code binds the constraint to SoCs with one or more UARTs. At the end of line 1, the action type for this constraint is defined. In this example, failing of this constraint leads to a stop of all processing due to the error statement. It is also possible to only generate a warning during the check process, which does not abort further processing. The second line describes a message which is displayed when an error or a warning occurs. Finally, the condition is defined by an expression. If this expression is evaluated as true, the constraint is fulfilled and the check is successful. The expression in this example can be summarized as follows: *"either all UARTs of this SoC have the option IRQ disabled, or if at least one UART does not, an Interrupt controller has to be instantiated for this SoC"*.

There are many more capabilities to use plausibility checks to ensure a reliable multiprocessor system configuration or to reduce the waste of FPGA resources, such as frequency dependent baud rate check for the UART, constraints for unnecessary components, like a CPU with no connections to other cores, and checking for FPGA resources, e.g. LUTs[3] (Look-Up Tables) or Block RAMs. The Eclipse Modeling Framework also offers the opportunity to enable use case dependent check rules, for instance to limit the number of PWM (pulse-width modulation) devices in the case of a Lego robot application. This could be useful if only a limited set of motor drivers is available.

2.3 Resource Model

For an efficient calculation of the resource consumption on an FPGA we define resource models. However, these models strongly depend on the FPGA vendor and the FPGA

[2] Universal Asynchronous Receiver Transmitter (RS-232)
[3] Small logic units on Xilinx FPGAs that implement boolean functions.

ISBN 978-3-88579-286-4

itself, and are therefore only useable for a small family of similar FPGA types. For LavA we created models for two FPGA families from Xilinx, which are used very frequently—the Spartan-3E and Virtex-5 series. The Spartan-3E family is a low-cost series with 4-input LUTs, whereas the Virtex-5 series offers high-end FPGAs with more resources and a 6-input LUTs design. Furthermore, it should be taken into account that the required resources of the specified hardware design depend on the synthesis software that is used and their settings. For the LavA resource models we use the Xilinx ISE Design Suite 10.1 with default settings.

Due to the high variety of potential hardware configurations it is only possible to measure a selection of multiprocessor systems to identify trends in the resource consumption. For the LavA resource models we focus on the LUTs consumption and the occupied Block RAMs. The required measurement results can easily be extracted from the ISE Design Suite. It provides a hierarchical, itemized list of the resource consumption for each hardware component in the design. The resource estimations for the MPSoC and SoC components are the most difficult of all because they implement virtually no logic, but instead they interconnect all the other components, like processors or peripherals. However, to take these resources into account for our estimation, we only use a simple approximiation for the MPSoC and SoC components.

The functions below exemplarily show the calculations of the LUTs for the IPC[4], CAN[5] and UART peripheral devices. Like the other attributes, used to configure the components of the multiprocessor system, also the cost functions are annotated to each device in the meta model (not shown in Figure 1). The calculation of the total costs for each resource is realized by the *Xtend* language of the Eclipse Modeling Framework. The cost calculation is embedded into the plausibility check and reports an error in the case of a mismatch between the chosen FPGA and the calculated resources. This enables the developer in an early stage of the design process to change the multiprocessor system or to replace the FPGA with a more suitable one.

$$LUT4_{IPC} = (\frac{Connection.Width}{8} \times 90) + 230$$

$$LUT4_{CAN} = 933 + (47 \times Filters) + (90 \times Buffer)$$

$$LUT4_{UART} = \begin{cases} 82 & , Baud \leq 38400 \\ 68 & , Baud > 38400 \end{cases}$$

Table 1 shows the estimated and the measured LUT consumption for three configured multiprocessor systems on a Spartan-3E FPGA. The difference between the estimation and the measurement is quiet low with a maximum of 2.08 percent in our tests.

[4] device for inter-processor communication
[5] controller-area network device (basically used in the automotive sector to connect control units)

System	Prediction	Synthesis Results	Difference
MBLite (Timer 32; UART 19200)	1651	1681	-2.08 %
MIPS (UART 57600; IPC 16) MIPS (CAN with Filter; IPC 16) MIPS (CAN with Filter; IPC 16)	8872	8861	0.12 %
MBLite (CAN with Filter; IPC 32) MIPS (CAN with Filter; IPC 32) ZPU (UART 115200; Timer 64; IPC 32)	9581	9632	-0.53 %

Table 1: Ressource estimation for Spartan-3E (4-input LUTs)

3 LavA Hardware

LavA currently supports three different types of processors (MB-Lite [MBL], Plasma MIPS [Pla], and ZPU [Har]) providing different characteristics in speed of computation, size or frequency. Each processor can be combined with peripherals, like a UART, timer or CAN bus controller, connected via a Wishbone [WB] bus. To communicate with the Lego Mindstorms NXT components we integrated two new peripherals into LavA (Figure 2). The I^2C peripheral is used to read the values from the sensors whereby some Lego sensors don't support I^2C, and instead output the data by means of an analog voltage. To avoid a further peripheral device for the input of analog signals, we extend those analog sensors with a Philips A/D Converter that is equipped with an I^2C interface. The second new peripheral is a PWM device that controls the servo motors via a full bridge driver. Additionally, our circuits, for the connection between the sensors/actuators and the FPGA, provide level shifter to level out different voltages of the FPGA and the Lego components.

4 Automated Hardware Design

Our current work deals with an even more comfortable way to design and to work with application-specific hardware structures. For this purpose we designed a hardware API, an adaptable software representation of our hardware components. This API consists basically of C++ class templates which have to be instantiated as global objects. The template mechanism allows to configure these components at compile time, whereas the global instantiation makes sure that firstly, they are amenable to static analysis and secondly, necessary boot-time initialization is performed. Figure 3 shows the shared memory template that is used to communicate between two processors. A shared memory instance can be configured by the use of three attributes: First, it has to be defined which processor is coupled with the shared memory, then an identification number and the size of the shared memory have to be specified.

To automatically generate an instance of the meta model, that is, a hardware model for a concrete MPSoC, we use an extended version of the parser library PUMA [ULS11] (C, C++, and AspectC++). The parser analyzes the software by searching for global instances

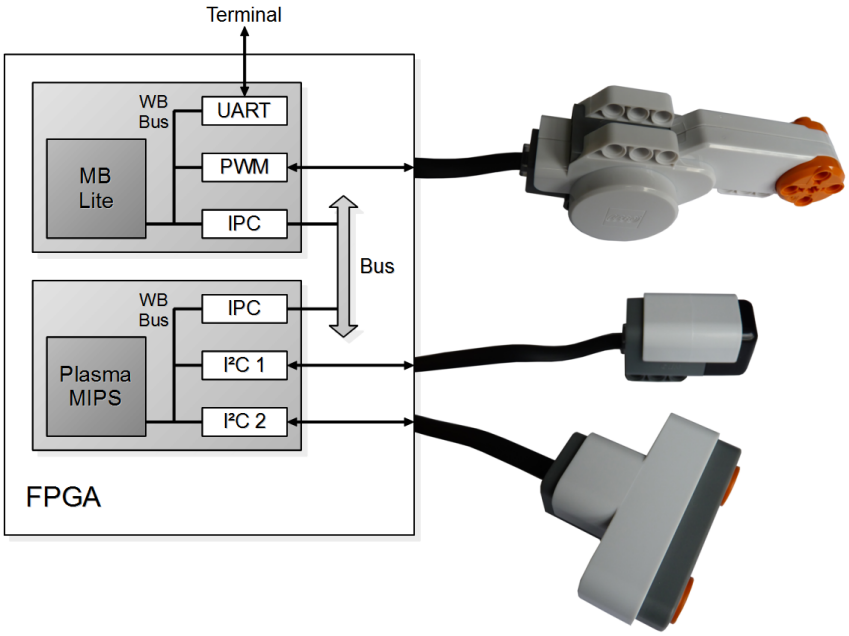

Figure 2: Multiprocessor system with connected Lego Mindstorms NXT sensors and actuators

of hardware API classes. Based on the information collected by the PUMA parser we manipulate the source code, as it may be necessary to allocate memory addresses and IRQ numbers to the devices of each SoC. The result of this step is the manipulated source code and an automatically generated model representing the hardware.

The great advantage of this approach is the opportunity to completely hide the hardware details behind the HW API, and thus it offers the possibilty to generate application-specific hardware structures in a very short period of time.

5 Related Work

A number of robotics projects are already utilizing configurable hardware. Many of them use special-purpose hardware components in order to boost the performance of specific tasks suchs as servo control [SS07], impedance control in a robotic hand [CLW+09], or image processing [GH08]. The paper by Leong et al. [LT05] gives a nice overview about the different promising application areas. Besides this, the authors argue that one barrier for adoption of FPGAs has been *"the large amount of specialized knowledge required to use them"*. The LavA project aims at improving this situation by providing common framework, into which robotics hardware components can be integrated easily.

There is also work that discusses robotics platforms from an architectural point of view.

```
template<typename SoCList = NullType, int Id = 0, int Size = 4096>
struct SharedMemory : public AbstractDevice {

    enum { _ClassType = InternalDevice };
    enum { _Base = 0xD0000000 };
    enum { _Size = 0x10000 };

    template<int I> void instance() {
        setAddress(CPU::_IO_Base + _Base + I * _Size);
    }
};

SharedMemory<Pair<1,2>,0,4096> sharedmem0;
```

Figure 3: Shared Memory Template

For instance, D'Souza et al. argue for application-specific static configuration of hardware structures. Their paper presents "morphing" bus structures [DKV07], which are realized with FPGAs. However, a convenient language that would facilitate the design of such systems has not been presented. Brandt et al. describe a hardware architecture for flexible robot modules [BLC+08]. The modules are equipped with a MicroBlaze soft core and a communication controller. Peripheral interconnects such as UARTs are added in an application-specific manner. Once again proper tool and language support seems to be missing.

In the embedded systems domain Thompson et al. have presented a framework for the automated generation of MPSoCs [TNS+07]. They use Kahn Process Networks [Kah74] as model to describe the application and map the processes to hardware resources based on a design space exploration. However, this framework limits developers to Kahn Process Networks, which are more common for streaming-based multimedia applications. Lukovic and Fiorin describe an automated design flow for a network-on-chip based MPSoC [LF08]. This approach is only suitable for Xilinx FPGAs because they extend the design flow offered by the Xilinx EDK.

With LavA, we provide a flexible solution to design application-specific multiprocessor systems based on a common model-driven software development (MDSD) platform. Furthermore, to the best of our knowledge, LavA is the first framework that combines the automated generation of multiprocessor systems with the domain of robotic applications.

6 Conclusion and Future Work

In this paper we introduced our domain specific language to describe multiprocessor systems on a high abstraction level for robotic applications. Due to the model-driven approach, the LavA framework enables a straightforward and less error-prone interface to

ISBN 978-3-88579-286-4

the hardware design. Therefore, syntactic and semantic checks are performed on the high-level hardware model. Furthermore, LavA provides a resource model to estimate the required FPGA resources in an early stage of the design process. Additionally, we have presented how to generate multiprocessor systems by analyzing the hardware access patterns in the C/C++ application code.

For the near future we plan to integrate our CiAO [LHSP+09] operating system into the LavA framework. CiAO is a highly-configurable, aspect-oriented operating system for embedded systems. The integration of CiAO into the LavA framework will shift developer's view from the HW API to the OS API and thus to a higher abstraction level. Our goal is the co-configuration of the operating system and the hardware structures with a single framework.

Acknowledgment

This work is supported by the German Research Council (DFG) under grant no. SP 968/4-1 and by the German Research Council within the Collaborative Research Center SFB 876 "Providing Information by Resource-Constrained Data Analysis", project "Resource Efficient and Distributed Platforms for Integrative Data Analysis"[6].

References

[BLC+08] D. Brandt, J.C. Larsen, D.J. Christensen, R.F.M. Garcia, D. Shaikh, U.P. Schultz, and K. Stoy. Flexible, FPGA-Based Electronics for Modular Robots. In *Proceedings of the IROS Workshop on Self-Reconfigurable Robots, Systems and Applications*, pages 9–13, 2008.

[CLW+09] Z.P. Chen, N.Y. Lii, K. Wu, H. Liu, Z.X. Xue, M.H. Jin, Y.W. Liu, S.W. Fan, and T. Lan. Flexible FPGA-based controller architecture for five-fingered dexterous robot hand with effective impedance control. In *Robotics and Biomimetics (ROBIO), 2009 IEEE International Conference on*, pages 1063 –1068, dec. 2009.

[DKV07] C. D'Souza, Byung Hwa Kim, and R. Voyles. Morphing Bus: A rapid deployment computing architecture for high performance, resource-constrained robots. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 311 –316, april 2007.

[Ecl] Eclipse Modeling Project. http://www.eclipse.org/modeling/.

[GH08] H. GholamHosseini and Shuying Hu. A High Speed Vision System for Robots Using FPGA Technology. In *Mechatronics and Machine Vision in Practice, 2008. M2VIP 2008. 15th International Conference on*, pages 81 –84, dec. 2008.

[Har] Øyvind Harboe. ZPU: http://opensource.zylin.com/zpu.htm.

[Kah74] Gilles Kahn. The semantics of a simple language for parallel programming. In J. L. Rosenfeld, editor, *Information processing*, pages 471–475, Stockholm, Sweden, Aug 1974. North Holland, Amsterdam.

[6] http://sfb876.tu-dortmund.de/

[Leg] Lego Mindstorms NXT. http://mindstorms.lego.com.

[LF08] Slobodan Lukovic and Leandro Fiorin. An Automated Design Flow for NoC-based MPSoCs on FPGA. In *The 19th IEEE/IFIP Intl. Symposium on Rapid System Prototyping (RSP '08)*, pages 58–64, June 2008.

[LHSP+09] D. Lohmann, W. Hofer, W. Schröder-Preikschat, J. Streicher, and O. Spinczyk. CiAO: An Aspect-Oriented Operating-System Family for Resource-Constrained Embedded Systems. In *Proc. USENIX*, pages 215–228, 2009.

[LT05] P.H.W. Leong and K.H. Tsoi. Field programmable gate array technology for robotics applications. In *Robotics and Biomimetics (ROBIO). 2005 IEEE International Conference on*, pages 295 –298, 0-0 2005.

[MASS11] Matthias Meier, David Austin, Horst Schirmeier, and Olaf Spinczyk. TMPL: A Hardware Transactional Memory Product Line. In *Proceedings of the Workshop on Multiprocessor Systems on (Programmable) Chips (MPSoC 2011)*, Istanbul, Turkey, July 2011. IEEE Computer Society Press. to appear.

[MBL] MB-Lite Microprocessor: http://www.opencores.org/project,mblite.

[Pla] Plasma MIPS: http://www.opencores.org/project,plasma.

[SS07] Xiaoyin Shao and Dong Sun. Development of a New Robot Controller Architecture with FPGA-Based IC Design for Improved High-Speed Performance. *Industrial Informatics, IEEE Transactions on*, 3(4):312 –321, nov. 2007.

[TNS+07] Mark Thompson, Hristo Nikolov, Todor Stefanov, Andy D. Pimentel, Cagkan Erbas, Simon Polstra, and Ed F. Deprettere. A framework for rapid system-level exploration, synthesis, and programming of multimedia MP-SoCs. In *The 5th IEEE/ACM Intl. Conf. on Hardware/software codesign and system synthesis (CODES+ISSS '07)*, pages 9–14, New York, NY, USA, 2007. ACM.

[ULS11] Matthias Urban, Daniel Lohmann, and Olaf Spinczyk. *PUMA: An Aspect-Oriented Code Analysis and Manipulation Framework for C and C++*. No. 6580. Berlin, 2011.

[WB] Wishbone: http://www.opencores.org/opencores,wishbone.

ISBN 978-3-88579-286-4