Abstract

Hardware and software can be co-designed as a single system to obtain high-performance computing solutions that use a minimum of resources such as CPU cycles and memory. Hardware/software co-design is often a complex and time-consuming process. Software functionality is typically developed independently from hardware functionality and functional elements are not reused across similar products. Co-design is particularly important in System on-Chip (SoC) development where an entire system consisting of both software and hardware is integrated on a single chip. Developing high-performance computing solutions in the form of SoCs in a repeatable manner is technically possible, but knowing how to apply software concepts in a hardware context and vice versa is still demanding at best. This paper summarizes the concepts and considerations that outline a case study that aims at combining the software product family paradigm with run-time reprogrammable hardware technology. The goal of the study is to define a formal framework for developing product families on-chip in a repeatable manner.

1. Introduction

The software product family paradigm and run-time reprogrammable hardware technology are brought together in a case study in designing a Real-time Adaptive Signal Processing (RASP) system. RASP is designed as a dynamically reconfigurable product family on-chip and targets hard real-time applications such as monitoring data streams on-the-fly. Performance, stability, configurability and development costs are key issues in the study and rely strongly on run-time variability dependencies. The next sections shortly introduce the concepts relevant to RASP.

1.1. Software Product Families

Software reuse is probably the most promising approach to the cost-effective development and evolution of quality software. An example of reuse is the successful adoption of so-called software product families (or lines) in industry. A software product family typically consists of a product family architecture, a set of components and a set of products. Each product derives its architecture from the product family architecture, instantiates and configures a subset of the product family components and usually contains some product-specific code [1]. All in all, the product family approach aims at maximizing market coverage by diversifying the product range while minimizing development costs through software reuse and repeatable software development solutions.

1.2. Run-time Reprogrammable Hardware

A general purpose microprocessor is a flexible solution for executing software defined functionality, but the performance of a software implementation is limited compared to a dedicated implementation in hardware. Reprogrammable hardware technology in the form of FPGAs (Field Programmable Gate Arrays) has matured over the past years and now allows for reprogramming (parts of) the FPGA at run-time. Run-time reprogrammable hardware implies the flexibility of software and the performance of a hard-wired implementation such as ASICs (Application Specific Integrated Circuits).

1.3. Dynamically Reconfigurable Systems

The term ‘reconfigurable system’ applies to a broad range of systems. It originally referred to systems in which hardware is reorganized to adapt to a changing context, but is now also used for systems that can be reconfigured during any of the development or deployment phases such as design or execution. A system is dynamically reconfigurable if its hardware and/or software configuration is deployment adaptable, most notably at run-time.

An application area of dynamically reconfigurable systems is speeding up computation. As shown in Figure 1, if a task $T$ can be partitioned into two sequential subtasks $T_{seq1}$ and $T_{seq2}$, each subtask can be processed with a run-time configuration that is optimized for a part of the total computation. In addition, if task $T$ can be partitioned into two parallel subtasks $T_{par1}$ and $T_{par2}$, both subtasks can be processed simultaneously due to the intrinsic parallelism available in run-time reprogrammable hardware such as FPGAs. RT in Figure 1 stands for Reconfiguration Time.

1.4. Dynamically Reconfigurable Product Families On-Chip

There is a need to integrate whole systems, including peripherals, on a single chip. The System on-Chip (SoC) con-
cept implements system functionality in the form of chip technology on a single die. SoC development relies heavily on the high levels of integration that have become economically viable due to recent advances in semiconductor manufacturing. The ever increasing gate count - Moore’s law is expected to hold true for at least another decade - allows for more complicated designs on the same die size that go beyond the functionality of core logic. Fully utilizing the large number of transistors in future chips has been identified as an architectural problem almost a decade ago [2]. However, only recently it became clear that so-called billion-gate architectures will be judged by how efficiently they support distributed hardware without placing intractable demands on programmers [3].

Although FPGAs have traditionally been used for prototyping and designing ASICs, the continuously increasing gate density and switching speed have made FPGAs more successful in terms of performance than their microprocessor counter parts [4]. FPGAs with a gate count of 10,000,000 or more are already commercially available and do not only provide enough gates to implement basic SoCs, but the FPGA can also be reconfigured on-the-fly to adapt to run-time specific contexts.

2. RASP Considerations

As said, RASP is a case study in designing a dynamically reconfigurable product family on-chip. The study combines the software product family paradigm with run-time reprogrammable hardware technology and describes the product family architecture of a dynamically reconfigurable system in the form of a SoC. The benefits of a dynamically reconfigurable product family on-chip are the following:

- A single hardware platform supports a family of related products.
- The hardware consists of inexpensive and standardized Commercial-Off-The-Shelf (COTS) components.
- The performance of a hard-wired solution and the flexibility of software.
- The algorithm that is best suited for the task is selected on-the-fly.
- Development is according to the software product family paradigm.

Design problems in externally integrated systems are often solved by adding glue logic, just like design problems in software development are often solved by adding glue code. SoC components often depend strongly on each other and when a problem in a SoC component occurs, large parts of the system have to be at least revisited, if not redesigned. Dependencies between components are a serious problem in mainstream software development, but they have considerably more impact in SoC development due to the high level of integration and the merging of software functionality with hardware functionality. The dependency problem is only compounded when the SoC is dynamically reconfigurable and easily underestimated.

Success in SoC development strongly depends on the engineers’ ability to conceive complex designs that are a combination of software and hardware under strong time-to-market pressure as well as on interconnecting (configuring) SoC components reliably in a plug-and-play fashion [5].

The problems in SoC (re)configuration, using FPGA technology or otherwise, are similar to the problems that are addressed in software product family research. The dependency problem is similar to the variability dependency issues addressed in our earlier work [6]. We have suggested a formalization of so-called variability dependencies in [7]. The formalization is used in [8] to derive a taxonomy and hierarchy of variability dependencies to pinpoint the type and impact of dependencies, respectively. The RASP study uses the software product family paradigm to introduce software engineering approaches to SoC development in an attempt to ease the design and integration of SoC components.

3. Summary and Future Work

It still remains a question if dynamically reconfigurable product families on-chip can meet the demands of hard real-time requirements together with system properties such as safety, security, reliability and usability. One of the main problems is to manage the dependencies between SoC components during both development and deployment, most notably at run-time. Too many dependencies or implicit dependencies is almost a guarantee for less than expected performance, stability, configurability and increases development costs.

The goal of the RASP study is to define a formal framework for developing product families on-chip in a repeatable manner through application of the software product family paradigm. Particular emphasis will be partitioning the system into functional elements and the granularity of these elements in relation to the intrinsic parallelism available in gate arrays such as FPGAs. Early results in the RASP study indicate that the software product family paradigm is useful for managing the complexity associated with the dependencies inherent to configuring a product family on-chip.

References