A Structured System Methodology for FPGA Based System-on-a-Chip Design

Pete Sedcole, Peter Y. K. Cheung, George Constantinides & Wayne Luk
Imperial College, Exhibition Road
London SW7 2BT, UK

The ever increasing quantities of logic resources combined with heterogeneous integrated performance enhancing primitives in high-end FPGAs creates a design complexity challenge that requires new methodologies to address. We present a structured system based design methodology which aims to increase productivity and exploit reconfigurability in large scale FPGAs. The methodology is exemplified by Sonic-on-a-Chip, a video image processing system.

1. Introduction

The continuing compound growth in transistor density due to advances in process technology is not being matched by a corresponding growth of integrated circuit designer productivity. The increase in the cost of application specific IC design is a primary threat to the semiconductor roadmap [4], a problem compounded by increasing mask set costs, long design, fabrication and test cycles, and the inflexibility of the end product. Field Programmable Gate Arrays are regarded as a panacea for these problems, since their regular structure and scalability facilitates silicon design and test. However, escalating FPGA heterogeneity due to the integration of performance enhancing primitives (such as memories and microprocessors) coupled with increasing transistor density results in a design complexity challenge, necessitating the development of new design methodologies to increase productivity.

We present a *structured system* design methodology based on a paradigm of architecture reuse; instead of designing a system in an FPGA from the bottom-up by connecting together blocks of predesigned IP in an ad hoc manner, the system is constructed from the top-down by defining the logical and physical structure (i.e. the architecture) first. Key aspects of the methodology are the use of *modularity*, high levels of *abstraction* and *orthogonalisation of concerns*, such as the separation of communication and computation, thus avoiding system-level timing closure issues. The methodology enables reconfigurability to be exploited for customisation and in-field partial upgrades.

Since the methodology is based on architectural reuse, the choice of architecture is critical; we exemplify the methodology with a single architectural instance *Sonic-on-a-Chip* [3], a video processing system based on Sonic [1]. The architecture comprises several layers. Logical and physical layers describe the system topology and infrastructure, and combined form a platform for application development. Applications built on the platform include hardware modules, operating system layers and algorithm encoding.

2. Sonic-on-a-Chip

Logical layer: The general logical layer for Sonic-on-a-Chip is shown in Figure 1. The operation of this system and its optimisations for video processing has been described previously [3]. However, it should be noted that (a) system-level design is facilitated by inherent modularity, (b) the number and complexity of each modular processing element (PE) may vary, (c) within each PE, a router (fixed in design but programmable) is used to manage communication, separating this from the computation being performed in the fully customisable engine. Furthermore, data are passed into and out of the engine via *stream buffers* which, being composed of RAM primitives may be formed into a number of different configurations (a two-input two-output option is depicted in the diagram).

Physical layer: The physical layer of the architecture defines the implementation of the logical system structure on the FPGA, including the positioning of the modules and the intermodular routing. It is possible to integrate the infrastructure and modules together at design time (during synthesis or as netlists). In our methodology the integration is at run-time, which we term *late integration*. This increases the advantages of modularity and enables systems to be rapidly constructed (with no further hardware design) from pre-implemented modules. Importantly, to achieve late integration of a variable number of modules of different sizes, system-level intermodular routing is fixed and regular, which leads to deterministic timing. In this respect our implementation is similar to DISC [5], although on a larger scale.

Application implementation: Applications for Sonic-on-a-Chip must be expressible in dataflow graph format,

Proceedings of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'04)
0-7695-2230-0/04 $ 20.00 IEEE

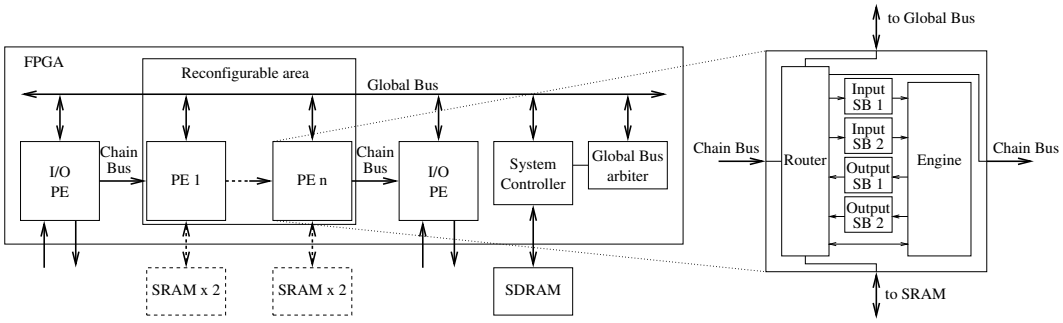

Figure 1. The general logical structure of Sonic-on-a-Chip.

where each node in the graph is mapped to a processing element. To implement an application, control software loads the required module configuration bitstreams into certain locations in the FPGA configuration memory and then programs the PE routers to correctly direct data from module to module. In situations where more than one application runs concurrently, it is usual for a software operating system to manage shared resources. In our methodology, we use a novel mechanism to represent the dataflow graph topology in a format that the operating system can interact with: For each PE in the application, the application software spawns a new software process. These are termed *ghost processes* and do not perform any processing. Connectivity is represented by redirecting inputs and outputs of the ghost processes through IPC FIFOs. A Configuration Manager layer (within the OS) interacts with the ghost processes in order to download bitstreams and perform low-level programming of the hardware.

3. Feasibility studies

Simulations: A functional, synthesizable simulation model of the Sonic-on-a-Chip system has been developed, targeting the Xilinx Virtex II Pro family of FPGAs. For this simulation, bus widths were set to 32 bits and stream buffers constructed from up to 16 Block RAM primitives. To demonstrate the advantages of parallelising stream buffers, a 16-parallel convolution engine was developed which convolves a 3x3 kernel with a 64x64 image. This was able to be clocked at 84% of the pixel sample rate.

Modular implementation: The implementation of a processing element has been investigated. This follows the Xilinx partial reconfigurable design flow [2], although new hard macros had to be developed for constraining signal routing across horizontal module boundaries. A hard macro was also used to constrain the Global Bus routing to specific tristate lines. While the implementation was successful, it was observed that some structures in the FPGA routing fabric (such as longlines) are not well suited to modular design, and the place and route tool (Xilinx PAR F.31) does not currently obey horizontal module boundary directives. This is due to the structure of the Virtex II configuration memory.

4. Conclusion

A new design methodology has been outlined, which aims to improve productivity through the imposition of a structure on FPGA-based SoC designs. Modularity, abstraction and the separation of communication from computation all facilitate design. Late integration enables modules to be developed independently of each other and of system-level timing.

The methodology is exemplified by the Sonic-on-a-Chip video processing architecture. Preliminary investigations indicate that while this architecture is functionally viable, changes are required to FPGA routing, configuration memory, and tools to fully benefit from this design methodology.

Acknowledgements: The authors would like to thank the Commonwealth Scholarship Commission, the New Zealand Vice Chancellors' Committee and Xilinx for their support.

References

[1] S. D. Haynes, J. Stone, P. Y. K. Cheung, and W. Luk. Video image processing with the Sonic architecture. *IEEE Computer*, 33(4):50–57, April 2000.

[2] D. Lim and M. Peattie. Two flows for partial reconfiguration: module based or small bit manipulation. Application Note 290, Xilinx, 2002.

[3] N. P. Sedcole, P. Y. K. Cheung, G. A. Constantinides, and W. Luk. A reconfigurable platform for real-time embedded video image processing. In *Proc. Field–Programmable Logic and Applications*, 2003.

[4] Semiconductor Industry Association. International technology roadmap for semiconductors, 2001.

[5] M. J. Wirthlin and B. L. Hutchings. A dynamic instruction set computer. In *Proc. IEEE Symposium on FPGAs for Custom Computing Machines*, 1995.