Leveraging Reconfigurability in the Design Process

Lesley Shannon

Supervised by Dr. Paul Chow

Department of Electrical and Computer Engineering, University of Toronto

Toronto, Ontario, Canada, M5S 3G4

lesley@eecg.toronto.edu

1. ABSTRACT

We are investigating an on-chip design methodology for embedded computing systems implemented on FPGAs. The objective is to exploit the platform's reconfigurability, such that system development can occur on the final implementation platform. This is comparable to the software development process where applications are typically designed on a workstation that is representative of the platform for the final product, not a simulator that models the processor.

Designing on the target technology is also appealing for FPGA designs. Their potential size and complexity makes simulation a very time consuming process that takes orders of magnitude longer than on-chip execution. An on-chip design methodology better leverages the main advantage of reconfigurability – the user may redesign the system while avoiding non-recurring costs. It also allows the user to quickly obtain real-time information about system performance. Then hardware design begins to resemble software design, where debugging and profiling instrumentation are easily added to the software to obtain runtime information.

Figure 1 illustrates our proposed design flow, where we assume that the initial design is a pure software implementation. Once the software functionality is verified, the design is profiled. If it fails to meet the design specifications, then the system is divided, or partitioned, into hardware and software components. The designer iterates between repartitioning the design and profiling the system until it meets the design specifications.

Our preliminary work towards an on-chip design methodology produced two on-chip profiling tools for analyzing the performance of embedded computing systems. The first is SnoopP, a non-intrusive, real time, Snooping Profiler (SnoopP) for software applications that can obtain clock cycle accurate information on software performance [2]. Figure 2 illustrates how SnoopP is implemented as an independent hardware module that snoops a processor's executing program counter to measure the number of clock cycles a processor spends executing a particular code region. This information can then be used to determine portions of code that should be moved to hardware to improve overall performance.

Once a system is partitioned into hardware and software components, it is important to ensure that the work load is balanced among the different modules. If embedded computing systems are modelled as Systems Integrating Modules with Predefined Physical Links (SIMPPL), then Watching Over Data Streaming On Computing element linKs (WOoDSTOCK) can be used to detect stalls and bottlenecks in a system [1]. The SIMPPL model, shown in Fig-

ure 3, represents processors and dedicated logic as software and hardware Computing Elements (CEs) respectively. The off-chip communication structures (dotted arrows) are determined by off-chip peripherals, however, the on-chip inter-CE communication links (solid arrows) are fixed as asynchronous FIFOs.

Figure 4 illustrates how WOoDSTOCK's system-tailored monitors are associated with each of the CEs in the system. The monitor controller determines the runtime of the profiler and provides an interface for reading the profile data off-chip. By monitoring the status of the FIFOs that provide input data and transmit output data from a CE, WOoDSTOCK is able to detect the three types of bottlenecks shown in Figure 5. In each of these three situations, CE 1 is the potential bottleneck to system performance, where it may not consume data quickly enough in cases (a) and (c) or produce data quickly enough as in case (b).

Currently, we are investigating how to facilitate the reuse of CEs in our system design methodology using the SIMPPL model. The physical integration of new CEs is facilitated by using asynchronous FIFOs for all inter-CE connections, but this does not address the situation where a CE's functionality must be altered to meet the requirements of the new application. Figure 6 illustrates the proposed architecture for a hardware CE. The SIMPPL controller is designed to interface the Hardware IP core, or Processing Element (PE), to the rest of the system. It processes data requests sent from other CEs and its own SIMPPL Control Sequencer (SCS). The SCS contains a local set of instructions used to send and request data needed by the PE. The SCS allows the user to program how a CE interacts with the rest of the system without having to redesign the PE, thus simplifying the adaptation of CEs to different applications.

2. PHD CANDIDATE INFORMATION

Name: Lesley Shannon

Supervisor: Dr Paul Chow

Institution: University of Toronto

Contact Information: Please see above

Publication Forums: FPT, FPGA

Estimated Graduation Date: June 2006

3. REFERENCES

- [1] L. Shannon and P. Chow. Maximizing System Performance: Using Reconfigurability to Monitor System Communications. In *IEEE Int. Conference on Field-Programmable Technology*, pages 231–238, Dec. 2004.
- [2] L. Shannon and P. Chow. Using Reconfigurability to Achieve Real-Time Profiling for Hardware/Software Codesign. In *ACM Int. Symposium on Field-Programmable Gate Arrays*, pages 190–199, Feb. 2004.

Figure 1: The proposed reconfigurable hardware/software design flow.

Figure 2: The SnoopP architecture.

Figure 3: A SIMPPL model of a generic system.

Figure 4: The WOoDSTOCK architecture.

Figure 5: Examples of the different types of bottlenecks detectable by WOoDSTOCK: (a) interior bottleneck, (b) input bottleneck, and (c) output bottleneck.

Figure 6: The internal structure of a hardware CE.