Using Reconfigurability to Achieve Real-Time Profiling for Hardware/Software Codesign

Lesley Shannon and Paul Chow
Dept of Electrical and Computer Engineering University of Toronto, Canada
{lesley, pc}@eecg.toronto.edu

Background

Hardware/Software Codesign
Conceptually, hardware/software codesign means a processor plus dedicated logic.

Decreasing technology sizes allow a complete design to be implemented on a reconfigurable platform.

Question: How should the Hardware/Software Codesign Methodology change to reflect the new design environment?

Software Design Methodology

Start Write Software Module
Debug System
Add Next Software Module
Does System Work?
No Yes
Complete System?
No Yes
Done

This software design methodology allows the programmer to debug on the final implementation platform.

The same paradigm can be applied to hardware/software codesigns implemented on FPGAs.

Embedded System Design Methodology

Start Write Software/Hardware
Profile
Rewrite Software/Hardware
Simulate Functionality
Met Design Specs?
No Yes
Done

To use an on-chip design flow for this methodology, we need to create the infrastructure.

Here we introduce the first tool for this infrastructure: a profiler to assess software performance.

GNU's software profiler: gprof
gprof inserts code into the user's application to enable the counting of function calls.

The function call count is precise, however the execution time profile is not accurate.

gprof samples the Program Counter and increments the execution of the appropriate function by the sample period.

For this to work, the total application runtime must be significantly larger than the sample period.

On-Chip Software Profiler

SnoopP: A Snooping Profiler

Profiles Processors on FPGAs providing:
1) Real Time Information
2) Clock - Cycle Accurate Results
3) Non-Intrusive Interface to the Executing Software
4) Scalable Modular Design

General Architecture

System Bus

SnoopP's Code Segment Clock-Cycle Counters

valid_pc
CPU
PC

Clock-Cycle Counter for Code Segment 1

PC Comparators:
PC=<i>th address AND PC>=<i>o address

Counter: Increment Enable

System Implementation

On-Chip Peripheral Bus (OPB)

Other Peripherals
SnoopP
valid_instr
MicroBlaze

PC, EX

Evaluation Methodology

1) Use gprof to obtain an initial profile on a Sun Blade 1000 Solaris OS version 8.
2) Select code regions for SnoopP to profile based on results.
3) Use SnoopP data to determine where the majority of execution is spent.

Results

SnoopP and gprof profiles can be VERY different.

Dhrystone Profile

Percentage Execution Time

0.0 5.0 10.0 15.0 20.0 25.0 30.0

Profiling Functions

\[ \text{gprof} \quad \text{SnoopP: SW multiply} \quad \text{SnoopP: HW multiply} \]

gprof results were normalized as it attributed 44% of the execution time to gprof overhead.

AES Profile

Percentage Execution Time

0.0 10.0 20.0 30.0 40.0 50.0 60.0 70.0

Profiling Functions

\[ \text{gprof} \quad \text{SnoopP: SW multiply} \]

SnoopP profiled only 26% of the static code size and measured 96% of the execution time.

Conclusions and Future Work

We created a software profiler for an on-chip design methodology.

The experimental results demonstrated that SnoopP was able to obtain more accurate information about the performance of the software.

This information could be integrated into Hardware/Software Codesign tools to provide better partitioning choices.

We are investigating methods to better measure the performance of a system comprising both hardware and software components.

Funding for this research has been provided by an Ontario Graduate Scholarship and the Natural Sciences and Engineering Research Council of Canada.