ASC-Based Acceleration in an FPGA with a Processor Core Using Software-Only Skills

Evgeny Fiksman, Yitzhak Birk

Electrical Engineering Department,
Technion - Israel Institute of Technology,
Haifa, Israel
fiks@technion.ac.il birk@ee.technion.ac.il

Oskar Mencer

Department of Computing,
Imperial College,
London, UK
o.mencer@imperial.ac.uk

Abstract

A Stream Compiler (ASC) generates netlists for hardware (FPGA) accelerators from C-like descriptions, obviating the need for hardware skills. We present a backend adapter that enables integration of such accelerators with a processor core in the same FPGA. Development of hybrid ASC-accelerated applications using software-only skills is thus made possible, as illustrated by a hybrid power-conscious iDCT implementation.

1. Introduction

The custom computing approach is becoming widespread, and recent systems feature tight integration between a microprocessor core and a reconfigurable area. A prominent example is the Xilinx Virtex-II Pro FPGA. Support of the ever growing demand for rapid implementation of such systems requires effective tools for (i) determination of the parts of the application that should be accelerated, (ii) generation of efficient accelerators, and (iii) integration of the accelerator with the processor core. Moreover, such tools should best only require software skills, which are both more abundant and are already required for the software parts of the solution.

In the past, both (ii) and (iii) were extremely time consuming and required hardware design skills. More recently, tools such as A Stream Compiler (ASC) [1] successfully address (ii) while only requiring software skills. In essence, ASC is a C++ library and, as such, can be compiled by a standard C++ compiler. When compiled, ASC code becomes an executable that either acts as a bit-level (RTL) simulation or produces a circuit in the form of a hardware netlist.

We present an "adapter" that enables an integrated solution for (ii) and (iii) based on ASC, and only requires software skills. In conjunction with ASC, it thus enables the rapid development of a complete, efficient hardware-accelerated application without requiring hardware skills. Its use and effectiveness are demonstrated for *iDCT* on a Virtex-II Pro platform.

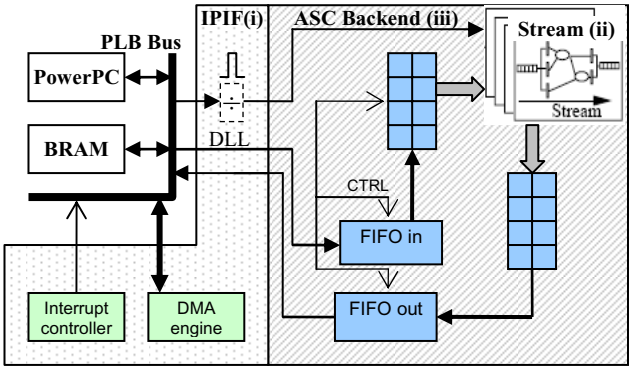

Figure 1: System based on Virtex II® FPGA

2. Adapter Architecture Overview

Xilinx's *EDK* tool, used for embedded-application development, enables creation of an *Intellectual Property Interface* (IPIF) module connected to the *Processor Local Bus* (PLB) as shown in Figure 1. Our IPIF module (i) (dotted pattern) comprises a fixed, accelerator independent part, typically comprising Xilinx-provided modules (E.g., DMA, interrupt controller and DLL); an ASC-generated accelerator (ii) (Stream), and a flexible part (iii) (dashed pattern) connecting them. Parts (i) and (iii) jointly constitute the adapter.

The flexible part of the adapter (iii) matches the speeds and data path widths of (i) and (ii). It probes the data structures of (ii) for information regarding the data path widths and the number of accelerator inputs (gray arrows), and generates the SER/DES with the appropriate widths and clocks. The FIFO's interface toward (i) operates at the *PLB* clock, and the interface toward (ii) operates at the (ii) clock. Finally, (iii) exports its own interface (connection points) in order to connect to the fixed part of the adapter (i). The results are expressed in (automatically generated) code that is written in PamDC and is executed after the generation of the accelerator (ii) logic. The adaptive part (iii) thus overcomes the inability of ASC-generated modules to be connected automatically as a hardware accelerator to other modules.

14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'06)
0-7695-2661-6/06 $20.00 © 2006 IEEE

For convenience, the use of the adapter-generation code is specified in the *makefile* of the accelerator generation. The result is that from a programmer perspective, the accelerator itself (ii) and the accelerator-specific portion of the adapter (iii) are jointly generated by ASC in the form of a combined RTL netlist. This netlist acts like a "*black box*" that is imported into the IPIF module along with the Xilinx-provided parts. The system assembly, map and place & route operations are performed by the Xilinx EDK software.

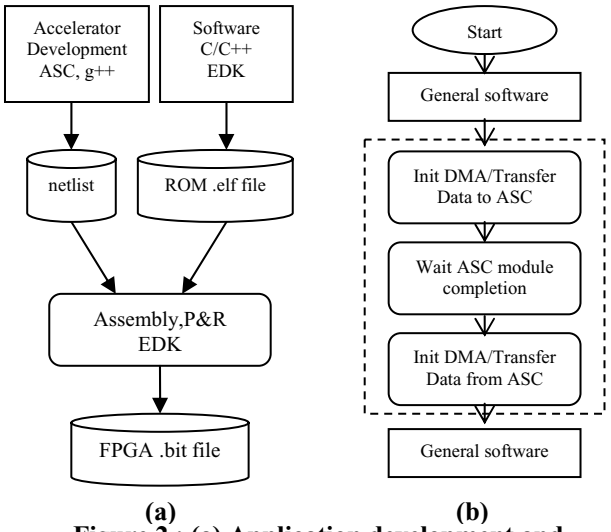

Figure 2 : (a) Application development and (b) Execution flows

3. Application Development Methodology

We assume that the HW/SW partitioning is performed separately, marking a section(s) of software for hardware acceleration. The accelerator generation process, depicted in Figure 2, is as follows:

1) Rewrite the desired section in the ASC language, and place in a separate file. (Top left block in (a).)
2) In the ASC *makefile*, specify the name of the IPIF adapter, data width, and optimization policy.
3) Compile the accelerator and execute it in order to generate the accelerator+ adapter netlist.
4) Replace the original code with ASC module commun. functions (Figure 2(b)→top right block in (a)).
5) Copy the netlist to the IPIF directory and invoke the EDK platform generation process to generate the final downloadable bit file. (Remainder of (a)).

4. Evaluation Example

The system uses the Memec Design Virtex-II Pro FF672 Development Kit. An *iDCT* example is used for evaluation of system performance and its integration process. The implementation has 37 add/subtract, 11 multiply, and 12 bit-shift operations per 8-input block. We compare a pure software application executed on the PowerPC processor core vs. a hybrid software/ASC accelerator. The PPC is clocked at 100 MHz in both cases. Power results are based on a rough measurement of the DC current from the 1.5V board power supply.

4.1 Execution results

The "software only" configuration uses the PowerPC core, with no other hardware configured within the FPGA. PowerPC current is *0.081A* when idle and *0.1A* during execution. Average software execution time per input set is ***10.2µs/block***. It thus requires $\boldsymbol{1.6 \cdot 10^{-6}}$ ***Joule/block***.

The ASC-generated *iDCT* accelerator (including the backend) occupies *3,341* slices, and the configured FPGA consumes *0.12A* in idle state. The accelerator runs on a *33.3 MHz* clock (down from a potential *38.3MHz* due to DLL limitations). The internal buffer size is *512x32* bits; therefore, each of the program iterations handles *64 8x32*-bit blocks. The execution time for the *64* blocks is *32.6µs*. The result is an average execution time of ***0.5µs/block*** and energy consumption of $\boldsymbol{2.0x10^{-7}}$ ***Joule/block***.

5. Related Work

Hybrid systems like Dash [4] demonstrate the ability to build applications without hardware design skills, but must generate all the hardware platform by themselves.

Others show the power consumption of image processing algorithms. [2] uses the *Galapagos* system, and [3] presents a hardware-implemented *DCT* algorithm in conjunction with a Xilinx Microblaze processor. Both confirm the energy savings and speedup obtainable by implementing critical software sections in hardware.

6. Conclusions

The joint use of ASC and the adapter introduced in this work is demonstrated to enable rapid development of highly efficient hybrid SW/HW applications while using only software skills. In our *iDCT* example, we show a 20X speedup and at least 7X reduction in energy consumption relative to a software-only solution. This can be improved by increasing buffer sizes and including an additional DMA engine.

7. References

[1] Oskar Mencer, David J. Pearce, Lee W. Howes, Wayne Luk, "Design Space Exploration with A Stream Compiler," Proc. IEEE International Conference on Field- Programmable Technology (FPT'03), December 2003

[2] J. Noguera, R. M. Badia, "Power-Performance Trade-Offs for Reconfigurable Computing," Proc. International Conference on Hardware-Software Codesign and System Synthesis (CODES+ISSS04), September 2004

[3] M. Ouellette, D. Connors, "Analysis of Hardware Acceleration in Reconfigurable Embedded Systems," Proc. 19th International Parallel and Distributed Processing Symposium (IPDPS 2005), April 2005

[4] J.M. Saul, "Hardware/Software Codesign for FPGA-Based Systems," HICSS, 32nd Annual Hawaii International Conference on System Sciences, 1999.