FlexSoC
Past, Present & Future

Magnus Själander*, Martin Thuresson†

* *VLSI Research Group, Department of Computer Science and Engineering, Chalmers University of Technology, Sweden*
† *High Performance Computer Architecture Group, Department of Computer Science and Engineering, Chalmers University of Technology, Sweden*

ABSTRACT

The FlexSoC program, launched in 2003, aims to develop new architectural techniques and programming models for high performance System on Chip (SoC). The target platforms for FlexSoC are embedded systems such as PDAs and high performance mobile phones. Important properties for embedded systems are long battery life, high performance and the flexibility to adapt to new protocols and standards. This abstract presents some of the work conducted by the participants of the FlexSoC program.

KEYWORDS: System on Chip, Embedded, Reconfigurable, Compression

1 FlexSoC

The design-spectrum for embedded systems ranges from custom ASICs to general purpose processors (GPPs). Using an ASIC solution makes it possible to tailor the hardware for the specific application thus achieving an efficient solution with the lowest power dissipation that meets required performance. The downside is the total lack of flexibility which makes it costly to adapt to new standards. GPPs on the other hand offers flexibility but at the cost of high power dissipation which is often related to the instruction fetch and decode. Todays embedded systems are often built around one or several GPPs that controls a number of hardware accelerators which is used for computational intensive tasks. With this approach flexibility and programming becomes a difficult issue.

The proposed idea [HJLE+03] is to replace the fixed instruction set architecture (ISA) with the possibility to tune the instruction set per application (AS-ISA[2]). This could then in real-time be converted to a native ISA that is executed on the hardware. An obvious advantage of this approach is that the program could be efficiently stored in memory and reduce the power dissipation for instruction fetch and decode.

For further information about the FlexSoC program please visit our web site[3].

[1]E-mail: {hms,martin}@ce.chalmers.se
[2]Application Specific ISA
[3]`http://www.flexsoc.org`

2 VLSI Issues[4]

The aim of the FlexSoC program is to design flexible System on Chip (SoC). In our work we have taken a bottom up approach by starting to look at the datapath components and how they can be made flexible. We have especially addressed the potential of adjusting the computational width depending on actual operand size. In traditional design the datapath components is created for the maximum operand size making them slow and power dissipating when operating on smaller operand sizes, which is often the case.

2.1 Twin-Precision Multiplier

In our work we have shown how to design an efficient twin-precision multiplier [SELE04] that is capable of adjusting the computational width to smaller operand sizes. In a single instruction multiple data (SIMD) like fashion the twin-precision multiplier is also capable of performing two small multiplications in parallel.

The idea behind the twin-precision multiplier is that when performing an $N/2$-bit multiplication [5] in an N-bit multiplier only one quarter of the partial products are used. For a radix-2 tree multiplier the number of adders doing any useful computation for an $N/2$-bit multiplication is also roughly one quarter of the total number of adders. By forcing the unused partial products to zero when doing an $N/2$-bit multiplication the dynamic power can be eliminated in the adders not doing any useful computation. When forcing unused partial products to zero the critical path delay is also reduced. The shorter path delay can be harnessed in a flexible system by either increasing the clock speed for higher performance or lowering the supply voltage to achieve even higher power dissipation reduction.

We have also shown that by applying reconfigurable power gating it is possible to reduce static power dissipation in different regions of a twin-precision functional unit [SDLEE05]. This was done by using the super cutoff CMOS (SCCMOS) technique on the adders in the reduction tree of the twin-precision multiplier and a tailored supply grid.

The results show that the power dissipation of a 16-bit twin-precision multiplier operating on 8-bit operands can be more than three times lower than for a conventional 16-bit multiplier.

2.2 Future Work

On the way up towards efficient instruction decoding we are currently looking into audio decoders as a set of applications that can be run on a flexible datapath. The main issues that will be address in this work are: i) organization of application specific functional units ii) flexible interconnect structures and iii) flexible control.

The envisioned datapath will consist of application specific functional units such as the discrete cosine transform (DCT) and Huffman decoder, which is commonly used in audio applications. The functional units have to be connected with a flexible interconnect structure in order to make it possible to create different pipeline combinations. It is also possible that a general purpose processor (GPP) is needed to run application code that can not be mapped onto a specific functional unit.

---

[4] Work primarily done by Magnus Själander

[5] Multiplication with half the operand size as of an N-bit multiplier

3 Architectural Issues[6]

There are many architectural challenges that could be addressed in our framework. Background and application studies has shown that memory is an important component in our target systems and we have chosen to start by addressing the static code size. By reducing the amount of memory for program code, we not only dissipate less memory, the cost of the system may also be smaller and the cache hierarchy can be more efficiently utilized.

3.1 Static Code Size Compression

In our work [TS05] we have evaluated different schemes for static code compression with dynamic decompression in the instruction fetch pipeline stage. Our focus has been on dictionary based compression schemes. In the baseline algorithm, identical sections of code inside the program are stored as one copy in one location, the dictionary, and the original instructions are replaced with *codewords* that identify the correct dictionary entry. This simple compression scheme allows efficient storage and execution with low performance overhead.

Previous work has looked at different ways of allowing more instructions to be candidate for the dictionary using more flexible codewords. Instead of identical sequences of instructions, they are allowed to differ in either full instructions [LSSC03] or operands [CLR03]. We have compared these and analyzed what type of flexibility in the codewords that has the largest gain in compression ratio and introduced a framework in which the previous schemes can be combined.

Our results show that operand parameters alone are most efficient in compressing the programs. We also noted that with our combined approach, we can achieve the same compressibility with smaller dictionary and less codewords. This makes it possible to execute the programs more efficiently. This work also shows that the general instruction set architecture is not an efficient representation of the algorithm. The fact that the dictionaries of various applications look quite different from each other also support our idea of AS-ISAs.

3.2 Future Work

Continuing the work on efficient storage and execution, we are looking into ways to efficiently transfer not only instructions, but also data. Our current studies look at the data transfered between CPU and memory, but a long term goal is to use the same efficient representation at other levels in the memory hierarchy as well. Our initial results shows that significance based compression is a well suited tool for this.

4 Acknowledgment

This research has been sponsored by the Swedish Foundation for Strategic Research (SSF) under the FlexSoC program.

[6]Work primarily done by Martin Thuresson

References

[CLR03] Marc L. Corliss, E. Christopher Lewis, and Amir Roth. DISE: a programmable macro engine for customizing applications. In Doug DeGroot, editor, *ISCA '03: Proceedings of the 30th annual international symposium on Computer architecture*, volume 31, 2 of *Computer Architecture News*, pages 362–373. ACM Press, June 2003.

[HJLE+03] John Hughes, Kjell Jeppson, Per Larsson-Edefors, Mar y Sheeran, Per Stenström, and Lars "J." Svensson. FlexSoC: Combining Flexibility and Efficiency in SoC Designs . In *Proceedings of the IEEE NorChip Conference*, 2003.

[LSSC03] Jeremy Lau, Stefan Schoenmackers, Timothy Sherwood, and Brad Calder. Reducing code size with echo instructions. In *Proceedings of the international conference on Compilers, architectures and synthesis for embedded systems*, pages 84–94. ACM Press, 2003.

[SDLEE05] M. Själander, M. Drazdziulis, P. Larsson-Edefors, and H. Eriksson. A low-leakage twin-precision multiplier using reconfigurable power gating. In *Proceedings of the 2005 IEEE International Symposium on Circuits and Systems*, 2005.

[SELE04] M. Själander, H. Eriksson, and P. Larsson-Edefors. An efficient twin-precision multiplier. In *Proceedings of the 2004 International Conference on Computer Design*, October 2004.

[TS05] Martin Thuresson and Per Stenstrom. Evaluation of extended dictionary based static code compression schemes. In *Proceedings of the International Conference on Computing Frontiers*, pages 77–86. ACM, ACM Press, 2005.