See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/3806036>

Flexible design of SPARC cores: A quantitative study

Conference Paper · February 1999

DOI: 10.1109/HSC.1999.777389 · Source: IEEE Xplore

CITATIONS

5

READS

24

2 authors:

Tomás Bautista

Universidad de Las Palmas de Gran Canaria

41 PUBLICATIONS 202 CITATIONS

SEE PROFILE

Antonio Nunez

Universidad de Las Palmas de Gran Canaria

141 PUBLICATIONS 468 CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

AGATE-325630-2 [View project](#)

Flexible Design of SPARC Cores: A Quantitative Study

TOMÁS BAUTISTA AND ANTONIO NÚÑEZ
CAD Division, IUMA – Applied Microelectronics Research Institute.
University of Las Palmas de Gran Canaria.
Tel. no.: +34 928 451250 Fax no.: +34 928 451243
E-35017 Las Palmas de Gran Canaria, Canary Islands.

E-mail: bautista@cma.ulpgc.es

Abstract

In this paper we present experimental results obtained during the modelling, design and implementation of a full set of versions of SPARC v8 Integer Unit core aimed for embedded applications in digital media products. VHDL has been the description language, Synopsis tools those for the logical synthesis, and Duet Technologies' Epoch has been used for the physical layout of the final circuits. They have been mapped to a 0.35 μm , three metal layers process. The quantitative results given characterize suitable points in the design space. They show how much microarchitecture, design, datapath granularity and module decisions affect performance and cost functions. Design space exploration down to physical layouts is made possible by modelling techniques based on configurable VHDL descriptions.

1 Introduction

Codesign techniques for problem solving rely on algorithmic partitioning and assignment to hardware and software based tasks. Tasks mapped to hardware are aimed for speed and often require development of fixed specific processors. In addition a new market has emerged with many companies offering Intellectual Property designs in the form of either hard cores or soft cores, trying to use socket interface draft standards. IP cores as well as proprietary cores are also intended for their reuse in new product developments with short time to market.

However in recent years the advantages of programmable solutions have also been highlighted. These solutions rely on standard processors available as cores for embedded systems. This approach helps in software development since they are based on well established processor architectures and efficient optimising compilers. Process technology ad-

vances are also bringing these processors to speed marks that make software solutions ever attractive.

As a combination of these trends general-purpose processor architectures are evolving introducing family derivations including better support for domain-specific problems. One of these domains is signal processing and in particular the digital media domain including audio, video, graphics and signal processing. Variations and derivations from established architectures include additional scalar units, additional extended precision accumulators, multiply and accumulate (MAC) units, additional floating point (FP) units, different data type support and formats in registers and register files—for instance to support limited SIMD operations—and architectural extensions to the instruction set supported by those units. In the digital media domain, examples of this are provided by MIPS, SPARC, HP PA-RISC, or HITACHI and ARM processor families among others. A comparison of the state of the art features offered by these processors are given in http://www.mips.com/Documentation/isa5_tech_brf.pdf, including also the Intel Pentium family MMX extension.

Pentium and HP PA-RISC are proprietary technologies. MIPS and ARM are open technologies in the sense that the architecture is aimed to be licensed to different manufacturers. SPARC is an open architecture based on architecture compliance and design certification. MIPS has proven its success in the digital media domain, for low and medium end applications. SPARC architectures have proven an equivalent potential to MIPS architectures in their respective evolution.

However the implementation parameters will remain key aspects for success, for given technologies. Embedded systems and many low-end general purpose systems are very cost sensitive. It is critical to find ways to reduce cost while increasing performance. Performance must not only be measured in cycles/task, often optimised at pipeline and architectural levels, but must also include clock frequency and power consumption achieved. Physical aspect ratio is also essential for optimising embedded cores with those application-

specific memory, coprocessor or peripheral device modules that have to fit each other on-chip.

The purpose of our research has been to study, for the digital media domain, the impact on real state, clock speed, power consumption and other physical implementation functions, of the following variables and parameters:

- ISA extensions including MAC, FP, register types, and special operators
- Microarchitectural and design decisions for a given ISA
- Control of physical design by tool management with emphasis in modules and datapath granularity decisions at synthesis time and at placement and routing time.

With this aim the open SPARC architecture has been chosen. Stages conducted have been:

- Development of VHDL versions of SPARC v8 ISA with IU, FPU and VIS extensions
- Development of a VHDL based synthesis and compilation technique with easily configurable VHDL options, for flexible synthesis of experimental versions of each design and its parameterized variations.

This paper reports on results obtained so far from the design experiences conducted for the Integer Unit of SPARC. Section 2 comments on pertinent SPARC features and main stream compiler technology to address SPARC variations and extensions. Section 3 gives a few relevant details about the strategy and technique used for modelling of processor cores. Section 4 presents the layout of the experiments for assessing the impact of microarchitectural and design decisions, and that of modules and granularity as physical design decisions. Section 5 presents a summary of main results and their discussion. Section 6 concludes the paper and conclusions.

2 SPARC architecture, compilers and kernels

SPARC is a RISC style instruction-set architecture that defines the instructions, register structure and data types for the integer unit (IU) and an IEEE 754 standard floating point unit. It allocates opcodes and defines a standard interface for a coprocessor unit. It assumes a linear 32-bit virtual address space for user application programs [New91].

The reasons for choosing SPARC¹ as a core are due to our purpose to use it in different applications, and in particular in low-cost multimedia applications [BMCN96]. With this application in mind, there were several reasons for making this decision:

1. Architecture related reasons:

- SPARC is an open and scalable architecture, which leaves the designer with a great degree of freedom. There are a lot of possible design alternatives depending on the performance desired.
- SPARC is very well-suited for real-time and embedded applications. There are even specific extensions upon the basic SPARC specifications especially developed for using SPARC cores in embedded systems.
- Another interesting feature for using SPARC in embedded systems is that the architecture presumes the presence of a windowed register file. This is attractive in applications where processes of different nature coexist and where fast traps handling is required. Support for this is achieved with the windowed register file since it allows fast context switching.

2. Compiler and kernel related reasons:

- For the processor cores in embedded systems a software must be resident in order to handle proper start-up mechanisms, device intercommunication, interrupts management, and main process execution. For producing machine code for the processor the GNU C compiler has been used. By taking the GNU C compiler as a reference [Sta91, Dar96], the view a compiler has is a Register Transfer Level view of the machine and what are the valid Register Transfer Language expressions. That is, the compiler's view of a processor is that of the Instruction Set Architecture. This includes any architectural extensions. From this knowledge the compiler is capable to produce an optimized machine code program that will correspond to a certain task described in the programming language.
- There are software development tools already created for SPARC that can help in the design, efficient compilation and debugging of the software to be run on these cores.
- The introduction of Windows CE opens the area to new multimedia applications for hand-held devices based on non-Intel processors. There is extensive support given by other operating systems including real-time ones for the SPARC platform.

3 Modelling strategy for the processor core

By making a traditional customized partition of one of these systems, a high degree of freedom has resulted for deciding the main features of the processor core. In this way architectural and implementation particularities can be better exploited.

¹It has been chosen more specifically the SPARC 32-bit version 8 specification

The modelling process [BMCN97] is highly based on the use of a graphical schematics tool with which a system can be described as an interconnection of different elements. At the same time, each element can be described by means of another schematics or by a HDL description. By making a suitable partitioning on the type of basic building blocks that should be part of every schematics, the whole system can be easily conceived and debugged.

For a processor core, the first partition to be made corresponds to the traditional view of a processor as split into a Data Path and a Control Unit. On a second partition step, the Data Path is based upon an RTL view. In this view the Data Path is conceived as built of different elements for making the calculations and data storing needed in the execution of the set of implemented instructions. Additionally, special registers are inserted to act as pipeline elements for both data and control signals.

On the other hand, the Control Unit is split into two sub-units. The first one is for managing the decoding and dispatching execution of instructions, as well as for sequencing management. The second one is the main heart of the whole machine, with the central Finite State Machine which rules all the actions performed and the elements for producing proper control signals for the different elements. These signals depend on the FSM, Instruction Register contents, traps management, external signals, and other elements.

As shown in [BMCN97], this centralized alternative is better suited than the one based on distributed FSMs (see figure 1) because it allows a better control of the instruction set, in a flexible design workbench. This strategy offers additional significant benefits which result in a well suited environment for making needed modifications on the instruction set in what respects to the decoding logic.

Modifying the instruction set can imply the modification, removal or appearance of specific processor resources in the architecture. This operation is flexible indeed due to the facilities provided by the graphical environment and the features of the hardware language. This also applies to implementation and architectural decisions upon localized elements of the architecture. Other parameterization options are also possible due to the capabilities of the description language, in which, by means of constants, specific behaviour options can be enabled or disabled in the synthesis steps.

4 Experiments: Impact of modules use and microarchitectural decisions

Based on a defined Instruction Set Architecture a large amount of processors can be built. Even between two processors with the same microarchitecture, just a difference in the implementation technique of a subpart can make a major difference for good or for bad.

The environment set up for our experiments allows that, for the same instruction set, different design alternatives can

(a) With distributed FSMs

(b) With a centralized FSM

Figure 1: Two possible organizations for generating control signals

be evaluated and compared. These are introduced into the models by working on special-purpose elements of the architecture or by reorganising some of them and their interconnections. Taking this into account, a set of versions has been developed with the following distinguishing parameters²:

1. Ratios of datapath modules vs. standard cells.
2. Branch prediction:
 - (a) always predict taken,
 - (b) predict taken only if backwards.
3. Checking of possible modification of the condition codes by the previous instruction on branches. As seen in fig. 2, at the decode stage of the branch instruction in cycle 'c', when fetching the instruction in the delay slot, the address for the next instruction to fetch has to be calculated. The problem is that Instruction 1 could change the condition also in 'c', so the address calculated in this cycle is predicted³. A parameter to evaluate for making a prediction or not is to check if the condition codes could be modified by Instruction 1 or not.

²These design references will be used as alternatives in tables 1, 2 and 3.

³If we wait for the condition codes to be set by Exec. 1 for deciding the address to set in 'd', the cycle time increases.

4. Calculation of the address for the next fetch

- with just an add operation in a cycle
- with two additions. This way, the two alternatives are calculated at a time, the proper one is chosen depending on the sequencing decisions and the other one is saved in a register. If at any time the decision was wrong (for instance, in mispredictions), then the saved previously is retrieved.

5. Bypass mechanism set

- from the end of the execution stage of the previous instruction to the decode stage, or
- from the beginning of the write-back stage of the previous stage to the execution stage.

Figure 2: Timing of conditional branches

5 Results

Within this bench of versions, a set of implementations have been carried out. The technology chosen has been a 3-metal 1-poly 0.35 μm CMOS process. The physical synthesis of this subset of implementations has been developed in automatic mode—as the tools can be further forced to achieve specific physical design constraints—. This way, benefits of microarchitectural and design decisions can be better evaluated.

Tables 1, 2 and 3 summarize quantitative data from the experiments. The full set of implementations, their physical layouts, and other measurements and ratios obtained can be seen in [Bau99]. Fig. 3 gives two sample layouts.

The name of the tables summarize combinations of several physical design parameters. Option ‘2.a, 4.a, 5.a’ in table 1 means an implementation with ‘predict taken’, ‘one add per cycle’ and ‘by-pass from end of execution stage’ and refer to section 4.

Many qualitative considerations can be analyzed from these data, since they show significant performance variations. This analysis is best carried out jointly with data from

other on-chip circuit blocks required by the embedded application under consideration, and stresses the convenience of the flexible design strategy set up for the experiences.

Table 1: ‘Lowest’ amount of modules

Options	Area (sq. mm)	Power (mW/MHz)	Freq. (MHz)	Transistors
2.a, 4.a, 5.a	15.2719	10.21932	53.38	184602
2.a, 3, 4.a, 5.a	18.7347	10.68084	66.27	187564
2.a, 4.b, 5.a	15.3189	10.07378	58.28	186088
2.a, 3, 4.b, 5.a	13.5150	10.68283	59.36	187884
2.a, 4.b, 5.b	15.6237	10.52504	61.70	184035
2.b, 4.b, 5.a	14.8836	10.55471	67.19	187580
2.b, 3, 4.b, 5.a	13.2435	9.98221	70.93	185909

Table 2: ‘Mean’ amount of modules

Options	Area (sq. mm)	Power (mW/MHz)	Freq. (MHz)	Transistors
2.a, 4.a, 5.a	16.9508	10.35614	54.95	173583
2.a, 3, 4.a, 5.a	11.4791	9.42461	58.67	169896
2.a, 4.b, 5.a	12.0548	9.37507	89.18	171634
2.a, 3, 4.b, 5.a	7.3983	9.17711	90.63	172104
2.a, 4.b, 5.b	10.3395	9.56579	75.65	169724
2.b, 4.b, 5.a	7.8112	9.14947	90.44	172228
2.b, 3, 4.b, 5.a	12.6047	9.63190	82.77	173816

Table 3: ‘Highest’ amount of modules

Options	Area (sq. mm)	Power (mW/MHz)	Freq. (MHz)	Transistors
2.a, 4.a, 5.a	12.3987	10.10201	65.99	171978
2.a, 3, 4.a, 5.a	11.5498	10.09086	70.84	171846
2.a, 4.b, 5.a	11.8015	9.96977	94.18	174285
2.a, 3, 4.b, 5.a	10.2570	10.09608	88.43	175240
2.a, 4.b, 5.b	13.1659	9.99910	76.78	173631
2.b, 4.b, 5.a	12.6801	9.64188	96.94	174667
2.b, 3, 4.b, 5.a	10.9568	9.79798	94.27	174292

As a general conclusion, the more standard cells used the less optimal the design results. As far as more modules are inserted inside the design, speed and other functions usually get improved. In some occasions this is not the rule. Some reasons for this can be found, in part, in the detailed connection mechanisms of the design tools, which coupling has not been fully achieved.

On the other hand, area is an item somehow unpredictable when modules and standard cells coexist on the same design. From the results it is clear how much designer intervention and guidance is advisable when an area/shape-optimized design is desired.

6 Conclusions

In this paper we have shown some experimental results from the modelling, design and implementation of set of versions of SPARC v8 Integer Unit. As it has been demonstrated, the impact of microarchitecture and design features as well

(a) c33_a

(b) c34_a

Figure 3: Two different versions of the SPARC v8 IU

as the impact of the use of custom modules, can be decisive in the performance obtained from a design. This is in addition to the plenty of architectural decisions that can be incorporated into large embedded processor designs at different levels.

Further work will report on similar quantitative studies including Register File sizes and formats, FPU's, and VIS extensions.

7 Acknowledgements

The authors thank the support obtained from project DSIPS, CICYT no. TIC970953.

References

[Bau99] T. BAUTISTA. "Flexible Generation and Mod-

elling of SPARC Cores for Specific-Application Domains". PhD thesis, School of Telecommunication Engineers, University of Las Palmas de G.C. (1999).

[BMCN96] T. BAUTISTA, G. MARRERO, P.P. CARBALLO, AND A. NÚÑEZ. Towards a low-cost processor architecture for multimedia. In J. FIGUERAS, editor, "XI Conference of Design of Integrated Circuits and Systems", pages 445–450. Universidad Politecnica de Catalunya, CPDA, Diagonal, 647, E-08028 Barcelona, Spain (November 1996).

[BMCN97] T. BAUTISTA, G. MARRERO, P.P. CARBALLO, AND A. NÚÑEZ. Rapid-prototyping of high-performance RISC cores with VHDL. In CAPT. GREG PETERSON AND DR. PHILIP WILSEY, editors, "Rapid Systems Prototyping with VHDL", pages 43–52, Arlington, VA (October 1997). VHDL International, IEEE Computer Society.

[Dar96] G. DART. An investigation into processor descriptions within GCC. Technical Report, CMA, University of Las Palmas de G.C. (1996).

[New91] M. NEWMAN. "SPARC Strategy and Technology". Sun Microsystems, Inc. (1991).

[Sta91] R.M. STALLMAN. "Using and Porting GNU CC" (June 1991).