

# Experiences with Retargetable Compilation using Catalyst

W. De Rammelaere, E. Hilken, T. Lawell, R. McGarity, P. Le Moenner, P. Kritzinger, F. Steininger, M. Thomas

Motorola SPS, Le Mirail, BP 1029

31023 Toulouse Cedex, FRANCE

Werner.De.rammelaere@motorola.com

## 1. Introduction

The design of a System on a Chip (SoC) is a challenging task, and tackled quite differently by different companies. The problems involved are in fact common to many marketplaces. First, the market requirements tend to be vague and change during the project. With a long design cycle, a system is often outdated by the time it is completed. Second, the marketplace is tiered. The hardware and software requirements for the high end are different from those of the low end, so different design points are required. Third, there are tight cost and power consumption constraints. Lastly, the software effort is greater than the hardware effort, especially because traditional methods use assembly language for the DSP algorithms. From the CAD perspective we can claim that SoC design is imposing new constraints on the tools, without a clear leader tool for SoC design.

Solving these multitude of problems is key for business success: faster Time to Market and a flexible design environment are the key elements to counter these problems. The current methodologies used in industry to achieve this success often take a conservative design approach to minimize risk. As a consequence state of the art tools and methodologies are not easily adopted. The hardware -software codesign effort remains difficult and making the software of the SoC run on the hardware is a long difficult process.

In order to tackle these problems in the context of DSP intensive SoC designs, we developed the Catalyst flow, which represents a flow to design domain specific instruction set processors (DSIP). It was extensively discussed in [5]. One key enabler to the solution is the use of a retargetable compiler such as Chess[2], which allows fast analysis of, and improvements to, a chosen ISA, followed by a fast and efficient compilation and simulation of the chosen target architecture.

## 2. Evolutions of A Voice Processing Module

### 2.1 A DSIP for Voice Codecs such as G723, G729, EFR and HR.

To evaluate the flow we have decided to design a VPM (Voice Processor Module) testchip for the application domain of voice coders: the ETSI and ITU standard bodies provide ANSI-C reference code

for a number of popular voice codec algorithms such as G723, G729, EFR<sup>1</sup> and HR<sup>2</sup>, along with testbenches to verify the code. Since the given algorithms have been implemented on various popular DSP engines, they provide a nice test case and allow for comparison with real products. In addition these algorithms are popular in many applications, and one can easily think of adding these algorithms into the system as a IP block that communicates with the rest of the system via e.g. memory-mapped IO. The block thus fits into the work on IP creation within Motorola SPS.

The Catalyst team explored several architectures, using the methodology described in [5]. Working at a high level of design abstraction, we were able to quickly make such trade-offs as:

- variation of the number of registers (data registers, base address registers, ...)
- number of read/write access to the data memory and mechanisms to access the memory
- bundling of instructions: a dedicated instruction can be added to the architecture that groups consecutive instructions into one.

After about 5 months of C profiling and architectural trade-offs, we finalized on a first architecture, LSE (Load Store Engine). Its main characteristics are depicted in tables 1 and 2 below. We will expand on the meaning of the tables further on. LSE has been realized in silicon, and is fully functional.

The LSE uses a 16-bit instruction word and a 16-bit datapath (except for the 32-bit accumulator). This harvard-architecture machine can do up to 2 MAC operations/cycle in parallel with a 2 operand load to feed the MAC units. Since the core is mainly meant for DSP algorithms, we support mainly indirect, indexed and post increment addressing schemes. This simplification of the addressing schemes plays an important role in hardware design and verification. The philosophy of the methodology calls for a move of the hardware complexity towards compiler complexity; since they are designed in tandem this is

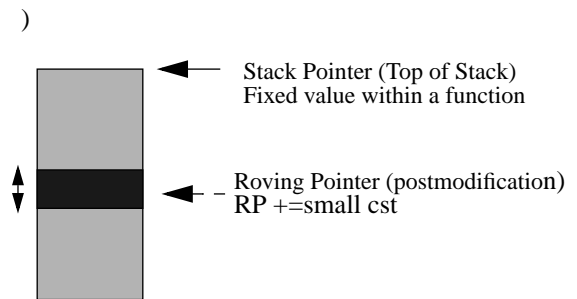
<sup>1</sup>. Enhanced Full Rate GSM

<sup>2</sup>. Half rate GSM

a major advantage, since the resulting hardware is much simpler to design and verify, and interrupts etc are easier dealt with. One example of this complexity move is the existence of a SW pipeline instead of a HW pipeline: for various reasons every load from data memory gets buffered into an intermediate register before being routed to the actual data registers. This 2-cycle mechanism is modeled in the ISA explicitly, and it is scheduled by the compiler. It is essentially the same as a HW pipeline, except that there is some overhead in the ISA and hence also in code density. The 2 MAC units are not symmetrical: because of a limited bandwidth to memory we cannot always perform two MAC operations, and hence we decided to drop this feature in the next evolutions. The voice codecs cannot easily exploit it in their most MIPS intensive parts anyhow. The LSE has 1 pipeline in the instruction decode stage: one cycle is used for fetch-partial decode of the instruction. This also means that a minority of the instructions (jumps etc) execute in 2 cycles, since the machine stalls one cycle to get a correct new instruction.

In order to test the flexibility of the technology, to exploit the experiences gained from the LSE design and to exploit new features of the compiler, we have gone through 2 more iterations at the nMI level. The first improved version of LSE, VoIP, took about 1 man month to achieve (at the nMI and nDI description level), and the G723 and G729 codecs were used as driving vehicle of the ISA definition, as well as the EFR algorithm. The actual primitive instructions of the LSE core were kept untouched to a large extent, but the degree of parallelism, the ISA itself, the number of registers etc changed. Also, we exploited new features. The performance results are shown in the same table, next to the LSE results.

The VoIP design uses less registers in total, partially because of better support in the compiler for larger (32-bit) data registers than the standard data type of the target machine in the compiler. In addition, it uses a 'roving pointer' scheme to access constants and variables on the stack: we noticed that the LSE needed too much opcode space to generate constants. A new compiler feature allowed us to use a constant memory to store these constants, and to retrieve them in a postmodify way: the CP (constant pointer) points to the next constant that will be loaded, and it changes by a few digits (part of the instruction) every time a new load occurs. This results in a better code density. A similar mechanism is used to access variables on the stack (Roving pointer RP



**Figure 1** - Roving Pointer 'travels' (roves) through Stack Memory

Finally we have reiterated on the ISA and architecture some more (2 man months), mainly focussing at improving the code density, as this is key for a large number of embedded applications. We also removed the software pipeline instructions: as we found out that the major addressing scheme could be the postmodification scheme, the need for the pipeline at that level (critical path issues) could be moved into the address access of the data memory, hence removing the critical path in the data-read part of the memory access. The resulting core, eclipse, can have one or 2 stages in the instruction decode section, and we are preparing it so it can have a hw pipeline in the execution units. This should allow the core to run in a 200Mhz range below 1.8V in 0.25um technology. We are also introducing the support of bytes in the core.

## 2.2 Results

Results shown below can still be improved by

Arch	Mips	Code Size
LSE	25	20KB
VoIP	22	18KB
eclipse	19	16KB

**Table 1 Performance Results on EFR codec**

adapting the C code more towards the architecture: no major modifications have been performed on the actual C code. In addition, we have not yet introduced very dedicated instructions in our cores: most codecs spend 25-30% of their MIPS in 1 or 2 code segments, and it is surely possible to make a few very dedicated instructions which improve the MIPS numbers locally, and seriously. This decision is motivated by the fact that we wanted to avoid making an ASIP core instead of a DSIP core; this DSIP can be a basis for even more specific solutions.

Also, as compiler and architecture are closely linked together, major improvements can still be expected by closely following the compiler

improvements with the hardware: hardware pipelining, multiple data-type support and e.g. inter-function optimizations in the compiler will have a positive impact on the overall results. The cost to do so is not negligible: the automated HDL generation needs to follow the evolutions in nDI and nMI, and the associated synthesis scripts may also be affected. .

Feature	LSE	VoIP	eclipse
Instr. width	16	16	16
Data Width	16,32	16,32	16,32,48
Addressing	in,id,pi	pi,rp,cp	pi,rp,cp
SW Pipeline	1	1	0
IHW pipe	1	1	1-2
EHW pipe	0	0	0 or 1
main diff	2 MACs	constant	bytes
CG	1-3	1-2, cp	1-2,cp
# RF (16-bit)	40	27	27
cycles/instr	1 (2)	1 (2)	1 (2-3)

**Table 2: Main Architectural Characteristics.**

in=indirect;id=indexed;pi=post increment;IHW=Instruction HW; EHW= execution unit HW;CG= constant generation cycles; RF=Register File; cp=constant pointer;rp=roving pointer

### 3. Conclusions

We believe that, for the application area of embedded systems on silicon, we have proven that the methodology works: the architectural improvements reduce total system costs, reduce hardware design time, ease the effort to port to a new process technology, virtually eliminate the need for writing DSP software in assembler, and can enable Motorola to react more quickly to market changes. We also experienced that numerous optimizations are still possible, and that this is only the start of a trend towards more DSIP oriented SoC designs. However, the boundaries of the approach are still under study, and the more business related aspects (e.g. customer support) need serious attention. As the Catalyst project progresses, we will further demonstrate this new approach, and aim at converting it into a Motorola strategic asset.

### 4. References

[1] R. Braek and O. Haugen: *Engineering real time*

- [2] G. Goossens, I. Bolsens, B. Lin, F. Catthoor, "Design of heterogeneous ICs for mobile and personal communication systems, *Proc. IEEE ICCAD-94*, November 94, pp. 524-531.
- [3] Michel Mouly, Marie-Bernadette Pautet, *The GSM System for Mobile Communications*, Cell & Sys, 1992.
- [4] A. Fauth, J. Van Praet, M. Freerick, "Describing Instruction Set Processors Using nML". *Proc. European Design and Test Conference*, Paris, March, 1995.
- [5] W.De Rammelaere. K.Eckert et all, *Catalyst: A DSIP Design Flow development in Industry*, ISSS99 Conference, November 99,