

EXPERIENCES WITH AUTOMATIC GENERATION OF AUDIO BAND  
DIGITAL SIGNAL PROCESSING CIRCUITS

Jan M. Rabaey(\*) and Robert W. Brodersen

University of California, Berkeley  
(\*) Current affiliation : IMEC, B-3030 Leuven, Belgium

Abstract

A set of digital signal processing circuits, generated by the Lager design synthesis system ([Po84], [Ra85]), is described. This system maps a behavioral, assembler level description of an algorithm into a set of concurrent operating, customized processors and dedicated i/o circuitry. The nature of this restricted, but flexible target architecture makes it possible to span a large range of applications in the field of speech processing, audio and telecommunications.

Introduction

Automated synthesis of circuits has become very popular recently. Most of these so called 'silicon compilers' however are basically function (module) generators and can only translate a user defined design structure (module definitions and interconnections) into a layout description of the circuit (e.g. [De82]). This is definitely not sufficient for the system designer, who wants to translate his behavioral, algorithmic description (block diagram, flow graph) into a circuit layout and this in a fairly automatic way. Until now, only a very limited number of systems aiming at this level of input description have been published (e.g. [Fo83], [Va84]).

The Lager system, which has been used for the generation of the circuits described in this paper, belongs to the latter class. While the above mentioned systems tend to hardwire the algorithm into a customized datapath, Lager is oriented towards microprogrammable structures with decision making capabilities. This has the advantage that the automatic generation and implementation of complete signal processing systems on a single chip is feasible, while the former systems are more oriented towards filtering operations.

After a description of the target architecture and the software tools, composing the Lager framework, an overview of the Lager generated applications will be given. Two of these applications will then be studied in more detail. The paper is concluded with a number of reflections on the architecture and the synthesis process.

The Target Architecture

The most important architectural feature of the Lager system ([Po84]) is the use of concurrency : the computations are distributed over a set of simple, bit-parallel pipelined processors, which have been optimized to perform a certain subtask of the algorithm. Interprocessor communication pro-

ceeds over a customized network of bit-serial paths. This communication protocol is completely transparent and conflict free and causes a minimal hardware overhead. The system supports a parallel data bus for sample rate input/output operations, while slower, frame rate operations proceed over an interrupt driven host i/o buffer.

The core processor is very simple, dedicated and parametrizable. Data and control paths are strictly separated. The major parts of the datapath are :

- an arithmetic unit with four pipeline stages, which allow for simultaneous memory access, shift, complement-add and i/o-operation. A parallel-serial multiplication procedure has been selected to avoid an expensive array multiplier. All arithmetic is performed in two's complement notation.

- a parameterizable RAM-memory for local variables.
- an address arithmetic unit with two index registers for indexed addressing and table look up.

The controller is kept to a minimum, being a master program counter, a microcode ROM delivering undecoded control words and a slave program counter for program loops. This controller only supports a simple subprogram but NO branches, neither unconditional or conditional. A user defined finite state machine allows for decision making operations : depending upon the processor status (sign of accumulator, values of index registers), a condition bit is set, which is used to control a conditional write operation.

Due to this simplicity, up to four or five processors, optimized for a particular task, can be put on one chip (for a 3 to 4 micron technology). This increases the throughput and the computational capabilities in a substantial way.

The Lager Synthesis System

A complete set of tools have been developed to support the layout synthesis and the debugging of the algorithmic description. A software tool flowgraph of the framework is pictured in Fig. 2.

The user input to the system consists of a single description, the 'design file', which defines a set of processor parameters (as e.g. the wordlength), variable declarations and i/o definitions. The main part of the description however consists of a specification of the algorithm in the form of an assembler program for the processors. This input can be debugged using an efficient, interactive and dedicated emulator/simulator. At the present time, a higher level front end for the Lager system has been developed. At this level, the algorithm is described using the flowgraph oriented Silage language ([Hi85]), which is compiled into the

'design file'-language using a specialized software compiler.

The layout synthesis process proceeds in two major steps : in a first pass, the symbolic input description is mapped into the target architecture. The resulting hardware description contains the parameters for the basic macrocells (or modules) and the connectivity information (Note that this is the input level for the commercial silicon compilers !). The actual layout generation starts with the creation of the modules using a flexible module generator. Modules are produced by stacking handcrafted library cells on top of each other, making the connections by abutment. The library contains about 170 basic cells. This is followed by the placement and routing phase, performed using an interactive floorplanning tool.

#### Application field

The selection of this particular target architecture confines the application field of Lager to the speech, audio and telecommunication field with sampling frequencies ranging from 8 kHz to 150 kHz (with a clocking frequency of 5 MHz). The number of processors on one chip normally ranges from one to four for a 3-4 micron technology, though an eight processor implementation of fir-filters has proven to be feasible. An overview of the Lager generated applications is given in Table 1. It demonstrates that the architecture can support a wide range of applications, going from straightforward, computation intensive filtering over simple higher speed applications to complete system implementations including heavy decision making.

The rest of the paper will be devoted to a more detailed discussion of two of those applications, located at opposite ends of the scope of the compiler. The first example demonstrates the capability of the system to implement low speed, computationally intensive algorithms, while the second handles a high speed adaptive function.

#### Full duplex LPC vocoder

A block diagram of the implemented vocoder algorithm is shown in Fig. 3 and is composed of two major parts, an analysis and a synthesis function. The sampled speech signal enters or leaves the system at a rate of 8 kHz. The extracted or transmitted speech parameters, being the energy contents, the pitch and ten reflection coefficients, are exchanged at a frame rate of about 90 Hz with a host processor. To support this decimation/interpolation, the host interface is used, acting as a data buffer between the fast signal processing front end and the slow host processor.

The major functions to implement are the analysis and synthesis LPC lattice filters (10th order), the computation of the correlations, the generation of the excitations and the pitch tracking. The distribution of the computations over a set of concurrent processors has been based on following observations : - the required arithmetic accuracy differs over the subfunctions of the algorithm : the pitch tracking function e.g. is based on decision making and integer operations and does not perform high precision arithmetic. The wordlength in this processor can be limited to 18. The correlator on the other hand has to perform squaring and dividing and needs high accuracy, hence the 26 bit wordlength of this

processor.

- Parts of the analysis and synthesis functions have been merged to make efficient use of the looping facilities offered by the controller. It is efficient to implement the lattice filter of both the analysis and synthesis functions in one processor, which realizes the lattice-core as a subroutine.

- A number of minor subfunctions have been scattered over the processors to equalize the utilization of the processor power. The feasibility and optimality of this operation depends upon the precedence relations and a minimization of the number of interprocessor communications.

The above demonstrates that the processor assignment is a non-trivial task and is based on a balancing of often contradictory requirements. It is the conviction of the author that a rule based system seems to be the appropriate way to tackle the problem, this until precise heuristics and/or procedures have been defined. At the present time, the allocation is done manually by inspection of the block diagram (which often yields reasonable results at the first try !). Note the importance of high level description languages as Silage in this prospect : at this level, different processor allocations can be compiled and compared in a very short time.

Inspection of the implemented microprogram shows that 35% of the 1500 instructions, executed per sample, include a part of a variable-variable multiplication. A speed up of the serial parallel multiplication with a factor of two would make it possible to merge the filter and correlator processors. This can be achieved with a minimal hardware overhead by adding an optional second order Booth's decoding module. This block has recently been added to the standard Lager library.

The pitch tracking is a typical example of a highly decision making oriented function, which can hardly be handled by data path oriented or bit serial architectures. The implemented function is derived from an algorithm attributed to Gold. Every six samples, a new pitch is determined by a voting mechanism, which compares the computer pitch of six signals, derived from the original speech signal. The decimation, inherent to this technique, is built in naturally into the architecture in the form of a modulo-6 indexing counter in the address arithmetic unit. The crucial element in this processor is the user defined, decision making finite state machine, which counts 18 minters.

The automatically produced layout of the vocoder is pictured in Fig. 4. The total dimension of the chip is 7mm & 7mm and means a 50% area increase compared to a handcrafted design of the same function. A substantial area reduction could be achieved with a more efficient and flexible floorplanner as the Flint-tool of LagerII. The automatic approach also resulted in a small increase in active area, this due to the more general building blocks used e.g. for the interprocessor communications and for the control. The use of Lager however resulted in an enormous reduction in design effort, compared to the two man-years used in the handcrafted design. Note that the complete 'design file' for this example amounts to 400 lines of code.

### Decision feedback equalizer and timing recovery for ISDN

This example has been selected to demonstrate the top end capabilities of the Lager Architecture. The described functions are part of the U-interface of the ISDN and perform the line equalization and the recovery of the timing clock from the received signal. The derived clock is then used to drive the sampling device, the echo canceller and also the transmitter. A block diagram of the realized system and its environment are shown in Fig. 5. The main problem with this application is that the input data rate amounts to 144 Kbits/sec. This means that, for a clock frequency of 5 Mhz, only 35 instruction cycles are available per sample.

A two processor solution has been selected to meet this constraint : the first processor implements the decision feedback equalizer (dfe), while the second one executes the recovery function. The latter consists of a combination of additions and decision making operations, based on the signs of the last three equalized signals, and fits easily into the allotted time frame. The first processor however has to realize the function pictured in Fig. 6, which includes the execution of 13 multiplications ! Fortunately, the delay line values  $a_i$  are binary and can only assume the values  $\{-1, +1\}$ . This can be implemented in an efficient way in the Lager architecture : the complete delay line is merged into one 12 bit word, with two bits for each stage (10 for +1, 01 for -1). Updating of the delay lines consists in a two bit right shift, with the new value added at the msb-side. This representation can then be routed as a single coefficient into the modified parallel-serial multiplier (alternatively adding and subtracting the coefficients). This results in a 26 cycle implementation of both the fir-filter function and the updating of the coefficients. With a total of 34 cycles, the described realization clearly stays within the specifications.

A photograph of the Lager produced chip is shown in Fig. 7. The total die size is 5mm x 4mm (4u nmos) and contains 8300 transistors.

Recent experiments have shown that a denser solution with only one processor (in 25 cycles per sample) is feasible when using a more complicated, multiple branch controller. In this configuration, the original program counter is replaced by a sequencing finite state machine with 70 minterms. Other experiments have equally shown that, for algorithms with complex state diagrams, a complicated controller results in denser and faster solutions than the arithmetic power oriented solution. Note that this observation is only valid for dedicated controller structures : an implementation of the same algorithm on the TMS320 ([Ma82]) would take 80 cycles for the dfe-function only. This can be attributed to lack of pipelining in the datapath and the less flexible and inefficient conditional operations.

### Conclusions

The experiments with the Lager system have clearly demonstrated that the automated design synthesis techniques allow for a fast and reliable translation from algorithm to silicon. The key features of the Lager success are the restriction to a particular target architecture and the consistent use of

the module concept.

The same exercises also learned us the weak points of the original Lager system :

- The architecture needs additional flexibility and arithmetic power to extend the range of applications. A set of extensions to the target architecture have been performed or are in progress : addition of a Booth's decoder, more powerful decision making and i/o-operations, more flexible controllers, on chip a/d and d/a, etc.
- This added flexibility asks for more sophisticated mapping techniques. Different approaches are currently investigated, including rule based systems. It is this authors conviction that the mapping process is an iterative and interactive procedure, not only scanning over different hardware configurations but also over different algorithm alternatives. It is important to realize that the mapping problem is the key issue to the success of the automated layout compilation in the next few years.

The major advantage of the synthesis approach is the opening it provides for the system or algorithmic engineer to the world of VLSI realization. In this way, innovative algorithms can be tested more easily against realizability and usefulness.

### Acknowledgements

This research has been sponsored in part by the Defense Advance Research Project Agency, Contract No. MDA903-79-C-0429.

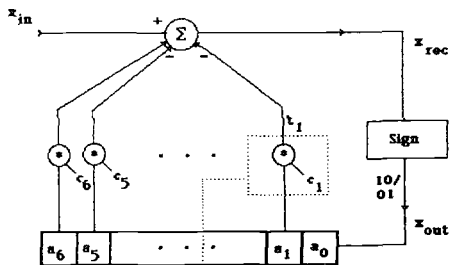
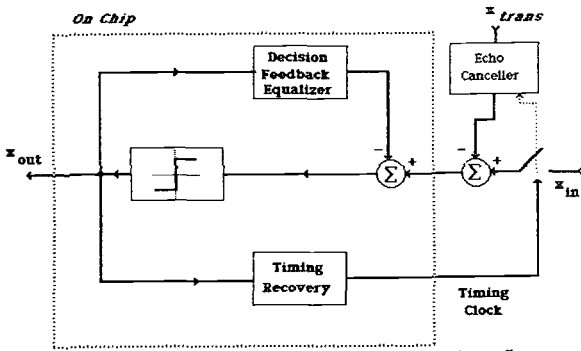
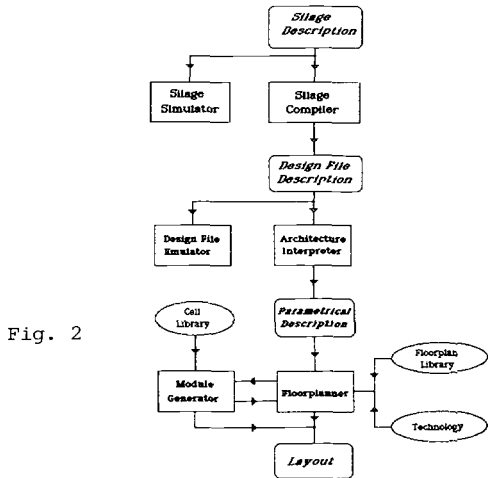
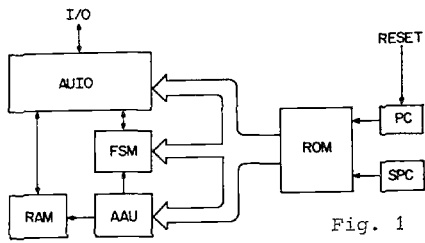
### References

- [De82] P. Denyer, D. Renshaw, N. Bergmann, "A Silicon Compiler for VLSI Signal Processors", Proc. ESSCIRC 1982, Brussels, pp. 215-218, Sept. 1982.
- [Fo83] J. Fox, "The MacPitts Compiler : A view from the Telecommunication Industry", VLSI Design, pp. 30-37, May/June 1983.
- [Hi85] P. Hilfinger, "Silage, A High Level Language and Silicon Compiler for Digital Signal Processing", Proc. CICC 1985, Portland, May 1985.
- [Ma82] S. Magar et al., "A Microcomputer with Digital Signal Processing Capabilities", Proc. IEEE ISSCC 1982, San Francisco, pp. 32-33, Feb. 1982.
- [Po84] S. Pope, J. Rabaey and R. Brodersen, "Automated Design of Signal Processors Using Macro-cells", from "VLSI Signal Processing", IEEE Press, 1984, pp. 239-251.
- [Ra85] J. Rabaey, S. Pope and R. Brodersen, "An Integrated Automated Layout Generation System for DSP Circuits", IEEE Trans. on Computer Aided Design, CAD-4, July 1985.
- [Tz85] C. Tzeng, "Timing recovery in digital subscriber loops", Phd-dissertation, Mem. No. UCB/ERL M85/29, U.C. Berkeley, April 1985.
- [Va85] J. Vandewalle e.a., "A Unified Box of VLSI Building Tools for Digital Signal Processing", Proc. ICCD '84, Portchester, October 1984.

Design	Processors	Transistors	Sample Rate
FIR	1	6600	8 KHz
Audio Equalizer	1	6900	30 KHz
Speech Scrambler	1	17000	8KHz
Residual Coder	1	17000	8KHz
Decision Feedback Equalizer	2	8300	144KHz
16 band Filterbank	2	23000	14KHz
300 Baud Modem	2	20000	9.6KHz
LPC Vocoder	3	27400	8KHz
ADPCM	2	25000	8KHz

Table 1 : Overview of Lager Generated DSP circuits

Processor Architecture



$$t_i = c_i \text{ if } a_i = 10 \text{ else } t_i = -c_i$$

$$c_{i+1} = c_i + k \cdot \text{err} \cdot a_i$$

$$\text{err} = \text{sign}(x_{\text{rec}} - c_0 \cdot a_0)$$

