The Symbolics Ivory Processor: A 40 Bit Tagged Architecture Lisp Microprocessor

Clark Baker, David Chan, Jim Cherry, Alan Corry, Greg Efland,
Bruce Edwards, Mark Matson, Henry Minsky, Eric Nestler,
Kalman Reti, David Sarrazin, Charles Sommer, David Tan
and Neil Wilhelm

VLSI Systems Group
Symbolics Cambridge Research Center

Abstract

This paper describes the VLSI implementation of a third generation symbolic processor optimized for the Lisp language. The design effort has resulted in a single chip implementation of a complete CPU which efficiently executes Lisp primitives and offers performance, system integration and manufacturability advantages over current approaches to Lisp processors.

1. Introduction

Lisp processors have been implemented in a variety of technologies with a number of objectives in mind. Contemporary Lisp processors use more board area, more power, and are more expensive than contemporary VLSI microprocessors [1][2][3]. As the market for symbolic processing matures there is a need for both higher performance, lower cost stand-alone systems, and embedded systems. The Ivory architecture was initiated with the intent of designing a single chip processor which provides on-chip support for a rich Lisp software environment.

This paper describes the resulting micro-architecture and some of the implementation details of the Symbolics Ivory, a single chip symbolic microprocessor.

2. Ivory Microarchitecture

The Ivory microprocessor implements a 4-stage pipeline as shown in Figure 1. The first stage fetches the instruction, decodes it, and adjusts the program counter. The second stage fetches the initial microinstruction, computes the operand address, and adjusts the stack pointer. The third stage fetches the operands and computes the result. The fourth stage stores the result, unless a fault has occurred, in which case it restores the state of the third stage.

PROGRAM:	I	D	E	C
PUSH A				
PUSH B	ADD	PUSH B	PUSH A	——
ADD	POP C	ADD	PUSH B	PUSH A
POP C	——	POP C	ADD	PUSH B
	——	——	POP C	ADD
	——	——	——	POP C

Figure 1. Ivory Pipeline Stages

Instructions spend only a single cycle in the first two pipeline stages, but can spend an arbitrary number of cycles in the execute stage. Simple instructions, such as "push", "add", and "eq" execute in a single cycle. Conditional branches are resolved in the second (D) stage. A taken conditional branch executes in two cycles; a not-taken branch in one.

Figure 2 shows a block diagram of the Ivory CPU. Instructions are fetched directly from a 32 word (up to 64 instructions) direct-mapped instruction cache. The cache, which is filled by an autonomous prefetcher, serves to buffer instructions arriving from memory and hold small program loops. A bypass path provides the instruction directly from memory when the first stage is stalled on a cache miss.

The operand address calculation data path contains the stack frame pointer registers and a 32-bit adder/subtractor. It computes the address of the operand and stack pointer adjustment according to the macroinstruction. This data path is also used in parallel with the main data path to accelerate function call/return.

Microcode is stored in a 1200 word x 180 bit ROM. This ROM reads 8 words in parallel, allowing a late 1-of-4 select of the next microinstruction to be based on the current ALU condition.

The main data path contains a 128 word top-of-stack cache, a 32-word scratchpad (which contains a duplicate of the top word on the stack in a fixed location), the ALU, and tag checking logic. The ALU includes an adder, boolean unit, shift/mask logic, and support for one-bit-per-cycle integer multiply/divide. Tag checking is done in parallel with with the ALU operation, so that in the common cases no time penalty is paid for type checking [4]. Similarly, ECC checking of data from memory is done in parallel with the ALU using the on-chip ECC logic. Bypass paths for both the top-of-stack cache and scratchpad forward the result of the previous instruction to the ALU as necessary.

An on-chip 16 entry fully associative cache holds recent virtual to physical address translations. The cache is maintained by microcode, with hardware to assist searching in-memory translation tables and cache replacement selection.

The Ivory processor supports a pipelined memory bus which can have up to four outstanding requests at once. An associative queue of outstanding request addresses is maintained for detecting when instructions arrive from memory and installing them into the instruction cache. The memory interface protocol is implemented by an independent state machine which arbitrates between on-chip users of the memory system and other bus masters.

Hardware included for testing includes scan-out registers on the microcode ROM and real time observation of the micro program counter. All hardware defined registers are accessible from high level language primitives. The microcode ROM may be checksummed via the scan-out path while running Lisp.

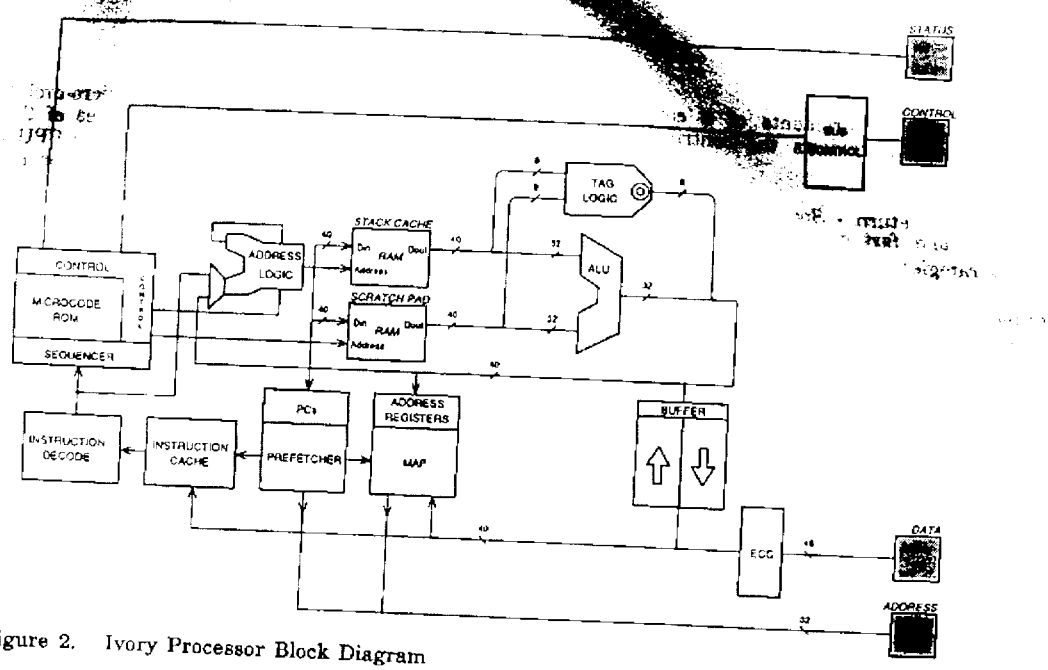

Figure 2. Ivory Processor Block Diagram

3. VLSI Implementation

Ivory was designed using a highly integrated proprietary VLSI design system called NS [5], which is implemented on the Symbolics 3600 Lisp machine. This provides a highly interactive design and simulation environment and in addition, a symbolic layout methodology that allows the complete chip data base to be designed in a process independent manner. The current design is electrically optimized for a 2u/1.6u double level metal CMOS process, although the data base may be retargetted to 1.2u and 1.0u processes to decrease cost and increase speed. The layout density in all styles of layout is comparable to orthogonal geometry hand designs. The design contains approximately 390,000 transistor sites with 255,000 placements. A total elapsed time of 10 months was needed to progress from architecture specification to the first tape-out.

A two phase clock is employed which is routed throughout the chip. Registers and latches use nMOS pass transistors with restoring inverters as the basic storage device. An extra synchronization clock is used to generate memory timing signals.

In line with the ability to retarget the design using the symbolic layout tools and the sheer impossibility of hand optimizing 390,000 transistors, the circuit design of the chip was kept fairly simple. The majority of logic in the design is static - either fully complementary CMOS logic gates, pseudo nMOS gates for high fan-in NOR gates or nMOS pass transistor logic for selected muxes. Dynamic gates are only used in the microcode ROM, instruction decoder and the RAMs. The RAMs are implemented using a standard 6 transistor static cell, while the CAMs in the design use a 10 transistor static cell. All control logic is in the form of static, fully complementary standard cells. The symbolic layout for control modules was automatically generated using a standard cell place and route system. A standard cell library, memory library and data-path library is used for most of the designs in the chip. Customization of logic and circuits is kept to a minimum to ensure uniformity of performance and reduce the amount of verification required at the circuit level. As an example, the design employs five 32 bit adders which all have to operate twice every clock cycle. The static adder provides a good compromise between size and speed. Based on a 4-bit Manchester carry adder it employs a passaround path controlled by a distributed pseudo-nMOS lookahead gate to improve speed. The bit pitch is 40u while the length of the adder is 232u in 2u rules. It may be used as a single unit or combined into a carry-select adder. The typical speed for a 2u process is around 25nS.

Extensive use was made of simple layout generators for various modules, this capability being integrated into the NS design system. Examples include the ECC generation logic, instruction-decode rom and microcode rom. In these cases and in all programmable arrays the generation of the module was tied directly to the appropriate data structure and released microcode in the Lisp architectural simulator, thus alleviating any possible confusion with regard to architecture revisions. The following table summarizes the contribution of devices and area of the various types of circuitry in Ivory.

	Transistors(%)	Area(%)	Relative Density
Control	7.6	13.	?
Data-path	32.6	17.2	3.3
Memory	23.9	7.9	5.4
u-Code	31.9	11.4	4.8
Pads	3.6	9.9	.64
Routing	0	40	0

4. Feature Review

With respect to the requirements of a commercial Lisp processor the following reviews the contribution of Ivory hardware:

- Fast call and return - Specialized datapaths, parallel operations, and fast cycle time support the complex calling strategies required by Lisp.
- Runtime type checking - Parallel tag processor, late-branch ROM and comprehensive trap logic support generic arithmetic and pointer manipulation.
- Specialized Lisp operations - Pipelined memory interface and high level microcoded primitives support efficient implementation of operations such as CAR and CDR.
- Garbage Collection - On chip hardware to facilitate efficient GC algorithms such as the Ephemeral GC [6].
- Virtual Memory Support - On chip translation buffer, microcoded cache-miss backup and the support of CDR-coded lists (more compact physical memory representation).

Other features which improve performance and lower system integration cost include:

- A programmable interleaved memory interface to allow a wide range of memory system speeds and architectures to be used - ranging from small high speed caches to four-way interleaved standard MOS memories.
- A fast coprocessor interface, primarily used to provide high floating point performance.
- Fast "vector" instructions for garbage collection, database searching and graphics applications.

Depending on the system configuration, initial versions of Ivory will allow performance increases of 3-5 times that of current Symbolics products. Ivory supports the full Genera 7 symbolic processing environment. Figure 3 shows a chip photograph of the 2u Ivory Lisp microprocessor.

5. Summary

This paper has presented the micro-architecture and some of the significant technical features of the Symbolics Ivory CPU. It is the first single chip Lisp CPU that combines all the features required of next generation Lisp CPUs.

6. References

[1] Motorola, "**MC68020 32-Bit Microprocessor User's Manual**," Prentice Hall, 1985.

[2] J. Crawford, "**Architecture of the Intel 80386,**" *ICCD '86*, pp. 155-160.

[3] C. Hansen, D. Freitas, E. Hudson, J. Moussouris, S. Przybylski, T. Riordan and C. Rowen, "**A RISC Microprocessor with Integral MMU and Cache Interface,**" *ICCD '86*, pp. 145-148.

[4] D. Moon, "**Architecture of the Symbolics 3600,**" *Proceedings of the 12th Symposium on Computer* Architecture, June 1985, pp. 76-83.

[5]: J. Cherry, H. Shrobe, N. Mayle, C. Baker, H. Minsky, K. Reti, and N. Weste, "**NS: An Integrated Symbolic Design System,**" *VLSI '85*, E. Hoerbst, ed., Elsevier Science Publishers B. V., North-Holland, Amsterdam, 1986, pp. 325-334.

[6] D. A. Moon. "**Garbage Collection in a Large Lisp System,**" *1984 ACM Symposium on Lisp and Functional Programming*, pp. 235-246.

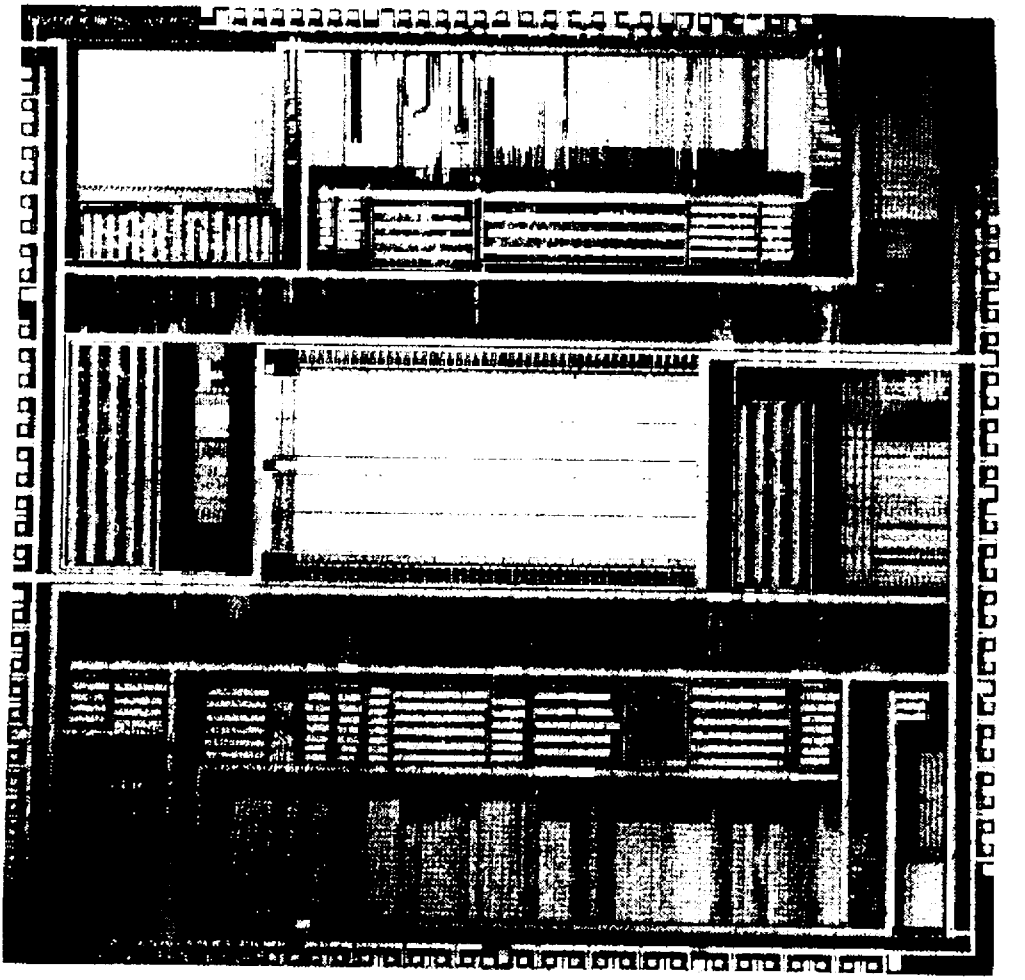

Figure 3. Ivory Photomicrograph