A 333-MHz dual-MAC DSP Architecture for Next-Generation Wireless Applications

Ravi K. Kolagotla, Jose Fridman, Marc M. Hoffman, William C. Anderson, Bradley C. Aldrich, David B. Witt, Michael S. Allen, Randy R. Dunton, and Lawrence A. Booth, Jr.
Analog Devices / Intel Joint DSP Development Center, 1501 S. Mopac Exp., Austin, TX 78746, USA

Abstract - We introduce the first DSP core developed at the Analog Devices and Intel Joint DSP Development Center. The 16-bit fixed-point core combines some of the best features of traditional DSPs and micro-controllers and compares favorably with dual-MAC DSPs on DSP specific benchmarks and with micro-controllers on micro-controller specific benchmarks. In addition, the core supports a rich set of alignment independent packed byte instructions to enable an efficient implementation of 3G algorithms in next-generation wireless applications. The deep and fully interlocked pipeline allows the core to run at 333-MHz in the 0.18-um TSMC process.

I. INTRODUCTION

The convergence of voice and video in next-generation wireless applications requires a processor that can efficiently implement advanced 3G algorithms. The Frio core is a dual-MAC modified Harvard architecture based processor that has been designed to have good performance on both voice and video algorithms. In addition, some of the best features of micro-controllers have been incorporated into the Frio core to allow it to replace both a DSP and a micro-controller on low cost wireless handheld applications.

The DSP features of the Frio core include one instruction port and two separate data ports to a unified 4GB memory space, two 16-bit single-cycle throughput multipliers, two 40-bit split data ALUs, two 32-bit pointer ALUs with support for circular and bit-reversed addressing, two loop counters that allow nested zero overhead looping, and hardware support for on-the-fly saturation and clipping.

The micro-controller features of the Frio core include arbitrary bit manipulation, mixed 16-bit and 32-bit instruction encoding for code density, memory protection, stack pointers and scratch SRAM for context switching, flexible power management, and an extensible nested and prioritized interrupt controller for real-time control.

The multimedia features of the Frio core include four auxiliary 8-bit data ALUs, and a rich set of alignment independent packed byte operation instructions. These instructions enable the acceleration of fundamental operations associated with video and imaging based applications such as are found in 3G wireless algorithms.

Execution time predictability is achieved with lockable caches that can be configured as SRAM, static branch prediction and data-independent instruction execution.

The architecture of the Frio core is introduced in section II. Section III describes the details of the first implementation of the Frio core, and section IV summarizes the performance of the Frio core on DSP specific benchmarks as well as image, video and micro-controller specific benchmarks.

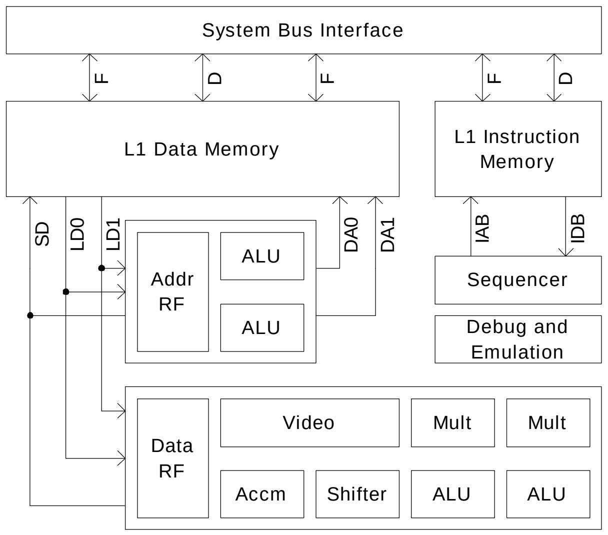

Fig. 1. Block diagram showing major subsystems

II. ARCHITECTURE

The Frio core is a modified Harvard architecture based processor. Instructions and data reside in separate L1 memories, but share a common L2 memory. All addresses are 32-bit allowing the Frio core to address a unified 4GB address space. Figure 1 shows the major units in the Frio core. The execution unit contains an eight entry 32-bit data register file, two 16x16-bit multipliers, two 40-bit split ALUs, one 40-bit shifter, four 8-bit video ALUs and two 40-bit split accumulators. Keeping the accumulators separate from the data registers allows for efficient load/store architecture to co-exist with the accumulator based design of traditional DSPs.

The data address generator contains two 32-bit address ALUs and an address register file. The address register file consists of six 32-bit general-purpose pointer registers and four 32-bit circular buffer addressing registers. The four circular buffer addressing registers have associated base, length and modifier registers, all of which are 32-bit wide. The address register file also includes a frame pointer to point to the current procedure's activation record and separate user and kernel stack pointer registers. The sequencer includes two zero overhead hardware loop buffers.

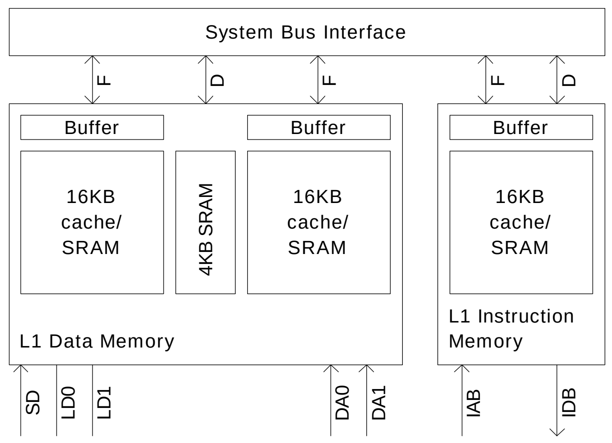

Fig. 2. Block diagram of L1 memories

A. Instruction Set

The instruction set for the Frio core has two types of instructions: those used primarily for DSP-oriented computation, and those used for micro-controller and general tasks. Specific Frio instructions are tuned for their corresponding task, but in general instructions can be inter-mixed with no restrictions.

Generally, DSP instructions read two 32-bit operands from the data register file, compute results, and either store these results back to the data register file or accumulate in the two internal accumulators. Each MAC unit is capable of computing a 16x16 bit signed or unsigned integer or fractional multiplications. Each ALU unit is capable of a 32-bit operation on two 32-bit inputs, a 16-bit operation on two 16-bit inputs, or two 16-bit operations on a pair of packed 16-bit inputs.

The micro-controller specific instructions provide the essential instructions needed to perform basic processor control and arithmetic operations. This set of operations includes load/store, arithmetic, logical, bit manipulation, branching, and decision making functionality. Conditional register move instructions are provided to allow efficient implementation of short if-then-else statements.

The load/store instructions support the following addressing modes: auto-increment, auto-decrement, indirect, circular, bit-reversed, indexed with immediate offset, post-modify with non-unity stride, pre-decrement store on stack pointer.

Each DSP instruction may be issued by itself, or in parallel with two load or one load and one store instruction.

B. Memory Architecture

The Frio core contains an L1 instruction memory and a separate banked dual-ported L1 data memory as shown in Figure 2. This allows two data load operations or one load operation and one store operation to occur in parallel with an instruction fetch. Both the L1 instruction and L1 data memories may be configured as either caches or as SRAM. Caches support ease of use and relieve the programmer from explicitly managing data movement into and out of the L1 memories. This allows code to be ported to the Frio core quickly, with no performance optimization required for the memory organization, without having to understand and program the system DMA controller.

The L1 instruction memory can be configured as either 16KB SRAM or as 16KB of lockable 4-way set-associative instruction cache. The Instruction Address Bus (IAB) is 32-bits wide and the Instruction Data Bus (IDB) is 64-bits wide. Misses in the L1 instruction memory are sent to the system bus interface using the F port. An external bus master, DMA for example, can use the D port to access the L1 instruction memory when it is configured as SRAM.

The L1 data memory consists of two 16KB superbanks and a separate 4KB of SRAM for scratchpad use. Each of the 16KB superbanks can be configured as either 16KB SRAM or as 16KB of 2-way set associative data cache. The data address busses (DA0 and DA1) are 32-bits wide. The load data busses (LD0 and LD1) and the store data bus (SD) are 32-bits wide. Misses in the L1 data memory are sent to the system bus interface using the two F ports. An external bus master, DMA for example, can use the D port to access the superbanks in the L1 data memory when they are configured as SRAM.

The memory management unit in the Frio core supports both protection and selective caching of memory. In general the memory is divided up into regions in which the memory management rules apply. The Frio core supports four different page sizes: 1KB, 4KB, 1MB, and 4MB. Each page is described by a set of cacheability and protection properties. These properties are stored locally for the most active regions in memory in Cacheability Protection Lookaside Buffers (CPLBs). Each of the three major buses (IAB, DA0 and DA1) have memory management.

C. Operating Modes

The Frio core has five modes: user, supervisor, emulation, idle and reset. The user, supervisor and emulation modes provide basic protection for system and emulation resources. Application-level code has restricted access to system resources. The system acts on behalf of the user through system calls whenever application-level code requires access to system resources. System resources include protected registers and protected instructions.

The Frio core is in supervisor mode when it is handling an interrupt at some level, or a software exception. The Frio core enters emulation mode as a result of an emulator event such as a watchpoint match or an external emulation request.

In the emulator mode, the Frio core fetches instructions from a JTAG scannable emulation instruction register. These instructions bypass the memory system and are directly fed to the instruction decoder.

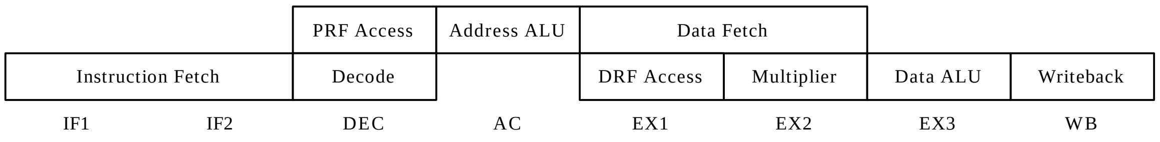

Fig. 3. Pipeline diagram

D. Interrupt System

The interrupt system on the Frio core is nested and prioritized and on an interrupt the processor state is saved on the kernel stack. The interrupt system consists of five basic types of events: emulation, reset, non-maskable interrupt, exceptions and general-purpose interrupts. Each basic event has an associated register to hold the return address, as well as an associated return-from-event instruction. The Frio core processes all interrupts and exceptions in the supervisor mode. The emulation event is processed in the emulation mode.

A higher priority event preempts a lower priority event. In addition, general-purpose interrupts can be configured to be self-nesting at the same priority level. This allows an external interrupt controller to extend the number of available interrupt inputs.

Reset is treated as an event and is lower priority than emulation. Upon exit from reset, the Frio core starts execution from the reset service vector in the supervisor mode. A return from interrupt instruction is used to exit the reset service routine and start execution in the user operating mode.

E. Debug Features

The debug features of the Frio core include watchpoint, trace, performance monitor and cycle counters. The watchpoint unit consists of six instruction address watchpoints and two data address watchpoints. The six instruction address watchpoints may be combined in pairs to create three instruction address range watchpoints. The two data address watchpoints may be combined to create a data address range watchpoint. Each of the watchpoints and watchpoint ranges is also associated with a 16-bit event counter. The trace buffer consists of a 16-element FIFO that records discontinuities in program flow. Discontinuities due to hardware loops are not recorded in the trace buffer. The trace buffer also has a compression feature that can be used to compress one or two levels of software loops. The performance monitor unit contains two 32-bit special purpose counters and registers to count the number of cycles or occurrences of an event of interest. There is also a 64-bit free running cycle counter that can be used for code profiling purposes.

III. IMPLEMENTATION

The first implementation of the Frio core uses an eight stage pipeline. The pipeline is fully interlocked with smart interlocking. This allows for the minimum number of stalls to be inserted to maintain program correctness. Table 1 describes the operations performed during each stage and Figure 3 shows the arrangement of the stages.

Table 1: Description of pipeline stages

	Name	Description
IF1	Instruction Fetch 1	Start L1 instruction memory access
IF2	Instruction Fetch 2	Finish L1 instruction memory access and align instruction
DEC	Decode	Start instruction decode and read Pointer RF
AC	Address Calculation	Data addresses and branch target address
EX1	Execution Stage 1	Read Data RF, start access of L1 data memory
EX2	Execution Stage 2	Finish accesses of L1 data memory and start execution of dual cycle instructions
EX3	Execution Stage 3	Execution of single cycle instructions
WB	Writeback	Write architectural states to Data and Pointer register files and process events

This arrangement of pipeline stages allows for the results of load instructions to be forwarded to the execution units without stalls. Both data accumulation using the data ALUs and pointer updates using the address ALUs are single cycle operations. This allows the Frio core to achieve a very efficient cycle count on critical inner loops.

Alignment independent byte operation is provided with control signals forwarded from the address ALUs to MUXes in the data register file.

The deep pipeline allows the Frio core to run at 333-MHz in the TSMC 0.18-um process.

IV. PERFORMANCE BENCHMARKS

The balanced execution and data memory bandwidth of the Frio core allows for efficient implementation of most DSP inner loop kernels. Figure 4 shows a code segment for a Finite Impulse Response (FIR) filter. Software pipelining and loop unrolling techniques are used to effectively adapt the algorithm to the computational pipeline [1]. It consists of two nested loops: a 2 instruction inner loop (LP1STR, LP1END), and a 7 instruction outer loop (LP0STR, LP0END). Each line of the inner loop has two MAC instructions. The operands for the MAC instructions come from registers R0 and R1, and are accessed as 16-bit register halves: R0.H (high) and R0.L (low). The load instructions use pointers I0 in post-decrement mode (R0.H=W[I0--]) to fetch a single 16-bit delay line input, and I3 in post-increment mode (R1=[I3++]) to

```
I0=delay_line_pointer; I1=input_pointer; I2=output_pointer; I3=coefficient_pointer;
P0=N/2; P1=T/2-1; R0 = [I1++] || R1 = [I3++]; [I0++M0] = R0;
        LSETUP (LP0STR, LP0END) LC0=P0                                             ;
LP0STR: LSETUP (LP1STR, LP1END) LC1=P1                                             ;
                A1  = R0.H*R1.L ,         A0  = R0.L*R1.L  || R0.H = W[I0--]       ;
LP1STR:         A1 += R0.L*R1.H ,         A0 += R0.H*R1.H  || R0.L = W[I0--] || R1 = [I3++] ;
LP1END:         A1 += R0.H*R1.L ,         A0 += R0.L*R1.L  || R0.H = W[I0--]       ;
        R2.H = (A1 += R0.L*R1.H), R2.L = (A0 += R0.H*R1.H) || R0   =  [I1++] || R1 = [I3++] ;
                                                              [I0++M0] = R0             ;
LP0END:                                                       [I2++]   = R2             ;
```

Fig. 4. FIR Filter implementation on the Frio core

fetch a pair of 16-bit coefficients. The inner loop is executed O(T/2) times and the outer loop is executed O(N/2) times for an effective cycle count of O(NT/2).

Note that the Frio core has no load to use latencies between load and compute instructions. This reduces the need for loop unrolling to only one iteration, improves code size, and allows simple assembly programming.

Table 2 summarizes the performance of the Frio core on the kernels of typical DSP benchmarks.

Table 2: Performance on DSP Algorithms

Algorithm	Cycle Count
N sample T tap real FIR Filter	NT/2
N sample T tap complex FIR Filter	2NT
N sample T tap real LMS Adaptive Filter	3NT/2
N sample T tap complex LMS Adaptive Filter	5NT/2
N sample B stage Biquad IIR Filter	5NB/2
N sample L section Lattice Analysis Filter	2NL
N sample L section Lattice Synthesis Filter	2NL
N sample T tap Convolution	NT/2
N sample Maximum or Minimum with or without Index	N/2
N dimensional Vector Dot Product	N/2
N dimensional Vector Sum	N
N dimensional Weighted Vector Sum	N
N dimensional Sum of Squares	N/2
N dimensional Weighted Sum of Squares	N
N dimensional Euclidean Distance	N
N dimensional Weighted Euclidean Distance	3N/2
Complex FFT Butterfly	3
256-point complex FFT (including bit-reversal)	3176
Viterbi for GSM (16 states, 378 soft decision symbols)	6357

In addition to the DSP kernels, the Frio core was designed to allow efficient implementations of typical image and video processing applications such as are found in next-generation wireless applications. Table 3 summarizes the performance of the Frio core on the kernels of typical image and video processing benchmarks.

Table 3: Performance on Image and Video Algorithms

Algorithm	Cycle Count
8x8 Discrete Cosine Transform (DCT)	284
8x8 Inverse Discrete Cosine Transform (IDCT)	404
NxN Sum of Absolute Differences (SAD), N=8,16	$N^2/4 + 2$
NxN Sum of Squared Differences (SSD), N=8,16	$3N^2/4$
NxN ZigZag Scan (Classical), N=8	$N^2 + 4$
NxN ZigZag Scan (Vertical), N=8	$N^2 + 8$
NxN ZigZag Scan (Horizontal), N=8	$N^2/2 + 11$
NxN Motion Compensation (1/2X), N=8	$3N^2/8$
NxN Motion Compensation (1/2Y), N=8	$N^2/4$
NxN Motion Compensation (1/2XY), N=8	$3N^2/4$
NxN Motion Compensation (Integer), N=8	$N^2/4$
Image Convolution with 3x3 Kernel (per pixel)	5
RGB24-YCrCb 4:4:4 Color Space Conversion (per group)	12

V. SUMMARY

We presented a new dual-MAC DSP core that combines some of the best features of traditional DSPs and micro-controllers. In addition, the rich set of multimedia instructions make the core ideally suited for implementing 3G algorithms in next-generation wireless applications.

ACKNOWLEDGMENTS

We would like to thank the large group of past and present Intel and Analog Devices employees who have contributed to the development of the Frio core, only some of whom are listed here: L. Basto, B. Bayerdorffer, B. Bonner, D. Borland, L. Bunimovich, K. Chalmers, D. Coleman, J. DeLargy, J. Dickol, T. Dinh, D. Gantala, M. Gill, R. Gopal, W. Hooper, R. Inoue, R. Jackson, S. Kannan, R. Kaul, C. Marler, J. Mason, S. Menon, J. Monaco, G. Overkamp, R. Peri, M. Perry, L. Phan, J. Reed, J. Revilla, C. Roth, E. Silvera, R. Singh, A. Sustaita, and T. Tomazin.

We also thank everyone who helped make this project a success.

REFERENCES

[1] J. L. Hennessy and D. A. Patterson, “Computer Architecture: A Quantitative Approach,” Second Edition, Morgan Kaufmann Publishers, 1996.