SH-4 Next-Generation DSP Architecture for VoIP

Joel Martinez
Peter Carbone
Srinivas Mandavilli
Dante Chu

SuperH RISC Microprocessor
Hitachi Semiconductor (America) Inc.
www.hitachi.com/semiconductor

Tutorial Goals
By the end of this tutorial, you should know the answers to these questions:

- What features in the SH-4 architecture enable efficient floating-point DSP?
- What specific SH-4 instructions are used for DSP?
- How can you improve DSP performance through software optimization?
- How many MACs per cycle can be achieved on the SH-4?
- What's the advantage of an SH-4 in a VoIP application?
Housekeeping

- Please ask questions
  - There is a lot of material to cover
- All telephones and pagers off, please
  - They disrupt the seminar and annoy people seated near you
- Fill out the evaluation forms as we go
  - Your feedback is important
- Recordings are not permitted

Tutorial Outline

- Why the SH-4 for DSP?
- History, Roadmap
- SH-4 Microprocessor core
- Summary of instruction set
- Floating point instructions
- System peripherals
- FIR, LMS, FFT filter examples and optimization techniques
- SH-4 VoIP implementation and tools
Classes of Processors for DSP

Traditional DSPs
- Conventional DSPs
- VLIW-based DSPs
  - MAP
- Superscalar DSPs

DSP-enhanced CPUs
- Hybrid RISC/DSPs
  - SH3-DSP
- Superscalar RISC
  - SH-4

Hitachi Super RISC Processors for DSP

<table>
<thead>
<tr>
<th>Year</th>
<th>SH-5</th>
<th>SH-4</th>
<th>SH3-DSP</th>
<th>SH-3</th>
<th>SH-DSP</th>
<th>SH-2</th>
<th>SH-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1995-97</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1998</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1999</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- SH7709 (80MHz)
- SH7707 (60MHz)
- SH7708S (60MHz)
- SH7709R (100MHz)
- SH7718R (100MHz)
- SH7750 (200MHz)
- SH7750 (266MHz)
- SH7751 (167MHz, PCI)
- SH7729 (187MHz)
- SH7729 (133MHz)
- SH7709A (133MHz)
- SH7604 (28MHz)
- SH7040/1 (33MHz)
- SH7032/34 (20MHz)
- SH7020 (20MHz)
- SH7615 (60-100MHz)
- SH7612 (60MHz)

10/100 Ethernet or USB
Why the SH-4 for DSP?

- 2-way superscalar
- Single cycle FMAC
- Single cycle 4-way dot product (FIPR)
- 4-cycle 4X4 matrix-vector multiplication
- 32 floating point registers (two banks)
- 32/64-bit load/store
- Delayed branch

SH-4 Device Family

<table>
<thead>
<tr>
<th>Part Number</th>
<th>TD4417750</th>
<th>TD4417750F</th>
<th>TD4417750VD128</th>
<th>TD4417750SF167</th>
<th>TD4417750VDF200</th>
<th>TD4417750VF200</th>
<th>TD4417750VF308</th>
<th>TD4417750VF408</th>
<th>TD4417751</th>
<th>TD4417751F</th>
<th>TD4417751VF133</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency (MHz)</td>
<td>200</td>
<td>167</td>
<td>128</td>
<td>167</td>
<td>200</td>
<td>266</td>
<td>167</td>
<td>133</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance, x1.8 RISC (MIPS)</td>
<td>360</td>
<td>301</td>
<td>230</td>
<td>301</td>
<td>360</td>
<td>479</td>
<td>301</td>
<td>239</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance, x7 FPU (GFLOPS)</td>
<td>1.4</td>
<td>1.2</td>
<td>0.9</td>
<td>1.2</td>
<td>1.4</td>
<td>1.9</td>
<td>1.2</td>
<td>0.9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Core Voltage (3.3V I/O)</td>
<td>1.95V</td>
<td>1.8V</td>
<td>1.5V</td>
<td>1.8V</td>
<td>1.35</td>
<td>1.5V</td>
<td>1.5V</td>
<td>1.35 V</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cache (KB) instruction/data</td>
<td>8/16</td>
<td>8/16</td>
<td>8/16</td>
<td>8/16</td>
<td>16/32</td>
<td>16/32</td>
<td>8/16</td>
<td>8/16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I/O</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Power, typical</td>
<td>1.9 W</td>
<td>1.25 W</td>
<td>400 mW</td>
<td>400 mW</td>
<td>-bd</td>
<td>-bd</td>
<td>400 mW</td>
<td>240 mW</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>External Bus</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>32</td>
<td>32</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Package</td>
<td>256 BGA</td>
<td>208-QFP</td>
<td>208-QFP</td>
<td>208-QFP</td>
<td>208-QFP</td>
<td>208-QFP</td>
<td>208-QFP</td>
<td>208-QFP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process</td>
<td>0.25</td>
<td>0.25</td>
<td>0.25</td>
<td>0.18</td>
<td>0.15</td>
<td>0.15</td>
<td>0.18</td>
<td>0.18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T-emp (possibly 40 b-85)</td>
<td>-</td>
<td>yes</td>
<td>yes</td>
<td>-</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Samples</td>
<td>now</td>
<td>now</td>
<td>now</td>
<td>Q2/00</td>
<td>TBD</td>
<td>TBD</td>
<td>Q2/00</td>
<td>Q2/00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mass Production</td>
<td>now</td>
<td>now</td>
<td>now</td>
<td>Q3/00</td>
<td>TBD</td>
<td>TBD</td>
<td>Q3/00</td>
<td>Q3/00</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Tutorial Outline

- Why the SH-4 for DSP?
- History, Roadmap
- SH-4 Microprocessor core
- Summary of instruction set
- Floating point instructions
- System peripherals
- FIR, LMS, FFT filter examples and optimization techniques
- SH-4 VoIP implementation and tools

2-way Superscalar Architecture

- Executes two instructions per cycle
- RISC and FPU instructions executed in parallel
- 200MHz 32-bit RISC engine @ 360 MIPS
  - 32-bit registers
  - 16-bit instructions
- 200MHz FPU engine @ 1.4 GFLOP
  - 32-bit registers
  - 16-bit instructions
- Access four instructions per memory access
  - 64-bit external bus
32-bit RISC Processor

- Load-store Architecture
- Separate RISC and FPU registers
- All registers are 32-bit
- Sixteen general registers
- Eight general shadow registers
- Seven control registers
- Four system registers
- Multiply Accumulate

Cache Memory

- 8 kbyte instruction cache
  - Holds 4k instructions
  - Direct mapped
  - 256 entries, 32-byte cache line
- 16 kbyte data cache
  - Configurable as 8 kbyte cache and 8 kbyte RAM
  - Direct mapped
  - 512 entries, 32 byte cache line
  - Copy-back or write-through mode
- Store Queue
  - 2 x 32 bytes
### General Registers

<table>
<thead>
<tr>
<th>General Registers</th>
<th>Bank Registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0_Bank0</td>
<td>R0_Bank1</td>
</tr>
<tr>
<td>R1_Bank0</td>
<td>R1_Bank1</td>
</tr>
<tr>
<td>R2_Bank0</td>
<td>R2_Bank1</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>R7_Bank0</td>
<td>R7_Bank1</td>
</tr>
<tr>
<td>R8</td>
<td></td>
</tr>
<tr>
<td>R15</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Control Registers</th>
<th>System Registers</th>
<th>Program Counter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status Register</td>
<td>MAC High Register</td>
<td>Program Counter</td>
</tr>
<tr>
<td>Global Base Register</td>
<td>MAC Low Register</td>
<td>Saved Program Counter</td>
</tr>
</tbody>
</table>

### Sixteen 32-Bit General Registers

**Used for Data Processing and Address Calculations**

- **R0 functions as:**
  - an Index Register in “Indexed Addressing” and in “Displacement Addressing” Modes
  - a Fixed Source Register
  - Destination Register

- **R15 functions as:**
  - General Register
  - Stack Pointer during exception processing

- **R0-R7 are banked**
  - 2 Register Banks are provided
  - Bank 0 used in User Mode
Control Registers

- Status Register - SR
- Saved Status Register - SSR
  - The current contents of SR are saved in SSR in the event of an exception or interrupt
- Saved Program Counter - SPC
  - The address of an instruction at which an interrupt or exception occurs is saved to SPC

Control Registers Cont’d

- Global Base Register - GBR
  - GBR is referenced as the base address in a GBR-referencing MOV instruction
- Vector Base Register - VBR
  - VBR is referenced as the branch destination base address in an event of an exception or interrupt
- Saved General Register - SGR
  - The contents of R15 are saved to SGR in the event of an exception or interrupt
- Debug Register - DBR
  - When user break debug is enabled, DBR is referenced as the branch destination base address instead of VBR
System Registers

- Multiply and accumulate registers - MACH/MACL
  - MACH/MACL is used to store the results of a multiply or a multiply accumulate instruction.
- Procedure Register - PR
  - The return address is stored in PR in a subroutine call using a branch or jump instruction.
- Program Counter - PC
  - PC indicates the instruction fetch address

Floating-point Unit

- Conforms to IEEE 754 standard
- 32 single precision or 16 double precision registers
- Two Rounding Modes
  - Round to Nearest and round to zero
- Two denormalization modes
  - Flush to zero and treat denormalized number
- Six exception sources
  - FPU error, invalid operation, divide by zero, overflow, underflow, and inexact
- Comprehensive instructions
  - single/double precision, graphics, system control
Floating-point Registers

31  0

**BANK0**

Single Precision Floating Point Register


Double Precision Floating Point Register

DP0  DP1  DP2  DP3  DP4  DP5  DP6  DP7  DP8  DP9  DP10  DP11  DP12  DP13  DP14  DP15

Single Precision Floating Point Vector Register

FV0  FV1  FV2  FV3  FV4  FV5  FV6  FV7  FV8  FV9  FV10  FV11  FV12  FV13  FV14  FV15

**XMTRX**

Four x Four Single Precision Matrix Register

Thirty-two 32-bit Floating-point General Registers
Configurable to four combinations

- **FR0, FP1, FP2 ... XF13, XF14, XF15**
  - 32 single-precision floating point registers
  - Switch between banks
- **DP0, DP2, DP4 ... XD12, XD14**
  - 16 double-precision floating point registers
  - Switch between banks
- **FV0, FV4, FV8, FV12**
  - 4 single-precision floating point vector registers
- **XMTRX**
  - 4 x 4 single-precision matrix register
Floating-point Control Registers

- Floating-point Status/Control Register, FPSCR
- Floating-point Communication Register, FPUL
  - Data transfer between FPU registers and CPU registers is carried out via the FPUL

Floating Point Accelerator

Two 128-bit simultaneous data transfers
Four fmuls in one cycle
4-input fadd in one cycle
Versatility of the SH-4 Arithmetic Accelerator

3D Graphics Geometry (1.2 GFLOPS)

\[
\begin{align*}
\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ 1 \end{bmatrix} &= \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{bmatrix}
\end{align*}
\]

A single SH-4 instruction, FTRV, can perform this matrix-vector multiplication every 4 cycles

16-tap, 40-sample Block FIR (1.6 MACs/cycle)

\[
\begin{align*}
\begin{bmatrix} Y_i \\ Y_{i+1} \\ Y_{i+2} \\ Y_{i+3} \end{bmatrix} &= \begin{bmatrix} x_i \\ x_{i+1} \\ x_{i+2} \\ x_{i+3} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} + \cdots + \begin{bmatrix} x_{i-12} \\ x_{i-13} \\ x_{i-14} \\ x_{i-15} \end{bmatrix} \begin{bmatrix} c_{12} \\ c_{13} \\ c_{14} \\ c_{15} \end{bmatrix}
\end{align*}
\]

1024-point, radix-2 FFT (35.4k cycles)

\[
\begin{align*}
\begin{bmatrix} in_1 \\ in_2 \\ -1 \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ a & -b & -a & b \\ b & a & -b & -a \end{bmatrix} \begin{bmatrix} in_{1,r} \\ in_{1,i} \\ in_{2,r} \\ in_{2,i} \end{bmatrix} \quad \Rightarrow \quad \begin{bmatrix} out_{1,r} \\ out_{1,i} \\ out_{2,r} \\ out_{2,i} \end{bmatrix}
\end{align*}
\]

A single DIF butterfly

Tutorial Outline

- Why the SH-4 for DSP?
- History, Roadmap
- SH-4 Microprocessor core
- Summary of instruction set
- Floating point instructions
- System peripherals
- FIR, LMS, FFT filter examples and optimization techniques
- SH-4 VoIP implementation and tools
Data Formats

- **Registers**
  - Operands are always long-words - 32 bits
  - byte or words are sign-extended into long words
- **Memory**
  - Accessible as byte, word or long word
  - Selectable for big endian or little endian

SH-4 Basic Pipelines

<table>
<thead>
<tr>
<th>General Pipeline</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>General Load/Store Pipeline</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Special Pipeline</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Special Load/Store Pipeline</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Floating-point Pipeline</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Floating-point Extended Pipeline</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
</tr>
</tbody>
</table>
Five Stage Pipeline

- An instruction is executed as a combination of basic pipelines
- A basic pipeline consist of 5 stages
  - Instruction Fetch (I)
  - Decode and register read (D)
  - Execution (EX/SX/F0/F1/F2)
    - Operation, address calculation or floating point computation
  - Data access (NA, MA)
    - Non-memory data access or memory data access
  - Write-back (S/FS)
    - write-back or floating point computation

Two-way Superscalar

- Queue holds 8 instructions which reduce pipeline stalls
- Two independent decode logic
- Instructions execute in parallel

![Diagram of Five Stage Pipeline and Two-way Superscalar](image-url)
Instruction Groups

- MT group
  - Compare, test (example: CMP, NOP and TST)

- EX group
  - Integer (example: ADD, AND, DIV, OR, XOR, SUB and SWAP)

- BR group
  - Branch

- LS group
  - Load/store (example: FMOV and MOV)

- FE group
  - Floating (example: FADD, FDIV, FCMP and FMUL)

- CO group
  - System Control (example: JMP, LDC, STC and CLRS)

Instruction Combinations
Executed in Parallel

<table>
<thead>
<tr>
<th></th>
<th>EX</th>
<th>FE</th>
<th>LS</th>
<th>BR</th>
<th>MT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>EX</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Floating Point</td>
<td>FE</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Load/Store</td>
<td>LS</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Branch</td>
<td>BR</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Compare</td>
<td>MT</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>System control</td>
<td>CO</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Pipelined Execution

Parallel execution

<table>
<thead>
<tr>
<th>Operation</th>
<th>Instruction</th>
<th>I</th>
<th>D</th>
<th>EX</th>
<th>NA</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>R2,R1</td>
<td>I</td>
<td>D</td>
<td>EX</td>
<td>NA</td>
<td>S</td>
</tr>
<tr>
<td>MOV.L</td>
<td>@R4,R5</td>
<td>I</td>
<td>D</td>
<td>EX</td>
<td>MA</td>
<td>S</td>
</tr>
<tr>
<td>ADD</td>
<td>R2,R1</td>
<td>I</td>
<td>D</td>
<td>EX</td>
<td>NA</td>
<td>S</td>
</tr>
<tr>
<td>MOV.L</td>
<td>@R4,R5</td>
<td>I</td>
<td>D</td>
<td>EX</td>
<td>MA</td>
<td>S</td>
</tr>
</tbody>
</table>

SuperH Instruction Set

- **SH-DSP**: 154 types DSP instructions
- **SH3-DSP**: 160 types MMU instructions
- **SH-1**: 56 types
- **SH-2**: 62 types 32-bit multiplier/accumulator
- **SH-3**: 68 types MMU instructions
- **SH-3E**: 84 types Floating-point instruction (single precision)
- **SH-4**: 91 types Floating-point instruction (single and double precision)

Copyright 2000 Hitachi Semiconductor (America)
## SH-4 Instruction Set

<table>
<thead>
<tr>
<th>Function</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Transfer</td>
<td>MOV, MOV, MOVT, SWAP, XTRCT</td>
</tr>
<tr>
<td>Arithmetic Operation</td>
<td>ADD, ADDV, ADDC, SUB, SUBV, SUBC, MULS, MULU, DIV1, DIV0S, DIV0U, CMP/cond, EXT S, EST U, NEG, NECG, MAC, MUL, D MulS, DMULU, DT, FIPR, FTRV, FRCHG, FSCHG, FCNVDS, FCNVSD (Floating point)</td>
</tr>
<tr>
<td>Logical Operation</td>
<td>AND, OR, XOR, TST, NOT, TAS</td>
</tr>
<tr>
<td>Shift</td>
<td>SHAR, SHLL, SHAL, SHLR, ROTL, ROTR, ROTCL, ROTCR, SHLRn, SHLLn, SHAD, SHLD</td>
</tr>
<tr>
<td>Branch</td>
<td>BRA, BSR, JMP, JSR, RTS, BF, BT</td>
</tr>
<tr>
<td></td>
<td>BRAF, BSRF, BF/S, BT/S</td>
</tr>
<tr>
<td>System Control</td>
<td>LDC, STC, NOP, CLRT, SETT, LDS, STS, CLR MAC, RTE, TRAPA, SLEEP, LDTLB, PREF, CLRS, SETS</td>
</tr>
<tr>
<td></td>
<td>OCB (Cache invalidation and write-back operation)</td>
</tr>
</tbody>
</table>

## Addressing Modes
- Register direct
- Register indirect
- Register indirect with post-increment
- Register indirect with pre-decrement
- Register indirect with displacement
- Indexed register indirect
- GBR indirect with displacement
- Indexed GBR indirect
- PC-relative with displacement
- PC-relative
- Immediate
Highlighted Addressing Modes

- Register indirect with post-increment
  - Mnemonic: @ Rn+
  - Effective address is register Rn contents. A constant is added to Rn after instruction execution

- Register indirect with pre-decrement:
  - Mnemonic: @ -Rn
  - Effective address is register Rn contents. Decremented by a constant value beforehand

  Constant is 1, 2, 4 or 8 for byte, word, longword or quad word operand, respectively.

Branch Instructions

- Branch if true/false - BT, BF
  - 8 bit displacement relative to PC (+/- 128 instructions)

- Branch if true/false - BT/S, BF/S
  - same as BT with delay slot

- Unconditional branch - BRA
  - - 12 bit displacement relative to PC (+/- 2K instructions)

- Absolute branch - JMP
  - Provides ability to branch beyond 4Kbytes

- Branch/jump to subroutine - BSR, JSR
  - Return Address is stored in PR register (PC->PR)

- Return from subroutine - RTS
  - Return Address is retrieved from PR register (PR->PC)
Conditional Compare Instruction

- **CMP/cond**
  - This instruction compares two registers and sets the T bit if the condition -"cond" is true
  - **CMP/EQ** If Rn=Rm set T bit
  - **CMP/GE** If Rn>=Rm, signed set T bit
  - **CMP/GT** If Rn>Rm, signed set T bit
  - **CMP/HI** If Rn>=Rm, unsigned set T bit
  - **CMP/HS** If Rn>Rm, unsigned set T bit
  - **CMP/PL** If Rn>0 set T bit
  - **CMP/PZ** If Rn>=0 set T bit
  - **CMP/STR** If any bytes are equal set T bit
  - **CMP/EQ** If R0=immediate set T bit

Tutorial Outline

- **Why the SH-4 for DSP?**
- **History, Roadmap**
- **SH-4 Microprocessor core**
- **Summary of instruction set**
- **Floating point instructions**
- **System peripherals**
- **FIR, LMS, FFT filter examples and optimization techniques**
- **SH-4 VoIP implementation and tools**
Floating-point Instructions

- FABS  absolute value
- FADD  add
- FCMP  compare
- FCNVDS convert to single precision
- FCNVSD convert to double precision
- FDIV  divide
- FIRP  inner product
- FLDI0 load immediate 0.0
- FLDI1 load immediate 1.0
- FLDS  load to system register
- FLOAT convert from integer

Floating-point Instructions (cont’d)

- FMAC multiply accumulate
- FMOV move
- FMOV move extension
- FMUL multiply
- FNEG negate value
- FRCHG FR-bit change
- FSCHG Sz-bit change
- FSQRT square root
- FSTS store system register
- FSUB subtract
- FTRC truncate and convert to integer
- FTRV transform vector
Tutorial Outline

- Why the SH-4 for DSP?
- History, Roadmap
- SH-4 Microprocessor core
- Summary of instruction set
- Floating point instructions
- System peripherals
- FIR, LMS, FFT filter examples and optimization techniques
- SH-4 VoIP implementation and tools

Sixteen Bit Instruction Enhances System Performance

- Store twice the number of instructions
  - On chip 8kbyte cache holds 4k instructions
  - On chip 8kbyte RAM holds 4k instructions

- Efficient external memory accesses
  - 4 instructions fetched with 64-bit external bus
  - 2 instructions fetched with 32-bit external bus

- Reduce CPU bus utilization
  - Cache lines (16 instructions or 32-bytes) are filled quickly
Memory Bus Interface Controller

- Direct connection to SDRAM, SRAM, Burst ROM, PCMCIA (Ver. 2.1) - performs memory refresh
- Configurable 64/32/16/8 - bit external bus
- Seven configurable memory areas
  - Memory area divided into 64Mbyte per chip select pin
  - Configurable bus width
  - Configurable wait states
  - Configurable as SDRAM, SRAM, Burst ROM or PCMCIA
- Big or little endian

Four Channel DMAC

- 8, 16, 32-bit or 32-byte transfer size
- Dual or single address mode
- Transfer request initiated by
  - External source
  - On-chip module
  - Time interval using on-chip timer
- Cycle steal or burst mode
Interrupt Controller/Exception Handler

- Five external interrupt pins with 15-level priority
- Set individual priority level for on-chip peripherals
- Bank registers enable fast context switching
  - switch between bank registers

Interrupt Response Time

- /NMI & Peripheral latency = 7 clocks min
- IRL latency = 8 clocks min

<Diagram showing interrupt response time>
Serial Debug (JTAG)

- Download code
- Read/write registers and memory
- Set breakpoints

Advanced Debug Capabilities

- Real time trace output of branch source and destination addresses
void main(void)
{
    long a[10];
    long j;
    int i, min, max;
    for (i = 0; i < 10; i++)
    {
        j = rand();
        if (j < 0)
        {
            j = -j;
        }
        a[i] = j;
    }
}

E10A Emulator / HDI Trace Output

void main(void)
{
    long a[10];
    long j;
    int i, min, max;
    for (i = 0; i < 10; i++)
    {
        j = rand();
        if (j < 0)
        {
            j = -j;
        }
        a[i] = j;
    }
}

Clock Pulse Generator

- Main clock can be set at 1/2, 1, 3 or 6 times the external clock
- CPU, external bus, internal peripheral bus can be set independently
- Maximum frequency
  - CPU = 200MHz
  - External bus = 100MHz
  - Internal peripheral bus = 50MHz
Power down modes

- Sleep - CPU off, peripherals on
- Standby - CPU off, peripherals off
- Module Standby - CPU on, peripherals on

Other modules

- PCI
- Memory management unit
- H/W break control for debug
- Timers
- Real-time clock
- Serial communication interface
- Serial debug interface - JTAG compliant
- Smart Card interface
- General purpose I/O
SuperH Development Tool Summary

- **Third Party Software**
  - Cygnus, Green Hills, Metrowerks, Hitachi, Diab, WRS

- **Operating Systems**
  - Ariel, VxWorks, Nucleus+, Windows CE, OS-9, SuperTask, ThreadX, GEOS-RTX, CMX-RTX, Ecos, QNX, Linux

- **Hardware Development Tools**
  - Emulators: HP, Orion, Sophia, Applied Microsystems, Lauterbach, Cross Products, Hitachi, EST
  - Evaluation boards: Densan, Hitachi, Sophia

- **Application Support**
  - Architectural Analysis/Uilities: CardTools
  - Peripheral Driver Wizard: Stenkil
  - Telephony Middleware: Vovida, Lucent
  - Co-verification: Hitachi IBIS Simulation Models, Mentor, Cadence, CoWare

---

Wind River Systems

- **Center of Excellence**

- **Tornado v1.01**
  - Integrated development environment
  - Source-level debugger/simulator
  - System- and task-level aware

- **VxWorks**
  - RTOS task management
  - Interrupt and exception handling
  - Network support
  - GNU Compiler Tool Chain
  - DSP aware WinView support

- **E10A support in development**
Integrated Development Environment with DSP and RISC Combined

Evaluate resources used for DSP
DSP is just another task
Real-time DSP Performance Analysis

Tutorial Outline

- Why the SH-4 for DSP?
- History, Roadmap
- SH-4 Microprocessor core
- Summary of instruction set
- Floating point instructions
- System peripherals
- FIR, LMS, FFT filter examples and optimization techniques
- SH-4 VoIP implementation and tools
Why the SH-4 for DSP processing

- Single Cycle MAC (FMAC)
- Single cycle 4-way dot product (FIPR)
- 4-cycle 4X4 matrix-vector multiplication (FTRV)
- 2-way superscalar
- 32 floating point registers (two banks)
- 32/64-bit load/store
- Delayed branch

FMAC

- FMAC FR0, FRm, FRn

\[ FR0 \times FRm + FRn \rightarrow FRn \]

- Pitch: 1 cycle
- Latency: 3 cycles
- Load can be dispatched in parallel with FMAC
- Close to 1 MAC per cycle can be sustained in real applications
FIPR

- FIPR  FVm, FVn

\[
FR[m] \times FR[n] + FR[m+1] \times FR[n+1] + FR[m+2] \times FR[n+2] + FR[m+3] \times FR[n+3] \rightarrow FR[n+3]
\]

- Pitch: 1 cycle
- Latency: 4 cycles
- 4 MACs/cycle
- 4-way dot-product

FTRV

- FTRV  XMTRX, FVn

\[
\]

- Pitch: 4 cycle
- Latency: 7 cycles
- 4 MACs/cycle
- 4X4 matrix vector multiplication
Floating Point Accelerator

Two 128-bit simultaneous data transfers
Four fmul operations in one cycle
4-input fadd in one cycle

Two-way Superscalar Dispatch

- Dispatches two instructions per cycle
- Instruction classes:
  - Integer/Floating point Load/Store (LS) (MOV, fmov, ...)
  - Floating point execution (FE) (FMAC, FIPR, FTRV, ...)
  - Integer Execution (EX) (ADD, SUB, MUL, ...)
  - Branch (BR) (BSR, BF/S, BF)
  - Compare (MT) (CMP/GE, NOP, ...)

<table>
<thead>
<tr>
<th>MT</th>
<th>EX</th>
<th>BR</th>
<th>LS</th>
<th>FE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
</tr>
</tbody>
</table>
Two-way Superscalar Dispatch

Eight instruction queue drastically reduce pipeline stalls

Two independent decode logic

Instructions execute in parallel

Branch unit (BR)
Integer unit (EX)(MT)
Load/store unit (LS) (MT)
Floating point unit (FE)

Dual Register Banks

- 32 floating point registers: two banks of 16 registers each
- Very useful for software pipelining, coefficient storage
- Single cycle switching between banks (frchg)
- Computations on one bank while load/store on other
64-bit load/store, delayed branch

- Register-pair load/store
- Mitigates load/store bottleneck
- Single cycle switch between 32-bit and 64-bit size (fschg)
- Postincrement
- Delayed branch, instruction can be scheduled along with branch (reduced loop overhead)
- Compare instruction decoupled from branch

SH-4 DSP programs use

- Loop unrolling
- Simultaneous data fetch and computations
- Software pipelining
Programming examples

- **FIR1**: Simple implementation, MAC + 2 loads, ~3 cycles / MAC
- **FIR2**: Outer loop unrolled, MAC + 1 load, ~1 cycle MAC
- **FIR3**: FTRV based, FTRV + 4 adds + 4 loads, ~0.5 cycles/MAC
- **LMS update**: (2 cycles/update)
- **FFT**: (4 cycles/complex butterfly)

---

**FIR I**

- 2 loads + FMAC
- 3-cycles / MAC

```c
for (n=0; n < numPoints; n++){
    y[n] = 0.0;
    for (k=0; k < numTaps; k++){
        y[n] += c[k] * x[n-k];
    }
}
```
FIR I implementation

- 8 cycles
- Load and branch have 2-cycle latency

```
.firm	1:
    fmov.s  @r0+, fr0
    fmov.s @r1+, fr1
    fmac fr0, fr1, fr2
    add #1, r11
    cmp/ge r7, r11
    b/s .firm	2
    nop

    fmov.s fr2, @r6
    add #4, r5
    add #4, r6
    dd #1, r10
    cmp/ge r8, r10
    b/s .firm	3
    nop
```

FIR I implementation: scheduled

- 5-cycles / MAC

```
.firm	1:
    mov r4, r0
    mov r5, r1
    mov #0, r11
    fldi0 fr2

    .firm	2:
    fmov.s @r0+, fr0
    fmov.s @r1+, fr1
    fmac fr0, fr1, fr2
    add #1, r11
    cmp/ge r7, r11
    b/s .firm	1
    nop
```

```
.firm	2:
    nop
    fmov.s fr2, @r6
    add #4, r5
    add #4, r6
    dd #1, r10
    cmp/ge r8, r10
    b/s .firm	3
    nop
```
FIR I implementation: unrolled

- 3-cycles / MAC
- Load bottleneck

```
.LoopOut:
mov    r4, r0
mov    r5, r1
mov    @r0+, fr0
fmov.s fr0, @r1+, fr1
fmac fr1, fr0, fr2
fmov.s fr2, @r6
add    #4, r5
add    #4, r6
add    #1, r10
cmp/ge r8, r10
bf/s   .LoopOut
```

- FR0 bottleneck

```
fmov.s @r0+, fr0
fmac fr0, fr1, fr2
fmov.s fr2, @r6
add    #8, r11
fmov.s fr0, @r1+, fr1
cmp/ge r7, r11
bf/s   .LoopIn
fmac fr1, fr0, fr2
```

FIR II

- Outer loop unrolled, coefficient reuse, 1 load + FMAC
- ~ 1 cycle/MAC

```java
for (n=0; n < numPoints; n+=4){
y[n] = 0.0;
y[n+1] = 0.0;
y[n+2] = 0.0;
y[n+3] = 0.0;
}
for (k=0; k < numTaps; k++){
y[n] += c[k] * x[n-k];
y[n+1] += c[k] * x[n-k+1];
y[n+2] += c[k] * x[n-k+2];
y[n+3] += c[k] * x[n-k+3];
}
```
**FIR II**

- **3-cycles / MAC**  
- **Coefficient in FR0**
- **Reused 4x**

**FIR II Unrolled and Scheduled**

- **1.5-cycles / MAC**  
- **Unrolled and scheduled**
- **Tends to 1 MAC per Cycle with greater unrolling**
FIR III (FTRV)

- 4 loads + FTRV + 4 adds
- ~0.5 cycles /MAC
- Coefficients in XMTRX

```c
firLoop (int offset){
    for (n=offset; n < numPoints; n+=16)/
        y[n] += x[n]*c[0] + x[n-1]*c[1] + x[n-2]*c[2] + x[n-3]*c[3];
    }
}
```

firLoop(offset = 0);
firLoop(offset = 1);
firLoop(offset = 2);
firLoop(offset = 3);

Copyright 2000 Hitachi Semiconductor (America)
FIR III (FTRV)

- FTRV: 4 MACS/cycle
- Coefficient reuse
- 4 FADDs to accumulate partial products

\[
\begin{align*}
\text{mov.s} & \quad @r2+,fr0 \\
\text{mov.s} & \quad @r2+,fr1 \\
\text{mov.s} & \quad @r2+,fr2 \\
\text{mov.s} & \quad @r2+,fr3 \\
\text{ftrv} & \quad xmtrx, fv0 \\
\text{fadd} & \quad fr0, fr6 \\
\text{fadd} & \quad fr1, fr7 \\
\text{fadd} & \quad fr2, fr4 \\
\text{fadd} & \quad fr3, fr5 \\
\text{mov.s} & \quad fr5, @r4 \\
\text{flid0} & \quad fr6 \\
\text{fadd} & \quad fr6, @r4 \\
\text{mov.s} & \quad @r2+,fr12 \\
\text{add} & \quad #16, r4 \\
\text{mov.s} & \quad @r2+,fr13 \\
\text{fadd} & \quad fr0, fr6 \\
\text{mov.s} & \quad @r2+,fr14 \\
\text{fadd} & \quad fr1, fr7 \\
\text{mov.s} & \quad @r2+,fr15 \\
\text{fadd} & \quad fr2, fr4 \\
\text{fadd} & \quad fr3, fr5
\end{align*}
\]

FIR III (FTRV): Scheduled

- Two register sets (FR0-3, FR12-15) to avoid data dependencies
- Software pipelining
- ~ 0.5 cycles/MAC

\[
\begin{align*}
\text{ftrv} & \quad xmtrx, fv0 \\
\text{mov.s} & \quad fr6, @r4 \\
\text{flid0} & \quad fr6 \\
\text{mov.s} & \quad @r2+,fr12 \\
\text{add} & \quad #16, r4 \\
\text{mov.s} & \quad @r2+,fr13 \\
\text{fadd} & \quad fr0, fr6 \\
\text{mov.s} & \quad @r2+,fr14 \\
\text{fadd} & \quad fr1, fr7 \\
\text{mov.s} & \quad @r2+,fr15 \\
\text{fadd} & \quad fr2, fr4 \\
\text{fadd} & \quad fr3, fr5
\end{align*}
\]
**FFT**

- Uses FTRV
- 4 cycles/complex butterfly

A single DIF butterfly:

1024-point, radix-2 FFT (35.4k cycles)

\[
\begin{array}{c}
\text{In}_1 \\
\text{In}_2 \\
\text{W}=a+jb \\
\text{Out}_1 \\
\text{Out}_2
\end{array}
\]

\[
\begin{bmatrix}
1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 \\
a & -b & -a & b \\
b & a & -b & -a
\end{bmatrix}
\begin{bmatrix}
\text{In}_1_r \\
\text{In}_1_i \\
\text{In}_2_r \\
\text{In}_2_i
\end{bmatrix}
\rightarrow
\begin{bmatrix}
\text{Out}_1_r \\
\text{Out}_1_i \\
\text{Out}_2_r \\
\text{Out}_2_i
\end{bmatrix}
\]

**LMS Update**

- Uses FMAC
- ~2 cycles per LMS update
- Dual-load/stores help alleviate load/store bottleneck

```c
y = 0.0;
for (k=0; k < numTaps; k++){
    y += c[k] * x[n-k];
}
error = yref - y;
scale = error * gamma;
for (k=0; k < numTaps; k++)
    c[k] += scale * x[k];
```
LMS Update

- Load/Store intensive
- Double-width load/store
- Ping pong register files

```
fmov   @R1+,XD8
fmov   @R2+,XD12
fmov   @R2+,XD14
fmov   @R1+,XD10

fmac   FR0,FR10,FR14
fmac   FR0,FR11,FR15
fmac   FR0,FR8,FR12
fmac   FR0,FR9,FR13

fmov   DR14,@-R2
fmov   DR12,@-R2
frchg
```

LMS Update (Scheduled)

- ~ 2 cycles/update

```
fmac   FR0,FR10,FR14
fmov   @R2+,XD12
fmac   FR0,FR11,FR15
fmov   @R2+,XD14
fmac   FR0,FR8,FR12
add    #16, R2
fmac   FR0,FR9,FR13
fmov   @R1+,XD8
fmov   DR14,@-R2
dt     R0
fmov   @R1+,XD10
fmov   DR12, @-R2
frchg
bl/s   loop2
add    #32, R2
```
VoIP Performance

- **Speech Coding Algorithms**
  - ITU-T G.723.1 (5.3/6.3 kbps) 25 MHz 73K
  - ITU-T G.711 (64 kbps) 1 MHz 3K
  - ITU-T G.729A Annex C (8kbps) 17.5 MHz 39K
  - ITU-T G.728 (16/12.8/9.6 kbps) 36 MHz 30K
  - ITU-T G.726 (40/32/16/8 kbps) 11 MHz 32K
  - ITU-T G.729 Annex C (8kbps) 25 MHz 40K
  - ITU-T G.729E (11.8 kbps) 58 MHz 54K

- **Telephony modules**
  - DTMF Gen and Det. - ITU Q.23 1 MHz 9K
  - Line Echo Canceller - ITU-T G.168 < 7 MHz tail length of 8 ms 6K
  - Caller ID Gen. - ITU V.23, Bellcore GR-30-CORE NA 4.5 K
  - Call Progress Tones Gen. - ITU X.96 1 MHz 4.5K

SH-4 as a DSP

- Single cycle MAC
- FIPR/FTRV : > 2 MACS/cycle in real applications
- 64-bit load/store bandwidth (superscalar dispatch with MAC)
- Reduced Loop Overhead (delayed branch + superscalar)
- Large register file for efficient loop unrolling and software pipelining
- Superscalar dispatch of non-MAC operations
Tutorial Outline

- Why the SH-4 for DSP?
- History, Roadmap
- SH-4 Microprocessor core
- Summary of instruction set
- Floating point instructions
- System peripherals
- FIR, LMS, FFT filter examples and optimization techniques
- SH-4 VoIP implementation and tools

VoIP Simplified

*Transporting Voice information within IP packets*

- Analog to Digital
  - Sample voice and digitize
- Digital Signal Processing
  - Compress voice
  - Line echo cancellation and VAD
- Packetize
  - Assemble compressed voice sample into frames
  - Insert frames into IP packet
Creating IP Packets (1)

- **Step 1**: An analog voice signal is received
- **Step 2**: The signals are converted to a Pulse Code Modulation (PCM) digital stream (16 bits every 125 µs) and filtered for line echo
- **Step 3**: PCM stream is analyzed for silence suppression and presence of tone
- **Step 4**: PCM samples are assembled into frames and compressed with vocoder.
  - G.729a creates 10 ms long frame with 10 bytes of speech
  - It compresses PCM stream to 8 kbps
Creating IP Packets (2)

- **Step 5:** Frames are converted into IP packets
  - first RTP packet with 12 byte header
  - then 8 byte UDP with source & destination socket
  - finally 20 byte IP header containing source & destination gateway’s IP addresses are added

- **Step 6:** Then, the packet is sent onto the Internet where Routers and Switches examine the destination IP address, route it properly and deliver it to the destination
  - Internet routing process may take several nodes and jumps from network to network

- **Step 7:** When destination receives the packet, it goes through the inverse process for playback

---

Comparison of Circuit switched PSTN and Internet (packet data network)

- **PSTN**
  - Dedicated path, Large Bandwidth, Guaranteed delivery
  - No IP address, No extra computing, No sharing of bandwidth

- **IP Network**
  - No Dedicated path, No Bandwidth assignment, No Guaranteed delivery
  - IP address, Extra computing, Complete sharing of bandwidth
Who Needs VoIP Service?

*VoIP enables low-cost alternative to long-distance PSTN*

- **Consumer**
  - Cable Modem
  - xDSL
- **SOHO**
  - Cable Modem
  - xDSL
- **Enterprise**
  - LAN
  - PBX
  - Residential Gateway
  - Gateway
  - Gateway
  - PBX, IP Phone
  - PBX, IP Phone

Residential Gateway

- Manages
  - Home LAN
  - WAN interface for the last mile
  - Appliances

Based on Cisco White paper: “Networking Technologies Incorporated in the Cisco Networks Product Development Kit”
Traditional VoIP Implementations Require a Separate DSP and CPU

- DSPs efficiently execute telephony middleware tasks while RISCs efficiently execute control tasks

DSP-Intensive Tasks
- Voice Compression/Decompression
- Tone Detection/Generation
- Echo Cancellation
- Silence Suppression
- DTMF
- Other middleware

Other Tasks - RISC
- Telephony Protocols
- Network Protocols
- Management
- Routing
- RTOS

SH-4 Client Telephony VoIP Solution

- Floating Point Unit combined with two-way superscalar architecture performs DSP-intensive tasks

Faster Time-to-Market
- Middleware (in development)
- Multitasking vs. multiprocessing
- Excellent debug environment
- JTAG

Lower Cost Solution
- One-chip vs. two
- One memory subsystem vs. two
- Direct connect to SDRAM

Lower risk
- Proven solution, reference platform
Hitachi VoIP Reference Platform

- Comprehensive VoIP middleware and roadmap to optimization and support for new speech compression standards
- Processors ideal for VoIP with roadmaps to higher speed and higher integration
- Strategic alliances with networking and telephony industry leaders

SH-4 VoIP Reference Design (2Q/00)
**VoIP Middleware**

- **Speech Coding Algorithms**
  - ITU-T G.723.1 (5.3/6.3 kbps)  25 MHz  73K
  - ITU-T G.711 (64 kbps)  1 MHz  3K
  - ITU-T G.729A Annex C (8kbps)  17.5 MHz  39K
  - ITU-T G.728 (16/12.8/9.6 kbps)  36 MHz  30K
  - ITU-T G.726 (40/32/16/8 kbps)  11 MHz  32K
  - ITU-T G.729 Annex C (8kbps)  25 MHz  40K
  - ITU-T G.729E (11.8 kbps)  58 MHz  54K

- **Telephony modules**
  - DTMF Gen and Det. - ITU Q.23  1 MHz  9K
  - Line Echo Canceller - ITU-T G.168 < 7 MHz  (tail length of 8 ms)  6 K
  - Caller ID Gen. - ITU V.23, Bellcore GR-30-CORE  NA  4.5 K
  - Call Progress Tones Gen. - ITU X.96  1 MHz  4.5 K

---

**How do you determine how many VoIP channels can be supported?**

<table>
<thead>
<tr>
<th></th>
<th>All G.711</th>
<th>2<em>711/ 2</em>729</th>
<th>All G.729</th>
</tr>
</thead>
<tbody>
<tr>
<td>G.711 + other telephony</td>
<td>1 + 14</td>
<td>25 + 14</td>
<td>25 + 14</td>
</tr>
<tr>
<td># of Channels</td>
<td>10</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>MHz Required</td>
<td>150</td>
<td>60</td>
<td>156</td>
</tr>
<tr>
<td>G.729 + other telephony</td>
<td>25 + 14</td>
<td>25 + 14</td>
<td>25 + 14</td>
</tr>
<tr>
<td># of Channels</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>MHz Required</td>
<td>78</td>
<td>156</td>
<td></td>
</tr>
<tr>
<td>RTOS</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Network, Bridging, Control Code</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td><strong>Total MHz (SH-4 is 200MHz)</strong></td>
<td><strong>180 MHz</strong></td>
<td><strong>188 MHz</strong></td>
<td><strong>186 MHz</strong></td>
</tr>
</tbody>
</table>

Other telephony - DTMF, VAD, Call Progress Tones, LEC
SH-4 for Client Telephony

Simplified Product Development and Faster Time to Market

- Multitasking instead of multiprocessing
  - Single instruction stream
- Eliminates inter-processor communication
  - DSP algorithms treated like any other task
  - Simple programming environment
- Single hardware and software design environment
  - C compiler
  - Advanced tools from Hitachi and third party developers such as WindRiver Systems (Tornado)
- Low Power for power sensitive applications
  - 200MHz SH7750: 900mW
  - 167MHz SH7751: 400mW

Announced SH based Cable Modems

- Samsung Telecommunications
  - SH3-DSP (SH7729) processor in new InfoRanger cable modem (SCM-200R) for data and voice
  - SH3 (SH7709) processor in data-only modems
- E-Tech
  - SH7729 in integrated telephony cable modem, the ICT-100 modem

See our demos at the exhibition hall: # 1118

Thank you for selecting the SH vendor tutorial