HITACHI SuperH

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

HITACHI SuperH

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?

HITACHI SuperH

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

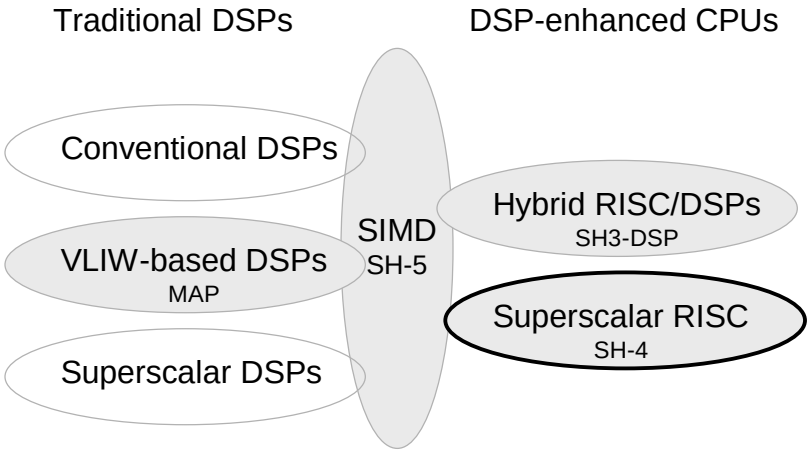
HITACHI SuperH

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

HITACHI SuperH

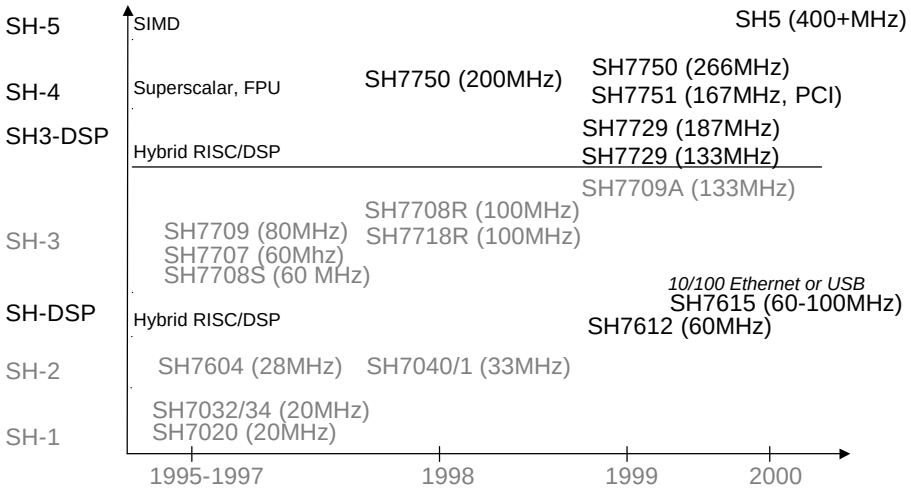
Classes of Processors for DSP



 5

HITACHI SuperH

Hitachi Super RISC Processors for DSP



 6

HITACHI SuperH

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

 7

HITACHI SuperH

SH-4 Device Family

Part Number	HD6417750 PB200M	HD6417750 F167	HD6417750 VF128	HD6417750S VF167	HD6417750R VF200	HD6417750R F266	HD6417751 F167	HD6417751 VF133
Frequency (MHz)	200	167	128	167	200	266	167	133
Performance, x1.8 RISC (MIPS)	360	301	230	301	360	479	301	239
Performance, x 7 FPU (GFLOPS)	1.4	1.2	0.9	1.2	1.4	1.9	1.2	0.9
Core Voltage (3.3V I/O)	1.95V	1.8V	1.5V	1.8V	1.35	1.5V	1.5V	1.35 V
Cache (Kbyte) instruction/data	8/16	8/16	8/16	8/16	16/32	16/32	8/16	8/16
Other Features	-	-	-	-	-	-	PCI	PCI
Power typical	1.9 W	1.25 W	400 mW	400 mW	tbd	tbd	400 mW	240 mW
External Bus	64	64	64	64	64	64	32	32
Package	256 BGA	208-QFP	208-QFP	208-QFP	208-QFP	208-QFP	208-QFP	208-QFP
Process	0.25	0.25	0.25	0.18	0.15	0.15	0.18	0.18
I-Temp (possible) -40 to +85	-	yes	-	yes	-	-	yes	-
Samples	now	now	now	Q2/00	*TBD*	*TBD*	Q2/00	Q2/00
Mass Production	now	now	now	Q3/00	*TBD*	*TBD*	Q3/00	Q3/00

 8

HITACHI SuperH

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

 9

HITACHI SuperH

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

 10

HITACHI SuperH

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

 11

HITACHI SuperH

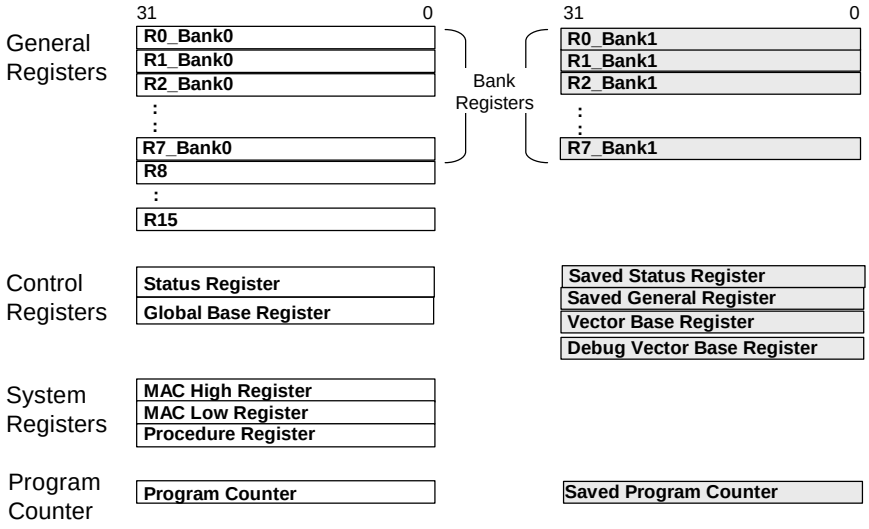
Cache Memory

- 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

 12

HITACHI SuperH

General Registers



 13

HITACHI SuperH

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

 14

HITACHI SuperH

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

HITACHI SuperH

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 is referenced as the branch destination base address instead of VBR

HITACHI SuperH

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

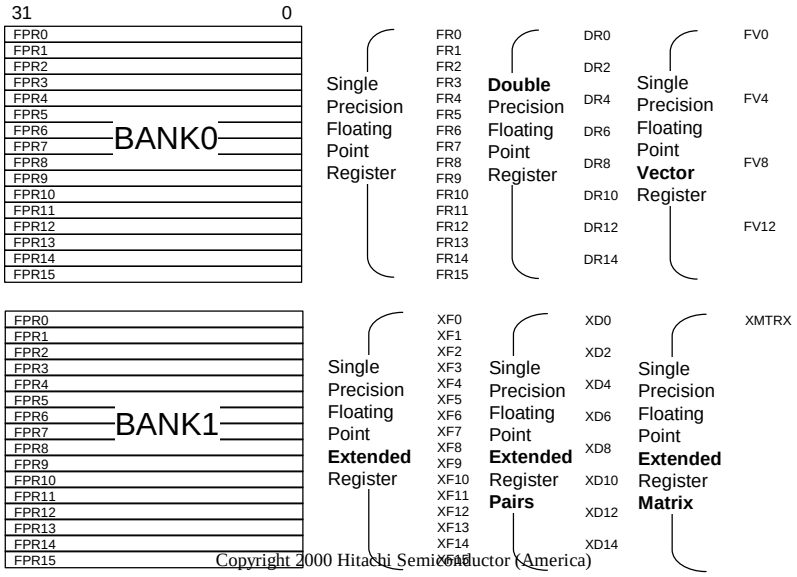
HITACHI SuperH

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

HITACHI SuperH

Floating-point Registers



HITACHI SuperH

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

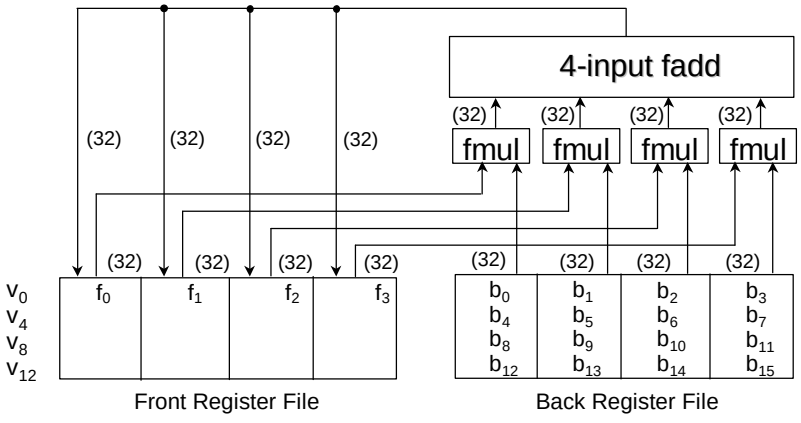
HITACHI SuperH

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

HITACHI SuperH

Floating Point Accelerator



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

HITACHI SuperH

Versatility of the SH-4 Arithmetic Accelerator

3D Graphics Geometry (1.2 GFLOPS)

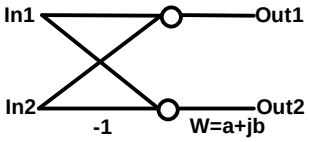
$$\begin{pmatrix} Y1 \\ Y2 \\ Y3 \\ 1 \end{pmatrix} = \begin{pmatrix} c11 & c12 & c13 & c14 \\ c21 & c22 & c23 & c24 \\ c31 & c32 & c33 & c34 \\ c41 & c42 & c43 & c44 \end{pmatrix} \times \begin{pmatrix} x1 \\ x2 \\ x3 \\ 1 \end{pmatrix}$$

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)

$$\begin{pmatrix} Y_i \\ Y_{i+1} \\ Y_{i+2} \\ Y_{i+3} \end{pmatrix} = \underbrace{\begin{pmatrix} x_i & x_{i-1} & x_{i-2} & x_{i-3} \\ x_{i+1} & x_i & x_{i-1} & x_{i-2} \\ x_{i+2} & x_{i+1} & x_i & x_{i-1} \\ x_{i+3} & x_{i+2} & x_{i+1} & x_i \end{pmatrix}}_{\text{Stage 1}} \times \begin{pmatrix} c0 \\ c1 \\ c2 \\ c3 \end{pmatrix} + \ldots + \underbrace{\begin{pmatrix} x_{i-12} & x_{i-13} & x_{i-14} & x_{i-15} \\ x_{i-11} & x_{i-12} & x_{i-13} & x_{i-14} \\ x_{i-10} & x_{i-11} & x_{i-12} & x_{i-13} \\ x_{i-9} & x_{i-10} & x_{i-11} & x_{i-12} \end{pmatrix}}_{\text{Stage 4}} \times \begin{pmatrix} c12 \\ c13 \\ c14 \\ c15 \end{pmatrix}$$

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



A single DIF butterfly

$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ a & -b & -a & b \\ b & a & -b & -a \end{pmatrix} \times \begin{pmatrix} In1\_r \\ In1\_i \\ In2\_r \\ In2\_i \end{pmatrix} = \begin{pmatrix} Out1\_r \\ Out1\_i \\ Out2\_r \\ Out2\_i \end{pmatrix}$$

HITACHI SuperH

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

HITACHI SuperH

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

HITACHI SuperH

SH-4 Basic Pipelines

General Pipeline

I	D	EX	NA	S

General Load/Store Pipeline

I	D	EX	MA	S

Special Pipeline

I	D	SX	NA	S

Special Load/Store Pipeline

I	D	SX	MA	S

Floating-point Pipeline

I	D	F1	F2	FS

Floating-point Extended Pipeline

I	D	F0	F1	F2	FS

HITACHI SuperH

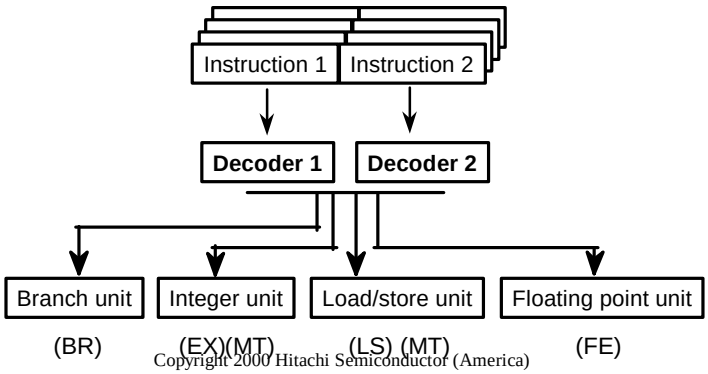
Five Stage Pipeline

- 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

HITACHI SuperH

Two-way Superscalar

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



HITACHI SuperH

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)

HITACHI SuperH

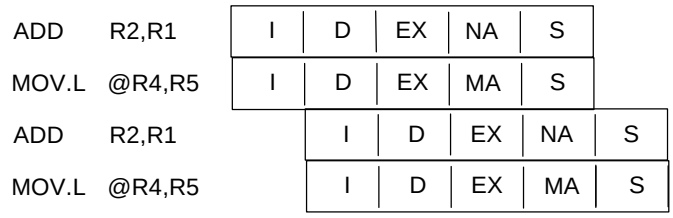
Instruction Combinations Executed in Parallel

		EX	FE	LS	BR	MT
Integer	EX		✓	✓	✓	✓
Floating Point	FE	✓		✓	✓	✓
Load/Store	LS	✓	✓		✓	✓
Branch	BR	✓	✓	✓		✓
Compare	MT	✓	✓	✓	✓	✓
System control	CO					

HITACHI SuperH

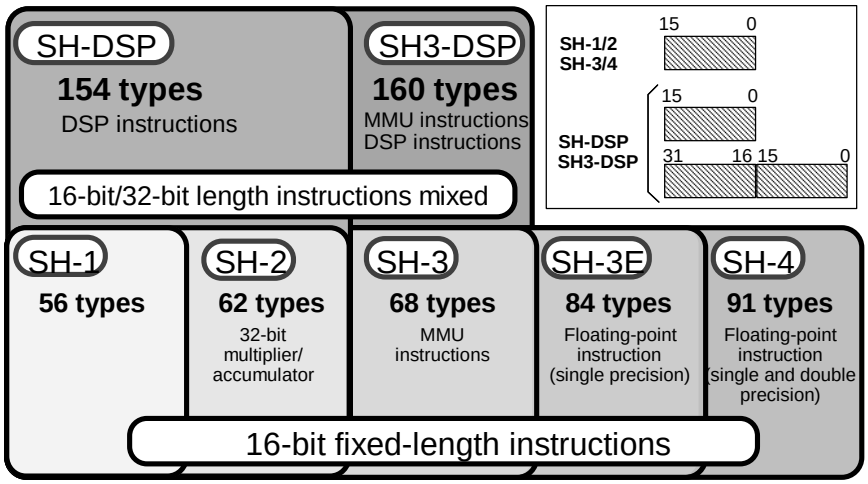
Pipelined Execution

Parallel execution



HITACHI SuperH

SuperH Instruction Set



HITACHI SuperH

SH-4 Instruction Set

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

HITACHI SuperH

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

HITACHI SuperH

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 contstant value beforehand

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

HITACHI SuperH

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)

HITACHI SuperH

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

HITACHI SuperH

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

HITACHI SuperH

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

HITACHI SuperH

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

HITACHI SuperH

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

 41

HITACHI SuperH

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

 42

HITACHI SuperH

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

HITACHI SuperH

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

HITACHI SuperH

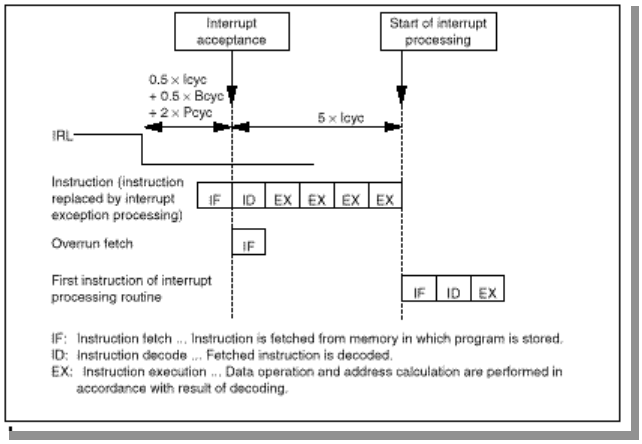
Interrupt Controller/Exception Handler

- 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

HITACHI SuperH

Interrupt Response Time

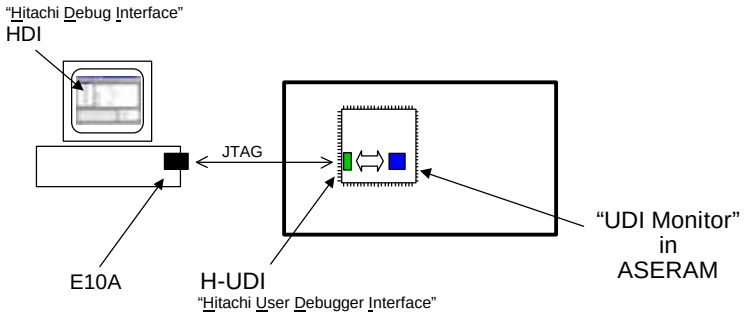
- /NMI & Peripheral latency = 7 clocks min
- IRL latency = 8 clocks min



HITACHI SuperH

Serial Debug (JTAG)

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

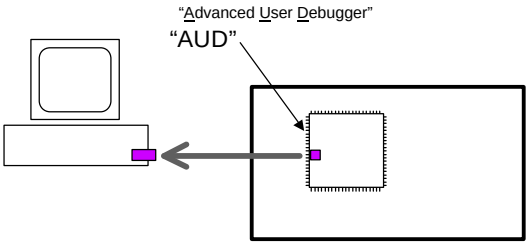


 47

HITACHI SuperH

Advanced Debug Capabilities

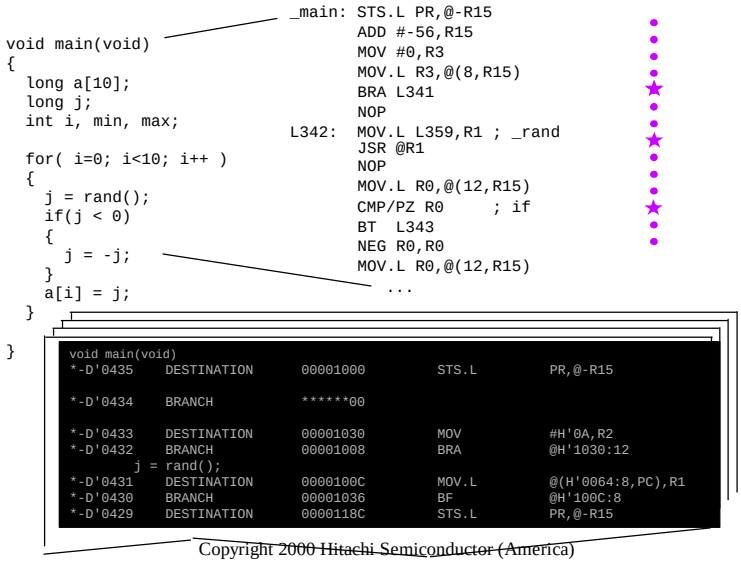
- Real time trace output of branch source and destination addresses



 48

HITACHI SuperH

E10A Emulator / HDI Trace Output



HITACHI SuperH

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

HITACHI SuperH

Power down modes

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

 51

HITACHI SuperH

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

 52

HITACHI SuperH

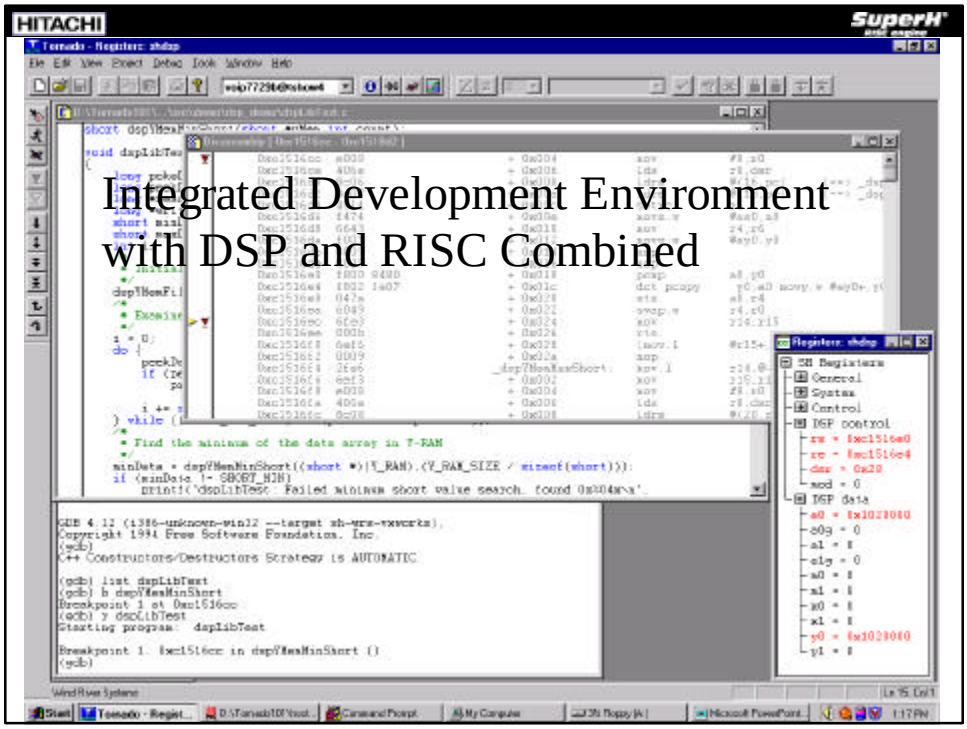
SuperH Development Tool Summary

- 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\Utilities: CardTools
  - Peripheral Driver Wizard: Stenkil
  - Telephony Middleware: Vovida, Lucent
  - Co-verification: Hitachi IBIS Simulation Models, Mentor, Cadence, CoWare

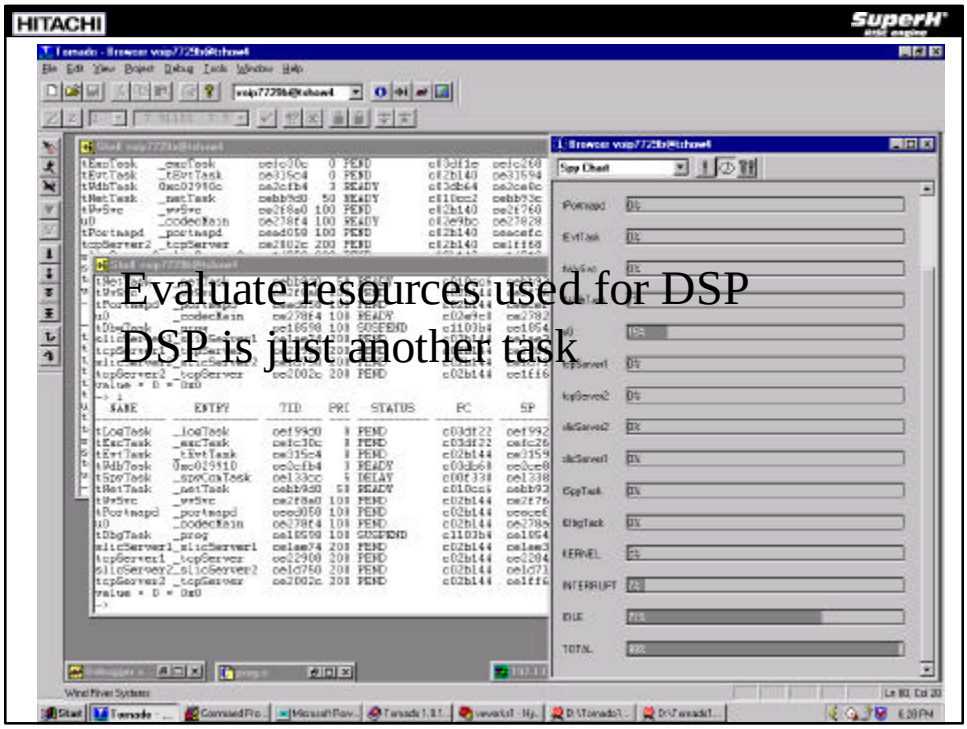
HITACHI SuperH

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

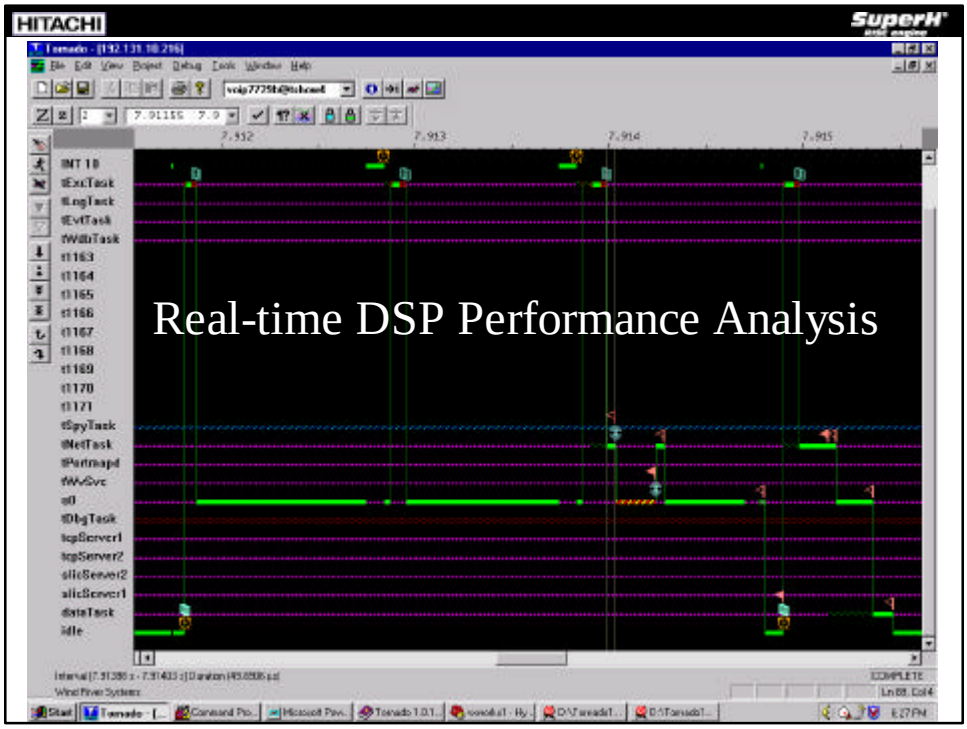


Integrated Development Environment with DSP and RISC Combined



Evaluate resources used for DSP

DSP is just another task



HITACHI SuperH

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

HITACHI SuperH

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

HITACHI SuperH

FMAC

- FMAC FR0, FRm, FRn

  *FR0 * FRm + FRn → 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

HITACHI SuperH

FIPR

- FIPR FVm, FVn

*FR[m] * FR[n] + FR[m+1] * FR[n+1] +*
*FR[m+2] * FR[n+2] + FR[m+3] * FR[n+3] → FR[n+3]*

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

HITACHI SuperH

FTRV

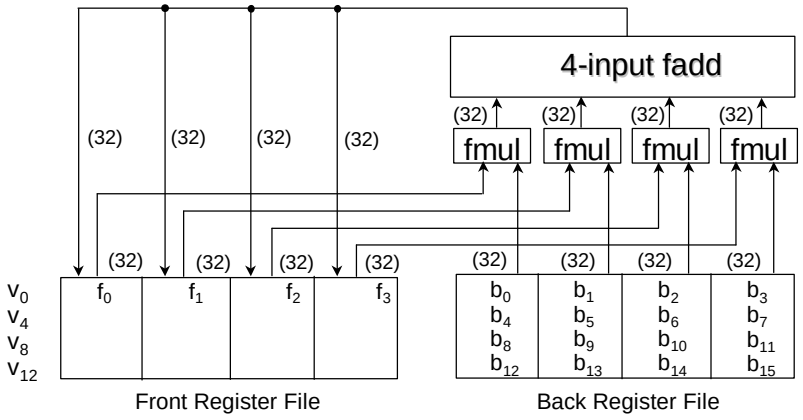
- FTRV XMTRX, FVn

*XF[0]*FR[n] + XF[4]*FR[n+1] + XF[8]*FR[n+2] + XF[12]*FR[n+3] → FR[n]*
*XF[1]*FR[n] + XF[5]*FR[n+1] + XF[9]*FR[n+2] + XF[13]*FR[n+3] → FR[n+1]*
*XF[2]*FR[n] + XF[6]*FR[n+1] + XF[10]*FR[n+2] + XF[14]*FR[n+3] → FR[n+2]*
*XF[3]*FR[n] + XF[7]*FR[n+1] + XF[11]*FR[n+2] + XF[15]*FR[n+3] → FR[n+3]*

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

HITACHI SuperH

Floating Point Accelerator



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

HITACHI SuperH

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, ...`)

	MT	EX	BR	LS	FE
MT	0	X	X	X	X
EX	X	0	X	X	X
BR	X	X	0	X	X
LS	X	X	X	0	X
FE	X	X	X	X	0

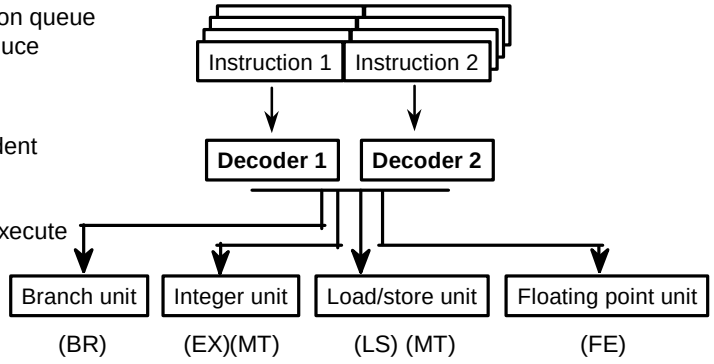
HITACHI SuperH

Two-way Superscalar Dispatch

Eight instruction queue drastically reduce pipeline stalls

Two independent decode logic

Instructions execute in parallel



HITACHI SuperH

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

HITACHI SuperH

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

HITACHI SuperH

SH-4 DSP programs use

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

HITACHI SuperH

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)

HITACHI SuperH

FIR I

- 2 loads + FMAC
- 3-cycles / MAC

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


HITACHI SuperH

FIR I implementation

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

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

.LoopIn:
    fmov.s @r0+, fr0
    fmov.s @r1+, fr1

    fmac  fr0, fr1, fr2

    add   #1,r11
    cmp/ge r7,r11

    bf/s   .LoopIn
    nop

fmov.s fr2, @r6

add   #4, r5
add   #4, r6
dd    #1,r10
cmp/ge r8,r10
bf/s   .LoopOut
nop
```


HITACHI SuperH

FIR I implementation: scheduled

- 5-cycles / MAC

```
LoopOut:
    mov   r4,r0
    mov   r5,r1

    mov   #0,r11
    fldi0   fr2

LoopIn:
    fmov.s  @r0+,fr0
          add #1,r11
    fmov.s  @r1+, fr1
          cmp/ge r7,r11
    bf/s    LoopIn
          fmac fr0,fr1,fr2

nop
fmov.s fr2,@r6

add   #4, r5
add   #4, r6

add   #1,r10
cmp/ge r8,r10
bf/s   .LoopOut
nop
```


HITACHI SuperH

FIR I implementation: unrolled

- 3-cycles / MAC
- Load bottleneck
- FR0 bottleneck

```
.LoopOut:
        mov     r4,r0
        mov     r5,r1

        mov     #0,r11
        fldi0   fr2
.LoopIn:
        fmov.s  @r0+, fr0
        fmov.s  @r1+, fr1
        fmac    fr0, fr1, fr2
        fmov.s  @r0+, fr0
        fmov.s  @r1+, fr1
        fmac    fr0, fr1, fr2
        fmov.s  @r0+, fr0
        fmov.s  @r1+, fr1
        fmac    fr0, fr1, fr2
        fmov.s  @r0+, fr0
        fmov.s  @r1+, fr1
        fmac    fr0, fr1, fr2
        fmov.s  @r0+, fr0
        fmov.s  @r1+, fr1
        fmac    fr0, fr1, fr2
        fmov.s  @r0+, fr0
        fmov.s  @r1+, fr1
        fmac    fr0, fr1, fr2
        fmov.s  @r0+, fr0
        add     #8,r11
        fmov.s  @r1+, fr1
        cmp/ge  r7,r11
        bf/s    .LoopIn
        fmac    fr0, fr1, fr2

        fmov.s  fr2, @r6

        add     #4, r5
        add     #4, r6

        add     #1,r10
        cmp/ge  r8,r10
        bf/s    .LoopOut
        nop
```


HITACHI SuperH

FIR II

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

```
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];
    }
}
```

HITACHI SuperH

FIR II

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

```
.LoopOut:
    mov   r4,r0
    mov   r5,r1

    fldi0    fr5
    fldi0    fr6
    fldi0 fr7
    fldi0    fr8

.LoopIn:
    fmov.s @r0+, fr0
    fmov.s @r1+, fr1
    fmov.s @r1+, fr2
    fmov.s @r1+, fr3
    fmov.s @r1+, fr4
    add    #-12, r1

    fmac  fr0, fr1, fr5
    fmac  fr0, fr2, fr6
    fmac  fr0, fr3, fr7
    fmac  fr0, fr4, fr8

    cmp/ge        r9,r0
    bf/s   .LoopIn
    nop

fmov.s fr8, @-r6
fmov.s fr7, @-r6
fmov.s fr6, @-r6
fmov.s fr5, @-r6
add    #32, r6

add    #16, r5

add    #4,r10
cmp/ge r8,r10
bf/s   .LoopOut
nop
```


HITACHI SuperH

FIR II Unrolled and Scheduled

- 1.5-cycles / MAC
- Unrolled and scheduled
- Tends to 1 MAC per Cycle with greater unrolling

```
.LoopOut:
    mov     r4,r0
    add     #4, r2

    fmov.s  @r1+, fr1
    fldi0   fr5
    fmov.s @r1+, fr2
    fldi0   fr6
    fmov.s @r1+, fr3
    fldi0   fr7
    fmov.s  @r1+, fr4
    fldi0   fr8
    fmov.s  @r0+, fr0
    mov     #0,r11

.LoopIn:
    fmac    fr0, fr1, fr5
            fmov.s      @r2+, fr1
    fmac    fr0, fr2, fr6
            fmov.s @r2+, fr2
    fmac    fr0, fr3, fr7
            fmov.s @r2+, fr3
    fmac    fr0, fr4, fr8
            fmov.s      @r0+, fr0
    add     #-8, r1
            fmov.s      @r2+, fr4

    fmac    fr0, fr1, fr5
            fmov.s      @r1+, fr1
    fmac    fr0, fr2, fr6
            fmov.s      @r1+, fr2
    fmac    fr0, fr3, fr7
            fmov.s      @r1+, fr3
    fmac    fr0, fr4, fr8
            fmov.s      @r0+, fr0
    cmp/ge  r9,r0
            fmov.s      @r1+, fr4

    bf/s    .LoopIn
    add     #-8, r2

fmov.s  fr8, @-r6
add     #4,r10
fmov.s  fr7, @-r6
cmp/ge  r8,r10
fmov.s  fr6, @-r6
add     #16, r5
fmov.s  fr5, @-r6
mov     r5,r1

add     #32, r6

bf/s    .LoopOut
mov     r5,r2

nop
```

HITACHI SuperH

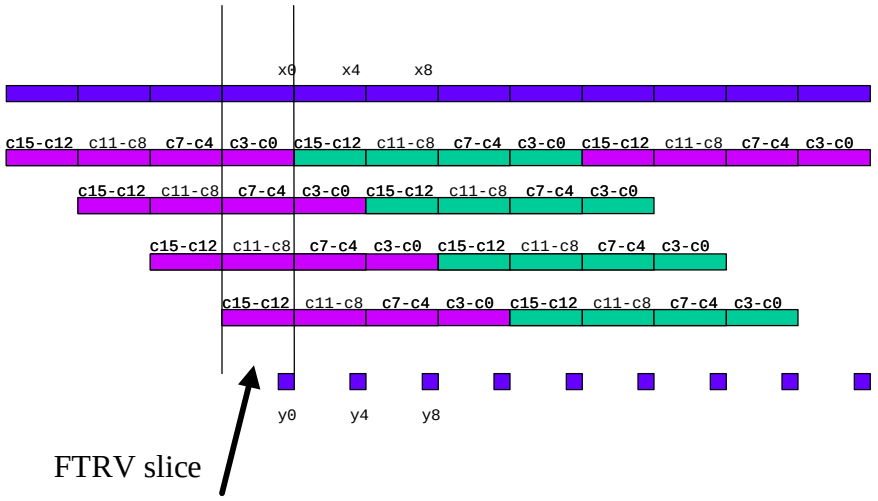
FIR III (FTRV)

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

```
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];
        y[n+4]  += x[n]*c[4]  + x[n-1]*c[5] + x[n-2]*c[6]  + x[n-3]*c[7];
        y[n+8]  += x[n]*c[8]  + x[n-1]*c[9] + x[n-2]*c[10] + x[n-3]*c[11];
        y[n+12] =  x[n]*c[12] + x[n-1]*c[13]+ x[n-2]*c[14] + x[n-3]*c[15];
    }
}

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


HITACHI SuperH

FIR III (FTRV)



HITACHI SuperH

FIR III (FTRV)

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

```
fmov.s      @r2+,fr0
fmov.s      @r2+,fr1
fmov.s      @r2+,fr2
fmov.s      @r2+,fr3

ftrv    xmtrx, fv0

fadd    fr0, fr6
fadd    fr1, fr7
fadd    fr2, fr4
fadd    fr3, fr5

fmov.s fr5, @r4
fldi0  fr5
add    #16, r4
```


HITACHI SuperH

FIR III (FTRV): Scheduled

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

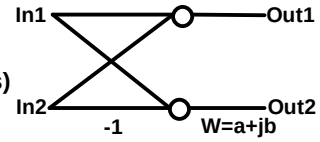
```
ftrv        xmtrx, fv0
    fmov.s  fr6, @r4
fldi0       fr6
    fmov.s  @r2+,fr12
add         #16, r4
    fmov.s  @r2+,fr13

fadd        fr0, fr6
    fmov.s  @r2+,fr14
fadd        fr1, fr7
    fmov.s  @r2+,fr15
fadd        fr2, fr4
fadd        fr3, fr5
```


HITACHI SuperH

FFT

- Uses FTRV
- 4 cycles/ complex butterfly

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



A single DIF butterfly

$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ a & -b & -a & b \\ b & a & -b & -a \end{pmatrix} \times \begin{pmatrix} In1\_r \\ In1\_i \\ In2\_r \\ In2\_i \end{pmatrix} = \begin{pmatrix} Out1\_r \\ Out1\_i \\ Out2\_r \\ Out2\_i \end{pmatrix}$$

HITACHI SuperH

LMS Update

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

```
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];
  }
```


HITACHI SuperH

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
```


HITACHI SuperH

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
bf/s    loop2
        add     #32, R2
```


HITACHI SuperH

VoIP Performance

Speech Coding Algorithms	MHz/channel	Code Size
– 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-C0RE	NA	4.5 K
– Call Progress Tones Gen. - ITU X.96	1 MHz	4.5K

 85

HITACHI SuperH

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

 86

HITACHI SuperH

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

 87

HITACHI SuperH

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

 88

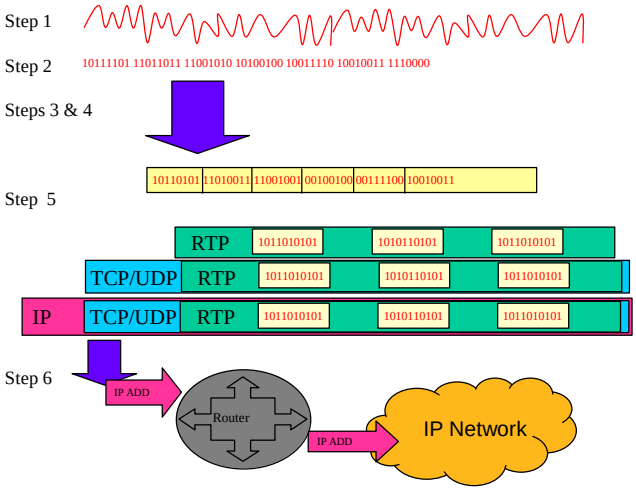
HITACHI SuperH

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

HITACHI SuperH

Creating IP Packets



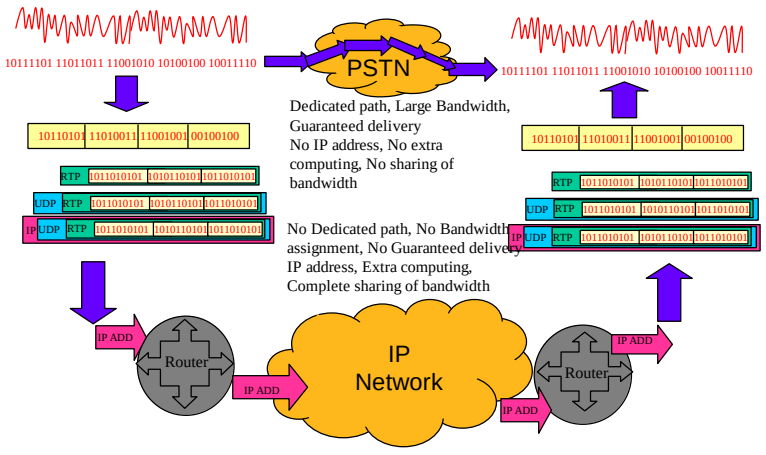
HITACHI SuperH

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

HITACHI SuperH

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



HITACHI SuperH

Who Needs VoIP Service?

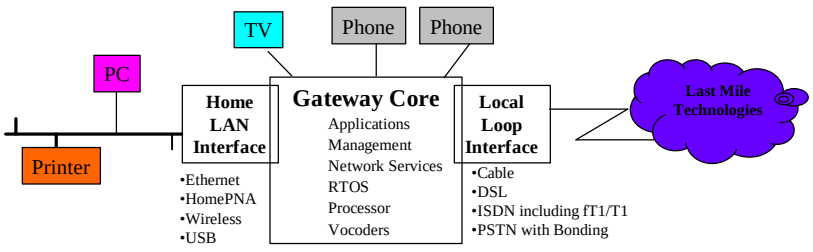
VoIP enables low-cost alternative to long-distance PSTN

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

HITACHI SuperH

Residential Gateway

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

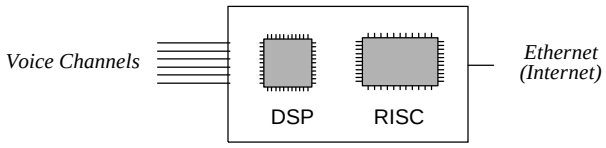


Based on Cisco White paper:
"Networking Technologies Incorporated in the Cisco Networks Product Development Kit"

HITACHI SuperH

Traditional VoIP Implementations Require a Separate DSP and CPU

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

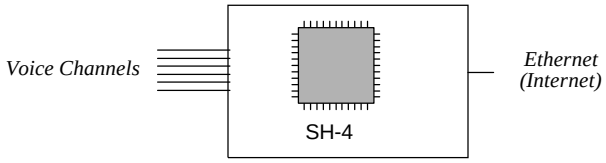


- DSP-Intensive Tasks
  - Voice Compression/Decompression
  - Tone Detection/Generation
  - Echo Cancellation
  - Silence Suppression
  - DTMF
  - Other middleware
- Other Tasks - RISC
  - Telephony Protocols
  - Network Protocols
  - Management
  - Routing
  - RTOS

HITACHI SuperH

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
- Low risk
  - Proven solution, reference platform

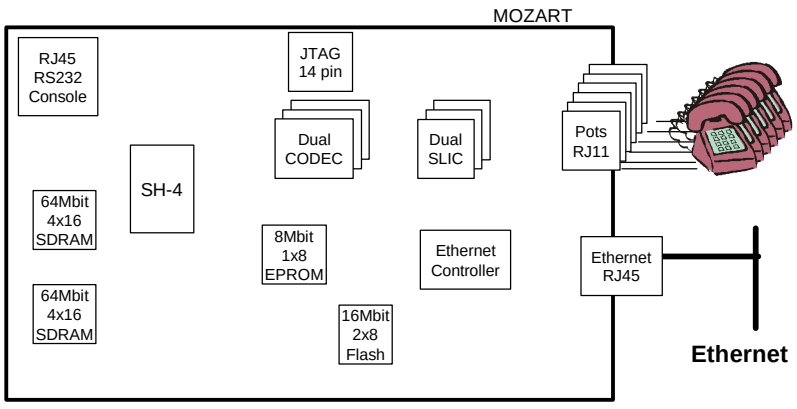
HITACHI SuperH

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

HITACHI SuperH

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



HITACHI SuperH

VoIP Middleware

Speech Coding Algorithms	MHz/channel	Code Size
– 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-C0RE	NA	4.5 K
– Call Progress Tones Gen. - ITU X.96	1 MHz	4.5 K

HITACHI SuperH

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

	All G.711	2*711/ 2*729	All G.729
G.711 + other telephony	1 + 14	1 + 14	
# of Channels	10	4	
MHz Required	***150***	***60***	
G.729 + other telephony		25 + 14	25 + 14
# of Channels		2	4
MHz Required		***78***	***156***
RTOS	**5**	**5**	**5**
Network, Bridging, Control Code	**25**	**25**	**25**
Total MHz (SH-4 is 200MHz)	**180 MHz**	**188 MHz**	**186 MHz**

Other telephony - DTMF, VAD, Call Progress Tones, LEC

HITACHI SuperH

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 Sytems (Tornado)
- Low Power for power sensitive applications
  - 200MHz SH7750: 900mW
  - 167MHz SH7751: 400mW

HITACHI SuperH

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

