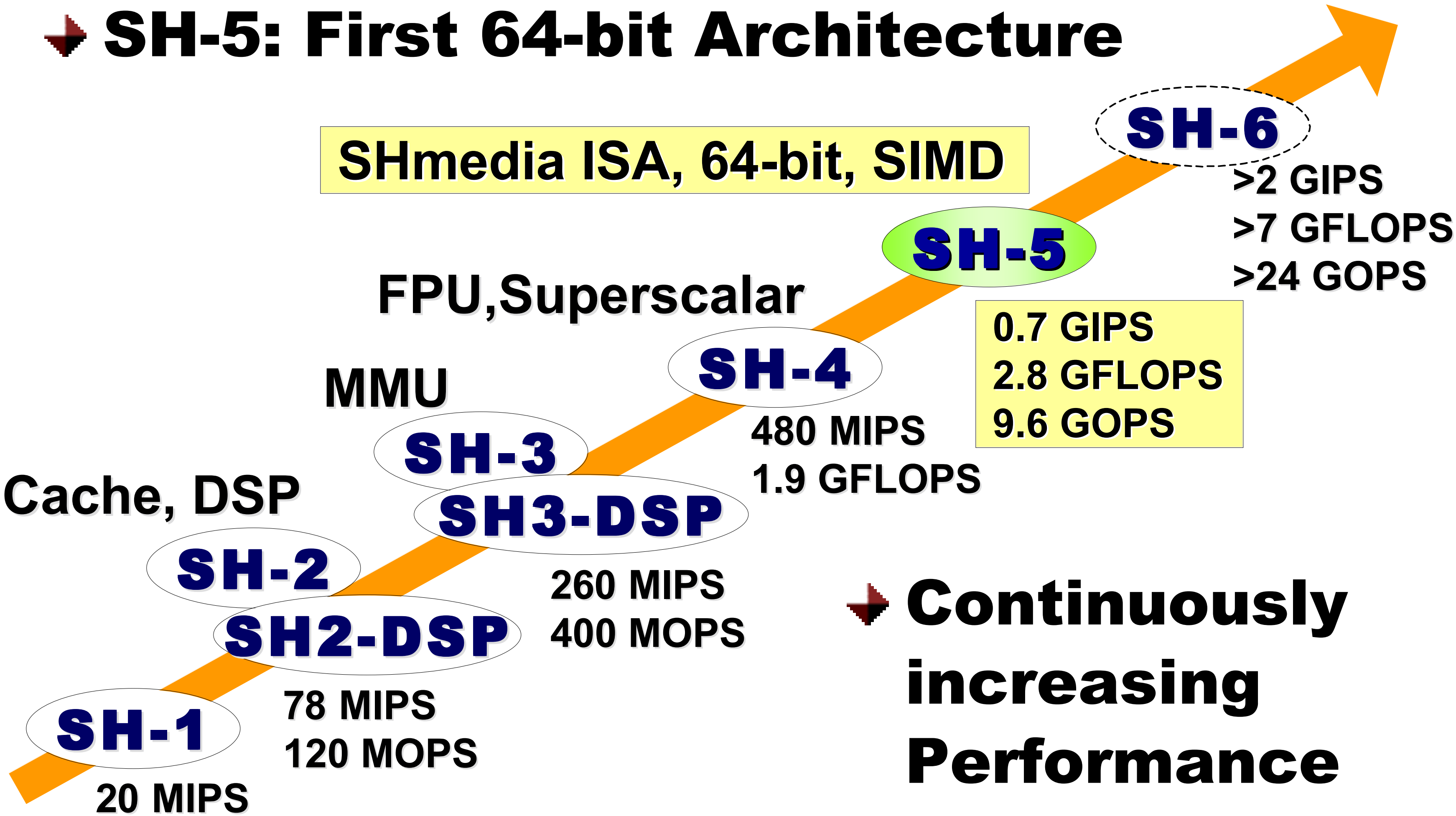
SH-5: A First 64-bit SuperH Core with Multimedia Extension

Fumio Arakawa

Hitachi, Ltd.

SuperH Roadmap

- SH-5: First 64-bit Architecture



- Continuously increasing Performance and integration

SH-5 Target Markets

SuperH™ RISC engine

Consumer Market

Digital Home Appliances

Digital TV, Set-Top-Box
Network

Car Information Systems

Navigation System
Telematics, ITS

Balancing Needs

Low Price

Small die & code size
System-on-chip

Low Power

Low cost package
No-fan system

High Performance

64-bit architecture, SIMD, Vector FPU
7-stage superpipeline

SH-5 Specification

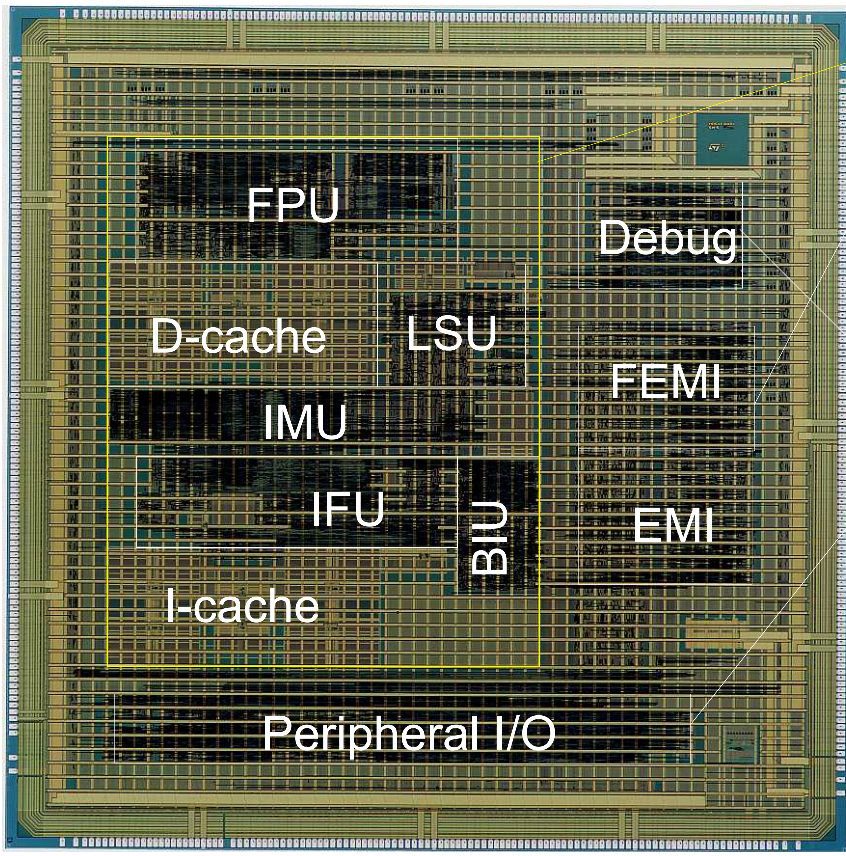
SuperH™ RISC engine

- **Supply Voltage: 1.5 V**
- **Operating Frequency: 400 MHz**
- **Cache: I/D 32/32 KB** (4-way set-associative)
- **TLB: I/D 64-entry** (full-associative)
- **SuperHyway** (Internal Standard Bus)
  - 64-bit, 200 MHz, Split-transaction, 3.2 GB/s
- **Performance**
  - Dhrystone: 714 MIPS (v1.1) and 604 MIPS (v2.1)
  - Peak SIMD: 9.6 GOPS (8 bit) and 1.6 MMACS (16 bit)
  - Peak Floating-point: 2.8 GFLOPS

SH-5 Micrograph

SuperH™ RISC engine

- 1st Cut of SH-5 (Evaluation Chip)



SH5 core

Memory Interface
- EMI (DDR-SDRAM)
- FEMI(SRAM,Flash)

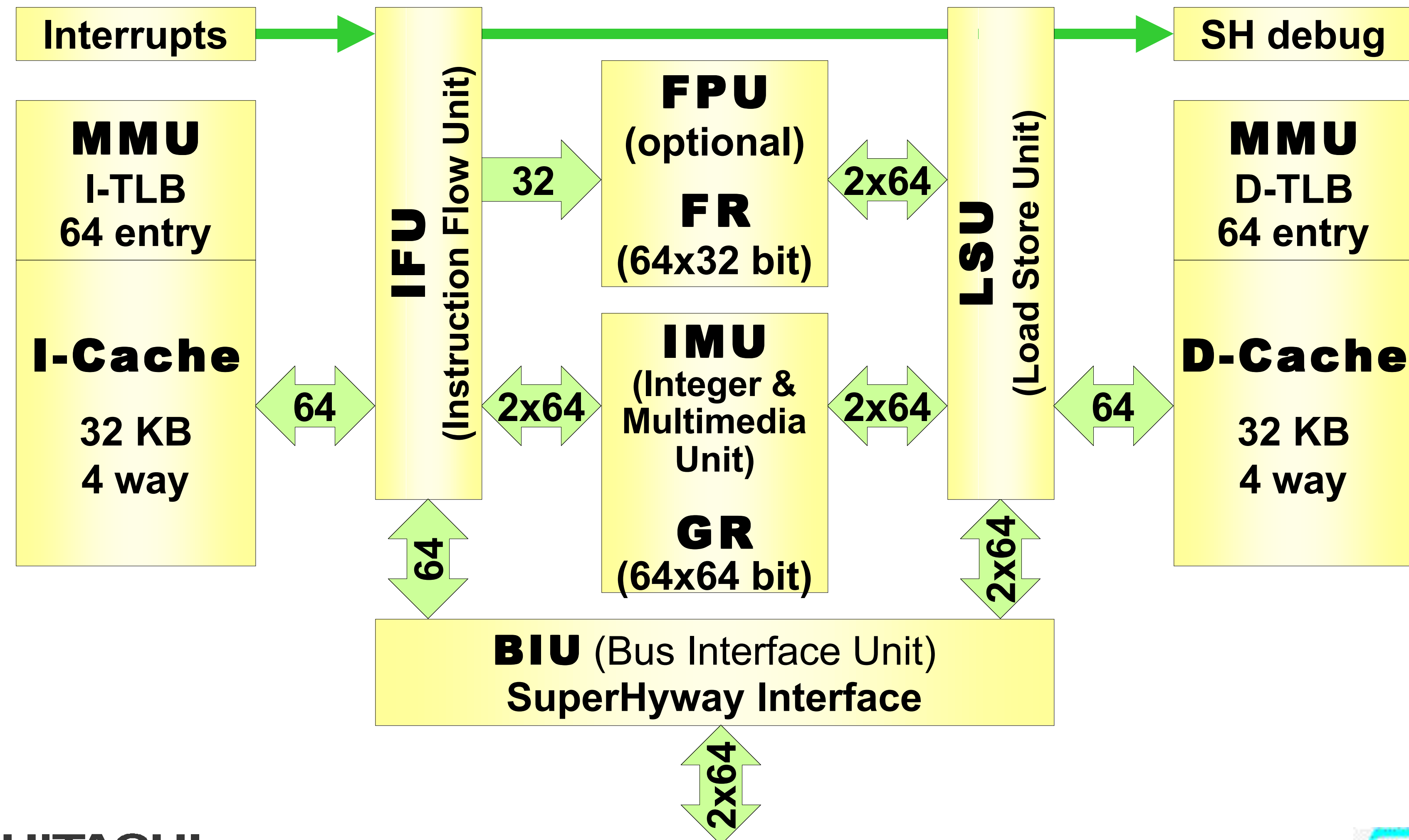
Debug

PCI, Serial, etc.

IFU: Instruction Fetch Unit
IMU: Integer Multimedia Unit
LSU: Load Store Unit
BIU: Bus Interface Unit
EMI: External Memory Interface
FEMI: Flash EMI

SH-5 Core Block Diagram

SuperH™ RISC engine



Superpipeline

- **SH-4**
  - **5-stage pipeline**
- **SH-5**
  - **7-stage pipeline**
  - **x1.5 Higher MHz**
  - **x1.5 Longer Latency is Hidden**
    - Rich register states hide execution time like load latency
    - Split-branch architecture and target preload hide branch latency

	Branch Latency (I-cache + Decode)	**Load Latency** (ALU+ D-cache)	
SH-4	F, D	E1, E2	W
	2 cycles	2 cycles	
SH-5	F1, F2, D	E1, E2, E3	W
	3 cycles	3 cycles	

F,F1,F2: Instruction Fetch; D: Instruction Decode
E1,E2,E3: Execution; W: Write Back

Rich Register States

SuperH™ RISC engine

64 x 64-bit
General-purpose Registers

R63	(zero)
.....	
R19	(T)
R18	(PR)
R17	(MACH,MACL)
R16	(GBR)
.....	
R0	

TR7
.....
TR0

8 x 64-bit
Target Registers

64 x 32-bit
Floating-point Registers

R63
.....
R32 (FPUL)
.....
R0

FPSCR

Floating-point Status and Control Register

64 x 64-bit
Control Registers

CR63	
..... The 46 reserved CRs are not implemented.	
CR0	(SR)

PC

Program Counter

SHcompact Registers mapped on SHmedia Registers

Split Branch Architecture

SuperH™ RISC engine

- **Prepare Target Instructions**
  - **PTA/l Label,TRa** (TRa = &Label)
  - **PTABS/l Rn,TRa** (TRa = Rn)
  - **PTREL/l Rn,TRa** (TRa = Rn+PC)

  l=L/U:likely/unlikely preload is useful

- **Branch Instructions** (Examples)
  - **BLINK TRb,Rd** (Rd=PC+4; PC=TRb)
  - **BEQ/l Rm,Rn,TRc** (if(Rm==Rn) PC=TRc)

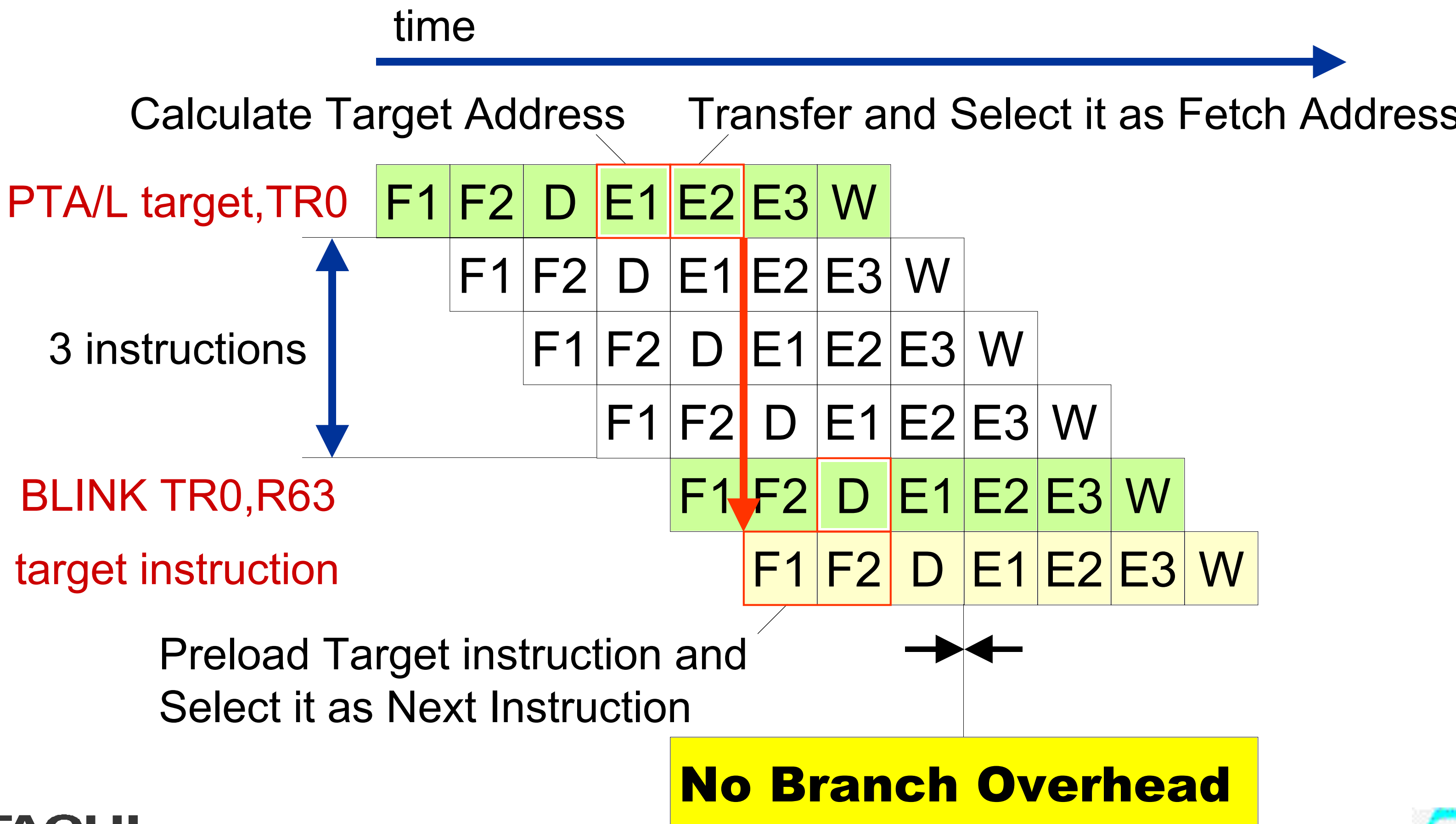
  l=L/U:likely/unlikely taken

  - **Static prediction with likely bit**
  - **Compare and correct prediction miss**

TRa,TRb,TRc: Target Registers; PC: Program Counter
Rm,Rn,Rd: General-purpose Registers

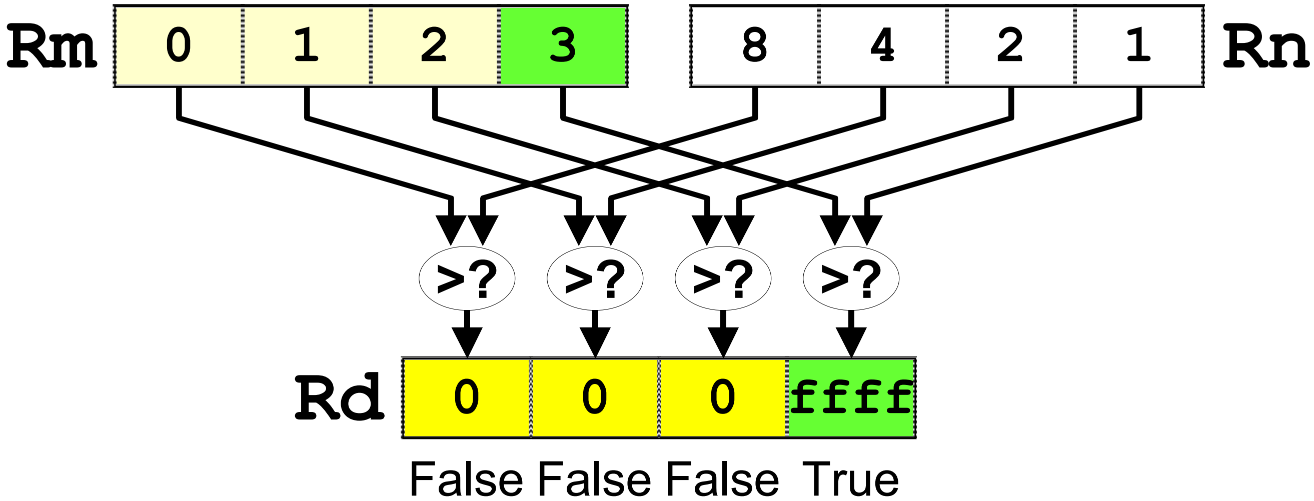
No Branch Overhead

- **In case of three or more instructions between PTA and BLINK**

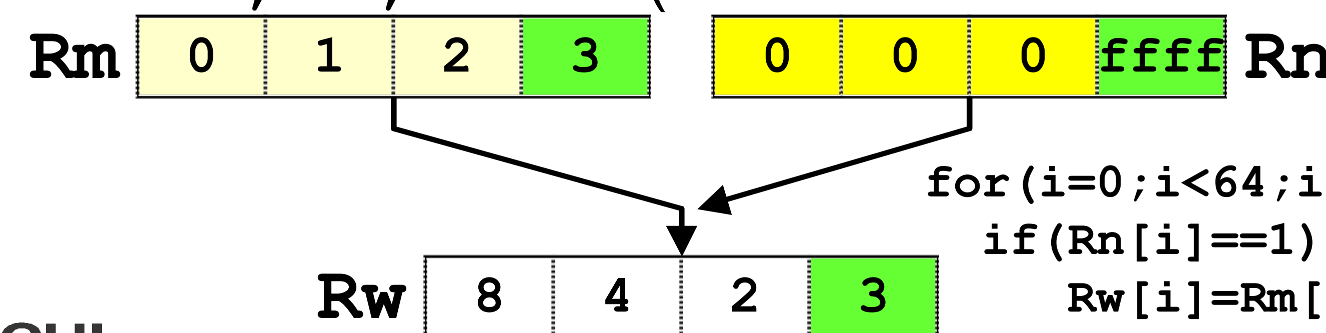


SIMD Instructions

- **MCMPGT.W Rm,Rn,Rd** (Compare)



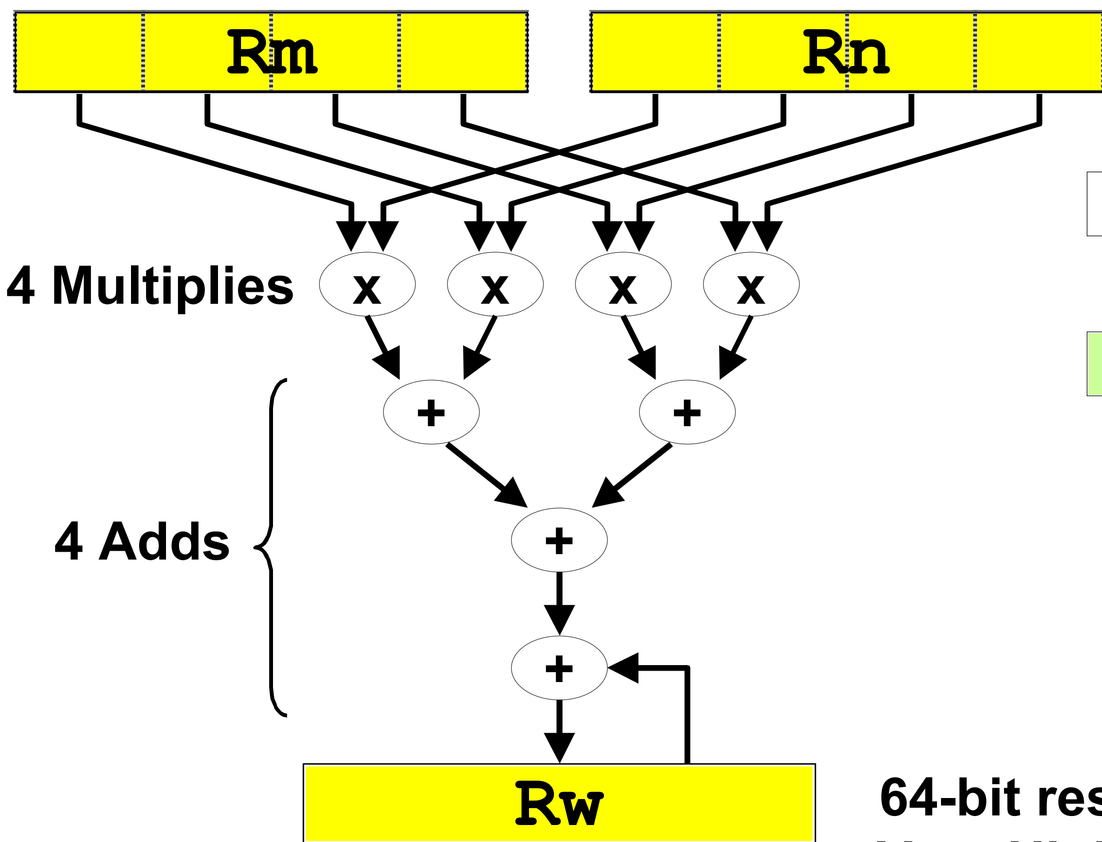
- **MCMV Rm,Rn,Rw** (Bitwise Conditional Move)




```
for(i=0;i<64;i++)
  if(Rn[i]==1)
     Rw[i]=Rm[i]
```

SIMD Instructions (Cont'd)

- `MMULSUM.WQ Rm,Rn,Rw` (Multiply-accumulate)



issued every cycle

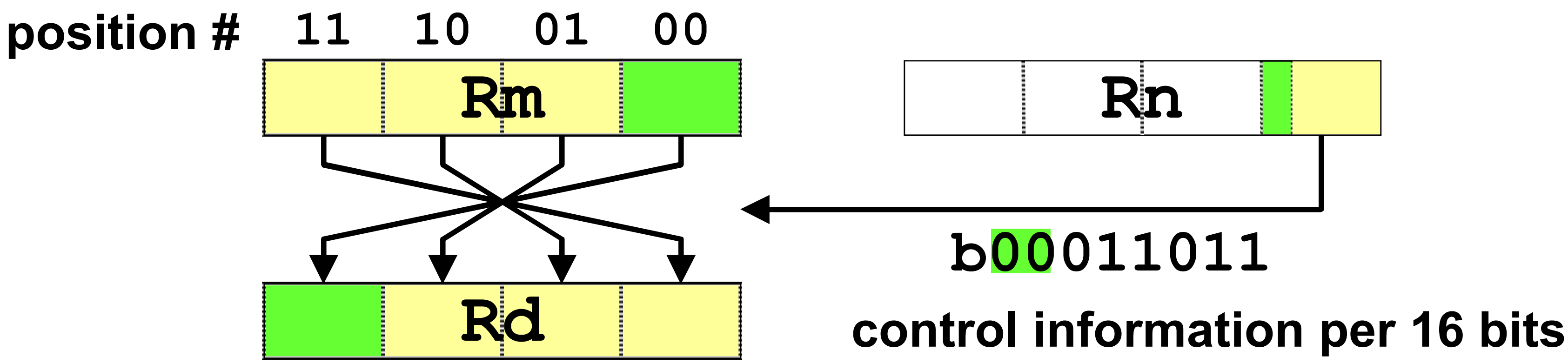
8 operations/cycle

3.2 GOPS
1.6 MMACS
@ 400 MHz

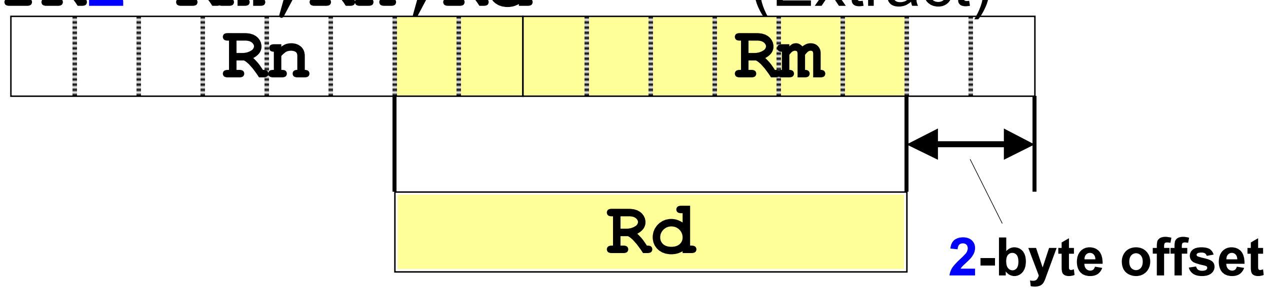
64-bit result:
Very High Accuracy
(No rounding or saturation is necessary)

SIMD Instructions (Cont'd)

- **MPERM.W Rm,Rn,Rd** (Permute)



- **MEXTR2 Rm,Rn,Rd** (Extract)



7 instructions for 1-7 byte offsets

SIMD v.s. Multiple Issues

SuperH™ RISC engine

- **SH-5: 4-way SIMD for 16-bit Data**
  - x4 Peak Performance (Same Operations in Parallel)
  - Data Alignment Overhead Cycles
  - Lower Cost: Simple Control and Small Area Overhead
    - Simple Datapath Division: 64 bits into 4 x 16 bits
- **Reference Design: Multiple Issues**
  - Three Issues w/o Execution Module Duplication
    - Minimizing Area Difference from SIMD
    - 1 Load/Store, 1 Multiplier, etc.
    - Four or more issues are not effective without the duplication.
  - x3 Peak Performance (Different Operations in Parallel)
  - Higher Cost: Complicated Control for Multiple Issues

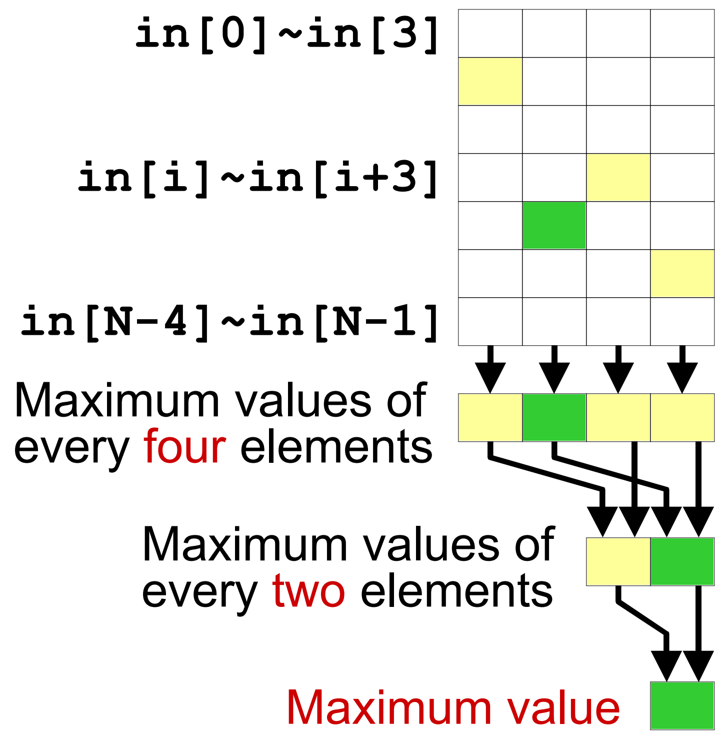
Example: Vector Maximum

- **Find the location and value of the maximum value in a vector**
  - Data Type: 16-bit Fixed Point

Kernel C Source

```
for(i=1;i<N;i++)
 if(maxValue < in[i]){
     maxLocation = i;
     maxValue = in[i];
}
```

- **SIMD Algorithm**
  1. search every four elements
  2. search the four values



Vector Maximum (cont'd)

SuperH™ RISC engine

- **Non-SIMD Code**

  Å(Loop part)
  - 6 instructions/loop
  - Repeat N-1 Times

```
CMPGT  R3,R4,R6
CMVNE  R6,R3,R4
CMVNE  R6,R2,R5
LDX.W  R0,R2,R3
ADDI   R2, 2,R2
BNE    R1,R2,T0
```

- **SIMD Code**

  Å(Loop part)
  - 7 instructions/loop
  - Repeat N/4-1 Times

```
MCMPGT.W R3,R4,R6
ADD      R8,R7,R8
MCMV     R3,R6,R4
MCMV     R8,R6,R5
LDX.Q    R0,R2,R3
ADDI     R2, 8,R2
BNE      R1,R2,T0
```

T0: Loop Top Address
R0: pointer to in
R1: N x2
R2: i x2
R3: in [i]
R4: maxValue
R5: maxLocation (x2)
R6: compare result
R7: 0x04040404
R8: i,i+1,i+2,i+3

Vector Maximum (3 Issues)

SuperH™ RISC engine

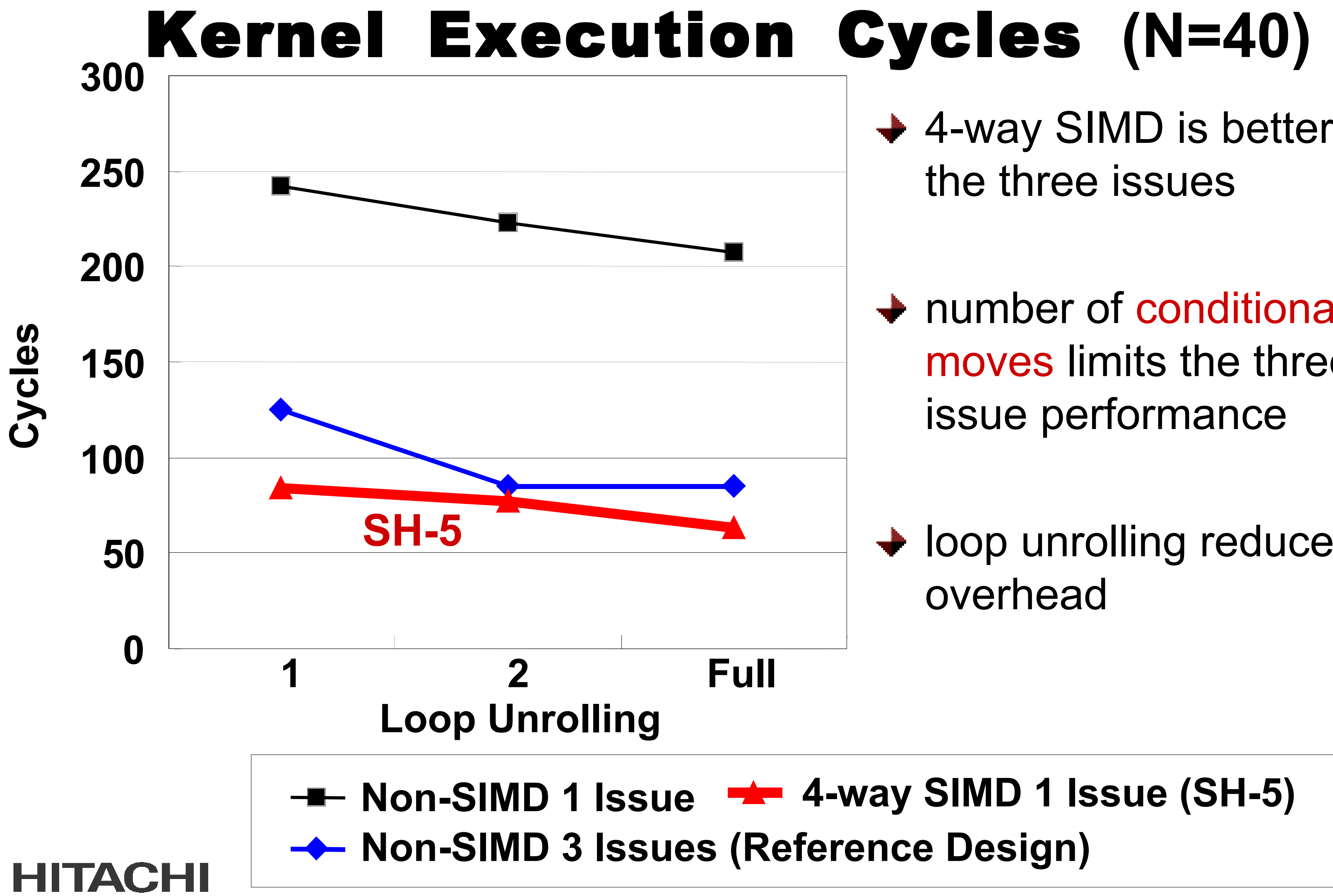
- **Non-SIMD Twice-unrolled Code for Three Issues** (Loop part)
  - 11 instructions/loop, 4 cycles/loop, Repeat N/2-1 times
- **a CMVNE is issued every cycle**
  - Three issues are enough to achieve the best performance (assuming no module duplication)

Issue Slot #1	#2	#3
CMVNE R6,R3,R4;	LDX.W R0,R2,R3;	ADDI R2, 2,R2;
CMVNE R6,R2,R5;	CMPGT R9,R4,R6;	
CMVNE R6,R9,R4;	LDX.W R0,R2,R9;	ADDI R2, 2,R2;
CMVNE R6,R2,R5;	CMPGT R3,R4,R6;	BNE R1,R2,T0;

T0: Loop Top Address	R2: i x2	R5: maxLcation (x2)
R0: pointer to in	R3: in [i]	R6: compare result
R1: N x2	R4: maxValue	R9: in[i] for unrolling

Vector Maximum (Results)

SuperH™ RISC engine



- 4-way SIMD is better than the three issues
- number of conditional moves limits the three-issue performance
- loop unrolling reduces loop overhead

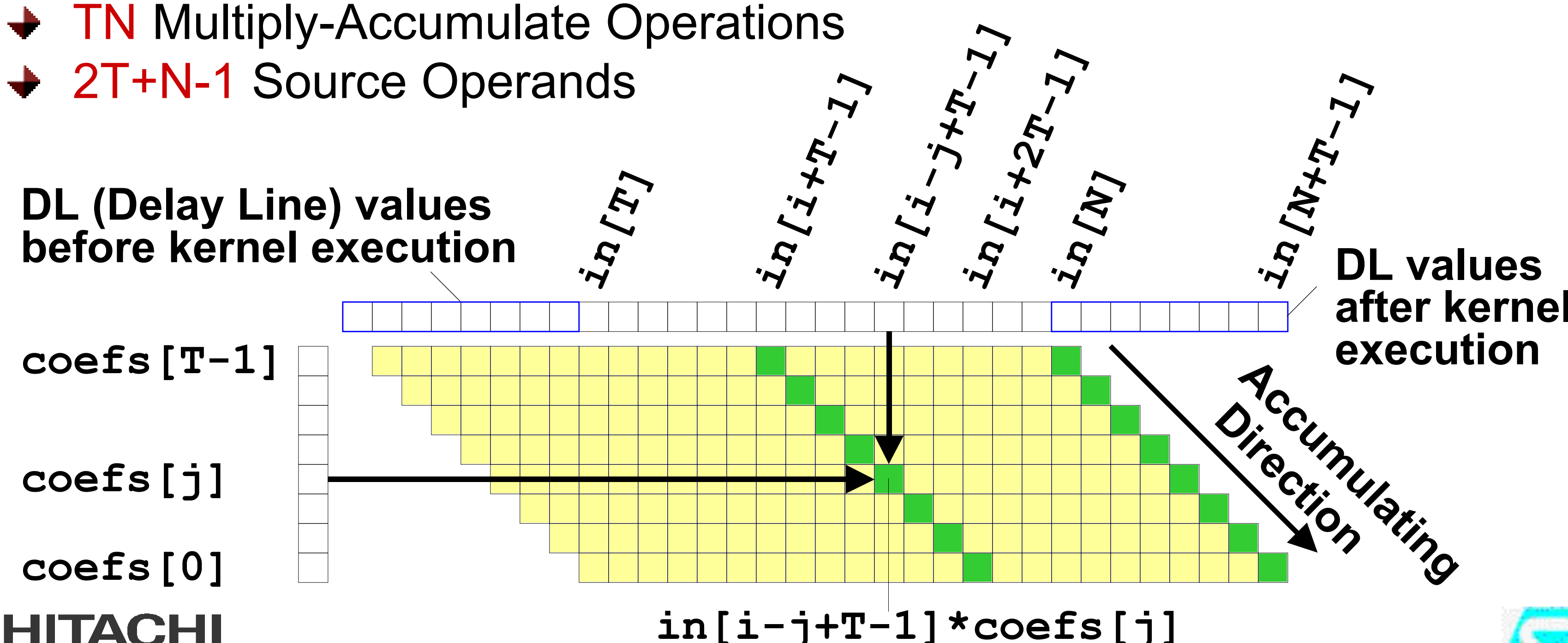
HITACHI Inspire the Next

Example: Real Block FIR

Kernel C Source

```
for(i=0;i<N;i++){
   sum[i]=0;
   for(j=T-1;j>=0;j--)sum[i]+=in[i-j+T-1]*coefs[j];
   if(scaling) sum[i]*=FACTOR;
}/* in[0:T-1]: DL copy, in[T:N+T-1]: new input */
```

- TN Multiply-Accumulate Operations
- 2T+N-1 Source Operands



Real Block FIR (Cont'd)

- **Non-SIMD Code** (Inner Loop part)
  - 6 instructions/loop
  - Repeat TN Times

```
LD.W      R0, 0,R4
LD.W      R1, 0,R6
ADDI      R0,-2,R0
ADDI      R1, 2,R1
MMACFX.WL R4,R6,R10
BNE/L     R0,R2,T0
```

T0: Loop Top Address	R4: coefs
R0: pointer to coefs	R6-R8: in
R1: pointer to in	R10-R13: sum
R2: pointer next to coefs	

- **SIMD Code Unrolled Four Times**
  - 13 instructions/loop
  - Repeat TN/16 times

```
LD.Q       R0, 0,R4
LD.Q       R1, 0,R6
ADDI       R0,-8,R0
ADDI       R1, 8,R1
MMULSUM.WQ R4,R6,R10
MEXTR6     R6,R7,R8
MMULSUM.WQ R4,R8,R11
MEXTR4     R6,R7,R8
MMULSUM.WQ R4,R8,R12
MEXTR2     R6,R7,R8
MMULSUM.WQ R4,R8,R13
ADDI       R6, 0,R7
BNE/L      R0,R2,T0
```

Real Block FIR (3 Issues)

- **Non-SIMD Code Unrolled Four Times for Three Issues**
  - 11 instructions/loop, 4 cycles/loop, Repeat TN/4 times
  - Software pipelining is applied to avoid pipeline stalls.
- **an MMACFX is issued every cycle**
  - Three issues are enough to achieve the best performance (assuming no module duplication)

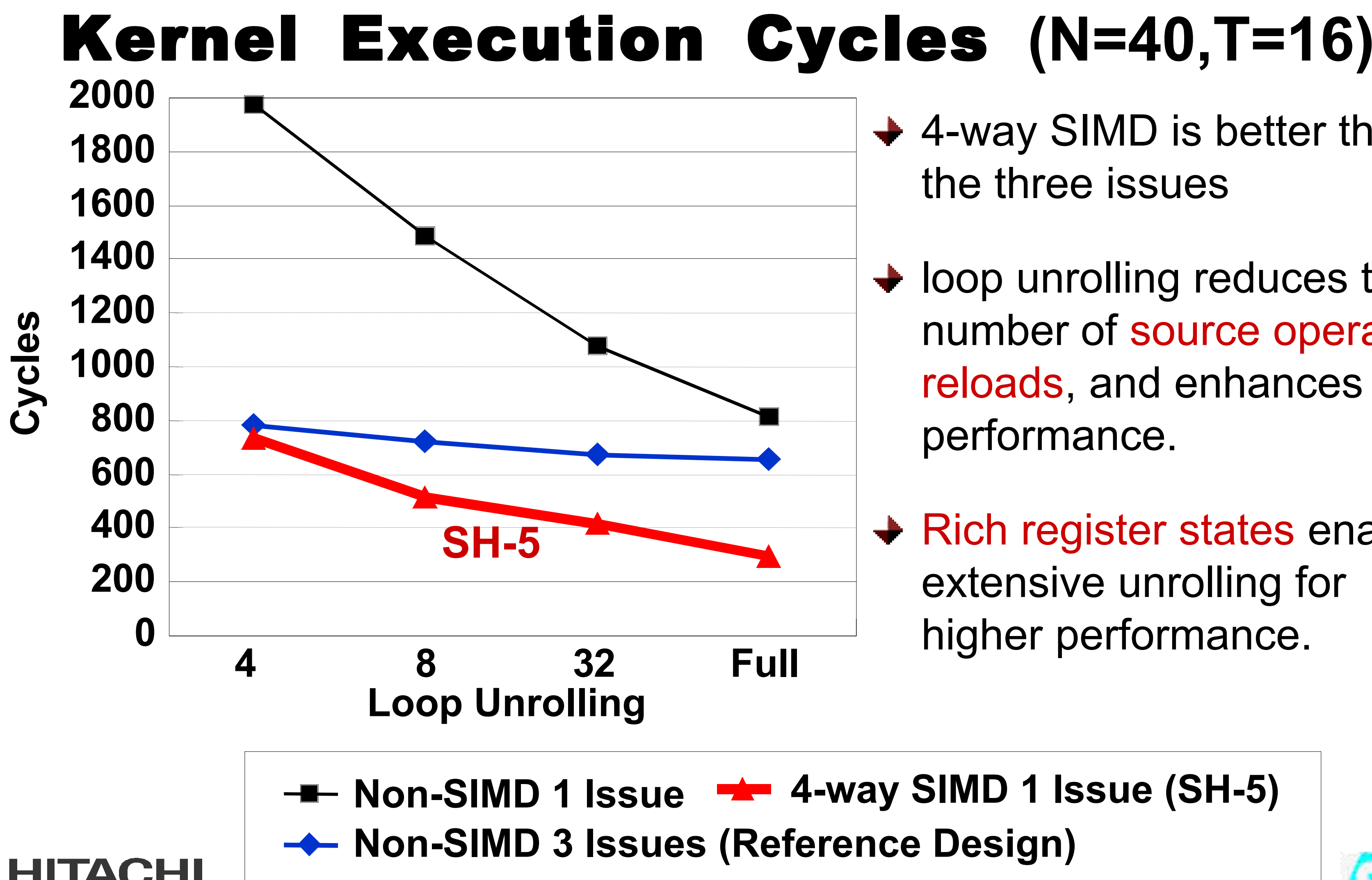
Issue Slot #1	#2	#3
`MMACFX.WL R4,R6,R10;`	`LD.W R0,0,R4;`	`ADDI R1, 4,R0`
`MMACFX.WL R5,R6,R11;`	`LD.W R1,2,R6`	
`MMACFX.WL R5,R7,R10;`	`LD.W R0,2,R5;`	`ADDI R0,-4,R1`
`MMACFX.WL R4,R7,R11;`	`LD.W R1,0,R7;`	`BNE/L R0,R2,T0`

T0: Loop Top Address
R0: pointer to coefs

R1: pointer to in
R2: pointer next to coefs
R4,R5: coefs

R6,R7: in
R10,R11: sum

Real Block FIR (Results)

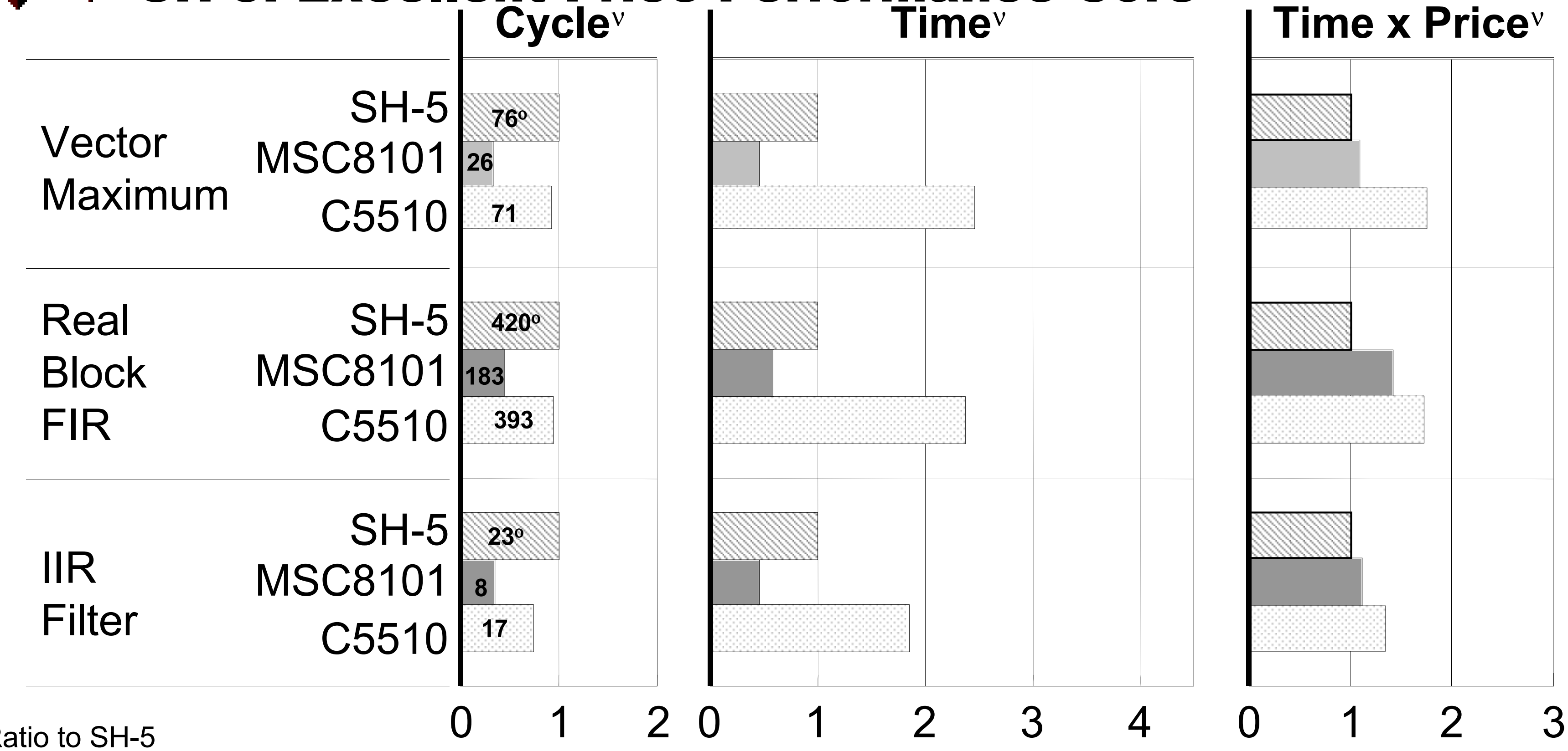


- 4-way SIMD is better than the three issues
- loop unrolling reduces the number of source operand reloads, and enhances performance.
- Rich register states enable extensive unrolling for higher performance.

Result Comparison

SuperH™ RISC engine

- **SH-5: Excellent Price-Performance Core**



^ν Ratio to SH-5

º With caches preloaded. With empty caches 192, 716 and 101 (estimated) cycles for Vec Max, Block FIR and IIR respectively.

MSC8101 300MHz ($96) and TI C5510 160MHz ($29) data from "Buyer's Guide to DSP Processors" 2001 Edition by BDTI
SH-5 results are projected based on execution on ISS (expected to be published by BDTI in the near future).

Hitachi is projecting that the SH-5 operating at 400MHz is priced at $40 in 10,000 units lots at the end of 2002.

Conclusion

- **SH-5:**
  - Good Balance of Performance, Power, and Price
  - Targeting Cost-sensitive Consumer Market
- **SIMD is Better than Multiple Issues**
  - for Multimedia Applications
  - Both Performance and Cost
- **Future Plan: SH-6 and Beyond**
  - Next-generation process: integrate more logic within a reasonable cost
  - SIMD + Multiple Issues will be the "Next Approach"