

Assessing Cortex-R4 and Cortex-A8 Signal and Media Processing Performance

Insight, Analysis, and Advice on Signal Processing Technology



Assessing Cortex-R4 and Cortex-A8 Signal and Media Processing Performance

Jeff Bier
Berkeley Design Technology, Inc.
Berkeley, California USA
+1 (510) 665-1600

info@BDTI.com
<http://www.BDTI.com>

© 2007 Berkeley Design Technology, Inc.



Presentation Goals

By the end of this workshop, you should know:

- In what ways the Cortex-R4 and Cortex-A8 are similar or different
- DSP and media-processing features of the Cortex-R4 and Cortex-A8
- How the Cortex-R4 and Cortex-A8 compare to other ARM cores, other CPUs and DSPs
- How to take advantage of the Cortex-R4 and Cortex-A8's features

© 2007 BDTI

INSIGHT • ANALYSIS • ADVICE
ON SIGNAL PROCESSING TECHNOLOGY

2

© 2007 Berkeley Design Technology, Inc.

Assessing Cortex-R4 and Cortex-A8 Signal and Media Processing Performance

BDTi

Selected ARM Processors at a Glance

| | ARM9E | ARM11 | Cortex-R4 | Cortex-A8 w/NEON* |
|---|--------------------------|--------------------------|--------------------------|--|
| Typical clock rate | 265 MHz | 335 MHz | 300 MHz – 475 MHz | 450 MHz – 1100 MHz |
| Instruction sets | ARMv5E, Thumb | ARMv6, Thumb, Thumb2 | ARMv7, Thumb, Thumb2 | ARMv7, Thumb, Thumb2, NEON |
| Pipeline stages | 5 | 8 | 8 | 13 + 10 (NEON) |
| DSP/media instructions | Minor | Minor | Minor | Extensive (NEON) |
| Multiply-accumulate throughput (fixed-point) | 1 × 32-bit 1 × 16-bit | 1 × 32-bit 2 × 16-bit | 1 × 32-bit 2 × 16-bit | 2 × 32-bit 4 × 16-bit 8 × 8-bit Float: 2 × 32-bit |
| Data bus | 32-bit | 64-bit | 64-bit | 64-/128-bit |
| Branch prediction | No | Yes | Yes | Yes |

© 2007 BDTI
INSIGHT • ANALYSIS • ADVICE
ON SIGNAL PROCESSING TECHNOLOGY
* NEON is optional
3

BDTi

Cortex-R4 versus Cortex-A8

| | Cortex-R4 | Cortex-A8 |
|--------------------------------------|--|--|
| Target application attributes | Deeply embedded systems with RTOS | User applications with full-featured OS |
| Sample applications | Storage, wireless, automotive, networking | High-end cellular, set-top boxes, automotive, printers |
| Special features | Dual-issue for load/store, branch prediction | NEON, limited dual-issue, high clock rate |
| Parallel load/store | Yes | Yes |
| Load latency | 2 cycles | Typically 1 cycle (NEON) |
| Dedicated SIMD unit | No | Optional (NEON) |
| Floating-point unit | Optional (VFP) | Optional (VFP Lite, bundled with NEON unit) |
| Conditional execution | Yes | Yes |

© 2007 BDTI
INSIGHT • ANALYSIS • ADVICE
ON SIGNAL PROCESSING TECHNOLOGY
4

© 2007 Berkeley Design Technology, Inc.

Assessing Cortex-R4 and Cortex-A8 Signal and Media Processing Performance

BDTi

NEON Signal Processing Extensions

- Single-instruction multiple-data (SIMD) instruction set extensions
 - Independent register file
 - Viewed as 32 × 64-bit or 16 × 128-bit
 - Each register treated as packed 8-, 16-, 32-, or 64-bit data
- Data types supported:
 - 8-, 16-, 32-, 64-bit signed or unsigned integer
 - Single-precision (32-bit) floating point
- Supports aligned and unaligned data accesses

- ARM tools support for NEON
 - Vectorizing compiler with support for intrinsics
 - Assembler
 - Cycle-accurate model for SoC Designer environment
- Optimized function libraries for Cortex-A8 with NEON
 - OpenMAX DL API optimized libraries for H.264, AAC decoding
 - Other modules and kernels planned

© 2007 BDTI
INSIGHT • ANALYSIS • ADVICE
ON SIGNAL PROCESSING TECHNOLOGY
5

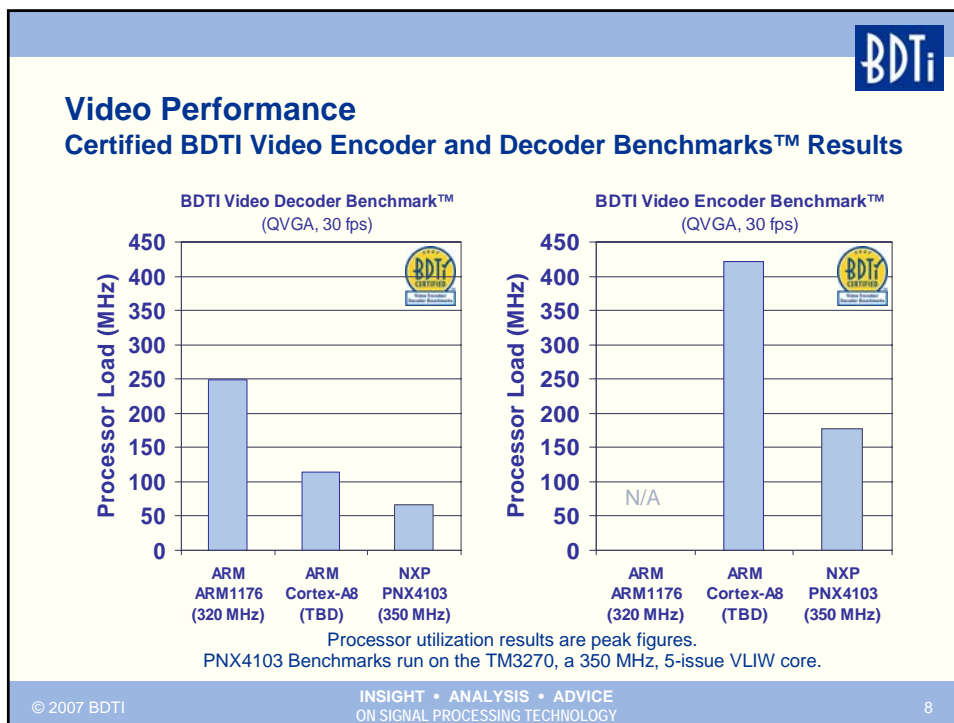
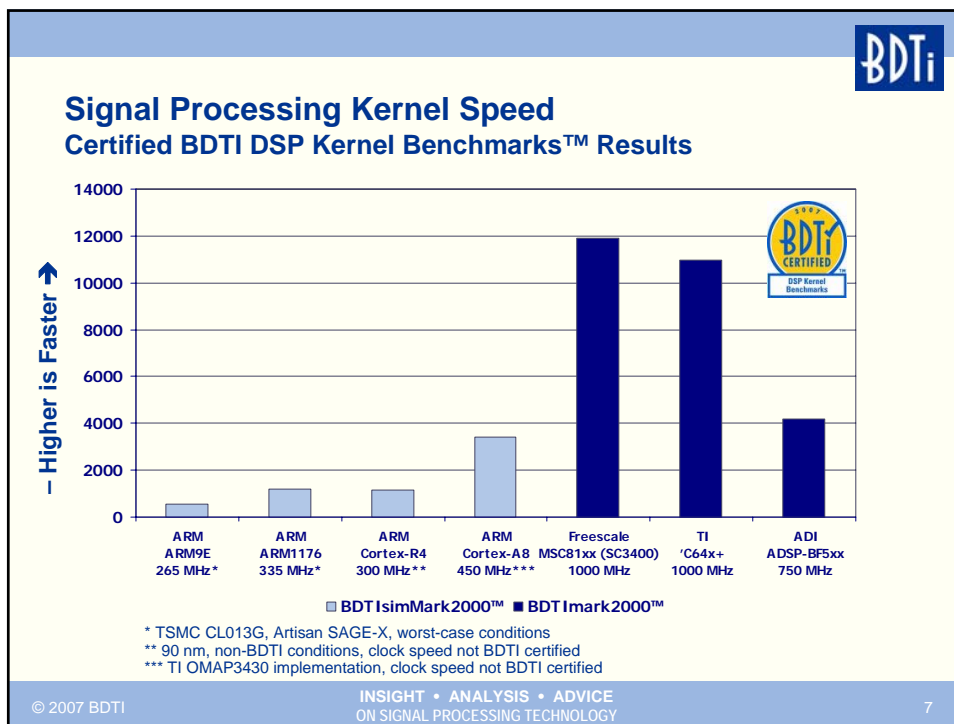
BDTi

Per-Cycle Throughput on Signal Processing Kernels Certified BDTI DSP Kernel Benchmarks™ Results

| Processor | BDTIimark2000™/MHz | BDTImark2000™/MHz |
|-----------------|--------------------|-------------------|
| ARM ARM7 | ~1.0 | - |
| ARM ARM9E | ~2.0 | - |
| ARM ARM1176 | ~3.5 | - |
| ARM Cortex-R4 | ~3.8 | - |
| ARM Cortex-A8 | ~7.5 | - |
| MIPS MIPS24KEc | ~3.0 | - |
| CEVA CEVA-X1620 | ~8.0 | - |
| TI 'C64x+ | - | ~11.0 |
| ADI ADSP-BF5xx | - | ~5.5 |

© 2007 BDTI
INSIGHT • ANALYSIS • ADVICE
ON SIGNAL PROCESSING TECHNOLOGY
6

Assessing Cortex-R4 and Cortex-A8 Signal and Media Processing Performance



Assessing Cortex-R4 and Cortex-A8 Signal and Media Processing Performance



Programming Tips and Tricks for DSP/Multimedia On Cortex-R4 and Cortex-A8

- Know your optimization targets
 - What are the cycle/memory/other targets?
- Take a hierarchical approach
 - Use profiling to identify optimization candidates
 - Begin by making key code sections compiler-friendly
 - Know the compiler
 - Be prepared to write assembly code
- Know the architecture
 - Make effective use of SIMD operations
 - Select algorithms and organize data for SIMD
 - Optimize for dual-issue
 - Consider all available instructions

© 2007 BDTI

INSIGHT • ANALYSIS • ADVICE
ON SIGNAL PROCESSING TECHNOLOGY

9



Programming Tips and Tricks (Continued)

- Make use of software pipelining
 - To mask instruction latencies
 - But know your register limitations
- Memory accesses are costly
 - Keep data in registers and re-use it
 - Fully exploit memory bandwidth
 - Organize data flow to minimize cache misses
- Use off-the-shelf components where appropriate

© 2007 BDTI

INSIGHT • ANALYSIS • ADVICE
ON SIGNAL PROCESSING TECHNOLOGY

10

© 2007 Berkeley Design Technology, Inc.

Assessing Cortex-R4 and Cortex-A8 Signal and Media Processing Performance



Example: FIR Filter Kernel

C implementation of FIR kernel

$$y[n] = \sum_{k=0}^{T-1} x[n-k]h[k]$$

```

#define N 40
#define T 16

for (n=T-1; n<N+T; n++) {
    for (k=0,SUM=0; k<T; k++) {
        SUM += x[n-k] * h[k];
    }
    y[n] = SUM;
}
    
```



Analysis: Compiled FIR Filter for Cortex-R4

```

|L1.24|
MOV    r0,#0
MOV    r2,r0

|L1.32|
SUB    r6,r1,r0
ADD    r7,r4,r0,LSL #1
ADD    r6,r3,r6,LSL #1
ADD    r0,r0,#1
LDRH   r7,[r7,#0]
Dual issue |
CMP    r0,#0x10
LDRH   r6,[r6,#0]
SMLABB r2,r6,r7,r2
Dual issue |
BLT    |L1.32|
ADD    r0,r5,r1,LSL #1
ADD    r1,r1,#1
CMP    r1,#0x38
STRH   r2,[r0,#0]
BLT    |L1.24|
    
```

Annotations:

- 2 instructions per load (points to ADD r7, r4, r0, LSL #1 and ADD r6, r3, r6, LSL #1)
- 2 stall cycles (points to LDRH r6, [r6, #0])
- 2 branches, 2 instructions per branch (points to BLT |L1.32| and BLT |L1.24|)

Assessing Cortex-R4 and Cortex-A8 Signal and Media Processing Performance

BDTi

Inner Loop Cycle Count for Cortex-R4

| | | | | |
|---|---|---|---|---|
| <pre> L1.24 MOV r0,#0 MOV r2,r0 L1.32 SUB r6,r1,r0 ADD r7,r4,r0,LSL #1 ADD r6,r3,r6,LSL #1 ADD r0,r0,#1 LDRH r7,[r7,#0] CMP r0,#0x10 LDRH r6,[r6,#0] SMLABB r2,r6,r7,r2 BLT L1.32 ADD r0,r5,r1,LSL #1 ADD r1,r1,#1 CMP r1,#0x38 STRH r2,[r0,#0] BLT L1.24 </pre> | } | = | { | <pre> #define N 40 #define T 16 for (n=T-1; n<N+T; n++) { for (k=0,SUM=0; k<T; k++) { SUM += x[n-k] * h[k]; } y[n] = SUM; } </pre> |
|---|---|---|---|---|

9 cycles per tap = 0.11 taps per cycle*

*Inner loop only

© 2007 BDTI
INSIGHT • ANALYSIS • ADVICE
ON SIGNAL PROCESSING TECHNOLOGY
13

BDTi

Give the Compiler a Hand

| | | | | |
|---|---|---|---|--|
| <h3>Human-friendly</h3> <pre> #define N 40 #define T 16 for (n=T-1; n<N+T; n++) { for (k=0,SUM=0; k<T; k++) { SUM += x[n-k] * h[k]; } y[n] = SUM; } </pre> | } | = | { | <h3>Compiler-friendly</h3> <pre> #define N 40 #define T 16 for (n=N; n; n--) { for (k=T,SUM=0; k; k--) { SUM += x[n-k] * h[k]; } y[n] = SUM; } </pre> |
|---|---|---|---|--|

Count downwards in "for" loops

© 2007 BDTI
INSIGHT • ANALYSIS • ADVICE
ON SIGNAL PROCESSING TECHNOLOGY
14

Assessing Cortex-R4 and Cortex-A8 Signal and Media Processing Performance



Give the Compiler a Hand

Human-friendly

```
#define N 40
#define T 16

for (n=T-1; n<N+T; n++) {
    for (k=0,SUM=0; k<T; k++) {
        SUM += x[n-k] * h[k];
    }
    y[n] = SUM;
}
```

Compiler-friendly

```
#define N 40
#define T 16
xp = x+15;
for (n=N; n; n--) {
    short *xt = xp++;
    short *ht = h;
    for (k=T,SUM=0; k; k--) {
        SUM += *xt-- * *ht++;
    }
    *y++ = SUM;
}
```

Make pointer increment explicit



Analysis: Compiled FIR Filter for Cortex-R4

```
|L1.24|
LDR    r2, |L1.92|
MOV    r1, r4
MOV    r0, #0x10
ADD    r4, r4, #2
MOV    r3, #0
```

```
|L1.44|
SUBS   r0, r0, #1
LDRH  r7, [r1], #-2
LDRH  r12, [r2], #2
Dual issue |SMLABB r3, r7, r12, r3
          |BNE |L1.44|
SUBS   r5, r5, #1
STRH  r3, [r6], #2
BNE   |L1.24|
```

1 instruction per load

2 stall cycles

2 branches
1 instruction per branch

6 cycles per tap = 0.17 taps per cycle

Assessing Cortex-R4 and Cortex-A8 Signal and Media Processing Performance

BDTi

Adding SIMD: The Simple Approach

| | | | | | |
|------------|-------|--------------|---|---------------|-----|
| ... | | | | | |
| L1.44 | LDRD | r6,[r1],#-8 | | | |
| | | | ← | 1 stall cycle | |
| | LDRD | r10,[r2],#8 | | | |
| | | | ← | 1 stall cycle | |
| | SUBS | r0,r0,#4 | | | |
| | SMLAD | r3,r6,r10,r3 | | | |
| | | | ← | 1 stall cycle | |
| Dual issue | SMLAD | r3,r7,r11,r3 | | | |
| | BNE | L1.44 | | | ... |

4 taps in 8 cycles = 0.5 taps per cycle

© 2007 BDTI
INSIGHT • ANALYSIS • ADVICE
ON SIGNAL PROCESSING TECHNOLOGY
17

BDTi

Add Software Pipelining

| | | | | | |
|------------|--------|--------------|--|--|--|
| | | | | | |
| | LDRD | r4,[r1],#-8 | | | |
| | LDRD | r8,[r2],#8 | | | |
| | ... | | | | |
| LOOP | LDRD | r6,[r1],#-8 | | | |
| | SMLAD | r3,r4,r8,r3 | | | |
| | LDRD | r10,[r2],#8 | | | |
| | SMLAD | r3,r5,r9,r3 | | | |
| | SUBS | r0,r0,#8 | | | |
| | LDRGTD | r4,[r1],#-8 | | | |
| | SMLAD | r3,r6,r10,r3 | | | |
| | LDRGTD | r8,[r2],#8 | | | |
| Dual issue | SMLAD | r3,r7,r11,r3 | | | |
| | BNE | LOOP | | | |
| | ... | | | | |

8 taps in 13 cycles = 0.62 taps per cycle

© 2007 BDTI
INSIGHT • ANALYSIS • ADVICE
ON SIGNAL PROCESSING TECHNOLOGY
18

© 2007 Berkeley Design Technology, Inc.

Assessing Cortex-R4 and Cortex-A8 Signal and Media Processing Performance

BDTi

Fully Optimized FIR Inner Loop for Cortex-R4

```

loop  ldrd  rx, [rx, #x]           smlad rx, rx, rx, rx           smlad rx, rx, rx, rx
      smuad rx, rx, rx           smlad rx, rx, rx, rx           smlad rx, rx, rx, rx
      smuad rx, rx, rx           smlad rx, rx, rx, rx           ldr  rx, [rx, #x]
      smlad rx, rx, rx, rx       ldrd  rx, [rx], #x            smlad rx, rx, rx, rx
      smlad rx, rx, rx, rx       ldrd  rx, [rx, #x]            smlad rx, rx, rx, rx
      ldrd  rx, [rx, #x]          smlad rx, rx, rx, rx           smlad rx, rx, rx, rx
      smuad rx, rx, rx           smlad rx, rx, rx, rx           smlad rx, rx, rx, rx
      smuad rx, rx, rx           smlad rx, rx, rx, rx           smlad rx, rx, rx, rx
      smlad rx, rx, rx, rx       smlad rx, rx, rx, rx           smlad rx, rx, rx, rx
      smlad rx, rx, rx, rx       ldrd  rx, [rx, #x]            subs  rx, rx, #x
      ldrd  rx, [rx], #x          smlad rx, rx, rx, rx           ldrd  rx, [rx, #x]
      ldrd  rx, [rx, #x]          smlad rx, rx, rx, rx           ldmia rx!, {rx-rx}
      smlad rx, rx, rx, rx       smlad rx, rx, rx, rx           mov  rx, rx, ASR #x
      smlad rx, rx, rx, rx       smlad rx, rx, rx, rx           mov  rx, rx, ASR #x
      smlad rx, rx, rx, rx       ldrd  rx, [rx], #-x            pkhtb rx, rx, rx, ASR #x
      smlad rx, rx, rx, rx       ldrd  rx, [rx, #x]            pkhtb rx, rx, rx, ASR #x
      ldrd  rx, [rx, #x]          smlad rx, rx, rx, rx           strd rx, [rx, #x]
      smlad rx, rx, rx, rx       smlad rx, rx, rx, rx           bgt  loop
  
```

64 taps in ~65 cycles = ~0.99 taps/cycle

© 2007 BDTI
INSIGHT • ANALYSIS • ADVICE
ON SIGNAL PROCESSING TECHNOLOGY
19

BDTi

Cortex-A8 – NEON SIMD Example Instruction

VMUL.I32.S16 Q8, D2, D4

Performs four 16 x 16 → 32-bit multiplies

The diagram shows two 64-bit registers, D2 and D4, each divided into four 16-bit segments. Red arrows indicate that the first 16 bits of D2 are multiplied by the first 16 bits of D4, the second 16 bits of D2 by the second 16 bits of D4, and so on. The results of these four multiplications are then packed into a single 128-bit Q8 register, with each 32-bit result occupying a 32-bit segment.

© 2007 BDTI
INSIGHT • ANALYSIS • ADVICE
ON SIGNAL PROCESSING TECHNOLOGY
20

Assessing Cortex-R4 and Cortex-A8 Signal and Media Processing Performance

BDTi

Cortex-A8 Assembly Coded FIR Filter Fragment

```

VLD1.16@128   {D6,D7}, [r2]!
VEXT.8        D1, D11, D0, #2
Dual issue   VMLA.I32.S16   Q4, D1, D7
              VLD1.16@128   {D4,D5}, [r2]
              VLD1.16@128   {D2,D3}, [r3], r1
              VEXT.8        D2, D2, D3, #2
Dual issue   VMUL.I32.S16   Q8, D2, D4
              VEXT.8        D3, D3, D10, #2
Dual issue   VMLA.I32.S16   Q8, D3, D5
              VEXT.8        D0, D10, D11, #2
              VMLA.I32.S16   Q8, D0, D6
Dual issue   VADD.I32       D8, D8, D9
              VST1.16@128   {D0,D1,D2,D3},[r3],r1
              VSUM.I32       D8, D8, D8
Dual issue   VSHR.S32       D0, D8, #18
              VST1.32       {D16, D17}, [r3]
    
```

© 2007 BDTI
INSIGHT • ANALYSIS • ADVICE
ON SIGNAL PROCESSING TECHNOLOGY
21

BDTi

Cortex-A8 Stalls

```

VLD1.16@128   {D6,D7}, [r2]!
VEXT.8        D1, D11, D0, #2
Dual issue   VMLA.I32.S16   Q4, D1, D7
              VLD1.16@128   {D4,D5}, [r2]
              VLD1.16@128   {D2,D3}, [r3], r1
              VEXT.8        D2, D2, D3, #2
Dual issue   VMUL.I32.S16   Q8, D2, D4
              VEXT.8        D3, D3, D10, #2
Dual issue   VMLA.I32.S16   Q8, D3, D5
              VEXT.8        D0, D10, D11, #2
              VMLA.I32.S16   Q8, D0, D6
Dual issue   VADD.I32       D8, D8, D9
              VST1.16@128   {D0,D1,D2,D3},[r3],r1
              VSUM.I32       D8, D8, D8
Dual issue   VSHR.S32       D0, D8, #18
              VST1.32       {D16, D17}, [r3]
    
```

MAC forwarding –
No stall here

2 stalls (on D8)

2 stalls (on D8)

© 2007 BDTI
INSIGHT • ANALYSIS • ADVICE
ON SIGNAL PROCESSING TECHNOLOGY
22

Assessing Cortex-R4 and Cortex-A8 Signal and Media Processing Performance



Conclusions – Cortex-R4 and Cortex-A8

- Cortex-R4:
 - Comparable DSP throughput to ARM11
 - Roughly 2X DSP throughput vs. ARM9E
 - Energy and cost/performance results TBD
 - Moderate instruction set and microarchitecture complexity
- Cortex-A8
 - Roughly 3-8X DSP throughput vs. ARM11, Cortex-R4
 - Comparable to mid-range DSPs
 - Energy and cost/performance results TBD
 - High instruction set and microarchitecture complexity



Conclusions – The Big Picture

- General-purpose CPUs' DSP capabilities have increased rapidly
 - Instruction-set enhancements
 - Microarchitecture enhancements
 - Implementation, fabrication enhancements
- There's no free lunch: trade-offs include increased complexity, area, and power
- DSP-specific application development support is being built out for DSP-enabled CPUs
 - But much remains to be done
- Approaching full performance potential requires significant software optimization effort
 - And knowledge of algorithms, architecture, and tools

Assessing Cortex-R4 and Cortex-A8 Signal and Media Processing Performance



For More Information...

Subscribe to BDTI's **Inside DSP** newsletter and website: www.INSIDEdsp.com

Visit www.BDTI.com:

- BDTI Certified™ benchmark scores for dozens of processors
- *Pocket Guide to Processors for DSP*
 - Basic stats on over 40 chips and cores
- Articles, white papers, and presentation slides
 - Processor architectures and performance
 - Video, DSP applications
 - Video, DSP software development
- *comp.dsp* FAQ

