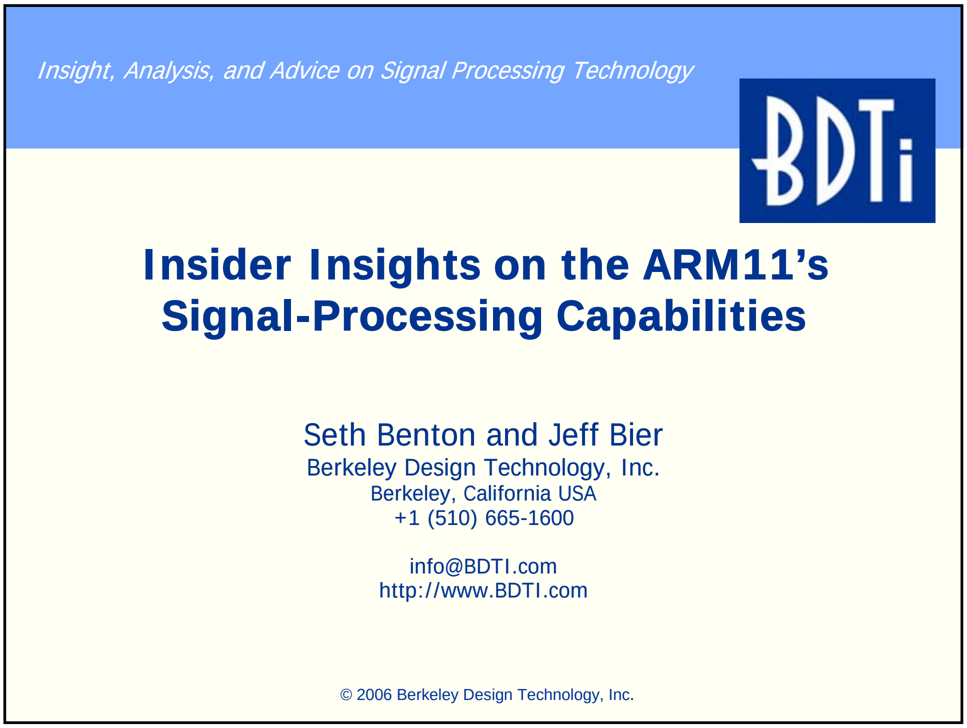
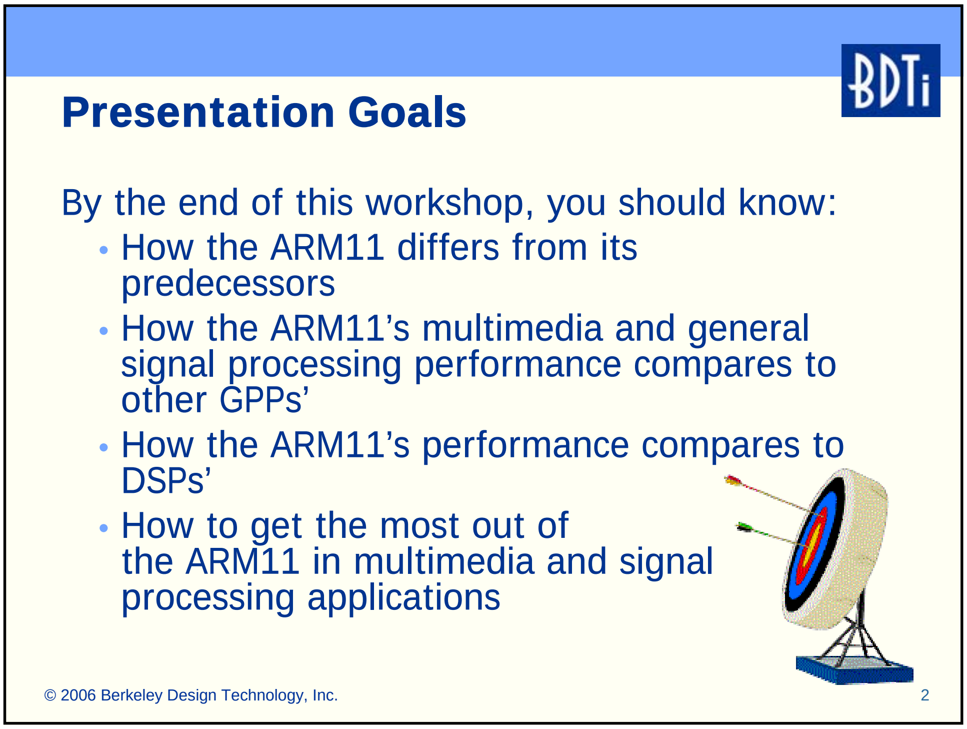


Insight, Analysis, and Advice on Signal Processing Technology

Insider Insights on the ARM11's Signal-Processing Capabilities

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

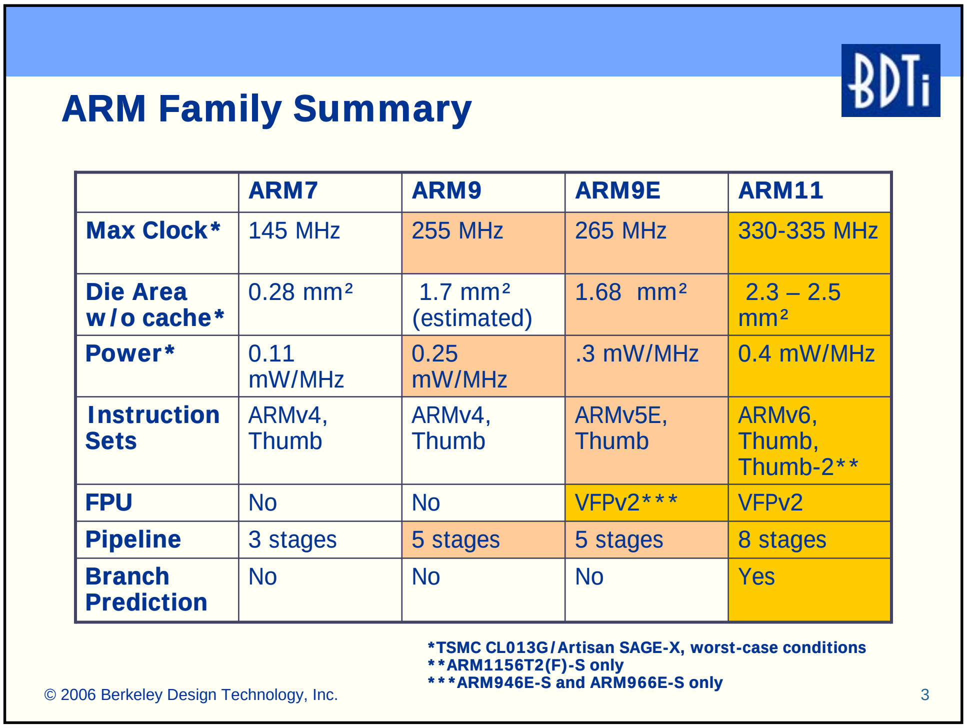
info@BDTI.com
http://www.BDTI.com



Presentation Goals

By the end of this workshop, you should know:

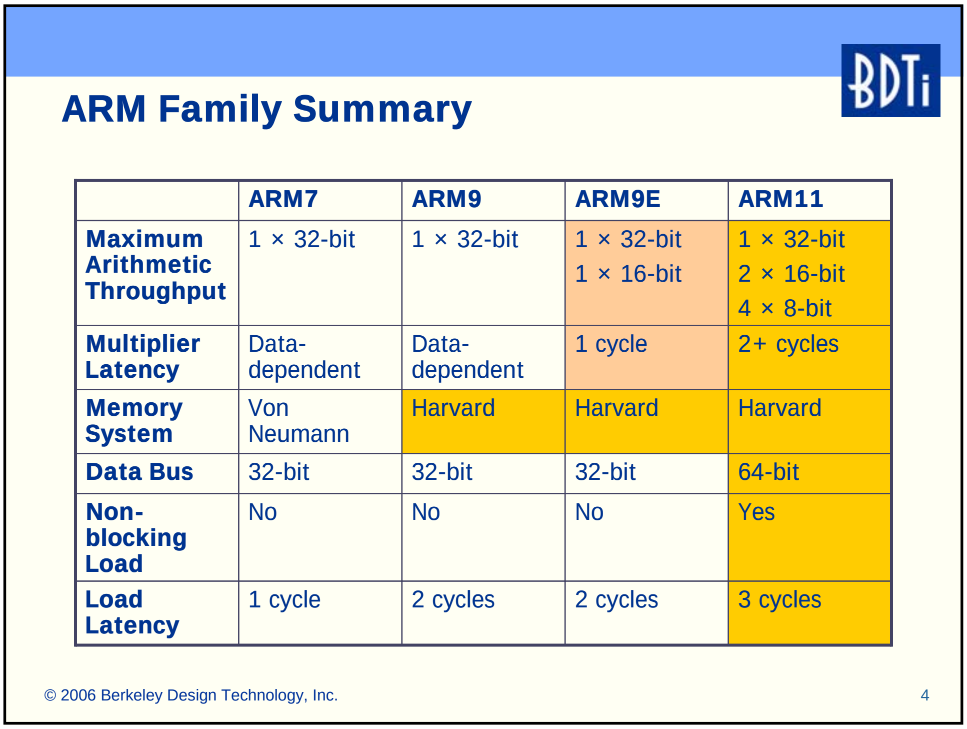
- How the ARM11 differs from its predecessors
- How the ARM11's multimedia and general signal processing performance compares to other GPPs'
- How the ARM11's performance compares to DSPs'
- How to get the most out of the ARM11 in multimedia and signal processing applications



ARM Family Summary

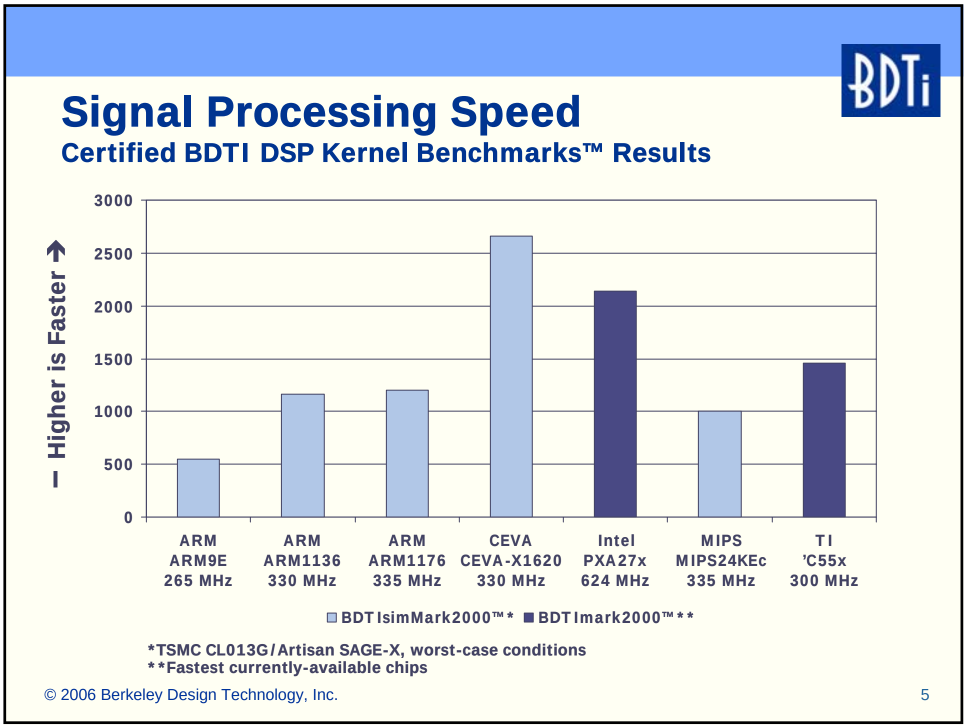
	ARM7	ARM9	ARM9E	ARM11
Max Clock*	145 MHz	255 MHz	265 MHz	330-335 MHz
Die Area w/o cache*	0.28 mm²	1.7 mm² (estimated)	1.68 mm²	2.3 – 2.5 mm²
Power*	0.11 mW/MHz	0.25 mW/MHz	.3 mW/MHz	0.4 mW/MHz
Instruction Sets	ARMv4, Thumb	ARMv4, Thumb	ARMv5E, Thumb	ARMv6, Thumb, Thumb-2**
FPU	No	No	VFPv2***	VFPv2
Pipeline	3 stages	5 stages	5 stages	8 stages
Branch Prediction	No	No	No	Yes

***TSMC CL013G/Artisan SAGE-X, worst-case conditions**
****ARM1156T2(F)-S only**
*****ARM946E-S and ARM966E-S only**



ARM Family Summary

	ARM7	ARM9	ARM9E	ARM11
Maximum Arithmetic Throughput	1 × 32-bit	1 × 32-bit	1 × 32-bit 1 × 16-bit	1 × 32-bit 2 × 16-bit 4 × 8-bit
Multiplier Latency	Data-dependent	Data-dependent	1 cycle	2+ cycles
Memory System	Von Neumann	Harvard	Harvard	Harvard
Data Bus	32-bit	32-bit	32-bit	64-bit
Non-blocking Load	No	No	No	Yes
Load Latency	1 cycle	2 cycles	2 cycles	3 cycles

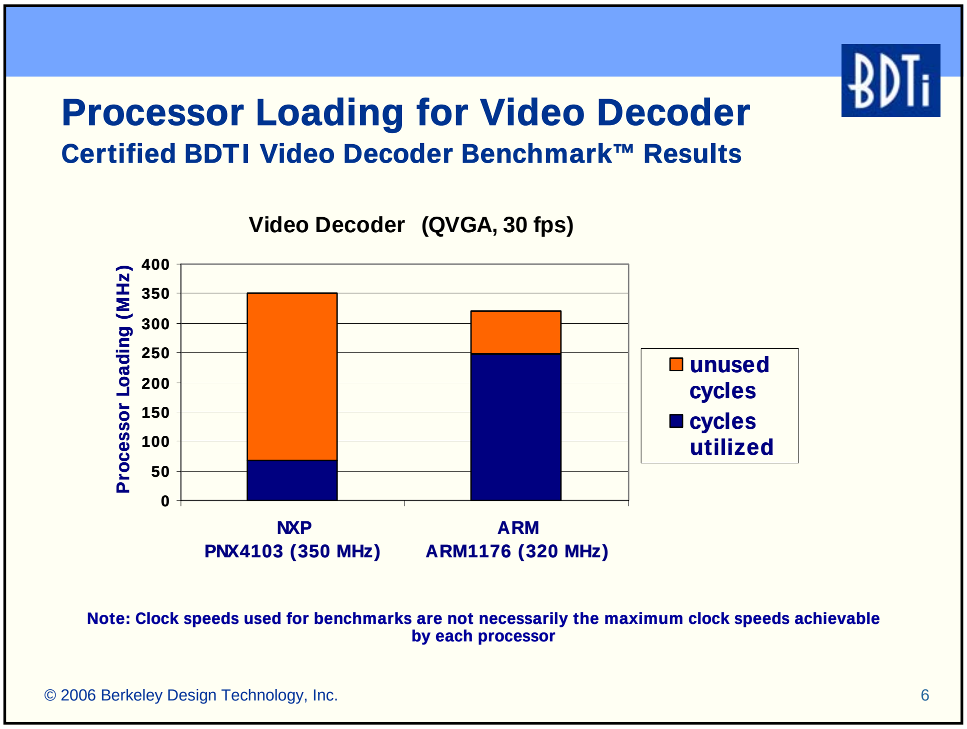


Signal Processing Speed

Certified BDTI DSP Kernel Benchmarks™ Results

*TSMC CL013G/Artisan SAGE-X, worst-case conditions
**Fastest currently-available chips

5

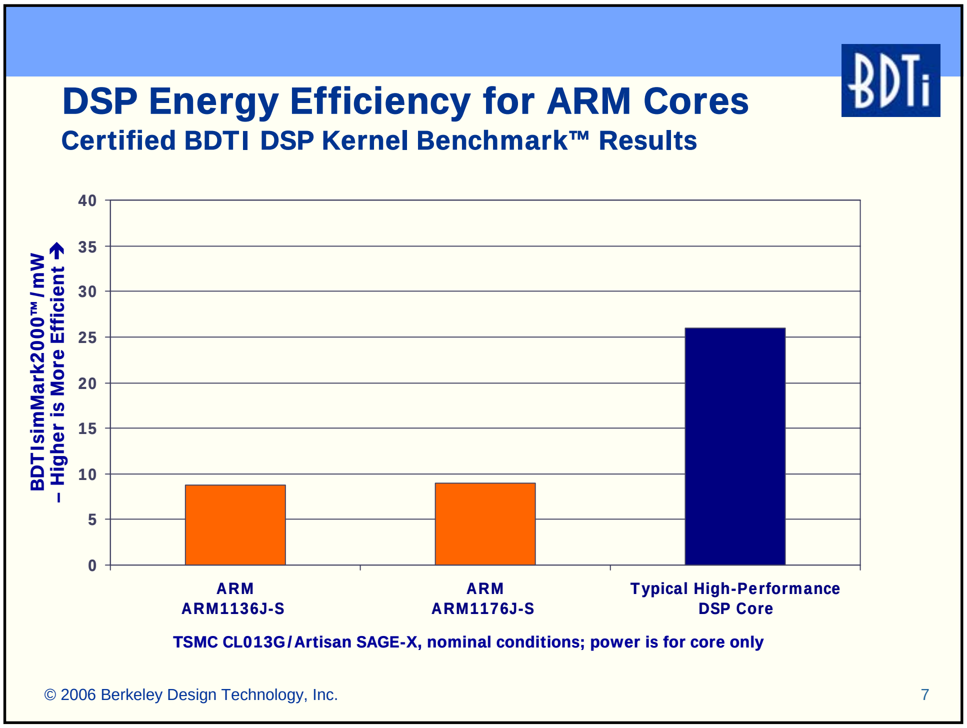


Processor Loading for Video Decoder

Certified BDTI Video Decoder Benchmark™ Results

Note: Clock speeds used for benchmarks are not necessarily the maximum clock speeds achievable by each processor

6

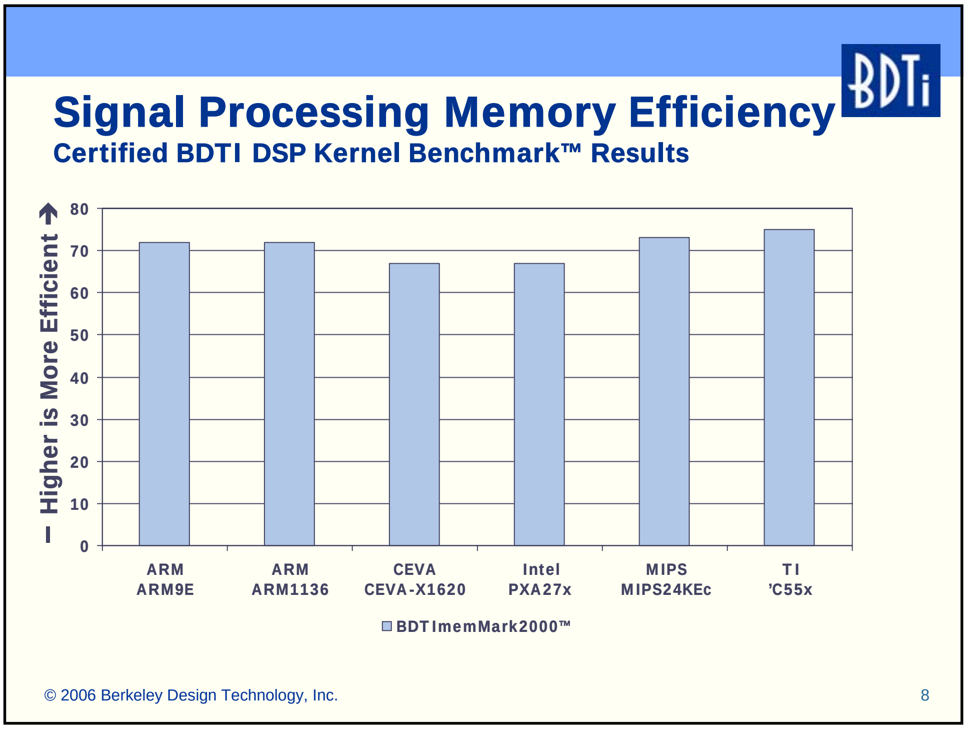


DSP Energy Efficiency for ARM Cores

Certified BDTI DSP Kernel Benchmark™ Results

TSMC CL013G/Artisan SAGE-X, nominal conditions; power is for core only

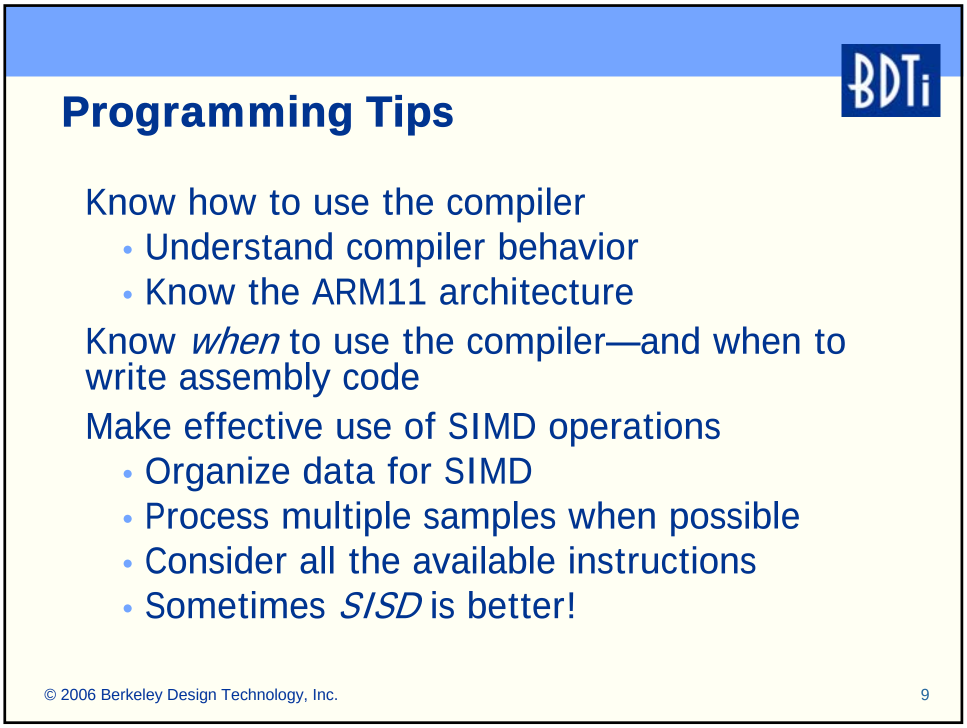
© 2006 Berkeley Design Technology, Inc. 7



Signal Processing Memory Efficiency

Certified BDTI DSP Kernel Benchmark™ Results

© 2006 Berkeley Design Technology, Inc. 8



Programming Tips

Know how to use the compiler

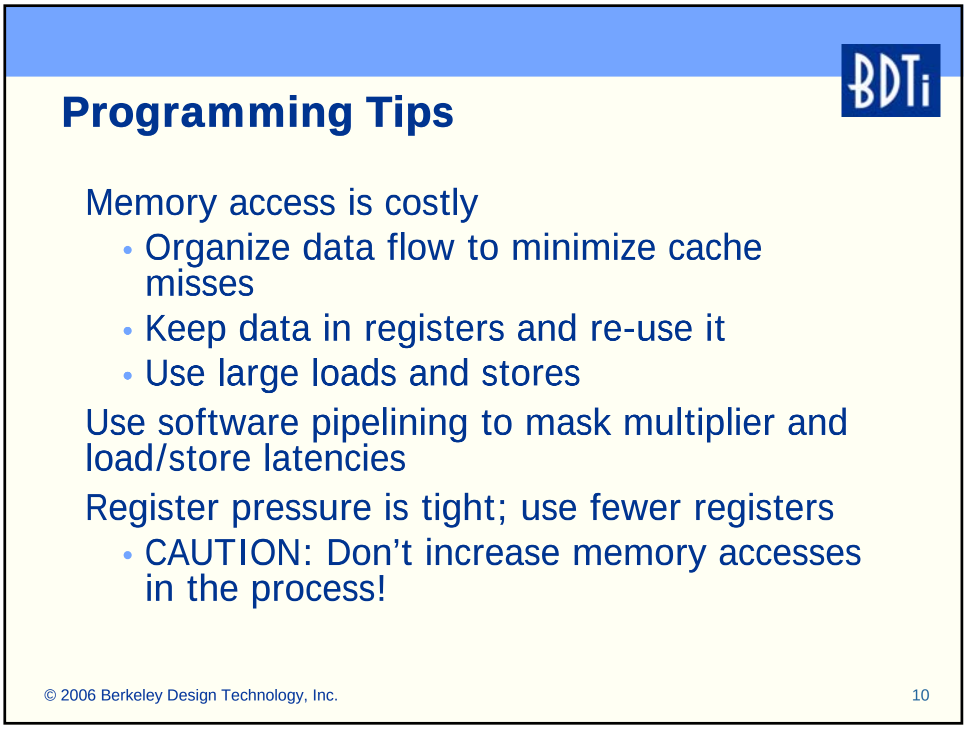
- Understand compiler behavior
- Know the ARM11 architecture

Know *when* to use the compiler—and when to write assembly code

Make effective use of SIMD operations

- Organize data for SIMD
- Process multiple samples when possible
- Consider all the available instructions
- Sometimes *SISD* is better!

9



Programming Tips

Memory access is costly

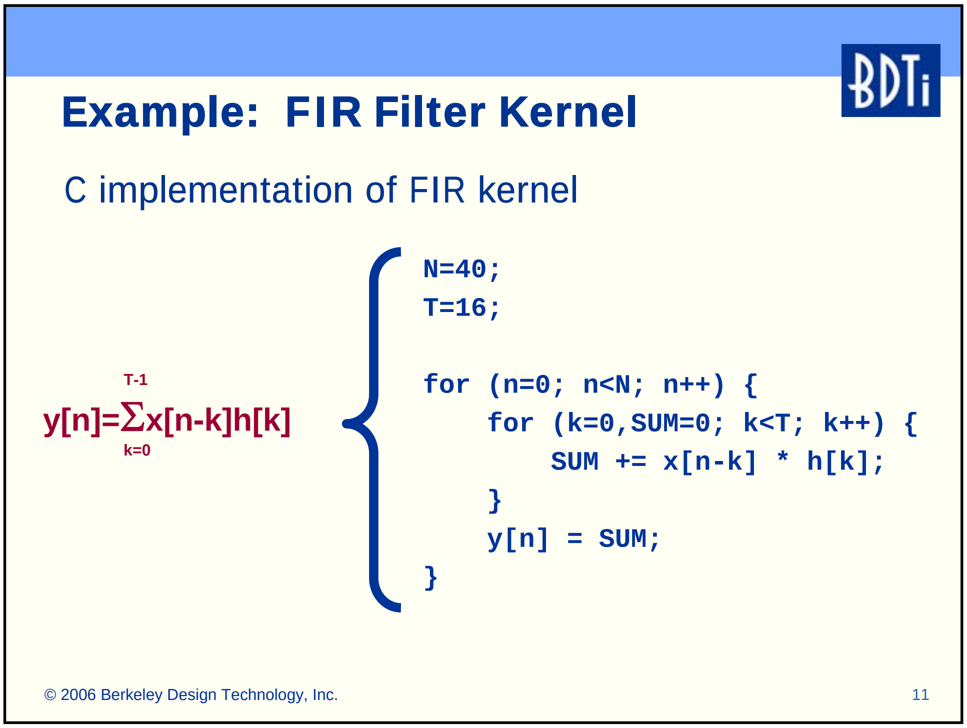
- Organize data flow to minimize cache misses
- Keep data in registers and re-use it
- Use large loads and stores

Use software pipelining to mask multiplier and load/store latencies

Register pressure is tight; use fewer registers

- CAUTION: Don't increase memory accesses in the process!

10



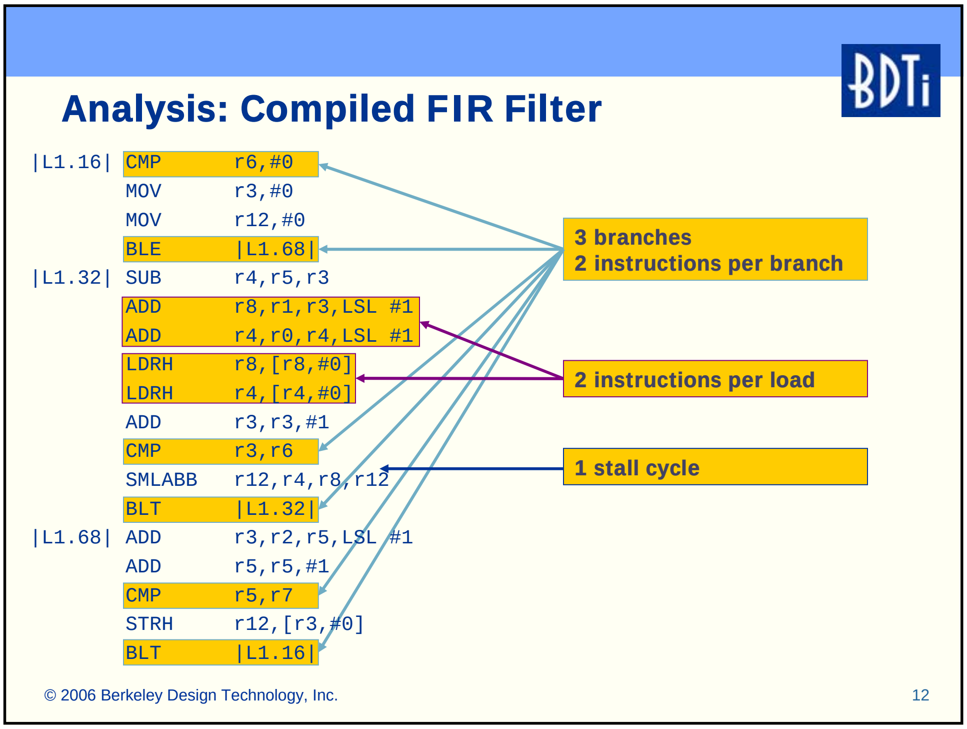
Example: FIR Filter Kernel

C implementation of FIR kernel

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

```
N=40;
T=16;

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

© 2006 Berkeley Design Technology, Inc. 11



Analysis: Compiled FIR Filter

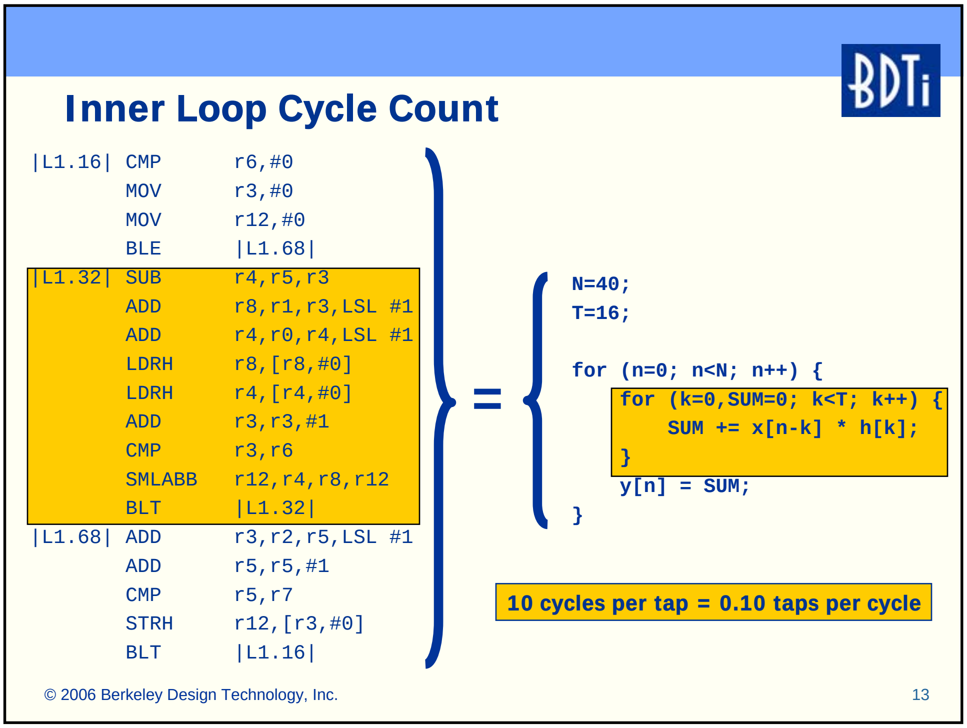
```
|L1.16| CMP     r6,#0
        MOV     r3,#0
        MOV     r12,#0
        BLE     |L1.68|
|L1.32| SUB     r4,r5,r3
        ADD     r8,r1,r3,LSL #1
        ADD     r4,r0,r4,LSL #1
        LDRH    r8,[r8,#0]
        LDRH    r4,[r4,#0]
        ADD     r3,r3,#1
        CMP     r3,r6
        SMLABB  r12,r4,r8,r12
        BLT     |L1.32|
|L1.68| ADD     r3,r2,r5,LSL #1
        ADD     r5,r5,#1
        CMP     r5,r7
        STRH    r12,[r3,#0]
        BLT     |L1.16|
```

3 branches
2 instructions per branch

2 instructions per load

1 stall cycle

© 2006 Berkeley Design Technology, Inc. 12



Inner Loop Cycle Count

```
|L1.16| CMP      r6,#0
        MOV      r3,#0
        MOV      r12,#0
        BLE      |L1.68|
|L1.32| SUB      r4,r5,r3
        ADD      r8,r1,r3,LSL #1
        ADD      r4,r0,r4,LSL #1
        LDRH     r8,[r8,#0]
        LDRH     r4,[r4,#0]
        ADD      r3,r3,#1
        CMP      r3,r6
        SMLABB   r12,r4,r8,r12
        BLT      |L1.32|
|L1.68| ADD      r3,r2,r5,LSL #1
        ADD      r5,r5,#1
        CMP      r5,r7
        STRH     r12,[r3,#0]
        BLT      |L1.16|
```

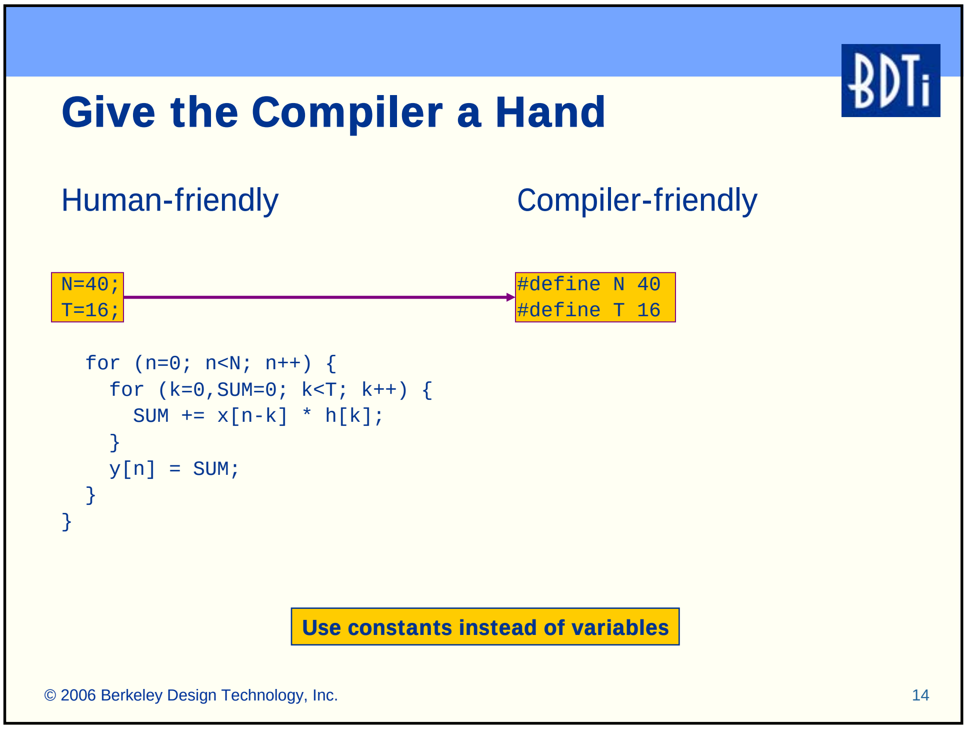
=

```
N=40;
T=16;

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

10 cycles per tap = 0.10 taps per cycle

© 2006 Berkeley Design Technology, Inc.

13



Give the Compiler a Hand

Human-friendly

Compiler-friendly

```
N=40;
T=16;
```

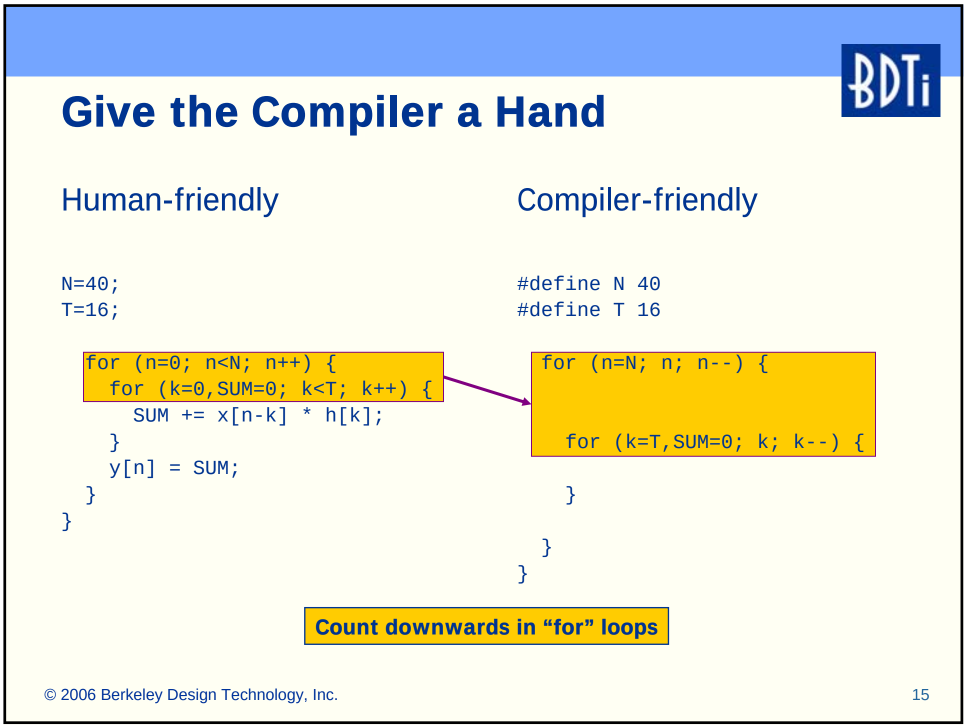
```
#define N 40
#define T 16
```

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

Use constants instead of variables

© 2006 Berkeley Design Technology, Inc.

14



Give the Compiler a Hand

Human-friendly

```
N=40;
T=16;

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

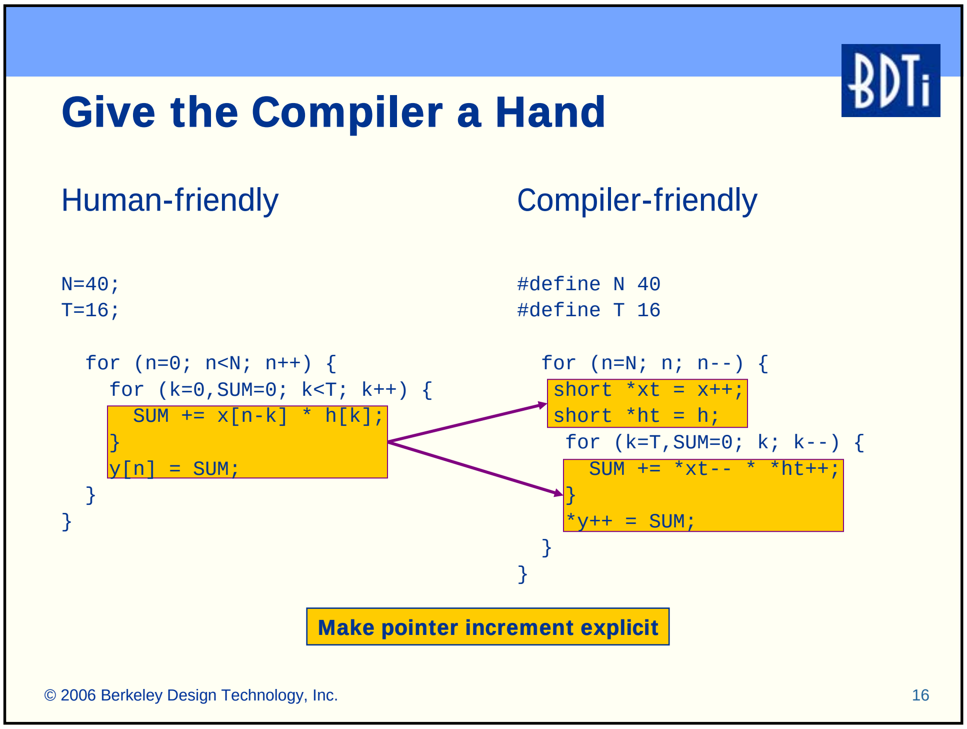
Compiler-friendly

```
#define N 40
#define T 16

  for (n=N; n; n--) {


    for (k=T,SUM=0; k; k--) {

    }

  }
}
```

Count downwards in "for" loops

© 2006 Berkeley Design Technology, Inc.

15



Give the Compiler a Hand

Human-friendly

```
N=40;
T=16;

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

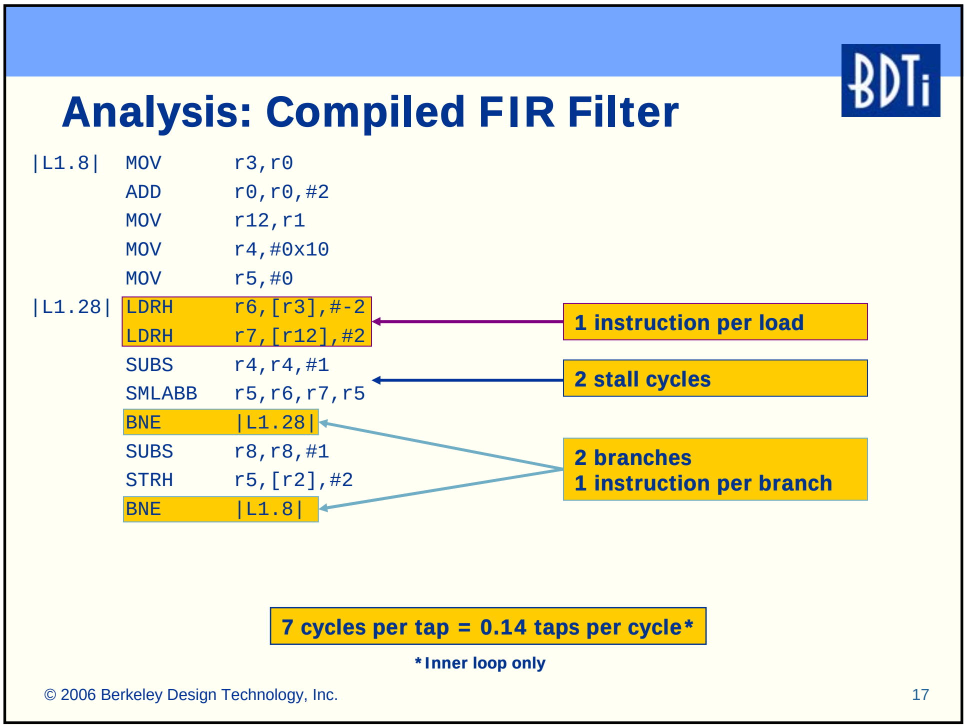
Compiler-friendly

```
#define N 40
#define T 16

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

Make pointer increment explicit

© 2006 Berkeley Design Technology, Inc.

16



Analysis: Compiled FIR Filter

```
|L1.8|  MOV      r3,r0
        ADD      r0,r0,#2
        MOV      r12,r1
        MOV      r4,#0x10
        MOV      r5,#0
|L1.28| LDRH     r6,[r3],#-2
        LDRH     r7,[r12],#2
        SUBS     r4,r4,#1
        SMLABB   r5,r6,r7,r5
        BNE      |L1.28|
        SUBS     r8,r8,#1
        STRH     r5,[r2],#2
        BNE      |L1.8|
```

1 instruction per load

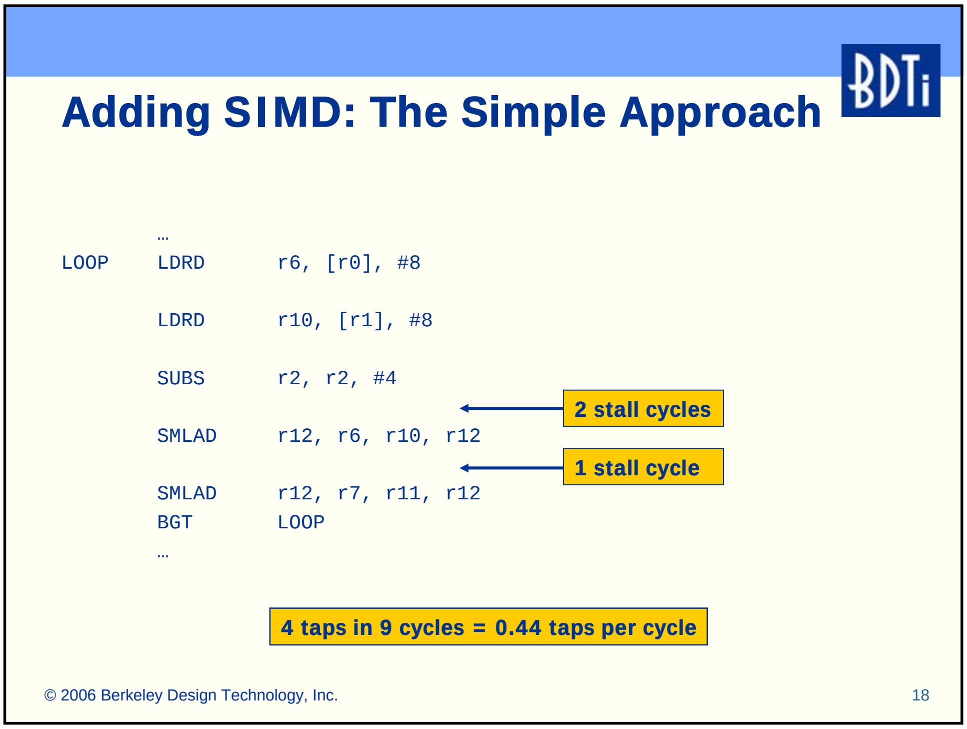
2 stall cycles

2 branches
1 instruction per branch

7 cycles per tap = 0.14 taps per cycle*

***Inner loop only**

17



Adding SIMD: The Simple Approach

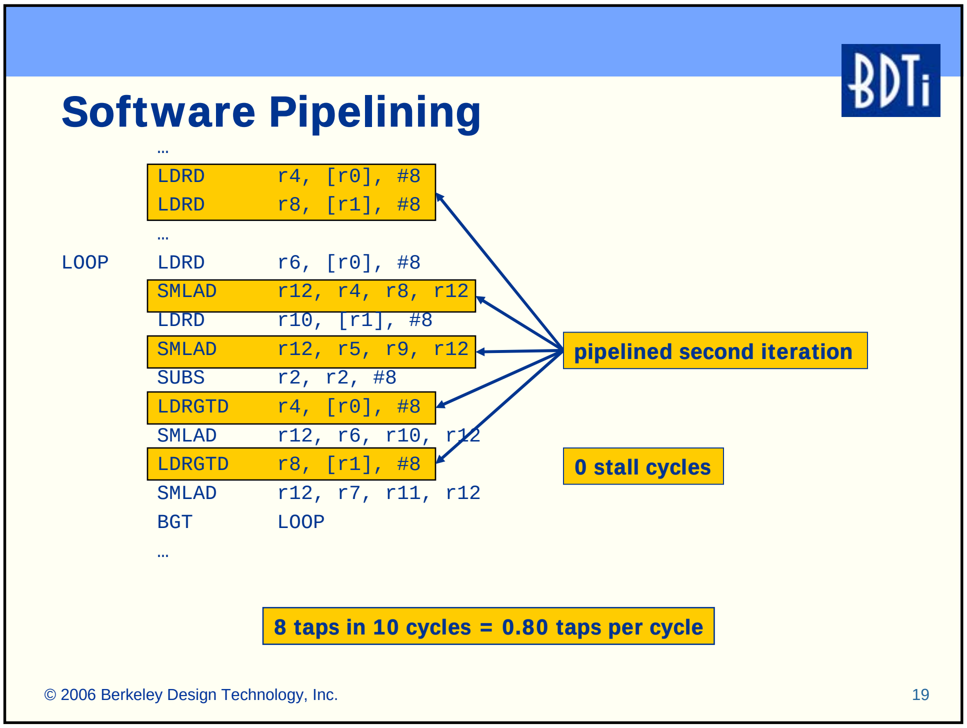
```
        …
LOOP    LDRD      r6, [r0], #8

        LDRD      r10, [r1], #8

        SUBS      r2, r2, #4

        SMLAD     r12, r6, r10, r12

        SMLAD     r12, r7, r11, r12
        BGT       LOOP
        …
```

2 stall cycles

1 stall cycle

4 taps in 9 cycles = 0.44 taps per cycle

18



Software Pipelining

```
        …
        LDRD      r4, [r0], #8
        LDRD      r8, [r1], #8
        …
LOOP    LDRD      r6, [r0], #8
        SMLAD     r12, r4, r8, r12
        LDRD      r10, [r1], #8
        SMLAD     r12, r5, r9, r12
        SUBS      r2, r2, #8
        LDRGTD    r4, [r0], #8
        SMLAD     r12, r6, r10, r12
        LDRGTD    r8, [r1], #8
        SMLAD     r12, r7, r11, r12
        BGT       LOOP
        …
```

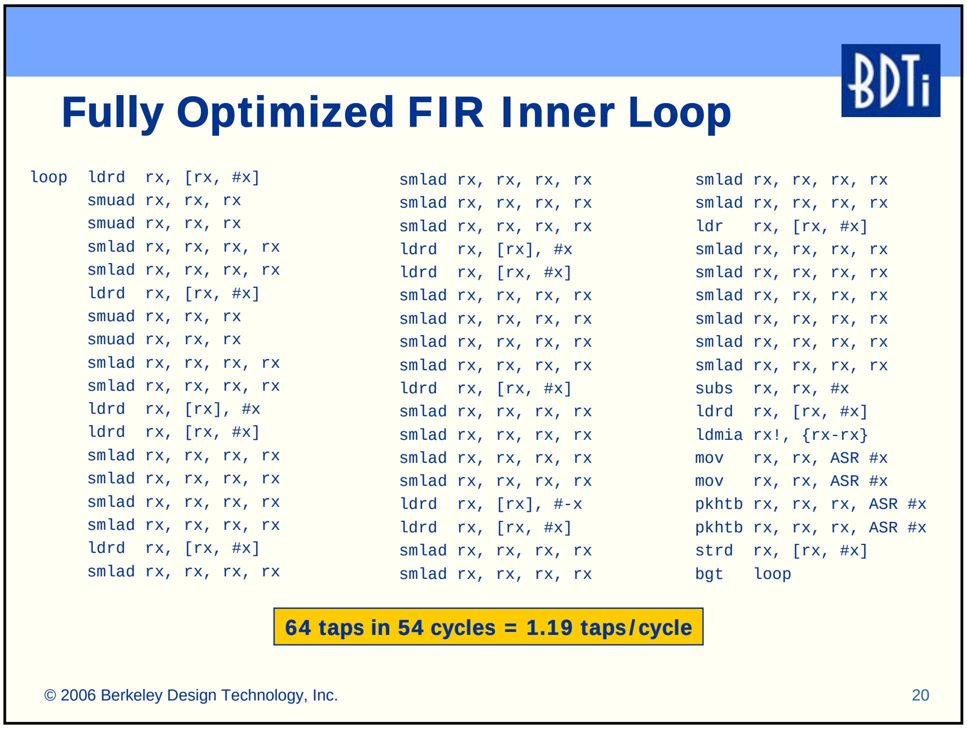
pipelined second iteration

0 stall cycles

8 taps in 10 cycles = 0.80 taps per cycle

© 2006 Berkeley Design Technology, Inc.

19



Fully Optimized FIR Inner Loop

```
loop  ldrd  rx, [rx, #x]
      smuad rx, rx, rx
      smuad rx, rx, rx
      smlad rx, rx, rx, rx
      smlad rx, rx, rx, rx
      ldrd  rx, [rx, #x]
      smuad rx, rx, rx
      smuad rx, rx, rx
      smlad rx, rx, rx, rx
      smlad rx, rx, rx, rx
      ldrd  rx, [rx], #x
      ldrd  rx, [rx, #x]
      smlad rx, rx, rx, rx
      smlad rx, rx, rx, rx
      smlad rx, rx, rx, rx
      smlad rx, rx, rx, rx
      ldrd  rx, [rx, #x]
      smlad rx, rx, rx, rx
```

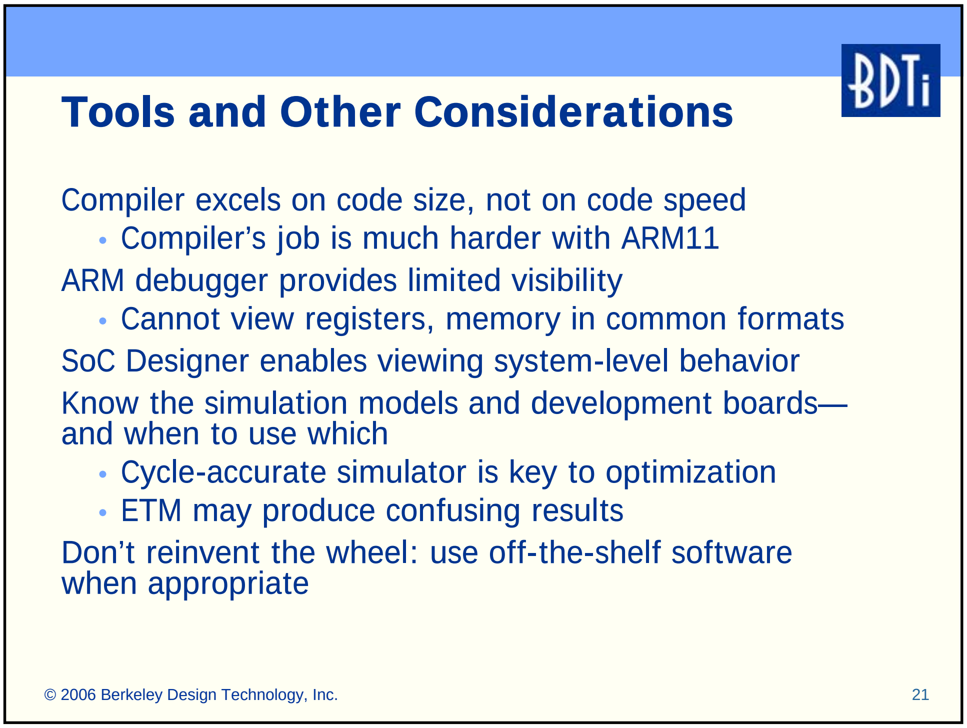
```
smlad rx, rx, rx, rx
smlad rx, rx, rx, rx
smlad rx, rx, rx, rx
ldrd  rx, [rx], #x
ldrd  rx, [rx, #x]
smlad rx, rx, rx, rx
smlad rx, rx, rx, rx
smlad rx, rx, rx, rx
smlad rx, rx, rx, rx
ldrd  rx, [rx, #x]
smlad rx, rx, rx, rx
smlad rx, rx, rx, rx
smlad rx, rx, rx, rx
smlad rx, rx, rx, rx
ldrd  rx, [rx], #-x
ldrd  rx, [rx, #x]
smlad rx, rx, rx, rx
smlad rx, rx, rx, rx
```

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

64 taps in 54 cycles = 1.19 taps/cycle

© 2006 Berkeley Design Technology, Inc.

20



Tools and Other Considerations

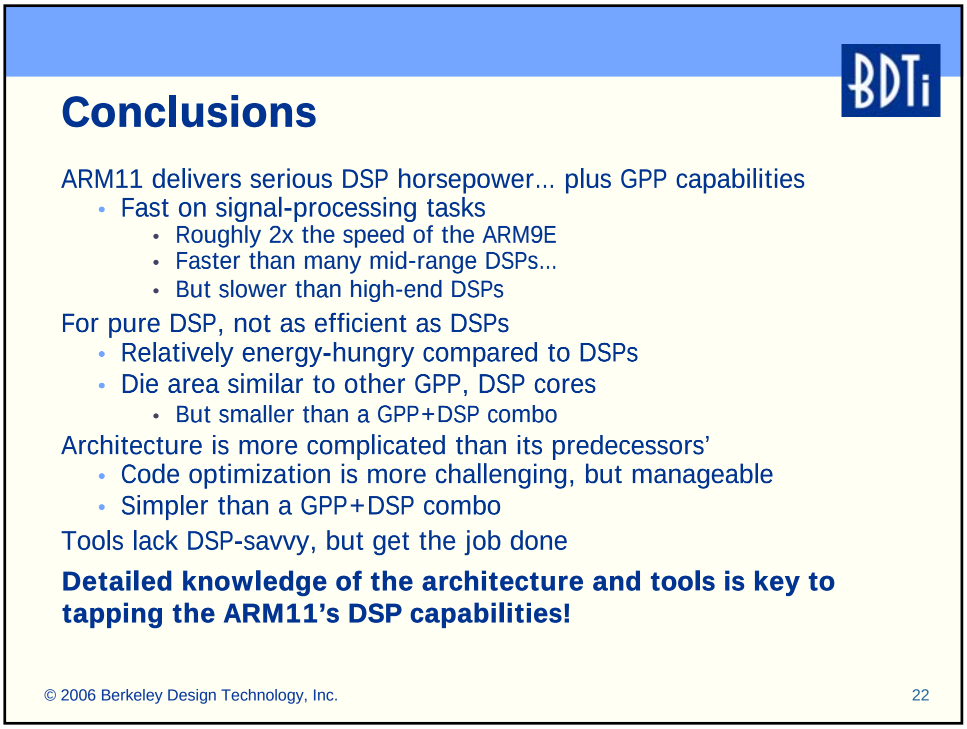
Compiler excels on code size, not on code speed
- Compiler's job is much harder with ARM11

ARM debugger provides limited visibility
- Cannot view registers, memory in common formats

SoC Designer enables viewing system-level behavior

Know the simulation models and development boards—and when to use which
- Cycle-accurate simulator is key to optimization
- ETM may produce confusing results

Don't reinvent the wheel: use off-the-shelf software when appropriate



Conclusions

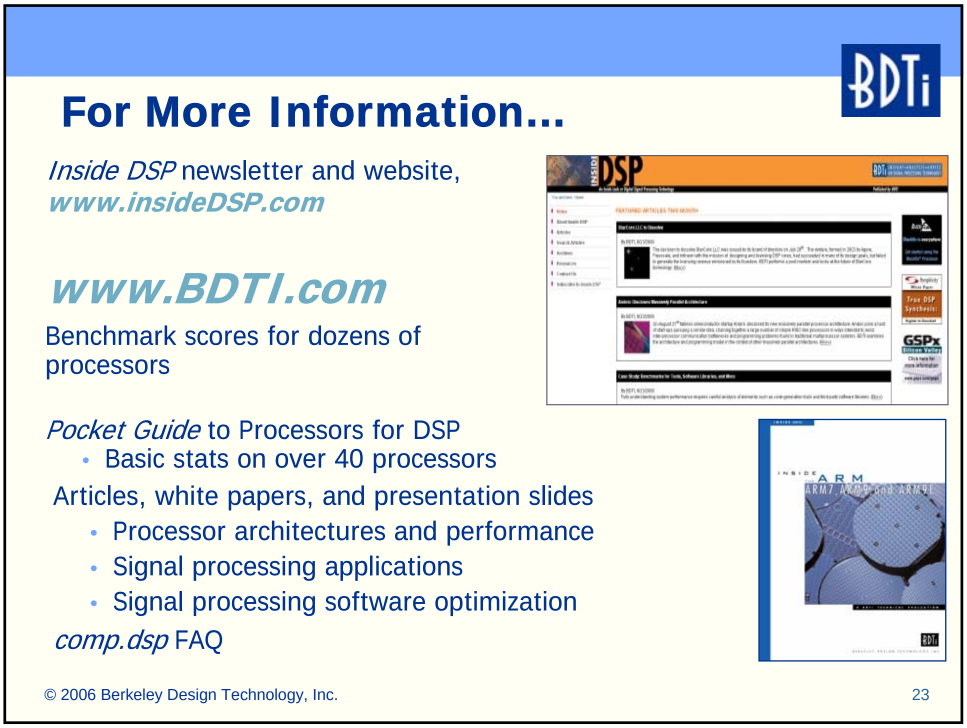
ARM11 delivers serious DSP horsepower... plus GPP capabilities
- Fast on signal-processing tasks
  - Roughly 2x the speed of the ARM9E
  - Faster than many mid-range DSPs...
  - But slower than high-end DSPs

For pure DSP, not as efficient as DSPs
- Relatively energy-hungry compared to DSPs
- Die area similar to other GPP, DSP cores
  - But smaller than a GPP+DSP combo

Architecture is more complicated than its predecessors'
- Code optimization is more challenging, but manageable
- Simpler than a GPP+DSP combo

Tools lack DSP-savvy, but get the job done

Detailed knowledge of the architecture and tools is key to tapping the ARM11's DSP capabilities!



INSIDE DSP
INSIDE ARM
ARM7, ARM9 and ARM9E
BDTi

For More Information…

Inside DSP newsletter and website, *www.insideDSP.com*

www.BDTI.com

Benchmark scores for dozens of processors

Pocket Guide to Processors for DSP

- Basic stats on over 40 processors

Articles, white papers, and presentation slides

- Processor architectures and performance
- Signal processing applications
- Signal processing software optimization

comp.dsp FAQ

23