



The Sandbridge Sandblaster SB3000 Multithreaded CMP Platform

John Glossner, Ph.D., Co-Founder, CTO & EVP

jglossner@SandbridgeTech.com

1 North Lexington Ave, 10th Floor
White Plains, New York 10601
914-287-8500

Agenda

Sandbridge Introduction

- ➔ **Company Background**
- ➔ **Motivation**

Sandblaster Platform

- ➔ **ISA**
- ➔ **Vector Architecture**
- ➔ **Multithreading**
- ➔ **Saturating arithmetic**

Hardware

- ➔ **Low Power Core**
- ➔ **SB3000**
- ➔ **RF**

Software

- ➔ **IDE**
- ➔ **Compiler**
- ➔ **Simulator**

Communications

- ➔ **WCDMA, GPRS**
- ➔ **GPS**
- ➔ **802.11b**

Applications

- ➔ **H.264 / MPEG4**
- ➔ **MP3**

Summary

Sandbridge Technologies

Fabless
Semiconductor
Company

developing...

Software
reconfigurable

Wireless chipsets
for low power
applications

WORLD
ECONOMIC
FORUM

COMMITTED TO
IMPROVING THE STATE
OF THE WORLD

E-Gang
Bridging The Gaps

Scott Woolley, 09.01.03

Guenter Weinberger
CEO, Sandbridge Technologies

EGANG
Wireless
Visionaries

[Click Here For Full Coverage](#)

Introduction:

• The Wonderful World Of
Wirelessness

Guenter Weinberger:

• Overcoming Incompatible
Standards

Stephen Tann:

Guenter Weinberger runs a company that is barely two years old, has only 40 employees and has yet to earn a dime. But, boy, does this guy have big dreams. "I don't want to appear too far away from reality," he says, "but we have the technology to become the next Intel."

What 78,000-employee Intel is to the PC industry, Weinberger thinks his shop, **Sandbridge Technologies**, can be to the cell phone industry—which spent \$20 billion on chips last year.

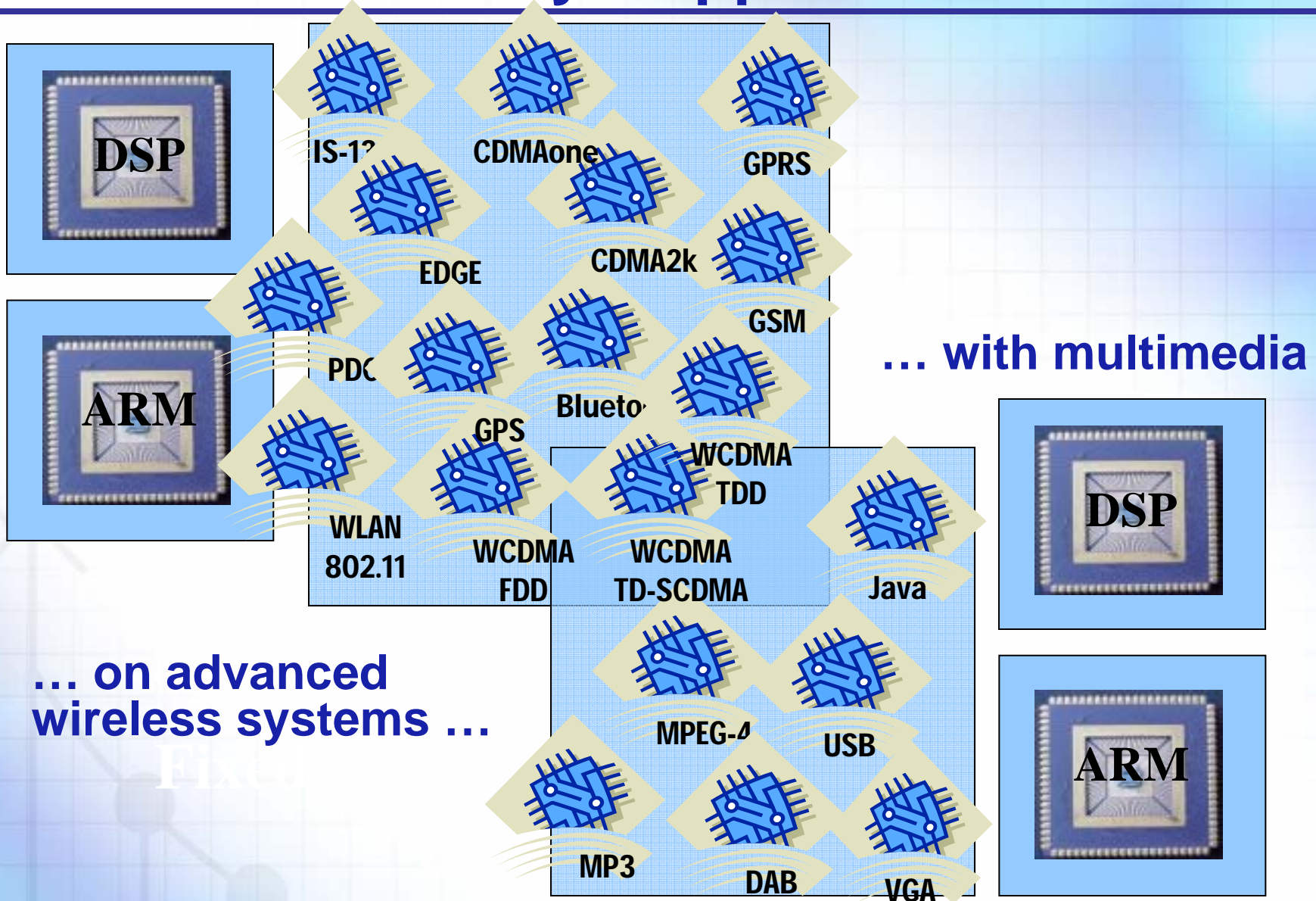
He aims to achieve this audacious feat by solving one of the most glaring—and annoying—problems for cell customers: the alphabet soup of incompatible standards that hinders a multitude of networks from easily hooking up with one another. Today's global traveler must lug different phones for different countries (and endure incompatible voice mail boxes and different phone numbers and bills).

MPSoC
July 2005

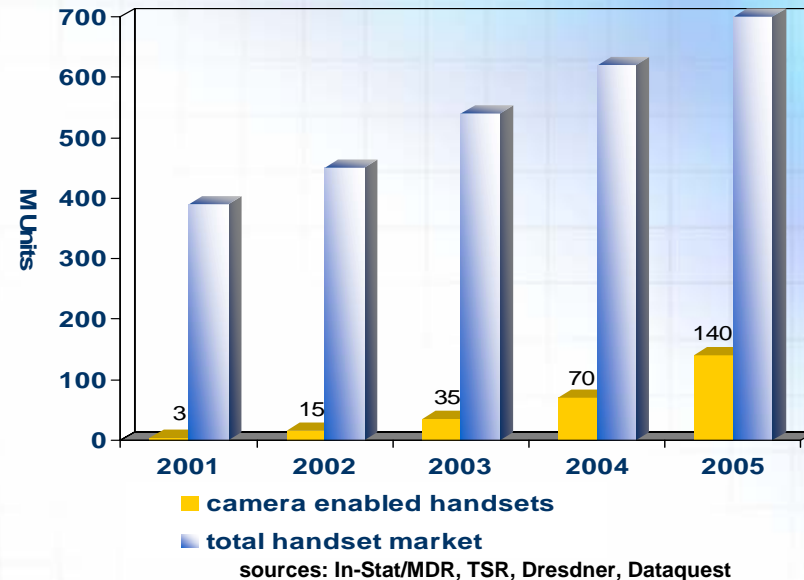
SANDBRIDGE
TECHNOLOGIES



A Whole Industry's approach failed ...



What is it that we eventually carry with us ?

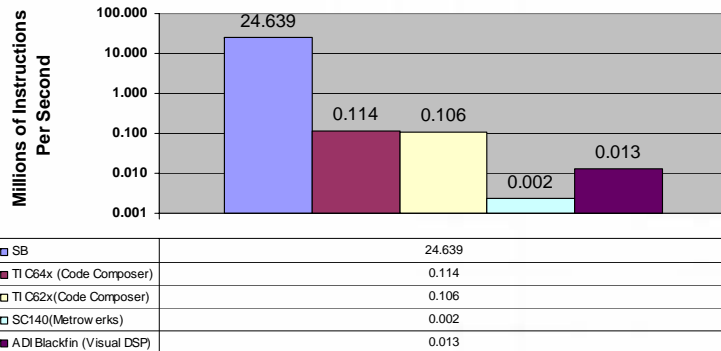


**It has a color screen, camera, audio and antenna ...
... but all features need high computing performance
and ultra low power consumption**

- **Wireless communication 2G – 2.5G – 3G – WLAN – BT – etc.**
 - GSM/IS-95a,b/IS-136/PDC/ iDEN – CDMA2k/GPRS/EDGE – FDD/TDD/TD-SCDMA/Jap.WCDMA/CDMA2k-3x– 802.11a,b,g
- **Radio broadcast GPS – radio – TV – etc.**
 - Location based services/911/tracking services – AM/FM/DAB – Sat./Terr.TV
- **Encryption – decryption – media encode – media decode**
- **Games – speech to text – natural language processing**

Handset design at C level in min. time

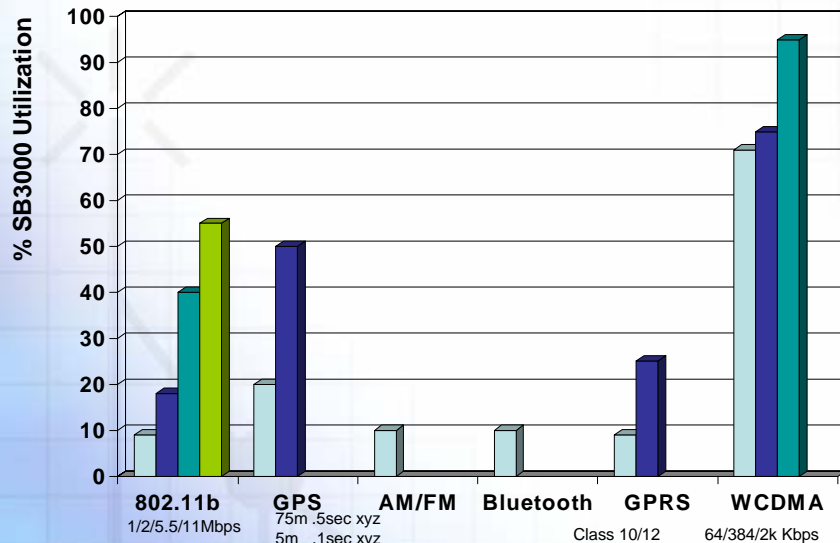
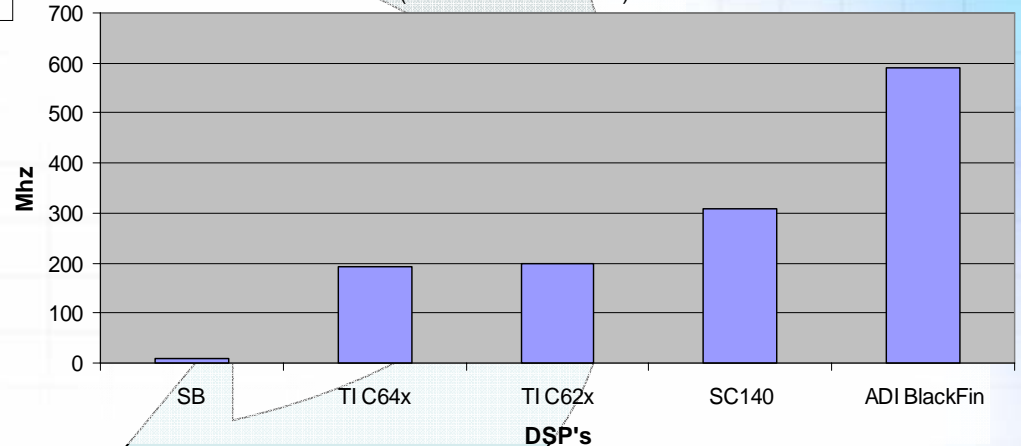
Simulation Speed
(1GHz Laptop)



Complete system design in C

- Cycle accurate simulator delivers immediate feedback
- Design for conformance in C

AMR Encoder
(out-of-the-box C code)

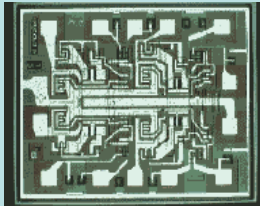


System performance by design

- Parallelizing compiler generates production assembly code
- Multithreading (sea-of-threads) ensures concurrent execution

The Sandbridge Approach ...

SandBlaster™ DSP



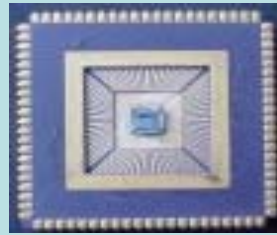
- Programmable
- Ultra-low power
- High-performance
- Multithreaded

SandBlaster™ Tools



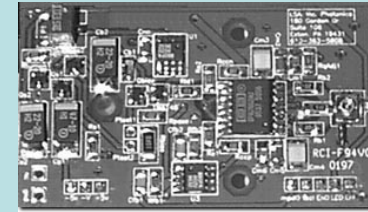
- Improved productivity C compiler
- 70% reduction In time-to-market
- User-friendly

DSP Platform



- Scalable & Programmable
- Integrated Sandblaster cores
- Up to 2Mbit/sec data rate
- 40,000 RISC MIPS
- Low Cost 0.13um CMOS
- Integrated protocol stack

DSP Ref Design



- Low Cost
- Power Efficient
- Ultra-high performance
- Fully tested / validated
- Dedicated Customer Support
- Flexible and upgradeable

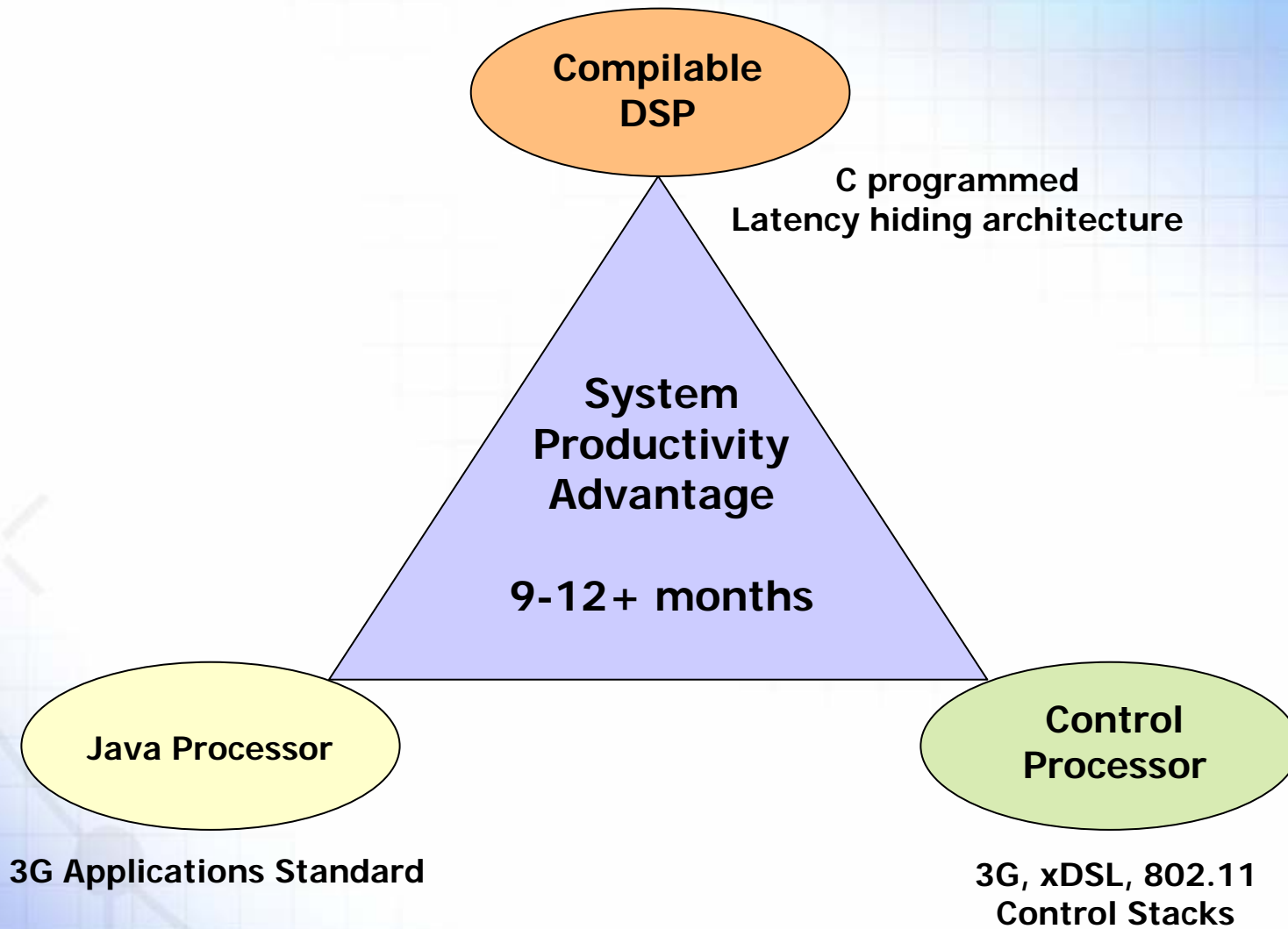
Core technology equally applicable to Networking, Storage, Automotive, GP-DSP, etc.



Architecture



Sandblaster Architecture Performs



Multithreaded Architecture Enables C

Key to Low Power Implementation

Code&Data Sharing Across Threads

Sea of Threads

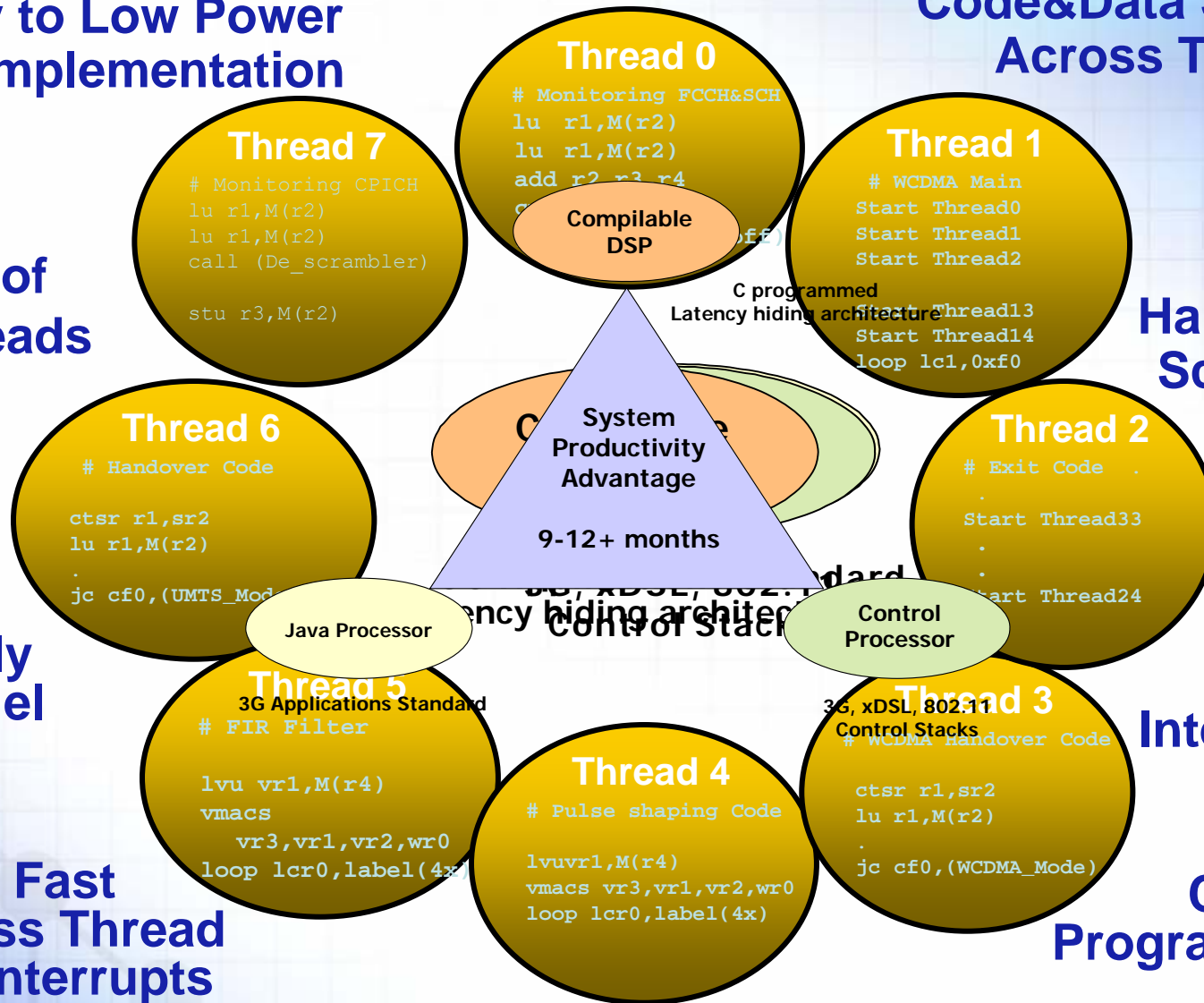
Hardware Scheduled

Highly Parallel

Fully Interlocked

Fast Cross Thread Interrupts

C Programmed



Parallelism

Multiple cores (MP)

➔ 4 cores

Multithreaded (TLP)

➔ 8 threads/core

Compound Instructions (ILP)

➔ 3 operations out of

- Integer
- Load/Store
- Branch
- Vector

Vector (DLP)

➔ 4 data parallel operations

Performance

Peak

- ➔ 3 operations/cycle
- ➔ 16 RISC-ops/cycle
- ➔ 4 MACS/cycle

Example

```
L0: lvu %vr0,%r3,8
    || vmulreds %ac0,%vr0,%vr0,%ac0
    || loop %lc0,L0
```

- ➔ load vector update: 4 16-bit loads + address update
- ➔ vector multiply and reduce: 4-16 bit saturating multiplies + 4 32-bit saturating adds
- ➔ loop: decrement, compare against zero and branch

20 tap FIR

- ➔ 3.92 taps/cycle sustained including automatic multithreading
- ➔ ~16 RISC-ops/cycle sustained

Saturating Arithmetic

Many DSP applications require saturating arithmetic

Saturation means

- Results greater than largest representable number are saturated to the largest representable number
- Results less than the smallest representable number are saturated to the smallest representable number

- 4-bit precision example:

$$A = 0.101 = 0.625$$

$$+ B = 0.111 = 0.875$$

$$= S = 01.100 = 1.5$$

$$S | = 1.100 = -.5$$

$$\langle S \rangle = 0.111 = 0.875$$

$$A = 1.011 = -0.625$$

$$+ B = 1.001 = -0.875$$

$$= S = 10.100 = -1.5$$

$$S | = 0.100 = +.5$$

$$\langle S \rangle = 1.000 = -1.0$$

Saturating Arithmetic (2)

Saturating arithmetic operations are not associative

4-bit precision example

$$\begin{aligned} & \langle \langle \langle -1.0 * -1.0 \rangle + \langle 0.5 * 0.5 \rangle \rangle + \langle -0.5 * 0.5 \rangle \rangle \\ &= \langle \langle 0.875 + 0.25 \rangle - 0.25 \rangle \\ &= 0.875 - 0.25 = 0.625 \end{aligned}$$

$$\begin{aligned} & \langle \langle -1.0 * -1.0 \rangle + \langle \langle 0.5 * 0.5 \rangle + \langle -0.5 * 0.5 \rangle \rangle \rangle \\ &= \langle 0.875 + \langle 0.25 - 0.25 \rangle \rangle \\ &= 0.875 + 0 = 0.875 \end{aligned}$$



Sandbridge Low Power Hardware

Low Power IDLE Instructions

Architecturally possible to

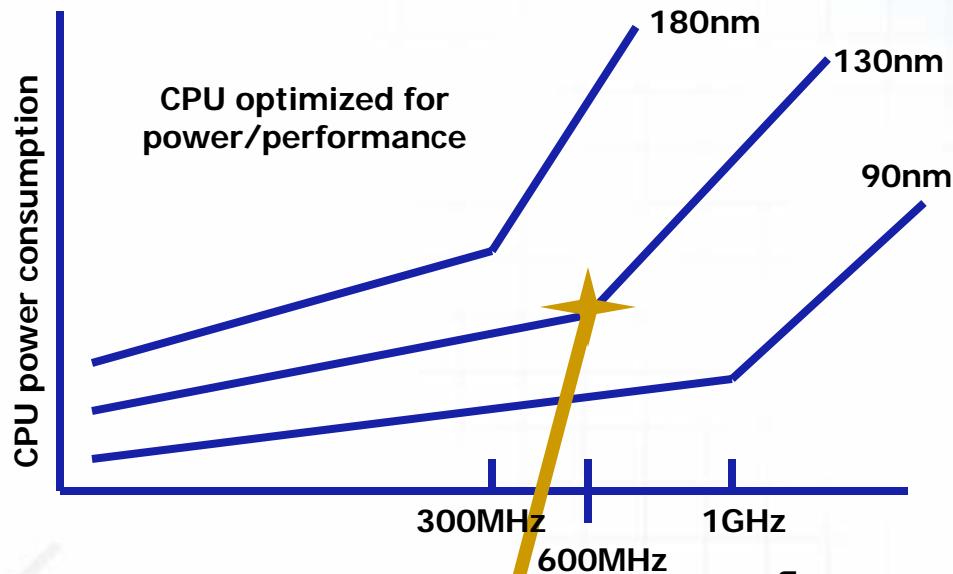
➔ Turn off 1 or more processors

- All clocks disabled
- Instruction fetch disabled
- Memory disabled
- Memory state not preserved

➔ Turn off 1 or more threads within a processor

- Clocks disabled on a per thread basis
- Instruction fetch disabled on a per thread basis
- Memory state preserved (including registers)
- Threads awaken via interrupt

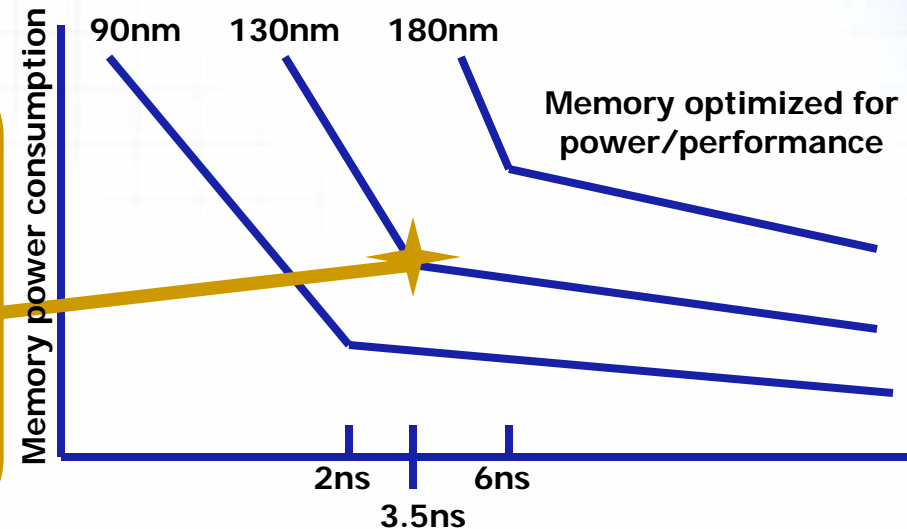
Using Multithreading to Optimize for Power



- Multiple cycles to access memory
- Slow memory accesses hidden by multithreading

Multithreading decouples CPU from MEM, combining best power/performance for CPU AND MEM

- Optimized CPU power
- Optimized memory power
- Optimized overall power/performance



SandBlaster Pipeline

	0	1	2	3	4	5	6	7	8	9	10
Ld/St	Inst Dec	RF Read	Agen	Xfer	Int. Ext	Mem 0	Mem 1	Mem 2	WB		
ALU	Inst Dec	Wait	RF Read	Exec1	Exec2	Xfer	WB				
I_Mul	Inst Dec	Wait	RF Read	Exec1	Exec2	Exec3	Xfer	WB			
V_Mul	Inst Dec	RF Read	MPY1	MPY2	Add1	Add2	Xfer	WB			
V_Mul Reduce	Inst Dec	RF Read	MPY1	MPY2	Add1	Add2 RF Rd	Reduce 1	Reduce 2	Reduce 3	Reduce 4	WB

Staggered Read/Write

- ➔ Allows single write-port register files

Very Long Reduce

- ➔ Shifted/Offset against vector pipe

Interlock Checking Hardware

The multithreaded implementation is transparent

- ➔ **No interrupt restrictions**
- ➔ **Load / Branch delay slots not visible**

Multithreading hides instruction execution latencies

No interlock checking hardware is required

- ➔ **One exception – long loads**

Single Register File Write Ports

A single compound instruction can contain a vector load or store and a vector operation.

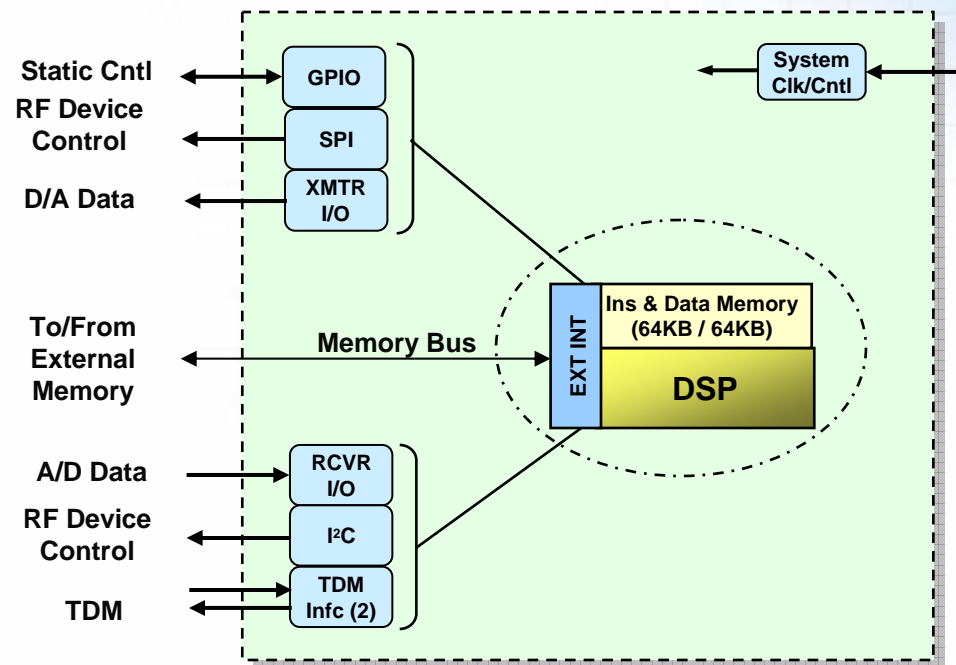
```
stvu    %vr1, r0, 8 || vmac %vr5, %vr4, %vr3, %ac0
```

- Our design requires a single write port per space
 - int 1R/1R, vec 2R/1W, and acc 1R
- VLIW's may require 14R/5W for the same computation
- If load, up to 9W simultaneous write ports may be required

Our design staggers Load/Stores in multiple ways:

- Time staggered
 - different pipeline stages
- Spatially staggered
 - Banked register files
- Architecturally staggered
 - Separate architected register spaces (e.g. Integer, Vector, Accumulator)

SBTC 8/02



- 0.18um CMOS ASIC
- Single DSP Core
- SW Programmable
- External Bus for L2 memory
- Internal Inst/Data memory
- Control Interfaces: I²C, SPI, TDM, A/D, D/A

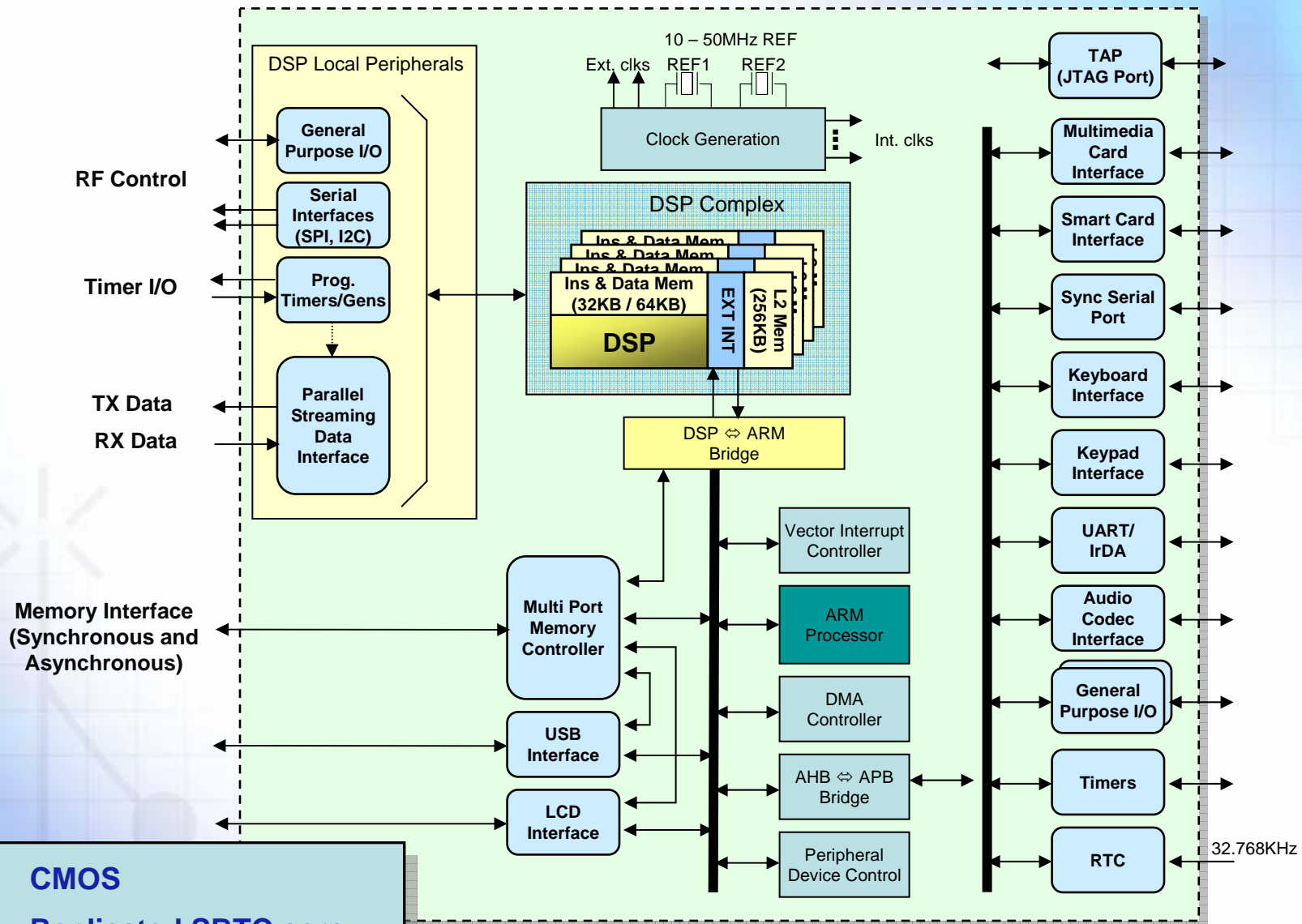
SBTC Digital Card



MPSoC
July 2005

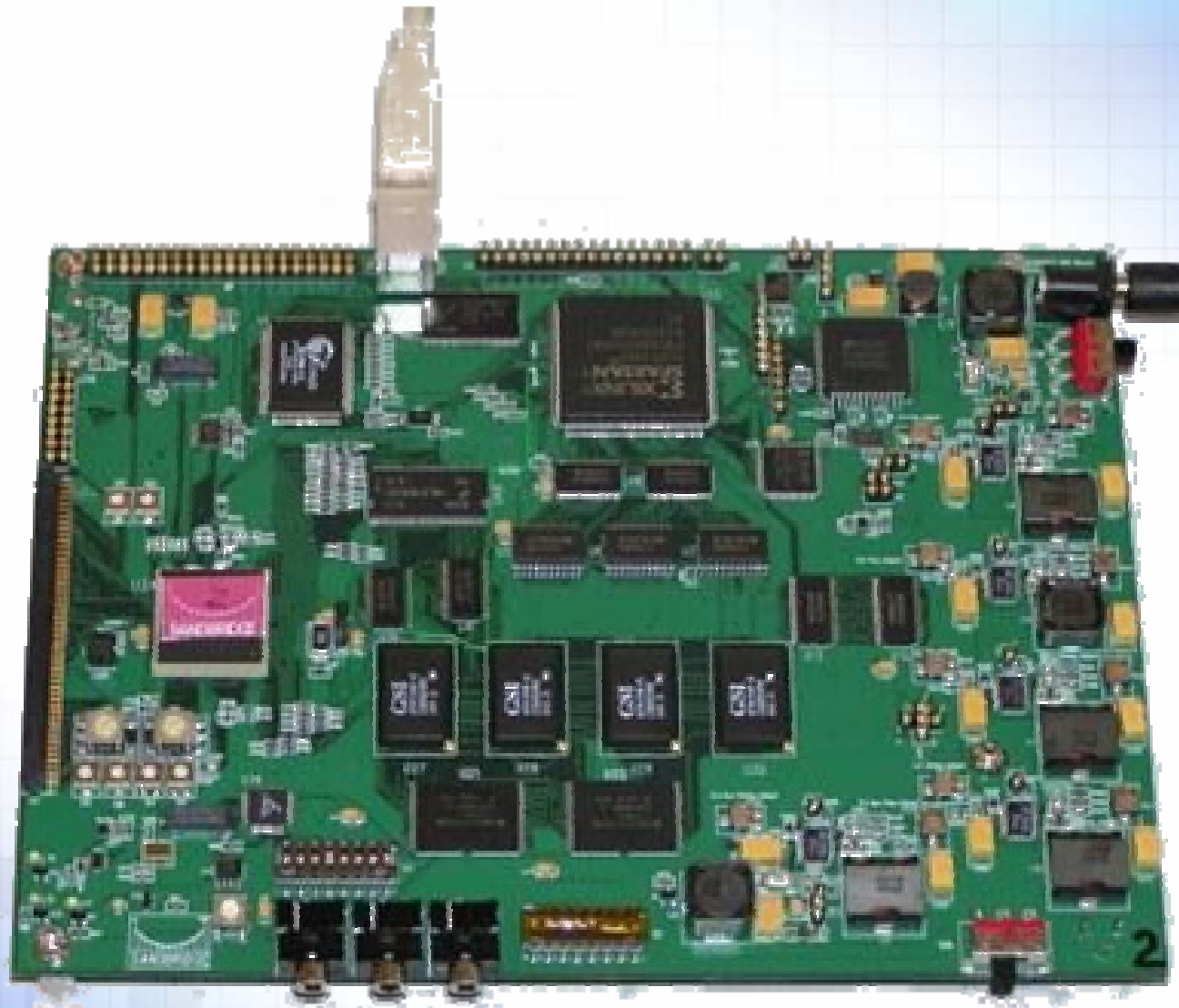
SANDBRIDGE
TECHNOLOGIES 

SB3000 Handset Chip



- CMOS
- Replicated SBTC core
- Low Power design

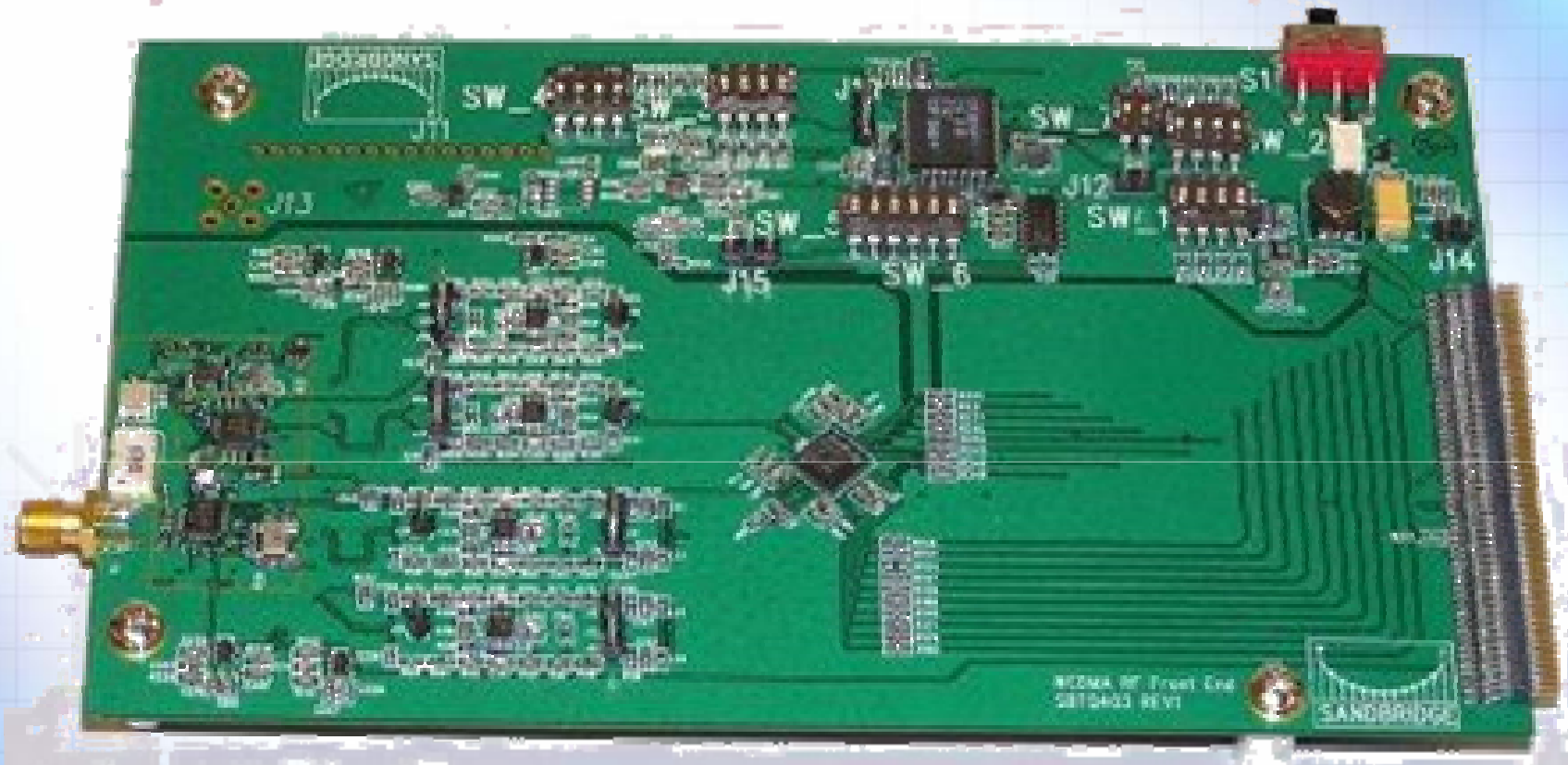
SB3000 Digital Card



MPSoC
July 2005

SANDBRIDGE
TECHNOLOGIES 

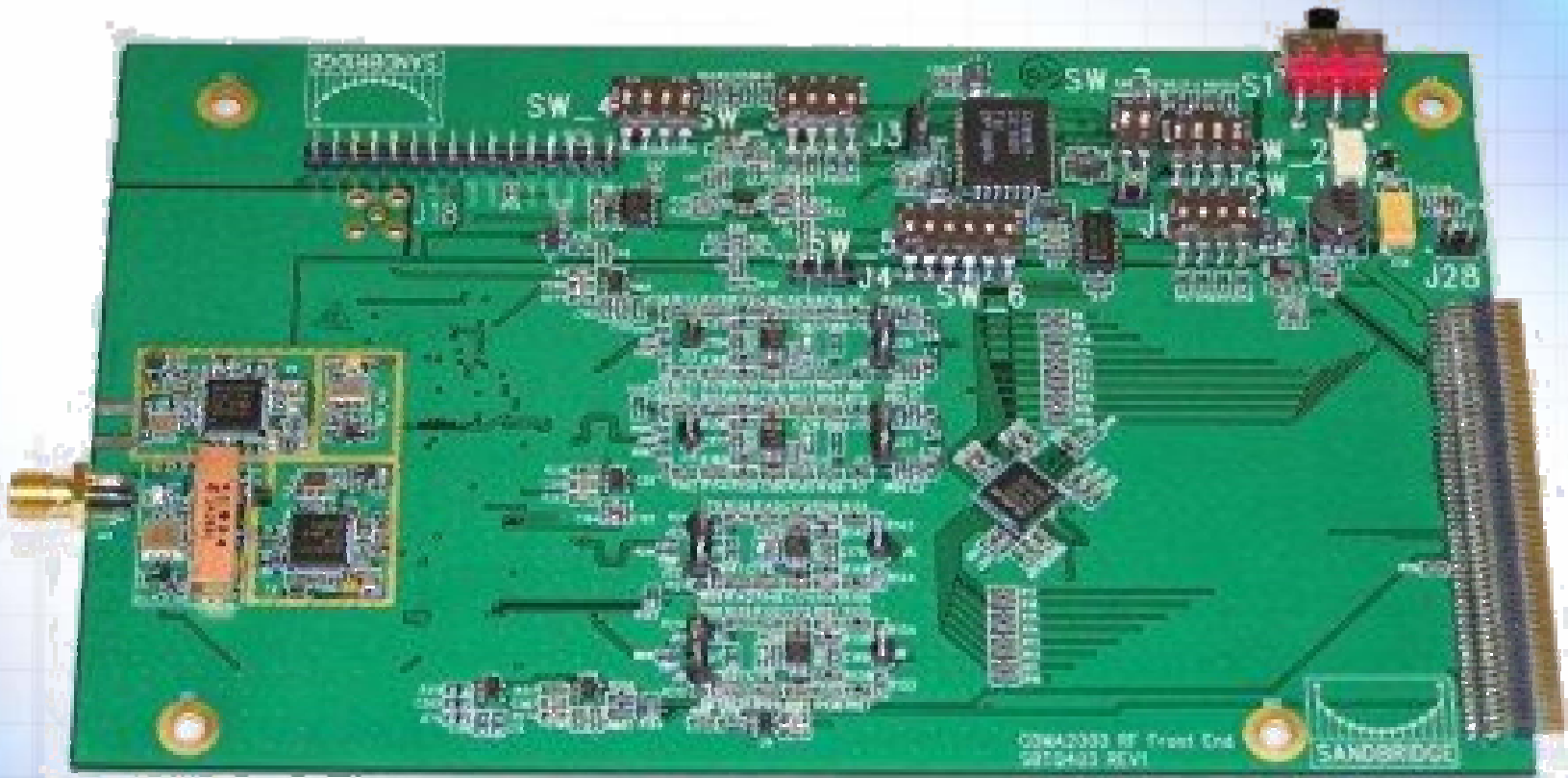
WCDMA 2Mbps Front End Card



MPSoC
July 2005

SANDBRIDGE
TECHNOLOGIES

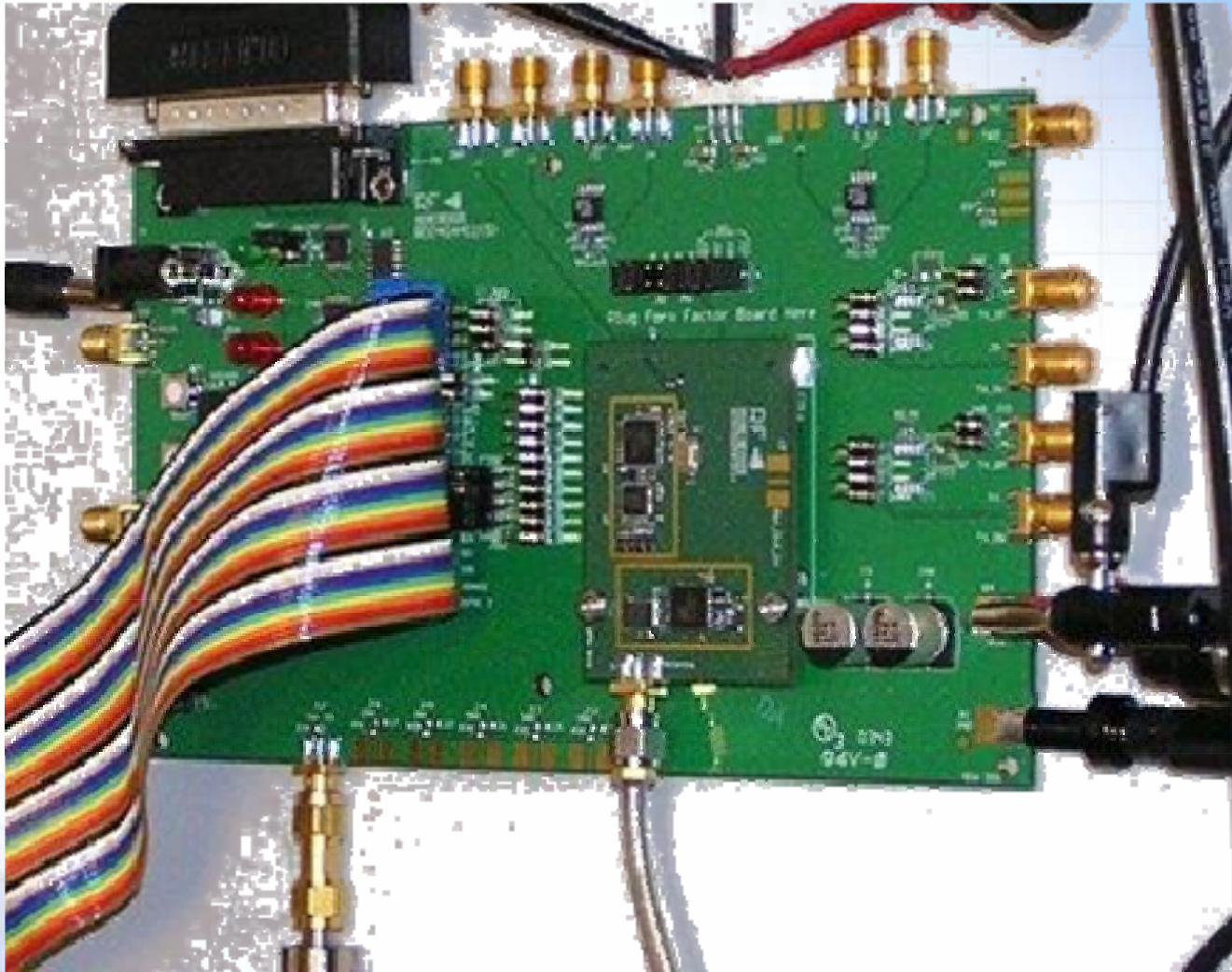
CDMA-2k Front End Board



MPSoC
July 2005

SANDBRIDGE
TECHNOLOGIES 

GSM/GPRS Front End Card



MPSoC
July 2005

SANDBRIDGE
TECHNOLOGIES 

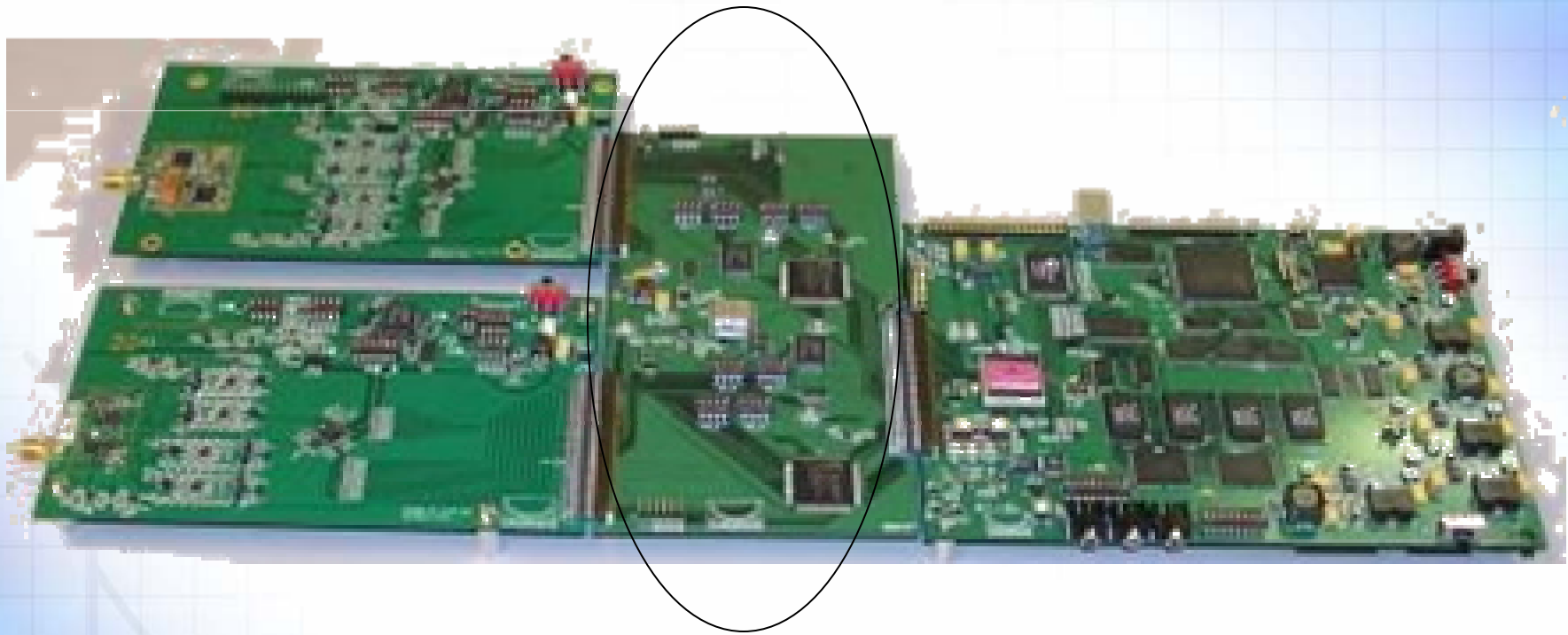
802.11b WLAN Front End



MPSoC
July 2005

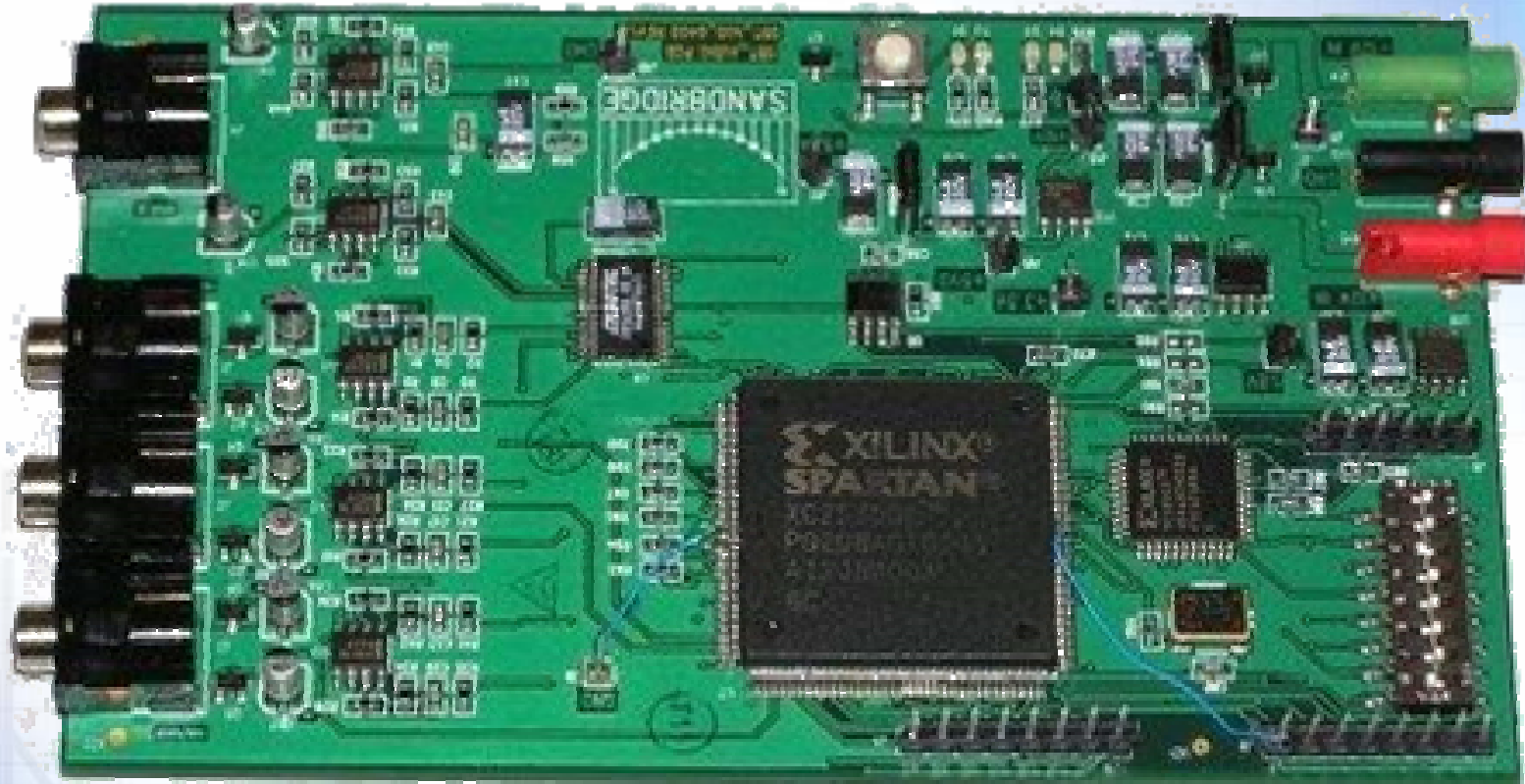
SANDBRIDGE
TECHNOLOGIES 

Multiplexer Board Block Diagram



MPSoC
July 2005

Audio Board



MPSoC
July 2005

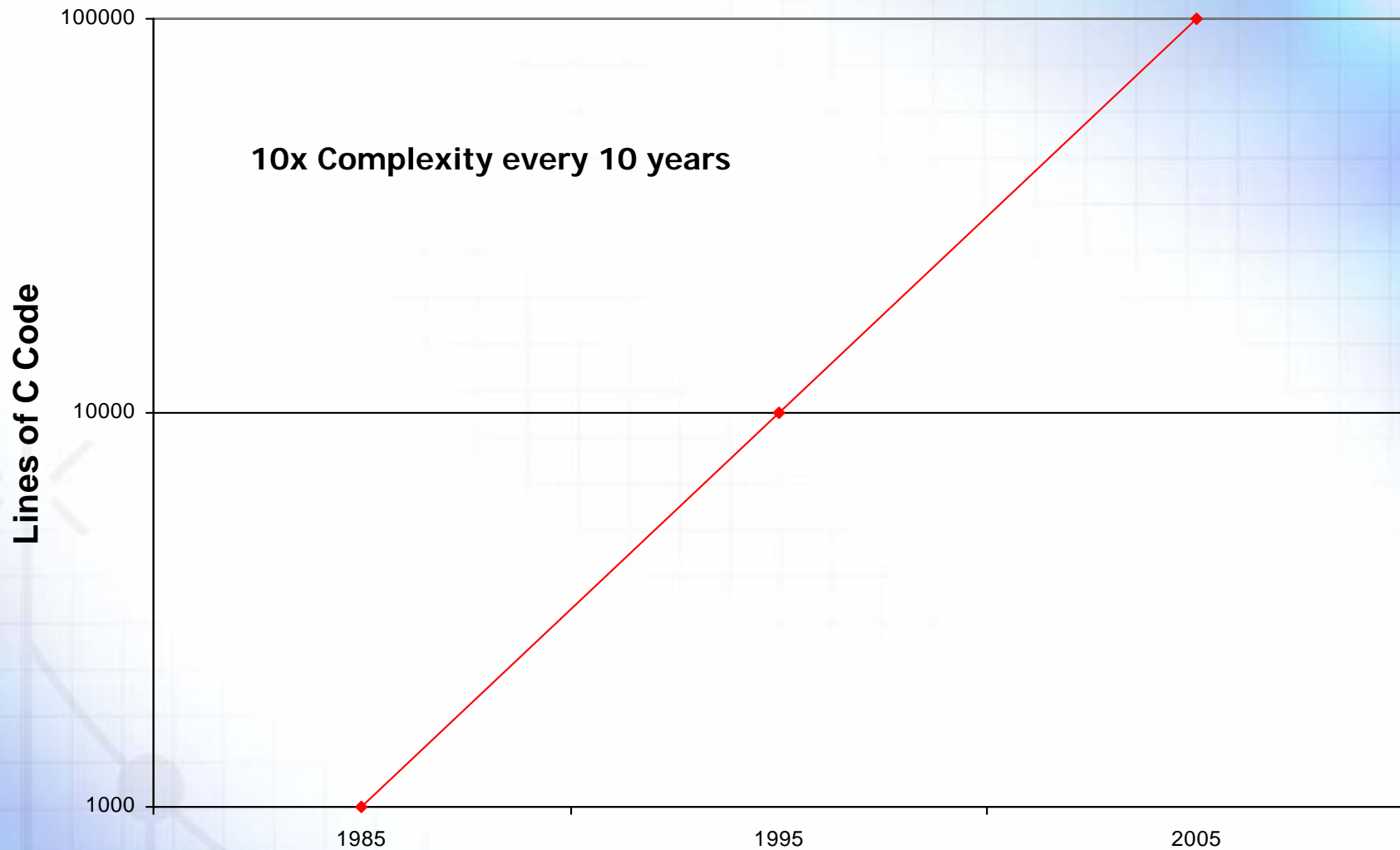
SANDBRIDGE
TECHNOLOGIES 



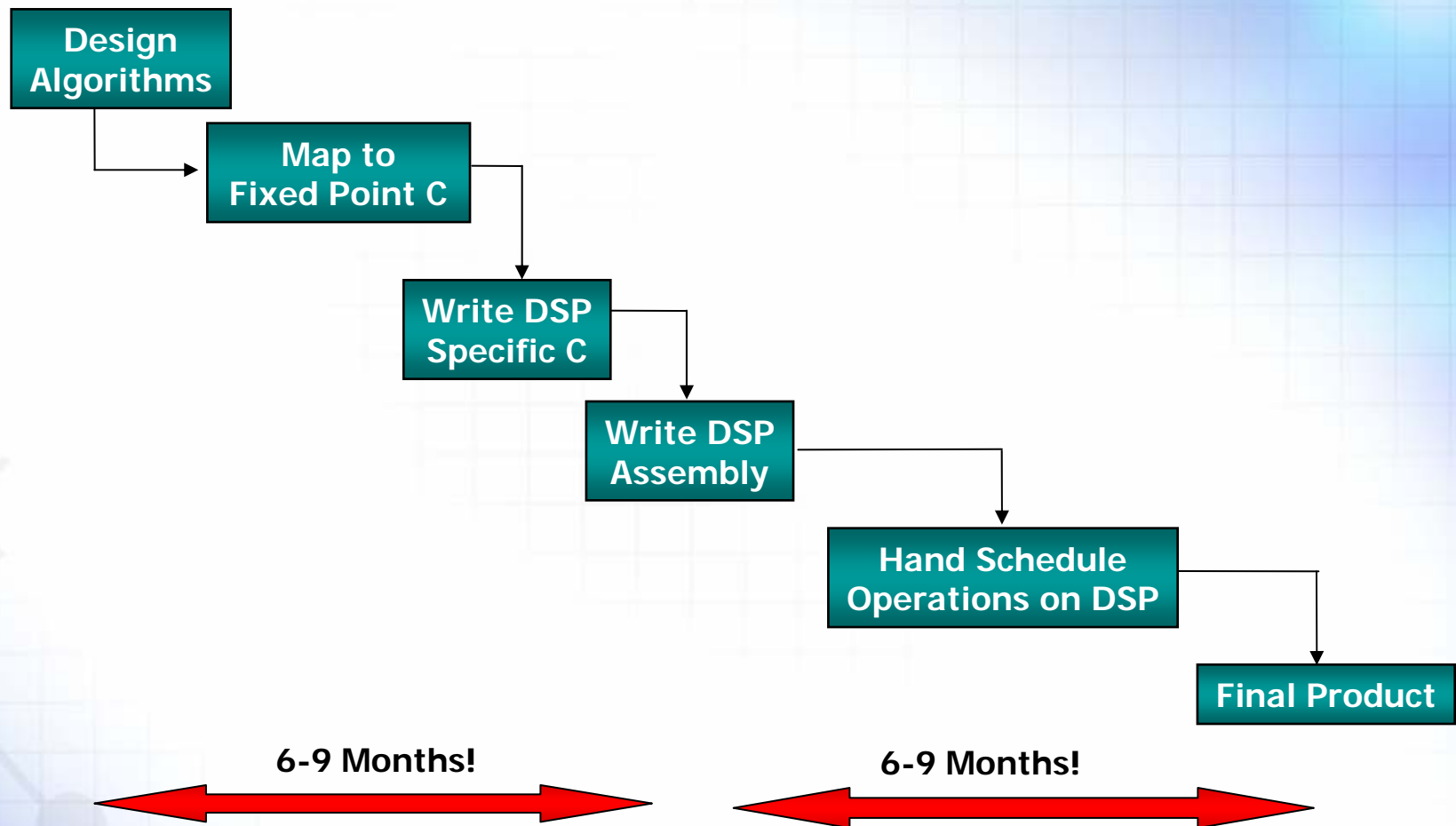
Sandbridge Software Tools



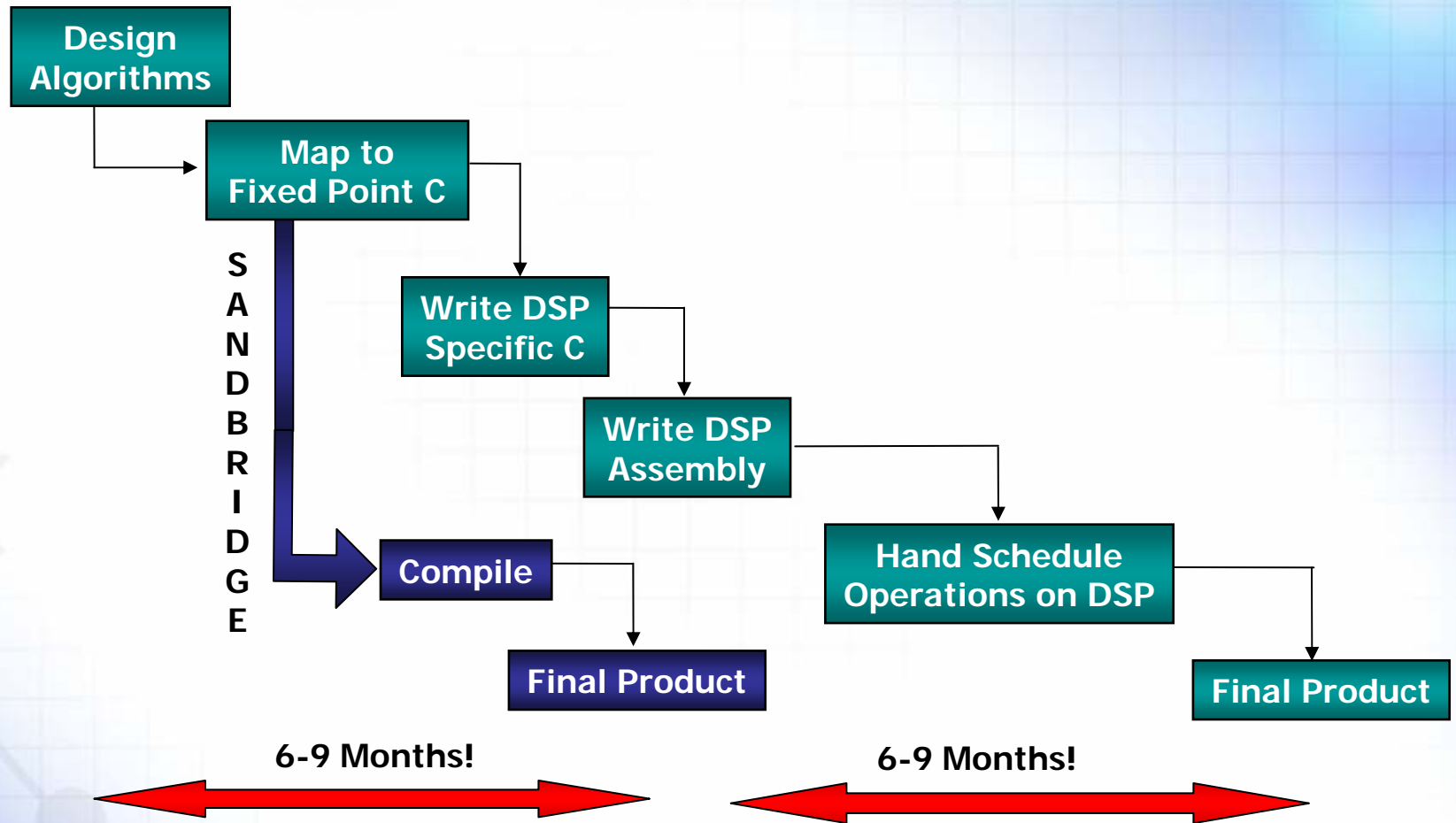
DSP Application Complexity



Compiler saves R&D and time-to-market ..



Compiler saves R&D and time-to-market ...



Sandblaster™ Provides Dramatic Improvement

System Software

Compilation Tools

- **Compiler**
- **Assembler**
- **Linker**
- **Loader**

Library

- **C library**
 - Standard C & Math
- **Device drivers**

Simulator

- **Just-in-time**
 - Models peripherals
- **Cycle-accurate C**
- **Cycle-accurate VHDL**

IDE

- **Netbeans based**
- **Integrated S/W debug**

RTOS

- **Light-weight kernel**
 - Based on POSIX API
 - Filesystem

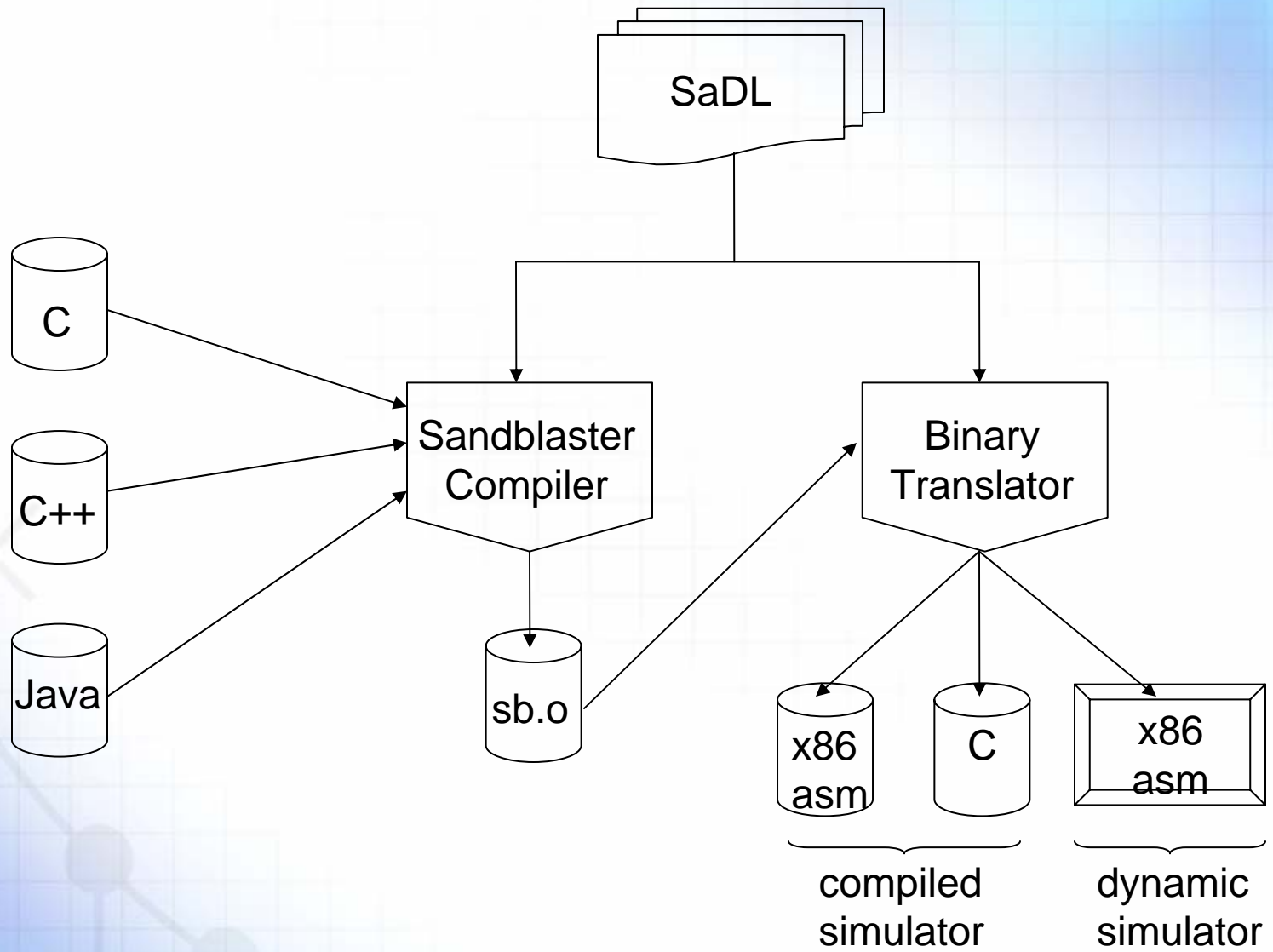
Test-Cases

- **DSP kernels**
- **DSP applications**
- **Commercial test-suites**
 - Plum-Hall, Perennial, Nullstone, CosY
- **Nightly builds**

H/W Debugger

- **Breakpoint/profile**
- **JTAG**

Sandblaster Tools



Compiler Optimizations – Dragon Book +

DSP Optimizations

Saturation Arithmetic
Fixed Point Semantic Analysis
Bit-exact ETSI compliance

Vector Optimizations

Vector Loads
Vector Stores
Vector Arithmetic
Vector Reduction
Saturating Vector Operations

Multithreaded Optimizations

OpenMP
Automatic Parallelization
Automatic Multithreading

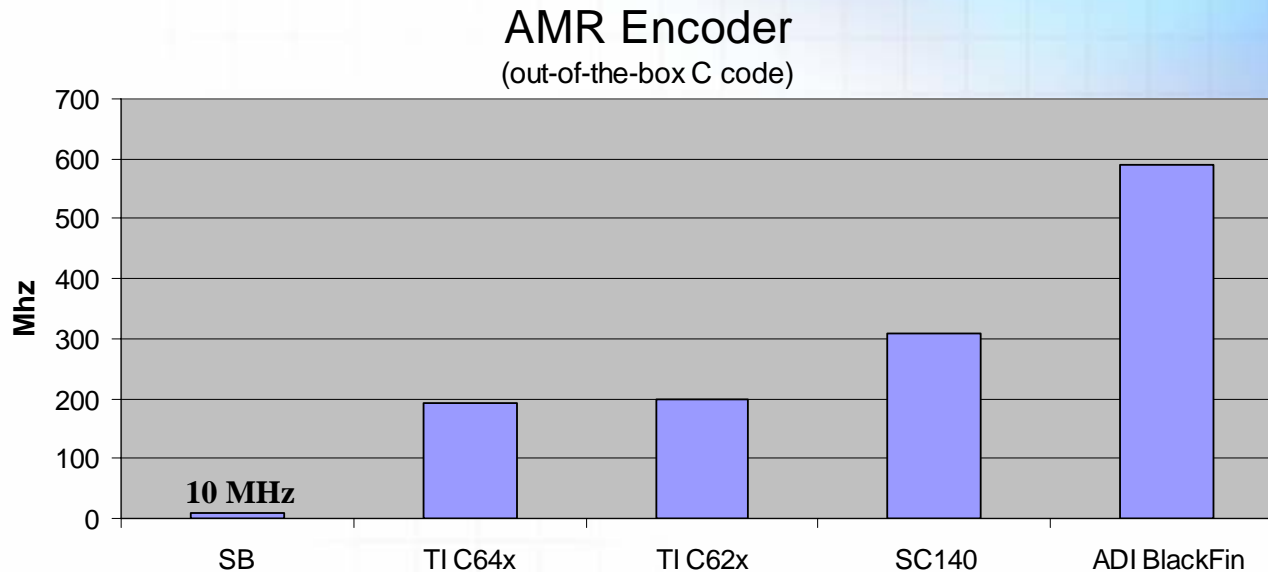
Loop Optimizations

Loop Invariant Code Motion
Strength Reduction
Induction Variable Elimination
Loop Splitting
Software Pipelining

Interprocedural Opts.

Constant Propagation
Memory Disambiguation
Function Inlining
Alias Analysis

Sandblaster AMR Results

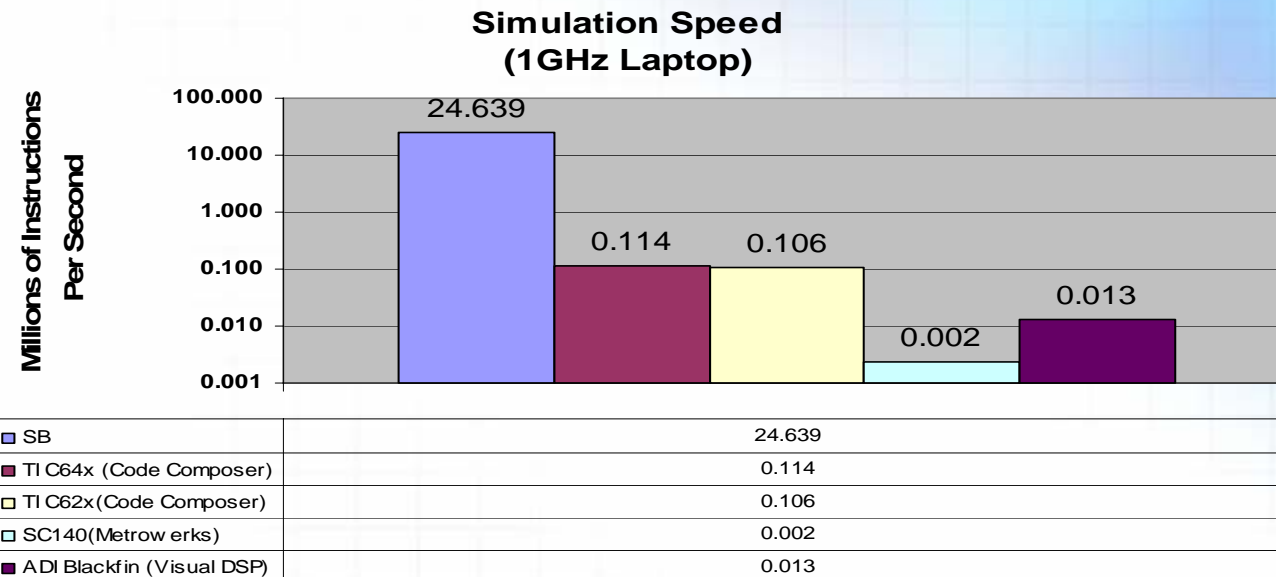


Programmed in C or Java DSP's

- **Super-computer class compiler**
 - Vectorization
 - DSP instruction generation
- **Standard Library**
 - Printf();
- **POSIX pthreads or Java threads**
- **50k+ testcases used for validation**
 - Industry standards: Plum-Hall, perennial, nullstone

MPSoC
July 2005

Sandbridge AMR Simulation Results



Compiled Simulator

- **JIT “Flash” compilation**
- **Up to 100 MHz on high end x86**
- **Multi-threaded supported**

Up to 4 orders of magnitude faster

- **Dramatic development time reduction**
- **Significant productivity improvement**

Multithreaded Programming

Automatic Multithreading of DSP Kernels

- **Compiler can vectorize and multithread**
- **Uses pthreads as underlying infrastructure**

Multiple threads usage via pthreads library

- **POSIX API**
- **Complete support for thread management, synchronization, and communication**
- **Thread-safe version of the C library**
- **Entire coding is done in C**

Applications multithreading in Java

- **Inherently a multithreaded language**

Multithreaded H/W with multithreaded S/W

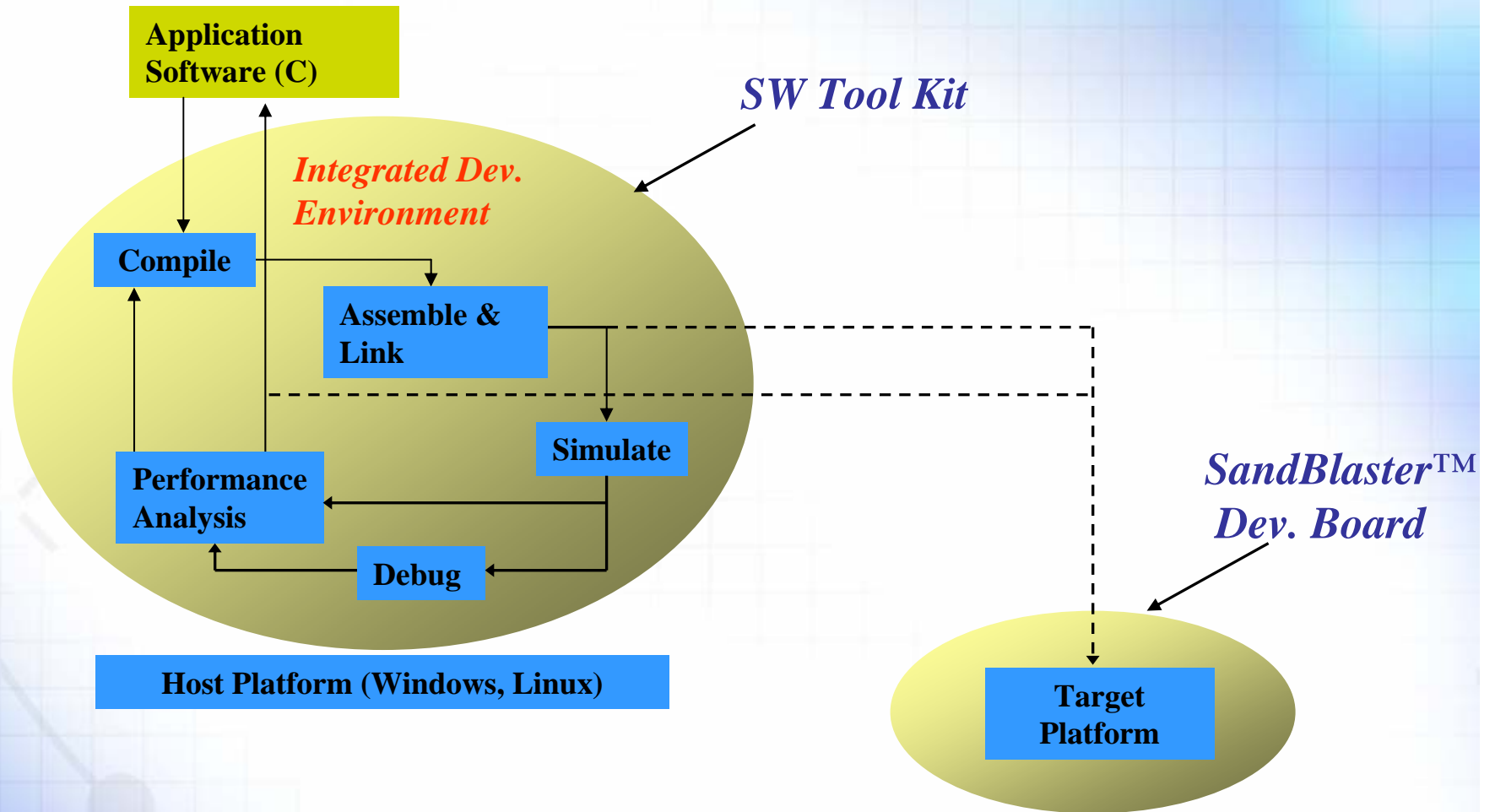
- **Automatic mapping of parallelism**
- **Easy parallel programming methodology**

Java Support

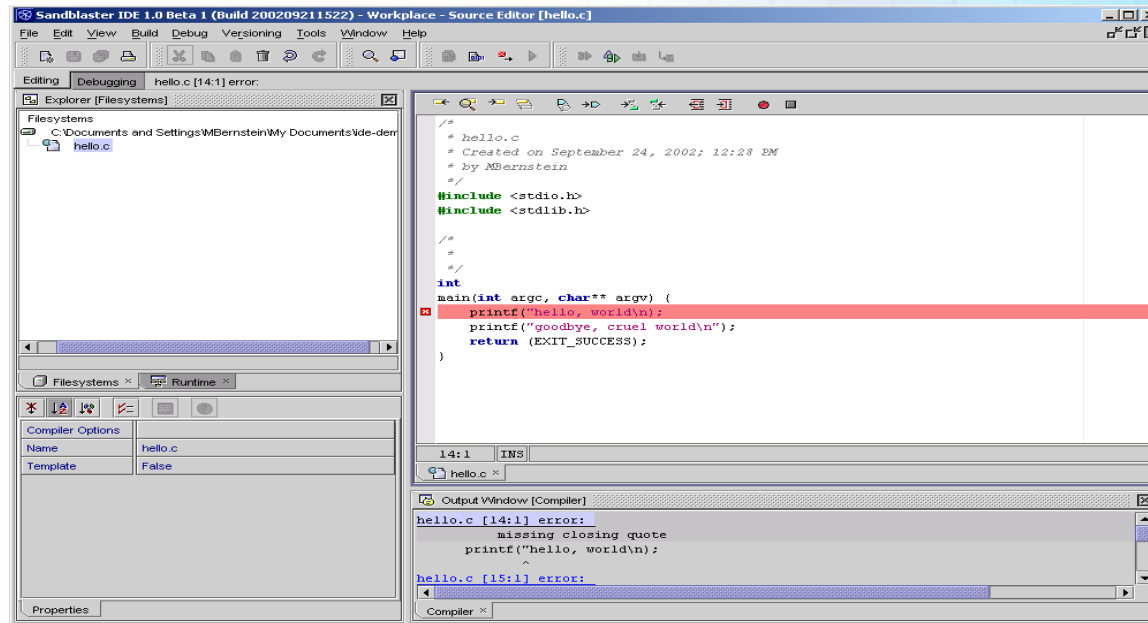
Java J2ME implementation

- **KVM 1.0 bytecode engine**
- **CLDC 1.0**
- **MIDP 1.0 support provided**
 - MIDP 2.0 in process
- **Multiple Java threads execute on multiple H/W thread units**
 - First known hardware multithreaded KVM
 - Sandblaster tools compile KVM with Java-specific optimizations
 - Java is another application on the Sandblaster processor
 - A java thread is scheduled on any available hw thread unit
 - Dynamic number of hardware thread units may be used
 - Synchronization mechanisms fully supported
 - Multithreaded garbage collection supported

Development Environment



Integrated Development Environment (IDE)



Based on Java open source netbeans

➔ **Enhanced with**

- C compilation and editing tools
- Source debugger
- Project management
- Scripting languages

Automatic Error recognition

Works in multiple languages too!

Variables, Threads and Memory View in Debugger

The screenshot displays the Sandblaster IDE 1.0 (Build 040206) - Source Editor [hello.c] interface. The main window shows the source code for a C program named 'hello.c'. The code includes standard headers and a main function that prints 'Hello!\n' and returns EXIT_SUCCESS.

The Debugger Window [Memory] is open, showing a memory view of the program's memory. The Start Address is set to 'argv' and the Byte Count is 256. The memory view shows a list of memory addresses and their corresponding values, all of which are 0x00000000.

The Threads window shows a single thread with ID 1, File 'hello.c', and Line 13.

The Watches window shows two watches: 'argv' with value 0x0 and 'argc' with value 0.

The Call Stack window shows the current call stack, with the top entry being 'main' in 'hello.c' at line 13. Other entries include 'osStartMain', 'osCallUser', 'pthread_tramp', 'osThreadTramp', and two unknown entries '??'.

```
/*  
 * hello.c  
 * Created on January 12, 2004; 1:29 PM  
 * by vkalashnikov  
 */  
#include <stdio.h>  
#include <stdlib.h>  
  
int  
main(int argc, char** argv) {  
    printf("Hello!\n");  
    return (EXIT_SUCCESS);  
}
```

Debugger Window [Memory]
Start Address: argv Evaluate
Byte Count: 256
00000000: 0000 0000 0000 0000 0000 0000
0000000C: 0000 0000 0000 0000 0000 0000
00000018: 0000 0000 0000 0000 0000 0000
00000024: 0000 0000 0000 0000 0000 0000
00000030: 0000 0000 0000 0000 0000 0000
0000003C: 0000 0000 0000 0000 0000 0000
00000048: 0000 0000 0000 0000 0000 0000
00000054: 0000 0000 0000 0000 0000 0000
00000060: 0000 0000 0000 0000 0000 0000
0000006C: 0000 0000 0000 0000 0000 0000
00000078: 0000 0000 0000 0000 0000 0000
00000084: 0000 0000 0000 0000 0000 0000
00000090: 0000 0000 0000 0000 0000 0000
0000009C: 0000 0000 0000 0000 0000 0000
000000A8: 0000 0000 0000 0000 0000 0000
000000B4: 0000 0000 0000 0000 0000 0000
000000C0: 0000 0000 0000 0000 0000 0000
000000CC: 0000 0000 0000 0000 0000 0000
000000D8: 0000 0000 0000 0000 0000 0000
000000E4: 0000 0000 0000 0000 0000 0000
000000F0: 0000 0000 0000 0000 0000 0000
000000FC: 0000 0000

Threads

*	ID	File	Line
	1	hello.c	13

Watches

Expression	Value
argv	0x0
argc	0

Call Stack

*	Function	File	Line
*	main	hello.c	13
	osStartMain	- unknown -	-
	osCallUser	- unknown -	-
	pthread_tramp	- unknown -	-
	osThreadTramp	- unknown -	-
	??	- unknown -	-
	??	- unknown -	-

Snapshot of Profile Information

Sandblaster IDE 1.0 (Build 040213) - Profiler [hello.stats]

File Edit View Build Debug Versioning Tools Window Help

Code completion DB has been created for the filesystem C:\Projects\sbapps.

Editing Debugging

Explorer [Filesystems]

Filesystems

- C:\Projects\sbapps
 - libug
 - dot
 - demos
 - ide tests
 - board
 - compilation
 - hello.c
 - hello.o
 - hello.sbx
 - hello.stats
 - infinite-loop.c
 - infinite-output-loop.c
 - infinite-scanf-loop.c
 - int-min.c
 - parity.c
 - parity.events
 - parity.events.data
 - parity.stats
 - print-args.c
 - print-args.mak
 - simple-output.c
 - simple-output.mak
 - simple-scanf.c
 - stderr-output.c
 - debugger
 - make

Files containing profile statistics data

Filesystems x

Name hello.stats

Template False

Properties

Summary Sections Functions Opcodes Cache Instructions

Function Statistics

Mode: User Match: <Any>

Function	v	Num.Calls(...)	Total Func...	Min. Func...	Max. Func...	Total Tree ...	Min. Tree ...	Max. Tree ...	Total Wall...	Min. Wall(U)	Max. v
		22	734	662	681				443058	441990	443
_exit	0	0	0	0	0	0	0	0	0	0	0
_divs	0	0	0	0	0	0	0	0	0	0	0
_error	1	6	6	6	6	6	6	6	6	6	6
_getcwd	0	0	0	0	0	0	0	0	0	0	0
_ldivr	0	0	0	0	0	0	0	0	0	0	0
_ldivrem	0	0	0	0	0	0	0	0	0	0	0
_ldivremu	0	0	0	0	0	0	0	0	0	0	0
_ldivs	0	0	0	0	0	0	0	0	0	0	0
_malloc_ato...	1	7	7	7	7	7	7	7	7	7	7
_malloc_ato...	0	0	0	0	0	0	0	0	0	0	0
_malloc_ato...	0	0	0	0	0	0	0	0	0	0	0
_memset	0	0	0	0	0	0	0	0	0	0	0
_sbrk	0	0	0	0	0	0	0	0	0	0	0
_stlflush	3	95	23	42	122	23	69	1091	23	1038	
_stlflushall	1	147	147	147	200	200	200	200	200	200	200
_sim_trace	0	0	0	0	0	0	0	0	0	0	0
_sinit	1	6	6	6	6	6	6	6	6	6	6
_snakebuf	1	44	44	44	128	128	128	434	434	434	434
_start	0	0	0	0	0	0	0	0	0	0	0
_stat	0	0	0	0	0	0	0	0	0	0	0
_swhatbuf	1	45	45	45	45	45	45	270	270	270	270
_swrite	1	27	27	27	27	27	27	996	996	996	996
_swsetup	1	29	29	29	163	163	163	469	469	469	469
_cleanup	1	7	7	7	207	207	207	207	207	207	207
_exit	0	0	0	0	0	0	0	0	0	0	0
_close	0	0	0	0	0	0	0	0	0	0	0
_exit	1	20	20	20	227	227	227	246	246	246	246
_fmtformat	1	107	107	107	107	107	107	107	107	107	107
_free	0	0	0	0	0	0	0	0	0	0	0
_fstat	0	0	0	0	0	0	0	0	0	0	0
_hasnewline	1	13	13	13	46	46	46	46	46	46	46
_isatty	0	0	0	0	0	0	0	0	0	0	0
_main	1	10	10	10	484	484	484	1759	1759	1759	1759
_malloc	1	32	32	32	39	39	39	39	39	39	39
_malloc_privat...	0	0	0	0	0	0	0	0	0	0	0
_malloc_privat...	0	0	0	0	0	0	0	0	0	0	0
_mallocz	0	0	0	0	0	0	0	0	0	0	0

hello.stats x

Event Viewer (GUI)

The screenshot displays the Sandblaster IDE 1.0 (Build 040213) interface. The main window is titled "Events [producer-consumer.events]". The interface includes a menu bar (File, Edit, View, Build, Debug, Versioning, Tools, Window, Help), a toolbar, and a workspace divided into several panes.

Explorer [Filesystems]: Shows a tree view of the project structure, including folders like board, compilation, debugger, make, multithread, project, profquery, scratch, and support. Files under multithread include concurrent-output.c, concurrent-output.events, concurrent-output.o, concurrent-output.sbx, concurrent-output.stats, eventread.exe, producer-consumer.c, producer-consumer.events, producer-consumer.o, producer-consumer.sbx, and producer-consumer.stats.

OS Objects: Lists the following objects:

- Hardware Thread #00
- Hardware Thread #01
- Software Thread #00000001
- Software Thread #00010001
- Mutex @0x01200310

Timeline: A horizontal axis labeled "Cycle Count" ranges from 134845000 to 134852000. A vertical red line is positioned at cycle 13484643. Colored dots (blue, green, red) are plotted along horizontal lines for each OS object, representing events.

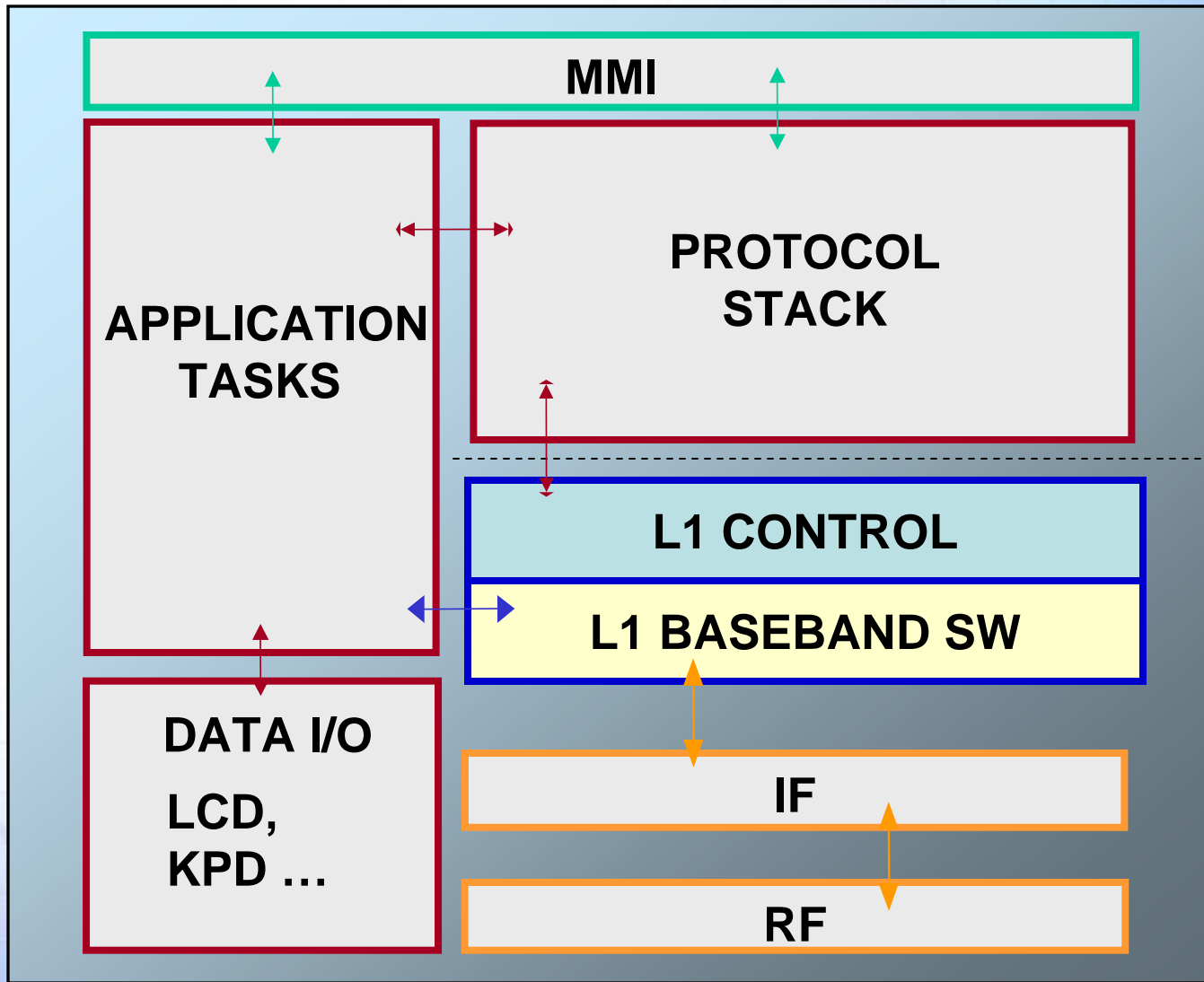
Show Events Dialog: A modal dialog box titled "Show Events" is open, displaying a list of events with navigation controls. The selected event is "Mutex @0x01200310".

Event	Thread
Mutex @0x01051818	Hardware Thread #00
Mutex @0x01051908	Hardware Thread #01
Mutex @0x01051984	Software Thread #00000001
Mutex @0x01053718	Software Thread #00010001
Mutex @0x010F3954	Mutex @0x01200310
Mutex @0x010F3EC8	
Mutex @0x010F3F08	

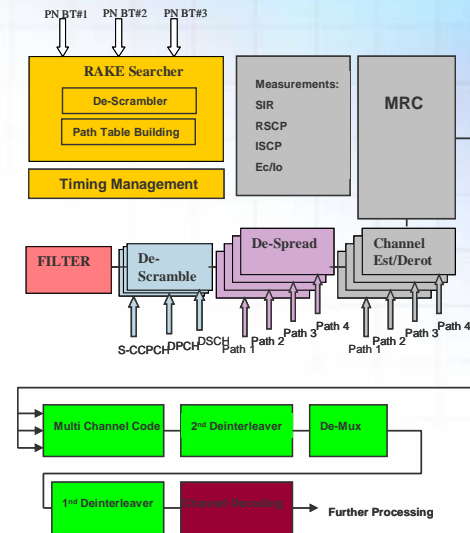
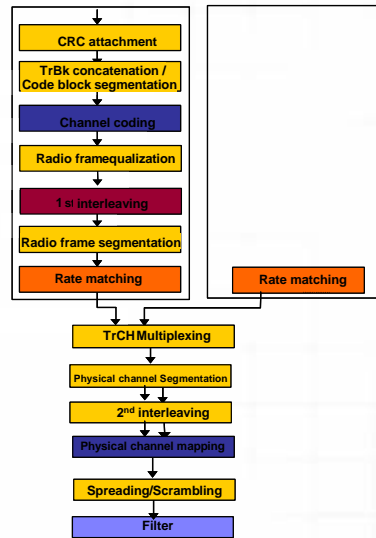


Communications Systems Implementation

Integration



Real-time WCDMA Performance



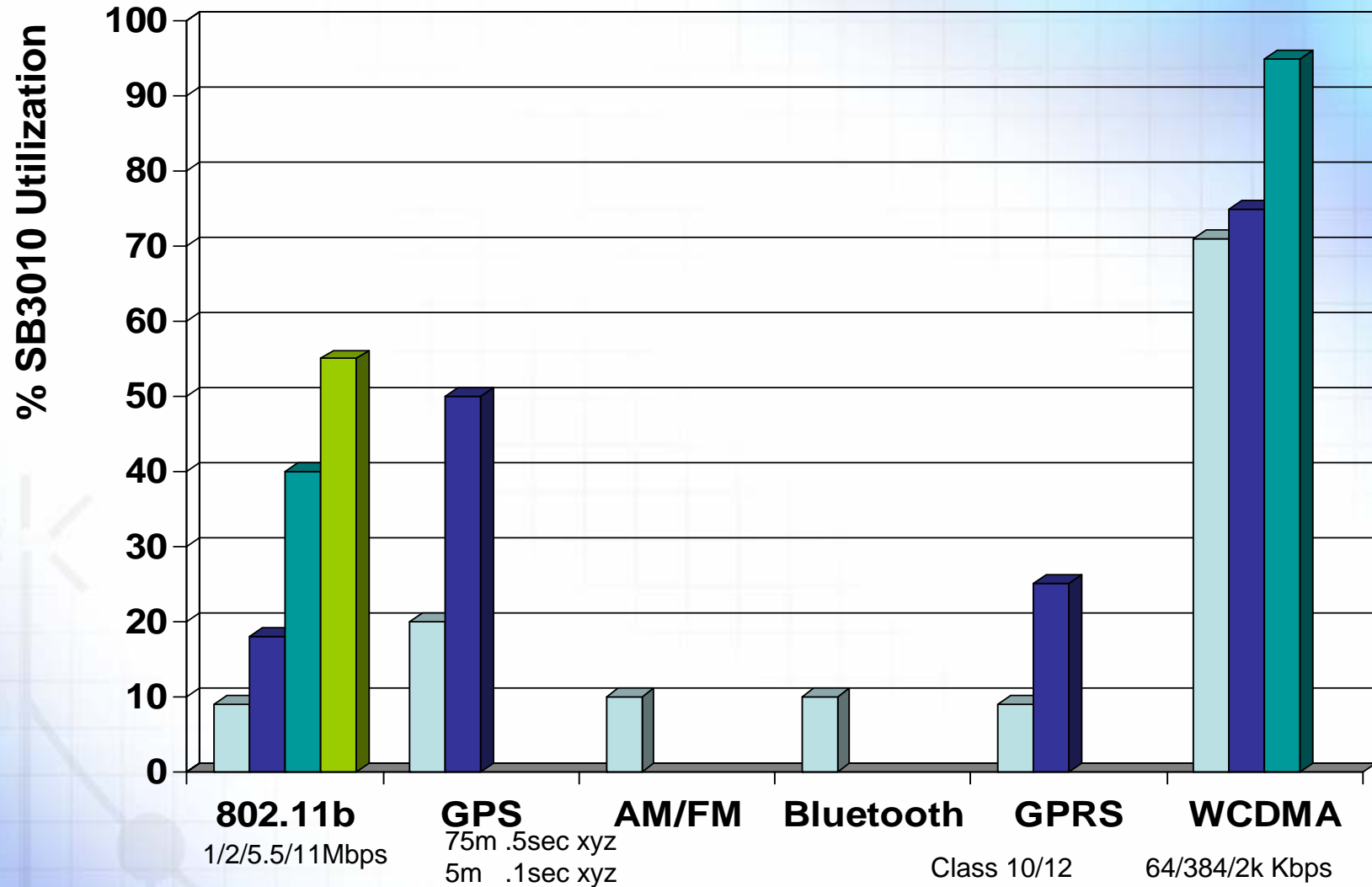
Real-time chip, bit, and symbol rate processing

- ➔ 1 SB9600 chip for 2Mbps Rx concurrently with 768kbps Tx
- ➔ <75% utilization for 384kbps Rx / 384kbps Tx

Includes functions traditionally implemented in H/W

- ➔ Turbo Decoder
- ➔ Rake Receiver
- ➔ Tx/Rx Filters

Communications Results



Summary

Multithreaded baseband processor

- ➔ High-performance and low-power
- ➔ DSP, Java, and Control processing

Sophisticated compiler technology

- ➔ Automatically generates DSP operations
- ➔ Automatically multithreads applications
- ➔ Hand coded performance

Reconfigurable Communications Protocols

- ➔ WCDMA, GSM, GPRS, etc.
- ➔ 802.11b, Bluetooth, etc.

Multimedia capability

- ➔ MP3
- ➔ MPEG4