C6000 Compile Tools / PBC Agenda

- CCS 1.2 Announcement
- C6000 Release 4.0
- Profile Based Compiler
- Roadmap

TI DSP Compile Tools Value Proposition

For the embedded software developer, TI’s DSP Compile Tools - co-developed with TI’s DSPs - offer the highest performance and code density in the industry due to architecture-specific optimizations as well as application-level analysis including interactive feedback, tuning, profiling, and system memory allocation.

TI Compile Tools Current Focus

- Architecture Co-development - Compiler and architecture work in unison
- High performance - alleviates the need to hand code assembly
- High code density - reduces system cost by minimizing memory requirements
- Architecture Specific Optimizations - Compiler possesses the knowledge of the expert hand coded assembly writer.
- Unique Interactive Tuning and Feedback
- Application-level optimizations - Utilizes knowledge of entire application to optimize key components
- Profile Based Compiler - Makes the right tradeoff along a two dimensional codesize vs performance graph
- Visual Linker - Eases System Memory Allocation
- Moving Forward → Unified Build Environment and Alchemy

Compiler Status/Roadmap - Platforms

- C6000
  - Industry’s Best Tuned and Out of the Box C performance
  - 4.0 Meets Internal Goals - 65% NatC, >80% OptC, >95% LinASM
  - Take C64x performance to C62x Levels
  - Continue to improve “out of the box” C performance
- C5000
  - Code Size better than Arm with Thumb mode
  - Mnemonic Assembler ensures compatibility
  - Need to add more functionality into Assembler
  - Initial Benchmarks in place end of March
  - Will use to drive 2.0 Goals

Industry leading real-time tools reduce cost, risk and development time

Enhancements

In Code Composer Studio 1.2

DSP/BIOS II

Flexibility, scalability and ease of implementation

New Compiler Tools

Visualize and optimize for maximum productivity

New Cores

All customers can start today!

Slash product development time over 50%

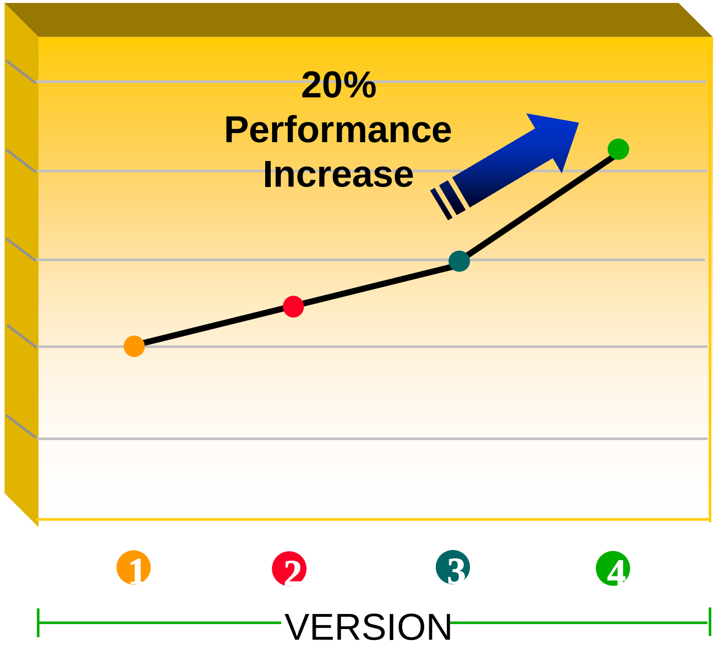
Development Time

New C6000™ Compile Tools

#1 DSP Compiler Extends Performance Lead

www.ti.com/sc/c6000compiler

Out-of-the-box Compiler Performance Improvement

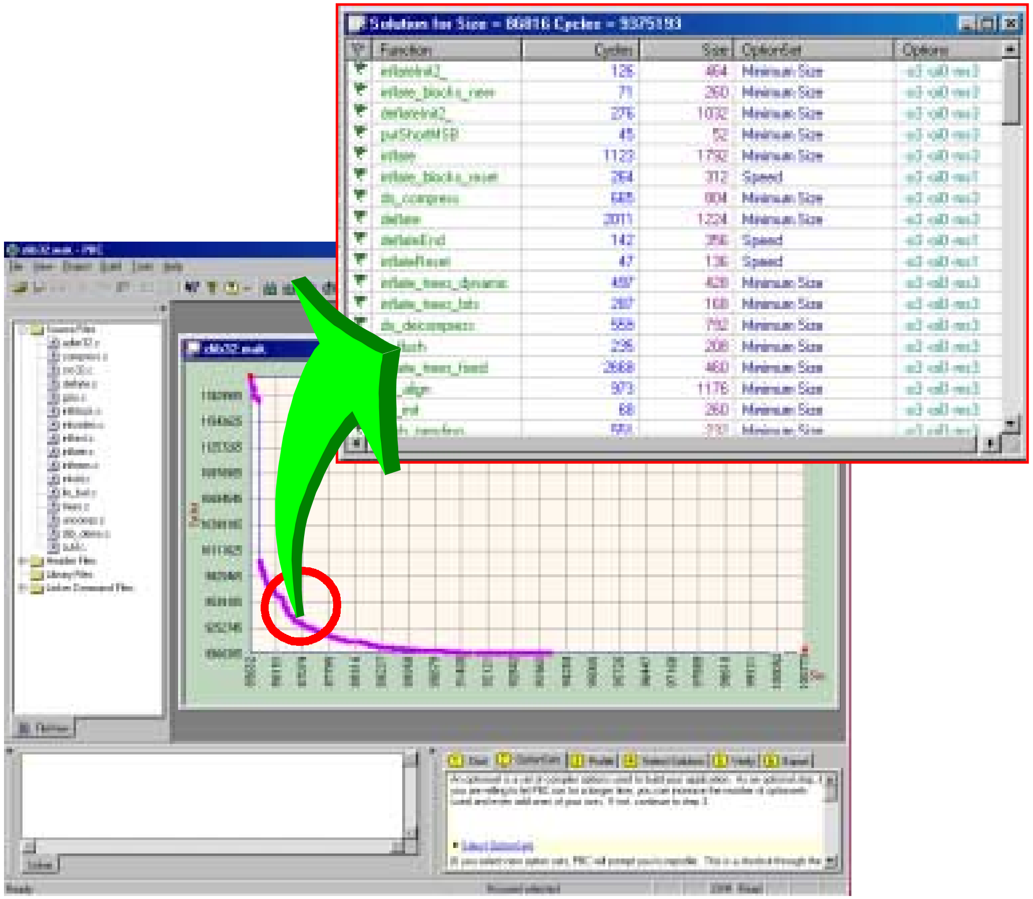


- Achieves 80-90% performance vs. hand coded assembly
- Performance statistics backed up with real code examples downloadable today
- Out-of-the-box C code focus has produced more that 20% performance improvement
- Unique compiler feedback
- Support for C+ +

Continuation of Speaker Notes

New C6000™ Compile Tools

Visualize and optimize code size and performance trade-offs



Solutions for Size = 86816 Cycles = 9375193

Function	Cycles	Size	OptionSet	Options
inflateInit2_	126	464	Minimum Size	-o3 -ol0 -ms3
inflate_blocks_new	71	260	Minimum Size	-o3 -ol0 -ms3
deflateInit2_	276	1032	Minimum Size	-o3 -ol0 -ms3
putShortMSB	45	52	Minimum Size	-o3 -ol0 -ms3
inflate	1123	1792	Minimum Size	-o3 -ol0 -ms3
inflate_blocks_reset	264	312	Speed	-o3 -ol0 -ms1
do_compress	685	804	Minimum Size	-o3 -ol0 -ms3
deflate	2011	1224	Minimum Size	-o3 -ol0 -ms3
deflateEnd	142	356	Speed	-o3 -ol0 -ms1
inflateReset	47	136	Speed	-o3 -ol0 -ms1
inflate_trees_dynamic	497	428	Minimum Size	-o3 -ol0 -ms3
inflate_trees_bits	287	168	Minimum Size	-o3 -ol0 -ms3
do_decompress	558	792	Minimum Size	-o3 -ol0 -ms3
flush	236	208	Minimum Size	-o3 -ol0 -ms3
ate_trees_fixed	2668	460	Minimum Size	-o3 -ol0 -ms3
_align	973	1176	Minimum Size	-o3 -ol0 -ms3
_init	68	260	Minimum Size	-o3 -ol0 -ms3
[illegible]	551	232	Minimum Size	[illegible]

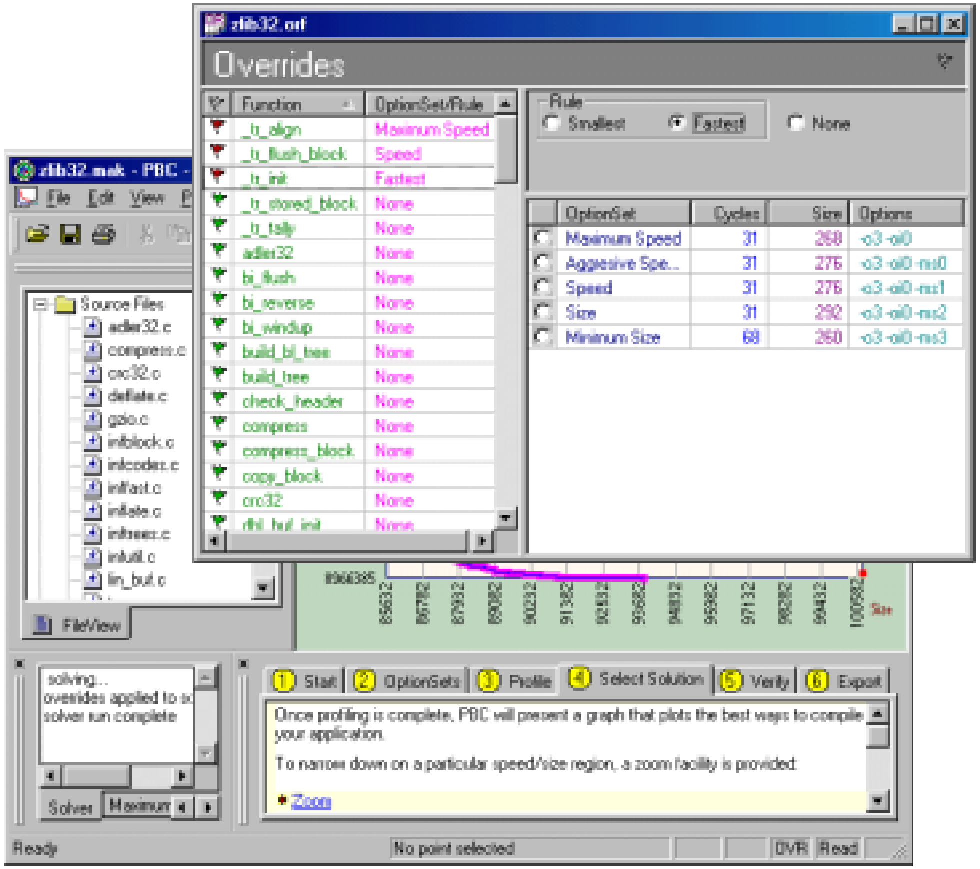
PROFILE-BASED COMPILER SOLUTIONS

- Build and profile multiple build option sets
- Automatically plot a 2D graph of code size vs performance
- Graphically select the optimum combination of size and speed for your application
- Click to build desired performance and code size trade-off in seconds

Continuation of Speaker Notes

Profile Based Compiler Details

Visualize and optimize code size and performance trade-offs



Function	OptionSet/Rule
_tr_align	Maximum Speed
_tr_flush_block	Speed
_tr_init	Fastest
_tr_stored_block	None
_tr_tally	None
adler32	None
bi_flush	None
bi_reverse	None
bi_windup	None
build_bl_tree	None
build_tree	None
check_header	None
compress	None
compress_block	None
copy_block	None
crc32	None
dhl_buf_init	None

Rule: ○ Smallest ● Fastest ○ None

	OptionSet	Cycles	Size	Options
○	Maximum Speed	31	268	-o3 -oi0
○	Aggesive Spe...	31	276	-o3 -oi0 -ms0
○	Speed	31	276	-o3 -oi0 -ms1
○	Size	31	292	-o3 -oi0 -ms2
○	Minimum Size	68	260	-o3 -oi0 -ms3

solving...
overides applied to s
solver run complete

Once profiling is complete, PBC will present a graph that plots the best ways to compile your application.

To narrow down on a particular speed/size region, a zoom facility is provided.

- Zoom

Ready | No point selected | OVR | Read

PROFILE-BASED COMPILER SOLUTIONS

- Express Assistant to Start
- On-line Tutorial
- Includes Ready to Run Demo
- File Overrides for ISR, etc.

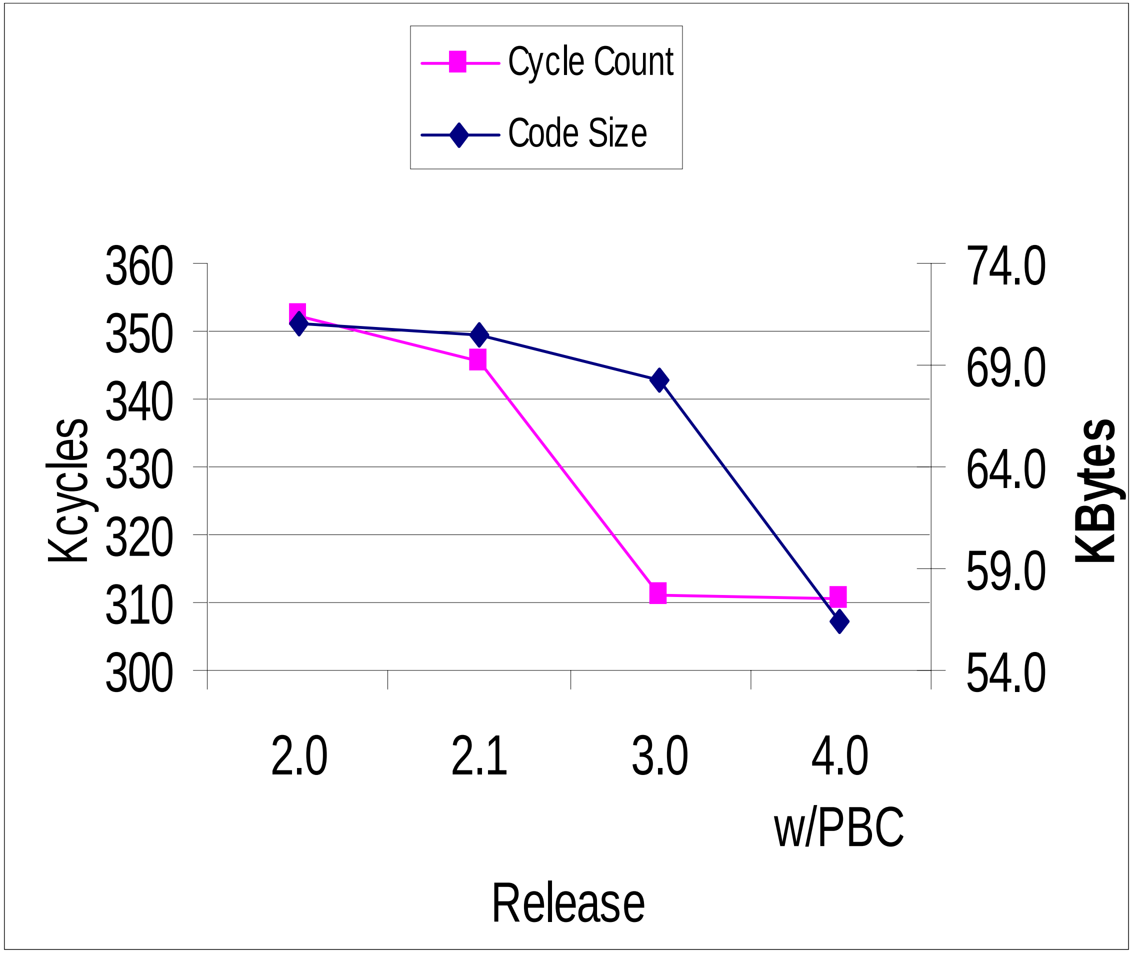
PBC Results on EFR GSM

- 288Kcycles at 60 Kbytes
- 311Kcycles at 56 Kbytes
- **<u>Fastest -</u>**

276Kcycles at 65 Kbytes

- **<u>Lowest Code Size -</u>**

45Kbytes at 1.25 Mcycles



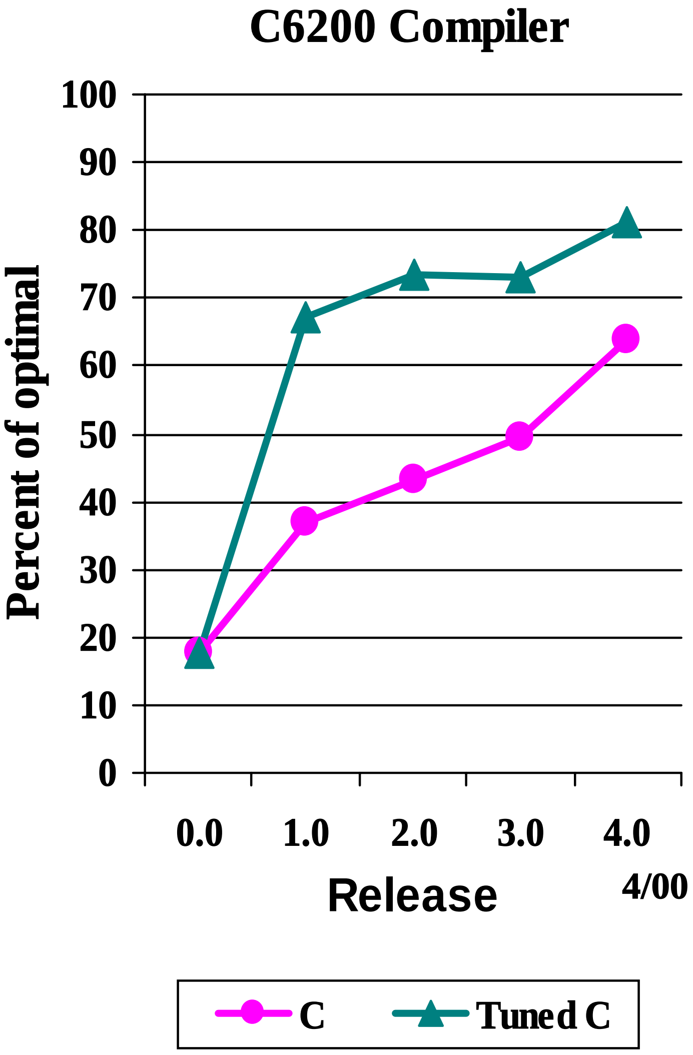
Performance Roadmap - Two Vectors

- Compiler gathers system/application-level information
  - Use profiling to get run-time behavior knowledge
  - Feed the compiler more system details (memory maps, libraries) to gain more contextual knowledge
  - Continue to develop optimizations to utilize these new sources of information
  - Continue to drive Architecture Specific Optimizations
- Interactive Visual tuning tools for the User
  - Identify performance critical code and provide suggestions for improvement
  - Graphical System Optimization
  - Automatically choose the best compiler optimization levels for an application based on user criteria

Driving Performance!

Benchmarking

- Methodology
  - Representative benchmarks created with both C and optimal hand coded assembly implementations
  - Each benchmark wrapped in a process that self checks correctness and reports timing
  - Performance of the compiler output compared to the optimal assembly
  - Process automated for nightly update
- Benefits
  - Benchmark analysis provides direction for compiler improvements
  - Measurable way to track compiler progress
  - Gives developers immediate feedback on impact of potential optimizations
  - Enables competitive benchmarking

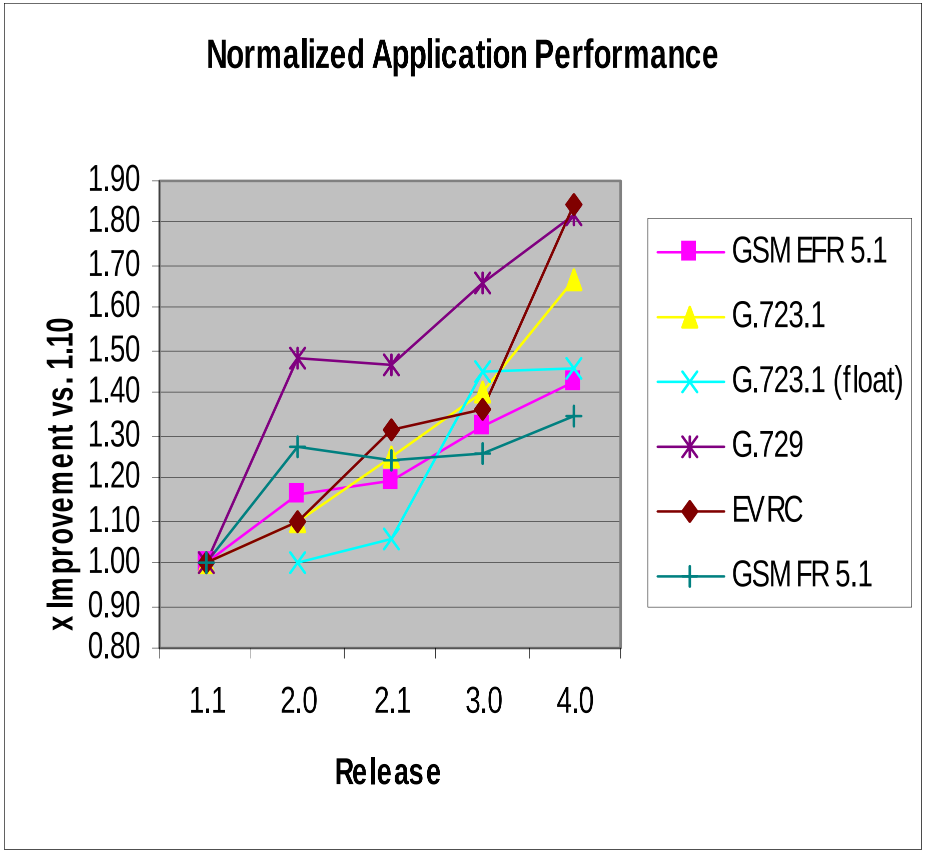


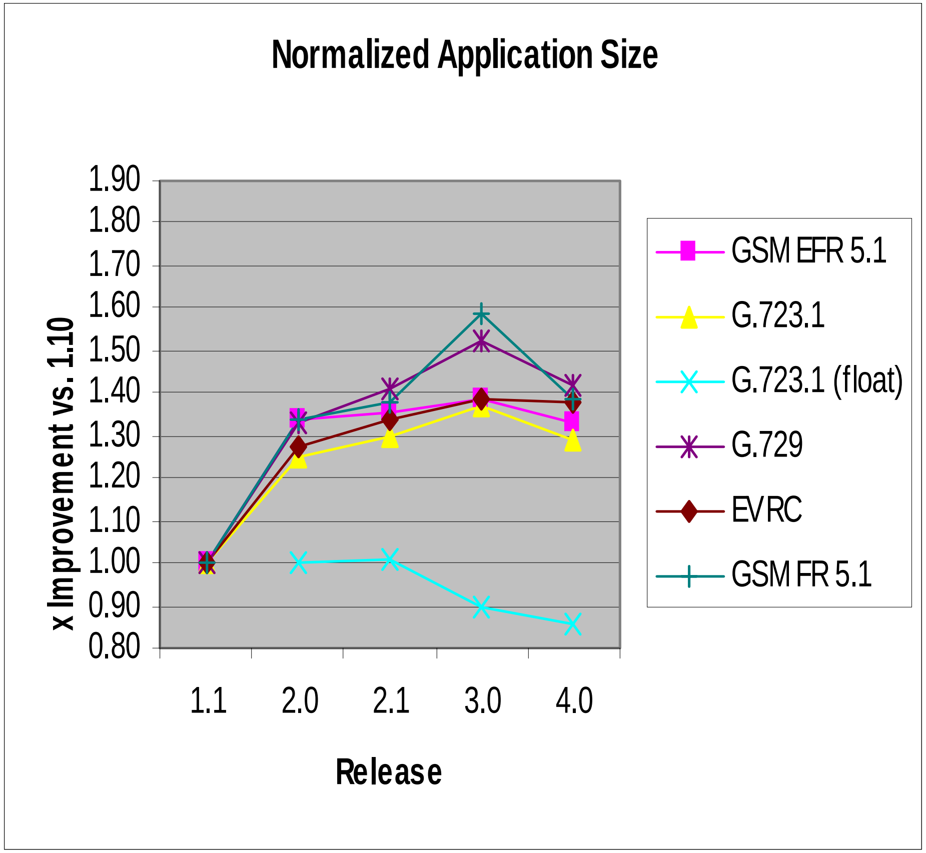
Full Algorithms

- Provides large pieces of DSP code to validate - improves compiler robustness
- Tracks out of the box algorithm performance
- Tracks code size vs performance
- Run on large data sets
- Run on small data sets with many option combinations
- Adding more control applications to grade code size

Algorithms

http://www.micro.ti.com/asp/sds/c6x/metrics/release_results.html





C6000 Benchmarks (on the TI Website)

Algorithm	*Used in*	*Assembly Cycles*	*Assembly Time (µs)*	*C Cycles* (Rel 4.0)	*C Time (µs)*	*% Efficiency vs Hand Coded*
Block Mean Square Error *MSE of a 20 column image matrix*	For motion compensation of image data	348	1.16	402	1.34	**87%**
Codebook Search	CELP based voice coders	977	3.26	961	3.20	**100+%**
Vector Max *40 element input vector*	Search Algorithms	61	0.20	59	0.20	**100+%**
All-zero FIR Filter *40 samples, 10 coefficients*	VSELP based voice coders	238	0.79	280	0.93	**85%**
Minimum Error Search *Table Size = 2304*	Search Algorithms	1185	3.95	1318	4.39	**90%**
IIR Filter *16 coefficients*	Filter	43	0.14	38	0.13	**100+%**
IIR – cascaded biquads *10 Cascaded biquads (Direct Form II)*	Filter	70	0.23	75	0.25	**93%**
MAC *Two 40 samples vector*	VSELP based voice coders	61	0.20	58	0.19	**100+%**
Vector Sum *Two 44 sample vectors*		51	0.17	47	0.16	**100+%**
MSE *MSE between two 256 element vectors*	Mean Square Error computation in Vector Quantizer	279	0.93	274	0.91	**100+%**

TI ‘C62x Compiler Performance Rel 4.0 : Execution Time in µs @ 300 MHz

Compiler Status/Roadmap

- C6000
  - Industry’s Best Tuned and Out of the Box C performance
  - 4.0 Met Internal Goals
    - 65% NatC, >80% OptC, >95% LinASM
  - Take C64x performance to C62x Levels
  - Continue to improve “out of the box” C performance