AVR® C Code Benchmarks

Enclosed is a 15 page presentation on C Code Benchmarks which depicts the ***AVR*** microcontroller against other popular 8-bit and 16-bit processors. The code is drawn from a varied selection of customer supplied applications, and was compiled using the native compiler for that processor.

Discussion of the Benchmarking Process

Making a C Code benchmark is *not* an objective issue and presenting these figures may result in comments which may have experience which differs from what we present here. The following general comments should be noted on C Code benchmarking:

- Which microcontroller is the most code-efficient is application dependent, and there is no single best microcontroller for all applications.
- Benchmarks, like statistics, can be constructed to show anything. Every microcontroller has its strengths, and by constructing applications which utilizes these specific strengths, any microcontroller can be claimed to be the most code-efficient.
- Our goal with this presentation is not to claim that we are the single most code-efficient microcontroller, but to support our claim that we have a very High-Level Language suitable architecture.

The ***AVR*** is not the most code-efficient of the group, but can be found among the top three in seven of the nine benchmarks and, as shown in the summary, it comes out favorably compared to the other controllers. No applications have been omitted from the benchmark due to bad performance of the AVR.

Comments on the Summaries

Accumulated

Since not all code could be compiled for all compilers, all the indexes are relative to ***AVR***. Only the applications that could be compiled for both applications are summed and the ratio between the code sizes is displayed.

Normalized

Since not all code could be compiled for all compilers, all the indexes are relative to ***AVR***. Only the applications that could be compiled for both applications are used and the ratio between every two applications is computed. The averaged ratio is displayed in the figure.

These results are coherent with the Accumulated results on the previous slide, but there is some variation in the factors due to different methods of computing the factor.

Comments on the Summaries

Most of the compilers used are from IAR Systems Ltd. The advantage of using the same compiler company for most of the processors is that the difference in the code size does not depend on the global optimization which all compilers will benefit from, but largely due to architectural differences.

IAR does not have any Compiler for ARM7/ARM Thumb.

Some of the compilers exist in later versions, so the results might be better for some of the microcontrollers.

The ***AVR*** C Compiler is, compared to many of our competitors, a relatively young compiler. The AVR C Compiler still gains significant code size decrease in every new release of the compiler.

C Code Benchmark Presentation

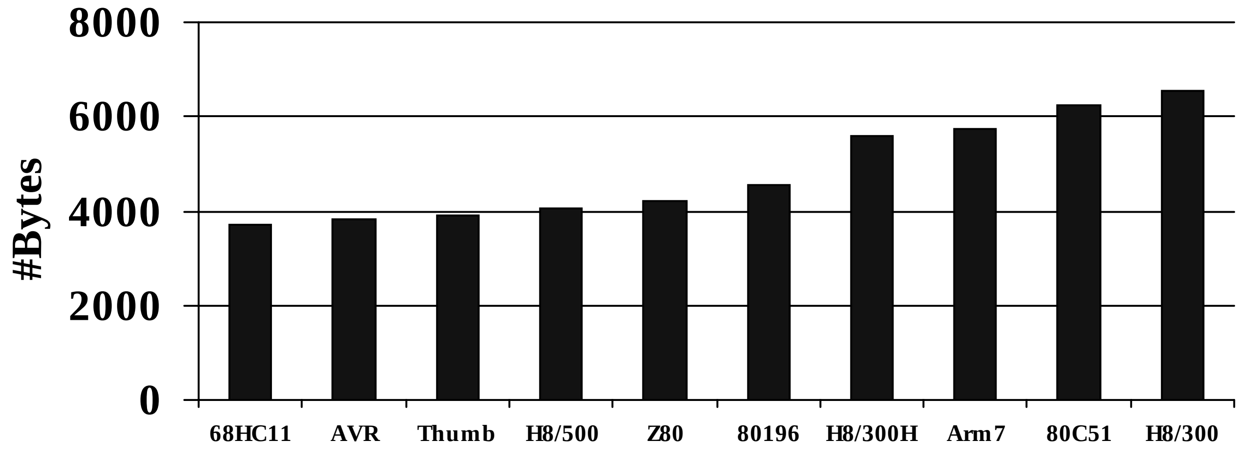
NOV98

C Code Benchmarks

- Nine applications, all based on actual customer code
- Varied application areas
- Byte usage in individual applications
- Summarized results reported as
  - Normalized Accumulated Results
  - Averaged Normalized Results

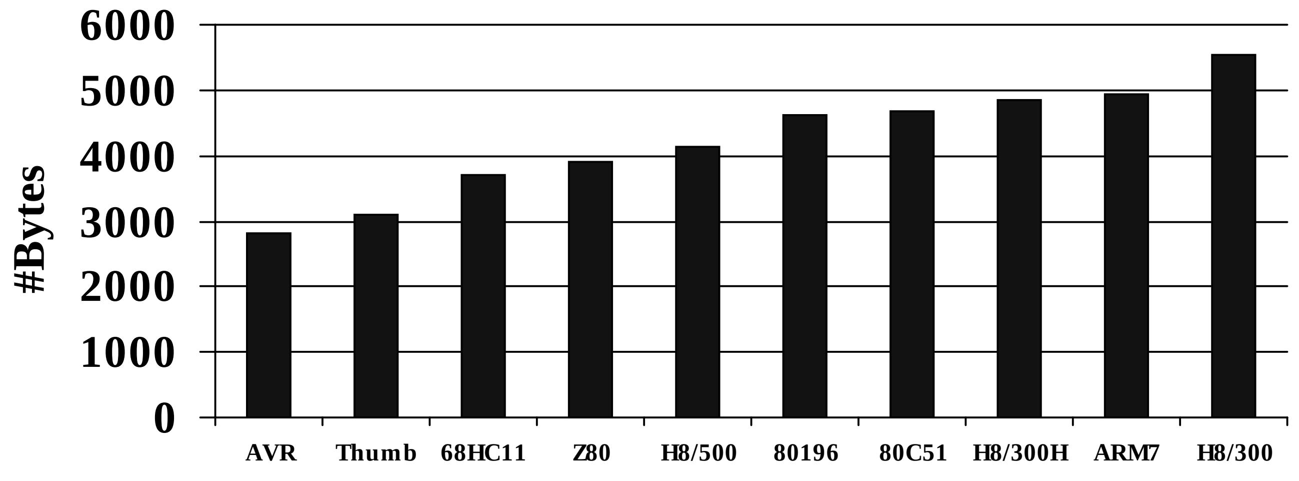
Pager protocol

- Three layer protocol
- Includes simple driver



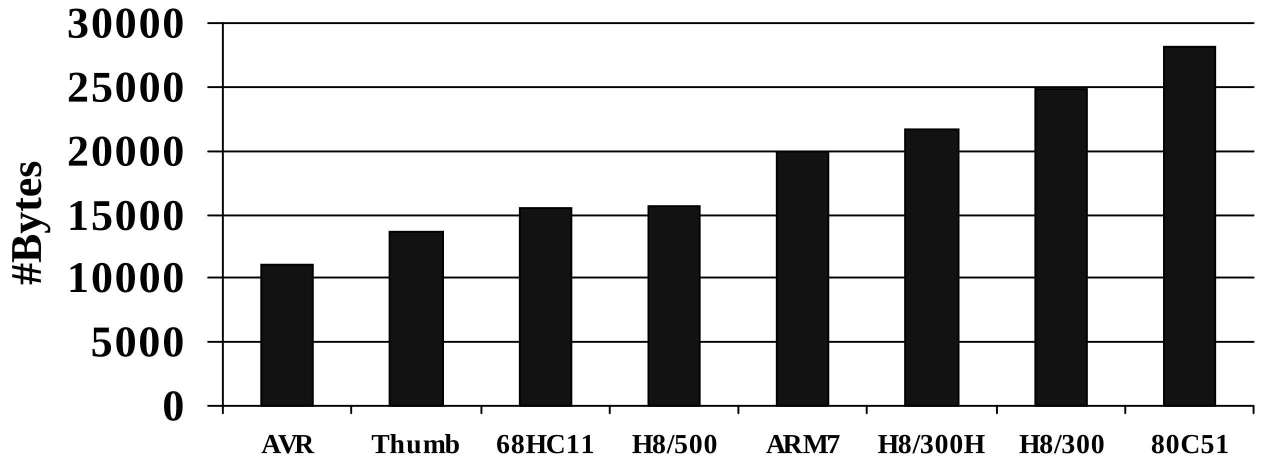
Analog Telephone I

- SIM Interface
- Parts of display driver



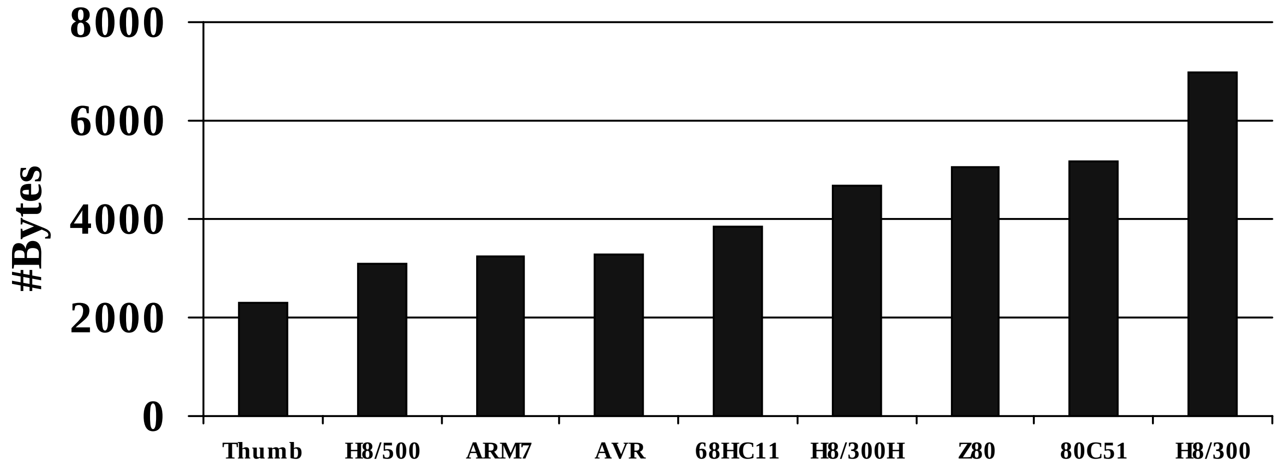
Analog Telephone II

- Automatically generated code
- State Machine based



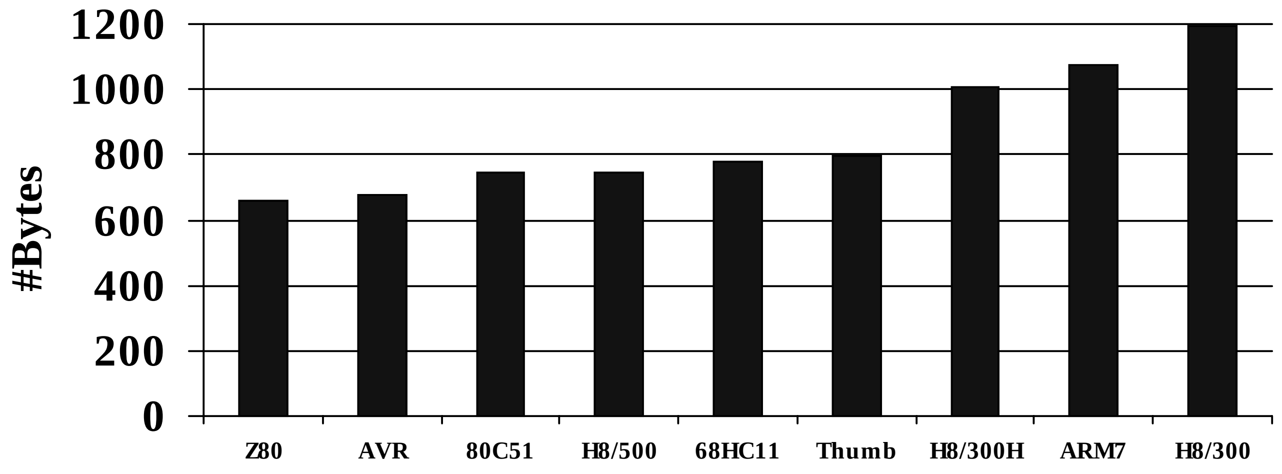
Reed-Solomon

- **Reed-Solomon Encoder/Decoder**



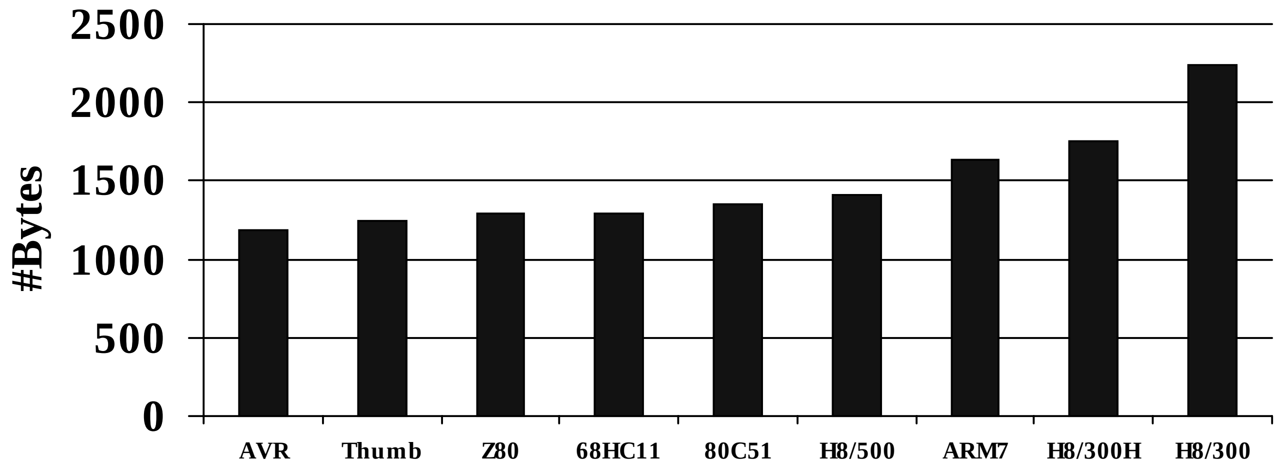
Car Radio Control

- Skeleton application
- Control Flow and Bitfields



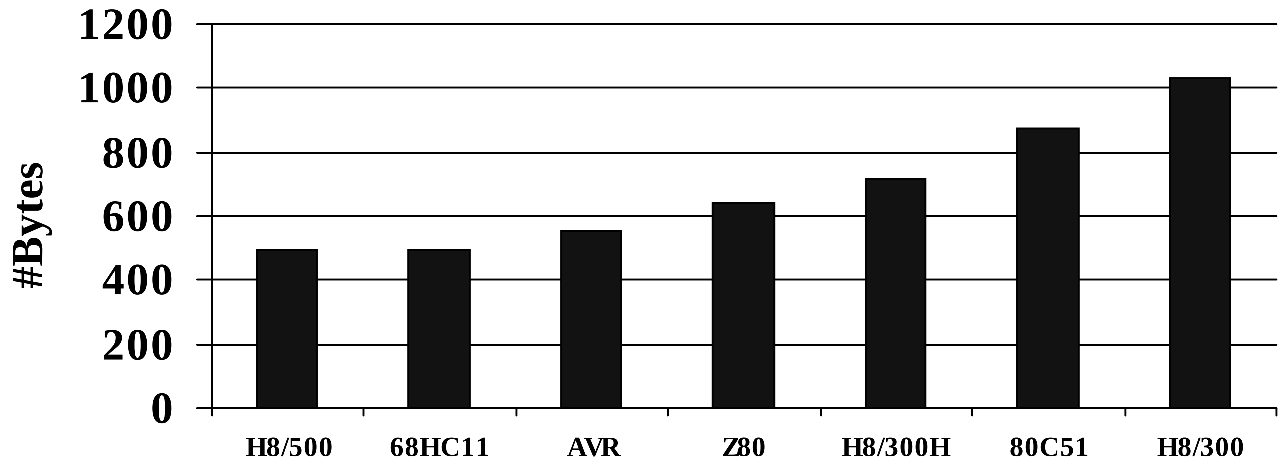
C Bitfields

- Benchmark code from customer
- 8-bit and 16-bit bitfield variables



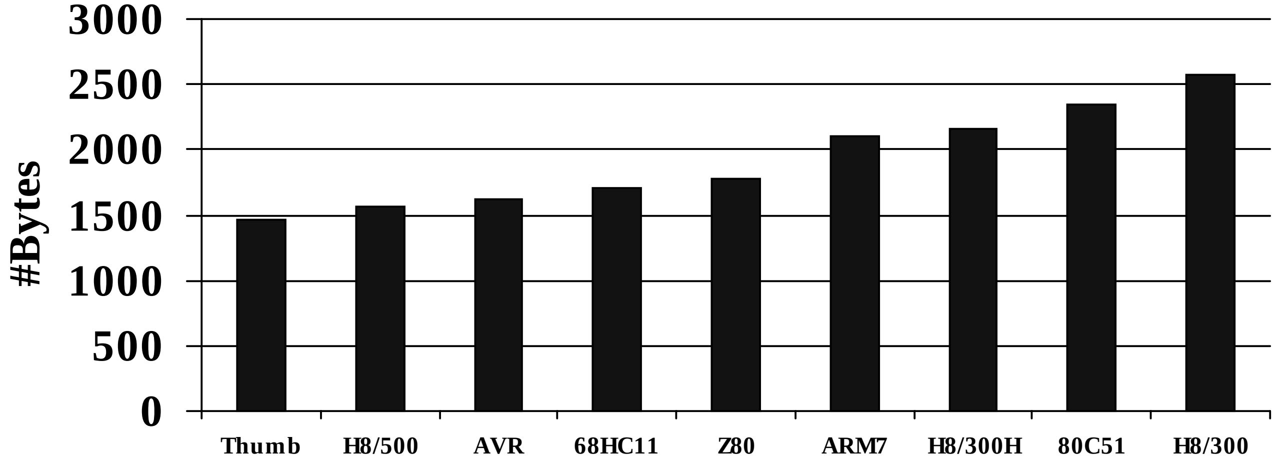
Analog Telephone III

- Representative collection of routines from an analog telephone application



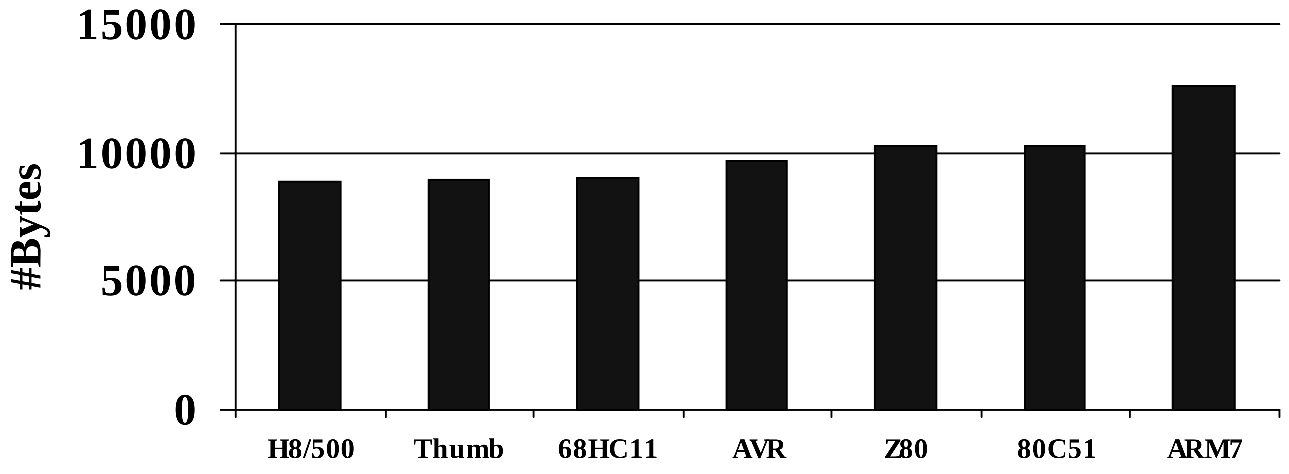
DES Algorithm

- Encryption/Decryption algorithm



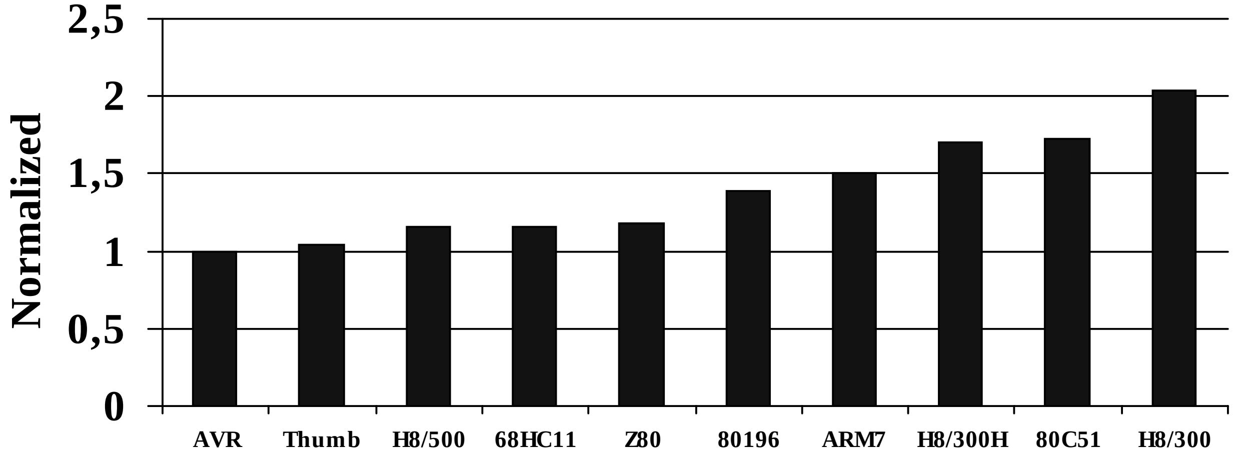
Navigation Application

- Complete application
- Communication, measurement and computations



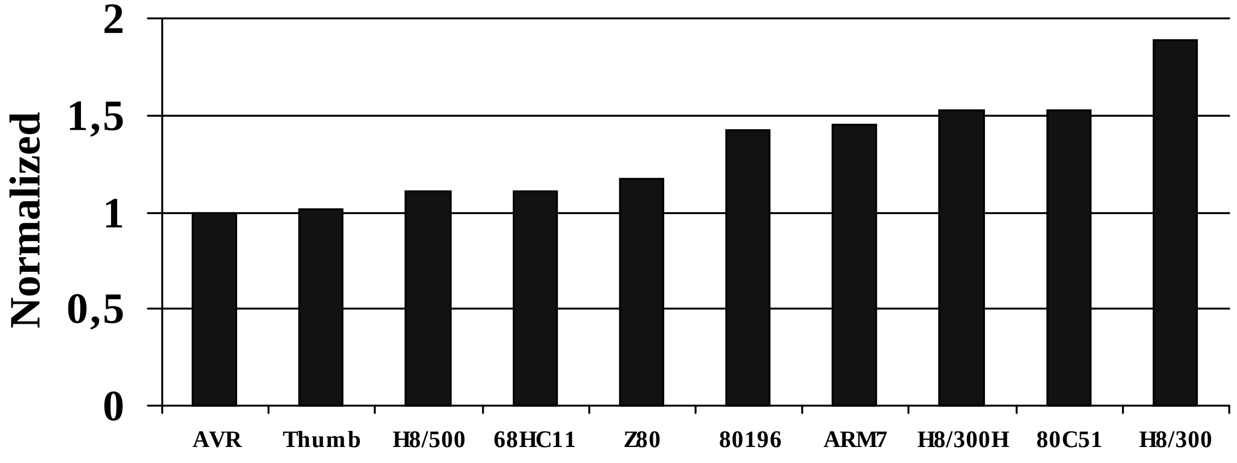
Accumulated Over All Benchmarks

- Indexes based on accumulated sizes
- Large applications count more than small applications



Normalized over all Benchmarks

- Averaged indexes from all applications
- All applications count evenly



In Summary

- **Nine C Code Benchmarks from various application areas**
- **No single microcontroller best for all applications**
- ***AVR* in the top range for all the applications**

C Compilers Used

- AVR: IAR ICCA90 version 1.40
- 80C51: IAR ICC8051 version 5.20
- Thumb: ARM tcc version 1.02b
- ARM: ARM armcc version 4.66b
- 80196: IAR icc196 version 5.20a
- Z80: IAR iccz80 version 4.03a
- H8/300(H): IAR icch83 version 3.22
- H8/500: IAR icch8500 version 2.92g
- 68HC11: IAR icc6811 version 4.20B