

The Challenges of Devising Next Generation Automotive Benchmarks

Patrick Leteinturier and Laurent Beaurenaut

Infineon Technologies AG

Copyright © 2008 SAE International

ABSTRACT

More than ever, microcontroller performance in cars has a direct impact on the driving experience, on compliance with improved safety, ever-stricter emissions regulations, and on fuel economy. The simple microcontrollers formerly used in automobiles are now being replaced by powerful number-crunchers with incredible levels of peripheral integration. As a result, performance can no longer be measured in MIPS (Millions of Instructions Per Second). A microcontroller's effectiveness is based on coherent partitioning between analog and digital, hardware and software, tools and methodology. To make an informed choice among the available devices, the designer needs benchmarks that are specific to automotive applications, and which provide a realistic representation of how the device will perform in the automotive environment. This presentation will provide an overview of the challenges of developing such benchmarks, the hardware required for their implementation, and the process by which a new suite of benchmarks that addresses these concerns is being implemented by the EEMBC (Embedded Microprocessor Benchmark Consortium). The goal of this suite is to take every factor into consideration to create an accurate measurement of a system's true performance. This will allow the development of a benchmark recognized by worldwide automotive vendors for its efficacy in measuring the performance of every component in automotive applications, including μ C cores, buses, peripherals, memory, compilers, profilers, operating systems, software drivers, and auto-code generators. This new benchmark is also targeted to be a tool to define performance requirements for the members of the automotive chain: OEMs, Tier1s, silicon suppliers, and tool vendors.

INTRODUCTION

Historically, innovations in automotive electronic systems have been primarily driven by emission regulations. Those regulations have become increasingly stringent over the years. The need to reduce fuel consumption and CO₂ emissions, combined with the ability to maintain a high level of drivability (in other words, the "fun to drive" feeling), create new

challenges for engine efficiency. Powertrain requirements coupled with the introduction of "by-wire" technologies push toward the development of new safe and dependable system concepts.

WHY VEHICLES REQUIRE BETTER PERFORMANCE

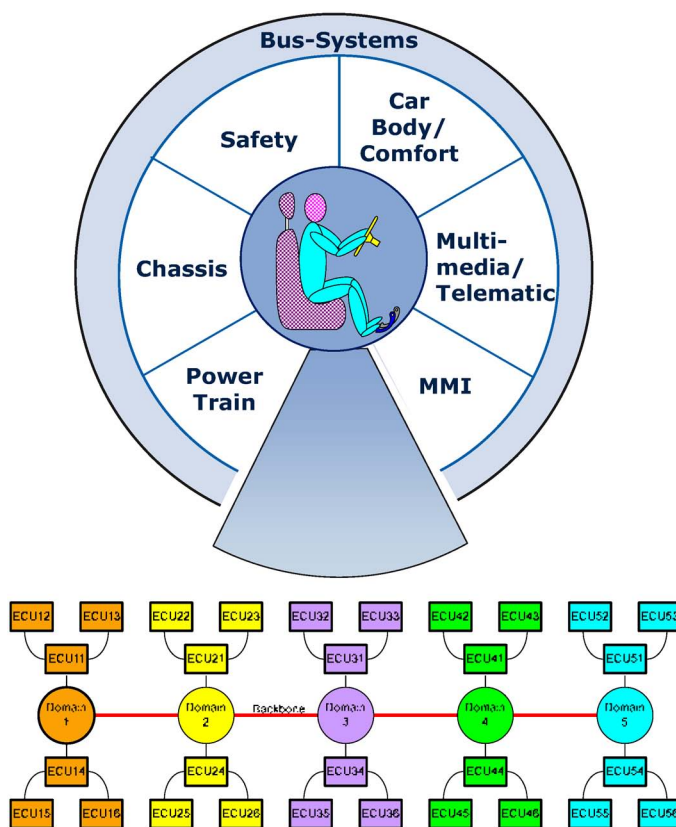


Figure 1: Domain-based Architecture

FUTURE SYSTEM EVOLUTION – New trends in automotive system partitioning are expected to generate radical changes in automotive architecture, at both hardware and software levels. The automobile's electronics have constantly grown in complexity over the years. Today's electronic architectural concept has reached its limits for further upgrades. In a premium car,

up to 80 ECUs are connected together via multiple buses and gateways. Attaching further buses would bring additional communication bandwidth at the cost of extremely great system complexity. Therefore, the trend is now evolving towards a domain-based structure (see Fig. 1). Each domain (Powertrain, Body, Safety, Infotainment, Chassis, Driver assistance) will be controlled by a main ECU, each of them being connected to the other via a high-speed backbone bus (FlexRay). This global centralization of functions (which does not exclude local decentralization of tasks by introducing smart sensors and smart actuators inside an ECU) allows a considerable increase in the amount of data transferred within a car, at a higher reliability level and with less overall system complexity than existing solutions. However, this leads to an increasing demand on the microcontroller's computational power.

Software development has also been highly impacted by these evolution trends. More complex control strategies are used, and more hardware functions migrate to software. As a result, the embedded code size has been increasing exponentially over time. Advanced tasks such as OBD and safety monitoring put an additional load on the main CPUs. For example, in future EURO 6-compliant engine control units, OBD and safety monitoring are expected to account for more than 40% of the total software code size.

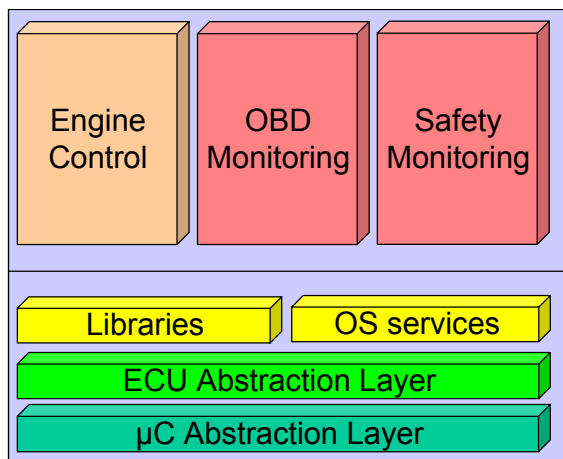


Figure 2: Software Partitioning

To cope with this growing software complexity, the automotive industry is moving towards new development methodologies. The broader use of autocode, software encapsulation and the introduction of Hardware Independent Software (e.g. AUTOSAR) has allowed greater reusability across several hardware platforms, along with an increase in robustness. The use of optimized hand-coded software is also declining, and in the future it will probably be limited to a few dedicated functions like DSP calculations and special libraries. This move towards “universal software” directly leads to higher system performance to compensate for the added overheads.

MICROCONTROLLER PERFORMANCE - In the past, the performance of microcontrollers in powertrain systems has increased exponentially over time at a rate of 2x every 2-3 years, following Moore's law. This rate is expected to slow down to 2x every 3-4 years due to technological constraints which will generate additional system cost.

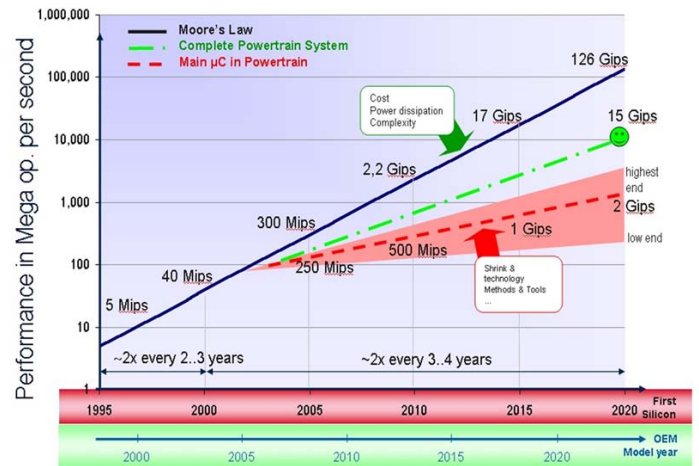


Figure 3: Projected Performance Requirements

Typically, the performance needs of automotive applications have mainly been met by increasing the clock frequency of the microcontrollers. In 2000, the typical clock frequency for a microcontroller in an engine control unit was 40MHz. Today's microcontrollers in powertrain applications can reach CPU frequencies up to 200 MHz. However, this value represents a limit that is difficult to cross with existing technologies. Increasing frequency requires higher supply current for a device. This generates more heat, which must be dissipated. Technical solutions exist at the packaging level to handle this situation, such as extra substrate layers, thick copper, heat slugs and heat spreaders, and enhanced BGA. However, the current cost of those innovations prohibits their implementation in the near future (see Fig. 4).

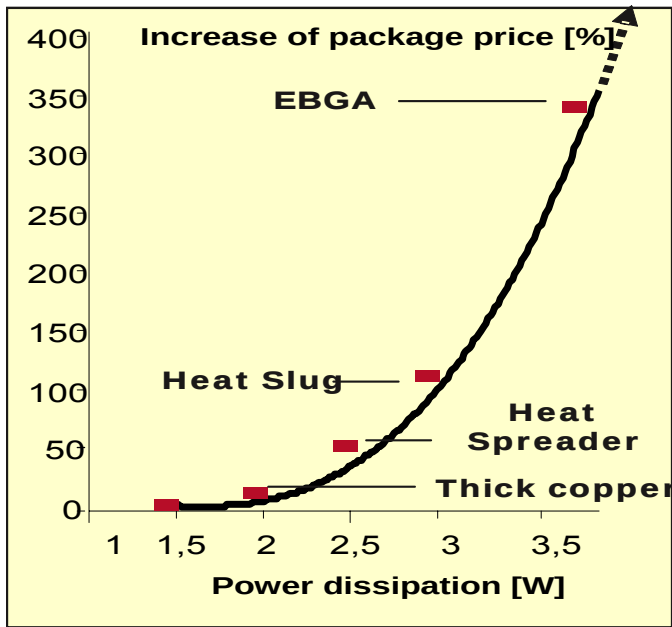


Figure 4: Package Cost vs. Power Dissipation

There are other factors that significantly degrade performance at higher frequencies. For example, the speed of embedded non-volatile memories cannot track the ever-increasing operating clock frequencies. Embedded Flash modules have a typical latency ranging from 10ns to 40ns, resulting in many CPU idle cycles. This type of hardware mismatch significantly limits the relative performance increase provided by the CPU's higher clock speeds.

In order to meet future performance requirements, the automotive microcontroller manufacturers are following an innovation path which is analogous to what the desktop industry experienced several years ago, and whose implementation has actually already started, at least for high-end systems. Those innovations include, for example, efficient data transfer with burst-based buses or crossbar switches, the implementation of fast cache and RAM memories, faster non-volatile memories, and generally everything that speeds up memory access.

A complementary approach to increase overall system performance is to increase the throughput of the microcontrollers' cores by e.g. multithreading, optimized instruction sets (e.g. DSP calculations), single-instruction-multiple-data (SIMD), or superscalar architectures, where several instructions can be executed in one cycle.

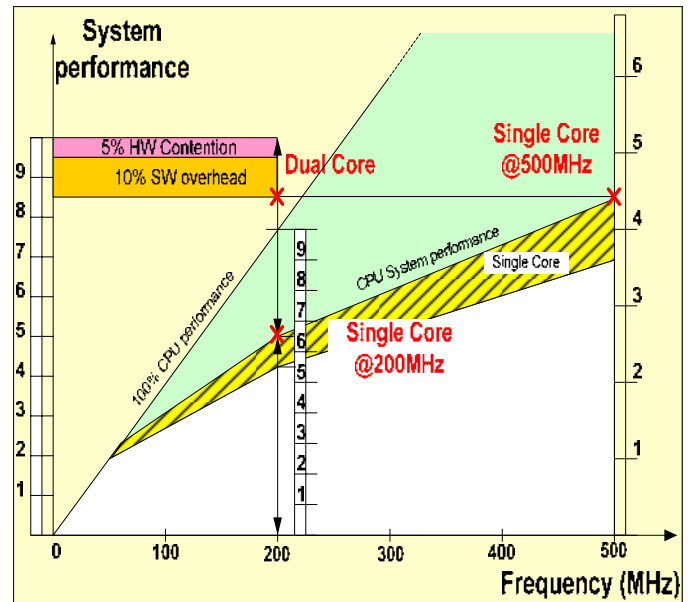


Figure 5: Package Cost vs. Power Dissipation

Another way to increase system performance is to implement two or more CPU cores in the same die. Several multicore architectural approaches can be followed at a hardware level: for example, a heterogeneous architecture (with a main CPU executing the application software and a smaller Peripheral Control Processor driving the peripherals), or a homogenous architecture (where two identical cores process different tasks in parallel). In the latter case, some performance loss will be experienced due to the synchronization overhead between the two cores, as well as access conflicts to the same resources (e.g. simultaneous access to the same peripheral or memory). However, this approach yields a significant increase in the effective performance of a microcontroller while keeping the clock frequency at a reasonable value (see Fig. 5).

On the software level, code efficiency can also be improved by means of tool optimizations. Compiler vendors have been continuously developing and implementing optimization strategies to improve code size and execution time. The code generated by a compiler must be tailored to a particular controller to make the best use of the available hardware features, and the optimization process should also take the entire code generation chain into account, from the auto-coder to the linker and locator.

STATE OF THE ART FOR BENCHMARKS

Choosing a microcontroller for a given application is a key decision that Tier1s (or OEMs) must take. They must ensure that the device meets the design requirements in terms of functionality and memory size, as well as in terms of cost. They also must ensure that the particular device provides enough computational power to run the application. The ideal way to ensure that the correct device is selected is to port the intended

application to the potential microcontrollers and see if it works. But this task can be extremely complex and time consuming, so the industry must rely on methods to assess the performance of the microcontroller's core as well as the advantages for the system provided by its peripheral sets, internal buses, and memory structure. Furthermore, in order to obtain the most complete assessment of the effective performance of the controller, the autocode generator, the compiler's chain, the operating system, the low-level software and the libraries must also all be included in this evaluation process.

Two popular benchmarks have emerged over the years to evaluate the performance of a microcontroller: Dhrystone 2.1 and the EEMBC AutoBench (Automotive Test Suite 1.1).

Dhrystone 2.1 was created in 1988. It is a small (only a few kBytes), C-code benchmark based on integer calculations. The benchmark results are given in Dhrystone MIPS (DMIPS), which is the (normalized) number of iterations per second executed by the target. The main advantage of Dhrystone is its portability: it can easily be run on almost every target. The major drawback of this benchmark is that it is not at all representative of real automotive embedded application software: for example, no floating point calculations or typical DSP algorithms are used. It is small enough to be completely cached, so that the memory system is not stressed, etc. Above all, since there is no independent organization for certifying the DMIPS performance of a controller, results can be artificially boosted using compiler tweaking or an older version of the benchmark. Therefore, the Dhrystone benchmark cannot be considered an accurate means for measuring system performance.

To some extent, EEMBC benchmarks overcome the Dhrystone's weaknesses and therefore represent a significant improvement in that area. The automotive test suite, AutoBench v1.1, is based on sixteen C-code benchmarks. The results of those 16 benchmarks are then combined by taking the geometric mean into an aggregate score called the AutoMark, which represents the overall performance of the microcontroller (the highest score for a powertrain microcontroller is AutoMark 100, achieved by Infineon's TC1796). Since the results must be certified by EEMBC, it ensures the credibility of the benchmark results. Nevertheless, even if the algorithms used by the benchmark suite are closer to the ones used in real applications, the AutoMark is not fully representative of system performance. For example, the code of the benchmarks can be put into fast RAM memories (which generally provides faster access times in comparison to Flash memories) or the peripheral set of the microcontroller is not taken into account.

While more complex hardware and software solutions are being implemented to address the performance challenge of future automotive systems, it appears that there is nothing available today to fairly and accurately assess the effectiveness of those solutions. Therefore, a new paradigm must be invented.

A NEW BENCHMARK DIRECTION

As already indicated, the applications that will run cars of 2020 are very different from those used in 1990. Not only is the code different, but the process to develop the code is also different. Future application development will utilize more abstract languages such as Matlab/Simulink or UML (Universal Modeling Language). The table below shows the evolution of the different programming languages used for an EMS (Engine Management System). In 1990 the code was entirely developed in assembly code on 8-bit controllers.

	1990	2000	2010	2020
Controller	8bit	32bit	Dual32	Quad32
Assembly Code	100%	30%	5%	0%
C-code	0%	70%	65%	10%
Automatically generated code	0%	0%	30%	90%

Table 1: Code generation for various µC

A benchmark is desired to assess performance, but it is getting more complex to assess the type of performance and to share the responsibility between the different parties. In 1990, the software was created and executed entirely by one person with very few tools. Today the software is developed by several people, several teams, several companies, with several tools and processes. At the end the only measure is system performance to cost. The battle between the different µC, SW module and tool suppliers will be terrible.