# A Single DSP Core Communication Engine Solution for Wireless Handsets and Convergence Devices

*Floyd Lazare, TTPCom and Stefano Angioni, StarCore LLC*

TTPCom Limited
Melbourne Science Park, Cambridge Road, Melbourn, Royston, Hertfordshire SG8 6HQ, UK
Tel: +44 1763 266266, Fax +44 1763 261216, Email: floyd.lazare@ttpcom.com,
URL: www.ttpcom.com

StarCore LLC,
ibc International Business Center, Ismaninger Str. 17-19, DE-81675 München, Germany
Tel. +49 89 413006 44, Fax +49 89 413006 64, Email: Stefano.angioni@starcore-dsp.com
URL: www. starcore-dsp.com

## ABSTRACT

The much-heralded concept of creating a single-core cellular modem has now become reality. TTPCom's latest version of their Cellular Baseband Engine (CBE 2000) combines both DSP and MCU functions on a single core, resulting in a greatly simplified programming model. This provides a more flexible way to partition tasks for easier maintenance and higher programming efficiency. In this paper, we present an innovation that demonstrates a new system architecture. StarCore's VLES (variable-length execution set) technology allows software developers to develop both signal processing and control code entirely in C and compile it into a seamlessly integrated application.

## 1. INTRODUCTION

TTPCom has recently demonstrated a fully functional E-GPRS cellular modem running entirely on a single-core modem based on a StarCore processor core. The variable-length execution set (VLES) architecture of the StarCore™ SC1000-family cores, coupled with an optimizing C compiler, makes it possible to integrate the complete modem on a single core. Integration of the complete modem on a single core reduces the overall cost of the system, simplifies handset design, completely frees up a second core for hosting applications, enables much faster integration of the sophisticated services featured in modern mobile devices, and removes bottlenecks by enabling the modem and the application environments to evolve separately.

## 2. CLASSIC DUAL-CORE MODEM SOFTWARE ARCHITECTURE

The first analog cellular modems were based on a single discrete MCU (microcontroller), based on a Complex Instruction Set Computer (CISC). When the cellular standards evolved from analog to digital, a DSP was added to the modem. At about the same time, there was a shift toward Reduced Instruction Set (RISC)-based MCUs. The dual-core architecture, based on an MCU and a DSP, evolved from several discrete parts to a single ASIC in the mid-nineties. The additional performance required to support packet data, such as GPRS and EGPRS, was achieved by increasing the cores' speed and by adding further hardware accelerators to the system. Typical software partitioning for a dual-core modem is shown in Figure 1. This has the applications software and Layers 2 and 3 of the protocol stack running on the MCU. The lowest layer is split into two components which we shall refer to as Layer 1 control and the physical layer software. For reasons explained below, these run on the MCU and DSP respectively.

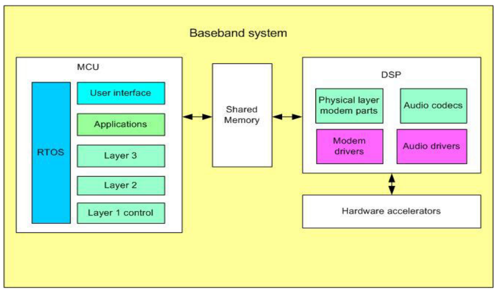

**Figure 1. Classic dual-core modem software architecture**

As described in the following subsections, each type of software has very different requirements.

### 2.1. Physical layer software

The physical layer software implements functions such as channel equalization, demodulation, channel coding/decoding, speech transcoding, echo and noise cancellation and audio equalization. These are archetypal DSP tasks, characterized by a high computational throughput, tight loops, little branching and a relatively simple structure. Such code has traditionally been run on DSPs that provide efficient fixed point arithmetic through the provision of hardware resources such as multiply accumulators and support for the addressing modes and looping that is currently employed. Often these DSPs had a non-orthogonal architecture that made it difficult to design efficient compilers and so much of the code needed to be written in assembler. It is also usual to reduce the computational burden placed on the DSP by providing function-specific hardware modules to perform some of the most intensive functions. These must be carefully selected so as to not unduly reduce the flexibility of the modem so that it can adapt to the ever-evolving communications standards. In recent years this has seen the move from GSM through GPRS to EDGE, raising the computational requirements of the physical layer software from a couple of tens of MIPS to in excess of 100 MIPS for a fully software implementation of an EDGE physical layer.

### 2.2. Layer 1 control and protocol stack software

The layer 1 control and protocol stack software is handled by the MCU and involves tasks related to the upper layers of the protocol stack and the user interface. The Protocol Stack and layer 1 control software is quite different from the physical layer software. In a typical implementation it is composed of a large number of inter-communicating operating system tasks. The software is much larger that the physical layer code and has a much more complex structure. These requirements make it essential for this code to be written in a high level language, usually C, and that the compiler is able to achieve a good code density on the target processor. Here too the MIPS requirement has also increased as the standards have evolved. However, even for EDGE the computational load is much lower than that required in the physical layer.

### 2.3. Applications software

The applications software has typically lived alongside the protocol stack software in the MCU but has evolved at a phenomenal rate. After the emergence of high-resolution LCD displays, a built-in camera, and convergence capabilities, there was an explosion of new handset functionalities, such as 2D and 3D graphics, mobile multimedia, and Internet connectivity. Although typical feature-phone applications can live alongside the protocol stack software, the growing MIPS requirement and the need to evolve the applications more rapidly than the modem, which itself is subject to timely and costly certification processes, means that separating the two is a growing trend. Currently, the life cycle of applications is around 6 months, while that of the modem is typically 12 to 18 months. Extrapolating the classic dual-core approach, you could add a third core for applications only, as shown in Figure 2.

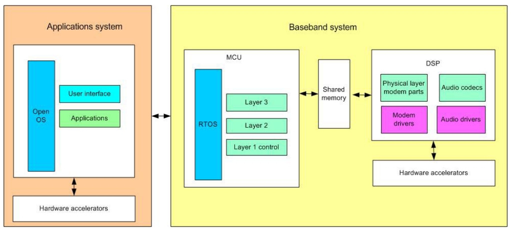

**Figure 2. Dual-core modem with an applications core**

However this adds to the complexity of the inter-processor communications and memory provisioning. The solution is costly in terms of silicon area and power consumption.

### 2.4. Limits of the dual-core modem approach

With the rapid growth in applications and the increased functionality required in the modem it is desirable to be able to develop these independently. One way of achieving this is to add a specialist applications core, resulting in a three core solution. However, this results in the need for more complex inter-core communications and memory sub-systems. An alternative approach is to make use of recent enhancements in processor technology to run both the physical layer processing and the protocol software on the same core, leaving the second core free to run applications.

## 3. SINGLE-CORE MODEM SOFTWARE ARCHITECTURE

If it were possible to move the complete protocol stack (from L1 to L3) on the single-core modem, then the MCU could be completely available for applications. In Figure 3, the single-core modem handles all modem processing, including call signaling and audio codecs.
Because the protocol stack requires a relatively small amount of processing power compared with the physical layer software, the average processing power required by the DSP to handle the modem tasks doesn't increase significantly. However, the real-time system performance (peak MIPS) must handle additional tasks without sacrificing the time criticality of the protocol stack. By converging all of the modem functionality (the protocol and physical layer software) onto a single core, the applications are no longer limited by the requirements of the modem. We still have only two cores in the chip, but now we can have two separate but parallel roadmaps. This is essential when we consider the life span of handsets today and the applications and communications technology roadmaps that we expect to see over the next 12 to 18 months:

- Applications: digital rights management (DRM), instant messaging, push-to-talk, A-GPS, streaming video, accelerated 3D Java gaming
- Communications: EDGE, SAIC, HSDPA, Dual Transfer Mode, TD-SCDMA

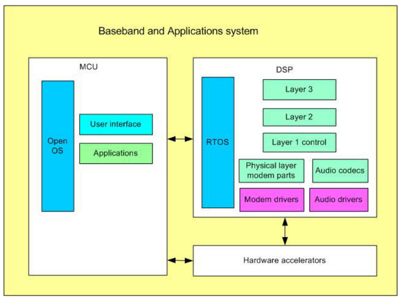

**Figure 3. Single-core modem with an applications core**

This parallel and asynchronous roadmap is already a real problem for both handset and silicon developers and is set to become more significant as handset functionality evolves, especially with the importance of open operating systems such as WinCE and Symbian. The separation of the modem and application environments simplifies handset design and enables much faster integration of the sophisticated services that are demanded by modern mobile users. Furthermore, having two rather than three cores can lead to lower costs and power consumption. The move toward open operating systems such as Windows CE and Symbian is assisted by the separation of applications because it reduces the conflicting real-time requirements of the operating system and modem. To achieve these objectives the DSP must be able to run both signal processing code and control code. The control code must be compiled and the resulting code needs to have a code density that is comparable to that achieved on specialist MCUs. It must also perform signal processing with a similar efficiency to specialist DSPs. If it is possible to achieve this with the DSP code written in a high level language, then this is a further advantage. These requirements are satisfied by StarCore's VLES processors.

## 4. STARCORE'S SC1200 CORE

Historically, DSPs and MCUs have been designed for and targeted at very different application areas. The current generation of DSP cores is based on the very-long instruction word (VLIW) architecture, which takes advantage of the characteristics of signal-processing architectures: the predictability of the instruction sequence and the ability to process data in parallel. Cores that use VLIW architecture depend on simple instructions that encode a single operation. Their advantage comes from issuing and executing instructions in parallel groups rather than one at a time. VLIW DSP cores typically use 32-bit wide instruction words—double the size of those used by conventional DSPs. This allows designers to use larger register sets to enhance performance. The wider word is necessary, however, because information about which unit will execute the instructions must be included in the instruction word. A downside of the long instruction word, however, is high program-memory usage, which translates into additional cost for RAM or ROM. Power consumption is also high. The RISC core's primary role as controller makes its desired capabilities a bit more generic. This reduces its performance in signal-processing tasks. However, as the architectures have evolved and the need for convergence between the two has become greater in many different product areas, some more powerful "hybrid" cores have emerged that combine high-performance signal processing with the rich addressing modes, good interrupt performance, ease of programming, and compact code required by timing-critical control code and protocol software. The variable-length execution set (VLES) architecture of the SC1000-family cores, coupled with an optimizing C compiler, allows both physical layer software and protocol stack software to run efficiently with excellent code density, making it possible to replace the MCU and the DSP cores with a single StarCore core. The resulting lower royalties and smaller silicon footprint reduce the overall cost of the system. The StarCore SC1200 is a licensable core with a scalable, highly flexible architecture that yields an excellent power-to-performance ratio.

## 4.1. Scalable performance

The SC1000-family instruction set is independent of the number of execution units. Designers can select the member of the SC1000-family of software-compatible cores that best meets the applications' requirements. The SC1200 core, described in Figure 4, consists of the following:

- Data Arithmetic Logic Unit (DALU), which contains two Arithmetic Logic Units (ALUs) and a DALU register file.
- Address Generation Unit (AGU), which contains two Address Arithmetic Units (AAUs), a Bit Mask Unit (BMU), and an address generator register file.
- Program Sequencer and Control unit (PSEQ).

Designers can implement the core in a wide variety of power-consumption, frequency, and die-size combinations. The SC1200 core can reach a high frequency of operation at low voltage and provides 2 million Multiply and Accumulate (MAC) operations per second (2 MMAC) for each megahertz of clock frequency.

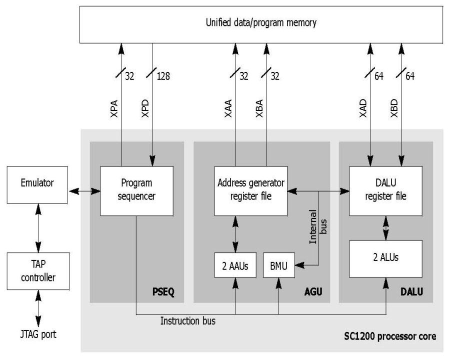

**Figure 4. Block diagram of the SC1200 core**

## 4.2. Variable Length Execution Set (VLES)

The VLES software model used by the SC1200 addresses the requirements of high performance in DSP kernels and compact code density in system applications. All SC1200 instruction words are 16 bits wide. Most instructions are encoded with one word. Each SC1200 instruction encodes an atomic (lower-level) operation. Since fewer bits are needed to encode atomic operations, the 16-bit instruction set becomes fully orthogonal and very rich in the functionality that it supports. In order to execute signal-processing kernels, a set of SC1200 instructions can be statically grouped by the C compiler (or assembly level programmer) to be executed in parallel. The SC1200 executes a VLES with up to two DALU instructions and two AGU instructions at the same time. The result is a powerful instruction set that yields an impressive code size compared with other VLIW software methods.

## 4.3. Instruction pipeline stages

The SC1200 core has a five-stage pipeline, which keeps latency low when branching and allows control code to run very efficiently. Figure 5 illustrates the five pipeline stages.

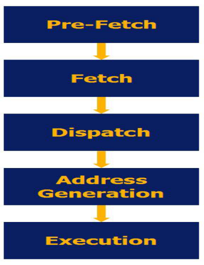

**Figure 5. Instruction pipeline stages**

Table 1 provides an overview of the operations performed at each stage of the pipeline.

| Pipeline Stage | Description |
|---|---|
| Pre-fetch | • Generate addresses for program fetch<br>• Update fetch counter (FC) |
| Fetch | • Read fetch set from memory |
| Dispatch | • Dispatch instructions<br>• Decode AGU instructions |
| Address Generation | • Decode DALU instructions<br>• Generate addresses for data load and store operations<br>• Perform address calculations: normal and change-of-flow<br>• Perform AGU arithmetic instructions<br>• Update AGU registers |
| Execution | • Read source operands to DALU<br>• Read source register for memory store operations<br>• Perform data calculations (multiply and add)<br>• Write DALU results to destination registers<br>• Write destination register for memory load operations |

**Table 1. Pipeline stages overview**

## 4.4. High code density for minimized cost

The VLES software model provides the high code density of 16-bit instruction set architecture without sacrificing the high-performance features found in a 32-bit instruction set architecture. The VLES for DSP kernel operations provides the flexibility to include powerful prefixes for more complex instructions, when required. The SC1200 core has a rich and orthogonal instruction set, a major portion of which focus on control code that often occupies most of the application code. DSP kernels and applications can be developed in C programming language. An optimizing compiler generates parallel instructions while maintaining a high level of code density. An orthogonal instruction set and programming model, along with single data space and byte addressability, enable the compiler to generate efficient code.Hardware-supported integer and fractional types enable application developers to choose their own style of code development or to use coding techniques derived from an application-specific standard. Table 2 shows the maximum MCPS and total memory requirements for some common 3GPP speech codecs on an SC1200 core.

| Codec | Max MCPS | Program memory (KBytes) | ROM data (KBytes) | RAM data (KBytes) | Total memory (KBytes) |
|---|---|---|---|---|---|
| FR Encoder & Decoder | 1.63 | 10.2 | 0.6 | 1.6 | 12.4 |
| Hr Encoder & Decoder | 9.9 | 40 | 16 | 2 | 58 |
| EFR Encoder & Decoder | 10 | 40 | 26 | 5.5 | 71.5 |
| AMR with Built-in EFR Encoder & Decoder | 10 | 52 | 30 | 5.5 | 87.5 |

**Table 2. Maximum MCPS and memory requirements for some common 3GPP speech codecs on an SC1200 core**

### 4.5. Support for multitasking operations

The hardware optimizes stack support by providing both a normal stack and an exception stack. The wide data buses can be utilized to provide optimized context switch support. Two push and two pop instructions can be grouped in one cycle.

### 4.6. Typical system-on-chip configurations

The SP1203 is a full-featured, ready-to-market platform that reduces risk, cost, and time-to-market. This platform boasts a flexible memory subsystem, complete with program- and data-cache controllers that increase performance when accessing external cached address space, but also provide a pass-through feature for direct access to non-cached address space. A mapped accelerator interface unit allows easy attachment of accelerators and other custom memory-mapped hardware that requires high-speed communication with the core.

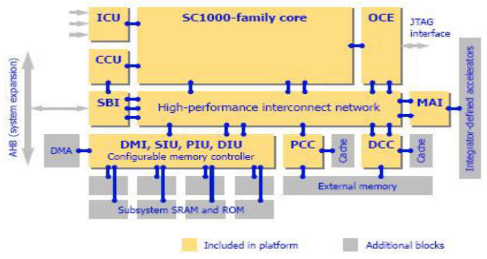

**Figure 6. Block diagram of SC1200 platform**

## 5. CELLULAR BASEBAND ENGINE (CBE)

In 2004, TTPCom showcased a fully functional cellular modem running entirely on a single core based on a StarCore processor core (shown in Figure 7) at the 3GSM World Congress in Cannes, France, CTIA Wireless in Atlanta, Georgia, and Electronica USA in San Francisco, California. TTPCom has worked closely with StarCore LLC during this project to implement its CBE.

TTPCom's CBE is a scalable and portable cellular handset modem design. The CBE core platform allows silicon vendors to create a complete baseband chip. The platform is supplied as an intellectual property (IP) package that provides all the necessary sub-systems needed for a multimode cellular handset baseband modem. TTPCom's latest CBE 2000 IP allows any combination of GSM, GPRS, E-GPRS, and W-CDMA functionality to be supported by the single-core modem, depending on a silicon vendor's specific market requirements.

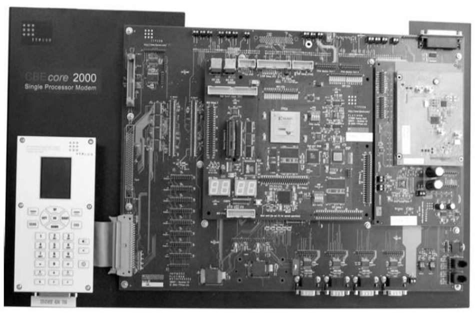

**Figure 7. Single-core cellular modem reference platform developed by TTPCom on StarCore technology**

Figure 8 is a software view of the single-core modem CBE 2000 system architecture. This diagram also illustrates TTPCom's

- Framework, which is a flexible architecture for partitioning of control and signal processing tasks, and
- Mapal, which is a 'software bus' for integrating new applications and related codecs.

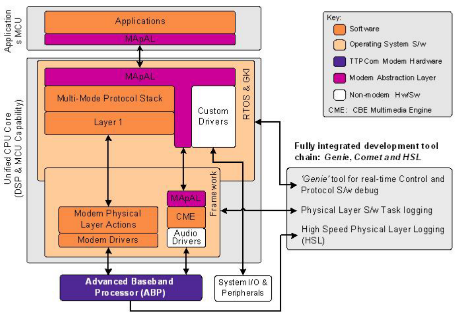
Applications MCU
Applications
MApAL
Unified CPU Core (DSP & MCU Capability)
MApAL
Multi-Mode Protocol Stack
Layer 1
Custom Drivers
RTOS & GKI
Framework
Modem Physical Layer Actions
Modem Drivers
MApAL
CME
Audio Drivers
Advanced Baseband Processor (ABP)
System I/O & Peripherals
Key:
Software
Operating System S/w
TTPCom Modem Hardware
Modem Abstraction Layer
Non-modem Hw/Sw
CME: CBE Multimedia Engine
Fully integrated development tool chain: Genie, Comet and HSL
'Genie' tool for real-time Control and Protocol S/w debug
Physical Layer S/w Task logging
High Speed Physical Layer Logging (HSL)

**Figure 8. Software view of the single-core modem CBE 2000 system architecture**

## 6. SUMMARY

The evolution of baseband chips toward a single-core modem architecture can be viewed as "freeing up" the MCU for applications in order to meet the rapidly evolving needs of the handset market. Over time, the separation of the modem and applications onto two distinct cores will provide a great deal of scope for developers to optimize the modem core and its subsystem separately from the application core and its subsystem. Eventually, the modem's cost and power consumption can be reduced to the lowest levels possible, creating a true commodity component that can be deployed in a range of application-differentiated products.

## 7. REFERENCES

[1] S. Rader, J. Corleto-Mena, M. Pandya, A. Bansal, F. Shearer, and N. Marshall, "Mobile Extreme Convergence: A Streamlined Architecture to Deliver Mass-Market Converged Mobile Devices", *white paper*, Motorola, Inc., February 2004.

[2] "Developing a complete cellular modem on a single processor", *Embedded Systems Europe*, April 2004.

[3] "SC1000-Family Processor Core Reference Manual", StarCore, June 10, 2004.