A Low Power Open Multimedia Application Platform for 3G Wireless

James Song, Thomas Shepherd, Minh Chau, Ayesha Huq, Ikram Syed, Somdipta Roy, Achuta Thippana, Kaijian Shi+, Uming Ko.

Texas Instrument Inc.
12500 TI Boulevard,
Dallas, TX 75243, USA
s-song@ti.com

+IP Design Service, Synopsys Inc.
14911 Quorum Drive,
Dallas, TX 75254, USA
Kaijian.Shi@synopsys.com

ABSTRACT

This paper describes a SOC design of a complex, low power and high performance open multimedia application platform(OMAP™) for 3G wireless. The design integrates a high performance DSP subsystem based on a low power TMS320C55x DSP and an MPU subsystem based on the ARM9 Microprocessor for the optimal combination of high performance with low power consumption. This paper explains the system design and the SoC implementations of the platform.

1. Introduction

Many factors drive the growth of the 2.5/3G wireless markets, but one of the most important is the proliferation of the applications and the growing functionality of wireless appliances. 3G wireless applications require full-featured multimedia services. DSP intensive operations such as MP3 audio and MPEG video encoding/decoding for video conferencing have significant impact on the application performance. It becomes essential that a chip design targeted for 3G wireless applications must provide DSP capabilities[1][2]. Currently, the MPU providers try to address this issue by extending the existing MPU instruction sets to include a number of multimedia functions[3], such as the ARM™ v6 instruction set. Since the included multimedia functions are restricted and the MPU architecture is not optimized for DSP processes, the DSP performance improvement provided by the MPU instruction extension is limited. A superior solution is a system with a dual core (DSP and MPU) architecture to more effectively address the needs for proficient DSP capabilities in 3G wireless applications.

In this paper, the design of the dual core architecture is described which addresses the requirements and challenges in 3G wireless applications. The SoC physical implementation of the design is also examined in detail.

2. The Dual Core Architecture

This unique dual core architecture offers an attractive solution to both DSP and MPU developers, providing the low power real-time signal processing capabilities of a DSP coupled with the command and control functionality of a microprocessor. This low power dual core architecture forms a platform for various 3G wireless multimedia applications. The system structure of the dual core platform is shown in Figure 1.

The MPU core has an instruction cache, a data cache and a write buffer. The DSP subsystem includes a DSP core, a software configurable instruction cache, embedded SRAM and a program ROM. It also has an embedded internal DMA controller for simultaneous data transfers and hardware accelerators for video processing, pixel interpolation and motion estimation.

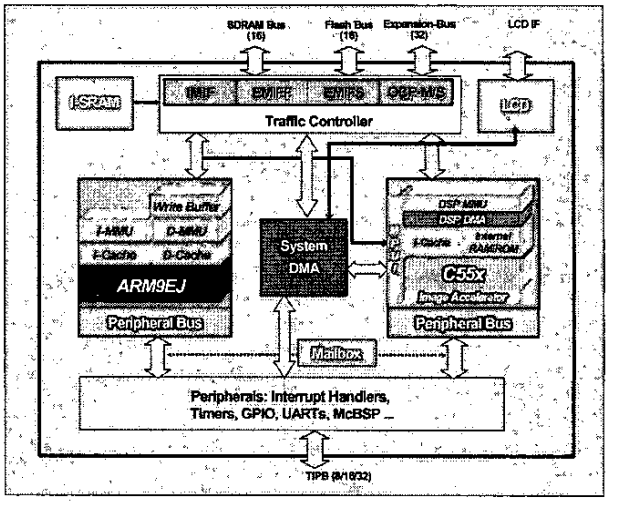

Figure 1 Platform Core Architecture

The MPU is always the master in the system and is responsible for setting up and bringing the DSP out of reset. Once the DSP is out of reset, it can start executing the DSP code. The DSP applications can be either in its own local memory or can be in the shared system memory. The DSP tasks and resources are controlled dynamically through the DSP-MMU by the MPU.

In order for effective communication to exist between the MPU and the DSP subsystems, a handshaking protocol is used. The handshaking between the MPU and DSP cores is

performed through a set of inter-processor mailboxes. In a typical application, the MPU will send a message to the DSP to start performing a certain task by writing to the MPU2DSP mailbox after the DSP-MMU is configured. Once the DSP completes the task, it writes to the DSP2MPU mailbox to indicate that it is done with the current task. This kind of handshaking maintains the system coherency.

To support memory accesses, an optimized traffic controller was designed that allows dynamically configurable data throughput and asynchronous or synchronous scalable operations between the system interface, the DSP and the MPU subsystem. The programmability of the traffic controller allows for efficient performance utilization of the cores, as well as maintaining low system power at the same time. The traffic controller also supports various arbitration algorithms that can be used to control and utilize the available bandwidth efficiently.

The design provides various external memory interfaces to allow glueless hookup to standard memories such as Flash, SRAM, ROM and low power mobile SDRAM/DDR. Also, a rich set of peripherals was designed to support other multimedia functions.

To boost the system performance, the design implemented a 16 channel programmable DMA controller to enable 2-D graphic processing and simultaneous transfers of data between the host and the memories, the host and the peripherals, the internal and the external memories, and from peripherals to other peripherals.

3. SoC Debugging Features

Since the platform architecture is based on multiple cores, it is important to provide an efficient and relative-easy means to trace back an error to the source of the problem. In this platform, various emulation and debugging facilities were implemented such as a Catscan and a window tracer at the system level besides the core specific emulators and tracers in the DSP and MPU subsystems. The Catscan facility allows the users to set a breakpoint, execute emulation, stop the execution at the breakpoint and dump out internal data and states through scan chains. This feature allows developers to isolate a problem fast and easy. The window tracer enables the users to define an address window to monitor bus transactions of all shared target interfaces. The information of the bus transaction is captured when the address of the access address falls in the defined window. The captured data is sent out through a serial tracer interface for software and hardware debugging. These target tracers provide visibility of the order of program execution on a per target basis in the complex system.

4. Low Power Platform

Since this application platform is targeted towards wireless devices, longer battery life is a necessity. Power reduction techniques were implemented at the Process, Architecture, Logic Design and Physical Design phases of the development. The dual core architecture enables assigning a task to one of the processors that is best suited for the task. This dynamic task allocation leads to a significant reduction in the number of processor cycles required to perform a task, which leads to a significant decrease in power consumption.

The OMAP architecture ensures that the software/OS has full control on all the clocking and idle modes of platform. If an application does not need a particular resource, the software can put the resource in an idle mode and even can turn off the power of the resource. This feature enables the application developer to write the application code such that the power consumption is minimized. An aggressive strategy of local clock gating coupled with the reduction of unnecessary signal/bus toggling and an optimal floor-plan has been used to further reduce power consumption.

5. High Performance Graphical Display System

A high performance graphical display system is one of the main requirements in 3G wireless applications, particularly in quality video streaming and games. Many features were designed in the platform to provide for high performance graphical display capabilities.

5.1 Dedicated DMA Channel for LCD Display

A high priority, dedicated DMA channel was implemented specifically for the LCD Controller. This dedicated DMA-LCD channel ensures minimum latency in real-time LCD operations. Also, the system DMA channels were enhanced to implement in hardware essential graphic display operations such as 2D image rotation, transparent copying, solid color fills, and support for multibuffering.

5.2 Multibuffering Support

In most instances, the user doesn't want to see the process of polygons or lines appearing on the screen one after another as they build a single frame. To address this concern, a multibuffering strategy was implemented. The user can designate two sections of memory as frame buffers. While one section (Front Buffer) is repeatedly being rendered to the LCD , the MPU, DSP, or DMA can update the second memory section (Back Buffer) with the next frame of data. Hardware was implemented in the LCD Controller to allow the two buffers to be swapped at the next vertical sync of the LCD. This process can be repeated for future frames. This will produce flicker free

animations and create an impression of smooth movement [1].

To support the technique of multibuffering, two sets of DMA configuration registers (shadow registers) were implemented in the system DMA controller. The programming register set is used to configure the back frame buffer and the active register set controls the front frame buffer. The registers contain the base address of the image in the frame buffer, the type of addressing modes (consecutive read addresses, or 2D addressing modes for image rotation), as well as the image size.

To assist the software programmer, the hardware provides status registers to verify which buffer is currently being rendered to the screen, as well as generating interrupts at the end of each DMA frame transfer.

5.3 Transparent Copy with Source Color Key

Another beneficial graphical operation that was implemented in hardware is the transparent copy functionality for DMA transfers. This feature is very useful for game programmers. Each sprite (a moving image on screen) in a game is represented in memory as a rectangular region. However, the image the sprite represents is usually irregular shaped, such as a person or spaceship. The ability to copy only the irregular shape onto a background, and not the entire rectangle, is desired.

To achieve this functionality, a source color key register was implemented in the DMA channels. This register is programmed by the user to the correct pixel value of the default color key that was created by the artist. A DMA channel is setup to transfer the sprite into the frame buffer. As the DMA transfer is occurring, each pixel of the sprite is compared to the value in the color key register. If there is a match, then the memory write to the frame buffer is not performed. This process continues until the entire sprite is copied. The desired effect is the background remains unchanged and the irregular shape is copied to the frame buffer.

6. Security Access Control

Security control is another important requirement for 3G wireless applications using the Internet. A 3G design must provide the confidentiality to the system ensuring that only the active participants are able to understand the content of the transferred information. Also, the design must prevent data received to be maliciously altered during transmission and must guarantee the communication to the active participants who are able to prove their authentication. Moreover, a 3G design for security must not wrongfully repudiate the transaction of the parties in the transaction, and the design must protect users against pseudonym and anonymity.

To address those issues, a security-distributed scheme was introduced. In this scheme, an optimized combination of selected hardware blocks and a protected software execution environment was implemented. The implementation contained the security control module, which serves as a special purpose hardware that creates an environment for protecting sensitive information from accesses by non-trusted software.

In addition, the secure ROM/RAM modules provide protected executions, and an Efuse is used with a public key to provide a secure boot process in which the boot code/device is protected. A HW-based Random Number Generator (RNG) enhanced the security compared with using a SW-based pseudo RNG. The HW accelerators were used for a crypto engine, which helped enhance the power and performance of the system. Together, all the modules made up a robust design structure to ensure the security for applications that run on the open platform.

7. Open Application Platform

A key component of this application platform is an open software architecture that supports application development and provides a flexible upgrade capability. The architecture includes a framework for developing software that targets the system design and Application Programmer Interfaces (API) for executing software on the target system.

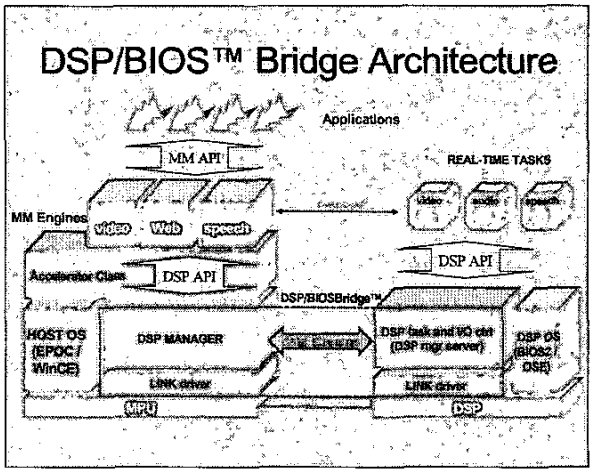

Figure 2 The platform software architecture

To simplify software development, the DSP software architecture is abstracted from the RISC environment. The abstraction is accomplished by defining an architectural interface that allows the RISC to be the system master. The DSP/BIOS bridge interface provides communication that enables the RISC applications and device drivers to:

- Initiate and control DSP tasks
- Exchange messages with the DSP
- Stream data to and from the DSP
- Perform status inquiries

8. SoC Implementation

The integration phase of the design utilized Jupiter® for floor planning, Apollo® for global and detail routing, and clock tree synthesis; Star-RCXT® for parasitic extraction; Hercules® for design rule check, and TI internal sign-off tools for cross-talk, EM and IR-drop checks. Logic and physical combined optimization [4] is essential to this high performance and low power design. Physical Compiler® was used to perform such an optimization. The physical implementation methodology of the platform is described below.

To shorten the time-to-market, the physical design was done hierarchically. The DSP subsystem physical implementation was separated from the MPU system. The DSP subsystem and the MPU system achieved timing closure in layout individually and in parallel. Then, both were abutted with their interface pins aligned to form the platform in layout.

The DSP subsystem and the MPU system are million gate designs. A hierarchical timing was developed for the timing closure flow. The MPU system was partitioned into a MPU core, 11 subchips and 22 soft blocks based on functionality, interconnects and layout constraints. Subchips were only created on those blocks which could be placed in the chip without causing routing congestions and had clear interfaces and derivable quality input and output (IO) constraints. The subchips achieved timing closure individually and in parallel. Then, they were integrated into the MPU system by their timing and physical models. Next, The MPU system was optimized by Physical Compiler® using a two pass placement optimization strategy. In the first pass, hold violations were ignored to focus on optimizing the design for speed and area. In the second pass, A restrictive incremental placement optimization was used to fix hold violations and reduce utilization.

By selectively abstracting 11 blocks into subchips, the design size and complexity at the MPU system level were able to be reduced to speed up timing closure process. By applying the logic and physical combined optimization flow to the soft blocks in the MPU system, quality optimization results were able to be achieved that correlated well with post-layout results.

After achieving timing closure, the DSP subsystem and the MPU system were abutted to complete the dual core platform. Then, post-layout timing was checked using a sign-off STA flow in 10 PVT conditions and 8 operation modes. EM, IR drops and cross talk were checked to meet sign off requirements.

To prevent long nets from being driven by weak cells, which are the main cause of design rule violations and signal integrity problems, the max capacitance of the standard cells were scaled down in the libraries during the placement optimization. Also, the cells' maximum fanout was confined to be 8 loads to control load distribution. This signal integrity problem prevention strategy worked well. There were only a few design rule and cross-talk violations in post-layout.

Long, large fanouts or feedback subchip IO nets can cause antenna rule violations and inaccurate IO path timing at top-level layout. To resolve this issue, a rule based IO buffer insertion and placement method was developed which detected the problematic IO nets, inserted strong buffers to isolate the nets from the IO pins and placed the buffers close to the IO pins. As the result, the subchip IO nets became short point-to-point nets.

9. Summaries

The OMAP Platform was successfully designed and implemented in eight months, and has increased the performance of advanced applications that include graphics, multimedia content, and Java as much as 8x while reducing standby current as much as 10x in wireless handsets and PDAs. As a result of the excellent engineering effort and dual core architecture, the OMAP platform has been integrated into five different chipsets as first passed silicon.

10. References

[1] Sergei Savchenko, Msc., 3D Graphics Programming Games and Beyond, Sams Publishing, 2000, Indianapolis Indiana, p 14

[2] P.E. Gronowski, W.J. Bowhill, et al. High performance microprocessor design. IEEE Journal of Solid State Circuits, May, 1998, pp 676-685

[3] Bill Moyer, Low-power designs for embedded processor" Proc. of IEEE Vol. 89, No. 11, 2001

[4] Hojat and Villarubbia "An Integrated Placement and Synthesis Approach for Timing Closure of PowerPC Microprocessor" ICCD 1997