Associative Processing Array Solves Mobile App Processor Challenges

Ed Jacobs, NeoMagic - July 16, 2003

To implement compelling multimedia applications, the computing requirements for a handheld wireless device increase by over 50-fold when compared to today's voice-only phones. For instance, a phone that delivers MP3 music playback capability only needs about 50 MIPS of computing power, a level readily available from today's mobile phone chip sets. But to deliver compelling, demand-generating 3D gaming content, the computing requirement increases up to 3000 MIPS, a level of performance not available in traditional mobile chip sets.

To handle these multimedia demands, mobile designers are adding RISC-based application processors into their baseband architecture that are optimized specifically to handle functions like video streaming, MP3 audio, and more. The problem with this approach, however, is that these RISC-based processors fail to provide the scalability and low power consumption necessary in today's mobiles. Even if special, dedicated-function hardware accelerators are added to the architecture, these RISC approaches do not provide the flexibility needed in next-generation mobiles.

Recognizing this problem, a new architecture has been developed, which solves the performance problem, and does so at low power. This new architecture, called an associative processing array (APA), employs massively parallel single-instruction, multiple-data (SIMD) computing in a unique implementation that merges processing and memory together. By processing data in a parallel array of computing elements, and storing data in the same array, the APA architecture performs more computations at lower power than sequential processors.

APA: How it Works
APA technology is a massively parallel single-instruction multiple-data (SIMD) processor. The uniqueness of the APA architecture is two-fold: it is an array of parallel processing elements, each merged with memory. The performance of the APA architecture is easily scalable by increasing the array size of the processing elements.

By merging the processing element and memory into one arrayed element, the APA architecture solves a major problem associated with traditional register-based Load/Store and DSP architectures: the need to move data from the register/SRAM file, into the ALU/MAC, and back. This data movement causes both power consumption and performance issues because special design techniques must be used to overcome pipelining stalls, such as register forwarding and pipeline stage interlocks.
The APA architecture enables performance by using up to 81,920 (512*160) simple logical operations in parallel. In practice, the APA architecture operates on "slices" of data and, in the examples below, performs 512 arithmetic operations in parallel. In order to minimize power consumption, the APA multimedia engine can run either synchronous or asynchronous to the rest of the chip. For example, for MPEG-4 (Simple profile, level 3 decoding, 30 CIF frames per second at data rates up to 384kbps), the APA array executes the lion's share of the processing while running at very low clock rate, asynchronous to the rest of the chip, yet using very low power.

As a flexible and programmable multimedia engine, the APA architecture has the processing power to implement not only video codec algorithms, but also such diverse functions as digital still camera (DSC) enhancements, 3D graphics, voice recognition and image object tracking. DSC functionality includes low-light imaging computed for each frame, CMOS sensor bayer for YUV format conversion and de-speckle and noise reduction. These are all, obviously, computation- and data movement-intensive tasks.

To accomplish such tasks efficiently, the associative processor array operates as a smart cache, loading data to the cache/array and then processing that data without it leaving the cache. All processing is reduced to a combination of two primitives: compare and write, executed inside the associative memory. Any logical and mathematical operation can be expressed as a truth table, and compare/write can execute any truth table. The following illustrations demonstrate associative compare and write operations.

**Figures 1 to 4** construct the basic hierarchy of the array, and illustrate the compare operation. Additional unary operations performed by APA including shift row and move column provide enhanced flexibility in arranging data and performing computations.

![Figure 1](image)

**Figure 1:** A single 160-bit APA cached-processor word stores several values.

**Figure 2** shows a 160-bit APA cached-processor word. A word of this length is capable of storing several values of different bit lengths. In this example, the APA data word includes two separate 8-bit input variables and anticipates a 9-bit result value from logical operations.
What is distinct about the APA architecture is that each cached-processor word is able to perform a comparison to a comparand value in place, without moving data. The comparison logic is embedded and built in to the implementation of the APA memory cells, hence it is truly an "intelligent" memory. This localization of the logical operations within the memory itself allows the APA architecture to use significantly less power than a RISC Load/Store architecture because it doesn't have to move data from memory, perform a compare in an ALU and return the result to a flag register.

Comparing Bits
Following a typical comparison operation in the associative processor, several subsets of bits of a full 160-bit word are selected for comparison with a comparand value, while other bits are masked off. A comparand value is sent to each of the unmasked bits in the word, the word compares each unmasked bit and returns a one-bit flag indicating whether its value matched in all non-masked bits. This one-bit flag is stored in a register, called the tags register, and can be manipulated by bit-wise logic (see Figure 2 above).

Figure 3 illustrates APA depth, showing an array of four APA words. Note that all bits in every word are aligned so that the same bit positions are either masked or selected in all APA words in the array. Although the figure shows four APA words, current development efforts focus on economical handheld system solutions with arrays of 512 APA words, although the architecture is easily scalable to 16,000 APA words and beyond.
Figure 3: An array of four APA words is shown. Each data value is bit-aligned in all words. The figure shows a compare operation executed in parallel on all words in the array.

**Figure 4** illustrates a write operation. Note that the write occurs only in the rows identified by the information in the tags register. It also makes use of the mask to identify the bit location within the APA data words that the new value is to be written into. In this example, only the least significant bit of the result area of each APA word is unmasked thus the results of the write will be placed in this set of bit locations within each APA word marked for write by the tags.

![Write Operation Diagram](image)

Figure 4: A write operation executed in parallel on all words in the array that were identified by the tags result of the prior compare.

**ARM and APA**

Typically, applications processors contain both a RISC processor and a multimedia engine. Many handset designers today implement the applications processor with an ARM-based chipset, due to its popularity and availability of industry-standard software tools. In this example, APA implements the multimedia engine.

It’s important to note that the ARM processor and APA are complimentary: ARM is efficient for control-intensive code, while APA is efficient for heavy data processing. ARM is the master and is responsible for configuring the hardware, running the operating system and executing Java and other applications that have limited data. Finally, ARM initiates tasks on APA and reacts to their completion.

The hardware does not dictate the partition of work between processors. For load sharing, smaller applications that run well on APA may be run on ARM instead. For example, during full video codec, voice can be handled by what otherwise would be a near idle ARM. Also, many software applications already exist for the ARM architecture and have been optimized for the ARM926, such as execution of Java, and do not need to be ported to run on APA.

With the ARM and APA-based multimedia engine running concurrently in the same applications processor chip, both processors have access to a shared DRAM and on-chip SRAM. The ARM runs the OS and manages memory allocation of the DRAM. The ARM uses a communication protocol to
initiate tasks to be run on APA. For example, in an MPEG-4 encoding application, the ARM instructs APA to execute motion estimation for all macroblocks in a frame by passing APA a motion estimation task ID, pointers to the current and reference frames, a pointer to the motion estimation output area in DRAM, search range and other parameters. When APA has completed the task for the entire frame, it sends an interrupt to the ARM.

**Wrap Up**
Intensive multimedia processing calls for special-purpose hardware assistance to enhance overall performance for multimedia on traditional RISC or DSP architectures (such as dedicated special-function hardware blocks). Although this special purpose hardware may improve computing performance, its use is single-purpose, and it cannot be used for other purposes making it inflexible for future algorithms, and costly to the overall system design.

APA is a robust alternative for handheld multimedia devices. It provides high-performance parallel processing, power efficiency, scalability, and flexibility. Additionally, it has a clean interface with the ARM processor making it a strong solution for handling multimedia tasks in a mobile, handheld environment.

**About the Author**
*Ed Jacobs* is a staff architect at NeoMagic. Ed has an MS from Stanford University and is the author on four patents related to computer hardware design. He can be reached at ejacobs@neomagic.com.