Journal of VLSI Signal Processing 41, 21–34, 2005

Accelerating Mobile Video: A 64-Bit SIMD Architecture for Handheld Applications

N.C. PAVER, M.H. KHAN, B.C. ALDRICH AND C.D. EMMONS
Intel Corporation, 1501 S. Mopac, Suite 400, Austin, Texas, 78746, USA

Received July 2003; Revised January 2004; Accepted March 2004

Abstract. Providing quality mobile video applications in hand-held mobile devices requires increased computational capability. Using Single Instruction Multiple Data (SIMD) techniques to expose and accelerate the data parallelism inherent in video processing increases performance in handheld and wireless systems. The paper introduces a new 64-bit SIMD coprocessor of the Intel® XScale® microarchitecture which is optimized for low-power handheld applications. The architecture blends the SIMD media processing style with the capabilities of the XScale microarchitecture. This paper provides an overview of the architecture, its instruction set, programming model, the pipeline organization and functional units. The paper also describes how key features of architecture improve the performance of video applications as compared to a scalar implementation. The performance and power improvements based upon measured results are analyzed to show how the opportunities of power savings by reducing the frequency and voltage can be realized.

Keywords: wireless video, SIMD, multi-media, architecture, SOC

1. Introduction

With the convergence of communication and computation capabilities in wireless devices, the growth of many applications, such as video capture, video conferencing, video display are enabled. The demand for increased video image resolution and higher quality output challenges the performance of current wireless platforms. These platforms are composed of a processor, a memory sub-system and many other peripherals integrated as a system on a chip (SoC). As battery life is a premium in handheld applications, it is essential that enhancements to the basic architectures are power efficient.

Optimizing instruction set architecture is a logical approach towards attaining power efficient acceleration of multi-media applications. This paper introduces Intel® Wireless MMX™ technology, a new 64-bit Single Instruction Multiple Data (SIMD) style multi-media coprocessor. An overview of the architecture is presented with details of the micro-architecture provided. The performance characteristics of the SIMD architecture are evaluated with analysis of measured results of a MPEG-4 video decoding application.

2. Technology Overview

Single instruction multiple data processing (SIMD)[1] is a common form of parallel processing used extensively today in desktop and mobile computing platforms [2–5].

Using special SIMD instructions it is possible to operate on multiple data elements packed into a single register, where each data element is treated as an independent item. For example, in a packed addition operation four additions are calculated in parallel using SIMD techniques rather than the single addition performed in a scalar processor. Other styles of SIMD processing techniques have been proposed, some of which are 32-bit architectures [6].

Identifying the opportunity for parallel processing of data elements is key for the successful application of SIMD techniques. Fortunately many multi-media applications do contain data parallelism suitable for SIMD acceleration. For example, in image processing the same operation may be repeated on many pixels in the image. While SIMD processing is not a new technique, applying the technique to handheld and wireless platforms does offer new opportunities to increase power efficiency of multimedia applications. Wireless MMX technology brings the SIMD style of media processing into the handheld platform. In particular it integrates equivalent functionality to all of Intel®, MMX™ technology [7] and the integer functions from SSE [3] with the XScale micro-architecture [8]. The base functionality of both MMX and SSE are augmented with features to ensure the seamless integration with the load/store architecture [9] of the XScale microarchitecture.

Like MMX technology and SSE, Wireless MMX technology utilizes 64-bit wide SIMD instructions, which allows it to concurrently process up to eight data elements in a single cycle. This style of programming is well known to existing software developers. There are three supported packed data types (8-bit byte, 16-bit half word and 32-bit word) and the 64-bit double word. The elements in these packed data types may be represented as signed or unsigned fixed point integers.

SIMD operations improve the clock cycle utilization by performing multiple operations in a single instruction. Increased performance can also be translated to increased power efficiency. The rest of the paper shows how the SIMD extensions improve performance and power for mobile video applications.

3. Architecture

The Intel® XScale® microarchitecture [8, 10] is a high speed, power efficient implementation of the ARM V5TE Architec-ture [11–13]. The implementation of the microarchitecture employs a 7-stage pipeline coupled with a 32 KByte instruction cache and a 32 KByte data cache.

The ARM architecture supports extension of the baseline processor functionality by use of a pre-defined framework for adding up to 16 coprocessors. This framework includes definition of instruction encoding space that can be used to extend the instruction set and basic classifications of instruction types. The ARM coprocessor model supports three classes of coprocessor instruction:

- Load/Store—transfers data to or from the coprocessor to main memory
- Register transfer—transfers the contents of registers between the coprocessor and the main core
- Coprocessor Data Processing—operate on data from the coprocessor registers and retain the result in a coprocessor register.

Wireless MMX™ technology uses the coprocessor framework to extend the base functionality of the XScale microarchitecture. For the end user, wireless MMX technology appears as a seamless extension of the XScale microarchitecture instruction set and programming model.

As specified by the ARM architecture, the XScale core is responsible for fetching instructions and data from memory and delivering them to the coprocessor. Instructions are fetched and dispatched by the XScale core, with wireless MMX instructions being routed to the coprocessor for execution. The XScale core is also responsible for all communications with the data caches. The address calculations for memory transactions are calculated from values stored in the XScale core register file. Wireless MMX technology takes advantage of the existing memory subsystem of the XScale microarchitecture without the need for extra dedicated memories, and the power consumption associated with them. The power efficiency is further improved by using advanced power management, where the Wireless MMX unit is only activated, when required, on an instruction by instruction basis.

The coprocessor is tightly coupled to the XScale microarchitecture so there is a single thread of control. This simplifies the task of code development compared to other multi-core solutions.

3.1. Micro Architecture

The Wireless MMX unit comprises five key functional units to implement the programmers model. Figure 1 shows the organization of the functional units within the coprocessor.

The Shift and Permute Unit is responsible for performing shift and permute operations. These operations include alignment, logical and arithmetic shift, rotation, packing and shuffling.

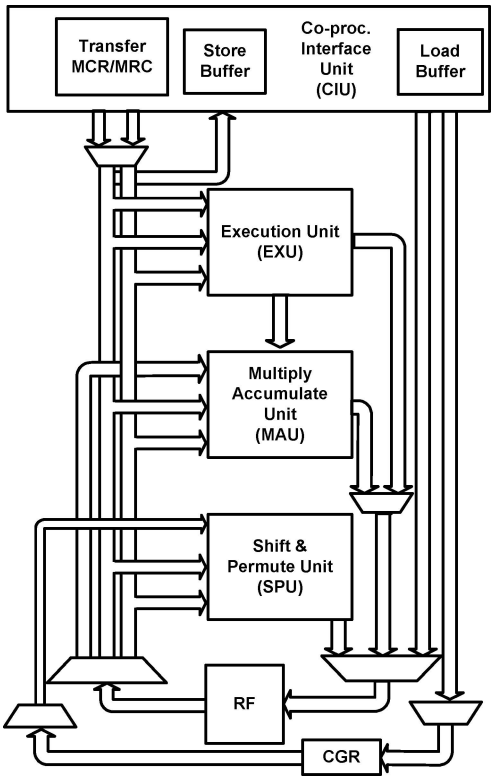

Figure 1. Wireless MMX micro-architecture.

The Execute Unit is responsible for performing arithmetic and logic operations and it also provides a saturation capability. Operands may be received from the SPU when saturation is required and the CIU when the transfer instructions are issued.

The Multiply and Accumulate Unit is responsible for performing all multiply and accumulate operations. Operands are also received from the EXU when executing the sum of absolute difference instruction. The MAU unit is a three stage pipeline with internal accumulator forwarding.

The Coprocessor Interface Unit transfers data between the Wireless MMX™ unit registers and the Intel® XScale® microarchitecture. In addition to supporting coprocessor data transfer, it is also responsible for storing and loading data to and from the memory.

The Main Register File (RF) is organized as sixteen 64-bit registers, located in the coprocessor 0 space (CP0). The large register file allows the Wireless MMX instructions to support an increased number of intermediate values in complex calculations. For example, multiple output samples of a filter may be calculated in parallel. The increased storage allows the programmer to take advantage of the spatial and temporal data locality as found in many multi-media applications. This reduces the required load/store bandwidth and improves processing efficiency. Alignment support instructions also increase the effectiveness of this data re-use. These combined techniques are referred to as *Multi-Sample Technology.*

The control and status registers are mapped into coprocessor 1 space (CP1). There are four 32-bit general-purpose control registers used for alignment and shift control. As the shift and alignment offset is usually invariant across the inner loops the registers are designed to hold constant values. This saves the use of a packed data register.

There are also a number of control and status registers mapped onto coprocessor 1 space:

wCASF-SIMD Arithmetic Status Flags— A set of four flags for each element of a byte/half-word/word/double-word operation

- N when the result is negative
- C when there is a carry out
  Z if the result is zero
- V if the result over-flowed

wCSSF-SIMD Saturation Flags which set the respective flag if an operation on a particular element saturates

wCON-coprocessor Control Register— support for reducing memory traffic on a context switch.

3.1.1. Integrating SIMD and Conditional Execution. A key feature of the XScale micro-architecture is the ability to define a predicate that can be used to determine whether a particular instruction is executed as a function of the current state of the processor arithmetic flags. This conditional execution mechanism increases programming efficiency by removing explicit compares and branch instructions. To seamlessly integrate the SIMD coprocessor into the ARM architecture, wireless MMX instructions can also be conditionally executed. It is also possible for the coprocessor to generate arithmetic flags that can be used later in conditional execution. For a particular SIMD operation, a set of arithmetic flags is generated for each data element processed. For example, if four half-word elements are processed then four sets of flags are generated and stored in the wCASF. Each set of flags consists of the N, Z, C and V indicators.

From the WCASF, the SIMD flags can be combined and transferred to the XScale core current program status register (CPSR) where they can be subsequently used for conditional execution. Combining the SIMD flags and transferring them to the CPSR allows new predicates for conditional execution to be supported. If the SIMD flags are combined with an "OR" operation then an "if any" predicate can be created. For example for the condition "if any SIMD field is zero execute the following instruction", this would set the Z flag in the CPSR if any Z flag is set in the WCASF. Any subsequent instruction that used conditional execution based upon zero being set would effectively be using the "if any" predicate. Similarly if the SIMD flags are combined with an "AND" operation the "if all" predicate can be created (e.g. "if all fields are zero then execute the next instruction"). This mechanism for combining the SIMD flags and using them in conditional execution is known as *group conditional execution.*

3.2. Pipeline Structure

To achieve the same clock speed as the XScale core, the Wireless MMX unit employs the same super-pipelined structure, as shown below in Fig. 2. Here in the first two stages of the pipeline (F1, F2) the instructions are fetched from the instruction cache. In the next stage (ID) the instruction is decoded and the operation is determined. At this point if a wireless MMX instruction is identified it is transferred to the coprocessor during the ID stage. After decode the operands are read from the register file in the RF stage. Once the operands are available the main execution of the operation happens in XI and X2 with the final result writeback in XWB.

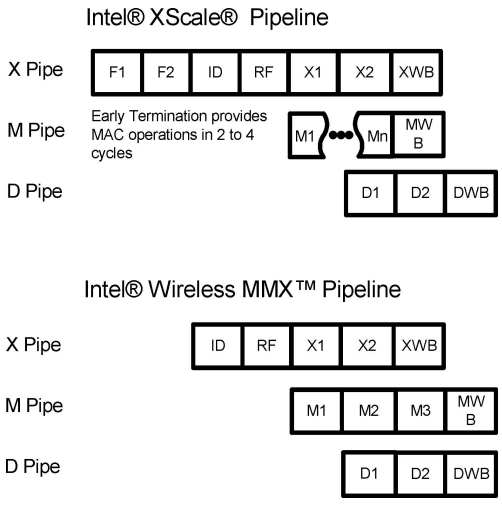

Figure 2. Pipeline organisation.

As well as the main execution pipeline the Wireless MMX unit contains two other side pipelines, one for memory operations routed via the XScale core (Dl, D2 and DWB) and another side pipeline for multiply operations. (Ml, M2, M3 and MWB). The addition of these two side pipelines allows longer duration operations (e.g. memory access) to complete in one pipeline while new instructions can continue to be processed in the other pipelines.

The XScale microarchitecture with Wireless MMX technology is a single issue machine. The Wireless MMX and main core pipelines operates in lock step, providing a single thread of control with no complex synchronizations required between the two units. An instruction can be issued to any resource in the main core pipeline or coprocessor pipeline that is not busy.

The architecture allows instructions to be retired out of order. This is possible because the instructions are committed in the X1 stage so anything past this is guaranteed to complete. The pipeline organization also supports multiple out-standing loads, which improves memory throughput. This feature combined with the out-of-order completion, allows non-dependent instructions to execute, reducing the impact of memory latency in system on a chip applications.

3.3. Instruction Set Overview

Wireless MMX technology provides a rich set of instructions that perform parallel operations on multiple data elements packed into 64-bit registers. In addition, backwards compatibility is provided with existing XScale microarchitecture coprocessor 0 instructions, which operate on XScale core registers (TMIA, TMIAxy, TMIAPH).

In total there are 43 new instructions in the architecture. Table 1 provides an overview of the instruction set.

These instructions have been designed to provide equivalent functionality to MMX technology and integer SSE to accelerate the porting of applications from the desktop and mobile PC platforms to handheld and wireless systems.

The architecture provides enhancements above and beyond the baseline MMX technology. In particular the three operand instruction format of Wireless MMX^{TM} technology allows a destination register to be specified

Table 1. Key instruction overview of wireless MMX technology.

Instruction	Description
WACC	Addition of all 8 × bytes, 4 × half words, or 2 × words in 1 reg
WADD/WSUB	Add/Subtract 8 × bytes, 4 × half words, or 2 × words
WALIGN	Extracts 64-bit value on byte boundaries from 2 × 64 bit.
WAND/WOR/ WXOR	64-bit logical operations
WAVG2	Unsigned average on vectors of 8- or 16-bit data
WCMPEQ WCMPGT	Compare 8 × bytes, 4 × half words, or 2 × word elements in parallel. Result is mask of 1 's if true or 0's if false
WMAC	Multiply four signed or unsigned 16-bit half words in parallel and accumulate with a 64-bit register.
WMAX/WMIN	Vector maximum/minimum selection
WMADD	Multiply four 16-bit words in parallel and add
WMUL	Multiply four signed 16-bit words in parallel. Low- or high-order 16 bits of 32-bit result are produced
WROR/WSRA/WSLL/WSRL	Rotate right, shift arithmetic/logical right, left shift of 4 half words, 2 words, or 64-bit double word, in parallel
WPACK	Pack double word to words or words to bytes
WSAD	Sum of absolute differences on 8 × Byte or 4 × 1 6-bit data.
WSHUFH	Shuffles 16-bit data specified by an 8-bit immediate
WUNPCK	Unpack 8 × bytes, 4 × 1 6-bit half words, or 2 × 32-bit words
WLDR/WSTR	Load/Store Byte, half word, word or double word
TANDC TORC	Logical operations across the fields of the SIMD PSR (wCASF) and sends the result to the XScale core CPSR
TBCST	Broadcasts a value from the XScale core source register to every element in the packed destination register
TMIA	32 × 32 signed multiply-accumulate using operands from the two source XScale core registers
TMIAxy	16 × 16-bit multiply-accumulate selecting high/ low 16-bits from two source XScale core registers
TMIAPH	Dual 16 ×16 multiply accumulate into 64-bit using signed 16-bit operands from the two source XScale core registers

that is different from both source registers. This eliminates the need to copy the source operands to avoid over-writes that are common in the original MMX architecture. Code density can be improved by removal of the now redundant MOVQ instruction used to save copies of the source operands.

The three operand instruction format also allows accumulating functions that were not previously possible. Wireless MMX technology provides a single instruction multiply accumulate operation (WMAC) which is useful for signal processing operations. It also provides an accumulating version of sum of absolute differences (WSAD) which is used for video encode operations.

4. Architectural Support for Video

The video encoder/decoder is one of the more demanding application domains targeting both desktop and mobile platforms. The video coding standards MPEG-1, MPEG-2, MPEG-4 [14, 15], ITU-T H.261, H.263 [16] and the upcoming H.264 [17] all employ a transform based hybrid motion compensated encoding scheme. In these standards, both spatial as well as temporal redundancy in a sequence of images is exploited to reduce the amount of data which is to be transmitted or stored. The algorithms used to implement the compression have been evolving from one standard to the next, and in general maintain a high degree of similarity. The more computational intense algorithms also display a high degree of parallelism. This parallelism is particularly well suited to the SIMD instruction support introduced with Wireless MMX technology.

The flexibility provided by a software implementation of the critical components of the codecs allows both adaptation to a specific standard as well as addressing the enhancements introduced in the future.

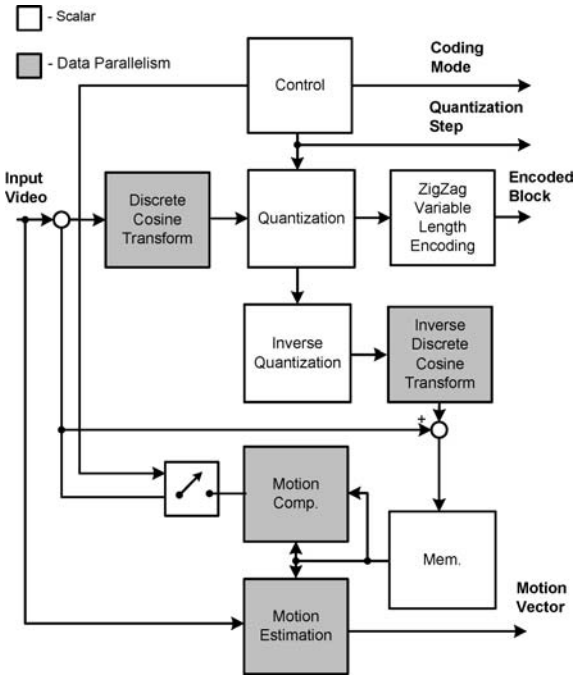

Figure 3. MPEG-4 simple profile video encoder.

4.1. Data Parallelism

To successfully employ SIMD optimization techniques it is important to identify the data parallelism opportunities in an application. There are many such opportunities to use SIMD techniques in both video encode and decode.

Figure 3 shows a typical video encoder, in this case MPEG-4. The units shaded indicate the algorithm sub functions where data parallelism can be accelerated by Wireless MMX technology.

Spatial compression removes redundant data within any given image. It is applied to all image data during video compression by applying the 8×8 2D Discrete Cosine Transform, DCT, followed by quantization and Huffman encoding. Temporal compression removes redundant data within a sequence of images. It is accomplished by taking advantage of the fact that there is a great deal of similarity between sequential frames of motion video. Objects moving across the field of view do not change very much between frames, they often only move to different positions. This similarity between frames is used to reduce the amount of transmitted data required by the digital motion video standards and also places the highest burden on the video encoder.

There are basically two types of image coding, intra and predictive. The into coded pictures do not employ temporal compression techniques and are encoded as still images without referring to other frames. In contrast, the predictive coded picture relies on reference pictures which are acquired at a different point in time. The reference picture may be a single picture which occurred at an earlier point in time, or it may be two pictures which occurred at both an earlier and later point in time. These are referred to as P-pictures and B-pictures respectively.

During the temporal encoding process, component blocks of sequential video frames are compared at displaced positions which represent candidate motion vectors in the horizontal and vertical directions. The task of calculating displacement values, motion estimation, involves finding the best match between corresponding areas within two sequential video frames. This motion estimation is a key area where Wireless MMX^{TM} technology can offer acceleration.

The video decoder performs the opposite to the video encoder; it decodes a compressed video stream and displays it. Figure 4 shows a typical MPEG-4 video decoder and the data parallelism opportunities that Wireless MMX can accelerate.

The compressed video stream is provided as input to the decoder. The inverse operations indicated by the intra or predictive coding modes are then performed. If the image has been into coded, the

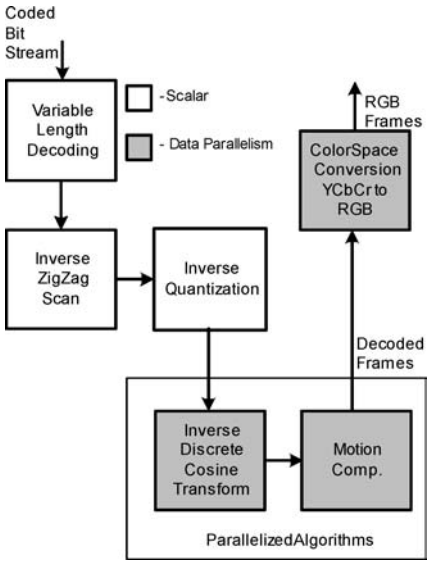

Figure 4. MPEG-4 simple profile video decoder.

decompression involves Huffman decoding followed by inverse quantization and inverse 8 × 8 DCT. If the image has been predictive coded, the decoding operations will also include motion compensation. Motion compensation and inverse DCT are two areas that exhibit the data parallelism that is suitable for SIMD acceleration.

For video conferencing applications, a mobile device will capture image data from a CCD or CMOS sensor and compress this for transmission to the remote party while simultaneously decoding the compressed stream sent from the other end of the video link. The image data from both video streams must also be displayed on the local display resources. This implies some manipulation of the image data since the output and input to video codecs is in YCbCr color space and the display is normally provided in RGB color space. The color space conversion is a significant part of the decode-to-display pipeline. The percentage of time spent in each part of a typical decode-display pipeline is shown in Fig. 5. The exact profile obtained is a function of image content, bit-rate, resolution and compression standard. However, this shows that the same areas that were identified earlier as containing data parallelism also account for a significant proportion of the execution time of the decoder. Any improvements to these routines therefore have a big impact on overall decoder performance.

Motion compensation, IDCT and color conversion account for the majority of the cycles consumed during the video decode and display. Wireless MMX technology provides significant improvement by processing multiple data items at once as described in the following sections.

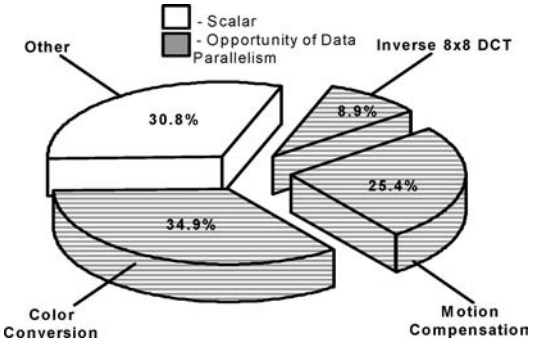

Figure 5. Typical MPEG-4 decode profile of the coastguard sequence at GIF resolution.

4.2. Programming Model

The programmers model is an extension of the XScale microarchitecture programming model and, as such, Wireless MMX instructions can be interleaved with XScale microarchitecture instructions. The management of data pointers and loop control are implemented using the XScale registers with all arithmetic processing being performed by the coprocessor.

The Wireless MMX unit and the XScale processor also share the same cache and memory hierarchy so there is no complex interaction required to exchange data between the two. This allows Wireless MMX technology to be targeted easily at the critical routines and data to be shared without concern of synchronization or data coherency issues. The Wireless MMX unit and the XScale core share a single thread of computation which makes code development and debug simple. With packed data types each SIMD instruction can process multiple data elements. This reduces the required instruction cache bandwidth to process the same amount of data and this subsequently improves the power efficiency by reducing the number of instruction cache accesses.

4.3. Instruction Support

Wireless MMX technology contains a number of features that improve the efficiency of video applications. The general SIMD operations such as addition, subtraction, and multiplication, provide the basic components for algorithm development. In addition to these several special instructions are introduced which accelerate particular aspects of video encoding and decoding.

The motion estimation and motion compensation are both algorithms which naturally operate on byte boundaries. In order to expose the parallelism of the algorithms for SIMD implementations, it becomes necessary to align the data prior to presentation to the execution resources. The WALIGNI and WALIGNR instructions provide a mechanism for reducing the overhead associated with the alignment. The WALIGNI instruction is useful when the alignment is known beforehand, and the WALIGNR instruction is useful when the exact alignment is calculated when the algorithm executes. Both of these instructions operate on register pairs. Figure 6 shows the operation of the WALIGN instruction.

Motion compensation is a prime candidate for SIMD acceleration. When the motion vector indicates a

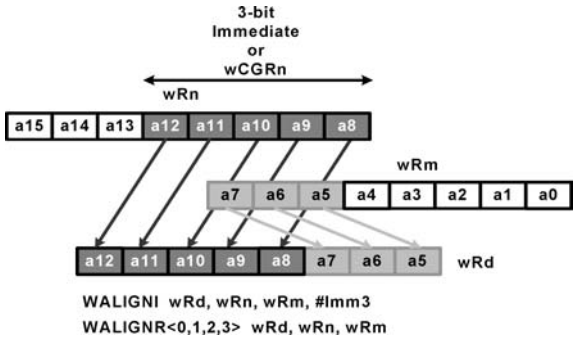

Figure 6. The alignment instruction.

non-integer displacement, the spatial interpolation of the pixels may be indicated in one of three fashions, horizontal, vertical, and diagonal, (1/2X, 1/2Y, and 1/2XY). The two way average (WAVG2) instruction is provided to accelerate this function. The instruction allows the average of up to 8 pixels pairs to be calculated independently in parallel. Figure 7 shows the byte version of this instruction where 8 pixels can be processed. The instruction also provides the capability for optional biased rounding which is useful for the different rounding specifications of each standard.

For horizontal interpolation, WALIGN and WAVG2 can be used together effectively. The WALIGN instruction is first used to extract a 64-bit value from two source operands at any byte boundary. It is then used again to obtain a one pixel shifted version of the aligned row.

$$\text{Aligned Row} : \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1 \times 0$$
$$\text{Aligned} + 1 \text{ Row} : \times 8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1$$

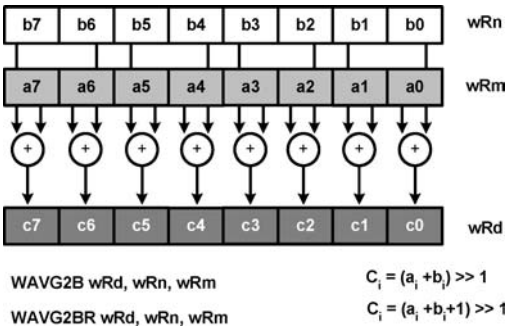

Figure 7. The byte average instruction.

```
@ r0 - Pointer to 8x8 block in the source plane.
@ r1 - Pointer to 8x8 block in the destination plane.
@ r2 - Width of the source and destination plane.

    MOV      r5,#7          @Setup mask 0x7
    AND      r7,r0,r5       @Isolate alignment
    MOV      r12,#4         @Setup loop count
    SUB      r0,r0,r7       @Clear pntr LSB's
    WLDRD    wR0, [r0]      @Load 8 source pixels
    ADD      r8,r7,#1       @Increment alignment+1
    WLDRD    wR1, [r0,#8]   @Load 8 source pixels
    TMCR     wCGR1, r7      @Setup alignment control
    ADD      r0, r0, #1     @Add image width
    WLDRD    wR2, [r0]      @Load 8 source pixels
    TMCR     wCGR2, r8      @Setup alignment control
    WLDRD    wR3, [r0,#8]   @Load 8 source pixels
LOOP:
    ADD      r0,  r0, r2    @Add image width
    SUBS     r12, r12,#1    @Decrement loop count
    WALIGNR1 wR4, wR0,wR1   @Align source pixels
    WALIGNR2 wR5, wR0,wR1   @Align+1 source pixels
    WLDRD    wR0, [r0]      @Load 8 source pixels
    WAVG2BR  wR6, wR4,wR5   @Average 8 pixel pairs
    WLDRD    wR1, [r0,#8]   @Load 8 source pixels
    ADD      r0,  r0, r2    @Add image width
    WSTRD    wR6, [r1]      @Store 8 pixels
    ADD      r1,  r1, r2    @Add image width
    WALIGNR1 wR4, wR2,wR3   @Align 8 source pixels
    WALIGNR2 wR5, wR2,wR3   @Align+1 8 source pixels
    WLDRD    wR2, [r0]      @Load 8 source pixels
    WAVG2BR  wR6, wR4,wR5   @Average 8 pixel pairs
    WLDRD    wR3, [r0,#8]   @Load 8 source pixels
    ADD      r0,  r0,r3     @Add image width
    WSTRD    wR6, [r1]      @Store 8 pixels
    ADD      r1,  r1,r3     @Add image width
    BNE LOOP
```

Figure 8. Wireless MMX assembly code for an 8×8 block 1/2X interpolation.

The data aligned in this fashion can be presented to the WAVG2B instruction. However, if the three LSBs of the pointer indicate a 64-bit boundary, then one of the alignments will not be necessary and a slightly more efficient code sequence can be developed. Similarly if the LSB's indicate byte 7 within a 64-bit word, an alignment can also be eliminated. The assembly code for performing the general case of 1/2X interpolation is provided in Fig. 8. The interpolation is performed on a 8×8 block which may be aligned on a byte boundary in memory. The throughput is 19 cycles per loop iteration resulting in a total of 92 cycles including the overhead for setup and loop management. Since the sequence is fairly straightforward, unrolling the loop completely is a reasonable approach to eliminate 15 additional cycles providing $\sim$16% additional performance improvement.

Compared to the sealer implementation, the 1/2X interpolation for an 8×8 block using Wireless MMX^{TM} executes approximately 3 times faster. Although the

WAVG2B instruction is applied only 8 times compared to 64 scaler instructions to process the block, a speedup of 8× is not achieved. This is due to the fact that for the SIMD approach the arithmetic represents only a portion of the total cycles. Additional operations are necessary to manage the alignment when preparing the operands for presentation to the parallel execution resources. These operations include isolating the three LSB's of the pointer to the block, transferring them to the alignment control resisters using the TMCR instruction, and performing the alignment using the WALIGNR instruction.

For video encoding, motion estimation and motion search algorithms can consume as much as 40% of the encode operation [18]. The objective of motion estimation is to find a best match for a source block belonging to some frame N, in a search area existing in a temporally displaced frame N-k. The region of the displaced frame is referred as the search region. The current standards have source blocks both of 16 × 16 and 8 × 8 size. Some proprietary algorithms use source blocks of 32 × 32, 4 × 4 and 2 × 2. The 16 × 16 block size is most commonly used with the 8 × 8 block size in advanced prediction modes for H.263 and MPEG-4. The selection of whether a 16 × 16 or 8 × 8 block should be used for the motion search is based on a comparison of the minimum distortion between a 16 × 16 block and four 8 × 8 blocks.

To determine how well a source block matches a candidate displaced block, the distortion between the source and target blocks is compared. The comparison criteria are generally based on distance and several measures can be used based on complexity and efficiency. The Sum of Absolute differences (SAD) is the most commonly used criteria, as it reduces the complexity of the implementation:

$$\mathrm{SAD} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |a(i, j) - b(i, j)|, \quad N = 2, 4, 8, 16$$

The sum of absolute difference instruction (WSADB) is designed to accelerate exactly this part of the motion estimation function of a video encoder. The WSADB instruction, shown in Fig. 9 can perform eight absolute difference calculations in parallel, accumulate all of the results and keep a running total. This allows a whole row of an 8 × 8 block to be compared to a corresponding candidate 8 × 8 block row in a single instruction.

Other instructions which play a role in accelerating video applications include the WMADD instruction which contributes to the improved IDCT performance used in the residual compression and the WMAC which allows efficient filtering operations for audio data.

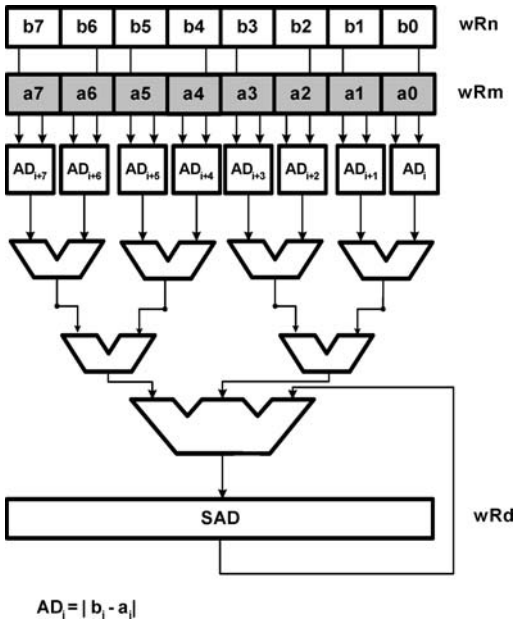

Figure 9. The sum of absolute difference instruction.

4.4. *Data Organization for Video*

Video compression algorithms display predictable data access patterns. For example, many operations are often limited to a 2 dimensional block (e.g 8 × 8). The larger register file of the Wireless MMX unit can be effectively used as a level-0 cache and the data locality of the algorithm optimized. With sixteen 64-bit registers an entire 8 × 8 block can be stored in only eight registers. This still leaves eight registers available for intermediate data calculation and temporary storage. This is particularly useful for the motion search used in video encode as it enables the source 8 × 8 block to be stored in the register file and reused for the comparison at each candidate search position. This dramatically reduces the required load bandwidth of the search algorithm, with the corresponding reduction in power consumption.

Another method of utilizing the large register files is to concurrently compute a number of different block comparisons in parallel and keep all the running totals in different registers. For example, with Wireless MMX technology it is possible to perform a single pass

half-pixel motion search, by calculating the 8 different half-pixel block SAD calculations and keeping all the intermediate results active. At the end of the single pass through the data, the lowest SAD value is selected as the best match. This technique makes maximum use of the loaded data and again contributes to reducing the power consumption by minimizing memory accesses. It is also possible to use a combination of macro-block storage and multiple output calculations to further improve video processing efficiency.

5. Implementation

Wireless MMX technology was implemented as a co-processor to the XScale microarchitecture in a system on chip. The system was composed of XScale, Wireless MMX, memory sub-system with instruction and data caches each of 32 KByte size. The caches were organized as 32-way associative with round-robin replacement method. The implementation supports the capability to scale the operating frequency of the processor. The implementation also allows scaling the voltage. As voltage decrease so does the maximum supported frequency.

6. Power and Performance Results

The SIMD 64-bit architecture extensions offer improvement in performance on wireless video applications and offer interesting power savings opportunities. This section presents some preliminary results for Wireless MMX technology acceleration on video decoding performance The results compare performance and power consumption with and without Wireless MMX technology enabled. An MPEG-4 simple profile decoder [14, 19], running on a bare metal (no OS) system, was used for the experiments. Different clock frequency, different video data clips and bitrates were used to demonstrate the benefits of the Wireless MMX technology. As with any video decoder, the observed results will vary with image content, resolution, bitrate, operating system and other system considerations.

6.1. Kernel Acceleration

Figure 5 showed a typical distribution of the cycles during a scalar implementation of the MPEG-4 video decoder. This figure showed that the key kernels which can be optimized for a video decoder using Wireless MMX are the motion compensation (integer and fractional), inverse DCT and color conversion. Word and sub-word level data parallelism can be extracted effectively on these kernels. Figure 10 shows the typical average acceleration on each of these kernels. Here Int_MC represents integer motion compensation. Also, *MC_R* and *MC_NR* represent non-integer motion compensation with and without rounding respectively for 8 $\times$ 8 macro block of image data. *CC_YUV_RGB* represent color conversion from YUV to RGB and *Inv_DCT* refers to 8 $\times$ 8 inverse Discrete Cosine Transformation (DCT) of 16 bit precision. The kernel routines were optimized to take advantages of the optimized instructions (such as WMAC, WMADD, WAVG2 etc.). Multi-sample techniques were used to make effective use of the register file. Software pipeline, memory prefetching was also used to reduce dependency on the memory latency.

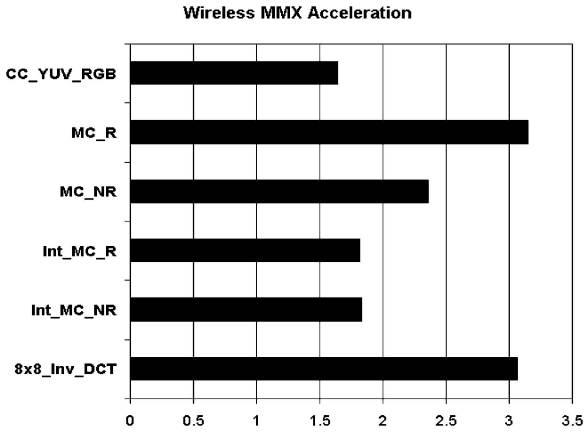

Figure 10. Wireless MMX acceleration for key kernels ii MPEG-4 video decode.

6.2. Application Acceleration

The motion compensation, color conversion and inverse DCT kernels comprise more than 60% of the cycles of a scalar implementation of a video decode applications (Fig. 5). The gain demonstrated in Fig. 10 directly translates to a performance improvement in the end application. Figure 11 shows the frame rate of a laboratory test of a video decoder running on test silicon of the XScale microarchitecture with the Wireless MMX unit enabled and disabled. The video codec is an MPEG-4 decoder, simple visual profile, running on a bare metal system (without OS) decoding the standard

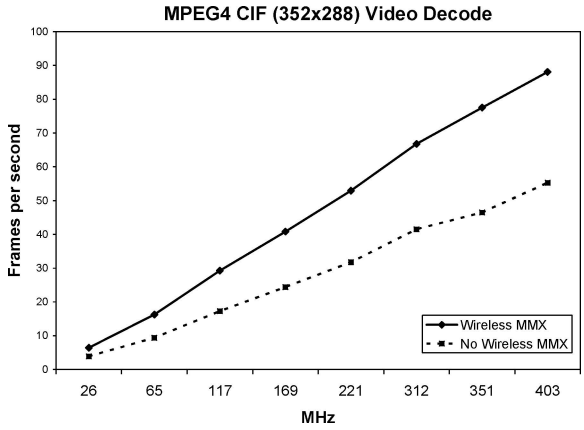

Figure 11. Framerate achieved with and without wireless MMX for MPEG-4 decode of coastguard.

coast guard sequence [15] at GIF (352 × 288) resolution. At the same frequency, Wireless MMX technology gives approximately a 60% improvement in decode frames per second.

This improvement in processing efficiency can be utilized in a number of ways.

- The frame rate can be maintained using a lower fre quency (and power) to achieve it.
- A higher frame rate can be supported at the same fre quency
- A high resolution video stream can be supported at the same frequency.

6.3. *Frequency Scaling*

For many applications, the number of frames displayed per second rate is kept constant. Video conferencing, real-time streaming video are examples of such rate controlled applications. Improvement on frames per second offered by Wireless MMX technology can be used to reduce the operating frequency while maintaining the same performance, for a given frames per second target. The savings in frequency is measured by frequency scaling factor. Average frequency scaling is measured as follows. Here f_{Scalar} and f_{SIMD} refer to frequency requirements of scalar and Wireless MMX implementation respectively. Average frequency scaling factor for Wireless MMX technology for video decode applications can be measured by comparing the slope of the graph shown in Fig. 11. For the measured data in Fig. 11 the average frequency scaling is 1.6.

$$FS = \frac{f_{\text{Scalar}}}{f_{\text{SIMD}}} \tag{1}$$

Since the power consumed in a device is linearly proportional to the operating frequency, the frequency scaling (FS) allows the device to consume a corresponding smaller amount of power.

6.4. *Voltage Scaling*

Using frequency scaling with Wireless MMX technology allows the same video frame rate to be achieved at lower clock frequency. At this lower frequency it may also be possible to reduce the supply voltage. As power is a quadratic function of voltage, any reduction in supply voltage has a quadratic impact on power consumption. Wireless MMX technology supports both voltage and frequency scaling.

The impact of the voltage scaling on the power can be quantified by measuring *Power efficiency*. *Power efficiency* between a scalar implementation and Wireless MMX technology can be defined as follows.

$$PE = \frac{P_{\text{Scalar}} - P_{\text{SIMD}}}{P_{\text{Scalar}}} \tag{2}$$

Here P_{Scalar} and P_{SIMD} refer to power consumption of scalar and Wireless MMX implementation respectively.

Figure 12 compares the power efficiency between Wireless MMX and a scalar implementation with both voltage and frequency scaled compared to just frequency scaled. The exact savings depends on the targeted frame-rates, for higher frame-rate the difference in frequency requirements (with and without Wireless MMX) is higher offering a larger range for scaling the voltage. It is also interesting to note that impact of voltage scaling on the power efficiency is significant. The experiments show that voltage scaling increase the power efficiency improvement by up to $\sim 2.4\times$.

6.5. *Adaptive Frequency and Voltage Scaling*

Changing frequency and voltage during application run-time can be used to take advantage of processor idle time to reduce power consumption. There are many possible algorithms [20, 21] to trigger voltage and frequency changes which are beyond the scope of this

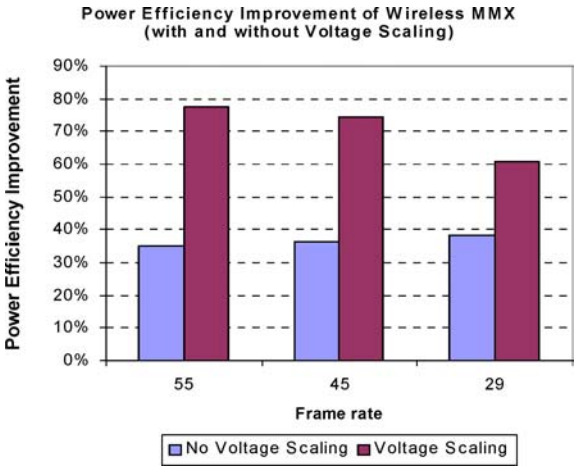

Figure 12. Power efficiency improvement due to voltage scaling at different video decode framerates.

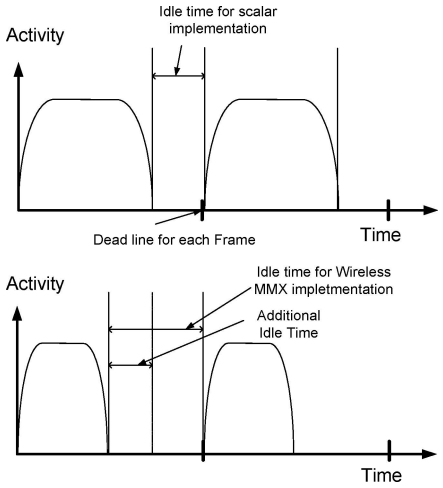

Figure 13. Idle time improvement during video decode using wireless MMX technology.

paper, but many of these algorithms use a function of processor activity and idle time to determine power management policies. For rate controlled video decode algorithms, the regularity of the frame-arrivals, frame-deadlines simplifies the frequency and voltage management [21]. The more idle time between computing the next frame and it being required is an opportunity to reduce power consumption. Figure 13 shows that by using Wireless MMX technology it is possible to compute the frame earlier and have more processor idle time where the voltage and frequency can be reduced and power savings accrued.

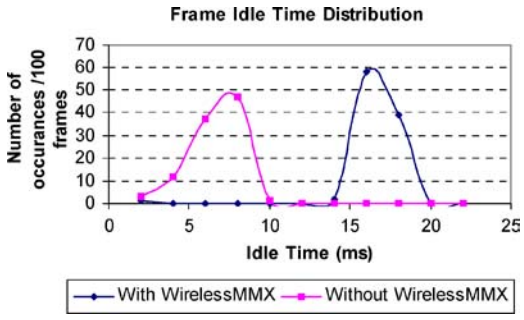

Figure 14. Distribution of slack time between frames.

The voltage and frequency scaling algorithms typically have a lead time (for detection of power saving opportunity and the time to change between power states). The transition time of frequency and voltage changes takes a finite time to happen, in particular voltage changes may take many mSeconds. These two factors pose a lower bound on the usable idle time, i.e. an idle time lower than this bound cannot effectively be used for adaptive voltage and frequency management.

For the coastguard sequence at 30 frames a second (GIF size), the histogram of the idle time between frames (frame idle time) is shown in Fig. 14 This shows that Wireless MMX technology can make use of voltage and frequency management more effectively due to the additional idle time between frames. For rate controlled video applications, this offers an opportunity for extra power savings.

The processing time for each frame depends on the number of motion vectors and their types (integer or fractional). In a scalar implementation the decode time of a frame varies considerably based on the motion vectors, leading to a wider spread in the idle time distribution. In contrast, using Wireless MMX technology the decode time for each frame does not vary as much as the cost to process each motion vector is smaller. This leads to a narrower distribution of the idle time. A narrower distribution on the idle time can make it easy to adopt many predictive algorithms [20] for power reduction, since the idle time is now more predictable.

To demonstrate that the advantage in idle time holds true across various encoding rates and different types of video sequences, percentage idle time has been measured for multiple scenarios. Figure 15 shows the percentage idle time as a function of video clip bit-rate for three different GIF (352×288) video sequences, with Wireless MMX unit enabled (WMMX) and disabled (NoWMMX). It can be noted that the decline in

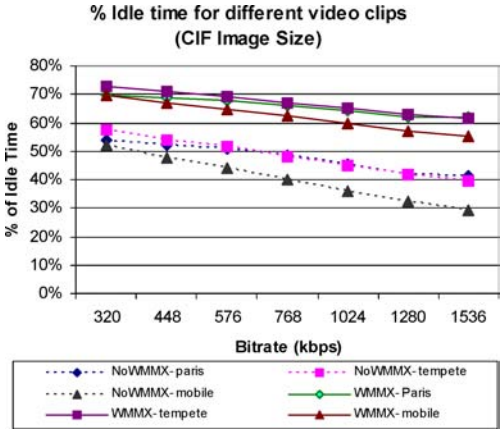

Figure 15. Comparison of idle time vs. bitrate with and without wireless MMX technology.

percentage idle time for Wireless MMX is of a lower rate than without it.

7. Summary

Wireless MMX technology has been optimized to allow power effective wireless video applications. In this paper we have described the architecture and instruction set of Wireless MMX technology as it applies to the video technology.We have disclosed some preliminary measurements and benchmarks which display the performance delivered by this technology. The relative power saving also have been quantified for different video clips and bit rates and the effect of frequency and voltage scaling has been demonstrated.

Acknowledgments

We acknowledge the significant contribution of the entire Wireless MMX technology development team in Austin, TX, Chandler, AZ and the systems engineering in Hudson, MA.

References

1. Michael J. Flyn, *Computer Architecture: Pipelined and Par allel Processor Design*, Jones & Bartlett Pub, May 10,1995.
2. Alex Peleg and Uri Weiser, "MMX Technology Extension to Intel Architecture," *IEEE Micro*, vol. 16, no. 4, 1996, pp. 42–50.
3. Keith Diendroff, "Pentium III = Pentium II+ SSE," *Micro Processors Report*, vol. 13, no. 3, 1999, pp. 6–11.
4. Ruby, Lee,"Subword Parallelism in MAX-2," *IEEE Micro*, vol. 16, no. 4, 1996, 51–59.
5. Tremblay, Marc, et al., "VIS Speeds Media Processing," *IEEE Micro*, vol. 16, no. 4, 1996, pp. 10–20.
6. D. Brash, "ARM-V6 Architecture White Paper" http://www.arm.com/support/White_Papers, January 2002.
7. Uri. Weiser, et al., *The Complete Guide to MMX™ Technology*, Mcgraw-Hill, 1997, ISBN 0-07-006192-0.
8. http://www.intel.com/design/intelxscale/
9. "Intel® XScale® Microarchitecture for the PXA255 Processor User Manual," http://www.intel.com/design/pca/applicationsprocessors/manuals/278796.htm
10. J.L. Hennesy and D.A. Patterson, *Computer Architecture: A Quantitative Approach,* 2nd ed., San Francisco, California: Morgan Kaufmann, 1995.
11. David Seal, *Advanced RISC Machines Architecture Reference Manual*, Prentice Hall, 1996. ISBN 0-201-73719-1.
12. S.B. Furber, *ARM System-on-Chip Architecture*, Addison Wesley, 2000. ISBN 0-201-67519-6.
13. http://www.arm.com/aboutarm/55CE4Z/$File/ARM_Architecture.pdf
14. "MPEG4 Overview (V.21)," Edited by Rob Konen, ISO/IEC JTC1/SC29/WG11 N4668.
15. International Organization for Standardization, "ISO/IEC JTC1/SC29/WG11N1902 14496-2 Committee Draft (MPEG-4)." November 1997.
16. IUT-T Recommendation H.263: "Video Coding for Low Bitrate Communication," Geneve, 1996.
17. "Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 ISO/IEC 14496-10 AVC)," T. Wiegand and G Sullivan (ed.), March, 2003.
18. Peter Kuhn, *Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation*, Kluwer Academic Press, ISBN 0-7923-8516-0.
19. "Integrated Performance Primitive," http://intel.com/software/products/ipp/
20. Benini et al., "A survey of Design Techniques for System-Level Dynamic Power Management," *IEEE Transactions on Very Large Scale Integration Systems (VLSI)*, vol. 8, no. 3, 2000.
21. Frank Bellosa, "Bibliography on Power Management," http://www4.informatik.uni-erlangen.de/Projects/PowerManagement/Bibliography/
22. Pouwelse et al., "Power-Aware Video Decoding," 2001 Picture Coding Symposium, April 2001.

Nigel C. Paver has 13 years experience with the ARM architecture, and in the Intel PCA Components group in Austin, Texas, he is responsible for the architecture and implementation of multimedia

coprocessors for the Intel XScale micro-architecture. He is also involved in product architecture and definition of Intel PCA processors. Before Intel, Nigel was one of the lead designers of the early AMULET asynchronous ARM microprocessors at the University of Manchester. He was also vice president in a startup company which used asynchronous design techniques to produce a low-power asynchronous DSP core. Nigel holds a Master of Science degree and Ph.D. in computer science from the University of Manchester and a Bachelor of Science degree in electronics from UMIST.
nigel.paver@intel.com

Moinul Khan is a multimedia product architect at Intel Corporation PCA Components group. He is responsible PCA graphics and security architecture. His research interests are virtual prototyping, signal processing algorithms and architecture and communications networking. Before joining Intel he was a technology specialist and founding member of a startup at ATDC, Georgia. He worked on his doctoral research at Georgia Center for Advanced Telecommunications Technology at Georgia Institute of Technology. He received his B.Tech form Indian Insti-ture of Technology and MSEE from Georgia Tech. He also worked as a research member for Canadian Institute for Telecommunications Research and Bell Communications Laboratories.

Bradley C. Aldrich joined Intel in 1997 where he is currently an architect within the PCA Components Group. His current work includes the development of coprocessor instruction support in addition to image capture and display technologies for XScale based application processors. He was previously a member of the Intel/Analog Devices joint development architecture team responsible for video enhancements for the Micro Signal Architecture. Prior to that he was a video system architect in Intel's Digital Imaging and Video Division working on CMOS sensors, still cameras, and tethered PC based video peripherals. He has also worked as a device engineer for Motorola and as a test engineer for Tektronix. He received a BSEE in 1988 and MSEE in 1994 from the University of Texas at San Antonio.

Christopher D. Emmons received a Bachelor of Science degree in Computer Science from the University of Texas at Austin in 2003. He joined Intel in 2001 and is currently a multimedia architect responsible for algorithm development and performance optimization for handheld products within the PCA Components Group. Prior to this he worked as an applications engineer providing performance and power analysis in support of product marketing groups. His research interests include video compression, operating system design, and dynamic resource management.