Architecture Considerations for Multi-Format Programmable Video Processors

Jonah Probell

VideoBits.org
Research, Consulting, Training, Entrepreneurship
Waltham, MA, USA

Abstract. Many different video processor architectures exist. Its architecture gives a processor strength for a particular application. Hardwired logic yields the best performance/cost, but a programmable processor is important for applications that support multiple coding standards, proprietary functions, or future changes to application requirements.

Programmable video processor architectures achieve best performance through the use of parallelism at the data (SIMD), instruction (VLIW), and multiprocessor level, and optimally sized ALU, multiplier, and load/store datapaths.

Because low-cost memory architectures are not optimized for the random access patterns of video processing, the performance of video processors is often limited by memory bandwidth rather than processing resources. Careful data organization alleviates memory bandwidth limitations.

When choosing a video processor it is important to consider many factors, particularly performance, cost, power consumption, programmability, and peripheral support.

Index Terms—processor architecture, SIMD, VLIW, multiprocessing, software programmable processor, hardwired processor, data tiling

1. INTRODUCTION

Algorithm experts write software to implement creative methods for encoding, decoding, or enhancing digital video, but might not know how to extract maximum performance from their processor. Chip designers have experience enhancing the capabilities of processors through forms of parallelism and special purpose logic, but might not know how to apply those to build a processor specialized for video. This paper introduces algorithm experts and processor designers to each other's concepts through examples of video processors and applications in common commercial use today.

2. HARDWIRED VS. PROGRAMMABLE PROCESSOR ARCHITECTURES

Processors are either hardwired (fixed function) or software programmable. Hardwired processors are designed to perform a single function. Programmable processors complete complex functions by performing various simple instructions in the order specified by a software program.

The flexibility of a programmable processor to perform different functions requires a general-purpose architecture that compromises on performance for each function. Each data processing instruction reads input data from a storage element and writes its result to a storage element. By performing complex functions as a series of simple instructions, many more data reads and writes are performed in a programmable processor than in a hardwired processor. Each read and write takes time, reducing overall throughput, and consumes power, wasting energy and generating excess heat. A programmable processor requires sufficient internal storage to efficiently perform its most demanding function. That consumes a larger silicon area, resulting in greater manufacturing expense.

Hardwired processors are preferable for highly reliable, low cost, or low power applications. Processors consisting of simple algorithm-module based designs lend themselves to surveillance, military, and aerospace markets. Products sold in high volumes, such as DVD players or set-top boxes for a single content delivery system like DirectTV, call for chips containing hardwired processors designed for a small silicon die area and resulting low costs. For mobile phones with a single video content delivery system or other battery powered handheld applications a hardwired processor optimized for low energy consumption is important.

Programmable processors are preferable for applications that support many standards or proprietary processing. Applications that decode content from the Internet, such as IPTV set-top boxes or portable media players, benefit from the ability to support different standards with different software. Applications with proprietary encoding schemes, such as advanced surveillance systems that allocate a large number of bits to regions of interest while coding the rest of the frame with high compression, require the ability for system designers to write their own software.

Most video processor architectures achieve a balance

between the flexibility and the performance required for their target application by combining elements of hardwired and programmable architectures. A hardwired video encoder processor, such as the Vweb VW2000, performs block level rate control in an internal microcode processing engine.[1] Because of the great number of modes in the H.264 coding standard (CABAC/CAVLC, MBAFF, FMO, macroblock partitioning, intra prediction modes, ½ / ¼ pixel interpolation) most hardwired H.264 video processors such as the Conexant / Amphion CS7050 have an embedded RISC processor core that executes instructions from firmware to determine the mode of operation to perform on each NAL packet or video block.[2] To support multiple coding standards such as H.264, VC-1, and MPEG-2, some hardwired processors are enhanced with logic specific to multiple standards, such as support for IEEE and integer DCTs, 4x4 and 8x8 residual transform blocks, 2-tap bilinear and 6-tap bicubic interpolation filters, and multiple Huffman code probability tables.[3][4] The Hantro 7170 is one such processor, and supports decode of H.264 Baseline Profile, MPEG-4 Simple Profile, H.263 Profile 0, VC-1 Simple and Main profiles, and JPEG Baseline profile.[5]

Programmable processors for video applications are often enhanced with hardwired coprocessors, accelerators, or instruction set extensions that eliminate the need for a large number of software instructions. Entropy coding or decoding is a function commonly performed in a hardwired state machine within programmable processors, such as in the Tensilica Video Engine.[6] Other video processing functions that are commonly accelerated with specialized hardwired logic include multi-pixel sum of absolute differences for motion estimation and filtering such as for inter-pixel interpolation and deblocking filters.

For a programmable processor for digital video to succeed in the market there must be software available for it. Some vendors of programmable processors, such as Broadcom with the BCM7411D HD disc player ASSP, provide the software required for target applications and do not support customer programming of the processor.[7] Some vendors hardwire the instructions for their processor into ROM, rendering the processor nearly as inflexible as a hardwired processor.

3. PARALLELISM IN PROGRAMMABLE PROCESSORS

Video applications have a high degree of data parallelism. For example, each coefficient in a block is scaled by the same quantization (QP) factor and adjacent vertical pixels are multiplied by the same horizontal filter coefficients and vice versa. Where the video processing algorithm has data level parallelism the performance of a processor is greater if the processor has single instruction multiple data (SIMD) instructions that, with the execution of a single instruction, operate on adjacent data elements in parallel.

All but the lowest resolution digital video systems employ processors with SIMD instructions. SIMD instructions have been added as instruction set extensions to all commercially significant general-purpose processor instruction sets, as shown in Table 1.

Table 1. SIMD extensions to commercial instruction sets.

Instruction set	Extension
x86	Intel MMX, AMD 3DNow!, Intel SSE
PowerPC	Freescale/IBM/Apple AltiVec
ARM	ARM NEON
SPARC	Sun VIS
MIPS	Silicon Graphics MDMX, Lexra Radiax, MIPS Technologies DSP ASE

Even with SIMD extensions, general-purpose processor architectures perform relatively poorly on video processing applications. Only x86 processors, with exceptional clock speeds and widespread availability, are used alone for many video processing applications — especially cost insensitive ones.

Extensible instruction set architectures such as those from ARC and Tensilica can be extended to achieve good video performance. Both vendors are working on video processing extension packages for their architectures.

Video applications have a high degree of instruction level parallelism. Many constituent algorithms require non-interdependent instructions that can be performed in separate parallel execution units. For example, one row of a block is multiplied by filter coefficients, the next row of a block is loaded from memory to the register file, and the address of the next block is calculated. Where video processing algorithms have instruction level parallelism, processor performance is greater if simultaneous operations in separate execution units are controlled by a very long instruction word (VLIW).

VLIW architectures are rare among general-purpose processor architectures but common among digital signal processor architectures (DSPs). Table 2 shows the type and number of execution unit in common VLIW processors.

Table 2. Common VLIW DSP processors.[8][9][10][11]

Vendor	Processor	Address	ALU	MAC
Texas Instruments	TMS320'62 TMS320'64	6		2
Freescale / StarCore	SC140	2	4	
Analog Devices	TigerSHARC	2	2	
PixelWorks / Equator	BSP-15	2	2	

Even with VLIW SIMD instruction sets, general-purpose DSP architectures can only perform mid-range video processing applications. Even at its relatively high clock speed of 600 MHz, the Texas Instruments TMS320DM6446 DaVinci chip is only capable of decoding H.264 at SDTV resolution.[12]

Superscalar processors perform simultaneous instructions on multiple execution units but allocate instructions to execution units at run time. Such processors require much more hardware complexity for a smaller performance benefit than is achieved by the instruction level parallelism of a VLIW, which relies on the allocation of instructions to execution units at compile time. Superscalar architectures only make sense for processors that must support a large base of legacy code, such as the x86 instruction set—this is not the case for video processor applications.

Video application software can be written with a high degree of thread level parallelism. In such software constituent algorithms are performed independently and simultaneously in threads run on separate parallel processors. Because of the spatial dependencies of blocks within a frame, it is inefficient to divide video decode between threads by assigning portions of a frame to each thread. Because of the temporal dependencies of blocks between frames, it is inefficient to divide video decode between threads by assigning each frame to a different thread.[1] Video decoding applications can be efficiently divided among multiple processor threads by running constituent algorithms in parallel threads and passing data between the threads treating each as a separate stage in a dataflow pipeline. For example, the inverse DCT of residual coefficients for a block can be performed in one thread, the inter prediction read and interpolation of the block performed in another thread, the deblocking filter of the previous block performed in another thread, and the entropy decoding of the next block performed in another thread.[2] Video encoding applications have more flexibility to trade off performance for parallelism. For example, an encoder performing SADs for a motion estimation search can assign the motion search within each of a range of reference frames to a unique thread. Where the video processing application has thread level parallelism, the performance of a system is greater if separate processors execute the constituent threads simultaneously in parallel.

Multiprocessor architectures are rare among general-purpose processors or DSPs but common among video processors. The Cradle Technologies CT3616 has 16 DSP processors and 8 general-purpose processors.[13] The Avieon Neuron 1 has four DSP processors and one general-purpose ARM 9 processor.[14] The Telairity T1P2000 chip has five identical DSP processors.[15]

For video applications designed to exploit parallelism of same-type data elements, such as parallel motion estimation searches in different reference frames or parallel decoding of separate I-frames or picture groups, homogeneous multiprocessing (processing on multiple identical processors) is most efficient. With each processor executing the same algorithms on similar quantities of data, the simplest programming model is to use identical processors. For video applications in which the processing load is divided between constituent algorithmic steps in a dataflow pipeline manner, heterogeneous multiprocessing (processing on non-identical processors) is most efficient. For video coding standards with a relatively simple frequency to spatial domain transform algorithm, such as the inverse integer DCT of H.264, and a relatively complex interpolation filtering algorithm, such as the half pixel interpolation of H.264, the processor executing the inverse iDCT thread has far less computation work to complete for each frame and can therefore be a smaller and simpler processor than the processor executing the interpolation thread. Tuning each processor for its intended algorithm yields a smaller, less expensive chip that consumes less energy. The benefits of tuning the

[1] Encoders are free to divide frames temporally and, for standards that support slices, to divide the frame spatially. Decoding for standards that support no interframe coding, such as motion JPEG, can easily divide frames temporally.

[2] Video coding standards that support arbitrary slice ordering, such as H.264 Baseline and Extended Profiles, require that deblocking occur only after an entire frame is decoded. For such coding standards, the deblocking thread works on a block from the previous frame, rather than the previous block.

processors within a multiprocessor system must be weighed against the added complexity of multiple compilers and assembly languages.

The best performing programmable processor architectures are those optimized at the multiprocessor system architecture level, VLIW execution unit level, SIMD data level, and instruction set level.

4. DATAPATH WIDTHS

YCrCb video data are represented with 8 bits of resolution in most applications. The sizes of parameters and coefficients coded in the bitstreams of popular video standards vary widely but, for many standards, are expressed in 16 or fewer bits. The filter coefficients and algorithms of H.264 and VC-1 were chosen to ensure that even intermediate values of calculations will not overflow 16-bit arithmetic operations within processors.[3][4] The adders, shifters, ALUs, multipliers, and other data processing elements within programmable video processors are typically 16-bits wide. An IEEE1180-1990 compliant DCT for MPEG-2 requires 22 bit intermediate values and the 10- and 12-bit resolution samples supported by the H.264 Fidelity Range extensions (FRext) require 18 and 20 bit resolution values. Processors intended for such applications yield greater performance with wider datapaths or hardwired accelerators for algorithms that require wider intermediate data values.

The registers within a processor must be large enough to hold a range of SIMD data values. In a processor designed to handle video coding standards for which the theoretical worst-case video sequence will consist of a large number of 4x4 blocks, such as H.264, 4-way SIMD parallelism makes full use of datapaths without cycles of unused datapath bandwidth. Greater SIMD parallelism improves the average case but yields little performance improvement when processing a worst-case stream. An H.264 processor, which must perform 16-bit operations on 4-way SIMD data, will yield the best performance with 64-bit wide registers. A processor designed for a coding standard that operates on 8x8 blocks with 8x8 transform matrices yields better performance with 8-way SIMD parallelism and 128-bit registers.

To keep a processor fed with data to avoid load stalls, on-chip caches and SRAM data stores must be wide enough to accommodate loading a full processor register in a single cycle. An on-chip memory that is wider than the processor registers has no performance benefit and reduces the number of separately addressable memory locations that otherwise give software greater flexibility. If a system includes multiple heterogeneous processors with different internal register widths then, to avoid stalls, each shared memory should match the widest width of registers within the processors that access it.

To reduce chip package cost and minimize I/O power consumption it is desirable to use a significantly narrower data path width to off-chip memory than is used for on-chip RAM. If the memory bandwidth requirements of the application are sufficiently low, a processor with 64-bit or 128-bit wide on-chip memories only requires a 32-bit or 16-bit wide interface to external DRAM. DRAM interface width is less costly in chips packaged with a stacked DRAM die in the same device. 3D graphics chips, such as those from ATI and Nvidia, require and achieve much greater DRAM bandwidth by using wide high-speed DRAM interfaces to DRAM chips within the same package.[16] Due to access delays, doubling interface datapath width does not double bandwidth. In order to determine what off-chip DRAM width is appropriate for a video application, it is important to perform an estimate of the memory bandwidth requirements of the application and a calculation of the memory bandwidth available at the off-chip DRAM interface.

5. MEMORY BANDWIDTH

As semiconductor process technologies shrink, it is feasible to design chips with greater data processing capability. The rates and bandwidths of data transfer between chips do not increase at the same rate as data processing capability. As a result, the performance of processors on video applications is constrained by memory bandwidth to off-chip DRAM memory devices in ever more applications.

A video processor performs three types of accesses to off-chip memory devices:

1. Bitstream (encoder writes, decoder reads)
2. Uncompressed frame (encoder reads, decoder writes)
3. Reads of stored frame buffer(s) data for motion estimation or compensation.

Because the bitstream is compressed, it consumes a negligible amount of bandwidth in most video applications. Reading or writing the uncompressed frame is significant, but the reads required for motion compensation or good quality motion estimation use most of the bandwidth consumed by typical video processors. For systems limited by memory bandwidth constraints, choosing a video coding standard that does not use inter predicted frames alleviates the limitation. To achieve comparable quality, such video coding methods require greater bitstream transmission bandwidth or

storage space, but yield memory bandwidth savings in the encoder and decoder processors.

Because of their common use in PCs, DDR SDRAM chips are sold in large volumes at commodity prices. This makes such memory technology compelling for video processing systems. Because the most common processing tasks in PCs access memory locations at successive linear addresses, DDR SDRAMs are organized in linear rows of 512 to 2k bytes. Therefore if a data element is accessed from the same row as the previous access data transfer will occur on the immediately following SDRAM clock edge. If an access is made to a different row then the transfer will require several clock cycles as the SDRAM switches active rows.

If frame buffer data is stored in SDRAM in raster scan order then each line of a motion compensation or motion estimation block read usually requires access to a different SDRAM row, incurring a row switching delay for each line of the block. A worst-case video sequence, which includes B predicted small blocks with ¼ pel vertical interpolation filtering, can require time to read all blocks in the frame that, due to SDRAM row switching delays, exceeds the frame decoding period. This could break real-time decoding and drop one or more frames.

Delay cycles are reduced, and memory interface performance increased, if rectangular tiles of frame buffer pixel data are stored with one per SDRAM row. The largest possible tiles are used such that the amount of data in each is less than or equal to that in one SDRAM row. For example, a 32x32 pixel tile of 8-bit per pixel frame data fits into a 1K byte SDRAM row. Any pixels of a tile can be read from SDRAM with a single row switching delay.

If frame buffer data is stored in SDRAM in tiled order then each motion compensation or motion estimation block read might fall within a single tile, requiring just a single row switching delay period. If a block falls across the edge between two or the corners of four tiles then two or four SDRAM row switching delays are incurred. In no case will reads of tiled frame buffer data require more row switching delay periods than block reads of frame buffer data stored in raster scan order. Larger tiles and smaller blocks give statistically better SDRAM performance, partially offsetting the increased number of accesses required by using smaller blocks.

For tiled frame buffers, many software instructions and clock cycles are required to calculate the memory address locations and strides, perform the scatter for frame buffer writes, and perform gather for motion estimation or motion compensation reads. Furthermore, image-capture devices, many video decoder post-processors, and display devices require data in either progressive or interlaced raster scan order. Special hardwired logic in the video processor, cache controller, DMA controller, or memory interface is often beneficial for accessing frame buffer data in tiled order and converting between tiled order and raster scan order.

For video encoders that perform motion estimation searches for P and B frames, it greatly reduces off-chip memory bandwidth if an on-chip level 2 cache is implemented to store not only the current SAD block but also surrounding data. Motion search algorithms tend to have a large amount of overlap between successive or parallel SAD operations, so caching data adjacent to a SAD block will often eliminate the need to access that data off-chip for the next SAD.

6. CHOOSING A VIDEO PROCESSOR

The market for video processors supports many different video processor chip designs because there are many applications, each with unique requirements.[17] When performing a technical evaluation of video processors for a system design many factors should be considered.[18]

A video processor must meet the pixel rate requirements of the application. The pixel rate is the product of the height, width, and rate at which full frames are displayed. For many coding standards, the time required to process a worst-case sequence is nearly an order of magnitude greater than for typical sequences. It is important to consider whether the application requires processing the theoretical worst-case or if the system can be designed for the likely worst-case with some reasonable margin and the freedom to drop frames in cases of exceptionally bad sequences. Though, for most standards high quality encoding requires significantly more computational processing than decoding, since the decoder must be able to handle any standard-compliant bitstream, decoders have less flexibility than encoders to trade off quality with processor performance requirements. Encoder designers are free to choose which coding tools within the standard to exploit and to what degree.

Cost is of little concern for aerospace applications, but for applications such as mobile phones and low-end DVD players the cost of the video processor is critical to product success. The largest contributors to the cost of a video processor are the silicon die size and the package. Package cost is minimized by a smaller number of pins and simpler, lower-bandwidth interfaces to off-chip memory and peripherals. Using a smaller manufacturing process technology reduces the silicon die size and the per-chip cost, though smaller process technologies have

higher manufacturing setup costs and are only justified if the video processor will be sold in high volumes. Engineering, marketing, sales, and other business expenses also contribute to the cost of a video processor.

For battery powered systems, such as mobile phones, portable media players, and camcorders, low energy consumption is critical to product success. Energy is lost to current leakage through transistor gates and to transistor switching. Using low leakage and low voltage manufacturing technology reduces leakage energy loss. Careful chip design at the microarchitecture level, including simulations with switching activity interchange format (SAIF) models of the processor running on real video data, reduces the active energy consumed by a video processor. Running the processor at a low clock speed and powering down the processor when there is no work to perform are system level design techniques that save energy. Coding video with I frames instead of P and B frames requires less processing and fewer off-chip memory accesses, both of which save energy.

For system software engineers to implement the algorithms required by their product on a programmable processor requires that the processor vendor support programming. Many algorithms are developed and tested with C code. C compilers for VLIW digital signal processors are effective at partitioning software instructions across the execution units of a VLIW processor. Few compilers make effective use of SIMD parallelism. Video processors supported by compilers that extract SIMD data level parallelism offer an advantage to software developers. Similarly, compilers today are incapable of automatic partitioning of application code into parallel threads that take advantage of multiprocessing. For video applications, this is left to software architects. Video software development is also simplified by the availability of a good debugger and code profiler for the chosen processor.

To take best advantage of the features of a video processor and system peripherals, programmers often optimize critical sections of code in assembly language. Video processors with relatively simple instruction sets and thorough instruction set documentation offer an advantage. Some video processors do not have any compiler support and can only be programmed in assembly language. Though their vendors typically provide software for common video processing functions, these processors are very difficult to program.

Finally, in order to meet application requirements, the peripherals and interfaces of a chip are critical. A video capture or encoder chip requires the correct image sensor device interface for the system. An encoder device often requires noise filtering and sometimes interlacing or deinterlacing. Decoder devices targeting progressive displays require deinterlacing and devices targeting displays of a different resolution than the encoded material require scalers. Many decoders include noise reduction filtering and image or color enhancement. Such functions can be implemented in accelerators, as part of hardwired processors, or in software in a programmable processor if it has sufficient performance.

Encoder and decoder devices must support the off-chip memory device requirements of target applications. This includes one or more DRAM interface standards, non-volatile memory standards such as SD / MMC, a bitstream source interface such as optical disc or broadcast demodulator, and interfaces to other devices such as through USB.

7. REFERENCES

1. VW2000 MPEG-2 Video Encoder Product Brief. Vweb Corporation.
2. CS7050 H.264 Decoder. Amphion Semiconductor Ltd. (2004)
3. Richardson, I.: H.264 and MPEG-4 Video Compression. (2003)
4. Srinivasan, S.: An Introduction to VC-1. (2005)
5. http://www.hantro.com/index.php?107 (2006)
6. Ezer, G.: High-Performance Multicore Video Decoder Technology Preview. Tensilica Inc. (2005)
7. http://www.broadcom.com/press/release.php?id=799861
8. TMS320C6414, TMS320C6415, TMS320C6416, Fixed-Point Digital Signal Processors. Texas Instruments Incorporated. (2005)
9. StarCore SC140 Application Development Tutorial. Freescale Semiconductor, Inc. (2004)
10. http://www.analog.com/processors/processors/tigerSHARC/technicalLibrary/newArch.html (2006)
11. Pixels to Packets: Enabling Multi-Format High Definition Video. Equator Technologies. (2004)
12. TMS320DM644x Processors – Video Benchmarks. Texas Instruments Incorporated. (2005)
13. Product Brief: CT3600 Family of Multiprocessor DSPs. Cradle Technologies Inc. (2005)
14. http://www.avieon.com/neurondflow.php (2005)
15. Product Brief T1P2000 Video Processor. Telairity Semiconducor (2005)
16. http://www.chipworks.com/news/2005_xbox360.asp (2005)
17. http://videobits.org/vendors.html (2006)
18. Processors for Consumer Video Applications, Berkeley Design Technologies, Inc. (2005)