White Paper

Stream Processing: Enabling the new generation of easy to use, high-performance DSPs

> "By re-thinking the roles of the architecture, programming model and compiler tools, SPI has created a new class of DSPs that makes parallel processing practical."
>
> - *Will Strauss, Forward Concepts*

Part number: WP-00003-014
Dated: June 2008

Stream Processors, Inc. (SPI)
455 DeGuigne Drive
Sunnyvale, CA 94085-3890 USA
T: +1 408.616.3338 F: +1 408.616.3337
info@streamprocessors.com

1 Executive Summary

Modern media, image and signal processing applications have an insatiable appetite for compute performance. Although a software programmable processor would be ideal to handle the evolving requirements, today's processors are computationally inefficient, forcing systems designers to make difficult tradeoffs and sometimes use less flexible and more expensive approaches, such as FPGAs or custom ASICs. Furthermore, the regular improvement in processor clock frequency we have learned to enjoy over the last couple of decades has slowed to a crawl.

The industry agrees: the only viable way forward is parallel processing, and while modern VLSI technology in principle allows 100's of cores to be integrated onto a single chip, software development productivity and bandwidth challenges to date have been largely unsolved. Addressing the problem from a new tools-oriented perspective, Stream Processors, Inc. (SPI) brings significant processor performance and efficiency improvements without sacrificing software development productivity.

SPI's Stream Processing™ Architecture combines data-parallelism with a compiler-managed memory hierarchy that enables 1000's of instructions to be in flight every clock cycle. C programming in a single instruction flow provides programmers with easy access to predictable performance without the load-balancing or memory management complexities commonly associated with traditional DSPs or multi-core processors.

SPI's processor solutions empower manufacturers with a software-driven alternative to custom ASICs, FPGAs and multi-DSP solutions to enable differentiation and considerably faster time to market. Storm-1™ Series is a new class of DSP that offers more than 200 GMACS of compute performance with industry-leading power-efficiency that enable designers to implement applications directly in C. A best-of-breed Eclipse-based C development environment enables a broad application fit, such as video surveillance, video conferencing, video analytics, multi-function printers, and LTE/WiMax base stations.

2 Insufficiency of Current Solutions

Media and signal processing requirements are rising sharply. Standards, such as H.264 video compression, can save disk space and network bandwidth, but at the cost of increased computational needs. At the same time, new standards are emerging and there is a need for market-specific tuning and competitive differentiation.

These goals are more easily met in software than in fixed-function hardware. However, traditional processors and DSPs have not yet been successful at approaching the higher efficiency level of special-purpose ASICs. SPI's solutions move beyond the traditional curve to a new location on the flexibility/performance graph, combining 'C'-programmability with ASIC-level efficiencies:

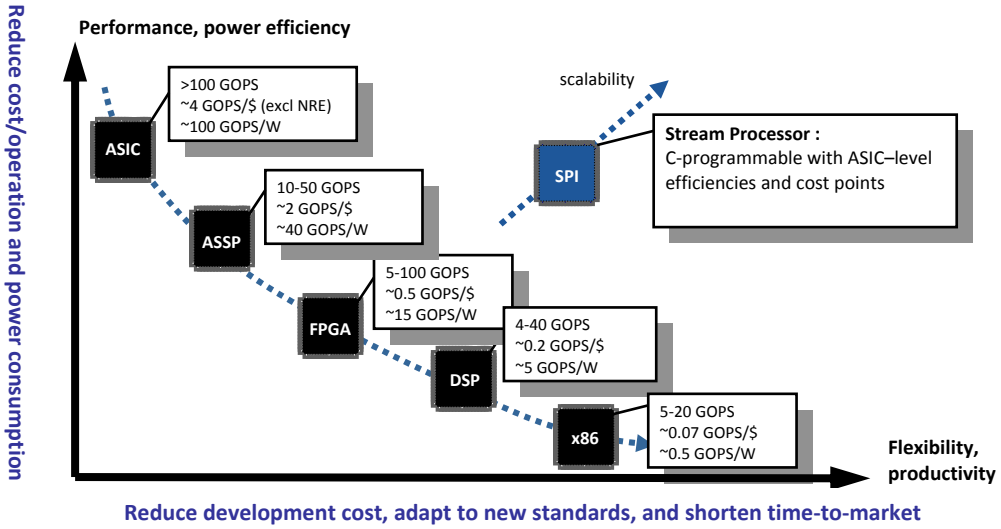

Figure 1. The tradeoff line of solutions

3 Processor Trends

Historically, once processors become powerful and price-competitive enough, they tend to replace fixed-function chips as the dominant vendor choice. In audio, DSPs became the preferred solution over ASSPs because they enabled new formats and shorter time to market. A similar shift occurred for graphics when programmable graphics processors (GPUs) overtook earlier generations of special-function ASSPs.

Processor speeds have also increased over time, driven by higher transistor densities. However, more power is consumed as efficiency improves slower than density, and memory speed lags behind processor speed. The traditional strategy to address this imbalance has been ever larger caches and deeper pipelines, but with instruction-level parallelism "mined out", the industry agrees that the only viable way forward is to put more cores working in parallel on the chip.

While modern VLSI technology allows 100's of cores to be put on a single die, without leveraging locality, bandwidth becomes a bottleneck. Furthermore, programmers may end up spending most of their time on partitioning, synchronizing and load-balancing applications across cores, while unpredictability and insufficient tools support makes it difficult and time-consuming to reach performance. Bandwidth (i.e. locality) and ease of programming (i.e. tools support), not arithmetic, are the critical aspects in modern processor architectures.

A second recent trend is to add on-chip, application-specific hardware blocks. This approach does not address the underlying need for more processor performance, and essentially is just an ASSP model where less flexibility and longer time to market is traded for higher efficiency.

With SPI's Stream Processor Architecture, flexibility *and* performance can be maintained while providing a close to linear path for scalability.

4 Making Parallelism Work – *Bandwidth*

Keeping the ALUs fed is a key issue for parallel processors. Every cycle, thousands of operations can be performed, and with main memory 100's of cycles away, managing data becomes critical to achieving performance.

> ***Analogy***: Imagine doing a plumbing project ... you start and you see you need a pipe ... so you drive to the store ... back with the pipe you discover you need a fitting ... so you drive to the store ... come back with a fitting ... and discover you need solder ... (etc.)

This is analogous to how conventional hardware caches work, and with parallel processors, a single cache miss may mean tens of thousands of processing cycles lost! In addition, for streaming type applications conventional hardware caches are inefficient due to the limited data reuse and unpredictability caused by cache misses.

So, what to do? You make a shopping list! Instead of getting data from main memory right when you discover it is needed, you determine what you need upfront (i.e., compiler pre-allocates local memory and sets up dependencies), and use a smart DMA engine that fetches a batch of data from external memory in time for the task to be executed. This is the basic stream processing principle for data movement. Moreover, the memory hierarchy allows the compiler to use dataflow analysis to maximize data locality to keep data close to the ALUs.

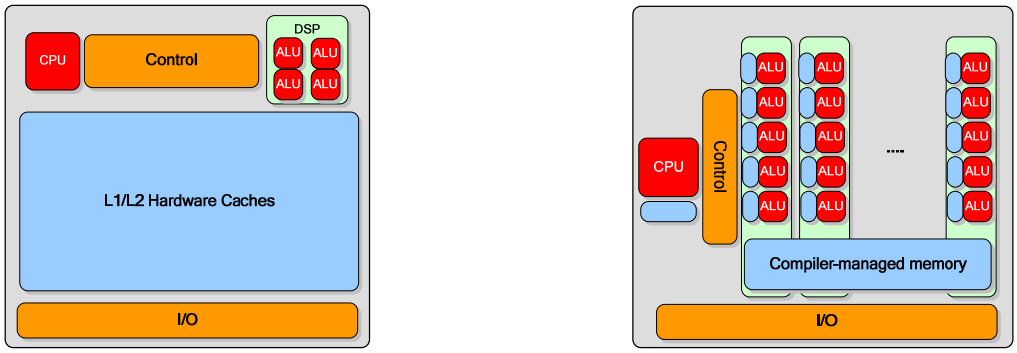

Figure 2. The efficiency of a compiler managed memory allows more compute logic while execution becomes predictable to allow for compilation and data movement automation

5 Making Parallelism Work – *Ease of Programming*

Parallel processors introduce several elements that do not map well to traditional programming models and tools. In traditional architectures, to get performance, the programmer is mostly left with having to manually schedule memory and DMA. For parallel processors with multiple cores competing for resources, manually trying to manage this can quickly become unworkable.

In contrast, the stream processing model, the C compiler manages movement of batches of data that is synchronized with associated kernels for processing. A kernel is an atomic compute-intensive inner loop that processes a batch of data (stream) and will always execute in a known number of cycles based on the size of the stream. This deterministic behavior allows for efficient high-level C compilation and scheduling automation, avoiding the need for proprietary hardware-specific languages and tools.

Outlined in the following sections, SPI leverages data-parallelism which provides implicit load-balancing and eliminates multi-core synchronization issues. For media and signal processing applications, data-parallel execution (all cores run the same code in lock step, on different data) consistently outperforms thread-parallel (each core runs different code) approaches for media and signal processing due to higher utilization and less hardware overhead.

6 Execution and Programming Model

At the top level, the execution model is based on running main application threads on an industry-standard RISC core, that make kernel function calls to a data-parallel unit (DPU) for acceleration. A kernel is a compute-intensive inner loop that process a batch of data records (called a stream) in parallel that are present in on-chip memory. Streams are characterized by a pointer, length, and access pattern. Compile-time analysis sets up stream allocation, and runtime DMA loads kernels and streams based on dependencies, concurrent with execution.

A few keywords define kernels and streams and since parallel processing is within kernels only, a single instruction flow is preserved. Managed by the compiler, kernels often form pipelines, sharing intermediate stream results to avoid external memory roundtrips.

In the simple example below, *format* and *encode* are two kernels. *spi_load* and *spi_store* are SPI keywords.

Main thread code

```
stream  pixel    in_str(STREAM_SIZE);
stream  pixel    tmp_str(STREAM_SIZE);
stream  pixel    out_str(STREAM_SIZE);

i = 0;
while (i < image_size) //loop through image
{
        spi_load(in_str, &image[i]);
        //load stream

        format(in_str, tmp_str);
        //kernel function call

        encode(tmp_str, out_str);
        //kernel function call

        spi_store(out_str, &result[i]);
        //store stream

        i += STREAM_SIZE;
}
```

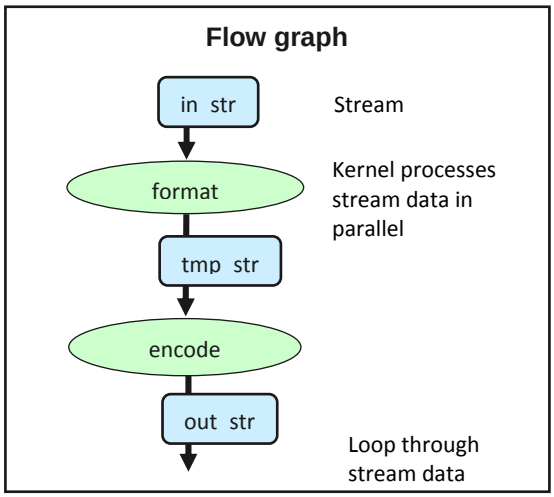

Figure 3. Software organization is similar to traditional DSP programming with attention to "critical loops"

Figure 4 below illustrates basic H.264 motion search. The concurrency of execution and stream loads is automated by DMA based on runtime dependencies, allocation and data live times set up at compile-time.

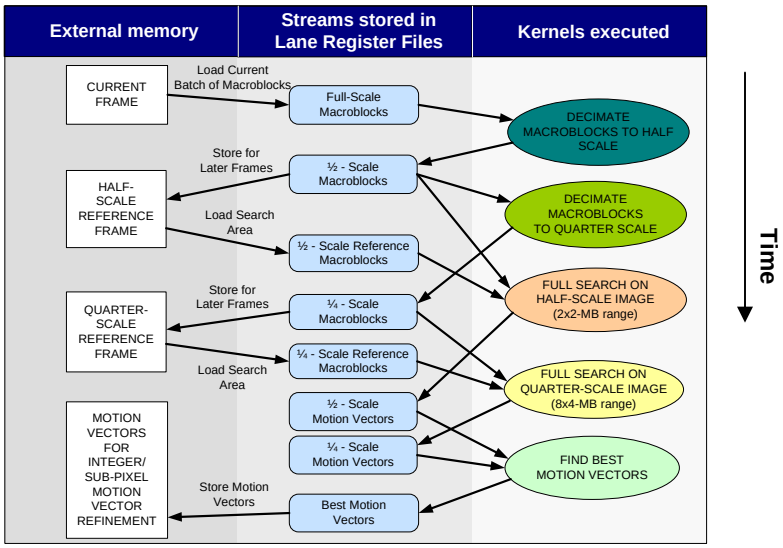

Figure 4. H.264 video encoder motion search example

7 Hardware Architecture

The SPI Stream Processor™ Architecture includes a host CPU (System MIPS) for system tasks and a DSP Subsystem with a MIPS core for the DSP threads and a Data Parallel Unit (DPU) for kernel execution.

For users that do not develop own DSP-level code, the architecture is an SoC with an API to a DSP subsystem with libraries.

The DPU Dispatcher manages runtime kernel and stream loads. One kernel at a time is executed across parallel lanes, operating on streams present in the Lane Register Files.

Storm-1 has 16 lanes; each has five ALUs organized in a VLIW. Each ALU is capable of single 32-bit, dual 16-bit or quad 8-bit arithmetic or MAC ops per cycle. Distributed operand register files allows for a large working set and 10 TBps of ALU bandwidth.

The Stream Load/Store Unit provides gather/ scatter with a variety of stream access patterns. The InterLane Switch is a compiler-scheduled full crossbar for lane data exchange.

The architecture represents a significant advancement over traditional SIMD and vector architectures due to the fine-grained data sharing and concurrent data loads and execution.

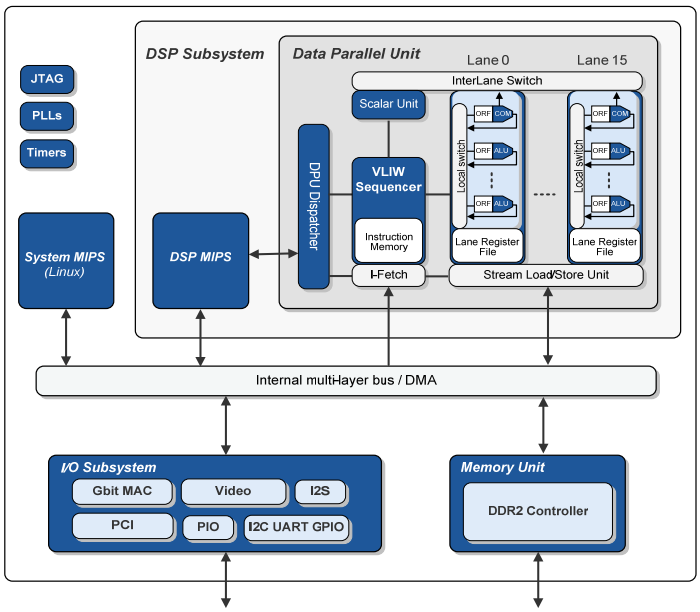

Figure 5. Storm-1 block diagram, a 16-lane stream processor

8 Development Tools

C programming in a single instruction flow with predictable, tools-managed memory management provide a practical approach to parallel processor development. A few keywords are added to ANSI C to tag kernels and define streams, conserving C reference code structure. Intrinsics are available for DSP-type instructions (e.g. DOT product) that have no clear C equivalent.

An Eclipse-based IDE provides a standard-based environment for project management, compiling, profiling and debugging, including target simulation and device execution. It runs natively under Linux, and for PC Windows, Fedora 8 Linux VM is used.

The Stream Processor Compiler (SPC) generates the VLIW executable and C calls for the DSP subsystem that is compiled via standard GCC for MIPS. Performance features such as software pipelining and loop unrolling are supported.

A cycle-accurate visual profiler shows data loads and execution traces with links to corresponding C source. A GDB-based debugger provides programmable breakpoints and single-stepping capabilities.

A Linux-based development kit is available. A messaging framework is provided to link in DSP libraries and manage communication between the System and DSP MIPS cores.

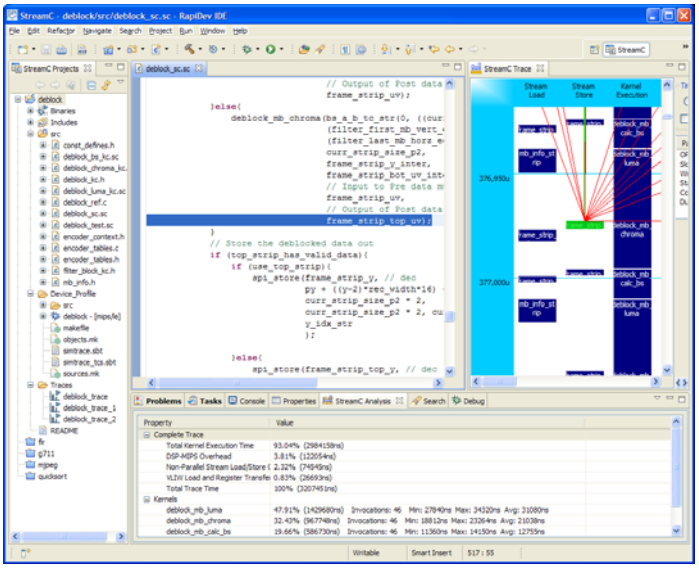

Figure 6. Stream Tools Eclipse IDE with source-level debug and cycle-accurate profiler

9 Performance Evaluation

Massive parallelism, high ALU bandwidth and concurrent, compiler-optimized execution and data loads provide the basis for excellent compute performance. At the instruction level, a 15-stage pipeline feeds a 384-bit instruction to five 32-bit ALUs in a 12-way VLIW that extracts instruction-level parallelism, which combines with 16 lanes at the data-parallel level. Each ALU is fully featured with support for more than 100 instructions, and can perform one 32-bit, dual 16-bit and quad 8-bit operations per cycle. With 16 lanes and 5 ALUs per lane, this adds up to 320 8-bit operations per cycle, including MACs.

Explicit data movement with predictable delays allow SPI's compiler to use sophisticated VLIW scheduling strategies, while offering the programmer high levels of visibility and consistent, predictable feedback. The compiler sets up dependencies for runtime background DMA that overlap with execution — there is no "reconfiguration time" between kernels.

Compared to conventional processor architectures in the same fabrication process and die size, stream processing improves performance and power-efficiency by an order of magnitude for a large group of DSP applications. In fact, the die size can be smaller than a function-specific ASIC due to the reuse of the processing area and efficient memory structure.

A wide range of DSP applications have been ported to the architecture, such as 3D graphics, image recognition, wireless baseband, IP forwarding, and video compression. Unlike conventional vector or SIMD architectures, the finer granularity of the stream processor architecture allows it to also perform well on non-trivial applications with complex control code or serial dependencies, such as H.264 deblocking, 4x4 intraprediction or error diffusion as used in printers. More lanes and ALUs can be added for scalability while retaining source code compatibility.

Description	Storm-1 SP16HP - G220
Metrics	
Frequency	700 MHz
Performance (8x8-bit multiply-add)	320/cycle ; 224 GMACS
DSP kernel performance example	
8 x 8 Forward DCT	5.94 cycles/8x8 block, 7.55 Gpixels/s
3 x 3 2D Convolution Filter	0.13 cycles/pixel, 5.06 GPixles/s
RBGB to YUV Color Space Conversion	0.36 cycles/sample, 1938 Msamples/s
MIMO MMSE Receiver for WiMax/LTE Base Station	0.05 cycles/bit, 14 Gbps
4x4 Motion Estimation Full Search	0.22 cycles/search point
4x4 Forward Integer Transform and Quantization	1.8 cycles/4x4 block
H.264 Deblocking Filter	58.3 cycles/macroblock
Floyd-Steinberg Error Diffusion	1.07 cycles/pixel, 654 Mpixles/s
Application performance example	
H.264 BP Encoder	180 fps D1, Full HD (1080p30)
H.264 BP Decoder	270 fps D1, 45 fps Full HD (1080p30)
Scan-to-PDF 300 dpi MRC pipeline	>80 ppm

Figure 7. Benchmark examples

9.1 Sample Application 1: Video surveillance

Driven to provide better actionable information at lower costs, security companies are deploying new technology in the areas of content analysis and management. IP-based surveillance offers a greater degree of freedom over traditional analog CCTV solutions, allowing vendors to reduce cost/channel while differentiating beyond basic compression and DVR functionality. Such features include:

- improve quality via preprocessing, e.g., MCTF, de-interlacing and image stabilization
- optimize bitrate and rate control – adaptive QP per macroblock based on motion and region (e.g., faces)
- reduce multi-stream bitrates and decoder loads via H.264 SVC (Scalable Video Coding)
- camera tamper detection, e.g., out of focus, covered, repositioned
- robust motion detection based on object labeling
- object indexing for fast video search and retrieval
- multi-megapixel IP cameras with digital PTZ, e.g., multi-window and 180/360 degree de-warping
- video content analysis, e.g., trip-wire and people counting
- authentication and encryption, e.g., 802.1X and watermarking
- decoding and transcoding for NVRs with arbitrary input/output formats and aspect rations
- display processing, e.g., alphablending, overlays, scaling/tiling with support for arbitrary formats
- surveillance tuned image sensor pipeline, e.g., illumination flattening and low-light noise reduction

Fixed-function ASICs are generally not flexible enough to support or keep up with this new wave of functionality, and combining an ASIC codec with a separate DSP may not be cost-effective, nor may a traditional DSP have enough horsepower. For this functionality to be deployed on a mass-level base (such as in every IP camera and on every DVR port); DSP performance must increase significantly while keeping cost down. Even seemingly mundane analytics, such as robust motion detection, have not yet been available at reasonable cost due to high performance requirements.

Another emerging product category is the Network Video Recorder (NVR) where software programmability is required to guarantee decoding and transcoding support for the wide variety of IP Cameras and formats.

Enter SPI: The Storm-1 Series is the market's only solution to offer the DSP performance and cost points to support the required functionality. The SP16HP is well-suited as a single-chip 16-channel DVR and its low-power sibling, the Storm-1 SP8LP, is ideal for a single-chip IP camera solution. SPI and third party partners provide libraries and reference design kits for DVRs, NVRs, video encoder and IP cameras for an easy, fast path to product, while maintaining the flexibility of software for future upgrades and vendor differentiating features.

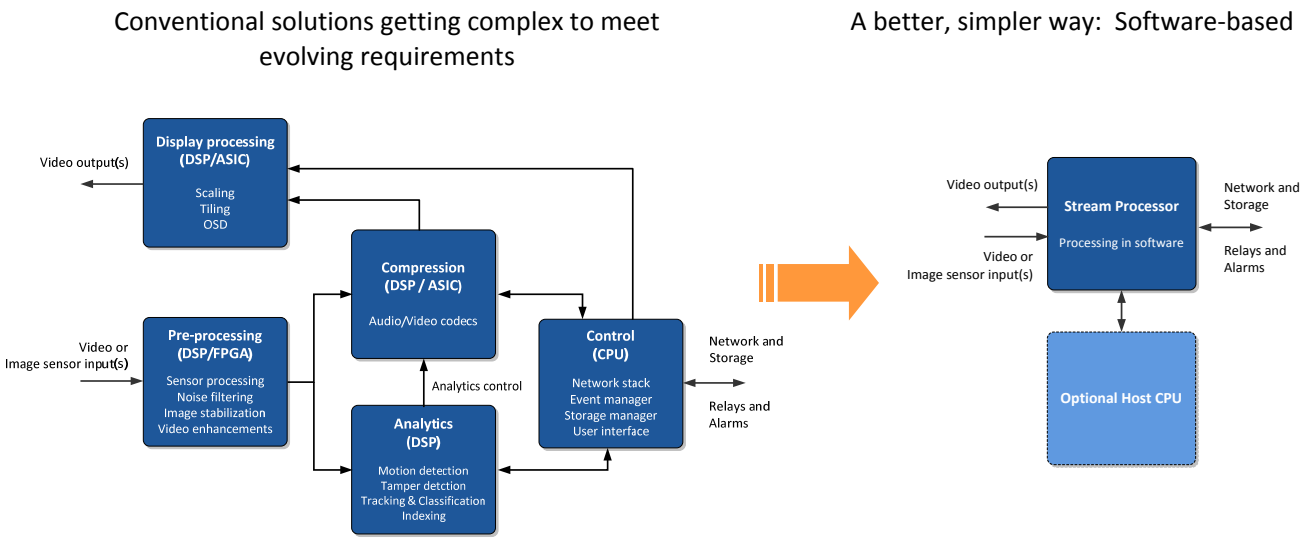

Figure 8. Moving from heterogeneous hardware-oriented to a simpler software-based model

9.2 Sample Application 2: Video conferencing

Manufacturers of video conferencing equipment have traditionally used DSPs since much of the secret sauce is in the encoder and rate control mechanisms. With the introduction of high-definition, H.264, and an emerging array of advanced techniques to deliver an in-person experience, processing requirements are skyrocketing.

Examples of processing tasks for video conferencing include:

- increasing resolution and framerate, e.g., 1920x1080p60 or 1280x720p120
- content-adaptive compression, e.g., spending more bits on faces, gestures
- emerging codecs, e.g., H.264 SVC (Scalable Video Coding) and H.264 Fast Profile
- background subtraction and modeling, motion compensated temporal filtering
- virtual camera/eye viewpoints and headsize compensation for immersive camera views
- 3D video
- beam-forming and immersive audio
- advanced rate control mechanisms
- I/O processing, e.g., alphablending, high-quality scaling in any aspect ratio, and presentation transitions

There are also specific requirements for this market, such as H.241 MTU packetization, low end-to-end latency, error resiliency, and legacy interoperability (e.g., H.263++).

A common approach to date has been to use multiple traditional DSPs and partition the tasks and/or the data across, or combine with FPGAs. Storm-1 represents a simpler approach. With the unique capability to encode a full HD 1080p30 video stream in H.264 down to 1 Mbps, and raw computational performance equivalent to more than ten traditional DSPs, next generation video conferencing products can be realized at significantly lower costs while maintaining improved scalability for 720p120 and beyond.

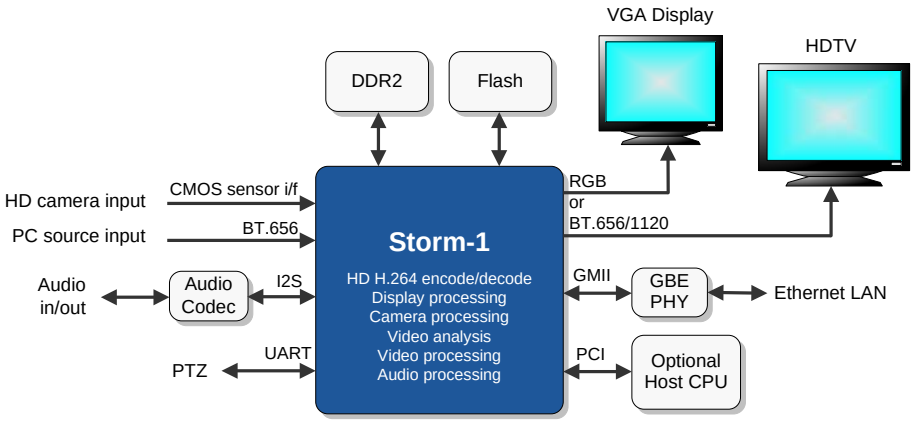

Figure 9. A single-chip HD H.264 encode/decode video conferencing solution

9.3 Sample Application 3: Multi-function printers

Multi-function printers (MFPs) need to perform demanding real-time image processing and provide the image path for print and scan/copy, both of which remain an area of active innovation. To date, ASICs have been the traditional solution for much of this processing, but with silicon technology heading toward deep sub-micron to keep up with requirements, NRE costs are getting out of hand. With the trend to focus on core competencies, there is also an increasing emphasis on algorithm and software development over hardware design.

Examples of tasks in a simplified contone image scan path suitable for Storm-1:

- input correction, e.g., deskewing, gain/offset
- content analysis, e.g., anti-counterfeit detection, OCR
- advanced halftone descreening / moiré pattern removal, e.g. various fixed and adaptive filters
- analyze and tag, e.g., text/graphics/image layering, blob detection/labeling/segmentation
- RGB to CMYK color space conversion, e.g., via 3D trilinear LUT
- scale, subsample
- correction, e.g., speck removal, punched hole removal, tear removal, background detect/remove
- compression, e.g., JPEG, G4, JPEG2000

The benefits offered by SPI include reduced development costs, reduced risk, improved TTM, improved product quality, and longer platform life, this at equivalent unit manufacturing costs to ASICs.

The performance level of Storm-1 enable a high-quality, high-compression MRC scan-to-PDF pipeline at over 80 pages per minute on a single chip. SPI offers an MFP software library which include print renderer and MRC scan pipeline.

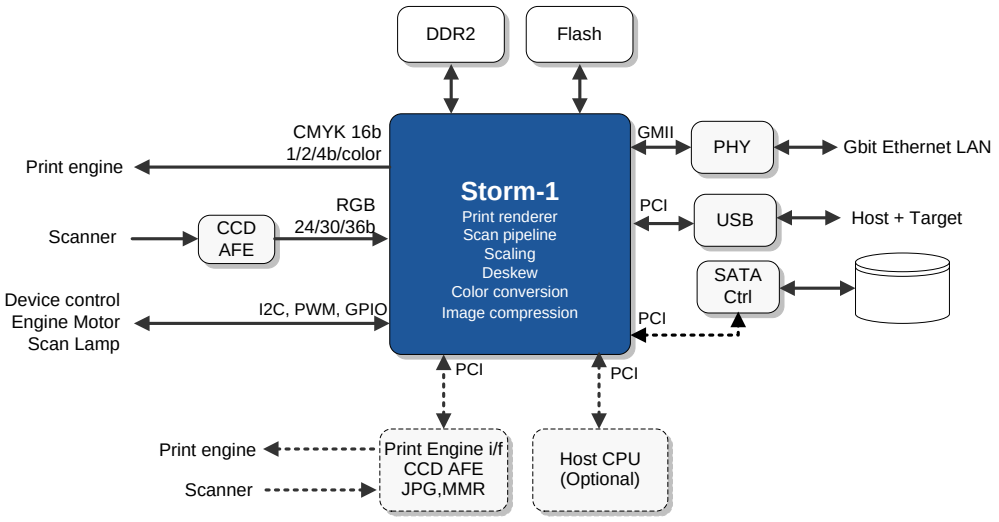

Figure 10. A single-chip MFP print/scan implementation

10 References

[1] Brucek Khailany, Ted Williams, Jim Lin, Eileen Long, Mark Rygh, DeForest Tovey, and William J. Dally, “A Programmable 512 GOPS Stream Processor for Signal, Image, and Video Processing”
International Solid State Circuits Conference (ISSCC), February 2007

[2] Jung Ho Ahn, William J. Dally, Brucek Khailany, Ujval J. Kapasi, Abhishek Das, “Evaluating the Imagine Stream Architecture”
Proceedings of the 31st Annual International Symposium on Computer Architecture, Munich, Germany, June 2004

[3] William J. Dally, Ujval J. Kapasi, Brucek Khailany, et al, “Stream Processors: Programmability with Efficiency“
ACM Queue, Vol. 2, No. 1, March 2004

[4] Ujval J. Kapasi, Scott Rixner, William J. Dally, Brucek Khailany, et al,”Programmable Stream Processors”
IEEE Computer, August 2003.

[5] Brucek Khailany, “The VLSI Implementation and Evaluation of Area- and Energy-Efficient Streaming Media Processors”
Ph.D. Thesis, Dept. of Electrical Engineering, Stanford University.

[6] Brucek Khailany, William J. Dally, Scott Rixner, et al, "Exploring the VLSI Scalability of Stream Processors”
Ninth Symposium on High Performance Computer Architecture, February 8-12, 2003, Anaheim, CA.

[7] Ujval J. Kapasi, William J. Dally, Scott Rixner, et al, “The Imagine Stream Processor”
IEEE International Conference on Computer Design, September 16-18, 2002, Freiburg, Germany

[8] John D. Owens, Scott Rixner, et al, “Media Processing Applications on the Imagine Stream Processor”
IEEE International Conference on Computer Design, September 16-18, 2002, Freiburg, Germany

[9] Ujval J. Kapasi, Peter Mattson, William J. Dally, et al, “Stream Scheduling”
Concurrent VLSI Architecture Technical Report 122, March 2002.

[10] Brucek Khailany, William J. Dally, Scott Rixner, Ujval J. Kapasi, Peter Mattson, Jinyung Namkoong, John D. Owens, Brian Towles, and Andrew Chang
IEEE Micro, March/April 2001, pp. 35-46.

[11] Peter Mattson, “A Programming System for the Imagine Media Processor”
Ph.D. Thesis, Dept. of Electrical Engineering, Stanford University.