Case Study: Digital Image Processing Using ARM Platform

The Challenge

Determine the best hardware/software architecture for a digital image-processing system-on-chip (SoC). Potential end-user applications include digital cameras and facility security systems. The objective was a simple, low cost, device that allowed users to perform composition tasks on an input video stream and capture the composed frame to JPEG encoded storage.

System Requirements

The goal was to reduce cost and power consumption while maintaining viewfinder display quality with a refresh rate of 30 Frames Per Second (FPS) during JPEG encoding of a frame. Reducing the number of gates needed to implement the functionality minimized cost, while reducing the system clock rate helped minimize power consumption.

The SoC included an ARM9™ processor core and an on-chip bus, as well as memory and other necessary peripherals. Although a hardware implementation of the JPEG CODEC was within the project scope, the cost and power constraints indicated running as much functionality as was feasible on the existing processor.

Approach

Foresight's Resource Aware Modeling and Simulation (RAMS) environment was used to model and verify the behavior of the system and perform tradeoff analysis of implementation alternatives for the JPEG encoder. Three alternatives were evaluated:

- An all software implementation of the JPEG encoder, executing on the ARM9™ core

- An all hardware implementation of the JPEG encoder, implemented in custom silicon

- A mixed hardware/software implementation of the JPEG encoder, where the computationally intensive "DCT" and "Quantize" functions were implemented in hardware, and the Huffman encoder was implemented in software.

Behavioral Model

The top-level data flow diagram (DFD) of the behavioral model is included below. It shows the image processing pipeline from raw video input to JPEG encoder. The definition of the JPEG encoder block is also presented.
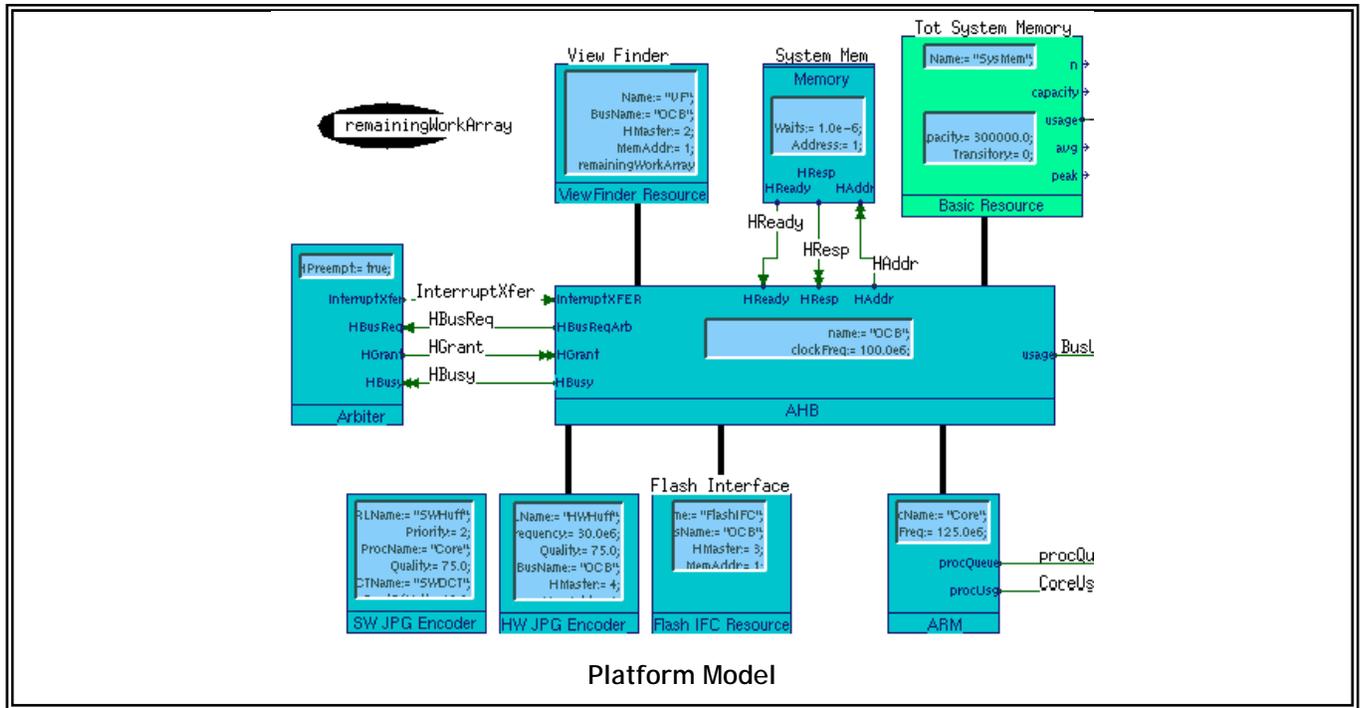

Behavioral Model

Platform Model
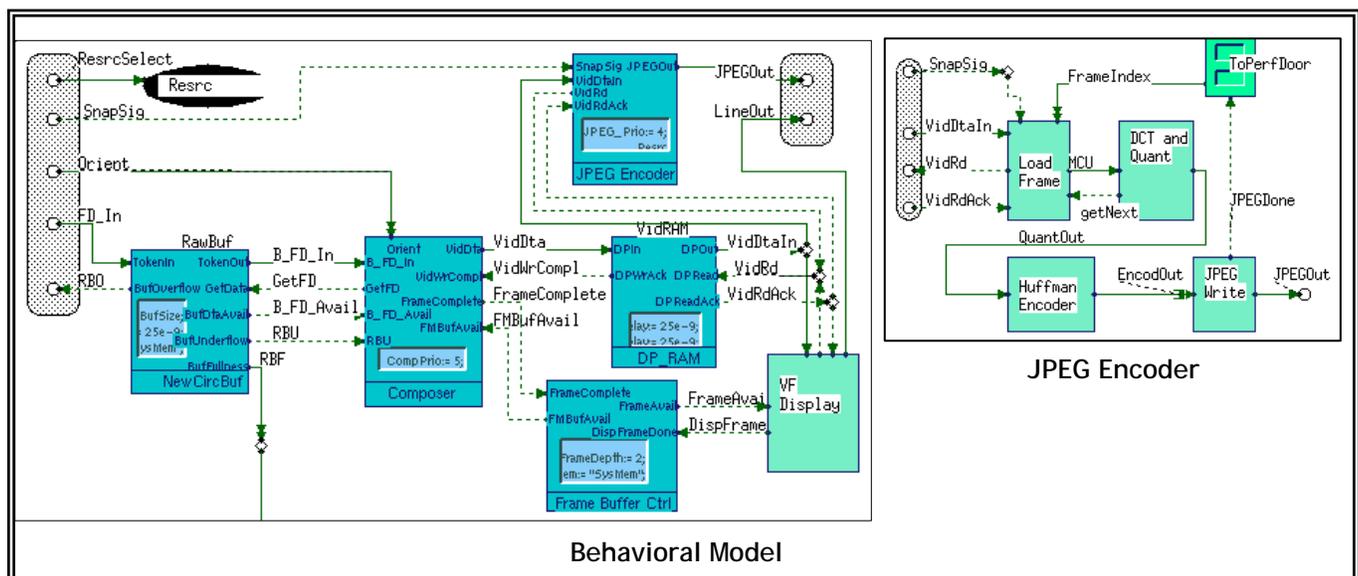
The top level platform model DFD is shown below. The performance of a variety of hardware /software trade-offs were evaluated by simulating the system with this platform.

Platform Model

Composite Model

A composite model is formed when a behavioral model is mapped to a platform model and the system is simulated. For this case, the Composer component was mapped to software running on the ARM™ core in all configurations. The "DCT & Quant" and Huffman Encoder components in the JPEG encoder were evaluated as either software running on the ARM™ core, or ASIC implementations.

A simple simulation control dashboard was created using Altia Design™[1] to aid visualization of the user interface and simulation results. This dashboard is shown in the appendix at the end of this paper. It contains controls to switch among the three tradeoff alternatives, a push button to initiate JPEG encoding and real-time displays for frame rate, JPEG encoding progress, bus utilization, ARM™ core utilization, number of simultaneous software threads, input buffer utilization with overflow indicator and volatile memory utilization. (Note that Foresight has similar I/O widgets that serve the same purposes. However, for this case, the team wanted to explore the UI prototyping capabilities of Altia Design™.)

Analysis Results

For this series of trade-offs, a 120 Mhz ARM9™ core processor and 100 Mhz[2] Advanced High-performance Bus (AHB) were used. Regardless of the implementation of the JPEG encoder, simulation showed that the image-

---

[1] Altia Design™ is a powerful user interface prototyping application produced by <u>Altia, Inc.</u>

[2] Clock speed values are configurable as "parameters" to the processor and bus model components.

processing pipeline was able to sustain a refresh rate of 35 Frames per Second (FPS) when simultaneous JPEG encoding was NOT occurring.

JPEG Encoder Implemented in Software

As shown below in Dashboard 1, with the JPEG Encoder implemented entirely in software, the sustained refresh rate dropped to 27 FPS (below the 30 FPS requirement); this delay was present for 0.08 second, or a period of about 3 frames. The load on the processor was 100% throughout the JPEG encoding process. The peak number of software threads on the CPU was 3 concurrent threads. Both the composer and the JPEG encoder were slowed by contention for processing bandwidth. As a result, the input buffer overflowed, causing data loss.

JPEG Encoder Implemented in Hardware

As presented below in Dashboard 2, with the JPEG Encoder implemented in hardware, the sustained refresh rate during encoding remained at 35 FPS and the process only took 0.05 seconds. The processor load was unaffected because dedicated hardware did the work. While the CPU load did not increase, the bus was more congested during encoding due to the two masters (ARM™ processor and hardware JPEG CODEC) contending for bus cycles. Fortunately, this bus congestion was not a problem because both frame rate and encoding times met the design requirements. The hardware implementation required approximately 16k additional gates (optimized against an ASIC library) plus memories.

JPEG Encoder Implemented in a Combination of Hardware and Software

Dashboard 3 shows the results when the DCT and Quantization operations were implemented in hardware, with the other operations running as software on the CPU. In this configuration, the system was capable of sustaining a refresh rate of 35 FPS while encoding. The peak CPU load was only 2 concurrent threads. During encoding, the input buffer contained data, but it did not overflow. The hardware implementation in this case required roughly 8k additional gates + memory. The memory requirements (used for matrix multiply) were about the same as in the hardware-only scenario.

Summary

While implementing the JPEG encoder entirely in hardware provided the best performance, this scenario also represented the highest cost for silicon and power consumption. A combination of hardware and software implementations met system performance requirements with less silicon, resulting in a lower cost. An "all software" solution did not meet the system requirements and therefore could be ruled out immediately.

This example focuses on only one of many analyses that could have been performed using this same model. Other trade-offs could also have been explored, including scheduling algorithms, bus protocols, etc., to further improve the design.

The RAMS methodology with the Foresight system modeling toolset enables rapid exploration of the design space with accurate, measurement-driven evaluation of each alternative configuration. This actual data based approach takes the guesswork out of effective systems optimization and design verification eliminating costly iterations and getting your product to market on time with the right features and performance at the lowest cost!

These, and many other, models are available for download from the Foresight support site at:

    http://support.foresightsystems-mands.com/SoftwareDownloads/ExampleModels/Foresight%20Demo%20Models.htm

For access to the Foresight Support web site, please contact: fs_support@foresight-mands.com

Foresight Systems M&S

Appendix

Dashboard 1: All Software

Dashboard 2: All Hardware

Dashboard 3: Mixed HW/SW