SMART SPEED TECHNOLOGY™

Results of Modeling for Embedded Applications

Mike Olivarez and Brian Beasley
Freescale Semiconductor

Abstract: This document presents the results of using the SpecC Methodology for creating architectures offered by Freescale for the mobile device space, implementing Freescale's Smart Speed Technology. The solutions incorporate a generic processor architecture for code compatibility, but requires additional peripherals to enable the full system level requirements of the solution. A summary of how the methodology has been used to enable the system to contain the needed capabilities shows that a higher level of confidence can be introduced into a complex embedded system at higher levels of abstraction. The overview will give a short explanation of the SpecC Methodology and some examples of how the methodology was used to verify the system for meeting the desired needs for performance, cost, and market requirements, benefiting both the engineering and business aspects of creating a solution.

Key words: Modeling, SpecC, Methodology, Smart Speed, MXC, i.MX, Specification, Architecture, Communication, Energy.

1. INTRODUCTION

The trend toward the use of high level languages such as C/C++ [1, 6, 9] as a basis for the next generation modeling tools and platform methodology to encompass design reuse has come to be a great mechanism. As there is no commercially available proven tool suite to fully implement C to silicon, the key has been to implement the methodology using means available at the current time, in the hope that the tools industry will catch up soon. The Methodology chosen, independent of any tools or language, was the SpecC Methodology. It is a well documented method of going from the C Model of

Olivarez, M., Beasley, B, 2007, in IFIP International Federation for Information Processing, Volume 231, Embedded System Design: Topics, Techniques and Trends, eds. A. Rettberg, Zanella, M., Dömer, R., Gerstlauer, A., Rammig, F., (Boston: Springer), pp. 231–240.

the problems, to creating the hardware and software needed to bring the solution to market.

As the problem is much larger and there is a need to avoid proprietary information, only an overview of how the methodology was used to create the architecture will be shown in this document. The document will cover the SpecC Methodology at the high level, and break down some of the pieces to show how the models helped with the implementation of Freescale's wireless and mobile processors, and how finding solutions early in the process allowed for trade-offs to be made based on performance versus cost versus market needs.

This document is organized as follows: In section 2, the SpecC Methodology is explained at a high level. In sections 3, 4, and 5, brief overviews of the different model types and how they impacted the system are explained. Section 6 gives the overall benefit of using the methodology to get products to market. Conclusions are provided in section 7.

2. THE SPECC METHODOLOGY

The SpecC Methodology has uniquely defined models for Specification, Architecture, Communication and Implementation [1, 2, 7]. These models allow the system designer to specify abstracted capabilities based on a C program of what the system needs to accomplish. Progressing through the various models reduces the amount of abstraction until one gets to the implementation details. The Specification, Architecture, and Communications models break the system design tasks down, to be more manageable, yet work toward the common goal. As the system's requirements and definitions increase in complexity, it is much more important to split into parts the tasks using a method that allows them to be divided amongst a team of people to work on, across disciplines and geographies. As the definition of an embedded system becomes blurred from the Application Specific Integrated Circuit (ASIC), which performs a single task, to the fully Asymmetrical Parallel Processing environment such as Freescale's MXC (MX300-30 and MXC275-30 for the sake of this document) [9] line of cellular processors including communications and applications processing, the ability to work more efficiently among various teams is critical to the success of the program.

Utilizing the well defined models allows for different members to work independently on various tasks, yet bring the solution together successfully. The methodology's well defined steps for simplicity allows it to function at the various levels, whether it be at the specification and design of the consumer device and how the various components need to be integrated; or

at the chip level, and how the hardware and software needs to interact and what blocks need to be defined; or at the definition of a specific block and the capability it must contain. Using this top down approach for system level design in a systematic fashion allows for the smooth interaction of the data and helps to create the prioritization of where resources need to be assigned to create a solution. The problems that are not on the critical path can be given a lower prioritization and resources can be quickly identified and aligned using the methodology.

3. SPECIFICATION MODEL

Early definition of the system specification based on requirements and use cases is critical for ensuring the system will meet the needs. As the definition of embedded system is blurred, it becomes increasingly difficult to meet the needs, especially if the system will need to meet future requirements as well. To adjust for this complexity, it is a challenge to understand if designing in additional capabilities will have the effect of being a "value add" to the system, or if it will simply be cost prohibitive to the system. Using the Specification Model and well defined use cases, allows one to make the cost benefit analysis of the system, based on how well it meets the use cases.

The example to be used is the inclusion of the floating point co-processor to the ARM1136J-S™ processor. This allows generic programs to take advantage of floating point capabilities, without having to emulate them in software, thus dramatically improving the performance of algorithms used by applications for audio, graphics and physics, to name a few. At the system level definition stage, it was not known if floating point was a hard requirement, thus, a cost analysis method was needed to determine if the benefit would make the solution cost prohibitive. Using the Specification Model of the SpecC Methodology allowed for the cost analysis to be performed. The Specification Model is a high-level model of the functionality that can test the algorithmic differences, without knowing the implementation details. Timing information can be ignored while cycle approximate capabilities are used to determine the overall difference. This higher level of abstraction allows for more simulations to be run quickly, to enable the system designer to have ample data to make the trade-off analysis. An additional benefit was the ability to estimate performance capabilities of adding parallelism into the system based on the defined use cases, without incurring a large computational increase for running the models. Since the Intellectual Property (IP) for this case is provided by ARM, as well as

simulation tools, the ARM Development Suite (ADS) was used for the basis of these tests.

The setup of the model utilized the Color Space Conversion application used for video. It was broken into behaviors with the top being the application, a second being the I/O, and the other being the floating point processing. There were multiple benefits of using this as a test case. One is that the mathematical algorithm is widely available [8]. A second benefit is that the fixed point code for the function that had been previously verified is also available, so the trade-off between hardware implementation and software implementation can be easily measured. A third benefit is that since the code is small and primarily floating point, the results can easily be multiplied by the percentage of floating point code in a use case to determine a benefit for other use cases by knowing the tendency, without having to fully test other use cases. For instance, since the geometry processing for 3D graphics would also be primarily floating point calculations, it will mimic the benefit shown by this smaller test.

An additional benefit of this kind of test that is an addition to the SpecC Methodology, is that power trade-offs can also be estimated at this point. The power can be estimated using Equation 1 (shown in Figure 1), when the IP blocks to be used are known or can be reasonably estimated. If the needed additional IP has not been specified, the maximum gate count and power specifications can also be determined using this method. The result is the solution can be determined based on the: 1) cost of the IP blocks and rough silicon area needed; 2) performance trade-off of the system as determined from the hardware and software; 3) power utilization of the system and impact on battery life.

From Table 1, we see that the additional hardware for performing floating point calculations, equates to an 825% performance improvement overall, greatly enhancing the floating point performance. Equation 1 shows the energy savings based on the cycles saved (X), and the amount of logic that is added (Y). The energy savings is independent of the technology used and additional techniques within the implementation that may save even more energy, resulting in longer battery life. From all of these factors, one can determine if the additional hardware provides a benefit that would be a "value add" to the overall system, based on the cost of adding the additional capability. Also shown is an alternative fixed point solution, with the results in Table 1. If it is determined that the "value add" is not of a high enough benefit, optimizations in the software can help accomplish a similar benefit, though for floating point intensive applications and dynamic range, the floating point co-processor brings the best benefit.

Of course in business, everyone has an obligation to control cost, and these techniques allow the system designer to do his/her part in the analysis.

The cost for the hardware and software versions can be calculated, based on the size of the code and the size of the additional hardware for the different proposed solutions. At this time, any licensing fees can also be added in and amortized over the project based on the number of units to be sold to obtain the per unit cost. From this, the project can move forward and tasks distributed across resources, or it may be determined that a scrub of the "specification" may need to occur. As this is an iterative process and may benefit multiple projects, the use of the Specification Model and the ease as to which it can be changed, makes this process much easier and more efficient over previous ad hoc methods.

Table 1. Summary of improvements over Software Floating Point.

Improvements over Floating Point Software	Floating Point Hardware Improvement Percentage	Fixed Point Software Improvement Percentage
Application cycles	825%	837%
Floating point calculation cycles	1525%	793%[1]
Instructions issued for floating point calculations	2223%	923%[1]
Application energy savings	83%	88%
Floating point calculation energy savings	91%	88%[1]
Code memory footprint savings	16%	16%
Arithmetic code development time	15 min.	~120 min.

$$PercentEnergySavings = (1 - (\frac{1}{X} * (1 + Y))) * 100\%$$

Figure 1. Power savings, based on X as the cycle reduction multiplier, and Y as the additional logic adder.

4. ARCHITECTURE MODEL

Once the specification has been created, and the behaviors required are being met, the next step is to decide the best way to implement. Behaviors such as adding a floating point unit or not are still a high-level trade-off, but they're made easy if the decision is whether or not to add the capability. The architecture model goes into what is the most efficient way to implement the behaviors that have been defined. Again, the analysis can include not only the engineering aspects of architectural exploration, but also the cost, die size, and time to market aspects.

[1] Results are not just isolated arithmetic, but also contain I/O requirements for use of the data structures, as opposed to the calculations used for floating-point arithmetic.

Within the architectural exploration process of refining the specification, trade-offs of the System on-chip (SoC) IP blocks can be measured for the performance impact of hardware versus software for implementing the behavior, as well as how the various blocks are connected within the system. Various IP blocks that implement the behavior can be tested, as well as various hardware/software partitions. In many cases, reuse is most important for backward compatibility and time to market, and specific IP will be taken off the shelf. In other cases, new IP can be introduced with different hardware and software trade-offs. In either case, modeling the system will ensure the behaviors are functioning such that the system specification is met [3]. Performance of the behavior can still be quite different in how the SoC is architected, even with the same IP blocks being used. The trade-off analysis to be used for the example in this section is the placement of the level 2 (L2) cache in the system.

Architectural exploration was performed to see if the L2 cache subsystem should be dedicated to the ARM processor, or if it should be shared across multiple processing elements (PE) within the system. The characteristics for the memory subsystem were set up, based on the times required for data to be obtained from the level 1 (L1, instruction and data, I-cache and D-cache respectively) cache of the CPU and level 2 cache (L2) accesses. The statistical model used 30% of the instructions as memory access (10% stores, 20% loads overall). The I-cache was set to a 98% hit rate, and the D-cache was set to a 95% hit rate, while the L2 utilized a 50% hit rate and other blocks were given statistical access rates to complete the model. The memory arbitration scheme was done using priority arbitration, where the priority was set, based on the processing element number, with the CPU as 1 and the others sequentially added up to 5 elements in the model.

The simulation looked at the cycles per instruction (CPI) the ARM processor should be able to achieve, as well as the theoretical CPI (tCPI) of the SoC, based on each of the PEs having different definitions of instructions. Since the two architectures are utilizing the same types of PEs, the measurement is a good comparison of the two architectures. If the two systems will be utilizing different PEs, the effective CPI (eCPI) could be calculated based on simulations of test cases as defined in [4]. Due to the difference in the speed of the CPU and its higher memory bandwidth needs compared to the other PEs used in the test, having the L2 dedicated to the CPU gave the best performance, as shown by the lower CPI numbers, for both the CPU and the SoC tCPI[2]. The test was also run with a faster memory interface, as could be accomplished using double data rate memory and shared L2, which improved on the numbers, but still wasn't as efficient as

[2] tCPI based on statistical simulations and is an approximate measure of what the architecture may achieve. eCPI is based on measured test cases using actual applications.

what was accomplished using the architecture when the L2 cache is dedicated to the ARM processor.

Using the Architecture Model, it was shown that the L2 cache was a greater benefit to the much faster processing element. A secondary result was the number of external memory access was reduced by allowing all the PEs to utilize the on-chip memory. Lowering the overall number of off-chip memory access could be an energy savings, but the difference was not shown to be great enough to warrant the decrease in performance for this case. In a separate case where the dominate PE was not significantly faster than the other PEs, it was shown that sharing the on-chip cache memory was more beneficial for the overall performance and lowered the number of off-chip memory accesses, thus conserving precious system energy. Even though the PEs may not change, the subtle changes in the behaviors defined through architecture exploration can have a significant impact on the overall capabilities of the system.

The Architecture Model allows one to quickly make trade-offs between different configurations, implementing the behaviors defined by the Specification Model. Quickly changing between scheduling and priorities allows the system designer to make architectural decisions, while abstracting out information such as the communication protocol and overall bus structure. This method can be used hierarchically so that the problem is more manageable and can be split between different architects and teams.

5. COMMUNICATIONS MODEL

The Communications Model allows less abstracted system timing to be introduced to get a true performance estimation of the behaviors. The Specification and Architecture models can only use a rough estimation of the timing, and didn't include any overhead that the bus structures would introduce. When including the various IP blocks of the system, one must also take into account the interface portion, as the various IP vendors may not use the same bus interface for all the blocks. The Communications Model enables this to be taken into account.

As the various PEs are added to the system, the arbitration schemes and bus structures have a large performance impact, as well as cost and time to market implications. The easiest method is to connect every processing element (including the memories) to a single bus. The bus will need to arbitrate between all the elements, and will only be as fast as the slowest element. The best performance would have to be a mesh network of busses, so that every processing element has a point-to-point connection with every

other element at the rate of the fastest element. This latter method is too expensive, and wastes much of the resources added to the system.

In the MXC and i.MX products, various bus structures are utilized to maximize the performance and cost. For the high performance processing elements and the memory interface, a Smart Speed Switch has been introduced, to allow the faster processing elements to communicate on a point-to-point basis, with fast access to memory. These processing elements have a standardized bus interface definition, to ensure ease of connectivity between standard IP blocks with the highest performance. The slower peripherals use a slower, more cost-effective bus and are interfaced to the shared resource. Such a method allows for the slower devices to be efficiently integrated into the system, without impacting the performance of the more timing- and bandwidth-critical components.

Since the two domains still need to be able to share resources such as memory, and allow inter-process communication, transducers were added to the system to close the gaps. The two primary standards used in this case are ARM's AMBA AHB bus protocol for the high speed communications, and the Freescale Intellectual Property Interface (IPI) structures for the less performance-critical elements.[3] Bridges have been created between the different bus architectures, and modeled for performance using an internal modeling tool based on SystemC [6]. A result of the modeling was the creation of the Smart Speed Switch, which is an AHB-compliant crossbar that allows up to five master elements to talk to slave elements simultaneously, giving the relative ability to enable data throughput of a 665 MHz bus.

A key design parameter the system will need to meet for the mobile space is to efficiently complete use cases without wasting resources. The best method for lowering clock speed and saving precious battery life is to utilize parallelism in the system, and using dedicated processing elements, as was shown previously with the floating point example. Looking at the complex behaviors of the system, parallelism will need to exist not only by the various PEs working simultaneously, but also in the communications, such that the PEs are not starved. By using the Communications Model and adding in the requirements for the protocol and bridging between protocols, one can determine various bus configurations for the system. Through such modeling of the use cases and deciding on the PEs for the architecture, it was determined that there was sufficiently enough high bandwidth or low latency peripherals to warrant a switch which allows for up to five masters. Such PEs can be represented by the ARM, L2 Cache, Image Processing Unit,

[3] The Freescale IPI standard includes a range of bus structures for different speed peripherals. The slower speed bus is used, as it connects the non-speed critical processing elements for the system mentioned in this document.

Multi-channel DMA Unit, Memory Controller, High Speed I/O, Graphics Processor, and bridges to lower bandwidth/latency PEs, though all of these may not be a master. Complex use cases can be used to best define the communications requirements for such systems.

Of course, the parallelism doesn't come for free. There are costs of silicon space and energy consumption to be factored. Equation 1 can be used once again to estimate the costs over a previously utilized solution. The ability to quickly configure and change the model allows for the communications structures to be changed to meet the desired cost targets, before getting too deep into implementation details.

6. CREATING SMART SPEED TECHNOLOGY

Smart Speed Technology is the brand that defines a system solution, based on providing enough performance for the use cases, without wasting resources. Unlike the personal computing space, where processors are run as fast as possible, even if the use case defines an idle state, thus wasting resources, processors for the mobile space must be able to conserve as much resources as possible, without degrading the user experience. Keys to accomplish this goal are to begin with well-defined use cases that create the foundation of the solution, and to add some use cases that are good candidates for future requirements. From the use cases, the Specification Model can be built, tested and verified. As capabilities are added that could drain the precious limited energy resource of the battery, additional capabilities to limit power consumption can be added. In the case of MXC and i.MX processors, additional power saving capabilities such as Dynamic Voltage and Frequency Scaling and Dynamic Process Temperature Compensation have been added to conserve energy.

Benchmarking can be run on the final solution, to verify the methodology and improve upon it. Via the use of published data from Synchromesh Computing's benchmarking of the i.MX31 processor [10, 11], it can be seen that the modeling of the system made for accurate trade-off and specification analysis. Though the difference is 17% greater efficiency by the final solution over what was projected for floating point tests, it can be seen that the data is "Close enough for all practical purposes" [5] to make the decisions at a high level of abstraction. This affirms the use of the Specification Model as a tool for making high-level system trade-offs. Additional benchmarks based on Consumermark™ and STREAM benchmarks shows the architecture can efficiently perform the use cases they test, verifying the use of the L2 Cache and overall system communications. The benchmarking also proved the power savings the architecture was

estimated to obtain. Based on the benchmarks, it can be seen that the overall goals of the system for performance and cost and the overall price/ performance targets specified have been met.

7. CONCLUSION

It can be seen from the techniques shown, as well as in the benchmark results of the final product, that the modeling effort used to design the architecture for the MXC and i.MX products allowed for Smart Speed Technology to be highly leveraged. The solutions can meet the use cases computationally, yet still achieve low clock speeds and energy efficiency for the mobile system. This achievement can be attributed directly to the SpecC Methodology for performance with additional analysis for power and cost to make a solution that is successful in the market for mobile consumer devices.

REFERENCES

[1] D. Gajski, J. Zhu et al. “SpecC: Specification Language and Design Methodology” Kluwer Academic Publishers, 2000.
[2] www.ics.uci.edu/~specc
[3] L. Cai, M. Olivarez, D. Gajski, “Introduction of System Level Architecture Exploration Using the SpecC Methodology”, International Symposium on Systems and Circuits, May 2001.
[4] M. Olivarez, B. Beasley, “Use Effective Cycles/Instruction as a True CPU Benchmark”, Portable Design, May 2005.
[5] S. Nichols, “Rules for Engineering”, Various Lectures: University of Texas at Austin, 1987-.
[6] http://www.systemc.org
[7] A. Gerstlauer, R. Dömer, D. Gajski, “System Design: a Practical Guide of with SpecC”, Kluwer Academic Publishers 2001.
[8] V. Bhaskaran and K. Konstantinides, “Image and Video Compression Standards: Algorithms and Architectures, Second Edition”, Kluwer Academic Publishers, 1997.
[9] http://www.freescale.com/smartspeed
[10] http://www.eembc.com
[11] “The Freescale Semiconductor i.MX31 Processor: Cool, Powerful”, Synchromesh Computing, 2005.