Simulation and Verification of a Mixed-Signal Programmable System-on-a-Chip

Monte Mar and Bert Sullam

{monte.mar,bert}@cypressmicro.com
Cypress MicroSystems, Inc.
Bothell, WA 98072

Abstract

An overview of a simulation strategy for the verification of one of the first mixed-signal field-programmable system on a chip (FPSOC) is presented. The FPSOC integrates a microcontroller, FLASH memory, programmable digital blocks, and programmable analog blocks. A proprietary digital Verilog with analog extensions was used to verify system interactions between the analog and digital blocks. First-pass functional silicon was obtained because of the simulation methodology.

Introduction

Recent advances in programmable devices have provided a viable solution for rapid prototyping of complex systems. Time-to-market can be minimized and engineering changes can be made late in the design cycle. The main focus for programmable devices has been on devices such as FPGAs, CPLDs, and reconfigurable processors aimed at mostly digital systems. Discrete programmable analog arrays have also been released as products, albeit with limited acceptance [1][2][3]. For embedded systems, the typical application makes use of a small processor or controller which co-ordinates the execution and processing of data from peripheral devices. The integration of a microcontroller and programmable analog and digital blocks allows realization of single chip solutions for embedded systems, allowing for a compact, low-cost solution.

A mixed-signal programmable architecture provides inherent problems during design, simulation, and verification. During design, the definition and evaluation of the micro-architecture and the communication between blocks provide significant challenges [4]. Bad decisions at this stage can lead to problems in developing the software programming model. For the digital portions, digital design flow methods such as RTL synthesis and digital simulation methods provide fast and efficient methods for developing and verifying the micro-architecture. For the analog portion, ad-hoc techniques are available but no real methods and tools analogous to the digital flow are available. Thus simulation becomes important for early micro-architecture design, pre-silicon design verification and for later use in verifying designs. Additional simulation methods are needed to handle interactions between the analog and digital sections. For chip verification prior to tape-out, behavioral modeling is essential to ensuring proper interaction between analog and digital domains. Time-step driven analog simulation is computationally expensive, slowing down the overall system simulation. This means that some form of behavioral modeling is necessary. However, accuracy of the behavioral model is needed to properly verify system function. In this paper, the simulation approach to verifying the functionality of a mixed-signal Field Programmable System on a Chip (FPSOC) device will be presented. A discussion of the analog micro-architecture and simulation methodology is presented. In the next section, the strategy used for simulation and verification is presented. A few examples of the chip simulation and measured results are discussed to show the effectiveness and efficiency of the modeling approach.

Device Description

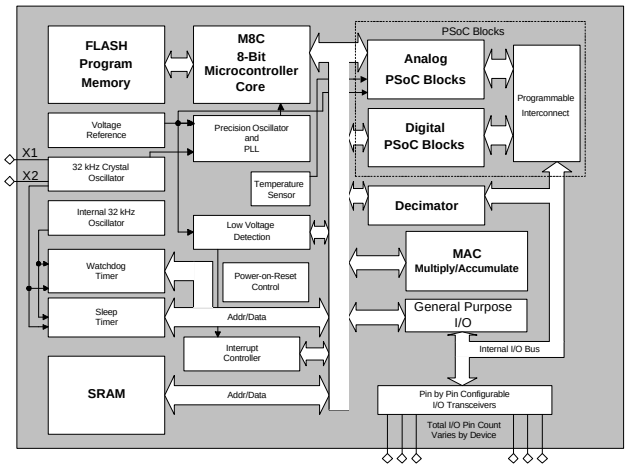

Figure 1. Block diagram of the FPSOC device.

Figure 1 shows a block diagram of the part.

The device contains an 8-bit microcontroller, flash memory, SRAM, digital programmable blocks, and analog programmable blocks. The analog array consists of programmable switched-capacitor blocks, continuous time amplifier blocks and programmable reference generators. The programmable interconnection between

these blocks enables them to be combined in ways to create more complex functions or to extend data resolution. Examples of these analog functions include DACs, various types of ADCs, programmable gain amplifiers, comparators and filters.

The digital programmable array consists of blocks that may be configured as variable precision timers, counters, pulse width modulators, UARTs, and CRC generators. The analog and digital blocks have a large range of clocking choices that can be harmonically related, but independent of the microcontroller clock. The analog and digital programmable blocks are controlled by register settings. Various functions are created by programming the values in the registers, which are mapped into the I/O address space of the microcontroller. Functions can be preconfigured during the initialization of the part or they can be dynamically reconfigured during operation.

Analog Micro-Architecture and Simulation Strategy

A. Analog Micro-Architecture

The definition of the analog micro-architecture is a key feature in the design of a mixed signal programmable processor. The level of abstraction used in the blocks characterizes the architecture. In FPGAs and CPLDs, the level of abstraction focuses on logic gate primitives and flip-flops or latches. Complex systems are built by programming the routing resources to combine gates and latches into complex functions built through hierarchy. In contrast, analog programmable arrays generally have a trade-off between abstraction and performance. Using programmable blocks at low levels of abstraction, for example differential pairs and current mirrors, provides great flexibility [1]. However, the necessary routing resources for combining the blocks exposes the circuit to extra loading and parasitic coupling to noise sources in other parts of the array. These limitations make it difficult to predict the bandwidth and noise performance of the resulting programmable circuit. There is value in the analog programmability, but it is difficult to achieve high performance.

More recent efforts at analog arrays have focused on higher levels of abstraction and more complex routing architectures [2]. Op-amps are used as the building blocks, and various passive elements are provided to allow higher-level functions such as filters and gain stages to be constructed. The higher level of abstraction enables better prediction of performance. It also limits the choices of topology for circuits, providing a much easier framework for constructing synthesis tools [5]. A recently reported device focuses the architecture towards the implementation of continuous time filters, providing OTAs and capacitor and routing resources [3]. By limiting both topology and the types of functions that can be implemented, extremely good performance (roughly 16 bit) can be achieved in spite of the programmable architecture.

In the FPSOC architecture, the emphasis is on general purpose analog signal processing. Since the target application is the 8-bit microcontroller market, a large number of analog function blocks are needed in the smallest possible area. The analog circuits need to support 8-10 bit resolutions. Higher resolutions are useful, but they require 2 byte operations within the micro to process. In addition, gain amplifier paths in the range of 50-100 need to be supported for sensor applications. This requirement means that either large capacitors need to be provided for switched-capacitor stages to minimize kT/C noise or some type of continuous time processing is needed for lower noise amplification.

Under these constraints, the array was architected with functional blocks defined at a fairly high level of abstraction. Only single-ended circuitry was used. Rather than providing a pool of op-amps and a pool of capacitor, switch, and resistor resources, the analog programmable block was defined as an op-amp and dedicated resources. Two types of blocks were created, one with an op-amp and resistors dedicated to continuous time signal processing and the other with an op-amp and capacitors for switched-capacitor signal processing. The interconnections are broken down into local connections within the block to configure the function and inter-block connections that govern how blocks are connected to create more complex functions.

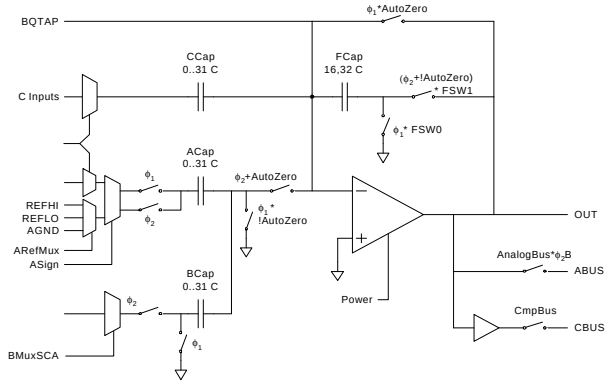

Figure 2. Block diagram of a SC block.

A basic diagram of the switched-capacitor block is shown in Figure 2. The block is built around a switched-capacitor integrator with 3 capacitor arrays for the input branch and an adjustable integrating capacitor. Functions

like a delta-sigma modulator, a gain amplifier, or summing amplifier can be implemented in one block. Blocks can be combined to implement biquad filters or other functions. More detailed descriptions of the blocks are found elsewhere [6].

Ideally, the programming model for the analog array would be similar to a VLIW architecture. A long control word allows more flexibility in determining how the resources can be connected within a block. Since the programming model has the control registers for the block in the microcontroller I/O address space, the control word was broken into 4 bytes.

B. Analog Array Simulation Strategy

Designing the array as a set of self-contained blocks provides advantages in simulation. For simulation of the FPSOC architecture, it is desirable to have a relatively fast simulation. The digital portion could be verified quickly using Verilog. A solution was needed that would allow analog models to be incorporated within a Verilog simulation. Time-step driven modes of Verilog were to be avoided due to the slower speed of execution that would limit the number of machine cycles that could be simulated.

A high-level model usually implies loss of simulation accuracy. However, previous work in the modeling of $\Delta-\Sigma$ modulators has shown that a fairly simple behavioral model can be used while still maintaining good accuracy [7]. The basic approach is to exploit the sampled time nature of two-phase switched capacitor circuits. The value at the output of a SC circuit is on valid on the falling edges of the two clock phases. The circuit can then be modeled in terms of finite difference equations. Furthermore, the difference equations can be implemented using an event-driven formulation provided that no zero delay loops are present in the circuit. Non-ideal effects can be incorporated in the difference equation model, and this can further improve the accuracy of the simulation. The initial implementation of the array simulation was performed using the synchronous data flow (SDF) domain of Ptolemy [8].

While the event-driven formulation works well for the SC circuits, an extension is needed for the continuous time blocks. It was noted that SC circuits typically have clock rates that are much higher than the bandwidth of the signals being processed, i.e. the signals are oversampled. Under this assumption, the continuous time signal processing is modeled as sampled data signals. The clock rates of the SC circuits set the sampling rate at which the continuous time blocks are modeled.

Figure 3 shows a primitive programmable SC block. Figure 4 shows an implementation of the event driven formulation for the signal processing. The formulation uses charge as signal stored, and this requires a slight modification from the classic node voltage analysis. The programmable interconnect and capacitor values are handled through extra lines in the code. Note that the output goes valid on the falling edge of the clock edges phi1 and phi2. Input sampling occurs on the falling edge, allowing changes to propagate through the blocks.

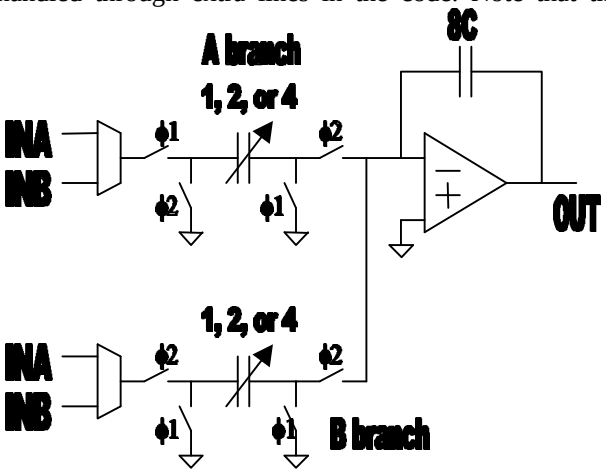

Figure 3. Block diagram of a simple SC block.

```
always @(negedge PHI1)
begin
        AGND = 2.5;
        ina = MA0*IN1A + MA1*IN1B - AGND;
        // Account for charge on branch A
        cina0 = CA0 * ina;
        cina1 = 2.0 * CA1 * ina;
        cina2 = 4.0 * CA2 * ina;
end

always @(negedge PHI2)
begin
        AGND = 2.5;
        ina = AGND;
        inb = MB0*IN2A + MB1*IN2B - AGND;
        deltaq = 0.0;
        // Integrate charge from branch
        deltaq = deltaq + cina0 - ina*CA0;
        deltaq = deltaq + cina1 - 2*ina*CA1;
        deltaq = deltaq + cina2 - 4*ina*CA2;
        cina0 = 0.0;
        cina1 = 0.0;
        cina2 = 0.0;
        // Handle the B branch
        deltaq = deltaq - CB0 * inb;
        deltaq = deltaq - CB1 * 2.0 * inb;
        deltaq = deltaq - CB2 * 4.0 * inb;
        oaout = deltaq/8.0 + AGND;
end
```

Figure 4. Verilog description of the simple SC block.

Digital Programmable Blocks

A. *Architectural Considerations*

The digital PSOC blocks are designed to offer an array of peripheral functions commonly used in micro-controller applications. The key difference between the PSOC architecture and previous micro-controller architectures is the configurability of these digital resources. Timers, counters, PWMs, and CRC generators are available in all digital PSOC blocks. A second type of digital block combines these functions and adds capabilities for UARTs and SPI communications. In this scheme the granularity of configuration is very coarse. A few register bits determine the function and mode of operation. The advantage of this approach is that the digital PSOC block can be optimized to support hardware structures that are common to this set of functions. In contrast, an FPGA implementation has a very fine level of granularity. It would provide these and many more potential functions, but at a much higher cost in terms of chip area and configuration complexity. The key to the PSOC approach is reflected in the fact that the target micro-controller market is extremely cost sensitive and chip area must be minimized.

The basic digital PSOC block is 8-bits wide and in the current implementation, there are 8 blocks organized as a linear array. Nearest neighbor blocks may be programmatically chained together to create functions of larger data widths. For example, the 8 available blocks can be 8 8-bit timers, or 4 16-bit timers, or 2 32-bit timers or any arbitrary combination of these and other available functions. The configuration is achieved through the use of a peripheral register programming interface in firmware. If a given application only requires a fixed set of digital peripheral functionality, the configuration programming can typically be done as part of an application's initialization sequence. Figure 5 illustrates some possible configurations of the digital PSOC blocks.

The digital PSOC blocks are designed to interface with the analog array to create higher level functions. For example, a switched capacitor analog block may be configured as an incremental A/D converter. In this configuration, the analog block drives the comparator output signal with a duty cycle proportional to the analog input voltage. A digital PSOC block, using the comparator output signal as a gate to a counter configuration can integrate the result of the duty cycle over the required period of time. An additional timer can be used set the overall conversion time, generating an interrupt when the integrated result is ready to read. Since the timers and counter can be made any multiple of 8-bits, there is a lot of support for different conversion rates and data resolution requirements.

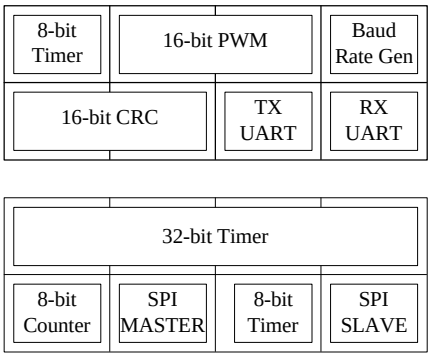

Figure 5 Example Digital PSOC configurations

These digital resources can be used even more efficiently through time multiplexing. During application runtime, a few register writes can change one or more block functions on the fly. To illustrate the potential of this approach, consider the example of a remote sensor application. Typically, the system may be in a low power state during a dormant period, which may be implemented with a 32-bit timer, configured from 4 digital PSOC blocks. When the timer reaches its terminal count, an interrupt can wake up the system. The firmware executing at this point can configure the analog blocks to support the required sensor signal chain and take the samples. The digital blocks may be used to support the analog acquisition, or set up as communications blocks send the result to a host. After the acquisition and transmission of the data, the previous 32-bit timer configuration can be reloaded, and the next wait period can be initiated.

Integrated System Simulation and Verification

A. *Problem Description*

As illustrated in the last section, analog blocks, digital blocks, and microcontroller firmware are combined in various ways to create the desired system level functionality. The range of functional blocks and the large number of possible connections creates complexity problems for system simulation. At the lowest level, interconnections between analog and digital blocks need to have the correct handshake for both signal value and timing. A second layer of complexity is the fact that a given function may be mapped to many different combinations of blocks. For example, a 10-bit DAC can be created from two switched-capacitor analog blocks, however, there may be twelve or more legal mappings of this DAC in a 2 x 4 array of analog PSOC blocks. Each of these different combinations requires a slightly different interconnection and feature interaction to verify. A final layer of complexity is created by the use of

dynamic reconfiguration.

B. Verification Strategy

To achieve the desired system verification within the constraints of the time and tool budget, a strategy was developed in which the entire chip, including the analog subsystems, could be simulated with Verilog. A standard Verilog netlist was generated for the digital portions of the chip, including the microcontroller and the digital PSOC blocks. The structural portions of the analog array, which implemented the programmability and interconnections, were also netlisted in Verilog. Hand crafted behavioral models were substituted for the leaf cell analog blocks, which implement the switch capacitor or continuous time functions.

The functionality of the analog behavioral models was verified against circuit simulation. The models were written so that when the values are sampled at a particular time in the clock cycle, the output of the behavioral model closely matches the silicon. However, rise times and behavior outside the sampled region may not be accurate. This abstraction was sufficiently accurate for verifying communication between the analog blocks and between analog and digital blocks.

A minor enhancement was added to a proprietary Verilog simulator in order to seamlessly support this methodology. Real valued variables were used to compute and pass signal voltages in the analog behavioral models. This allowed real valued signals to be passed from module to module through the hierarchy without any special declarations. A standard Verilog wire construct would be sufficient. This also allowed the same netlist, (except for the leaf cell models) to be used in all phases of the design (simulation, layout, lvs).

The fast simulation time achieved in this modeling scheme facilitated the functional validation of complex systems, providing close matching to execution in silicon and capture of any conceptual errors in the design. In addition, more combinations of function mappings could be simulated. The integrity of the structural portion of the analog subsystems, such as array interconnections, could be verified as efficiently as that of a digital simulation.

C. Mixed Signal Functional Verification Example: SAR

Analog to digital converters are commonly used in microcontroller applications. The FPSOC architecture offers a number of types and combinations of A/D converters, created through a combination of analog blocks, digital blocks and/or firmware support. One such configuration available in the FPSOC architecture is the successive approximation (SAR) A/D. The SAR A/D converter requires the following building blocks: A digital to analog converter (DAC), a comparator, and a method to sequence successive writes to the DAC based on the comparator output. In its simplest description the SAR algorithm is a binary search on the DAC code that best matches the input voltage.

Depending on the resolution required, a DAC can be created from one or more switched capacitor analog blocks. For simplicity, one analog block will be considered, with a resulting data resolution of 6-bits. Another analog block was allocated to function as a comparator.

The binary code for the DAC can be written or read through the microcontroller register interface. Similarly, the result of the comparison may be accessed by the microcontroller through a bit in a read-only register. With this capability, the SAR algorithm sequencing can be implemented easily in firmware. Figure 6 shows a block diagram of this function.

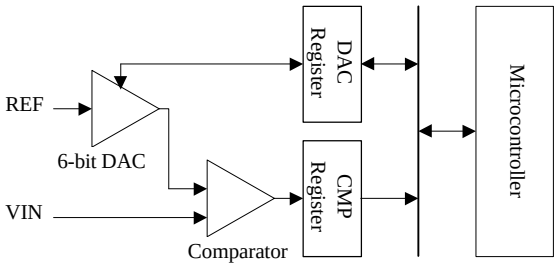

Figure 6. High Level Block Diagram of SAR6 A/D

The binary code for the 6-bit DAC is in signed magnitude format. A code of all zeros corresponds to analog ground, which is nominally mid supply. The references were selected to be +/- a bandgap around analog ground. With a nominal bandgap value of 1.25C, our total DAC range is 1.25V to 3.75V, which corresponds to DAC binary values of 111111 to 011111, respectively. The SAR firmware algorithm first determines the sign of the input voltage and then sets the sign of the DAC voltage to the opposite polarity. When the algorithm is complete, the DAC outputs the same voltage, but opposite polarity so that the sum at the comparator is as close to analog ground as the resolution will allow. With 64 levels across 2.5V, the resolution of this DAC is 39mV/bit.

This SAR6 simulation was run on a Verilog netlist of the full chip, with analog enhancements. The code was assembled and loaded into the flash memory model. After power-on reset and a boot sequence, the main code

configures the PSOC device and then a loop runs continuous conversions. The input voltage is applied to a chip pin, initially set in behavioral code to a fixed value. The firmware consists of about 30 lines of assembly code to program the PSOC device, then about another 30 lines of code to run the SAR conversion algorithm.

A 3.0V input voltage is +0.5V relative to analog ground, therefore the expected result of the DAC output after the SAR algorithm is 2.0Vor –0.5V relative to analog ground. Below is a table of simulation results for an input value of 3.0V.

Hex Code	Binary Code	DAC Voltage	Comp	Decision
20	100000	2.5	0	Keep Sign
30	110000	1.875	1	Clear
28	101000	2.1875	0	Keep
2C	101100	2.03125	0	Keep
2E	101110	1.95325	1	Clear
2D	1.01101	1.99218	1	Done!

The result of the behavioral simulation shows that the expected code is 2D, which in signed magnitude format is –13 or in terms of voltage, (13*39mV/lsb) = -507mV relative to analog ground.

Figure 7 shows the current value of the DAC output, DAC register value, and the comparator value logged as waveforms from the Verilog simulation.

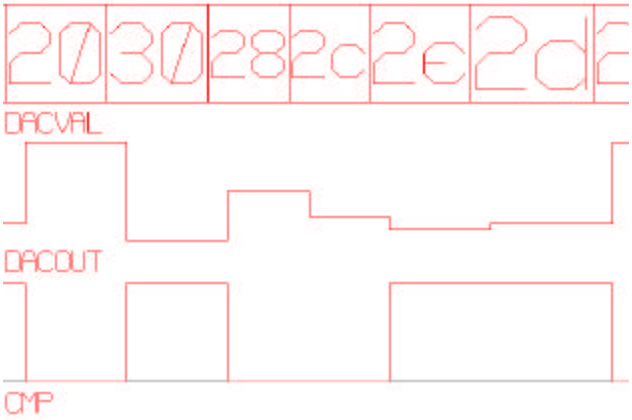

Figure 7. Simulation waveforms of one SAR conversion

Figure 8 shows the results running the identical set up and firmware on the target silicon. The highest voltage at the beginning of the waveform corresponds the start of conversion and to analog ground. The binary search from 2.5V to 2.0V is clearly evident. The intermediate steps and final value of –508mV relative to analog ground corresponds quite closely to simulation results. One conversion of this SAR algorithm runs in about 30 seconds on a 550Mhz Pentium III Linux PC.

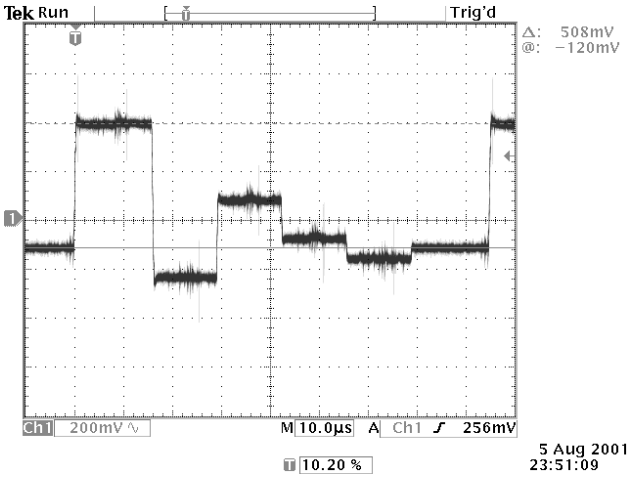

Figure 8. DAC output waveform running on silicon

Conclusion

In this paper, an approach to the design and verification of a mixed-signal programmable system-on-a-chip was presented. By using simplified, but accurate event-driven behavioral models for the programmable analog cells, complete system simulation was performed in Verilog. System verification of complex algorithms can be done in minutes instead of hours or days.

References

[1] E. K. F. Lee and P. G. Gulak, “A CMOS Field-Programmable Analog Array”, in Proceedings of ISSCC, Feb. 1991, pp. 186-187.

[2] Anadigm LTD, “AN10E40 Datasheet”, http://www.anadyne-micro.com.

[3] Lattice Semiconductor, Inc., “Lattice ispPAC Devices”, http://www.latticesemi.com.

[4] K. Keutzer, S. Malik, R. Newton, J. Rabaey, and A. Sangiovanni-Vincentelli, “System Level Design: Orthogonalization of Concerns and Platform-Based Design”, IEEE Transactions on CAD, Vol. 19, No. 12, Dec. 2000.

[5] S. Ganesan and R. Vemuri, ``A Methodology for Rapid Prototyping of Analog Systems," International Conference on Computer Design (ICCD'99), IEEE Computer Society, October 1999.

[6] Cypress MicroSystems, Inc., “PSOC MCU Devices”, http://www.cypressmicro.com.

[7] M. F. Mar and R. W. Brodersen, “Simulation Techniques for Systems with Oversampling A/D Converters”, Proc. of the IEEE ASIC Conference, Rochester, New York, 1993.

[8] J. T. Buck, S. Ha, E. A. Lee, and D. G. Messerschmitt, ``Ptolemy: A Framework for Simulating and Prototyping Heterogeneous Systems," Int. Journal of Computer Simulation, special issue on ``Simulation Software Development," vol. 4, pp. 155-182, April 1994.

[9] J. T. Buck, S. Ha, E. A. Lee, and D. G. Messerschmitt, “Ptolemy: A Framework for Simulating and Prototyping Heterogeneous Systems”, Int. Journal of Computer Simulation, special issue on “Simulation Software Development,” vol. 4, pp. 155-182, April 1994.