Improving SoC Design Flows with Robust and Precise Embedded Memory Models

Jay Abraham

Silicon Metrics Corp., Austin TX, USA

Abstract

Practically all digital System on Chip (SoC) designs contain embedded ROM, RAM, or register file memories of various sizes. These types of embedded memory on average consume 30 - 50 percent of die area and this percentage is growing at an astounding 25 percent annually [1]. With so many designs containing memory devices, on-chip critical timing paths will either start or end at memory address, data, or control pins. In addition to clock circuitry and I/O ring power consumption, memory power consumption is the most significant factor for on-chip power expenditure [2]. Therefore, in order to generate high-quality SoC designs, one must have the ability to accurately model embedded memory devices [3]. Existing methods for embedded memory characterization and modeling for timing and power take too many shortcuts. These methods result in poor quality models that when used in design flows for nanometer technology processes cause timing closure issues and unpredictable power analysis results. This paper presents a unique methodology for accurate and complete timing and power model generation for embedded memory. Unlike current solutions that advocate simplifications that sacrifice accuracy and correctness of memory models, our solution enables processing and analysis of the entire memory in order to automatically generate accurate and complete embedded memory models. The benefits of using this solution with two SoC design flows will be discussed.

1 Introduction

With the increasing use of embedded memories in SoC designs (see figure 1), it has become more difficult for design construction, optimization, and analysis tools to efficiently perform their function. Standard cells are the building blocks of modern IC designs. However, embedded memories cannot be characterized and treated like standard cells [1, 3]. This means that design construction and optimization tools like physical synthesis must rely on inaccurate memory models and are therefore unable to accurately account for the correct timing and power of the instantiated memory. Instead, the tool users rely on approximated constraints or poorly correlated models and must rectify approximations at final analysis.

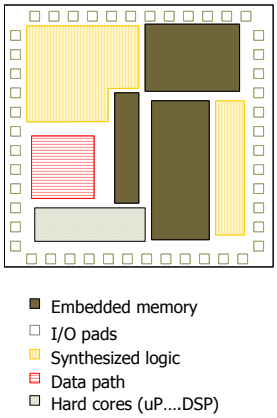

Figure 1: Typical SoC Design Layout.

To mitigate these inaccuracies, SoC teams are forced to gaurdband and over-design. However, over-designing to accommodate timing inaccuracies translates into increased die area and increased manufacturing costs. Over designing to accommodate power inaccuracies directly influences chip pin-out and packaging costs. Lack of information and inaccurate estimates can delay power related decisions, thus negatively impacting design schedules. This is especially troublesome in mixed-signal designs where pin-out can determine the success or failure of analog components. The implications of all these trends are clear: memory characterization and modeling is of significant concern to SoC design teams and accurate memory models are required in all phases of design (see figure 2).

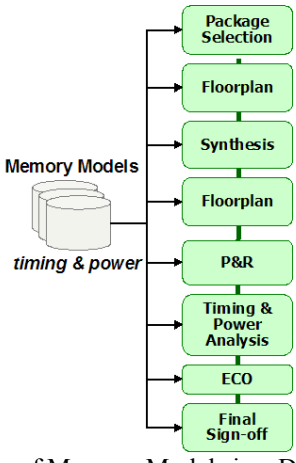

Figure 2: Use of Memory Models in a Design Flow.

2 Traditional Solutions for Generating Memory Models

Traditional solutions for memory model generation are mainly repackaged memory compiler based techniques and lack significant levels of automation. Typically they consist of either a netlist prune/cut based approach to approximate electrical behavior of the memory device or, the use of memory compilers to generate required models. Both of these approaches will be considered in detail.

3 Pruned Netlist or Cut Based Memory Model Generation

Some design teams attempt to simulate a memory with SPICE by means of a pruned netlist or cutting approach. This manual methodology can be a very time consuming and error prone task. Due to capacity issues, it is impossible to simulate an entire memory using SPICE. Therefore, the memory netlist must be cut and pruned. Netlist cutting creates a piecemeal analysis of the memory components. The address decoding logic, wordline, bitcell, bitlines, column multiplexing, and output logic are analyzed individually using SPICE and the final model is composed of calculations based on the individual results. Netlist pruning is the process of isolating one or more addresses in the core while discarding the rest. The result is a smaller netlist on which SPICE can be used effectively. This process is error prone and time consuming, especially for designs in the post-layout phase. Standby and active power usage are estimated based on the size of the memory.

As shown in the figure 3, the process for generating a memory model with a netlist prune/cut based approach is fairly complex. The phases for critical path identification (for pruning purposes), reduction of the extracted netlist, and specification of the measurements and stimulus to drive SPICE are not trivial. These tasks require transistor-level expertise as well as significant preparation time. Once again, there can be significant loss of accuracy due to the cutting process and reduction of the netlist. Cutting leaves lots of dangling nets which if tied off with lumped capacitors will not reflect the actual behavior of active transistor devices [1]. Additionally, it is vital that this approach find the correct critical path to prune because this is the only path in the entire memory that will be simulated with SPICE. Without full transistor-level analysis of the netlist, including all of the detailed parasitics, it is difficult to manually determine the actual critical path.

Typically, the final accuracy of a netlist prune/cut based method is better than relying on a compiler generated memory model; however, significant inaccuracies still exist. When comparing to SPICE, about 25% - 40%. Error is usually observed. This error is significant for today's nanometer SoC designs. Finally, a netlist prune/cut method does not address the need for accurate power characterization. Instead a maximum power usage value can only be estimated. This estimation can have dire consequences for a today's battery operated devices.

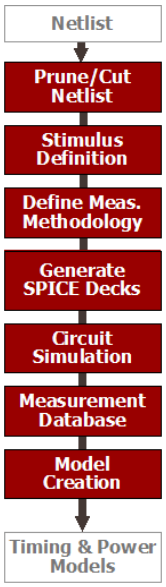

Figure 3: Typical Flow for Netlist Prune/Cut Based Characterization and Model Generation.

4 Memory Compiler Generated Memory Models

Memory compilers generate models of a specific memory instance by interpolating the performance from the data of a few known instances. Therefore, the actual memory instance instantiated in the design may not have been characterized. Additionally, the chosen instances are characterized by pruning and cutting statistically determined critical paths. Pruning and cutting as explained in the previous section adds additional error to the characterization process. Interpolation and extrapolation is accomplished with various mathematical functions, such as linear interpolation, statistical curve fitting, time series analysis, and splines. Given that interpolation and extrapolation isn't an exact match, there can be significant error when comparing the interpolated data to the actual results [7].

In order to verify accuracy, the following experiment was conducted with an 8X2bit single port RAM. This analysis consisted of running SPICE for various memory constraints such as access time and address setup. The data from these measurements were then compared to data from typical memory compiler generation processes.

We created a compiler using standard memory compiler

techniques which involve selective characterization of unique memory types, i.e. user selected address and data input/output bus widths, aspect ratio, etc. This method is favored by commercial compiler developers utilizing a netlist cut or prune based approach [8]. The characterized data was then interpolated to generate data for the memory model in question (the 8X2bit single port RAM). Due to runtime limits of running SPICE on larger memories, a small memory was chosen for this example. The graphs for SPICE versus compiler data are shown in figures 4 - 7. Due to runtime overhead for power consumption data (>3 days per measurement) we had to use aggressive netlist cutting techniques. This approach is typical in commercial compiler designs. Data for power accuracy is shown in figure 7.

As shown in the graphs in figures 4 - 7, cutting, pruning, estimating, gaurdbanding, interpolating, and extrapolating can inject a significant amount of error in to the final analysis of the memory. In this example a maximum error of about 110% was observed (see figures 8 and 9). Given that the memory device is often in the critical path of most SoC designs, inaccuracies as shown below will significantly impact the final performance metrics of the design.

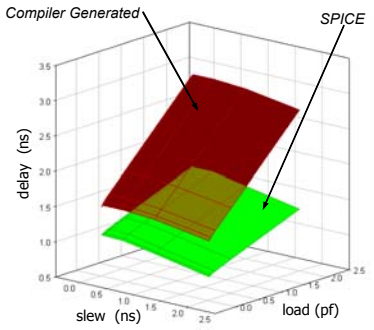

Figure 4: Access Time.

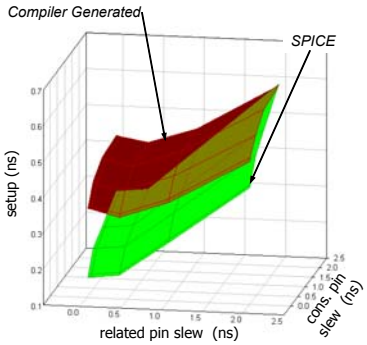

Figure 5: Address Setup.

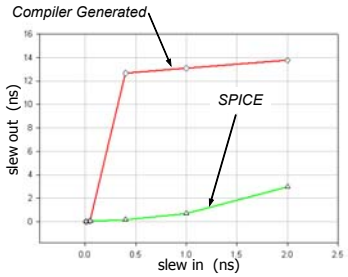

Figure 6: Output Slew.

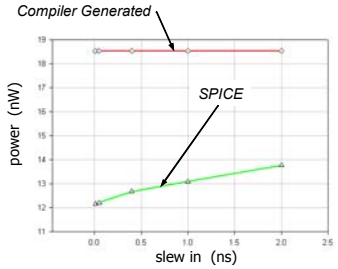

Figure 7: Average Power.

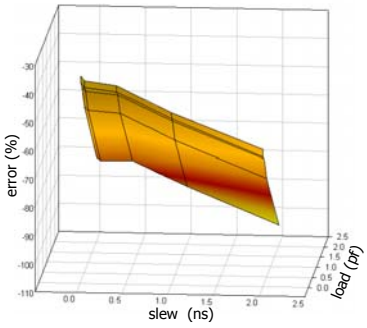

Figure 8: Access Time Error (-40% to -100%) for Compiler vs. SPICE.

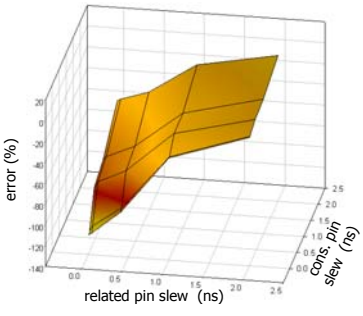

Figure 9: Address Setup Error (-2% to -110%) for Compiler vs. SPICE.

Additionally, memory models generated from compilers for timing analysis are usually created for single process, voltage, and temperature (PVT). If the model is not designed to represent the actual operating point

(environment), then there is a real danger the memory may not operate within the timing constraints [1]. Additionally, compiler generated memories rely on foundry data available during shipment of the compiler. Therefore, the memory models generated may not be built from the most recent foundry SPICE models. These deficiencies in modeling accuracy can result in failed power-on of the chip.

Standard compiler based memory designs only marginally support ultra-low power design efforts. Due to estimation in standard power characterization methods for memories, power usage is often approximated to a fixed value. However, EDA tools utilizing the ALF [4] and Liberty® (Synopsys .lib) [5] library formats require variable average memory power usage. In the absence of adequate characterization solutions, memory compilers will insert gaurdbanded estimations which are closer to maximum power usage than average power usage. As a result, power consumption budgets have artificial constraints and additional guardbanding. In ultra low power applications, this type of guardbanding may overburden other parts of the design where the excesses must be compensated for in order to meet strict power consumption limits.

5 A Contemporary Memory Characterization and Modeling Solution

The latest solutions for advanced memory modeling should strive to improve the accuracy of the models, while also providing an automated, high-throughput flow for the generation of the models. Ideally characterization and model generation should be a push button process. One of the ways to accomplish high throughput runtime is to leverage hierarchical simulators. Array reduction capabilities in these simulators are suited for memory devices. Additionally, the variance to traditional SPICE is often only 2% - 4% , resulting in highly accurate timing and power models [6].

The flexibility provided by this method enables designers to generate memory models that fit the unique needs of a particular design including the actual operating point(s) (PVT). Because the entire post-layout netlist is actually simulated, designers no longer need to compensate for estimation in the measurement process. The models generated are instance specific for the SoC design that they will be embedded in. Because the memory model is applicable to a particular design, guardband is user-controllable. Most importantly memory characterization no longer requires netlist cutting, netlist pruning, or synthesis of the measurements of the individual memory components.

In order to verify memory characterization and modeling with, we built the following automated flow (see figure 10).

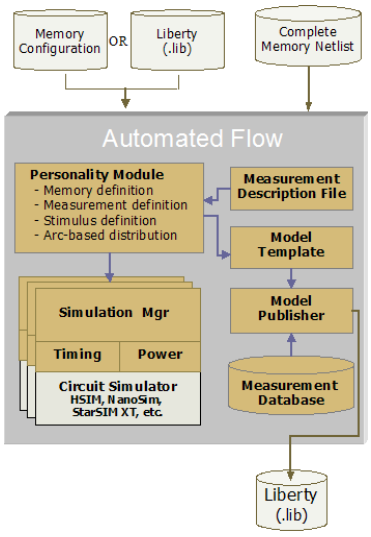

Figure 10: Automated Flow for Memory Characterization and Model Generation.

The inputs to this system consist of a complete memory netlist with interconnect parasitics and a memory configuration file. The memory configuration hierarchically describes the function of the memory, including pins, functional relationship between pins, and in some cases the timing and power arcs within the memory. The memory configuration input is simple to generate and is typically 15-30 lines of Tcl code. Based on the configuration input, the system automatically determines the required measurements. The alternate input is the industry standard Liberty™ (.lib) source file. Once the measurements have been determined, simulation is performed with a high-speed simulator. Data is stored in a measurement database and model writers generate Liberty™ (.lib) models. We also built a model verification capability to verify syntax and semantics of the generated models.

Results from this methodology showed significant promise in terms of the accuracy of the models generated as well as the throughput of the system. In terms of accuracy, the graphs in figures 11 and 12 show the results of comparing measurements from this system compared to exact SPICE measurements.

As depicted in figures 11 and 12, the new models generated with the automated flow utilizing high-speed simulators generates models that are very accurate. Typical variance from SPICE is 2% - 3%, with a maximum variance of about 5%. Throughput is extremely efficient as well. Utilizing a parallel processing technique for arc by arc parallel distribution as well as SPICE deck optimization, the single port RAM (size: 8X2) was

processed in approximately 15 minutes. As a result of the hierarchical processing capability of simulators, larger memories are characterized very quickly. For example, a 32Kb memory requires 2.5 hours and a 256Kb memory requires 9 hours.

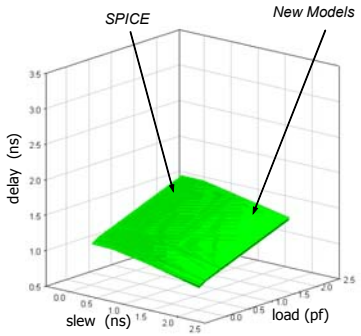

Figure 11: Access Time.

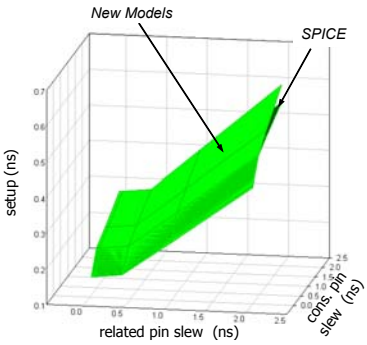

Figure 12: Address Setup.

6 Using Accurate Models in a Contemporary Design Flows

Correct and accurate memory timing models are needed for timing driven place and route, static timing analysis, and final timing sign-off. Accurate memory power models are needed for low cost package determination.

It is important to eliminate complexity from the memory characterization and modeling process to turn characterization and memory model generation into a pushbutton process. Requirements for automation include; automatic stimulus generation, arc-based job distribution, automatic deck creation, archiving, and prepackaged timing and power methodologies that match the target model. High throughput for characterization is obtained with the use of hierarchical simulators, parallelized distribution of simulation jobs that are granularized at the per-arc level, and intelligent spice deck creation to utilize hierarchical processing.

Our characterization and modeling solution addresses these key issues of timing and power closure with embedded memories in advanced SoC designs. With a pushbutton process for generating precise memory models for timing and power, designers can generate memory models for their unique application.

The solution solves power accuracy issues by providing easily configured power acquisition capability. Complex analysis can be performed to determine average power consumption in design-specific configurations. Designers can account for typical utilization patterns such as sequential access versus random access and generate power numbers appropriately. Analysis of various modes, such as low power or quiescent modes is also possible. This gives designers the information needed to make important decisions earlier in the design cycle such as decisions related to packaging, chip pin-out, floor planning, and power routing.

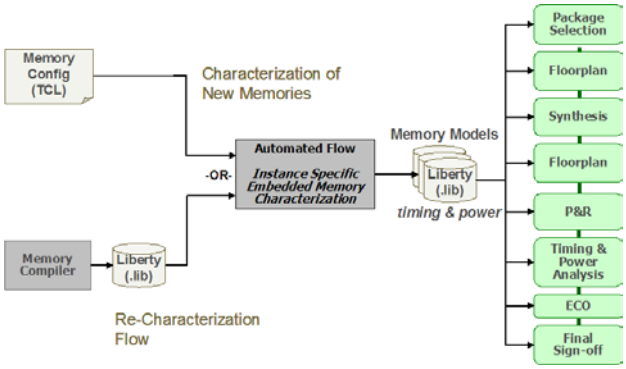

Figure 13: Improved Design Flow with Robust and Precise Embedded Memory Models.

The recommended flow utilizing memory characterization and modeling is shown in figure 13. Two flows that are applicable to SOC designers are shown. The first is a re-characterization flow in which designers take existing memory models (generated by a compiler) and perform a re-characterization step to improve accuracy of the models. Typically the model format will be in the public domain Liberty format (also known as .lib). With this methodology, users can characterize for unique PVT conditions required for their design. The second flow is for characterization of custom memories. These could be hand crafted memories or compiler generated memories that have been modified. In this case, there are no existing models. The suggested flow would involve creating a memory configuration file that is then read by the automated characterization system. This flow is also useful for re-characterization, when an existing memory model generated by a compiler may not be of good quality. In this case, model generation from scratch would be the preferred solution. The flows described liberate memory design teams to concentrate on circuit design and memory architecture development rather than

the tedious and time-consuming process of verifying and supporting existing designs.

7 Conclusion

Time-to-market, custom package tooling, die real-estate requirements, and the ensuing design closure issues create a complex decision making process that demands accuracy and flexibility. Embedded memories are often at the core of this process due to their increased dominance in SoC designs and can negatively impact a design flow. Traditional methods for memory model generation and characterization are often inaccurate and use inefficient processes. A new solution that leverages high capacity simulators offers the capability to generate accurate models with high throughput. This system was verified with test cases to demonstrate the capability to generate accurate models for various memory types. The system is flexible for acquiring data for timing and power. Data described in this paper showed a minimal 2% - 3% variance from SPICE. The use of this system in SoC design flows will improve quality of design. The commercial application based on this research, SiliconSmart MR from Silicon Metrics, showed that accurate timing and power memory models of any size can be generated with a rapid, precise, and pushbutton process. Thus enables designers to successfully implement high-speed and high-density embedded memory-based SoC designs that meet today's demanding cost and schedule requirements.

References

[1] Eric Hall and George Costakis, "Developing a Design Methodology for Embedded Memories", *Integrated System Design*, January 2000.

[2] Alberto Macii, Luca Benini, and Massimo Poncino, "Memory Design Techniques for Low Energy Embedded Systems", *Kluwer Academic Publishers*, 2002.

[3] Tegze Haraszti, "CMOS Memory Circuits", *Kluwer Academic Publishers*, 2001.

[4] "Advanced Library Format for ASIC Technology, Cells, and Blocks version 2.0", *Accellera*, 2000.

[5] "Library Compiler Reference Manual volumes 1-3", *Synopsys*, 2001.

[6] Sam Wang and An-Chang Deng, "Delivering a Full-chip Hierarchical Circuit Simulation and Analysis Solution for Nanometer Designs", *Nassda*, May 2002.

[7] Ravi Agarwal and Patricia Wong, "Error Inequalities in Polynomial Interpolation and Their Applications, *Kluwer Academic Publishers*, July 1993.

[8] Charles Longway and You-Pang Wei, "Automatic Memory IP Characterization", *EEdesign*, December 6 2000.