Including variability in simulation of logic circuits

D. Kevin Cameron, *Member, IEEE*

Abstract—IC fabrication technology has outpaced EDA methodology for a number of years; there is little support for power management verification in languages like Verilog, and the old "min-typ-max" timing verification became useless somewhere around 45nm. The subject of this poster is the use of probability functions in conjunction with analog power modeling in (standard) languages like Verilog-AMS to provide combined power and timing sign-off simulation. The technique is complementary to static timing analysis (which is normally pessimistic) since it works in the usual test verification flow (with directed and random testing). Using analog behavioral modeling solves issues static timing analysis (STA) finds difficult and also addresses some problems related to clock-domain-crossing.

***Index Terms* — FinFET, Probability, Timing Analysis, Variability, Verilog-AMS, Sign-off, Simulation, Modeling**

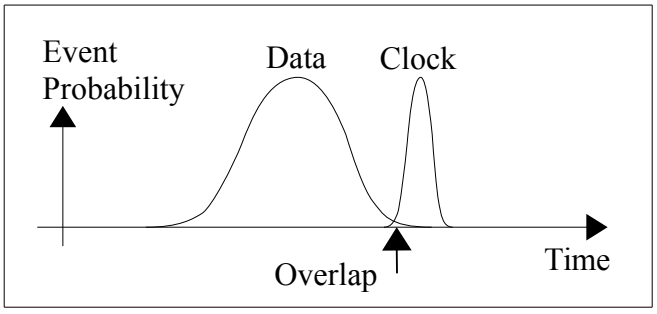
I. Introduction

The problem of variance in semiconductor manufacturing has been with us since the beginning, the recent change is that the device-to-device spread has become higher than the chip-to-chip or wafer-to-wafer spread in processes below 45nm. In processes above 45nm it was reasonable to run "corner" simulations on a design to see if it would work if the manufacturing process was off-center in some dimension. In practice this usually came down to running "minimum", "maximum" and "typical" speed simulations for the transistors as part of the "sign-off" process before committing to manufacture. Below 45nm any individual device can be fast or slow so there is an exponential explosion in the number of corners that would be required for a similar level of confidence.

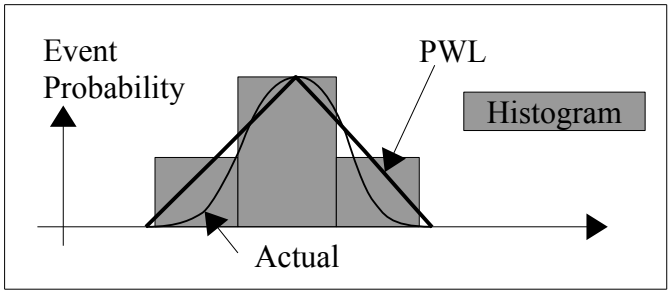
An early attempt at compressing these multiple simulations into a single simulation was a technique called "histogram" simulation, but since it appeared well before 45nm Silicon, it was of limited benefit; thermal and battery life issues have lead designers into using multiple power and clock domains on a single chip which was also not addressed by that technique. Here we describe a newer approach that leverages the behavioral modeling capabilities of languages like Verilog-AMS and VHDL.

II. A Probability Model For Simulating Logic

The goal of probability modeling is to incorporate the spread of behavior in the lower level component models so that when those models are combined to make up a complete design all the corner cases will be covered in a single simulation. In RTL based logic designs the mode of failure we are usually looking for is when data arrives too late at registers or "race" hazards, and more recently clock-domain-crossing (CDC) hazards. Any individual logic gate will have a spread in behavior which can be represented as a spread in delay of events through the gate model, downstream logic will similarly have a spread, and each level of logic contributes such that the spread on reaching the next register in RTL design will be from the time with all "fast" to the the time with all "slow" logic. The clock that drives the registers will likewise have a spread of its own, and condition we are looking for is an overlap of the data spread and clock spread such that we have less than 100% certainty when latching the data:



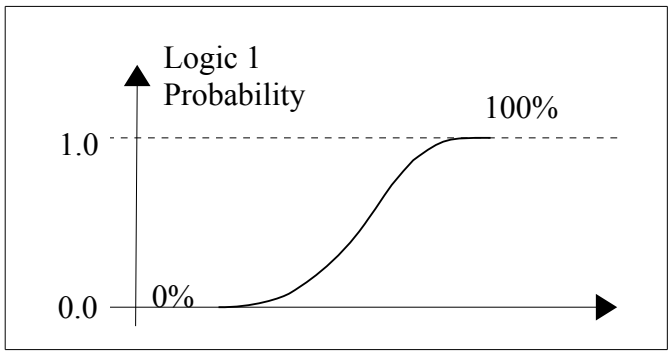
The earlier "histogram" simulation approach used the " min,typ,max" timing information (from SDF) for gates to create a series of events out of a given logic gate for an input event, an input 0/1 transition would cause multiple output events: e.g. one for the earliest possible event, another for typical timing,and one for the slowest possible timing, using discrete levels like 5% and 50%. This was a rough modeling of the delay characteristics of the gate in question, but fits with the modeling support/style seen in simulation languages like VHDL '93. A better approximation is to use piecewise-linear (PWL) models that are discrete in the first derivative and will break away from the absolute 0/100% in a more realistic manner but does not require more data points (three for PWL, vs. four for histogram in the simplest case):



. This paragraph of the first footnote will contain the date on which you submitted your paper for review.....

PWL signaling is the default in analog simulators, and if you use a language like Verilog-AMS you can define a waveform as a continuous function of time (not discrete in the derivatives) and it will be reduced to a PWL approximation during transient simulation according to accuracy requirements.

The event-time probability curve is not particularly useful in itself for simulation, but if we integrate it we get a curve that goes from 0% to 100% probability for an event occurring. For simplicity we can just view this as the probability of a logic '1' value, or negatively as the probability of a '0', which we can represent in simulation as a potential running from 0.0 to 1.0:



Waveforms in simulation then look very similar to what you would expect to see when simulating with voltages and currents, but transitions will stretch out over longer periods as you go through more layers of combinational logic.

Incorporating Load

In Verilog-AMS we can also define a corresponding flow component to go on the "wire", and that can be a loading factor (similar to capacitance) which can be incorporated into the probability function to increase the event delay on gates with higher fan-out. That allows for verifying load dependent delays which STA can't handle when the load is state dependent, and also enables reusing the same cell models with back-annotation. Other factors can be included into the computation of the probability function such as local supply voltage, and temperature which Verilog-AMS and VHDL support directly.

Verilog-AMS also supports automatic conversion of analog types to boolean/strength logic types using "connect modules" so existing test benches in Verilog can be used with a DUT (device under test) described using "probability logic". With the type described above, a 0.5 can be considered 'X'/unknown and above and below true/false respectively.

Both SysytemVerilog and VHDL have record types that can represent the same PWL data for the probability component of the signal model, but lack the "flow" component, and do not have automatic conversion of wire types or the ability to reduce continuous functions to PWL automatically.

Performance

While the kind of modeling normally done with the analog portions of Verilog-AMS and VHDL is associated with matrix solvers, a solver is only required in certain circumstances: where there are cross-coupled dependencies and rules like Kirchoff's Current Law to be obeyed. In the case of probability modeling this only occurs when the loading (flow) is dependent on the logic state, most other effects are not tightly coupled.

Model Creation

Models for logic gates are fairly easy to create since most functions can be built with N-input "and"/"or", and "not" operations, and the N-input gates can be built (logically) by chaining 2-input gates. The output probability of a 2-input *and* gate is just the multiplication of its inputs, for a 2-input *or* gate (inputs *a* and *b*) the output is: $1-((1-a)*(1-b))$. The calculated output can then be filtered through the delay function for the gate in question. Resolution of like-strength signals is just their average.

Failure Detection and CDC

Models for gates are fairly simple in concept, but latches & flip-flops are more complicated since they have to test for overlapping clock and data conditions and feed values into the next bank of logic. The absolute variability in clock signals is less important than the bank-to-bank variability if the registers in question are using the same clock, and that should be taken into consideration when setting up the simulation. In the case where the clocks on the registers are not correlated (the CDC case), the full variability of the clock needs to be used.

In the case where clock and data overlap the uncertainty can be propagated to downstream logic indicating the design failure and/or the possibility of a metastable state. This will appear as values not going completely to 1.0 or 0.0 even if downstream logic doesn't have timing problems. The propagated value will be the integral of the data value (the probability of a 1) multiplied by the clock event probability for the clock event transition period.

Yield Estimation

In addition to timing errors this modeling technique can be used to estimate yield if you know the probability of a component failing. By setting the logic probability limits short of the absolute 0.0 and 1.0 according to the likelihood of failure, e.g. with a 1% chance of failure the probability logic value would have limits like 0.01 and 0.99, and passing through another similar component will reduce the limits to 0.02 and 0.98. Running test vectors on the DUT should then give you pin values with limits corresponding to the yield seen on actual testers.

References

[1] D.K. Cameron, "Including variability in simulation of logic circuits", USPTO # **8,478,576**

[2] Paul B. Weil, "Method and apparatus for histogram based digital circuit simulator", USPTO # **5,383,167**

[3] Accellera, "Verilog-AMS Language Reference Manual" http://www.accellera.org/downloads/standards/v-ams/VAMS-LRM-2-4.pdf