A Modeling Methodology for Verifying Functionality of a Wireless Chip

Jesse E. Chen
Qualcomm Inc.
3165 Kifer Rd.
Santa Clara, CA 95051
(408)216-2578
jessec@qualcomm.com

ABSTRACT

This paper describes a modeling methodology for verifying functionality of a mixed signal wireless chip before tape out. Modeling methodologies for mixed signal chips can be distinguished by the way they deal with analog signals. The methodology uses a custom Verilog PLI function to model analog blocks for a digital simulator. The PLI function passes multiple real numbers through one port, in either direction, at any point in time. The key issues the methodology addresses are execution speed, capacity, model portability, and coverage.

1. INTRODUCTION

Functional verification becomes more critical as chips become more complex. Performance issues like noise and distortion are always a challenge but with increased complexity things like connectivity bugs, jumbled up programming sequences, and misinterpreted interface specifications cause most re-spins. If first silicon can be powered up and put into all modes of operation we can at least measure and diagnose performance issues. If the chip cannot be powered up, reset, or programmed, another chip spin would be required before anyone could test performance. For first silicon, functionality is more important than whether the noise figure or IIP3 [1] are a few dB off.

Today's wireless chips contain several radios for diversity and GPS [2]. Tomorrow's devices will cram even more radios onto a single chip or into a single package. Most radios operate over several channels and multiple gain states. In addition, each radio has several automatic calibration loops and power saving modes. The full functional verification space can easily span thousands of simulations, especially if we count diagnostics. To make matters worse some tests cover milliseconds of action. Even the latest mixed signal simulators take days to simulate a millisecond of chip level action. To achieve a reasonable degree of pre-tape out confidence we have little choice but to exploit the speed and capacity of a purely digital (i.e. event driven) simulator.

Digital blocks require no modeling work because digital simulators work directly with any state of the actual design (behavioral, RTL, gate). In contrast, analog blocks require complex hand crafted models.

In addition to running fast, the hand crafted models must also be portable; they must work in several design environments. Big companies get big by buying lots of small companies. Different small companies use different tools. Further, the best tools, design, and simulation environments for each team are often specific to each team's individual design requirements. Also, newly acquired teams rarely have time to learn a new environment or convert their intellectual property to a common design environment. SOC (System on a Chip) models must therefore work with multiple digital simulators. We also require our models to work in the analog design environment for four reasons: 1) Hand crafted models of analog blocks must be certified because they do not automatically match their circuits. If a hand crafted model works while the real circuit does not, we tape out a design bug. Side-by-side simulation of the model and schematic in the analog environment aligns the two views. 2) The analog designers know the low-to-middle layers of hierarchy better than anyone else and typically out number our verification engineers. Compatibility with the analog design environment brings the analog design team to bear on diagnostics and model certification. 3) The ability to mix and match behavioral and device level models of analog circuits accelerates diagnostics. A hand crafted model is often the primary suspect of an anomalous symptom, and rightfully so. By replacing a suspect model with its device level counterpart we can quickly convict or acquit the model. If the symptom persists with the device level model, the design is in error. If the symptom goes away, the model is in error. The device level model runs slower but that single diagnostic run finishes in far less time than we otherwise waste prosecuting the wrong culprit. 4) The fast behavioral models can greatly accelerate testbench development. After the testbench has been developed, we can replace the behavioral models under test with the device level counterparts and focus on testing rather than testbench development.

The challenge in simulating analog blocks with a digital simulator is to pass real number traffic between modules. One approach is not to bother. You could check connectivity with digital clocks and insert assertions (print statements) [3] to log how a true analog model would have responded to its various digital commands. Consider a receiver chain containing several programmable gain blocks. You could pump a digital clock signal down the chain to check connectivity through the signal path. To check end-to-end gain, each block in the chain could respond to its digital gain commands by asserting the selected gain (to the log file). A post processing script could then interrogate the log file and compute the end-to-end gain profile. Assertion based verification of the receiver works as long as you are only interested in the RF/analog receiver. Assertion based models cannot be used at the chip level because they do not exercise the ADC (analog to digital converter). The larger receiver chain includes baseband digital signal processing algorithms that don't work without real numbers flowing into the ADC models.

Real number traffic poses two problems for digital simulators, one associated with paralleled drivers and one associated with RF signals. The driver problem is that voltages contend while currents sum. Any resolution function that deals with multiple drivers must know whether the signals are currents or voltages. RF signals are troublesome because explicitly simulating every cycle of an RF carrier significantly slows execution. Even with a scaled down carrier frequency explicit simulation of every carrier cycle still requires receiver filters that actually filter. As shown later, a more elegant solution is to use baseband equivalent models [4]. Baseband equivalent models replace high frequency RF real signals with low frequency baseband complex signals. The complex signals require transmission of at least two real numbers across a single wire at each time point.

Before deciding on a PLI function approach we looked at four other ways to pass real numbers between modules. I refer to these alternatives as real-to-bits (R2B), wreal, VHDL, and System Verilog. I first discuss the R2B approach. To send a real number from one module to another, the sending module converts the real number to a 64-bit digital word. The 64 bits can be sent over a 64-bit bus or by "telegraphing" the bits across the wire. The receiving block then converts the digital word back to a real number. We cannot use parallel transmission because replacing the single wire with a 64-bit bus implies we would verify a different schematic than the one we tape out. The telegraph option sends the 64 bits across the wire at a bit rate at least 64 times higher than the sample rate of the real number. A typical real number sample rate is on the order of 100MHz. This would require a telegraph bit rate of 6.4 Gb/sec. Assuming we use baseband equivalent models for RF blocks, RF signals require transmission of two numbers during each sample period, which raises the bit rate to 12.8 GB/sec. This sample rate is high but tolerable. Current summing increases the bit rate even further depending on the number of currents entering a net. To sum three currents the bit rate rises to 19.2 Gb/sec. For currents, the B2R approach amounts to a crude form of TDMA (time division multiple access). One problem with the TDMA approach is that the time slots must be pre-assigned to the various current drivers on the net. If two signals accidentally share a time slot they either generate x-states (if they differ) or contribute only half of what they should (if they are the same). Similarly, if two drivers representing voltage sources accidentally drive the same voltage onto the same node they will not generate any symptoms of a contention problem. The B2R approach is portable but does not detect all connectivity bugs and is somewhat clumsy at summing currents.

"Wreal" refers to a VerilogAMS approach to passing real numbers that does not involve any analog simulator [5]. When we examined this approach, a wire in VerilogAMS declared as wreal could pass one real number, one way, all the time. The "one real number" limitation ruled out baseband equivalent models. A more subtle but important implication of this limitation was that wreals could not deal with multple drivers on a common net; wreals could not sum currents. The "one way" limitation meant that we could never reverse the flow of real number traffic, which is important when a limited pin count forces us to use some pins for bidirectional signals. The "all the time" limitation meant we could never have two drivers take turns driving a common node because we could never turn either driver off. (It is very common for two RF front ends to take turns driving common baseband legs.) Another problem with the wreal approach was that it was not well supported by all simulators; it was not portable.

VHDL and System Verilog are each rich in real number features but at the time we examined these options the netlister associated with our analog design environment did not support the relevant features. Without a netlister we could not directly verify lower layers of hierarchy, hierarchy described by schematics. VHDL and System Verilog were not sufficiently portable. Furthermore, most of our chip level verification teams prefer Verilog to VHDL. They use System Verilog but not for analog models. System Verilog is still matruring and not all simulators support the full capabilities of the language.

We settled on a custom Verilog PLI function (CVPLIF) for passing real number traffic. The PLI approach is not new [6], and neither is the idea of event driven simulation of baseband equivalent signals [7]. What appears to be new is a portable implementation of the key HDL enhancements proposed in [7] along with the ability to distinguish real numbers representing voltages and currents [6]. I refer to a wire driven by a CVPLIF as a “HyperWire”. The remainder of this paper describes the modeling and verification strategy we developed around the CVPLIF.

2. SIGNAL PATH STRATEGY (BASEBAND EQUIVALENT MODELS)

One of the key things the CVPLIF lets us do is baseband equivalent modeling in Verilog. Since the baseband algorithms on both sides of the wireless link operate only on the baseband data riding on the RF carrier we can accelerate run times by suppressing the RF carrier. The resulting models are called baseband equivalent models. Baseband equivalent models have been around for decades. They were used as early as 1929 in the context of rotating machine theory [8]. However, the ability to write pin accurate baseband equivalent models in Verilog is new. This section gives a brief geometric overview of baseband equivalent modeling and then discusses three key models: a down converter, a PLL (phase lock loop), and a filter.

A generic RF signal can be written as

RF(t)=I(t) cos(ωt)-Q(t) sin(ωt).

“ω” is the carrier frequency in radians per second. I(t) and Q(t) describe the baseband information riding on the carrier. The transmitter generates RF(t) from I(t) and Q(t). The receiver extracts I(t) and Q(t) from RF(t). Referring to Figure 1, a little high school geometry shows that RF(t) is the projection of the vector V(t) onto the ReRF axis. The vector V(t) spins about the origin like the second hand of an ultra fast clock running backwards. However, the length of the second hand varies slowly and the clock does not keep very good time because its speed slowly varies too. The I(t) and Q(t) signals of interest directly determine the slow deviations of V(t) from a fixed length, fixed speed, reverse second hand. Aside from the slow variations, V(t) spins about the origin at the RF carrier frequency. Relative to the fixed ReRF-ImRF axes, V(t) and its projections therefore vary rapidly. Simulation in the ReRF-ImRF reference frame requires very high sample rates. To simulate 2.5GHz sinusoids for example, we need sample rates on the order of 50GHz. Such a high sample rate can easily strain the available computing capacity in terms of speed and memory. However, relative to the reference frame ReBB-ImBB that spins more or less along with V(t), i.e. at the carrier frequency, V(t) varies slowly. I(t) and Q(t) are the coordinates of V(t) in the rotating reference frame. Simulating V(t) in the ReBB-ImBB reference frame requires sample rates related only to the bandwidth of I(t) and Q(t), which are orders of magnitude less than the carrier frequency. The baseband equivalent signal, BBeq(t)=I(t)+j*Q(t), is essentially a slowly varying phasor. The dramatic reduction in sample rate required for a baseband equivalent signal significantly improves run times. However, baseband equivalent models require that we pass at least two real numbers, I(t) and Q(t), between modules instead of one real number, RF(t).

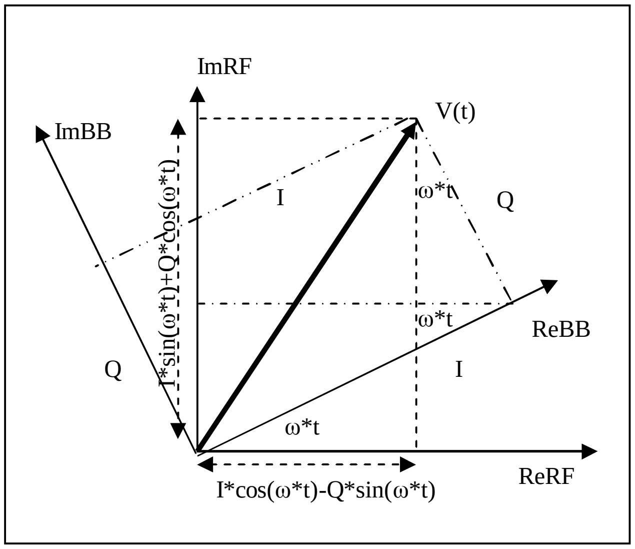

Figure 1, Baseband Equivalent Signals

We also send spectral data along with the temporal (IQ) data. We send the carrier frequency so that the down converter model can monitor the RF carrier and LO frequencies and output the IF frequency (i.e. the down converted frequency), which can be zero. The IF-frequency is still referred to as a carrier frequency. By simply checking the carrier frequency exiting the receiver chain we can determine whether the synthesizer was tuned to the correct channel. The down converted carrier frequency can also be used to monitor the frequency of an FM signal.

The down converter model does more than just apply a gain to the baseband equivalent input signal and compute the output carrier frequency. Ideally, a direct conversion down converter extracts I(t) and Q(t) from RF(t). Referring again to Figure 1, the rotating axes (ReBB-ImBB) then represent the I- and Q- outputs of an ideal down converter. However, in the down converter model, the ReBB-ImBB axes actually spin at the LO frequency. The LO frequency does not necessarily equal the carrier frequency. If a small frequency error exists, V(t) spins relative to the down converter model at a rate equal to the frequency error (i.e. the IF frequency).

The PLL model must be compatible with the signal path model. The baseband equivalent models in the signal path require a real number representing LO (local oscillator) frequency. The real number requirement is not a problem because schedule constraints usually force us to develop a high level PLL model ahead of the actual circuit design. Furthermore, we do not need a detailed PLL model at the SOC level because the PLL interfaces are fairly well specified. At the chip level we simulate the PLL with a crude, flat, fast model. The PLL model is a non-linear phase domain model capable of simulating fine tuning transients associated with closing the analog feedback loop after coarse digital tuning. A simple delay models coarse tuning (capacitor bank selection). The output of the VCO is a real number representing frequency. Dividers between the VCO and down converter are simple gain blocks. (We build more detailed models of the PLL but we use them in a stand alone mode to verify that the internal state machines and registers can sweep through all channels and that the PLL can lock at each channel. The high level and detailed PLL models must satisfy the same functional specifications.)

Figure 2 shows the IQ trajectory of an 802.11a short training sequence at the output of a direct conversion down converter simulated with Verilog models equipped with the CVPLIF. The fine tuning transient of the PLL keeps the "pretzel" from exactly retracing itself. As the LO frequency settles to the correct target frequency, the traces grow closer together. We include an option in the PLL model to skip both coarse and fine tuning transients because the transients do not affect most of the functional tests.

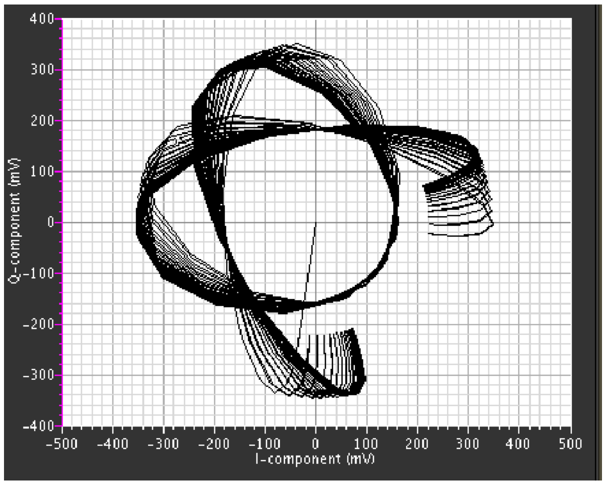

Figure 2 IQ pretzel trajectory

Since the receiver model uses baseband equivalent models, and since no SOC functional test requires jammers, there is no functional need to explicitly model the filters. We only need to ensure the filters are programmed to the correct bandwidth. We therefore also send the signal bandwidth along with I, Q, and the carrier frequency. The bandwidth component of the HyperWire saves us from 1) having to model analog filters with digital equivalents and 2) having to input a wide band signal or sweep the frequency of a sinusoidal input. For a direct conversion receiver, the filter model outputs a bandwidth HyperWire component equal to the minimum of the incoming bandwidth and the programmed filter bandwidth. As long as the input bandwidth signal is large enough we can easily check filter programming by checking the BW component exiting the receiver chain. We could check BW programming with an assertion but the HyperWire approach gives us everything at a convenient single output point.

We send the DC component of the signal as another component of the HyperWire signal because it cannot be combined with the baseband equivalent signal. Viewed from the baseband reference frame (ReBB-ImBB) in Figure 1, a vector fixed in the ReRF-ImRF reference frame spins clockwise at the carrier frequency.

3. PSEUDO-ELECTRIC SIGNALS

As mentioned earlier, models of analog blocks must be hand crafted. Hand crafted models do not get taped out. Modeling boundaries separate what we tape out from what we do not tape out. We push the modeling boundaries as far down the hierarchy as we can to verify as much of the schematics and embedded logic as possible. On the other hand, we do not push the modeling boundaries down so far that we must account for loading between analog blocks. (Digital simulators do not easily simulate loading.) The resulting modeling boundaries force us to deal with the differences between voltages and currents.

Voltage and current sources behave differently when paralleled. Paralleled active voltage drivers always constitute a design bug, a bug classified as voltage contention. Paralleled current drivers do not contend; they sum. However, a net current with no place to flow constitutes current contention. We do not care exactly what happens during either form of contention; we only need to know that contention occurred.

The CVPLIF uses the logic value of the HyperWire to flag contention. If no contention exists on a HyperWire its logic value is either 1'b0 or 1'b1. If contention exists the CVPLIF drives the logic value to 1'bx. If no active CVPLIF drivers exist, the logic value of the HyperWire is 1'bz.

Individual HyperWire signals can have dimensions of volts, amps or be dimensionless. Signals with units are called pseudo-electric signals. Dimensionless signals contend like voltage signals.

To monitor current you must terminate the wire with a voltage driver; you must give the current some place to flow. A current probe is a voltage driver, possibly driving zero volts. If you do not give a current signal a current probe to sink the current, the CVPLIF drives the net to 1’bx, signifying current contention.

The voltage driver used as a current monitor serves another useful purpose. Consider a bias current generator and a bias current consumer. One bias generator should connect to just one bias consumer. If a bias pirate were accidentally connected to the same bias generator, the two current probes (i.e. voltage drivers) would contend and the CVPLIF would create an obvious 1’bx symptom of the connectivity bug.

Pseudo-electric signals pass through HyperWires in different directions at the same time. You can transmit voltage one way and current the other without turning either driver off. This feature makes it very easy to track current consumption. Current consumption is not only a critical and programmable functional parameter in itself; it is also a key symptom of problems that can occur during scan chain tests. The power supply is a voltage driver. Each port connected to the supply can drive a pseudo-electric current signal back to the supply while simultaneously monitoring the pseudo-electric voltage signal. The current drawn by each block can depend on the various modes of operation. The pseudo-electric current signals sum at the voltage driver making it possible to generate current consumption profiles. Current consumption profiles quickly reveal unexpected high current modes and verify low power modes.

The CVPLIF lets you mix and match the various signal types passing through a port. One such application is to detect a bias current mismatch. Not all biases with the same nominal value are created equal; some can drift a lot over temperature while others drift only a little. A sloppy 50uA bias generator accidentally connected to a sensitive 50uA bias consumer would likely escape detection until the customer measured performance (noise figure for example) over temperature. The ability to easily pass multiple numbers between modules offers an easy detection mechanism. Along with the pseudo-electrical signal representing nominal current we send a dimensionless signal representing the grade of the bias. The bias consumer not only checks the nominal value, it also checks the grade. If either falls out of range the consumer model shuts down the output, thereby creating an obvious symptom of the connectivity bug. The break in signal flow is easy to find. Once the affected block is identified, a survey of it’s internal “health variables” quickly pinpoints the problem. A similar approach can be used to distinguish quiet and noisy power supplies. Since the bias grade is dimenstionless, it also creates a check for multiple bias generators accidentally connected to a single bias consumer because paralleled dimensionless drivers contend like voltage drivers.

4. CONCLUSION

We verified functionality of a complete SOC using the CVPLIF to model analog blocks in Verilog. The resulting run times supported broad coverage; real number traffic let us verify the complete end-to-end signal path in detail; and portability kept the project on schedule because each design team could apply their specific design tools to a common set of models. The above factors allowed the team to find and fix well over 100 functional bugs at various levels before tape out. Aside from three fairly minor bugs the first silicon was fully functional.

Although the main motivation behind our modeling methodology was functional verification, the ability to work with baseband equivalent models also makes it easy to model common impairments such as IQ mismatch, phase noise, thermal noise, and common non-linearities (like AM/AM and AM/PM conversion) in a Verilog model of the RF/analog subsystem.

5. ACKNOWLEDGMENTS

I extend my most sincere thanks to the following people for their verification work on the SOC chip: Chan Hong Park, Paul Dankoski, Subhash Sangam, Zohaib Usmani, Craig Tainsky, Sandeep Komirishetty, Adithya Mopuri, Jeongsik Yang, and Jinux Kim. I also thank Jonathan David, Jim Caravella, Mike Laisne, George Crouse, Shoufeng Mu, Christian Holenstein, Xiang Li, and the rest of the Qualcomm rfa_modelers alias for many rigorous and vigorous discussions on verification and modeling. I thank Beomsup Kim for his support and the opportunity to drive this methodology to fruition. I would especially like to thank Henry Chang of Designers Guide Consulting for developing the custom PLI function and for his extraordinary support of our aggressive schedules.

6. REFERENCES

[1] Razavi, Behzad. RF Microelectronics. Prentice Hall. 1998.

[2] Hadjichristos, A. et al. Single-Chip RFCMOS UMTS/EGSM Transceiver with Integrated Receive Diversity and GPS. IEEE Custom Integrated Circuits Conference. 2008.

[3] Stolzman, Richard. Understanding Assertion-Based Verification. EDA Tech Form. http://www.edadesignline.com/showArticle.jhtml?articleID=192200468

[4] Chen, J. E. Modeling RF Systems. http://designers-guide.org/Modeling/modeling-rf-systems.pdf

[5] Joeres, S. Groh, H. Heinin, S. Event Driven Modeling of RF Front Ends. IEEE Behavioral Modeling and Simulation Conference. 2007.

[6] Sheffler, Thomas. Design of a Switch-Level Analog Model for Verilog. IEEE Behavioral Modeling and Simulation Conference. 2008.

[7] Joeres, Stefan. Heinin, Stefan. Functional Verification of Radio Frequency SoCs Using Mixed-Mode and Mixed-Domain Simulations. IEEE Behavioral Modeling and Simulation Conference. 2006.

[8] Park, R.H. Two-Reaction Theory of Synchronous Machines. AIEE Transactions, Vol 48, pages 716-730, (1929).