# Efficient Functional Verification for Mixed Signal IP

Jonathan David
Cadence Design Systems, Inc
2655 Seely Ave., Bldg 8
San Jose, CA 95134
(408) 894-2646
j.david@ieee.org

## ABSTRACT

In this paper, we describe a methodology for verifying the functional operation of a Mixed-Signal Circuit Block to be used in a larger integrated circuit. Methodology requirements are outlined, including: compatibility with digital verification at the chip level, compatibility with circuit analysis at the block level, and compatibility with project schedule and resource availability. The methodology is described with examples on a case-study using a 10/100 Ethernet Physical Layer implementation. Results from the Case Study show the benefits of applying the methodology to future Mixed-Signal design projects.

## 1. INTRODUCTION

The push to integrate Analog and Mixed-Signal circuits with ASIC designs is influenced by a number of factors; Adoption of a System-on-Chip design approach for fast turn-around of High Value ASICs, increasing use of serial signaling interfaces like Ethernet,[1] Serial-ATA, PCI-Express™[2], and FB-DIMM[3], demand for smaller, lighter, cheaper, faster or lower-power end units, and application of other custom circuits[4] to increase system throughput. While the benefits can be high, integration puts at risk not only the high costs of a mask set for nanometer-scale process geometries, but weeks of bench-test and debug, re-design and re-verification, and fabrication schedule, for an elapsed time that can make the difference between profitable venture or missed opportunity. Mixed-signal design verification methodology improvements have the potential for a large return on the invested effort.

The "Digital Revolution" has been enabled by an ability to describe designs at a high level of abstraction[8, 11], and then use these descriptions to generate the detailed design automatically. This has enabled a clear distinction between Functional Verification (Will my circuit perform the expected functions?) and other performance analysis (will the circuit Run at the desired speed?, have a high enough yield?, low enough power?, etc.) that are used to judge the quality of a specific design implementation.

Analog Design, continues to focus on creative circuit topology, and the optimization of circuit parameters for best performance. The distinction between "function" and "performance" is rarely a clear one.

### 1.1 Mixed Signal IP Verification Requirements

In determining the requirements for a Mixed Signal Verification methodology, we are, here, intending to focus on the issue of correct function, but considering performance where appropriate to the analog side of the design. Considering the mixed nature of this effort we make the first two requirements:

- Compatibility with the System (full-chip) Design Verification Methodology.
- Compatibility with the Circuit Design Analysis Methodology

The first will require us to use, and validate, a digital-HDL model of the mixed signal block for use in system verification efforts. In addition, for top-level interface, we will primarily use the same methods (Bus Functional Models or Transaction Level Models) for describing the external digital interfaces. An advantage of this is the possibility for reuse of models already in use for the digital verification effort.

The second will require working with "schematic" data directly wherever possible, and that analysis results from the mixed signal analysis tool can be correlated to the circuit analysis tool, and re-processed by the same analysis tools already in use. Also, generation of additional data, or extra simulations should not interfere with circuit design and analysis efforts. Finally, as the digital designers want to know that the RTL is correct, the analog designers will want the ability to ensure that all their functional requirements have been verified.

The final requirement will be more related to project management and scheduling:

- Compatibility with the Project Schedules and Resource availability.

There are a number of implications we can determine from this. Verification runs shouldn't tie up hardware needed for other work indefinitely, and they shouldn't take so long as to jeopardize the project delivery date. Practically this will require: a limit on run time, preference for a number of short runs over a single long run, predictability of total resources needs, and minimization of engineering interaction is to complete the effort.

### 1.2 Verification Challenges

While there are a number of difficulties to be overcome in this effort, there are two major ones that we address here: simulation run time, and the need for significant engineer interaction in circuit analysis.

The "run-time" challenge is due to the exponential degradation of circuit analysis engine performance with circuit size. For SPICE[7] class solvers, the exponent of degradation is around 2.

For the newer class of FastSpice solvers, it is significantly lower, at a potential price of reduced accuracy, and a large dependency on the nature of the circuit. With the large scale of the IP blocks being used today (10-50K transistors in a High Speed SERDES), and the complicated test patterns that must be run, completing a run in a reasonable time frame is a challenge that must be addressed.

The role of the engineer in circuit analysis is due to the nature of the design problem, with most designers making a large number of simulation runs as they converge on a near-optimal circuit. Attempts to address this [5] have typically depended on the circuit designer learning to write code in a programming language. While advantageous for a bookkeeping of circuit verification efforts, few circuit designers look forward to coding opportunities. While coding will be required in the proposed methodology, the challenge will be to allow the circuit designer to work with familiar tools, and use stored tool setup information in any developed scripts.

## 2. METHODOLOGY OVERVIEW

The proposed methodology has two scopes, each of which is described in a following section. The first scope is the entire Mixed-Signal circuit block under Verification. Here we start the major block (the "top" block for our IP[1] team) verification effort using the proposed Digital (RTL[2]) model, and then extend the same tests to the circuit level by replacing RTL with the transistor level description, for selected blocks. The second scope is at the minor-block level, where additional circuit analyses can be run to verify circuit functionality, and confirm model equivalence. Cell level (digital or analog) verification is not considered here, as it would already be complete, or completed in the same manner as the sub-block level we will describe.

By dividing the verification effort into a number of smaller, shorter simulation runs, the maximum use of spare CPU hours can be made, and both Verification Turn-around time and Designer interference minimized. At the top level this is accomplished doubly by separating tests into separate scripts where possible, and by substituting a limited number of transistor blocks at each time.

For circuit verification we utilize a Verification Environment for analog design with interfaces to preferred simulation and analysis tools. Providing specification checking, comparison between circuits and models, and a perl[3] scripting ability this allows the verification engineer to make use of designer simulation setups.

In each case, scripts are developed for each set of simulations to be run, and these scripts are re-run as the design effort progresses, with a report set summarizing the script success or failure to complete, the success or failure of the test it performs, and pointing to detailed report information on the test run in case debugging is needed.

[1] IP – Intellectual Property; as "IP" a design block may be made available for integration into more than one ASIC or SOC.

[2] RTL – Register-Transfer-Level; digital HDL that is limited to constructs that can be automatically synthesized. This is preferred for digital verification as it allows compilation to hardware accelerators for design debug and frequently for testing software to be run on a system.

[3] Perl – Practical Extraction and Reporting Language

## 2.1 TOP-LEVEL VERIFICATION

The method proposed here for Top-Level verification has been used in digital verification for a long time, namely replace one block at a time with its detailed description, and re-run the top level tests. The challenge here is that the detailed descriptions are now analog circuits, and in some cases will need analog signals from other blocks that we wish to leave at a "behavioral" representation. Since the Verilog-AMS[9] language allows for automatic insertion of connections at design time, it may be possible to leverage that capability to reduce the number of Analog behavioral models needed to support the Mixed-Signal Verification.

### *2.1.1 Common Interface types*

Most signal interfaces can be classified into three types, voltage, current, and "balanced". A consideration of each of these will help determine which circuit blocks will require modeling, and what type of models will be needed.

#### *2.1.1.1 Voltage interface*

A Voltage interface is typified by a low driving impedance, and a High receiving resistance, such as is typically seen in standard MOS logic circuits. Where a behavioral model represents logic 1's and 0's – namely High and Low voltages, mapping between logic value and analog values is fairly accurate, if a few fixed parameters are considered, namely Vhigh, Vlow, Vthresh, Tr, Tf, Rout, Cin, and delay. Since this is a single wire interface, it is relatively simple to determine which representation of the signal is needed at each point and provide the right one. For this reason, on most interfaces between analog and "real" logic models, the automatically inserted models will suffice, at least for the purpose of functional verification. One drawback for this kind of interface in analog circuitry is that the resistance of a long connection wire, taken with the high input resistance at the receiving end, makes for a high sensitivity to noise voltages due to capacitive coupling from adjacent signals.

#### *2.1.1.2 Current interface*

The current interface is the classic solution to the coupled noise problem. For static references, the preferred signaling method is to use a low input impedance circuit, driven by a high output impedance. The common implementation of this, is the bias current source, driving a diode connected input of a current mirror. The equivalent RTL model will only show if the current signal is present or not. Even if only one reference current value is generated, one must still determine if the signal is of a source type or a sink type. To provide an automatic insertion connection for these signals would require definition and characterization of as many biases as are present in the design, and adding custom discipline types to all the affected RTL models to ensure the correct connections.

In the section on BLOCK-LEVEL VERIFICATION a methodology for calibrating an analog model of the bias block will be shown, that will even allow automatically inserted connections from the bias circuit to a logic model, with an appropriate choice of logic values. While requiring an analog model of a bias block, this will not require a mixed-signal model for the case where 1 bias block drives several other blocks.

### *2.1.1.3 Balanced interface*

The other approach to minimizing coupled noise on a signal path, involves a two-wire interface, typically with both driving and receiving interface impedances matched to the line, at least to some extent. Reserved for more dynamic signals, the nature of this type of signal is not easily representing in a digital model, except by the use of a real variable. In limited cases, like the 10/100 Ethernet interface that is our case study, or the SERDES designs mentioned earlier, an equivalent logic value can be determined for up to 4 values on two wires. Since there is currently no provision for automatic insertion of connect models for a two wire interface, an analog or mixed signal model will be needed.

### *2.1.1.4 Data Converters*

Digital-to-Analog Converters (DAC) and Analog-to Digital Converters (ADC) are examples of a class of blocks that are truly Mixed-Signal in nature, and may need to be modeled in a mixed-signal manner to improve simulation run-times for other blocks..

## *2.1.2 Test planning and partitioning*

Having now identified some cases where an Analog or Mixed-signal model is not needed, we now look at the details of implementing our partitioning strategy. We partition the testing in 3 ways.

1. We implement tests separately, so that each test can be run in parallel on a number of separate CPU's. For example, with the 10/100 Ethernet Phy., we have separate tests for configuration in Full-Duplex 100M mode, Full-Duplex 10M mode, Half-Duplex 100M mode, and Half-Duplex 10M mode. We also have separate Block transmission tests in each mode, in combination, short & long packets, fixed & random patterns.

2. We create multiple configurations of the design, testing a minimal number of blocks at the transistor level in each configuration so that each test can run in a reasonable time. A wrapper module supports the comparison between the design and its model. Behavioral models are used in place of the rest of the design.

3. We create a separate set of configurations for interface checking. Since interface checking will require 2 or more blocks to be at the transistor level, we will limit the testing to that required to validate the interface only, and create one or more separate smaller test for the block itself allowing more detailed or longer tests, as shown in Figure 1.

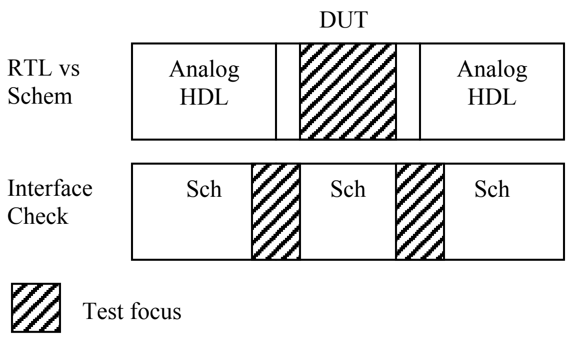

**Figure 1 Partitioning Internal and Interface Tests**

### *2.1.2.1 Test Plan Matrix*

Table 1 shows the Test Plan for a simplified set of configurations and list of tests for the Ethernet Phy. Even in the planning stages, over 50 tests and 30 configurations are proposed, with a requirement to build and check analog or mixed models of approximately 10 design blocks. We eliminate redundant tests, i.e. 10BT tests of 100TX specific circuits. Not shown in the simplified list in Table 2 are a number of sub-blocks in the receiver to be treated separately, due to high transistor counts.

**Table 1 Simplified Text Plan Matrix**

| | **CONF** | | **X_FD100** | | | | **X_FD10** | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Configuration** | **Cnf FD100** | **Cnf FD10** | **X_sml_ones** | **X_sml_rndm** | **X_lrg_ones** | **X_lrg_rndm** | **X_sml_ones** | **X_sml_rndm** | **X_lrg_ones** | **X_lrg_rndm** |
| Functional Verif | X | X | X | X | X | X | X | X | X | X |
| Functional Verif ExtLpBk | X | X | X | | | X | X | | | X |
| **TRANS model ck** | **X** | **X** | **X** | **X** | **X** | | **X** | **X** | **X** | |
| **TRANS IF ck Int** | **X** | **X** | **X** | | | | **X** | | | |
| **TRANS IF ck - pads** | **X** | **X** | **X** | | | | **X** | | | |
| BIAS model ck | X | X | X | | | | X | | | |
| BIAS IF ck TRANS | X | X | X | | | | X | | | |
| BG model ck | X | X | X | | | | X | | | |
| BG IF chk | X | X | X | | | | X | | | |
| RCVR model chk | - | - | - | - | - | - | - | - | - | - |
| RCVR IF chk | - | - | - | - | - | - | - | - | - | - |

**Table 2 Simplified Configuration List for Test Planning**

| **Configuration** | **Common_BG** | **Common_PLL** | **Bias_Port** | **Trans** | **AnCtr** | **DigCtr** | **RCVR** | **Channel Out** | **Channel In** | **2nd Phy** |
|---|---|---|---|---|---|---|---|---|---|---|
| Functional Verif | R | R | R | R | R | R | R | R | R | R |
| Functional Verif ExtLpBk | R | R | R | R | R | R | R | R | | |
| **TRANS model ck** | **A** | R | **A** | **X** | R | R | R | M | | |
| **TRANS IF ck Int** | S | R | S | **S** | S | R | R | M | | |
| **TRANS IF ck - pads** | **A** | R | **A** | **S** | R | R | M | S | | |
| BIAS model ck | A | R | X | A | R | R | M | M | | |
| BIAS IF ck TRANS | S | R | S | S | S | R | M | M | | |
| BG model ck | X | M | A | M | R | R | * | M | | |
| BG IF chk | S | S | S | S | R | R | * | M | | |
| RCVR model chk | - | - | - | - | - | R | X* | A | | |
| RCVR IF chk | - | - | - | - | - | R | S* | S | | |

Model Type Legend

| | | | |
|---|---|---|---|
| R | RTL (digital model) | * | Expand subblocks |
| S | Schematic (transistors) | - | Depends on expansion |
| G | Gate (gate level netlist) | A | Analog Behavioral |
| M | Mixed-Signal Behavioral | | |
| X | Compare - Schematic + RTL ref model | | |
| E | Extracted - from layout with parasitics | | |

### *2.1.3 RTL vs. Schematic Comparison*

The Model Check simulations accomplish both Functional Testing of the block checked, and a direct comparison of behavior between the block as designed and the digital model of the block used for System Functional Verification. To perform this checking we use a special wrapper for the block as shown in Figure 2, which contains interface monitors to convert any non-logic interface values.

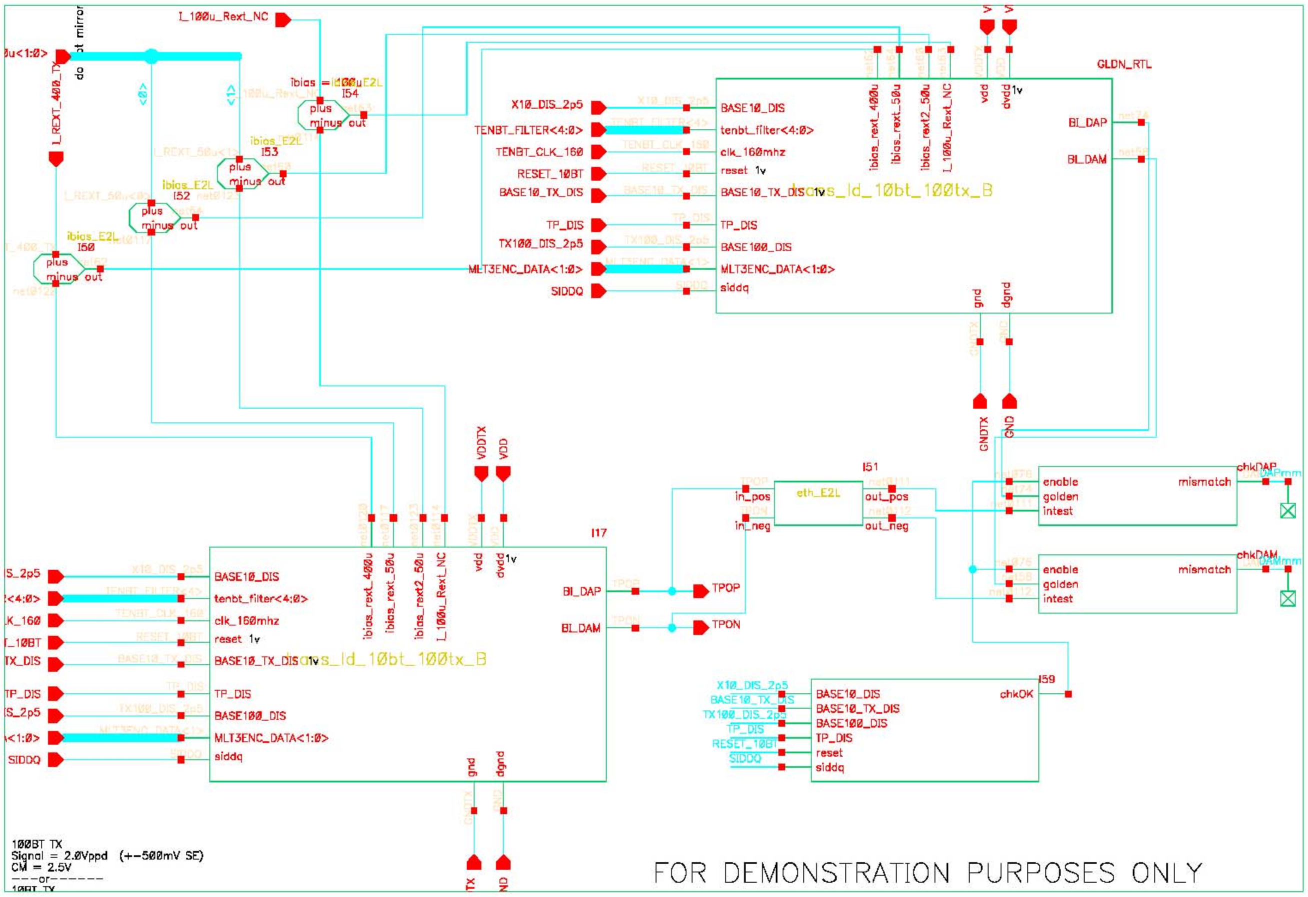

**Figure 2 Example Schematic for RTL vs Schematic Comparison**

Within this view we have 3 Mixed Signal behavioral models which are of interest. The first is a current monitor shown in Listing 1, with an output of the proper value if the current thru is acceptable. Second, shown in Listing 2, we have an interface monitor to convert the transmitter output voltages to the corresponding logic values used in the RTL code. This assumes proper output loading as the monitor provides NO load. Time and voltage tolerance values for edge detection are set at the maximum value consistent with the tolerance in the logic compare module, shown in Listing 3.

The compare module adds a message to the log file in the case of any mismatches. We have a simple logic model that disables the checks when the logical outputs are not expected to match.

**Listing 1 ibias_E2L: Bias Current Interface monitor**

```
// Verilog-AMS HDL for  VFS_AMS_PHY180.ibias50u_E2L:verilogams
// last revised:  07/20/04 jbdavid
`include "constants.vams"
`include "disciplines.vams"
// DEFINE & TIMESCALE :
`timescale 1ns/10ps
//=====================================================
module  ibias_E2L ( // PINS :
  inout plus, minus, // monitor branch
  output out  );       // logic out
// REGISTER and WIRE TYPES
  electrical plus, minus;
  reg out;
// PARAMETERS:  (Comment each one)
  parameter real ibias = 100u; // 100ua default
  parameter real ibias_tol = 0.75 from (0:1);
//---------------------------------------------------------------------------
  always @(above(I(plus,minus) - ibias*ibias_tol, 1n, 1u)) out = 0;
  always @(above(ibias*ibias_tol - I(plus,minus), 1n, 1u)) out = 1;
//---------------------------------------------------------------------------
  analog begin
    V(plus,minus) <+ 0;
  end
endmodule
```

**Listing 2 eth_E2L: Ethernet Interface Monitor**

```
// Verilog-AMS HDL for  VFS_AMS_PHY180.eth_E2L:verilogams
// last revised:  07/20/04 jbdavid
`include "constants.vams"
`include "disciplines.vams"
// DEFINE & TIMESCALE :
`timescale 1ns/10ps
```

```
module  eth_E2L (
   output out_neg, out_pos,
   input in_neg, in_pos           ); // end of port declarations
// REGISTER and WIRE TYPES
   logic out_pos;
   electrical in_pos;
   logic out_neg;
   electrical in_neg;
   reg inneg, inpos;
// INTERNAL NODES :
   assign out_pos = inpos;
   assign out_neg = inneg;
// PARAMETERS:  (Comment each one)
   parameter real vth = 0.5; // diff input threshold for detection
//---------------------------------------------------------------------------
   always @(cross(V(in_pos,in_neg) - vth,1, 100p, vth*0.1)) begin
      inpos = 1; inneg = 0;
   end
   always @(cross(V(in_pos,in_neg) - vth,-1, 100p, vth*0.1)) begin
      inpos = 0; inneg = 0;
   end
   always @(cross(V(in_pos,in_neg) + vth,1, 100p, vth*0.1)) begin
      inpos = 0; inneg = 0;
   end
   always @(cross(V(in_pos,in_neg) + vth,-1, 100p, vth*0.1)) begin
      inpos = 0; inneg = 1;
   end
endmodule
```

**Listing 3 logic_sig_compare: Reports Signal Mismatch**

```
// Verilog-AMS HDL for  VFS_AMS_PHY180.logic_sig_compare:verilogams
// last revised:  07/20/04 jbdavid
`include "constants.vams"
`include "disciplines.vams"
`timescale 1ns/10ps
//===================================================
 module  logic_sig_compare ( // goes here
   input golden, intest, enable,
   output mismatch       ); // end of port declarations
// INTERNAL NODES :
   reg faild, enabled;
   assign mismatch = enabled&&(golden^intest);
// PARAMETERS:  (Comment each one)
   parameter real tcheck_ns = 3.0; // time between mismatch and faild
   parameter real endelayLH = 0; // ns
   parameter real endelayHL = 0; // ns
// LOCAL VARIABLES:  (Comment each one)
   real eventstart;
//---------------------------------------------------------------------------
   initial faild = 0;
   always @(posedge enable) #(endelayLH) enabled = 1;
   always @(negedge enable) #(endelayHL) enabled = 0;
   always @(posedge mismatch) begin
      eventstart = $realtime;
      #(tcheck_ns) if (mismatch!==0)  begin
          faild = faild + 1;
          $display("SPECFAIL %m: @ %d GLD: %b TST %b",
                   eventstart, golden, intest);
      end
   end
endmodule
```

## 2.2 BLOCK-LEVEL VERIFICATION

The primary analog model needed to complete the Transmitter block Verification is for the Bias_port block. Here we define a model for a current mirror, where the Reference input voltage as a function of input current (the diode curve for the diode connected Mos) is captured in a look-up table. On the output side we model the current gain dependency on the output voltage.

To capture the data for this lookup table we run two simulation sweeps, one sweeping the input current to generate the diode curve, the other sweeps the load voltage to capture the current gain, down into the triode region of the output device. If done to a zero volt difference, this will ensure that the table will contain a 0 gain entry for the case where the Source-Drain voltage of the output is 0, as it would be if the current mirror is driving a standard E2L interface element. This allows us to know that for a bias source, the correct logic value for "on" is 1, as the output is referenced to the supply. For a bias sink, "on" = 0. In the case of the architecture used by this team, when the powerdown state is asserted (siddq = 1) the outputs are clamped to remove any voltage from the receiver, ground for a bias source, and supply for a bias sink. This feature has not yet been added to the model shown here.

While most simulation environments will measure and report the desired values, turning this data into a specification report, and lookup files has in the past typically required programming expertise, and hundreds of lines of code such as that previously presented by this author[5]. However the Virtuoso Specification Driven Environment[12] used for this second phase allows specification limits to be associated with measures, and is easily configured to build the look up tables required for this model. Once all the desired tests, sweeps and calibrations are developed, their execution can be exported to a perl script, and added to the design teams regression suite.

**Listing 4 Bias_port vloga view (simplified)**

```
// VerilogA for VFS_AMS_PHY180, Bias_port, vloga
// last revised:  8/07/04 jbdavid
// DESCRIPTION   :
//  this is a calibrated behavioral model
//
// LIMITATIONS   : no output clamp on SIDDQ
`include "constants.vams"
`include "disciplines.vams"
//=======================================
module Bias_port(   I_REXT_REF100,   SIDDQ,  VDD,
   VSS,    I_REXT_100,    );  // PINS :
   output [1:0] I_REXT_100;
   input       I_REXT_REF100;
   input       SIDDQ, VDD, VSS;

   electrical [1:0] I_REXT_100;
   electrical      I_REXT_REF100;
   electrical      SIDDQ, VDD, VSS;
// PARAMETERS:  (Comment each one)
// Model calibrated at '20:48:06' on 'Sat Aug 7 2004' by 'jbdavid'
   parameter real VdsatIN = 1.013308; //VdsatIN=0.5
   parameter real VthSIDDQ = 1.2;
   parameter real VminOut = 0.7;
   parameter real Roffext = 1.000571e+12; // Roffext = 1G
   parameter real Cdsrext100_1 = 2.698685e-13 ; // Cdsrext100_1 = 200f
   parameter real Cdsrext100_0 = 2.698685e-13 ; // Cdsrext100_0 = 200f
// LOCAL VARIABLES:  (Comment each one)
   integer siddq; // = 1(true) if siddq > vth other wise 0
   real Iref_rext, Vdd; // variable to minimize use of access functions
   real Hext100_1 ;
   real Hext100_0 ;
   real vdsatin;
```

```
//-----------------------------------------------
   resistor  #(.r(Roffext)) RinIext        (I_REXT_REF100,  VSS);
   capacitor #(.c(Cdsrext100_1)) C8  (I_REXT_100[1],  VDD);
   capacitor #(.c(Cdsrext100_0)) C9 (I_REXT_100[0],  VDD);
//---------------------------------------------------------------------------
   analog begin
      siddq = V(SIDDQ,VSS) > VthSIDDQ?1:0; // "bias" signal, dont need edge
      vdsatin = VdsatIN;
      if (!siddq) begin
         V( I_REXT_REF100  ,VSS) <+ vdsatin;
      end
      Iref_rext  = I( I_REXT_REF100  ,VSS);
      Hext100_1 = $table_model( V(VDD,I_REXT_100[1]),
            "Hext100_1.tbl", "1CL");
      I(VDD,  I_REXT_100[1]) <+ Iref_rext*Hext100_1;
      Hext100_0 = $table_model( V(VDD,I_REXT_100[0]),
            "Hext100_0.tbl", "1CL" );
      I(VDD,  I_REXT_100[0]) <+ Iref_rext*Hext100_0;
      I(VDD , VSS) <+ 3* Iref_rext;
   end
endmodule
```

## 3. CASE-STUDY RESULTS

For a completed 10/100 Ethernet Phy. project in 0.18u CMOS, the following simulations were run for estimates of the total verification effort, using three different settings for the analog solver. The first uses the Spectre® solver, the second adds acclerated model evaluation to the spectre solver, and the third uses the Ultrasim™ FastSpice solver.

**Table 3 Top Level Run Results**

| Configuration | Trans Count | Ams (spectre) | Ams (Accel.) | Ams (Ultrasim) |
|---|---|---|---|---|
| Functional Verif | 0 | 2.75 min | 2.75 min | -- |
| Functional Verif (schematics) | | -- | -- | 31 hr |
| TRANS model ck | 547 | 64.8 hr | 19.7 hr | 3.5 hr |
| TRANS IF ck Int | 1126 | 114.7 hr | 37.8 hr | 4.5 hr |

VSdE[12] was used to calibrate the behavioral models, with the work estimates and run times shown in Table 4. The testplan provides both block level specification test and data collection for calibration. Even as improvements in transient solvers allow evaluation of larger circuit blocks in a given time frame, considering that these blocks would be used in over 50 tests for this design, even the 22% speed up provided with the FastSpice solver is significant. We expect reuse of models and calibration setups to provide additional productivity improvement.

**Table 4 Block Level Modeling and Calibration Development**

| Block | Modeling | Testplan | Runtime |
|---|---|---|---|
| Bias_port | 12 hours | 8 hours | 20 min |
| Common_BG | 12 hours | 12 hours | 20 min |

After accounting for all the tests and configurations, we project this methodology would provide at least a 7x reduction in overall simulation time with the Spectre solver, and 3x with the faster Ultrasim solver. It also increases the capability for parallel operation. These benefits justify the investment in model development and calibration setup.

## 4. CONCLUSIONS

A methodology for Mixed Signal functional verification has been developed which provides functionally verified digital models to the system verification effort, and Functional & Performance Verification reports to the mixed signal design team. The time to complete each separate test has been minimized through the partitioning of the transistor level tests, with a small reduction in total CPU hour requirements, but a great increase in opportunities to reduce the total cycle time thru parallelism, and use of spare cycles on existing compute hardware. The effort to create, validate, and calibrate analog or mixed signal models where required, is shown to be a worthwhile investment.

## 5. ACKNOWLEDGMENTS

My thanks to a number of coworkers at Cadence for assistance with this project, review of the paper and the example design database.

## 6. REFERENCES

[1] J. Yang, J. Kim, S. Byun, C. Conroy, B. Kim, "A Quad-Channel 3.125GB/s/ch Serial-Link Transceiver with Mixed-Mode Adaptive Equalizer in 0.18um CMOS", *ISSCC Dig. Tech. Papers*, pp 176-177, Feb 2004.

[2] "PCI Express™ Base Specification Revision 1.0a", PCI-SIG, www.pcisig.com, April 2003.

[3] H. David, M. McTague, "Fully Buffered DIMM (FB-DIMM) Design Considerations" presented at *Intel Developer Forum*, developer.intel.com, Feb 2004

[4] D.J. Deleganes, M Marany, G. Geannopoulos, K. Kreitzer, A.P. Singh, and S. Wijeratne, "Low-Voltage-Swing Logic Circuits for a 7GHz X86 Integer Core", *ISSCC Dig. Tech. Papers*, pp 154-155, Feb 2004.

[5] J. David, "Functional Verification of a Differential Operational Amplifier", *Proc. Intl. Cadence Usergroup Conference 2001*, paper F50, Dec 2001

[6] J. Vandenbussche, G. Gielen, M. Steyart, *Systematic Design of Analog IP Blocks*, Boston:Kluwer, 2003

[7] L. W. Nagel, "SPICE2: A Computer Program to Simulate Semiconductor Circuits," University of California, Berkeley, Memo. no. ERL-M250, May 1975.

[8] Y.Chu, D. L. Dietmeyer, J. R. Duley, F. J. Hill. M. R. Barbacci, C. W. Rose, G. Order, B. Johnson, and M. Roberts, " Three Decades of HDLs – Part I: CDL through TI-HDL," *IEEE Design Test Comput*., vol. 9, pp 69-81, June 1992.

[9] K. Kundert, O. Zinke, *The Designer's Guide to Verilog-AMS*, Boston:Kluwer, 2004

[10] Dan Fitzpatrick, Ira Miller, *Analog Behavioral Modeling with the Verilog-A Language* Boston: Kluwer, 1998

[11] Samir Palnitkar, *Verilog HDL* Mountain View: SunSoft, 1996

[12] *Virtuoso® Specification-driven Environment User Guide, Product version 4.1* San Jose: Cadence Design Systems, 2004 sourcelink.cadence.com