Radio Receiver Mixer Model for Event-Driven Simulators to support Functional Verification of RF-SOC Wireless Links

Jonathan B. David, *Senior Member, IEEE*

***Abstract*—The baseband-equivalent approach is used to derive a SystemVerilog model of a double-balanced receiver mixer to support functional verification of the transceiver block in a wireless communications integrated circuit. Results using the transceiver model at the chip verification level are reported.**

***Index Terms*—baseband-equivalent, behavioral modeling, design verification, Mixed analog-digital integrated circuits Transceiver, wireless communications.**

I. Introduction

HIGH-VALUE consumer devices require low-power wireless connectivity in order to interact with content or databases stored on the network, and to connect to audio input/output channels for hands-free operation. Device designers demand integration in order to keep the component and manufacturing costs manageable, as well as to trade device size, weight and power for increased functionality. While the "digital revolution" continues to take on functionality from the analog and RF circuitry for these links, integrating the DSP and Radio in the same chip requires a functional verification approach that is compatible with both the DSP and RF/Analog circuits in the Radio, employing real-number modeling in an event-driven simulator.

While the majority of blocks in the radio circuitry can be modeled as signal addition, multiplication, variable gain, or data conversion in a straight forward manner, applying this approach to the receiver mixer will result in a model that must process the signals many times per cycle of the RF carrier [9-13], as well as require an appropriate base-band filter. This reduces the simulation run speed significantly compared to verification of the baseband processor by itself.

Applying a baseband-equivalent approach to the RF signals[2,3], at the cost of a more complicated mixer model [6,7] results in significantly improved verification run times, enabling verification of the full wireless-link signal path including both DSP and Radio, allowing multi-radio RF-SOC's to be designed with re-spins only for parametric tuning.

Manuscript received August 8, 2010.

J. B. David is with the QCT Division of Qualcomm, Inc, 3165 Kifer Rd. Santa Clara, CA 95054 USA phone: 408-216-4304; fax: 408-533-9632; e-mail: jbdavid@qualcomm.com.

While significant work on modeling RFIC circuits exists[14,16-19], most of it focuses either on fast behavioral models to support system design[5,7,8], or behavioral models supporting spectral analysis with noise and distortion to support circuit analysis[6,9,10]. To support functional verification[4], circuit performance metrics are not required, while simulation speed is of critical importance. Thus the only non-idealities included in verification models are those for which a functional adjustment is designed in, either in the baseband digital processor, or by way of a digitally controlled calibration loop.

A. Approach

Following a description of a typical wireless-link block diagram, and an overview of the baseband-equivalent signal representation, the model mathematics will be derived in section II, and the baseband-equivalent receiver mixer model will be introduced in section III. In section IV, the test programs and simulation results will be presented. The final section will summarize the experience applying this model to the verification of an RF-SOC.

B. Baseband Equivalent Signal Representation

A modulated signal can represented in general form as

$$S_{RF} = re\{[I(t) + jQ(t)]e^{j\omega t}\}, \text{(where } \omega = 2\pi F) \qquad (1)$$

which can also be represented (thru the use of trigonometric identities) as

$$S_{RF} = I(t)\cos(\omega t) - Q(t)\sin(\omega t). \qquad (2)$$

This form gives insight to the double-balanced mixer used in a Quadrature Amplitude Modulation (QAM) transmitter, where I & Q analog signals are mixed with 90° out-of-phase oscillator signals (cos(wt) & -sin(wt) respectively to create the RF signal being transmitted. This ideal signal is entirely determined at any point in time by the triple (I, Q, freq). In analyses where the frequency is not changing, (e.g. system-level design studies[7]) this can be simply be reduced to the complex pair (I + jQ), where we have moved entirely to the frame of reference of the carrier. Since programming the frequency is an important part of the transceiver functionality, we keep the frequency component for verification purposes.While the primary signal components are I, Q and frequency, it is helpful in some verification contexts to add a

couple of other bookkeeping terms to the signal vector; one for bandwidth and one for the DC component of the voltage or current. This results in the entire signal vector consisting of 5 items {I, Q, freq, BW, DC}. This vector is carried between verilog models using the "hyperwire" technique in [1].

Its important to note that although other modulation schemes (FM, FSK, PSK, GFSK DQPSK, 8DPSK etc) have their own "natural" frame of reference and notation, converting them to this format is straightforward, allowing development of a generic mixer model, suitable for all direct conversion radio architectures.

C. Typical Wireless Link Diagram

The simplified block diagram for a direct-conversion wireless link transceiver is shown in Figure 1.

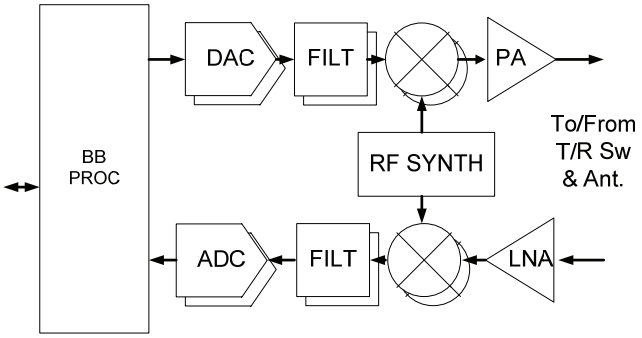

Figure 1 Wireless Link Block Diagram

For functional verification purposes, the model of the DAC and ADC can usually simply convert between real number signals on the analog side and digital logic values on the digital side. While the filters may include a gain component that affects the signal path, the correct setting for filter bandwidth controls can be verified by adding a bandwidth component to the RF signal vector. This will be generated by the TX filter and checked for compatibility in the RX filter. This bandwidth component is also used in the mixer model to determine if the LO/RF frequency separation is too large for valid signal reception.

Both the PA and LNA are modeled simply as gain elements with the gain being applied only to the I & Q parts of the RF signal vector, unless there is some impairment required by functional verification.

II. Model Mathematical Derivation

Tom Lee [15] puts the fundamental theory so succinctly it's worth quoting directly:

"At the core of all mixers presently in use is a multiplication of two signals in the time domain. The fundamental usefulness of multiplication may be understood from examination of the following trigonometric identity:

$$(A\cos\omega_1 t)(B\cos\omega_2 t) = \frac{AB}{2}[\cos(\omega_1-\omega_2)t + \cos(\omega_1+\omega_2)t]\text{"} \quad (3)$$

Here we represent one the signal being transmitted as A cos (ω_1t) and the carrier as B cos (ω_2t), taking the product gives us the sum and difference terms of the two frequencies. This is basic Amplitude Modulation (AM) as the amplitude of the first is "carried" on the second.

Where *F* varies with time, we make the substitution from the constant frequency case, $\omega t = \int_0^t \omega\, dt = \omega \int_0^t dt$, which is the instantaneous phase of the signal.

$$\phi = \int_0^t \omega\, dt = 2\pi \int_0^t F dt\,. \quad (4)$$

With this definition, let's take another look at the AM signal mixing action we had in equation 3.

Here we'll use *A(t)* as the baseband signal, the carrier as cos φ_{RF}. In our receiver we'll mix this with our Local Oscillator, we'll call cos φ_{LO}. So now equation 3 becomes

$$(A\cos\phi_{RF})(\cos\phi_{LO}) = \frac{A}{2}[\cos(\phi_{RF1}-\phi_{LO}) + \cos(\phi_{RF}+\phi_{LO})] \quad (5)$$

Down converting mixers will have a filter to select the difference term and reject the sum term, so the output of the mixer is just the product of a baseband term and of the cosine of the difference in phase of the two carriers, a quantity which can be determined by integration of the instantaneous frequency over the course of an analysis, starting from some zero or known phase reference point.

$$\Delta\phi = \phi_{RF} - \phi_{LO} \quad \& \quad \Sigma\phi = \phi_{RF} + \phi_{LO}$$

$$(I\cos\phi_{RF} - Q\sin\phi_{RF})(\cos\phi_{LO}) = \frac{I}{2}M'_{11} - \frac{Q}{2}M'_{12}$$

$$(I\cos\phi_{RF} - Q\sin\phi_{RF})(-\sin\phi_{LO}) = -\frac{I}{2}M'_{21} + \frac{Q}{2}M'_{22} \quad (6)$$

$$M'_{11} = \cos(\Delta\phi) + \cos(\Sigma\phi)$$
$$M'_{12} = \sin(\Delta\phi) + \sin(\Sigma\phi)$$
$$M'_{21} = -\sin(\Delta\phi) + \sin(\Sigma\phi)$$
$$M'_{22} = \cos(\Delta\phi) - \cos(\Sigma\phi)$$

Keeping only the difference terms we get:

$$I_{REC} = \frac{I}{2}[\cos(\Delta\phi)] - \frac{Q}{2}[\sin(\Delta\phi)]$$
$$Q_{REC} = \frac{I}{2}[\sin(\Delta\phi)] + \frac{Q}{2}[\cos(\Delta\phi)] \quad (7)$$

Should a phase imbalance be required for calibration or baseband processor verification, a constant term can be added to M_{11} and M_{12} and subtracted from M_{21} and M_{22}. Gain and phase imbalance[14] and DC offset impairments are also easily added if needed.

Modeling a receiver requires only integration of the frequency components, and a little real algebra. Next we'll translate this into SystemVerilog.

III. Model Description

The model core is written in SystemVerilog [20] and uses real ports to pass values two and from the wrapper. Aside from the algebra, there is a check of the frequency difference between the LO and RF inputs, and the bandwidth input, this function will null the output of the model if the difference is larger than the bandwidth.

Rather than use an initial block to initialize the model, this does all initialization at the rising edge of the enable signal. This ensures that all critical variables are reinitialized correctly, even if low power simulation is enabled, and the mixer is "powered down" with all internal variables set to the unknown state.

A negative frequency on the LO_Q input is used to indicate that the Phase Rotation of the LO is inverted, and the Q input leads the I input, with the consequent sign changes in the Matrix terms.

The final always block updates the integration value and recalculates the output variables. The integration assumes stepwise constant frequency terms, and so uses the prior values of the frequency to calculate the integral of each completed time step, in case a new frequency value triggered the update.

Cadence' Incisive Simulator version 9.2 used for these simulations does not yet support the math utility functions added SystemVerilog [20] in 2009, so a library of similar C functions has been compiled and linked into the simulator via the verilog PLI interface. The "qc_" prefix has been used for each of these functions to prevent conflicts with future versions of simulators that incorporate these math functions. It is noted that the list of math functions in [1800-2009] is missing an absolute value function, which it is hoped will be added in a future version of the standard.

Listing 1 rx_mixer_generic.sv

```
// SystemVerilog HDL for  rfraff.rx_mixer_generic:svlog_real
//------------------------------------------------------------------------------
// LIMITATIONS   :
// DEFINE & TIMESCALE :
`timescale 1ns/1ps
//===============================================
module  rx_mixer_generic_sv ( // goes here
// PORTS : (all input, output, inout declarations)
  input logic enable,   // only logic controls are made as pins
  input var real RXin_linph, RFin_lquad, // BB equiv I and Q
  input var real RXin_freq, RFin_BW, // Fcarrier, Signal  BW
  input var real LOin_Freq_inph, LOin_Freq_quad,
  input var real Gain, // this is the core mixer gain
  output var real BB_I_Iout, // output is current
  output var real BB_Q_Iout, // output is current
  output var real BB_BW // same bandwidth for all 4 pins
); // end of port declarations
//------------------------------------------------------------------------------
// REGISTER and WIRE TYPES
reg Fi_ok, Fq_ok; //status LO input freq and sig BW
// LOCAL VARIABLES:  (Comment each one)
real RXin_freq_last, LOin_Freq_inph_last;
// prior value for integration
real time_last, dt; //time variables for integration
integer  Ndelt; // for modulo integration
real DeltaPhi; // Phase difference between RF and LO
real two_pi, pi, piby2, signQ; // "pi constants" & LO rot direction
real M11, M12, M21, M22; // Matrix relating <I,Q>in to <I,Q>out
real Eff_gain;// gain after BW checks
//------------------------------------------------------------------------------
  always @(posedge enable) begin //initialize model
    RXin_freq_last = RXin_freq;
    LOin_Freq_inph_last = LOin_Freq_inph;
    time_last = $realtime;
    dt = 0;
    piby2 = $qc_asin(1.0);
    pi = piby2*2.0;
    two_pi = 2.0*pi;
    signQ =  (LOin_Freq_quad>0)? 1.0 : -1.0;
    BB_I_Iout = RXin_linph;  // at t0 M11 M22 = 1 M12 m21 = 0
    BB_Q_Iout = signQ*RFin_lquad;
  end
  always @(LOin_Freq_inph, LOin_Freq_quad,
                          RXin_freq, RFin_BW) begin
    Fi_ok = $qc_abs(LOin_Freq_inph-RXin_freq) < RFin_BW;
    Fq_ok =
$qc_abs($qc_abs(LOin_Freq_quad)-RXin_freq) < RFin_BW;
    Eff_gain = (Fi_ok&&Fq_ok)?Gain:0.0;
    BB_BW = (Fi_ok&&Fq_ok)?RFin_BW:0.0;
  end
  always @(LOin_Freq_quad)
            signQ =  (LOin_Freq_quad>0)? 1.0 : -1.0;
  always @(LOin_Freq_inph, RXin_freq,
                RXin_linph, RFin_lquad) begin // update events
    #0 dt = ($realtime - time_last)*1e-9; // timescale is in ns
    if (dt > 0.0) begin // update only if time has moved
    // diff of the integral is the integral of the diff
    // - integrate DeltaF to get DeltaPhase
      DeltaPhi =  DeltaPhi
           + (RXin_freq_last - LOin_Freq_inph_last)*dt*two_pi;
//use Prior values until next delta time
      Ndelt   = DeltaPhi/two_pi; // get 2*pi units
      DeltaPhi =  DeltaPhi - Ndelt*two_pi; // subtract them
    end
    RXin_freq_last = RXin_freq; //update the history vars
    LOin_Freq_inph_last = LOin_Freq_inph;
    time_last = $realtime;

    // matrix factors
    M11     = $qc_cos(DeltaPhi)*Eff_gain;
    M12     = -1.0*$qc_sin(DeltaPhi)*Eff_gain;
    M21     = signQ*$qc_sin(DeltaPhi)*Eff_gain;
    M22     = signQ*$qc_cos(DeltaPhi)*Eff_gain;

    BB_I_Iout   = M11 * RXin_linph + M12 * RFin_lquad;
    BB_Q_Iout = M21 * RXin_linph + M22 * RFin_lquad;

  end
endmodule
```

IV. Simulation Results

The model test uses a sinusoidal stimulus consisting of baseband quadrature sine wave with I leading Q at a frequency of 2 MHz, with data clocked at 100 Mhz. Testing of direct conversion radio blocks is typically done with this type of signal, even when other modulation schemes are used in by the baseband processor. Testing of all PLL control and RX gain modes can be completed with this stimulus prior to use with the baseband processor.

The monitor for the baseband outputs is also clocked at the 100 MHz testbench clock rate. This block processes the baseband signals to determine frequency, peak and RMS signal amplitude, dc value, as well as phase relationship between the I and Q baseband signals. This allows each test to be automated, comparing the RF carrier + baseband frequency, with the LO frequency to determine the expected mixer baseband output frequency and phase rotation.

Listing 2 Mixer Test code

```
initial begin
  $timeformat(-9,3, " ns",10);
  dut_enable = 0;
  RxGain[1] = 3.0;
  RxGain[2] = 3.0;
  RxGain[3] = 3.0;
  RxGain[4] = 3.0;
  RxGain[5] = 3.0;
  RxGain[6] = 3.0;
  RxGain[7] = 3.0;
  RxGain[8] = 3.0;
  RxGain[9] = 3.0;
  RxGain[10] = 3.0;
  xvdd_volts = 1.3;
  #0.01 QcAsserts = 0;
   #2000;
  QcAsserts = `FAILSLOGD;
  for (TestID = 1; TestID <= `TESTMAX; TestID = TestID+1) begin
   case (TestID)
     1 : begin
        `REPORT_RUNPOINT("Frf = Flo", TestID)
        Flo = 2.4e9;       Frf = 2.4e9;    Fbb = 2e6;   LO_IleadsQ = 1;
      end
     2: begin
        `REPORT_RUNPOINT("Frf < Flo", TestID)
        Flo = 2.4005e9;    Frf = 2.4e9;    Fbb = 2e6;   LO_IleadsQ = 1;
      end
     3: begin
        `REPORT_RUNPOINT("Frf << Flo", TestID)
        Flo = 2.404e9;     Frf = 2.4e9;    Fbb = 2e6;   LO_IleadsQ = 1;
      end
     4: begin
        `REPORT_RUNPOINT("Frf > Flo", TestID)
        Flo = 2.4e9;       Frf = 2.4005e9; Fbb = 2e6;   LO_IleadsQ = 1;
      end
     5: begin
        `REPORT_RUNPOINT("Frf >> Flo", TestID)
        Flo = 2.4e9;       Frf = 2.402e9;  Fbb = 2e6;   LO_IleadsQ = 1;
      end
     6: begin
        `REPORT_RUNPOINT("Frf = -Flo", TestID)
        Flo = 2.4e9;       Frf = 2.4e9;    Fbb = 2e6;   LO_IleadsQ = 0;
      end
     7: begin
        `REPORT_RUNPOINT("Frf < -Flo", TestID)
        Flo = 2.4005e9;    Frf = 2.4e9;    Fbb = 2e6;   LO_IleadsQ = 0;
      end
     8: begin
        `REPORT_RUNPOINT("Frf << -Flo", TestID)
        Flo = 2.404e9;     Frf = 2.4e9;    Fbb = 2e6;   LO_IleadsQ = 0;
      end
     9: begin
        `REPORT_RUNPOINT("Frf > -Flo", TestID)
        Flo = 2.4e9;       Frf = 2.4005e9; Fbb = 2e6;   LO_IleadsQ = 0;
      end
     10: begin
        `REPORT_RUNPOINT("Frf >> -Flo", TestID)
        Flo = 2.4e9;       Frf = 2.402e9;  Fbb = 2e6;   LO_IleadsQ = 0;
      end
   endcase
   TESTID = TestID;
   Fexpect = $qc_abs(Frf + Fbb - Flo);
   IleadsQexpect = LO_IleadsQ ~^ (Frf + Fbb > Flo);
   LOsrc.IleadsQ = LO_IleadsQ;
   RXsrc.fm = Fbb/1e6; //src bbfreq is in MHz
   #1000;
   RF_amp = 1.0;
   xvdd_volts = 1.3;
   xvdd.en = 1;
   rfrx_test_rfin_en = 0;
   lotest_en = 1;
   bb_test_vout_en_i = 1;
   bb_test_vout_en_q = 1;
   #1000;
   rfrx_test_rfin_en = 1;
   dut_enable = 1;
   #1000;
   BB_mon_start = 1;
   repeat (20) @(RX_BBMon.ICrossings);
   BB_mon_start = 0;
   BB_Ifreq = RX_BBMon.xIFreq;
   BB_Qfreq = RX_BBMon.xQFreq;
   IQPhasing = RX_BBMon.I_leads_Q;
   FromBBmonI = RX_BBMon.xIPeakAc;
   FromBBmonQ = RX_BBMon.xQPeakAc;
   PeakInput   = RXsrc.peak;
   Radius      = $qc_sqrt(FromBBmonI*FromBBmonI +
                 FromBBmonQ*FromBBmonQ);
   if(PeakInput>0.0) begin
     ActualGain  = (20.0)*$qc_log10(Radius/PeakInput);
          //Gain in dB.
     GainOK  = ($qc_abs(ActualGain - RxGain[TestID])<2.0);
   end
   testpassed = (IleadsQexpect === RX_BBMon.I_leads_Q)
              && ($qc_abs(BB_Ifreq - Fexpect) <2e3)
              && ($qc_abs(BB_Qfreq - Fexpect) <2e3)
              && GainOK;
   $strobe("%s @ %t: RXTEST%02d   %s  Gain:%g dB, Frf: %g Hz,
                Flo: %g   LO:%s Hz, Fbb: %g Hz, BB:%s",
     testpassed?"SPECINFO":"SPECFAIL" ,
```

```
      $realtime,
      TestID,
      testpassed?" Passed :":" Failed :",
      ActualGain, $qc_abs(Frf), $qc_abs(Flo),
      LO_IleadsQ?   "I leads Q": "Q leads I",
       $qc_abs(Fexpect),
        IQPhasing? "I leads Q": "Q leads I");
    if (!testpassed)  QcAssertFailIncrement;
    #1000;
    lotest_en = 0;
    #1000;
    rfrx_test_rfin_en = 0;
    dut_enable = 0;
  end
  #10_000 QcAsserts_SimReporter;
  #1 $finish;

end
```

A study of an early Verilog-AMS version of this model showed a 100x speedup of the model over the equivalent functional passband model where the signal is updated at 40 times/carrier cycle. The pass-band model was implemented with simple signal multiplication followed by a 3rd order discrete time filter to model the Baseband filter.

A variation of this model was used in for the verification of the radio block in a Wireless IC. The verification model of the radio block was then re-used to verify the end-to-end functionality of the wireless-link in a chip-to-chip environment, verifying functionality of bi-directional packet transfer, with most individual verification tests taking less than 2 hours to complete.

Listing 3 Logfile showing test results

```
SPECINFO @ 15388.000 ns: RXTEST01    Passed :
SPECINFO @ 33228.000 ns: RXTEST02    Passed :
SPECINFO @ 47898.000 ns: RXTEST03    Passed :
SPECINFO @ 60648.000 ns: RXTEST04    Passed :
SPECINFO @ 70528.000 ns: RXTEST05    Passed :
SPECINFO @ 85398.000 ns: RXTEST06    Passed :
SPECINFO @ 103228.000 ns: RXTEST07    Passed :
SPECINFO @ 117908.000 ns: RXTEST08    Passed :
SPECINFO @ 130668.000 ns: RXTEST09    Passed :
SPECINFO @ 140548.000 ns: RXTEST10    Passed :
Results: second column
Gain:3.00487 dB, Frf: 2.4e+09 Hz, Flo: 2.4e+09   L
Gain:3.00867 dB, Frf: 2.4e+09 Hz, Flo: 2.4005e+09
Gain:3.00355 dB, Frf: 2.4e+09 Hz, Flo: 2.404e+09
Gain:2.99961 dB, Frf: 2.4005e+09 Hz, Flo: 2.4e+09
Gain:2.98611 dB, Frf: 2.402e+09 Hz, Flo: 2.4e+09
Gain:3.00598 dB, Frf: 2.4e+09 Hz, Flo: 2.4e+09   L
Gain:3.00735 dB, Frf: 2.4e+09 Hz, Flo: 2.4005e+09
Gain:3.00546 dB, Frf: 2.4e+09 Hz, Flo: 2.404e+09
Gain:3.0074 dB, Frf: 2.4005e+09 Hz, Flo: 2.4e+09
Gain:2.98868 dB, Frf: 2.402e+09 Hz, Flo: 2.4e+09
Results: third column
LO:I leads Q Hz, Fbb: 2e+06 Hz, BB:I leads Q
LO:I leads Q Hz, Fbb: 1.5e+06 Hz, BB:I leads Q
LO:I leads Q Hz, Fbb: 2e+06 Hz,   BB:Q leads I
LO:I leads Q Hz, Fbb: 2.5e+06 Hz, BB:I leads Q
LO:I leads Q Hz, Fbb: 4e+06 Hz, BB:I leads Q
LO:Q leads I Hz, Fbb: 2e+06 Hz, BB:Q leads I
LO:Q leads I Hz, Fbb: 1.5e+06 Hz, BB:Q leads I
LO:Q leads I Hz, Fbb: 2e+06 Hz, BB:I leads Q
LO:Q leads I Hz, Fbb: 2.5e+06 Hz, BB:Q leads I
LO:Q leads I Hz, Fbb: 4e+06 Hz, BB:Q leads I
```

Listing 4 QcSimReporter Results

```
      *
       *
        *                  ********
         *        *******
          * ******
           **
Simulation PASSED
```

V. Conclusion

A baseband equivalent functional model of a direct-conversion mixer has been shown. Using this type of model for the receiver mixer in an RF-SOC enables full chip functional verification including interaction between the baseband processor with the on-chip transceiver.

Acknowledgment

The author thanks Jess Chen for many fruitful modeling, methodology and mathematical discussions.

References

[1] Chen, J.E.; , "A modeling methodology for verifying functionality of a wireless chip," *Behavioral Modeling and Simulation Workshop, 2009. BMAS 2009. IEEE* , vol., no., pp.96-101, 17-18 Sept. 2009

[2] Chen, J., "Baseband-equivalent behavioral modeling approach for RF" Tutorial presented at *Behavioral Modeling and Simulation, 2001. BMAS 2001.* Available: http://www.bmas-conf.org/2001/presentations/bmas01-chen.pps.gz

[3] Tuinenga, P.W.; , "Models rush in where simulators fear to tread: extending the baseband-equivalent method," *Behavioral Modeling and Simulation, 2002. BMAS 2002. Proceedings of the 2002 IEEE International Workshop on* , vol., no., pp. 32- 40, 6-8 Oct. 2002

[4] Chang, H.; Kundert, K., "Verification of Complex Analog and RF IC Designs," *Proceedings of the IEEE* , vol.95, no.3, pp.622-639, March 2007

[5] Staszewski, R.B.; Fernando, C.; Balsara, P.T., "Event-driven Simulation and modeling of phase noise of an RF oscillator," *Circuits and Systems I: Regular Papers, IEEE Transactions on [Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on]* , vol.52, no.4, pp. 723-733, April 2005

[6] Gunther, O.; Hartmann, M.; Wenig, P.; Weigel, R.; , "Behavioral Mixer Model for Rapid System Simulation of a 77GHz FMCW-Radar Transceiver," *Microwave Conference, 2007. APMC 2007. Asia-Pacific* , vol., no., pp.1-4, 11-14 Dec. 2007

[7] Jie He; Jun Seok Yang; Yongsup Kim; Kim, A.S.; , "System-Level Time-Domain Behavioral Modeling for A Mobile WiMax Transceiver," *Behavioral Modeling and Simulation Workshop, Proceedings of the 2006 IEEE International* , vol., no., pp.138-143, 14-15 Sept. 2006

[8] Verspecht, Jan; Verbeyst, Frans; Vanden Bossche, Marc; Van Esch, Patrick; , "System Level Simulation Benefits from Frequency Domain Behavioral Models of Mixers and Amplifiers," *Microwave Conference, 1999. 29th European* , vol.2, no., pp.29-32, Oct. 1999

[9] Soury, A.; Ngoya, E.; , "Using sub-systems behavioral modeling to speed-up RFIC design optimizations and verifications," *Integrated*

Nonlinear Microwave and Millimetre-Wave Circuits, 2008. INMMIC 2008. Workshop on , vol., no., pp.165-168, 24-25 Nov. 2008

[10] Root, D.E.; Wood, J.; Tufillaro, N.; , "New techniques for non-linear behavioral modeling of microwave/RF ICs from simulation and nonlinear microwave measurements," *Design Automation Conference, 2003. Proceedings* , vol., no., pp. 85- 90, 2-6 June 2003

[11] Yang, W.; Carter, H.; Yan, J.; , "A high-level VHDL-AMS model design methodology for analog RF LNA and mixer," *Behavioral Modeling and Simulation Conference, 2004. BMAS 2004. Proceedings of the 2004 IEEE International* , vol., no., pp. 125- 129, 21-22 Oct. 2004

[12] Root, D.E.; Verspecht, J.; Sharrit, D.; Wood, J.; Cognata, A.; , "Broad-band poly-harmonic distortion (PHD) behavioral models from fast automated simulations and large-signal vectorial network measurements," *Microwave Theory and Techniques, IEEE Transactions on* , vol.53, no.11, pp. 3656- 3664, Nov. 2005

[13] Verspecht, J.; Gunyan, D.; Horn, J.; Jianjun Xu; Cognata, A.; Root, D.E.; , "Multi-tone, Multi-port, and Dynamic Memory Enhancements to PHD Nonlinear Behavioral Models from Large-signal Measurements and Simulations," *Microwave Symposium, 2007. IEEE/MTT-S International* , vol., no., pp.969-972, 3-8 June 2007

[14] Pui-In Mak; Seng-Pan U; Martins, R.P.; , "A front-to-back-end modeling of I/Q mismatch effects in a complex-IF receiver for image-rejection enhancement," *Electronics, Circuits and Systems, 2003. ICECS 2003. Proceedings of the 2003 10th IEEE International Conference on* , vol.2, no., pp. 631- 634 Vol.2, 14-17 Dec. 2003

[15] Thomas H. Lee, *The Design of CMOS Radio-Frequency Integrated Circuits*, New York, NY: Cambridge University Press, 1998.

[16] Cidronali, A.; Fagotti, R.; Loglio, G.; Magrini, I.; Manes, G.; , "Full-Spectrum Behavioral Model for Linearized Sub-Harmonic Mixer Extracted by Large-Signal Vectorial Measurements," *Microwave Symposium Digest, 2006. IEEE MTT-S International* , vol., no., pp.481-484, 11-16 June 2006

[17] Cidronali, A.; Fagotti, R.; Magrini, I.; Camprini, M.; Manes, G.; , "A Linearized Mixer Model for Reducing the Complexity of System-Level Analysis," *Integrated Nonlinear Microwave and Millimeter-Wave Circuits, 2006 International Workshop on* , vol., no., pp.134-137, 30-31 Jan. 2006

[18] Kraemer, M.; Dragomirescu, D.; Plana, R.; , "Nonlinear behavioral modeling of oscillators in VHDL-AMS using Artificial Neural Networks," *Radio Frequency Integrated Circuits Symposium, 2008. RFIC 2008. IEEE* , vol., no., pp.689-692, June 17 2008-April 17 2008

[19] Kraemer, M.; Dragomirescu, D.; Plana, R.; , "A Nonlinear Order-Reducing Behavioral Modeling Approach for Microwave Oscillators," *Microwave Theory and Techniques, IEEE Transactions on* , vol.57, no.4, pp.991-1006, April 2009

[20] "IEEE Standard for System Verilog-Unified Hardware Design, Specification, and Verification Language," *IEEE STD 1800-2009* , vol., no., pp.C1-1285, 2009