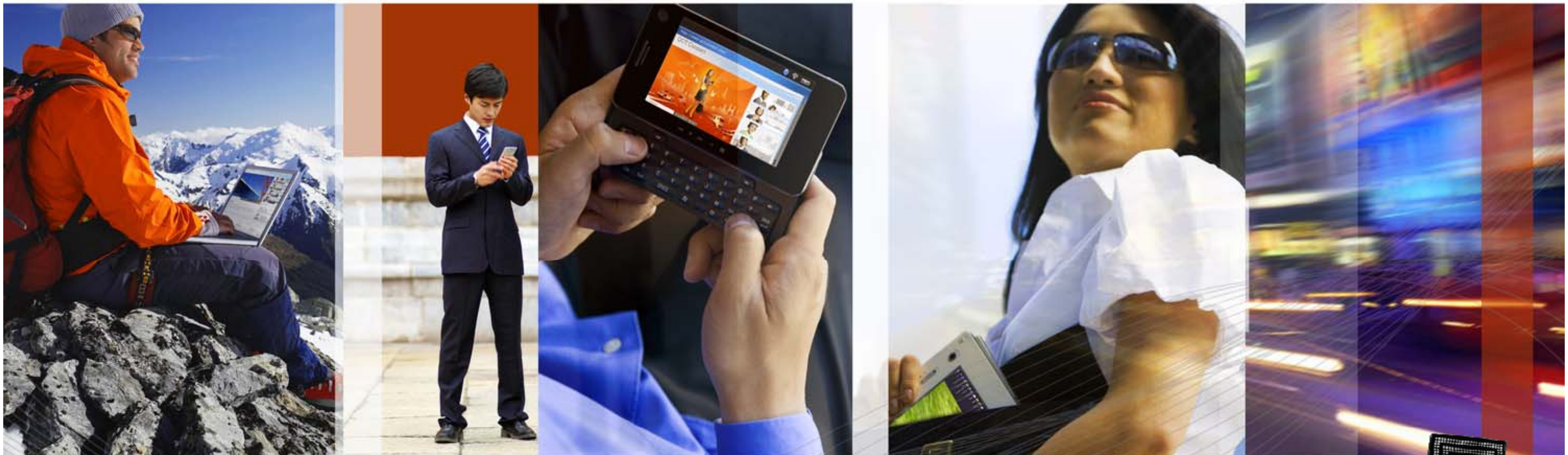


QUALCOMM CDMA Technologies

REDEFINING MOBILITY

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

Jonathan David, SMIEEE, Sr Staff Engr, Qualcomm

IEEE BMAS 2010

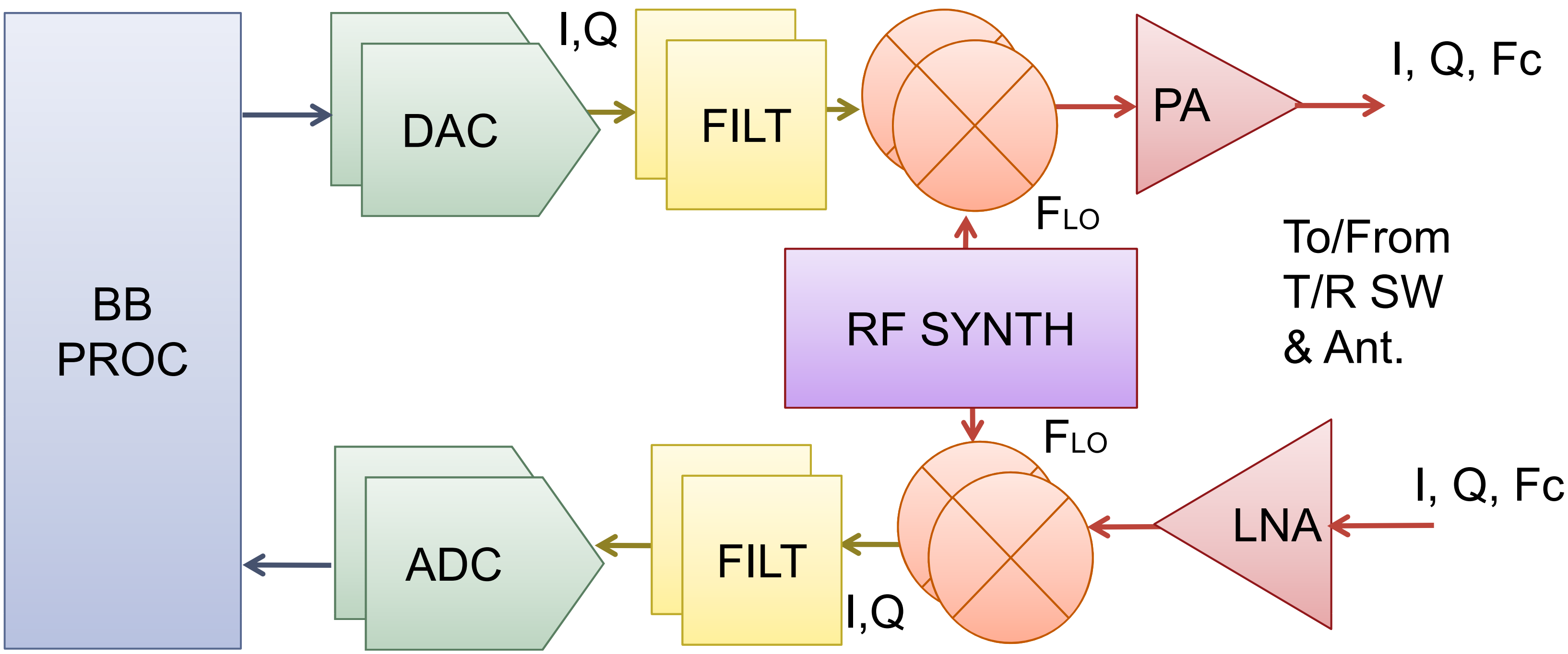
Agenda

- Background and Motivation
- Mixer theory
  - Ugly Math
  - Pretty result
- Model code
- Test code
- Results

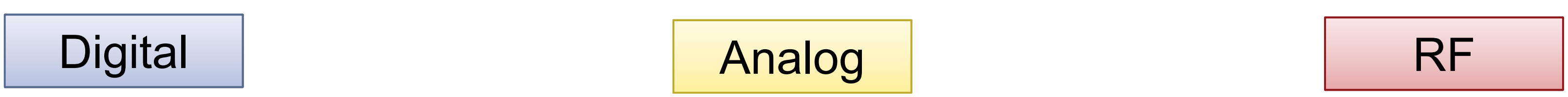
Drivers for Consumer Device IC's

- Wires – Weight – Wallet
  - Consumers like Simplicity,
    - Convenience, Affordability
  - ~~Cables~~
  - ~~Power Outlet~~
  - Batteries
- Wireless connectivity is improvement.
  - If LOW power and LOW cost.
  - Lower Power -> more digital & simpler analog[Horowitz] -> smaller process nodes
  - Lower Cost -> more integration
- More Digital More Integration -> More risk of respin for functional failure
  - RF functional verification is key enabler to success.

Block Diagram of Generic Wireless tranciever



- Using real number modeling – models for all blocks are fairly straightforward except the RX mixer.
  - RF I & Q values must be rotated to BB I & Q values based on phase difference between Carrier and Local Oscillator.



Representing RF signals

$$S_{RF} = re\{[I(t) + jQ(t)]e^{j\omega t}\}$$

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

- I, Q, f [= ω/2π] are sufficient to represent most RF signals.
  - FM: I = signal Ampl Q = 0, variation in f carries signal information
  - AM: I = signal, Q = 0, f is carrier frequency
  - QAM: I & Q carry Baseband signal, f is channel frequency.
- System Level simulations (Matlab)
  - assume fixed frequency,
  - use complex number for I & Q.
  - Assumption doesn't hold for functional verification.

Mixer Theory

- “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:

$$(A\cos\omega_1 t)(B\cos\omega_2 t)$$

$$=\frac{AB}{2}[\cos(\omega_1-\omega_2)t+\cos(\omega_1+\omega_2)t]$$

”

- Tom Lee [15]

The Ugly Math – Quadrature mixer

1: Get an instantaneous value for frequency(phase)

$$\omega t = \int_0^t \omega \, dt = \omega \int_0^t dt$$

$$\phi = \int_0^t \omega \, dt = 2\pi \int_0^t F dt$$

2: This makes our Mixer Equation:

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

3: define a few other terms

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

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

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

$$M'_{11} = \cos(\Delta\phi) + \cos(\Sigma\phi)$$

$$M'_{12} = \sin(\Delta\phi) + \sin(\Sigma\phi)$$

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

$$M'_{22} = \cos(\Delta\phi) - \cos(\Sigma\phi)$$

4: Dropping the summing terms due to filtering
Our Quadrature receiver gives us two output signals:

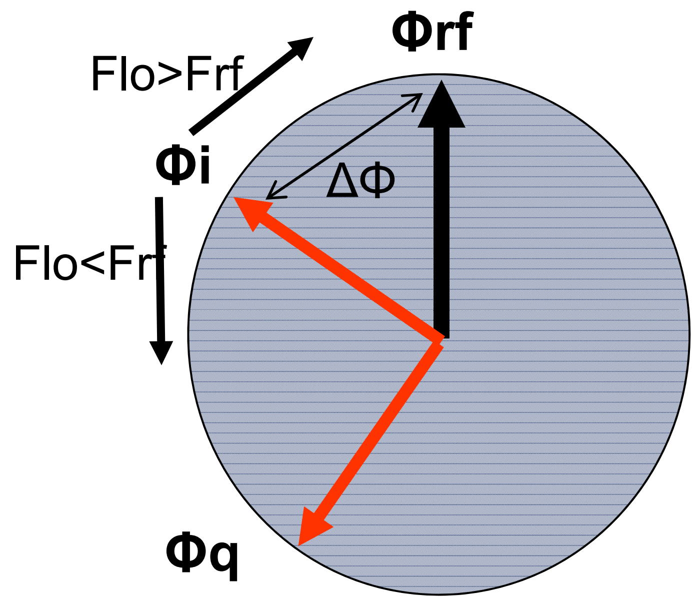
$$I_{REC} = \frac{I}{2}[\cos(\Delta\phi)] - \frac{Q}{2}[\sin(\Delta\phi)]$$

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

The Pretty Result

- The mixer model only needs to know the Difference in the phase between the RF Frequecy and LO frequency to calculate the Baseband output of the mixer.
- Since integration is linear, the difference in phase can be calculated by integrating the difference in frequency.

$$R_I = G_I\left[I(t)\cos(\Delta\phi) - Q(t)\sin(\Delta\phi)\right]$$
$$R_Q = G_Q\left[I(t)\cos(\Delta\phi - \pi/2) - Q(t)\sin(\Delta\phi - \pi/2)\right]$$



Model Code – module and variable declarations

```
module  rx_mixer_generic_sv ( // goes here
  input logic enable,                        // only logic controls are made as pins
  input var real RXin_Iinph, RFin_Iquad,    // 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
//----------------------------------------------------------------------------
reg Fi_ok, Fq_ok; //status LO input freq and sig BW
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
```

Model Code – module initialization

```
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_Iinph;  // at t0 M11 M22 = 1 M12 m21 = 0
    BB_Q_Iout = signQ*RFin_Iquad;
  end
```

Model Code – sanity check inputs

```
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;
```

Model Code – Integration to get deltaPhi

```
always @(LOin_Freq_inph, RXin_freq, RXin_Iinph, RFin_Iquad) 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
         // - 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;
```

Model Code – core equations

```
     // 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_Iinph + M12 * RFin_Iquad;
    BB_Q_Iout   = M21 * RXin_Iinph + M22 * RFin_Iquad;
  end
endmodule
```

Testing the model – 10 cases

TestID	Description	Frf	Flo	Fbb	LO IleadsQ	Gain
1	"Frf = Flo"	2.4G	2.4G	2M	T	3dB
2	"Frf < Flo"	2.4G	2.4005G	2M	T	3dB
3	"Frf << Flo"	2.4G	2.404G	2M	T	3dB
4	"Frf > Flo"	2.4005G	2.4G	2M	T	3dB
5	"Frf >> Flo"	2.402G	2.4G	2M	T	3dB
6	"Frf = -Flo"	2.4G	2.4G	2M	F	3dB
7	"Frf < -Flo"	2.4G	2.4005G	2M	F	3dB
8	"Frf << -Flo"	2.4G	2.404G	2M	F	3dB
9	"Frf > -Flo"	2.4005G	2.4G	2M	F	3dB
10	"Frf >> -Flo"	2.402G	2.4G	2M	F	3dB

Testing the model - expected results

```
Fexpect = $qc_abs(Frf + Fbb - Flo);
IleadsQexpect = LO_IleadsQ ~^ (Frf + Fbb > Flo);
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
```

Testing the model – Did we pass (Verilog)

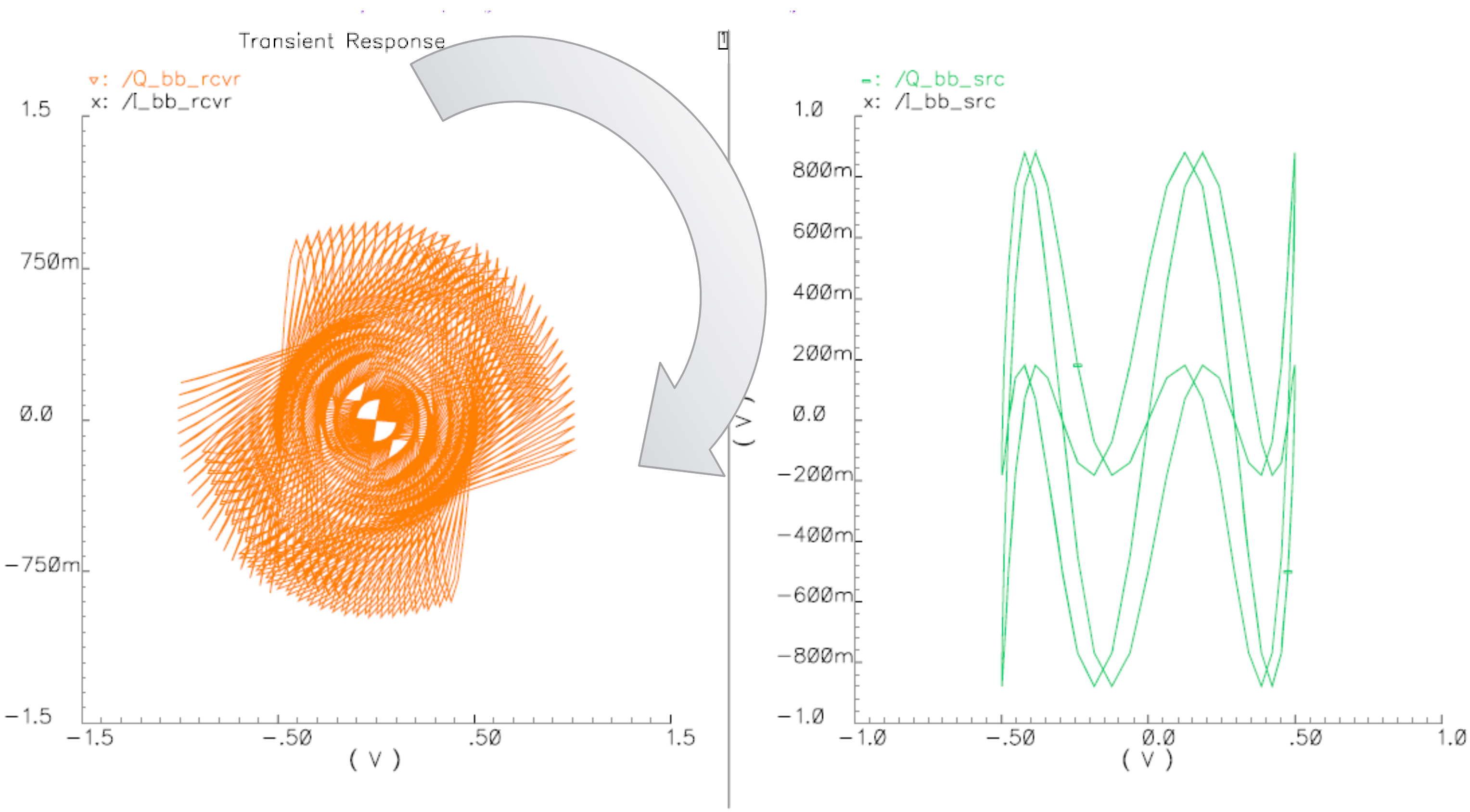
```
testpassed = (IleadsQexpect === RX_BBMon.I_leads_Q)
                 && ($qc_abs(BB_Ifreq - Fexpect) <2e3)
                 && ($qc_abs(BB_Qfreq - Fexpect) <2e3)
                 && GainOK;
$strobe("%s @ %t: RXTEST%02d   %s  Gain:%g dB, Frf: %g Hz,
                      Flo: %g   LO:%s Hz, Fbb: %g 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;
```

Automated Test Results

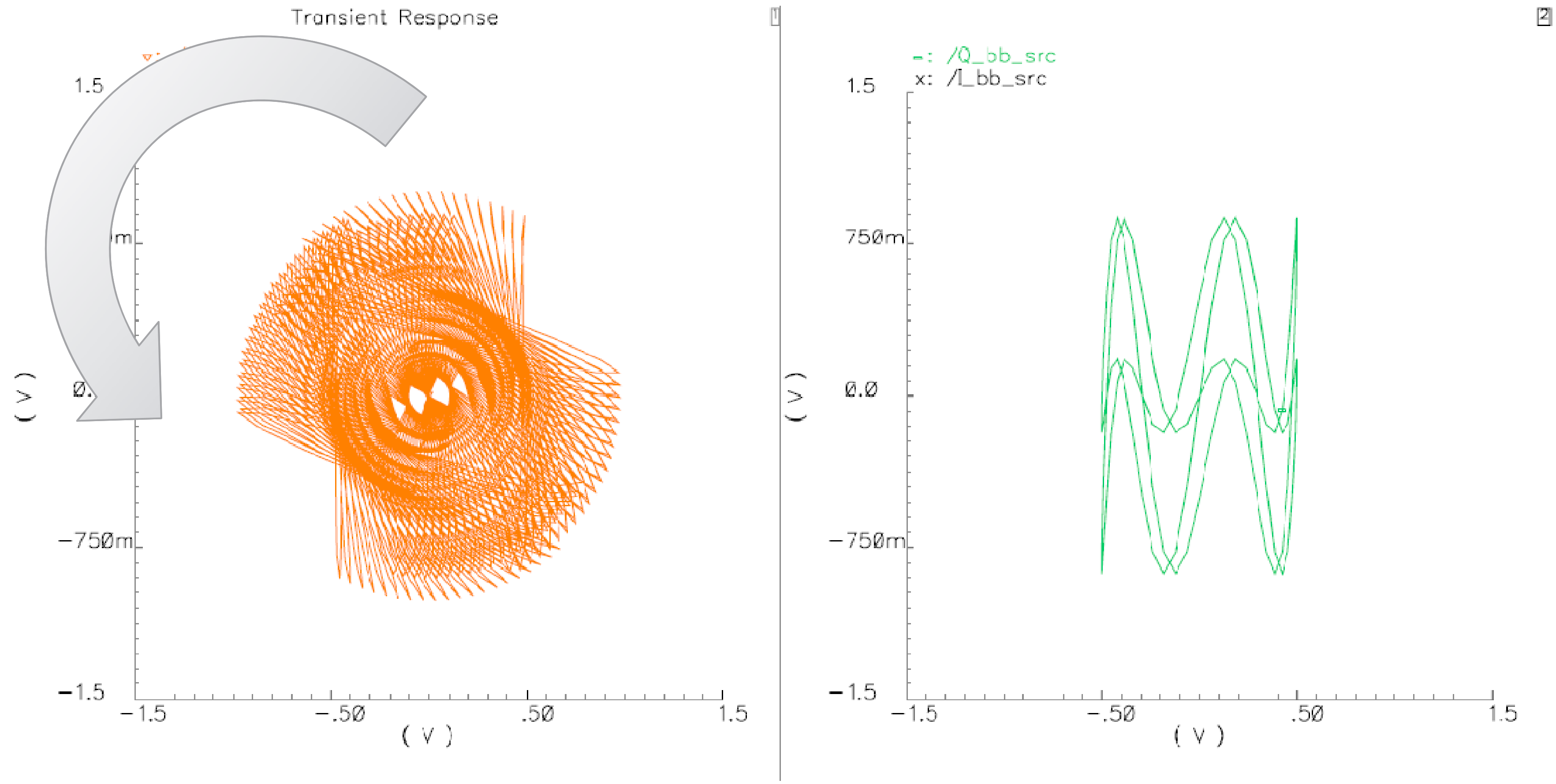
SPECINFO @ 15388.000 ns: RXTEST01 Passed : Gain:3.00487 dB, Frf: 2.4e+09 Hz, Flo: 2.4e+09 LO:I leads Q Hz, Fbb: 2e+06 Hz, BB:I leads Q
SPECINFO @ 33228.000 ns: RXTEST02 Passed : Gain:3.00867 dB, Frf: 2.4e+09 Hz, Flo: 2.4005e+09 LO:I leads Q Hz, Fbb: 1.5e+06 Hz, BB:I leads Q
SPECINFO @ 47898.000 ns: RXTEST03 Passed : Gain:3.00355 dB, Frf: 2.4e+09 Hz, Flo: 2.404e+09 LO:I leads Q Hz, Fbb: 2e+06 Hz, BB:Q leads I
SPECINFO @ 60648.000 ns: RXTEST04 Passed : Gain:2.99961 dB, Frf: 2.4005e+09 Hz, Flo: 2.4e+09 LO:I leads Q Hz, Fbb: 2.5e+06 Hz, BB:I leads Q
SPECINFO @ 70528.000 ns: RXTEST05 Passed : Gain:2.98611 dB, Frf: 2.402e+09 Hz, Flo: 2.4e+09 LO:I leads Q Hz, Fbb: 4e+06 Hz, BB:I leads Q
SPECINFO @ 85398.000 ns: RXTEST06 Passed : Gain:3.00598 dB, Frf: 2.4e+09 Hz, Flo: 2.4e+09 LO:Q leads I Hz, Fbb: 2e+06 Hz, BB:Q leads I
SPECINFO @ 103228.00 ns: RXTEST07 Passed : Gain:3.00735 dB, Frf: 2.4e+09 Hz, Flo: 2.4005e+09 LO:Q leads I Hz, Fbb: 1.5e+06 Hz, BB:Q leads I
SPECINFO @ 117908.00 ns: RXTEST08 Passed : Gain:3.00546 dB, Frf: 2.4e+09 Hz, Flo: 2.404e+09 LO:Q leads I Hz, Fbb: 2e+06 Hz, BB:I leads Q
SPECINFO @ 130668.00 ns: RXTEST09 Passed : Gain:3.0074 dB, Frf: 2.4005e+09 Hz, Flo: 2.4e+09 LO:Q leads I Hz, Fbb: 2.5e+06 Hz, BB:Q leads I
SPECINFO @ 140548.00 ns: RXTEST10 Passed : Gain:2.98868 dB, Frf: 2.402e+09 Hz, Flo: 2.4e+09 LO:Q leads I Hz, Fbb: 4e+06 Hz, BB:Q leads I

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

Example Waveforms Flo > Frf



Lissajou plot Flo < Frf



Summary

- Passband model is conceptually simpler:
  - Single real value sampled at 8-20x RF carrier
  - Mixer is simple multiplication with approx sinusoid Quad LO signal.
  - followed by 3rd or 4th order discrete time IIR filter.
- This model is approx 100x faster in simulation than Passband model
- Inserting RF signal path in chip-chip simulations is only 2-3x longer run times vs all digital signal path (with DAC->ADC channel model)
  - This Run time is acceptable given ability to check FULL functionality of Radio control and data interfaces.