Real Number Modeling of RF Circuits

Jakub Dudek
Analog Devices Inc.
Wilmington Ma USA

Joshua Nekl
Analog Devices Inc.
Cork Ireland

Keith O'Donoghue
Analog Devices Inc.
Cork Ireland

Abstract

Radio Frequency (RF) circuits are generally not considered as candidates for SystemVerilog Real Number Modeling (RNM). The complexity and precision requirements of RF circuits such as Phase Locked Loops (PLLs) and RF Mixers have traditionally kept real number modeling efforts at bay. However, the high speeds associated with RF circuits make them cumbersome in traditional mixed signal simulation approaches such as AMS models and Fast Spice simulators. We demonstrate that RF circuits are excellent candidates for RNM modeling in the context of a complex mixed-signal RF SoC. Furthermore, RNM models present an opportunity to build a fast and efficient simulation platform for test and firmware development, while also being perfectly suited for leveraging of UVM based random constrained verification techniques. With this in mind, we present a full suite of RNMs used in a 2.4GHz Transceiver SoC, including coding examples of such models.

I. INTRODUCTION

Radio Frequency (RF) circuits are generally not considered as candidates for SystemVerilog Real Number Modeling (RNM). The complexity and precision requirements of RF circuits such as Phase Locked Loops (PLLs) and RF Mixers have traditionally kept real number modeling efforts at bay. Furthermore, the high frequency signaling associated with RF circuits make them cumbersome in traditional mixed signal simulation approaches such as AMS models and Fast Spice simulators. We demonstrate that RF circuits are excellent candidates for RNM modeling in the context of a complex mixed-signal RF SoC. RNM models present an opportunity to build a fast and efficient simulation platform for test and firmware development, while also being perfectly suited for leveraging of UVM based random constrained verification techniques. With this in mind, we present a full suite of RNMs used in a 2.4GHz Transceiver SoC, including coding examples of such models.

Figure 1 shows a high level block diagram of the transceiver. Focus will be maintained on the Mixers and the PLLs as they are by far the more complex and interesting models.

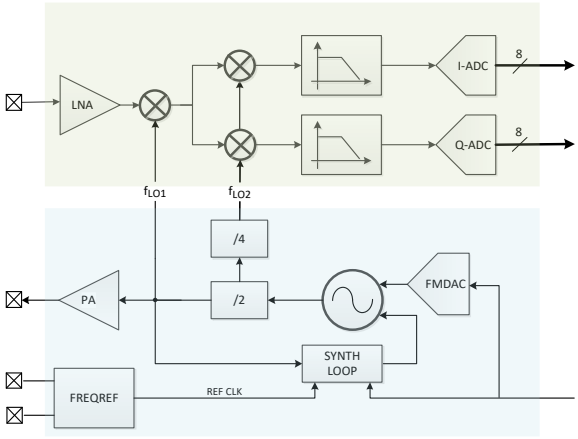

Figure 1: RF Analog Front End

Specifically, we will discuss the following modeled blocks:

- PLL
  - VCO
  - Phase Detect/Charge Pump block
  - Loop Filter
  - Divider
- Mixer
- Base Band Filter

II. MOTIVATION

It is important to clearly state the objectives of any modeling effort. While it is often stated what a particular model or modeling strategy will be used for, it is equally important to define what the models should not be used for in the context of their shortcomings and limitations.

The purpose of real number modeling of RF circuits is to create a fast and "accurate-enough" full chip simulation environment which can be heavily leveraged for the following tasks in the development cycle.

A. Fast development of test cases

Developing and debugging test cases is in itself a time consuming task. A test case in development may be run dozens of times before being finalized with the simulation time on every iteration directly impacting schedule and cost. A traditional mixed signal co-simulation platform is therefore a poor choice for test case development as it will increase engineering idle time as well as simulation resource usage.

A digital simulation environment without analog models does not provide a complete device under test (DUT) in terms of test case development. Analog circuits in a large SoC tend to have many digital controls, and the sequencing of these controls is important. Furthermore, in the case of an RF receiver, many control loops exist between the analog and digital domains (automatic gain control, offset and frequency corrections) and need to be appropriately exercised in a variety of test cases. An RTL only simulation setup is thus a poor choice for test case development.

A Verilog only simulation environment with real number models has the potential to offer the best of both worlds: fast turnaround in simulation and access to all DUT features.

B. Firmware development

Firmware development traditionally happens on FPGA platforms where the turnaround time is as fast as it can get outside of actual silicon. This is indeed the fastest way to develop firmware, but stable FPGA images are not typically available until later in the design/verification cycle. A fast verilog-only simulation platform can be

available much earlier in the design cycle and can give firmware engineers an early chance at developing initial firmware routines. These firmware routines can be included in the verification work and leveraged in various verification test cases. This synergy opportunity between verification and firmware can alleviate the need for verification engineers to create their own software routines to exercise the DUT.

C. Random constrained simulations

A further advantage of a fast turnaround platform is the ability to run a substantial number of simulations. This in turns allows for the introduction of a reasonable amount of system level randomization which gives the ability to test the radio and full DUT over many different packet parameters and radio setups. Attributes such as channel number, packet length, packet type, frequency deviation, gain error, offset error, input power etc. can be thrown into a mix of a random constrained test cases which can generate excellent coverage in system level simulations.

Figure *2* shows a coverage report snapshot of hundreds of receive and transmit simulations over some of the parameters mentioned above.

Average Grade	Covered Grade	Goal	Weight	Uncovered Bins	Excluded Bins	Total Bins	Item	Name
100.00%	100.00% (6/6)	100%	1	0	0	6	CoverPoint	rx_cg.rx_pkt_q_size
100.00%	100.00% (3/3)	100%	1	0	0	3	CoverPoint	rx_cg.rx_protocol
100.00%	100.00% (9/9)	100%	1	0	0	9	CoverPoint	rx_cg.rx_preamble_size
100.00%	100.00% (3/3)	100%	1	0	0	3	CoverPoint	rx_cg.rx_postamble_size
100.00%	100.00% (3/3)	100%	1	0	0	3	CoverPoint	rx_cg.rx_midamble_size
100.00%	100.00% (7/7)	100%	1	0	0	7	CoverPoint	rx_cg.rx_channel
100.00%	100.00% (2/2)	100%	1	0	0	2	CoverPoint	rx_cg.rx_header_size
100.00%	100.00% (2/2)	100%	1	0	0	2	CoverPoint	rx_cg.rx_pid_crc_size
100.00%	100.00% (2/2)	100%	1	0	0	2	CoverPoint	rx_cg.rx_whiten
100.00%	100.00% (97/97)	100%	1	0	0	97	CoverPoint	rx_cg.rx_dbm
100.00%	100.00% (10/10)	100%	1	0	0	10	CoverPoint	rx_cg.rx_f_err
100.00%	100.00% (23/23)	100%	1	0	0	23	Cross	rx_cg__rx_protocol_X_rx_preamble_size
100.00%	100.00% (18/18)	100%	1	0	0	18	Cross	rx_cg__rx_protocol_X_rx_channel
100.00%	100.00% (6/6)	100%	1	0	0	6	Cross	rx_cg__rx_protocol_X_rx_whiten
100.00%	100.00% (70/70)	100%	1	0	0	70	Cross	rx_cg__rx_f_err_X_rx_channel
100.00%	100.00% (6/6)	100%	1	0	0	6	Cross	rx_cg__rx_protocol_X_rx_midamble_size
100.00%	100.00% (6/6)	100%	1	0	0	6	Cross	rx_cg__rx_protocol_X_rx_postamble_size
100.00%	100.00% (3/3)	100%	1	0	0	3	Cross	rx_cg__rx_protocol_X_rx_header_size
100.00%	100.00% (4/4)	100%	1	0	0	4	Cross	rx_cg__rx_protocol_X_rx_pld_crc_size
100.00%	100.00% (237/237)	100%	1	0	0	237	Cross	rx_cg__rx_protocol_X_rx_dbm
100.00%	100.00% (24/24)	100%	1	0	0	24	Cross	rx_cg__rx_protocol_X_rx_f_err
100.00%	100.00% (13/13)	100%	1	0	0	13	Cross	rx_cg__rx_protocol_X_rx_pkt_q_size

Figure 2: Functional coverage of RF parameters and packet parameters

Note the coverage of parameters such as protocol type, packet size, bursts, channel number, input power and various crosses of these parameters. Cross coverage such as input power crossed with frequency deviation can be created which yields valuable insights into the limitations of the entire RF signal chain. While such extensive coverage would be impractical in a typical analog co-simulation environment, a sub-set of these tests can be selected with identical random constrained characteristics for further analysis in the co-simulation environment.

D. What the RNM strategy is not

It is important to note what RNM RF models are not intended to accomplish. RF performance is best verified with a dedicated RF simulator and as such, RNMs are not intended as a method of identifying issues related to RF performance such as noise, linearity or interference. The performance of the RF RNMs only need to be good enough to support correct system level use case behavior. For example, an incoming packet at a given input power should create the right internal signal levels, at the correct frequency, across the RX signal chain to generate the appropriate automatic gain correction (AGC) response from the AGC sub-system. In general, RNM modelling of gain can be accurate to within 1-2dBm across the RX signal chain. While this performance is not acceptable from an RF circuit standpoint, it is sufficient for exercising a wide variety of system level scenarios.

III. TX PATH

The TX path consists of a 2.4GHz Fractional-N PLL, Frequency Modulation DAC (FMDAC) and a Power Amplifier (PA). Figure 3 shows a block diagram of the transmitter. Note that the PLL is also used by the RX path.

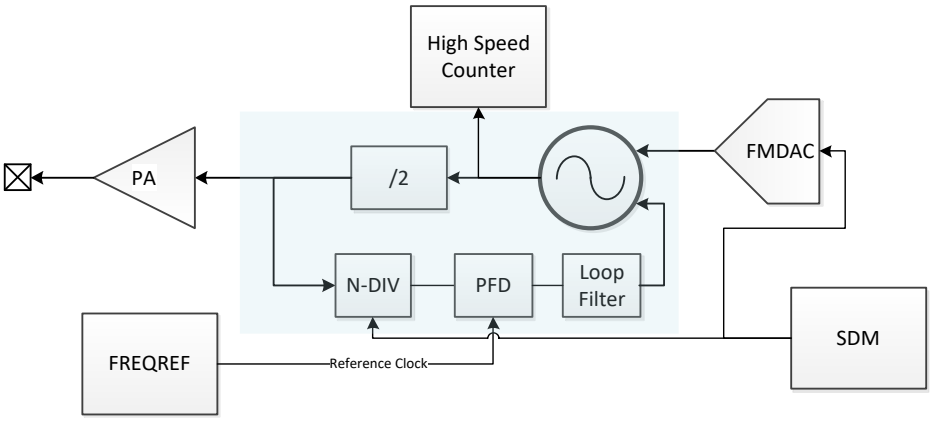

Figure 3: TX Path

A. PLL

The loop consists of a 5GHz voltage controlled oscillator (VCO), phase detector, charge pump, loop filter, and divider. The VCO frequency can be calibrated according to the desired carrier output by dialing in coarse and fine calibration codes. The calibration codes are generated in an open loop configuration by measuring the output frequency. The High speed frequency counter is designed in gates so the model is obtained for free by simply netlisting the circuit and compiling the standard cell library.

1) VCO

The voltage controlled oscillator sets the frequency of operation for transmit and receive. It is a 2-port oscillator with low frequency port as part of closed-loop configuration and a high frequency port that is used for modulation during TX.

The nominal oscillating frequency is determined using a lookup table based on input trim signals. This frequency is then increased or decreased based on low and high port input voltages each scaled by their own kV (voltage to frequency gain factor). For simplicity, the voltage to frequency gain is assumed to be linear and modeled based on design spec or analog simulations. The kV gains can also be adjusted by trim codes with corresponding gain value obtained from a lookup table.

This high frequency circuit must be modeled carefully to avoid excessive simulation slowdown. Rather than compute vco period upon each vco oscillation edge, internal frequency and period terms are generated and computed only when their respective input signals that affect them change (essentially caching their values).

```
assign f_vco = fc + kv_lp * (vtune_lp-0.6) + kv_hp * (vtune_hp_p-vtune_hp_m);
assign hperiod = (1s/f_vco)/2.0;

always begin
wait(pwr_ok);
       #hperiod vco = !vco & pwr_ok;
end
```

In the code above, the frequency of the VCO is continuously updated based on the nominal frequency plus the low port tuning times its kV and the high port tuning times its kV. Note that the low port is single ended swinging between 0 and 1.2V.

2) Divider

The output of the VCO is divided down by a factor provided from a digital sigma delta modulator. While the actual circuit consists of dynamic logic due to high speed operation and is spilt into two separate counters, this was simplified in simulation as a simple integer divider.

```
always @(posedge clk_vco_p or negedge reset_n)
begin
        if(!reset_n) begin
                cntr <= 0;
                div_clk <= 0;
        end else if(cntr == 0) begin
                cntr <= n - 1;
                div_clk <= ~div_clk;
        end else begin
                cntr <= cntr - 1;
                if(cntr == n/2) div_clk <= 0;
        end
end
```

3) Phase detect

Like the divider, the phase frequency detector was a simple digital circuit that can be easily modeled. It was a textbook phase comparator consisting of two edge clocked flops, delay and reset circuit.

```
always @(posedge ref_clk or negedge rst)
begin
        if(!rst) begin
                pup = 1'b0;
        end else if(pdown) begin
                pdown <= 1'b0;
        end else begin
                pup <= 1'b1;
        end
end

always @(posedge div_clk or negedge rst)
begin
        if(!rst) begin
                pdown = 1'b0;
        end else if(pup) begin
                pup <= 1'b0;
        end else begin
                pdown <= 1'b1;
end
```

4) Charge pump

The charge pump circuit was likewise straightforward, taking the duration of up & down digital signals and scaling them by a current conversion factor. This circuit can take trim values to adjust the current. The actual current scaling happens in the loop filter part of the code

```
always @(posedge pup)
        pup_time = $realtime;

always @(posedge pdown)
        pdn_time = $realtime;

always @(negedge pup)
        void'(filter($realtime-pup_time));

always @(negedge pdown)
        void'(filter(pdn_time-$realtime));
```

5) Loop filter:

The loop filter is one of the more complex circuits to model. Initially a simplified model was used, leading to instability and the PLL failing to lock. This is a high order circuit with several RC components as show in Figure 4

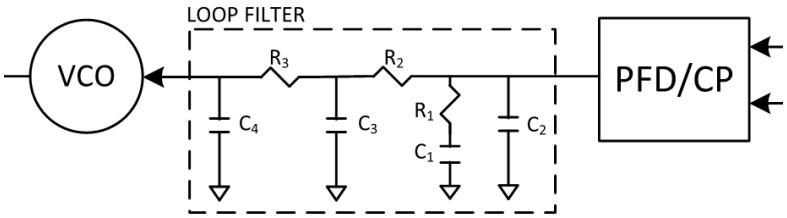

Figure 4: PLL loop filter

Rather than try to simplify the model, the RC circuit is modeled using the discrete components of the actual filter. This is done using a simple one pass circuit solver as the R*C time constants are greater than the input reference frequency. When an input event occurs (charge pump value changes), the amount of charge on the input capacitor is computed (ΔQ=I*t) along with change in voltage (ΔV=ΔQ/C). This value is then clamped at the supply rails. Next, all the currents flowing through resistances are computed ([V1-V2]/R). Lastly, the voltage on each capacitor node is updated using previous equations (ΔV=I*t/C).

Unlike spice which iterates over multiple passes until convergence is received, this simple circuit evaluator completes in a single pass. Again, since the reference period is less than R*C time constants, this approximation provides reasonable accuracy for our simulation purposes. Additional iterations would slow down simulation.

```
function void filter(realtime t);
        real Ir1, Ir2, Ir3;

        t /= 1s;

        // sanity check
        t = min(t, +Txtal);
        t = max(t, -Txtal);

        // integrate charge pump current onto C2
        Vc2 += Icp*t/C2;

        // compute currents, I=V/R
        Ir1 = (Vc2-Vc1)/R1;
        Ir2 = (Vc2-Vc3)/R2;
        Ir3 = (Vc3-Vc4)/R3;

        // compute voltages: V=I*t/C
        Vc1 += Ir1*Txtal/C1;
        Vc2 -= (Ir1+Ir2)*Txtal/C2;
        Vc3 += (Ir2-Ir3)*Txtal/C3;
        Vc4 += Ir3*Txtal/C4;

        Vc2 = min(Vc2, Vdd);
        Vc2 = max(Vc2, 0);

endfunction
```

Since the loop filter is modeled using individual R & C components similar to the actual loop filter, any trim codes that change R & C values to adjust bandwidth are easily modeled by adjusting these parameters based on a lookup table.

6) PLL lock

Figure 5 shows the PLL locking to 1.9232 GHz in simulation. The frequency error is also shown along with a zoomed view of the settled frequency value and frequency error. While the dialed in settings for the PLL is to lock at 1.9232 GHz, a small frequency error can be observed. The PLL circuit frequency error is specified as +-40ppm, which should be 76.928 kHz. In this particular example, a maximum error of 25 kHz can be observed, well within the specified error. The real number model thus meets the analog specification for the PLL frequency output.

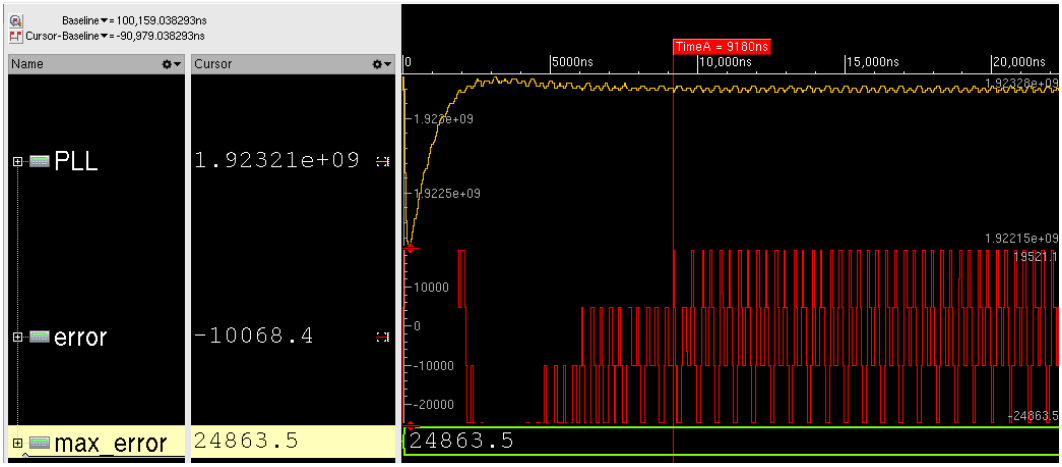

Figure 5: PLL lock at 1.9232 GHz

Similarly, the circuit lock time is specified to lock to a stable frequency within 45us. This specification is easily met by this model as shown in Figure 5

B. Frequency Modulation DAC

The FM DAC is modeled as a simple DAC. It takes a binary code and is scaled by a reference voltage to form an output voltage.

The output of the DAC drives the high port of the VCO during RF transmission. The PLL loop adjusts dynamically as the VCO is modulated with data to be transmitted. An example is shown in Figure 6

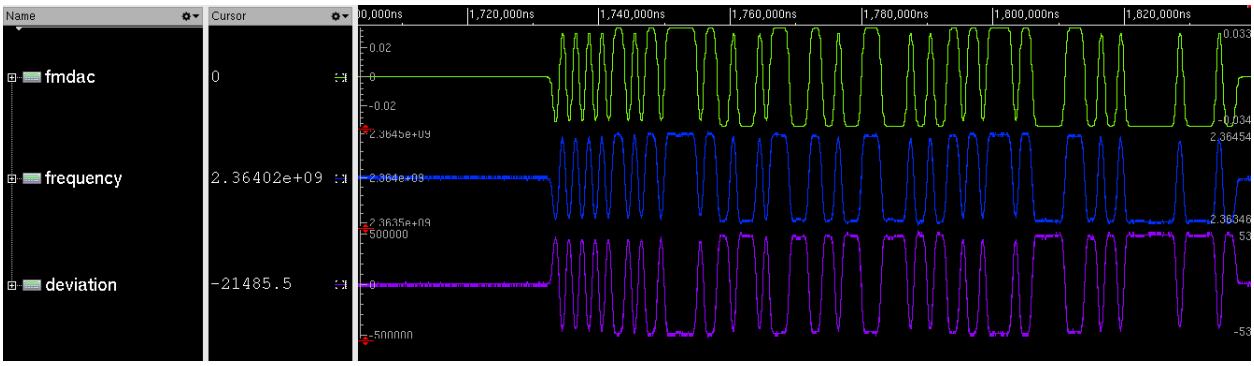

Figure 6: FMDAC modulation of VCO, transmit operation

IV. RX Path

Figure 7 shows a simple block diagram of the RX analog signal chain. The topology is a traditional sliding IF quadrature architecture with a low noise amplifier, RF mixer, IF mixer, I/Q baseband filters and I/Q Analog to digital converters.

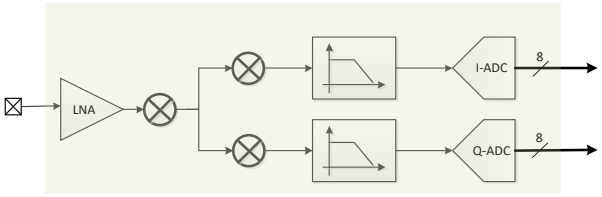

Figure 7: RX Path: Low noise amplifier, Sliding-IF mmixers and ADCs

A. RF Input stimulus

A key concept in the RF test environment is to use an RF square wave stimulus rather than a sine wave stimulus as in the real world. A high speed modulated square wave generates simulation events at its carrier frequency (2.4GHz on average). Within the context of an event based simulator, a sine wave carrier would generate events at a multiple of 2.4GHz. For example a 32x sampled sine wave period would trigger events at 77GHz (effectively "clocking" the design at 77GHz). This leads to prohibitive simulation times and debug waveform dumps of several tens of gigabytes.

The carrier square wave is modulated using a GFSK scheme where symbols are first shaped by a Gaussian filter to generate the frequency deviation used by the carrier. The following code shows the calculation of the Gaussian coefficients as well as the application of the filter to the input signal.

```
function calc_coeff;
    real s;
    s=sqrt(log(2))/(2*PI*(0.5)/T);
    for(int i=0; i<=3*T; i++) begin
        int t;
        t=i-3*T/2;
        c[i]=1/(sqrt(2*PI)*s)*exp(-((t/s)**2)/2);
    end
endfunction

function real gfsk_filter;
    real y;
    y=0;
    for(int i=0; i<=3*T; i++)
        y += x[i]*c[i];
    return y;
endfunction
```

The output of the filter function generates the deviation to be applied to the carrier wave. As mentioned earlier, the carrier wave is generated as a simple clock with variable period.

```
rf_tstep = 1/(fc+f_err+fdev+fdrift)/2;
task gen_rf;
    forever begin
        wait(rf_tstep > 0);
        #(rf_tstep *1s);
        vif.rf_r = vif.rf ? amp : -amp;
    end
endtask
```

The *rf_tstep* (square wave half period) is comprised of the carrier wave period and the deviation calculated by the Gaussian filter. Note that it also incorporates a frequency error and a frequency drift that is used across a variety of test cases.

Figure 8 shows the input symbols and their filtered version.

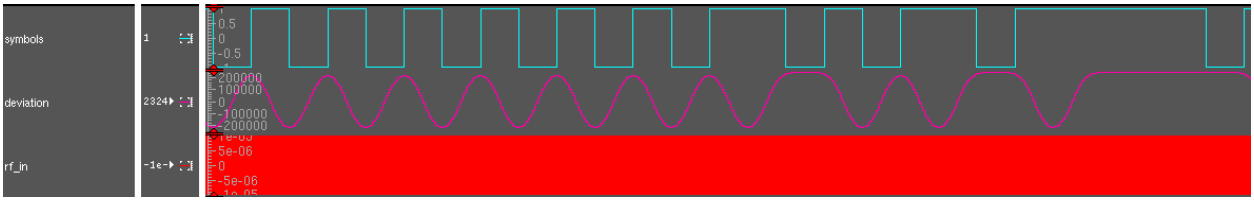

Figure 8: RF input. Note that is not possible to depict a 2.4GHz signal with deviations as small as 250 kHz in a way that can be observed by the naked eye.

Note that to represent a 2.4GHz waveform with variation as small as 250 kHz in frequency, the simulation timescale must be set to 10fs resolution.

B. Mixer

A mixer takes an RF input signal at a frequency f_{RF}, mixes it with a LO signal at a frequency f_{LO}, and produces an IF output signal that consists of the sum and difference frequencies, $f_{RF} \pm f_{LO}$. In the RX chain, the sum is filtered out and the difference term is processed making the mixer a down conversion operation. The sliding IF topology used in this design uses two mixers in series. The first mixer down converts to an intermediate frequency (472-500MHz). This is followed by a final down conversion to DC where quadrature clocks are used to generate the I and Q mixer output signals. Figure 9 shows the sliding IF down converter.

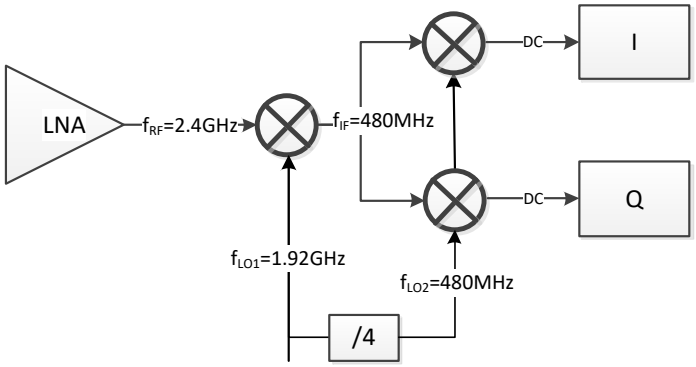

Figure 9: Sliding IF receiver

The mixers are modeled as simpler binary multipliers. The first mixer is simply the LO binary value where 1 is 1 and 0 is -1.

```
assign i_f = lo ? rf : -rf;
```

The quadrature down converters follow a similar approach:

```
assign I = lo_cos_p ? +i_f : lo_cos_n ? -i_f : 0;
assign Q = lo_sin_p ? +i_f : lo_sin_n ? -i_f : 0;
```

Signals lo_sin_p(n) and lo_cos_p(n) are shown in Figure 10:

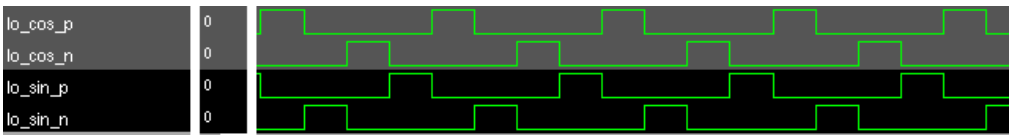

Figure 10: quadrature local oscillator, ¼ frequency of VCO

The coded mixers above are ideal and have infinite bandwidth. In reality the mixer has a 1st order low pass responses with cut off frequencies at 1 GHz and 10MHz respectively. The cutoff is implemented with a first order IIR with variable time step.

1) First order IIR filter

The first mixing stage is ideal with infinite bandwidth. This has fundamental frequency components at ~2.5GHz +/- 2GHz ~= 500MHz & 4.5GHz. To model bandwidth limitations and remove high frequency components requires a filter. To satisfy Nyquist, this would require a filter with a sample rate greater than 2x 4.5GHz ~= 9GHz. Given that the filter is a discrete system with square wave signaling to capture even a minimum of 5 odd harmonics would push the required sample rate close to 100GHz and slow simulator performance to a crawl.

In order to take advantage of the event driven simulator, a first order digital IIR filter was implemented that utilizes a dynamic timestep. On each simulator event (input change), the filter is evaluated. This results in a filter which has a non-constant sample rate. Since the filter coefficients are dependent on the sample rate, the timestep must be measured and each coefficient updated accordingly.

The basic structure of the first order IIR filter is shown in Figure 11. One can derive an analytical expression for the filter coefficient β based on 3dB cutoff frequency relative to sample rate as show in Equations (1) and (2).

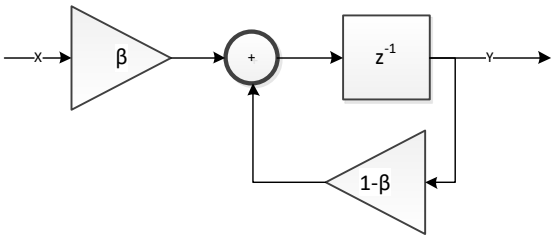

Figure 11 First order IIR filter

$$\omega_c = 2\pi \frac{f_c}{f_s} = 2\pi f_c T_s \tag{1}$$

$$\beta = -1 + cos(\omega_c) + \sqrt{(cos(\omega_c) - 3)(cos(\omega_c) - 1)} \tag{2}$$

$$\beta \cong \omega_c \; for \; f_c \ll f_s \tag{3}$$

The effective sample rate T_s is the time between update events. When an update event occurs, the filter should use the input value just before the update event that existed during the previous time period, not the new value.

If the cutoff frequency is a decade or two less than the effective sample rate, the approximation in Equation (3) may be used. The coefficient computation is thus a single multiply (by cached constant $2\pi f_c$). This is valid for the second mixer stage. In the first mixer stage, the bandwidth is ~1GHz which is close enough to the sampling period that the more accurate Equation (2) must be used. The sqrt() and cos() functions can be imported from libm using the DPI-C interface, or $sqrt and $cos functions if the simulator supports them.

2) Mixer Gain

The mixer gain is specified to be programmable to levels of 3dB, 7dB and 10dB from input to output. The gain is specified as a voltage gain. It is important to realize that mixer code has intrinsic gain, which needs to be factored in the specified gain across the mixer.

We can access the situation qualitatively. The first mixer mixes a square wave with a square wave and filters out the f_1+f_2 component. We expect a 2x reduction in amplitude from the removal of the high frequency product. The 1 Ghz 1st order roll off should also remove some frequency harmonics of our input square wave, leaving us with the fundamental who's amplitude is 4/pi greater than the amplitude of the square wave. We also expect about 0.8dB filter droop at f_1-f_2 reducing the signal further. The RMS value of the output signal should thus be

$$20 * log\left(\frac{1}{2} * \frac{4}{\pi} * \frac{1}{\sqrt{2}}\right) - 0.9dB = -7.8dB$$

However we also expect some high frequency component given a 20dB/decade roll off. The high frequency component f_1+f_2 is located near 4.32 GHz and the first odd harmonic of the input square wave is at 7.2 GHz. We expect some of this power to remain within the signal.

From simulation, Figure 12 shows a loss of 6.56dB across the first mixer which is within expectations.

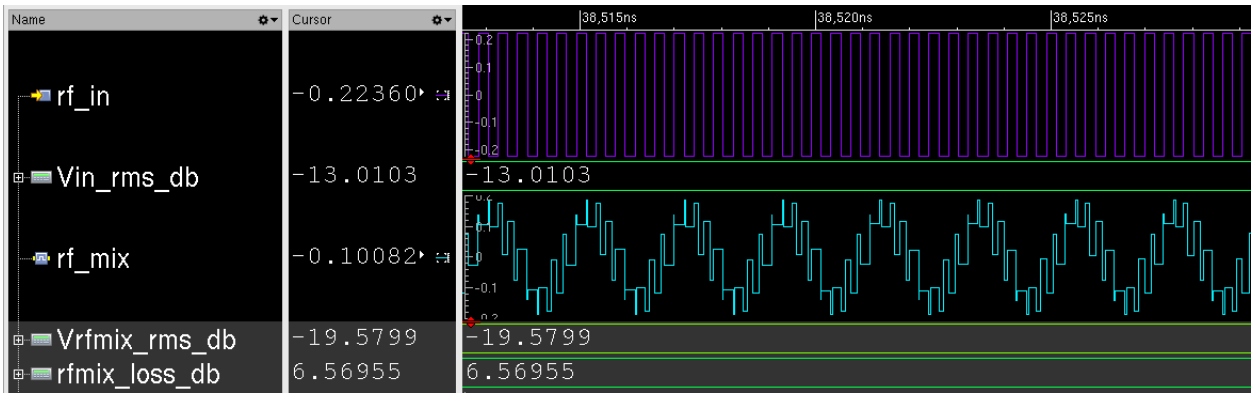

Figure 12: First down conversion loss

A similar approach can be taken for the second set of mixers, where the output magnitude is computed as $A = \sqrt{I^2 + Q^2}$. There should be no filter droop in this case as the output is at DC, and similar arguments can be made for high frequency components remaining within the signals. We should expect a loss smaller than 6dB. As shown in Figure 13, the loss is around 4.5dB, again within expectation.

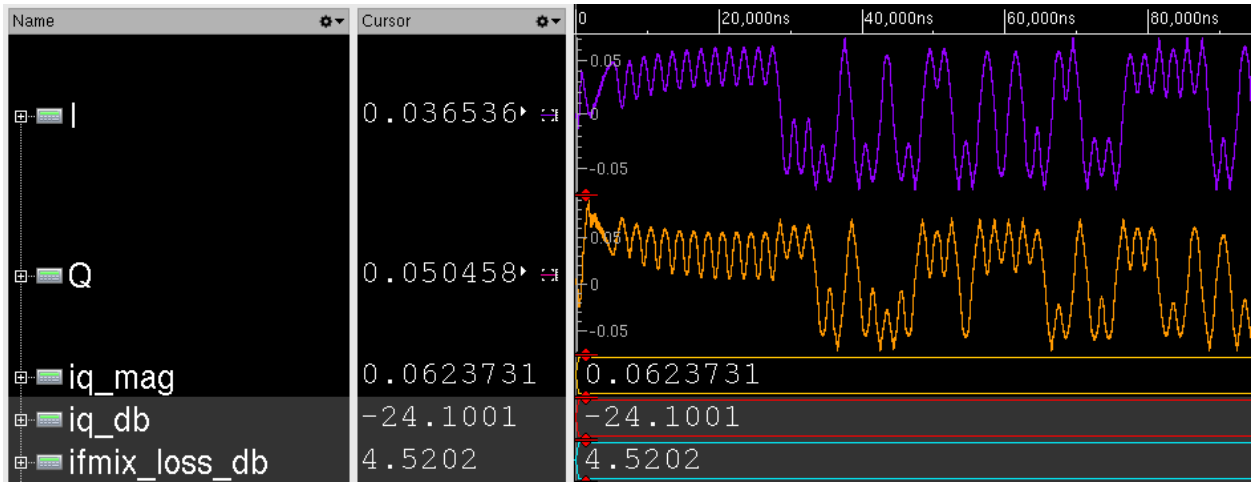

Figure 13: Second down conversion loss

We therefore see a 11.1dB loss across the mixer. To implement the programmable end to end gains of 3, 7 and 10dB, actual factors of 14.1, 18.1 and 21.1dB are used in the code.

C. Base Band Filter

The Base Band Filter (BBF) digital model approximates a 4th order analog Butterworth filter used in the RX signal chain. The BBF also contains models for the required Offset Correction Loop (OCL) and gain decoders. The

sampling rate of the filter is set to 1 GHz which gives good simulation performance while approximating a continuous time filter.

1) Filter Design.

The BBF has a programmable bandwidth so coefficients are generated for each bandwidth settings using the butter() function in MATLAB

```
%> [b, a] = butter(4, 2*f_cuttoff/fs)
```

where f_cutoff is the cutoff frequency and fs is 1GHz. Cutoff frequencies for all bandwidth settings can be found in TABLE *1* (TT 27C, fast and slow corners are not currently implemented).

With the transfer function generated by the butter function, two 2nd order sections are generated using the tf2sos function

```
%> [sos, g] = tf2sos(b, a, 'inf');
```

where b and a are the numerators and denominators of the transfer function obtained by butter(). The coefficients and gain generated map to the DF2 structure as shown in Figure 14: DF2 structure.

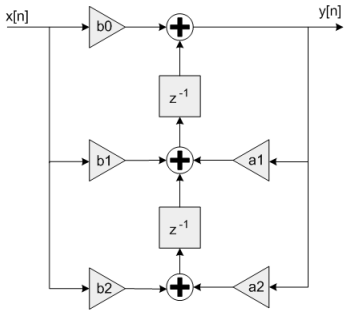

Figure 14: DF2 structure

The following verilog code implements the above DF2 structure.

```
// first biquad
always@(posedge clk) begin
    x01 <= in0*b01 - out0*a01 + x02;
    x02 <= in0*b02 - out0*a02;
end

always@(x01)
  out0 = (in0*b00+x01);n
assign in1 = out0;

// second biquad
always@(posedge clk) begin
    x11 <= in1*b11 - out1*a11 + x12;
    x12 <= in1*b12 - out1*a12;
end

always@(x11)
  out1 = (in1*b10+x11);

assign out = g*out1;
```

2) Offset and Offset Correction Loop (OCL)

Random offsets are implemented using the testbench uvm configuration class, where offsets for I and q channels are randomized within constraints. The offset is fetched from the config db and assigned to the I filter and Q filters.

The OCL DAC is modeled as a simple code to real converter. The code step is 287mV. Note that the DAC is differential.

```
always @* begin
  if(DAC_EN & pwr_ok)
    offset = ($itor(DACIN)-127.5)*0.00287;
  else
    offset = 0.0;
end

assign DACOUTP = offset/2.0;
assign DACOUTN = -offset/2.0;
```

3) Simulation Results

The filter model was simulated independently and its frequency response was analyzed. All bandwidth settings were simulated in a verilog environment where a single tone was provided at 5 kHz increment between 100 kHz and 10 MHz. Figure 15 shows the frequency response of each of the 64 bandwidth settings.

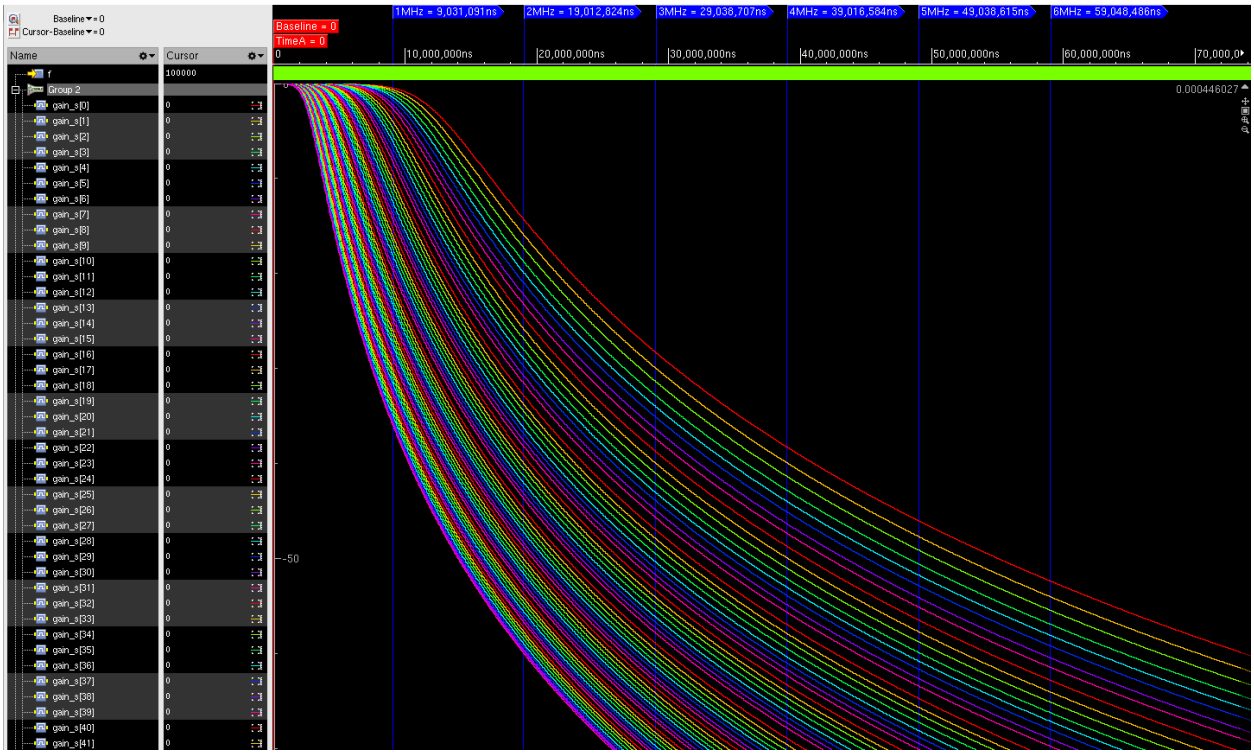

Figure 15: Frequency response across bandwidth

Table 1 shows the simulated cutoff frequencies, rounded to 1 kHz.

TABLE 1:
BBF BANDWIDTH

CODE	TT , 27C, SPICE	VERILOG	ERROR	CODE	TT , 27C, SPICE	VERILOG	ERROR
0	1328101.81	1314000	1.06%	32	549165.9323	549000	0.03%

1	1266196.628	1254000	0.96%	33	538544.5765	534000	0.84%
2	1215326.145	1213000	0.19%	34	528890.941	529000	0.02%
3	1164995.92	1154000	0.94%	35	519364.5579	514000	1.03%
4	1126444.108	1114000	1.10%	36	511626.5409	512000	0.07%
5	1091262.119	1091000	0.02%	37	504320.3792	505000	0.13%
6	1052428.108	1052000	0.04%	38	495696.6199	493000	0.54%
7	1022473.971	1014000	0.83%	39	489140.6319	489000	0.03%
8	983100.803	974000	0.93%	40	480230.5068	475000	1.09%
9	948733.0974	949000	0.03%	41	472049.5501	472000	0.01%
10	920005.7182	913000	0.76%	42	464693.6563	465000	0.07%
11	891016.0856	891000	0.00%	43	457409.7114	454000	0.75%
12	867963.5136	868000	0.00%	44	451242.0616	451000	0.05%
13	846792.9633	847000	0.02%	45	445671.1226	446000	0.07%
14	823225.5478	814000	1.12%	46	438910.033	434000	1.12%
15	804601.1181	795000	1.19%	47	433675.5673	433000	0.16%
16	775181.1703	773000	0.28%	48	424790.1806	425000	0.05%
17	754615.3917	753000	0.21%	49	418719.7235	414000	1.13%
18	739042.868	733000	0.82%	50	413700.3972	413000	0.17%
19	720930.8281	714000	0.96%	51	408182.0897	408000	0.04%
20	702996.72	695000	1.14%	52	402412.4104	403000	0.15%
21	689829.5004	690000	0.02%	53	398256.7737	394000	1.07%
22	676246.9466	673000	0.48%	54	393476.3285	393000	0.12%
23	664113.1658	664000	0.02%	55	389574.1767	390000	0.11%
24	645797.0399	646000	0.03%	56	383285.4778	384000	0.19%
25	631496.2679	632000	0.08%	57	378359.7957	374000	1.15%
26	620409.0867	614000	1.03%	58	374269.8676	373000	0.34%
27	607585.4109	608000	0.07%	59	369713.025	370000	0.08%
28	594834.6272	593000	0.31%	60	365039.5804	365000	0.01%
29	585340.1136	585000	0.06%	61	361551.211	362000	0.12%
30	575587.3503	573000	0.45%	62	357669.2521	354000	1.03%
31	566711.7214	567000	0.05%	63	354452.2802	353000	0.41%

The step response of the filter is shows in Figure 16 for bandwidth code 8. A group delay of about 2 us can be seen as per circuit specification.

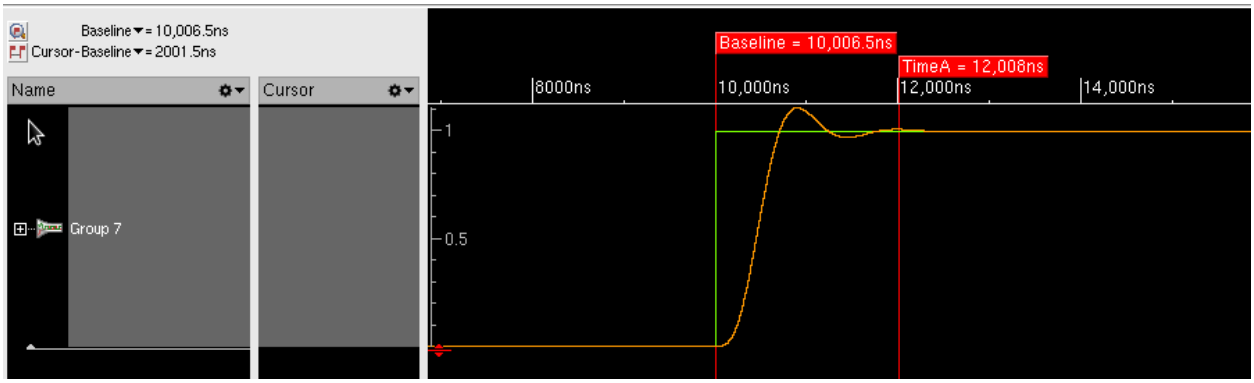

Figure 16: Step response

4) Filter Gain

The BBF also features programmable gains of up to 30dB. However, the filter itself should exhibit no gain. This can be confirmed in simulation when comparing the IQ vector magnitude at the input and output of the BBF where the loss is shown to be negligible (0.002dB)

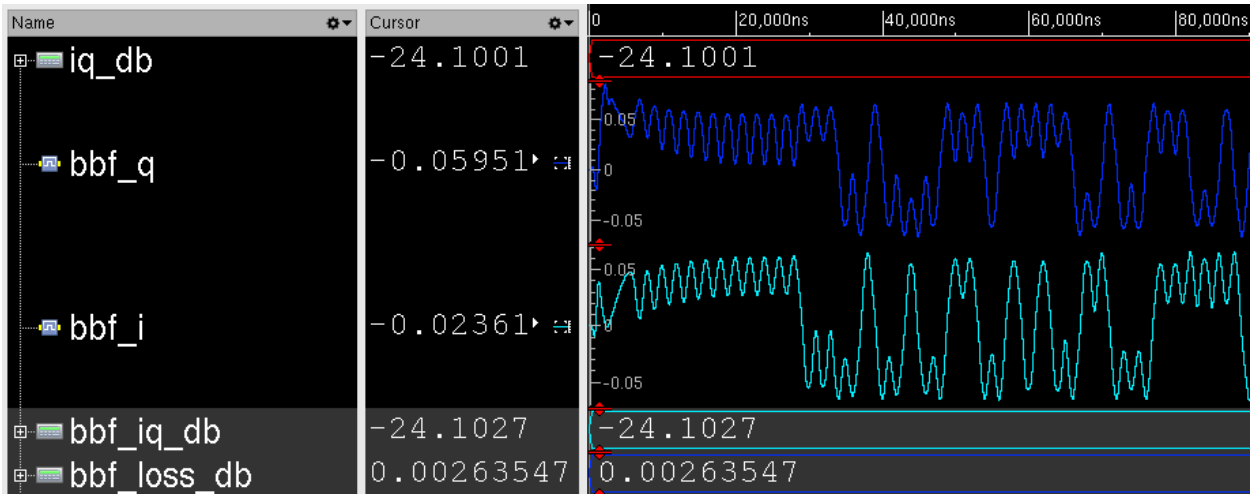

Figure 17: BBF gain

5) A note on sampling rate

The BBF will introduce aliasing into the baseband due to it sampling signals with attenuated high frequency components. In particular this will have an impact on performance in simulation with interferers. Adjacent channel attenuation will not meet specification and incorrect circuit behavior can occur in simulations with blockers.

The higher the sampling rate of the BBF, the better the adjacent channel response. However, as we push into high GHz sampling rates, not only does simulation time suffer, but coefficient start losing precision as fs/fc becomes increasingly small. For example, with an fc around 500 kHz and a sampling rate around 10 GHz, fc/fs is 0.00005 with filter coefficients in order of 10^{-16}. Filter stability also becomes a concern in these regions. The RX chain real number modeling remains limited to only provide partial support for simulations with interferers. While it is possible to have simulation working correctly, when blocker power is near maximum specification, errors start to occur, and signals are not recovered properly.

V. System Connectivity

Verifying system connectivity in a fully integrated RF transceiver (TRX) SoC is a crucial task in the overall verification effort. In a typical integrated RF TRX SoC there are many hundreds of static and dynamic control signals which cross the analog-digital boundary as well as a myriad of power domains, external pad connections and

high frequency TX and RX signal chains. A key component to verifying system connectivity is addressed in each block RNM where a supply, control and bias connectivity check is implemented within each module. This ensures that if key power supplies, static controls or bias connections are not present the model will gate the block's RF/analog output preventing correct reception or transmission of packet data. Robust connectivity checking in each block RNM across the SoC, sub-system and inter-block connectivity is tackled from the earliest phase of the development cycle via an automatic netlisting strategy. The netlist strategy comprises of a number of steps to ensure all static and dynamic analog-digital connections are in place as intended by each block designer as well as all connections to the external pads and across multiple analog subsystems. The steps can be summarized as follows:

- Ensure the top level 'config' view captures the desired analog RNM views at the block level
- Automatically place all static and dynamic registers which control analog blocks as pins on the digital sub-system symbol
- Automatically connect symbol pins by name across the chip top level schematic view
- Any registers not present and/or connected at the chip top level are flagged
- Netlister will proceed if no errors are flagged during the automatic flow from Register → Digital Symbol → Chip Top Schematic

Implementation of the above strategy early in the design cycle can flush out connectivity bugs and ensure connectivity is automatically verified to the RNM block level before any simulations are carried out.

VI. Simulation Performance

It is of interest to analyze simulation performance when it comes to models which are effectively clocked in the multiple GHz range. While it may appear as though these will create a significant load for the simulator, it is important to note that high frequencies are most problematic when they generate many events. If very few events are generated at high speed, then it is perfectly possible that the RTL, which runs at far slower rates will actually demand more simulation time because of the sheer numbers of registers to clock.

Profiling runs were done with basic TX and RX test cases to analyze the simulation impact of the RF models. Sample results are shown in Figure 18 and Figure 19.

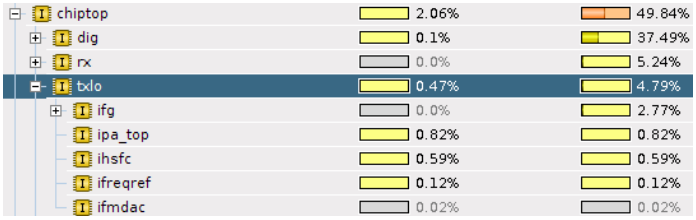

Figure 18: TX simulation performance

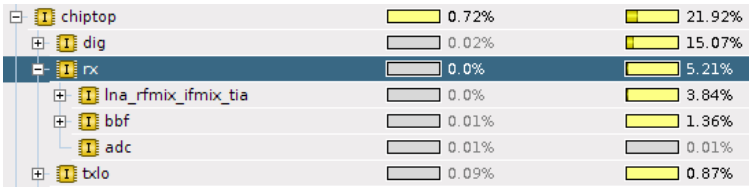

Figure 19: RX simulation performance

It can be seen from these profile run that GHz models do have a non-negligible impact on simulation time. In both TX and TX test cases, the models take around 5% of simulation resources. Note however the simulation resource required by the RTL. The resource drain for TX and RX are 37% and 15% respectively. The simulation bottleneck is thus in the processing of the RTL, rather than the RF models. This in fact confirms what has been seen throughout the work done, where many thousands of random constrained simulation were run using RF models with a manageable impact on simulation time.

VII. LIMITATIONS

The current versions of the model do have performance limitations. A specific example involves an RX simulation with a blocker. For the purpose of illustration, a sim is run with a carrier wave only blocker 3 channels away from the wanted signal. No wanted signal is present. After down conversion the expected blocker is expected to be found at 6MHz (3 channels of 2MHz bandwidth away from DC). An FFT plot of the captured output is shown in Figure 20

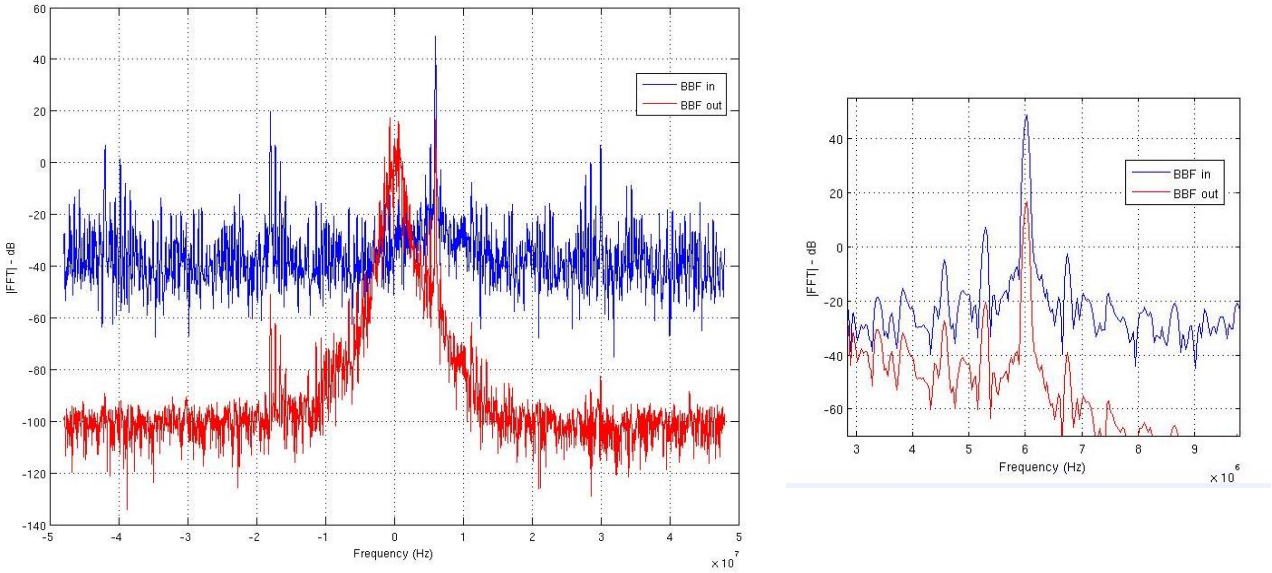

Figure 20: Spectrum of input and output of BBF

The expected tone can be found exactly at 6MHz (blue plot). The BBF also applies the expected 30dB attenuation at 6MHz (note the BBF gain is set to 30dB, attenuation at 6MHz is 60dB for an effective attenuation of 30dB). However it is clear that non negligible spectral artifacts are present at DC around -60dB in relation to the input tone.

If a wanted signal was present, it would be down converted to DC, mixing with the artifacts added by the models. It is confirmed that simulation with blockers lack robustness with many falsely detected packets. Simulations work reliably with blockers if the blocker power is well below the maximum power allowed by the specification. Thus it is still possible to functionally exercise blockers with the proposed model, but not to test their performance to specification.

The above artifacts are under investigation, but likely culprits are related to aliasing and non-uniform sampling rates in the system. Starting with a simple carrier, the input can be seen as a single tone sampled uniformly at 2 times the carrier frequency. When modulation is added, edges move by as little as a few kHz effectively generating a non-uniformly sample waveform. At the output of the mixer model, some signals can change as fast as a few picoseconds.

Figure 21: Mixer output. Note the fast 6ps pulse

The BBF would have to have a sampling rate in the high GHz to avoid aliasing. Such speeds are clearly impractical. Improving this shortcoming is among the goals for future work surrounding RF verilog models.

In general, model performance continues to be under evaluation. From a functional system level stand point, no major limitations were found with the exception of blocker simulation

VIII. Model Validation

Model validation can take different forms, depending on the intended use of the models. For instance, models used for tape out sign off require a robust comparison with transistor behavior. Specific points of comparison and error tolerance need to be agreed upon by system and analog engineers.

A second approach consists of validating the model against the analog specification. For instance a PLL can have a specified maximum FSK error, lock time, vco tuning voltage range, frequency error, etc. Models can be tested against these parameters and acceptable levels of performance can be agreed upon. Test plans can also be devised for models and a model must be fully tested before it can be used in simulations that gate tape out.

The purpose of the RNM models is to enable system level testing. Transistor level simulation and analog model co-simulation exist for the purpose of signing off. As such, the requirements for model validation are greatly relaxed in this environment. Models are validate via system level behavior, namely the ability to reliably transmit and receive packets

For the TX path, this mainly means a PLL locked to a specified frequency within a given time and below a given frequency error. The specification for the PLL circuit is a lock time of 40us and frequency error of 40ppm. Checkers are implemented to measure these metrics in each simulation.

For the RX path, the ability to down convert the incoming data and recover the transmitted bits is the main criteria. The ability to reliably operate control loops such as CFO, AFC and AGC is also of importance, and checkers are put in place to ensure these control loops operate with the specified switching points, gain settings, etc…

Model validation remains an area for improvement in the real number modeling flow. However given their status as system level testing enablers, other types of simulations are used to ensure performance and any connectivity that may not be covered by real number models.

IX. Conclusion

The purpose of this paper has been to demonstrate the feasibility of using real number models for a GHz radio transceiver. Real number models can be used both on the TX and RX side to model complex analog circuits such as PLL and mixers which can in turn be used in system level simulations. A full verilog representation of a device allows for faster test development, randomization and support of other groups such as firmware, evaluation and test.

High frequency digital models are achievable with reasonable simulation performance. It has been shown possible to construct a real phase lock loop model with a high port for modulation that performs reasonably close to the analog schematic. Similarly, RF stimulus can be used and down converted using real RF mixing code, allowing data bits to recover from a GFSK modulated RF Carrier. The simulation performance allows of run of thousands of randomized simulations daily, thus meeting a general speed criteria for verification needs.

Ongoing characterization work also aims at establishing the performance limitations of the models. Preliminary data shows promising results in terms of meeting some of the analog circuits' performance specification, such as frequency error and lock time on the PLL. In time, full understanding of the frequency domain performance of these models will prove useful.

REFERENCES

[1] Fesque, L.; Bidegaray-Fesquet, B. "IIR Digital Filtering of Non-uniformly Sampled Signals via State Representation", International Journal of Signal Processing, vol. 09, no. 10, pp. 2811-2821, 2010