A Modeling Methodology for Verifying Functionality of a Wireless Chip

Jess Chen

Qualcomm

Santa Clara

Objectives of functional verification

- Fully functional first tape out.
- Check basic functionality with no impairments.
  - Power up/down
  - Frequency control
  - Gain control
  - Mode control
  - Current consumption (time and data permitting)
  - Focus on the signal path
- Big bug safari.
  - Twisted and/or garbled digital commands
  - Missing commands
  - Connectivity errors (analog and digital)
  - Circular logic
  - Positive feedback in calibration loops
  - Registers connected properly

Non-objectives of functional verification

- Performance
  - Noise
  - Nonlinearity
- Corner analysis
  - Temperature variations
  - Process variations
- We trust the designers to deliver performance.
- Functionality is the main concern in the first tape out because if the chip is not functional, we will probably have to re-spin the chip before any performance measurements can be made.

Constraints

- Verification/modeling engineers cannot touch the schematics.
- Simulations must run in a digital design environment.
  - Lots of long tests to run.
  - Diagnostics requires even more runs.
  - Top level test pilots live and breath in a digital environment.
- Models must be portable.
  - Big companies get big by acquiring lots of small companies.
  - Models must also run in the analog design environment.

Why must the models be compatible with the analog environment?

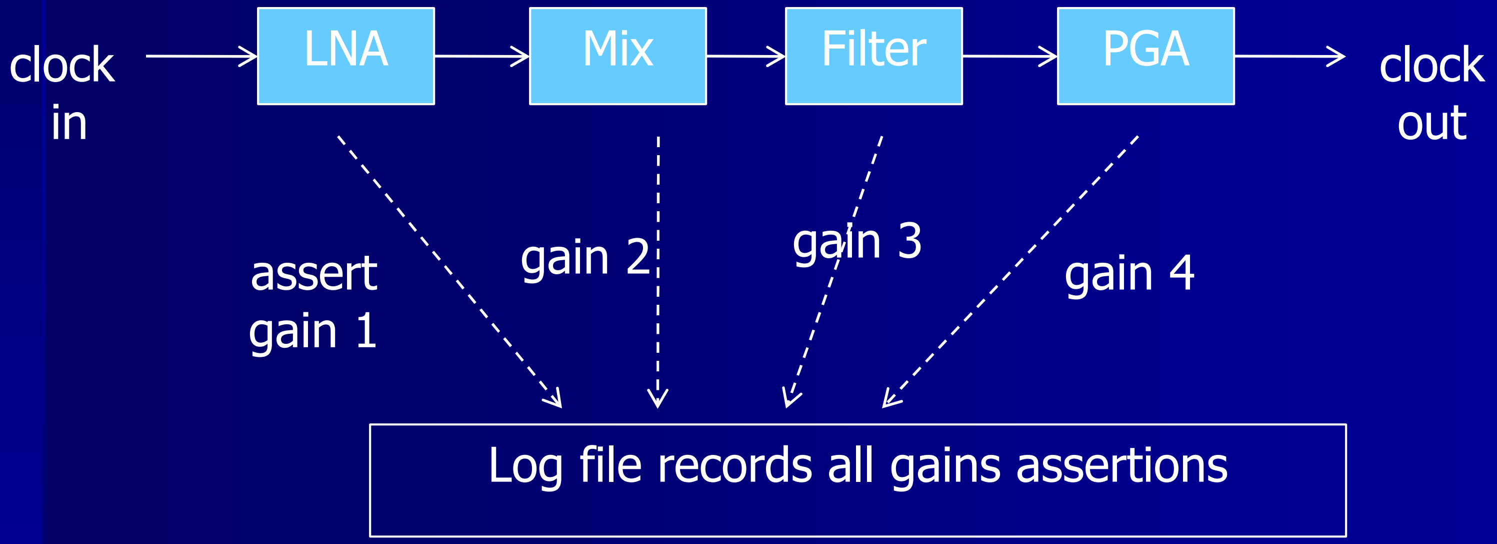
- Analog models must be hand crafted.
  - They do not automatically match their circuits.
  - The model could work while the ckt does not.
  - Side-by-side model/circuit simulation catches mismatches.
- Model/circuit swapping can accelerate diagnostics.
- Leverage the analog design team.

Main problem with event driven simulators

- How to deal with real number traffic between analog blocks?
- Digital simulators can be distinguished by how they traffic real numbers.

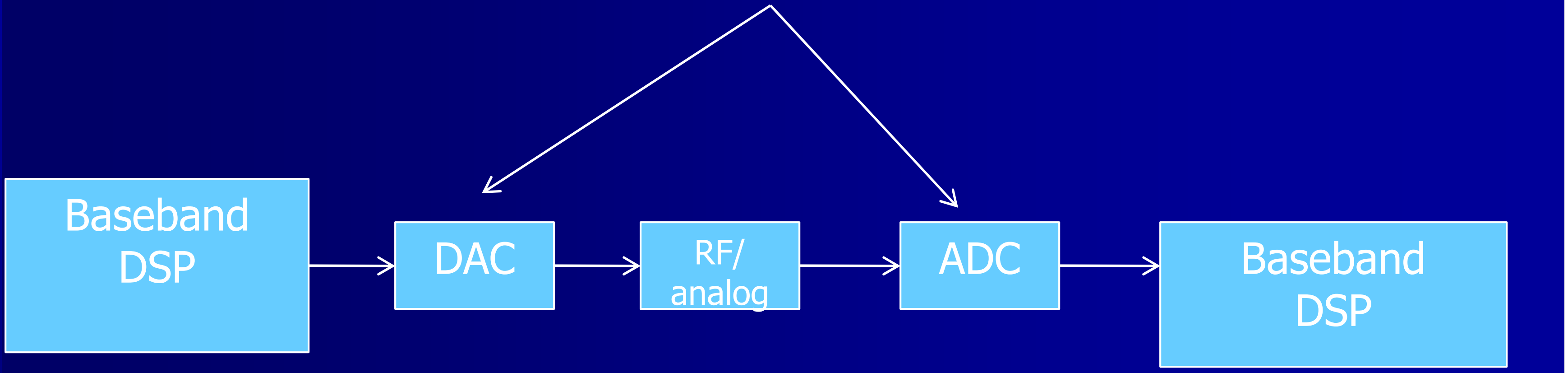
Do we need real number traffic?

- All digital, assertion-based verification.



Real numbers exercise the end-to-end system

- Symptom based verification deals with real number traffic.



Challenges with real number traffic in event driven simulators

- Parallel real number drivers
  - Voltages contend
  - Current sum
- RF real signals
  - Brute force
    - Very high sample rates.
    - Requires digital equivalent filters
  - Baseband equivalent
    - Pass multiple real numbers on a single wire.

Options for real number traffic

- Bits2real and real2bits
  - 64 bit bus
  - Telegraph bits across a single wire
  - Current summing??
- Wreals
  - One real number, one way, all the time.
  - Portability?
- VHDL and System Verilog
  - Netlisters (i.e. portable to analog environment?)
- PLI function (Designers Guide)

Key features of the Designer's Guide Verilog PLI function

- Pass real vectors between Verilog modules.
  - Vector length is unlimited.
  - Vectors can contain pseudo-electrical signals.
- Bidirectional real number traffic.
- High impedance state.
- Switch between voltage and current output on the fly.
- Resolution function for real number drivers.
  - Flag contention (voltage and current).
  - Flag an un-driven condition.
  - Sum currents.

Baseband equivalent modeling: This is why real vector traffic matters

- Baseband equivalent models require vector traffic.
- What’s good about BBeq models?
  - BBeq models run fast by suppressing the carrier but still produce realistic signals at the ADC inputs.
  - With realistic ADC inputs the entire signal path, including baseband DSP algorithms, can be verified with practical run times.
  - IQ swapping is easy to check.
  - A less important benefit is that RF impairments are easily modeled. (Functionality of calibration circuits may require impairment modeling.)
    - IQ mismatch
    - AM/AM, AM/PM, PM/AM distortion
    - Phase noise

Baseband equivalent RF signals

- $RF(t) = i(t) * \cos(\omega * t) - q(t) * \sin(\omega * t)$
- $RF(t) = \mathrm{Real}\,[\{i(t) + j * q(t)\}\, e^{j * \omega * t}]$
- $BBeq(t) = i(t) + j * q(t)$

A geometric view of modulation and demodulation (an alternative to Hilbert transforms)

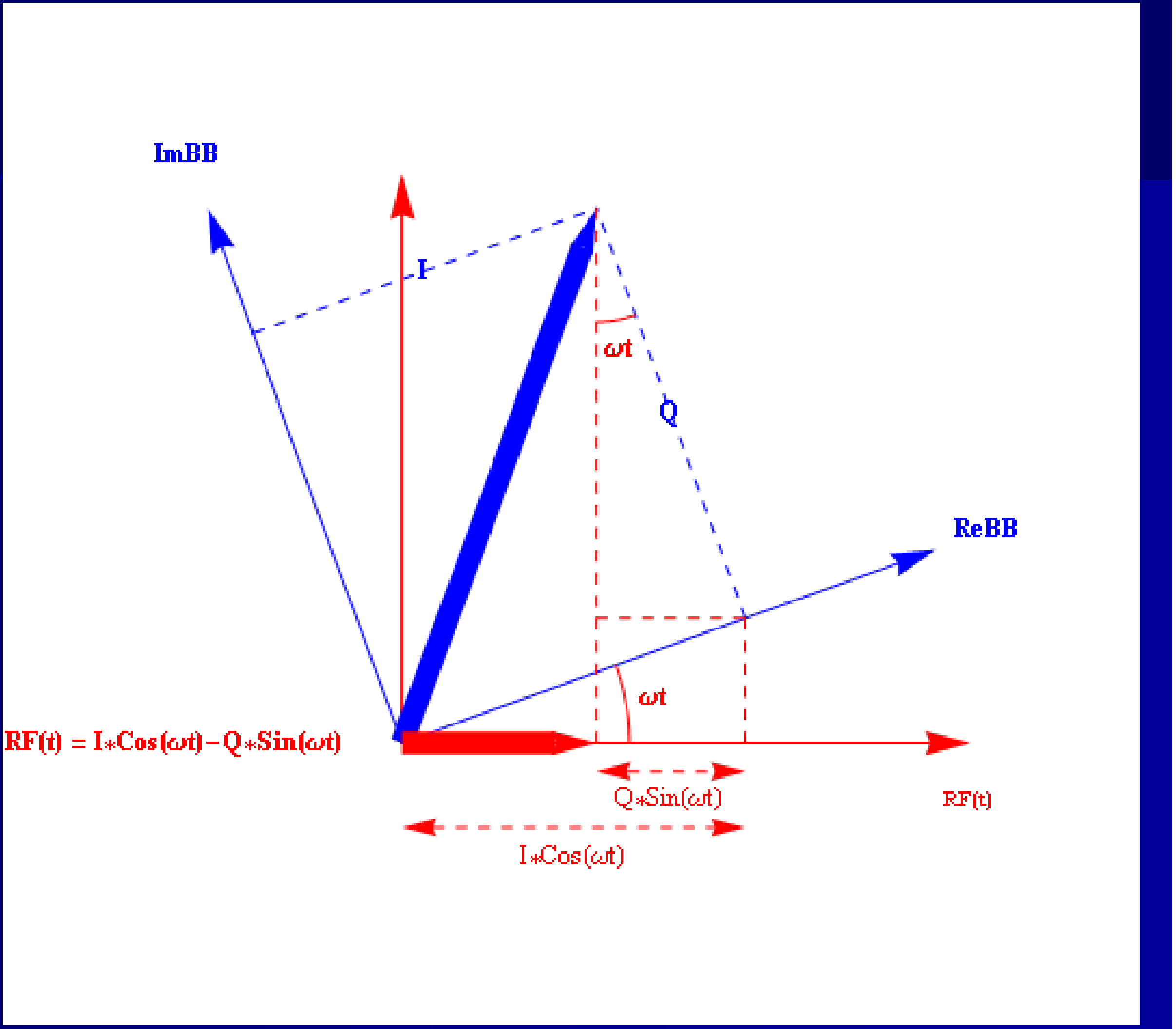
RF(t) is the projection (shadow) of the heavy blue vector onto the fixed horizontal axis.

A transmitter converts the blue IQ components into the RF signal (red projection).

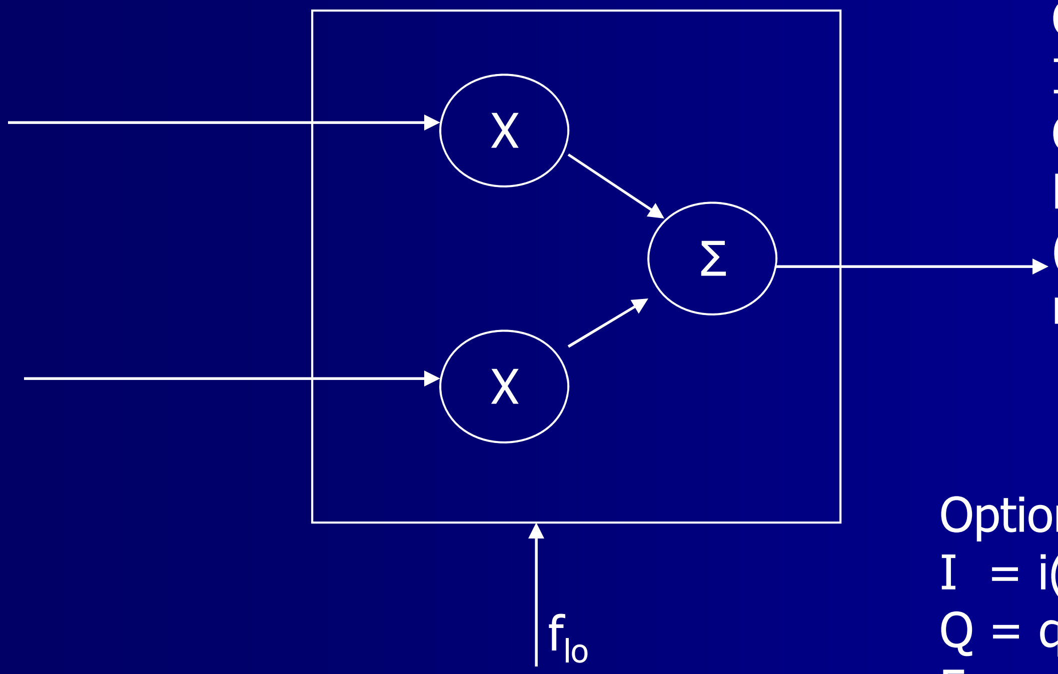
A receiver extracts IQ components from an RF signal. → ReBB-ImBB axes represent the ideal receiver output.

For direct conversion, ideally the blue vector and blue axis rotate together at ω radians per second

(Show the cool animations.)



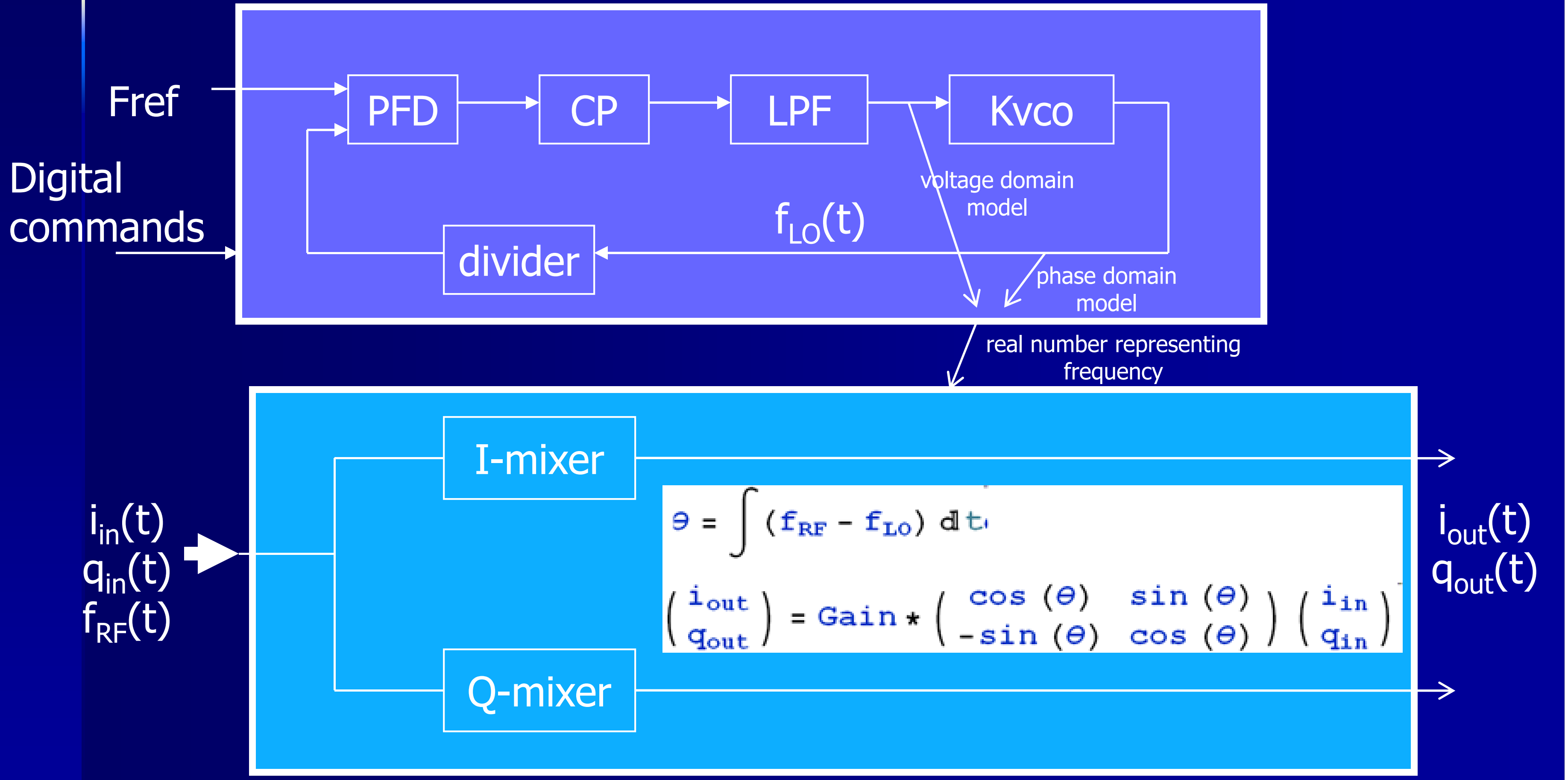
Transmitter HyperWire signals



Option 1:
I = Constant
Q = 0
Freq = f_{center}+ $\Delta f(t)$
(useful for PLL modulation)

Option 2:
I = i(t)
Q = q(t)
Freq = constant

Integrating the PLL and Receiver Mixer models



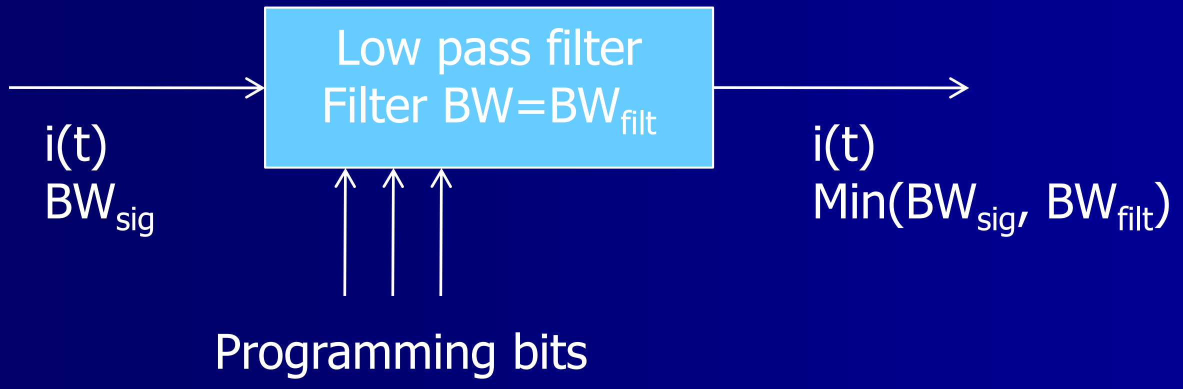
$$\theta = \int (f_{RF} - f_{LO})\, dt$$

$$\begin{pmatrix} i_{out} \\ q_{out} \end{pmatrix} = Gain * \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} i_{in} \\ q_{in} \end{pmatrix}$$

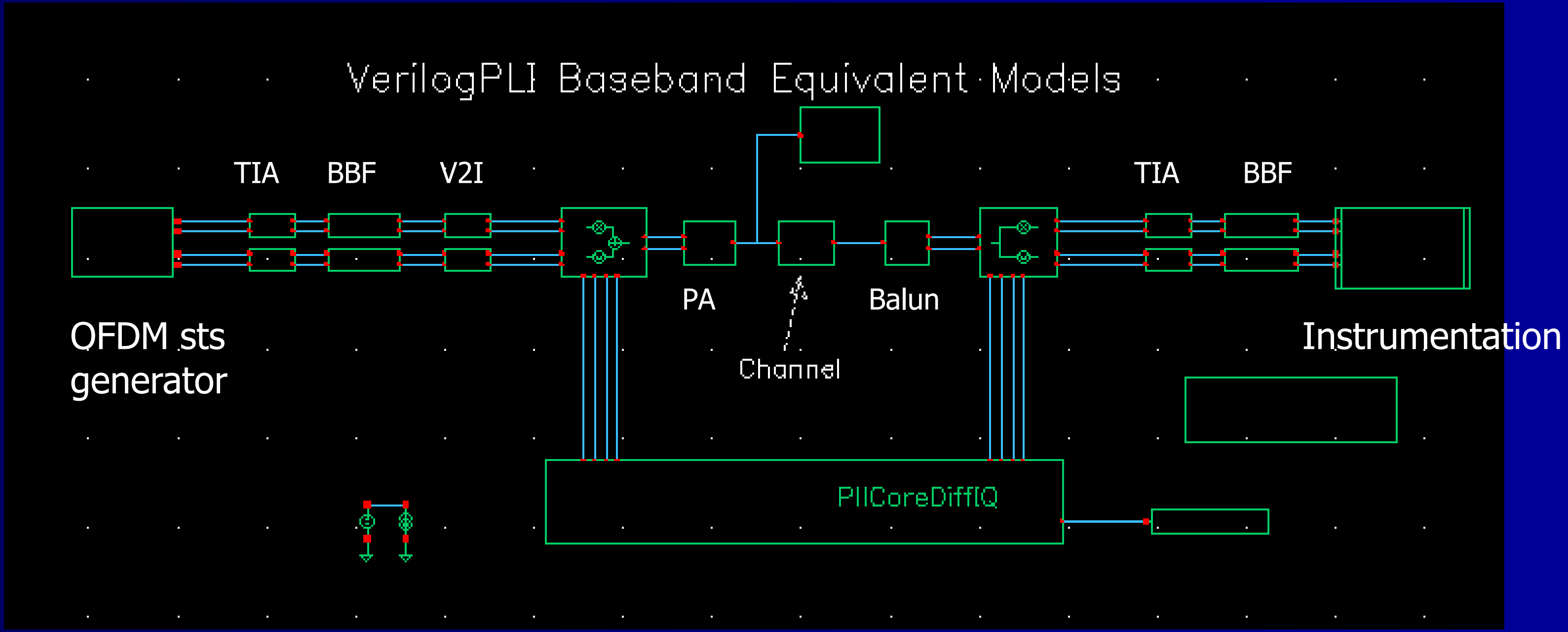
Qualcomm

The simple (preferred) filter model

- Jammers and ACPR are performance issues, not functional issues.
- Check programmable filter bandwidth.



Demonstration model (not the actual SOC)



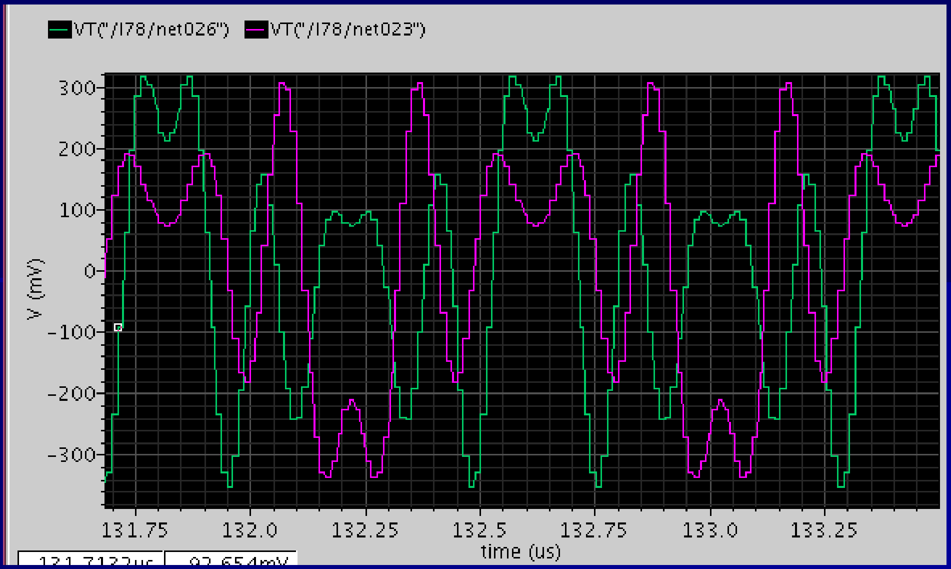
TX PLL signals are fixed (open loop).
RX PLL signals are generated from a closed loop model.

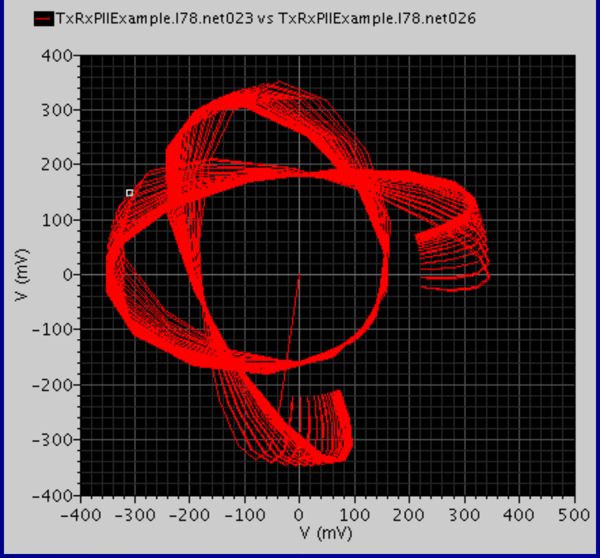
802.11a/g short training sequence pretzel

Time domain waveforms don't visually tell much of a story.

However, the IQ trajectory clearly shows the effects of frequency PLL transients on the receiver output.

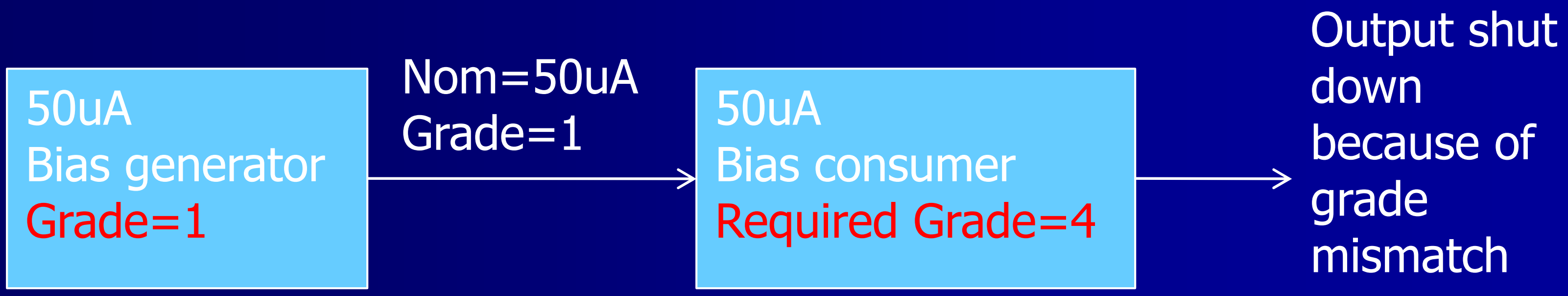
Frequency offset between the carrier and LO causes the RX output pretzel to slowly rotate. The IQ trajectory comes closer to retracing itself as the frequency offset decays to zero.





Bias currents: Another reason to pass real vectors

- Not all bias generators are created equal, even if they have the same nominal value.
- Some bias consumers can use a sloppy bias while others need a tightly controlled bias.
- A sloppy 50uA bias connected to a sensitive 50uA consumer is a rare bug but it is one that would most likely only be found by the customer.



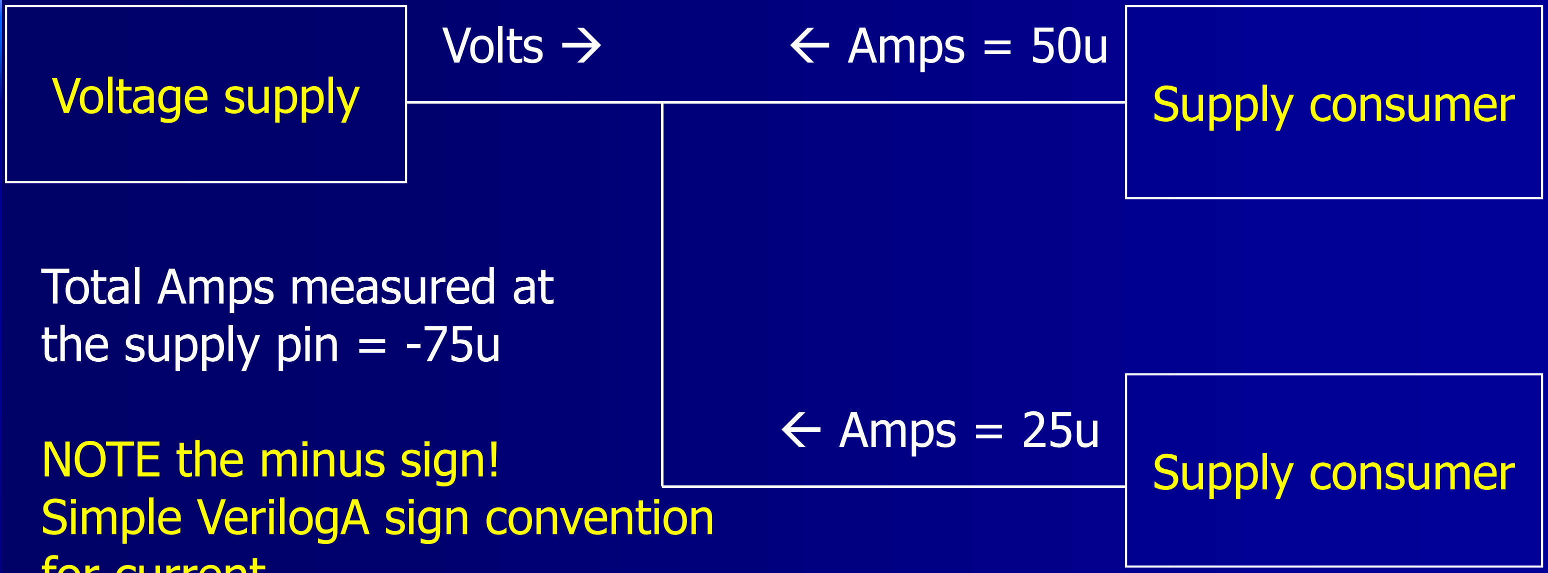
50uA
Bias generator
Grade=1

Nom=50uA
Grade=1

50uA
Bias consumer
Required Grade=4

Output shut down because of grade mismatch

Pseudo-electric signals



Voltage supply

Volts →

← Amps = 50u

Supply consumer

← Amps = 25u

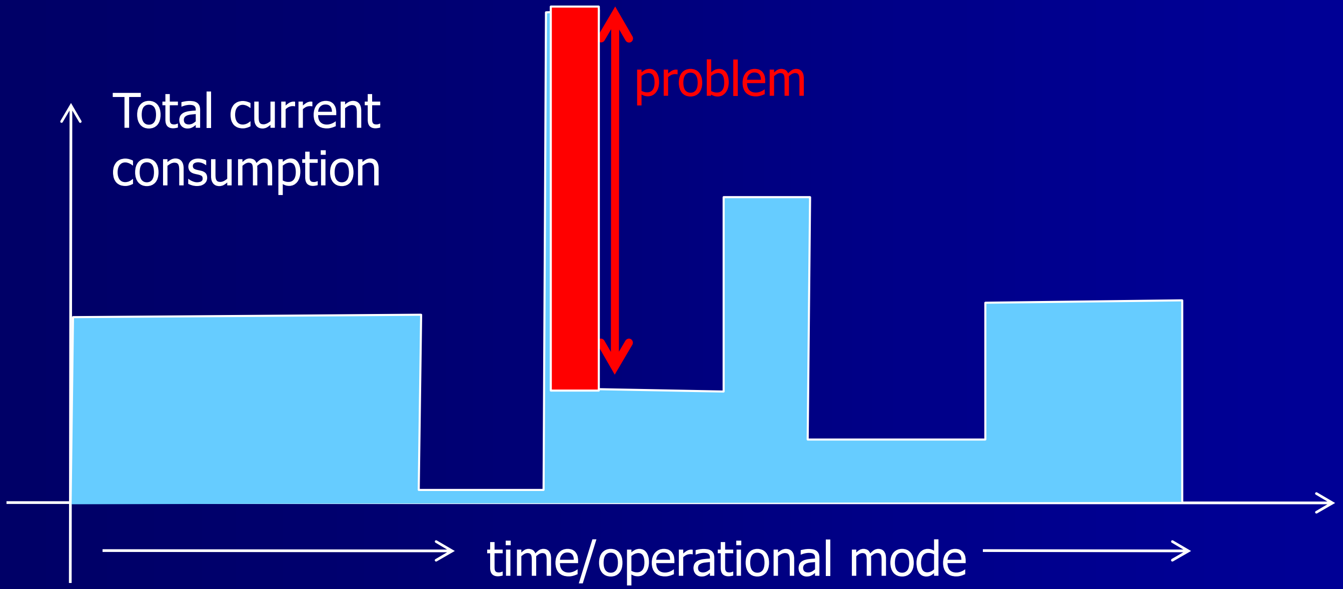
Supply consumer

Total Amps measured at the supply pin = -75u

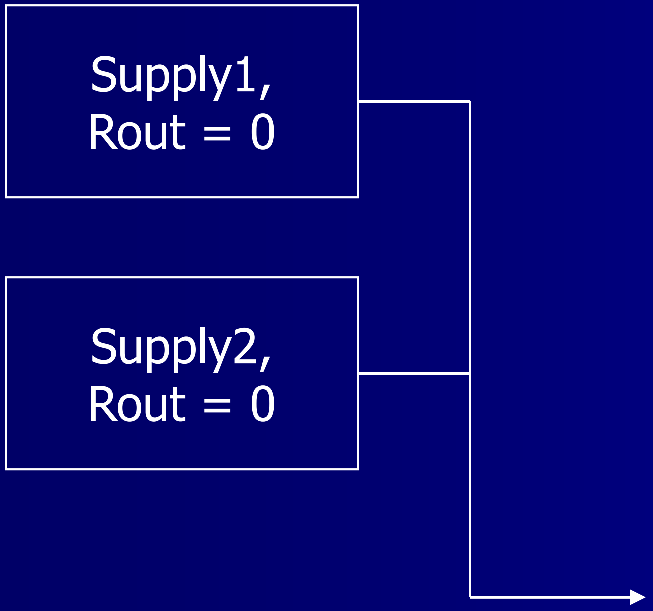
NOTE the minus sign!
Simple VerilogA sign convention for current.

Amps can depend on the Voltage. (Need a small delay to ensure causality.)
Example: if (Volts>1.1) amps = 50u; else amps = 0;

Current consumption profiles



Voltage contention (KVL violations)



Supply1,
Rout = 0

Supply2,
Rout = 0

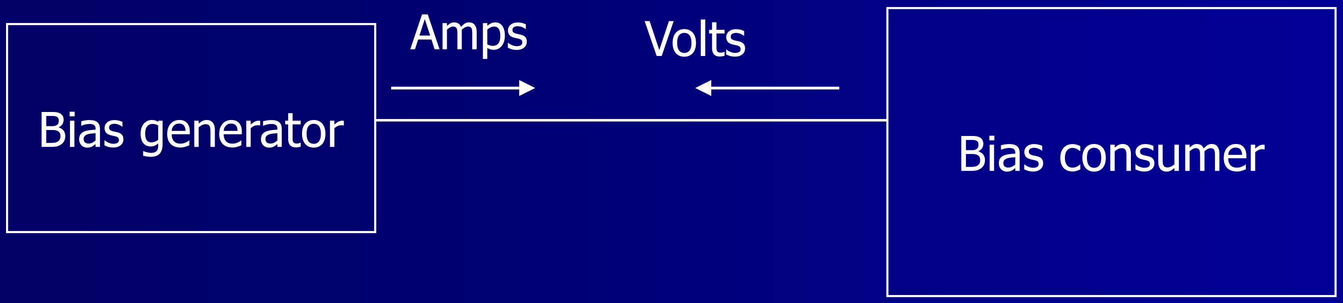
Supply1	Driven	Driven	Un-driven	Un-driven
Supply2	Driven	Un-driven	Driven	Un-driven
Wire logic state	X	0 or 1	0 or 1	Z, if no other drive exists

KCL violations

- Currents with no place to go force the logic value of the net to 1’bx.
- You will see lots of warnings at t=0 because the current sinks (voltage sources) do not necessarily get declared before the current sources.

Using voltage contention to detect a bias pirate

Normal situation.



Amps

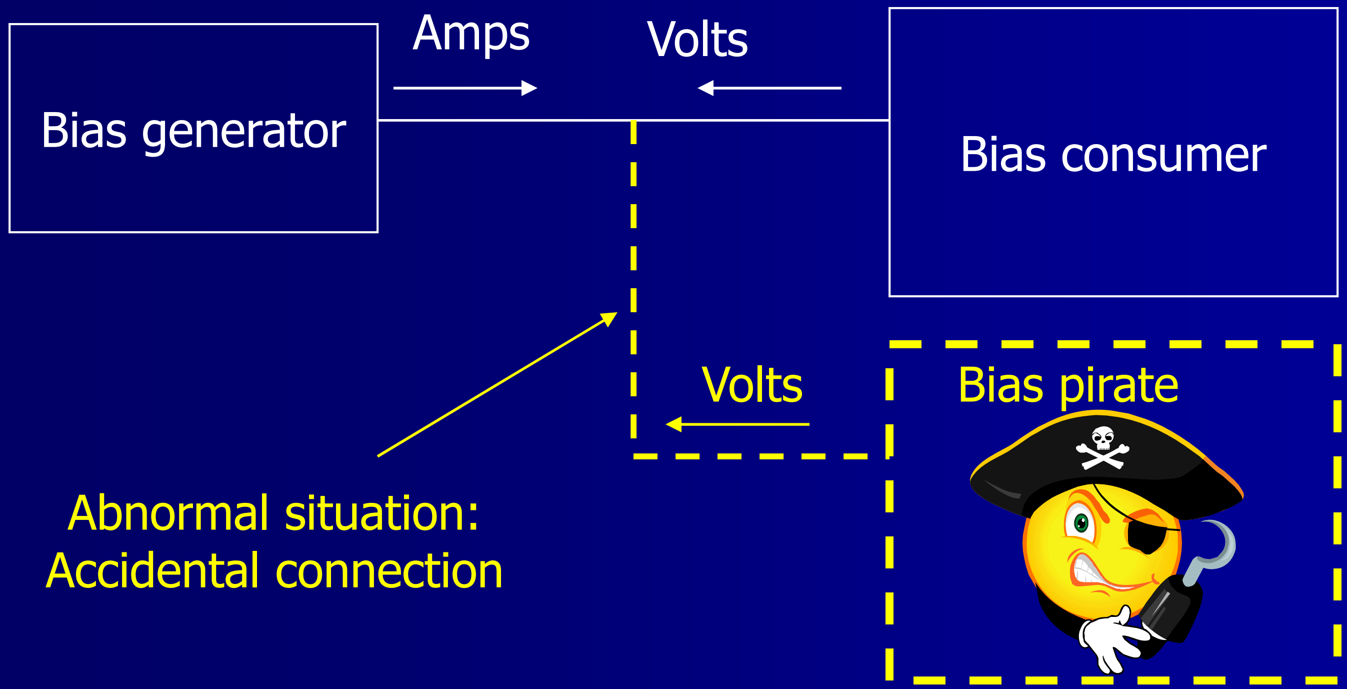
Volts

Bias generator

Bias consumer

Using voltage contention to detect a bias pirate

The testbench can look for an
X-state on the bias wire.



Abnormal situation:
Accidental connection

Conclusions

- We applied this methodology to a complete wireless SOC.
  - Run times supported broad coverage.
  - Chip verified from end-to-end.
  - Portability helped maintain schedule.
- Over 100 functional bugs found before tape out.
- Fully functional first silicon aside from three very minor and subtle bugs.