2016 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Challenges in UVM + Python Random Verification Environment for Digital Signal Processing Datapath Design

Shabbar Vejlani

Ashok Chandran

High Speed Convertor Group

Analog Devices India

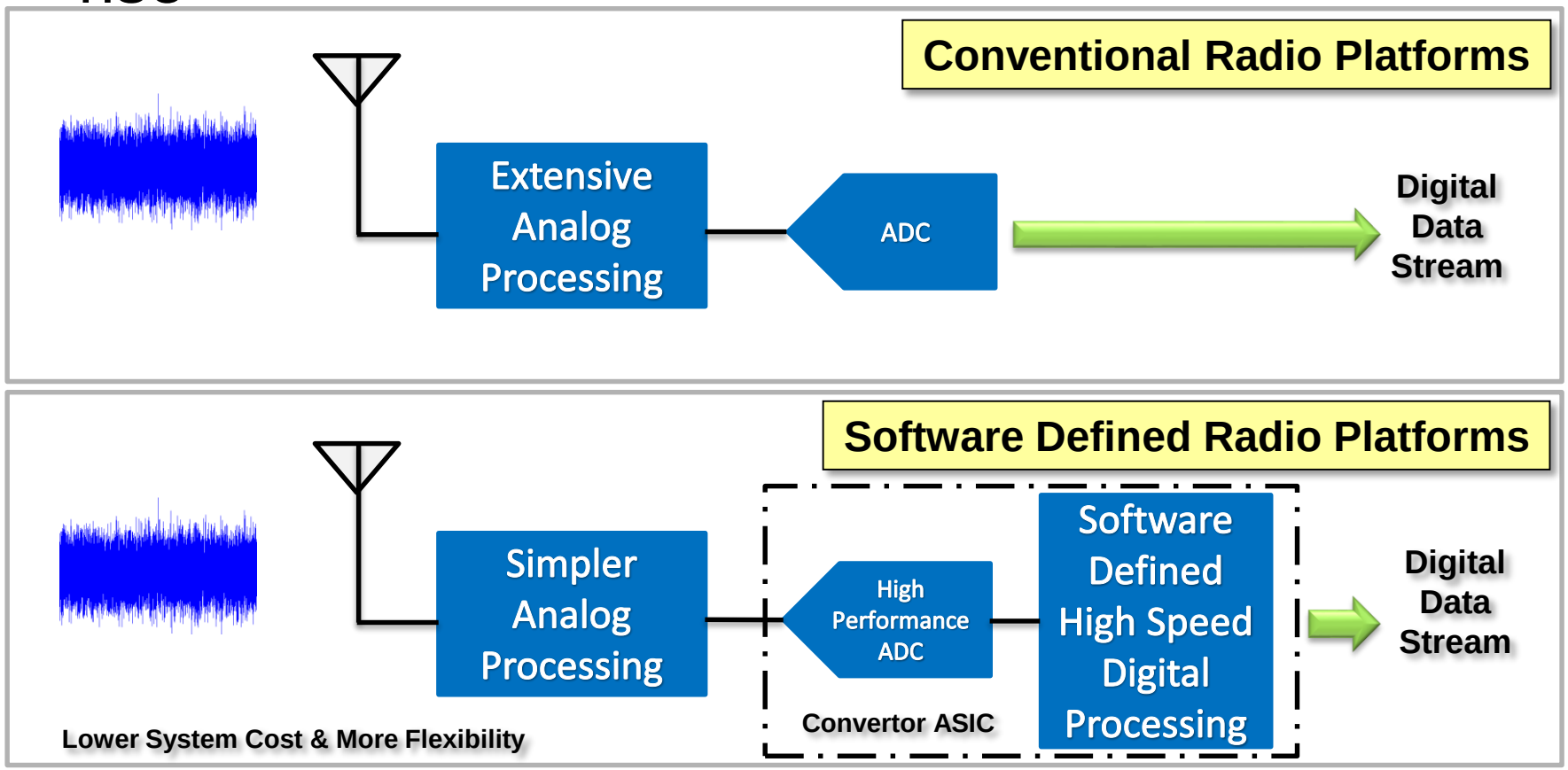
2016 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Agenda

- DSP Datapath Verification
- Traditional Approach
- Proposed Approach
  - Constrained Random on DSP
  - UVM + Python Framework
- Results and Learnings

DSP Datapath Verification

- Software Defined Radio (SDR) Platforms are on the rise



DSP Datapath in SDR

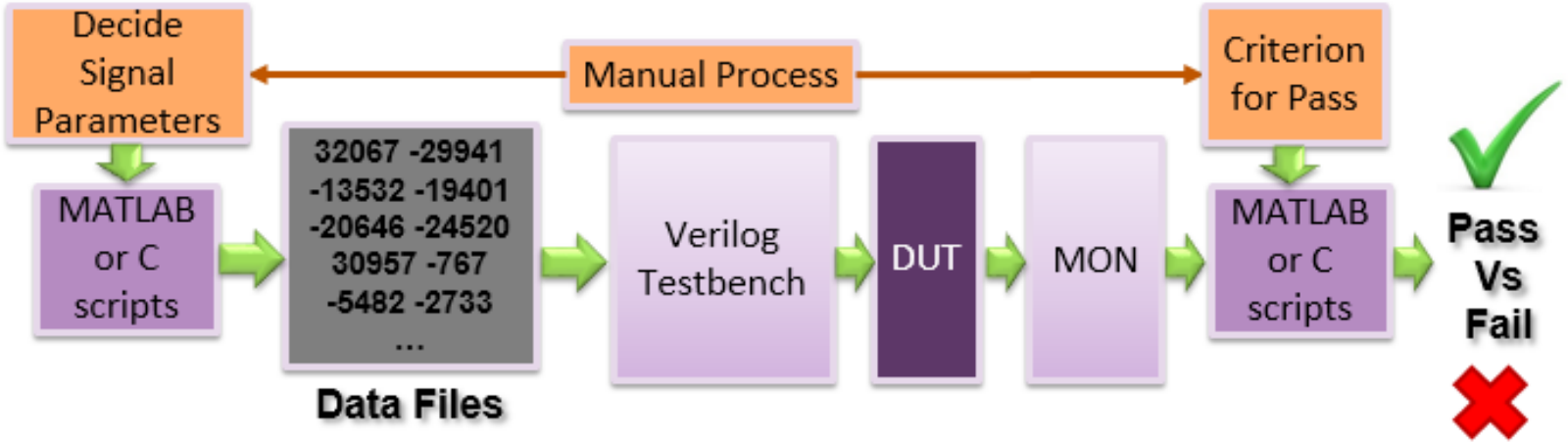
- Digital Signal Processing
  - Various Interpolation/Decimation FIR Filters for Sample rate conversion
  - Numerically Controlled Oscillators for Frequency Up/Down Shifting
  - Digital Gain Control for Signal Amplitude Scaling
- Complexity in High Speed DSP Datapaths
  - More Parallel Paths => clocking complexity
  - Programmable Filter Combinations => More muxing and control complexity
  - and more …

DSP Datapath Verification

- Frequency Domain Approach
  - Verification Goal
    - Ensure filter response to signal meets the frequency domain performance
  - Data Inputs to DUT
    - Single tone, Multi-tone , Fat band signals
  - Checks at output
    - SFDR, SNR, NSD, THD, DC, FFT bin
- Time Domain Approach
  - Verification Goal
    - Data checked against model
  - Data Inputs to DUT
    - Single tone, Multi-tone , Impulses, random, ramp signals
  - Checks at output
    - Data by data checking (with margins if needed)
    - FFT and analysis in F-domain

Traditional DSP Verification Approach

- Past Approach
  - Frequency domain verification
  - Directed Verification Methodology
  - MATLAB/C scripts for data generation
  - MATLAB/C post-process scripts for checks



Traditional DSP Verification Approach

- Disadvantages
  - Directed verification is not scalable as complexity increases
  - Quality gaps due to various trade-offs needed to meet time
- Boils down to
  - Significant Gap between problem at hand and methodology used

- Need serious upgrade in methodology

New DSP Verification Approach

- Key Highlights of new approach:
  - Apply Constrained Random Verification (CRV) on DSP Datapath
  - UVM + Python Framework

2016 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Constrained Random Verification (CRV) on DSP Datapath

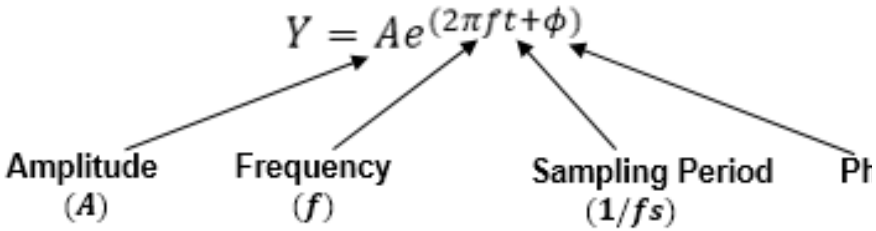
- Frequency Domain Randomization
- Frequency Domain Scoreboard

CRV on DSP Datapath

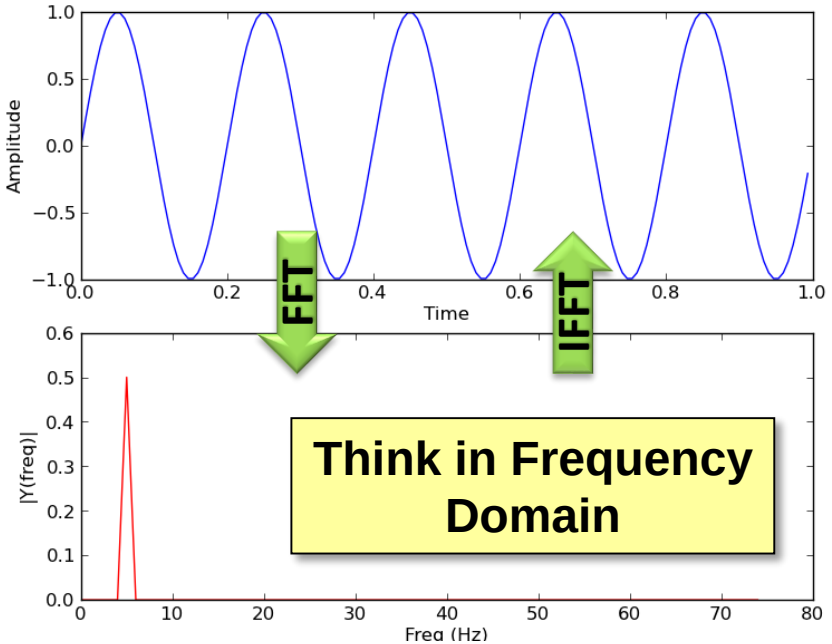
- Constrained Random Verification (CRV)
  - Principle is to use Compute Power to overcome human capacity limits
  - Constraints applied to generate intelligent stimulus
- Extending CRV to Frequency Domain
- Key challenges in doing this are in:
  - Signal Generation
  - Integrity Check / Scoreboarding
- Restricting discussion to Tone generation aspects

Random Signal Generation - Variables

- First step is to identify the random variables
  - Signal Parameters



  - FFT Parameters
    - Number of samples
    - FFT size

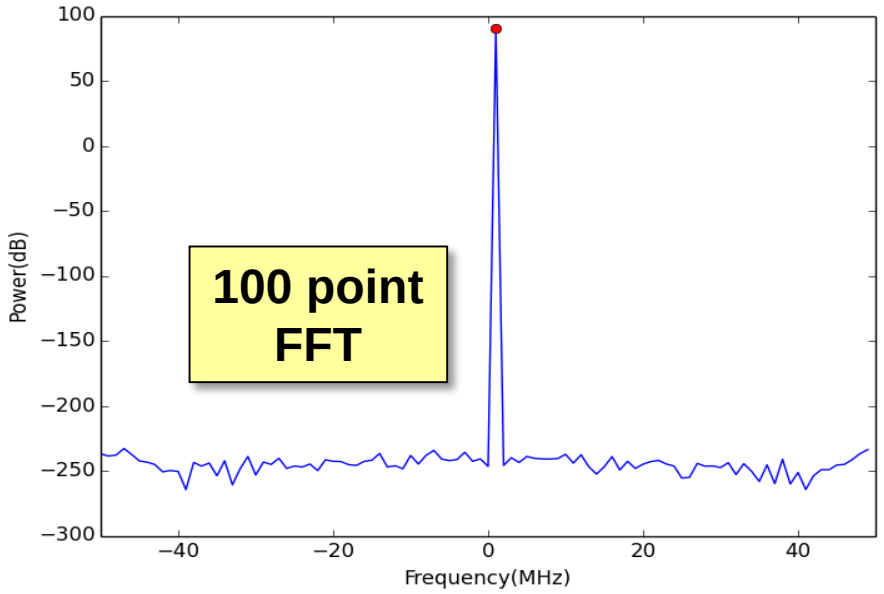


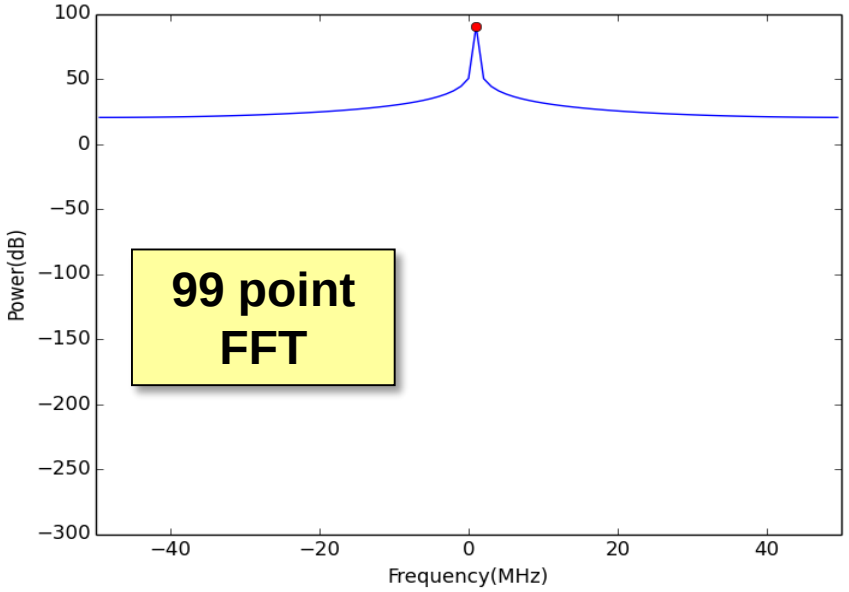
Random Signal Generation - Constraints

- What are the requirements from the signal ?
  - Should be generated for the filter specifications
  - Should be of excellent spectral quality to unmask design corner-case performance bugs
  - Should enable ease of performance measurement
- Need constraints to get a good signal generation

Random Signal Generation – Constraints (contd …)

- Spectral Leakage
  - FFT size limits the possible generated frequencies
  - Below are plots of FFT taken on a 1Mhz sine sampled at 100Mhz

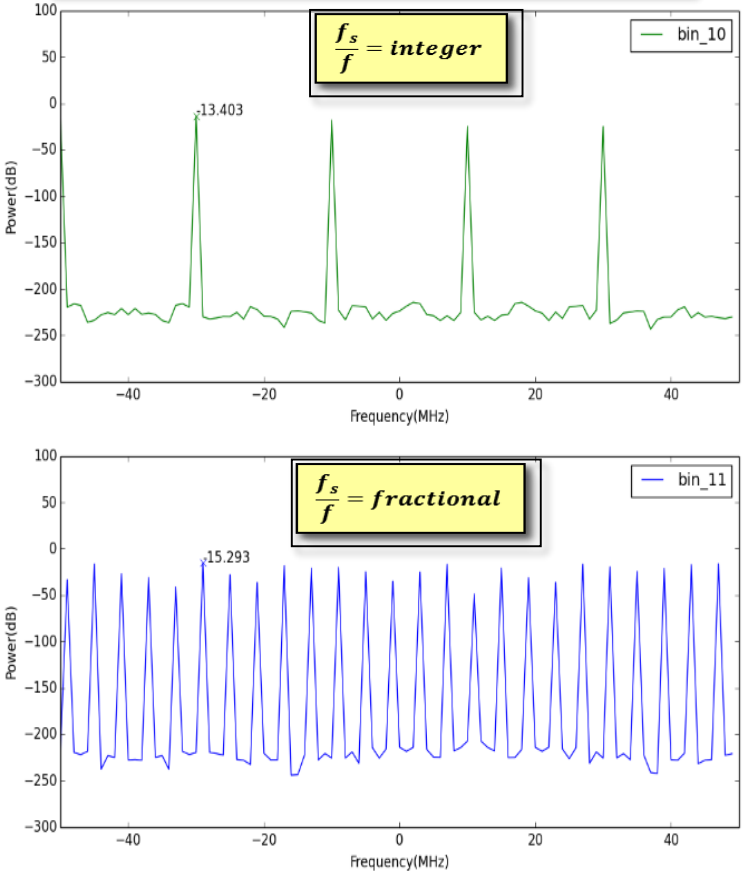




Random Signal Generation – Constraints (contd …)

- Noise Quantization
  - Quantization noise becomes periodic when sampling (fs) and tone frequency (freq) has integer relation
  - Solution - Use prime number bins or FFT size

10Mhz sine quantized to 16bits with 100 point FFT



Random Signal Generation – Constraints (contd …)

- Digital has fixed point precision (say DAC input) – Need to quantize real signals
  - Use rounding when doing real value to fixed precision conversion
  - More than 3dB improvement with rounding
- Based on design configuration, signal should fit inside bandwidth of the filters

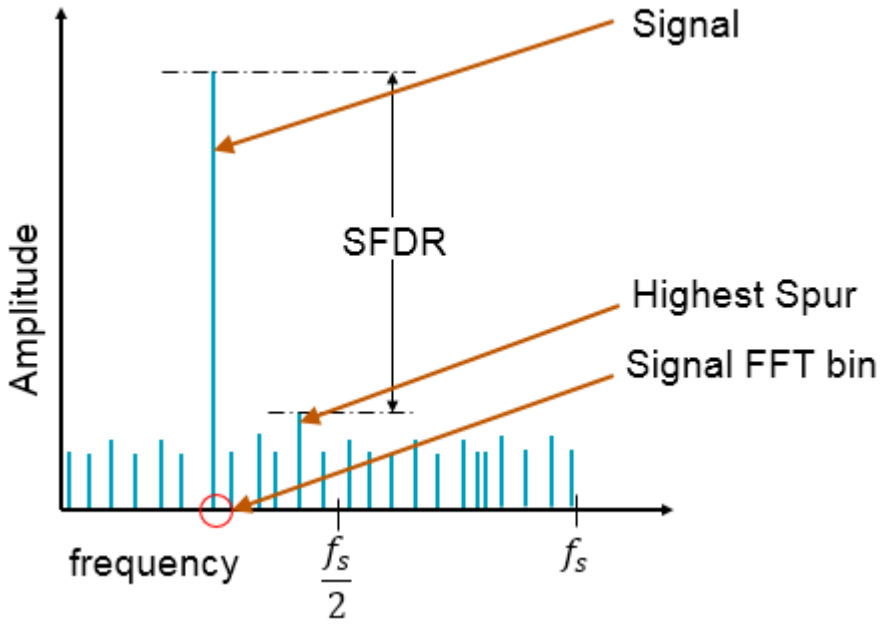
Frequency Domain Scoreboard

- Design translates the frequency domain of the input
  - Changes sampling rate, performs filtering etc.
  - Need dynamic prediction for a random signal

- Input is random => Dynamic scorboarding is needed
  - Predict the output tone FFT bin based on design mode and input random tone bin

Frequency Domain Scoreboard (contd …)

1. Check SFDR (spurious free dynamic range), SNR, THD , NSD requirement is met
2. Check o/p signal falls in expected bin predicted based on input random tone bin and design model



UVM + Python Framework

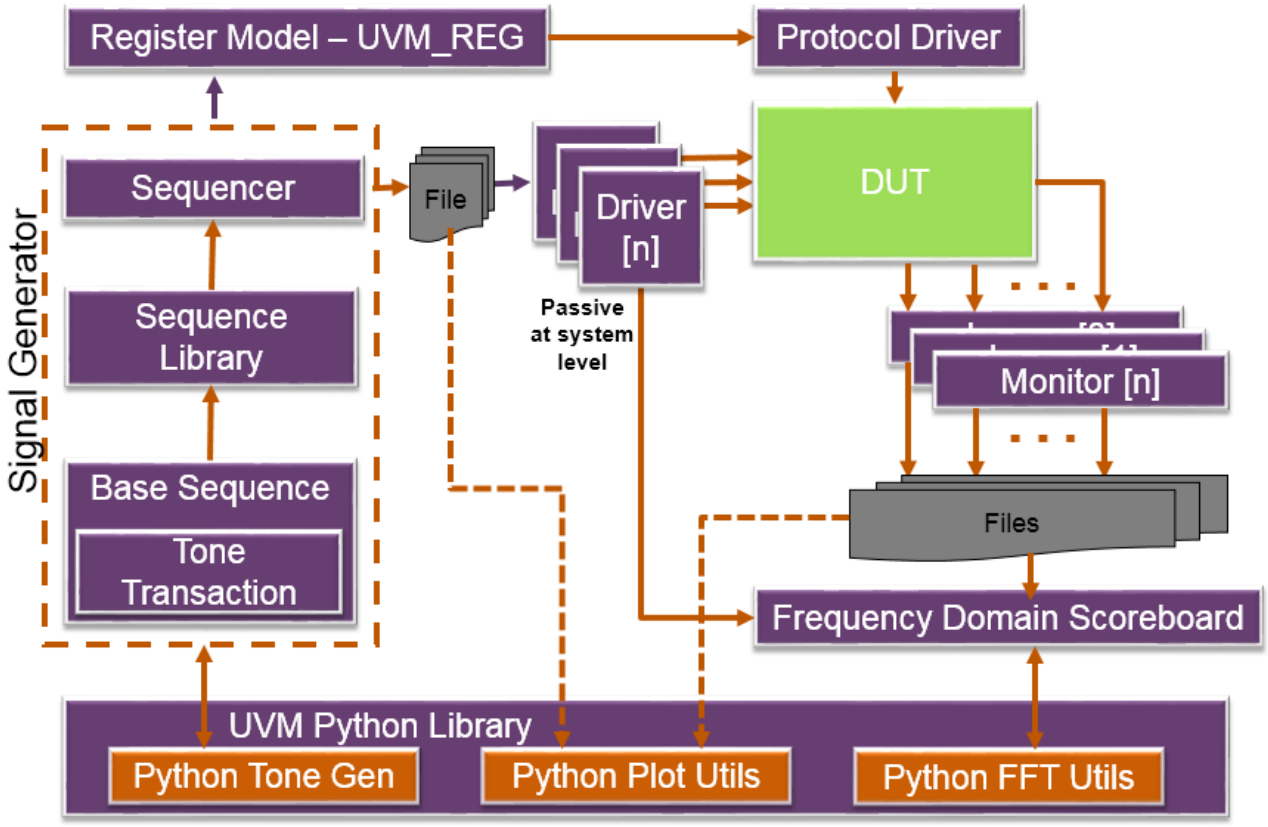
- UVM for Scalability and Block-to-System reuse
- Python for bringing signal processing power to UVM

UVM + Python Framework

- Python: Open source & powerful library support
  - *Numpy* – Array processing
  - *Scipy* – Signal processing operations
  - *Matplotlib* – Data visualization / plots
  - *Pandas* – Data analysis
  - All of these bundled under *anaconda* distribution

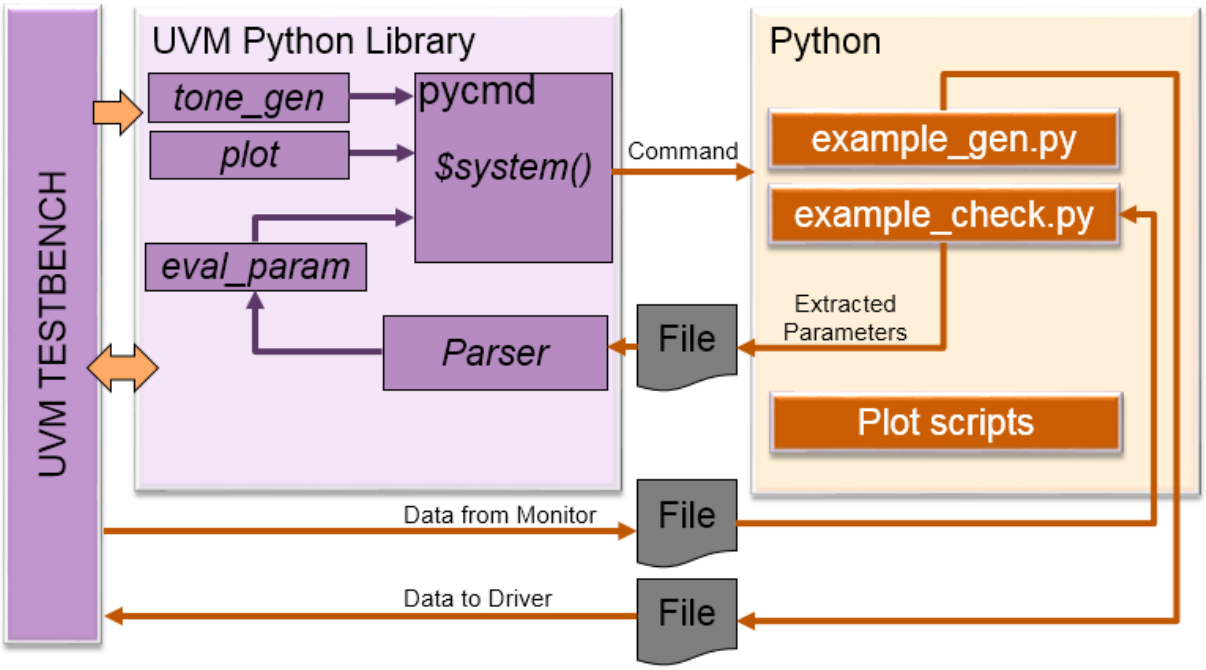
UVM + Python Framework (contd…)

- Signal files generated by sequence
- Configurable monitors and drivers to manage clocking
- Frequency domain scoreboard



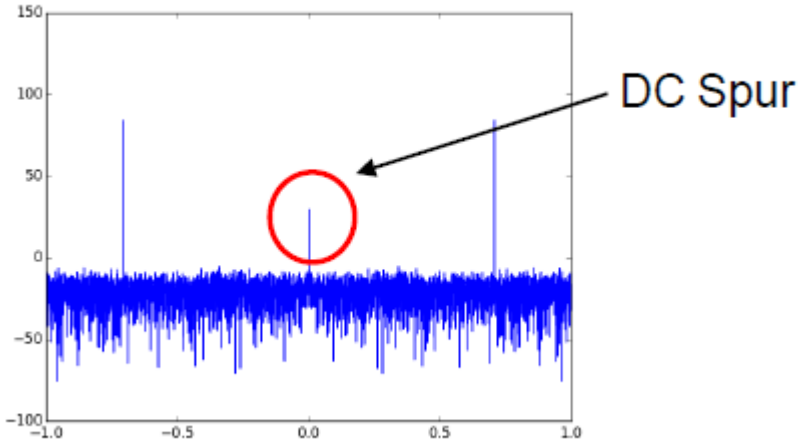
UVM + Python Framework (contd …)

- Integration to UVM
  - Verilog $system call to pass data from UVM to Python
  - Text based data handoff from Python to UVM
  - Partitioned into separate library

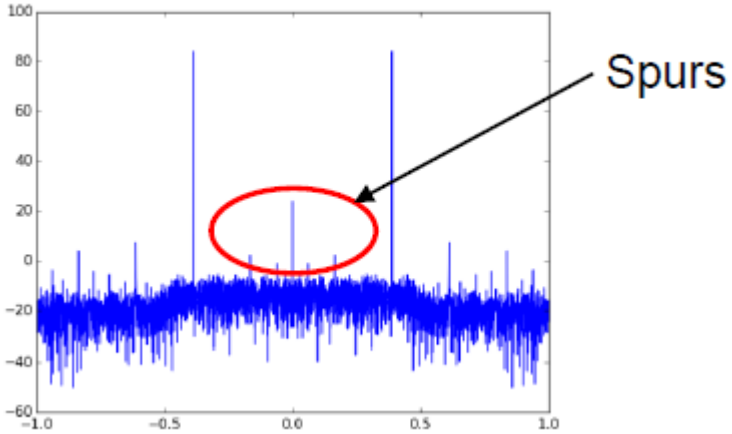


Data Path Bugs Caught - Examples

- Wrong DUT Masking Logic resulted in DC spur
  - Very few seeds failed

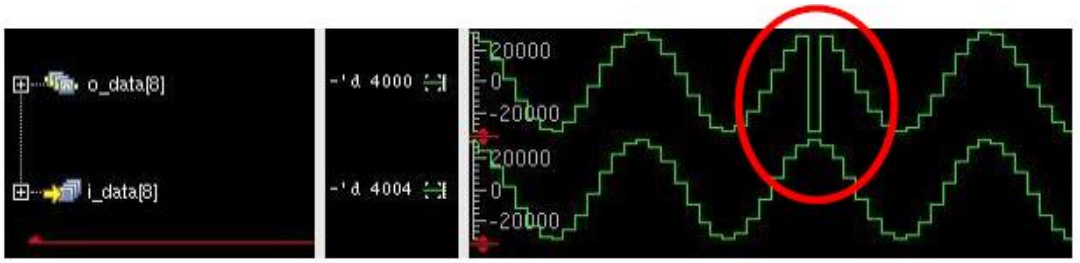


- Truncation instead of rounding inside DUT

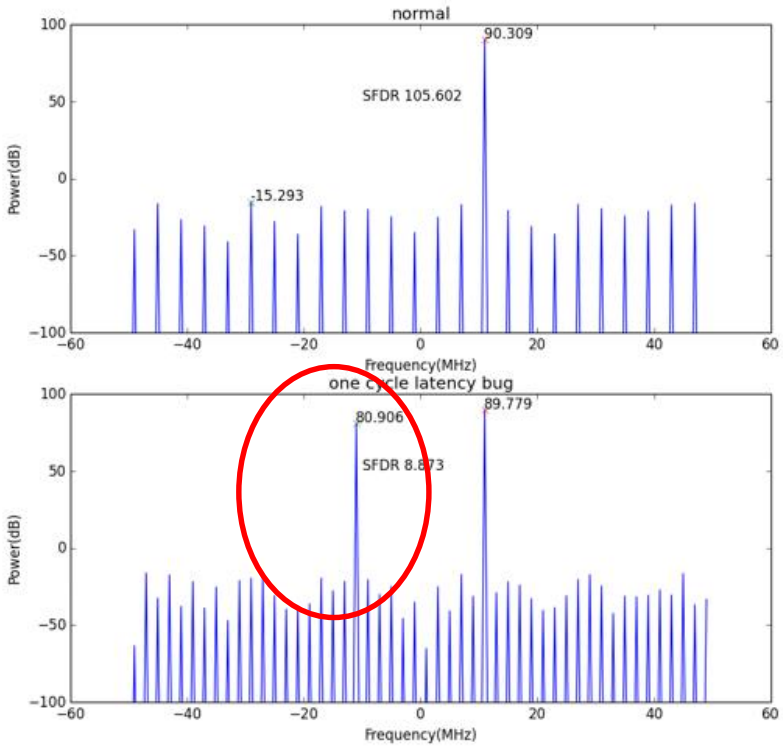


Data Path Bugs Caught - Examples

- Wrong saturation logic



- I/Q path latency mismatch



Data Path Bugs Caught - Examples

- Interface issues (clocking , retiming bugs) will result in severe SFDR failure
  - Data getting missed
  - Repeating data
  - Alternate zero data

Conclusion

- Novel constrained random approach has been deployed on datapath verification
  - Helped improve quality and reduced verification time significantly
  - Approach scales better for higher complexity
- UVM + Python Framework was presented
  - UVM features enables reuse from block to system
  - Python libraries bring signal processing capability to UVM

Acknowledgements

- ADI High Speed Convertor Design and Verification Team
  - Bangalore, Beijing , San Diego and Wilmington
- ADI - CAD and IS Teams