

Using StatEye for IEEE Backplane Evaluation

Ali Ghiasi - Broadcom
aghiasi@broadcom.com

Steve Anderson - Xilinx
steve.anderson@xilinx.com



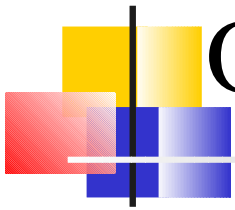
What is StatEye

- ♦ Non partisan open source Matlab tool to evaluate electrical or optical channel.
- ♦ Developed initially by Anthony Sanders and Edoardo Prete of Infineon.
- ♦ <http://www.StatEye.org/> was launched in April 2004 by all developers.
 - ♦ Current release version of StatEye is 2.1B
 - ♦ StatEye 3.0 Beta is now available with GUI and better file arrangement
- ♦ StatEye was developed during OIF CEI development and now additional 6 standards body are considering to use StatEye.

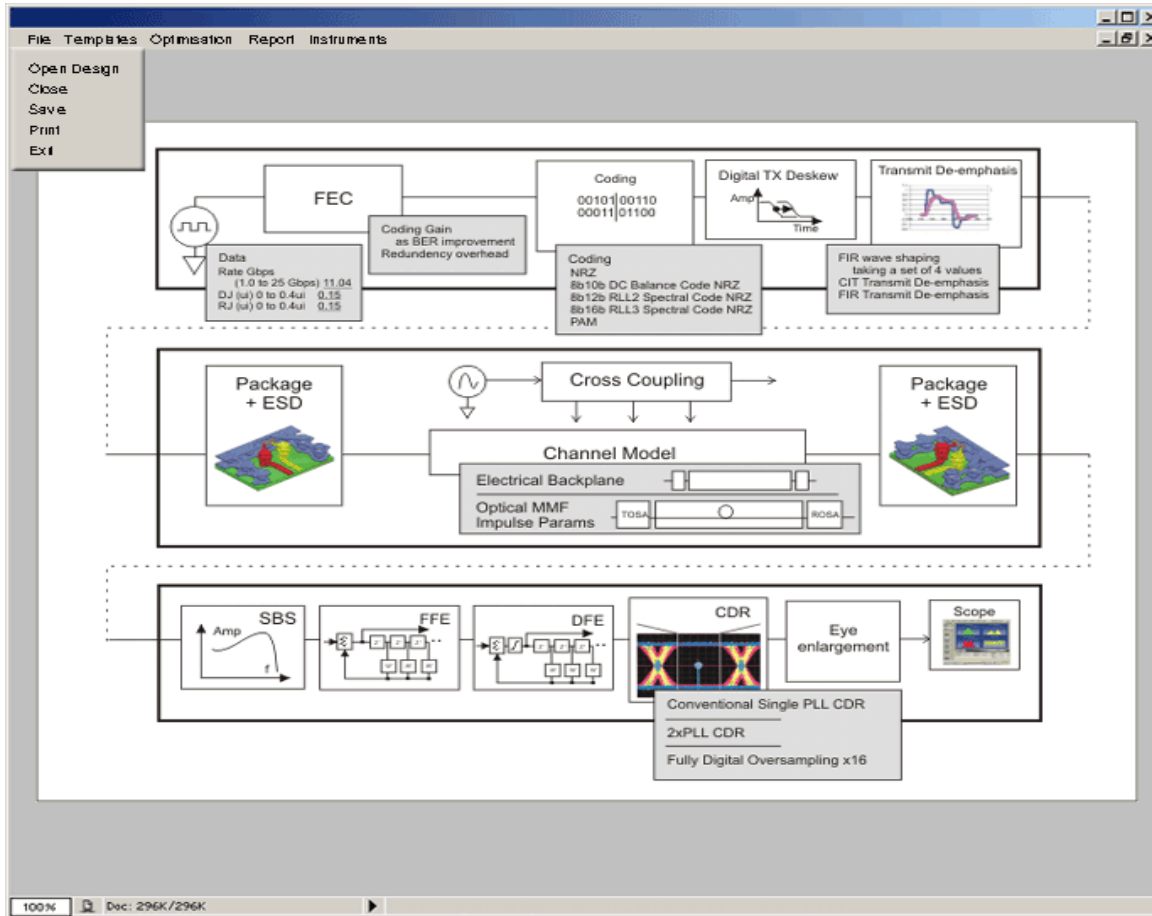


StatEye Supported Features

- ♦ TX FFE
- ♦ RX FFE and DFE
- ♦ NRZ and PAM4
 - ♦ Current version does not support Duobinary.
- ♦ Imports two, four port S-parameters, or ABC parameters can be imported.
- ♦ Support crosstalk aggressors
- ♦ StatEye has recently added MMF fiber based on Cambridge models for the LRM group.



Overview of StatEye 3.0 GUI



Driven from Matlab 6.5 script with parameters or new GUI (shown)

Generates a 1 UI pulse

Adds RJ and DJ

Shapes pulse for slew rate of silicon

User can apply any combo of techniques shown on diagram

Filters such as DFE can be optimised automatically

Includes non-linear components such as DFE

Computes ISI

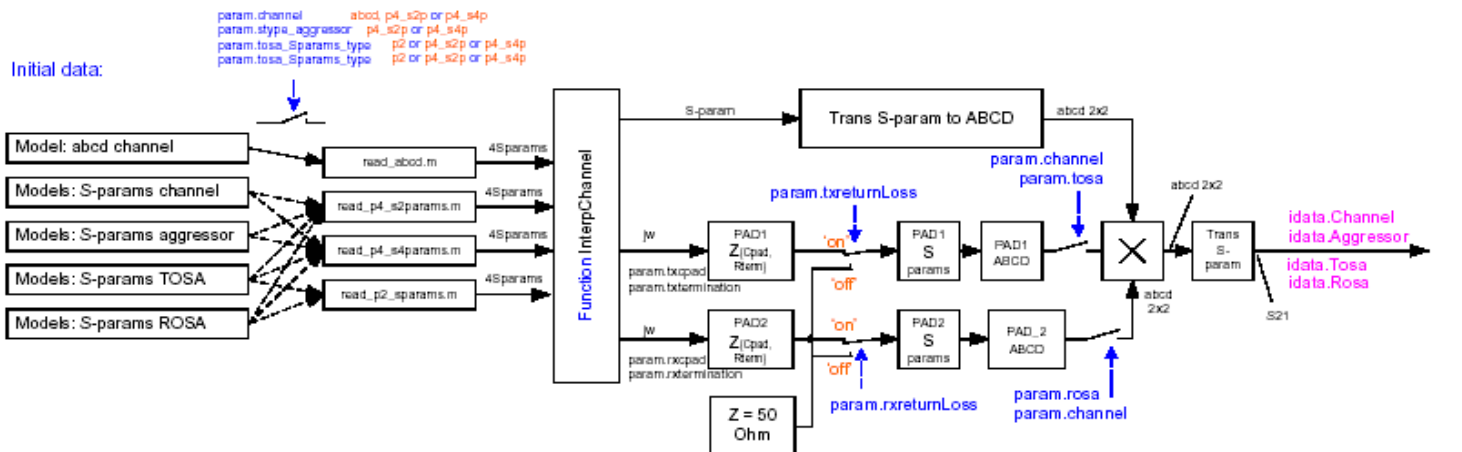
Shows output pulse, spectrum, bathtub curve and eye diagram for different BERs.

Source StatEye



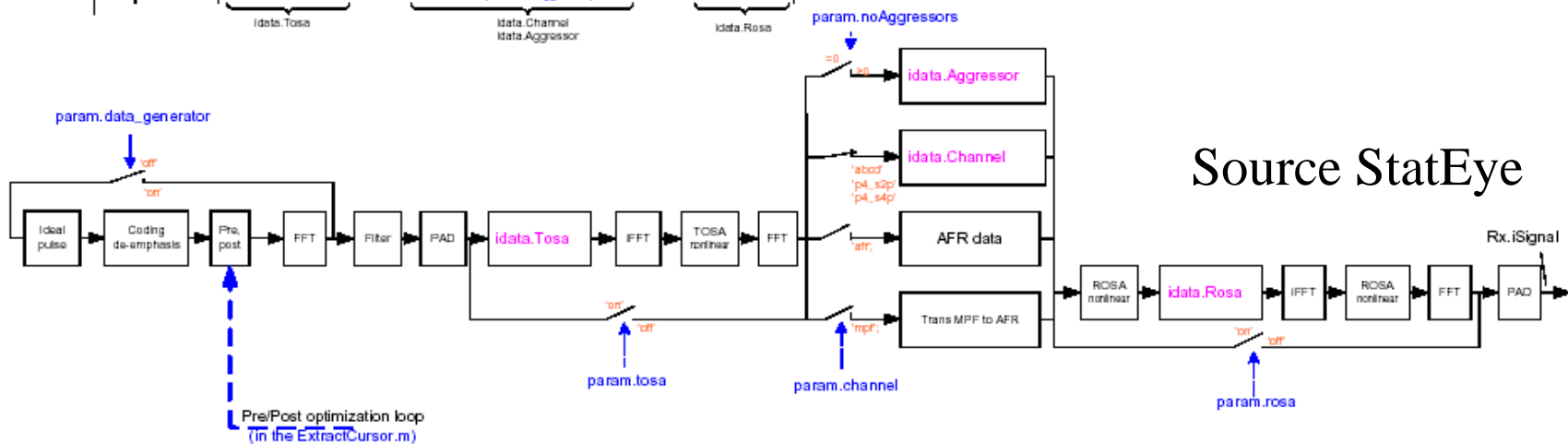
StatEye 3.0 Block Diagram

InterChannels.m v3.0b

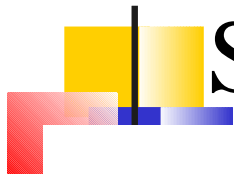


tosa	rosa	noAggressors	Calculated before the optimisation			Relative time
off	off	>0	(Filter x PAD x Channel/Aggressor x PAD)	N-ROSA	(S-ROSA x PAD)	3
off	off	>0	(Filter x PAD x S-TOSA)	N-TOSA	(Channel/Aggressor x PAD)	3
off	off	>0	(Filter x PAD x S-TOSA)	N-TOSA	(Channel/Aggressor x PAD)	3
on	on	>0	(Filter x PAD x S-TOSA)	N-TOSA	(Channel/Aggressor x PAD)	5

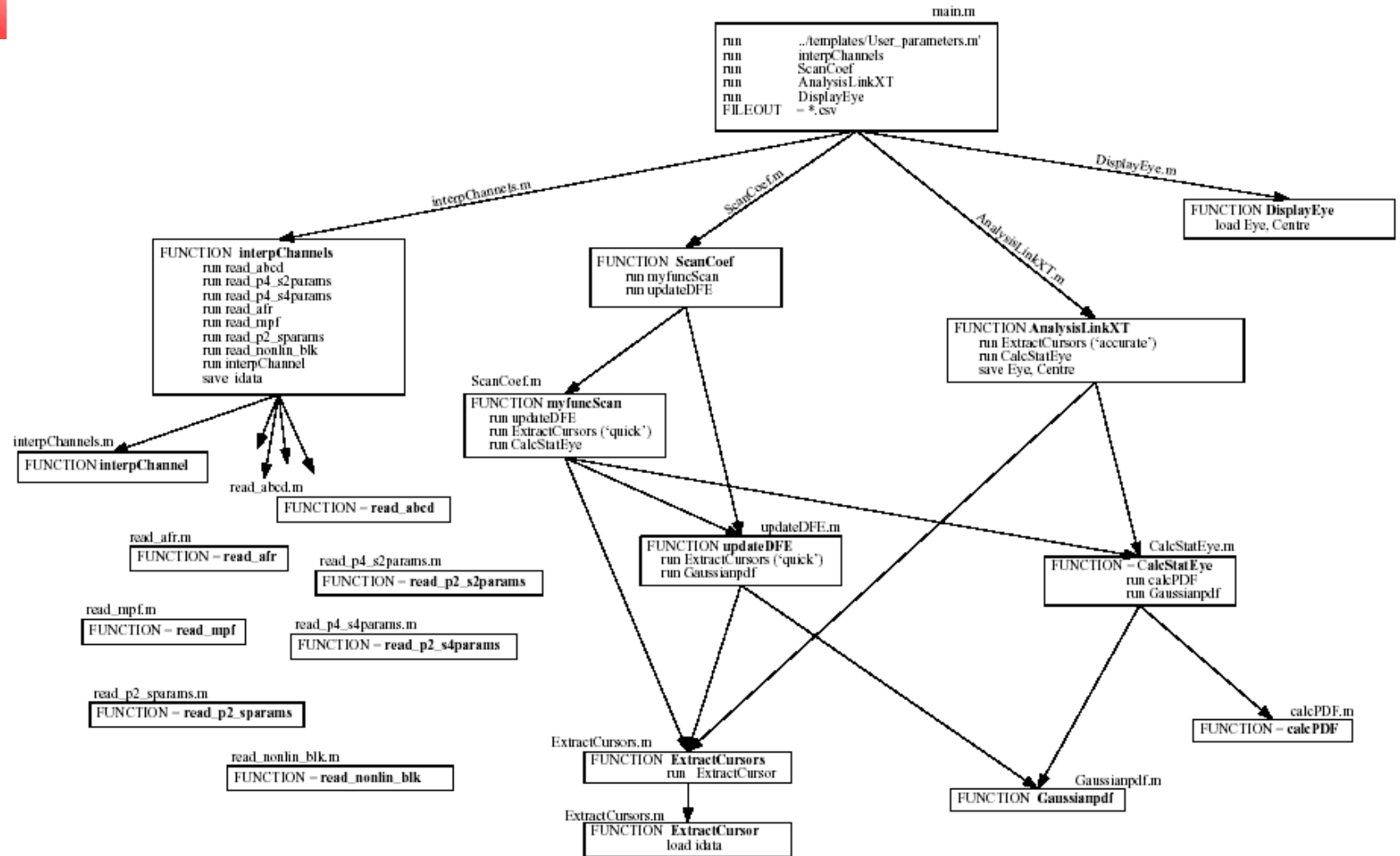
Labels: idata.Tosa, idata.Channel, idata.Aggressor, idata.Rosa



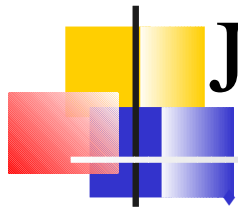
Pre/Post optimization loop (in the ExtractCursor.m)



StatEye 3.0 File Structure

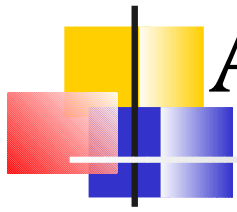


Source StatEye



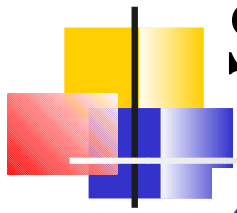
Justifications for Using StatEye

- StatEye build on top of Matlab provide vast set of communication tools for analysis and development.
- ♦ StatEye provides result in minutes instead of hour and day typically would be required by Hspice.
- ♦ According to StatEye “A 48 processor processor farm running for two months to verify StatEye against HSPICE, including thousands of simulations in 10Gbps+”.
- ♦ Fast, open source, and already debugged.



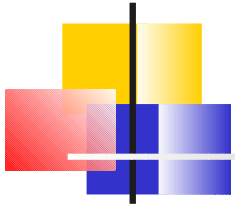
Area of Potential Improvement

- ♦ Current release of the StatEye account for transmitter and receiver return loss by adding a shunt cap.
 - ♦ For more accurate result one has to externally combine the TX, channel, and RX s-parameters.
 - ♦ StatEye might be open to make this improvement if we request.



Stat Eye – More Detail

- ♦ Statistical Description of a Received Signal
- ♦ A way of specifying transmitter & channel
- ♦ Allows calculation of BER or limits on BER
 - ♦ Similar to additive random noise \nearrow BER (common in communication theory)
- ♦ Older methods too limited
 - ♦ Eye template method doesn't work with closed eye
 - ♦ Measurement/analysis time may be too long for newer error rate specs such as BER<1e-15
 - ♦ MJS (Fiber Channel) does not include ISI and crosstalk
- ♦ Future: May become part of
 - ♦ Vector Network Analyzers
 - ♦ ADS (Software Package)



Builds on MJS

(Method of Jitter Specification)

improvement

- Old (Fiber Chan MJS)
Included

- RJ of SERDES
- DJ of SERDES

- Horizontal Eye Closure
Only

- Useful back when signals were not attenuated significantly

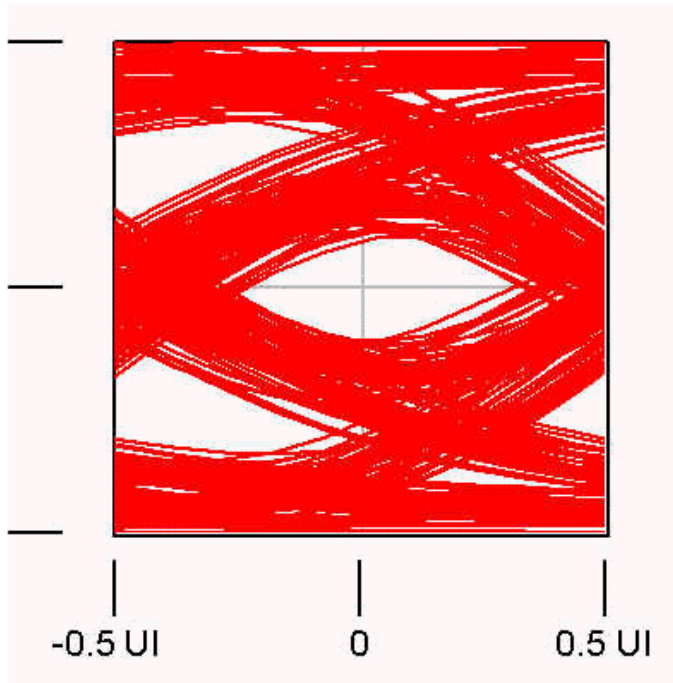
- New (Stat Eye)
Includes

- ISI
- Crosstalk
- RJ of SERDES
- DJ of SERDES

- Horizontal *and* Vertical
Eye Closure

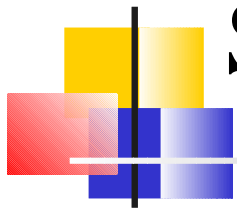
- Needed for BR > 3 Gbps

Stat Eye Versus Scope Eye

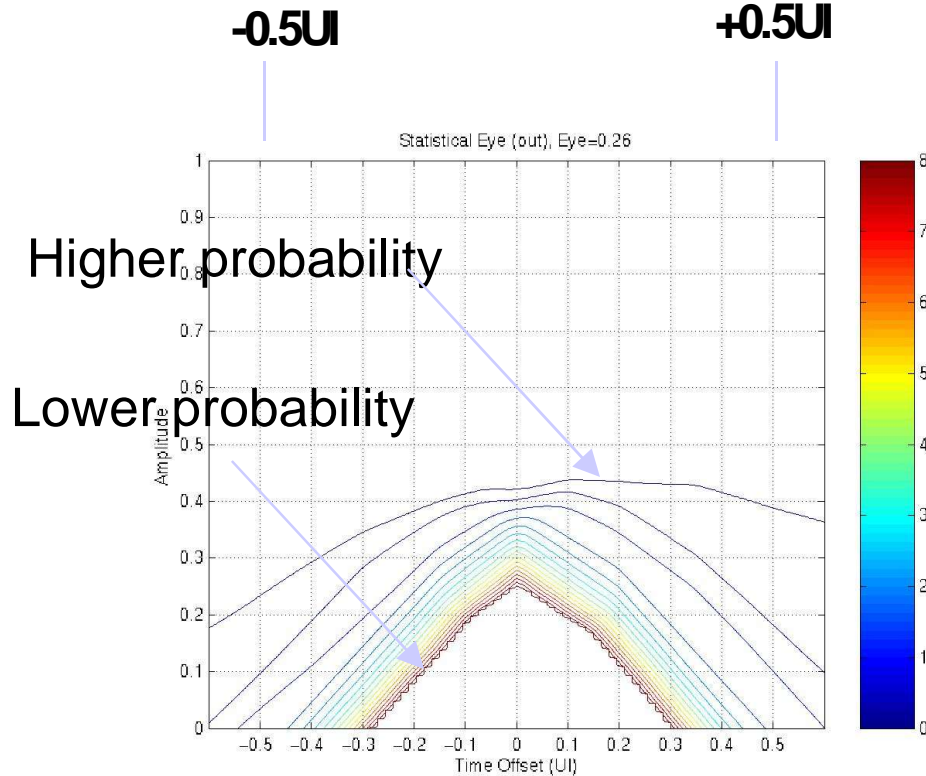


Scope Eye

- Scope Eye:
 - Rough measure of quality of Received Signal, widely used
 - Made by overlaying Oscilloscope traces
 - Eye opening becomes smaller with longer Measurement time
 - Implies random nature of received signal
 - Input = received voltage



Stat Eye Versus Scope Eye



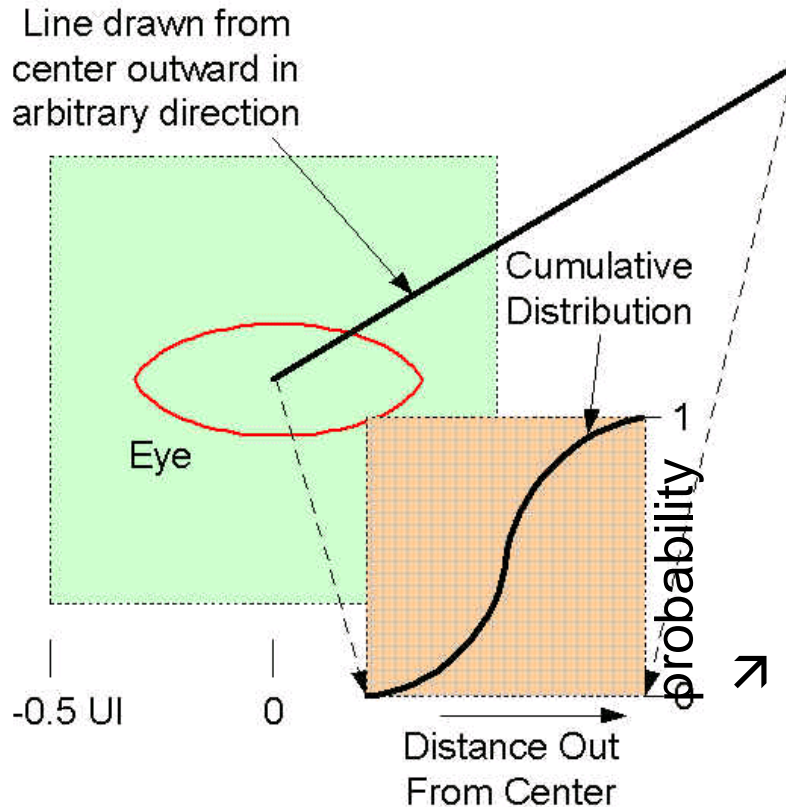
Stat Eye

- ◆ **Stat Eye**

- ◆ Set of probability contours
- ◆ Calculated, not measured
 - ◆ Hor axis is with respect to sampling point
- ◆ Only top half is shown because of symmetry
- ◆ Inputs:
 - ◆ SERDES Jitter Statistics
 - ◆ Channel S-parameters
 - ◆ Crosstalk S-parameters

Exercise – Assume an Open Eye,

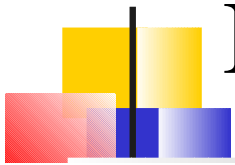
Draw Radial Line Out From Eye Center, Imagine How the Probability of Including the Signal Increases Along This Line



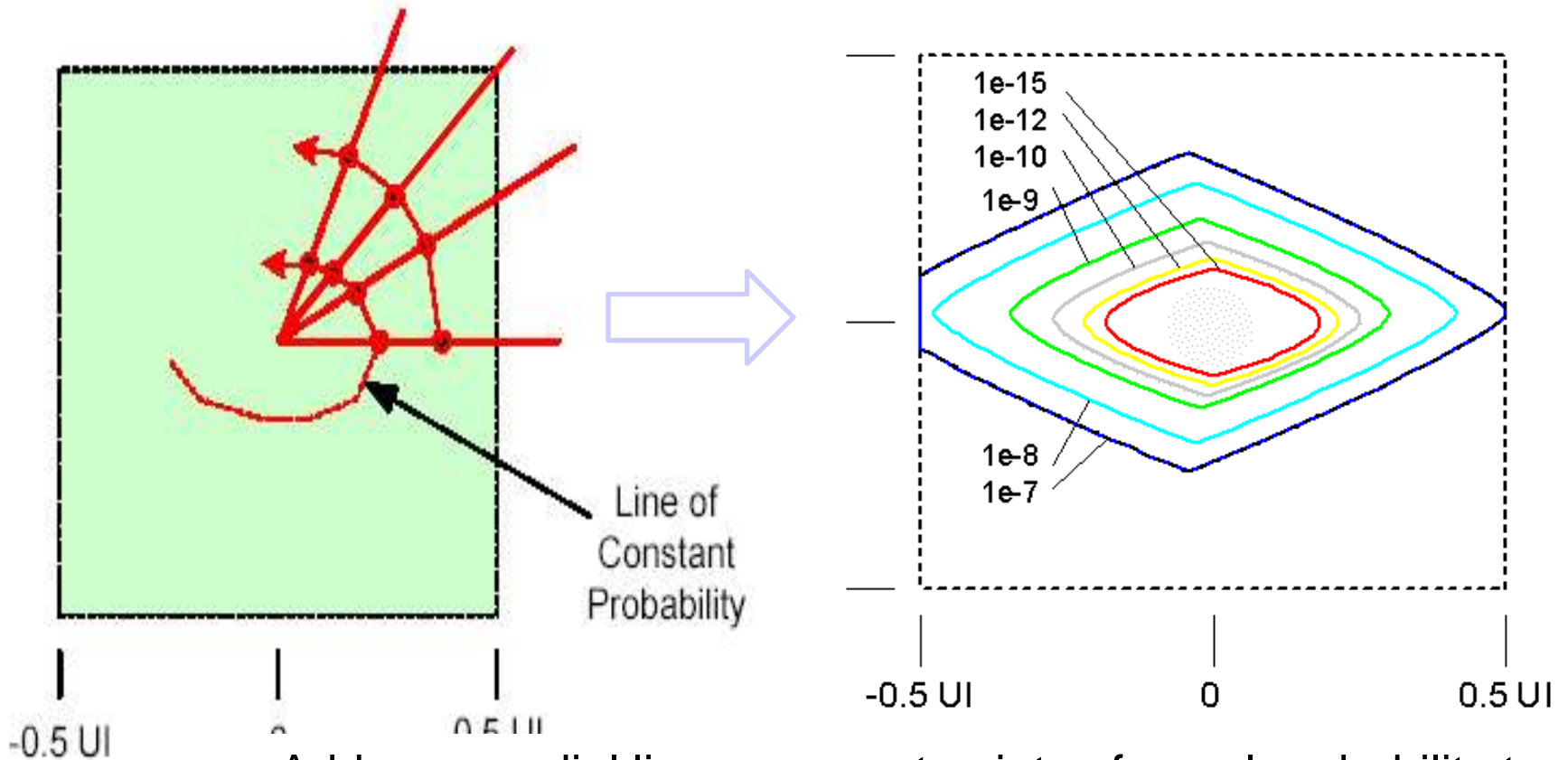
Signal almost never found at eye center (sample point)

Moving out from center, signal is more and more likely to be found

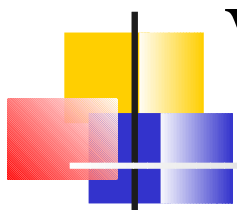
Cumulative distribution function shows probability of finding signal within a given distance out from center



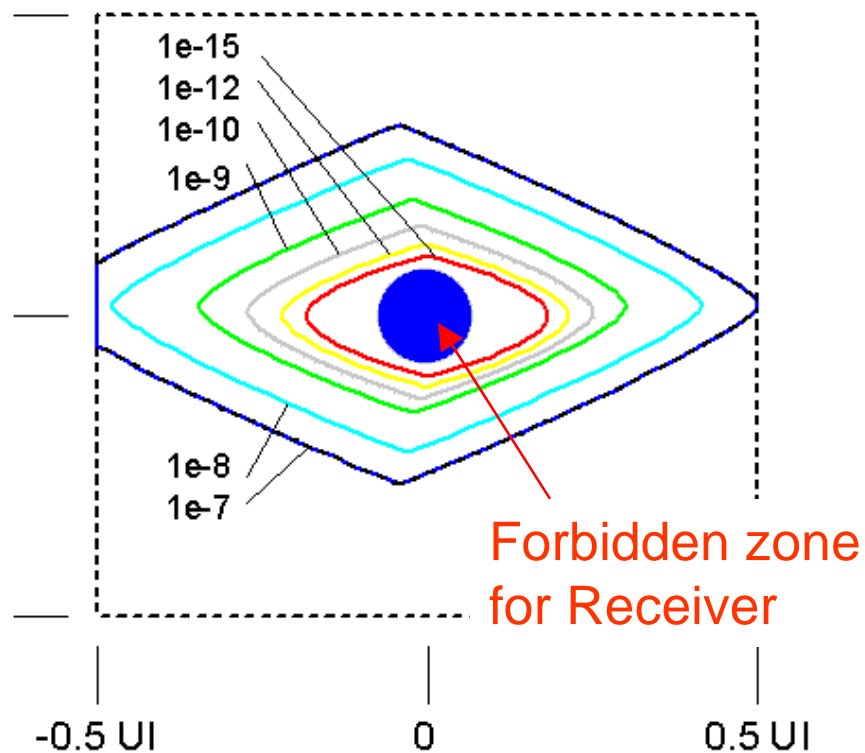
Building a Stat Eye



Add more radial lines, connect points of equal probability to form a contour. Each contour represents probability that signal will be found within the contour



Why is Stat Eye Useful ?



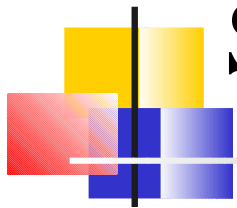
If we know that Rx is
error-free for signals
outside of blue circle

AND

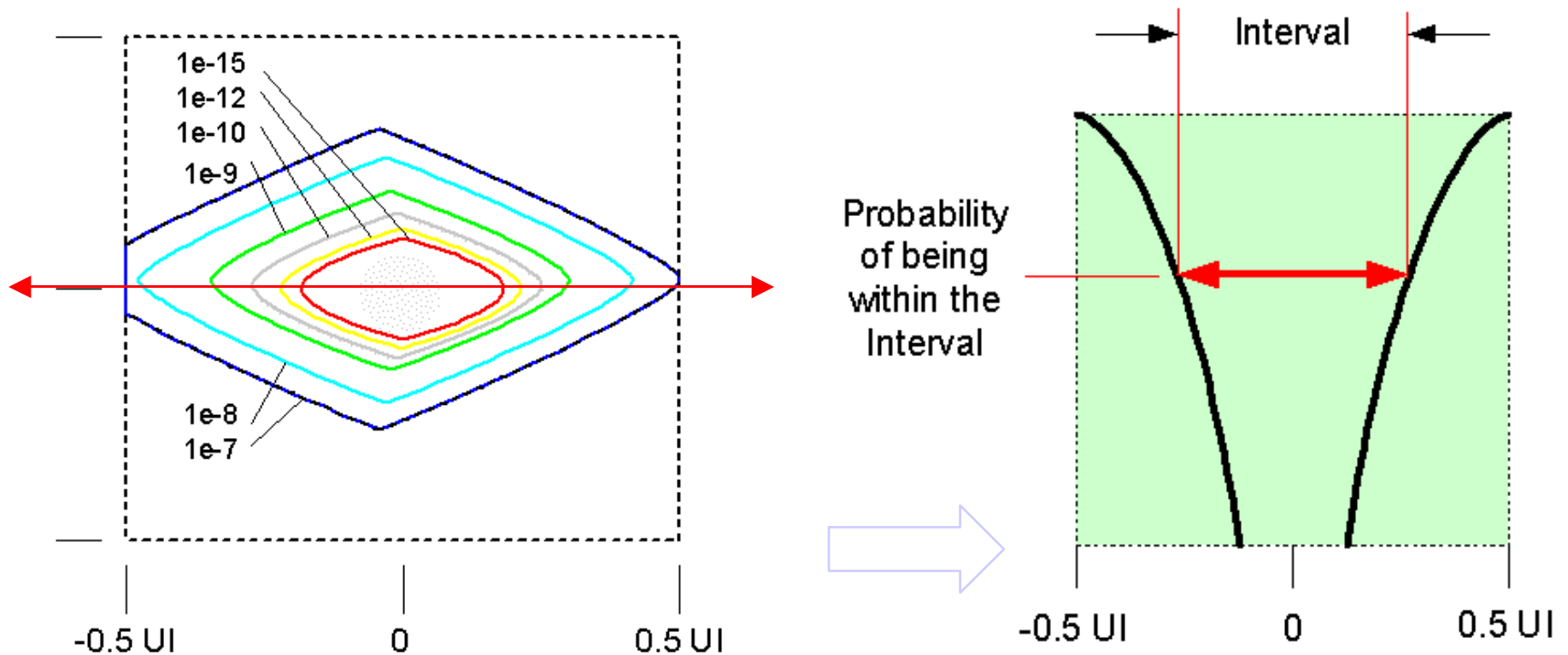
Blue circle is entirely
contained within the
1e-15 contour

THEN

BER < 1e-15

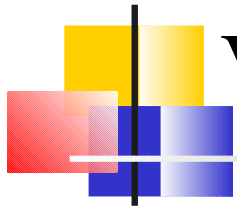


Stat Eye to Bert Scan (Bathtub)



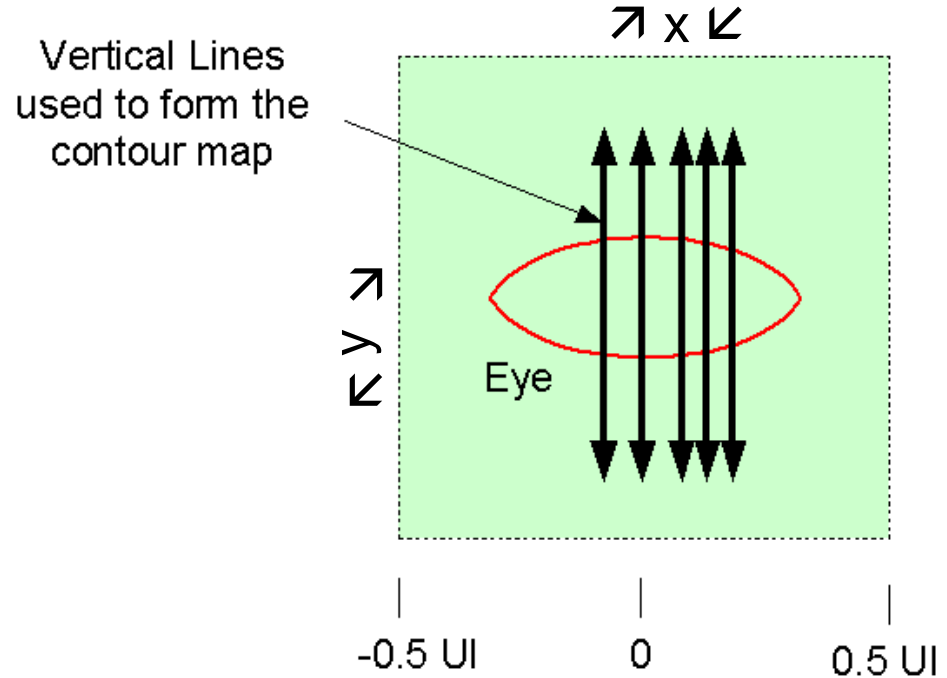
Slice Through Contours along Horizontal Axis of Stat Eye = Bert Scan
Note: Not the same as Bert Scan Calculated from RJ, DJ

Stat Eye in More Detail



Vertical Instead of Radial

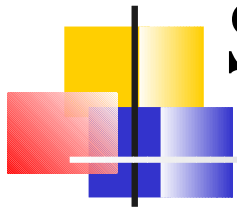
- The actual stat eye algorithm finds the CDF along vertical lines instead of radial
- WHY? Because this allows separation of the overall pdf* into less dependent vertical and horizontal pdfs:



$$p(x, y) = p_y(y | x) p_x(x)$$

Overall pdf Vertical pdf Horizontal pdf

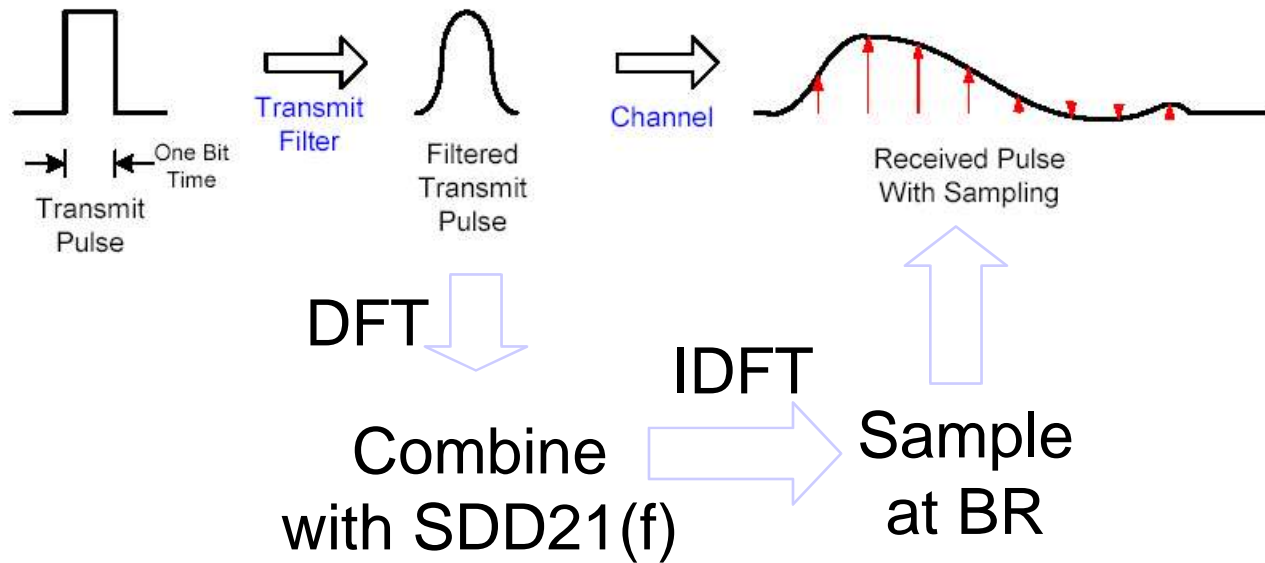
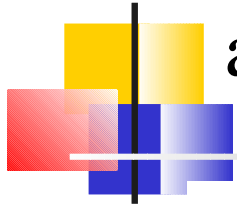
pdf* = probability density function



Steps to Generate Stat Eye

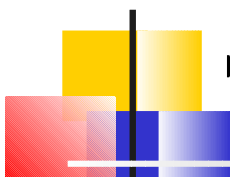
1. Ignore Crosstalk for the moment
2. Measure S-parameters of channel
3. Generate a pulse response from S-params
4. Find $p_y(y|x_0)$ the conditional vertical pdf for first value of $x=x_0$.
5. Repeat 4 for family of x values to get $p_y(y|x)$
6. Generate $p_x(x)$ the horizontal pdf from RJ, DJ values
7. Choose a vertical line at $x = x_s$. Combine results of 5 and 6 to get $p_s(y) = \text{pdf along vertical line}$
8. Repeat 7 for different x .

Steps 3,4: Generate Channel Pulse Response and sample at x_0 , x_0+T , etc.



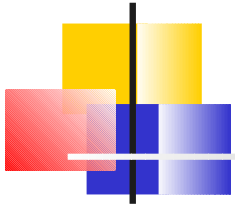
Pulse samples (cursors) are labeled
 $\dots C_{-2}, C_{-1}, C_0, C_1, C_2, \dots$

x_0 = position of any cursor within one UI interval is arbitrary
 T = One bit time

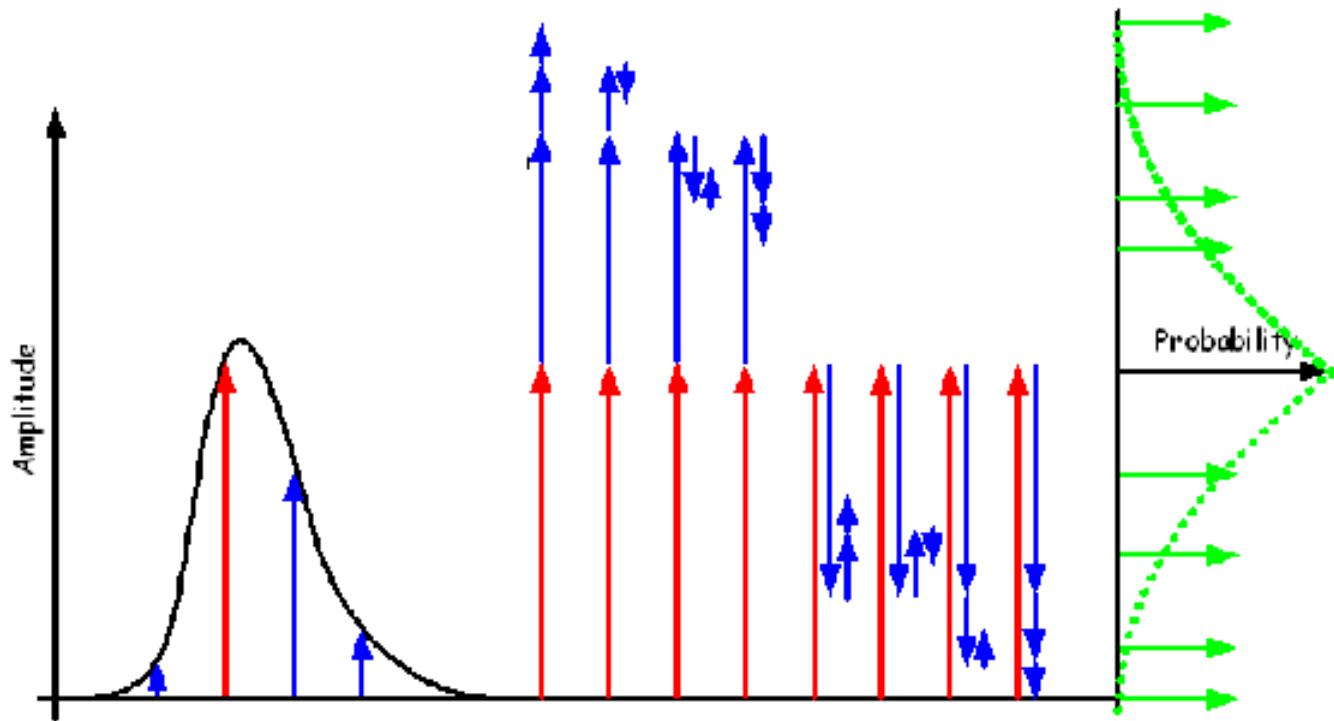


Step 4 Continued

- ♦ Find values of all combinations of cursors and their negatives. Sort into bins.
- ♦ WHY? A given sample of an arbitrary received signal is the superposition of all possible cursors.
- ♦ EXAMPLE: Suppose the only non-zero cursors are C_0 and C_1 . Contributing bit patterns can only be 00, 01, 10, or 11. These result only in sample values of $-C_0-C_1$, $-C_0+C_1$, C_0-C_1 , and C_0+C_1 (signal value for binary 0 is -1 , signal value for binary 1 is $+1$)
- ♦ For N non-zero cursors there will be 2^N combinations
- ♦ Mean always zero

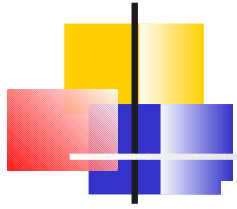


Step 4 Illustrated: Formation of positive half of a $p_y(y|x_0)$



C_0 Red /// C_{-1} , C_1 , and C_2
Blue

Source: Anthony Sanders



Step 5: Move over tiny amount (~ 0.001 UI) to $x_0 + \checkmark x$ And Repeat

Find $p_y(y|x_0 + \checkmark x)$

Find $p_y(y|x_0 + 2\checkmark x)$

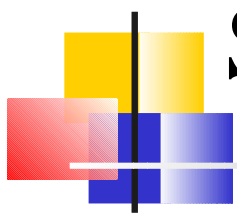
Find $p_y(y|x_0 + 3\checkmark x)$

...

Find $p_y(y|x_0 + 1000\checkmark x)$

Now we have a representation of $p_y(y|x)$

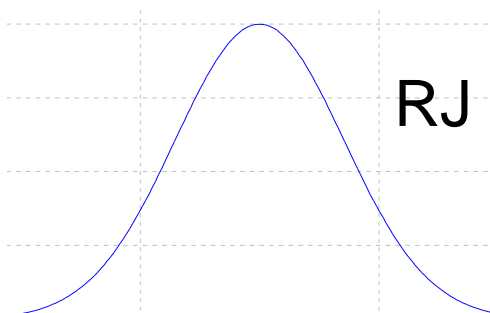
In processing the stat eye, $p_y(y|x)$ is a set of approximately 1000 to 2000 histograms, depending on how fine we want to make the steps in the contours



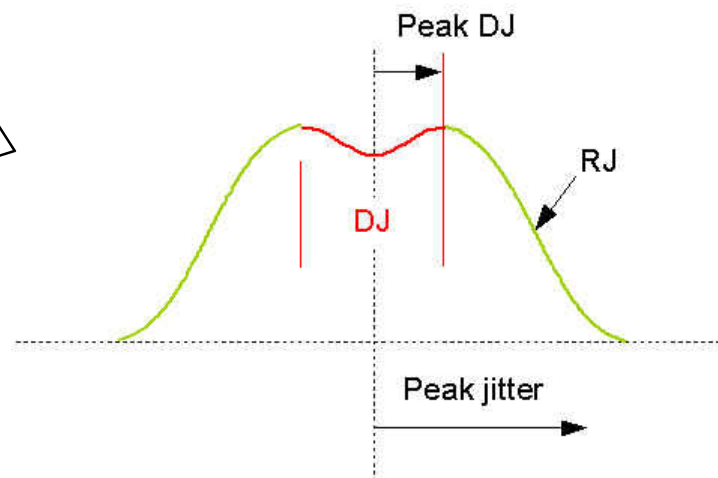
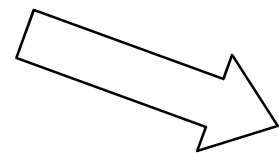
Step 6: Horizontal PDF (MJS)

DJ

$$p_{DJ}(x) = \frac{\delta(x-x_0)}{2} + \frac{\delta(x+x_0)}{2}$$



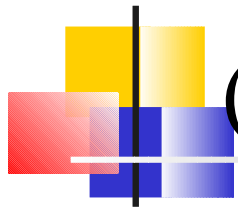
$$p_{RJ}(x) = \frac{1}{\sqrt{2\sigma^2}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$



$$p_x(x) = \frac{1}{2\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-x_0)^2}{2\sigma^2}\right) + \frac{1}{2\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x+x_0)^2}{2\sigma^2}\right)$$

horizontal pdf used in stat eye
 typical values are $\blacktriangle = 0.01UI$,
 $x_0 = 0.15UI$

Step 7: Generate a $p_s(y)$



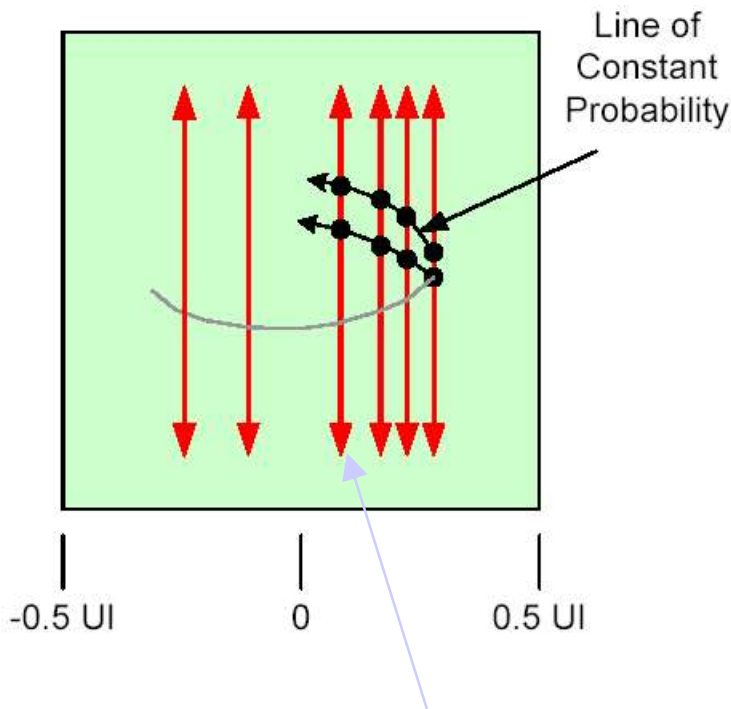
(vertical pdf line)

- ♦ Choose an $x=x_s$. Over a small enough region along x , the overall pdf does not vary with x . Then

$$p_s(y) = \Delta x \cdot p(x_s, y) = \Delta x \cdot p_y(y|x_s) p_x(x_s)$$

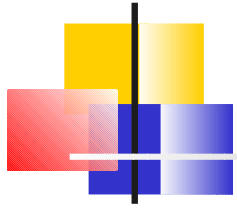
- ♦ Notice that $p_s(y)$ is not the same as the $p_y(y|x_s)$. The old function $p_y(y|x_s)$ is strictly an amplitude density at a point x_s . The new function $p_s(y)$ is an amplitude density with jitter included.

Done !



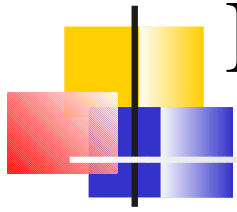
Representative function $p_s(y)$
defined along this line

- ♦ Find several functions $p_s(y)$ along the x axis
- ♦ Integrate each one from 0 out to y_0 to get probability $P_s(y < y_0)$
- ♦ Connect points of equal probability



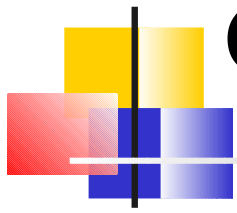
Choice of Algorithm in Building $p_y(y|x_0)$ Is Important

- ♦ Possibilities:
 - ♦ Generate every combination and assign to bin
 - ♦ NO GOOD, takes forever
 - ♦ But can be used to check results
 - ♦ Use convolution
 - ♦ Relatively fast
 - ♦ This part of Analysis usually completed in minutes



Including Crosstalk

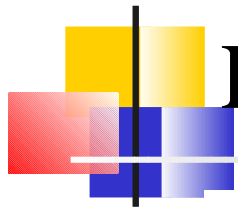
- ♦ Measure S-parameters of Crosstalk Channel
- ♦ Find pulse response for Crosstalk Channel
- ♦ Add Crosstalk Cursors and their negatives to previously found Channel Cursors in every combination
- ♦ Repeat for Each Crosstalk Channel



Closed Eye

- ◆ Stat Eye algorithm allows almost any combination of Linear Equalization (LE) and/or Decision Feedback Equalization (DFE)
- ◆ Continuous equalization is being added
- ◆ Precursor removal can be included
- ◆ Algorithm performs an optimization when finding filter coefficients
 - ◆ sampling position within eye is allowed to vary by the amount of the specified jitter
 - ◆ The filter coefficients are found that minimize the maximum error of equalization

Closed Eye Continued

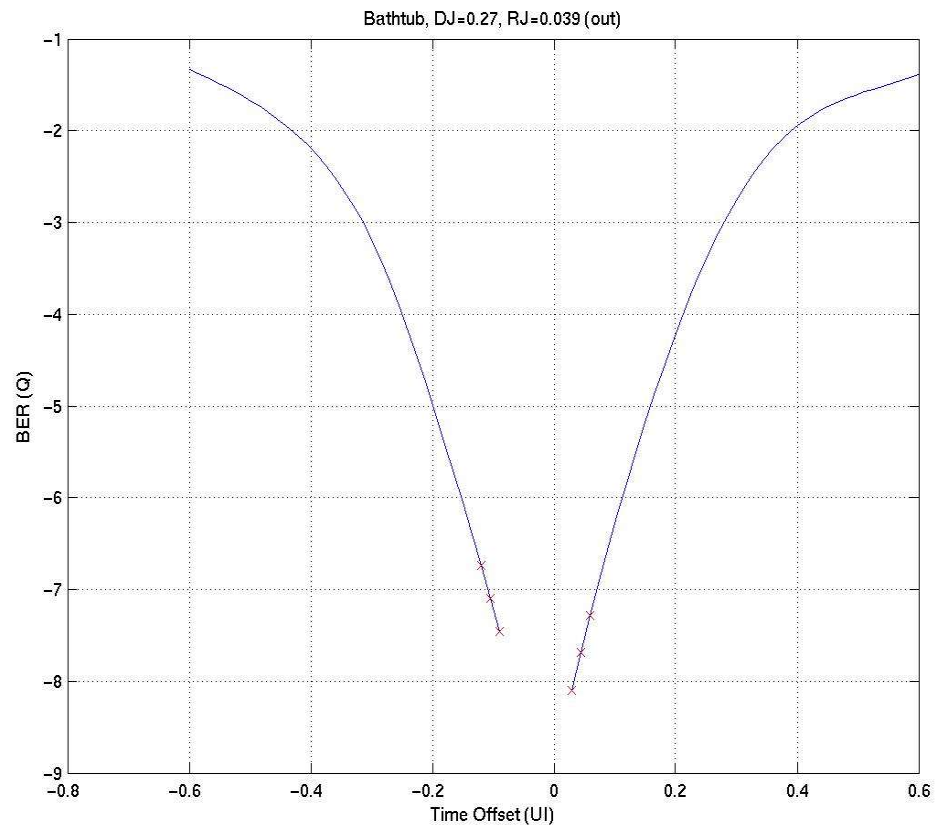
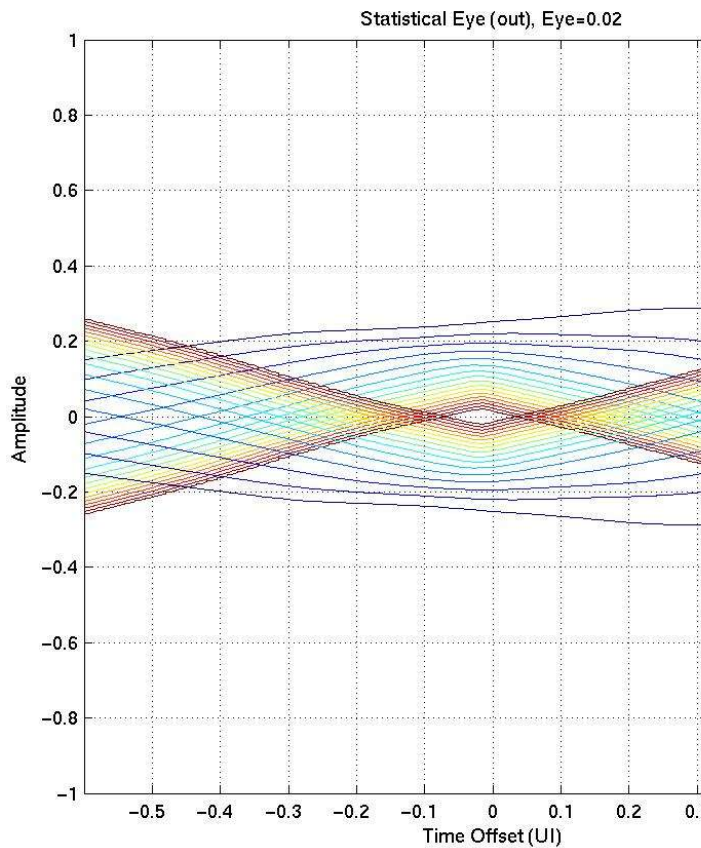


Example C_1 (1st Postcursor)

- ♦ Ideally, equalization cancels this completely
- ♦ Jitter causes a family of cancellation values
- ♦ Choose a tap weight that will minimize the maximum of this family of values
- ♦ Extend this process to include all of the Pre- and Postcursors for which equalization is to occur
- ♦ This part of the algorithm takes the longest (as long as one or two hours)



Example Stat Eye Plots (No Crosstalk)



Example Stat Eye Plots (No Crosstalk)

Previously Presented (anderson_02_0704)

