

Commit Request 4p17_1: Fast Top-K Gaussian Mixture ADC Clip Solver

Hansel D'Silva and Samuel Kocsis
Amphenol Corporation

IEEE 802.3 Channel Operating Margin (COM) Open Source Project Ad Hoc

Revision History

Revision	Date	Comment
0.5	23-Sep-2026	▪ Initial Draft.
1.0	29-Sep-2026	▪ Added a side titled 'H. Results'. ▪ Added a slide titled 'For Consideration'.
1.5	29-Sep-2026	▪ Replaced keyword “tap” with “cursor”.

List of Supporters

- Adee Ran, CISCO.
- Adam Gregory, Samtec.
- Hossein Shakiba, Huawei.

Summary

- Fast Top-K Gaussian Mixture modeling preserves the clipping-threshold physics of the Slow PDF/CDF method while significantly reducing computational complexity during COM equalization search.
 - Add Clip Method “Fast_1” in addition to “Slow” and “Fast”.
- The proposed Clip Method compared to “Slow” gives nearly a 3x reduction in run time with negligible difference in COM (less than ± 0.1 dB).

For Consideration

- For the IEEE 802.3dj- 200G draft, we still need to continue to use the “Slow” method to adhere to the standard method and language.
- Support adding the new “Fast_1” method for a possible consideration in the next generation IEEE 802.3dv- 400G draft.

Problem Statement

- Reduce the computational complexity of ADC clipping threshold estimation while preserving dependence on P_{qc} and maintaining accuracy relative to the PDF/CDF-based method.
 - In IEEE 802.3 COM, the ADC clipping threshold is obtained by solving: $P(X > \text{adc_clip}) = P_{qc}$ using signal PDF generation, PDF convolution, CDF construction, and inverse-CDF evaluation.
 - Since ADC clipping is evaluated repeatedly during the FOM equalization search (7203 EQ combination with TXLE turned OFF and 9.227043 Million combinations with TXLE turned ON), the PDF/CDF-based approach can become a ***significant runtime contributor***.
- The Clip Method of “Fast” uses peak signal excursion, it is conservative for it assumes worst-case constructive addition of all pulse-response cursors and ignores the noise model and does not account for target P_{qc} .
 - It uses $\text{adc_clip} = \text{sum}(\text{abs}(\text{sampled_pulse_response}))$.

1. Introduction

A. ADC Clipping in IEEE 802.3 COM

adc_clip is the ADC input threshold, and Pqc is the target probability that the ADC input exceeds this threshold:

$$P(X > \text{adc_clip}) = P_{qc}$$

Where, the ADC input is: $X = \sum h_k A_k + N$

h_k : sampled pulse response cursors

A_k : PAM symbols

N : Gaussian noise

0] Clip Method= "Slow"

```
get_PSDs(),
237 if strcmpi(param.clip_method, 'Slow')
238     sig_aftert_ctle_pdf = get_pdf_from_sampled_signal(sampled_pulse_response,param.levels,OP.BinSize);
239     noise_after_ctle_pdf = sig_aftert_ctle_pdf;
240     sigma_noise = sqrt(result.S_in_rms.^2+result.S_rn_rms.^2+result.S_xn_rms.^2+result.S_tn_rms.^2+result.S_rj_rms.^2);
241     noise_after_ctle_pdf.y = 1/(sqrt(2*pi)*sigma_noise)*exp(-noise_after_ctle_pdf.x.^2/(2*sigma_noise.^2))*OP.BinSize;
242     sig_noise_after_ctle_pdf= conv_fct(sig_aftert_ctle_pdf,noise_after_ctle_pdf);
243     sig_noise_after_ctle_cdf = cumsum(sig_noise_after_ctle_pdf.y);
244     ctile_signal_sigma = sqrt(sum((sig_noise_after_ctle_pdf.x.^2).*sig_noise_after_ctle_pdf.y));
245     adc_clip=-CDF_inv_ev(param.P_qc, sig_noise_after_ctle_pdf,sig_noise_after_ctle_cdf);
246     result.ctile_signal_sigma=ctile_signal_sigma;
247 else
248     adc_clip = sum(abs(sampled_pulse_response));
249 end
250 adc_lsb = 2*adc_clip/(2^param.N_qb-1);
251 sigma_Q = adc_lsb/sqrt(12);
252 S_qn = sigma_Q.^2/f_b*ones(1, num_ui);
253 result.adc_clip=adc_clip;
254 result.S_qn = S_qn;
255 result.S_qn_rms = sqrt(sum(result.S_qn)* delta_f);
```

PDF of Signal at the CTLE output

Build Gaussian noise PDF

Signal+ Noise PDF

PDF to CDF

RMS sigma

Inverse CDF to ADC clip
 $P(X > \text{adc_clip}) = P_{qc}$

1] Clip Method= "Fast"

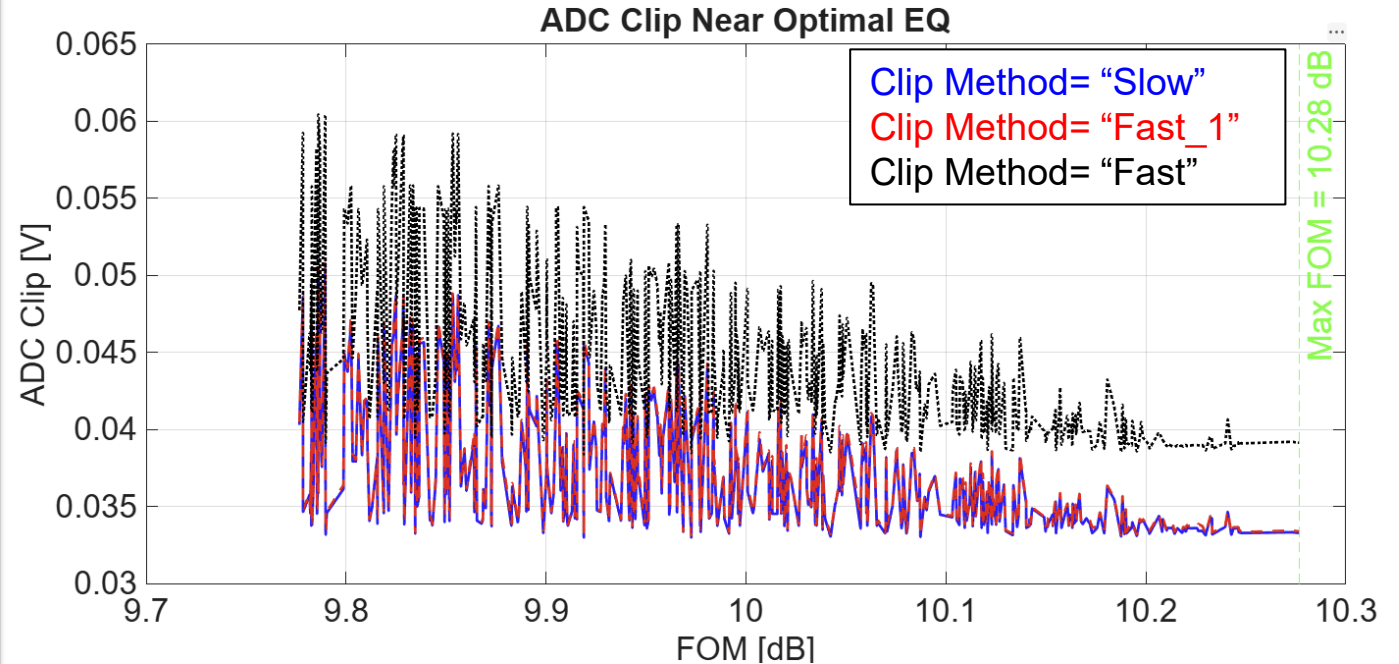
- Uses peak signal excursion, it is conservative.
- Assumes worst-case constructive addition of all pulse-response cursors.
- Ignore noise model, avoids PDF/ CDF and inverse CDF.
- Does not account for target Pqc.

- PDF generation+ Convolution+ CDF inversion are computationally expensive.
- Executed repeatedly during FOM equalization search consisting of 7203 combinations.

B. ADC Clip Method Comparison

Parameter	Clip Method = "Slow"	Clip Method = "Fast"	Clip Method = "Fast_1"
Modulation	Yes	No	Yes
Signal Model	Signal PDF	Peak Swing Estimate	Top-K States
Noise Model	Noise PDF	Ignored	Gaussian Residual
Distribution Construction	PDF Convolution	None	Gaussian Mixture
Clipping Probability	CDF-based	Not Evaluated	Tail Evaluation
ADC clip	Inverse CDF	$\sum h $	Bi-section Search
Run Time (per call)	0.21 sec	0.0004 sec	0.01 sec
Run Time (COM)	14.50 minutes	6.23 minutes	6.10 minutes

Note. EQ space consists of 7203 combinations.



Both "Slow" and "Fast_1" estimate the same clipping threshold adc_clip :
 $P(X > \text{adc_clip}) = P_{qc}$

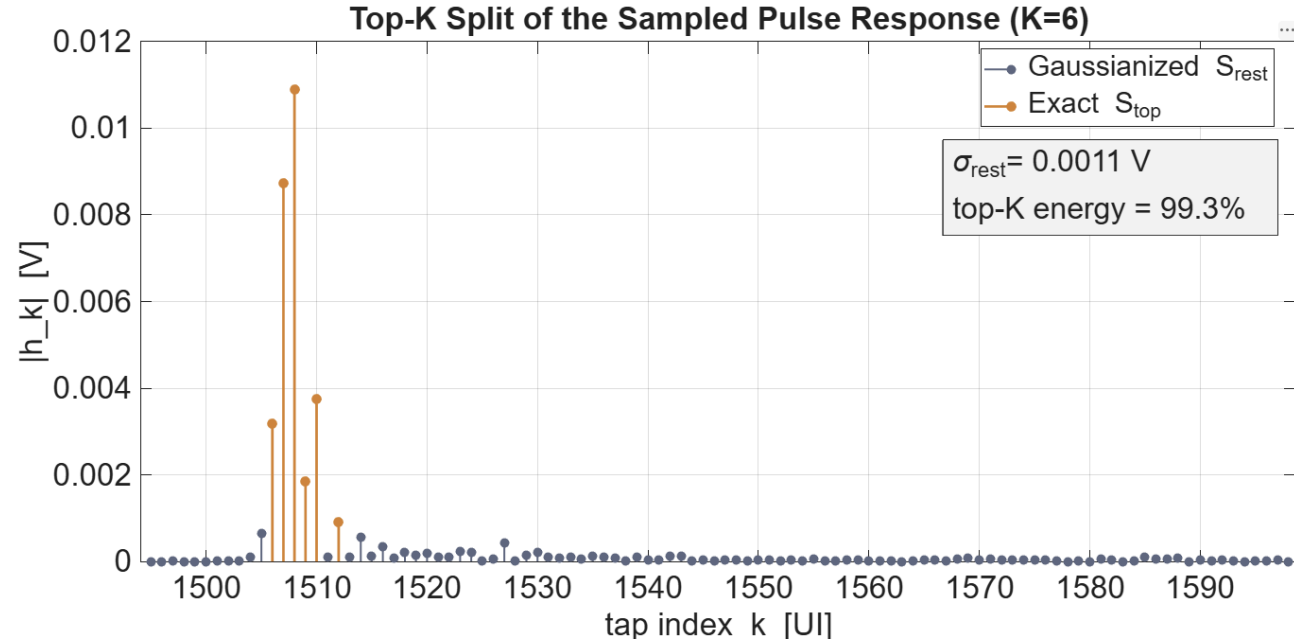
- Fast Top-K = Exact dominant ISI + Statistical residual ISI/noise + Direct tail-probability evaluation.
- Fast Top-K retains the clipping-probability physics of the Slow method while reducing runtime by eliminating PDF generation, convolution, CDF construction, and inverse-CDF evaluation.

C. Split the Pulse Response by Cursor Magnitude

- Central idea: Not “All Cursors” matter equally.
- Sort the cursors by magnitude: $|h_1| \geq |h_2| \geq |h_3| \dots$
 - Few strongest cursors dominate “ISI”.
 - Many smaller cursors mainly contribute to “variance”.
- Signal decomposition,
$$X = S_{\text{top}} + S_{\text{rest}} + N$$

Where,

 - S_{top} : K-strongest cursors.
 - S_{rest} : all remaining cursor.
 - N : Gaussian Noise.



- The goal is to retain dominant ISI exactly while simplifying weak-cursor processing.
- The residual ISI contribution is Gaussianized using its second-order statistics, consistent with Central Limit Theorem behavior for a large number of independent PAM-symbol contributions.

D. Top-K Gaussian Mixture Model

1. Exact component

- $S_{\text{top}} = \sum_{k=1:TopK} h_k A_k \dots A_K \in \{-1, -1/3, +1/3, +1\}$
- Enumerate all symbol combinations.
- In the case of PAM4, if one decides $K=6$ then evaluate $4^6 = 4096$ exact states.

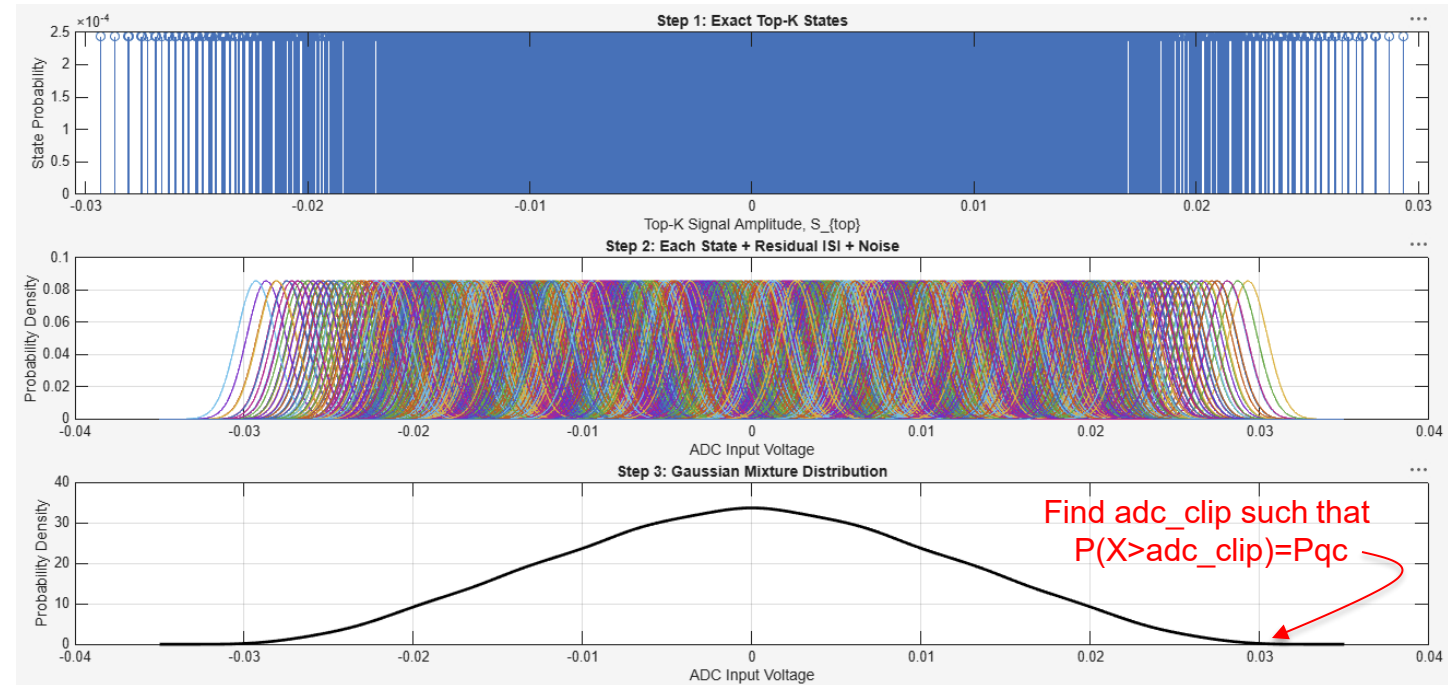
2. Gaussian component

- Remaining cursors along with noise contribute to “variance” only.
- $S_{\text{rest}} + N \sim N(0, \sigma_{S_{\text{rest}}}^2) \dots$ where, $\sigma_{S_{\text{rest}}}^2 = \sigma_X^2 \sum h_{\text{rest}}^2 + \sigma_{\text{noise}}^2$
- Combine with noise: $\sigma_{S_{\text{rest}}} = \sqrt{\sigma_{S_{\text{rest}}}^2 + \sigma_{\text{noise}}^2}$

- Strong cursors are modelled as Exact while weak cursors + Noise are modelled as Gaussian.

E. Gaussian Mixture Model of ADC Input

- Step 1: Enumerate Top-K states
 - Each state corresponds to one exact ISI realization.
 - $S_{\text{top}} = \sum_{k=1:TopK} h_k A_k \dots A_K \in \{-1, -1/3, +1/3, +1\}$.
 - Enumerate all possible outputs of the strongest K cursors.
 - PAM4, K=6 gives $4^6 = 4096$ exact ISI states.
- Step 2. Residual ISI+ Noise
 - Residual cursors and noise introduce uncertainty.
 - $S_{\text{rest}} + N \sim \mathcal{N}(0, \sigma_{S_{\text{rest}}}^2)$.
 - The colored curves represent: $X|_{S_{\text{top}}} = s_i \sim \mathcal{N}(s_i, \sigma_{S_{\text{rest}}}^2)$
- Step 3. ADC input distribution
 - Adding all weighted Gaussian components produces the ADC input distribution:
 $f_X(x) = \sum_i w_i f_i(x) \dots$ Gaussian Mixture Model.
 - Solve for the clipping threshold: $P(X > \text{adc_clip}) = P_{qc}$.



$$P_{qc} = 1e-7 \text{ and } \text{adc_clip} = 0.0322$$

- Fast Top-K replaces PDF convolution with a Gaussian Mixture Model.
- Strong ISI cursors are modeled exactly, while residual ISI and noise are modeled statistically.

F. Direct Tail Probability Evaluation

- Instead of building a PDF and CDF, we compute directly:

$$P(X > t)$$

using Gaussian tails: $Q(z) = 0.5 * \text{erfc}(z / \sqrt{2})$

where, $Q(z) = P(Z > z)$ and $Z \sim N(0, 1)$

- Overall tail probability:

$$P(X > t) = \sum_i w_i Q\left(\frac{t - s_i}{\sigma_{s_{rest}}}\right)$$

- Each Gaussian component contributes a tail probability determined by the Q-function.
- We directly compute $P(X > t)$ and solve for the threshold where it equals the target clipping probability P_{qc} .

G. Code Review

```
function adc_clip = fast_clip_topK(sampled_pulse_response, ...
    L, sigma_X2, sigma_noise, ...
    P_qc, K)
```

```
% Force pulse response into a row vector.
h = sampled_pulse_response(:).';
N = numel(h);
```

```
% Generate normalized PAM-L levels.
m = L/2;
odd_levels = 2*(1:m) - 1;
```

For PAM4,
L= 4; m= L/2= 2;
odd_levels= 2*(1:2)-1= [1 3]

```
a_levels = [-fliplr(odd_levels), odd_levels] / (L - 1);
p_levels = ones(1, L) / L;
```

For PAM4,
a_levels= [-3 -1 +1 +3]/3
p_levels= [1/4 1/4 1/4 1/4]

```
% Select the K strongest cursors.
```

```
[~, idx_sorted] = sort(abs(h), 'descend');
K_eff = min(K, N);
idx_top = idx_sorted(1:K_eff);
idx_rest = idx_sorted(K_eff+1:end);
```

Indices of sorted abs(h) in descending order
99.99%, Keff= K
Select indices of strongest cursors
Select indices of not strongest cursors

```
h_top = h(idx_top);
h_rest = h(idx_rest);
```

Sampled pulse response extraction

```
% Approximate the contribution of all non-strongest cursors as Gaussian.
```

```
sigma_Srest2 = sigma_X2 * sum(h_rest.^2);
% Standard deviation of: S_rest + N
sigma_rest = sqrt(sigma_Srest2 + sigma_noise.^2);
```

$\sigma_{rest} = \sqrt{\sigma_{rest}^2 + \sigma_{noise}^2}$
=> RMS width used for every Gaussian component in the later Gaussian-mixture model.

Variance from remaining ISI cursors
=> For independent symbols:
 $Var(hA) = h^2 \cdot Var(A) \text{ \& } Var(\sum(h_k A_k)) = \sigma_X^2 \cdot \sum(h_k^2)$

```
% Exact PMF of: S_top = sum(hk*Ak)
% State representation:
% sums(i) = possible value of S_top
% wts(i) = probability of sums(i)
% Start with one state: amplitude = 0 & probability = 1
sums = 0;
wts = 1;
```

```
for kTap = 1:numel(h_top)
```

```
hk = h_top(kTap);
```

```
% Every existing state is expanded
% using every PAM symbol level.
sums_new = sums(:) + hk * a_levels(:).';
wts_new = wts(:) * p_levels(:).';
```

```
% Flatten back into row vectors.
sums = sums_new(:).';
wts = wts_new(:).';
```

```
end
```

```
var_S_top = sigma_X2 * sum(h_top.^2);
ctle_signal_sigma = sqrt(var_S_top + sigma_rest.^2);
```

Note
sums(:)= Column Vector
hk * a_levels(:).'= Row Vector

```
[-0.8; -0.2667; -0.2667; 0.8]
+ [-0.2 -0.667 0.0667 0.2]
=
[-1.0000 -0.8667 -0.7333 -0.6000;
-0.4667 -0.3334 -0.2000 -0.0667;
0.0667 0.2000 0.3334 0.4667;
0.6000 0.7333 0.8667 1.0000]
```

- Every row= previous state
- Every column= PAM level

```
% Gaussian-mixture tail probability.
inv_sqrt2 = 1/sqrt(2);
Qfun = @(z) 0.5 * erfc(z ./ sqrt(2));
inv_sigma_rest = 1 / sigma_rest;
scaled_sums = sums * inv_sigma_rest;
```

```
tail_mix = @(t) ..
sum(wts * Qfun(t*inv_sigma_rest - scaled_sums));
```

```
% Solve: P(X > t) = P_qc
Smax = sum(abs(h));
```

```
t_lo = 0;
t_hi = Smax + 10*sigma_rest;
```

Clip level cannot be negative

```
tol = OP.BinSize;
% tol = 1e-6;
```

```
while (t_hi - t_lo) > tol
    t_mid = 0.5*(t_lo + t_hi);
```

Every iteration cuts the search range in half

```
if tail_mix(t_mid) > P_qc
    t_lo = t_mid;
else
    t_hi = t_mid;
end
```

Threshold is too low for too many samples exceed it.

```
end
```

```
adc_clip = t_hi;
```

After repeatedly halving the interval:
 $t_{lo} < \text{true solution} < t_{hi}$
Where the interval becomes extremely small.

$Q(z) = 0.5 \cdot \text{erfc}(z/\sqrt{2})$
 $Q(z) = P(Z > z)$ for $Z \sim N(0,1)$
Example, $Qfun(1) = 0.1587$
=> $P(Z > 1) = 15.87\%$

Total probability that the ADC input exceeds the threshold t

- sums= means of the Gaussian components.
- wts= weights (probabilities) of the Gaussian components.
- sigma_rest= common standard deviation of all components

- Let sums(i)= s_i; wts(i)= w_i
=> $S_{top} = s_i$ with probability w_i
=> $X = s_i + G$ where $G \sim N(0, \sigma_{rest}^2)$

Probability of exceeding threshold t

- $P(X > t | S_{top} = s_i)$
= $P(s_i + G > t)$
= $P(G > t - s_i)$
= $P((G/\sigma_{rest}) > (t - s_i)/\sigma_{rest})$
= $Q((t - s_i)/\sigma_{rest})$

Weight by state probability

- That state only occurs with probability: w_i
- $w_i \cdot Q((t - s_i)/\sigma_{rest})$

Sum over all states

$P(X > t) = \sum_i w_i Q\left(\frac{t - s_i}{\sigma_{rest}}\right)$

$P(X > \text{adc_clip}) = P_{qc}$ using bisection

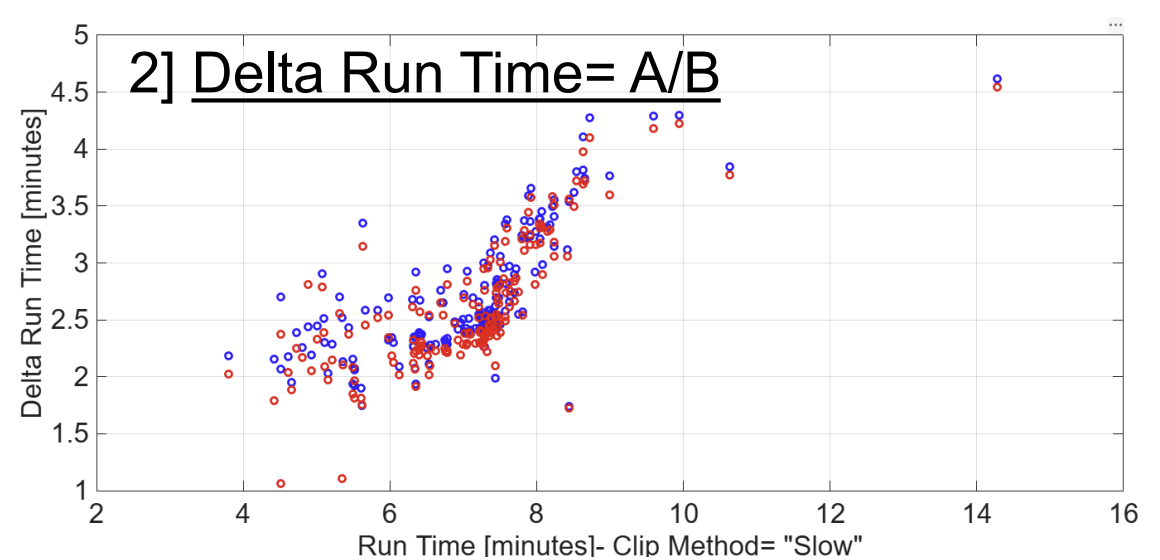
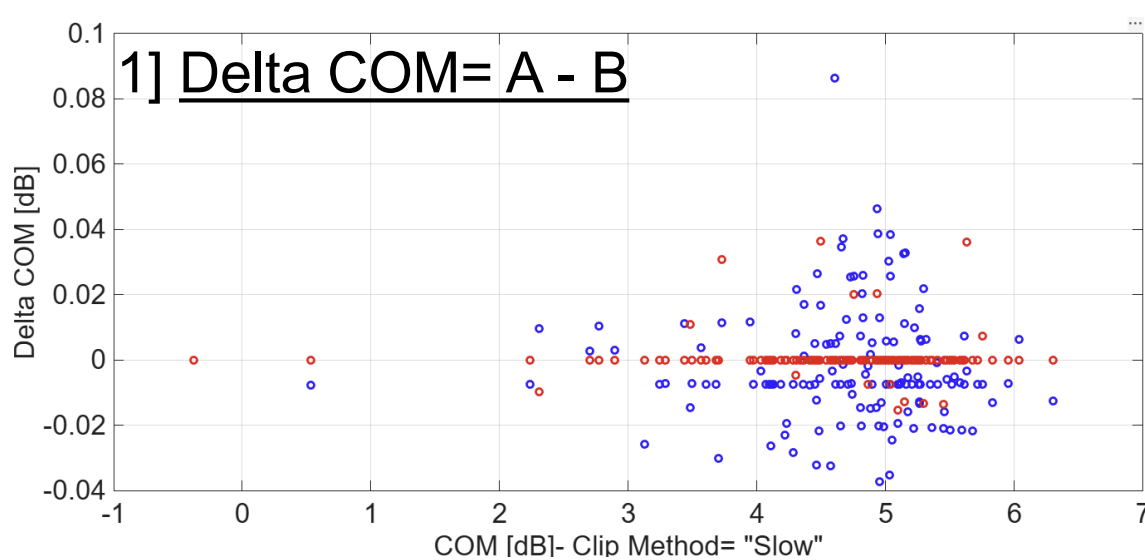
If too many samples exceed the threshold, the threshold is too low, so increase it. Otherwise decrease it. Each iteration halves the search interval until the desired accuracy is reached.

- To be applied in get_PSDs() which is called during EQ search.

H. Results

o A= Clip Method of “Slow” vs. B= Clip Method of “Fast”

o A= Clip Method of “Slow” vs. B= Clip Method of “Fast_1”



- Evaluated KR and CR channels from the IEEE 802.3dj Public Area for Class A package model Test 1.
- The proposed method “Fast_1” gives results comparable to “Slow”.

2. Commit Request

A. Purpose for Commit Request

- Commit request related to Clip Method of “Fast_1”.
- Fast Top-K Gaussian Mixture modeling preserves the clipping-threshold physics of the Slow PDF/CDF method while significantly reducing computational complexity during COM equalization search.
- Branch: adc_clip_fast_1.
https://opensource.ieee.org/hansel.dsilva/com_code/-/branches
- Git branch file pointer:
https://opensource.ieee.org/hansel.dsilva/com_code/-/blob/adc_clip_fast_1/release/com_ieee8023_4p16p0_adc_clip_fast_1.m

B. Navigate to Hansel_Dsilva/com_code

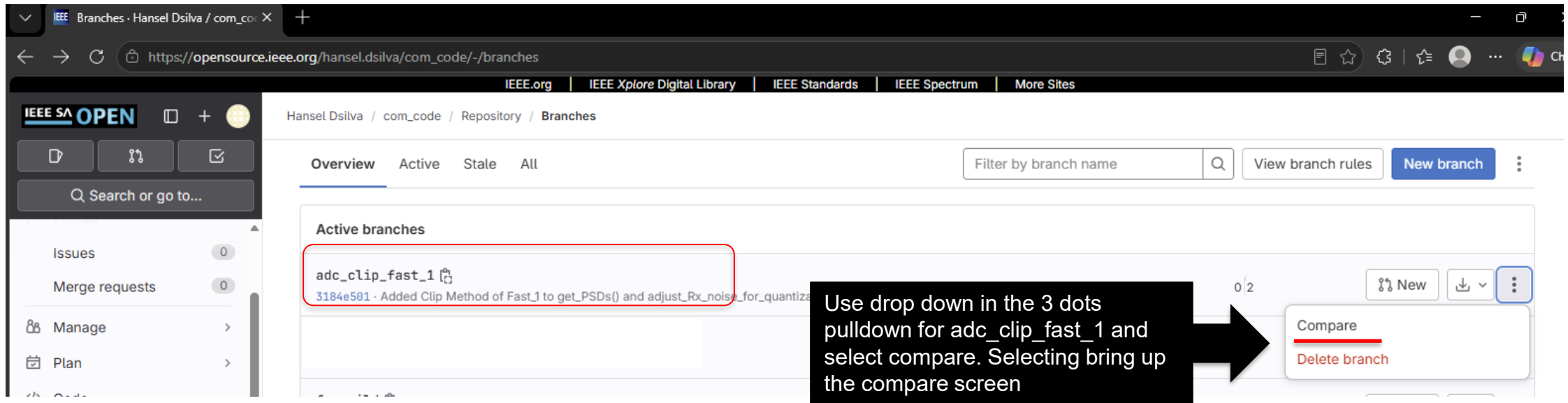
https://opensource.ieee.org/802-com/com_code/-/forks

The screenshot shows the IEEE Open Source Project page for the 802-COM / com_code project. The page displays a list of forks, with 11 forks in total (8 public, 0 internal, and 3 private). The fork by Hansel Dsilva is highlighted with a red box and a black arrow pointing to it with the text "Navigate to Hansel_Dsilva/com_code".

Forker	Repository	Description	Stars	Forks	Issues	Comments	Updated
Baidyanath Dutta	com_code	Merge branch 'KR_PKGB_CONFIG_for_d2p1' into 'main'	0	0	0	0	Updated 2 months ago
Norman Swenson	com_code	add 4p10p0 beta release	0	0	0	0	Updated 4 months ago
Mohammad Shakiba	com_code	add configuration file folder and some configuration files	0	0	0	0	Updated 4 months ago
Adee Ran	com_code	Merge branch 'signal_flow_graph' into 'main'	0	0	0	0	Updated 3 months ago
Hansel Dsilva	com_code	rename for 4.12.0_beta1 to 4.12.0	0	0	0	0	Updated 16 minutes ago
Kent Lusted	com_code	rename for 4.12.0_beta1 to 4.12.0	0	0	0	0	Updated 2 weeks ago
Richard Mellitz	com_code	rename for 4.12.0_beta1 to 4.12.0	2	0	0	0	Updated 1 day ago
Adam Gregory	com_code	deleting fork version	0	0	0	0	Updated 1 month ago

C. Checking changes of a branch to the last release

https://opensource.ieee.org/hansel.dsilva/com_code/-/branches



IEEE SA OPEN

Hansel Dsilva / com_code / Repository / Branches

Overview Active Stale All

Filter by branch name View branch rules New branch

Active branches

adc_clip_fast_1
3184e581 · Added Clip Method of Fast_1 to get_PSDs() and adjust_Rx_noise_for_quantiza

0/2 New Download

Use drop down in the 3 dots pulldown for adc_clip_fast_1 and select compare. Selecting bring up the compare screen

Compare
Delete branch

D. Code Updates

- 1 m file changed.
- 1 m files added.
- No additional config csv and xlsx files are included.

Files changed,
src/get_PSDs.m

Files added,
src/fast_clip_topK.m

E. New or changed keywords

- Configuration file parameters.

Keywords (not case sensitive)	Default	Units	Comment
Clip Method	Slow	None	■ It takes “Slow”, “Fast” or “Fast_1”. ■ “Slow” involves PDF/ CDF based. ■ “Fast” involves $\text{sum}(\text{abs}(\text{sampled_pulse_response}))$ which does not account for noise. ■ “Fast_1” involves the Fast Top-K Gaussian Mixture ADC Clip Solver.

New Outputs

None

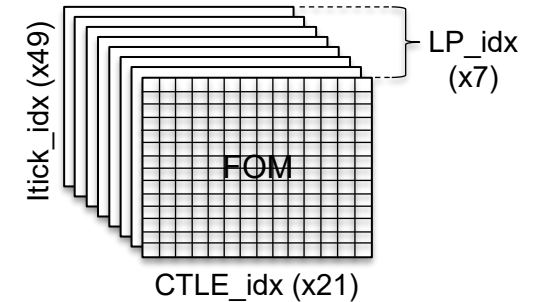
Thank You

A. Equalization Search Space

Parameter	Definition	Values	Count
c(-2)	TX equalizer, coeff. -2	[0:0.02:0.14]	8
c(-1)	TX equalizer, coeff. -1	[-0.34:0.02:0]	18
c(0)	TX equalizer, minimum coeff. 0	0.54	N.A.
c(1)	TX equalizer, coeff 1	[-0.2:0.02:0]	11
ctle_index	CTLE Gain 1	[-20:1:0]	21
g_LP_index	CTLE Gain 2	[-6:1:0]	7
itickn	Sampling clock offset	[-24:1:24]	49

```
% Skip combinations with small  
values of c(0), not guaranteed to  
be supported by all transmitters.  
if txffe_cur < param.tx_ffc_c0_min  
    continue;  
end
```

TXLE combinations = $(8 \times 18 \times 11) - 303 = 1281$



= $49 \times 21 \times 7$

= 7203 combinations without TXLE

= $1281 \times 49 \times 21 \times 7$

= 9.227043e6 combination with TXLE

= 9.227043 Million combinations with TXLE

- With TXLE sweep being OFF there are **7,203 combinations**.
- With TXLE sweep being ON there are **9.227043 Million combination**.

B. List of CR and KR Channels on 802.3dj Public Area

- Below are the 200 Gb/s CR and KR channel data (TP0 to TP5)- with crosstalk.

1. akinwale_3dj_01_2311: x4
2. weaver_3dj_02_2311: x12
3. akinwale_3dj_01_2310: x7
4. lim_3dj_07_2309: x1
5. lim_3dj_04_230629: x1
6. lim_3dj_03_230629: x1
7. weaver_3dj_elec_01_230622: x8
8. shanbhag_3dj_01_2305: x6
9. shanbhag_3dj_02_2305: x4
10. kocsis_3dj_02_2305: x5
11. weaver_3dj_02_2305: x36
12. mellitz_3dj_02_elec_230504: x27
13. weaver_3dj_02_2303: x5
14. mellitz_3dj_02_2303: x54

Parameter	Symbol	Value	Units
Device termination model			
Single-ended device capacitance for stage 1	$C_d^{(1)}$	40×10^{-6}	nF
Single-ended device capacitance for stage 2	$C_d^{(2)}$	90×10^{-6}	nF
Single-ended device capacitance for stage 3	$C_d^{(3)}$	110×10^{-6}	nF
Single-ended device series inductance for stage 1	$L_s^{(1)}$	0.13	nH
Single-ended device series inductance for stage 2	$L_s^{(2)}$	0.15	nH
Single-ended device series inductance for stage 3	$L_s^{(3)}$	0.14	nH
Single-ended bump capacitance	C_b	30×10^{-6}	nF
Device package model, class A			
Transmission line parameter γ_0	γ_0	5×10^{-4}	1/mm
Transmission line parameter a_1	a_1	8.9×10^{-4}	ns ^{1/2} /mm
Transmission line parameter a_2	a_2	2×10^{-4}	ns/mm
Transmission line parameter τ	τ	6.141×10^{-3}	ns/mm
Transmission line 1 length, Test 1	$z_p^{(1)}$	33	mm
Transmission line 1 length, Test 2	$z_p^{(1)}$	12	mm
Transmission line 1 characteristic impedance	$Z_c^{(1)}$	87.5	Ω
Transmission line 2 length	$z_p^{(2)}$	1.8	mm
Transmission line 2 characteristic impedance	$Z_c^{(2)}$	92.5	Ω
Device package model, class B			
Transmission line parameter γ_0	γ_0	5×10^{-4}	1/mm
Transmission line parameter a_1	a_1	6.5×10^{-4}	ns ^{1/2} /mm
Transmission line parameter a_2	a_2	2.93×10^{-4}	ns/mm
Transmission line parameter τ	τ	6.141×10^{-3}	ns/mm
Transmission line 1 length, Test 1, Tx / Rx	$z_p^{(1)}$	45 / 44	mm
Transmission line 1 length, Test 2, Tx / Rx	$z_p^{(1)}$	30 / 29	mm
Transmission line 1 characteristic impedance	$Z_c^{(1)}$	87.5	Ω
Transmission line 2 length	$z_p^{(2)}$	2	mm
Transmission line 2 characteristic impedance	$Z_c^{(2)}$	95	Ω
Transmission line 3 length	$z_p^{(3)}$	1.3	mm
Transmission line 3 characteristic impedance	$Z_c^{(3)}$	100	Ω
Transmission line 4 length	$z_p^{(4)}$	1.5	mm
Transmission line 4 characteristic impedance	$Z_c^{(4)}$	78	Ω

- Total of 171 channels from the IEEE 802.3dj Public Area are evaluated against the Class A package model for Test 1.
 - Total cases= 171 cases.

C. COM Spreadsheet for Package Class A

data rate, die load, ref impedance				I/O control			Operational				SAVE_CONFIG2MAT	0	
Parameter	Setting	Units	Information	DIAGNOSTICS	1	logical	ERL Pass threshold	11	dB			Receiver testing	
f_b	106.25	GBd		DISPLAY_WINDOW	1	logical	COM Pass threshold	3	db			RX_CALIBRATION	0
f_min	0.05	GHz		CSV_REPORT	0	logical	DER_0	2.00E-04				Sigma BBN step	5.00E-03
Delta_f	0.01	GHz		RESULT_DIR	.\results\CAKR_{date}\		T_r	0.00400	ns			ICN parameters	
C_d	[0.4e-4 0.9e-4 1.1e-4; 0.4e-4 0.9e-4 1.1e-4]	nF	[TX RX]	SAVE_FIGURES	0	logical	FORCE_TR	1	logical	for legacy but required		T_t	6.000
L_s	[0.13 0.15 0.14; 0.13 0.15 0.14]	nH	[TX RX]	Port Order	[1 3 2 4]		PMD_type	C2C	for MMSE use C2C only			f_v	0.371
C_b	[0.3e-4 0.3e-4]	nF	[TX RX]	RUNTAG	KR_pkgA		EW	1				T_ft	4.250
R_0	46.25	Ohm		COM_CONTRIBUTION	0	logical	MLSE	1	logical			T_nt	4.250
PKG_NAME	PKG_LowR_CLASSA PKG_LowR_CLASSA		TX RX				ts_anchor	1				f_f	0.524
z_pselect	[1 2]						sample_adjustment	[-24 24]				f_n	0.524
L	4						Local Search	2				f_1	0.010
M	32						flim	6.70E+10	Hz			f_2	67.000
filter and Eq							zero_pad	1	logical			A_ft	0.600
f_r	0.55	*fb					Filter: Rx FFE					A_nt	0.600
c(0)	0.55		min				ffe_pre_tap_len	6	UI	d_w			
c(-1)	0	[-0.34; 0.02]	[min:step:max]				ffe_post_tap_len	8	UI	N_fix-d_w			
c(-2)	0	[0.14; 0.02]	[min:step:max]				ffe_pre_tap1_max	0.7	(normalized)	w_max(d_w) and -w_min(d_w)			
c(-3)	0		[min:step:max]				ffe_post_tap1_max	0.7	(normalized)	w_max(d_w+2) and -w_min(d_w+2)			
c(-4)	0		[min:step:max]				ffe_tapn_max	0.7	(normalized)	all other fixed w_max and w_min			
c(1)	0	[-0.2; 0.02]	[min:step:max]				num_ui_RXFF_noise	4096					
N_b	1	UI					Floating Tap Control						
b_max(1)	0.85		As/dffe1				N_bg	2	0 1 2 or 3 groups	N_wg			
b_max(2..N_b)	0		not used				N_bf	4	taps per group	N_wf			
b_min(1)	0		As/dffe1				N_f	80	UI span for floating taps	Nmax-d_w-1			
b_min(2..N_b)	0	S	not used				bmaxg	0.05	max FFE value for floating taps	all floating w_max and w_min			
g_DC	[-20; 1.0]	dB	[min:step:max]				N_tail_start	9	(UI) start of tail taps limit	not supposed to be used but untested			
f_z	42.50	GHz					TS_SRCH_MODE	full-sweep					
f_p1	42.50	GHz					Clip Method	Slow					
f_p2	106.25	GHz					Non-zero Local Search Method	0					
g_DC_HP	[-6; 1.0]		[min:step:max]										
f_HP_PZ	1.328125	GHz											

Note.

Set Clip Method to "Slow"

Set TS_SRCH_MODE to "full-sweep"

Set sample_adjustment to "[-24 24]"

Non-zero Local Search Method" to 0.

Toggle Clip Method to "Slow", "Fast" and "Fast_1".

..START	PKG_LowR_CLASSA	Units	Information
Parameter	Setting		
package_tl_gamma0_a1_a2	[0.0005 0.00089 0.0002]		
package_tl_tau	0.006141	ns/mm	
package_Z_c	[87.5 87.5; 95 95; 100 100; 100 100]	Ohm	
R_d	[46.25 46.25]	Ohm	[TX RX]
z_p(TX)	[12 33 33 33; 1.8 1.8 1.8 1.8; 0.000; 0.000]	mm	[test cases]
z_p(NEXT)	[12 33 33 33; 1.8 1.8 1.8 1.8; 0.000; 0.000]	mm	[test cases]
z_p(FEXT)	[12 33 33 33; 1.8 1.8 1.8 1.8; 0.000; 0.000]	mm	[test cases]
z_p(RX)	[12 33 33 33; 1.8 1.8 1.8 1.8; 0.000; 0.000]	mm	[test cases]
C_p	[0.4e-4 0.4e-4]	nF	[TX RX]
A_v	0.385	V	Vf=0.400
A_fe	0.385	V	Vf=0.399
A_ne	0.481	V	Vf=0.400
..END			