INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

HYDERABAD

Physically Unclonable Functions

Srihari Padmanabhan, Aikya Oruganti

11th November, 2025
11th December, 2025

The Idea

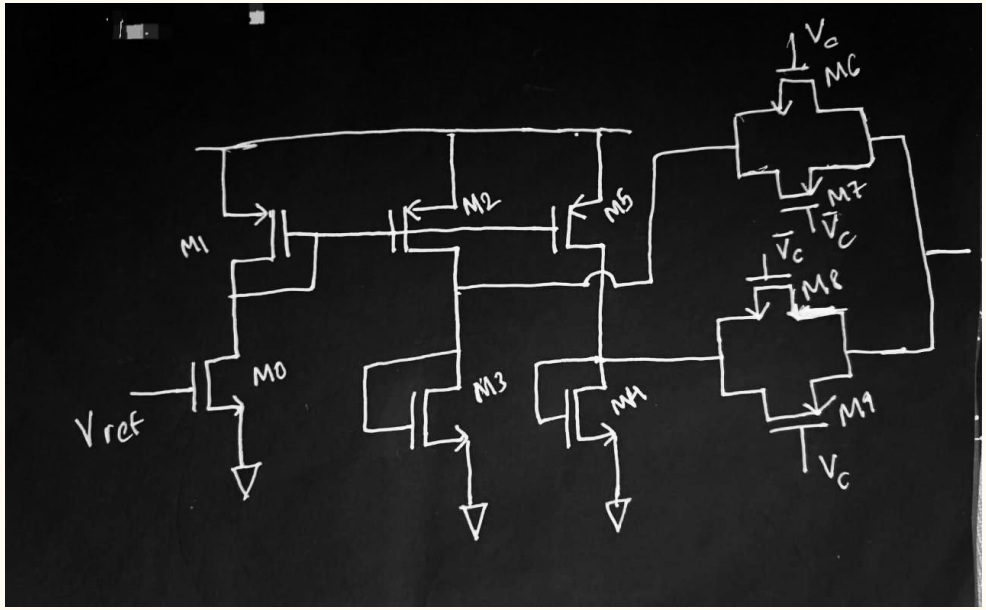
- Leverage the variations in transistors configured as PMOS current mirrors to generate distinct currents
- Regenerate V_ref using diode-connected NMOS
- Choose a V_ref for the next stage based on the challenge bit
- Implement two parallel chains, each consisting of 32 cascaded stages
- Compare the voltages to generate a response bit

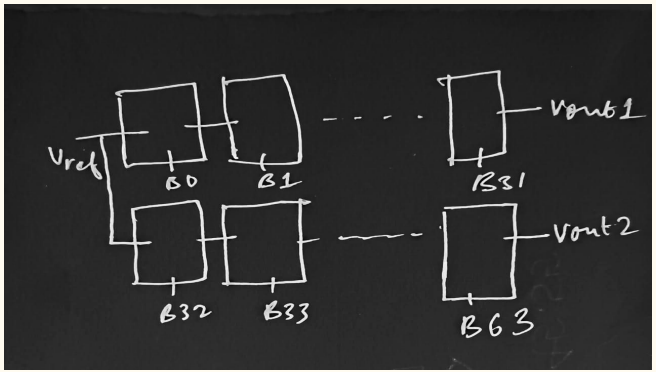
Why it works

- Current mirrors are highly sensitive to threshold voltage and mobility variations from manufacturing, giving excellent device-to-device uniqueness.
- Nonlinear analog current behavior and the amplification of minor mismatches across stages result in high entropy and randomness in response bits.
- The inherent analog nonlinearity in current mirrors makes modeling attacks (such as machine learning) several times harder than for conventional (digital) strong PUFs, giving it a significant security edge.

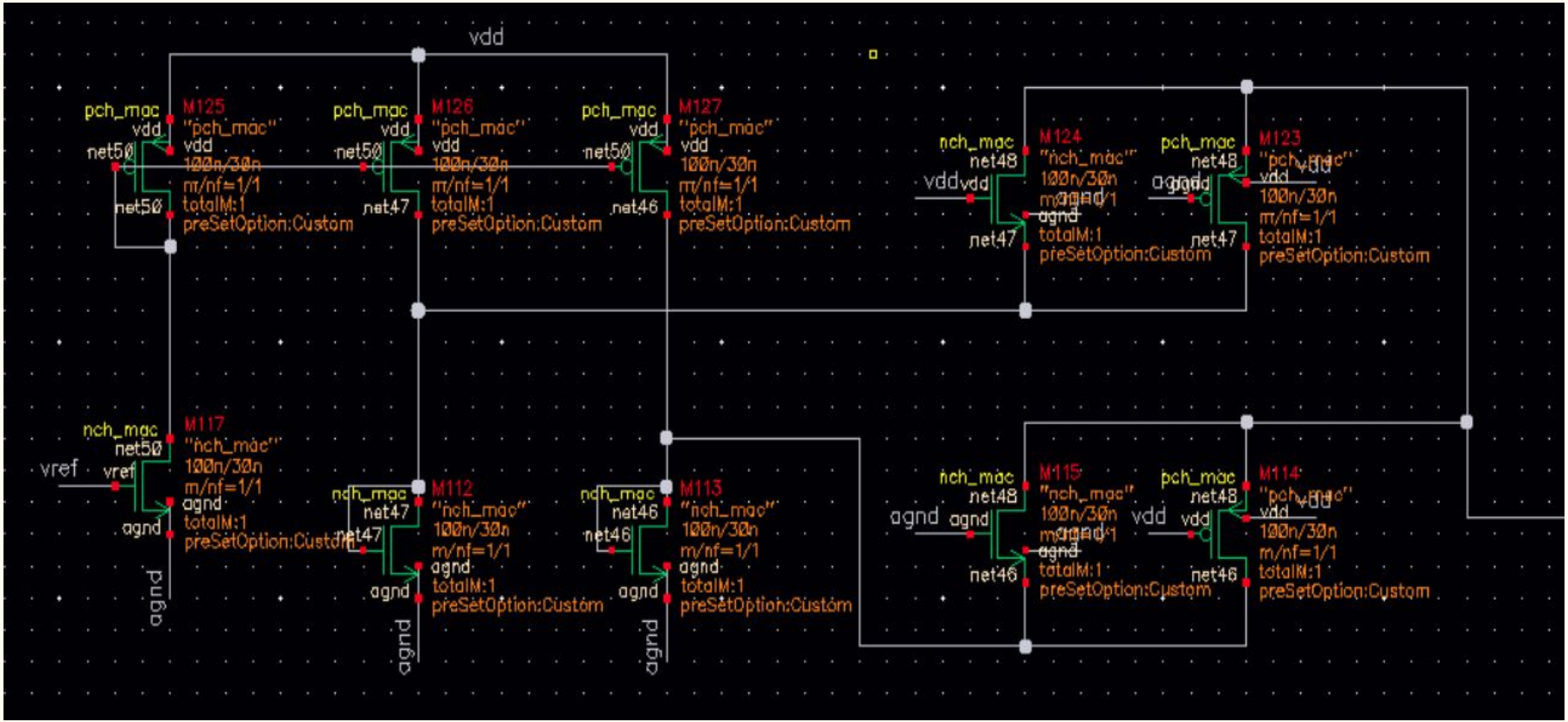
The circuit

- M0: acts as a reference NMOS, drawing some current I set by V_ref
- M1, M2, M5: current mirrors, I is replicated in the next two arms
- M3, M4: diode connected NMOS, Regenerate Vref corresponding to the current flowing through that arm
- M6, M7: pass the drain voltage of M3 if the challenge bit is 1
- M8,M9: pass the drain voltage of M4 if the challenge bit is 0

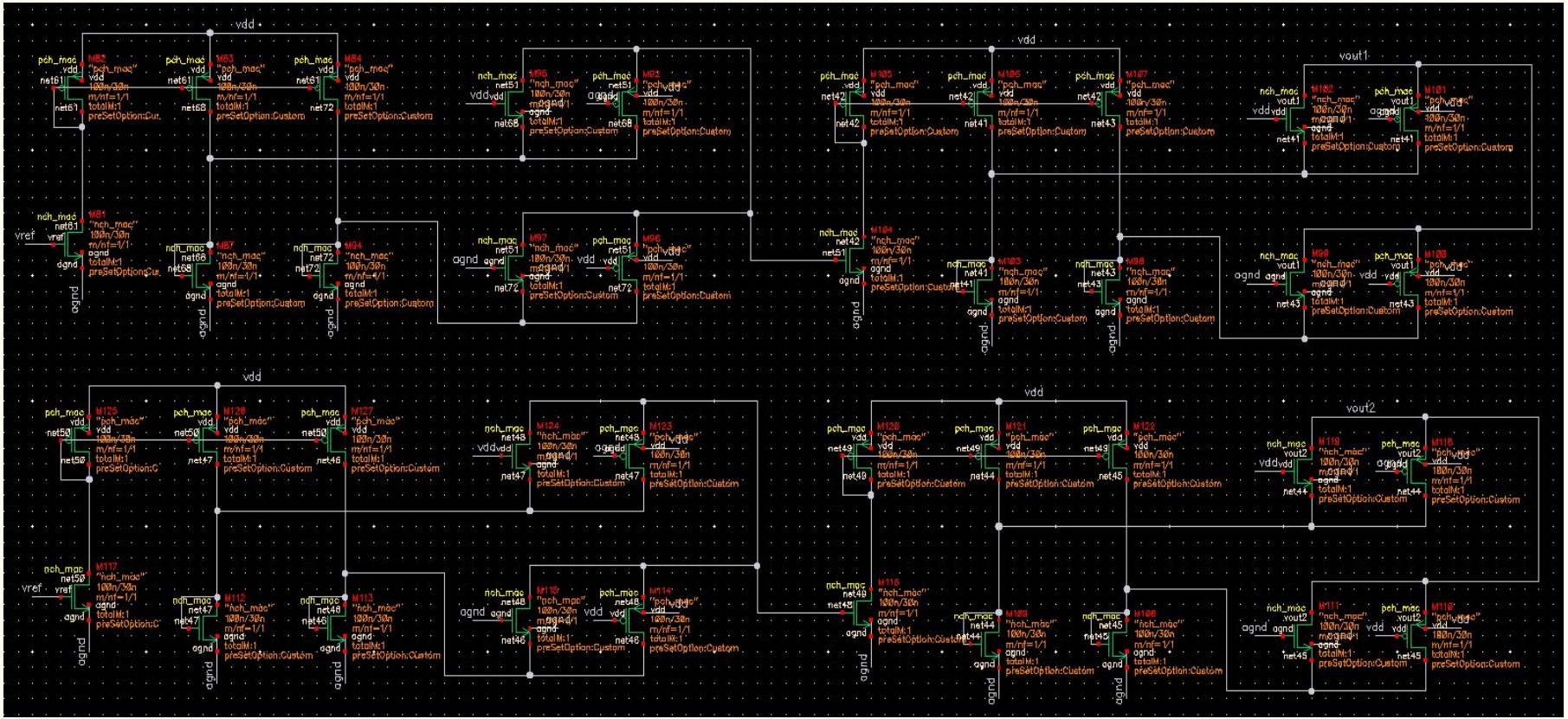




Schematic- Single Stage



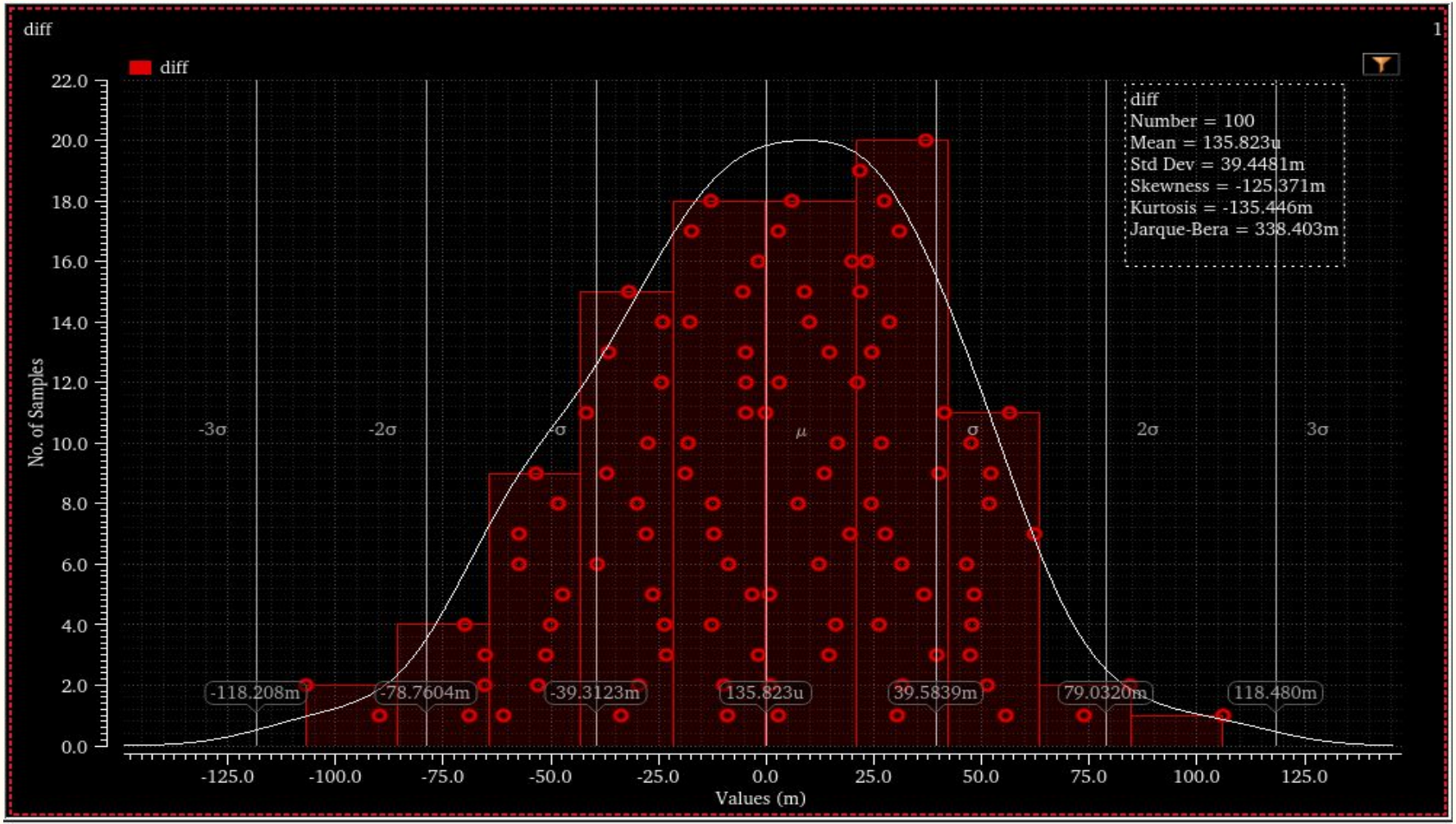
Setup for a 4-bit challenge



Monte Carlo

Test	Name	Yield	Min	Target	Max	Mean	Std Dev	Cpk	Errors
Yield Estimate: 100 %(100 passed/100 pts) Confidence Level: <not set> Filter: <not set>									
– tgate_dc									
	vout1	100% (100/1...	774.4m	info	946.4m	885m	25.34m		0
	vout2	100% (100/1...	796.5m	info	955.8m	885.2m	28.94m		0
	diff	100% (100/1...	-106.9m	info	106.1m	135.8u	39.45m		0

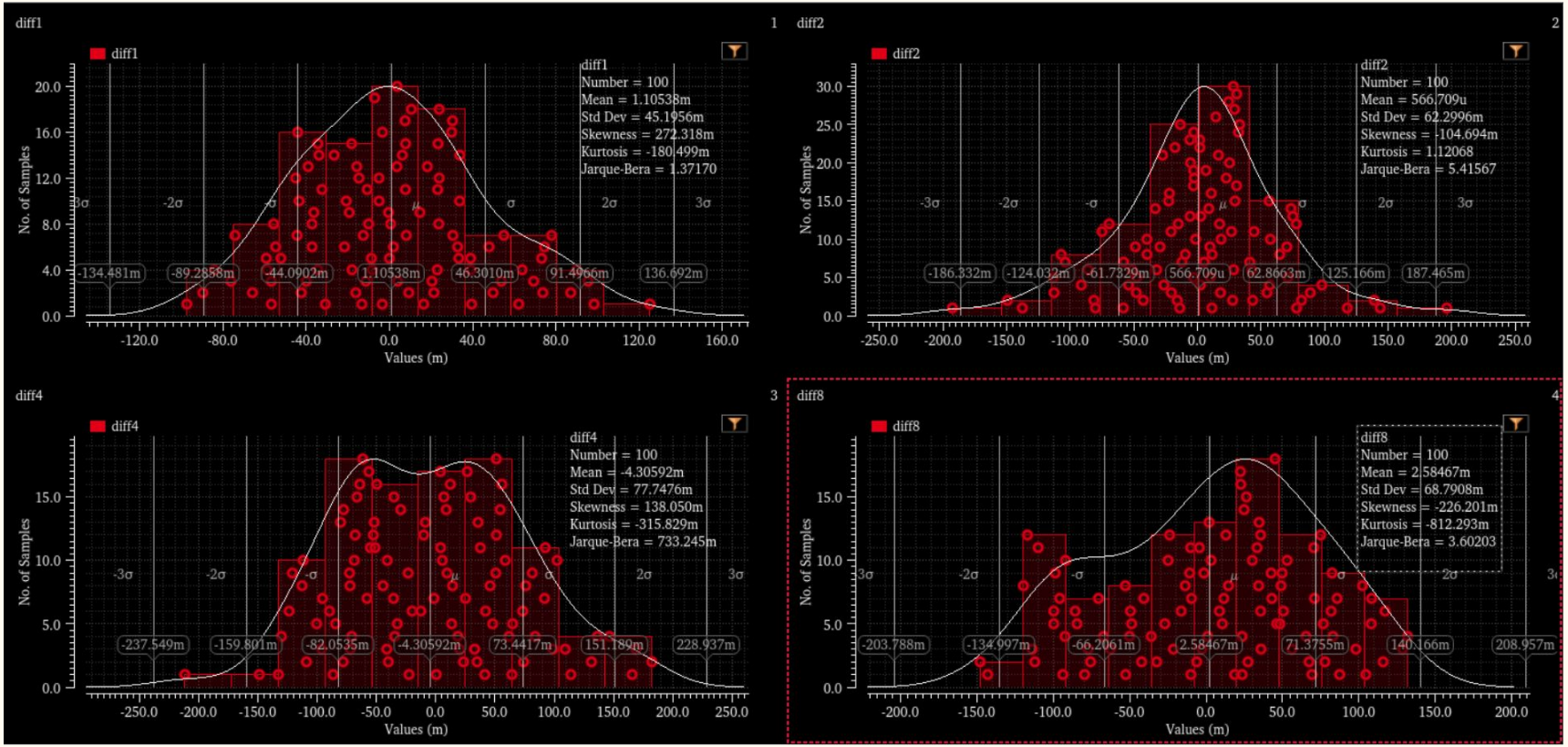
vout1, vout2: Outputs of the two chains
diff= vout1-vout2
response=1 if diff>0, 0 otherwise

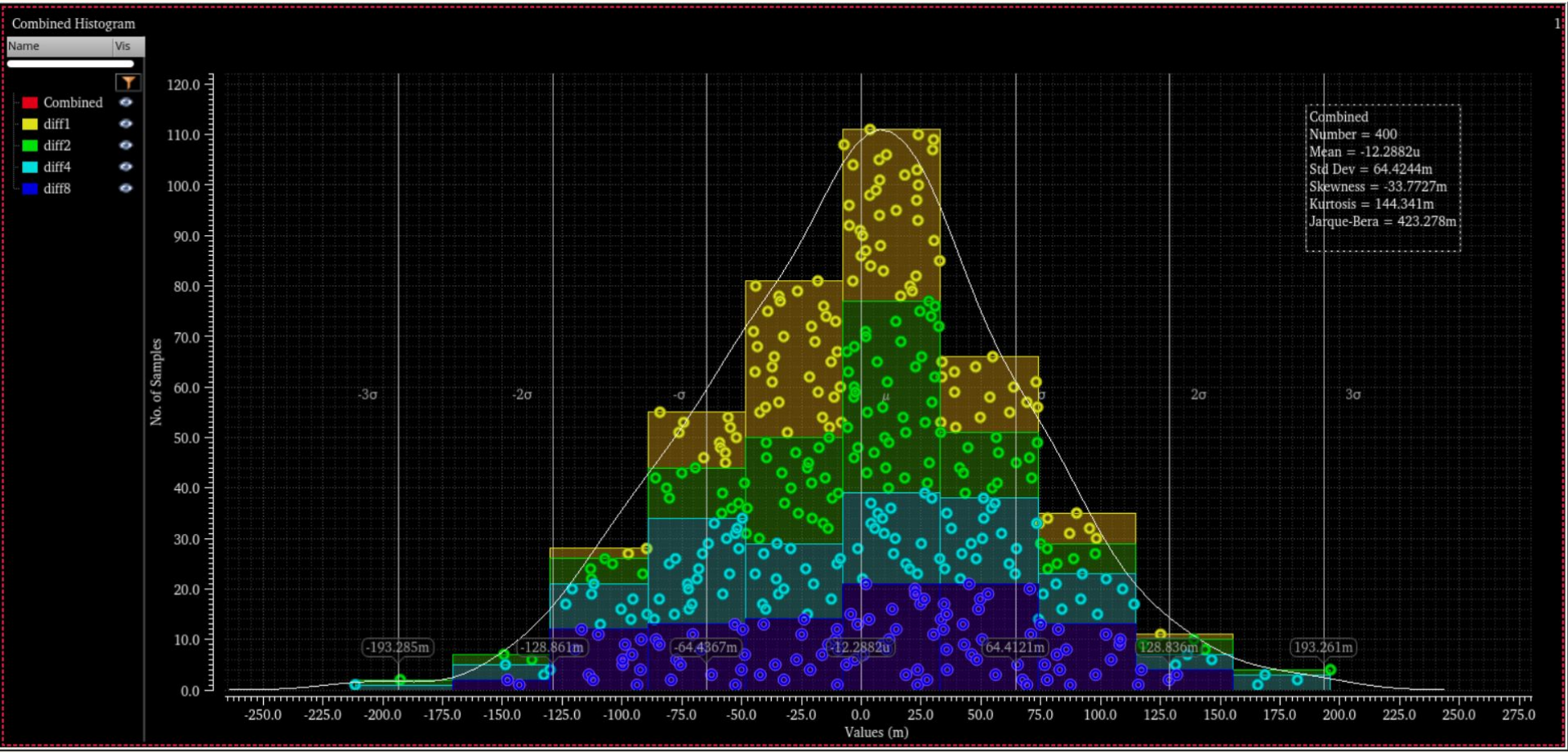

diff
1
diff
diff
Number = 100
Mean = 135.823u
Std Dev = 39.4481m
Skewness = -125.371m
Kurtosis = -135.446m
Jarque-Bera = 338.403m
No. of Samples
Values (m)
-3σ
-2σ
-σ
μ
σ
2σ
3σ
-118.208m
-78.7604m
-39.3123m
135.823u
39.5839m
79.0320m
118.480m
22.0
20.0
18.0
16.0
14.0
12.0
10.0
8.0
6.0
4.0
2.0
0.0
-125.0
-100.0
-75.0
-50.0
-25.0
0.0
25.0
50.0
75.0
100.0
125.0

Results

- The histogram illustrates the distribution of the difference in output voltages of the two chains.
- The distribution adheres approximately to a Gaussian profile, confirming non-ideal, yet symmetric(low skewness), process variability across the chains.
- The mean(135.82 μ) is close to zero, supporting unbiased response bit extraction.
- A moderate standard deviation(~39 mV) ensures sufficient response dispersion, enabling strong inter-device uniqueness.
- Low Kurtosis(peak sharpness) helps bit mapping stability.
- The values are concentrated around the mean indicating high entropy.

Some more results (Varying number of stages)




Combined Histogram
Name
Vis
Combined
diff1
diff2
diff4
diff8
Combined
Number = 400
Mean = -12.2882u
Std Dev = 64.4244m
Skewness = -33.7727m
Kurtosis = 144.341m
Jarque-Bera = 423.278m
No. of Samples
Values (m)
-3σ
-2σ
-σ
μ
σ
2σ
3σ
-193.285m
-128.861m
-64.4367m
-12.2882u
64.4121m
128.836m
193.261m

Test Setup

- 8 bit challenge per PUF
- 256 total challenges - output response of 256 bits
- 30 chips (iterations of Monte Carlo mismatch)

Challenges Faced:

- Limited number of Monte Carlo simulations can be run (num chips * num bits in response $<$ 10k). Cannot replicate research paper test setup eg 25 chips, 10k challenges. (High compute + RAM requirement)

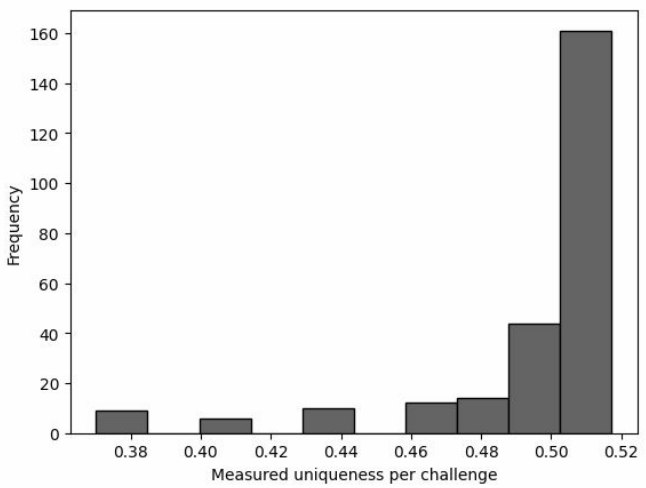
Results

- Uniqueness: 0.495
- Reliability: 0.992
- Uniformity: 0.523
- Bit aliasing: 0.523

Uniqueness

Overall uniqueness: Averaged Hamming distance between 256 length outputs over all pairs of chips. Ideal value is 50%.

```python
def challenge_wise_uniqueness(all_bits):
    """
    For each challenge j, compute mean HD over chip pairs (column-wise).
    Returns:
      ch_uniq: shape (n_challenges,)
    """
    n_chips, n_ch = all_bits.shape
    ch_uniq = np.zeros(n_ch)
    for j in range(n_ch):
        col = all_bits[:, j]           # (n_chips,)
        pair_hd = []
        for i in range(n_chips - 1):
            pair_hd.append(col[i] ^ col[i+1:])  # xor vs later chips
        ch_uniq[j] = np.mean(np.concatenate(pair_hd))
    return ch_uniq
```



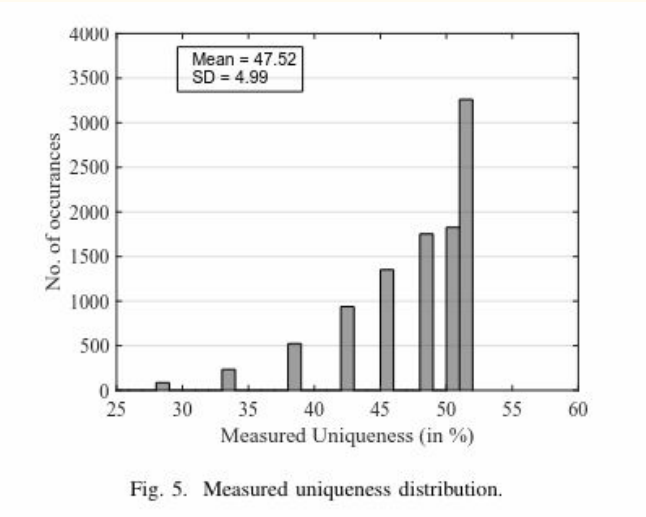
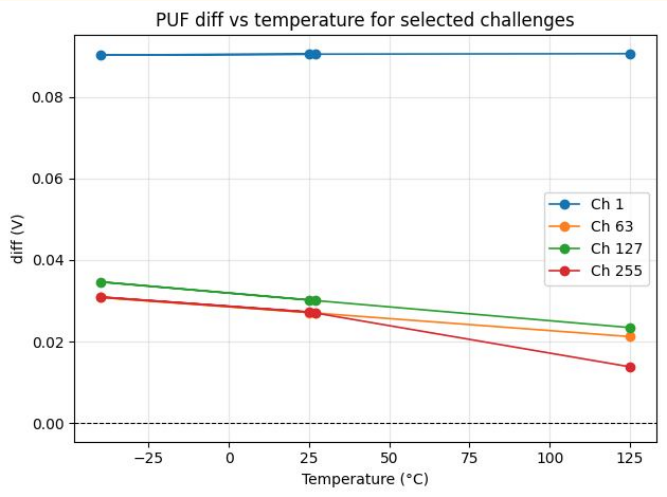


Fig. 5. Measured uniqueness distribution.

References [1] G. C. Gisha, A. S. Chakraborty, R. S. Chakraborty, B. A. Jose, and J. Mathew, “Diode-Triode Current Mirror Inverter PUF: A Novel Mixed-Signal Low Power Analog PUF,” in *Proc. IEEE Int. Midwest Symp. Circuits Syst. (MWSCAS)*, 2023, doi: 10.1109/MWSCAS57524.2023.10405902.

Reliability

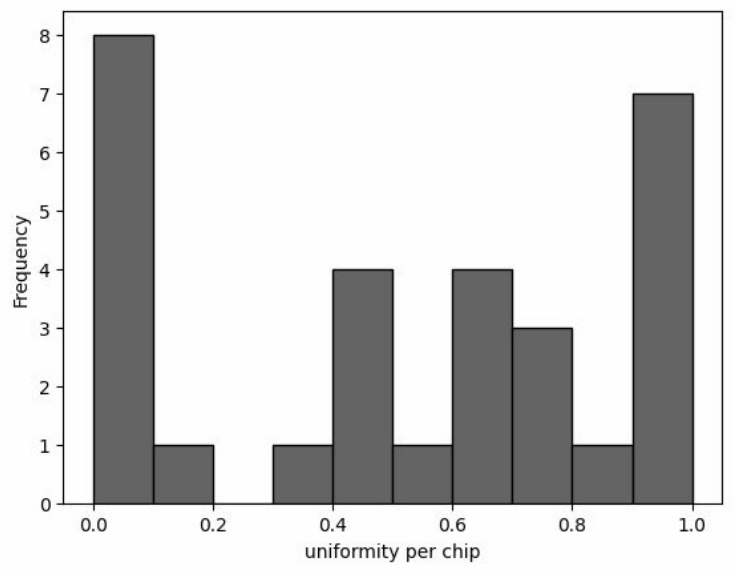
Reliability measures the rate at which a particular PUF generates the same bit string. Ideal value is 100 %.




```python
def reliability_from_diffs(diff_temps):
    """
    diff_temps : (N, 4)  [nominal, -40, 25, 125]

    Returns:
        rel_per_ch : (N,)  reliability per challenge
        rel_mean   : scalar mean reliability
        rel_std    : scalar std of reliability
    """
    # bits at all temps
    bits_all = (diff_temps > 0).astype(int)      # (N, 4)
    bits_ref = bits_all[:, 0][:, None]           # (N, 1), nominal as reference

    same = (bits_all == bits_ref).astype(int)    # 1 if matches nominal
    same_env = same[:, 1:]                       # (N, 3) for T1,T2,T3 only

    rel_per_ch = same_env.mean(axis=1)           # reliability per challenge
    rel_mean   = rel_per_ch.mean()               # average reliability
    rel_std    = rel_per_ch.std()                # spread of reliability

    return rel_per_ch, rel_mean, rel_std
```

Uniformity

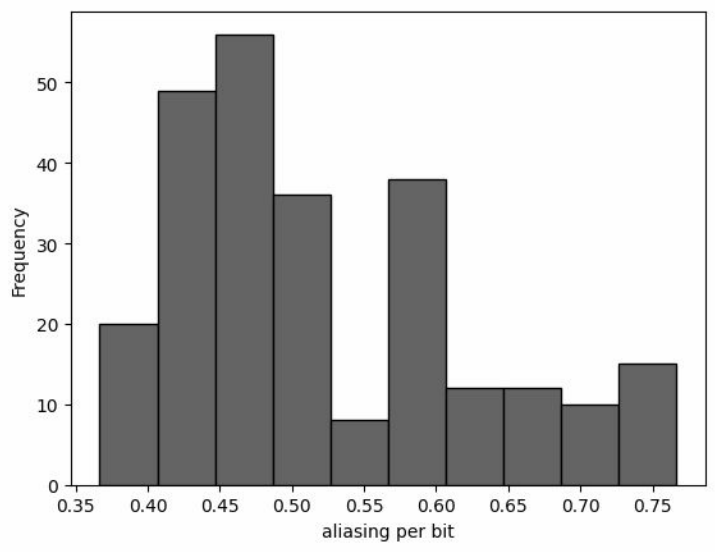
Uniformity measures the degree of variation of the responses of a single PUF chip across different challenges (here over 256 challenges). For maximum random behaviour the average uniformity should be 50%.




```
uniformities = []

for i in range(n_chips):

    uniformity = bits[i].mean()
    uniformities.append(uniformity)

uniformities = np.array(uniformities)  # shape (n_chips,)
```

Bit aliasing

Bit aliasing measures the degree of variation in responses of different PUF chips to the same challenge. The ideal value is 50% (Half the PUFs produce a 1, and half 0)




```
def bit_aliasing(all_bits):
    return all_bits.mean(axis=0)
```