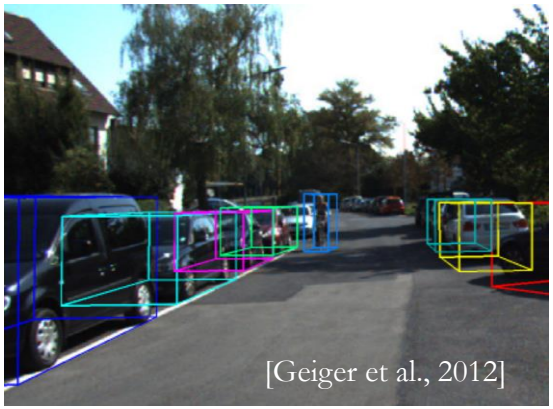
Architecting a Stochastic Computing Unit with Molecular Optical Devices

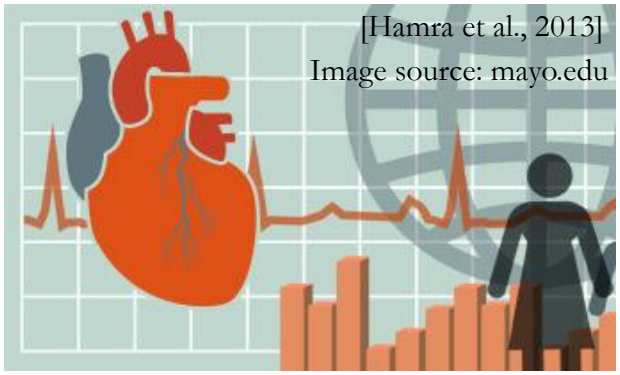
Xiangyu (Mike) Zhang, Ramin Bashizade, Craig LaBoda,
Chris Dwyer*, Alvin R. Lebeck

Duke University *Parabon Labs

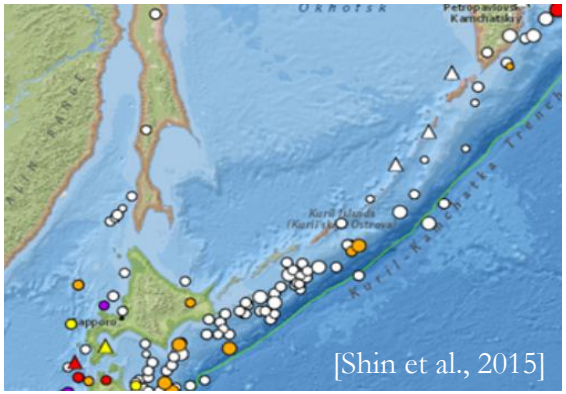
Stochastic (Probabilistic) Computing



Computer vision



Medical statistics



Earthquake prediction

Statistical Machine Learning

- Probabilistic algorithms (e.g. Markov Chain Monte Carlo):
  - Key to performance: generating samples.
- **Problem: Sampling overhead is TOO HIGH:**
  - A sample from a simple distribution: >500 cycles.

Using Markov Chain Monte Carlo method

Left image

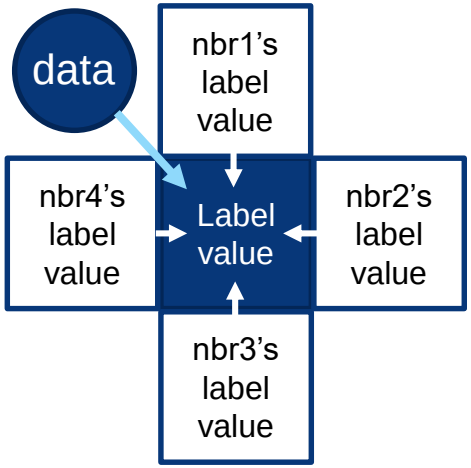
Right image

- Stereo Vision: reconstruct the depth information from image pairs.
  - Matching corresponding pixels.
  - Calculate disparity: location difference $x_l - x_r$.

Using Markov Chain Monte Carlo method

Left image

Right image



Label dependency

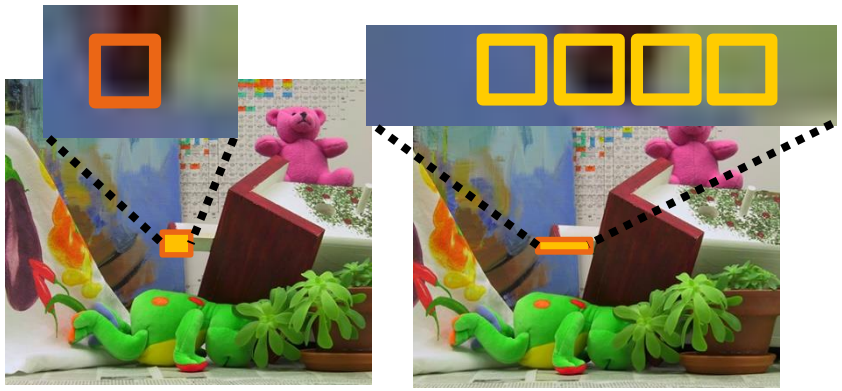
Energy function:

$E(Data, Lables)$

- Label: disparity on that pixel.

```
while(not converged) {
        for each pixel {
                Gibbs Sampling
        }
}
```

Using Markov Chain Monte Carlo method

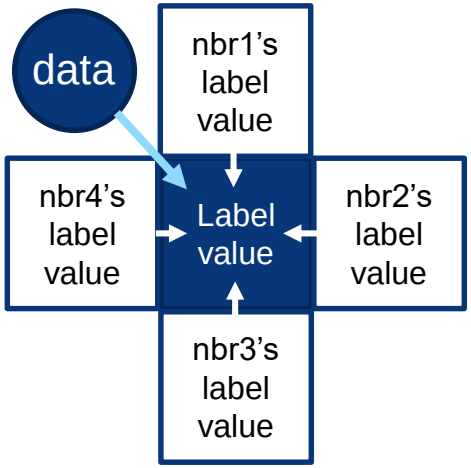


Left image

Right image

Pixel

Possible Matchings



Label dependency

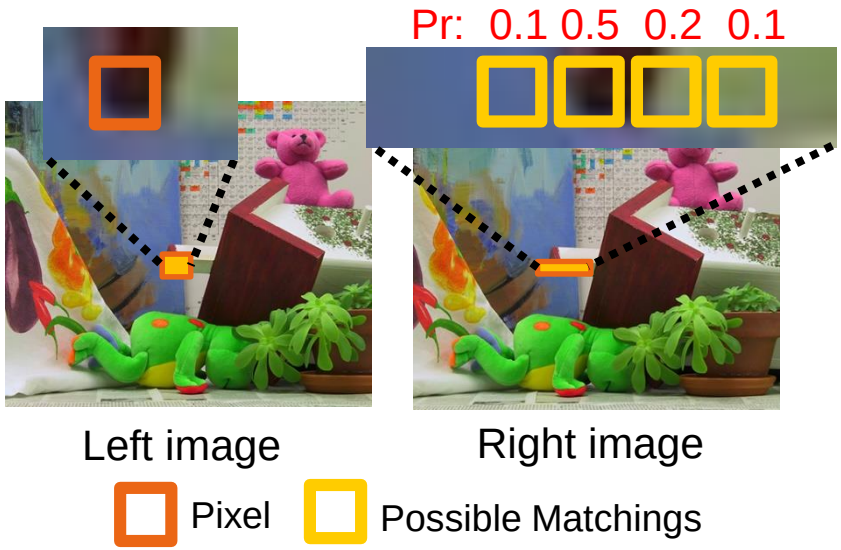
Energy function:

$E(Data, Lables)$

- Label: disparity on that pixel.

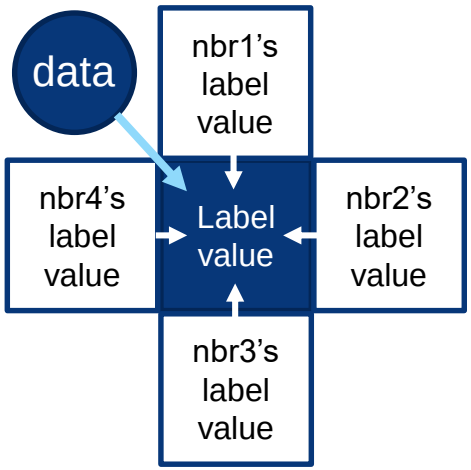
```
while(not converged) {
        for each pixel {
                1) compute probabilities of each possible label;
                2) randomly assign new label based on the probabilities;
        }
}
```

Using Markov Chain Monte Carlo method



Left image Right image

Pixel Possible Matchings



Label dependency

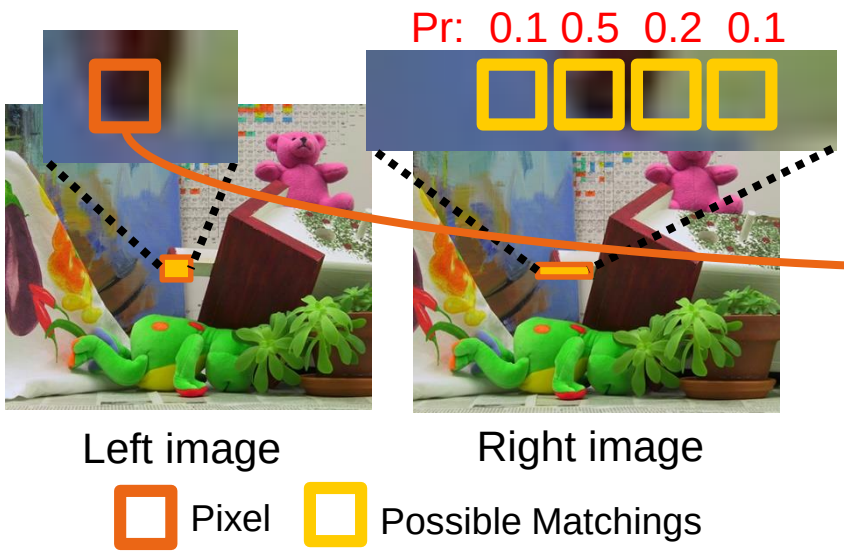
Energy function:

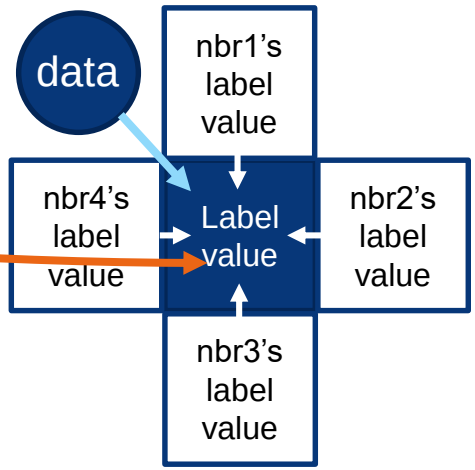
$E(Data, Lables)$

- Label: disparity on that pixel.

```
while(not converged) {
        for each pixel {
                1) compute probabilities of each possible label;
                2) randomly assign new label based on the probabilities;
        }
}
```

Using Markov Chain Monte Carlo method





Label dependency

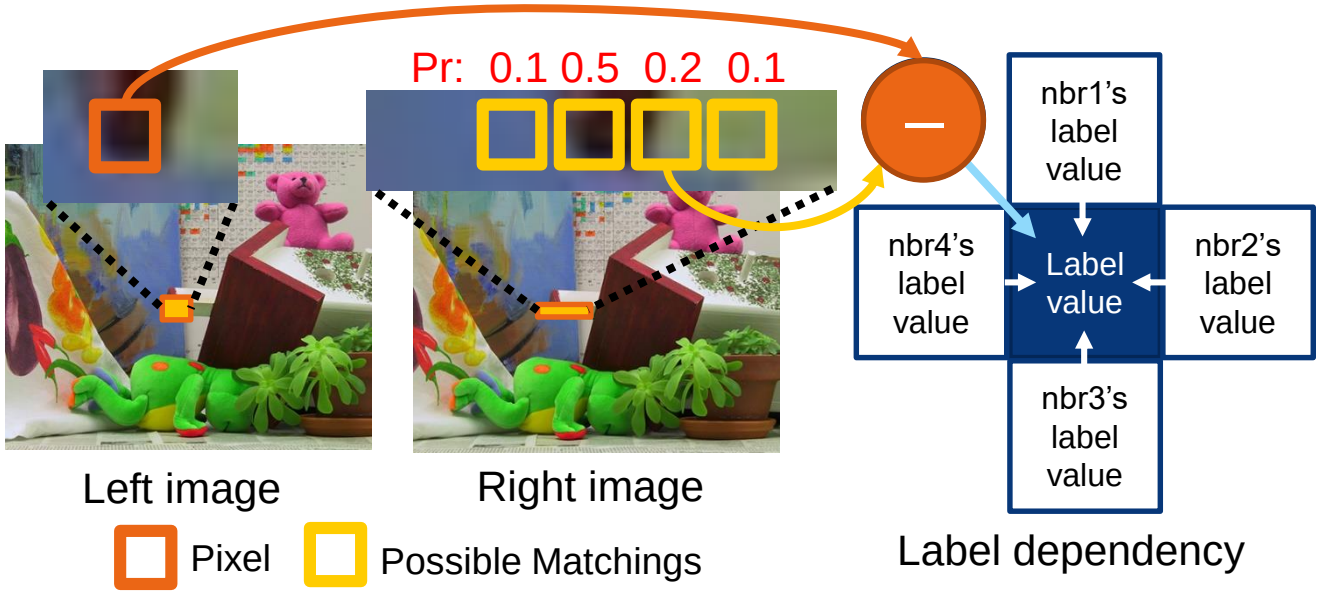
Energy function:

$E(Data, Lables)$

- Label: disparity on that pixel.

```
while(not converged) {
        for each pixel {
                1) compute probabilities of each possible label;
                2) randomly assign new label based on the probabilities;
        }
}
```

Using Markov Chain Monte Carlo method

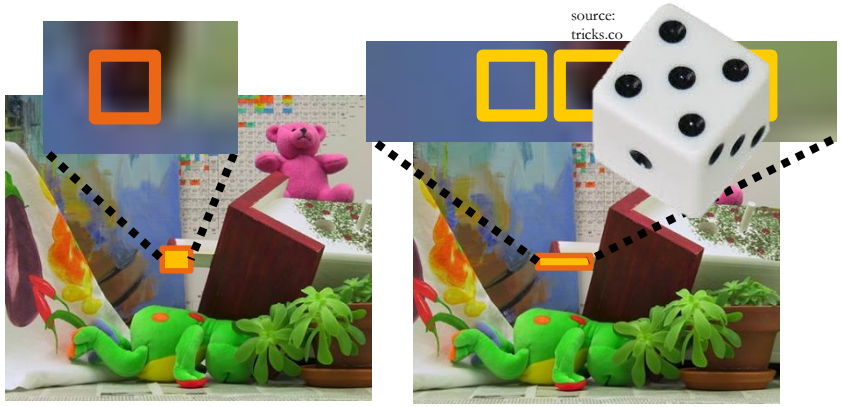


Energy function:
$E(Data, Lables)$

- Label: disparity on that pixel.

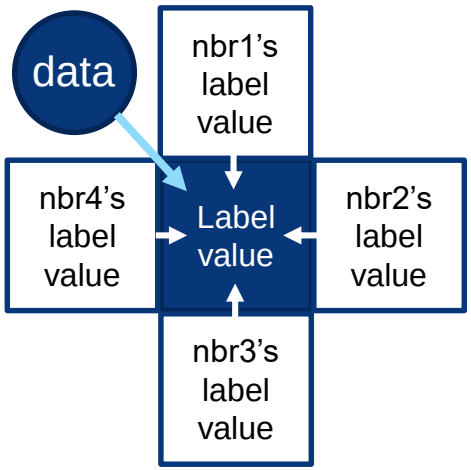
```
while(not converged) {
        for each pixel {
                1) compute probabilities of each possible label;
                2) randomly assign new label based on the probabilities;
        }
}
```

Using Markov Chain Monte Carlo method



Left image Right image

Pixel Possible Matchings



Label dependency

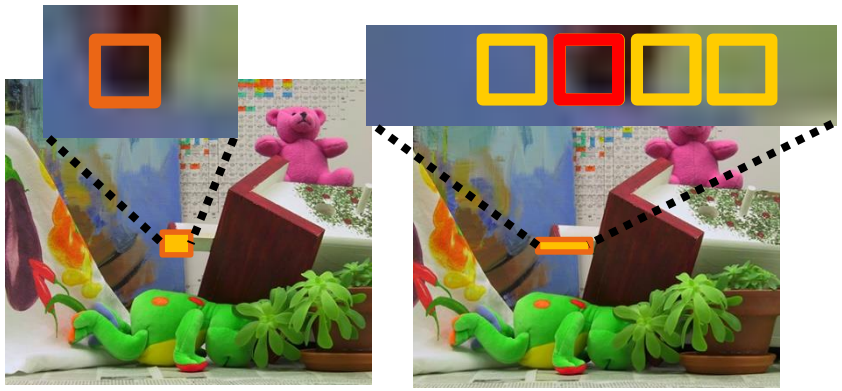
Energy function:

$E(Data, Lables)$

- Label: disparity on that pixel.

```
while(not converged) {
        for each pixel {
                1) compute probabilities of each possible label;
                2) randomly assign new label based on the probabilities;
        }
}
```

Using Markov Chain Monte Carlo method

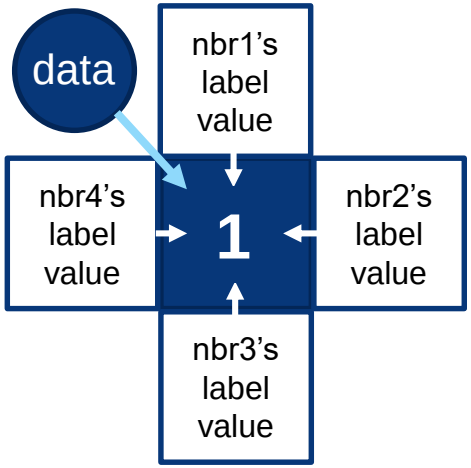


Left image

Right image

Pixel

Possible Matchings



Label dependency

Energy function:

$E(Data, Lables)$

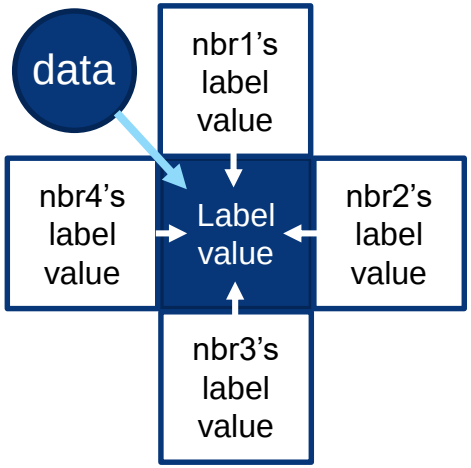
- Label: disparity on that pixel.

```
while(not converged) {
        for each pixel {
                1) compute probabilities of each possible label;
                2) randomly assign new label based on the probabilities;
        }
}
```

Using Markov Chain Monte Carlo method

Left image

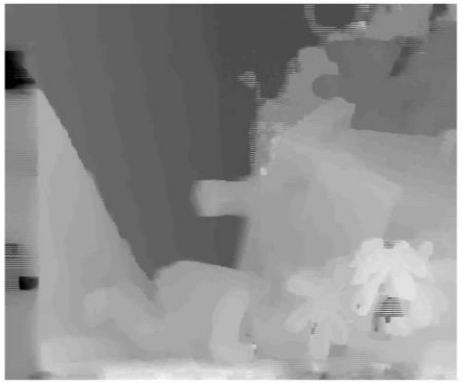
Right image



Label dependency

Energy function:

$E(Data, Lables)$



Disparity map
(lighter is closer)

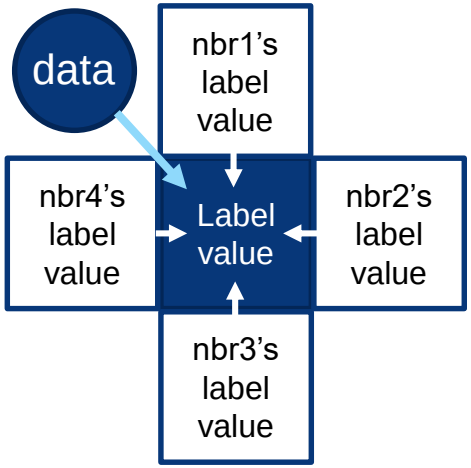
- Label: disparity on that pixel.

```
while(not converged) {
        for each pixel {
                1) compute probabilities of each possible label;
                2) randomly assign new label based on the probabilities;
        }
}
```

Using Markov Chain Monte Carlo method

Left image

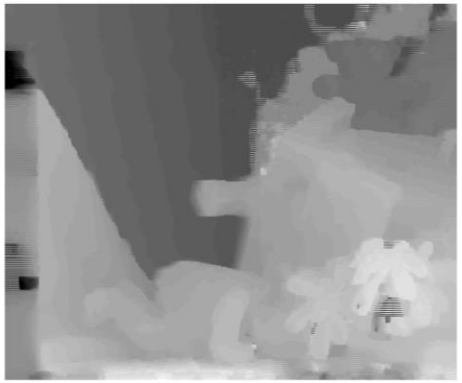
Right image



Label dependency

Energy function:

$E(Data, Lables)$



Disparity map
(lighter is closer)

- Label: disparity on that pixel.

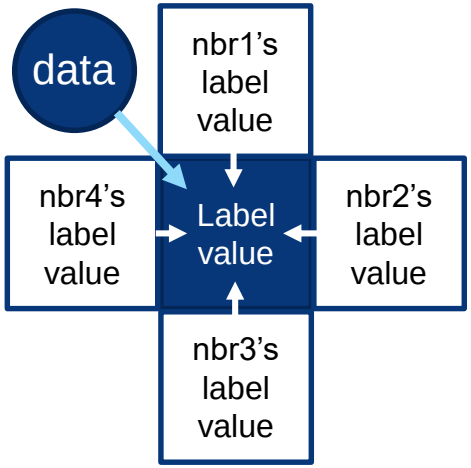
```
while(not converged) {
        for each pixel {
                1) compute probabilities over each possible label;
                2) randomly assign new label based on the probabilities;
        }
}
```

Slow!

Using Markov Chain Monte Carlo method

Left image

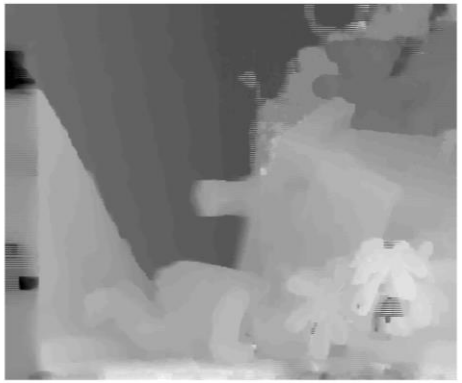
Right image



Label dependency

Energy function:

$E(Data, Lables)$



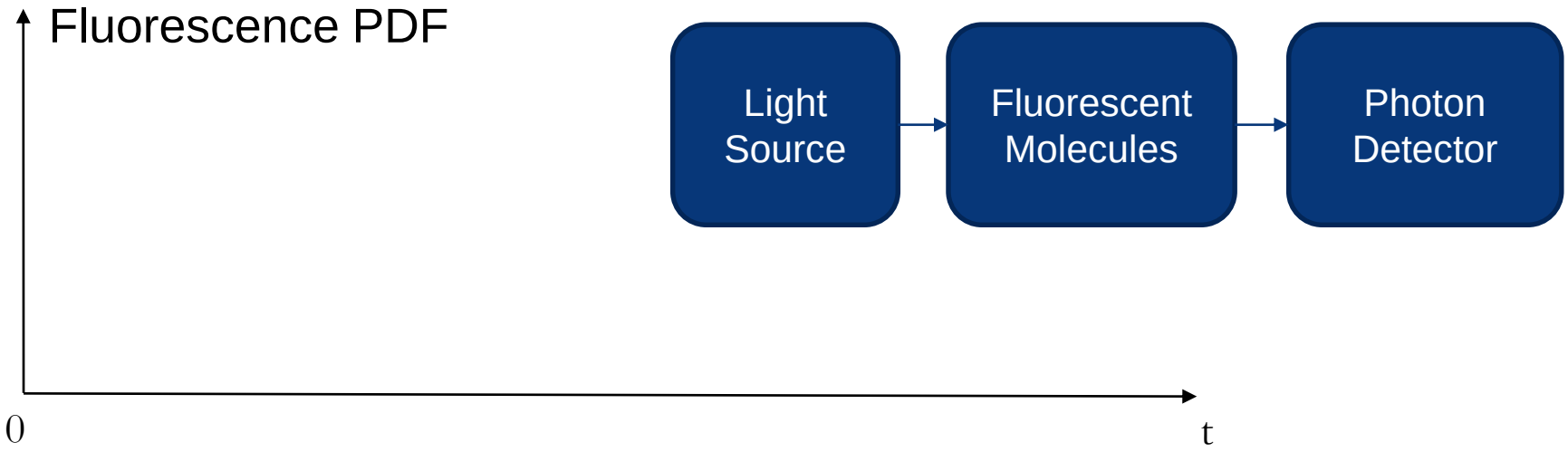
Disparity map
(lighter is closer)

- Label: disparity on that pixel.

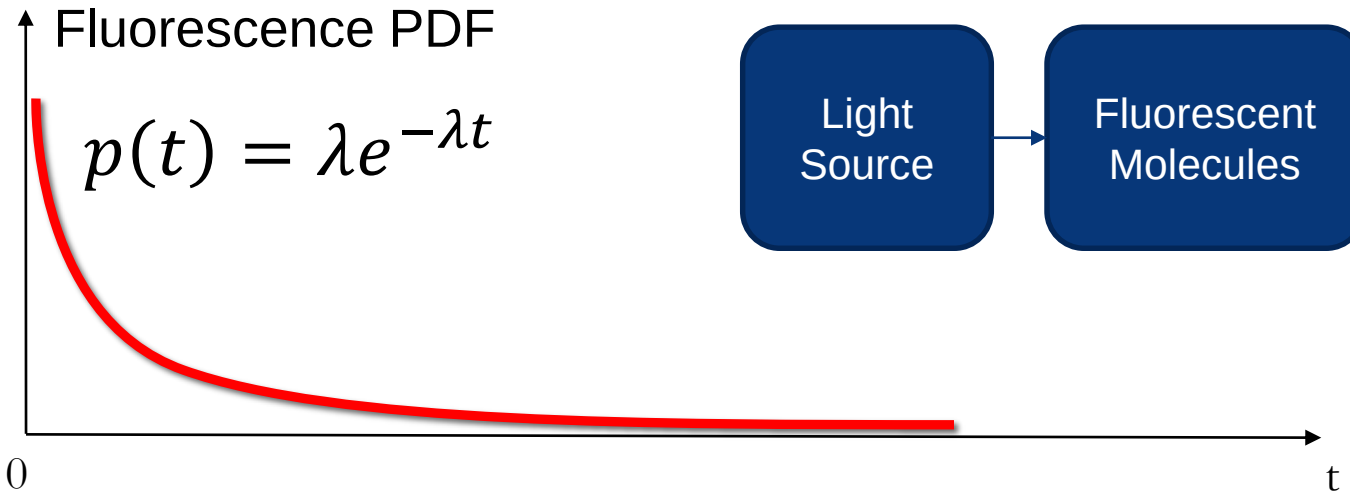
```
while(not converged) {
        for each pixel {
        }
}
```

Emerging Technology + Hardware specialization
[Wang et al., 2016]

Review: Sampling Using Molecules

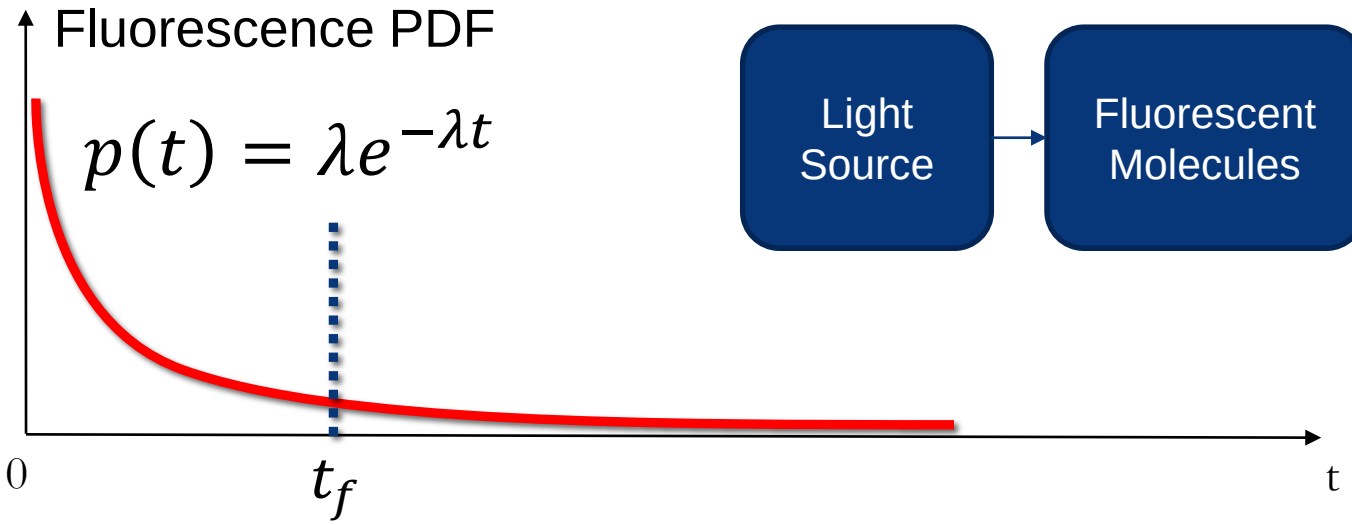


Review: Sampling Using Molecules



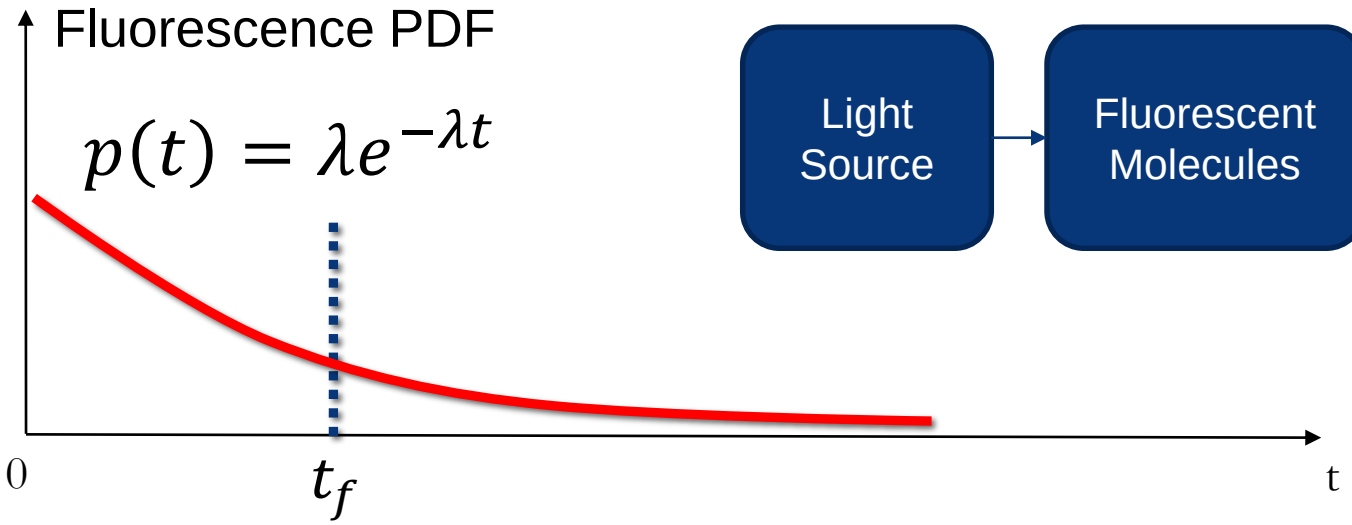
- Single fluorophore: exponential distribution.

Review: Sampling Using Molecules



- Single fluorophore: exponential distribution.
- Record Time to Fluorescence t_f.

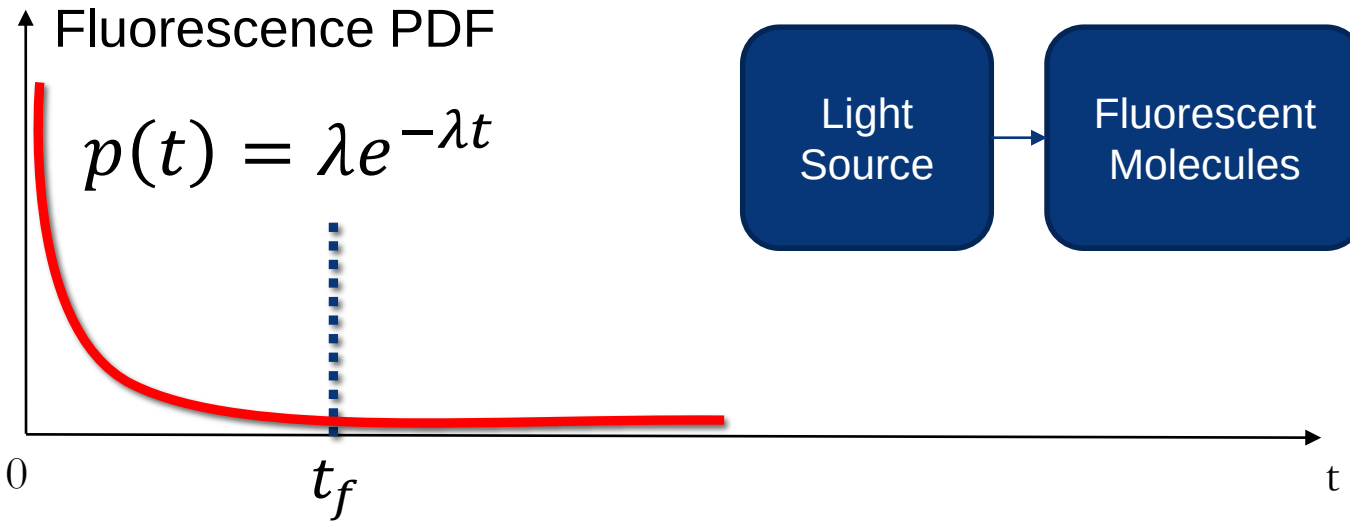
Review: Sampling Using Molecules



- Single fluorophore: exponential distribution.
- Record Time to Fluorescence t_f.
- Parameterize distributions by decay rate (λ):

$$\lambda \propto \frac{1}{2}\,\text{intensity} \times \text{concentration}$$

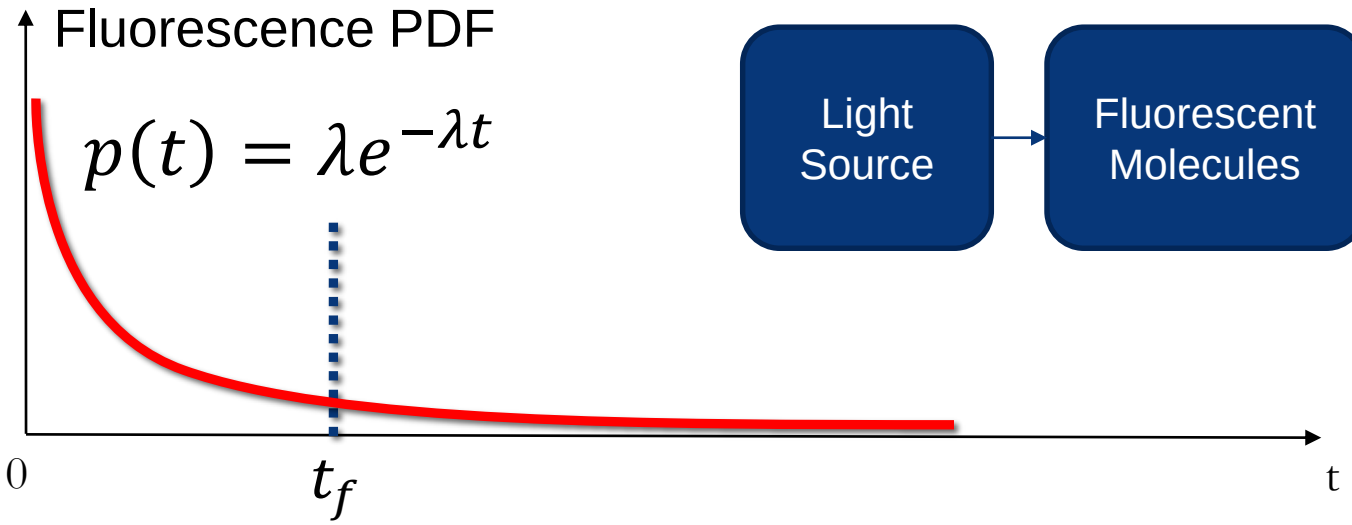
Review: Sampling Using Molecules



- Single fluorophore: exponential distribution.
- Record Time to Fluorescence t_f.
- Parameterize distributions by decay rate (λ):

$$\lambda \propto \text{intensity} \times 2\,\text{concentration}$$

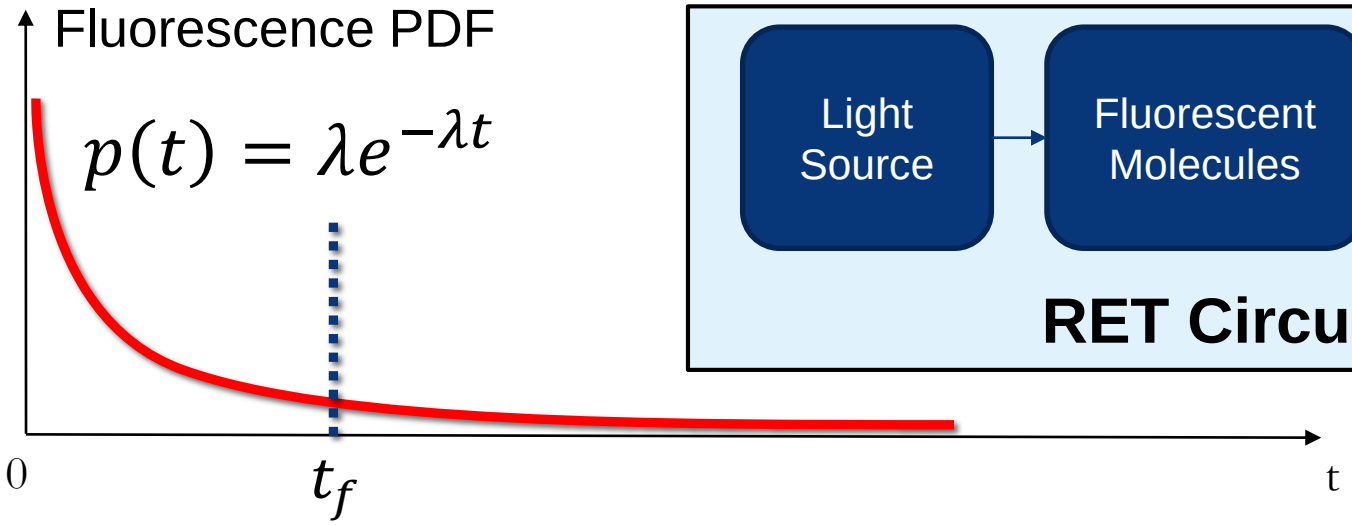
Review: Sampling Using Molecules



- Single fluorophore: exponential distribution.
- Record Time to Fluorescence t_f.
- Parameterize distributions by decay rate (λ):

$$\lambda \propto \text{intensity} \times \text{concentration}$$

Review: Sampling Using Molecules

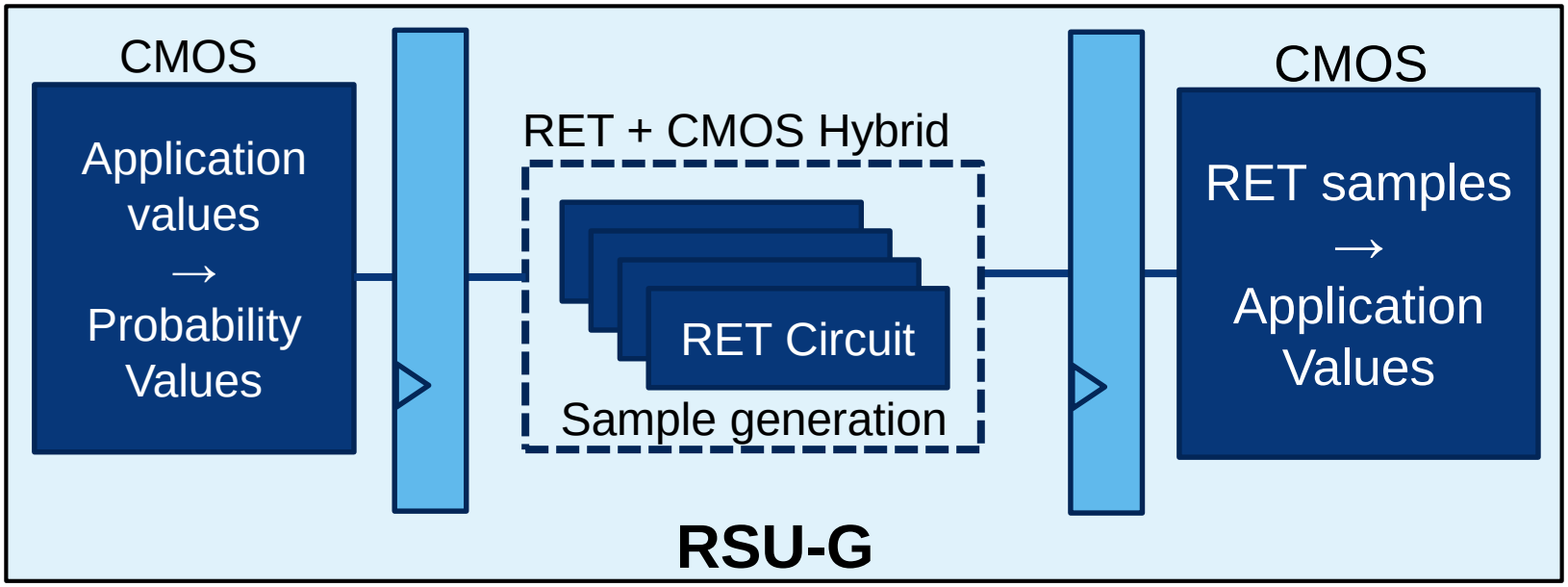


- Single fluorophore: exponential distribution.
- Record Time to Fluorescence t_f.
- Parameterize distributions by decay rate (λ):

$$\lambda \propto \text{intensity} \times \text{concentration}$$

- Implemented in Resonance Energy Transfer (RET) circuit.

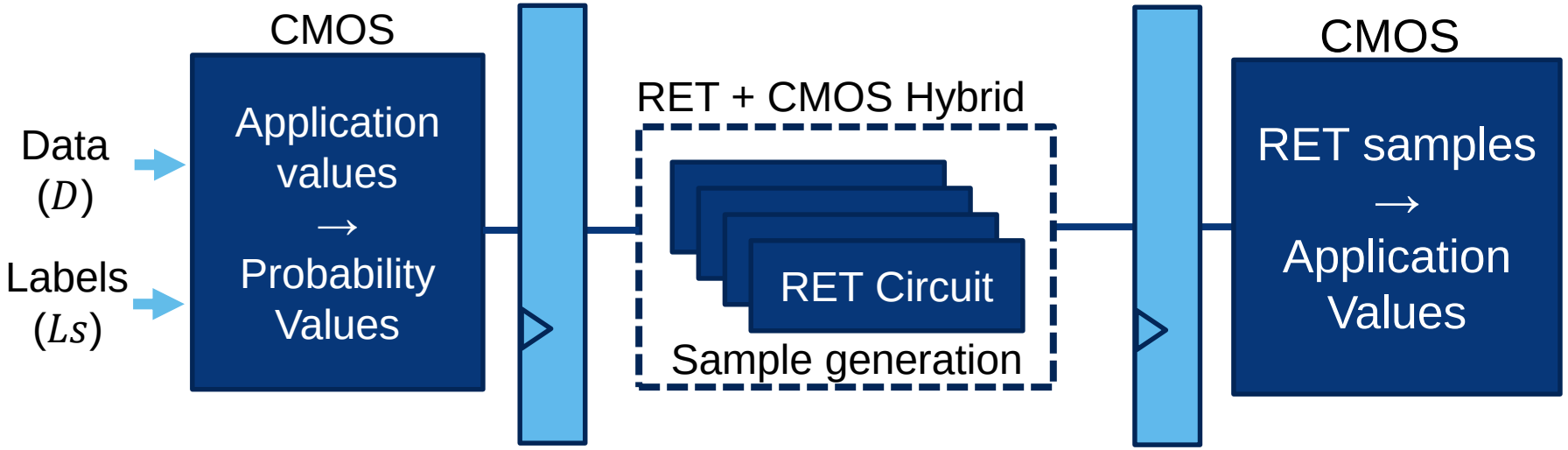
Review: RET-based Gibbs Sampling Unit (RSU-G)



- Accelerates inner loop computation:

1) compute probabilities of each possible label;
2) randomly assign new label based on the probabilities;

Review: RET-based Gibbs Sampling Unit (RSU-G)

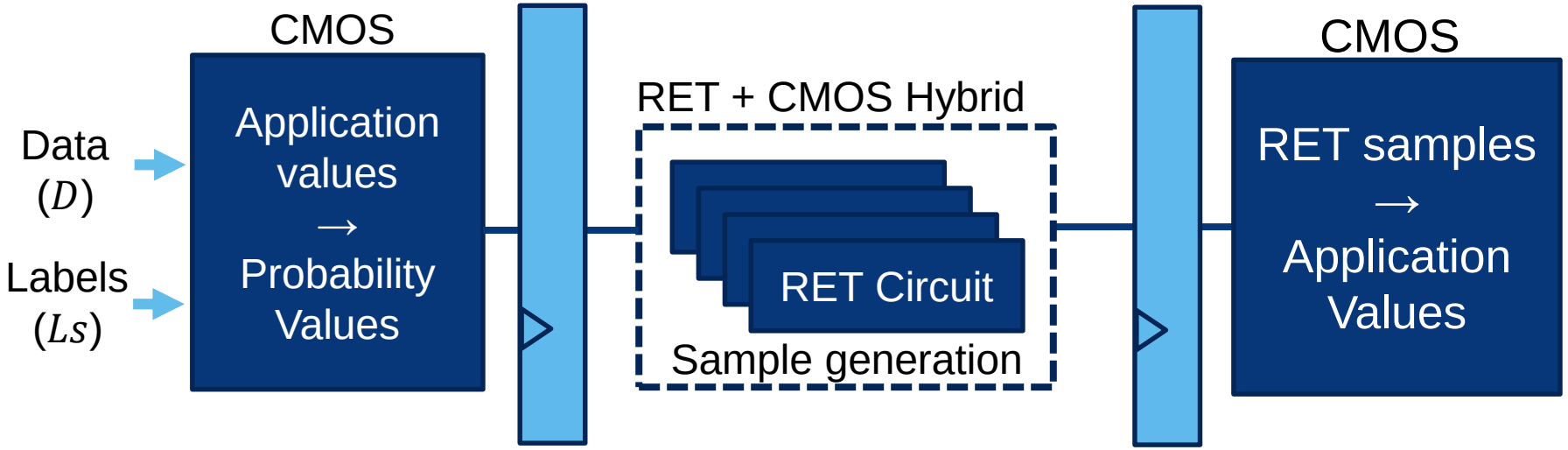


$$\lambda = \exp(-E(D, L))$$

- Accelerates inner loop computation:

1) compute probabilities of each possible label;
2) randomly assign new label based on the probabilities;

Review: RET-based Gibbs Sampling Unit (RSU-G)



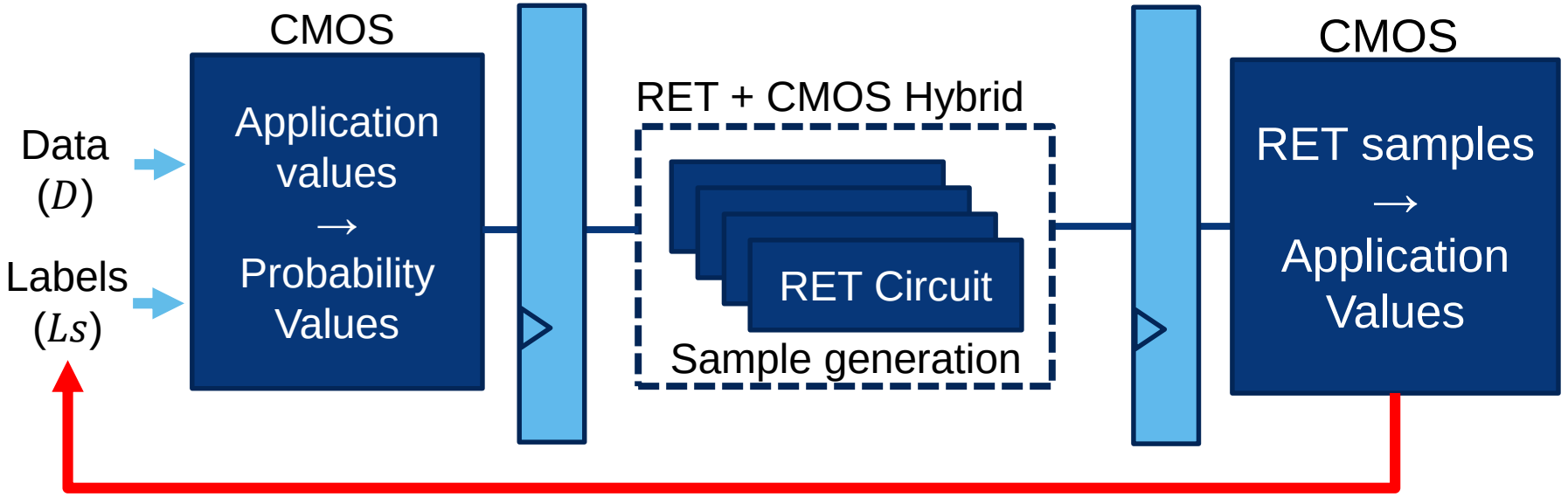
$$\lambda = \exp(-E(D,L)) \qquad p(t) = \lambda\exp(-\lambda t) \qquad L = argmin(t_1, t_2, \dots)$$

"First-to-fire" [Wang et al., 2015]

- Accelerates inner loop computation:

1) compute probabilities of each possible label;
2) randomly assign new label based on the probabilities;

Review: RET-based Gibbs Sampling Unit (RSU-G)



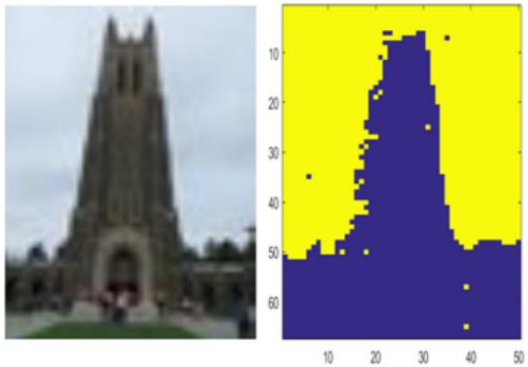
$\lambda = \exp(-E(D,L))$ $p(t) = \lambda\exp(-\lambda t)$ $L = argmin(t_1, t_2, \dots)$

"First-to-fire" [Wang et al., 2015]

- Accelerates inner loop computation:

1) compute probabilities of each possible label;
2) randomly assign new label based on the probabilities;

Review: RET-based Gibbs Sampling Unit (RSU-G)



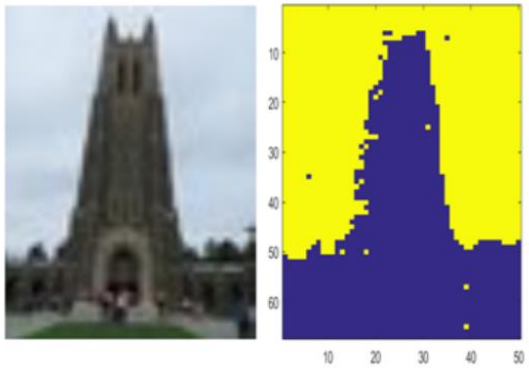
Original Prototype result

Image Segmentation

[Wang et al., 2016]

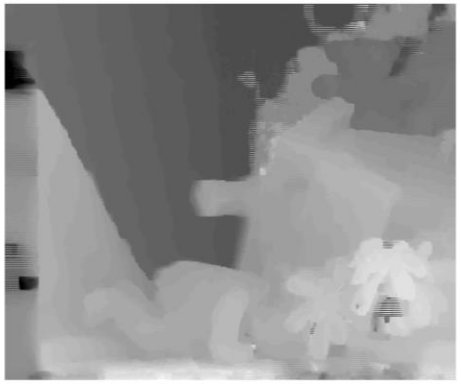
- Speedups over GPU:
  - Augmented GPU 3-34x
  - Discrete Accelerator 21-84x
- First experimental demonstration of RSU-G.

Review: RET-based Gibbs Sampling Unit (RSU-G)

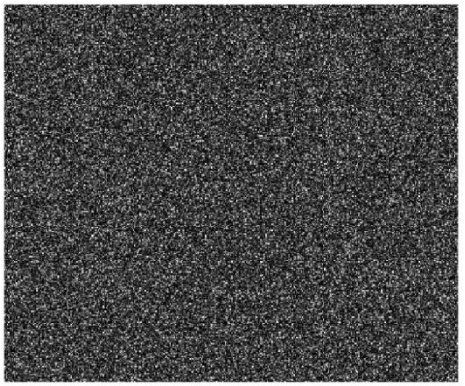


Original Prototype result

Image Segmentation
[Wang et al., 2016]



Software disparity map

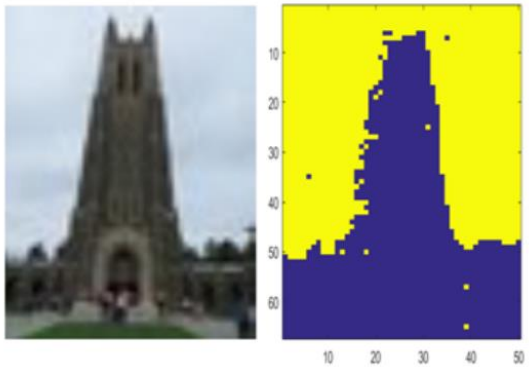


RSU-G disparity map

Stereo vision *teddy* dataset

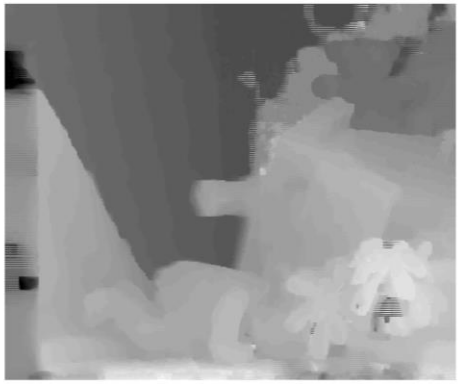
- Speedups over GPU:
  - Augmented GPU 3-34x
  - Discrete Accelerator 21-84x
- First experimental demonstration of RSU-G.
- Result quality: doesn't reach the level we need!

Review: RET-based Gibbs Sampling Unit (RSU-G)

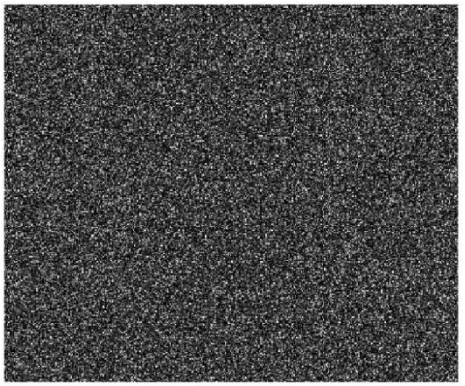


Original Prototype result

Image Segmentation
[Wang et al., 2016]



Software disparity map



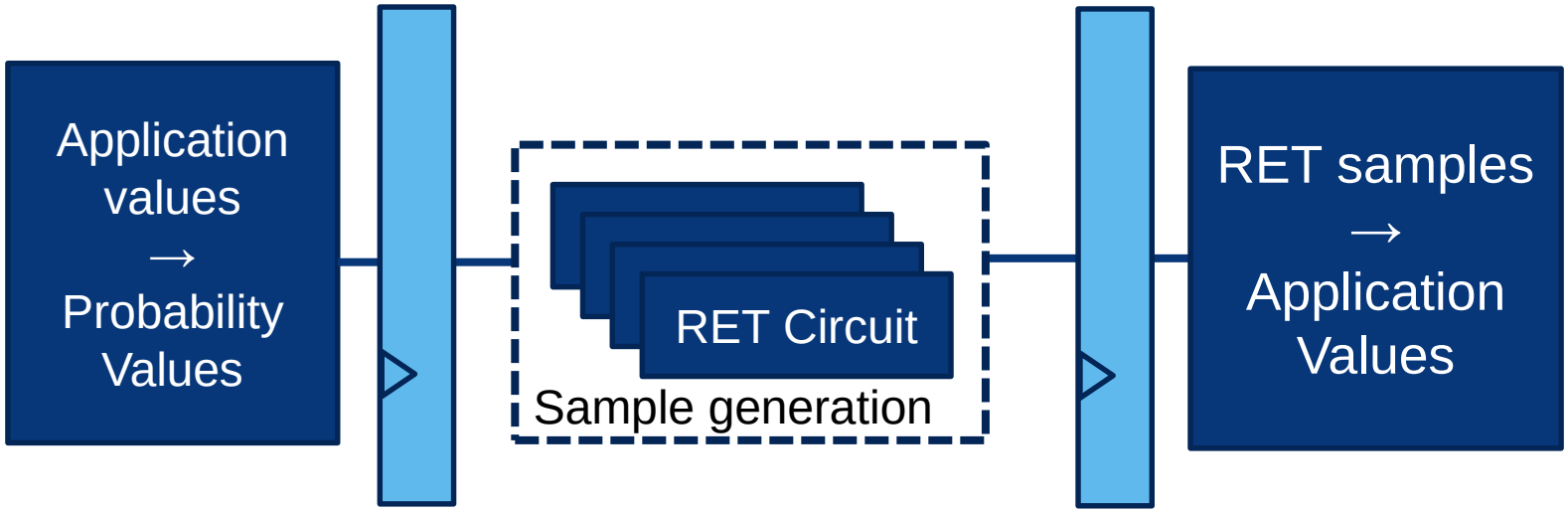
RSU-G disparity map

Stereo vision *teddy* dataset

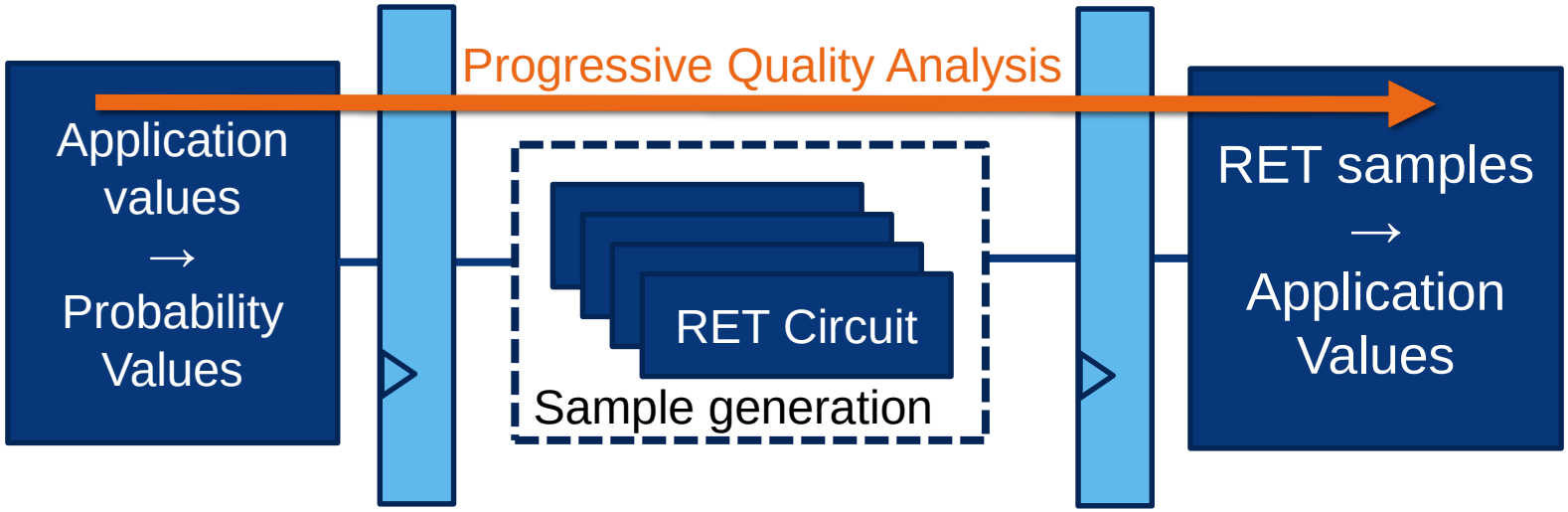
- Speedups over GPU:
  - Augmented GPU 3-34x
  - Discrete Accelerator 21-84x
- First experimental demonstration of RSU-G.
- Result quality: doesn't reach the level we nee

How to preserve result quality?

How to do this? Naïve Approach

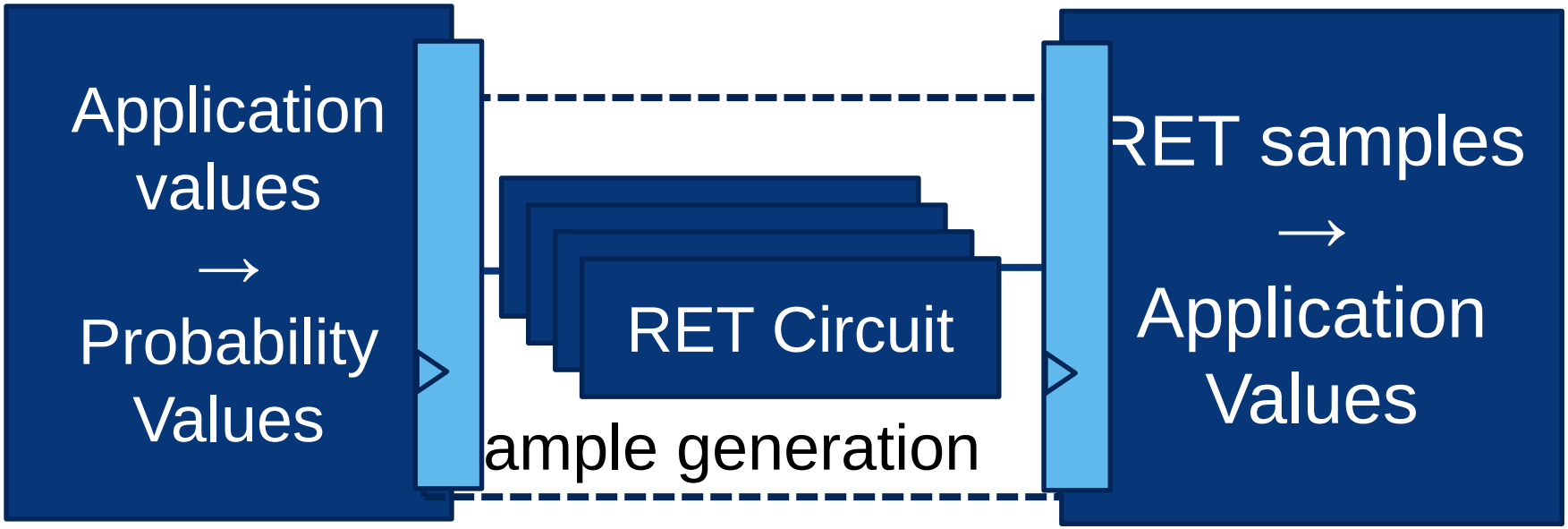


How to do this? Naïve Approach



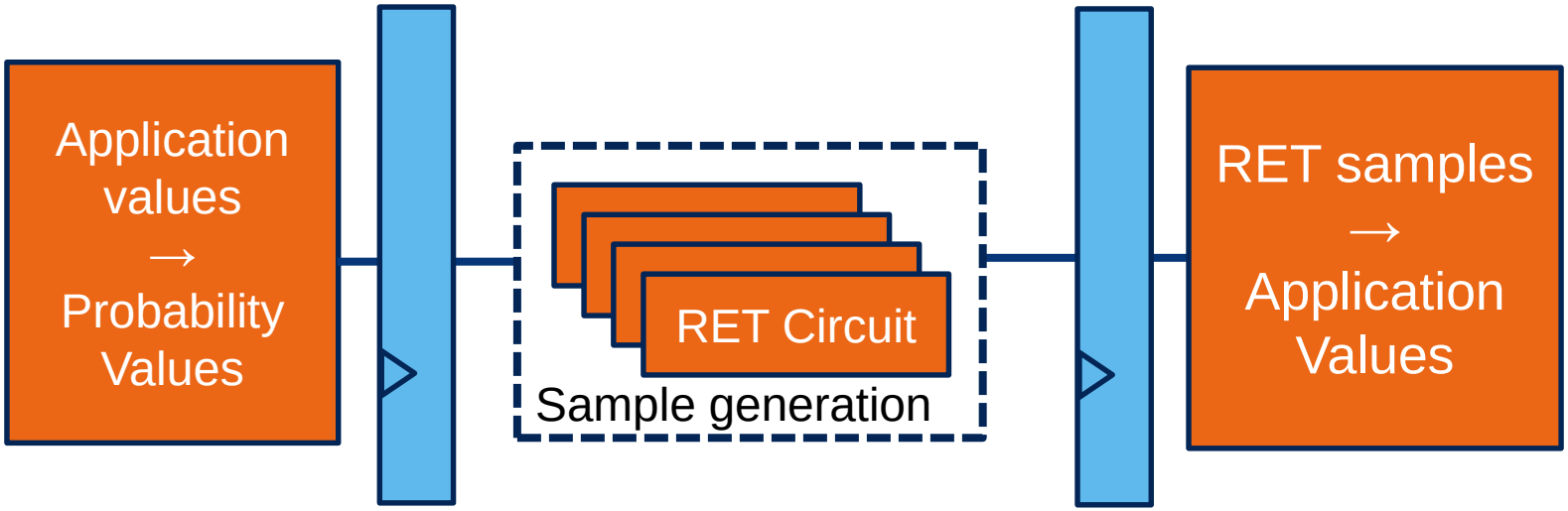
- Lacks both precision and dynamic range.

How to do this? Naïve Approach



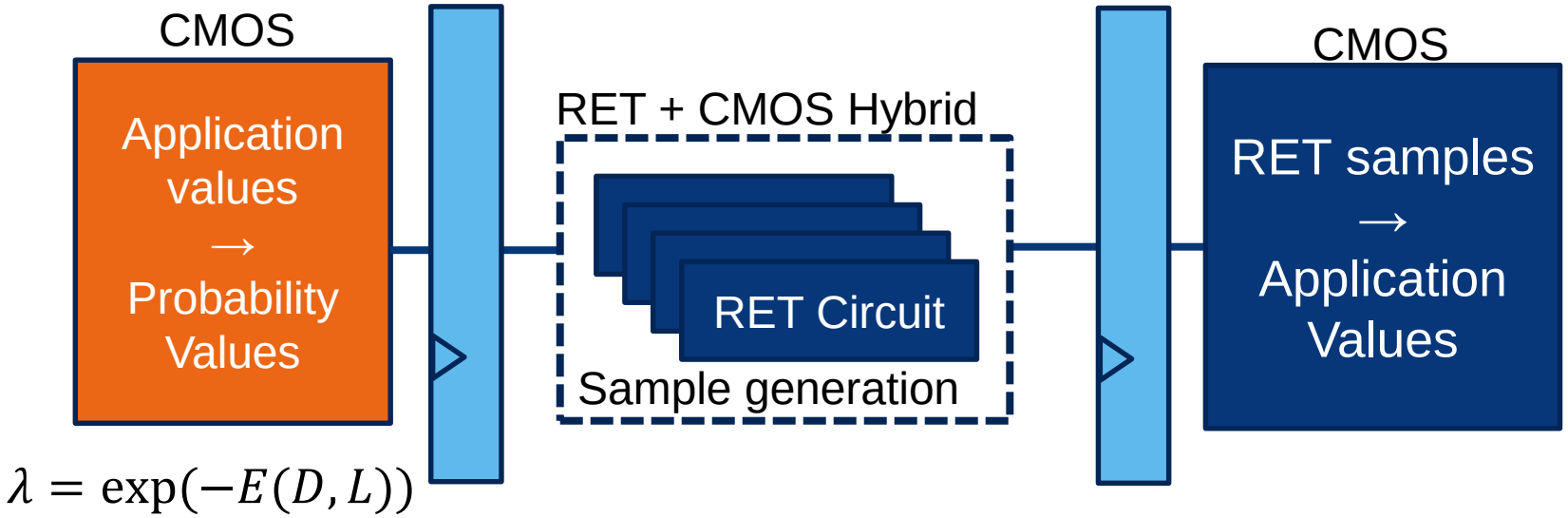
- Lacks both precision and dynamic range.
- Naively increase #bits: too much area/power.

How to do this? Naïve Approach



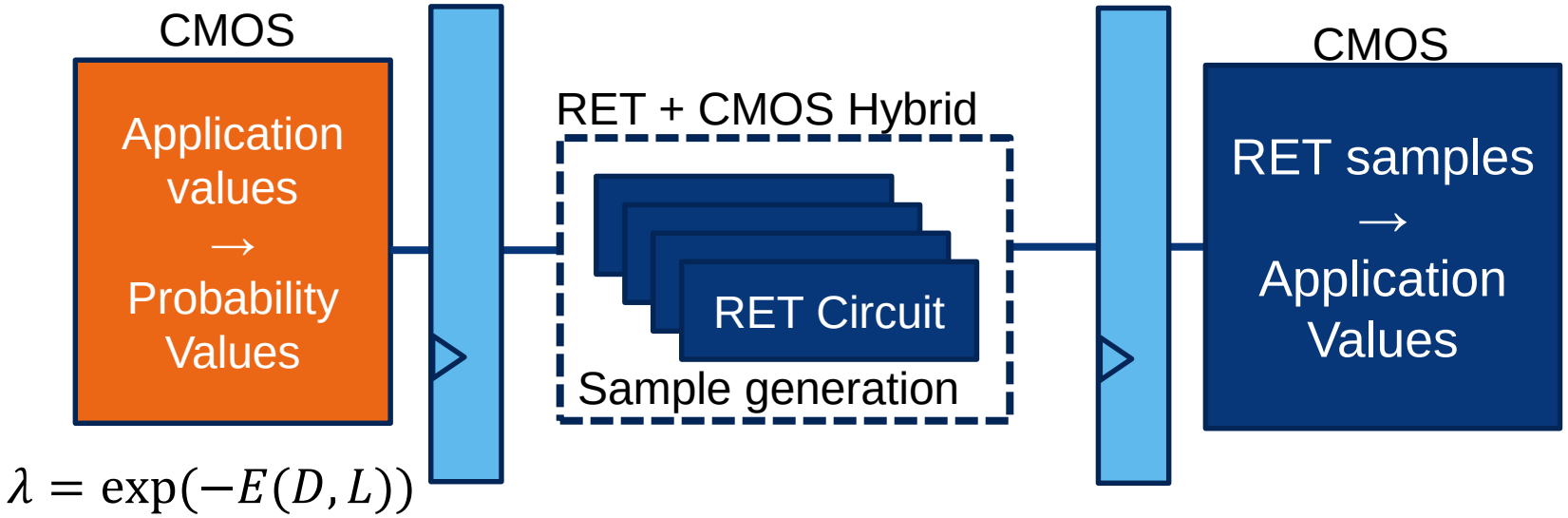
- Lacks both precision and dynamic range.
- Naively increase #bits: too much area/power.
- Need new microarchitecture to achieve:
  - **High** result quality.
  - **Sizable** performance benefits.
  - **Minimal** area/power.
  - **More** flexibility.

Improving Decay Rate (λ) Dynamic Ranges



- New microarchitectural techniques:
  - Decay rate scaling.
  - Probability cut-off.
  - 2^n approximation.
- Requires only 4 unique decay rates.
- Detail described in the paper.

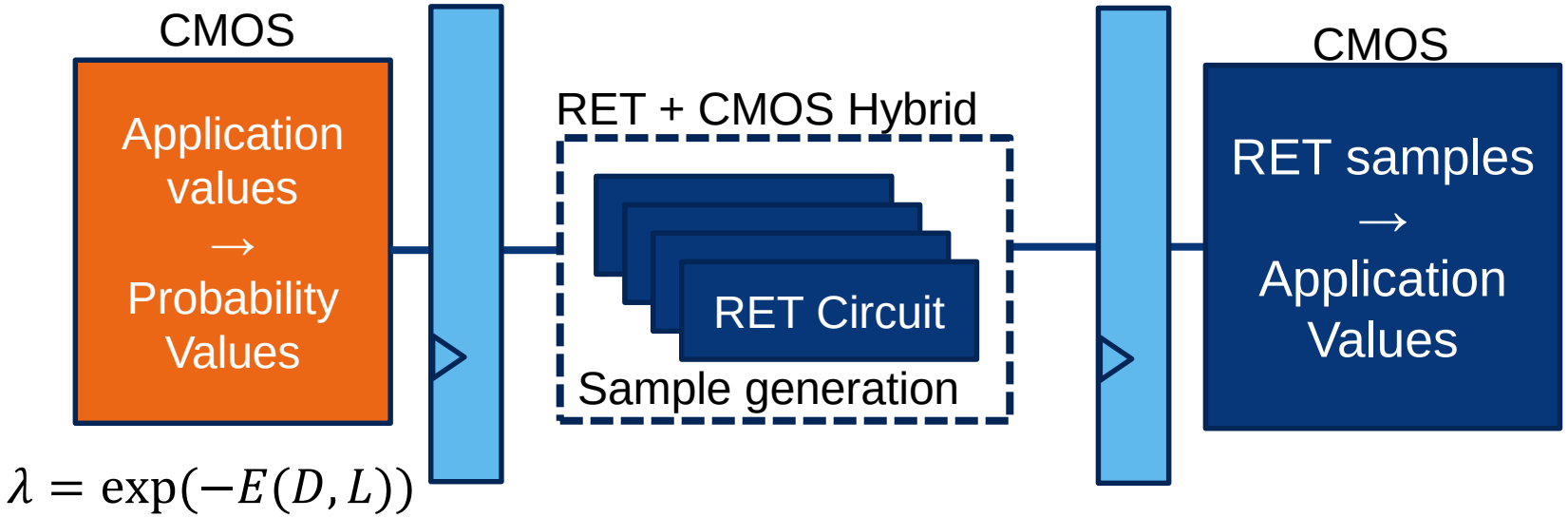
Improving Decay Rate (λ) Dynamic Ranges



- New microarchitectural techniques:
  - Decay rate scaling.
  - Probability cut-off.
  - 2^n approximation.

  } **High** result quality.
- Requires only 4 unique decay rates.
- Detail described in the paper.

Improving Decay Rate (λ) Dynamic Ranges

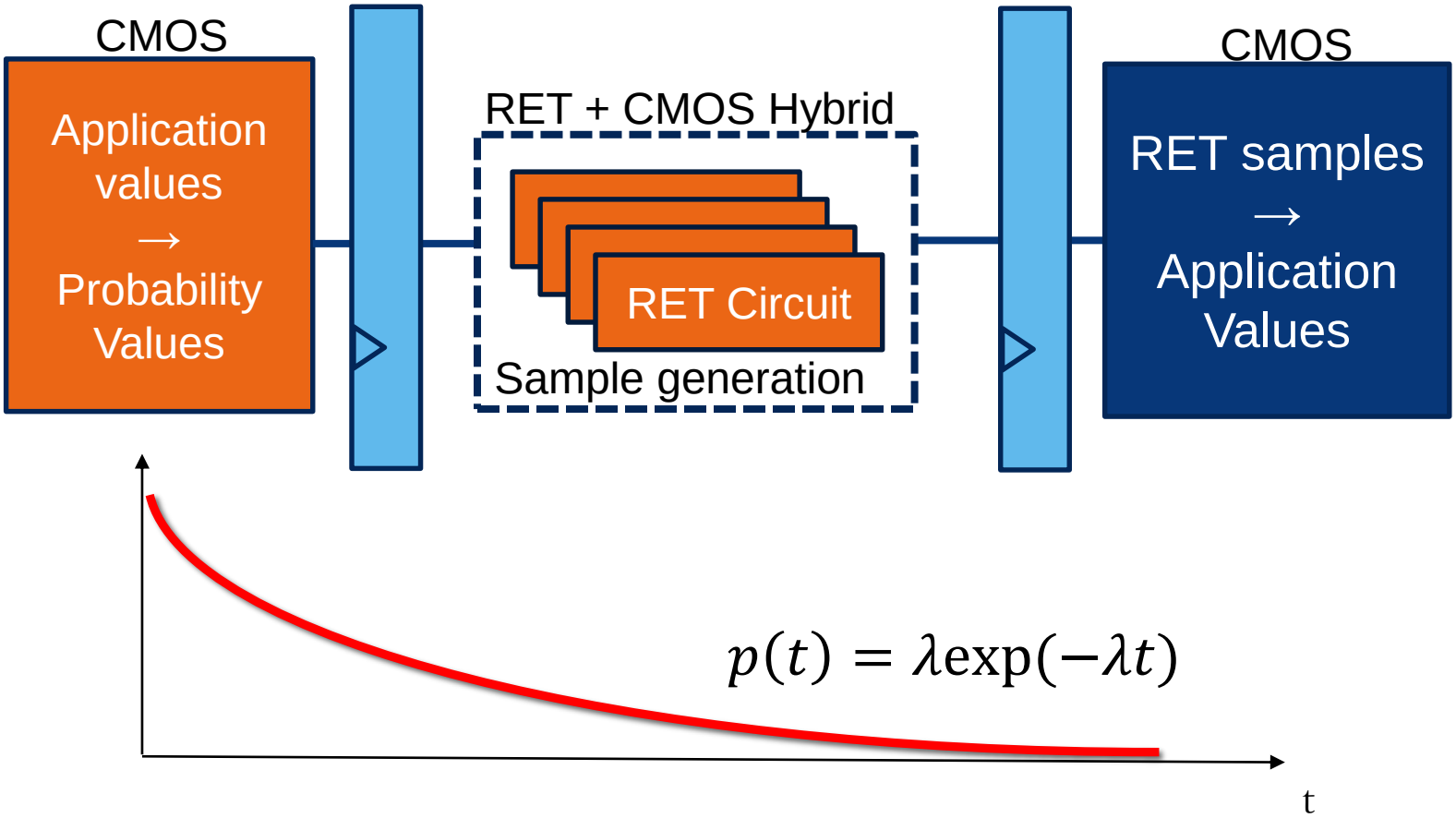


- New microarchitectural techniques:
  - Decay rate scaling.
  - Probability cut-off.
  - 2^n approximation.

  } **High** result quality.

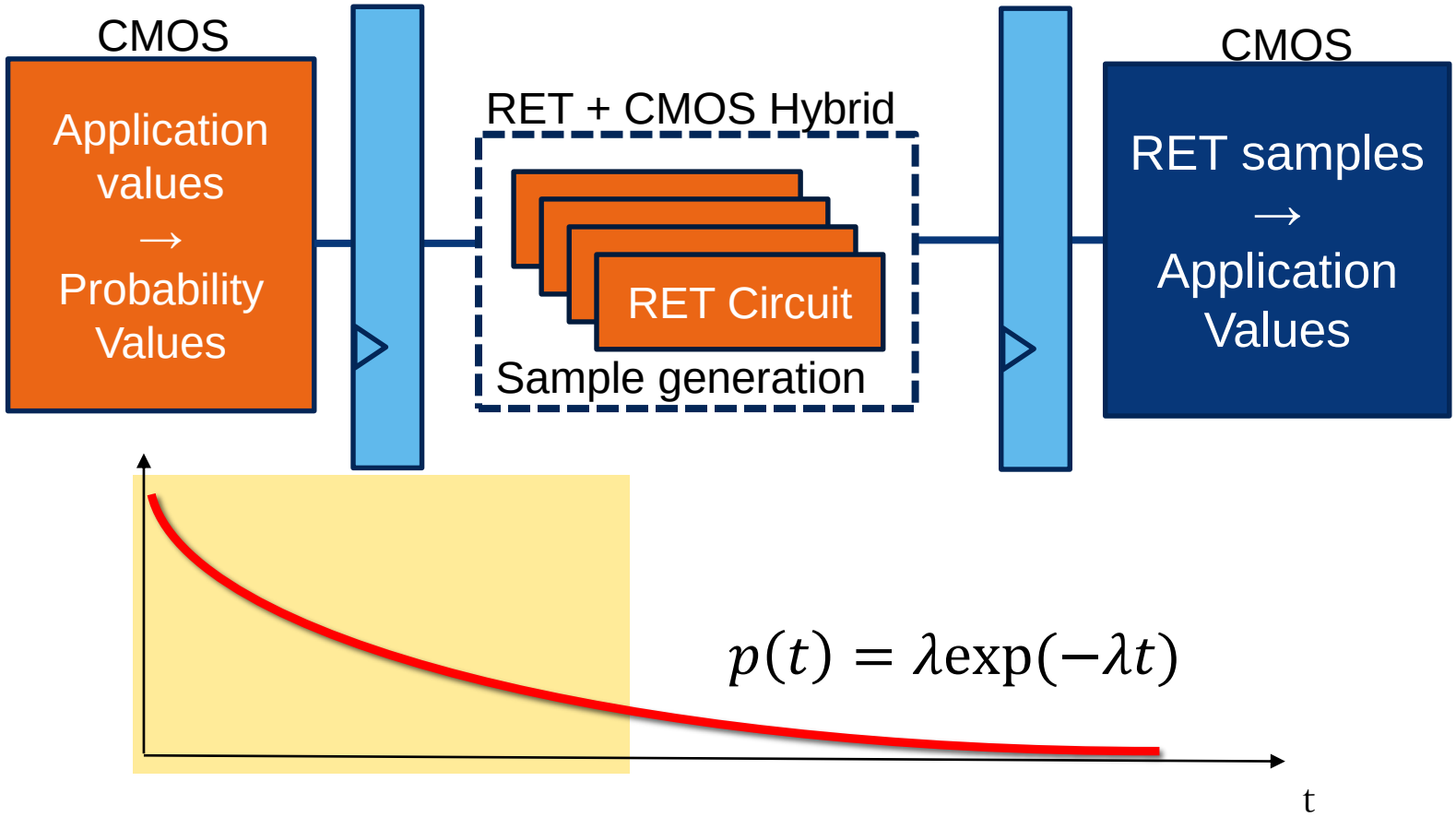
  — **Minimal** area/power.
- Requires only 4 unique decay rates.
- Detail described in the paper.

Exploring Sample Generation



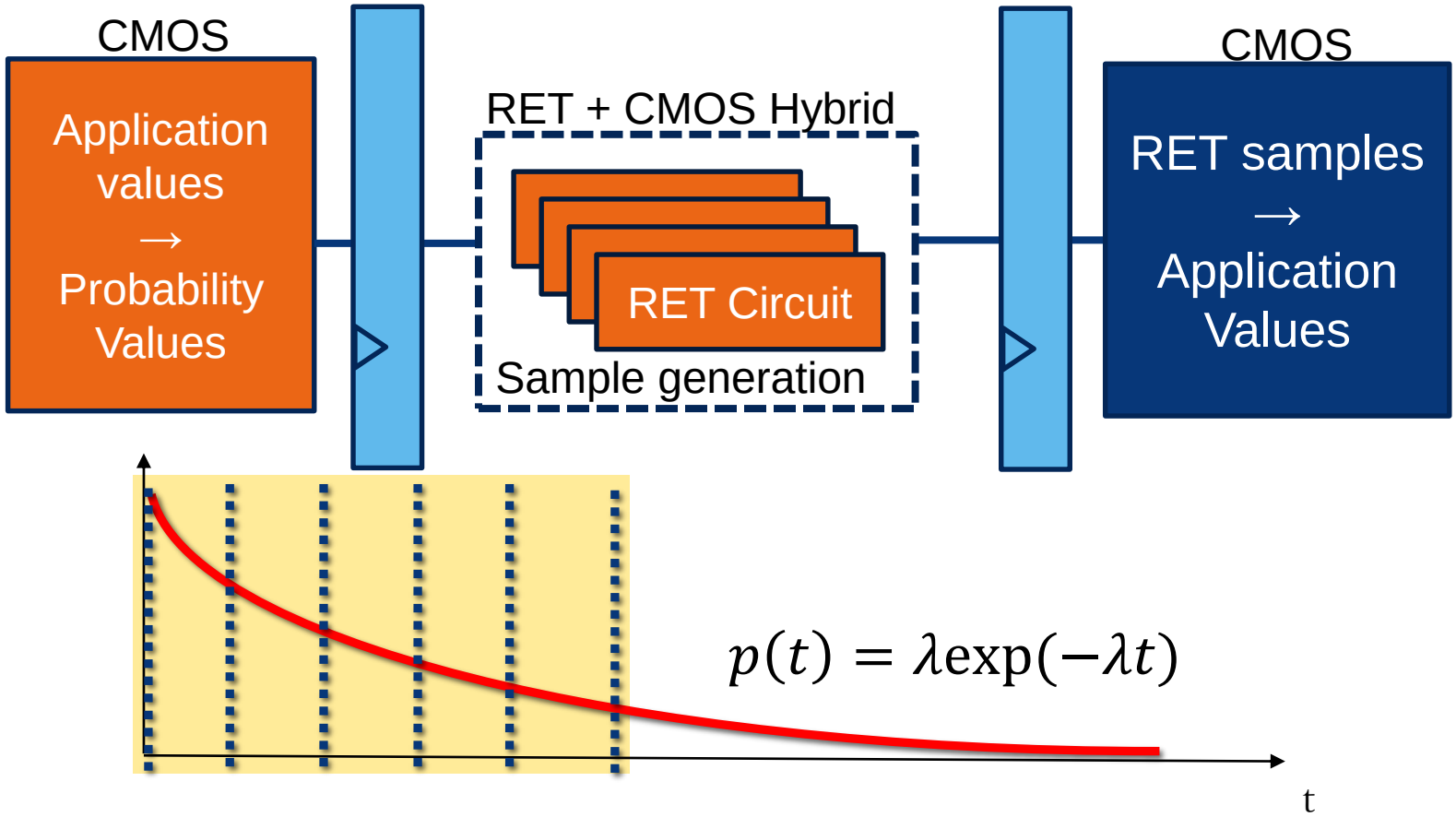
- Measuring Time to Fluorescence (TTF).
- Two Parameters:
  - Truncation probability.
  - Timing resolution.

Exploring Sample Generation



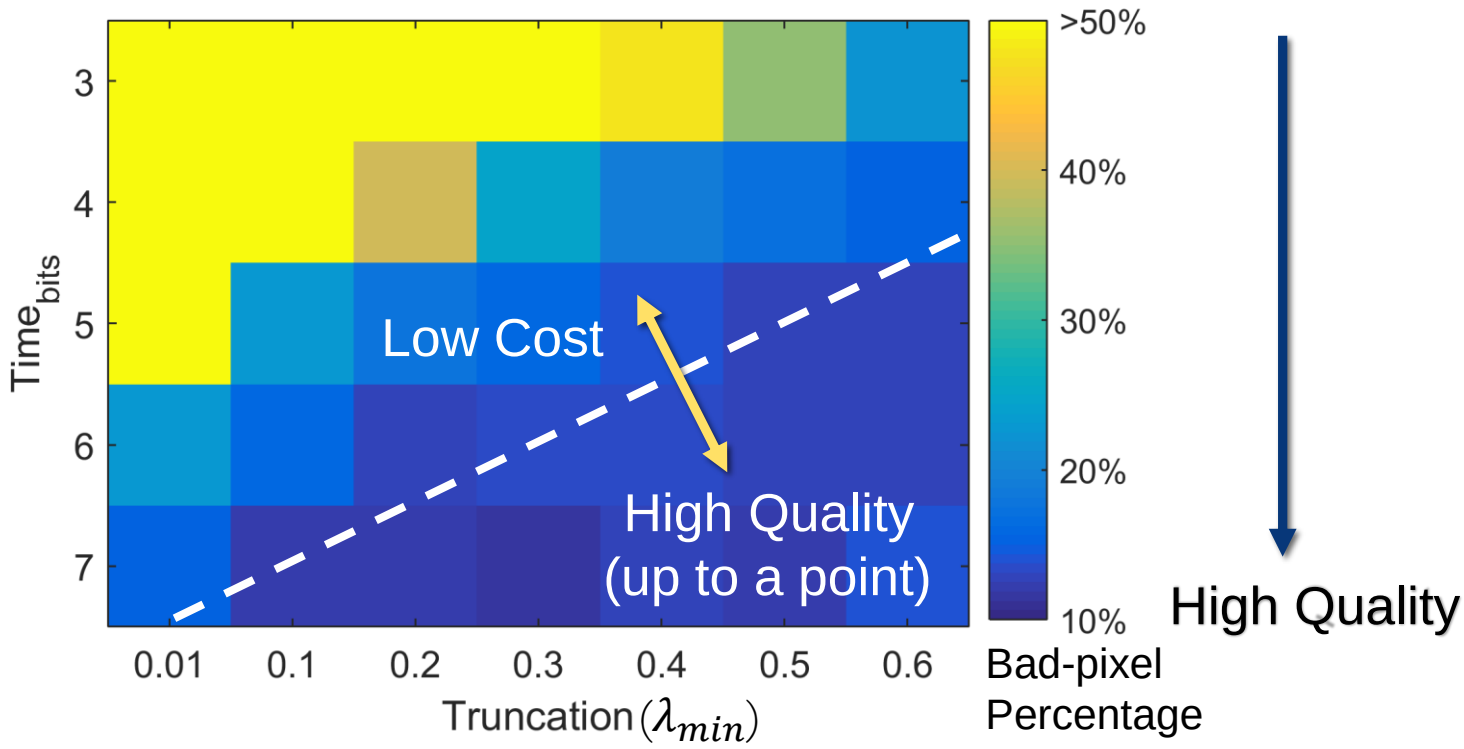
- Measuring Time to Fluorescence (TTF).
- Two Parameters:
  - Truncation probability.
  - Timing resolution.

Exploring Sample Generation



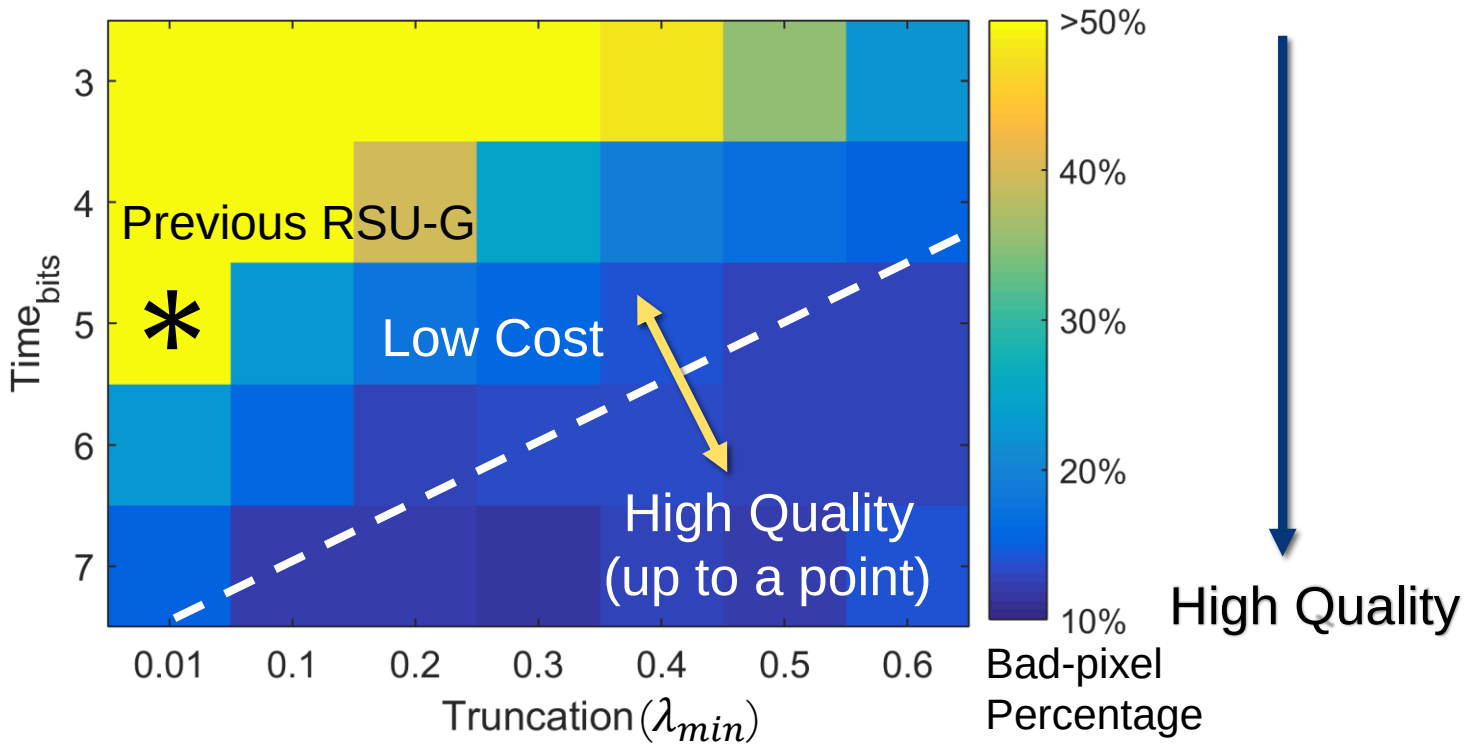
- Measuring Time to Fluorescence (TTF).
- Two Parameters:
  - Truncation probability.
  - Timing resolution.

Timing Resolution vs. Truncation Probability



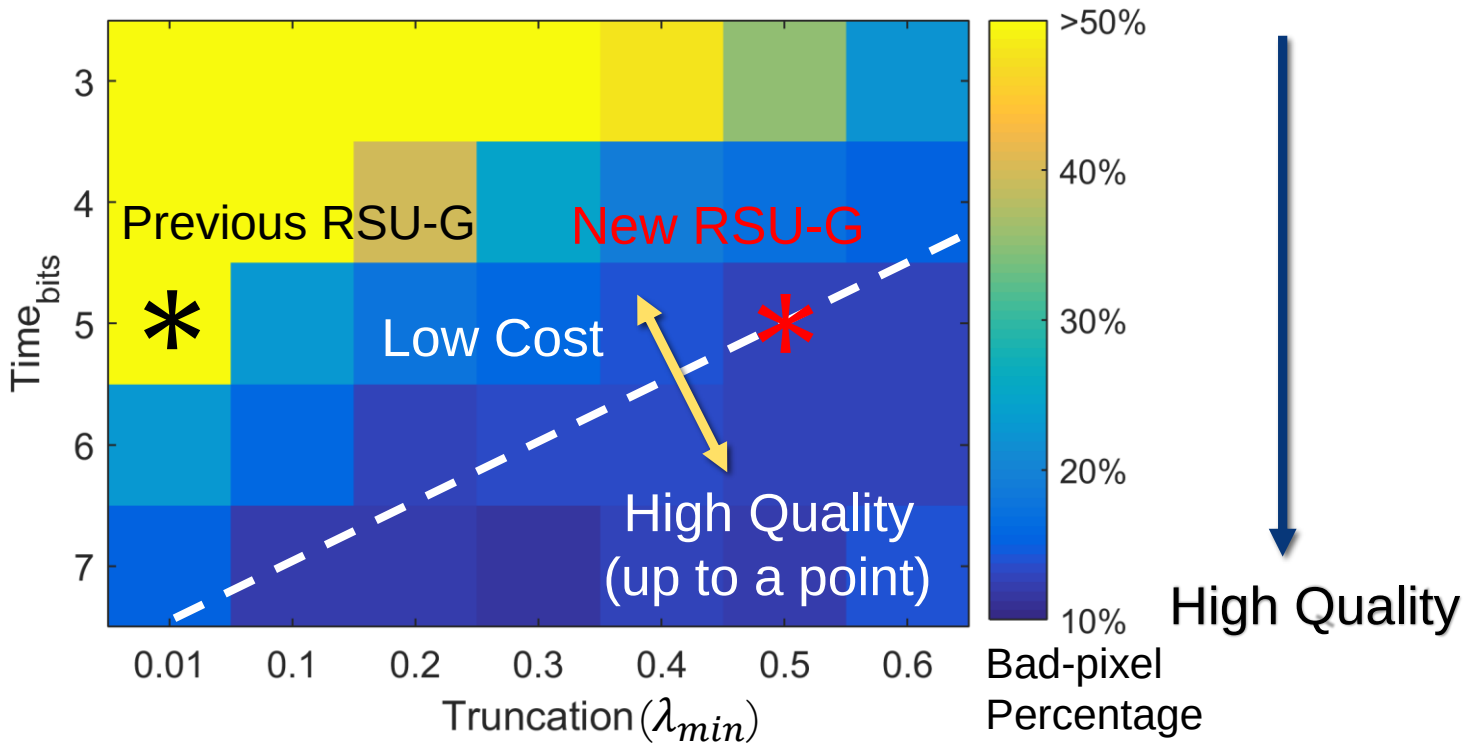
- Dashed line: equal result quality.
- Pr $(t \geq t_{max})$ = 0.5 for λ_{min} (< 0.5 for other λs).

Timing Resolution vs. Truncation Probability



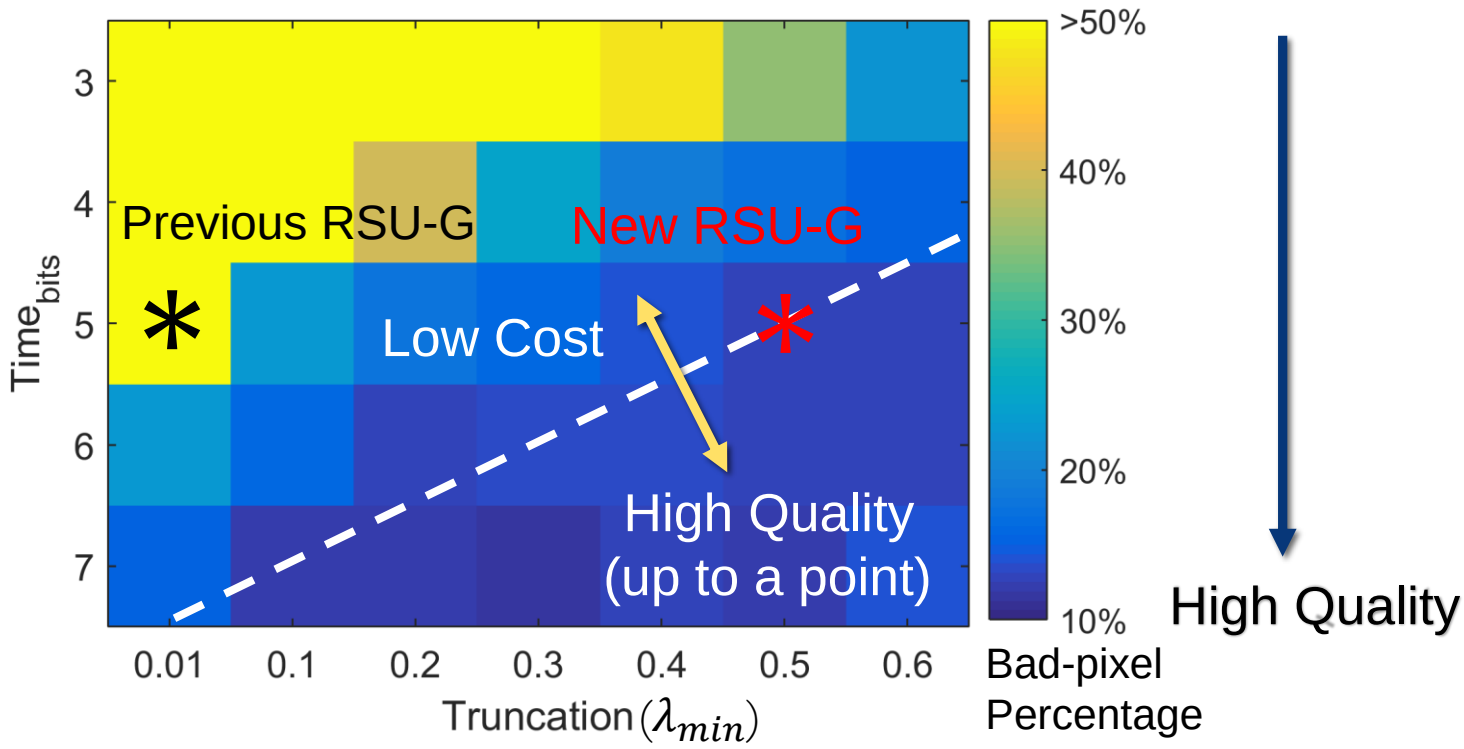
- Dashed line: equal result quality.
- Pr $(t \geq t_{max})$ = 0.5 for λ_{min} (< 0.5 for other λs).

Timing Resolution vs. Truncation Probability



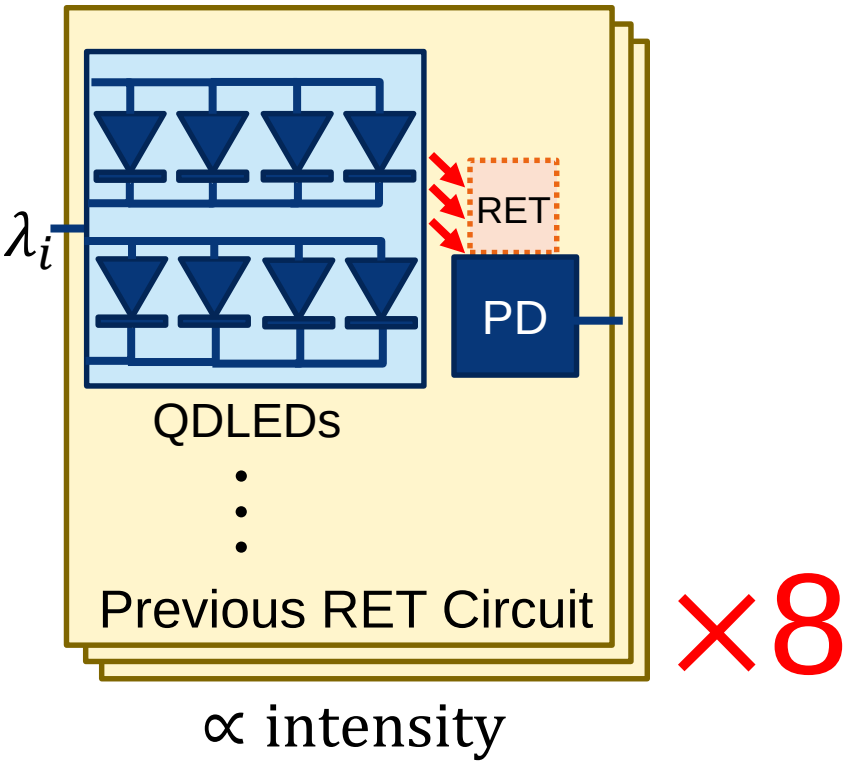
- Dashed line: equal result quality.
- Pr ($t \geq t_{max}$) = 0.5 for λ_{min} (< 0.5 for other λs).

Timing Resolution vs. Truncation Probability



- Dashed line: equal result quality.
- Pr ($t \geq t_{max}$) = 0.5 for λ_{min} (< 0.5 for other λs).
- Fluorophores can emit photons beyond max detection time.
- Need 8 RET replica to avoid structural hazard.

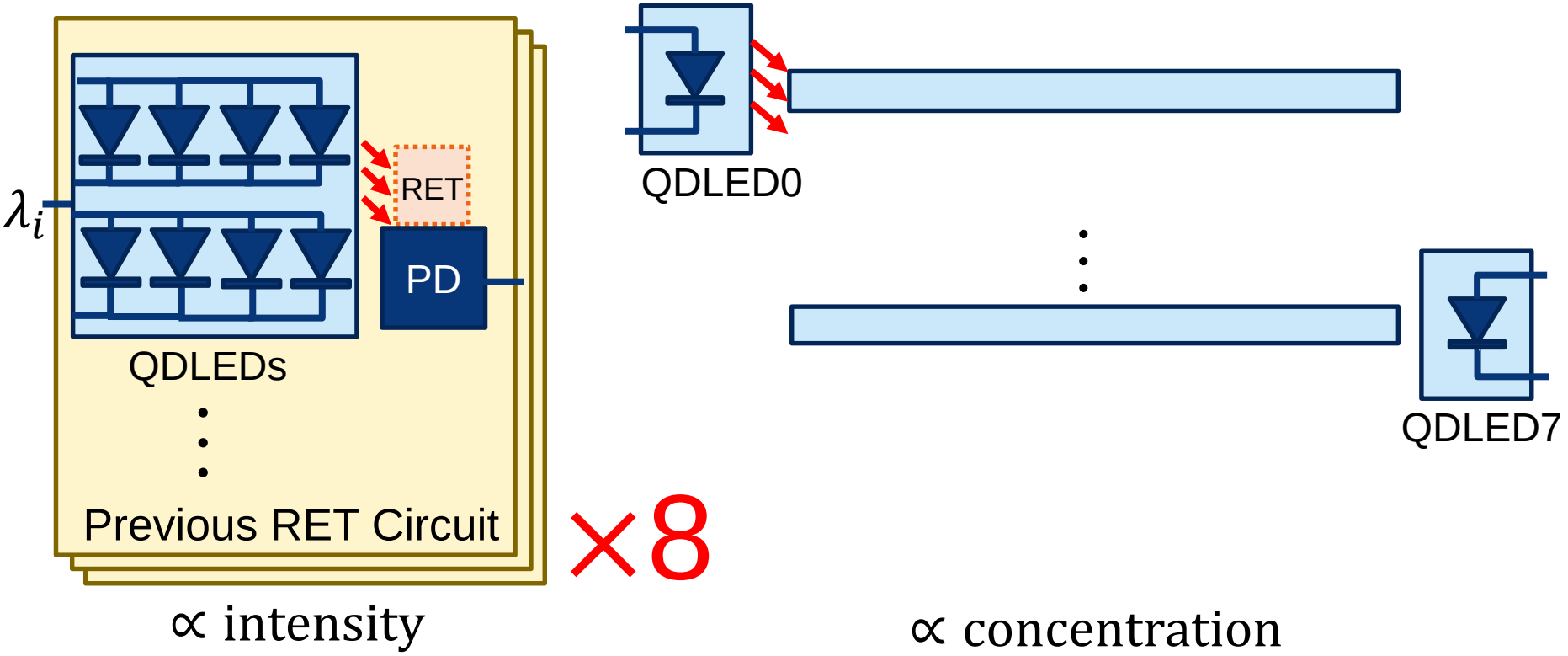
Re-design RET Circuit



×8

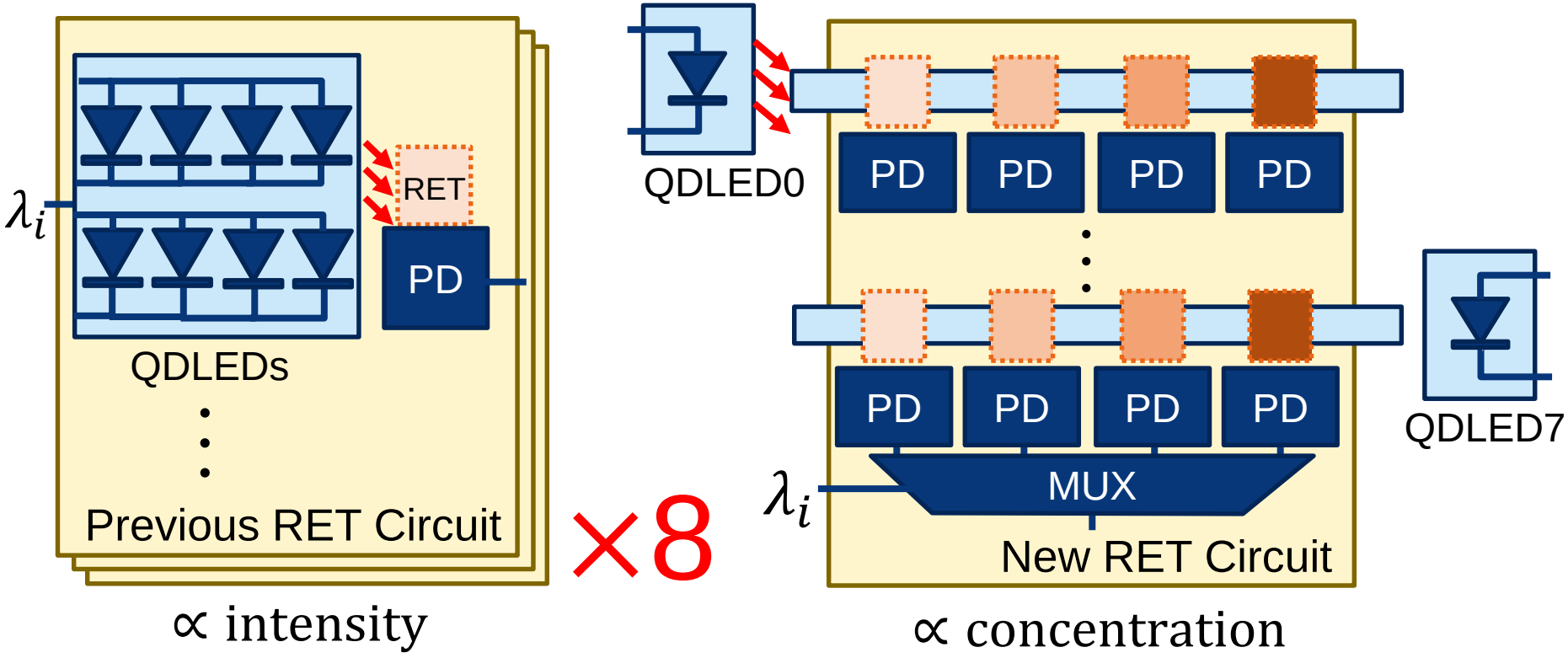
- Previous RET Circuit: QDLEDs dominates the area.

Re-design RET Circuit



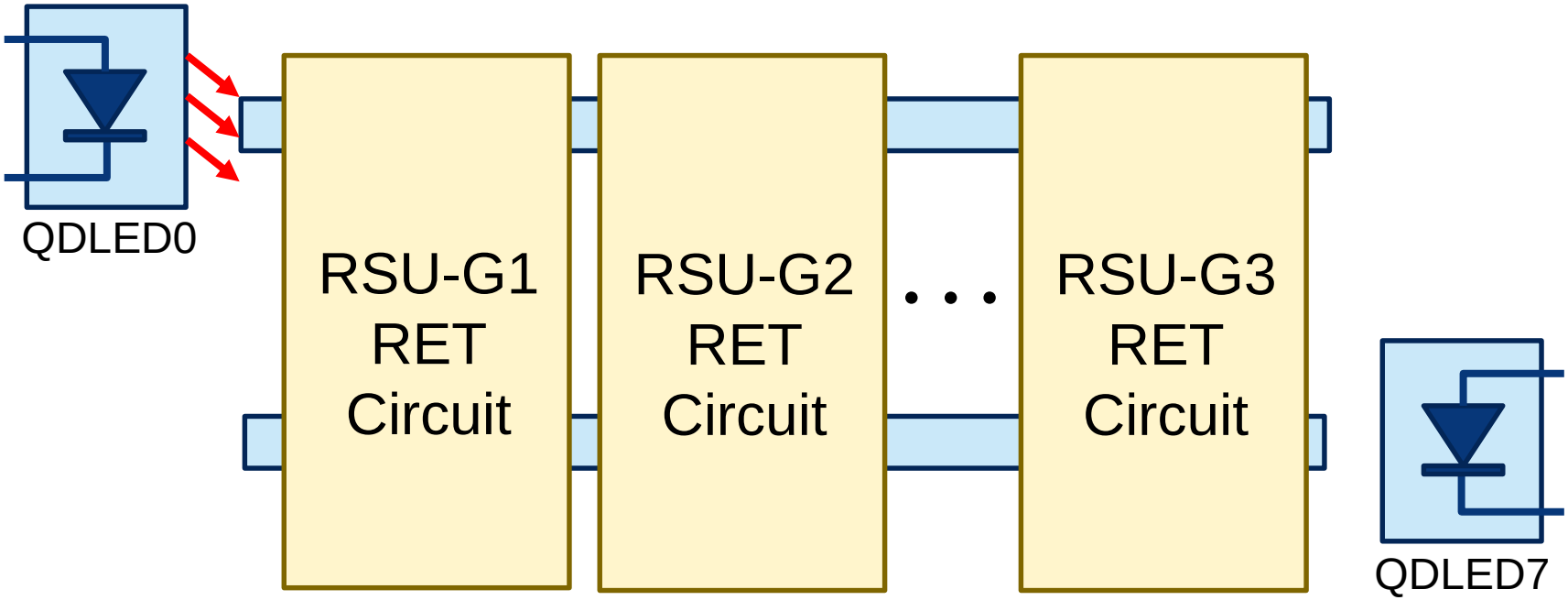
- Previous RET Circuit: QDLEDs dominates the area.
- New RET Circuit:
  - Fixed intensity, 8 waveguides lighted in round robin.

Re-design RET Circuit



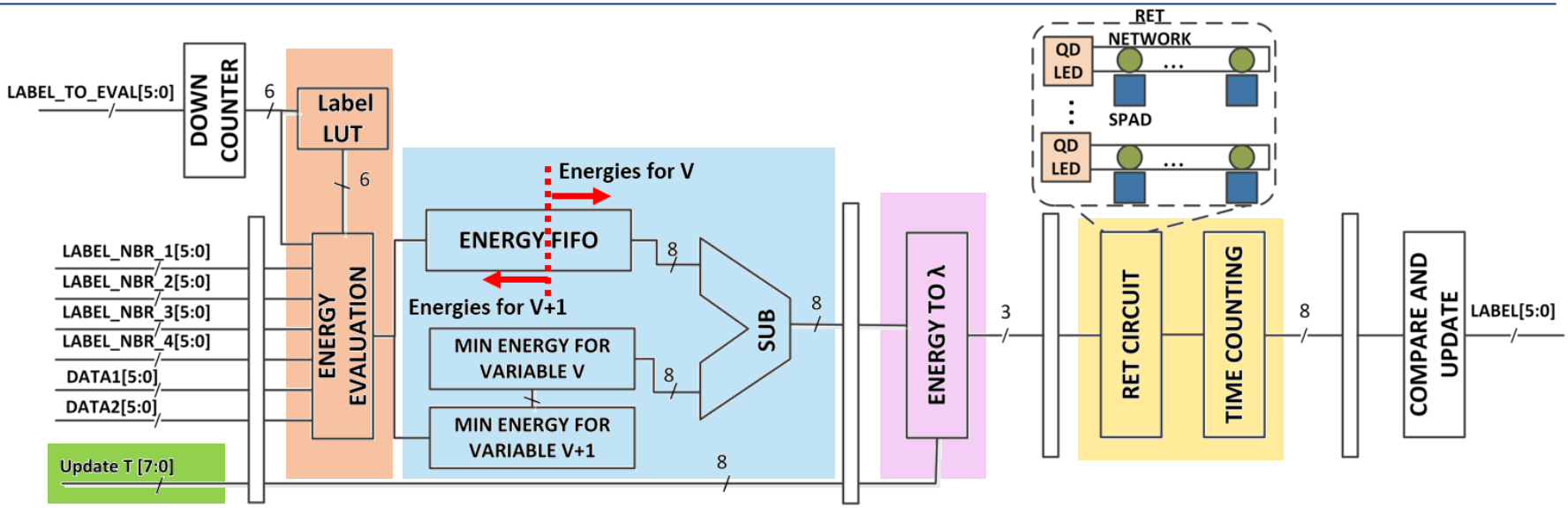
- Previous RET Circuit: QDLEDs dominates the area.
- New RET Circuit:
  - Fixed intensity, 8 waveguides lighted in round robin.
  - Exploit 4 unique decay rates -> 4 unique concentrations.

Sharing Light Sources



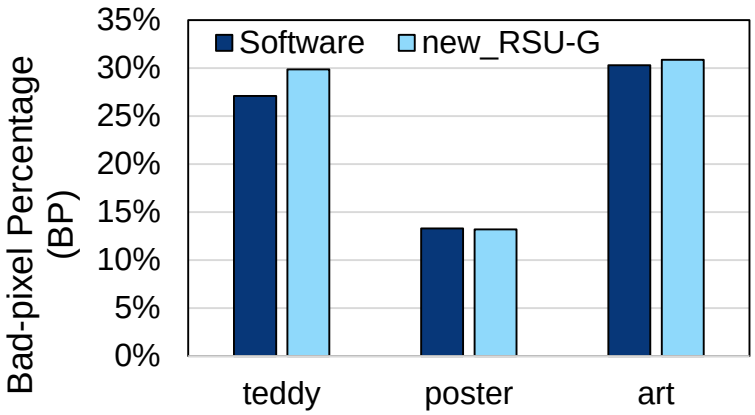
- Multiple RET circuits can share light sources.
- Further reduces area/power.
- Opportunities to use one light source per chip.

New RSU-G

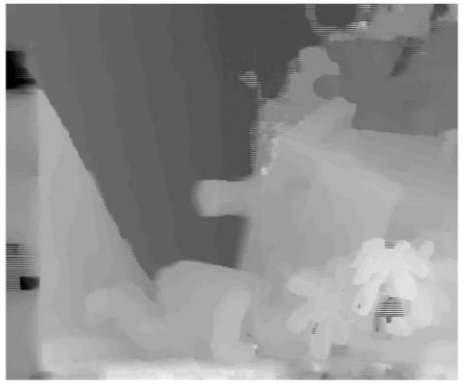


- Circuit and microarchitecture changes:
  - Improved decay rate (λ) dynamic ranges.
  - New RET circuit and peripheral circuits.
  - Support multiple energy function $E(D, L)$ for more applications.
  - Efficient λ conversion.
- Software visible (ISA) change:
  - Addition interface for simulated annealing.
- Details of the pipeline in the paper.

Result Quality



Stereo vision result quality
(lower is better)



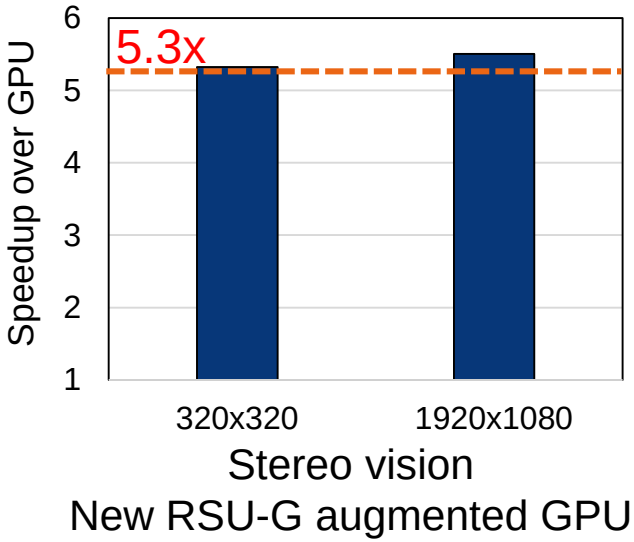
Software disparity map

New RSU-G disparity map

Stereo vision ***teddy*** **dataset**

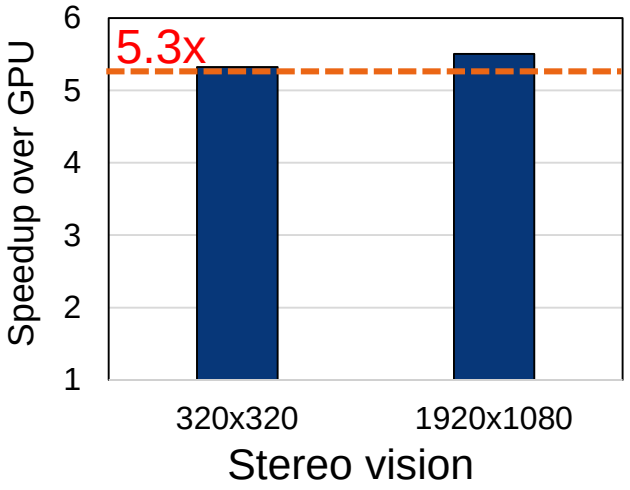
- Standard benchmarks and metrics:
  - Stereo vision.
  - Motion estimation.
  - Image segmentation.
- New RSU-G provides same result quality as software.
- Other two application results in the paper.

Performance / Area / Power

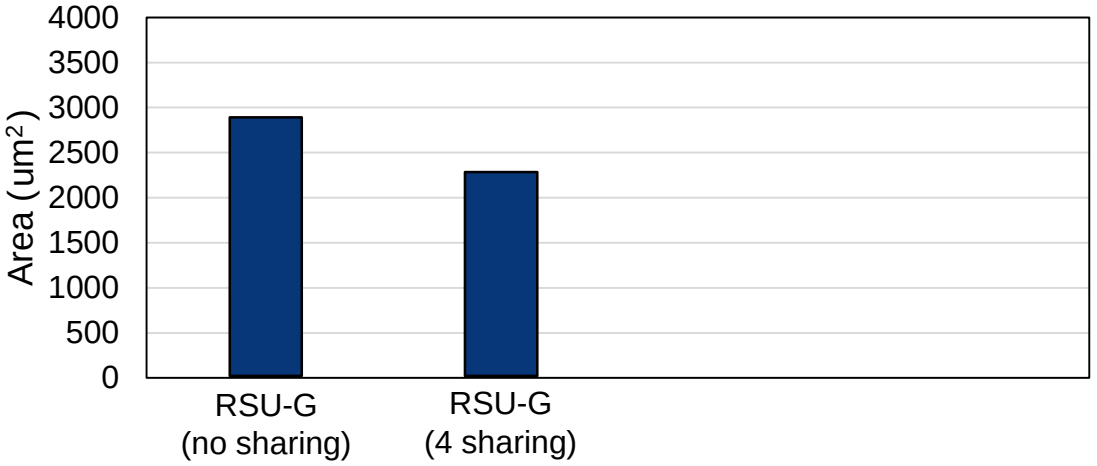


- Preserve speedups.

Performance / Area / Power



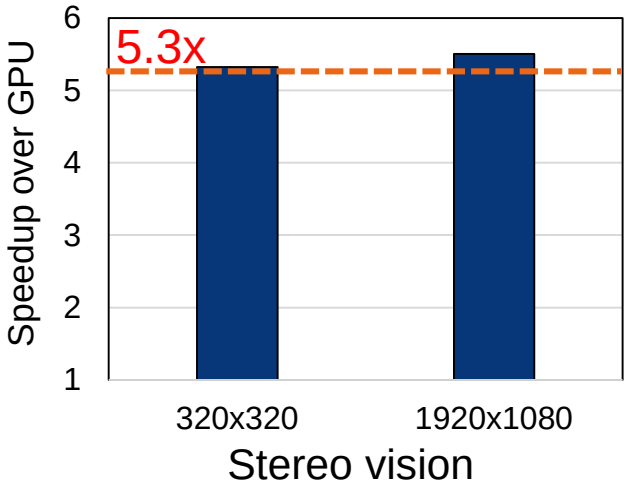
New RSU-G augmented GPU



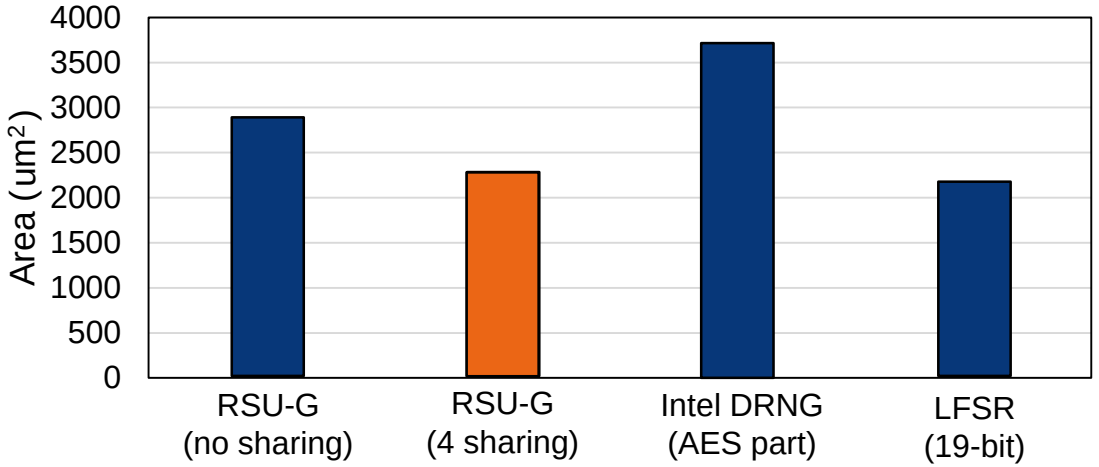
Area: RSU-G

- Preserve speedups.
- New RSU-G: 1.00x area, 1.27x power vs. the previous.

Performance / Area / Power



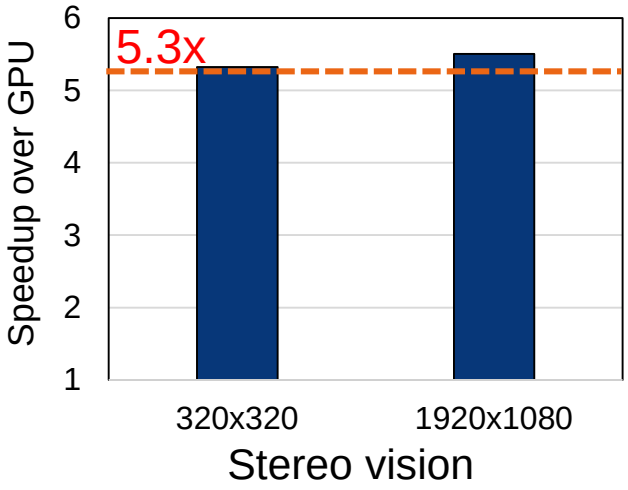
New RSU-G augmented GPU



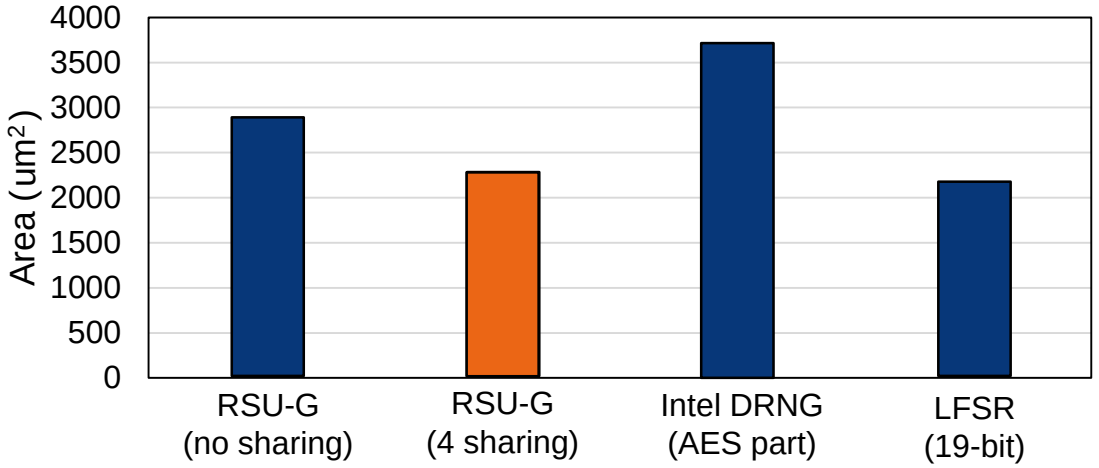
Area: RSU-G vs. CMOS alternatives

- Preserve speedups.
- New RSU-G: 1.00x area, 1.27x power vs. the previous.
- Replace RET part with CMOS sample generation designs:
  - 0.62x area vs. Intel DRNG (AES part). [Hofemeier, 2012]
  - 1.05x area vs. 19-bit LFSR.

Performance / Area / Power



New RSU-G augmented GPU



Area: RSU-G vs. CMOS alternatives

- Preserve speedups.
- New RSU-G: 1.00x area, 1.27x power vs. the previous.
- Replace RET part with CMOS sample generation designs:
  - 0.62x area vs. Intel DRNG (AES part). [Hofemeier, 2012]
  - 1.05x area vs. 19-bit LFSR.
- RSU-G: high quality quantum randomness.

Conclusion

- MCMC: general framework in statistical machine learning.
- Quality analysis on standard benchmarks and metrics.
- Previous RSU-G: didn’t provide required result quality.

Conclusion

- MCMC: general framework in statistical machine learning.
- Quality analysis on standard benchmarks and metrics.
- Previous RSU-G: didn't provide required result quality.
- New RSU-G:
  - **High** result quality.
  - **Sizable** performance benefits.
  - **Minimal** area/power.
  - **More** flexibility.

Conclusion

- MCMC: general framework in statistical machine learning.
- Quality analysis on standard benchmarks and metrics.
- Previous RSU-G: didn't provide required result quality.
- New RSU-G:
  - **High** result quality.
  - **Minimal** area/power.
  - **Sizable** performance benefits.
  - **More** flexibility.
- Opportunities to further reduce area/power/fabrication cost.
- Some techniques are applicable to pure-CMOS accelerators.

Conclusion

- MCMC: general framework in statistical machine learning.
- Quality analysis on standard benchmarks and metrics.
- Previous RSU-G: didn't provide required result quality.
- New RSU-G:


High result quality.


Minimal area/power.


Sizable performance benefits.


More flexibility.

- Opportunities to further reduce area/power/fabrication cost.
- Some techniques are applicable to pure-CMOS accelerators.
- Future Work:
  - Supporting more mathematical MCMC models.
  - Quality analysis on wider application domains.

Thank you

- Q&A