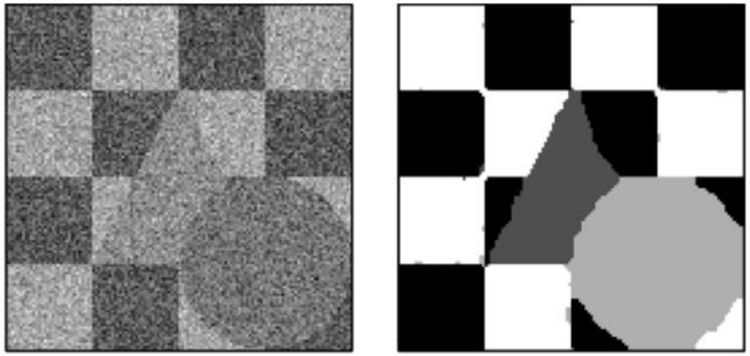
Accelerating Markov Random Field Inference Using Molecular Optical Gibbs Sampling Units

Siyang Wang, Xiangyu Zhang, Yuxuan Li, Ramin Bashizade, Song Yang, Chris Dwyer, Alvin Lebeck

Duke University

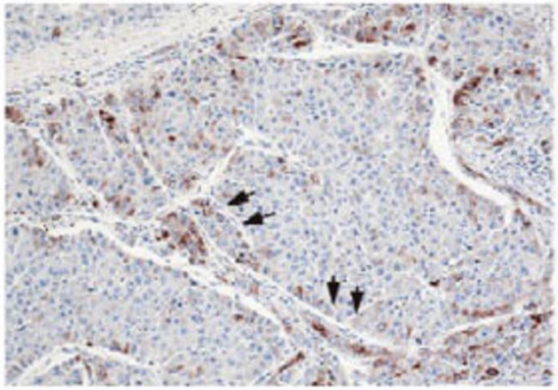
Probabilistic Computing



Input noisy image (SNR=5dB) CNN-MRF Segmented Result

Image Segmentation
[Sziranyi et al., 2000]

Motion Estimation
[Chen et al., 2010]



Predicting hepatitis B virus
[Ye et al., 2003]

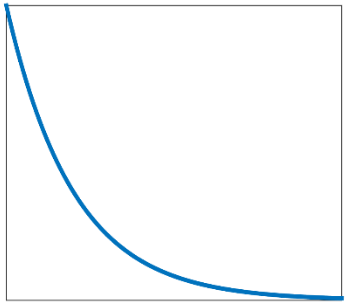
Computer Vision **Medical Diagnosis**

- Machine Learning is in the air!
- Probabilistic algorithms (e.g. Markov Chain Monte Carlo):
  + potential for generalized frameworks.
  + the only viable approach for certain problems.

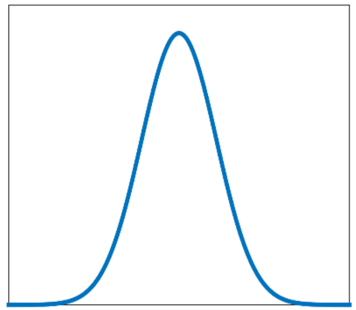
  Key: generating samples.

Problem: Sampling Overhead

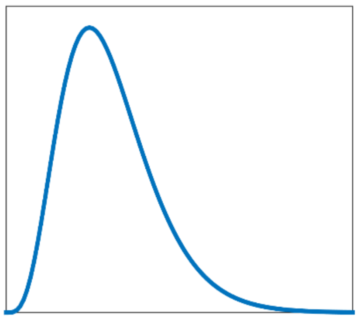
Running from C++ library on Intel E5-2640



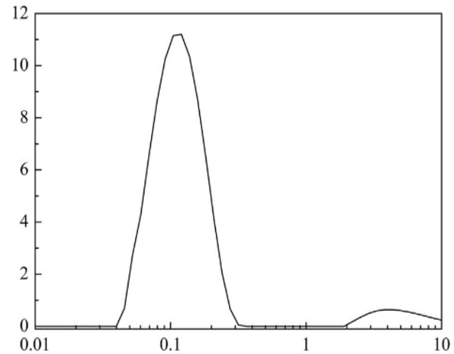
Exponential
588 Cycles



Normal
633 Cycles



Gamma
800 Cycles



(Xu et al., 2016)

Complex Distribution
… Cycles?

- Probabilistic algorithms need many iterations.
- Each iteration requires ≈ **billion** samples.
- **Sampling overhead is TOO HIGH.**
- Alternative 1: Deterministic algorithm approximation:
  - Complex mathematical derivation, limited accuracy.
- Alternative 2: Can we use hardware specialization?

Hardware Specialization Comparison

Property	LFSR	Intel DRNG [Hofemeier, 2012]	Probabilistic CMOS [Chakrapani et al., 2006]	Digital Stochastic Circuits [Mansinghka et al., 2014]
Quality (true random number generation)	✗	✓	✓	✓
Complexity (simple post-processing)	✗	✗	✗	✗
Flexibility (parameterizability)	✗	✗	✓	✓
Flexibility (arbitrary distribution)	✗	✗	✗	✗
Functionality (application value in, application value out)	✗	✗	✗	✓

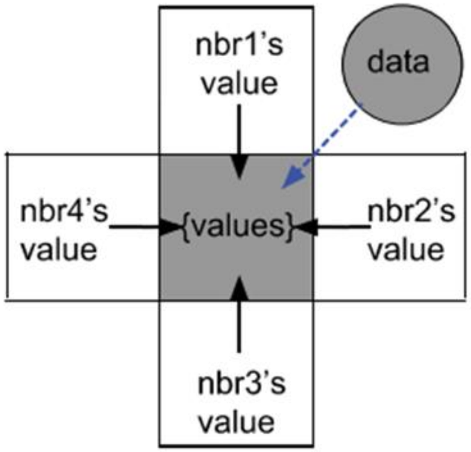
Hardware Specialization Comparison

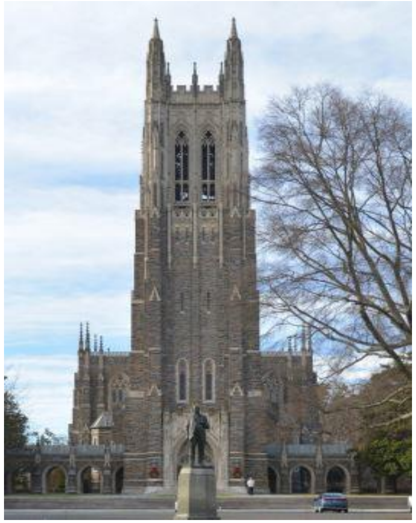
Property	LFSR	Intel DRNG [Hofemeier, 2012]	Probabilistic CMOS [Chakrapani et al., 2006]	Digital Stochastic Circuits [Mansinghka et al., 2014]	Our Proposal
Quality (true random number generation)	✗	✓	✓	✓	✓
Complexity (simple post-processing)	✗	✗	✗	✗	✓
Flexibility (parameterizability)	✗	✗	✓	✓	✓
Flexibility (arbitrary distribution)	✗	✗	✗	✗	✓
Functionality (application value in, application value out)	✗	✗	✗	✓	✓

Outline

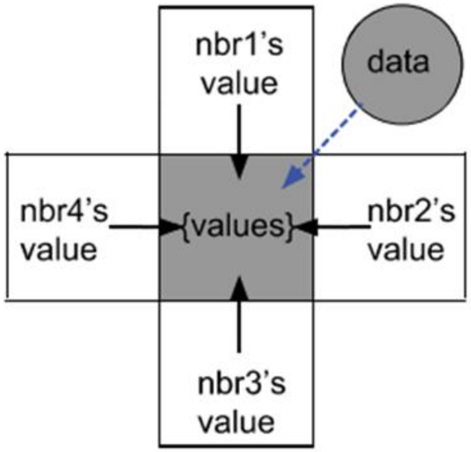
- Motivation
- Background
- RET-based Sampling Unit (RSU)
- RSU Architectures
- Evaluation

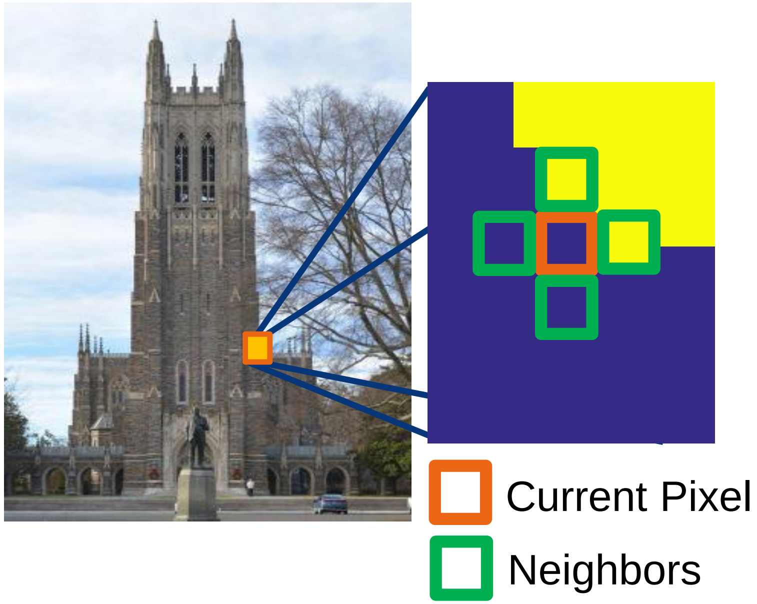
Solving First-order Markov Random Field (MRF)



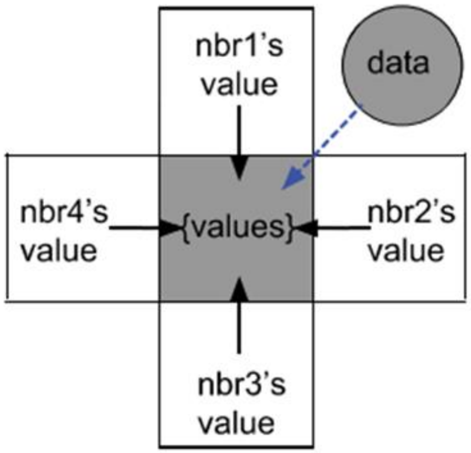


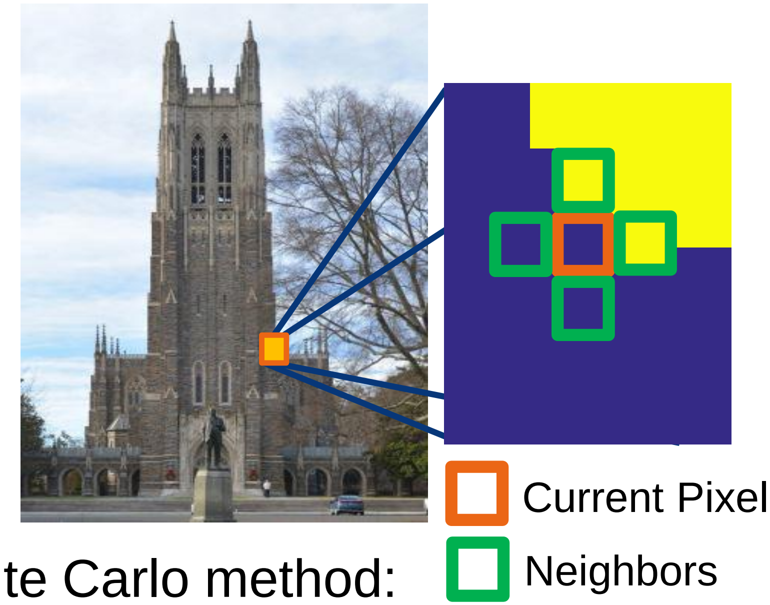
Solving First-order Markov Random Field (MRF)





Solving First-order Markov Random Field (MRF)

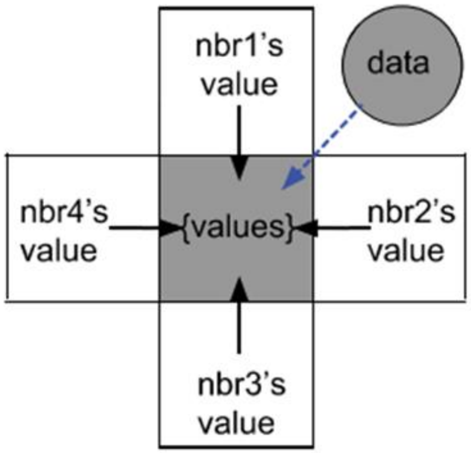


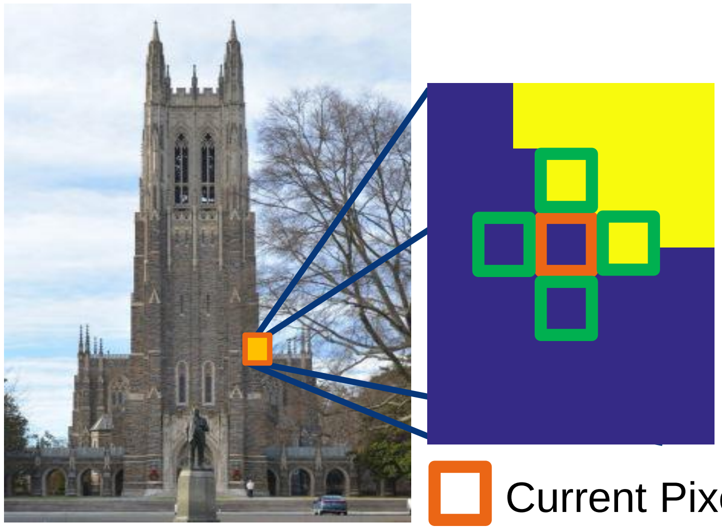


- Markov Chain Monte Carlo method:

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

Solving First-order Markov Random Field (MRF)





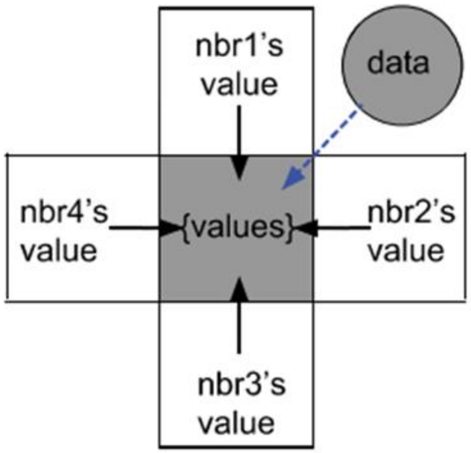
Current Pixel

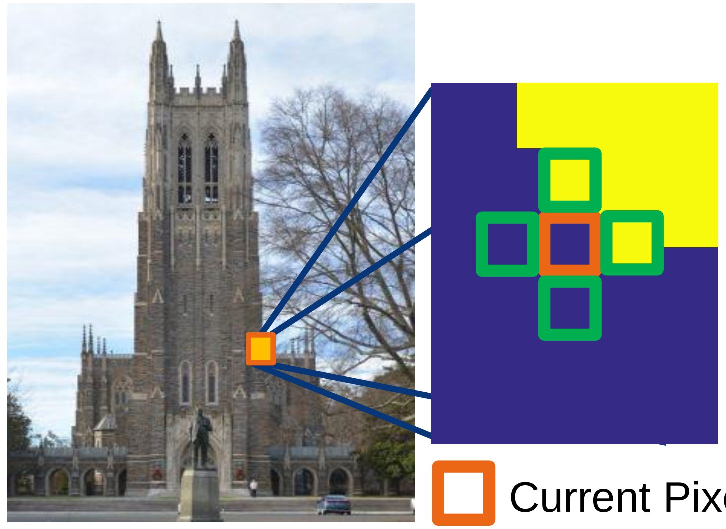
Neighbors

- Markov Chain Monte Carlo method:

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

Solving First-order Markov Random Field (MRF)



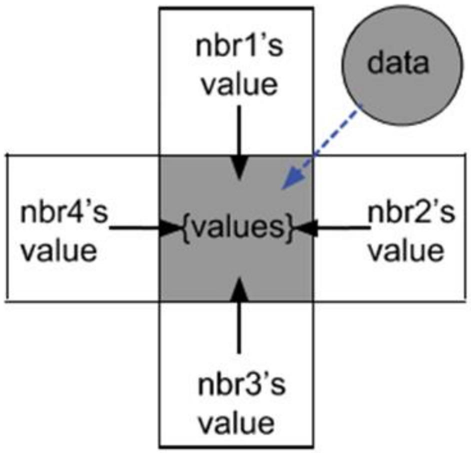


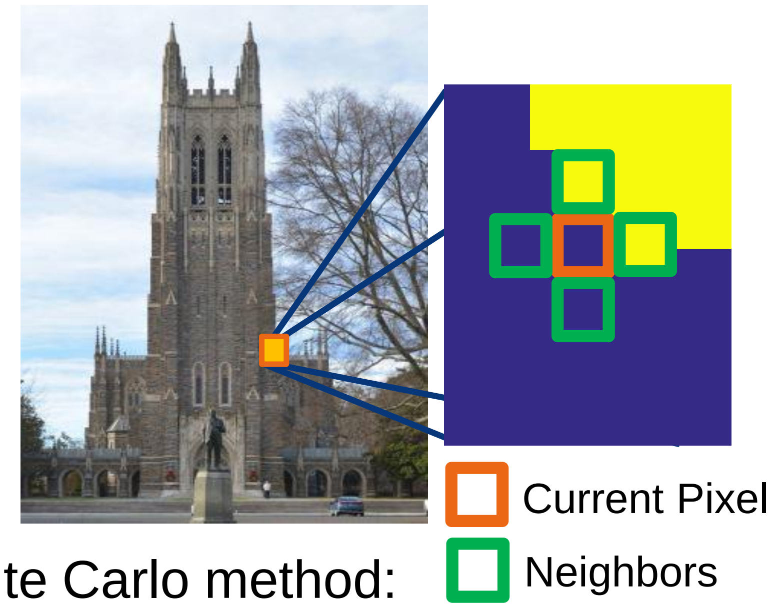
- Markov Chain Monte Carlo method:

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

p(Foreground)=0.3   p(Background)=0.7

Solving First-order Markov Random Field (MRF)





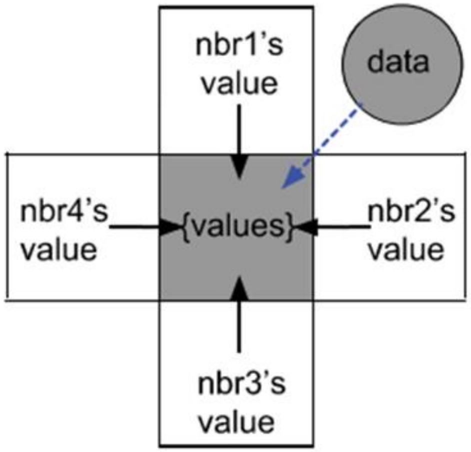
- Markov Chain Monte Carlo method:

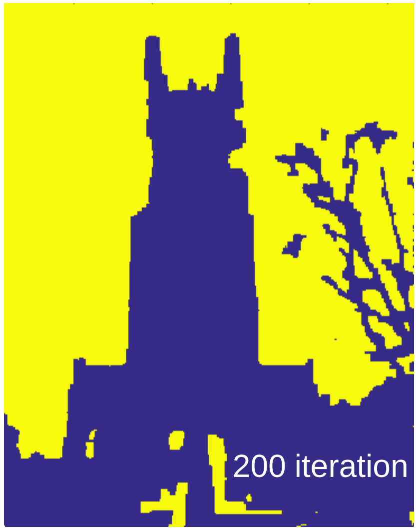
```
while(not converged) {
        for each pixel {
                1) compute probabilities of each possible la
                2) randomly assign new label based on the    abilities;
        }
}
```

p(Foreground)=0.3   p(Backgrou

Source: psychic-junkie

Solving First-order Markov Random Field (MRF)



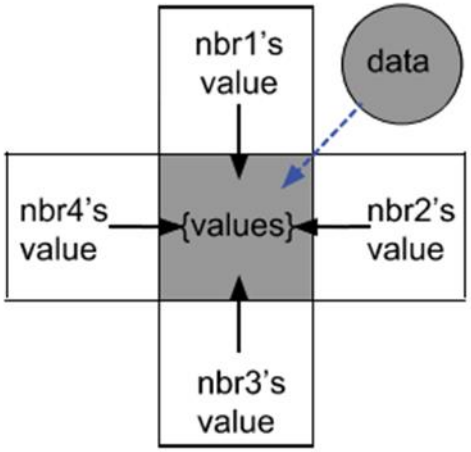


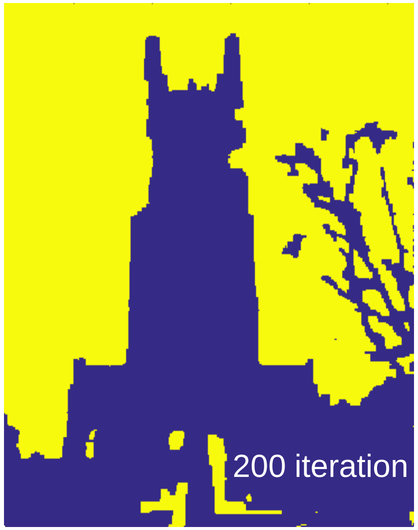
- Markov Chain Monte Carlo method:

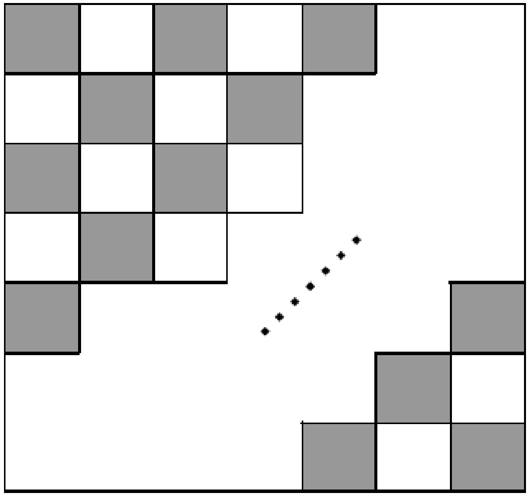
```
while(not converged) {
        for each pixel {
                1) compute probabilities of each possible label;
                2) randomly assign new label based on the probabilities;
        }
}
```

p(Foreground)=0.3   p(Background)=0.7

Solving First-order Markov Random Field (MRF)







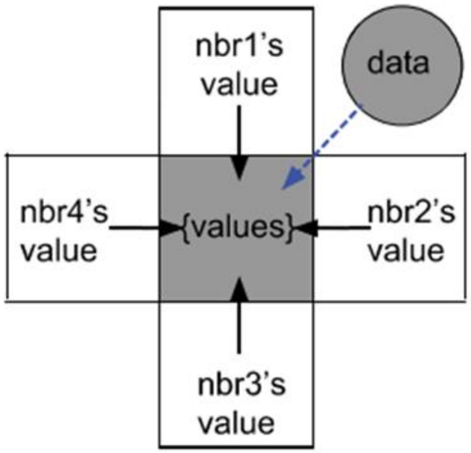
- Markov Chain Monte Carlo method:

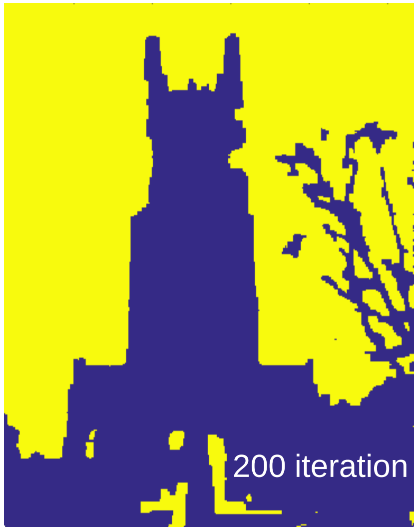
```
while(not converged) {
        for each pixel {
                1) compute probabilities of each possible label;
                2) randomly assign new label based on the probabilities;
        }
}
```

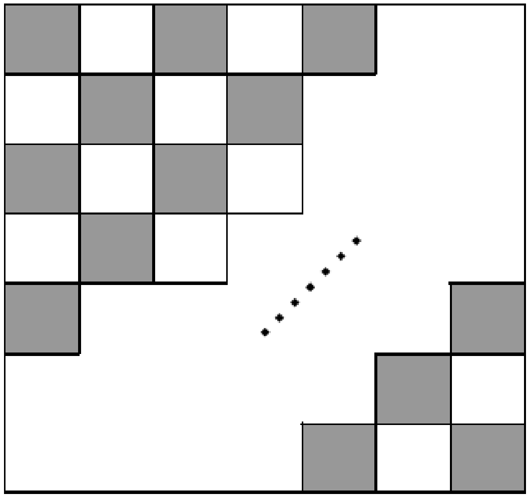
p(Foreground)=0.3   p(Background)=0.7

- Checkerboard update.

Solving First-order Markov Random Field (MRF)







- Markov Chain Monte Carlo method:

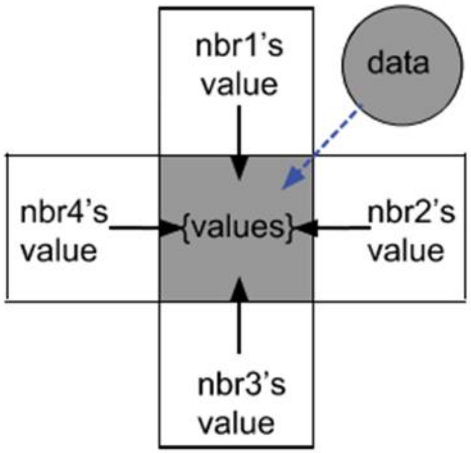
```
while(not converged) {
        for each pixel {
                1) compute probabilities of each possible label;
                2) randomly assign new label based on the probabilities;
        }
}
```

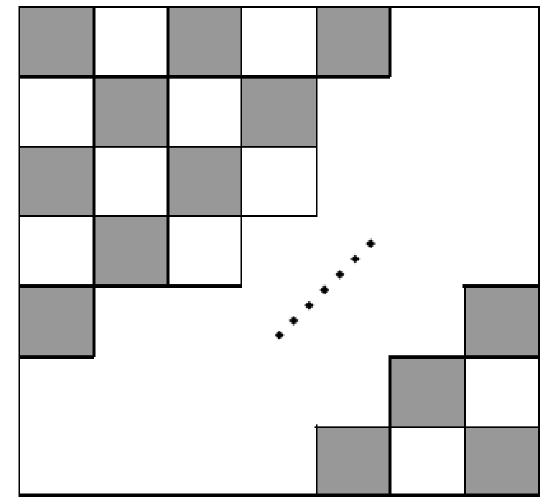
Slow!

p(Foreground)=0.3   p(Background)=0.7

- Checkerboard update.

Solving First-order Markov Random Field (MRF)





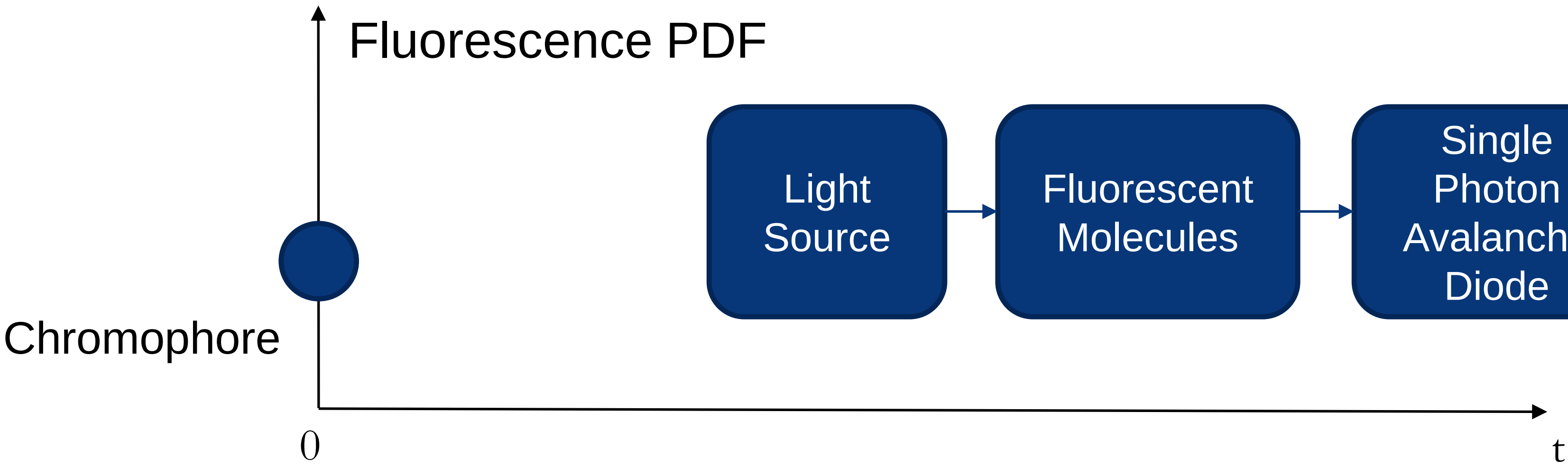
- Markov Chain Monte Carlo method:

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

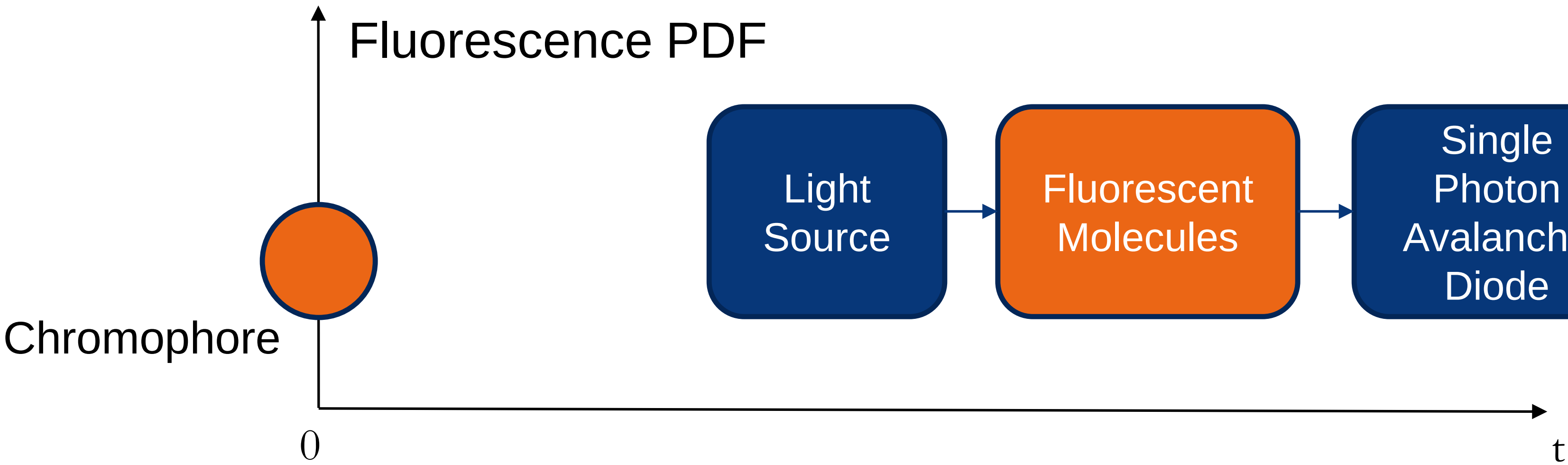
p(Foreground)=0.3 p(Background)=0.7

- Checkerboard update.

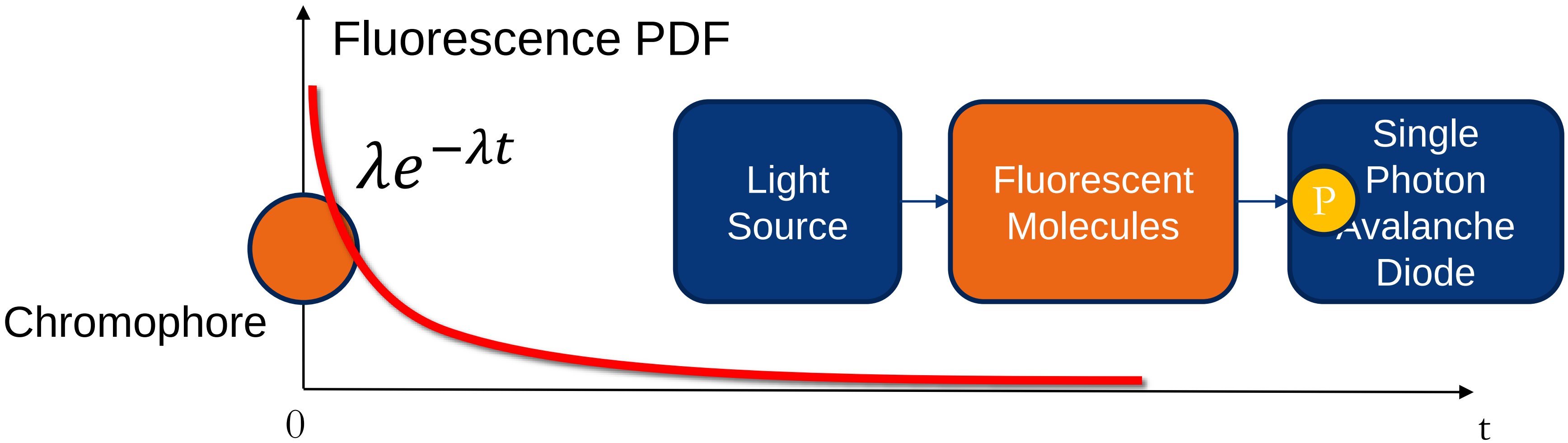
Sampling Using Molecules



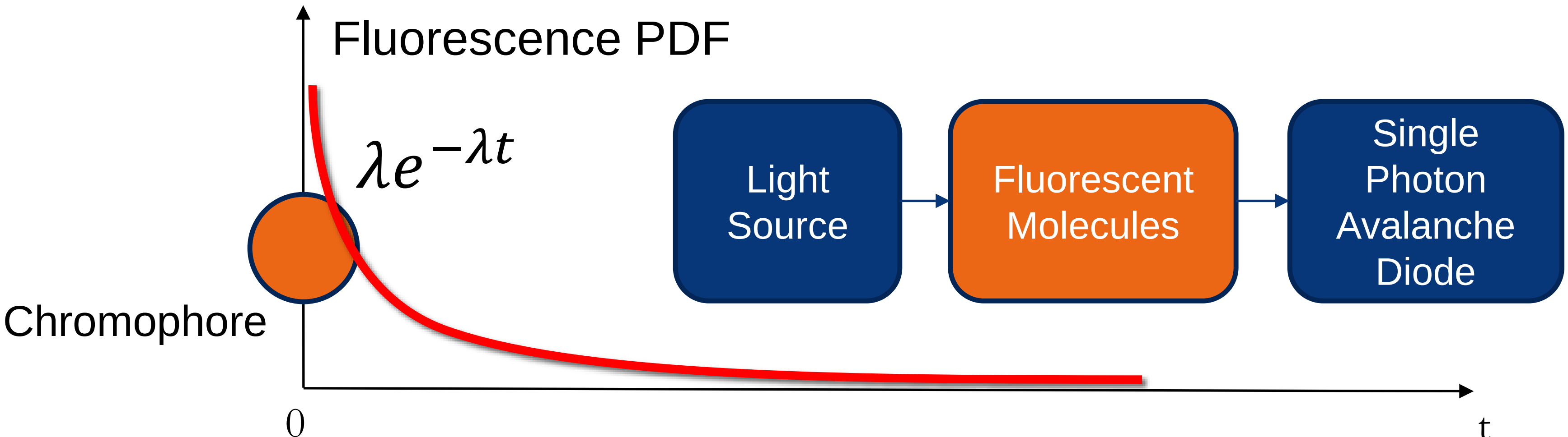
Sampling Using Molecules



Sampling Using Molecules

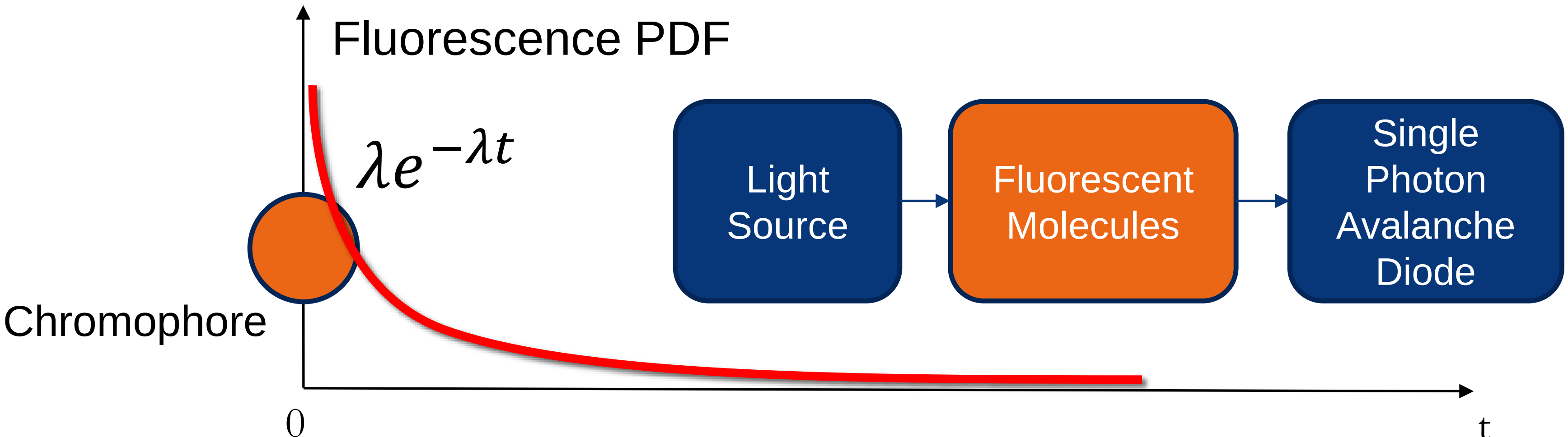


Sampling Using Molecules



Record Time to Fluorescence t

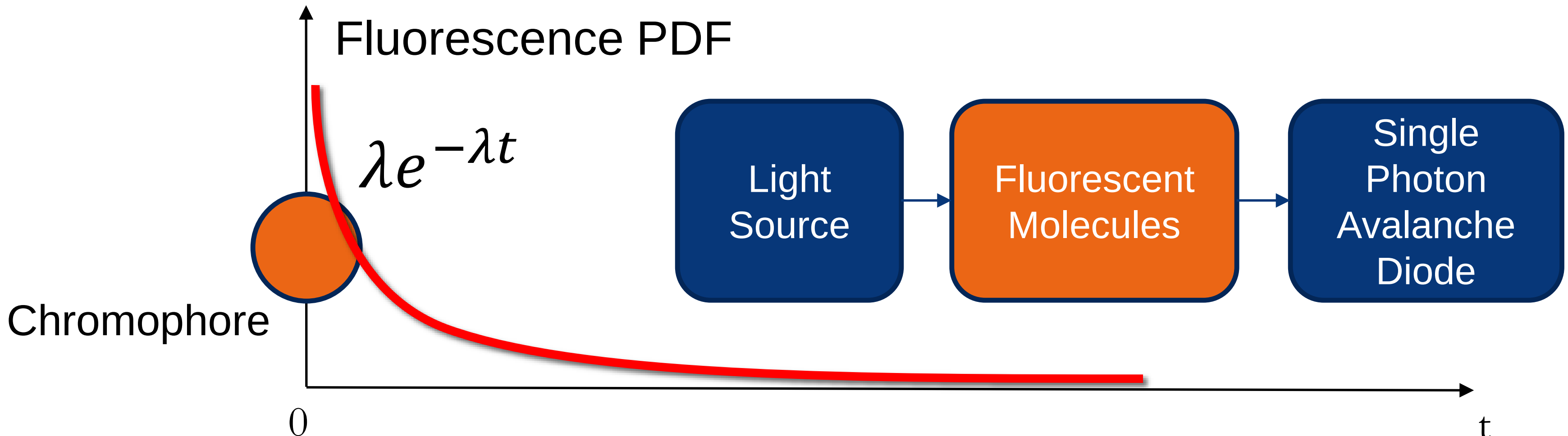
Sampling Using Molecules



Record Time to Fluorescence t

- High quality quantum randomness.
- Single chromophore: exponential distribution.

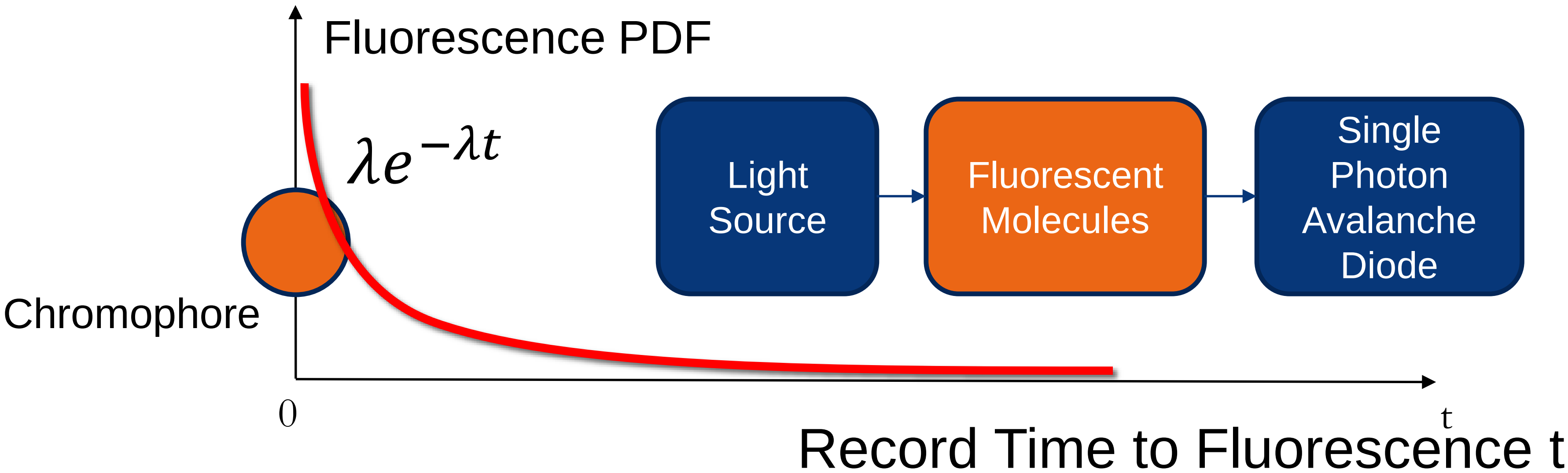
Sampling Using Molecules



Record Time to Fluorescence t

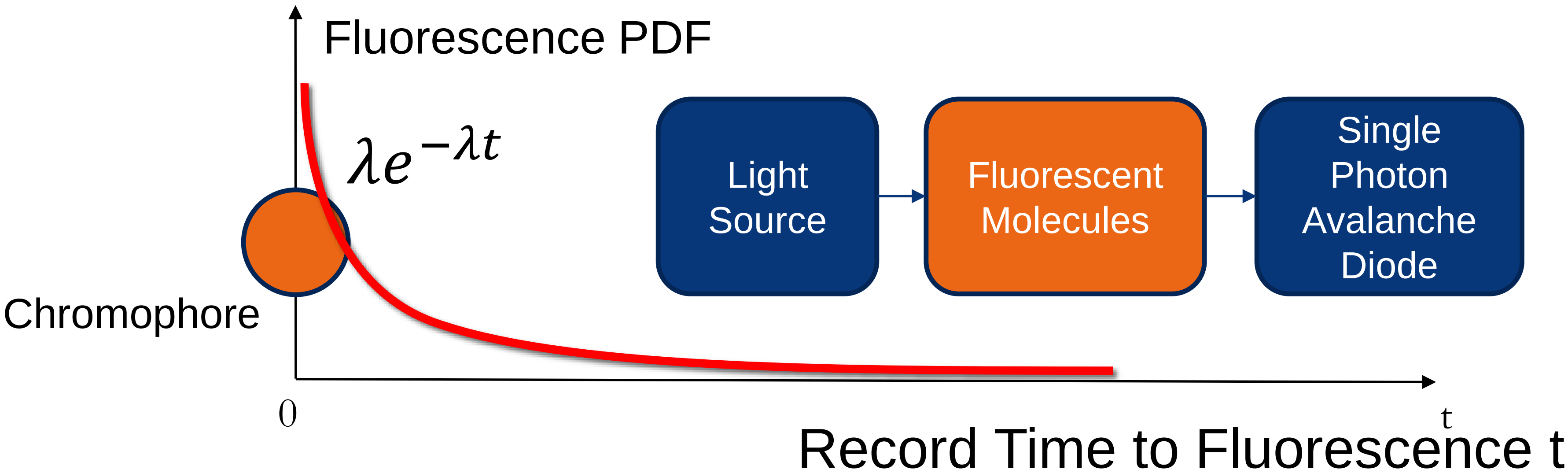
- High quality quantum randomness. ✓ True randomness
- Single chromophore: exponential distribution.

Sampling Using Molecules



- High quality quantum randomness. ✓ True randomness
- Single chromophore: exponential distribution.
- Many ways to parameterize distributions [Wang et al., 2015]

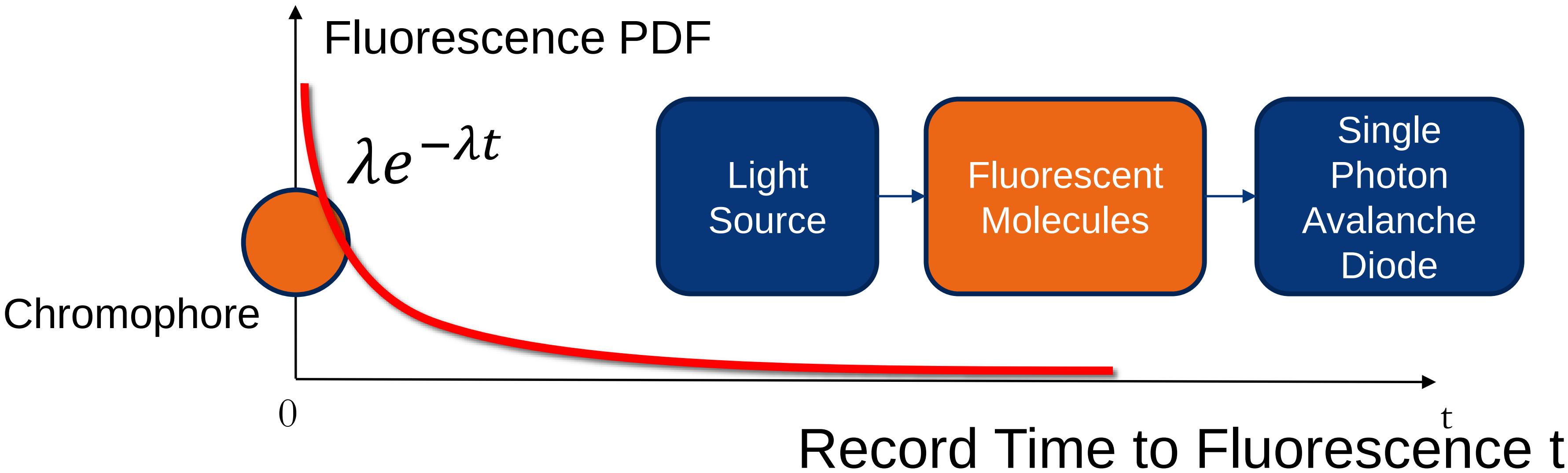
Sampling Using Molecules



- High quality quantum randomness. ✓ True randomness
- Single chromophore: exponential distribution.
- Many ways to parameterize distributions [Wang et al., 2015]

$$\frac{\text{p(Foreground)}}{\text{p(Background)}} = \frac{\lambda_1}{\lambda_2} = \frac{\text{Intensity}_1}{\text{Intensity}_2}$$

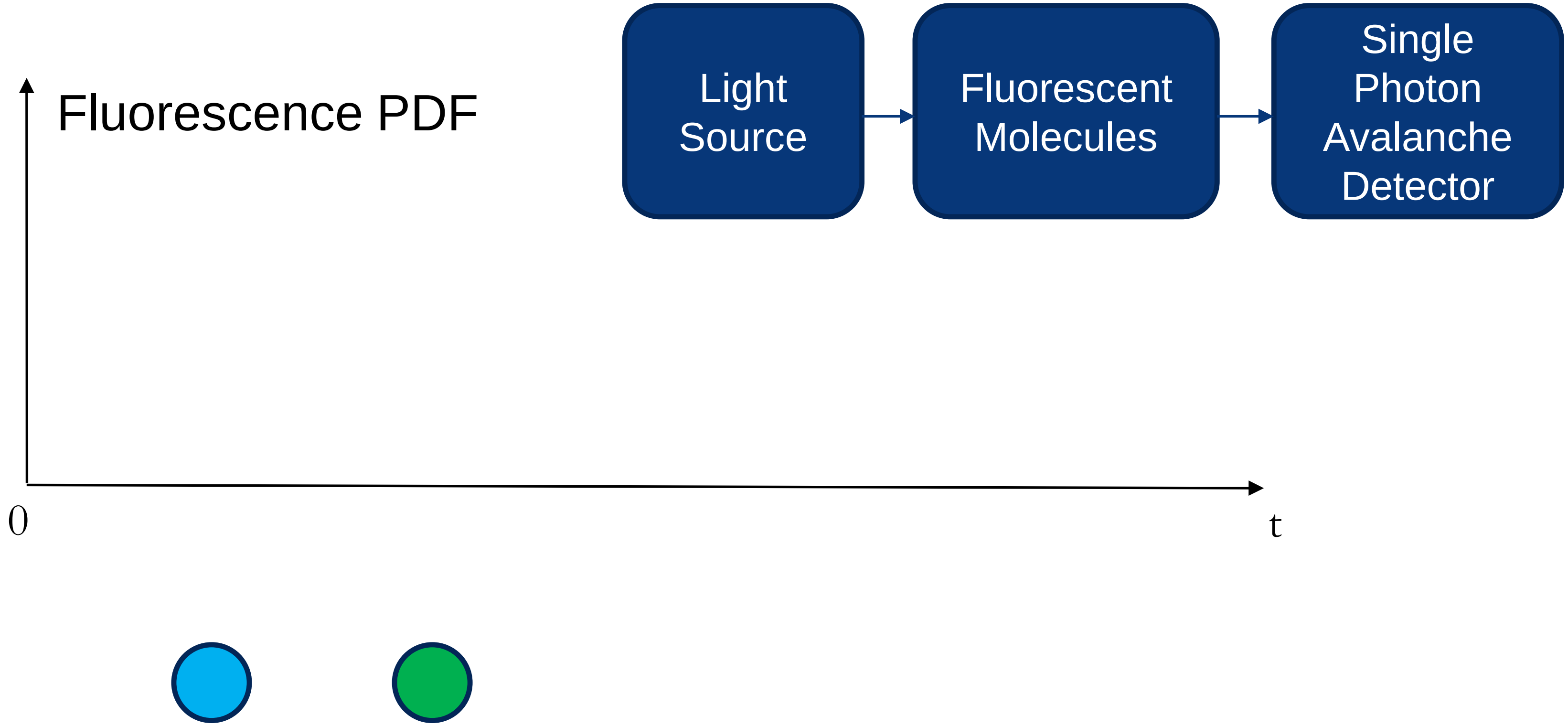
Sampling Using Molecules



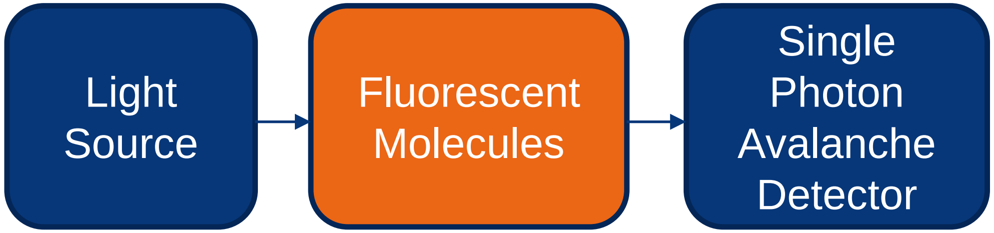
- High quality quantum randomness. ✓ True randomness
- Single chromophore: exponential distribution.
- Many ways to parameterize distributions [Wang et al., 2015]

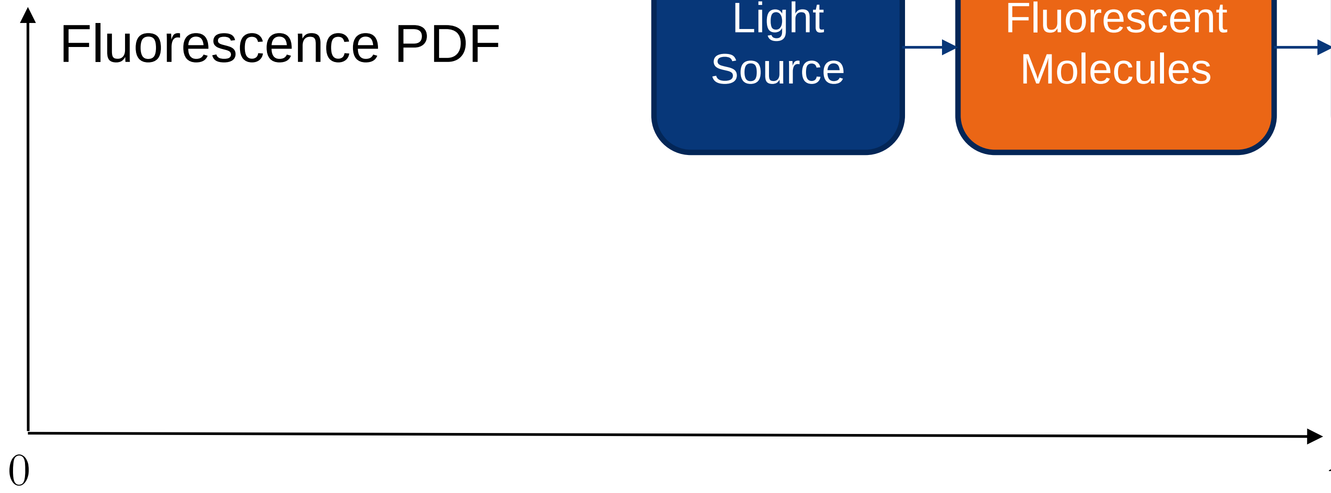
$$\frac{\text{p(Foreground)}}{\text{p(Background)}} = \frac{\lambda_1}{\lambda_2} = \frac{\text{Intensity}_1}{\text{Intensity}_2}$$ ✓ Parameterizability

Resonance Energy Transfer Network

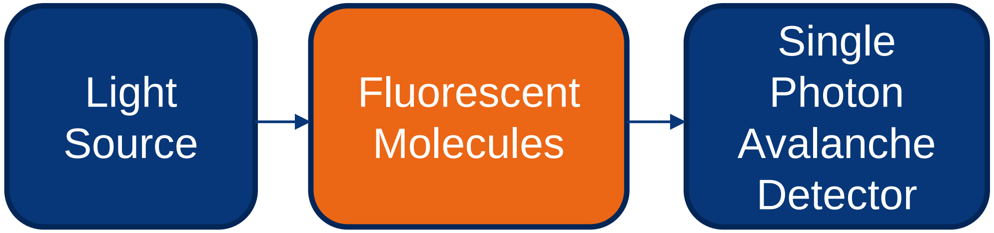


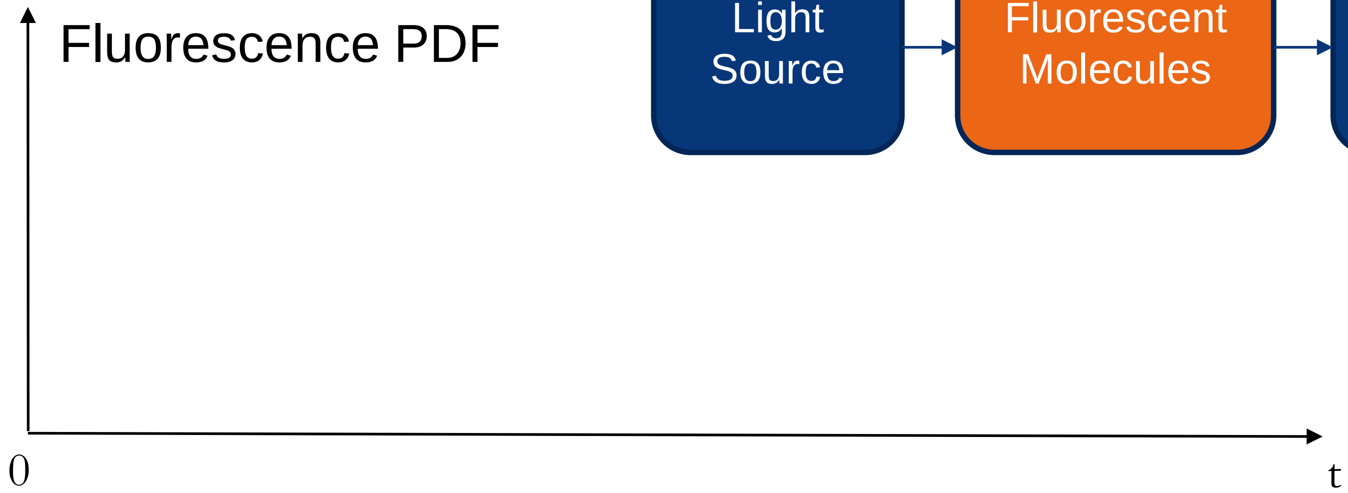
Resonance Energy Transfer Network





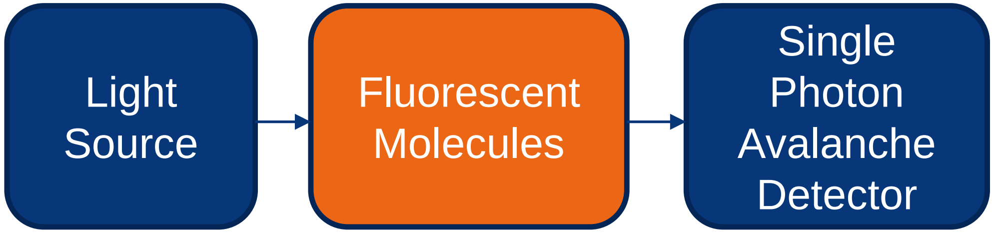
Resonance Energy Transfer Network

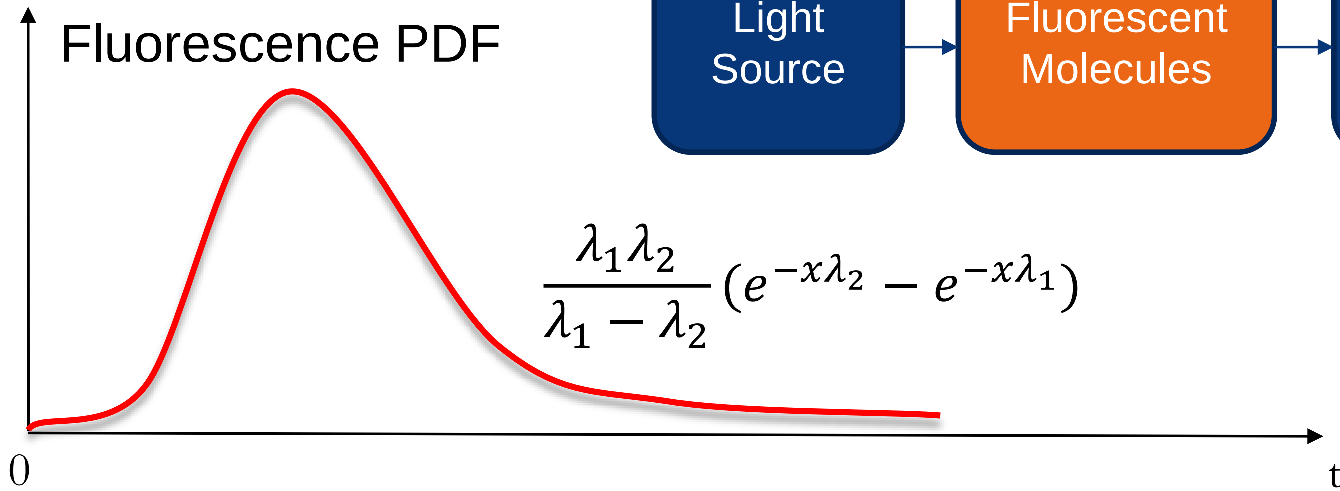




Resonance Energy Transfer

Resonance Energy Transfer Network

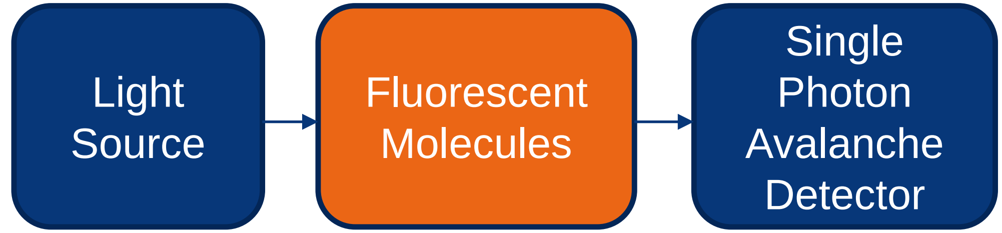


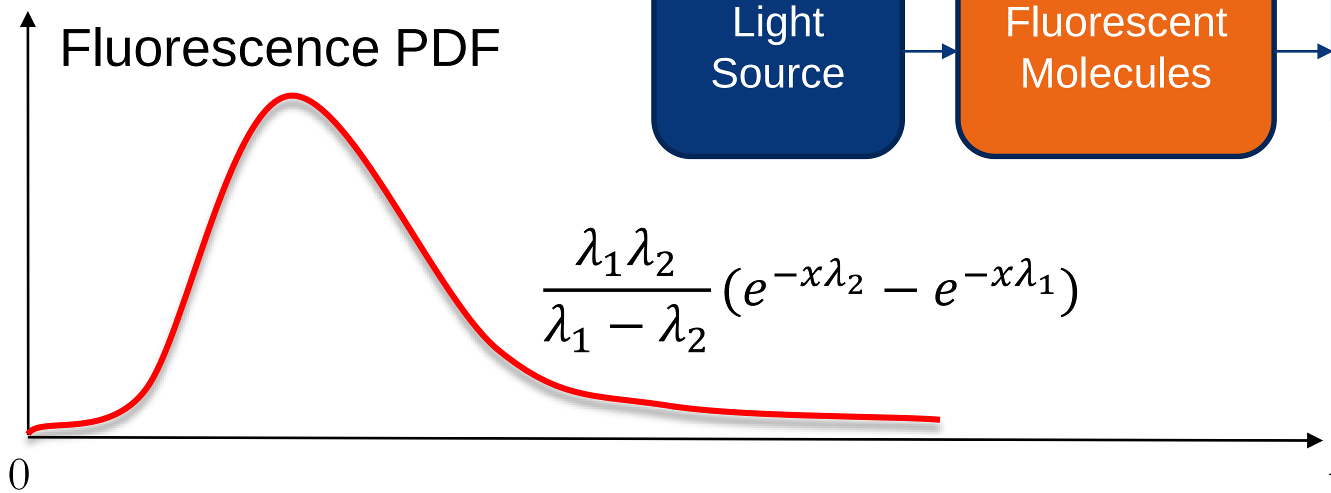


$$\frac{\lambda_1 \lambda_2}{\lambda_1 - \lambda_2}\left(e^{-x\lambda_2} - e^{-x\lambda_1}\right)$$

Resonance Energy Transfer

Resonance Energy Transfer Network



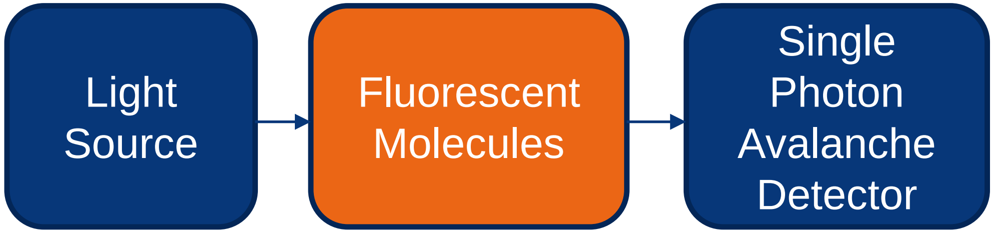


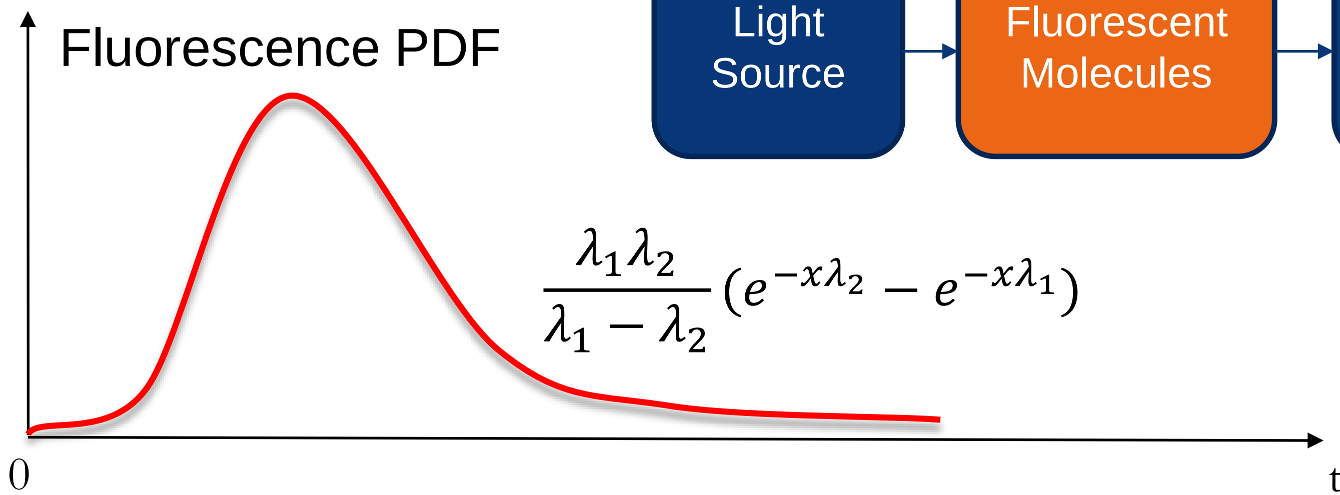
$$\frac{\lambda_1 \lambda_2}{\lambda_1 - \lambda_2}\left(e^{-x\lambda_2} - e^{-x\lambda_1}\right)$$

Resonance Energy Transfer

- Multi-chromophore structure: phase-type distribution [Wang et al., 2015].

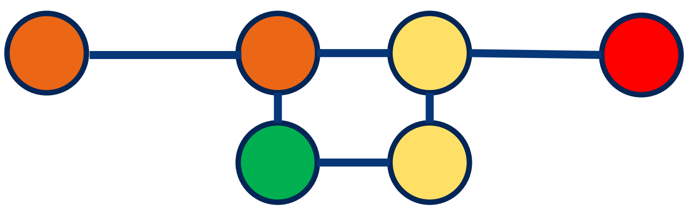
Resonance Energy Transfer Network





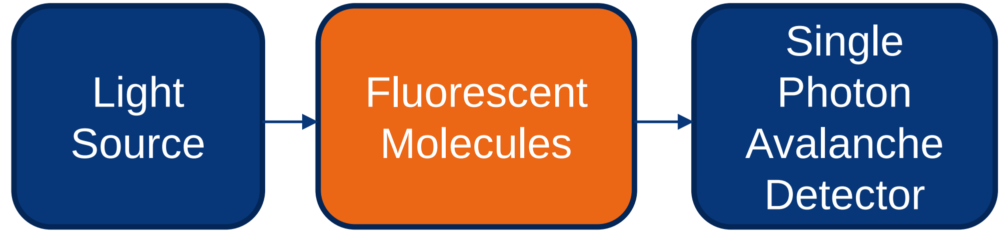
$$\frac{\lambda_1\lambda_2}{\lambda_1 - \lambda_2}\left(e^{-x\lambda_2} - e^{-x\lambda_1}\right)$$

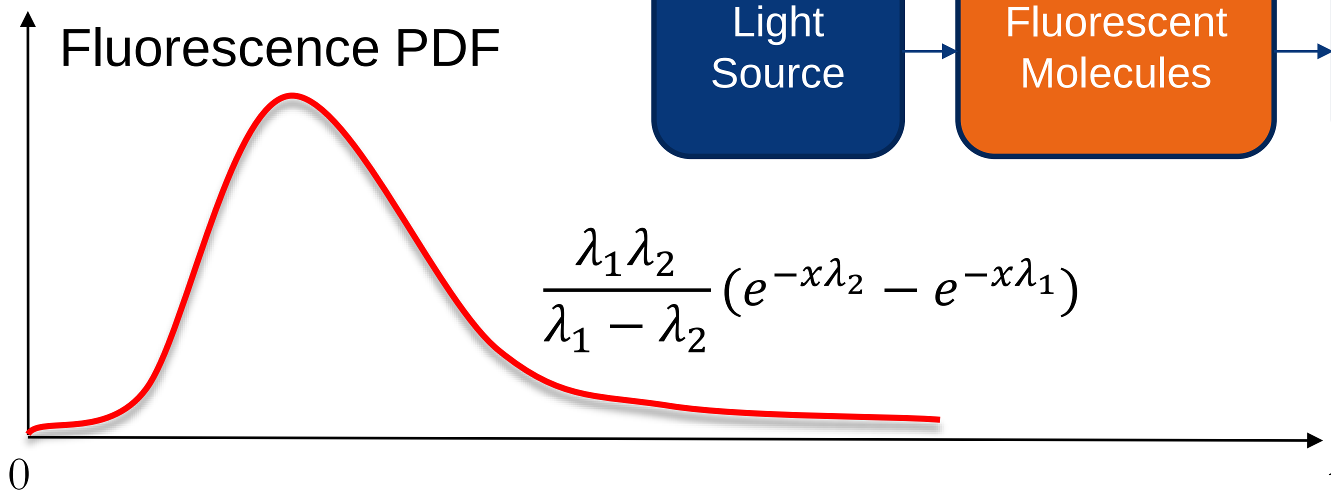
Resonance Energy Transfer



- Multi-chromophore structure: phase-type distribution [Wang et al., 2015].

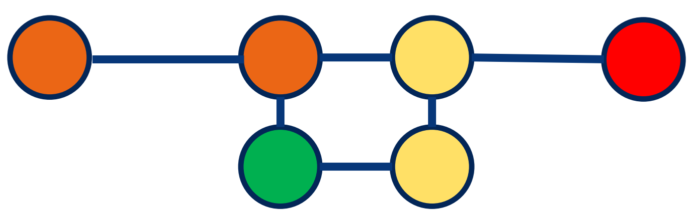
Resonance Energy Transfer Network





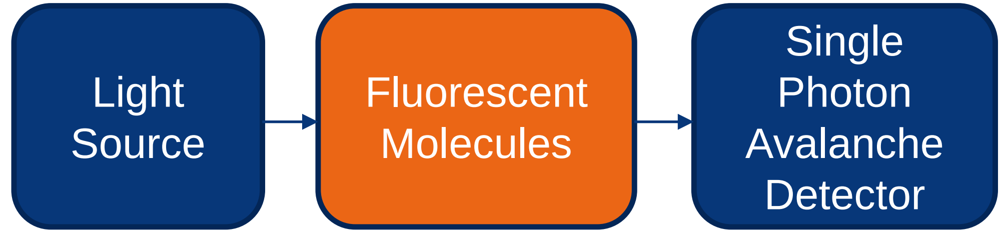
$$\frac{\lambda_1 \lambda_2}{\lambda_1 - \lambda_2}\left(e^{-x\lambda_2} - e^{-x\lambda_1}\right)$$

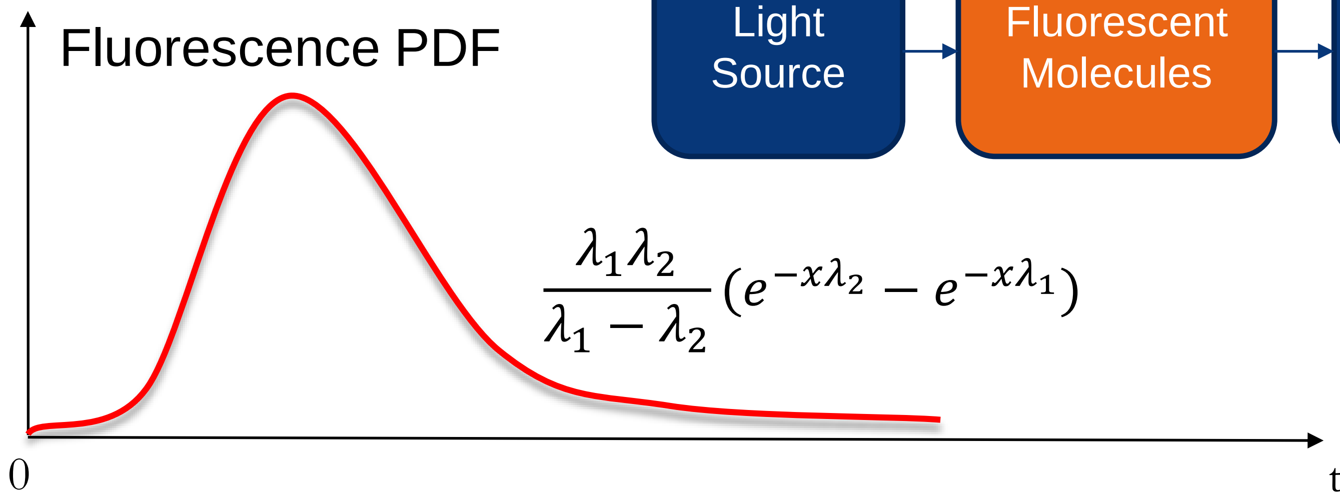
Resonance Energy Transfer



- Multi-chromophore structure: phase-type distribution [Wang et al., 2015].
- Can fit most distributions to phase-type distribution [Asmussen et al., 1996].

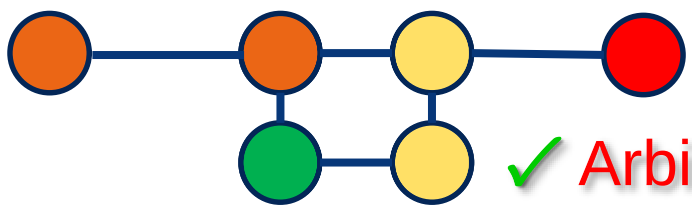
Resonance Energy Transfer Network





$$\frac{\lambda_1 \lambda_2}{\lambda_1 - \lambda_2}\left(e^{-x\lambda_2} - e^{-x\lambda_1}\right)$$

Resonance Energy Transfer

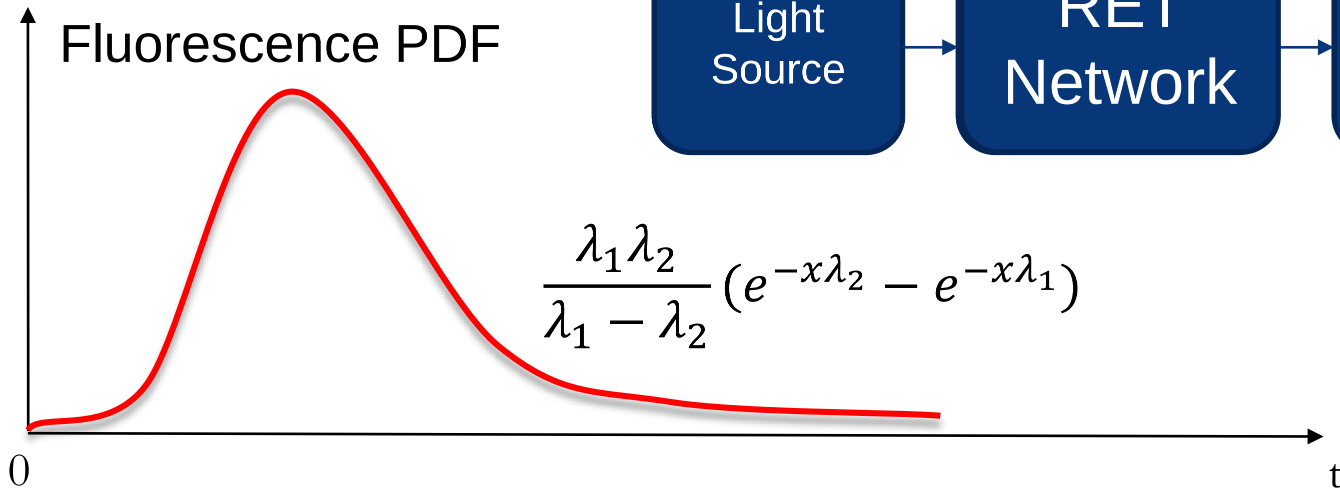


✓ Arbitrary distribution

- Multi-chromophore structure: phase-type distribution [Wang et al., 2015].
- Can fit most distributions to phase-type distribution [Asmussen et al., 1996].

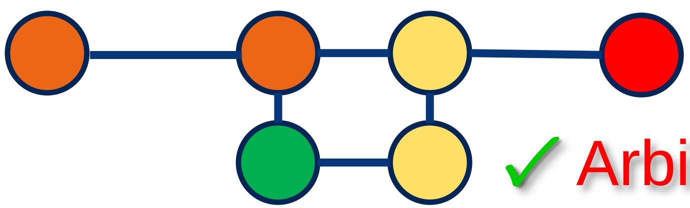
Resonance Energy Transfer Network

RET Circuit



$$\frac{\lambda_1\lambda_2}{\lambda_1-\lambda_2}\left(e^{-x\lambda_2}-e^{-x\lambda_1}\right)$$

Resonance Energy Transfer



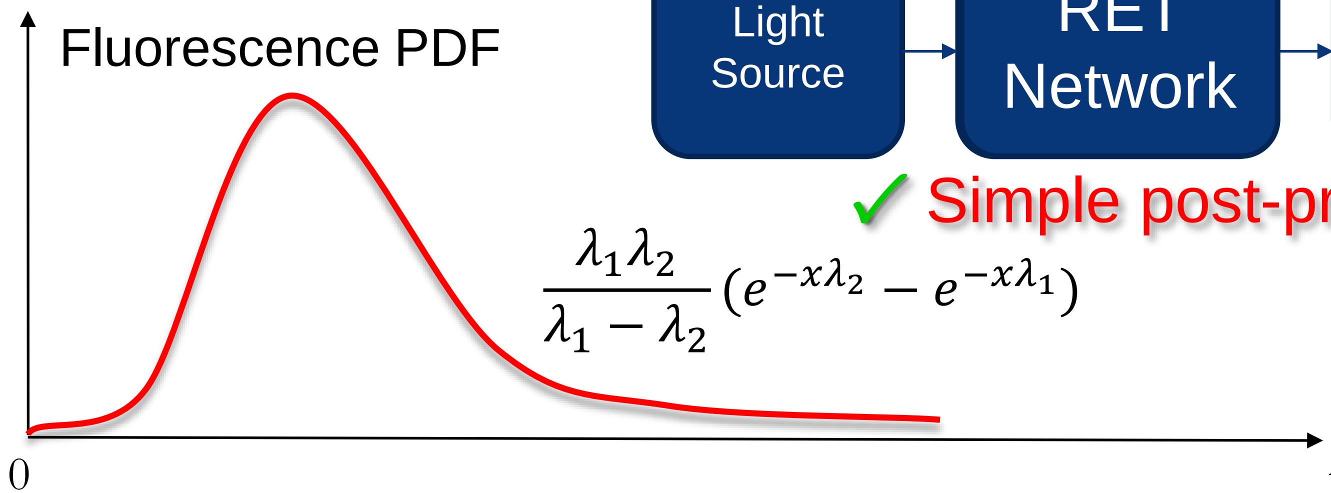
✓ Arbitrary distribution

- Multi-chromophore structure: phase-type distribution [Wang et al., 2015].
- Can fit most distributions to phase-type distribution [Asmussen et al., 1996].

Resonance Energy Transfer Network

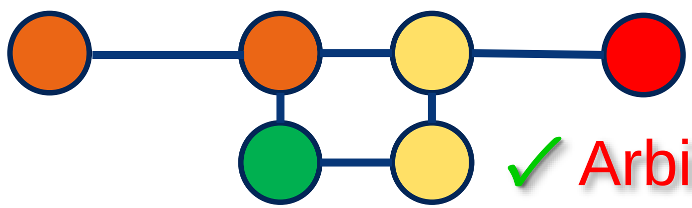
RET Circuit

✓ Simple post-processing



$$\frac{\lambda_1 \lambda_2}{\lambda_1 - \lambda_2}\left(e^{-x\lambda_2} - e^{-x\lambda_1}\right)$$

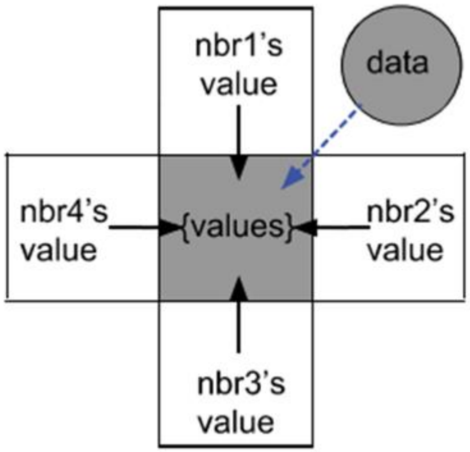
Resonance Energy Transfer

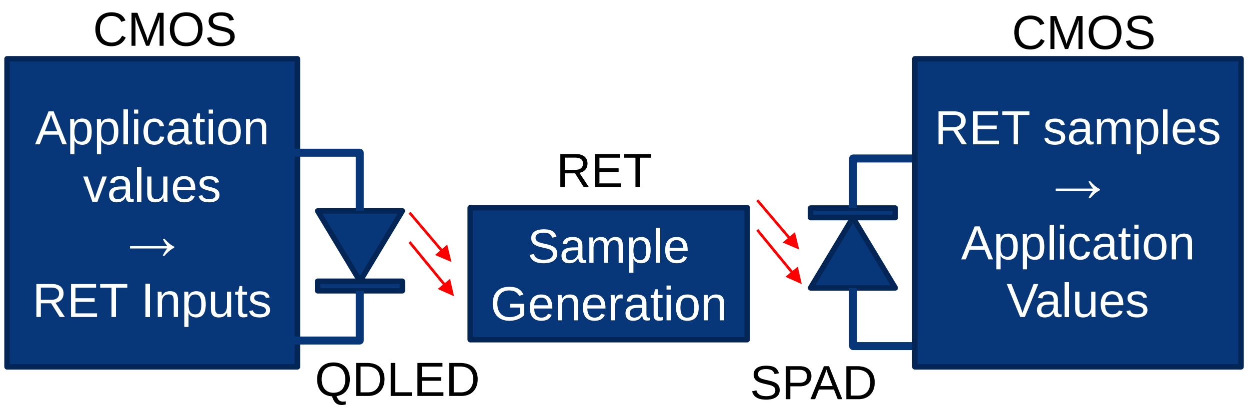


✓ Arbitrary distribution

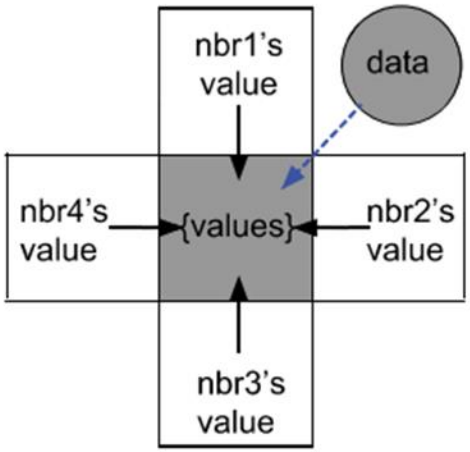
- Multi-chromophore structure: phase-type distribution [Wang et al., 2015].
- Can fit most distributions to phase-type distribution [Asmussen et al., 1996].

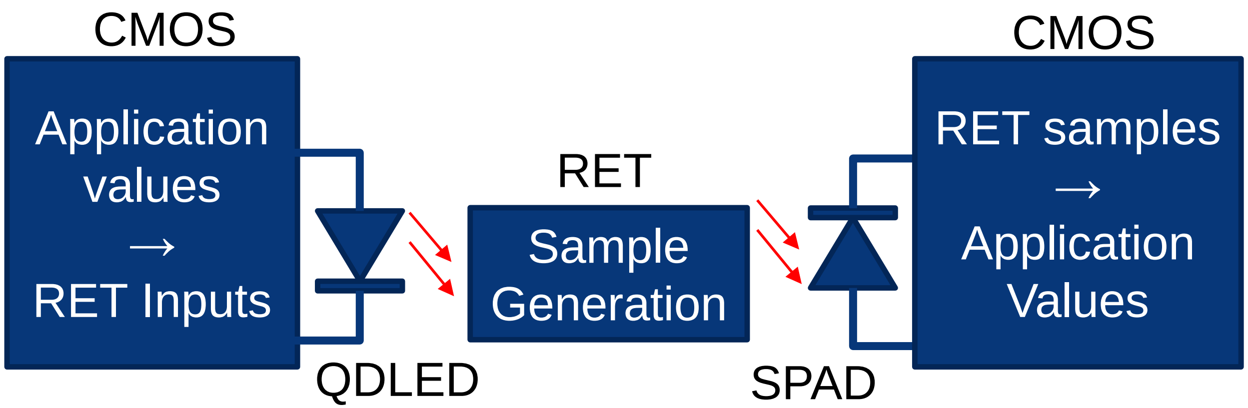
RET-based Gibbs Sampling Unit (RSU-G)





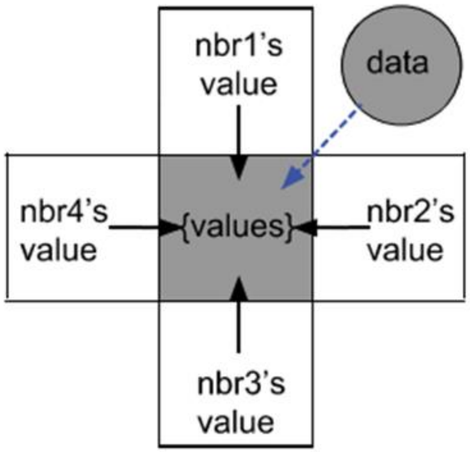
RET-based Gibbs Sampling Unit (RSU-G)

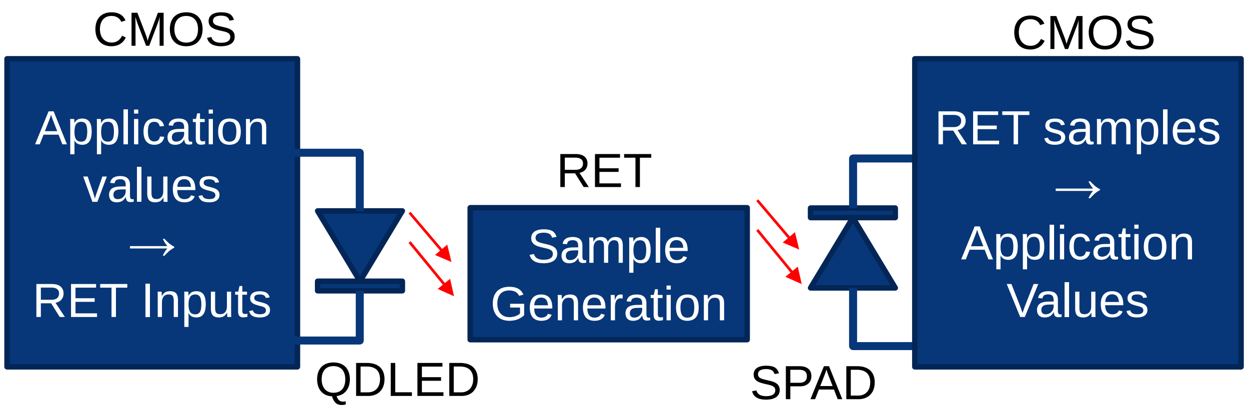




- Hybrid of CMOS + RET technology.

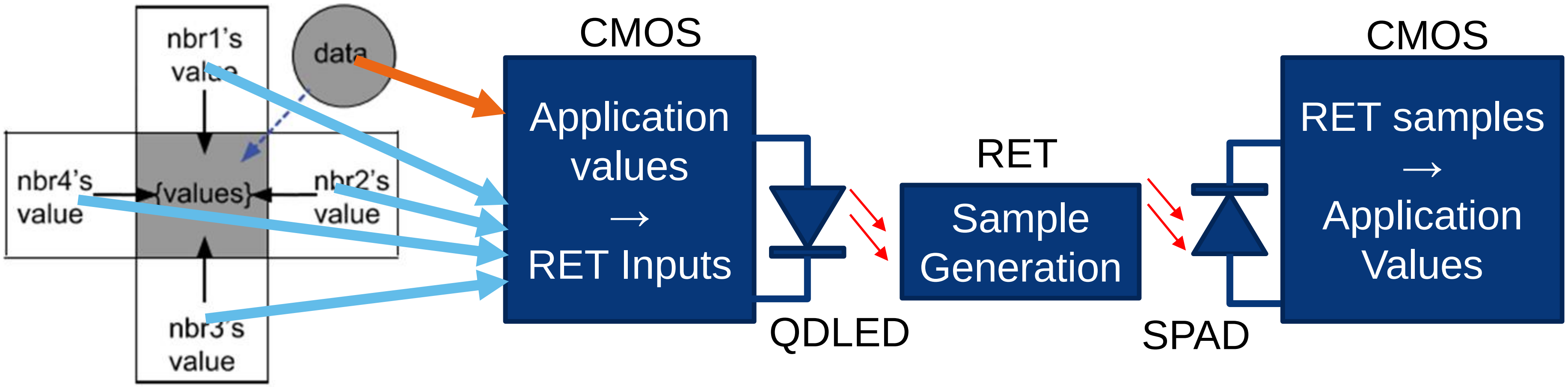
RET-based Gibbs Sampling Unit (RSU-G)





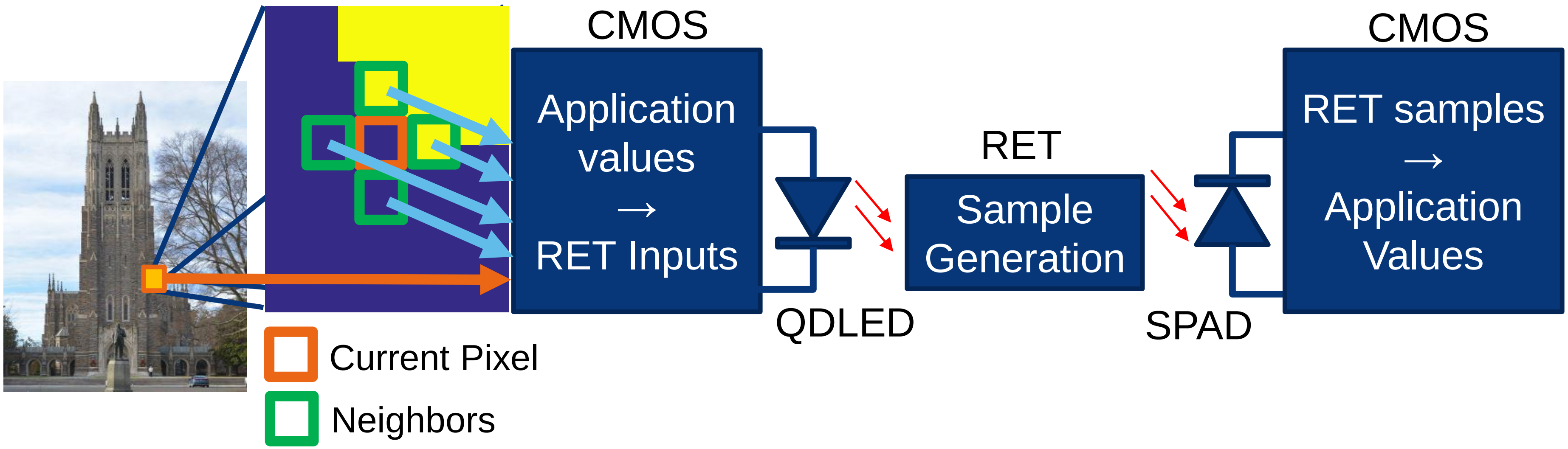
- Hybrid of CMOS + RET technology.
- Label-in label-out.
- Potentially generalize for arbitrary application interface.

RET-based Gibbs Sampling Unit (RSU-G)



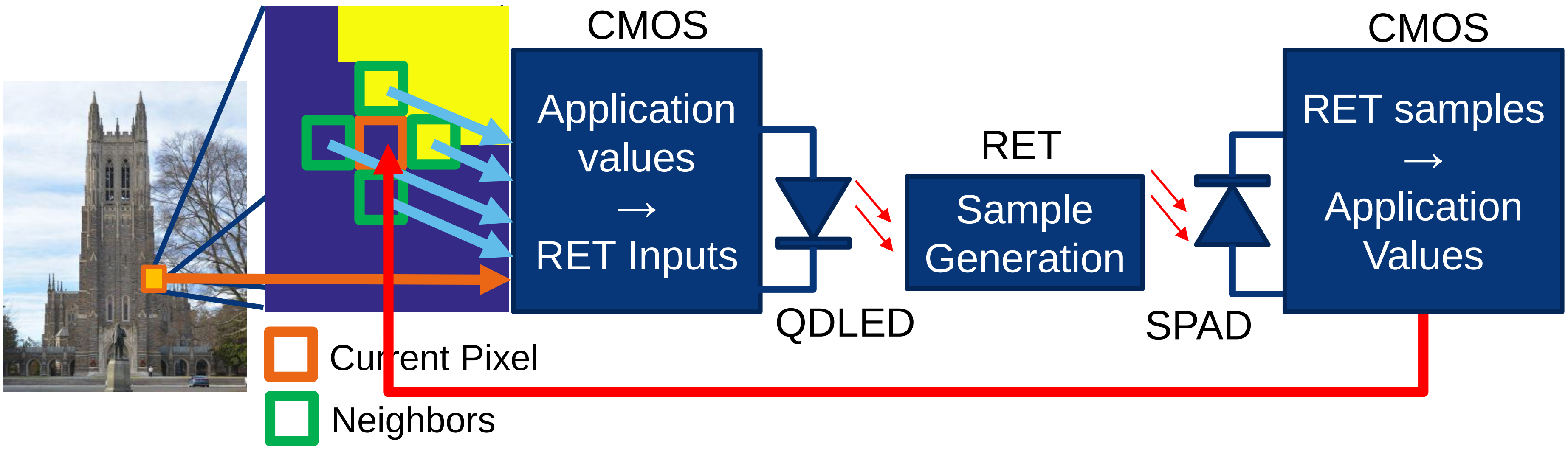
- Hybrid of CMOS + RET technology.
- Label-in label-out.
- Potentially generalize for arbitrary application interface.

RET-based Gibbs Sampling Unit (RSU-G)



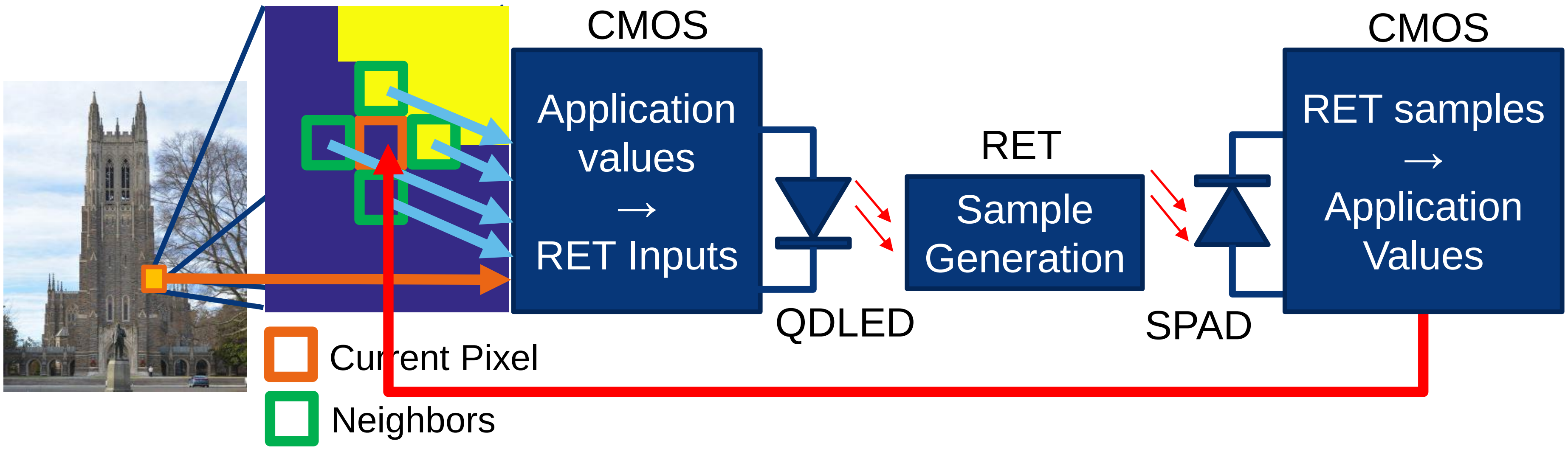
- Hybrid of CMOS + RET technology.
- Label-in label-out.
- Potentially generalize for arbitrary application interface.

RET-based Gibbs Sampling Unit (RSU-G)



- Hybrid of CMOS + RET technology.
- Label-in label-out.
- Potentially generalize for arbitrary application interface.

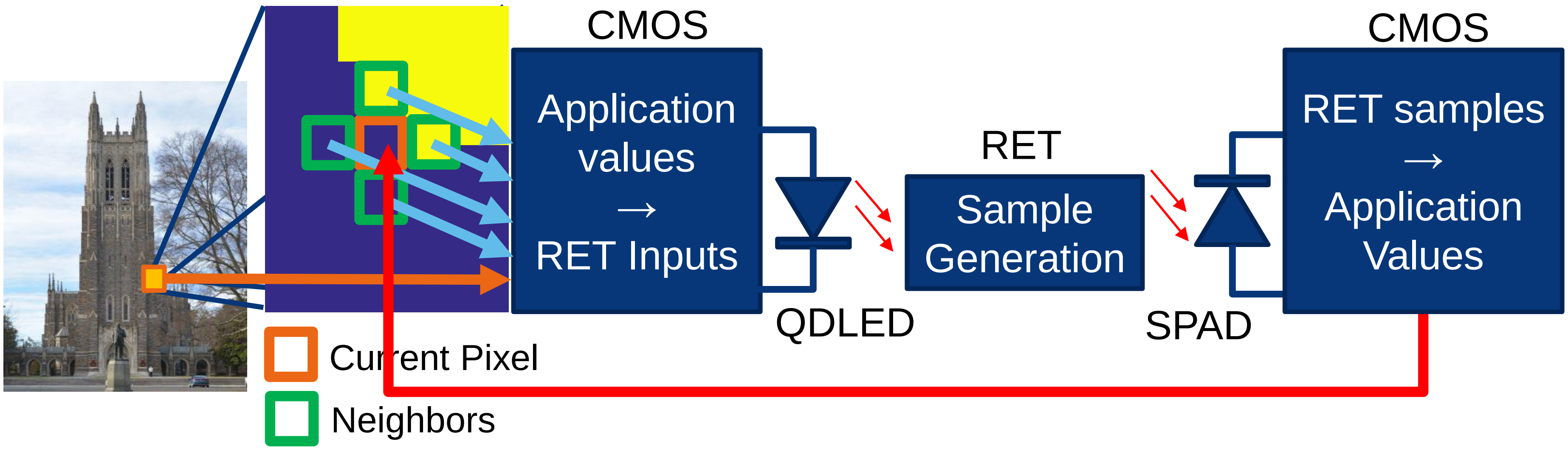
RET-based Gibbs Sampling Unit (RSU-G)



- Hybrid of CMOS + RET technology.
- Label-in label-out.
- Potentially generalize for arbitrary application interface.
- Accelerates inner loop per-pixel computation.

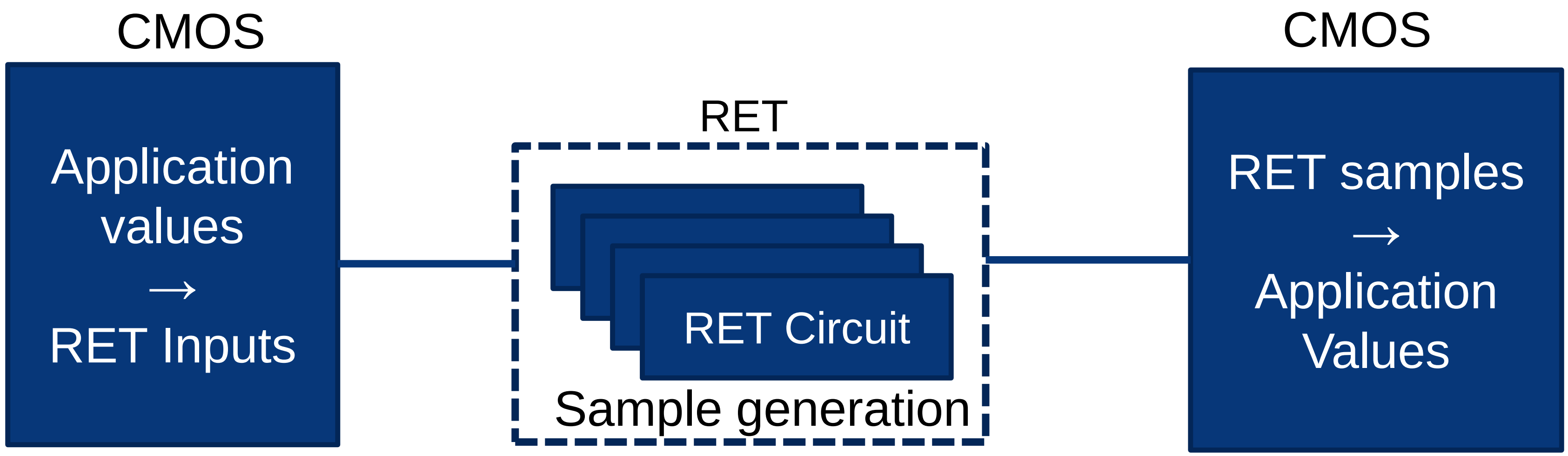
RET-based Gibbs Sampling Unit (RSU-G)

✓ Functionality, more than just generating random numbers

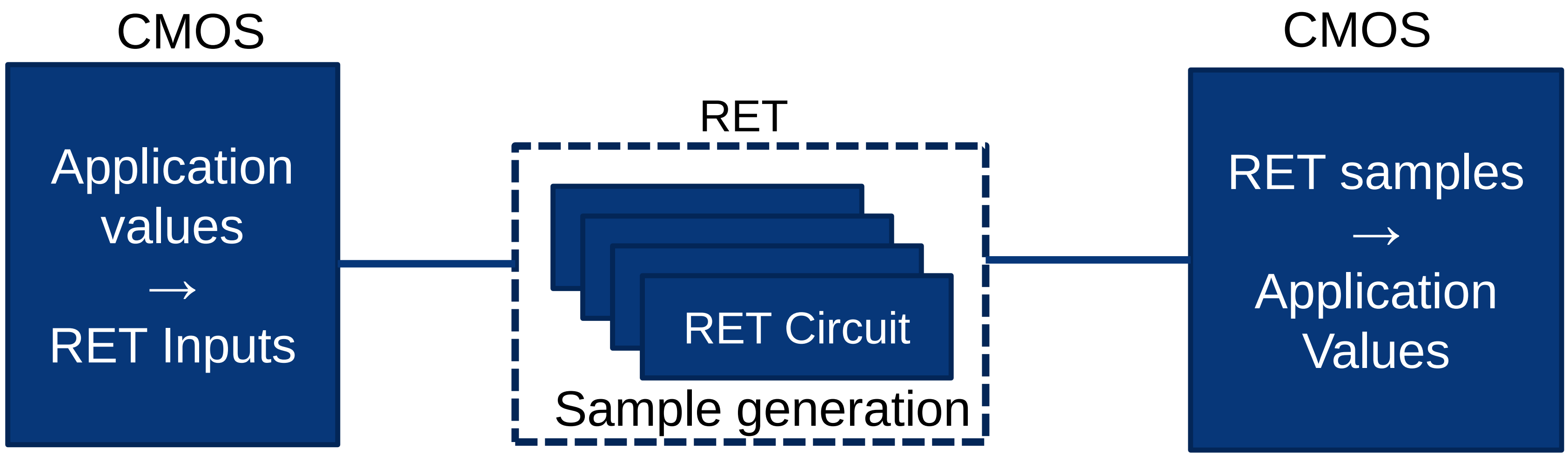


- Hybrid of CMOS + RET technology.
- Label-in label-out.
- Potentially generalize for arbitrary application interface.
- Accelerates inner loop per-pixel computation.

RSU-G Implementation

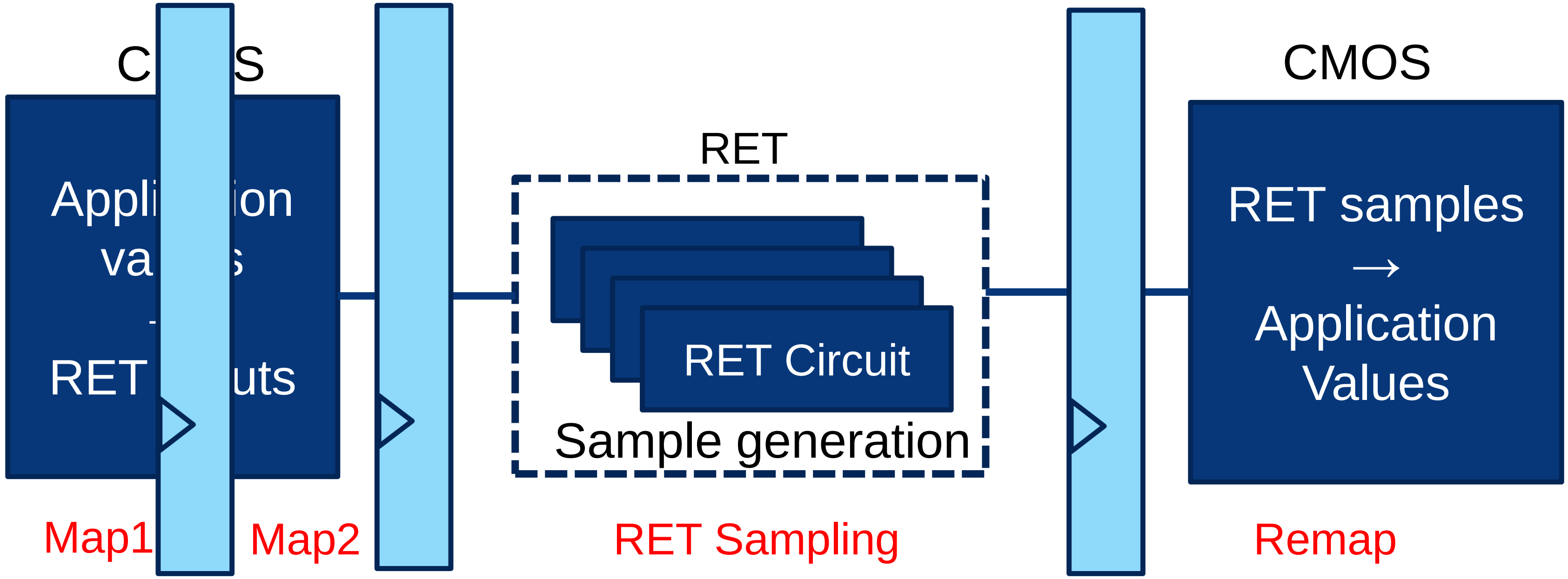


RSU-G1 Implementation



- Evaluate one possible label per cycle.

RSU-G1 Implementation

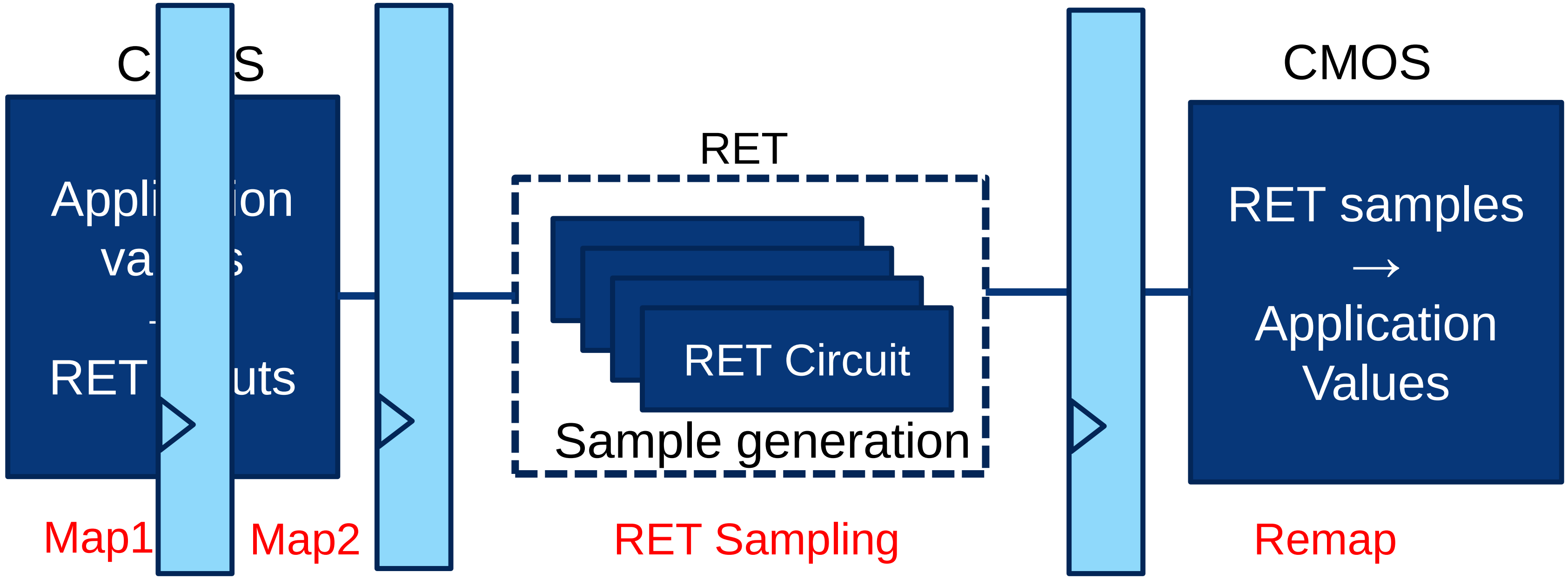


7-Stage :

Cycles	1	2	3	4	5	6	7	8
Label_1	Map1	Map2	RET Sampling				Remap	
Label_0		Map1	Map2	RET Sampling				Remap

- Evaluate one possible label per cycle.

RSU-G1 Implementation

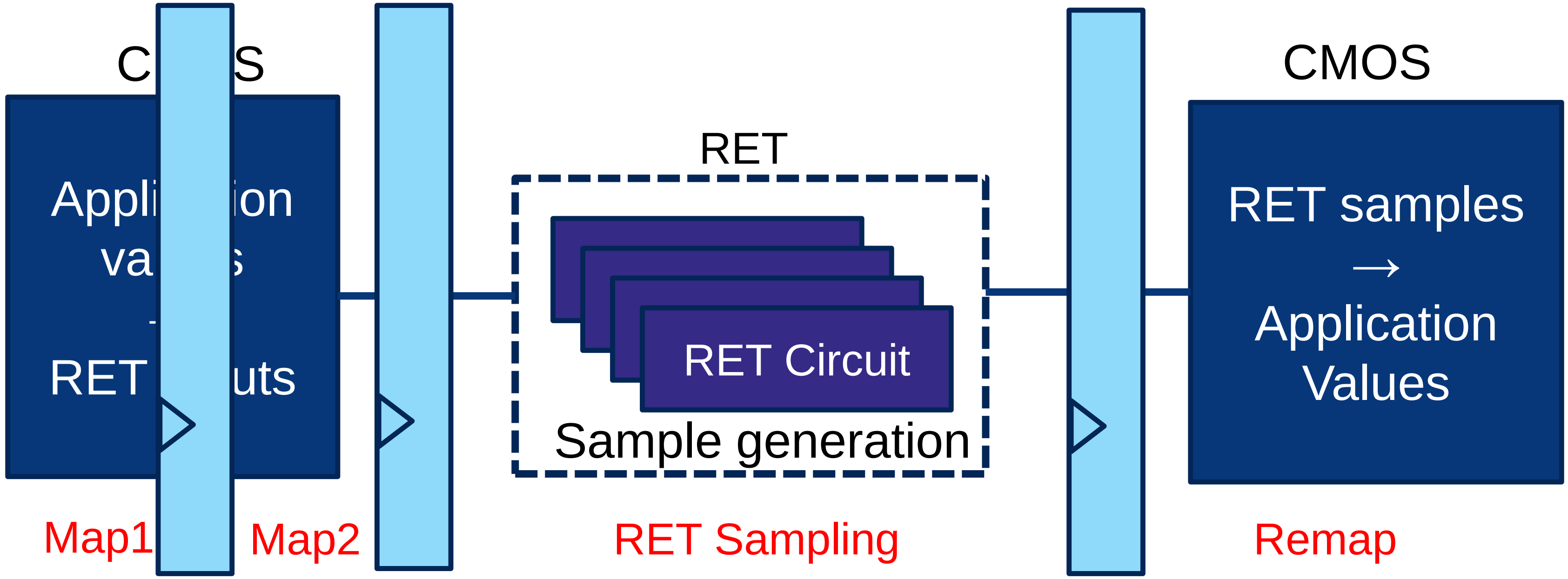


7-Stage :

Cycles	1	2	3	4	5	6	7	8
Label_1	Map1	Map2	RET Sampling				Remap	
Label_0		Map1	Map2	RET Sampling				Remap

- Evaluate one possible label per cycle.
- Time(cycles) to determine a new label = #possible labels

RSU-G1 Implementation

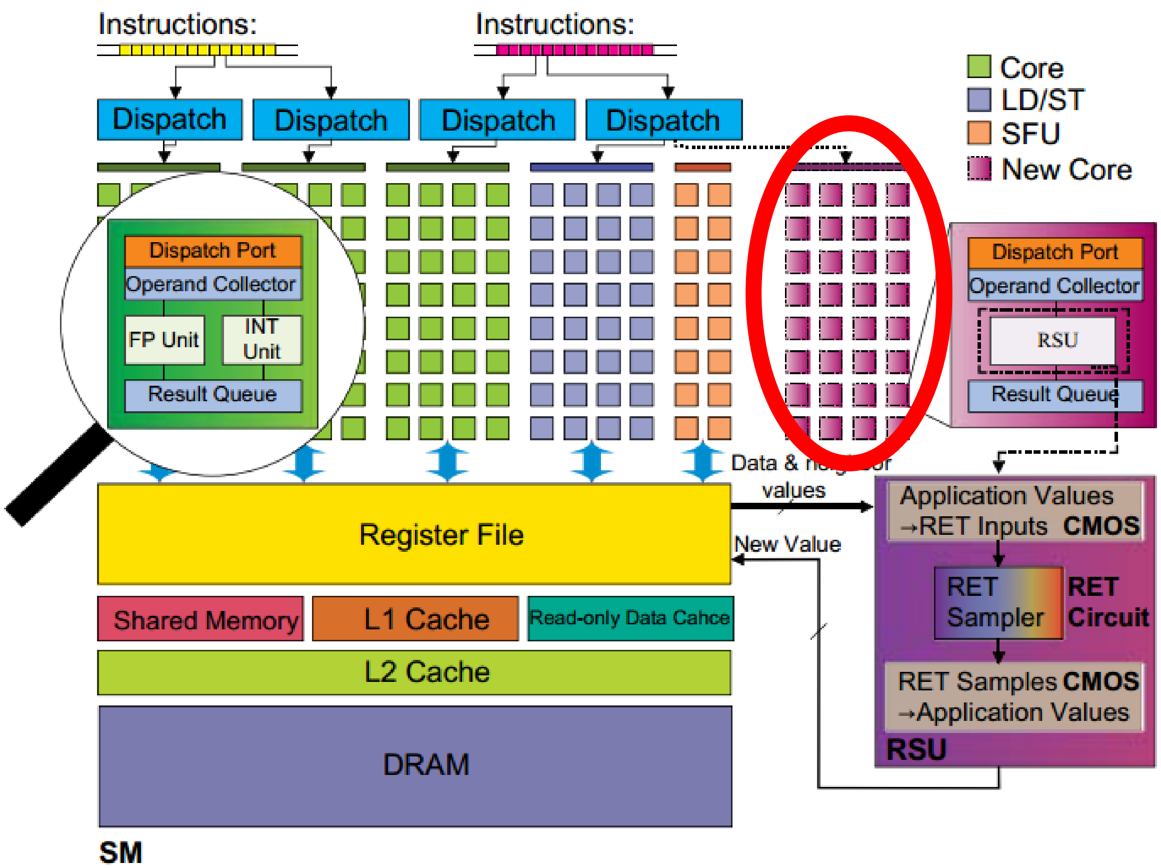


7-Stage :

Cycles	1	2	3	4	5	6	7	8
Label_1	Map1	Map2	RET Sampling				Remap	
Label_0		Map1	Map2	RET Sampling				Remap

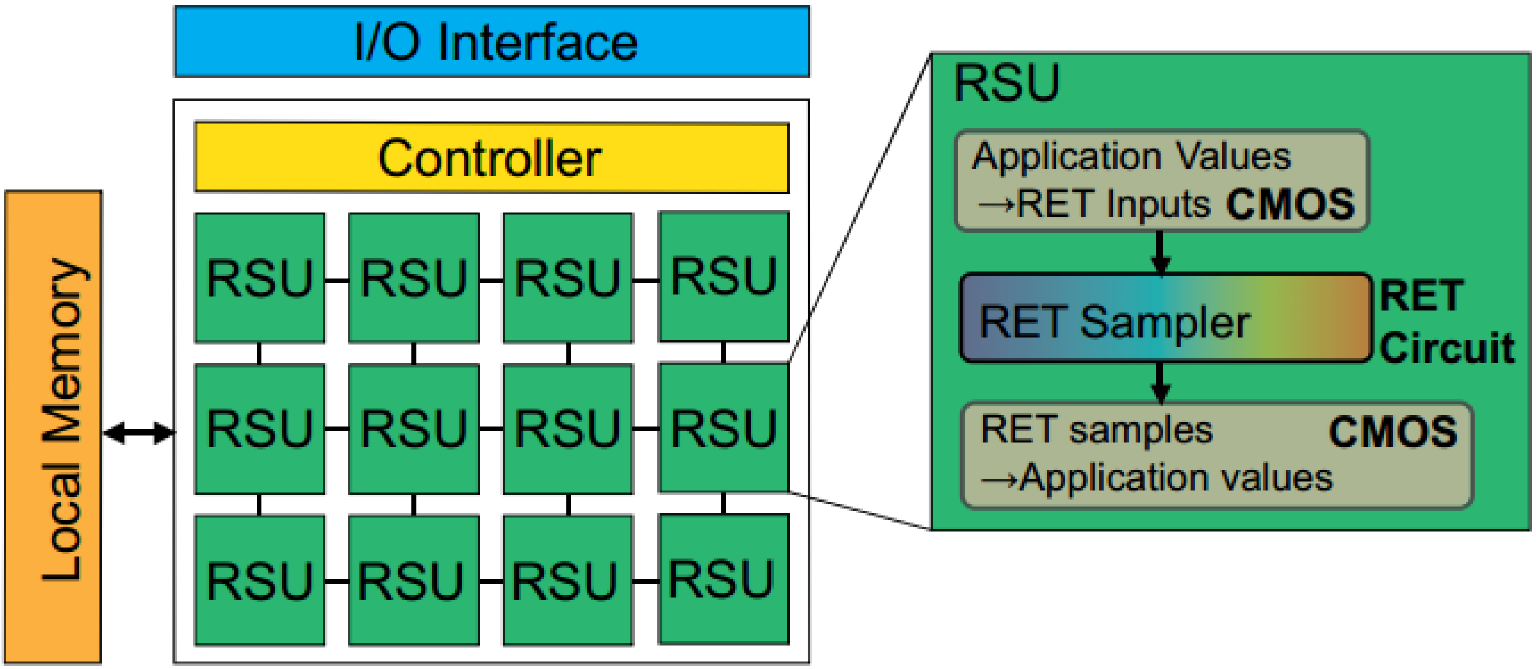
- Evaluate one possible label per cycle.
- Time(cycles) to determine a new label = #possible labels
- At 1GHz system clock, 4 RET circuit replica to avoid structural hazard.

Augmenting GPU with RSUs



- Labels are packed into 32-bit/64-bit registers.
- Modified ISA to support RSU.

Designing as a discrete accelerator

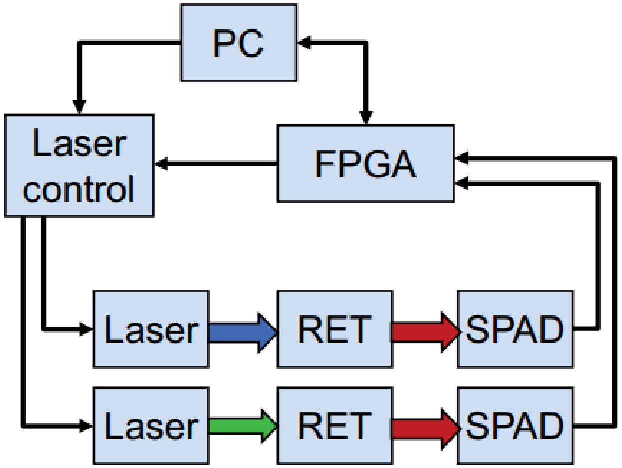


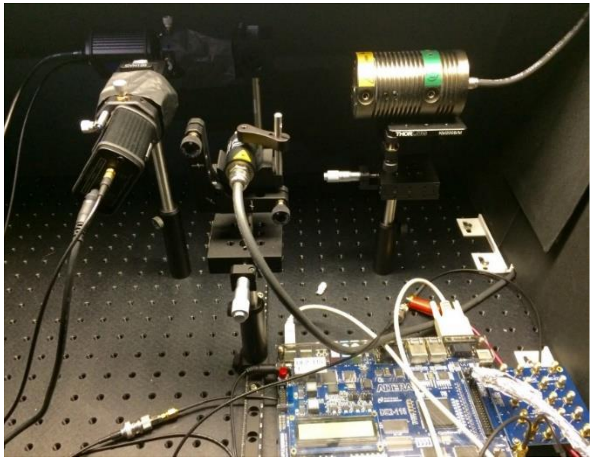
- Customized control and data movement logic.
- Highest performing approach.
- Memory bandwidth limits the upper bound.

Outline

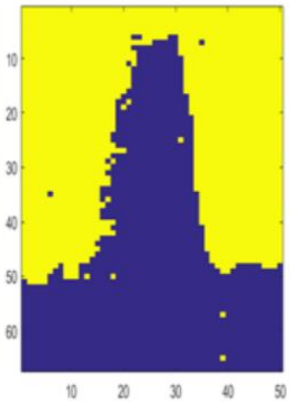
- Motivation
- Background
- RET-based Sampling Unit (RSU)
- RSU Architectures
- Evaluation
  - Macro-Scale Prototype
  - Performance (emulation)
  - Power/Area (synthesis)

Macro-scale Prototype





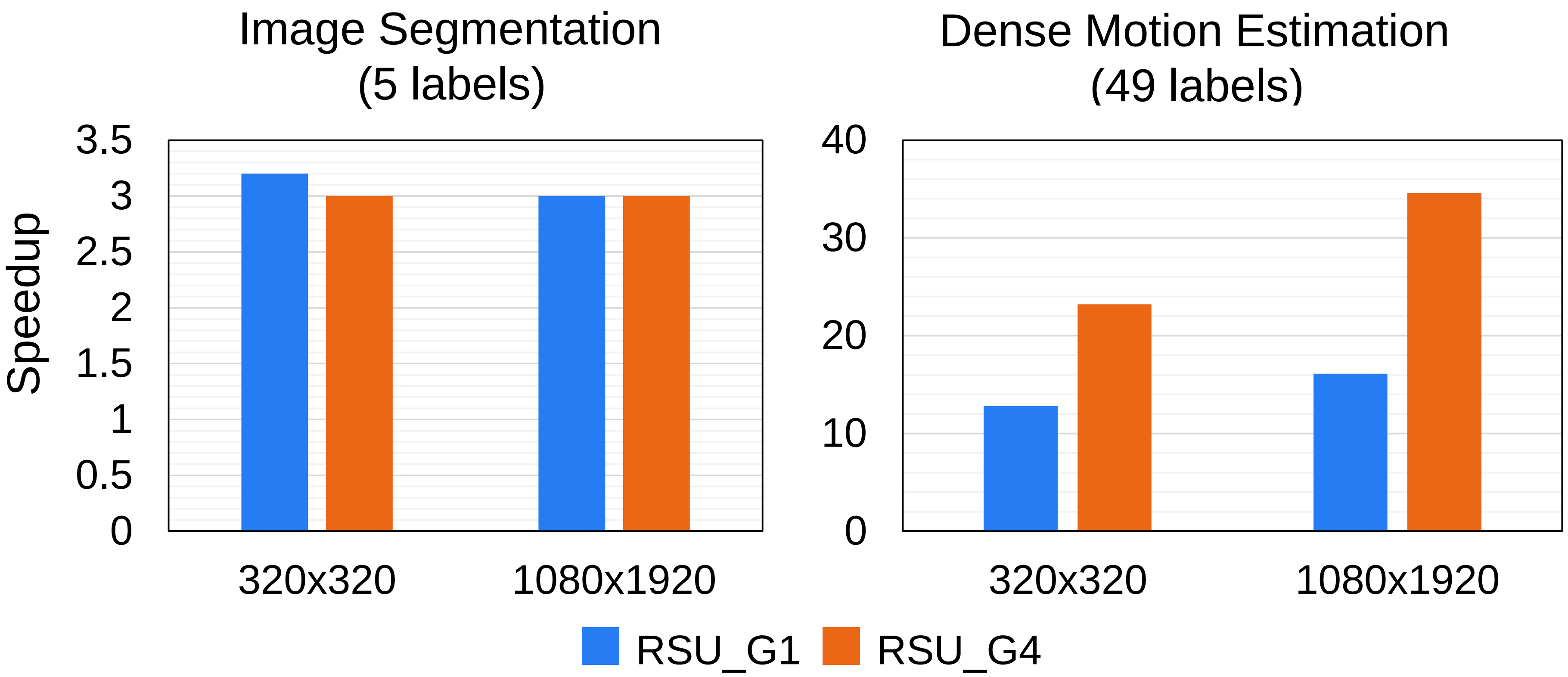
Original



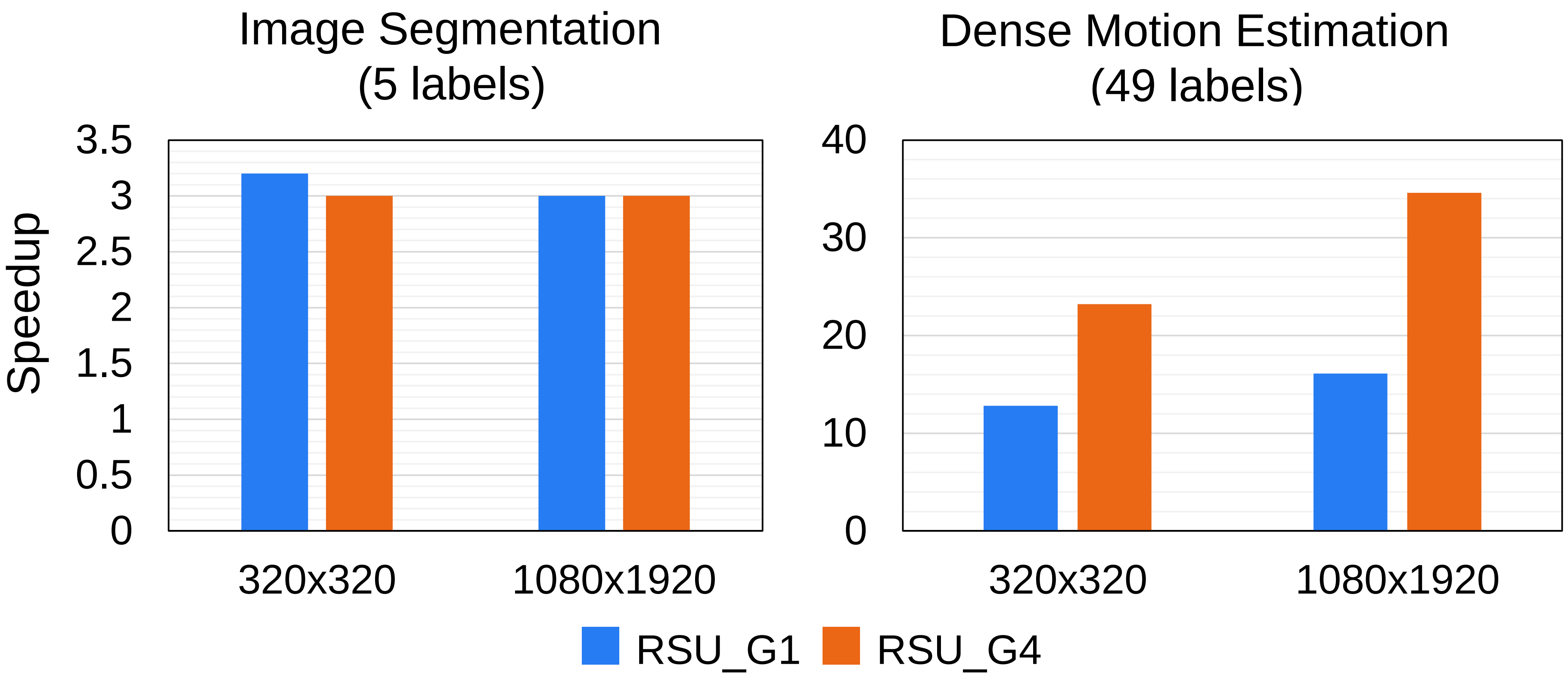
Prototype result

- First demonstration of RET-based stochastic computing.
- Demonstrate the capability to parameterize distributions.
- Demonstrate an actual application: foreground-background image segmentation.

Performance: RSU-G augmented GPU

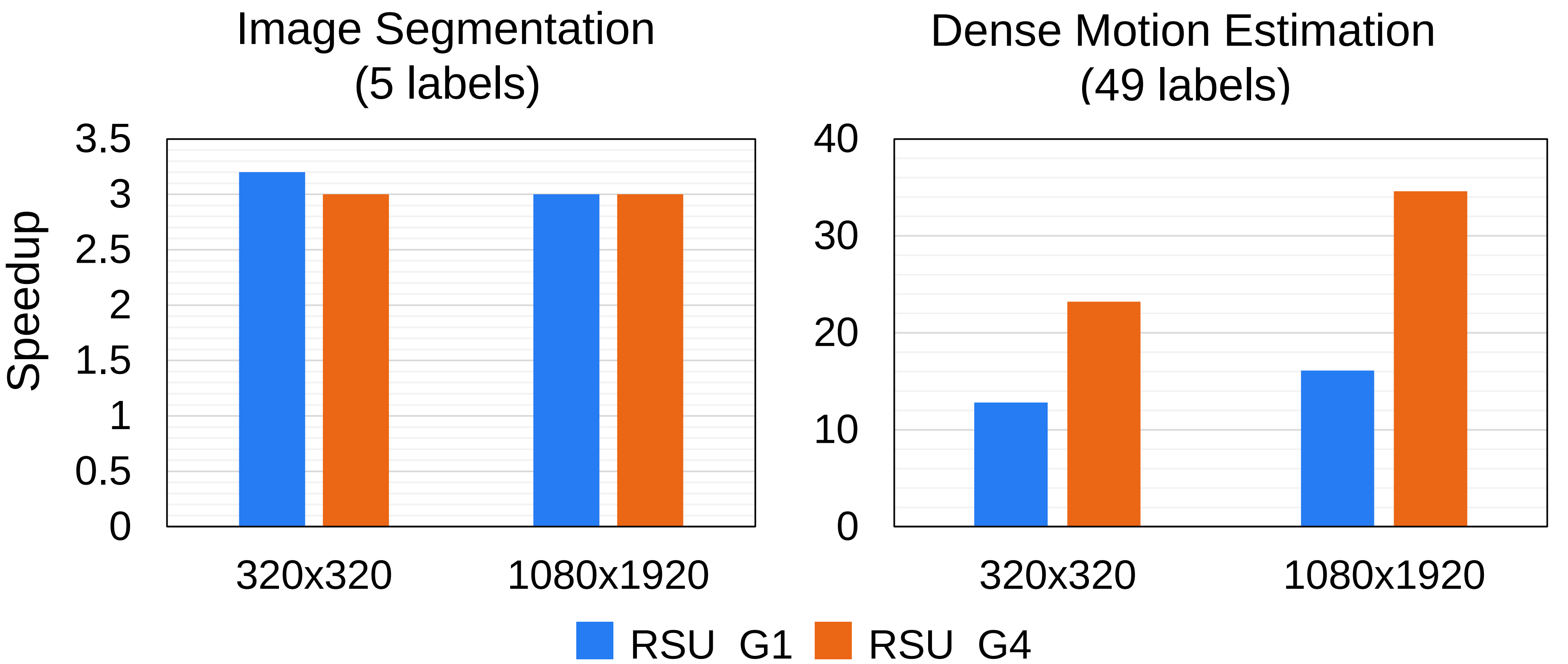


Performance: RSU-G augmented GPU



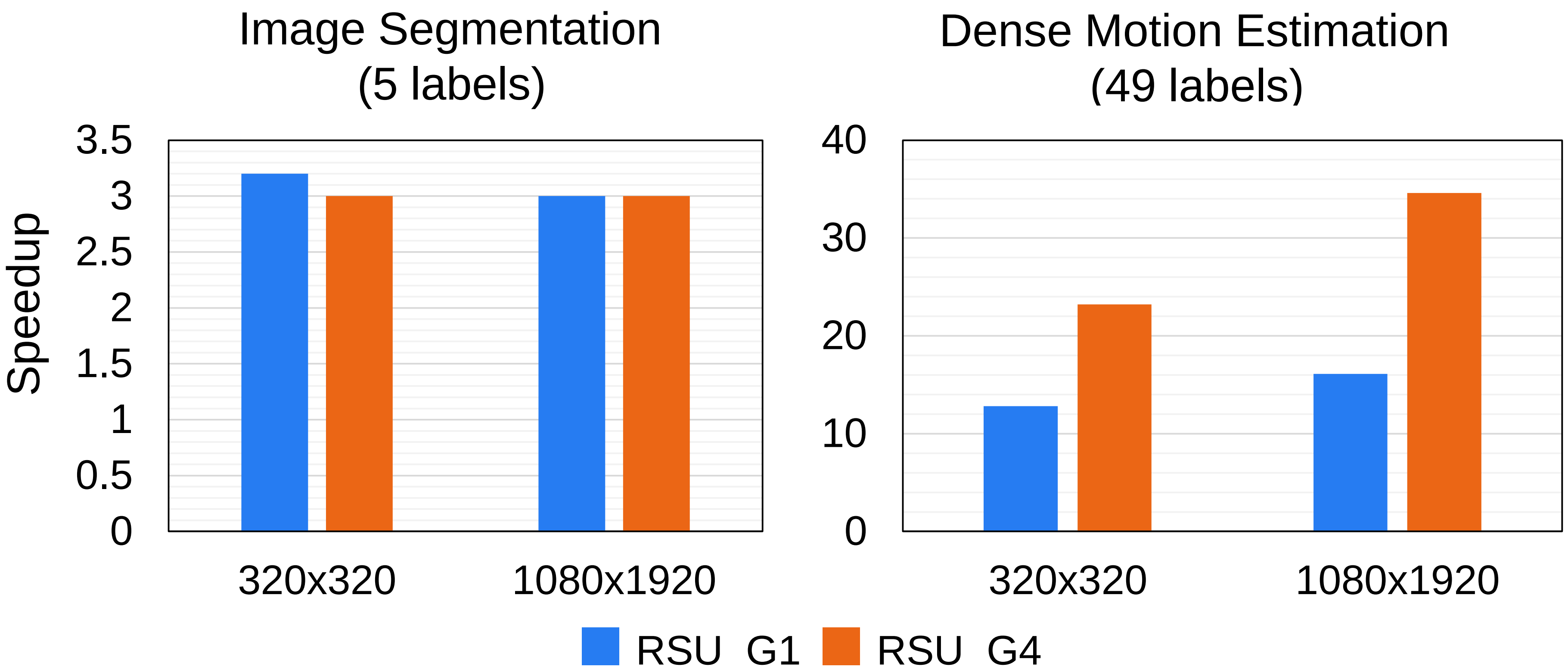
- RSU-G4 (Emulated): evaluate 4 possible labels per cycle.

Performance: RSU-G augmented GPU



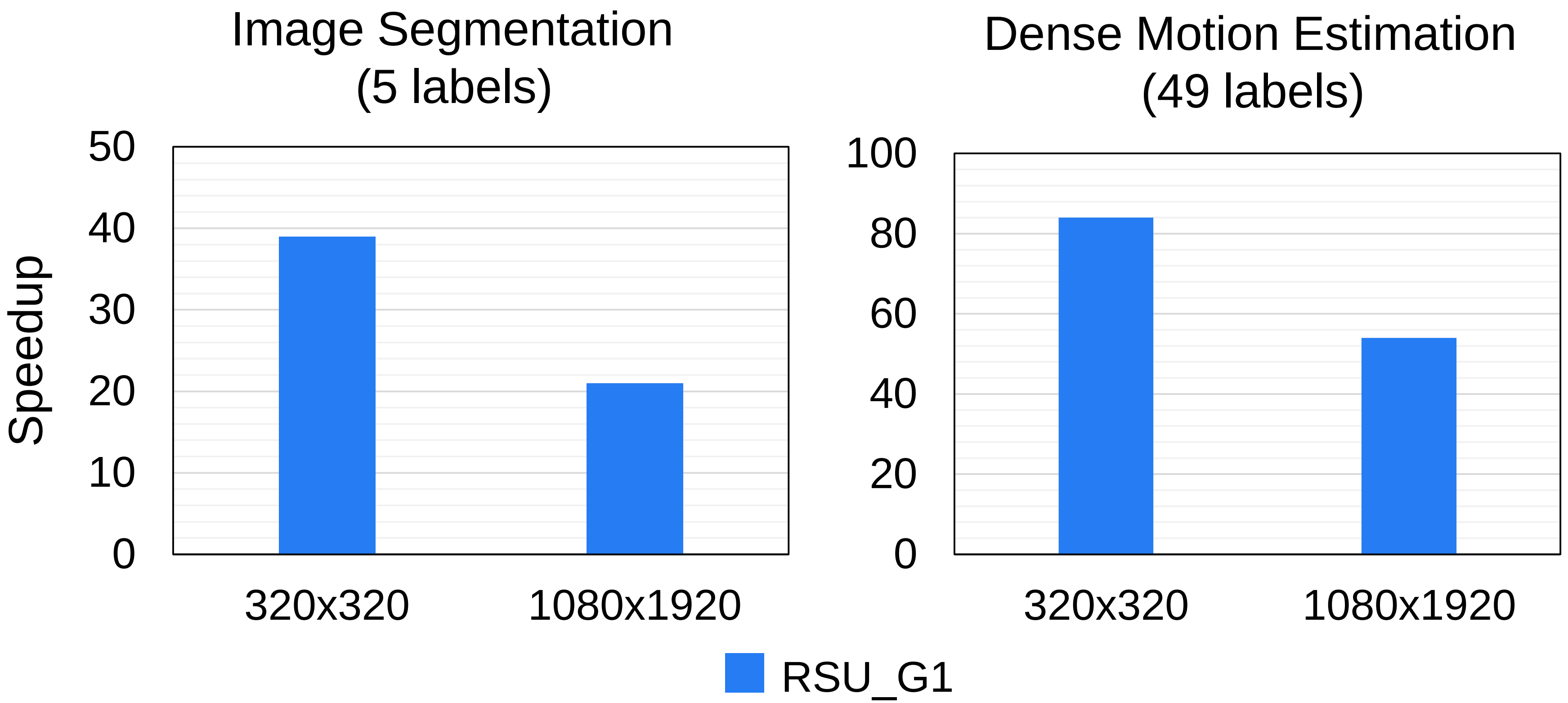
- RSU-G4 (Emulated): evaluate 4 possible labels per cycle.
- Speedup over standard GPU: 3-34

Performance: RSU-G augmented GPU



- RSU-G4 (Emulated): evaluate 4 possible labels per cycle.
- Speedup over standard GPU: 3-34
- More labels, higher speedup.

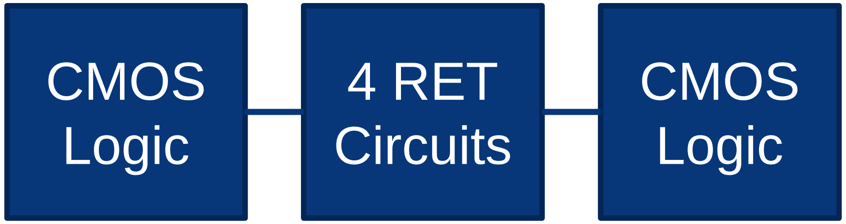
Performance: Discrete Accelerator



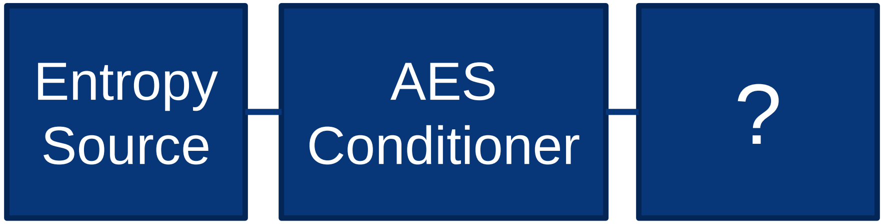
- Assuming 336GB/s DRAM BW.
- Speedup over standard GPU: 21-84

Duke Architecture

Power/Area



RSU_G1 Gibbs Sampling Unit



Intel Digital Random Number Generator (DRNG) [Hofemeier, 2012]

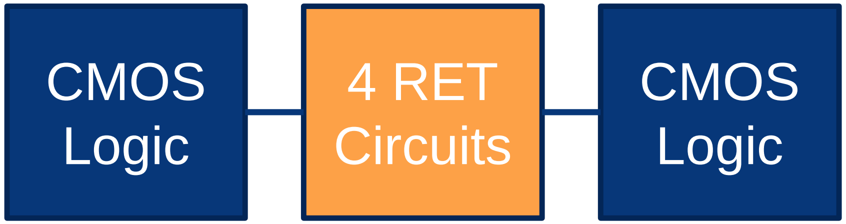
Power/Area

RSU_G1 Gibbs Sampling Unit

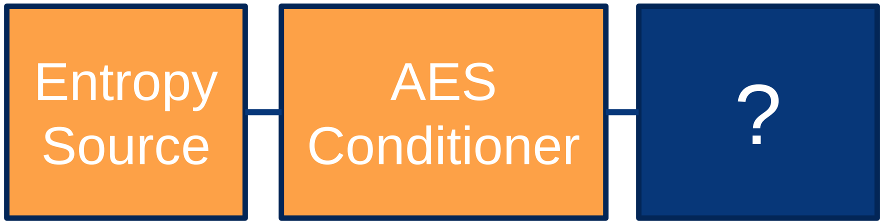
Intel Digital Random Number Generator (DRNG) [Hofemeier, 2012]

- Sampling occurs in first two Intel DRNG stages.

Power/Area



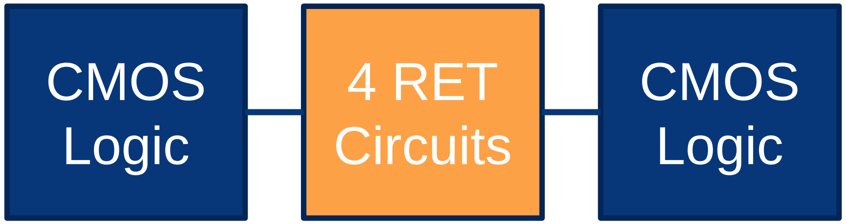
RSU_G1 Gibbs Sampling Unit



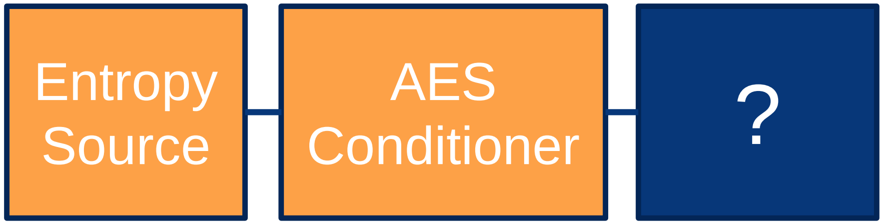
Intel Digital Random Number Generator (DRNG) [Hofemeier, 2012]

- Sampling occurs in first two Intel DRNG stages.
- Intel DRNG: higher throughput, but not easily parameterized.

Power/Area



RSU_G1 Gibbs Sampling Unit



Intel Digital Random Number Generator (DRNG) [Hofemeier, 2012]

15nm technology	4 RET Circuits	AES-256 conditioner
Power	0.16 mW	1.20 mW
Area	1600 μm^2	1570 μm^2

- Sampling occurs in first two Intel DRNG stages.
- Intel DRNG: higher throughput, but not easily parameterized.
- 4 RET Circuits: consume 13% power, occupy similar area as AES256.

Power/Area

RSU_G1 Gibbs Sampling Unit

Intel Digital Random Number Generator (DRNG) [Hofemeier, 2012]

15nm technology	4 RET Circuits	AES-256 conditioner
Power	0.16 mW	1.20 mW
Area	1600 μm^2	1570 μm^2

- Sampling occurs in first two Intel DRNG stages.
- Intel DRNG: higher throughput, but not easily parameterized.
- 4 RET Circuits: consume 13% power, occupy similar area as AES256.
- Arbitrary distributions: add negligible power/area on RSU-G.

Conclusion

Conclusion

- RSU: CMOS+RET to support probabilistic computing.
- Implement RSU-G for Gibbs Sampling acceleration.
- First experimental demonstration of RSU-G.

Conclusion

- RSU: CMOS+RET to support probabilistic computing.
- Implement RSU-G for Gibbs Sampling acceleration.
- First experimental demonstration of RSU-G.
- Speedup over standard GPU:
  - Augmented GPU 3-34
  - Discrete Accelerator 21-84

Conclusion

- RSU: CMOS+RET to support probabilistic computing.
- Implement RSU-G for Gibbs Sampling acceleration.
- First experimental demonstration of RSU-G.
- Speedup over standard GPU:
  - Augmented GPU 3-34
  - Discrete Accelerator 21-84
- Achieve desirable properties:

✓ High quality
✓ High flexibility
✓ Low complexity
✓ High functionality

Conclusion

- RSU: CMOS+RET to support probabilistic computing.
- Implement RSU-G for Gibbs Sampling acceleration.
- First experimental demonstration of RSU-G.
- Speedup over standard GPU:
  - Augmented GPU 3-34
  - Discrete Accelerator 21-84
- Achieve desirable properties:

✓ High quality
✓ High flexibility
✓ Low complexity
✓ High functionality

- Ongoing work:
  - full programmability
  - integration with CMOS
  - high-order MRF
  - longevity

Thank you

- Q&A