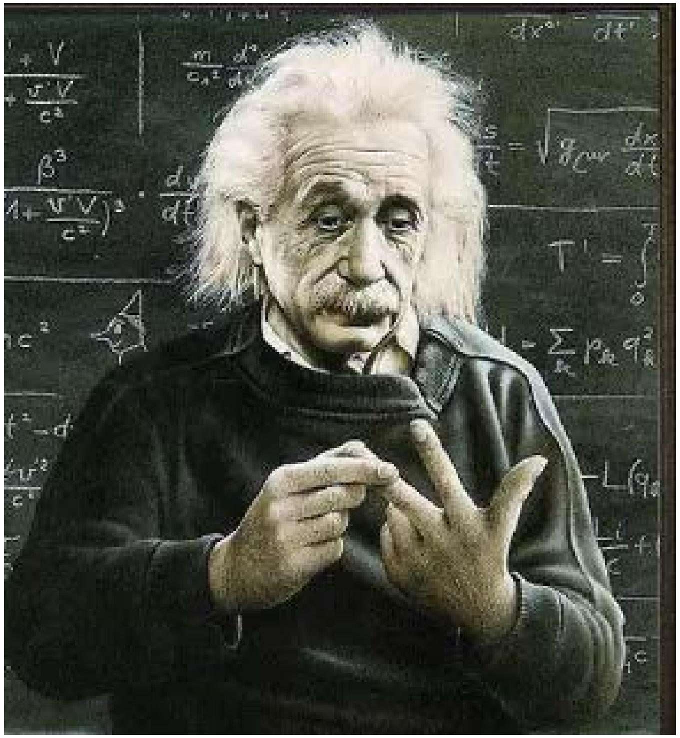
Unary Computing:
The Stochastic Circuit Approach

John P. Hayes

Workshop on Unary Computing
Phoenix, Arizona
June 22, 2019

COMPUTER SCIENCE AND ENGINEERING
UNIVERSITY OF MICHIGAN

Unary Computing



Unary Computing

- Counting to five

1 11 111 1111 11111

- To twenty

1111111111 1111111111

- To one thousand

1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111
1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111
1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111
1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111
1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111
1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111
1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111
1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111
1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111
1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111 1111111111

Unary Computing

- Unary computing has ancient origins, and is frequently reinvented in different forms.
- A good starting point is the paper “Unary Processing“ by W.J. Poppelbaum et al. in *Advances in Computers*, 1987
- Their definition of **unary**: Any representation of information in which all digits have the same weight.
- They classify unary processors into **two major forms**:

Deterministic	**Stochastic**
Compact representation	Sparse representation
Rapid calculation	Slower calculation
Complex circuitry	Simple circuitry
Low noise immunity	High noise immunity

Stochastic Computing

- **Primary characteristics**:
  - Sparse representation
  - Slower calculation
  - Simple circuitry
  - High noise immunity
- **Secondary characteristics**:
  - Lower accuracy
  - Lower power
  - Massive parallelism
  - Biological compatibility
  - Complex behavior

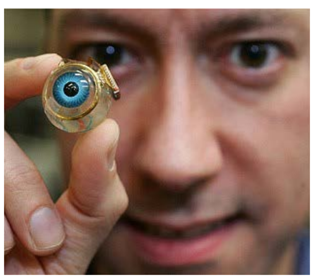
Motivating Application: Retinal Implant

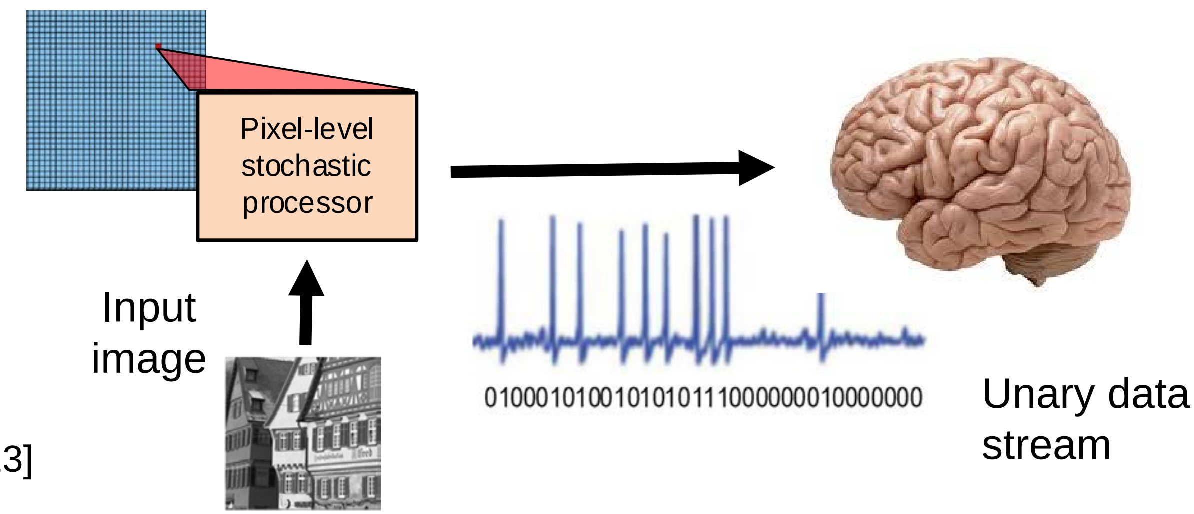
- **Goal**

  Chip that can be implanted in the human eye to replace the functions of a damaged retina

- **Structure and function**

  Array of pixel processors that sense and process light images, and map them to electrical pulse streams that can be injected into the optic nerve and sent to the brain





[Alaghi et al., *DAC* 2013]

Retinal Implant (contd)

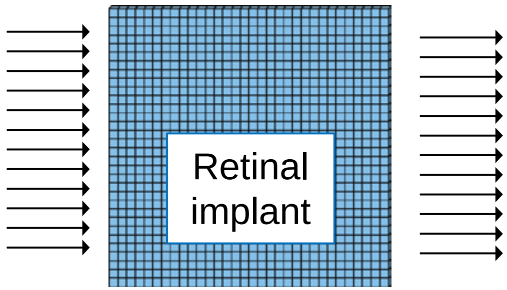
- **Requirements**

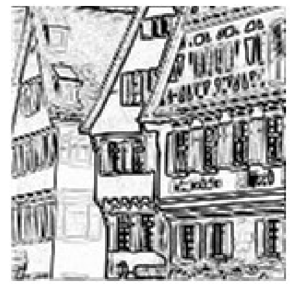
  Massive parallelism
  Tiny processors
  Tiny power dissipation
  Biological compatibility

- Stochastic computing appears uniquely qualified to meet all these requirements

Retinal Implant (contd)

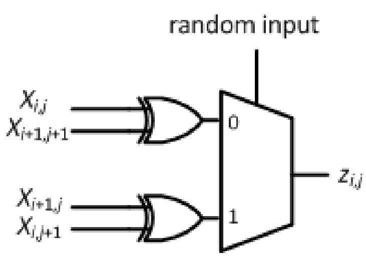
Input image





Edges detected by array of stochastic processing elements

Stochastic edge detector element



Edge detector output after: 4 32 256
Clock cycles

So What is Stochastic Computing?

- In a nutshell, SC is **computing with (pseudo) random bit-streams**, i.e. unary sequences, that represent probabilities
- **Advantages**
  - Small size, low power, and high error tolerance
  - Use (or not) of conventional logic technologies like CMOS
  - Progressive precision
  - Bio-compatibility
  - Randomness
- **Disadvantages**
  - Low accuracy and long computing time
  - Special design requirements
  - Complex accuracy/time/cost trade-offs
  - Randomness
- **Our Motivation**
  - SC is well suited to neuromorphic and AI applications

Related Technologies

- **Neuromorphic computing**

Spike train of neural impulses:

Stochastic number: 01000101001010101110000000010000000

- **Quantum computing**

Analog and digital aspects: $|\Psi\rangle = c_0|0\rangle + c_1|1\rangle$

Signal states (qubits) are probabilistic

c_k^2 = probability of $\Psi = k$

Stochastic Numbers (SNs)

- An SN is a (pseudo) random bit-stream **X** in which each bit has a probability *X* of being 1.
- **X**'s **numerical value** is $\widehat{X}$= (no. of 1s in **X**)/(length *N* of **X**)
- Examples of SNs:

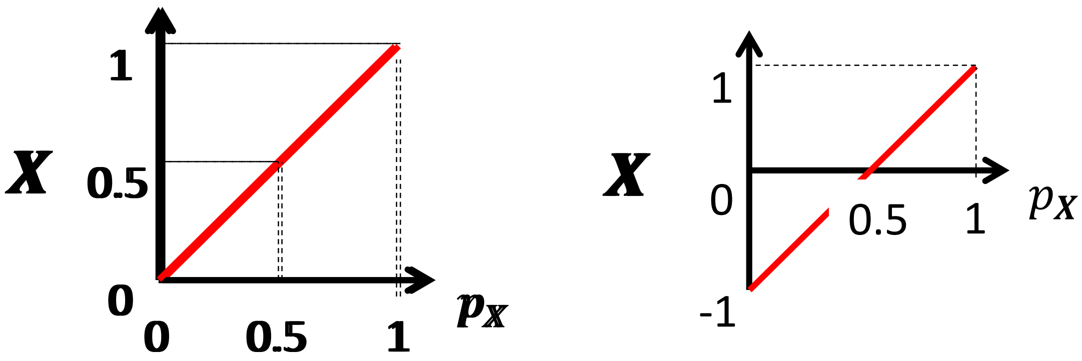
Target (exact) value	Bit-pattern	Length	Measured (estimated) value
$X = 0.5$	**X** = 1010	$N = 4$	$\hat{X} = 2/4 = 0.5$
$X = 0.5$	**X** = 01010110	$N = 8$	$\hat{X} = 4/8 = 0.5$
$X = 0.75$	**X** = 110110111101	$N = 12$	$\hat{X} = 8/12 = 0.75$
$X = 0.75$	**X** = 1101101	$N = 8$	$\hat{X} = 5/8 = 0.625$

16% inaccuracy

SN Formats: Unipolar and Bipolar

- If an SN **X**'s value is interpreted as $X = p_X$, only positive numbers are represented. This is ***unipolar*** format.
- If **X**'s value is interpreted as $2p_X - 1$, then positive and negative numbers can be represented. This is ***bipolar*** format.

Unipolar: $X = p_X$ Bipolar: $X = 2p_X - 1$



- Also data must be *scaled* to lie in the unit interval [0,1]

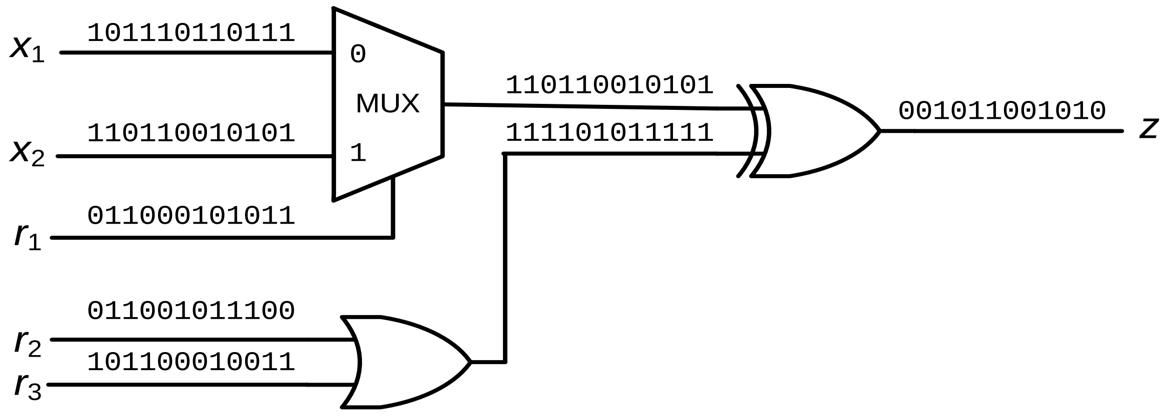
Application Areas for SC

- Analog and hybrid analog-digital tasks such as control systems and small neural networks
  - Early work from 1960s – 2000s
- Several SC applications have been investigated, but only a few have been implemented in hardware
- Recent breakthroughs:
  - Decoding chips for low density parity check (LDPC) error-correcting codes [Naderi et al. 2011]
  - Image-processing circuits [Li & Lilja 2011], [Alaghi et al. 2013]
  - General synthesis techniques for stochastic circuits: [Qian et al. 2011], [Alaghi & Hayes 2012]
  - Applications to (hybrid) deep neural networks
  - Applications enabled by more accurate SC methods

A Little History

Dates	Topics	References
1967 - 79	Definition and basic concepts of SC	Gaines 1967 Poppelbaum 1967
1980 -1999	Advances in the theory of SC Small applications of SC, e.g. to controller design	Jeavons et al. 1994 Toral et al. 1990
2000 - present	Application to decoding of LDPC error-correcting codes General circuit design methods Application to image processing, neural nets, etc. Advances in the theory of SC	Gaudet & Rapley 2003 Qian et al. 2011 Alaghi & Hayes 2012 (Many)

Two Faces of a Stochastic Circuit



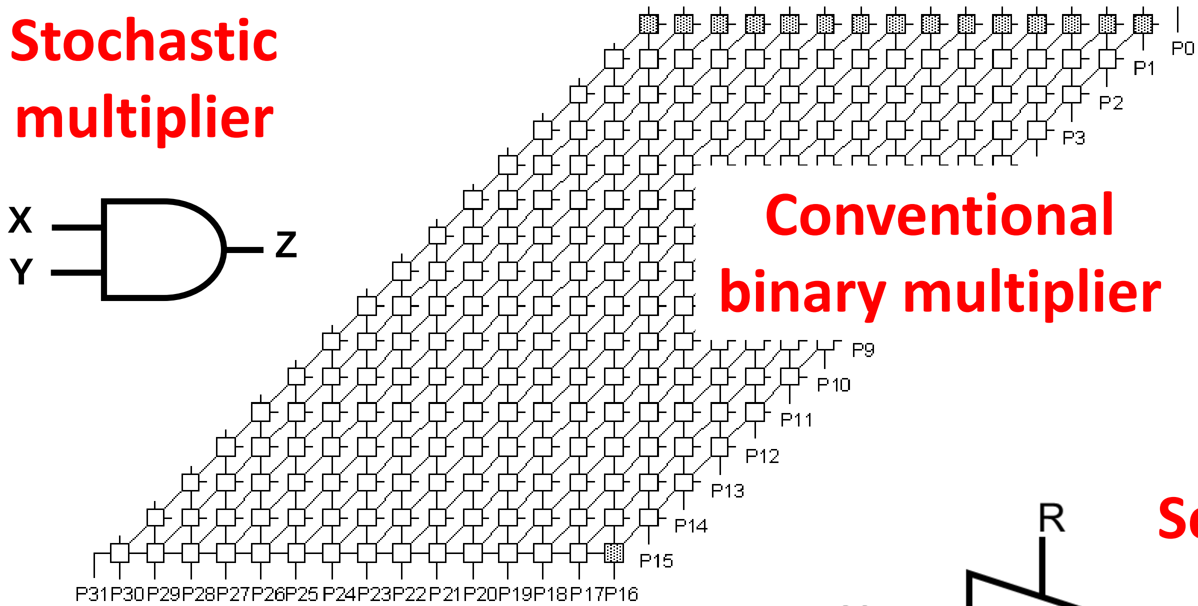
- A logic circuit *C* in which a number *X* is encoded by a **randomized** bit-stream **X** whose numerical value depends on bit probabilities
- The design target is some **arithmetic function**, in this case,
  $$F(X_1, X_2) = -0.25 \times (X_1 + X_2)$$
- *F* depends in a non-obvious way on *C*'s **logic function**, in this case,
  $$f(x_1, x_2, r_1, r_2, r_3) = (x_1 \wedge \bar{r}_1 \vee x_2 \wedge r_1) \oplus (r_2 \vee r_3)$$
  and on the input (bipolar) number values, and ancillary random constants

Stochastic Circuits (contd)

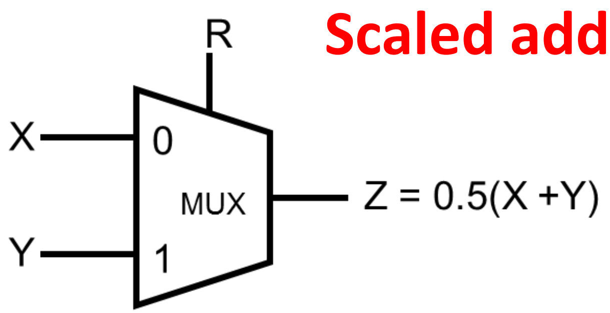
- Key advantages: low hardware cost and power

Stochastic multiplier

Conventional binary multiplier

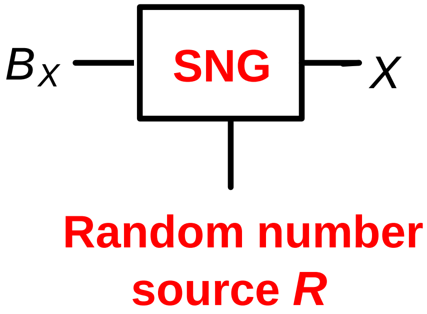


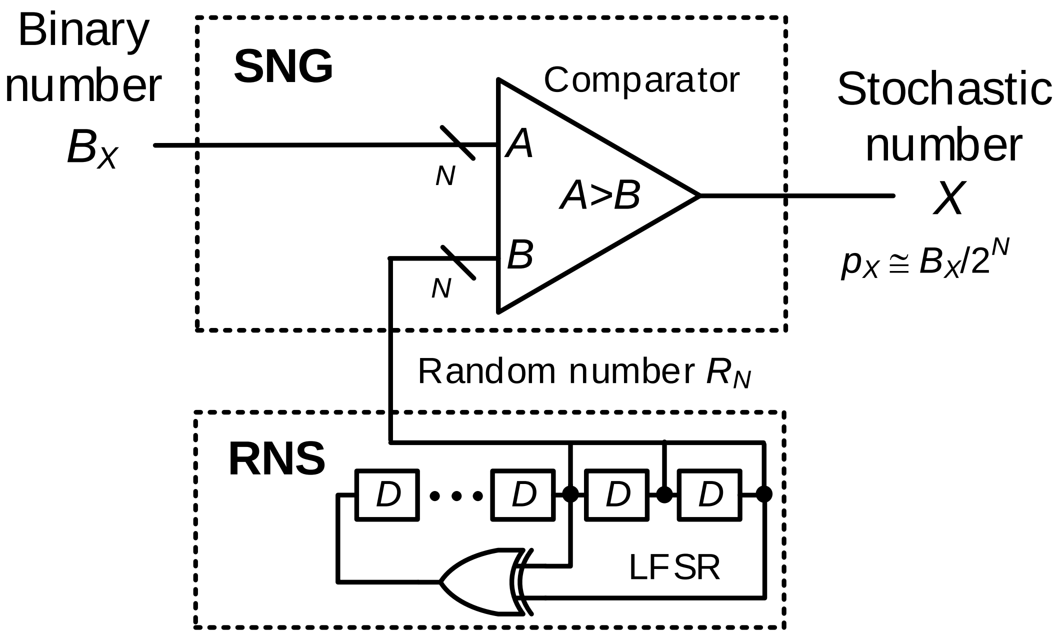
Scaled adder



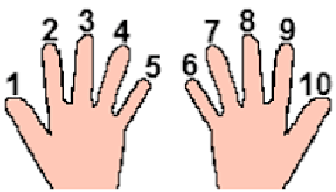
Data Conversion Circuits

- Binary to stochastic conversion:



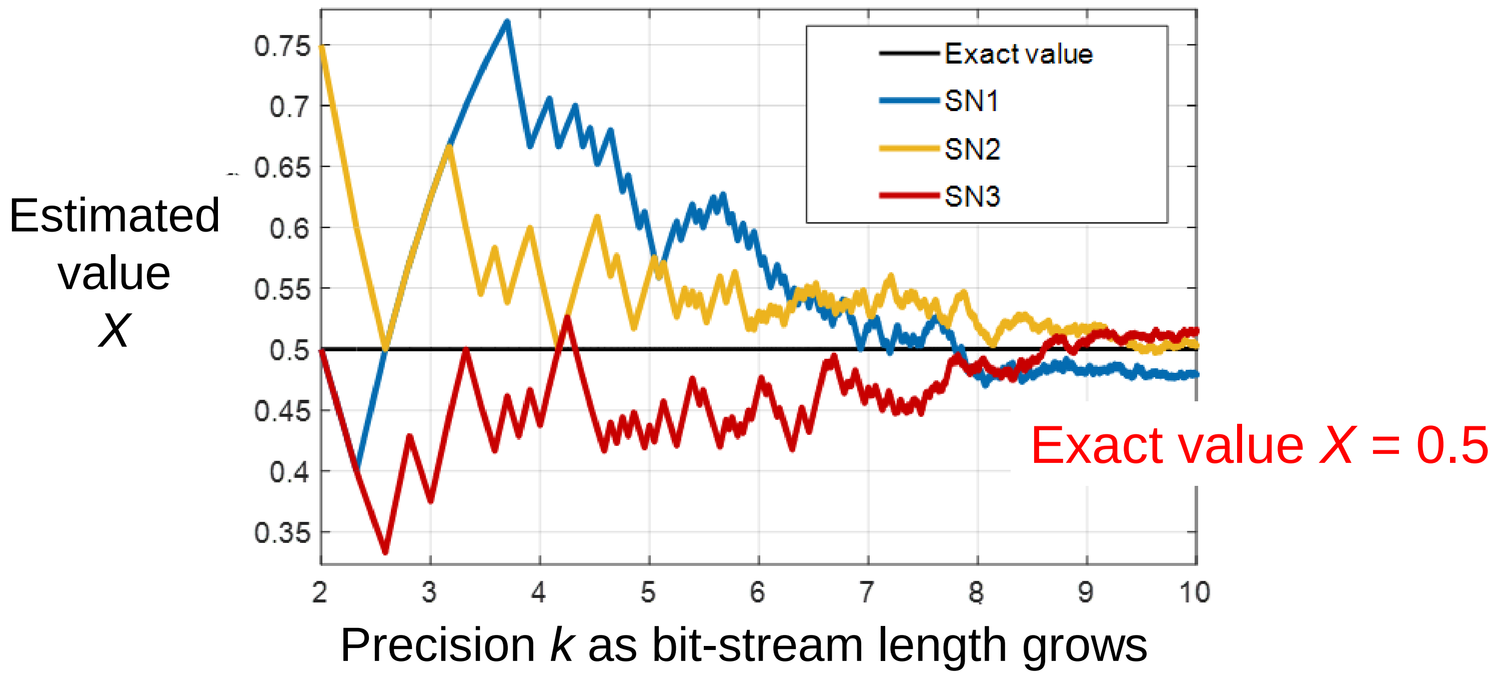


- Stochastic to binary conversion:

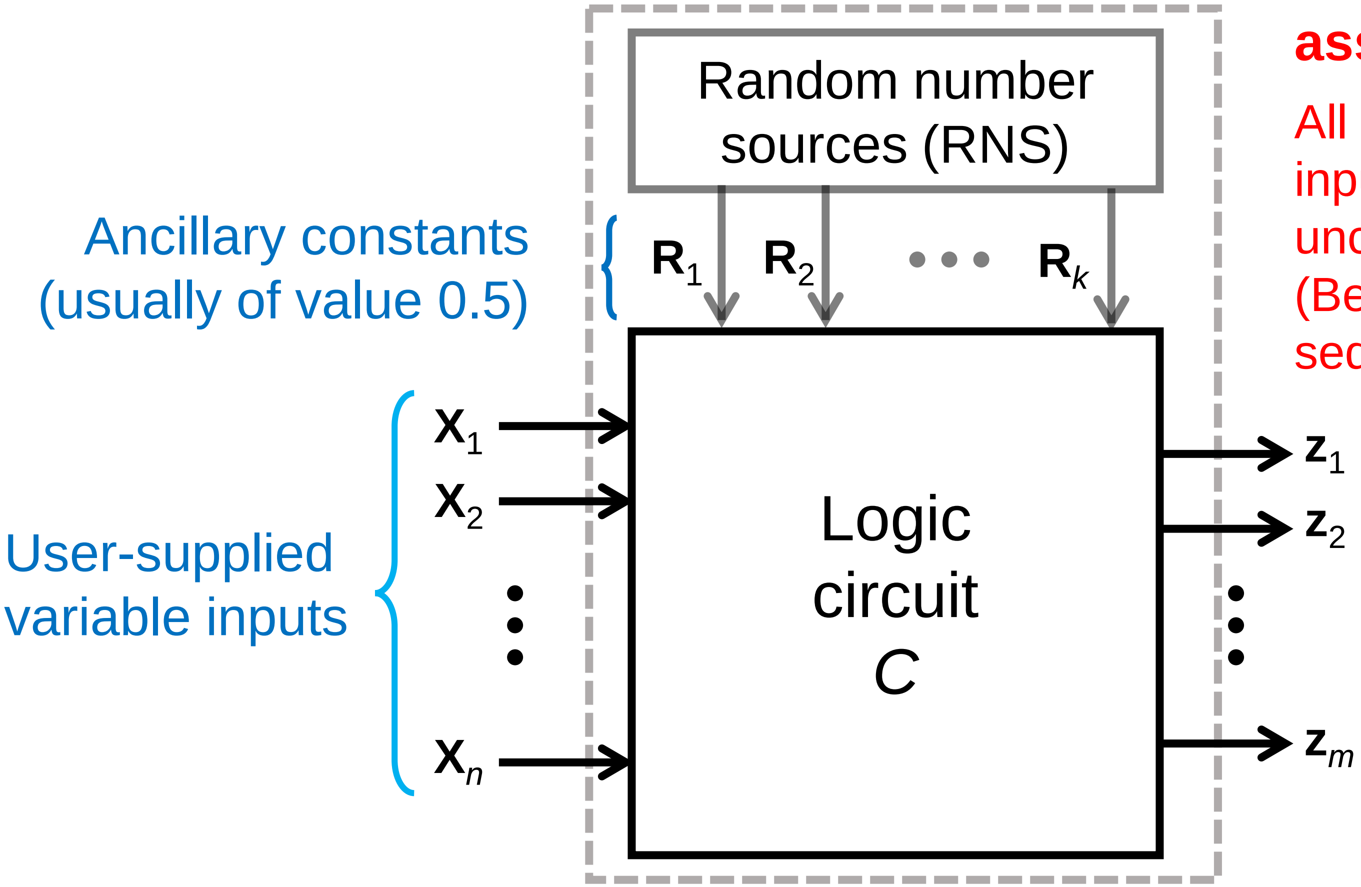


Accuracy of SNs

- Longer bit-streams tend to provide better value estimates
- But length grows exponentially with desired precision



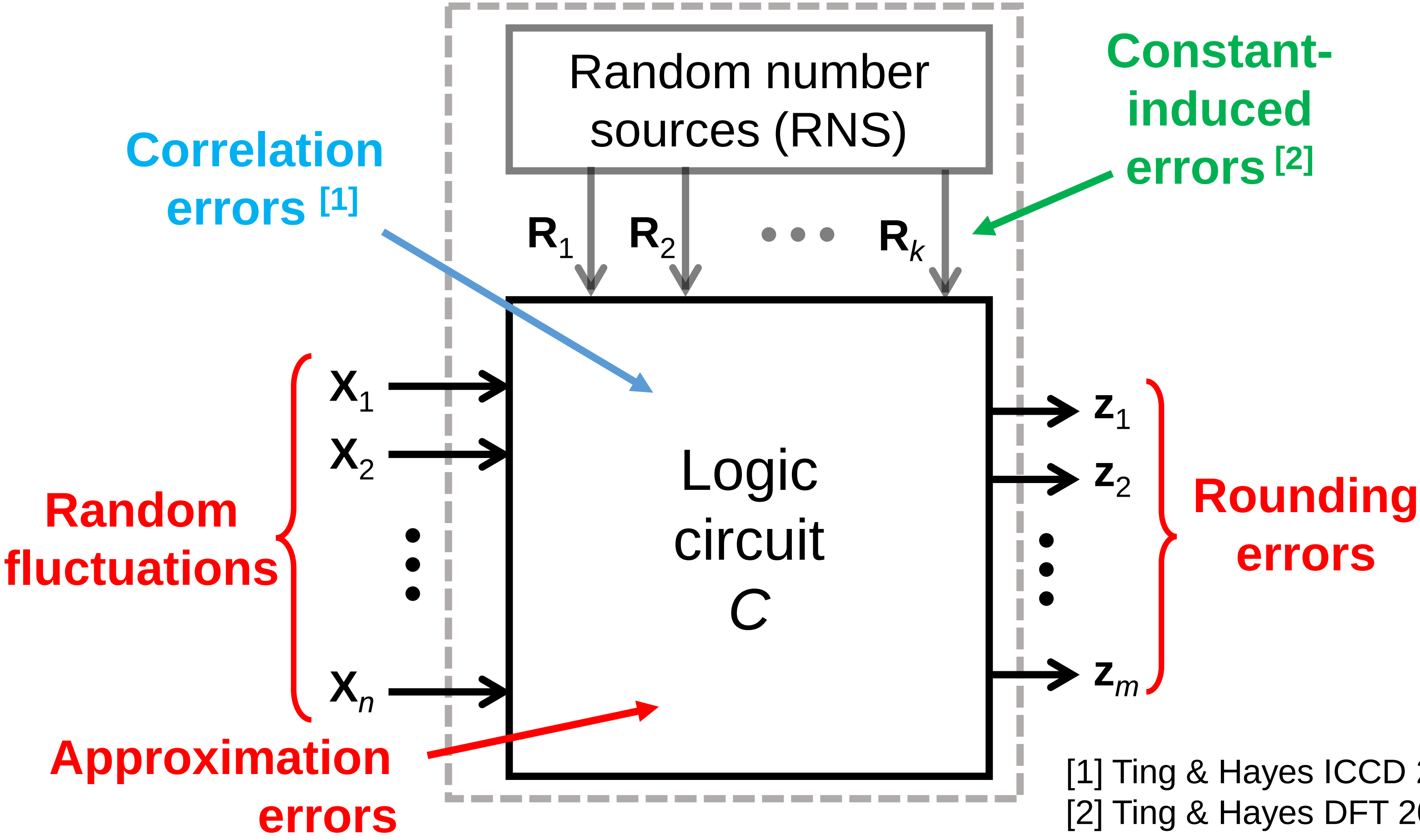
Stochastic Circuit Structure



Standard assumption

All $\mathbf{R}_i$ and $\mathbf{X}_j$ inputs are uncorrelated (Bernoulli sequences)

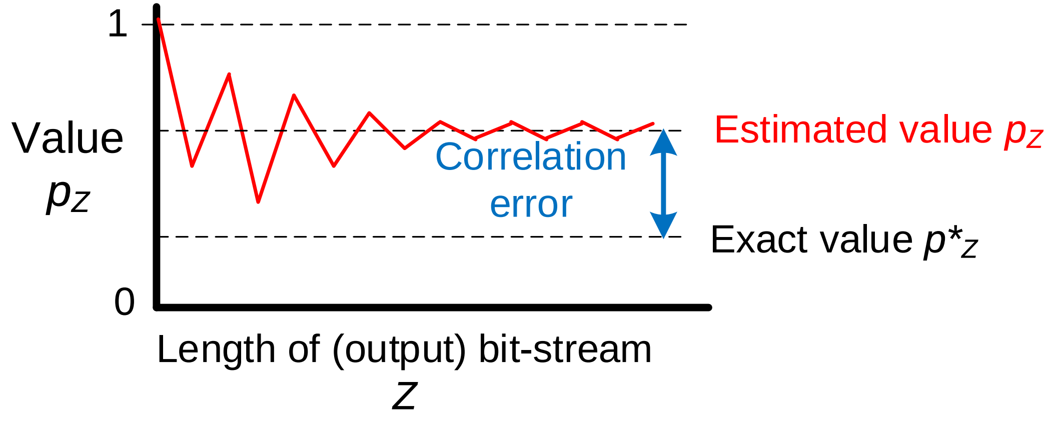
Sources of Inaccuracy



[1] Ting & Hayes ICCD 2016
[2] Ting & Hayes DFT 2017

Correlation Problem

- Correlation is an inherent part of SC because interacting SNs produce results that are dependent, often in subtle ways.
- Unlike random fluctuation errors, correlation errors cannot be eliminated just by increasing bit-stream length.



- Decorrelation is a solution, but it's expensive (and tricky).

Correlation is Hard to Measure

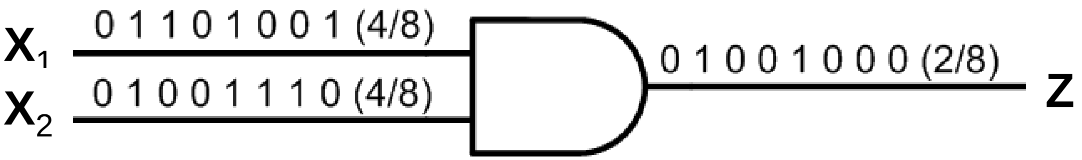
- Independent bit-streams *X, Y* $p_{X \wedge Y} = p_X \times p_Y$
- Standard (Pearson correlation) $\rho(X,Y) = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}$

- SCC metric designed for use in SC [Alaghi and Hayes, *ICCD*'13]

$$SCC(X,Y) = \begin{cases} \frac{p_{X \wedge Y} - p_X p_Y}{\min(p_X, p_Y) - p_X p_Y} & \text{if } p_{X \wedge Y} > p_X p_Y \\ \frac{p_{X \wedge Y} - p_X p_Y}{p_X p_Y - \max(p_X + p_Y - 1, 0)} & \text{otherwise} \end{cases}$$

- SCC = +1 (-1) for maximum (minimum) overlap of 1s and 0s between the bit-streams
- Unlike ρ, SCC is unaffected by the values of the bit-streams

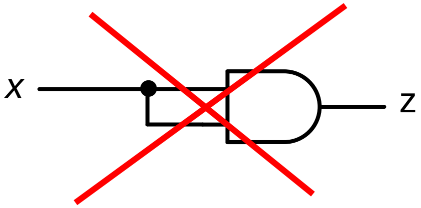
Correlation: AND Gate Multiplier



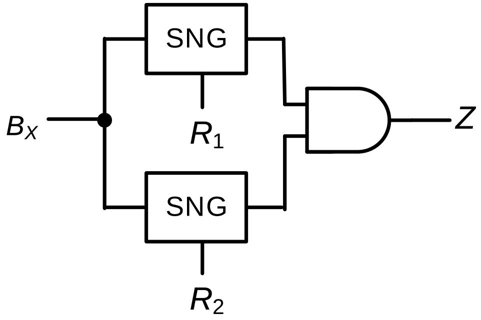
- $p(\mathbf{Z}) = p(\mathbf{X}_1)\, p(\mathbf{X}_2)$, which we write as $Z = X_1X_2$, so an AND gate is a multiplier of SNs.
- This is accurate only if $\mathbf{X}_1$ and $\mathbf{X}_2$ are **uncorrelated**, that is, statistically independent.
- What if the AND inputs are correlated? Two viewpoints:
  - The AND becomes an inaccurate multiplier. For example, if $\mathbf{X}_1 = \mathbf{X}_2 = \mathbf{X}$, then $p(\mathbf{Z}) = p(\mathbf{X})$.
  - It implements a different function accurately.

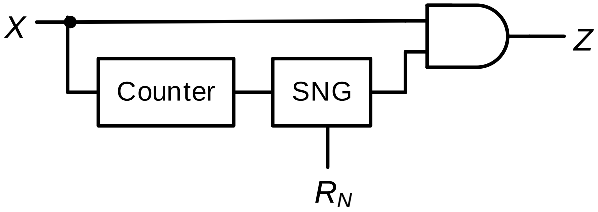
AND Gate Correlation

- *Squaring*: Suppose $X_1 = X_2 = X$ and the target function is X^2, i.e., squaring The function implemented is $Z = X$.

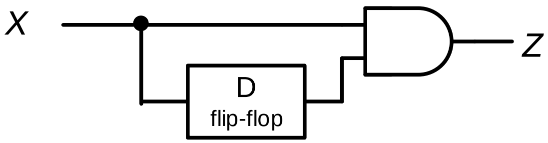


- **Regeneration**: Costly way to compute X^2.





- **Isolation**: Less costly way to compute X^2.



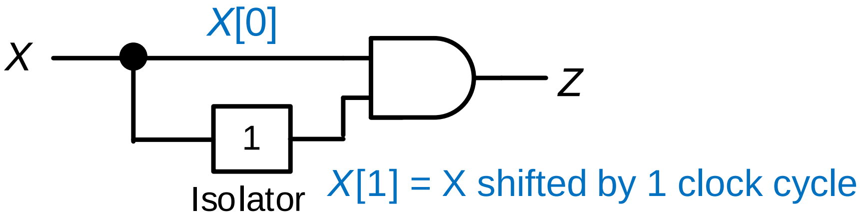
Isolation-Based Decorrelation (IBD)

- Most stochastic circuits are designed to work accurately only with input SNs that are independent (Bernoulli) sequences **X** = $x[0],x[1],x[2]$, … where $x[i]$ denotes the bit at time (clock cycle) *i.*
- IBD exploits the temporal independence among successive bits of **X**. If SN **X** = **X**[0] is delayed by $i \geq 1$ cycles, then **X**[0] and its delayed version **X**[*i*] are uncorrelated.

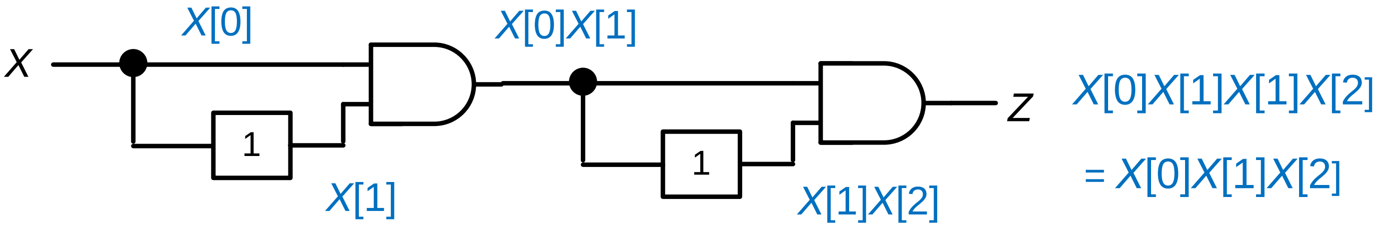
- ***Problem***: Where do we insert isolators in a stochastic circuit to ensure sufficient decorrelation? How do we optimize their number? [Ting and Hayes, ICCD 2016]

IBD Example 1

- Using IBD, we can implement a good squarer thus:

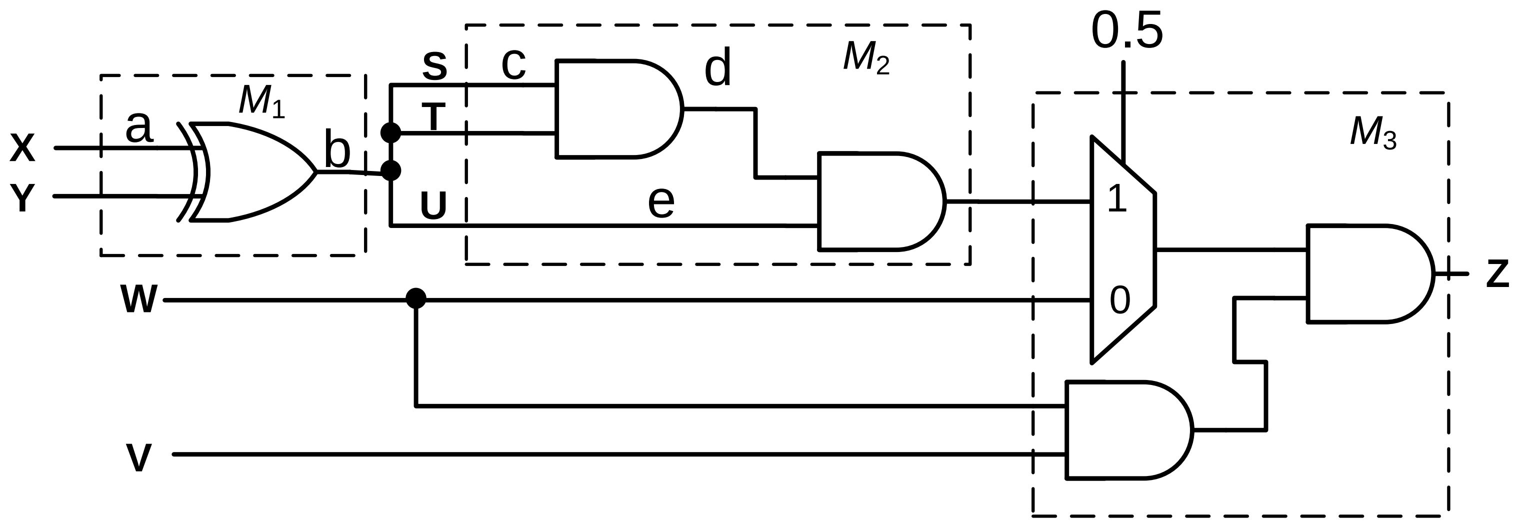


- This reduces correlation errors considerably.
- Now let's concatenate 2 decorrelated squarers to compute X^4.



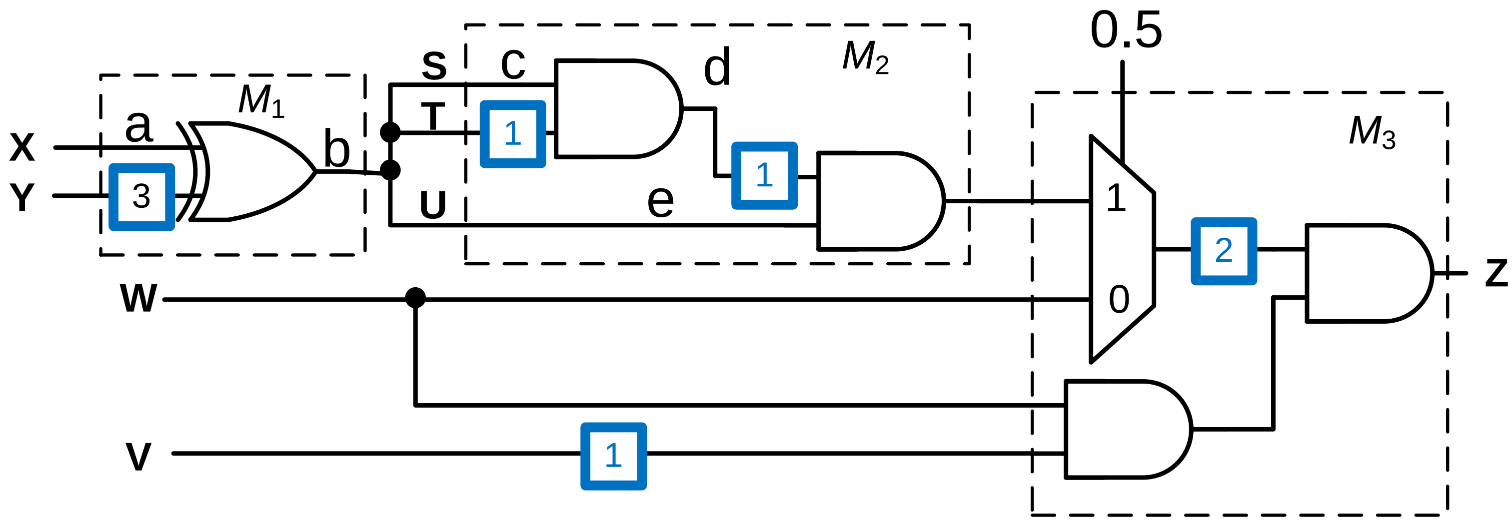
- This circuit actually computes X^3 instead of X^4 *!*

IBD Example 2



- This is a "system" composed of three library modules M_1, M_2, M_3 all of whose inputs we want to decorrelate.
- It computes the polynomial $Z = 0.5WV(X + Y - 2XY)^3 + 0.5W^2V$

IBD Example 2 (contd)

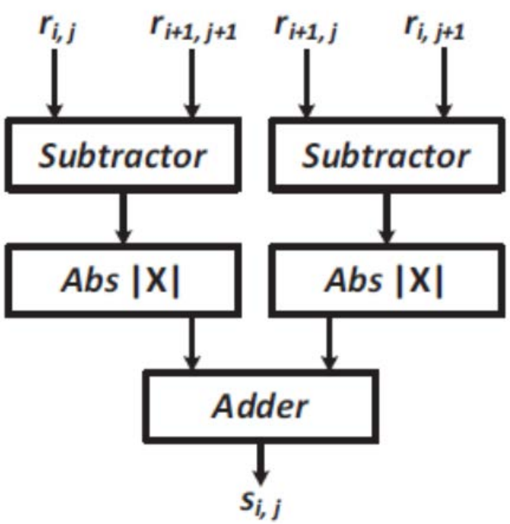


n denotes a sequence of *n* isolators (forming an *n*-bit shift register)

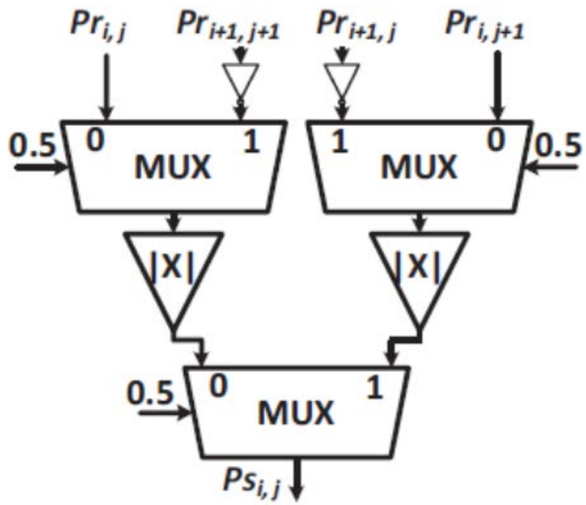
"Good" Correlation

Edge-detection calculation (Roberts cross operation):

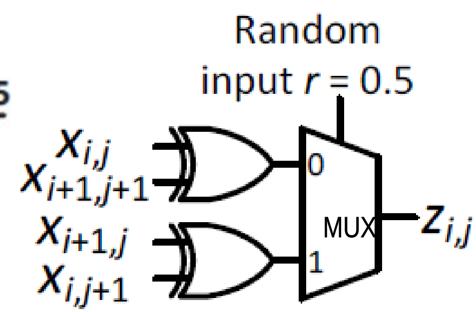
$$s_{i,j} = \frac{1}{2}(|r_{i,j} - r_{i+1,j+1}| + |r_{i,j+1} - r_{i+1,j}|)$$



(a) Conventional non-SC-implementation



(b) Direct SC-based implementation[1]



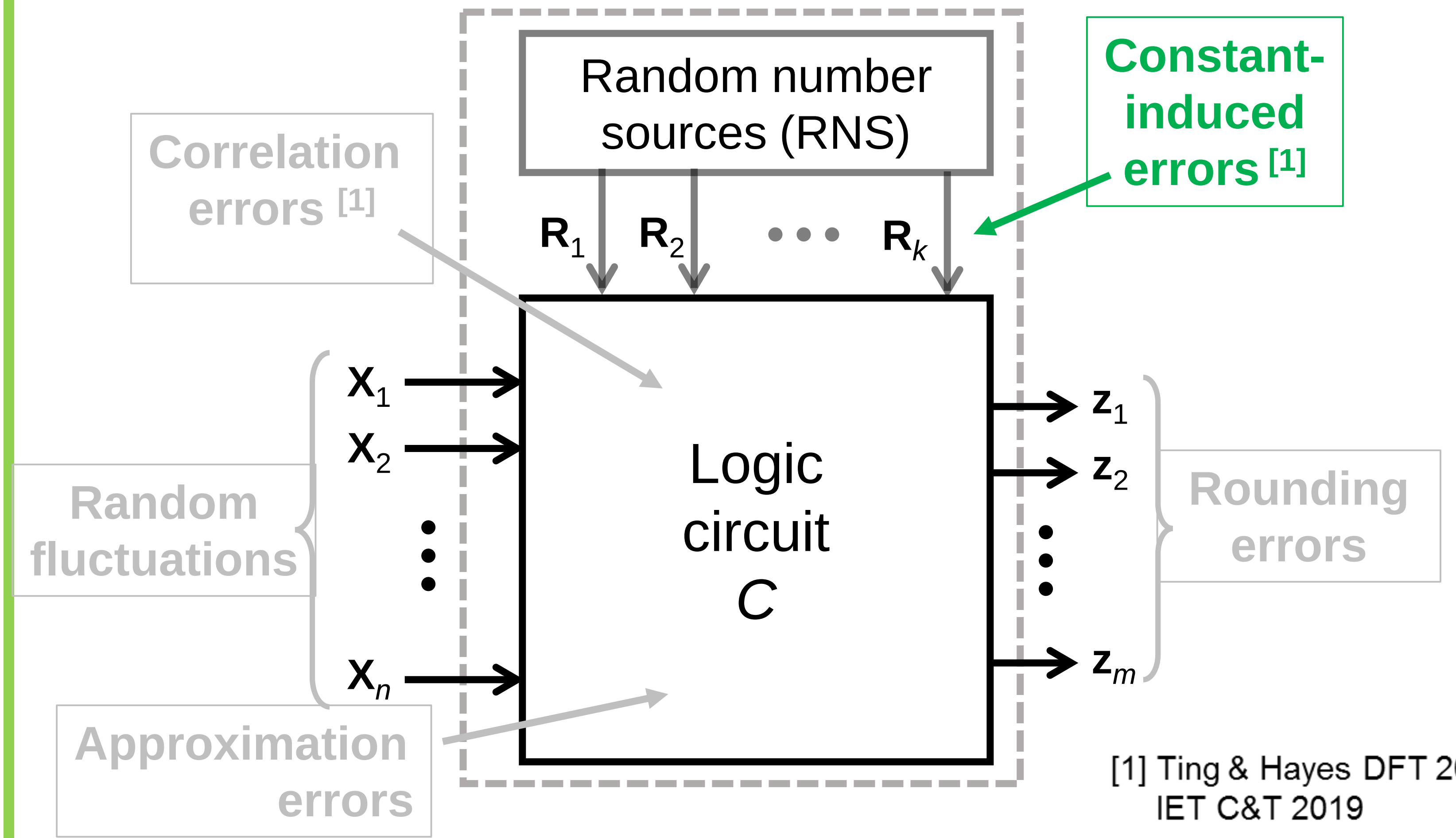
(c) SC-based design exploiting correlation[2]

[1] Li et al. *IEEE-TVLSI* 2014
[2] Alaghi & Hayes *DAC* 2013

Correlation Summary

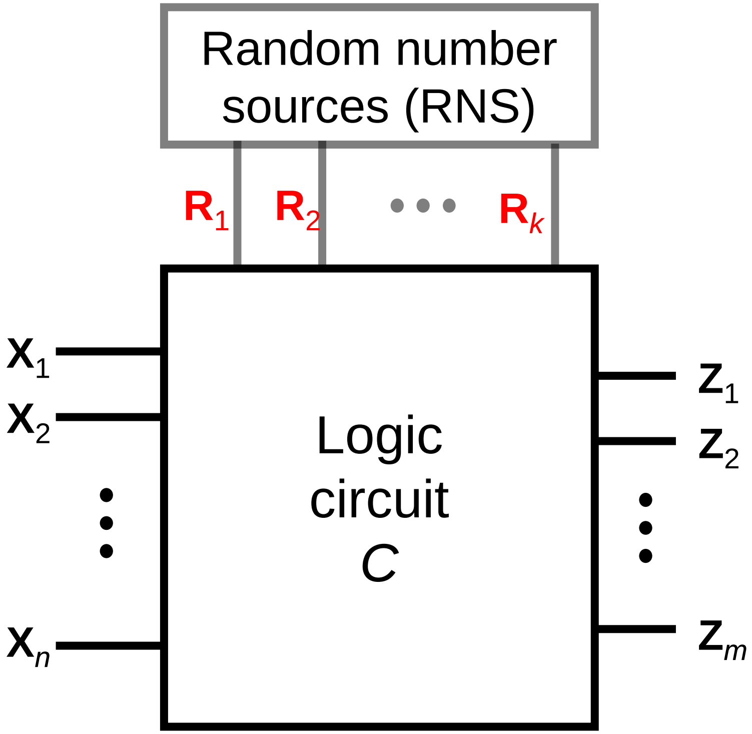
- Correlation is an inherent and complex feature of SC which affects accuracy and functionality.
- Decorrelation is usually needed for accurate operation of larger circuits.
- Isolators provide a promising decorrelation method.
- Interestingly, correlation can sometimes be used as a resource to simplify SC, as in edge detection
- There’s a lot about correlation and decorrelation we still don’t understand

An Unexpected Source of Inaccuracy



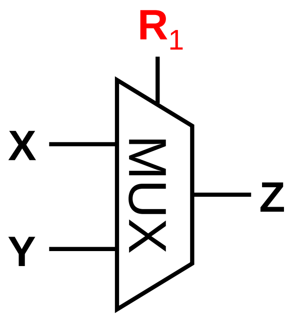
[1] Ting & Hayes DFT 2017, IET C&T 2019

Constant-Induced Errors



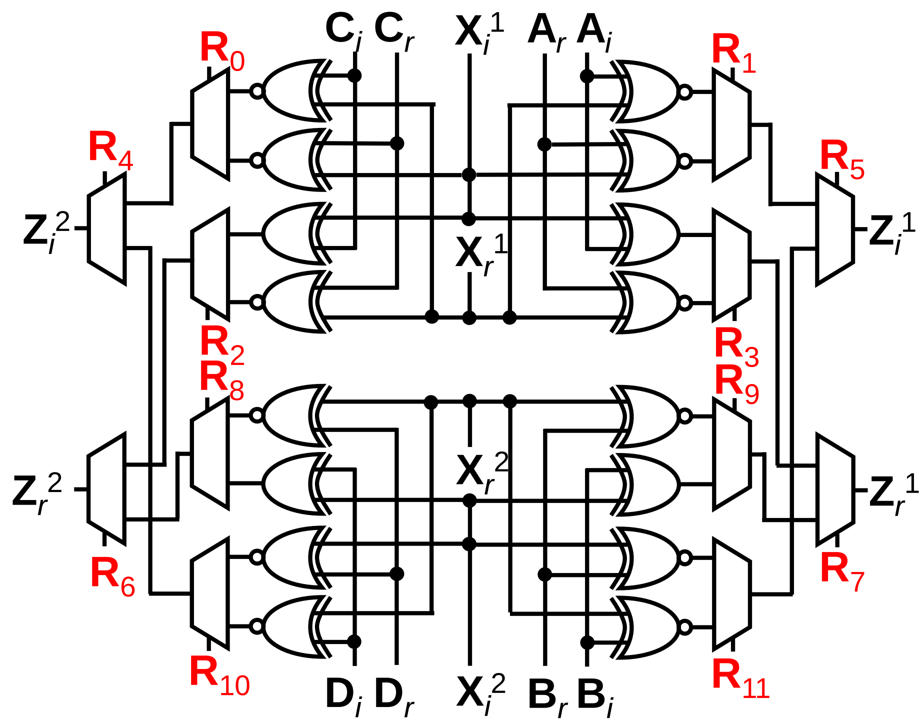
- Constant inputs $\mathbf{R}_i$ are essential in SC design
- We found that these constants
  - Introduce significant random fluctuation errors
  - But the errors are completely removable!

Constants in Stochastic Circuits



Standard scaled adder

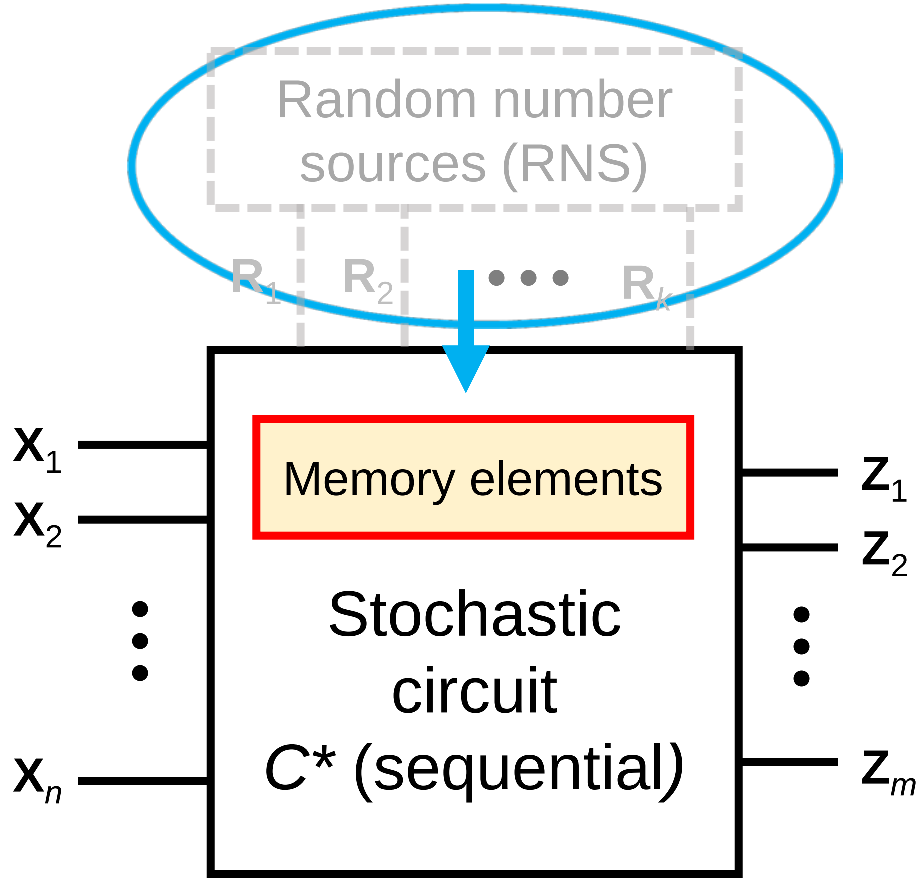
R_i denotes SNs with constant value 0.5



Complex matrix multiplier for quantum circuit simulation [Paler et al. DFT 2013]

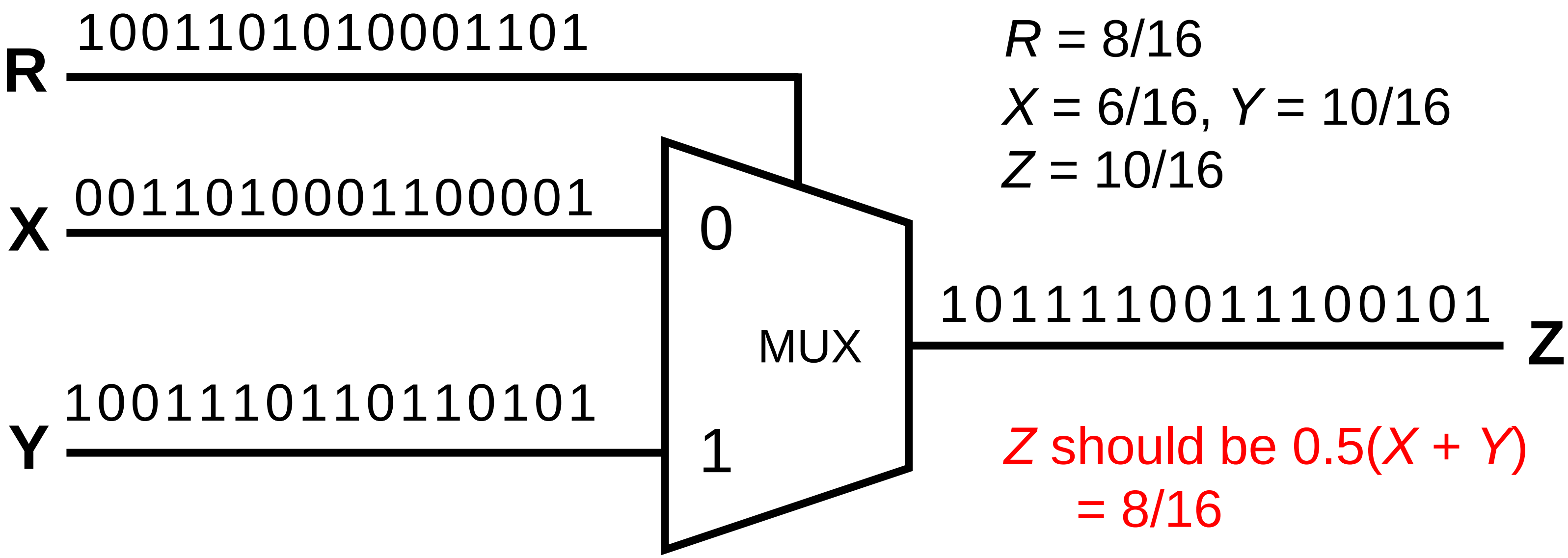
Constant-Induced Errors

We can eliminate constant inputs by transferring their function to memory. via algorithm *CEASE*
[Ting & Hayes DFT 2017]



Constant-Induced Errors: Adder

- **R** is a constant SN of value 0.5
- It selects half the bits of **Z** from **X** and half from **Y** **on average**



R = 8/16

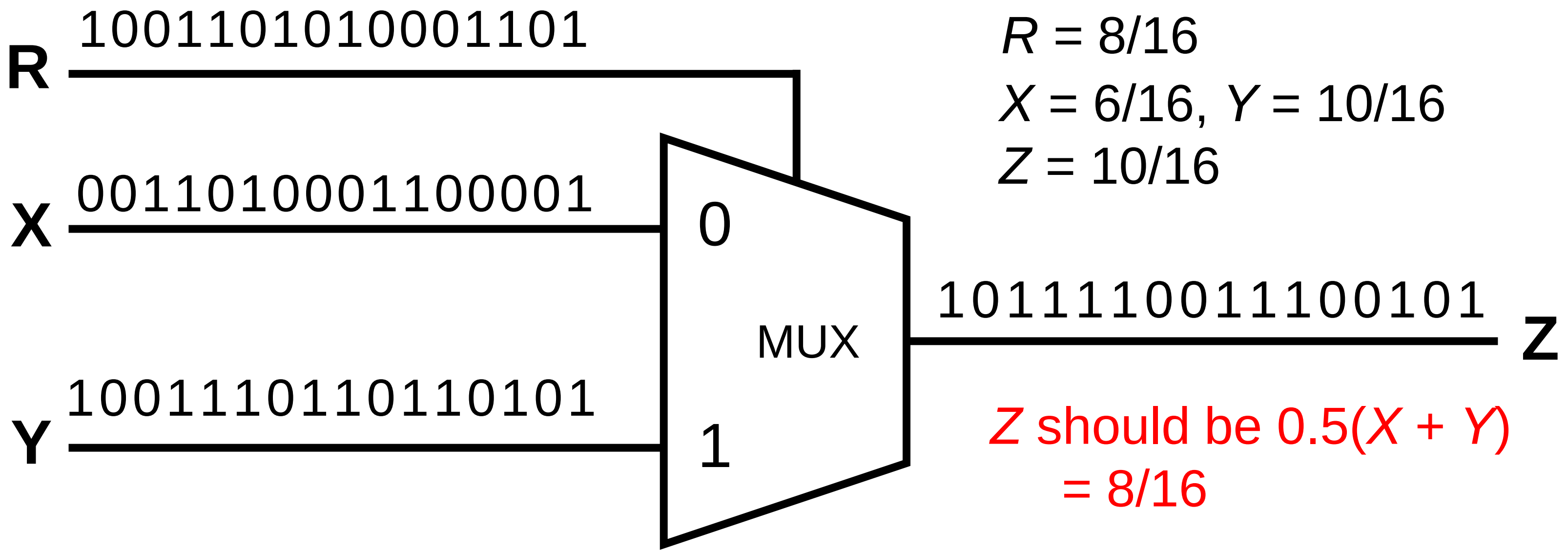
X = 6/16, *Y* = 10/16

Z = 10/16

Z should be 0.5(*X* + *Y*)
= 8/16

Constant-Induced Errors: Adder

- **R** is a constant SN of value 0.5
- It selects half the bits of **Z** from **X** and half from **Y** **on average**



$R = 8/16$

$X = 6/16$, $Y = 10/16$

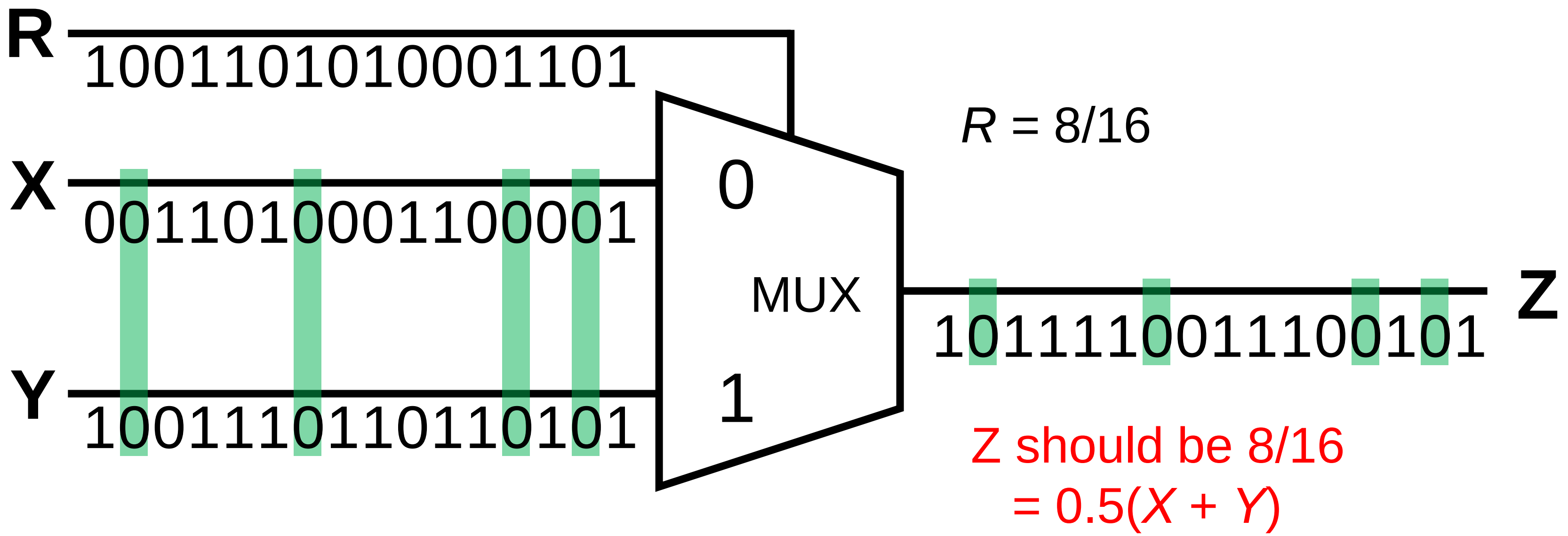
$Z = 10/16$

Z should be $0.5(X + Y)$
$= 8/16$

- **R** affects both the number and quality of the samples of **X** and **Y** due to its random fluctuations

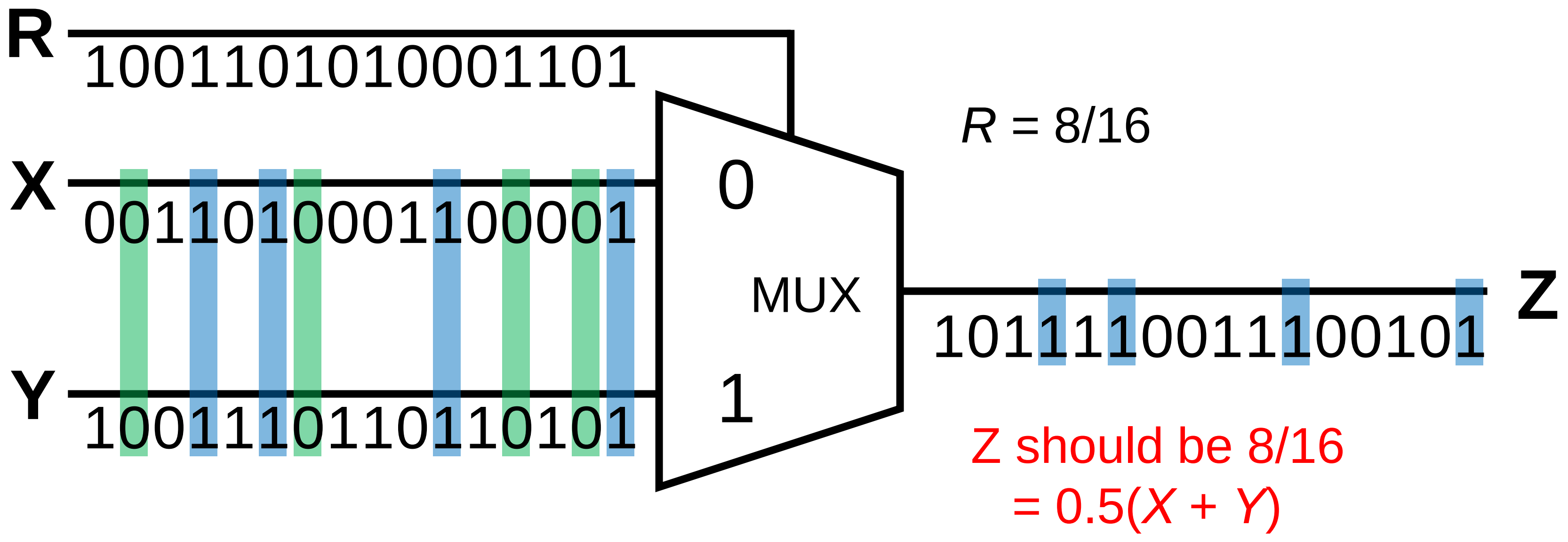
Constant-Induced Errors: Adder

- Consider the adder's response to *xy* = 00 (green)



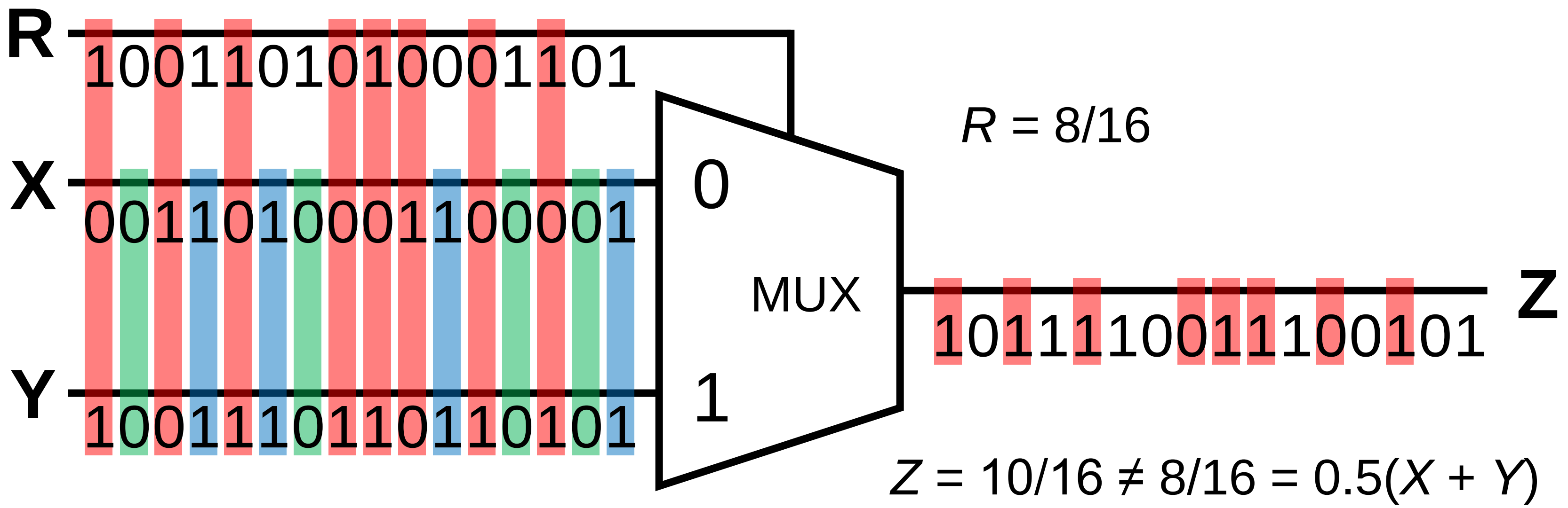
Constant-Induced Errors: Adder

- Consider the adder's response to *xy* = 00 (green)
- Now consider the adder's response to *xy* = 11 (blue)
- In both cases *Z* is exact and error-free



Constant-Induced Errors: Adder

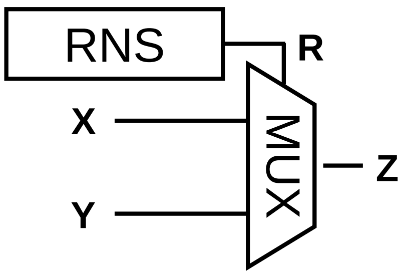
- Finally, consider the adder's responses to *xy* = 01 and 10 (red)
- We expect the responses to be half 0s and half 1s, but 6 instead of the expected 4 logical 1s are produced



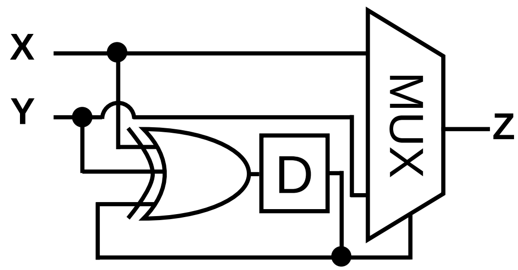
R = 8/16

$Z = 10/16 \neq 8/16 = 0.5(X + Y)$
a 25% error!

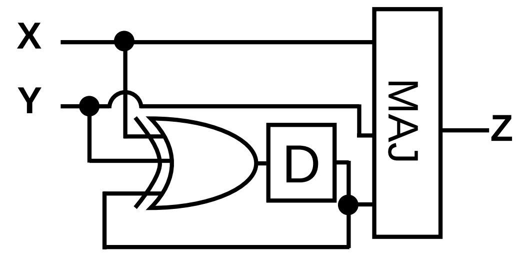
Constant-Free Adders



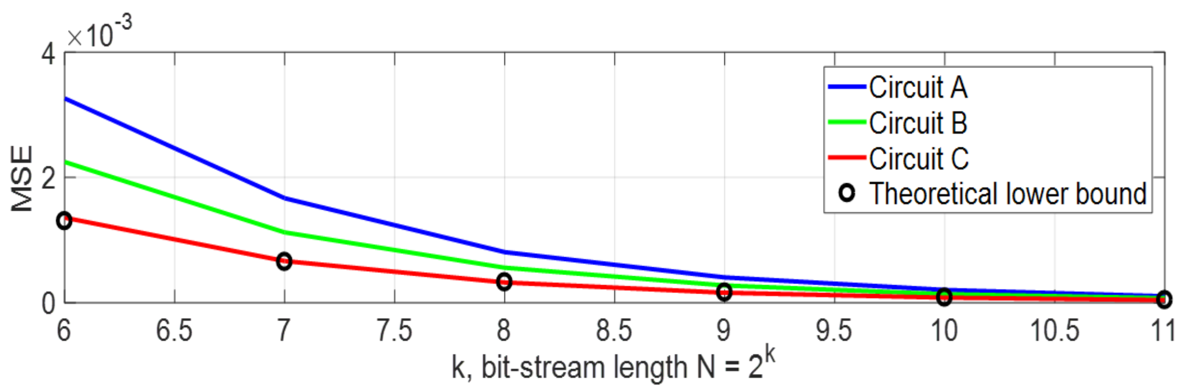
Circuit A
Standard combinational design



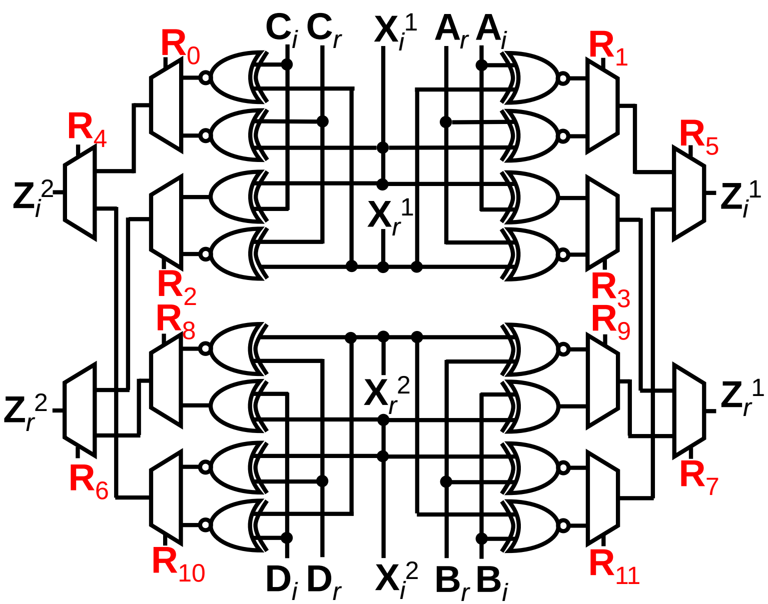
Circuit B
Ad hoc sequential design



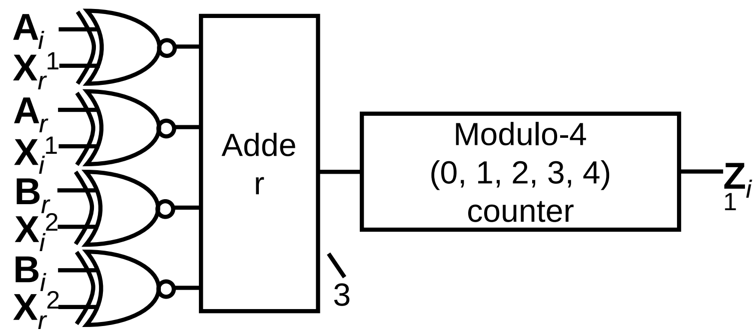
Circuit C
Optimal sequential design by the *CEASE* algorithm



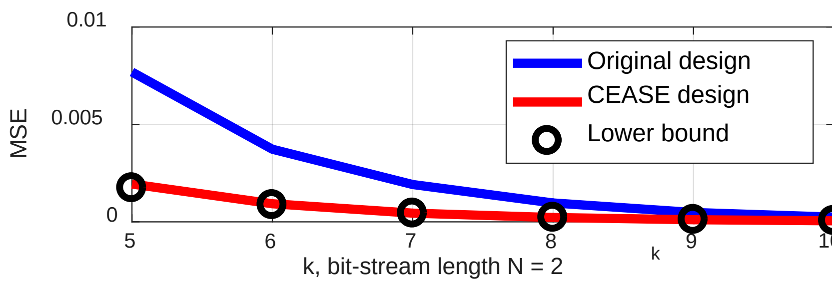
Constant-Free Matrix Multiplier



Original stochastic design for the matrix multiplier design [Paler et al. 2013]



Constant-free *CEASE* design for the output Z_i^1 [Ting & Hayes. 2019]



Exploiting Randomness

- Uncontrolled randomness in SC leads to low accuracy.
- Can we take advantage of SC's intrinsic randomness?
- Some applications benefit from randomness, but the amount of randomness must be carefully controlled
- <u>*Example*</u>: Dithering, which SC can provide automatically.

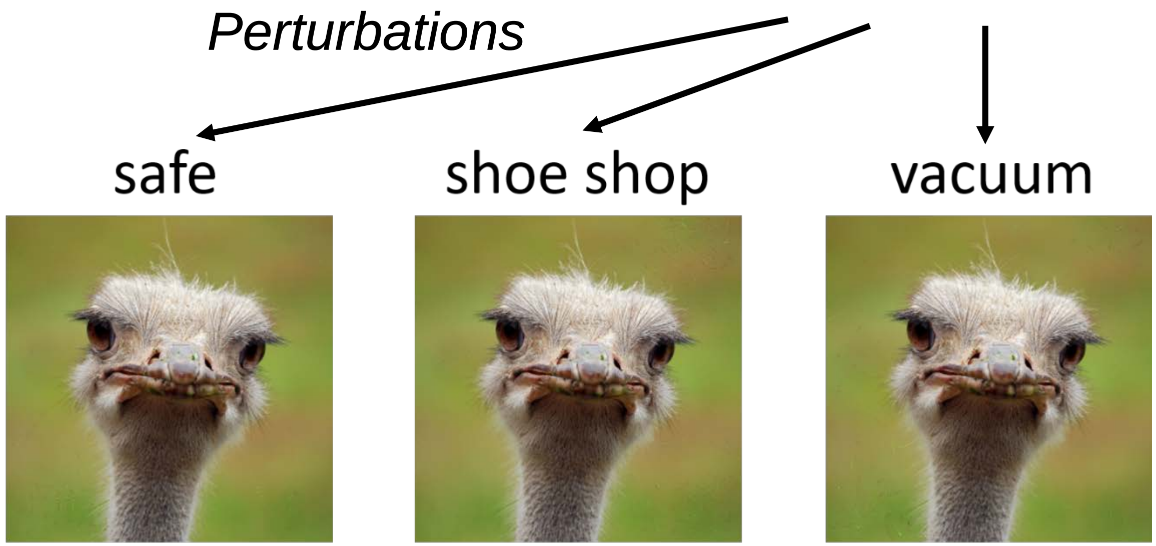
(a) (b) (c) (d)

(a) Original grayscale image, (b) binarized image, (c) binarized image with good dithering, (d) binarized image with excessive dithering.

[Ting & Hayes, ICCAD 2019, to appear]

Ex. 2: *Hardening Neural Neworks*

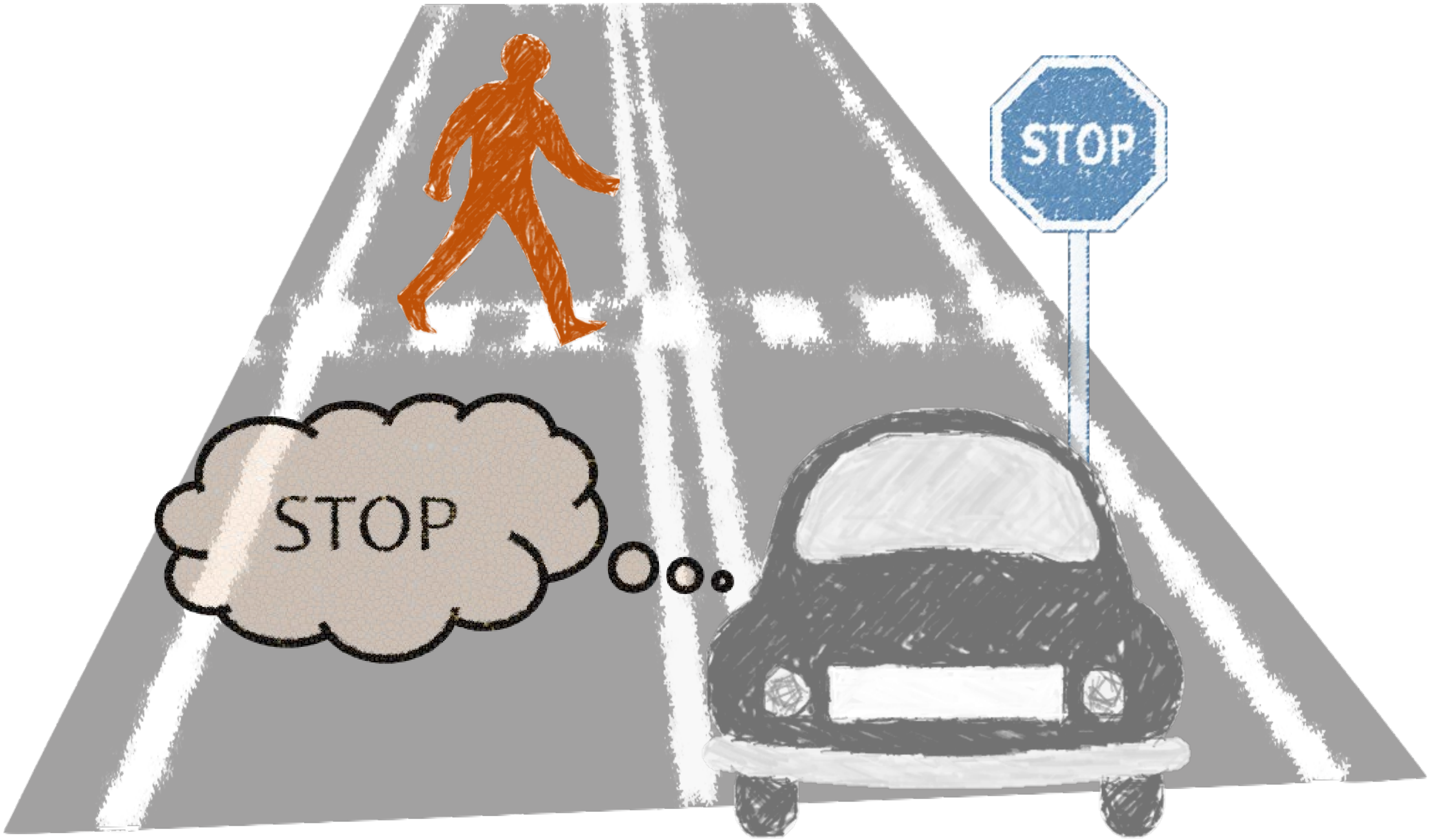
- Adding carefully designed perturbations to an image can lead NNs to misclassify it. This is called an adversarial attack
- Attack on the Inception-v3 classifier [1]



[1] Chen et al. *AAAI* 2018

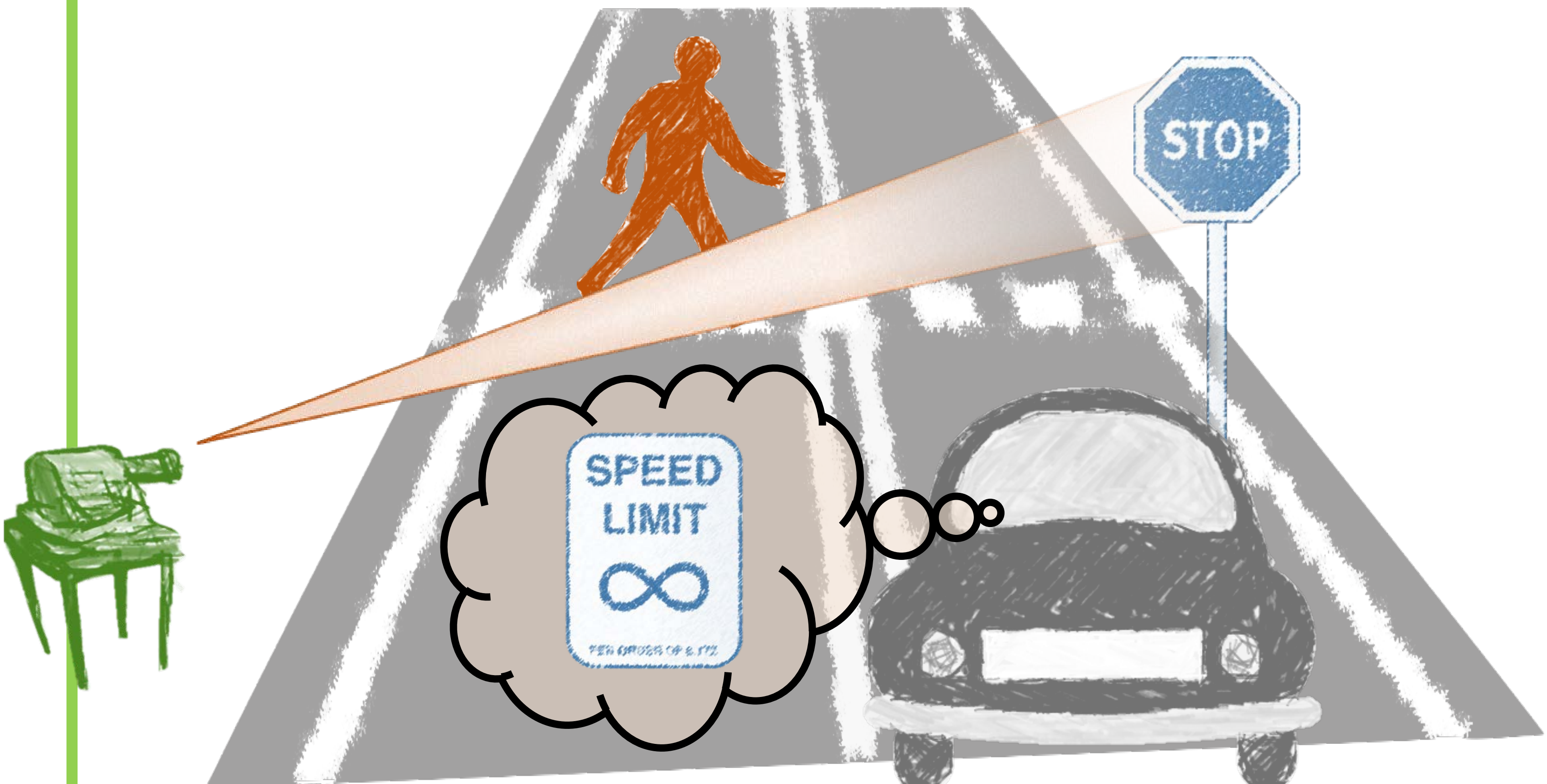
Ex. 3: Security Threat

- Autonomous car using DNN for traffic sign recognition



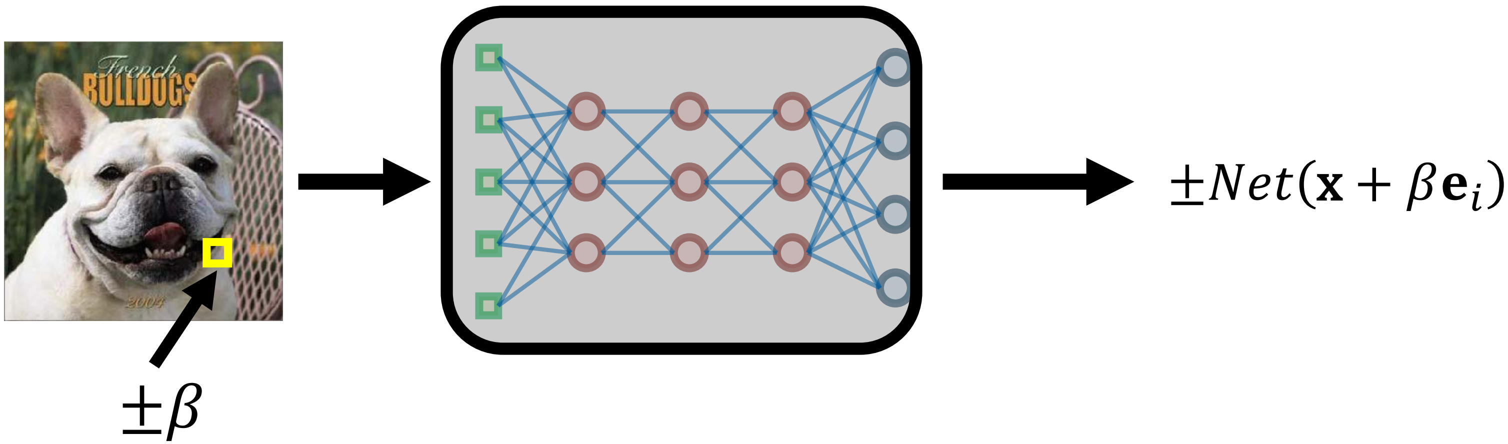
Ex. 3: Security Threat (contd)

- Autonomous car using DNN for traffic sign recognition



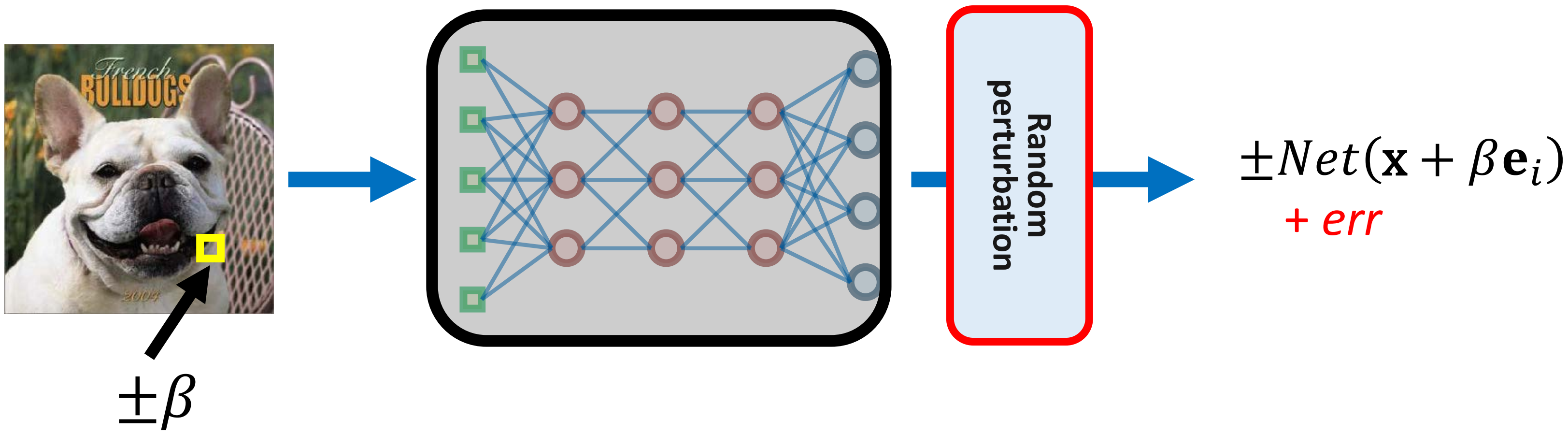
Attack on Black-Box NN

- Black-box setting:
  - DNN’s implementation is concealed
  - Details like number of layers are unknown to the attacker
- Zeroth-order attack[1]:
  - Attacker sends test images to DNN
  - Output responses are leveraged to generate a black-box attack

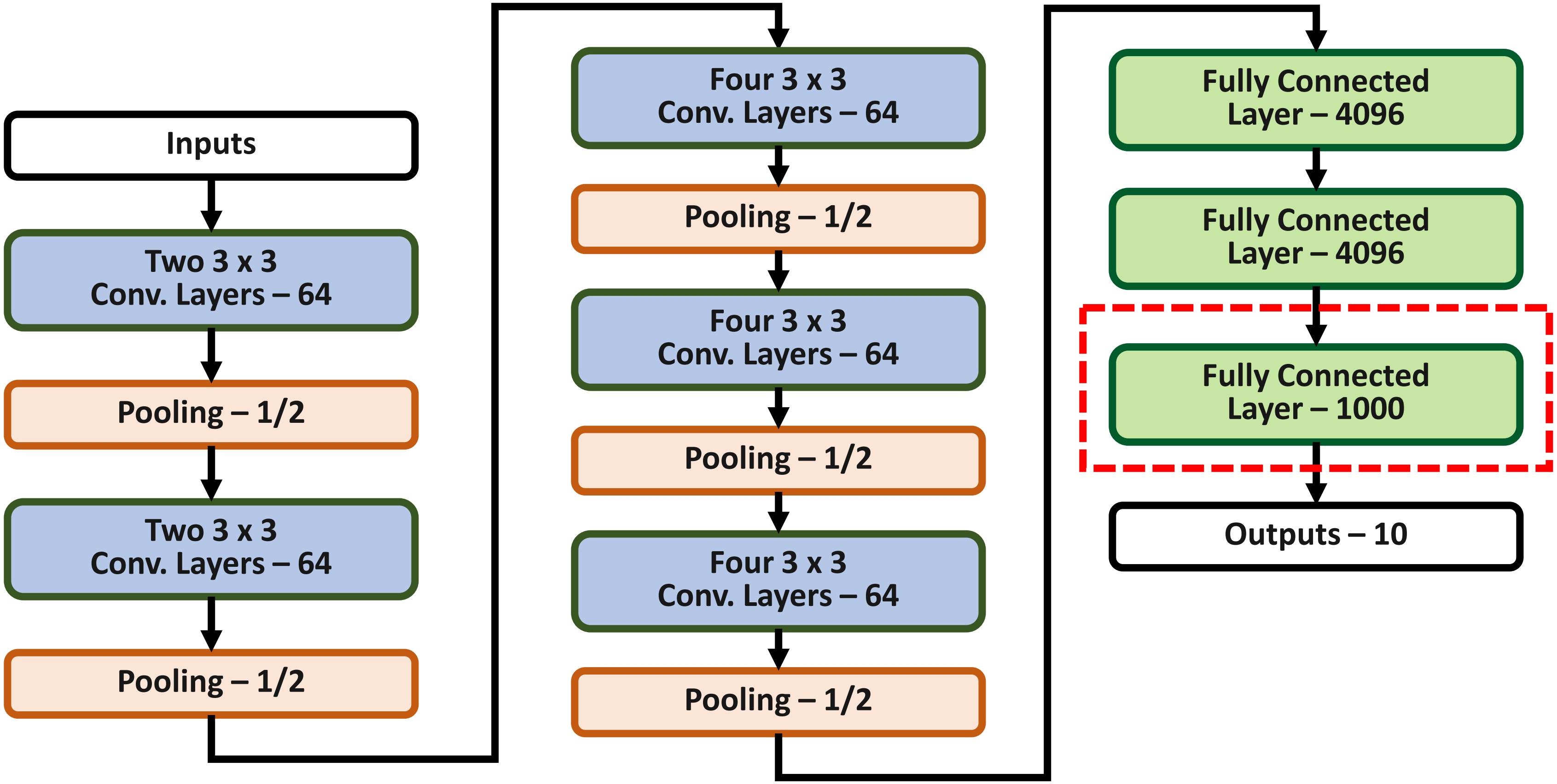


Attack on Black-Box NN (contd)

- Make black-box attacks costlier to generate:
  - Add random perturbations to the input-output responses via an SC layer
  - The DNN must be trained with the added randomness, so it learns to operate in noisy environments

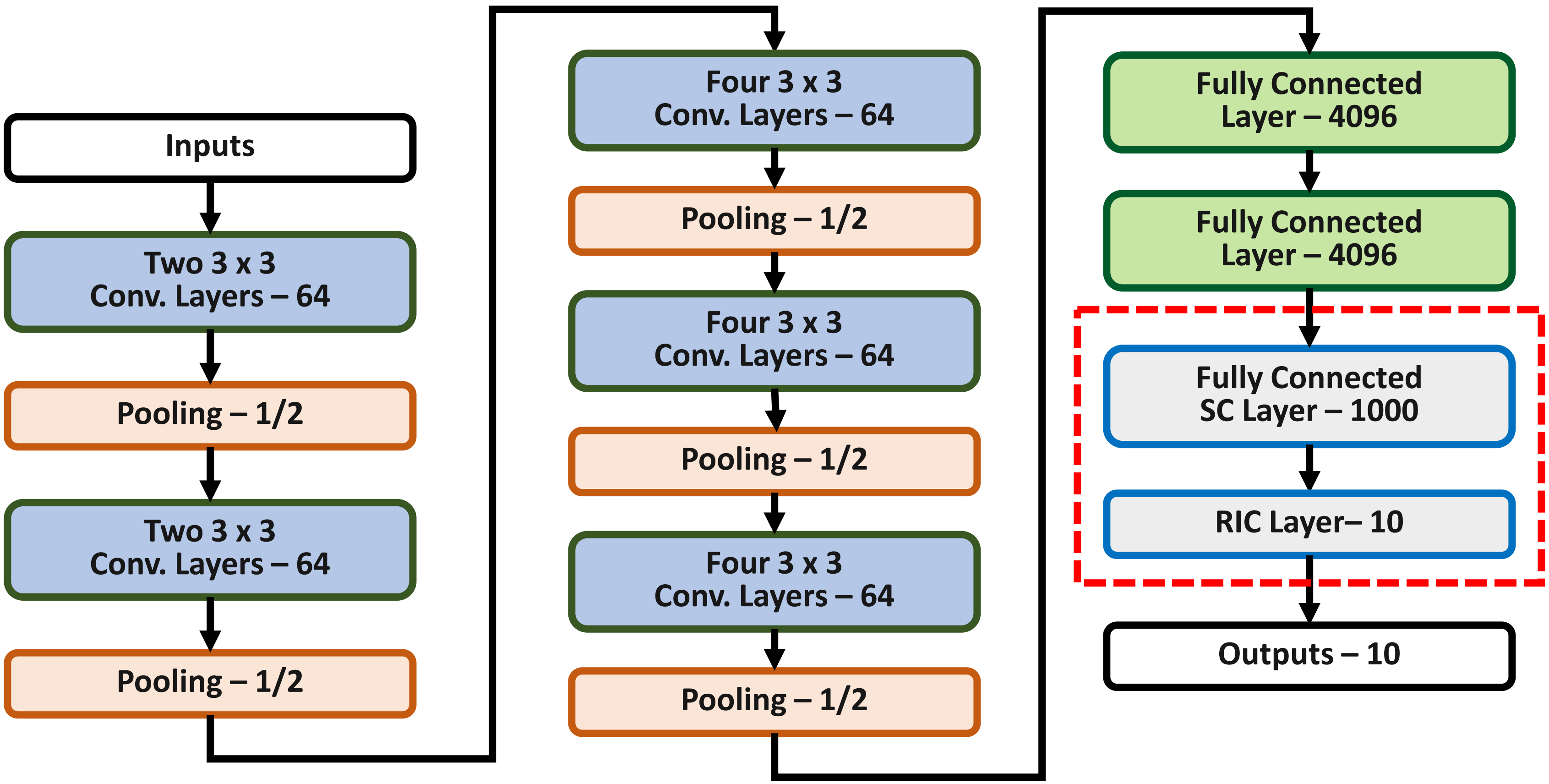


VGG-19 NN Trained on CIFAR-10 (contd)



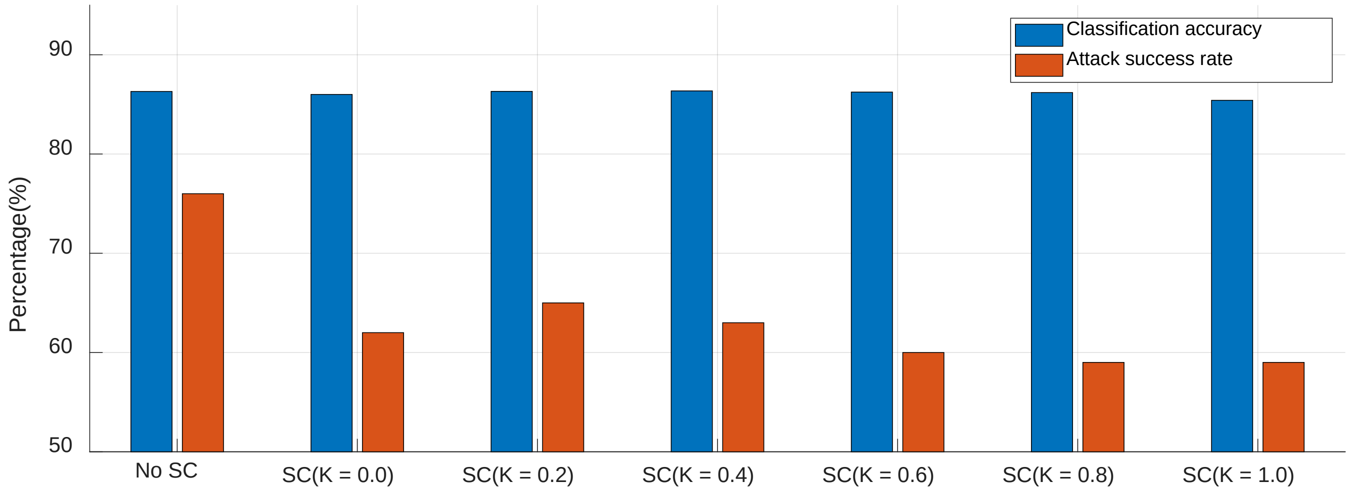
VGG-19 NN Trained on CIFAR-10 (contd)

- Replace last fully-connected layer with an SC implementation.
- Apply ZOO[1], a type of black-box attack, on SC-protected NN



Experimental Results

- Attack success rate (ASR) is the proportion of successful attacks generated within 5,000 optimization iterations
- ASR is reduced from 76% down to 59% without affecting classification accuracy



[Ting & Hayes, ICCAD 2019, to appear]

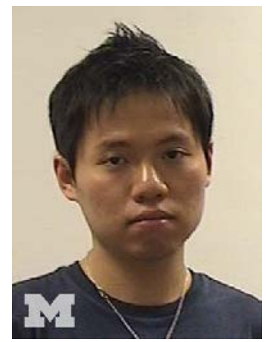
Summary

- Stochastic unary circuits offer the advantages of simple circuitry, low power, bio compatibility, error tolerance, and progressive precision
- Their disadvantages are limited application range, slow calculation, low accuracy, and complex design trade-offs
- Careful design can mitigate many of the disadvantages of stochastic computing
- Some features like randomness and correlation can be either a blessing or a curse
- Many aspects of SC behavior are still poorly understood

Acknowledgements

- Ph.D. students Armin Alaghi, Sean Chen, Paishun Ting, and Tim Baker.






- Collaborators at the University of Stuttgart: Florian Neugebauer and Ilia Polian
- Support from the U.S. National Science Foundation and the German Alexander von Humboldt Foundation