动态随机计算及其应用

刘思廷

2021.06.02

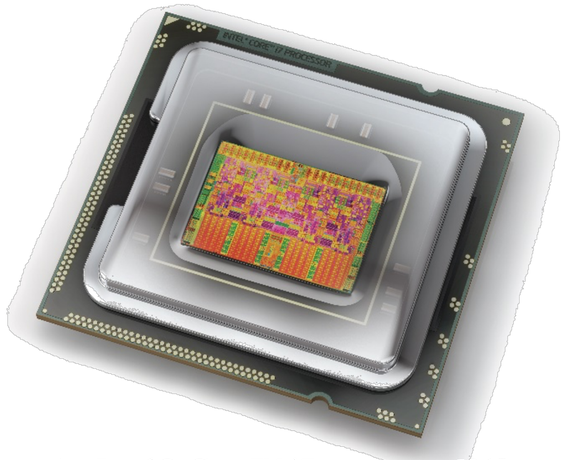
报告提要

- 研究背景
- 动态随机计算（DSC）
    1. 数字信号处理
    2. 求解微分方程
    3. 神经网络训练
- 结论

研究背景

集成电路设计面临挑战：

- 暗硅（dark silicon）
- 计算密集型任务（如机器学习算法等）



An Intel® Core™ i7 processor [21].

导致：

- 性能牺牲
- 高能耗

传统计算方法 ➡️ 随机计算

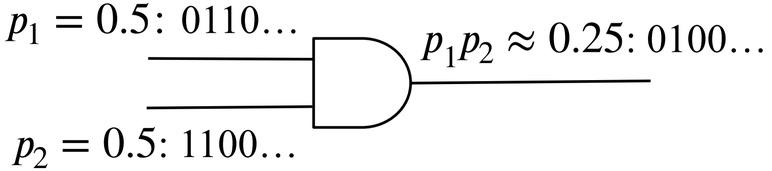
- 随机计算中，数值以随机0和1组成的序列表示，数值大小与序列中1出现的概率相关。

$01011101\ldots$ 5/8 $(01011101)_2$ 93

$00011101\ldots$ 4/8 $(00011101)_2$ 29

随机序列　　BCD编码

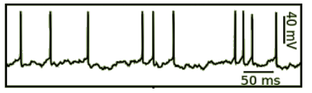
- 不考虑随机序列生成部分，随机计算电路所需资源仅约为传统计算电路的1/100。



单极型随机乘法器（与门）

- 随机计算与大脑中神经频率编码（rate coding）方式类似

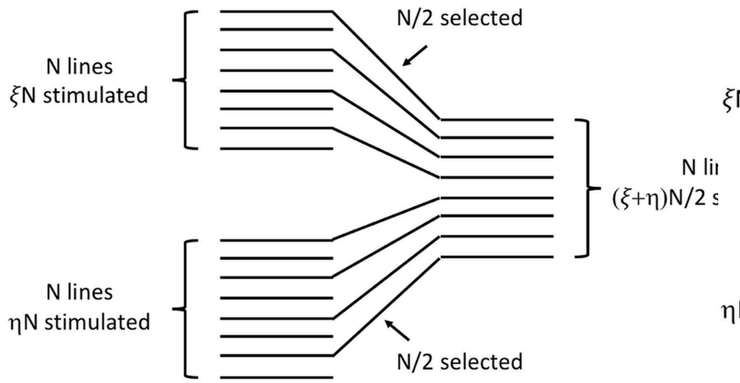
$01011101\ldots$　　随机序列



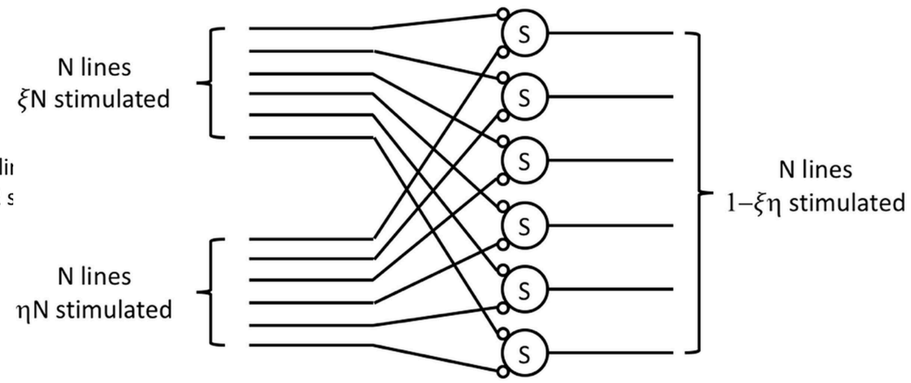
神经脉冲序列

随机计算的起源

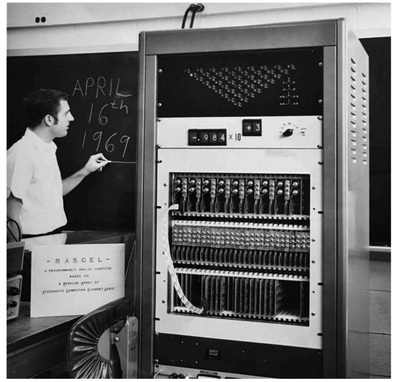
- 20世纪50年代，冯诺依曼等计算机先驱以大脑中神经连接为模型，探索了如何利用不可靠器件完成可靠的计算，构成随机计算的雏形。



通过从两组随机传输线中随机选取一半构成新的一组传输线，计算平均值

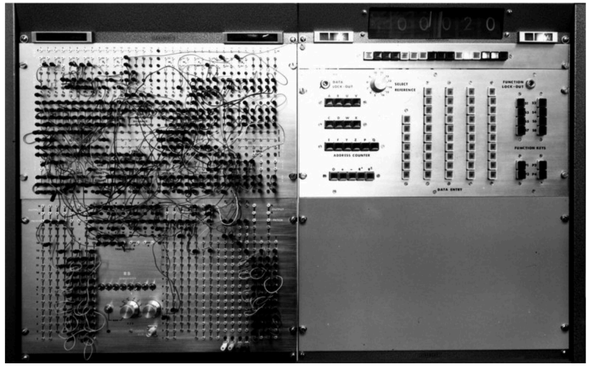


通过与门实现并行随机乘法操作



RASCEL
University of Illinois

Paramatrix
University of Illinois

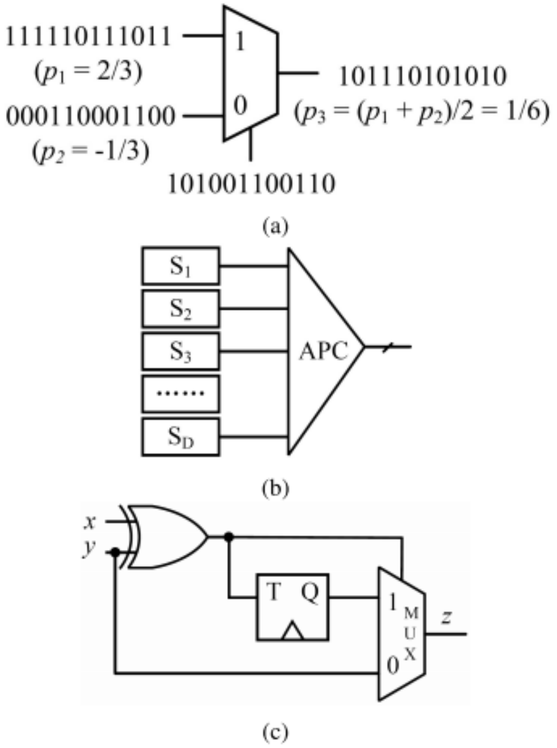


Phase computer
Standard Telecommunication Laboratories

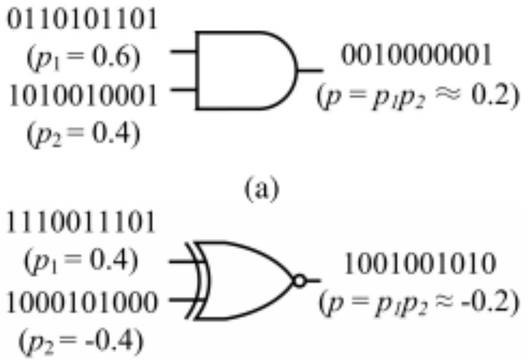
- 60年代至90年代，随机计算理论获得了一定进展，出现了诸多随机计算原型机。

随机计算的再次兴起

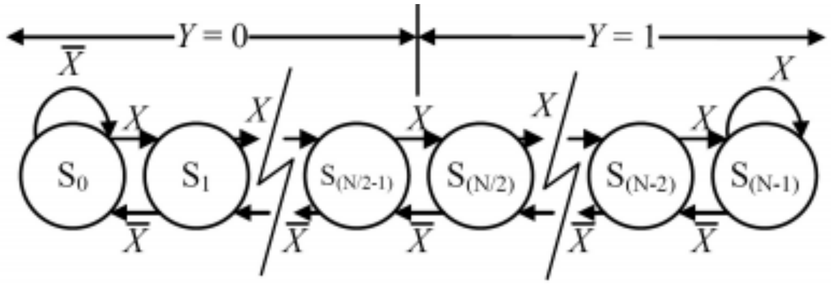
- 2000年前后，加拿大学者Brown C. Card提出各种基于随机计算的算术电路，可以使用极小的硬件资源满足神经网络的计算需求。随机计算伴随神经网络的兴起再次成为研究热点，并出现一系列基于随机计算的神经网络加速器设计。



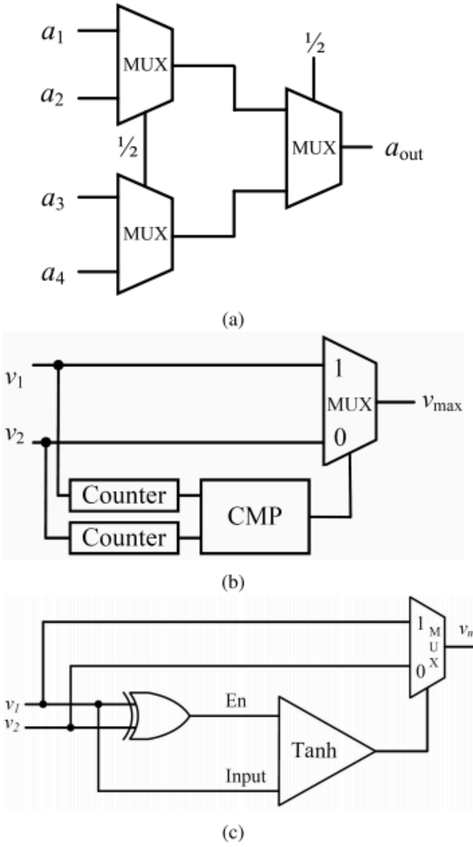
随机加法器设计



随机乘法器



随机激活函数电路



随机池化电路设计

随机计算与其它计算方法的联系

- 概率计算 （probabilistic computing）

利用贝叶斯、蒙特卡洛分析等概率相关理论进行数据建模、推理、优化等，但较少应用于通用计算。有时也指利用物理器件随机特性实现某些指定算法的计算方法。

- 近似计算（approximate computing）

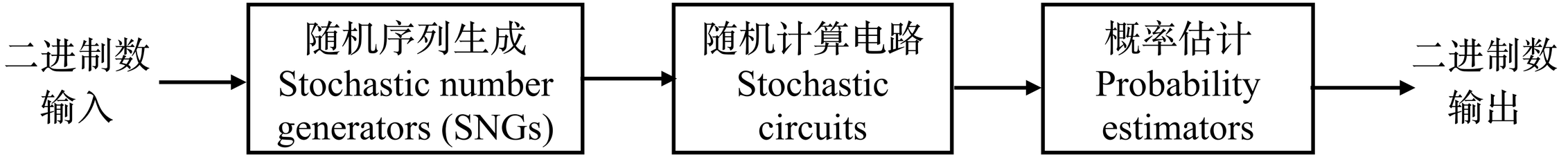
利用某些应用对计算准确度不敏感性，简化算法流程、计算电路等，以降低算法、电路的复杂度、能耗等。

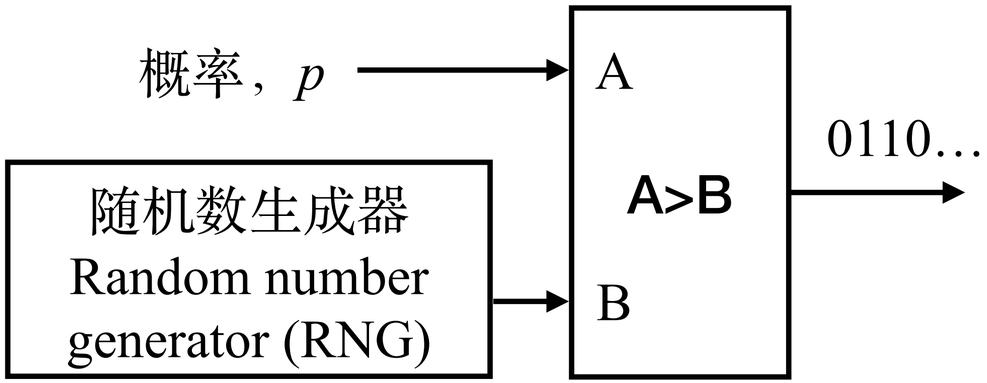
- 随机计算 （stochastic computing）

使用随机序列编码数值，利用概率相关理论进行基本计算（加、乘、积分、多项式等），得到的结果通常为准确值的无偏估计。可使用随机计算电路级联实现复杂运算。

随机计算基础——随机计算系统

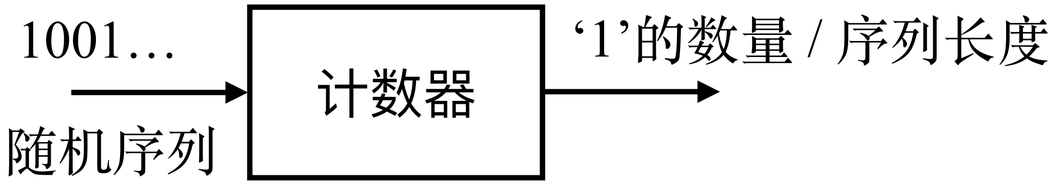
随机计算系统一般由随机序列发生阵列、随机计算电路、随机-二进制转换器构成。





随机序列生成器（SNG）

生成随机序列中1的概率为 p



概率估计

序列长度通常选取为2的幂次方

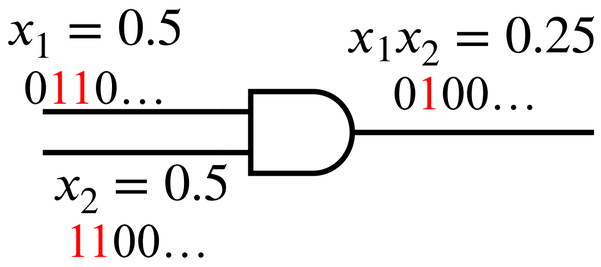
随机计算基础

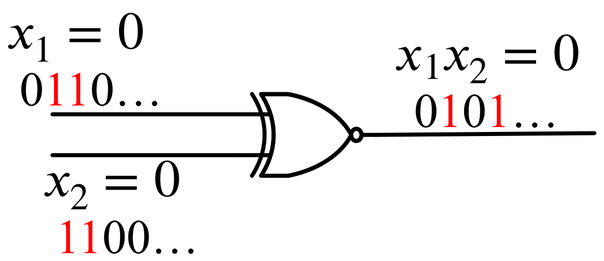
数的表示：假设一个由0和1组成的随机序列中，1出现的概率为p

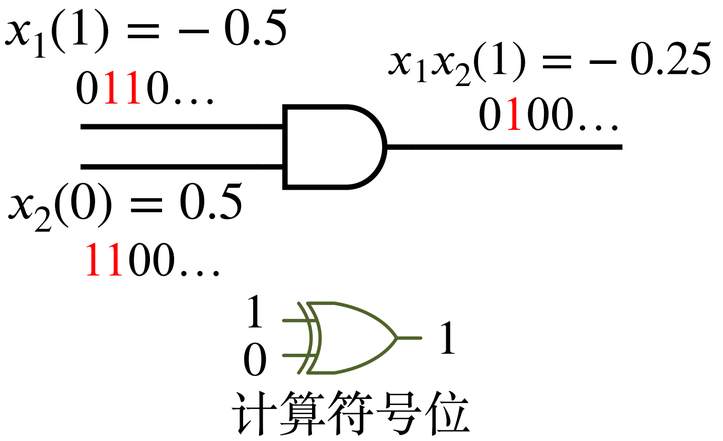
表示方法	所表达数值与序列概率的映射关系	表示范围	符号位	示例
单极型	$x = p$	[0, 1]	无	0010101001 ➡️ $x = p = 0.4$
双极型	$x = 2p - 1$	[-1, 1]	无	0010101001 ➡️ $x = 2p - 1 = -0.2$
有符号型	$\|x\| = p$	[-1, 1]	有	0010101001 sign: -1 ➡️ $x = p$ ✖ $-1 = -0.4$
非线性型	$x = p / (1 - p)$	[0, +∞)	无	0010101001 ➡️ $x = p / (1 - p) = 2/3$

随机计算电路

使用不同的表达方式，有时对应不同的随机计算电路。







单极型、双极型及有符号随机乘法器

要求输入随机序列相互独立

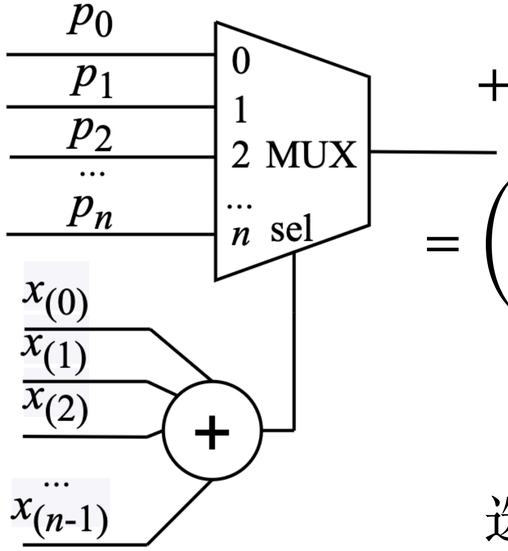
随机计算电路

随机计算电路一般为组合逻辑电路或时序电路。随机计算电路本身不具有随机性，但可以按位处理随机序列，通过简单逻辑实现复杂计算。

$$p_{out} = \binom{n}{0} p_0 (1-x)^n + \binom{n}{1} p_1 (1-x)^{n-1} x^1 + \dots$$

$$+ \binom{n}{k} p_k (1-x)^{n-k} x^k + \dots + \binom{n}{n} p_n x^n$$

$$= \binom{n}{k} \sum_{k=0}^{n} (1-x)^{n-k} x^k$$



并行独立x

选择器电路与二进制加法器实现
伯恩斯坦(Bernstein)多项式

$$Y = 0 + 0 \times X_1 + 0 \times X_2 + X_1 X_2$$

Walsh-Hadamard 变换

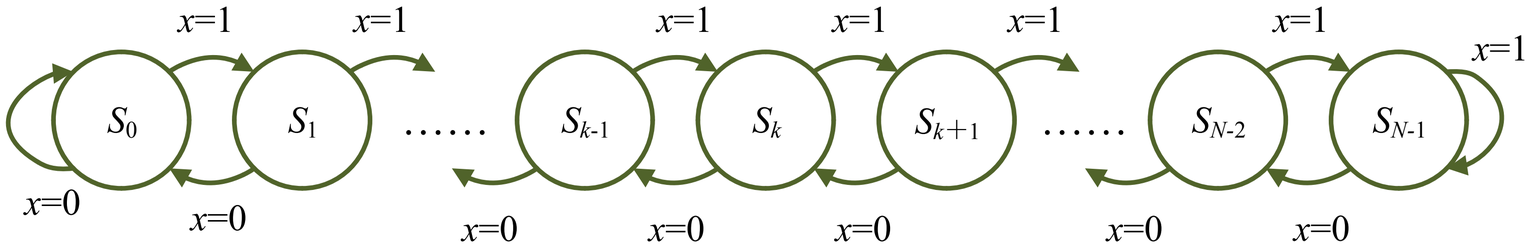
$$\mathrm{BF} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix}$$

00	01	10	11
0	1	1	0

$$y = x_1 \oplus x_2$$

随机计算电路——时序电路

随机计算电路一般为组合逻辑电路或时序电路。随机计算电路本身不具有随机性，但可以按位处理随机序列，通过简单逻辑实现复杂计算。



基于有限状态机的随机计算电路（状态转移图）

概率转移矩阵

$$P=\begin{bmatrix} 1-x & x & 0 & 0 & & \\ 1-x & 0 & x & 0 & \cdots & \mathbf{0} \\ 0 & 1-x & 0 & x & & \\ & \vdots & & & \ddots & \vdots \\ & \mathbf{0} & & & 0 & x \\ & & & & 1-x & x \end{bmatrix}$$

随机积分器

平衡状态概率分布

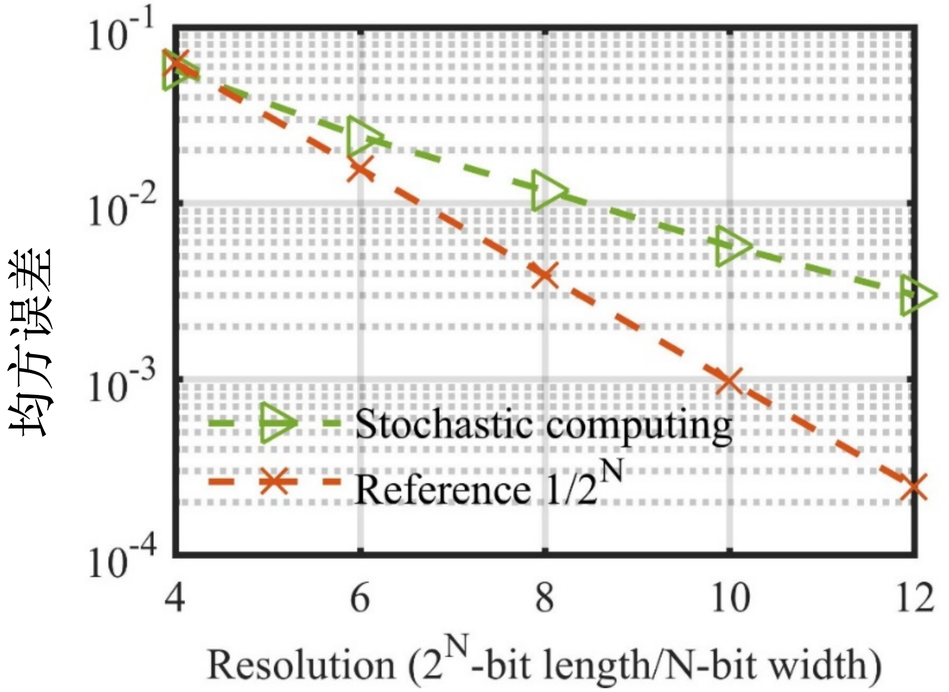
$$P[S=S_i]=\frac{\left(\frac{x}{1-x}\right)^i}{\sum_{j=0}^{N-1}\left(\frac{x}{1-x}\right)^j}$$

随机计算电路的设计挑战

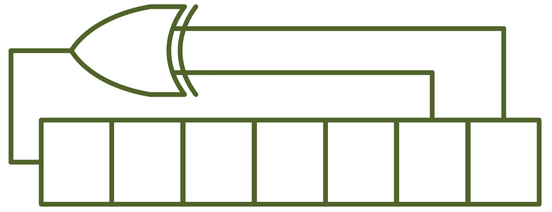
- 由于基于概率理论，产生的较为准确的结果往往需要大量样本，即足够的序列长度。而增加1位精度，序列长度成指数增长，导致高延时、高能耗。解决方案：利用尽可能短的序列产生尽可能高准确度的结果。

- 产生随机序列需要复杂的随机序列发生器，通常由线性反馈移位寄存器（LFSR）与比较器实现，可占据随机计算系统80%硬件资源。高效的和/或利用物理器件自然特性的随机源有望成为可能的解决方案。

- 随机计算与大脑信息传递方式相似，它们之间的联系尚未完全探索。



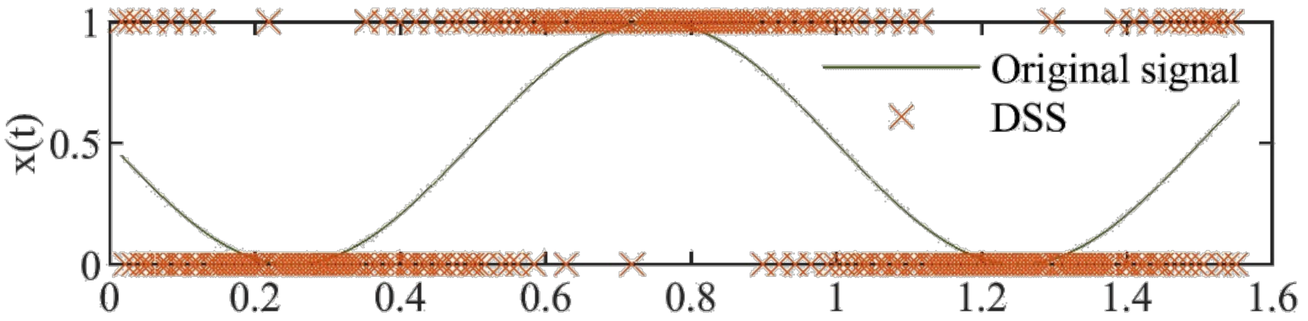
随机序列长度与均方误差的关系
（使用伪随机序列）



LFSR作为伪随机信号源

动态随机计算

- 动态随机计算 (DSC)：使用动态随机序列和随机计算电路进行计算的方法。
- 动态随机序列(DSS)：假设一个由随机的0和1组成的数列$\{X_k\}$和一个相同长度的数字信号$\{x_k\}$，对于任意$k = 0,1,\ldots$，满足$\mathbb{E}[X_k] = x_k$，则称$\{X_k\}$是一个编码信号$\{x_k\}$的动态随机序列（单极型）。动态随机序列中每个比特位都对应一个数字信号值，无需处理整个随机序列即可实时产生结果，提高编码/计算效率。
- 例子：使用一个动态随机序列编码一个数字正弦信号

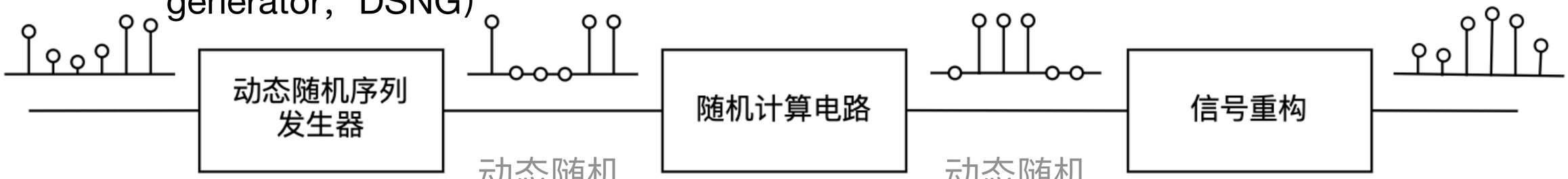


动态随机序列（Dynamic stochastic sequence，DSS）

信号编码方法	所表达数值与序列期望的映射关系
双极型	$\mathbb{E}[X_k] = (x_k + 1)/2$
有符号型 {-1,0,1}	$\mathbb{E}[\lvert X_k \rvert] = \lvert x_k \rvert$ $\text{sign}(X_k) = \text{sign}(x_k)$

动态随机计算系统

（Dynamic stochastic number generator，DSNG）



数字信号输入，由ADC等采集或存储于储存设备中

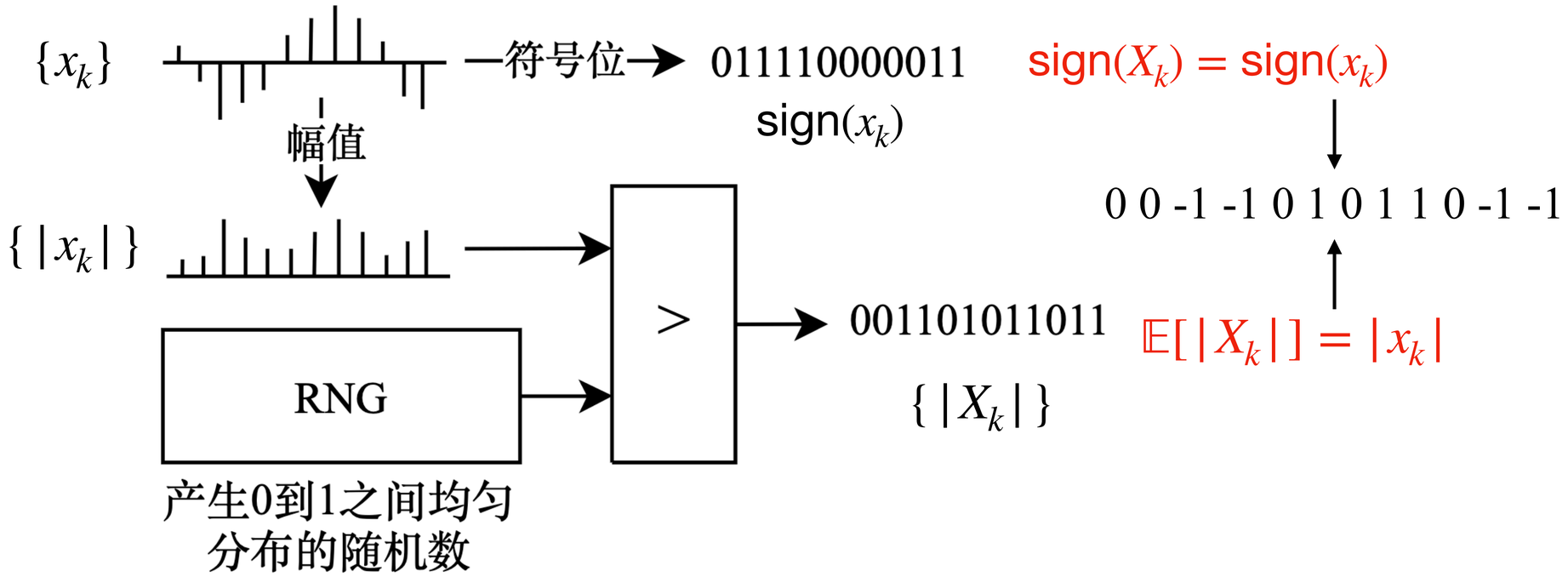
动态随机序列

动态随机序列

数字信号输出，可被转换为模拟信号或存储于储存设备中等待进一步处理

- 动态随机序列也可以直接由模拟信号，通过模拟比较器与随机信号比较，采样获得。
- 在一些情况下，随机计算电路，如随机积分器，既可以完成计算功能，同时兼具信号重构的作用。

动态随机序列的生成（有符号型）



动态随机序列发生器（DSNG）

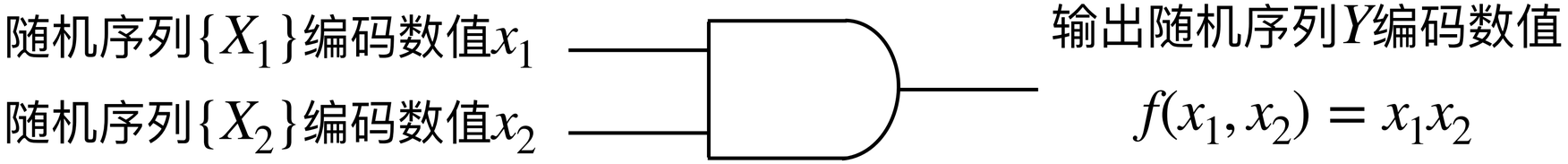
每个时钟周期，一个幅值与一个随机数进行比较
产生编码幅值的序列；符号位序列被直接提取。

动态随机计算电路

- 组合逻辑电路

假设一个组合逻辑电路，实现随机计算函数 $f(x_1, x_2, \ldots)$，其中，$\{X_1\}, \{X_2\}, \ldots$分别为传统随机序列，编码 $x_1, x_2, \ldots$；则当$\{X_1\}, \{X_2\}, \ldots$为动态随机序列编码数字信号 $x_1[n], x_2[n], \ldots$时，该组合逻辑实现复合函数计算 $f(x_1[n], x_2[n], \ldots)$。

例：

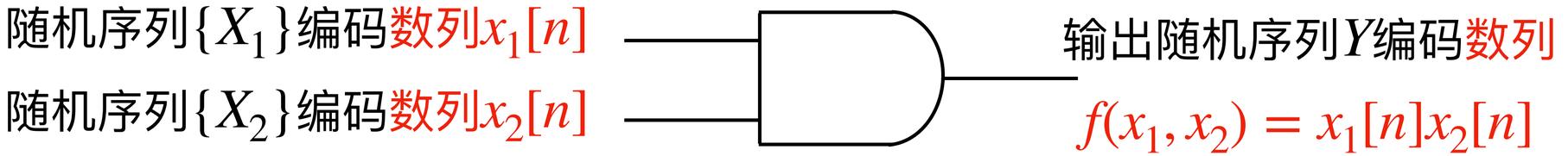


动态随机计算电路

- 组合逻辑电路

假设一个组合逻辑电路，实现随机计算函数 $f(x_1, x_2, \ldots)$，其中，$\{X_1\}, \{X_2\}, \ldots$分别为传统随机序列，编码 $x_1, x_2, \ldots$；则当$\{X_1\}, \{X_2\}, \ldots$为动态随机序列编码数字信号 $x_1[n], x_2[n], \ldots$时，该组合逻辑实现复合函数计算 $f(x_1[n], x_2[n], \ldots)$。

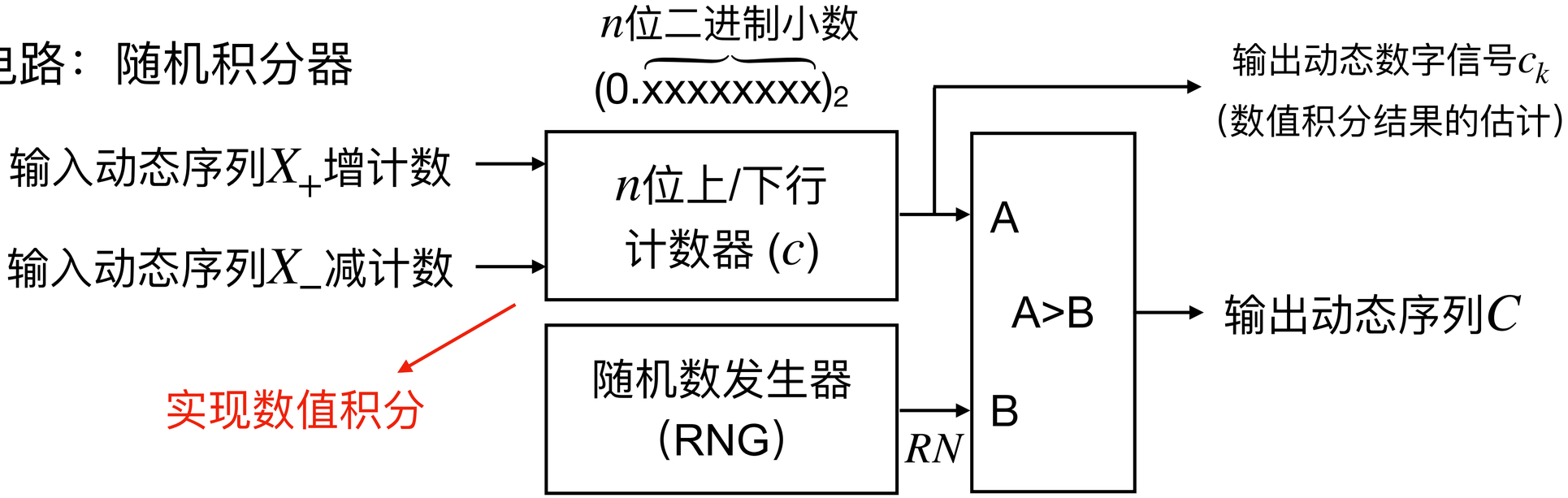
例：

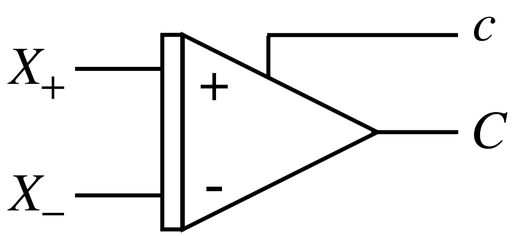


当$\{X_1\}$为动态随机序列，编码数字信号$x_1[n]$，根据动态随机序列定义，即$\mathbb{E}[X_{1,k}] = x_1[k]$；$\{X_2\}$为动态随机序列，编码数字信号$x_2[n]$，即$\mathbb{E}[X_{2,k}] = x_2[k]$，因此，输出序列中第$k$位，$\mathbb{E}[Y_k] = \mathbb{E}[X_{1,k} \wedge X_{2,k}] = \mathbb{E}[X_{1,k}]\mathbb{E}[X_{2,k}] = x_1[k]x_2[k]$，即$Y$编码数字信号$x_1[n]x_2[n]$。

动态随机计算电路

- 时序电路：随机积分器





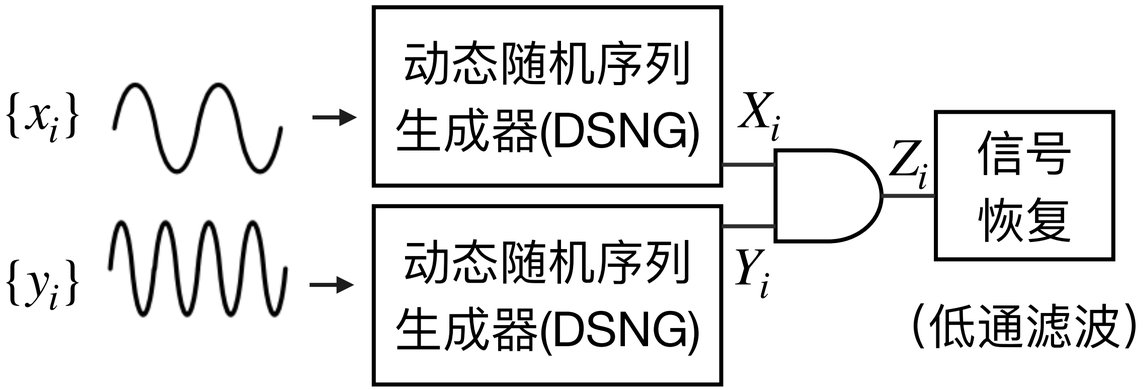
随机积分器符号

1. 在每个时钟周期，计数器值更新：$c_{i+1} = c_i + \frac{1}{2^n}(X_{+,i} - X_{-,i})$，即

$$c_i = \frac{1}{2^n}\sum(X_{+,k} - X_{-,k})，且\mathbb{E}[c_i] = \frac{1}{2^n}\sum(x_+[k] - x_-[k])。$$

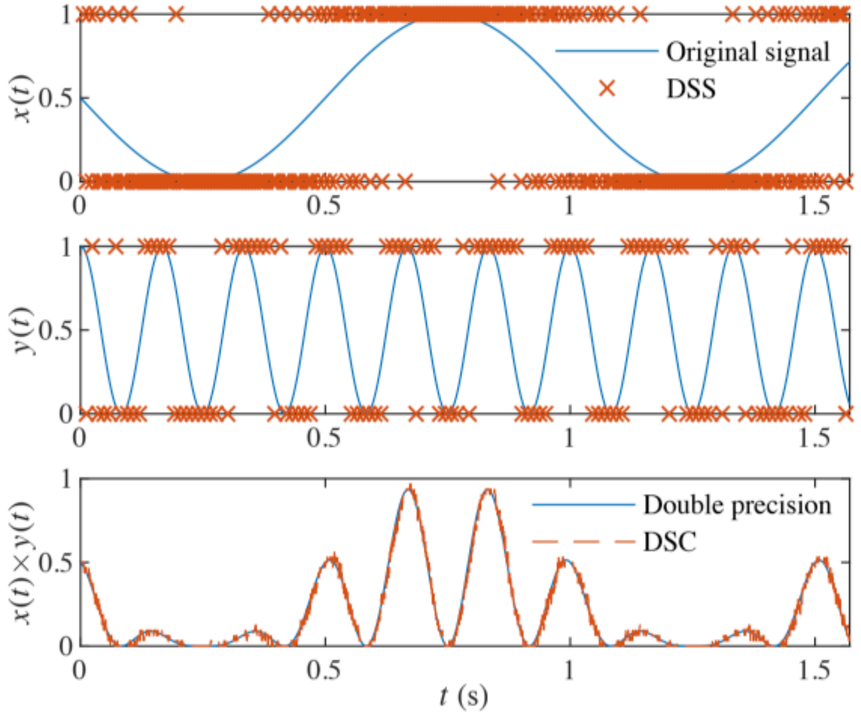
2. 随后经过更新后的计数器值与一个随机数*RN*比较产生编码c的动态随机序列*C*，即RNG与比较器构成动态随机序列生成器。

课题1: 基于动态随机计算的数字信号处理



$$\mathbb{E}[X_i] = x_i,\ \mathbb{E}[Y_i] = y_i \rightarrow \mathbb{E}[Z_i] = \mathbb{E}[X_i]\mathbb{E}[Y_i] = x_i y_i$$

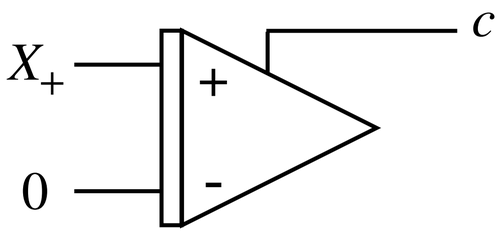
动态随机序列$\{Z_i\}$编码的信号为 $z_i = x_i y_i$

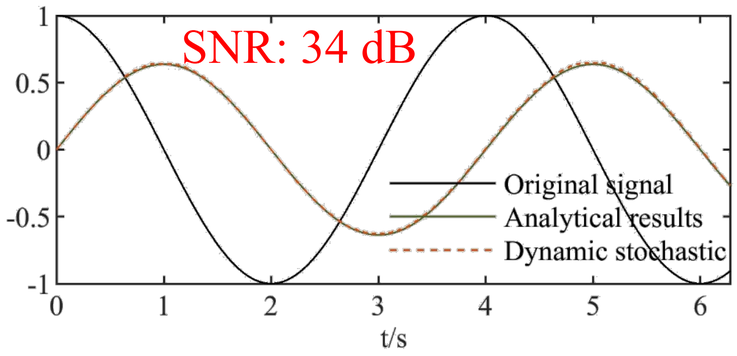


	面积 (nm^2)	功耗 (uW)	计算时钟数	计算时间 (ns)	能耗 (fJ)	信噪比 (dB)
动态随机计算	81.76	12.70	1	0.70	50.79	24.20
传统随机计算	66.59	10.27	32	9.28	1314.85	23.30
定点数计算	79.64	13.79	1	0.72	55.18	20.84
比例 (Dyn.:Conv.)	1.23	1.24	0.031	0.075	0.039	--

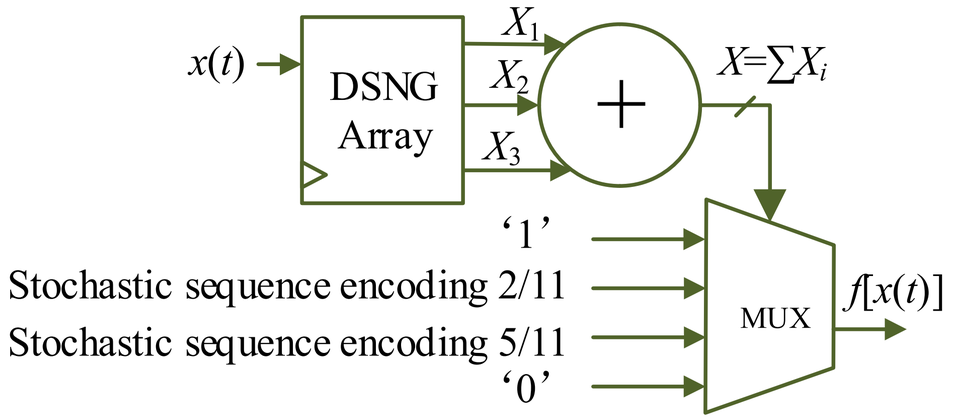
基于动态随机计算的其它信号处理

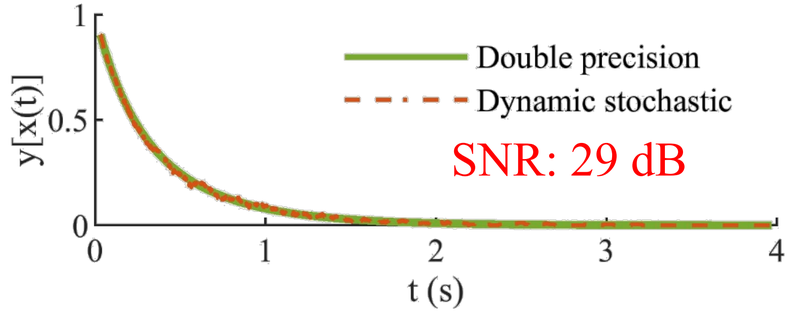
- 利用随机积分器实现数字信号数值积分



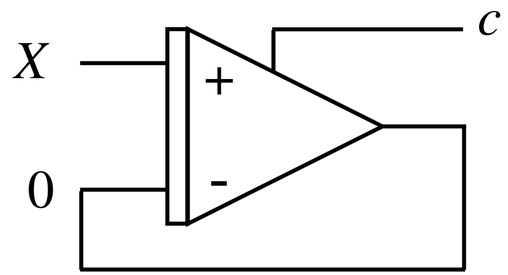


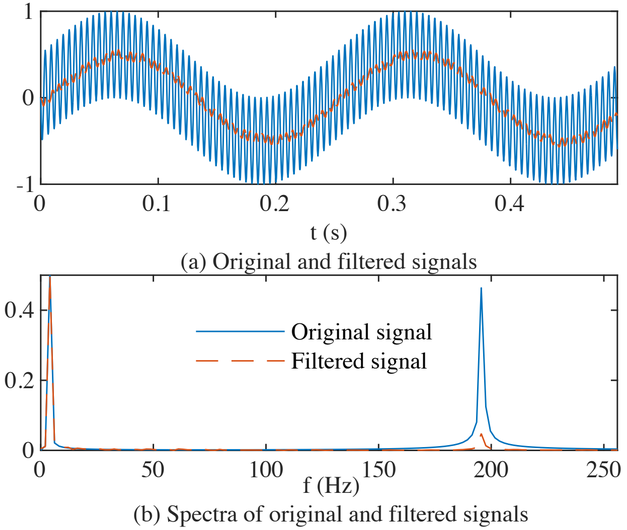
- 基于多路器的信号发生功能





- 基于随机积分器的IIR滤波





信号发生功能，利用已有指数函数生成幂次方累加

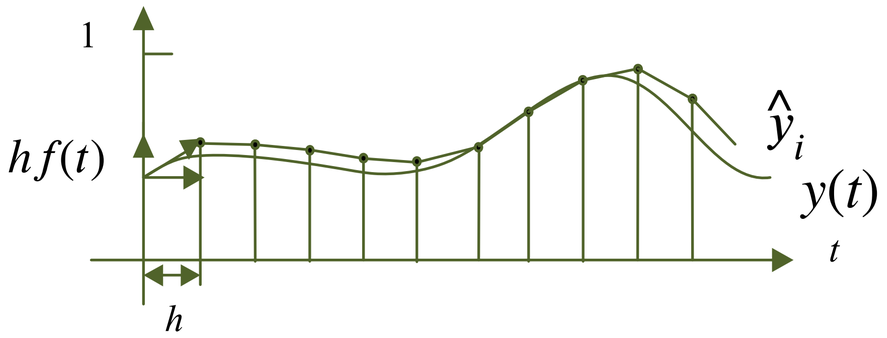
信号 $f\left[x(t)\right] = 1/11(2e^{-6t} + 3e^{-4t} + 6e^{-2t})$.

课题2: 基于动态随机计算的微分方程求解

- 常微分方程

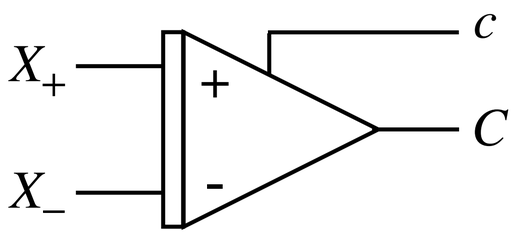
$$\frac{\mathbf{d}y(t)}{\mathbf{d}t} = f(t)$$

- 欧拉方法 (Euler's method)：通过对离散导函数的不断累加得到数值解



$$\hat{y}_i = h\sum_{k=0}^{i-1} f(t_k)$$

- 基于随机积分器的微分方程求解电路



输出序列C是一个动态随机序列，用于编码信号$\{c_i\}$。

- 如果输入动态随机序列X_+与X_-编码数字信号$\mathbf{d}y(t)/\mathbf{d}t$或$f(t)$，即，

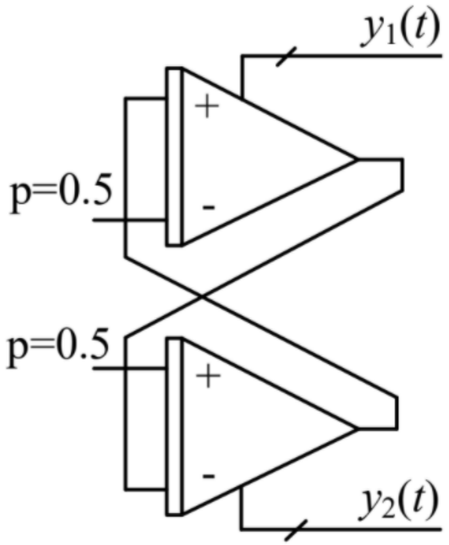
$$\mathbb{E}[X_{+,i} - X_{-,i}] = f(t_i)。$$

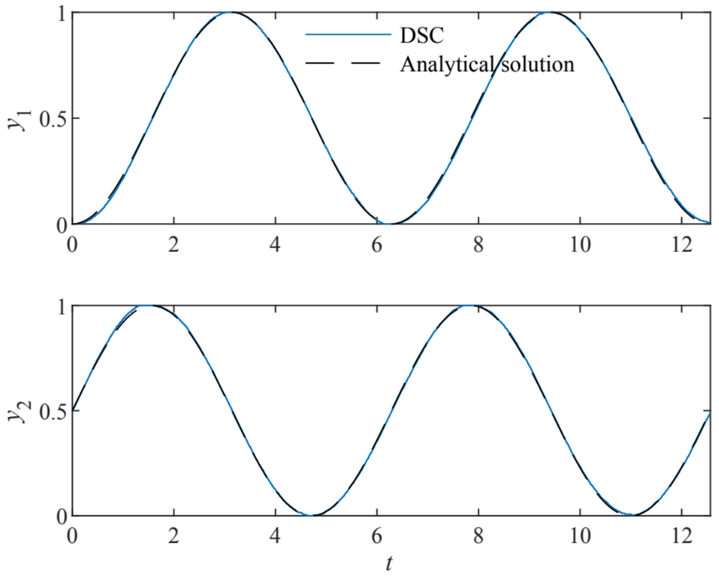
随机积分器即提供该微分方程的欧拉解的无偏估计。其中步长h取决于积分器中计数器宽度。

典型常微分方程求解

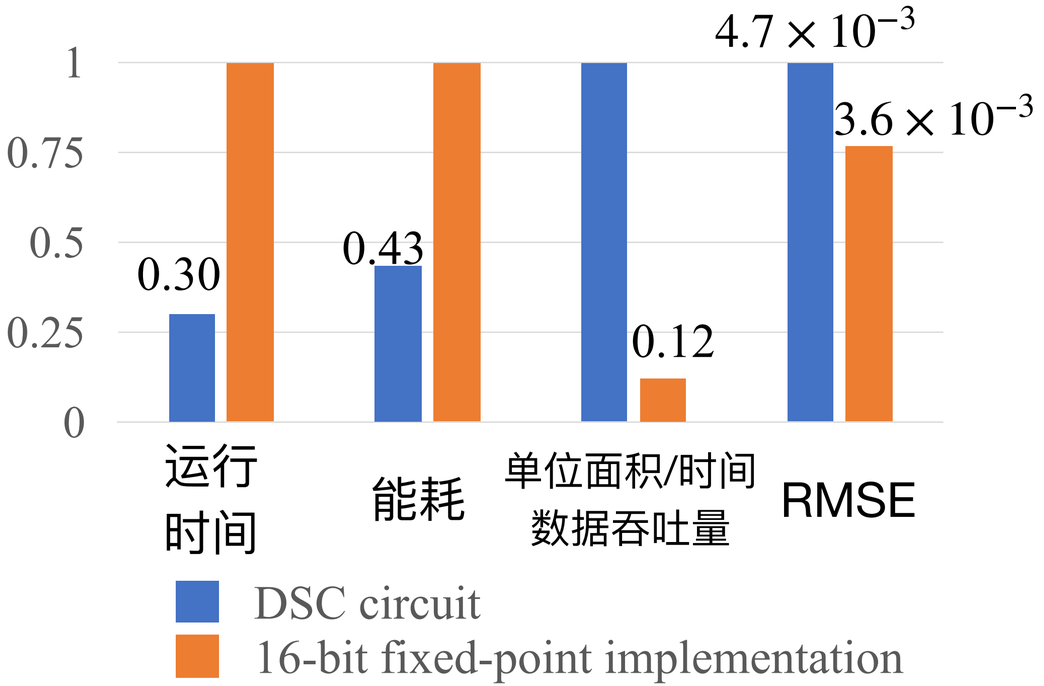
- 简单的微分方程系统

$$\begin{cases} \dfrac{\mathrm{d}y_1(t)}{\mathrm{d}t} = y_2(t) - 0.5 \\ \dfrac{\mathrm{d}y_2(t)}{\mathrm{d}t} = 0.5 - y_1(t) \end{cases}$$





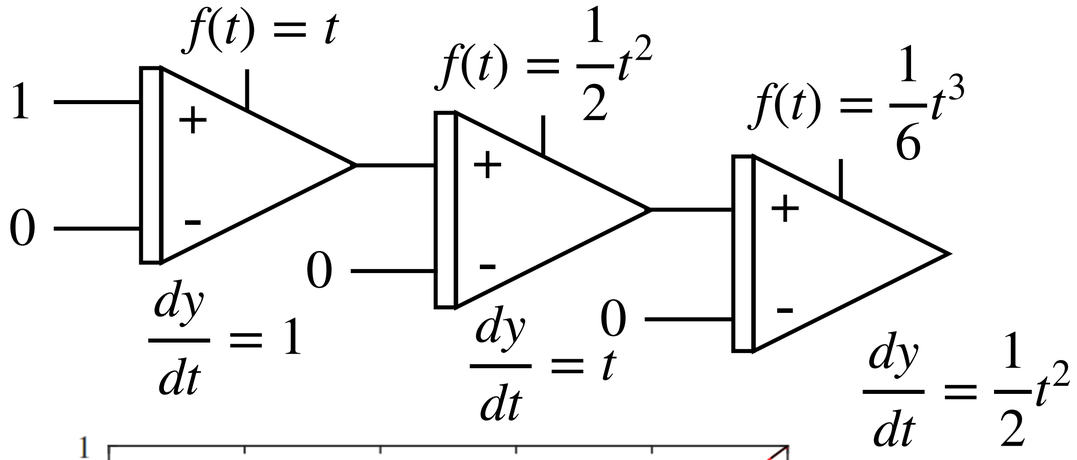
归一化硬件电路测试结果

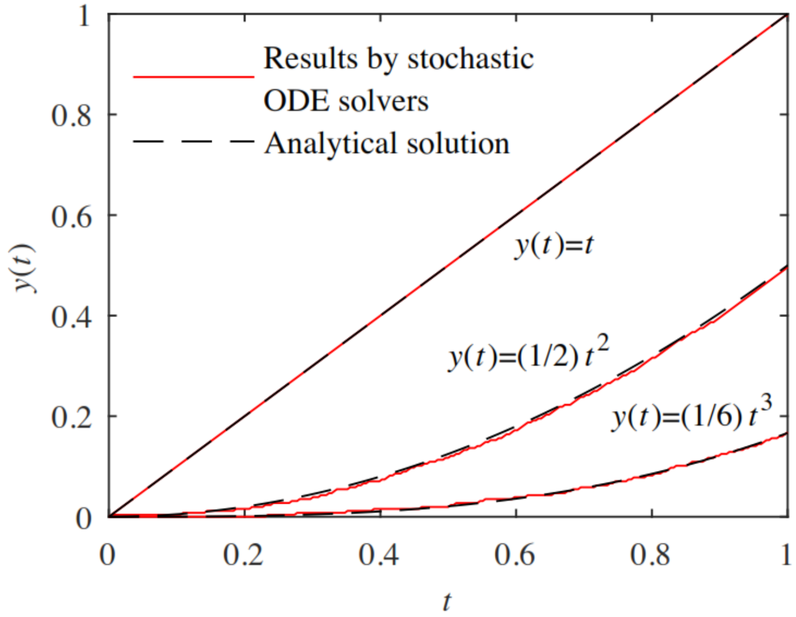


动态随机（DSC）电路与16位定点二进制电路的比较

典型常微分方程求解

- 非齐次微分方程



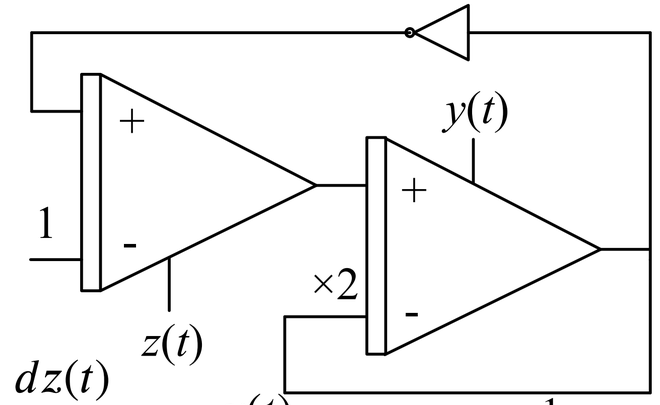


RMSE:0

RMSE:
5.7×10^{-3}

RMSE:
3.9×10^{-3}

- 二阶齐次微分方程

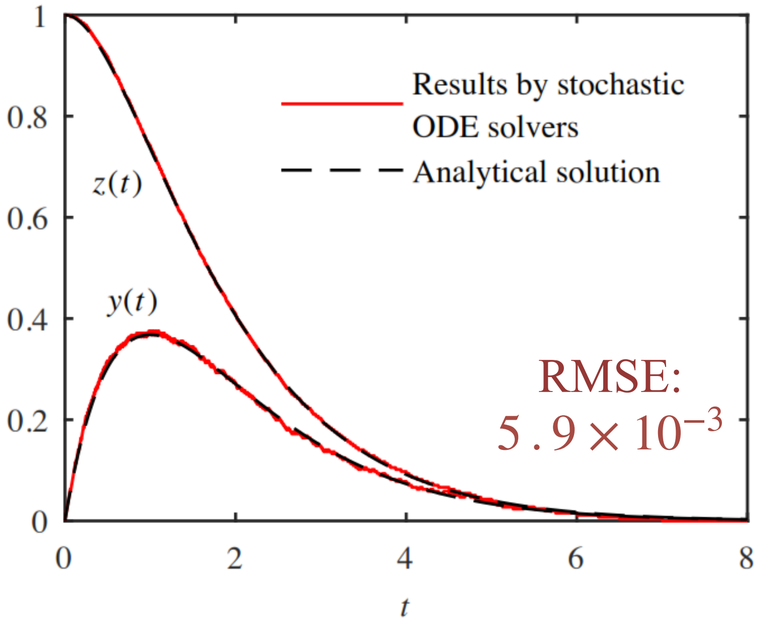


$$\frac{d^2y(t)}{dt}+\frac{2dy(t)}{dt}+y(t)=0$$

$$\frac{dy(t)}{dt}=z(t)-2y(t)$$

$$\frac{dz(t)}{dt}=-y(t)$$

$$c_{i+1}=c_i+\frac{1}{2^N}(a_i-2b_i)$$



偏微分方程求解——拉普拉斯方程

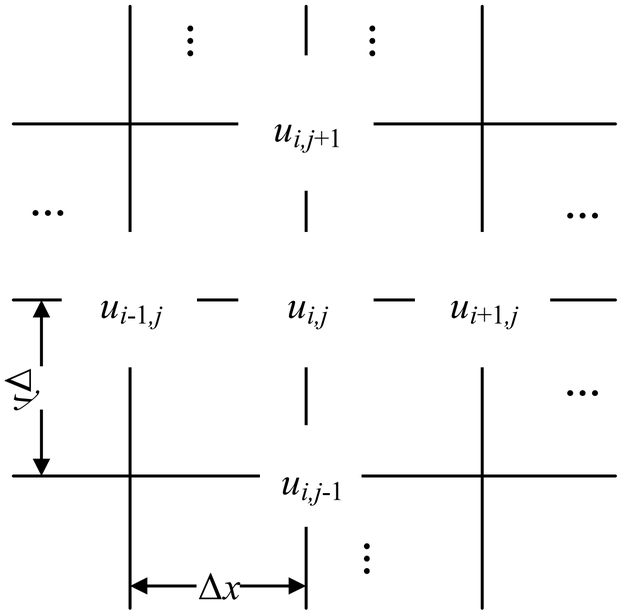
$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \quad \Rightarrow \quad \nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \mathrm{d}u/\mathrm{d}t$$

$$\Downarrow$$

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{\Delta x^2} \quad \Rightarrow \quad \nabla^2 u = \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{\Delta x^2} + \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{\Delta y^2} = \mathrm{d}u/\mathrm{d}t$$

$$\Downarrow$$

有限差分法



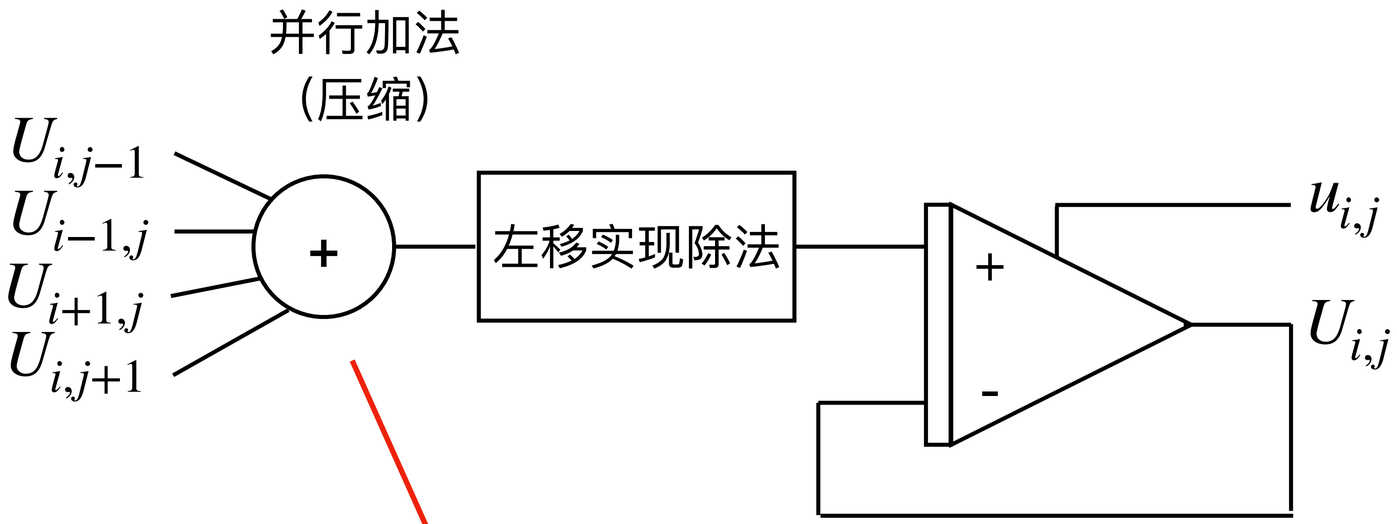
$$\Delta x = \Delta y \Rightarrow \boxed{u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{i,j} = \mathrm{k}\mathrm{d}u/\mathrm{d}t \quad (i = 1,\ldots,N,\ j = 1,\ldots,N)}$$

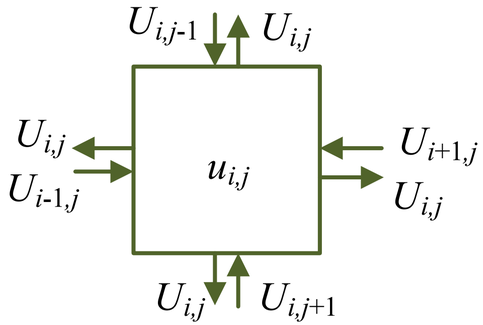
当$\frac{\mathbf{d}u}{\mathbf{d}t} = 0$时，所得即为拉普拉斯方程的稳态解。

可通过对以上微分方程求解得出

$$u_{i,j}[n] = 4\Delta x^2 \mathbf{d}t \sum_{k=0}^{n-1} \frac{1}{4}\{u_{i-1,j}[k] + u_{i+1,j}[k] + u_{i,j-1}[k] + u_{i,j+1}[k] - 4u_{i,j}[k]\}.$$

动态随机拉普拉斯算子实现

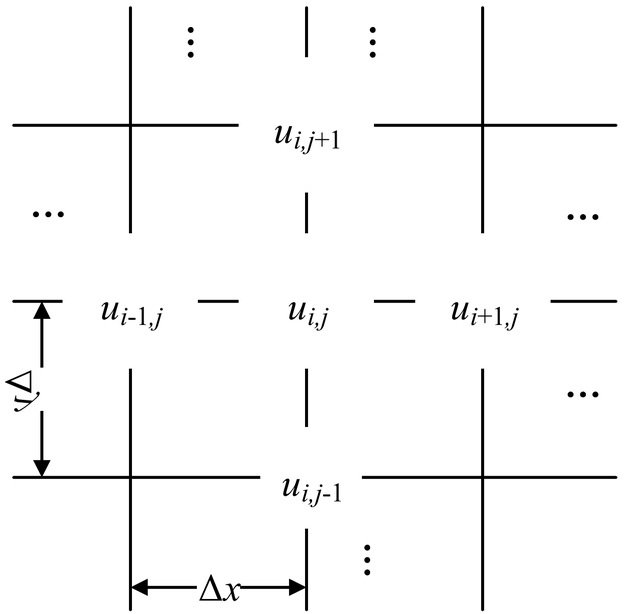


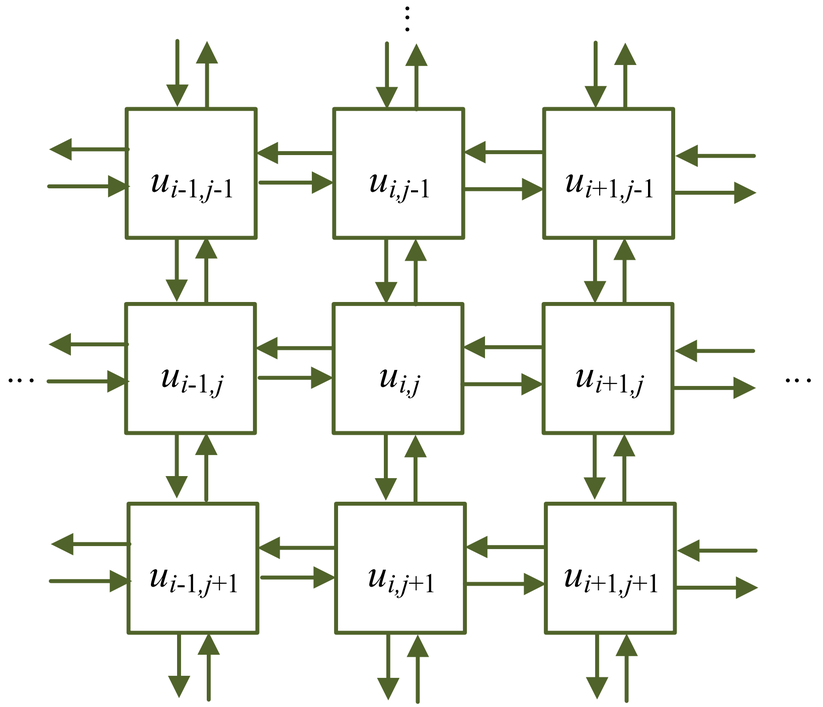


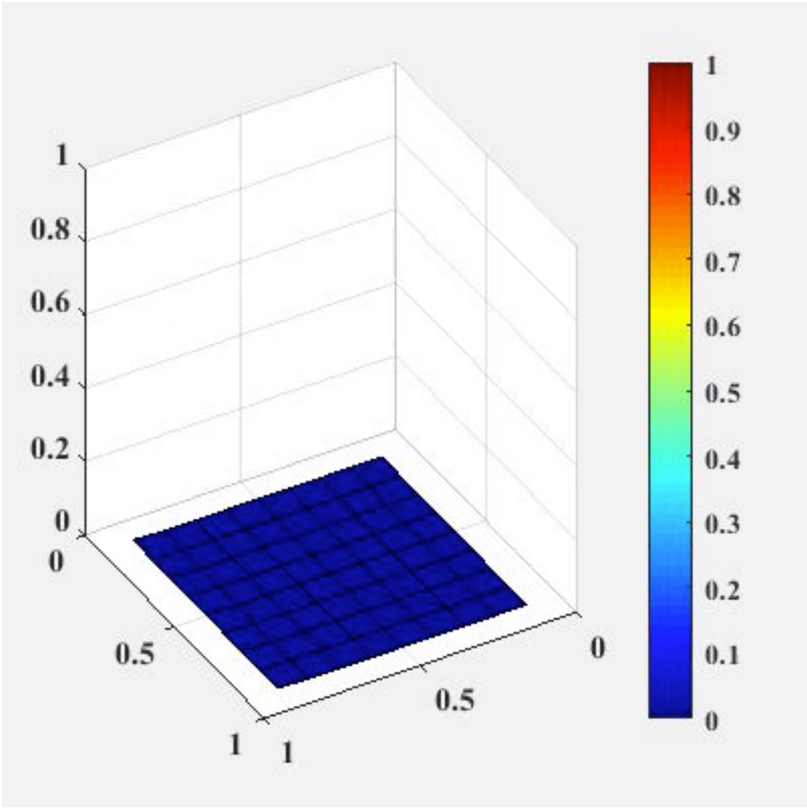
动态随机拉普拉斯
算子符号

$$u_{i,j}[n] = 4\Delta x^2 \mathbf{d}t \sum_{k=0}^{n-1} \frac{1}{4}\{u_{i-1,j}[k] + u_{i+1,j}[k] + u_{i,j-1}[k] + u_{i,j+1}[k] - 4u_{i,j}[k]\}.$$

动态随机拉普拉斯算子网络求解拉普拉斯方程



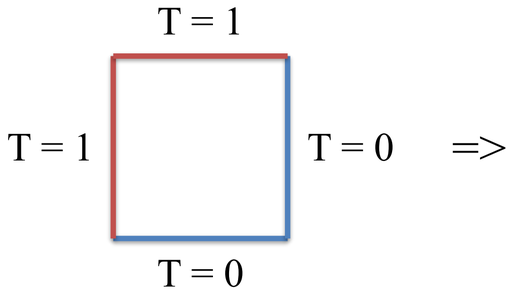




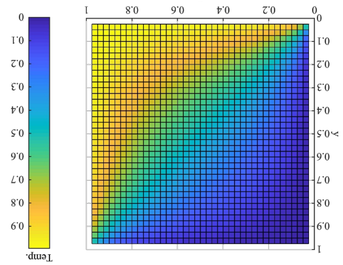
- 拉普拉斯算子电路网络被用于在区域$[0, 1]^2$中求解拉普拉斯方程,

$$0 = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$

$$\text{边界条件 } u(x, y) = \begin{cases} 1 & \text{when } y = 0 \text{ or } x = 1 \\ 0 & \text{when } y = 1 \text{ or } x = 0 \end{cases}$$

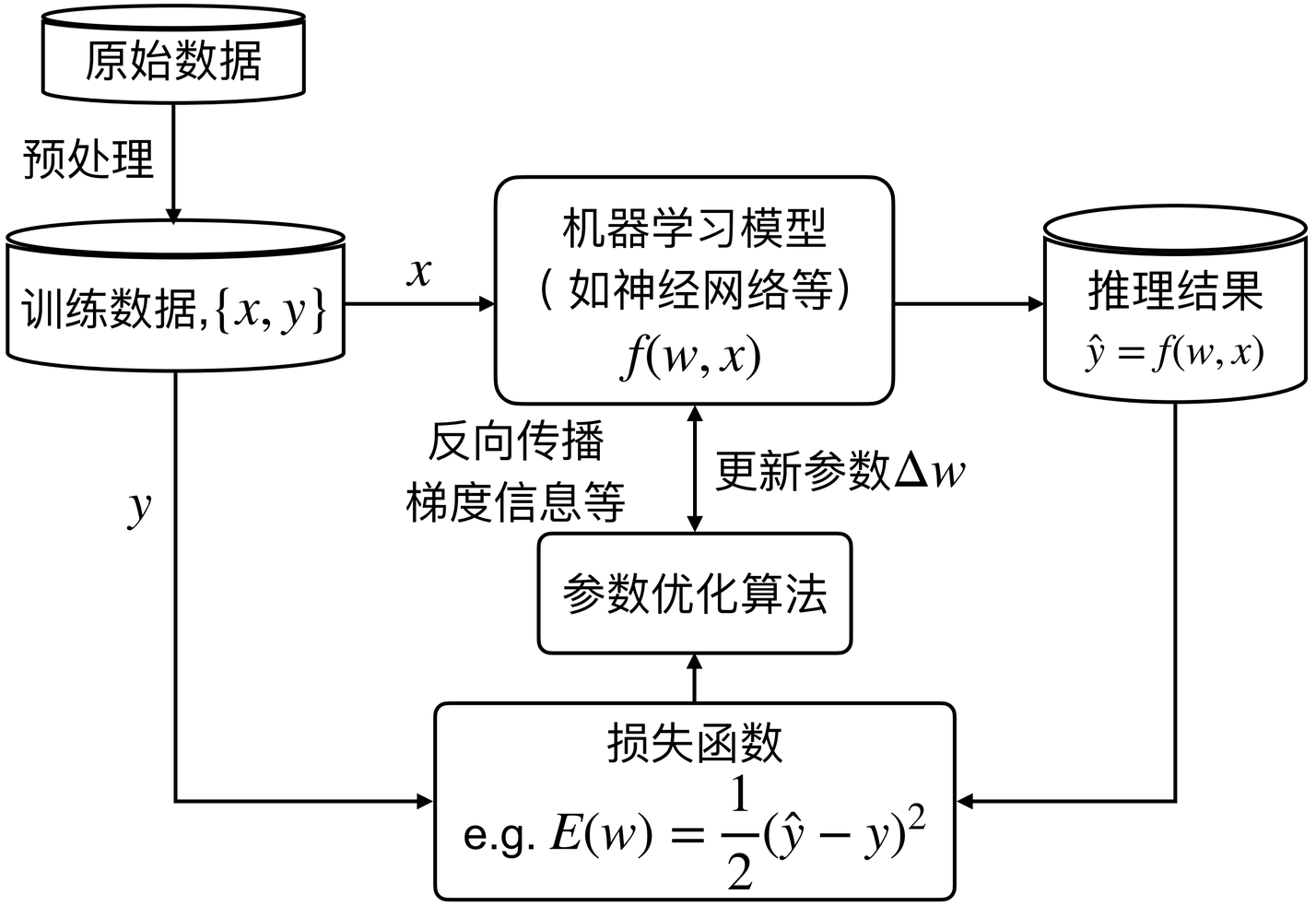


=>

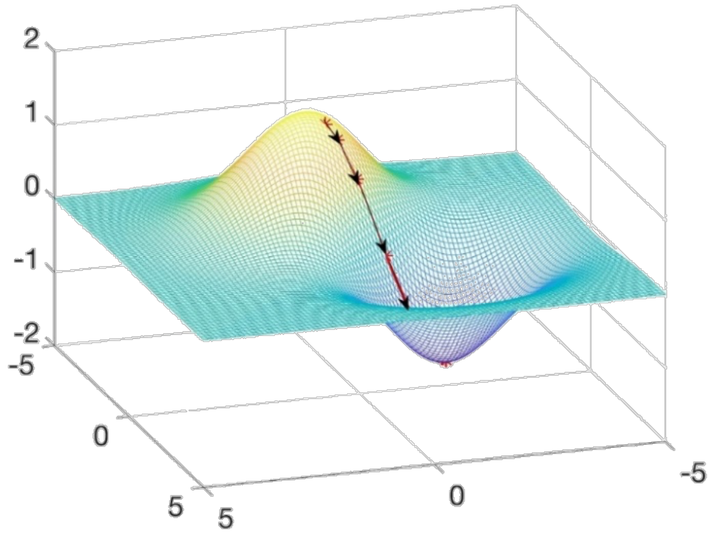


课题3: 基于动态随机计算的神经网络参数更新

- 监督学习



- 梯度下降



- 损失函数对待优化参数的偏导数

$$\frac{\partial E}{\partial w} = \frac{\partial \hat{y}}{\partial w}(\hat{y} - y)$$

- 参数更新

$$w_{i+1} = w_i - \mu \frac{\partial E_i}{\partial w} = w_i - \mu \frac{\partial \hat{y}}{\partial w}(\hat{y}_i - y_i)$$

基于动态随机计算的梯度下降电路

- 损失函数对待优化参数的偏导数

$$\frac{\partial E}{\partial w}=\frac{\partial \hat{y}}{\partial w}(\hat{y}-y)$$

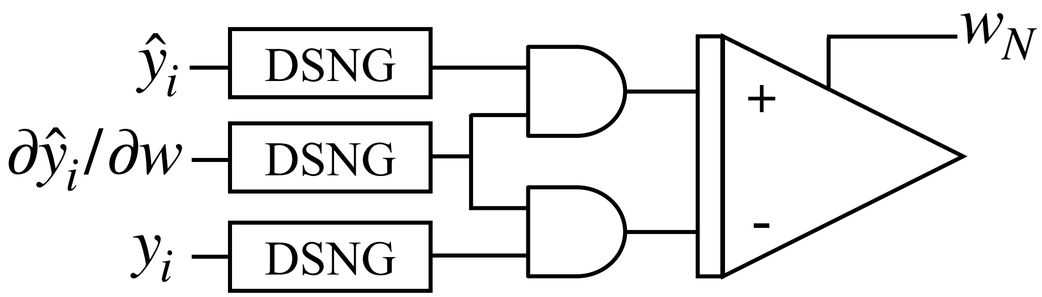
- 参数更新

$$w_{i+1}=w_i-\mu\frac{\partial E}{\partial w}=w_i-\mu\frac{\partial \hat{y}}{\partial w}(\hat{y}_i-y_i)$$

$$=w_i+\mu\left(\frac{\partial \hat{y}_i}{\partial w}\hat{y}_i-\frac{\partial \hat{y}_i}{\partial w}y_i\right)$$

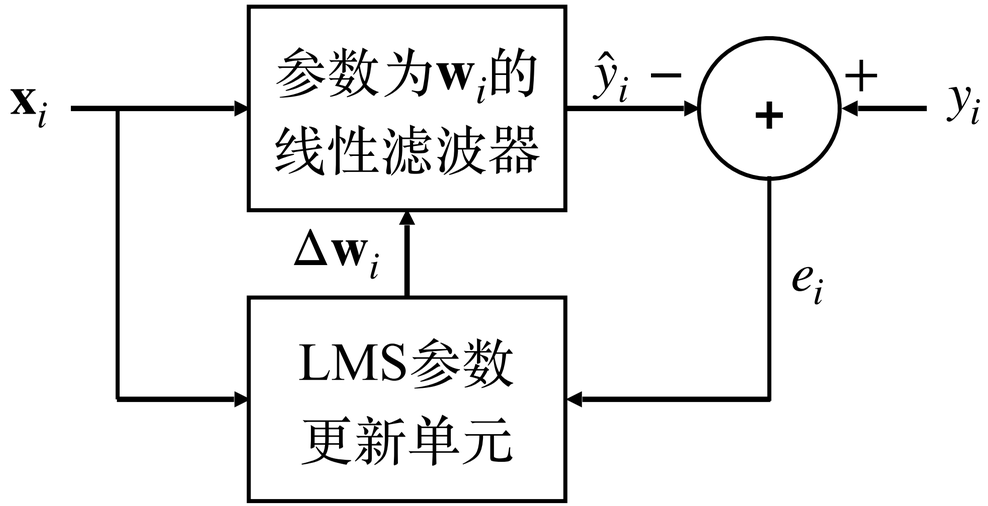
$$w_N=w_0+\mu\sum_{i=0}^{N-1}\left(\frac{\partial \hat{y}_i}{\partial w}\hat{y}_i-\frac{\partial \hat{y}_i}{\partial w}y_i\right)$$

随机积分器　随机乘法器　随机乘法器



基于DSC的梯度下降电路原型设计

基于LMS的自适应滤波参数更新



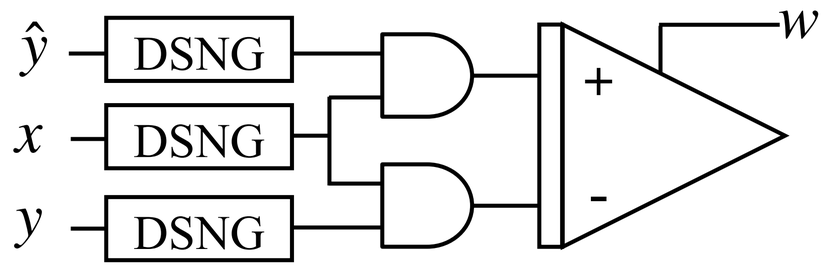
- 线性滤波器

$$\hat{y}_i = \mathbf{w}_i \mathbf{x}_i$$

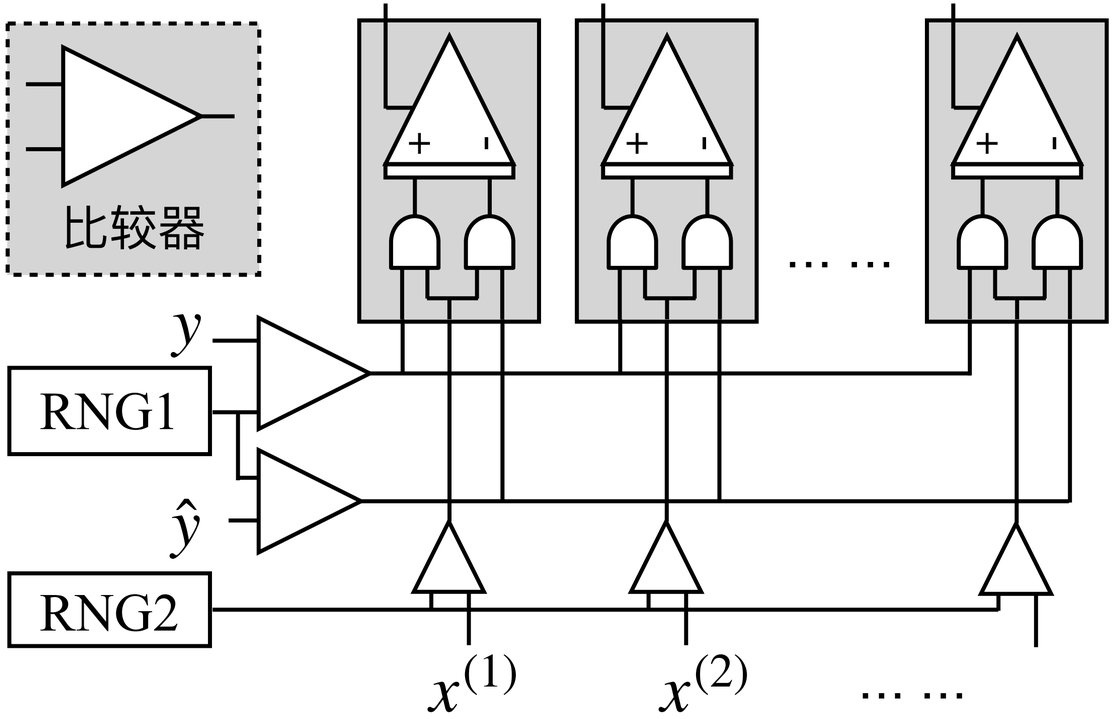
- 最小均方（Least Mean Square）参数更新

$$w_{i+1} = w_i + \mu\left(\frac{\partial \hat{y}_i}{\partial w}\hat{y}_i - \frac{\partial \hat{y}_i}{\partial w}y_i\right)$$

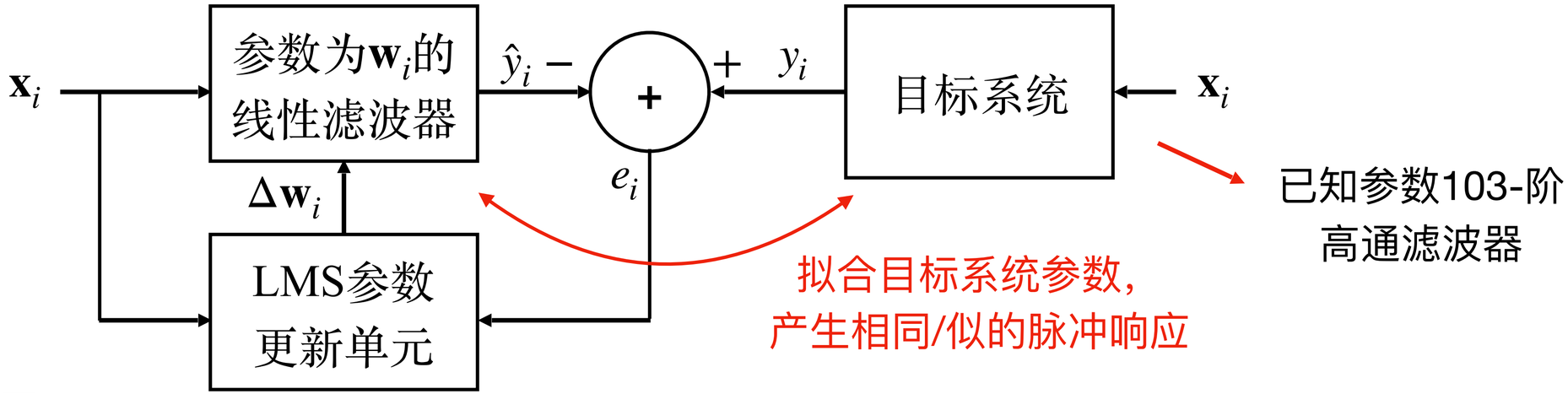
$$= w_i + \mu(x_i\hat{y}_i - x_i y_i)$$



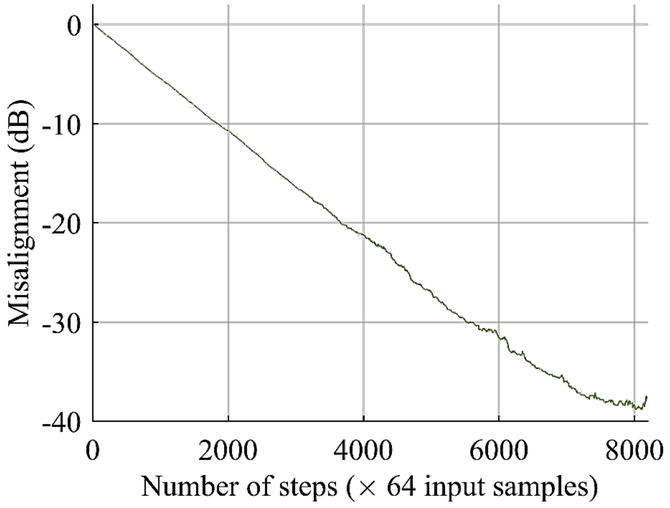
随机梯度下降电路更新单个参数

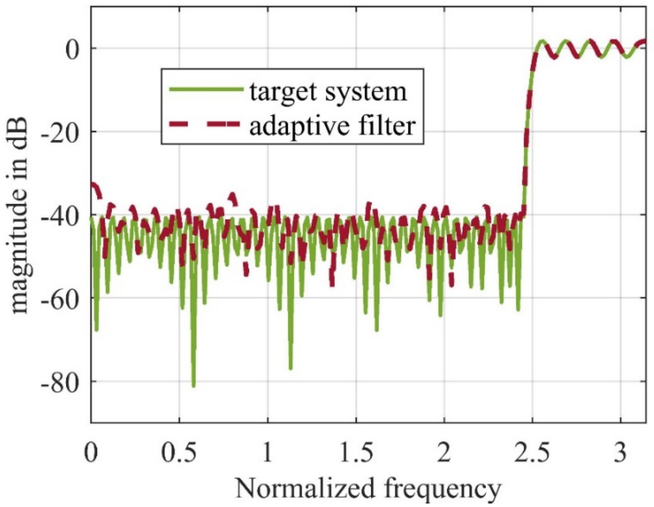


应用：基于自适应滤波器的系统识别



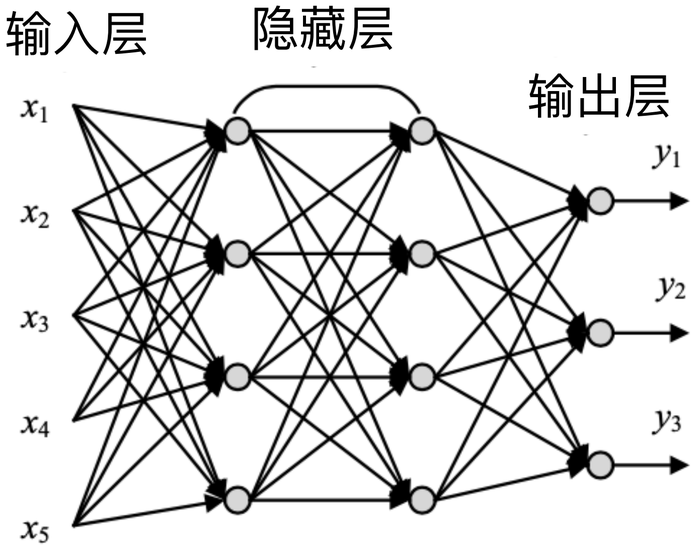
- 实验结果



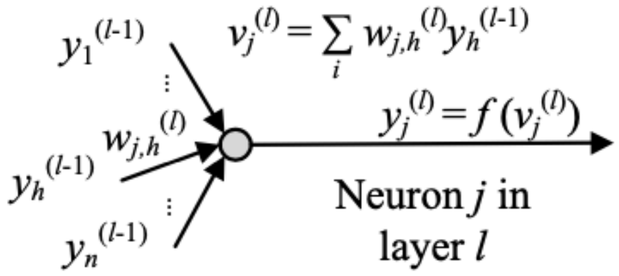


	DSC	传统SC
计算时间 (us)	6.0×10^{1}	6.5×10^{4}
总能耗 (nJ)	7.0×10^{2}	1.5×10^{6}
单位面积/时间吞吐量 (Sa./us/um^2)	6.7×10^{-5}	1.0×10^{-8}
失调 (dB)	-23	-6

基于动态随机计算的神经网络参数更新



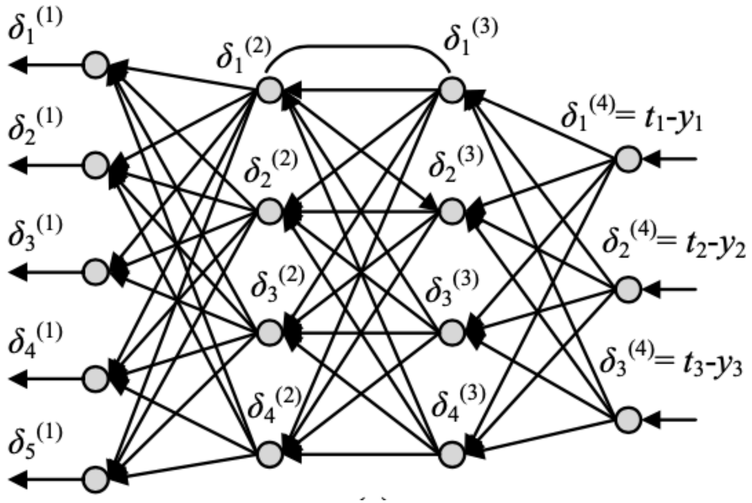
- 前向传播



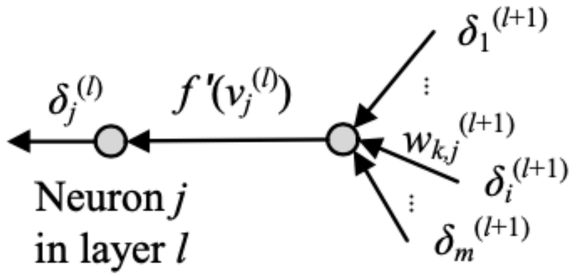
$$\delta_j^{(l)} = \delta_{j,+}^{(l)} - \delta_{j,-}^{(l)}$$

由期望输出值产生　由实际输出值产生

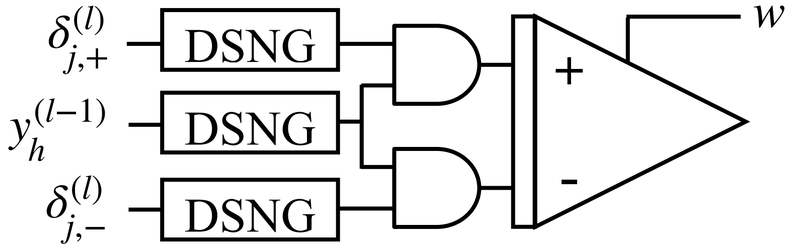
$$\Delta w_{j,h}^{(l)} = -\delta_j^{(l)} y_h^{(l-1)} = -(-\delta_{j,+}^{(l)} + \delta_{j,-}^{(l)}) y_h^{(l-1)}$$



- 反向传播



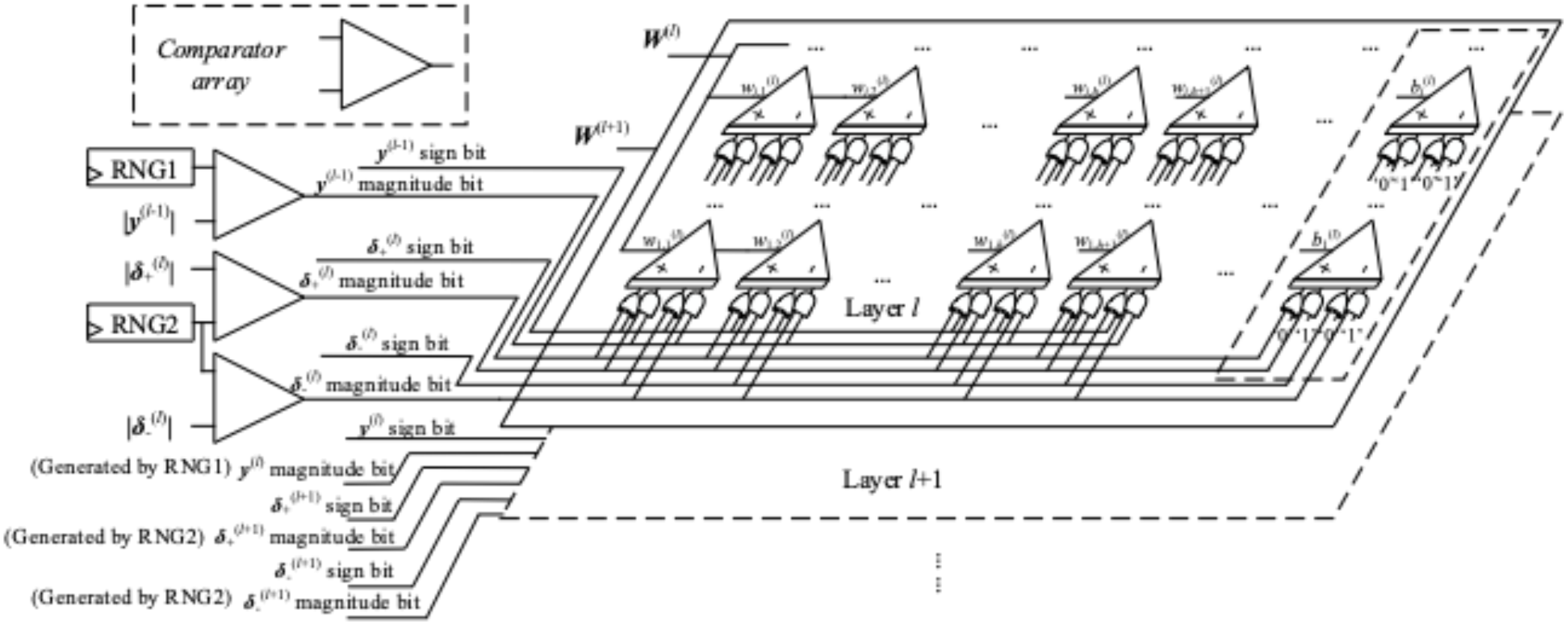
$$\Delta w_{j,h}^{(l)} = \frac{\partial E}{\partial w_{j,h}^{(l)}} = -\delta_j^{(l)} y_h^{(l-1)}$$



动态随机电路用于神经网络参数更新

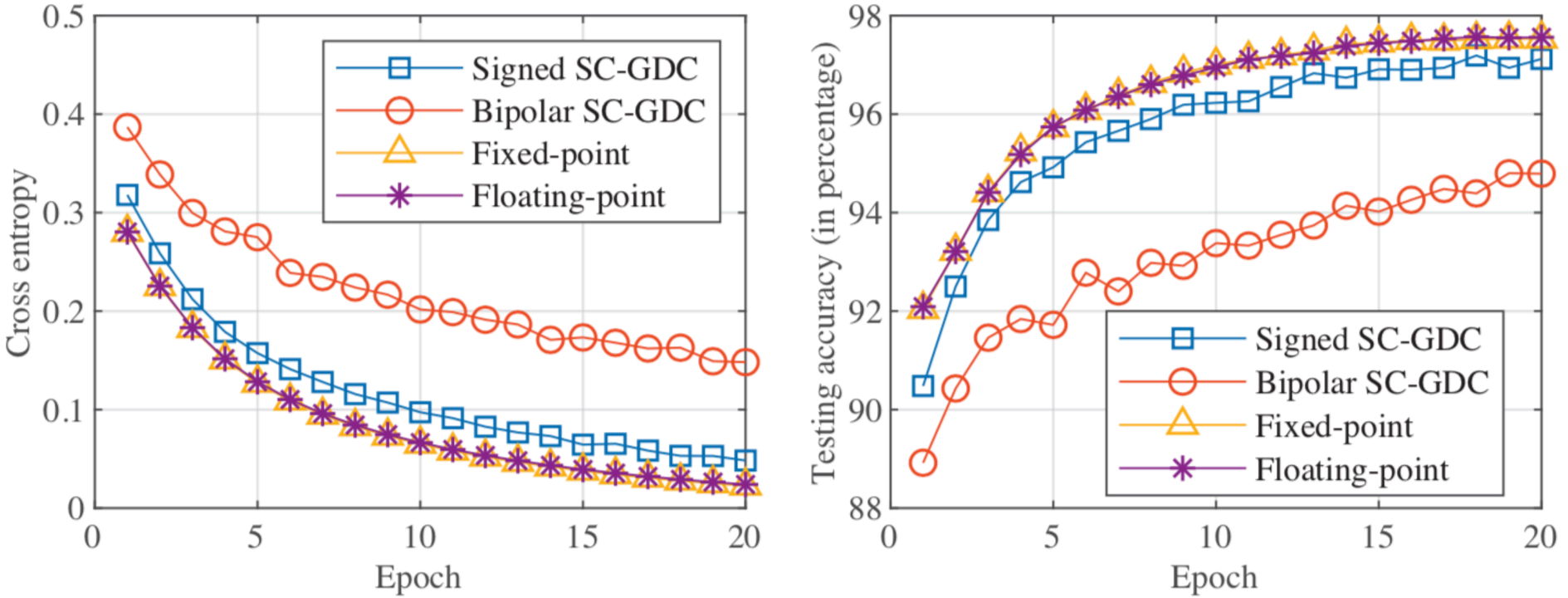
实验

- 实验采用一个全联接神经网络784-128-128-10对所设计电路进行测试，数据集采用MNIST



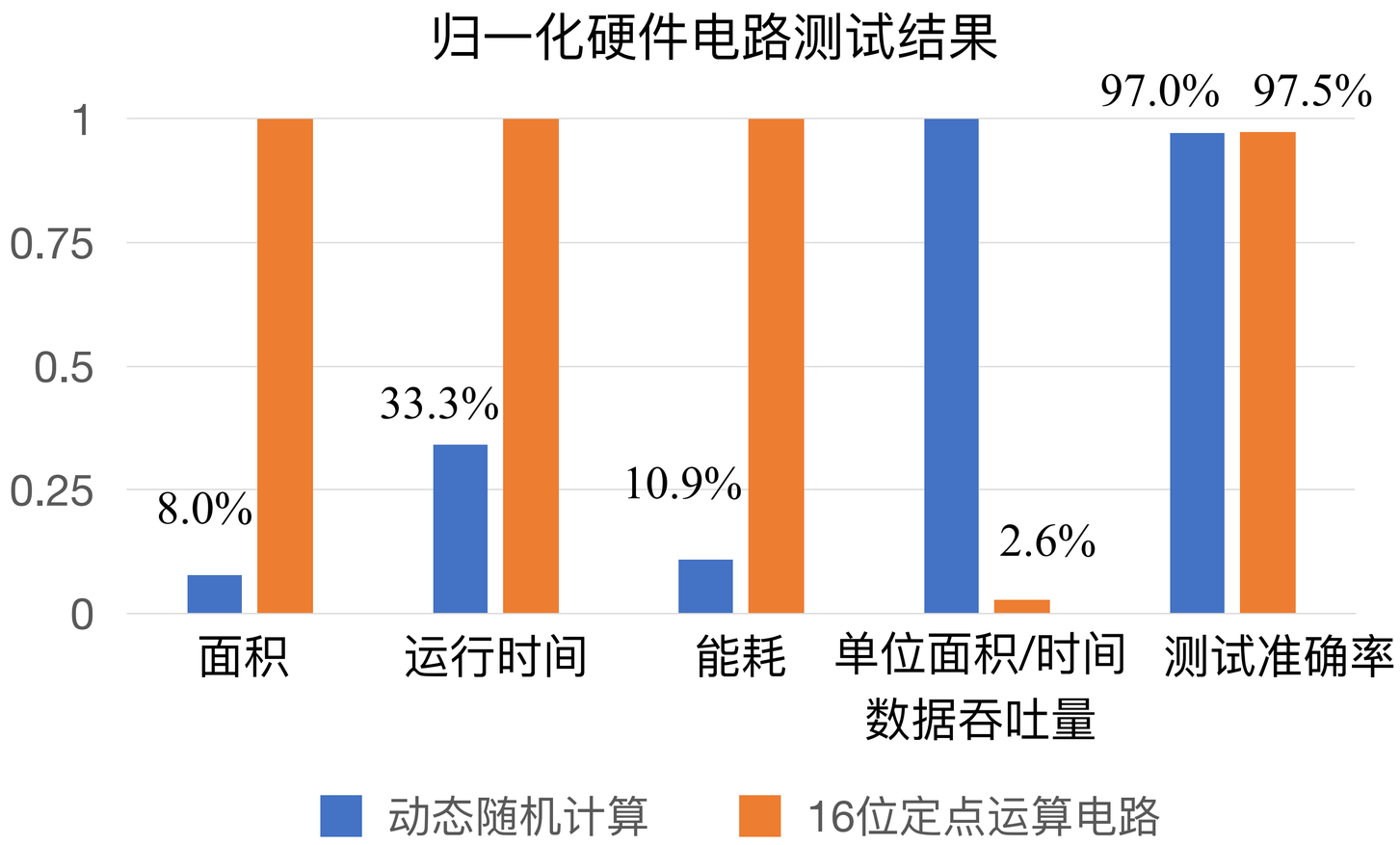
实验结果

- 计算准确度



由结果可见，采用有符号的动态随机计算 (DSC)取得与定点/浮点运算相似的收敛速度与测试准确率。

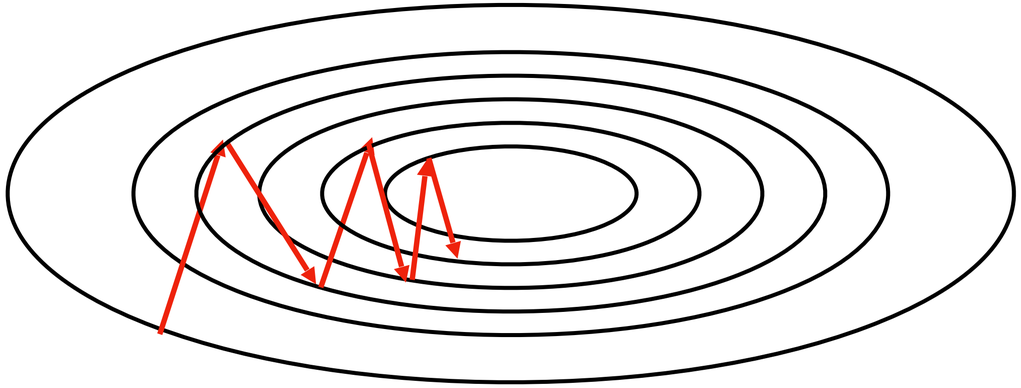
硬件性能评估



动态随机电路与16位定点二进制电路的比较

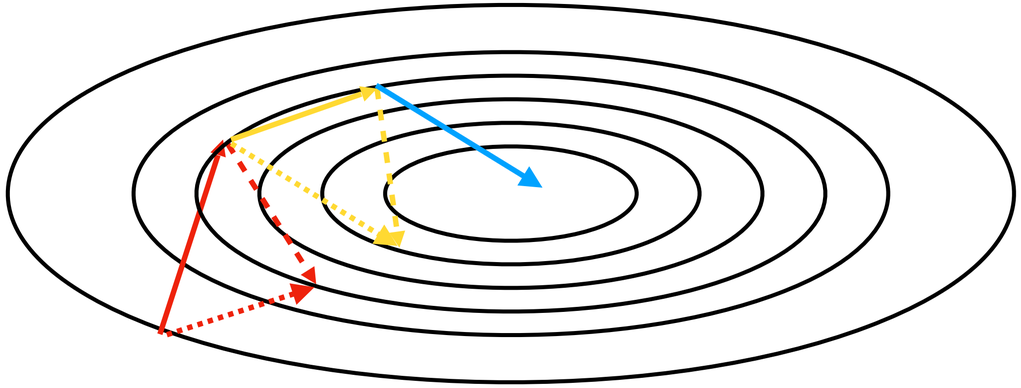
基于动态随机计算的动量梯度下降

- 动量梯度下降



原始梯度下降算法

$$w_{i+1} = w_i + \mu g_i$$



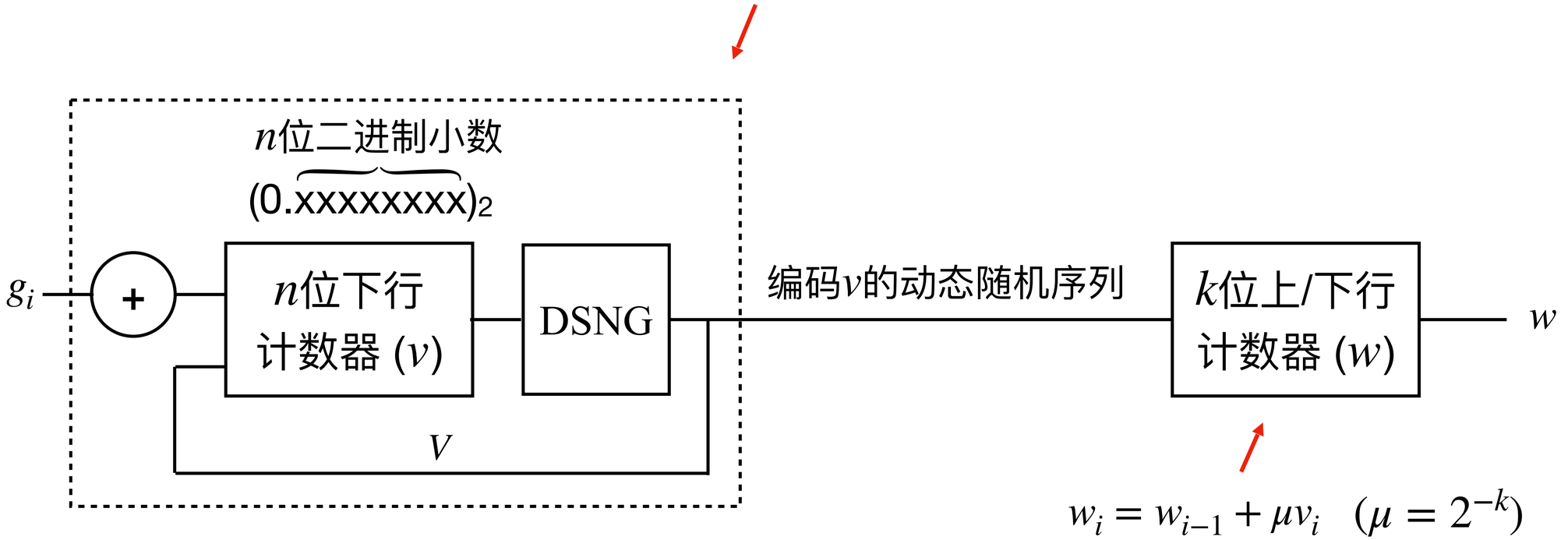
动量梯度下降算法

$$v_i = m v_{i-1} + g_i$$

$$w_i = w_{i-1} + \mu v_i$$

基于动态随机计算的动量梯度下降实现

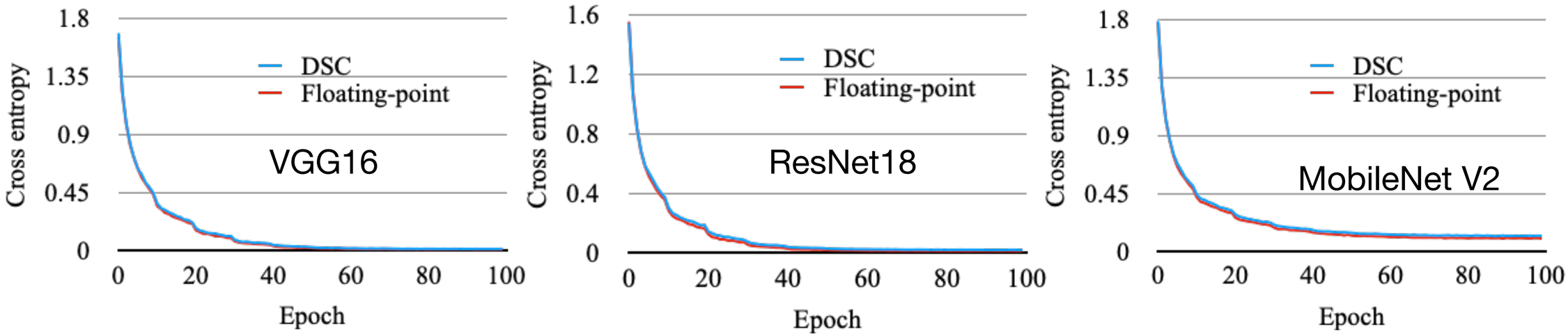
$$v_i = mv_{i-1} + g_i \quad => \quad v_i = v_{i-1} - (1-m)v_{i-1} + g_i \quad => \quad \text{令} 1-m = 2^{-n}$$



避免梯度信息的多次计算，可以更好的嵌入已有的机器学习框架

仿真实验

- 所设计电路被用于训练更为复杂的网络结构VGG16、ResNet18和MobileNet V2，使用CIFAR10作为训练数据集。学习率以系数0.5，每10次完整训练为周期衰减。



测试准确率 (%)	VGG16	ResNet18	MobileNetV2
动态随机计算（DSC）	90.23	91.36	88.51
浮点数运算	90.55	91.85	88.82

结论

- 动态随机计算可以使用动态随机序列配合相应逻辑电路实现复杂的功能，如数字信号处理、求解微分方程、神经网络参数更新等；
- 可以实现的算法中往往存在反复累加的操作，可以在不损失精度的情况下由随机积分器高效完成；
- 所设计的电路具有相比定点计算更高的性能和能效；
- 动态随机计算具有完成低能耗、高性能边缘计算的潜力。

主要参考文献

[1] Siting Liu, Warren J. Gross and Jie Han. “Introduction to Dynamic Stochastic Computing,” IEEE Circuits and Systems Magazine 20.3 (2020): 19-33.

[2] Siting Liu, Honglan Jiang, Leibo Liu and Jie Han. “Gradient Descent using Stochastic Circuits for Efficient Training of Learning Machines,” in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), vol. 37, no. 11, pp. 2530-2541, Nov. 2018.

[3] Siting Liu, and Jie Han. “Dynamic Stochastic Computing for Digital Signal Processing Applications.” 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp 1-6.

[4] Siting Liu, and Jie Han. “Hardware ODE solvers using stochastic circuits.” 2017 Proceedings of the 54th Annual Design Automation Conference (DAC), pp 1-6.

[5] Siting Liu, and Jie Han. “Energy efficient stochastic computing with Sobol sequences.” 2017 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1-4.

[6] Siting Liu, and Warren J. Gross. “Gradient descent with momentum using dynamic stochastic computing.” 2021 Hardware Aware Efficient Training (HAET) Workshop in International Conference on Learning Representations (ICLR) (accepted), pp. 1-5.

[7] Y. Liu, S. Liu, Y. Wang, F. Lombardi and J. Han, "A Survey of Stochastic Computing Neural Networks for Machine Learning Applications," in IEEE Transactions on Neural Networks and Learning Systems (early access).

[8] Warren J. Gross, and Vincent C. Gaudet, eds. Stochastic Computing: Techniques and Applications. Springer International Publishing, 2019.

http://www.ece.ualberta.ca/~jhan8/

Jie Han
(PhD, Delft 2004; BSc, Tsinghua 1999)

Professor
Department of Electrical and Computer Engineering
Room 13-358, Donadeo Innovation Centre for Engineering
University of Alberta
Edmonton, AB, Canada T6G 1H9
Tel: 1-780-492-1361
Email: jhan8@ualberta.ca

http://www.isip.ece.mcgill.ca/

Warren J. Gross
Louis-Ho Faculty Scholar in Technological Innovation
Chair of the Department of Electrical & Computer Engineering
Professor
Department of Electrical & Computer Engineering
McConnell Engineering Building
3480 University Street, Room 633
Montreal, QC, Canada H3A 0E9
Email: warren.gross@mcgill.ca

谢谢

Q&A

siting.liu@mail.mcgill.ca