基于随机计算的乘累加计算电路

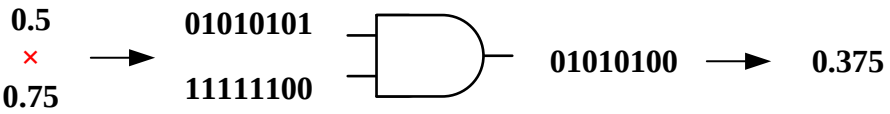
实验要求与目的

本实验要求实现基于随机计算的乘累加电路，并尝试多种不同的随机计算单元。阅读文献，设计实现基于 Sobol 序列的随机计算乘累加电路。本实验最多三人一组，实验大体要求列举如下，详情参考实验材料。

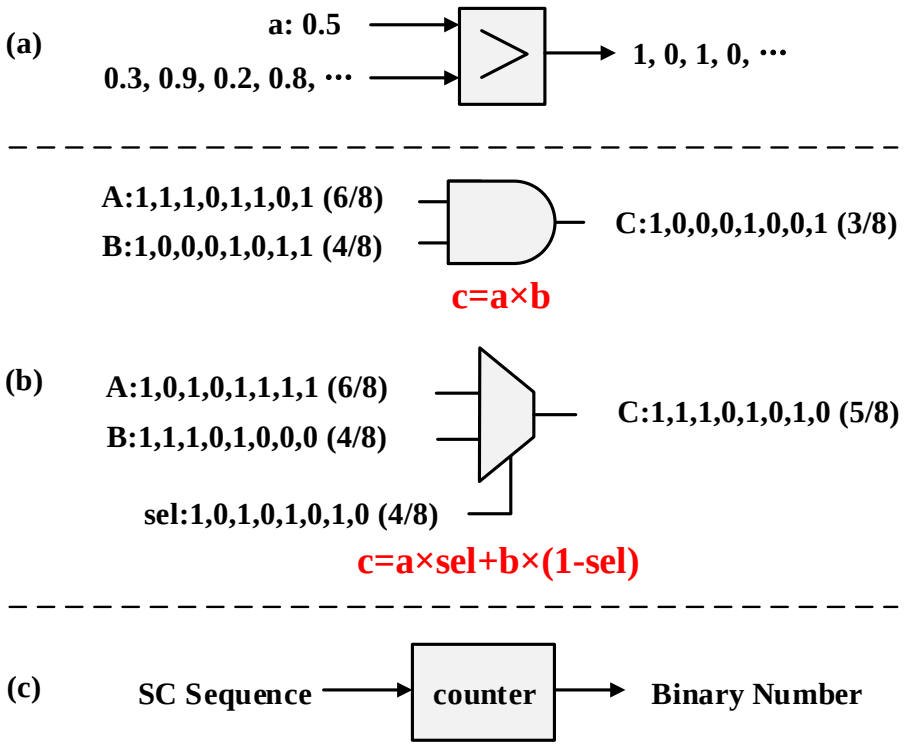
- MATLAB 实现 LFSR+MUX、Sobol+MUX 和 Sobol+ADD 共 **3 种** 8 输入 sc 乘累加计算单元，比较他们的精度。
- 设计并实现基于 Sobol 序列的 8 输入 sc 乘累加计算单元 A1，并提高其比特级并行度，设计实现并行架构 A2。

随机计算基础

深度神经网络（Deep Neural Network，DNN）广泛地被应用在计算机视觉、自然语言处理、语音识别等各个领域，且取得了显著的成就。然而随着网络深度的增加，其计算量急剧增加，越来越多的研究者开始关注 DNN 的硬件加速器设计。DNN 的乘累加运算占总运算量的 90%以上，由此导致的巨大硬件资源开销和功耗使得其在嵌入式设备等硬件资源受限的平台上的部署变得困难。另一方面，由于摩尔定律走到尽头，难以单纯依靠更先进的工艺来缩小芯片面积。因此，迫切需要新颖的替代计算范例克服这个障碍。随机计算（Stochastic Computing, SC）作为一种新型计算范式，为解决上述问题提供了可能。利用随机计算，可以用极低的硬件开销为代价实现一些常见的计算电路，例如乘法器和放缩加法器等。本实验将设计、实现、优化基于随机计算的乘累加电路。



SC 将[0,1]范围内的二进制数转化为一串随机的 0,1 序列表示，序列中 1 所占的比例即为数的大小。例如，序列{1,0,1,0}，{1,1,0,0}，{1,0,1,0,1,0,1,0}均可以表示数 0.5。应用 SC 后，原来很复杂的运算可以由简单的电路来实现。上图说明了 SC 乘法的具体过程：通过将二进制数输入（0.5，0.75）转化为 SC 序列后，逐比特通过与门得到输出序列，根据与门性质，输出的序列表示的数（0.375）即为两数相乘的结果。因此乘法在 SC 中仅需要一个与门即可实现，从而大大降低面积和功耗。



一个完整的 SC 系统可以大体分三个部分：**序列生成单元**，**计算单元**和**序列转化单元**。序列生成单元将二进制数转换为 SC 序列，计算单元实现系统内部的计算，序列转化单元将 SC 序列转化回二进制数。计算单元可能包含多个子模块，例如一个神经网络的一层，可能同时包含卷积、批量归一、池化等运算，这些运算中，如果所有参与运算的数均由 SC 序列表示，则称这个系统为**全 SC 域**的计算系统。如果计算单元中某些模块的操作数是二进制数，另外一些是 SC 序列，则称这个系统为 **SC-二进制域混合**计算系统。SC 作为一种近似计算方法，往往以牺牲精度为代价降低资源开销，因此全 SC 域的计算系统的优点往往是资源开销小；另一方面，SC-二进制混合计算系统往往有着精度更高的优势。。

图（a）给出了一个序列生成单元的例子，其中待转换的二进制数为 0.5，将他与[0,1]之间的随机数进行比较，随机数大于 0.5 则输出 1，反之输出 0，保证了输出序列中 1 的占比为 0.5。图（b）给出了 SC 域乘法和放缩加法的实现电路。其中 SC 的乘法仅需要一个与门实现，而带缩放的加法（c=(a+b)/2）仅需要一个 MUX 实现。图（c）是一个序列转换单元，他将 SC 序列转化为二进制数，通过用计数器统计序列中 1 的个数来实现。例如计数器统计到图(b)中乘法器的输出序列有 3 个 1，再考虑到序列长度为 8，那么最终输出的二进制数为 3/8。

序列的生成与相关性

SC 的准确性主要依赖于序列的相关性和序列的长度。**一般来讲，序列长度越长，精度越高**。当序列长度较长时，可以表示的概率的精度越高，例如长度为 8 的序列可以表示的最小概率为 1/8，而长度为 32 的序列可以表示的最小概率为 1/32。序列的相关性也是影响计算结果的重要因素。假设两个序列相关性极大，比如一个序列的取值是另一个序列的取反，那么两个序列通过与门得到的是一个全零序列，显然无法实现乘法运算。**一般而言，参与运算的序列间相关性为 0 时，运算精度最高**。

根据图（a）所示的序列生成单元的结构，可以知道序列间的相关性与产生序列时用到的随机数有着极大的关系。假设两个序列是通过相同的随机数产生的，那么两个序列间的相关性就特别高。为保证两个序列拥有的降低的相关性，需要保证他们使用的随机数之间的相关性较低。硬件实现随机数发生器往往采用线性反馈移位寄存器（LFSR）。为保证两个 LFSR 生成的随机数拥有较低相关性，往往采用不同的 LFSR 的结构。通常 n-bit 的定点数需要2^n长度的 SC 序列来确保准确率，此时需要的 LFSR 的长度也是 n 比特。LFSR 的相关内容请查阅资料，本文末尾也给出了一些 LFSR 的结构。

实验内容

本实验要求实现**一个带序列生成单元的 8 输入 SC 乘累加计算单元（两两相乘的积相加）**。本实验仅考虑**无符号数**的输入，输入二进制数位宽为 **5 比特**。

- 阅读参考文献[1]和[2]中关于 SC 的基本概念以及乘法和基于 MUX 的放缩加法的实现原理（文章其他部分可忽略）。
- 用 MATLAB 实现实验所需的 8 输入 SC 乘累加计算单元，其中随机数发生器采用 LFSR，加法采用 MUX，评估计算单元的精度。注意，该计算单元得到的结果是一个序列，统计精度的时候， 该序列需要转换回二进制数进行统计。评估精度的指标采用平均绝对误差（Mean Absolute Error，MAE）），定义为

$$\mathrm{MAE} = \frac{1}{n}\left(\sum_{i=1}^{n}\left(|\text{SC 计算结果} - \text{精确结果}|\right)\right)$$

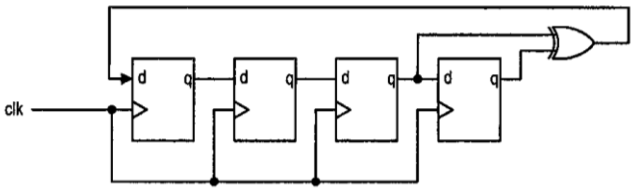
  其中 n 是实验次数。每次实验输入采用随机生成，要求 n 大于等于 1000。合理选择 LFSR 的结构，取得尽量小的 MAE（不得大于 0.8）。**注：用 MUX 实现 SC 加法将最终结果按照一定倍数放缩，在计算 SC 序列 MAE 的时候需要回乘该放缩因子以取得期望的结果**。
- Sobol 序列作为一种低密度序列，拥有比 LFSR 产生的随机数序列更好的数学特性，请阅读参考文献[3]和[4]中关于 Sobol 序列生成算法以及基于 Sobol 序列的序列生成单元的部分。将前面实现乘累加计算单元中的 LFSR 替换成 Sobol 发生器，重复上述实验，计算 MAE 并与 LFSR 下的 MAE 对比。
- 将 MUX 替换成多 SC 序列到二进制的转换单元，该单元的输入为 8 个乘积的序列，假设序列长度为 L，则该单元在 L 个时钟周期后输出最终的乘累加的二进制数形式，该加法单元记为 ADD。随机数序列仍然采用 Sobol，加法采用 ADD 实现，计算 MAE，并与之前的结果比较。
- 阅读参考文献[5]和[6]，实现一个串行（一个周期输出 1 个比特）Sobol 序列发生器，基于该 Sobol 序列发生器，实现一个 8 输入 SC 乘累加计算单元，其中加法用 ADD 实现。该乘累加计算单元记为 A1。
- 在 A1 的基础上，修改架构，使得一个周期可以处理 2 个序列比特，以提升计算的速度，该 2 并行的架构记为 A2。
- 用 Vivado 对 A1 和 A2 进行综合，比较两者的资源消耗和吞吐率。

提交要求补充

- 请将实验报告、软件代码、硬件代码打包成一个文件夹压缩后提交。

- 文件夹和压缩包均以以下格式命名：final01_组号。``final01"表示选择的是本实验，组号为两位数字，个位数需要在前面补零，例如组号为 3 则命名为``final01_03"。
- 文件夹分为 3 个部分：实验报告（一个 PDF 文件），软件代码（一个文件夹），硬件代码（一个文件夹）。
- 实验报告须为 pdf 文件，要包括本实验中要求的各点（软件设计思路、硬件设计思路、电路图、验证结果图、硬件综合结果、成员分工等等）。报告开头写出成员名称和学号。
- Vivado 综合选择 V7 系列 VC709（同 Lab1）。
- 涉及数据的地方请附上软件截图，例如 Vivado 综合结果、软件仿真得到的 MAE 数据等。
- 软件代码需要给出必要的注释（输入输出的解释，以及必要的逻辑单元的注释）。
- 硬件代码需要包括设计部分（电路实现）和验证部分（testbench），代码给出必要注释（输入输出的解释，以及必要的逻辑单元的注释）。
- **严禁抄袭**。

- LFSR 参考：



4bit LFSR 电路图

Register size	Feedback expression
2	$q_1 \oplus q_0$
3	$q_1 \oplus q_0$
4	$q_1 \oplus q_0$
5	$q_2 \oplus q_0$
6	$q_1 \oplus q_0$
7	$q_3 \oplus q_0$
8	$q_4 \oplus q_3 \oplus q_2 \oplus q_0$
16	$q_5 \oplus q_4 \oplus q_3 \oplus q_0$
32	$q_{22} \oplus q_2 \oplus q_1 \oplus q_0$
64	$q_4 \oplus q_3 \oplus q_1 \oplus q_0$
128	$q_{29} \oplus q_{17} \oplus q_2 \oplus q_0$

LFSR 反馈表达式

参考文献：

[1] Armin, Alaghi, John, et al. Survey of Stochastic Computing. ACM Transactions on Embedded Computing Systems (TECS), 2013.

[2] Alaghi A , Qian W , Hayes J P . The Promise and Challenge of Stochastic Computing. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2017.
[3] Alaghi, Armin, and John P. Hayes. "Fast and accurate computation using stochastic circuits." 2014 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2014.
[4] Bratley P, Fox B L. Algorithm 659: Implementing Sobol's quasirandom sequence generator[J]. ACM Transactions on Mathematical Software (TOMS), 1988, 14(1): 88-100.
[5] Dalal I L, Stefan D, Harwayne-Gidansky J. Low discrepancy sequences for Monte Carlo simulations on reconfigurable platforms[C]//2008 International Conference on Application-Specific Systems, Architectures and Processors. IEEE, 2008: 108-113.
[6] S. Asadi, M. H. Najafi and M. Imani, "A Low-Cost FSM-based Bit-Stream Generator for Low-Discrepancy Stochastic Computing," 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2021, pp. 908-913, doi: 10.23919/DATE.