Modeling and Verification of Stochastic Computing Circuits

Theory, Models and Properties with Demonstrations in PRISM

Prepared by the FLUENT research collaboration

Utah State University **Chris Winstead**
Zhen Zhang
Tom Prouty
University of Utah **Chris Myers**
University of Southern Florida **Hao Zheng**

Contents

List of Tables

List of Figures

Introduction

Stochastic computing is a form of approximate arithmetic where real numbers are encoded as (pseudo)random binary sequences, commonly called "bitstreams". A collection of bitstreams are filtered through a relatively simple logic network to transform their probability densities (the relative frequency of 1's in the stream). These probability transformations yield sophisticated computations. Results are collected by sampling the bitstream density at the network's outputs. Research interest has particularly focused on applications in neuromorphic computing and error correction decoding [@gross2019stochastic].

The models in this repository use the PRISM language to represent several classes of stochastic computing circuits. Their purpose is to apply stochastic modeling and verification tools to answer key questions about stochastic computing circuits, from the simplest models up to complex systems.

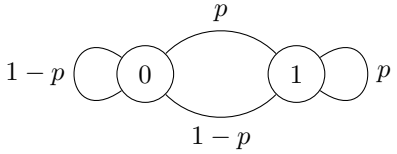
For each model, a *property list* is specified using the PRISM syntax. The most basic properties include the circuit's steady state behavior, and its transient convergence toward the steady state. In both properties, we shall be interested in the circuit's accuracy and precision (relative to the intended function), and its ergodicity (i.e. its dependency on the initial state). In transient properties, we shall address the convergence speed, along with more advanced concerns such as deadlock and livelock conditions.

Bitstreams and Stochastic Numbers

A *bitstream* $b(t) \in \{0, 1\}$ is a sequence or signal composed of binary values. The *probability density* $p_b(t)$ is the probability that $b(t) = 1$. A *stochastic number* $S(t) \in \mathbb{R}$ is a real number associated to the density of one (or more) bit streams by a one-to-one mapping $f_S(p_b(t))$. The mapping f_S is called the *numerical representation*, and is analogous to numerical formats in conventional digital logic (unsigned, signed, fixed-point, floating-point, etc).

A *discrete stochastic bit stream* (DSBS) is a binary discrete-time Markov process (DTMC) encoding a density $p(t)$, $0 < p(t) < 1$, with $t \in \mathbb{N}$. For example, the density $p = 2/5$ is encoded by the sequence $0, 1, 0, 0, 1, \ldots$, and also by the sequence $1, 1, 0, 0, 0, \ldots$, and so on.

A *continuous stochastic bit stream* (CSBS) is a binary continuous-time Markov process encoding a density $p(t)$ with $t \in \mathbb{R}$.



DSBS Markov process.

Numerical Representations

At the physical level, stochastic logic circuits operate on bitstreams, processing densities between zero and one. Stochastic numbers introduce an abstract level of interpretation. The same logic gate can perform a variety of distinct functions depending on which numerical representation is chosen.

Unipolar Stochastic Numbers

The most direct numerical representation is the mapping $S = p$.

Bipolar Stochastic Numbers

$S = 2p - 1$

Likelihood Ratios

$S = \frac{p}{1-p}$

Combinational Stochastic Logic

Basic Logic Gates

The most basic stochastic computing circuits are combinational gates. For example, given two ideal, uncorrelated DSBS signals A and B, the operation $Q = A\&B$ delivers $p_Q = p_A p_B$, so it can be used as a multiplier. Another common example is the interleaving MUX: $Q = SA + \overline{S}B$, so that $p_Q = p_S p_A + (1 - p_S)p_B$. A MUX-AND network can be used to implement a system of polynomial computations.

When implemented as a memoryless tree (a simple combinational network with no feedback), stochastic circuits converge in mean to the desired result. Steady-state probabilities are insensitive to gate delays and no synchronization is required. Unfortunately most stochastic computing applications are not memoryless trees, and two related problems emerge: *correlation* leading to inaccuracy, and *live-locking* where a sub-circuit enters a fixed deterministic state.

MUX Logic

Sequential Stochastic Logic

Single Flip-Flop Circuits

Correlation and Regeneration

Asynchronous Stochastic Circuits

Continuous-Time Bit-Streams

Combinational Logic

Single Flip-Flop Circuits