Paper Review

Space-Time Algebra: A Model for Neocortical Computation

Reference: Smith, James. "Space-time algebra: A model for neocortical computation." ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA). IEEE, 2018.

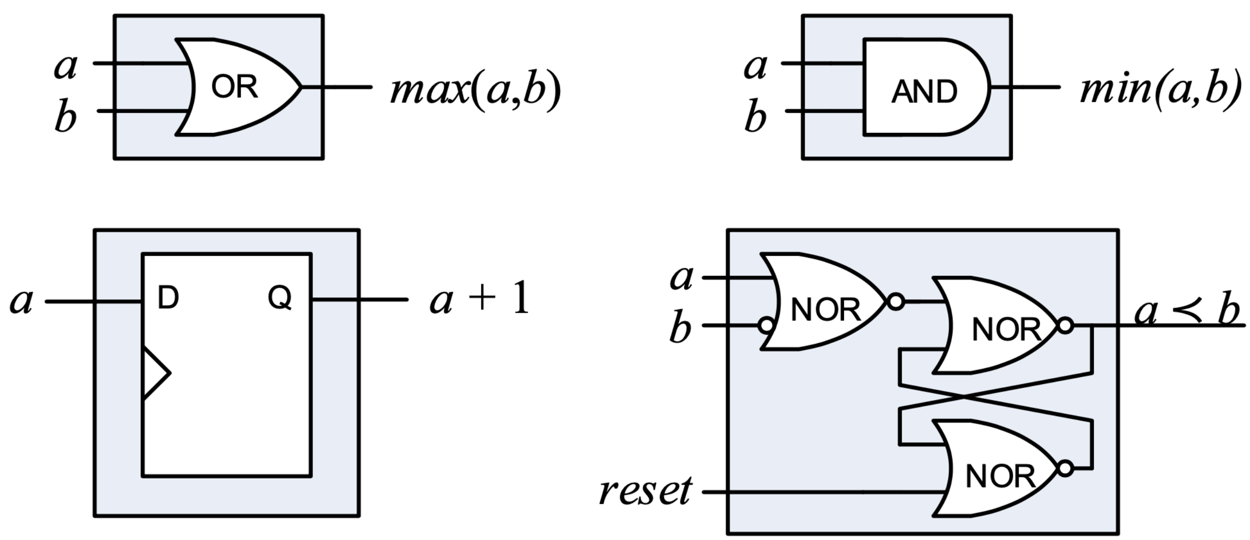
The paper reviewed here was published in *ISCA 2018*, which is one of the premier computer architecture conferences. The author of this paper is **James E. Smith**, who is a well-known computer architect with a number of inventions under his belt (e.g., the 2-bit Smith predictor), and a recipient of the prestigious *Eckert-Mauchly Award*.

Reverse architecting the brain (or neocortex, which is the brain's house of intelligence) to build intelligent efficient computers has been considered a *holy grail* in the field of Computer Architecture. Biological neural networks facilitate computing in the brain via 'spikes', which are all-or-nothing events characterized by their presence at a given point in time. The paper proposes a rigorous foundational mathematical abstraction to model a special type of such spiking neural networks directly inspired from the neocortical computational paradigm, that encode and process information in the form of spike timings. This mathematical abstraction, termed 'Space Time Algebra', can be considered as a counterpart to conventional Boolean algebra, that can model 'Space Time' functions instead of Boolean functions. A space-time function is any function with inputs and outputs as non-negative integers, that satisfies the three properties of *computability*, *causality* and *invariance*. Computability ensures the function is physically implementable with bounded inputs and outputs. Causality and invariance adhere to the natural flow of time and imply that 1) outputs can only be affected by inputs preceding them in time, and 2) if all inputs shift in time by a fixed amount, the outputs also shift in time by the same amount. The underlying hypothesis is that biologically inspired spiking neural networks operating on spike timings, termed as 'Temoporal Neural Networks (TNNs) are basically space-time functions that operate on discrete timing spike inputs.

min(a,b)	Emits an output at the same time as the first arriving input
increment(a)	Emits an output one time unit after the input
less-than(a,b)	Emits an output at the same time as 'a' if 'a' arrives before 'b'; no output otherwise

Space-time algebra primitives

A strong result of space-time algebra is that all space-time functions can be implemented using the three primitives '*min*', '*increment*' and '*less-than*', i.e., these three primitives are functionally complete for the set of space-time functions (shown in the above table). Previous work on 'Race Logic' has shown how conventional CMOS gates can be repurposed to implement graph algorithms such as shortest/longest path and boosted decision trees very efficiently using much simpler gates, if the inputs are encoded as timings of logic signal transitions (1->0 or 0->1). For example, 'min(a,b)' can be implemented using a single AND gate, if 'a' is encoded as a 1->0 signal transition at time instant 'a' and 'b' is encoded as signal transitioning from 1->0 at time instant 'b'. This Race Logic implementation methodology has been extended by Smith in this paper to a 'Generalized Race Logic (GRL)' consisting of the the three primitives implemented using logic gates as shown below. Further, he states that the GRL primitives provide a complete CMOS implementation of space-time algebra.



<u>GRL primitives: 'min', increment and 'less-than', along with 'max' implemented using standard CMOS logic gates.</u>

Another interesting contribution made by the author is the implementation of a generic SRM0 excitatory spiking neuron as well as winner-take-all inhibition using space-time primitives. The neuron implementation utilizes bitonic sorters, where each min/max comparator element in the sorter is implemented using just two gates (1 AND and 1 OR gate). In conclusion, not only can space-time algebra be used to implement SRM0 neurons and corresponding neural networks, but can also serve as the algebraic foundation underlying Generalized Race Logic implementation methodology.