Temporal Computing

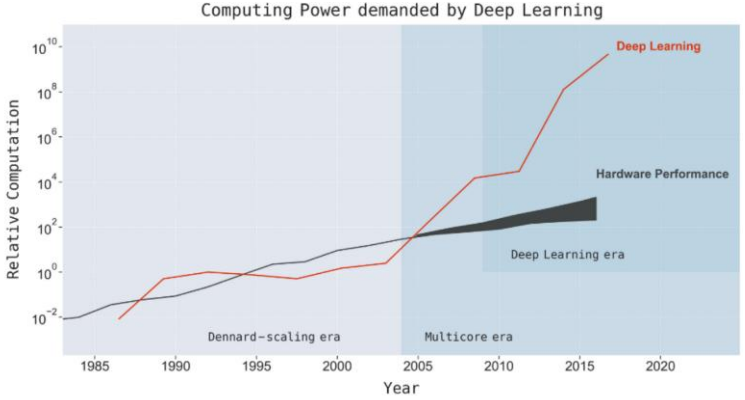
Jonny Edwards

WHY?

Benchmark	Error Rate	Polynomial			Exponential		
		Computation Required (flops)	Environmental Cost (CO_2)	Economic Cost ($)	Computation Required (flops)	Environmental Cost (CO_2)	Economic Cost ($)
ImageNet	Today: 9.00%	10^{23}	10^{5}	10^{6}	10^{24}	10^{6}	10^{7}
	Target 1: 5%	10^{26}	10^{8}	10^{9}	10^{30}	10^{13}	10^{14}
	Target 2: 1%	10^{33}	10^{16}	10^{16}	10^{92}	10^{74}	10^{75}

Progress in accuracy is linked to computational burden which is linked to carbon footprint

Carbon emissions (lbs), and economic costs ($USD)



Hardware performance cannot keep pace

THE COMPUTATIONAL LIMITS OF DEEP LEARNING *Neil C. Thompson, Kristjan Greenewald* MIT (JULY 2022)

By way of comparison, OpenAI's GPT-3 and Meta's OPT were estimated to emit more than 500 and 75 metric tons of carbon dioxide, respectively, during training. GPT-3's vast emissions can be partly explained by the fact that it was trained on older, less efficient hardware. But it is hard to say what the figures are for certain; there is no standardized way to measure carbon dioxide emissions, and these figures are based on external estimates or, in Meta's case, limited data the company released.

MIT Technology Review

“We need to rethink the entire stack — from software to hardware,” says Aude Oliva, MIT director of the MIT-IBM Watson AI Lab and co-director of the MIT Quest for Intelligence. “Deep learning has made the recent AI revolution possible, but its growing cost in energy and carbon emissions is untenable.”

What is Temporal Computing?

- Computing with delays in time so the number 6 becomes
  - <click> *6 seconds* <click>
- Addition 9 =
  - <click> *6 seconds* <click> *3 seconds* <click>
- All vital operations performed (2014)
- THE CONJECTURE
  - Addition 9 =
    - <click> *6* ***femtoseconds*** <click> *3* ***femtoseconds*** <click>
    - **petahertz processing**

7 Reason for Thinking Temporally

1. Time is free.
2. No Need for gates ... accumulate and scale
3. The representation is more efficient.
4. Easy to Parallelise.
5. Clocks are Fast.
    1. Waves are even faster.
6. Can be implemented in Analog and Digital
7. And on a variety of media from CMOS to Radio Waves
8. Its what the brain does.

Simple clocks are still fast ...

Oscillator name	State variable	Frequency (Hz)	Energy/cycle (J)	Possible coupling mechanism
Ring oscillator	Electric	Up to 20 GHz	10^{-15}	Electrical
Relaxation oscillator based on phase-transitions	Electrical	Up to 10 GHz	10^{-17}	Electrical
LC oscillator	Electrical	Up to 100 GHz		Electrical
Superconducting oscillator	Electric and magnetic	Several 10 GHz	10^{-17}	Electrical, inductive, and capacitive
Mechanical (NEMS) oscillator/ RBO	Mechanical	Up to 20 GHz	10^{-14}	Electrical or mechanical
Spin torque oscillator (STO)	Magnetic	Upward 50 GHz	10^{-15}	Electric, magnetic or spin wave
Chemical	Electrochemistry	10^{2}	No data	No data
Magnetic anisotropy controlled parametric	Magnetic	Up to 20 GHz	No data	Electrical
Spin-Hall oscillator	Magnetic	Up to 20 GHz	10^{-16}	Electric, magnetic or spin wave
SET device	Electric	10 GHz	10^{-18}	Electrical

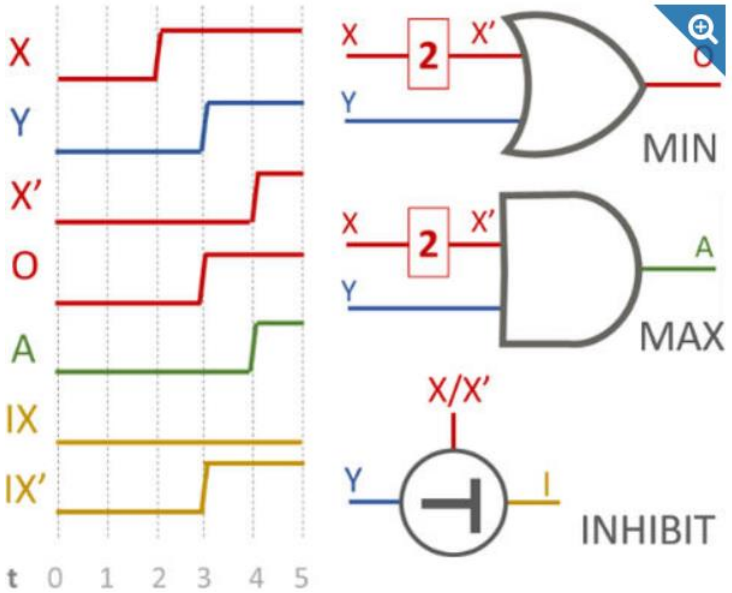
Temporal Computing: Groups

- NIST Uni Maryland (Advait Madhaven)
- Tim Sherwood's group Uni Cal. Santa Barbara (Advait Madhaven)
- UGEMM group (Wisconsin)
- Prof James Smith's (Wisconsin) work
- York Group (Me and Simon!)
- Temporal Computing (Me and Prof Alex Yakovlev and team)

NIST

Min/Max/inhibit

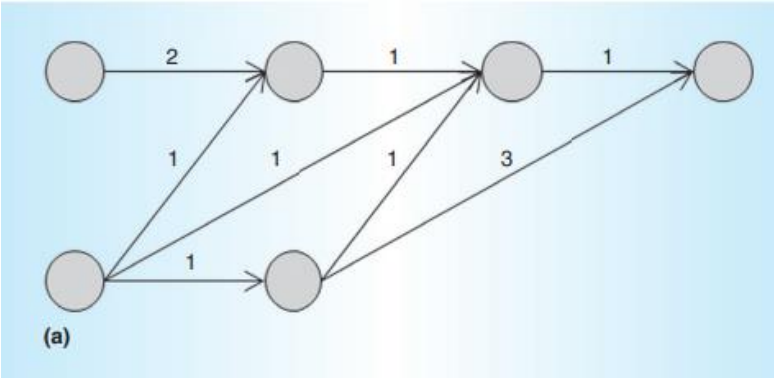
Synch-par

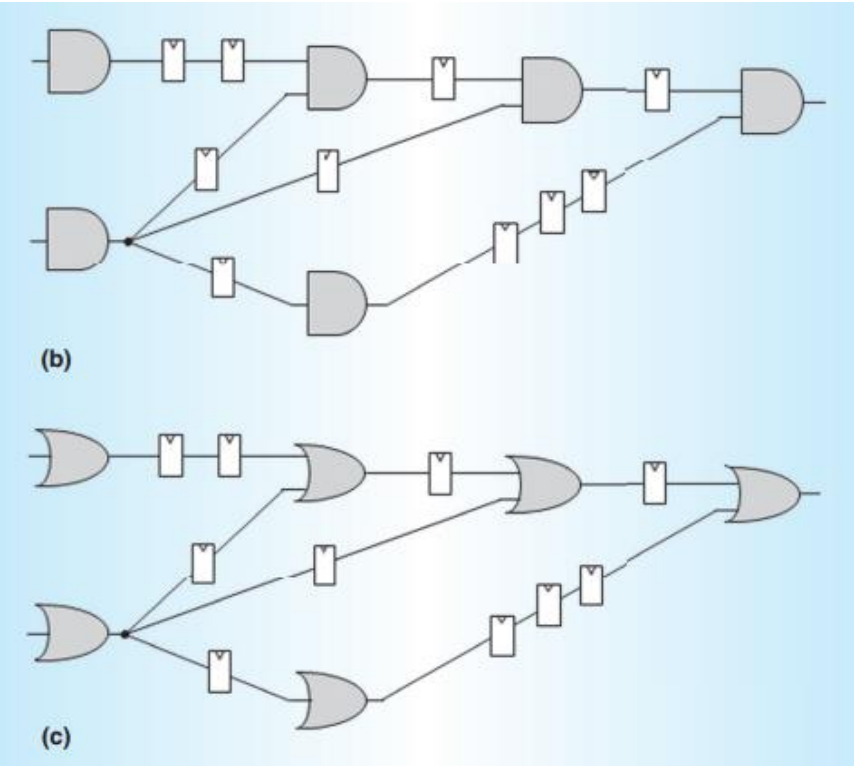


https://www.nist.gov/programs-projects/temporal-computing

NIST/ Uni Cal Santa Barbara

Min/Max
Synch-par

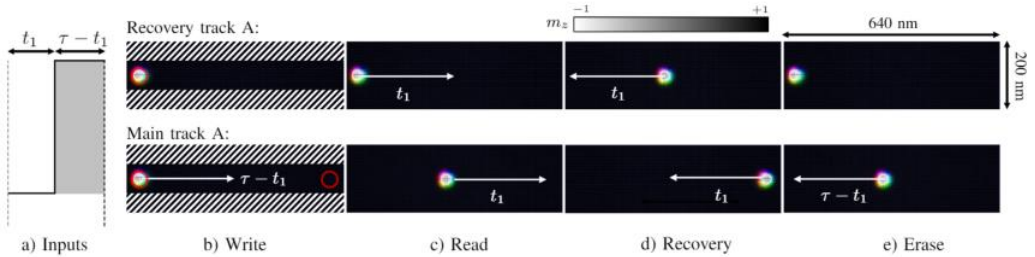


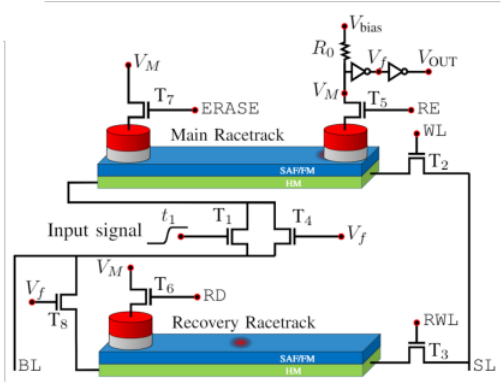


A. Madhavan, T. Sherwood and D. Strukov, "Race Logic: Abusing Hardware Race Conditions to Perform Useful Computation," in *IEEE Micro*, vol. 35, no. 3, pp. 48-57, May-June 2015, doi: 10.1109/MM.2015.43

Racetracks

Red/Write/Erase



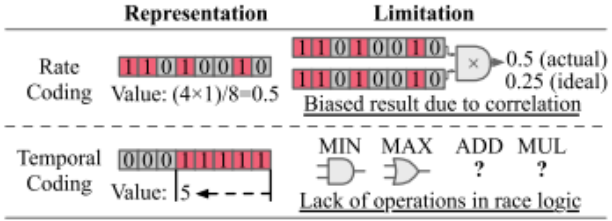


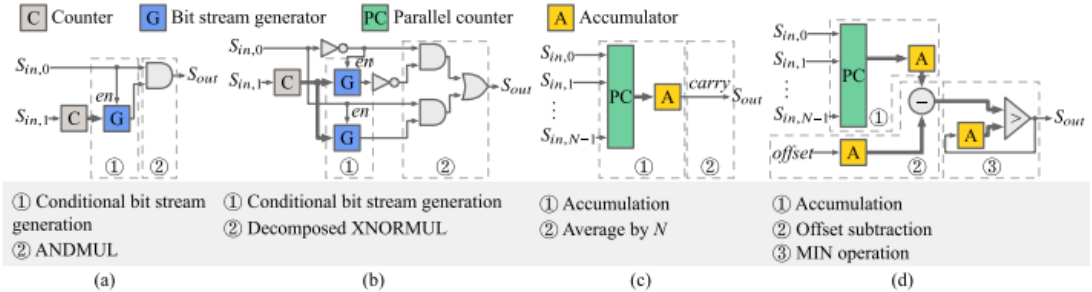
H. Vakili *et al*., "Temporal Memory With Magnetic Racetracks," in *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, vol. 6, no. 2, pp. 107-115, Dec. 2020, doi: 10.1109/JXCDC.2020.3022381.

UGEMM

Min/Max/Add/Mul

Synch-par





D. Wu, J. Li, R. Yin, H. Hsiao, Y. Kim and J. S. Miguel, "uGEMM: Unary Computing for GEMM Applications," in IEEE Micro, vol. 41, no. 3, pp. 50-56, 1 May-June 2021, doi: 10.1109/MM.2021.3065369.

James Smith

Min/Max/Gtr/less/equal

Synch-par

Table 1. All 2-ary *s-t* functions.

$a < b$	$a = b$	$b < a$	function	name	symbol
a	a or b	b	if $a < b$ then a ; else b	*min*	$\wedge$
a	a or b	∞	if $a \leq b$ then a ; else ∞	*less or equal*	$\preccurlyeq$
a	∞	a	if $a \neq b$ then a ; else ∞	*not equal*	$\neq$
a	∞	b	if $a < b$ then a else if $b < a$ then b ; else ∞	*exclusive min*	x$\wedge$
a	∞	∞	if $a < b$ then a ; else ∞	*less than*	$\prec$
b	a or b	a	if $a \geq b$ then a ; else b	*max*	$\vee$
b	∞	a	if $a > b$ then a else if $b > a$ then b ; else ∞	*exclusive max*	x$\vee$
∞	a or b	a	if $a \geq b$ then a ; else ∞	*greater or equal*	$\succcurlyeq$
∞	a or b	∞	if $a = b$ then a ; else ∞	*equal*	$\equiv$
∞	∞	a	if $a > b$ then a ; else ∞	*greater than*	$\succ$

2.2 Symbols and Notation

Symbols for the primitive functions are shown in Figure 2.

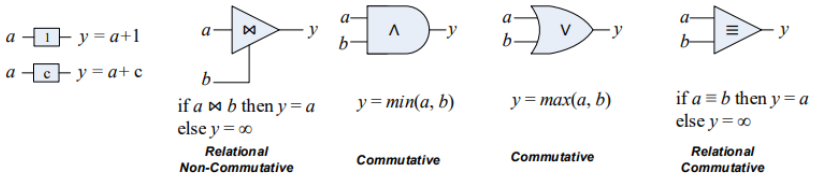


Figure 2. Symbols representing the various primitive operators that may be used in network schematics. The symbol ⋈ represents any of the non-commutative relational operations.

https://arxiv.org/abs/2001.04242

And Finally …

Min/Max/Add/Mul/Sub/MAC

Sequential Multiplexed

A Variable Bitwidth Asynchronous Dot Product Unit

Jonny Edwards[1], Adrian Wheeldon[2], Rishad Shafik[2], and Alex Yakovlev[2]

[1]Temporal Computing Ltd, Newcastle upon Tyne, UK
[2]μSystems Group, Newcastle University, Newcastle Upon Tyne, UK

***Abstract*— We demonstrate a many operand asynchronous dot product with adjustable bit width for one of the multiplication tuple operands. The generalised algorithm called MADD relies on the commutative aspects of both the addition order and multiplication tuple operands. The approach is deeply wedded to asynchrony due to optimisation of the variable bitwidth memory architecture. Although specialised, this algorithm has a significant practical application in deep learning models particularly in managing the variable operand size due to the data variability, layering and optimisation process.**

I. Introduction and Prior Work

Dot products are the workhorse of many signal and data processing methods. Latterly they have gained prominence as the main distributed processing element within deep learning

Algorithm 1 Dot-product calculation with MADD

```
1:  procedure DotProduct(memory)
2:      acc, height ← 0
3:      i ← MAXVAL
4:      while i > 0 do          ▷ Iterate over the whole memory
5:          height ← height + memory[i]
6:          acc ← acc + height + memory[i]
7:          i ← i − 1
8:      end while
9:      control ← 1
10:     return acc              ▷ Contains the dot product
11: end procedure
```

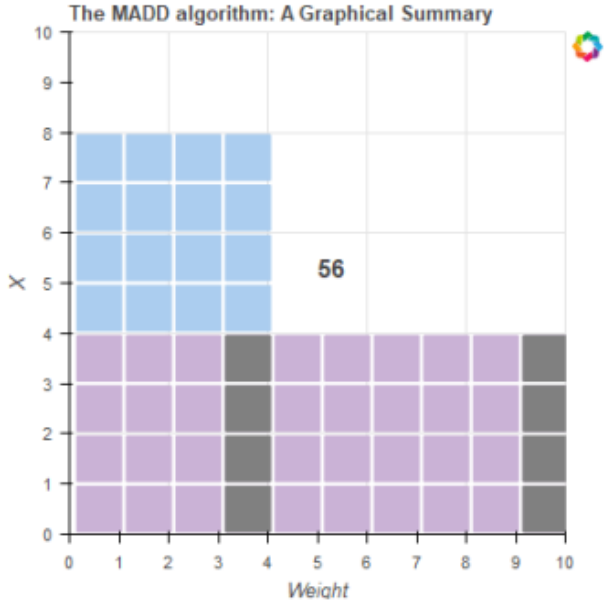


Fig. 4. A geometric view of the MADD algorithm. Here we perform the operation $(4\times10)+(4\times4)$ right to left by accumulating the "height" at the index point

•A Variable Bitwidth Asynchronous Dot Product Unit Fresh Idea
Jonny Edwards, Adrian Wheeldon, Rishad Shafik, and Alex Yakovle 2019
ASYNCH19

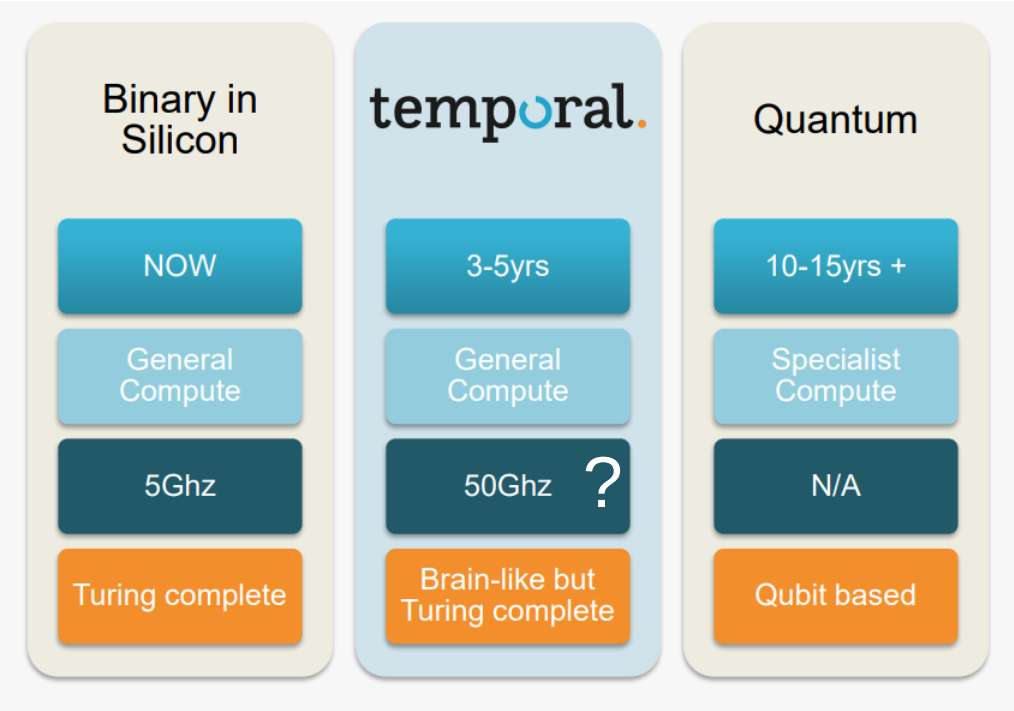
- Even the definition is contentious
- Research is “patchy” at best
- The benefits are too “hand-wavey”
  - There’s no convincing “Physics”
- Race logic is simple
  - The graph solutions are space-time tradeoffs
  - Min/max are only a subset of useful operations
    - Tropical algebras may help
  - The memory work is very isolated from compute
- No mention of Turing Completeness
- My work is only partially in hardware
- There is no fixed platform
- No taxonomy
- There’s no killer problem

- We have a starting point here.
  - Taxonomies and definitions
- Ramps and Capacitors are useful.
  - And a start has been made.
- Infinite memory is interesting.
- General speedup may be the killer problem.
- An implementation platform may come along with world-changing characteristics.
- The 1-bit idea …

THE ELEPHANT(S) in the room

1. MODERN HARDWARE is NOT going to scale for next-gen AI

2. QUANTUM COMPUTING will not be useful in my lifetime.


Binary in Silicon
NOW
General Compute
5Ghz
Turing complete
temporal.
3-5yrs
General Compute
50Ghz ?
Brain-like but Turing complete
Quantum
10-15yrs +
Specialist Compute
N/A
Qubit based