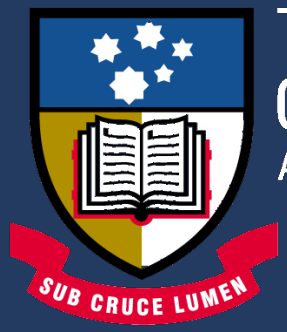
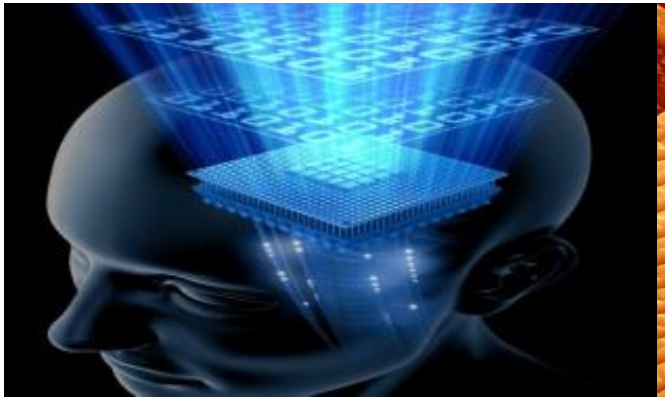


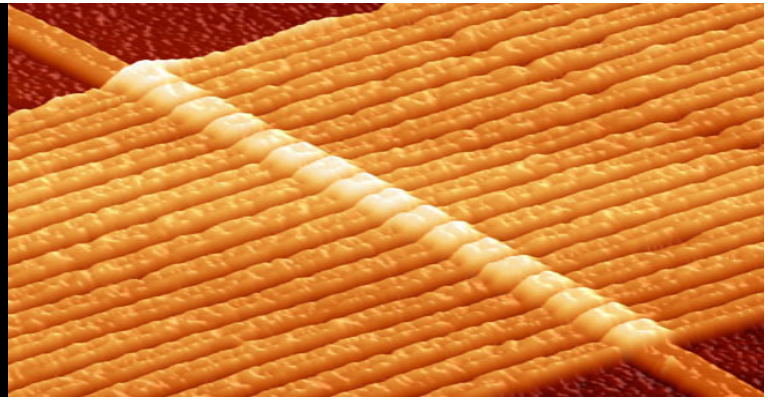
THE UNIVERSITY OF ADELAIDE
AUSTRALIA

Realising Soar Long-Term Memory in Hardware

1. A Hardware Semantic Cache
   Muhammad Usman Khan

2. Long-term Memory using Memristors
   Peng Wang





A Hardware Semantic Cache

Muhammad Usman Khan

We are building hardware for Soar. Why start with semantic memory?

- Soar's semantic memory was designed for efficiency
  - simple, consistent, well-defined data structure
  - small set of efficient retrieval modes
  - we don't need to build an SQL server
- Requirements suit hardware
  - fine grained parallelism
  - lots of transistors
  - limited switching activity ⇒ low power
- Opportunity to support new features
  - Base-level activation
  - Spreading activation
- A stepping-stone to episodic memory

Soar semantic memory refresher

- Semantic memory objects consist of one or more working memory elements (WMEs):

  {Long Term Identifier (LTI), Attribute, Value}

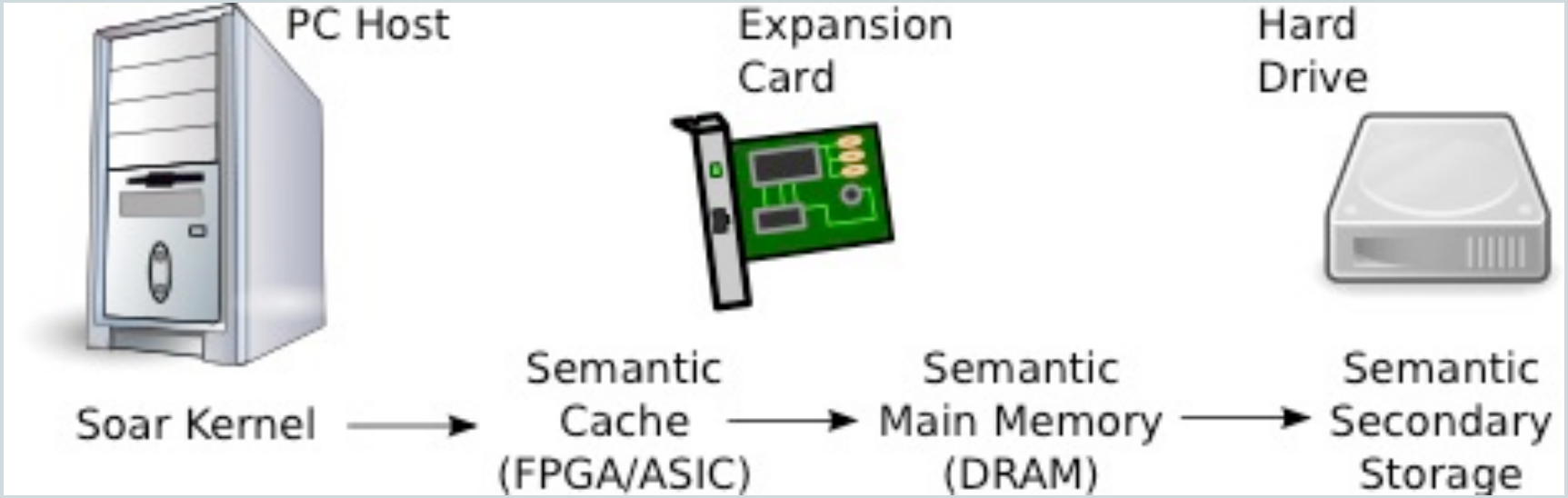
- A cue-based retrieval specifies WMEs to match
  - Exact match of attribute and value
    Exact match of an LTI value
    Exact match attribute, but with any value
  - Where multiple objects match, the one with highest activation is retrieved
  - A cue can be modified with the LTIs of objects that must not match – so you can iterate over objects that match a cue
- Stores and retrievals send or receive an object and its immediate augmentations (WMEs)

Requirements

- Scalable to very large knowledge bases
  - for general agents
  - persist for long periods
- Low latency
  - real time reactivity
  - deeper search, richer knowledge representation
- New features
  - base-level activation
  - spreading activation
  - parallel with decision cycle

Our Answer: Hierarchy and Virtual Memory

- We know how to make large, scalable, high-performance memory systems: *memory hierarchy* and *virtual memory*.



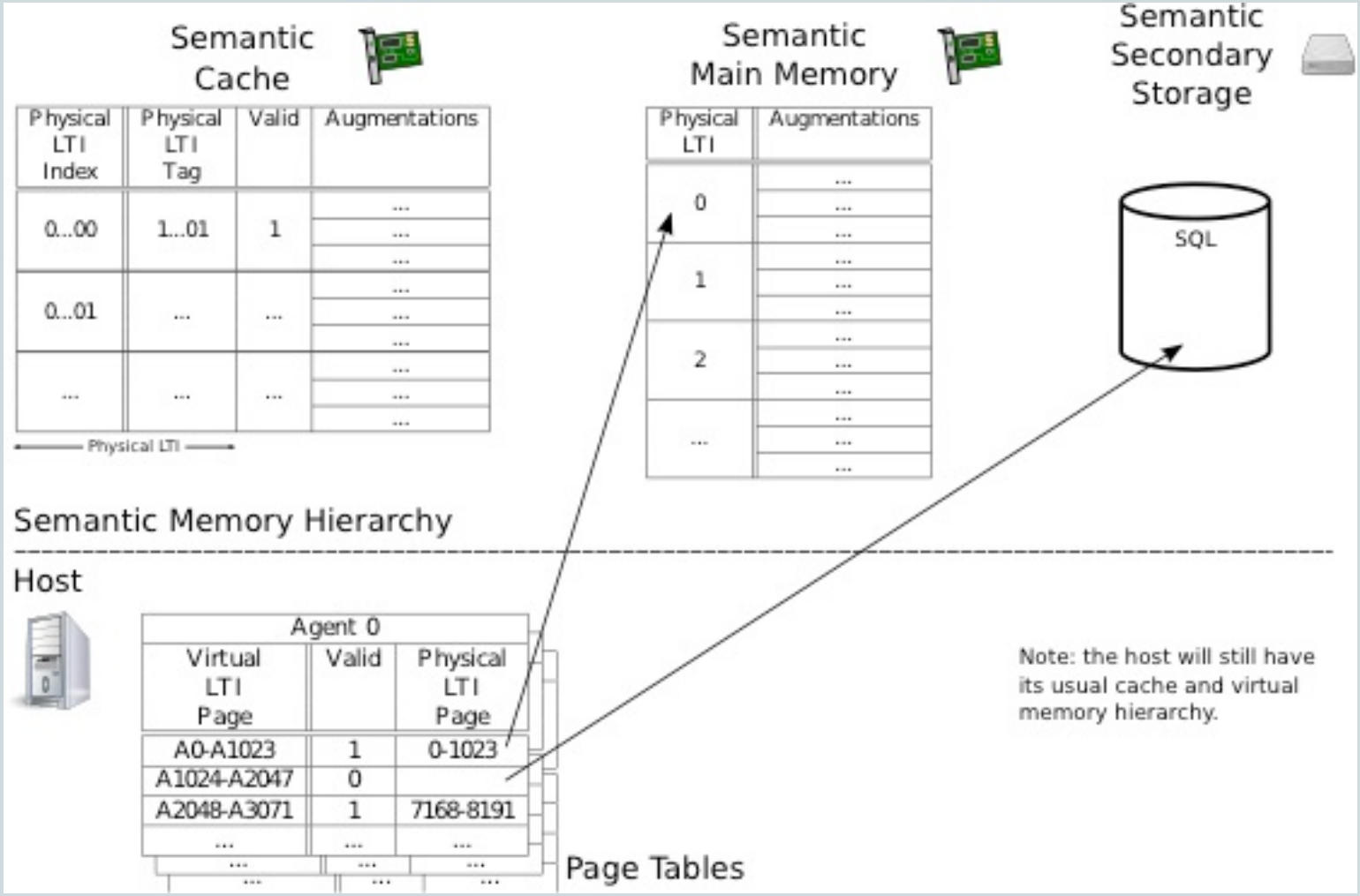
Why not just use the memory hierarchy of a conventional computer?

Differences between semantic and conventional computer memory are opportunities for optimisation.

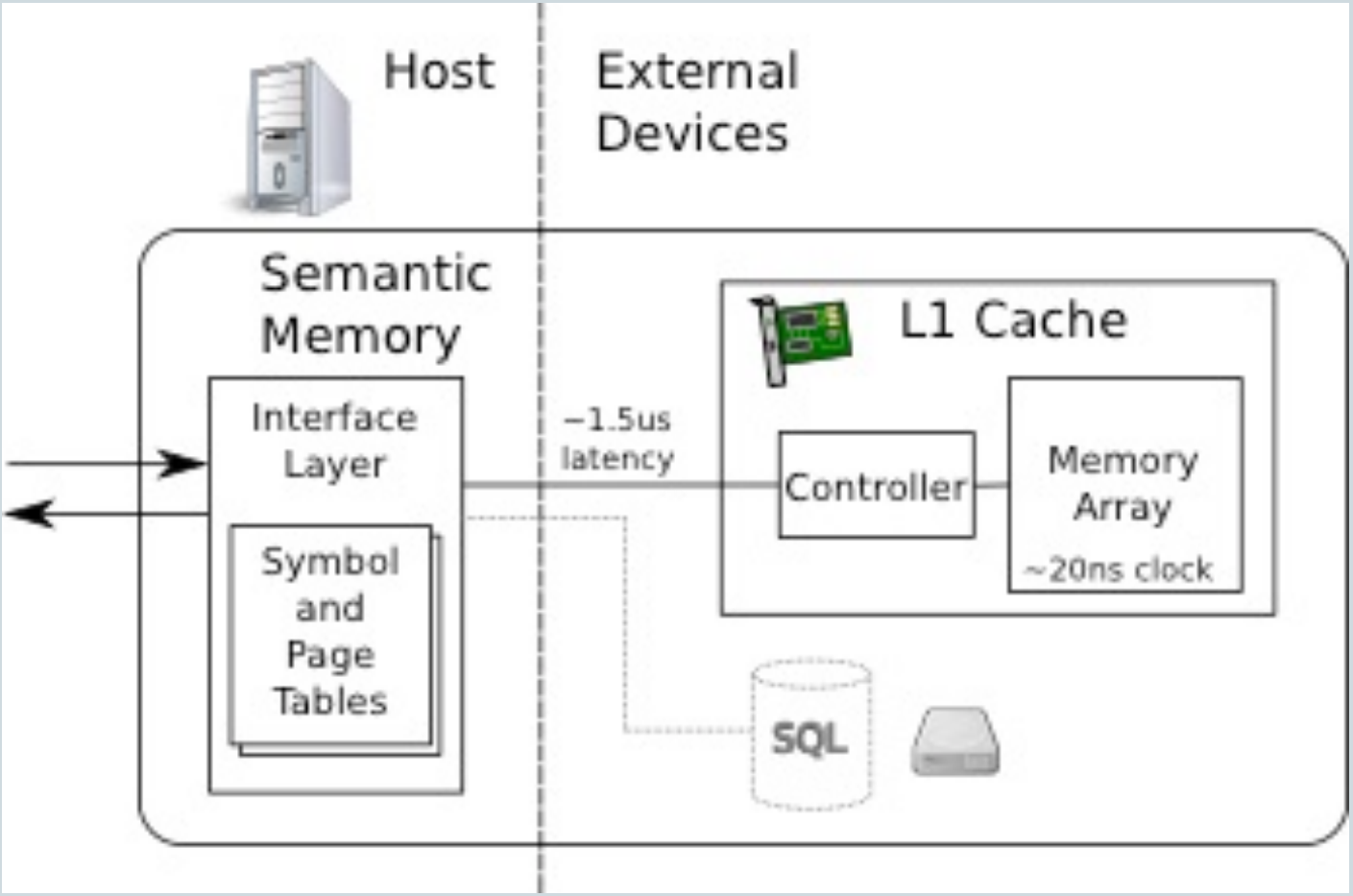
Soar semantic memory uses:

- Cue-based retrieval
- Multiple cues
- Multiple matches to a cue
- Activation
  - explicit requirement to return the object with the highest activation
  - a good match with hierarchical memory
    temporal locality: expected due to context
    spatial locality: (adjacent LTIs) may occur
- Spreading activation
  - can exploit the spatial locality explicit in the data structure

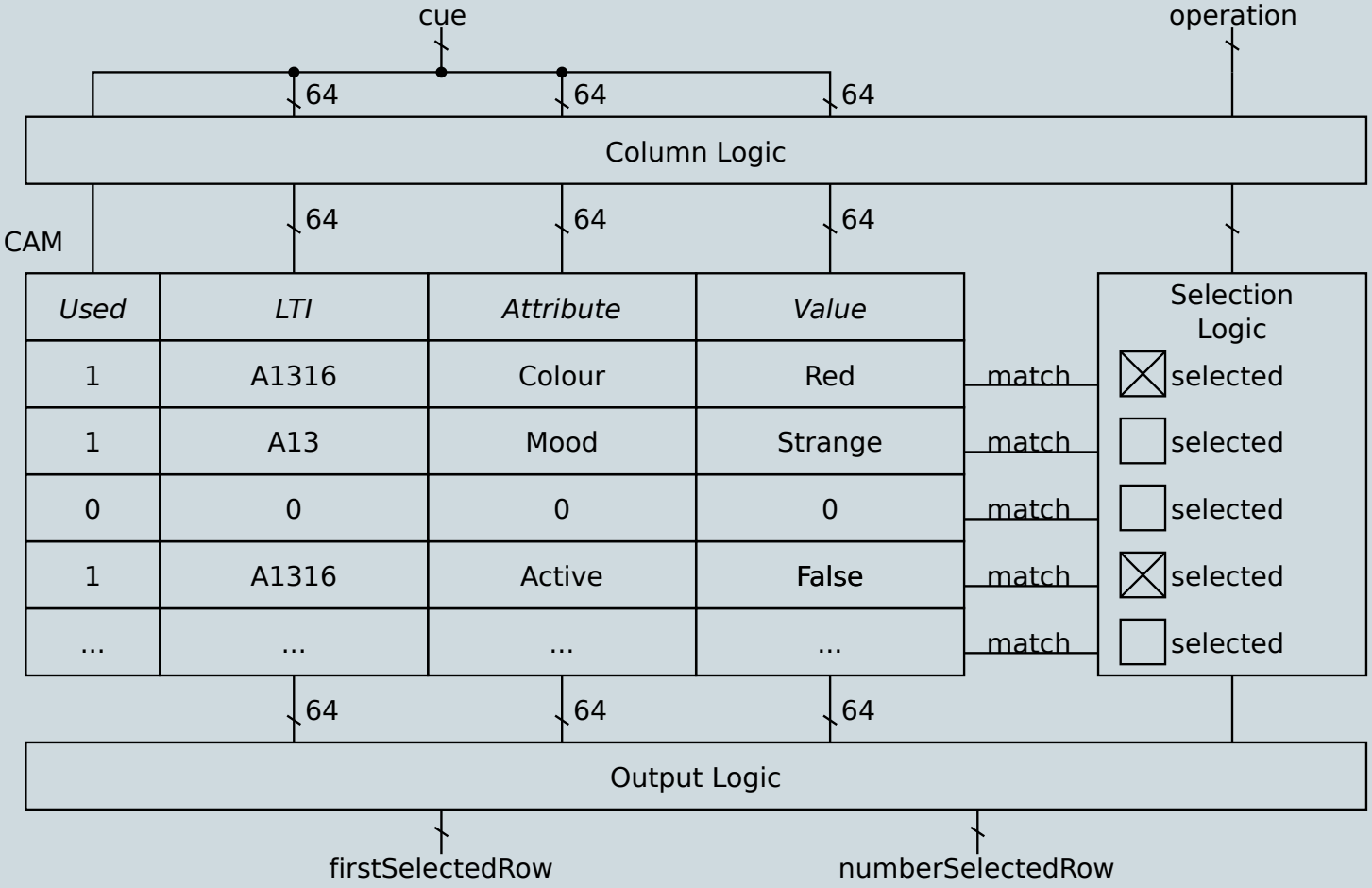
Semantic memory system organisation



First step: semantic cache



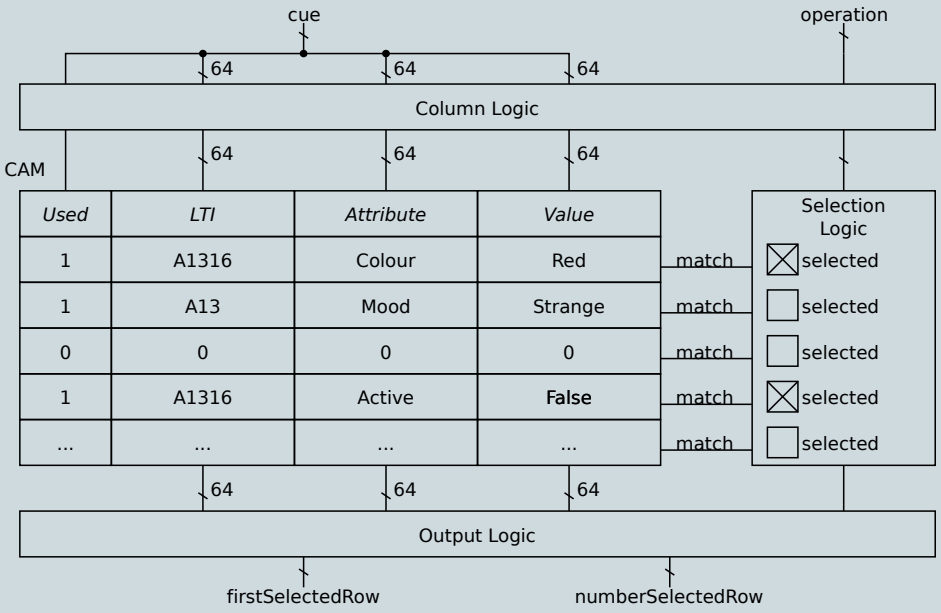
Semantic cache microarchitecture RCAM



Benefits of this structure

- Deals neatly with variable-sized objects
  - Any WME can occupy any row in memory
  - No problems with fragmentation
- Parallel cue-based search
  - Each extra cue requires only one extra memory clock cycle
- Leverages conventional CAM
- The RCAM is useful for other Soar hardware
  - Many different operations can be performed using the selection scheme
  - A selection stack can be added for nested iterations
  - Value comparisons (e.g. >=0, <0, =0) are possible for working memory operations

Cue-based retrieval example



1. Select all used rows that match the first cue
2. For each additional cue: deselect all rows that do not match the cue
3. Read the LTI of the first selected row
4. Deselect all rows and then select all rows that match the LTI
5. While there are rows selected:
   read the first selected row and deselect this row

Implementation results

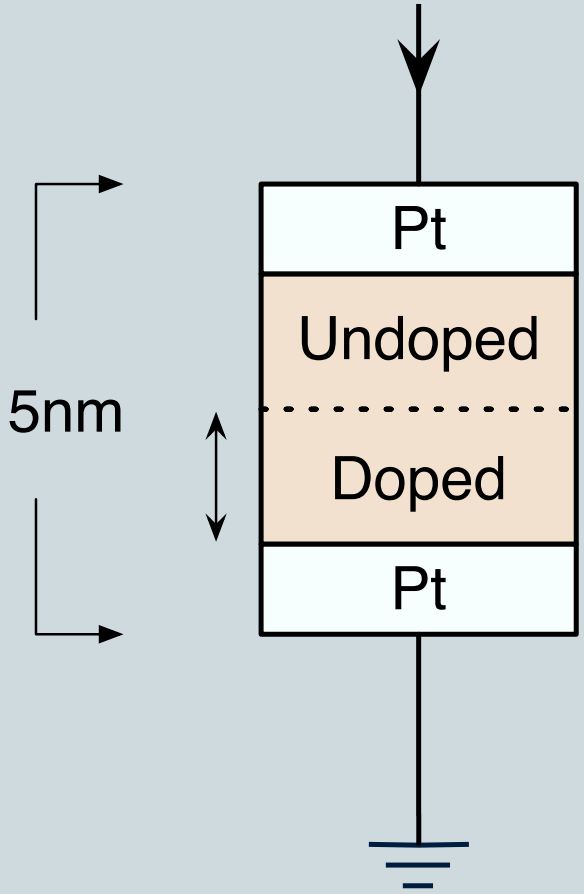
- Xilinx Virtex 4 XC4SX55
  - 32-bit PCI interface to PC
  - 20ns memory clock cycle time
  - 20us latency for a cue-based retrieval (mainly bus latency!)
  - c.f. 189us for 400MB SQL semantic memory[1]
  - but a small FPGA – 32 WMEs only
- Improvements
  - XC7V2000T FPGA has capacity for 4096 WMEs Memory latency would still be hidden behind bus latency
  - Expect much greater density from an ASIC (e.g. CAMs with capacity for 100k WMEs are available[2])
  - 64-bit PCIe would improve latency

1. Derbinsky, Laird and Smith, "Towards Efficiently Supporting Large Symbolic Declarative Memories," *10th ICCM*, 2010.
2. e.g. Renesas 20Mbit Standard TCAM (R8A20410BG)

Long-term Memory using Memristors

Peng Wang

What is a Memristor?



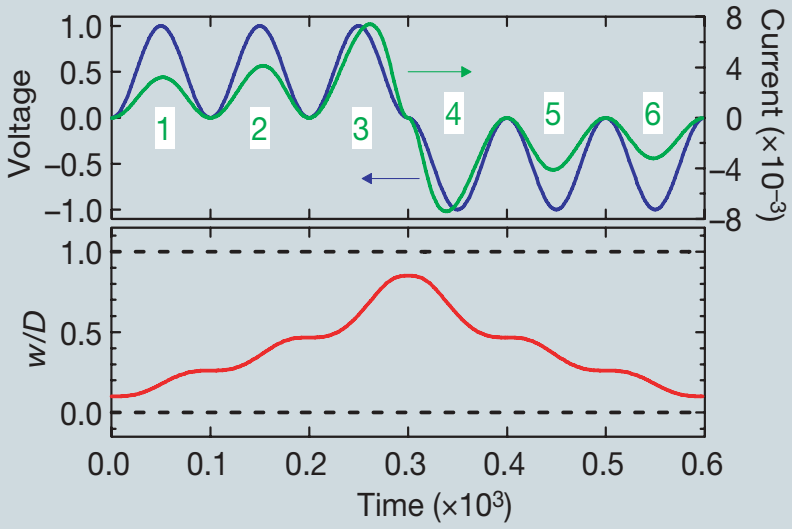
- Typical structure: metal-oxide-metal
- Demonstrated cell area: $4F^2$ vs $12F^2$ (DRAM main memory)
- When a current is applied, it changes its resistance as the doped region shrinks or expands
- The change to resistance persists for over 10 years – useful as a non-volatile memory element

Source: ITRS, "International Tecnology Roadmap for Semiconductors," 2011.

Memristor

- Digital
  - High resistance & low resistance to represent binary values
- Analog
  - Theoretically, an ideal memristor's resistance has a linear relationship with the charge that has passed through it
  - In analog applications, we can control the resistance by controlling the current and hence the charge

Figure 1: Memristor's analog properties



Source: Strukov et al, The Missing Memristor Found, Nature Letter, 2008

Why use Memristors for long-term memory?

- **High density**
  - We can pack a great many memristors on a chip
  - Unlike transistors, you can stack many layers of memristors on a chip
  - The layers of memristors can be built on top of a bottom layer of CMOS transistors
- **Low power & high speed**

	Feature size(F)	Read time	Write time	Retention time	Write cycle	Reading voltage
memristor	5nm	< 50ns	0.3ns	> 10 years	10^{12}	$0.15V$
DRAM	36nm	< 10ns	< 10ns	64ms	$> 10^{16}$	$1.8V$
FLASH	22nm	0.1ms	0.1ms	10 years	10^{4}	$1.8V$

Source: ITRS, "International Tecnology Roadmap for Semiconductors," 2011.

What does our proposed long-term memory look like?

LTI	Attribute	Value	Activation
element 0			A_0
element 1			A_1
element 2			A_2
...			...
element n-1			A_{n-1}

CAM

- Content addressable memory (CAM) for storage & parallel search
- Activation circuits for each memory element

Activation

- Base-level activation provides a way to reflect the relevance of information in memory

$$B = \ln \sum_{j=1}^{n} t_j^{-d}$$

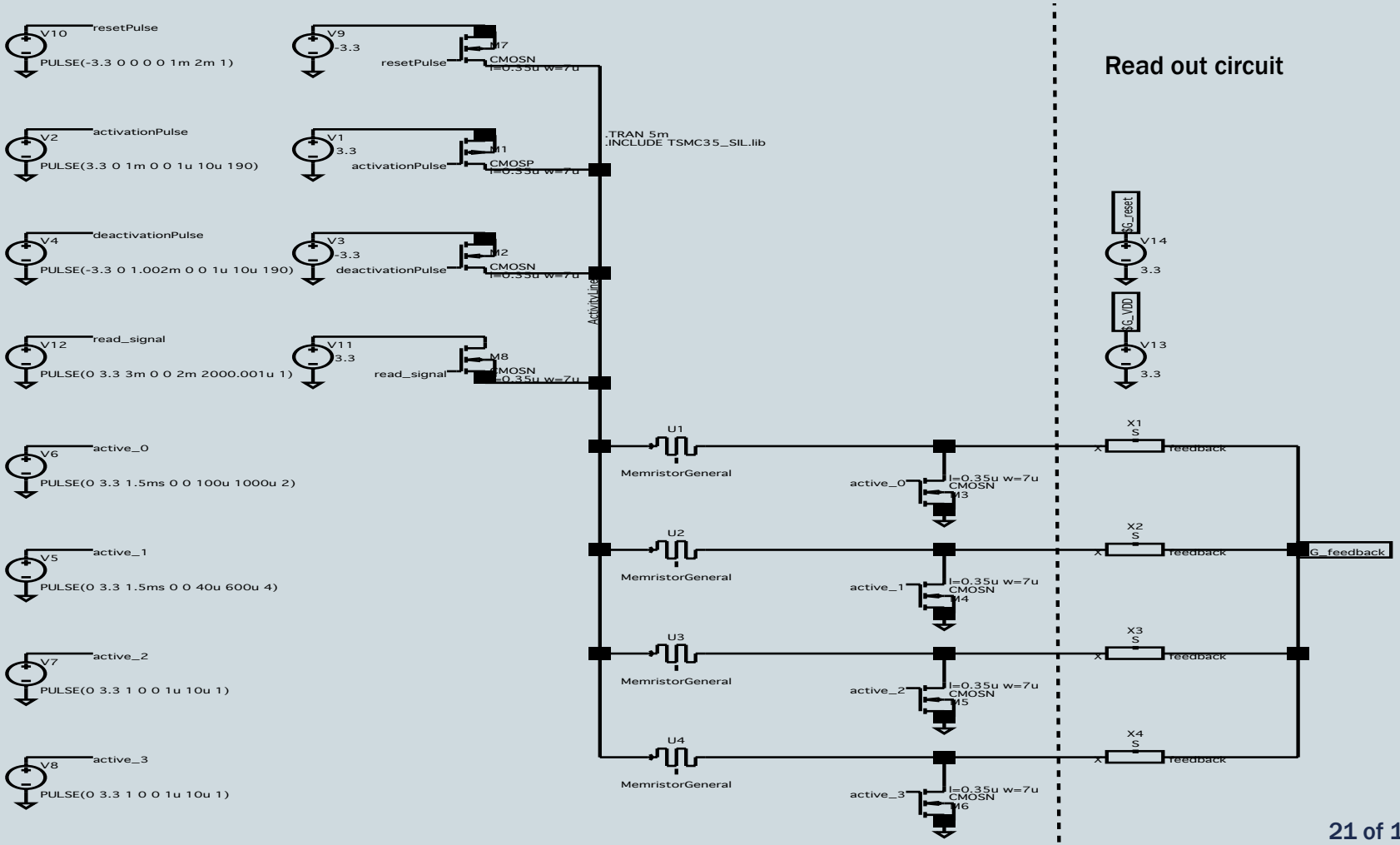
- *B* is computationally complex to maintain and severely affects real-time performance in the current realization of Soar (Soar Manual)

- We propose dedicated hardware to maintain activation

A memristor based analog approach

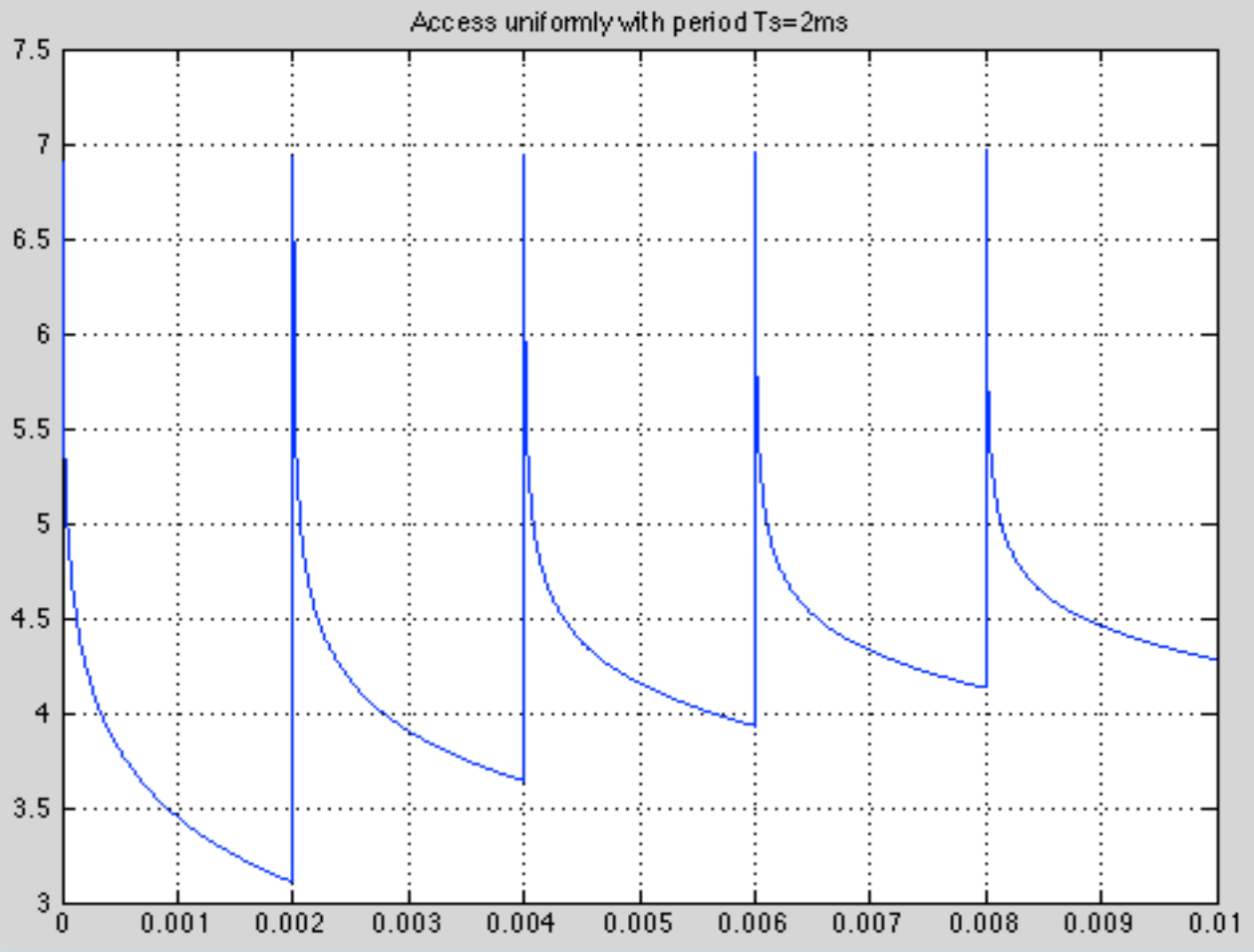
- Spice simulation based on a memristor model extracted from fabricated memristors in publications
- Using voltage pulses to control conductance level hence activation
  - Pulse polarity
  - Pulse width
  - Pulse height
- Low conductance for inactive rows to reduce power dissipation

Activation circuits using memristors



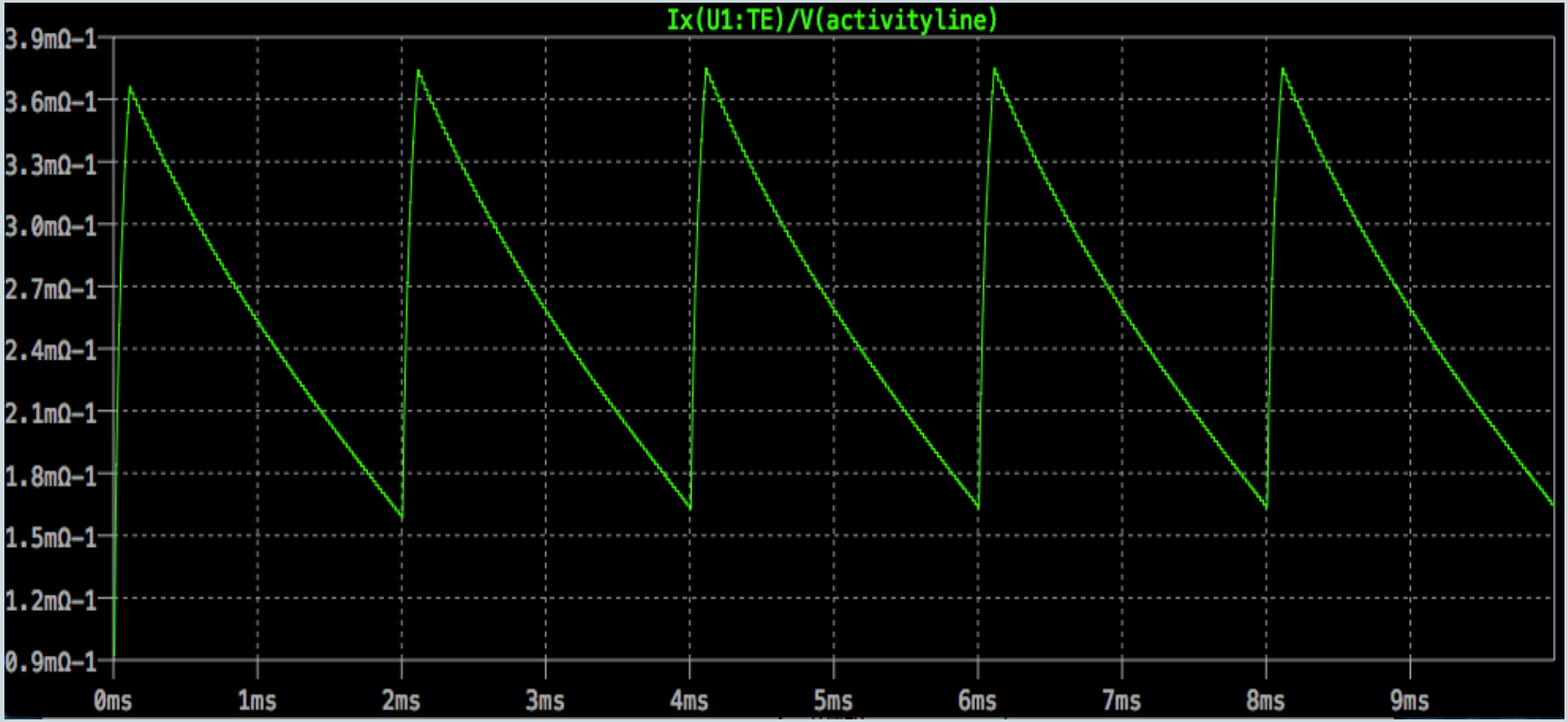
The target

Figure 2: Base level activation with uniform access period of 2ms



Achieved memristor activation

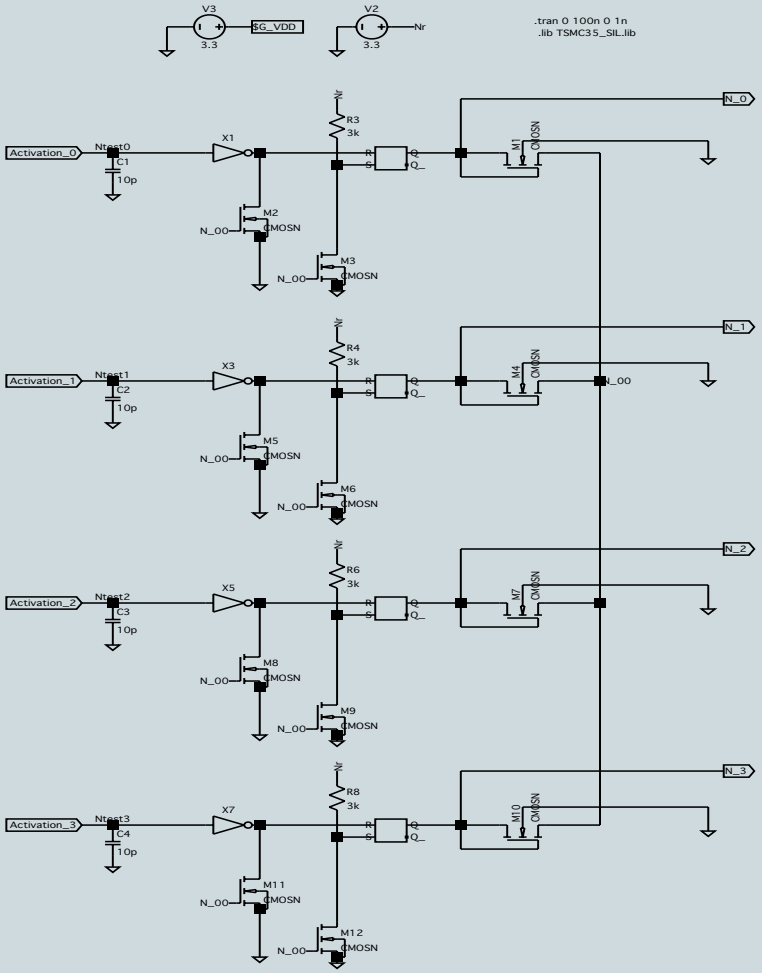
Figure 3: Memristor based activation with access period of 2ms



Differences

- The activation function achieved using memristors reflects frequency and recency of memory access
  - Frequency – increases with accesses
  - Recency – decays when it's not accessed
- It is not the original base level activation function
  - It increases and decays at different rates

Read operation



- Find the maximum activation
  - Voltage racing using RC circuits
  - Compares all activation level in parallel

Summary

- We are designing memristive long-term memory for Soar
  - Memristor based content addressable memory
  - Memristor based activation circuits
- Significance
  - Quick search
  - Hardware based activation
  - Scalable to large, low power memories
- Further work
  - Evaluate area, power and latency of the activation circuit
  - Evaluate the accuracy and resolution of the circuit that finds the maximum activation
  - Characterize the activation function and evaluate its effectiveness for memory retrieval tasks
  - Memristive CAM

QUESTIONS?