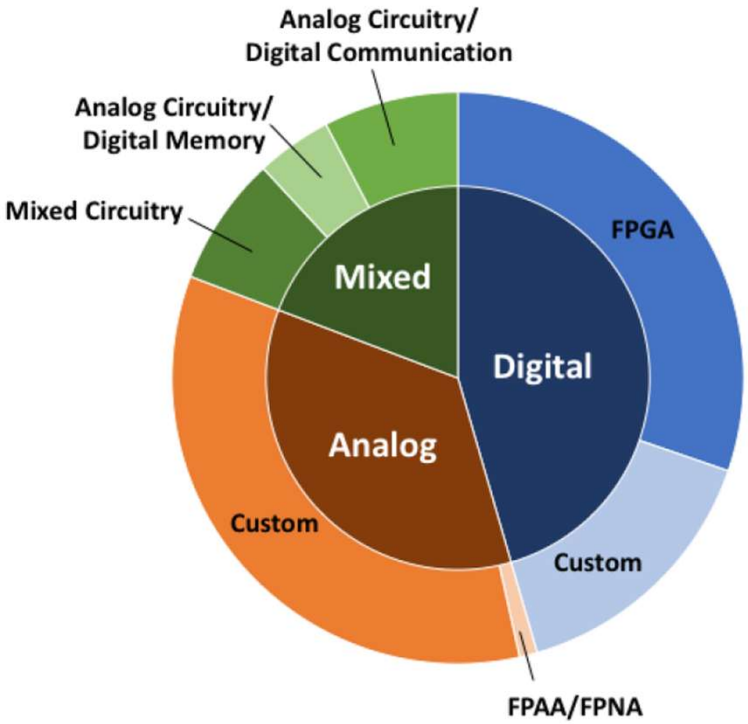
Davide Bertozzi

University of Manchester (UK)

Neuromorphic Communications

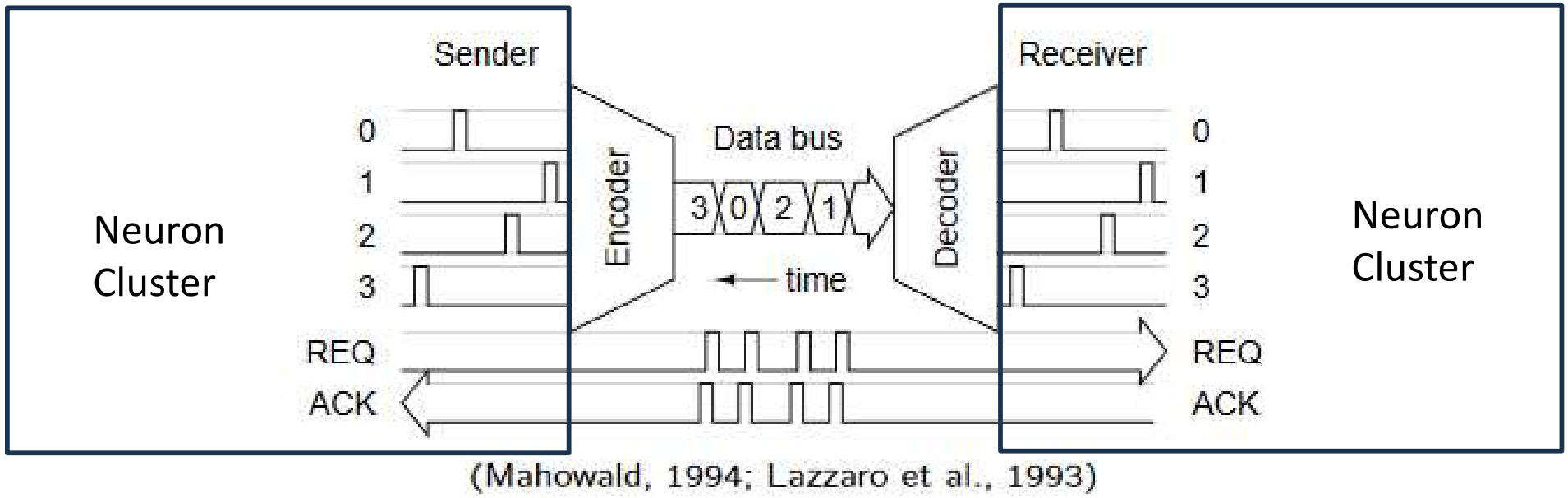


Popularity of design styles for neuromorphic computing
C.D. Schuman et al., «A Survey of Neuromorphic Computing and Neural Networks in Hardware», 2017.

Regardless the specific implementation style of artificial synapses and neurons, the communication infrastructure is DIGITAL

Address Event Representation

Answers the question: how to digitally encode a spike?



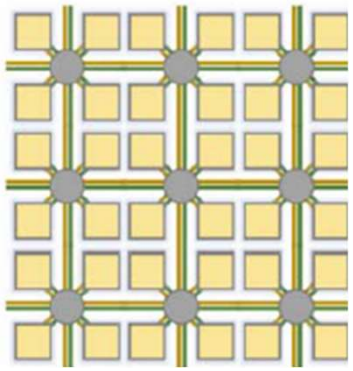
(Mahowald, 1994; Lazzaro et al., 1993)

Each spike is explicitly encoded by its location (its address) and implicitly encoded by the time that its address is sent to the data bus.

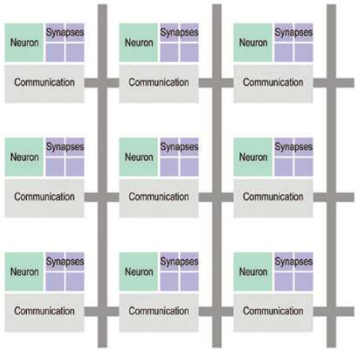
*The **asynchronous implementation** preserves the event-driven nature of the neuromorphic system*

Beyond AER Busses

Today the AER encoding needs to be handled on top of a more scalable communication architecture: Networks-on-Chip

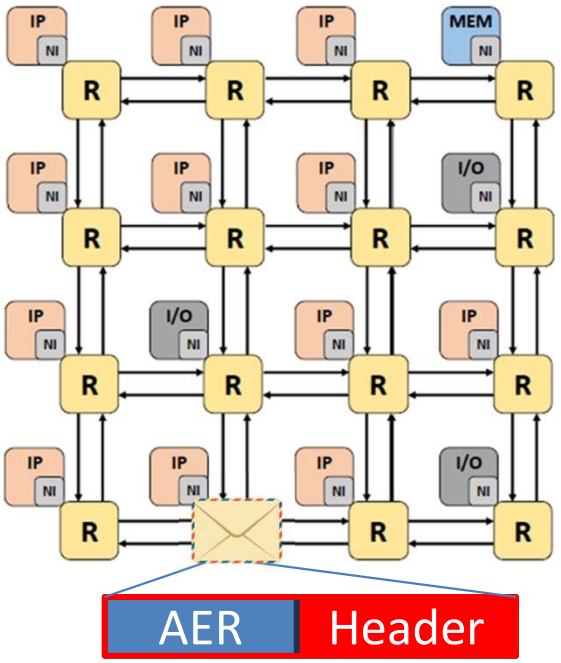


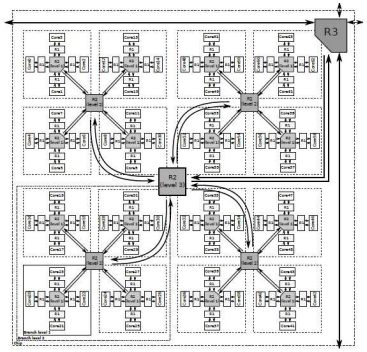
Loihi



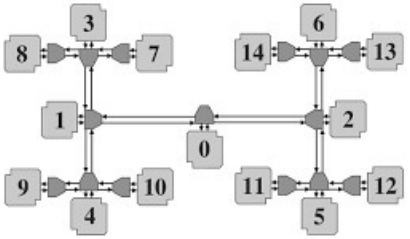
TrueNorth

Concept
2D mesh NoC

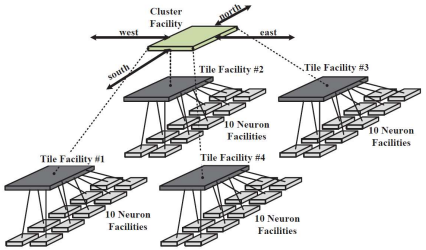




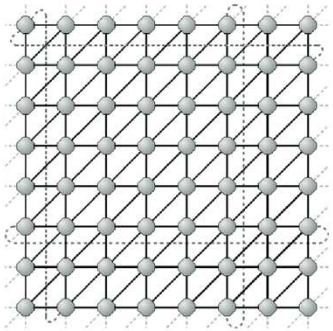
Dynap



Neurogrid



Carrillo et al.



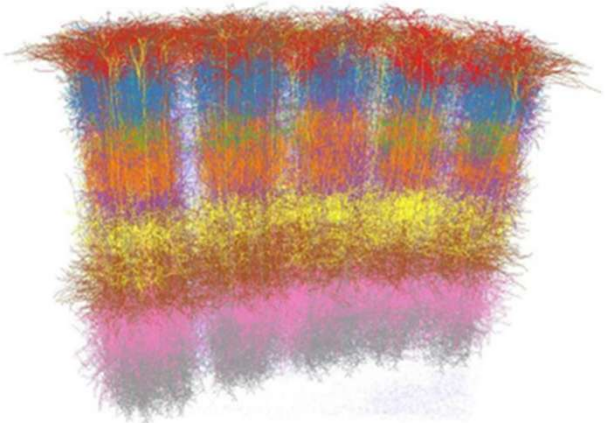
SpiNNaker1

The most relevant NoC implementations are asynchronous

Biological Plausibility

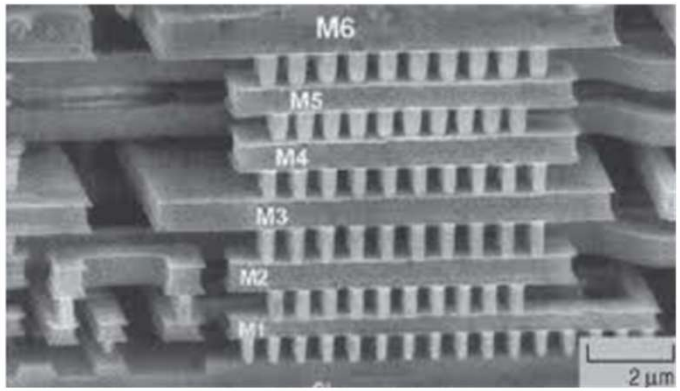
- **NoCs are not biologically-plausible architectures**
  - Brain's 3D connectivity *vs. microelectronics planar interconnection*
  - Direct signaling with high fan-in/fan-out *vs multi-hop signaling with multicast communication capability (largely inefficient)*

Brain: complex 3-D structure



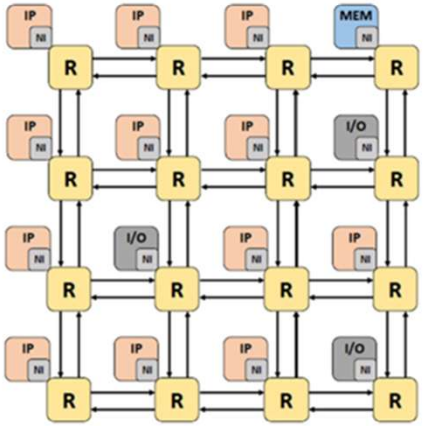
Rat cortex (Oberlaender et al.)

Microelectronics



Source: UMC

Network-on-chip



- **Connectivity is key element of brain's capabilities**
  - We can expect NoCs to become a major source of inefficiency soon

Not Soon, but Now!

Criticality-accelerating factor: **NoCs have been typically force-fitted into the neuromorphic computing domain, without customization for the new application domain**

<<the scalability of neuromorphic systems is mainly restricted by communication requirements. Indeed, some of the main bottlenecks in the construction of large-scale re-configurable neuromorphic computing platforms are the bandwidth, latency, and memory requirements for routing address-events among neurons>>

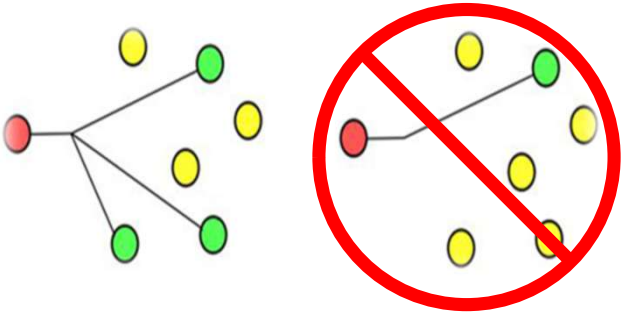
S. Moradi, N. Qiao, F. Stefanini and G. Indiveri, "A Scalable Multicore Architecture With Heterogeneous Memory Structures for Dynamic Neuromorphic Asynchronous Processors (DYNAPs)," in IEEE Transactions on Biomedical Circuits and Systems, vol. 12, no. 1, pp. 106-122, Feb. 2018, doi: 10.1109/TBCAS.2017.2759700

What's Wrong?

- Asynchronous networks-on-chip desirable for event-driven communications

➔ CHALLENGEs:

- Democratizing asynch. design
- Making power and area efficiency available through asynch. design



- AER packets have to be multicast to lots of end nodes

➔ CHALLENGES:

- Addressing schemes of multicast end nodes are poorly scalable
- Branching in hardware comes with large overhead

- The synaptic connectity of neurons must be programmable to map different SNNs

➔ CHALLENGE:

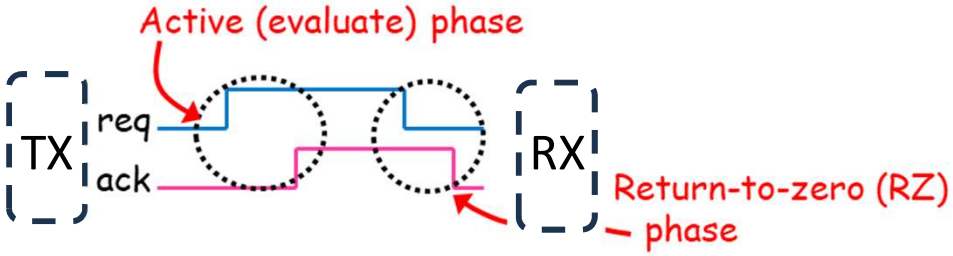
- Prohibitive storage requirements

What's Wrong with Asynchronous Design?

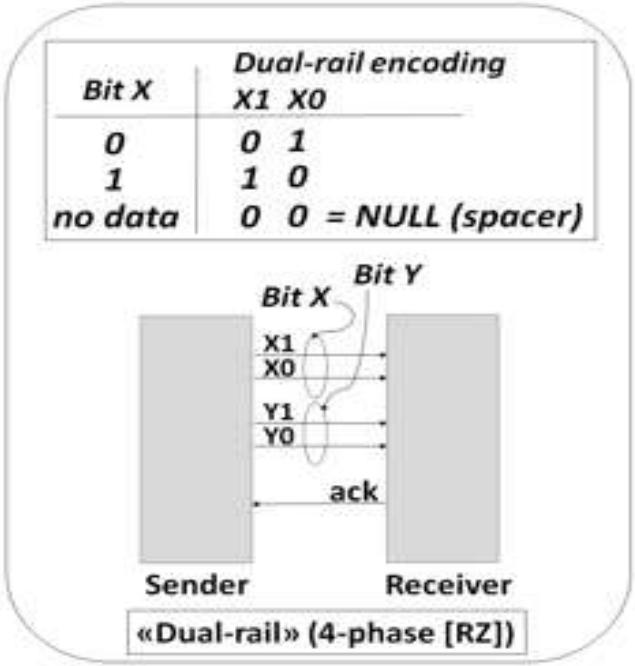
The design style - Asynchronous design decisions oriented to extreme robustness and to simplify large-scale physical design

The signalling protocol

Frequent choice: 4-Phase Handshaking



The data encoding scheme



Frequent choice: Delay-insensitive codes

- «req» replaced by encoded data bits
- Correct operation regardless gate and wire delay

Simplify hardware design and enforce extreme robustness

…..at the cost of low throughput, high area occupancy, poor coding efficiency and high energy per bit

The Problem

- **Is this really the best match for the ultra-low power targets of neuromoprhic computing?**
- **For many practical systems, these robustness guarantees are overdesigned..**

There is a quest for a less conservative design style leading to major overall cost benefits!

What's Wrong with Asynchronous Design?

Tool flow - Custom flows employed that derive from the asynchronous digital design tools initially developed at Caltech in the 1990s

Top-down refinement of functional router specifications

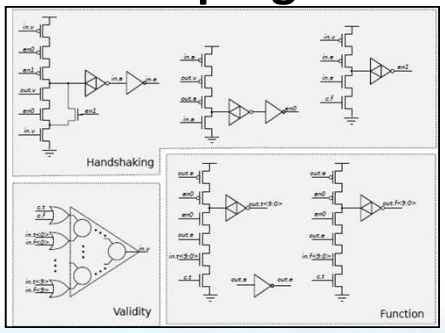
CHP: programming notations to specify and compose circuits

```
chp{
   *[[v(in)];
   [sig.t → out.d⇑; [n(in) ∧ n(sig)]; out.d⇓;
   []
   sig.f → skip]; [n(in) ∧ n(sig)];
   ]
}
```

HSE: intermediate representation of CHP constructs

```
hse{
   *[[ sig.t → out.b[i].d[j] := in.b[i].d[j]; in.a
   +; en0−;
   ([out.a]; out.b[i].d[j]−; [~out.a]), ([n(in)];
   in.a−); en0+
   []
   sig.f → en1+; in.a+; [n(in)]; en1−; in.a−;
   ]]
}
```

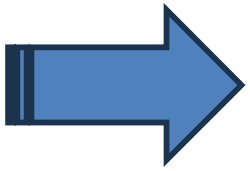
From HSE programs to production rules



Source: S.Moradi, Q.Ning, F.Stefanini, G. Indiveri: «A Scalable Multicore Architecture With Heterogeneous Memory Structures for Dynamic Neuromorphic Asynchronous Processors (DYNAPs)». IEEE Trans. Biomed. Circuits Syst. 12(1): 06-122 (2018)

The Problem

- The abstract design entry leads to “unnatural” microarchitecture designs, with suboptimal efficiency
- These are not the tools commonly used by industry



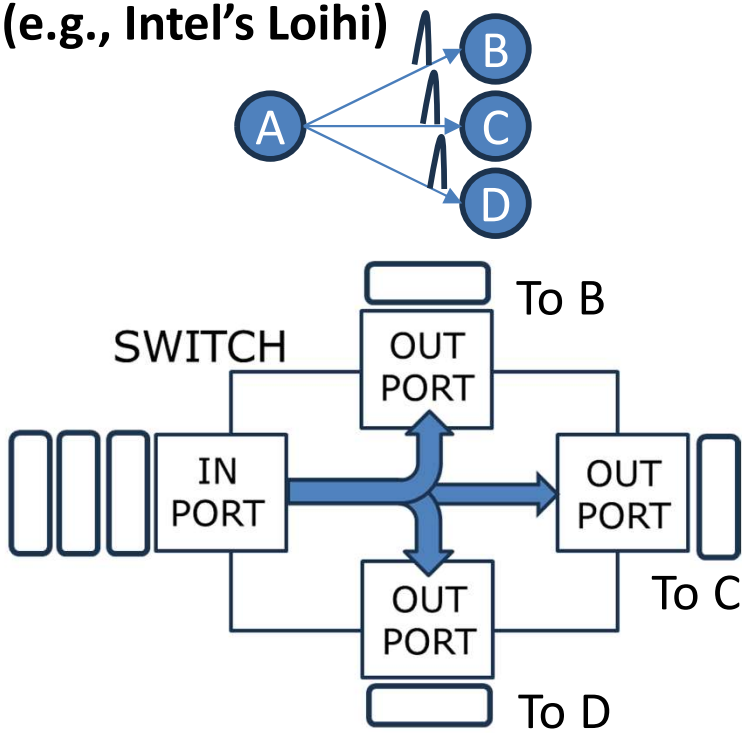
There is a quest for:

- **Design entry at a lower abstraction layer (e.g., RTL)**
- **Using mainstream industrial clocked CAD tools**

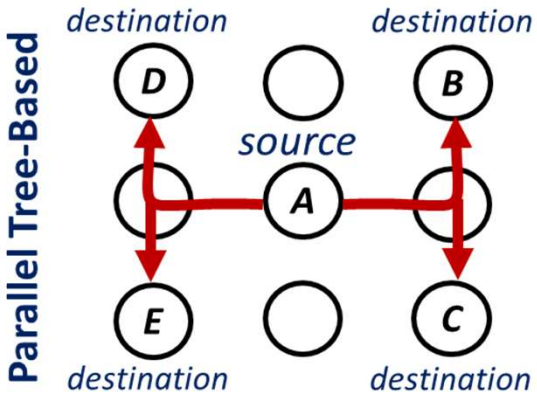
What's Wrong with Multicast?

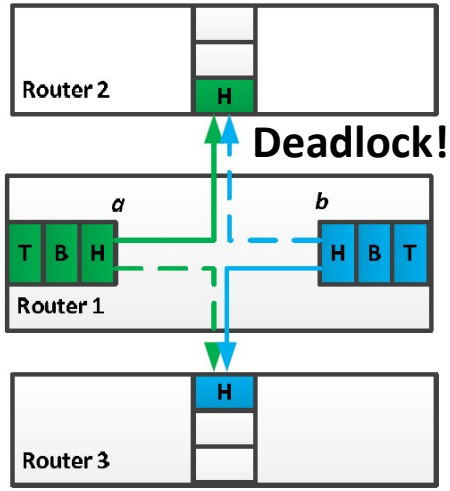
- Efficient and robust implementation of multicast (i.e., one to many) transactions is still an open issue

Common approach: Packet Replication (e.g., Intel's Loihi)



Why not parallel tree-based multicast?



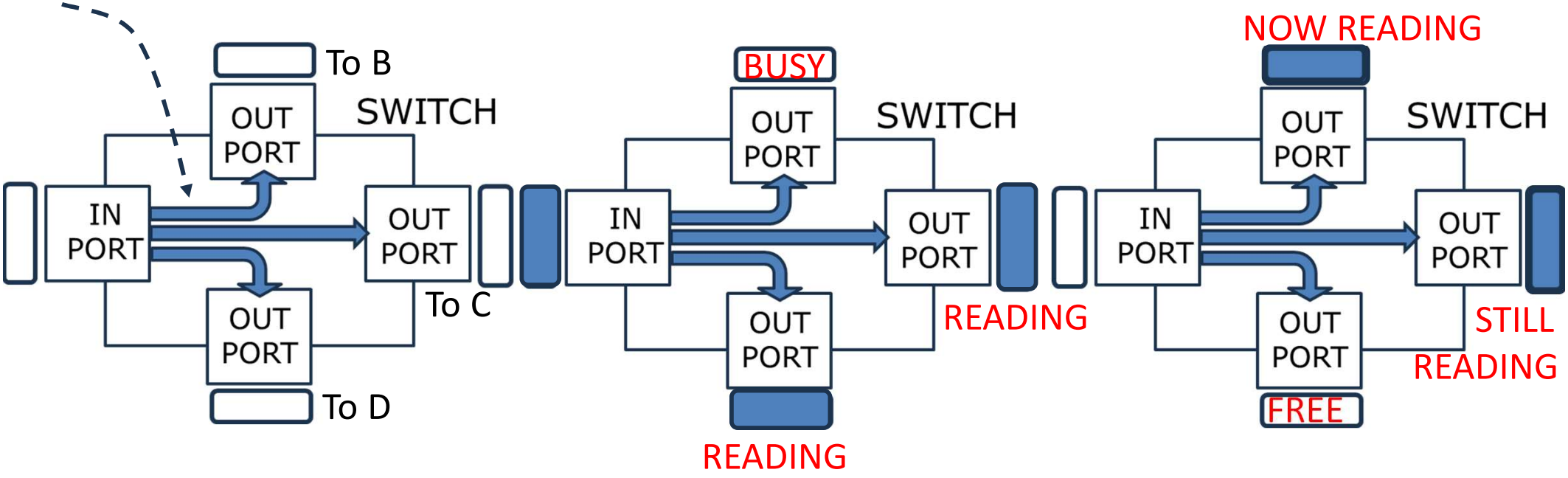


20+ years of NoCs teach that deadlock avoidance comes with large overhead

What's Wrong with Multicast?

- Efficient and robust implementation of multicast (i.e., one to many) transactions is still an open issue

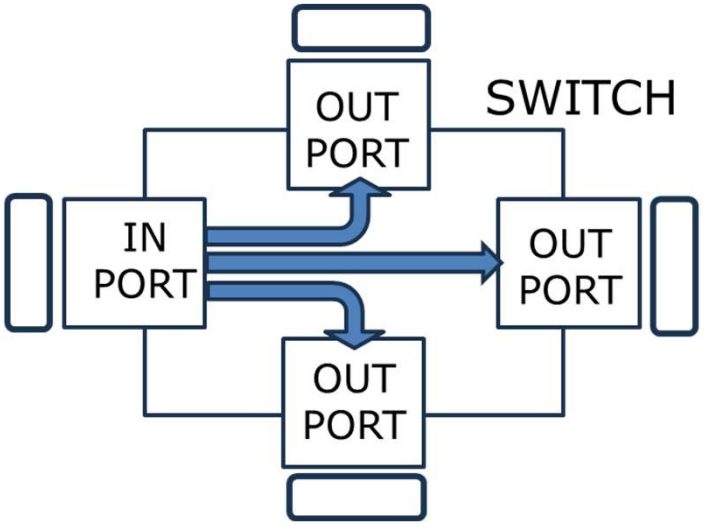
Solution: state-of-the-art <u>asynch</u>. switches (never applied to neuromorphic computing) use input buffers with multiple read ports for concurrent and independent readout



What's Wrong with Multicast?

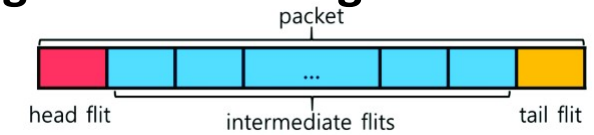
- Efficient and robust implementation of multicast (i.e., one to many) transactions is still an open issue

Solution: state-of-the-art <u>asynch</u>. switches (never applied to neuromorphic computing) use input buffers with multiple read ports for concurrent and independent readout



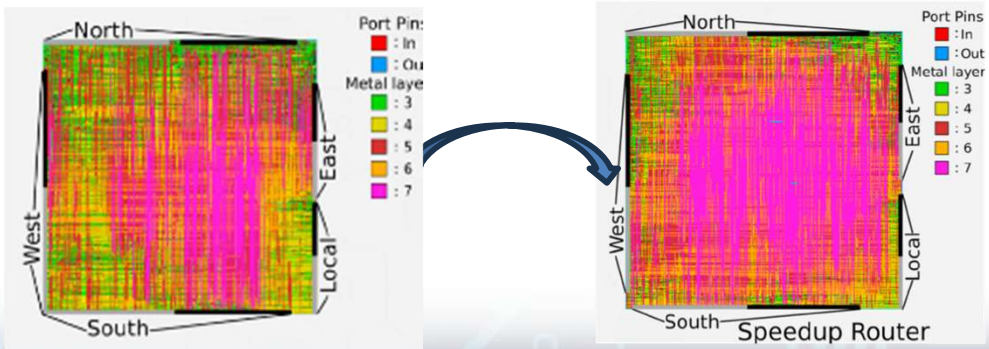
Deadlock freedom requirements still severe:

a) **VCT switching: buffers as large as worst-case packets**



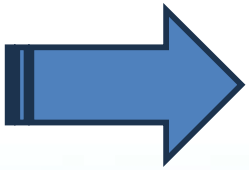
b) **Processing one packet at a time at IN ports**

Large wiring overhead that jeopardizes timing closure



The Problem

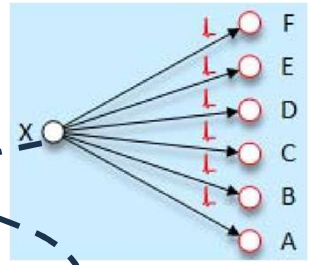
- **The efficient and robust implementation of multicast branching is still a long way off**
- **Several solutions still conservatively prefer packet replication at the source….**



Need for deadlock-free hardware-enabled multicast routing, to be made available through asynchronous design

What's Wrong with Multicast?

- Addressing scheme of multicast end nodes

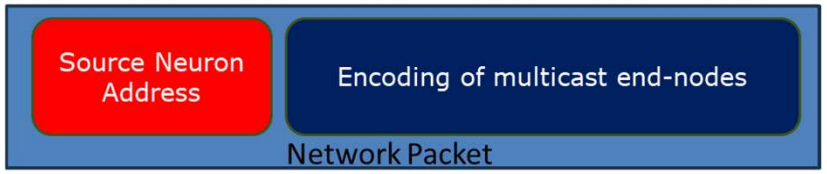


Neuron has memory

Fanout	
Entry 1	Neuron F
……	…..
…..	…..
Entry F	Neuron A

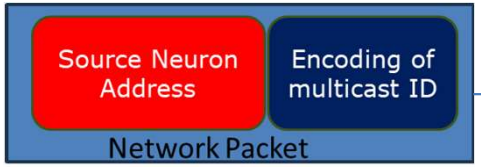
This is already a large overhead (often kept off-chip)

AER NoC packet

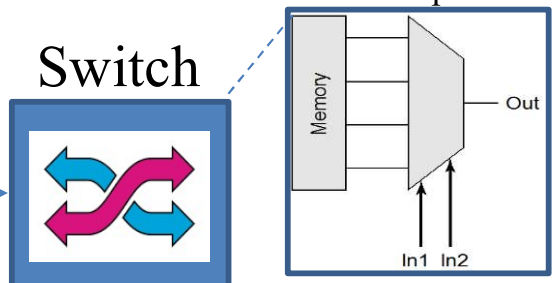


Inherent Challenge:
Large packet header overhead

AER NoC packet



Switch



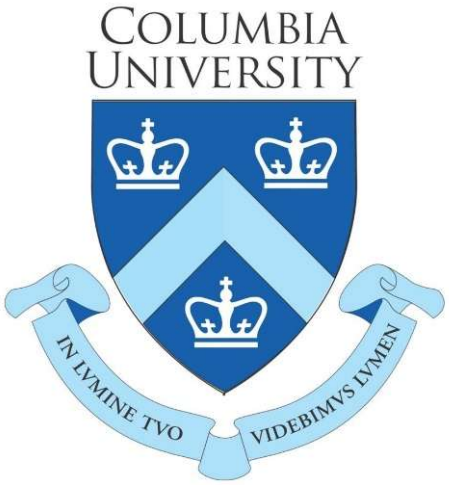
Take South & East!

Inherent Challenge:
Large look-up table overhead (e.g., 10s kBs)

Key Contribution: TaBuLA

TaBuLA: an ultra-low power *and* flexible asynchronous interconnect technology dealing with the efficiency limitations of current neuromorphic NoCs

Davide Bertozzi



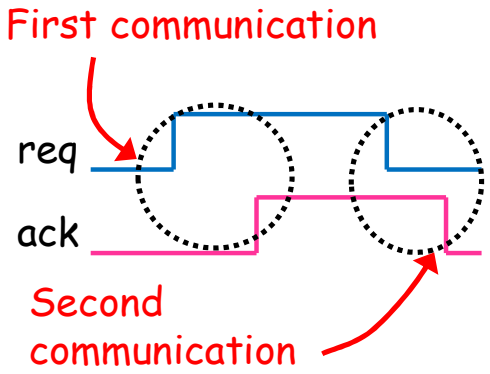
Steve Nowick

- **What's wrong with asynchronous design?**
  - **Asynchronous Design Style**
    - ✓ High-performance signalling + High coding efficiency….
    - ❖ *…at the cost of relative timing constraints (RTCs)*
  - **Asynchronous Design Flow**
    - ✓ Synthesis tool flow with industrial CAD tools
    - ✓ Synchronous-equivalent design flexibility
    - ✓ Design entry at microarchitectural level
    - ❖ *..at the cost of iterative synthesis*
- **What's wrong with multicast communication?**
  - Hardware-enabled packet branching
    - Cost-effective and deadlock-free multicast
    - ❖ *..at the cost of restricting packet format*
  - Multicast addressing
    - ✓ Work in progress

Largely outperforming state-of-the-art NoCs for edge neuromorphic computing

(1) Foundation – Async. Design Style

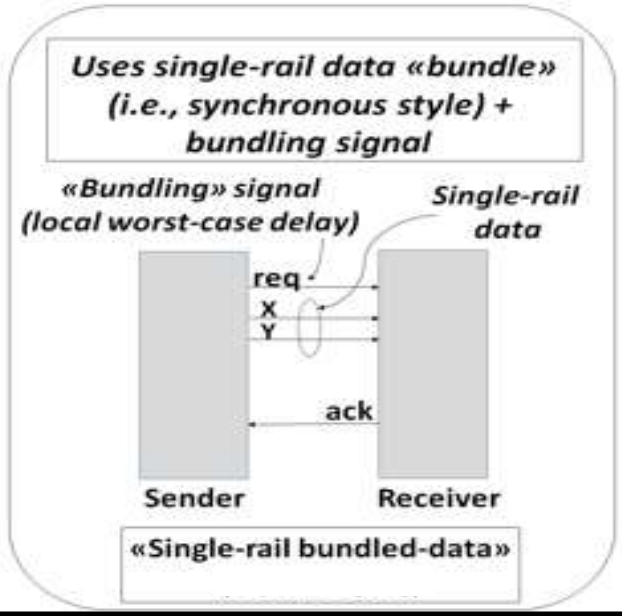
2-phase handshaking



Potential for
- *High throughput*

CHALLENGE:
- ***Switch design with control signals not returning to zero.***

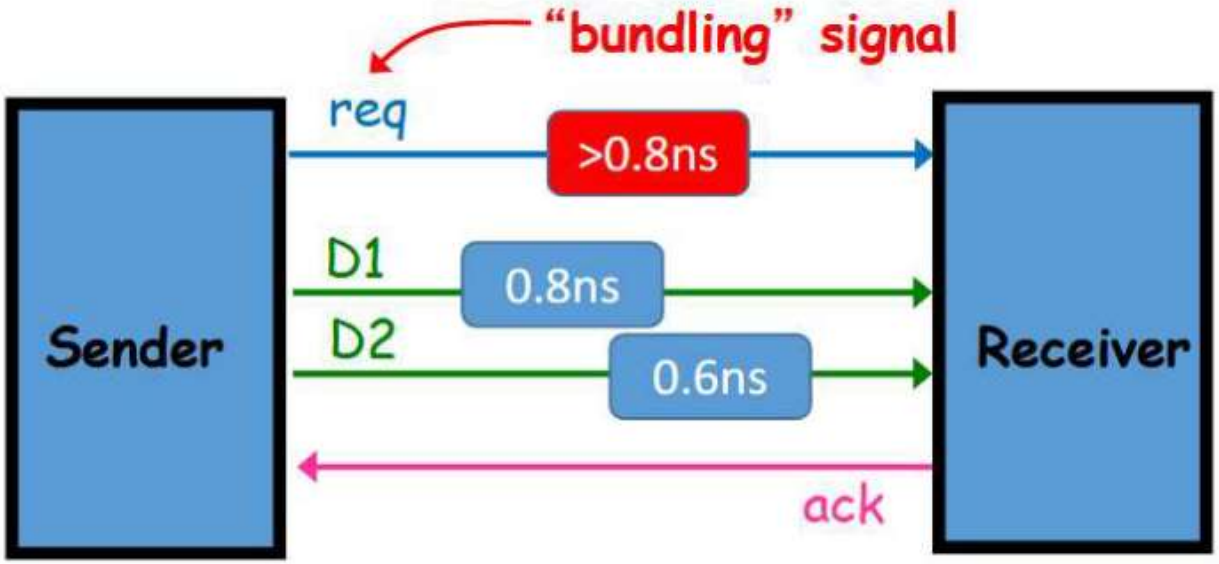
Single-rail bundled-data



Potential for
- *Higher coding efficiency*
- *Lower area*
- *Lower energy-per-bit*
- *Higher-radix switches*

CHALLENGE:
- ***One-sided relative timing constraint (see next)***

Timing in Bundled-Data NoCs

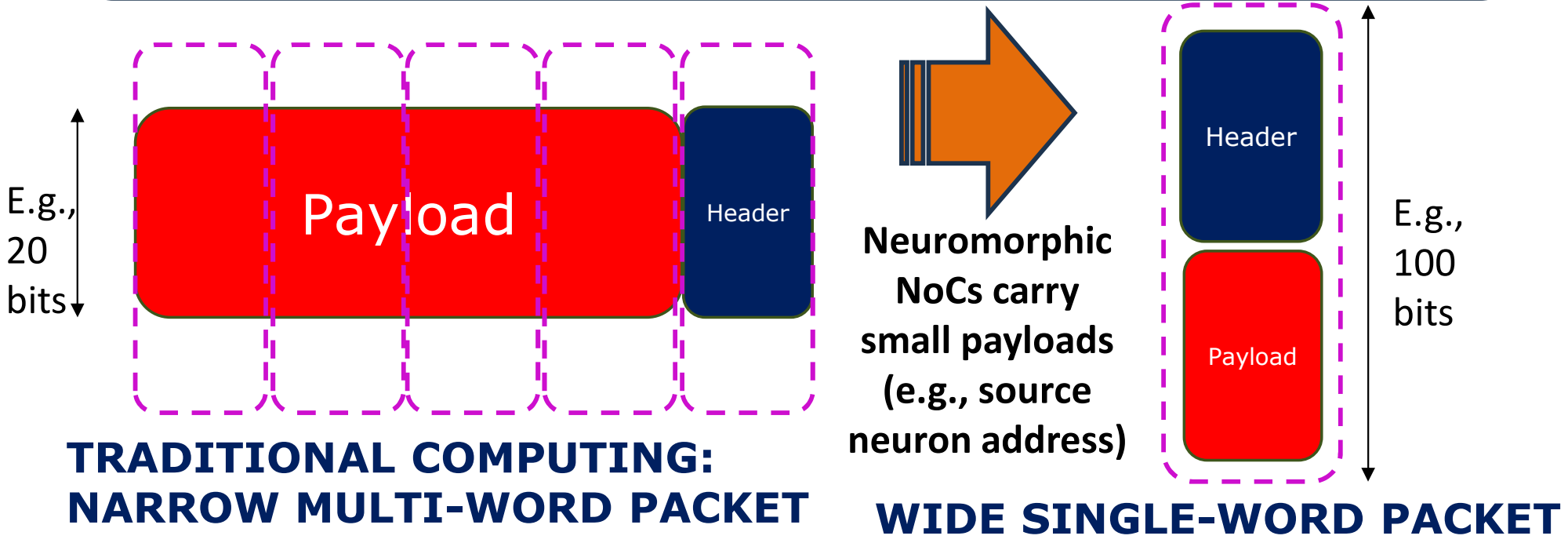


Bundling Constraint: the REQ delay should be longer than the worst-case data delay. Unlike synchronous timing, such constraints are localized.

CHALLENGE:
- *It is a relative timing constraints (RTCs) typically met by inserting small matched delays on REQ*
- *Commercial CAD tools offer poor support for RTCs, since they "accept" min/max delays constraints with absolute timing only.*

(2) Foundation - Architecture

Compact single-word (flit) packets are the most appropriate encoding format for spike messages in scaled process nodes



TRADITIONAL COMPUTING: NARROW MULTI-WORD PACKET

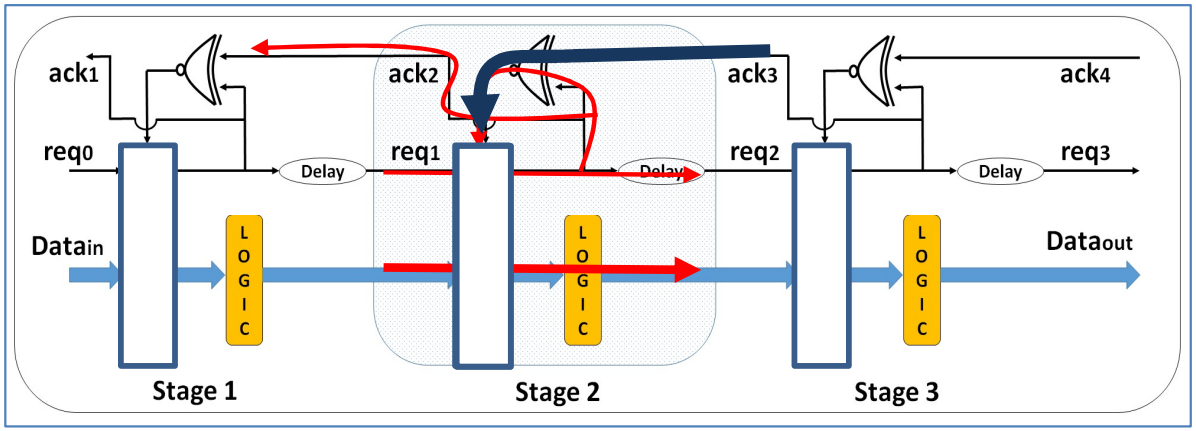
WIDE SINGLE-WORD PACKET

- ❑ **Potential for radical architectural streamlining**
- ❑ **Deadlock-free multicast from the ground up**

3) Foundation: Asynchronous Pipeline

The NoC switch is based on a *high-throughput* and *low-overhead* asynchronous pipeline: Mousetrap

- It uses a two-phase protocol and single-rail bundled-data encoding
- Mousetrap uses simple control circuits (a single exclusive-NOR gate) and data registers (a single bank of level-sensitive D-latches)

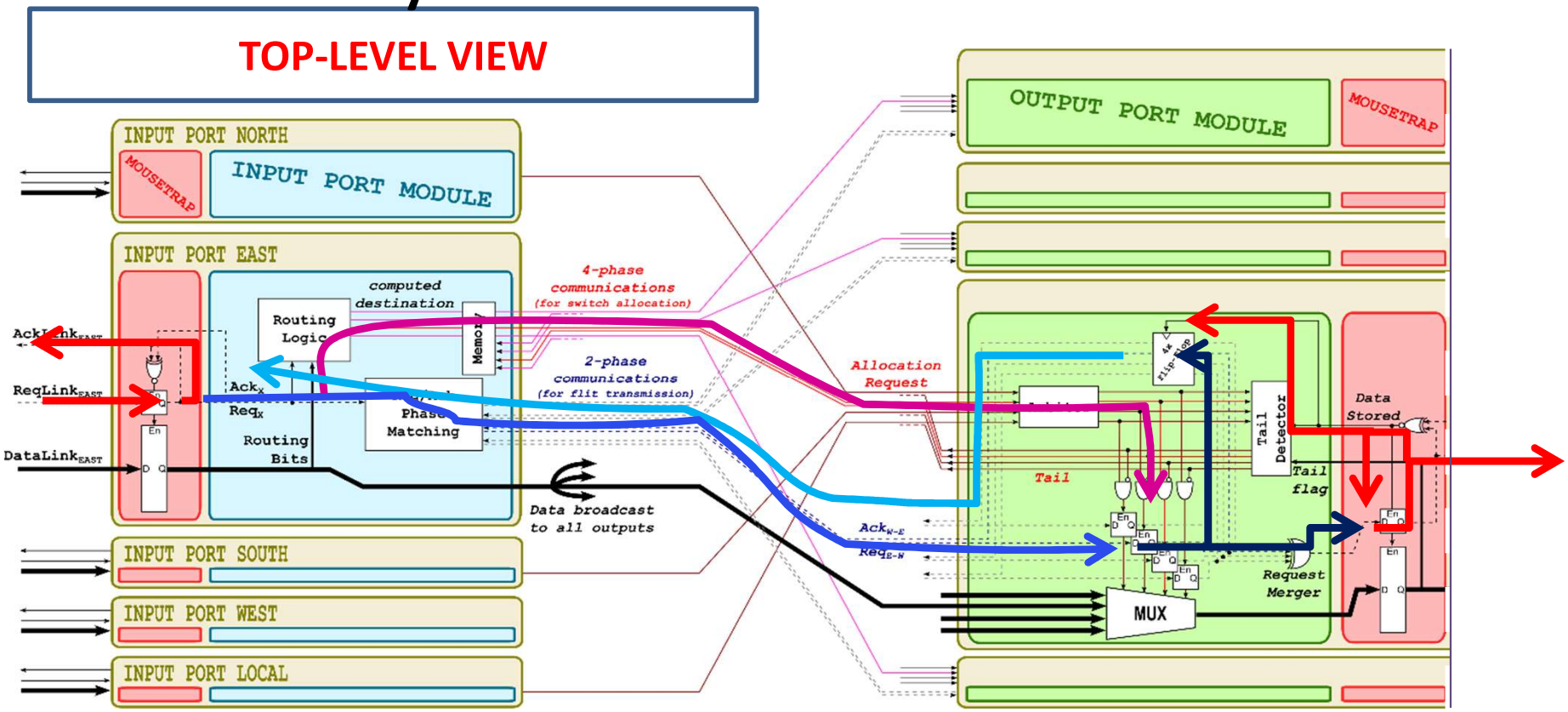


M. Singh and S. M. Nowick, "*MOUSETRAP: high-speed transition signaling asynchronous pipelines*", IEEE Tran. on VLSI, 2007.

Low latency: single-latch registers are *normally transparent,* only closing transiently to protect data immediately after it enters a stage

Low power/area: trivial implementation overhead

TaBuLA Asynchronous Switch Architecture



A. Ghiribaldi, D. Bertozzi and S.M. Nowick, "A transition-signaling bundled data NoC switch architecture for cost-efficient GALS multicore systems," DATE 2013

Request received from the input, immediately acknowledged as it passes the pipeline.

Fast 2-phase propagation path for the forward request

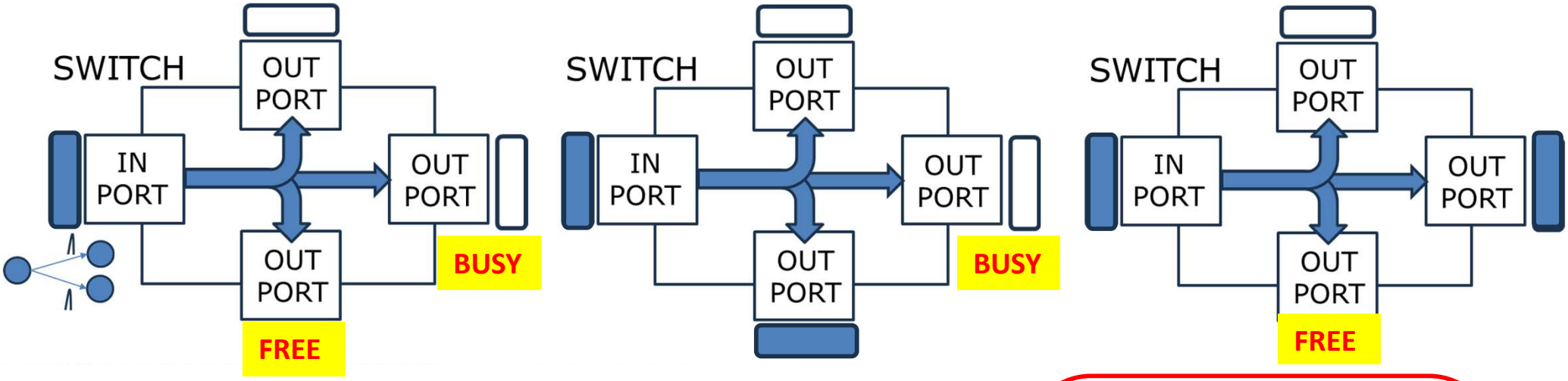
Concurrently, header goes through routing logic and switch allocation

Data arrives at the output pipeline stage and closes the output Mousetrap

Ack sent backwards. I/O path is deasserted and a new packet is enabled in.

Multicast Support

Concurrent and independent readout of multicast packets from output ports with a single read port per input buffer while enabling full switch-level concurrency



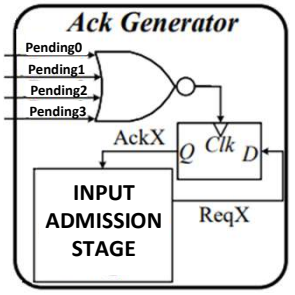
Distinctive challenge with asynch. multicast

Detecting completion of operations:

- across a wide range of multicast tx. patterns
- with arbitrary downstream traffic delays

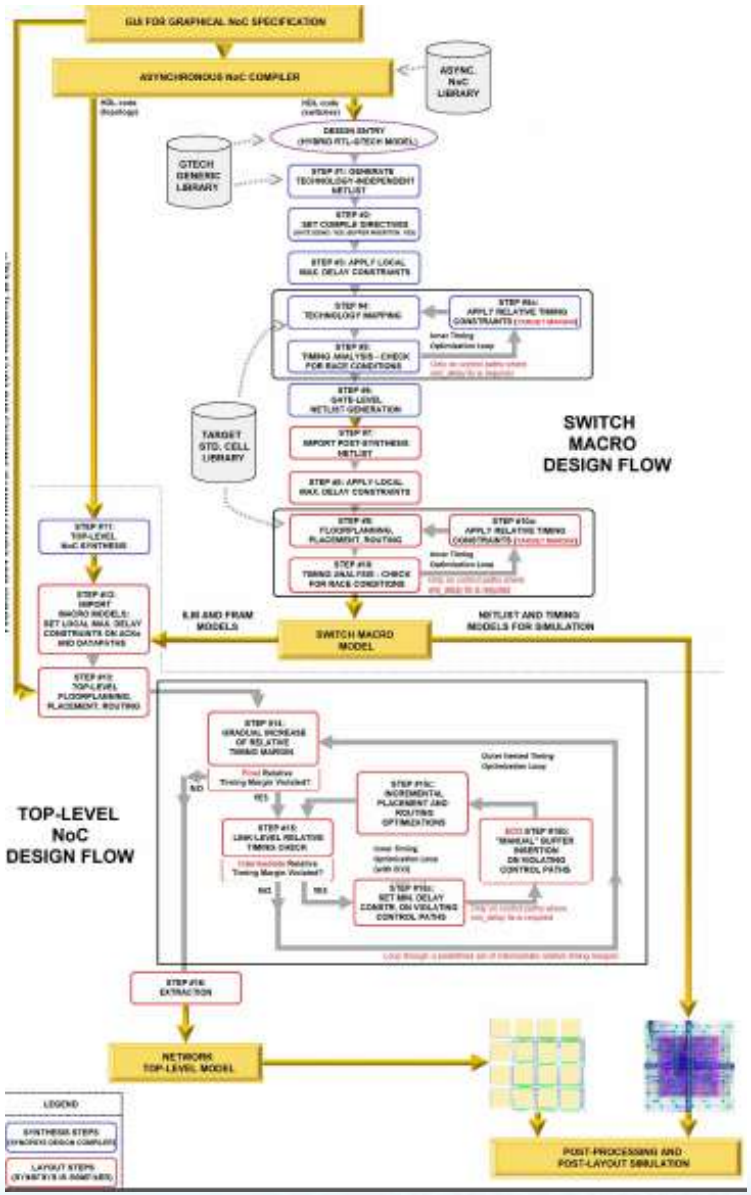
without the use of a fixed-rate strobing clock.

TaBuLA achieves this at trivial circuit overhead



Key intuition: in a 2-phase protocol Ack is always a delayed version of Req

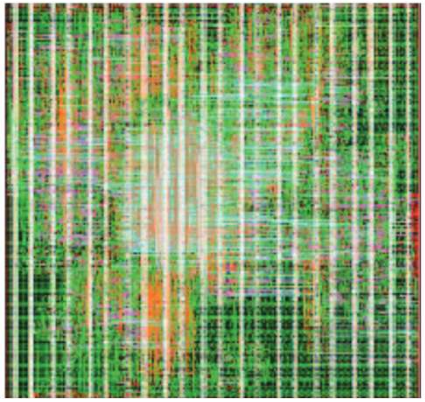
TaBuLA's CAD Tool flow



- ❑ Uses mainstream industrial CAD tools
- ❑ From RTL to layout

Delivers:

- ❑ Synchronous-equivalent design flexibility
- ❑ Tight control over relative timing constraints
- ❑ Effective convergence of several bundling constraints in parallel

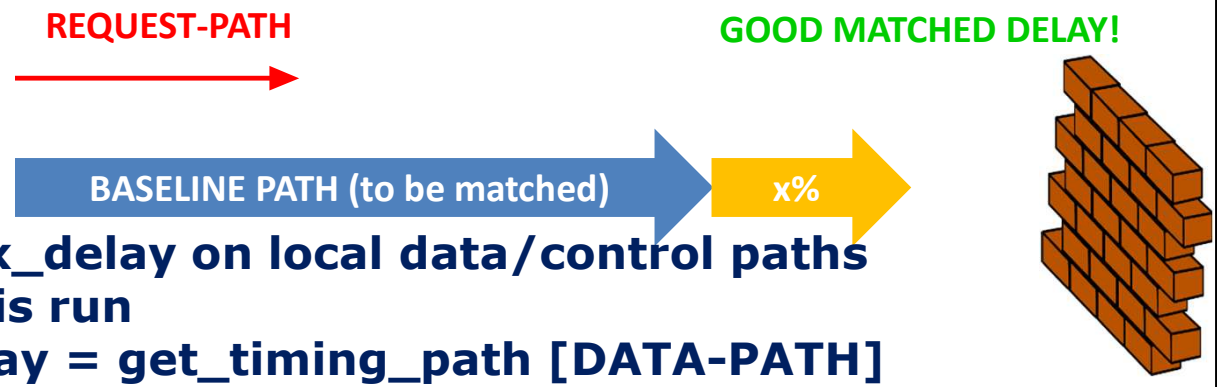


Worked in 40/22/14nm

Now working on FPGA!

Timing Optimization Loop

A *baseline iterative procedure* that fine-tunes control path delays associated with locally-optimized datapath delays



Set_max_delay on local data/control paths
Synthesis run
DataDelay = get_timing_path [DATA-PATH]
TargetReqDelay = $DataDelay*(1+x%)
set_min_delay = $TargetReqDelay [REQUEST-PATH]
Iterative synthesis run

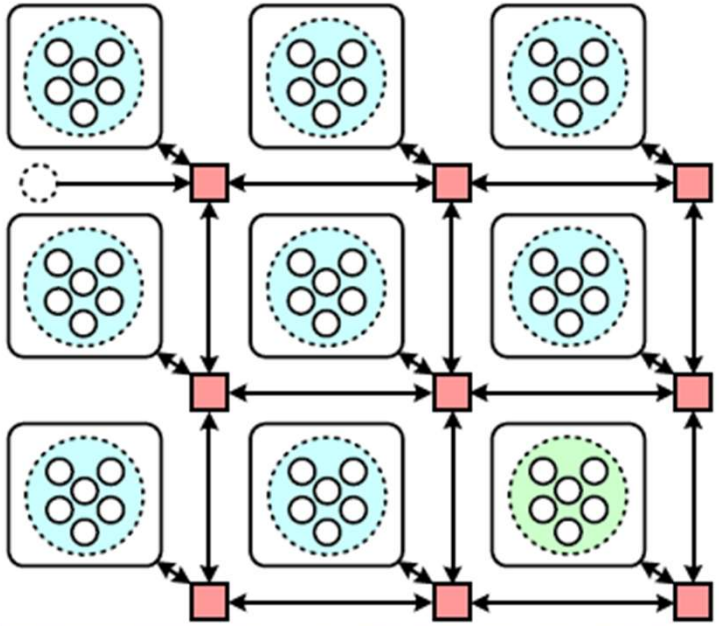
The relative timing constraint should not be overdesigned not to waste performance through idleness
set_max_delay = $Upperbound[REQUEST-PATH]

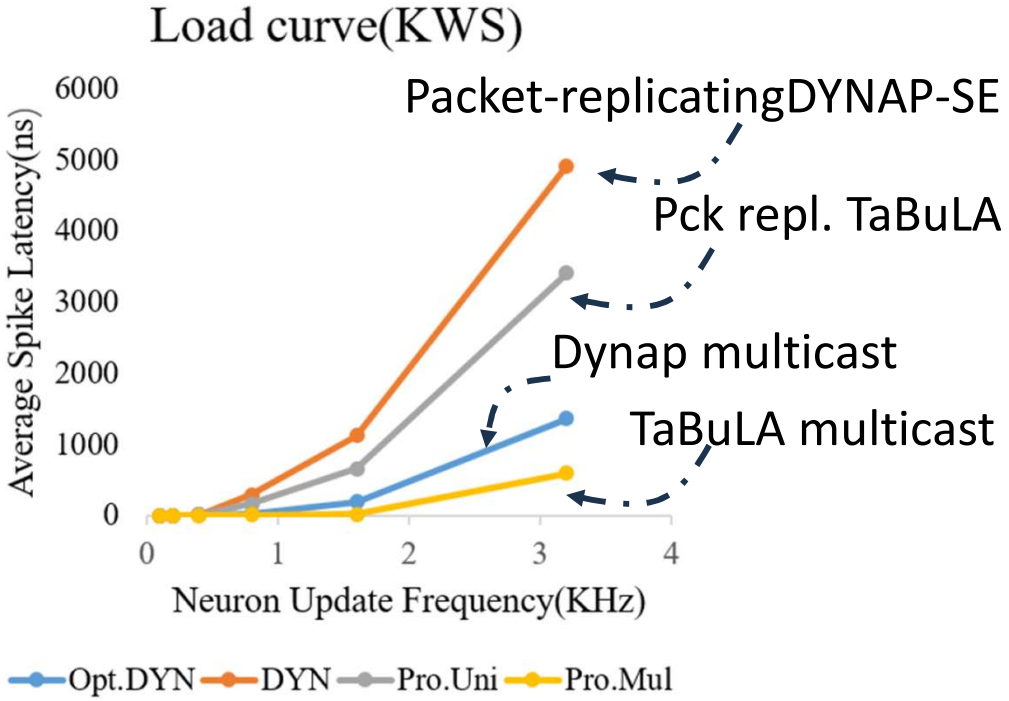
Experimental Validation: Edge Neuromorphic Computing (1)

Comparison with state-of-the-art neuromorphic NoC for **edge computing** in 22nm GlobalFoundries technology (post-layout analysis): **DYNAP-SE from Univ. Zurich**

Network-level comparison

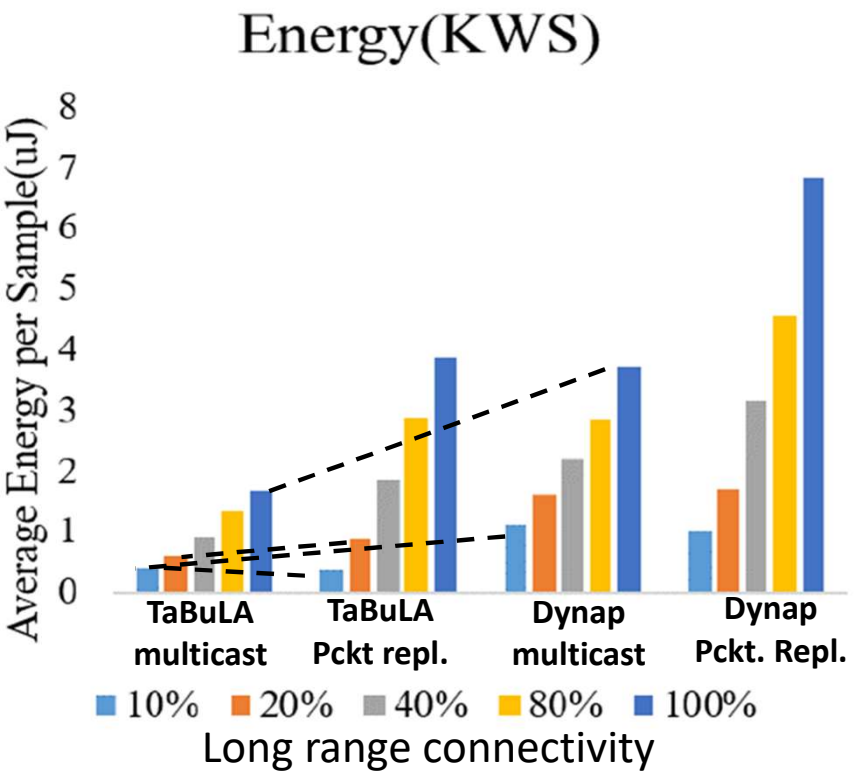
Testbench: execution traces from the simulation of 3-layer Recurrent Spiking Neural Networks
Application: 1-word key word spotting





- ✓ ***Full-featured TaBuLA*** **exhibits up to 63% higher saturation rate over multicast-enabled DYNAP-SE NoC.**
- ✓ ***Naive packet replication approaches*** **fail to compete with multicast-enabled NoCs.**

Experimental Validation: Edge Neuromorphic Computing (2)



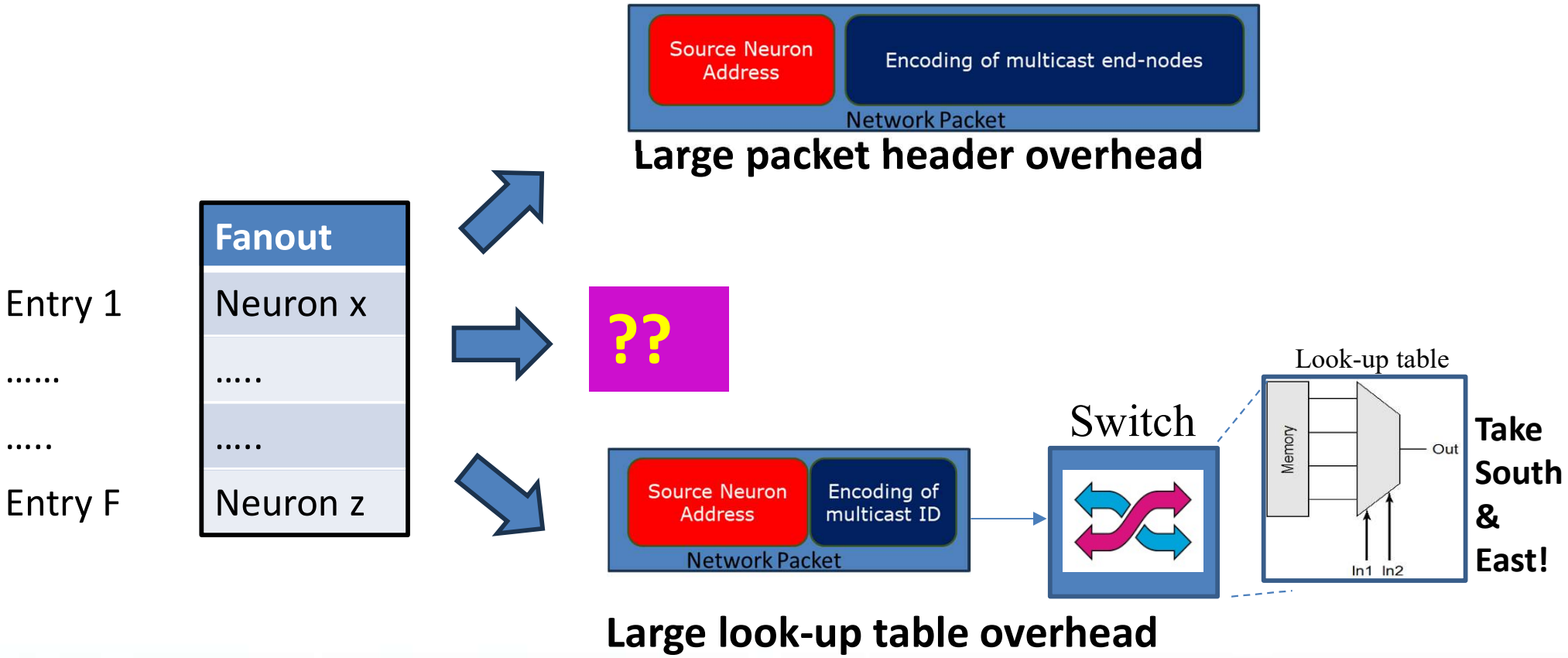
- ✓ Packet replication with *pckt. repl. TaBuLA* is the most energy-efficient solution with overly high sparsity (<10% long-range connections).
- ✓ Hardware multicast in *TaBuLA Multicast* pays off already at sparsity levels above 20%, with energy savings over *pckt. repl.* ranging from 31 to 68% with long-range connectivity of 20% and 100%, respectively.

- ✓ Energy savings of *TaBuLA Multicast* over *DYNAP-SE Multicast* range from 41% to 59% depending on the sparsity level.

A new neuromorphic processor from University of Zurich with TaBuLA NoC inside (NeoCorAl) is currently being qualified on 22nm silicon (test chip arrived, PCB under development)

Future Work

- Cut down on the significant overhead for addressing multicast end nodes
  - Critical issue for large-scale neuromorphic systems



Conclusions

- AI systems at all scales are far from biological efficiency
- Neuromorphic computing (SNNs + custom-tailored hardware) is a promising way of thinking out of the box
- Neuromorphic communications are coming out as a major power and scalability bottleneck
- New advances are needed in asynchronous NoC design and hardware support for multicast transactions
- TaBuLA is gaining momentum for neuromorphic communications in edge computing platforms
  - ✓ Ultra-low power, designed with mainstream CAD tools
- How to efficiently address a large number of multicast end nodes is still an open issue