

Programmable Switch Hardware

ECE/CS598HPN

Radhika Mittal

Conventional SDN

- Programmable *control plane*.
- Data plane can support high bandwidth.
 - But has limited flexibility.
- Restricted to conventional packet protocols.

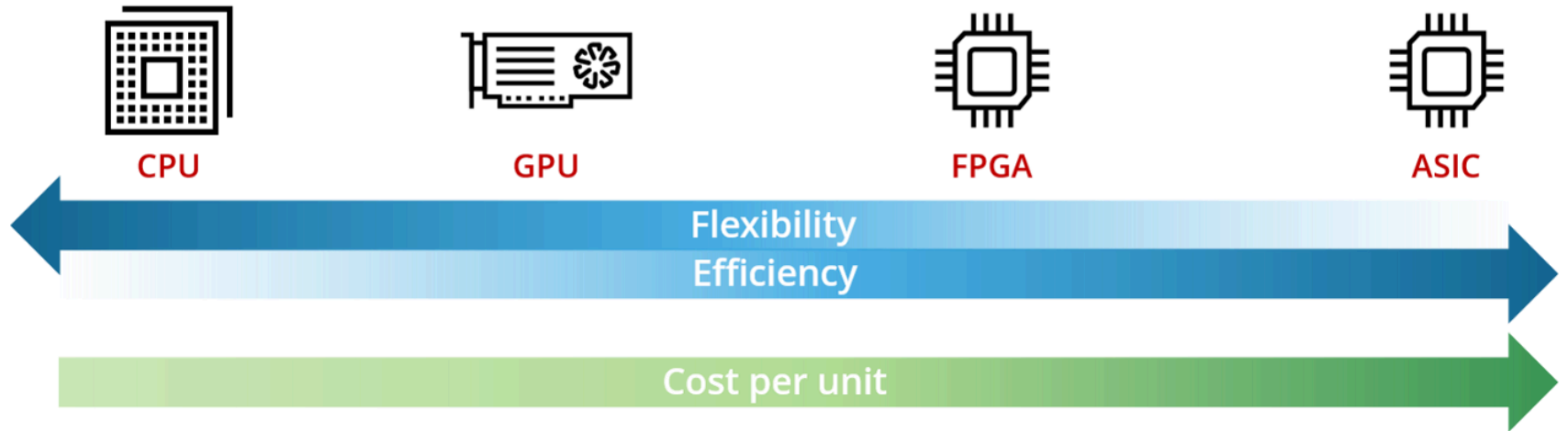
Software Dataplane

- Very extensible and flexible.
- Extensive parallelization to meet performance requirements.
 - Might still be difficult to achieve 100's of Gbps.
- Significant cost and power overhead.

Programmable Hardware

- More flexible than conventional switch hardware.
- Less flexible than software switches.
- Slightly higher power and cost requirements than conventional switch hardware.
- Significantly lower than software switches.

Other alternatives?



Forwarding Metamorphosis: Fast Programmable Match- Action Processing in Hardware for SDN

Pat Bosshart, Glen Gibb, Hun-Seok Kim,
George Varghese, Nick McKeown, Martin Izzard,
Fernando Mujica, Mark Horowitz

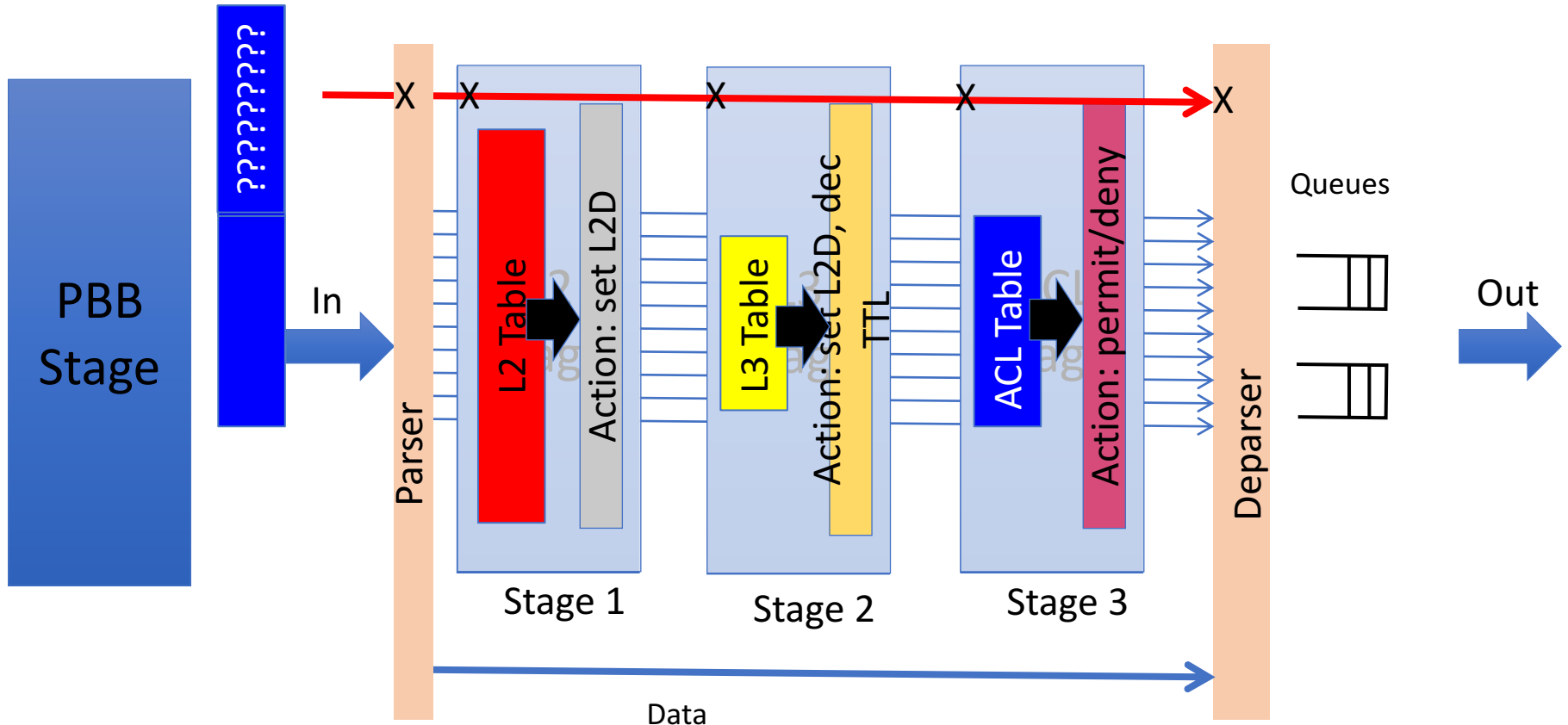
Acknowledgements: Slides from Pat Bosshart's SIGCOMM'13 talk

Fixed function switch

L2: 128k x 48
Exact match

L3: 16k x 32
Longest prefix
match

ACL: 4k
Ternary match



What if you need flexibility?

- Flexibility to:
 - Trade one memory size for another
 - Add a new table
 - Add a new header field
 - Add a different action
- SDN accentuates the need for flexibility
 - Gives programmatic control to control plane, expects to be able to use flexibility
 - OpenFlow designed to exploit flexibility.

What about Alternatives?

Aren't there other ways to get flexibility?

- Software? 100x too slow, expensive
- NPUs? 10x too slow, expensive
- FPGAs? 10x too slow, expensive

What the Authors Set Out To Learn

- How to design a flexible switch chip?
- What does the flexibility cost?

RMT Switch Model

Enables flexibility through?

- Programmable parsing: support arbitrary header fields
- Ability to configure number, topology, width, and depths of match-tables.
- Programmable actions: allow a flexible set of actions (including arbitrary packet modifications).

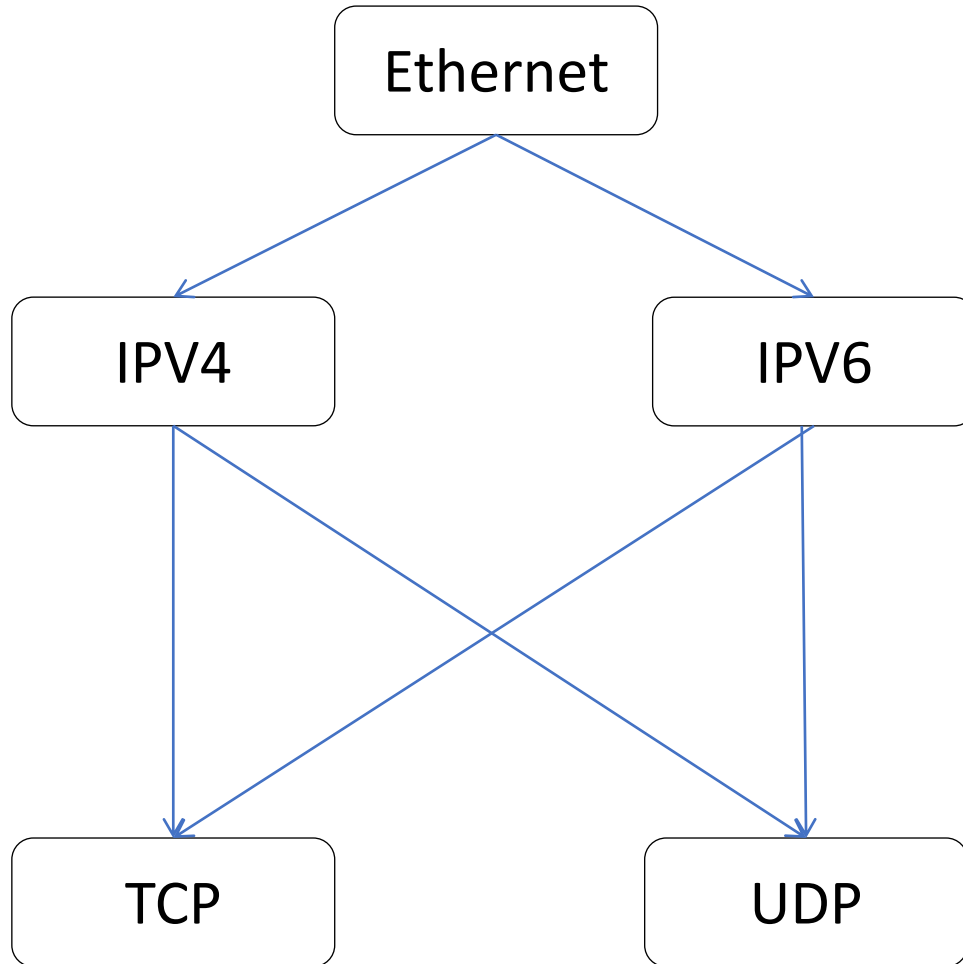
What's Hard about a Flexible Switch Chip?

- Big chip
- High frequency
- Wiring intensive
- Many crossbars
- Lots of TCAM
- Interaction between physical design and architecture

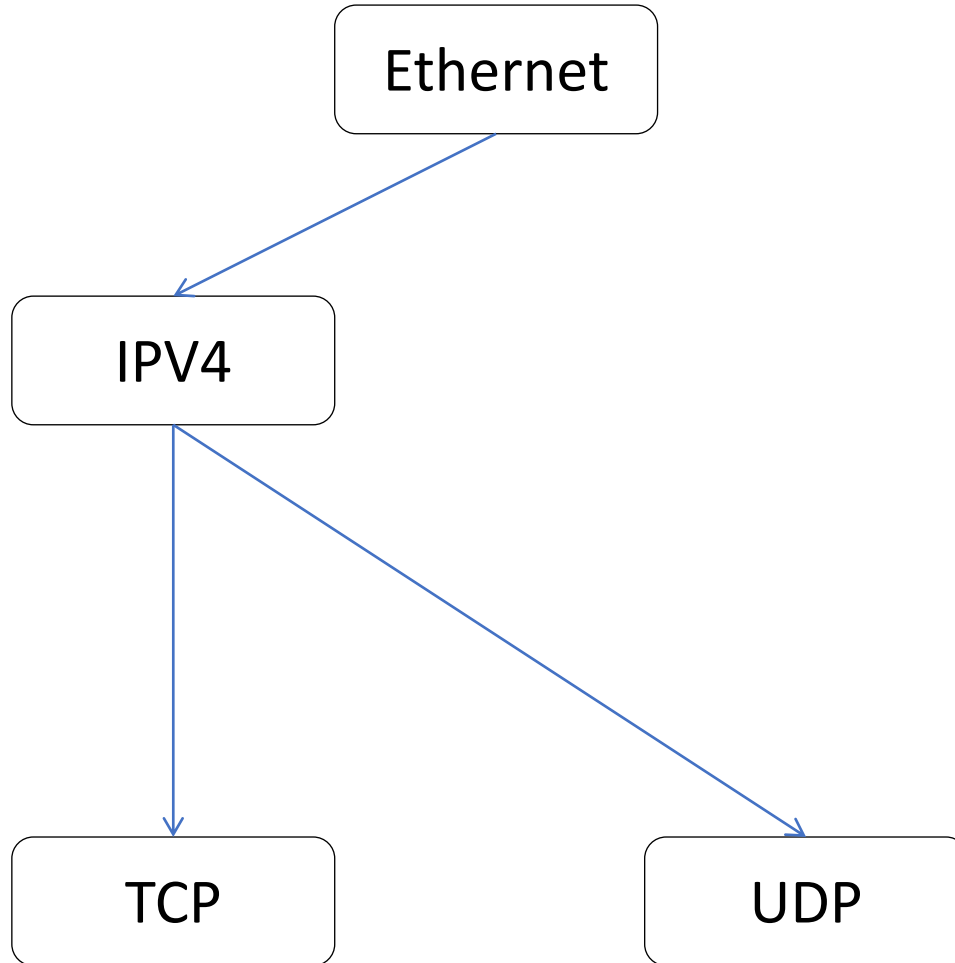
The RMT Abstract Model

- Parse graph
- Table graph

Arbitrary Fields: The Parse Graph



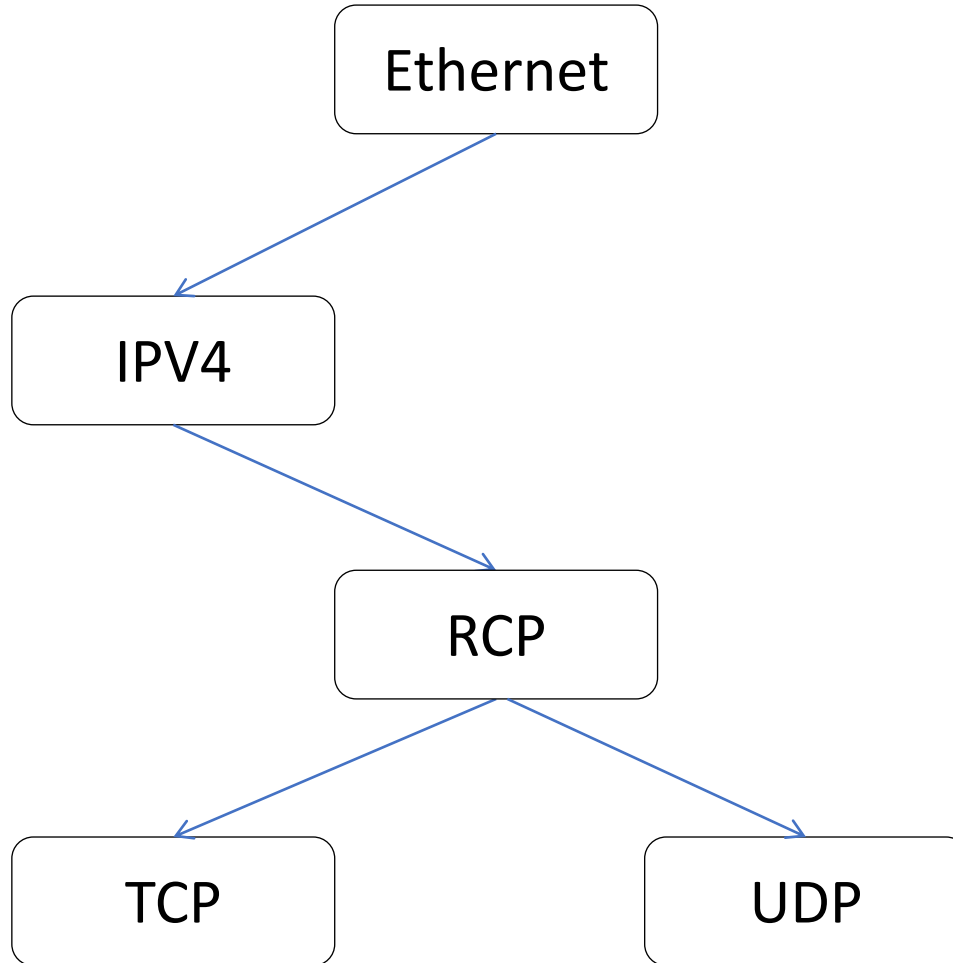
Arbitrary Fields: The Parse Graph



Arbitrary Fields: The Parse Graph

Packet:

Ethernet	IPV4	RCP	TCP
----------	------	-----	-----



Arbitrary Fields: Programmable Parser

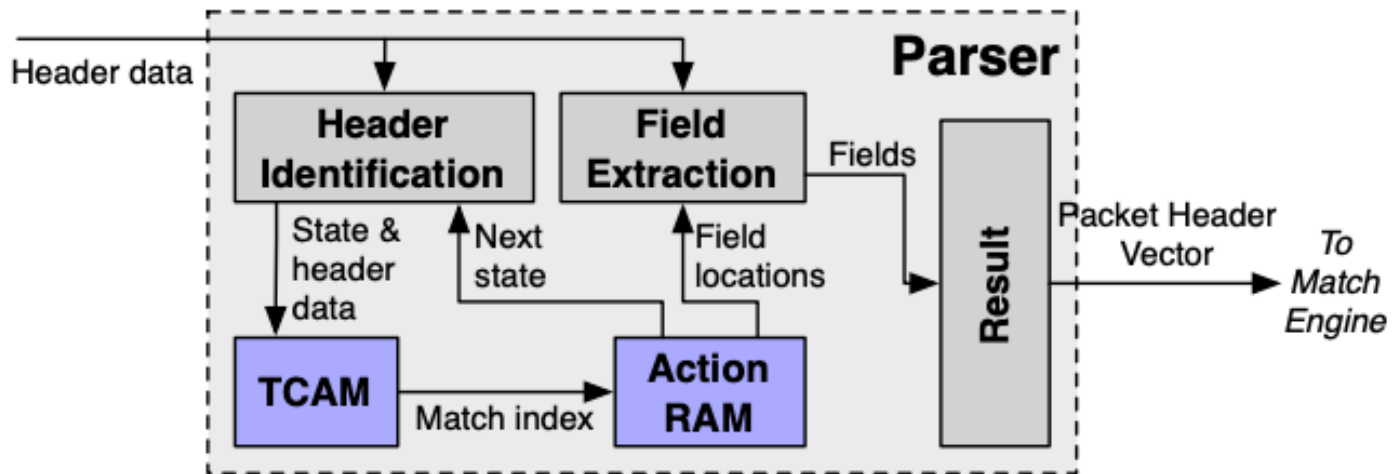
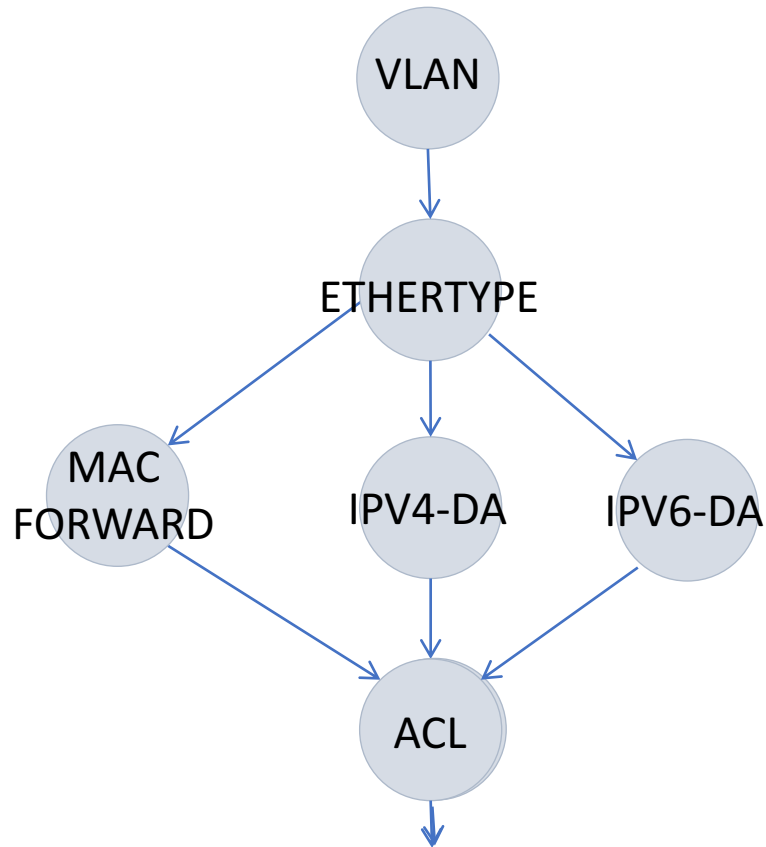
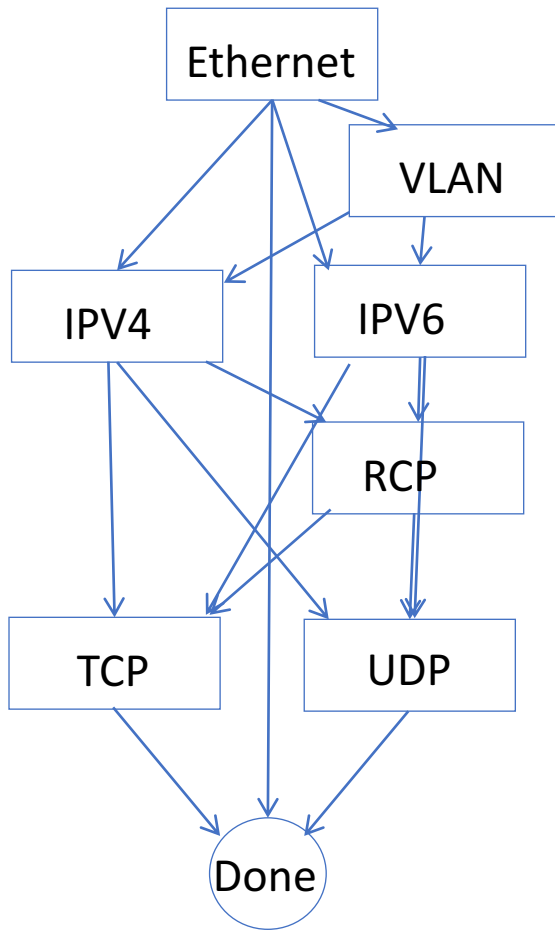


Figure 4: Programmable parser model.

Reconfigurable Match Tables: The Table Graph



Changes to Parse Graph and Table Graph



Parse Graph

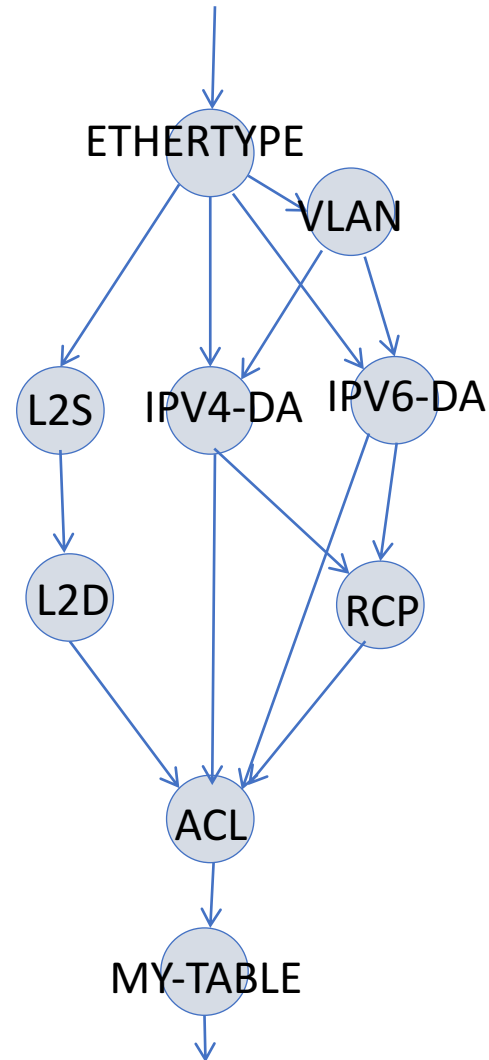
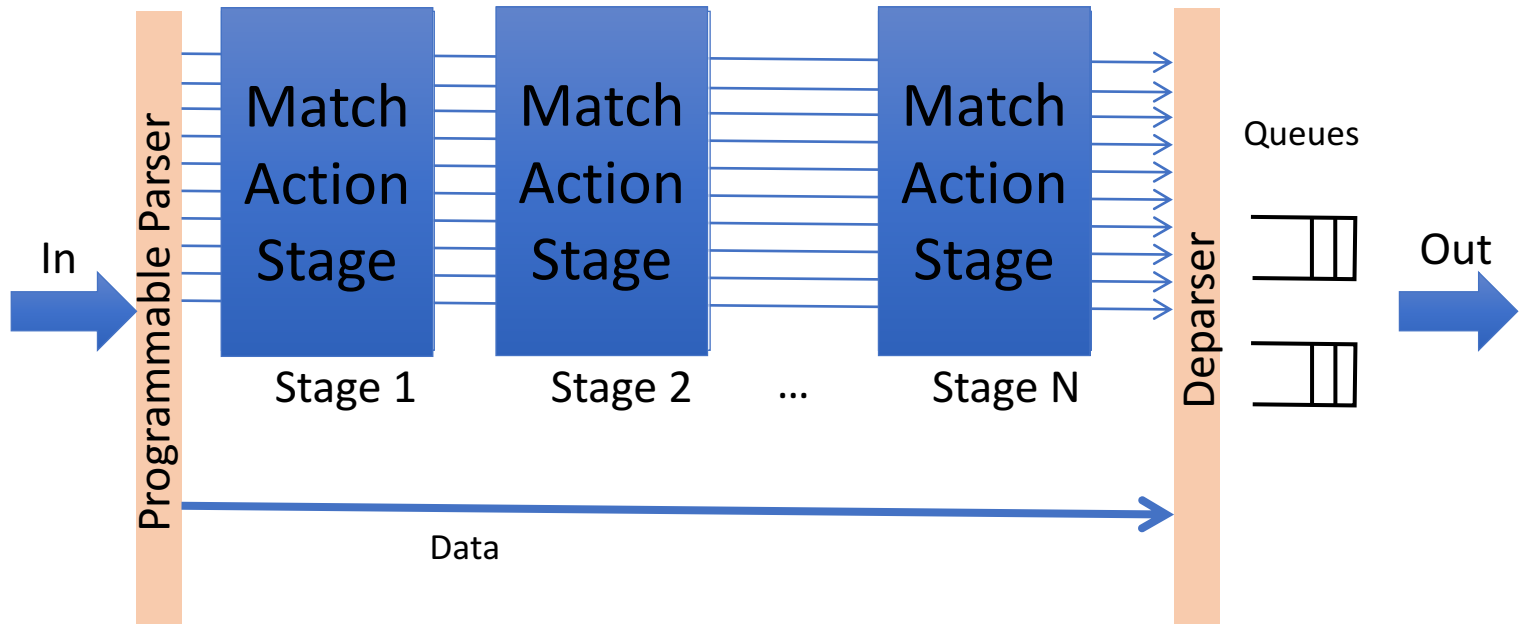


Table Graph

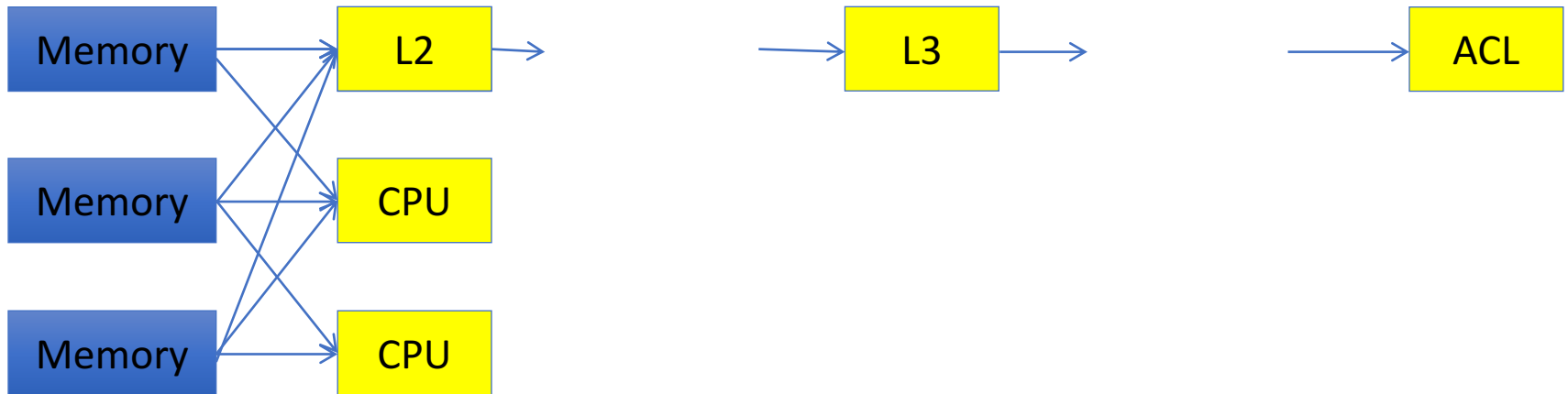
But the Parse Graph and Table Graph
don't show you how to build a switch

Match/Action Forwarding Model

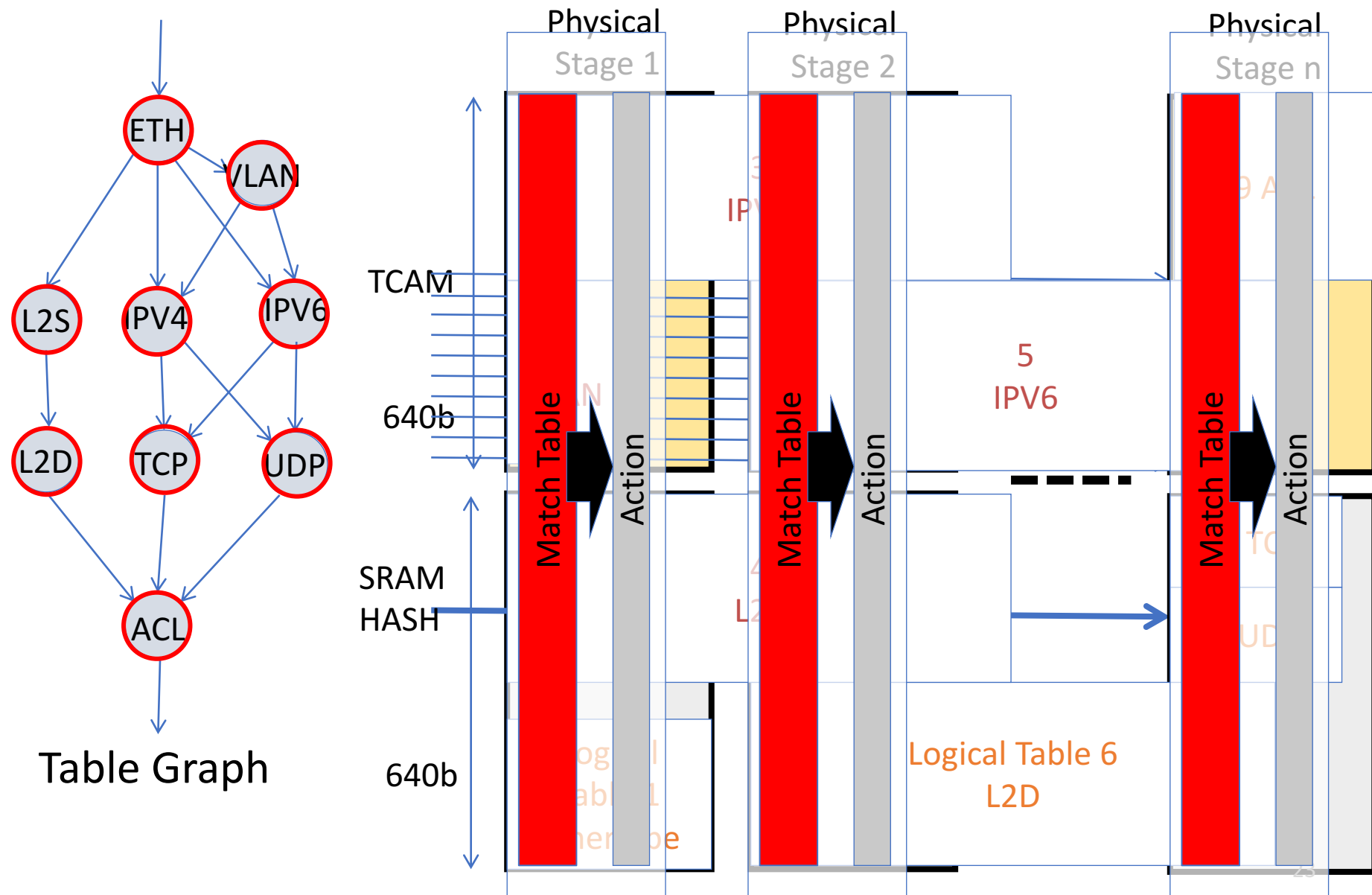


Performance vs Flexibility

- Multiprocessor: memory bottleneck
- Change to pipeline
- Fixed function chips specialize processors
- Flexible switch needs general purpose CPUs



RMT Logical to Physical Table Mapping



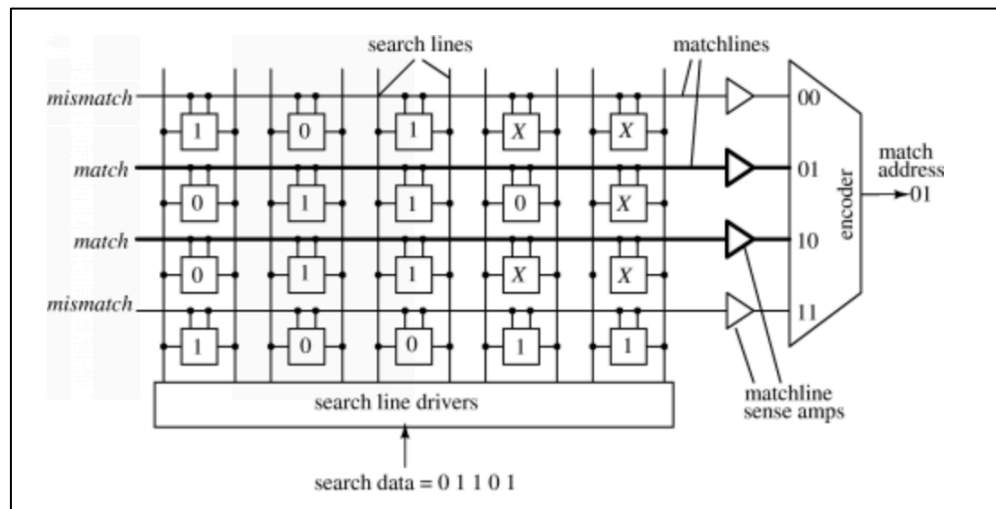
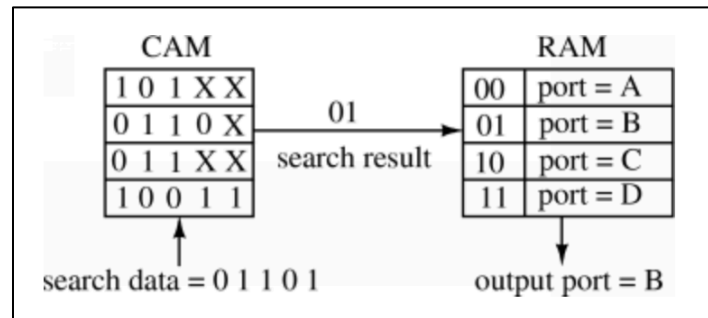
Detour: CAMs and RAMs

- RAM:
 - Looks up the value associated with a memory address.

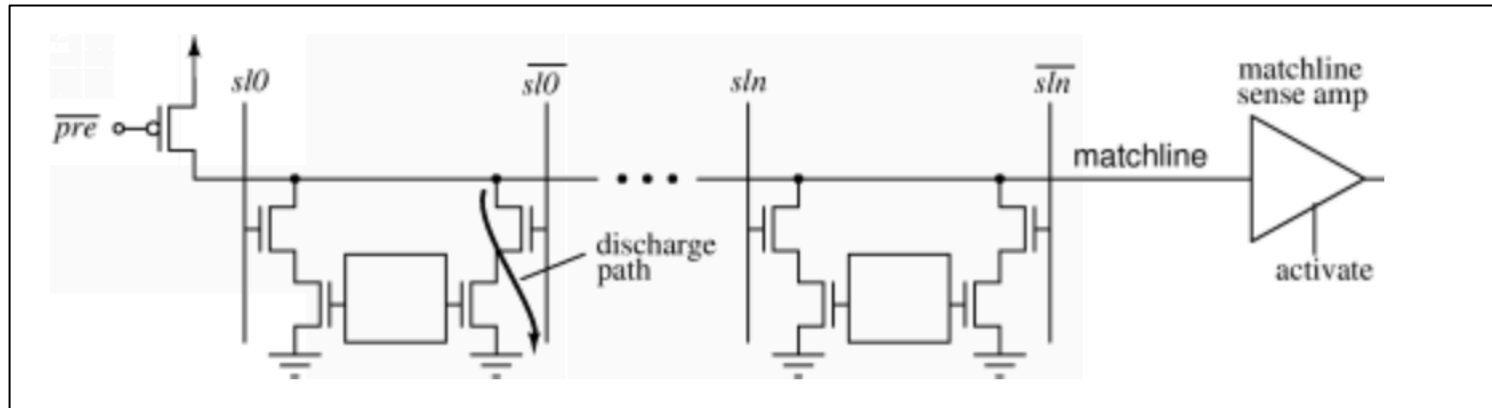
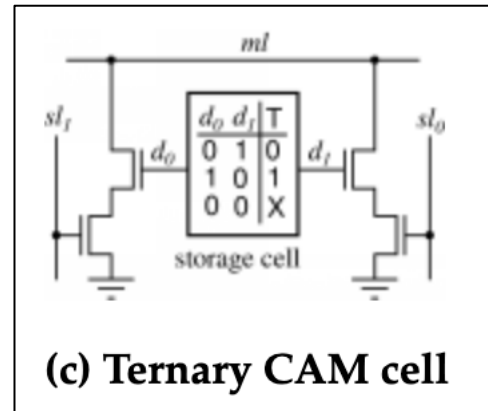
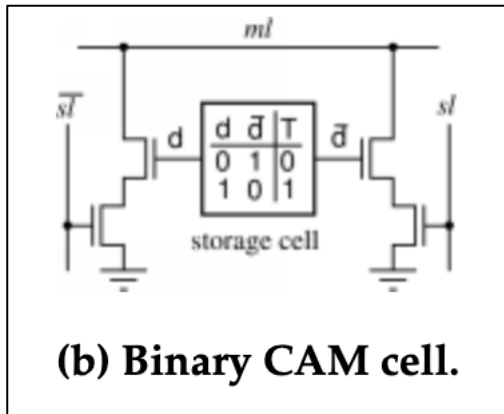
- CAM
 - Looks up memory address of a given value.
 - Two types:
 - Binary CAM: Exact match (matches on 0 or 1)
 - Can be implemented using SRAM.
 - Ternary CAM (TCAM): Allows wildcard (matches on 0, 1, or X).

Detour: CAMs

Line No.	Address (Binary)	Output Port
1	101XX	A
2	0110X	B
3	011XX	C
4	10011	D

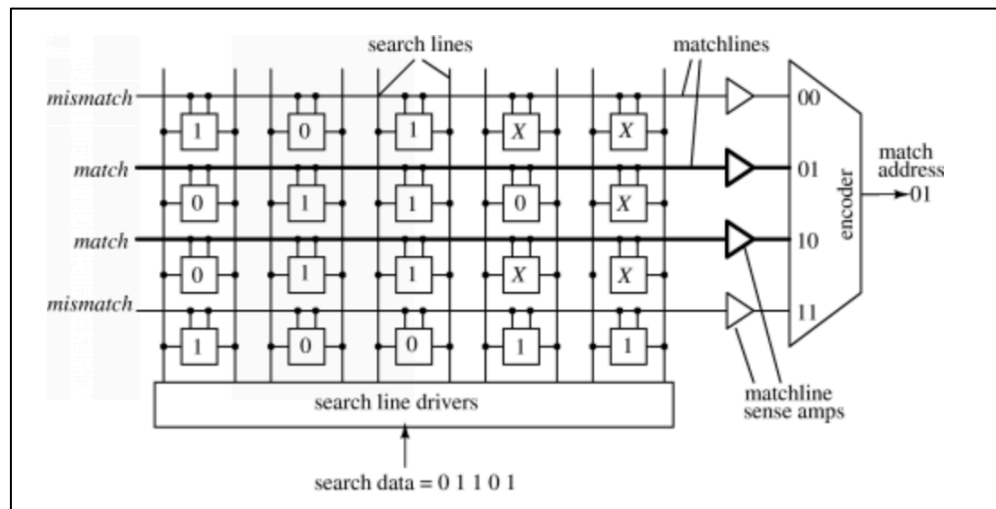
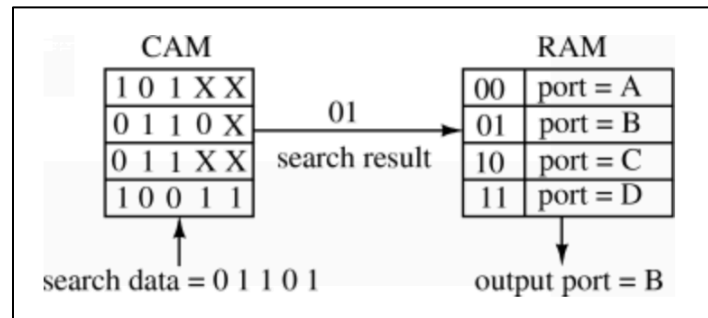


Detour: CAMs

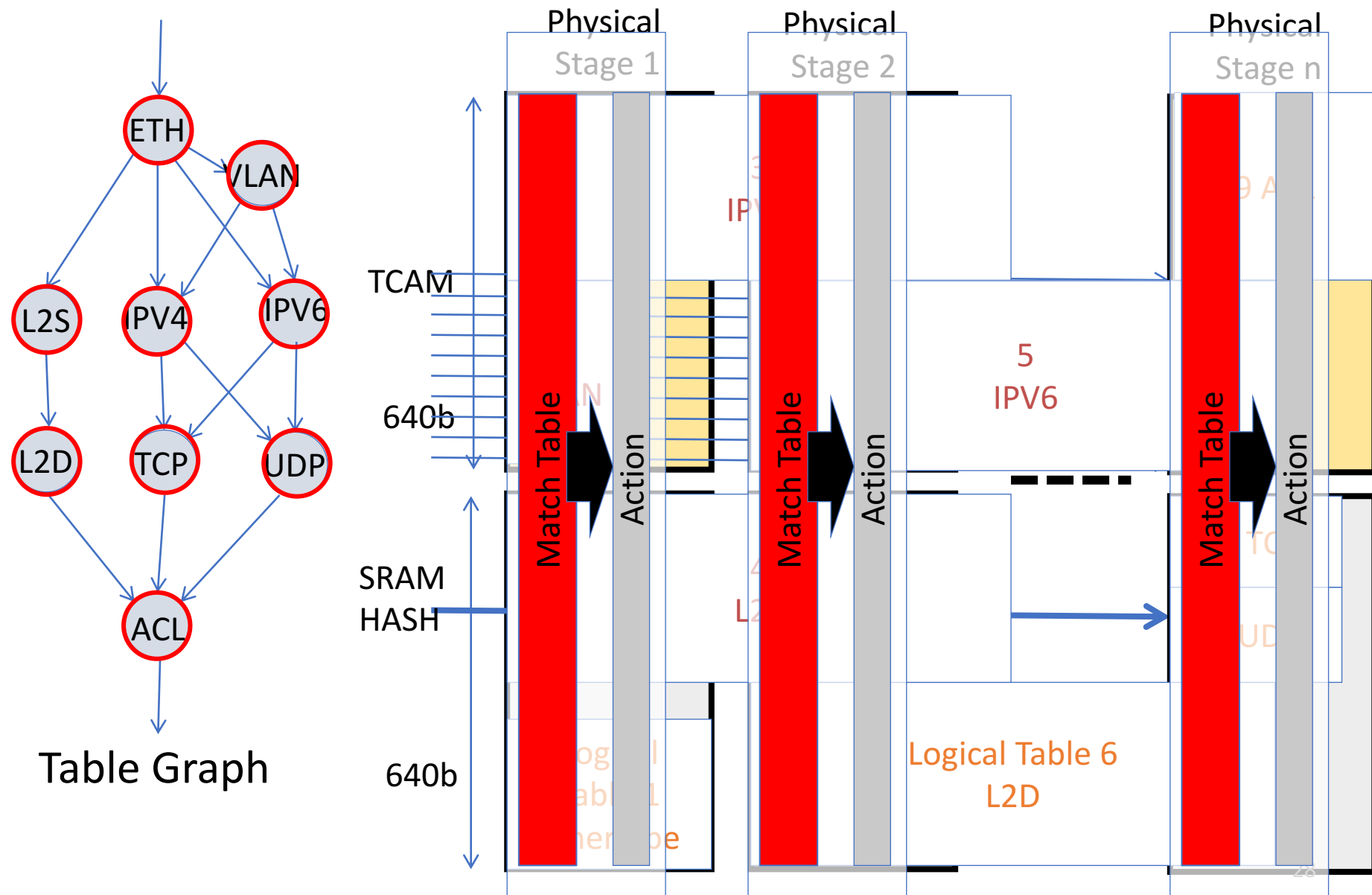


Detour: CAMs

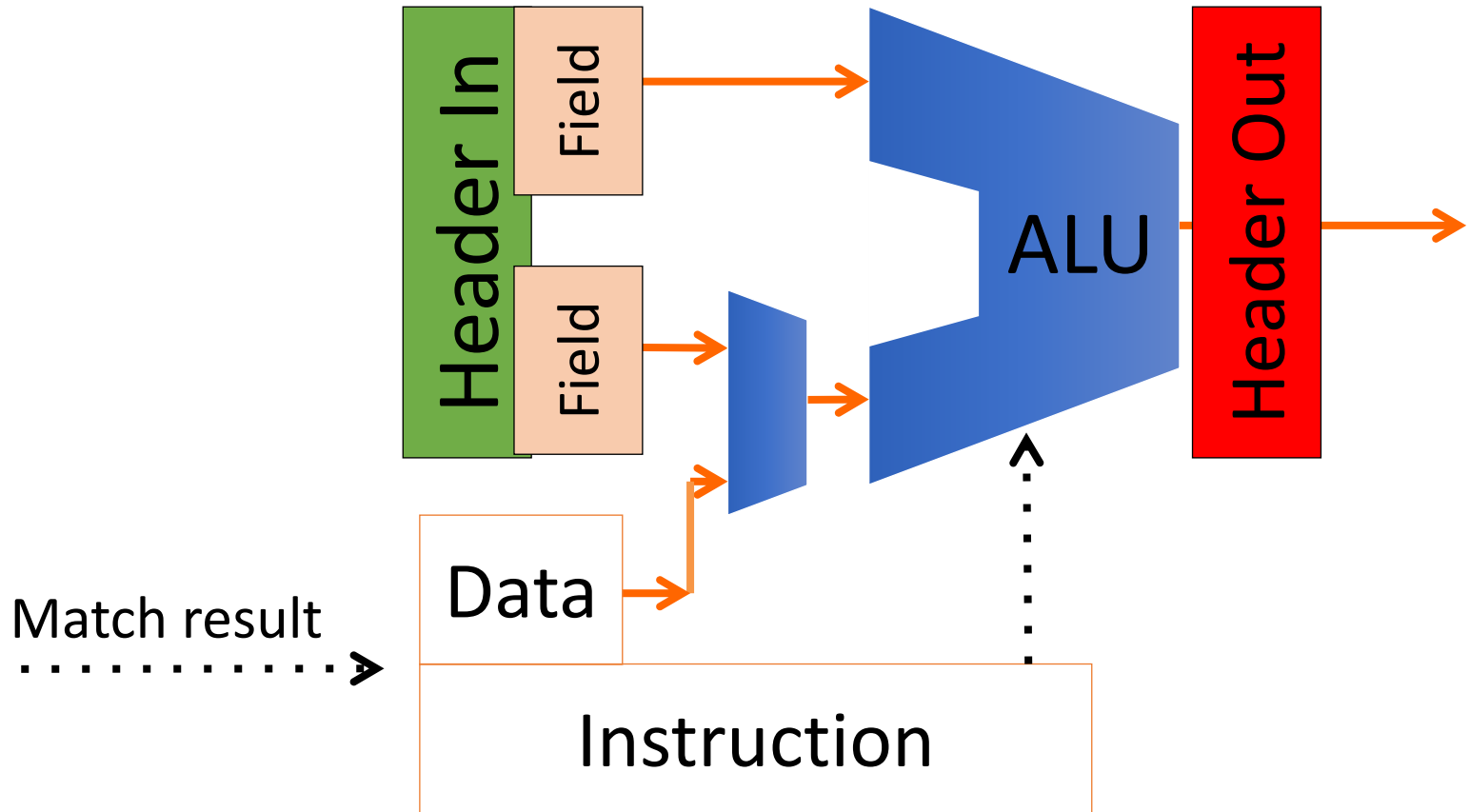
Line No.	Address (Binary)	Output Port
1	101XX	A
2	0110X	B
3	011XX	C
4	10011	D



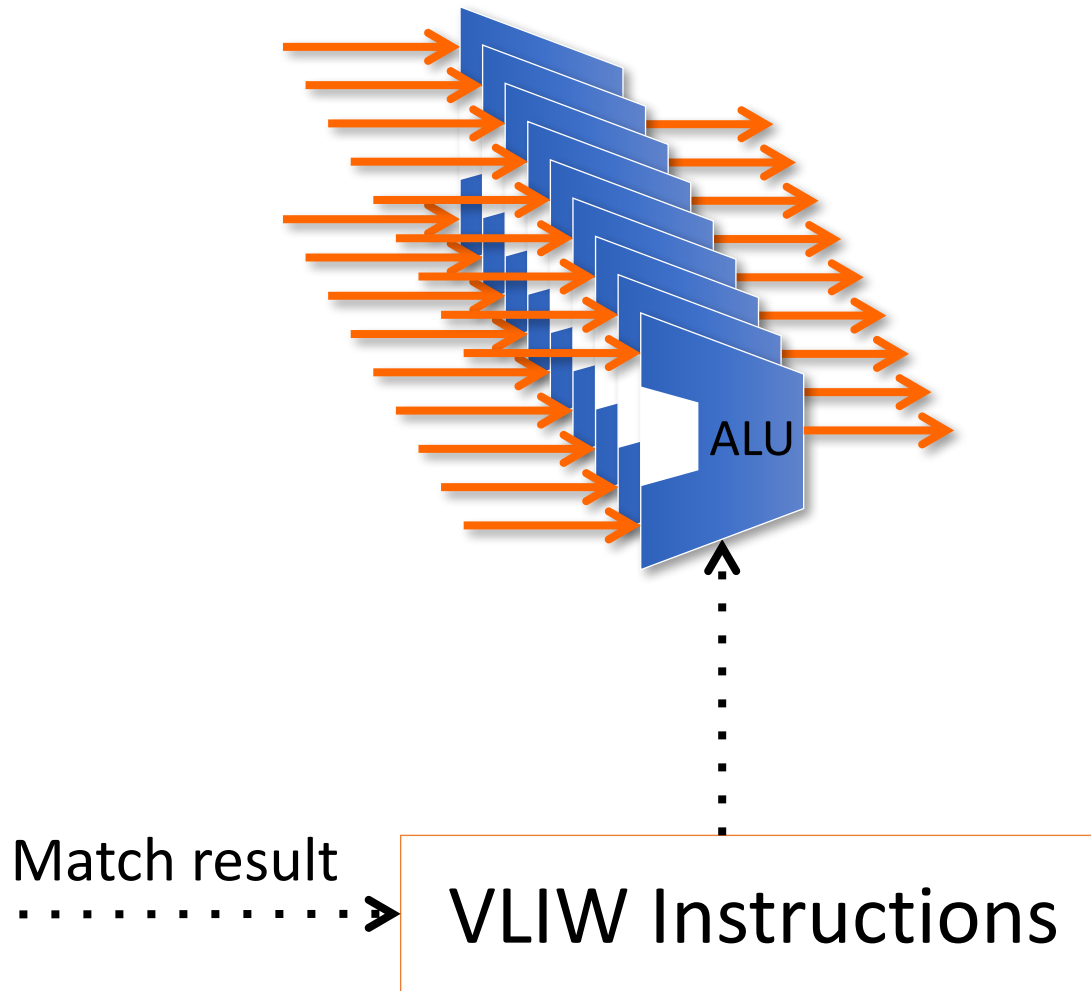
RMT Logical to Physical Table Mapping



Action Processing Model



Modeled as Multiple VLIW CPUs per Stage



RMT Switch Design

- 64 x 10Gb ports
 - 960M packets/second
 - 1GHz pipeline
- Programmable parser
- 32 Match/action stages
- Huge TCAM: 10x current chips
 - 64K TCAM words x 640b
- SRAM hash tables for exact matches
 - 128K words x 640b
- 224 action processors per stage
- All OpenFlow statistics counters

Outline

- Conventional switch chip are inflexible
- SDN demands flexibility...sounds expensive...
- How do I do it: The RMT switch model
- Flexibility costs less than 15%

Cost of Configurability: Comparison with Conventional Switch

- Many functions identical: I/O, data buffer, queueing...
- Make extra functions optional: statistics
- Memory dominates area
 - Compare memory area/bit and bit count
- RMT must use memory bits efficiently to compete on cost
- Techniques for flexibility
 - Match stage unit RAM configurability
 - Ingress/egress resource sharing
 - Allows multiple tables per stage
 - Match memory overhead reduction and multi-word packing

Chip Comparison with Fixed Function Switches

Area

Section	Area % of chip	Extra Cost
→ IO, buffer, queue, CPU, etc	37%	0.0%
→ Match memory & logic	54.3%	8.0%
→ VLIW action engine	7.4%	5.5%
Parser + deparser	1.3%	0.7%
Total extra area cost		14.2%

Power

Section	Power % of chip	Extra Cost
→ I/O	26.0%	0.0%
→ Memory leakage	43.7%	4.0%
Logic leakage	7.3%	2.5%
RAM active	2.7%	0.4%
TCAM active	3.5%	0.0%
→ Logic active	16.8%	5.5%
Total extra power cost		12.4%

Conclusion

- How do we design a flexible chip?
 - The RMT switch model
 - Bring processing close to the memories:
 - pipeline of many stages
 - Bring the processing to the wires:
 - 224 action CPUs per stage
- How much does it cost?
 - 15%
- Lots of the details how we designed this in 28nm CMOS are in the paper

Limitations on Flexibility

- Your thoughts!

Since 2013....

- RMT switch has been commercialized
 - Barefoot Tofino
 - 6.5Tb/s
- Adoption of these switches?

Your opinions

- Pros
 - Proposes RMT as a more flexible alternative to SMT and MMT.
 - Shows viability of a flexible design.
 - Evaluates cost and power requirements, shows they are not significantly high.
 - (In contrast to RouteBricks)
 - Flexible memory allocation mechanism is innovative and efficient.

Your opinions

- Cons
 - Programmability limitations not discussed? Is it Turing-complete?
 - What are the scalability bottlenecks?
 - Why $N=32$?
 - Conflates memory allocation with match-action processing.
 - No programmability interface.
 - How are low-level configurations generated?
 - No actual hardware
 - Security?

Your opinions

- Ideas
 - A compiler for RMT
 - What can RMT's programmability enable?
 - Extending the level of programmability / lifting restrictions.