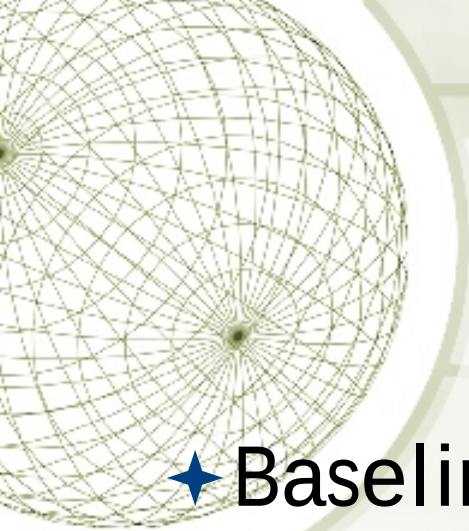
DPICO: A High-Speed Deep Packet Inspection Engine Using Compact Finite Automata

Chris Hayes

Rensselaer Polytechnic Institute (formerly UMASS Lowell)

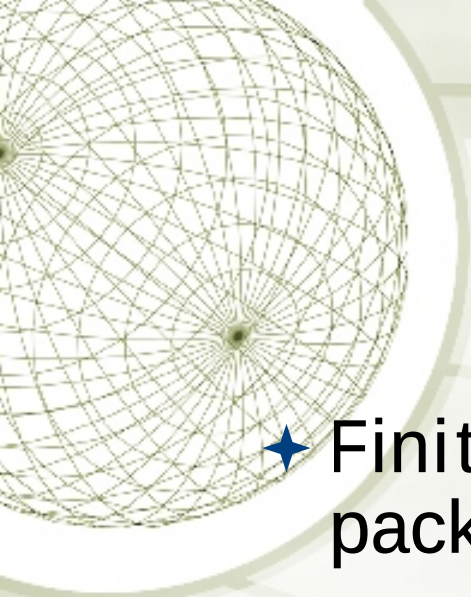
Yan Luo

University of Massachusetts Lowell



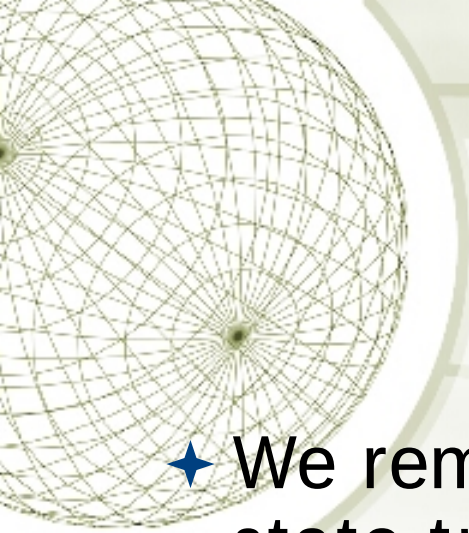
Agenda

- Baseline Design
- Design with Compression
  - Content Addressable Memory
  - Interleaved Memory Banks
  - Data Packing
- Memory Savings
- Results



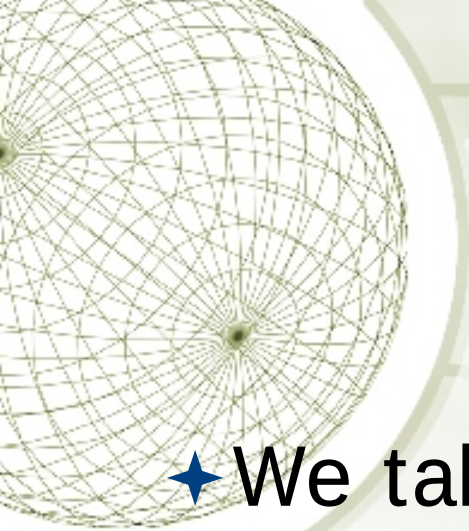
Baseline Design and Motivation

- Finite Automata are the basis for many packet filtering techniques.
- We propose a technique for implementing packet filtering in hardware.
- Standard Moore Machine Next-State memory architecture.
- Advantages:
  - Speed
- Disadvantages:
  - Memory utilization - redundant information can be present based on a given finite automaton. **This is what we seek to reduce.**



How to Improve the Utilization

- We remove repeated information in the state transition table.
  - Many transitions for a state may have the same next state pointer.
  - We want to combine these into a single transition. We accomplish this by creating two types of transitions:
    - Labeled transition - followed if the label matches the input character.
    - Default transition - followed if no label matches input character.
  - The most frequently repeated next state pointer in each state becomes the default transition pointer.



How to realize the compression

- We take advantage of three technologies:
  - Content-Addressable Memory
  - Interleaved Memory Banks
  - Data Packing

Content Addressable Memory

- Drawback to the baseline design is the fixed state size.
- Adding default and labeled transitions give a mechanism to compress the state.
- Use the CAM to provide a search mechanism to find the next-state transition based on the input character.
  - Each state has its own CAM, since each state will require its own associative lookup.

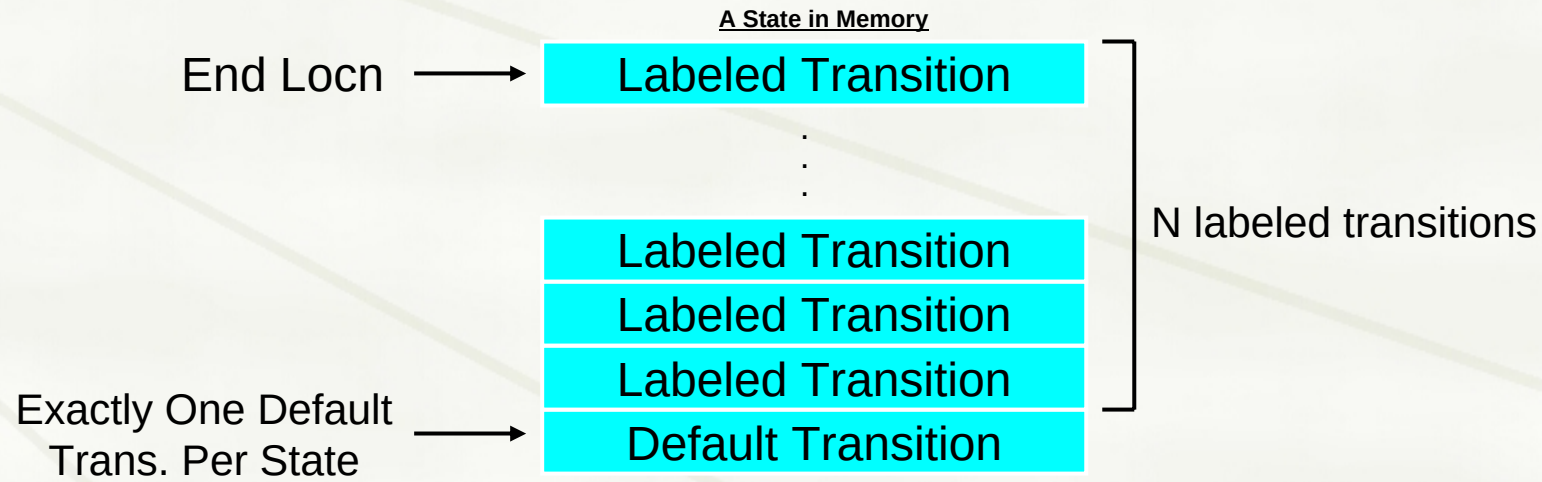
Content Addressable Memory (2)

Labeled Transition

Label	Next State Pointer

Default Transition

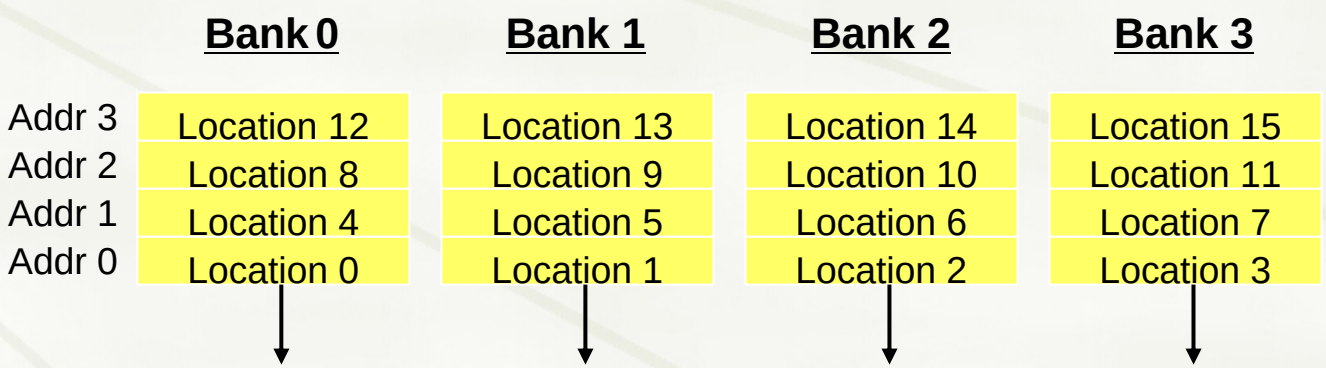
End Locn	Match ID	Next State Pointer



Issue: Need to Search through each labeled transition to resolve next state. (Could take Many Clocks)

Interleaved Memory

- FPGAs can have hundreds of banks of memory.
- Each bank can be read in parallel.
- Read/Write bandwidth increased by a factor of n, where n is the number of banks.
- Example with four banks:



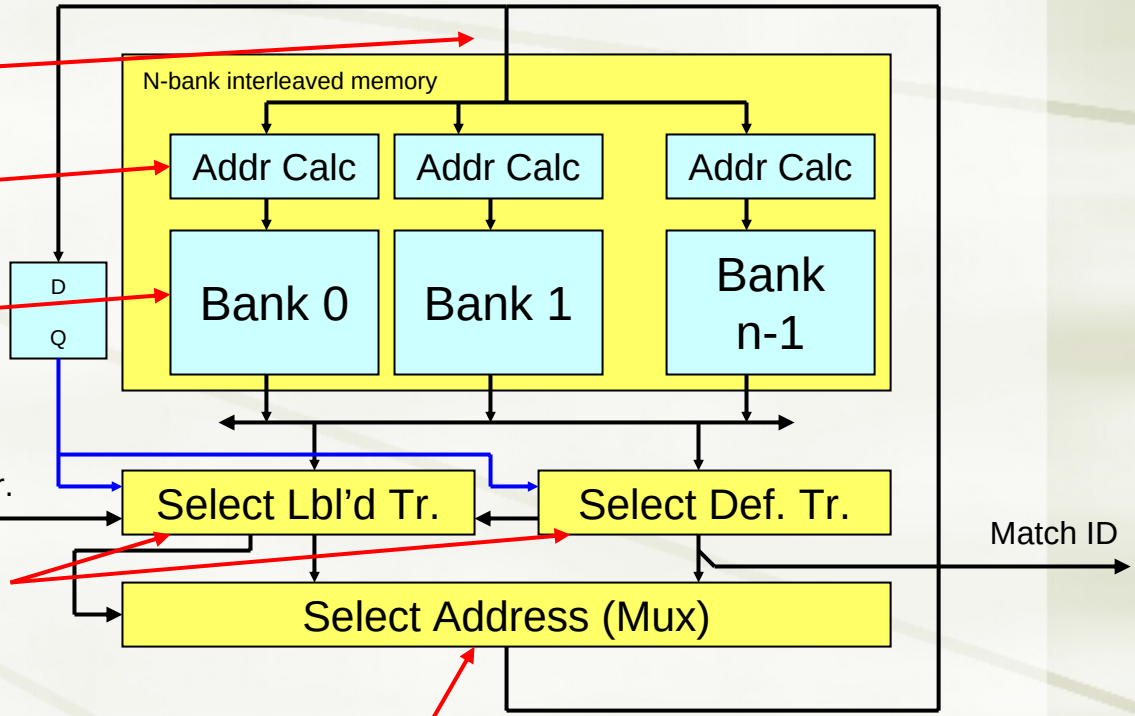
Note: By Controlling the Read address to each bank, we can read any 4 continuous locations simultaneously. This allows us to evaluate multiple transitions in a single clock cycle

The Design

The current state address is input to the interleaved memory

The individual address is calculated for each RAM

The default and labeled transition info are read from the RAM



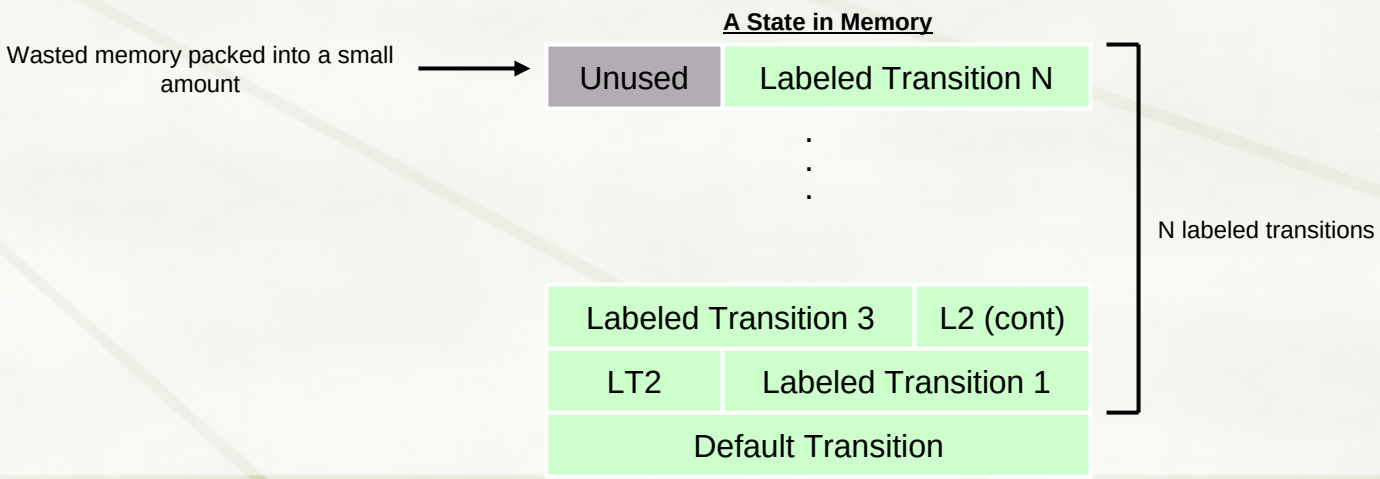
We select the labeled transition that matches the input character. Simultaneously, we read the default transition information and output the Match ID.

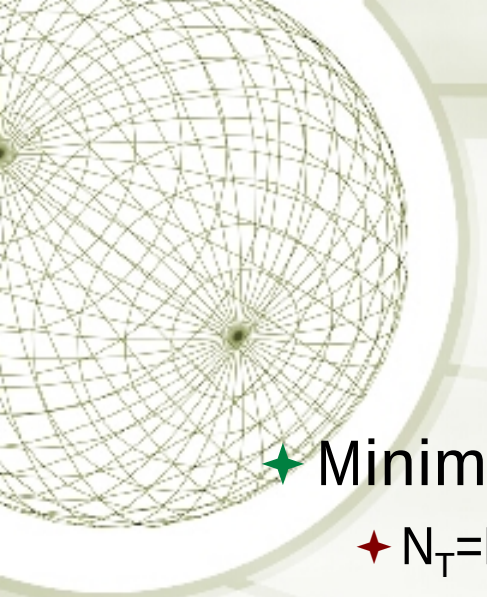
Finally, we select the next state pointer from the labeled transition logic or the default transition logic depending on whether the labeled transition logic was successful.

Reduces Storage while Keeping a Constant Transition Time

Packing

- Labeled transitions are likely to be smaller than Default Transitions.
- We can pack the labeled transitions into memory so that much less memory is wasted.
- Packing reduces the number of banks needed to account for the largest number of transitions per state.

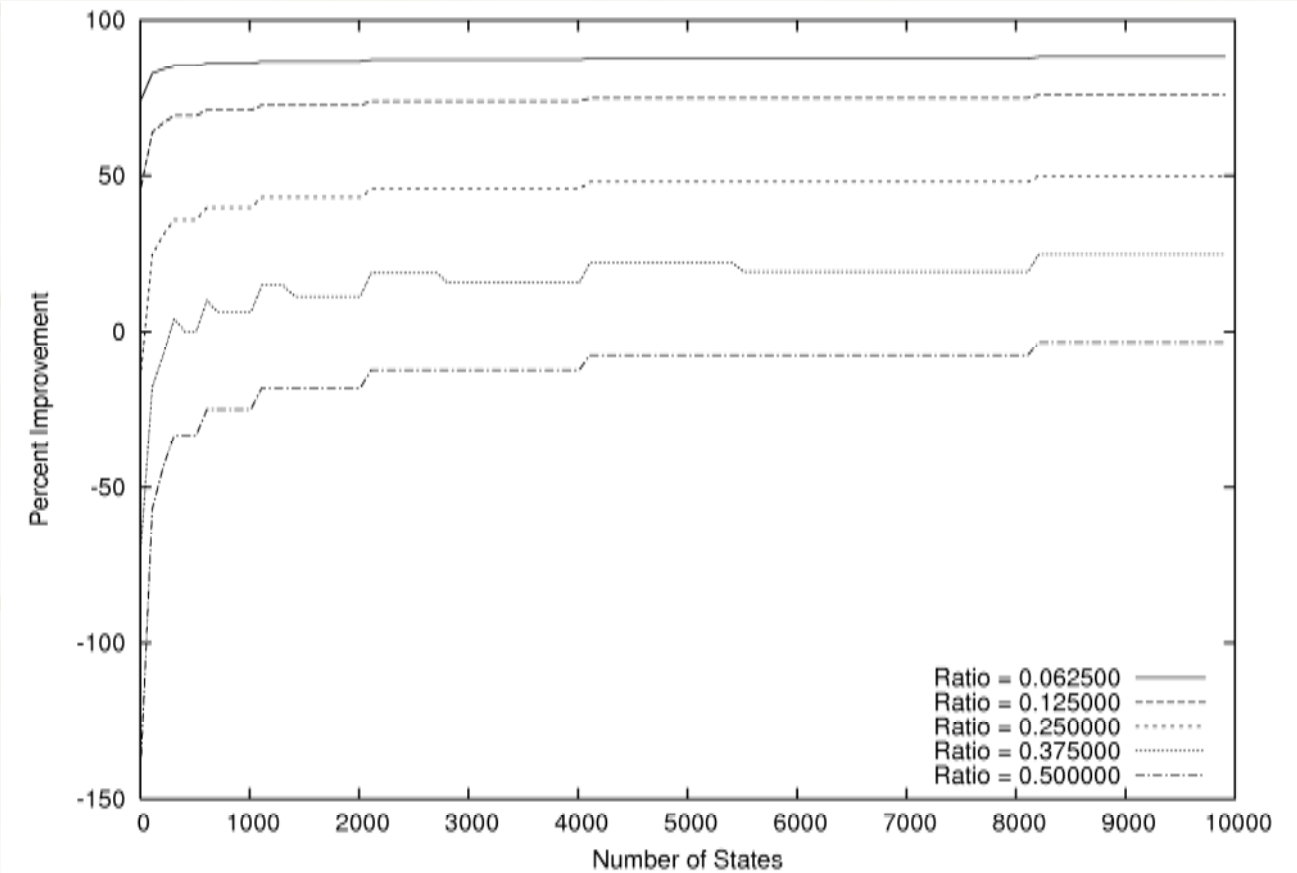




Packing (2)

- Minimum Size = $N_T(8+\lg(N_T))+N_S(8+\lg(N_M)+\lg(N_T))$
  - N_T=Number of Transitions
  - N_S=Number of States
  - N_M=Number of Match IDs
- We can evaluate the space savings potential by finding the ratio of average transitions per state to the number of possible transitions.
  - Transition Ratio = #AvgTrans/256
  - As seen on the next slide, finite automations with transition ratios of less than 0.5 are fit for this method.

Potential Memory Savings



Results

Results from Conv. Program

Ruleset	# of Rules	DFA Baseline Memory Size (bits)	DPICO Unpacked D^2FA Memory Size (bits)	DPICO Minimum D^2FA Memory Size (bits)	Trans. Ratio (*r*)	% Savings
imap	46	16,923,528	715,139	571,171	0.018	96.5%
ftp	76	11,723,205	534,688	418,552	0.017	96.4%
netbios	633	2,198,208	66,556	54,388	0.011	97.5%
nntp	13	8,008,479	330,339	268,809	0.017	96.6%
exploit	122	56,596,540	7,355,320	5,001,178	0.046	91.2%

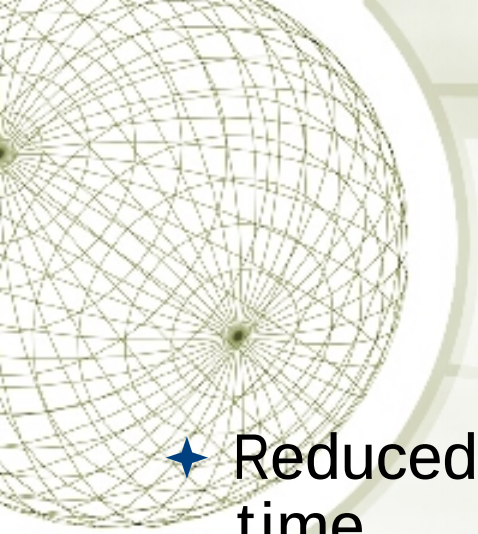
Size and Speed Results

# of Banks	LUT	REG	f_{max} (MHz)	BW_{max} (Mbps)
2	114	129	144.9	1159.2
4	183	204	122.3	978.4
8	320	352	106.9	855.2
16	698	642	98.7	789.6
32	1672	1252	84.8	678.4
64	3541	2346	78.9	631.2
128	7659	4810	74.0	592.0
256	16052	9563	68.1	544.8

- Baseline Design
  - 267.7 MHz (2141.6 Mbps)
  - No LUT or REG
- Non-Pipelined design
- Implemented in a Xilinx Virtex 4 SX35

Size and Speed Results (2)

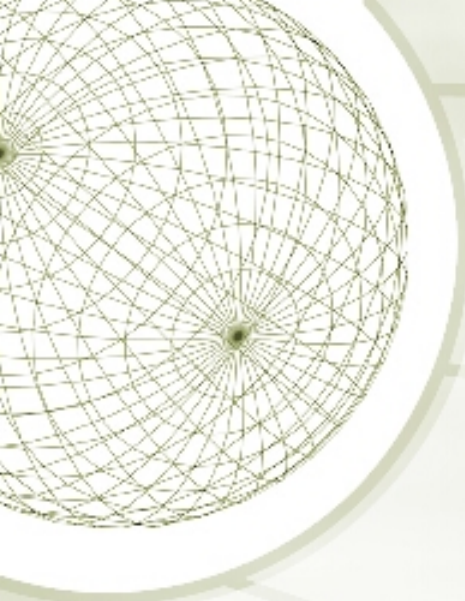
QuickTime™ and a
decompressor
are needed to see this picture.



Conclusion

- Reduced storage while keeping a constant transition time.
- Great space solution if the ratio of avg. transitions to number of possible transitions << 0.5.
- Minimizing the the maximum number of transitions for a state will increase the speed of the design.
- Pipelined solutions can run at 250 MHz in contemporary parts (e.g. Xilinx Virtex 4) by time multiplexing multiple data streams into one engine.
  - (250 MHz equates to 2Gbps input data rate)
- Design is scalable using tradeoff between memory and speed (up to 17Gbps).

Questions?



Backup Slides

Results based on Kumar et al.

Ruleset	Projected Baseline Memory Size (bits)	Projected DPICO Minimum Memory Size (bits)	Transition Ratio (*r*)	% Savings
Cisco590	68,195,050	44,757,032	0.34	34.4%
Cisco103	80,979,350	36,008,373	0.23	55.5%
Cisco7	14,190,060	8,438,994	0.29	40.5%
Linux56	50,091,270	16,629,394	0.17	66.8%
Linux10	46,654,764	26,996,384	0.29	42.1%
Snort10	171,990,900	15,295,048	0.05	91.1%
Bro648	20,749,008	3,936,212	0.09	81.0%

Ruleset	Projected DFA Baseline Memory Size (bits)	Projected DPICO Minimum D^2FA Memory Size (bits)	Transition Ratio (*r*)	% Savings
Cisco590	68,195,050	1071299	0.008	98.4%
Cisco103	80,979,350	36,008,373	0.010	98.2%
Cisco7	14,190,060	8,438,994	0.026	95.3%
Linux56	50,091,270	16,629,394	0.016	97.0%
Linux10	46,654,764	26,996,384	0.086	83.3%
Snort10	171,990,900	15,295,048	0.016	97.3%
Bro648	20,749,008	3,936,212	0.004	99.0%