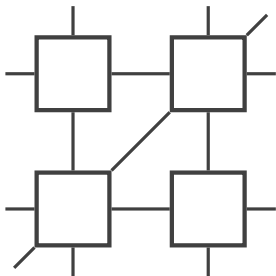


# **Order-exploiting on-chip routing table minimization for a multicast supercomputer network**

.....

**Andrew Mundy, Jonathan Heathcote, Jim Garside**

## SpiNNaker (Spiking Neural Network Architecture)



- 18 cores per node
  - 64 KiB DTCM
  - 32 KiB ITCM
- 1 router per node
  - **1024 entry prioritized TCAM**
  - *Default routing*
- Full machine
  - 57 600 nodes
  - 1M cores

## SpINNaker Router Behavior

Key-Mask	Route
0000	NE N
X111	S
1XXX	3 4

*Default-routed* packets travel in a straight line

**What if tables are too big?**

## Benchmarks

Locally-connected

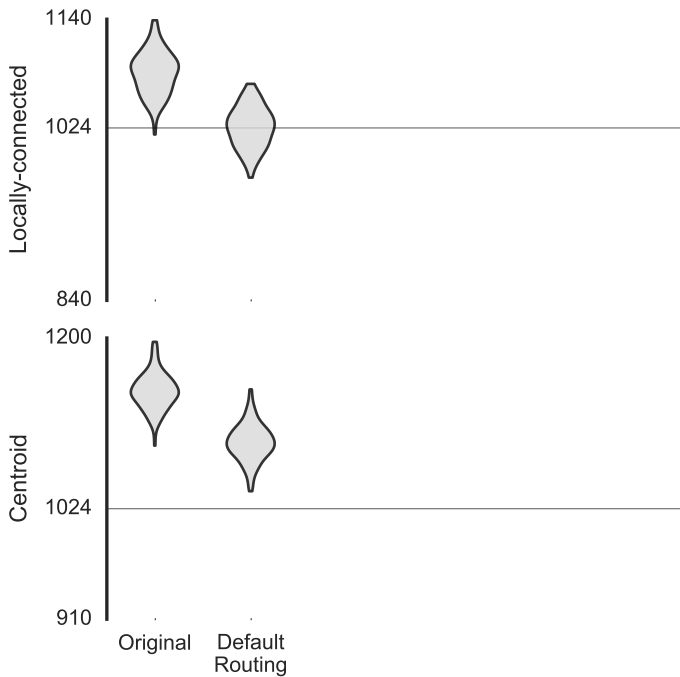


Centroid



Also –

- Broken links



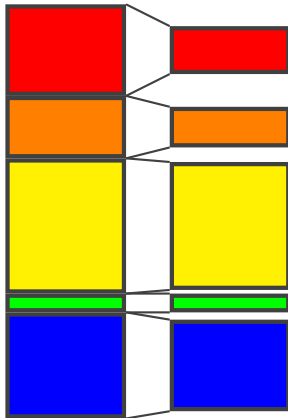
## Minimization with Espresso

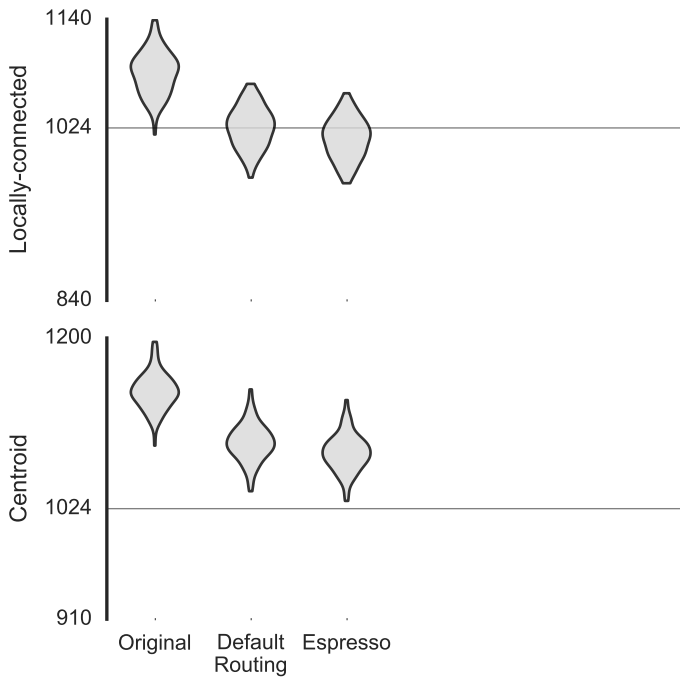
- Break into subtables with the same route
- Minimize each subtable exactly

### Exact?

0000 and 0001  $\rightarrow$  000X

0001 and 0010  $\rightarrow$  ~~00XX~~





## Minimization with Espresso

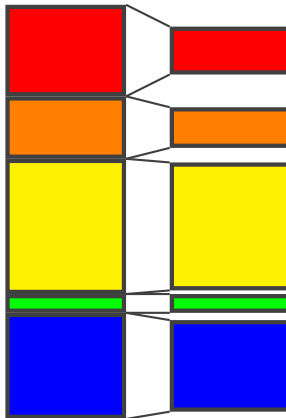
- Good when coarse routing decisions can be made with few bits
  - Source-based routing doesn't allow this
- Ignores the prioritization of the TCAM



## Order-exploiting minimization

- Break into subtables with the same route
- **Sort in order of subtable size**
- Minimize each subtable **to avoid collisions with lower subtables**

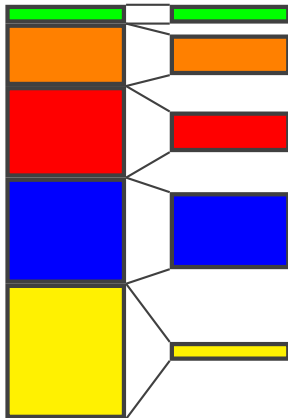
0001 and 0010  $\rightarrow$  00XX  
*allowed* iff. no lower table  
contains 0000 or 0011

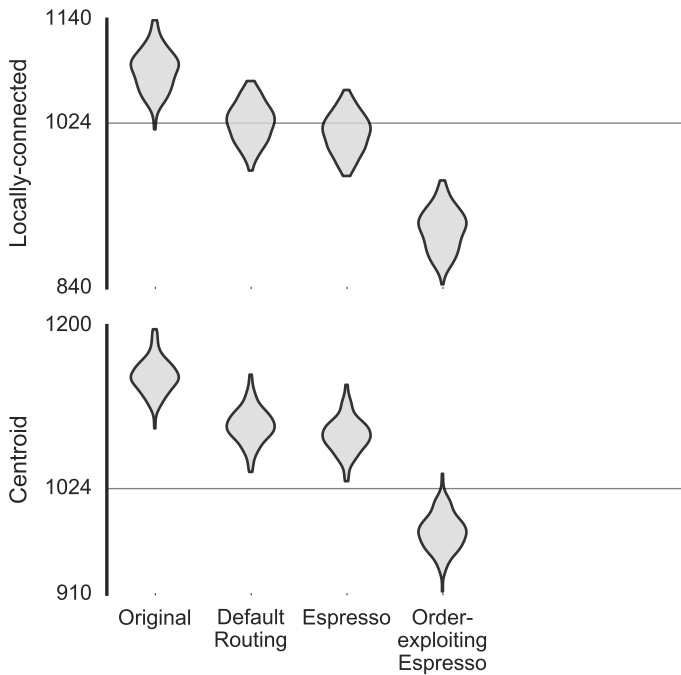


## Order-exploiting minimization

- Break into subtables with the same route
- **Sort in order of subtable size**
- Minimize each subtable **to avoid collisions with lower subtables**

0001 and 0010  $\rightarrow$  00XX  
*allowed* iff. no lower table  
contains 0000 or 0011





# On-chip routing table minimization

Espresso – 6.23 s per table

# On-chip routing table minimization

Espresso – 6.23 s per table  $\times$  57 600 nodes...

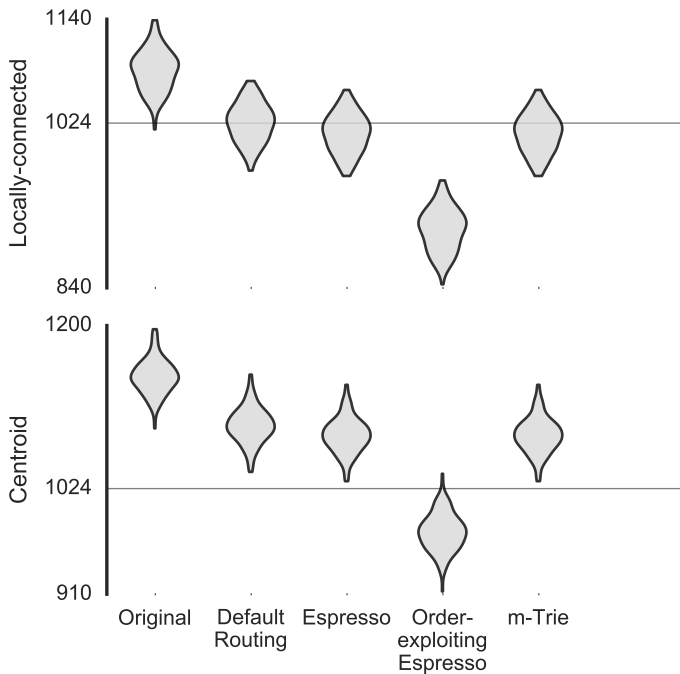
## On-chip routing table minimization

Espresso – 6.23 s per table  $\times$  57 600 nodes...4 days

- Problem is trivially parallel – use SpiNNaker
- **BUT** Espresso *is* big
- **AND** needs a lot of memory

Other people have looked at this:

- R. Lysecky and F. Vahid, “On-chip logic minimization,” in *Design Automation Conference, 2003. Proceedings*, Jun. 2003, pp. 334–337
- S. Ahmad and R. Mahapatra, “An efficient approach to on-chip logic minimization,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 15, no. 9, pp. 1040–1050, Sep. 2007, ISSN: 1063-8210 –  
**m-Trie**



## On-chip routing table minimization

Same problems as before –

- Good when coarse routing decisions can be made with few bits
- Ignores the prioritization of the TCAM

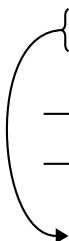
### **Challenge**

- Simple minimizer (fit in ITCM)
- Small data structures (fit in DTCM)
- Exploit the ordering of the TCAM (minimize well)



## Ordered-Covering


Key-Mask	Route	Aliases
1011	NE S	1011
0100	NE S	0100
1101	SW 2	1101
1110	SW 2	1110
1101	SW 2	1101
1110	SW 2	1110
XXXX	NE S	1011 0100



- Sort entries in ascending number of Xs
- Annotate entries with keys they are expected to match
- Greedily merge entries with equivalent routes
  - Subject to two rules...

## Up-check rule

No entry in the *merge* may become *covered* by another entry.

0011	E S	0011				
1100	E S	1100				
00XX	N	...		00XX	N	...
				XXXX	E S	0011 1100

e.g., 0011 becomes covered by 00XX.

## Down-check rule

No *aliased entry* below the merge may become covered.

1101	SW 2	1101		11XX	SW 2	...
1110	SW 2	1110		XXXX	NE S	1100 ...
XXXX	NE S	1100 ...				

e.g., 1100 becomes covered by 11XX.

# Algorithm

- While table is larger than desired
  - Get the largest valid merge
  - If the merge is empty, break
  - Otherwise apply the merge

Most potential merges will break the *up-* or *down-check* rules, so...

## Resolving the *up-check*

Remove from the merge any entry which would become *covered* through being merged.

0000	N NE	0000		
1000	N NE	1000		
1110	N NE	1110		
00XX	S	...		

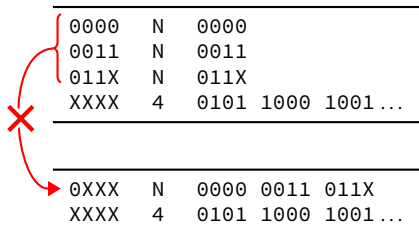
00XX	S	...
XXX0	N NE	...

## Resolving the *up-check*

Remove from the merge any entry which would become *covered* through being merged.

0000	N NE	0000		0000	N NE	0000
1000	N NE	1000	}			
1110	N NE	1110				
00XX	S	...		00XX	S	...
			→	1XX0	N NE	1000 1110

## Resolving the *down-check*



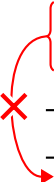
0000	N	0000			
0011	N	0011			
011X	N	011X			
XXXX	4	0101	1000	1001	...
0XXX	N	0000	0011	011X	
XXXX	4	0101	1000	1001	...

Convert an X to either 0 or 1...

0XXX covers 0101 – try to turn 0XXX into:

0XX0, 0X1X or 00XX

## Resolving the *down-check*



0000	N	0000
0011	N	0011
011X	N	011X
XXXX	4	0101 1000 1001...

0XXX	N	0000 0011 011X
XXXX	4	0101 1000 1001...

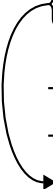
$\{0000, \cancel{0011}, \cancel{011X}\} \rightarrow 000\underline{0}$

$\{\cancel{0000}, 0011, 011X\} \rightarrow 0X\underline{1}X$

$\{0000, 0011, \cancel{011X}\} \rightarrow 00\underline{X}X$



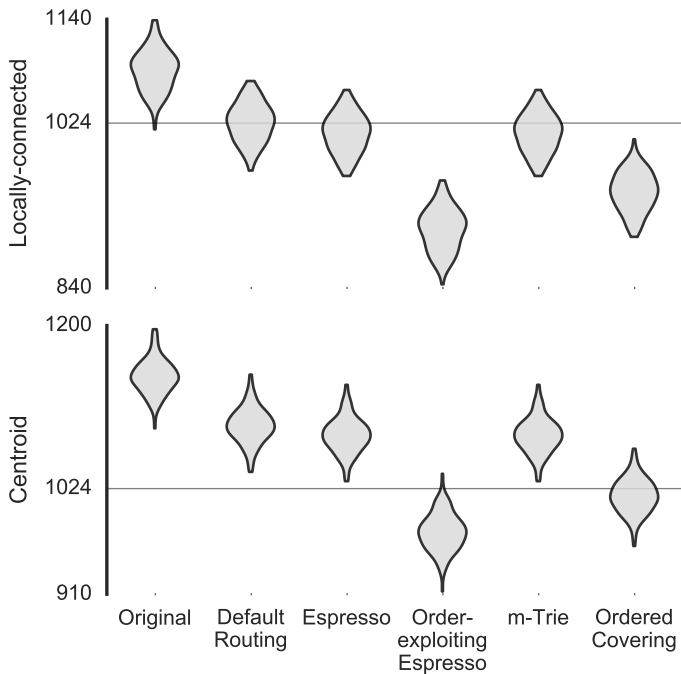
## Resolving the *down-check*

	0000	N	0000
	0011	N	0011
	011X	N	011X
	XXXX	4	0101 1000 1001...
<hr/>			
	011X	N	011X
	00XX	N	0000 0011
	XXXX	4	0101 1000 1001...

{0000, ~~0011~~, ~~011X~~} → 0000

{~~0000~~, 0011, 011X} → 0X1X

{0000, 0011, ~~011X~~} → 001XX



## On-chip memory usage

Peak heap usage –

<b>Benchmark</b>	<b>Total</b> / KiB	Table / KiB
Locally-connected	18.4	13.3
Centroid	18.8	14.0

If reclaiming memory – every merge of  $\geq 10$  entries decreases memory usage

## Timing

### Ordered-Covering on SpiNNaker –

<b>Model</b>	<b>Load time / s</b>	<b>Exec. time / s</b>	
		<b>Sufficient</b>	<b>Fully</b>
Locally-connected	3.8	13.9	25.6
Centroid	3.6		25.6

- Locally-connected benchmark –  $64.5\times$  faster on SpiNNaker
- Centroid –  $2.8\times$  faster

As networks scale...

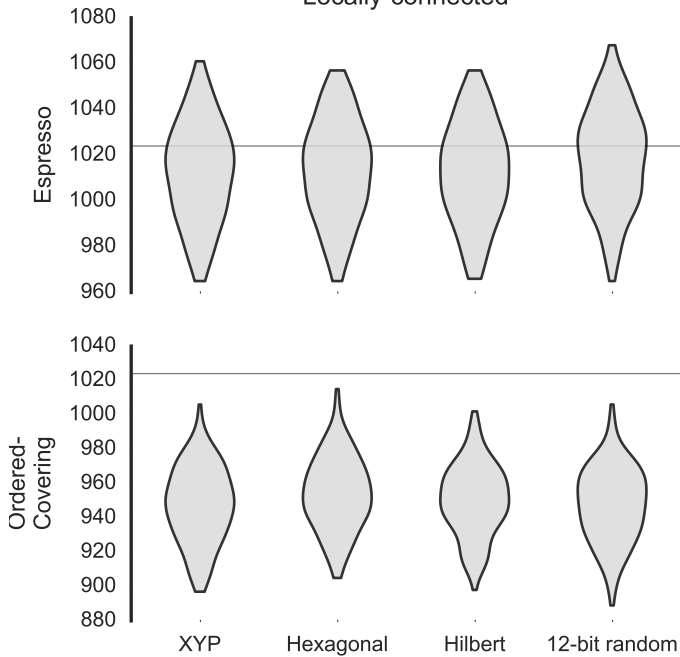
# Thank You

Any questions?

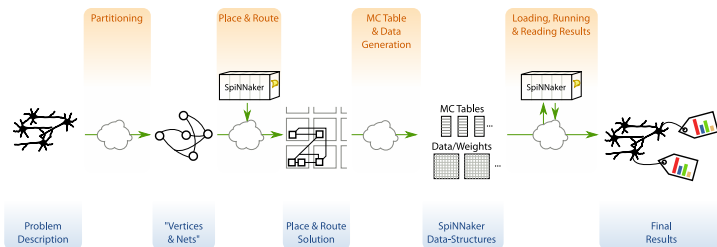
## Selected references

- S. Furber *et al.*, “The SpiNNaker Project,” *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652–665, May 2014, ISSN: 0018-9219
- J. Navaridas *et al.*, “SpiNNaker: Enhanced multicast routing,” *Parallel Computing*, vol. 45, pp. 49–66, 2015, Computing Frontiers 2014: Best Papers, ISSN: 0167-8191
- H. Liu, “Routing table compaction in ternary CAM,” , *Micro, IEEE*, vol. 22, no. 1, pp. 58–64, Jan. 2002, ISSN: 0272-1732

## Locally-connected

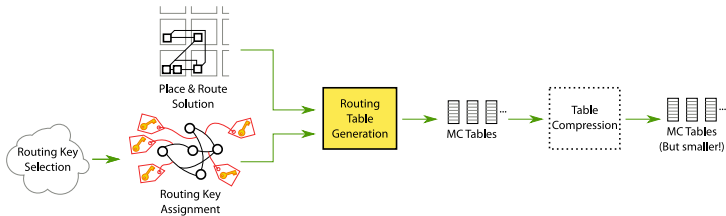


# Using SpiNNaker





## Using SpiNNaker (2)



# (Half)-SpiNNaker

