

# NoCOUT: NoC Topology Generation with Mixed Packet-switched and Point to-Point Networks

Jeremy Chan\* and Sri Parameswaran  
School of Computer Science and Engineering  
The University of New South Wales

[sridevan@cse.unsw.edu.au](mailto:sridevan@cse.unsw.edu.au)

\*currently at Sonics Inc

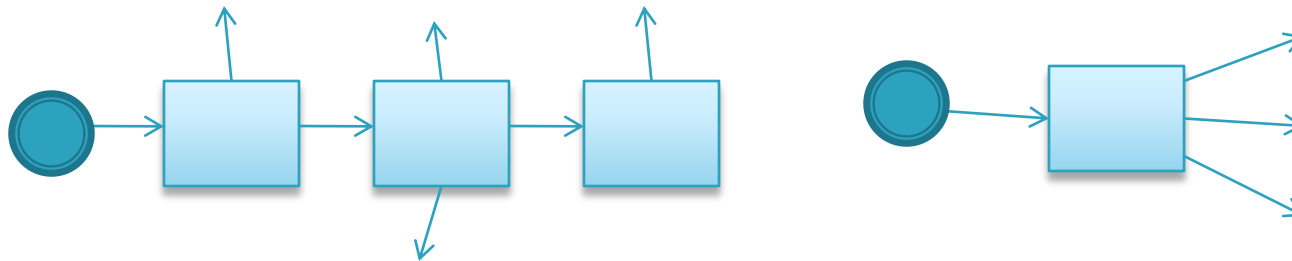
# Introduction

- ▶ SoCs are ubiquitous; smart interconnects/NoC are needed provide high performance, low power solutions for many computing devices (set top boxes, game consoles, mobile phones).
- ▶ Regular topologies too power hungry applications for today's interconnection networks with tens of cores with asymmetric communication patterns.
- ▶ These devices feature custom topologies; e.g. multi-level crossbar, hybrid interconnect architectures



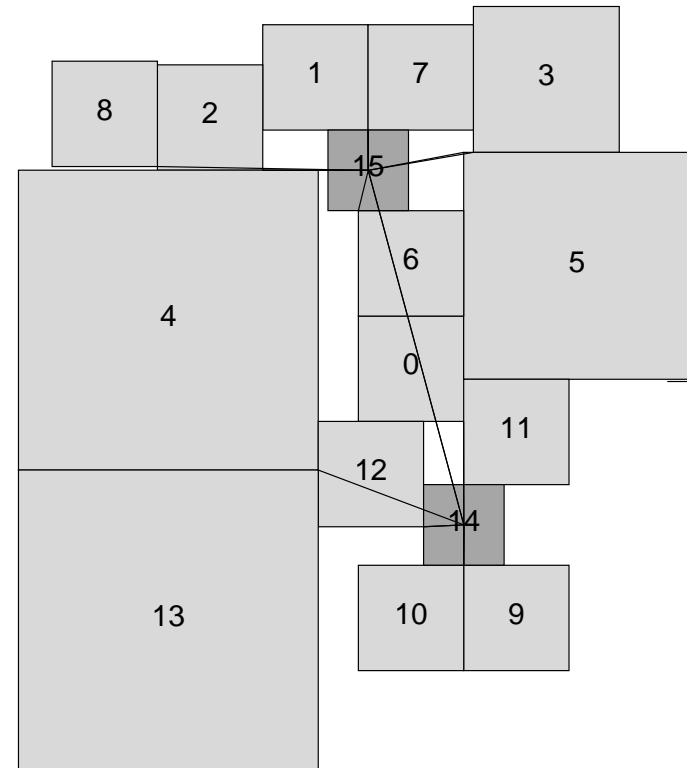
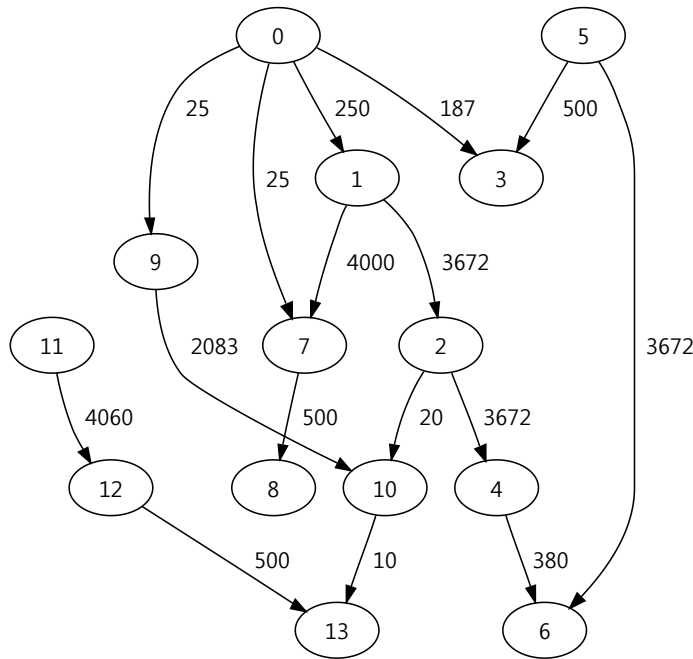
# Motivation

- ▶ Customizing topologies leads to:
  - Reducing energy by reducing hop count and router count
  - Minimizing gate count
  - Possibly higher performance
- ▶ Finding the right topology is difficult (large parameterization space, complex trade offs)



# Problem Description

- ▶ Given the **application trace**, **physical design estimates**, generate an energy-efficient **topology** and **floorplan** that meets design constraints.



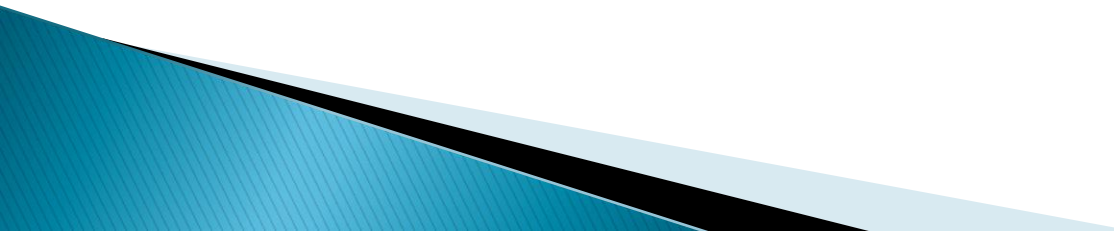
# Related Work

- ▶ Krishnan et al. 2006
  - ▶ Used fixed floorplan as optimization starting point
  - ▶ Solved using ILP based formulations of floorplanning and topology generation
- ▶ Pinto et al. 2006
  - ▶ Mapped to *k*-median and multi-commodity min-cost flow
- ▶ Murali et al. 2006
  - ▶ Min-cut based partitioning heuristic to assign routers and greedy route assignment
  - ▶ Supports deadlock free routing using turn prohibition

Floorplan First  
Router insertion  
Zero router area

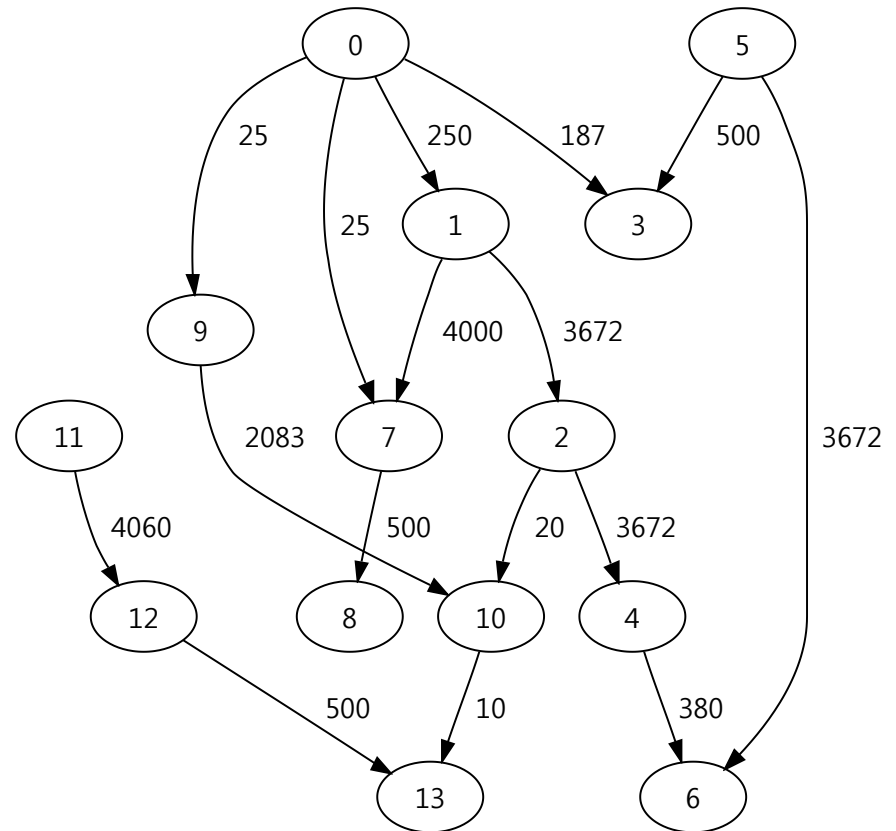
No point-to-point networks

# Our Contribution

- ▶ Propose an iterative optimization algorithm for synthesizing a energy-minimized topology
    - Greater flexibility and better results, but at a cost of higher runtime
  - ▶ Supports a mixture of Point-to-Point and routed networks (Hybrid Network)
  - ▶ Fine grain energy model taking into account leakage and active and dynamic router energy
- 

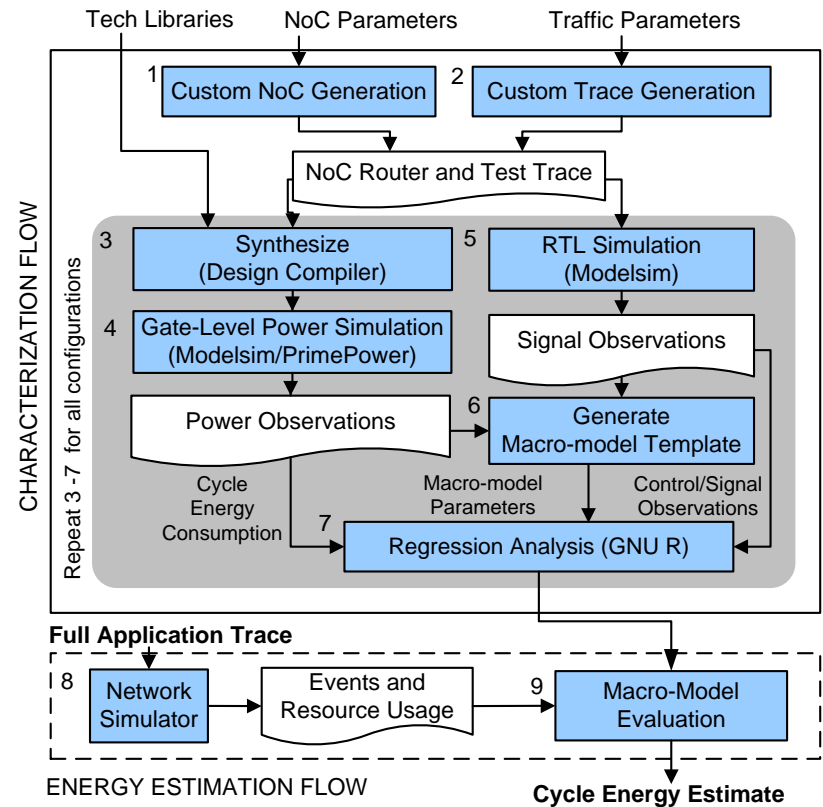
# Application Model

- ▶ Application Volume Graph
  - nodes are initiators/targets)
  - edges volume per application epoch
- ▶ Application run-time parameter  $t$  adjusts average activity rates



# Energy Model

- ▶ Energy and Area Models calibrated using linear regression (NoCEE)
- ▶ Active energy, dynamic energy and static leakage energy
  - Increasing ports increases the cost of dynamic and static energy
  - Configuration specific
  - Fixed buffer depth





# Energy and Area Models

- ▶ Analytical Model for Total NoC energy
  - Routers R, Links L and Network Interface N

$$E_{noc} = \sum_{r \in R} E(r) + \sum_{e \in L} E(e) + \sum_{n \in N} E(n)$$

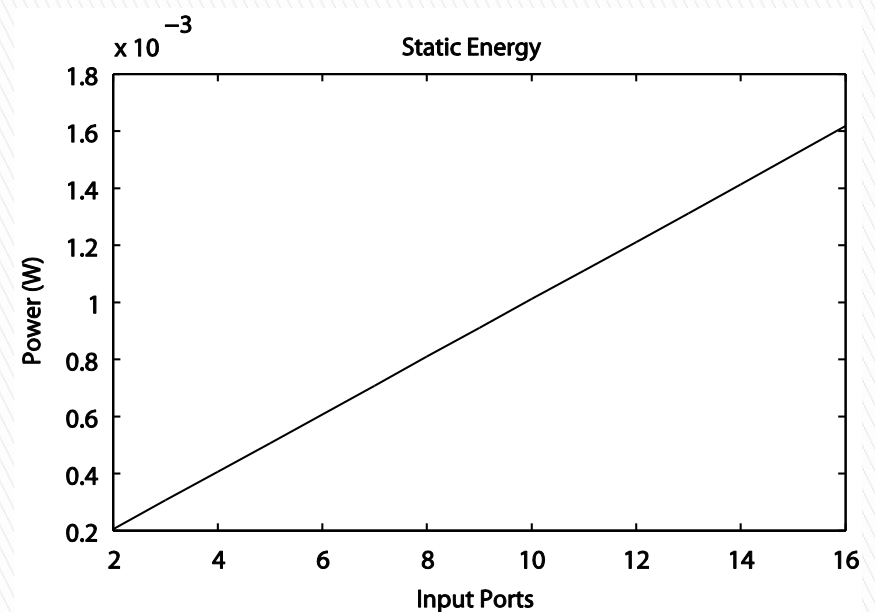
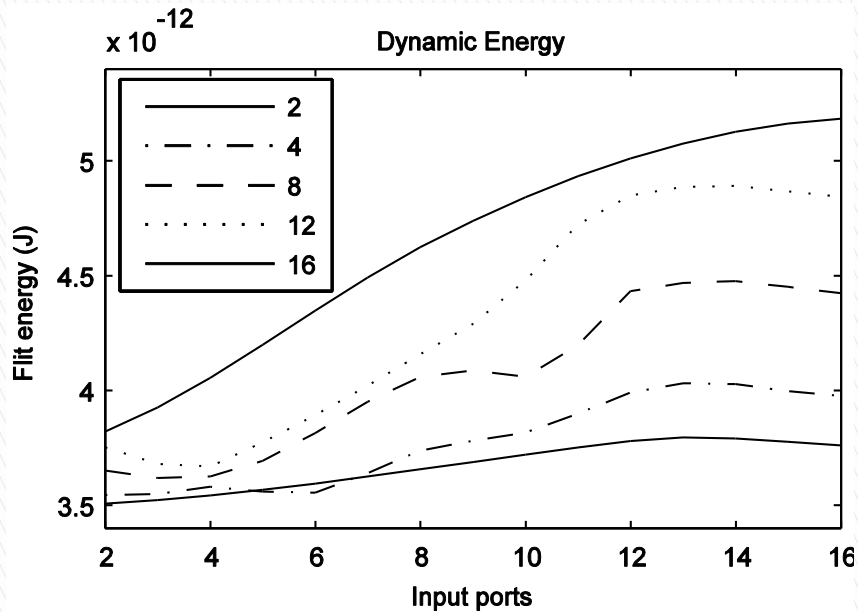
- ▶ Single Router

$$E_{router}(r_i) = E_{dyn}(r_i) + E_{wire}(r_i) + E_{static}(r_i) + E_{leak}(r_i)$$

- ▶ Total Router Energy

$$E_{noc} = \sum_{(i,j) \in L} [E_{dyn\_s}(n_i) + E_{dyn\_t}(n_j) + C_{wdij}] \cdot vol_{ij} \\ + \sum_{n \in N} E_{static}(n) + E_{leak}(n, f) \cdot t$$

# Energy Model



Dynamic Energy Scaling

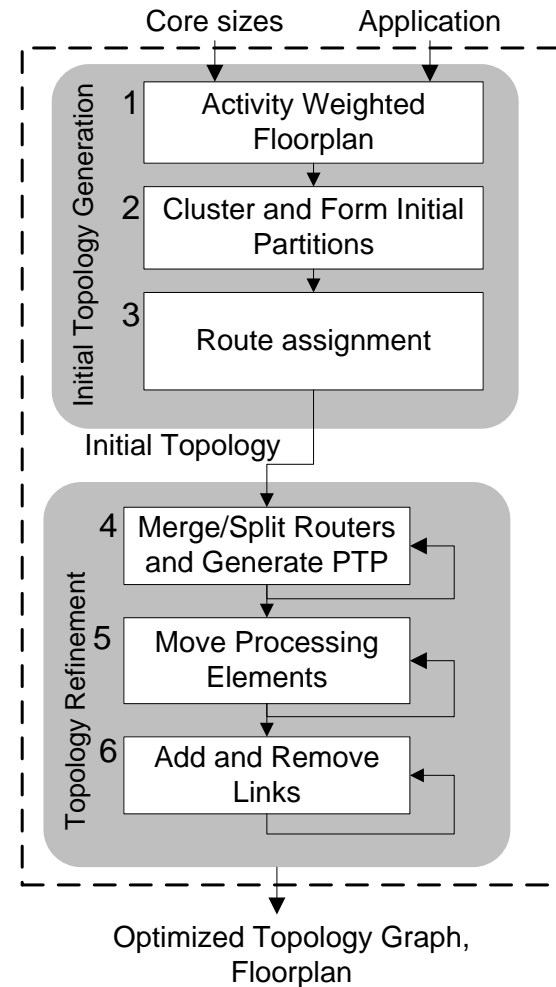
Static Energy Scaling

# Floorplan

- ▶ Capo 10.2 used for floorplanning
  - topology graph and cores sizes used as inputs with weight activity on edges.
  - nets between used to connect communicating cores
- ▶ Cores assumed to have network interface in corners, point-to-point links connected on edges, routers in centre of rectangular area
- ▶ Evaluating link energy requires estimating wire-length
- ▶ Most system-level floorplans use non-deterministic algorithms
  - Inherently unstable, makes comparison of two topologies difficult
  - We take the best of 100 floorplan runs (slow!)

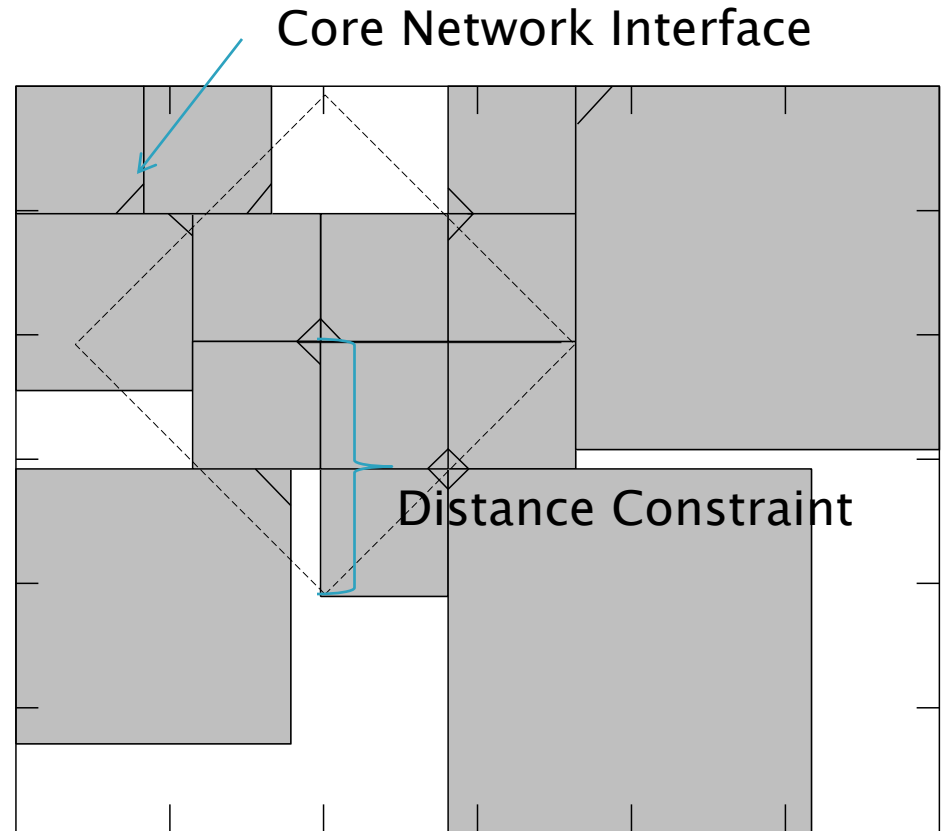
# NoCOUT Algorithm

- ▶ Create Initial Topology Generation
- ▶ Greedy prohibition strategy/Tabu search
- ▶ Types of moves separated into three phases:
  1. Coarse refinement
    - Merge, Split and Split Merge operations, point-to-point creation
  2. Fine partition
  3. Route refinement

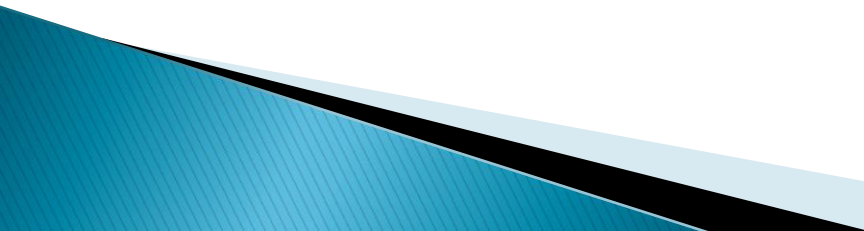


# Initial Topology

- ▶ Perform a activity weighted floorplan
- ▶ Group highly communicating cores
- ▶ Use distance constraint  $D$  to help determine number of routers
- ▶ Experimentally found setting  $D$  to size of largest core produced reasonable partitioning

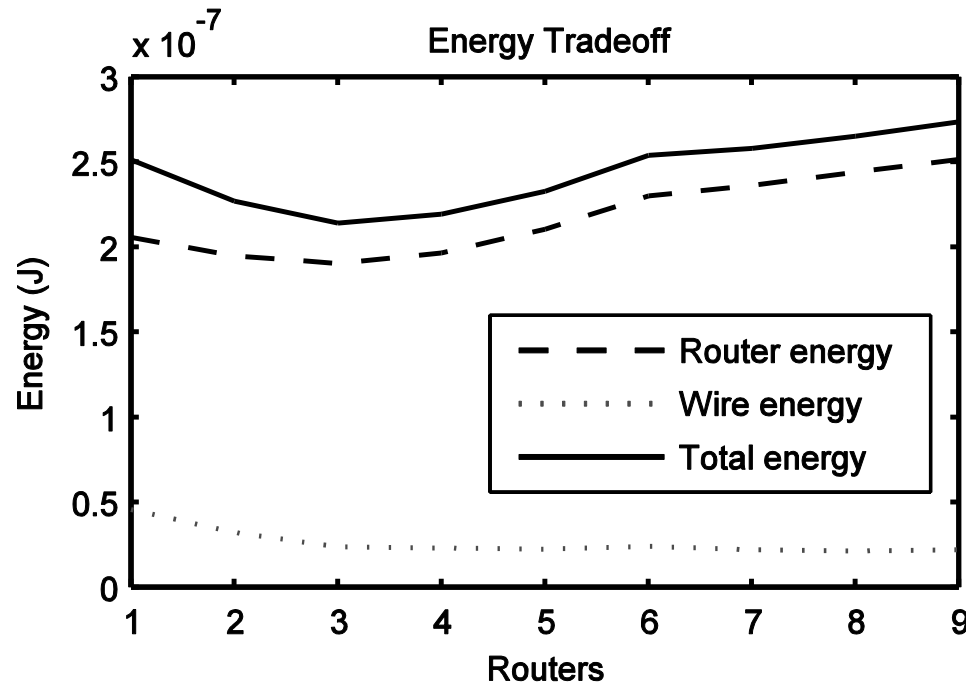


# Coarse Partition Assignment

- 1: **while** merge, split and split-move candidates exist do
  - 2:     Attempt best merge, split, split-move or multi-merge
  - 3:     Perform fine partition refinement (return refined  $T$ )
  - 4:     **if** last merge/split or split-move improves on topology then
  - 5:         Accept refined topology  $T$ , *adjust tabu list*
  - 6:     **else**
  - 7:         Add that merge/split/split-move to the tabu list
  - 8: **return** topology
- 

# Merge Router

- ▶ Merging two routers can reduce energy
  - decreasing the hop count (fewer buffering stages)
- ▶ Each merge pair is evaluated based on potential energy gained, traffic
- ▶ Best merge pair evaluated and accepted, if less than current

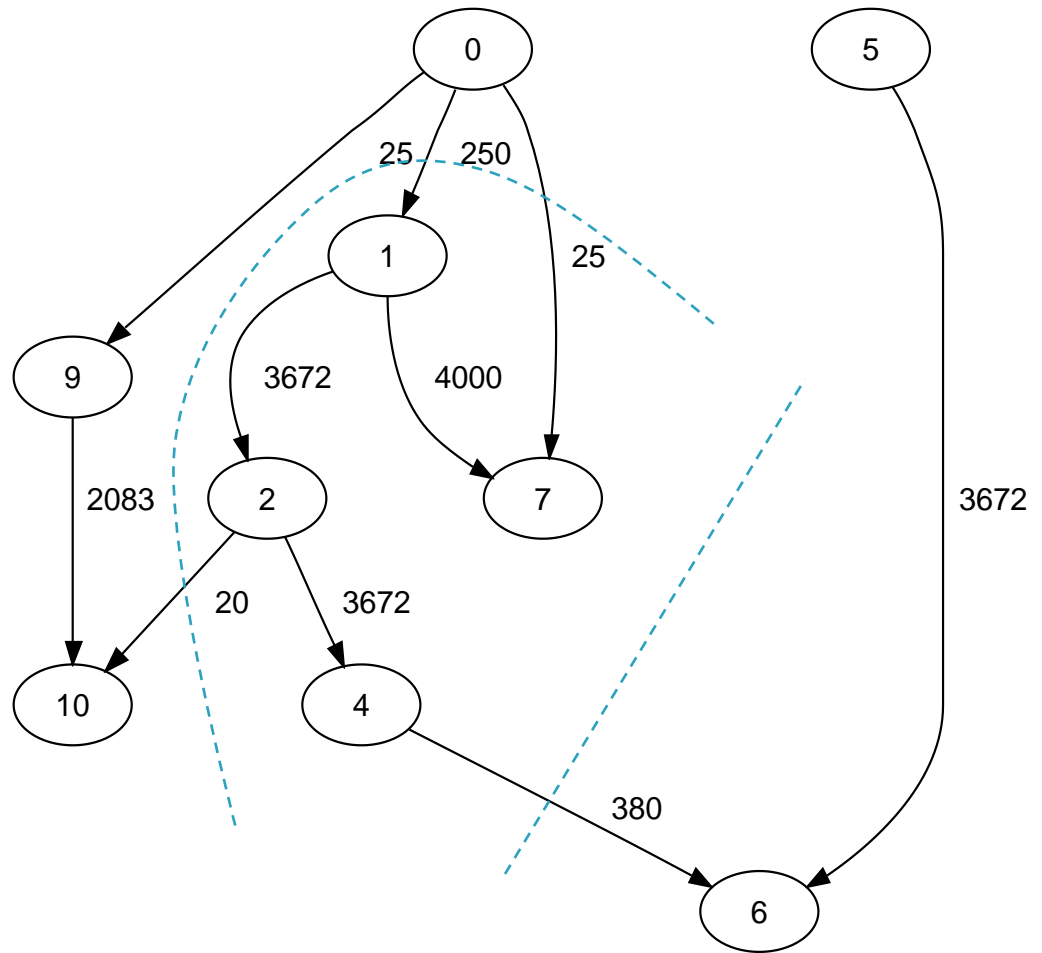


Merging causes wire energy to increase and dynamic energy to increase

Merging reduces router energy due to fewer buffers, slightly increase wire energy

# Split Router

- ▶ Multiple splits/cuts possible
- ▶ Depending on the traffic distribution may not necessarily want a balanced cut



Application Volume Sub-graph

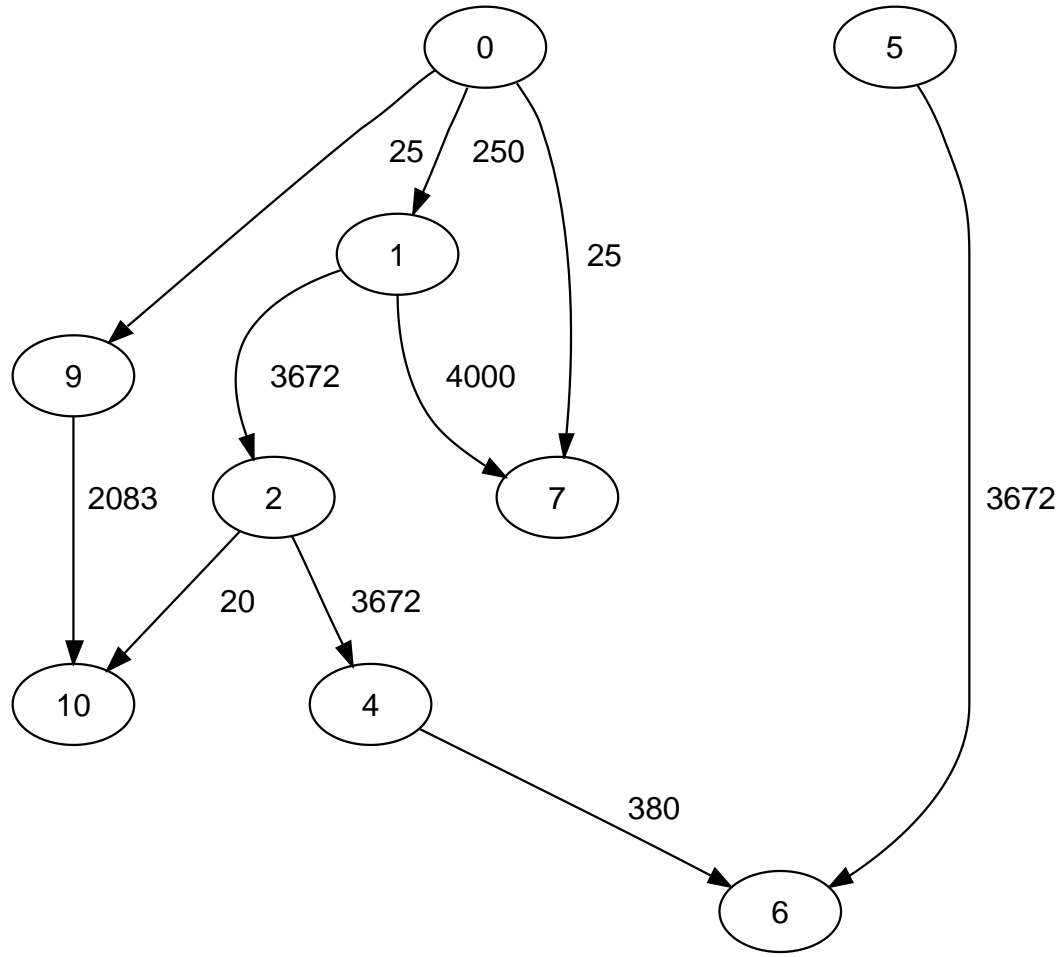


# Split Router Algorithm

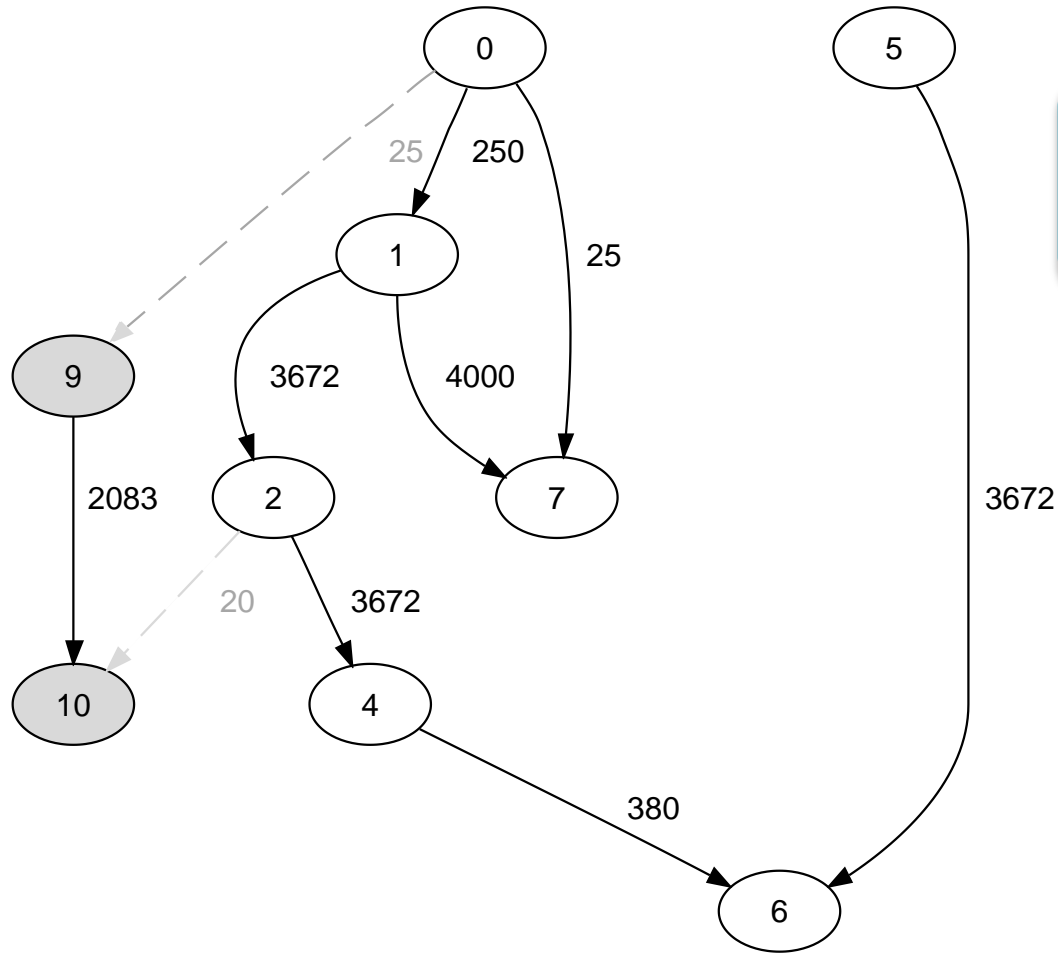
Get Split Candidates

- 1: **for** each router partition  $p$  *in*  $P$  **do**
- 2: Construct application subgraph  $A'$  *from*  $p$
- 3: Add edges in subgraph  $A'$  *to edge list*  $EL$
- 4: Sort edge list  $EL$  *by communication volume (ascending order)*
- 5: **while** partition  $p$  *can be split further* **do**
- 6:     Split N Partition( $A0, EL$ )
- 7:     **if**  $\text{Gain}(p) > \text{split threshold}$ , *add to split candidates*
- 8: **return** split candidates

# Split Router Example



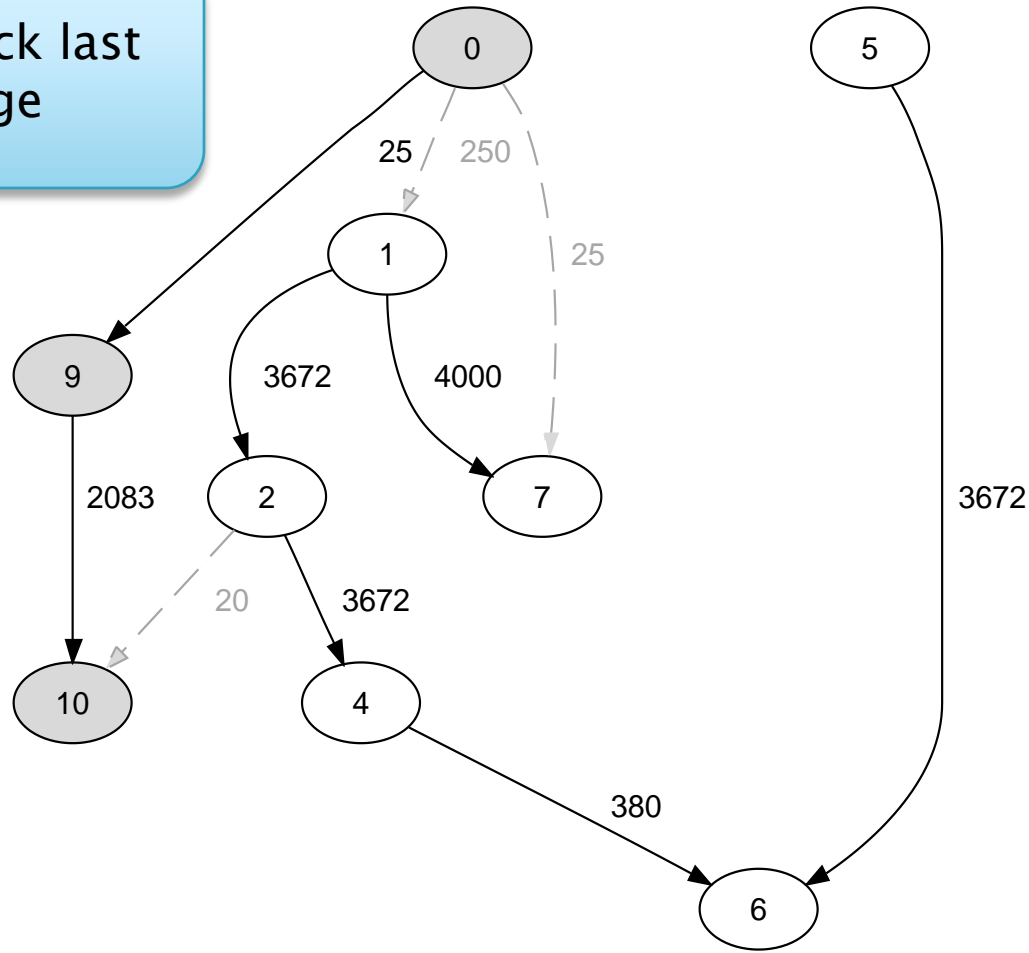
# Split Router Example



remove min  
weighted edges  
first {20, 25}

# Split Router Example

add back last edge



continue removing min-edge until next partition is formed {25,25}

# Add Point-to-Point Connection

- ▶ PTP links have additional network interface cost.
- ▶ PTP links added late in the optimization phase when no other merge or split operation suggested to avoid prematurely adding ptp links
- ▶ Two passes to find point-to-point links
  - Evaluate the effect replacing the network streams in the original graph with point-to-point links
  - Evaluate replacing all network interfaces on a core with point-to-point connections.
- ▶ After, point-to-point links added and accepted (decreases energy), return to merge/split optimization phase

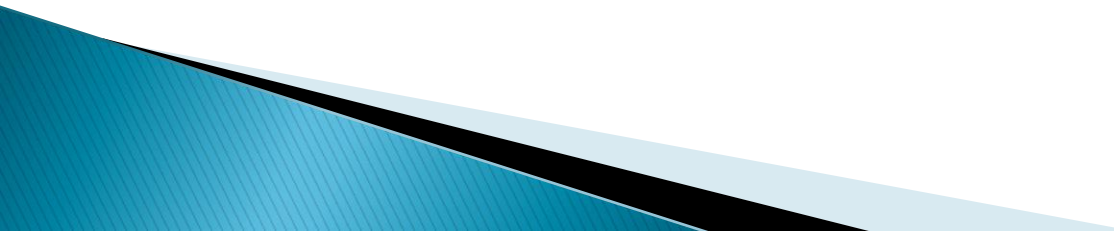
# Fine partition refinement

```
1: while move candidates  $\neq 0$ ; and  $u < U_{\text{threshold}}$  do
2:   Attempt best move candidate
3:   Perform route refinement (return refined  $T$ )
4:   if Last move improves topology then
5:     Accept refined topology  $T$ 
6:   else
7:     Add Last move to the tabu list, increment  $u$ 
8: return topology
```

threshold  
limits the time  
spent in each  
phase without  
gain

Tabu lists used to prohibit  
infeasible parts of the design  
space

# Route Refinement

- ▶ New links are added and removed based on potential energy gain
  - ▶ Use the change in traffic.
  - ▶ A weighted lottery based selection, links with higher probability.
  - ▶ Removal of critical routes are prohibited.
- 

# Experimental Setup

- ▶ Three graph categories:
  - Asymmetric (G1–G4)
  - Symmetric benchmark (G8–G9)
  - Larger benchmarks (G5, G10)
- ▶ Adjusted core size and activity rates (G6, G7)

	Graph	PE	edge	Flits '000
G1	263decmp3dec	14	16	23.56
G2	263encmp3dec	12	12	230.2
G3	mp3encmp3dec	13	12	16.52
G4	vopd	13	12	3.116
G5	imp	27	96	11040
G6	large	14	16	23.56
G7	long	14	16	23.56
G8	broadcast	16	120	120.0
G9	mesh4x4	16	40	48.0
G10	random-a	100	200	283.0



# Results

Graph	Mincut				NoCOUT w/o PTP				NoCGEN with PTP				Imp
	R	L	H	P (mW)	R	L	H	P (mW)	R	L	H	P (mW)	
G1	2	23	1.003	7.45	4	24	1.068	7.38	1	17	0.012	6.01	19
G2	2	18	1.000	6.50	2	18	1.001	6.50	0	12	0.000	4.93	24
G3	2	21	1.010	6.43	2	22	1.001	6.45	1	14	0.020	4.90	24
G4	2	23	1.008	7.46	3	24	1.110	7.39	0	12	0.000	5.14	31
G5	8	68	1.236	42.5	13	78	1.290	39.5	1	58	0.121	26.6	37
G6	3	24	1.031	7.79	3	23	1.026	7.69	1	17	0.012	6.24	20
G7	1	21	1.000	5.05	1	21	1.000	5.05	2	19	0.786	4.70	7
G8	1	30	1.000	24.0	1	30	1.000	24.0	1	30	1.000	24.0	0
G9	2	34	1.356	15.5	2	34	1.356	15.5	2	38	1.002	15.5	3
G10	20	215	1.389	75	14	218	1.221	73.5	15	224	0.627	66.5	11

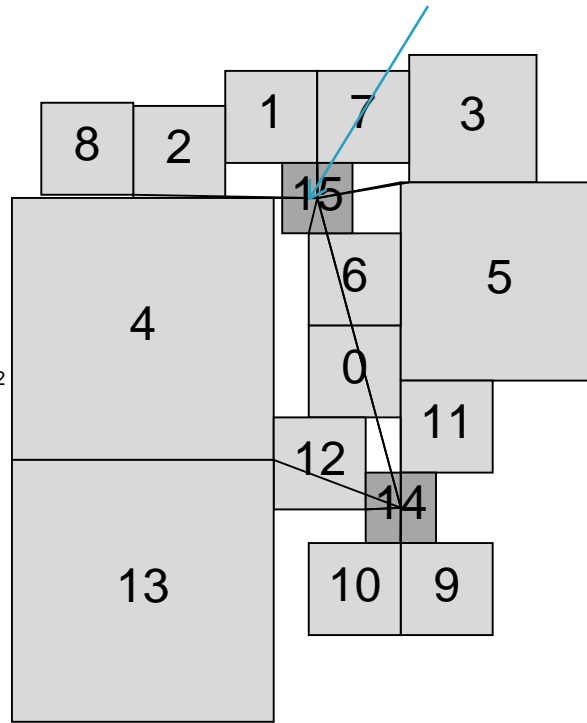
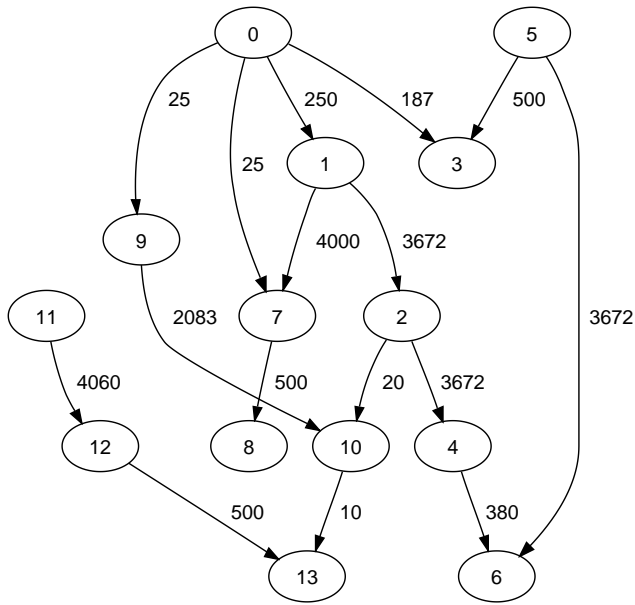
Lower Hop/Wire length energy saving is distance

# Results – CPU runtime (sec)

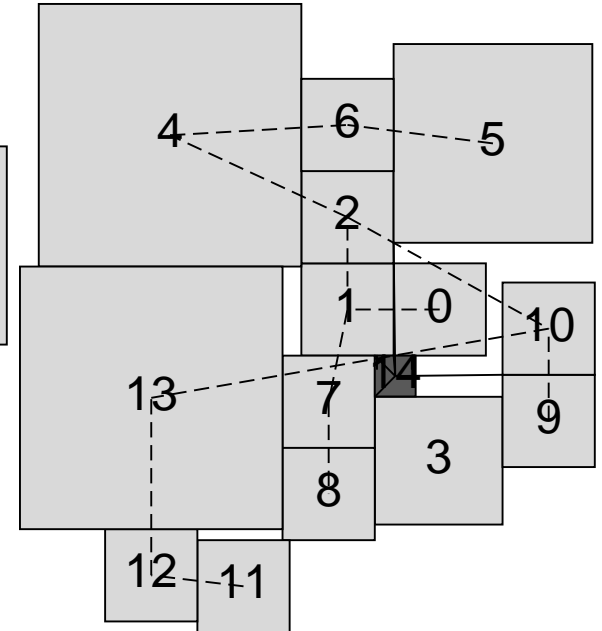
Graph	Execution Time (Mincut)	Execution Time (NoCOUT)
G1	50	150
G2	40	20
G3	37	310
G4	37	280
G5	200	1200
G6	51	375
G7	50	55
G8	50	375
G9	50	220
G10	250	1800

# 263decmp3dec

highly communicating cores  
places close to each other



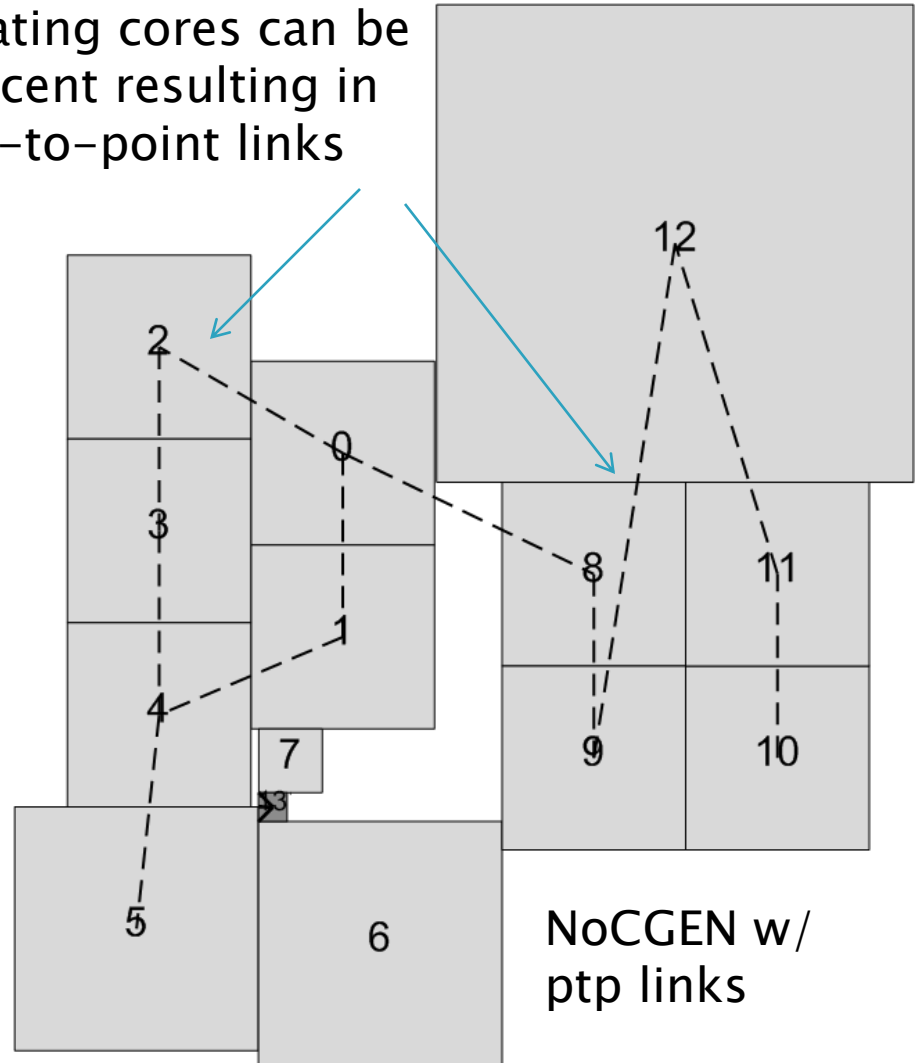
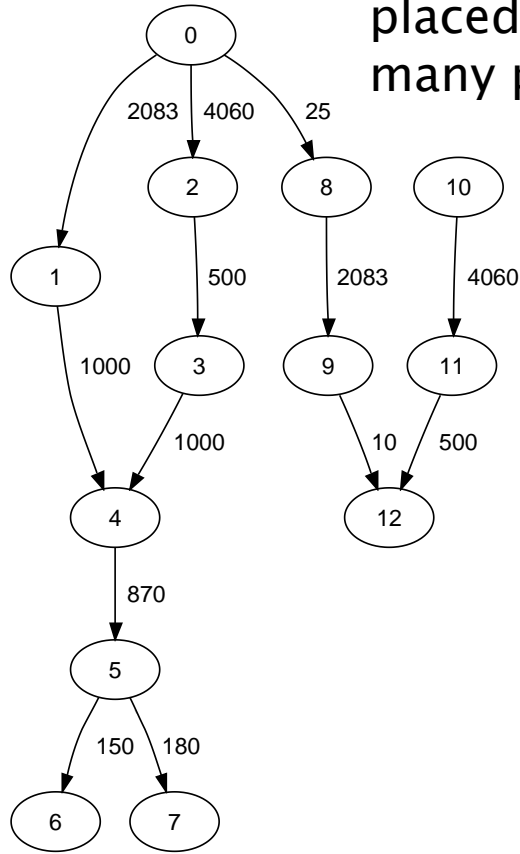
Mincut



NoCGEN w/  
ptp links

# mp3encmp3dec

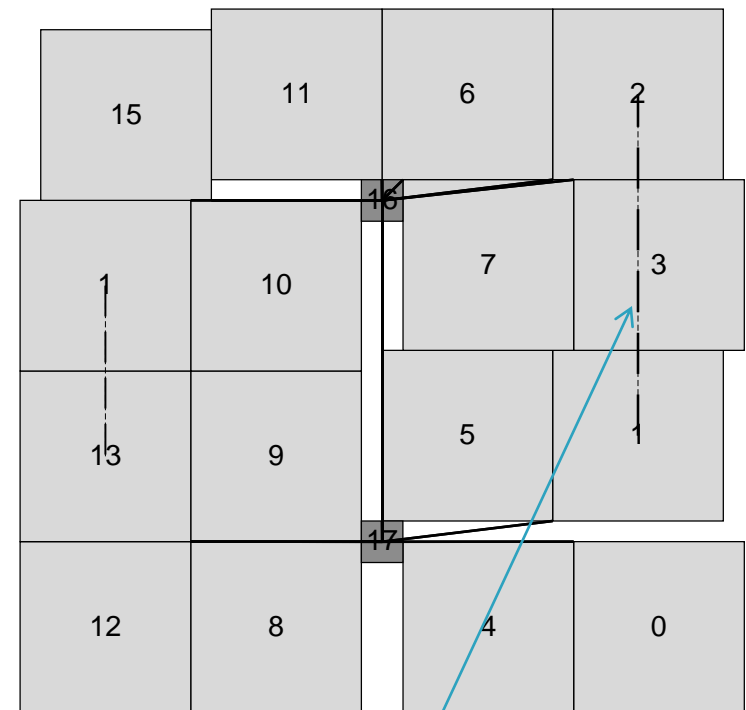
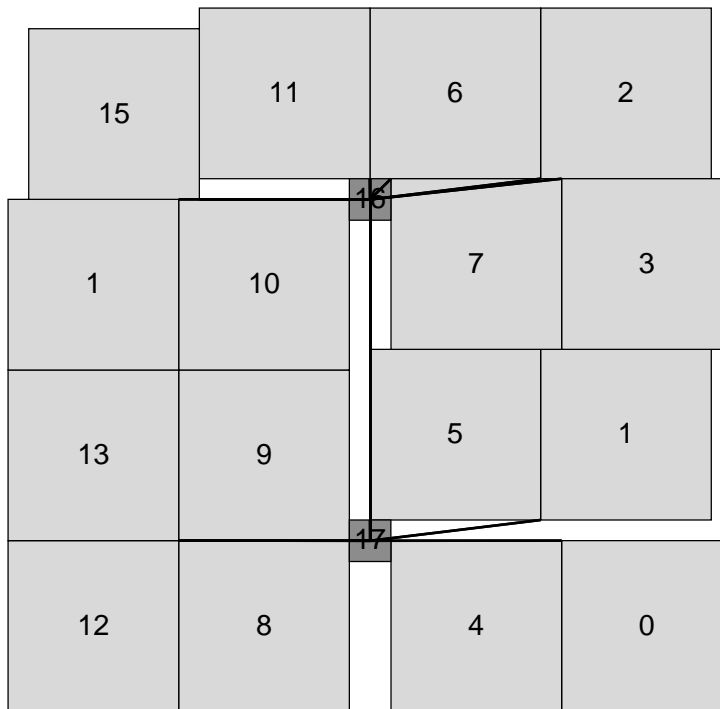
Communicating cores can be placed adjacent resulting in many point-to-point links



NoCGEN w/  
ptp links

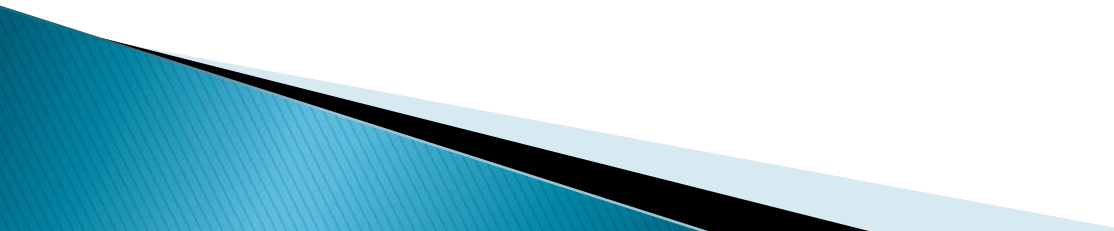
# mesh4x4

Even regular traffic patterns may benefit from point-to-point networks

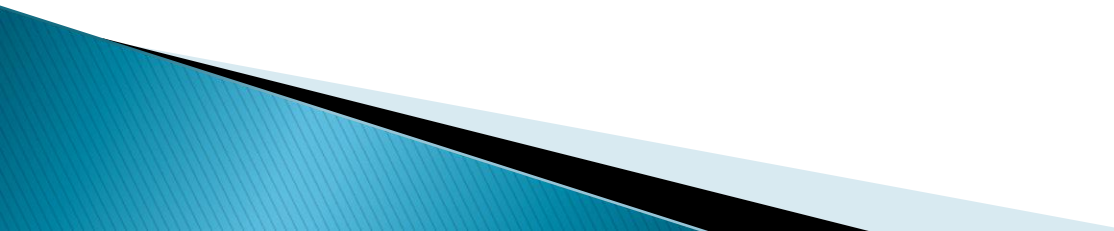


System-level floorplanner may not place near the desired

# broadcast

- ▶ Every core communicates equally with other cores
  - ▶ Due to large number of communicating partners a crossbar architecture is produced.
  - ▶ May vary with energy models (fewer or more routers)
- 

# Conclusion

- ▶ Adding point-to-point links can improve energy consumption in applications where initiators only communicate with few cores.
  - ▶ Proposed an iterative energy-optimization algorithm to automates the creation of a NoC.
  - ▶ We shown that using our design flow we can produce superior results to the min-cut flow without point-to-point support.
- 

**Thank You**

A decorative graphic at the bottom of the slide consisting of a dark blue wavy shape on the left, a black horizontal bar in the middle, and a light blue wavy shape on the right.