Network-on-Chip 最前線

～研究の始め方から最新動向まで～

松谷＠慶應

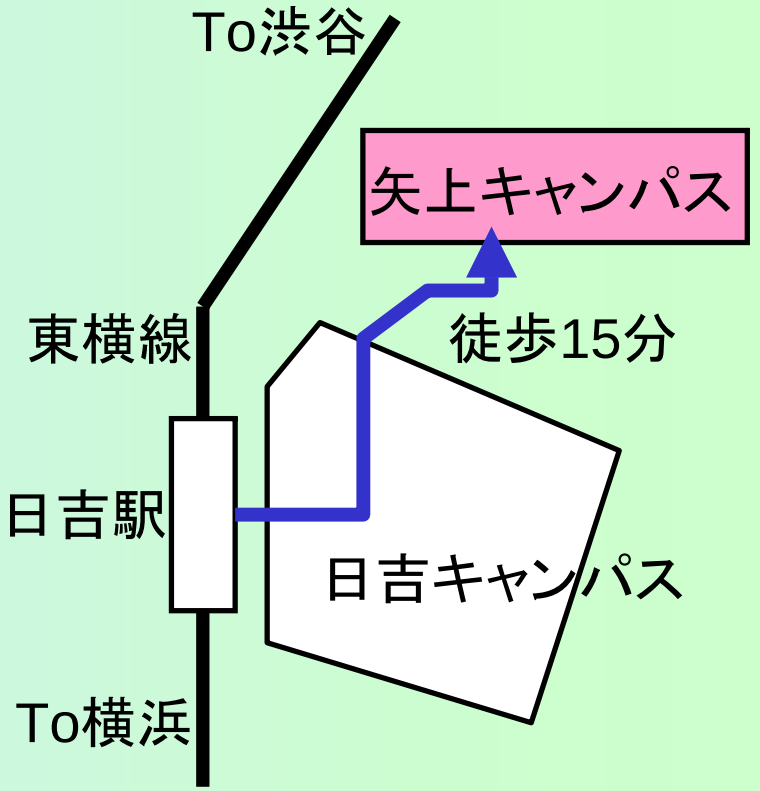
Revised version (Aug 21, 2008)

ご連絡

- NoC generator のソースコードを見るには,
  - http://www.am.ics.keio.ac.jp/~matutani/papers/router-20080808.tar.bz2
- さらに NoC の Verilog コードを生成するには,
  - Perl コマンドが必要
- さらに RTL シミュレーションを試すには,
  - Verilog simulator が必要
  - 今日のデモでは iverilog (http://bleyer.org/icarus) を使用
- (補足) 今回の NoC generator は一部の機能を限定
  - テクノロジに依存する部分 (メモリマクロ,電力,レイアウト関連)
  - 理解を妨げる複雑な機能 (マニアックなトポロジ,ルーティング)
  - NDA 的に OK なら, 可能な限り, 必要なデータ出します

自己紹介: 松谷 宏紀

- 現在の所属
  - 慶應大学 理工学部 訪問研究員
  - 日本学術振興会 特別研究員(PD)
  - 天野研に居候中
    (矢上キャンパス 26-112B)



- 経歴
  - 1982年 埼玉県川口市生まれ
  - 2004年 慶應大学 環境情報学部 卒業 (村井研)
  - 2006年 慶應大学 理工学研究科 修士課程 卒業 (天野研)
  - 2008年 慶應大学 理工学研究科 博士課程 卒業 (天野研)

発表の流れ

- Network-on-Chip (NoC) の概要
  - ネットワークトポロジ
  - パケットルーティング
  - ルータアーキテクチャ
- NoC の研究の始め方
  - NoC シミュレータ
  - ルータ回路 (NoC generator)
  - NoC の評価方法
- NoC 研究の動向
  - 最近ホットなトピック
  - 予測機構による低遅延ルータ

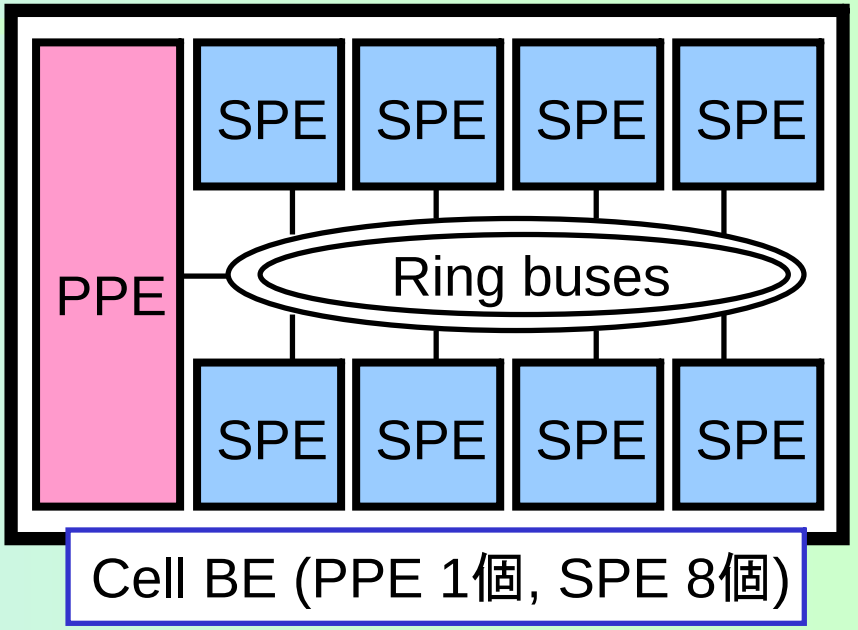
[松谷,鯉渕,天野,吉永]

はじめに: マルチコア化への流れ

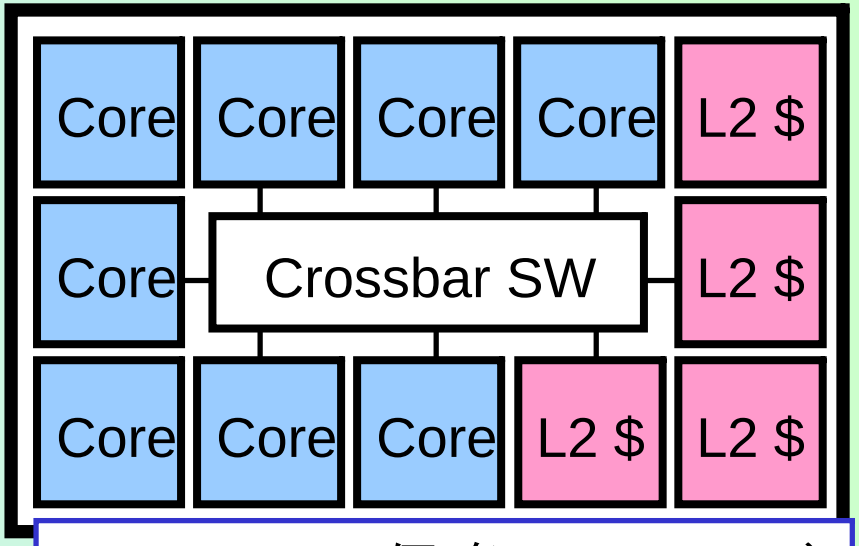
- 半導体技術の進歩
  - 複数の計算コアを集積可
  - 消費電力の増加

動作周波数の向上は頭打ち

- マルチコア化
  - コア数を増やす (2～80コア)
  - 動作周波数は低く抑える
  - 並列化でスループットを稼ぐ
- マルチコアの例
  - STI Cell BE [Pham, ISSCC'05]
  - Sun T1(Niagara) [Kongetira, micro'05]



Cell BE (PPE 1個, SPE 8個)



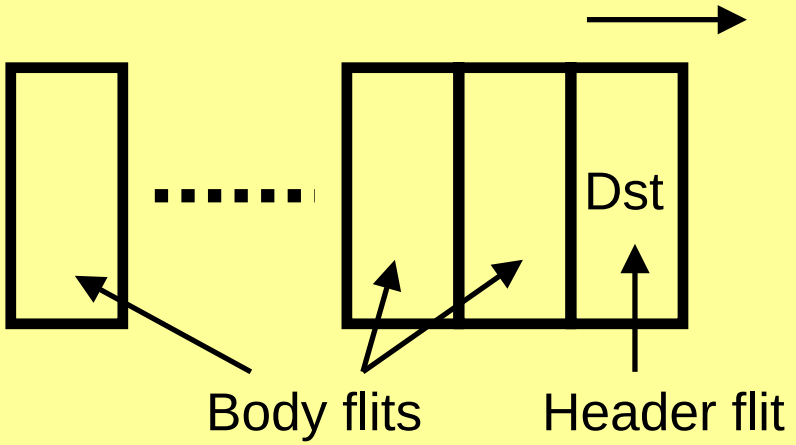
Sun T1(コア8個,各コア4スレッド)

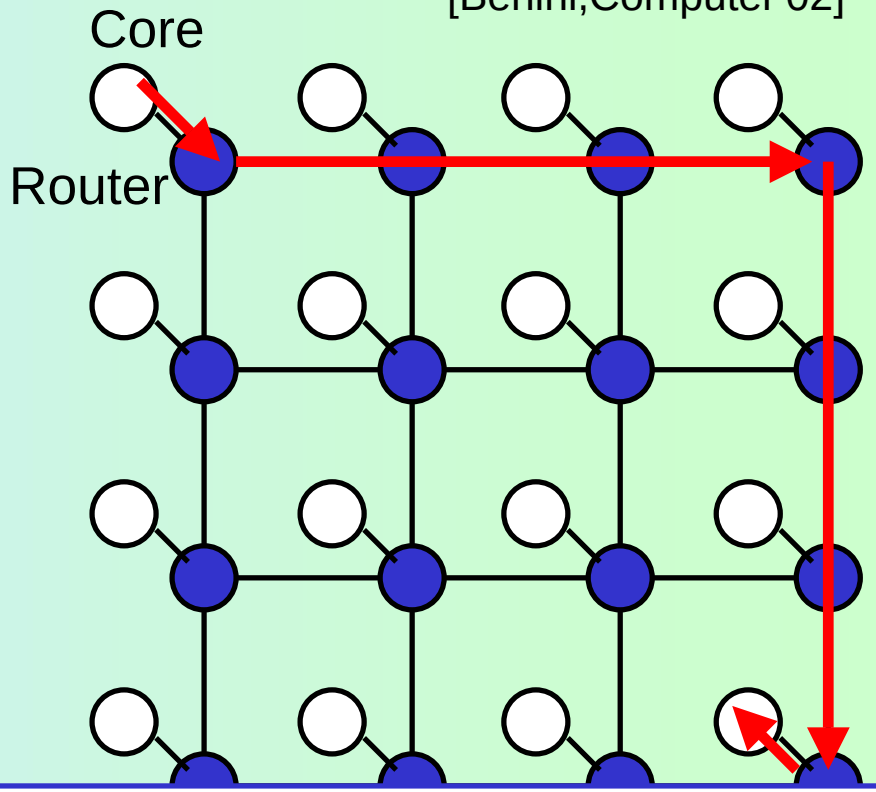
コアの接続方式: バス vs. ネットワーク

- オンチップバス
  - ARM AMBA
  - IBM CoreConnect
- Network-on-Chip (NoC)
  - ネットワーク状に接続
  - パケットスイッチング

[Dally,DAC'01]
[Benini,Computer'02]

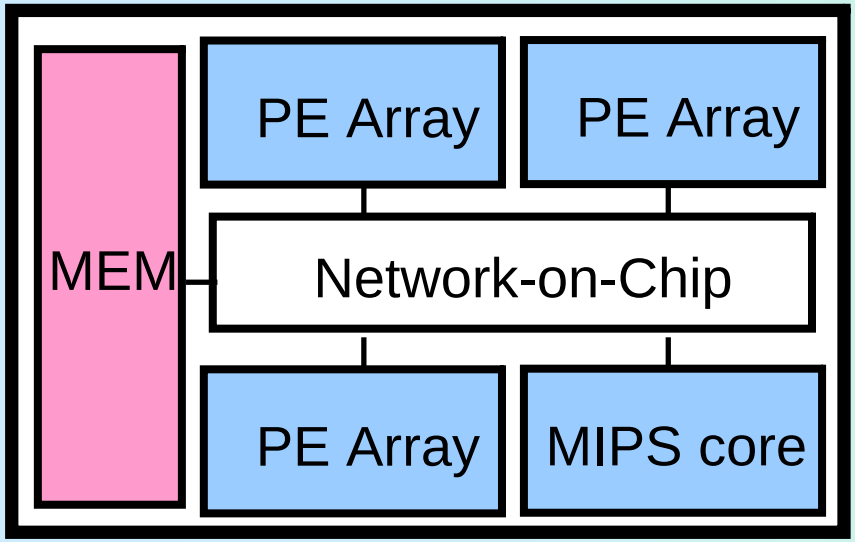
パケットの構造



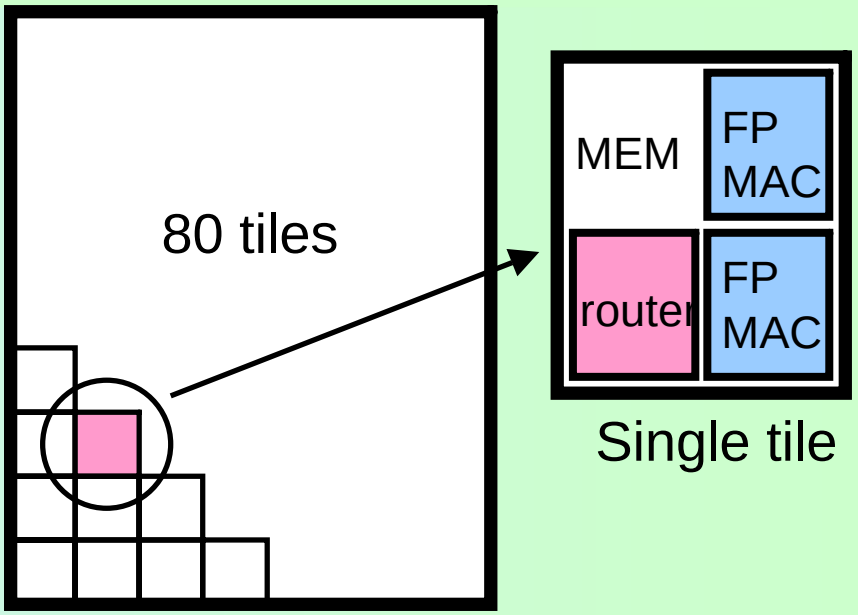


オンチップバスに代わる結合網として Network-on-Chip(NoC)が注目

Network-on-Chips の応用



MuCCRA [天野, ASSCC'07]



Intel 80-core chip [Vangal,ISSCC'07]

オンチップバスの置換え	コア数を増やして電圧低減	科学技術演算向け
小規模 System-on-Chip	組込マルチメディア処理	ハイパフォーマンス系

面積コストの削減

高スループット化,低遅延化

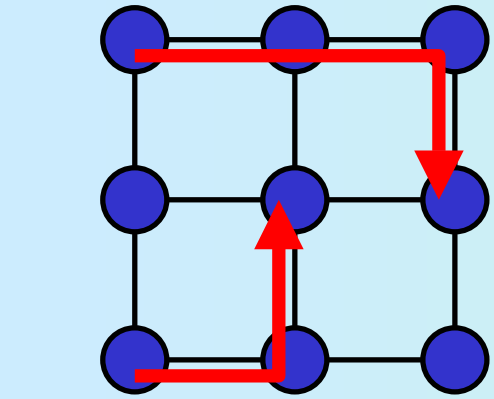
消費電力の削減

Network-on-Chip の研究分野

- いろいろなアプローチ
  - ソフトウェア レベル
  - アーキテクチャ レベル
  - 回路 レベル

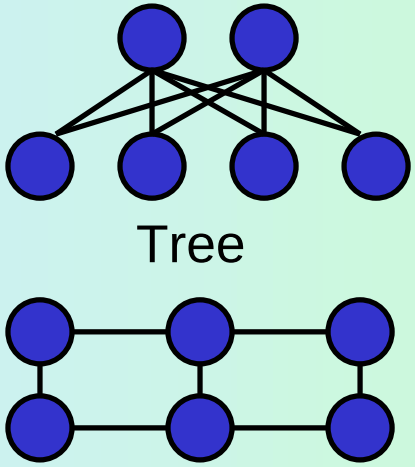
Software Level	OS, task scheduling
Architecture Level	Topology, routing, router architecture
Circuit Level	3D IC, power gating
Device Level	

- ネットワークアーキテクチャ

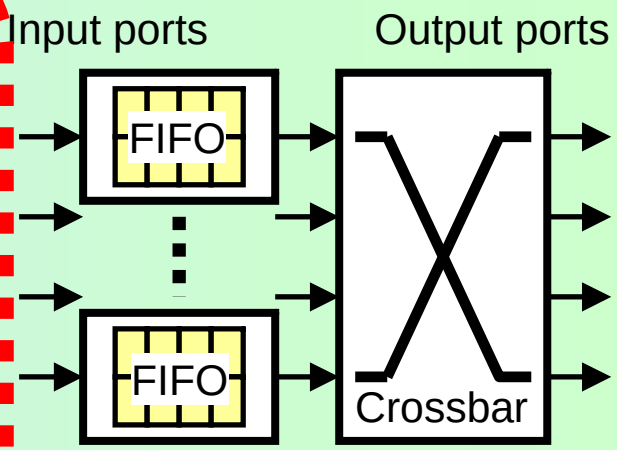


Deadlock-free routing

ルーティング,フロー制御



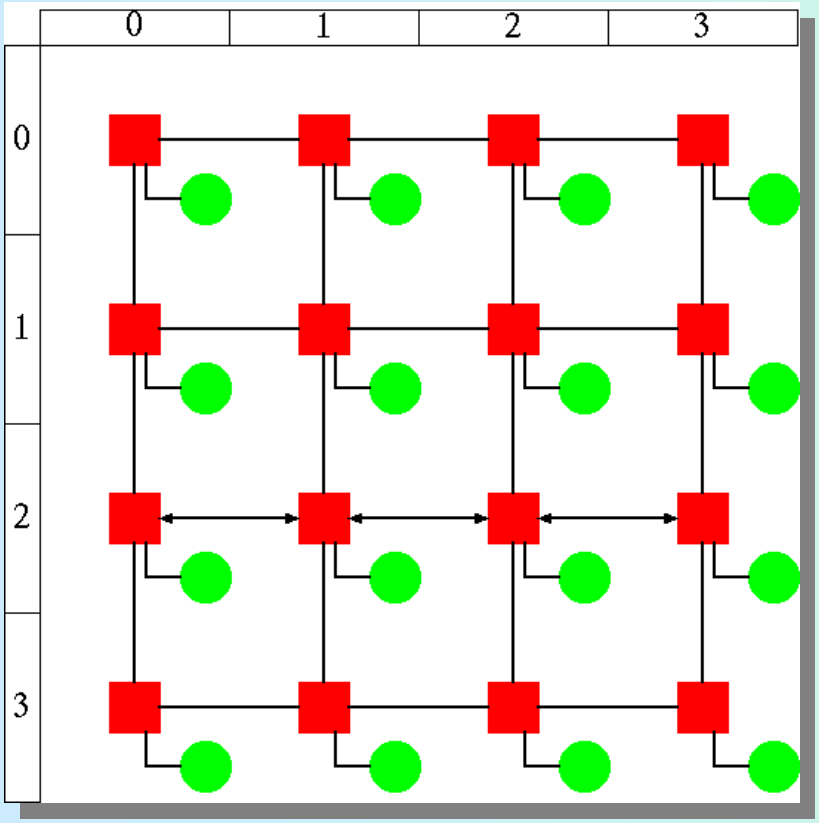
ネットワークトポロジ

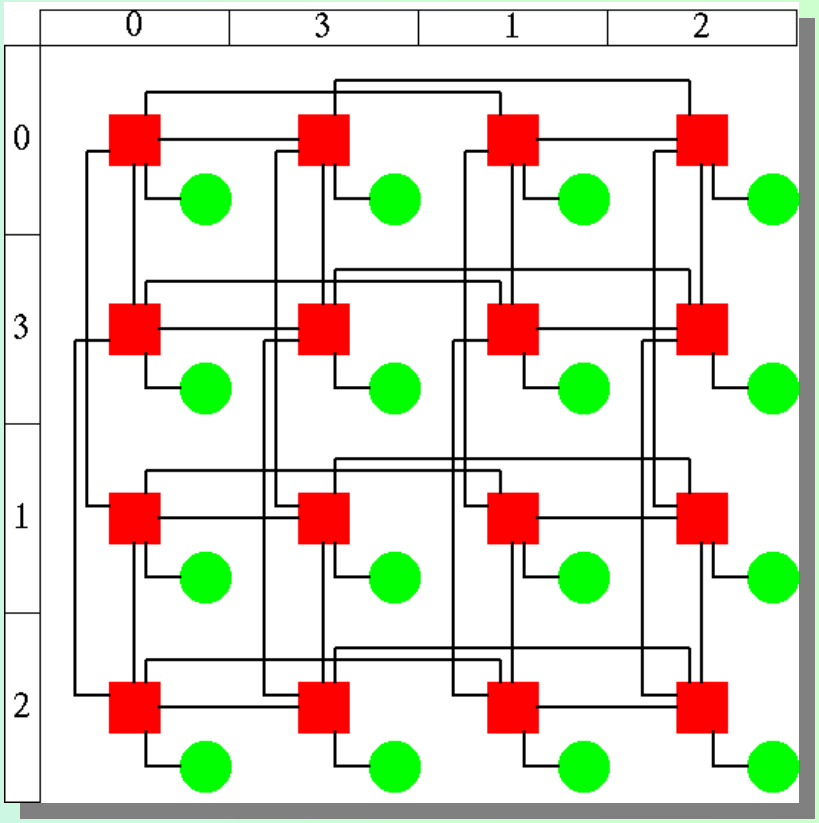


ルータアーキテクチャ

NoC のトポロジ: Mesh & Torus

- 2-D Mesh
  - RAW [Taylor, ISCA'04]
  - Intel's 80-core [Vangal, ISSCC'07]
- 2-D Torus [Dally, DAC'01]
  - メッシュの2倍の帯域





ルータ　計算コア

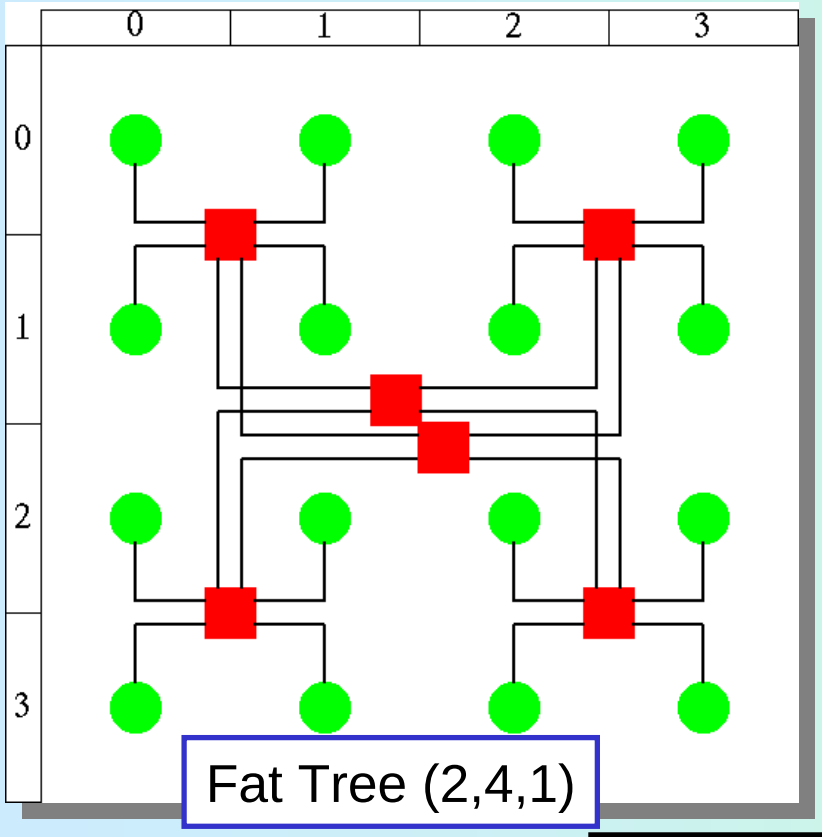
NoC のトポロジ: Fat Tree

- Fat Tree (p, q, c)
  - p: 上位リンクの数
  - q: 下位リンクの数
  - c: コアのポート数

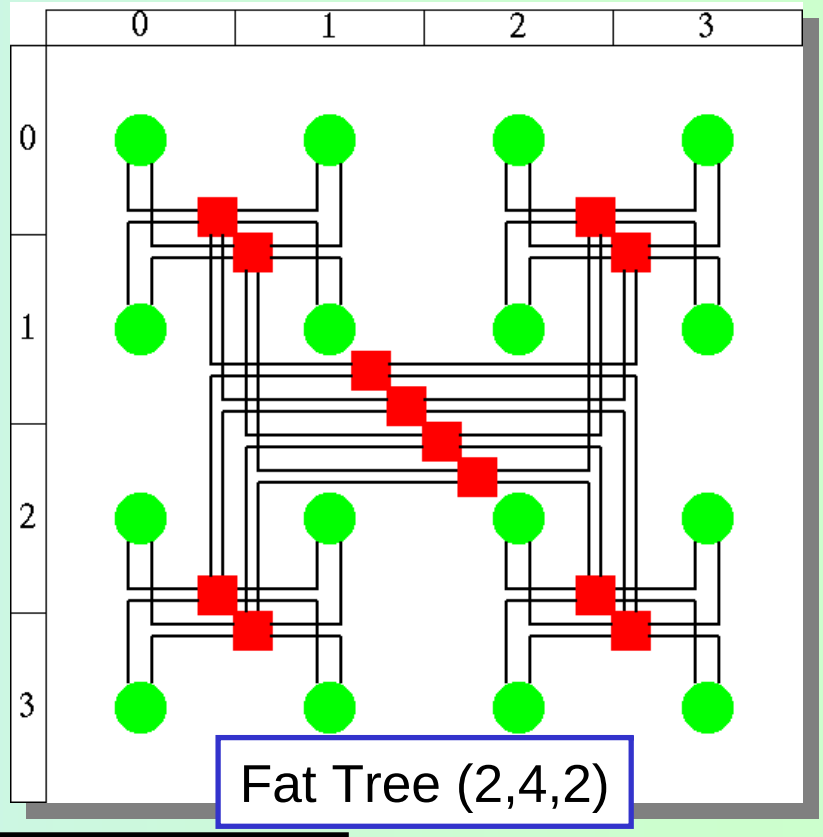
SPIN [Andriahantenaina,DATE'03]

SCORE [Caspi, FPL'00]

ACM [Furtek, FPL'04]



Fat Tree (2,4,1)



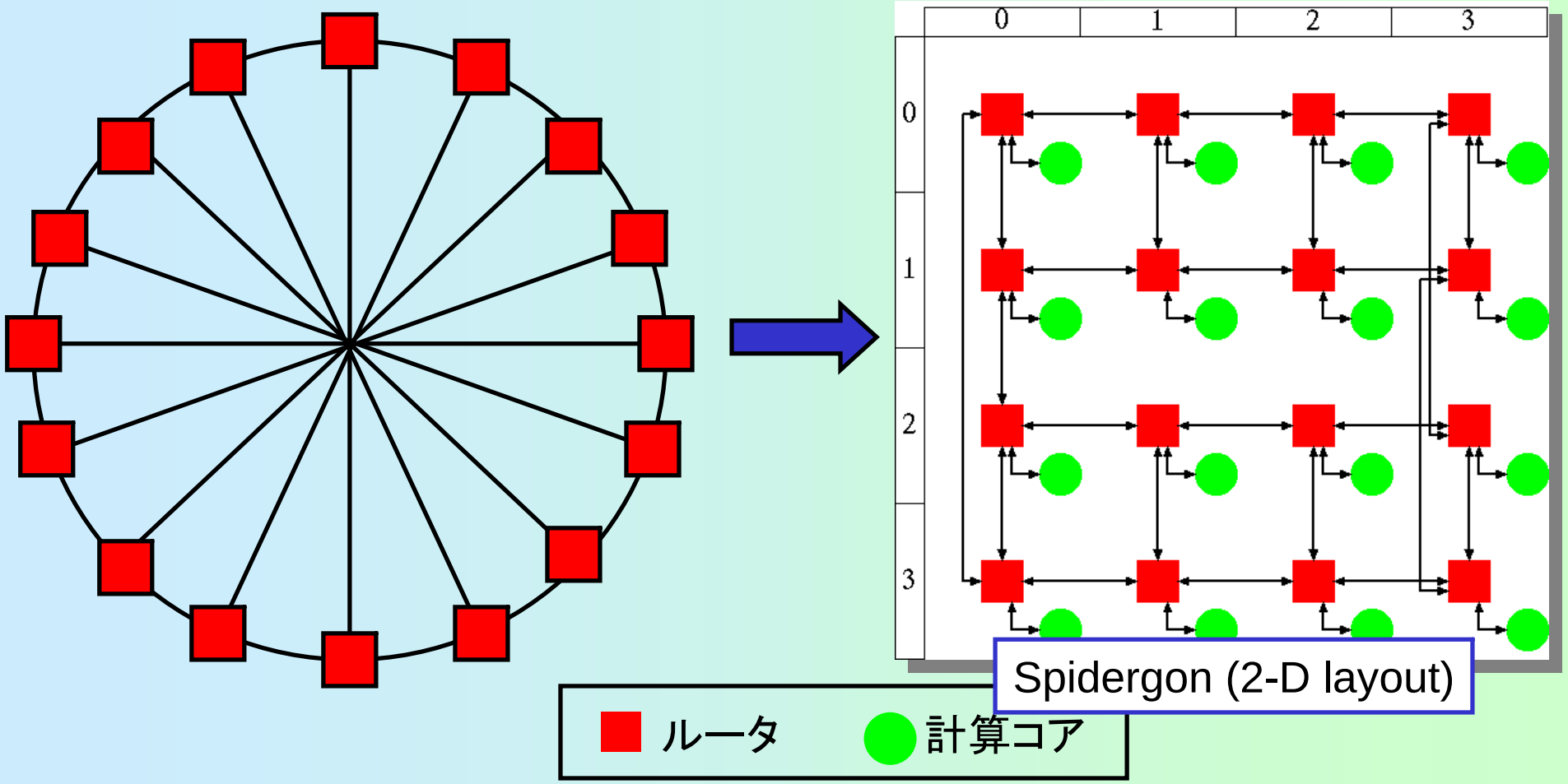
Fat Tree (2,4,2)

ルータ　計算コア

NoC のトポロジ: その他 (1)

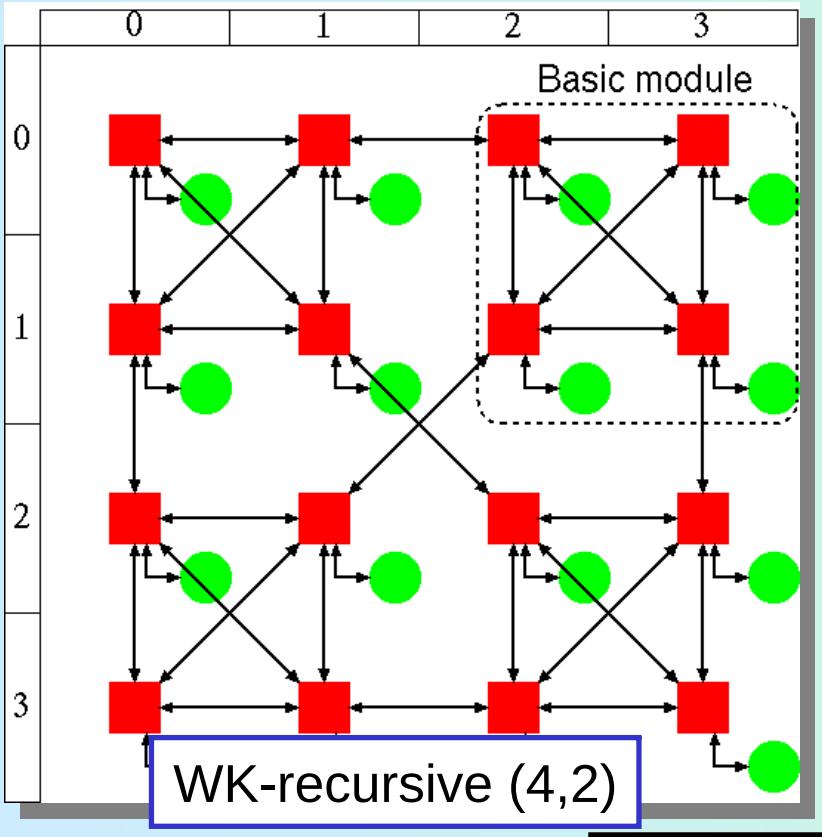
- Spidergon
  - リング + 対角線上に追加リンク
  - Node degree 3; コスト効率が良い

[Coppola, ISSOC'04]
[Bononi, DATE'06]



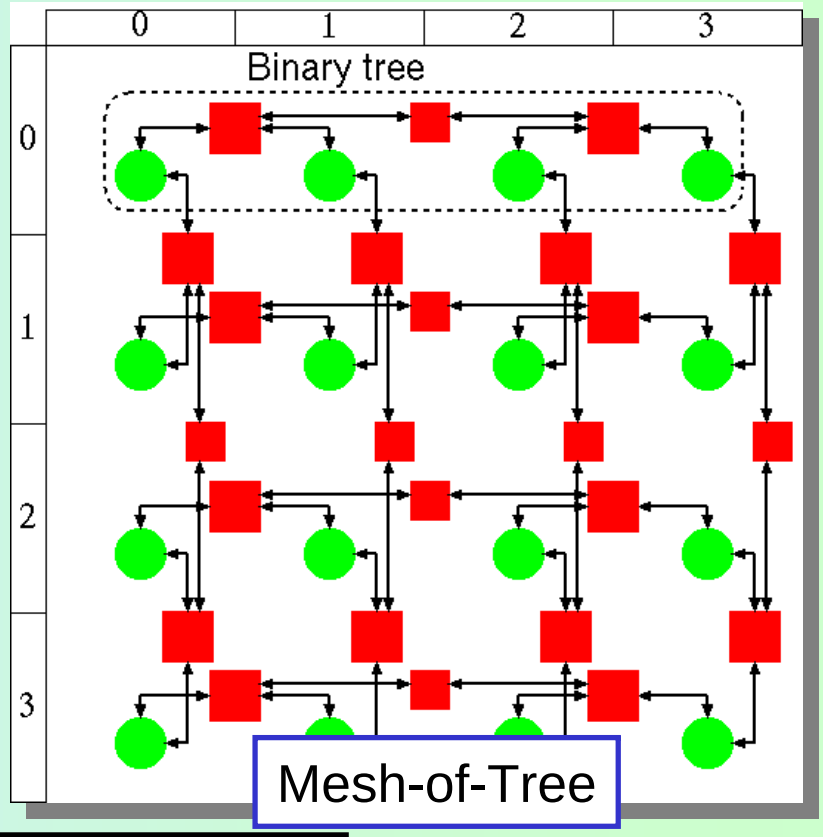
NoC のトポロジ: その他 (2)

- WK-recursive (*d*,*k*)
  - *d*-node の完全グラフ
  - 再帰的に *k* 回結合



[Vecchia, FCGS'88]
[Rahmati, ICCD'06]

- Mesh-of-Tree
  - メッシュ状にコアを並べる
  - 縦/横方向にツリーで結合



[Leighton, Math System theory'84]

ルータ　計算コア

最近のオンチップネットワーク

システム名	トポロジ	ルーティング	スイッチング	フロー制御
MIT RAW	2-D mesh (32bit)	XY DOR	WH, no VC	Credit
UPMC SPIN	Fat Tree (32bit)	Up*/down*	WH, no VC	Credit
QuickSilver ACM	H-Tree (32bit)	Up*/down*	1-flit, no VC	Credit
UMass Amherst aSOC	2-D mesh	Shortest-path	Pipelined CS, no VC	Timeslot
Sun T1	Crossbar (128bit)	-	-	Handshake
Cell BE EIB	Ring (128bit)	Shortest-path	Pipelined CS, no VC	Credit
TRIPS (operand)	2-D mesh (109bit)	YX DOR	1-flit, no VC	On/off
TI				
I				
-				

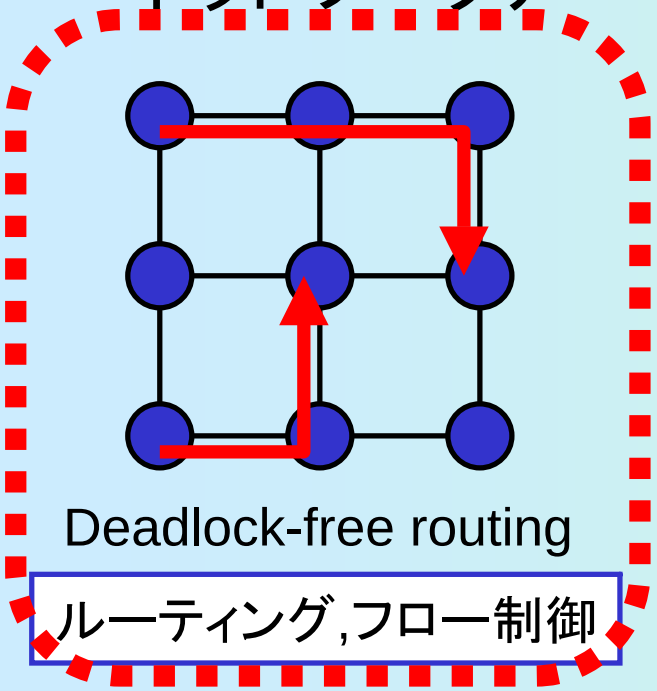
- ネットワークトポロジのまとめ
  - 考慮すべきは, スループット, ルータ面積, 配線量, 配線遅延
  - NoC では配線は豊富に使えるが, ルータ面積は小さくしたい
- 実際は 2-D mesh が良く使われる
  - レイアウトが容易, かつ, パケットルーティングも単純

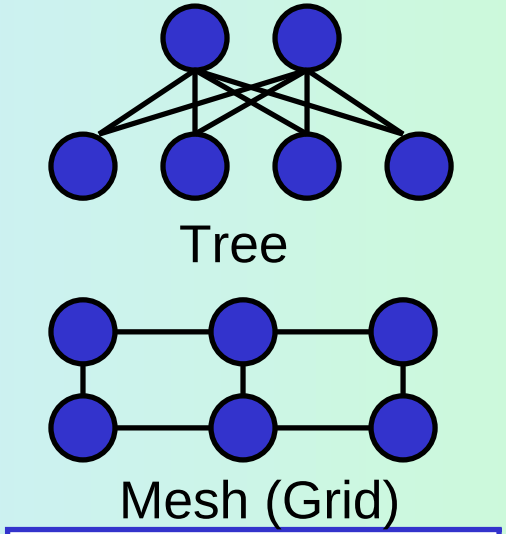
On-Chip Network Architecture

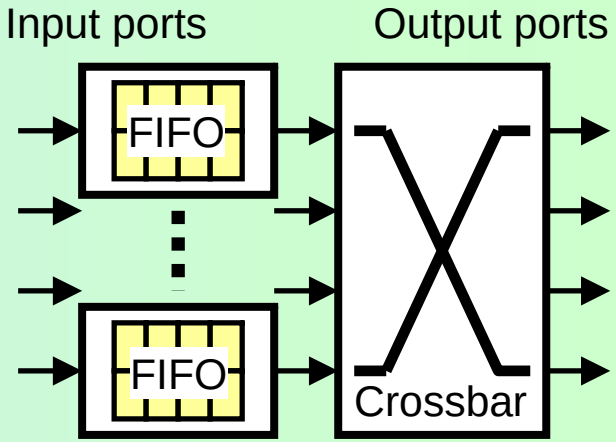
- いろいろなアプローチ
  - ソフトウェア レベル
  - アーキテクチャ レベル
  - 回路 レベル

Software Level	OS, task scheduling
Architecture Level	Topology, routing, router architecture
Circuit Level	3D IC, power gating
Device Level	

- ネットワークアーキテクチャ

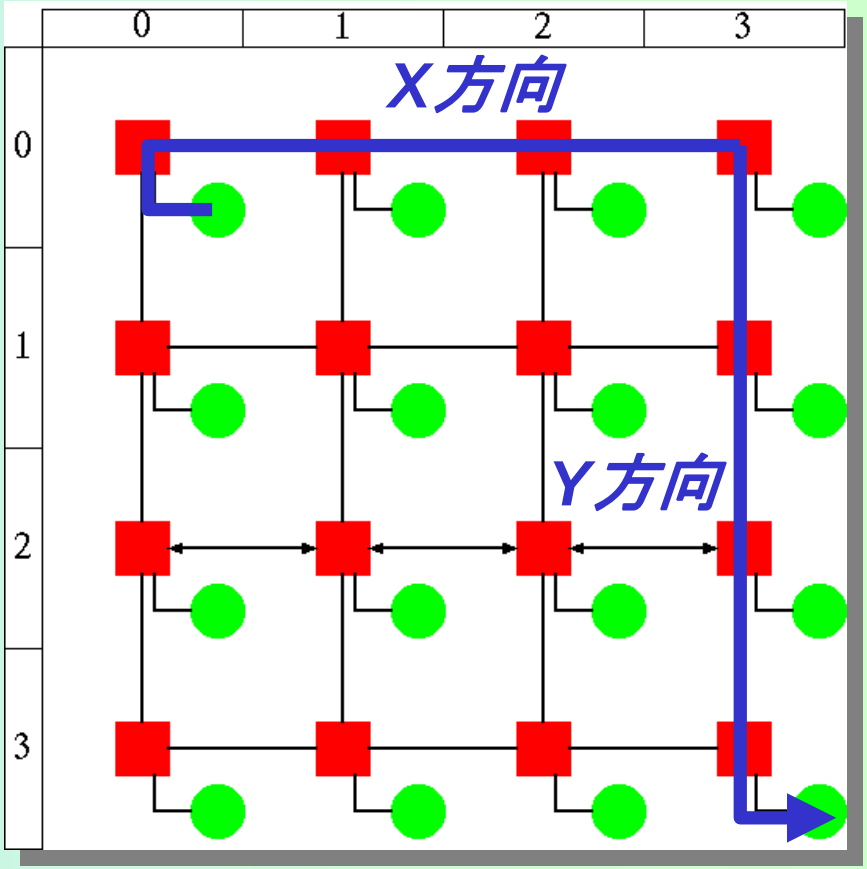






ルーティング：固定型ルーティング

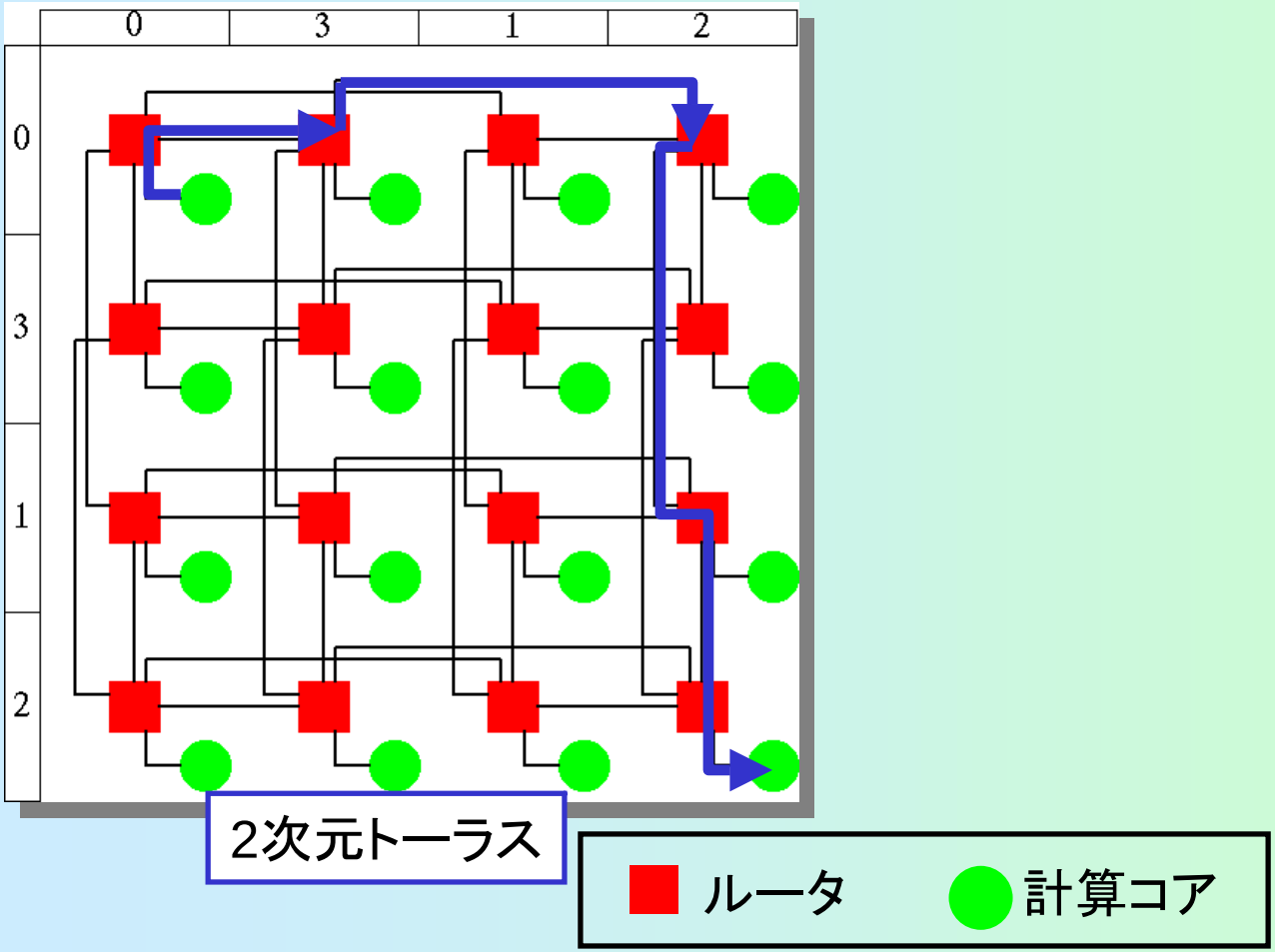
- 固定型ルーティング
  - Source-destination 間の経路は1つに固定
- ランダム型ルーティング
  - Source-destination 間に複数の経路
  - ランダムに1つを選択
- 適応型ルーティング
  - Source-destination 間に複数の経路
  - 混雑に応じて1つを選択



例) 次元順ルーティング

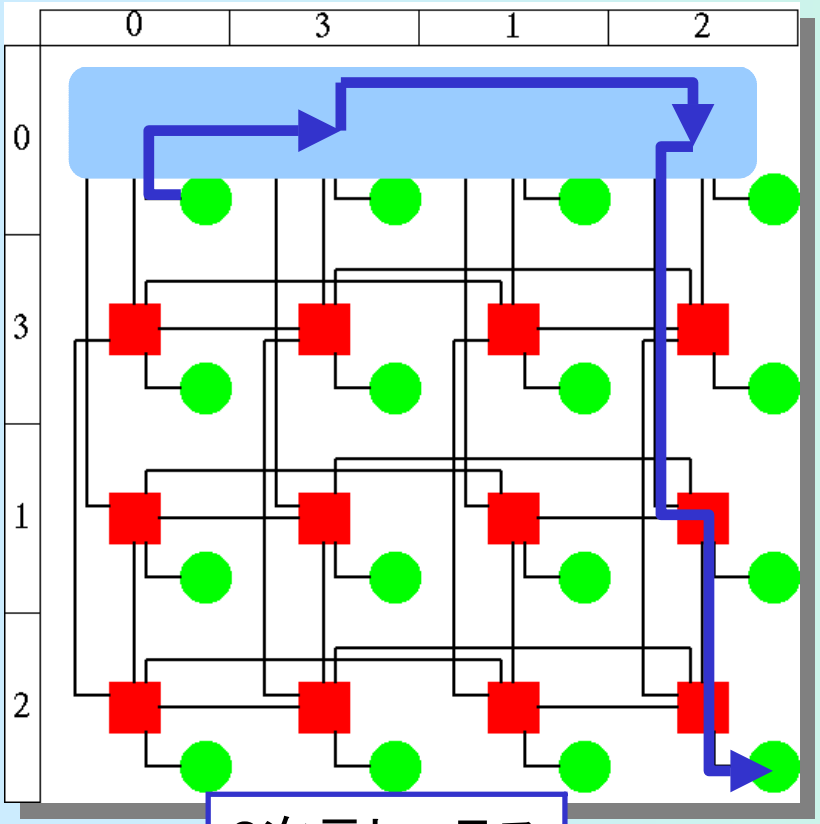
ルーティング：次元順ルーティング (トーラス)

- 次元順ルーティング
  - X方向 → Y方向
  - 仮想チャネルが必要

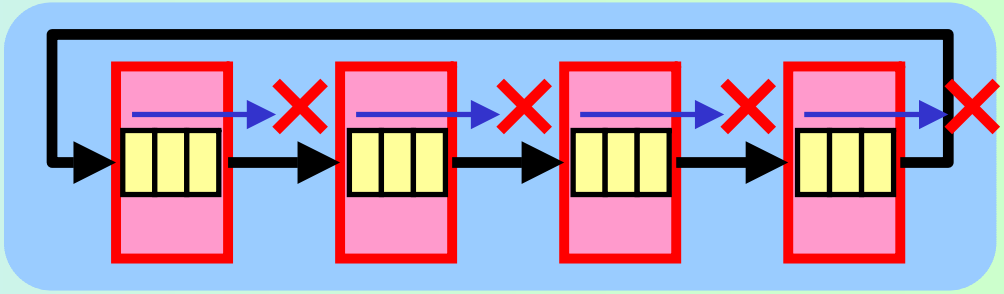


ルーティング：次元順ルーティング (トーラス)

- 次元順ルーティング
  - X方向 → Y方向
  - 仮想チャネルが必要

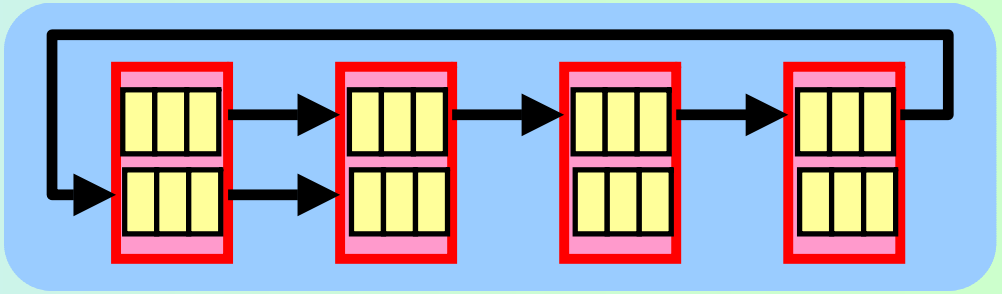


2次元トーラス



循環依存(サイクル)が発生 → デッドロック

バッファを多重化
(仮想チャネル)

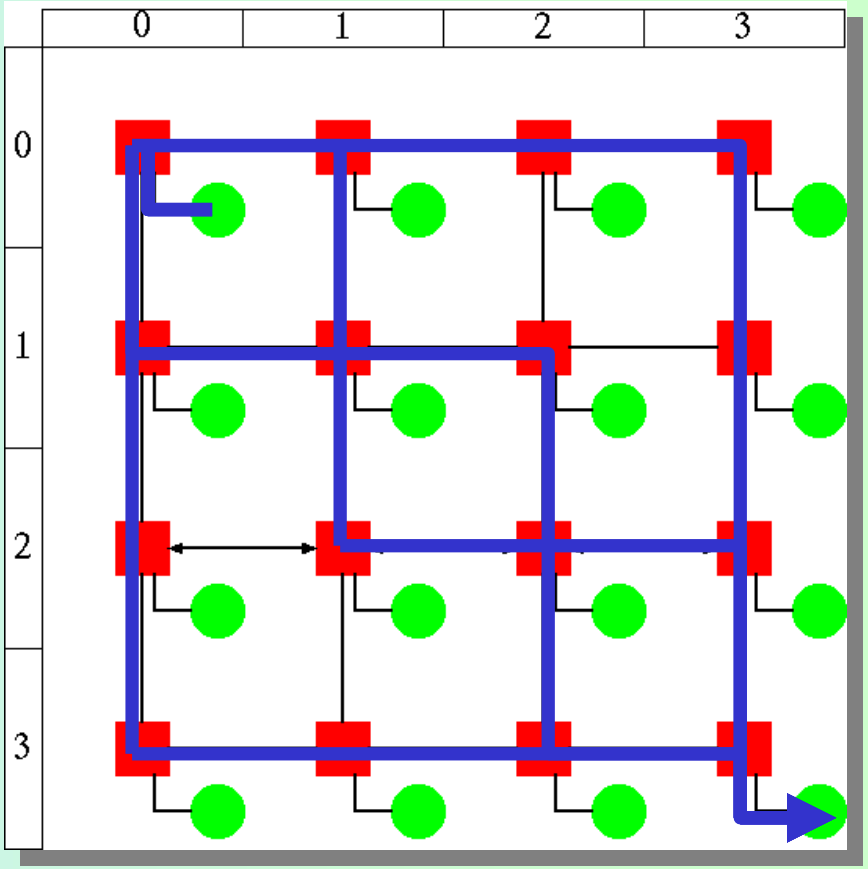


循環依存を断ち切る → デッドロックフリー

ルータ　計算コア

ルーティング：適応型ルーティング

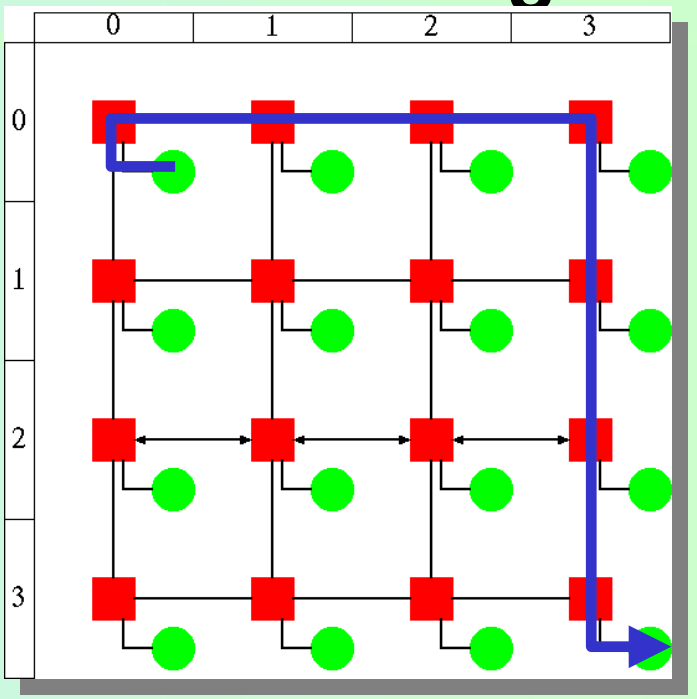
- 固定型ルーティング
  - Source-destination 間の経路は1つに固定
- ランダム型ルーティング
  - Source-destination 間に複数の経路
  - ランダムに1つを選択
- 適応型ルーティング
  - Source-destination 間に複数の経路
  - 混雑に応じて1つを選択

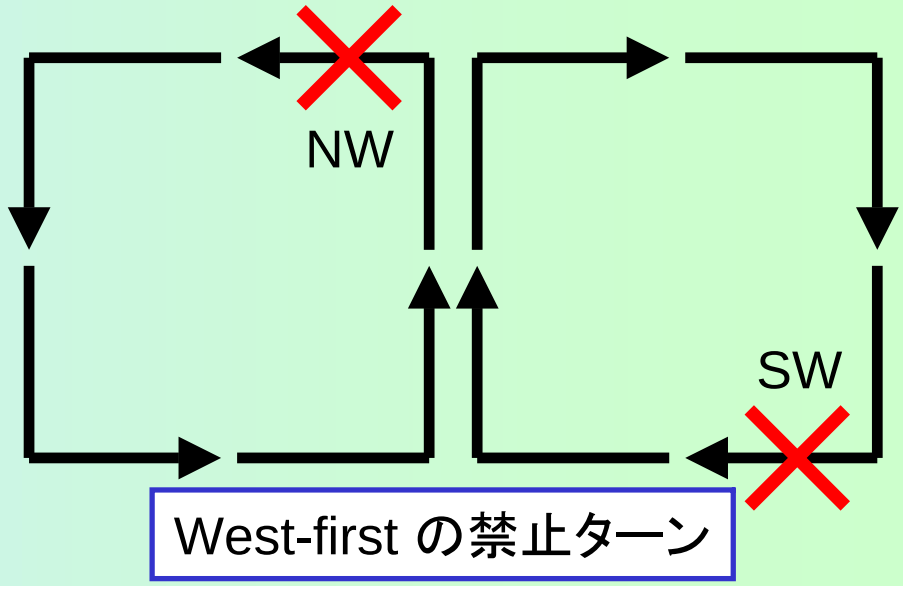


例) West-first, Negative-first, North-last, Odd-even, Opt-y, DP

ルーティング：West-first routing

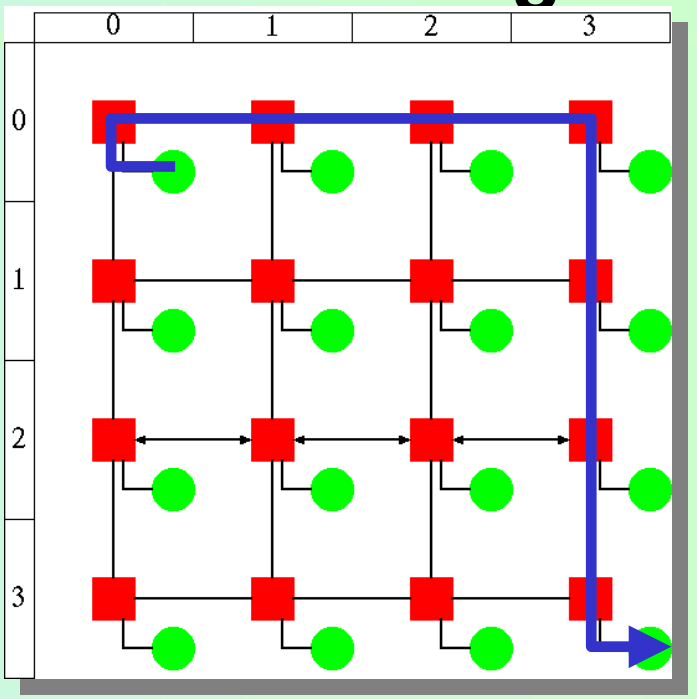
- 固定型ルーティング
  - Source-destination 間の経路は1つに固定
- ランダム型ルーティング
  - Source-destination 間に複数の経路
  - ランダムに1つを選択
- 適応型ルーティング
  - Source-destination 間に複数の経路
  - 混雑に応じて1つを選択

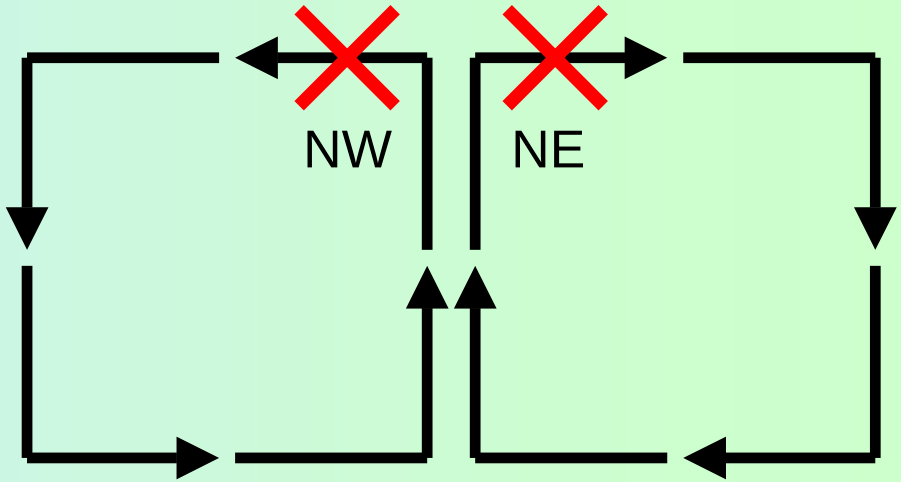




ルーティング：North-last routing

- 固定型ルーティング
  - Source-destination 間の経路は1つに固定
- ランダム型ルーティング
  - Source-destination 間に複数の経路
  - ランダムに1つを選択
- 適応型ルーティング
  - Source-destination 間に複数の経路
  - 混雑に応じて1つを選択

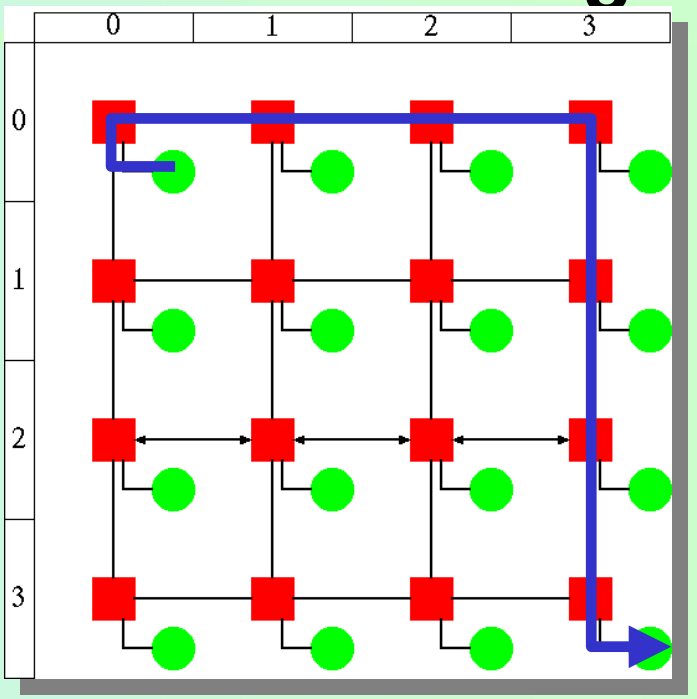


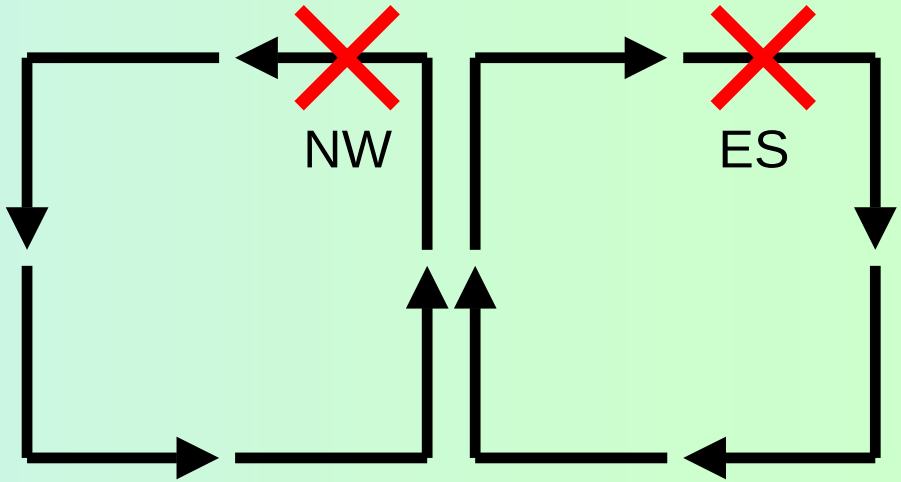


North-last の禁止ターン

ルーティング：Negative-first routing

- 固定型ルーティング
  - Source-destination 間の経路は1つに固定
- ランダム型ルーティング
  - Source-destination 間に複数の経路
  - ランダムに1つを選択
- 適応型ルーティング
  - Source-destination 間に複数の経路
  - 混雑に応じて1つを選択



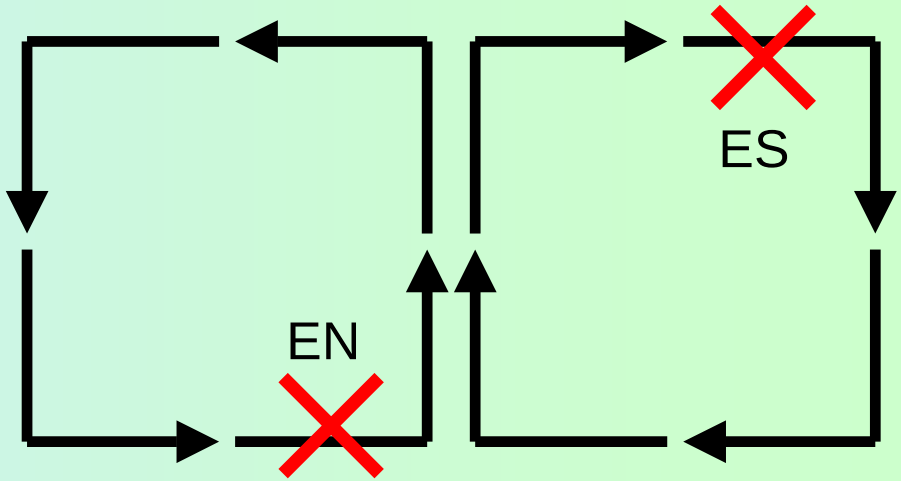


Negative-first の禁止ターン

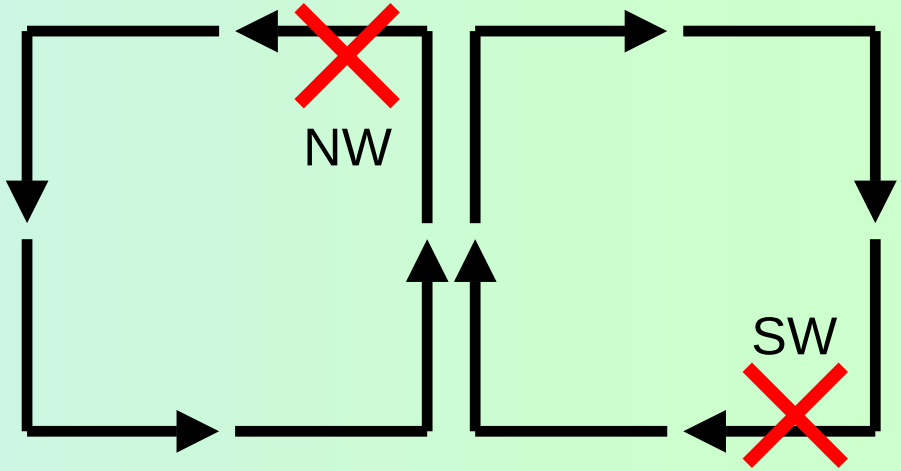
ルーティング：Odd-even turn-model

- 固定型ルーティング
  - Source-destination 間の経路は1つに固定
- ランダム型ルーティング
  - Source-destination 間に複数の経路
  - ランダムに1つを選択
- 適応型ルーティング
  - Source-destination 間に複数の経路
  - 混雑に応じて1つを選択

偶数列か奇数列かによって禁止ターン違う



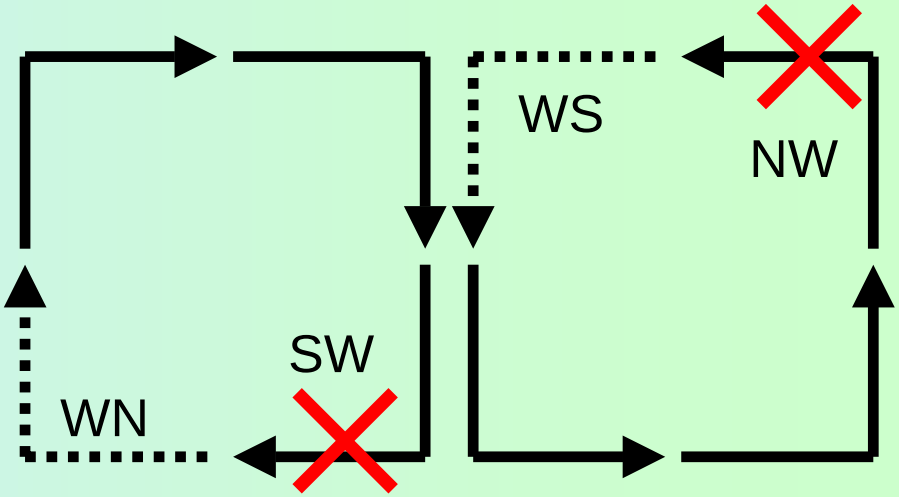
Odd-even (偶数列) の禁止ターン



Odd-even (奇数列) の禁止ターン

ルーティング: Opt-y routing (1/3)

- 固定型ルーティング
  - Source-destination 間の経路は1つに固定
- ランダム型ルーティング
  - Source-destination 間に複数の経路
  - ランダムに1つを選択
- 適応型ルーティング
  - Source-destination 間に複数の経路
  - 混雑に応じて1つを選択
- Fully adaptive routing
  - 仮想チャネル (VC)を用い，任意のターンを許可
  - NS方向に VC 2本

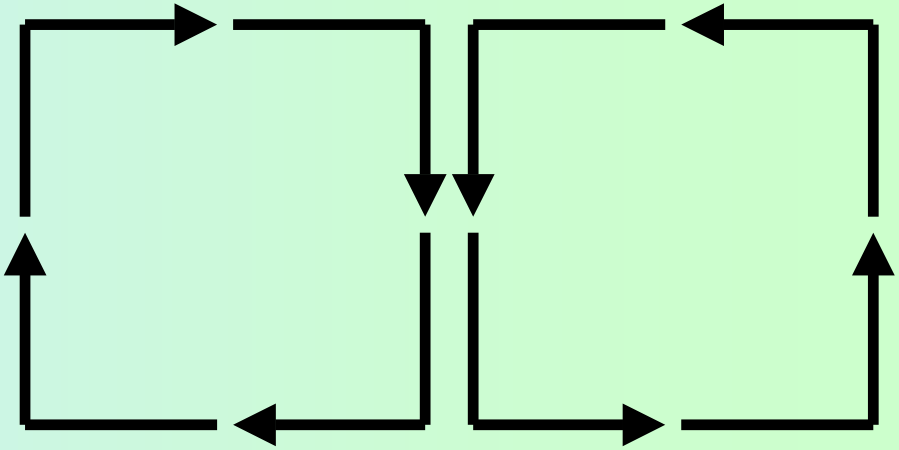


NS方向に仮想チャネル0を使う場合

(※) 点線のターンは「これ以上 West 方向に進まないとき」のみ許可

ルーティング: Opt-y routing (2/3)

- 固定型ルーティング
  - Source-destination 間の経路は1つに固定
- ランダム型ルーティング
  - Source-destination 間に複数の経路
  - ランダムに1つを選択
- 適応型ルーティング
  - Source-destination 間に複数の経路
  - 混雑に応じて1つを選択
- Fully adaptive routing
  - 仮想チャネル (VC)を用い，任意のターンを許可
  - NS方向に VC 2本

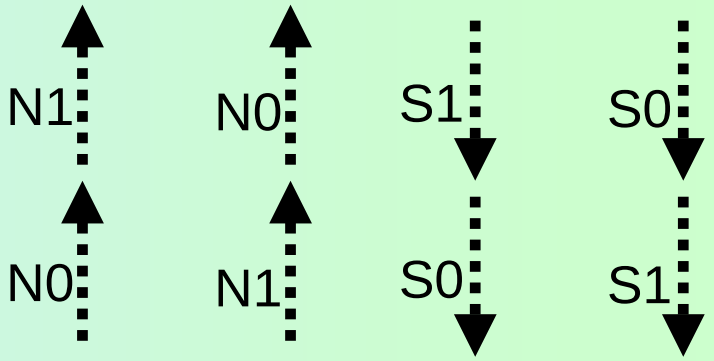


NS方向に仮想チャネル1を使う場合

(※) 点線のターンは「これ以上 West 方向に進まないとき」のみ許可

ルーティング：Opt-y routing (3/3)

- 固定型ルーティング
  - Source-destination 間の経路は1つに固定
- ランダム型ルーティング
  - Source-destination 間に複数の経路
  - ランダムに1つを選択
- 適応型ルーティング
  - Source-destination 間に複数の経路
  - 混雑に応じて1つを選択
- Fully adaptive routing
  - 仮想チャネル (VC)を用い，任意のターンを許可
  - NS方向に VC 2本



NS方向の仮想チャネル番号切替え

(※) 点線のターンは「これ以上 West 方向に進まないとき」のみ許可

最近のオンチップネットワーク

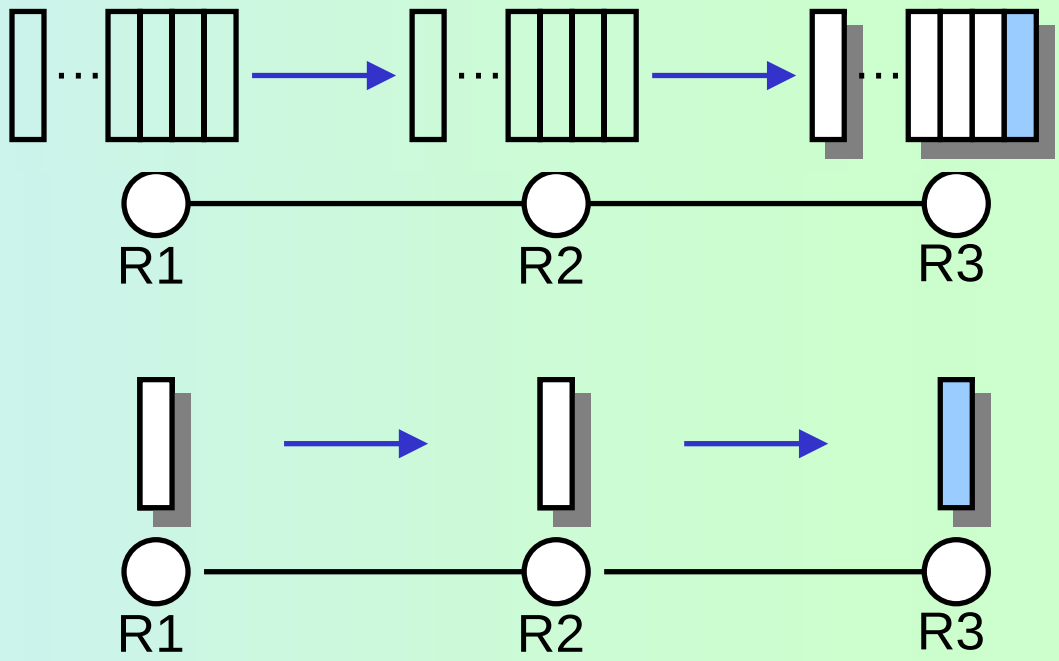
システム名	トポロジ	ルーティング	スイッチング	フロー制御
MIT RAW	2-D mesh (32bit)	XY DOR	WH, no VC	Credit
UPMC SPIN	Fat Tree (32bit)	Up*/down*	WH, no VC	Credit
QuickSilver ACM	H-Tree (32bit)	Up*/down*	1-flit, no VC	Credit
UMass Amherst aSOC	2-D mesh	Shortest-path	Pipelined CS, no VC	Timeslot
Sun T1	Crossbar (128bit)	-	-	Handshake
Cell BE EIB	Ring (128bit)	Shortest-path	Pipelined CS, no VC	Credit
TRIPS (operand)	2-D mesh (109bit)	YX DOR	1-flit, no VC	On/off
TI				
I				
-				

- パケットルーティングのまとめ
  - NoC では単純な固定型ルーティングが多く使われている
  - 非最短経路は消費電力(回路のスイッチング)の増加を招く
- ルーティングはトポロジによって(ある程度)決まる
  - Mesh/torus なら次元順ルーティング

パケットスイッチング：3種類の方法

- Store-and-forward (SAF)
  - パケット単位で転送
  - 大きなバッファ
  - T = D × (Lh + Lb)
- Wormhole (WH)
  - フリット単位で転送
  - 小さなバッファ
  - T = D × Lh + Lb
- Virtual-cut through
  - フリット単位で転送
  - 大きなバッファ

T=転送時間, D=ホップ数
Lh=ヘッダ長, Lb=ボディ長

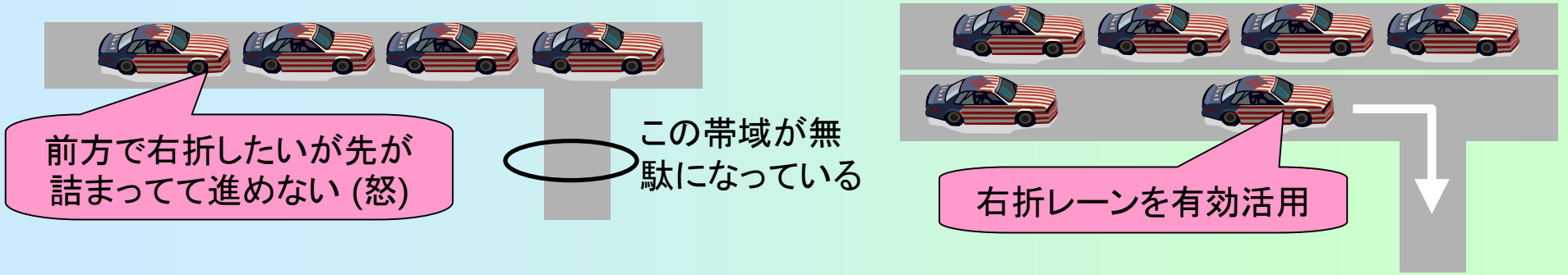


ホップ数3, パケット長 10-flit で, 1-flit 転送に1-clock かかるとき, 転送時間の合計は?

SAFの場合　3 * (1 + 10) = 33 clock
WHの場合　3 * 1 + 10 = 13 clock

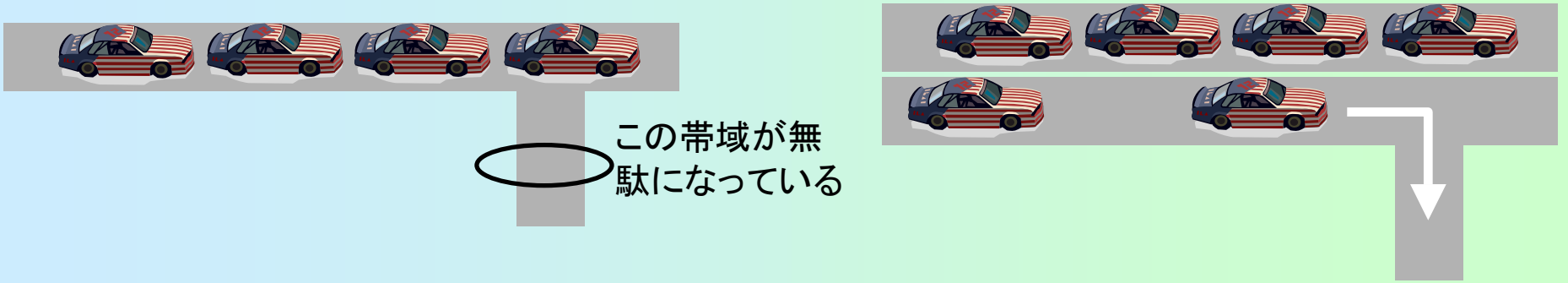
仮想チャネル(VC)機構

- VC の利点 – 先詰まりの防止 (1車線と2車線の道路の例)

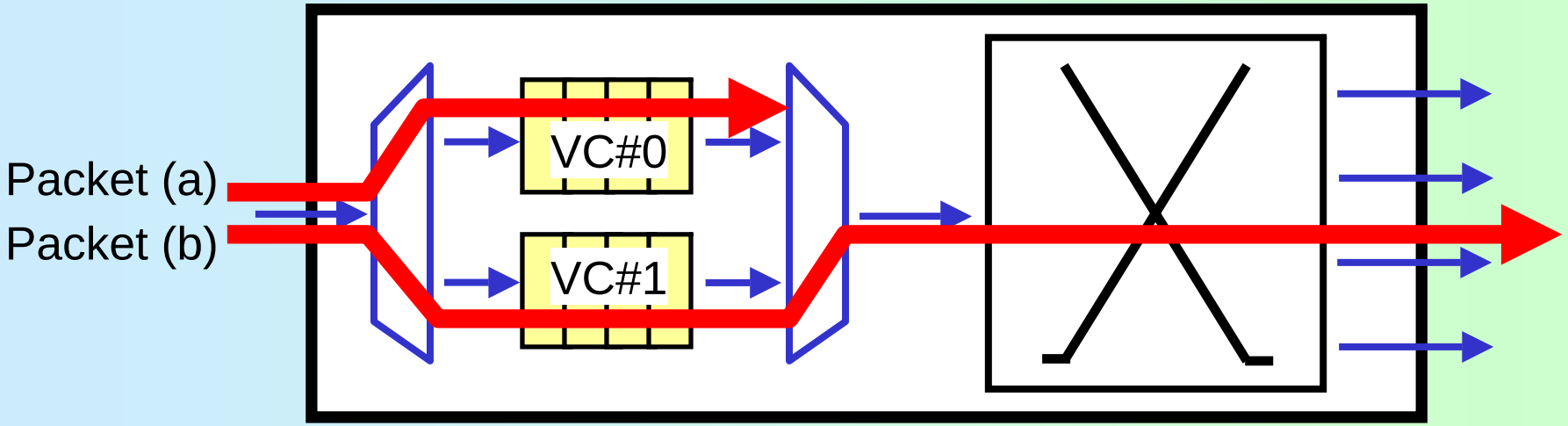


仮想チャネル(VC)機構

- VC の利点 – 先詰まりの防止 (1車線と2車線の道路の例)



- VC の実装 – 1物理ポートの時分割多重 [Dally,TPDS'92]



パケット(a)は先が詰まって進めないので, 先にパケット(b) がクロスバを通過

最近のオンチップネットワーク

システム名	トポロジ	ルーティング	スイッチング	フロー制御
MIT RAW	2-D mesh (32bit)	XY DOR	WH, no VC	Credit
UPMC SPIN	Fat Tree (32bit)	Up*/down*	WH, no VC	Credit
QuickSilver ACM	H-Tree (32bit)	Up*/down*	1-flit, no VC	Credit
UMass Amherst aSOC	2-D mesh	Shortest-path	Pipelined CS, no VC	Timeslot
Sun T1	Crossbar (128bit)	-	-	Handshake
Cell BE EIB	Ring (128bit)	Shortest-path	Pipelined CS, no VC	Credit
TRIPS (operand)	2-D mesh (109bit)	YX DOR	1-flit, no VC	On/off
T				

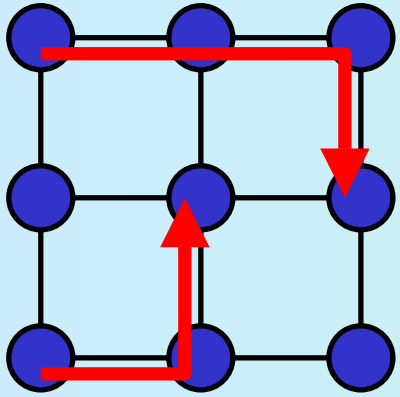
- パケットスイッチングのまとめ
  - NoC では Wormhole 方式が主流
- Wormhole 方式
  - バッファサイズが小さく, 通信遅延が小さい → NoC 向き
  - ただ, head-of-line blocking の影響を受ける →仮想チャネル

On-Chip Network Architecture

- いろいろなアプローチ
  - ソフトウェア レベル
  - アーキテクチャ レベル
  - 回路 レベル

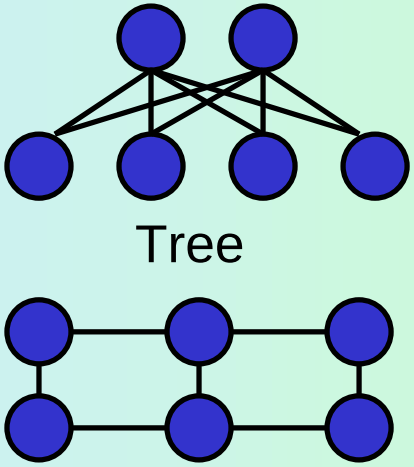
Software Level	OS, task scheduling
Architecture Level	Topology, routing, router architecture
Circuit Level	3D IC, power gating
Device Level	

- ネットワークアーキテクチャ

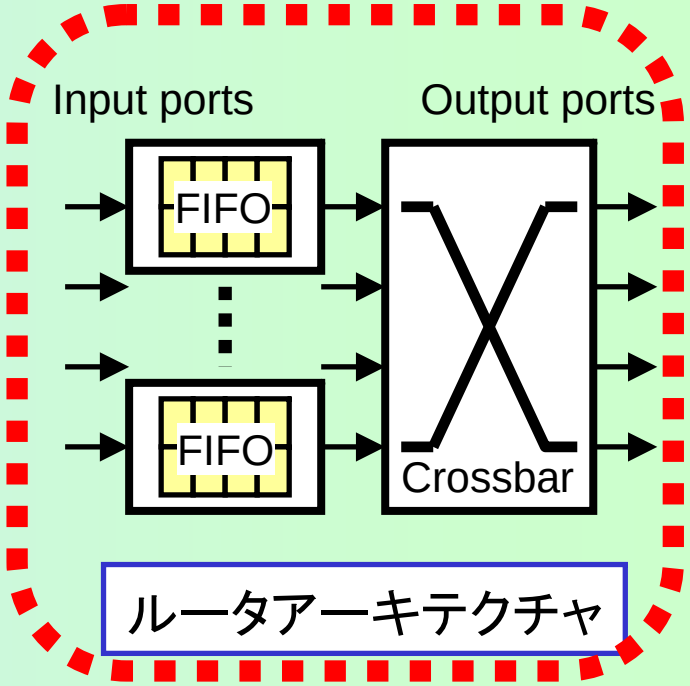


Deadlock-free routing

ルーティング,フロー制御



ネットワークトポロジ

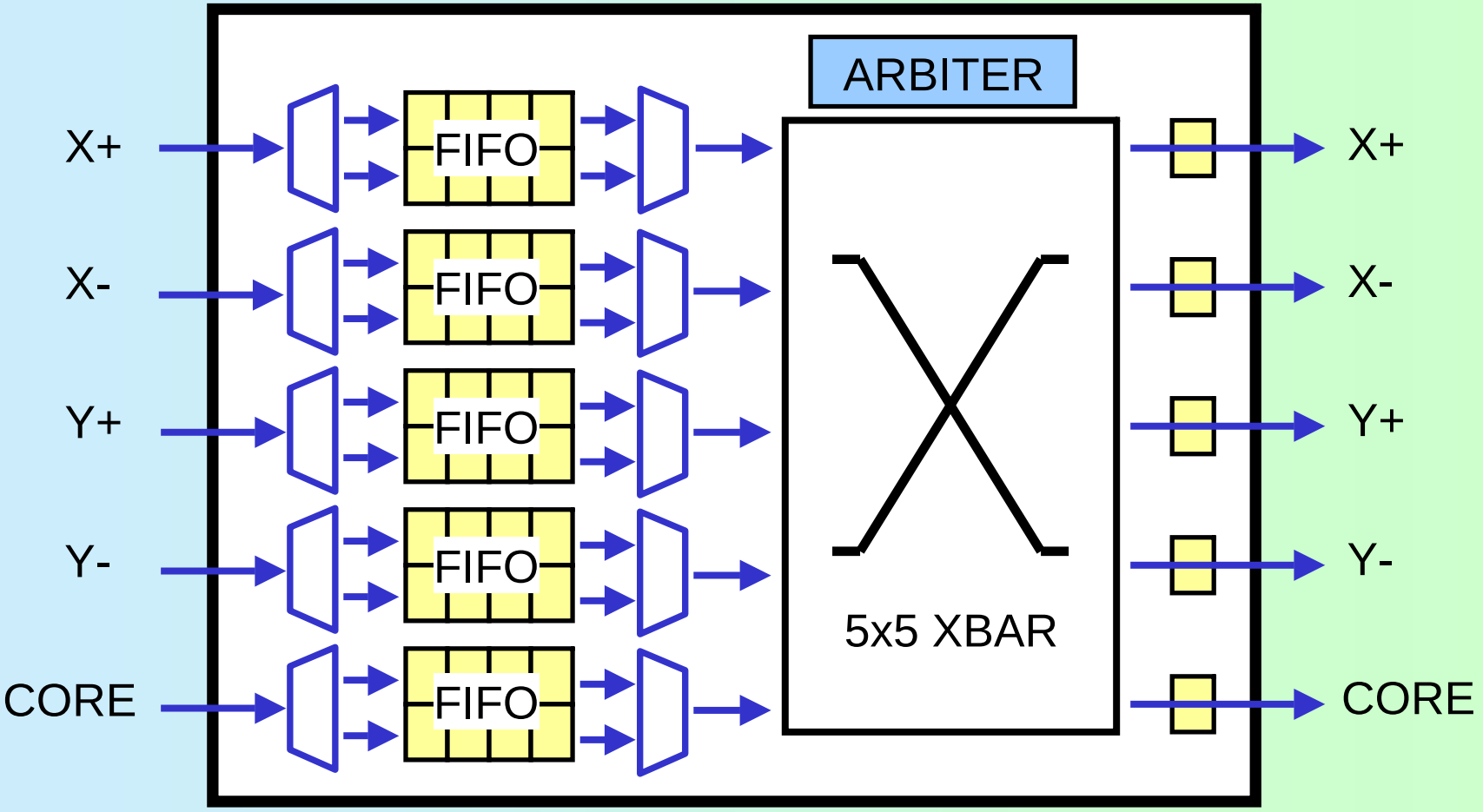


ルータアーキテクチャ

オンチップルータ: ハードウェア構成

- 5入力5出力の WH ルータ, データ(フリット)幅は 64-bit

1ポート当り複数の入力バッファ (この図では 2系統) を持つ → 仮想チャネル2本

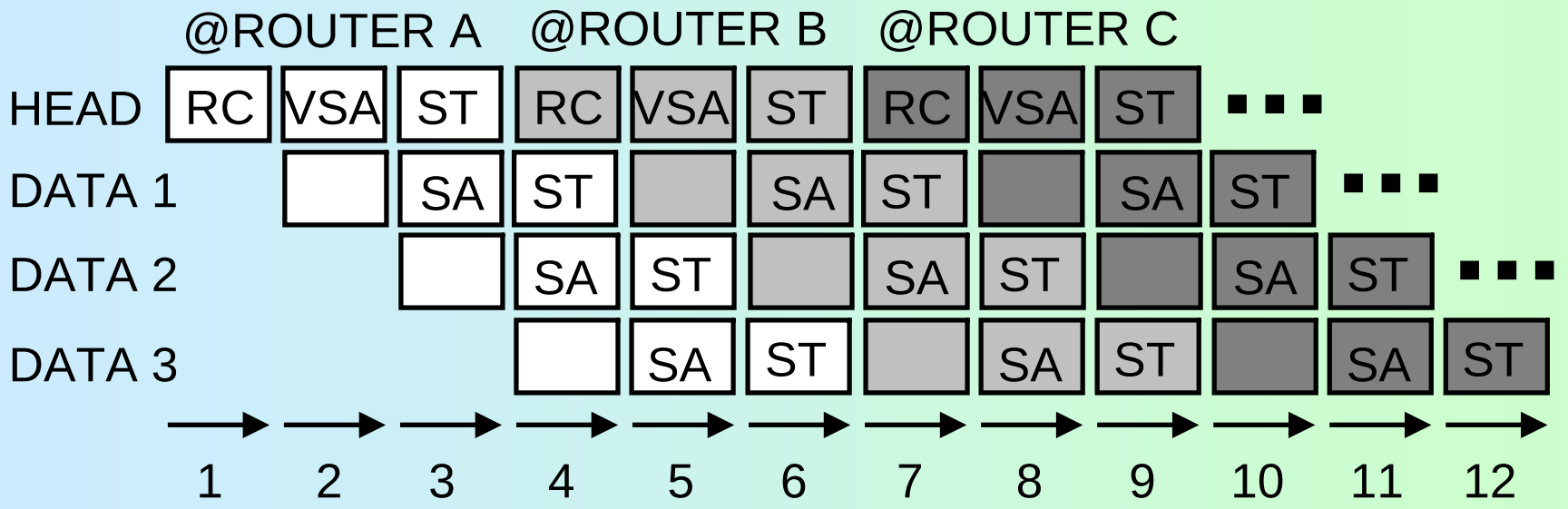


配置配線後のゲート数は 15～30 [kgates]で, 全体の6割が FIFO

オンチップルータ: パイプライン構造

- 衝突しなければ 3 cycle でヘッダがルータを通過
  - RC　(Routing computation)
  - VSA (Virtual channel / switch allocation)
  - ST　(Switch traversal)

- 例) ルータ(a) からルータ(c) にパケットを転送



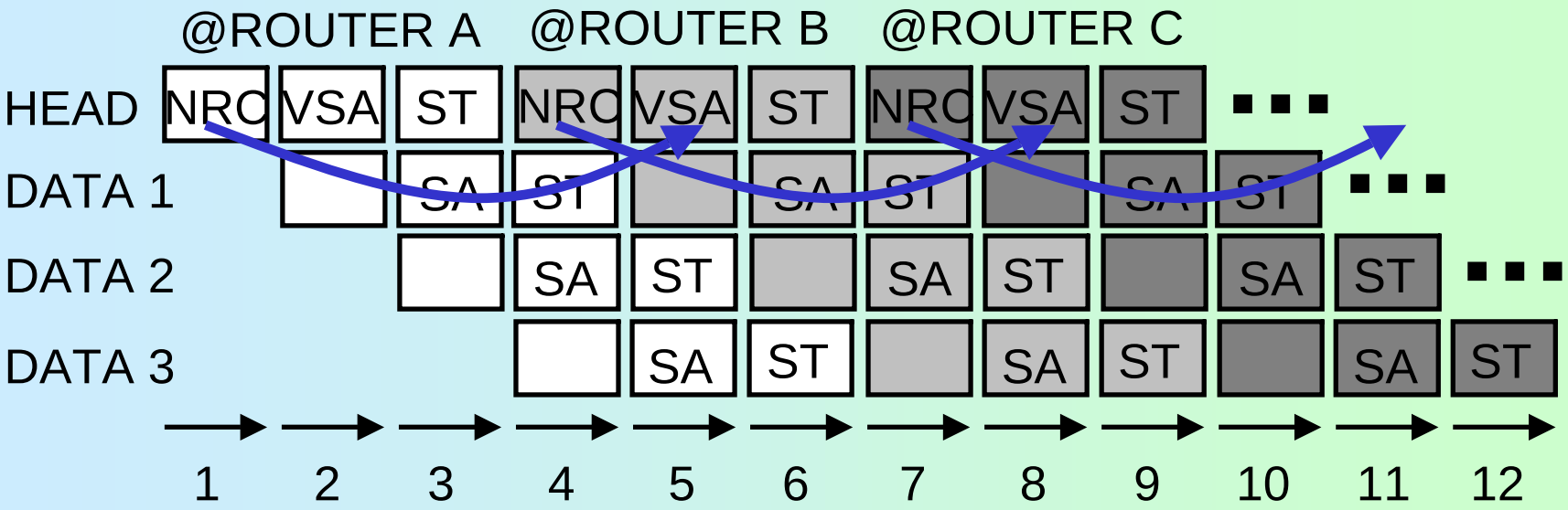
ヘッダがルータ(a)に注入され, データ3がルータ(c)を通過するまで12サイクル

オンチップルータ: Look-ahead 型ルータ

- 衝突しなければ 3 cycle でヘッダがルータを通過
  - NRC (Next routing computation)
  - VSA (Virtual channel / switch allocation)
  - ST　(Switch traversal)

NRCが終わらなくても VSAを実行できる

NRC: 次ルータのRCを実行 (自ルータのRCは手前のルータに任せる)

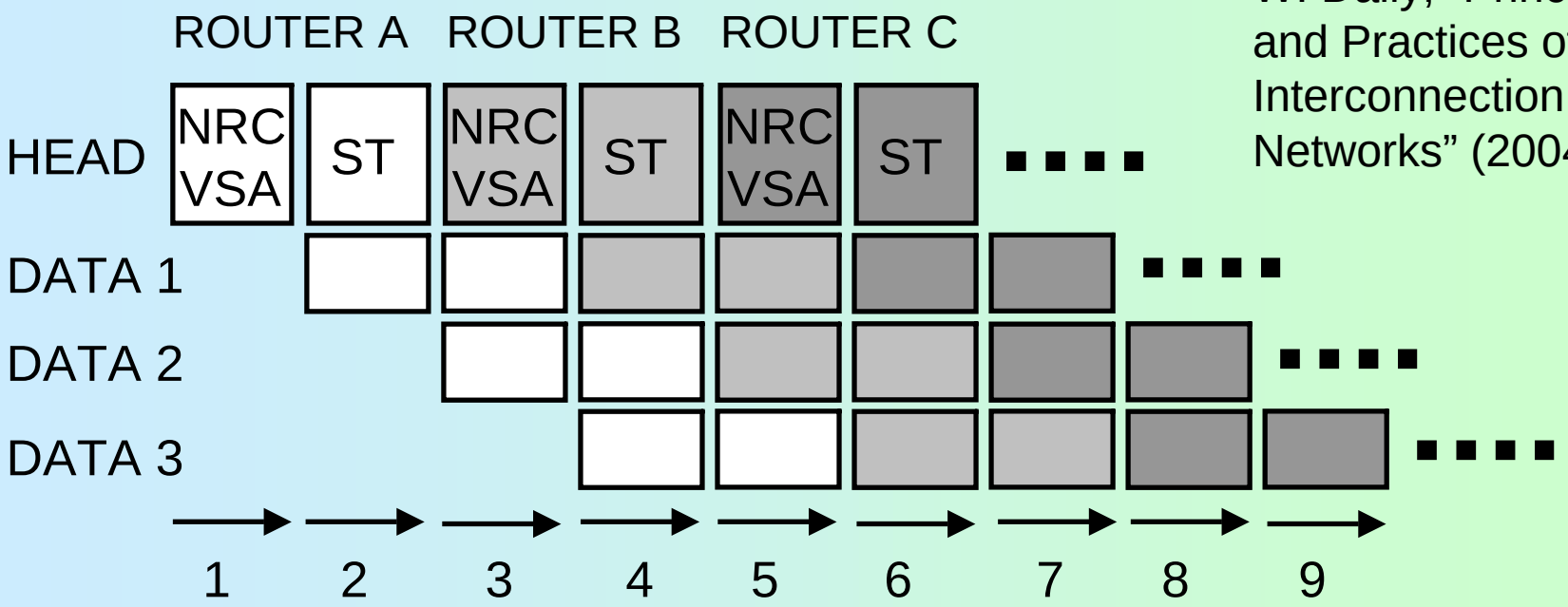


ルータ(b)の出力ポートはルータ(a)が決め, ルータ(c)の出力ポートはルータ(b)が…

オンチップルータ: 低遅延ルータ

- 衝突しなければ 2 cycle でヘッダがルータを通過
  - NRC + VSA (Next routing computation / switch allocation)
  - ST (Switch traversal)

NRC と VSA に依存性がないので並列実行できる → 2サイクル転送

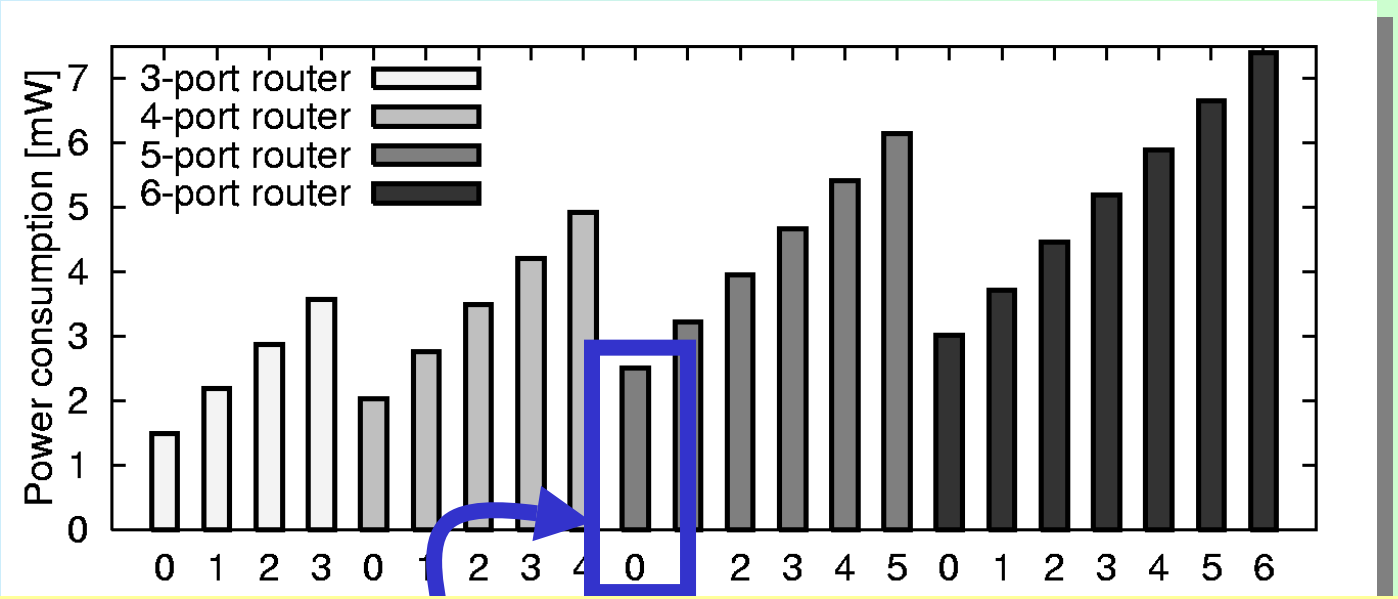


W. Dally, “Principles and Practices of Interconnection Networks” (2004)

ヘッダがルータ(a)に注入され, データ3がルータ(c)を通過するまで9サイクル

オンチップルルータ: 消費電力の解析

- 90nmで配置配線し, 200MHz でシミュレーション



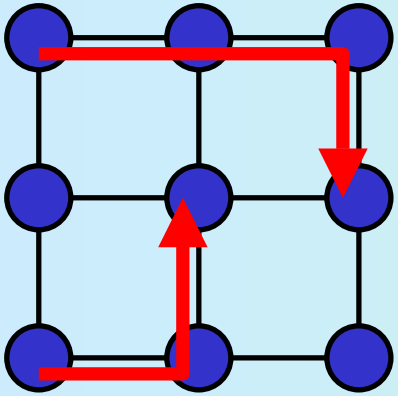
- ルータアーキテクチャのまとめ
  - 考慮すべきは, 面積(バッファ), パイプライン段数, 消費電力
  - パイプライン段数を減らす → 通信遅延が減る
- 消費電力の削減が重要
  - パケットを転送するときの電力 → 動作電圧を下げる
  - 常に漏れ出す(リーク)電流 → 電力供給自体を止める

On-Chip Network Architecture

- いろいろなアプローチ
  - ソフトウェア レベル
  - アーキテクチャ レベル
  - 回路 レベル

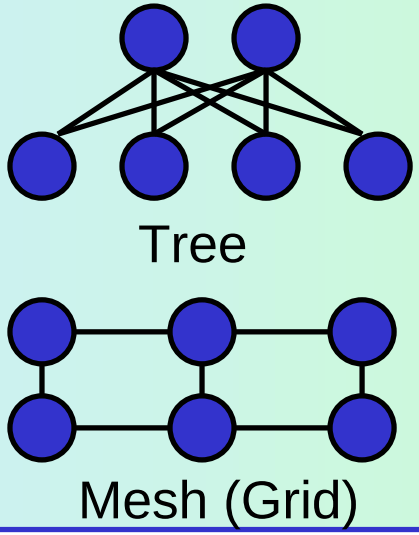
Software Level	OS, task scheduling
Architecture Level	Topology, routing, router architecture
Circuit Level	3D IC, power gating
Device Level	

- ネットワークアーキテクチャ

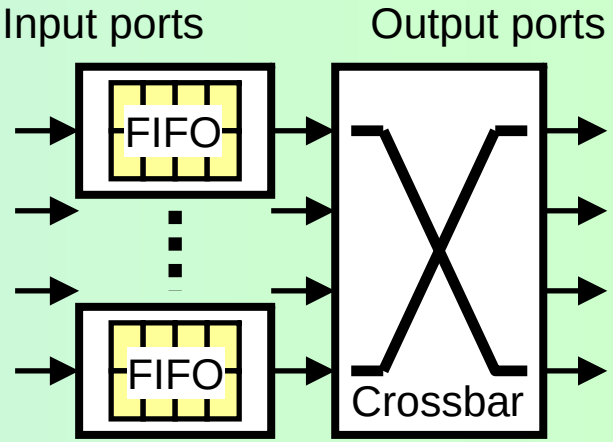


Deadlock-free routing

ルーティング,フロー制御



ネットワークトポロジ



ルータアーキテクチャ

参考書

- 結合網 (トポロジ,ルーティング,ルータ) 全般
  - W. Dally, et.al, “Principles and Practices of Interconnection Networks”, Morgan Kaufmann (2004)
  - J. Duato, et.al, “Interconnection Networks: An Engineering Approach”, Morgan Kaufmann (2003)

- NoC 関連
  - L. Benini, et.al, “Networks-on-Chips: Technology and Tools”, Morgan Kaufmann (2006)

発表の流れ

- Network-on-Chip (NoC) の概要
  - ネットワークトポロジ
  - パケットルーティング
  - ルータアーキテクチャ

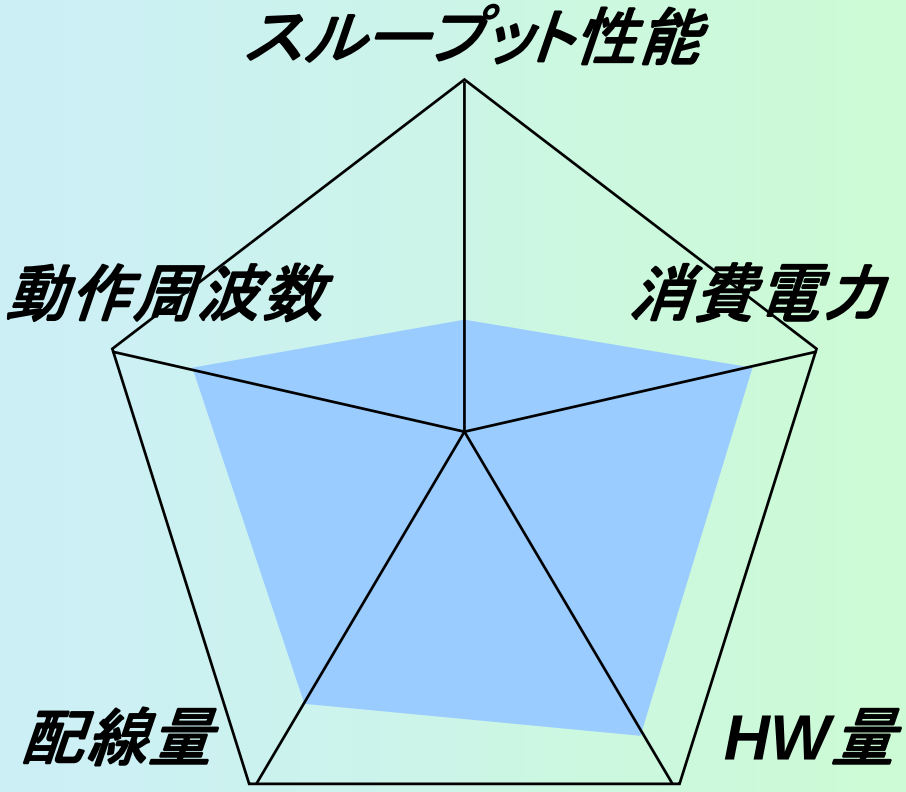
- NoC の研究の始め方
  - NoC シミュレータ
  - ルータ回路 (NoC generator)
  - NoC の評価方法

- NoC 研究の動向
  - 最近ホットなトピック
  - 予測機構による低遅延ルータ

[松谷,鯉渕,天野,吉永]

NoC の評価方法: 5つの評価項目

トレードオフの議論に落ち着くことが多い → 提案手法はできるだけ多角的に評価



評価には, C++ネットワークシミュレータ, Verilog のNoC 回路を使用

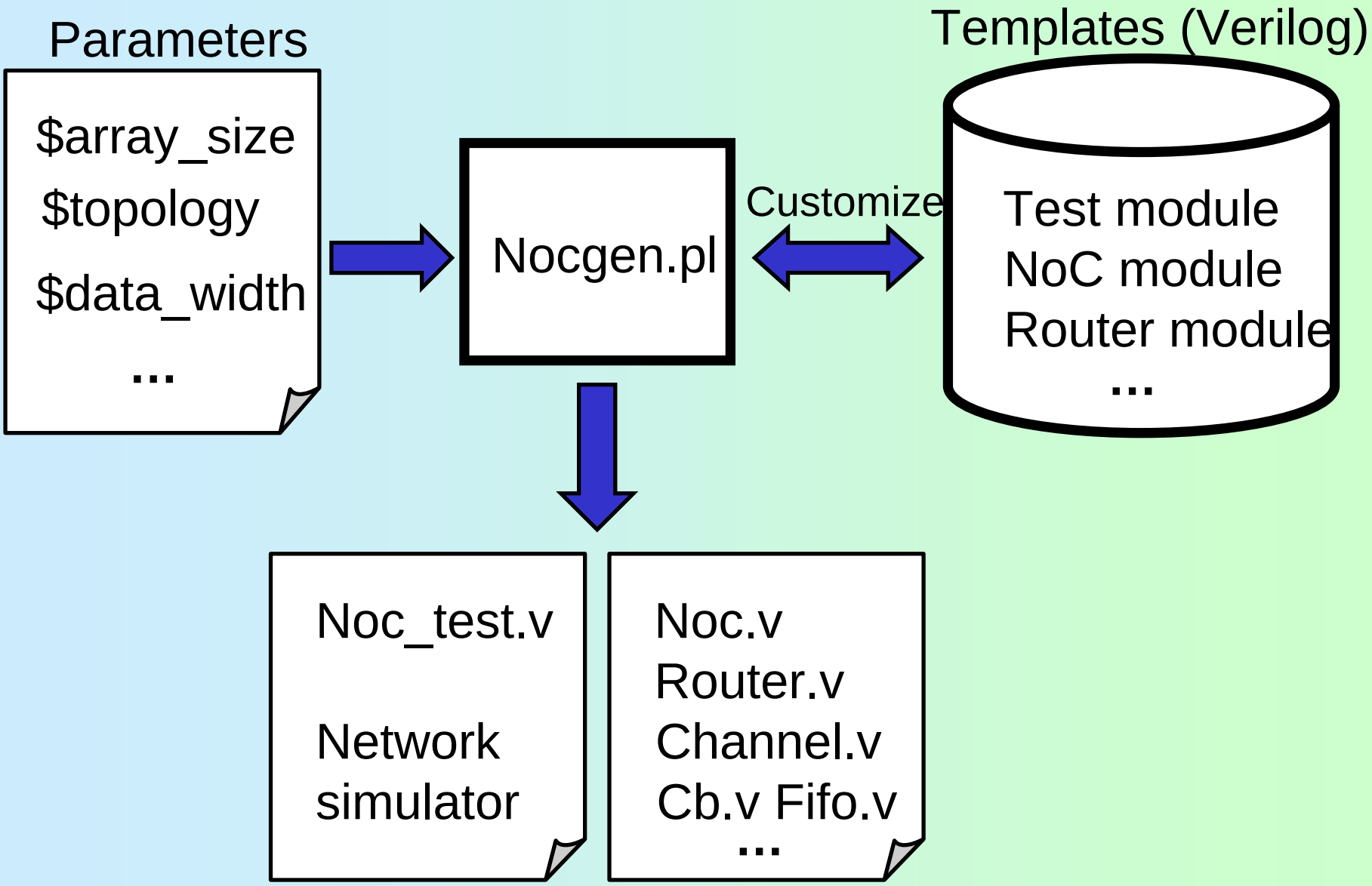
NoC シミュレータ

- スループット(Thr), 通信遅延(Lat), 電力(Pw)を測定できるもの

	開発言語	Thr	Lat	Pw	URL
Booksim	C++	○	○		http://cva.stanford.edu/books/ppin/
SICOSYS	C++	○	○		http://www.atc.unican.es/SICOSYS/
PoPnet	C++	○	○	○	http://www.princeton.edu/~peh/orion.html
irr_sim	C++	○	○	○	Amano lab.
Noxim	SystemC	○	○	○	http://noxim.sourceforge.net/
NoCsim	SystemC	○	○	○	http://sourceforge.net/projects/nocsim

(※) 電力見積り: アクション (ルータのデータ転送など) ごとの消費エネルギーをテーブルとして持ち, アクションの回数に応じてトータルの消費電力を見積もる

NoC generator: NoCのVerilogコード生成



NoC generator: 使い方

- NoC を生成する
  - ./nocgen.pl
- NoC を消去する
  - ./nocgen.pl clean
- パラメータを変更する
  - nocgen.pl を修正
  - 例えば,
  - $array_size = 8;
  - $topology_type = mesh;
  - $data_width = 64;
- $array_size
  - 1次元のノード数
- $topology_type
  - mesh, torus, spidergon, fat trees, h-tree, …
- $vch_num
  - 仮想チャネル数
- $data_width
  - リンクのデータ幅 (64-bit)
- $routing_type
  - XY routing (ecube), source routing, DP, …

NoC generator: 生成されるファイル群

- Define.h (65 lines)
  - データ幅などの定義
- Noc_test.v (2,796 lines)
  - テストプログラム
  - パケット衝突テスト
- Noc.v (1,932 lines)
  - ルータ同士を結合
- Router.v (763 lines)
  - ポート数分の入力チャネルとクロスバを結合
- Channel.v (321 lines)
  - 入力チャネル
  - 経路計算とバッファ機能
  - Rtcomp.v (61 lines)
  - Fifo.v (83 lines)
- Cb.v (691 lines)
  - クロスバスイッチ
  - Mux.v (124 lines)
  - Muxcont.v (121 lines)
  - クロスバ中の Muxcont がアービタに相当

NoC generator: シミュレーションの実行

- シミュレーション内容
  - パケット衝突テスト
  - Uniform traffic (右図)
  - 詳細は noc_test.v を参照
- シミュレーション結果
  - 送信フリット数の統計
  - 受信フリット数の統計
- 確認
  - 消失フリットは無いか？
  - 無ければ検証クリア

```
# (STEP * 200)
send_packet(1 → 0, 7-flit);
send_packet(2 → 0, 7-flit);
send_packet(3 → 0, 7-flit);
…
# (STEP * 200)
send_packet(0 → 1, 7-flit);
send_packet(2 → 1, 7-flit);
send_packet(3 → 1, 7-flit);
…
# (STEP * 200)
send_packet(0 → 2, 7-flit);
```

Verilog ソースコードの解説 (1)

Noc.v (1,932 lines)

ルータ同士を結合

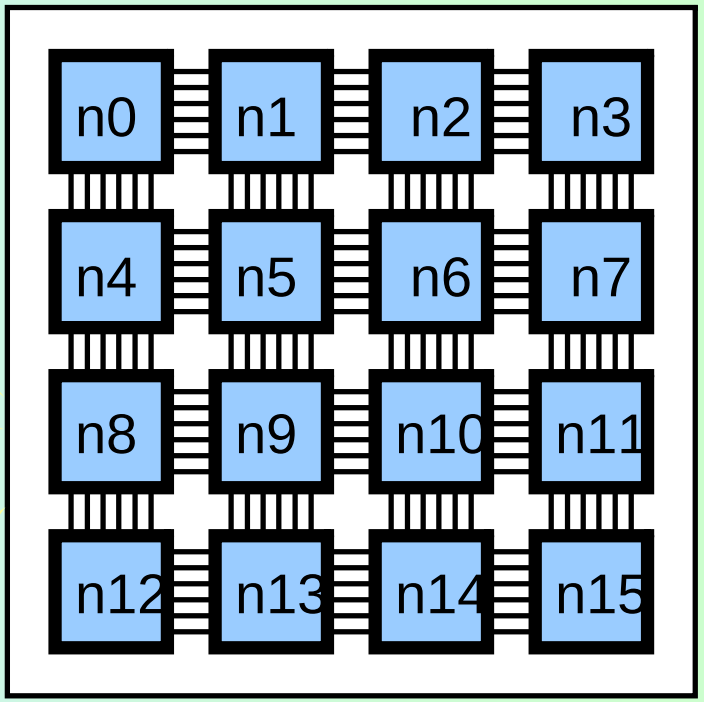
Noc_test.v (2,796 lines)

ネットワークシミュレータ

パケット衝突テスト

Define.h (65 lines)

データ幅などの定義



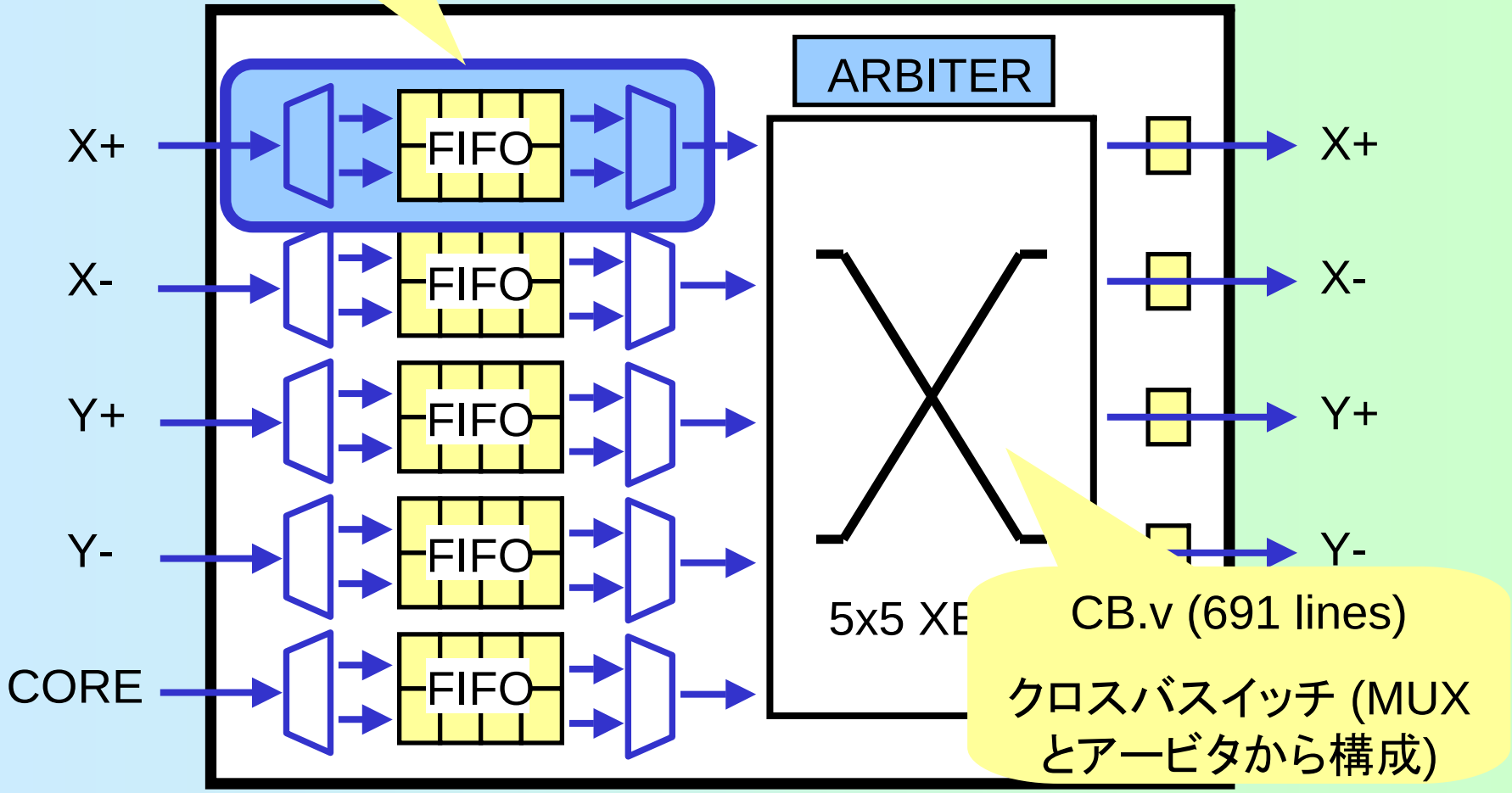
Verilog ソースコードの解説 (2)

Channel.v (321 lines)

入力バッファ, 経路計算

Router.v (763 lines)

ポート数分の入力チャネルとクロスバを接続

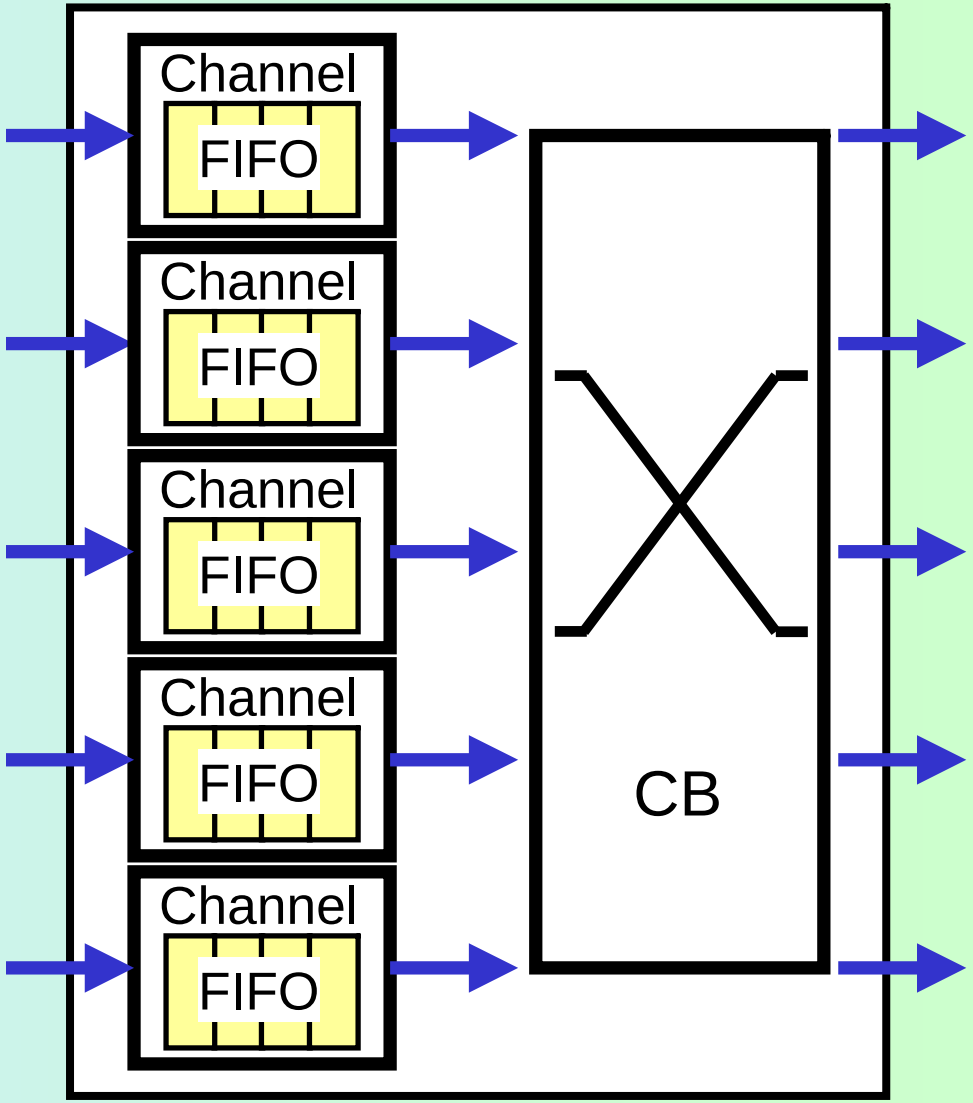


CB.v (691 lines)

クロスバスイッチ (MUXとアービタから構成)

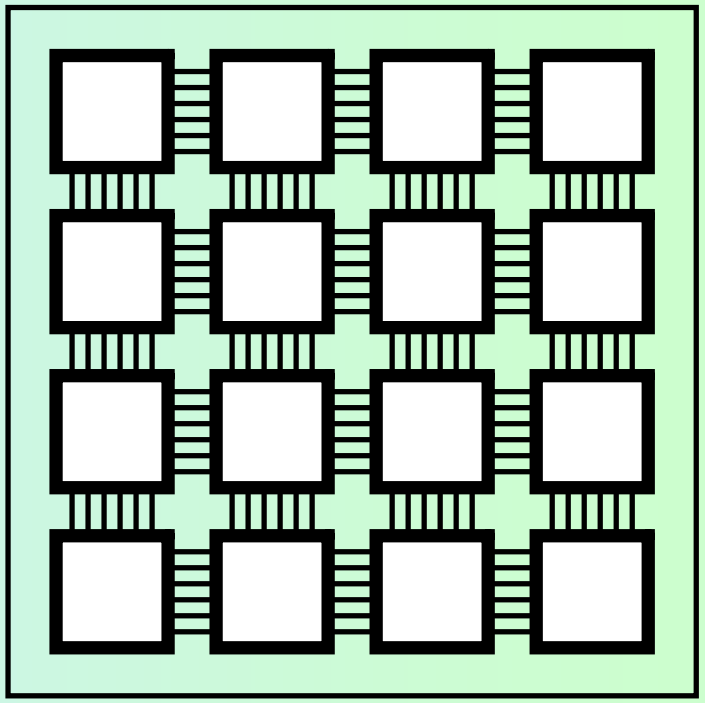
合成結果: OSU 180nm standard cell

- 5-port router (14 kgates)
  - Channel (16.35%)
  - Channel (16.35%)
  - Channel (16.35%)
  - Channel (16.35%)
  - Cb (18.18%)
- Channel (2.3 kgates)
  - Fifo (77.44%)
  - Misc (22.56%)
- Cb (2.6 kgates)
  - Mux * 5
  - Muxcont * 5



合成結果: OSU 180nm standard cell

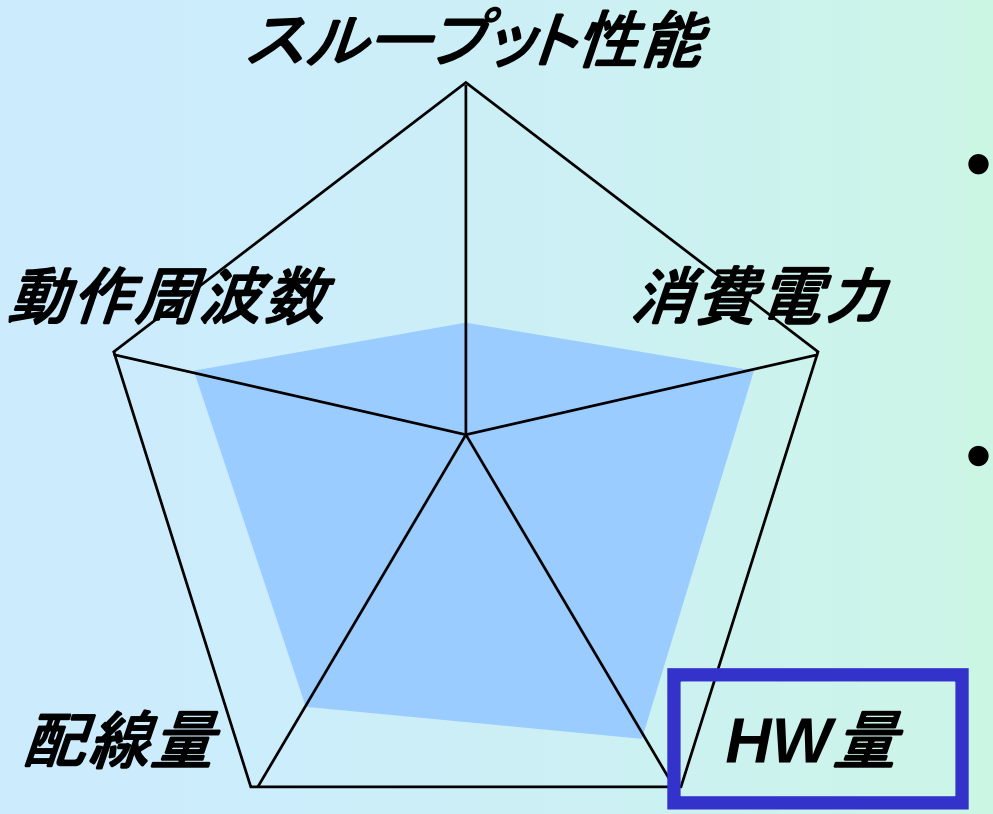
- 16-node NoC (176 kgates)
  - 3-port router (4.54%)
  - 4-port router (6.15%)
  - 4-port router (6.15%)
  - 3-port router (4.54%)
  - 4-port router (6.15%)
  - 5-port router (7.95%)
  - 5-port router (7.95%)
  - 4-port router (6.15%)
  - 4-port router (6.15%)
  - 5-port router (7.95%)
  - 5-port router (7.95%)
  - 4-port router (6.15%)
  - 3-port router (4.54%)
  - 4-port router (6.15%)
  - 4-port router (6.15%)
  - 3-port router (4.54%)



合成後の遅延は 3.08 [nsec]

最大動作周波数 324 [MHz]

NoC の評価項目: ハードウェア量



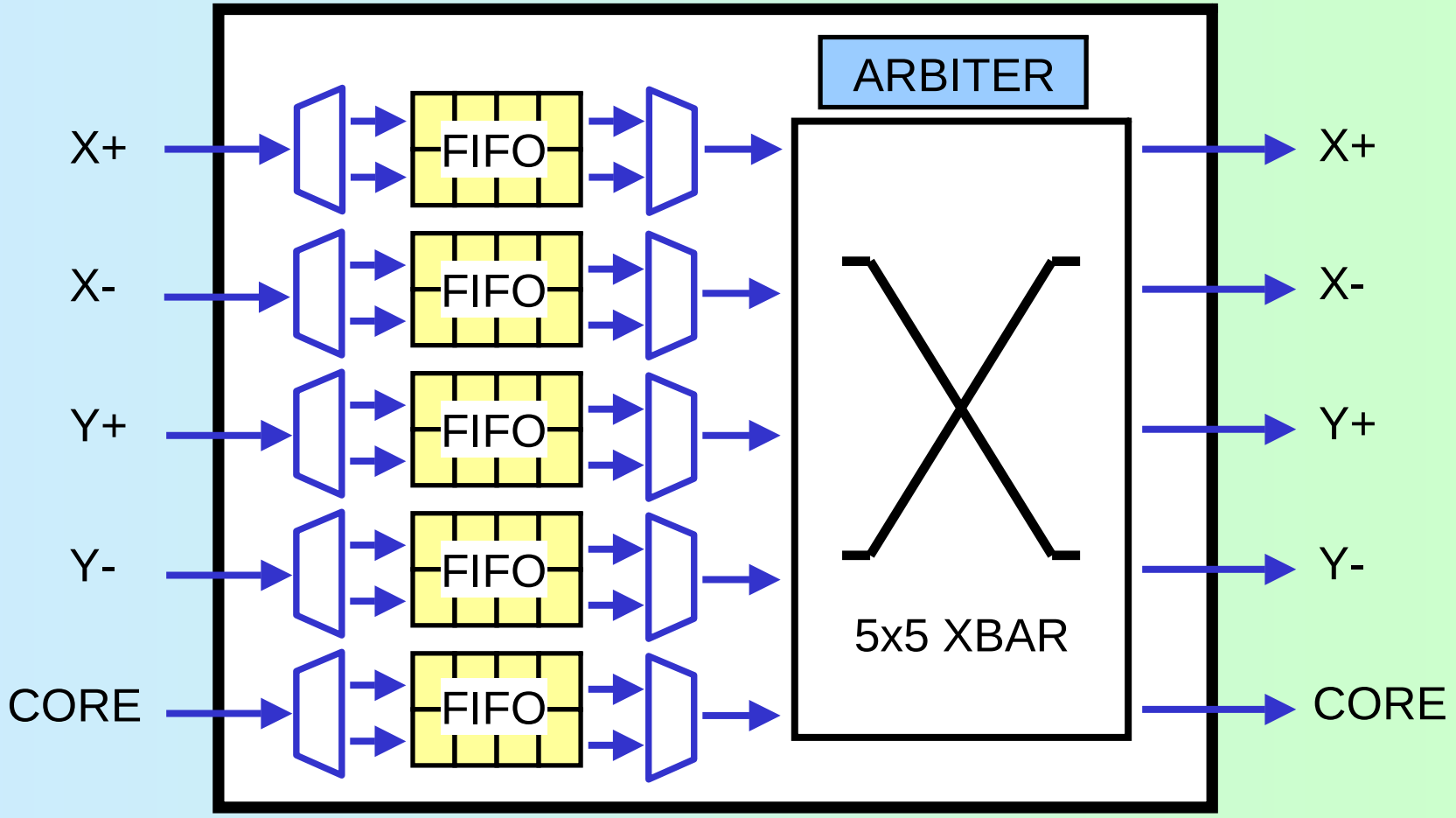
- ルータ回路を実装
  - Verilog-HDL で記述
  - NC Verilog で動作検証
- 合成
  - 90nm CMOS ライブラリ
  - Design Compiler で合成
- 配置配線
  - 90nm CMOS ライブラリ
  - Astro で配置配線

- 回路のゲート数を見積る

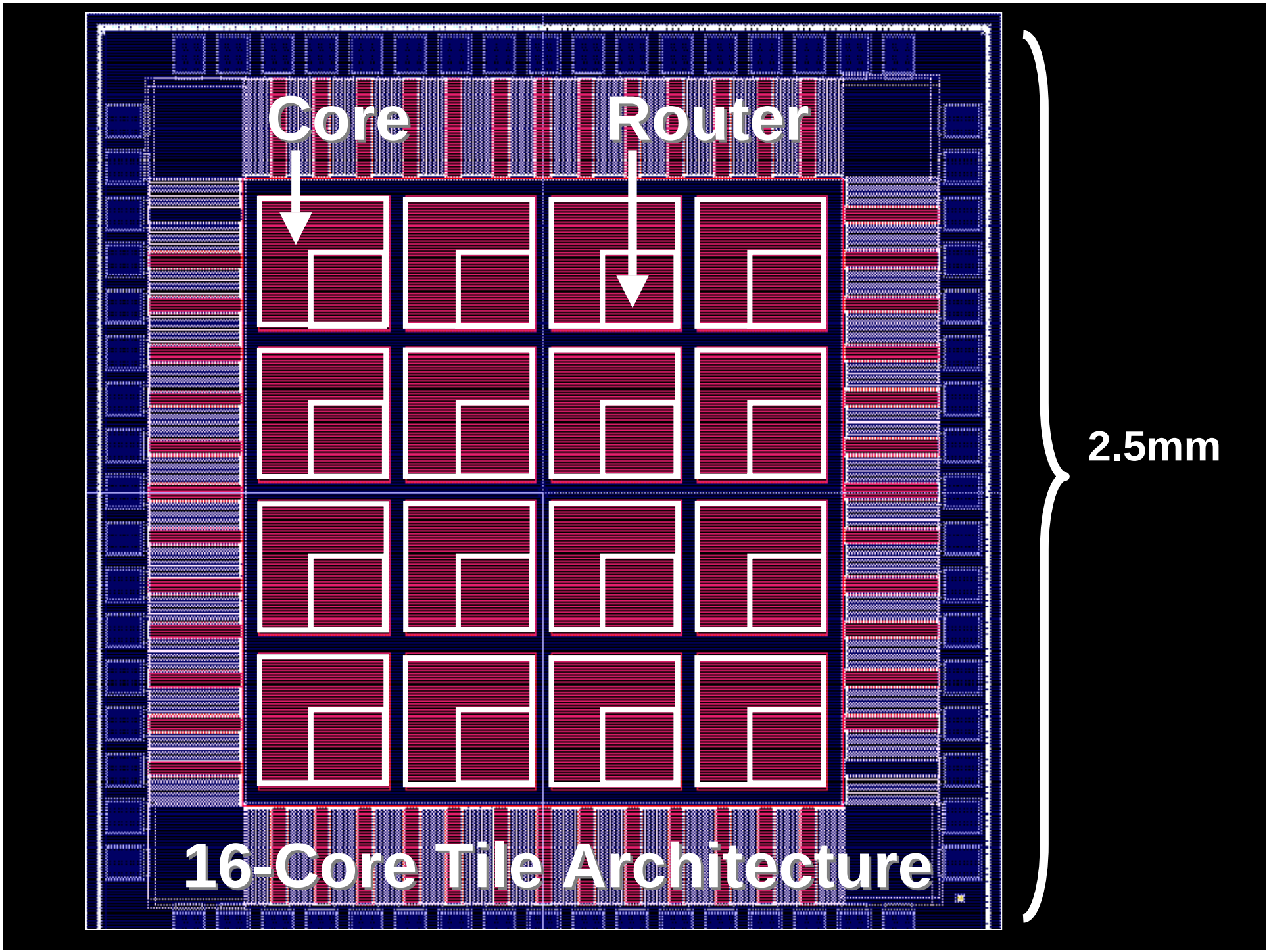
NoC の評価項目: ハードウェア量

- 5入力5出力の WH ルータ, データ(フリット)幅は 64-bit

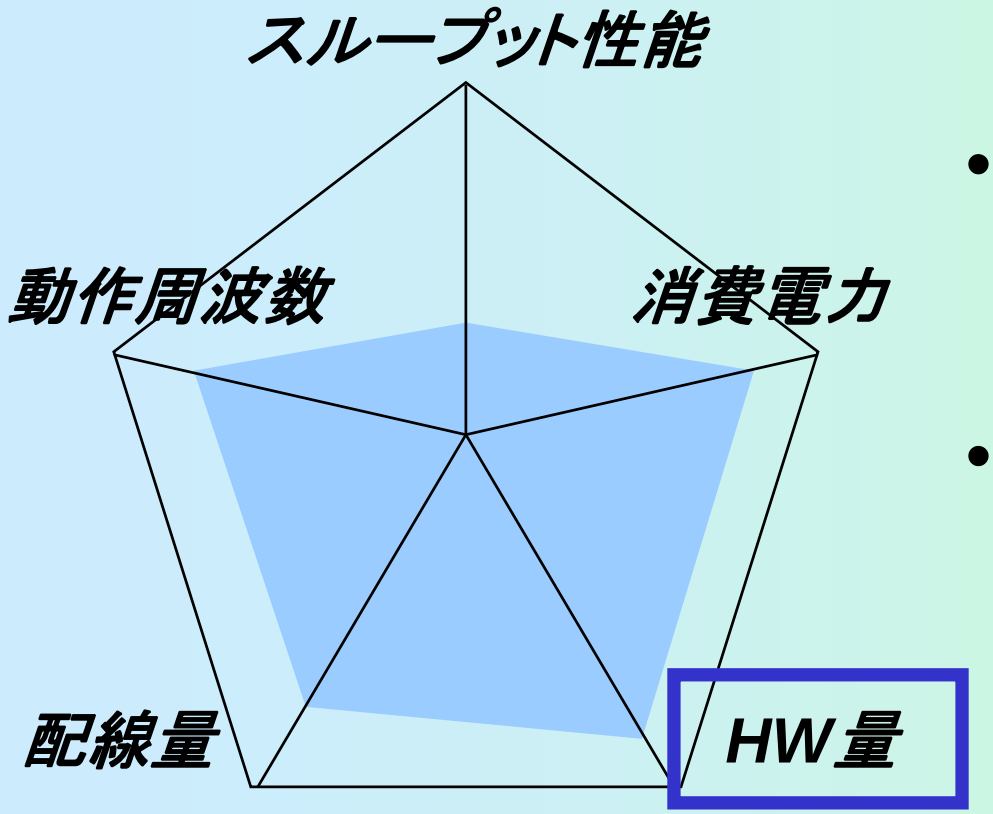
1ポート当り複数の入力バッファ (この図では 2系統) を持つ → 仮想チャネル2本



配置配線後のゲート数は 15～30 [kgates]で, 全体の6割が FIFO


Core
Router
2.5mm
16-Core Tile Architecture

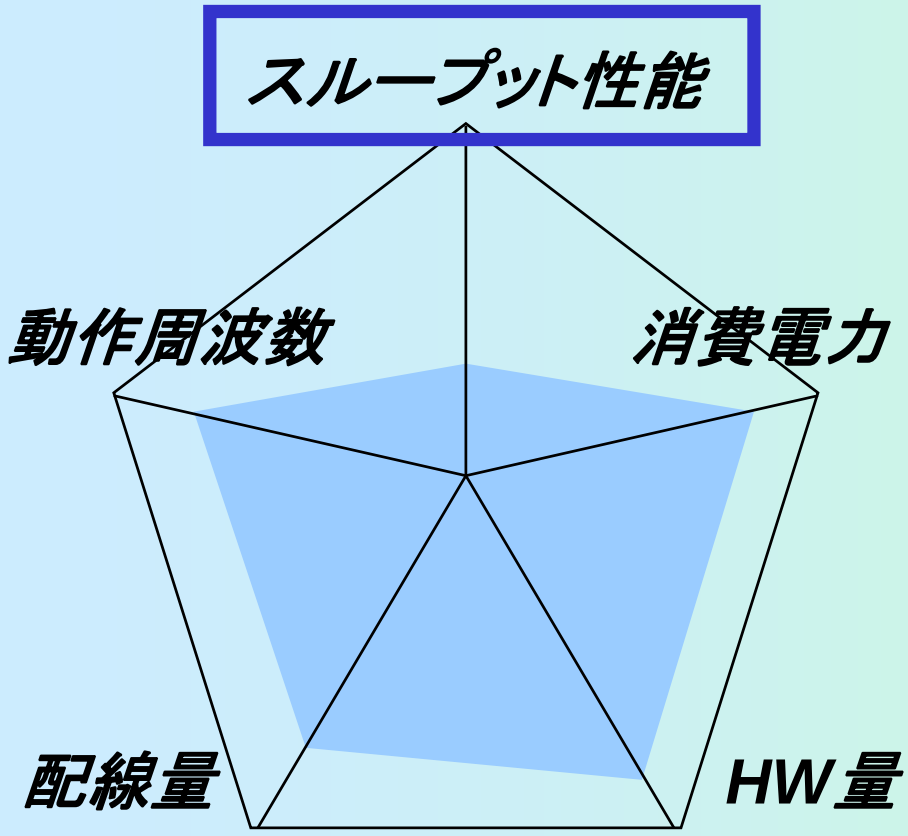
NoC の評価項目: ハードウェア量



- ルータ回路を実装
  - Verilog-HDL で記述
  - NC Verilog で動作検証
- 合成
  - 90nm CMOS ライブラリ
  - Design Compiler で合成
- 配置配線
  - 90nm CMOS ライブラリ
  - Astro で配置配線

- 回路のゲート数を見積る

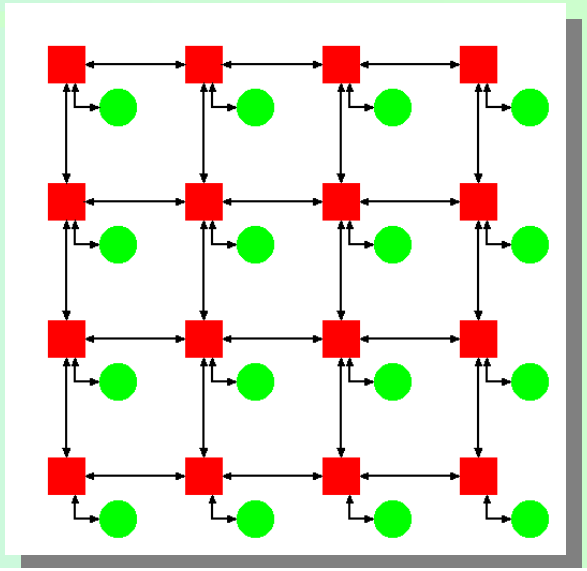
NoC の評価項目: スループット性能

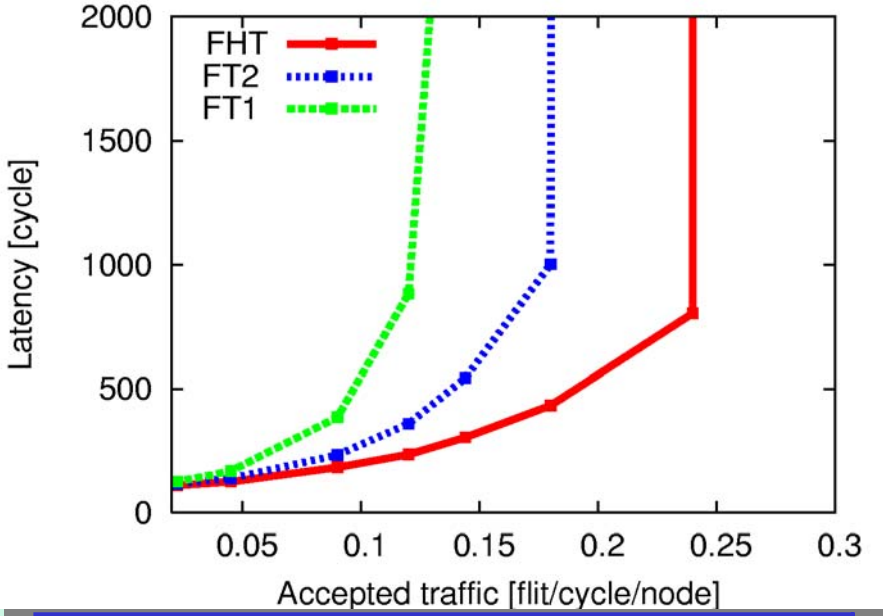


- ✕ RTL シミュレーション
  - 実ルータ回路を使用
  - 遅すぎる
- ○ ネットワークシミュレータ
  - Cycle accurate
  - 当研究室で長年開発
- トラフィックパターン
  - 良いベンチマークが無い
  - NAS Parallel Benchmark (NPB) からトレースを採取
- スループット (received flits /core/cycle)を見積る

NoC の評価項目: ネットワークシミュレータ

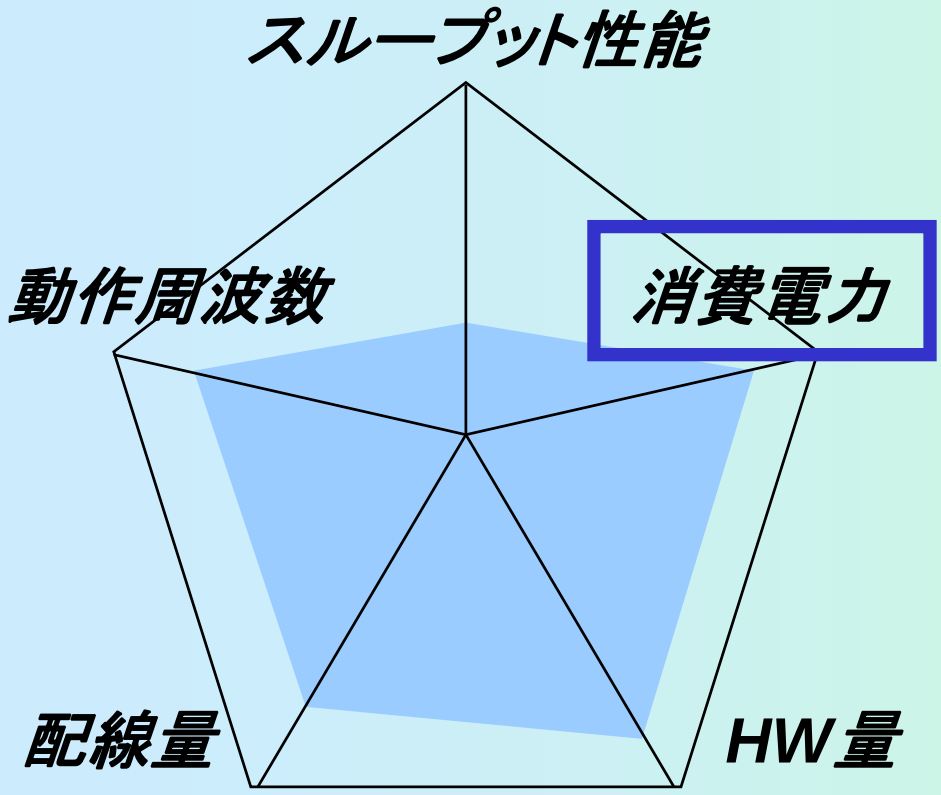
- ネットワーク構成
  - コア数 (16～256コア)
  - トポロジ (mesh, torus, trees)
- ルータの構造
  - スイッチング (WH, VCT)
  - 仮想チャネル数 (1～n本)
  - バッファサイズ (1～nフリット)
- パケット構成
  - ボディ長, ヘッダ長
- ルーティング
  - DOR, up*/down*, Turn
- 他, 必要に応じて適宜実装





各種トポロジのスループット [flit/core/cycle]

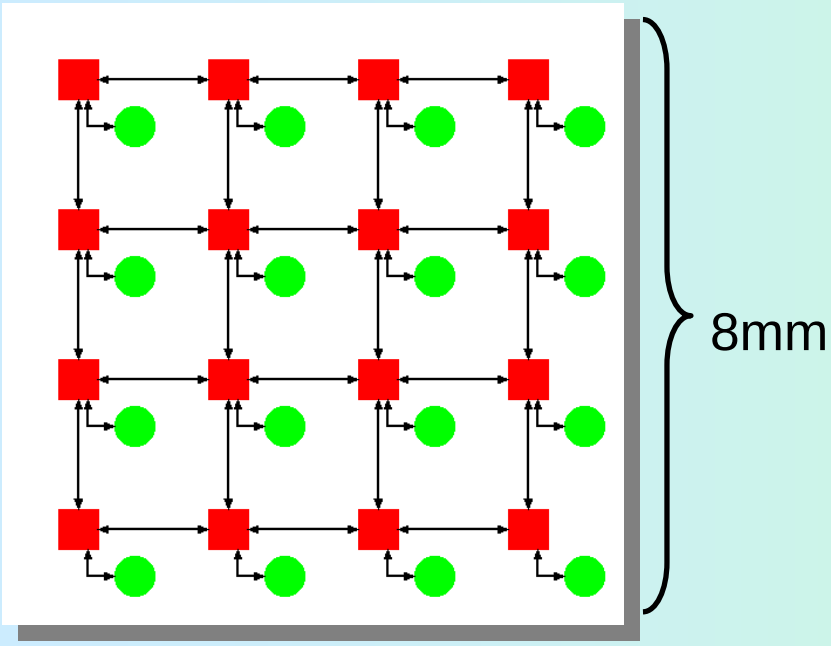
NoC の評価項目: 消費電力



- ルータ回路を実装
  - 90nm CMOS ライブラリ
  - Astro で配置配線
- 配置配線後シミュレーショ
  - NC Verilog
- 消費電力の解析
  - バックアノテーション
  - Power Compiler
- 評価結果
  - スイッチング電力
  - リーク電力
  - フリット転送エネルギー

NoC の評価項目：フリット転送エネルギー

- フリットエネルギー E_{flit}
  - 1-flit を宛先まで送る
  - 平均何ジュールかかる?



$$E_{flit} = w \cdot H_{ave}(E_{sw} + E_{link})$$

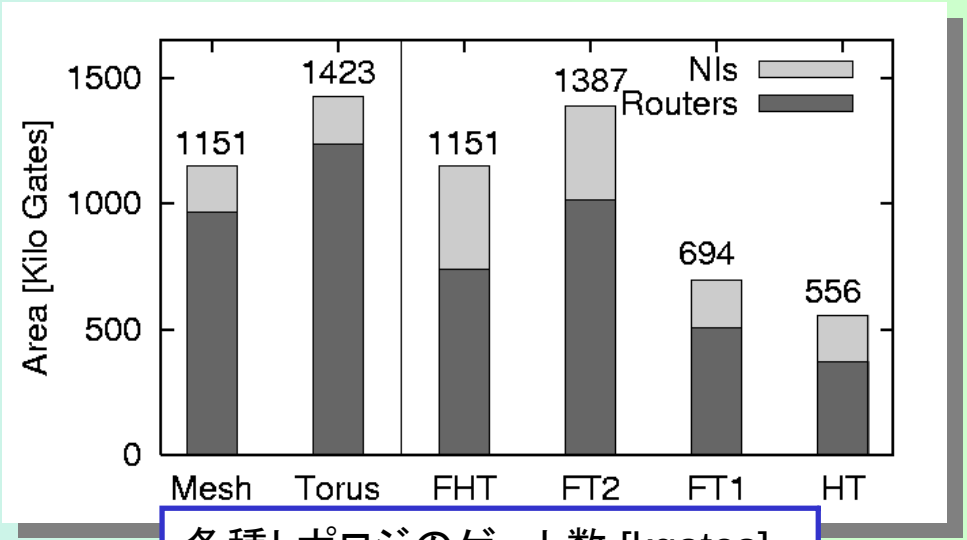
[Wang, DATE'05]

- シミュレーション環境
  - 8mm角チップ (16 / 64コア)
  - 90nm CMOS
- 転送エネルギー E_{sw}
  - 1-bit 転送 @ ルータ(NI)
  - ルータ回路のゲートレベル解析 (Power compiler)
  - 0.257 [pJ / hop] for routers
  - 0.101 [pJ / hop] for NI
- 配線エネルギー E_{link}
  - 1-bit 転送 @ リンク
  - 0.180 [pJ / mm]

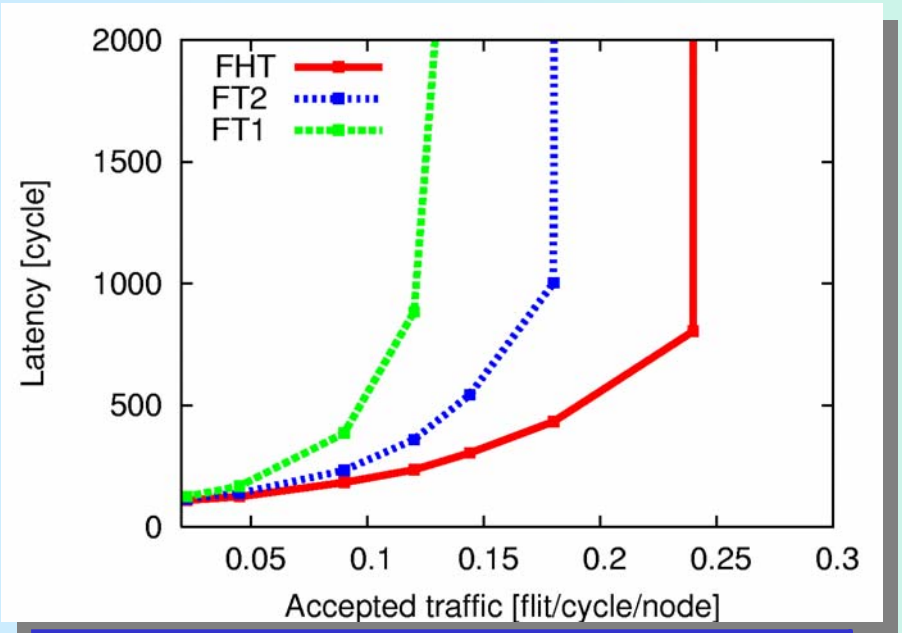
[Ho, IEEE Proc'01]

NoC の評価手法: まとめ

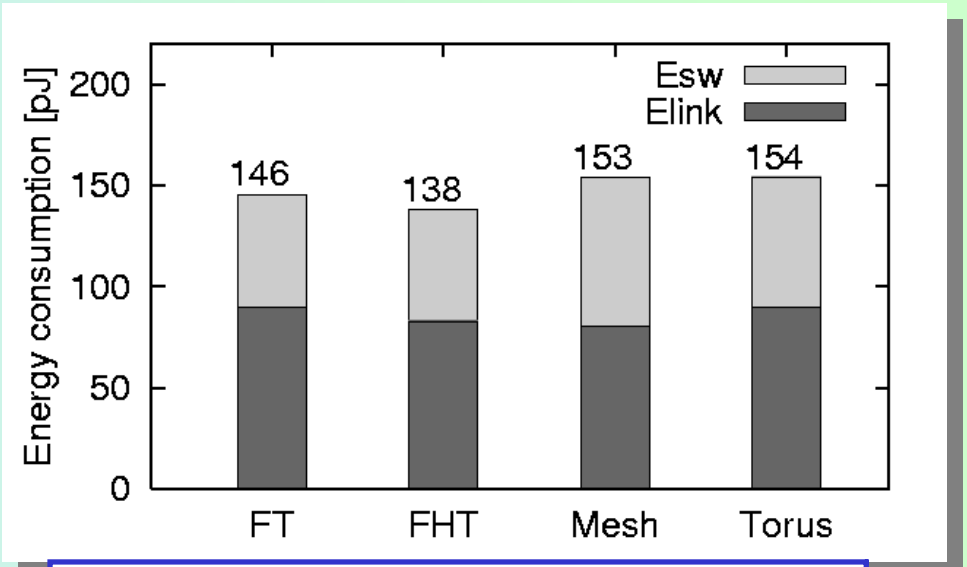
- 必須の評価項目
  - ハードウェア量
  - スループット性能
  - 消費電力 or エネルギー



各種トポロジのゲート数 [kgates]



各種トポロジのスループット [flit/core/cycle]



各種トポロジのフリット転送エネルギー [pJ]

発表の流れ

- Network-on-Chip (NoC) の概要
  - ネットワークトポロジ
  - パケットルーティング
  - ルータアーキテクチャ

- NoC の研究の始め方
  - NoC シミュレータ
  - ルータ回路 (NoC generator)
  - NoC の評価方法

- NoC 研究の動向
  - 最近ホットなトピック
  - 予測機構による低遅延ルータ

[松谷,鯉渕,天野,吉永]

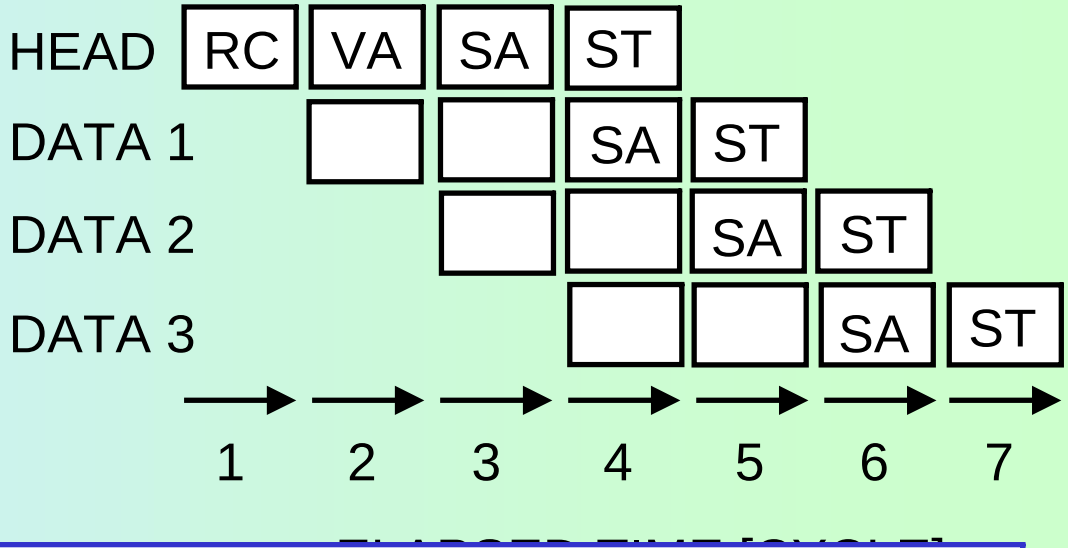
最近のオンチップネットワーク

システム名	トポロジ	ルーティング	スイッチング	フロー制御
MIT RAW	2-D mesh (32bit)	XY DOR	WH, no VC	Credit
UPMC SPIN	Fat Tree (32bit)	Up*/down*	WH, no VC	Credit
QuickSilver ACM	H-Tree (32bit)	Up*/down*	1-flit, no VC	Credit
UMass Amherst aSOC	2-D mesh	Shortest-path	Pipelined CS, no VC	Timeslot
Sun T1	Crossbar (128bit)	-	-	Handshake
Cell BE EIB	Ring (128bit)	Shortest-path	Pipelined CS, no VC	Credit
TRIPS (operand)	2-D mesh (109bit)	YX DOR	1-flit, no VC	On/off
TRIPS (on-chip)	2-D mesh (128bit)	YX DOR	WH, 4 VCs	Credit
Intel SCC	2-D torus (32bt)	XY,YX DOR, odd-even TM	WH, no VC	Stall/go
Intel Teraflops NoC	2-D mesh (32bit)	Source routing	WH, 2 lanes	On/off
TILE64 iMesh	2-D mesh (32bit)	XY DOR	WH, no VC	Credit

最近ホットな研究: 低遅延ルータ

- ルータアーキテクチャ
  - 低遅延ルータ
- ネットワークトポロジ
  - Flattened Butterfly
- 3次元積層
  - 3-D NoC アーキテクチャ
- ソフトエラー耐性
  - エラー検出/再送, 訂正
- ハードエラー耐性
  - ネットワークの冗長化

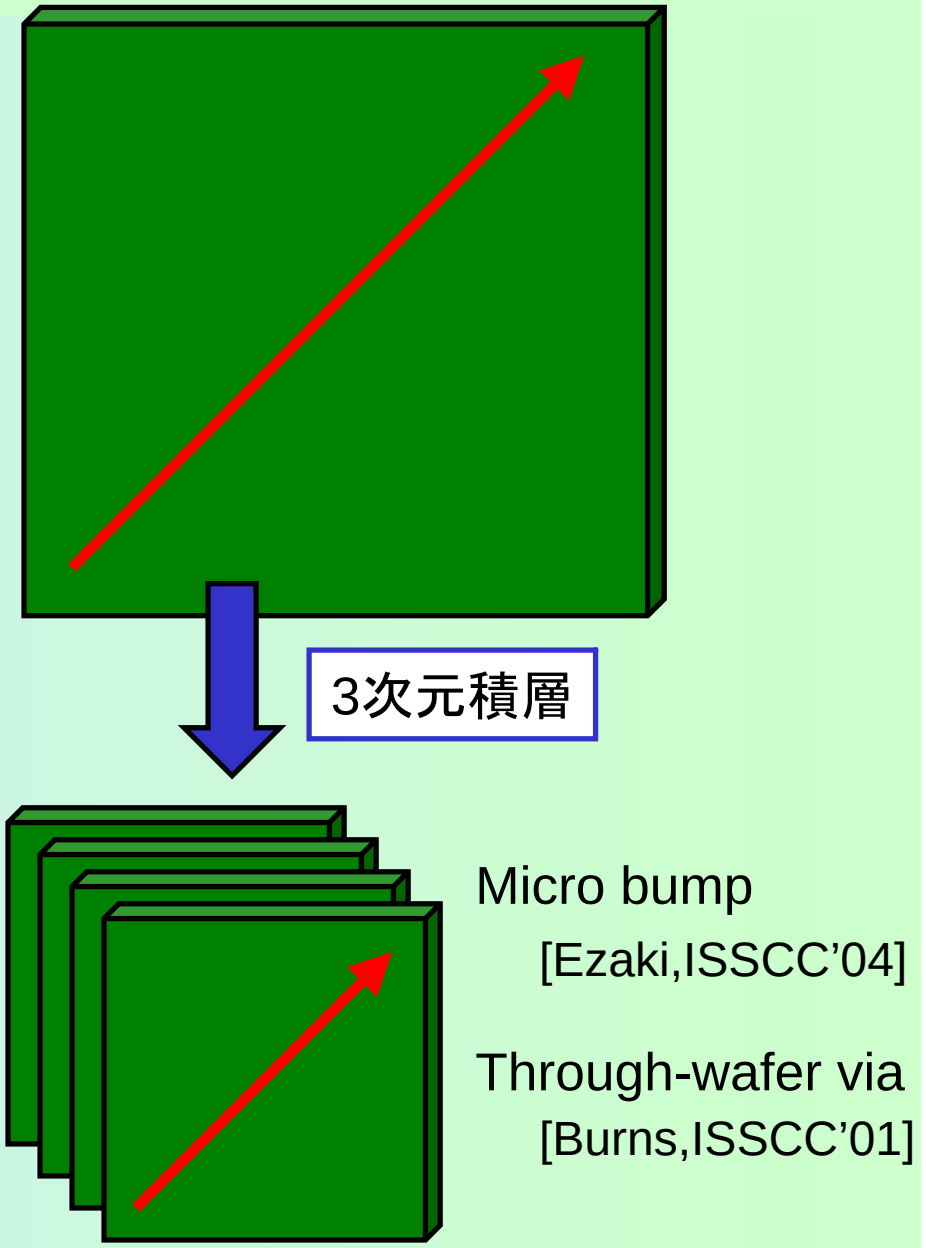
- ルータのパイプライン段数
  - 一般的には 3～4 段
  - 通信遅延に影響を与える
- パイプライン段数を減らす
  - 1段に処理を詰め込む
  - 投機的に実行する



発表の最後で,我々が設計している低遅延オンチップルータを紹介

最近ホットな研究: 3-D NoC アーキテクチャ

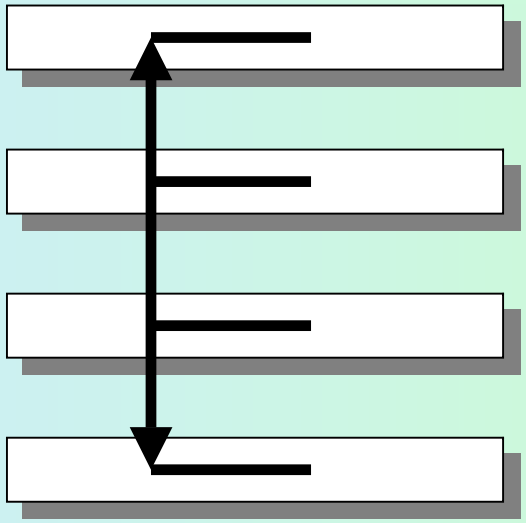
- ルータアーキテクチャ
  - 低遅延ルータ
- ネットワークトポロジ
  - Flattened Butterfly
- 3次元積層
  - 3-D NoC アーキテクチャ
- ソフトエラー耐性
  - エラー検出/再送, 訂正
- ハードエラー耐性
  - ネットワークの冗長化



最近ホットな研究: 3-D NoC アーキテクチャ

- ルータアーキテクチャ
  - 低遅延ルータ
- ネットワークトポロジ
  - Flattened
- 3次元積層
  - 3-D NoC
- ソフトエラー
  - エラー検出
- ハードエラー
  - ネットワー

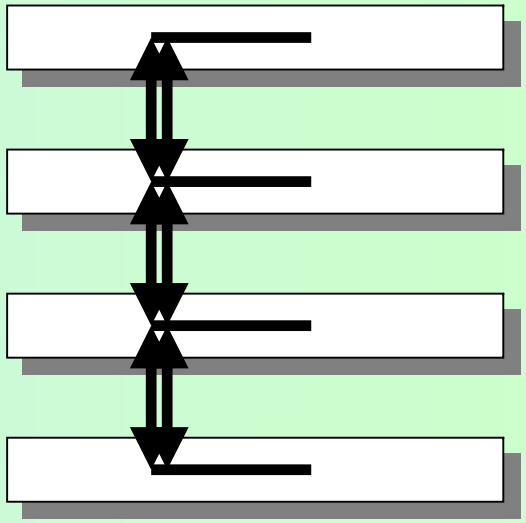
- 3-D Network-on-Chips
  - XY次元: プレーン内通信
  - Z次元:　プレーン間通信



Vertical bus

[Li, ISCA'06]

Single bus (only a single transfer at the same time)



Vertical crossbar

[Kim, ISCA'07]

Segmented buses (multiple transfers at the same time)

最近ホットな研究: ソフトエラー耐性技術

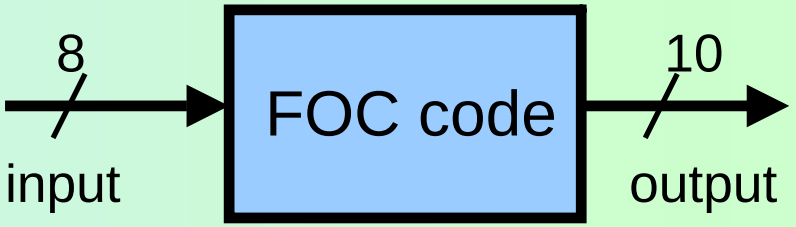
- ルータアーキテクチャ
  - 低遅延ルータ
- ネットワークトポロジ
  - Flattened Butterfly
- 3次元積層
  - 3-D NoC アーキテクチャ

- ソフトエラー耐性
  - エラー検出/再送, 訂正
- ハードエラー耐性
  - ネットワークの冗長化

- クロストーク回避
  - 隣接ワイヤ間距離を広げる
  - シールドを入れる

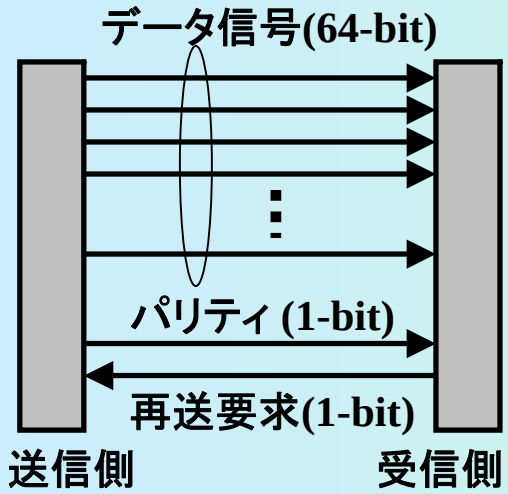
リソース的に非効率

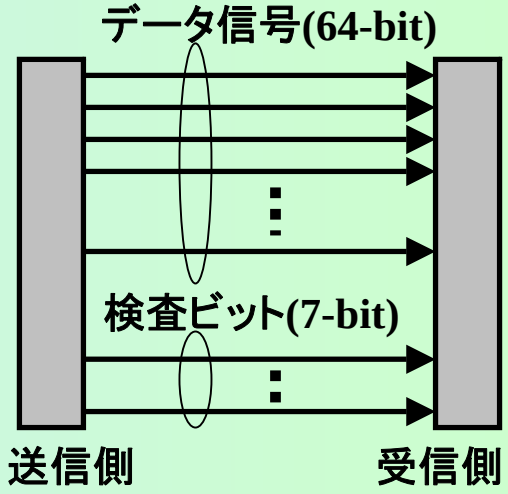
- クロストーク回避コード
  - 101→010 等の遷移を回避
  - Forbidden overlap cond.
  - Forbidden transition cond
  - Forbidden pattern cond.



最近ホットな研究: ソフトエラー耐性技術

- エラー検出,再送
  - ○ 付加ビットが少ない
  - × 誤りが多いと再送が増え,スループット性能が悪化
- エラー訂正
  - × 付加ビットが多い
  - × 符号化処理が必要
  - ○ パケットの再送は不要





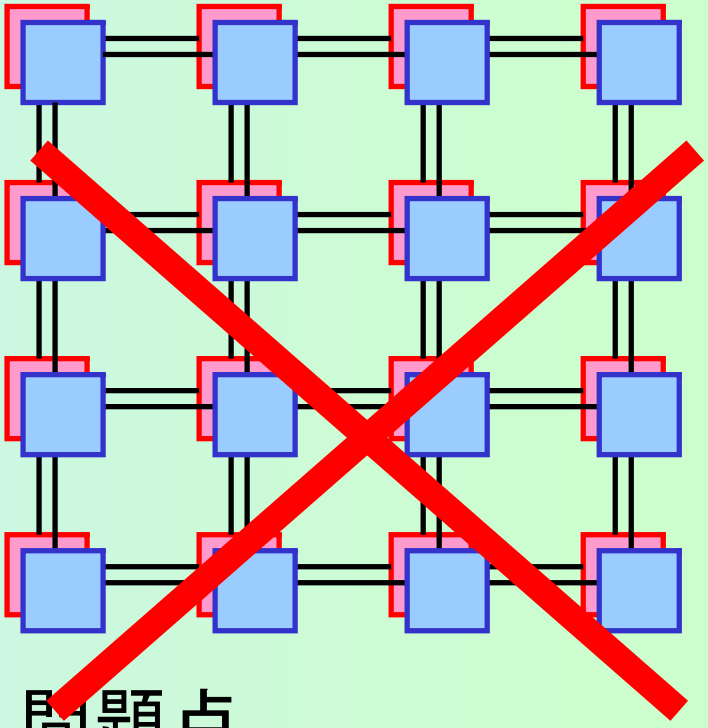
最近は,クロストーク回避コード(CAC) とエラー訂正コード (SEC) を統合した “joint CAC/SEC コード” の研究が盛ん

最近ホットな研究: ハードエラー耐性技術

- ルータアーキテクチャ
  - 低遅延ルータ
- ネットワークトポロジ
  - Flattened Butterfly
- 3次元積層
  - 3-D NoC アーキテクチャ

- ソフトエラー耐性
  - エラー検出/再送, 訂正
- ハードエラー耐性
  - ネットワークの冗長化

- 単純二重化



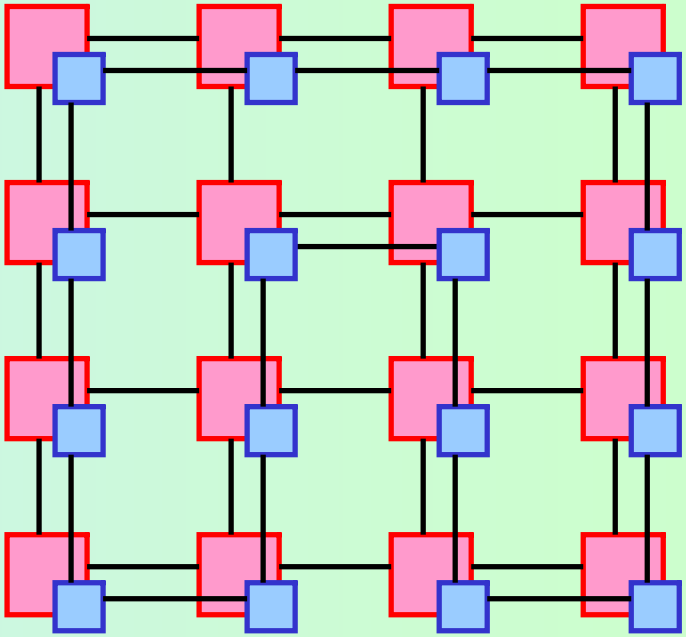
- 問題点
  - ハードウェア量2倍
  - コストが大きすぎる

Original router　Spare router

最近ホットな研究: ハードエラー耐性技術

- ルータアーキテクチャ
  - 低遅延ルータ
- ネットワークトポロジ
  - Flattened Butterfly
- 3次元積層
  - 3-D NoC アーキテクチャ
- ソフトエラー耐性
  - エラー検出/再送, 訂正
- ハードエラー耐性
  - ネットワークの冗長化

- Default Backup Path



- Backup path (ring)
  - 全部のコアを一筆書き
  - 最低限の追加ハードウェア

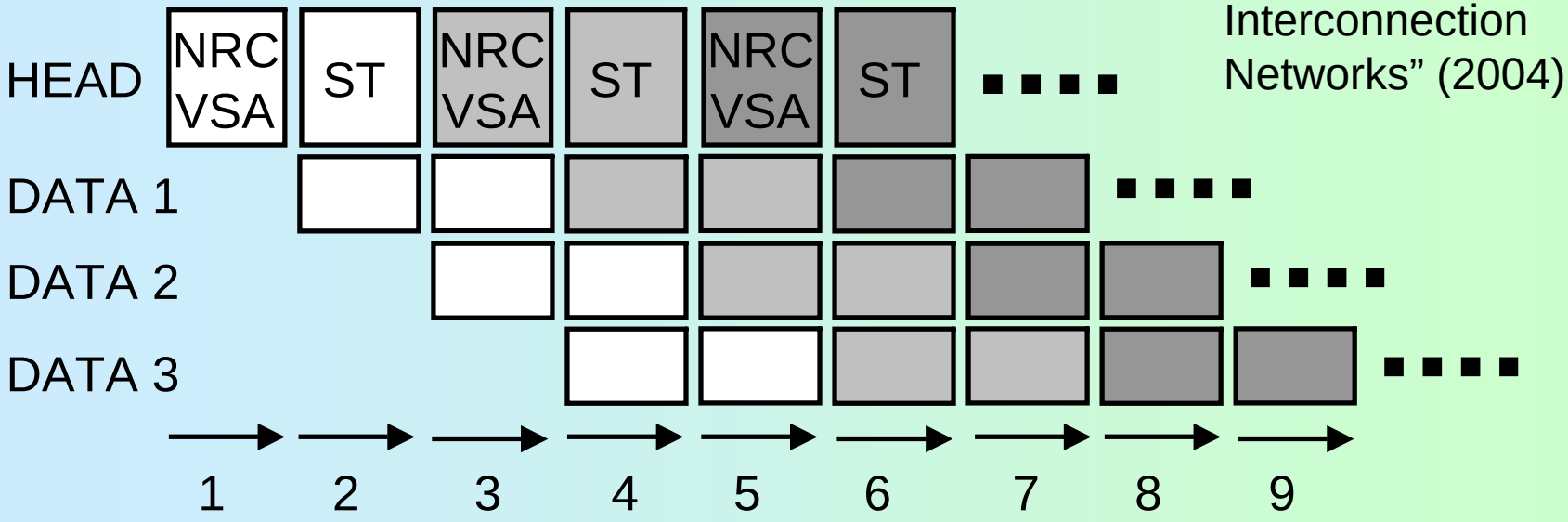
Original router　Spare router

予測機構を持った低遅延オンチップルータアーキテクチャ

松谷 宏紀　(慶大)
鯉渕 道紘　(NII)
天野 英晴　(慶大)
吉永　努 (電通大)

一般的な低遅延ルータ: 2段パイプライン

- 別アプローチ 1 – Express virtual channels
  - 非隣接ルータ間に仮想的なバイパス経路 [Kumar,ISCA'07]
  - 隣接間通信が多いと効果が薄い
- 別アプローチ 2 – Preferred path
  - XYルーティングを想定し, パケットが直進すると予測
  - クロスバを迂回する低遅延なパス [Michelogiannakis,NOCS'07]



Interconnection Networks" (2004)

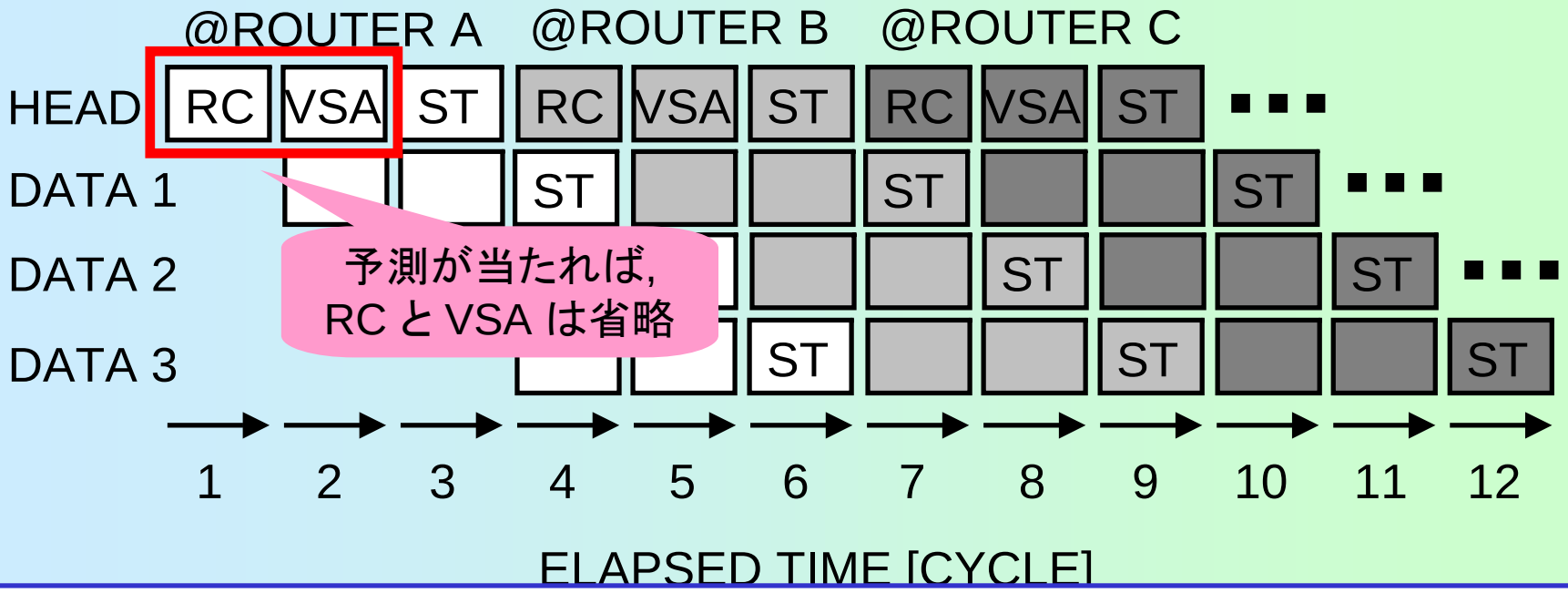
1-cycleルータもあるが,1ステージに詰込み過ぎ → 動作周波数悪化

予測ルータ: Yet another 1-cycle router

[吉永,SACSIS'07]
[鯉渕,SACSIS'08]

- 予測による1サイクル転送
  - どの出力ポートが使われるか予測する (RC をプレ実行)
  - 予測した出力ポートでクロスバ調停を済ます (SA をプレ実行)

予測が正しければ ST だけで転送 (1サイクル転送)



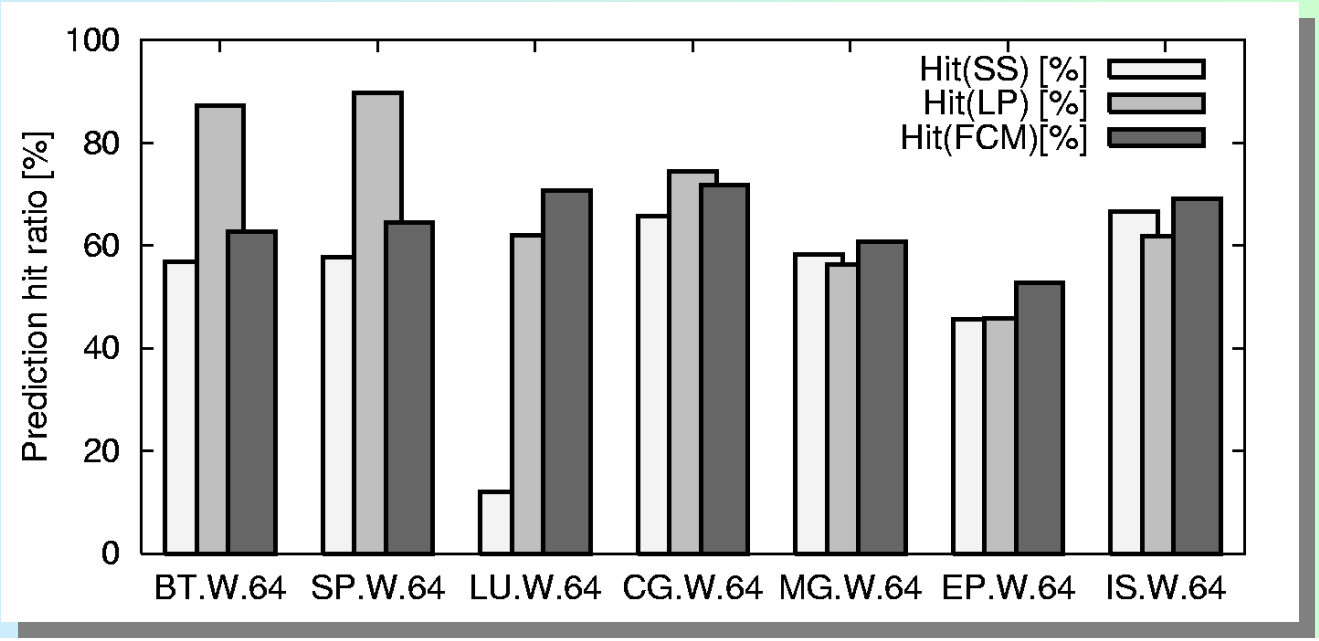
1ステージに処理を詰込まない;予測が70%当たれば1.6サイクル転送

予測ルータ: いろんな予測アルゴリズム

- Random
- Static straight (SS)
  - 直進ポート予測
  - 例) 右から来たら左へ, 上から来たら下へ
- Custom
  - どのポートを予測すべきか 2-bit レジスタに格納
- Least Port (LP)
  - 前回使用したポートをもう一度使うと予測
- Finite context method (FCM)
  - 最も頻繁に出現する n個のコンテキストパターンで予測
  - 0th-order FCM, 1st-order FCM, 2nd-order FCM
- Sampled pattern matching (SPM)
  - 履歴テーブル, パターンマッチング

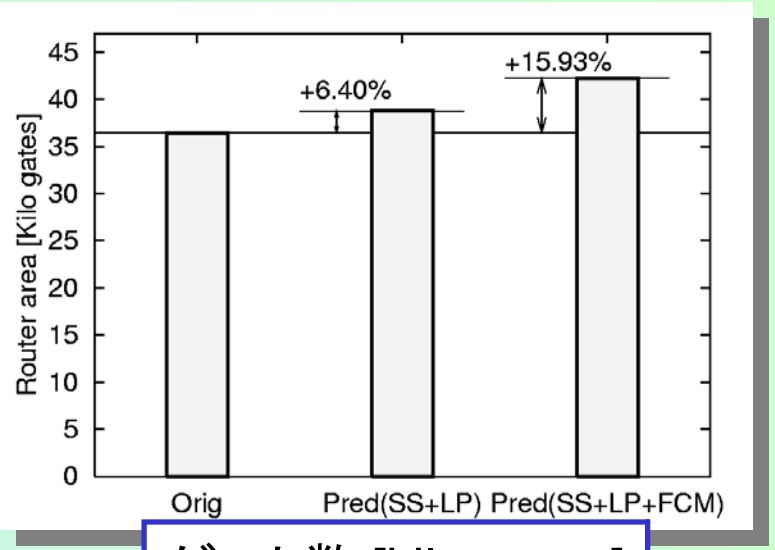
予測ルータ: いろんな予測アルゴリズム

- 既存の投機的な低遅延転送
  - 次元順ルーティングの規則性を活用 [Kumar,ISCA'07]
  - static straight (SS) 相当 [Michelogiannakis,NOCS'07]
  - SS は隣接通信,小規模ネットワークでは全然効かない
- 予測ルータ
  - 複数の予測アルゴリズムを内蔵
  - トポロジ,ルーティング,トラフィックごとに切り替え可能

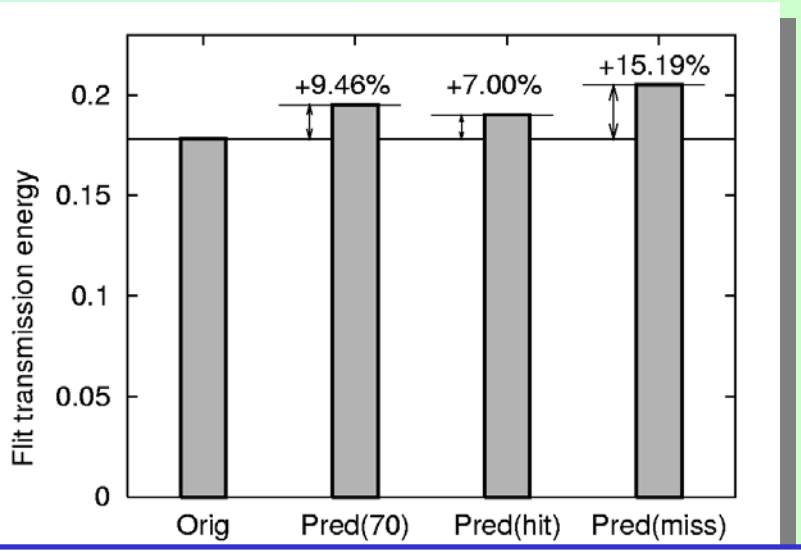


予測ルータ: 予測ルータのハードウェア実装

- 3種類のルータ
  - オリジナルルータ
  - 予測ルータ (SS + LP)
  - 予測ルータ (SS + LP + FCM)
- 予測ルータの実装
  - 65nm standard cell
  - Design compiler で合成
  - Astro で配置配線
- 消費電力の解析
  - レイアウト後シミュレーション
  - Power Compiler



ゲート数 [kilo gates]



面積オーバヘッドは 6.4%～16%, 電力オーバヘッドは 9.7%前後

発表のまとめ

- Network-on-Chip (NoC) の概要
  - ネットワークトポロジ
  - パケットルーティング
  - ルータアーキテクチャ
- NoC の研究の始め方
  - NoC シミュレータ
  - ルータ回路 (NoC generator)
  - NoC の評価方法
- NoC 研究の動向
  - 最近ホットなトピック
  - 予測機構による低遅延ルータ

[松谷,鯉渕,天野,吉永]

Any Questions?