動的再構成可能データベース処理エンジンとクエリコンパイラの検討

三好健文[1)]　寺田裕太[2)]　川島英之[3)]　吉永努[1)]

1)電気通信大学大学院情報ネットワークシステム学研究科

2)電気通信大学電気通信学部

3)筑波大学大学院システム情報工学研究科

動的再構成可能~~データベース~~ストリーム処理エンジンとクエリコンパイラの検討

三好健文[1)] 寺田裕太[2)] 川島英之[3)] 吉永努[1)]

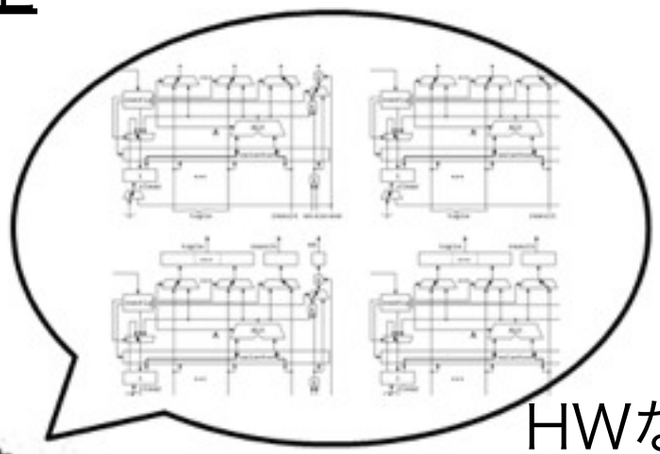
1)電気通信大学大学院情報ネットワークシステム学研究科

2)電気通信大学電気通信学部

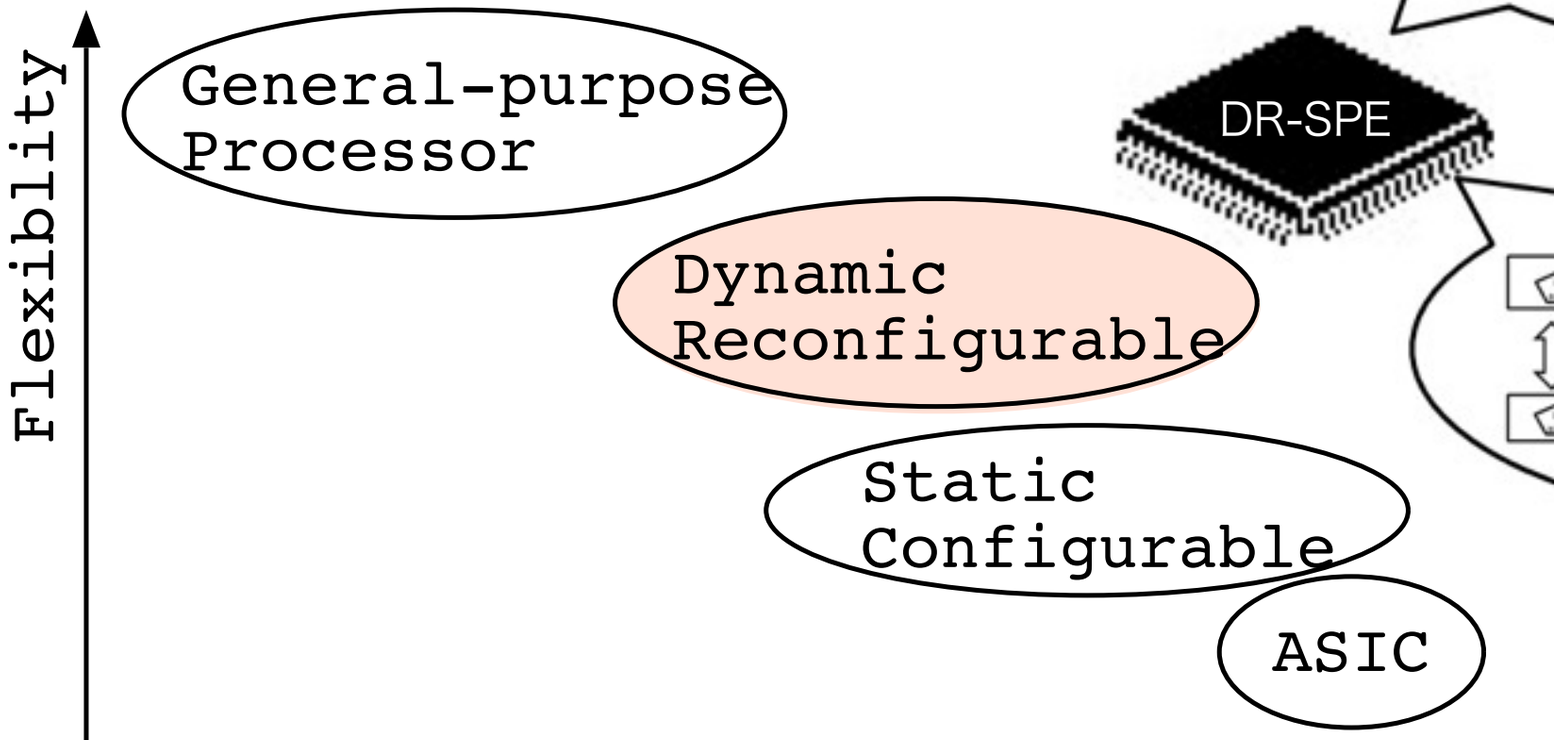
3)筑波大学大学院システム情報工学研究科

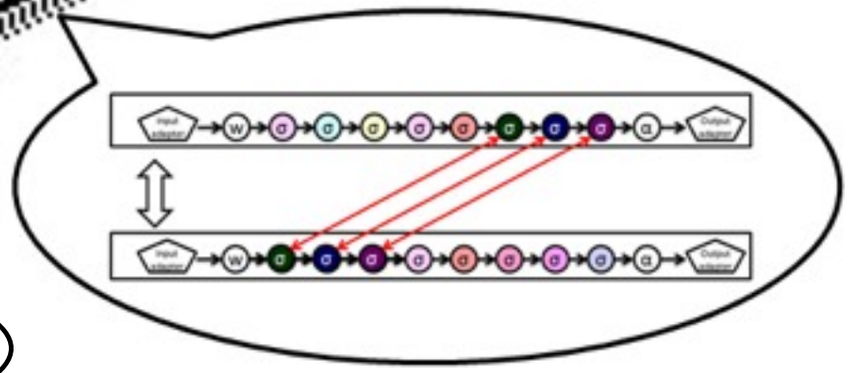
動的再構成可能ストリーム処理エンジン (DR-SPE)

専用ハードウェアとしての高い処理能力と柔軟性を併せ持つ動的再構成可能プロセッサの一種



HWならではの並列処理による高い処理性能





瞬時にクエリプランを切り替え可能

概要

- 専用ハードウェアによる高速なストリーム処理と動的クエリ最適化可能な柔軟性を有する動的再構成可能ストリーム処理エンジンのプロセッサアーキテクチャを設計
- HWリソース量/動作周波数および再構成に必要な時間を評価

目次

- 背景 - ストリームとストリーム処理エンジン
- 既存手法と課題 - Streams on Wires -
- 動的再構成可能ストリーム処理エンジン
- プロセッサアーキテクチャの設計と評価
- まとめと今後の課題

目次

- 背景 - ストリームとストリーム処理エンジン
- 既存手法と課題 - Streams on Wires -
- 動的再構成可能ストリーム処理エンジン
- プロセッサアーキテクチャの設計と評価
- まとめと今後の課題

(データ)ストリーム

- 時事刻々と変化するデータ
  - 東京証券取引所　2ms
  - ルータ 300Tbps
  - 産業用ロボット(モータ制御) 1ms
  - 監視カメラ 30fps * 台数

ストリーム処理

データストリーム： 進展する高速化（322 Tbps, Ring of Steel）
アプリケーション： 高度な解析*を要求（異常検知, 行動解析）

↓

解析には **Programming** が必要

汎用言語(C++/Java)

利点：高い記述能力
欠点：最適化が困難

⟺

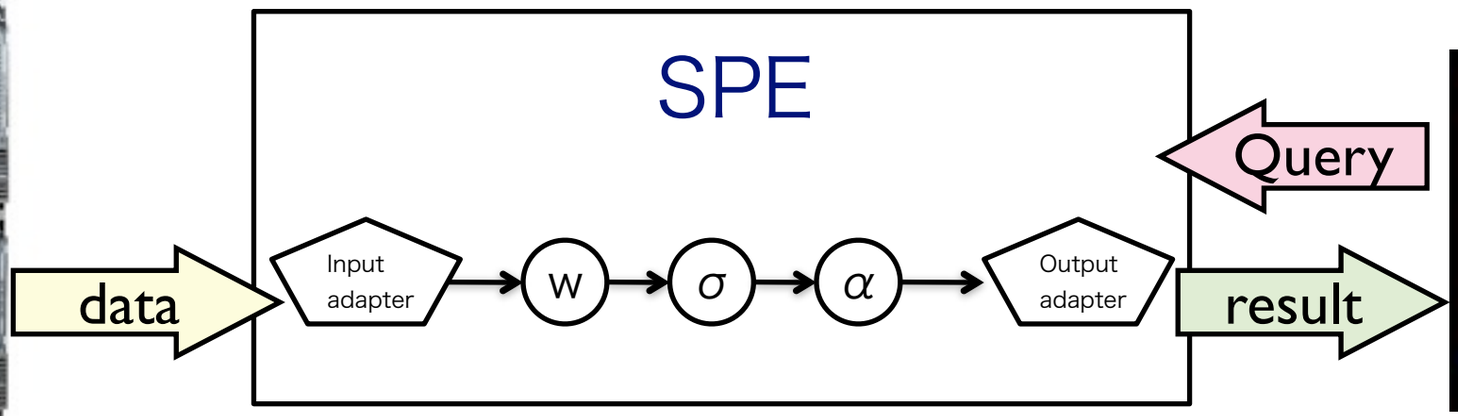
専用言語(SQL (+UDF))

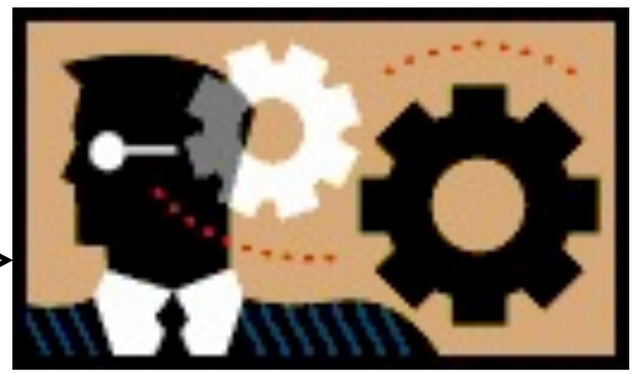
利点：最適化が容易
欠点：低い記述能力

▶ Stream Processing Engine

*解析のみならず, 信憑性なども要求される可能性有

ストリーム処理エンジン





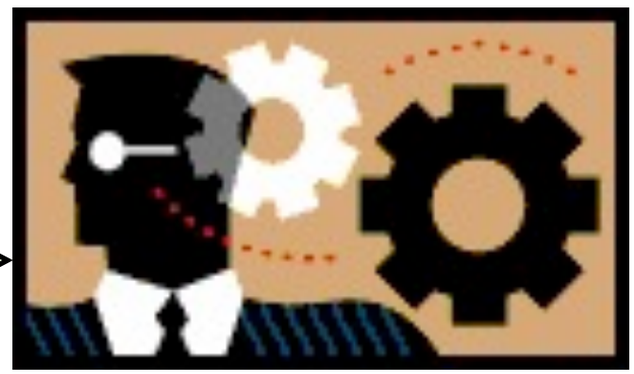
Users/Apps.

- 連続的問合せは演算木に変換
- データ到着毎に問合せ結果が出力
- RDBと類似した演算子
  - w(Window)： 窓演算（1分間だけ）
  - σ (Selection)： 選択演算（一部タプルを抜粋）
  - α (Aggregation)： 集約演算（数え上げ）

取引No.	取引者名	株式名	取引株数	取引金額
1	A	XXX	20	2000
2	B	YYY	39	1498
3	A	XXX	54	2030
4	C	XXX	23	2020
・	・	・	・	・
・	・	・	・	・
・	・	・	・	・
・	・	・	・	・
・	・	・	・	・

ストリーム処理エンジン

専用ハードウェア化による
高速化の実現



Users/Apps.

- 連続的問合せは演算木に変換
- データ到着毎に問合せ結果が出力
- RDBと類似した演算子
  - w(Window)：　　窓演算（1分間だけ）
  - σ (Selection)：　選択演算（一部タプルを抜粋）
  - α (Aggregation)：集約演算（数え上げ）

取引No.	取引者名	株式名	取引株数	取引金額
1	A	XXX	20	2000
2	B	YYY	39	1498
3	A	XXX	54	2030
4	C	XXX	23	2020
・	・	・	・	・
・	・	・	・	・
・	・	・	・	・
・	・	・	・	・
・	・	・	・	・

目次

- 背景 - ストリームとストリーム処理エンジン
- 既存手法と課題 - Streams on Wires -
- 動的再構成可能ストリーム処理エンジン
- プロセッサアーキテクチャの設計と評価
- まとめと今後の課題

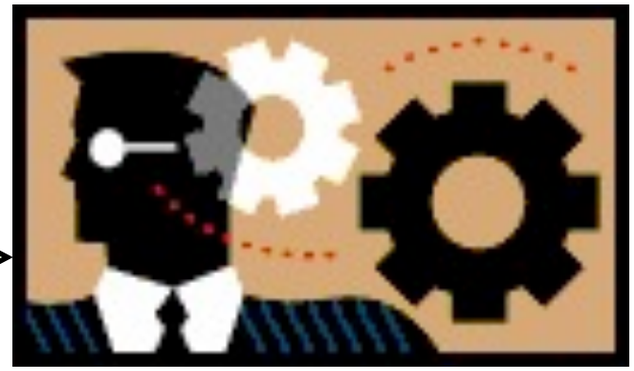
Streams on Wires[1][2]

FPGAを用いたストリーム処理エンジンの実現

Data

Query

Result



Users/Apps.

- 連続的問合せは演算木に変換
- データ到着毎に問合せ結果が出力
- RDBと類似した演算子
  - w(Window)：　　窓演算（1分間だけ）
  - σ (Selection)：　選択演算（一部タプルを抜粋）
  - α (Aggregation)：集約演算

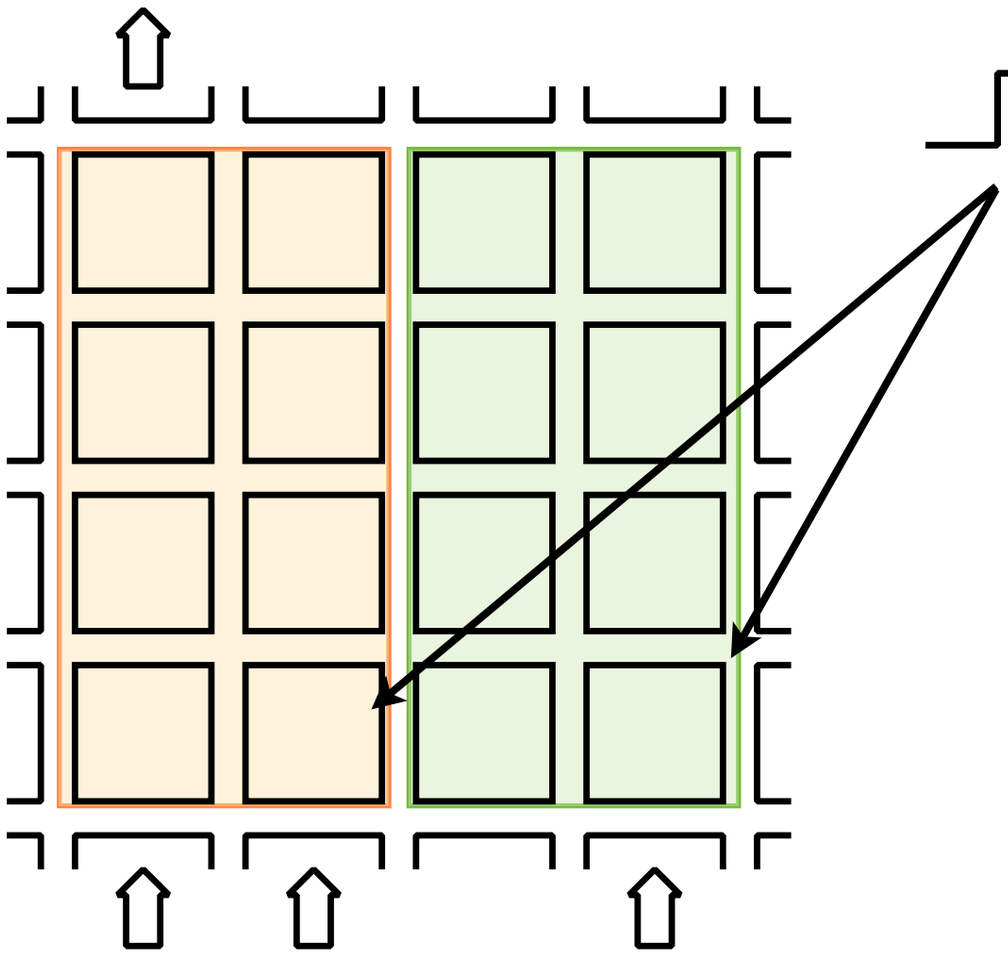
FPGA:
自由に処理内容を設計可能なデバイス

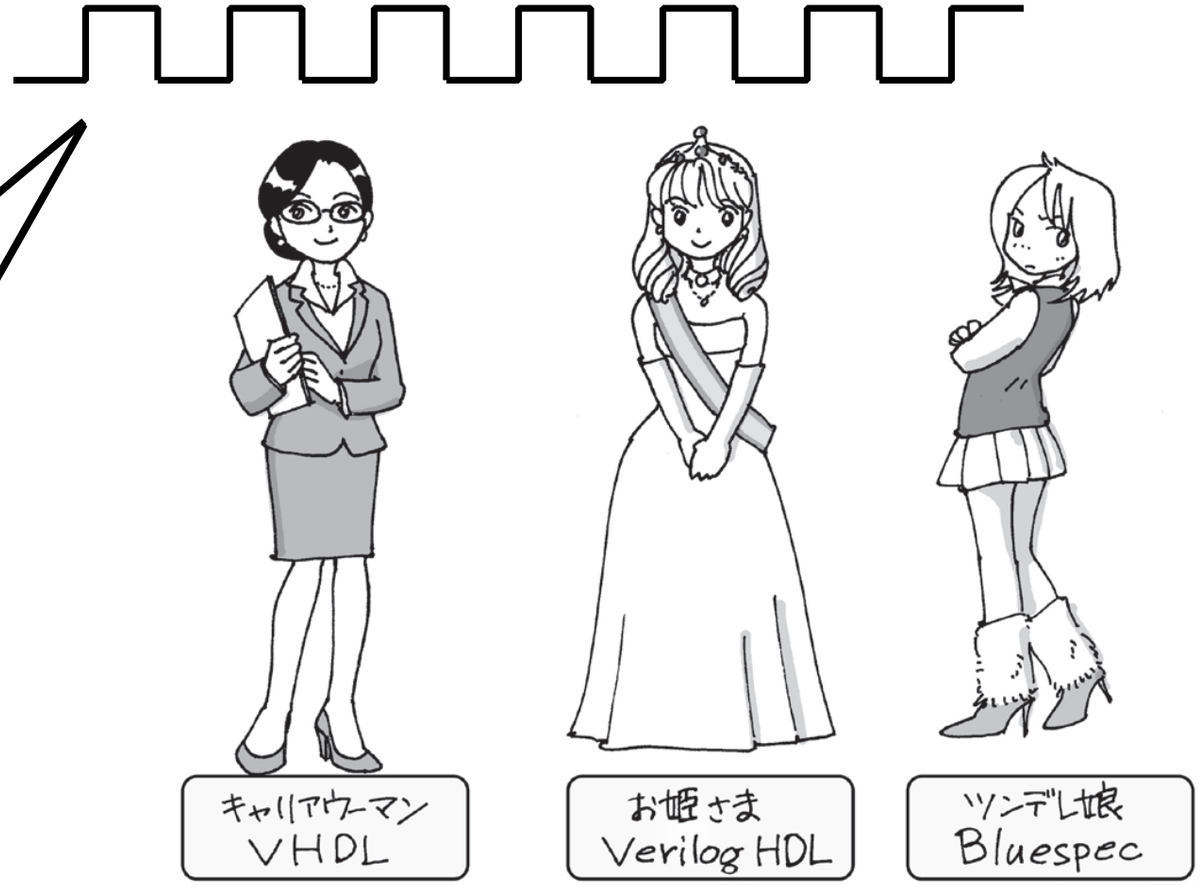
1) Rene Mueller, Jens Teubner, and Gustavo Alonso. Streams on wires: a query compiler for fpgas. *Proc. VLDB Endow.*, Vol.2, No.1, pp. 229–240, 2009.
2) Rene Mueller, Jens Teubner, and Gustavo Alonso. Glacier: a query-to-hardware compiler. In *SIGMOD '10: Proceedings of the 2010 international conference on Management of data*, pp. 1159–1162, New York, NY, USA, 2010. ACM.

FPGAとは？

Field Programmable Gate Array

- 論理回路・データパスを自由に作り込める
- クロックレベルの同期と細粒度並列性の活用





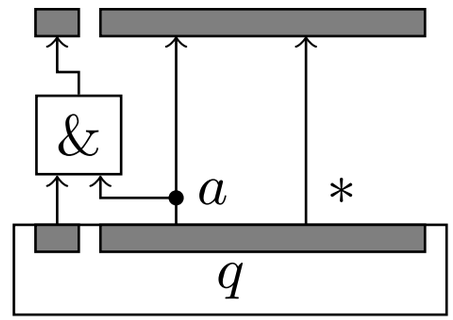
出典: CQ出版　Interface 2011年2月号より

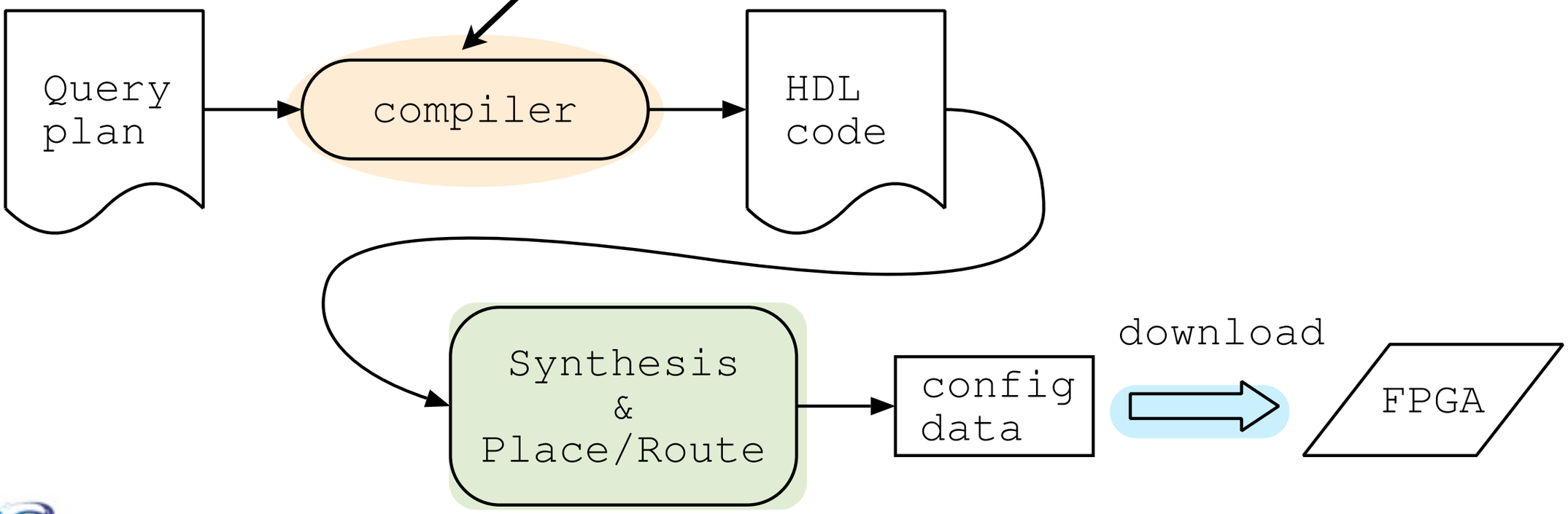
Streams on Wires

FPGA開発環境を利用したクエリの構成

Algebra = ビルディングブロック

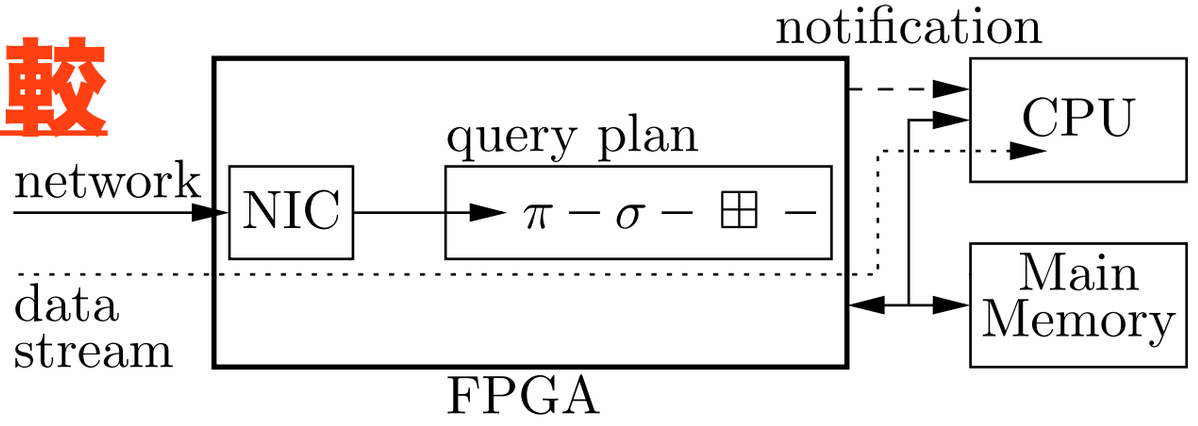
$\pi_{a_1,\ldots,a_n}(q)$	projection
$\sigma_a(q)$	select tuples where field a contains `true`
$\circledast_{a:(b_1,b_2)}(q)$	arithmetic/Boolean operation $a = b_1 \star b_2$
$q_1 \cup q_2$	union
$agg_{b:a}(q)$	aggregate agg using input field a, $agg \in \{$`avg`, `count`, `max`, `min`, `sum`$\}$
q_1 `grp`$_{x\|c}$ $q_2(x)$	group output of q_1 by field c, then invoke q_2 with x substituted by the group
$q_1 \boxplus^t_{x\|k,l}$ $q_2(x)$	sliding window with size k, advance by l; apply q_2 with x substituted on each wind.; $t \in \{\text{time}, \text{tuple}\}$: time-, or tuple-based
$q_1 \circledcirc q_2$	concatenation; position-based field join

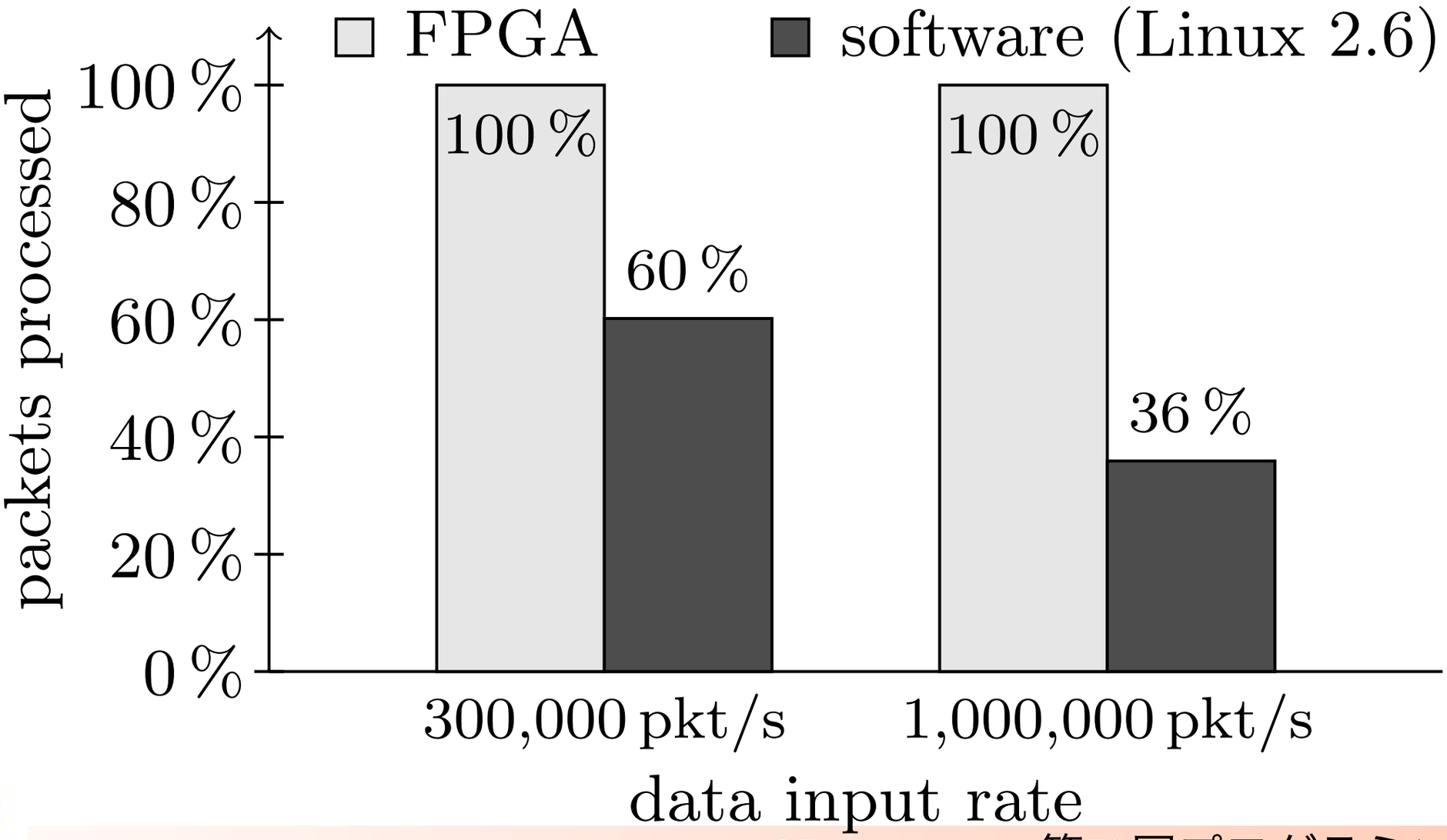




Streams on Wires

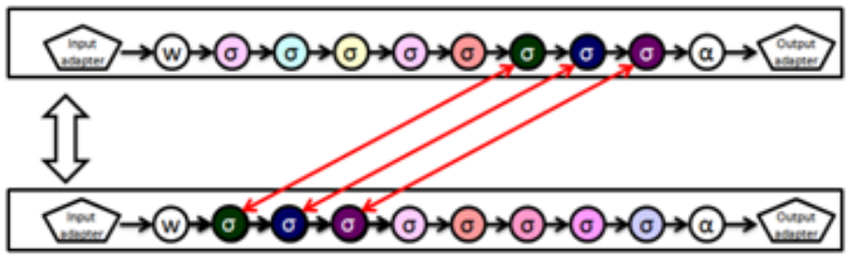
プロセッサとの処理性能比較



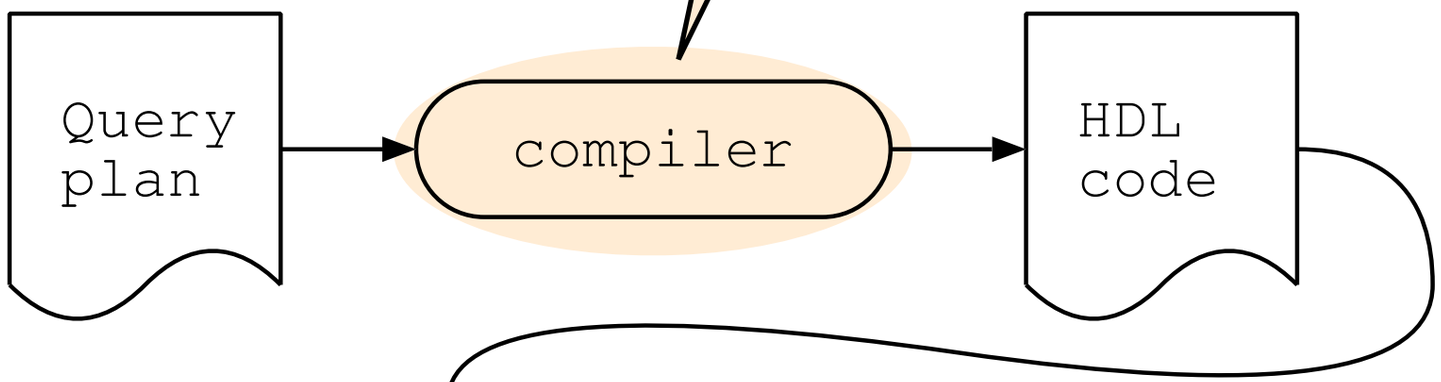


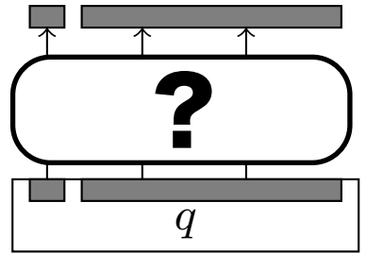
Streams on Wiresの問題点

動的クエリ最適化の適用が困難

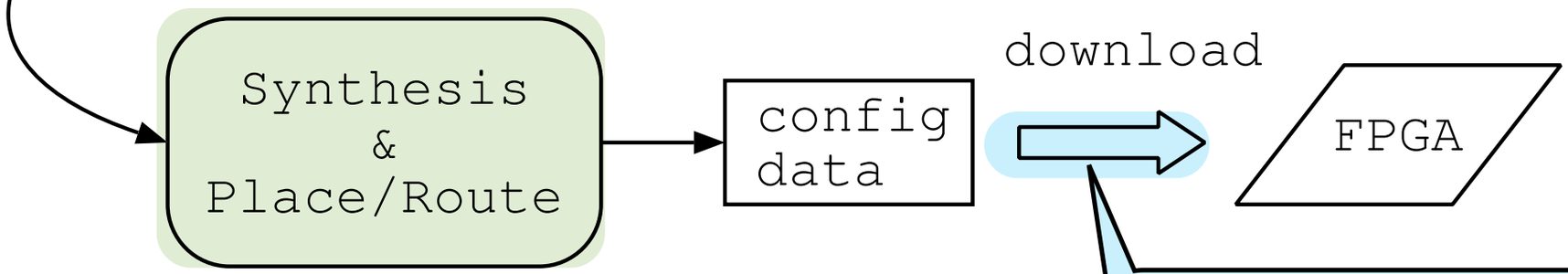


(1) クエリに対する回路の意識が必要





- 演算規模
- 配線の複雑



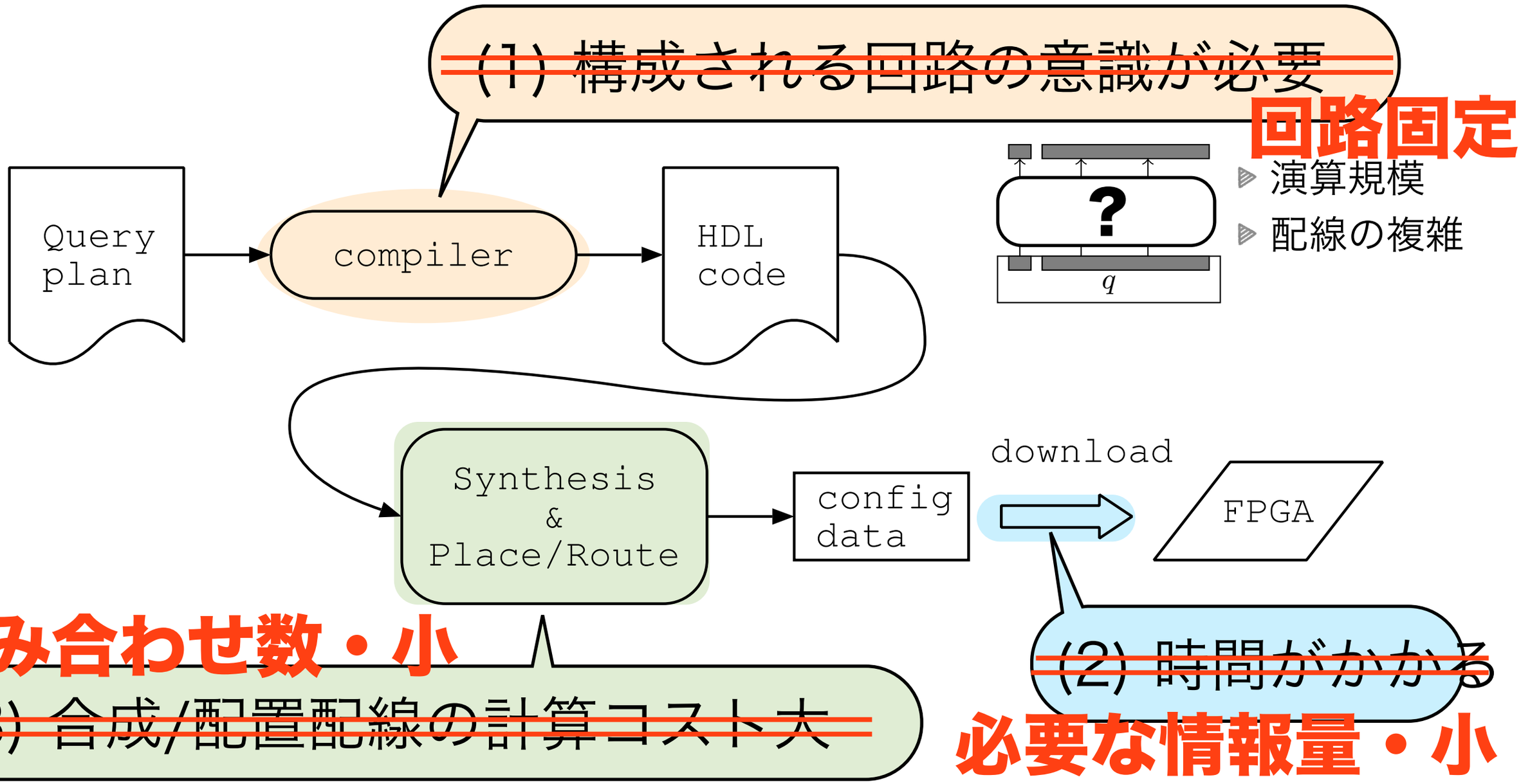
(2) 時間がかかる

(3) 合成/配置配線の計算コスト大

手軽にクエリを構成しなおすことが困難

Streams on Wiresの問題点の解決

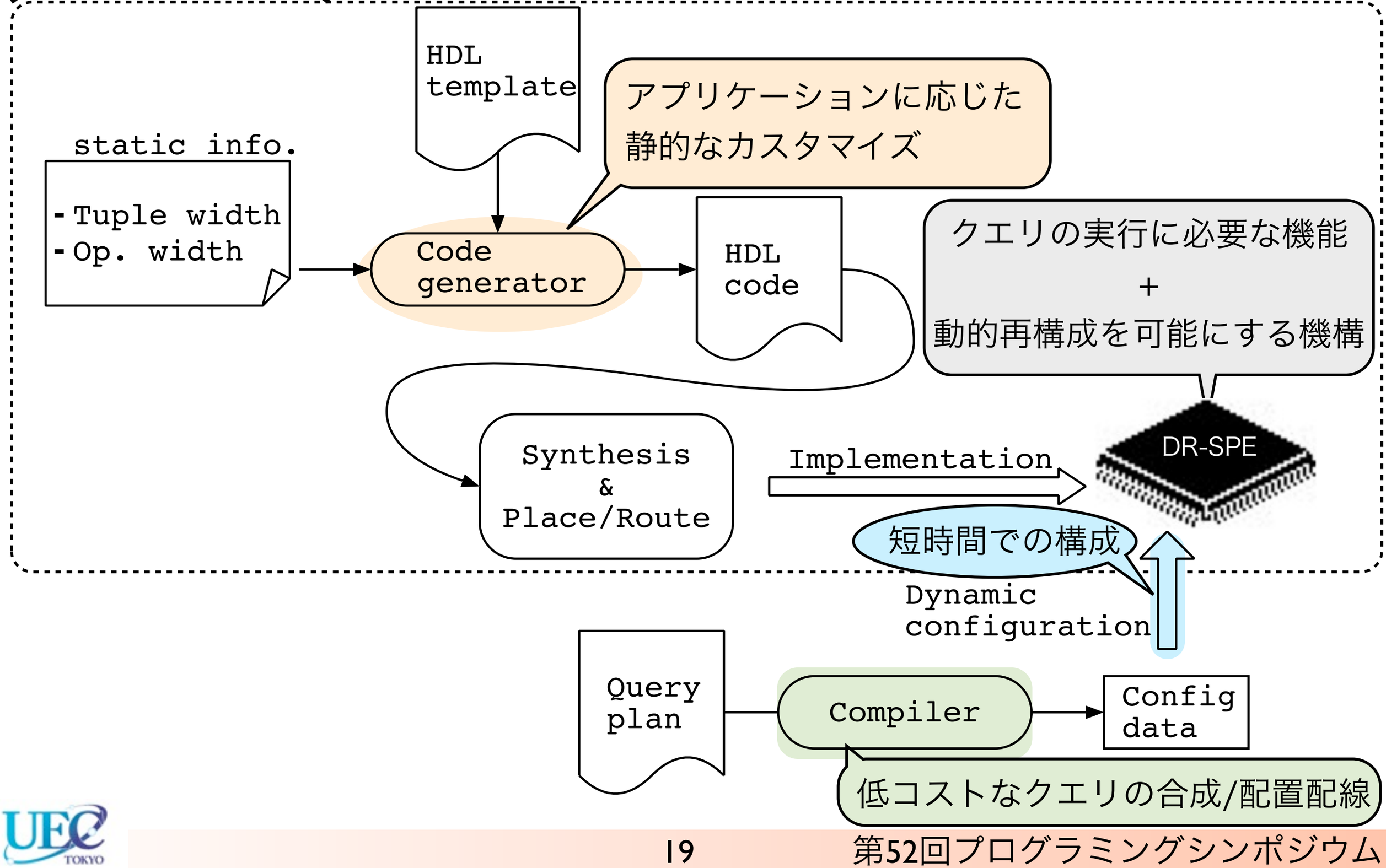
粗粒度の“動的再構成”で解決!!



目次

- 背景 - ストリームとストリーム処理エンジン
- 既存手法と課題 - Streams on Wires -
- 動的再構成可能ストリーム処理エンジン
- プロセッサアーキテクチャの設計と評価
- まとめと今後の課題

動的再構成可能ストリーム処理エンジン (DR-SPE)



DR-SPEの要求仕様

- ストリームデータ処理を実行可能
  ＝ Streams on Wires同等の機能を実現可能
- 高い演算性能の実現
  - サイクルレベルでの処理
  - (パイプライン)並列性の活用
- 動作を実行時に追加/変更できること

DR-SPEの設計課題

- ハードウェアリソース量増加の抑制
  - リソース使用率の向上
- 最大動作周波数低下の抑制
  - 接続関係の柔軟性とスイッチによる信号遅延増加のトレードオフ
- 再構成時間の最小化
  - 柔軟性と構成情報量のトレードオフ

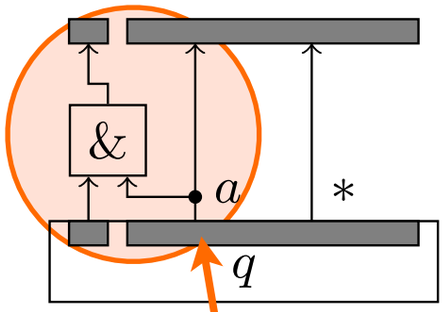
目次

- 背景 - ストリームとストリーム処理エンジン
- 既存手法と課題 - Streams on Wires -
- 動的再構成可能ストリーム処理エンジン
- プロセッサアーキテクチャの設計と評価
- まとめと今後の課題

DR-SPEで実現すべき演算

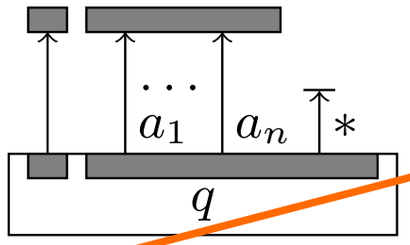
= Streams on Wires同等の演算機能

$\sigma_a(q)$



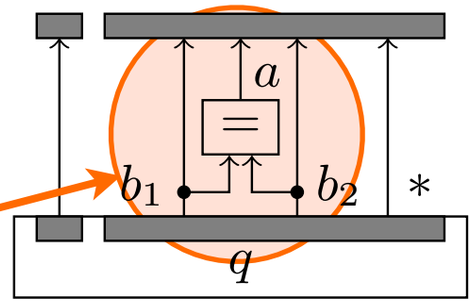
.

$\pi_{a_1,\ldots,a_n}(q)$

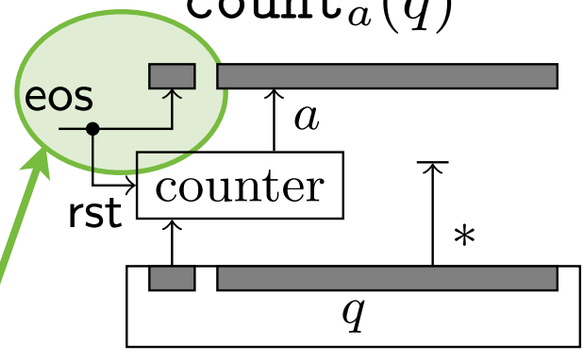


.

$\ominus_{a:(b_1,b_2)}(q)$



$\mathtt{count}_a(q)$

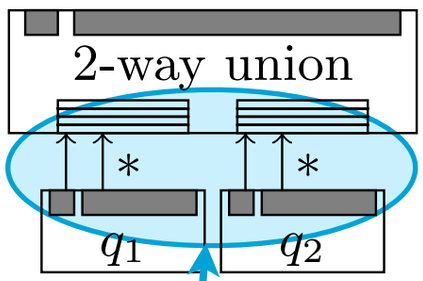


,

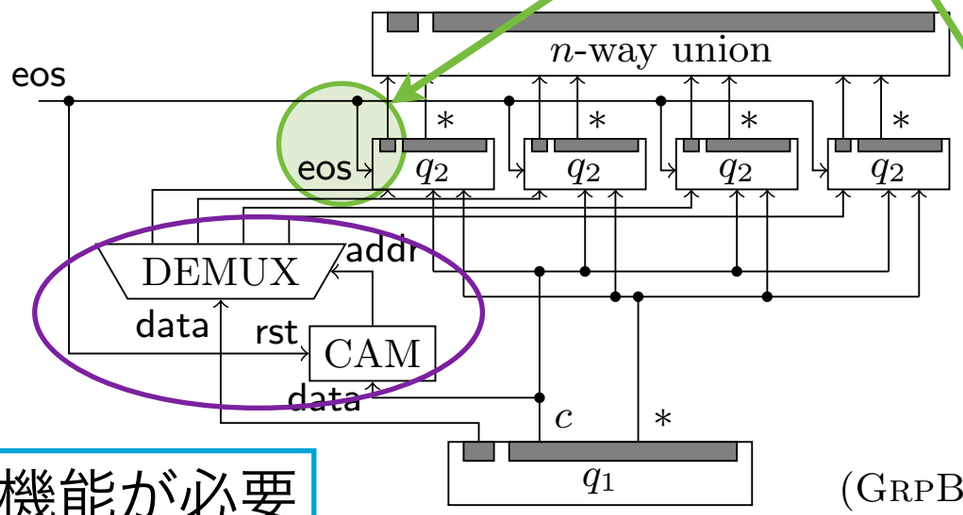
算術/論理演算機能が必要

入出力制御機能が必要

$q_1 \cup q_2$

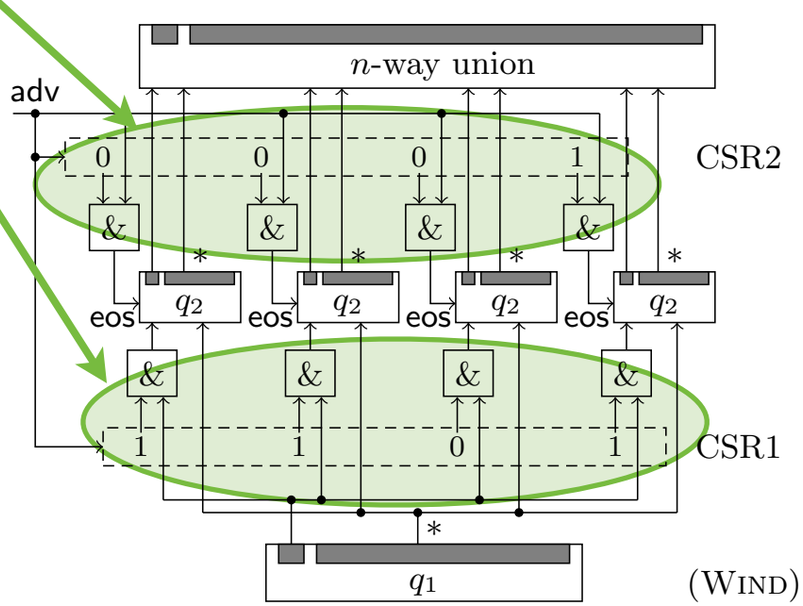


$q_1 \boxplus_{x|k,l} q_2(x)$



(GRPBY)

q_1 grp$_{x|c}$ $q_2(x)$



(WIND)

パスの切り替え機能が必要

共通項をくくりだしてリソース使用率を向上させる

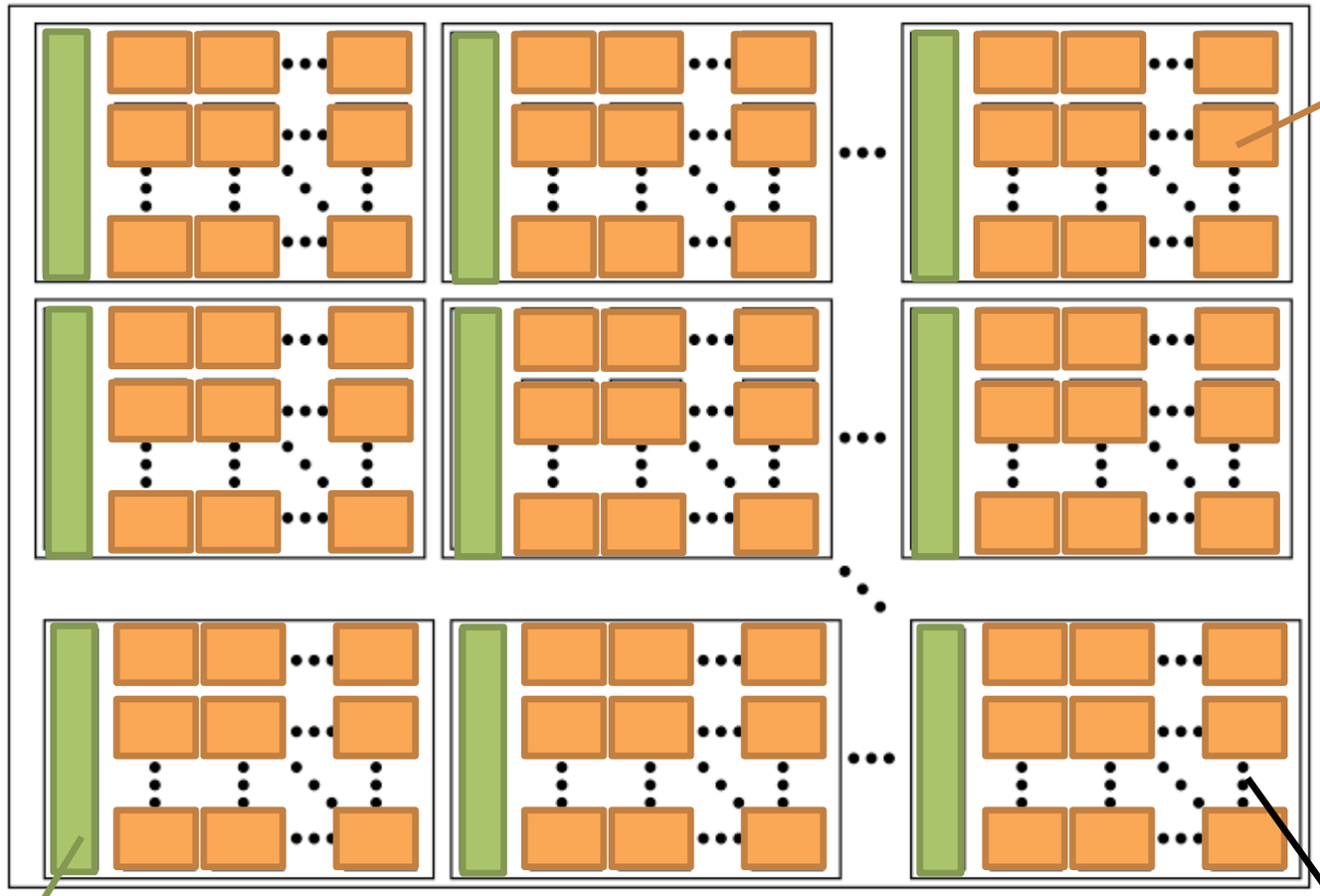
他の動的再構成可能プロセッサとの違い

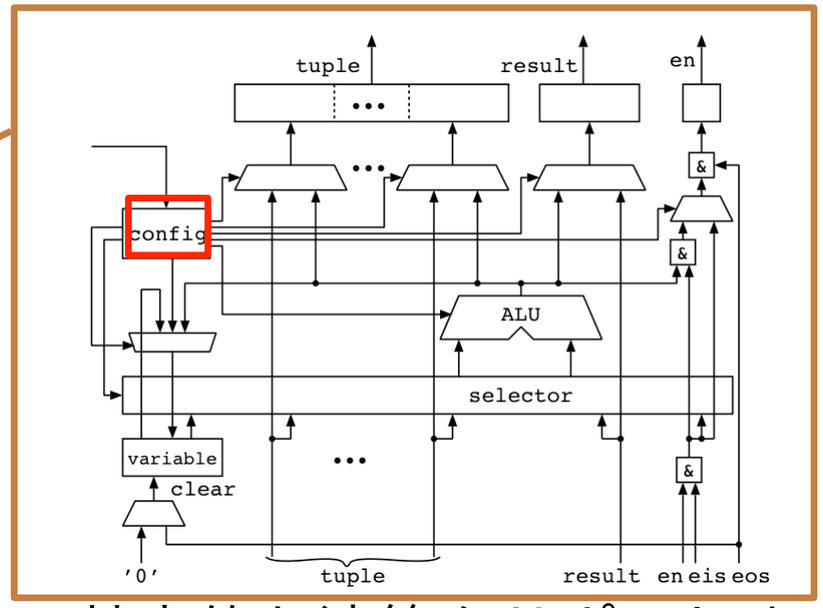
ADRES, FE-GA, DRP, DAP-DNAなどとの違い

- 窓演算に対応する入出力制御
- データの集合に対する操作が必要
- 複数データの流れの切り替え操作
- データの演算だけではない(v.s. フィルタ)

スイッチボックスおよび
ストリーム入出力制御器でサポート

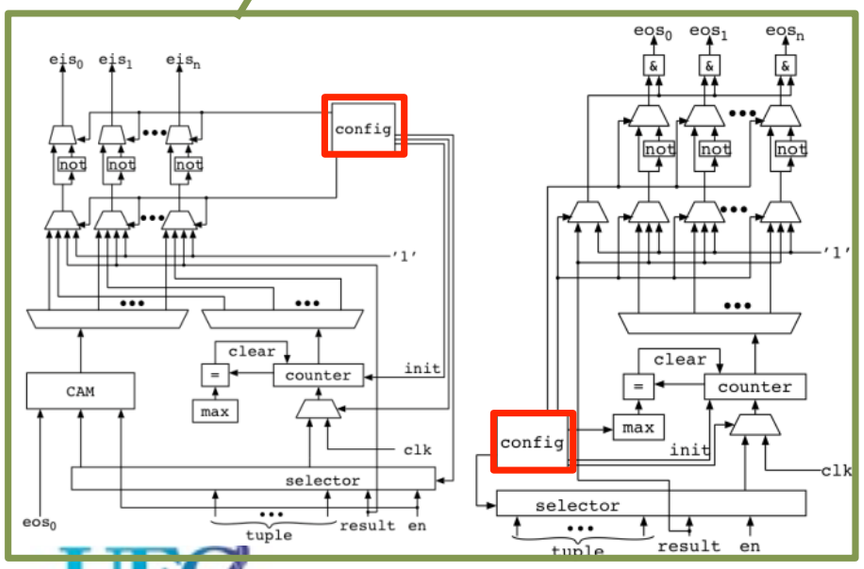
DR-SPE プロセッサアーキテクチャ



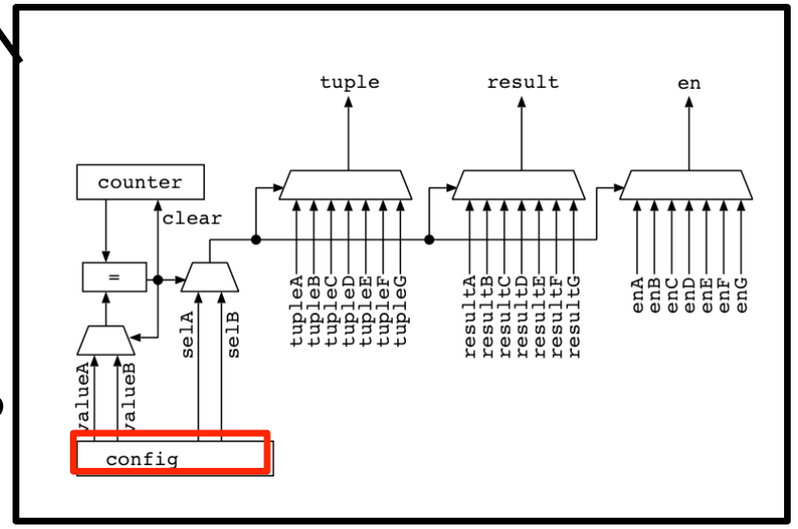


基本的な演算をサポートする
単位演算ユニット

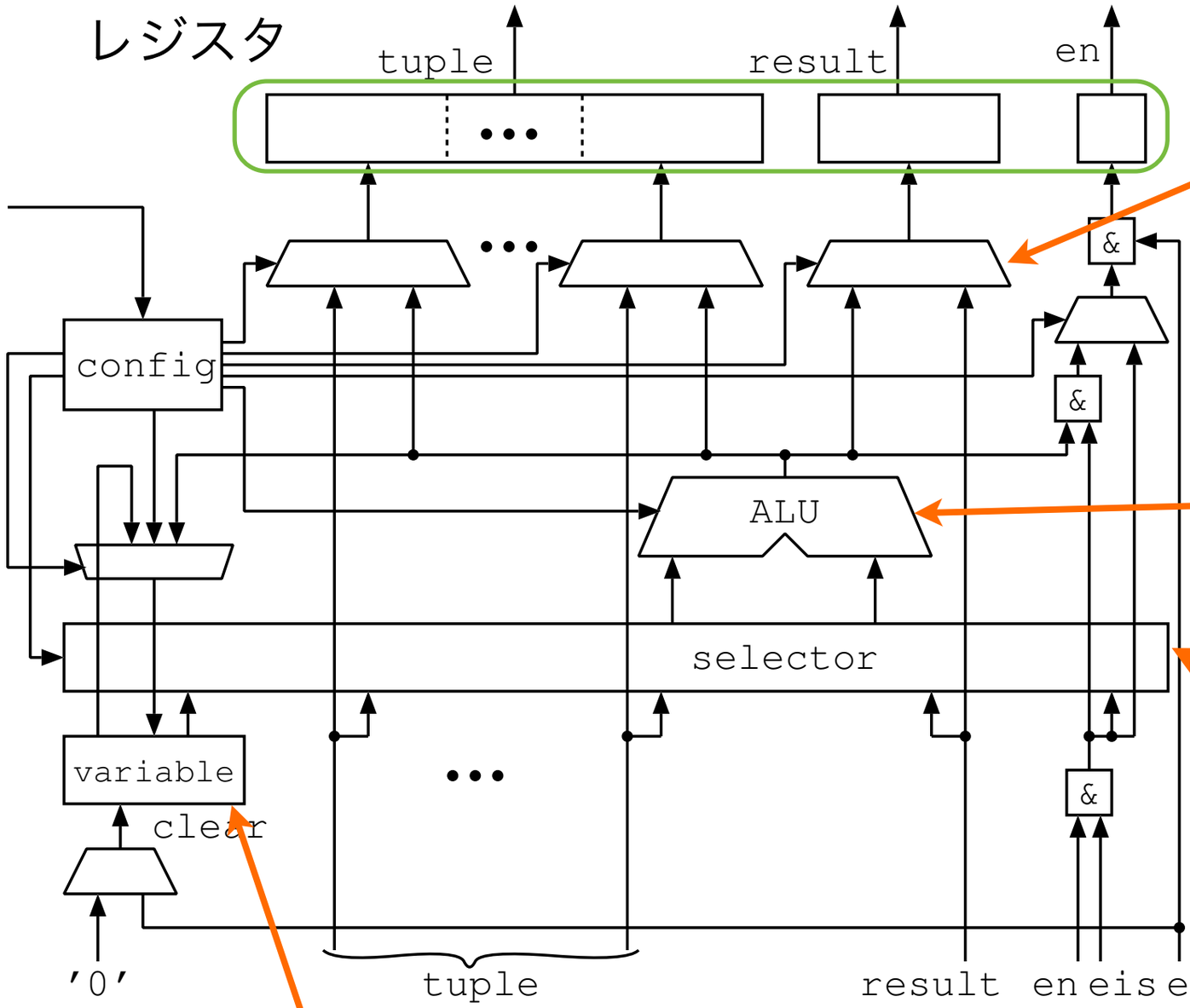
所望のユニット接続を実現する
スイッチボックス



ストリーム入出力を制御する
ストリーム入出力制御器



単位演算ユニット



出力データとする値の選択
スルー or 演算結果

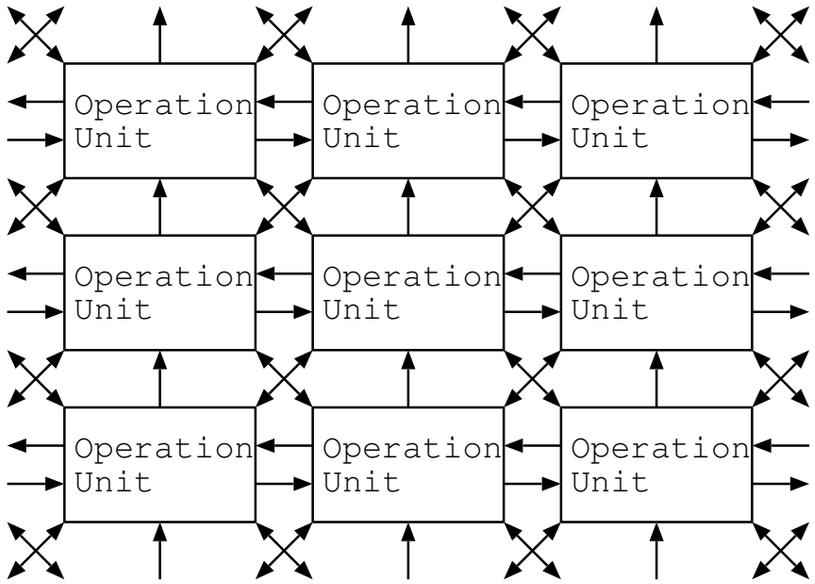
算術/論理演算
加減算，比較など15種

入力データから演算対象
ワードを選択可能

aggregation/対定数演算用の
内部変数レジスタ

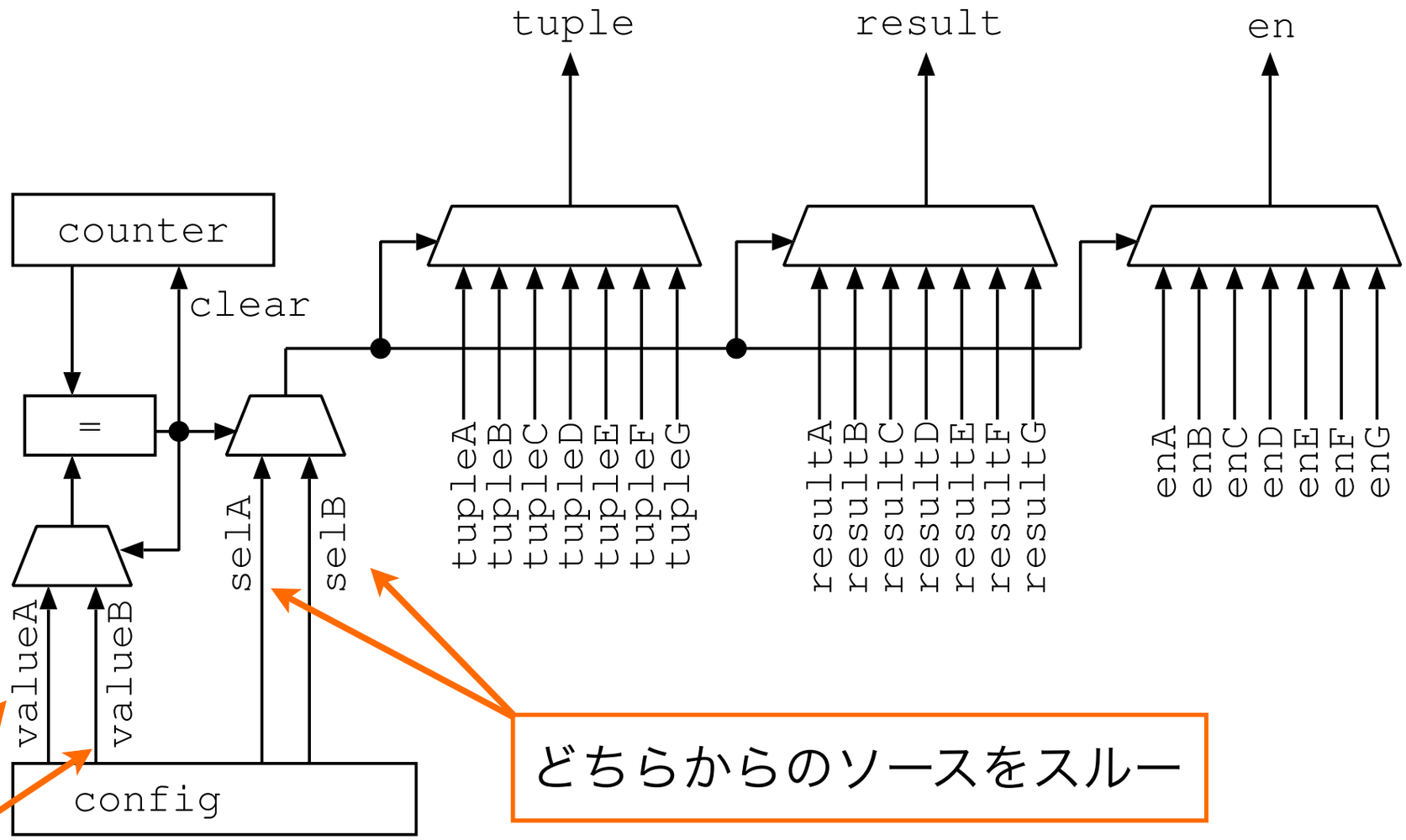
算術/論理演算の実現

スイッチボックス



7方向の接続をサポート

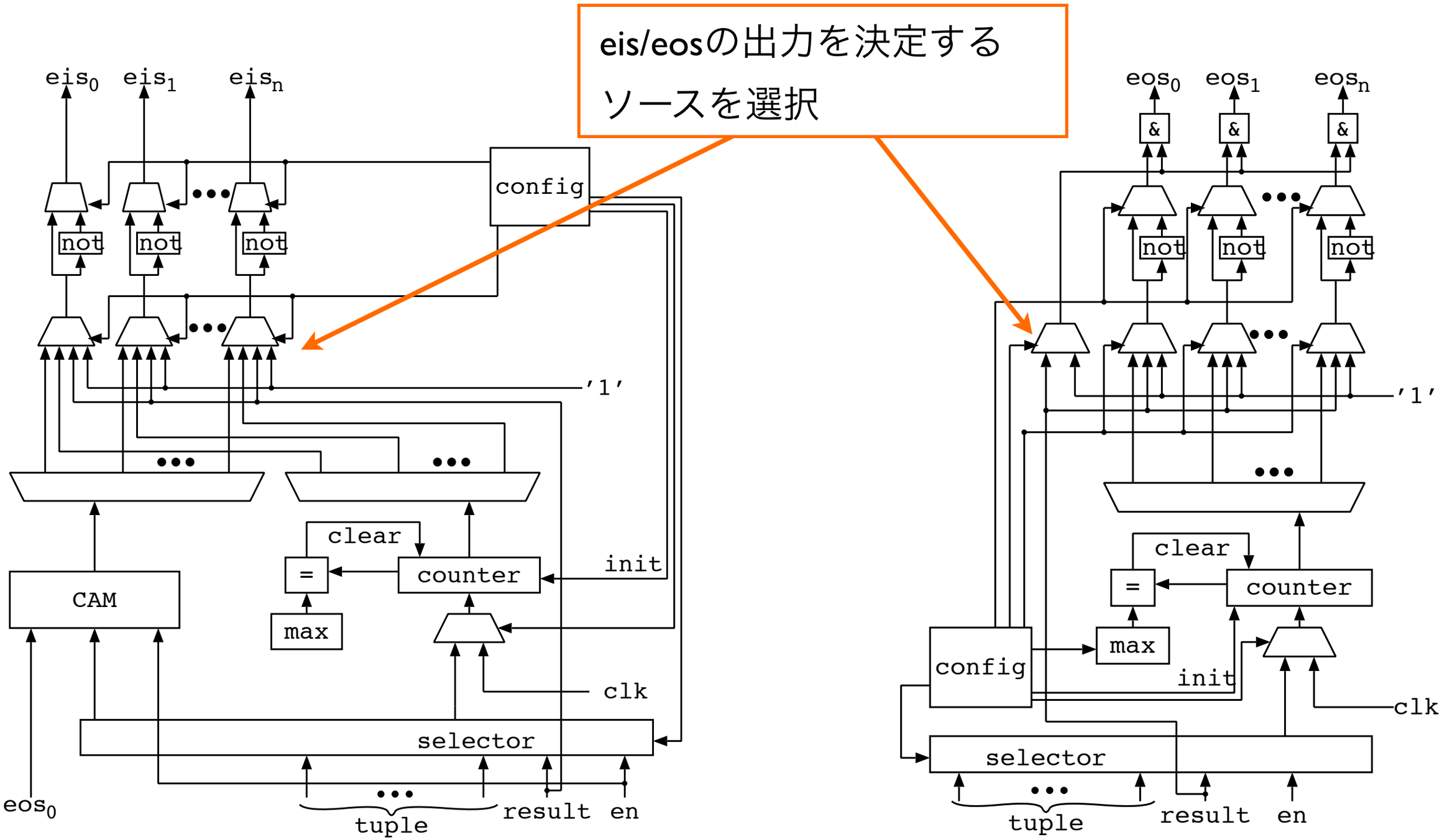
周期的な接続元の選択を実現



どちらからのソースをスルー

ソースを切り替える期間を設定

ストリーム入出力制御器



クロックあるいは入力データによってeis/eosを周期的な切り替えを実現

クエリコンパイラ

所望のクエリをDR-SPE上に構成する

- 与えられたクエリをAlgebra集合に分解
- Algebraのデータ授受関係をグラフ化
- AlgebraをのDR-SPE中の各要素に割当
  - Grouping，Windowingは同一の単位演算ユニットグループへ
- 各要素の設定値を決定する

設計したDR-SPEの評価

- HWリソースの増加量
- 最大動作周波数の低下量(→ 処理性能)
- 再構成時間

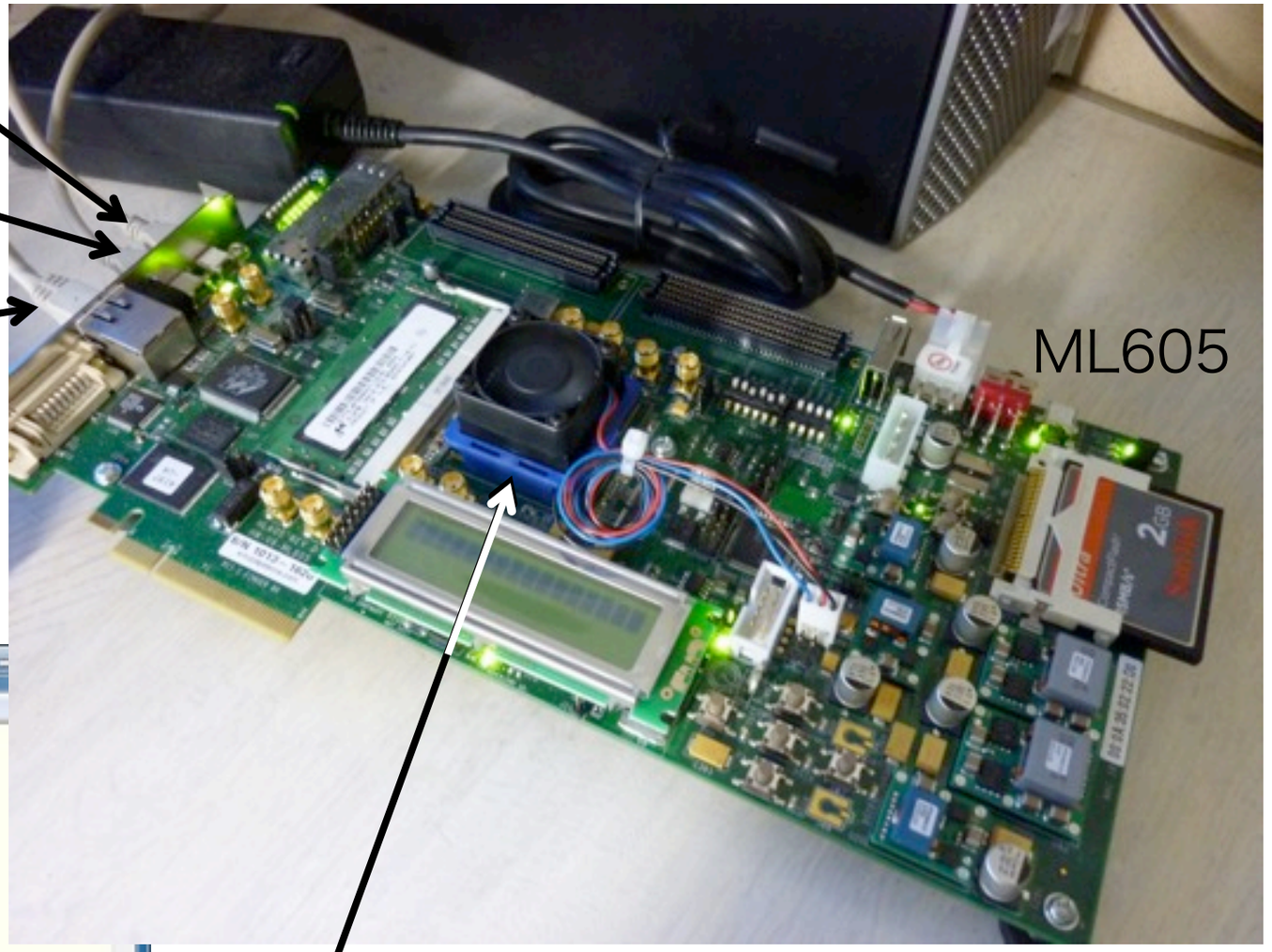
HWリソース/信号遅延の評価環境

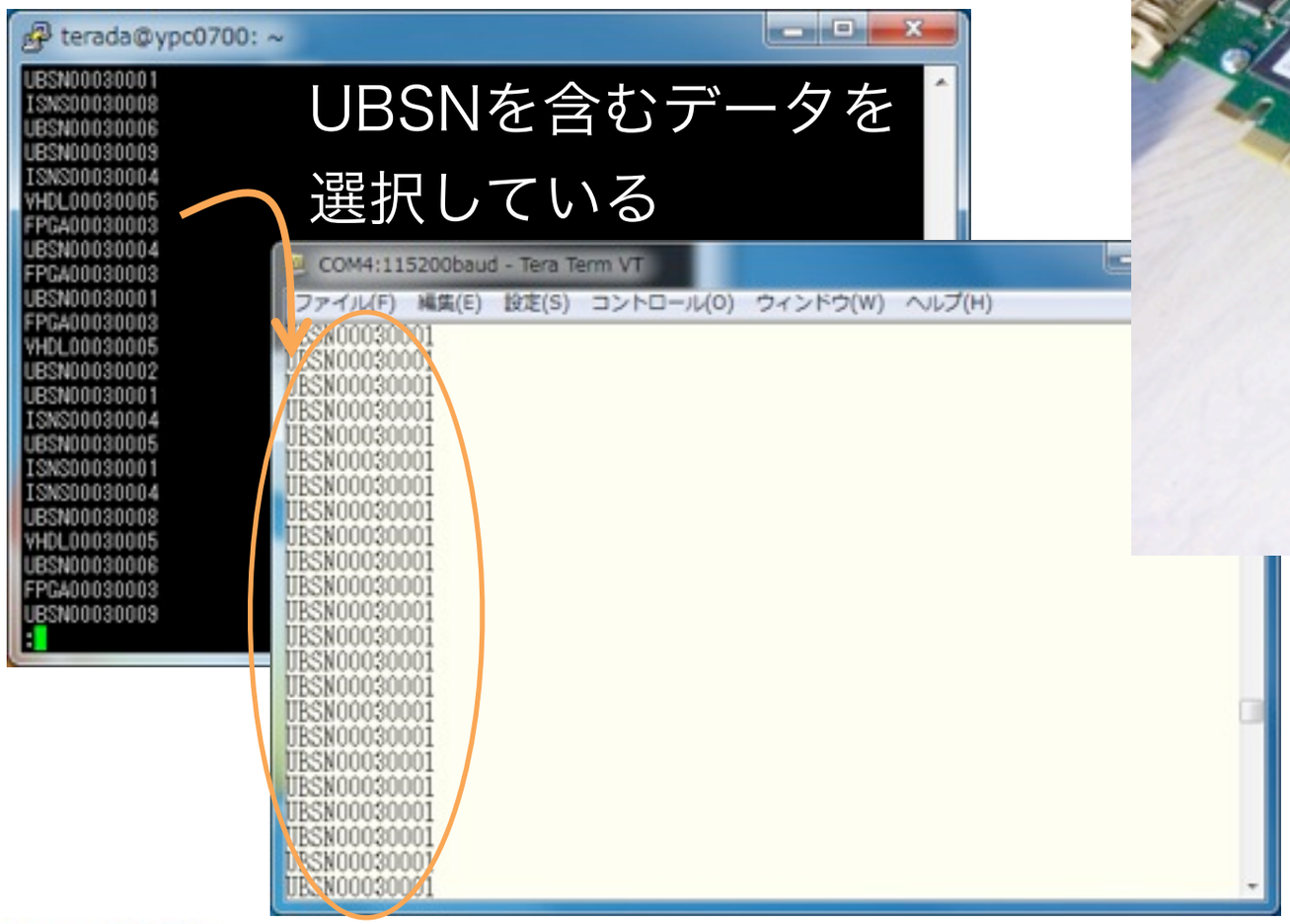
FPGA上への合成により評価

デバッグコンソール用入出力

FPGAコンフィギュレーション用JTAG

Ethernet





FPGA: Xilinx XC6VLX240T

たとえばMIPSプロセッサなら

30-50個くらい入る

合成結果(HWリソース量と信号遅延)

対象とするDR-SPEの構成

Tuple bit width	96 bit
Operator bit width	32 bit
#. of units in a block	8
#. of max union ways	8

単位演算ユニット

#. of Slice Registers	181
#.of Slice LUTs	483

スイッチボックス

#. of Slice Registers	22
#.of Slice LUTs	285

ストリーム入出力制御器

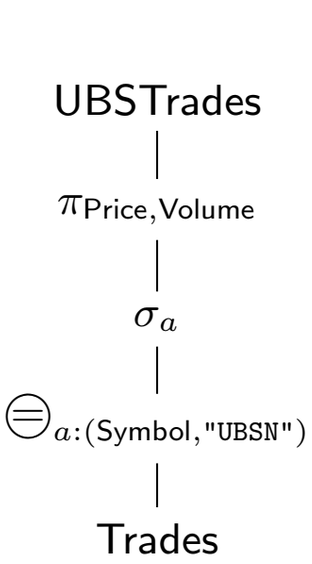
#. of Slice Registers	24
#.of Slice LUTs	74

10x10のDR-SPEのハードウェア使用量

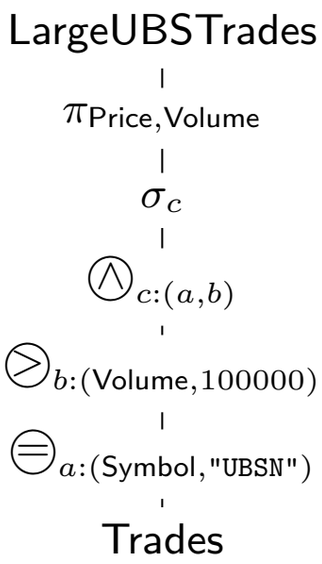
#. of Slice Registers	20,038	6%
#. of slice LUTs	88,421	56%

最大動作周波数: 172.1MHz

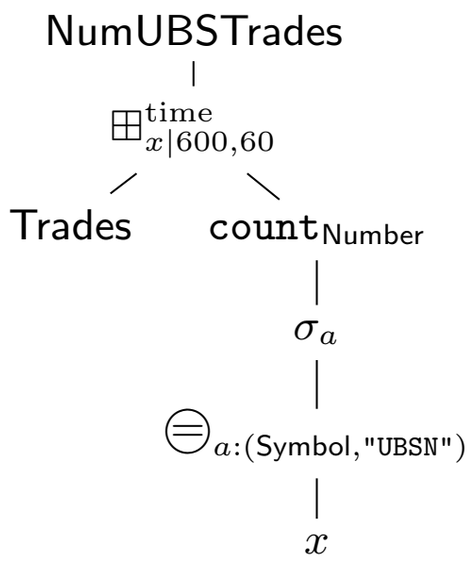
FPGAへのクエリの直接構成時との比較



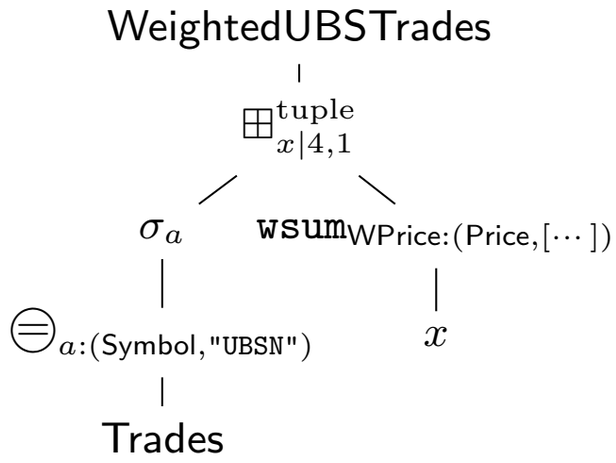
(a) Query Q_1



(b) Query Q_2



(c) Query Q_3



(d) Query Q_4

FPGA上に直接構成した場合

Query	レジスタ数/LUT数
Q1	202/8
Q2	270/10
Q3	585/272
Q4	698/283

最大動作周波数: 約200MHz
= 19200Mbps

V.S.

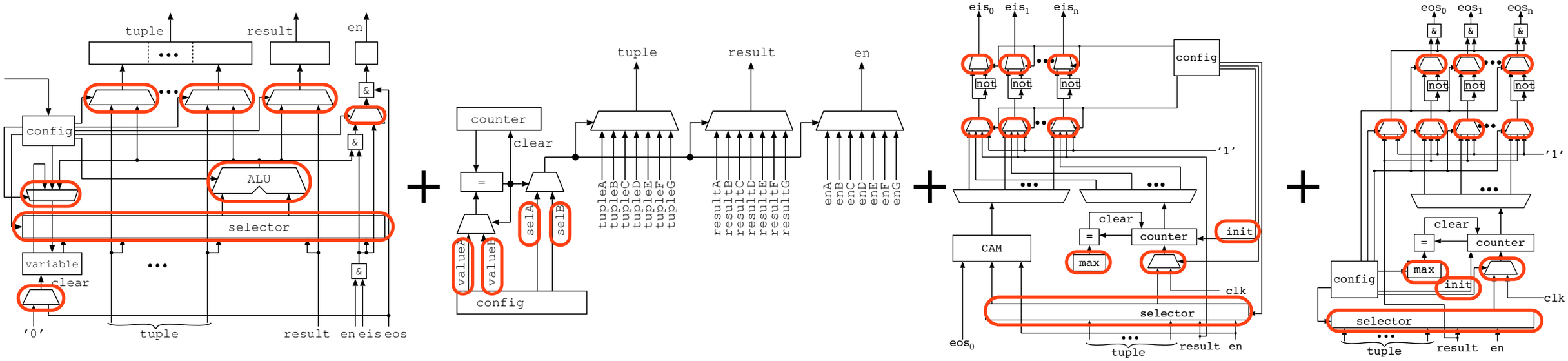
DR-SPE上に構成した場合

Query	ユニット数	レジスタ数/LUT数
Q1	2	362/966
Q2	4	724/1932
Q3	11	1991/5313
Q4	15	2715/7254

最大動作周波数: 172.1MHz
= 16521Mbps

動的再構成にかかる時間

構成要素を設定するために必要なデータサイズ



対象とするDR-SPEの構成

Tuple bit width	96 bit
Operator bit width	32 bit
#. of units in a block	8
#. of max union ways	8

= 85bit/単位演算ユニット

→ 1Mbpsでも85μ秒で設定

目次

- 背景 - ストリームとストリーム処理エンジン
- 既存手法と課題 - Streams on Wires -
- 動的再構成可能ストリーム処理エンジン
- プロセッサアーキテクチャの設計と評価
- まとめと今後の課題

まとめ

- 動的再構成可能ストリーム処理エンジンを設計
  - Streams on Wires同等の演算機能
  - クロックレベルでの高い処理性能
- HWリソース量/信号遅延/再構成時間を評価
  - HWリソース増加率 = ~4倍程度 / ~25倍程度
  - 最大動作周波数低下率 = 86%程度
  - 再構成時間 = 数十μ秒オーダ

今後の課題

- クエリコンパイラの整備
  - 配置最適化，コンパイル時間の短縮
- アーキテクチャの最適化
  - もっとさぼれるところはさぼる
  - 接続関係の見直し
- I/F，I/Oまわりの実装 → 実アプリでの評価