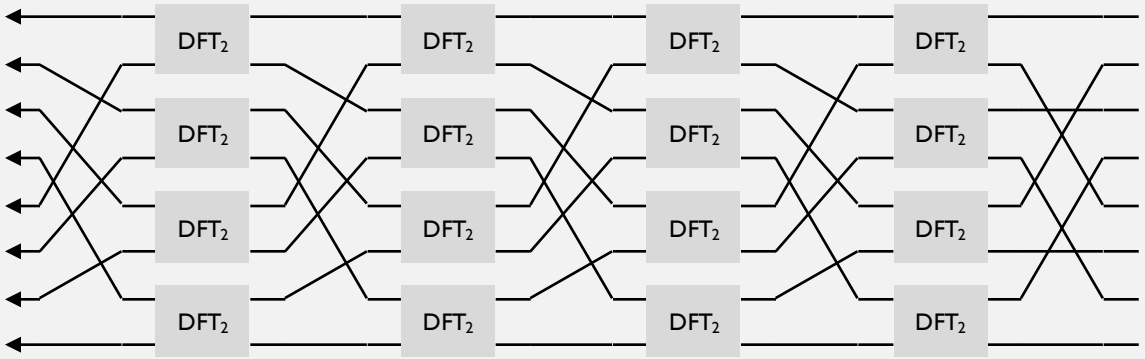
Optimal circuits for Streamed Linear Permutations Using RAM

François Serre, Thomas Holenstein, and Markus Püschel

Motivation: Folding FFTs

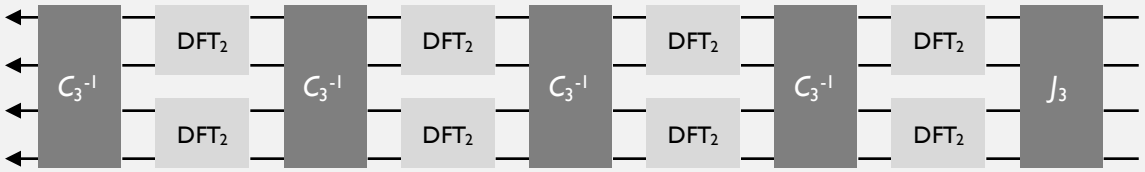
Pease FFT on n elements



Area: $O(n \log n)$

Throughput: $n \frac{\text{samples}}{\text{cycle}}$

Pease FFT on n elements, folded with w ports

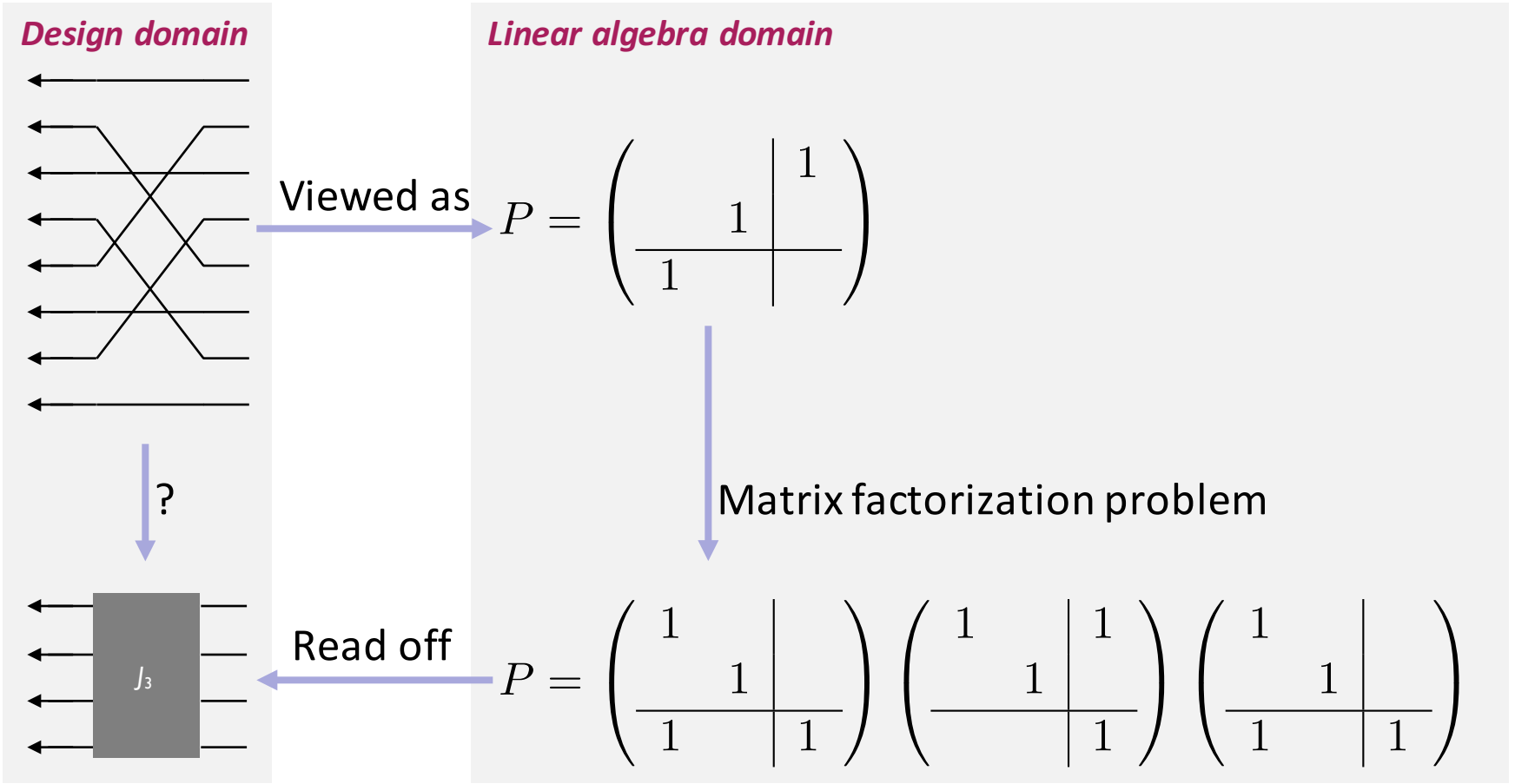


Area: $O(w \log n)$

Throughput: $w \frac{\text{samples}}{\text{cycle}}$

We generate switching-optimal Verilog designs for these permutations

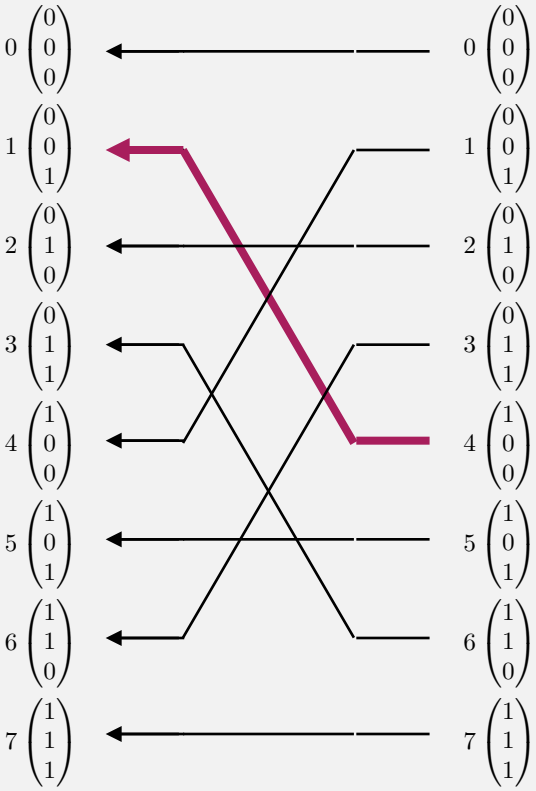
Overview of our method



Problem is phrased as a matrix factorization

Bit reversal is a “linear” permutation

Bit reversal on 8 elements



$$\begin{pmatrix}1\\0\\0\end{pmatrix} \mapsto \begin{pmatrix}0\\0\\1\end{pmatrix}$$

$$\begin{pmatrix}1\\1\\1\end{pmatrix} \mapsto \begin{pmatrix}1\\1\\1\end{pmatrix}$$

$$\begin{pmatrix}0\\1\\1\end{pmatrix} \mapsto \begin{pmatrix}1\\1\\0\end{pmatrix}$$

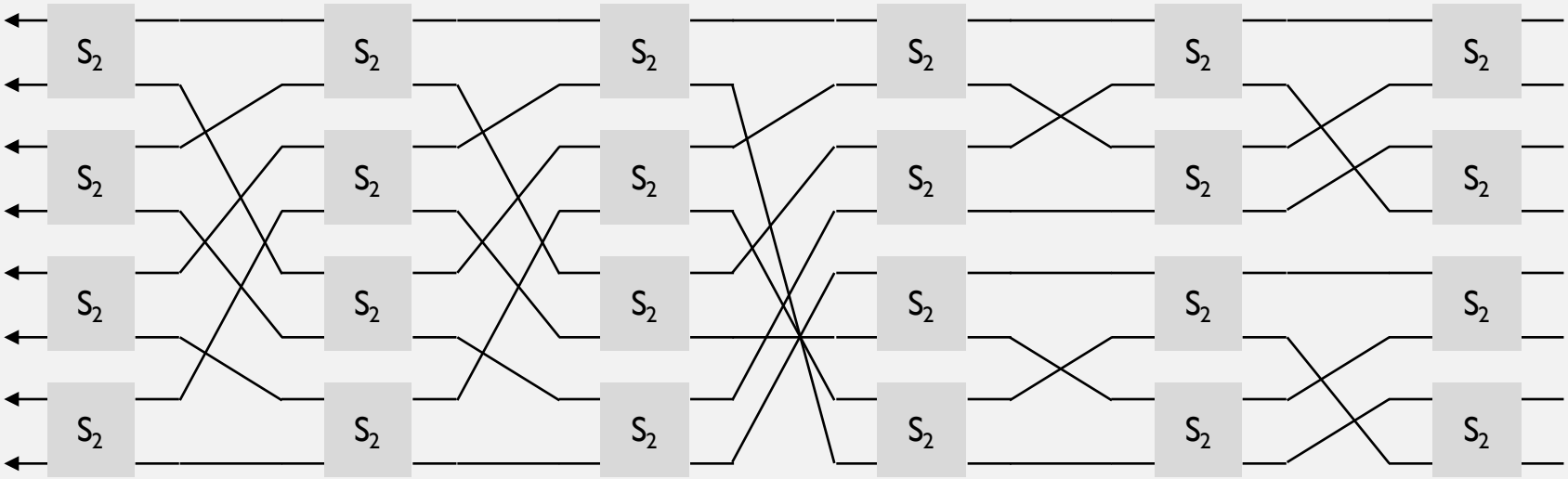
x $\qquad$ $P \cdot x$

$$P = \begin{pmatrix} & & 1 \\ & 1 & \\ 1 & & \end{pmatrix}$$

A permutation is linear if it maps linearly the bits of its addresses

Other linear permutations

A streaming sorting network (M. Zuluaga et al.)



$$\begin{pmatrix} & 1 & \\ & & 1 \\ 1 & & \end{pmatrix} \quad \begin{pmatrix} & 1 & \\ & & 1 \\ 1 & & \end{pmatrix} \quad \begin{pmatrix} 1 & 1 & \\ 1 & & 1 \\ 1 & & \end{pmatrix} \quad \begin{pmatrix} 1 & & \\ & & 1 \\ & 1 & \end{pmatrix} \quad \begin{pmatrix} 1 & & \\ & 1 & 1 \\ & 1 & \end{pmatrix}$$

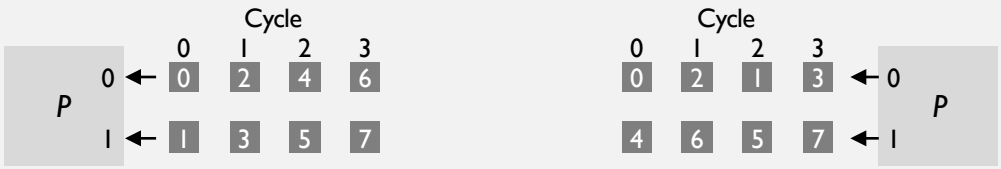
Other linear permutations

- ✔ Identity
- ✔ Bit reversal
- ✔ Stride permutations (perfect shuffle, matrix transpositions)
- ✔ Hadamard reordering
- ✔ Gray code reordering

Most permutations used in signal processing algorithms are linear

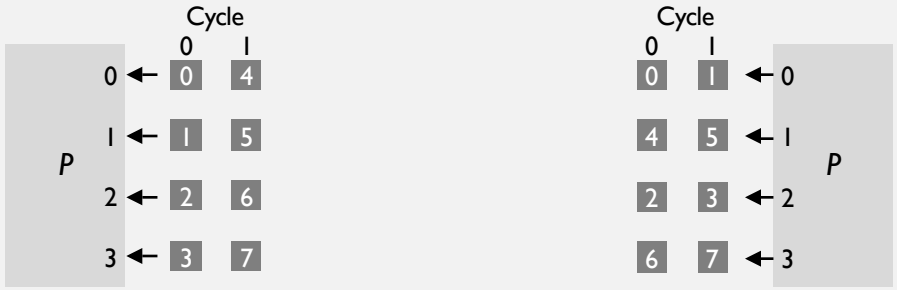
Blocking the matrix

Bit-reversal on $2^n = 8$ elements, $2^t = 4$ cycles, $2^k = 2$ ports



$$P = \begin{pmatrix} & & & 1 \\ & & 1 & \\ & 1 & & \\ 1 & & & \end{pmatrix}$$

Bit-reversal on $2^n = 8$ elements, $2^t = 2$ cycles, $2^k = 4$ ports



$$P = \begin{pmatrix} & & 1 & \\ & & & 1 \\ & 1 & & \\ 1 & & & \end{pmatrix}$$

Why?

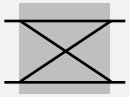
A lower bound on the number of switches

Theorem

Considering $P = \begin{pmatrix} P_4 & P_3 \\ P_2 & P_1 \end{pmatrix}$ such that P_1 is $k \times k$,

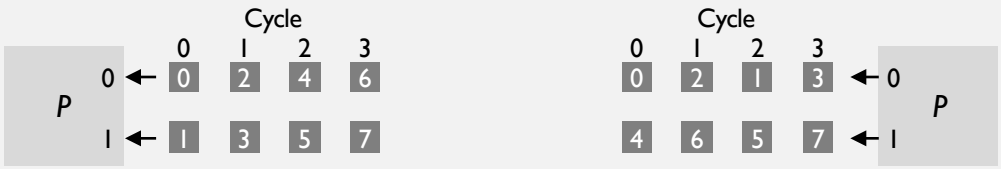
Implementing P with 2^k ports requires at least 2^{k-1} rk P_2 switches.

Proof: see paper



Blocking the matrix

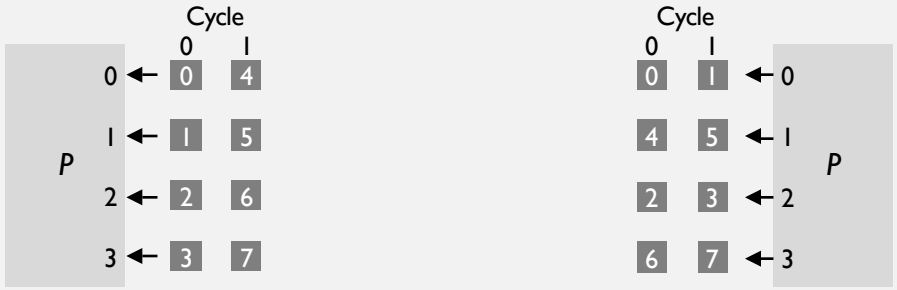
Bit-reversal on $2^n = 8$ elements, $2^t = 4$ cycles, $2^k = 2$ ports



$$P = \begin{pmatrix} & & & 1 \\ & & 1 & \\ & 1 & & \\ 1 & & & \end{pmatrix}$$

Requires 1 switch

Bit-reversal on $2^n = 8$ elements, $2^t = 2$ cycles, $2^k = 4$ ports



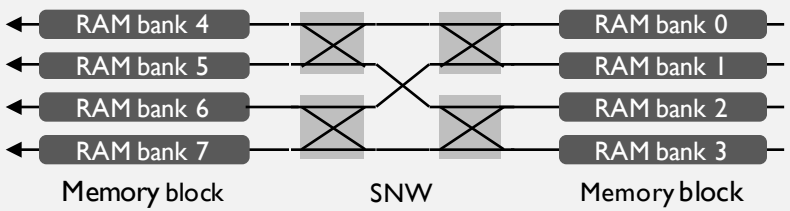
$$P = \begin{pmatrix} & & 1 & \\ & & & 1 \\ & 1 & & \\ 1 & & & \end{pmatrix}$$

Requires 2 switches

This bound is sharp!

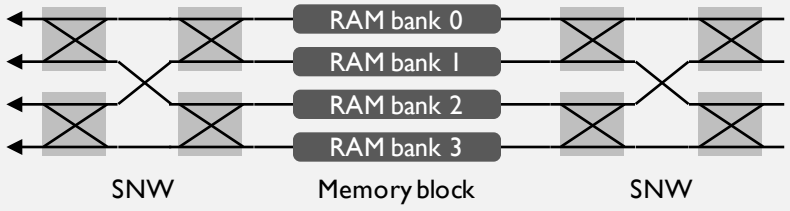
Related Work

RAM/SNW/RAM



- Milder et al., *Automatic generation of streaming datapaths for arbitrary fixed permutations*, DATE'09
- Chen et al., *Automatic generation of high throughput energy efficient streaming architectures for arbitrary fixed permutations*, FPL'15

SNW/RAM/SNW

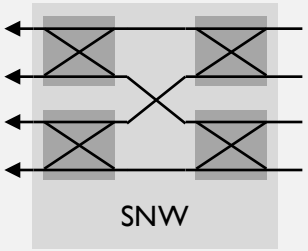


- Püschel et al., *Permuting streaming data using RAMs*, JACM'09
- Chen et al., *Energy and memory efficient mapping of bitonic sorting on FPGA*, FPGA'15

We use the same blocks

Building blocks

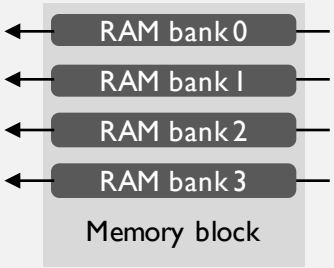
Spatial permutations – Switching network



$$\begin{pmatrix} I_{n-k} & \\ A & B \end{pmatrix}$$

Implementation uses 2^{k-1} rk A switches

Temporal permutations – Memory banks



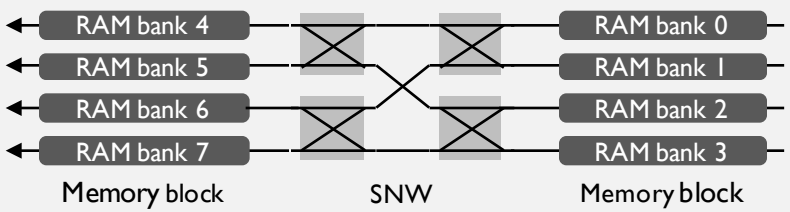
$$\begin{pmatrix} C & D \\ & I_k \end{pmatrix}$$

Total memory capacity: the dataset size

We need to decompose a general matrix into these.

General linear permutation

RAM/SNW/RAM



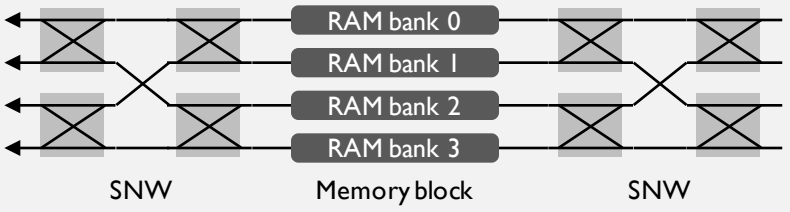
$$\begin{pmatrix} P_4 & P_3 \\ P_2 & P_1 \end{pmatrix} = \begin{pmatrix} L_4 & L_3 \\ & I_k \end{pmatrix} \cdot \begin{pmatrix} I_t & \\ C_2 & C_1 \end{pmatrix} \cdot \begin{pmatrix} R_4 & R_3 \\ & I_k \end{pmatrix}$$

2^{k-1} rk P_2 switches

Example: bit reversal: 1 switch

$$\left(\begin{array}{cc|c} & 1 & 1 \\ \hline 1 & & \end{array}\right) = \left(\begin{array}{cc|c} 1 & & 1 \\ & 1 & \\ \hline & & 1 \end{array}\right) \left(\begin{array}{cc|c} 1 & & \\ & 1 & \\ \hline 1 & & 1 \end{array}\right) \left(\begin{array}{cc|c} 1 & & 1 \\ & 1 & \\ \hline & & 1 \end{array}\right)$$

SNW/RAM/SNW



$$\begin{pmatrix} P_4 & P_3 \\ P_2 & P_1 \end{pmatrix} = \begin{pmatrix} I_t & \\ L_2 & L_1 \end{pmatrix} \cdot \begin{pmatrix} C_4 & C_3 \\ & I_k \end{pmatrix} \cdot \begin{pmatrix} I_t & \\ R_2 & R_1 \end{pmatrix}$$

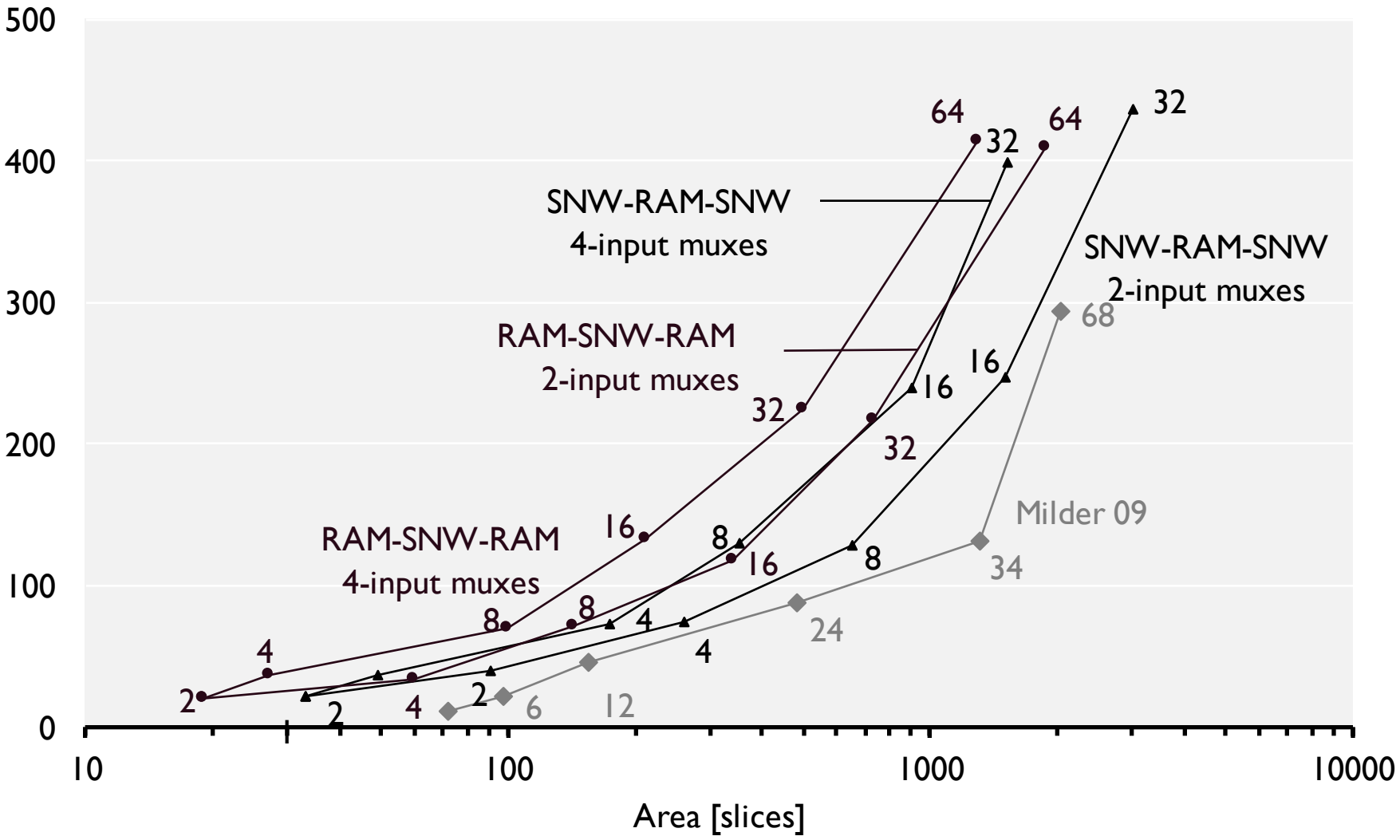
2^{k-1} max(rk P_2, n - rk P_4 - rk P_1) switches

Example: bit reversal: 2 switches

$$\left(\begin{array}{cc|c} & 1 & 1 \\ \hline 1 & & \end{array}\right) = \left(\begin{array}{cc|c} 1 & & 1 \\ & 1 & \\ \hline 1 & & 1 \end{array}\right) \left(\begin{array}{cc|c} 1 & & 1 \\ & 1 & \\ \hline & & 1 \end{array}\right) \left(\begin{array}{cc|c} 1 & & \\ & 1 & \\ \hline 1 & & 1 \end{array}\right)$$

In both cases, an optimal decomposition can be computed in $O(n^3)$

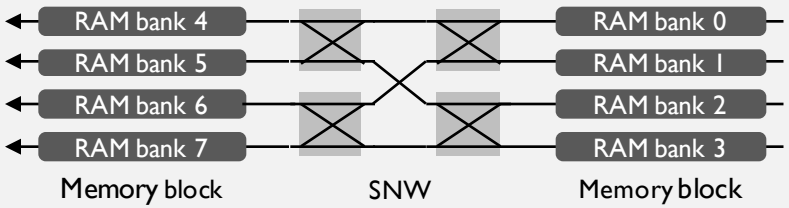
Results


Bit-reversal, 2^n = 2048 on Xilinx Virtex-7 FPGA
Throughput [Gbits/s]
500
400
300
200
100
0
10
100
1000
10000
Area [slices]
SNW-RAM-SNW
4-input muxes
SNW-RAM-SNW
2-input muxes
RAM-SNW-RAM
2-input muxes
RAM-SNW-RAM
4-input muxes
Milder 09
64
32
64
32
16
32
16
8
16
8
8
4
4
4
2
2
4
2
6
12
24
34
68

Conclusion

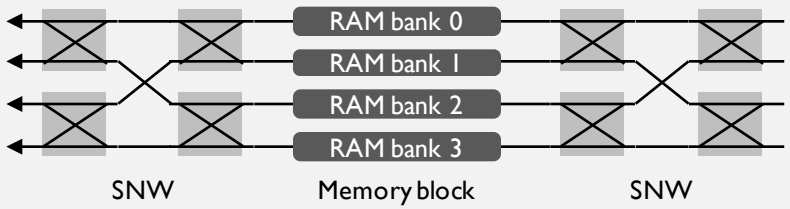
- **Exact routing complexity of streaming linear permutations**
- **Two architectures:**

RAM/SNW/RAM



- **Always switching optimal**
- **Uses a memory capacity of twice the dataset**

RAM/SNW/RAM



- **Almost always switching optimal**
- **Uses a memory capacity equal to the dataset**

Online generator: acl.inf.ethz.ch/research/SLPGen