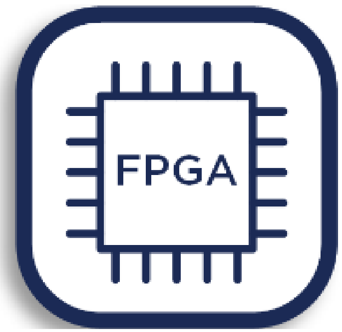
ETH zürich - Department of Computer Science

Memory-Efficient Fast Fourier Transform on Streaming Data by Fusing Permutations

François Serre and Markus Püschel

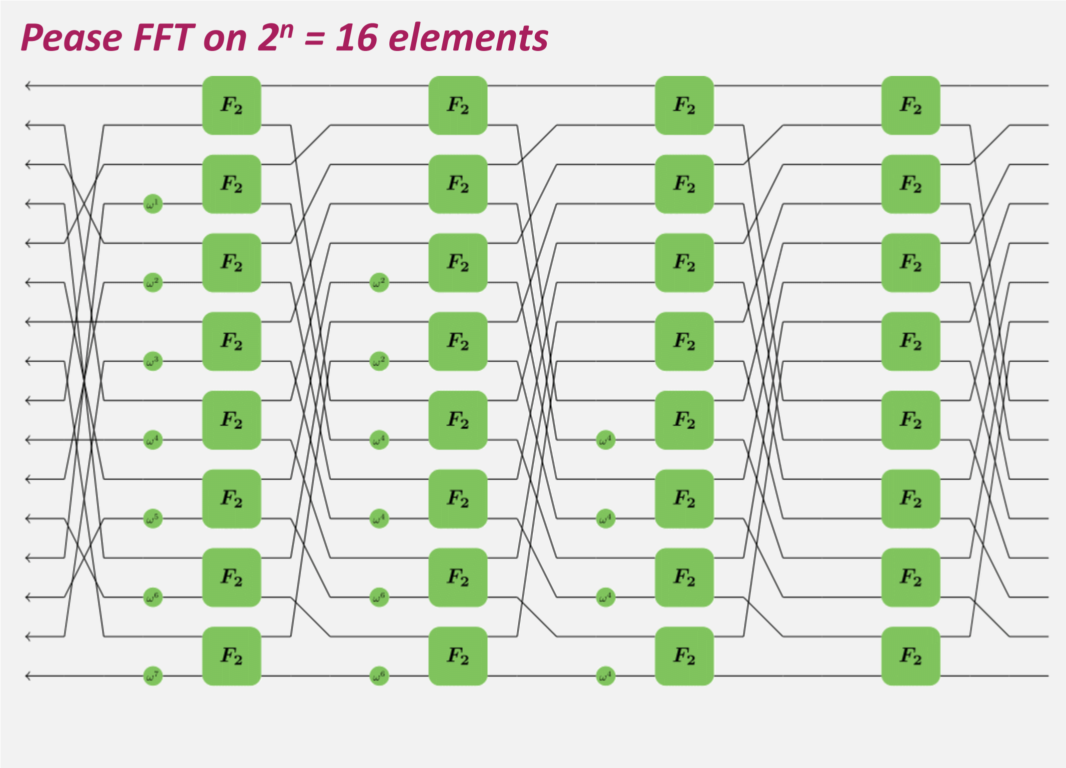
Motivation

FFT



We focus on the implementation of small, but performant designs

Motivation: Implementing FFTs

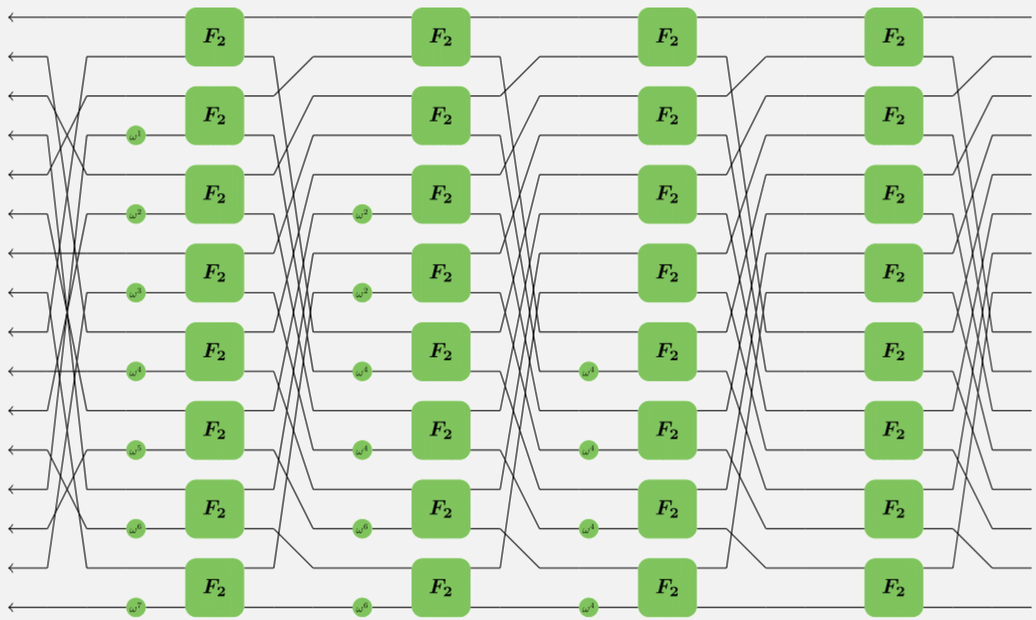


High throughput (1 transform per cycle), but high use of ressources!

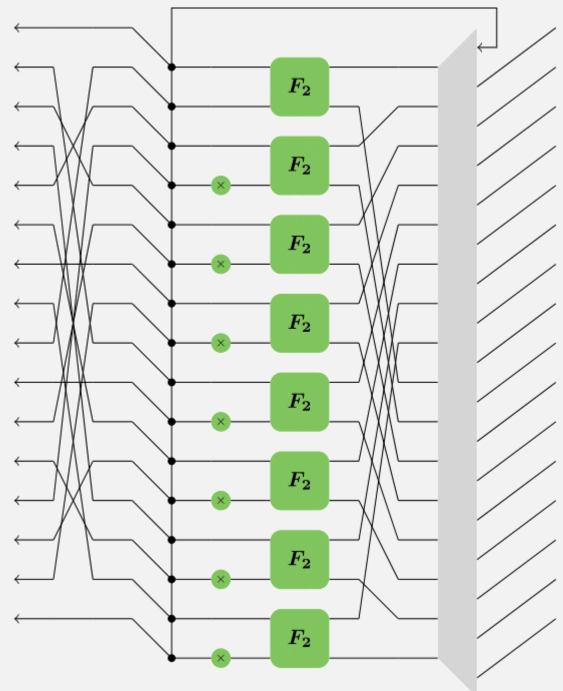

We have full-thrust…
…but other positions are missing

Motivation: Implementing FFTs [Milder et al., TODAES12]

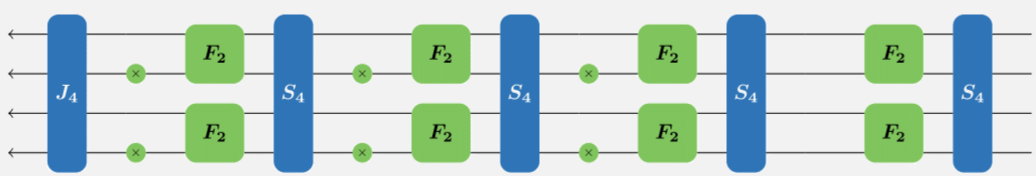
Pease FFT on 2^n = 16 elements



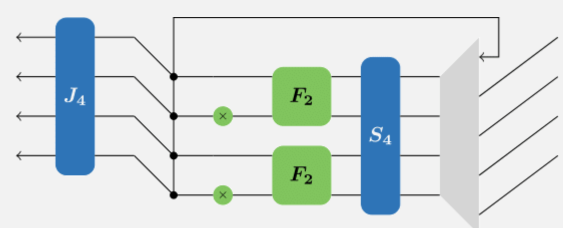
Iterative reuse

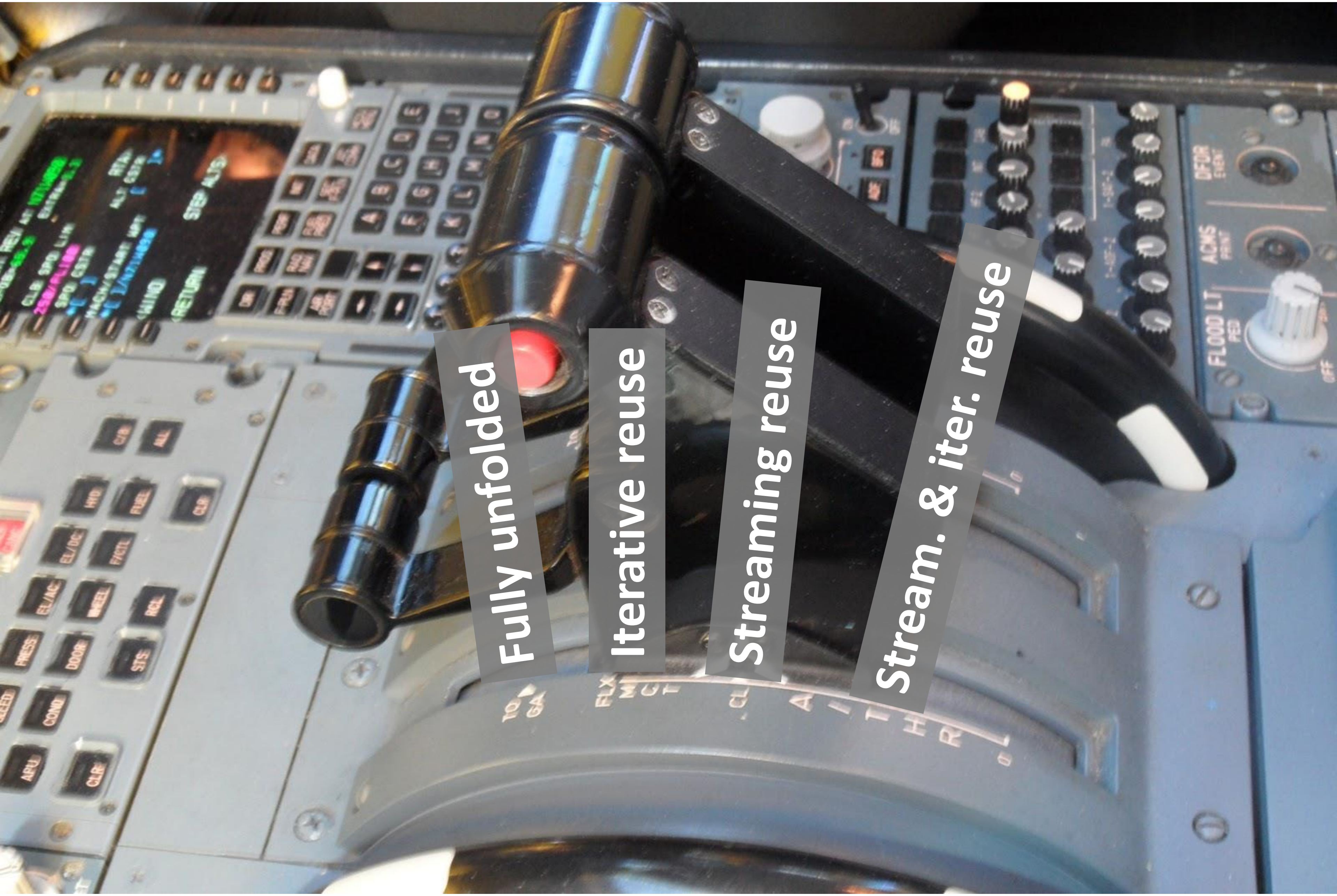


Streaming reuse with a streaming width of 2^k = 4

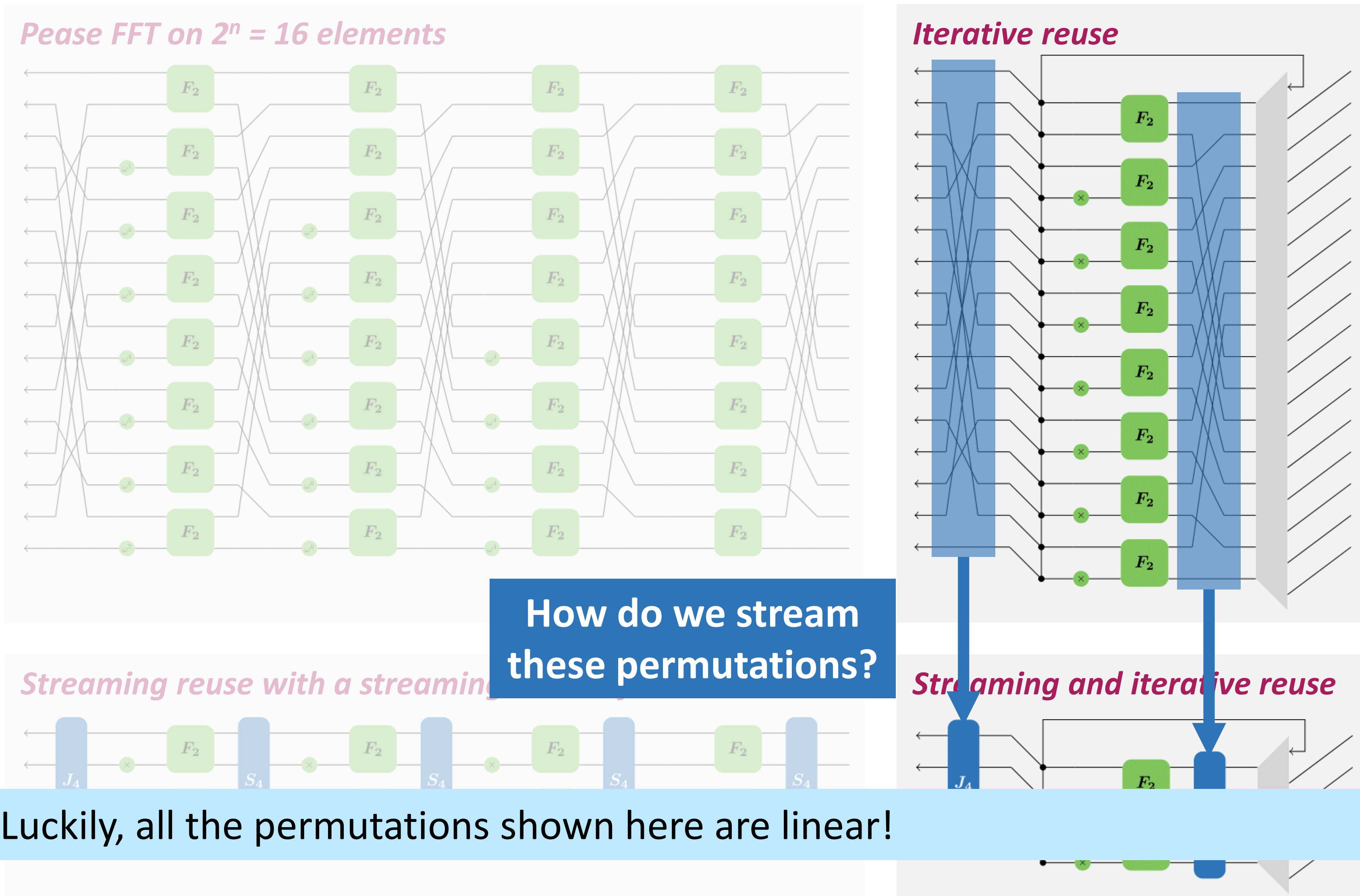


Streaming and iterative reuse




Fully unfolded
Iterative reuse
Streaming reuse
Stream. & iter. reuse

Motivation: Implementing FFTs [Milder et al., TODAES12]

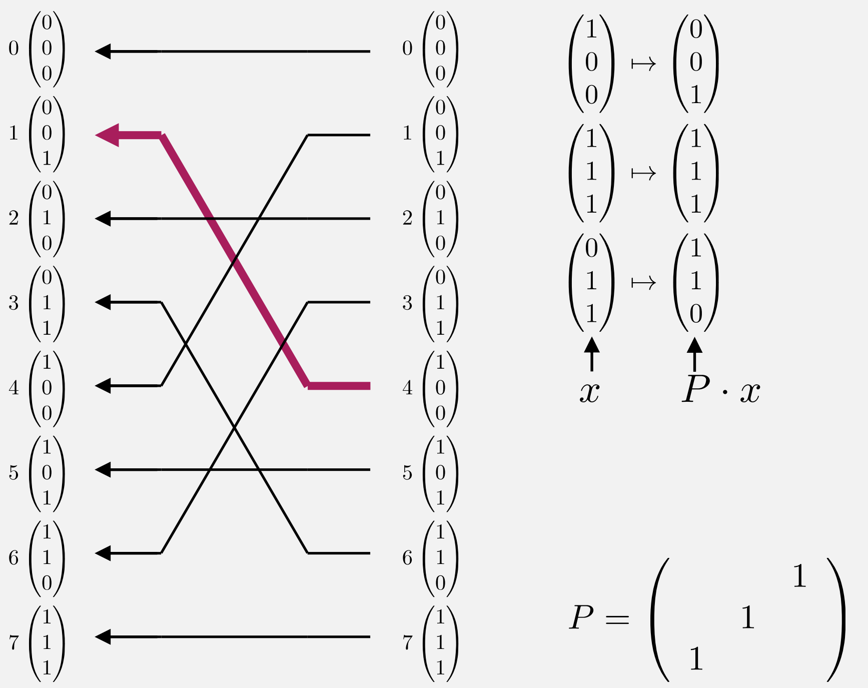


Luckily, all the permutations shown here are linear!

Streaming Linear permutation [Serre/Hollenstein/Püschel, FPGA16]

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

Bit-reversal on 8 elements

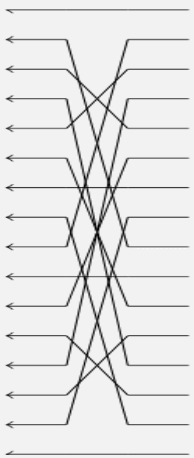


Almost all permutations in DSP algorithms are linear:

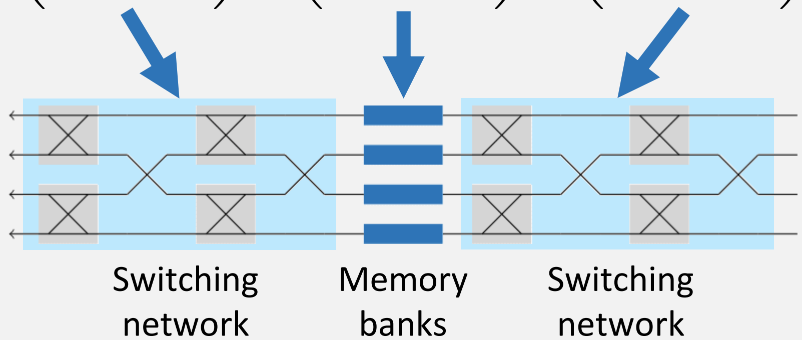
- **Identity**
- **Perfect shuffle**
- **Stride permutations**
- **Hadamard reordering**
- **Gray code reordering**

Streaming Linear permutation [Serre/Hollenstein/Püschel, FPGA16]

Bit reversal on 16 elements



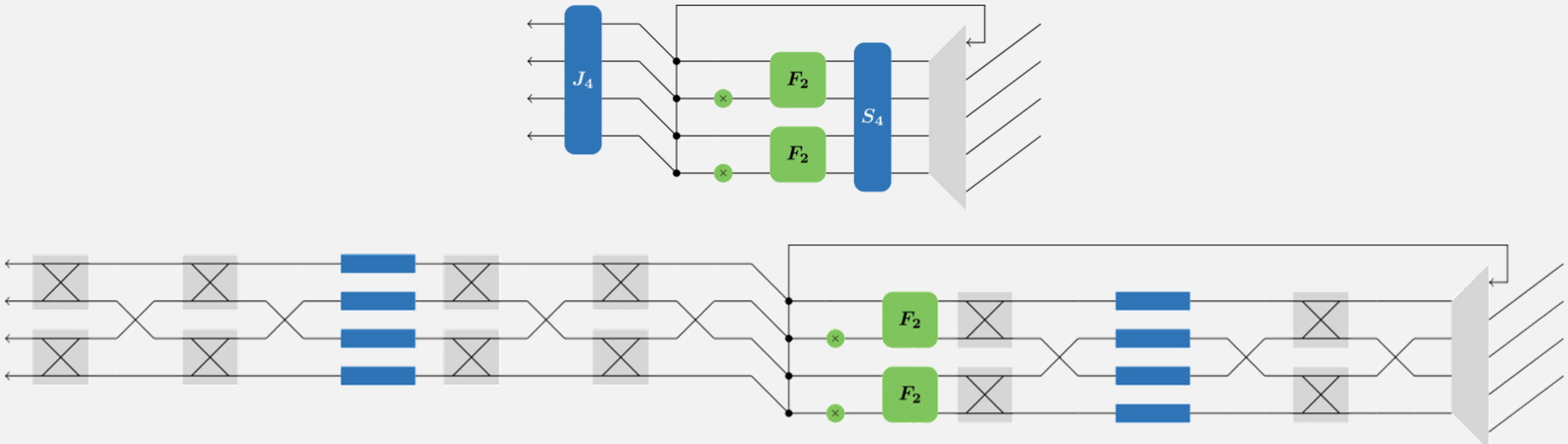
$$\begin{pmatrix} P_4 & P_3 \\ P_2 & P_1 \end{pmatrix} = \begin{pmatrix} I_t & \\ L_2 & L_1 \end{pmatrix} \cdot \begin{pmatrix} C_4 & C_3 \\ & I_k \end{pmatrix} \cdot \begin{pmatrix} I_t & \\ R_2 & R_1 \end{pmatrix}$$



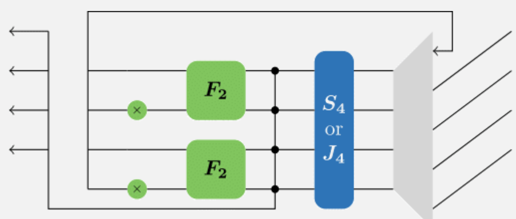
- **Minimal number of RAM banks** [Koehn/Athanas, ICCAD16] Corollary 1
- **Minimal latency on the temporal part** [Koehn/Athanas, ICCAD16] Lemma 2
- **Minimal RAM capacity** [Koehn/Athanas, ICCAD16] Section 2
- **Minimal number of switches** [Serre/Püschel, LAA16]
- **Lightweight control (And/Xor based)**
- **Limitation: single linear permutation**

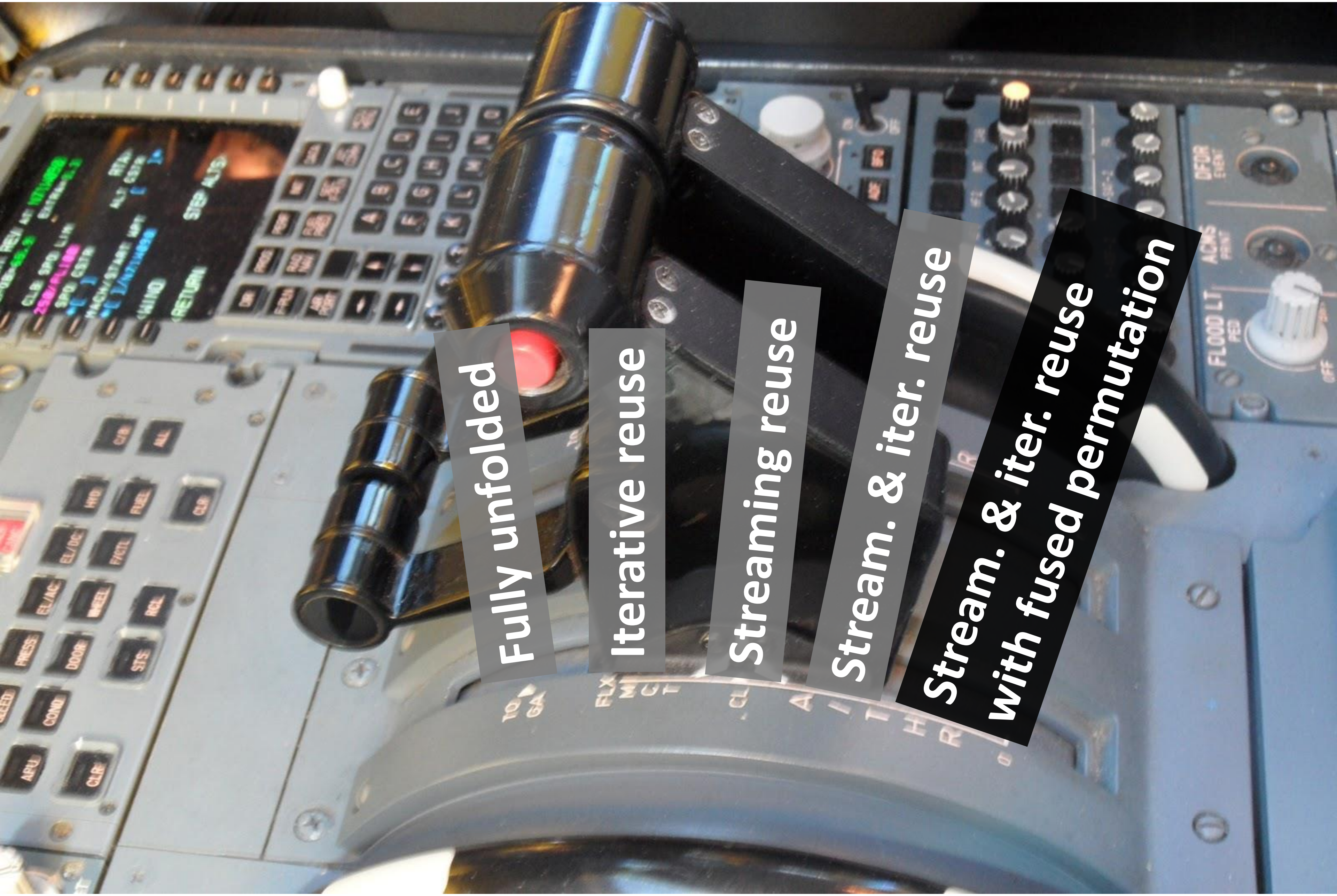
Fusing linear permutations

Streaming and iterative reuse



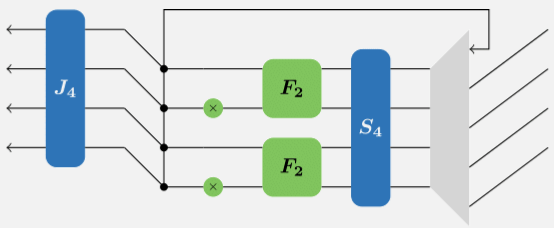
Streaming and iterative reuse with fused permutation

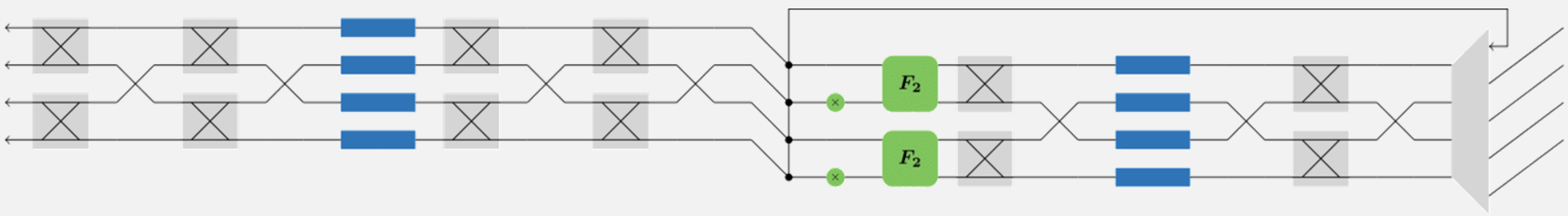



Fully unfolded
Iterative reuse
Streaming reuse
Stream. & iter. reuse
Stream. & iter. reuse with fused permutation

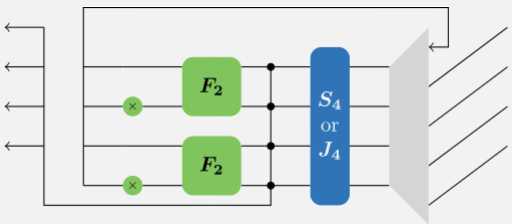
Fusing linear permutations

Streaming and iterative reuse





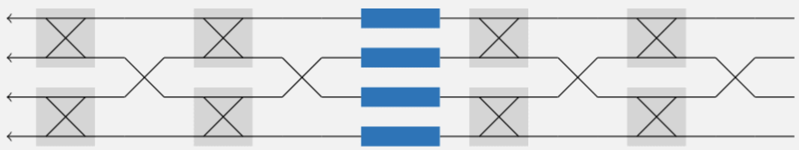
Streaming and iterative reuse with fused permutation



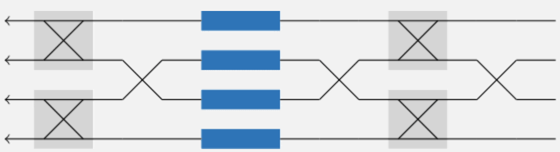
A linear permutation datapath only works with a SINGLE permutation!

Fusing linear permutations

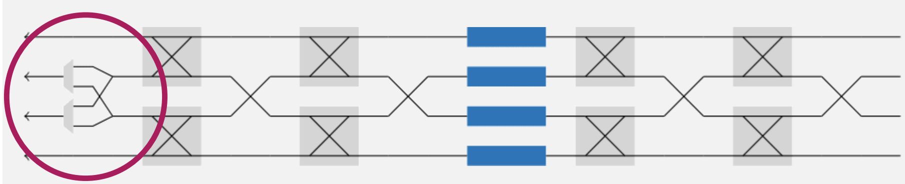
Streaming bit-reversal (8 elements)



Streaming perfect-shuffle (8 elements)



Streaming bit-reversal and perfect-shuffle

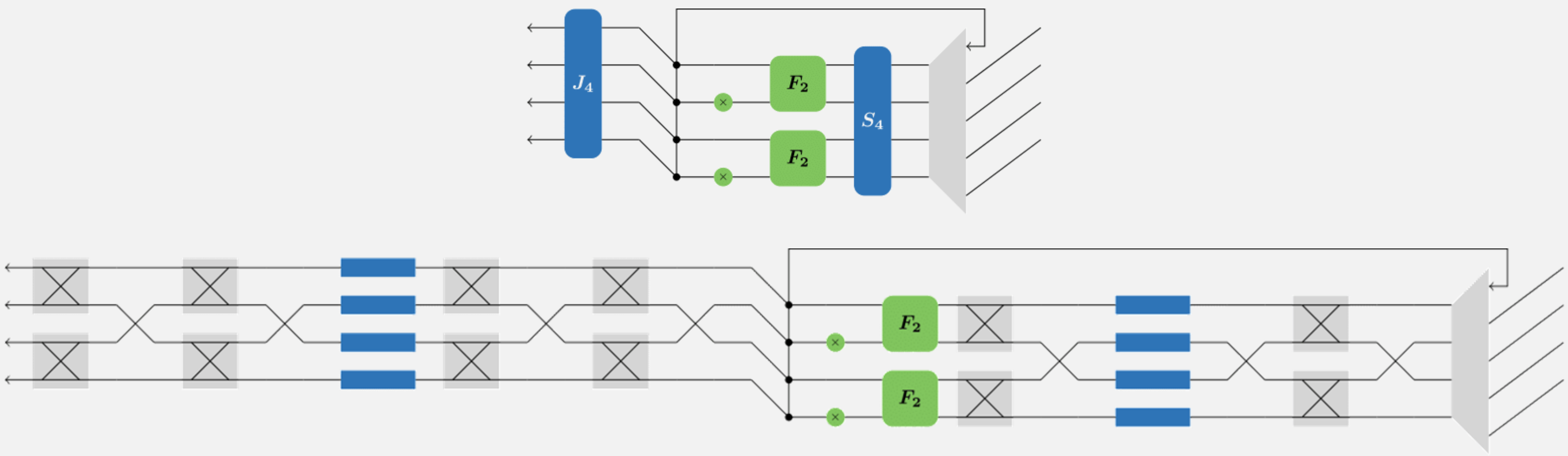


Recipe:

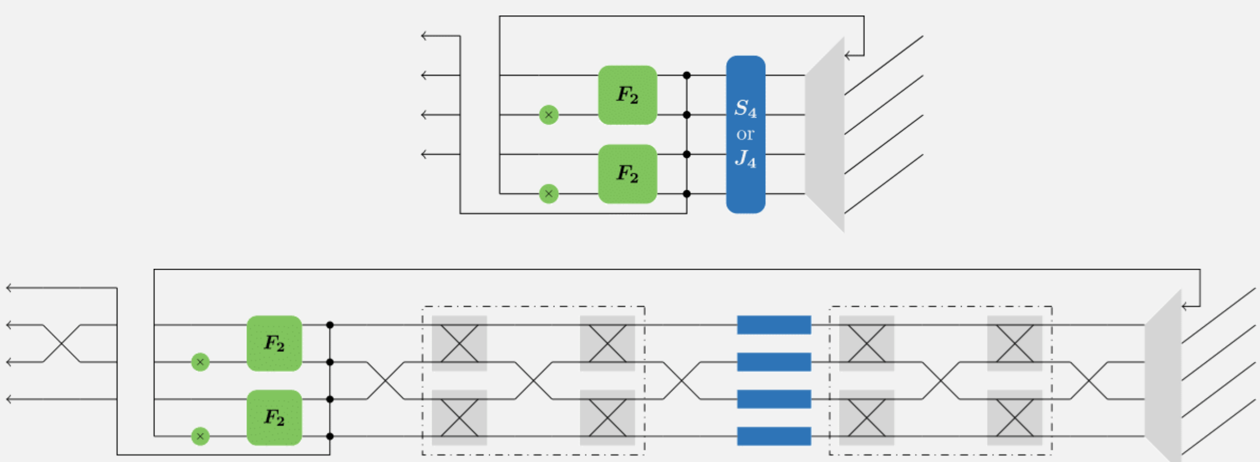
- **Use the matrix decomposition on both permutations**
- **Group temporal permutations**
  - **Easy in theory**
  - **«Touchy» in practice…**
- **Implement switching networks handling several permutations (details in paper…)**

Fusing linear permutations

Streaming and iterative reuse



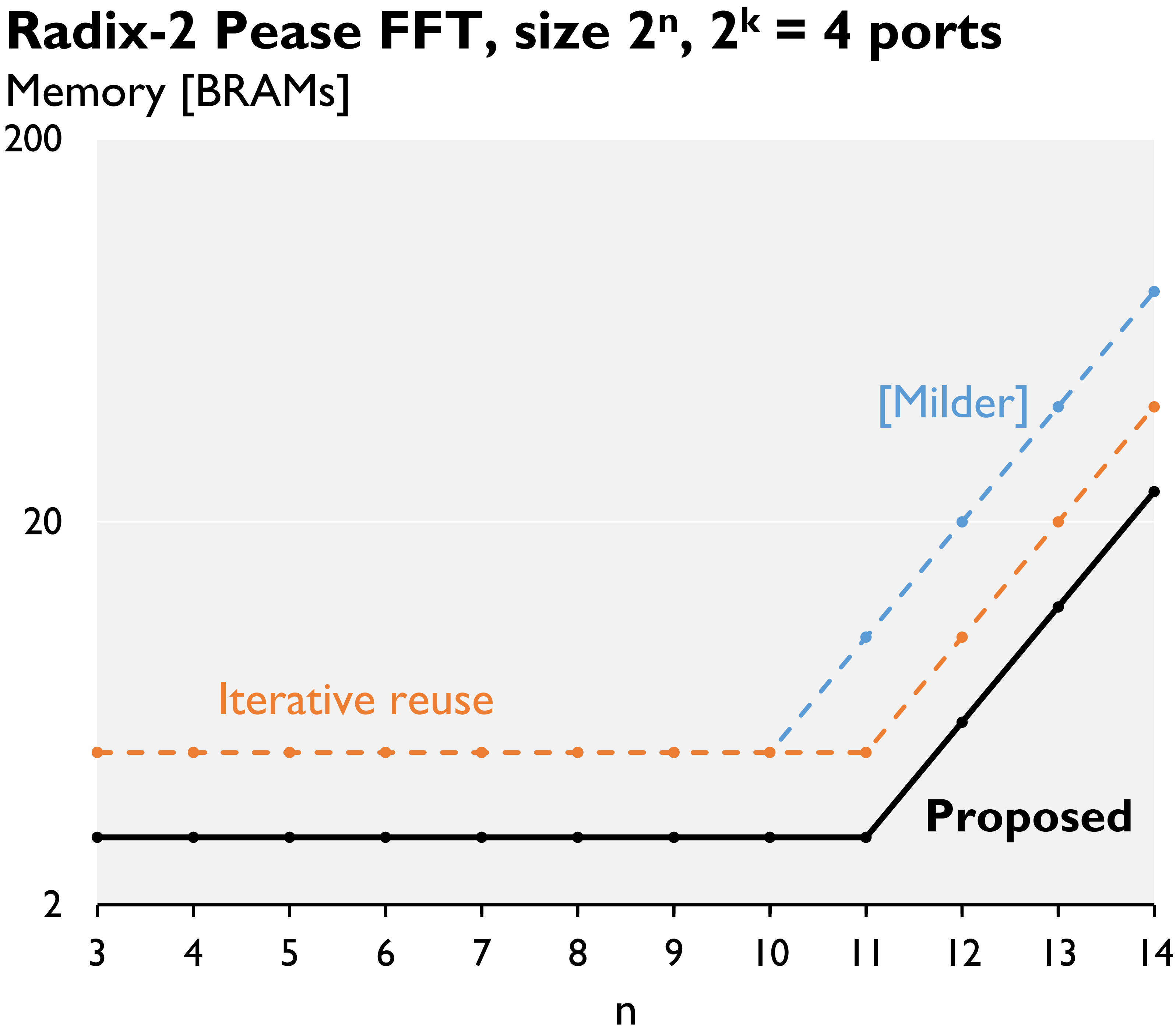
Streaming and iterative reuse with fused permutation



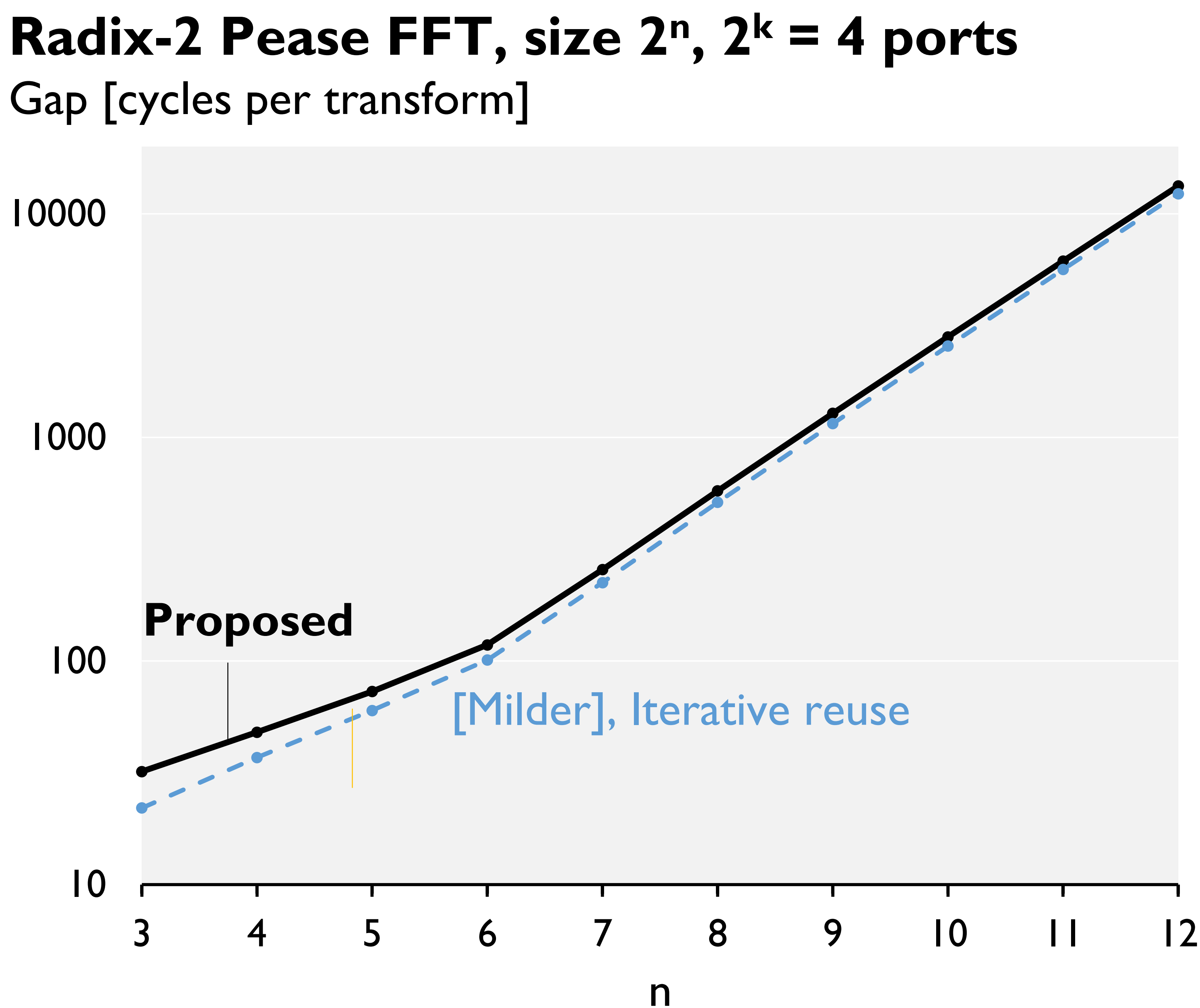
Results

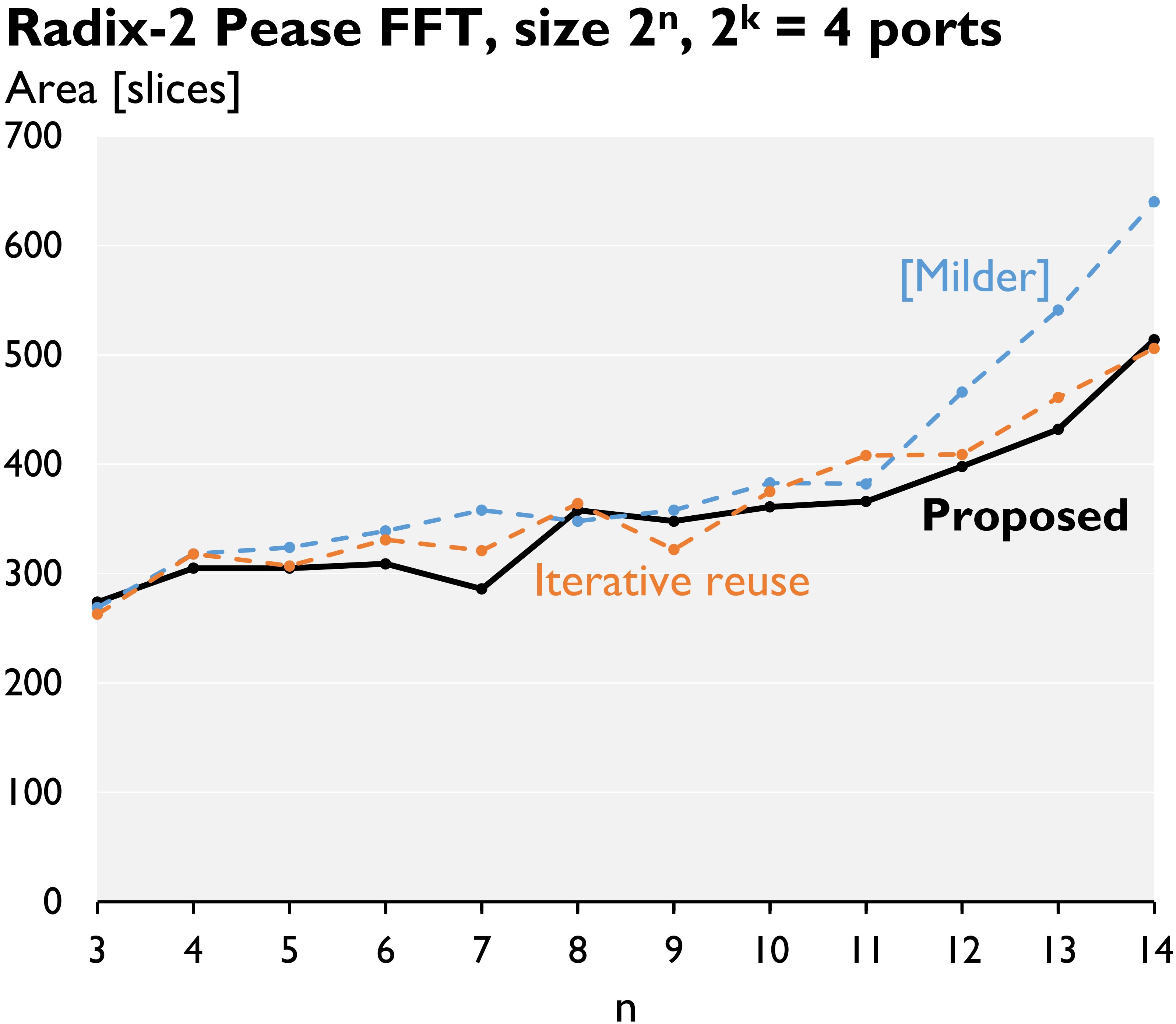
- 16 bits fixed-point
- Targeting ±400MHz on a Virtex 7
- Vivado 2014.4 after place and route
- Temporal permutations explicitly requested as BRAMs

Other measurements are available on the generator website



Radix-2 Pease FFT, size 2ⁿ, 2ᵏ = 4 ports
Memory [BRAMs]
200
20
2
[Milder]
Iterative reuse
Proposed
3
4
5
6
7
8
9
10
11
12
13
14
n


Radix-2 Pease FFT, size 2^n, 2^k = 4 ports
Gap [cycles per transform]
10000
1000
100
10
Proposed
[Milder], Iterative reuse
3
4
5
6
7
8
9
10
11
12
n



Radix-2 Pease FFT, size 2^n, 2^k = 4 ports
Area [slices]
700
600
500
400
300
200
100
0
3
4
5
6
7
8
9
10
11
12
13
14
n
[Milder]
Proposed
Iterative reuse

Limitations of the generator

- Twiddle factors storage is not optimized
- Many features of the Spiral generator are not supported yet
- Pipelining decisions are made with a basic heuristic

Related work

- **Generic streaming permutation methods can be used instead:**
  - **Koehn/Athanas, ICCAD16**
  - **Milder et al., DATE09**
  - **Parhi, IEEE Trans. CAS II 92 (register based)**
- **Specific streaming permutations:**
  - **Järvinen et al., ASAP04 (stride permutation, register based)**
  - **Garrido et al., TCAS II 17 (bit-reversal, register based)**

Generator for linear permutations/FFTs and benchmarks: https://acl.inf.ethz.ch/research/hardware