

# FFTTA

Low-Power Programmable Processor for Fast Fourier Transform Based on Transport Triggered Architecture

May 14, 2019

Jakub Žádník & Jarmo Takala

{jakub.zadnik, jarmo.takala}@tuni.fi

Tampere University, Tampere, FINLAND

# The Goal

Design a FFT processor that is:

- ▶ Programmable
- ▶ Low-enough-power to be comparable with ASIC

Improving previous work<sup>1</sup> by decreasing instruction memory power consumption.

---

<sup>1</sup>T. Pitkänen. (2014). *Fast Fourier Transforms on Energy-Efficient Application-Specific Processors*. PhD thesis, Tampere University of Technology.

Low-Power Programmable Processor  
for **Fast Fourier Transform**  
Based on Transport Triggered Architecture

# Fourier Transform (DFT vs. FFT)

Discrete Fourier Transform (DFT):

$$X(k) = \frac{1}{N} \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N} = \frac{1}{N} \sum_{n=0}^{N-1} x(n) \overbrace{W_N^k}^{\text{twiddle factor}}{}^n$$

Fast Fourier Transform (FFT):

$$N = 2^q, \quad q = 1, 2, 3, \dots$$

# Fourier Transform (twiddle factors)

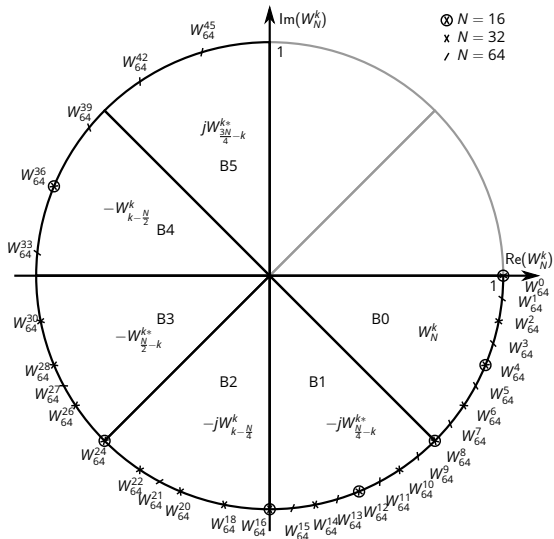


Figure: Twiddle factors of FFT sizes 8,16 and 64. Only  $N/8 + 1$  twiddle factors need to be stored (B0)

# Fourier Transform (butterflies)

- ▶ Possible to break down large FFT to small elementary FFTs
- ▶ Radix-4  $\Rightarrow$  support for FFT sizes  $4^k$  only
- ▶ Radix-2  $\Rightarrow$  support for FFT sizes  $2^k$

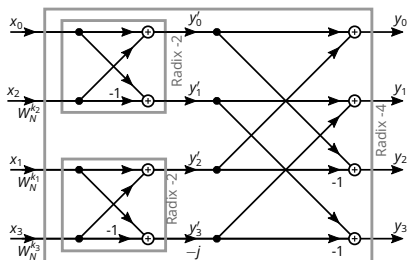


Figure: Radix-4 and radix-2 butterflies - the base computation units

Radix-2:

$$y'_0 = x_0 + W_N^{k_2} x_2 ; y'_2 = x_1 + W_N^{k_1} x_3$$

$$y'_1 = x_0 - W_N^{k_2} x_2 ; y'_3 = x_1 - W_N^{k_3} x_3$$

Radix-4:

$$y_0 = x_0 + W_N^{k_1} x_1 + W_N^{k_2} x_2 + W_N^{k_3} x_3$$

$$y_1 = x_0 - jW_N^{k_1} x_1 - W_N^{k_2} x_2 + jW_N^{k_3} x_3$$

$$y_2 = x_0 - W_N^{k_1} x_1 + W_N^{k_2} x_2 - W_N^{k_3} x_3$$

$$y_3 = x_0 + jW_N^{k_1} x_1 - W_N^{k_2} x_2 - jW_N^{k_3} x_3$$

## Example run: in-place computation (N=16)

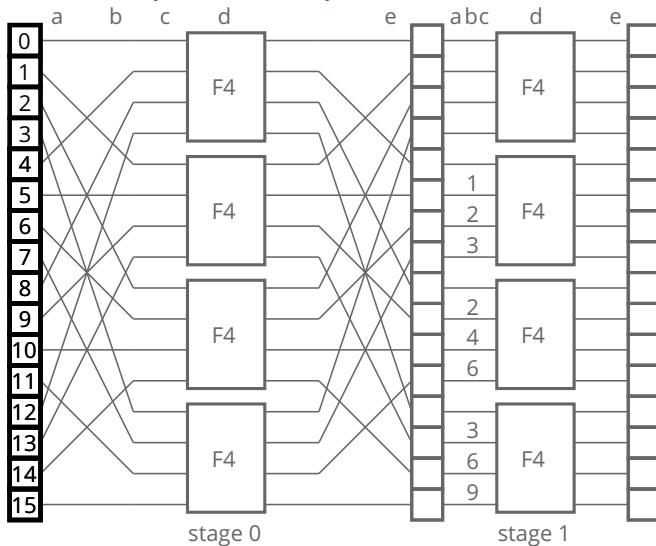
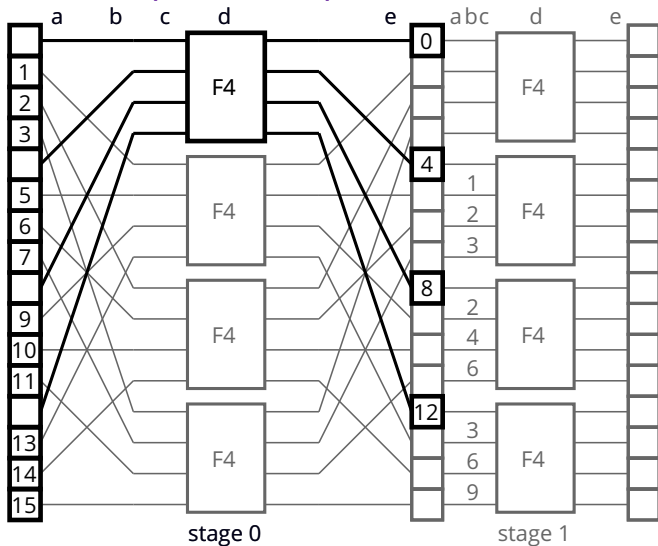


Figure: Starting with an input array of complex numbers in a memory

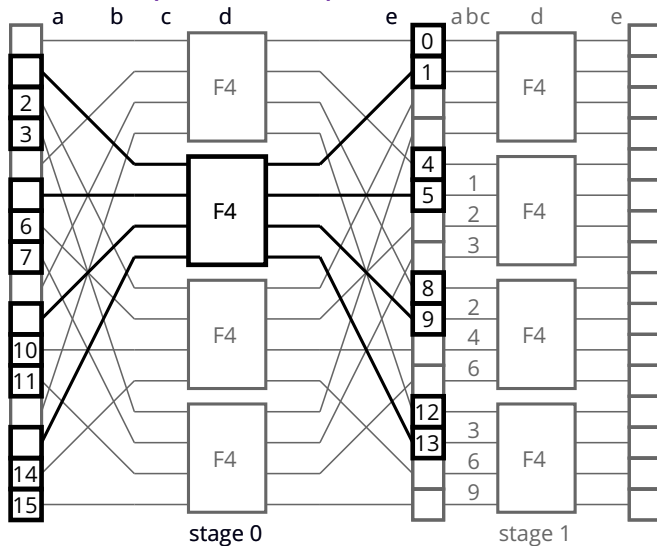
## Example run: in-place computation (N=16)



**Figure:** a - memory read x; b - twiddle factor read; c - multiply x with tf; d - butterfly; e - memory write back

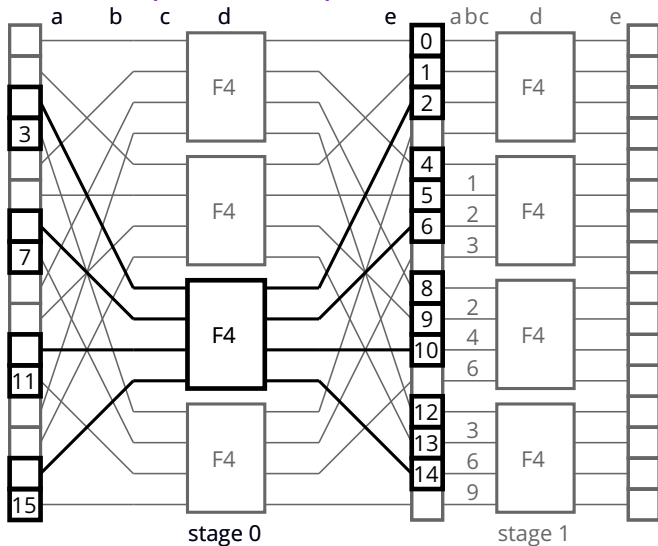


## Example run: in-place computation (N=16)



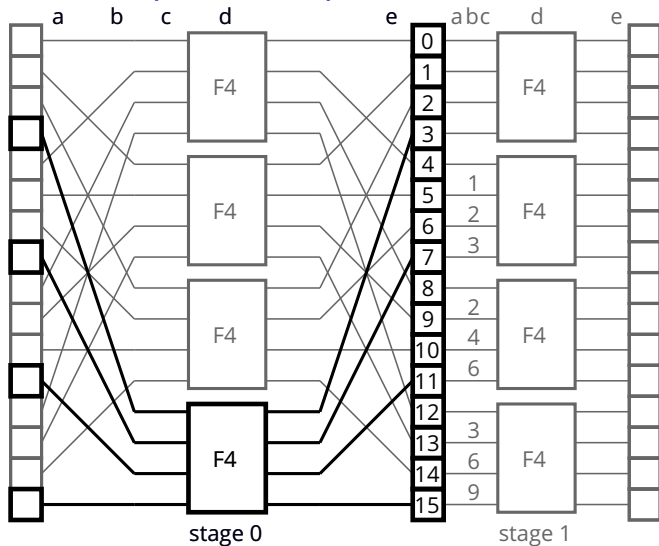
**Figure:** a - memory read  $x$ ; b - twiddle factor read; c - multiply  $x$  with  $tf$ ; d - butterfly; e - memory write back

## Example run: in-place computation (N=16)



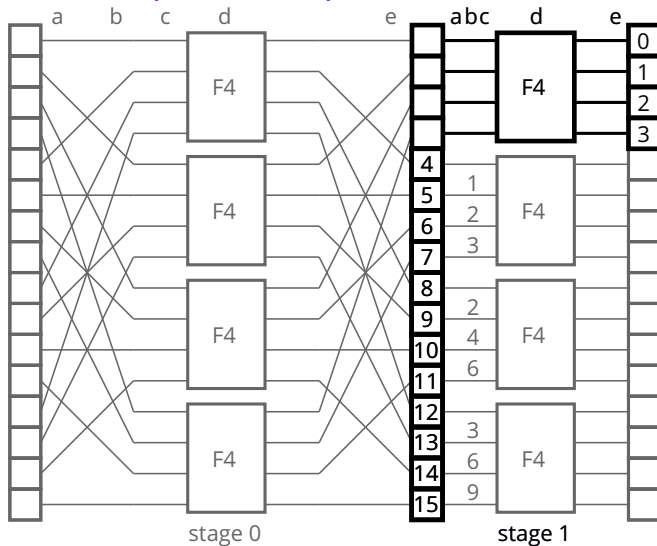
**Figure:** a - memory read  $x$ ; b - twiddle factor read; c - multiply  $x$  with  $tf$ ; d - butterfly; e - memory write back

## Example run: in-place computation (N=16)



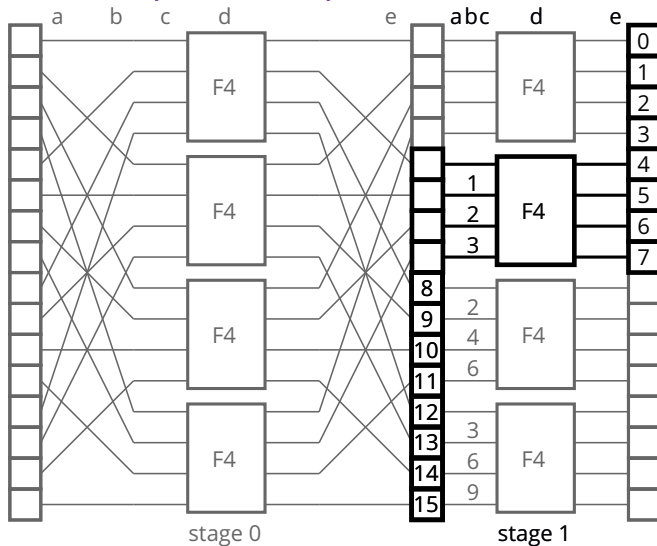
**Figure:** a - memory read  $x$ ; b - twiddle factor read; c - multiply  $x$  with  $tf$ ; d - butterfly; e - memory write back

## Example run: in-place computation (N=16)



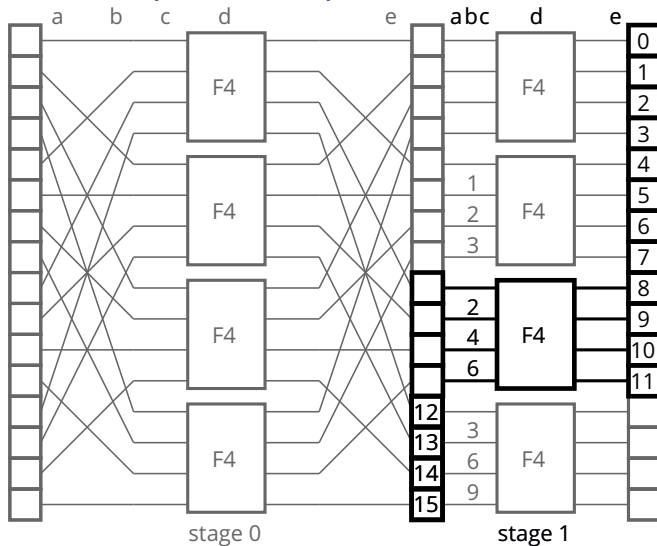
**Figure:** a - memory read  $x$ ; b - twiddle factor read; c - multiply  $x$  with  $tf$ ; d - butterfly; e - memory write back

## Example run: in-place computation (N=16)



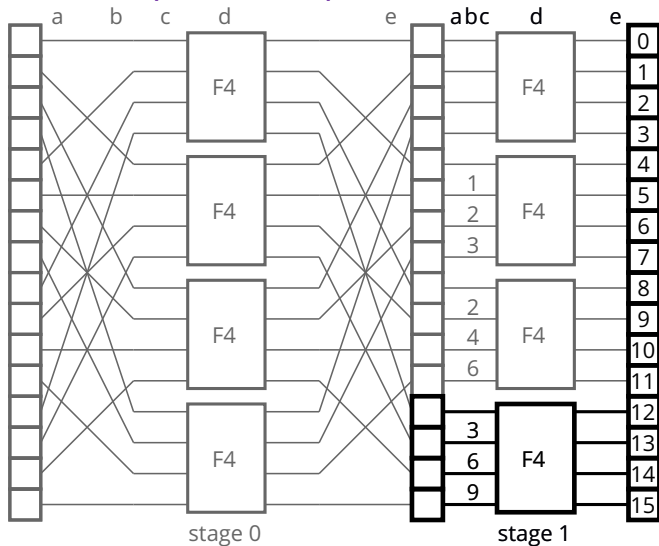
**Figure:** a - memory read  $x$ ; b - twiddle factor read; c - multiply  $x$  with  $tf$ ; d - butterfly; e - memory write back

## Example run: in-place computation (N=16)



**Figure:** a - memory read x; b - twiddle factor read; c - multiply x with twiddle factor; d - butterfly; e - memory write back

## Example run: in-place computation (N=16)



**Figure:** a - memory read x; b - twiddle factor read; c - multiply x with twiddle factor; d - butterfly; e - memory write back

# Example run: in-place computation (N=16)

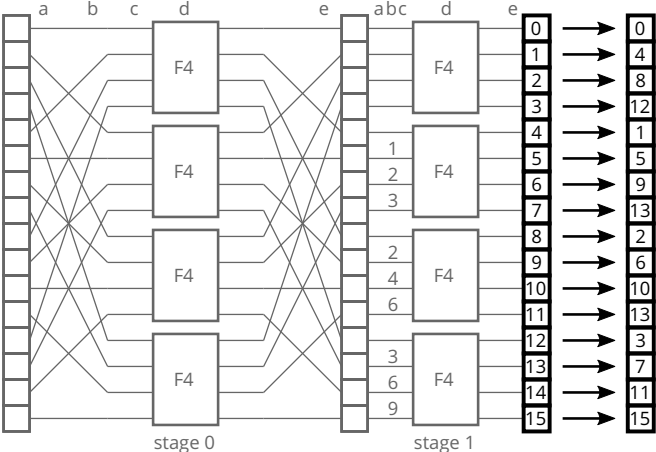


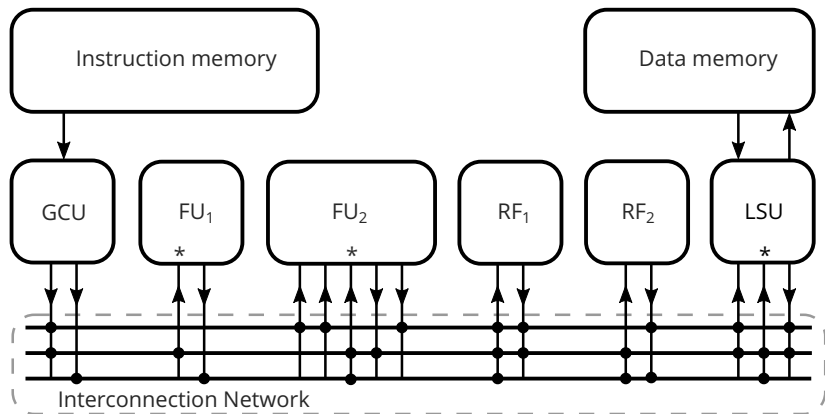
Figure: Reorder output (not part of the processor)



Low-Power **Programmable** Processor  
for Fast Fourier Transform  
Based on **Transport Triggered**  
**Architecture**

# Transport Triggered Architecture (TTA)

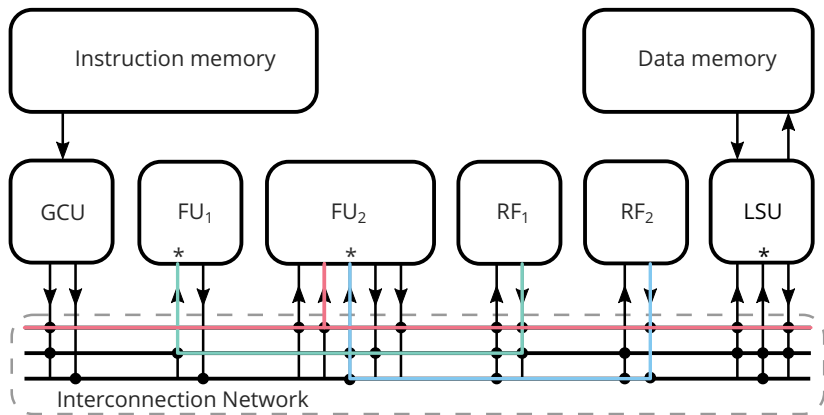
- ▶ Parallel architecture
- ▶ Only 1 instruction - *move*



**Figure:** Example TTA architecture; GCU - general control unit; FU - functional unit; RF - register file; LSU - load-store unit

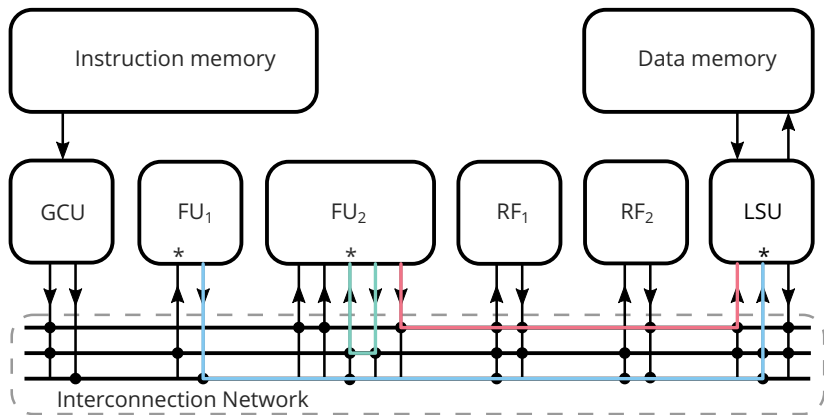
# Transport Triggered Architecture (TTA)

bus0	bus1	bus2
2 -> FU2.o2	RF1.0 -> FU1.t.neg	RF2.0 -> FU2.t.add
FU2.r1 -> LSU.o	FU2.r0 -> FU2.t.mul	FU1.r -> LSU.t.stw

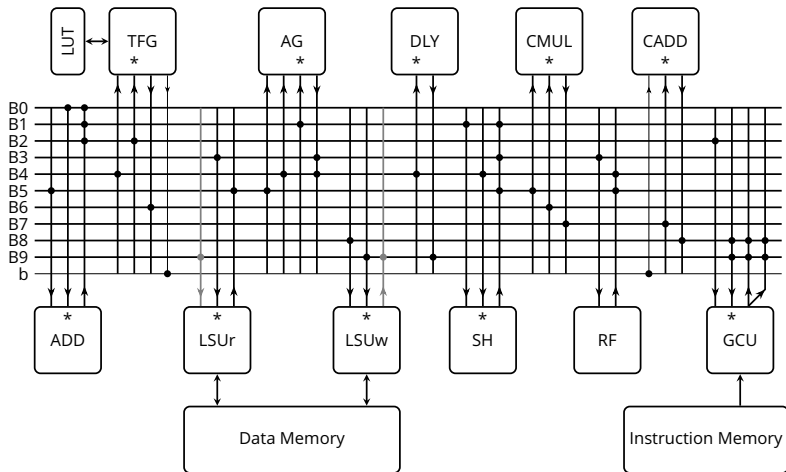


# Transport Triggered Architecture (TTA)

bus0	bus1	bus2
2 -> FU2.o2	RF1.0 -> FU1.t.neg	RF2.0 -> FU2.t.add
FU2.r1 -> LSU.o	FU2.r0 -> FU2.t.mul	FU1.r -> LSU.t.stw



# The Processor Architecture



**Figure:** ADD - adder; TFG - twiddle factor generator; LUT - lookup table; AG - address generator; DLY - rotating register (delay); SH - shifter; CMUL - complex multiplier; CADD - complex adder (butterfly)

# Instruction schedule

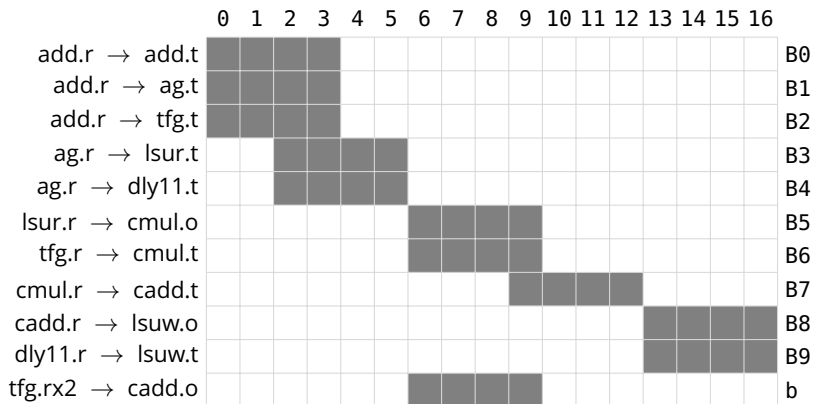


Figure: Bus reservation table of one radix-4 butterfly

# Instruction schedule

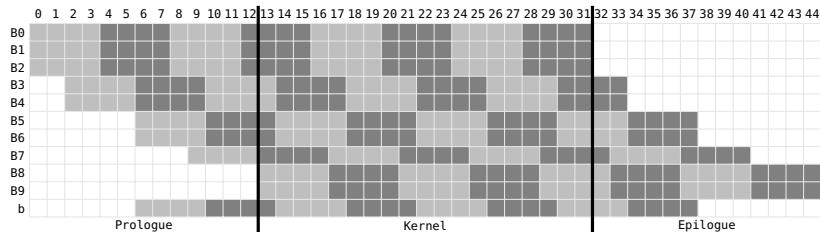


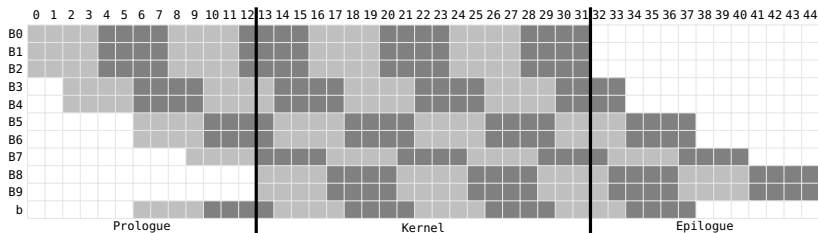
Figure: Bus reservation table of full 16-point FFT (not incl. setup code - 6 instr.)

**Low-Power** Programmable Processor  
for Fast Fourier Transform  
Based on Transport Triggered  
Architecture



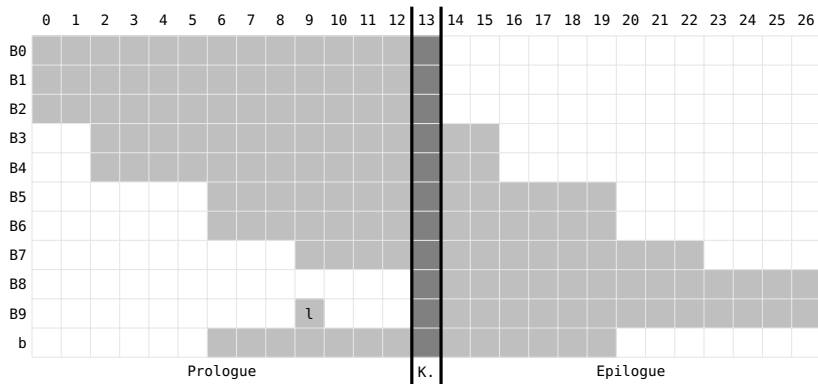
# Introducing Loop Buffer

- ▶ One repeated instruction word in kernel => unnecessary instruction memory fetches
- ▶ Memory fetching consumes power



# Introducing Loop Buffer

- ▶ Kernel is compressed into one instruction word
- ▶ The instruction word is saved in a loop buffer (small memory cache) and repeatedly executed from there

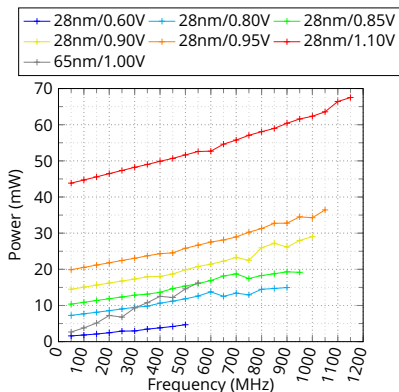


## Results

- ▶ Instruction memory power consumption is negligible (<1% of total)

#	contributor	%
1	data memory	43
2	complex multiplier	22
3	twiddle factor generator	16
4	complex adder	5.1
5	interconnect	4.1
6	rotating register	2.9
7	address generator	1.9
8	loop buffer	1.4
9	$\Sigma$ of <1% contributors	3.6

**Table:** Power consumption contributors (1024-point FFT, 28nm, 0.6V, 450MHz)



**Figure:** Synthesis results (power consumption)

# Results

## comparison of 1024-point FFT architectures

Power (mW)		FFT/mj		normalized FFT/mj	
28nm	4.2	[5]	77131	[5]	52865
[5]	8.9	28nm	20916	[7]	18498
65nm	12.2	[7]	13360	65nm	7171
[4]	17.6	65nm	7171	[9]	3609
[7]	29.0	[4]	2641	28nm	3243
[9]	60.4	[9]	802	[4]	3196

---

[4] M. Garrido, R. Andersson, F. Qureshi, and O. Gustafsson, "Multiplierless unity-gain SDF FFTs," 2016

[5] M. Garrido, S. Huang, and S. Chen, "Feedforward FFT hardware architectures based on rotator allocation," 2018

[7] S. Huang and S. Chen, "A high-parallelism memory-based FFT processor with high SQNR and novel addressing scheme," 2016

[9] T. Pitkänen, J. Takala, "Low-power application-specific processor for FFT computations," 2011

# Credits

# TTA-based Co-design Environment (TCE)

- ▶ GUI+CLI toolset<sup>2</sup> for designing TTA processors
  - ▶ Architecture designer (GUI)
  - ▶ HDL generator (generates the full processor RTL)
  - ▶ Batteries included (tutorial, HDL for basic FUs)
  - ▶ C & assembly compilers
  - ▶ Simulator (visualizing data moves)
  - ▶ and more...
- ▶ Started as a MOVE project by Henk Corporaal<sup>3</sup> (1990s)
- ▶ Continued by a CPC group ([openasip.org](http://openasip.org)) led by Jarmo Takala<sup>4</sup> and now Pekka Jääskeläinen<sup>4</sup> (early 2000s - now)

---

<sup>2</sup>P. Jääskeläinen, T. Viitanen, J. Takala, and H. Berg, "HW/SW co-design toolset for customization of exposed datapath processors," in *Computing Platforms for Software-Defined Radio*, pp. 147–164. Springer, 2017

<sup>3</sup>Eindhoven University of Technology

<sup>4</sup>Tampere University

**Thank you!**