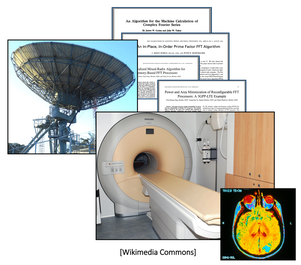
A **Generator** Of Memory-Based, **Runtime-Reconfigurable** $2^n3^m5^k(7^l\ldots)$ **FFT** Engines

A Case Study in Agile Hardware DSP Development.

Angie Wang, Jonathan Bachrach, Borivoje Nikolić, UC Berkeley

FFTs Are Everywhere!

- Hardware FFTs *(re)*implemented countless times...



[Wikimedia Commons]

Core DSP in

- LTE,
- Wi-Fi,
- MRI,
- Radio Astronomy,
- Spectrum Analyzers,
- Image Compression,
- Object Recognition,
- *... You name it!*

Hardware is Domain-Specific.

Case Study: Runtime-reconfigurable SDR for LTE + Wi-Fi

Wi-Fi 802.11ac

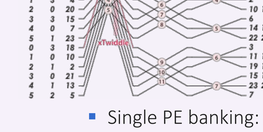
- Symbol duration 3.2µs + 800 ns GI

BW (MHz)	20	40	80	160
FFT Size	64	128	256	512

LTE

- Symbol duration 66.67µs + 4.76µs CP

BW (MHz)	1.25	2.5	5	10	15	20
FFT Size	128	256	512	1024	1536	2048
					Non-2^n	



FFT Sizes for SC-FDMA Precoding						
12	96	216	384	648	972	
24	108	240	432	720	1080	
36	120	288	480	768	1152	
48	144	300	540	864	1200	
60	180	324	576	900	1296	
72	192	360	600	960		

A *Universal* Hardware Solution?

Reconfigurable IP Cores

- ☺ Consistent, well-documented performance, *when supported*
- ☹ Cannot be easily modified to support new standards
- ☹ Not resource-optimized for specific applications
- ☹ Tied to platforms/closed $$$ IP

FFT Generators

- ☺ Architecture regularity to cover wide range of applications
  - Great for design space exploration!
- ☺ Minimum design overhead
- ☹ Existing generators have incomplete feature sets
  - Not good enough for SDR

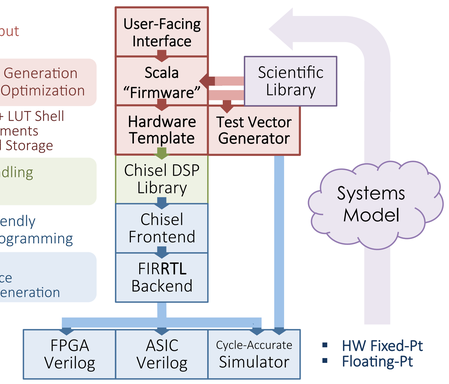
Chisel for DSP

Towards a Unified Development Environment

- Parameter Input

Fills in

- DSP Constant Generation
- Performance Optimization
- Control Logic + LUT Shell
- Processing Elements
- Parameterized Storage
- Overflow Handling
- Rounding
- Generator Friendly
- Functional Programming
- Auto Pipeline
- Range Inference
- Node Graph Generation



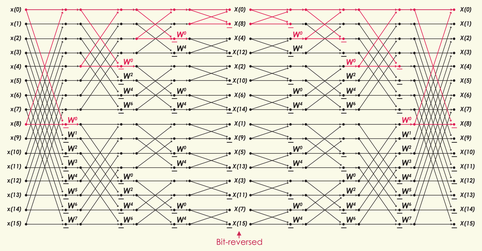
- HW Fixed-Pt
- Floating-Pt

Cooley-Tukey Algorithm (CTA)

- Most commonly used, typically for 2^n decompositions

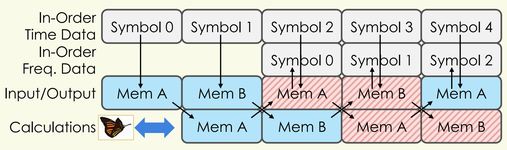
$$X[k_1,k_2]=\sum_{n2}\underbrace{W_N^{n_2k_1}}_{\text{Twiddle}}\left\{\sum_{n1}x[n_1,n_2]W_{N_1}^{n_1k_1}\right\}W_{N_2}^{n_2k_2}$$

- *Duality:* Decimation-in-Frequency (DIF) ↔ Decimation-in-Time (DIT)



DIT↔DIF Duality for Memory Reduction

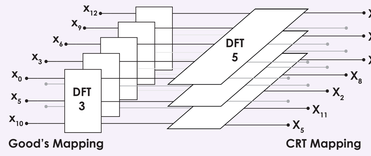
- Alternating between DIT/DIF every 2a[th] symbol allows for in-place IO and 2N memory
  - ≥ 50% memory savings over Spiral



Prime Factor Algorithm (PFA)

- Eliminates twiddles for coprime decompositions

$$X[k_1,k_2]=\sum_{n2}\left\{\sum_{n1}x[n_1,n_2]W_{N_1}^{n_1k_1}\right\}W_{N_2}^{n_2k_2}$$



- PFA → CTA decomposition
  - ☺ Minimizes twiddle ROM size (LTE/Wi-Fi 1718 twiddles)
    - Twiddle addresses renormalized to support different FFT sizes
  - ☺ Partitions ROMs by coprimes (great for reconfigurability!)

24-pt *(PFA+CTA)* FFT SFG

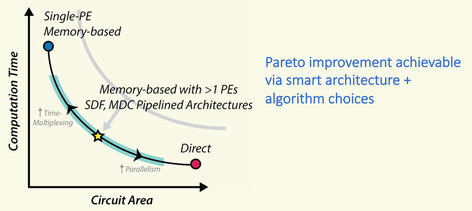
- Forward/reverse decompositions are alternated (custom index-mapping scheme) $(k_3,k_2,k_1)_{2a}\rightarrow(n_1,n_2,n_3)_{2a+2}$
  - Index vectors generated with Base-r arithmetic

- Single PE banking: $b_0=\left(\sum_{i=0}^{i<r}n_i\right)\bmod r_{max}$

$$b_{0<i<r}=(b_0+j)\bmod r_{max}$$

Why Memory-Based?

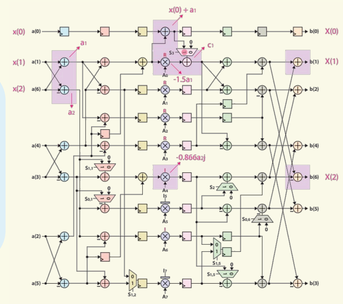
- Easy to adapt to *reconfigurable* non 2^n architectures
- Given throughput constraint *(i.e. continuous dataflow)*, optimal # of parallel processing elements (PEs) can be calculated, used
  - Complicates conflict-free scheduling (ongoing research)



Pareto improvement achievable via smart architecture + algorithm choices

Reconfigurable Radix-2,3,4,5,7 WFTA Butterfly

- WFTA uses fewer multipliers, but more adders
- Reuses adders/multipliers in radix-7 BF to reduce hardware
  - 2 radix-2 BF ops in parallel
- Optimal stage bypassing for hardware optimization with unused radices
- Custom stage pipelining

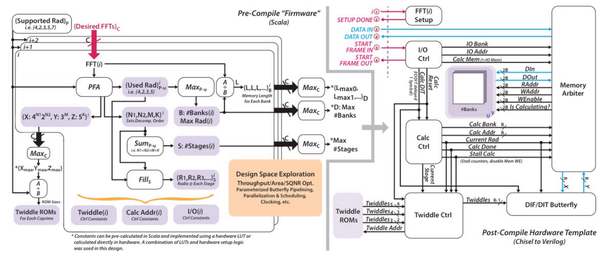


Radix-3 Configuration

FFT Generator

Scala "Firmware" ➡ Hardware Template

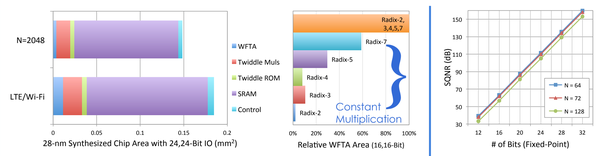
- Calculates constants for LUT generation
- Calculates hardware parameters
- Configurable blocks controlling dataflow between I/O, memories, & butterfly
- Design complexity primarily in control logic



Resource & Performance Comparison

	Fixed FFT N = 2048			LTE/Wi-Fi SDR
	Chisel FFT	Spiral Streaming	Spiral Iterative	**Chisel FFT**
Data Memory	2N	7.3N	4N	4,576 → 2.23N$_{MAX}$
Twiddle ROM	0.75N	0.99N	N	1,718 → 0.84N$_{MAX}$
Computation Cycles	~3,100	512	11,287	≤ 2N
Radix	1 Radix-4/2×2	Radix-4	Radix-2	1 Radix-5,4,3,2×2
# of Real Multipliers	12	56	4	26

- Compared to Spiral's radix-2 (streaming width = 2) iterative FFT:
  - 50% data memory + 25% twiddle ROM savings, higher throughput
- LTE/Wi-Fi FFT uses 10 SRAM banks: 2 × (4×512 + 240)
- Calculation running at 2× the IO rate for continuous dataflow
  - 2 Radix-2 BFs scheduled in parallel (rearranged to avoid bank conflict)



- Chip area dominated by SRAM (3N+2N reduction is great!)
- 25% area penalty to support *all* Wi-Fi/LTE DFT + IDFT sizes (compared to N = 2048)

- SQNR from comparison with floating-pt ScalaNLP/Breeze FFT
- Future exploration of data path optimizations (i.e. block floating point)

Acknowledgements

- DARPA CRAFT (HR0011-15-1-0010), NSF-GRFP (DGE-1106400)
- ASPIRE & BWRC faculty, staff, and students (especially S. Bailey, P. Rigge, S. Twigg, C. Yarp)

References

- C.-F. Hsiao, et al., "A Generalized Mixed-Radix Algorithm for Memory-Based FFT Processors"
- J. Chen, et al., "Hardware Efficient Mixed Radix-25/16/9 FFT for LTE Systems"
- C. S. Burrus and P.W. Eschenbacher, "An In-Place, In-Order Prime Factor FFT Algorithm"
- F. Qureshi, et al., "Unified Architecture for 2, 3, 4, 5, and 7-point DFTs based on Winograd Fourier Transform Algorithm"