Discrete Fourier Transform IP Generator

Grace Nordin, James C. Hoe, and Markus Püschel
Dept. of Electrical and Computer Engineering
Carnegie Mellon University

The Paradox of Reusable IPs

- Boon to productivity
  - zero effort required
  - zero knowledge required
  - zero chance to introduce new bugs

  Why repeat what is already been done?
- Bane to optimality
  - finding the right functionality with the right interface
  - design tradeoff -- performance, area, power, accuracy.....

  Are you getting what you really wanted?

- Solution: parameterized automatic IP generators
  - zero effort, knowledge or bugs
  - allows application specific customization
  - facilitates design exploration

Discrete Fourier Transform IPs

- Discrete Fourier Transform (DFT)
  - important building block in DSP applications
  - numerous design “cores” available
- Some commonly supported options in IP libraries
  - transform sizes
  - number format
  - i/o data ordering
  - a small number of microarchitecture choices (e.g., min area, max speed)
- Customized design tradeoff in our generated IPs
  - degree of parallelism in microarchitecture *(min ↔ max)*
  - resource preference *(e.g. BRAM vs. LUT in FPGAs)*

Extensible to other common linear DSP transforms

Outline

- Introduction
- Formula-Driven Design Generation
- Microarchitecture Parameterization
- Resource Parameterization
- Experimental Results
- Conclusions

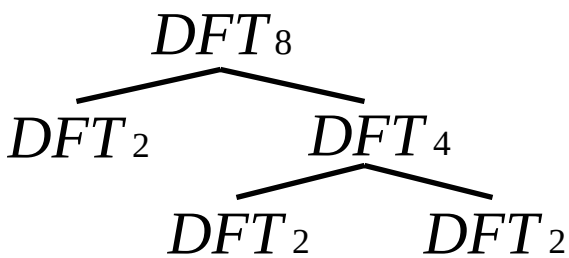
Transforms as Formulas [www.spiral.net]

Transform DFT_n parameterized matrix

Recursion $$DFT_{nm} \rightarrow \left(DFT_n \otimes I_m\right) \cdot D \cdot \left(I_n \otimes DFT_m\right) \cdot P$$

- a breakdown strategy
- product of sparse matrices

Algorithm As Tree



- recursive application of rules
- uniquely defines an algorithm
- efficient representation
- easy manipulation

Algorithm As Formula

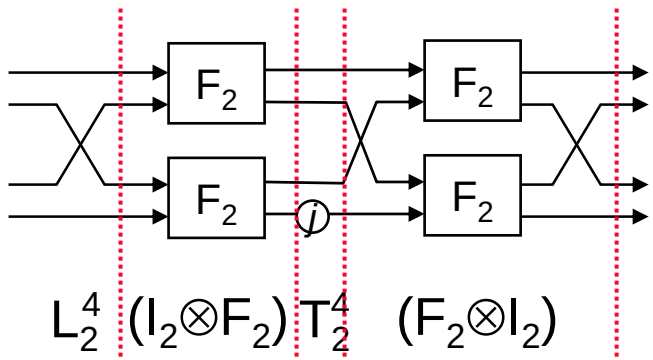
$$DFT_8 = \left(F_2 \otimes I_4\right) \cdot D \cdot \left(I_2 \otimes \left(\left(F_2 \otimes I_2\right) \cdots\right)\right) \cdot P$$

- few constructs and primitives
- uniquely defines an algorithm
- can be translated into code

Formula to Datapath

- Given $M \cdot x$ where M is
  - $M = A \cdot B$ — apply B, then A
  - $M = I_n \otimes A$ — apply A, n times in parallel
  - $M = A \otimes I_n$ — apply A, n times in parallel taking inputs at stride n
  - M is a permutation — permute x
  - M is a diagonal — scale x
  - etc.

- formulas are a natural HW description
- formulas allow manipulation
- formulas can be translated into Verilog



Outline

- Introduction
- Formula-Driven Design Generation
- Microarchitecture Parameterization
- Resource Parameterization
- Experimental Results
- Conclusions

Pease DFT

- Simple regular structure embodied in formula

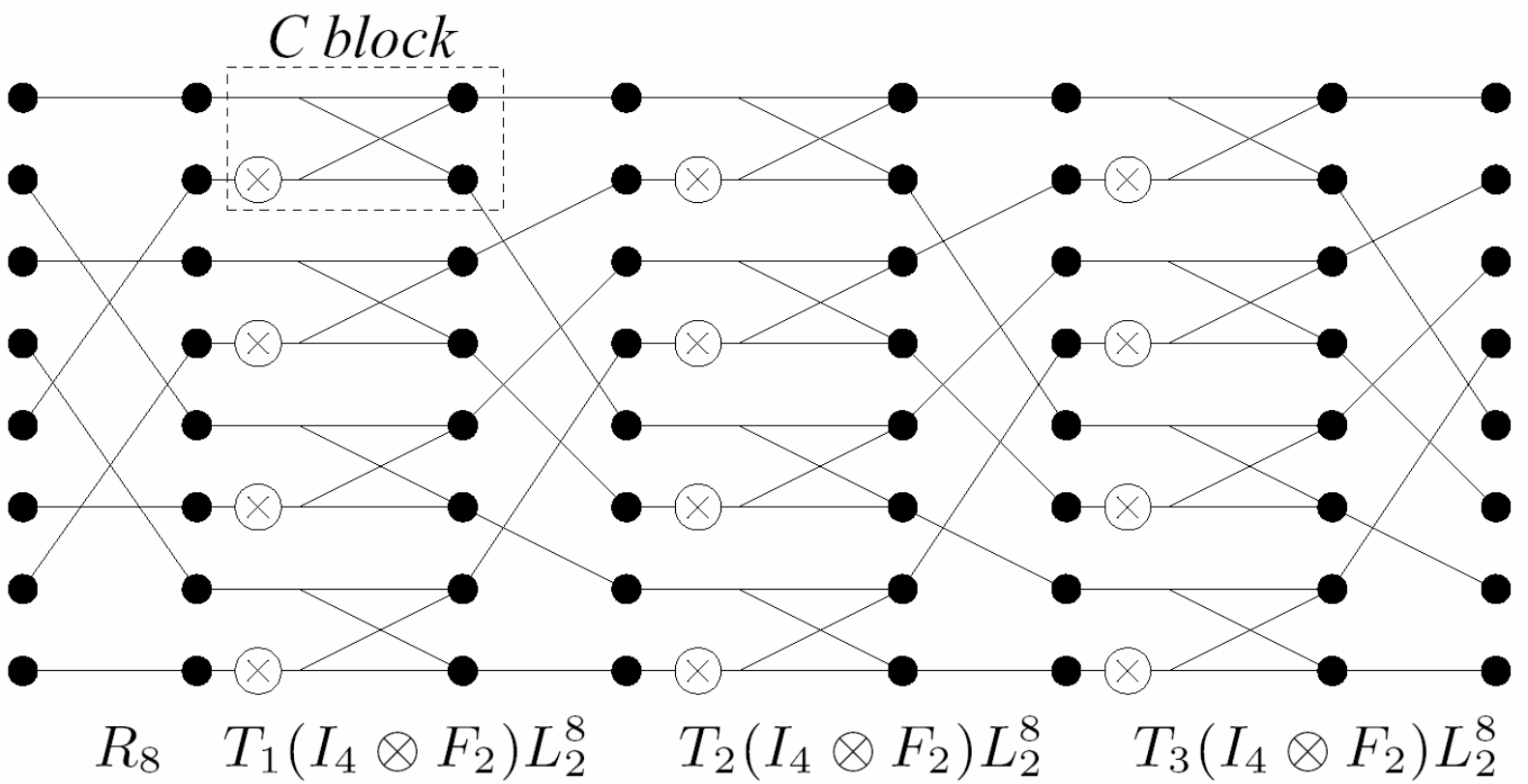
$$\mathrm{DFT}_{2^k} = \left(\prod_{i=0}^{k-1} L_2^{2^k} (I_{2^{k-1}} \otimes F_2) T_{k-i} \right) R_{2^k}$$

where $T_{k-i} = L_{2^{k-i-1}}^{2^k} (I_{2^i} \otimes D_{2^{k-i-1}}^{2^{k-i}}) L_{2^{i+1}}^{2^k}$

- Example

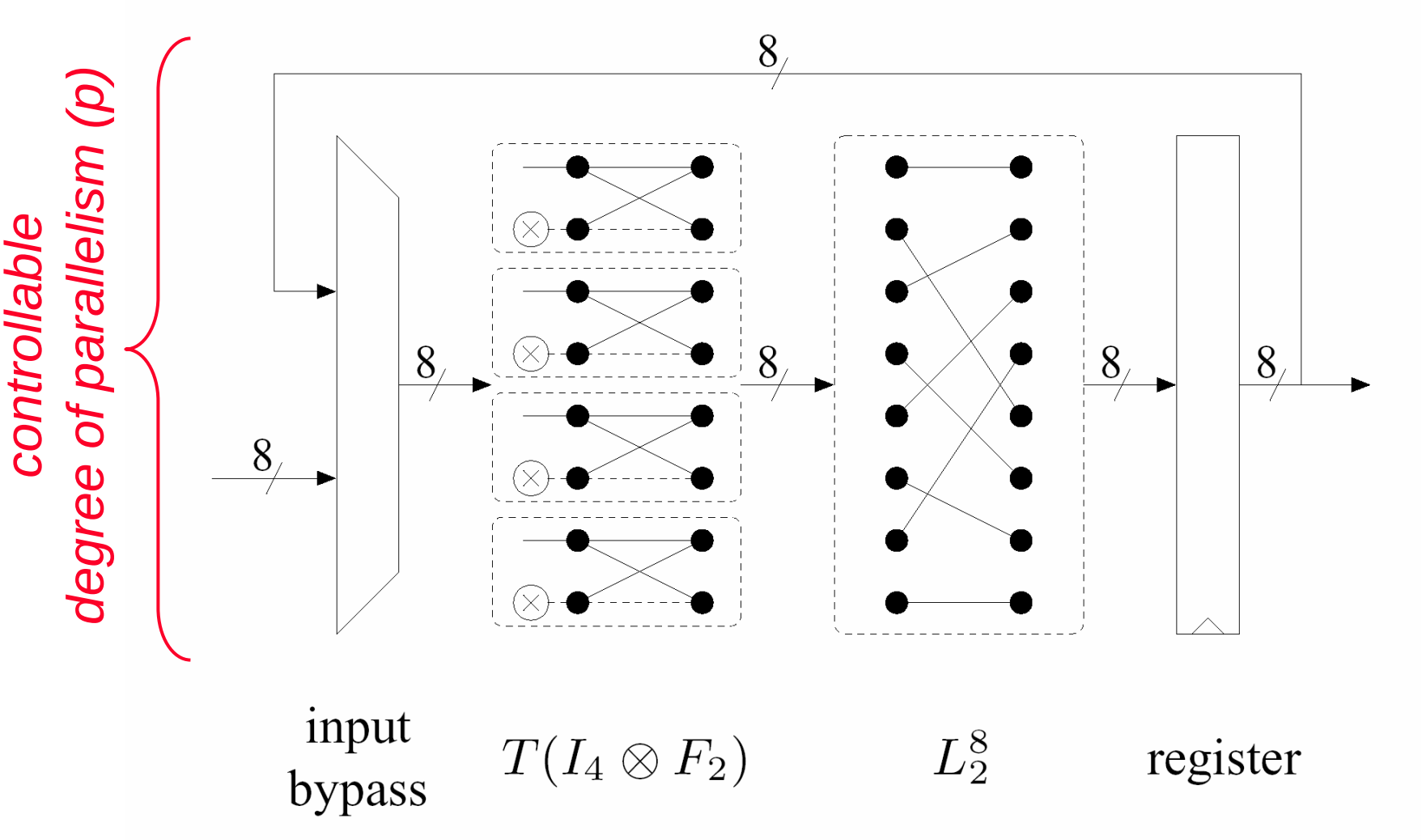
$$\mathrm{DFT}_8 = \left(L_2^8 (I_4 \otimes F_2) T_3\right) \left(L_2^8 (I_4 \otimes F_2) T_2\right) \left(L_2^8 (I_4 \otimes F_2) T_1\right) R_8$$

Pease DFT Example: DFT_8

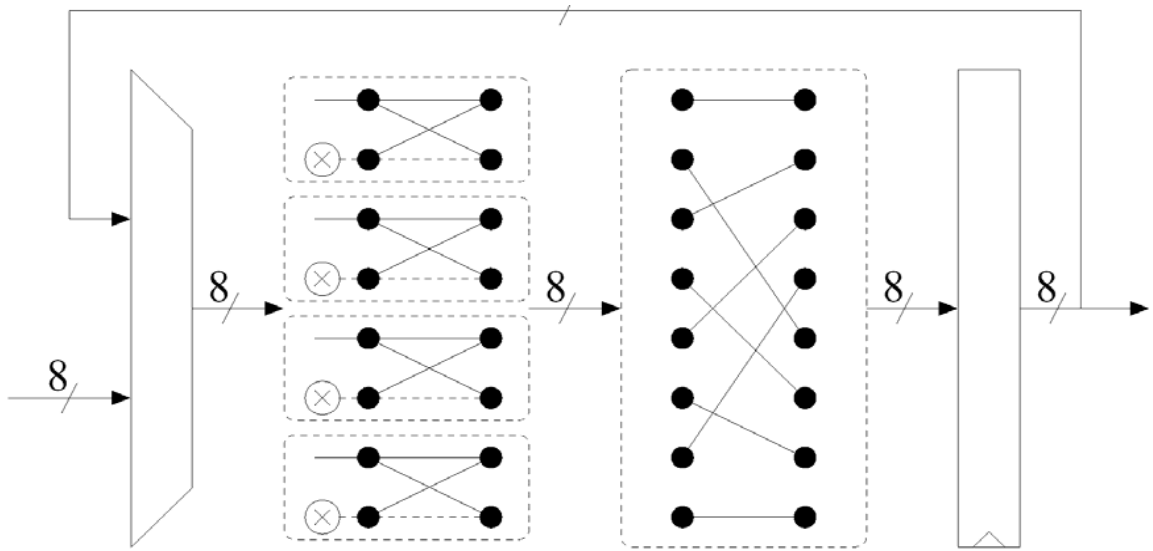


Repeating column structure ⇒ hardware reuse with zero perf. penalty

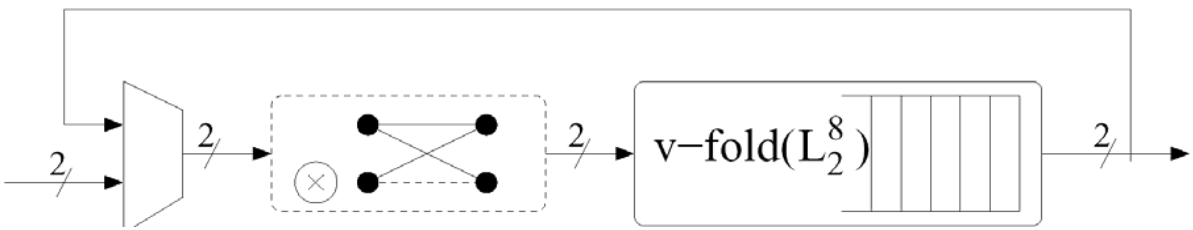
Horizontally Folded Pease DFT



V-folding according to p



p_{max} =n/2=4



p_{min} =1

Latency = $t\left(\frac{n}{2p}(\log(n)-1)\right)$

Outline

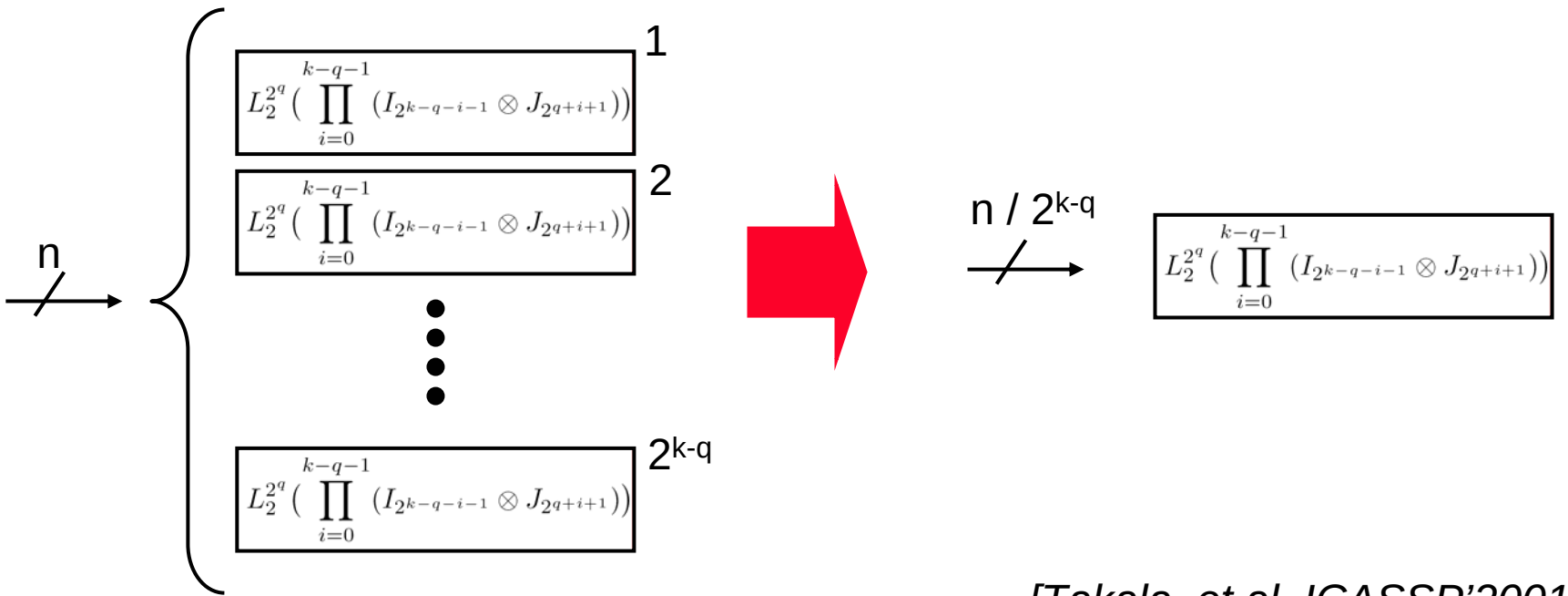
- Introduction
- Formula-Driven Design Generation
- Microarchitecture Parameterization
- Resource Parameterization
- Experimental Results
- Conclusions

V-folding of Stride Permutations

- Stride Permutation

$$L_2^n = I_{2^{k-q}} \otimes L_2^{2^q} \Big(\prod_{i=0}^{k-q-1} (I_{2^{k-q-i-1}} \otimes J_{2^{q+i+1}}) \Big)$$

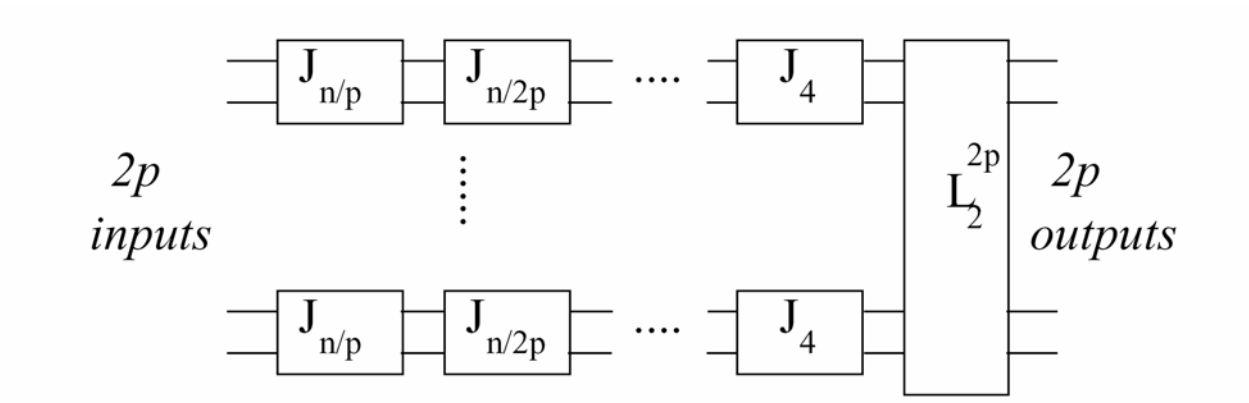
- In other words



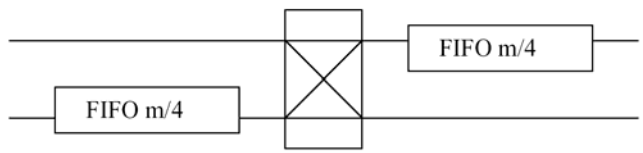
[Takala, et al. ICASSP'2001]

V-folding of Stride Permutations

$$L_2^{2^q}\left(\prod_{i=0}^{k-q-1}\left(I_{2^{k-q-i-1}} \otimes J_{2^{q+i+1}}\right)\right)$$



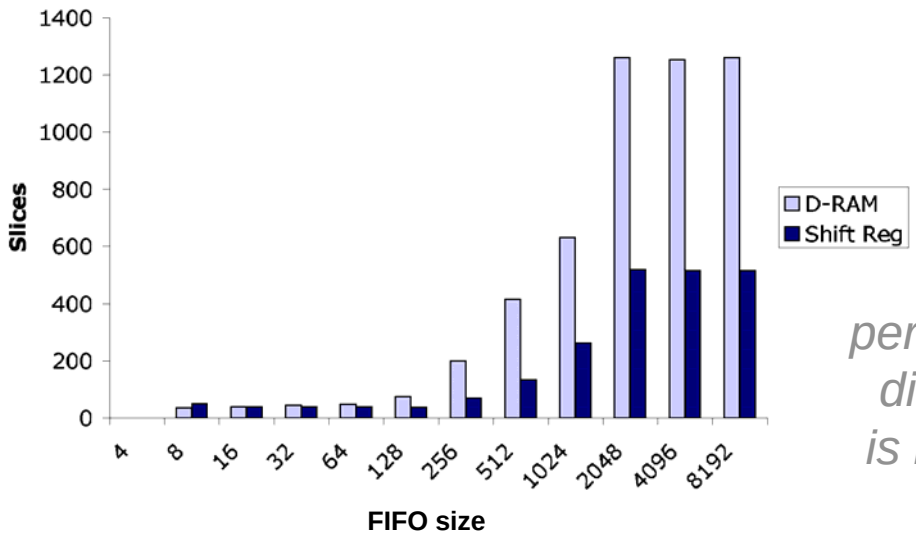
where J_m =



[Takala, et al. ICASSP'2001]

FIFO: BRAM vs. CLBs

- J-matrix FIFOs are a significant part of logic resources
- FIFOs can be constructed from
  - shift registers using CLB slices, or
  - circular buffers using CLB slices (distributed RAM), or
  - circular buffers using BRAM memory macros
- "Exchange rate" of shift registers vs. circular buffer



performance difference is negligible

Let user set the context-dependent break-even point

Outline

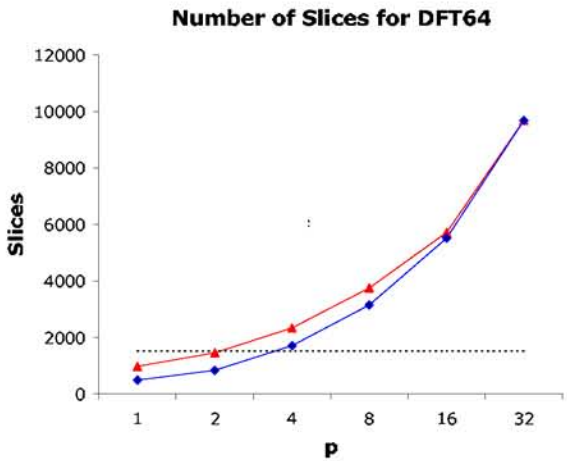
- Introduction
- Formula-Driven Design Generation
- Microarchitecture Parameterization
- Resource Parameterization
- Experimental Results
- Conclusions

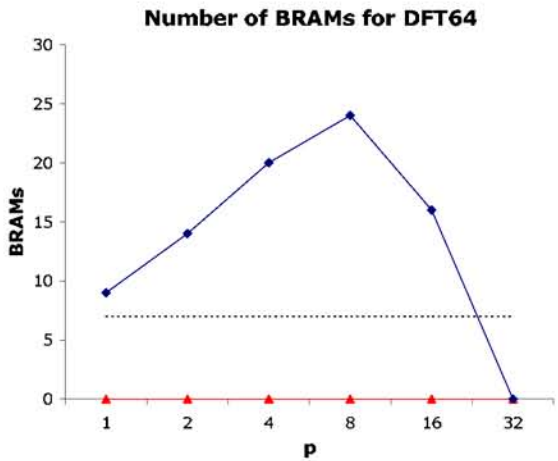
Xilinx LogiCore Library

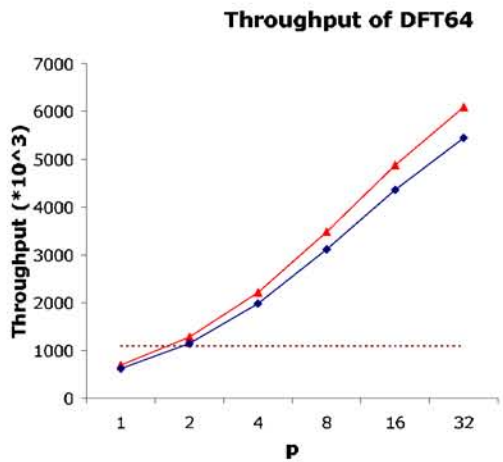
- DFT based on Radix-4 Cooley-Tukey
  - range of sizes
  - streaming vs. burst I/O
  - fixed-point scaling modes
  - in/out data ordering

- Evaluation
  - DFT of 64, 1024 and 2048
  - burst I/O interface, bit-reversed-ordering
  - Xilinx Virtex2-Pro XC2VP100-6
  - Xilinx ISE version 6.1.03i

DFT_{64}

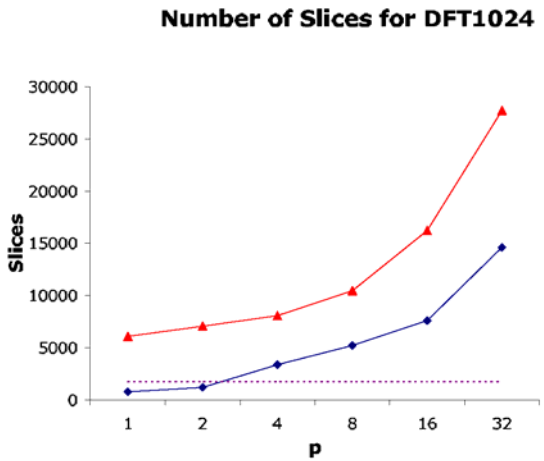

Number of Slices for DFT64
Slices
12000
10000
8000
6000
4000
2000
0
1
2
4
8
16
32
p

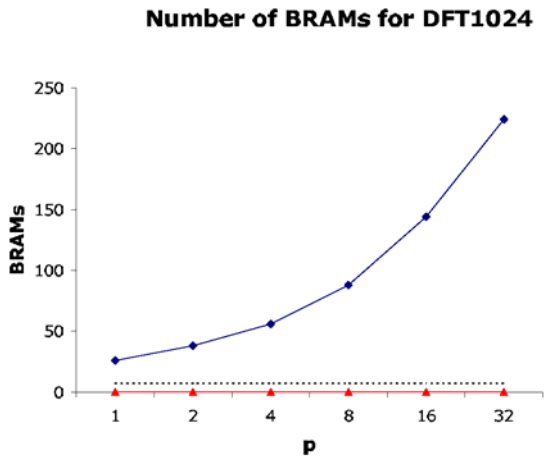

Number of BRAMs for DFT64
BRAMs
30
25
20
15
10
5
0
1
2
4
8
16
32
p

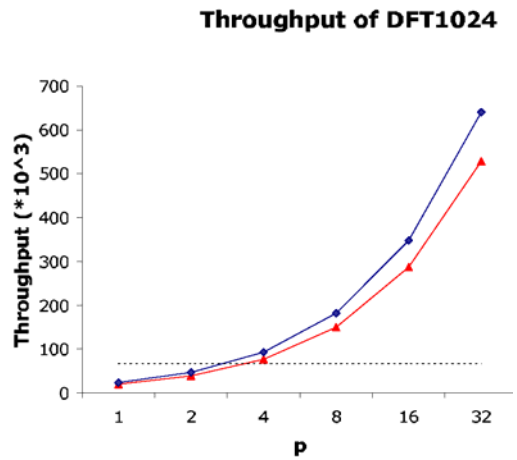

Throughput of DFT64
Throughput (*10^3)
7000
6000
5000
4000
3000
2000
1000
0
1
2
4
8
16
32
p

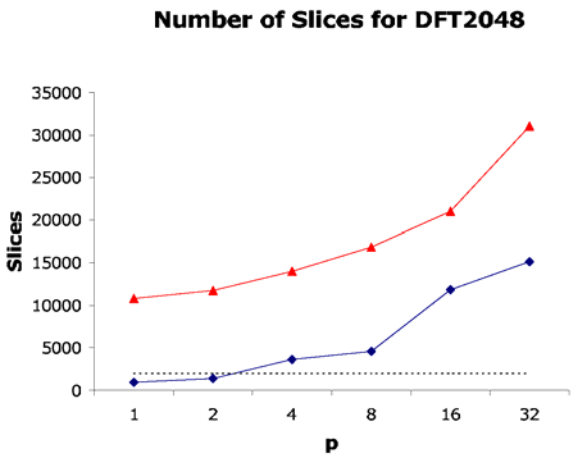

Min BRAM
Min Slice
Xilinx

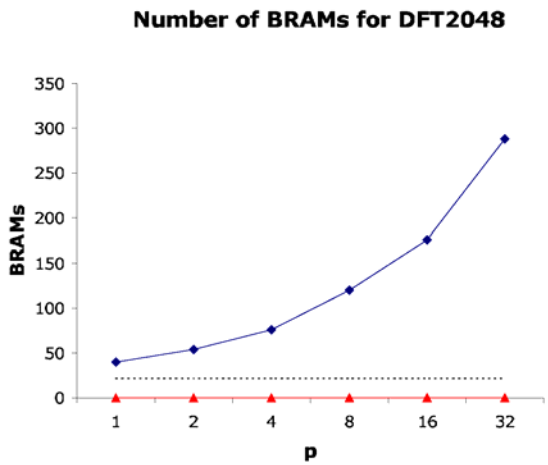
DFT_{1024} and DFT_{2048}

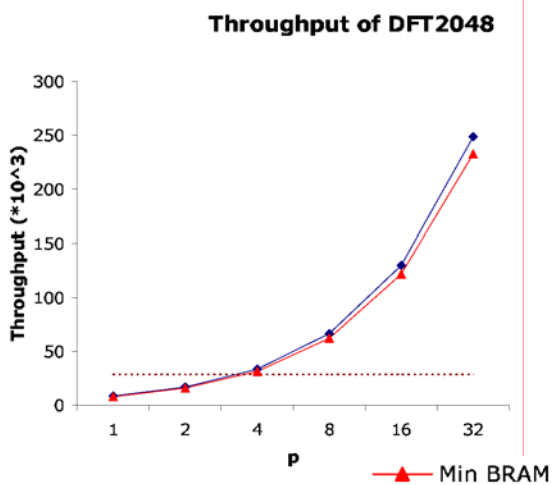












Outline

- Introduction
- Formula-Driven Design Generation
- Microarchitecture Parameterization
- Resource Parameterization
- Experimental Results
- Conclusions

Related Work

- Kumhom, Johnson, Nagvajara, ASIC/SOC 2000
  - universal FFT processor microarchitecture based on processing elements interconnected by on-chip reconfigurable network
  - microarchitecture is scalable in the number of elements
  - supports both Cooley Tukey and Pease

- Choi, Scrofano, Prasanna, Jang, FPGA'2003
  - mapped radix-4 Cooley-Tukey algorithm onto log(n)/2 DFT_4 primitives
  - scalable datapath between 1 element and 4 elements at a time
  - show energy and performance improvements from scaling
  - does not show same tradeoff point as Xilinx can be covered

Conclusions

- Parameterized IP Generator
  - easy to use
  - allows customization
- Prototype implementation of DFT generator
  - parameterized performance/cost tradeoff
  - parameterized resource usage preference
- Key results
  - generator is efficient, i.e., the Xilinx design point can be matched
  - customization allows advantage in a chosen dimension relative to Xilinx DFT cores