Efficient area-throughput tradeoff for FFT

Harish Nagisetty
Devendra Ponnapureddy
Carrol Speir

December 2nd, 2022

Inventors

- Harish Nagisetty
- Devendra Ponnapureddy
- Carrol Speir

Introduction

Our design allows us to exchange throughput for area without increases in power consumption or latency.

Problem Definition

To achieve the highest possible throughput, we must implement a fully parallel, pipelined FFT in hardware. For some applications, a slower design may be acceptable in exchange for reduced logic area.

A fully parallel design is very power-efficient. We would like to retain this benefit when trading throughput for area.

Prior Art

Most approaches involve multiplexing the input data and twiddle factors.

Twiddle factors may need to be stored in memory and regular multipliers are used instead of constant multipliers.

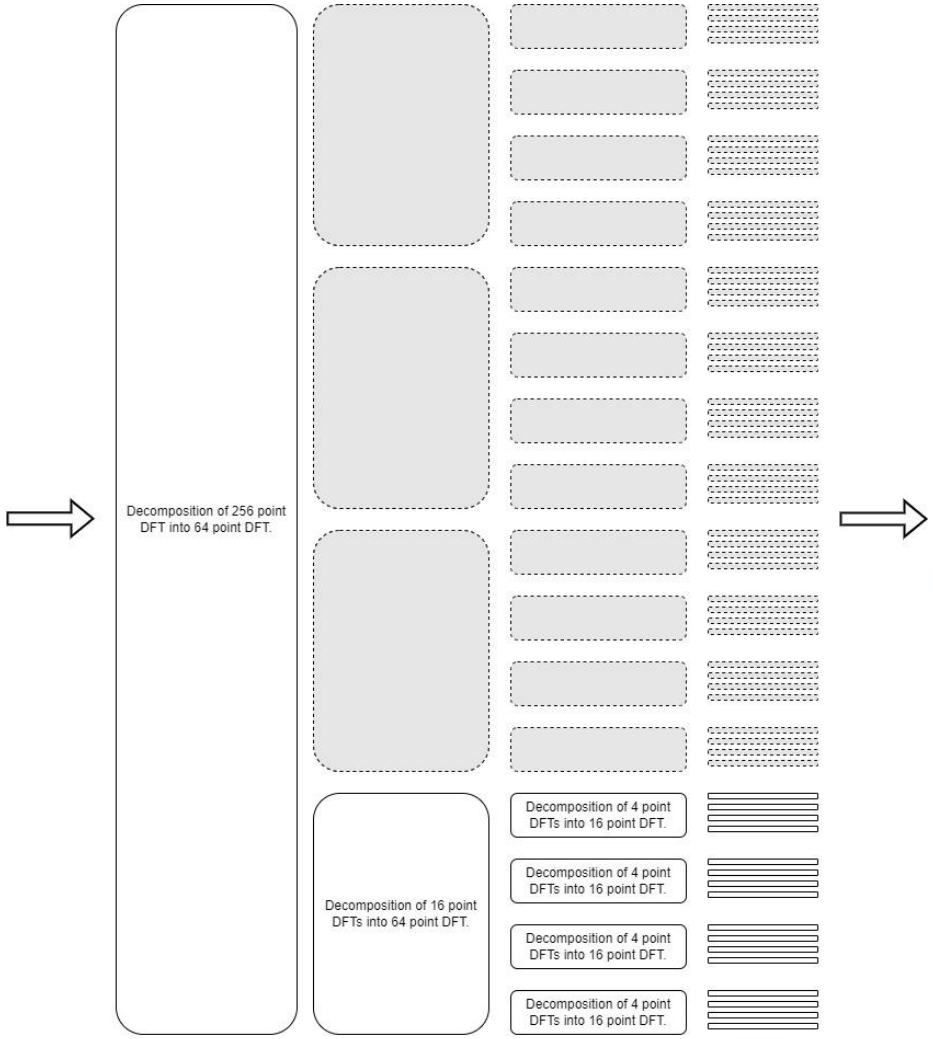
The use of regular multipliers and multiplexers can result in a significant increase in power consumption. When one operand of a multiplication is constant, the operation can be implemented very efficiently.

Proposed Solution

Examining the DIF-FFT algorithm, we see that all stages except the first contain repeated and independent sub-problems.

We need to implement the first stage fully, only one sub-problem from the second stage, and the corresponding number of sub-problems from further stages.

Proposed Solution (continued)



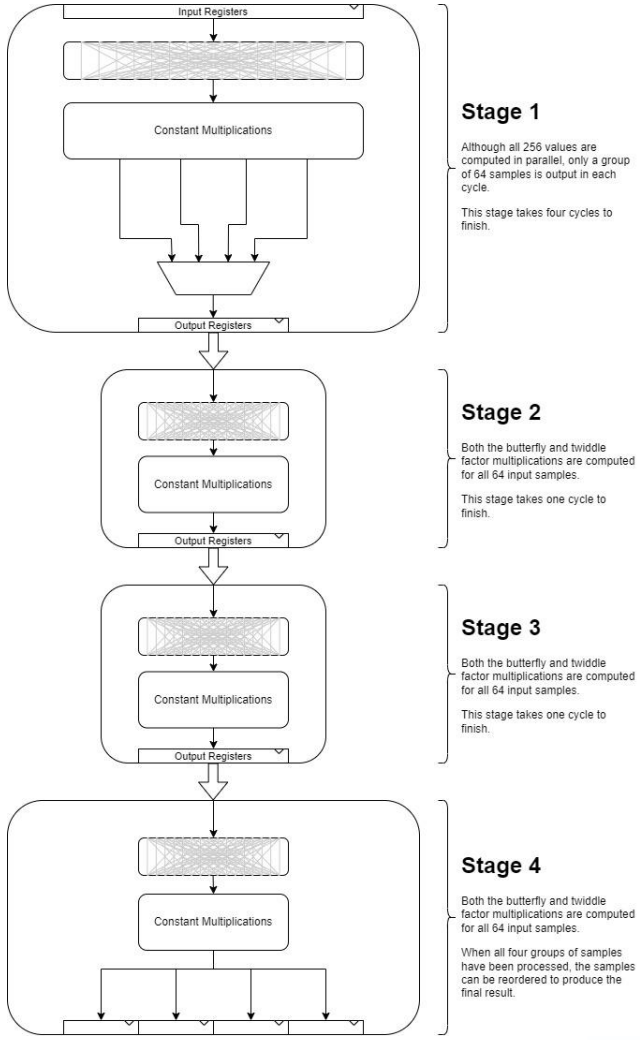
Proposed Solution (continued)

Consider a radix-4, 256 point FFT as an example.

The first stage computes a 256-sample result in the first cycle. It provides the first 64 samples to the pipeline formed by the 2^{nd}, 3^{rd}, and 4^{th} stages. The remaining samples are sent to the pipeline in groups of 64 over the next three cycles. When all four groups are finished, the data can be reordered to obtain the result.

A fully parallel design can accept new input every cycle, but this design can only accept new input every four cycles.

Proposed Solution (continued)



Proposed Solution (continued)

This approach reduces the area to ¼ (for a very large radix-4 FFT) while reducing throughput by the same factor.

Minimal multiplexing is required only for the first stage, and all the multiplications involve one constant operand. We achieve a nice tradeoff between area and throughput while retaining all other advantages of a fully parallel design.

What do you think needs to be claimed

An implementation of the FFT algorithm, where only a portion of the further stages is implemented in hardware. The hardware for the further stages is reused to compute the full transform.

These further stages involve constant multipliers (where one operand is constant), and do not contain multiplexing, making this design power-efficient.