# VLSI Implementation of a Pipelined 128 points 4-Parallel radix-$2^3$ FFT Architecture via Folding Transformation

James J. W. Kunst jjwk89@gmail.com
Kevin H. Viglianco kevinviglianco@gmail.com
Daniel R. Garcia dani6rg@gmail.com

June 15, 2020

Fundación
FULGOR

# Tabla de Contenidos

Sección 1

# Introduction

# Introduction

Introduction:

## Objetives

- Design and implement a 4-parallel pipelined architecture for the Complex Fast Fourier Transform (CFFT) based on the radix-$2^3$ algorithm with 128 points using folding transformation and register minimization techniques based on ... .
- Frecuency of implementation: 500MHz.
- Optimization with CSD[a] multipliers.
- Test the design with a mixture of two sinusoids using MATLAB.
- Generate power-area-timing report with different optimizations.

[a] Canonic Signed Digit

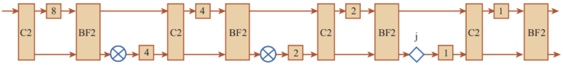
### Workflow

1. Obtain the equations that correspond to Butterfly structure of radix-$2^3$ FFT for 8 points.
2. Apply this idea to design a 2-parallel pipelined architecture radix-$2^3$ 16-points FFT via folding transformation.
3. Traslate this to a 128-points model.
4. Elaborate a float-point simulator in Matlab of the 128-points.
5. Elaborate a synthesizable verilog code HDL and verify the DFT functionality.
6. Generates power-area-timing report with different optimizations.

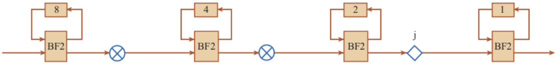
## Types of pipelined Radix-2 FFT architectures

1. Feedfoward Multi-Path Delay Commutator (MDC).
2. Single-Path Delay Feedback architectures (SDF).

- Both butterflies and multipliers are in 50% utilization ... .
- The SDF use registers more efficiently.
- We will focus on the feedfoward MDC asrchitecture.




(1) . R2MDC(N-16)




(2) . R25DF(N-16)

Sección 2

# The Radix-$2^3$ FFT Algorithm

# The Radix-$2^3$ FFT Algorithm

The Radix-$2^3$ FFT Algorithm:

**N-point DFT of an input sequence $x[n]$**

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot W_N^{nk}, \quad k = 0, 1, ..., N-1 \tag{1}$$

where $W_N^{nk} = e^{-j\frac{2\pi}{N}nk}$.

- Direct computation of the DFT is inefficient because it does not exploit the properties of:
    1. Symmetry: $W_N^{k+N/2} = -W_N^k$
    2. Periodicity: $W_N^{k+N} = W_N^k$
- The FFT based on Cooley-Tukey algorithm reduce the number of operations from $O(N^2)$ for the DFT to $O(Nlog_2N)$ for the FFT.

# The Radix-$2^3$ FFT Algorithm



## Divide and Conquer approach

- We can calculate the DFT in series of $s = log_{\rho} N$ stages, where $\rho$ is the base of the *radix*.
- This is based on the decomposition of an N point DFT into successively smaller DFTs.
- In our case, the numeber of stages is:

$$s = log_2(128) = 7 \tag{2}$$

## Methods to design FFT algorithms

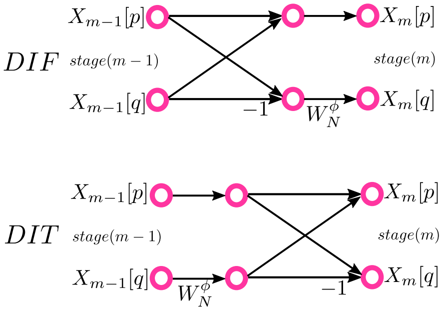
1. **Decimation In Time (DIT):** Splitting successively data sequence $x[n]$ by a factor of 2.
2. **Decimation In Frcuency (DIF):** Splitting successively the data sequence $X[k]$ by a factor of 2.

# The Radix-$2^3$ FFT Algorithm

The Radix-$2^3$ FFT Algorithm:

## DIT and DIF Butterflies

The difference is the instant in which the multiplication by $W_N^{\phi}$ is accomplished.

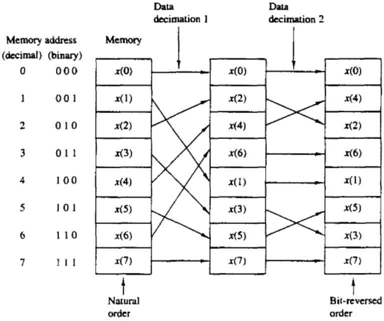


# The Radix-$2^3$ FFT Algorithm

## The Radix-$2^3$ FFT Algorithm:

### Order of samples in DIT and DIF

- The input samples in FFT algorithms DIF are organized in natural order but its output has not in order.
- The opposite is for DIT.




Order of intpus and outputs for DIF FFT.

# The Radix-$2^3$ FFT Algorithm

## The Radix-$2^3$ FFT Algorithm:

### Mathematic expression of radix-8 butterfly element

$$C_{8k+0} = \sum_{n=0}^{N/8-1} \left\{ [(x_n + x_{n+\frac{N}{2}}) + (x_{n+\frac{N}{4}} + x_{n+\frac{3N}{4}})] + [(x_{n+\frac{N}{8}} + x_{n+\frac{5N}{8}}) + (x_{n+\frac{3N}{8}} + x_{n+\frac{7N}{8}})] \right\} W_N^{0n} W_{N/8}^{nk}$$

$$C_{8k+4} = \sum_{n=0}^{N/8-1} \left\{ [(x_n + x_{n+\frac{N}{2}}) + (x_{n+\frac{N}{4}} + x_{n+\frac{3N}{4}})] - [(x_{n+\frac{N}{8}} + x_{n+\frac{5N}{8}}) + (x_{n+\frac{3N}{8}} + x_{n+\frac{7N}{8}})] \right\} W_N^{4n} W_{N/8}^{nk}$$

$$C_{8k+2} = \sum_{n=0}^{N/8-1} \left\{ [(x_n + x_{n+\frac{N}{2}}) - (x_{n+\frac{N}{4}} + x_{n+\frac{3N}{4}})] - j[(x_{n+\frac{N}{8}} + x_{n+\frac{5N}{8}}) - (x_{n+\frac{3N}{8}} + x_{n+\frac{7N}{8}})] \right\} W_N^{2n} W_{N/8}^{nk}$$

$$C_{8k+6} = \sum_{n=0}^{N/8-1} \left\{ [(x_n + x_{n+\frac{N}{2}}) - (x_{n+\frac{N}{4}} + x_{n+\frac{3N}{4}})] + j[(x_{n+\frac{N}{8}} + x_{n+\frac{5N}{8}}) - (x_{n+\frac{3N}{8}} + x_{n+\frac{7N}{8}})] \right\} W_N^{6n} W_{N/8}^{nk}$$

$$C_{8k+1} = \sum_{n=0}^{N/8-1} \left\{ [(x_n - x_{n+\frac{N}{2}}) - j(x_{n+\frac{N}{4}} - x_{n+\frac{3N}{4}})] + W_N^{N/8}[(x_{n+\frac{N}{8}} - x_{n+\frac{5N}{8}}) - j(x_{n+\frac{3N}{8}} - x_{n+\frac{7N}{8}})] \right\} W_N^{n} W_{N/8}^{nk}$$

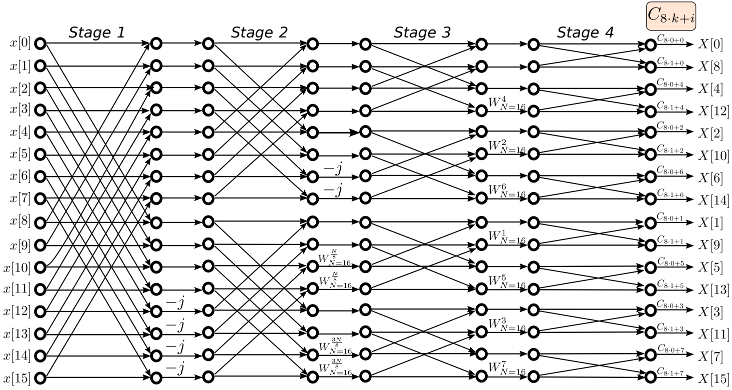
$$C_{8k+5} = \sum_{n=0}^{N/8-1} \left\{ [(x_n - x_{n+\frac{N}{2}}) - j(x_{n+\frac{N}{4}} - x_{n+\frac{3N}{4}})] - W_N^{N/8}[(x_{n+\frac{N}{8}} - x_{n+\frac{5N}{8}}) - j(x_{n+\frac{3N}{8}} - x_{n+\frac{7N}{8}})] \right\} W_N^{5n} W_{N/8}^{nk}$$

$$C_{8k+3} = \sum_{n=0}^{N/8-1} \left\{ [(x_n - x_{n+\frac{N}{2}}) + j(x_{n+\frac{N}{4}} - x_{n+\frac{3N}{4}})] + W_N^{3N/8}[(x_{n+\frac{N}{8}} - x_{n+\frac{5N}{8}}) + j(x_{n+\frac{3N}{8}} - x_{n+\frac{7N}{8}})] \right\} W_N^{3n} W_{N/8}^{nk}$$

$$C_{8k+7} = \sum_{n=0}^{N/8-1} \left\{ [(x_n - x_{n+\frac{N}{2}}) + j(x_{n+\frac{N}{4}} + x_{n+\frac{3N}{4}})] - W_N^{3N/8}[(x_{n+\frac{N}{8}} - x_{n+\frac{5N}{8}}) + j(x_{n+\frac{3N}{8}} - x_{n+\frac{7N}{8}})] \right\} W_N^{7n} W_{N/8}^{nk}$$

## Radix-$2^k$ Implementation

The quantity of rotators of an architecture radix-$2^k$ (with $k > 1$) is less than the radix-2.




Flow graph of a radix-$2^3$ 16-point DIF DFT.

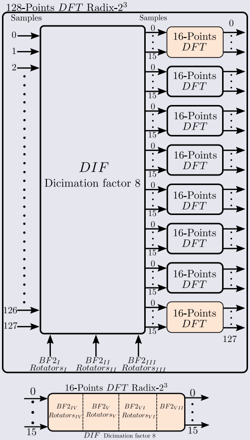
## The Radix-$2^3$ FFT Algorithm

- Applying (12) for the 128 point DFT and calculating each coefficient for $k = 0, 1, ..., (128/8) - 1$:

$$C_{8k+i} = \sum_{n=0}^{128/8-1} \{\cdot\}$$

  We get a sequence in chain of butterflies with its corresponding rotation factor.

- The 128 point DFT goes through a processes that involve tree stages of butterflies to arrive finally to a set of eight DFT where each of one is a 16 point DFT.



# Questions

**Thanks!**

Fundación
FULGOR