The Fast Fourier Transform (FFT)

Lecture 20

Josh Brake
Harvey Mudd College

Learning Objectives

By the end of this lecture you will be able to:

- Recall the basic mathematical structure of the Discrete Fourier Transform (DFT)
- Understand how the FFT is used to efficiently compute the DFT
- Be able to sketch a block diagram of the basic blocks needed to implement an FFT on an FPGA.

Outline

- Review of the Fourier Transform
  - Continuous Fourier Transform
  - Discrete Fourier Transform (DFT)
- The Fast Fourier Transform (FFT)
- The FFT on an FPGA

The Discrete Fourier Transform (DFT)

The Discrete Fourier Transform (DFT)

Frequency Domain Coefficients

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot W^{k \cdot n}$$

Time Domain Coefficients

$$x[n] = \sum_{n=0}^{N-1} X[k] \cdot W^{-k \cdot n}$$

$$W = \exp[-j2\pi/N] = \cos\left(\frac{2\pi}{N}\right) - j\sin\left(\frac{2\pi}{N}\right)$$

- $x[n]$: time domain samples (complex)
- N: number of samples
- $X[k]$: frequency domain coefficients (complex)
- $W = \exp[-j2\pi/N]$: “roots of unity” or twiddle factors (note: sometimes sign is flipped)
- n: time domain index
- k: frequency domain index

Example Signal

$$x[n] = \cos(2\pi f n \Delta t)$$

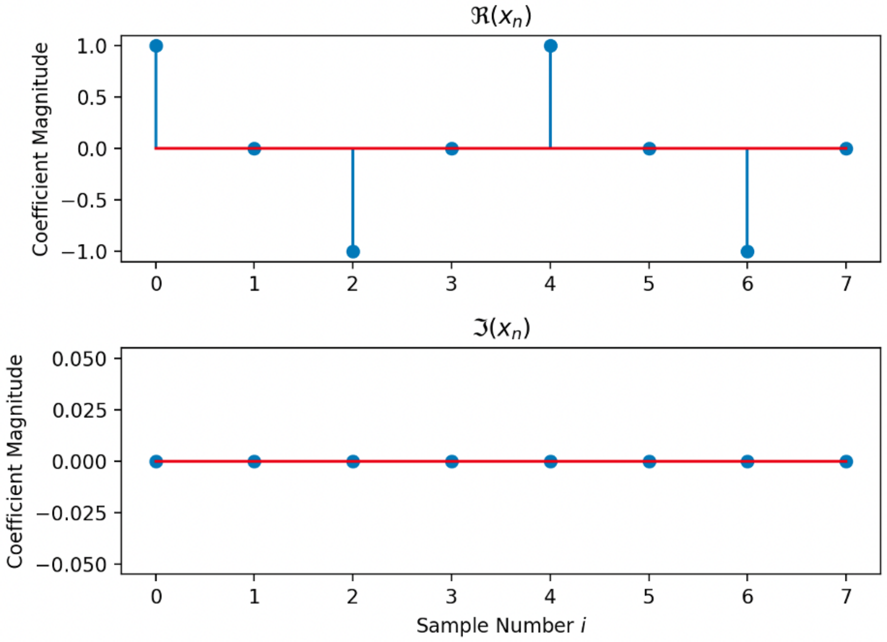
- $x[n]$: Samples
- f : Signal frequency
- n: Index

Consider a signal with $N = 8$ samples

- $x[n] = \cos(2\pi f n \Delta t)$
- $n = 0, 1, 2, \ldots, N - 1$
- $\Delta t = 1/f_s$ where f_s is the sampling frequency
- $W = \exp[-j2\pi/N] = \exp[-j\pi/4]$

Example Signal

$$x[n] = \cos(2\pi f n\Delta t)$$



Set $f = 2$ Hz and $f_s = 8$ Hz. So, this means that we have 4 samples per period of the sinusoid.

$$x[n] = [1, 0, -1, 0, 1, 0, -1, 0]$$

Example: DFT computation

- Compute DFT of the signal
- $W = \exp[-j2\pi/N] = \exp[-j\pi/4]$

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot W^{k \cdot n}$$

DFT Computational Complexity

- How many multiplications do we need to perform to compute each Fourier coefficient?
  - N complex multiplications ($2N$ real multiplies if signal is real; {$4N$}{.blank-underline} real multiplications if signal is complex)
  - Assuming a real input signal, each frequency requires $2N$ multiplies and $2(N-1)$ additions since it costs $N-1$ additions to add N numbers and we need to do this for both real and imaginary parts.
  - So, overall computational complexity for a single frequency is $2N+2(N-1)=4N-2$ and we have N frequencies for a total of $N(4N-2)$ computations.
  - This means the DFT is $O(N^2)$.

Can we do better?

The Fast Fourier Transform

The Fast Fourier Transform

How can recursion help us to simplify the computation?

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot W^{k \cdot n}$$

The derivation in the following slides is based on this paper.

A Tutorial-style Single-cycle Fast Fourier Transform Processor

Alec Vercruysse*
M. Weston Miller*
Joshua Brake†
David Harris
avercruysse@g.hmc.edu
wmiller@g.hmc.edu
jbrake@hmc.edu
David_Harris@hmc.edu
Department of Engineering, Harvey Mudd College
Claremont, CA, USA

History: DFT to FFT

- Algorithm invented by Carl Friedrich Gauss around 1805
- Published in 1965 by Cooley (IBM) and Tukey (Princeton)
- Widely used throughout signal processing.

An Algorithm for the Machine Calculation of Complex Fourier Series

By James W. Cooley and John W. Tukey

Symmetries of the roots of unity

- $W^n = \exp\left(-j\frac{2\pi}{N}n\right)$
- $W^n = W^{(n+Nk)}$
- $W^n = -W^{n+N/2}$
- $W^{Nk} = 1$

Decimation in Time (DIT) DFT

- Express the DFT as a sum of two DFTs by splitting the signals into two based on even and odd indices.

$$X[k] = \sum_{n=0}^{N/2-1} x[n] \cdot W^{k \cdot n}$$

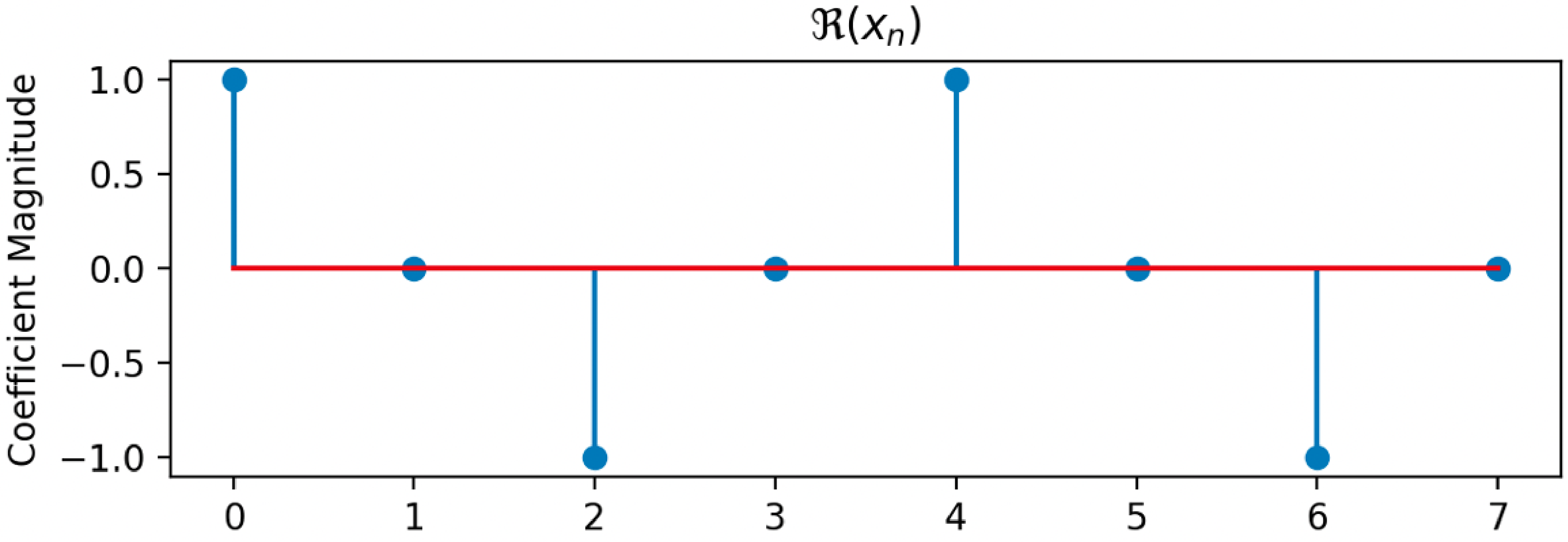
$$X[k] = \sum_{n=0}^{N-1} x[2n] \cdot W^{k \cdot 2n} + \sum_{n=0}^{N-1} x[2n+1] \cdot W^{k \cdot (2n+1)}$$

$$X[k] = \sum_{n=0}^{N-1} x[2n] \cdot W^{k \cdot 2n} + {\color{red} W^{k}} \sum_{n=0}^{N-1} x[2n+1] \cdot W^{k \cdot 2n}$$

Decimation in Time (DIT) DFT

$$X[k] = \sum_{n=0}^{N-1} x[2n] \cdot W^{k \cdot 2n} + W^k \sum_{n=0}^{N-1} x[2n+1] \cdot W^{k \cdot 2n}$$

Which samples are in each sum?

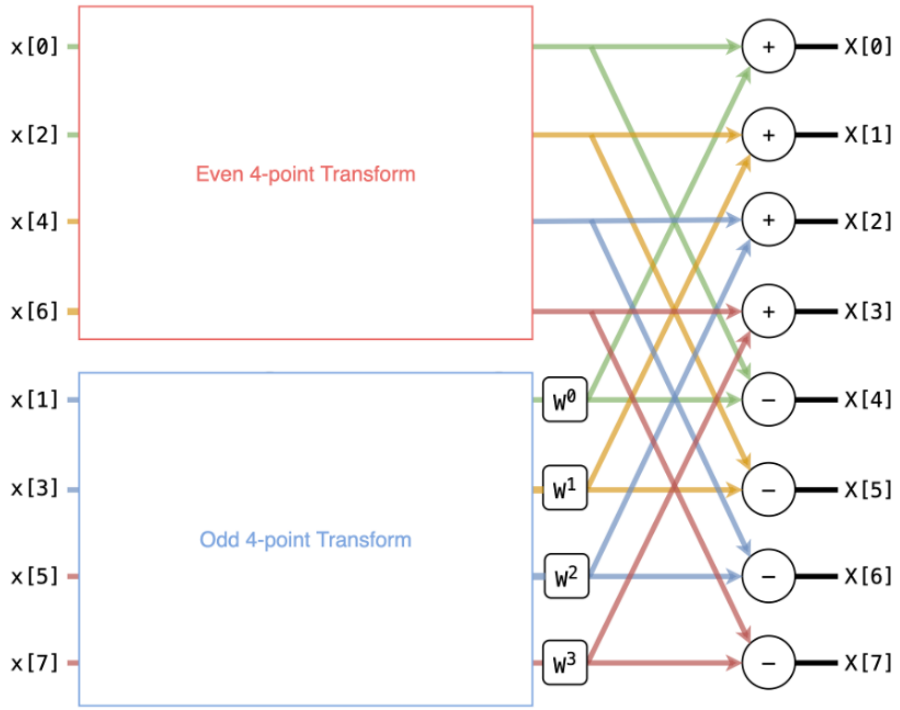


DFT Recursion

$$X[k] = \sum_{n=0}^{N-1} x[2n] \cdot W^{k \cdot 2n}$$

$$+ W^{k} \sum_{n=0}^{N-1} x[2n+1] \cdot W^{k \cdot 2n}$$

This division is equivalent to two separate $N/2$ DFTs with a final multiplication and addition per output term.



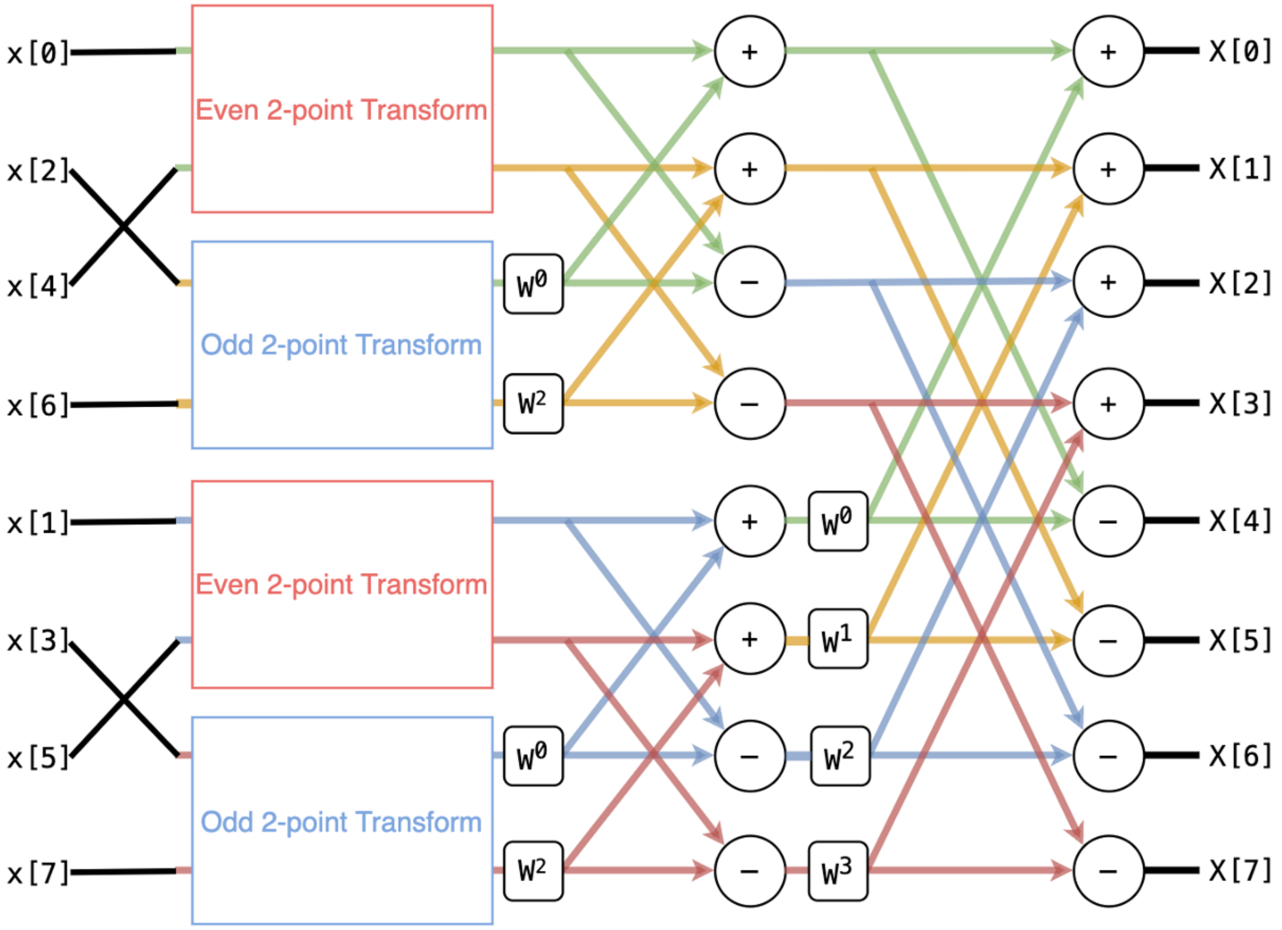
Use recursion to solve 4-point transforms

- Apply the DIT recursion formula to find the 4-point DFT as the sum of two 2-point DFTs.
- Write $X[1]$ (the first output of the 4-point DFT) explicitly using the equation below.

$$X[k] = \sum_{n=0}^{N-1} x[2n] \cdot W^{k \cdot 2n} + W^{k} \sum_{n=0}^{N-1} x[2n+1] \cdot W^{k \cdot 2n}$$

$$X[k] = x[0] \cdot W^{k \cdot 0} + x[2] \cdot W^{k \cdot 2} + W^{k} \left(x[1] \cdot W^{k \cdot 0} + x[3] W^{k \cdot 2} \right)$$

Use recursion to solve 4-point transforms



What is the 2-point DFT?

$N = 2$

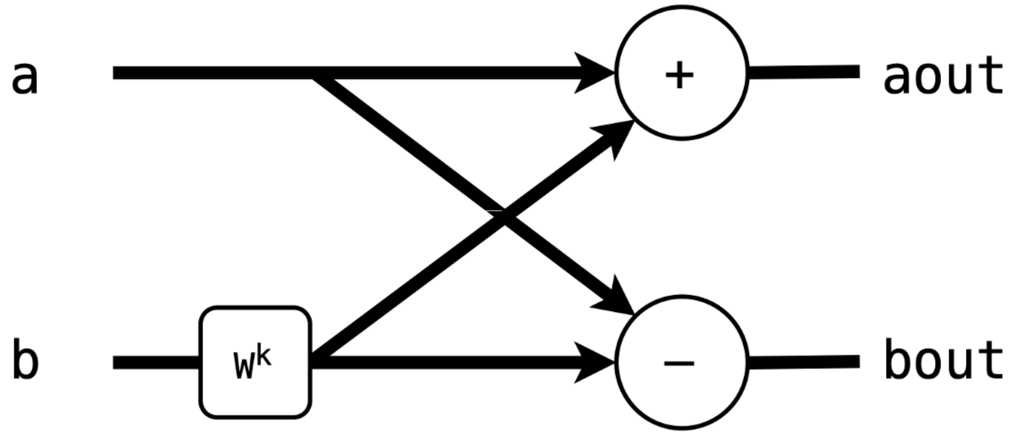
$$X[k] = \sum_{n=0}^{N-1} x[2n] \cdot W^{k \cdot 2n} + W^k \sum_{n=0}^{N-1} x[2n+1] \cdot W^{k \cdot 2n}$$

$X[k] = x[0] \cdot W^{k \cdot 0} + W^k \left(x[1] \cdot W^{k \cdot 0}\right)$

$X[k] = x[0] + W^k \cdot x[1]$

The butterfly unit

- Two complex inputs: A & B
- Multiplies B by the twiddle factor
- Computes sum and difference



FFT Activity

Build your own FFT

Using the template diagrams on the following slides, fill in the missing information.

- The inputs are samples $x[n]$. You need to determine which samples should be connected to each input.
- The rounded rectangles indicate multiplications by twiddle factors W^k in the butterfly units. You need to determine what the twiddle factors should be.
- The circle indicate either addition or subtraction operations

The most straightforward way to figure this out is to evaluate the time-decimated DFT equation and look for where the nesting occurs. (e.g., the 4-point DFT is the combination of two, 2-point DFTs.)

Step 1: 4-point FFT

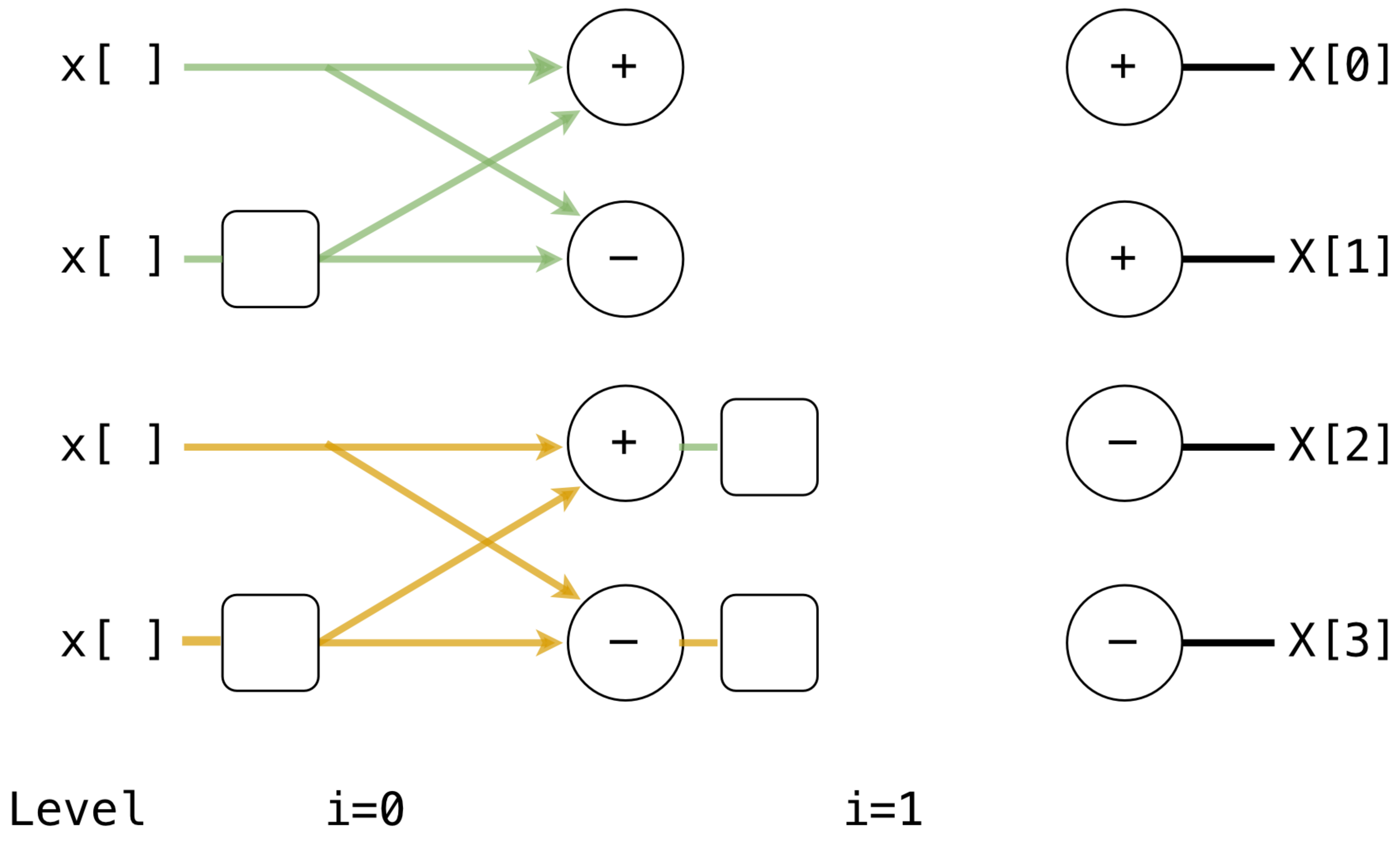
Complete the diagram below

- [ ] Fill in the rounded rectangles with the appropriate twiddle factors
- [ ] Draw arrows to route signals properly

Hints:

- Each stage includes $\log_2(N)$ butterfly units.

Step 1: 4-point FFT

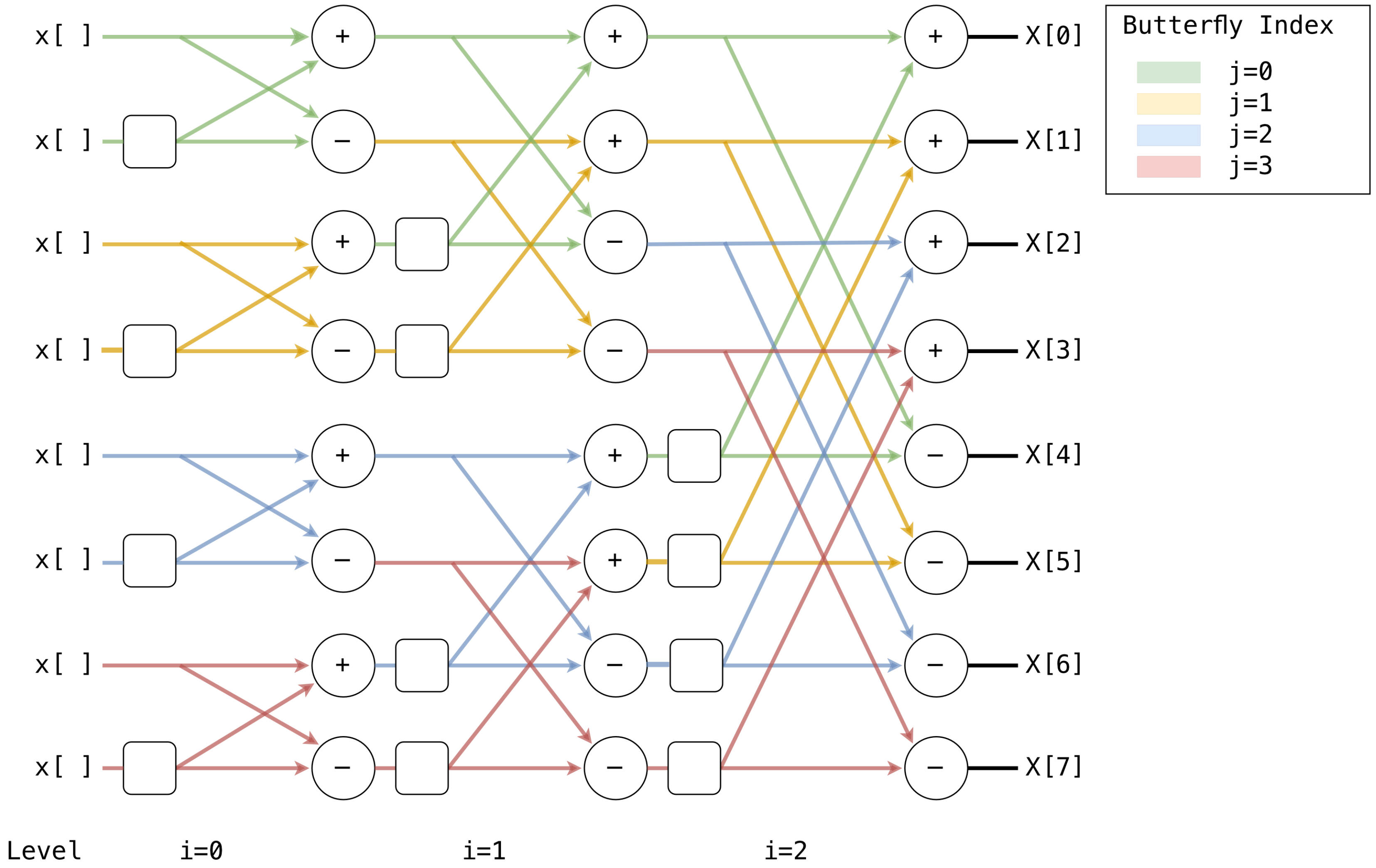


Step 2: 8-point DFT

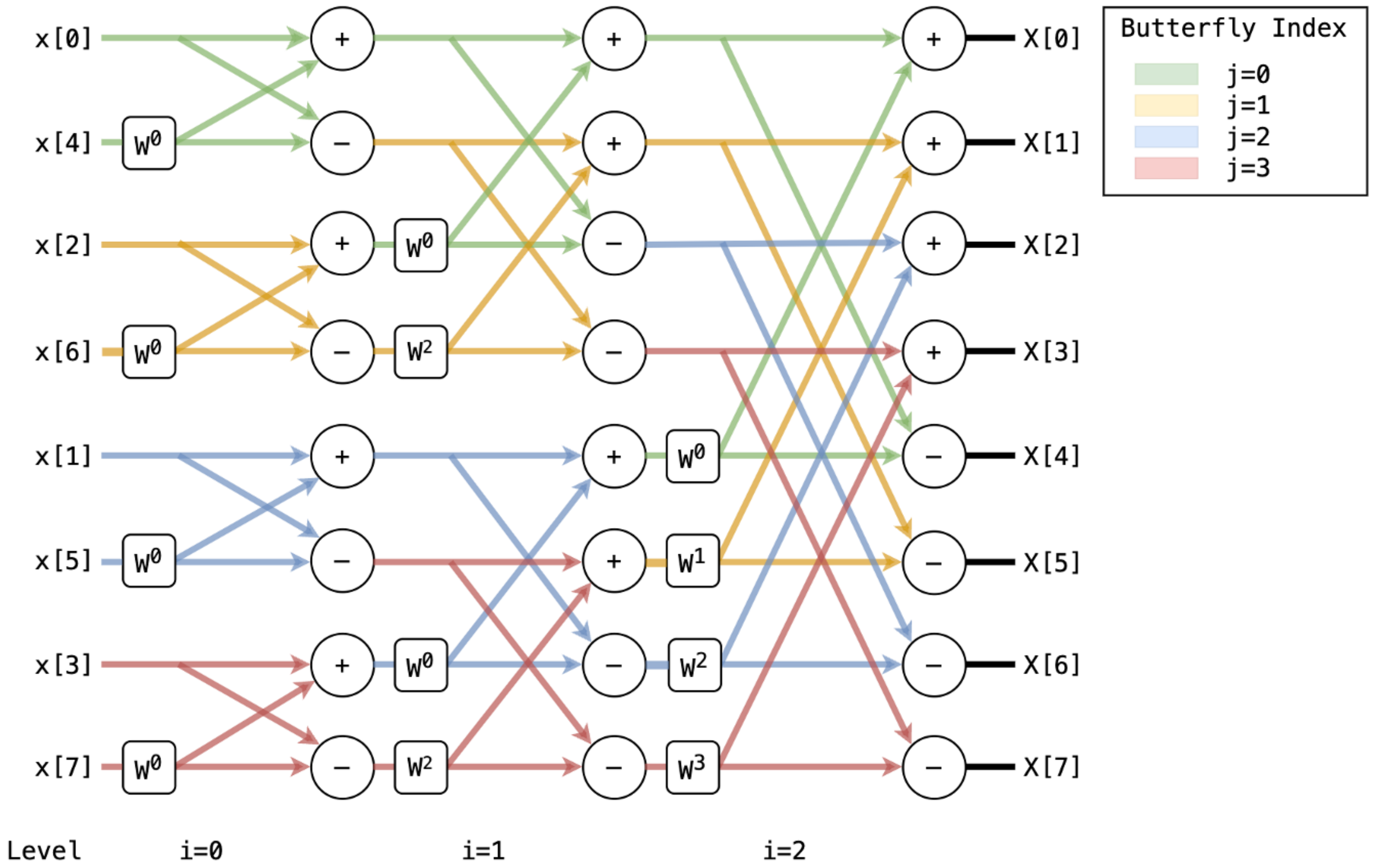
Using the process from the previous slide, repeat this for an 8-point DFT.

Note that the top-left portion of the diagram is identical to the 4-point DFT. You simply add another copy below on the bottom left and then combine them with one more stage of butterflies.

Step 2: 8-point DFT



Solution


x[0]
x[4]
x[2]
x[6]
x[1]
x[5]
x[3]
x[7]
W0
W2
W1
W3
X[0]
X[1]
X[2]
X[3]
X[4]
X[5]
X[6]
X[7]
Butterfly Index
j=0
j=1
j=2
j=3
Level
i=0
i=1
i=2

Summary

- N-point FFT can be computed with…
  - $\log_2(N)$ levels of transforms
  - Each transform has $N/2$ steps of butterfly operations
  - Each butterfly operation consists of a complex multiplication by a twiddle factor and a complex addition and subtraction.
  - Total of $(N/2)\log_2(N)$ complex multiplications are needed.

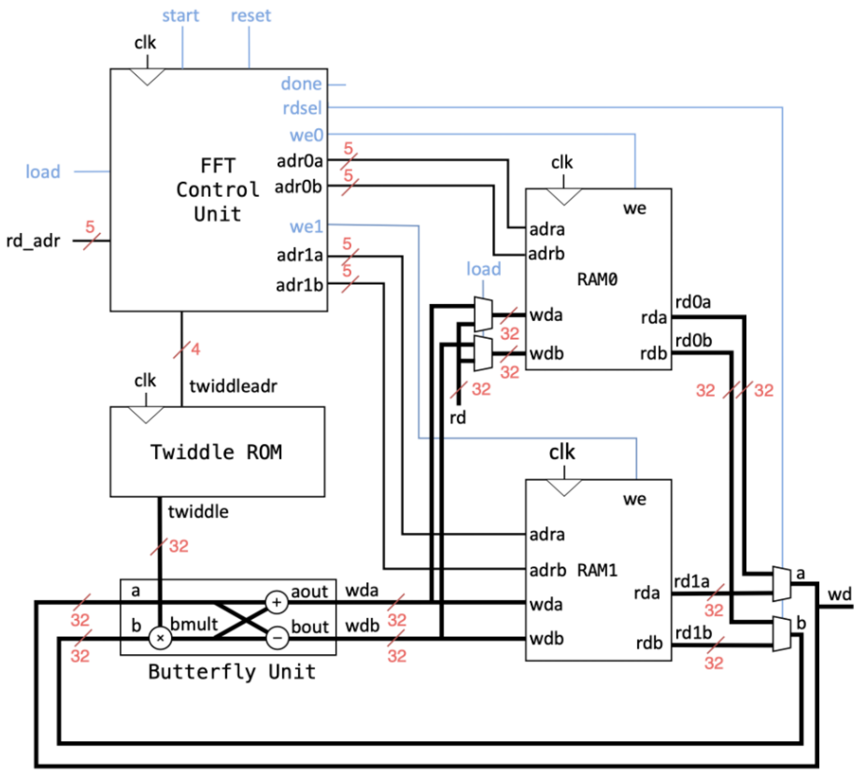
FFT in Software

```
Bit-reverse-copy (x, X)              // initialize X to bit-reversed order
w=e^(-j*2*pi/N)
w[0] = 1;                          // initialize twiddle factor table:
for k = 1 to N/2 - 1
    w[k] = w[k-1] * w                // w[k] holds wk

M = 2                              // size of the transform at current level
t = log2(N)-1                      // stride through twiddle factors
for s = 0 to log2(N)-1               // level s
    for k = 0 to N-1 by M            // iterate over M-point transforms
        for j = 0 to M/2 - 1         // butterflies in M-point transform
            a = X[k+j]
            b = X[k+j+M/2]
            twiddle = w[j << t]
            p = b * twiddle          // multiply by appropriate twiddle factor
            X[k+j] = a + p
            X[k+j+M/2] = a - p
    M = M << 1                     // boost size of transform for next level
    t = t - 1                      // stride of twiddle factors gets smaller
```

FFT in Hardware

- Consider the FFT as a processor like the designs in E85
- Two major components
  - Datapath
  - Controller



A. Vercruysse, M. W. Miller, J. Brake, and D. Harris. 2022. A Tutorial-style Single-cycle Fast Fourier Transform Processor. https://doi.org/10.1145/3526241.3530329

Conclusion

By the end of this lecture you will be able to:

- Recall the basic mathematical structure of the Discrete Fourier Transform (DFT)
- Understand how the FFT is used to efficiently compute the DFT
- Be able to sketch a block diagram of the basic blocks needed to implement an FFT on an FPGA.