

# **ELEC 421**

## **Digital Signal and Image Processing**



**Siamak Najarian, Ph.D., P.Eng.,**  
Professor of Biomedical Engineering (retired),  
Electrical and Computer Engineering Department,  
University of British Columbia

## Course Roadmap for DSP

| Lecture    | Title   |
|------------|---|
| Lecture 0  | Introduction to DSP and DIP   |
| Lecture 1  | Signals   |
| Lecture 2  | Linear Time-Invariant System  |
| Lecture 3  | Convolution and its Properties  |
| Lecture 4  | The Fourier Series  |
| Lecture 5  | The Fourier Transform   |
| Lecture 6  | Frequency Response  |
| Lecture 7  | Discrete-Time Fourier Transform   |
| Lecture 8  | Introduction to the z-Transform   |
| Lecture 9  | Inverse z-Transform; Poles and Zeros  |
| Lecture 10 | The Discrete Fourier Transform  |
| Lecture 11 | Radix-2 Fast Fourier Transforms   |
| Lecture 12 | The Cooley-Tukey and Good-Thomas FFTs                                       |
| Lecture 13 | The Sampling Theorem  |
| Lecture 14 | Continuous-Time Filtering with Digital Systems; Upsampling and Downsampling |
| Lecture 15 | MATLAB Implementation of Filter Design                                      |

# Lecture 11: Radix-2 Fast Fourier Transforms

## Table of Contents

- Recap of DFT and DTFT; what is the FFT?
- The DFT formula
- The naive DFT formula is  $O(N^2)$
- Characteristics of FFT algorithms
- Simplifications involving  $W_N$
- Decimation in time
- The DIT formula
- Example with  $N = 8$ : block diagram
- Completed block diagram (first stage)
- Computational cost of first-stage decomposition
- Going down another level
- Completed block diagram (second stage)
- Going down to length-2 DFTs
- Completed block diagram (all stages)
- The final computational cost is  $O(N \log_2 N)$
- The "butterfly"
- Matrix interpretation of decimation in time
- $F_8$  in terms of  $F_4$
- Twiddle factors
- Decimation in frequency

## Recap of DFT and DTFT; what is the FFT?

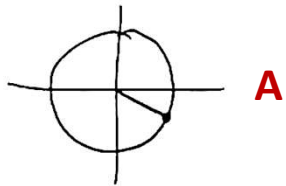
- First, what we want to go back to in this lecture is the discrete Fourier transform which we introduced last time. That is basically an entirely numerical operation that MATLAB does when we ask a computer for FT. We showed that what we really do when we do an FFT in MATLAB is we are computing the DTFT, the discrete-time Fourier transform, which is ideally a continuous function of  $\omega$  and we are sampling it at equal intervals.
- So, if we are asked for a **10**-point DFT, what we are getting is the DTFT sampled at  $2\pi/10$ . Here, we are getting these equally-spaced points. And, if we want a more finely-spaced DTFT, then we just ask for a longer FFT. That is, we would say, give us **200** points or a **1000** points. In that way, basically MATLAB or any other software like that can do something that approximates the continuous frequency operations that we want. In the future, we will talk a little bit more about how good that sampling has to be. That is what the sampling theorem is all about.
- What we want to focus on in this lecture and in the next one are **efficient methods for computing the discrete FT**. Note that the MATLAB command is called FFT and not DFT. The reason for that is the F stands for “fast”. Let us say we want to do Fourier transforms. If we look at the naive way of writing the Fourier transform, which we are going to do next, we realize that it takes potentially a lot of multiplications and a lot of adds (i.e., additions). Here, we are not going to focus much on computational complexity or various algorithms, but we **do** want to talk about that a little bit in this lecture. It became clear that if we wanted to compute the DFT's of matrices that were thousands of elements on a side, we would probably need to have a more efficient method. Otherwise, even the computers of today could not handle them and so these efficient methods were discovered that can make the whole DFT process a lot faster. That is what we want to emphasize in the next couple of lectures.

## The DFT formula

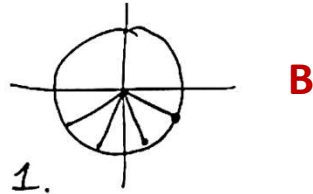
THE DFT

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-jk \frac{2\pi}{N} n} \quad (1) \quad \begin{matrix} n = 0, 1, \dots, N-1 \\ k = 0, 1, \dots, N-1 \end{matrix}$$

$$= \sum_{n=0}^{N-1} x[n] W_N^{kn}$$

$$W_N = e^{-j \frac{2\pi}{N}} \quad (2)$$


**A**

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


**B**

AN  $N^{\text{th}}$  root of 1.

$$W_N, W_N^2, W_N^3, \dots, W_N^N \quad N^{\text{th}} \text{ roots of } 1$$

- Let us just revisit the formula for the DFT, **(1)**. The formula for the DFT is basically saying we have **N** numbers of these little **x[n]**'s coming in and we multiply them by this complex scalar, **(\*)**, and then we sum them up. Here, **n** ranges from **0** to **N-1** and **k** ranges from **0** to **N-1**, too.
- So, we take **N** numbers in, we get **N** numbers out. Those **N** numbers that we get out are samples of the DTFT. It is like a discrete representation of frequency. We also abbreviated this to say we could write **(1)** in a slightly more compact way by giving this complex exponential a name, i.e.,  **$W_N^{kn}$** . Here,  **$W_N$**  is defined as **(2)**.
- Just as a reminder that  **$W_N$**  is like saying that we have a point on the unit circle, shown in **A**. If we were to take  **$W_N$**  to the **N**th power, we would come all the way back to **1**. So, this is an **N**th root of **1**. And, the other **N**th roots of **1**, shown in **B**, are what we get if we were to square it, cube it, or take it to the fourth power, and so on.

## The naive DFT formula is $O(N^2)$

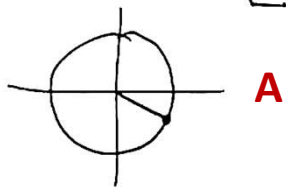
THE DFT

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-jk \frac{2\pi}{N} n} \quad (1) \quad \begin{matrix} n = 0, 1, \dots, N-1 \\ k = 0, 1, \dots, N-1 \end{matrix}$$

(\*\*)

$$= \sum_{n=0}^{N-1} x[n] W_N^{kn}$$

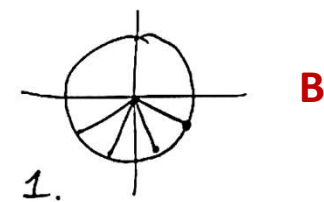
$\rightarrow N^2$  (complex) MULTIPLIES  
 $N(N-1)$  (complex) ADDS

$$W_N = e^{-j \frac{2\pi}{N}} \quad (2)$$


**A**

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

AN  $N^{\text{th}}$  root of 1.



**B**

$$W_N, W_N^2, W_N^3, \dots, W_N^N \quad N^{\text{th}} \text{ roots of } 1$$

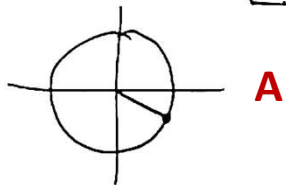
- As shown in **B**, there are **N**  $N^{\text{th}}$ -roots of **1**. So, if we have  $W_N, W_N^2, W_N^3$ , all the way up to  $W_N^N$ , which is actually **1** itself, these are all  $N^{\text{th}}$  roots of **1**. This means that there are  $N^{\text{th}}$  square roots of **1**, in general, and those are evenly-spaced around the unit circle.
- For the moment, we are going to assume that the input and the output could be general complex numbers. **Let us think about how many multiplications and additions we need to do to accomplish the computation of  $X[k]$  in (1).** That means in order to get  $X[k]$  values, we need to do **N** complex multiplies. This is because inside this sum (\*\*), we have **N** products and then we have **N** values that we are computing for  $X[k]$ . So, overall, what we have is **N.N =  $N^2$  complex multiplies**. And, then again inside the sum (\*\*), we need to add up these **N** numbers. This is basically like **N-1** additions inside the sum and we have to do that **N** times, too. So, there is  **$N(N-1)$  complex additions**.

## The naive DFT formula is $O(N^2)$

THE DFT

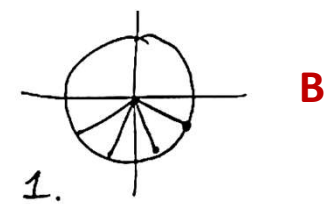
$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-jk \frac{2\pi}{N} n} \quad \begin{matrix} n=0, 1, \dots, N-1 \\ k=0, 1, \dots, N-1 \end{matrix} \quad (*) \quad (1)$$

$$= \sum_{n=0}^{N-1} \overbrace{x[n] W_N^{kn}}^{(**)} \quad \left[ \begin{array}{l} N^2 \text{ (complex) MULTIPLIES} \\ N(N-1) \text{ (complex) ADDS} \end{array} \right]$$

$$W_N = e^{-j \frac{2\pi}{N}} \quad (2) \quad \text{A}$$


$$W_N = e^{-j \frac{2\pi}{N}} \quad \text{B}$$

AN  $N^{\text{th}}$  root of 1.



$$W_N, W_N^2, W_N^3, \dots, W_N^N \quad N^{\text{th}} \text{ roots of } 1$$

$$(a+bj)(c+dj) = (ac-bd) + (bc+ad)j \quad (3)$$

- Now, let us figure out how many corresponding **real** multiplies and **real** additions we need to do. It is going to be more or less **4** times the number we got before. This is because based on (3), if we want to do a complex multiply between  $(a+bj)$  and  $(c+dj)$ , that means we have to compute the **4** real products shown on the right side of equation (3).
- There are some faster ways of doing that if we really needed to. But, for the moment, the point we want to emphasize is that this whole thing requires **order of  $N^2$  operations**. When **N** is like a thousand dimensional vector, that is like a million operations, and as **N** gets bigger, things get worse and worse.



## The naive DFT formula is $O(N^2)$

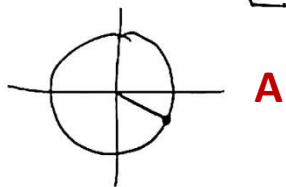
THE DFT

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-jk \frac{2\pi}{N} n} \quad (1) \quad \begin{matrix} n = 0, 1, \dots, N-1 \\ k = 0, 1, \dots, N-1 \end{matrix}$$

(\*\*)

$$= \sum_{n=0}^{N-1} x[n] W_N^{kn}$$

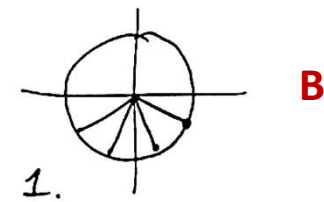
$\rightarrow N^2$  (complex) MULTIPLIES  
 $N(N-1)$  (complex) ADDS

$$W_N = e^{-j \frac{2\pi}{N}} \quad (2)$$


A

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

AN  $N^{\text{th}}$  root of 1.



B

$$W_N, W_N^2, W_N^3, \dots, W_N^N \quad N^{\text{th}} \text{ roots of } 1$$

$$(a+bj)(c+dj) = (ac-bd) + (bc+ad)j \quad (3)$$

- What can we do to make this process more efficient? On the face of it, if the equation of sum for  $X[k]$  was just a normal matrix, a matrix that had say **random entries**, there really is not that much more we could do to make our life any easier and the process any faster. This is because matrix multiplication is inherently this  $N^2$  operation. But what we are going to use is the fact that this is a **very special matrix multiplication**. If we look at what this matrix multiplication is, we see that the entries of that matrix fall apart into really interesting and simple patterns. That will make our life much easier and will enable us to drive the **computational cost** much lower than  $N^2$ . That is what is called the **FFT**.

## Characteristics of FFT algorithms

FFT (FAST FOURIER TRANSFORM)  
REDUCE # OF OPERATIONS TO  $O(N \log_2 N)$

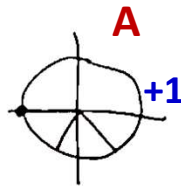
- The FFT is not like a single monolithic algorithm. The FFT generally refers to any scheme that makes the DFT faster. The FFT stands for **fast Fourier transforms**, and in general, these algorithms reduce the number of operations to the order of  $N \log_2(N)$ .
- For example, let us say we have  $N = 1000$ . So,  $N^2 = (1000)^2 = 1,000,000$ . Also,  $N \log_2(N) = (1000) \cdot \log_2(1000) = (1000) \cdot \log_{10}(1000) / \log_{10}(2) = 9965.7 \approx 10,000$ . That is the difference between 1,000,000 operations for the naive way (also called the **brute force method**) and 10,000 operations for the smart way. This is a savings by a factor of 100, just for a matrix that is a 1000 elements long, which is really not that long. So, we can see this is really worthwhile if we can make it happen.
- The notation of  $O(N \log_2(N))$  is sometimes called **computational complexity**.

## Simplifications involving $W_N$

FFT (FAST FOURIER TRANSFORM)  
REDUCE # OF OPERATIONS TO  $O(N \log_2 N)$

- DECOMPOSITIONS INTO SMALLER DFTS

- SIMPLIFICATIONS  $W_N^{(kN)} = 1$  (1)



- What we are going to do to make this possible is observe a bunch of things about the character of the DFT. The main idea is going to be **decompositions into smaller DFT's**. There is going to be this neat way of *recursively* breaking up a long FFT into a whole bunch of shorter FFTs, which are easy to compute.
- We are also going to leverage some simple things. For example, there are a lot of **simplifications that are related to  $W_N^{(kN)}$** . For instance, we know that whenever we see  $W_N^{(kN)}$  (where  $k$  is an integer), that is equal to **+1**, as shown by (1). It is like going around the unit circle  $k$  times and ending up at **+1**, as shown in **A**. So, anytime we see something like this,  $W_N^{(kN)}$ , we can immediately turn it into a **+1**. That is, multiplying by this number,  $W_N^{(kN)}$ , is no longer a complex multiply, and nothing is happening.

## Simplifications involving $W_N$

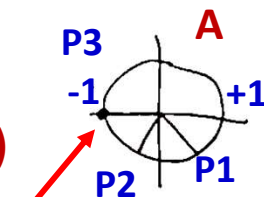
FFT (FAST FOURIER TRANSFORM)  
REDUCE # OF OPERATIONS TO  $O(N \log_2 N)$

- DECOMPOSITIONS INTO SMALLER DFTS

- SIMPLIFICATIONS

$$W_N^{(kN)} = 1 \quad (1)$$

$$W_N^{N/2(\text{odd } k)} = -1 \quad (2)$$



- $W_N^{n(k+N)} = W_N^{k(n+N)} = W_N^{kn}$  PERIODICITY (3)

- $W_N^{k(N-n)} = W_N^{-kn} = (W_N^{kn})^*$  SYMMETRY (4)

- In the same way, if we have  $W_N^{(N/2)(\text{odd } k)}$ , (2), we will end up at **-1**. In graph **A**, let us say we have  $W_6^3$ . That is, if we take  $W_6$  at **P1** to the third power, we end up at point **P3**, which is at **-1**. So, these are some other special numbers where we end up at the opposite end of the unit circle. Multiplying something by **-1** is again nothing computationally stressful. It is just like flipping the sign, so that is not a problem either.
- There are a bunch of other facts, shown in (3) and (4), that we are going to definitely leverage throughout this whole process.
- Remember that when we do the DFT operation, we are assuming that  $x[n]$  and  $X[k]$  are periodic with period **N**. That means we are thinking of the input signal as a periodic discrete-time signal. Also, we are thinking about the output frequency as a periodic set of numbers. So, that means whenever we see something like  $W_N^{n(k+N)}$  or  $W_N^{k(n+N)}$ , the values of the exponents are going to be bigger than **N**, but we can always take out multiples of **N** to come back to a simpler number shown in (3) by  $W_N^{kn}$ . This is because we know that  $W_N^N = 1$ .

## Simplifications involving $W_N$

FFT (FAST FOURIER TRANSFORM)  
REDUCE # OF OPERATIONS TO  $O(N \log_2 N)$

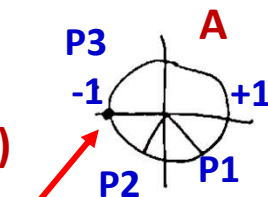
- DECOMPOSITIONS INTO SMALLER DFTS

- SIMPLIFICATIONS  $W_N^{(kN)} = 1$  (1)

$$W_N^{N/2(\text{odd } k)} = -1$$
 (2)

- $W_N^{n(k+N)} = W_N^{k(n+N)} = W_N^{kn}$  PERIODICITY (3)

- $W_N^{k(N-n)} = W_N^{-kn} = (W_N^{kn})^*$  SYMMETRY (4)



- Even though we may see numbers like (3) in the matrix, they really cannot be any more than  $N$  unique  $W$ 's inside the matrix to use in order to compute the FFT. So, these are called **periodicities**, as shown in (3).
- There are also **symmetries** that are shown in (4). For example, if we have  $W_N^{k(N-n)}$ , again, we can take out this  $kn$ , and turn it into  $W_N^{-kn}$ , which is the same thing as  $(W_N^{kn})^*$ , which is the conjugate of  $W_N^{kn}$ . This property is also very useful and easy to implement.
- We are going to talk about ways of leveraging these kinds of pieces of information to make our life easier!

## Decimation in time

DECIMATION IN TIME ( $N$  EVEN)

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk} \quad (1)$$

$$= \overbrace{\sum_{n \text{ EVEN}} x[n] W_N^{nk}}^{(*)} + \overbrace{\sum_{n \text{ ODD}} x[n] W_N^{nk}}^{(**)} \quad (2)$$

$$\stackrel{n=2r}{=} \sum_{r=0}^{N/2-1} x[2r] W_N^{2rk} + \sum_{r=0}^{N/2-1} x[2r+1] W_N^{(2r+1)k} \quad (3)$$

- What we want to talk about first is an algorithm that is called **decimation in time FFT**, or **DIT FFT**. We are going to start by assuming that  $N$  is an **even number**. That is not a restrictive assumption because keep in mind that many times when we are doing the DFT, we are thinking about the DFT as a way of sampling the underlying continuous frequency response. That means we are often free to choose what  $N$  we want. If we have a weird FFT that is of length **987**, we could just bump that up to  $2^{10} = 1,024$ , and then just get a finer sampling. So, choosing an even  $N$  is not an issue.
- Because we are going to be writing  $X[k]$  formula so many times, **(1)**, we are definitely going to stick with  $W_N$  notation. Given that  $N$  is an even number, we split the sum into something that involves only the even entries of  $x[n]$ , **(\*)**, and something that only involves the odd entries of  $x[n]$ , **(\*\*)**, as shown in **(2)**. In **(2)**, we are not changing anything by doing this.
- If  $n$  in **(\*)** is even, we can think about that as  $n = 2r$ , where  $r$  is some integer. Let us say  $N = 6$ . So,  $n = 0,1,2,3,4,5$  in **(1)**. This is a different way of saying that the sum **(\*)** is going from, **0** to **4**, i.e.,  $n = 0,2,4$ . We can instead make this sum go from  $r = 0$  to  $N/2 - 1 = 6/2 - 1 = 2$ . Using  $n = 2r$ , for  $n = 0$ , we get  $r = 0$ , for  $n = 2$ , we get  $r = 1$ , and for  $n = 4$ , we get  $r = 2$ , or  $r = 0,1,2$ . By writing it in this way, we will have a simpler index that goes up to roughly half the length (i.e., to  $N/2-1$ ), as shown in **(3)**.

## Decimation in time

DECIMATION IN TIME ( $N$  EVEN)

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk} \quad (1)$$

$$= \overbrace{\sum_{n \text{ EVEN}} x[n] W_N^{nk}}^{(*)} + \overbrace{\sum_{n \text{ ODD}} x[n] W_N^{nk}}^{(**)} \quad (2)$$

$$\stackrel{n=2r}{=} \sum_{r=0}^{N/2-1} x[2r] W_N^{2rk} + \sum_{r=0}^{N/2-1} x[2r+1] W_N^{(2r+1)k} \quad (3)$$

$$= \sum_{r=0}^{N/2-1} x[2r] (W_N^2)^{rk} + W_N^k \sum_{r=0}^{N/2-1} x[2r+1] (W_N^2)^{rk} \quad (4)$$

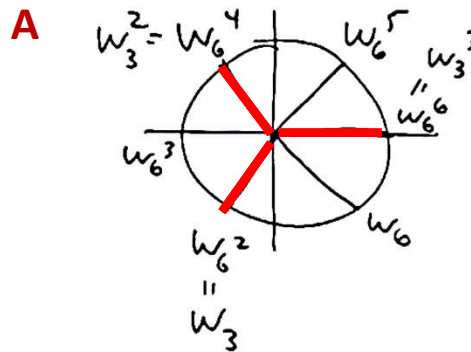
- As pointed out before, and for our example, in **(3)**, for the first sum,  $r$  would go from **0** to **2**, i.e.,  $r = 0, 1, 2$ . That would be the same as saying  $x[2r]$  is  $x[0]$ ,  $x[2]$ , and  $x[4]$ , i.e., going from  $x[0]$  to  $x[4]$ . So, we are not really changing anything in this sum.
- In the same way and for the second sum in **(3)**, we could write an odd number as  $n = 2r + 1$ . And, again if this was equal to a 6 DFT,  $N = 6$ , this would be like  $N/2 - 1 = 6/2 - 1 = 2$ , and we would be summing over  $x[1]$ ,  $x[3]$ , and  $x[5]$ . After re-writing **(3)** in a slightly different way, we arrive at **(4)**.
- This is where the magic happens. Each sum in **(4)** looks like we are taking a length- $N/2$   $X[k]$  for each of the sum. This looks suspiciously like the formula for a smaller DFT.



## The DIT formula

$$X[k] = \sum_{r=0}^{N/2-1} x[2r] (W_N^2)^{rk} + W_N^k \sum_{r=0}^{N/2-1} x[2r+1] (W_N^2)^{rk} \quad (1)$$

(\*)



$$= \underbrace{\sum_{r=0}^{N/2-1} x[2r] W_{N/2}^{rk}}_{\text{LENGTH } N/2 \text{ DFT OF EVEN ENTRIES}} + W_N^k \underbrace{\sum_{r=0}^{N/2-1} x[2r+1] W_{N/2}^{rk}}_{\text{LENGTH } N/2 \text{ DFT OF ODD ENTRIES}} \quad (2)$$

(\*\*)      (\*\*\*)

$G[k]$        $W_N^k H[k]$        $k=0,1,\dots,N-1$

→

$$X[k] = G[k] + W_N^k H[k] \quad k=0,1,\dots,N-1 \quad (3)$$

- **What is this  $W_N^2$ ?** Let us just take a quick look at this term in graph **A**. Say we have again  $N = 6$ . This is like the point  $W_6^2$  shown in **A**.  $W_6^2$  and  $W_6^3$ , and so on, are also shown.
- If we were to square  $W_6$ , we would get  $W_6^2$  which is the same as  $W_3$ . So,  $W_3 = W_6^2$  and they are the same point in **A**. Also,  $W_3^2 = W_6^4$  and so on.
- So, when we are looking at every other one of the terms, shown at the end of **red** lines, we are actually looking at a length-3 DFT instead. Now, if we look at **(\*)** in **(1)**, what we are going to do is, everywhere that we see  $W_N^2$ , we can turn it into  $W_{N/2}$ . By this substitution, we arrive at **(2)**.
- Equation **(2)** looks exactly like two shorter DFT's. We can actually write **(2)** in a slightly more compact way, **(3)**. Here, for example,  $G[k]$  is the formula for the DFT of  $x[2r]$ .



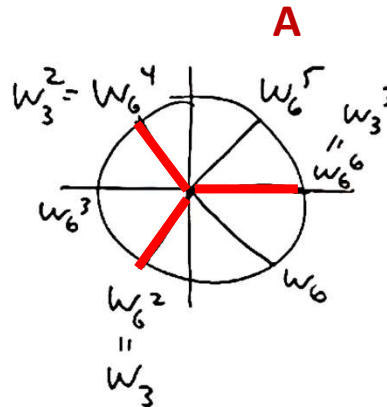
## The DIT formula

$$X[k] = \sum_{r=0}^{N/2-1} x[2r] (W_N^2)^{rk} + W_N^k \sum_{r=0}^{N/2-1} x[2r+1] (W_N^2)^{rk} \quad (1)$$

(\*)

**B**

$$\left\{ \begin{array}{l} N=6 \\ G[0] = G[3] \\ G[1] = G[4] \\ G[2] = G[5] \end{array} \right\}$$



$$= \underbrace{\sum_{r=0}^{N/2-1} x[2r] W_{N/2}^{rk}}_{\text{LENGTH } N/2 \text{ DFT OF EVEN ENTRIES}} + W_N^k \underbrace{\sum_{r=0}^{N/2-1} x[2r+1] W_{N/2}^{rk}}_{\text{LENGTH } N/2 \text{ DFT OF ODD ENTRIES}} \quad (2)$$

(\*\*)      (\*\*\*)

$G[k]$        $W_N^k H[k] \quad k=0,1,\dots,N-1$

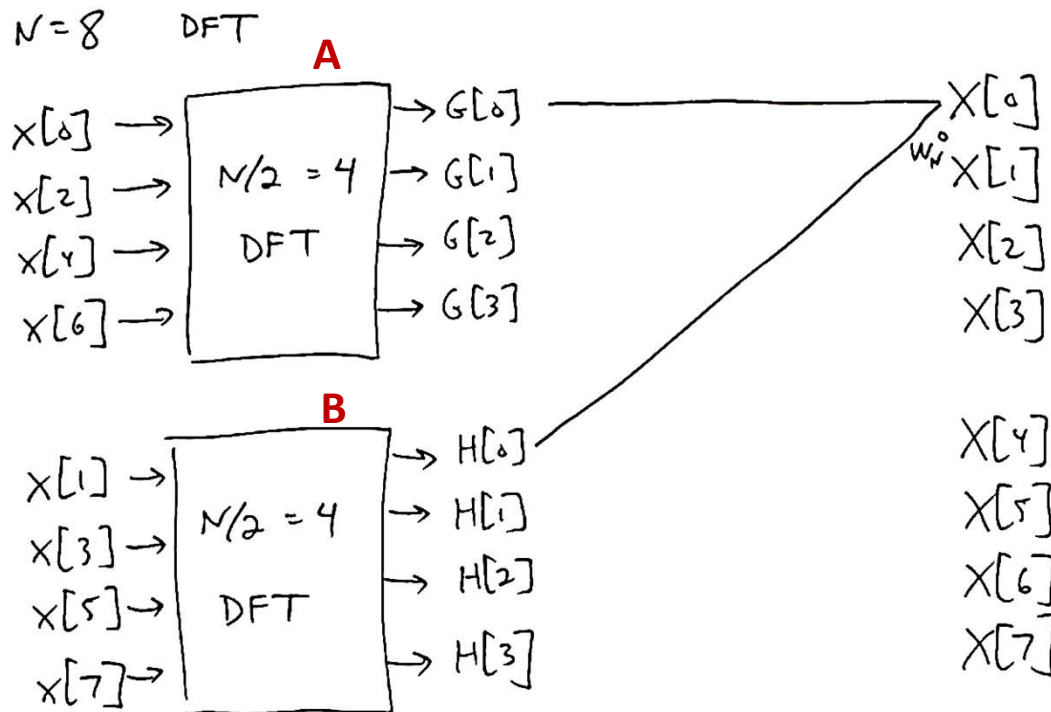
→

$$X[k] = G[k] + W_N^k H[k] \quad k=0,1,\dots,N-1 \quad (3)$$

- **Conclusion:** So, the idea here is that to compute, say a length-6 DFT, we can instead do a length-3 DFT,  $G[k]$ , and another length-3 DFT,  $H[k]$ , and then combine them together.
- Keep in mind that even though we are doing two shorter DFT's here, we get the entire double length DFT. This is done by multiplying through by the Factor  $W_N^k$ , as  $k$  ranges from 0 to  $N-1$ . This is shown in (3).
- Another thing to keep in mind is that, for example, for  $N = 6$ , this is like saying for  $G[k]$  we have  $G[0]$ ,  $G[1]$ , and  $G[2]$ . Here, and as shown in B, we will have 3 unique entries. And when it comes back to looking for  $X[3]$  in (3),  $G[0]$  wraps around  $G[3]$ , (i.e.,  $G[0] = G[3]$ ),  $G[1]$  wraps around  $G[4]$ , and so on. That is, they are equal to each other. As a result, when we need to call a higher entry of  $k$  than we have in our short DFT, we just wrap them around because we know this is a periodic system.

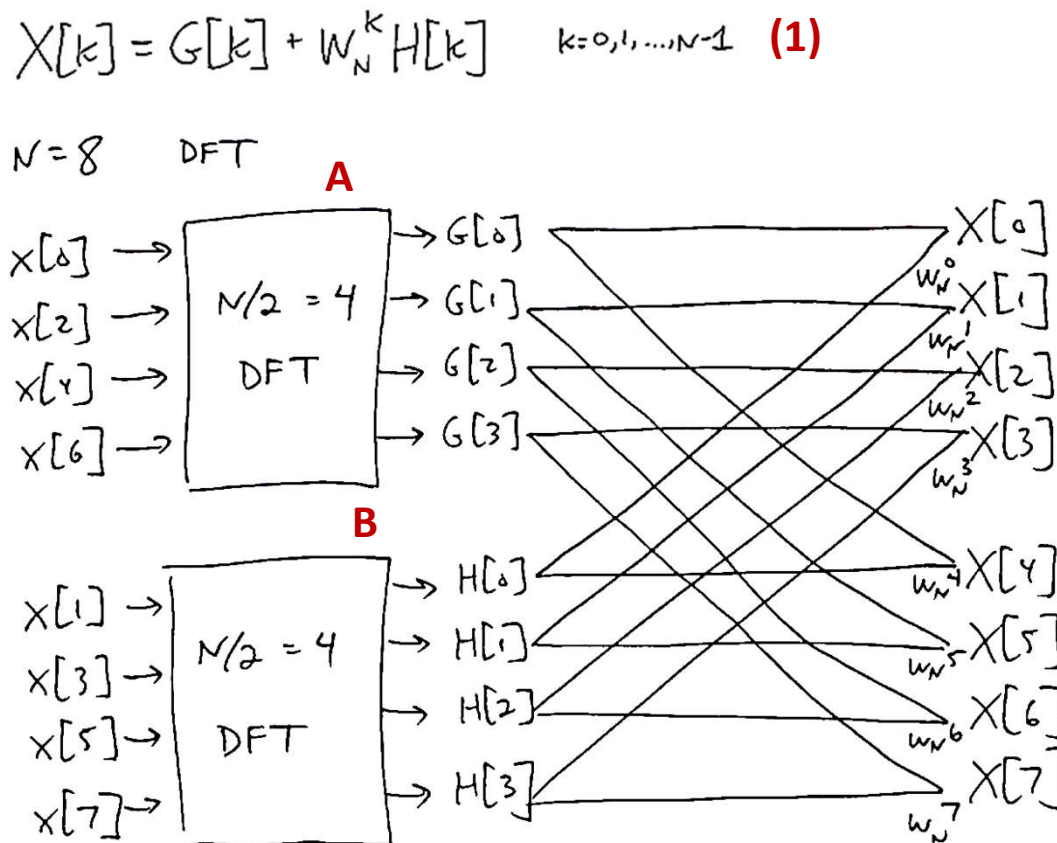
## Example with $N = 8$ : block diagram

$$X[k] = G[k] + W_N^k H[k] \quad k=0,1,\dots,N-1 \quad (1)$$



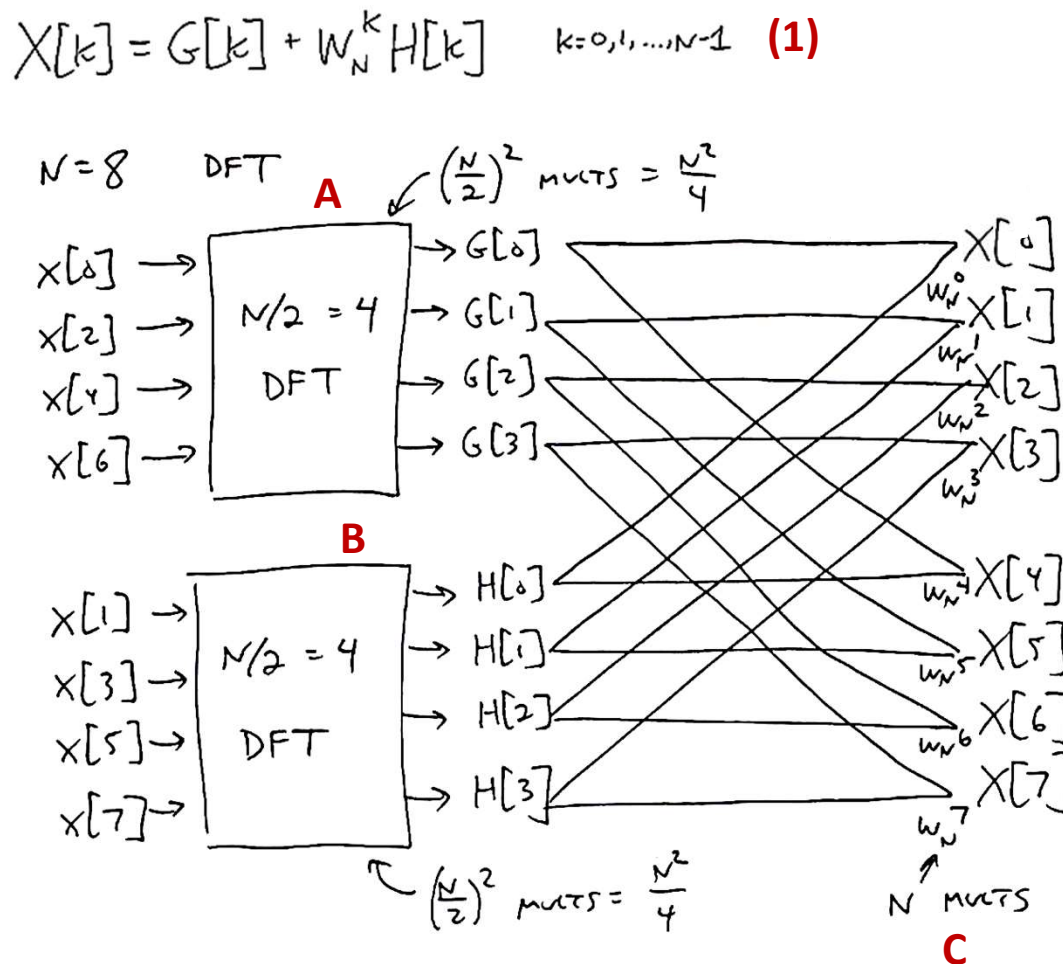
- Let us draw a picture so that we can explain (1) more graphically. Let us say we want to compute an 8 point DFT, i.e.,  $X[0]$  to  $X[7]$ . We take the even entries of  $x[n]$  and we put them through a length- $N/2$  DFT (shown in box **A**), which equals 4 DFT. Then we take the odd entries of  $x[n]$  and put them through another length- $N/2$  DFT (shown in box **B**), which also equals 4 DFT. What we get out are 4 numbers from **A**,  $G[0]$  to  $G[3]$ , and 4 numbers from **B**,  $H[0]$  to  $H[3]$ . Eventually, what we want to get out are the final DFT values, i.e., the  $X[k]$  values, ranging from  $X[0]$  to  $X[7]$ .
- For example, let us see what formula (1) tells us about how to get  $X[0]$ . It says we take  $G[0]$  and we add it to,  $H[0]$  multiplied by  $W_N^0$ . This process is shown by lines in the diagram. We will repeat the same process for the other entries of  $x[n]$  and we will fill in the **top half** of this diagram.

## Example with $N = 8$ : block diagram



- As another example, let us see what formula (1) tells us about how to get  $X[4]$ . It says we take  $G[4]$  and we add it to,  $H[4]$  multiplied by  $W_N^4$ . That is,  $X[4] = G[4] + W_N^4 \cdot H[4]$ . But, since  $G[k]$  and  $H[k]$  are length-4 DFT, we can show that  $G[4] = G[0]$  and  $H[4] = H[0]$ . So,  $X[4] = G[0] + W_N^4 \cdot H[0]$ . This process is also shown by lines in the diagram. We will repeat the same process for the other entries of  $x[n]$  and we will fill in the **bottom half** of this diagram. This completes a schematic picture of what is happening.
- Conclusion:** In summary, we put the even and the odd signals through shorter DFT's and we then combine those DFT's in a specific way to get our final output.

## Completed block diagram (first stage)

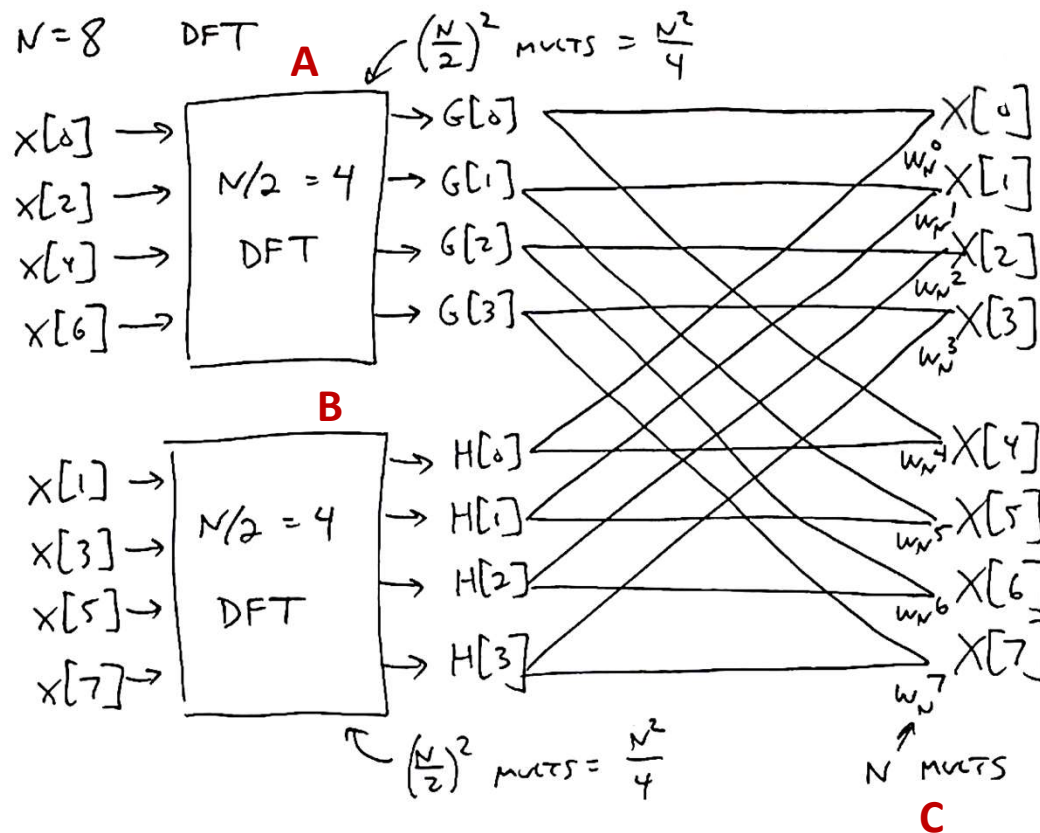


- **What do we gain by doing this?** For an  $N$  DFT, we normally would need  $N^2$  **multiplications**. How many **multiplications** do we need for this new process? In box **A**, when doing this DFT, we have  $(N/2)^2$  multiplications (abbreviated as “**MULTS**”). In box **B**, we have another  $(N/2)^2$  multiplications. At **C**, we have  $N$  extra multiplications that have to happen to combine the things to get the output.
- **What do we have in the end?** In box **A**,  $(N/2)^2$  **MULTS** leads to  $N^2/4$  **MULTS** and in box **B**, we will have the same, i.e.,  $N^2/4$  **MULTS**. So, altogether, we have the total as: **MULTS** =  $N^2/4 + N^2/4 + N \approx N^2/2$ .
- **Conclusion:** By writing things in the described way, we can reduce our multiplications by half.

→ **total**  $\approx \frac{N^2}{2}$  **MULTS**

## Going down another level

$$X[k] = G[k] + W_N^k H[k] \quad k=0,1,\dots,N-1 \quad (1)$$

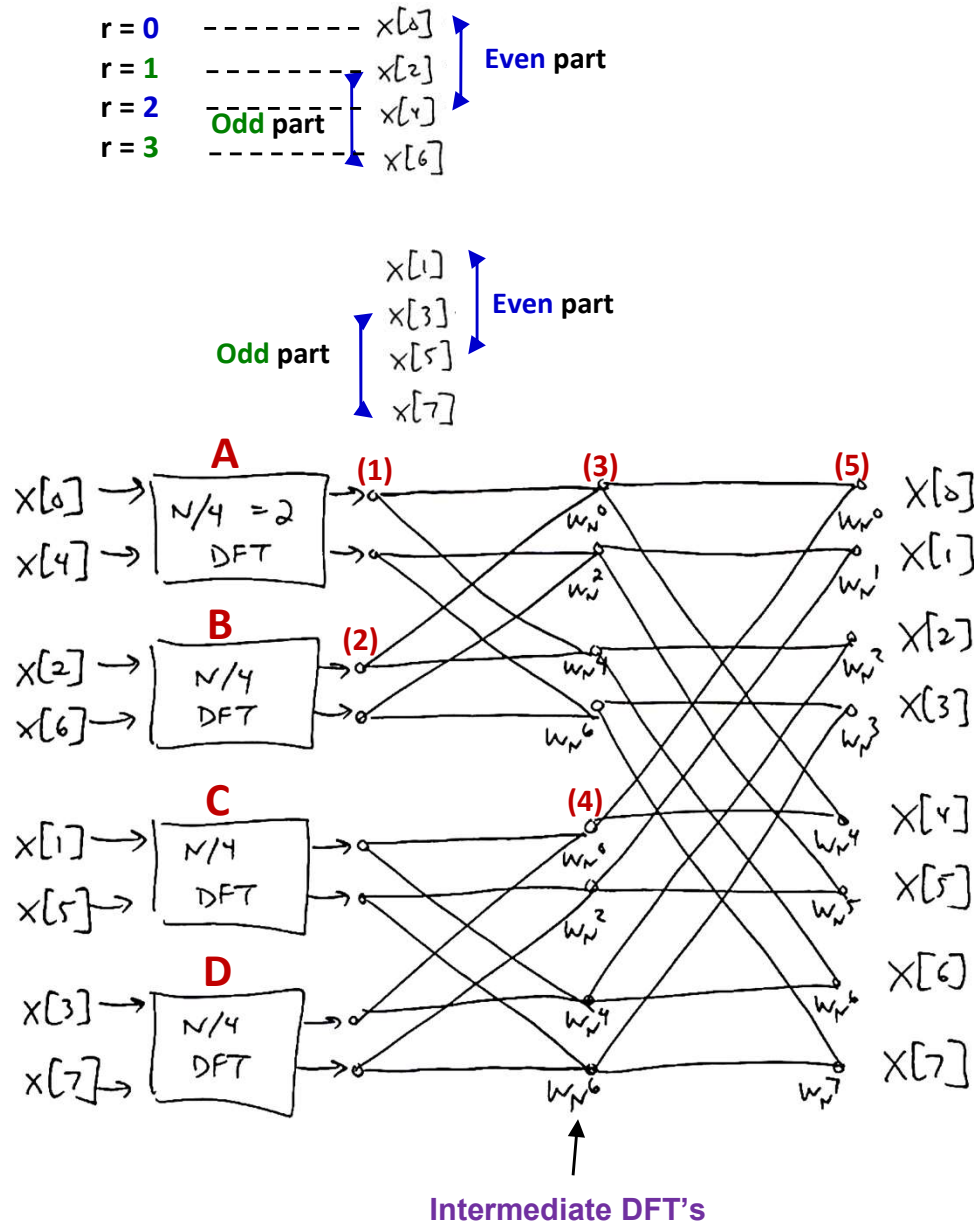


- The method is already pretty good. But now, we think why would we stop there! Box **A** is an even number or an even length DFT of 4,  $N/2 = 4$ . So, again in this case, **why don't we just redo the process for these shorter DFT's?**
- For example, we could turn the block of **A** into 2 length-2 DFT's. We can now rewrite this block by breaking it down further. So, let us now see what would happen in the top half.

→ **total**  $\approx \frac{N^2}{2} \text{ mults}$



## Completed block diagram (second stage)



- In boxes **A** and **B**, this would be like saying we would have  $x[0]$  and  $x[4]$ , as the even parts, and  $x[2]$  and  $x[6]$  as the odd parts. Here, we can say that  $x[0]$  to  $x[6]$  happen from  $r = 0$  to  $r = 3$ . So, for example,  $x[4]$  will happen at  $r = 2$ , which is an **even** number. In boxes **C** and **D**,  $x[1]$  and  $x[5]$  are the even parts, and  $x[3]$  and  $x[7]$  are the **odd** parts.
- We are now **reordering the input** in a slightly different way. For example, after we reorder the inputs of  $x[0]$  to  $x[6]$ , we put  $x[0]$  and  $x[4]$  through block **A**, with a length- $N/4$  (or 2) DFT, and  $x[2]$  and  $x[6]$  through block **B**, with the same DFT length of 2. We can also draw little dots here to remind ourself that some nodes here are to be used later.
- Next, we are going to combine **4** inputs, i.e.,  $x[0]$ ,  $x[4]$ ,  $x[2]$ , and  $x[6]$ , in order to get **4** outputs,  $X[0]$ ,  $X[1]$ ,  $X[2]$ , and  $X[3]$ . For example, to get **(3)**, we combine **(1)** and **(2)** and then we multiplied them by  $w_N^0$ . After completing these, we will have our **intermediate DFT's**.
- Now we have to combine these intermediate DFT's to get our final  $X[k]$ 's. For example, we need to combine **(3)** and **(4)**, to get **(5)**, which is basically our  $X[0]$ . Here, we are using a slightly different diagram, which is kind of duplicating the diagram we had before. Then again, this makes our life a little bit easier here!

## Going down to length-2 DFTs

LENGTH 2 DFT:

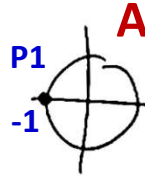
$$x[0], x[1] \rightarrow X[0], X[1]$$

$$X[k] = \sum_{n=0}^1 x[n] W_2^{nk} \quad (1)$$

$$= \sum_{n=0}^1 x[n] (-1)^{nk} \quad (2)$$

$$X[0] = x[0] + x[1] \quad (3)$$

$$X[1] = x[0] - x[1] \quad (4)$$

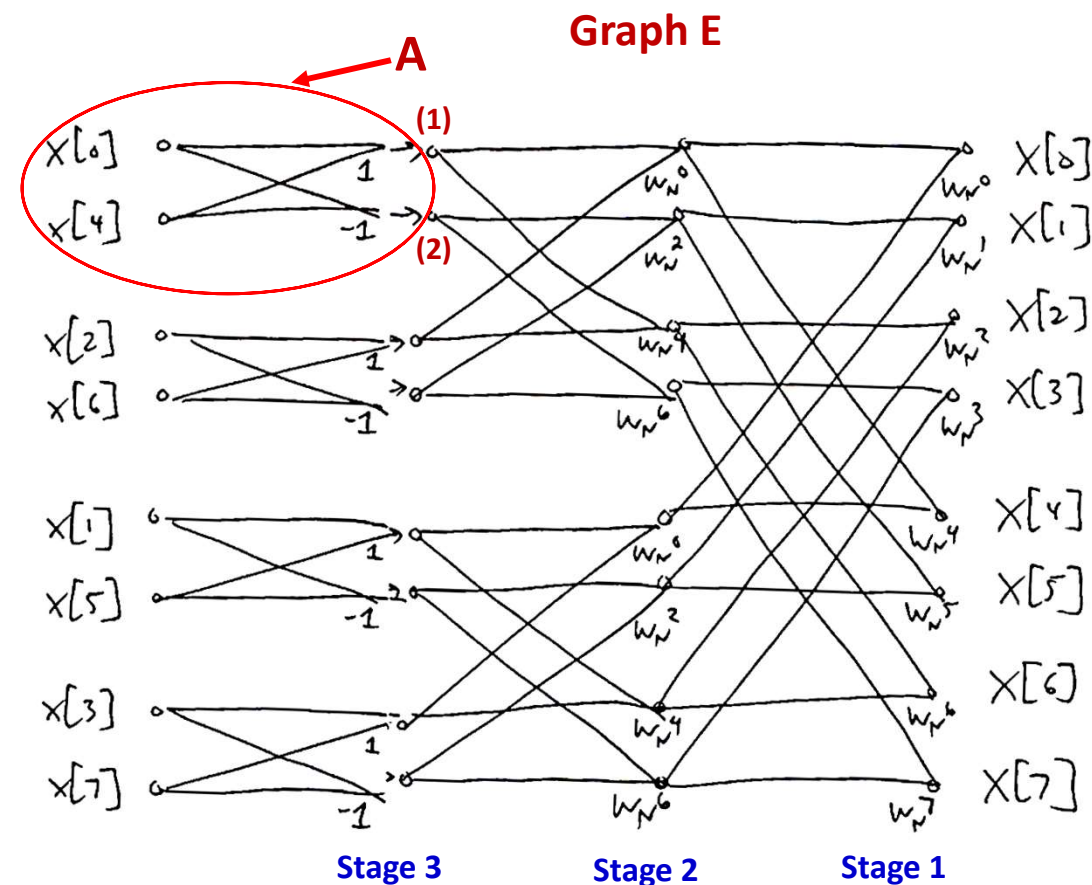


- At this point, this is about as far as we can go. Let us now focus on what exactly this length-2 DFT is.
- Let us write the equation for  $X[k]$ . Our formula is shown in (1). Here,  $W_2$  is like -1, i.e., it is always P1 (or -1) in A. This simplifies (1) to (2). This is actually a very easy thing to write. If we write this out, this is like saying that  $X[0]$  is just adding these two entries, and  $X[1]$  is like subtracting them, as shown in (3) and (4). There is no multiplication here at all!

From before:

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

## Completed block diagram (all stages)

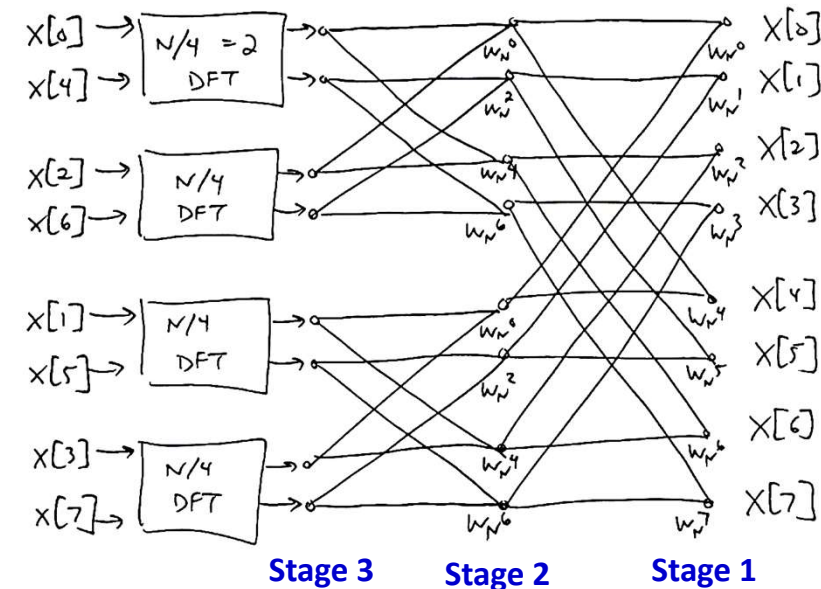


- So, all we need to do to get to nodes **(1)** and **(2)** in **A**, is very simple operations. Here, all we are doing is multiplying  $x[4]$  by **+1**, and multiplying  $x[0]$  by **-1**. We see that these smallest DFT's are actually really simple. At the end of the day, length-**8** DFT can be easily decomposed like this.
- **During this process, how many complex multiplies,  $W_N^k$ , are there?** Not very many. We need to see how many stages of decomposition we are doing. Suppose that **N** is a power of **2**. In that case, we can recursively break the DFT into  **$\log_2 N$  stages**.
- In our example, **N** was **8** or  **$N = 2^3$** . So,  **$\log_2 2^3 = 3$**  stages. By referring to **Graph E**, we can clearly see these **3 stages**.



## The final computational cost is $O(N \log_2 N)$

Graph E



IF  $N$  IS A POWER OF 2, WE  
CAN RECURSIVELY BREAK THE DFT INTO  
 $\log_2 N$  STAGES.

→  $N \log_2 N$  MULTIPLIES

$$N = 2^{10} = 1024$$

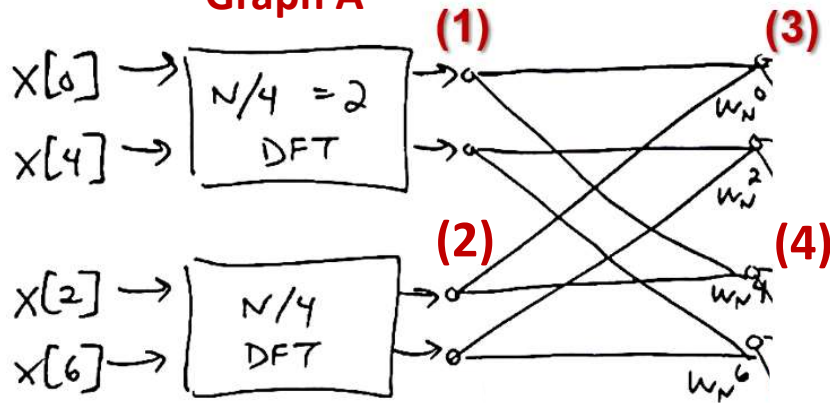
$$N^2 \approx 1000000 \text{ vs.}$$

$$N \log_2 N = 10240. \quad \times 100$$

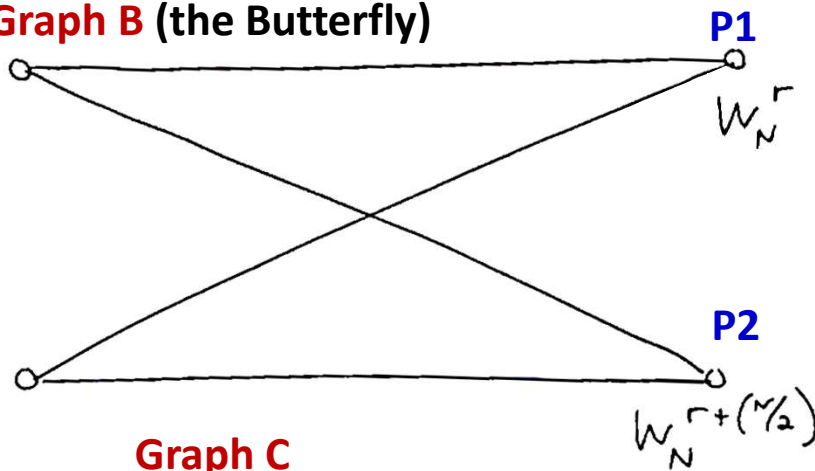
- Now, let us compute the **total amount of computation**. At every stage, all we are doing is multiplying by these complex numbers,  $W_N^k$ . We have  **$N$  multiplies** to do at every stage, so that means that the total will be  **$N \log_2 N$  multiplies**. This is where the computational savings comes in.
- Let us say instead of  $N = 2^3$ , we have  $N = 2^{10}$ , which is **1,024** or approximately **1000**. Here, we clearly see the difference between  **$N^2$  multiplies** for the routine matrix operation, which is **1,048,576** or approximately **1,000,000**, versus  **$N \log_2 N$  multiplies**, which is equal to **10,240** or approximately **10,000**. This is a speed of **100** times. That is why we wanted to do FFT this way.
- Conclusion:** If we just use routine matrix operations, then we would be losing out on all the computational savings that we could get.

## The "butterfly"

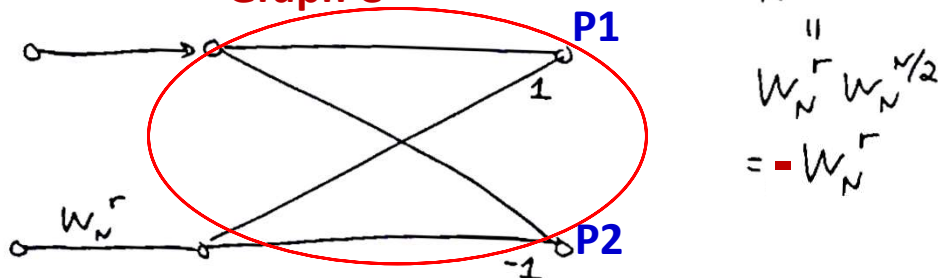
Graph A



Graph B (the Butterfly)



Graph C



- We can still do a little bit better! We observe that in **Graph A**, at each of these stages, there is a pattern. We use (1) and (2), to get (3) and (4).
- So, we see there is a **fundamental unit (the butterfly)**, shown in **Graph B**, where we have four elements, two to the left and two to the right. And, we are always combining the same two in this way. At node **P1**, we have some power of  $r$ , let us call it  $W_N^r$  and at node **P2**, we have  $W_N^{r+(N/2)}$ .
- Let us confirm this by an example. If we compare node (3) with node (4), they are separated by  $N/2 = 8/2 = 4$  or  $W_N^{(8/2)} = W_N^4$ . The same pattern applies to other corresponding nodes. These are always separated by 4. That is,  $W_N^r$  vs.  $W_N^{r+(N/2)}$  (or  $W_N^{r+4}$ ) or  $W_N^r$  vs.  $W_N^{r+4}$ .
- Given that  $W_N^{N/2}$  is always -1, this is like saying we have  $W_N^r$  multiplied by -1. Now, what we are doing is just multiplying **P1** by  $(+1) \cdot W_N^r$  and multiplying **P2** by  $(-1) \cdot W_N^r$ .
- As shown in **Graph C**, we can also go one step further and make this easier by saying that before we do the operation shown in the red oval, we multiply the node in the bottom by  $W_N^r$  and then all we have to do is add it at **P1** and subtract it at **P2**. That means now we actually have half fewer computations than before. It can be shown that there are some other advantages in using method, such as computer storage.

## Matrix interpretation of decimation in time

matrix  $W_N^{kn}$

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \\ X[4] \\ X[5] \\ X[6] \\ X[7] \end{bmatrix} = \begin{bmatrix} \text{?} \\ \text{?} \\ \text{?} \\ \text{?} \\ \text{?} \\ \text{?} \\ \text{?} \\ \text{?} \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \\ x[4] \\ x[5] \\ x[6] \\ x[7] \end{bmatrix}$$

- Let us see what the matrix interpretation of decimation in time is. First, we need to compute the elements of matrix  $W_N^{kn}$ . To do that, we go back to our original formula for the DFT, **(1)**. Using **(2)**, we can compute the elements of by varying  $k$  and  $n$ .

$$\text{DFT: } X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn} \quad \text{(1)} \quad \begin{matrix} n = 0, 1, \dots, N-1 \\ k = 0, 1, \dots, N-1 \end{matrix}$$

$$W_N = e^{-j\frac{2\pi}{N}} \quad \text{(2)}$$

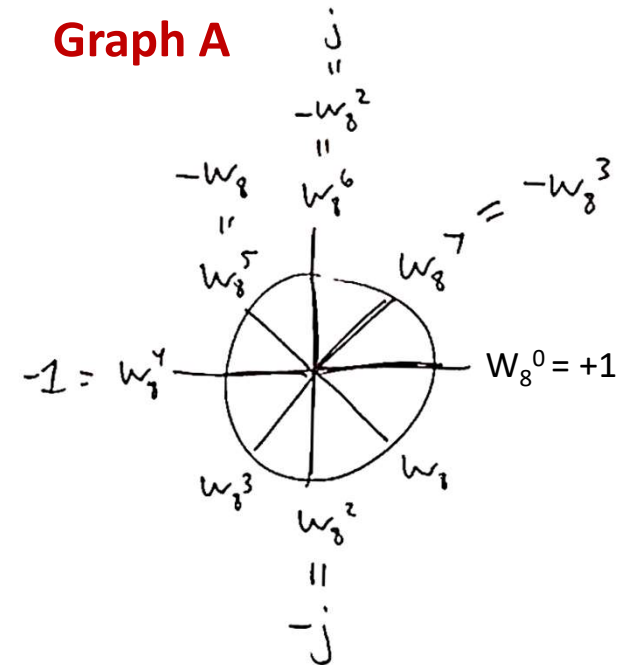
## Matrix interpretation of decimation in time

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \\ X[4] \\ X[5] \\ X[6] \\ X[7] \end{bmatrix} = \begin{bmatrix} k=0 \rightarrow 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ k=1 \rightarrow 1 & W_8 & W_8^2 & W_8^3 & -1 & -W_8 & -W_8^2 & -W_8^3 \\ k=2 \rightarrow 1 & -j & -1 & j & 1 & -j & -1 & j \\ & 1 & W_8^3 & j & W_8 & -1 & -W_8^3 & -j & -W_8 \\ & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ & 1 & -W_8 & W_8^2 & -W_8^3 & -1 & W_8 & -W_8^2 & W_8^3 \\ & 1 & -W_8^2 & -1 & W_8^2 & 1 & -W_8^2 & -1 & W_8^2 \\ k=7 \rightarrow 1 & -W_8^3 & -W_8^2 & -W_8 & 1 & W_8^3 & W_8^2 & W_8 \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \\ x[4] \\ x[5] \\ x[6] \\ x[7] \end{bmatrix}$$

$$\text{DFT: } X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn} \quad (1) \quad \begin{matrix} n = 0, 1, \dots, N-1 \\ k = 0, 1, \dots, N-1 \end{matrix}$$

$$W_N = e^{-j\frac{2\pi}{N}} \quad (2)$$

Graph A



- For instance, for the third row, we have  $k = 2$ , and the  $W_N$ -terms can be computed using  $W_N^{kn} = W_8^{2 \times n}$ . Here,  $n = 0, 1, \dots, 7$ . For  $n = 0$ , we have  $W_8^{2 \times 0} = W_8^0 = +1$ . For  $n = 1$ ,  $W_8^{2 \times 1} = W_8^2 = -j$ , etc. We can use **Graph A** for all the  $W_N$ -terms.

## Matrix interpretation of decimation in time

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \\ X[4] \\ X[5] \\ X[6] \\ X[7] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & w_8 & w_8^2 & w_8^3 & -1 & -w_8 & -w_8^2 & -w_8^3 \\ 1 & -j & -1 & j & 1 & -j & -1 & j \\ 1 & w_8^3 & j & w_8 & -1 & -w_8^3 & -j & -w_8 \\ \hline 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -w_8 & w_8^2 & -w_8^3 & -1 & w_8 & -w_8^2 & w_8^3 \\ 1 & -w_8^2 & -1 & w_8^2 & 1 & -w_8^2 & -1 & w_8^2 \\ 1 & -w_8^3 & -w_8^3 & -w_8 & 1 & w_8^3 & w_8^2 & w_8 \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \\ x[4] \\ x[5] \\ x[6] \\ x[7] \end{bmatrix}$$

(1) (3)  
(2) (4)  
**Even columns**

- When we examine this matrix more closely, we start to observe some *symmetries*.
- Let us first look at the even columns. Also, let us draw a horizontal line in the middle (the **blue** line). After we split the matrix up by columns, we see that in the **even columns** (the **red** boxes), the top half and the bottom half are the same. An example is the top half, **(1)**, and the bottom half, **(2)**, of the second even column from the left.
- In the **odd columns** (the **green** boxes), such as in the column of **(3)** and **(4)**, the top and bottom halves are the same but with **the opposite signs**.
- So, what does this mean? This is like saying that we can write this whole matrix in a slightly different way.

## Matrix interpretation of decimation in time

Even columns =

$$= \begin{bmatrix} I_{4 \times 4} \\ I_{4 \times 4} \end{bmatrix} \begin{matrix} \nearrow \\ 8 \times 4 \end{matrix} \begin{matrix} (1) \\ \text{DFT for } N=4 \end{matrix}$$

$$= \begin{bmatrix} I_{4 \times 4} \\ I_{4 \times 4} \end{bmatrix} \begin{matrix} (2) \\ F_4 \end{matrix}$$

Matrix (1) is an 8x4 matrix:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W_8^2 & -1 & -W_8^2 \\ 1 & -1 & 1 & -1 \\ 1 & -W_8^2 & -1 & W_8^2 \end{bmatrix}$$

Matrix (2) is a 4x4 matrix:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W_4 & W_4^2 & W_4^3 \\ 1 & -1 & 1 & -1 \\ 1 & -W_4 & -1 & W_4 \end{bmatrix}$$

- Let us write the **decomposition of the DFT** in a slightly different way. It is like saying if we want to get the outputs,  $X[0]$  to  $X[7]$ , we could rearrange the matrix of  $W_N^{kn}$ .
- Let us first look at the even entry matrix. The even columns matrix is like a **4x4** identity matrix,  $I_{4 \times 4}$ . So, this is like an **8x4** matrix that is multiplied by matrix **(1)**. Note that, e.g.,  $W_8^2$  can be simplified to be equal to  $W_4^1$  or just  $W_4$ , and so on.
- We see that matrix **(2)** is exactly the same as the DFT matrix for  $N = 4$ . That is, this is the decomposition of the matrix into the smaller DFT. We show this matrix by  $F_4$ . This is like saying that to get the even parts we need the **4** DFT matrix.



$F_8$  in terms of  $F_4$ 

ODD COLUMNS =

$$= \begin{matrix} \text{(1)} \\ \left[ \begin{array}{c} I_{4 \times 4} \\ -I_{4 \times 4} \end{array} \right] \end{matrix} \begin{matrix} \text{(2)} \\ \left[ \begin{array}{cccc} 1 & 1 & 1 & 1 \\ w_8 & w_8^3 & -w_8 & -w_8^3 \\ w_8^2 & -w_8^2 & w_8^2 & -w_8^2 \\ w_8^3 & w_8 & -w_8^3 & -w_8 \end{array} \right] \end{matrix}$$

$$= \begin{matrix} \left[ \begin{array}{c} I_{4 \times 4} \\ -I_{4 \times 4} \end{array} \right] \end{matrix} \begin{matrix} \left[ \begin{array}{cccc} 1 & & & \\ & w_8 & & \\ & & w_8^2 & \\ & & & w_8^3 \end{array} \right] \end{matrix} \begin{matrix} \left[ \begin{array}{cccc} 1 & 1 & 1 & 1 \\ 1 & w_8^2 & -1 & -w_8^2 \\ 1 & -1 & 1 & -1 \\ 1 & -w_8^2 & -1 & w_8^2 \end{array} \right] \end{matrix}$$

diagonal matrix  $F_4$

- Now, let us look at the odd columns. In (1), we have the top part as  $+I_{4 \times 4}$ , and  $-I_{4 \times 4}$  as the bottom part. In (2), we can factor out a **diagonal matrix** where all the elements are 0 except for possibly the elements that are on the main diagonal. What is left, the third matrix, is the same Fourier matrix we had before,  $F_4$ .

## Twiddle factors

$$F_8 = \begin{bmatrix} I & I \\ I & -I \end{bmatrix} \begin{bmatrix} I & \boxed{\begin{bmatrix} 1 & w_8 & w_8^2 & w_8^3 \end{bmatrix}} \\ F_4 & F_4 \end{bmatrix} \begin{bmatrix} x_{\text{even}} \\ x_{\text{odd}} \end{bmatrix}$$

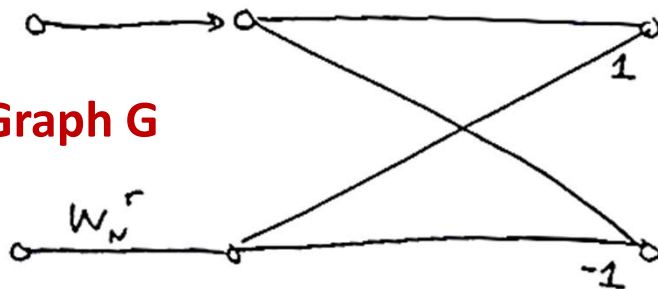
(1)                      (2)                      (3)                      (4)

↑                      ↑                      ↑

SUMS AND DIFFERENCES IN "BUTTERFLY"      "TWIDDLE FACTORS"      LENGTH 4-DFTS

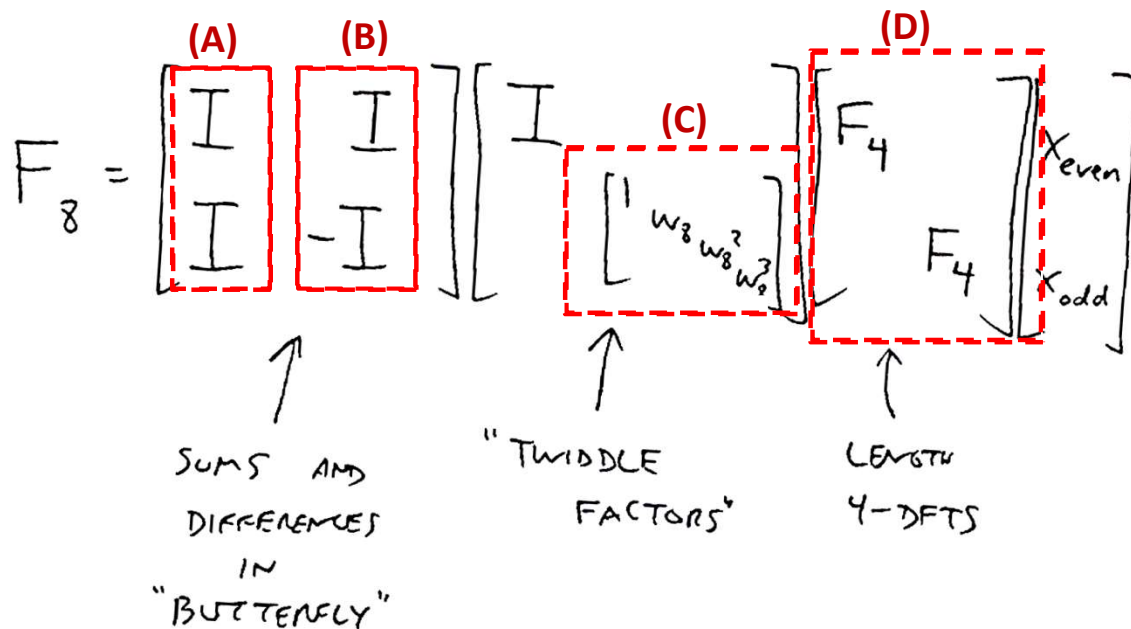
- Now, let us put everything together. To compute, the  $8 \times 8$  Fourier matrix,  $F_8$ , is like taking **identity matrices (1)**, with the even and odd columns, times **(2)**, which has one of the identity matrix  $I$ , and the other part, which is a matrix of vectors, called "**Twiddle Factors**". Then, we multiply by **2** length-4 Fourier transforms, **(3)**, and then finally we multiply by the matrix of the even entries and the odd entries, **(4)**.
- So, we are exposing the fact that **(3)** are the length-4 DFT's, the matrix of vectors inside **(2)** are the twiddle factors, and **(1)** are basically the sums and the differences in **butterfly**, as shown in **Graph G**.

**Graph G**

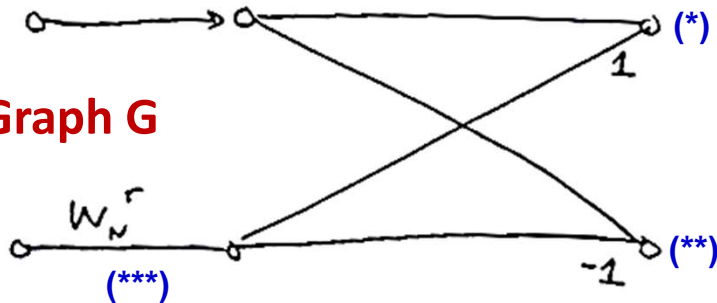




## Twiddle factors



**Graph G**



- As shown in **Graph G**, for every butterfly, we have a sum  $(*)$ , a difference  $(**)$ , and a multiplication by a complex number  $W_N^r$ ,  $(***)$ .
- In  $F_8$  formula, **(A)** is the sums, **(B)** is the differences, **(C)** is the multiplication, and **(D)** is the length-4 DFT's.
- If we wanted to, we could further decompose **(D)**, into length-2 DFT's with additional twiddle factors.
- Conclusion:** This whole method is called a **decimation in time** strategy. The reason for this nomenclature is that we are taking the time-domain signal and chunking it up into different pieces. This is like we are keeping on separating them out into smaller and smaller sub-pieces.

## Decimation in frequency

DECIMATION IN FREQUENCY FFT

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk} \quad (1)$$

EVEN SAMPLES OF  $X[k]$ :  $X[2r] \quad r = 0, 1, \dots, \frac{N-1}{2}$

$$X[2r] = \sum_{n=0}^{N-1} x[n] W_N^{2nr} \quad (2)$$

$$= \underbrace{\sum_{n=0}^{\frac{N}{2}-1} x[n] W_N^{2nr}}_A + \underbrace{\sum_{n=\frac{N}{2}}^{N-1} x[n] W_N^{2nr}}_B \quad (3)$$

- There is also another algorithm called **decimation in frequency FFT** or **DIF FFT**. That means instead of keeping the output order fixed and shuffling around the input, we keep the input order fixed and **shuffle around the output**.
- One way we can think about this is that again we start with the DFT equation, (1). Now let us suppose we ask for just the even samples of this. That is like saying we have even samples of  $X[k]$ , where  $k = 2r$ . Here,  $r = 0, 1, \dots, (N-1)/2$ .
- After splitting (2) into two separate sums, we arrive at (3). Now, sum **A** looks like a shorter DFT and sum **B** can also be thought of as a shorter DFT.

## Decimation in frequency

$$X[2r] = \sum_{n=0}^{N/2-1} x[n] W_{N/2}^{nr} + \sum_{n=0}^{N/2-1} x[n + \frac{N}{2}] W_N^{2(n + \frac{N}{2})r} \quad (1)$$

$$\rightarrow X[2r] = \sum_{n=0}^{N/2-1} \left( x[n] + x[n + \frac{N}{2}] \right) W_{N/2}^{nr} \quad (2)$$

LIKE AN  $N/2$  DFT OF SUMMED INPUT  
(TOP HALF + BOTTOM HALF)

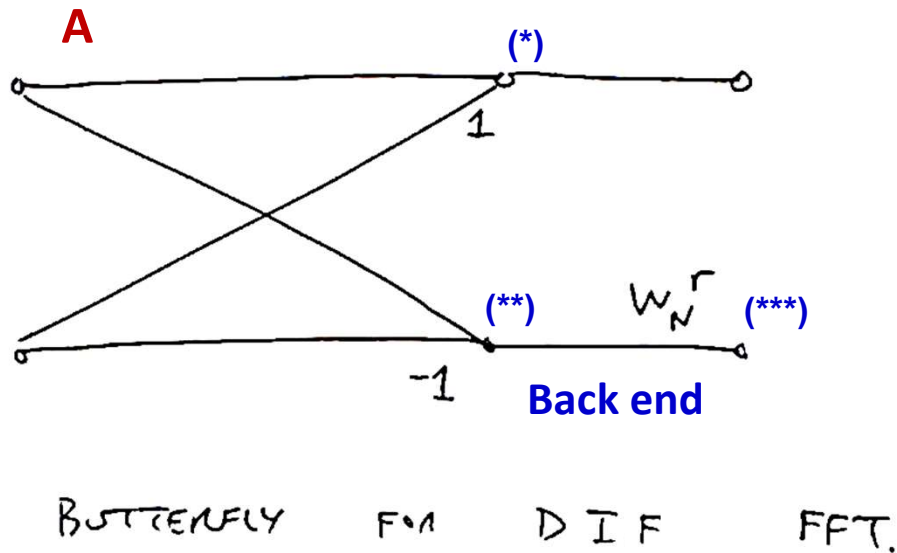
FOR ODD ENTRIES OF  $X[k]$ , CAN SHOW

$$X[2r+1] = \sum_{n=0}^{N/2-1} \left( x[n] - x[n + \frac{N}{2}] \right) \underbrace{W_N^n}_{\text{TWIDDLE}} W_{N/2}^{nr} \quad (3)$$

- After reindexing the previous equation, we arrive at (1). Further simplification leads to (2). Equation (2) means that we can get the even entries of the output,  $X[2r]$ , by adding up the top half and the bottom half of the input and then taking a short DFT of that. Put it differently, this is like an  $N/2$  DFT of the summed input, basically the top half **plus** the bottom half.
- In the same way, we can show that the odd entries of  $X[k]$ , i.e.,  $X[2r+1]$ , look like a shorter DFT applied to the **difference** between the top half and the bottom half, as shown in (3). And again, we have an extra little factor,  $W_N^n$ , the twiddle factor, that comes into play.

- Conclusion:** The above method, **decimation in frequency FFT**, is pretty similar to the decimation in time. The only thing that we are doing here differently is that we are now getting the output in terms of even entries and odd entries. Here, we can keep the input in the same order if we wanted to. This is like a “**your choice situation**” about whether you want to get the output in a weird order or to get the input in a weird order. In fact, we can show that these two algorithms are basically equivalent, i.e., DIF FFT (the decimation in frequency FFT) and DIT FFT (the decimation in time FFT). There is no computational difference either.

## Decimation in frequency



- In **decimation in frequency FFT**, or **DIF FFT**, we have a slightly different butterfly, as shown in **A**. Here, instead of having a butterfly where we do the twiddle factors on the front end, we have a butterfly that we do the twiddle factors on the **back end**.
- In the butterfly diagram, DIF FFT is like saying we add these two elements, (\*), we subtract these two elements, (\*\*), and then we multiply the result by an extra complex factor,  $W_N^r$ , (\*\*\*).
- **Conclusion:** Any decimation in time can be flipped around to be a decimation in frequency. The basic idea is that we can do DFT's easily just by a series of stages. All we are doing at each stage is adding and subtracting elements and doing this little twiddle to modify things for the next stage. This algorithm is already pretty smart and efficient.

## Decimation in frequency

LENGTH- 80 DFT  $80 = 2^4 \cdot 5$

- Next time, we will talk about an even smoother way of doing the method we just covered, where we can *eliminate the twiddle factors entirely*.
- So, what we are going to show later is that suppose we wanted to do a length-**15** DFT. One option is to turn it into a length-**16** DFT by **zero padding** and do a power of **2** DFT. Note that a length-**15** is a non-power-of-2 signal length.
- The other thing is that we can just take the length-**15** DFT and turn it into a bunch of length-**3** DFT's and length-**5** DFT's. So, basically we are factoring the dimensionality of the input. It is like a ***prime factoring algorithm***, where we say in what way we can break down the input into its most basic compliments.
- As an example, if we had a length-**80** DFT, **80** is equal to **16**×**5** or **2**<sup>4</sup>×**5**. So, the idea is that we could turn this into a whole bunch of length-**2** and length-**5** DFT's with no twiddle factors, using the method we are going to discuss in the next lecture.

# End of Lecture 11