1. **[DFT and FFT]**
You are to compute *by hand* the DFT of the following sequence:

$$\mathbf{x} = \{1, 2, 3, 4\}$$

(a) Compute the DFT of $\mathbf{x}$ using the definition of the DFT (do not use an FFT). Show your work and give exact answers.

Solution:

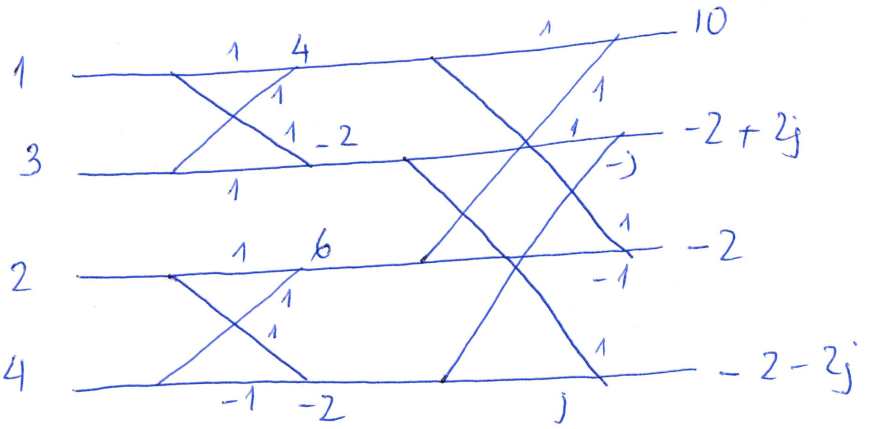
$$F[k] = \sum_{n=0}^{N-1} x_n e^{-j\frac{2\pi}{N}nk}$$

$$= 1 + 2(-j)^k + 3(-1)^k + 4(j)^k$$

$$= \{10, -2+2j, -2, -2-2j\}.$$

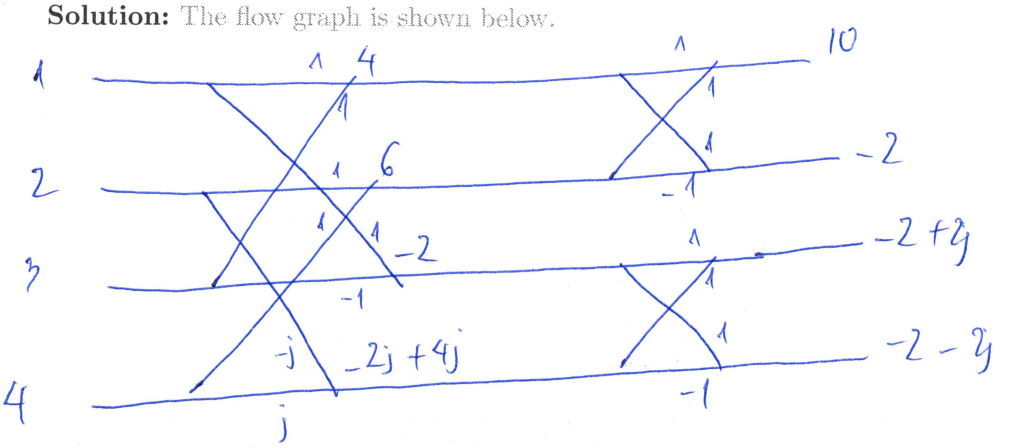
(b) Use a decimation-in-time radix-2 FFT to compute the DFT of $\mathbf{x}$. Neatly draw and label the flow graph used to compute your answer.

Solution: The flow graph is shown below.

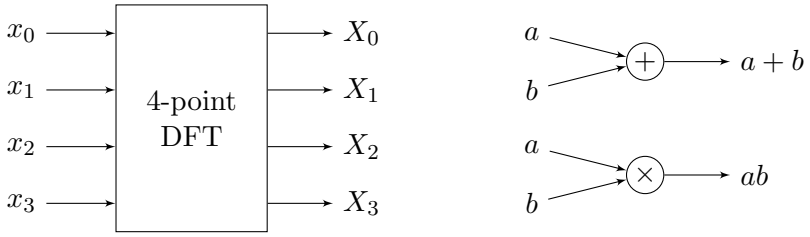


(c) Repeat part (b) using decimation-in-frequency.

Solution: The flow graph is shown below.



2. **[FFT Architecture]**
Suppose that you wish to design a circuit to perform a length-8 DFT. You are given a pair of chips that compute the DFT of a length-4 complex input sequence. The inputs and outputs of this chip are all complex numbers. You also have access to complex multiplication and addition circuits, each of which has two complex inputs and one complex output.



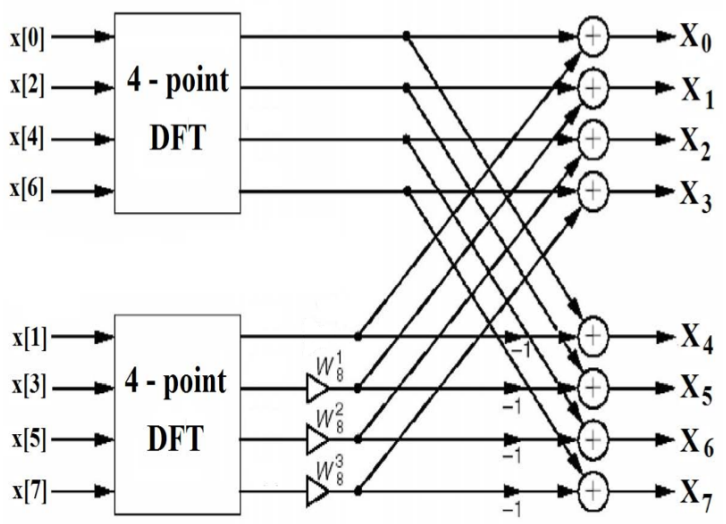
(a) Your goal is to use as few complex multiplication circuits as possible. Fortunately, you don't need to use a multiplier circuit to multiply by $+1$, -1, $+j$, or $-j$. Explain why these are trivial multiplications.

Solution: Assume the complex numbers are represented as a pair of signed integers. To multiply by 1, do nothing. To multiply by -1, invert the sign bits. To multiply by $\pm j$, swap the real and imaginary parts (connect the real output to the imaginary input and the imaginary output to the real input) and flip the sign bits as necessary. These operations require at most an inverter, which is negligible compared to a multiplier circuit.

(b) Show how you would connect two chips and the multiplication and addition circuits to compute a length-8 decimation-in-time FFT. Explain your solution. How many nontrivial multiplications are required?

Solution:

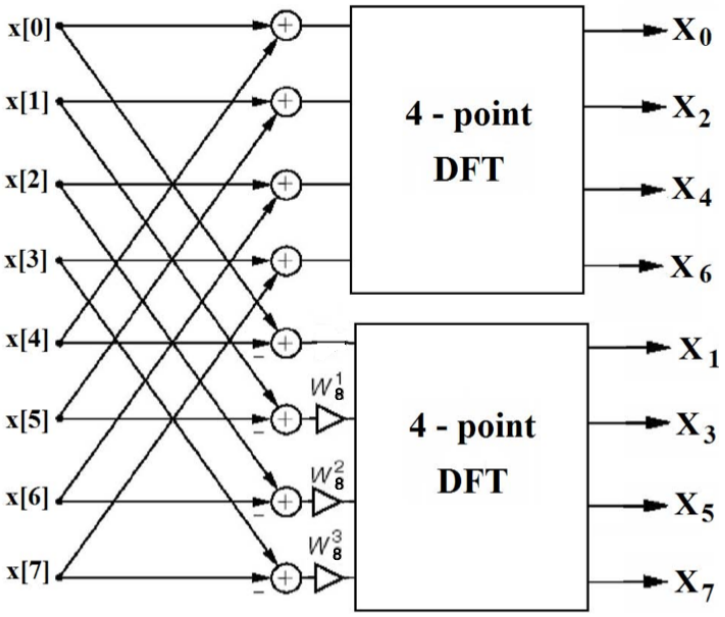
The following diagram shows how to wire the two DFT chips for decimation-in-time. In this figure the triangle represents complex multiplication by the factor adjacent to it. The weights in the top half are the negatives of the weights in the bottom half, so we can reduce the number of complex multiplies required by applying the weights to the the bottom four DFT chip outputs once and then negating them as necessary. $W_8^0 = 1$ and $W_8^2 = -j$ are both trivial multiplications, so there are a total of two non-trivial complex multiplications.



(c) Repeat part (b) using decimation-in-frequency.

Solution:

The following diagram shows how to wire the two DFT chips for decimation-in-frequency. In this figure the triangle represents complex multiplication by the factor adjacent to it. As before, we can reduce the number of complex multiplications by factoring out the weights on the bottom half. Once again, $W_8^0 = 1$ and $W_8^2 = -j$ are both trivial multiplications, so there are a total of two non-trivial complex multiplications.



3. **[Radix-3 FFT]**
In this problem, you will derive a radix-3 decimation-in-frequency FFT. Consider a length-N sequence $x[0], \ldots, x[N-1]$, where N is a power of 3. Denote the DFT by $X[0], \ldots, X[N-1]$.

(a) Derive an expression for $X[3r]$, that is, for $X[0], X[3], X[6], \ldots$, as a $\frac{N}{3}$-point DFT of time-aliased samples.

Solution:

$$\begin{aligned}
X[3r] &= \sum_{n=0}^{N-1} x[n] W_N^{3nr} \\
&= \sum_{n=0}^{N-1} x[n] W_{N/3}^{nr} \\
&= \sum_{n=0}^{N/3-1} x[n] W_{N/3}^{nr} + \sum_{n=N/3}^{2N/3-1} x[n] W_{N/3}^{nr} + \sum_{n=2N/3}^{N-1} x[n] W_{N/3}^{nr} \\
&= \sum_{n=0}^{N/3-1} \left(x[n] + x[n+\frac{N}{3}] + x[n+\frac{2N}{3}] \right) W_{N/3}^{nr}
\end{aligned}$$

(b) Derive an expression for $X[3r+1]$, that is, for $X[1], X[4], X[7], \ldots$, as a $\frac{N}{3}$-point DFT of aliased and modulated samples. Repeat for the remaining outputs, $X[3r+2]$.

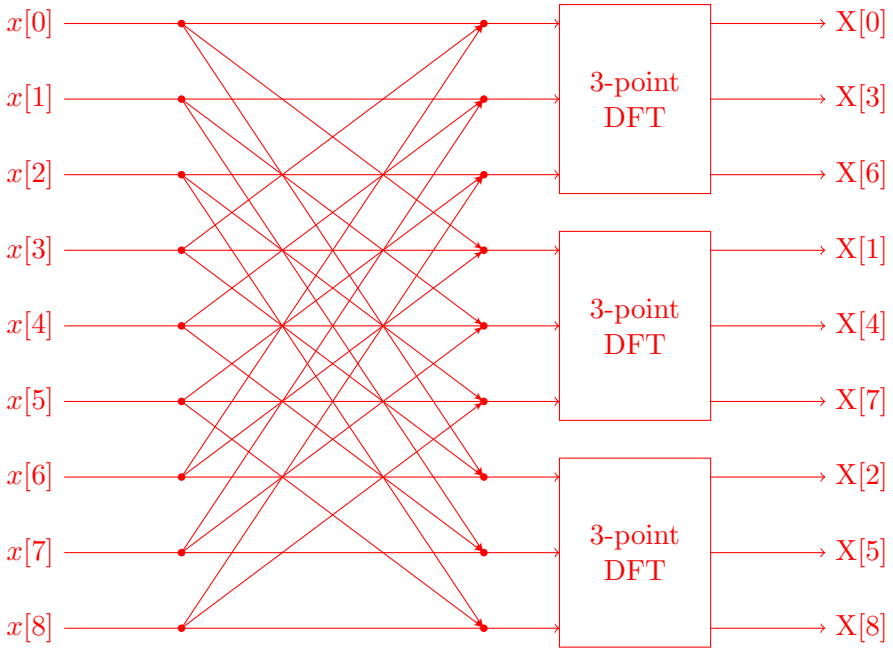
Solution:

$$\begin{aligned}
X[3r+1] &= \sum_{n=0}^{N-1} x[n] W_N^{n(3r+1)} \\
&= \sum_{n=0}^{N-1} x[n] W_N^n W_{N/3}^{nr} \\
&= \sum_{n=0}^{N/3-1} x[n] W_N^n W_{N/3}^{nr} + \sum_{n=N/3}^{2N/3-1} x[n] W_N^n W_{N/3}^{nr} + \sum_{n=2N/3}^{N-1} x[n] W_N^n W_{N/3}^{nr} \\
&= \sum_{n=0}^{N/3-1} \left(x[n] W_N^n + x[n+\frac{N}{3}] W_N^{n+\frac{N}{3}} x[n+\frac{2N}{3}] W_N^{n+\frac{2N}{3}} \right) W_{N/3}^{nr} \\
&= \sum_{n=0}^{N/3-1} \left(x[n] + W_3^1 x[n+\frac{N}{3}] + W_3^2 x[n+\frac{2N}{3}] \right) W_N^n W_{N/3}^{nr}
\end{aligned}$$

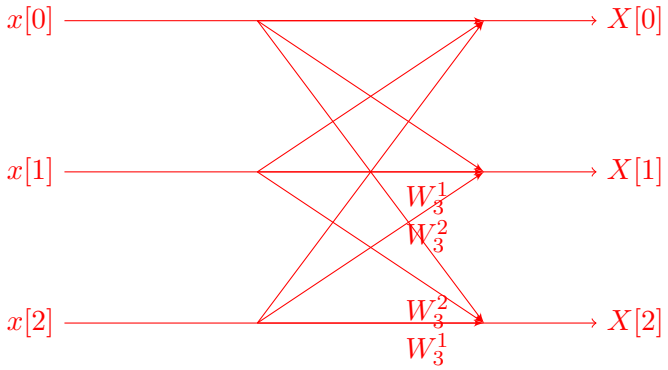
$$\begin{aligned}
X[3r+2] &= \sum_{n=0}^{N-1} x[n] W_N^{n(3r+2)} \\
&= \sum_{n=0}^{N-1} x[n] W_N^{2n} W_{N/3}^{nr} \\
&= \sum_{n=0}^{N/3-1} x[n] W_N^{2n} W_{N/3}^{nr} + \sum_{n=N/3}^{2N/3-1} x[n] W_N^{2n} W_{N/3}^{nr} + \sum_{n=2N/3}^{N-1} x[n] W_N^{2n} W_{N/3}^{nr} \\
&= \sum_{n=0}^{N/3-1} \left(x[n] W_N^{2n} + x[n+\frac{N}{3}] W_N^{2n+\frac{2N}{3}} x[n+\frac{2N}{3}] W_N^{2n+\frac{4N}{3}} \right) W_{N/3}^{nr} \\
&= \sum_{n=0}^{N/3-1} \left(x[n] + W_3^2 x[n+\frac{N}{3}] + W_3^1 x[n+\frac{2N}{3}] \right) W_N^{2n} W_{N/3}^{nr}
\end{aligned}$$

(c) Draw a flow graph for the radix-3 decimation-in-frequency FFT of a length-9 sequence. To save space, you may represent each length-3 DFT as a block with three inputs and three outputs and then draw a second diagram showing the length-3 DFT.

Solution: The flowgraph is shown below. The weights are omitted for clarity and can be found from part (b).



The 3-point DFT is shown below.



4. **[Fast Convolution]**
You are given two sequences, $\{x[n]\}_{n=0}^{349}$ and $\{h[n]\}_{n=0}^{31}$, and are asked to compute their linear convolution $y[n] = (x * h)[n]$. You decide to use the DFT to speed up the computation.

(a) What is the length of the sequence $y[n]$?

Solution: $350 + 64 - 1 = 413$

(b) Find the smallest number of zeros that should be padded to each sequence so that the linear convolution can be computed using the DFT.

Solution: The sequences must be the same length as the output. Pad 63 zeros onto $x[n]$ to get a length 413 sequence $\tilde{x}[n]$ and pad 349 zeros onto $h[n]$ to get a length 413

sequence $\tilde{h}[n]$. Then take the length 413 DFT and compute the convolution as follows:

$$y[n] = \text{DFT}^{-1}\left[\text{DFT}(\tilde{x}[n])\text{DFT}(\tilde{h}[n])\right].$$

(c) To further speed computation, you decide to use a radix-2 FFT to compute the DFT. How should the sequences be padded so that their linear convolution can be computed using the smallest possible radix-2 FFT?
Solution: The sequences must have the same length and that length must be a power of 2. The smallest power of 2 larger than 413 is 512. Pad 162 zeros to $x[n]$ and 448 zeros to $h[n]$ to give two length 512 sequences. Then take the length 512 DFT, multiply, and invert to find $y[n]$.

5. **[Fast Convolution]**
In high-performance DSP applications, we must deal with very high sample rates and large filters. With so much data, computational efficiency is critical. Consider a (fairly small) vector $\mathbf{x}$ of 40,000 samples that is to be filtered with a length 256 FIR filter $\mathbf{h}$.

(a) First, consider the old-fashioned way: direct linear convolution. Determine the number of real multiplications and additions required to convolve $\mathbf{x}$ with $\mathbf{h}$.
Solution: Let $N = 40000$ be the length of $x[n]$ and $M = 256$ be the length of $h[n]$. From the Chapter 14 notes, the number of multiply-accumulates for linear convolution is

$$\frac{M(M-1)}{2} + M(N-M+1) + \frac{M(M-1)}{2} = MN = 10240000.$$

(b) Next, consider "fast convolution" using two complex-valued radix-2 FFTs and one inverse FFT. The transform length is chosen as an appropriate power of two. Assume reduced butterflies with one complex multiplication per butterfly. Determine the number of real multiplications and additions required to implement this convolution.
Solution: The output will have length $M + N - 1 = 40255$ The next largest power of 2 is $K = 2^{16} = 65536$. This method of fast convolution requires two FFTs, one inverse FFT, and one complex multiply. With reduced butterflies, each FFT and inverse FFT takes $\frac{K}{2}\log_2 K$ multiplications and $K\log_2 K$ additions for a total of

$$3\left(\frac{K}{2}\log_2 K\right) + K = K\left(1 + \frac{3}{2}\log_2 K\right) = 65536\left(1 + \frac{3}{2}(16)\right) = 1638400$$

complex multiplications and

$$3K\log_2 K = 3(65536)(16) = 3145728$$

complex additions.
If each complex multiplication requires four real multiplications and two real additions and each complex addition requires two real additions, then the total number of real multiplications is $4(1638400) = 6553600$ and the number of real additions is $2(1638400) + 2(3145728) = 9568256$. Note that there are more efficient implementations that require fewer operations.

(c) Since the data vector is much longer than the filter length, the overlap-add algorithm yields a more e cient solution. Determine the size of the power-of-two length FFT that minimizes the total number of multiplies per output sample using the overlap-add method of fast convolution. Estimate the number of real multiplications required by the overlap-add method. Justify your answer.

Solution: With the overlap-add method, we split the input $x[n]$ into frames of length L. Let K be the (to-be-decided) power-of-two length of the FFT for fast convolution. Since $K = L + M - 1$, we have that $L = K - 256 + 1 = L - 255$.

We first precompute the length-K FFT of $h[n]$. This requires $\frac{K}{2} log_2 K$ complex multiplication, and one length-K inverse FFT, for a total of

$$\frac{K}{2}\log_2 K + K + \frac{K}{2}\log_2 K = K(1 + \log_2 K)$$

complex multiplies per frame.

There are $\lceil \frac{N}{L} \rceil = \left\lceil \frac{40000}{K-255} \right\rceil$ frames. Thus, the total number of complex multiplications is

$$\left\lceil \frac{40000}{K-255} \right\rceil K(1 + \log_2 K) + \frac{K}{2}\log_2 K$$

If there are four real multiplications for each complex multiplications, then there are

$$\left\lceil \frac{40000}{K-255} \right\rceil 4K(1 + \log_2 K) + 2K\log_2 K$$

real multiplications. Let's compute this expression for a few values of K

K	Real multiplications
256	368644096
512	3204096
1024	2408448
2048	2306048
4096	2441216
8196	2965504

The minimal computation is achieved by $K = 2048$.

(d) Compare the computational efficiency of these three methods of convolution. Note that multiplication is generally much more expensive than addition.

Solution: The numbers of real multiplications required by three convolution methods are

Method	Real multiplications
Linear	10000000
Fast Convolution	6553600
Overlap Add	2306048