Fast Fourier Transform (FFT)
(Theory and Implementation)

Learning Objectives

- DFT algorithm.
- Conversion of DFT to FFT algorithm.
- Implementation of the FFT algorithm.

DFT Algorithm

- The Fourier transform of an analogue signal x(t) is given by:

$$X(\omega) = \int_{-\infty}^{+\infty} x(t) e^{-j\omega t} dt$$

- The Discrete Fourier Transform (DFT) of a discrete-time signal x(nT) is given by:

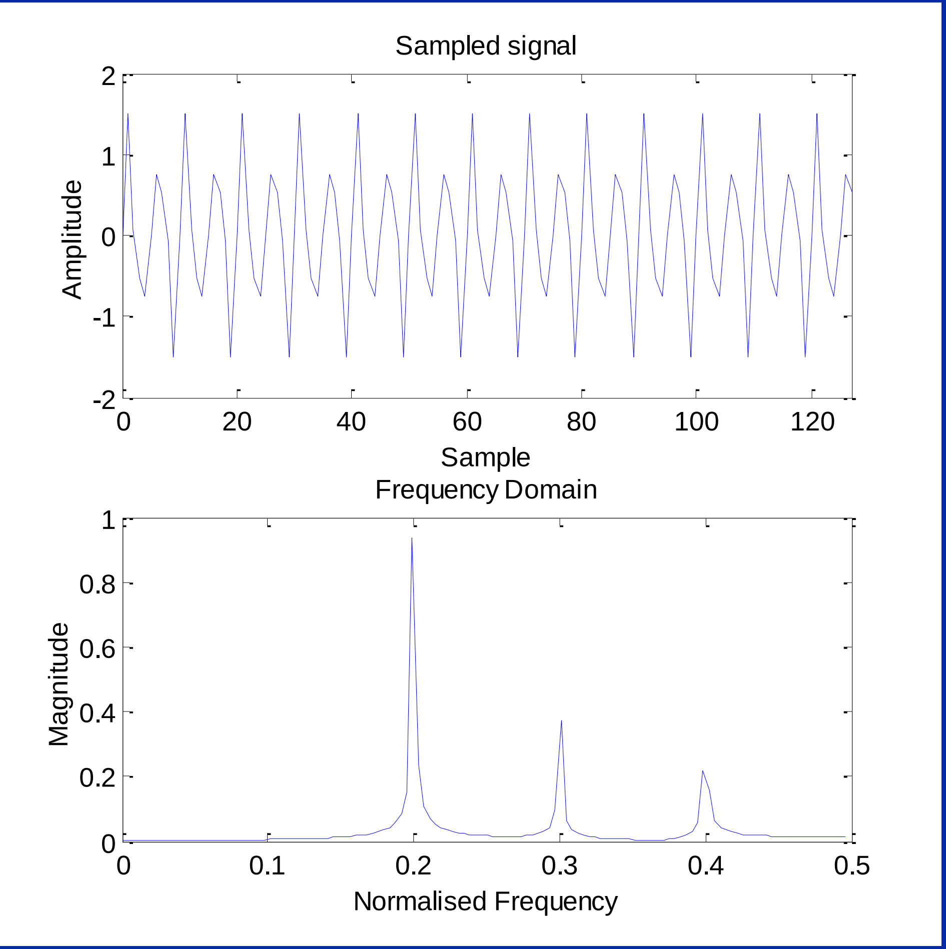
$$X(k) = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}nk}$$

  - Where:

$$k = 0, 1, \ldots N-1$$
$$x(nT) = x[n]$$

DFT Algorithm

- If we let: $e^{-j\frac{2\pi}{N}} = W_N$ then: $X(k) = \sum_{n=0}^{N-1} x[n] W_N^{nk}$



DFT Algorithm

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

x[n] = input
X[k] = frequency bins
W = twiddle factors

$X(0) = x[0]W_N^{0} + x[1]W_N^{0*1} + \ldots + x[N-1]W_N^{0*(N-1)}$

$X(1) = x[0]W_N^{0} + x[1]W_N^{1*1} + \ldots + x[N-1]W_N^{1*(N-1)}$

:

$X(k) = x[0]W_N^{0} + x[1]W_N^{k*1} + \ldots + x[N-1]W_N^{k*(N-1)}$

:

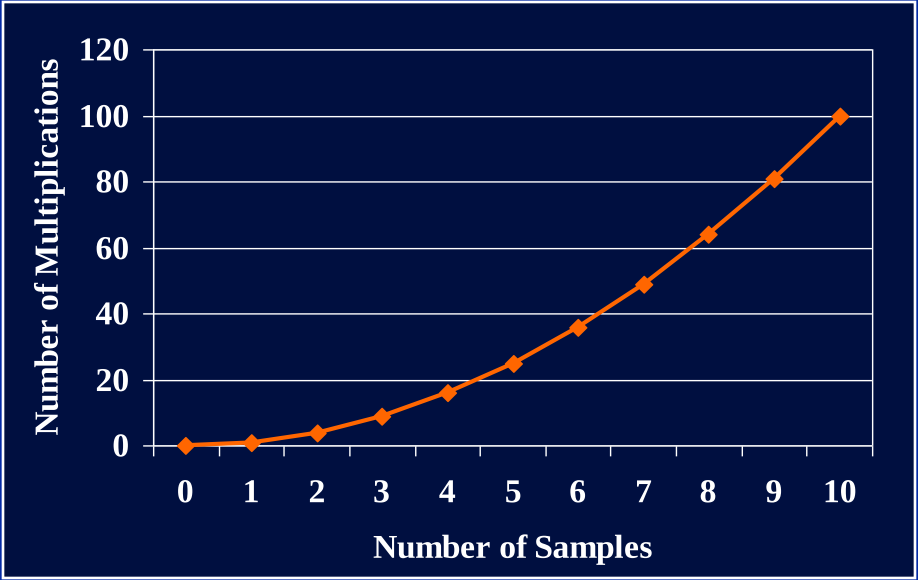
$X(N-1) = x[0]W_N^{0} + x[1]W_N^{(N-1)*1} + \ldots + x[N-1]W_N^{(N-1)(N-1)}$

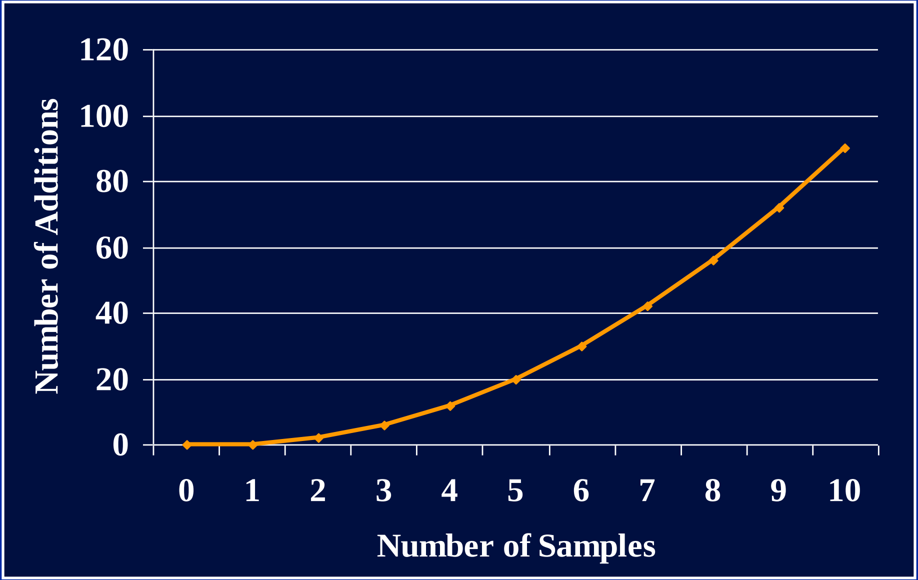
Note: For N samples of x we have N frequencies representing the signal.

Performance of the DFT Algorithm

- The DFT requires N^2 (NxN) complex multiplications:
  - Each X(k) requires N complex multiplications.
  - Therefore to evaluate all the values of the DFT (X(0) to X(N-1)) N^2 multiplications are required.
- The DFT also requires (N-1)*N complex additions:
  - Each X(k) requires N-1 additions.
  - Therefore to evaluate all the values of the DFT (N-1)*N additions are required.

Performance of the DFT Algorithm





- Can the number of computations required be reduced?

DFT $\rightarrow$ FFT

- A large amount of work has been devoted to reducing the computation time of a DFT.
- This has led to efficient algorithms which are known as the Fast Fourier Transform (FFT) algorithms.

DFT → FFT

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

x[n] = x[0], x[1], …, x[N-1]

- **Lets divide the sequence x[n] into even and odd sequences:**
  - **x[2n] = x[0], x[2], …, x[N-2]**
  - **x[2n+1] = x[1], x[3], …, x[N-1]**

DFT → FFT

- **Equation 1 can be rewritten as:**

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x[2n] W_N^{2nk} + \sum_{n=0}^{\frac{N}{2}-1} x[2n+1] W_N^{(2n+1)k} \qquad [2]$$

- **Since:**

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

$$W_N^{(2n+1)k} = W_N^k \cdot W_{\frac{N}{2}}^{nk}$$

- **Then:**

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x[2n] W_{\frac{N}{2}}^{nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x[2n+1] W_{\frac{N}{2}}^{nk} = Y(k) + W_N^k Z(k)$$

DFT → FFT

- The result is that an N-point DFT can be divided into two N/2 point DFT's:

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

N-point DFT

- Where Y(k) and Z(k) are the two N/2 point DFTs operating on even and odd samples respectively:

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x_1[n] W_{\frac{N}{2}}^{nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x_2[n] W_{\frac{N}{2}}^{nk}$$
$$= Y(k) + W_N^k Z(k)$$

Two N/2-point DFTs

DFT → FFT

- **Periodicity** and **symmetry** of W can be exploited to simplify the DFT further:

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x_1[n] W_{\frac{N}{2}}^{nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x_2[n] W_{\frac{N}{2}}^{nk} \qquad [3]$$

$$\vdots$$

$$X\left(k+\frac{N}{2}\right) = \sum_{n=0}^{\frac{N}{2}-1} x_1[n] W_{\frac{N}{2}}^{n\left(k+\frac{N}{2}\right)} + W_N^{k+\frac{N}{2}} \sum_{n=0}^{\frac{N}{2}-1} x_2[n] W_{\frac{N}{2}}^{n\left(k+\frac{N}{2}\right)}$$

Or:

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

: Symmetry

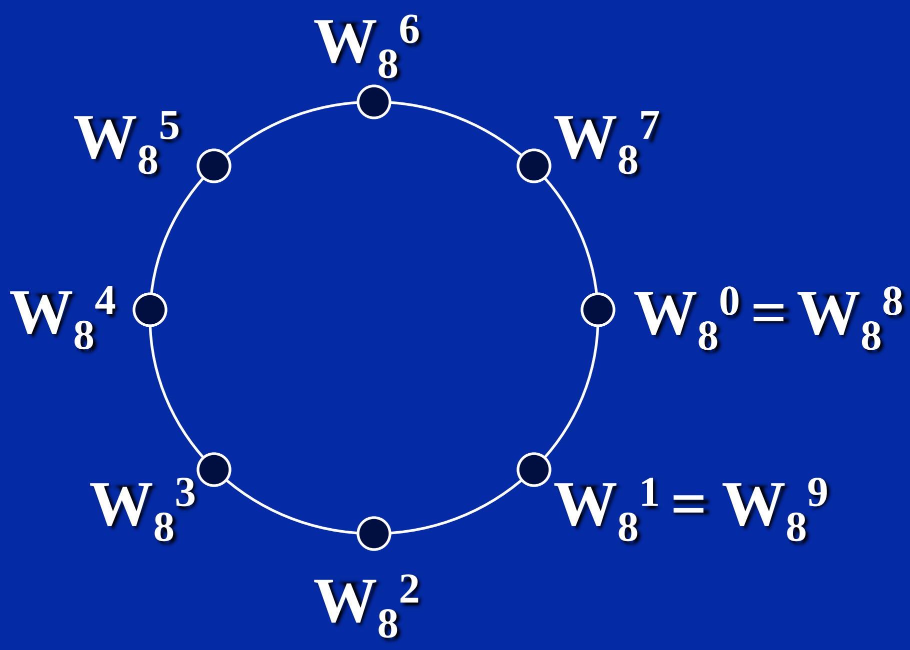
And:

$$W_{\frac{N}{2}}^{k+\frac{N}{2}} = e^{-j\frac{2\pi}{N/2}k} e^{-j\frac{2\pi}{N/2}\frac{N}{2}} = e^{-j\frac{2\pi}{N/2}k} = W_{\frac{N}{2}}^k$$

: Periodicity

DFT → FFT

- **Symmetry and periodicity:**



$$W_N^{k+N/2} = -W_N^k$$

$$W_{N/2}^{k+N/2} = W_{N/2}^k$$

$$W_8^{k+4} = -W_8^k$$

$$W_8^{k+8} = W_8^k$$

DFT → FFT

- Finally by exploiting the symmetry and periodicity, Equation 3 can be written as:

$$X\left(k+\frac{N}{2}\right)=\sum_{n=0}^{\frac{N}{2}-1}x_1[n]W_{\frac{N}{2}}^{nk}-W_N^k\sum_{n=0}^{\frac{N}{2}-1}x_2[n]W_{\frac{N}{2}}^{nk}$$
$$=Y(k)-W_N^kZ(k)$$

[4]

DFT → FFT

$$X(k) = Y(k) + W_N^k Z(k); \quad k = 0, \ldots \left(\frac{N}{2} - 1\right)$$

$$X\left(k + \frac{N}{2}\right) = Y(k) - W_N^k Z(k); \quad k = 0, \ldots \left(\frac{N}{2} - 1\right)$$

- **$Y(k)$ and $W_N^k Z(k)$ only need to be calculated once and used for both equations.**
- **Note: the calculation is reduced from 0 to N-1 to 0 to (N/2 - 1).**

DFT → FFT

$$X(k) = Y(k) + W_N^k Z(k); \quad k = 0, \ldots \left(\frac{N}{2} - 1\right)$$

$$X\left(k + \frac{N}{2}\right) = Y(k) - W_N^k Z(k); \quad k = 0, \ldots \left(\frac{N}{2} - 1\right)$$

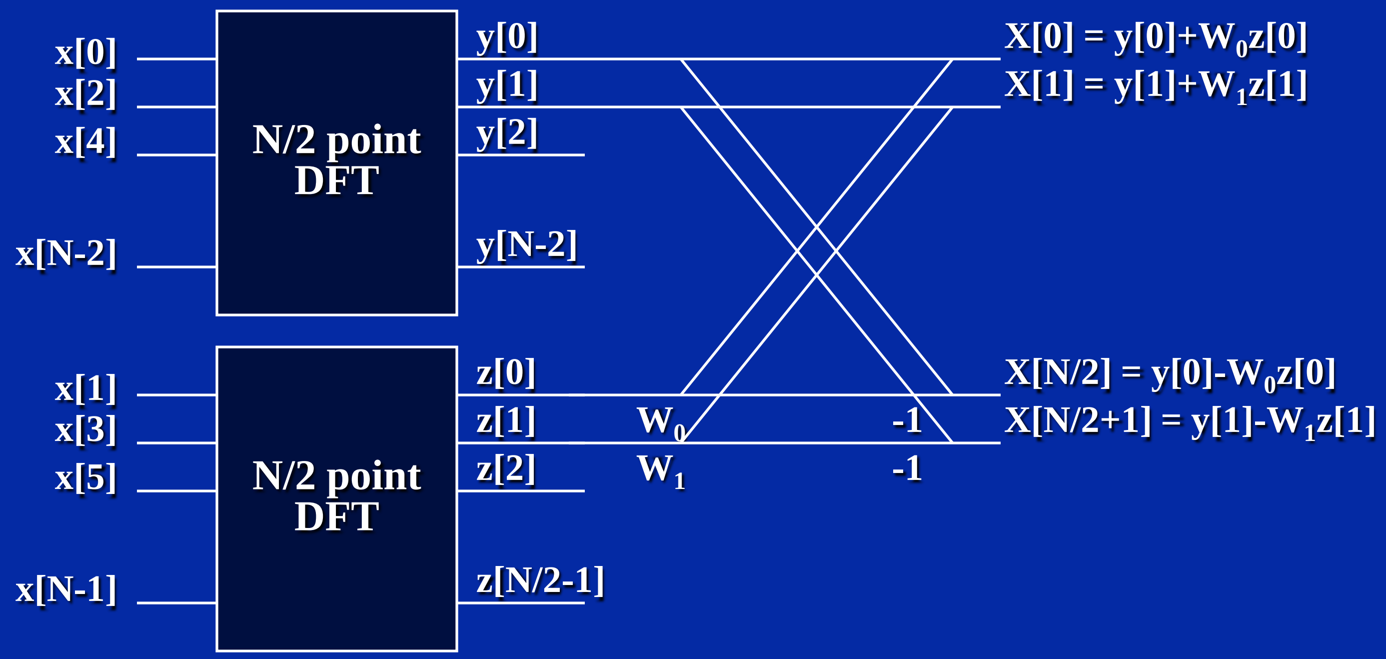
- **Y(k) and Z(k) can also be divided into N/4 point DFTs using the same process shown above:**

$$Y(k) = U(k) + W_{\frac{N}{2}}^k V(k) \qquad Z(k) = P(k) + W_{\frac{N}{2}}^k Q(k)$$

$$Y\left(k + \frac{N}{4}\right) = U(k) - W_{\frac{N}{2}}^k V(k) \qquad Z\left(k + \frac{N}{4}\right) = P(k) - W_{\frac{N}{2}}^k Q(k)$$

- **The process continues until we reach 2 point DFTs.**

DFT → FFT



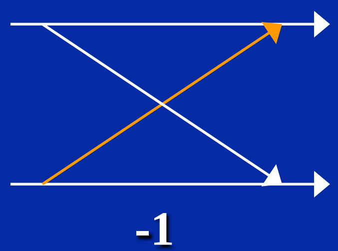
- Illustration of the first decimation in time FFT.

FFT Implementation

- **Calculation of the output of a 'butterfly':**

$Y(k) = U_r + jU_i$ → $U' = U_r' + jU_i' = X(k)$

$W_N^k Z(k) = (L_r + jL_i)(W_r + jW_i)$ → $L' = L_r' + jL_i' = X(k+N/2)$

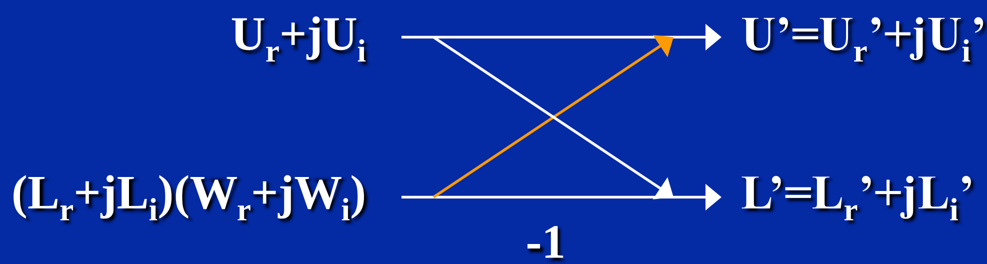


Key:	U = Upper	r = real
	L = Lower	i = imaginary

- **Different methods are available for calculating the outputs U' and L'.**
- **The best method is the one with the least number of multiplications and additions.**

FFT Implementation

- **Calculation of the output of a 'butterfly':**



$$(L_r + jL_i)(W_r + jW_i) = L_rW_r + jL_rW_i + jL_iW_r - L_iW_i$$

$$\begin{aligned} U' &= \left[(L_rW_r - L_iW_i) + j(L_rW_i + L_iW_r)\right] + \left[U_r + jU_i\right] \\ &= (L_rW_r - L_iW_i + U_r) + j(L_rW_i + L_iW_r + U_i) \end{aligned}$$

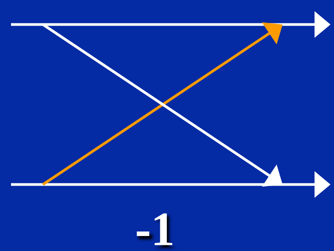
$$\begin{aligned} L' &= (U_r + jU_i) - \left[(L_rW_r - L_iW_i) + j(L_rW_i + L_iW_r)\right] \\ &= (U_r - L_rW_r + L_iW_i) + j(U_i - L_rW_i - L_iW_r) \end{aligned}$$

FFT Implementation

- **Calculation of the output of a 'butterfly':**

$U_r + jU_i$

$(L_r + jL_i)(W_r + jW_i)$



$U' = U_r' + jU_i' = (L_r W_r - L_i W_i + U_r) + j(L_r W_i + L_i W_r + U_i)$

$L' = L_r' + jL_i' = (U_r - L_r W_r + L_i W_i) + j(U_i - L_r W_i - L_i W_r)$

- **To further minimise the number of operations (* and +), the following are calculated only once:**

$temp1 = L_r W_r$ $\quad temp2 = L_i W_i$ $\quad temp3 = L_r W_i$ $\quad temp4 = L_i W_r$

$temp1\_2 = temp1 - temp2$ $\quad temp3\_4 = temp3 + temp4$

$U_r' = temp1 - temp2 + U_r = temp1\_2 + U_r$

$U_i' = temp3 + temp4 + U_i = temp3\_4 + U_i$

$L_r' = U_r - temp1 + temp2 = U_r - temp1\_2$

$L_i' = U_i - temp3 - temp4 = U_i - temp3\_4$

FFT Implementation (Butterfly Calculation)

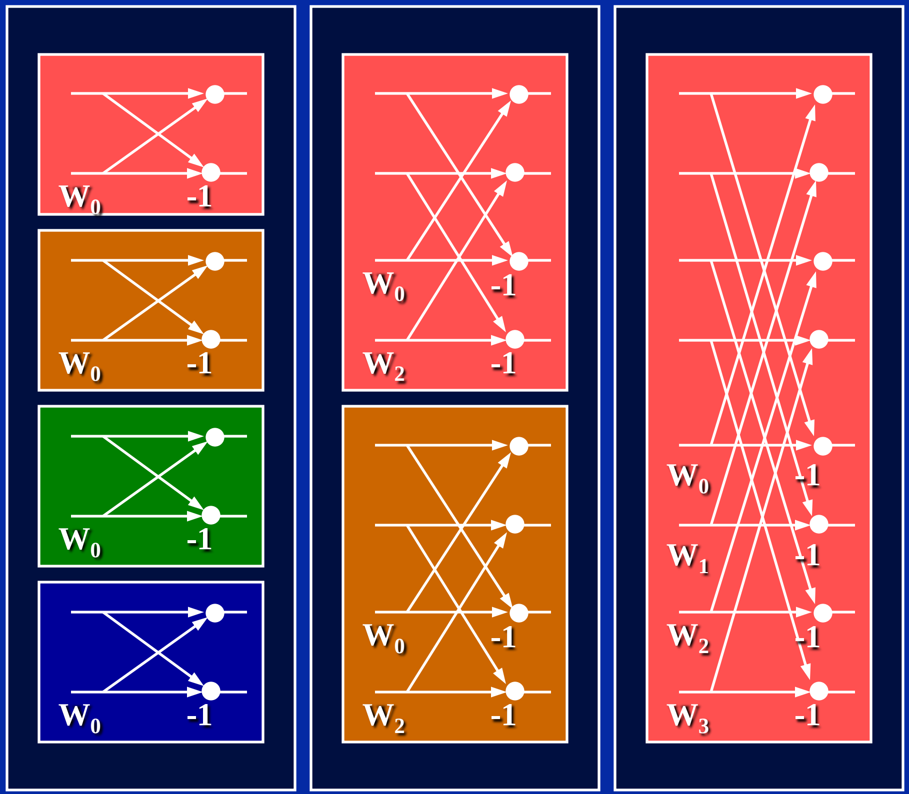
- Converting the butterfly calculation into 'C' code:

```
temp1 = (y[lower].real * WR);
temp2 = (y[lower].imag * WI);
temp3 = (y[lower].real * WI);
temp4 = (y[lower].imag * WR);

temp1_2 = temp1 - temp2;
temp3_4 = temp 3 + temp4;

y[upper].real = temp1_2 + y[upper].real;
y[upper].imag = temp3_4 + y[upper].imag;
y[lower].imag = y[upper].imag - temp3_4;
y[lower].real = y[upper].real - temp1_2;
```

FFT Implementation

- To efficiently implement the FFT algorithm a few observations are made:
  - Each stage has the same number of butterflies (number of butterflies = N/2, N is number of points).
  - The number of DFT groups per stage is equal to $(N/2^{stage})$.
  - The difference between the upper and lower leg is equal to $2^{stage-1}$.
  - The number of butterflies in the group is equal to $2^{stage-1}$.

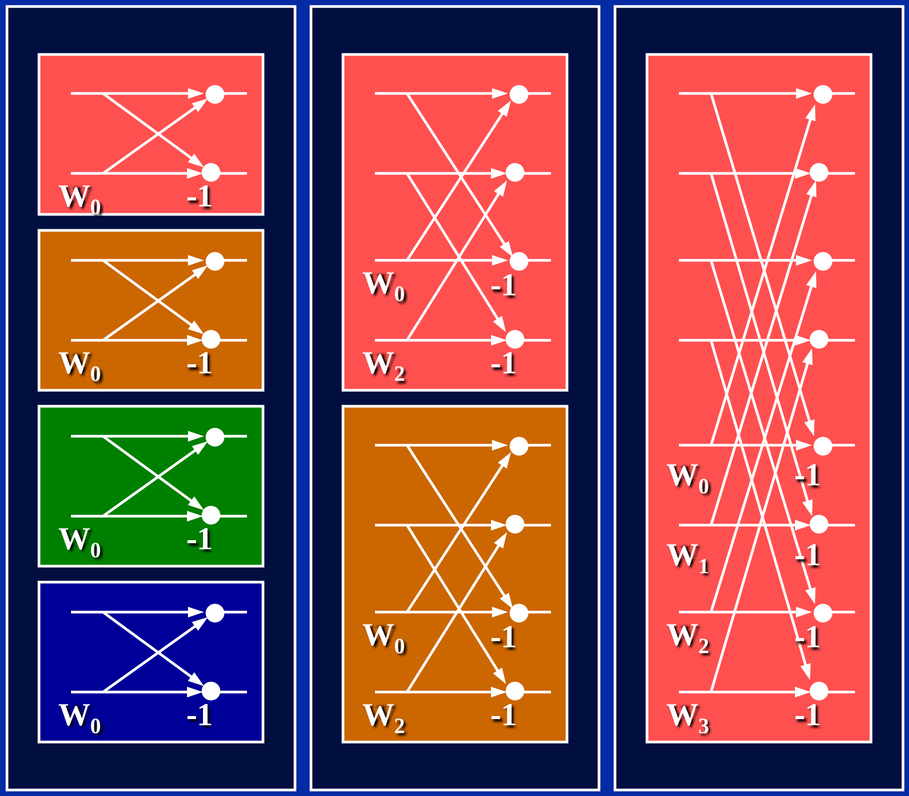
FFT Implementation



Example: 8 point FFT

- **Decimation in time FFT:**
  - **Number of stages = $\log_2 N$**
  - **Number of blocks/stage = $N/2^{stage}$**
  - **Number of butterflies/block = $2^{stage-1}$**

FFT Implementation

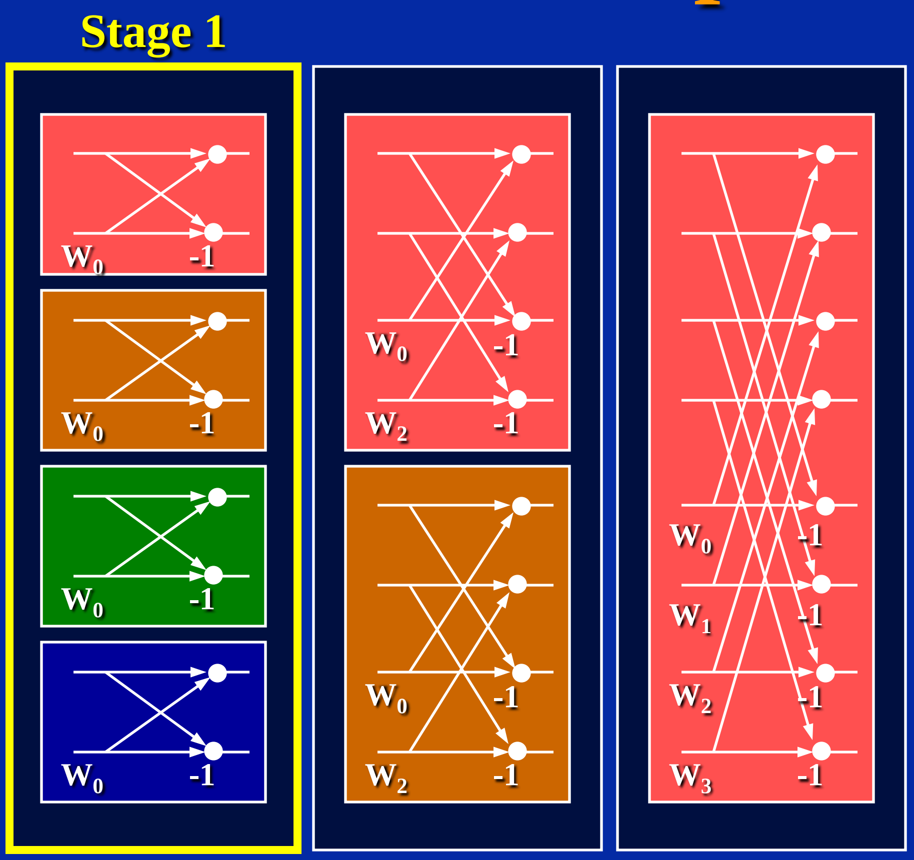


Example: 8 point FFT

(1) Number of stages:

- **Decimation in time FFT:**
  - Number of stages = $\log_2 N$
  - Number of blocks/stage = $N/2^{stage}$
  - Number of butterflies/block = $2^{stage-1}$

FFT Implementation



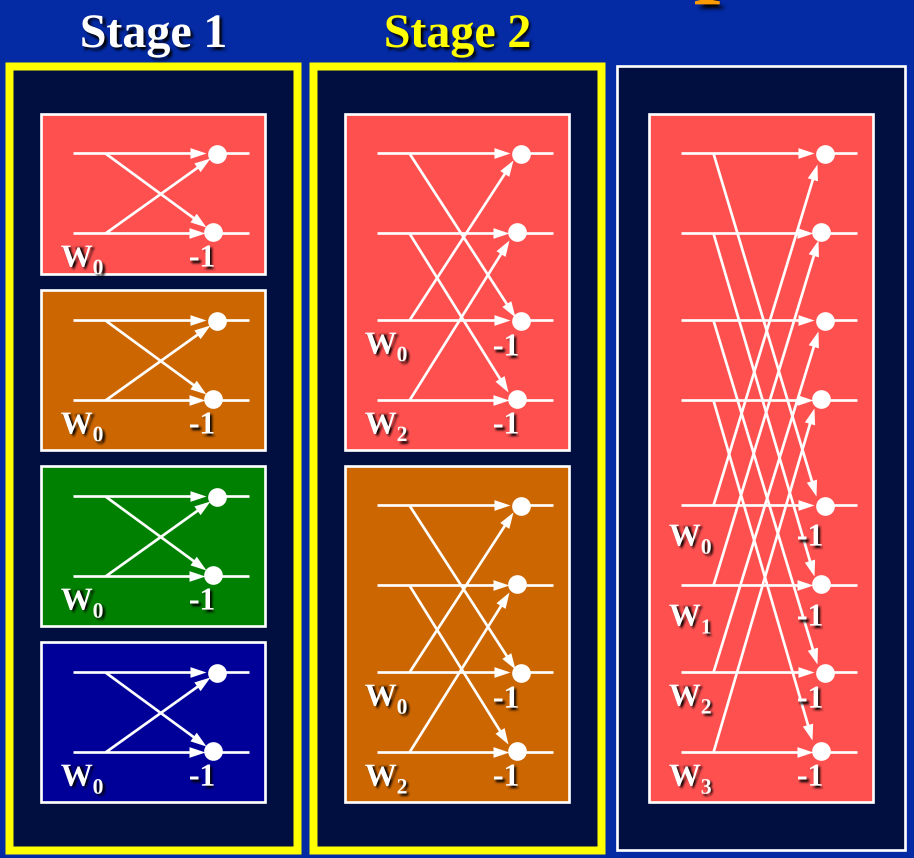
Example: 8 point FFT

(1) Number of stages:

- $N_{stages} = 1$

- **Decimation in time FFT:**
  - **Number of stages = $\log_2 N$**
  - **Number of blocks/stage = $N/2^{stage}$**
  - **Number of butterflies/block = $2^{stage-1}$**

FFT Implementation



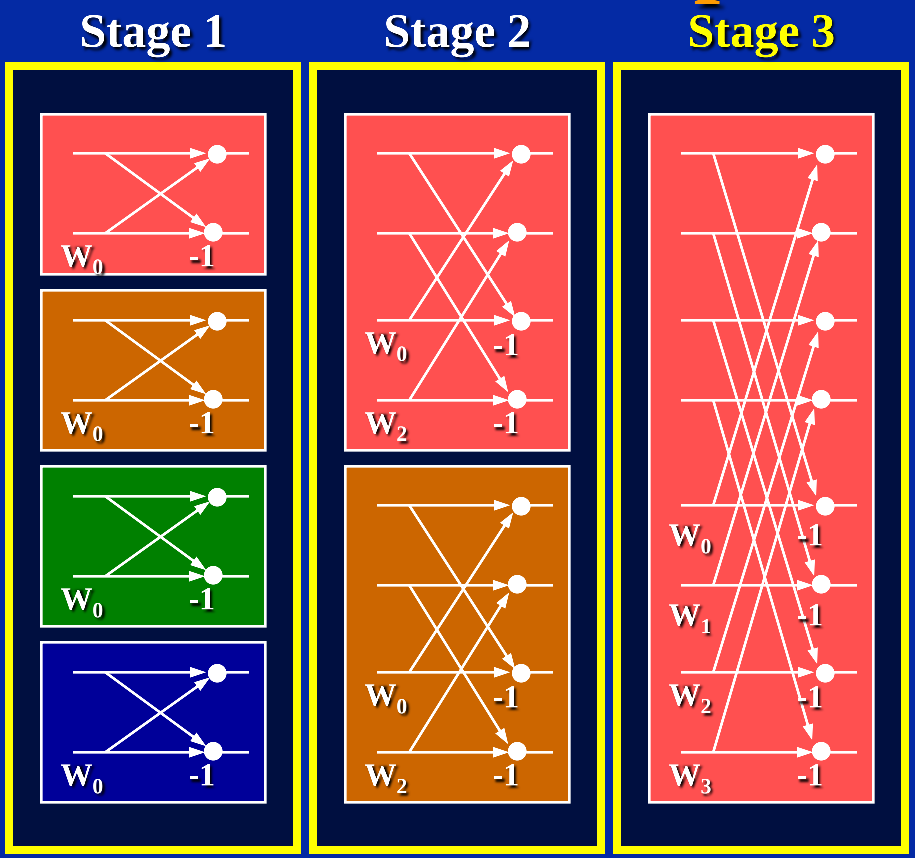
Example: 8 point FFT

(1) Number of stages:

- $N_{stages} = 2$

- **Decimation in time FFT:**
  - Number of stages = $\log_2 N$
  - Number of blocks/stage = $N/2^{stage}$
  - Number of butterflies/block = $2^{stage-1}$

FFT Implementation



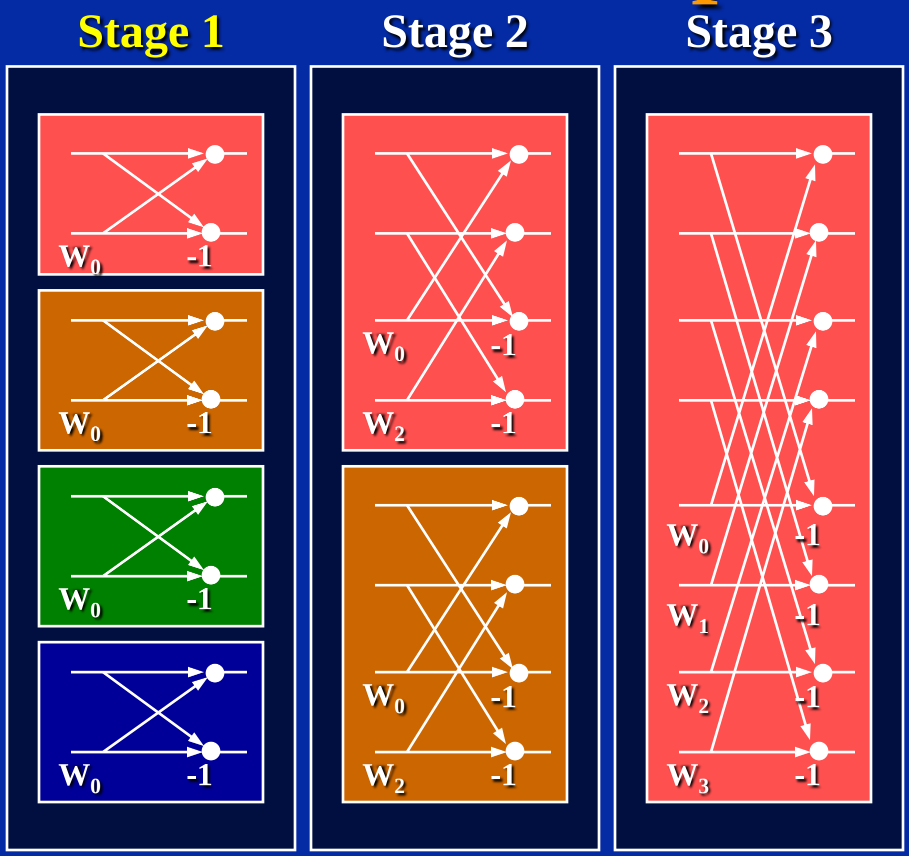
Example: 8 point FFT

(1) Number of stages:

- $N_{stages} = 3$

- **Decimation in time FFT:**
  - Number of stages = $\log_2 N$
  - Number of blocks/stage = $N/2^{stage}$
  - Number of butterflies/block = $2^{stage-1}$

FFT Implementation



Example: 8 point FFT

(1) Number of stages:

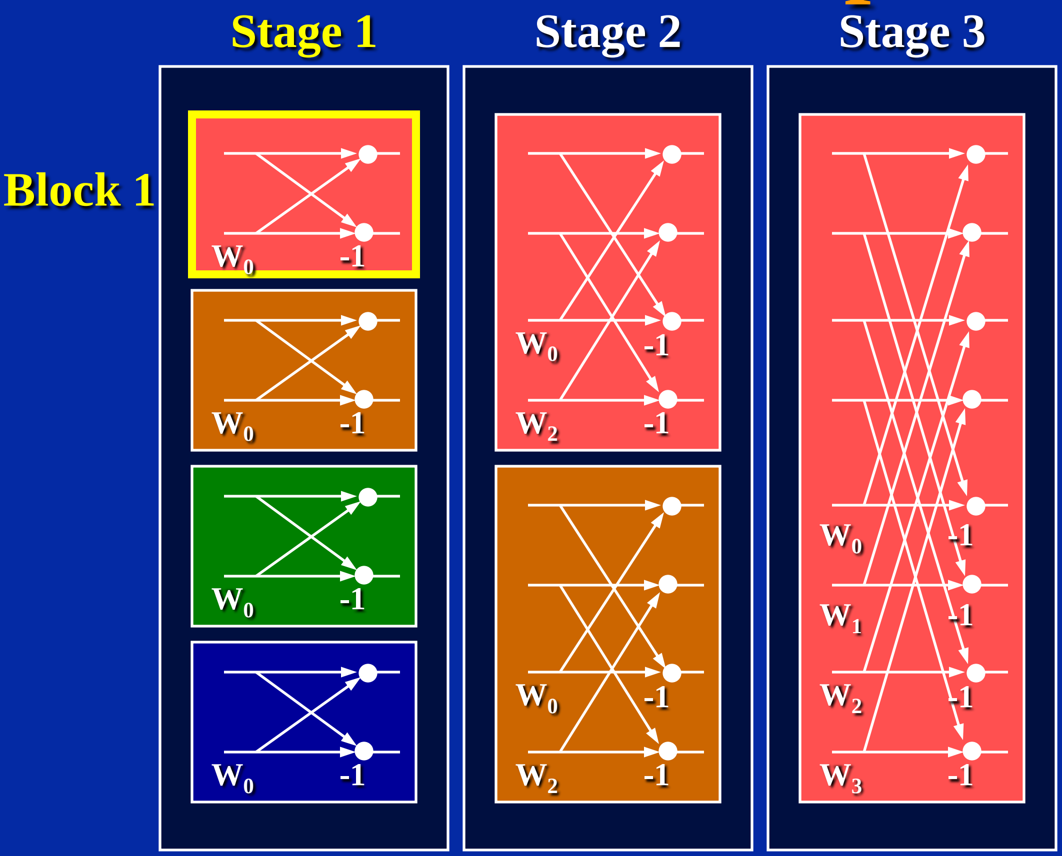
- $N_{stages} = 3$

(2) Blocks/stage:

- Stage 1:

- Decimation in time FFT:
  - Number of stages = $\log_2 N$
  - Number of blocks/stage = $N/2^{stage}$
  - Number of butterflies/block = $2^{stage-1}$

FFT Implementation



Example: 8 point FFT

(1) Number of stages:

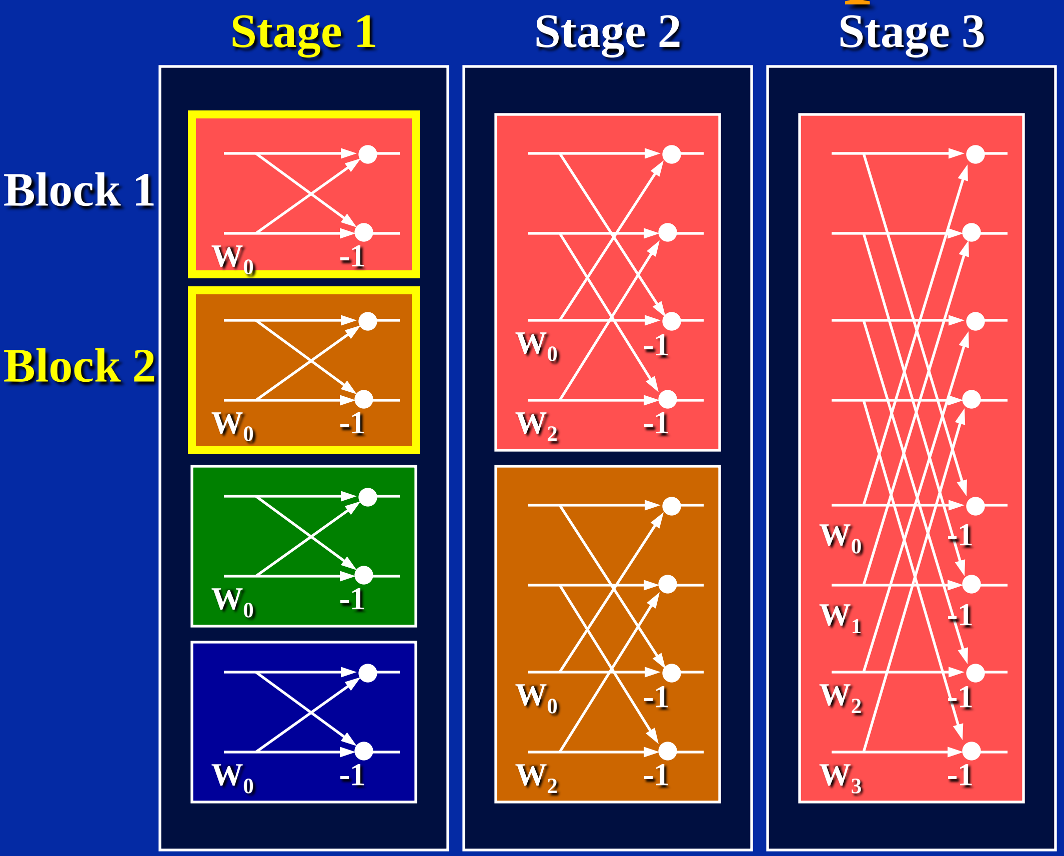
- $N_{stages} = 3$

(2) Blocks/stage:

- Stage 1: $N_{blocks} = 1$

- **Decimation in time FFT:**
  - Number of stages = $\log_2 N$
  - Number of blocks/stage = $N/2^{stage}$
  - Number of butterflies/block = $2^{stage-1}$

FFT Implementation



Example: 8 point FFT

(1) Number of stages:

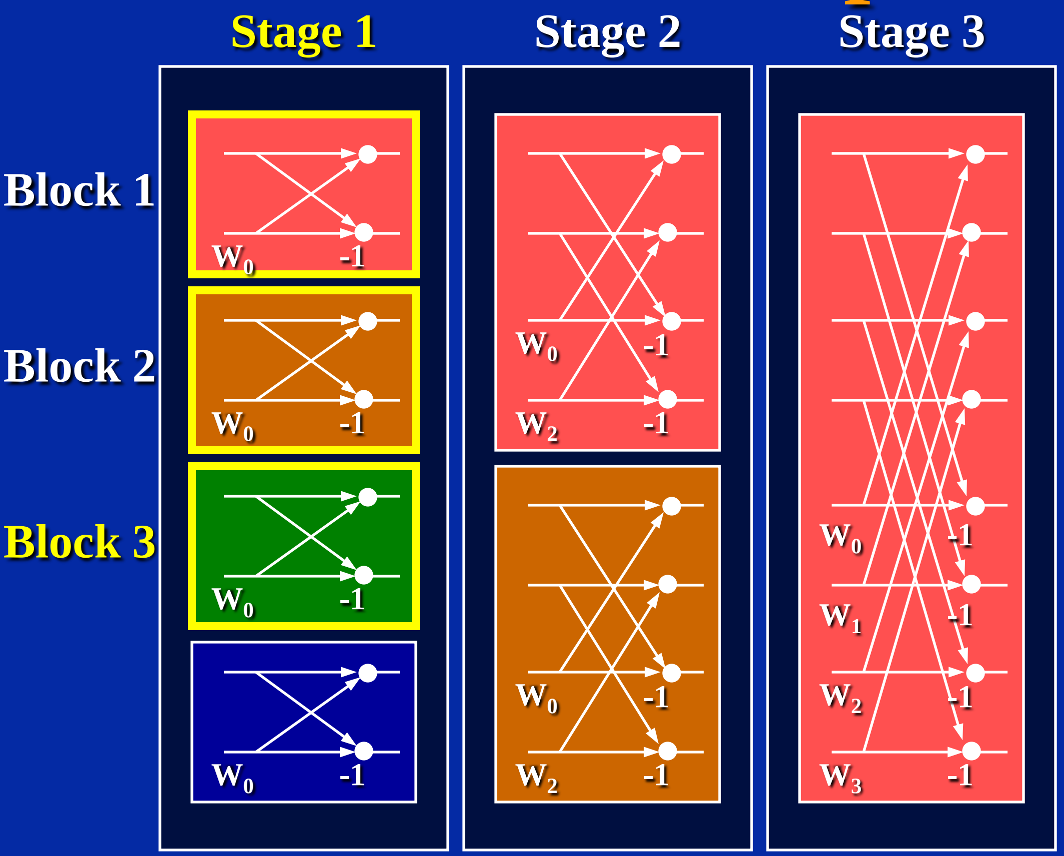
- $N_{stages} = 3$

(2) Blocks/stage:

- **Stage 1: $N_{blocks} = 2$**

- **Decimation in time FFT:**
  - **Number of stages = $\log_2 N$**
  - **Number of blocks/stage = $N/2^{stage}$**
  - **Number of butterflies/block = $2^{stage-1}$**

FFT Implementation



Example: 8 point FFT

(1) Number of stages:

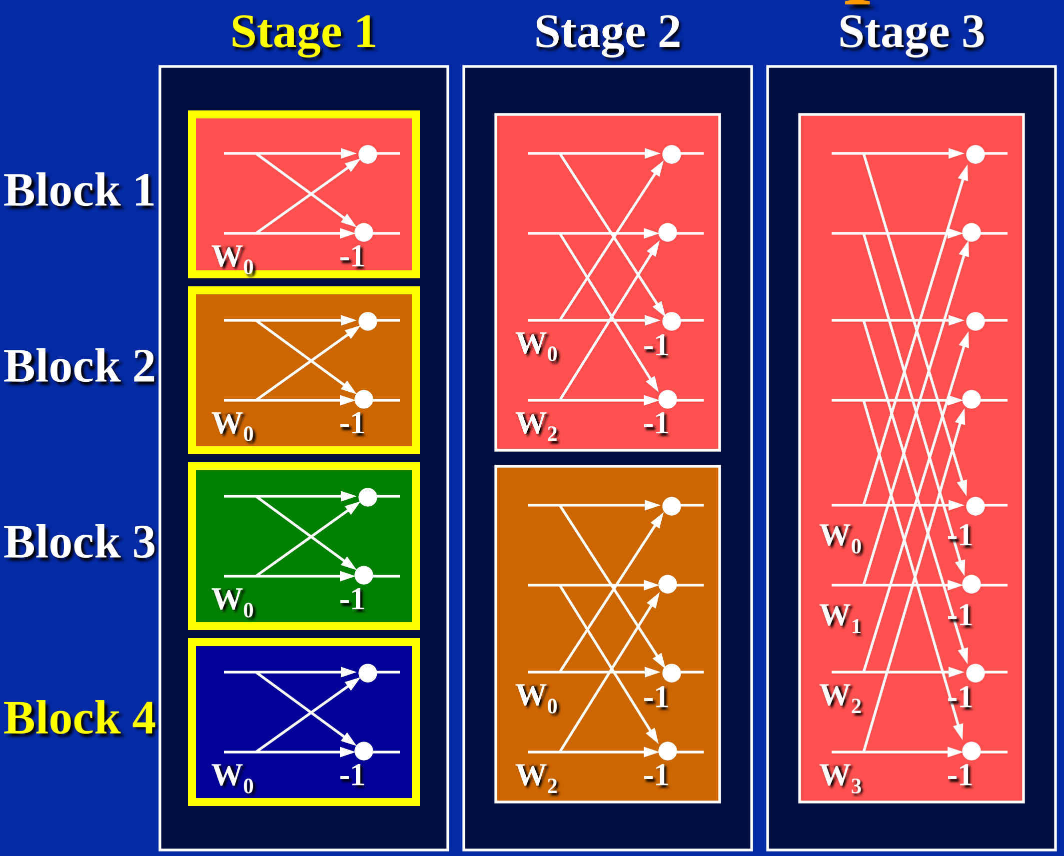
- $N_{stages} = 3$

(2) Blocks/stage:

- Stage 1: $N_{blocks} = 3$

- **Decimation in time FFT:**
  - Number of stages = $\log_2 N$
  - Number of blocks/stage = $N/2^{stage}$
  - Number of butterflies/block = $2^{stage-1}$

FFT Implementation



Example: 8 point FFT

(1) Number of stages:

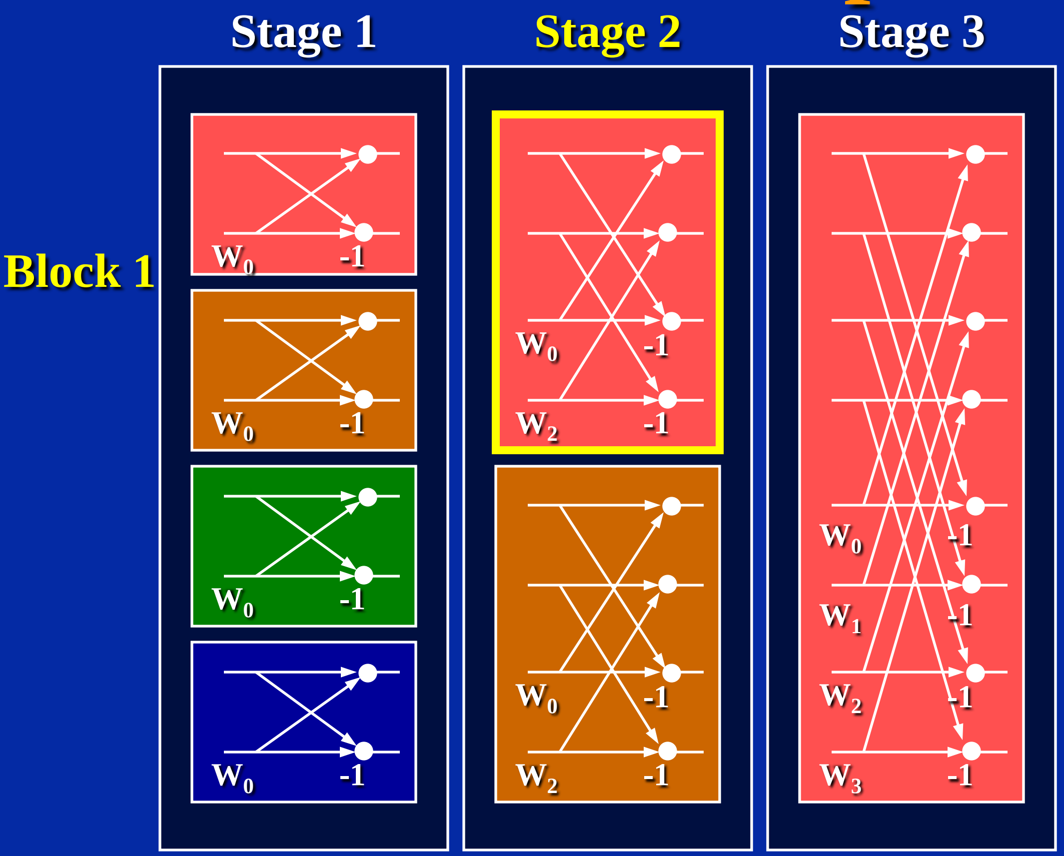
- $N_{stages} = 3$

(2) Blocks/stage:

- Stage 1: $N_{blocks} = 4$

- **Decimation in time FFT:**
  - Number of stages = $\log_2 N$
  - Number of blocks/stage = $N/2^{stage}$
  - Number of butterflies/block = $2^{stage-1}$

FFT Implementation



Example: 8 point FFT

(1) Number of stages:

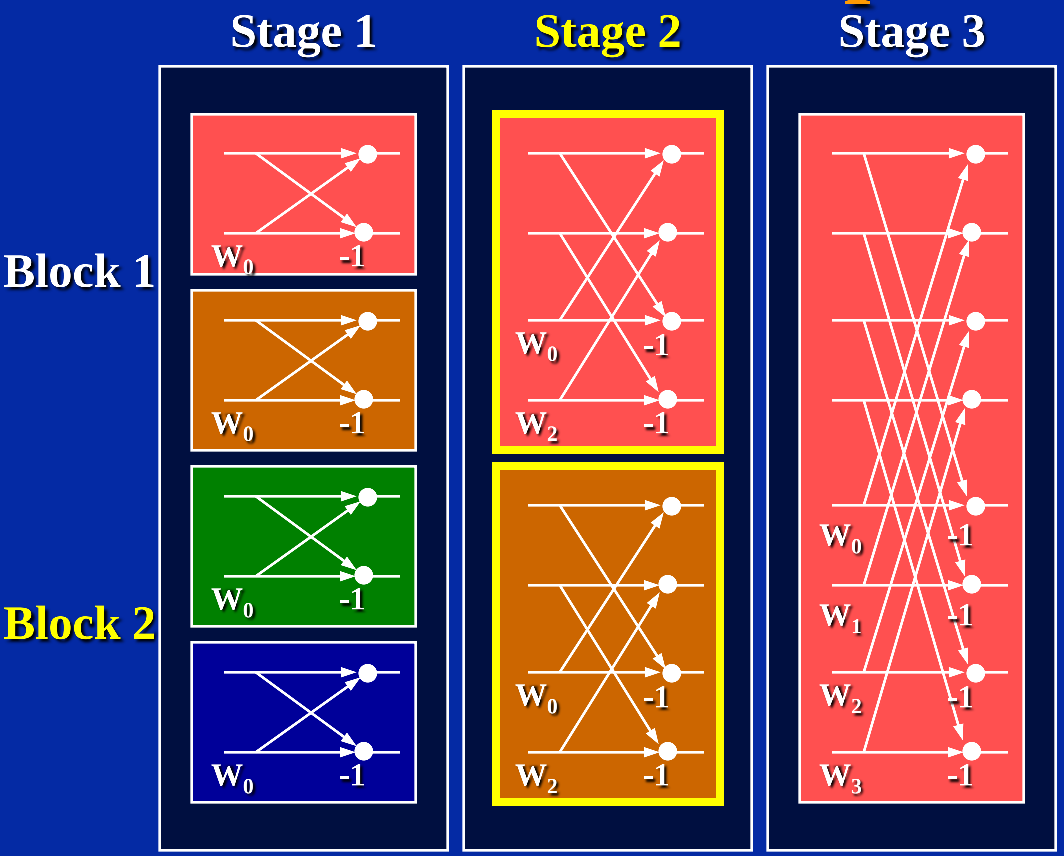
- $N_{stages} = 3$

(2) Blocks/stage:

- Stage 1: $N_{blocks} = 4$
- Stage 2: $N_{blocks} = 1$

- **Decimation in time FFT:**
  - Number of stages = $\log_2 N$
  - Number of blocks/stage = $N/2^{stage}$
  - Number of butterflies/block = $2^{stage-1}$

FFT Implementation



Example: 8 point FFT

(1) Number of stages:

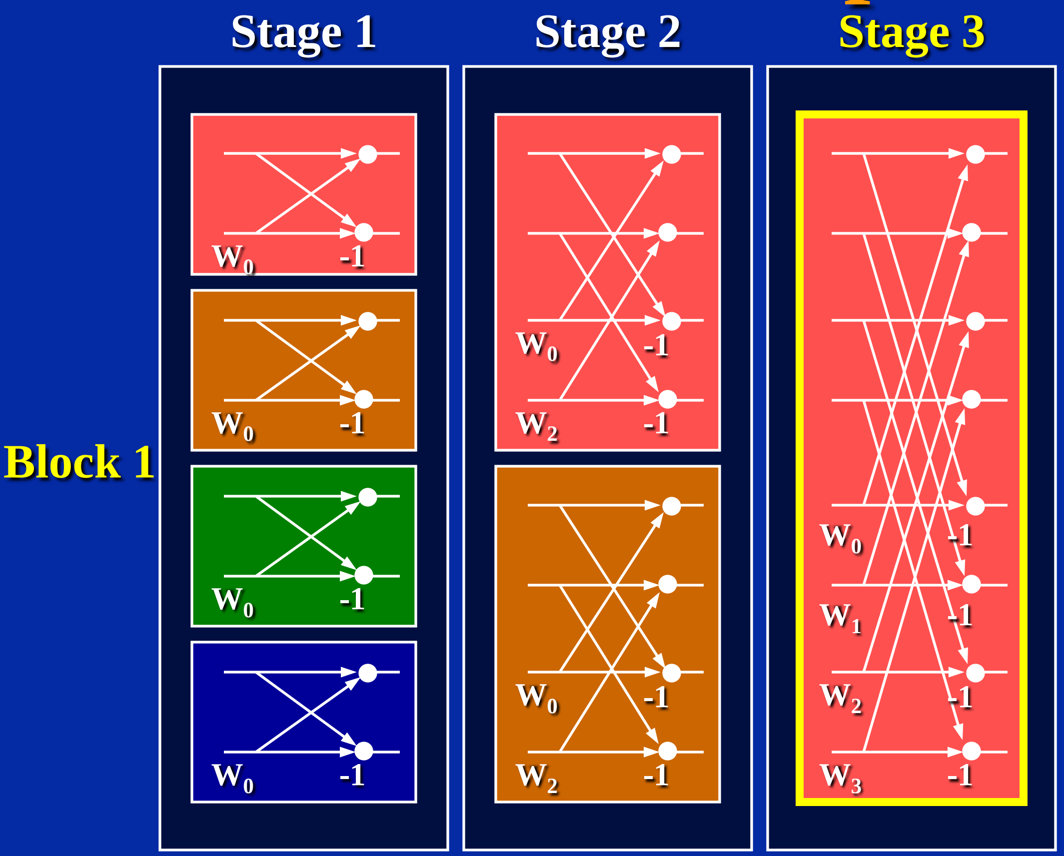
- $N_{stages} = 3$

(2) Blocks/stage:

- Stage 1: $N_{blocks} = 4$
- Stage 2: $N_{blocks} = 2$

- **Decimation in time FFT:**
  - Number of stages = $\log_2 N$
  - Number of blocks/stage = $N/2^{stage}$
  - Number of butterflies/block = $2^{stage-1}$

FFT Implementation



Example: 8 point FFT

(1) Number of stages:

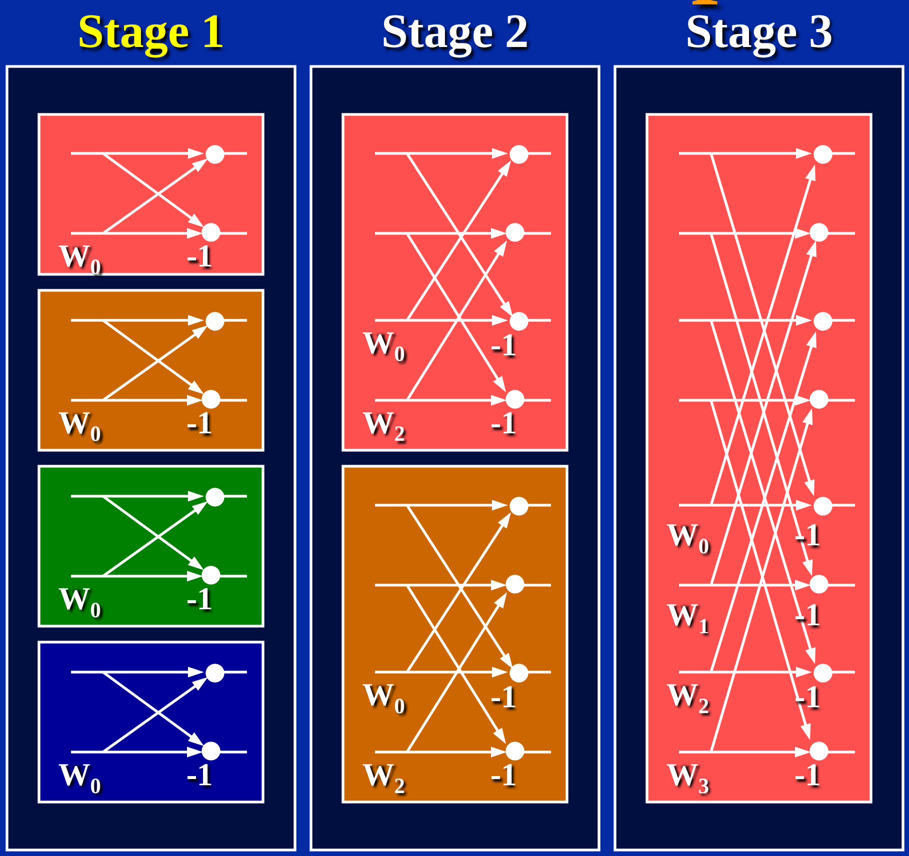
- $N_{stages} = 3$

(2) Blocks/stage:

- Stage 1: $N_{blocks} = 4$
- Stage 2: $N_{blocks} = 2$
- Stage 3: $N_{blocks} = 1$

- Decimation in time FFT:
  - Number of stages = $\log_2 N$
  - Number of blocks/stage = $N/2^{stage}$
  - Number of butterflies/block = $2^{stage-1}$

FFT Implementation



Example: 8 point FFT

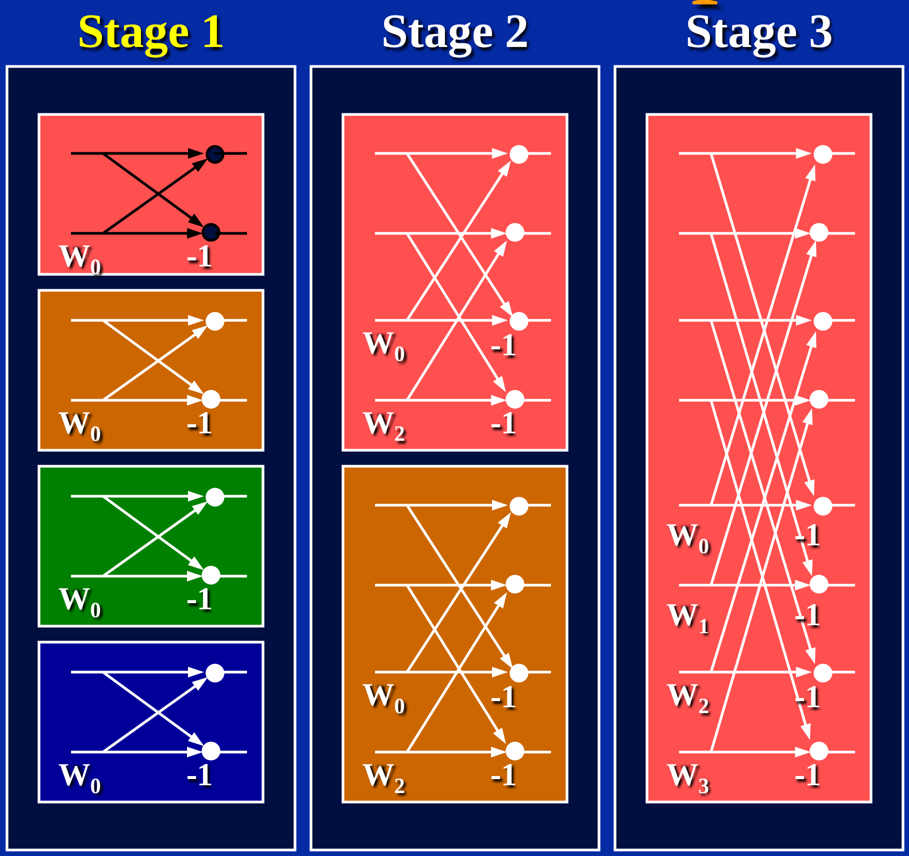
(1) Number of stages:
- $N_{stages} = 3$

(2) Blocks/stage:
- Stage 1: $N_{blocks} = 4$
- Stage 2: $N_{blocks} = 2$
- Stage 3: $N_{blocks} = 1$

(3) B'flies/block:
- Stage 1:

- **Decimation in time FFT:**
  - Number of stages = $\log_2 N$
  - Number of blocks/stage = $N/2^{stage}$
  - Number of butterflies/block = $2^{stage-1}$

FFT Implementation



Example: 8 point FFT

(1) Number of stages:

- $N_{stages} = 3$

(2) Blocks/stage:

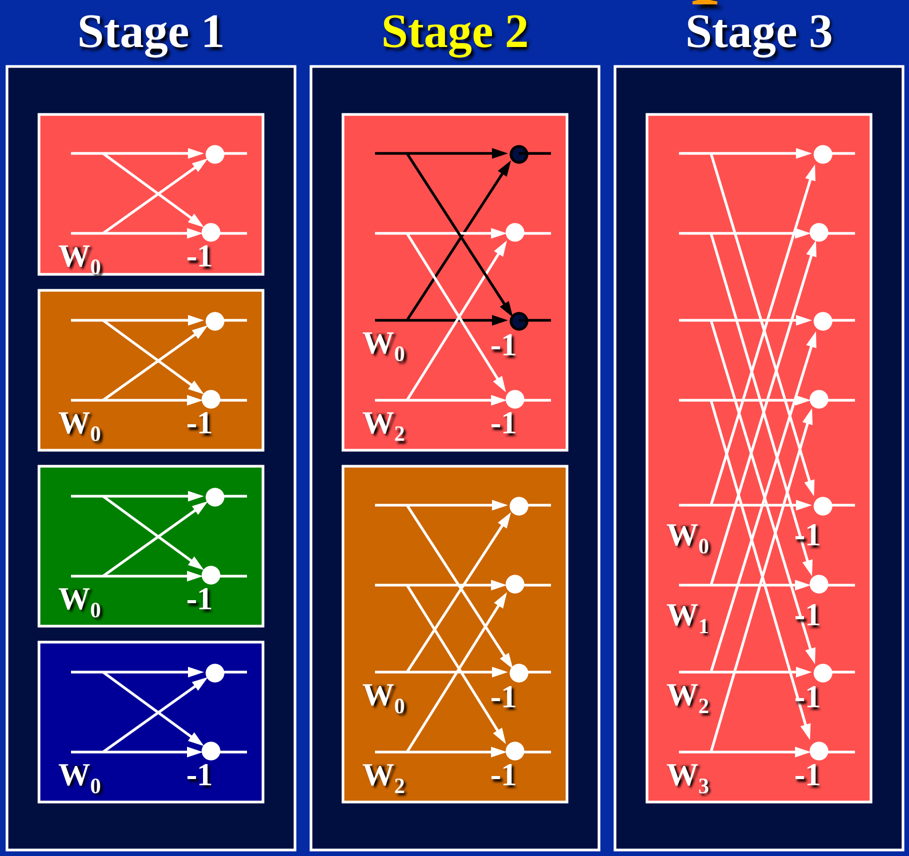
- Stage 1: $N_{blocks} = 4$
- Stage 2: $N_{blocks} = 2$
- Stage 3: $N_{blocks} = 1$

(3) B'flies/block:

- Stage 1: $N_{btf} = 1$

- **Decimation in time FFT:**
  - Number of stages = $\log_2 N$
  - Number of blocks/stage = $N/2^{stage}$
  - Number of butterflies/block = $2^{stage-1}$

FFT Implementation



Example: 8 point FFT

(1) Number of stages:

- $N_{stages} = 3$

(2) Blocks/stage:

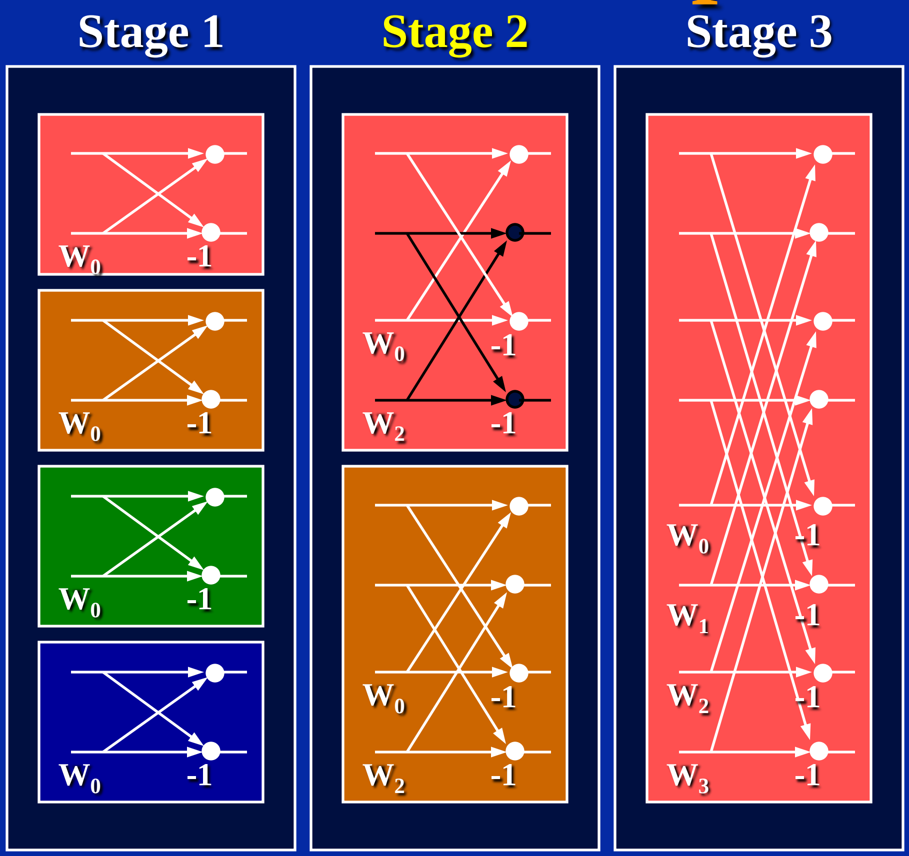
- Stage 1: $N_{blocks} = 4$
- Stage 2: $N_{blocks} = 2$
- Stage 3: $N_{blocks} = 1$

(3) B'flies/block:

- Stage 1: $N_{btf} = 1$
- Stage 2: $N_{btf} = 1$

- **Decimation in time FFT:**
  - Number of stages = $\log_2 N$
  - Number of blocks/stage = $N/2^{stage}$
  - Number of butterflies/block = $2^{stage-1}$

FFT Implementation



Example: 8 point FFT

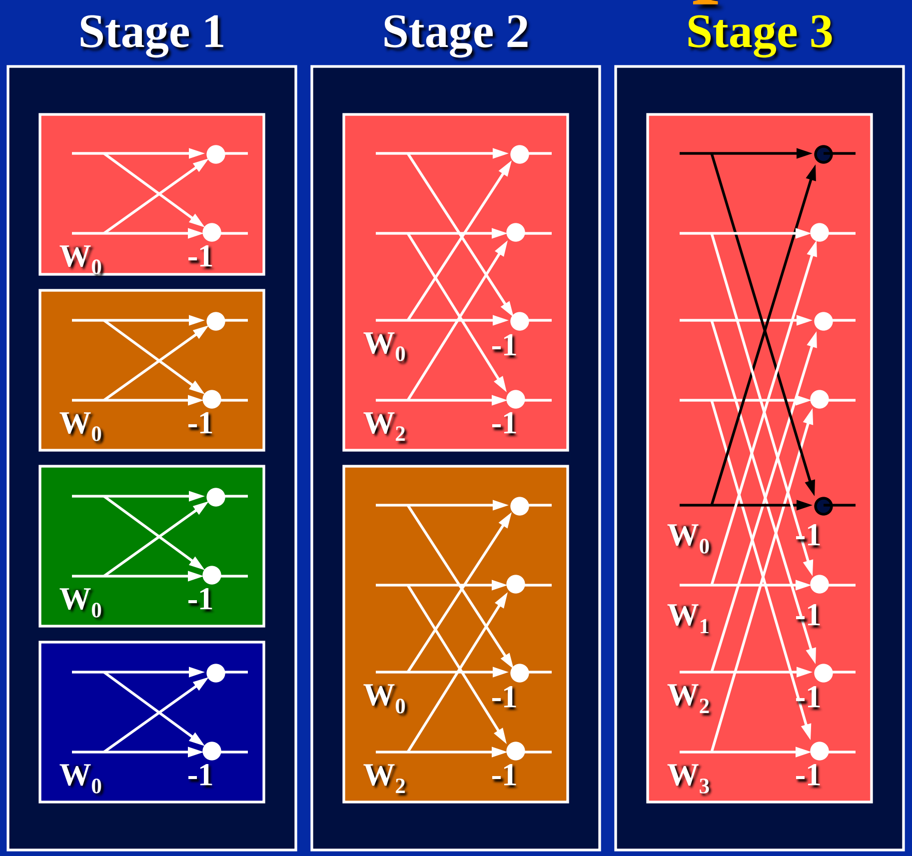
(1) Number of stages:
- $N_{stages} = 3$

(2) Blocks/stage:
- Stage 1: $N_{blocks} = 4$
- Stage 2: $N_{blocks} = 2$
- Stage 3: $N_{blocks} = 1$

(3) B'flies/block:
- Stage 1: $N_{btf} = 1$
- Stage 2: $N_{btf} = 2$

- **Decimation in time FFT:**
  - Number of stages = $\log_2 N$
  - Number of blocks/stage = $N/2^{stage}$
  - Number of butterflies/block = $2^{stage-1}$

FFT Implementation



Example: 8 point FFT

(1) Number of stages:

- $N_{stages} = 3$

(2) Blocks/stage:

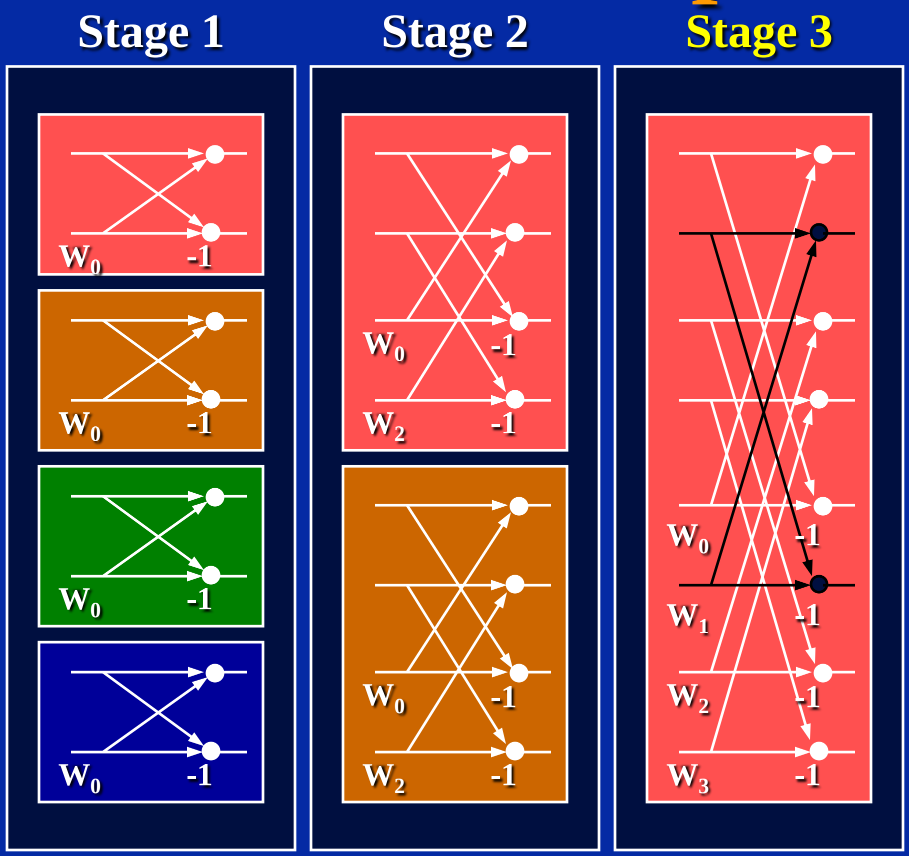
- Stage 1: $N_{blocks} = 4$
- Stage 2: $N_{blocks} = 2$
- Stage 3: $N_{blocks} = 1$

(3) B'flies/block:

- Stage 1: $N_{btf} = 1$
- Stage 2: $N_{btf} = 2$
- Stage 3: $N_{btf} = 1$

- Decimation in time FFT:
  - Number of stages = $\log_2 N$
  - Number of blocks/stage = $N/2^{stage}$
  - Number of butterflies/block = $2^{stage-1}$

FFT Implementation



Example: 8 point FFT

(1) Number of stages:

- $N_{stages} = 3$

(2) Blocks/stage:

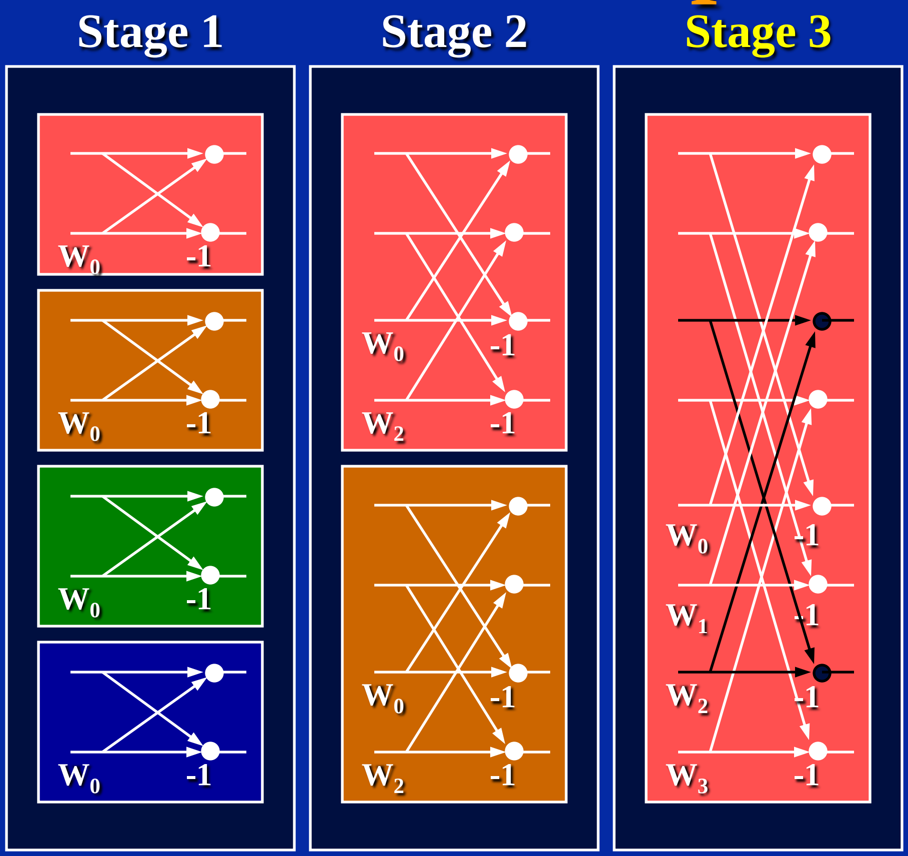
- Stage 1: $N_{blocks} = 4$
- Stage 2: $N_{blocks} = 2$
- Stage 3: $N_{blocks} = 1$

(3) B'flies/block:

- Stage 1: $N_{btf} = 1$
- Stage 2: $N_{btf} = 2$
- Stage 3: $N_{btf} = 2$

- **Decimation in time FFT:**
  - Number of stages = $\log_2 N$
  - Number of blocks/stage = $N/2^{stage}$
  - Number of butterflies/block = $2^{stage-1}$

FFT Implementation



Example: 8 point FFT

(1) Number of stages:

- $N_{stages} = 3$

(2) Blocks/stage:

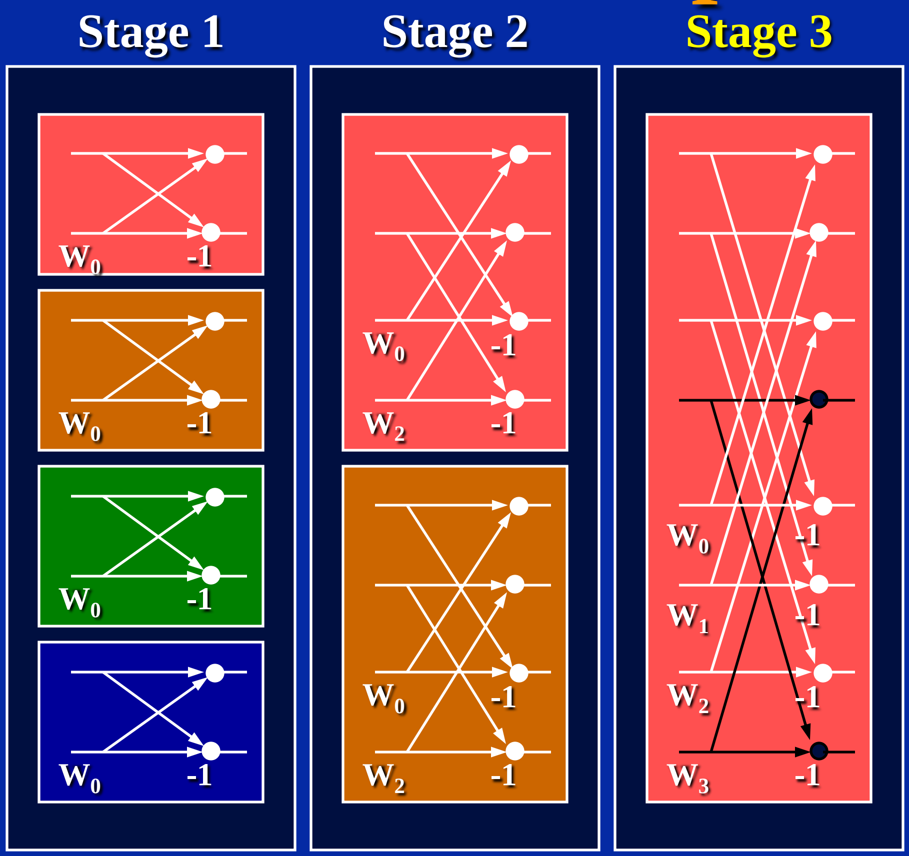
- Stage 1: $N_{blocks} = 4$
- Stage 2: $N_{blocks} = 2$
- Stage 3: $N_{blocks} = 1$

(3) B'flies/block:

- Stage 1: $N_{btf} = 1$
- Stage 2: $N_{btf} = 2$
- Stage 3: $N_{btf} = 3$

- **Decimation in time FFT:**
  - Number of stages = $\log_2 N$
  - Number of blocks/stage = $N/2^{stage}$
  - Number of butterflies/block = $2^{stage-1}$

FFT Implementation



Example: 8 point FFT

(1) Number of stages:

- $N_{stages} = 3$

(2) Blocks/stage:

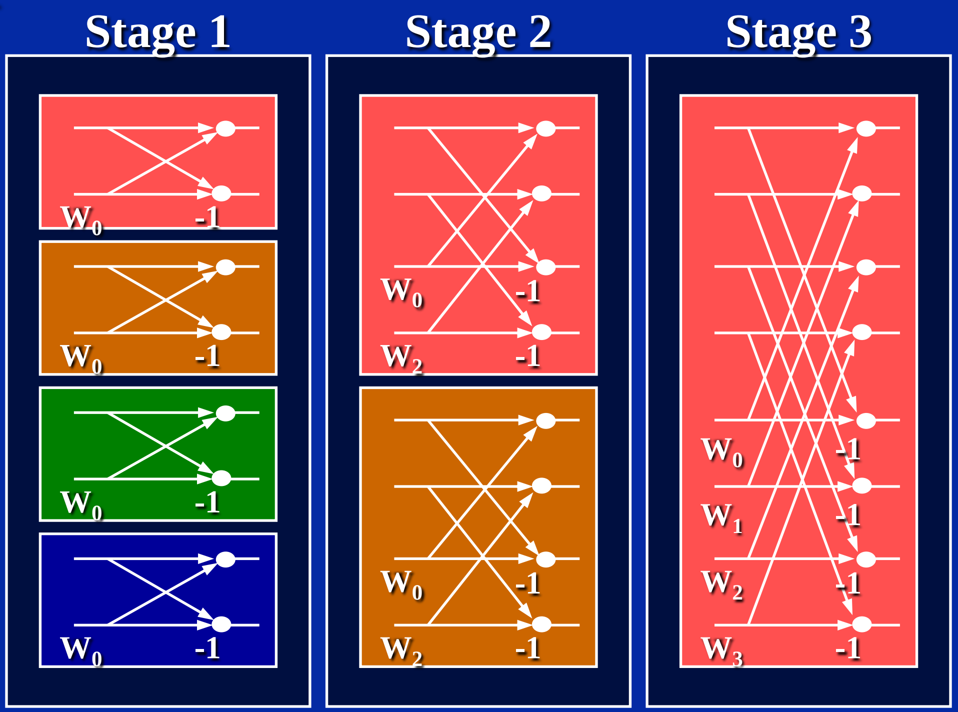
- Stage 1: $N_{blocks} = 4$
- Stage 2: $N_{blocks} = 2$
- Stage 3: $N_{blocks} = 1$

(3) B'flies/block:

- Stage 1: $N_{btf} = 1$
- Stage 2: $N_{btf} = 2$
- Stage 3: $N_{btf} = 4$

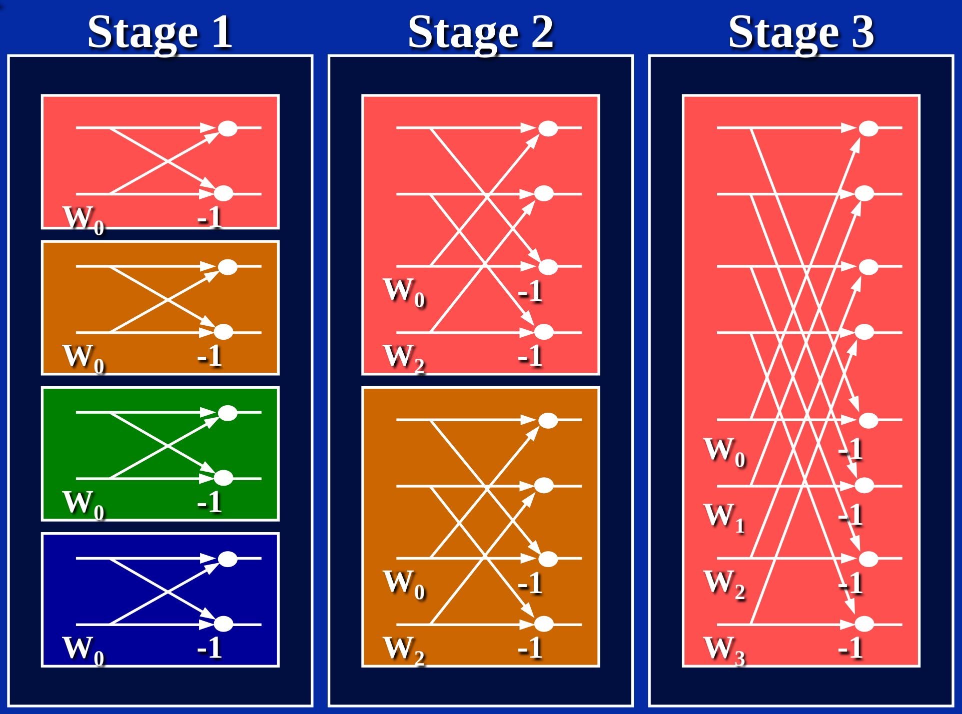
- **Decimation in time FFT:**
  - Number of stages = $\log_2 N$
  - Number of blocks/stage = $N/2^{stage}$
  - Number of butterflies/block = $2^{stage-1}$

FFT Implementation



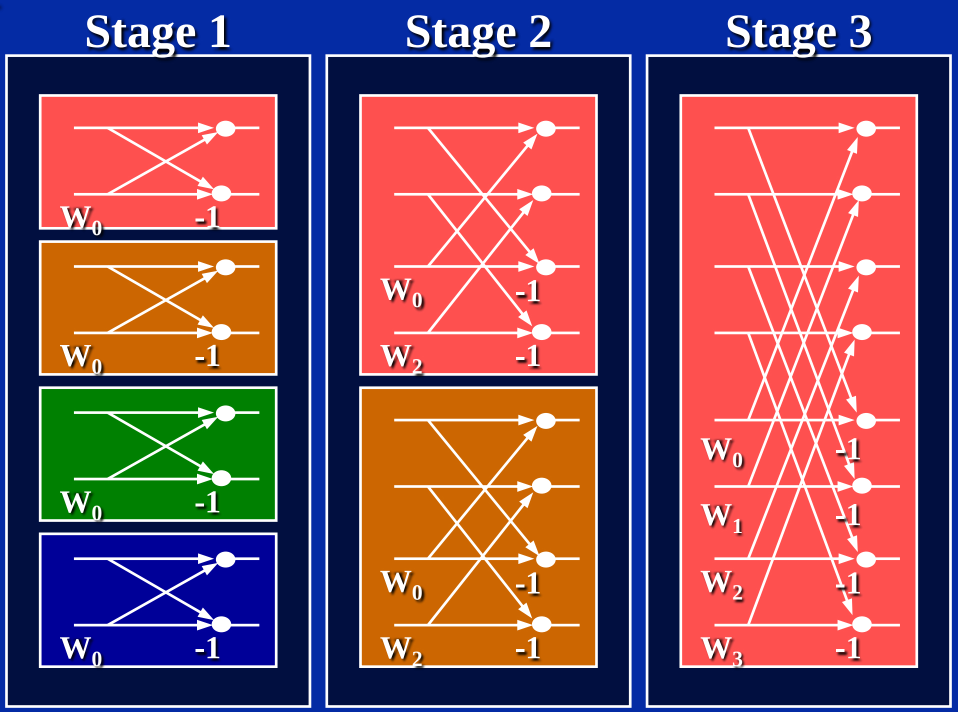
	Stage 1	Stage 2	Stage 3
Start Index	0	0	0
Input Index	1	2	4
Twiddle Factor Index	N/2 = 4		

FFT Implementation



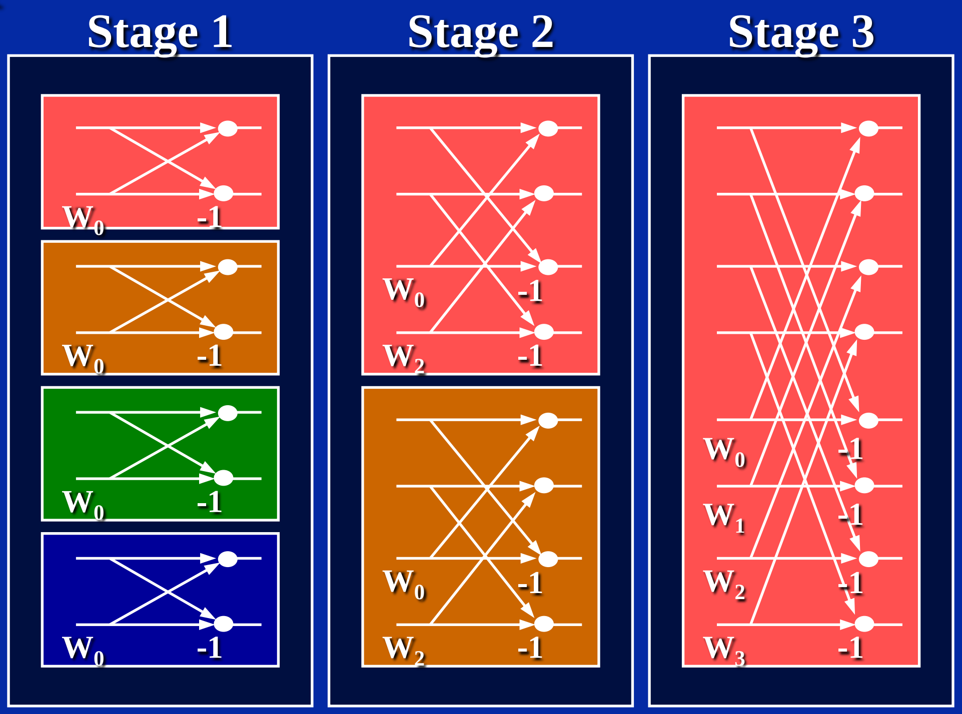
Start Index	**0**	**0**	**0**
Input Index	**1**	**2**	**4**
Twiddle Factor Index	**N/2 = 4**	**4/2 = 2**	

FFT Implementation



	Stage 1	Stage 2	Stage 3
Start Index	0	0	0
Input Index	1	2	4
Twiddle Factor Index	N/2 = 4	4/2 = 2	2/2 = 1

FFT Implementation



	Stage 1	Stage 2	Stage 3
Start Index	0	0	0
Input Index	1	2	4
Twiddle Factor Index	N/2 = 4	4/2 = 2	2/2 = 1
Indicies Used	W_0	W_0, W_2	W_0, W_1, W_2, W_3

FFT Implementation

- The most important aspect of converting the FFT diagram to 'C' code is to calculate the upper and lower indices of each butterfly:

```
GS = N/4;               /* Group step initial value */
step = 1;               /* Initial value */
NB = N/2;               /* NB is a constant */

for (k=N; k>1; k>>1)                        /* Repeat this loop for each stage */
{
    for (j=0; j<N; j+=GS)                   /* Repeat this loop for each block */
    {
        for (n=j; n<(j+GS-1); n+=step)  /* Repeat this loop for each butterfly */
        {
            upperindex = n;
            lowerindex = n+step;
        }
    }
    /* Change the GS and step for the next stage */

    GS = GS << 1;
    step = step << 1;
}
```

FFT Implementation

- **How to declare and access twiddle factors:**

(1) How to declare the twiddle factors:

```
struct {
    short real;  // 32767 * cos (2*pi*n) -> Q15 format
    short imag;  // 32767 * sin (2*pi*n) -> Q15 format
} w[] = { 32767,0,
          32767,-201,
           ...
         };
```

(2) How to access the twiddle factors:

```
short temp_real, temp_imag;

temp_real = w[i].real;
temp_imag = w[i].imag;
```