FFT

- ***Karl Friedrich Gauss (1777-1855)***
- ***J. W. Cooley and J. W. Tukey, 1965***
- ***Evaluation & Multiplication of Polynomials I***
- ***Evaluation & Multiplication of Polynomials II***
- ***Fast Fourier Transforms***
- ***Interpolation/splines/FEM etc***

FTT: $x_k = \omega_N^k = e^{2\pi i k/N}$

$$y_k \equiv \mathcal{FT}_N[a_n] = \sum_{n=0}^{N-1} (\omega_N^k)^n a_n = \sum_{n=0}^{N-1} e^{i2\pi nk/N} a_n$$

The trick: $n = n_0 + 2\, n_1 + 2^2\, n_2 + \ldots + 2^p\, n_p$

$$\omega_N^n = \omega_N^{n_0}\, \omega_{N/2}^{n_1}\, \omega_{N/4}^{n_2} \cdots \omega_2^{n_p}$$

$$\sum_n \omega_N^{nk} = \sum_{n_0=0,1} \omega_N^{n_0 k} \sum_{n_1=0,1} \omega_{N/2}^{n_1 k} \cdots \sum_{n_p=0,1} \omega_2^{n_p k}$$

HIGH BIT FIRST

$$y_k \equiv \mathcal{FT}_N[a_n] = \sum_{n=0}^{N-1} (\omega_N^k)^n a_n = \sum_{n=0}^{N-1} e^{i2\pi nk/N} a_n$$

spit polynomial into low/high pieces:

$$y_k = \sum_{n=0}^{N/2-1} e^{i2\pi nk/N} a_n + e^{i\pi k} \sum_{n=0}^{N/2-1} e^{i2\pi nk/N} a_{n+N/2}$$

➔ One N = Two N/2 Fourier transforms

even k $\quad y_{2\tilde{k}} = \sum_{n=0}^{N/2-1} e^{i2\pi n\tilde{k}/(N/2)} [a_n + a_{n+N/2}]$

odd k $\quad y_{2\tilde{k}+1} = \sum_{n=0}^{N/2-1} e^{i2\pi n\tilde{k}/(N/2)} \omega_N^n [a_n - a_{n+N/2}]$

LOW BIT FIRST

$$y_k \equiv \mathcal{FT}_N[a_n] = \sum_{n=0}^{N-1} (\omega_N^k)^n a_n = \sum_{n=0}^{N-1} e^{i2\pi nk/N} a_n$$

spit polynomial into even/odd pieces:

$$y_k = \sum_{n=0}^{N/2-1} e^{i2\pi nk/(N/2)} a_{2n} + \omega_N^k \sum_{n=0}^{N/2-1} e^{i2\pi nk/(N/2)} a_{2n+1}$$

➔ One N = Two N/2 Fourier transforms

low k

$$y_k = \sum_{n=0}^{N/2-1} e^{i2\pi nk/(N/2)} [a_n + \omega_N^k a_{n+N/2}]$$

high k

$$y_{k+N/2} = \sum_{n=0}^{N/2-1} e^{i2\pi nk/(N/2)} [a_n - \omega_N^k a_{n+N/2}]$$

THUS T(N) = 2 T(N/2) + C_0 N: T(N) » NLOG(N)

$$y_{2k} = \mathcal{FT}_{N/2}[\, a_n + a_{n+N/2} \,]$$

$$y_{2k+1} = \mathcal{FT}_{N/2}[\, \omega_N^n (a_n - a_{n+N/2}) \,]$$

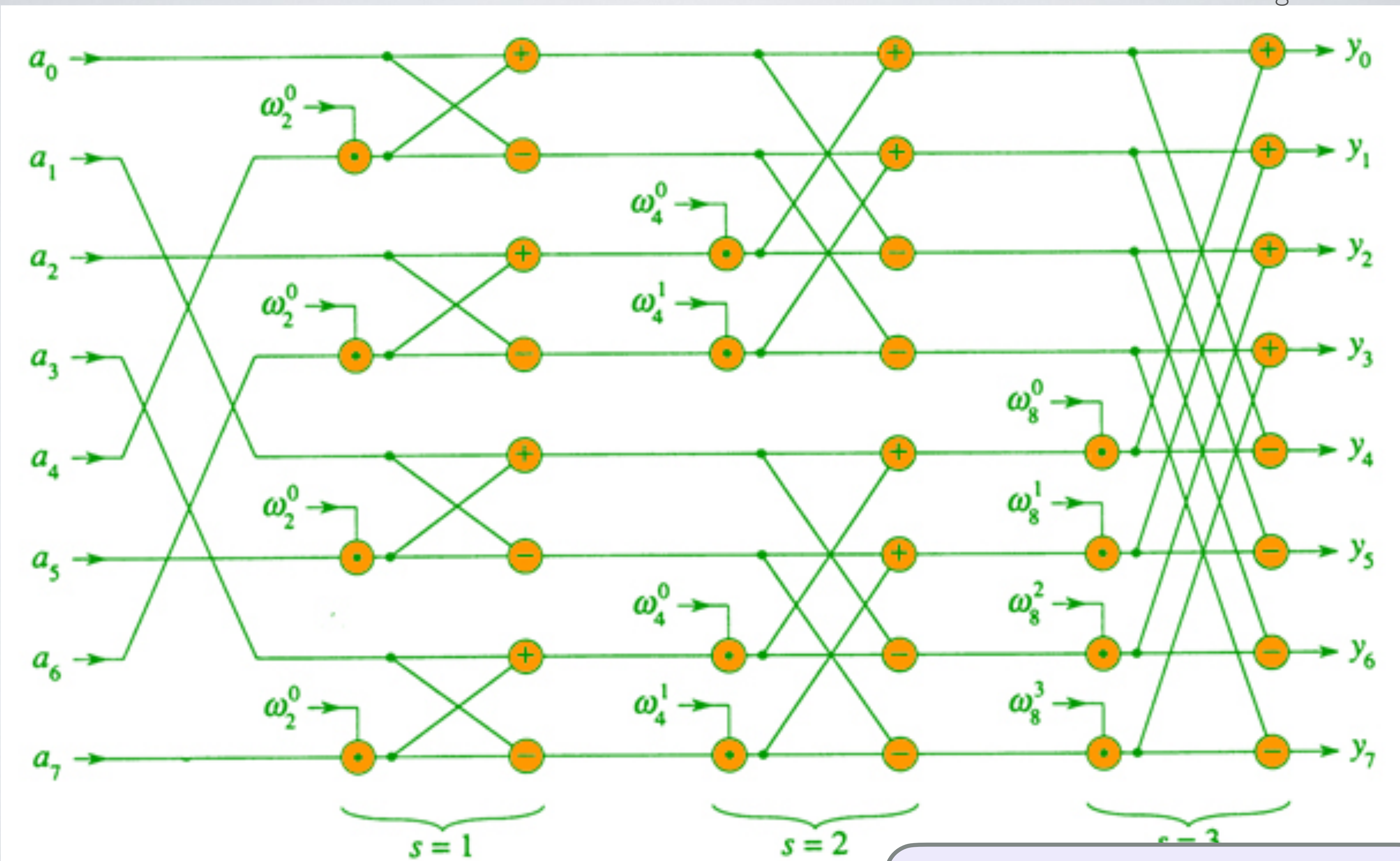
OR

$$y_k = \mathcal{FT}_{N/2}[a_{2n} + \omega_N^k a_{2n+1}]$$ k = 0…N/2 -1

$$y_{k+N/2} = \mathcal{FT}_{N/2}[a_{2n} - \omega_N^k a_{2n+1}]$$ k = N/2…N -1

BUTTERFLY ITERATIVE SOLUTION

bit reverse | 2 blocking | 4 blocking | 8 blocking



$$\omega_N^k = (e^{2\pi i/N})^k = e^{2\pi i k/N}$$

INDIXING INSIDE BLOCKS AND BETWEEN BLOCKS

Nblocks = N/Bsize

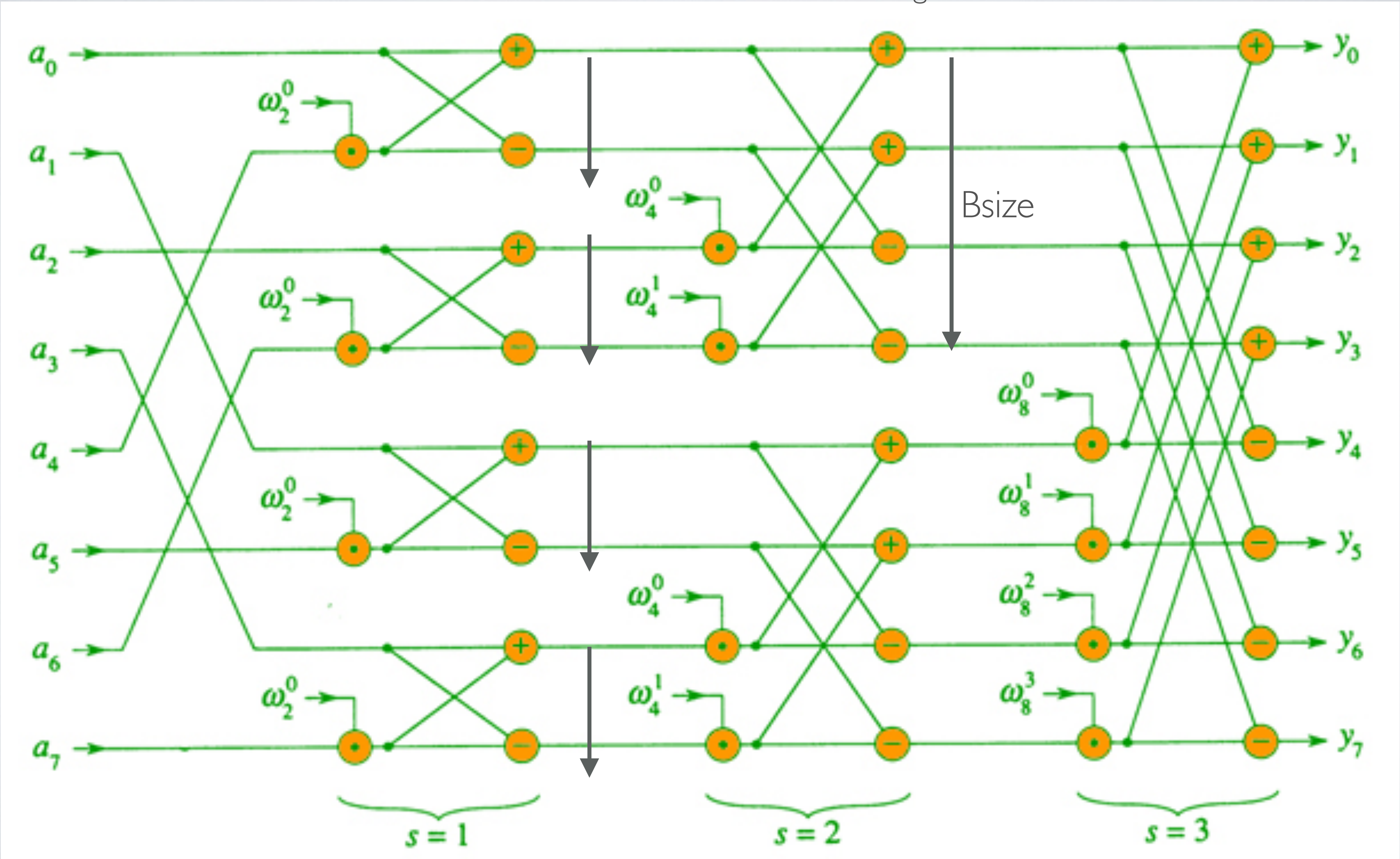
$$\omega_N^k = (e^{2\pi i/N})^k = e^{2\pi i k/N}$$

bit reverse

Bsize =2 blocking

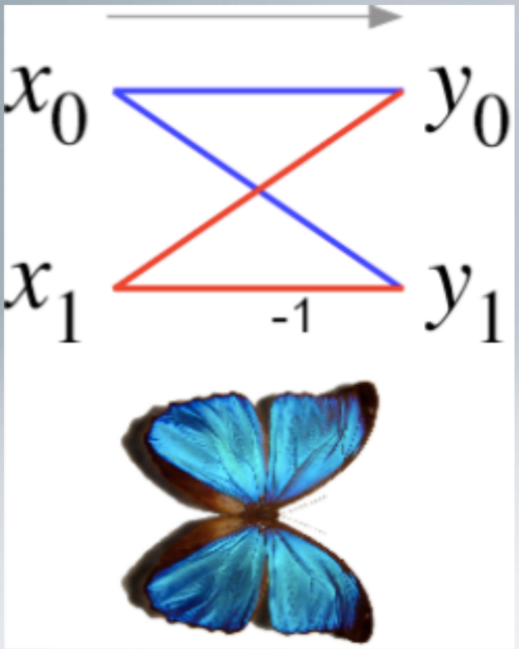
Bsize = 4 blocking

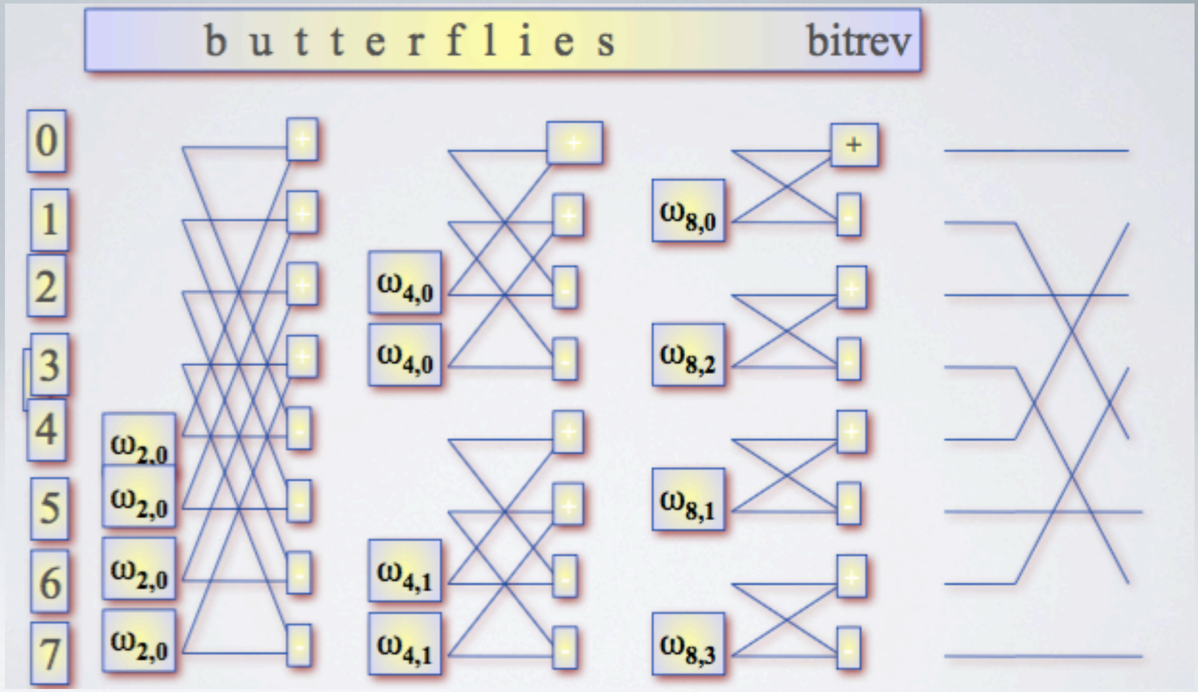
Bsize =8 blocking



FFT BUTTERFLY NETWORK

Left to Right High low to even odd $\omega_{N,k} = e^{2\pi i k/N}$





For Fun Some FFT Coding Style

```
void FFT(Complex * Ftilde, Complex * F, Complex * omega, int N)
{
   int log2n = log2(N);
   Complex high, low;
   int Bsize = 0;
   int Nblocks = 0;

   for(unsigned int x = 0; x < N; x++)
     Ftilde[reverseBits( x, N)] = F[x];

   for(int s = 0; s < log2n; s++)
     {
       Bsize = pow(2,s+1);
       Nblocks  = N/Bsize;

       for(int nb = 0;nb < Nblocks ; nb++)                    // slow to next
Nblocks
        for(int pairs  = 0; pairs <  Bsize/2; pairs++)        // fast index over
nomber of  Pairs = Bsize/2
          {
           int  k = pairs +  nb * Bsize;                  //low pairs index : {x
high = low + Bsize/2
           low = Ftilde[k];
           high =  omega[Nblocks*pairs]*Ftilde[k
+  Bsize/2];    //(omega_Bsize)^pairs = (omega_N^(Nblocks*pairs)
           Ftilde[k] = low + high;
           Ftilde[k + Bsize/2]  = low -  high;
          }
     }
}
```

INDEXING: (LIKE CUDEDX = BLOCKIDX.X * BLOCKDIM.X + THREADIDX.X;)

Example: Increment Array Elements

Increment N-element vector a by scalar b

Let’s assume N=16, blockDim=4 -> 4 blocks

blockIdx.x=0	blockIdx.x=1	blockIdx.x=2	blockIdx.x=3
blockDim.x=4	blockDim.x=4	blockDim.x=4	blockDim.x=4
threadIdx.x=0,1,2,3	threadIdx.x=0,1,2,3	threadIdx.x=0,1,2,3	threadIdx.x=0,1,2,3
idx=0,1,2,3	idx=4,5,6,7	idx=8,9,10,11	idx=12,13,14,15

```
int idx = blockDim.x * blockId.x + threadIdx.x;
```
will map from local index threadIdx to global index

Common Pattern!

NB: blockDim should be >= 32 in real code, this is just an example