FFT (Fast Fourier Transform)

2005年12月7日
北海道大学　大学院情報科学研究科
複合情報学専攻
吉川　浩

FFT(高速フーリエ変換)

- 説明の流れ
  - FFTとは？
    - FFTとは何かについて簡単に説明する
  - FFTのアイデア
    - 計算の高速化の工夫について雰囲気を理解してもらう
  - FFTのハードウェア化について
    - ハードウェア化の方法と特徴について
  - まとめ

FFT(高速フーリエ変換)とは？

- FFTは離散フーリエ変換(DFT)を高速に計算する手法
  - DFTとは、時間軸上でサンプリング（離散化）して得られたデータ列 $\{x_k\}$ に対するフーリエ変換。結果も離散的$\{X_k\}$
    - サンプリングデータ列

      $x_0=f(0),\ x_1=f(\Delta t),\ x_2=f(2\Delta t)\ ,\ x_3=f(3\Delta t),\ \ldots\ ,\ x_{N\text{-}1}=f((N\text{-}1)\Delta t)$

      に対して：

      離散フーリエ変換(DFT)の式 $$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N} \quad (k=1,\ 2,\ \ldots\ N\text{-}1)$$

    - 積分計算の必要はなく、積と和の演算だけで計算できる
  - 効率よくDFTを計算するためには？
    - N を上手に選ぶと同じ計算が何度も出てくるようになる。
    - 更に計算順序にも工夫をすると計算量を大幅に削減できる。 → FFT

FFTのアイデア

ここではFFTの基本を理解してもらうことが目的なので
N=4 についてのみ説明し、雰囲気を掴んでもらう。

FFTのアイデア(1)

- サンプル数 N=4 で説明する。このときのDFTは次の式で表すことができる

$$X_k = \sum_{n=0}^{3} x_n W^{kn} \quad (k = 0 \sim 3)$$

(ただし$W = e^{-i2\pi/N} = e^{-i\pi/2}$ とおいた)

- これを行列表示する

$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 \\ W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^9 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

ここで言いたいこと:

このままではWの累乗や W^k と x_n との積和を大量に計算しなければならないので大変だ！

FFTのアイデア(2)

- そこで、まず方程式の係数行列を簡素化する:

重要 $W = e^{-i2\pi/N}$ の周期性に着目。
$W^{k+N} = W^k, \quad W^{k+N/2} = -W^k$ という関係を使うと係数行列を簡素化できる。

$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 \\ W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^9 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & -W^0 & -W^1 \\ W^0 & -W^0 & W^0 & -W^0 \\ W^0 & -W^1 & -W^0 & W^1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

- 2行目と3行目を入れ替えると規則性が見える・・・

$$\begin{bmatrix} X_0 \\ X_2 \\ X_1 \\ X_3 \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & -W^0 & W^0 & -W^0 \\ W^0 & W^1 & -W^0 & -W^1 \\ W^0 & -W^1 & -W^0 & W^1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_2 \\ x_1 \\ x_3 \end{bmatrix}$$

次ページへ

FFTのアイデア(3)

- 実は係数行列は分解できることが分かる:

$$\begin{bmatrix} X_0 \\ X_2 \\ X_1 \\ X_3 \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & -W^0 & W^0 & -W^0 \\ W^0 & W^1 & -W^0 & -W^1 \\ W^0 & -W^1 & -W^0 & W^1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} A & A \\ B & -B \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$A = \begin{bmatrix} W^0 & W^0 \\ W^0 & -W^0 \end{bmatrix}, \quad B = \begin{bmatrix} W^0 & W^1 \\ W^0 & -W^1 \end{bmatrix}$$

- 分解された係数行列は次のような積に変形できる:

$$\begin{bmatrix} X_0 \\ X_2 \\ X_1 \\ X_3 \end{bmatrix} = \begin{bmatrix} A & O \\ O & B \end{bmatrix} \begin{bmatrix} I & I \\ I & -I \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

(I は単位行列、O は零行列)

FFTのアイデア(4)

- 更に次のように計算を進めると:

$$\begin{bmatrix} X_0 \\ X_2 \\ X_1 \\ X_3 \end{bmatrix} = \underbrace{\begin{bmatrix} A & O \\ O & B \end{bmatrix}\begin{bmatrix} I & I \\ I & -I \end{bmatrix}\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{\text{これを先に計算}} = \begin{bmatrix} A & O \\ O & B \end{bmatrix}\begin{bmatrix} x_0 + x_2 \\ x_1 + x_3 \\ x_0 - x_2 \\ x_1 - x_3 \end{bmatrix} = \underbrace{\begin{bmatrix} \boxed{W^0(x_0 + x_2)} + \boxed{W^0(x_1 + x_3)} \\ \boxed{W^0(x_0 + x_2)} - \boxed{W^0(x_1 + x_3)} \\ \boxed{W^0(x_0 - x_2)} + \boxed{W^1(x_1 - x_3)} \\ \boxed{W^0(x_0 - x_2)} - \boxed{W^1(x_1 - x_3)} \end{bmatrix}}_{\text{同じ項が沢山でてくる}}$$

- 実質的な計算は非常に少なくて済むことがわかる
  - このように N をうまく選ぶと計算量を大幅に削減できる
  - 一般にFFTの N は $2^m(m=1,2,\ldots)$ となるように選ぶ

FFTをハードウェア化する

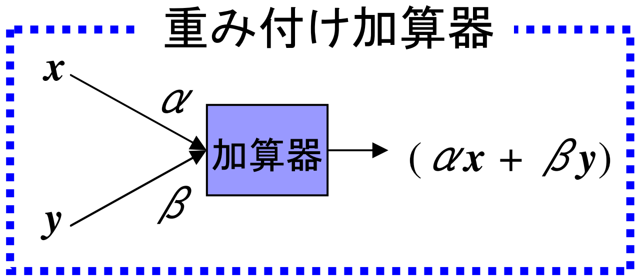
ここではFFTをハードウェア化するために、まず最初に計算のフローを示し、次にハードウェア化に必要な演算（バタフライ演算）について説明する。

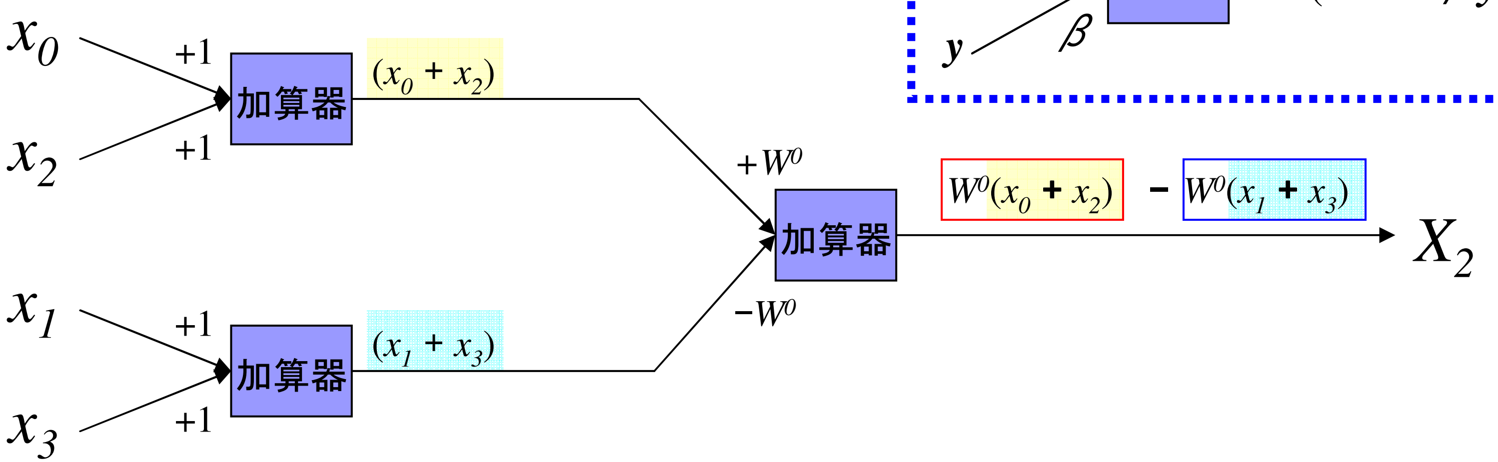
FFTをハードウェア化する(1)

- 計算フローを図示してみる

$$\begin{bmatrix} X_0 \\ X_2 \\ X_1 \\ X_3 \end{bmatrix} = \begin{bmatrix} \boxed{W^0(x_0+x_2)} + \boxed{W^0(x_1+x_3)} \\ \boxed{W^0(x_0+x_2)} - \boxed{W^0(x_1+x_3)} \\ \boxed{W^0(x_0-x_2)} + \boxed{W^1(x_1-x_3)} \\ \boxed{W^0(x_0-x_2)} - \boxed{W^1(x_1-x_3)} \end{bmatrix}$$

  - 例えば X_2 を求めるには？

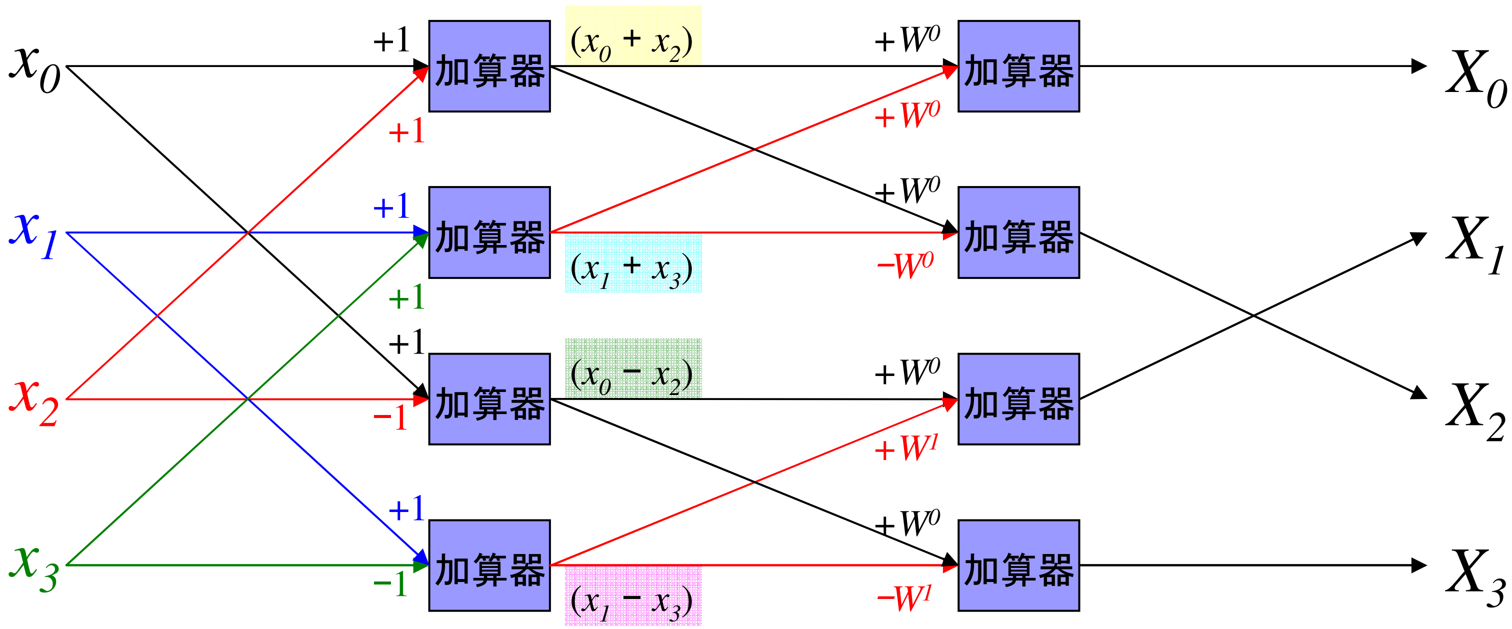




FFTをハードウェア化する(2)

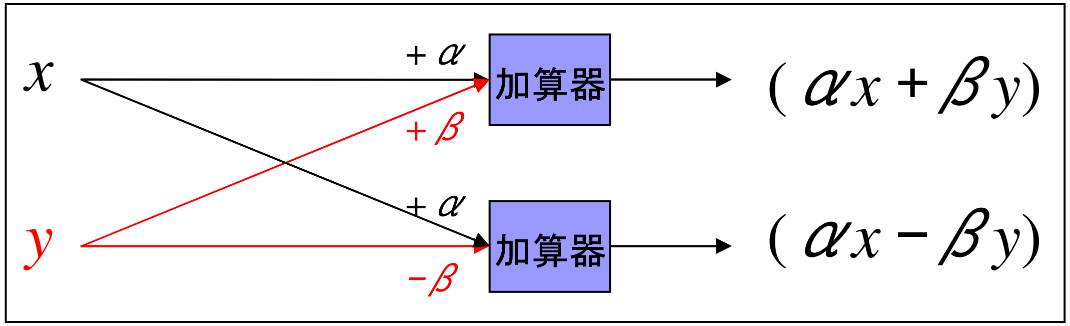
- 全ての計算フローを図示すると

$$\begin{bmatrix} X_0 \\ X_2 \\ X_1 \\ X_3 \end{bmatrix} = \begin{bmatrix} \boxed{W^0(x_0+x_2)} + \boxed{W^0(x_1+x_3)} \\ \boxed{W^0(x_0+x_2)} - \boxed{W^0(x_1+x_3)} \\ \boxed{W^0(x_0-x_2)} + \boxed{W^1(x_1-x_3)} \\ \boxed{W^0(x_0-x_2)} - \boxed{W^1(x_1-x_3)} \end{bmatrix}$$



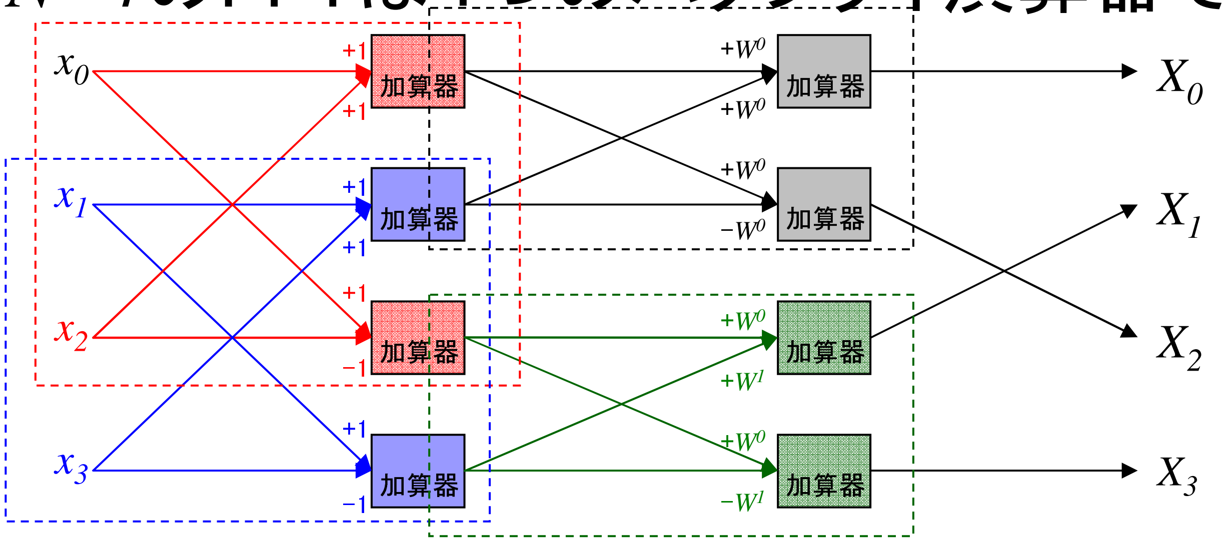
FFTをハードウェア化する(3)

- バタフライ演算器(重み付け加算・減算回路)



加算回路と
乗算回路だけで
実現できる

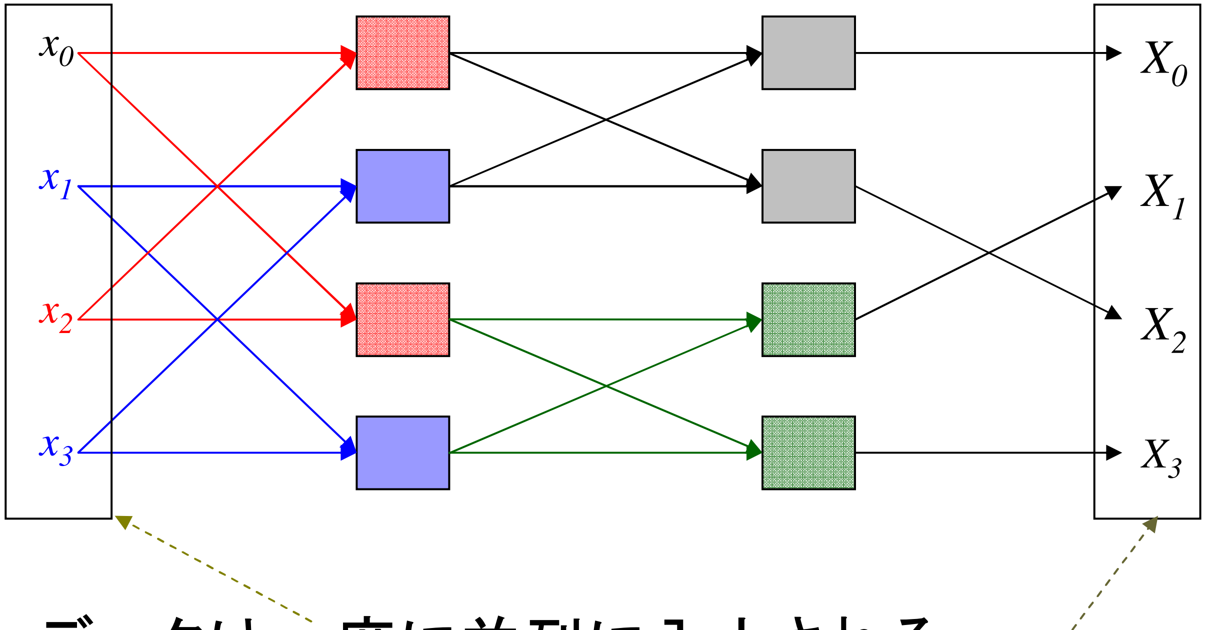
- N=4のFFTは4つのバタフライ演算器で実現可能



一般に
$N=2^m$のときの
バタフライ演算器
の個数は$m \cdot 2^{m-1}$個

FFTをハードウェア化する(4)

- ハード化されたFFTの特徴



  - データは一度に並列に入力される
  - 計算結果も一度に並列に出力される
  - 構成回路はバタフライ演算回路のみ
  - 並列処理である(データが左から右へ一斉に流れる)
  - 処理時間 = 回路の伝播遅延(propagation delay)

まとめ

- FFTはDFTを高速に行なう手法
  - サンプル数 N を 2^m に制限することにより高速化
- FFTはバタフライ演算だけで構成される
  - バタフライ演算は重み付けの加算・減算
- FFTはハードウェア化のメリットを活かせる好例である
  - 低コストでハード化、集積化できる
    - 条件分岐がなく単純な積和計算のみ
    - バタフライ回路（積和演算回路）の繰り返し（回路のコピー）
  - ハード化により驚異的な速度で計算できる
    - データは一度に入力され、並列計算され、結果は一度に出力される
    - 計算にかかる時間は回路の伝播遅延（propagation delay）のみ
      - propagation delayが1クロック内なら1クロックで全ての計算が終了

参考文献:

- ビギナーズシリーズ「デジタルフーリエ変換」
  中村尚五　著
  東京電機大学出版局

おわり