演算機能回路

第13週

2011/12/12

泉知論

立命館大学 理工学部 電子情報デザイン学科

除算器、開平器、…

division, square-root, ...

除算

- $A \div B = Q$ 余り R
- 筆算の手順をそのまま回路化すれば実現可能

除算の筆算

- 4bit ÷ 4bit の符号無除算の例

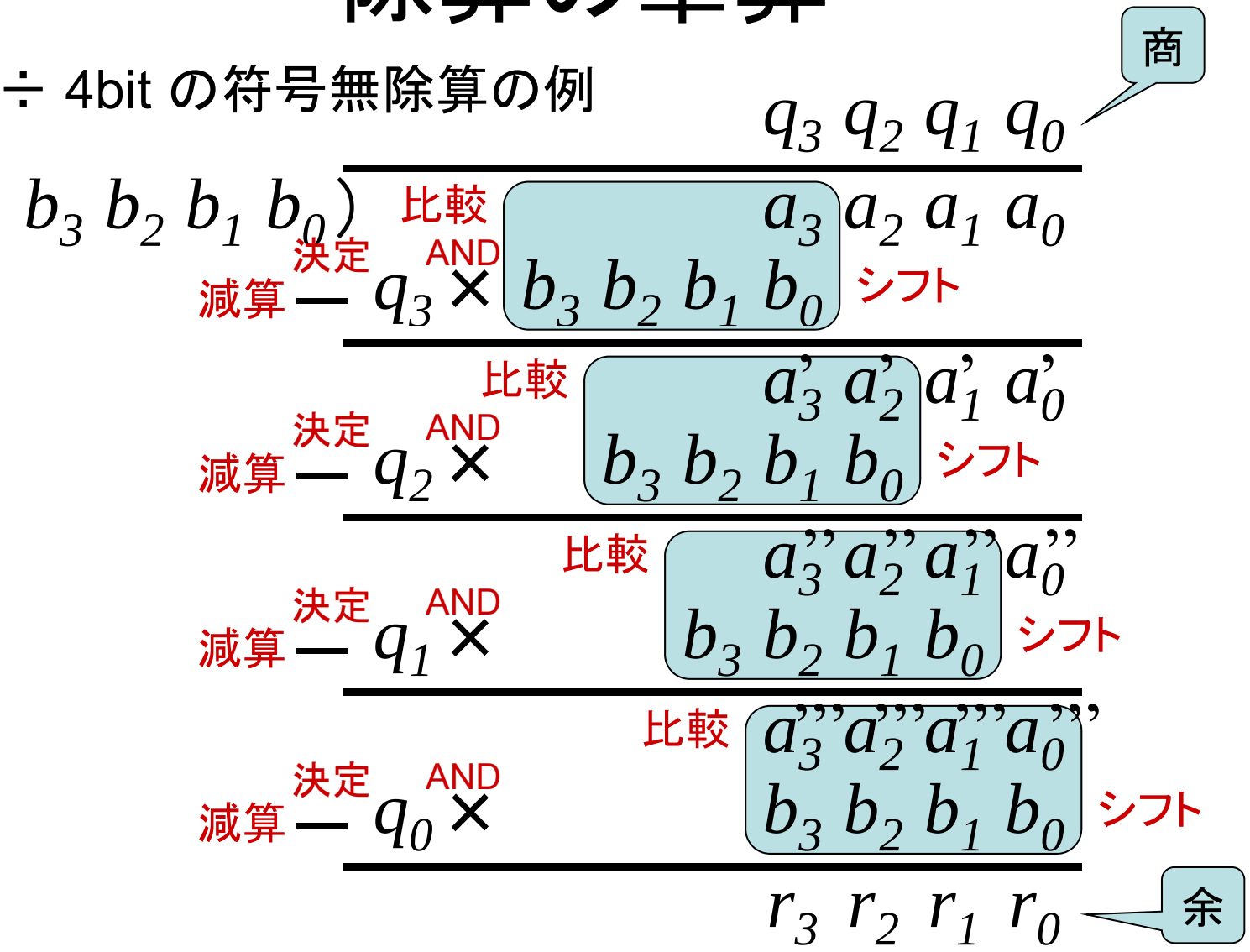
$$
\begin{array}{rl}
 & q_3\ q_2\ q_1\ q_0 \quad \text{商} \\
\hline
b_3\ b_2\ b_1\ b_0) & a_3\ a_2\ a_1\ a_0 \\
- & q_3 \times\ b_3\ b_2\ b_1\ b_0 \\
\hline
 & a'_3\ a'_2\ a'_1\ a'_0 \\
- & q_2 \times\ \ \ b_3\ b_2\ b_1\ b_0 \\
\hline
 & a''_3\ a''_2\ a''_1\ a''_0 \\
- & q_1 \times\ \ \ \ b_3\ b_2\ b_1\ b_0 \\
\hline
 & a'''_3\ a'''_2\ a'''_1\ a'''_0 \\
- & q_0 \times\ \ \ \ b_3\ b_2\ b_1\ b_0 \\
\hline
 & r_3\ r_2\ r_1\ r_0 \quad \text{余}
\end{array}
$$

演習13A

- 次の符号無し2進数の割り算をせよ。商は整数の範囲で求め、余りも求めよ。
  - ◆1111 ÷ 11
  - ◆10001001 ÷ 1101
  - ◆11000111 ÷ 1100100
  - ◆10101010 ÷ 1000
- 同様に、商は小数点第2位までもとめ、余りも求めよ。

除算の筆算

- 4bit ÷ 4bit の符号無除算の例



除算の処理手順と基本的構成法

- 以下の繰り返し
  - 除数をシフト
  - 被除数（剰余）と除数の比較
  - 商の現在の1ビットを決定
  - 被除数（剰余）から除数と商1ビットの積を減算
- 組合せ回路、繰り返し演算、パイプライン演算、いずれでも実現可能

注：比較と減算の繰り返しの順番を入れ替えることはできないので、回路をツリー状に構成するテクニックは使えない。

除算器の構成と使用について

- 具体的な回路構成については省略する。
  - みなさんは、もう、設計できるはず！
- 実際の設計現場では、ライブラリから選択することが多い。ただし、構成や特徴を理解しておかねば、良い選択はできない。
- 乗算器と同じく、チューンナップのための高度なテクニックが多数存在する。将来、本当に設計の必要に迫られたときに調査されたし。
- 現場ではむしろ除算を使わないで済む工夫が必要。

除算器削減の工夫

- 定数除算を逆数の定数乗算に置き換える
  - 例 $x \div c \quad \Rightarrow \quad x \times \frac{1}{c}$
- 式変形して除算の回数を減らす
  - 例 $\sum \frac{x_i}{n} \quad \Rightarrow \quad \frac{\sum x_i}{n}$
- 除算のない（少ない）近似式を考える

など

筆算からの演算器設計の原則

- みなさんはもう、筆算の手順があれば、どんな演算回路でも作れる！（はず？）
- これまで学んできたこと
  - ２進数（２の補数）で数を表現
  - ２進数の筆算に従って計算を回路化
  - 数式上の工夫、回路構造上の工夫
  - 組合せ回路、繰り返し演算、パイプライン演算
  - 実現のための部品
    - 基本ゲート、セレクタ、加算器(CPA, CLA, CSA)、シフタ、レジスタ、ステートマシン

加算、減算、乗算、除算、平方根、…

三角関数、指数、対数…

sin, cos, tan, exp, log, ...

より複雑な演算(関数)の実現

- 筆算が与えられた演算⇒できる！
- 単純な筆算では計算できない演算(関数)
  - sin, cos, tan, exp, log, ...
- 級数展開による多項式近似
  - sin 関数の例
- CORDIC アルゴリズム
  - 複素ベクトル操作の繰り返しで三角関数や乗除算を計算する洗練された古典的アルゴリズム
- 表を作っておいてそれを見る

級数展開を適当な次数で打ち切って近似計算

$$\sin x = \sum_{n=1}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

級数展開による実現

- 数学的に直接的な実現方法
- 乗算器、定数乗算器、加算器を駆使すれば実現可能
- 通常、組合せ回路としての実現は×。マルチサイクル、あるいは、パイプラインで実現。
- 必要精度に応じて展開の次数、ビット幅を調整可能
- しかし、そのぶん、レイテンシが増大。

表参照型演算器（単純構成）

- 複雑な関数を実現する「コロンブスの卵」的な単純な方法
- 入力値に対する出力値を、表として持っておく
- 与えられた x に対応する y を表から引いて出力する

表 → メモリで実現

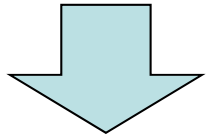
参照表

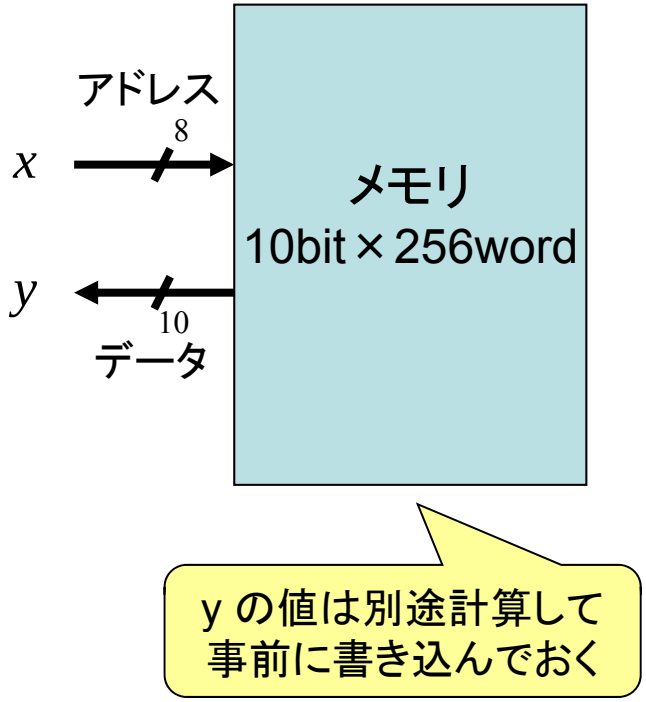
例: $y = 512 \sin(2\pi x / 128)$

x ... 8bit, y ... 10bit

- 入力値 x (n bit)
  → メモリのアドレス
- 出力値 y (m bit)
  → メモリのデータ

x	y
127	6
:	:
2	12
1	6
0	0
-1	-6
-2	-12
:	:
-127	6



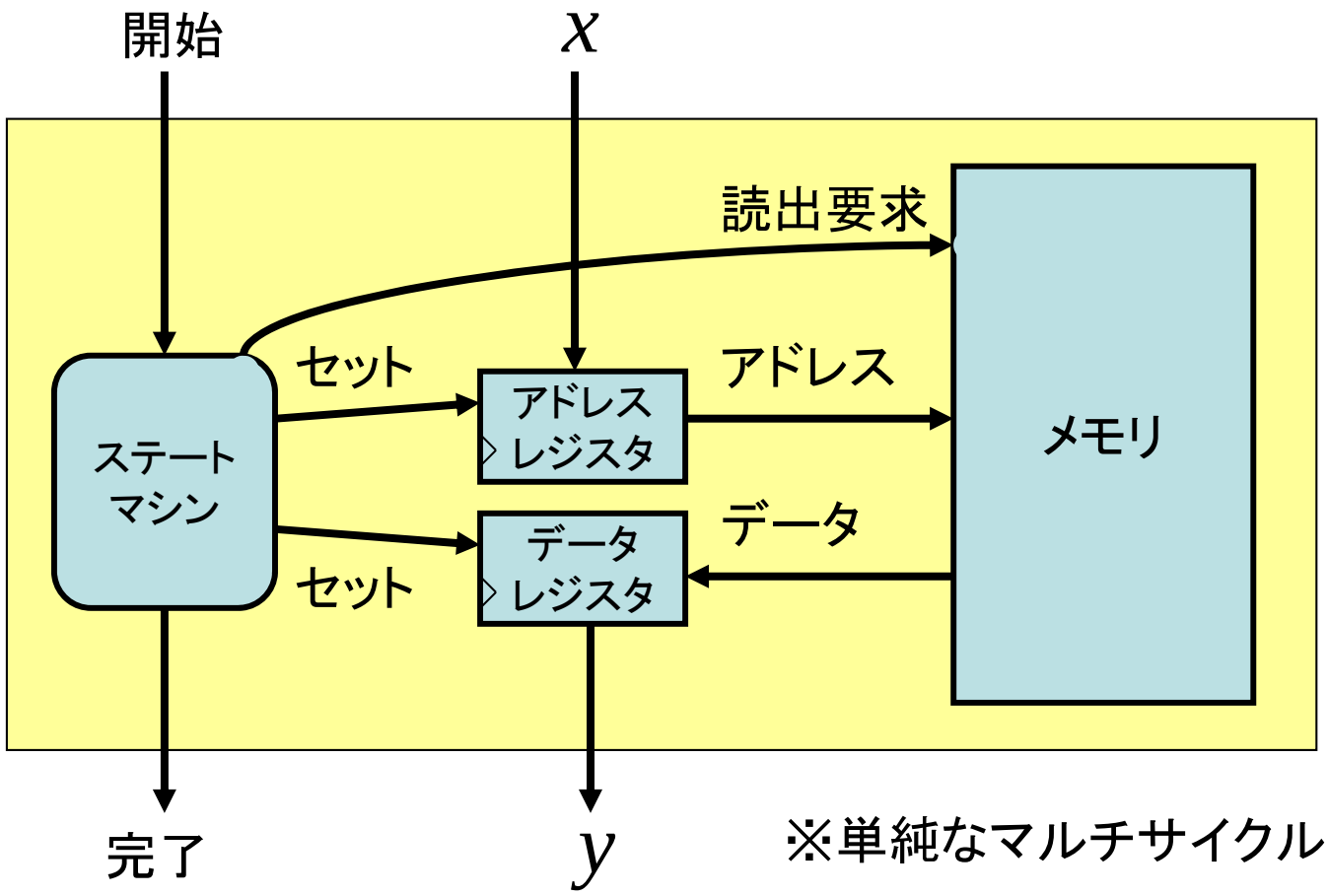


- $m \times 2^n$ bit のメモリ

表参照型演算器の構成

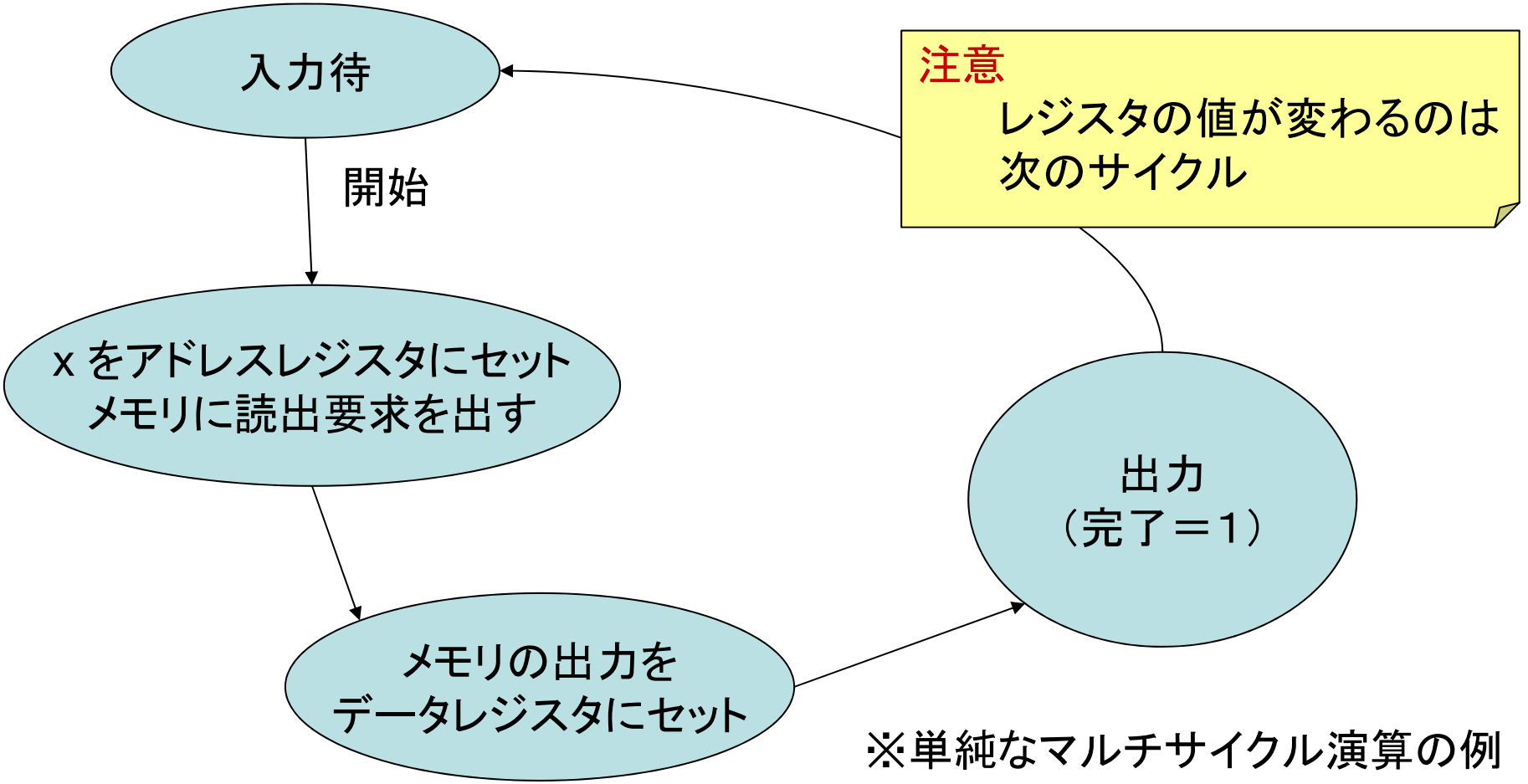
- 表を保持するメモリ(ROM)
- メモリを読み出すための制御回路
- 通常、メモリの読み書きに数サイクル必要
  - 組合せ回路による実現は不可
  - マルチサイクル演算
  - パイプライン演算(メモリがパイプライン的な読出可能であれば)

表参照型演算器の単純構成例



※単純なマルチサイクル演算の例

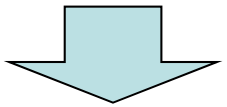
表参照型演算器の状態遷移例



※単純なマルチサイクル演算の例

表参照型演算器の改良

- 入力 n bit 、出力 m bit ⇒ $m \times 2^n$ bit のメモリ
- 範囲＆精度を上げるには膨大なメモリが必要

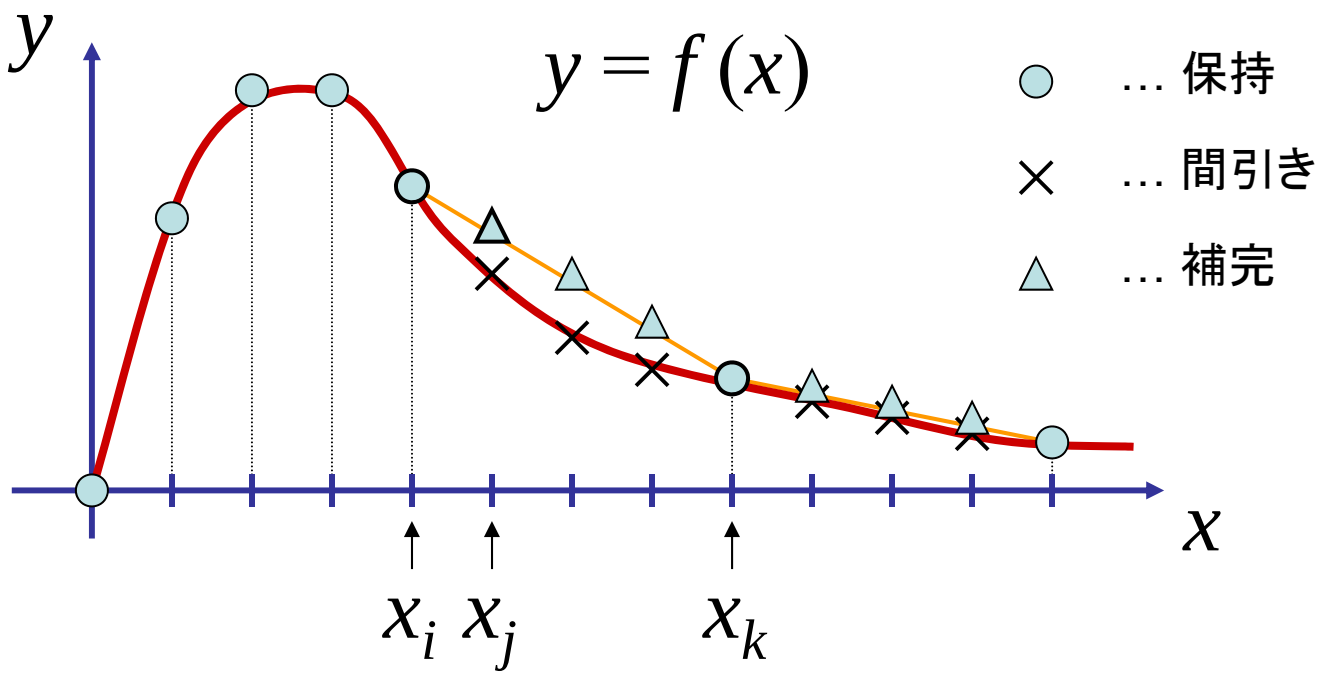


表に保存するデータを削減

- 周期性、対称性を利用 (例：sin, cos は 0～π/2 で十分)
- 間引きと補完（線形補完）
- 可変区間
  - 関数のグラフが直線的な部分⇒区間を広く
  - 関数のグラフが曲線的な部分⇒区間を狭く
- 補正回路、補完回路、区間表、区間判定回路が必要

  ⇒精度、回路規模、速度性能のバランスの考慮が必要

補完と可変区間



$$f(x_j) \cong f(x_i) + \frac{x_j - x_i}{x_k - x_i} \times (f(x_k) - f(x_i))$$

演習13B

- 被除数 4bit ÷ 除数 4bit = 商4bit 余4bit のマルチサイクル除算器のブロック図を描いてみよう。
- 五次(x^5 まで)の級数展開で sin 関数を計算するパイプライン乗算器のブロック図を描いてみよう。
- 表参照型演算器における線形補完回路のブロック図を描いてみよう。