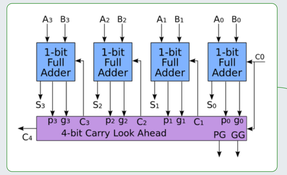
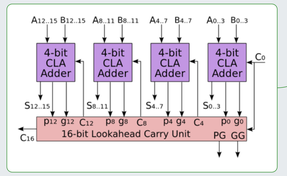
# 計算機算數

- 整數
  - 表示法
    - 無號數
      - 數值範圍
    - 有號數
      - Sign & Magnitude
        - 有 ±0
        - 運算後需要另外設定正負
      - 1's Complement
        - 二進位數值取 NOT
        - 有 ±0
        - 運算需要另外修正
      - 2's Complement
        - 就是 1 的補數加 1
        - 常見題型
          - 數值範圍
          - 數值轉換
  - 運算
    - + −
      - overflow detection
    - *
      - unsigned
        - 傳統
          - 手寫式、直觀的乘法
            - 被乘數左移、乘上乘數LSB，乘積位移(左)再累加
        - hardware friendly
          - 傳統之改良
            - 被乘數不動
            - 乘積左半清空、右半為乘數
            - 以乘積的 LSB 乘上被乘數，累加回乘積的左半
            - 乘積右移
      - signed
        - Booth's algo.
          - base on hardware friendly algo.
          - 乘積之外額外多一個虛構位元，初始 0，且要接收乘積右移移出的位元
          - 一次看兩個 bit：LSB & 虛構位元
          - 00, 11 無動作
          - **10：累加 −M 到 product**
            - 以 2 的補數想像，開頭 1 就是負的
              - Booth's algo 的證明就是以此為基礎
          - **01：累加 +M 到 product**
      - 可以使用數值範例推導
    - /
      - 相對少考，暫時略過
- 浮點數
  - 表示法
    - IEEE754 標準
      - single
        - 1, 8, 23
        - bias = [2^(E−1)] − 1
          - 127
          - 使用 bias notation，指數為無號數
          - 可用無號數判斷，浮點數比較就快得多
        - 正規化, normalization
          - 一律化為 1.x * 2的冪次方
      - double
        - 1, 11, (64−1−11)
        - bias = 1023
    - 非 IEEE754 類型
      - eg. IBM base16
      - 自創型、給範圍或精確度，求欄位寬度
        - Exponent 管數值範圍
        - Fraction 管數值精確度
    - 標準型考題推薦手法
      - Exponent
        - 都是 0
          - ±0
          - 非正規化數
            - 0.x * 2^(−126)
            - 1 * 2^−126 為正規化數的最小值，若要表示更"小"的數，則無法以正規化數表示
        - 全開
          - ±∞
            - 其他為零
          - NaN
            - 其他**非零**
      - bias 轉換
        - 01111111 為 基準點(零)
          - 負數往上減
          - 正數往下加
  - 運算
    - +
      - 步驟
        - 1. 指數對齊，調整小的
        - 2. 有效數字相加
        - 3. 運算結果正規化
        - 4. Rounding
          - if Rounding 後不為正規化數(進位了)
            - 再正規化一次
      - 其他
        - 不滿足結合律
          - 1*10^38 + 1*10^−38 + 1
    - *
      - 步驟
        - 1. IEEE754 下，**指數相加再減一**
          - 因為 bias 的部分重複加了
        - 2. 有效數字相加
        - 3. 正規化 & Rounding
          - 若 Rounding 後不為正規化數，則再正規化一次
        - 4. 設定正負號
    - Rounding
      - 判斷依據
        - LSB
          - 最右邊、最小、本來就有的 bit
        - Guard
          - 額外的延伸位元之一
        - Round
          - 額外的延伸位元之二
        - **非延伸位元**
        - Sticky
          - 視最後(最右)捨去的位元
            - 為零
              - Sticky bit = 0
            - 非零
              - Sticky bit = 1
            - i.e. OR 運算
            - 捨去的位元數也稱作
              - Units in the Last Place (ULP)
      - 判斷規則
        - G R 超過一半則進位
        - G R 小於一半不進位
        - G R 剛好一半(10)
          - S = 1
            - 進位
            - 表示剛剛截去的部分非零, i.e. 整體過半
          - S = 0
            - **Round to nearest even**
              - 回頭去看 LSB
              - 進位至最近的**偶數**
            - 朝向 ±∞
            - 直接捨去

- 電路設計
  - adder
    - 基礎
      - 1 bit full adder
        - input
          - a, b, Cin
        - output
          - Sum
            - 可由真值表寫出布林函式
              - Gate delay 為 3
                - sum of product & not gate
              - 也可再繼續化簡
                - a ⊕ b ⊕ c
                - 不好推導，可記
                - 使用 XOR 的狀況下，
                - gate delay 另依題目指示計算
          - Carry Out
            - ab + bc + ac
              - Gate delay 為 2
              - 可簡單推導，推薦記！
          - Circuit
            - 最簡版的暫時略過
            - 可使用 PLA 建構
    - n bits adder
      - 類型
        - RCA
          - 簡單串連
          - Critical path delay = 2n
          - Sum delay = 2n + 1
        - 無腦硬體
          - 布林函數，瘋狂代入替換就對了
          - 實作困難 & 成本過高
        - CLA
          - 推導起點：ab + bc + ac
            - i.e. c[i+1] = a[i]*b[i] + (a[i]+b[i])*c[i]
            - "產生"
              - generate g[i] = a[i] * b[i]
            - "傳遞"
              - propagate p[i] = a[i] + b[i]
            - example
              - 4 bit CLA
                - c4 = g3 + p3g2 + p3p2g1 + p3p2p1g0 + p3p2p1c0
              - 16 bit CLA
                - G0 = g3 + p3g2 + p3p2g1 + p3p2p1g0
                - 就是兩層式的 = 4*4 bit CLAs
            - **Gate delay 計算**
              - ai, bi → gi, pi
                - 1 gate delay
                - 1–bit adder 的產生傳遞
              - gi, pi → G, P
                - 2 gate delay
                - 4–bit adder 的產生傳遞
              - G, P → C
                - 2 gate delay
                - 4–bit adder 的 Carry out
              - C → c
                - 2 gate delay
                - 每一個 1–bit adder 的 carry out
              - Carry Out all set –> sum
                - 3 gate delay
                - sum of product & not gate
                - 使用 XOR 的狀況下，
                - gate delay 另依題目指示計算
        - carry save
          - 多個數相加的模式下
            - 傳統
            - 進位儲存
              - 所需的加法器少一個
              - 最長路徑(以加法器為單位)少一個
      - 評估
        - **Critical path delay 定義為 Carry Out 正確所需的時間**
  - multiplier
    - sequential
      - 循序電路，需要儲存單元
      - hardware friendly
        - Booth's algo.
    - combinational
      - 組合電路，只有邏輯閘
      - 快速
        - 簡單講就是加法器陣列
        - 比較快的原因
          - 無儲存單元、不需以 clock 控制，沒有時脈消耗，只有 gate delay
          - 容易最佳化(課本說的)
          - 容易管線化
      - Wallace tree
        - 基本上只是排列方式不同而已
- Overflow detection
  - 符號
  - 數值範圍
  - Software
    - 判斷
      - add 的溢位只有兩種可能
        - 正 + 正 = 負
        - 負 + 負 = 正
      - sub 類推
    - 步驟，以加法為例
      - 1. 先以無號數相加
        - 避免硬體發現溢位而終止程序
        - 直接被終止的話軟體是要偵測個屁
      - 2. 測試結果與來源同號與否
        - 用 slt 以有號數的方式與 $0 測試
        - 同號則無溢位
        - 異號即溢位
  - Hardware
    - 判斷
      - MSB 的 Carry In 與 Carry Out
      - c[n] ⊕ c[n−1]

- 用產生、傳遞的概念去記，硬背很難背