Part 1: Finite Precision Implementation Issues

2024年5月4日 14:15

Part1-FinitePrecisionImplementationIssues

- Fixed point and floating point number representations
    - decimal number system

      $(456)_{10} = 4 \times 10^2 + 5 \times 10^1 + 6 \times 10^0$

      $(3705.86)_{10} = 3 \times 10^3 + 7 \times 10^2 + 0 \times 10^1 + 5 \times 10^0 + 8 \times 10^{-1} + 6 \times 10^{-2}$

    - binary number system

      $(\underset{2^4}{1}\underset{2^3}{1}\underset{2^2}{0}\underset{2^1}{0}\underset{2^0}{1})_2 = (2^4)_{10} + (2^3)_{10} + (2^0)_{10} = (16)_{10} + (8)_{10} + (1)_{10} = (25)_{10}$

      $(101.01)_2 = (2^2)_{10} + (2^0)_{10} + (2^{-2})_{10} = (4)_{10} + (1)_{10} + (0.25)_{10} = (5.25)_{10}$

    - two's complement

      所有负数在当前位数下，正数部分取反加一

    - sign extension

      Multiplication of 2 n-bit numbers gives a 2n bit product.

      when multiply positive and negative num together, first sign extent.

    - fixed point number

      Dynamic range =
      $$20log_{10}\left(\frac{\text{largest number}}{\text{smallest positive number excluding 0}}\right)$$

      Precision = difference between 2 consecutive numbers

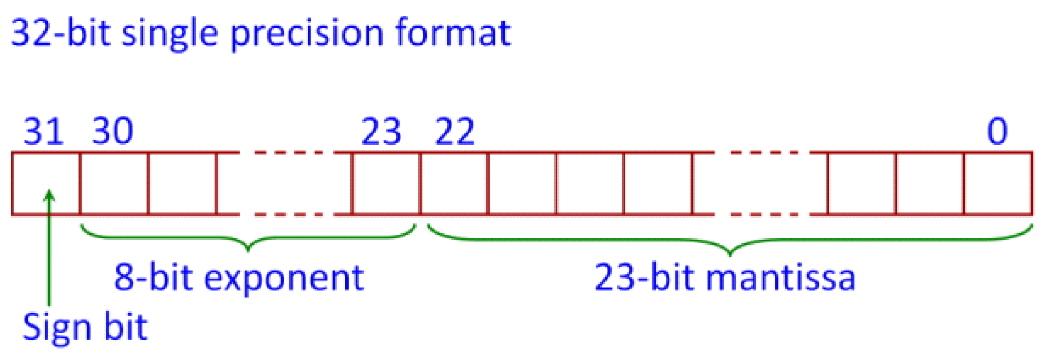
      Examples:

      Q0.15 format, precision = 2^{-15}

      Q15.0 format, precision = 1

    - floating point number
    - IEEE 754

      focus on 32bit representation.



exponent is biased by +127. that means if 127, fact exp is 0, if 130 fact is 3, if 126 fact is -1

Exponent:
Normal range 1 to $2^E - 2$ (1 to 254)

Special cases:
For 0 and denormalized numbers, exponent = 0
For ±∞ and NaNs, exponent = $2^E - 1$ (255)

denormalized number is very small number with exp 2^-126, that means leading integer is 0 rather than 1

- Quantization, truncation, and round-off
  - quantizer

    Typically, a quantizer is mid-tread (origin on tread of staircase)
    - Number of quantizer levels = $2^n - 1$
    - Quantization step size $$Q = \frac{V_{max} - V_{min}}{2^n - 1}$$
  - quantization error

Quantization error

Recall rounding results:
Additive noise model:
quantized output = input + quantization error

$$[x(n)] = x(n) + e(n)$$

- The probability density function for $e(n)$ is uniform from $-Q/2$ to $Q/2$
- mean of $e(n)$ = 0
- maximum $e(n)$ = $Q/2$
- mean square error = variance of $e(n)$ = $Q^2/12$
- root mean square error = std dev of $e(n)$ = $0.29Q$

Addition of uncorrelated noise sources:

- mean = $\eta_1 + \eta_2$
- variance = $\sigma_1{}^2 + \sigma_2{}^2$

○ signal to quantization noise ratio

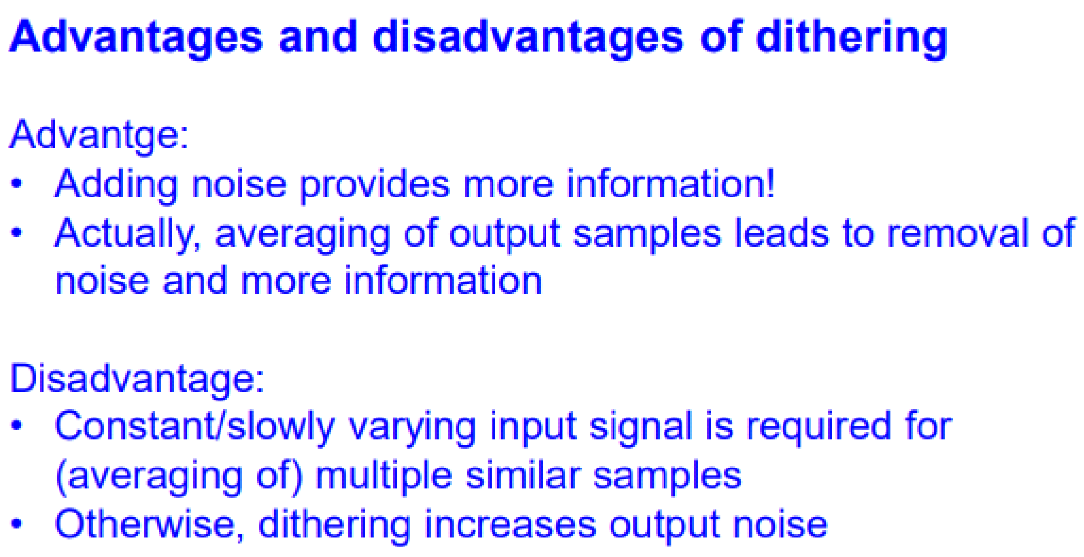
Signal to Quantization Noise Ratio (SQNR)

- Range of input signal = $-V_{max}$ to V_{max}
- Signal variance = $\sigma_x{}^2$
- Quantization noise variance = $\sigma_e{}^2$

$$\begin{aligned} SQNR &= 10\log\left(\frac{\sigma_x^2}{\sigma_e^2}\right) \\ &= 10\log\left(\frac{\sigma_x^2}{Q^2/12}\right) \\ &= 10\log\left(\frac{3\sigma_x^2 2^{2n}}{V_{max}^2}\right) \\ &= 6.02n + 4.77 + 20\log(\sigma_x/V_{max}) \end{aligned}$$

SQNR can also be improved by over-sampling and low pass filtering. Doubling the sampling rate improves the SQNR by 3 dB.

○ dithering



Advantages and disadvantages of dithering

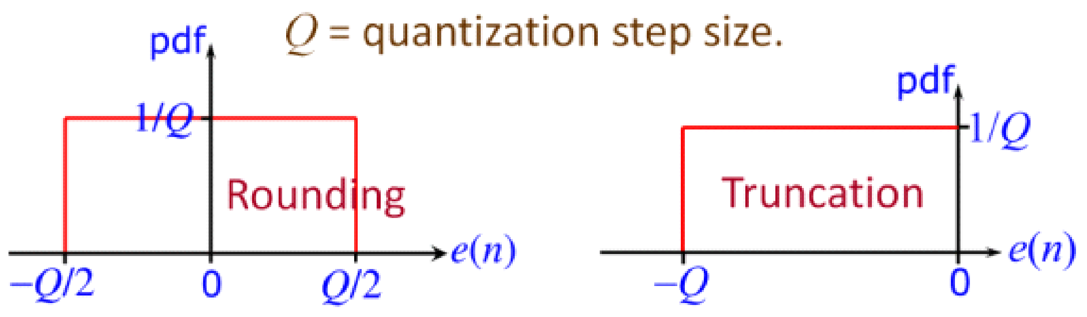
Advantge:
- Adding noise provides more information!
- Actually, averaging of output samples leads to removal of noise and more information

Disadvantage:
- Constant/slowly varying input signal is required for (averaging of) multiple similar samples
- Otherwise, dithering increases output noise

  - subtractive dithering
  - methods for word-length truncation

The probability of $x(n)$ having any specific value is zero.

The probability density function for $e(n)$ is uniform.



Matlab demo: pdf of $e(n)$

65

- Error analysis
  - statistical properties of the error - mean and variance
    量化噪声分析复习
  - auto-correlation and power spectral density of zero-mean noise
  - effect of input noise at the output of a system
  - addition of uncorrelated noise sources
  - output noise of an FIR filter
  - quantization noise in cascaded systems
  - quantization noise model for IIR filter
- ~~Finite word length effects [slides 108-126]~~
  - ~~coefficient quantization effect in FIR filter~~
  - ~~coefficient quantization effect in IIR filter~~

Part 2: Peripherals for DSP Applications

2024年5月4日 14:30

Part2-PeripheralsForDSPApplications

- Sampling
  - uniform/periodic sampling
    - Input sinusoid of frequency $f_0 + kf_s$ for any integer k

$$x_c(t) = \sin(2\pi(f_0 + kf_s)t)$$

  -
    - Same sampling frequency
    - Same discrete-time output signal

$\rightarrow$ f_0 and $f_0 + kf_s$ are indistinguishable after sampling

Nyquist sampling $f_s>2B$

d) $50 = f_0 + kf_s = (-25) + 75$
So $f_0 = -25$, or it yields samples identical to 25 Hz

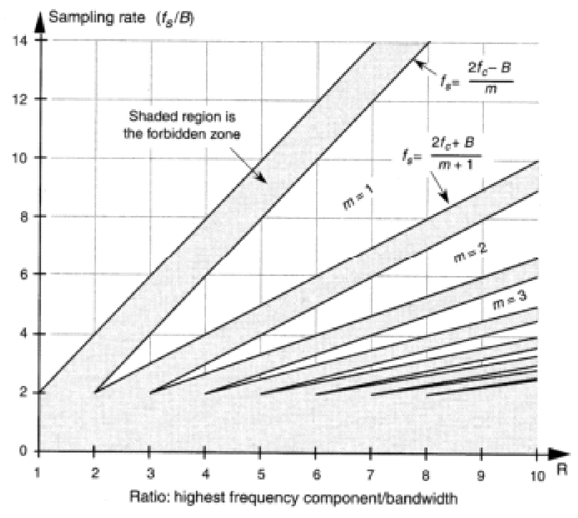
  - sampling low-pass signals
  - aliasing
  - sampling band-pass signals

In general, for any integer m, sampling at

$$\frac{2f_c + B}{m+1} \le f_s \le \frac{2f_c - B}{m}$$

such that $f_s \ge 2B$, gives no aliasing

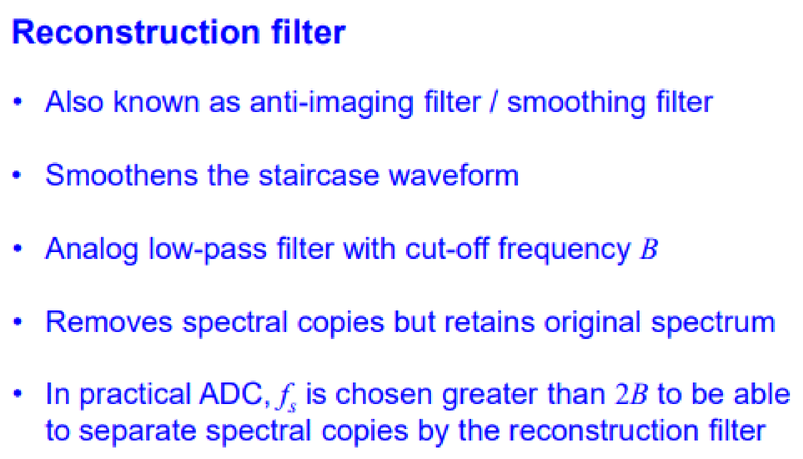
Regions of possible band-pass sampling rates



25

- Peripherals

- amplifier
  to adapt the dynamic range of ADC
- anti-aliasing filter
  cut high frequency noise
- ADC
  Sampling and quantization
- DAC
  - Converts the binary sample to the analog voltage
  - Holds the analog value until the next sample, thus producing a staircase waveform
- reconstruction filter

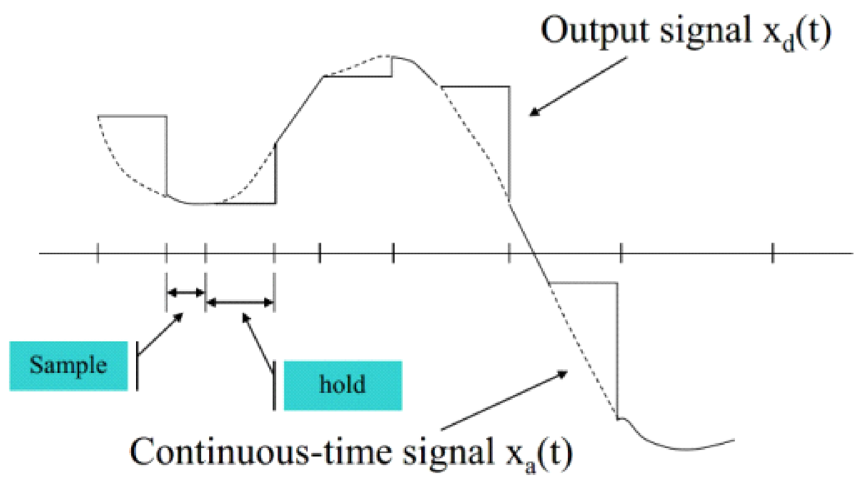


Reconstruction filter

- Also known as anti-imaging filter / smoothing filter
- Smoothens the staircase waveform
- Analog low-pass filter with cut-off frequency B
- Removes spectral copies but retains original spectrum
- In practical ADC, f_s is chosen greater than $2B$ to be able to separate spectral copies by the reconstruction filter

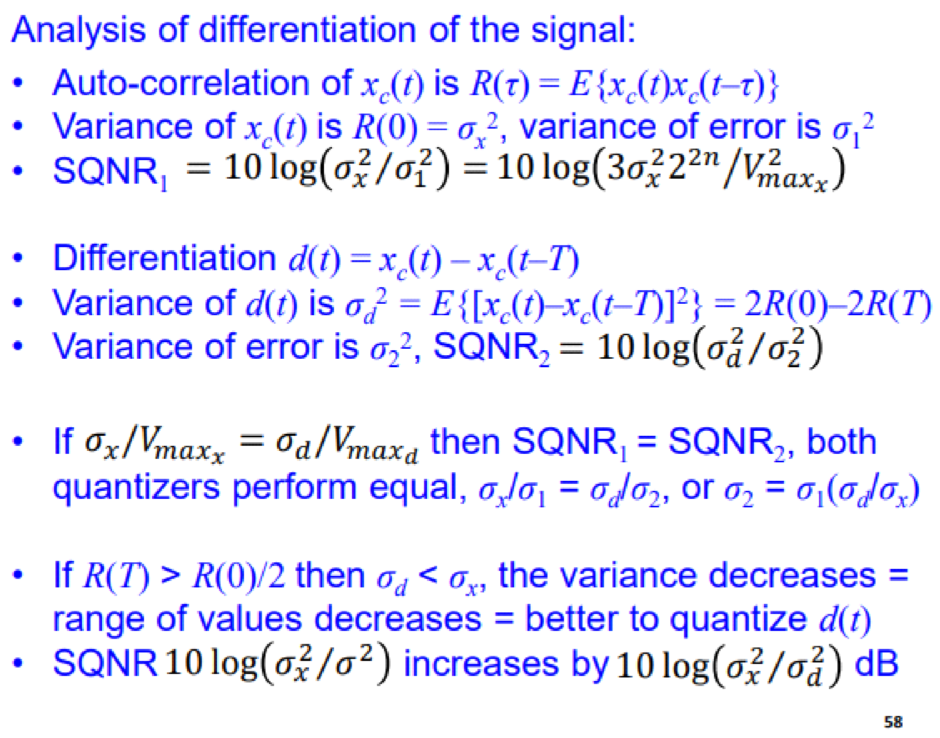
- internal and external peripherals

- Analog to digital converters
  - sample and hold circuit



- counter ADC
  through DAC compare ref V. one by one, very slow
- successive approximation ADC
  dichotomy, complexity O(log n)
- dual slope integrating ADC
  cuz analog components aging, use integral
  also slow, calc the time.
- parallel (flash) ADC
  fast, low resolution, need many space, for symmetric, divided by 17
- sigma-delta ADC

  1. one bit quantizer
  2. differential ADC, introduce more low pass error since integrator is low pass filter.



Analysis of differentiation of the signal:

- Auto-correlation of $x_c(t)$ is $R(\tau) = E\{x_c(t)x_c(t-\tau)\}$
- Variance of $x_c(t)$ is $R(0) = \sigma_x^2$, variance of error is σ_1^2
- SQNR$_1$ $= 10\log(\sigma_x^2/\sigma_1^2) = 10\log(3\sigma_x^2 2^{2n}/V_{max_x}^2)$

- Differentiation $d(t) = x_c(t) - x_c(t-T)$
- Variance of $d(t)$ is $\sigma_d^2 = E\{[x_c(t)-x_c(t-T)]^2\} = 2R(0)-2R(T)$
- Variance of error is σ_2^2, SQNR$_2$ $= 10\log(\sigma_d^2/\sigma_2^2)$

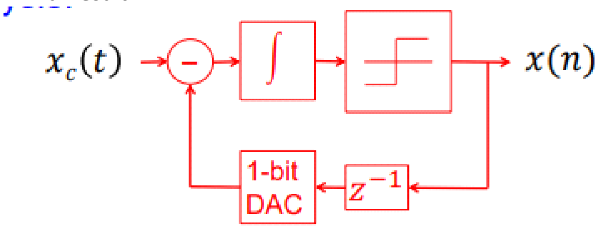
- If $\sigma_x/V_{max_x} = \sigma_d/V_{max_d}$ then SQNR$_1$ = SQNR$_2$, both quantizers perform equal, $\sigma_x/\sigma_1 = \sigma_d/\sigma_2$, or $\sigma_2 = \sigma_1(\sigma_d/\sigma_x)$

- If $R(T) > R(0)/2$ then $\sigma_d < \sigma_x$, the variance decreases = range of values decreases = better to quantize $d(t)$
- SQNR $10\log(\sigma_x^2/\sigma^2)$ increases by $10\log(\sigma_x^2/\sigma_d^2)$ dB

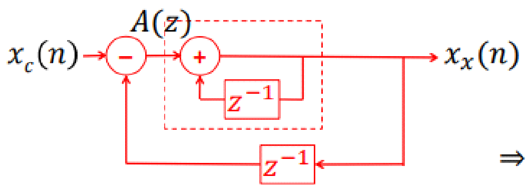
58

3 oversampling: 2 times of fs, half the omega frequency of input, double height,
also have 3dB gain of SQNR, and higher R(T)

4 final result:



Consider only the signal:



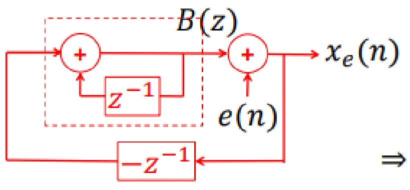
$$A(z) = X_c(z) - z^{-1}X_x(z)$$

$$X_x(z) = \frac{A(z)}{1-z^{-1}}$$

$$\Rightarrow X_x(z) = X_c(z)$$

- No change in signal (differentiator cancels integrator)

Consider only the error:



$$B(z) = \frac{-z^{-1}X_e(z)}{1-z^{-1}}$$

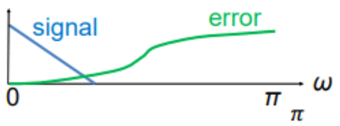
$$X_e(z) = B(z) + E(z)$$

$$\Rightarrow X_e(z) = (1-z^{-1})E(z)$$

- Highpass error (differentiator)

Error is reduced because

- Highpass error $X_e(z) = (1 - z^{-1})E(z)$
- Magnitude of frequency response $\left|H(e^{j\omega})\right| = 2\sin\frac{\omega}{2}$
- $e(n)$ power spectral density = N



- $x_e(n)$ variance $\sigma_{x_e}^2 = \int_{-\pi}^{\pi} \left(2\sin\frac{\omega}{2}\right)^2 N \frac{d\omega}{2\pi} = 2N$
- $x_e(n)$ variance after lowpass filter

$$\sigma_{x_e}^2 = \int_{-B}^{B} \left(2\sin\frac{\omega}{2}\right)^2 N \frac{d\omega}{2\pi} \approx \int_{-B}^{B} \omega^2 N \frac{d\omega}{2\pi} = \frac{B^3 N}{3\pi}$$

- 2 times sampling = B becomes half = $\sigma_{x_e}^2$ becomes 1/8 = 9dB more SQNR

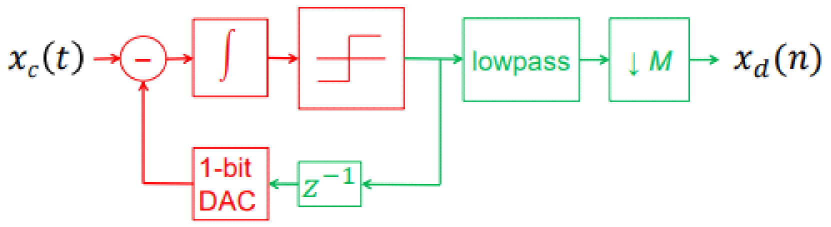
67

3) Decimator

Converts from 1-bit to n-bits

$$\begin{tikzcd} x(n) \arrow[r, "1\text{ bit}", "Mf_s\text{ sampling}"'] & \boxed{\text{lowpass}} \arrow[r, "n\text{ bits}", "Mf_s\text{ sampling}"'] & \boxed{\downarrow M} \arrow[r, "n\text{ bits}", "f_s\text{ sampling}"'] & x_d(n) \end{tikzcd}$$

4) Block diagram of sigma-delta ADC



first-order sigma-delta ADC (red = analog, green = digital)

5) SQNR

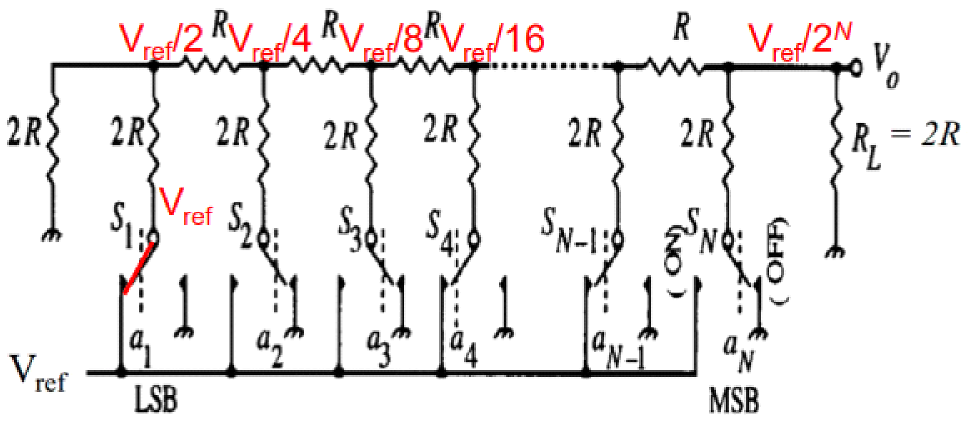
Since any quantizer SQNR = 6.02n + .., adding 1 bit resolution = 6dB more SQNR.

For oversampling:
- Each doubling of sampling = 3dB more SQNR = adding ½ bit

For oversampling and first order noise shaping:
- Each doubling of sampling = 9dB more SQNR = adding 3/2 bits

For oversampling and second order noise shaping:
- Each doubling of sampling = 15dB more SQNR = adding 5/2 bits

- Digital to analog converters
  - ladder DAC

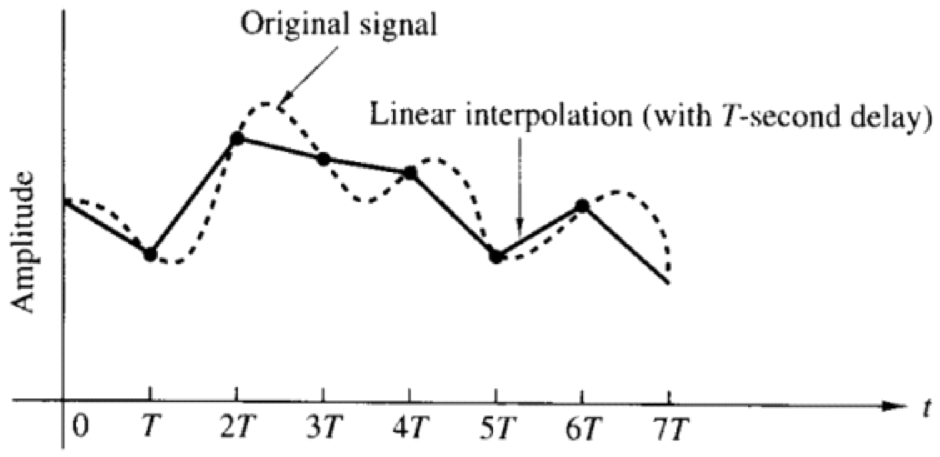


- a_1 sees resistances $2R + 2R$, so V_{ref} is halved
- $V_{ref}/2$ sees resistances $R + R$, so it is halved
- $V_{ref}/4$ sees resistances $R + R$, so it is halved

  :

  - first-order interpolation

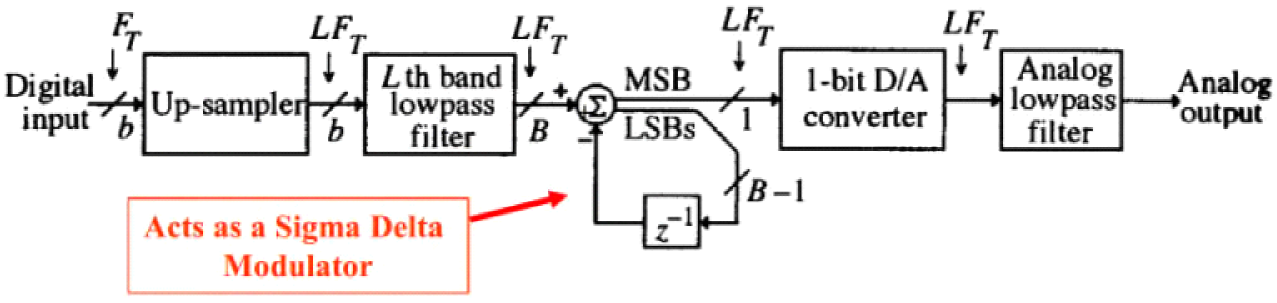
First-order (linear) interpolation



- First-order interpolation = connects successive samples with straight-line segments
- Straight line = order one

  - sigma-delta based DAC

Sigma-delta based DAC



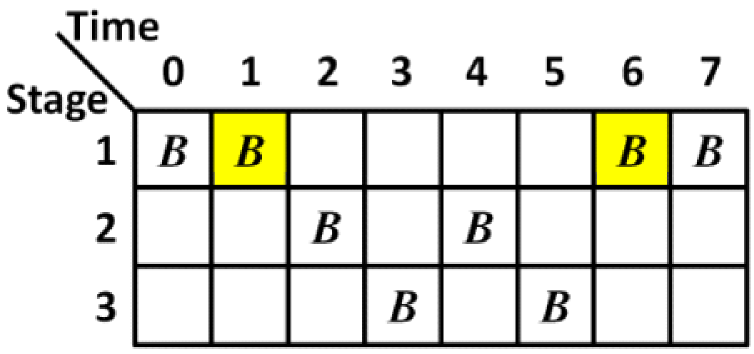
- Input upsampled to higher sampling frequency
- Quantized by 1-bit quantizer
- $B-1$ LSBs = quantization error
- Loop subtracts quantization error from next sample
- Downsampling not necessary since analog output
- Advantages: low-cost digital processing, simple analog filter

84

- ~~Real-time operations: Interrupts [slides 87-98]~~
  - ~~the sequence of events~~
  - ~~sources~~
  - ~~single-line interrupt~~
  - ~~multiple-line interrupt~~

- ~~vectored interrupt~~

- Real-time operations: scheduling
    - forbidden set



Time / Stage	0	1	2	3	4	5	6	7
1	*B*	*B*					*B*	*B*
2			*B*		*B*			
3				*B*		*B*		

Two yellow boxes in stage 1 show that initiation at time 5 is forbidden.

Forbidden set = {0,1,2,5,6,7}
Stage 1 utilization = ½
Maximum achievable throughput ≤ ¼

- permissible sequence

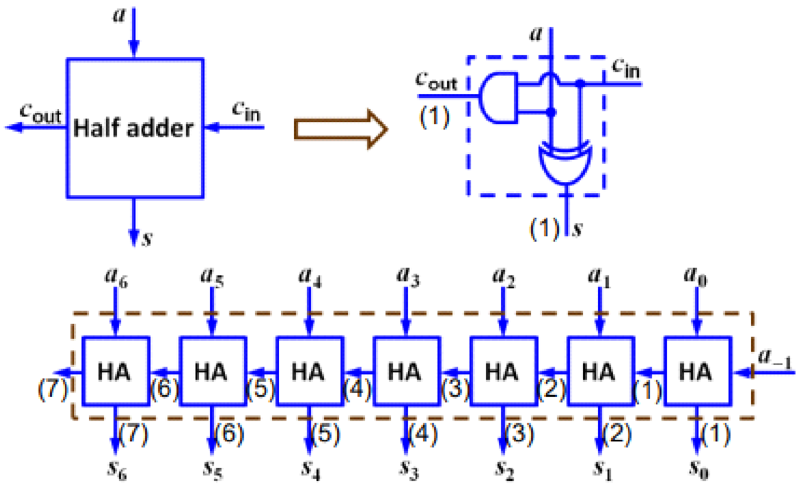
Part 3: Advanced Arithmetic Hardware

2024年5月4日　　14:30

[Part3-AdvancedArithmeticHardware(2)]

- Adder
    - rounding adder
    - half adder

Rounding adder

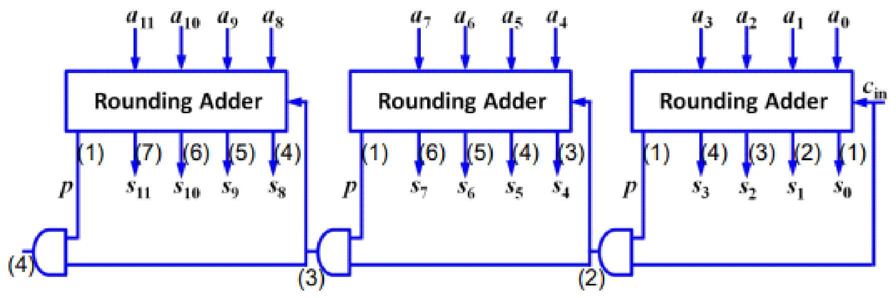


a
c_{out}
Half adder
c_{in}
s
(1)
HA
a_6 a_5 a_4 a_3 a_2 a_1 a_0
a_{-1}
(7) (6) (5) (4) (3) (2) (1)
s_6 s_5 s_4 s_3 s_2 s_1 s_0

Each half adder needs 1 gate delay. The *N*-bit rounding adder needs *N* gate delays to complete. This is because the carry bit propagates from stage to stage.

- carry look-ahead in rounding adder

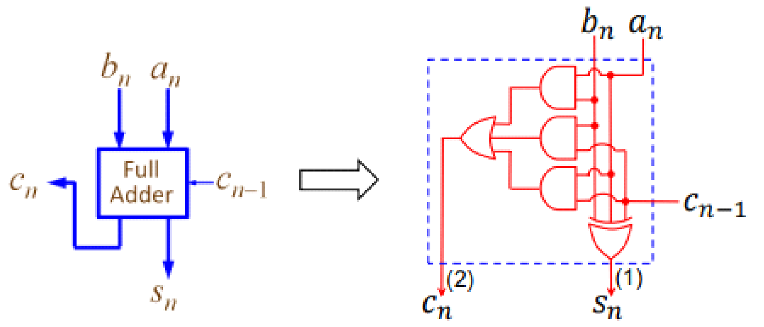
Therefore, cascading of such blocks is possible.
Carry-out takes 1 (to compute first *p*) + 3 (propagation through 3 gates) = 4 gate delays.



- full adder

Full Adder

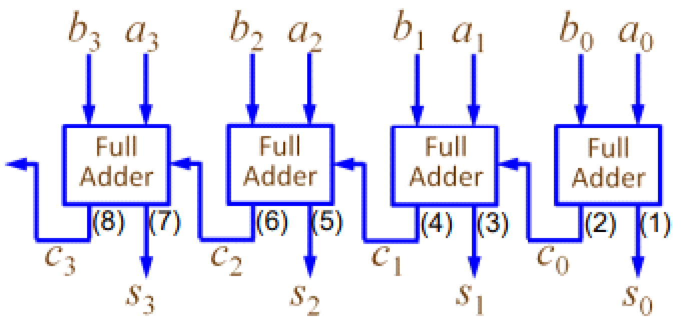
Bitwise addition of two 2's complement numbers requires adding not only two input bits but also the carry bit.



$$s_n = a_n \oplus b_n \oplus c_{n-1}$$
$$c_n = a_n \cdot b_n + b_n \cdot c_{n-1} + a_n \cdot c_{n-1}$$

8

- ripple carry adder

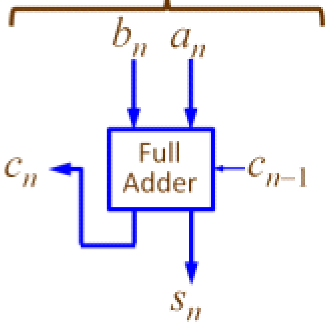


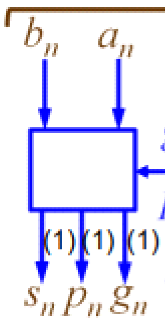
Computing each carry from valid inputs needs 2 gate delays. Therefore, computing c_3 needs 8 gate delays.

In general, an N stage ripple carry adder needs $2N$ gate delays to compute c_{N-1}.

- carry look-ahead adder

Instead of building this, we build this





$g_n = a_n \bullet b_n$

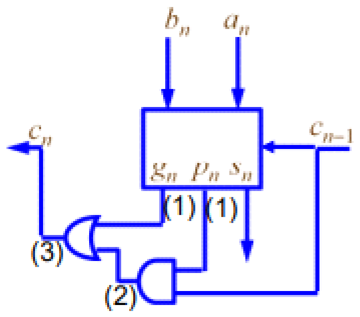
$p_n = a_n \oplus b_n$

$s_n = a_n \oplus b_n \oplus c_{n-1}$

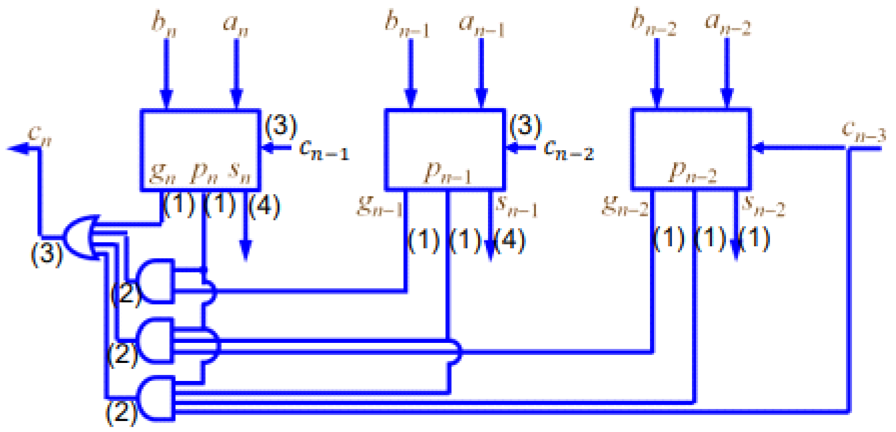
Carry generation for 1 bit:

$c_n = g_n + p_n \bullet c_{n-1}$

c_n needs 1 gate delay to compute g_n and p_n, plus 2 gate delays = 3 gate delays.

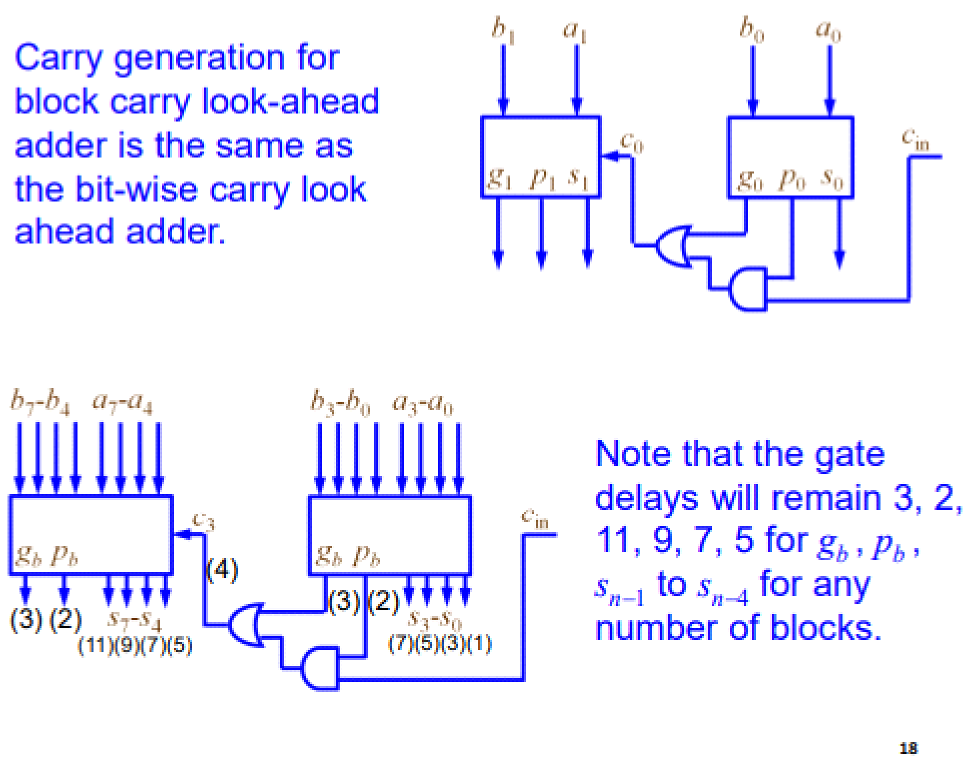


Carry generation for 3 bits:



$c_n = g_n + p_n \bullet g_{n-1} + p_n \bullet p_{n-1} \bullet g_{n-2} + p_n \bullet p_{n-1} \bullet p_{n-2} \bullet c_{n-3}$

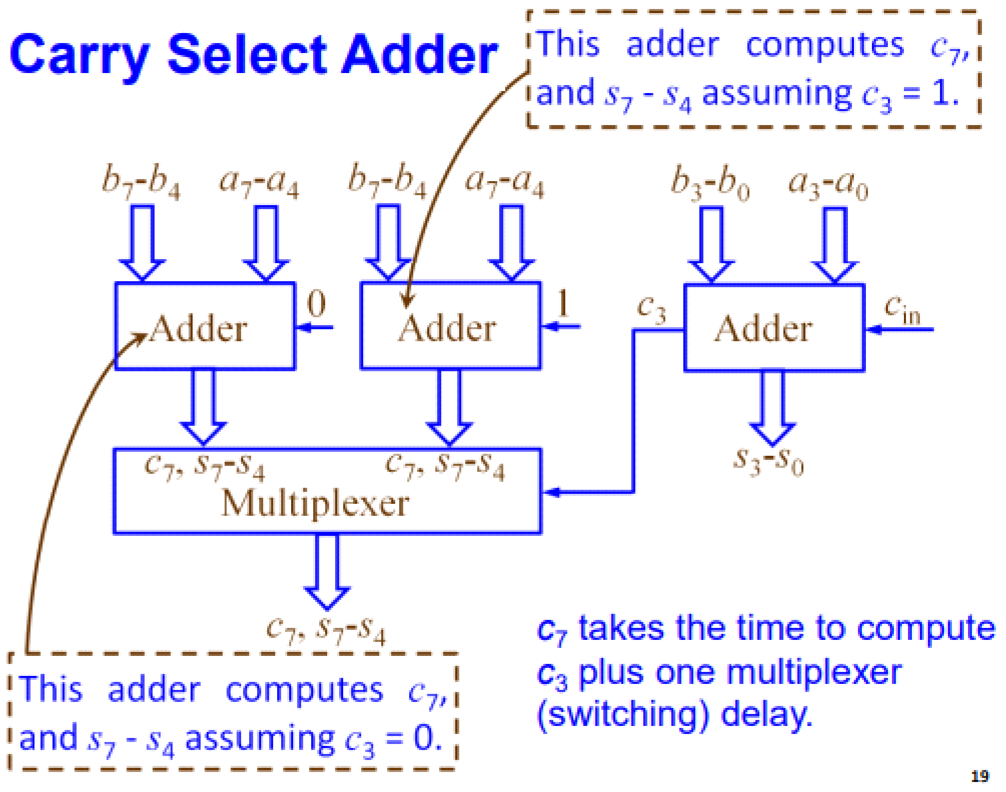
c_n still takes 3 gate delays.
Note that the gate delays will remain 3, 1, 1, 4 for c_n, g_n, p_n, and s_n for any number of bits n.



Carry generation for block carry look-ahead adder is the same as the bit-wise carry look ahead adder.

Note that the gate delays will remain 3, 2, 11, 9, 7, 5 for g_b, p_b, s_{n-1} to s_{n-4} for any number of blocks.

- carry select adder



Carry Select Adder

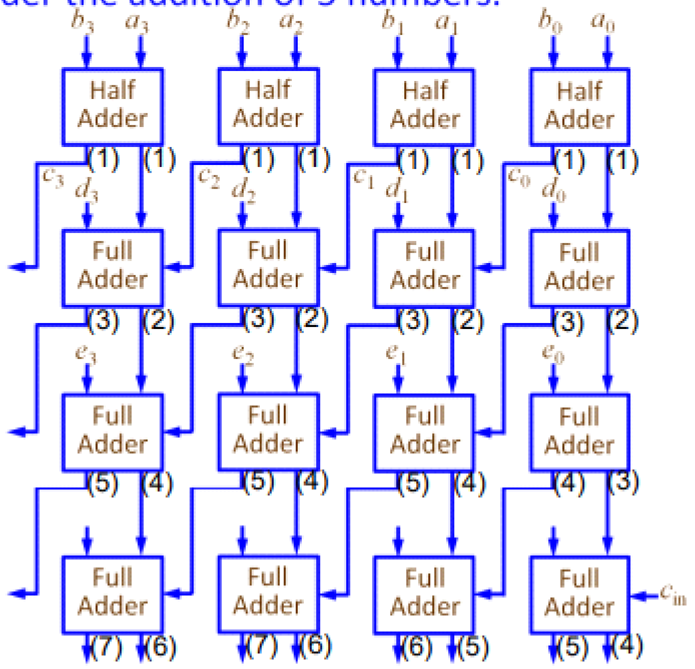
This adder computes c_7, and s_7 - s_4 assuming c_3 = 1.

This adder computes c_7, and s_7 - s_4 assuming c_3 = 0.

c_7 takes the time to compute c_3 plus one multiplexer (switching) delay.

- carry save adder

Now consider the addition of 5 numbers.



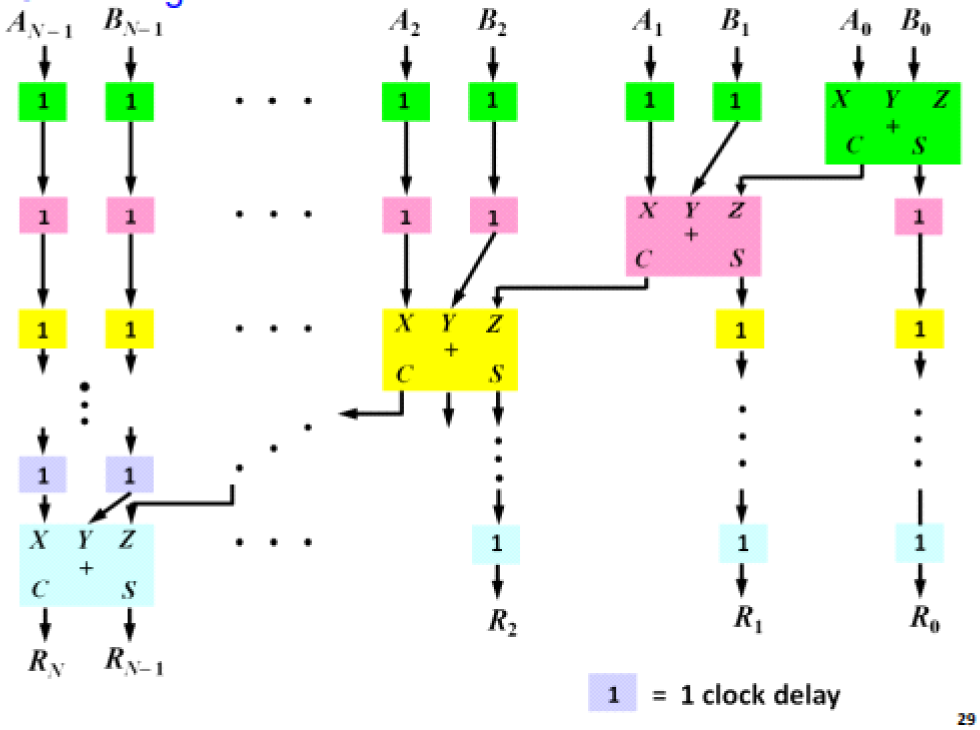
b_3 a_3 b_2 a_2 b_1 a_1 b_0 a_0
Half Adder
Full Adder
c_3 d_3 c_2 d_2 c_1 d_1 c_0 d_0
e_3 e_2 e_1 e_0
(1) (2) (3) (4) (5) (6) (7)
c_{in}

Only the carry bits for the last row ripple "horizontally".

25

- pipelined adder

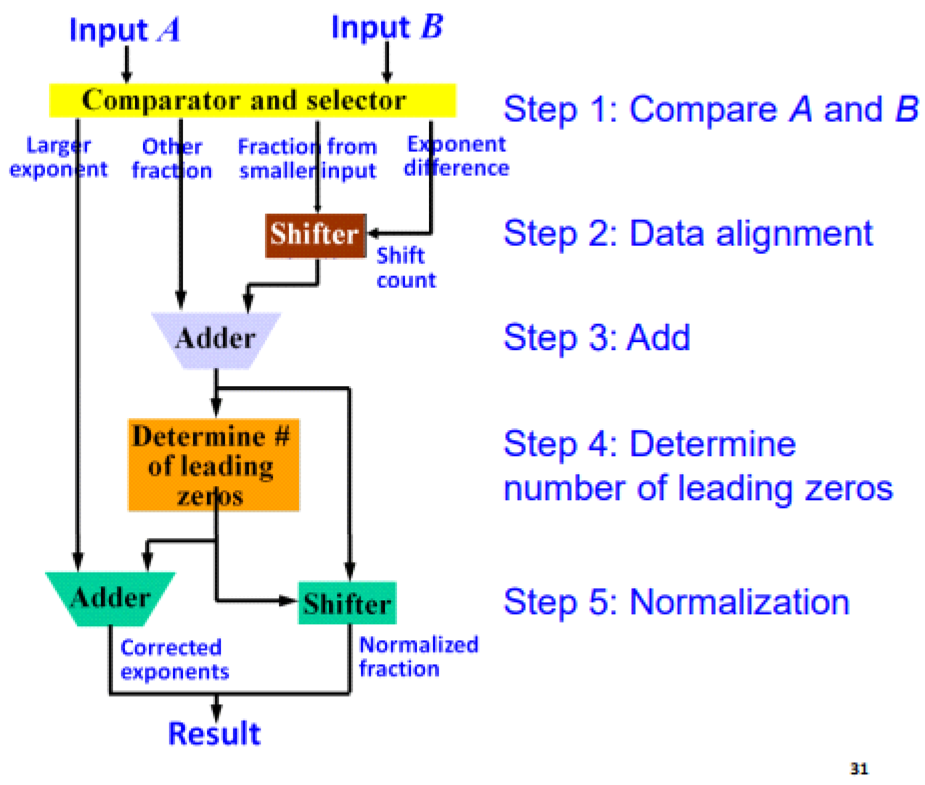
Cascading *N* full adders:



A_{N-1} B_{N-1} A_2 B_2 A_1 B_1 A_0 B_0
1
X Y Z + C S
R_N R_{N-1} R_2 R_1 R_0
1 = 1 clock delay

29

- floating-point adder



- ~~Multiplier [slides 37-51]~~
  - ~~integer-power-of-two multiplier~~
  - ~~Booth multiplier~~
  - ~~Wallace tree multiplier~~

Part 4: Algorithms and Architectures for VLSI

2024年5月4日 14:30

Part4-AlgorithmsArchitecturesForVLSI

- Algorithm strength reduction
  - fast Fourier transform

DFT is defined as

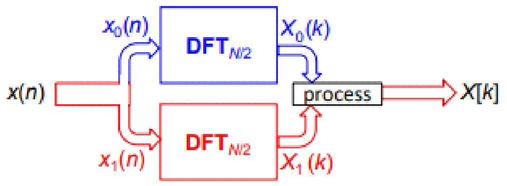
$$X(k) = X(z)|_{z=e^{j2\pi k/N}} = \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N}$$

for k = 0,1,…, N–1.

Direct computation of DFT requires N^2 complex multiply and $N(N–1)$ complex add.

Therefore: $X(k) = X_0(k) + e^{-\frac{j2\pi k}{N}} X_1(k)$

This leads to a divide and conquer approach known as the decimation-in-time FFT:



Each N/2 point DFT requires N^2/4 complex multiply and $N(N–2)$/4 complex add. The “process” requires N complex multiply and N complex add. Overall, the structure needs $N(N+2)$/2 complex multiply and N^2/2 complex add.

Recursion: Compute each N/2 point DFT using two N/4 point DFTs, and so forth. FFT requires about (N/2)log$N$ complex multiply and $N$log$N$ complex add.

6

  - complex multiplication

Complex Multiplication

Multiply a given number $a+jb$ with an input $A+jB$:

4 real multiply

$$(a+jb)(A+jB) = (aA - bB) + j(aB + bA)$$

2 real add

Rewrite it as:

$$(aA - bB + bA - bA) + j(aB + bA + bB - bB)$$
$$= \{(a+b)A - b(A+B)\} + j\{b(A+B) + (a-b)B\}$$

Let $c = a+b, d = a-b$ be precomputed, then:

3 real multiply

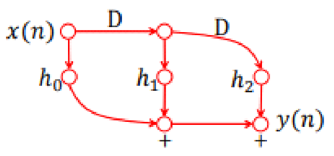
$$= \{cA - b(A+B)\} + j\{b(A+B) + dB\}$$

3 real add

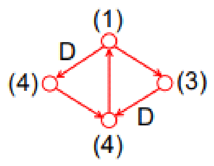
7

  - data flow graph

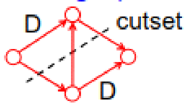
Data flow graph:
Nodes = computations such as add, multiply
Directed edges = data path between nodes, including any delay



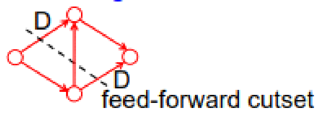
In a data flow graph, nodes are labelled by computation time, and edges are labelled by no of delays.

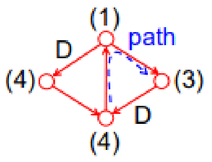


Cutset: set of edges which, if removed, makes the graph disjoint



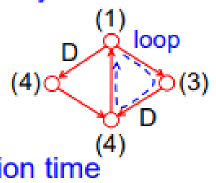
Feed-forward cutset: cutset with all forward direction edges





A path of a graph begins at some node and ends at any node.

A loop of a graph begins and ends at the same node.

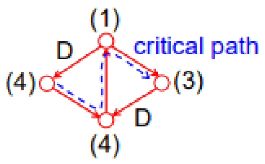


Loop bound of a loop = $\dfrac{\text{loop computation time}}{\text{no of delays in the loop}}$

Loop bound of the above loop = (1+3+4)/(0+1+0) = 8

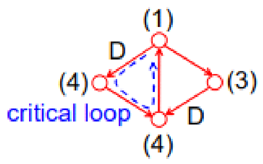
Critical path of a graph is the longest computation time among all zero delay paths.
Clock cycle is lower bounded by the critical path.

Critical path of the above graph = 4+4+1+3 = 12



Iteration bound is the maximum loop bound of a graph.
Critical loop is the loop with the maximum loop bound.

In this graph:
Iteration bound = 9
Critical loop = the left loop



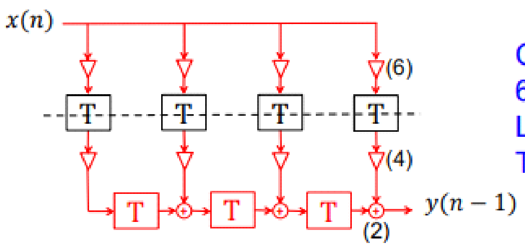
- Pipelining

Pipelining reduces the critical path and therefore increases the clock/sampling speed

- latency and throughput

Pipelining option 2: fine-grain pipelining
Let T_M = 10 gate delays, T_A = 2 gate delays

To achieve a critical path less than T_M , break the multiplier into two parts with processing times of 6 gate delays and 4 gate delays.



Critical path =
6 gate delays
Latency = 2
Throughput = 1

Pipelining reduces critical path by increasing latency.
We now study techniques such as retiming that reduces critical path without increasing latency.

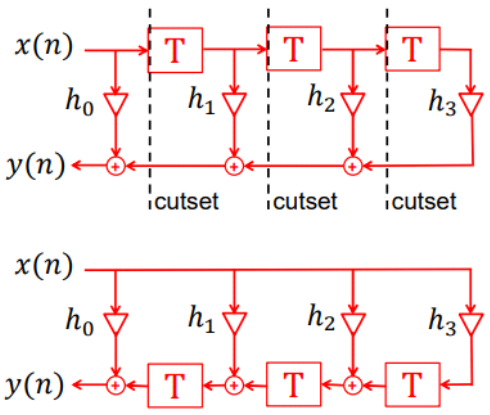
16

- FIR filters

Retiming

- cutset retiming

Pipelining option 3: retiming
reduce the critical path using transposed form FIR structure

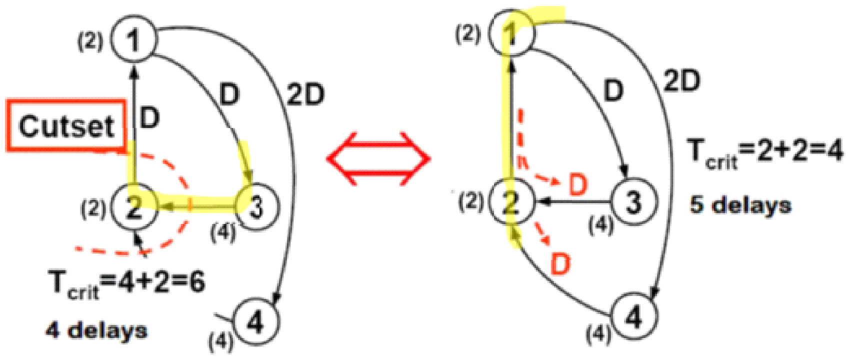


Critical path = T_M + T_A , independent of the length of the filter
This is better than other options, because latency is not increased, and it does not require additional latches.

- node retiming

Node Retiming

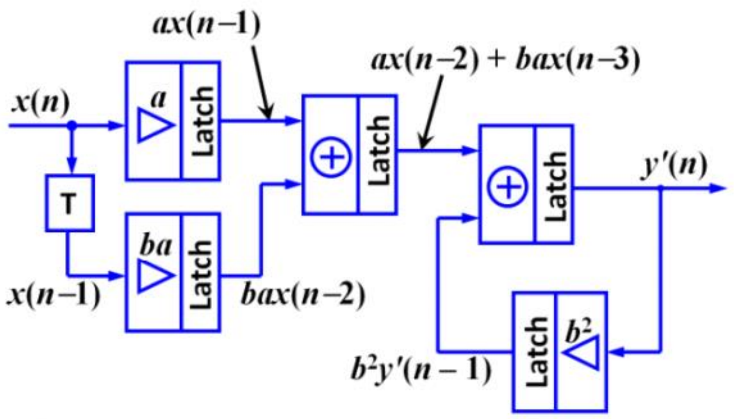
Consider a cutset around a node.



k=1 delay is added to the edges 3→2 and 4→2, and 1 delay is removed from the edge 2→1, to obtain the retimed graph.

Critical path before retiming = 4 + 2 = 6
Critical path after retiming = 4 or 2 + 2 = 4
However, no of delays increases from 4 to 5.

- Pipelined recursive filters
    - first order filters

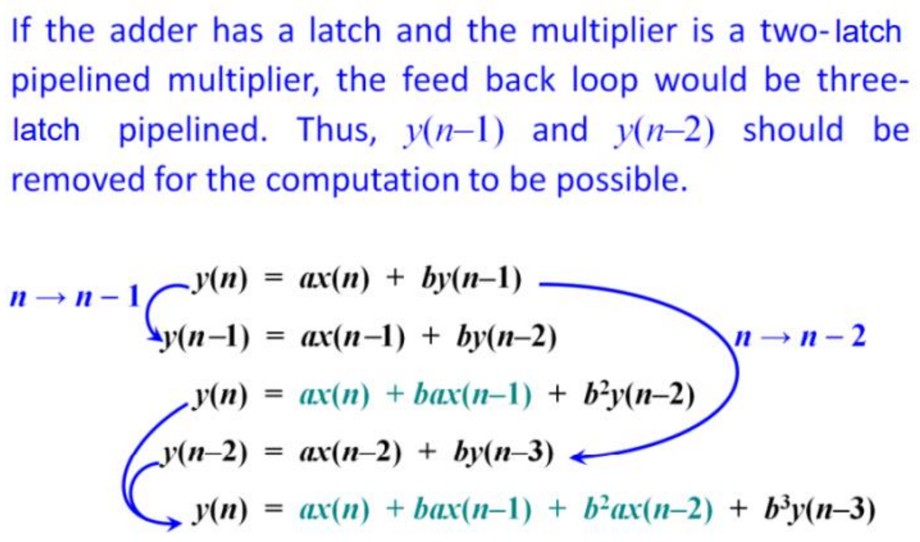


The input/output relationship for the above implementation is $y'(n) = ax(n-3) + bax(n-4) + b^2y'(n-2)$.

We have, $y(n) = ax(n) + by(n-1) = ax(n) + bax(n-1) + b^2y(n-2)$. Replacing n by $n-3$, we have $y(n-3) = ax(n-3) + bax(n-4) + b^2y(n-5)$.

It can be seen that $y'(n) = y(n-3)$.

Latency = 3 (because of the latch), throughput = 1



If the adder has a latch and the multiplier is a two-latch pipelined multiplier, the feed back loop would be three-latch pipelined. Thus, $y(n-1)$ and $y(n-2)$ should be removed for the computation to be possible.

$n \to n-1$

$$y(n) = ax(n) + by(n-1)$$

$$y(n-1) = ax(n-1) + by(n-2)$$

$n \to n-2$

$$y(n) = ax(n) + bax(n-1) + b^2y(n-2)$$

$$y(n-2) = ax(n-2) + by(n-3)$$

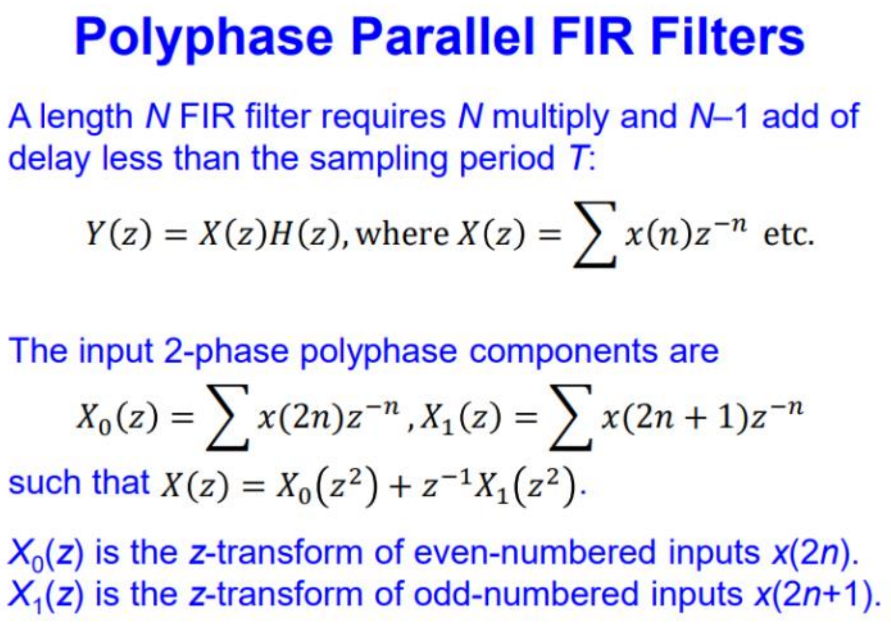
$$y(n) = ax(n) + bax(n-1) + b^2ax(n-2) + b^3y(n-3)$$

- ○ higher order filters

  P44 通过重构，把3个加法器减少到2个，类似于加法乘法结合律，减少运算符

  根据题目给的想使用多少latch的adder和multiplier，改写目标函数，进而实现电路

- Parallel processing
  - ○ polyphase parallel FIR filters



Polyphase Parallel FIR Filters

A length N FIR filter requires N multiply and N–1 add of delay less than the sampling period T:

$$Y(z) = X(z)H(z), \text{where } X(z) = \sum x(n)z^{-n} \text{ etc.}$$

The input 2-phase polyphase components are

$$X_0(z) = \sum x(2n)z^{-n}, X_1(z) = \sum x(2n+1)z^{-n}$$

such that $X(z) = X_0(z^2) + z^{-1}X_1(z^2)$.

$X_0(z)$ is the z-transform of even-numbered inputs $x(2n)$.
$X_1(z)$ is the z-transform of odd-numbered inputs $x(2n+1)$.

  - ○ low complexity parallel FIR filters
- Unfolding

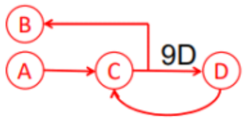
  【转载】VLSI数字信号处理系统：展开 – 哔哩哔哩 (bilibili.com)
  - ○ algorithm

Algorithm for Unfolding

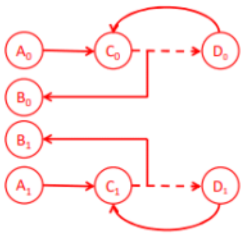
Systematic procedure for unfolding:

Step 1: For each node U in the original data flow graph, draw J nodes $U_0, U_1, \ldots, U_{J-1}$.

Original data flow graph:



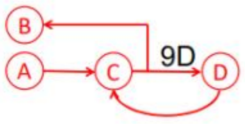
Replicated data flow graph:



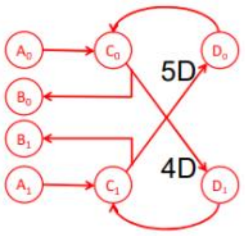
Step 2: For each edge $U \rightarrow V$ with m delays in the original data flow graph, draw J edges $U_i \rightarrow V_{(i+m)\%J}$ with $\left\lfloor \frac{i+m}{J} \right\rfloor$ delays for $i = 0, 1, \ldots, J-1$.

(% = mod or modulo, find the remainder.)

($\lfloor \bullet \rfloor$ = floor, same as truncation.)

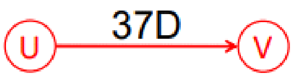


Edges without delay such as $A \rightarrow C$, $D \rightarrow C$ are unchanged.



Edge $C \rightarrow D$ has m = 9 delays.
So, $C_0 \rightarrow D_{(0+9)\%2} = D_1$ has $\lfloor(0+9)/2\rfloor = 4$ delays.
$C_1 \rightarrow D_{(1+9)\%2} = D_0$ has $\lfloor(1+9)/2\rfloor = 5$ delays.

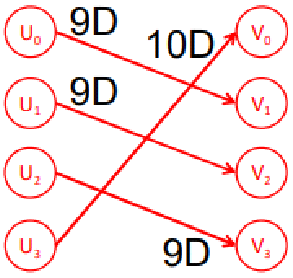
Example 2:



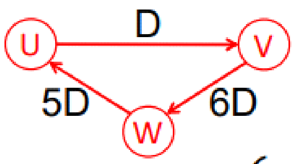
Let $J = 4$.

Recall that $U_i \rightarrow V_{(i+m)\%J}$ with $\lfloor (i+m)/J \rfloor$ delays.

$$\left\lfloor \frac{i+m}{J} \right\rfloor = \left\lfloor \frac{i+37}{4} \right\rfloor = \begin{cases} 9 & i = 0,1,2 \\ 10 & i = 3 \end{cases}$$



Example 3:

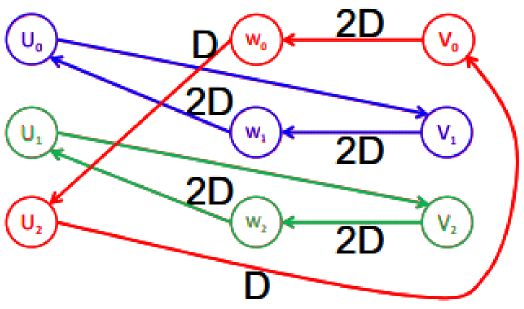


Let $J = 3$.

For $U_i \rightarrow V_{(i+1)\%3}$, $\left\lfloor \frac{i+m}{J} \right\rfloor = \left\lfloor \frac{i+1}{3} \right\rfloor = \begin{cases} 0 & i = 0,1 \\ 1 & i = 2 \end{cases}$

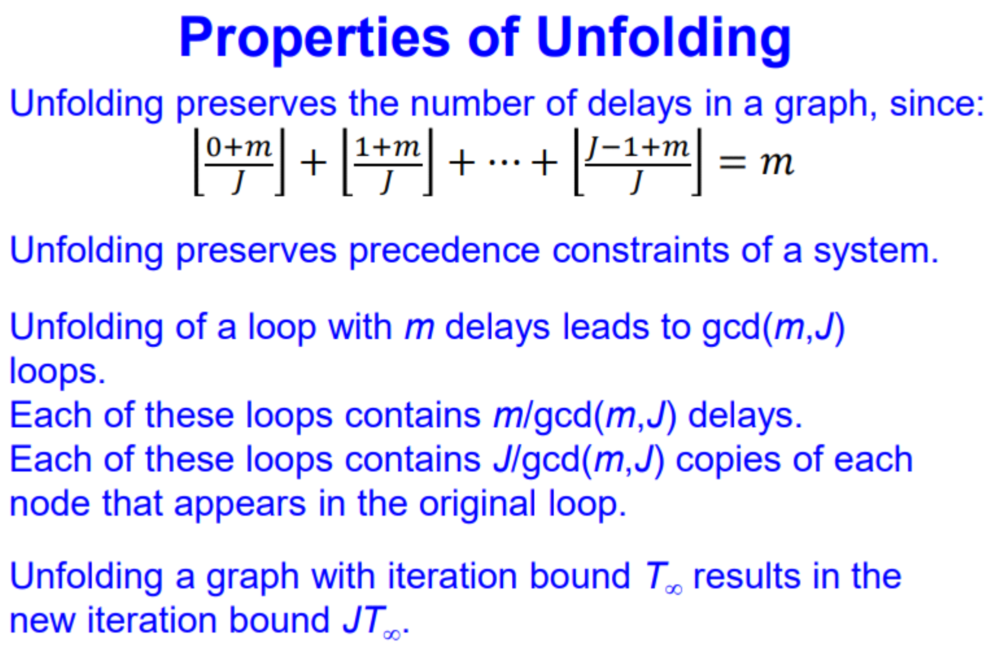
For $V_i \rightarrow W_{(i+6)\%3}$, $\left\lfloor \frac{i+m}{J} \right\rfloor = \left\lfloor \frac{i+6}{3} \right\rfloor = 2$

For $W_i \rightarrow U_{(i+5)\%3}$, $\left\lfloor \frac{i+m}{J} \right\rfloor = \left\lfloor \frac{i+5}{3} \right\rfloor = \begin{cases} 1 & i = 0 \\ 2 & i = 1,2 \end{cases}$



Consists of 3 disjoint loops since gcd(12 delays, J=3) = 3.

- properties



Properties of Unfolding

Unfolding preserves the number of delays in a graph, since:

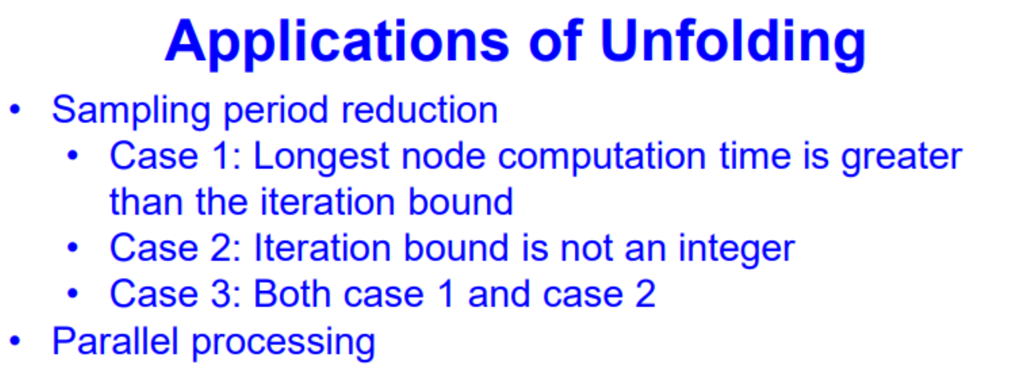
$$\left\lfloor \frac{0+m}{J} \right\rfloor + \left\lfloor \frac{1+m}{J} \right\rfloor + \cdots + \left\lfloor \frac{J-1+m}{J} \right\rfloor = m$$

Unfolding preserves precedence constraints of a system.

Unfolding of a loop with *m* delays leads to gcd(*m*,*J*) loops.
Each of these loops contains *m*/gcd(*m*,*J*) delays.
Each of these loops contains *J*/gcd(*m*,*J*) copies of each node that appears in the original loop.

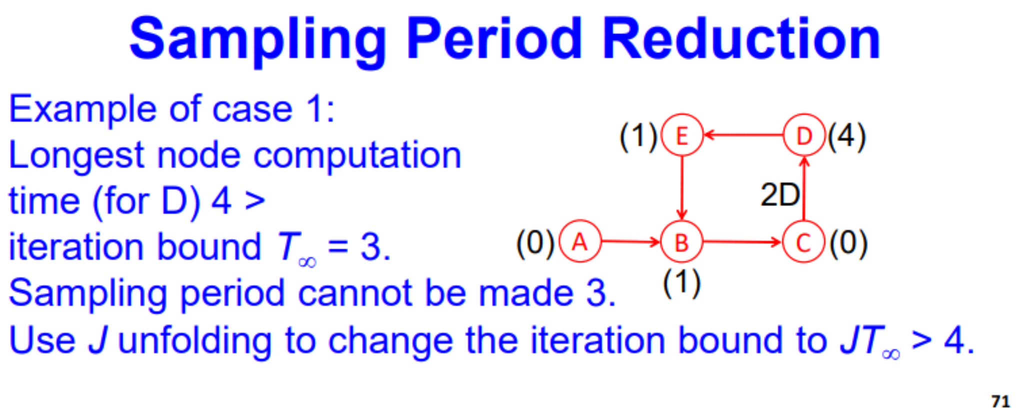
Unfolding a graph with iteration bound T_∞ results in the new iteration bound JT_∞.

- applications



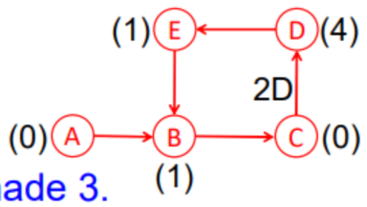
Applications of Unfolding

- Sampling period reduction
  - Case 1: Longest node computation time is greater than the iteration bound
  - Case 2: Iteration bound is not an integer
  - Case 3: Both case 1 and case 2
- Parallel processing



Sampling Period Reduction

Example of case 1:
Longest node computation time (for D) 4 > iteration bound T_∞ = 3.
Sampling period cannot be made 3.
Use *J* unfolding to change the iteration bound to JT_∞ > 4.

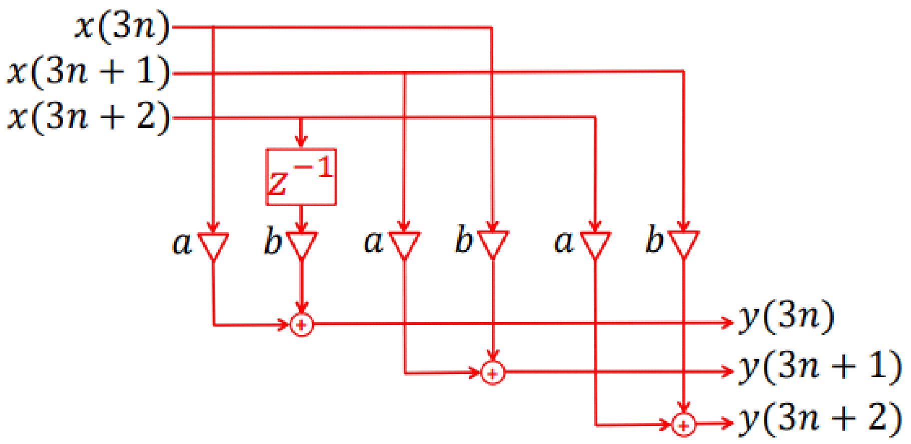


$$\begin{tikzcd} & (1)\,E \arrow[d] & & D\,(4) \arrow[ll] \\ (0)\,A \arrow[r] & B\,(1) \arrow[rr] & & C\,(0) \arrow[u, "2D"'] \end{tikzcd}$$

71

- parallel FIR filters

Parallel architecture:



In unfolding factor *J* parallel structures, since $T_c = JT_s$, a latch z^{-1} of 1 clock cycle produces an effective delay of *J* sampling periods, JT_s .

In the above structure, $x(3n+2)$ passing through z^{-1} is delayed by 3 sampling periods, or becomes $x(3n-1)$.

Since $T_s = 1/3\ T_c$, the sampling frequency is tripled. Latency = 1 clock, throughput = 3 inputs/clock

75

- ~~Parallel recursive filters [slides 79-92]~~
  - ~~first order filters~~
  - ~~higher order filters~~
- ~~Folding [slides 93-105]~~
  - ~~folding an FIR filter~~
  - ~~folding an IIR filter~~

Part 5: Programming and Architecture for Digital Signal Processor

2024年5月4日 14:30

Part5-ProgrammingArchitectureForDSP(1)

- Real-time processing on a DSP processor
    - sampling frequency consideration

Sample-by-sample processing

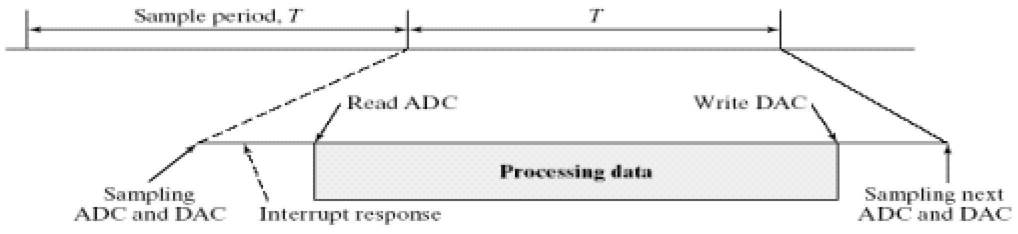


Figure 3.16 Detailed interrupt timing at each sampling interval for an I/O operation

Here we show the detailed sampling interval for sample-by-sample processing mode.

Real-Time processing refers to the digital processing of data within the sampling interval.

DSP must finish processing within 1 sample interval, T_S

Require 1 word FIFO, memory or register to hold incoming data

Block processing

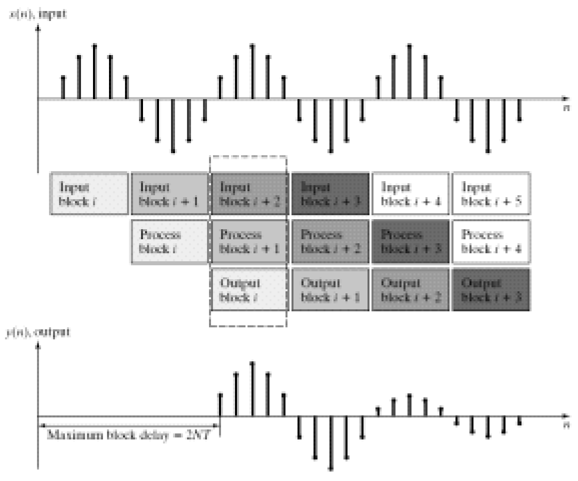
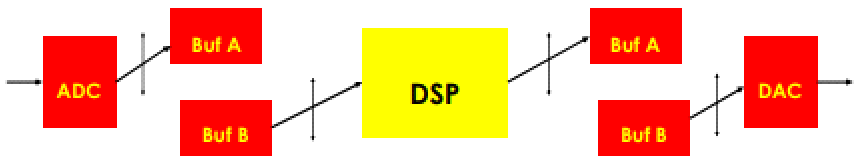


Figure 3.17 Block processing of an input signal in a block of five samples

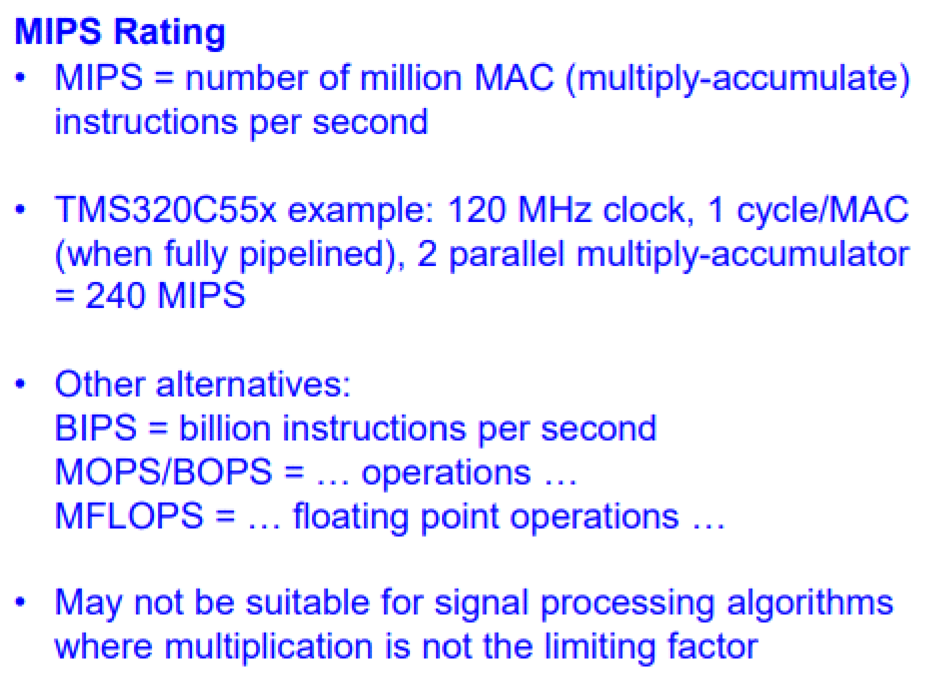
- Collect N samples at a time
- Processing must finish within N sample period, NT
- Normally used in FFT, Compression
- Requires double or triple buffering to perform acquisition and processing
- $T_p < N*T_s$
- A maximum block delay of $2NT_s$

Double buffering



- Advantage:
  - Can process more data for some given T_s due to the reduced setup time
  - Or use a higher F_s compared to the sampling approach
- Disadvantage:
  - More memory required
  - More effort in programming
  - Processing latency

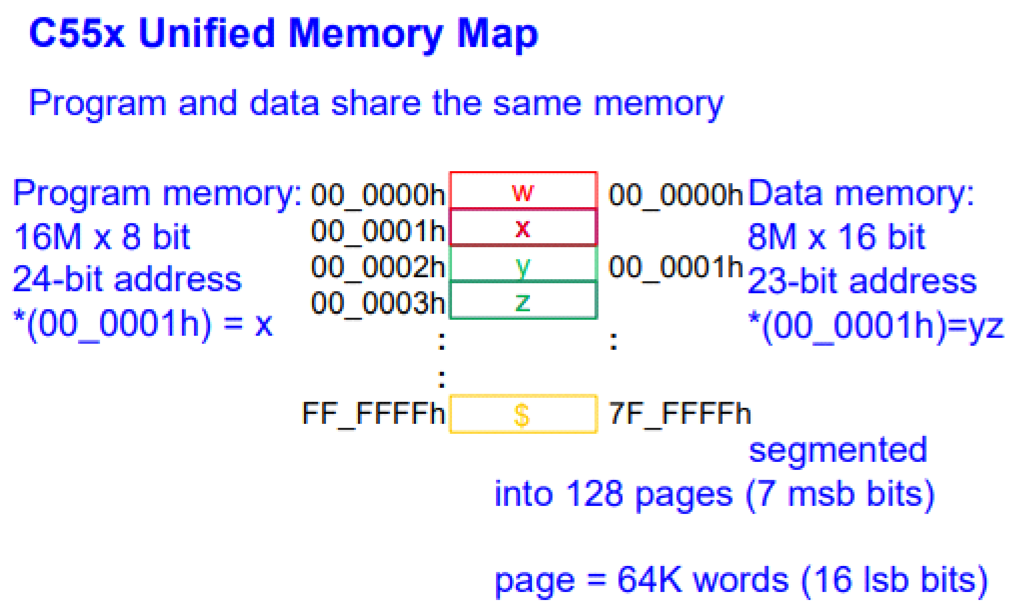
- DSP performance ratings



MIPS Rating

- MIPS = number of million MAC (multiply-accumulate) instructions per second
- TMS320C55x example: 120 MHz clock, 1 cycle/MAC (when fully pipelined), 2 parallel multiply-accumulator = 240 MIPS
- Other alternatives:
  BIPS = billion instructions per second
  MOPS/BOPS = … operations …
  MFLOPS = … floating point operations …
- May not be suitable for signal processing algorithms where multiplication is not the limiting factor

- ~~DSP selection [slides 15-25]~~
  - ~~hardware platforms~~
  - ~~fixed point versus floating point~~
  - ~~TMS family~~
- TMS320C55x programming
  - memory map and registers



C55x Unified Memory Map

Program and data share the same memory

Program memory:
16M x 8 bit
24-bit address
*(00_0001h) = x

Data memory:
8M x 16 bit
23-bit address
*(00_0001h)=yz

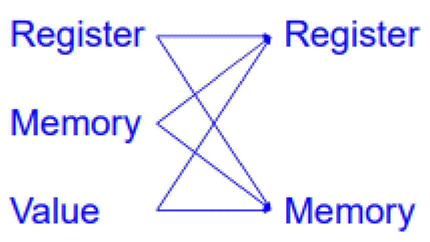
segmented into 128 pages (7 msb bits)

page = 64K words (16 lsb bits)

- Registers to address data memory and I/O space: normally 16 bit (address within page), with X (extended) 23 bit (includes page address)
  - Auxiliary: AR0 to AR7
  - Coefficient data pointer (CDP), XCDP, used to address coefficient memory
  - Circular buffer registers
  - Data page (DP), XDP
  - Peripheral data
  - Stack pointer registers

○ MOV instructions

MOV src, dst



○ memory addressing modes

Memory Addressing Modes in C55x

- Absolute (constant) addressing
- Direct (offset) addressing
- Indirect (pointer) addressing

○ MAC instructions

○ parallelism and pipelining

○ FIR filtering

- TMS320C55x architecture
    - ~~CPU [slides 76-82]~~
    - ~~bus [slides 83-95]~~
    - arithmetic operations with conditions

~~Part 6: Design and Development Tools for Digital Signal Processors [all slides]~~

2024年5月4日 14:42

~~Part6-DesignAndDevelopmentToolsForDSPs(3)~~

- ~~Development environments~~
- ~~Debugging facilities~~
    - ~~code composer studio~~
- ~~Assembly language tools~~
- ~~C development tools~~

ref sheet

2024年5月7日 0:31

Reference Sheet