Division Algorithms and Hardware Implementations

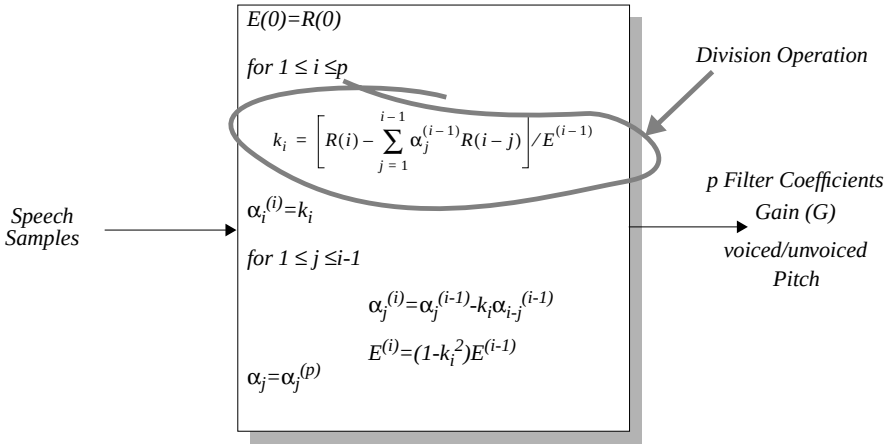
Sherif Galal

Dung Pham

EE 213A: Advanced DSP Circuit Design

Motivation

- **Implementation of LPC speech coder**
- **Division operation required in Levinson-Durbin algorithm**

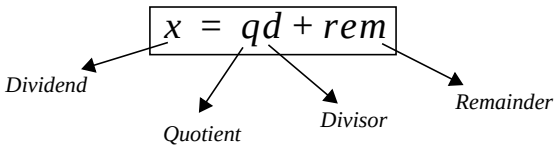


Basic Division Algorithms

- **Digit Recurrence Algorithm**
  - **Restoring Division**
  - **Non-Restoring Division**
  - **SRT Division (Sweeney, Robertson, and Tocher)**
- **Multiplicative Algorithm**
- **Approximation Algorithms**
- **CORDIC Algorithm**
- **Continued Product Algorithm**

Useful site
http://www.ecs.umass.edu/ece/koren/arith/simulator

Definitions and Notations



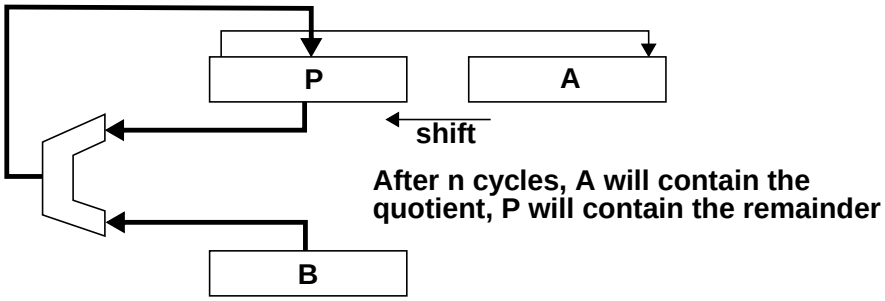
$|rem| < |d| ulp$

unit in the last position
ulp=1 for integer quotient
ulp=r^{-n} for radix-r representation with n-digit quotient

- **Two types of division operations**
  - **Integer division: with integer operands and result**
  - **Fractional division: operands and results are fractions**
- **Any division algorithm can be carried out independent of**
  - **Position of the decimal point**
  - **sign of operands**

Restoring Division Algorithm

- Put x in register A, d in register B, 0 in register P, and perform n divide steps (n is the quotient wordlength)
- Each step consists of
  - (i) Shift the register pair (P,A) one bit left
  - (ii) Subtract the contents of B from P, put the result back in P
  - (iii) If the result is -ve, set the low-order bit of A to 0 otherwise to 0
  - (iv) If the result is -ve, restore the old value of P by adding the contents of B back in P



After n cycles, A will contain the quotient, P will contain the remainder

Restoring Division Example

P	A	Operation
00000	1110	Divide 14 = 1110 by 3 = 11. B register always contains 0011
00001	110	step 1(i): shift
-00011		step 1(ii): subtract

-00010	1100	step 1(iii): quotient is -ve, set quotient bit to 0
00001	1100	step 1(iv): restore
00011	100	step 2(i): shift
-00011		step 2(ii): subtract

00000	1001	step 2(iii): quotient is +ve, set quotient bit to 1
00001	001	step 3(i): shift
-00011		step 3(ii): subtract

-00010	0010	step 3(iii): quotient is -ve, set quotient bit to 0
00001	0010	step 3(iv): restore
00010	010	step 4(i): shift
-00011		step 4(ii): subtract

-00001	0100	step 4(iii): quotient is -ve, set quotient bit to 0
00010	0100	step 4(iv): restore

- The quotient is 0100 and the remainder is 00010
- The name *restoring* because if subtraction by *b* yields a negative result, the *P* register is restored by adding *b* back

Non-Restoring Division Algorithm

- A variant that skips the restoring step and instead works with negative residuals
- If P is negative
  - (i-a) Shift the register pair (P,A) one bit left
  - (ii-a) Add the contents of register B to P
- If P is positive
  - (i-b) Shift the register pair (P,A) one bit left
  - (ii-b) Subtract the contents of register B from P
- (iii) If P is negative, set the low-order bit of A to 0, otherwise set it to 1
- After n cycles
  - The quotient is in A
  - If P is positive, it is the remainder, otherwise it has to be restored (add B to it) to get the remainder

Non-Restoring Division Example

P	A	Operation
00000	1110	Divide 14 = 1110 by 3 = 11. B register always contains 0011
00001	110	step 1(i-b): shift
+00011		step 1(ii-b): subtract b (add two's complement)
11110	1100	step 1(iii): P is negative, so set quotient bit to 0
11101	100	step 2(i-a): shift
+00011		step 2(ii-a): add b
00000	1001	step 2(iii): P is +ve, so set quotient bit to 1
00001	001	step 3(i-b): shift
+11101		step 3(ii-b): subtract b
11110	0010	step 3(iii): P is -ve, so set quotient bit to 0
11100	010	step 4(i-a): shift
+00011		step 4(ii-a): add b
11111	0100	step 4(iii): P is -ve, set quotient bit to 0
+00011		Remainder is negative, so do final restore step
00010		

- The quotient is 0100 and the remainder is 00010
- *restoring* division seems to be more complicated since it involves extra addition in step (iv)
- This is not true since the sign resulting from the subtraction is tested at adder o/p and only if the sum is +ve, it is loaded back to the P register

SRT Division Algorithm

- **Pre-normalization of divisor ($1/2 \le d \le 1$) and dividend ($x < d$)**

For an n-bit quotient, n iterations are needed
Each iteration involves 4 intermediate steps

Start by defining a residual (or partial remainder) w and setting w[0]=x

step1: *One digit left-shift of w[j] to produce rw[j]*
step2: *Determine the quotient q_{j+1} using quotient-digit select function*

$$q_{j+1}=SEL(w[j],d)$$

step3: *Generation of d q_{j+1}*
step4: *Generation of $w[j+1]=rw[j] - d\ q_{j+1}$*

Quotient-digit Select Function

- **The Quotient-digit set plays a crucial role in the complexity of implementation**
- **Restoring algorithm => $0 \le q_j \le r-1$**
- **Non-Restoring algorithm => q_j {-1,1}**
- **SRT quotient-digit selection function**

$$q_{j+1} = \begin{bmatrix} 1 & 1/2 \le 2w[j] \\ 0 & -1/2 \le 2w[j] < 1/2 \\ -1 & 2w[j] < -1/2 \end{bmatrix}$$

- **SRT division is very fast in the case of consecutive zeroes in q ($w[j+1]=rw[j] - d\ q_{j+1}=rw[j]$)**

SRT Division Example

- **Example d=0.1101, x=0.011000**

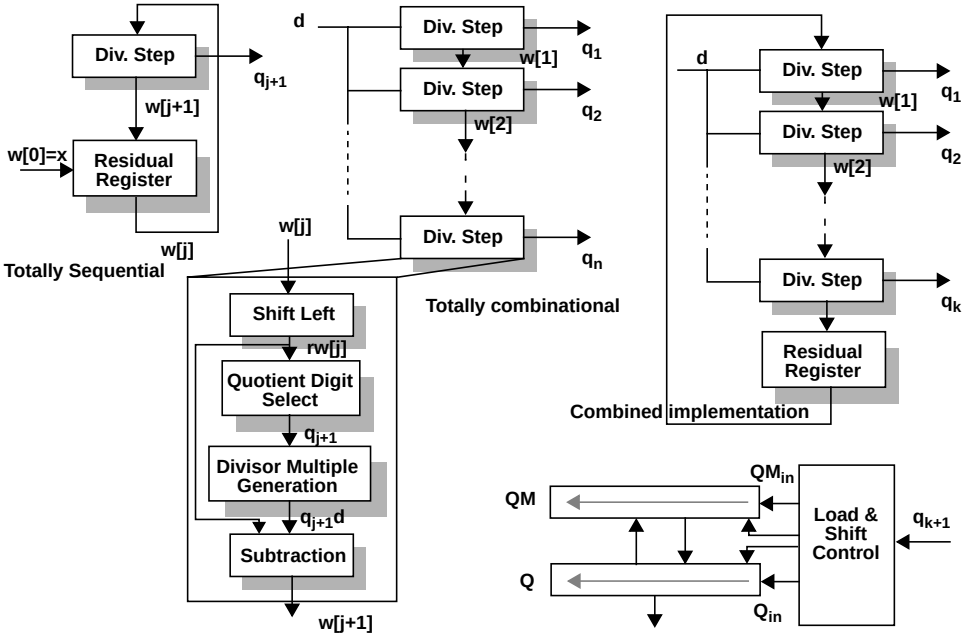
Table 1: Example of radix-2 SRT Division

j	2w[j]	q_{j+1}	Q[j]	QM[j]
0	0.110000	1	0	0
1	1.111000	0	0.1	0.0
2	1.110000	0	0.10	0.01
3	1.100000	0	0.100	0.011
4	1.000000	-1	0.1000	0.0111
5	1.101000	0	0.01111	0.01110
6	1.010000	-1	0.011110	0.011101

- **Quotient Conversion (on-the-fly conversion algorithm) with initial conditions Q[0]=QM[0]=0**

$$Q[k+1] = \begin{bmatrix} Q[k] + q_{k+1} r^{-(k+1)} & q_{k+1} \geq 0 \\ QM[k] + (r - |q_{k+1}|) r^{-(k+1)} & q_{k+1} < 0 \end{bmatrix} \quad QM[k+1] = \begin{bmatrix} Q[k] + (q_{k+1} - 1) r^{-(k+1)} & q_{k+1} > 0 \\ QM[k] + ((r-1) - |q_{k+1}|) r^{-(k+1)} & q_{k+1} \leq 0 \end{bmatrix}$$

SRT Division Implementation



References

- **Stuart F. Oberman, and Michael J. Flynn, “Division Algorithms and Implementations ,” IEEE Transactions on Computers, vol. 46, no. 8, August 1997.**
- **“Computer Arithmetic: A Quantitative Approach” by John L. Hennessey & David P. Patterson, Second Edition**
- **COMS 252A Course Notes “ Digital Arithmetic ”, Professor Ercegovac**