



UNIVERSITY OF MASSACHUSETTS
Dept. of Electrical & Computer Engineering

Digital Computer Arithmetic

ECE 666

Part 7a
Fast Division - I

Israel Koren
Spring 2008

ECE666/Koren Part.7a.1

Copyright 2008 Koren

Fast Division - SRT Algorithm

◆ 2 approaches:

- * First - conventional - uses add/subtract+shift, number of operations linearly proportional to word size n
- * Second - uses multiplication, number of operations logarithmic in n , but each step more complex
- * **SRT** - first approach

◆ Most well known division algorithm - named after Sweeney, Robertson, and Tocher

◆ Speed up nonrestoring division (n add/subtracts)
- allows 0 as a quotient digit - no add/subtract:

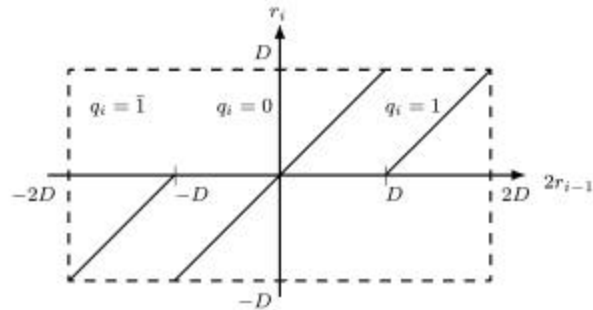
$$q_i = \begin{cases} 1 & \text{if } 2r_{i-1} \geq D \\ 0 & \text{if } -D \leq 2r_{i-1} < D \\ \bar{1} & \text{if } 2r_{i-1} < -D \end{cases}$$

$$r_i = 2r_{i-1} - q_i \cdot D$$

ECE666/Koren Part.7a.2

Copyright 2008 Koren

Modified Nonrestoring Division



- ◆ **Problem:** full comparison of $2r_{i-1}$ with either D or $-D$ required
- ◆ **Solution:** restricting D to normalized fraction $1/2 \leq |D| < 1$
- ◆ Region of $2r_{i-1}$ for which $q_i=0$ reduced to

$$-D \leq -\frac{1}{2} \leq 2r_{i-1} < \frac{1}{2} \leq D$$

ECE666/Koren Part.7a.3

Copyright 2008 Koren

Modified Nonrestoring [®] SRT

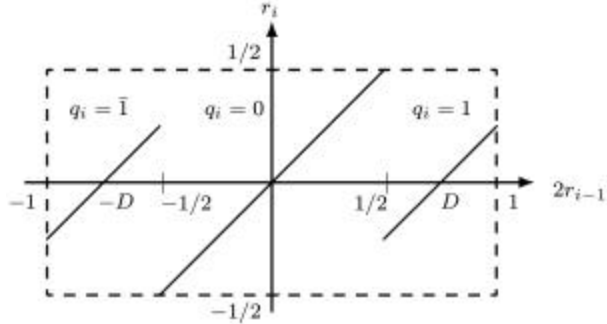
- ◆ **Advantage:** Comparing partial remainder $2r_{i-1}$ to $1/2$ or $-1/2$, not D or $-D$
- ◆ Binary fraction in two's complement representation
 - * $\geq 1/2$ if and only if it starts with 0.1
 - * $\leq -1/2$ if and only if it starts with 1.0
- ◆ Only 2 bits of $2r_{i-1}$ examined - not full comparison between $2r_{i-1}$ and D
 - * In some cases (e.g., dividend $X > 1/2$) - shifted partial remainder needs an integer bit in addition to sign bit - 3 bits of $2r_{i-1}$ examined
- ◆ Selecting quotient digit:

$$q_i = \begin{cases} 1 & \text{if } 2r_{i-1} \geq 1/2 \\ 0 & \text{if } -1/2 \leq 2r_{i-1} < 1/2 \\ \bar{1} & \text{if } 2r_{i-1} < -1/2. \end{cases}$$

ECE666/Koren Part.7a.4

Copyright 2008 Koren

SRT Division Algorithm



◆ Quotient digits selected so $|r_i| \leq |D|$ final remainder $< |D|$

◆ Process starts with normalized divisor - normalizing partial remainder by shifting over leading 0's/1's if positive/negative

◆ Example:

* $2r_{i-1} = 0.001xxxx$ ($x = 0/1$); $2r_{i-1} < 1/2$ - set $q_i = 0$, $2r_i = 0.01xxxx$ and so on

* $2r_{i-1} = 1.110xxxx$; $2r_{i-1} > -1/2$ - set $q_i = 0$, $2r_i = 1.10xxxx$

◆ SRT is nonrestoring division with normalized divisor and remainder

Extension to Negative Divisors

$$q_i = \begin{cases} 0 & \text{if } |2r_{i-1}| < 1/2 \\ 1 & \text{if } |2r_{i-1}| \geq 1/2 \text{ \& } r_{i-1} \text{ and } D \text{ have the same sign} \\ \bar{1} & \text{if } |2r_{i-1}| \geq 1/2 \text{ \& } r_{i-1} \text{ and } D \text{ have opposite signs} \end{cases}$$

◆ Example:

Dividend

$X = (0.0101)_2 = 5/16$

Divisor

$D = (0.1100)_2 = 3/4$

$r_0 = X$	0	.0	1	0	1		
$2r_0$	0	.1	0	1	0	$\geq 1/2$ set $q_1 = 1$	
Add $-D$	+	1	.0	1	0	0	
r_1	1	.1	1	1	0		
$2r_1 = r_2$	1	.1	1	0	0	$\geq -1/2$ set $q_2 = 0$	
$2r_2 = r_3$	1	.1	0	0	0	$\geq -1/2$ set $q_3 = 0$	
$2r_3$	1	.0	0	0	0	$< -1/2$ set $q_4 = \bar{1}$	
Add D	+	0	.1	1	0	0	
r_4	1	.1	1	0	0	negative remainder & positive X	
Add D	+	0	.1	1	0	0	correction
r_4	0	.1	0	0	0	corrected final remainder	

◆ Before correction $Q = 0.100\bar{1}$ - minimal SD repr. of $Q = 0.0111$ - minimal number of add/subtracts

◆ After correction, $Q = 0.0111\text{-ulp} = 0.0110_2 = 3/8$; final remainder = $1/2 \times 2^{-4} = 1/32$

Example

◆ $X=(0.00111111)_2=63/256$ $D=(0.1001)_2=9/16$

$r_0 = X$	0	.0	0	1	1	1	1	1	1	
$2r_0$	0	.0	1	1	1	1	1	1	0	$< 1/2$ set $q_1 = 0$
$2r_1$	0	.1	1	1	1	1	1	0	0	$\geq 1/2$ set $q_2 = 1$
Add $-D$ +	1	.0	1	1	1					
r_2	0	.0	1	1	0	1	1	0	0	
$2r_2$	0	.1	1	0	1	1	0	0	0	$\geq 1/2$ set $q_3 = 1$
Add $-D$ +	1	.0	1	1	1					
r_3	0	.0	1	0	0	1	0	0	0	
$2r_3$	0	.1	0	0	1	0	0	0	0	$\geq 1/2$ set $q_4 = 1$
Add $-D$ +	1	.0	1	1	1					
r_4	0	.0	0	0	0	0	0	0	0	zero final remainder

- ◆ $Q = 0.0111_2 = 7/16$ - not a minimal representation in SD form
- ◆ **Conclusion:** Number of add/subtracts can be reduced further

Properties of SRT

- ◆ **Based on simulations and analysis:**
- ◆ **1.** Average "shift" = 2.67 - $n/2.67$ operations for dividend of length n
 - * $24/2.67 \sim 9$ operations on average for $n=24$
- ◆ **2.** Actual number of operations depends on divisor D - smallest when $17/28 \leq D \leq 3/4$ - average shift of 3
- ◆ If D out of range $(3/5 \leq D \leq 3/4)$ - SRT can be modified to reduce number of add/subtracts
- ◆ **2** ways to modify SRT

Two Modifications of SRT

- ◆ **Scheme 1:** In some steps during division -
 - * If D too small - use a multiple of D like $2D$
 - * If D too large - use $D/2$
 - * Subtracting $2D$ ($D/2$) instead of D - equivalent to performing subtraction one position earlier (later)
- ◆ **Motivation for Scheme 1:**
 - * Small D may generate a sequence of 1's in quotient one bit at a time, with subtract operation per each bit
 - * Subtracting $2D$ instead of D (equivalent to subtracting D in previous step) may generate negative partial remainder, generating sequence of 0's as quotient bits while normalizing partial remainder
- ◆ **Scheme 2:** Change comparison constant $K=1/2$ if D outside optimal range - allowed because ratio D/K matters - partial remainder compared to K not D

Example - Scheme 1 (Using 2D)

- ◆ Same as previous example -
- ◆ $X=(0.00111111)_2=63/256$ $D=(0.1001)_2=9/16$

$r_0 = X$	0	.0	0	1	1	1	1	1	1		
$2r_0$		0	.0	1	1	1	1	1	1	$< 1/2$ set $q_1 = 0$	
$2r_1$	0	0	.1	1	1	1	1	1	0	subtract $2D$	
Add $-2D$ +	1	0	.1	1	1					instead of D	
r_2	1	1	.1	1	0	1	1	1	0	0	set $q_1 = 1$ and $q_2 = 0$
$2r_2$		1	.1	0	1	1	1	0	0	0	set $q_3 = 0$
$2r_3$		1	.0	1	1	1	0	0	0	0	$\leq -1/2$ set $q_4 = \bar{1}$
Add D +	0	.1	0	0	1						
r_4		0	.0	0	0	0	0	0	0	0	zero final remainder

- ◆ $Q = 0.100\bar{1}_2 = 7/16$ - minimal **SD** representation

Scheme 1 (Using D/2)

- ◆ Large D - one 0 in sequence of 1 's in quotient may result in 2 consecutive add/subtracts instead of one
- ◆ Adding $D/2$ instead of D for last 1 before the single 0 - equivalent to performing addition one position later - may generate negative partial remainder
- ◆ Allows properly handling single 0
- ◆ Then continue normalizing partial remainder until end of sequence of 1 's

Example

- ◆ $X=(0.01100)_2=3/8$; $D=(0.11101)_2=29/32$
- ◆ Correct 5-bit quotient - $Q=(0.01101)_2=13/32$
- ◆ Applying basic **SRT** algorithm - $Q=0.10\bar{1}1\bar{1}$ - single 0 not handled efficiently

- ◆ Using multiple $D/2$ -

$r_0 = X$	0 .0 1 1 0 0		
$2r_0$	0 .1 1 0 0 0	$\geq 1/2$	set $q_1 = 1$
Add $-D$	+ 1 .0 0 0 1 1		
r_1	1 .1 1 0 1 1		
$2r_1$	1 .1 0 1 1 0		set $q_2 = 0$
$2r_2$	1 .0 1 1 0 0	0	add $D/2$ ($q_3 = \bar{1}$)
Add $D/2$	+ 0 .0 1 1 1 0	1	instead of D
r_3	1 .1 1 0 1 0	1	set $q_3 = 0$ and
$2r_3$	1 .1 0 1 0 1		$q_4 = \bar{1}$
$2r_4$	1 .0 1 0 1 0		$\leq -1/2$ set $q_5 = \bar{1}$
Add D	+ 0 .1 1 1 0 1		
r_5	0 .0 0 1 1 1		final remainder = $7/32 \cdot 2^{-5}$

- ◆ $Q = (0.100\bar{1}\bar{1})_2=13/32$ - single 0 handled properly

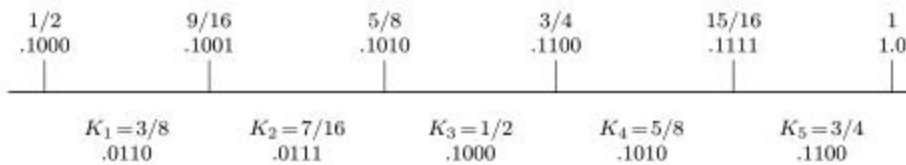
Implementing Scheme 1

- ◆ Two adders needed
 - * One to add or subtract D
 - * Second to add/subtract $2D$ if D too small (starts with 0.10 in its true form) or add/subtract $D/2$ if D too large (starts with 0.11)
- ◆ Output of primary adder used, unless output of alternate adder has larger normalizing shift
- ◆ Additional multiples of D possible - $3D/2$ or $3D/4$
- ◆ Provide higher overall average shift - about 3.7
 - but more complex implementation

Modifying SRT - Scheme 2

- ◆ For $K=1/2$, ratio D/K in optimal range $3/5 \leq D \leq 3/4$ is
 $6/5 \leq D/K = D/(1/2) \leq 3/2$ or
 $(6/5)K \leq D \leq (3/2)K$
- ◆ If D not in optimal range for $K=1/2$ - choose a different comparison constant K
- ◆ Region $1/2 \leq |D| < 1$ can be divided into 5 (not equally sized) sub-regions
- ◆ Each has a different comparison constant K_i

Division into Sub-regions



- ◆ 4 bits of divisor examined for selecting comparison constant
- ◆ It has only 4 bits compared to 4 most significant bits of remainder
- ◆ Determination of sub-regions for divisor and comparison constants - numerical search
- ◆ **Reason:** Both are binary fractions with small number of bits to simplify division algorithm

Example

- ◆ $X=(0.00111111)_2=63/256$; $D=(0.1001)_2=9/16$
- ◆ Appropriate comparison constant - $K_2=7/16=0.0111_2$
- ◆ If remainder negative - compare to two's complement of $K_2 = 1.1001_2$

$r_0 = X$	0	.0	0	1	1	1	1	1	1	
$2r_0$	0	.0	1	1	1	1	1	1	0	≥ 0.0111 set $q_1 = 1$
Add $-D$	+	1	.0	1	1	1				
r_1		1	.1	1	1	0	1	1	1	0
$2r_1 = r_2$		1	.1	1	0	1	1	1	0	≥ 1.1001 set $q_2 = 0$
$2r_2 = r_3$		1	.1	0	1	1	1	0	0	≥ 1.1001 set $q_3 = 0$
$2r_3$		1	.0	1	1	1	0	0	0	< 1.1001 set $q_4 = \bar{1}$
Add D	+	0	.1	0	0	1				
r_4		0	.0	0	0	0	0	0	0	zero final remainder

- ◆ $Q=0.100\bar{1}=0.0111_2=7/16$ - minimal **SD** form