Basic Division Scheme

台灣大學電子所
吳安宇
For Advanced VLSI Course
2002
Outline

- Shift/subtract division algorithm.
- Programmed division.
- Restoring hardware dividers.
- Nonstoring and signed division.
- Division by constants
- Radix-2 SRT division.
- High-Radix division.
Shift/Subtract Division Algorithms

- **z** Dividend: $z_{2k-1}z_{2k-2}\ldots z_1z_0$
- **d** Divisor: $d_{k-1}d_{k-2}\ldots d_1d_0$
- **q** Quotient: $q_{k-1}q_{k-2}\ldots q_1q_0$
- **s** Remainder[$z - (d \times q)$]: $s_{k-1}s_{k-2}\ldots s_1s_0$

Division can be done by a sequence of shifts and subtraction.
Overflow Check

- **Multiplication:**
  - product of two $k$-bit numbers is always $2k$ bits.
  \[
  z = [(d \times q)] + s \quad \text{integer}
  \]
  \[
  z < (2^k - 1)d + d = 2^k d
  \]

- **Division:**
  - quotient of a $2k$-bit number divided by a $k$-bit number may more than $k$ bits.
  \[
  2^{-2k} z = [(2^{-k} d) \times (2^{-k} q)] + 2^{-2k} s \quad \text{fractions}
  \]
  \[
  z_{\text{frac}} = [(d_{\text{frac}} \times q_{\text{frac}})] + 2^{-k} s_{\text{frac}}
  \]
  \[
  Z_{\text{frac}} < d_{\text{frac}}
  \]
Sequential Division Algorithm

- Left shift *partial remainder*, align to the term to be subtracted.

\[ S^{(j)} = 2s^{(j-1)} - q_{k-j}(2^k d) \quad \text{with} \quad s^{(0)} = z \quad \text{and} \quad s^{(k)} = 2^k s \]

| Shift left | Subtract |

After \( k \) iteration

\[ S^{(k)} = 2^k s^{(0)} - q(2^k d) = 2^k [z - (q \times d)] = 2^k s \]

Fractional version

\[ S^{(j)}_{\text{frac}} = 2s^{(j-1)}_{\text{frac}} - q_j d_{\text{frac}} \quad \text{with} \quad s^{(0)}_{\text{frac}} = z_{\text{frac}} \quad \text{and} \quad s^{(k)}_{\text{frac}} = 2^k s_{\text{frac}} \]
Example of sequential division with integer and fractional operands

<table>
<thead>
<tr>
<th>Integer division</th>
<th>Fractional division</th>
</tr>
</thead>
<tbody>
<tr>
<td>$z$</td>
<td>$Z_{\text{frac}}$</td>
</tr>
<tr>
<td>$2^4 d$</td>
<td>$d_{\text{frac}}$</td>
</tr>
<tr>
<td>$s(0)$</td>
<td>$s(0)$</td>
</tr>
<tr>
<td>$2s(0)$</td>
<td>$2s(0)$</td>
</tr>
<tr>
<td>$-q_3 2^4 d$</td>
<td>$-q_{-1} d$</td>
</tr>
<tr>
<td>$s(1)$</td>
<td>$s(1)$</td>
</tr>
<tr>
<td>$2s(1)$</td>
<td>$2s(1)$</td>
</tr>
<tr>
<td>$-q_2 2^4 d$</td>
<td>$-q_{-2} d$</td>
</tr>
<tr>
<td>$s(2)$</td>
<td>$s(2)$</td>
</tr>
<tr>
<td>$2s(2)$</td>
<td>$2s(2)$</td>
</tr>
<tr>
<td>$-q_1 2^4 d$</td>
<td>$-q_{-3} d$</td>
</tr>
<tr>
<td>$s(3)$</td>
<td>$s(3)$</td>
</tr>
<tr>
<td>$2s(3)$</td>
<td>$2s(3)$</td>
</tr>
<tr>
<td>$-q_0 2^4 d$</td>
<td>$-q_{-4} d$</td>
</tr>
<tr>
<td>$s(4)$</td>
<td>$s_{\text{frac}}$</td>
</tr>
<tr>
<td>$s$</td>
<td>$s_{\text{frac}}$</td>
</tr>
<tr>
<td>$q$</td>
<td>$q_{\text{frac}}$</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| $z$               | $01110101$          |
| $2^4 d$           | $1010$             |
| $s(0)$            | $01110101$          |
| $2s(0)$           | $01110101$          |
| $-q_3 2^4 d$      | $1010 \{ q_3 = 1 \}$ |
| $s(1)$            | $0100101$           |
| $2s(1)$           | $0100101$           |
| $-q_2 2^4 d$      | $00000 \{ q_2 = 0 \}$ |
| $s(2)$            | $100101$            |
| $2s(2)$           | $100101$            |
| $-q_1 2^4 d$      | $1010 \{ q_1 = 1 \}$ |
| $s(3)$            | $10001$             |
| $2s(3)$           | $10001$             |
| $-q_0 2^4 d$      | $1010 \{ q_0 = 1 \}$ |
| $s(4)$            | $0111$              |
| $s_{\text{frac}}$| $0111$              |
| $q_{\text{frac}}$| $1011$              |
| $Z_{\text{frac}}$| $01110101$          |
| $d_{\text{frac}}$| $1010$             |
| $s(0)$            | $01110101$          |
| $2s(0)$           | $01110101$          |
| $-q_{-1} d$       | $1010 \{ q_{-1} = 1 \}$ |
| $s(1)$            | $01000101$          |
| $2s(1)$           | $01000101$          |
| $-q_{-2} d$       | $00000 \{ q_{-2} = 0 \}$ |
| $s(2)$            | $100101$            |
| $2s(2)$           | $100101$            |
| $-q_{-3} d$       | $1010 \{ q_{-3} = 1 \}$ |
| $s(3)$            | $100001$            |
| $2s(3)$           | $100001$            |
| $-q_{-4} d$       | $1010 \{ q_{-4} = 1 \}$ |
| $s(4)$            | $0111$              |
| $s_{\text{frac}}$| $000000111$         |
| $q_{\text{frac}}$| $1011$              |
Programmed Division

{Using left shifts, divide unsigned 2k-bit dividend, z_high, z_low, storing the k-bit quotient and remainder.}

Registers:
- R0 holds 0 for counter
- Rs for divisor
- Rq for z_high & remainder
- Rq for z_low & quotient

{Load operands into registers Rd, Rs, and Rq}

\[
\text{div: load } \quad \text{Rd with divisor}
\]
\[
\text{load } \quad \text{Rs with z_high}
\]
\[
\text{load } \quad \text{Rq with z_low}
\]

{Check for exceptions}

\[
\text{branch } d_{by\_0} \text{ if } Rd = R0
\]
\[
\text{branch } d_{ovf} \text{ if } Rs > Rd
\]

{Initialize counter}

\[
\text{load } k \text{ into Rc}
\]

{Begin division loop}

\[
\text{d_loop: shift } \quad \text{Rq left 1 \{zero to LSB, MSB to carry\}}
\]
\[
\text{rotate } \quad \text{Rs left 1 \{carry to LSB, MSB to carry\}}
\]
\[
\text{skip } \quad \text{if carry = 1}
\]
\[
\text{branch } \quad \text{no_sub if } Rs < Rd
\]
\[
\text{sub } \quad \text{Rd from Rs}
\]
\[
\text{incr } \quad \text{Rq \{set quotient digit to 1\}}
\]
\[
\text{decr } \quad \text{Rc \{decrement counter by 1\}}
\]
\[
\text{branch } \quad \text{d_loop if } Rc \neq 0
\]

{Store the quotient and remainder}

\[
\text{store } \quad \text{Rq into quotient}
\]
\[
\text{store } \quad \text{Rs into remainder}
\]
\[
\text{d_{by\_0}: } \
\text{d_{ovf}: } \
\text{d_{done}: } 
\]
Programmed Division (con’t)

- Use *shift* and *add* to perform integer division by a processor.
- Two $k$-bit register to store the *partial remainder* and the *quotient*.
Restoring Hardware Dividers

- The basic eq. for signed division is
  \[ (d \times q) + s \quad \text{along with} \quad \text{sign}(s) = \text{sign}(z) \quad \text{and} \quad |s| < |d| \]

- For example
  
  \[
  \begin{align*}
  z &= 5 \quad d = 3 \quad \Rightarrow \quad q = 1 \quad s = 2 \\
  z &= 5 \quad d = -3 \quad \Rightarrow \quad q = -1 \quad s = 2 \\
  z &= -5 \quad d = 3 \quad \Rightarrow \quad q = -1 \quad s = -2 \\
  z &= -5 \quad d = -3 \quad \Rightarrow \quad q = 1 \quad s = -2
  \end{align*}
  \]

- The magnitudes of \( q \) and \( s \) are unaffected by the input signs.
Restoring Hardware Dividers (con’t)

- Because the magnitudes of $q$ and $s$ are unaffected by the input signs. Signed division can be converted into unsigned values and, at the end, the signs is determined by the sign bits or via complementation.

- This is the method of choice with the restoring division algorithm.
Restoring Hardware Dividers (con’t)

<table>
<thead>
<tr>
<th>z</th>
<th>0 1 1 1 0 1 0 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2^4d</td>
<td>0 1 0 1 0</td>
</tr>
<tr>
<td>-2^4d</td>
<td>1 0 1 1 0</td>
</tr>
<tr>
<td>s(0)</td>
<td>0 0 1 1 1 0 1 0 1</td>
</tr>
<tr>
<td>2s(0)</td>
<td>0 1 1 1 0 1 0 1</td>
</tr>
<tr>
<td>+(-2^4d)</td>
<td>1 0 1 1 0</td>
</tr>
<tr>
<td>s(1)</td>
<td>0 1 0 0 1 0 1 0 1</td>
</tr>
<tr>
<td>2s(1)</td>
<td>0 1 0 0 1 0 1 0 1</td>
</tr>
<tr>
<td>+(-2^4d)</td>
<td>1 0 1 1 0</td>
</tr>
<tr>
<td>s(2)</td>
<td>1 1 1 1 1 0 1 0 1</td>
</tr>
<tr>
<td>2s(2)</td>
<td>1 1 1 1 1 0 1 0 1</td>
</tr>
<tr>
<td>+(-2^4d)</td>
<td>1 0 1 1 0</td>
</tr>
<tr>
<td>s(3)</td>
<td>0 1 0 0 1 1 0 1 0 1</td>
</tr>
<tr>
<td>2s(3)</td>
<td>1 0 0 0 1 1 0 1 0 1</td>
</tr>
<tr>
<td>+(-2^4d)</td>
<td>1 0 1 1 0</td>
</tr>
<tr>
<td>s(4)</td>
<td>0 0 1 1 1 0 1 0 1 1</td>
</tr>
<tr>
<td>q</td>
<td>1 0 1 1 1 0 1 0 1 1</td>
</tr>
</tbody>
</table>

No overflow, since: (0111)_{two} < (1010)_{two}

Positive, so set q_3 = 1

Negative, so set q_2 = 0 and restore

Positive, so set q_1 = 1

Positive, so set q_0 = 1

Trial difference

MSB of

Partial

Complement

Sub

k-bit adder

Load

Quotient

k

c_{out}
c_{in}
Drawback of Restoring Division

- Timing issues because each $k$ cycles must be long enough to allow following events in sequence:
  - Shifting of the registers.
  - Propagation of signals through the adder (check carry)
  - Storing of the quotient digit. (storing)
- So the sign of the trial difference must be sampled near the negative edge. (drawback)
- To avoid such timing issues, nonrestoring division algorithm can be used.
Nonrestoring and Signed Division

- Always store difference in the partial remainder register.
- Allow partial remainder being temporarily incorrect (hence the name “nonrestoring”).
- For example:

<table>
<thead>
<tr>
<th>[Restoring]:</th>
<th>[Nonrestoring]:</th>
</tr>
</thead>
<tbody>
<tr>
<td>cycle $n$</td>
<td>cycle $n$</td>
</tr>
<tr>
<td>incorrect partial remainder $u - 2^k d$</td>
<td>incorrect partial remainder $u - 2^k d$</td>
</tr>
<tr>
<td>restore to $u$</td>
<td>skip restore</td>
</tr>
<tr>
<td>cycle $n+1$</td>
<td>cycle $n+1$</td>
</tr>
<tr>
<td>$2u - 2^k d$</td>
<td>$2(u - 2^k d) + 2^k d = 2u - 2^k d$ (the same as restoring)</td>
</tr>
</tbody>
</table>
Nonrestoring and Signed Division (con’t)

- Quotient digits are selected from the set \{1, -1\}, \(1 \rightarrow \text{sub}, -1 \rightarrow \text{add}\).
- Goal is to end up with a remainder matches the sign of the dividend. (dividend can be positive or negative).
- The rule for quotient digit selection becomes:

\[
\text{if } \text{sign}(s) = \text{sign}(d) \text{ then } q_{k-j} = 1 \text{ else } q_{k-j} = -1
\]
Nonrestoring Unsigned division example

<table>
<thead>
<tr>
<th></th>
<th>0 1 1 1 0 1 0 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$z$</td>
<td>0 1 0 1 0</td>
</tr>
<tr>
<td>$2^{4}d$</td>
<td>0 1 1 1 0</td>
</tr>
<tr>
<td>$-2^{4}d$</td>
<td>1 0 1 1 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0 0 1 1 1 0 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s^{(0)}$</td>
<td>0 1 1 1 0 1 0 1</td>
</tr>
<tr>
<td>$2s^{(0)}$</td>
<td>0 1 1 1 0 1 0 1</td>
</tr>
<tr>
<td>$+(-2^{4}d)$</td>
<td>1 0 1 1 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0 0 1 0 0 1 0 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s^{(1)}$</td>
<td>0 1 0 0 1 0 1 0</td>
</tr>
<tr>
<td>$2s^{(1)}$</td>
<td>0 1 0 0 1 0 1 0</td>
</tr>
<tr>
<td>$+(-2^{4}d)$</td>
<td>1 0 1 1 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>1 1 1 1 1 0 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s^{(2)}$</td>
<td>1 1 1 1 0 1</td>
</tr>
<tr>
<td>$2s^{(2)}$</td>
<td>1 1 1 1 0 1</td>
</tr>
<tr>
<td>$+2^{4}d$</td>
<td>0 1 0 1 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0 1 0 0 0 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s^{(3)}$</td>
<td>0 1 0 0 0 1</td>
</tr>
<tr>
<td>$2s^{(3)}$</td>
<td>0 1 0 0 0 1</td>
</tr>
<tr>
<td>$+(-2^{4}d)$</td>
<td>1 0 1 1 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0 0 1 1 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s^{(4)}$</td>
<td>0 1 1 1</td>
</tr>
<tr>
<td>$q$</td>
<td>1 0 1 1</td>
</tr>
</tbody>
</table>

No overflow, since: $(0111)_{two} < (1010)_{two}$

Positive, so subtract

Positive, so set $q_3 = 1$ and subtract

Negative, so set $q_2 = 0$ and add

Positive, so set $q_1 = 1$ and subtract

Positive, so set $q_0 = 1$
Partial Remainder variation for restoring and nonrestoring division

(a) Restoring.

(b) Nonrestoring.
Nonstoring Signed Division: Two Problems

The quotient with digits 1 and -1 must be converted to standard binary.

If the final remainder $s$ has a sign opposite that of $z$, a correction step addition $±d$ to the remainder and subtraction of $±1$ from the quotient, is needed.

Convert a $k$-digit BSD quotient to a $k$-bit 2’s complement number.

A. replace all -1 digits with 0s to get the k-bit number

$$ p = p_{k-1}p_{k-2}...p_0, p_i \in \{0, 1\} $$

B. complement $p_{k-1}$ and then shift $p$ left by 1 bit, inserting 1 to the LSB, get

$$ q = (\overline{p_{k-1}}p_{k-2}...p_01)_{2's-compl}. $$
Convert a $k$-digit BSD quotient to a $k$-bit 2’s complement number

Proof:

\[
(\overline{p_{k-1}p_{k-2} \cdots p_0}1)_{2's\text{-}compl.} = -(1 - p_{k-1})2^k + 1 + \sum_{i=0}^{k-2} p_i 2^{i+1}
\]

\[
= -(2^k - 1) + 2 \sum_{i=0}^{k-1} p_i 2^i
\]

\[
= \sum_{i=0}^{k-1} (2p_i - 1)2^i
\]

\[
= \sum_{i=0}^{k-1} q_i 2^i = q
\]

Note:  
(1) $q_i = 2p_i - 1$  
(2) $\sum_{i=0}^{k-1} 2^i = 2^k - 1$
Nonstoring Signed Division: example

Dividend = (33)_{10}
Divisor = (-7)_{10}

\[
\begin{array}{c|cccccccc}
& z & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
2^{4}d & 1 & 1 & 0 & 0 & 1 \\
-2^{4}d & 0 & 0 & 1 & 1 & 1 \\
\hline
s^{(0)} & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
2s^{(0)} & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
+2^{4}d & 1 & 1 & 0 & 0 & 1 \\
\hline
s^{(1)} & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\
2s^{(1)} & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\
+(-2^{4}d) & 0 & 0 & 1 & 1 & 1 \\
\hline
s^{(2)} & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
2s^{(2)} & 0 & 0 & 0 & 1 & 0 & 1 \\
+2^{4}d & 1 & 1 & 0 & 0 & 1 \\
\hline
s^{(3)} & 1 & 1 & 0 & 1 & 1 \\
2s^{(3)} & 1 & 0 & 1 & 1 \\
+(-2^{4}d) & 0 & 0 & 1 & 1 \\
\hline
s^{(4)} & 1 & 1 & 1 & 1 & 0 \\
+(-2^{4}d) & 0 & 0 & 1 & 1 & 1 \\
\hline
\end{array}
\]

- sign(s^{(0)}) ≠ sign(d), so set q_3 = -1 and add
- sign(s^{(1)}) = sign(d), so set q_2 = 1 and subtract
- sign(s^{(2)}) ≠ sign(d), so set q_1 = -1 and add
- sign(s^{(3)}) = sign(d), so set q_0 = 1 and subtract

Sign(s^{(4)}) ≠ sign(d)
Corrective subtraction

Remainder = (5)_{10}
Uncorrected BSD quotient

- 1's replaced by 0s
- Add 1 to correct

Quotient = (-4)_{10}
Division by Constants

- Here, we will consider only division by odd integers, since division by even integer can be performed by dividing by a odd integer then shift the result.
- For example: \( K/20 = K/(5*4) = (K/5)*(1/4) = (K/5) >> 2 \).
- If only a limited number of constant divisor are of interest, their reciprocals can be precomputed with an appropriate precision and stored in a table.
Division by Constants (con’t)

Faster constant division can be obtained for many small odd divisors by using that: for each odd integer \( d \) there exists an odd integer \( m \) such that \( d \times m = 2^n - 1 \).

\[
\frac{1}{d} = \frac{m}{2^n - 1} = \frac{m}{2^n (1 - 2^{-n})} = \frac{m}{2^n} (1 + 2^{-n})(1 + 2^{-2n})(1 + 2^{-4n})... 
\]

For example: \( d = 5 \), \( m = 3 \) and \( n = 4 \). Thus for 24 bits of precision,

\[
\frac{z}{5} = \frac{3z}{2^4 - 1} = \frac{3z}{16(1 - 2^{-4})} = \frac{3z}{16} (1 + 2^{-4})(1 + 2^{-8})(1 + 2^{-16})...
\]

Note that the next term \( (1 + 2^{-32}) \) would shift out the entire operand.
Division by Constants (con’t)

- Follow preceding example, to effect division by 5

\[
\begin{align*}
q & \leftarrow z + z \text{ shift-left } 1 \\
q & \leftarrow q + q \text{ shift-right } 4 & \{3z \text{ computed}\} \\
q & \leftarrow q + q \text{ shift-right } 8 & \{3z(1 + 2^{-4})\} \\
q & \leftarrow q + q \text{ shift-right } 16 & \{3z(1 + 2^{-4})(1 + 2^{-8})\} \\
q & \leftarrow q \text{ shift-right } 4 & \{3z(1 + 2^{-4})(1 + 2^{-8})(1 + 2^{16})\} \\
q & \leftarrow q \text{ shift-right } 4 & \{3z(1 + 2^{-4})(1 + 2^{-8})(1 + 2^{-16})/16\}
\end{align*}
\]
Radix-2 SRT Division: Review of nonrestoring division

- Reconsider radix-2 nonrestoring division algorithm for fractional operands.

\[ S^{(j)} = 2s^{(j-1)} - q_{-j}d \quad \text{with} \quad s^{(0)} = z \quad \text{and} \quad s^{(k)} = 2^k s \]

- Quotient is obtained with the digit set \{-1, 1\} and is then converted to the standard digit set \{0, 1\}.

New partial remainder

For \(2s^{(j-1)} = 0\)

\[ q_{-j} = -1 \quad \text{or} \quad q_{-j} = 1 \]

Shifted old partial remainder
Radix-2 SRT Division (con’t)

- Quotient is obtained using digit set \{-1,0,1\}.
- Quotient “0” is selected when \( q_j = 0 \) for \(-d \leq 2s^{(j-1)} < d\)
- Quotient “0” is simple shift, can speed up the division operation.
- But determined \(-d \leq 2s^{(j-1)} < d\) need trial subtraction. Would consume more time than they save!
Radix-2 SRT Division (con’t)

- **SRT**: Sweeney, Roberson, and Tocher discovered SRT division about the same time.

- *Normalized* divisor and *normalized* partial dividend.

- Divisor and partial dividend is limited in the range \([1/2,1)\) or \((-1,-1/2]\).

- Easier comparison can be used due to normalized divisor.
Radix-2 SRT Division (con’t)

Because of normalized divisor, comparison become:

- $2s^{(j-1)} > +\frac{1}{2} = (0.1)_{2's-compl}$ implies $2s^{(j-1)} = (0.1u_{-2}u_{-3}…)$ 2's-compl
- $2s^{(j-1)} < -\frac{1}{2} = (1.1)_{2's-compl}$ implies $2s^{(j-1)} = (1.0u_{-2}u_{-3}…)$ 2's-compl

- $2s^{(j-1)} > +\frac{1}{2}$ is given by $\overline{u}_0u_{-1}$, and $2s^{(j-1)} < -\frac{1}{2}$ is given by $u_0\overline{u}_{-1}$. Much easier!
### Example of Radix-2 SRT Division

| \( z \) | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| \( d \) | 1 | 0 | 1 | 0 |
| \(-d\) | 1 | 0 | 1 | 1 | 0 |

\[
\begin{align*}
\text{s}^{(0)} & : 0.01000 0101 \\
2\text{s}^{(0)} & : 0.10000 1011 \\
+(-d) & : 1.01110 \\
\text{s}^{(1)} & : 1.11110 1011 \\
2\text{s}^{(1)} & : 11010111 \\
\text{s}^{(2)} = 2\text{s}^{(1)} & : 1.11101 0111 \\
2\text{s}^{(2)} & : 11010111 \\
+d & : 0.10100 \\
\text{s}^{(3)} & : 0.01000 0111 \\
2\text{s}^{(3)} & : 0.10000 1110 \\
+(-d) & : 10110 \\
\text{s}^{(4)} = 2\text{s}^{(3)} & : 1.11111 \\
+d & : 0.10100 \\
\text{s}^{(4)} & : 0.10011 \\
\text{s} & : 0.00000 1001 \\
q & : 0.10111 \\
\overline{q} & : 0.01110 \\
\end{align*}
\]

- In \([-1/2, 1/2)\), so OK
- In \([1/2, 1)\), so OK
- \(\geq 1/2\), so set \(q_{-1} = 1\) and subtract
- \(\leq -1/2\), so set \(q_{-3} = -1\) and add
- \(\geq 1/2\), so set \(q_{-4} = 1\) and subtract
- Negative, so add to correct
- Uncorrected BSD quotient
- Convert and subtract ulp
Basic of High-Radix Division

\[ z \text{ Dividend} \quad \bar{z}_{2k-1}z_{2k-2}\ldots z_1z_0 \]

\[ d \text{ Divisor} \quad d_{k-1}d_{k-2}\ldots d_1d_0 \]

\[ q \text{ Quotient} \quad q_{k-1}q_{k-2}\ldots q_1q_0 \]

\[ s \text{ Remainder}[z - (d \times q)] \quad s_{k-1}s_{k-2}\ldots s_1s_0 \]

Ex: Radix-4 division (dot notation)

\[
\begin{array}{c}
\bullet \bullet \bullet \bullet \\
\end{array} \quad \begin{array}{c}
\bullet \bullet \bullet \\
\bullet \bullet \bullet \\
\bullet \bullet \bullet \\
\bullet \bullet \bullet \\
\end{array}
\]

\[ d \]

\[ \bar{z}_{2k-1}z_{2k-2}\ldots z_1z_0 \]

\[ -(q_3q_2)_{\text{two}} \quad d4^1 \]

\[ -(q_1q_0)_{\text{two}} \quad d4^0 \]

\[ s \]
## High-Radix Division example

### Radix-4 integer division

<table>
<thead>
<tr>
<th>$z$</th>
<th>0 1 2 3 1 1 2 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$4^4d$</td>
<td>1 2 0 3</td>
</tr>
</tbody>
</table>

#### $s^{(0)}$

<table>
<thead>
<tr>
<th>$s^{(0)}$</th>
<th>0 1 2 3 1 1 2 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$4s^{(0)}$</td>
<td>0 1 2 3 1 1 2 3</td>
</tr>
<tr>
<td>$-q_3 4^4d$</td>
<td>0 1 2 0 3 ${q_3 = 1}$</td>
</tr>
</tbody>
</table>

#### $s^{(1)}$

<table>
<thead>
<tr>
<th>$s^{(1)}$</th>
<th>0 0 2 2 1 2 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$4s^{(1)}$</td>
<td>0 0 2 2 1 2 3</td>
</tr>
<tr>
<td>$-q_2 4^4d$</td>
<td>0 0 0 0 0 ${q_2 = 0}$</td>
</tr>
</tbody>
</table>

#### $s^{(2)}$

<table>
<thead>
<tr>
<th>$s^{(2)}$</th>
<th>0 2 2 1 2 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$4s^{(2)}$</td>
<td>0 2 2 1 2 3</td>
</tr>
<tr>
<td>$-q_1 4^4d$</td>
<td>0 1 2 0 3 ${q_1 = 1}$</td>
</tr>
</tbody>
</table>

#### $s^{(3)}$

<table>
<thead>
<tr>
<th>$s^{(3)}$</th>
<th>1 0 0 3 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$4s^{(3)}$</td>
<td>1 0 0 3 3</td>
</tr>
<tr>
<td>$-q_0 4^4d$</td>
<td>0 3 0 1 2 ${q_0 = 2}$</td>
</tr>
</tbody>
</table>

#### $s^{(4)}$

<table>
<thead>
<tr>
<th>$s^{(4)}$</th>
<th>1 0 2 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s$</td>
<td>1 0 2 1</td>
</tr>
<tr>
<td>$q$</td>
<td>1 0 1 2</td>
</tr>
</tbody>
</table>

---

### Radix-10 fractional division

<table>
<thead>
<tr>
<th>$Z_{frac}$</th>
<th>.7 0 0 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_{frac}$</td>
<td>.9 9</td>
</tr>
</tbody>
</table>

#### $s^{(0)}$

<table>
<thead>
<tr>
<th>$s^{(0)}$</th>
<th>.7 0 0 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10s^{(0)}$</td>
<td>7.0 0 3</td>
</tr>
<tr>
<td>$-q_{-1}d$</td>
<td>.6 .9 3 ${q_{-1} = 7}$</td>
</tr>
</tbody>
</table>

#### $s^{(1)}$

<table>
<thead>
<tr>
<th>$s^{(1)}$</th>
<th>.0 7 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10s^{(1)}$</td>
<td>0.7 3</td>
</tr>
<tr>
<td>$-q_{-2}d$</td>
<td>.0 .0 0 ${q_{-2} = 0}$</td>
</tr>
</tbody>
</table>

#### $s^{(2)}$

<table>
<thead>
<tr>
<th>$s^{(2)}$</th>
<th>.7 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_{frac}$</td>
<td>.0 .0 7 3</td>
</tr>
<tr>
<td>$q_{frac}$</td>
<td>.7 0</td>
</tr>
</tbody>
</table>
Features of High-Radix Division

- Dividing binary number in radix $2^b$ reduces the cycles required by a factor of $b$, but each cycle is more difficult to implement:
  - The higher radix makes the guessing of the correct quotient digit more difficult.
  - The value to be subtracted are determined sequentially, one per cycle. Possible value to be subtracted become harder to generate.