High Radix Division

Ivor Page[1]

13.1 Division with Redundant Quotient

We are interested in speeding up the division process by generating more bits of the quotient during each iteration. In radix 4 division, two bits of the quotient are generated on each iteration. With a non-redundant quotient, the digits of the quotient are selected from $[-3,3]$, and the partial remainder is shifted two places in each step. The problem is in the selection of the quotient digit. It is difficult to guess correctly without considerable (time-consuming) effort, as we saw in the module on multi-precision arithmetic.

The problem is exacerbated by the need to use the carry-save form for storing the partial remainder. In this form, it isn't easy to tell the sign or the magnitude of the partial remainder. The solution is to make use of a redundant number system for the quotient so that an error in selecting the quotient digit in step i can be tolerated. This error is then corrected in step $i+1$. The greater the amount of redundancy, the larger the allowable error in each quotient digit. This, in turn, implies that fewer bits of the partial remainder need be examined in selecting each quotient digit. The magnitude of the partial remainder can be estimated from only a few of its (most significant) digits.

13.1.1 SRT With Redundant Quotient

We will develop a form of the SRT algorithm using a redundant digit set for the quotient. At first we will not use a carry-save adder. Therefore we will know the sign and magnitude of the partial remainder in every iteration. This exercise will enable us to develop the terminology and the diagrams that are used to describe the more complex division algorithms. Later we will extend the study to include the use of a carry-save adder. Approximations to the partial remainder and divisor will then be used in the selection of the next quotient digit.

In the first algorithm discussed, one bit of the quotient will be generated on each cycle. The quotient digits will be selected from $q_{-i} \in \{-1, 1\}$. The subscripts of q are negative here to reflect the fractional nature of the operands. Since the algorithm does not include the possibility for $q_{-i} = 0$, it is not strictly a version of SRT. That value of the quotient digit would correspond to skipping over 0s or 1s. The second algorithm will include this case.

The algorithm repeatedly applies the recurrence relation:

$$s^{(j)} = 2s^{(j-1)} - q_{-j}d$$

where $s^{(0)} = z$. The notation $s^{(k)}$ means $2^k s$.

[1]University of Texas at Dallas

Note: I have used the notation from the text in this section, but you might want to consider writing $2^k s_k$ in place of $s^{(k)}$. The latter does not make it clear that the k^{th} iteration of s is implied.

Figure 1 shows how the digits are selected according to the value of the old partial remainder left shifted one place, $2s^{(j-1)}$. At the left-hand end of the graph $2s^{(j-1)} = -2d$, and at the right-hand end, its value is $+2d$. For values of $2s^{(j-1)}$ in the range $[-2d, 0)$ the quotient digit value selected is $q_{-j} = -1$. The value $q_{-j} = +1$ is chosen for $2s^{(j-1)}$ in the range $[0, 2d)$. The large dot at $2s^{(j-1)} = 0$ indicates that $q_{-j} = +1$ is chosen for that value of the partial remainder.. The vertical axis of the graph shows the value of the corresponding new partial remainder, $s^{(j)}$. Its value is kept within the range $[-d, +d)$ by the process.

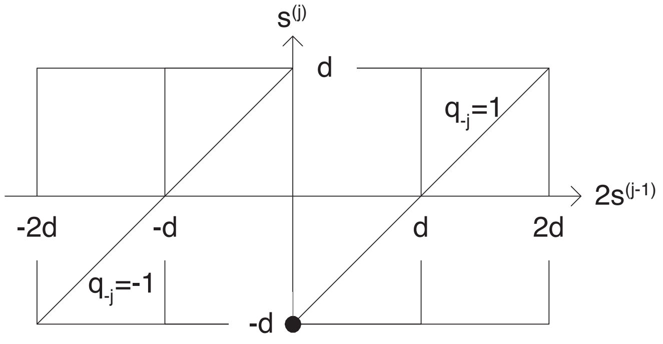

Figure 1: Quotient Digit Selection Graph

For $2s^{(j-1)} \geq 0$, d is subtracted from $2s^{(j-1)}$ and for $2s^{(j-1)} < 0$, d is added to $2s^{(j-1)}$. Each iteration diminishes the magnitude of the partial remainder.

We will not develop this algorithm further but, instead, move on to a version of SRT. The essential development comes from extending the quotient digit set to $q_{-i} \in \{-1, 0, 1\}$. The three values correspond to the addition of d, *do nothing*, and the subtraction of d respectively. The *do nothing* steps correspond to shifting over 0s or 1s. Figure 2 shows the corresponding diagram for quotient digit selection.

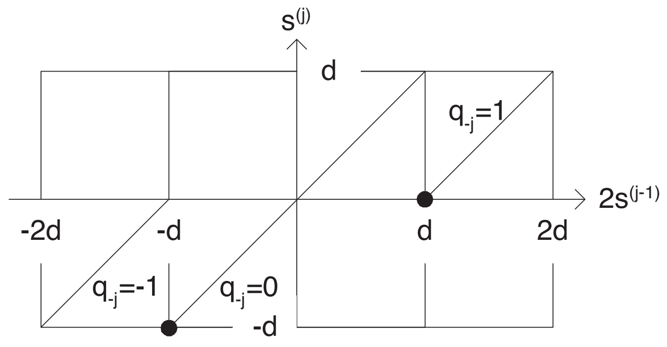

Figure 2: Quotient Digit Selection Graph for $q_{-j} \in \{-1, 0, 1\}$

The diagram shows only the parts of the diagonal lines corresponding to the values of the quotient digit that will be selected. For example, if $2s^{(j-1)} = d/2$, two possible values of the quotient digit could be chosen, $q_j = 0$ or $q_j = 1$. But, as indicated on the diagram, the value $q_{-j} = 0$ is always chosen by the algorithm.

Next, we insist that the partial remainder remain normalized within $[-1/2, 1/2)$. This step is just to develop a technique that we will need later. The initial setup is as follows:

$$\begin{array}{rcl} 1/2 & \leq & d < 1 \text{ positive normalized fraction.} \\ s & \in & [-1/2, 1/2) \text{ the partial remainder will be kept in this range.} \end{array}$$

In limiting the partial remainder, however, an extra bit will be needed at the right hand end of the partial remainder register, u, v. This is because a one-bit right shift of the dividend z might be necessary to achieve the initial normalization. Consider, for example, the problem $010\ 111_2 \div 111_2$, $23 \div 7$ in a 6-bit 2's complement system. The binary point is presumed to be just to the right of the sign bit, so the initial setup would be, $s^{(0)} = 0.010111$, $d = 111000$. s has been extended to 7 bits, including the sign, to preserve all its bits.

Here is how the partial remainder is kept in range and how the quotient digits are selected:

```
if($2s^{(j-1)} < -1/2$)    $q_{-1} = -1$;
else if($2s^{(j-1)} \geq 1/2$)    $q_{-1} = +1$;
else $q_{-1} = 0$;
```

The comparisons are simple and are implemented by examining the two most significant digits of the partial remainder, $2s^{(j-1)} \geq 1/2$ corresponds to $\overline{u_0}u_{-1} = 1$ and $2s^{(j-1)} < -1/2$ corresponds to $u_0\overline{u_{-1}} = 1$.

Note that in the first algorithm, $s \in [-d, d)$, the comparisons were even simpler. We just used the sign of $s^{(j)}$ to determine the next quotient digit.

Figure 3 shows the corresponding quotient digit selection diagram.

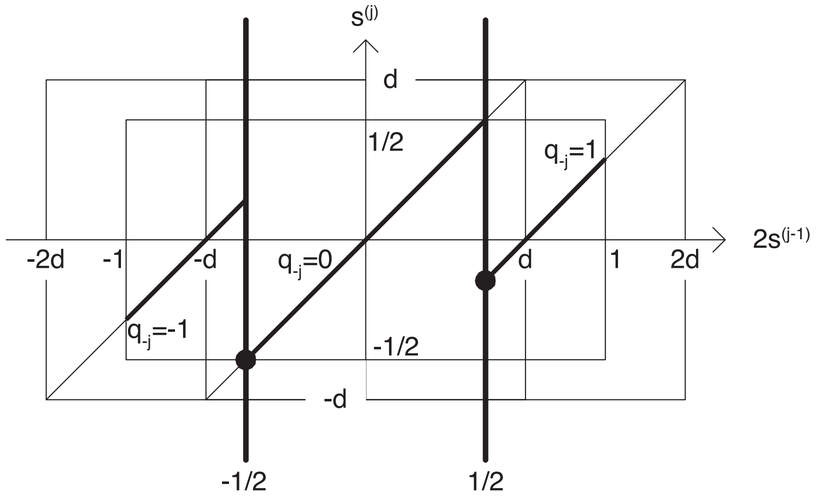

Figure 3: Quotient Digit Selection Graph for SRT

The part of the diagram within the inner rectangle applies to this algorithm. This rectangle corresponds to $2s^{(j-1)} \in [-1, 1)$. The corresponding values of the new partial remainder are $s^{(j)} \in [-1/2, 1/2)$. The three values of the quotient digit are represented by the three bold diagonal line segments. The bold vertical lines demarkate the three regions for quotient digit selection.

Remember, the x-axis represents the left-shifted current partial remainder, $2s^{(j-1)}$. Its value is restricted to $[-1, 1)$, as indicated by the inner rectangle. The sections of diagonal line within the inner rectangle correspond to the three possible values of the quotient digit, $q_{-j} \in \{-1, 0, 1\}$. Only one possibility is used by this algorithm for each point along the x-axis. For a specific point, say $2s^{(j-1)} = 1/4$, draw a vertical line from that point up to the diagonal for $q_{-j} = 0$. Then reflect the intersection onto the y-axis to get the value of the new partial remainder, $s^{(j)}$.

Here is the example from the text using this algorithm, 69 ÷ 10:

Integer Division
z = 0.01000101, d = 0.1010

z	.0100	0101	$z \in [-1/2, 1/2)$, no normalization necessary
d	.1010		$d \in [1/2, 1)$, no normalization necessary
$-d$	1.0110		
$s^{(0)}$	0.0100	0101	
$2s^{(0)}$	0.1000	1010	$\geq 1/2$ so set $q_{-1} = 1$
$+(-d)$	0.1010		subtract
$s^{(1)}$	1.1110	101	
$2s^{(1)}$	1.1101	01	$\in [-1/2, 1/2)$ so set $q_{-2} = 0$, skip
$s^{(2)} = 2s^{(1)}$	1.1101	01	
$2s^{(2)}$	1.1010	1	$\in [-1/2, 1/2)$ so set $q_{-3} = 0$, skip
$s^{(3)} = 2s^{(2)}$	1.1010	1	
$2s^{(3)}$	1.0101		$< -1/2$ so set $q_(- 4) = -1$
$+d$	0.1010		add
$s^{(4)}$	1.1111		Negative
$+d$	0.1010		add to correct
$s^{(4)}$	0.1001		
s	0.0000	1001	
q	0.100T		Uncorrected BSD value = 7
q	0.0110		Converted and corrected

The quotient was 7. We subtracted 1 because of the add-back correction step.

13.2 Using Carry Save Adders

In the next algorithm we make use of the redundancy in the quotient. First, we return to the form of the partial remainder where, $s^{(j)} \in [-d, d)$. The choice of q_{-j} is made as shown in Figure 4.

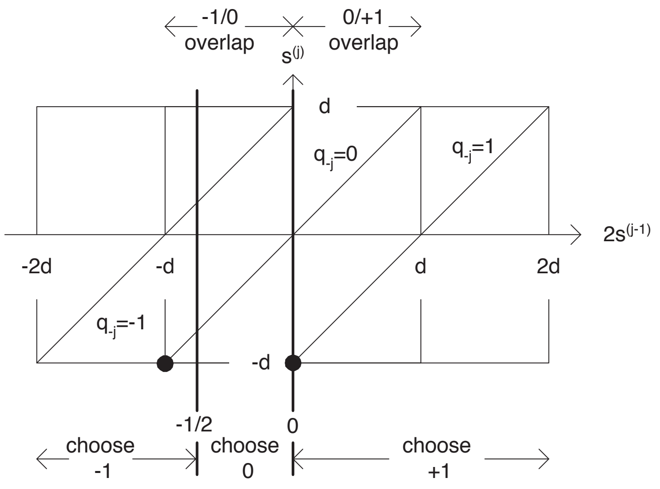

Figure 4: Quotient Digit Selection Graph with choice for $q_{-1} \in \{-1, 0, 1\}$

The two overlap regions allow choices for the next quotient digit from $\{-1, 0\}$ and from $\{0, 1\}$. The algorithm we shall develop uses two constants, $2q^{(j-1)} = -1/2$ and $2q^{(j-1)} = 0$ to define the boundaries between the choices, as indicated on the diagram. There are therefore three ranges for $2q^{(j-1)}$, equal to $[-2d, -1/2)$, $[-1/2, 0)$, and $[0, 2d)$, corresponding to choices of q_{-j} equal to -1, 0, and 1 respectively.

If the partial remainder is kept in carry-save form such that its true value can only be found by adding two registers, it is impossible to select the correct quotient digit in a non-redundant system without performing that add. The redundancy in the quotient number system does, however, allow a correct choice to be made by inspecting the sum of only the most significant 4 bits of the two registers holding the partial remainder. If those two registers hold $u = (u_1 u_0 u_{-1} \cdots)$ and $w = (w_1 w_0 w_{-1} \cdots)$ for the sum and carry components of the partial remainder, each of them is in the range $[-2d, 2d)$.

Then the following technique is used to select the quotient digit.

```
$t = u_{[-2,1]} + w_{[-2,1]}$   Add most significant 4 bits

if ($t < -1/2$)
  $q_{-j} = -1$
else if($t >= 0$)
  $q_{-j} = +1$
else
  $q_{-j} = 0$
```

The value of t can be compared with the constants -1/2 and 0 using only three bits, t_1, t_0, and t_{-1}.

Consider the effect of this truncation:

$s^{(j-1)}$		$t_{[-1,1]}$	
Actual Value	Fraction	Truncated Value	Fraction
00.1111'	$1 - ulp$	00.1	1/2
00.1000'	1/2	00.1	1/2
00.0000'	0	00.0	0
11.1111'	$-ulp$	11.1	-1/2
11.0111'	$-1/2 - ulp$	11.0	-1
11.0000'	-1	11.0	-1

In the table, some extreme examples have been given in which the maximum loss of 1s or 0s occurs during truncation. We see that the effect of truncation is to reduce the value by subtracting at most 1/2.

$s^{(j-1)}$	$[-1,\ -1/2)$	$[-1/2,\ 0)$	$[0,\ 1/2)$	$[1/2,\ 1)$
$t_{[-1,1]}$	-1	-1/2	0	1/2

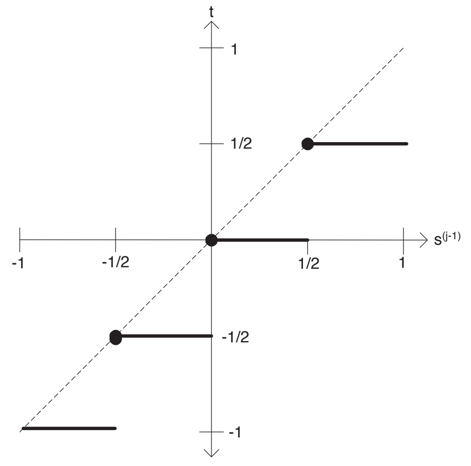

Figure 5: Effect of Truncation, $s^{(j-1)} \rightarrow t$

If $t < -1/2$, the true value of $s^{(j-1)}$ must be less than 0.

Similarly, if $t < 0$, then $s^{(j-1)} < 1/2$ and, since d is normalized, $1/2 \leq d$.

Truncation gives a value of t that is always less than or equal to $s^{(j-1)}$. Its value cannot be less than $s^{(j-1)}$ by 1/2 or more.

The three actual ranges of $s^{(j-1)}$ corresponding to the three ranges of $t_{[-1,1]}$ are as follows:

$t_{[-1,1]}$	$[-2d,\ -1/2)$	$[-1/2,\ 0)$	$[0,\ 2d)$
$s^{(j-1)}$	$[-2d,\ 0)$	$[-1/2,\ 1/2)$	$[0,\ 2d)$
q_{-j}	-1	0	1

Looking back at Figure 4 we see that these three ranges enable acceptable values of q_{-j} equal to -1, 0 and +1 respectively.

The design requires a 4-bit adder to generate t, together with logic to compare three of bits of t with constants -1/2 and 0. Alternatively the 8-bits to be added (4 from each of v and w) can be used as an address into a 256 entry ROM table that contains the corresponding values to q_{-j}. Only 2 bits would be needed for each entry.

As always, memory can be replaced by combinational logic. A 2-level AND-OR network PLA could be used to replace the ROM.

Here is a (long) example, $1627 \div 35$ in a $k = 6$ bit system.

The true sums are given in the comments for comparison with the values of t used in the arithmetic. In the example, “DN” means “do nothing”.

Integer Division

z = 00.011001011011, d = 00.100011

z	00.011001	011011	$s < d$, no normalization needed
d	00.100011		$d * 2^6 > 1/2$, no normalization needed
$-d$	11.011101		$-d * 2^6$
$s^{(0)}$	00.011001	011011	
$2s^{(0)}$	00.110010	11011	$>=$ 0 so set $q_{-1} = 1$
$+(-d)$	11.011101		subtract
$s^{(1)}_{sum}$	11.101111	11011	true_sum = 00.001111 11011
$s^{(1)}_{carry}$	00.100000		
$2s^{(1)}_{sum}$	11.011111	1011	2*true_sum = 00.011111 1011
$2s^{(1)}_{carry}$	01.000000		
			$t = 00.01$, $t_{[-1,1]} = 00.0$, set $q_{-2} = 1$
$+(-d)$	11.011101		subtract
$s^{(2)}_{sum}$	01.000010		true_sum = 11.111100 1011
$s^{(2)}_{carry}$	10.111010		
$2s^{(2)}_{sum}$	10.000101	011	2true_sum = 11.111001 011
$2s^{(2)}_{carry}$	01.110100		
			$t = 11.11$, $t_{[-1,1]} = 11.1 = -1/2$, $q_{-3} = 0$, DN
$s^{(3)}_{sum} = 2s^{(2)}_{sum}$	10.000101	011	
$s^{(3)}_{carry} = 2s^{(2)}_{carry}$	01.110100		
$2s^{(3)}_{sum}$	00.001010	11	2true_sum = 11.110010 11
$2s^{(3)}_{carry}$	11.101000		
$s^{(4)}_{sum} = 2s^{(3)}_{sum}$	00.001010	11	
$s^{(4)}_{carry} = 2s^{(3)}_{carry}$	11.101000		
			$t = 11.10$, $t_{[-1,1]} = 11.1 = -1/2$, $q_{-4} = 0$, DN
$2s^{(4)}_{sum}$	00.010101	1	2true_sum = 11.1001011
$2s^{(4)}_{carry}$	11.010000		
			$t = 11.10$, $t_{[-1,1]} = 11.1 = -1/2$, $q_{-5} = 0$, DN
$s^{(5)}_{sum} = 2s^{(4)}_{sum}$	00.010101	1	
$s^{(5)}_{carry} = 2s^{(4)}_{carry}$	11.010000		
$2s^{(5)}_{sum}$	00.101011		2true_sum = 11.001011
$2s^{(5)}_{carry}$	10.100000		
			$t = 11.00$, $t_{[-1,1]} = 11.0 = -1$, $q_{-6} = -1$, add
$+d$	00.100011		
$s^{(6)}_{sum}$	10.101000		
$s^{(6)}_{carry}$	01.000110		remainder is < 0 so add back and reduce q by 1
$+d$	00.100011		full 2's complement add is used in last step
	00.010001		

$q = (11000T) - 1 = 46$, remainder = 17

13.3 The p-d Plot

The p-d plot is used extensively to reason about quotient digit selection in high-radix dividers. It is a graph with y-axis equal to the shifted partial remainder and x-axis equal to the divisor value. See Figure 6. In the figure, $p = 2s^{(j-1)}$ is the shifted partial remainder. Its range is $[-2d, 2d)$. Values of d extend from 1/2 to 1 along the x-axis. The upper and lower sloping lines mark the upper and lower limits of the feasible region since $-2d \leq p < 2d$. The inner sloping lines show the limits of the region $-d \leq p < d$.

When d is almost 1.0, we see that, for the overlap region, the redundancy enables choices for the next quotient digit q_{-j}. For $-2d < p \leq -d$, $q_{-j} = -1$, for $-d \leq p < 0$, $q_{-j} \in [-1, 0]$, are valid choices, for $0 \leq p < d$, $q_{-j} \in [0, 1]$, are valid choices, and for $d \leq p < 2d$, $q_{-j} = 1$.

The two bold horizontal lines show the actual choice boundaries decided in the previous section. For the region between $-d$ and $+d$, $q_{-j} = 0$ is a valid choice.

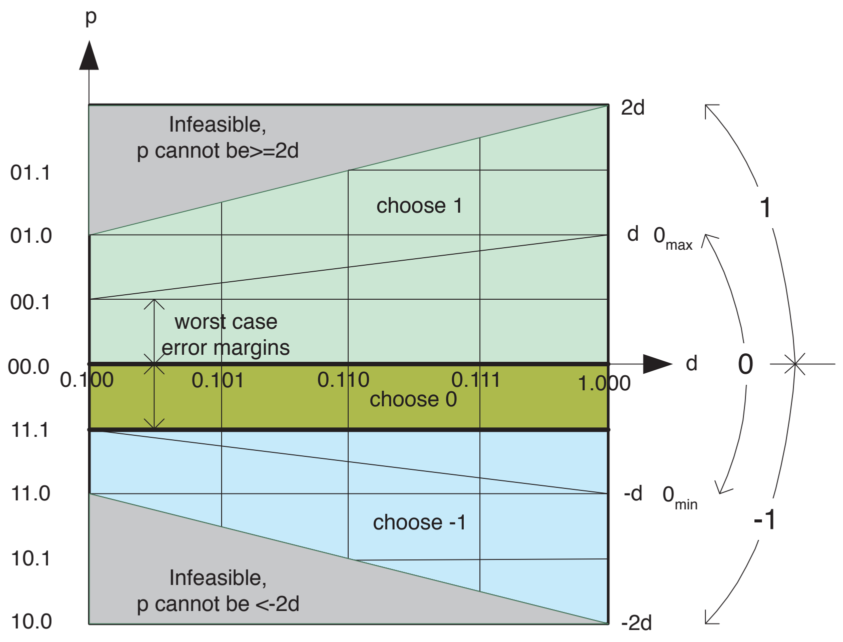

Figure 6: A p-d Plot corresponding to Figure 4

A maximum error of 1/2 is introduced by truncation. Since the decision point of $t = -1/2$, as depicted by the lower bold line, remains above the sloping line for $p = -d$, no error is made by selecting $q_{-j} = 0$ for any point within the region between the two bold lines. Similar arguments apply to the other two regions.

Note that the asymmetric nature of the boundary lines (at -1/2 and 0) reflects the asymmetry in the truncation process (towards $-\infty$).

In this example, horizontal lines were used for the boundaries between the regions of choice. Therefore the selection of the quotient digit is independent of the value of d. This is not always the case. In Radix-4 division the next quotient digit is selected by examining a few bits of both p and d.

13.4 Radix 4 Division

Figure 7 shows how the next quotient digit is selected, based on the old partial remainder left shifted 2 places, $4s^{(j-1)}$. Here, $q_{-j} \in [-3, 3]$.

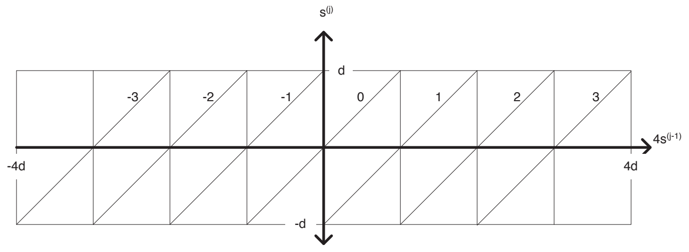

Figure 7: Quotient Digit Selection Graph for Radix-4 Division

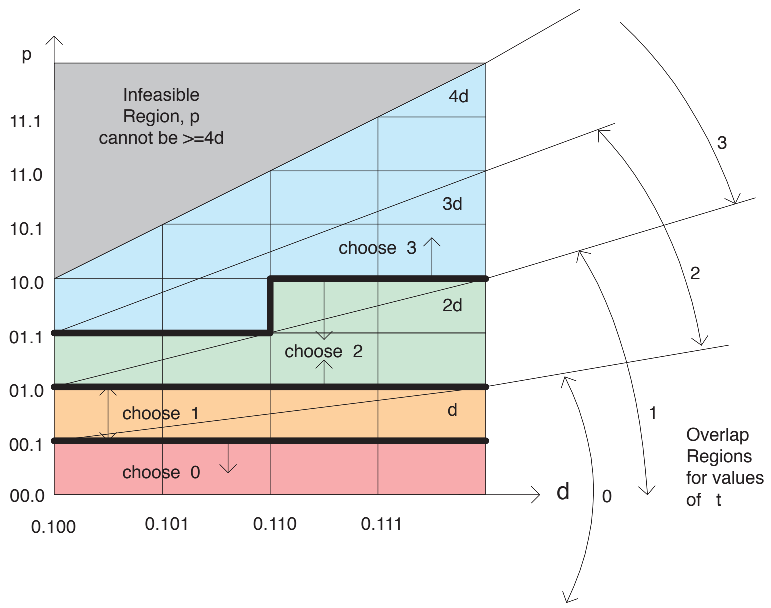

Figure 8: p-d plot for Radix-4 Division with $q_{-1} \in [-3, 3]$

For all but the end regions there are two possible choices for the next quotient digit. These diagrams are drawn with the assumption that there is no error in the value of t. The selection of the quotient digit depends on both p and d. Just one bit of d is needed, d_{-2}, to tell if d is in $[1/2, 3/4)$ or in $[3/4, 1)$. If an approximation to p is used then the boundary lines must be redrawn to accommodate the largest possible error in t. This design requires us to add $\pm 3d$ to the shifted partial remainder when the quotient digit is ± 3. One solution is to pre-compute $3d$ and keep it in an extra register. Another is to limit the quotient digits to $[-2, 2]$. This option is explored next.

The range of the partial remainder must be restricted so that we can safely choose $q_{-1} \in [-2, 2]$. Let $s^{(j-1)} \in [-hd, hd)$ for $h < 1$. Then $4s^{(j-1)} \in [-4hd, 4hd)$. In each iteration we will add 0, $\pm d$, or $\pm 2d$. In the worst case we add $\pm 2d$. When $4hd > 0$ we subtract, giving $4hd - 2d \leq hd$, or $h \leq 2/3$. We choose $h = 2/3$. An initial shift may be necessary so that $z \in [-2d/3, +2d/3)$, together with a corresponding adjustment to the final remainder.

The resulting quotient digit selection diagram is shown in Figure 9. Values of $q_{-j} = \pm 3$ are not used.

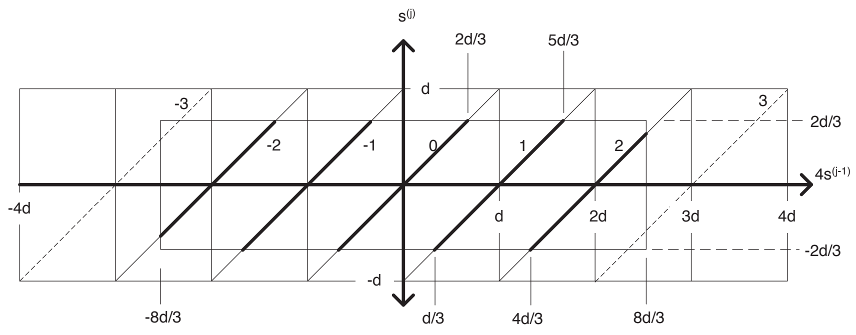

Figure 9: Quotient Digit Selection for Radix-4 Division where $q_{-j} \in [-2, 2]$

The resulting p-d plot is shown in Figure 10. At the right-hand end, when $d = 1$, we see the diagonal boundary lines pass through the points, $p = 8d/3,\ \ 5d/3,\ \ 4d/3,\ \ 2d/3,\ \ d/3$. These points come from the quotient digit selection diagram and mark the ends of the regions for each valid quotient digit. As before, the regions overlap, but the overlap regions are narrower than in Figure 8. The bold boundary lines become more stepped to accommodate these new regions. Note that these boundaries do not allow for any error in the values of p and d. Without further adjustment to allow for these errors, a carry-completion addition would be needed for calculating the new partial remainder on each iteration.

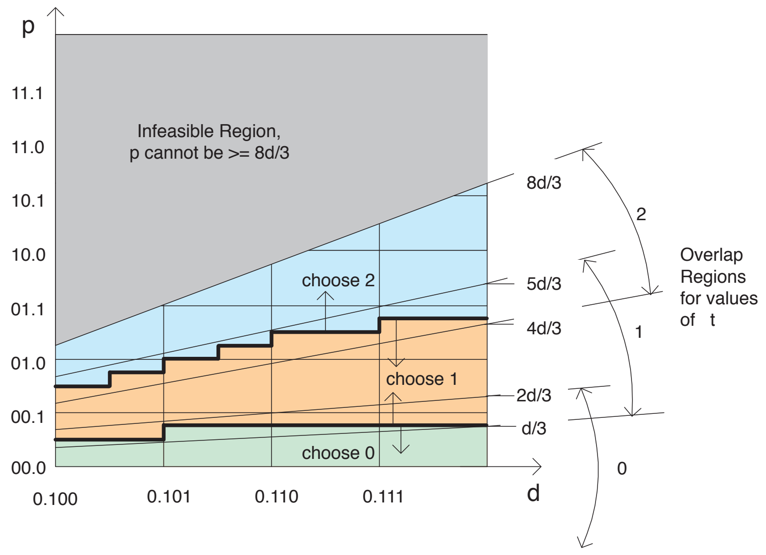

Figure 10: p-d plot for Radix-4 Division with $q_{-1} \in [-2, 2]$

13.4.1 Allowing for approximations in p and d

As mentioned above, the boundary lines on the p-d plot of Figure 10 assume that values of p and d are not truncated. This assumption is not practical since the next quotient digit will be generated by ROM table lookup and the size of the ROM would be prohibitive if all the bits in p and d were used as the address. Also, if truncation is not used, carry completion addition must take place on every iteration of the divide algorithm, which would greatly diminish its speed. For these reasons, both p and d are truncated.

The grid lines in Figure 10 represent the effects of truncation in which 4 bits of d, including its sign, and 4 bits of p are used. The bold lines show the boundaries between valid choices of the quotient digit. After truncation, the values of t_p and t_d are represented by the points at the intersections of these grid lines, only 18 points on this quadrant of the p-d plot are shown. If both p and d are allowed to be negative, there could be $2 \times 4 + 4 \times 14$ points. Since the bold lines do not lie on the grid lines, more bits of p and d are needed.

Truncation takes all of the thousands of p-d points within each of the rectangles formed by these grid lines and transforms them into the single point at the lower-left corner of that rectangle, as illustrated in Figure 11.

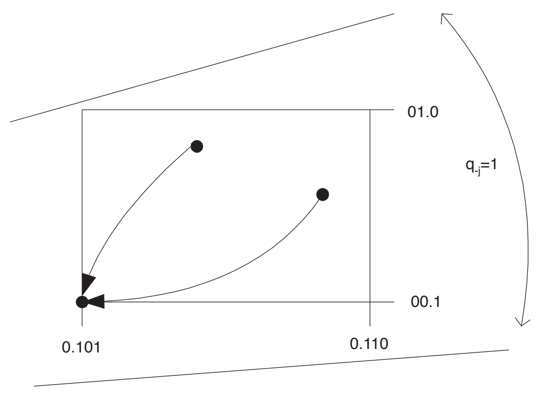

Figure 11: Effect of Truncation of Points Within One Rectangle of p-d Plot

One question remains. How do we determine the granularity of the truncation: how many bits are needed in the addition of the sum and carry components of p, and how many bits of d are needed?

The answer is that we have enough bits when each of the rectangles lies completely within one of the valid choice regions for the next quotient digit. All of the points within the rectangle depicted in Figure 11 are *covered* by the value of $q_{-j} = 1$.

Figure 12 shows that all but four of the rectangles are completely covered by one of the valid selection regions for q_{-j}. Triangles mark the lower lefthand corners of rectangles for which $q_{-j} = 0$ is a valid choice, while circles and squares mark the lower lefthand corners of rectangles for which $q_{-j} = 1$ and 2 respectively. It appears from the diagram more subdivision is necessary.

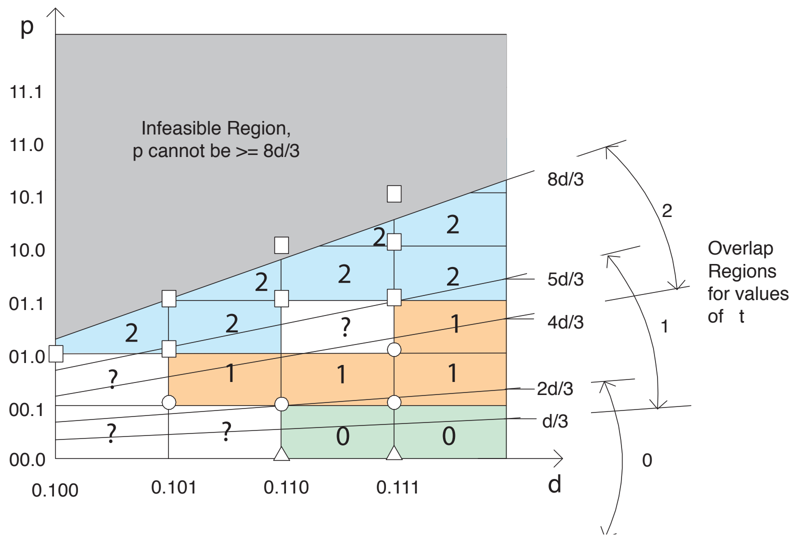

Figure 12: Selection of the quotient digit from truncated values of p and d

The effects of widening t_p by 1 bit are seen in Figure 13. One more subdivision of p and d is necessary to resolve the two remaining rectangles.

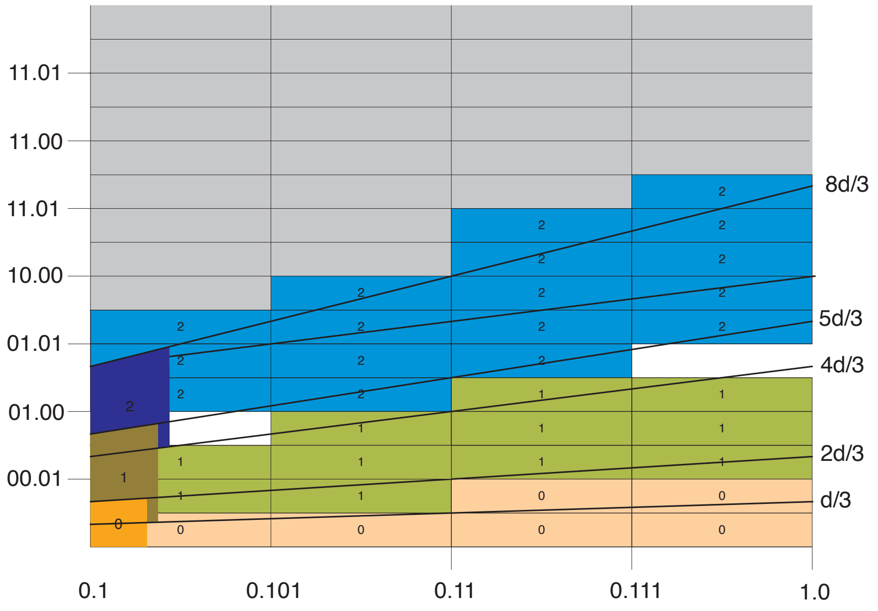

Figure 13: Selection of the quotient digit from truncated values of p and d

Finally, Figure 14 shows the completed p-d plot when there is enough resolution in the approximations to p and d. We will need 6 bits of p and 5 bits of d (including the sign bits).

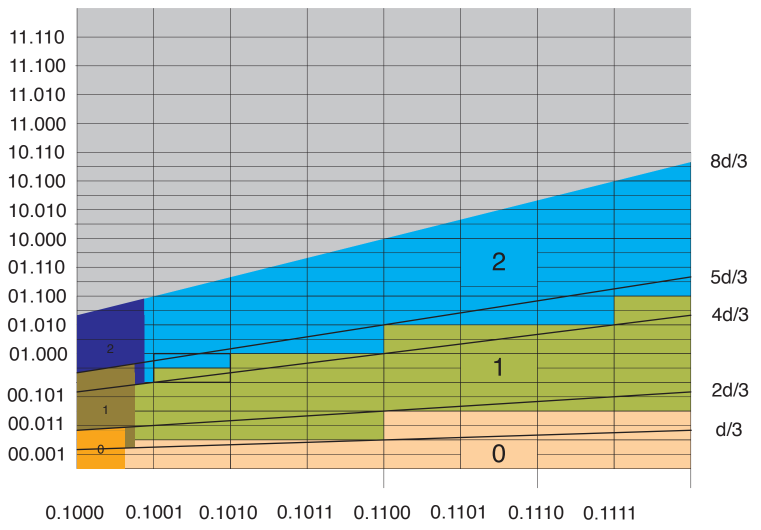

Figure 14: Selection of the quotient digit from truncated values of p and d

Next we see the contents of the ROM for the $+p$, $+d$ quadrant. Values of p and d are given in both decimal and binary notation. Only the part of the table covering the feasible region is given. The total number of ROM entries needed is $4 \times 32 \times 8 = 1024$.

		d							
		0.5	0.5625	0.625	0.6875	0.75	0.8125	0.875	0.9375
p		0.1000	0.1001	0.1010	0.1011	0.1100	0.1101	0.1110	0.1111
2.625	10.101								2
2.5	10.100								2
2.375	10.011							2	2
2.25	10.010						2	2	2
2.125	10.001					2	2	2	2
2	10.000					2	2	2	2
1.875	01.111				2	2	2	2	2
1.75	01.110			2	2	2	2	2	2
1.625	01.101			2	2	2	2	2	2
1.5	01.100		2	2	2	2	2	2	2
1.375	01.011	2	2	2	2	2	2	2	1
1.25	01.010	2	2	2	2	2	2	2	1
1.125	01.001	2	2	2	2	1	1	1	1
1	01.000	2	2	2	2	1	1	1	1
0.875	00.111	2	2	1	1	1	1	1	1
0.75	00.110	2	1	1	1	1	1	1	1
0.625	00.101	1	1	1	1	1	1	1	1
0.5	00.100	1	1	1	1	1	1	1	1
0.375	00.011	1	1	1	1	0	0	0	0
0.25	00.010	1	1	1	1	0	0	0	0
0.125	00.001	0	0	0	0	0	0	0	0
0.0	00.000	0	0	0	0	0	0	0	0

Here is an example using the p-d plot above.

Integer Division

$z = 00.01111111111_2 = 261888_{10}, d = 00.111110000_2 = 496_{10}$

$s^{(0)}$	00.011111111	110000000	$\in [-8d/3, 8d/3)$ no normalization necessary
$d2^9$	00.111110000		$\in [-1.0, 1.0)$ no normalization necessary
$4s^{(0)}$	01.111111111	000000000	$t_0 = 01111$ so $q_{-1} = 2$
$2d2^9$	01.111100000		subtract
$s^{(1)}$	00.000011111	000000000	
$4s^{(1)}$	00.001111100	000000000	$t_1 = 00011$ so $q_{-2} = 0$
$s^{(2)} = 4s^{(1)}$	00.111110000	000000000	$t_2 = 01111$ s0 $q_{-3} = 1$
$d2^9$	00.111110000		subtract
$s^{(3)}$	00.000000000	000000000	

The last 2 iterations yield $q_{-4} = q_{-5} = 0$. The quotient in redundant form is 20100. This translates to 528_{10}. The remainder is zero.