



## BIT-REPLICATION METHOD FOR UP- MULTIPLYING

*Robert A. Ulichney Shiufun Cheung*

Cambridge  
Research  
Laboratory

**Cambridge Research Laboratory**

**Technical Report Series**

**CRL 97/9**  
January 97

---

## Cambridge Research Laboratory

The Cambridge Research Laboratory was founded in 1987 to advance the state of the art in both core computing and human-computer interaction, and to use the knowledge so gained to support the Company's corporate objectives. We believe this is best accomplished through interconnected pursuits in technology creation, advanced systems engineering, and business development. We are actively investigating scalable computing; mobile computing; vision-based human and scene sensing; speech interaction; computer-animated synthetic persona; intelligent information appliances; and the capture, coding, storage, indexing, retrieval, decoding, and rendering of multimedia data. We recognize and embrace a technology creation model which is characterized by three major phases:

**Freedom:** The lifeblood of the Laboratory comes from the observations and imaginations of our research staff. It is here that challenging research problems are uncovered (through discussions with customers, through interactions with others in the Corporation, through other professional interactions, through reading, and the like) or that new ideas are born. For any such problem or idea, this phase culminates in the nucleation of a project team around a well-articulated central research question and the outlining of a research plan.

**Focus:** Once a team is formed, we aggressively pursue the creation of new technology based on the plan. This may involve direct collaboration with other technical professionals inside and outside the Corporation. This phase culminates in the demonstrable creation of new technology which may take any of a number of forms—a journal article, a technical talk, a working prototype, a patent application, or some combination of these. The research team is typically augmented with other resident professionals—engineering and business development—who work as integral members of the core team to prepare preliminary plans for how best to leverage this new knowledge, either through internal transfer of technology or through other means.

**Follow-through:** We actively pursue taking the best technologies to the marketplace. For those opportunities which are not immediately transferred internally and where the team has identified a significant opportunity, the business development and engineering staff will lead early-stage commercial development, often in conjunction with members of the research staff. While the value to the Corporation of taking these new ideas to the market is clear, it also has a significant positive impact on our future research work by providing the means to understand intimately the problems and opportunities in the market and to more fully exercise our ideas and concepts in real-world settings.

Throughout this process, communicating our understanding is a critical part of what we do, and participating in the larger technical community—through the publication of refereed journal articles and the presentation of our ideas at conferences—is essential. Our technical report series supports and facilitates broad and early dissemination of our work. We welcome your feedback on its effectiveness.

Robert A. Iannucci, Ph.D.  
Director

# **Bit-Replication Method for Up-Multiplying**

Robert A. Ulichney & Shiufun Cheung

January 1997

## **Abstract**

This technical report describes a simple and efficient method for expanding  $q$ -bit integer values to  $m$ -bit integer values ( $m > q$ ) by means of bit replication. First, we show that the optimal number of repetitions is given by  $\text{ceiling}\{m/q\}$  and that the method is equivalent to multiplication by the ideal gain when  $m/q$  is an integer. We then demonstrate that, in the case where  $m/q$  is not an integer, truncating the fraction bits to the right of the decimal point will lead to zero average error. The report also includes two suggestions for implementing the bit-replication process, both of which have a vast complexity advantage over a multiplier. Two examples are given at the end to illustrate the bit-replication process in action.

**© Digital Equipment Corporation 97**

This work may not be copied or reproduced in whole or in part for any commercial purpose. Permission to copy in whole or in part without payment of fee is granted for nonprofit educational and research purposes provided that all such whole or partial copies include the following: a notice that such copying is by permission of the Cambridge Research Laboratory of Digital Equipment Corporation in Cambridge, Massachusetts; an acknowledgment of the authors and individual contributors to the work; and all applicable portions of the copyright notice. Copying, reproducing, or republishing for any other purpose shall require a license with payment of fee to the Cambridge Research Laboratory. All rights reserved.

CRL Technical reports are available on the CRL's web page at  
<http://www.crl.research.digital.com>.

Digital Equipment Corporation  
Cambridge Research Laboratory  
One Kendall Square, Building 700  
Cambridge, Massachusetts 02139

## Introduction

There are several applications where the set of values within a particular range needs to be operated on at a larger, or higher-precision, range. In particular, we will consider the case where a  $q$ -bit integer input level  $L_i$  with values from the range

$$L_i \in \{ 0, 1, 2, \dots, 2^q - 1 \}$$

is to be scaled to a  $m$ -bit integer output level  $L_o$  with values from the range

$$L_o \in \{ 0, \dots, 2^m - 1 \}.$$

This can be achieved by performing a multiplication by an *ideal gain*,

$$G = \frac{2^m - 1}{2^q - 1}$$

followed by rounding to the nearest integer:

$$L_o = \text{Round}\{G \times L_i\}.$$

In this paper we will show that this ideal method can be closely approximated with a much simpler implementation of bit replication. We will also determine the optimal number of bits to replicate for minimum error.

## Series Expansion

For any real numbers  $a$  and  $b$ , where  $b$  is not equal to 0 or 1, the follow expression is valid:

$$\frac{a-1}{b-1} = \frac{a}{b} + \frac{(a/b)-1}{b-1}.$$

This expression can itself be used to expand the rightmost term further:

$$\frac{(a/b)-1}{b-1} = \frac{a}{b^2} + \frac{(a/b^2)-1}{b-1}.$$

Recursively repeating this expansion  $N$  times results in the series:

$$\frac{a-1}{b-1} = \sum_{i=1}^N \frac{a}{b^i} + \left( \frac{(a/b^N)-1}{b-1} \right).$$

In our case, using  $a = 2^m$  and  $b = 2^q$ , the ideal gain can be expressed as

$$G = \sum_{i=1}^N 2^{m-iq} + \left( \frac{2^{m-Nq} - 1}{2^q - 1} \right).$$

From this, we define the approximate gain to be

$$\hat{G} = \sum_{i=1}^N 2^{m-iq},$$

and the gain-error to be

$$E = \frac{1 - 2^{m-Nq}}{2^q - 1}.$$

For the purpose of analysis, let us assume that the output level  $L_o$  can be a real number (instead of an integer) and the rounding operation is unnecessary. The data scaling will therefore be implemented as follows:

$$L_o = \hat{G} \times L_i.$$

Under these assumptions, the gain-error  $E$  is proportional to the error in the output level.

## Bit Replication

We can show that multiplication by the approximate gain  $\hat{G}$  can be implemented simply as a replication of the input level bits. Since each term in  $\hat{G}$  is a power of 2, multiplication is equivalent to a shift of the input. Note also that the shifts corresponding to successive terms differ by  $q$  spaces, which is the same as the number of bits in the input. This means that there is no overlap of bits and no addition is necessary. Consider the example of converting a 3-bit input ( $q = 3$ ) to an 8-bit output ( $m = 8$ ). In Figure 1 the values of the 3 input bits are “A”, “B”, and “C” as shown. If the sum was carried out to 5 terms or repeat periods ( $N = 5$ ), the output value would have the bit values as indicated. This bit replication extends into the fraction bits to the right of the decimal point.

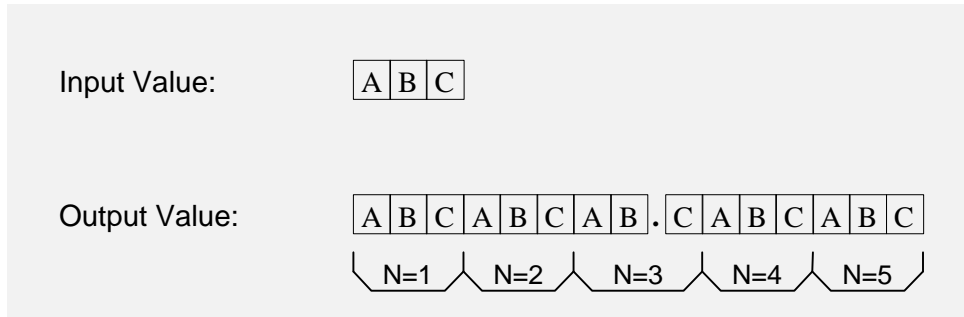


Figure 1. Example of scaling a 3-bit input value to 8 bits using the approximate gain  $\hat{G}$  with  $N = 5$ .

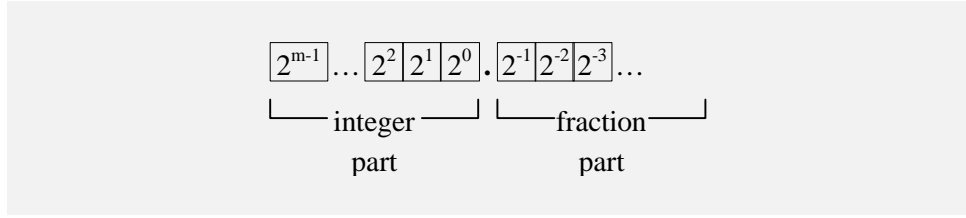


Figure 2. Representation of a real number with binary digits.

The weighting of the bit values in this representation is shown in Figure 2. Multiplying an input by the approximate gain will result in a real number with an integer part and a fraction part. Although the goal is to generate a  $m$ -bit integer output value, it may be useful to keep additional fraction bits for intermediate computations of the output value for higher accuracy.

In the next sections two aspects of the bit-replication process will be optimized in terms of minimizing the error. First, the optimal value of  $N$  is determined. Then for cases where the last repeat periods have both integer and fraction bits (as in the third period of Figure 1) the contribution of the fraction bits to the accuracy of the output level is found.

## Optimization of Repeat Periods

The optimal number of repeat periods  $N$  can be determined simply by solving for the value of  $N$  that minimizes  $|E|$ .  $|E|$  is zero exactly once when  $m - Nq = 0$ , so the optimal value should be

$$N = m/q.$$

The problem is that  $N$  can only take on integer values so this result will only occur when  $q$  is an exact multiple of  $m$ . However, if this condition does hold, the error is identically zero, and bit replication yields a perfectly scaled output level.

For the case where  $(m/q)$  is not an integer, the solution for the optimum  $N$  is more involved. In Figure 3  $|E|$  is plotted as a function of  $N$ . The graph reveals two candidate integer values for  $N$  on either side of the point  $(m/q)$ :  $N_{LO}$  is the nearest integer less than  $(m/q)$  and  $N_{HI}$  is the nearest integer greater than  $(m/q)$ . Since the denominator of  $|E|$  is constant with respect to  $N$ , we can consider the equivalent problem of minimizing the function

$$b = \left| 2^{m-Nq} - 1 \right|.$$

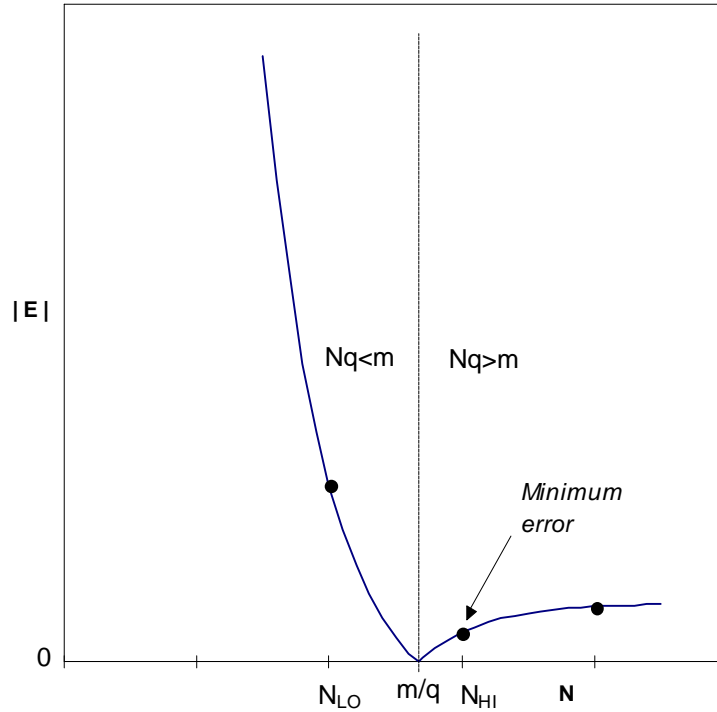


Figure 3. Absolute error as a function of  $N$ .

For  $N < m/q$ ,  $b$  increases monotonically with decreasing  $N$  because the exponent  $(m - Nq)$  becomes increasingly positive. This exponent is a positive integer, so the *smallest*  $b$  can be is 1.

For  $N > m/q$ ,  $b$  increases monotonically with increasing  $N$  because the exponent  $(m - Nq)$  becomes increasingly negative. ( $|E|$  asymptotically approaches  $1/(2^q - 1)$ .) The exponent  $(m - Nq)$  is a negative integer, so  $b < 1$ .

Therefore, of the two candidates,  $N_{LO}$  and  $N_{HI}$ , the value of  $N$  that minimizes  $|E|$  is  $N_{HI}$ . This is the nearest integer larger than the non-integer value  $(m/q)$ , or equivalently,

$$N = \text{ceiling}\{m/q\}.$$

## Contribution of Fraction Bits

Consider the case of  $q = 5$  and  $m = 8$ . The optimal number of repeat periods is

$$N = \text{ceiling}\{8/5\} = 2.$$

For a 5-bit input, the output is as shown in Figure 4. There are 2 fraction bits for this particular example. In all cases, the number of fraction bits due to repeating bits by the optimum  $N$  periods is



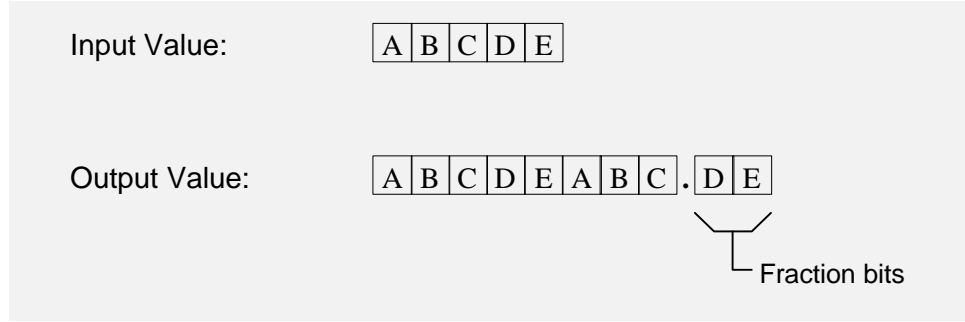


Figure 4. Scaling a 5-bit number to an 8-bit number with  $N = 2$  repeat periods.

$$b = Nq - m.$$

Note that there are no fraction bits for the error-free case where  $(m/q)$  is an integer.

In the following, we derive the average contribution of the fraction bits over all possible input levels. Since the fraction bits are simply the  $b$  least significant bits of each input level  $L_i$ , they will take on the values  $\{0, 1, 2, \dots, 2^b - 1\}$  in a periodic fashion as  $L_i$  increases. Note that there will be  $2^{q-b}$  complete periods in the range of  $L_i$ . The average contribution of the fraction bits over all  $L_i$  is therefore the same as the average contribution over one single period.

Let the value of the fraction bits be  $i$  and the contribution of the fraction bits to the output be  $F$ . We can then write

$$F = \frac{i}{2^b}$$

and the average contribution of the fraction bits to the output is therefore given by

$$\bar{F} = \frac{1}{2^b} \sum_{i=0}^{2^b-1} \frac{i}{2^b}.$$

Using the fact that

$$\sum_{i=0}^{M-1} i = M \left( \frac{M-1}{2} \right)$$

the above expression simplifies to

$$\bar{F} = \frac{1 - 2^{-b}}{2}.$$

Next, we derive the average output error due to the gain-error over all possible input levels. Using the change of variables  $b = Nq - m$ , we can rewrite the gain-error to be

$$E = \frac{1 - 2^{-b}}{2^q - 1}.$$

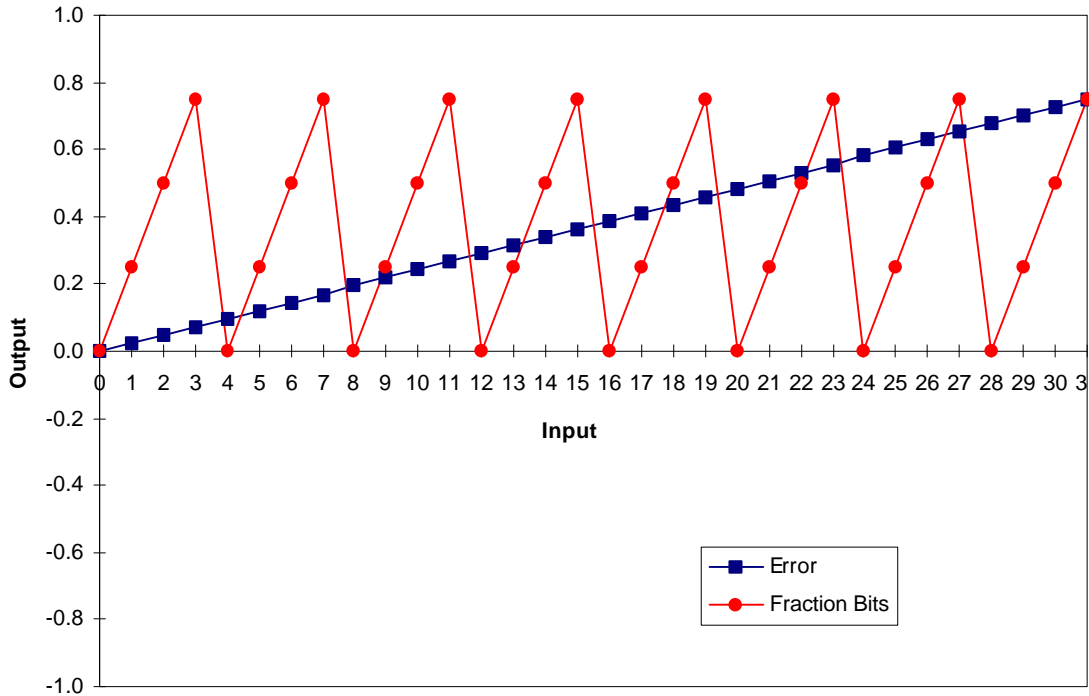


Figure 5. Output error as a function of input, and the contribution of fraction bits to the output value.

For each input level  $L_i$ , the output error  $e$  is  $E \times L_i$ . The *average output error* is then given by

$$\bar{e} = \frac{1}{2^q} \sum_{L_i=0}^{2^q-1} L_i \left( \frac{1-2^{-b}}{2^q-1} \right) = \frac{1-2^{-b}}{2}.$$

For our example of  $q=5$  and  $m=8$ , the contribution of the fraction bits  $F$  and output error  $e$  are plotted in Figure 5.

It is very interesting to note that for all cases,

$$\bar{e} = \bar{F}.$$

By this we can conclude that zero average error can be achieved by always truncating the fraction bits! Using our example of  $q=5$  and  $m=8$ .again, the output error is plotted in Figure 6 for the cases of no fraction bits used, one fraction bit used, and both fraction bits used. The average errors are 0.000, 0.250, and 0.375 ( $= (1-2^{-2})/2$ ) respectively.

## Implementation

We have proved that multiplication of a  $q$ -bit input level  $L_i$  by the ideal gain

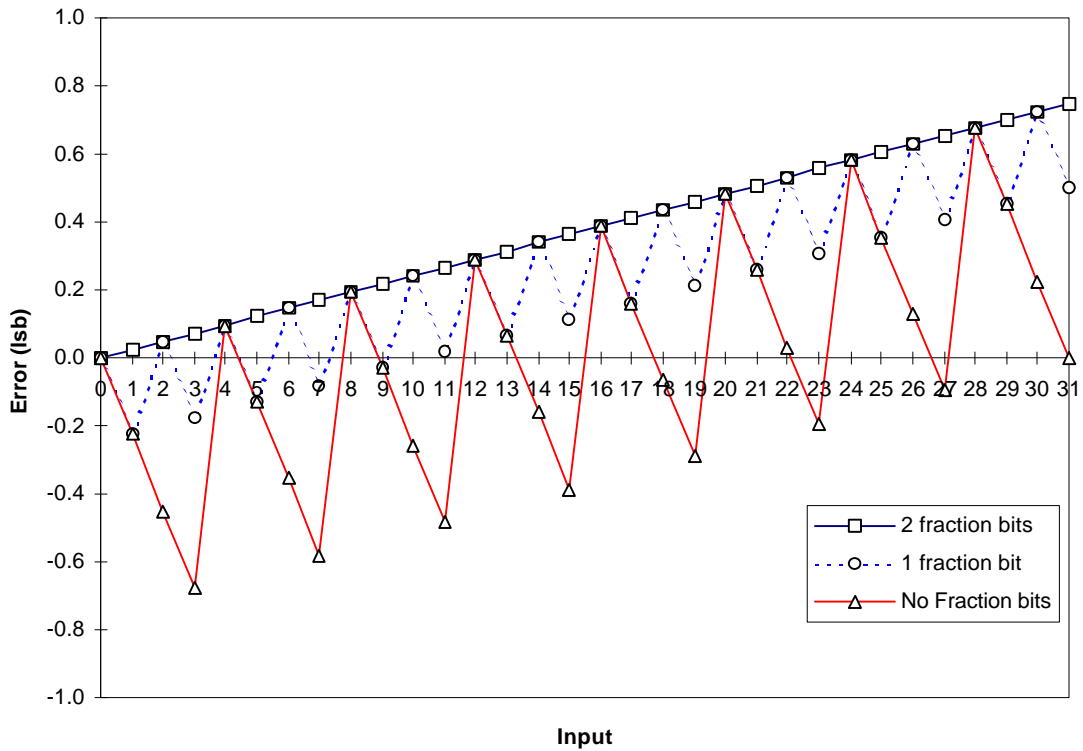


Figure 6. Output error as a function of number of fraction bits retained.

$$G = \frac{2^m - 1}{2^q - 1}$$

can be achieved with *zero average error* by simply replicating the input bits from the most significant part of the  $m$ -bit output register to the least significant part. Using additional fraction bits for intermediate computation can only increase average error.

The most straightforward hardware implementation of this bit replication method is the wired

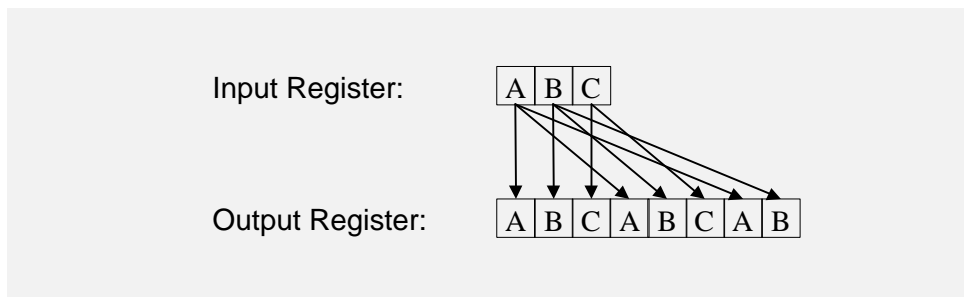


Figure 7. Wired implementation.

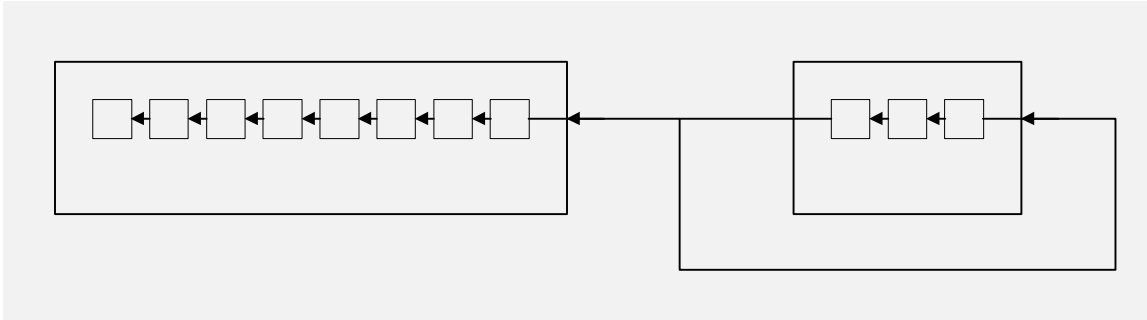


Figure 8. Shift-register implementation

approach as illustrated in Figure 7. In this figure we use the case of  $q = 3$  and  $m = 8$  as an example. “A”, “B”, and “C” represent input bit values.

Another implementation can use shift registers as depicted in Figure 8. The input register performs a circular shift left into an output shift register as shown. The operation is complete after  $m$  shifts.

Either case has a tremendous complexity advantage over a multiplier.

## Examples

We will end this document by using two examples to illustrate the bit-replication process in action. The first example is the case of  $q = 5$  and  $m = 8$ . In Table 1, we show, for each possible input level, the actual output level and its difference from the ideal output level. Note that the average error over all possible input levels is zero. Table 2 shows the case of  $q = 4$  and  $m = 12$ . In this case,  $m/q$  is an integer, so the bit-replication gives the exact answer and the error is zero for all input levels.

Input $L_i$		Output $L_o$		Ideal	Error
0000	( 0 )	0000000	( 0 )	0.00	0.00
0001	( 1 )	00001000	( 8 )	8.23	-0.23
00010	( 2 )	00010000	( 16 )	16.45	-0.45
00011	( 3 )	00011000	( 24 )	24.68	-0.68
00100	( 4 )	00100001	( 33 )	32.90	0.10
00101	( 5 )	00101001	( 41 )	41.13	-0.13
00110	( 6 )	00110001	( 49 )	49.35	-0.35
00111	( 7 )	00111001	( 57 )	57.58	-0.58
01000	( 8 )	01000010	( 66 )	65.81	0.19
01001	( 9 )	01001010	( 74 )	74.03	-0.03
01010	( 10 )	01010010	( 82 )	82.26	-0.26
01011	( 11 )	01011010	( 90 )	90.48	-0.48
01100	( 12 )	01100011	( 99 )	98.71	0.29
01101	( 13 )	01101011	( 107 )	106.94	0.06
01110	( 14 )	01110011	( 115 )	115.16	-0.16
01111	( 15 )	01111011	( 123 )	123.39	-0.39
10000	( 16 )	10000100	( 132 )	131.61	0.39
10001	( 17 )	10001100	( 140 )	139.84	0.16
10010	( 18 )	10010100	( 148 )	148.06	-0.06
10011	( 19 )	10011100	( 156 )	156.29	-0.29
10100	( 20 )	10100101	( 165 )	164.52	0.48
10101	( 21 )	10101101	( 173 )	172.74	0.26
10110	( 22 )	10110101	( 181 )	180.97	0.03
10111	( 23 )	10111101	( 189 )	189.19	-0.19
11000	( 24 )	11000110	( 198 )	197.42	0.58
11001	( 25 )	11001110	( 206 )	205.65	0.35
11010	( 26 )	11010110	( 214 )	213.87	0.13
11011	( 27 )	11011110	( 222 )	222.10	-0.10
11100	( 28 )	11100111	( 231 )	230.32	0.68
11101	( 29 )	11101111	( 239 )	238.55	0.45
11110	( 30 )	11110111	( 247 )	246.77	0.23
11111	( 31 )	11111111	( 255 )	255.00	0.00
<b>average error</b>					<b>0.00</b>

Table 1: The Input Levels, the Output Levels, and the Errors for the Case of  $q = 5$  and  $m = 8$

<b>Input <math>L_i</math></b>		<b>Output <math>L_o</math></b>		<b>Ideal</b>	<b>Error</b>
0000	( 0 )	000000000000	( 0 )	0.00	0.00
0001	( 1 )	000100010001	( 273 )	273.00	0.00
0010	( 2 )	001000100010	( 546 )	546.00	0.00
0011	( 3 )	001100110011	( 819 )	819.00	0.00
0100	( 4 )	010001000100	( 1092 )	1092.00	0.00
0101	( 5 )	010101010101	( 1365 )	1365.00	0.00
0110	( 6 )	011001100110	( 1638 )	1638.00	0.00
0111	( 7 )	011101110111	( 1911 )	1911.00	0.00
1000	( 8 )	100010001000	( 2184 )	2184.00	0.00
1001	( 9 )	100110011001	( 2457 )	2457.00	0.00
1010	( 10 )	101010101010	( 2730 )	2730.00	0.00
1011	( 11 )	101110111011	( 3003 )	3003.00	0.00
1100	( 12 )	110011001100	( 3276 )	3276.00	0.00
1101	( 13 )	110111011101	( 3549 )	3549.00	0.00
1110	( 14 )	111011101110	( 3822 )	3822.00	0.00
1111	( 15 )	111111111111	( 4095 )	4095.00	0.00
<b>average error</b>					<b>0.00</b>

Table 2: The Input Levels, the Output Levels, and the Errors for the Case of  $q = 4$  and  $m = 12$





™

## **Bit-Replication Method for Up-Multiplying**

*Robert A. Uichney Shiufun Cheung*

**CRL 97/9**  
January 97