Computers Math. Applic. Vol. 24, No. 3, pp. 3–12, 1992
Printed in Great Britain.

0097-4943/92 $5.00 + 0.00

ADAPTIVE m-ARY SEGMENTATION AND CANONICAL RECODING ALGORITHMS FOR MULTIPLICATION OF LARGE BINARY NUMBERS

ÇETIN K. KOÇ AND CHING-YU HUNG
Department of Electrical Engineering
University of Houston, Houston, TX 77204, U.S.A.

(Received June 1990)

Abstract—We propose a variable-length segmentation strategy which significantly reduces the average number of additions required by the m-ary segmentation and the canonical recoding algorithms for multiplication of large binary numbers. This strategy produces two new algorithms: the adaptive m-ary segmentation algorithm utilizes both the speedup inherent in high-radix multiplication and the ability to skip zero bits; the adaptive m-ary segmentation canonical recoding algorithm gains additional benefit from the increased probability of zero after the canonical recoding. The average number of additions required is computed using Markov chains.

1. INTRODUCTION

The binary add-and-shift algorithm requires at most n additions to compute $P = AB$, where multiplicand A and multiplier B are n-bit binary numbers, and an addition operation involves two (up to) $2n$-bit numbers [1,2]. The *sequential* add-and-shift algorithm computes the product by scanning the bits of multiplier B one or more than one bits at a time, and adding some multiple of multiplicand A to the partial product. The computation time of the multiplication operation is proportional to the total number of additions required. However, the partial products may also be generated and summed in *parallel*. The total execution time of the parallel multiplication can be decreased substantially by using the partial product matrix reduction or column compression techniques [3,4]. The parallel execution time and the amount of memory space required for partial product generation are still a function of the total number of partial products.

We are interested in obtaining recoding and segmentation methods to reduce the *average* number of partial products. Assuming that the multiplication algorithm computing the final product is the sequential add-and-shift algorithm, we concentrate on the reduction of the average number of additions required. There are two well-known methods to reduce the total number of additions: m-ary segmentation and canonical recoding. This paper contains the following results:

- In Section 2 we count the average number of additions required by the m-ary segmentation algorithm and show that the total number of additions can be minimized by proper selection of m.
- In Section 3 we introduce a Markov chain model in order to compute the average number of additions required by the canonical recoding multiplication algorithm.
- In Section 4 we propose a novel adaptive (data-dependent and variable-length) segmentation strategy to further reduce the average number of additions required by the m-ary multiplication algorithm.
- In Section 5 we apply the new segmentation strategy to the canonically recoded signed-digit number, obtaining the adaptive m-ary segmentation canonical recoding algorithm.
- Finally, in Section 6, these algorithms are compared in terms of the average number of additions required.

2. m-ARY SEGMENTATION

The m-ary segmentation (radix m) multiplication algorithm utilizes segmentation and preprocessing to reduce the number of additions. We restrict our attention to the case where m is a power of 2. Computations are performed by grouping bits together: radix 8 (octal) and radix 16 (hexadecimal) cases are well-known and often used. The following procedure describes the computation of the product using this algorithm.

The m-ary Segmentation Multiplication Algorithm

1. Compute and store wA for all $2 \leq w \leq m-1$.
2. Decompose B into d-bit words $B_j^* = (B_{jd+d-1}\, B_{jd+d-2} \ldots B_{jd+1}\, B_{jd})$ for $0 \leq j \leq k-1$ where $d = \log_2 m$ and $k = \frac{n}{d}$.
3. Set $P := 0$.
4. Repeat step 4a, for $j = k-1, k-2, \ldots, 1, 0$.
   4a. Compute $P := m\,P + B_j^* A$.
5. End.

For example, when $n = 12$, $m = 8$, and $d = 3$, then

$$B = (011100101001),$$
$$B^* = (011), (100), (101), (001).$$

The above procedure computes P as

$$P := (011)\,A = 3\,A,$$
$$P := m\,P + (100)\,A = 8\,P + 4\,A$$
$$P := m\,P + (101)\,A = 8\,P + 5\,A$$
$$P := m\,P + (001)\,A = 8\,P + A.$$

Since wA does not need to be computed for $w = 0$ and $w = 1$, the m-ary segmentation algorithm requires a total of $m-2$ additions in the preprocessing stage. However, we reduce the number of additions required during the second (add-and-shift) stage by a factor of $\log_2 m$. Since the probability that $w = 0$ is $\frac{m-1}{m}$, the average number of additions required by the m-ary segmentation algorithm is found as

$$T = m - 2 + \frac{n}{d}\left(1 - \frac{1}{m}\right), \tag{1}$$

where $d = \log_2 m$. Given n, there exists an optimal m such that the total number of additions is minimum. In Table 1 we tabulate the optimal $d = \log_2 m$, for $n = 2^6, 2^7, 2^8, \ldots, 2^{16}$.

Table 1. Optimal values of $d = \log_2 m$.

$n \rightarrow$	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}
m-ary	3	3	4	5	5	6	7	8	8	9	10
Adaptive m-ary	3	4	5	5	6	7	7	8	9	10	10
Adaptive m-ary canonical	3	3	4	5	5	6	7	8	8	9	10

3. CANONICAL RECODING

In computing $P = AB$, we may skip additions whenever the corresponding bit of the multiplier is zero. Since the average number of zero bits is $\frac{n}{2}$ for an n-bit binary number, the binary multiplication algorithm requires $\frac{n}{2}$ addition operations on the average. Recoding techniques (Booth recoding, bit-pair recoding, etc.) for sparse representations of binary numbers have been effectively used [1,2]. For example, the original Booth recoding technique [5] scans the bits of

the multiplier one bit at a time, and adds or subtracts the multiplicand to, or from, the partial product, depending on the value of the current bit and the previous bit. The modified versions of the Booth algorithm scan the bits of the multiplier two bits at a time [6] or three bits at a time [2]. These techniques are equivalent in the sense that a signed-digit representation which is based on the identity $2^k - 1 = 2^{k-1} + \cdots + 2^1 + 2^0$ is used to collapse blocks of ones appearing in a binary representation. Thus, in a signed-digit number with radix 2, three symbols $\{0, 1, \bar{1}\}$ are allowed for the digit set, in which $\bar{1}$ represents -1. A minimal signed-digit number $D = (D_{n-1}\, D_{n-2} \ldots D_1\, D_0)$ that contains no adjacent nonzero digits (i.e., $D_i \cdot D_{i-1} = 0$; $0 < i \leq n-1$) is called a *canonical signed-digit number* [1,7]. The canonical recoding algorithm uses the truth table given in Table 2 to recode a two's complement binary number $B = (B_{n-1}\, B_{n-2} \ldots B_1\, B_0)$ as $D = (D_{n-1}\, D_{n-2} \ldots D_1\, D_0)$, where $D_i \in \{0, 1, \bar{1}\}$.

The Canonical Recoding Algorithm

1. Extend the sign bit of B, i.e., set $B_n = B_{n-1}$.
2. Set $C_0 = 0$.
3. Repeat step 3a, for $i = 0, 1, 2, \ldots, n-1$.
   3a. Use the truth table in Table 2 to compute D_i and C_{i+1}.
4. End.

Once the canonical signed-digit representation of B is computed, we can use the add-and-shift algorithm to compute the product. At the ith step the algorithm adds A or $-A$ to the partial product if $D_i = 1$ or $D_i = \bar{1}$, respectively. If $D_i = 0$ we skip an addition. The average number of additions required by the canonical recoding multiplication algorithm is equal to the average number of nonzero digits in the canonically recoded signed-digit number. In the following, we introduce a Markov chain model for the computation of the average number of nonzero digits.

An n-bit binary number B uniformly distributed in the range $[0, 2^n - 1]$ can be viewed as a random process that generates one bit at a time. Each bit assumes a value of zero or one with equal probability and there is no dependency between any two bits. Thus, we have $\mathcal{P}(B_i = 0) = \mathcal{P}(B_i = 1) = \frac{1}{2}$, for $0 \leq i \leq n-1$. The signed-digit numbers produced by the canonical recoding algorithm can be modeled using a finite Markov chain. The state variables of the Markov chain are defined as the vectors (B_{i+1}, B_i, C_i). There are 8 states for the 8 possible combinations of input, as shown in Table 2. Let $(B_{i+1}, B_i, C_i) = (0, 0, 0)$ represent state s_0. We compute the output (D_i, C_{i+1}) as $(0, 0)$. Thus, the next state is $(B_{i+2}, B_{i+1}, C_{i+1}) = (B_{i+2}, 0, 0)$. Since $\mathcal{P}(B_{i+2} = 0) = \mathcal{P}(B_{i+2} = 1) = \frac{1}{2}$, the next states are $s_0 = (0, 0, 0)$ and $s_4 = (1, 0, 0)$, having equal probability. The state table given in Table 3 is produced by considering all 8 states and their successors. Let $\mathcal{P}_{ij}$ denote the probability that the successor state of s_i is s_j. We find $\mathcal{P}00 = \mathcal{P}04 = \frac{1}{2}$ and $\mathcal{P}0j = 0$, for $j = 1, 2, 3, 5, 6, 7$, from Table 3. The one-step transition probability matrix is given as

$$\mathcal{P} = \begin{bmatrix} 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 \end{bmatrix}. \tag{2}$$

Let π_i be the limiting probability of state s_i. The limiting probability for each state is found by solving the following system of linear equations [8,9]

$$\pi\mathcal{P} = \pi,$$

$$\sum_{i=0}^{7} \pi_i = 1,$$

Table 2. Canonical recoding.

Multiplier Bits		Assumed Carry-in	Recoded Digit	Assumed Carry-out
B_{i+1}	B_i	C_i	D_i	C_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	0	1

Table 3. State table for the canonical recoding algorithm.

State		Output	Next State	
s_i	(B_{i+1}, B_i, C_i)	(D_i, C_{i+1})	$B_{i+2} = 0$	$B_{i+2} = 1$
s_0	(0,0,0)	(0,0)	s_0	s_4
s_1	(0,0,1)	(1,0)	s_0	s_4
s_2	(0,1,0)	(1,0)	s_0	s_4
s_3	(0,1,1)	(0,1)	s_1	s_5
s_4	(1,0,0)	(0,0)	s_2	s_6
s_5	(1,0,1)	$(\bar{1},1)$	s_3	s_7
s_6	(1,1,0)	$(\bar{1},1)$	s_3	s_7
s_7	(1,1,1)	(0,1)	s_3	s_7

which gives

$$\pi = \left[\frac{1}{6}, \frac{1}{12}, \frac{1}{12}, \frac{1}{6}, \frac{1}{6}, \frac{1}{12}, \frac{1}{12}, \frac{1}{6}\right].$$

Having computed π_i and $\mathcal{P}_{ij}$ for all $0 \leq i, j \leq 7$, we can easily prove several interesting and important properties of the canonical recoding algorithm.

- The probability that a digit in a signed-digit number D is equal to zero is found by summing the limiting probabilities of the states for which output $D_i = 0$. From Table 3, we compute this as

$$\mathcal{P}(D_i = 0) = \pi_0 + \pi_3 + \pi_4 + \pi_7 = \frac{2}{3}. \tag{3}$$

- The probability that a digit is nonzero is found as $1 - \frac{2}{3} = \frac{1}{3}$, which implies that the average number of nonzero digits in the canonically recoded binary number D is equal to $\frac{n}{3}$. Thus, assuming A and $-A$ are given as input, the add-and-shift canonical recoding binary multiplication algorithm requires

$$T = \frac{n}{3} \tag{4}$$

additions on the average.

- The probability that $D_{i+1} = 0$, when $D_i = 0$ is found as

$$\mathcal{P}(D_{i+1} = 0 \mid D_i = 0) = \frac{\sum_{j=0,3,4,7} \pi_0 \mathcal{P}_{0j} + \pi_3 \mathcal{P}_{3j} + \pi_4 \mathcal{P}_{4j} + \pi_7 \mathcal{P}_{7j}}{\pi_0 + \pi_3 + \pi_4 + \pi_7} = \frac{1}{2}. \tag{5}$$

- Finally, the well-known property of the canonical recoding algorithm that

$$D_{i+1} \cdot D_i = 0 \text{ for } 0 \leq i \leq n-2,$$

can easily be proven. The probability that $D_{i+1} = 0$, when $D_i = 1$ or $D_i = \bar{1}$ for all $0 \leq i \leq n-1$, is found as

$$\mathcal{P}(D_{i+1} = 0 \mid D_i = 1 \text{ or } D_i = \bar{1}) = \frac{\sum_{j=0,3,4,7} \pi_1 \mathcal{P}_{1j} + \pi_2 \mathcal{P}_{2j} + \pi_5 \mathcal{P}_{5j} + \pi_6 \mathcal{P}_{6j}}{\pi_1 + \pi_2 + \pi_5 + \pi_6} = 1.$$

4. ADAPTIVE m-ARY SEGMENTATION

The m-ary segmentation multiplication algorithm decomposes the bits of the multiplier into words of length $d = \log_2 m$ bits each. Since the probability of a word of length d being zero is 2^{-d}, longer words have smaller zero word probabilities. In step 4a of the m-ary segmentation algorithm, we skip an addition whenever the current word is equal to zero. Thus, as m grows larger, the probability that we have to perform an addition operation in step 4a gets larger. However, short word lengths will increase the total number of additions due to Equation (1).

We propose an adaptive scheme that allows zero words of variable-length and improves zero-word probability while using relatively long words in the segmentation process. The following procedure recodes the binary number $B = (B_{n-1} B_{n-2} \ldots B_1 B_0)$, as the concatenation of zero and nonzero words $B^* = (B^*_{k-1}, B^*_{k-2}, \ldots, B^*_1, B^*_0)$. We denote the length of a word B^*_j with $L(B^*_j)$. We also define W_0 and W_1 as the sets of zero words and nonzero words, respectively. The proposed algorithm recodes the multiplier into zero words of variable length and nonzero words of length d, i.e.,

$$\begin{aligned} L(B^*_j) &= 1, 2, 3, \ldots, && \text{if } B^*_j \in W_0, \\ L(B^*_j) &= d, && \text{if } B^*_j \in W_1, \end{aligned}$$

for $j = 0, 1, 2, \ldots, k-1$. The procedure given below adaptively recodes the multiplier in order to reduce the average number of additions. In this procedure, ϕ denotes the empty word.

The Adaptive m-ary Segmentation Multiplication Algorithm

1. Compute and store wA for all odd w, $3 \leq w \leq m-1$.
2. Set $j = 0$ and $B^*_0 = \phi$.
3. For $i = 0, 1, 2, \ldots, n-1$, do one of the following:
   Case 0. ($B^*_j = \phi$). Append B_i to B^*_j.
   Case 1. ($B^*_j \in W_0$ and B_i is zero). Append B_i to B^*_j.
   Case 2. ($B^*_j \in W_0$ and B_i is nonzero). Set $j = j+1$ and $B^*_j = B_i$.
   Case 3. ($B^*_j \in W_1$ and $1 \leq L(B^*_j) \leq d-2$). Append B_i to B^*_j.
   Case 4. ($B^*_j \in W_1$ and $L(B^*_j) = d-1$). Append B_i to B^*_j. Set $j = j+1$ and $B^*_j = \phi$.
4. Set $k = j+1$ and $P = 0$.
5. Repeat Step 5a for $j = k-1, k-2, \ldots, 1, 0$.
   5a. Compute $P = 2^{L(B^*_j)} P + B^*_j A$.
6. End.

For example, for $d = 3$, we recode B given in Section 2, as

$$\begin{aligned} B &= (011100101001), \\ B^* &= (0), (111), (00), (101), (001). \end{aligned}$$

In order to compute the average number of nonzero words, which represents the number of additions required during the second stage of the algorithm, we define another Markov chain. As

in Section 3, we assume that the uniformly distributed binary number is generated and recoded bit by bit. State variable S is defined as

$$S = \begin{cases} 0, & \text{if } B_j^* \in W_0, \\ L(B_j^*), & \text{if } B_j^* \in W_1, \end{cases}$$

i.e., $S = 0$ when zero words are being collected, and S is equal to the length of the current nonzero word when nonzero words are being collected. Thus, we have $S = 0, 1, 2, 3, \ldots, d$. The probability that state $S = j$ succeeds state $S = i$ is denoted by $\mathcal{P}_{ij}$. Since $\mathcal{P}(B_i = 0) = \mathcal{P}(B_i = 1) = \frac{1}{2}$, we have

$$\mathcal{P}_{00} = \mathcal{P}_{01} = \frac{1}{2}.$$

Furthermore, once the first bit is equal to 1, we collect $d-1$ more bits to obtain a nonzero word of length d, which implies

$$\mathcal{P}_{i,i+1} = 1, \qquad \text{for } 1 \le i \le d-1.$$

After all d-bits are collected, depending on the value of B_i, the next state is either $S = 0$ (when $B_i = 0$) or $S = 1$ (when $B_i = 1$), i.e.,

$$\mathcal{P}_{d0} = \mathcal{P}_{d1} = \frac{1}{2}.$$

All the other $\mathcal{P}_{ij}$ are zero. The transition probability graph is shown in Figure 1. The one-step transition probability matrix of the adaptive m-ary segmentation algorithm for $d = 5$ is given as

$$P = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{6}$$

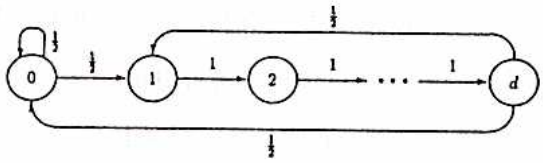

Figure 1. Transition probability graph for the adaptive m-ary segmentation algorithm.

Let C be the average number of nonzero words after all n bits have been received. This can be found by counting the number of transitions from state 0 to state 1. Let $\mathcal{P}^{(i)}$ denote the i-step transition probability matrix; then we have

$$C = \sum_{i=1}^{n} \mathcal{P}_{01}^{(i)}. \tag{7}$$

The i-step transition probability matrix $\mathcal{P}^{(i)}$ is simply found by computing the ith power of $\mathcal{P}$. An approximation for C can be given by computing the limiting probability vector π, which is found by solving the system of linear equations

$$\pi \mathcal{P} = \pi, \tag{8}$$

$$\sum_{i=0}^{d} \pi_i = 1. \tag{9}$$

This system of linear equations can be written as

$$\begin{aligned} \frac{\pi_0}{2} + \frac{\pi_d}{2} &= \pi_0, \\ \frac{\pi_0}{2} + \frac{\pi_d}{2} &= \pi_1, \\ \pi_i &= \pi_{i+1} \qquad \text{for } 1 \le i \le d-1, \\ \pi_0 + \pi_1 + \pi_2 + \cdots + \pi_d &= 1. \end{aligned}$$

Let $p = \pi_1 = \pi_2 = \cdots = \pi_d$, then we write this system of linear equations as

$$\begin{aligned} \pi_0 - p &= 0, \\ \pi_0 + dp &= 1, \end{aligned}$$

the solution of which give the limiting probabilities as

$$\pi_0 = \pi_1 = \pi_2 = \cdots = \pi_d = \frac{1}{d+1}.$$

For large n, we can approximate the average number of nonzero words as the number of times state 1 is visited in n steps, i.e.,

$$C \approx \tilde{C} = \frac{n}{d+1}. \tag{10}$$

This approximation is quite accurate, e.g., for $d = 3$ and $n = 64$, we have $C = 16.19$ and $\tilde{C} = 16.0$, and for $n = 128$, $C = 32.19$ and $\tilde{C} = 32.0$. The total number of additions required by the adaptive m-ary segmentation algorithm is equal to

$$T = \frac{m-2}{2} + C \approx \frac{m-2}{2} + \frac{n}{d+1}. \tag{11}$$

The least significant bit of every nonzero word must be 1. Thus, we need to compute wA only for $w = 3, 5, 7, \ldots, m-1$, i.e., $\frac{m-2}{2}$ additions are performed during the preprocessing stage. Similar to the m-ary segmentation multiplication algorithm, the total number of additions required by the adaptive m-ary segmentation multiplication algorithm can be minimized with proper selection of m. The optimal values of d are given in Table 1.

5. ADAPTIVE m-ARY SEGMENTATION CANONICAL RECODING

The canonical recoding algorithm reduces the number of additions, by recoding the multiplier, to obtain a signed-digit number with fewer nonzero digits [1,7]. On the other hand, the m-ary segmentation algorithm decomposes the multiplier B into m-ary words, and reduces the average number of additions by preprocessing certain multiples of multiplicand A. If we combine these two algorithms, i.e., apply constant-length segmentation to the canonically recoded signed-digit number, we will not have a satisfactory result, because:

- The probability that a word of length d is zero is much less than $\frac{2}{3}$, which is the probability to have a zero bit in canonically recoded signed-digit number D.
- Compared to m-ary segmentation, the preprocessing time is longer, since there are more d-digit signed-digit numbers than there are d-bit binary numbers. For example, for $d = 2$, the set of binary nubmers is $\{00, 01, 10, 11\}$, whereas the set of signed-digit numbers is $\{00, 01, 0\bar{1}, 10, \bar{1}0\}$.

Instead, we propose to combine the adaptive m-ary segmentation algorithm and the canonical recoding algorithm, to obtain the adaptive m-ary segmentation canonical recoding algorithm, which takes advantage of the increased zero bit probability introduced by canonical recoding and the reduced total computation time provided by m-ary segmentation. This algorithm applies adaptive segmentation to the signed-digit number $D = (D_{n-1}\, D_{n-2} \ldots D_1\, D_0)$ to obtain $D^* = (D^*_{k-1}\, D^*_{k-2} \ldots D^*_1\, D^*_0)$.

The Adaptive m-ary Segmentation Canonical Recoding Multiplication Algorithm

1. Canonically recode B to obtain D.
2. Compute and store wA for all canonically recoded d-digit numbers.
3. Execute steps 2–5 of the adaptive m-ary segmentation algorithm using D.
4. End.

For example, we first canonically recode $B = (011100101001)$ to obtain D and then adaptively decompose D to obtain D^* as

$$\begin{aligned} B &= (011100101001), \\ D &= (100\bar{1}00101001), \\ D^* &= (001), (00\bar{1}), (00), (101), (001). \end{aligned}$$

The average number of nonzero words, i.e., the number of additions required in the second stage of the adaptive m-ary segmentation canonical recoding algorithm, can also be computed with the aid of the Markov chains. The Markov chain corresponding to the adaptive m-ary segmentation canonical recoding multiplication algorithm also has $d+1$ states. The probability that state 0 is succeeded by state 0 is equal to $\mathcal{P}_{00} = \frac{1}{2}$, since $\mathcal{P}(D_{i+1} = 0 \mid D_i = 0) = \frac{1}{2}$, due to Equation (5) in Section 3. Similarly, $\mathcal{P}_{01} = \frac{1}{2}$. After d bits have been collected to form a nonzero word, depending on the value of D_i, either state 0 (when $D_i = 0$) or state 1 (when $D_i = 1$ or $D_i = \bar{1}$) succeeds state d. It follows from Equation (3) that we have $\mathcal{P}_{d0} = \frac{2}{3}$ and $\mathcal{P}_{d1} = \frac{1}{2}$, for large n.

The transition probability graph of the adaptive m-ary segmentation canonical recoding multiplication algorithm is shown in Figure 2. The one-step transition probability matrix P of the adaptive m-ary segmentation canonical recoding multiplication algorithm for $d = 5$ is given below.

$$P = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ \frac{2}{3} & \frac{1}{3} & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{12}$$

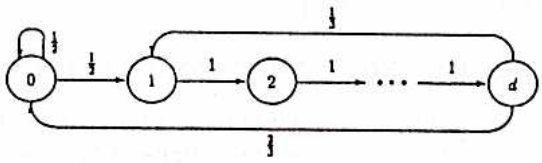

Figure 2. Transition probability graph for the adaptive m-ary segmentation canonical recoding algorithm.

The average number of nonzero is computed by computing the i-step transition probability matrix of the adaptive m-ary segmentation canonical recoding multiplication algorithm for $1 \leq i \leq n$. Using the technique given in the previous section, we find the approximate value of C by computing the limiting probabilities of all $d+1$ states. The solution of the system of linear equations given by (8) and (9), using the one-step transition probability matrix P in (12), is

$$\pi_0 = \frac{4}{3d+4}, \quad \text{and } \pi_1 = \pi_2 = \cdots = \pi_d = \frac{3}{3d+4}.$$

We thus find the approximate value of the average number of nonzero words for the adaptive m-ary segmentation canonical recoding multiplication algorithm, as

$$\tilde{C} = \frac{3n}{3d+4}. \tag{13}$$

The preprocessing time for the adaptive m-ary segmentation canonical recoding multiplication is the time required to compute wA for all canonically recoded binary numbers w with d bits. Note that the least significant digit of w is either 1 or $\bar{1}$, which implies that w is always an odd number. The largest positive value of w is

$$w_{\max} = \begin{cases} (1010\ldots101001) = \frac{1}{3}(2^{d+1}-5), & \text{if } d \text{ is even,} \\ (1010\ldots10101) = \frac{1}{3}(2^{d+1}-1), & \text{if } d \text{ is odd,} \end{cases}$$

The smallest positive value of w is 1. Also the smallest difference between any two positive w is equal to 2 (recall that w is always odd). Therefore, the number of d-digit canonically recoded positive numbers is found as $\frac{w_{\max}+1}{2}$. Additionally, there are as many negative w as there are positive w (the negative numbers are simply obtained by reversing 1's and $\bar{1}$'s). This implies that the total number of d-digit canonically recoded is equal to

$$w_{\max} + 1 = \frac{2}{3}[2^d + (-1)^{d+1}].$$

First we compute wA where w contains only one nonzero digit. After this, each value wA can be computed recursively from the already computed values by a single addition. Since A and $-A$ are already available, it follows that the total number of additions required by the adaptive m-ary segmentation canonical recoding multiplication algorithm is $w_{\max} + 1 - 2 + C$, i.e.,

$$\begin{aligned} T &= \frac{2}{3}[2^d + (-1)^{d+1}] - 2 + C, \\ &\approx \frac{2}{3}[2^d + (-1)^{d+1}] - 2 + \frac{3n}{3d+4}. \end{aligned} \tag{14}$$

Table 4. The average number of additions required by the algorithms.

Algorithm	preprocessing stage	add-and-shift stage
Canonical	–	$\frac{n}{3}$
m-ary	$m-2$	$\frac{n}{d}(1-\frac{1}{m})$
Adaptive m-ary	$\frac{m-2}{2}$	$\frac{n}{d+1}$
Adaptive m-ary canonical	$\frac{2}{3}[m+(-1)^{d+1}]-2$	$\frac{3n}{3d+4}$

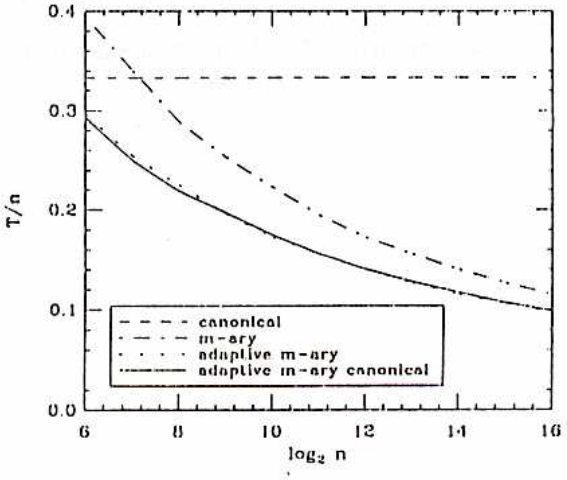

Figure 3. Comparison of the algorithms.

6. CONCLUSION

In Table 4 we summarize the average number of additions required by the add-and-shift algorithms. For the m-ary segmentation algorithms, the optimum values of $d = \log_2 m$ are found in Table 1 for $n = 2^6, 2^7, 2^8, \ldots, 2^{16}$. We have not considered the case where $n < 64$, since there exists fast multiplier circuits to multiply binary numbers with 32, 56, or 64 bits [1,2]. Furthermore, most advanced microprocessors can multiply integers as large as 2^{64}. The algorithms presented in this paper can be used for the multiplication of large binary numbers, particularly numbers with several hundred bits. There exist asymptotically faster algorithms for multiplication, e.g., the Toom-Cook algorithm and the Schönhage-Strassen algorithm [10]. However, segmentation and canonical recoding methods are easier to implement, do not require recursive computation or use of Fast Fourier Transform; the overhead is negligible.

Using the optimal values of d, the average number of additions required by the algorithms are plotted in Figure 3. We see that:

- When n is large (i.e., $n > 64$), the variable-length segmentation strategy is clearly superior to the constant-length segmentation.
- The canonical recoding algorithm requires fewer additions than the m-ary segmentation algorithm for $n < 128$. The adaptive m-ary segmentation algorithms require the fewest number of additions for all $n > 64$.
- For $n > 64$, the average number of additions required by the adaptive m-ary segmentation algorithm is approximately the same as that of the adaptive m-ary segmentation canonical recoding algorithm. Thus, when $n > 64$, if the adaptive segmentation strategy is applied, there is no need to canonically recode the multiplier.

References

1. K. Hwang, *Computer Arithmetic, Principles, Architecture, and Design*, John Wiley and Sons, Inc., (1979).
2. S. Waser and M. J. Flynn, *Introduction to Arithmetic for Digital System Designers*, Holt, Rinehart and Winston, CBS College Publishing, (1982).
3. L. Dadda, On parallel digital multipliers, *Alta Frequenza* **45**, 574–580 (1976). (Also reprinted in [11], pp. 126–132).
4. W. J. Stenzel, W. J. Jubitz and G. H. Garcia, A compact high speed parallel multiplication scheme, *IEEE Transactions on Computers* **26** (10), 948–957 (October 1977). (Also reprinted in [11], pp. 133–142).
5. A. D. Booth, A signed binary multiplication technique, *Q. J. Mech. Appl. Math.* **4** (2), 236–240 (1951). (Also reprinted in [11], pp. 100–104).
6. O. L. MacSorley, High-speed arithmetic in binary computers, *Proceedings of the IRE* **49**, 67–91 (January 1961). (Also reprinted in [11], pp. 14–38).
7. G. W. Reitwiesner, Binary Arithmetic, *Advances in Computers* **1**, 231–308 (1960).
8. J. G. Kemeny and J. L. Snell, *Finite Markov Chains*, Springer-Verlag, (1976).
9. S. M. Ross, *Introduction to Probability Models*, Academic Press, 3rd edition, (1985).
10. D. E. Knuth, *The Art of Computer Programming*, Volume 2, Seminumerical Algorithms, Addison-Wesley Publishing Co., 2nd edition, (1981).
11. E. E. Swartzlander, Ed., *Computer Arithmetic*. Benchmark Papers in Electrical Engineering and Computer Science, Vol. 21, Dowden, Hutchinson & Ross, Inc., (1980).