An Accumulation Algorithm for Distributive Multiplication

Rapheal Akwesi Foder
Diabene Senior High Technical School, Ketan, Sekondi-Takoradi, Ghana
Contact: (+233) 24 766 2516
Email: outsidequality2@gmail.com

Abstract

This preprint presents an accumulation algorithm for distributive multiplication. The method originates from an observation made during a secondary-school mathematics lesson, in which a student, while the teacher demonstrated conventional multiplication, applied a different arrangement of partial products and arrived at identical results. The procedure multiplies each digit of one factor by the entire other factor, combines adjacent partial products by accumulating the leading part of one into the running total of the preceding one, and appends the final digit. Three variants are formalized: two-digit by one-digit, two-digit by two-digit, and general m-digit by n-digit multiplication. For each variant, an explicit formula and worked examples are provided. A formal theorem establishes exactness for all integers, and a proof by induction is given. The method is shown to be algebraically equivalent to the distributive law combined with Horner's scheme with carry correction. Furthermore, the algorithm is reversible, allowing the recovery of intermediate values and original factors from the final product. The work is intended as a pedagogical exposition of an alternative organizational route to the same arithmetic result, not as a claim of a new arithmetic identity.

Keywords: multiplication algorithm; distributive law; partial products; Horner's scheme; accumulation; carry propagation; reverse engineering; pedagogy of arithmetic.

1 Introduction

Multiplication of multi-digit numbers is traditionally taught through column alignment, digit-by-digit products, and carried values. This arrangement is efficient but can obscure the underlying distributive structure, because the learner's attention is divided between the multiplication table, the placement of partial products, and the management of carries. Alternative organizations of the same arithmetic content have therefore been of recurring interest in mathematics education, both for remediation and for enrichment.

The present work documents one such alternative. During a mathematics lesson at Diabene Senior High Technical School, while the teacher was demonstrating multiplication on the board, the author applied a different arrangement of the same partial products and obtained the same answers. The arrangement was subsequently tested on further examples and found to hold. This

preprint formalizes that arrangement, states it as a theorem, supplies a proof, and positions it relative to established algorithms.

The method is referred to here as the **accumulation algorithm**. The term "accumulation" reflects the procedure of maintaining a running total, into which the leading part of each partial product is folded before the final digit is appended. The term "algorithm" reflects the fact that the method provides a step-by-step procedure that reorganizes, rather than replaces, the distributive law.

The contribution of this preprint is organizational and pedagogical. The arithmetic content is classical: the distributive law and Horner's scheme are both well established. What is offered is an explicit statement of a particular accumulation-based ordering of those classical ingredients, together with a correctness theorem, a set of worked examples suitable for classroom use, and a reverse-engineering methodology for factor recovery.

The remainder of the paper is organized as follows. Section 2 establishes notation. Sections 3, 4, and 5 present the three variants with formulae and examples. Section 6 states and proves the general theorem and provides extended examples. Section 7 introduces the reverse engineering of the algorithm. Section 8 relates the method to existing algorithms. Section 9 discusses the advantages of bypassing traditional methods. Section 10 concludes.

2 Notation and Preliminaries

Let N be an integer written in base 10 as

$$N = \sum_{i=0}^{n-1} d_i \, 10^i, \qquad d_i \in \{-9, \ldots, 9\},$$

where the leading digit $d_{n-1} \neq 0$ and the sign of N is carried by the digits. (For example, -47 is written with $d_1 = -4$, $d_0 = -7$.)

For an integer X, define the **truncated quotient** and the **signed remainder** by

$$q = \lambda(X) = \operatorname{trunc}\left(\frac{X}{10}\right), \qquad r = \ell(X) = X - 10\,\lambda(X),$$

where trunc denotes truncation toward zero. Under this convention $r \in \{-9, \ldots, 9\}$, and the identity

$$X = 10q + r \tag{1}$$

holds for every integer X, positive, negative, or zero. This separation is used at every carry-handling step below.

3 Variant A: Two-Digit by One-Digit Multiplication

3.1 Procedure

Let the two-digit number be $10a + b$ and the multiplier be c.

1. Compute $P_1 = ac$.

2. Compute $P_2 = bc$.

3. Split P_2 into q and r using identity (1).

4. Form the result $R = 10(P_1 + q) + r$.

3.2 Justification

$$(10a + b)c = 10ac + bc = 10P_1 + P_2 = 10P_1 + (10q + r) = 10(P_1 + q) + r.$$

The step is exact for all digits a, b, c.

3.3 Examples

Example 3.1. 68×8.

- $P_1 = 6 \times 8 = 48$.
- $P_2 = 8 \times 8 = 64$. Split: $q = 6$, $r = 4$.
- $R = 10(48 + 6) + 4 = 544$.

Verification: $68 \times 8 = 544$. ✓

Example 3.2. 81×5.

- $P_1 = 8 \times 5 = 40$.
- $P_2 = 1 \times 5 = 5$. Split: $q = 0$, $r = 5$.
- $R = 10(40 + 0) + 5 = 405$.

Verification: $81 \times 5 = 405$. ✓

4 Variant B: Two-Digit by Two-Digit Multiplication

4.1 Procedure

Let the factors be $10a + b$ and $10c + d$.

1. Compute $P_1 = a(10c + d)$.

2. Compute $P_2 = b(10c + d)$.

3. Split P_2 into q and r using identity (1).

4. Form the result $R = 10(P_1 + q) + r$.

4.2 Justification

$(10a + b)(10c + d) = 10a(10c + d) + b(10c + d) = 10P_1 + P_2 = 10P_1 + 10q + r = 10(P_1 + q) + r.$

4.3 Examples

Example 4.1. 21×34.

- $P_1 = 2 \times 34 = 68$.
- $P_2 = 1 \times 34 = 34$. Split: $q = 3$, $r = 4$.
- $R = 10(68 + 3) + 4 = 714$.

Example 4.2. 54×18.

- $P_1 = 5 \times 18 = 90$.
- $P_2 = 4 \times 18 = 72$. Split: $q = 7$, $r = 2$.
- $R = 10(90 + 7) + 2 = 972$.

Example 4.3. 99×99.

- $P_1 = 9 \times 99 = 891$.
- $P_2 = 9 \times 99 = 891$. Split: $q = 89$, $r = 1$.
- $R = 10(891 + 89) + 1 = 9801$.

5 Variant C: General m-Digit by n-Digit Multiplication

5.1 Procedure

Let $A = \sum_{i=0}^{m-1} a_i 10^i$ and $B = \sum_{j=0}^{n-1} b_j 10^j$.

1. For each $i = 0, 1, \ldots, m-1$, compute $P_i = a_i B$.
2. Initialize the accumulator $S = P_{m-1}$.
3. For $i = m-2, m-3, \ldots, 0$, update:

$$S \leftarrow 10\big(S + q_i\big) + r_i \tag{2}$$

   where $P_i = 10q_i + r_i$.

4. The result is $R = S$.

6 Main Theorem and Extended Examples

6.1 The General Formula

The operation of appending a digit r to an accumulated value X in base 10 is mathematically equivalent to shifting X by one decimal place and adding r. Therefore, the general computing formula is:

$$S_n = 10(P_1 + q) + r \tag{3}$$

where $q = \lambda(P_2)$ and $r = \ell(P_2)$.

6.2 Main Theorem

Theorem 1. *Let $A = \sum_{i=0}^{m-1} a_i 10^i$ and B be integers, with signed digits $a_i \in \{-9, \ldots, 9\}$. For each i, let $P_i = a_i B$, and let $q_i = \lambda(P_i)$, $r_i = \ell(P_i)$ be the truncated quotient and signed remainder. Define $S_{m-1} = P_{m-1}$ and, for $k = m-2, \ldots, 0$,*

$$S_k = 10\big(S_{k+1} + q_k\big) + r_k.$$

Then $S_0 = AB$.

Proof. Define the signed suffix value $A_k = \sum_{i=k}^{m-1} a_i 10^{i-k}$. The claim is $S_k = A_k B$. The base case holds by initialization. For the inductive step, suppose $S_{k+1} = A_{k+1} B$. Since $A_k = 10A_{k+1} + a_k$,

$$A_k B = 10A_{k+1}B + a_k B = 10S_{k+1} + P_k.$$

By identity (1), $P_k = 10q_k + r_k$, hence

$$A_k B = 10S_{k+1} + 10q_k + r_k = 10(S_{k+1} + q_k) + r_k = S_k.$$

By induction, $S_0 = A_0 B = AB$. □

6.3 Extended Worked Examples

The following examples demonstrate the insertion of values into formula (3), including the explicit splitting of q and r for artificial products involving positive and negative integers.

Case 1: Both Positive. 68×8.

- $P_1 = 48$, $P_2 = 64$.
- Split P_2: $q = 6$, $r = 4$.
- $S_n = 10(48 + 6) + 4 = 10(54) + 4 = 544$.

Case 2: One Positive, One Negative. $25 \times (-5)$.

- $P_1 = 2 \times (-5) = -10$.
- $P_2 = 5 \times (-5) = -25$.
- Split P_2: $q = -2$, $r = -5$.
- $S_n = 10(-10 + (-2)) + (-5) = 10(-12) - 5 = -125$.

Case 3: Both Negative. $(-25) \times (-20)$.

- $P_1 = (-2) \times (-20) = 40$.
- $P_2 = (-5) \times (-20) = 100$.
- Split P_2: $q = 10$, $r = 0$.
- $S_n = 10(40 + 10) + 0 = 10(50) + 0 = 500$.

Case 4: One Negative, One Positive. $(-47) \times 6$.

- $P_1 = (-4) \times 6 = -24$.
- $P_2 = (-7) \times 6 = -42$.
- Split P_2: $q = -4$, $r = -2$.
- $S_n = 10(-24 + (-4)) + (-2) = 10(-28) - 2 = -282$.

Case 5: Zero Involved. 34×0.

- $P_1 = 3 \times 0 = 0$.
- $P_2 = 4 \times 0 = 0$.
- Split P_2: $q = 0$, $r = 0$.
- $S_n = 10(0 + 0) + 0 = 0$.

Case 6: Zero Digit Inside a Factor. $30 \times (-26)$.

- $P_1 = 3 \times (-26) = -78$.
- $P_2 = 0 \times (-26) = 0$.
- Split P_2: $q = 0$, $r = 0$.
- $S_n = 10(-78 + 0) + 0 = -780$.

Case 7: Two Digits by Two Digits. 34×26.

- $P_1 = 3 \times 26 = 78$.
- $P_2 = 4 \times 26 = 104$.
- Split P_2: $q = 10$, $r = 4$.
- $S_n = 10(78 + 10) + 4 = 10(88) + 4 = 884$.

7 Reverse Engineering the Algorithm

The accumulation algorithm can be systematically reversed. If the final result S_n is known alongside any one of the intermediate values (P_1, q, or r), the original factors can be uniquely recovered. If only S_n is known, multiple valid factor pairs may exist.

7.1 Reverse Quotient Extraction

The reverse-engineering process begins with the extraction of the quotient (q) and the remainder (r). Starting from the general formula $S_n = 10(P_1 + q) + r$, expanding the terms yields $S_n = 10P_1 + 10q + r$. Rearranging to isolate the quotient and remainder gives:

$$q(r) = S_n - 10P_1 \tag{4}$$

The notation $q(r)$ denotes an integer formed by q in the higher-order positions and r in the units position. Because the operation is strictly base-10, the splitting rule is absolute: the very last digit of the evaluated result is always r, and all preceding digits form q.

7.2 Scenario 1: Given S_n and P_1

Formula: $S_n - 10P_1 = 10q + r$

Method: Subtract $10P_1$ from S_n. The last digit of the result is r, and the front digits form q, yielding $P_2 = 10q + r$. The common factor B (where $B = 10c + d$) dividing both P_1 and P_2 is found. Then $a = P_1/B$ and $b = P_2/B$. The original factors are $(10a + b)$ and B.

Examples:

- **Example 7.1.1:** Given $S_n = 544$ and $P_1 = 48$.
  - $544 - 10(48) = 64$. Last digit $r = 4$, front $q = 6$. $P_2 = 64$.
  - The greatest common divisor of 48 and 64 is 16. $B = 16$.
  - $a = 48/16 = 3$; $b = 64/16 = 4$. Original factors: 34×16.
- **Example 7.1.2:** Given $S_n = 9102$ and $P_1 = 888$.
  - $9102 - 10(888) = 222$. Last digit $r = 2$, front $q = 22$. $P_2 = 222$.
  - The greatest common divisor of 888 and 222 is 222. $B = 222$.
  - $a = 888/222 = 4$; $b = 222/222 = 1$. Original factors: 41×222.
- **Example 7.1.3:** Given $S_n = 714$ and $P_1 = 68$.
  - $714 - 10(68) = 34$. Last digit $r = 4$, front $q = 3$. $P_2 = 34$.
  - The greatest common divisor of 68 and 34 is 34. $B = 34$.
  - $a = 68/34 = 2$; $b = 34/34 = 1$. Original factors: 21×34.

7.3 Scenario 2: Given S_n and q

Formula: $10P_1 + r = S_n - 10q$

Method: Subtract $10q$ from S_n. The last digit is r; the preceding digits form $10P_1$. Divide $10P_1$ by 10 to obtain P_1. Calculate $P_2 = 10q + r$. Find the common factor B of P_1 and P_2, then $a = P_1/B$ and $b = P_2/B$.

Examples:

- **Example 7.2.1:** Given $S_n = 544$ and $q = 6$.

- $544 - 10(6) = 484$. Last digit $r = 4$; front digits 48 indicate $10P_1 = 480$.
- $P_1 = 480/10 = 48$. $P_2 = 10(6) + 4 = 64$.
- The greatest common divisor of 48 and 64 is 16. $a = 3, b = 4$. Original factors: 34×16.

- **Example 7.2.2:** Given $S_n = 9102$ and $q = 22$.
  - $9102 - 10(22) = 8882$. Last digit $r = 2$; front digits 888 indicate $10P_1 = 8880$.
  - $P_1 = 8880/10 = 888$. $P_2 = 10(22) + 2 = 222$.
  - The greatest common divisor of 888 and 222 is 222. $a = 4, b = 1$. Original factors: 41×222.

- **Example 7.2.3:** Given $S_n = 714$ and $q = 3$.
  - $714 - 10(3) = 684$. Last digit $r = 4$; front digits 68 indicate $10P_1 = 680$.
  - $P_1 = 680/10 = 68$. $P_2 = 10(3) + 4 = 34$.
  - The greatest common divisor of 68 and 34 is 34. $a = 2, b = 1$. Original factors: 21×34.

7.4 Scenario 3: Given S_n and r

Formula: $\frac{S_n - r}{10} = P_1 + q$

Method: Subtract r from S_n and divide by 10 to obtain the sum $P_1 + q$. Guess a digit b and factor B such that $b \times B$ ends in r, giving P_2. Extract q from P_2. Calculate $P_1 = (P_1 + q) - q$. Find $a = P_1/B$. The original factors are $(10a + b)$ and B.

Examples:

- **Example 7.3.1:** Given $S_n = 544$ and $r = 4$.
  - $(544 - 4)/10 = 54$. Thus $P_1 + q = 54$.
  - Testing P_2 ending in 4 yields $P_2 = 64$ (so $q = 6$).
  - $P_1 = 54 - 6 = 48$. The greatest common divisor of 48 and 64 is 16. $a = 3, b = 4$. Original factors: 34×16.

- **Example 7.3.2:** Given $S_n = 9102$ and $r = 2$.
  - $(9102 - 2)/10 = 910$. Thus $P_1 + q = 910$.
  - Testing P_2 ending in 2 yields $P_2 = 222$ (so $q = 22$).
  - $P_1 = 910 - 22 = 888$. The greatest common divisor of 888 and 222 is 222. $a = 4, b = 1$. Original factors: 41×222.

- **Example 7.3.3:** Given $S_n = 714$ and $r = 4$.
  - $(714 - 4)/10 = 71$. Thus $P_1 + q = 71$.
  - Testing P_2 ending in 4 yields $P_2 = 34$ (so $q = 3$).
  - $P_1 = 71 - 3 = 68$. The greatest common divisor of 68 and 34 is 34. $a = 2, b = 1$. Original factors: 21×34.

7.5 Scenario 4: Given Only S_n

Method: List all factor pairs of S_n. There is no unique solution. Multiple valid factor pairs will satisfy the accumulation algorithm. To verify a pair, split one factor as $10a + b$ and test if the algorithm produces S_n.

Examples:

- **Example 7.4.1:** Given $S_n = 544$.
  - Valid original problems include: 68×8, 34×16, and 17×32. Thus, the original factors cannot be uniquely determined.
- **Example 7.4.2:** Given $S_n = 714$.
  - Valid original problems include: 21×34, and 51×14. Thus, the original factors cannot be uniquely determined.
- **Example 7.4.3:** Given $S_n = 405$.
  - Valid original problems include: 81×5, and 45×9. Thus, the original factors cannot be uniquely determined.

8 Relationship to Existing Algorithms

The method presented here is not a new arithmetic identity. It is an exact reorganization of established procedures.

Distributive law and partial products. The decomposition $AB = \sum_i a_i 10^i B$ is the distributive law applied to base-10 place value. Each $P_i = a_i B$ is a partial product. The traditional column method computes these same partial products and sums them after shifting. The present method computes them in the same order but folds the shift into the accumulator at each step.

Horner's scheme. The recurrence $S \leftarrow 10(S + q_i) + r_i$ is Horner's rule applied to the digit sequence of A. The carry $q_i = \lambda(P_i)$ is folded into the accumulator and the remainder $r_i = \ell(P_i)$ is appended. The method is therefore a Horner evaluation of the digit polynomial of A at $x = 10$.

Vedic mathematics. The Urdhva-Tiryagbhyam sutra arranges products crosswise, forming combinations such as $a_i b_j + a_j b_i$. The present method arranges products rowwise, forming $a_i B$. The two arrangements are distinct.

9 Advantages of the Accumulation Algorithm

The accumulation algorithm offers specific pedagogical and mechanical advantages by bypassing the traditional column-alignment and carry-propagation methods.

Bypassing Carried-Value Bookkeeping. In traditional multiplication, the learner must simultaneously multiply, align columns, and add carried values. The accumulation algorithm isolates the carry. By explicitly defining q (the leading part) and r (the remainder), the method

forces the learner to separate the value that must be added from the value that must be appended. The carried value is thus handled in a single, discrete step rather than being tracked over the top of the multiplication table.

Bypassing Column Alignment. Traditional multiplication requires the placement of partial products in staggered columns, which is a frequent source of spatial errors. The accumulation algorithm uses a running accumulator S. The shift-and-append operation $(10S + r)$ inherently aligns the digits by moving the decimal place, eliminating the need to visually align staggered rows.

Direct Mapping to the General Formula. The formula $S_n = 10(P_1 + q) + r$ provides a computational template. The learner computes P_1, computes P_2, splits P_2 into q and r, adds $P_1 + q$, multiplies by 10, and adds r. The mechanical steps are sequential and independent, reducing cognitive load and the likelihood of procedural errors.

Unified Treatment of Signed Integers. As demonstrated in Section 6.3, the formula $S_n = 10(P_1 + q) + r$ handles positive, negative, and zero values without modification. When r is negative, the final step performs a subtraction rather than an addition, which naturally maintains the sign of the result. Traditional column multiplication requires separate rules and explicit sign management for negative integers. The accumulation algorithm unifies these cases under a single algebraic identity.

Reverse Engineering Capability. The algebraic structure of the algorithm allows for the reverse extraction of the quotient, remainder, and original factors, as demonstrated in Section 7. This provides a built-in mechanism for calculation verification and inverse problem solving, which is not structurally present in the traditional column method.

10 Conclusion

This preprint has formalized an accumulation algorithm for distributive multiplication, originating from a classroom observation in which a student applied a different arrangement of partial products and obtained the same results as the conventional method. Three variants have been stated with explicit formulae. A theorem has been stated and proved, establishing exactness for all integers under the truncation-toward-zero convention. Worked examples have been supplied for positive, negative, and zero cases. Furthermore, the algorithm has been shown to be reversible, allowing for the recovery of intermediate values and original factors under specified conditions.

The method is exact, elementary, and pedagogically transparent. It is not a new arithmetic identity, and it offers no asymptotic advantage over schoolbook multiplication. Its contribution is organizational: it presents the classical distributive content in an order that bypasses traditional column alignment and carry bookkeeping, making the process of shifting and appending explicit. It is offered as a supplementary exposition for the teaching and learning of multiplication, and as a documented example of a student-generated alternative route to a standard arithmetic result.

Appreciation

The author expresses sincere appreciation to DeepSeek for the formal organization of the formulae presented in this preprint. The author also expresses sincere appreciation to his elective mathematics teacher, Batu J. J. Yagah, for guidelines leading to the final output of this material, and for encouragement during the development of the method in the classroom setting from which it originated.

References

[1] Brousseau, G. (1997). *Theory of didactical situations in mathematics: Didactique des mathématiques, 1970–1990*. Kluwer Academic Publishers.

[2] Knuth, D. E. (1998). *The art of computer programming, Volume 2: Seminumerical algorithms* (3rd ed.). Addison-Wesley.

[3] Ma, L. (1999). *Knowing and teaching elementary mathematics: Teachers' understanding of fundamental mathematics in China and the United States*. Lawrence Erlbaum Associates.

[4] Williams, K. R. (2005). *Vedic mathematics* (Teacher's manual, Vol. 3). Motilal Banarsidass Publishers.

Correspondence concerning this preprint should be addressed to Rapheal Akwesi Foder, Diabene Senior High Technical School, Ketan, Sekondi-Takoradi, Ghana. Email: `outsidequality2@gmail.com`. *Telephone: (+233) 24 766 2516.*