PERGAMON Computers and Mathematics with Applications 0 (2005) 1–0
www.elsevier.com/locate/camwa

A New Symbolic Substitution Based Addition Algorithm

T. Imam* and M. Kaykobad
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology Dhaka
Dhaka, 1000, Bangladesh
tasadduq25@yahoo.com
kaykobad@cse.buet.ac.bd

(Received June 2004; revised and accepted June 2005)

Abstract—Symbolic substitution, a parallel processing technique, has been proposed in computing literature to perform fast, carry-free addition of numbers. Several algorithms and corresponding symbolic substitution tables have been developed for addition of two numbers represented in binary, modified signed digit (MSD), and canonical modified signed digit (CMSD) number systems. In this paper, however, we present a new symbolic substitution based algorithm and corresponding substitution table for the addition of two numbers represented in canonical modified signed digit (CMSD) notation. In contrast with the existing algorithm in this regard, that derives the addition result in MSD system, our algorithm derives the result in CMSD notation and thus it could be employed to perform symbolic substitution based associative addition of a set of CMSD numbers. © 2005 Elsevier Science Ltd. All rights reserved.

Keywords—Canonical modified signed digit, Symbolic substitution, Associative addition, Parallel processing, Substitution step.

1. INTRODUCTION

Addition is a basic operation involved in any arithmetic processing and so speeding up addition will in its turn accelerate the whole arithmetic process. Researchers have, therefore, made efforts to speed up addition by finding out appropriate computer representation of numbers and corresponding parallel carry-free addition process. Symbolic substitution, basically an iterative parallel pattern replacement process, has been proposed [1–4] to determine the resultant of addition of two numbers, represented in binary, modified signed digit (MSD), and canonical modified signed digit (CMSD) number systems. An important parameter of any symbolic substitution process is the required number of substitution steps. Three-step and two-step based algorithms for addition of numbers represented in MSD notation and a one-step based algorithm for addition of numbers represented in CMSD notation have already been proposed [2,3,5–7]. Symbolic

The authors would like to thank Professor M.A. Karim, Dean, Faculty of Engineering of the City College of the City University of New York, and the anonymous reviewers for their valuable suggestions.
*Address: 1/2 White Parade, Churchill, Victoria-3842, Australia.

PII:00

substitution processes for addition of binary numbers, that require steps equal to the bit-size of the numbers, have also been developed. In this paper, however, we present a new algorithm for addition of numbers represented in CMSD notation. Whereas the earlier algorithm derived the addition result in MSD notation, our algorithm derives the result in CMSD notation. The advantage of such an algorithm is that it can be applied to perform associative addition of a set of CMSD numbers. We also analyze the efficiency of our symbolic substitution process in terms of the required number of substitution steps when an optimization is introduced in the process.

2. SYMBOLIC SUBSTITUTION

The symbolic substitution technique [1–3,8] was developed as an optical computing method to take advantage of the optical parallelism for 2-D image processing and arithmetic computation. At the heart of the process is a pattern replacement operation, defined by a symbolic substitution rule, that converts some given patterns to some desired patterns. The substitution process, therefore, comprises two basic phases. In the first phase, *the recognition phase*, the inputs are scanned for the location of desired input patterns. In the second phase, *the substitution phase*, the desired output pattern is written into all the locations determined by the previous phase.

Figure 1 depicts the basic block architecture for a symbolic substitution process. The inputs to the system are the variables X & Y; C is an auxiliary variable and O is the output variable. At each substitution step, the symbolic substitution unit derives O and C as output patterns, based on the given input patterns, X and Y. The symbolic substitution rules are stored in memory, while the controller controls the recognition and substitution phase. For some applications multiple substitution steps are required. In such cases, the value of O and C are copied to X and Y, and the substitution process is iterated with the new value of X and Y.

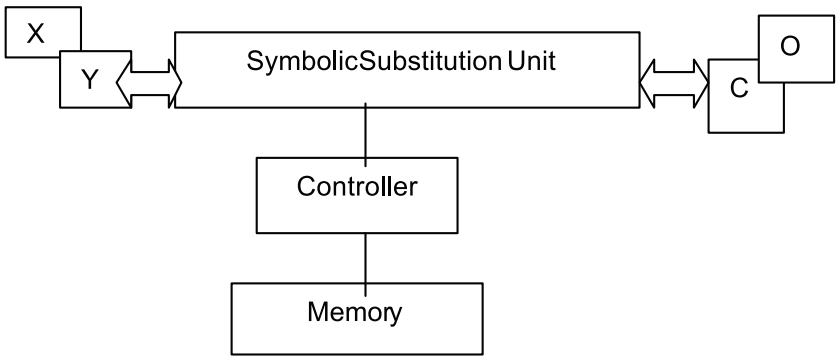

Figure 1. Symbolic substitution unit architecture.

Symbolic substitution technique has found its application in performing the addition of numbers [1–3,8]. The goal has been the development of symbolic substitution rules such that all the resultant bits can be derived in parallel. Addition of binary numbers may result in carries that propagate their effects on all the bits involved. This phenomenon is a hindrance to making the addition operation parallel. So researchers have proposed a number of nonbinary systems that allow carry-free or limited carry propagation arithmetic [3,9–13]. Modified signed digit (MSD) is one of these nonbinary number systems that have found broad applications in optical computing, especially in the context of symbolic substitution process. MSD is a weighted 2-based number system that comprises three types of digits: 0, 1, and -1. A symbolic substitution table and algorithm that requires only three steps and two steps have also been developed [3]. However, representation of numbers in MSD is nonunique and a variation of the MSD, the canonical modified signed digit (CMSD) system, was proposed. The highlighting feature of CMSD is that no two

consecutive digits in the representation are nonzero. This results in the unique representation of each number. A one-step symbolic substitution process for addition of CMSD number has already been developed and is as shown in Table 1 [2]. For convenience, throughout this document, we have used the symbol $\underline{1}$ to denote -1. In Table 1, D indicates the don't care condition.

Table 1. Symbolic substitution table for CMSD addition: one step.

A_i	B_i	A_{i-1}	B_{i-1}	S_i
$\underline{1}$	0	0	D	$\underline{1}$
0	$\underline{1}$	D	0	$\underline{1}$
0	0	$\underline{1}$	$\underline{1}$	$\underline{1}$
0	1	D	0	1
1	0	0	D	1
0	0	1	1	1
0	0	0	D	0
0	0	1	$\underline{1}$	0
0	0	$\underline{1}$	1	0
0	0	1	$\underline{1}$	0
1	$\underline{1}$	0	0	0
$\underline{1}$	1	0	0	0
$\underline{1}$	$\underline{1}$	0	0	0
1	1	0	0	0

The idea of the algorithm is to consider the CMSD representation of the two operands, A and B, as input patterns and derive the MSD representation of their summation as output pattern and using one symbolic substitution step. For each pair of digits, A_iA_{i-1}, B_iB_{i-1}, the resultant digits, S_i, are derived by a simple substitution and in a parallel carry-free manner, $\forall\, i = 0, \ldots, n$ where n is the digit-size of the operands.

3. NEW SYMBOLIC SUBSTITUTION BASED CMSD ADDITION

We present here a new algorithm that, although requiring more symbolic substitution steps than the previous one, derives the result in CMSD notation. If the given CMSD numbers are each of digit-size n, our algorithm requires $\lfloor n/2 \rfloor + 1$ symbolic substitution steps. The algorithm and the symbolic substitution tables have been presented in Tables 2–4 and Algorithm 3.1.

ALGORITHM 3.1. Algorithm for addition of CMSD numbers in $(\lfloor n/2 \rfloor + 1)$ steps.

`Let` A and B be two n-digit operands. Pad 2 '0' bits to the left and 1 '0' bit at the right of both operands.
`for` $j = 1$ to $\lfloor n/2 \rfloor + 1$ `do`
  `for` i=0 to $n - 1$ STEP 2 `pardo`
    `Set` $Y_{i+1}Y_i$ & $C_{i+3}C_{i+2}$ according to the symbolic substitution table on basis of the values of $A_{i+1}A_i$ & $B_{i+1}B_i$
  `Set` $A = Y$ & $B = C$
$Y = Y_n, Y_{n-1}, \ldots, Y_0$ is the resultant

Table 2. Symbolic substitution table for performing addition of CMSD numbers in $(\lfloor n/2 \rfloor + 1)$ steps.

$A_{i+1}A_i$	$B_{i+1}B_i$	Check: $A_{i-1}B_{i-1}$ Outputs $(Y_{i+1}Y_i, C_{i+3}C_{i+2})$		
		11	$\underline{11}$	Otherwise
00	00	01, 00	0$\underline{1}$, 00	00, 00
	01	–	–	01, 00
	0$\underline{1}$	–	–	0$\underline{1}$, 00
	10	0$\underline{1}$, 01	01, 00	CheckCond.1
	$\underline{1}$0	0$\underline{1}$, 00	01, 0$\underline{1}$	CheckCond.2
01	00	–	–	01, 00
	01	–	–	CheckCond.1
	0$\underline{1}$	–	–	00, 00
	10	–	–	0$\underline{1}$, 01
	$\underline{1}$0	–	–	0$\underline{1}$, 00

Note: CheckCond.1 and CheckCond.2 imply further conditions as have been illustrated in Tables 5 and 6.

Table 3. Symbolic substitution table for performing addition of CMSD numbers in $(\lfloor n/2 \rfloor + 1)$ steps: continued.

$A_{i+1}A_i$	$B_{i+1}B_i$	Check: $A_{i-1}B_{i-1}$ Outputs $(Y_{i+1}Y_i, C_{i+3}C_{i+2})$		
		11	$\underline{11}$	Otherwise
0$\underline{1}$	00	–	–	0$\underline{1}$, 00
	01	–	–	00, 00
	0$\underline{1}$	–	–	CheckCond.2
	10	–	–	01, 00
	$\underline{1}$0	–	–	01, 0$\underline{1}$
10	00	$\underline{1}$, 01	01, 00	CheckCond.1
	01	–	–	0$\underline{1}$, 01
	0$\underline{1}$	–	–	01, 00
	10	01, 00	0$\underline{1}$, 00	00, 00
	$\underline{1}$0	01, 00	0$\underline{1}$, 00	00, 00

Note: CheckCond.1 and CheckCond.2 imply further conditions as have been illustrated in Tables 5 and 6.

At each substitution step, the proposed algorithm considers three digits from each of the input operands, $A_{i+1}A_iA_{i-1}$, $B_{i+1}B_iB_{i-1}$ as input patterns, and using Tables 2–4, derives as output patterns the two variables Y and C (by determining the digits, $Y_{i+1}Y_i$ and $C_{i+3}C_{i+2}$, $\forall\, i = 0, \ldots, n-1$, where n is the digit-size of the operands) in parallel. In some cases, the

Table 4. Symbolic substitution table for performing addition of CMSD numbers in $(\lfloor n/2 \rfloor + 1)$ steps: continued.

$A_{i+1}A_i$	$B_{i+1}B_i$	Check: $A_{i-1}B_{i-1}$ Outputs $(Y_{i+1}Y_i, C_{i+3}C_{i+2})$		
		11	$\underline{11}$	Otherwise
$\underline{1}0$	00	0$\underline{1}$, 00	01, 0$\underline{1}$	CheckCond.2
	01	–	–	0$\underline{1}$, 00
	0$\underline{1}$	–	–	01, 0$\underline{1}$
	10	01, 00	0$\underline{1}$, 00	00, 00
	$\underline{1}0$	01, 00	0$\underline{1}$, 00	00, 00

Note: CheckCond.1 and CheckCond.2 imply further conditions as have been illustrated in Tables 5 and 6.

Table 5. CheckCond.1 for Tables 2–4.

A_{i+2}	B_{i+2}	Outputs $(Y_{i+1}Y_i, C_{i+3}C_{i+2})$
0	0	10, 00
1	$\underline{1}$	
1	1	
$\underline{1}$	1	
$\underline{1}$	$\underline{1}$	
otherwise	otherwise	$\underline{1}$0, 01

Table 6. CheckCond.2 for Tables 2–4.

A_{i+2}	B_{i+2}	Outputs $(Y_{i+1}Y_i, C_{i+3}C_{i+2})$
0	0	$\underline{1}$0, 00
1	$\underline{1}$	
1	1	
$\underline{1}$	1	
$\underline{1}$	$\underline{1}$	
otherwise	otherwise	10, 0$\underline{1}$

substitution process requires to check the digits, A_{i+2} and B_{i+2}, as well and Tables 5 and 6 are used according to the corresponding entry in Tables 2–4. After a step has been completed, the values of O and C are copied back as A and B for next substitution step. This substitution process is iterated $\lfloor n/2 \rfloor + 1$ times, at the end of which $Y = Y_n, Y_{n-1}, \ldots, Y_0$ holds the result of addition.

An example of the intermediate steps in the algorithm has been provided in Demonstration 3.1.

Demonstration 3.1.

$A = 0\ \underline{1}\ 0\ 1 = -3$

$B = 1\ 0\ \underline{1}\ 0 = 6$

==========

Padding two 0 at left and one zero at right of both A & B

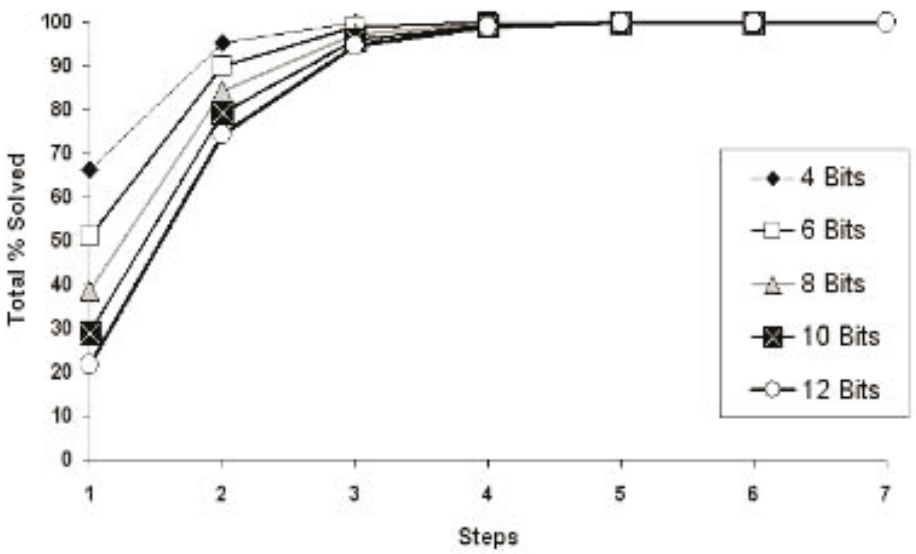

Figure 2. Graph of total of the percentage solved vs. required no. of steps.

A: 0 0 0 $\underline{1}$ 0 1 0
B: 0 0 1 0 $\underline{1}$ 0 0
==========
Y: 0 0 0 1 0 $\underline{1}$ 0 [STEP 1]
C: 0 0 0 0 0 0 0
==========
Y: 0 0 0 1 0 $\underline{1}$ 0 [STEP 2]
C: 0 0 0 0 0 0 0
==========
Y: 0 0 0 1 0 $\underline{1}$ 0 [STEP 3]
C: 0 0 0 0 0 0 0
==========
0 0 1 0 $\underline{1}$ = 3 [Final output]

4. FURTHER OPTIMIZATION

It may be noted that once the second input variable B reaches the value of all zero, the algorithm just repeats the input pattern into output pattern, and no effective substitution is made. Thus, the presented algorithm can be further improved by stopping the substitution process as soon as the auxiliary output variable C reaches the value of all 0. To demonstrate the optimization, a computer program was developed to simulate the addition process. For a given digit-size, n, the program could generate all the possible combinations of CMSD numbers (in total: $(2^{n+2} + (-1)^{n \bmod 2+1}/3))^2$ CMSD numbers) for that particular digit-size. For each of these combinations of CMSD numbers, the number of parallel steps, that are necessary to perform the addition, was recorded. From these data, the cumulative percentage of the total possible combinations that derived the addition result within a particular number of steps was calculated. The result of the analysis is shown in Figure 2. It can be seen that most of the combinations (more than 90%) require steps much fewer than the specified maximum of $\lfloor n/2 \rfloor + 1$ parallel processing steps. Hence for any combination of CMSD numbers generated at random, the algorithm with optimization will require much fewer substitution steps.

5. COMPARISON OF THE PROPOSED AND EXISTING ALGORITHM

The existing algorithms [2,3,5–7] can be implemented in optical symbolic substitution structure to perform the addition of MSD and CMSD numbers in one, two, or three steps. In contrast, the proposed algorithm requires more symbolic substitution steps than the existing one. However, the advantage of the proposed algorithm is that the output of this algorithm is in CMSD form, while the existing algorithm gives output in MSD form. Thus when applied in the context of associative addition of a set of CMSD numbers, the new algorithm gives better performance than the earlier ones.

We present here a comparison of the fastest earlier algorithm (the one-step CMSD addition) and our algorithm in the context of associative CMSD addition. Since the earlier one-step algorithm derives the addition result in MSD form, for associative addition the result has to be converted to CMSD notation before applying the addition algorithm again. Reitwiesner's algorithm [14], the most cited MSD to CMSD conversion algorithm [4,15,16], requires n symbolic substitution steps, where n is the digit-size of the MSD number to be converted. Since the addition of two n-digit CMSD numbers results in an $(n+1)$-digit MSD number, the earlier one step algorithm will require in total, $n+2$ symbolic substitution steps for each pair of numbers in the context of the associative CMSD addition (1 for addition and $n+1$ for conversion). Hence, if there are m CMSD numbers to be added, a symbolic substitution scheme based on the earlier algorithm will require about $(n+2)*(m-1)$ symbolic substitution steps. Our proposed algorithm, however, requires only about $(\lfloor n/2 \rfloor + 1)*(m-1)$ symbolic substitution steps for the same application.

6. CONCLUSION

Our proposed algorithm, therefore, shows significant superiority in terms of number of symbolic substitution steps in the context of associative addition of CMSD numbers. An application in which the process may be useful is the multiplication of two CMSD numbers where associative addition is an integral operation.

REFERENCES

1. J.U. Ahmed and A.A.S. Awwal, Polarization-encoded optical shadow-casting arithmetic-logic-unit design: Separate and simultaneous output generation, *Applied Optics* **31**, 5622–5631, (1992).
2. M.S. Alam, A.A.S. Awwal and A.K. Cherry, Opto-electronic symbolic substitution based canonical modified signed-digit arithmetic, *Optics and Laser Technology* **29**, 151–157, (1997).
3. A.A.S. Awwal and M.A. Karim, *Optical Computing: An Introduction*, John Wiley & Sons, New York, (1992).
4. K. Hwang and A. Louri, Optical arithmetic using signed-digit symbolic substitution, In *International Conference on Parallel Processing*, Vol. 1: Architecture (ICPP '88), August 1988, Pennsylvania, pp. 55–64.
5. S. Cotofana and S. Vassiliadis, Signed digit addition and related operations with threshold logic, *IEEE Transactions on Computers* **49** (3), 193–207, (March 2000).
6. R.S. Fyath, A.A.W. Alsaffar and M.S. Alam, Optical two-step modified signed-digit addition based on binary logic gates, *Optics Communications* **208**, 263–273, (2002).
7. A.K. Cherri and M. Hamad, Algorithms for optoelectronics implementation of trigonometric functions based on modified signed-digit numbers, In *The 14^{th} International Conference on Microelectronics*, December 2002, Beirut, Lebanon, pp. 109–113.
8. A. Louri, A symbolic substitution based parallel architecture and algorithms for high-speed parallel processing, In *Proc. of the 1990 ACM Annual Conference on Computer Science*, Washington, DC, pp. 173–179, (February 1990).
9. A. Avizienis, Signed-digit number representations for fast parallel arithmetic, *IRE Transactions on Electronic Computers* **10**, 389–400, (1961).
10. B. Parhami, Generalized signed-digit number systems: A unifying framework for redundant number representations, *IEEE Transactions on Computers* **39** (1), 89–98, (1990).
11. B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*, Oxford University Press, (2001).
12. S. Shieh and C. Wu, Asymmetric high-radix signed-digit number systems for carry-free addition, *Journal of Information Science and Engineering* **19** (6), 1015–1039, (2003).

13. D.S. Phatak and I. Koren, Hybrid signed-digit number systems: A unified framework for redundant number representations with bounded carry propagation chains, *IEEE Transactions on Computers* **43** (8), 880–891, (August 1994).
14. G.W. Reitwiesner, Binary arithmetic, *Advances in Computers* **1**, 231–308, (1960).
15. Ç.K. Koç, Parallel canonical recoding, *Electronics Letters* **32** (22), 2063–2065, (October 1996).
16. Ö. Eğecioğlu and Ç.K. Koç, Exponentiation using canonical recoding, *Theoretical Computer Science* **129** (2), 407–417, (1994).