SQUARE-ROOT ALGORITHMS FOR HIGH-SPEED DIGITAL CIRCUITS

by S. Majerski

Instytut Maszyn Matematycznych
Warszawa, POLAND

Abstract

Two binary algorithms for the square rooting of a sum of two numbers are presented. They are designed for high-speed digital circuits and are based on the classical nonrestoring method. The main difference lies in the replacement of subtractions and additions by the parallel reduction of three summands to two, their sum being unchanged, to eliminate a carry propagation. The term "parallel reduction" is introduced here for the carry-save addition of three summands, positive and negative as well. The two result summands form a successive partial remainder. Their most significant three-bit groups are used to determine the "digits" -1,0,+1 of the square root in a redundant notation. These digits are transformed into the conventional-notation bits, which are used in the further steps of the square-rooting process.

Derivation of the algorithms

The classical nonrestoring method is described, among others, by Y.Chu [1] and I.Flores [2]. The high-speed square-rooting algorithms, based on this method, are devised among others, by G.Metze [3] and V.G. Oklobdžija, M.D.Ercegovac [4]. The Metze's binary algorithms give the square-root value in the notation with the digits -1, 0, +1, too. They are specially fitted to obtain the results in notations with the minimum possible number of the non-zero digits. The nonredundant notations with the digits -1,0,+1 are discussed there too. The algorithms described in [4], cover also nonbinary number notations and take into account a bigger number of various digits of the redundant notations of the square root. There are known many other methods of the square rooting, which are applied in digital circuits, such as digit by digit method of J.E,Meggit [5], the method of T.C.Chen [6] using the redundant notation, or the cellular array method of J.C.Majithia [7].

The classical binary method for extracting an n-bit square root $Y \geq 2^{n-1}$, from a 2n-bit integer X, for $2^{2(n-1)} \leq X < 2^{2n}$, is based on the iteration

$$X_i = X_{i-1} - \left(2Y_{i-1} + y_i 2^{n-i}\right) y_i 2^{n-i} = \qquad /1/$$
$$= X_{i-1} - Y_i^2 \quad (i = 1,2,\dots,n),$$

where

$$Y \leq \sqrt{X} < Y+1, \qquad /2/$$

the symbols are defined as follows:

X_i - the ith successive partial remainder, $X_0 = X$,
y_i - the ith successive "digit" of the square root Y,
Y_i - $y_k 2^{n-k}$ - the ith partially-developed square root, $Y_0 = 0$.

The additions and substractions, occurring in the classical nonrestoring method, being used in digital circuits, correspond to the assumption, that $y_i \in \{-1,+1\}$. The square root is however usually presented in the equivalent, conventional binary form, and its bits are determined on the basis of the signs "-" and "+" of the successive partial remainders /instead of the digits -1,+1, corresponding to successive additions and subtractions/.

The process of the iterations /1/ is convergent, when $y_i = +1$ for $X_{i-1} \geq 0$, $y_i = -1$ for $X_{i-1} < 0$ and, when

$$|X^*_{i-1}| \leq 2\left(2Y_{i-1} + y_i 2^{n-i}\right), \qquad /3/$$

where $X^*_{i-1} = X_{i-1} 2^{-n+i}$, is the "shifted" successive partial remainder.

In the algorithm given in this paper, the two-summand partial remainders are used, to eliminate the carry propagation. Following that, the choice of either $y_i = -1$ or $y_i = +1$ must be based on the group of the most significant bits of the successive partial remainder. If the remainder sign cannot be determined from such bits groups, $y_i = 0$ must be taken. On the other hand, /1/ and /3/ implicate, that y_i may be equal 0, if

$$|X^*_{i-1}| \leq 2Y_{i-1} - 2^{n-i}, \qquad /4/$$

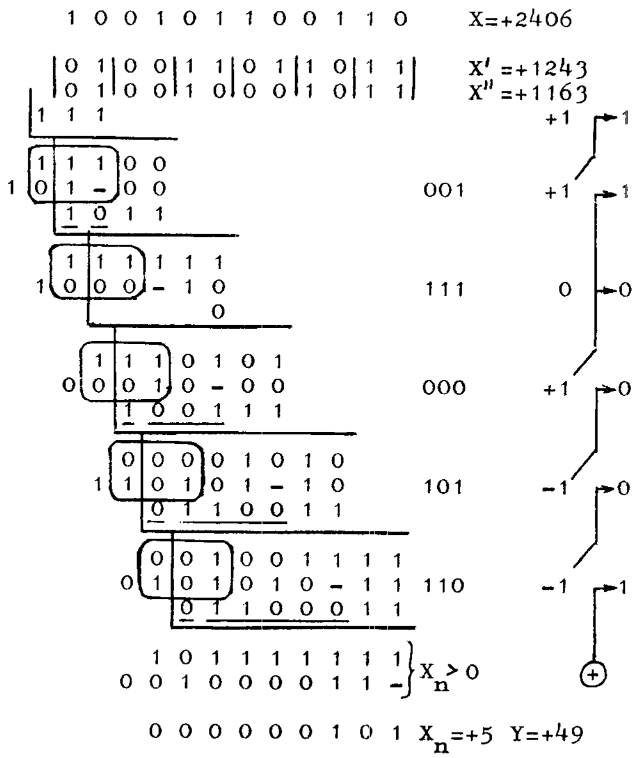
because the next partial remainder $X_i=X_{i-1}$ fulfills then /3/.

The analysis of /3/ and /4/ leads to the conclusion that for the determining of the digits $y_i \in \{-1,0,+1\}$, it is necessary and enough to examine the bits, taken from three binary positions of both summands of the shifted partial remainders. The weights of these examined positions are 2^n, 2^{n+1} and 2^{n+2}.

The first algorithm

The first algorithm, in which both partial-remainder summands are presented in the binary complementary notation, is explained in the numerical example 1. The square root Y of the sum X of two integers X' = 010011011011 (+1243) and X'' = 010010001011 (+1163) is there extracted.

The numerical example 1



1 0 0 1 0 1 1 0 0 1 1 0 X=+2406
X' =+1243
X'' =+1163
001
111
000
101
110
+1
+1
0
+1
-1
-1
X_n > 0
0 0 0 0 0 0 1 0 1 X_n=+5 Y=+49

The numbers X' and X'' are sliced into two-bit groups. In the first algorithm step, their most significant groups form two of three reduced summands. The third reduced summand is -2^{2n-2} /see /1//.

As a result of the parallel reduction of three summands, the two-summand partial remainder, in each algorithm step, is obtained. Such reduction can be executed in a row of one-position adders. The performance of such adder is described by

$$d+2e := a+b+c \qquad /5/$$

where a, b, c, are the bits taken from three reduced summands, and d, e, are the appropriate bits of two summands obtained after the reduction.

The most significant three-bit groups of both partial-remainder summands, encircled in the numerical example 1, determine the successive digit $y_i \in \{-1,0,+1\}$ of the square root Y. These three-bit groups, treated as three-bit numbers, are added modulo 8. The results/obtained in the complementary notation/ are shown on the right side of the example 1. They are transformed into digits -1,0,+1 according to the table

10x	-1	
110	-1	/6/
111	0	
0xx	+1	

where, the bits designed by the "x" are irrelevant for the transformation result. The respective digits -1,0,+1 of the square root Y, are shown further to the right of the numerical example 1.

The square-root bits are determined during the respective algorithm steps, in the way shown on the right side of the example 1. The digit +1 or -1 permits final determination of the square-root bits from all more significant binary positions than the position of this digit. When a digit +1 is obtained, the already determined digits y_i, beginning from the last examined digit ±1 or -1, without their signs, are taken as the bits of the square root. For a digit -1, the already determined digits, beginning from the last examined digit +1 or -1, without their signs, are taken and negated to get the square-root bits. The digit 0 does not permit to determine any square-root bits.

The transformation of the digits -1, 0,+1 into the square-root bits can be implemented on two shifted registers, which provide the parallel transfer between them.

Two summands, obtained as a result of the last parallel reduction, form the final remainder X_n. Its sign shows that, either the square-root value is correct /it is correct when $X_n \geq 0$/, or it is with an overflow /when $X_n < 0$/. Thus, the final-remainder sign has the same meaning for the determination of the square-root bits, as the digits -1 and +1 have /see right side of the numerical example 1/. When the final remainder is negative, the additional reduction step ought to be executed /if the correct remainder is required/.

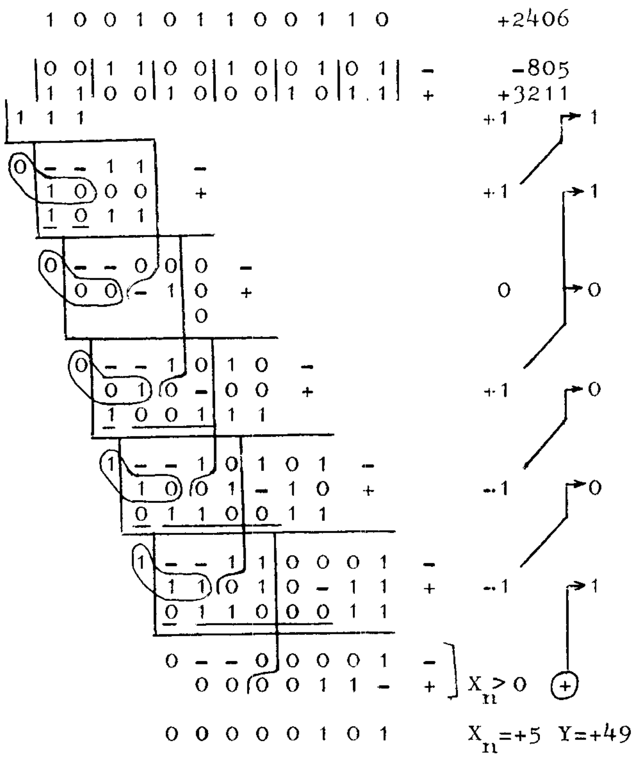
On the basis of the already known bits of the square root, the third of the reduced summands $-(2Y_{i-1}+y_i 2^{n-1})y_i 2^{n-i}$, in the complementary notation, is formed. For $y_i=-1$ it consists of two "ones" on the least significant binary positions, and of the already known square-root bits on further positions /they are underlined in the numerical example 1/. When $y_i=+1$, the above mentioned square-root bits, together with the sign bit, are additionally negated in the third summand /they are also underlined in the example/. For the digit $y_i=0$, the third summand is equal 0.

The square-rooting digital circuit, based on the described algorithm, has been patented in Poland [9].

The second algorithm

The second algorithm, in which the bit weights of the first partial-remainder summand are negative and of the second one are positive, is explained in the numerical example 2. The square root of a difference of two positive integers 0110010001011 /+3211/ and 0001100100101 /+805/ is there extracted.

The numerical example 2



The bits of the first of the three reduced summands have now the negative weights /it represents now the number −805/ and the bits of the second summand have positive weights. The third summand is, as in previous example, in the binary complementary form.

The parallel reduction of three summands to two, presented in the above mentioned notations, is now described by

$$-d+2e := -a+b+c \qquad /7/$$

where a, b, c, are appropriate bits taken from three reduced summands, and d, e, are appropriate bits of two summands obtained after the reduction /compare /5//. According to this formula the parallel reduction of three summands to two can be executed in a row of one-position adder-subtractors, the performance of which is described by the truth table

a	b	c	d	e
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

/8/

This table shows that the structure of the mentioned one-position adder-subtractor is not more complex than that of the one-position adder. The only difference is such, that, for the obtaining the "out-put-carry" function "e", the negation of the argument "a", instead of the "a" ought to be used in the appropriate Boolean formula.

Four binary positions of the three reduced summands are located in the numerical example 2, between two vertical lines. For these parts of the reduced summands a special addition /executed in a special addition circuit/, instead of the parallel reduction, is proposed, according to the schema

$$\begin{array}{cccc} - & - & y & y \\ x & x & x & x \\ \underline{y} & \underline{x} & \underline{x} & \underline{x} \\ y & - & - & y \\ & x & x & \end{array} \qquad /9/$$

where the bits with the positive weights are denoted by the "x", and with the negative ones, by the "y" /further more significant result bits are not taken into account here/.

The mentioned special addition ought to be executed in the special high-speed addition circuit, as it effectively determines the square-rooting speed.

The three most significant result bits yxx /they are encircled in the schema /9/ and in the numerical example 2/ determine the successive square-root digits −1, 0, +1, depending, whether they represent /in the binary complementary notation/ the negative number, zero, or positive number. The decoding is executed according to the table

1xx	-1	
000	0	
001	+1	/10/
01x	+1	

The appropriate square-root digits -1,0,+1 are shown on the right side of the numerical example 2.

The transformation of the digits -1, 0,+1 into the square-root bits is executed in the same way as in the previous algorithm. It is also shown on the right side of the numerical example 2. Also in the same way as previously the third summand is formed, for the parallel reduction in the successive algorithm step, on the basis of the already known square-root bits.

The parallel reduction need not be performed on the most significant bits, they are cut off in the numerical example 2 by the left vertical lines. The cutting bits do not influence on the further square-rooting process.

The patent of the square-rooting digital circuit,based on the described algorithm, has been claimed to Patent Office in Poland in 1981 [10].

Conclusion

The computer arithmetic algorithms of the complex operations are usually based on the sequence of the additions and subtractions, the execution time of which in the parallel digital circuits depends mainly upon the carry-propagation time. This propagation can be eliminated when redundant number notations are applied. It however, requires more bits for the number notations and leads to increased memory capacity which often determines the cost of the computer systems.

The solution of this problem, especially for very high-speed computer systems for the numerical computations, the author sees in the replacement of additions and subtractions, being the microoperations of complex operations, by the parallel reduction of the number of the summands with their sum being unchanged.

This paper presents examples of such solutions for the extracting of the square root.

Similar methods, using of the parallel reduction of three, four and more summands, can be applied for the division, for some elementary functions, as the exponent and logarithm functions and also for computation of such arithmetic expressions as scalar products, polynomials and function series. The appropriate digital circuits and systems have been designed and some of them are patented.

References

[1] Y.Chu, "Digital computer design fundamentals", McGraw-Hill, 1962.

[2] I.Flores, "The logic of computer arithmetic", Prentice-Hall, 1963.

[3] G.Metze, "Minimal square rooting", IEEE Trans.on Electr.Comp., vol.EC-14, pp.181-185, Apr.1965.

[4] V.G.Oklobdžija, M.D.Ercegovac, "An on-line square root algorithm", IEEE Trans.on Comp., vol.C-31, pp.70-75, Jan.1982.

[5] J.E.Meggit, "Pseudo division and pseudo multiplication processes", IBM Journ.of Res.and Devel., vol.6, No 2, pp.210-226, Apr.1962.

[6] T.C.Chen, "Automatic computation of exponentials, logarithms, ratios and square roots", IBM Journ.of Res.and Devel., vol.16, No 4,pp.380-388, July 72.

[7] J.C.Majithia, "Cellular array for extraction of squares and square roots of binary numbers", IEEE Trans.on Comp., vol. C-21, pp.1023-1024, 1972.

[8] A.Avizienis, "Signed-digit number representation for fast parallel arithmetic", IRE Trans.on Electr.Comp., vol.EC-10, pp.389-400, Sept.1961.

[9] S.Majerski, W.Majerski, "Układ cyfrowy do obliczania pierwiastka kwadratowego w zapisie binarnym" /Digital circuit for the extraction of the square root in a binary notation/, Patent No 109470, Poland, filed:July 29, 1977.

[10] S.Majerski, "Układ cyfrowy do obliczania pierwiastka kwadratowego w zapisie binarnym" /Digital circuit for the square root in binary notation/, additionally claimed to patent, Poland, regist.No P-231727, June 17, 1981.