Complement Representations in the Fibonacci Computer

P. Ligomenides and R. Newcomb

Intelligent Machines Program and Electrical Engineering Department
University of Maryland
College Park, MD.20742

Abstract

Two complement representations and a sign-magnitude one are introduced which allow for handling negative numbers using only binary coefficients in Fibonacci base expansions. These are developed for practical implementation in Fibonacci computers.

I. Introduction

Because of the desire to have fault tolerant computers, there is an interest in pursuing various types of computers. Among those of interest are computers which are based upon other than binary number systems and among these of especial interest is the Fibonacci computer[1,2]. The Fibonacci computer uses the Fibonacci numbers[3] as the number base in which calculations and operations are performed with the fault tolerance coming from the redundancy present in this base. Thus, the Fibonacci numbers form a complete set in themselves or when any one of the Fibonacci numbers is deleted, a property not present in the base two system[4]. In this context completeness means that any nonnegative integer can be expressed in terms of the base numbers with appropriate coefficients. These coefficients in the Fibonacci base can be binary numbers [3, Chap.12], in which case we call the representation the binary Fibonacci representation, BFR. In the BFR several possible representations are available for any given number, this leading to the desired redundancy. For example the so called minimum representation in which no two adjacent coefficients are 1 and the so called maximum representation where as many 1's as possible can be cited (though in some cases the minimum and the maximum representations are identical).

The thesis of Hoang[2] treats various aspects of arithmetic operations, including the basic ones, as addition, while in previous work it has been shown how the minimum and maximum representations can practically be generated[5] and how other than binary coefficients can be used through ternary and quaternary logic[6]. Here we return to the binary coefficient case and present complement and sign magnitude systems such that negative numbers can be handled in the Fibonacci computer without going beyond binary coefficients and without using negatively indexed Fibonacci numbers in the base system. The details are presented in the next section. Following the next section we give some discussion on the results including comments on the consistency of the derived representations and their use in physical implementations.

II. BFR and Conversion Algorithms

We take as given a positive number M of the special form

$$M = \sum_{j=2}^{m} F_j \tag{1a}$$

where F_j is the jth Fibonacci number, that is

$$F_j = F_{j-1} + F_{j-2}, \quad F_0 = 0 \text{ and } F_1 = 1. \tag{1b}$$

This M represents the maximum number which can be held in an (m-1)-bit Fibonacci computer, and, thus, M+1 serves as a modulus for a modular number system.

For our purposes we take the complement of a nonnegative number N with respect to the positive number A to be A - N. Of special interest to us are two choices, those for which A = M, giving the M's complement, and for which A = M + 1, giving the (M+1)'s complement, keeping in concert with the reduced radix (RRC) and radix (RC) complements of standard binary computers.

II.1. Magnitude Conversion Algorithm - A given number $\pm N$ in sign-magnitude (SM) form can be converted into a binary Fibonacci Representation (BFR) either in SM or in "M Complemented" or in "(M + 1) Complemented" forms, in the range (-M,M) defined by the m-bit Fibonacci representation which is based on the completeness of the representation[3].

$$N = \sum_{j=2}^{m} b_j F_j \tag{2}$$

where M is as in (1a) above. It should be noted that the RRC form in this case refers to the M's complement number system, while the RC form refers to the (M+1)'s complement number system.

The conversion algorithm shown in the flowchart of Figure 1 will convert the magnitude N into the BFR designated by equation (2). The algorithm may be implemented if the number $\pm N$ is provided in some physical form, such as an analog voltage (e.g., the output of a sensor device) or a binary number (e.g., the output of present analog-to-

CH1630-3/81/0000/0006$00.75 © 1981 IEEE

digital converters). The required components are voltage comparators and common binary arithmetic circuits. The Fibonacci numbers F_m, $F_{m-1}, \ldots, F_2$, must also be provided, either through a memory look-up table or by some "Fibonacci number generator" circuit, like the one discussed in Section II.5 of this paper.

It is of interest to observe that the N-to-BFR conversion algorithm shown in Figure 1 does provide the minimum form of the BFR, as is seen by tracing through the flowchart. That is, the algorithm produces naturally the minimum number of 1-bits in the resulting binary code.

II.2. Sign-Magnitude Representation - Having obtained the BFR for N the SM Fibonacci representation of $\pm N$ is obtained by simply appending a sign bit, b_s, in the most significant bit (MSB) position. Thus, the SM-BFR

$$\pm N \longrightarrow (b_s b_m \ldots b_2) \qquad (3)$$

is accomplished, where we designate $(b_s b_n \ldots b_2)$ as the code of the given number.

II.3. M Complemented Representation - One may observe that the (m-1)-bit Fibonacci representation (obtained through execution of the algorithm in Figure 1) of M - N is the 1's complement of the coefficients in the (m-1)-bit BFR of N. This is so because the BFR of M has a string of 1-bits, i.e.

$$M = \sum_{j=2}^{m} F_j \longrightarrow (11\ldots1) \qquad (4a)$$

so that

$$N \longrightarrow (b_m \ldots b_2) \qquad (4b)$$

$$M - N \longrightarrow (b'_m \ldots b'_2) \qquad (4c)$$

where

$$b_i \oplus b'_i = 1, \quad i = 2, \ldots m \qquad (4d)$$

such that

$$(b_m \ldots b_2) \oplus (b'_m \ldots b'_2) = (11\ldots1) \qquad (4e)$$

Because of this relation holding in the Fibonacci representation of N and its M's complement, M - N, we may use the common techniques available for deriving and manipulating the RRC in base-2 representations and arithmetic operations.

The algorithm for obtaining the M's complemented BFR representation of a given number $\pm N$ is:

1. Form the (m-1)-bit code for N, i.e. $(b_m \ldots b_2)$, using the algorithm of Figure 1.
2. Append a zero sign bit in the MSB position, i.e. $(0b_m \ldots b_2)$.
3. If the given number is positive, i.e. +N, the M's complement code is ready to be stored, i.e. $(b_s b_m \ldots b_2)$ where $b_s=0$.
4. If the given number is negative, i.e. -N, the augmented binary code $(0b_m \ldots b_2)$ is complemented and stored as $(1b'_m \ldots b'_2)$. The negative number is now recognized by a 1-bit in the sign bit position.

It is of interest to note that the minimum forms of BFR, i.e. those with the minimum number of 1-bits, may reduce the number of carry operations in addition procedures, and, therefore, they allow for faster execution of arithmetic operations in the binary Fibonacci number system.

In order to secure the minimum form in the M's complemented BFR step 4 of the above algorithm is modified as follows:

4'. If the given number is negative, i.e. -N, the binary code $(b_m \ldots b_2)$ is first maximized (i.e. placed in maximum form by exchanging 100 with 011 combinations[5]), a zero is then appended in the MSB position and the resulting code is complemented and stored. The negative number is now recognized by a 1-bit in the sign bit position and the complemented code is in minimum form.

One must observe that in order to maintain the minimum forms during the execution of arithmetic operations in the BFR using this algorithm complementation should be preceded by maximization of the form. Thus, if the subtraction $N_1 - N_2$ is called for, the addition $N_1 + (M - N_2)$ is performed instead. If the minimum forms of the operands are to be maintained, the BFR code for N_2 is recalled, maximized, and then complemented before it is used in the above addition. Note that the preservation of minimum forms in storage and arithmetic operations in BFR is not essential but only used to simplify and speed up the arithmetic operations.

As an example of application of the conversion algorithm consider $-N = -31$ which is to be converted into the 8-bit M = 53 complemented BFR.

a) Conversion of 31 to minimum-BFR:
Since 31-53 < 0 the number N = 31 is "within range". We have

$M=53=F_8+F_7+F_6+F_5+F_4+F_3+F_2 = 21+13+8+5+3+2+1$.

F_8	$31 - 21 > 0$	$b_8=1$	using the algorithm of Figure 1
F_7	$10 - 13 < 0$	$b_7=0$	
F_6	$10 - 8 > 0$	$b_6=1$	
F_5	$2 - 5 < 0$	$b_5=0$	
F_4	$2 - 3 < 0$	$b_4=0$	
F_3	$2 - 2 = 0$	$b_3=1$	
F_2	$0 - 1 < 0$	$b_2=0$	

b) Codes for -31:
The above process has given the 7 digit minimum BFR code for 31

$$31 \longrightarrow (1010010)_{min}$$

A sign bit is added to give an 8 digit code for +31 which can be maximized as per step 4'

$$+31 \longrightarrow (01010010)_{min} \longrightarrow (00111110)_{max}$$

$$-31 \longrightarrow (10101101)_{max} \longrightarrow (11000001)_{min}$$

It is to be noted that in forming a BFR for -31 one for +31 is obtained. It is also noted that in the M's complement system there is a +0 (the all 0 code) and a -0 (the all 1 code).

II.4. (M+1) Complemented Representation - The (M+1)'s complement representation is obtained by adding one to the M's complement representation when the number is negative or leaving it as in the M's complement form if positive. In this system +0 = -0 is the all 0 code while the all 1 code represents -1.

II.5. The Fibonacci Number Generator - For the implementation of the conversion algorithm of Figure 1 it is required that the sequence of Fibonacci numbers, F_m, $F_{m-1},\ldots,F_2$, be available. The algorithm may be programmed on a digital computer or implemented in an autonomously operating circuit, with the required Fibonacci number sequence stored before-hand in a table in memory. A generator which will produce the sequence of Fibonacci numbers in real time is also possible.

The solution to the Fibonacci number generating linear difference equation, equation (1b) above, may be obtained in closed form by use of the Z-transform as [7,p.62].

$$F_j = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^j - \left(\frac{1-\sqrt{5}}{2}\right)^j\right] \qquad (5a)$$

$$= \frac{1}{\sqrt{5}}\left[a^j - b^j\right] \qquad (5b)$$

where a and b are the two roots of $x^2 - x - 1 = 0$. This solution may be programmed to yield a routine, providing on demand in real time the required Fibonacci numbers, and it may be incorporated in the program which implements the conversion algorithm of Figure 1.

Alternatively, a hardware implementation of a Fibonacci number generator is also possible by directly implementing the relation (1b), the circuit being given in [7,p.62]. The circuit consists of two unit delays and a summer, and it is clocked to provide the sequence $F_2,F_3,\ldots,F_m$, given the initial values F_0 and F_1. But for the algorithm of Figure 1 the reverse sequence, F_m, $F_{m-1},\ldots,F_2$, is required. This can be implemented from the linear difference equation

$$F_{m-j} = F_{m-j+2} - F_{m-j+1}\ , \ j = 0,\ 1,\ \ldots,\ m. \qquad (6)$$

through the circuit shown in Figure 2. Equation (6) and the circuit of Figure 2 are supplied with initial conditions F_{m+2} and F_{m+1}.

III. Discussion

In the above we have presented representations which allow negative numbers to be conveniently represented in the Fibonacci computer using only binary coefficients and positively indexed Fibonacci numbers in the base. Indeed we have begun the base with F_2 though the whole theory carries through when starting with F_0 or F_1 (where starting with F_1 is convenient for error correction while starting with F_2 gives a more "efficient system"; we used F_2 to illustrate the generality).

Conversion to Fibonacci representations from decimal is relatively straightforward and simply implemented as clearly illustrated in Figure 1 for the BFR where only simple subtractions and comparisons are required. The addition of numbers in BFR is also relatively easy to perform and a treatment can be found in Hoang [2,Chap.3]. Likewise arithmetic operations may be performed consistently in the Fibonacci representations derived in the previous section. That is, the addition, subtraction, multiplication, or division, of two numbers represented in any one of the previously derived representations must result in a number also conforming to the representation chosen. Work in this direction is continuing. It should also be noted that complete equivalent ternary or quaternary Fibonacci representations in sign magnitude or complement form can be directly derived from the algorithms discussed in Section II. The ternary and quaternary representations of a positive number N are derived directly from the BFR, as discussed in our previous paper [6].

Complete representations in the range (-M,M) may be accomplished in various ways. The extension to negative even subscripted Fibonacci numbers used as weights in the ternary and quaternary Fibonacci representations is one way which follows from the discussion in our previous work[6]. The complement representations are directly derivable following a similar algorithm to the one discussed in Section II. The ternary and quaternary complement representations, together with a complete set of rules for consistent arithmetic operations in these number systems, will be discussed in forthcoming works.

References

[1] R. Newcomb, "Fibonacci Numbers as a Computer Base", Conference Proceedings of the Second Interamerican Conference on Systems and Informatics, Mexico City, November 27, 1974.

[2] V.-C. Hoang, "A Class of Arithmetic Burst-Error-Correcting Codes for the Fibonacci Computer", Ph.D. Dissertation, University of Maryland, Department of Electrical Engineering, December 1979.

[3] V. E. Hoggatt, Jr., "Fibonacci and Lucas Numbers", Houghton Mifflin Company, Boston, 1969.

[4] J. L. Brown, Jr., "Note on Complete Sequences of Integers", The American Mathematical Monthly, Vol. 68, No. 6, June-July 1961, pp. 557-560.

[5] P. Monteiro and R. W. Newcomb, "Minimal and Maximal Fibonacci Representations: Boolean Generation", The Fibonacci Quarterly, Vol. 14, No. 1, February 1976, pp. 9-12.

[6] P. Ligomenides and R. Newcomb, "Equivalence of Some Binary, Ternary and Quaternary Fibonacci Computers", Proc. 11th Int'l Symp. on Multiple-Valued Logic, May 27-29, 1981.

[7] J. A. Cadzow, "Discrete-Time Systems", Prentice-Hall, Englewood Cliffs, NJ, 1973.

Given N:

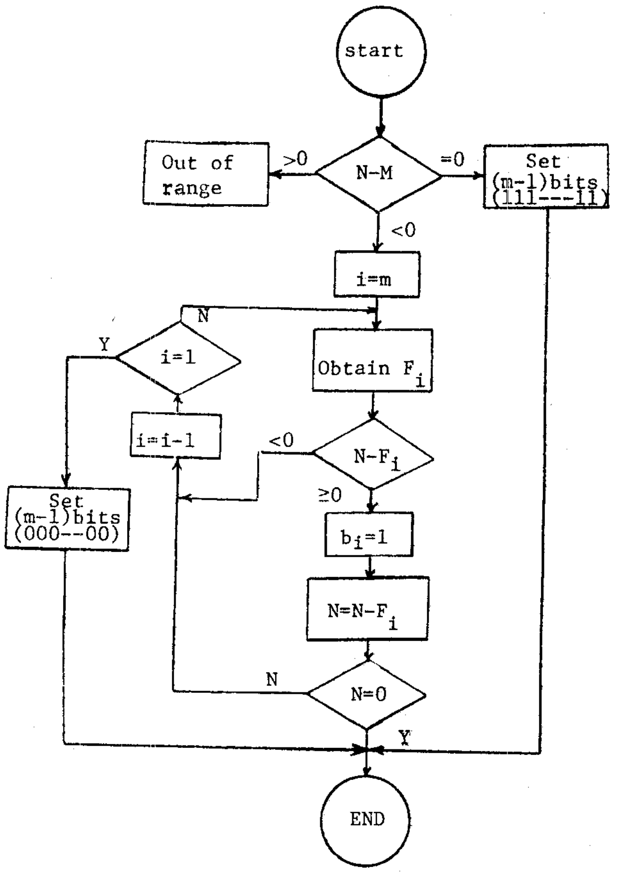

Figure 1
Flowchart for Algorithm
to Obtain BFR.

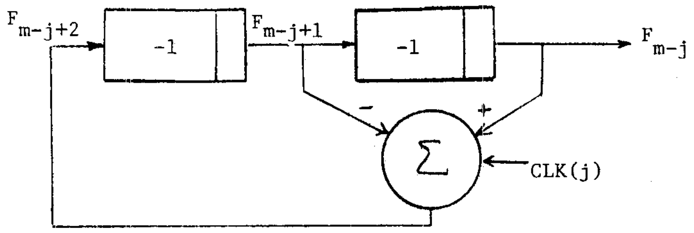

Figure 2
Reverse Sequence Fibonacci
Number Generator, F_m, F_{m-1}, ..., F_0