# MULTILEVEL FIBONACCI CONVERSION AND ADDITION

P. LIGOMENIDES and R. NEWCOMB
*University of Maryland, College Park, MD 20742*
*(Submitted March 1982)*

## I. INTRODUCTION

Recently we have been making studies [1, 2] on the Fibonacci number system that appears to be of considerable importance to the Fibonacci computer [3]. A specific result that appears to be of particular interest is that through multilevel coefficients on a Fibonacci radix system, efficient extension of representations can occur. For example, through a ternary coefficient system, a doubling of the range with half the number of digits over a binary system has been shown, while in fact allowing a restriction to only even-subscripted Fibonacci numbers in the radix [1]. Consequently, further investigation of various other properties and extensions has seemed warranted. This has led us to our present studies which indicate that the Fibonacci computer may have added features, over those of redundancy for error detection and correction already reported [4]. One such added feature lies in efficient processing techniques with these being based upon the conversion of numbers into special forms, including even- and odd-subscripted Fibonacci radix systems. With this in mind, we develop in this paper several new Fibonacci number representations, in particular even- and odd-subscripted ones, a signed ternary one, and conversions between them. These ideas are developed in Sections II and III. In particular, we give the details of conversions among the various ternary Fibonacci radix representations.

For reference purposes, we recall the defining recursion relation of the Fibonacci numbers [5]

$$F_i = F_{i-1} + F_{i-2},\ F_0 = 0,\ F_1 = 1. \tag{1.1}$$

At times we will use this expressed in the alternate form $F_i = F_{i+1} - F_{i-1}$. If $M = F_{m+2} - 2$, then any nonnegative integer $N$, $0 \leq N \leq M$, can be expanded in a Fibonacci representation using $(m - 1)$ binary coefficients on the Fibonacci numbers as a complete base or radix set. Previously, we have shown how these expansions can be made in terms of some ternary and quaternary coefficient sets [1]; we will extend these latter expansions here as needed for conversions and arithmetic operations.

## II. BINARY TO MULTILEVEL CONVERSIONS

Let an Unsigned Binary Fibonacci Representation, UBFR, be

$$N = \sum_{j=2}^{m} b_j F_j,\ b_j = \{0, 1\} \tag{2.1}$$

in the range $(0, M)$, where

$$M = \sum_{j=2}^{m} F_j = F_{m+2} - 2.$$

Such a representation can be derived for a given number $N$ using the conversion algorithm previously presented [2].

An Unsigned Quaternary Fibonacci Representation using only even-subscripted Fibonacci numbers, $UQFR_e$, is derived by the following formula [1]:

$$q_{2i}^e = b_{2i-1} + b_{2i} - b_{2i+1} \tag{2.2}$$

where

$$q_{2i}^e \in \{-1, 0, 1, 2\}, \; i = 1, 2, \ldots, k,$$

and

$$N = \sum_{j=2}^{m} b_j F_j = \sum_{i=1}^{k} q_{2i}^e F_{2i}. \tag{2.3}$$

The even-subscripted coefficients $q_{2i}^e$ take four possible values according to the conversion Table 2.1. As is known, see [1], several Unsigned Ternary Fibonacci Representations, UTFR, exist. Two of these that use only even-subscripted Fibonacci numbers, called $UTFR_e\{-1, 0, 1\}$ and $UTFR_e\{0, 1, 2\}$ may be derived from Table 2.1 by applying (1.1) on a UBFR, and thus eliminating either case 6 or case 1 by a prior conversion of the UBFR into the "minimum," UBFR(min), or the "maximum," UBFR(max), form respectively [6]. Of these two ternary representations, the one derived from the UBFR(min) is of particular interest because it allows positive-to-negative number conversion by a simple form of complementation, namely $-1 \leftrightarrow 1$ and $0 \leftrightarrow 0$. In this Signed Ternary Fibonacci Representation, $STFR_e\{-1, 0, 1\}$, the sign of the number is determined by the sign of the most significant nonzero coefficient [5, p. 56]. In the above, we note that the binary representation of (2.1) holds only for nonnegative numbers; hence, we have called it "unsigned," as well as those representations coming from it by the quaternary transformation (2.2). However, upon making the complementation just mentioned within a UTFR $\{-1, 0, 1\}$, negative numbers are contained within the system, which we consequently have called "signed" specifically to point out its broader nature.

TABLE 2.1

Binary-to-Quaternary Coefficient Conversion

| Case | $b_{2i-1}$ | $b_{2i}$ | $b_{2i+1}$ | $q_{2i}^e = b_{2i-1} + b_{2i} - b_{2i+1}$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | -1 |
| 2 | 0 | 1 | 0 | 1 |
| 3 | 0 | 1 | 1 | 0 |
| 4 | 1 | 0 | 0 | 1 |
| 5 | 1 | 0 | 1 | 0 |
| 6 | 1 | 1 | 0 | 2 |
| 7 | 1 | 1 | 1 | 1 |

Thus, the UBFR(min)-to-$STFR_e\{-1, 0, 1\}$ conversion is simply effected by

$$t_{2i}^e = b_{2i-1} + b_{2i} - b_{2i+1} \tag{2.4}$$

in the relation

$$N = \sum_{j=2}^{m} b_j F_j = \sum_{i=1}^{k} t_{2i}^e F_{2i}, \tag{2.5}$$

where $k = [(m+1)/2]$ is the integer portion of $(m+1)/2$. The range now becomes $(-M, M)$, where $M = F_{m+2} - 2 = F_{2k+2} - 2$ when $m$ is even.

The complete odd-subscripted Fibonacci representations may also be derived easily from a UBFR, in a manner analogous to the one discussed above for even subscripts. Indeed, a UBFR-to-UQFR$_o\{-1, 0, 1, 2\}$, as also the UBFR(min)-to-UTFR$_o\{-1, 0, 1\}$ and UBFR(max)-to-UTFR$_o\{0, 1, 2\}$ conversions, are all effected simply by almost identical conversion formulas [subtract "one" from all $i$ subscripts in relations (2.2) through (2.5)]. Again notice that the complementation operation $-1 \leftrightarrow 1$ and $0 \leftrightarrow 0$ allows for the positive-negative conversion, and thus provides the STFR$_o\{-1, 0, 1\}$.

## III. TERNARY CONVERSIONS AND ADDITION

In this section, we will discuss techniques for conversion between several ternary representations, the ones of interest being the full-, STFR$_f\{-1, 0, 1\}$,

$$N = \sum_{j=1}^{n} t_j F_j \tag{3.1a}$$

the even-, STFR$_e\{-1, 0, 1\}$,

$$N = \sum_{i=1}^{k} t^e_{2i} F_{2i} \tag{3.1b}$$

and the odd-, STFR$_o\{-1, 0, 1\}$,

$$N = \sum_{i=1}^{k} t^o_{2i-1} F_{2i-1} \tag{3.1c}$$

subscripted representations. In all cases, the coefficients $t_i$, $t^e_i$, $t^o_i$ are in the set $\{-1, 0, 1\}$.

Before discussion of the actual conversions, we show their use in terms of addition.

### A. Ternary Addition

Addition becomes a very simple matter if we assume the availability of numbers in the full-, even-, and odd-subscripted ternary representations of (3.1). Thus, let two numbers $N_1$ and $N_2$ be given, one in the even- and the other in the odd-subscripted form:

$$N_1 = \sum_{i=1}^{k} t^e_{2i} F_{2i} \quad \text{and} \quad N_2 = \sum_{i=1}^{k} t^o_{2i-1} F_{2i-1};$$

then their sum has the full representation with

$$t_j = \begin{cases} t^e_j, & j = \text{even}, \\ t^o_j, & j = \text{odd}. \end{cases} \tag{3.2}$$

That is, addition (and with it subtraction, because of readily executable complementation) occurs through the interleaving of the digits of the numbers being summed. The process, as illustrated in Figure 1, is especially convenient for hardware implementations.

### B. Full- to Even-/Odd-Subscript Conversion

As seen by the addition technique just discussed, it is convenient to be able to convert numbers from a full-subscripted form to a form using only even

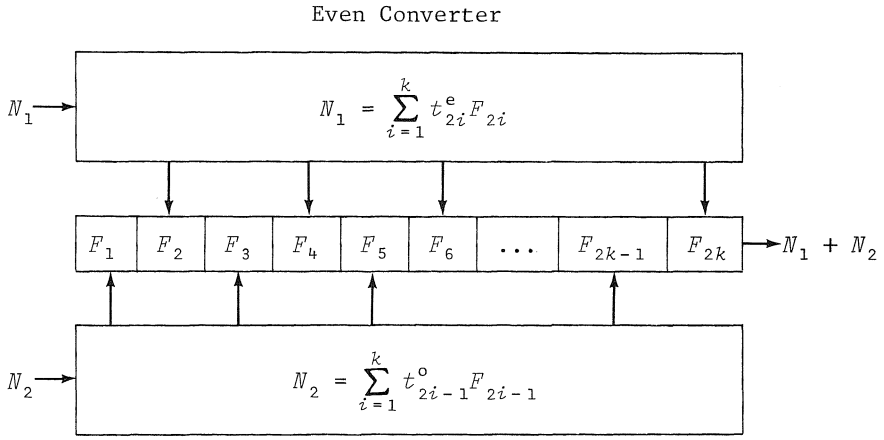


FIGURE 1. Ternary Addition

subscripts as well as to one using only odd subscripts. As the principles are identical for obtaining the odd-subscripted form, we will concentrate here on obtaining the even one.

Thus, let there be given a full-subscripted ternary representation. By substituting for odd-subscripted Fibonacci numbers

$$F_{2i-1} = F_{2i} - F_{2i-2} \qquad (*)$$

and writing $k = [(n+1)/2]$ for the integer part of $(n+1)/2$, we obtain:

$$N = \sum_{j=1}^{n} t_j F_j = \sum_{i=1}^{k} t_{2i} F_{2i} + \sum_{i=1}^{k} t_{2i-1}(F_{2i} - F_{2i-2}) \qquad (3.3a)$$

$$= \sum_{i=1}^{k} (t_{2i-1} + t_{2i} - t_{2i+1}) F_{2i}. \qquad (3.3b)$$

Here we again assume $t_j = 0$ for $j < 1$ and $j > n$. Thus, the coefficients

$$t^{e}_{2i} = t_{2i-1} + t_{2i} - t_{2i+1} \qquad (3.3c)$$

are those of an even-subscripted representation. However, if (3.3c) were to be applied directly to an arbitrary full-subscripted representation, it would lead to an even-subscripted representation with coefficients in the range of integers $\{\pm 3, \pm 2, \pm 1, 0\}$, i.e., a septenary rather than a ternary one. Table 3.1 shows these $3^3 = 27$ possibilities, where the fourth column gives the result of applying (3.3c) directly. As is seen in Table 3.1, there are eight possible out-of-code (i.e., nonternary) cases. In each out-of-code case, though, a preliminary preparation will bring the relevant coefficient back into code, the necessary preparation transformations being given in the final column of Table 3.1. That is, by making an appropriate substitution via a Fibonacci number identity in the original full-subscripted representation, an out-of-code converted coefficient can be brought back into code. Since some of the preparation transformations affect neighboring coefficients which could be brought out-of-code after transformation, it is necessary to continue the preparation until all coefficients become ternary when (3.3c) is finally applied. Since

TABLE 3.1

Full- to Even-Subscripted Conversion*
($t^e_{2i} = t_{2i-1} + t_{2i} - t_{2i+1}$)

| Case | $t_{2i-1}$ | $t_{2i}$ | $t_{2i+1}$ | $t^e_{2i}$ | Revert to Case | By |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | | Preliminary Preparation Transformations |
| 2 | 1 | 0 | 0 | 1 | | |
| 3 | -1 | 0 | 0 | -1 | | |
| 4 | 0 | 1 | 0 | 1 | | |
| 5 | 0 | -1 | 0 | -1 | | |
| 6 | 0 | 0 | 1 | -1 | | |
| 7 | 0 | 0 | -1 | 1 | | |
| 8 | 1 | -1 | 0 | 0 | | |
| 9 | -1 | 1 | 0 | 0 | | |
| 10 | 1 | 0 | 1 | 0 | | |
| 11 | -1 | 0 | -1 | 0 | | |
| 12 | 0 | 1 | 1 | 0 | | |
| 13 | 0 | -1 | -1 | 0 | | |
| 14 | 1 | 1 | 1 | 1 | | |
| 15 | 1 | -1 | 1 | -1 | | |
| 16 | 1 | -1 | -1 | 1 | | |
| 17 | -1 | 1 | 1 | -1 | | |
| 18 | -1 | -1 | -1 | 1 | | |
| 19 | -1 | 1 | -1 | 1 | | |
| 20 | 1 | 1 | 0 | 2 | 6 | $F_{2i-1} + F_{2i} = F_{2i+1}$ |
| 21 | -1 | -1 | 0 | -2 | 7 | $-F_{2i-1} - F_{2i} = -F_{2i+1}$ |
| 22 | 1 | 0 | -1 | 2 | 5 | $F_{2i-1} - F_{2i+1} = -F_{2i}$ |
| 23 | -1 | 0 | 1 | -2 | 4 | $-F_{2i-1} + F_{2i+1} = F_{2i}$ |
| 24 | 0 | 1 | -1 | 2 | 3 | $F_{2i} - F_{2i+1} = -F_{2i-1}$ |
| 25 | 0 | -1 | 1 | -2 | 2 | $F_{2i} + F_{2i+1} = F_{2i-1}$ |
| 26 | 1 | 1 | -1 | 3 | 1 | $F_{2i-1} + F_{2i} - F_{2i+1} = 0$ |
| 27 | -1 | -1 | 1 | -3 | 1 | $-F_{2i-1} - F_{2i} + F_{2i+1} = 0$ |

*For a full- to odd-subscripted conversion, all subscripts are shifted down by one in the table entries.

each preparatory transformation reduces by at least one the number of nonzero ternary coefficients in the full-subscripted representation being converted, successive applications of the preparatory transformations eventually will lead to a termination with only ternary $t^e_{2k}$ calculated according to (3.3c). It should be noted that preparatory transformations can be applied simultaneously to nonoverlapping strings of three consecutive coefficients, but that simultaneous application to overlapping strings should be avoided.

As an example, consider (see Table 3.1 for case numbers):

$$N = -3 = (F_1 - F_3) + F_4 - (F_5 - F_7) + F_8 - F_9, \quad \text{cases 22 \& 23} \tag{3.4a}$$

$$= -F_2 + F_4 + F_6 + (F_8 - F_9), \quad \text{case 24} \tag{3.4b}$$

$$= -F_2 + F_4 + (F_6 - F_7), \quad \text{case 24} \tag{3.4c}$$

$$= -F_2 + (F_4 - F_5), \quad \text{case 24} \tag{3.4d}$$

$$= -F_2 - F_3 \tag{3.4e}$$

$$= -F_4, \text{ by Eq. (1.1)} \tag{3.4f}$$

The ripple effect of the preparatory transformations is well-exhibited by this example.

It should be noted that none of the ternary representations of (3.1) need be unique, for example:

$$N = 4 = F_2 - F_3 + F_5 = F_2 + F_4 = -F_4 + F_6 = -F_1 + F_5 = F_1 - F_3 + F_5. \tag{3.5}$$

Conversion from full- to odd-subscripted representations uses identical techniques, the only difference being that one is subtracted from the indices in (3.3c).

## C. Even-/Odd-Subscript Conversion

The conversion of an odd- to an even-subscripted representation, or vice versa, is really a special case of the full- to even-subscripted, or odd-subscripted conversion. Thus, Table 3.1 applies. However, since the $t_{2k} = 0$ in the present case of Table 3.1, only $3^2 = 9$ of the cases occur with only two of these requiring preliminary preparation. This is illustrated in Table 3.2, where, also, in columns 2 and 6 we give the possible range of adjacent coefficients. In the two cases requiring preliminary preparation, we can actually carry out the adjustment after application of the conversion formula (3.3c) by using $3F_i = F_{i-2} + F_{i+2}$ [5, p. 59] for the replacement

$$2F_i = 3F_i - F_i = F_{i-2} - F_i + F_{i+2}. \tag{3.6}$$

It is seen in the three right-hand columns of Table 3.2 that adjacent coefficients are brought back into code. However, if two adjacent coefficients would become out-of-code by application of (3.3c), then (3.6) should only be applied to one of them so that the correction will remain in code. An example will illustrate this technique:

$$N = -24 = F_3 - F_5 + F_7 - F_9, \quad \text{given} \tag{3.7a}$$

$$= -F_2 + 2F_4 - 2F_6 + 2F_8 - F_{10}, \quad \text{by (*)} \tag{3.7b}$$

$$= -F_2 + (F_2 - F_4 + F_6) - 2F_6 + (F_6 - F_8 + F_{10}) - F_{10}, \text{ by (3.6)} \tag{3.7c}$$

$$= -F_4 - F_8. \tag{3.7d}$$

This last also results from one preliminary transformation on the given representation using Table 3.1.

In the case of even- to odd-subscripted conversions, or vice versa, it is seen that the "ripples" associated with the preliminary preparations can be avoided by one cycle of application of (3.6) after the use of (3.3c).

## IV. Discussion

Conversion between several Fibonacci number representations has been discussed here with special emphasis upon those representations which appear most useful for multivalued logic realizations of the Fibonacci computers. Of special interest along this line is the addition technique that is seen, via Figure 1, to be a simple matter of register loading when even- and odd-subscripted ternary representations are on hand. Besides this rather considerable advantage of the ternary Fibonacci representations, it is clear that they also

TABLE 3.2

Odd to Even Conversion Table
(for Even to Odd, interchange $t^e$ and $t^o$)

| Cases | $t^o_{2i-1} \rightarrow t^e_{2i} \leftarrow t^o_{2i+1}$ <br> $t^e_{2i-2}$ $= t^o_{2i-1} - t^o_{2i+1}$ $t^e_{2i+2}$ | | | Followed by (3.6) to give: <br> $t^e_{2i-2}$ $t_{2i}$ $t^e_{2i+2}$ | | |
|---|---|---|---|---|---|---|
| 1 | $\begin{Bmatrix}0\\1\\2\end{Bmatrix}$ | $-1 \quad -1$ <br> $0$ | $\begin{Bmatrix}-2\\-1\\0\end{Bmatrix}$ | | | |
| 2 | $\begin{Bmatrix}0\\1\\2\end{Bmatrix}$ | $-1 \quad 0$ <br> $-1$ | $\begin{Bmatrix}-1\\0\\1\end{Bmatrix}$ | | | |
| 3 | $\begin{Bmatrix}0\\1\\2\end{Bmatrix}$ | $-1 \quad 1$ <br> $-2$ | $\begin{Bmatrix}0\\1\\2\end{Bmatrix}$ | $\begin{Bmatrix}-1\\0\\1\end{Bmatrix}$ | $1$ | $\begin{Bmatrix}-1\\0\\1\end{Bmatrix}$ |
| 4 | $\begin{Bmatrix}-1\\0\\1\end{Bmatrix}$ | $0 \quad -1$ <br> $1$ | $\begin{Bmatrix}-2\\-1\\0\end{Bmatrix}$ | | | |
| 5 | $\begin{Bmatrix}-1\\0\\1\end{Bmatrix}$ | $0 \quad 0$ <br> $0$ | $\begin{Bmatrix}-1\\0\\1\end{Bmatrix}$ | | | |
| 6 | $\begin{Bmatrix}-1\\0\\1\end{Bmatrix}$ | $0 \quad 1$ <br> $-1$ | $\begin{Bmatrix}0\\1\\2\end{Bmatrix}$ | | | |
| 7 | $\begin{Bmatrix}-2\\-1\\0\end{Bmatrix}$ | $1 \quad -1$ <br> $2$ | $\begin{Bmatrix}-2\\-1\\0\end{Bmatrix}$ | $\begin{Bmatrix}-1\\0\\1\end{Bmatrix}$ | $-1$ | $\begin{Bmatrix}-1\\0\\1\end{Bmatrix}$ |
| 8 | $\begin{Bmatrix}-2\\-1\\0\end{Bmatrix}$ | $1 \quad 0$ <br> $1$ | $\begin{Bmatrix}-1\\0\\1\end{Bmatrix}$ | | | |
| 9 | $\begin{Bmatrix}-2\\-1\\0\end{Bmatrix}$ | $1 \quad 1$ <br> $0$ | $\begin{Bmatrix}0\\1\\2\end{Bmatrix}$ | | | |

conveniently allow numbers and their negatives to be represented without use of negative subscripts. The negative of a number is easily formed by the inversion of each coefficient in any ternary representation, also an advantage for hardware implementations using three-state devices or two-bits per cell binary equivalents [7]. Ternary Fibonacci representations also allow the sign of a number to be determined conveniently by observation of the most significant bit as mentioned in Section II.

The "ripples" that occur in the preliminary preparation transformations of Table 3.1 should be investigated for minimization and hardware implementation,

as well as for interfacing the Fibonacci processor with conventional binary processors and trade-offs between the two kinds of processors.

## REFERENCES


1. P. Ligomenides & R. Newcomb. "Equivalence of Some Binary, Ternary and Quaternary Fibonacci Computers." *Proceedings of the Eleventh International Symposium on Multiple-Valued Logic*, Norman, Oklahoma, May 1981, pp. 82-84.
2. P. Ligomenides & R. Newcomb. "Complement Representations in the Fibonacci Computer." *Proceedings of the Fifth Symposium on Computer Arithmetic*, Ann Arbor, Michigan, May 1981, pp. 6-9.
3. R. Newcomb. "Fibonacci Numbers as a Computer Base." *Conference Proceedings of the Second Interamerican Conference on Systems and Informatics*, Mexico City, November 27, 1974.
4. V. D. Hoang. "A Class of Arithmetic Burst-Error-Correcting Codes for the Fibonacci Computer." Ph.D. Dissertation, University of Maryland, December 1979.
5. V. E. Hoggatt, Jr. *Fibonacci and Lucas Numbers*. Boston: Houghton Mifflin Co., 1969.
6. P. Monteiro & R. W. Newcomb. "Minimal and Maximal Fibonacci Representations: Boolean Generation." *The Fibonacci Quarterly* 14, no. 1 (1976):9-12.
7. M. Stark. "Two Bits Per Cell ROM." *Digest of Papers, Compcon 81*, February 1981, San Francisco, pp. 209-212.


◆◇◆◇◆