Reduced Delay BCD Adder

Alp Arslan Bayrakçi and Ahmet Akkaş
Computer Engineering Department
Koç University
34450 Sarıyer, İstanbul, Turkey
abayrakci@ku.edu.tr ahakkas@ku.edu.tr

Abstract

Financial and commercial applications use decimal data and spend most of their time in decimal arithmetic. Software implementation of decimal arithmetic is typically at least 100 times slower than binary arithmetic implemented in hardware. Therefore, hardware support for decimal arithmetic is required. In this paper, a reduced delay binary coded decimal (BCD) adder is proposed. The proposed adder improves the delay of BCD addition by increasing parallelism. On the critical-path of the proposed BCD adder, there are two 4-bit binary adders, a carry network, one AND gate, and one OR gate. To make area and delay comparison, the proposed adder and previously proposed five decimal adders are implemented in VHDL and synthesized using 0.18 micron TSMC ASIC library. Synthesis results obtained for 64-bit addition (16 decimal digits) show that the proposed BCD adder has the shortest delay (1.40 ns). Furthermore, it requires less area than previously proposed three decimal adders.

1. Introduction

Human beings have preferred decimal as their number base for all calculations done by hand, since the time when the man learned to count on his ten fingers. This fact has never changed, although binary has been selected as the default base for almost all computers due to the storage and the speed efficiency of binary hardware [1]. The success of binary numbers was introduced in 1946, by the report of John von Neumann and his colleagues at the Institute for Advanced Study [2]. Afterwards, the designers have preferred binary computers due to the speed and the simplicity of binary arithmetic, but nowadays, there is an increasing demand for the decimal arithmetic hardware support in financial and commercial applications. This is due to three reasons, as stated in the paper of Cowlishaw [4]:

First, most fractional decimal numbers, such as 0.1, cannot be exactly represented in binary format and therefore, their approximate representations are used in binary arithmetic operations. This is not tolerable for most financial and commercial applications, which require exact representation of decimal numbers.

Second, the commercial databases contain more decimal data than binary data. Therefore, when the binary hardware is used, the decimal data is converted from decimal to binary and after it is processed, the binary data is converted back to decimal in order to store the result in decimal format. However, the conversion between decimal and binary formats causes too much delay [1].

Today the financial applications use decimal software running on top of the underlying binary hardware in order to produce exact decimal results, but the drawback of using decimal software is its speed. Software implementation of decimal arithmetic is about 100 to 1000 times slower than the binary implementation in hardware [4, 5]. When decimal operations are supported by software, the decimal processing overhead can even reach up to 90% for some applications like telephone billing [3]. This is the third reason for the requirement of the decimal arithmetic hardware in financial and commercial applications.

In all arithmetic units, whether binary or decimal, an adder is used. Therefore, it is not surprising that various addition techniques have been invented up to now, even for the decimal addition, which is much less popular than the binary addition. This paper does not only introduce a reduced delay BCD adder, but it also investigates the delay and the design properties of previously proposed five decimal adders.

The remainder of this paper is organized as follows: Section 2 gives brief background for decimal adders available in the literature. In Section 3, the reduced delay BCD adder is presented. Section 4 discusses the design details of adders presented in Section 2 and gives the synthesis results. Last section presents our conclusions.

1-4244-1027-4/07/$25.00 ©2007 IEEE

2. Previous Work

In this section, an overview is given for five decimal adders that have already been designed. These previously proposed popular BCD adders are implemented in VHDL and synthesized in order to make a comparison with the reduced delay BCD adder. The synthesis results and the design details of the adders are presented in Section 4.

The first adder, investigated in this paper, is the conventional decimal adder [8] and it is shown in Figure 1. For each decimal digit, it has two 4-bit binary adders and carry detection logic between the adders. The first level adders produce the binary addition results. If the result is greater than 9, a carry output is produced and the result of first level 4-bit adder is corrected by adding 6. Furthermore, the carry output is used as a carry input for the next digit. The main disadvantage of the conventional decimal adder is its low speed because all the first level 4-bit adders must wait for a number of 4-bit additions to get the right carry input.

The second adder is proposed in Hwang's patent [6]. This adder is designed to support both binary and decimal additions. A binary carry look-ahead adder (CLA) is used to add two input operands, which are either binary or decimal numbers. The result of the binary CLA is the correct result for binary inputs, but it needs to be corrected for decimal inputs. Therefore, in the decimal case, the binary CLA addition result and some of the carries (C[4],C[8],C[12],...) are used to compute the corrected result. In the correction step, digit generate and digit propagate signals are computed for each decimal digit (4-bit binary addition result). The digit generate signal becomes *one* if the 4-bit result is in the range of [10,15] and the digit propagate signal becomes *one* if the 4-bit result is 9. A carry look-ahead network takes these digit generate and digit propagate signals and produces outputs which are ORed with the carries coming from the binary CLA. The output of OR gate is used to determine whether 6 will be added for correction. Furthermore, the carry output of each decimal digit is also used to determine if an additional 1 will be added for correction.

The third adder is based on the paper of Shirazi et al., which uses the redundant binary coded decimal (RBCD) numbers [10]. The addition operation consists of three steps. First, inputs in BCD format are converted to RBCD format. In the second step, RBCD adder is used to compute the result. Finally, the result in the RBCD format is converted back to BCD format. The delay for the first two steps is constant and independent of the input size. On the other hand, converting RBCD result back to BCD requires carry propagation and it causes a delay correlated with the length of input operands. Therefore, RBCD adder is useful especially when constant-time addition operations are used many times, as Shirazi states in his paper [10].

The fourth adder, investigated in this paper, is proposed by Schmookler et al. [9]. It is the only decimal adder which finds the decimal carries before performing any addition. The idea is similar to the concept used in binary CLAs. For each decimal digit, two signals named as K and L are computed using input operands. K shows that the sum of two corresponding BCD digits will be greater than or equal to 10, and L shows that the sum will be greater than or equal to 8. The paper of Schmookler suggests that these signals can be used to find both the carries and the decimal addition result without using any correction step.

The fifth decimal adder is used in a 64-bit decimal floating-point adder proposed by Thompson et al. [11]. This adder consists of a pre-correction unit, which adds 6 to each digit, a Kogge-Stone binary adder, which adds two pre-corrected operands, and a post correction unit, which corrects the result by subtracting 6 (adding 1010) if it is required. Additions in the pre-correction and in the post-correction units are performed parallel for all digits. The post-correction unit is also responsible for the detection of correction (adding 1010).

3. Reduced Delay BCD Adder

The conventional BCD adder [8] is very simple, but also very slow due to the carry ripple effect. If the BCD addition is analyzed carefully, we see that there are three cases:

Case 1: The sum of two BCD digits is smaller than 9. In this case, it is certain that there is no carry output even if there is a carry input. Furthermore, the result for this digit does not require a correction.

Case 2: The sum of two BCD digits is greater than 9. In this case, a correction is required. Moreover, a carry output is produced regardless of the carry input.

Case 3: The sum of two BCD digits is exactly 9. In this case, the input carry determines whether a correction is required and whether a carry output is produced.

For the first two cases, the incoming carry has no effect on determining the carry output; therefore, the carry output can be determined without knowing the existence of the carry input. On the other hand, if the addition result is 9 (Case 3), then the input carry determines the existence of the carry output, which may ripple even up to the most significant digit. Therefore, Case 2 and Case 3 can be represented by a digit generate (DG) and a digit propagate (DP) signals, respectively. This representation is similar to the digit generate and the digit propagate representations in Hwang's patent [6].

Figure 2 shows how the DG and DP signals of a digit are computed in our design. After having all the DG and DP signals, the output carry for each digit can be found easily by Equation 1. Due to the nature of this equation, we can form DP by ANDing only Sum[0] and Sum[3] instead of using all bits of Sum[3:0]. The DG and DP signals can be

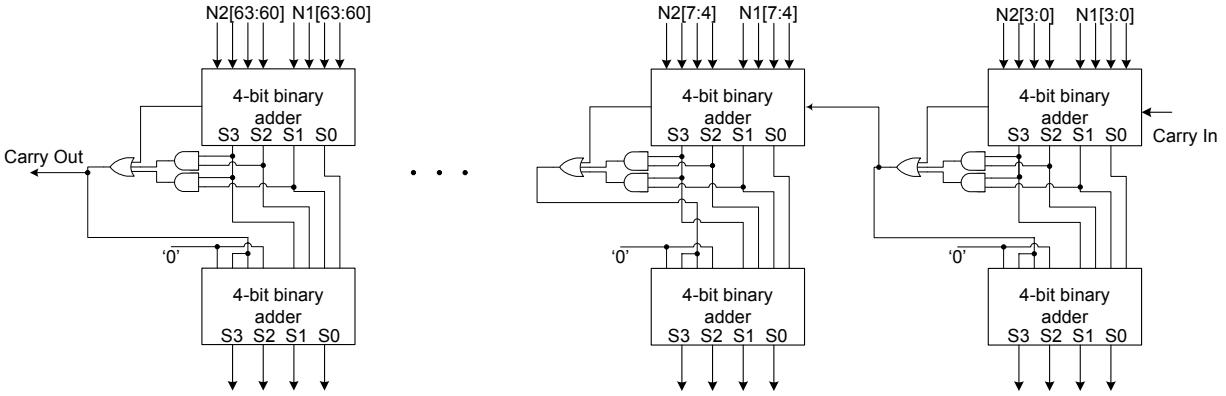

Figure 1. Conventional BCD Adder

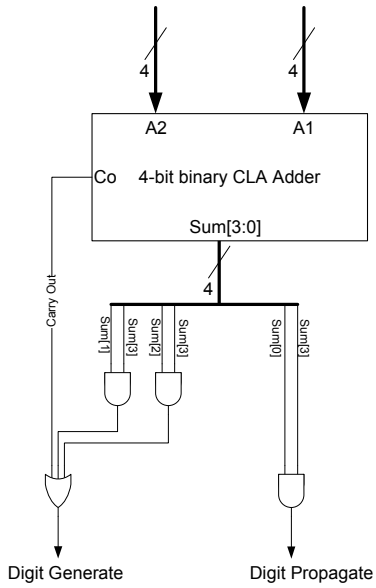

Figure 2. Adder + Analyzer Unit

utilized similar to the generate and propagate signals used in a binary CLA circuit. Therefore, all schemes developed for a binary CLA can be used in order to speed up carry computation.

$$OutputCarry = DG + DP \cdot InputCarry \qquad (1)$$

The combination of the first level 4-bit adders and the Carry Network is shown in Figure 3. The carry value for each digit is computed inside the Carry Network using Equation 1. The Carry Network can be any type of parallel prefix network or two level carry look-ahead logic can be used instead. The carries computed by Carry Network are

The value added for correction	Possible Cases	
	Input Carry from prev. digit	Output Carry to next digit
0	0	0
6	0	1
1	1	0
7	1	1

Table 1. The Selection of the Value to be Added for Correction

used in the correction step. Figure 4 shows the complete BCD adder including the 4-bit adders used for correction.

Correction is done by adding 0, 1, 6, or 7 to the binary sum coming from the first level adder. For each digit, the existence of the output carry and the input carry determine the value to be added for correction. Table 1 shows the correction value to be added for all cases. The correction step must fulfil two requirements. First, the carry, coming from the previous digit, should increment the binary sum of the related digit by 1. Second, the output carry of the related digit should determine whether the binary sum will be corrected by adding 6 or not. The correction by adding 6 is required only when there is a decimal carry-out coming from the carry network. Figure 4 shows how the correction step fulfils these requirements. As a result of such a design, the carries are utilized only in the correction step.

There are three differences between our design and the design presented in Hwang's patent. First, a full binary CLA is used in Hwang's patent for the binary addition of two input operands in order to support both decimal and binary additions. Whereas, in order to speed up the operation, 4-bit independent binary adders are used in our design, supporting only the decimal addition. By this way, the first

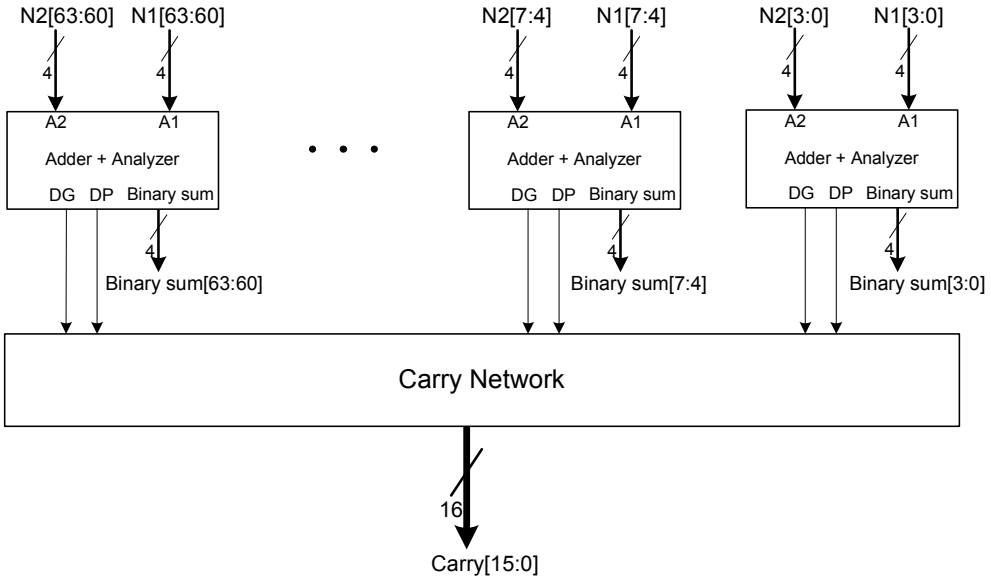

Figure 3. Adder + Analyzer + Carry Network

level binary summations are computed in constant time, independent from the length of the input operands.

Second, the digit generate signal used in our design is slightly different than the digit generate signal used in the patent. The digit generate signal used in the patent becomes *one* for a result that is in the range of [10,15]. However, the digit generate signal used in our design becomes *one* for a result that is greater than or equal to 10 (of course the active range becomes [10,18], as the sum of two BCD digits cannot be greater than 18). This is achieved by ORing the carry output of 4-bit binary adder and the outputs of two AND gates as shown in Figure 2. This also eliminates the use of an additional OR gate, which is used just before the correction step in Hwang's patent. In our design, first level 4-bit addition results are computed without utilizing carry inputs coming from the previous 4-bit adders. On the other hand, Hwang uses these carries in the full binary addition of two operands. Furthermore, these carries are also used one more time before the correction and it causes an additional OR gate delay as shown in Figure 5. This does not only bring one more gate delay, but it is also wasteful, as these carries are taken into account twice; in the full binary addition and before the correction step.

The last difference between the two design is related to the computation of the decimal digit carries. In Hwang's patent, a conventional CLA network is used. In our design, a parallel prefix network [7] is used to reduce the delay of decimal addition.

When the proposed adder shown in Figure 4 is considered, the key point of our design is the parallelism. Not only the first level adders perform their additions in parallel, but also the correction step is performed using independent 4-bit binary adders. As a result, all parts of the reduced delay BCD adder, except for the Carry Network, perform their operations in constant time, independent from the length of the input operands. Yet, there are different schemes like Kogge-Stone parallel prefix network, used in this paper, in order to decrease the delay of the Carry Network.

4. Synthesis Results

All adders referred in this paper are designed to support 64-bit decimal addition with BCD operands. They are synthesized using the TSMC 0.18 micron library. In order to make a fair comparison, three of these adders are slightly modified and details of modifications are given below. The conventional adder, RBCD adder proposed by Shirazi et al., and the reduced delay BCD adder presented in this paper are implemented exactly how they are presented.

The adder presented in Hwang's patent is modified to support only decimal addition. Therefore, "nine's complementer" block, which is used only in decimal subtraction, and the multiplexer, which is used to select either the binary or the decimal result, are removed. Furthermore, a 64-bit binary CLA is used instead of the binary ALU. All these modifications speed up the operation. The modified version of Hwang's adder is shown in Figure 5.

The other modified adder is the adder of Schmookler et al. [9]. This adder finds all carries before performing any addition. The boolean equations from 2a to 10 presented in [9] are used exactly, but the organization of the circuit is different than the devised circuit. Instead of using byte

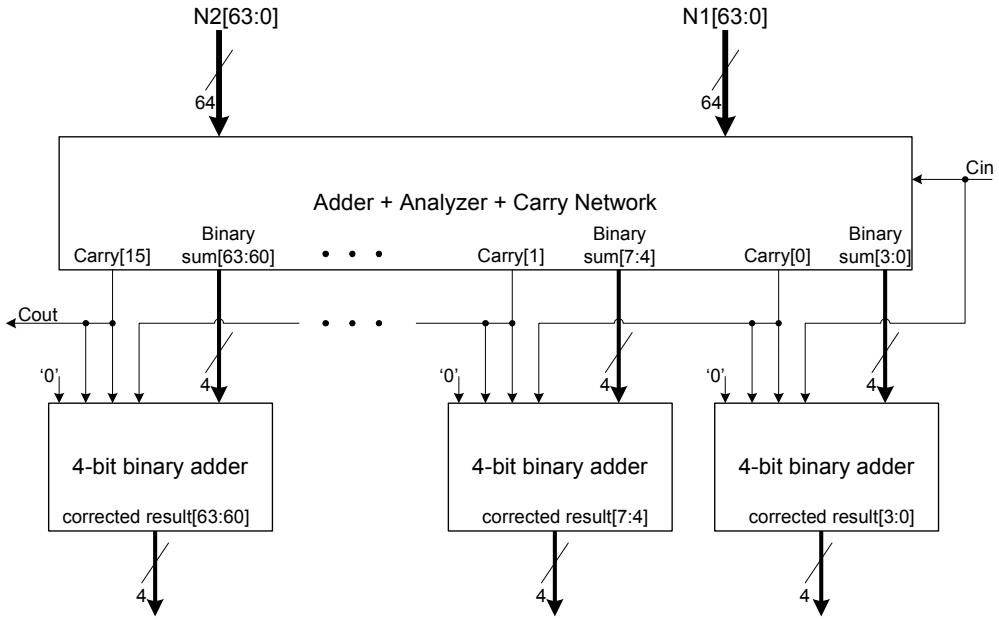

Figure 4. Full Circuit

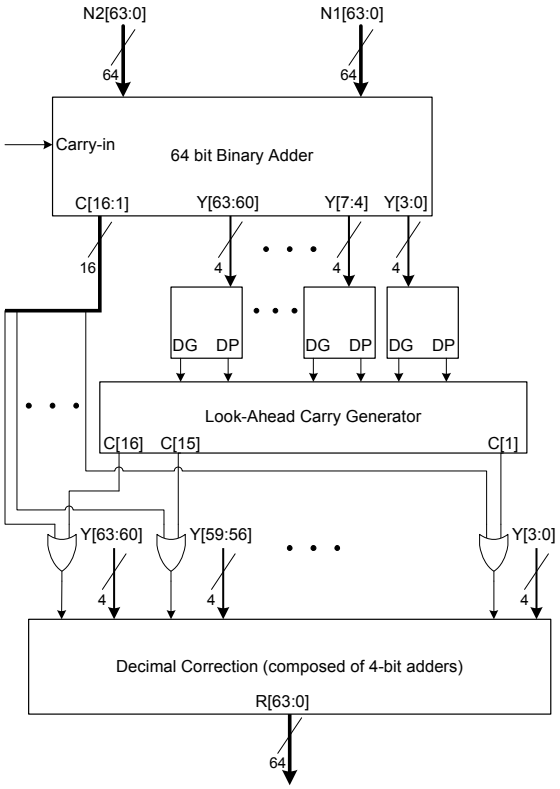

Figure 5. Hwang's Modified Adder

generate and byte propagate signals presented in [9], digit propagate (P) and digit generate (G) signals are computed by Equation 2 and 3, where Ni(0) represents the least significant bit of the related digit of the i^{th} operand. The meaning of K and L signals are already given in Section 2.

$$P = L \cdot (N1(0) + N2(0)) \tag{2}$$

$$G = K + (L \cdot N1(0) \cdot N2(0)) \tag{3}$$

Schmookler proposes a byte-wise circuit scheme with 6 levels using the byte generate and byte propagate signals. However, in our implementation, digit generate and digit propagate signals are the inputs of a two level CLA network.

The third modified adder is the adder used in the 64-bit decimal floating-point adder [11]. The boolean functions of the pre-correction and post-correction units are used without any modifications except that the boolean functions, used for other purposes than the decimal addition, are removed from the design. Another minor modification in this adder is the replacement of the 76-bit Kogge-Stone binary adder with a 64-bit Kogge-Stone binary adder to support only 64-bit decimal addition.

All adders referred in this paper are implemented in VHDL according to the details given above and in Section 2. They are synthesized using Mentor Graphics' Leonardo Spectrum synthesis tool and the TSMC 0.18 micron CMOS standard cell library. The synthesis results are presented in Table 2.

Adder	Delay (ns)	Area (gates)
1. Conventional BCD Adder	11.03	955
2. Hwang's adder	3.41	1762
3. RBCD Adder	4.74	3784
4. The CLA version of Schmookler's adder	1.54	1336
5. The adder of Thompson et al.	1.69	2369
6. Reduced Delay BCD Adder	1.40	1422

Table 2. Area and Delay Estimates for Decimal Adders

According to Table 2, the conventional decimal adder is very slow because of the carry propagation. The RBCD adder has significantly better delay than the conventional adder, but it requires a time consuming operation to convert RBCD number to BCD. The delay of modified version of the adder presented in Hwang's patent is better than the delay of RBCD adder, but it still requires two time consuming operations. The first operation is to perform 64-bit binary addition of two operands and the second operation is to detect carries for each digit position using carry look-ahead network. The other three adders perform decimal addition very efficiently. Among these adders, the proposed reduced delay BCD adder has the shortest delay.

When the area of the adders is under consideration, the worst one is RBCD adder whereas the best of these six adders is conventional adder. However, the delay of the conventional adder is much higher than the other adders. The second best adder, in terms of area, is the CLA version of Schmookler's adder. Reduced delay BCD adder also has a smaller area compared to the other adders except the conventional BCD adder. Its area is very close to Schmookler's adder and it has shorter delay than Schmookler's adder.

Reduced delay BCD adder can be modified in order to also support subtraction operation if the 10's complement arithmetic is supported by the architecture. The idea is as follows: If the operation is subtraction, the first operand should be added to the 9's complement of the second operand and the carry-in of the adder should be set to 1. Taking 9's complement is an easy and fast task as explained in the paper of Schmookler [9]. It should also be noted that the adder used in [11] can support both addition and subtraction operations even for the sign magnitude representation.

5. Conclusion

This paper presents a reduced delay BCD adder to perform decimal addition. The new proposed adder has the shortest delay among the decimal adders examined in this paper. It also requires less hardware than the previously proposed three decimal adders. The new decimal adder improves the delay of BCD addition by increasing parallelism. If we ignore one AND gate and one OR gate delay, the critical-path of the proposed BCD adder consists of the delay of two 4-bit binary adders and a carry network.

References

[1] W. Buchholz. Fingers or Fists? (The Choice of Decimal or Binary Representation). *Communications of the ACM*, 2(12):3–11, December 1959.

[2] A. H. Burks, H. H. Goldstine, and J. von Neumann. Preliminary Discussion of The Logical Design of An Electronic Computing Instrument. Technical report, Institute for Advanced Study, June 1946.

[3] M. F. Cowlishaw. The 'telco' benchmark. May 2002. URL: http://www2.hursley.ibm.com/decimal.

[4] M. F. Cowlishaw. Decimal Floating-Point: Algorism for Computers. In *Proceedings of 16th IEEE Symposium on Computer Arithmetic*, pages 104–111, June 2003.

[5] M. A. Erle, M. J. Schulte, and J. M. Linebarger. Potential speedup using decimal floating-point hardware. In *Conference Record of the Thirty-Sixth Asilomar Conference on Signals, Systems and Computers*, volume 2, pages 1073–1077, November 2002.

[6] I. S. Hwang. High Speed Binary and Decimal Arithmetic Unit. *United States Patent*, (4,866,656), September 1989.

[7] P. M. Kogge and H. S. Stone. A Parallel Algorithm for The Efficient Solution of a General Class of Recurrence Equations. *IEEE Trans. on Computers*, C-22(8), Aug. 1973.

[8] M. M. Mano. *Digital Design*, pages 129–131. Prentice Hall, third edition, 2002.

[9] M. S. Schmookler and A. W. Weinberger. High Speed Decimal Addition. *IEEE Transactions on Computers*, C-20:862–867, Aug. 1971.

[10] B. Shirazi, D. Y. Y. Young, and C. N. Zhang. RBCD: Redundant Binary Coded Decimal Adder. In *IEE Proceedings, Part E, No. 2*, volume 136, pages 156–160, March 1989.

[11] J. D. Thompson, N. Karra, and M. J. SchulteB. A 64-Bit Decimal Floating-Point Adder. In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, pages 297–298, February 2004.