Exploring CMOS Technology for High-Speed and Efficient BCD Adder Designs

Santhoshini P [1], Varsha M[2], Varshini S[3], and Prathiya P[4]
[1] *Assistant Professor, RMD Engineering college.*
[2,3,4] *UG Student, RMD Engineering college.*

***Abstract*—Applications of decimal arithmetic become more significant in the domains of science and finance. Conversions from binary to decimal and from decimal to binary are necessary when doing decimal arithmetic on binary hardware. These transformations result in imprecise outcomes that cost businesses money. Consequently, decimal hardware is critically needed. The paper suggests circuits for adding decimals and shows how complementary metal-oxide semiconductor (CMOS) technology might be used to implement them. The suggested circuits are simulated and their operation is confirmed using the LTSPICE SPICE simulator program. The circuits are compared to previous efforts in the literature and simulated utilizing 45nm, 65nm, and 180nm technologies. Owing to the dearth of prior research in the field and for comparative purposes, this work was additionally designed. Fourier Transform (FFT) is an important signal processing technique widely used in various applications, such as audio and image processing, telecommunications, and scientific computing [1]. Many electronic applications require the best FFT with the least area, low power consumption, high speed. To design an FFT, a multiplier and adder are required. To perform multiplication, adder is the important block. So, if addition is done faster, FFT also work faster. The working of various adders like ripple-carry adder, Sklansky adder, Kogge-Stone adder, and Brent-Kung adder is described in this thesis. By using these various adders working of FFT is also described in this thesis. The ripple-carry adder, Sklansky adder, Kogge-Stone adder, and Brent-Kung adder are implemented in Verilog using Cadence tools. By using these various adders, FFT using RCA, FFT using Sklansky adder, FFT using Kogge-Stone adder, and FFT using Brent-Kung adders are implemented in Verilog using Cadence tools. The performance of various FFT's is compared in terms of cell count, power, and delay.**

Key Words: **Addition circuits, BCD adders, CMOS, decimal arithmetic, decimal hardware, LTSPICE, PDP.**

I. INTRODUCTION

execute the computations with sufficient word length in order to attain the precision required by these applications, which in some cases requires more than 32 digits [1]. Converting between decimal and binary is necessary when doing calculations over binary units, and this adds unacceptably long latency [2], [3]. Furthermore There are many applications in finance and science that use decimal numbers. Commercial databases, for instance, include more decimal data than binary data. Decimal arithmetic standards are employed to, these conversions could yield imprecise outcomes, particularly when converting fractional amounts between binary and decimal systems [4], [5]. As a result, the precision is significantly impacted. Therefore, to create fewer rounding mistakes, it is preferable to directly do the calculation in decimal rather than in binary [6], [7]. As a consequence, the need for embedding decimal hardware components within processors becomes attractive and justified [1], [6], and [8]. New innovations in technologies and new financial and business applications arise starting year 2000. Usually, the numbers in the database of these applications are in decimal format. IBM is considered the first processor company to include dedicated decimal arithmetic units in their processors [1]. Examples where IBM uses decimal arithmetic unit are found in IBM eServer z900 [9], IBM POWER6, and IBM z10 [10]. Mathematical tools and solutions which have been con- sistently playing a prominent role in a variety of science and engineering fields to address technical issues (see, e.g., [11], [12]) do not offer effective solutions for such applications. In addition, software solutions are not favored for such applications since they are generally slower than their hardware counterparts for large scale applications [13]. As a result, experiments have been conducted in recent years to improve decimal arithmetic in electronic devices. Different addition techniques have been devised since the adder is utilized in arithmetic tasks, whether in decimal or binary units. This study compares and implements the Binary Coded Decimal (BCD) delay drop, as well as looks at the power reduction of five proposed decimal additions. In computers, BCD is often used to encode decimal values. BCD is a number representation system with

a range from 0 to 9, where each decimal digit is represented separately by 4-bit. For example, the decimal number 546 in BCD is (0101 0100 0110) BCD [14], [15], [16].One significant advantage that BCD representation has over binary representation is the ease with which a decimal number can be converted from its text representation to its BCD representation. Since many frequently used values between 0 and 1 cannot be precisely represented by fixed and floating point binary representations, this capability is helpful when working with fractional numbers.

A BCD adder adds two decimal digits with a carry-in input and generates the decimal sum and the carry-out output. Typically, it consists of two 4-bit binary adders that are integrated with correction and carry generation circuits. The two decimal operands and the carry-in are added using the first 4-bit binary adder. If the resulting sum is greater than 9 (invalid decimal digit) then a correction is mandatory. The correction is done by the second binary adder, which adds 6 to the result and passes any resulting carry to the output [17].

In the field of contemporary electronics, the utilization of CMOS (Complementary Metal-Oxide-Semiconductor) circuits has become remarkably prevalent due to their exceptional qualities. These circuits have discovered diverse applications across various domains, owing to their dis- tinctive advantages. A particularly noteworthy attribute is the capacity of CMOS transistors to achieve minimal time delay, a feature of immense significance when aiming for precision in electronic system design. Furthermore, they are acknowledged for their effectiveness in conserving power, a pivotal aspect in the creation of energy-efficient designs.

These attributes assume heightened significance during the development of electronic systems, directly influencing both speed and energy efficiency. The integration of CMOS technology has ushered in substantial progress in the realm of electronics, enabling the realization of high-performance, energy-efficient components such as microprocessors and memory modules. It is worth noting that numerous studies in the existing body of research have played a pivotal role in advancing the precision of CMOS circuits, further solidifying their prominence in contemporary electronics. Moreover, dedicated efforts have been invested in the evolu- tion of power supply technologies, consequently facilitating the emergence of energy-efficient, compact, and portable electronic systems. Additionally, pioneering methodolo- gies have paved the way for the creation of efficient and symmetrical CMOS logic circuits, unlocking fresh opportu- nities for design innovation. In tandem, these endeavors have made significant contributions to comprehending and analyzing the performance of CMOS circuits.

In essence, CMOS circuits, fortified by inventive calibration techniques, state-of-the-art power supply algorithms, inventive design approaches, and advanced analytical methods, persist as cen- tral components in modern electronics, catalyzing innovation and empowering the development of progressively intricate electronic systems. The main contribution of this work is the design of two- stage, correction-free, fast, and low-power one-digit decimal addition CMOS circuit. This one-digit addition circuit is utilized in designing higher digits decimal addition circuits which is used in decimal arithmetic units that support decimal-based applications. In addition, for comparison purposes, this work build five different decimal adders based on existing binary adders. The proposed decimal adder and the other five decimal adders are used to build 2-digit, 3-digit, and 4-digit decimal adders that are realized and simulated using different CMOS technology libraries.

All of the one-digit decimal addition circuits are imple- mented using CMOS technology, which has many advantages over other technologies. For example, power dissipation is very low in digital CMOS circuits, since they dissipate power during switching time only. Moreover, in comparison to other types of transistors, CMOS devices can be scaled down more easily. Additionally, recent advances in CMOS technologies have improved the intrinsic speed of CMOS devices, which made it faster than other type of devices. In general, CMOS technology is preferred in digital circuit design over other type of technologies due to its low fabrication cost.

2.LITERATURE REVIEW

BCD addition of n-digit operands is usually performed by cascading n 1-digit BCD adders in a ripple carry structure as illustrated by Figure 1, where, X_i and Y_i; $i = 0, 1, 2, \cdots, n-1$ are the decimal digits of the two operands. In this case, if the delay for the 1-digit BCD adder is assumed to be T, then the delay for the n-digit BCD adder is linearly increasing with n and

it can be estimated to be nT for large n. Performing a 1-digit BCD addition is achieved by adding the BCD codes of the two decimal digits in binary and if the result is greater than 9 or if a $carry_{out}$ is generated,

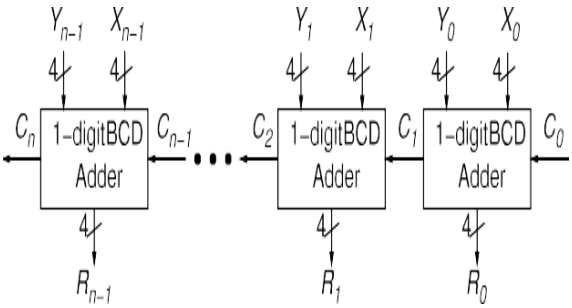

Fig 1: A block diagram for *n*-digit BCD adder.

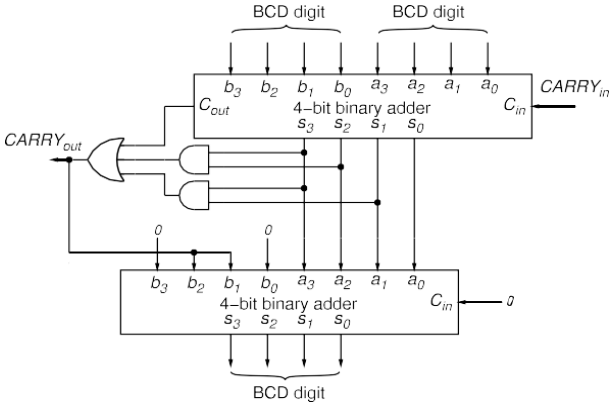

Fig 2: A conventional 1-digit BCD adder.

a correction is done by adding $(0110)_2$ to the result in order to guarantee a result in the range of $[0 - 9]$. For example, to perform $5 + 3$, we perform the binary operation $(0101)_2 + (0011)_2$. In this case the result, which is $(1000)_2$, is not greater than 9 and no correction is needed. On the other hand, to perform $6 + 7$, the binary operation $(0110)_2 + (0111)_2$ is performed. In this case the result, which is $(1101)_2$, is greater than 9 and a correction is required. Therefore, $(0110)_2$ is added to $(1101)_2$ to get $(0011)_2$ which is $(3)_{10} = (0011)_{BCD}$. In fact, this is the correct result. The last scenario is when a $carry_{out}$ is generated after adding the BCD codes of the two decimal digits in binary. This can happen if the input decimal digits are, for example, 8 and 9. Here, the operation $(1000)_2 + (1001)_2$ is performed and the result is $(0001)_2$. A $carry_{out}$ is generated in this case. It is clear that the result is wrong despite the fact the it is not greater than 9. This is detected by the generated $carry_{out}$ and a correction is needed. Hence, $(0110)_2$ is added to $(0001)_2$ to get the correct results $(0111)_2$ which is $(7)_{10} = (0111)_{BCD}$. Any generated carry is considered a $carry_{out}$ from the 1-digit BCD addition operation and must be rippled to next stage, if any. Based on the previous discussion, a 1-digit BCD adder is conventionally designed using two-stage binary addition. In the first stage, the initial binary addition of the BCD codes of the two decimal digits is preformed. Whereas, the second stage is required for correction whenever a correction is detected by a dedicated logic. A 1-digit BCD is illustrated by Figure 2 where it is clear that the correction is done once the result from the first binary adder is greater than 9 or a $carry_{out}$ is generated from the first binary adder.

There are many works in the literature that address the BCD addition operations and the implementation of BCD adders. Some of the proposed BCD adders in the literature are used for two-operand operation while the others are multi-operand BCD adders. Generally, one decimal digit operands or multi decimal digits operands are addressed. For example, the work in present two techniques to design high speed and economical decimal adders at the gate level. Another example, where also gate level implementation is used to design decimal addition/subtraction unit, is proposed . It can be easily figured out that most of the imple- mentations of the BCD adders presented in the literature are high-level implementations where ASICs or FPGAs environment is used. In fact, only few low-level imple- mentations at the transistor level are found. As discussed, one way to build a BCD adder is to use two-level binary addition with a correction circuit. Therefore, it is worth to address the low-level implementation of existing binary adder and use them to design BCD adders for comparison purposes. Therefore, in addition to the existing low-level implementations of BCD adders, this work uses different transistor level binary adder to design BCD adders that are used for comparison purpose with the proposed CMOS BCD adder.

III. PROPOSED BCD ADDER

In the sequel, we present, discuss, and realize a CMOS based fast BCD digit adder which we use to build higher order BCD adders. The adder achieves high speed since being correction- free. It is developed as a two-level netlist: netlist1 and netlist2 as illustrated by Figure 8. To do so, we first note that we let the BCD adder have two decimal inputs $A = A_3A_2A_1A_0$ and $B = B_3B_2B_1B_0$, where $A, B \in \{0, 9\}$ and C_{in} be the decimal carry input. Additionally, we let the output

decimal sum be $S \in \{0, 9\}$ and the output decimal carry be C_{out}. Then, the A and B terms are expressed in the Boolean functions in (6) and (7) in terms of two integers

$I = B_3B_2B_1$ and $J = A_3A_2A_1$ as follows:

$$A = (2 \times J) + A_0, \quad (6)$$

$$B = (2 \times I) + B0, \quad (7)$$

Since A and B are two decimal digits, we have $0 \leq J \leq 4$ and $0 \leq I \leq 4$.

As a result, the BCD adder output can be written as follows:

$$\{C_{out}, Sum\} = A + B + C_{in}$$

$$= (2 \times J + A_0) + (2 \times I + B_0) + C_{in}$$

$$= 2 \times (J + I) + (A_0 + B_0 + C_{in})$$

$$= 2 \times K + (A_0 + B_0 + C_{in}), \quad (8)$$

where $K = (J + I) = (K_3K_2K_1K_0)_{BCD}$ and $2 \times K = (K_3 K_2K_1K_00)_{BCD}$ is generated using Netlist1 of Figure 3.

K_3 represents the decimal carry output and its value is either 1 or 0. $(K_2K_1K_00)_{BCD}$ is an even decimal digits which means that the least significant bit (LSB) of $2 \times K$ is always 0. Therefore, only K_3, K_2, K_1, and K_0 are forwarded to Netlist2 in order to add the term $(A_0 + B_0 + C_{in})$ to $2 \times K$ to get the final result.

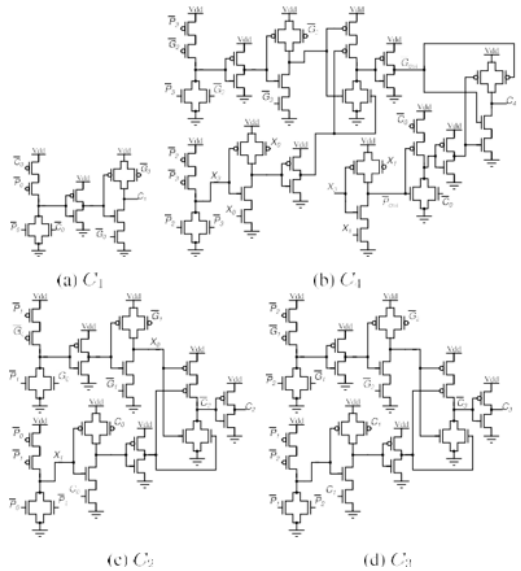

Fig 3: Novel 4 Bit CLA Architecture

The CMOS realization of Netlist1 is presented in Figure 9. The inputs to Netlist2 are K, A_0, B_0 and C_{in}, which are added to produce the final result which consists of the final decimal carry output C_{out} and the final decimal digit result $S = (S_3S_2S_1S_0)_{BCD}$. K must be aligned properly in this addition process such that $2 \times K$ is evntually added.

For instance, if the values of K_3, K_2, K_1 and K_0 are 1, 0, 0 and 1, respectively, then this means that the value of $2 \times K$ to be added to the value of $(A_0 + B_0 + C_{in})$ is $(12)_{10} = (1\ \ 0010)_{BCD}$. If we assume the values of A_0, B_0, and C_{in} are 1, and 0, respectively, then this means that the value of $A+B = \times K + (A_0 + B_0 + C_{in})$ is $(14)_{10} = (1\ \ 0100)_{BCD}$. C_{out}, S_3, S_2, S_1, and S_0 are computed to be 1, 0, 1, 0 and 0, respectively.

On this basis, C_{out}, S_3, S_2, S_1, and S_0 are computed for all possible combinations of K, A_0, B_0 and C_{in} in order to obtain an optimized NAND-NAND implementation for Netlist2. This NAND-NAND implementation of Netlist2 is presented by (10) and is realized using CMOS technology

IV. RESULTS AND DISCUSSIONS

For this research, *LTSPICE* program aided design tools are used as the standard circuit design and simulator tool. The circuits of the proposed decimal adder are implemented using dynamic logic gates using 45*nm* BSIM4 model for 1-digit, 2-digit, 3-digit, and 4-digit decimal operands. The transient analysis is performed at a frequency of 100MHz. The inverter has a width of 270*nm* for the p-MOS and 135*nm* for the n- MOS. For all simulations, the supply voltage is set to $1V$. In order to determine the critical path delay in each design, the delay of all paths from inputs to the most significant bit (MSB) of the sum output and the delay of all paths from input to the final decimal carry out are investigated. Beside the existing low-level decimal adder works pre- sented in literature, which are found to be few as mentioned in Section II, this work builds five different decimal adders for comparison purposes with the proposed decimal adders. These five adders are designed following the 1-digit BCD adder architecture presented in Figure 2, where low-level binary adder implementations from and are used in addition to the conventional low-level binary adder which is presented in different works in literature.. Similarly, 1-digit, 2-digit, 3-digit, and 4-digit decimal adder versions are built out of these five adders and the critical path delay is determined using the same way used to determined the critical path delay for the proposed decimal adder.In this work, L is set to be 45*nm*, 65*nm*, 180*nm* for the 45*nm* process, the

65*nm* process, and 180*nm* process; respectively. The transistor count *N* is the same no matter what process

V. CONCLUSION

A high performance 1-digit BCD adder is proposed. The adder is designed using two level netlist such that correction is interleaved within the design. The Boolean equations for each netlist are derived and realized in CMOS technology targeting 45nm technology. The 1-digit BCD adder is used to build larger BCD adders using ripple carry structure. The BCD adders are compared against existing designs. Some of these designs are realized in CMOS targeting 180nm and 65nm technologies. Therefore, the proposed design is also implemented in CMOS targeting 180nm and 65nm technologies for fair comparisons. Due to the lack of existing similar works in literature, following the traditional BCD adder design, this work build five BCD adders using existing binary adders in literature. All of the designs are simulated using LTSPICE tools for different operands length (1-digit, 2-digit, 3-digit, and 4-digit). The proposed BCD adders show an improvement in terms of speed and power compared to other designs.

REFERENCES

[1] M. Véstias and H. Neto, ‘‘Improving the area of fast parallel decimal multipliers,’’ Microprocessors Microsyst., vol. 61, pp. 96–107, Sep. 2018.

[2] H. Thapliyal and N. Ranganathan, ‘‘Design of efficient reversible logic- based binary and BCD adder circuits,’’ ACM J. Emerg. Technol. Comput. Syst., vol. 9, no. 3, pp. 1–31, Sep. 2013.

[3] L. Dadda, ‘‘Multioperand parallel decimal adder: A mixed binary and BCD approach,’’ IEEE Trans. Comput., vol. 56, no. 10, pp. 1320–1328, Oct. 2007.

[4] H. Calderon, G. Gaydadjiev, and S. Vassiliadis, ‘‘Reconfigurable universal adder,’’ in Proc. IEEE Int. Conf. Appl.-Specific Syst., Architectures Processors (ASAP), Jul. 2007, pp. 186–191.

[5] S. Siddha, ‘‘Area efficient 4-input decimal adder using CSA and CLA,’’ J. Sci. Technol., vol. 2, no. 6, pp. 550–553, Jun. 2013.

[6] M. F. Cowlishaw, ‘‘Decimal floating-point: Algorism for computers,’’ in Proc. 16th IEEE Symp. Comput. Arithmetic, Jun. 2003, pp. 104–111.

[7] Y. D. Ykuntam and S. H. Prasad, ‘‘A modified high speed and less area BCD adder architecture using mirror adder,’’ in Proc. 2nd Int. Conf. Smart Electron. Commun. (ICOSEC), Oct. 2021, pp. 624–627.

[8] M. Cornea, ‘‘IEEE 754R decimal floating-point arithmetic: Reliable and efficient implementation for Intel architecture platforms,’’ Intel Technol. J., vol. 11, no. 1, pp. 10–19, Feb. 2007.

[9] F. Y. Busaba, C. A. Krygowski, W. H. Li, E. M. Schwarz, and S. R. Carlough, ‘‘The IBM z900 decimal arithmetic unit,’’ in Proc. Conf. Rec. 35th Asilomar Conf. Signals, Syst. Comput., Nov. 2001, pp. 1335–1339.

[10] C. F. Webb, ‘‘IBM z10: The next-generation mainframe microprocessor,’’ IEEE Micro, vol. 28, no. 2, pp. 19–29, Mar. 2008.

[11] M. D. Al-Khaleel, M. J. Gander, and A. E. Ruehli, ‘‘Optimized waveform relaxation solution of RLCG transmission line type circuits,’’ in Proc. 9th Int. Conf. Innov. Inf. Technol. (IIT), Mar. 2013, pp. 136–140.

[12] M. Al-Khaleel and S.-L. Wu, ‘‘A mathematical analysis of discrete waveform relaxation algorithms for transmission line type circuits,’’ in Proc. IEEE Asia–Pacific Conf. Comput. Sci. Data Eng. (CSDE), Dec. 2021, pp. 1–5.

[13] M. Riaz-ul-haque, M. Shintani, and M. Inoue, ‘‘Decimal multiplication using combination of software and hardware,’’ in Proc. IEEE Asia–Pacific Conf. Circuits Syst. (APCCAS), Oct. 2018, pp. 239–242.

[14] O. Al-Khaleel, M. Al-Khaleel, Z. Al-QudahJ, C. A. Papachristou, K. Mhaidat, and F. G. Wolff, ‘‘Fast binary/decimal adder/subtractor with a novel correction-free BCD addition,’’ in Proc. 18th IEEE Int. Conf. Electron., Circuits, Syst., Dec. 2011, pp. 455–459.

[15] O. Al-Khaleel, Z. Al-Qudah, M. Al-Khaleel, R. Bani-Hani, C. Papachristou, and F. Wolff, ‘‘Efficient hardware implementations of binary-to-BCD conversion schemes for decimal multiplication,’’ J. Circuits, Syst. Comput., vol. 24, no. 2, Feb. 2015, Art. no. 1550019.

[16] B. Shirazi, D. Y. Y. Yun, and C. N. Zhang, ‘‘RBCD: Redundant binary coded decimal adder,’’ IEE Proc. E Comput. Digit. Techn., vol. 136, no. 2, p. 156, Mar. 1989.

[17] A. A. Bayrakci and A. Akkas, ‘‘Reduced delay BCD adder,’’ in Proc. IEEE Int. Conf. Application-Specific Syst., Architectures Processors (ASAP), Jul. 2007, pp. 266–271.