Low Area & Power Decimal Addition with Improved Speed using LING Adder Architecture

Sindhu T[1] Brindha P[2]
[1]M.E Student [2]Assistant Professor
[1,2]Department of Electronics and Communication Engineering
[1,2]Velalar College of Engineering and Technology, Erode-638012, Tamil Nadu, India

Abstract— Adder plays a major role in electronics, not only in computers but also in many types of digital systems in which the numeric data are processed. In several applications like monetary and business computation requires decimal arithmetic operator to directly process decimal data for accurate computations and hence adder must perform decimal addition. Faster decimal addition can be performed with an embedded world-wide binary adder, leading to considerable hardware sharing with binary input unit. Carry-select technique for the decimal adders can be improved by computing pairs of corrective carry out bits for all decimal positions in parallel. This was performed by the quaternary parallel prefix network and the selection is based on corresponding positional carry-in bits. Carry select sum and Carry select correction block with reduced area than the previous structures is used in this fused binary/decimal adder. Area improvement will reduce the cost and power consumption of the system. Ling adder structure is used in the prefix network for speed improvement.

Key words: Decimal computer arithmetic, Speculative decimal addition, Combined binary/decimal-add/sub, Carry-select correction.

I. INTRODUCTION

Decimal computer arithmetic has been around since the birth of modern computing. One of the earliest decimal computers, the ENIAC and calculators used a decimal base for its arithmetic operations. Realization of decimal arithmetic operations via hardware circuits that operate on Binary Coded Decimal (BCD) data has been an attractive research subject for decades. If the conversion of decimal numbers to their binary equivalent, perform binary arithmetic operations on it and convert the results back to decimal will not produce an accurate result.

Therefore, two methods for decimal addition operations are used. One method is by backing binary processors with libraries of optimized software programs that simulate decimal arithmetic operations with no conversion of decimal fractions to binary. This direction has influenced the IEEE 754-2008 standard for decimal floating point arithmetic to include Binary Integer Decimal (BID) encoding for the significance of decimal floating point data. To reduce storage cost of decimal data, Densely Packed Decimal (DPD) encoding is used at the cost of conversion to BCD encoding and the reverse, before and after each arithmetic operation respectively.

The second method is to operate directly on the Binary Coded Decimal data. Decimal addition forms the core of other arithmetic operations such as multiplication and division. For the fast decimal multipliers and dividers require efficient decimal additions. Here the decimal numbers are assumed as Binary Coded Decimal data. There are commercialized binary digital processors that support hardware units which directly operate on binary coded decimal data. Therefore, the requirement of high-performance and low power decimal applications can be satisfied. These decimal arithmetic hardware realizations often share circuits with binary arithmetic units for cost reduction. Fused binary/decimal units and unified add/subtract architectures save some silicon area.

In this paper, previous fused binary/decimal adders, focus on a carry-select correction (CSC) and carry-select sum-digit (CSS) technique are elaborated for area improvement and to further reduce costs and power dissipation.

Background information on previous speculative decimal add/subtract units and fused binary/decimal manipulation is offered in Section 2. Section 3 describes the key idea of modified CSC&CSS for area and power improvement and the Implementation details of the existing designs are provided in Section 3. The proposed method is described in Section 4. Finally, Section 5 draws our conclusions.

II. LITERATURE REVIEW

A. Conventional bcd adder

The parallel decimal arithmetic is becoming increasingly attractive with new applications of computers in a multiprogramming environment. The decimal addition offers significant improvement in addition, other methods requiring decimal correction. Decimal carries are represented by binary carry from the corresponding digit position or by a sum digit of ten or more. Correction is performed by adding six to each sum digit where carry is produced. The bit carries are still retained in decimal addition, when a binary adder is used with decimal correction. The carry look-ahead principle can be applied to a multi-digit it decimal adder.

The group carries generate and propagate signals can be formed for individual digits or for two-digit groups. For BCD adder the carry generates and carry propagate functions are previously defined which permits other groupings as well. When compared to 32-bit binary adder, the decimal adder requires about 18 percent more circuitry for the same number of logic levels.

B. Single correction speculation

SCS uses binary carry-save addition to compute intermediate results. This technique speculates BCD correction values and correct intermediate results while adding the input operands. It speculates over one addition. Speculation can be done by adding six along with each pair of digits from the original input operands. This makes the carry value 10, rather than 16. One of the main advantages

of this method is that the correct carries between digits are generated during the first addition, then the resulting sum digit needed to be corrected by subtracting six modulo 16 if the carry out is zero. When several operands are added together, the intermediate results are kept in binary carry-save format. This delays the carry-propagate addition until the end.

These techniques have linear delay, speculative BCD corrections and correct results based on carries from previous additions. This has an advantage of increased speed and less area consumption than previous techniques, but the delay gets increased.

C. Double correction speculation

DCS speculates over two additions. In this the first addition not need to be corrected. This technique selects between operand and speculated value, which is to be added to the sum and carry values. This removes the multiplexer in the critical path since the correction value is already selected while performing carry already selected while performing carry. Save addition to the sum and carry values. DCS technique also removes the logic needed to produce speculated value since the second operand is always added without a correction value.

One of the drawbacks in DCS is that determining the Speculation Correction value (SC) has been slightly more complex since two speculative additions needed to be corrected instead of one. It introduces linear delay, speculate BCD corrections and correct results based on carrier from previous additions. The speed gets increased and the area gets reduced in this technique.

D. Non-speculative addition

Non-Speculative Addition uses binary carry-save adder tree and produces the linear sum. They sum BCD input operands in a binary carry-save tree and passes carries generated along the way to the next more significant digit. This gives a preliminary binary sum term. The sum and carry-out terms are fed into combinational logic, which produces a decimal sum and carry corrections is needed. A tree of binary carry-save adders are used to add the input operands. Once all the input operands are added, combinational logic used to evaluate the sum corrections and the carry correction terms for the next more significant digit. These values are determined using the number of carries out of the current digit position and the preliminary sum digit. The sum and carry correction logic varies based on the number of input operands added.

This technique has logarithmic delay, adds the input operands using a tree of carry-save adders and produces a binary sum. They have an advantage of less delay than speculative adders with the cost of the same area and speed.

E. Conditional Speculative Fused Binary/Decimal adder

It performs addition and subtraction operation for both decimal and binary representations. This algorithm potentially leads to address with interesting area-delay figures. A conditional speculative adder is based on speculative addition which uses a conditional speculation to avoid a post-correction after carry evaluation. This leads to a lower dependency on the performance of the selected carry-tree topology. This gives the designer more flexibility to choose the adder architecture and area/latency trade-offs.

This method provides an efficient implementation of a binary/decimal combined unit using an existing binary parallel prefix adder and little additional hardware. This can also be implemented using a quaternary binary carry tree. This algorithm requires a minor post-correction. So that any carry-tree topology can be used without the delay penalty of other schemes based on post-correction.

III. DECIMAL ADDER WITH CSC & CSS BLOCK

A. Carry-select correction and carry-select sum-digits

A straightforward solution to put the black correction boxes of fast speculative fused binary/decimal adder off the critical delay path is described by the following equation

$$S_i = \begin{cases} (Z_i - 6D) & \text{if } c_{i+1} = 0 \text{ and } c_i = 0 \\ (Z_i - 6D)+1 & \text{if } c_{i+1} = 0 \text{ and } c_i = 1 \\ Z_i & \text{if } c_{i+1} = 1 \text{ and } c_i = 0 \\ Z_i + 1 & \text{if } c_{i+1} = 1 \text{ and } c_i = 1 \end{cases} \quad (1)$$

$$Z_i = X_i + (Y_i \boxplus A) + 6DA' \quad (2)$$

This four sub expressions are computed in parallel for $0 \le i \le k-1$. The interim sum-digit Z_i is given by Eq. (2) and c_{i+1} selects one of the corrected or intact interim sum digits. Decimal or hexadecimal generate and propagate signals are defined by Eq. set (3); they are computed at the same time as that of S_i. Decimal or hexadecimal generate and propagate signals are fed into a quaternary parallel prefix carry generation network that computes all c_is. These carry signals will select the correct sum-digits out of the above mentioned four interim sum values.

$$P_i = (Z_i \ge 15),\ G_i = (Z_i \ge 16) \quad (3)$$

The 4-way selection of sum term S_i can be improved via taking the selection by c_{i+1} off the critical delay path, as in [5]. But our modified carry-select method computes $c^0_{i+1} = G_i$, $c^1_{i+1} = G_i \vee P_i$, and the corresponding interim sum values, based on Eq. set (4), for $c_i=0$ and $c_i=1$, respectively. Then the final sum-digit S_i can be selected via the 2-way selection by c_i. This leads to reduced area, as compared to Carry Select box used in [6], via removing one 4-bit adder and one correction logic at the cost of adding one 4-bit incrementer.

$$S^0_i = Z_i - 6DG_i,\ S^1_i = Z_i - 6DG_iP_i + 1 \quad (4)$$

The expression for S^1_i is elaborated to get at Eq. (5), whose implementation can use the already available S^0_i, this will save more area. Note that despite of the additional delay, S^1_i computation remains off the critical delay path.

$$\begin{aligned} S^1_i &= Z_i - 6DG_i\overline{P_i} + \overline{1} = Zi - 6DG_i(1-P\overline{i}) + 1 \\ &= Z_i - 6DG_i + 6DG_iP_i \mp 1 \\ &= S^0_i + 1 + 6DG_iP_i \end{aligned} \quad (5)$$

B. Implementation details

This section describes the implementation details of combined binary/ decimal-add/subtract unit. Fig. 1 depicts the general architecture of combined binary/ decimal-add/subtract unit, which is based on carry-select correction of decimal interim sum-digits (CSC) and final carry-selection of sum-digits (CSS). The critical delay path is denoted by bold dashed lines. The implementation details of 4-bit digit preprocessor, CSC&CSS box, and Quaternary

Parallel Prefix Network (QPPN) shown in Fig. 1 are explained below.

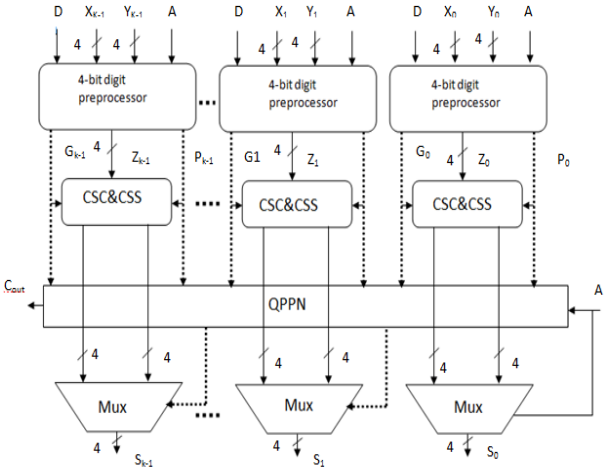

Fig. 1: Architecture of combined binary/decimal - add/subtract unit.

1) 4-Bit Digit Preprocessor

Let $X_i = x_3x_2x_1x_0$, $Y_i = y_3y_2y_1y_0$, $Y_i' = y_3'y_2'y_1'y_0' = (Yi \boxplus A) + 6DA'$ and $Z_i = z_3z_2z_1z_0$. Then Y_i' bits can be expressed in terms of Y_i's as in Eq. set (6). However, in previous adders, simpler expressions can be achieved. We use a 4-bit parallel prefix logic on the bits of X_i and Y_i' to compute Z_i, G_i, and P_i, as depicted by Fig. 2. White squares produce normal bit generate, propagate, and half-sum(h_o-h_3) signals, and black (gray) circles are normal (only g producing) parallel prefix nodes.

$y_0' = y_0$ xor A; $y_2' = y_2$ xor (A v Dy_1)
$y_1' = y_1$ xor (A v D);
$y_3' = (y_3$ xor A) v $D(y_2$ v $y_1)$ (6)

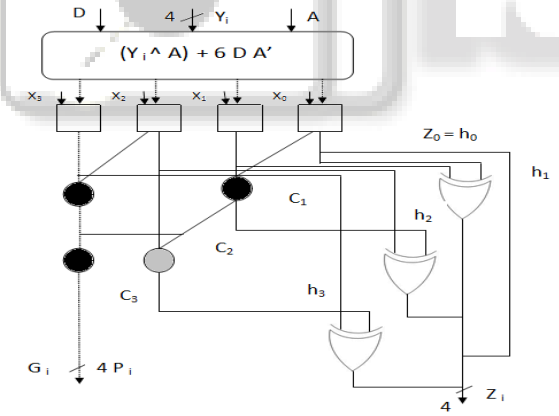

Fig. 2:Hardware realization of 4-bit digit preprocessor.

2) CSC&CSS Box As The Key Area Saving Component

Fig. 3 gives the logic of carry-select sum-digits, for c_i=0 and c_i=1 and the corresponding bit expressions for $Z_i' = Z_i - 6$ are as in Eq. set (7). S^0_i and S^1_i were defined by Eqs. (4) and (5), respectively. To avoid addition of the modified correction term $C = 6DG_iP_i$ in Eq. (5), note that C = 6 holds only when D = 1, Gi = 0 (i.e., $Z_i \leq 15$), and Pi=1 (i.e., $Z_i \geq 15$). This gives Z_i=15, which corresponds to S^0_i=9 or $S^1_i = \lfloor 16 \rfloor_{16} = 0$. The latter is realized in Fig. 3 by forcing non-zero bits of incrementer's output to zero.

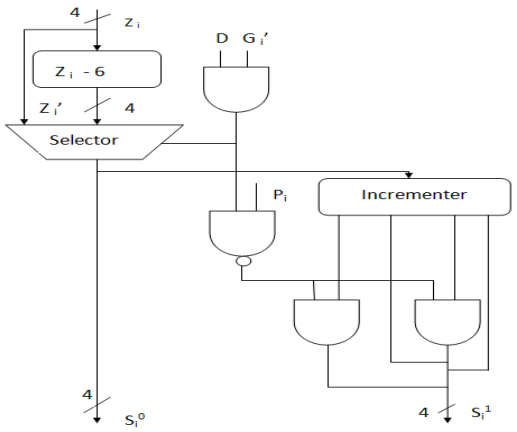

Fig. 3: Details of CSC&CSS box.

$Z_0' = Z0$; $Z_2' = Z_2$ xor Z_1
$Z_1' = Z1$; $Z_3' = Z_3 \wedge Z_2 \wedge Z_1$ (7)

The area consumed by this CSC&CSS scheme is obviously reduced, as compared to previous structures. Effective replacement of the two four bit adders by three XOR gates and four AND gates reduces the area consumed and it is used to derive half-sum signals form already produced p and g signals.

3) Quaternary Parallel Prefix Network (QPPN)

The decimal or hexadecimal carry-in signals can be produced via any realization of parallel prefix carry generation network, by recalling the generate (G_i) and propagate (P_i) signals for 4-bit digits, for both decimal and binary. A full Kogge-Stone (K–S) architecture shown in [5] with add/subtract control A as carry-in, for 16-digit decimal or 64-bit binary addition/subtraction. The triangles represent buffers and c_{out} is available at the same time as sum bits.

IV. PROPOSED WORK

In this section, a simple method is used to transform a prefix adder specification into a Ling adder for improving the speed. The introduction of Ling adder leads to the following changes in a conventional adder.

- The prefix operations are performed in pairs (g_j, a_{j-1}) instead of (g_j, a_j).
- Since the white circle operator is used, the prefix operations at the first level of the tree-like computation are simplified.
- The sum bit is obtained by the formula.

$$s_{i+1} = p_{i+1}h_{i+1} + (p_{i+1} \oplus a_i)\, h_{i+1}. \quad (8)$$

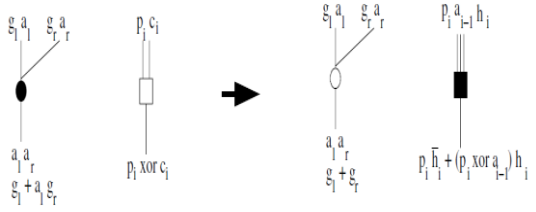

Fig. 4: Transformation of a prefix adder into a Ling adder.

The resulting Ling adder has the same topology, but with different logic in the first and last stages. The first stage of the carry-tree improves the speed, which requires a single gate level (NAND gate) instead of a complex gate (or-and-invert). Although the last stage of the Ling adder is more complex, the critical path contains a two input multiplexer which has a delay comparable to a two input xor gate.

V. RESULTS AND DISCUSSION

The proposed modified decimal adder with Ling Adder Circuit shown in the previous section is implemented and resulted from using VHDL and simulated in Xilinx ISE 9.1. For the designing of the modified decimal adder circuit, uses the structural and behavioral modeling. The VHDL codes for the gray circle and the white circle is written in behavioral modeling and are connected to each other by structural modeling.

A. Delay

Speed Parameters	Decimal adder with CSC & CSS block	Modified decimal adder with Ling adder
Minimum period(ns)	4.269	2.592
Minimum input arrival time before clock(ns)	6.657	5.670
Maximum output required time after clock(ns)	10.627	7.351
Maximum combinational path delay(ns)	12.493	9.034

Table 1: Comparison of Speed Parameters

A comparison of delay for both existing decimal adder circuit and modified decimal adder circuit is shown in the table 1. Which results, the total delay of modified decimal adder is 35% reduced compared to the existing decimal adder circuit.

B. Power

A comparison of power for both existing decimal adder circuit and modified decimal adder circuit is shown in the table 2.

Energy Parameters	Decimal adder with CSC & CSS block	Modified decimal adder with Ling adder
Dynamic Power(mW)	9.59	6.73
Total Power(mW)	59	56

Table 2: Comparison of Power Parameters

C. Area

Device Utilization Parameters (Spartan 3)	Decimal adder with CSC & CSS block	Modified decimal adder with Ling adder
Number of 4 I/p LUT's	34/4896	25/4896
Number of slices	27/2448	14/2448
Number of bounded IOBs	19/158	15/158

Table 3: Design Summary

Design summary of existing decimal adder circuit and modified decimal adder circuit is shown in the table 3. The design summary is also shows the there is a saving of 26% in the 4 I/p LUTs and 50% in the number of slices with the 20% in the bounded IOBs. Which shows that the proposed design have high speed and area efficient.

VI. CONCLUSION

In this paper, a new carry-select correction and carry-select sum-digit (CSC&CSS) logic is used in decimal adders, fused binary/decimal adders, and the corresponding unified add/sub- tract units. Such low overhead combinations and unifications are commonly desired in practical ALU designs. Ling adder concept is introduced in the prefix network for speed improvement. With the addition of Carry Select Correction and Carry Select Sum block the area and power of the adder is reduced an hence the new architecture will have less area and power consumption and the speed will also increase.

REFERENCES

[1] Anderson M. and Tsen C. "Performance analysis of decimal floating-point libraries and its impact on decimal hard- ware and software solutions", Proceedings of IEEE International Conference on Computer Design (ICCD), Vol. 45, No. 5, pp. 465–471, 2009.
[2] Anderson M. and Mathew S.K. "A 4-GHz 300-mW 64-bit integer execution ALU with dual supply voltages in 90-nm CMOS", IEEE J. Solid-State Circuits Vol. 40, No. 1, pp. 44–51, 2005.
[3] Chen D. and Han L. "Improved decimal floating-point logarithmic converter based on selection by rounding", IEEE Trans. Comput. Vol. 61, No.5, pp. 607–621, 2012.
[4] Cowlishaw M.F. "Decimal floating-point: algorism for computers", in: Proceedings of the 16th IEEE Symposium on Computer Arithmetic (ARITH '03)Vol. 12, No. 1, pp. 104–111, 2003.
[5] Ghassem Jaberipur and Morteza Dorrigiv, "Low area/power decimal addition with carry-select correction and carry-select sum-digits", Elsevier Journal Vol. 47, No. 1, pp. 443–451, 2014.
[6] Kenney R.D. and Schulte M.J. "High-speed multi operand decimal adders", IEEE Trans. Comput. Vol. 54, No. 8, pp. 953–963, 2005.
[7] Schmookler M. and Weinberger A. "High speed decimal addition", IEEE Trans. Comput. Vol. 20, No. 8, pp. 862–866, 1971.
[8] Vázquez A. and Antelo E. "Conditional speculative decimal addition", in: Proceedings of the Seventh Conference on Real Numbers and Computers (RNC7), pp. 47-57, 2006.
[9] Vázquez A. "High-Performance Decimal Floating-Point Units", Ph.D. Dissertation, Univ Santiago de Compostela, 2009.
[10] R.D. Kenney, M.J. Schulte "High-speed multioperand decimal adders", IEEE Trans. Comput. Vol. 54, No. 8, pp. 953–963, 2005.
[11] S.K. Mathew, M. Anders, B. Bloechel, T. Nguyen, R. Krishnamurthy, S. Borkar, "A 4-GHz 300-mW 64-bit integer execution ALU with dual supply voltages in 90-nm CMOS", IEEE J. Solid-State Circuits Vol. 40, No. 1, pp. 44–51, 2005.

