The CSI Journal on
Computer Science and Engineering
Vol. 13, No. 2, 2016
Pages 32-41
Regular Paper

Exploring Reconfigurability Options Among Decimal Adders

Samaneh Emami Mehdi Sedighi

Computer Engineering and Information Technology Department, Amirkabir University of Technology, Tehran, Iran

Abstract

Decimal arithmetic has become a hot research topic in recent years. Many hardware units have been designed and proposed to perform high performance and accurate decimal arithmetic operations. Traditionally, decimal arithmetic units have been designed as application-specific specialized hardware modules. But there is an emerging trend towards the design and implementation of reconfigurable structures to perform decimal arithmetic. This paper contributes to this trend by exploring different reconfigurability options in decimal adders, proposing new reconfigurable parallel prefix trees (PPTs), and presenting a reconfigurable combined binary/decimal adder with a variable input width. Our analysis shows that it is possible to combine two conventional PPTs to reach a reconfigurable version with a reasonable overhead. Furthermore, we will suggest two criteria for choosing which PPTs to combine and will compare these two criteria. Experimental results demonstrate that the reconfigurability in the proposed designs comes at the cost of at most 5% overhead in area.

Keywords: Decimal Arithmetic, Parallel Prefix Tree, Decimal Adder, Reconfigurable Hardware, Granularity.

1. Introduction

Human beings have traditionally used decimal arithmetic probably because of having ten fingers. As such, using decimal arithmetic appeared to be the natural choice for the first generation of computers such as ENIAC [1], UNIVAC [2], and IBM 650 [3]. Over time, however, the decimal arithmetic modules were gradually replaced by their easier-to-implement binary counterparts to the extent that binary arithmetic became the prevailing choice for designers of computer systems. But there is growing evidence that decimal arithmetic is emerging again in the modern computers. For instance, modern processors like IBM Power and System z processors [4-7] have specialized decimal hardware units.

One of the main reasons for the resurgence of decimal arithmetic is its accurate representation of decimal floating point (DFP) numbers. In contrast, a major problem of binary arithmetic is its inaccuracy in representing some non-integer numbers such as 0.1 that can cause unacceptable errors in financial and commercial applications. For instance, it has been reported that in a large telephone billing system, using binary arithmetic instead of decimal can result in an estimated annual loss of up to five million dollars [8]. It is safe to say that the lower precision of binary arithmetic in certain applications has become an inhibiting factor in modern computers that handle those applications.

A good evidence for the increasing importance of decimal arithmetic in recent years is that the IEEE 754-2008 standard for floating-point arithmetic [9] includes specifications for handling decimal floating-point numbers. In this standard, two decimal number formats for both software and hardware implementations of decimal arithmetic are presented:

Binary Integer Decimal (BID) is proposed for software implementations of decimal arithmetic and is used in IBM dec Floats modules [10], Intel DFP Math Library [11], and built-in GCC DFP types [12]. These implementations can eliminate the inaccurate representation errors, but they are usually slow and inefficient [13].

Densely Packed Decimal (DPD) is recommended for hardware implementations of decimal arithmetic and is used

in machines that have dedicated decimal hardware units, like IBM Power and System z processors [4-7]. These implementations usually provide both accuracy and performance [14]. Additionally, the dramatic increase in chip density has lowered the overall cost of hardware implementations. Consequently, hardware solutions are gaining prominence in industry [15].

There is also a recent trend toward the design and implementation of reconfigurable arithmetic hardware units. Traditionally, the reconfigurable platforms have been fine-grain modules found in commercial Field Programmable Gate Arrays (FPGA). Fine-grain modules can address bit-level granularity, but generally suffer from high reconfiguration overheads. The ultimate design goal would be to achieve an ASIC-like performance and FPGA-like flexibility, design time and cost. Therefore, the coarse-grain reconfigurable units appear to be a promising compromise between ASIC and FPGA.

While there are numerous studies on various hardware implementations of decimal arithmetic operations [16-21], there are only a few works that focus on the reconfigurebility aspects of designing such hardware. This paper explores different reconfigurability options and possibilities in decimal adders. To do so, the idea of combining existing conventional Parallel Prefix Trees (PPTs) will be considered first. As will be elaborated in detail, it is possible to combine two conventional PPTs to reach a reconfigurable version with a reasonable overhead. In doing so, one has to choose the original PPTs carefully in order to achieve low latency and area overheads. As such, we will suggest two criteria for choosing which PPTs to combine and will compare these two criteria. Some new reconfigurable PPTs will be introduced based on these criteria. Another aspect of reconfigurability can be in the form of varying input size and type. So a reconfigurable combined binary/decimal adder with a variable input width will be justified and presented subsequently.

The remainder of this paper is organized as follows: Section 2 provides a review of the literature in this field. Section 3 discusses the proposed reconfigurable architectures in detail. Synthesis results and their analysis are provided in Section 4. The paper is concluded in Section 5.

2. Prior Works

There are numerous studies on various implementations of basic arithmetic operations such as addition, subtraction, multiplication, and division. Among these studies, some focus on optimized but inflexible decimal hardware implementations [19-21] and some present decimal arithmetic units with some degree of flexibility [22-27]. For instance, some researchers have focused on efficient implementations of decimal arithmetic operations on reconfigurable architectures such as FPGAs. This is because the FPGA implementation can provide an added flexibility in terms of compliance with various standards and the desired objective that the implemented design is trying to reach. Nannarelli [22], for example, has studied FPGA-based acceleration of decimal arithmetic operations.

This study shows that applications requiring decimal operations can be expedited by an arithmetic processor implemented on an FPGA board that connects to a computer. This processor ran a telephone billing application and achieved a speed-up of around 10 over its execution on the CPU of the host computer. This achievement is mainly due to more flexibility in the FPGA implementation with respect to ASIC design. Vazquez and Dine chin [23] have also presented a new method for fast implementation of multi-operand decimal addition in current FPGAs. This method is based on pre- and post-corrections of the binary sum. With the pre-corrections, the hexadecimal carries correctly serve as decimal carries, which brings about the opportunity to utilize built-in carry chain in FPGAs. As a result, the authors have reported that their implementation on a Virtex-6 FPGA device halves the area and latency of previous reconfigurable implementations [24].

There are other works that propose reconfigurable architectures for efficient implementation of decimal arithmetic. Such architectures may have an inherent flexibility in terms of the input format and width, or operands' radices and implementation. Combined binary/decimal arithmetic circuits can also be included in this category. For example, Calderon et al. [25] have proposed a new adder/sub tractor unit. Their implementation can operate on sign-magnitude, unsigned, and different complement representations. Another similar work [26] presented a novel combined adder/sub tractor arithmetic unit for binary, BCD (Binary Coded Decimal), and single precision BFP (Binary Floating Point) representations.

Another example for a fully reconfigurable architecture specialized for efficient implementation of binary arithmetic can be found in [27]. This work employs Coarse Grain Reconfigurable Architectures (CGRAs) which are a compromise between ASICs and FPGAs since they provide better computational efficiency compared to FPGAs and better engineering efficiency compared to ASICs. The CGRA fabric introduced, called Dynamically Reconfigurable Resource Array (DRRA), is a parallel digital signal processing fabric with distributed arithmetic, logic, interconnect and control resources.

Overall, one can safely say that reconfigurability in arithmetic circuits may take different forms and shapes. The previous works have considered some of them. But there is still room to explore other aspects of reconfigurability such as hybrid adders that implement more than one Parallel Prefix Tree (PPT) depending on their configuration, or a combination of binary and decimal addition with variable width. These ideas will be discussed in the next section.

3. Proposed Architectures

Addition is a very basic arithmetic operation. Fast adders are needed in virtually any digital system. They are also necessary in other arithmetic operations like multiplication. The speed of an adder is usually governed by its ability to quickly handle the carry chain. A common structure for fast computation of the carry signal is called a Parallel Prefix Tree (PPT). Six conventional PPTs have been introduced in the past [28-33]. These PPTs differ in terms of critical path delay, area, fan-out, wiring tracks, and number of levels (or tree depth). That is why each one may be appropriate in some applications and inappropriate in others. However, there are applications in which two or more of these

conventional PPTs might be helpful or necessary. In such applications it will be beneficial to have a hybrid reconfigurable PPT that provides the advantages of more than one PPT but at the cost of one single hardware. This idea will be discussed later in this paper.

In all binary PPTs, the inputs are calculated from $p_i(= x_i + y_i)$ and $g_i(= x_i . y_i)$, where $x_i$and y_i are two corresponding bits of addend and augend. Also, every node of the tree presents carry propagation (P) and carry generation (G) signals. The implementation of each node in the PPTs is shown in figure 1. At the lowest level of trees, carry of each position is ready to be added to the partial sum in order to produce the final result.

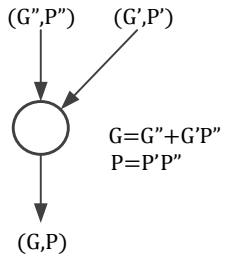


Figure 1. Implementation of a node in decimal PPTs

The implementation of a decimal PPT is similar to its binary counterpart, but its inputs, called P_i and G_i, are calculated using the following equations, where p_i^j and g_i^j are the corresponding binary propagate and generate signals:

$$P_i = p_i^3 p_i^2 p_i^1 p_i^0$$
$$G_i = g_i^3 + g_i^2 p_i^3 + g_i^1 p_i^2 p_i^3 + g_i^0 p_i^1 p_i^2 p_i^3 \quad (1)$$

The notion of reconfigurability in adders can take many different forms. In the next subsections, two of these various forms and possibilities will be explored: the prospect of combining two existing PPTs to form a new hybrid PPT that can be reconfigured into either of its two parent PPTs whereby providing their advantages albeit with some overhead; and the possibility of having a binary/decimal adder with a configurable input width.

3.1. Proposed Reconfigurable PPTs

The proposed reconfigurable PPT in this section can implement two different types of conventional PPTs such as Brent-Kung and Han-Carlson. The first question in designing such a hybrid PPT, is which two PPTs may be combined efficiently well to compensate for the inevitable reconfiguration cost. The preliminary answer to this question might be that trees with equal number of nodes appear to be suitable candidates. To explore this point, Ladner-Fischer (L-F) and Han-Carlson (H-C) trees were considered.

The implementations of L-F and H-C PPTs for 16-bit inputs with four and five levels are shown in figures 2 and 3, respectively. The gray nodes in both figures have the same inputs. These nodes can be shared easily in the hybrid PPT. The white nodes (in figure 2) and black nodes (in figure 3) have differing nodes in the two PPTs.

As the first step, a combined PPT with the same number of nodes as the original L-F and H-C trees was formed. This tree is depicted in figure 4. As this figure shows, the gray nodes require no changes in the data path. However, the white and black nodes need multiplexers to configure their varying data path according to the desired configuration. Needless to say, the multiplexers' selector signals determine which PPT is selected but for simplicity, they are not shown in the figure. Another important point in this figure is that the number of levels in the combined PPT is the same as the PPT with the higher number of levels. In this case, since H-C tree has 5 levels, the combined version also has 5 levels.

Multiplexers usually have considerable area and delay overheads. If they are implemented using complex gates (as opposed to transmission gates), their area and delay can even be more than a cell in the tree. So it seems plausible to assume that if redundant nodes are used instead of a shared node and multiplexer(s), the overall delay and area overhead would be smaller. As such, in the second step, the common nodes with same inputs (gray) remain intact. But the differing nodes are placed twice in the reconfigurable design. The resulting structure is shown in figure 5. In this figure, the gray nodes are shared between two PPTs whereas white and black nodes correspond to L-F and H-C PPT's, respectively. Notice that the redundant nodes are marked by the same numbers and only one of them is activated in each configuration.

The usage of redundant nodes in the PPT shown in figure 5 eliminates the need for the multiplexers whereby improving the area and delay of the reconfigurable PPT, as will be shown in Section 4. However, to explore the effect of structural matching between the original PPTs on the characteristics of the reconfigurable PPT, the first criterion used in selecting the basic trees (same number of nodes) was changed to structural similarity in the structures of chosen PPTs. The structural similarity was defined as the number of common (gray) nodes that can be found in the two PPTs. A careful examination of Han-Carlson and Brent-Kung (B-K) reveals that for 16-bit inputs, H-C has 32 nodes in five levels and B-K has 26 nodes in six levels. Figure 6 illustrates the implementation of B-K PPT.

In order to remain consistent with H-C node numbers, the nodes are not enumerated sequentially in this figure. Even though the total number of nodes in the PPTs are different, but there are 18 common nodes between H-C and B-K. So, these two were identified as similar structures. The common nodes were shared in the combined PPT. As for the nodes with different inputs, if only one input of a node is different (e.g., node 19), a MUX is used to steer the logic properly. But if both inputs of corresponding nodes are different (e.g., node 26), then instead of sharing the node and using MUX is, both nodes are kept. Figure 7 shows the resulting reconfigurable PPT that can implement H-C PPT (with gray and black nodes) or B-K PPT (with gray and white nodes).

As will be discussed in Section 4, the experimental results show that this approach provides superior results compared to the other approaches depicted in figures 4 and 5. Therefore, one may conclude that if a hybrid reconfigurable PPT is desired, then choosing the PPTs based on their structural similarity leads to better designs with smaller area and delay overheads compared to just considering equal number of nodes in them. To explore another kind of reconfigurability, a reconfigurable adder that takes operands of different types and sizes is introduced next.

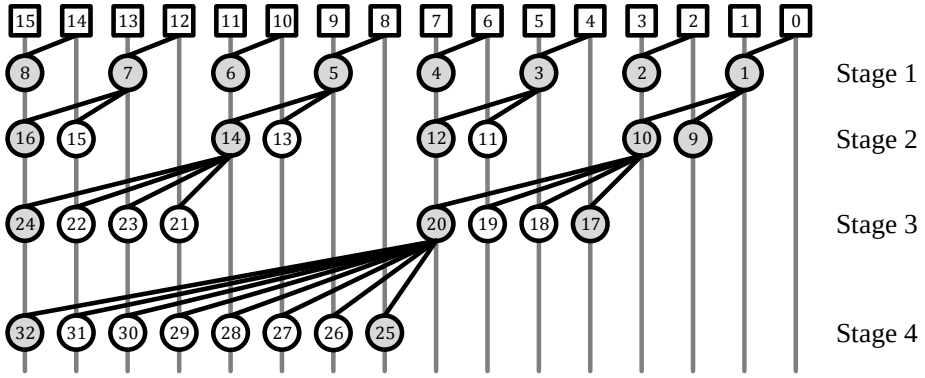


Figure 2. L-F Parallel Prefix Tree (reproduced from [33])

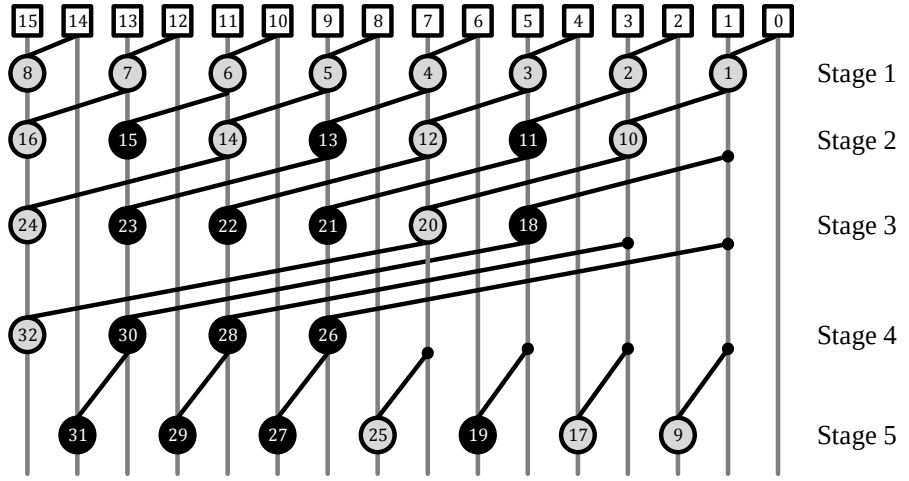


Figure 3. H-C Parallel Prefix Tree (reproduced from [31])

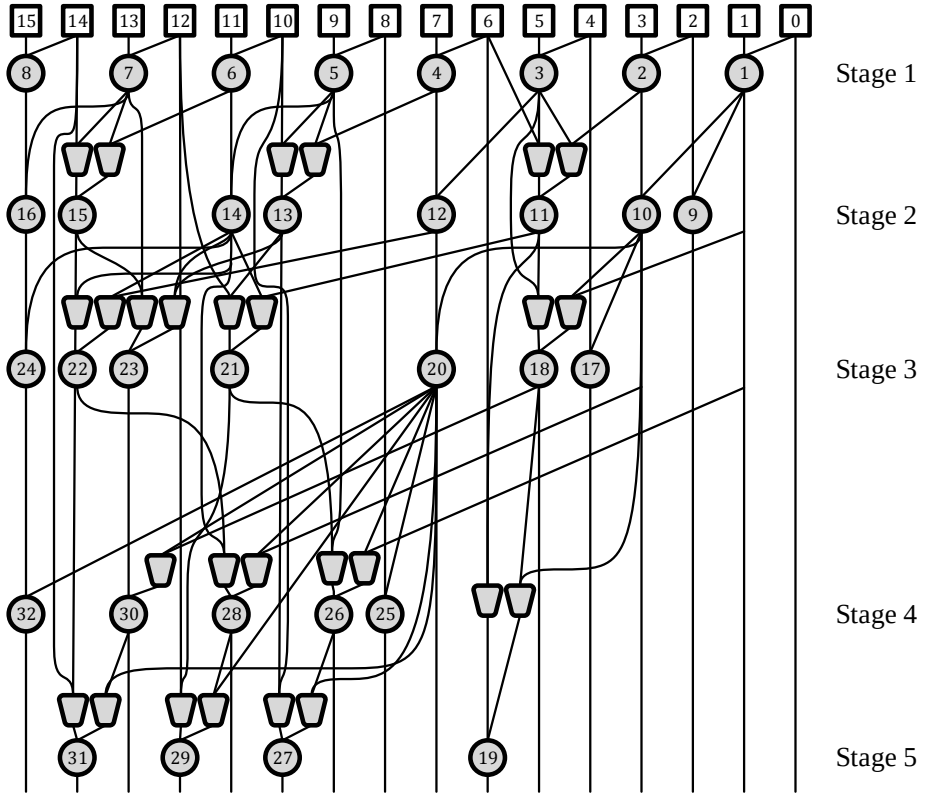


Figure 4. Reconfigurable PPT (Combined L-F/H-C), Step 1

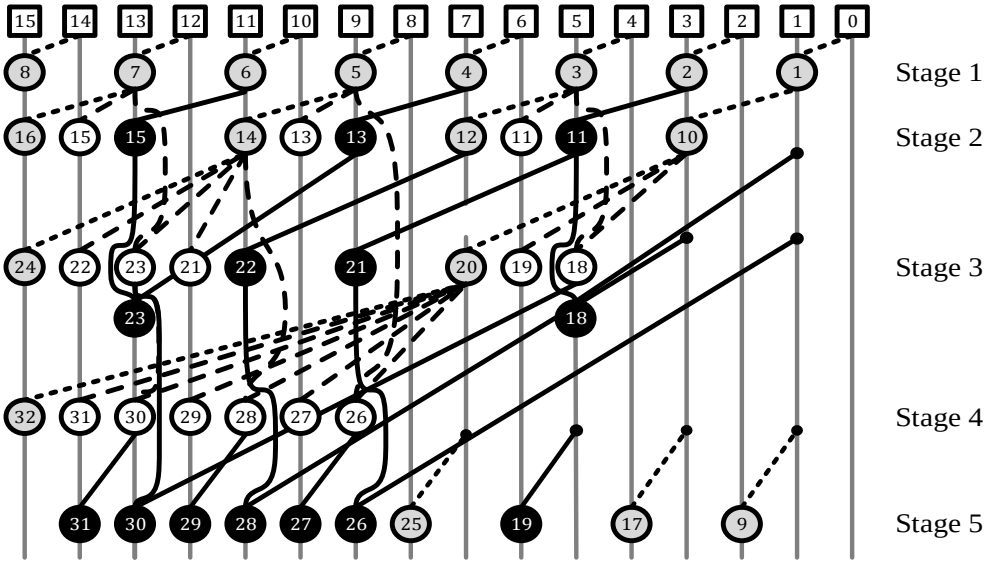


Figure 5. Reconfigurable PPT (Combined L-F/H-C), Step 2

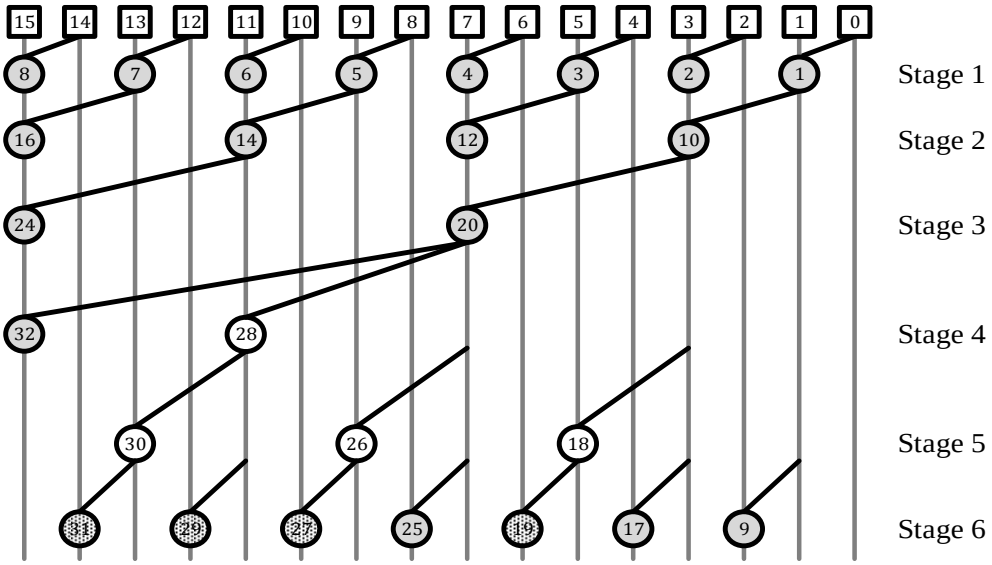


Figure 6. B-K Parallel Prefix Tree (reproduced from [28])

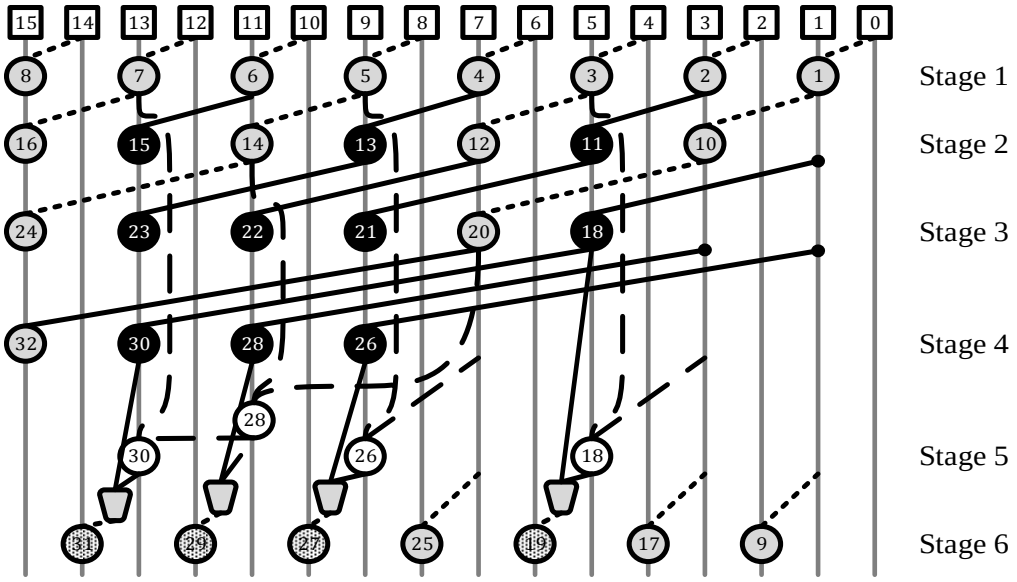


Figure 7. Reconfigurable PPT (Combined H-C/B-K), Step 3

3.2. Proposed Reconfigurable Adder

Modern processors are generally designed with a wide (like 64 bits) data path. A wide data path usually necessitates a wide ALU as well. For instance, a 64-bit processor requires a 64-bit adder. Research has shown that there are many applications in which a sizeable portion of operations are executed on shorter data structures (such as Smallint) as compared to longer ones (like Long) supported by the hardware [34]. The statistics found in literature indicates that 60% of binary operations contained in various applications such as airline systems, banking, financial analysis, insurance, etc. use 16-bit Smallint variables and only the remaining 40% of the operations need 32-bit Integers even though the processors still have to provide support for 32-bit operations [34].

In those circumstances, a wide adder is not always needed and having the option to configure the wide adder as two narrower adders that can work in parallel can offer a potentially considerable improvement in terms of throughput and/or performance. Furthermore, if equipped with power gating mechanism, the upper half of a wide adder can be switched off in case only the lower half is used whereby providing a power saving potential.

There are also many reported scenarios in which the input of an arithmetic unit might alternate between binary and decimal. For instance, in applications that provide scientific and financial services to their customers, the service providers often use binary arithmetic for the former and decimal for the latter group of services using the same (cloud) hardware infrastructure [35]. If only single-type arithmetic units are available, designers will have to use both binary and decimal hardware modules simultaneously in their implementations. A reconfigurable binary/decimal adder in these cases will certainly be beneficial.

Given these conditions, a new reconfigurable adder is proposed in this paper. The proposed adder, shown in figure 8, is a combined binary/decimal adder that can operate on different sizes of operands. In the proposed design, X_i and Y_i represent the binary or BCD digits of two input operands. A signal "$\mathcal{D}$" indicates that inputs are binary ($\mathcal{D} = 0$) or decimal ($\mathcal{D} = 1$). The design has two distinct parts that can operate independently or together. Each part can be configured as either a 4n-bit binary adder or an n-digit decimal adder. If configured to perform the addition in a cascaded form, the proposed adder will take the shape of either an 8n-bit binary adder or a 2n-digit decimal adder. Each part of the design is n comprised of "dual sum generation" units. The internal circuitry of these units is illustrated in figure 9 (reproduced from [36]). In this figure, the squares produce the bit generate, propagate, and half-sum ($\hbar_0$-$\hbar_3$) signals and the circles are ordinary parallel prefix nodes. The dual sum generation units compute P_i and G_i for each decimal digit that is fed into a quaternary PPT.

Each dual sum generation unit, also provides the result of its corresponding digit addition in two conditions: If the carry-in to that position is zero, the result is called S_i^0 and if the carry-in is one, the result is called S_i^1. In the lower half adder (the right hand side one), when quaternary PPT prepares correct decimal carries, the proper sum digit is selected. However, in the left hand side adder, the problem is slightly different. An "integrated" signal is defined that indicates the size of operation. If this signal is equal to one, two adders are cascaded to perform a 2n-digit decimal or 8n-bit binary addition. Otherwise, they implement two independent additions. The "carry selection" units receive the carry-out of right hand side adder and produce the correct control signal for selecting the appropriate sum. figure 10 shows the block diagram of the carry selection unit. In this figure, $P_i' = P_{i-1}P_{i-2} \dots P_n$, where $n < i < 2n$.

The proposed reconfigurable adder uses a Carry Select Adder (CSA) as its basis. This basic adder was chosen because it has fast implementations in literature. Depending on the design objectives that one pursues, other types of adder (such as Carry Skip Adder, Carry Look ahead Adder, etc.) could also be used in the proposed architecture.

The next section provides the experimental results and their analysis.

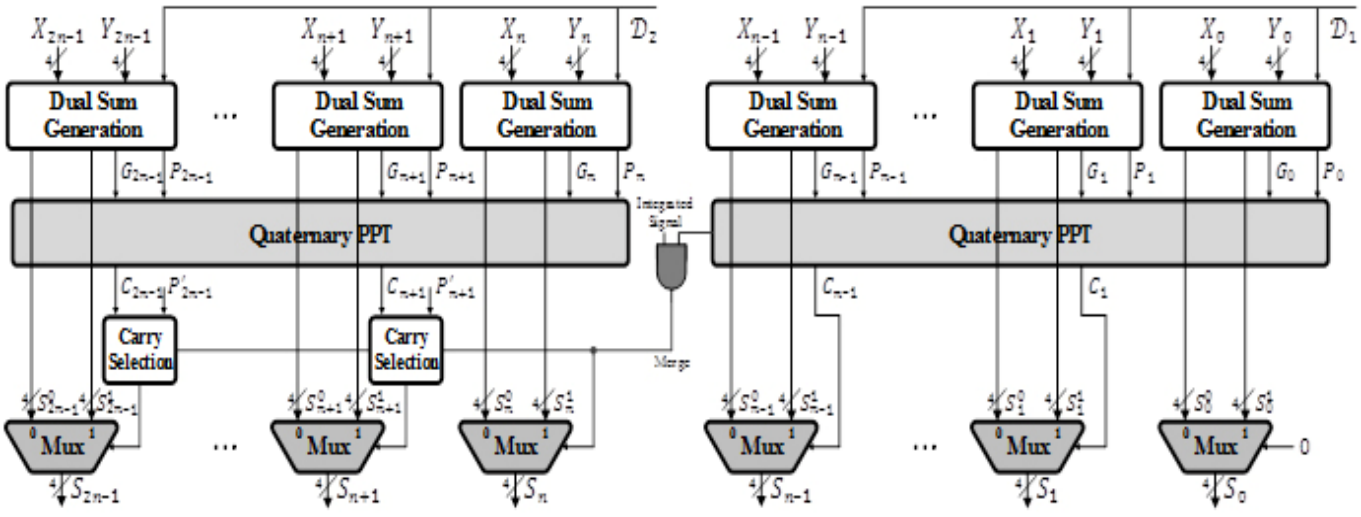


Figure 8. A reconfigurable combined binary/decimal adder

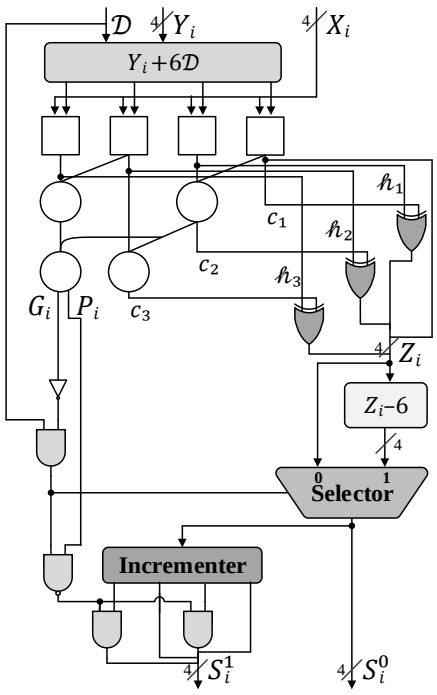


Figure 9. Dual sum generation unit structure (reproduced from [36])

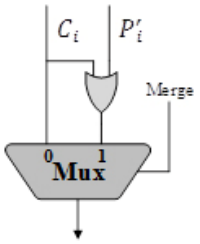


Figure 10. Carry selection unit structure

4. Comparison

To analyze the characteristics of the proposed reconfigurable designs, they were implemented at RT level using VHDL. The codes were validated using Modelsim. The area and critical path delay (both in terms of number of gates) of each design are listed in table 1. The column "Reconfigurability Overhead (Area)" shows the ratio of the corresponding reconfigurable PPT divided by the average area of its basic PPTs. Likewise, the column "Reconfigurability Overhead (Delay)" contains the ratio of the corresponding reconfigurable PPT delay divided by the average of its parent PPTs. As can be seen in the table, considering the structural similarity (step 3) produces the circuit with the smallest area with 28% average area overhead. But the minimum critical path delay and average delay overhead of 10% is obtained when equal number of nodes in the basic PPTs is considered (step 2).

Table 1. Comparison between basic PPTs and the proposed reconfigurable PPTs (for 16-bit wide inputs)

Parallel Prefix Tree		Area (Number of gates)	Reconfigurability Overhead (Area)	Critical Path Delay (Number of gates)	Reconfigurability Overhead (Delay)
Basic	Ladner-Fischer (L-F)	128	N/A	9	N/A
	Han-Carlson (H-C)	128	N/A	11	N/A
	Brent-Kung (B-K)	110	N/A	13	N/A
Reconfigurable	Combined L-F/H-C (Step 1)	209	1.63	19	1.9
	Combined L-F/H-C (Step 2)	170	1.33	11	1.1
	Combined H-C/B-K (Step 3)	152	1.28	15	1.25

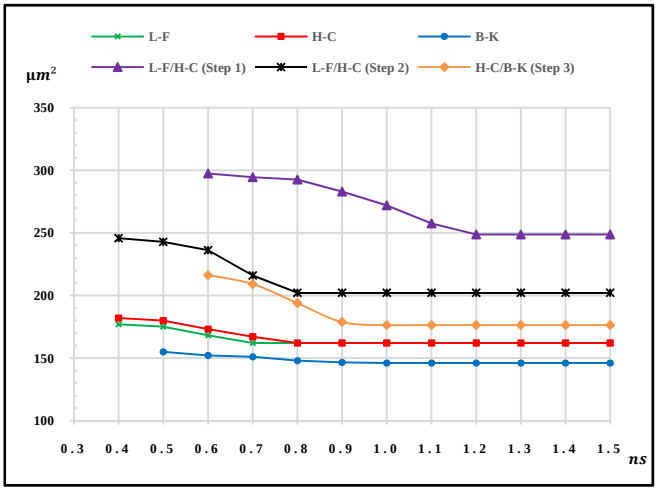


Figure 11. Area of 16-bit binary adders

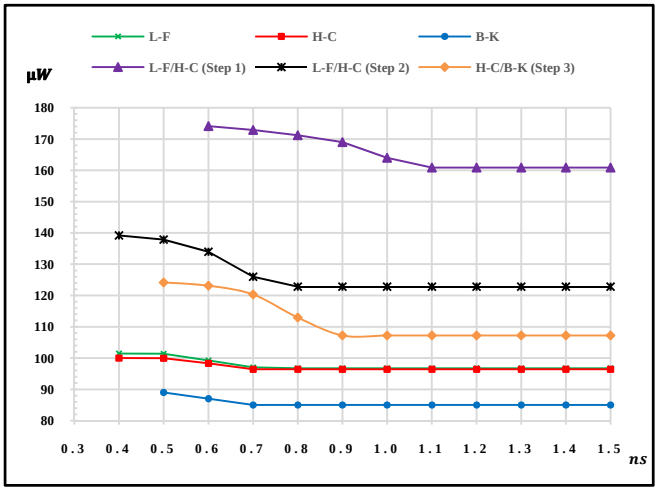


Figure 12. Power of 16-bit binary adders

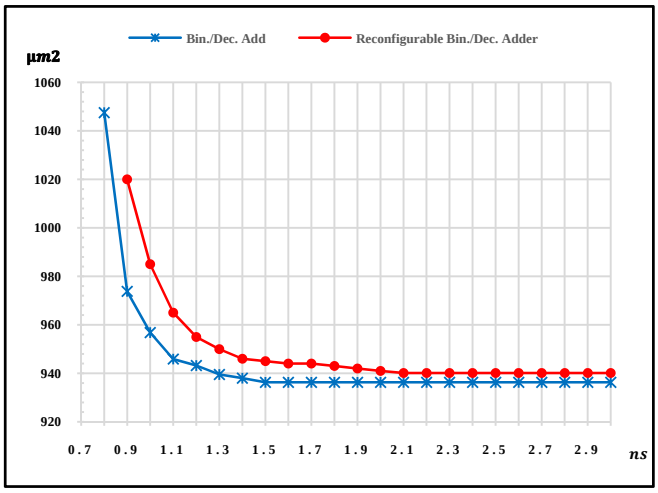


Figure 13. Area of 16-digit/64-bit binary/decimal adders

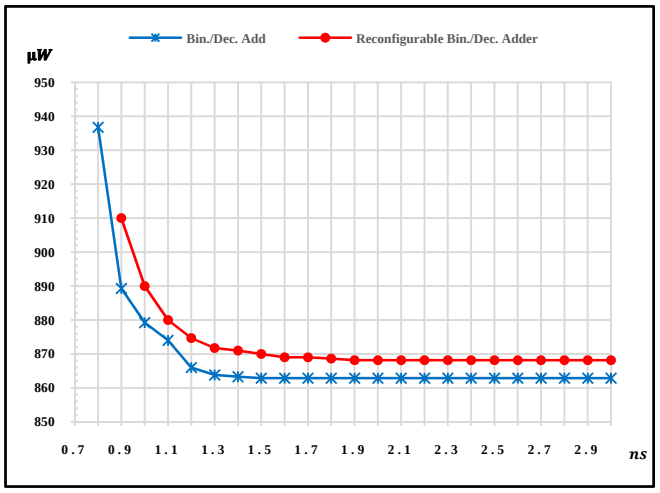


Figure 14. Power of 16-digit/64-bit binary/decimal adders

For more accurate analysis, the designs were also synthesized based on Nan Gate FreePDK45 Open Cell Library under typical process and normal operating conditions using Synopsys Design Compiler tool. The reconfigurable PPTs were synthesized with multiple target delays as shown on the horizontal axes of figures 11 and 12. These vertical axes in these figures represent the resulting area and power, respectively. The target delays are in the range of 0.3ns to 1.5ns with 0.1ns increments. If there is no point in the curves for a specific target delay, it means that no circuit could be found to satisfy the prescribed target delay. The curves demonstrate a general congruence with table 1.

Figures 13 and 14 depict the synthesis results of combined binary/decimal reconfigurable adder for area and power, respectively. To comply with IEEE 754-2008 standard [9], the adders are assumed to be 16-digit wide (when decimal) and 64-bit wide (when binary). The horizontal and vertical axes are similar to the ones in figures 11 and 12 except that the target delay range has been modified to 0.7-3ns in order to accommodate for the larger design in this case. The results are compared with the best combined binary/decimal adder found in literature [36]. As figure 13 shows, the area overhead of the proposed reconfigurable adder is at most 5%. Notwithstanding the worst case scenario, the average area overhead is far less than that.

5. Conclusion

In this paper, different options for reconfigurability in decimal adders were discussed. To this end, the possibility of combining two conventional PPTs in order to reach a hybrid reconfigurable PPT was explored. A few different criteria for choosing which two PPTs should be combined were considered and their effect on the resulting reconfigurable PPT was discussed.

In a different perspective, the concept of reconfigurability was extended to having an adder with flexible input size and type. The proposed reconfigurable adder can implement one 2n-digit or two n-digit binary/decimal additions concurrently. The synthesis results showed that for the same target delay, the advantages of reconfigurability in the proposed adder were reached with a maximum area overhead of 5% and average area overhead of 1%.

References

[1] H. Goldstine, and A. Goldstine, "The Electronic Numerical Integrator and Computer (ENIAC)," *IEEE Ann. Hist. Comput.*, vol. 18, no. 1, pp. 10–16, 1996.

[2] G. Gray, "UNIVAC I Instruction Set," *Unisys History Newslett.*, vol. 5, no. 3, 2001.

[3] F. E. Hamilton, and E. C. Kubie, "The IBM magnetic drum calculator type 650," *J. ACM*, vol. 1, no. 1, pp. 13–20, Jan. 1954.

[4] L. Eisen, J. W. Ward III, H. W. Tast, N. Mading, J. Leenstra, S. M. Mueller, C. Jacobi, J. Preiss, E. M. Schwarz, and S. R. Carlough, "IBM POWER6 Accelerators: VMX and DFU," *IBM Journal of Research and Development*, vol. 51, no. 6, pp. 663–684, 2007.

[5] A. Y. Duale, M. H. Decker, H. G. Zipperer, M. Aharoni, and T. J. Bohizic, "Decimal Floating-point in z9: An Implementation and Testing Perspective," *IBM Journal of Research and Development*, vol. 51, no. 1/2, pp. 217–228, 2007.

[6] E. M. Schwarz, J. S. Kapernick, and M. F. Cowlishaw, "Decimal Floating-point Support on the IBM System z10 Processor," *IBM Journal of Research and Development*, vol. 53, no. 1, pp. 4:1–4:10, 2009.

[7] S. Carlough, A. Collura, S. Mueller, and M. Kroener, "The IBM zEnterprise-196 Decimal Floating-Point Accelerator," *Proceedings of the 20th IEEE Symposium on Computer Arithmetic*, pp. 139-146, 2011.

[8] IBM Corporation, The "telco" Benchmark, http://speleotrove.com/decimal/telcoSpec.html,October 2016.

[9] Standards Committee, "754-2008 IEEE Standard for Floating-Point Arithmetic," IEEE Computer Society Standard, pp. 1–58, 2008.

[10] N. Chainani, "Decfloat: The Data Type of the Future," http://www.ibm.com/developerworks/data/library/techarticle/dm-0801chainani, January 2008.

[11] M. Cornea, "Intel decimal floating-point math library," https://software.intel.com/en-us/articles/intel-decimal-floating-point-math-library, October 2016.

[12] GCC, the GNU Compiler Collection, https://gcc.gnu.org, October 2016.

[13] M. Anderson, C. Tsen, L. K. Wang, K. Compton, and M. J. Schulte, "Performance Analysis of Decimal Floating-Point Libraries and its Impact on Decimal Hardware and Software Solutions," *Proceedings of the 27th IEEE International Conference on Computer Design*, pp. 465–471, 2009.

[14] H. Fahmy, R. Raafat, A. M. Abdel-Majeed, R. Samy, T. ElDeeb, and A. Farouk, "Energy and Delay Improvement via Decimal Units," *Panel on Decimal Arithmetic in Industry, Proceedings of the 19th IEEE Symposium on Computer Arithmetic*, pp. 221–224, 2009.

[15] L. K. Wang, M. A. Erle, C. Tsen, E. M. Schwarz, and M.J. Schulte, "A Survey of Hardware Desgins for Decimal Arithemetic," *IBM Journal of Research and Development*, vol. 54, no. 2, pp. 8:1–8:15, 2010.

[16] Á. Vázquez, and E. Antelo, "A High-performance Significand BCD Adder with IEEE 754-2008 Decimal Rounding," *proceedings of the 19th IEEE Symposium on Computer Arithmetic*, pp. 135-144, 2009.

[17] Á. Vázquez, *High-performance Decimal Floating-point Units*, Ph.D. dissertation, University of Santiago de Compostela, Santiago de Compostela, Spain, 2009.

[18] R. D. Kenney, and M. J. Schulte, "High-speed Multioperand Decimal Adders," *IEEE Transactions on Computers*, vol. 54, no. 8, pp. 953–963, 2005.

[19] Á. Vázquez, E. Antelo, and P. Montuschi, "A New Family of High-performance Parallel Decimal Multipliers," *Proceedings of the 18th IEEE Symposium on Computer Arithmetic*, pp. 195–204, 2007.

[20] D. Chen, L. Han, Y. Choi, and S. B. Ko, "ImprovedDecimal Floating-point Logarithmic Converter based on Selection by Rounding," *IEEE Transactions on Computers*, vol. 61, no. 5, pp. 607–621, 2012.

[21] Á. Vázquez, E. Antelo, and P. Montuschi, "Improved Design of High-performance Parallel Decimal Multipliers," *IEEE Transactions on Computers*, vol. 59, no. 5, pp. 679–693, 2010.

[22] A. Nannarelli, "FPGA based Acceleration of Decimal Operations," *Proceedings ofInternational Conference on Reconfigurable Computing and FPGAs*, pp. 146–151, 2011.

[23] A. Vazquez, and F. de Dinechin, "Multi-operand Decimal Tree Adders for FPGAs," *INRIA, Research Report*, 2010.

[24] G. Bioul, M. Vazquez, G. P. Deschamps, and G. Sutter, "Decimal Addition in FPGA," *Proceedings of the 5th Southern Conference on Programmable Logic*, pp. 101–108, 2009.

[25] H. Calderon, G. Gaydadjiev, and S. Vassiliadis, "Reconfigurable Universal Adder," *Proceedings of International Conference on Application-specific Systems, Architectures and Processors*, pp. 186–191, 2007.

[26] T. Mohit, V. Apurva, and K. Kavita, "A Novel Hardware Efficient Reconfigurable 32-bit Arithmetic Unit for Binary, BCD and Floating-point Operands," *International Journal of Engineering Science & Technology*, vol. 3, no. 5, pp. 4449–4464, 2011.

[27] M. A. Shami, "Dynamically Reconfigurable Resource Array," *Ph.D. Dissertation, KTH Royal Institute of Technology, Sweden*, 2012.

[28] R. Brent, and H. Kung, "A Regular Layout for Parallel Adders," *IEEE Transactions on Computers*, vol. C-31, no. 3, pp. 260–264, 1982.

[29] J. Sklansky, "Conditional-sum Addition Logic," *IRE Transaction Electronic Computers*, vol. EC-9, pp. 226–231, 1960.

[30] P. Kogge, and H. Stone, "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations,"*IEEE Transactions on Computers*, vol. C-22, pp. 786–793, 1973.

[31] T. Han, and D. Carlson, "Fast Area-efficient VLSI Adders," *Proceedings of the 8th IEEE Symposium on Computer Arithmetic*, pp. 49–56, 1987.

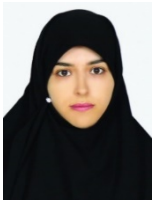
[32] S. Knowles, "A Family of Adders," *Proceedings of the 15th IEEE Symposium on Computer Arithmetic*, pp. 277–281, 2001.

[33] R. Ladner, and M. Fischer, "Parallel Prefix Computation," *Journal of ACM*, vol. 27, no. 4, pp. 831–838, 1980.

[34] M.Cowlishaw,"Doapplicationsactuallyusedecimaldata?, "http://speleotrove.com/decimal/decifaq1.html#dbstats, October 2016.

[35] NCR Corporation, http://www.ncr.com/products-and-services/cloud-services, October 2016.

[36] M. Dorrigiv, and G. Jaberipur, "Low Area/power Decimal Addition with Carry-select Correction and Carry-select Sum-digits," *Integration, the VLSI Journal*, vol. 47, no. 4, pp. 443–451, 2014.



Samaneh Emami received her B.S. and M.S. degrees in Computer Engineering from Shahid Beheshti University in 2008 and 2011, respectively. Since 2011, she has been pursuing her Ph.D. in Computer Engineering at the Department of Computer Engineering and Information Technology (CEIT), Amirkabir University of Technology. Her research interests include computer arithmetic, high-level synthesis and reconfigurable design.

E-mail: s.emami@aut.ac.ir

Mehdi Sedighi received his B.S. in Electrical and Computer Engineering from Sharif University of Technology in 1990 and his M.S. and Ph.D. in the same field from University of Colorado at Boulder in 1994 and 1998, respectively. Since late 2001, he has been with the Department of Computer Engineering and Information Technology at Amirkabir University of Technology where he is currently an associate professor. His research interests include VLSI design, synthesis of arithmetic circuits, embedded systems and quantum computing.

E-mail: msedighi@aut.ac.ir

Paper Handling Data:

Submitted: 07.10.2016
Received in revised form: 21.11.2016
Accepted: 01.12.2016
Corresponding author: Dr. Mehdi Sedighi,
Computer Engineering and Information Technology Department, Amirkabir University of Technology, Tehran, Iran.