ZKG INTERNATIONAL

ISSN: 2366-1313

Area Efficient Approximate Multiplier using 5:2 Compressors

L . Sravani
Department of VLSISD
Vidya Jyothi Institute of Technology,
Aziznagar, Hyderabad
sravanigoud2116@gmail.com

K . Baboji, Assistant Professor,
Department of E.C.E,
Vidya Jyothi Institute of Technology,
Aziznagar, Hyderabad
babojiece@vjit.ac.in

ABSTRACT:

Today research is going towards high efficiency in designs of semiconductor industry to give affordable and viable product to consumers. Area, power and delay are the parameters act as backbone for VLSI research. Among those Area efficient designs are having high end demand. So, in this paper work Designing effective approximate multiplier in terms of area efficiency is drawn. We use approximate compressors for attaining this. By employing these compressors in the Dadda multiplier structure provides better delay and area. This paper followed 5:2 compressor with reference to 4:2 compressor. Previous research work based 5:2 compressor generates overflow carry outputs, which we are reducing here. We show that this new architecture achieves comparable precision to that of earlier designs in terms of both space and energy consumption. We also give a forecast of the implemented design's potential utility in error-tolerant applications like picture smoothing and image multiplication.
KeyWords: Compressor, Multipliers. Dadda approach,

I. INTRODUCTION

Multipliers have found widespread application in digital signal processing and other areas. Researchers have attempted, and continue to attempt, to construct multipliers that are suitable for a wide variety of high-speed, low-power, compact VLSI implementations by achieving one or more of the following design targets: high speed, low power consumption, regularity of layout, and consequently reduced area. As a rule, multiplication is performed using the "add and shift" method. The number of accumulating partial products is the most crucial factor in evaluating the efficiency of parallel multipliers. One of the most popular methods [1] for reducing the number of necessary intermediate products is the Modified Booth algorithm. The Wallace Tree technique is useful for reducing the amount of sequential addition steps, which in turn results in speed increases. In addition, by combining the Modified Booth algorithm and the Wallace Tree method, we can reap the benefits of both approaches with a single multiplier. In addition to increasing silicon space and power consumption due to more sophisticated routing of interconnects, increasing parallelism also requires additional movement between the partial products and intermediate sums to be added [2]. The performance of "serial-parallel" multipliers, on the other hand, improves in terms of both space and power consumption, but at the expense of multiplier speed. Selecting a parallel or serial multiplier depends on the needs of the application. In this lecture, we will cover the basics of multiplication, from algorithms to architectures, and compare them based on their respective speeds, areas, powers, and combinations of these.

Thus, multipliers are extremely important in the realms of automated signal preparation and other areas of study nowadays. Researchers have attempted, and continue to attempt, to create multipliers that achieve all three of the following aims at the same time: fast speed, low force utilisation, consistent format, and therefore reduced area, or all three goals in a single multiplier. Add and shift calculation is the standard method of enlargement. The primary barrier that determines the presentation of an equal multiplier is the number of unfinished items to be added [3]. One of the most well-known approaches to decreasing the total number of fractional elements to add is the Modified Booth calculation. Calculating using Wallace Trees can save time by reducing the number of iterations required to do successive additions. To further our understanding, we can combine the advantages of the Modified Booth computation and the Wallace Tree approach into a single multiplier. While the number of transitions between intermediate items and moderate speeds remains constant with increasing parallelism,

In computer mathematics, n:2 compressors (or n:2 counters) are commonly employed, where n is the number of compressors, since they are effective at reducing n integers to 2 by multi-operand carry-save addition or parallel multiplication. When properly reproduced, a n:2 compressor may take n numbers and reduce them to only two. The n:2 compressor in slice I of the circuit receives n bits at position I and one or more carry bits at positions to the right, such as i^{th}i -1 and i^{th}-2. It generates two data bits at points it and i^{th}+1, and it may generate additional carry bits at positions it+1 and i^{th} +2. The following inequality, written as Equation 1, must be met if the circuit is to work properly.

$$n + \varphi1 + \varphi2 + \varphi3 + \ldots \leq 3 + 2\varphi1 + 4\varphi2 + 8\varphi3 \ldots\ldots(1)$$

Where φj is the total number of carry bits between slices I and i+j. A 4-2 compressor, as shown in Figure 1, is a popular choice since it may be constructed with a carry bit between adjacent slices(1 = 1). Cin represents right-most position carry information while Cout represents top-most position carry information. Both bits at output places I and i+1 are recognised as the sum and carry.

ISSN: 2366-1313

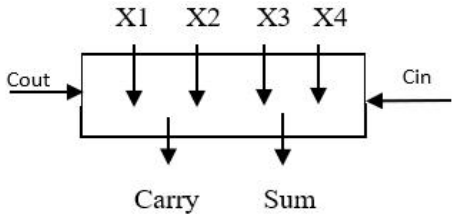

Fig 1 : 4:2 compressor Block Diagram

II. LITERATURE SURVREY

A 4:2 compressor architecture based on XOR and XNOR that consumes little power and performs admirably in fast, tree-structured multipliers. Using a 4: 2 compressor and an unique 5: 2 compressor, Chang et al. offer a design that can run on a minimal input voltage. have presented delay- and power-optimized topologies for the 4: 2 approximate compressor based on logic-level approximation. For a 4: 2 approximate compressor, a reconfigurable architecture is described, wherein flexibility is obtained by toggling between approximation and exact operations. have suggested a 4:1 approximation compressor that, by incorporating a module for error recovery, lowers the error profile of the compressor. At the time of multiplication, n2 columns are truncated (starting from the right in the whole partial product array). In this case, only the remaining columns are compressed.

Guo et al. describe a probabilistically motivated approximation compressor. An approximation multiplier framework is shown from the top down, with dynamic allocation between the 8: 2, 6: 2, and 4: 2 approximate compressors. A grouped error recovery approach is also presented as a means to enhance the multiplier's precision. In [5], a heterogeneous approximate multiplier was reported that relied on approximation adders and had a smaller mean error rate (MED). We use genetic algorithm-based approximate adders to achieve this goal. Esposito et al. propose an XOR-less (AND-OR based) compressor to cut down on both the mean and the standard deviation of errors. In order to improve the energy-to-error-rate ratio in image processing, Chang et al. propose a 4:1 compressor. Delay and power consumption can be minimised with the help of the 4: 2 and 5: 2 compressors proposed by Gorantla and Deepa [6]. Using an error rate, Reddy et al. have presented a new 4: 2 compressor configuration. To do this, the restrictions on size, latency, and power are loosened. The literature investigates optimum design with transmission gates because of the significant delay reduction that may be achieved compared to conventional CMOS-based logic. However, the biggest drawback is that the peak and trough times for various inputs are not consistent with one another.

III. EXISTING SYSTEM

In artificial intelligence and digital signal processing applications, multiplication is without a doubt a bottleneck. Therefore, high-speed multiplier architectures are required for these applications so that high-speed parallel operations can be performed with an acceptable degree of accuracy.

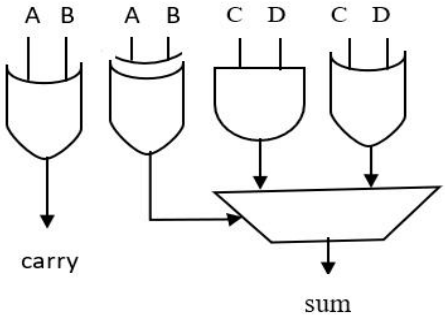

Fig 2: Compressor

By introducing approximation in multipliers, it is possible to achieve quicker computations using less complicated, delayed, and power-hungry gear while still maintaining acceptable levels of accuracy. In fig.2, we see the approximation compressor, which represents the most fundamental use of a 4-2 compressor and is effective thanks to the use of two full-adder (FA) 4-2 compressor has a number of different plans in the works. Since adder networks suffer from a propagation delay, the sum of partial products is the bottleneck in the multiplication process. Compressors are used to shorten the time it takes for a message to travel across the network. At each stage, a compressor does a simultaneous sum and carry calculation. The carry is added with a high-significance sum bit. This procedure is repeated until the desired output is achieved.

In an approximated compressor, the estimate of the information cin state is also the estimate of the carry output. In this way, the carry is untangled to cin by adjusting the estimation of the other 8 possibilities, making the inexact configuration necessary.

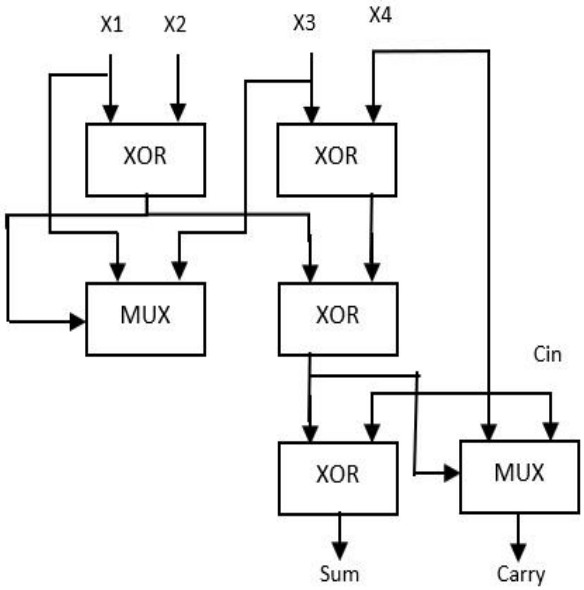

Fig 3: Enhanced 4:2 compressor

Due to the binary nature of the Carry outcomes, a wrong guess of their sign will lead to a differential estimate of two in the return. With an input pattern of "001" "001," the proper output is "010," or 2. The expected compressor will produce the "011" pattern by reordering the convey yield to cin (i.e. an estimation of 3). This significant distinction may not be satisfactory; in any case, it can be remunerated or

ISSN: 2366-1313

decreased by improving the cout and total signs. The below figure (fig 3), shows an example of 4:2 compressor-based multiplier design. To get the ultimate result, a multiplication processor must first generate partial products, which are then passed to full adders and half adders for the reduction process. Here among with half and full adders we are incorporating 4:2 compressor. These processes is called as reduction process. Without making tree structure it self this reduction process is applied. Then later tree structure is formed and then again compressors are applied to get final addition. Here we are following Wallace tree-based reduction process. From this research work we got an idea to implement the multiplication process by generating partial products using Dadda Tree approach and then those are reduced using 5:2 compressors. The following section explains that. And also these section shows two types of 5:2 compressor designs for theorical analysis.

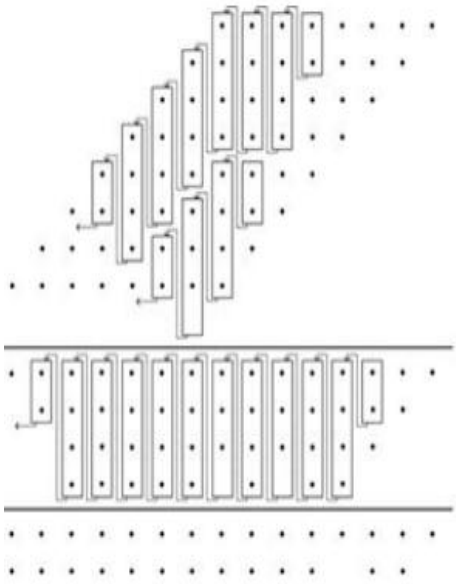

Fig 4: Multipier design using 4:2 compressor

IV. PROPOSED SYSTEM

Using the significance driven logic compression method, our PPM (partial product matrix) reduction circuitry is able to cut power usage with only a tiny amount of guesswork. higher-weighted importance indicators employ more Approximate high-order compressors are used in the medium significance weights, while incorrect compressors are used in the lower significance weights. There are two levels of circuitry in PPM reduction. Stage one is for all of the weights. Only the higher significance weights go to the second phase. Each weight has a maximum of two product terms when the second step is finished. In order to conserve power, we adopt a simple OR tree-based approximation for each lower significance weight. If there are n inputs, then nothing happens. We use OR tree inputs to approximate the accumulation outcome, on the other hand.

There shouldn't be more than two product words for each smaller importance weight after the initial phase is finished. Our approximate n:2 compressor, where n is the number of product terms in this weight, is used to conserve energy for each middle significance weight. Only the higher significance weights go to the second phase. When we want to go really precise, we use accurate compressors to bring down the PPM's maximum height. After the second stage is complete, the carry bit Cin is set to zero, and there are two product terms for each greater significance weight.

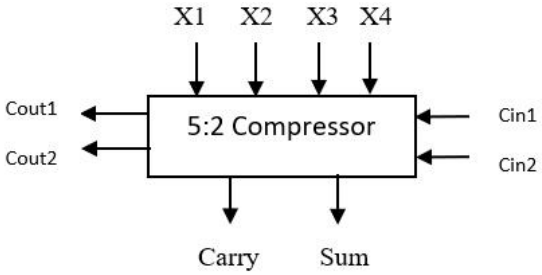

Fig5: 5:2 Compressor

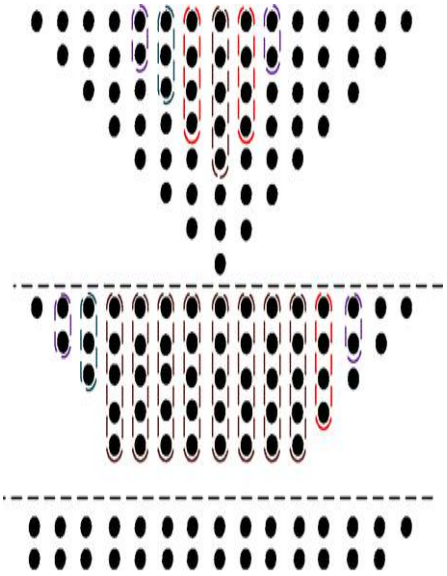

Fig 6 : Multiplier design using compressor

The above explanation shows the process of reduction using 5:2 compressor. Fig 5 is the general block diagram Exact 5:2 Compressor with cout1 and cout2 with carry and sum outputs. It has five inputs, two outputs, The terms COUT, CARRY, and SUM. That brief using full adders is shown in figure 7 5:2 compressor A,B,C,D or X1,X2,X3,X4 are the four inputs. Figure 6 shows the process of PPR (partial product reduction) using 5:2 compressor. Comparing to figure 7 we are taking reference of existing work 4:2 compressor to avoid extra carry flow bits and also to reduce area by using inversion operation.

ISSN: 2366-1313

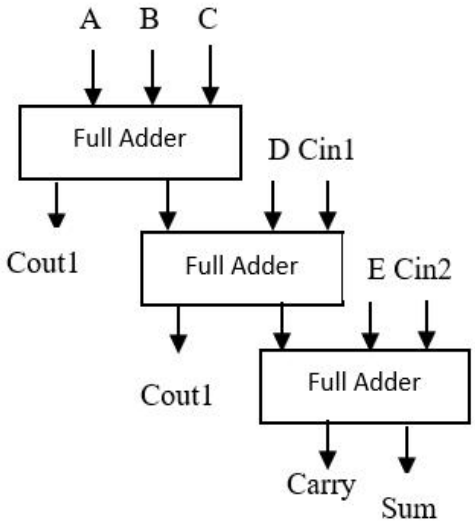

Fig 7: 5:2 Compressor Cycle

Approximate 5:2 Compressor:

Where CIN is the carry from the previous 5:2 compressor, relocated to the LSBs of the output (COUT) of the next compressor in the chain. The inaccuracy is eliminated by cascading the compressor in multiples of 2.

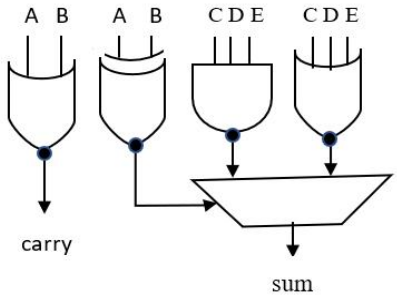

Fig8: Approximate 5:2 compressor

DADDA Multiplier was defined via the following three procedures:
To get the total argument strength, multiply each individual argument's strength by the strength of every other argument. Reduce the number of partial products to only two levels of full and half adders. After pairing off the wires, a regular calculator can be used to tally them up. It is intended that a multiplier can be created utilising dada multiplier design. Compressors are incorporated to the design of the multiplier in place of the more traditional full adders and half adders, thereby simplifying the multiplier. To evaluate the effect of implementing the proposed compressors in approximative multipliers, a DADDA multiplier is investigated. All partial products in the proposed multiplier are initially generated using AND gates. Half-adders, full-adders, and 5:2 compressors are used in the reduction process, with each bit of the partial result represented by a dot. Half-adders, full-adders, and a maximum of four rows for the partial products are used to get started. The second and final step involves computing the last two rows of partial products using half-adders, full-adders, and compressors. That's why, in addition to half-adders, full-adders, and compressors, the DADDA multiplier's reduction circuitry needs two stages of reduction.

V. RESULTS

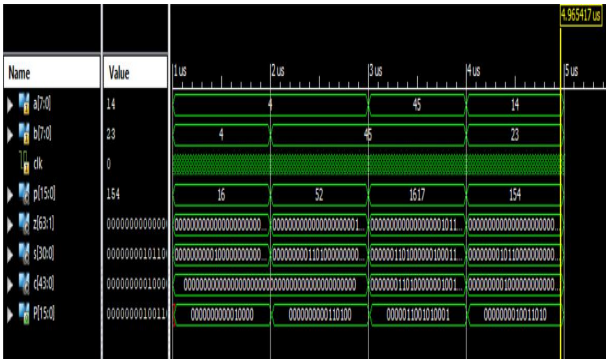

Fig 9: Simulation results using 5:2 compressor

```
Slice Logic Utilization:
 Number of Slice LUTs:                   83  out of  17600     0%
    Number used as Logic:                83  out of  17600     0%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:     83
   Number with an unused Flip Flop:      83  out of     83   100%
   Number with an unused LUT:             0  out of     83     0%
   Number of fully used LUT-FF pairs:     0  out of     83     0%
   Number of unique control sets:         1

IO Utilization:
 Number of IOs:                         33
 Number of bonded IOBs:                 33  out of    100    33%
    IOB Flip Flops/Latches:             16

Specific Feature Utilization:
 Number of BUFG/BUFGCTRLs:               1  out of     32     3%
```

Fig 10: Area results using 5:2 compressor

```
Timing Summary:
---------------
Speed Grade: -2

   Minimum period: No path found
   Minimum input arrival time before clock: 7.461ns
   Maximum output required time after clock: 0.575ns
   Maximum combinational path delay: No path found
```

Fig11: Delay results of 5:2 Compressor

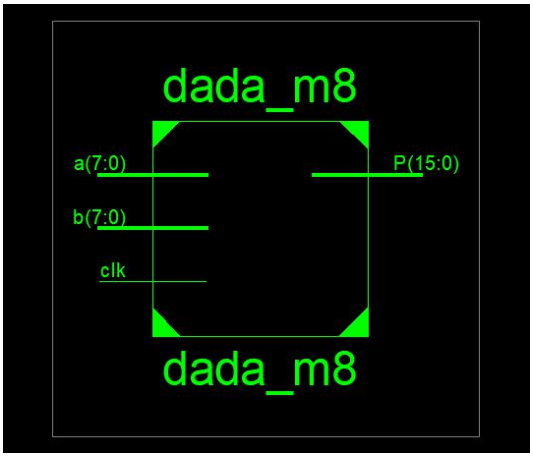

Fig12: Block Diagram

ISSN: 2366-1313

Fig 13: Technology Schematic

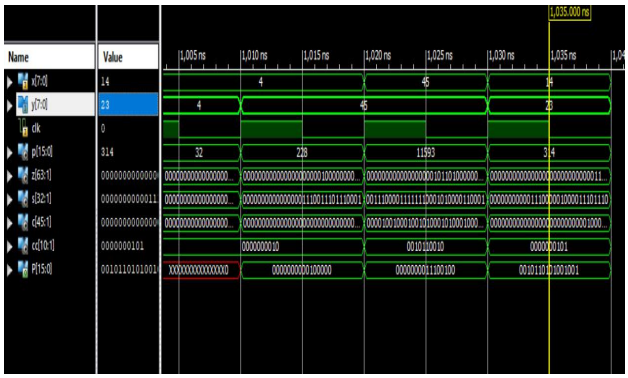

Fig 14: Simulation results using 4:2 compressor

```
Slice Logic Utilization:
 Number of Slice LUTs:                  83  out of  17600     0%
    Number used as Logic:               83  out of  17600     0%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:    83
   Number with an unused Flip Flop:     83  out of     83   100%
   Number with an unused LUT:            0  out of     83     0%
   Number of fully used LUT-FF pairs:    0  out of     83     0%
   Number of unique control sets:        1

IO Utilization:
 Number of IOs:                         33
 Number of bonded IOBs:                 33  out of    100    33%
    IOB Flip Flops/Latches:             16

Specific Feature Utilization:
 Number of BUFG/BUFGCTRLs:              1  out of     32     3%
```

Fig 15: Area results using 4:2 compressor

```
Timing Summary:
---------------
Speed Grade: -2

   Minimum period: No path found
   Minimum input arrival time before clock: 7.252ns
   Maximum output required time after clock: 0.575ns
   Maximum combinational path delay: No path found
```

Fig16:Delay results Of 5:2 Compressor

Comparison	Existing work	Proposed Work
Area	108(LUT's)	83(LUT's)
Delay	8.498ns	7.252ns
Power	0.065A	0.065A

Fig 17:Comparission Table

VI. CONCLUSION

There is a lot of information available on approximate adders, and computer arithmetic offers considerable operational benefits for approximate computation. The initial emphasis of this study was on compression's multiplicative potential. The original 5:2 approximation compressor design is provided in this publication. For approximation purposes, these approximate compressors are used in a multiplication of reduction modules. Comparing an accurate design to an approximate one, we find that the approximate compressors significantly reduce both Area and delay with constant power usage. This work examines the efficiency of approximate compressors based on the aforementioned metrics for inexact multiplication, and does so using two alternative approximate schemes (4:2 and 5:2 with inversion operation). To compress data more roughly, a DADDA multiplier has used an approximation compressor in its reduction module.

REFRENCES

[1]J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and Probabilistic Adders" IEEE Trans Computers vol. 63, no. 9, pp. 1760–1771, Sep 2013.

[2]V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy, "IMPACT: IMPrecise adders for low-power approximate computing," in Proc. Int. Symp. Low Power Electron Design, Aug. 2011, pp. 409–414.

[3] M. J. Schulte and E. E. Swartzlander Jr., "Truncated multiplication with correction constant" in Proc. Workshop VLSI Signal Process VI, 1993, pp. 388–396.

[4] C. Chang, J. Gu, and M. Zhang, "Ultra low-voltage low-power CMOS 4-2 and 5-2 compressors for fast arithmetic circuits," IEEE Trans. Circuits Syst., vol. 51, no. 10, pp. 1985–1997, Oct. 2004.

[5] D. Radhakrishnan and A. P. Preethy, "Low-Power CMOS pass logic 4-2 compressor for high-speed multiplication," in Proc. IEEE 43rd Midwest Symp Circuits Syst., 2000, vol.3, pp 1296–1298.

[6] J. Gu and C. H. Chang, "Ultra low-voltage, lowpower 4-2 compressor for high speed multiplications," in Proc. 36th IEEE Int. Symp. Circuits Syst., Bangkok, Thailand, May 2003, pp. v-321–v-324.

[7] M. Margala and N. G. Durdle, "Low-power lowvoltage 4-2 compressors for VLSI Applications," in Proc. IEEE Alessandro Volta Memorial Workshop Low-Power Design, 1999, pp. 84–90.

[8] D. Baran, M. Aktan, and V. G. Oklobdzija, "Energy efficient implementation of parallel CMOS multipliers"

ISSN: 2366-1313

[9] J. Ma, K. Man, T. Krilavicius, S. Guan, and T. Jeong, "Implementation of high performance multipliers based on approximate compressor design," presented at the Int. Conf. Electrical and Control Technologies, Kaunas, Lithuania, 2011.
[10] C. Liu, J. Han, and F. Lombardi, "A low-power, high-performance approximate multiplier with configurable partial error recovery," in Proc. Conf. Exhibit. (DATE), 2014, pp. 1–4.
[11] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan, "MACACO: Modeling and analysis of circuits," in Proc.IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD), Oct. 2011, pp. 667–673.
[12] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders," IEEE Trans. Comput., vol. 63, no. 9, pp. 1760–1771, Sep. 2013.