JES
Journal of Engineering Sciences

Vol 13, Issue 01, Jan /2022
ISSN NO:0377-9254

DESIGN OF APPROXIMATE FULL ADDRESS USING MAJORITY LOGIC

CHITTIMILLA PRAVALLIKA (2451-19-744-012),
EMBEDDED SYSTEMS AND VLSI DESIGN (ELECTRONICS AND COMMUNICATION ENGINEERING),
MATURI VENKATA SUBBA RAO ENGINEERING COLLEGE,
OSMANIA UNIVERSITY.

ABSTRACT

As a new paradigm in the nanoscale technologies, approximate computing enables error tolerance in the computational process; it has also emerged as a low power design methodology for arithmetic circuits. Majority logic (ML) is applicable to many emerging technologies and its basic building block (the 3-input majority voter) has been extensively used in digital circuit design. In this paper, we propose the design of a onebit approximate full adder based on majority logic. Thispaper has proposed Majority logic based one-bit and multi-bit approximate full adders; these designs show considerable savings in area, delay and number of gates while only incurring a modest loss in accuracy. Compared with the accurate full adder, the proposed designs result an improvement of at least up to 30% in delay is achieved in 16-bit approximate full adders compared to 8-bit approximate full adders.

INTRODUCTION

An adder is a digital circuit that performs addition of numbers. In many computers and other kinds of processors adders are used in the arithmetic logic units or ALU. They are also used in other parts of the processor, where they are used to calculate addresses, table indices, increment and decrement operators and similar operations. Although adders can be constructed for many number representations, such as binary-coded decimal or excess-3, the most common adders operate on binary numbers. In cases where two's complement or ones' complement is being used to represent negative numbers, it is trivial to modify an adder into an adder–subtractor. Other signed number representations require more logic around the basic adder. The half adder adds two single binary digits A and B. It has two outputs, sum (S) and carry (C). The carry signal represents an overflow into the next digit of a multi-digit addition. The value of the sum is 2C + S. The simplest half-adder design, pictured on the right, incorporates an XOR gate for S and an AND gate for C. The Boolean logic for the sum (in this case S) will be A′B + AB′ whereas for the carry (C) will be AB. With the addition of an OR gate to combine their carry outputs, two half adders can be combined to make a full adder. The half adder adds two input bits and generates a carry and sum, which are the two outputs of a half adder. The input variables of a half adder are called the augend and addend bits. The output variables are the sum and carry. The truth table for the half adder is: A full adder adds binary numbers and accounts for values carried in as well as out. A one-bit full-adder adds three one-bit numbers, often written as A, B, and C_{in}; A and B are the operands, and C_{in} is a bit carried in from the previous less-significant stage. The full adder is usually a component in a cascade of adders, which add 8, 16, 32, etc. bit binary numbers. The circuit produces a two-bit output. Output carry and sum typically represented by the signals C_{out} and S, where the sum equals $2C_{out}$ + S.

A full adder can be implemented in many different ways such as with a custom transistor-level circuit or composed of other gates. One example implementation is

Journal of Engineering Sciences
Vol 13, Issue 01, Jan /2022
ISSN NO:0377-9254

with $S = A \oplus B \oplus C_{in}$ and $C_{out} = (A \cdot B) + (C_{in} \cdot (A \oplus B))$.

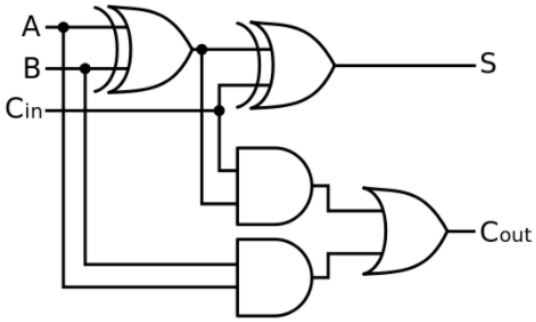


Fig 1 Circuit Diagram Of Full Adder

In this implementation, the final OR gate before the carry-out output may be replaced by an XOR gate without altering the resulting logic. Using only two types of gates is convenient if the circuit is being implemented using simple integrated circuit chips which contain only one gate type per chip.A full adder can also be constructed from two half adders by connecting A and B to the input of one half adder, then taking its sum-output S as one of the inputs to the second half adder and C_{in} as its other input, and finally the carry outputs from the two half-adders are connected to an OR gate. The sum-output from the second half adder is the final sum output (S) of the full adder and the output from the OR gate is the final carry output (C_{out}). The critical path of a full adder runs through both XOR gates and ends at the sum bit s. Assumed that an XOR gate takes 1 delays to complete, the delay imposed by the critical path of a full adder .It is possible to create a logical circuit using multiple full adders to add N-bit numbers. Each full adder inputs a C_{in}, which is the C_{out} of the previous adder. This kind of adder is called a ripple-carry adder (RCA), since each carry bit "ripples" to the next full adder. Note that the first (and only the first) full adder may be replaced by a half adder (under the assumption that $C_{in} = 0$).

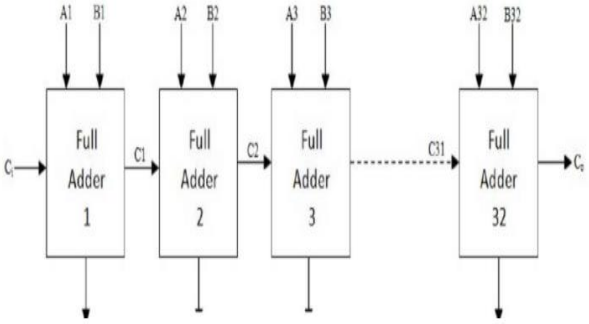


Fig 2 Ripple Carry Adder

The layout of a ripple-carry adder is simple, which allows fast design time; however, the ripple-carry adder is relatively slow, since each full adder must wait for the carry bit to be calculated from the previous full adder. The gate delay can easily be calculated by inspection of the full adder circuit. Each full adder requires three levels of logic. In a 32-bit ripple-carry adder, there are 32 full adders, so the critical path (worst case) delay is 3 (from input to carry in first adder) + 31 × 2 (for carry propagation in latter adders) = 65 gate delays. The general equation for the worst-case delay for a n-bit carry-ripple adder, accounting for both the sum and carry bits,A design with alternating carry polarities and optimized AND-OR-Invert gates can be about twice as fast.To reduce the computation time, engineers devised faster ways to add two binary numbers by using carry-lookahead adders (CLA). They work by creating two signals (P and G) for each bit position, based on whether a carry is propagated through from a less significant bit position (at least one input is a 1), generated in that bit position (both inputs are 1), or killed in that bit position (both inputs are 0). In most cases, P is simply the sum output of a half adder and G is the carry output of the same adder. After P and G are generated, the carries for every bit position are created. Some advanced carry-lookahead architectures are the Manchester carry chain, Brent–Kung adder (BKA),[5] and the Kogge–Stone adder (KSA).

Some other multi-bit adder architectures break the adder into blocks. It is possible to vary the length of these blocks based on the propagation delay of the circuits to optimize computation time. These block based adders include the carry-skip adder which will determine P and G values for each block rather than each bit, and the carry-select adder which pre-generates the sum and carry values for either possible carry input (0 or 1) to the block, using multiplexers to select the appropriate result when the carry bit is known.If an adding circuit is to compute the sum of three or more numbers, it can be advantageous to not propagate the carry result. Instead, three-input adders are used, generating two results: a sum and a carry. The sum and the carry may be fed into two inputs of the subsequent 3-number adder without having to wait for propagation of a carry signal. After all stages of addition, however, a conventional adder (such as the ripple-carry or the lookahead) must be used to combine the final sum and carry results. A full adder can be viewed as a 3:2 lossy compressor: it sums three one-bit inputs and returns the result as a single two-bit number; that is, it maps 8 input values to 4 output values. Thus, for example, a binary input of 101 results in an output of 1 + 0 + 1 = 10 (decimal number 2). The carry-out represents bit one of the result, while the sum represents bit zero. Likewise, a half adder can be used as a 2:2 lossy compressor, compressing four possible inputs into three possible outputs.Such compressors can be used to speed up the summation of three or more addends. If the addends are exactly three, the layout is known as the carry-save adder. If the addends are four or more, more than one layer of compressors is necessary, and there are various possible designs for the circuit: the most common are Dadda and Wallace trees. This kind of circuit is most notably used in multipliers, which is why these circuits are also known as Dadda and Wallace multipliers.In Boolean logic, the majority function (also called the median operator) is a function from n inputs to one output. The value of the operation is false when n/2 or more arguments are false, and true otherwise. Alternatively, representing true values as 1 and false values as 0, we may use the formula

$$\langle p_1, \ldots, p_n \rangle = \text{Majority}(p_1, \ldots, p_n) = \left\lfloor \frac{1}{2} + \frac{(\sum_{i=1}^{n} p_i) - 1/2}{n} \right\rfloor.$$

The "−1/2" in the formula serves to break ties in favor of zeros when n is even. If the term "−1/2" is omitted, the formula can be used for a function that breaks ties in favor of ones.

LITERATURE SURVEY:

[1] S.-L. Lu, "Speeding up processing with approximation circuits," Computer, vol. 37, no. 3, pp. 67–73, Mar. 2004.

Approximate circuit designs allow us to tradeoff computation quality (e.g., accuracy) and computational effort (e.g., energy), by exploiting the inherent error-resilience of many applications. As the computation quality requirement of an application generally varies at runtime, it is preferable to be able to reconfigure approximate circuits to satisfy such needs and save unnecessary computational effort. In this paper, we present a reconfiguration-oriented design methodology for approximate circuits, and propose a reconfigurable approximate adder design that degrades computation quality gracefully. The proposed design methodology enables us to achieve better quality-effort tradeoff when compared to existing techniques, as demonstrated in the application of DCT computing.

[2] J. Han and M. Orshansky, "Approximate computing: an emerging paradigm for energy-efficient design," in Proc. ETS, pp. 1-6, May2013

Approximate computing has recently emerged as a promising approach

to energy-efficient design of digital systems. Approximate computing relies on the ability of many systems and applications to tolerate some loss of quality or optimality in the computed result. By relaxing the need for fully precise or completely deterministic operations, approximate computing techniques allow substantially improved energy efficiency. This paper reviews recent progress in the area, including design of approximate arithmetic blocks, pertinent error and quality measures, and algorithm-level techniques for approximate computing.

[3]C. Labrado, H. Thapliyal and F. Lombardi "Design of Majority Logic Based ApproximateArithmetic Circuits," inProc. IEEE International Symposium on Circuits and Systems (ISCAS), pp. 2122-2125, May 2017.

The increasing amount of circuit density possible in CMOS technology has the consequence of also increasing the power consumption of circuits using the technology. One possible method of offsetting these increased power demands is to use approximate computing designs in circuits where complete accuracy is not a strict requirement. These circuits use fewer logic gates which reduces power consumption at the cost of accuracy. Another possible method for reducing power consumption is to use an emerging nanotechnology which is already low power in nature. Combining approximate computing with an emerging nanotechnology has the potential to further cut power consumption. Unfortunately, existing approximate computing circuits were designed using standard logic gates found in CMOS technology which in turn can limit their effectiveness when implemented with the majority based logic used by some emerging nanotechnologies. For that reason, we propose designs of approximate arithmetic units which are specifically designed for use in majority logic based technologies

[4]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 Electronics and Design (ISLPED) , pp. 409– 414, Aug. 2011.

Low-power is an imperative requirement for portable multimedia devices employing various signal processing algorithms and architectures. In most multimedia applications, the final output is interpreted by human senses, which are not perfect. This fact obviates the need to produce exactly correct numerical outputs. Previous research in this context exploits error-resiliency primarily through voltage overscaling, utilizing algorithmic and architectural techniques to mitigate the resulting errors. In this paper, we propose logic complexity reduction as an alternative approach to take advantage of the relaxation of numerical accuracy. We demonstrate this concept by proposing various imprecise or approximate Full Adder (FA) cells with reduced complexity at the transistor level, and utilize them to design approximate multi-bit adders. In addition to the inherent reduction in switched capacitance, our techniques result in significantly shorter critical paths, enabling voltage scaling. We design architectures for video and image compression algorithms using the proposed approximate arithmetic units, and evaluate them to demonstrate the efficacy of our approach. Post-layout simulations indicate power savings of up to 60% and area savings of up to 37% with an insignificant loss in output quality, when compared to existing implementations.

[5] Z. Yang, A. Jain, J. Liang, J. Han, and F. Lombardi, "Approximate xor/xnor-based adders for inexact computing," in Proc. 13th IEEE Conf. Nanotechnology (IEEE-NANO), pp. 690–693, Aug.2013.Power dissipation has become a significant issue for integrated circuit design in nanometric CMOS technology.

To reduce power consumption, approximate implementations of a circuit have been considered as a potential solution for applications in which strict exactness is not required. In inexact computing, power reduction is achieved through the relaxation of the often demanding requirement of accuracy. In this paper, new approximate adders are proposed for low-power imprecise applications. These adders are based on XOR/XNOR gates with multiplexers implemented by pass transistors. The proposed approximate XOR/XNOR-based adders (AXAs) are evaluated and compared with respect to energy consumption, delay, area and power delay product (PDP) with an accurate full adder. The metric of error distance is used to evaluate the reliability of the approximate designs. Simulation by Cadence's Spectre in TSMC 65nm process has shown that the proposed designs consume less power and have better performance (such as a lower propagation delay) compared to the accurate XOR/XNOR-based adder, while the error distance remains similar or better than other approximate adder designs

EXISTING SYSTEM

The majority gate performs a multi-input logic operation and is shown in Fig. 1; the inputs are A, B and C and the output is F. The logic expression of the 3- input majority gate (voter) is given by

$$F = M(A,B,C) = AB + BC + AC$$

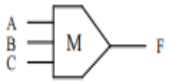
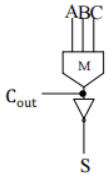


Fig 3 Majority gate (3-input voter)

Fig. 1 Majority gate (3-input voter) Research on the design of ML approximate circuits has only recently been pursued; In the authors proposed a one-bit full adder circuit. In this paper, we propose a few ML-based approximate full adders; one-bit as well as multi-bit approximate adders are considered.

one-bit approximate full adder (AFA1), AFA1produces the output S as the complement of C but introduces 2 errorswhen computing the output S

$$C_{out} = M(A,B,C)$$
$$S = \overline{C_{out}}$$

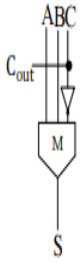


(a)

,

Fig 4One-bit approximate full adder AFAI

one-bit approximate full adder is presented and compared with one-bit accurate full adder.A new one-bit approximate full adder, namely, AFA2.

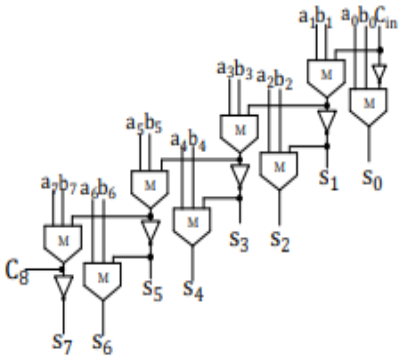


(a)

Fig 5 Schematic one-bit approximate full-adder AFA2

Consider an eight-bit adder with inputs, A =a7a6a5a4a3a2a1a0,B=b7b6b5b4b3b2b1b 0 , C and outputs, S = s7s6s5s4s3s2s1s0 , C8.We have designed eight-bit approximate adders by cascading two four-bit approximate adders by using AFA1212

and AFA2121, as they show better overall performance than the other two designs. The eight-bit approximate adders are shown in Fig.3.7; The designs significantly reduce the number of gates and delay but at the cost of a decrease in accuracy.



(A) MLFA8A

(B)MLFA8B

Fig 6 existing 8-bit approximate full adders

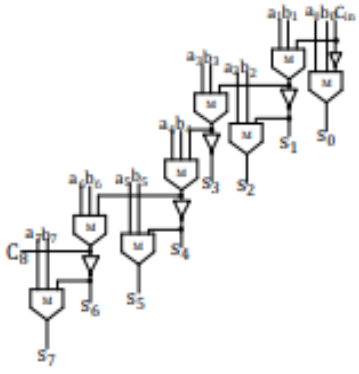


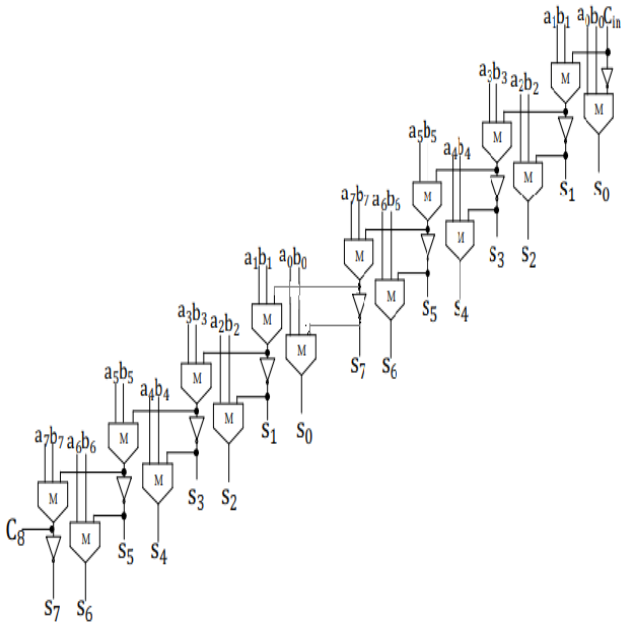
Table1 Eight Bit Approximate Full Adder Comparison

DESIGN	FFS	LUTS	DELAY	MED	NMED
MLFA8B	9	26	1.78ns	92.625	0.181
MLFA8A	9	25	1.434ns	4.875	0.093

PROPOSED SYSTEM

Consider an Sixteen-bit adder with inputs, A, B, C and outputs, S , C8.We have designed sixteen-bit approximate adders by cascading two eight-bit approximate adders by using MLFA8A and MLFA8B, as they show better overall performance than the other two designs. The proposed Sixteen-bit approximate adders are shown in Fig.3.8; Fig. 3.9. The proposed designs significantly reduce the number of gates and delay but at the cost of a decrease in accuracy.

(A)MLFA16A



(B)MLFA16B

Fig 7 proposed system architecture

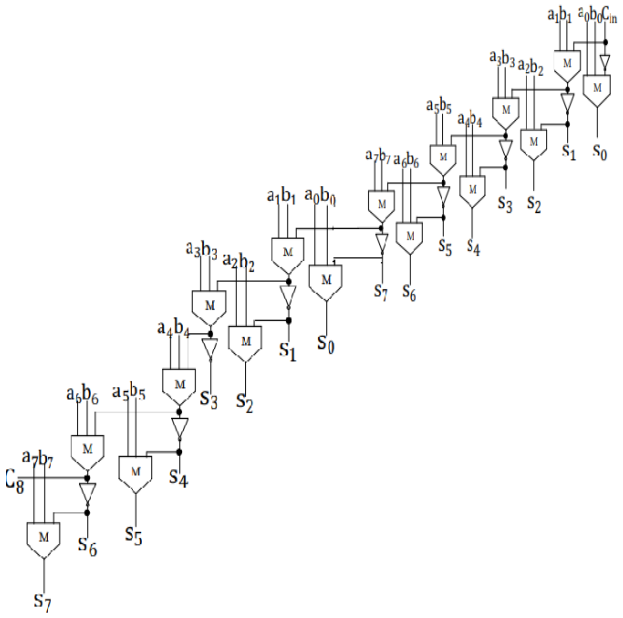


Table 2 Sixteen Bit Approximate Full Adder Comparison

DESIGN	FFS	LUTS	DELAY	MED	NMED
MLF16B	16	48	2.551ns	43,828	0.334
MLF16A	17	50	2.428ns	18,922	0.144

CONCLUSION

The paper has proposed Majority logic based one-bit and multi-bit approximate full adders; these designs show considerable savings in area, delay and number of gates while only incurring a modest loss in accuracy. Compared with the accurate full adder, the proposed designs result an improvement of at least up to 30% in delay is achieved in 16-bit approximate full adders.

REFERENCES

[1] S.-L. Lu, "Speeding up processing with approximation circuits," Computer, vol. 37, no. 3, pp. 67–73, Mar. 2004.

[2] J. Han and M. Orshansky, "Approximate computing: an emerging paradigm for energy-efficient design," in Proc. ETS, pp. 1-6, May2013

[3] C. Labrado, H. Thapliyal and F. Lombardi "Design of Majority Logic Based ApproximateArithmetic Circuits," inProc. IEEE International Symposium on Circuits and Systems (ISCAS), pp. 2122-2125, May 2017.

[4] 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 Electronics and Design (ISLPED) , pp. 409– 414, Aug. 2011.

[5] Z. Yang, A. Jain, J. Liang, J. Han, and F. Lombardi, "Approximate xor/xnor-based adders for inexact computing," in Proc. 13th IEEE Conf. Nanotechnology (IEEE-NANO), pp. 690–693, Aug.2013.

[6] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy. "Low-power digital signal processing using approximate adders,"IEEE Trans. Comput.-Aided Des. Integ. Circuits Syst, vol. 32,pp. 124–137, 2013. [7] S. Rehman, W. El-Harouni, M. Shafique, A. Kumar, and J. Henkel. "Architectural-Space Exploration of Approximate Multipliers,".in Proc. Int. Conf. Comput.-Aided Des. (ICCD), pp. 1-6,Nov.2016.

[8] N. Maheshwari, Z. Yang, J. Han, and F. Lombardi. "A design approach for compressor based approximate multipliers," in Proc. 28th Int. Conf. VLSI Des., pp. 209-214,Jan. 2015.

[9] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability ofapproximate and probabilistic adders," IEEE Transactions on Computers, vol. 62, no. 9, pp. 1760–1771, 2013.

[10] K. Walus and G. Jullien,"Design tools for an emerging SoCtechnology:quantum-dot cellular automata," in Proc. IEEE, vol. 94, no. 6,pp. 1225-1244,2006.

[11] C. Lent and P Tougaw, "A device architecture for computing withquantum dots," in Proc. IEEE, vol. 85,no. 4,pp. 541-557,1997.

[12] M. Vacca, M. Graziano, J. Wang, F. Cairo, G. Causapruno, G. Urgese,A. Biroli, and M. Zamboni, "Nanomagnet logic: An architectural leveloverview," Lecture Notes in Computer Science, pp. 223–256, 2014.

[13] A. Khitun and K. L. Wang, "Nano scale computational architectureswith spin

Vol 13, Issue 01, Jan /2022
ISSN NO:0377-9254

wave bus," Superlattices and Microstructures, vol. 38, no. 3,pp. 184–200, 2005.