ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Implementation of Novel High Radix Multiplier Using KOGGE Stone Adder

P. Naresh
M.Tech (VLSI-SD)
Department of ECE
Teegala Krishna Reddy Engineering College, (Meerpet), Hyderabad.

Ms. B. Ramya
Assistant professor
Department of ECE
Teegala Krishna Reddy Engineering College, (Meerpet), Hyderabad.

Dr.P.Ram Mohan Rao
FIE,CE(I), MISTE, MISH, MISCEE, MASCE(I), MISNT
Principal,
TKREC, (Meerpet), Hyderabad.

ABSTARCT

Higher radix values of the form _ = 2r havebeen employed traditionally for recoding of multipliers, andfor determining quotient- and root-digits in iterative division and square root algorithms, usually only for quite moderatevalues of r, like 2 or 3. For fast additions, in particularfor the accumulation of many terms, generally redundantrepresentations are employed, most often binary carry-saveor borrow-save, but in a number of publications it has beensuggested to recode the addends into a higher radix. It is shownthat there are no speed advantages in doing so if the radix isa power of 2, on the contrary, there are significant savings inusing standard 4-to-2 adders, even saving half of the operations in multi-operand addition.

Keywords: *asymmetric high-radix signed-digit number, computer arithmetic, numbersystem conversion, redundant number system, signed-digit numbers, VLSI design.*

INTRODUCTION

Addition is the most important and frequently used arithmetic operation in computer systems. Generally, two methods can be used to speed up the addition operation. One is to explicitly shorten the carry-propagation chain by using circuit design techniques, such as detecting the completion of the carry chain as soon as possible, carry look-ahead, etc. [1-3]. Another is to convert the operands from the binary number system to a redundant number system, e.g., the signed-digit number system or the residue number system, so that the addition become carry-free (CF) [2-5]. This implicitly eliminates the carry-propagation chain so that fast addition can be done, at the expense of conversion between the binary number system and the redundant number system. In this paper we focus on exploring signed-digit (SD) numbers.

tems are defined for any radix $r \geq 3$ and for the digit set $\{-\alpha, \ldots, -1, 0, 1, \ldots, \alpha\}$, whereis an integer such that $r/2 < \alpha < r$. In an OSD number system, the *redundancy index*ρ, defined as $\rho = 2\alpha - r + 1$, ranges from the minimal redundancy $r/2 + 1$ to the maximal redundancy $r - 1$. The most important contribution of OSD is to explore the possibility of performing carry-free addition and borrow-free subtraction for fast parallel arithmetic, if enough redundancy is used. The OSD number system was later extended to the *gener-alized signed-digit* (GSD) number system [6-9]. The GSD number system for radix $r > 1$has the digit set $\{-\alpha, \ldots, -1, 0, 1, \ldots, \beta\}$, where $\alpha \geq 0$ and $\beta \geq 0$. The redundancy is $\rho = + \beta + 1 - r$. So far, the most important contribution of the works on GSD [6-9] include unifying the redundant number representation and sorting the CF addition schemes for the GSD number system according to the radix r and redundancy index ρ. However, ideal single-stage CF addition has not been achieved, though two-stage CF addition has been shown to be doable for any GSD system with $r > 2$ and $\rho > 2$, or with $r > 2$ and $\rho = 2$ provided that $\alpha \neq 1$ and $\beta \neq 1$. For any GSD system with $r = 2$ and $\rho = 1$, or $\rho = 2$ and orβ equals 1, the limited-carry addition must be used.

There are many applications for the SD number representations, most notably in computer and digital signal processing systems. Specifically, the CF adder has been in-vestigated based on the redundant positive-digit numbers [10] and the symmetrical radix-4 SD numbers [11, 12] for high-speed area-efficient multipliers. The symmetrical radix-2 SD number representation has been used in the implementation of a RSA crypto-graphic chips [13], high-speed VLSI multipliers [14, 15], FIR filters [16], IIR filters [17], dividers [18], etc. Though arithmetic operations using these number representations can be done carry free, they have common difficulty in conversion to and from the binary number representation. Hence, in the past, many researchers have proposed specific ar-chitectures for number system conversion [15, 17, 19-22].

In this paper, we present the *asymmetric high-radix signed-digit* (AHSD) number system. The idea of AHSD is not new. A particular AHSD number system was called the radix-*r* stored-borrow (SB) number system in [7]. Most earlier works have focused on binary stored-borrow (BSB) number systems, where *r* = 2 [6, 14, 23-25]. Instead of pro-posing a new number representation, our purpose is to explore the inherent CF property of AHSD. The CF addition in AHSD is the basis for our high-speed addition circuits. The conversion of AHSD to and from binary will be discussed in detail. By choosing $r = 2^m$, where *m* is any positive integer, a binary number can be converted to its canonical AHSD representation in constant time. We will also present two simple algorithms for converting AHSD numbers to binary: the first stresses high speed and the other provides hardware reusability. Since the conversion from AHSD to binary has been considered the bottleneck of AHSD-based arithmetic computation, these algorithms greatly improve the performance of AHSD systems. For illustration, we will discuss in detail the example on AHSD(4), i.e., the radix-4 AHSD number system. The proposed approach is practical thanks to the simple conversion.

MULTI-OPERAND ADDITION

When adding a multiple of operands, say k of n digits, the fastest possible way to do it is to add them in a binary tree,each addition performed using a redundant representation ofthe intermediate sums, to allow constant time addition at thenodes of the tree, with bounded carry-propagation between positions, i.e, without any “ripple-effect”. We will hereconcentrate on adding a pair of digits from two operands,where it alternatively is possible to accumulate several digitsof the same weight in some redundant representation, anddigits) back into the wanted digit set, as suggested in [KS05]for decimal multi-operand addition.We are assuming that the initial operands are in somenon-redundant representation, at most needing some constanttime conversion or recoding of the non-redundantrepresentation. The leaves of the tree must be able to addtwo such addends, with their sum in the chosen redundantrepresentation employed internally in the tree. At the root ofthe tree, the final result must in general be converted back tosome non-redundant representation, most likely the same asthat of the original operands. The accumulation of k, n-digitoperands can thus be performed in time O(log k), with finalconversion in time O(log n).With radix _ _ 2, when the operand representationis employing the standard non-redundant digit set D =f0; 1; _ _ _ ; _ □ 1g, very often the symmetric, maximallyredundant digit set D0 = f□_ + 1; _ _ _ ;□1; 0; 1; _ _ _ ; _ □ 1gis used internally at the nodes, since then d 2 D) d 2 D0.Then at most a simple modification of the digit encoding isnecessary at the leaf nodes. This also permits some simplehandling of sign-magnitude or complement representationsof operands.Although this discussion applies to any value of _ _ 2,we will restrict the detailed analysis to situations where _is a power of 2.For _ = 2 the situation is particularly simple, as a pairingof the bits of two operands provides an encoding of eithertheir sum or difference in a redundant representation at theleaf nodes, also to be used at the internal nodes of the tree.Let x =Pn□10 xi2i and y =Pn□10 yi2i be two binaryintegers, then x□y =Pn□10 di2i with di = xi □yi being adigit in the redundant digit set f□1; 0; 1g. The pair of bits(yi; xi) then provides an encoding of the digit di, wherexi has positive weight and yi has negative weight. If thesum of the operands is wanted, y can easily be negated byinversion. We denote this the borrow-save encoding, and usethe notation di _ (dni;dpi) with the component of negativeweight in the first position.Alternatively, with the same operands, x+y =Pn□10 si2i,where si = xi +yi 2 f0; 1; 2g is representing the digit sum,the digits can be encoded as the pairs si _ (xi; yi). Thiscarry-save encoding can be employed in the rest of the tree.

But since there is no principal difference between the use ofthe carry-save and the borrow-save representation, we willgenerally consider the latter.Actual implementations of the internal radix 2, 4-to-2addition nodes of the tree will be described in Section 4below, but note that addition at the leaf nodes essentiallycomes for free, thus halving the number of operations to beperformed, as well as saving space.

THE DIGIT-WISE ADDITION PROCESS

Adding two numbers digit-wise in the maximally redundantset D = f□2r+1 ; 0; 2r□1g for radix _ = 2r,r _ 2, generates first an intermediate sum represented in thedigit set D0 = f□2r+1+2 2r+1□2g, from which carriesin the set C = f□1; 0; 1g are extracted, leaving behind digitsfrom a much reduced digit set D" = f□2r +22r □2g.Note that this digit set must contain at least 2r differentvalues. In a subsequent step, incoming carries are added intothe positions, such that all digits now are in the same digitset D as the operands. Addition thus consists in a numberof parallel digit-wise additions:

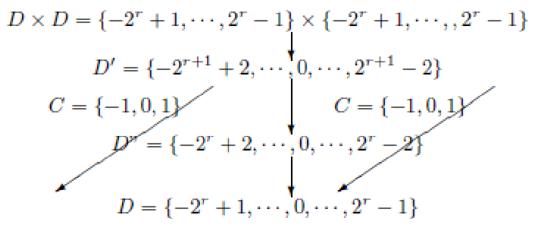
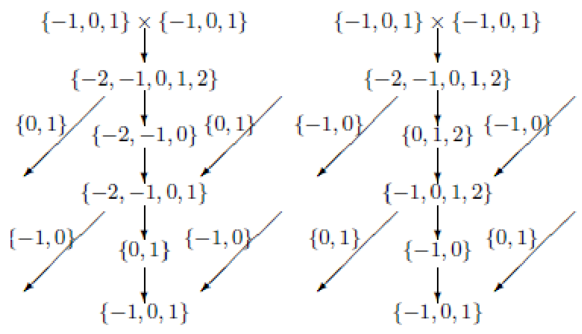

$D \times D = \{-2^r+1,\cdots,2^r-1\} \times \{-2^r+1,\cdots,,2^r-1\}$
$D' = \{-2^{r+1}+2,\cdots,0,\cdots,2^{r+1}-2\}$
$C = \{-1,0,1\}$
$C = \{-1,0,1\}$
$D'' = \{-2^r+2,\cdots,0,\cdots,2^r-2\}$
$D = \{-2^r+1,\cdots,0,\cdots,2^r-1\}$

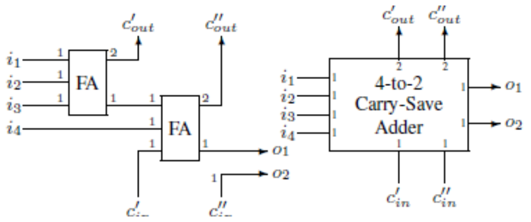
Figure 1. Maximally redundant radix 2r addition for r _ 2

The first step consists in the digit-wise addition, the nextin extracting the outgoing carry and modifying the digitaccordingly. The last step is adding an incoming carry tothe modified digit. The essential thing to observe is that theoutgoing carry is independent of the incoming.Note that this process is possible for any value of r _ 2,but not for r = 1, the binary borrow-save representation withD = f□1; 0; 1g, as we shall discuss below. Also observethat it involves three sequential digit-wise add or subtractoperations. However, the carry-extraction takes place at themost-significant end of the digit, and is thus in generalfaster than the add operations. Unless the encoding of thedigit values themselves is redundant, these operations willbe slower, the larger the digit set is.For radix 2, with the digit set f□1; 0; 1g, the situationseems more complicated. Following Avizienis [Avi61], thedigit set conversion here must take place in two phases fora total of five steps, which takes two forms, depending onthe sign of the incoming carry:



RADIX-2, 4-TO-2 ADDITION

Starting with the redundant radix 2 addition, severalpossible (e.g., two-bit [PGK01], and even three-bit [EL97])encodings of the digits are possible, some of which wereinvestigated in [Kor05]. All of these were shown to befeasible for addition in what has been denoted 4-to-2 adders,realizable by simple modifications of a carry-save, 4-to-2adder for the addition of two operands over the digit setf0; 1; 2g, as shown in Fig. 2. Using the two-bit encoding,with the digit value being the sum of the encoding bits,this type of adder was originally proposed by Weinburger in[Wei81]. There are several possible implementations of it,including some very efficient ones based on pass-transistorbased selectors [OSY+95].



ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Employing the binary borrow-save encoding as a bit-pair(xn; xp), where the left-most bit xn has negative weight, thepair encodes the digit value x = xp □ xn 2 f□1; 0; 1g. Additionof (xn; xp) and (yn; yp) can be realized by invertingsome of the connections as shown in Fig. 3.

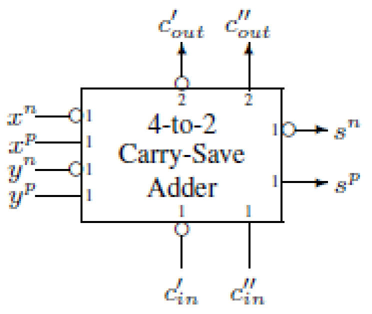


Figure 3. 4-to-2, borrow-save adder

Alternatively, the digit encoding could be 2’s complement,

where a pair (xh; xl) encodes the digit value d = □2xh +xl 2 f□1; 0; 1g (the pair (1; 0) excluded), with carry inborrow-save encoding. Fig. 4 shows an implementation, realizedby a carry-save adder with some signals inverted. Notethat the adder is working on signals from two neighboringpositions, due to the encoding where one signal has a weighttwice that of the other.

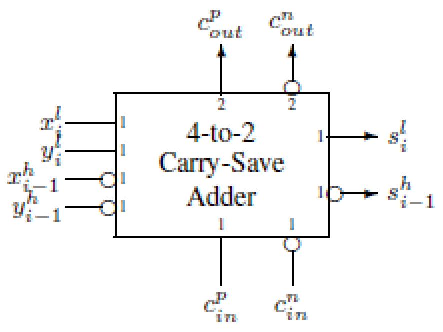


Figure 4. 4-to-2, 2’s complement adder

As shown in [Kor05], it may be noted that in a multi-digitadder composed of a linear array of such adders, the invertersare not needed on the internal carry signals. Similarly, whensuch adders are connected in an array (or tree) for multi operand addition, the inverters are not needed anywhereinternally. This also shows that an implementation usingborrow-save representation is identical to one using carrysave,except for some inverters at the boundary of the array

One such possibility for adding radix-16 digits, is combiningfour 4-to-2 borrow-save adders over the digit-setf□1; 0; 1g, which together provides an equivalent radix-16 digit adder over the same maximally redundant digitset as above. Since each digit now is encoded in 8 bits,compared to 5, a radix-16 digit adder now has 16 inputand 8 output lines, compared to respectively 10 and 5. Thusthe interconnect structure of the adders is more complexand requires more area. But note that only half as manysuch adders may be needed in a multi-operand additionarray, since the sum of two standard non-redundant binaryoperands is simply obtained by pairing the bits of theoperands, directly forming their sum in carry-save encoding(or by inverting one operand in borrow-save), to be used asfurther input.Expanding the 4-to-2 adders in terms of fulladders withtheir interconnections, as an equivalent to the above digitadders in Fig. 5, using the borrow-save encoding we findFig. 6, again not showing inversions on negative signals. Forgeneral r the delay of the circuit in Fig. 6 is independent ofthe radix 2r, and identical to that of a single 4-to-2 adder.

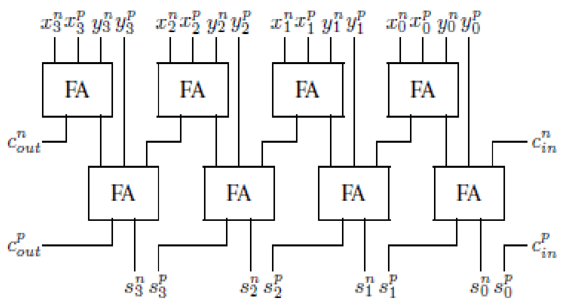


Figure 5. A radix 16 digit-adder design in 4-to-2 borrow-save encoding

Extension: The Kogge–Stone adder is a parallel prefix form carry look-ahead adder. It generates the carry signals in O(log n) time, and is widely considered the

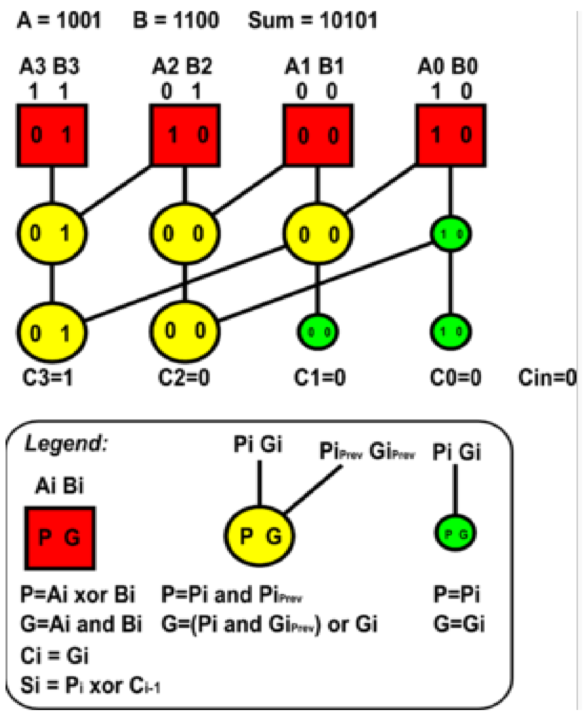
ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

fastest adder design possible. It is the common design for high-performance adders in industry.

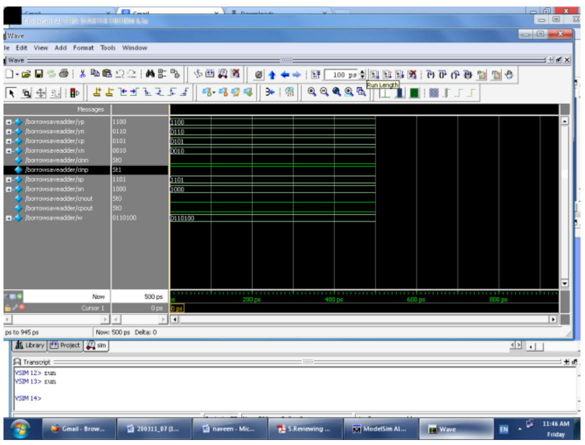
It takes more area to implement than the Brent–Kung adder, but has a lower fan-out at each stage, which increases performance. Wiring congestion is often a problem for Kogge–Stone adders as well.

An example of a 4-bit Kogge–Stone adder is shown to the right. Each vertical stage produces a "propagate" and a "generate" bit, as shown. The culminating generate bits (the carries) are produced in the last stage (vertically), and these bits are XOR'd with the initial propagate after the input (the red boxes) to produce the sum bits. E.g., the first (least-significant) sum bit is calculated by XORingthe propagate in the farthest-right red box (a "1") with the carry-in (a "0"), producing a "1". The second bit is calculated by XORingthe propagate in second box from the right (a "0") with C0 (a "0"), producing a "0".

The Kogge–Stone adder concept was developed by Peter M. Koggeand Harold S. Stone, which they published in 1973 in a seminal paper titled *A* Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations.



RESULTS


```
/borrowsaveadder/yp     1100
/borrowsaveadder/yn     0110
/borrowsaveadder/xp     0101
/borrowsaveadder/xn     0010
/borrowsaveadder/cinn   St0
/borrowsaveadder/cinp   St1
/borrowsaveadder/sp     1101
/borrowsaveadder/sn     1000
/borrowsaveadder/cnout  St0
/borrowsaveadder/cpout  St0
/borrowsaveadder/w      0110100
Now      500 ps
Cursor 1   0 ps
Now: 500 ps  Delta: 0
```

CONCLUSIONS

It has been shown, that the proposed recoding of addendsinto a higher radix of the form 2r for r _ 2, asin [Par93], [MI99], [JPG05], [JP07], [JG10], [GJ11], withdigits encoded in non-redundant 2’s complement representation,can not provide faster addition than using standardradix 2, carry-save or borrow-save adders applied directlyon non-redundant binary addends. On the contrary, formulti-operand addition the delay is significantlylargerwhenusing such recoding into a higher radix, despite the claim“Ultrahigh-Speed” in the title of [JP07]. Furthermore, whenemploying the carry- or borrow-save encoding, half of theadditions come for free, since just pairing two non-redundantbinary numbers forms their sum in carry-save, or (with oneof them inverted) in borrow-save. It is noted that (exceptfor some inverters at the boundary) the logic of the adderarray is identical, whether the carry-save or the borrow-saverepresentation is used.

REFERENCES

[Avi61] A. Avizienis. Signed-digit number representations for fastparallel arithmetic. IRE Transactions on Electronic Computers,EC-10:389–400, September 1961.

[EL97] M.D. Ercegovac and T. Lang. Effective Coding for FastRedundant Adders using the Radix-2 Digit Set f0; 1; 2; 3g.In Proc. 31st Asilomar Conf.

Signals Systems and Computers,pages 1163–1167, 1997.

[GJ09] S. Gorgin and G. Jaberipur. Fully Redundant Decimal Arithmetic.In Proc. 19th IEEE Symposium on Computer Arithmetic,pages 145–152. IEEE, June 2009.

[GJ11] S. Gorgin and G. Jaberipur. A Family of High Radix Signed

Digit Adders. In Proc. 20th IEEE Symposium on ComputerArithmetic, pages 112–121. IEEE, July 2011.

[HNN+87] Y. Harata, Y. Nakamura, H. Nagase, M. Takigawa, and N. Takagi.A High-Speed Multiplier Using a Redundant Binary AdderTree. IEEE Journal of Solid State Circuits, pages 28–34, 1987.

[JG10] G. Jaberipur and S. Gorgin. An Improved Maximally RedundantSignedDigitAdder. Computers and Electrical Engineering,36(3):491–502, May 2010.

[JP07] G. Jaberipur and B. Parhami. Stored-Transfer Representationswith Weighted Digit-Set Encodings for Ultrahigh-Speed Arithmetic.IET Circuits Devices Systems, 1(1):102–110, February2007.

[JPG05] G. Jaberipur, B. Parhami, and M. Ghodsi. Weighted Two-Valued Digit-Set Encodings: Unifying Efficient HardwareRepresentation Schemes for Redundant Number Systems.IEEE Transactions on Circuits and Systems, Regular Papers,

52(7):1348–1357, July 2005.

[KM06] P. Kornerup and J-M. Muller.Leading Guard Digits in FinitePrecision Redundant Representations. IEEE Transactions onComputers, 55(5):541–548, May 2006.

[KM10] P. Kornerup and D.W. Matula. Finite Precision NumberSystems and Arithmetic, volume 133 of Encyclopedia of Mathematicsand its Applications. Cambridge University Press,Sept. 2010.