|| Volume 5 || Issue 12 || December 2020 || ISSN (Online) 2456-0774
INTERNATIONAL JOURNAL OF ADVANCE SCIENTIFIC RESEARCH AND ENGINEERING TRENDS

DESIGN OF HIGH PERFORMANCE ACCURACY-CONFIGURABLE N-BIT

Denaboina Rajitha[1], Mahesh Gudivaka[2]
[1]*M.Tech scholar*, Dept of E.C.E, Jntu Kakinada, AP India, 9676680028,
[2]*Assistant Professor*, Department of E.C.E, Jntu kakinada, AP India, 9966620919,
mahesh.gudivaka@gmail.com.
rajithadenaboina404@gmail.com.

-- ***--

Abstract: - Approximate computing is an efficient approach for error-tolerant applications because it can trade off accuracy for power. Addition is a key fundamental function for these applications. In this paper, we proposed a low-power yet high-speed accuracy-configurable adder that also maintains a small design area. The proposed adder is based on the conventional carry look-ahead adder, and its configurability of accuracy is realized by masking the carry propagation at runtime. Compared with the conventional carry look-ahead adder, with only 14.5% area overhead, the proposed 16-bit adder reduced power consumption by 42.7% and critical path delay by 56.9%, most according to the accuracy configuration settings, respectively. Furthermore, compared with other previously studied adders, the experimental results demonstrate that the proposed adder achieved the original purpose of optimizing both power and speed simultaneously without reducing the accuracy.

--***---

I INTRODUCTION

As the scale of integration keeps growing, more and more sophisticated signal processing systems are being implemented on a VLSI chip. These signal processing applications not only demand great computation capacity but also consume considerable amount of energy. While performance and Area remain to be the two major design tolls, power consumption has become a critical concern in today's VLSI system design[]. The need for low-power VLSI system arises from two main forces. First, with the steady growth of operating frequency and processing capacity per chip, large currents have to be delivered and the heat due to large power consumption must be removed by proper cooling techniques. Second, battery life in portable electronic devices is limited. Low power design directly leads to prolonged operation time in these portable devices.

Addition usually impacts widely the overall performance of digital systems and a crucial arithmetic function. In electronic applications adders are most widely used. Applications where these are used are multipliers, DSP to execute various algorithms like FFT, FIR and IIR. Wherever concept of multiplication comes adders come in to the picture. As we know millions of instructions per second are performed in microprocessors. So, speed of operation is the most important constraint to be considered while designing multipliers. Due to device portability miniaturization of device should be high and power consumption should be low. Devices like Mobile, Laptops etc. require more battery backup.

So, a VLSI designer has to optimize these three parameters in a design. These constraints are very difficult to achieve so depending on demand or application some compromise between constraints has to be made. Ripple carry adders exhibits the most compact design but the slowest in speed. Whereas carry look ahead is the fastest one but consumes more area. Carry select adders act as a compromise between the two adders. In 2002, a new concept of hybrid adders is presented to speed up addition process by Wang et al. that gives hybrid carry look-ahead/carry select adders design. In 2008, low power multipliers based on new hybrid full adders is presented.

There are four factors that influence the power dissipation of CMOS circuits. They are technology, circuit design style, architecture, and algorithm. The challenge of meeting the contradicting goals of high performance and low power system operation has motivated the development of low power process technologies and the scaling of device feature sizes.

II LITERATURE SURVEY

Addition is the most common and often used arithmetic operation on microprocessor, digital signal processor, especially digital computers. Also, it serves as a building block for synthesis all other arithmetic operations. Therefore, regarding the efficient implementation of an arithmetic unit, the binary adder structures become a very critical hardware unit. In any book on computer arithmetic, someone looks that there exists a large number of different circuit architectures with different performance characteristics and widely used in the practice. Although many researches dealing with the binary adder structures have been done, the studies based on their comparative performance analysis are only a few.

This is about a digital circuit. For an electronic circuit that handles analog signals, see mixer. IN electronics, an **adder** or **summer** is a digital circuit that performs the addition of numbers. In many computers and other kinds of processors, adders are used in the arithmetic logic unit(s) and in other parts of the processor, where they are used to calculate addresses, table indices, and similar.

Although adders can be constructed for many numerical 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 a more complex adder. Different type of adders as follows

- **RIPPLE CARRY ADDERS (RCA)**

Concatenating the N full adders forms N bit Ripple carry adder. In this carry out of previous full adder becomes the input carry for the next full adder. It calculates sum and carry according to the following equations. As carry ripples from one full adder to the other, it traverses longest critical path and exhibits worst-case delay. Si = Ai xor Bi xorCi
Ci+1 = Ai Bi + (Ai + Bi) Ci; where i = 0, 1… n-1
RCA is the slowest in all adders (O (n) time) but it is very compact in size (O (n) area). If the ripple carry adder is implemented by concatenating N full adders, the delay of such an adder is 2N gate delays from Cin to Cout. The delay of adder increases linearly with increase in number of bits. Block diagram of RCA is shown in figure 1.

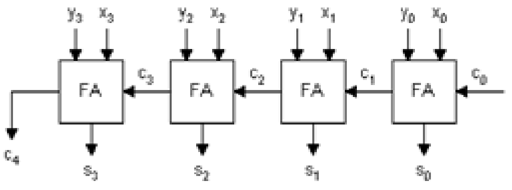
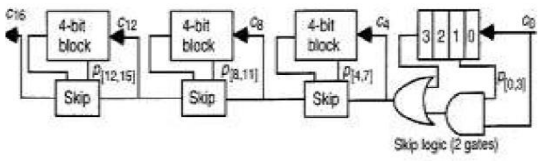


Figure: Block diagram of RCA

- **Carry Skip Adder (CSKA)**

A carry skip divides the words to be added in two groups of equal size of k-bits. Carry Propagate *pi* signals may be used within a group of bits to accelerate the carry propagation.
Block width tremendously affects the latency of adder. Latency is directly proportional to block width. More number of blocks means block width is less, hence more delay. The idea behind Variable Block Adder (VBA) is to minimize the critical path delay in the carry chain of a carry skip adder, while allowing the groups to take different sizes. In case of carry skip adder, such condition will result in more number of skips between stages.



Such adder design is called variable block design, which is tremendously used to fasten the speed of adder. In the variable block carry skip adder design we divided a 32-bit adder in to 4 blocks or groups. The bit widths of groups are taken as; First block is of 4 bits, second is of 6 bits, third is 18 bit wide and the last group consist of most significant 4 bits.
Table 1 shows that the logic utilization of carry skip and variable carry skip 32-bit adder. The power and delay, which are obtained also given in the table1. From table it can be observed that variable block design consumes more area as gate count and number of LUT's consumed by variable block design is more than conventional carry skip adder.

Carry look ahead depends on two things:

1. Calculating, for each digit position, whether that position is going to propagate a carry if one comes in from the right.
2. Combining these calculated values to be able to deduce quickly whether, for each group of digits, that group is going to propagate a carry that comes in from the right.

For each bit in a binary sequence to be added, the Carry Look Ahead Logic will determine whether that bit pair will generate a carry or propagate a carry. This allows the circuit to "pre-process" the two numbers being added to determine the carry ahead of time. Then, when the actual addition is performed, there is no delay from waiting for the ripple carry effect (or time it takes for the carry from the first Full Adder to be passed down to the last Full Adder). Below is a simple 4-bit generalized Carry Look Ahead circuit that combines with the 4-bit Ripple Carry Adder we used above with some slight adjustments:
For the example provided, the logic for the generate (g) and propagate (p) values are given below. Note that the numeric value determines the signal from the circuit above, starting from 0 on the far left to 3 on the far right:

$$
\begin{aligned}
C_1 &= G_0 + P_0 \cdot C_0 \\
C_2 &= G_1 + P_1 \cdot C_1 \\
C_3 &= G_2 + P_2 \cdot C_2 \\
C_4 &= G_3 + P_3 \cdot C_3
\end{aligned}
$$

Substituting C_1 into C_2, then C_2 into C_3, then C_3 into C_4 yields the expanded equations:

$$C_1 = G_0 + P_0 \cdot C_0$$
$$C_2 = G_1 + G_0 \cdot P_1 + C_0 \cdot P_0 \cdot P_1$$
$$C_3 = G_2 + G_1 \cdot P_2 + G_0 \cdot P_1 \cdot P_2 + C_0 \cdot P_0 \cdot P_1 \cdot P_2$$
$$C_4 = G_3 + G_2 \cdot P_3 + G_1 \cdot P_2 \cdot P_3 + G_0 \cdot P_1 \cdot P_2 \cdot P_3 + C_0 \cdot P_0 \cdot P_1 \cdot P_2 \cdot P_3$$

To determine whether a bit pair will generate a carry, the following logic works:

$$G_i = A_i \cdot B_i$$

To determine whether a bit pair will propagate a carry, either of the following logic statements work:

$$P_i = A_i \oplus B_i$$
$$P_i = A_i + B_i$$

2.1 Literature Survey

- **Carry Select Adder**

DESIGN of area- and power-efficient high-speed data path logic systems are one of the most substantial areas of research in VLSI system design. In digital adders, the speed of addition is limited by the time required to propagate a carry through the adder. The sum for each bit position in an elementary adder is generated sequentially only after the previous bit position has been summed and a carry propagated into the next position. The CSLA is used in many computational systems to alleviate the problem of carrying propagation delay by independently generating multiple carriers and then select a carrier to generate the sum. However, the CSLA is not area-efficient because it uses multiple pairs of Ripple Carry Adders (RCA) to generate partial sum and carry by considering carry input cin=0 and cin=1, then the final sum and carry are selected by the multiplexers (mux). The basic idea of this work is to use Binary to Excess-1 Converter (BEC) instead of RCA with cin=1 in the regular CSLA to achieve lower area and power consumption. The main advantage of this BEC logic comes from the lesser number of logic gates than the n-bit Full Adder (FA) structure.

- **Multiplexer**

In electronics, a multiplexer (or MUX) is a device that selects one of several analog or digital input signals and forwards the selected input into a single line.[1] A multiplexer of 2n inputs has n select lines, which are used to select which input line to send to the output.[2] Multiplexers are mainly used to increase the amount of data that can be sent over the network within a certain amount of time and bandwidth.[1] A multiplexer is also called a data selector. They are used in CCTV, and almost every business that has CCTV fitted, will own one of these.

An electronic multiplexer makes it possible for several signals to share one device or resource, for example one A/D converter or one communication line, instead of having one device per input signal.

On the other hand, a demultiplexer (or demux) is a device taking a single input signal and selecting one of many data-output-lines, which is connected to the single input. A multiplexer is often used with a complementary demultiplexer on the receiving end.[1]

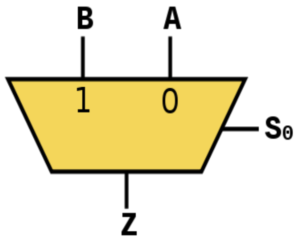
An electronic multiplexer can be considered as a multiple-input, single-output switch, and a demultiplexer as a single-input, multiple-output switch.[3] The schematic symbol for a multiplexer is an isosceles trapezoid with the longer parallel side containing the input pins and the short parallel side containing the output pin.[4] The schematic on the right shows a 2-to-1 multiplexer on the left and an equivalent switch on the right. The wire connects the desired input to the output.

In digital circuit design, the selector wires are of digital value. In the case of a 2-to-1 multiplexer, a logic value of 0 would connect I_0 to the output while a logic value of 1 would connect I_1 to the output. In larger multiplexers, the number of selector pins is equal to $\lceil \log_2(n) \rceil$ where n is the number of inputs.

For example, 9 to 16 inputs would require no fewer than 4 selector pins and 17 to 32 inputs would require no fewer than 5 selector pins. The binary value expressed on these selector pins determines the selected input pin.

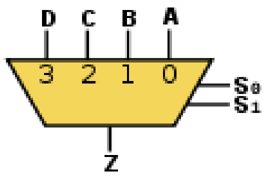
A 2-to-1 multiplexer has a boolean equation where A and B are the two inputs, is the selector input, and Z is the output:

$$Z = (A \cdot \overline{S}) + (B \cdot S)$$



This truth table shows that when $S=0$ then $Z=A$ but when $S=1$ then $Z=B$. A straightforward realization of this 2-to-1 multiplexer would need 2 AND gates, an OR gate, and a NOT gate.

Larger multiplexers are also common and, as stated above, require $\lceil \log_2(n) \rceil$ selector pins for n inputs. Other common sizes are 4-to-1, 8-to-1, and 16-to-1. Since digital logic uses binary values, powers of 2 are used (4, 8, 16) to maximally control a number of inputs for the given number of selector inputs.



4-to-1 mux

The boolean equation for a 4-to-1 multiplexer is:

$$F=\left(A\cdot\overline{S_0}\cdot\overline{S_1}\right)+\left(B\cdot S_0\cdot\overline{S_1}\right)+\left(C\cdot\overline{S_0}\cdot S_1\right)+\left(D\cdot S_0\cdot S_1\right)$$

- **Delay and Area Evaluation of Basic Blocks**

The AND, OR, and Inverter (AOI) implementation of an XOR gate is shown in Fig. below. The gates between the dotted lines are performing the operations in parallel and the numeric representation of each gate indicates the delay contributed by that gate. The delay and area evaluation methodology considers all gates to be made up of AND, OR, and Inverter, each having delay equal to 1 unit and area equal to 1 unit. We then add up the number of gates in the longest path of a logic block that contributes to the maximum delay.

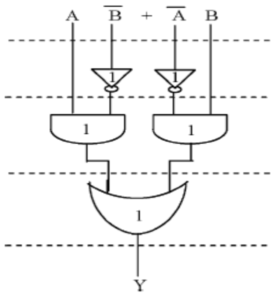


Figure: 2.1 Area and Delay evaluation of xor gate

III PROPOSED METHOD

Many signal processing blocks, especially those meant for video and speech, are error tolerant which makes it possible to use inaccurate arithmetic units. This is exploited in systems to save power and area as well as to reduce the delay. Approximation is mainly done using voltage overscaling and architectural approximation [1], [2]. In voltage over-scaling, supply voltage is scaled down leading to power savings, but causing increased delays. The upper part of the sum containing its most significant bits (MSBs) is obtained using accurate adders. Approximate logic is used to compute the lower part of the sum containing the remaining least significant bits (LSBs). In such approximate adders, for a given number of lower-order bits being approximated, the power consumed by the accurate upper part is almost the same. Power savings in the lower portion is typically due to reduction in the switching activity due to use of simpler gates. The simplest of these adders is the truncation adder, where the lower part is set to zero. Since there is no hardware requirement for the lower part, it has the largest savings in both power and area. However, it also results in significant errors [3]. There are other approximate adders, such as [7], [8] which do not split the output into approximate lower part and accurate upper part. Instead, the adder is divided into many subadders and carry is predicted. Here we can trade off power and accuracy by varying the size of subadders. However, these adders perform worse than the two-part segmented adders in terms of power-accuracy trade-off [9]. In this paper, therefore we focus on segmented adders with two parts – an accurate upper part and an inaccurate lower part. It is possible to match the power savings of the truncation adder if the approximate bits are set to a fixed value. In this paper, we propose to set it to a fixed value L that minimizes the mean error distance (MED). The value is chosen so that it is optimal for all inputs that have a symmetric probability mass function (PMF). If the input PMF is not symmetric, we show that it is close to optimal as long as the number of approximate bits is not too large. We quantify the power savings of various two-part adders' in terms of the power savings per bit and the MED. We have used the proposed adder in an image addition application to demonstrate its effectiveness.

3.1 Proposed Median Approximate Adder (MA)

Our focus in this paper is to try and minimise the MED, while aiming to achieve the low power consumption of the truncation adder. To do this, we need to have fixed values of SL and ck−1 so that, like the truncation adder, there is no hardware to find the lower part of the output. The values of SL and ck−1 are to be chosen such that the MED is minimized. For purposes of analysis, in this section, AH, AL, BH and BL refer to the corresponding decimal representation. In all cases, we assume that both inputs have N bits and k bits of the sum are approximated. The accurate sum therefore, is given by (AH + BH) 2k + (AL + BL).

A. Uniformly distributed inputs

This is the most common assumption for the PMF of the inputs. Assume that A and B have a uniform PMF with values in the range [0, 2N −1]. It is obvious that AL and BL will also have a uniform PMF with values within a range [0, 2k − 1]. Since the PMF of Z = AL + BL is the convolution of the PMFs of AL and BL, it is a triangular distribution with values between 0 and 2k+1 − 2. Since it is a symmetric PMF, the median value is the midway point, namely, 2k−1. If the binary representation of L is ck−1sk−1sk−2 ··· s0, then ck−1 = 0 and sk−1sk−2 ··· s0 = 11 ··· 1.

B. Inputs have a symmetric distribution

Here, it is assumed that the PMFs of both inputs A and B are symmetric, i.e. P (A = Q) = P (A = 2N − 1 − Q) and P (B = Q) = P (B = 2N −1−Q), where 0 <= Q <= 2N −1. In such a setting, we claim that the distribution of AL and BL are also symmetric, i.e. P (AL = Q) = P (AL = 2k − 1 − Q) and P (BL = Q) = P(BL

= 2k −1−Q), where 0 <= Q <= 2k − 1. A proof for this claim is as follows. Divide the range [0, 2N − 1] into 2N−k bins, each of size 2k. The i-th bin is given by [i2k, (i + 1)2k − 1], where 0 ≤ i ≤ 2N−k − 1. In each bin, the most significant N − k bits have a fixed value. For 0 ≤ Q ≤ 2k − 1, we have P(AL = Q) = 2N −k−1 i=0 P(A = 2ki + Q). Hence, P (AL = 2k − 1 − Q) = 2N −k−1 i=0 P(A = 2ki + 2k − 1 − Q). Since the PMF of A is symmetric, we have P(AL = 2k − 1 − Q) = 2N −k−1 i=0 P(A = 2N − 1 − (2ki + 2k − 1 − Q)) = 2N −k−1 i=0 P(A = 2k(2N−k − 1 − i) + Q) = P(AL = Q). The proof for the symmetry of BL is same as the above. If the PMFs of AL and BL are symmetric, the PMF of their sum (which is the convolution of the PMFs of the two inputs) is also symmetric. Since the range of Z is 0 to 2k+1 − 2 and Z is symmetric, the median of Z is 2k − 1

C. PMF of the inputs are arbitrary

In most of the applications seen in the literature, the number of approximate bits rarely exceeds N/2. As before, consider the division of the range [0, 2N − 1] into 2N−k bins, each of size 2k. For small k values, if the distribution of the 2k values within each bin is approximately uniform, then the distributions of AL and BL are also approximately uniform. This means that the PMF of Z = AL + BL is approximately triangular. Therefore, in the setting of small k, the median of Z is closely approximated by 2k − 1. Since most applications are likely to satisfy one of the three cases, the proposed median adder (MA) uses L = 2k − 1. However, if the PMFs of the inputs are known, it is possible to derive the PMFs of the lower k bits, which can then be convolved to obtain the PMF of the lower part sum. In this case, the median can be obtained exactly and L can be set to this value. D. Accuracy metrics for Median adder for uniformly distributed inputs An expression for MED of the median adder (E{|Z −L|}) with uniformly distributed inputs can be derived as follows. MED = 2 k−1 i=0 iP(|Z − L| = i) = 2 2 k−1 i=0 i 2k − i 22k = 2k 3 − 2−k 3 . (1) The expressions for NMED and NED are obtained using suitable normalization factors as follows. NMED = 2k − 2−k 3 · (2N+1 − 2) and NED = 1 − 2−2k 3 . The expression for MED and NED of the truncation adder are 2k − 1 and 1 − 2−k respectively. Clearly, MA is much better than truncation in terms of both metrics, while having the same power savings.

3.2 Extension Method

By exploiting the inherent tolerance feature, approximate computing can be adopted for a trade-off between accuracy and power. At present, this trade-off plays a significant role in such application domains [3]. As computation quality requirements of an application may vary significantly at runtime, it is preferable to design quality configurable systems that are able to trade off computation quality and computational effort according to application requirements [4] [5]. The previous proposals for configurability suffer the cost of the increase in power [5] or in delay [12]. We implemented the proposed adder, the conventional carry look-ahead adder (CLA), and the ripple carry adder (RCA) in Verilog HDL using a 45-nm library. Then we evaluated the power consumption, critical path delay, and design area for each of these implementations. Compared with the conventional CLA, with 1.95% mean relative error distance (MRED), the proposed adder reduced power consumption and critical path delay by 42.7% and 56.9%, respectively. We provided a crosswise comparison to demonstrate the superiority of the proposed adder. Moreover, we implemented two previously studied configurable adders to evaluate power consumption, critical path delay, design area, and accuracy. We also evaluated the quality of these accuracy configurable adders in a real image processing application.

3.3 Proposed Accuracy-Configurable Adder

Typically, a CLA consists of three parts: (1) half adders for carry generation (G) and propagation (P) signals preparation, (2) carry look-ahead units for carry generation, and (3) XOR gates for sum generation. We focus on the half adders for G and P signals preparation in part 1.

Consider an n-bit CLA; each part of it can be obtained as follows:

P□ = A□⊕B□, G□ = A□ · B□, (1) C□ = G□ + P□ · C□ି଼, (2) S□ = P□⊕C□ି଼. (3) Where i is denoted the bit position from the least significant bit. Note that owing to reuse of the circuit of Ai XOR Bi for Si generation, here Pi is defined as Ai XOR Bi instead of Ai OR Bi. Because C0 is equal to G0 , if G0 is 0, C0 will be 0. From (2), we find that C1 is equal to G1 when C0 is 0. In other words, if G0 and G1 are equal to 0, C0 and C1 will be 0. By expanding the above to i , Ci will be 0 when G0, G1, … , Gi are all 0. Evidently, we can achieve selectivity by adding a select signal. Figure 1(a) is a conventional half adder and Fig. 1(b) is a half adder to which the select signal has been added. The dashed frame represents the equivalent circuit of a 2-input XOR (M_X = 1). We can obtain the following: P is equal to A XOR B, and G is equal to A AND B when M_X = 1; when M_X = 0, P is equal to A OR B and G is 0. Thus, M_X can be considered as a carry mask signal.

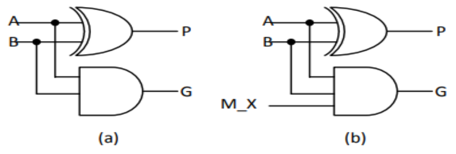


Fig. 1. (a) An accurate half adder, and (b) a half adder with a select signal.

Consider an n-bit CLA, whose half adders for G and P signals preparation are replaced by CMHAs. In this case, an nbit carry mask signal for each CMHA is required. To simplify the structure for masking carry propagation, we group four CMHAs and use a 1-bit mask signal to mask the carry propagation of the CMHAs in each group. The structure of a group with four CMHAs is shown in Fig. 3 as an example. A3-0, B3-0, P3-0, and G3-0 are 4-bit-length signals and represent {A3, A2, A1, A0}, {B3, B2, B1, B0}, {P3, P2, P1, P0}, and {G3, G2, G1, G0,}, respectively. M_X0 is a 1-bit signal and is connected to the four CMHAs to mask the carry propagation simultaneously. When M_X0 = 1, P3-0 = A3-0 XOR B3-0, and G3-0 = A3-0 AND B3- 0; when M_X0 = 0, P3-0 = A3-0 OR B3-0, and G3-0 = 0. We proposed an accuracy-configurable adder by using CMHAs to mask the carry propagation.

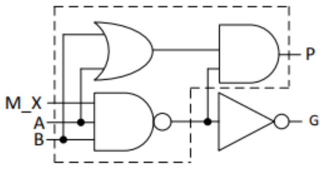


Fig. 2. A carry-maskable half adder.

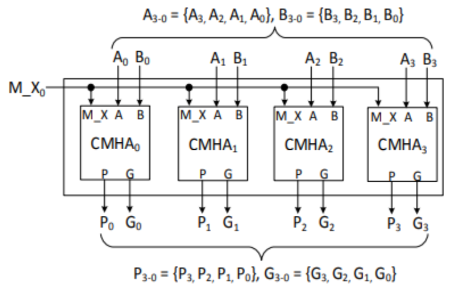

$A_{3\text{-}0}$ = {A_3, A_2, A_1, A_0}, $B_{3\text{-}0}$ = {B_3, B_2, B_1, B_0}
M_X_0
CMHA0
CMHA1
CMHA2
CMHA3
$P_{3\text{-}0}$ = {P_3, P_2, P_1, P_0}, $G_{3\text{-}0}$ = {G_3, G_2, G_1, G_0}

Fig. 3. Structure of a group with four CMHAs.

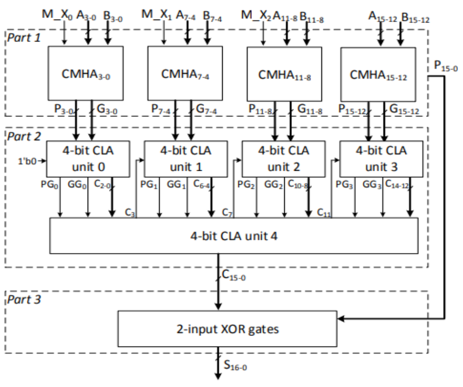


Fig. 4. Structure of the proposed 16-bit adder.

IV APPLICATIONS

In complicated data path system, multiplier is considered as a much bigger component in power consumption. Our carry prediction-based approximation uses generate bit to predict the carry from lower segments. The critical delay can be restrained to a smaller value with shorter critical path in carry propagation. Further extension of our technique to multiplier depends on the multiplication structure used in hardware implementation. There is a variety of hardware designs for multiplication, according to the structures of reduction tree. In this section, we apply our technique on three kinds of multiplication structures including array multiplier, Wallace multiplier, and Dadda multiplier.

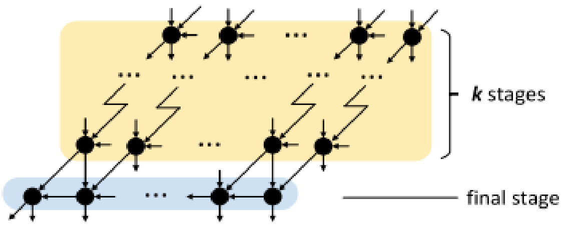


Figure 4.1 Basic structure of multiplier.

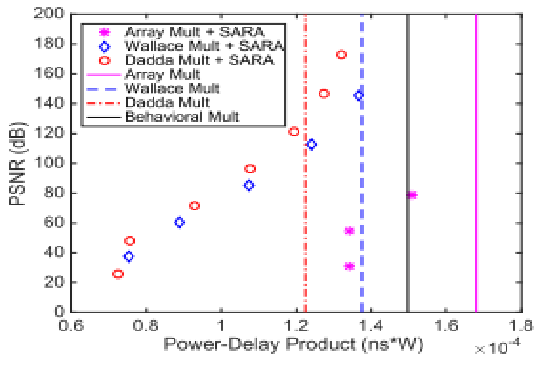


Figure 4.2 Multiplier: PSNR versus PDP.

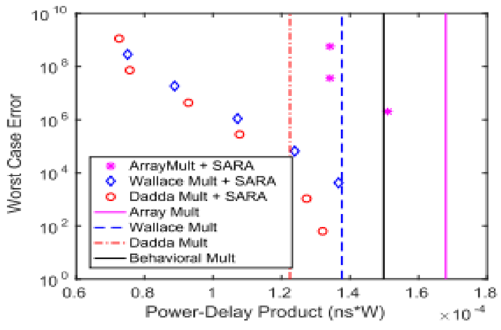


Figure 4.3 Multiplier: worst case error versus PDP.

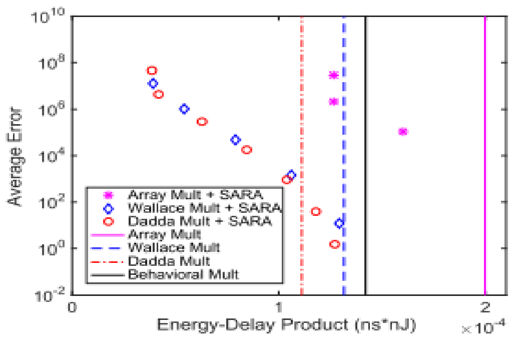


Figure 4.4 Multiplier: average error versus EDP.

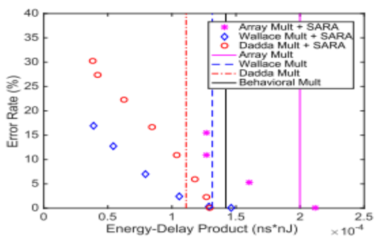


Figure 4.5 Multiplier: error rate versus EDP

Step 1) Generate all partial products by using an AND gate array.
Step 2) Combine the partial products in k stages by layers of half/full adder until the matrix height is reduced to two. Different types of structures depend on the reduction tree used to reduce the number of partial products in this step.
Step 3) Sum the resulting numbers in the final stage by a conventional adder. In array multiplier, the carry bits in one stage are propagated diagonally downward, which follows the basic shift-and-add multiplication algorithm.
Wallace multiplier based on Wallace tree combines the partial products as early as possible, which makes it faster than array multiplier.
Total error increases as more bits are configured in approximate mode. Approximate array multiplier shows larger error than approximate Wallace/Dadda multiplier at the same PDP level. It is because array multiplier has larger critical delay from internal stages in step 2 than Wallace/Dadda multiplier. Figs. 25 and 26 show the error versus EDP for both accurate and approximate multipliers. As more MUXes are set to propagate approximate carry, the average error in output increases to about 107, which as well achieves the best EDP. The worst case error rate of approximate Dadda multiplier is about 30%, while it comes to about 17% for approximate array multiplier and Wallace multiplier. As shown in Fig. 26, when approximate multipliers are working in completely accurate mode (error rate equals 0), EDP is larger than that of their accurate counterpart. In summary, the experimental results show that our technique can be successfully extended to highspeed multiplier designs.

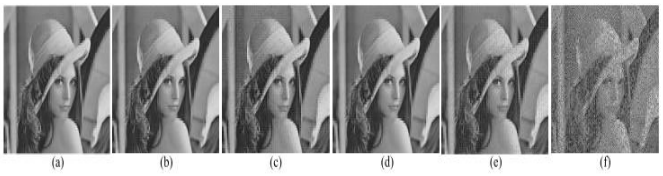


Figure 4.6 Comparison of image lenna. (a) Accurate adder. (b) SARA4. (c) SARA8. (d) SARA4-DAR2. (e) GDA. (f) RAP-CLA.

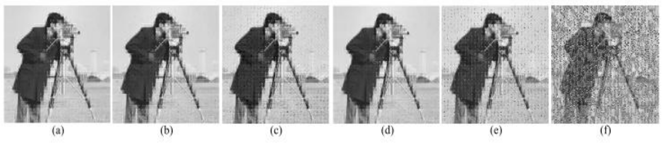


Figure 4.7 Comparison of image cameraman. (a) Accurate adder. (b) SARA4. (c) SARA8. (d) SARA4-DAR2. (e) GDA. (f) RAP-CLA.

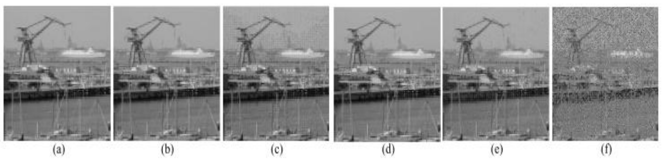


Figure 4.7 Comparison of image kiel. (a) Accurate adder. (b) SARA4. (c) SARA8. (d) SARA4-DAR2. (e) GDA. (f) RAP-CLA.

And due to the simple but effective structure of SARA, it provides an easy way for us to convert conventional multiplier into approximate design.

B. DCT Computation in Image Processing

DCT has been recognized as the basic in many transform coding methods for image and video signal processing. It is used to transform the pixel data of image or video into corresponding coefficients in frequency domain. Since human visual system is more sensitive to the changes in low frequency, the loss of accuracy in high-frequency components does not heavily degrade the quality of image processed by DCT. In addition, those components in different frequency have different tolerances to the degradation in the original data. It is a good example to show the reconfigurability of our design by applying them in VLSI implementation of DCT, computing in JPEG image compression.

We replace the adders in circuits with different configurations of SARA, SARA-DAR, GDA, as well as RAP-CLA. The results are obtained by numerical simulations on four images (Fig. 4.6-4.7) in MATLAB. As we know, after DCT process, data in different frequency domain have a different level of error tolerance. As shown in Fig. 31, matrix components in the top-left corner correspond to lower frequency coefficients that are sensitive to human vision, while those components in bottom-right corner might allow more errors. To utilize this feature for better energy-accuracy trade-off, we make following configuration for different designs. 1) SARA4: SARA4 with 4, 3, 2, 1 consecutive segments working in accurate mode are used to compute components in S1, S2, S3, and S4, respectively.

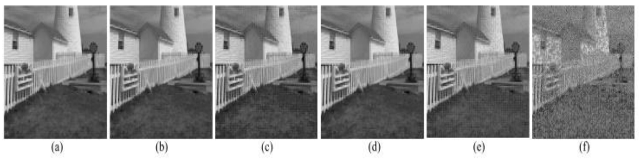


Figure 4.7 Comparison of image house. (a) Accurate adder. (b) SARA4. (c) SARA8. (d) SARA4-DAR2. (e) GDA. (f) RAP-CLA.

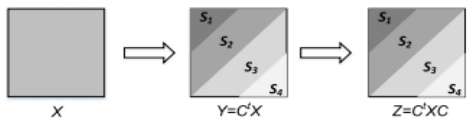


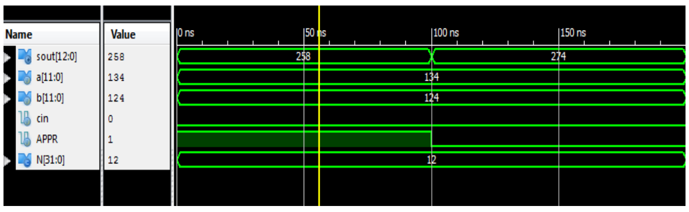
Figure 4.8 2D DCT

- Where MAXI is the maximum pixel value of the image.

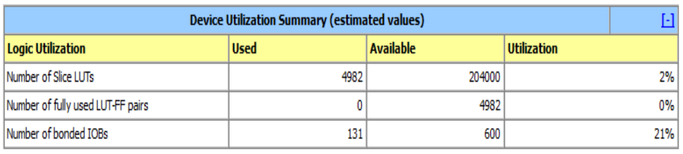
SARA4-DAR2 has the highest PSNR for every image among all configurable adders, which is close to the quality of accurate adder. Comparing SARA8 with GDA, they have similar PSNR and similar delay, but SARA8 has less power consumption according to the analysis in Section VI. SARA4-

DAR2 achieves better image quality than SARA4, but might result in more power due to additional logics for self-configuration. The image quality for different adders in DCT computing can also be demonstrated in Figs. 27–30. According to human vision, SARA and its DAR counterpart show better image quality than GDA and RAP-CLA in JPEG compression processing.

V SIMULATION RESULT



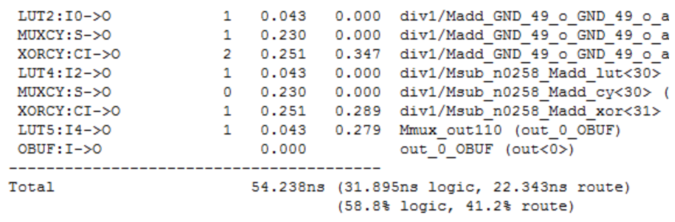
5.2 Design Summary



Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice LUTs	4982	204000	2%
Number of fully used LUT-FF pairs	0	4982	0%
Number of bonded IOBs	131	600	21%

The above result represents the synthesis implementation by using the Xilinx ISE software. From the above table, it is observed that only 4982 look up tables are used out of available204000. It indicates very less area (2%) was used for the proposed design.

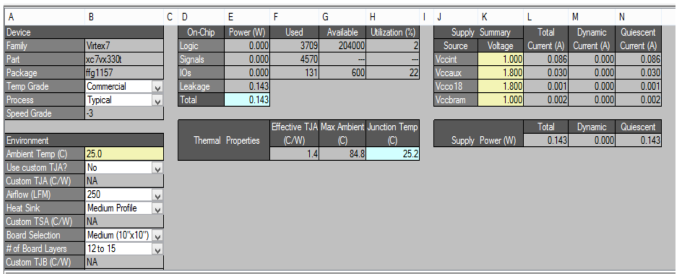
5.3 Time Summary



```
  LUT2:I0->O           1   0.043   0.000  div1/Madd_GND_49_o_GND_49_o_a
  MUXCY:S->O           1   0.230   0.000  div1/Madd_GND_49_o_GND_49_o_a
  XORCY:CI->O          2   0.251   0.347  div1/Madd_GND_49_o_GND_49_o_a
  LUT4:I2->O           1   0.043   0.000  div1/Msub_n0258_Madd_lut<30>
  MUXCY:S->O           0   0.230   0.000  div1/Msub_n0258_Madd_cy<30> (
  XORCY:CI->O          1   0.251   0.289  div1/Msub_n0258_Madd_xor<31>
  LUT5:I4->O           1   0.043   0.279  Mmux_out110 (out_0_OBUF)
  OBUF:I->O                0.000          out_0_OBUF (out<0>)
 ----------------------------------------
 Total                    54.238ns (31.895ns logic, 22.343ns route)
                                   (58.8% logic, 41.2% route)
```

The above result represents the time consumed such as path delays by using the Xilinx ISE software. The consumed path delay is 54.238ns.

5.4 Power Summary



A	B
Device	
Family	Virtex7
Part	xc7vx330t
Package	ffg1157
Temp Grade	Commercial
Process	Typical
Speed Grade	-3
Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	250
Heat Sink	Medium Profile
Custom TSA (C/W)	NA
Board Selection	Medium (10"x10")
# of Board Layers	12 to 15
Custom TJB (C/W)	NA

On-Chip	Power (W)	Used	Available	Utilization (%)
Logic	0.000	3709	204000	2
Signals	0.000	4570	---	---
IOs	0.000	131	600	22
Leakage	0.143			
Total	0.143			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	1.4	84.8	25.2

Supply Summary		Total	Dynamic	Quiescent
Source	Voltage	Current (A)	Current (A)	Current (A)
Vccint	1.000	0.086	0.000	0.086
Vccaux	1.800	0.030	0.000	0.030
Vcco18	1.800	0.001	0.000	0.001
Vccbram	1.000	0.002	0.000	0.002

	Total	Dynamic	Quiescent
Supply Power (W)	0.143	0.000	0.143

The above result represents the power consumed by using the Xilinx ISE software. The consumed power is 0.143uw.

VI CONCLUSION

In this paper, an accuracy-configurable adder without suffering the cost of the increase in power or in delay for configurability was proposed. The proposed adder is based on the conventional CLA, and its configurability of accuracy is realized by masking the carry propagation at runtime. The experimental results demonstrate that the proposed adder delivers significant power savings and speedup with a small area overhead than those of the conventional CLA. Furthermore, compared with previously studied configurable adders, the experimental results demonstrate that the proposed adder achieves the original purpose of delivering an unbiased optimized result between power and delay without sacrificing accuracy. It was also found that the quality requirements of the evaluated application were not compromised.

VII FUTURE SCOPE

In this paper, we propose an SARA design. It has significantly lower power/EDP than the latest previous work when comparing at the same accuracy level. In addition, SARA has considerable lower area overhead than almost all the previous works. The accuracy-power-delay efficiency is further improved by a DAR technique. We demonstrate the efficiency of our adder in the applications of multiplication circuits and DCT computing circuits for image processing.

REFERENCES

[1] Kuldeep Rawat, Tarek Darwish and Magdy Bayoumi, “A low power and reduced area Carry Select Adder”, 45th Midwest Symposium on Circuits and Systems, vol.1, pp.

[2] Y. Kim and L.-S. Kim, "64-bit carry-select adder with reduced area,” Electron. Lett. vol. 37, no. 10, pp. 614-615, May 2001.

[3] J. M. Rabaey, Digtal Integrated Circuits-A Design Perspective.Upper Saddle River, NJ:.

[4] Cadence, "Encounter user guide,” Version 6.2.4, March 2008.

[5] R. Priya and J. Senthil Kumar, “Enhanced area efficient architecture for 128 bit Modified CSLA”, International Conference on Circuits, Power and Computing Technologies, 2013.

[6] Shivani Parmar and Kirat pal Singh,”Design of high speed hybrid carry select adder”, IEEE

[7] I-Chyn Wey, Cheng-Chen Ho, Yi-Sheng Lin, and Chien-Chang Peng,” An Area-Efficient Carry Select Adder Design by Sharing the Common Boolean Logic Term”, Proceedings of the International MultiConference of Engineers and Computer Scientist 2012 Vol II,IMCES 2012,HongKong,March 14-16 2012.

[8] Ramkumar and Harish M Kittur," Low-Power and Area-Efficient Carry Select Adder", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, VOL. 20, NO.

[9] Ms. S.Manjui, Mr. V. Sornagopae," An Efficient SQRT Architecture of Carry Select Adder Design by Common Boolean Logic",IEEE, 2013.

[10] Youngjoon Kim and Lee-Sup Kim, "64-bit carry-select adder withreduced area", Electronics Letters, vol.37, issue 10, pp.614-615, May 2001.

[11] Yajuan He, Chip-Hong Chang and Jiangmin Gu, "An area efficient 64-bit square root Carry-Select Adder for low power applications", IEEE International Symposium on Circuits and Systems,vol.4, pp.4082-4085,

[12] Youngjoon Kim and Lee-Sup Kim, "A low power carry select adder with reduced area", IEEE International Symposium on Circuits and Systems, vol.4, pp.218-221, May 2001.

[13] Hiroyuki Morinaka, Hiroshi Makino, Yasunobu Nakase, Hiroaki Suzuki and Koichiro Mashiko, "A 64bit Carry Look-ahead CMOS Adder using Modified Carry Select",Proceeding of IEEE on Custom IntegratedCircuits Conference, pp.585-588.