A Fixed Delay Infinite-Bit Split Adder Architecture and Its Application in Real-Time Image Processing

Amjad Fuad Hajjar
Department of Electrical & Computer Engineering
King Abdulaziz University
PO Box 80204 Jeddah 21589 Saudi Arabia
Fax: +966 2 695-2686
Email: ahajjar@kau.edu.sa

***Abstract*—A fixed delay split adder is presented. The adder breaks the total addition into *sum* and *reminder* with an expected reminder percentage of about 0.33% of the total sum. The adder is capable of producing the outputs in 6 gate delays regardless of the inputs bit-size. The proposed adder is applied to two real-time image processing applications and results showed very close matching to the perfect cases when ignoring the reminder.**

***Index Terms*—approximate addition, binary adders, computer arithmetic, high-speed adders, image processing.**

I. Introduction

BINARY adders have been discussed comprehensively in the open literature for many years. Depending on the desired application, criteria such as speed, power, and hardware complexity (gate count or silicon area) are employed in choosing the optimum architecture. When area is an issue, sequential adders of small block sizes are preferred over the parallel architectures where the carry signals are computed all at once. Most of the other existing architectures for binary addition, in general, lay between these two extremes.

For applications of high computational complexities and high speed processing necessities, fast and yet small sized adders are desirable. Fortunately, some of these applications do not require perfect outcomes, especially when dealing with noise, filtering or human hearing or vision.

This paper presents a binary adder capable of adding two input operands in one clock cycle regardless of their bit size; yet the adder has a very small area (or gate count) compared to the existing adders. The following section briefly summarizes the major existing binary adders. Section III presents the proposed method of *splitting* the addition operation into a *sum* and a *reminder*, and describes the development of the stages as well as their statistics. The proposed adder is utilized in some image processing applications and results are shown in Section IV. Finally some concluding remakes are noted.

Manuscript received May 30, 2006.

A. F. Hajjar is with the Department of Electrical & Computer Engineering, King Abdulaziz University, Saudi Arabia. (email: ahajjar@kau.edu.sa)

II. Literature Review

Although the idea of using *approximate* addition has not been suggested in literature, it is rather beneficial to discuss some of the existing architectures, particularly in terms of delay and gate count: The well known Ripple adder is formed by cascading *n full adders* for an *n*-bit addition operation [1]. The adder complexity is *6n* gates, which is the least among all other adders; however, its delay to produce all sum bits is the worst since the last carry signal must wait 2*n* gate delay times.

Logarithmic time parallel adders are considered the fastest and can be categorized into two classes: *carry look-ahead* and *conditional-sum* algorithms [4]. The carry look-ahead adder computes, typically, four bits at a time in one block and propagates the fifth carry to the next stage. For a 16-bit adder for example, the delay is only 8 gate delays (instead of 32 for the ripple adder) trading more hardware complexity.

An even faster architecture is the *carry select adder*, where it basically duplicates the stages with one having '0' carry-in and the other having '1' in order to prepare the output faster. The output signals are then multiplexed based on the actual carry of each stage. This algorithm saves one gate delay for every 4-bit block trading of course hardware complexity.

Many variations and/or modifications to the aforementioned algorithms can be found in literature, e.g. [2,3,5,6,7]. Yet, the fastest algorithm still suffers the carry signal propagation especially when spanning the inputs to larger bit sizes.

III. The Split Adder

A. Basic Split Full Adder (SFA)

Consider an ordinary full adder where two bits, a_i and b_i, and a carry-in, c_i, are to be added. The sum and carry-out of this stage are:

$$
\begin{aligned}
s_i &= c_i \oplus a_i \oplus b_i \\
c_{i+1} &= a_i b_i + c_i (a_i + b_i)
\end{aligned}
\tag{1}
$$

Obviously c_{i+1} depends on the previous carry bit c_i and that's why the carry signal propagates all the way to the most significant bit causing that long delay. Nevertheless, if we *split* the addition of the full adder into a *sum* and a *reminder* such that the carry-out does not depend on the carry-in and the reminder is minimal, then the propagation problem is solved. Table I shows the truth table of the suggested split full adder (SFA) compared to the ordinary full adder.

TABLE I
SPLIT FULL ADDER TRUTH TABLE

			FA			SFA		
c_i	a_i	b_i	c_{i+1}	s_i		c_{i+1}	s_i	r_i
0	0	0	0	0		0	0	0
0	0	1	0	1		0	1	0
0	1	0	0	1		0	1	0
0	1	1	1	0		1	0	0
1	0	0	0	1		0	1	0
1	0	1	1	0	*	0	1	1
1	1	0	1	0	*	0	1	1
1	1	1	1	1		1	1	0

For the SFA, there are two cases out of eight where a reminder of size one exists. Thus, there is a 25% chance that the sum does not equal to the total addition. When mapping the outputs of a single SFA into Boolean equations, we find:

$$
\begin{aligned}
s_i &= c_i + (a_i \oplus b_i) \\
c_{i+1} &= a_i b_i \\
r_i &= c_i \cdot (a_i \oplus b_i)
\end{aligned}
\tag{2}
$$

A more important question is: for a given n-bit cascaded split adder, what is the probability that there will be a reminder of greater than zero, and what is the expected size of the reminder compared to the total sum? One might run an exhaustive simulation that covers all possible values for some inputs A and B in the range 0 to $2^n\!-\!1$ to extract the sum and the reminder, S and R, and to calculate the statistics of the reminder. Table II summarizes the mean, standard deviation, the maximum reminder percentage to the total sum, and the probability of having a reminder for different values of n.

Fig. 1 shows the gate level implementation of a signal stage SFA, and Fig. 2 shows the block diagram of an n-bit split adder. The total number of gates of this adder is thus $4n$ with a fixed delay of 2 gates to generate the outputs.

TABLE II
STATISTICAL ANALYSIS OF N-BIT SPLIT ADDER

n	8	10	12	14	16
μ	11.89%	11.95%	11.96%	11.97%	11.97%
σ	15.62%	15.56%	15.55%	15.54%	15.54%
max	50%	50%	50%	50%	50%
P{R>0}	65.99%	75.22%	81.95%	86.85%	90.42%

B. *Second Stage to SFA*

The results of Table II shows that if the reminder is ignored and only the sum is considered, the expected error is about 12% with a standard deviation of 15.5% and a maximum of 50% in the worst case. It also shows that it is almost certain that there will be a reminder of greater than zero and that the sum is less than the accurate addition. To improve these figures, a second stage is suggested such that the outputs from every two blocks of the first stage are considered. Here we are trying to adjust the sum bits of the first stage to better values by adding the associated reminders keeping in mind the carry propagation problem. Let the inputs to the second stage blocks be [s_{i+1} s_i] and [r_{i+1} r_i], and the outputs be [s^+_{i+1} s_i^+]. When considering two blocks of stage 1, there are four possible cases for the reminder vector: "00", "01", "10", and "11" of which the last three should fix the sum. Note that from Equation (2), it is impossible to have a reminder bit of '1' with a '0' sum bit. Thus, for the reminder of "01", the sum of the first stage is of the form [s_{i+1} 1]. Adding these vectors together yields a carry-out bit to a next stage. Consequently, the next second stage has an extra carry-in bit that should be considered in the addition process. Of course, this new carry sequence should not propagate through the successive stages; otherwise, the goal of this work is violated. A truth table is constructed to consider all possible cases for this stage. Table III shows all the different cases of this stage.

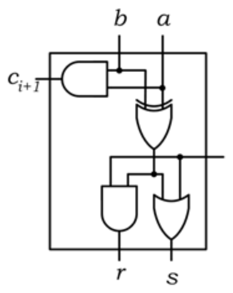


Fig. 1: The gate level implementation of a single stage Split Full Adder

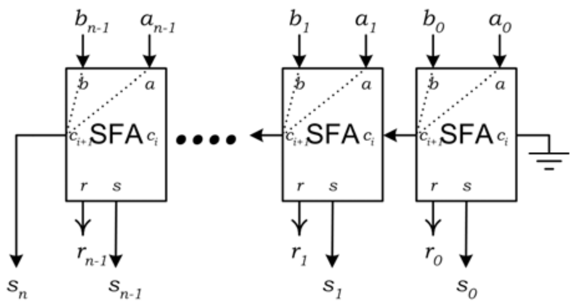


Fig. 2: An n-bit Split Adder

Luckily, one case should be modified to prevent carry propagation. It is when the input sum is "11", no reminder from the first stage, and there is a carry-in bit of '1' from a previous block. The output sum should be "100"; however, it is reduced to "11" with a reminder vector of "01", and a carry-out bit of '0'. The outputs are then:

$$
\begin{aligned}
s^+_{i+1} &= sc_i \cdot \overline{r}_i \cdot s_i + (r_i \oplus s_{i+1} \cdot \overline{r}_{i+1}) \\
s^+_i &= sc_i \cdot \overline{r}_{i+1} \cdot s_{i+1} + (sc_i \oplus s_i \cdot \overline{r}_i) \\
sc_{i+1} &= r_{i+1} + r_i \cdot s_{i+1} \\
r^+_i &= sc_i \cdot s_{i+1} \cdot s_i \cdot \overline{r}_{i+1} \cdot \overline{r}_i
\end{aligned}
\tag{4}
$$

Top

TABLE III
SECOND STAGE TRUTH TABLE

C	S		R			C+S+R		
sc_i	s_{i+1}	s_i	r_{i+1}	r_i		sc_{i+1}	s^+_{i+1}	s^+_i
0	0	0	0	0		0	0	0
0	0	0	0	1		x	x	x
0	0	0	1	0		x	x	x
0	0	0	1	1		x	x	x
0	0	1	0	0		0	0	1
0	0	1	0	1		0	1	0
0	0	1	1	0		x	x	x
0	0	1	1	1		x	x	x
0	1	0	0	0		0	1	0
0	1	0	0	1		x	x	x
0	1	0	1	0		1	0	0
0	1	0	1	1		x	x	x
0	1	1	0	0		0	1	1
0	1	1	0	1		1	0	0
0	1	1	1	0		1	0	1
0	1	1	1	1		1	1	0
1	0	0	0	0		0	0	1
1	0	0	0	1		x	x	x
1	0	0	1	0		x	x	x
1	0	0	1	1		x	x	x
1	0	1	0	0		0	1	0
1	0	1	0	1		0	1	1
1	0	1	1	0		x	x	x
1	0	1	1	1		x	x	x
1	1	0	0	0		0	1	1
1	1	0	0	1		x	x	x
1	1	0	1	0		1	0	1
1	1	0	1	1		x	x	x
1	**1**	**1**	**0**	**0**	***miss***	**0**	**1**	**1**
1	1	1	0	1		1	0	1
1	1	1	1	0		1	1	0
1	1	1	1	1		1	1	1

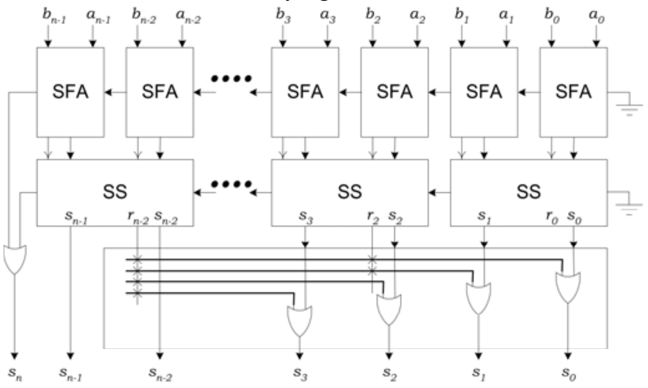
The simulation results of this modified split adder are of much better statistics. On the average, the expected reminder percentage is less than 2% with a standard deviation of about 5.3%. The maximum possible error in this case is 25%, and depending on the adder size, n, the probabilities of having a reminder are much lower. See Table IV.

TABLE IV
STATISTICAL ANALYSIS OF THE MODIFIED N-BIT SPLIT ADDER

n	8	10	12	14	16
μ	1.55%	1.60%	1.61%	1.61%	1.61%
σ	5.35%	5.34%	5.34%	5.34%	5.33%
max	25%	25%	25%	25%	25%
P{*R*>0}	12.40%	16.80%	20.97%	24.94%	28.71%

C. An OR Network Stage

A further improvement to the previous stage could be made. One might utilize the reminder bits of the second stage to improve the lower significant bits of the sum. Suppose that a reminder bit at the i^{th} position is r_i^+ = '1', then the total sum is missing a value of at least 2^i. If all the least significant bits of the sum s_j^+ for j=0 to i-1 are converted to '1's, then the sum will be closer to the correct total addition result (the best case is when all the sum bits are zeros). Consider the example where s^+= "1010" and r^+= "0010". The sum should be $s^+ + r^+$ = "1100". However, if we convert the last two bits of the sum to '1's then s^+= "1011", which is a closer value to the total addition of "1100". Fig. 3 shows the architecture of the split adder with this extra modifying OR network.



b_{n-1} a_{n-1} b_{n-2} a_{n-2} b_3 a_3 b_2 a_2 b_1 a_1 b_0 a_0
SFA
SFA
SFA
SFA
SFA
SFA
SS
SS
SS
s_{n-1} r_{n-2} s_{n-2}
s_3 r_2 s_2
s_1 r_0 s_0
s_n s_{n-1} s_{n-2} s_3 s_2 s_1 s_0

Fig. 3: An n-bit Split Adder with the modifying stage and an OR network

The statistical results of the adder with an OR network are shown in Table V. The average mean value of the reminder percentage is about 0.33% with a standard deviation of 1.23%. In this architecture, the maximum percentage error is about 11% and the probabilities of having a reminder for different values of n are, of course, the same as in the previous stage.

TABLE V
STATISTICAL ANALYSIS OF THE MODIFIED N-BIT SPLIT ADDER WITH AN OR NETWORK

n	8	10	12	14	16
μ	0.32%	0.33%	0.33%	0.33%	0.33%
σ	1.23%	1.23%	1.23%	1.23%	1.23%
max	10.84%	11.04%	11.09%	11.11%	11.11%
P{*R*>0}	12.40%	16.80%	20.97%	24.94%	28.71%

IV. SPLIT ADDER APPLICATIONS IN IMAGE PROCESSING

There are many real-time applications that require fast addition operations, yet their outputs can be approximated. The proposed split adder has a delay of three blocks (or 6 gates) regardless of the inputs bit-size; therefore, it can be used for such applications. Noise reduction in images is one of the well known applications in image processing. For a given noisy image, each pixel is replaced by the average of its 9 neighboring pixels; hence the noise standard deviation is reduced by a factor of $\sqrt{9}=3$. Fig. 4 shows an example of smoothing a noisy image. The original image is first shown followed by the smoothed one using a perfect adder. The proposed split adder with its three stages is used for every addition operation involved in the averaging process, and the intermediate results of each stage are shown. Of each stage, the error histograms are also shown. The error mean value of the first stage of this example is 26% which is very high, as can also be seen. When the second stage is installed, the error percentage is reduced to 4.4% and one may argue a better image is produced. Finally, when installing the OR network to the split adder, the average error percentage is dropped to 1.3%, and a much better image is produced.

Top

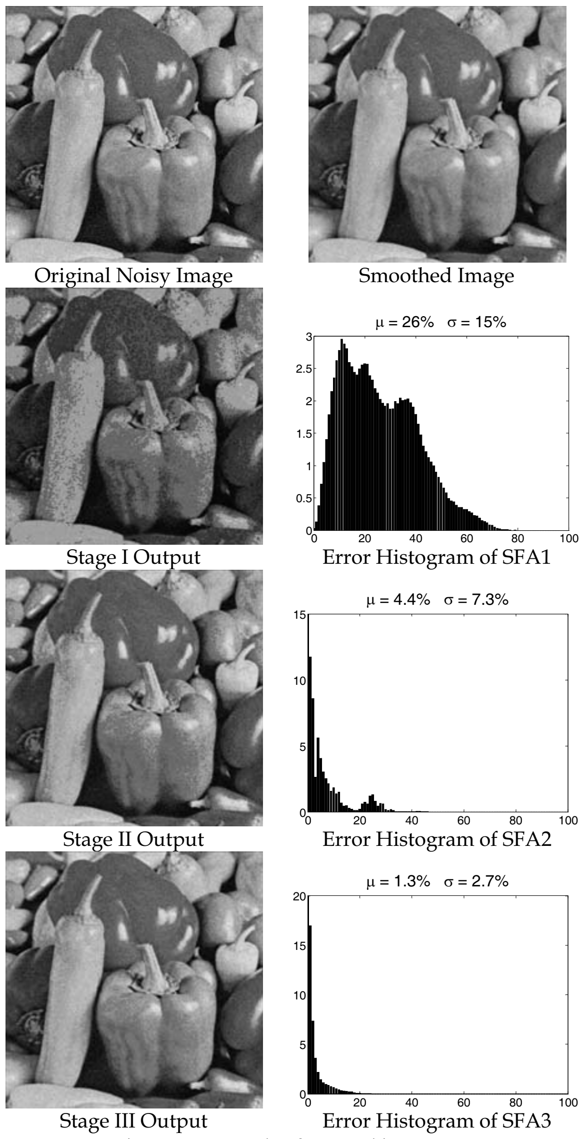


Fig. 4: An example of a smoothing process

Another application is image magnification, or zooming. This involves interlacing the original small image by blank columns and rows, hence doubling the size of the image. Then an averaging process is followed to smooth out the zero pixels. The averaging however is not uniform, and the surrounding pixels intensities are weighted as shown in Fig. 5.

¼	½	¼
½	1	½
¼	½	¼

Fig. 5: Pixels Weights in Zooming Convolution

Fig. 6 shows an example of an image zooming. The original 400×400 pixels image is shown followed by the perfectly enlarged image. The outputs of the different stages of the split adder are also shown. This example has better statistics (12%, 0.78%, and 0.17%); the final output of the split adder has an average error percentage of 0.17%, and even the second stage output of 0.78% error may be sufficient.

V. Conclusion

We presented a new idea of a binary adder capable of computing the sum in one clock cycle regardless of the operands bit size. The adder trades off speed and hardware complexity with the accuracy of the result. On the average, the output sum has an error of about 0.33% of the accurate sum with a standard deviation of 1.23%. The adder also produces the reminder bits, if needed, along with the sum.

The proposed adder is experimented in two applications: image smoothing and image enlargement. Results show that the approximated additions yield very small error percentages, and the output images are very close to the perfect cases.

References

[1] R. Katz, *Contemporary Logic Design*. Benjamin/Cummings Inc., pp. 249-257, 1992

[2] J. Lo, "A Fast Binary Adder with Conditional Carry Generation," *IEEE Tran. on Computers,* v. 46, n. 2, pp. 248-253, 1997

[3] C. Yu, C. Lin, B. Liu, "A Generalized Block Distribution Algorithm for Fast Carry-Skip Adder Design," *IEEE Region 10 Conf. TENCON,* v. 2 pp. 844-847, 1999

[4] W. Yeh, C. Jen, "On the Study of Logarithmic Time Parallel Adders," *IEEE workshop on Signal Processing Systems*, pp. 459-466, 2000.

[5] J. Bruguera, T. Lang, "Multilevel Reverse-Carry Adder," *Int'l Conf. on Computer Design,* pp. 155-162, 2000

[6] H. Vergos, C. Efstathiou, D. Nikolos, "High Speed Parallel-Prefix Modulo 2^n+1 Adders for Diminished-One Operands," *15th IEEE Symposium on Computer Arithmetic,* pp. 211-217, 2001

[7] J. Um, "An Optimal Allocation of Carry-Save Adders in Arithmetic Circuits," *IEEE Tran. on Computers,* v. 50, n. 3, pp. 215-233, 2001

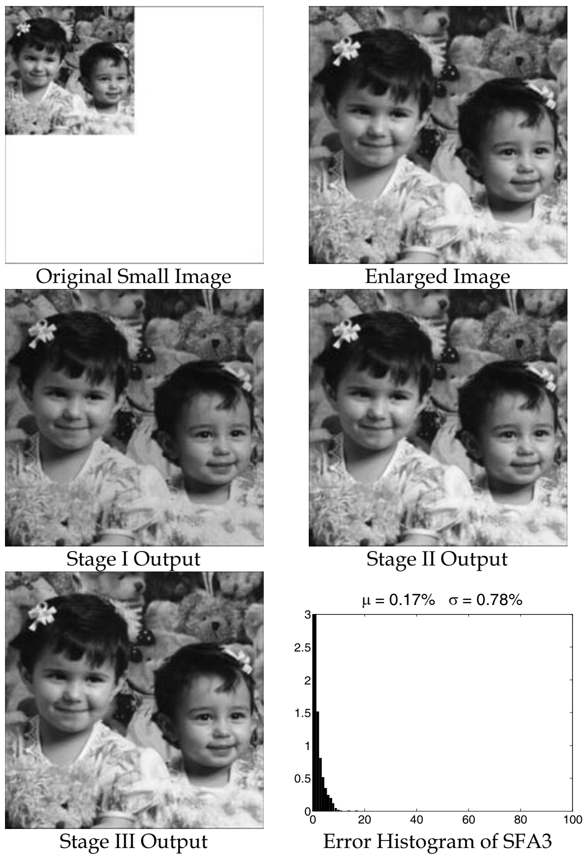


Fig. 6: An example of enlarging images (zooming)