High Accuracy Approximate Multiplier With Error Correction

Chia-Hao Lin and Ing-Chao Lin
Dept. of Computer Science & Information Engineering
National Cheng Kung University
Tainan, Taiwan
{p76004326, iclin}@mail.ncku.edu.tw

***Abstract*—Approximate computing has gained significant attention due to the popularity of multimedia applications. In this paper, we propose a novel inaccurate 4:2 counter that can effectively reduce the partial product stages of the Wallace Multiplier. Compared to the normal Wallace multiplier, our proposed multiplier can reduce 10.74% of power consumption and 9.8% of delay on average, with an error rate from 0.2% to 13.76%. The accuracy of amplitude is higher than 99%. In addition, we further enhance the design with error-correction units to provide accurate results. The experimental results show that the extra power consumption of correct units is lower than 6% on average. Compared to the normal Wallace multiplier, the average latency of our proposed multiplier with EDC is 6% faster when the bit-width is 32, and the power consumption is still 10% lower than that of the Wallace multiplier.**

Keyword—multiplier, approximate, power reduction, high accuracy

I. INTRODUCTION

Embedded systems and mobile systems have become ubiquitous, and energy and power consumption has have become a critical design constraints. Many methods that can improve energy and power efficiency have been proposed at various levels. However there are always a tradeoff between speed and power.

The performance of a traditional arithmetic circuit has limits. In order to further enhance performance, approximate arithmetic designs scarify accuracy to reduce energy consumption and reduce costs related to manufacturing, verification and testing [8]. These approximate arithmetic designs can be accepted in error-tolerance applications or applied in applications related to human senses such as sight and hearing.

Many methods have been proposed to achieve approximate circuits. There are some research directions for designing inaccurate circuits. V. Gupta et al. [1] made a reduction at the transistor level [1], D. Shin et al. [10] reduced the circuit area by simplifying the logic synthesis, and more studies have reduced circuit delay by adjusting circuit architecture [2].

Many approximate adders have been proposed. Sangjin et al. [9] used inaccurate 4:2 counters to build adders with fewer stages of logic. Lu et al. [5] proposed a fast adder that has a shorter carry chain, considering only previous k input bits, and generating the carry bit. Verma et al. [13] further enhanced Lu's adder with error detection and error correction. Zhu et al. [15][16][17] proposed four approximate adders, including ETAI, ETAII, ETAIIM and ETAIII. ETAI is an adder divided into accurate and inaccurate parts. The accurate part is a normal adder, and the inaccurate part uses simple approximate arithmetic. ETAII cuts off the carry chain of adders. ETAIIM modified ETAII by connecting the MSB parts to improve its accuracy. ETAIII is an adder with both accurate and inaccurate functions in less significant bits, and it uses a selector to select the function. Kahng et al. [1] divided the adders into several sub adders that are overlapped, and the accuracy of the results is configurable during runtime and improves the achievable tradeoff between performance/power and quality. Apart from the approximate adders, some approximate multipliers have also been proposed. Kyaw et al. [3] divided the multiplier into an accurate part and an inaccurate part. Kulkarni et al. [4] proposed a 2x2 inaccurate multiplier that can be used to build arbitrarily large power-efficient inaccurate multipliers. In addition, the design was enhanced to allow correct operation of the multiplier using a residual adder. The Kyaw multiplier can reduce power consumption by more than 50% compared to an array multiplier, but it has a high error rate and low accuracy. The Kulkarni multiplier can save power in a range between 31.78% and 45.4% compared to the array multiplier, but there is still a relatively high error rate, and its overhead for error detection and correction is large due to the small basic block.

The majority of approximate arithmetic designs have focused on adders, but multipliers are one of the primary sources of power consumption in digital signal processing applications such as Finite-Impulse-Response (FIR) filters [12]. Compared to approximate adders, more studies are needed to focus on designing approximate multipliers. In this paper, we propose a low power, high accuracy multiplier. Our major contributions are as follows:

- We propose a novel inaccurate 4:2 counter that can be used to build a basic power efficient inaccurate 4×4 Wallace multiplier.
- We can build arbitrarily large power efficient inaccurate multipliers by using inaccurate 4×4 Wallace multipliers.
- We further enhance our inaccurate Wallace multiplier (IWM) with error detection and correction (EDC) to provide accurate results. EDC circuits only increase power consumption and area slightly.
- Extensive experimental results have been done to compare the IWM to previous approximate multipliers.

This work was partially supported by the National Science Council of Taiwan under Grant No.100-2221-E-006-177

The rest of the paper is organized as follows: Section II introduces the basic Wallace multiplier and two inaccurate multipliers, inaccurate 4:2 counters and the metric for Approximate Design. Section III introduces the inaccurate 4:2 counter proposed in this paper and shows how it can be used to build multipliers. The experimental results are presented in Section IV, and Section V concludes the paper.

II. PRELIMINARIES

A. Wallace Multiplier

The Wallace multiplier [14] exercises the Wallace tree, which is an efficient and parallel multiplication algorithm by which to generate a result. The primary advantage of the Wallace tree is making an adding stage reduction by using half-adders and full-adders. The Wallace tree reduces the delay order of array multipliers from O(n) to O(log n). Fig. 1 shows a 4×4 Wallace multiplier dot-notation.

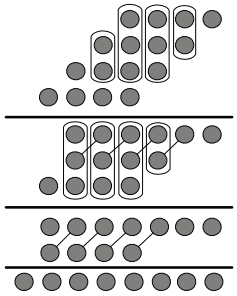

Fig. 1. A 4×4 Wallace multiplier dot-notation.

B. Inaccurate Multiplier

This section introduces two inaccurate multipliers. Kyaw et al. [3] separated the multiplier into an accurate part and an inaccurate part. The accurate part is a normal multiplier that can generate the multiplication of higher order bits. The inaccurate part is made up of simplified circuits proposed by [3] that can generate approximate results for lower order bits. The Kyaw multiplier can reduce the power consumption by more than 50% compared to an array multiplier, but it has a high error rate, and it is difficult to do error correction. Kulkarni et al. [4] proposed a 2×2 inaccurate multiplier as a building block. This 2×2 inaccurate multiplier represents a 3*3 output using three bits 111_2 instead of the usual four bits 1001_2. Error occurs with a magnitude of (9-7)=2, with a probability of 1/16. The delay and power of a multiplier built out of 2×2 inaccurate multipliers can be significantly reduced. The Kulkarni multiplier also enhances the design to allow for correct operation by using an adder.

C. Inaccurate 4:2 Counters

The work in [7] used inaccurate 4:2 counters to implement tree multipliers. An inaccurate 4:2counter is a counter with 4 inputs and 2 outputs which is used to effectively reduce adding stages. The difference between an inaccurate 4:2 counter and an ordinary counter is that an ordinary counter gives the sum 100_2 when all four inputs are 1, but an inaccurate 4:2counter simplifies the architecture by using two bits to present the 3-bit result. Fig. 2 and Fig. 3 show the architecture of the inaccurate 4:2 counter proposed by [7] and [9]. The work in [9] uses 11_2 to represent the sum 100_2, and the work in [7] further reduces the delay of the 4:2 counter by using 10_2 to represent the sum 100_2.

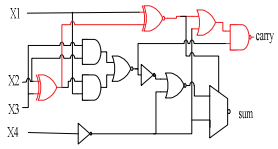

Fig. 2. An inaccurate 4:2 counter proposed by [9]

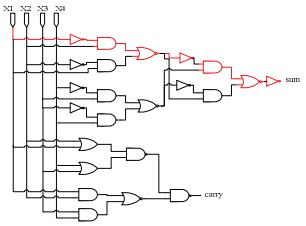

Fig. 3. An inaccurate 4:2 counter proposed by [7]

The delay of these two inaccurate 4:2 counters can be improved because their critical path is longer than two xor gates.

D. Metric for Approximate Design

In order to quantify errors in approximate designs, this section introduces some basic metrics:

Definition: *amplitude data*. Amplitude data is the data with bits having different weights. The bits that are closer to the most significant bit have higher weight than the bits that are close to the least significant bit.

Definition: information data. Information data is data with bits of identical weight.

- Pass rate is the probability of having a correct result, and can be represented as follows:

$$\text{Pass rate}=\frac{\#correct\ results}{\#total\ results}. \quad (1)$$

- Accuracy for amplitude data (ACC_{amp}) is used to quantify the amplitude of errors, and can be represented as follows:

$$\text{ACC}_{amp} = 1 - \frac{|R_c - R_e|}{R_c}, \quad (2)$$

where R_c denotes the correct result, and R_e is the result obtained by the proposed multiplier. For example, when the correct data is 1001_2, and the result data is 111_2, the ACC_{amp} will be 7/9.

- Accuracy for information data (ACC_{inf}) measures error significance as the Hamming distance, and can be represented as follows:

$$ACC_{inf} = \frac{1-B_e}{B_w}, \quad (3)$$

where B_e is the number of error bits, and B_w is the bit-width of the data. For example, when the correct date is 11100001_2, and the result data is 11010001_2, the ACC_{inf} will be 6/8.

The work in [11] used the product of pass rate and ACC_{amp} as a metric of error tolerance, and ACC_{inf} is a more meaningful metric for communication systems that mainly handle information data. Therefore, we will use the pass rate, ACC_{amp}, and ACC_{inf} obtained from experimental results to evaluate approximate multipliers.

III. METHODOLOGY

A. Proposed Inaccurate 4-bit Wallace multiplier

Our main idea is to use a 2:1 MUX to replace a 'xor' gate. Then the new 4:2 counter will have shorter delay. The layers of the Wallace multiplier are reduced by an inaccurate 4:2 counter, so the delay and power of the Wallace multiplier can be reduced.

We propose a novel architecture in order to reduce the delay of an inaccurate 4:2 counter, as shown in Fig. 4. X1 to X4 are the inputs. Sum and carry are the outputs. Note that the error occurs when all four summands are '1', and the output 111_2 is reduced to 10_2. This error is intentionally generated to achieve a simplified algorithm. This inaccurate 4:2 counter can be used to build an inaccurate 4×4 Wallace Multiplier, as shown in Fig. 5. An ordinary Wallace multiplier reduces the adding stages from three stages to two stages, but it can reduce the adding stages from four stages to two stages by using an inaccurate 4:2 counter to further reduce the delay and the power. Note that we use an inaccurate 4:2 counter to give the sum of a partial product, and the probability of a partial product becoming '1' is 1/4, so the error of the inaccurate 4:2 counter occurs with a probability of $(1/4)^4 = 1/256$, which is very low.

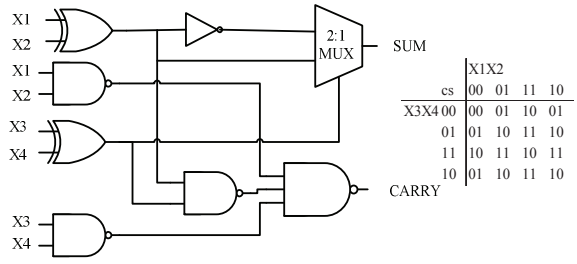

Fig. 4. The archietecture of proposed 4:2 counter

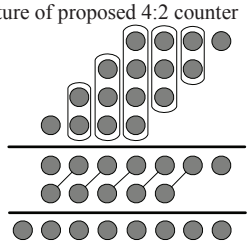

Fig. 5. **A** 4x4 Wallace multiplier dot-notation with 4:2 counter

B. Building Larger Multipliers

Larger multipliers are easily built by using inaccurate 4×4 Wallace multipliers [5]. Fig. 6 shows an example of building a 32×32 multiplier consisting of inaccurate 4×4 Wallace multipliers. The 32×32 multiplication is decomposed to three additions of 16×16 multiplication results. Each 16×16 multiplication is decomposed to three additions of 8×8 multiplication results. Finally, each 8×8 multiplication is decomposed to three additions of 4×4 multiplication results. Note that adding three partial products in the blocks can be reduced to adding two partial products by using full-adders (3:2 counters). Then these two partial products are added by a normal adder. We can build even larger multipliers using this flow chart. In addition, the accuracy of the multiplier is adjustable. For example, we can use an accurate multiplier for AH×BH, and then the ACC_{amp} of the multiplier can be improved. If larger multipliers are only built by inaccurate 4×4 Wallace multipliers, then the probability of having a correct result can be represented as follows:

$$\text{Pass rate} = \left(\frac{255}{256}\right)^{4^{(\log_2 w)-2}}, \tag{4}$$

Where w is the input bit-width of the multiplier.

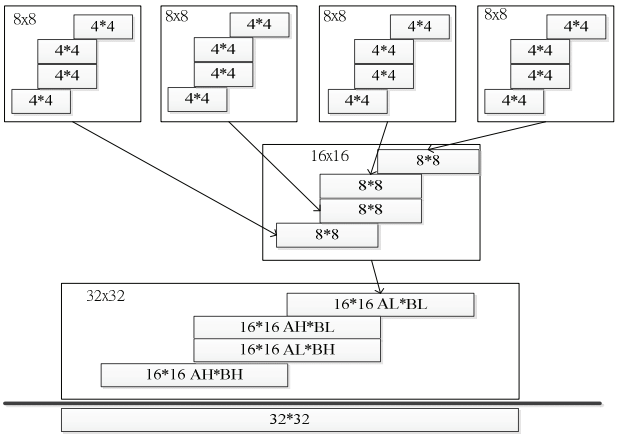

Fig. 6. Building larger multipliers from smaller blocks

In order to further reduce the delay of the multiplier, we separate the adder of the final stage into two sub sum generators, and the complete architecture is shown in Fig. 7. The first sum generator is a normal adder, and the second sum generator uses a carry predictor to reduce the error rate. The carry chain is reduced by this architecture. The signal arrival time in the oval is later than that on the left side; therefore, the carry predictor only considers the signal value on the left side of gray circle to reduce the multiplier delay. In the carry predictor, errors occur when S2~S5+C1~C4 produces a carry bit, and the sum of S6~S8+C5~C7 has all '1' values. The probability of having an error result can be represented as follows:

$$\text{Error rate} = \frac{1}{2^{cl}} \times \frac{2^k - 1}{2^{k+1}}, \tag{5}$$

Where cl denotes the bit-width of the carry predictor, and k is the bit-width of the first sum generator minus cl. Note that we only use this architecture in the final summation in order to prevent the pass rate from dropping too fast. For example, when we build a 16bit multiplier, the internal 8bit multiplier does not use this approximate adder.

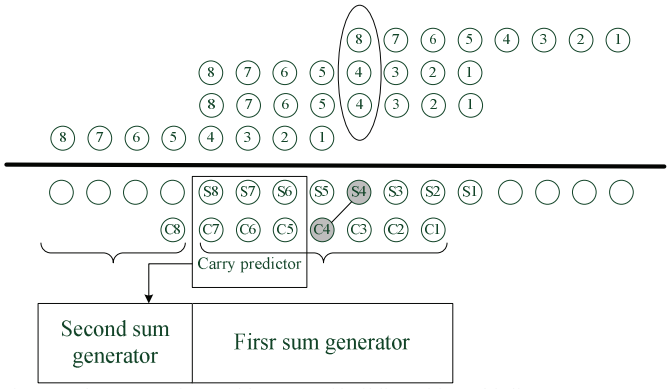

Fig. 7. The summation architecture of building 8bit multiplier

C. Accurate Operation Mode

It is simple to enhance error detection and error correction in our proposed multiplier. For a 4×4 inaccurate Wallace multiplier, error occurs when the multiplier bits and multiplicand bits are all 1. The partial products in the fourth column all become '1'. A 4×4 accurate multiplier gives the product 11100001_2, but a 4×4 inaccurate Wallace multiplier gives the product 11010001_2, in this situation. Note that the differences are the values of the fifth bit and the sixth bit and that this error is corrected if the fifth bit is forced to 0, and the sixth bit is forced to 1. Error detection can be easily implemented with an 'and' gate, as shown in Fig. 8. Error correction can be implemented with an 'or' gate and a 'nor' gate.

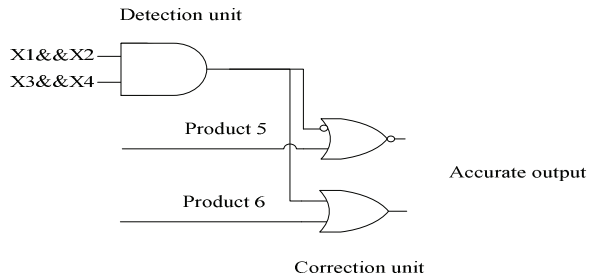

Fig. 8. An error detection and correction unit

Our proposed multiplier can generate accurate results with Error detection and correction (EDC) circuits. Fig. 9 shows the architecture of a 4×4 IWM with EDC. When operating in the accurate mode, the EDC circuits produce correct product 5 and product 6. When operating in the approximate mode, the EDC circuits will be either completely switched off or power gated to reduce the power consumption. Larger multipliers are built by using 4×4 IWM with EDC as mentioned in section III.B, and it is very easy to use some AND gates to detect the error of the carry predictor. When this error occurs, a stall signal is generated, and the inaccurate results should add '1' using a simple adder. With these simple error detection and correction circuits, our proposed multiplier can be implemented to have variable latency as in [13]. Users can adjust these 4×4 IWMs with EDC blocks, with part of them operating in the accurate mode, and part of them operating in the approximate mode. Our proposed multiplier can generate a result to applications with different accuracy requirements. When an application only needs low accuracy results, our proposed multiplier reduces the power consumption by turning to an approximate mode.

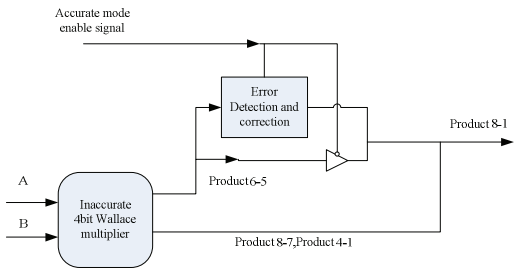

Fig. 9. A 4×4 IWM with EDC

IV. Experimental Results

The circuits are implemented in Verilog and are synthesized to gate-level netlists using the Synopsys Design Compiler with a standard TSMC 0.18m CMOS cell-library. Then we use the Synposys Design Compiler to analyze the delay, area, and the power of the circuits under consideration.

TABLE I. A COMPARISON OF 4:2COUNTER

	4:2counter[9]	4:2counter[7]	Proposed 4:2counter
Delay	0.86 ns	0.57 ns	0.53 ns
Area	136.382401	143.035202	129.729602
Delay of 4×4 IWM(ns)	2.24	2.05	1.99
Power of 4×4 IWM(uw)	232.5430	236.4769	230.9072

In order to evaluate our proposed 4:2 counter, we compared our proposed 4:2 counter to the 4:2 counters proposed by [7] and [9], as shown in TABLE I. Row 2 in TABLE I. shows the delay, and row 3 shows the area of these three inaccurate 4:2 counters. Our proposed 4:2 counter has minimum delay and minimum area. Row 4 and row 5 show the delay and the power of the 4×4 IWM built out of these 4:2 counters. The experimental results show that using our proposed 4:2 counter used to build a 4×4 inaccurate Wallace multiplier results in shorter delay and lower power usage.

We use the metrics mentioned in Section II.D to evaluate our proposed multipliers, the Kulkarni inaccurate multiplier[4] and the Kyaw inaccurate multiplier [3]. Moreover we also use the 4:2 counters proposed by [7] and [9] to build a larger multiplier by the same method. Fig. 10 compares the pass rate of these five multipliers. When input bit-width increases to 32, the pass rate of the Kulkarni inaccurate multiplier dramatically decreases due to its basic 2×2 block. The error rate of 2×2 block is 1/16. A 32×32 Kulkarni inaccurate multiplier is built out of 256 2×2 blocks, and therefore its pass rate is only 2% on average. The Kyaw inaccurate multiplier uses accurate multipliers to compute AH*BH for MSBs and computes LSBs using approximate multipliers. Therefore, the pass rate of the Kyaw multiplier is very low. Our proposed multiplier is built out of 4×4 blocks. The error rate of a 4×4 block is 1/256. The sum generator in the final summation may cause some errors, but the total pass rate is still high. When the input bit-width is increased to 32, the pass rate of the proposed multiplier is higher than 80% on average. The pass rate of the multipliers built out of 42counters [7] and [9] are equal to our proposed multiplier because the error occurs at the same inputs.

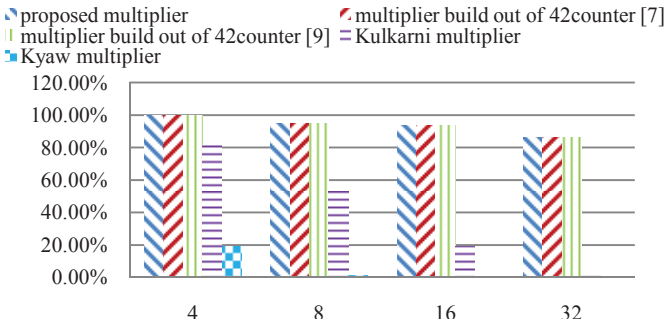

Fig. 10. Pass rate comparison

Fig. 11 shows that the ACC_{amp} of the proposed multiplier is higher than 99% on average, from 4 bits to 32 bits. The ACC_{amp} of the Kulkarni inaccurate multiplier is from 96% to 97%. The Kulkarni inaccurate multiplier has a lower ACC_{amp} since the error magnitude and the error rate of its 2×2 block are both higher than our proposed 4×4 block. The ACC_{amp} of the Kyaw inaccurate multiplier achieves 99.9% when the input bit-width is 32, but the ACC_{amp} cannot maintain the same quality when the input bit-width is decreased. The multipliers built out of the 42counter [7] have the same ACC_{amp} compared to our proposed multiplier since the error rate and the error magnitude of the basic 4×4 block are same as our 4×4 block. The ACC_{amp} of the multipliers built out of the 42counter [9] is slightly higher than our proposed multiplier since the error magnitude of the basic 4×4 block is lower than that of our proposed multiplier.

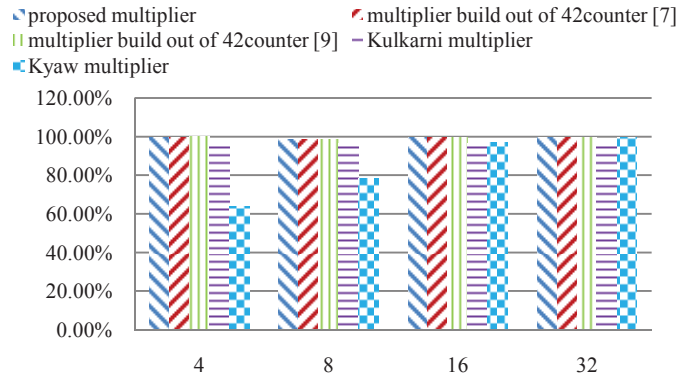

Fig. 11. ACC_{amp} comparison

Fig. 12 shows the ACC_{inf} of these five multipliers. The ACC_{inf} of our proposed multiplier dropped slightly from 99.9% to 99.3% when the input but-width increased from 4 to 32. The ACC_{inf} of the Kulkarni inaccurate multiplier dropped from 93% to 72% because of the error rate of its 2×2 block previously mentioned. The ACC_{inf} of the Kyaw inaccurate multiplier is lower than 70% on average due to the inaccurate part of the LSBs. We determined that our proposed multiplier has better performance from these three metrics. The multipliers built out of the 42counter [7] have the same ACC_{inf} compared to our proposed multiplier. The ACC_{inf} of the multipliers build out of the 42counter [9] is slightly lower than our proposed multiplier since the basic 4×4 block has more error bits when error occurs.

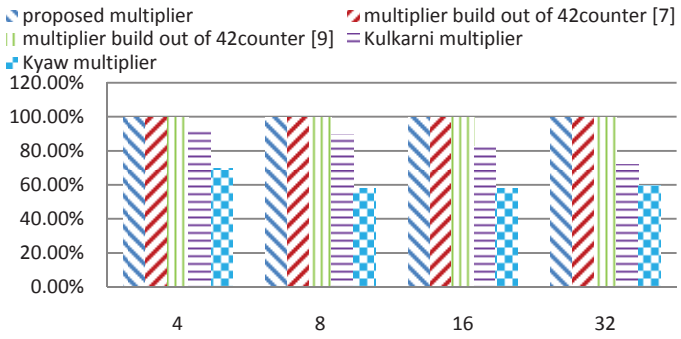

Fig. 12. ACC_{inf} comparison

In order to evaluate the delay improvement and the power improvement of our proposed multiplier, we used the Wallace multiplier as the criteria, and we compared our proposed multiplier to the Kulkarni inaccurate multiplier [4]. Fig. 13 compares the delay of multipliers in different bit-widths. The 32bit proposed multiplier achieved nearly 7% delay reduction compared to the Wallace multiplier, and the 32bit Kulkarni inaccurate multiplier increased the delay by 13.6%.

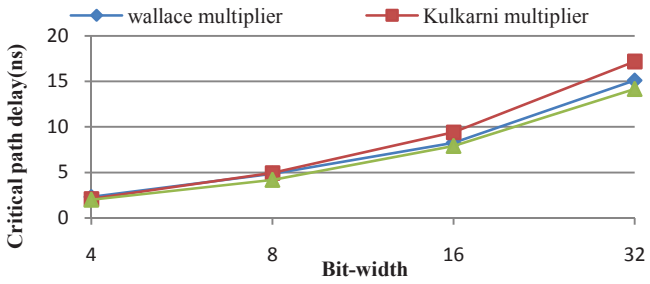

Fig. 13. The delay comparison

Fig. 14 compares the power of multipliers in different bit-widths. The 32bit proposed multiplier achieves nearly 13% power reduction compared to the Wallace multiplier, and the 32bit Kulkarni inaccurate multiplier achieves 18% power reduction.

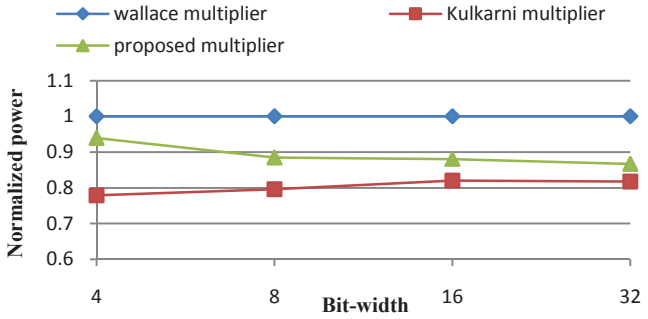

Fig. 14. The power comparison

Fig. 15 compares the EDP of multipliers in different bit-widths. Overall, our proposed multiplier has better performance when delay and power are both considered.

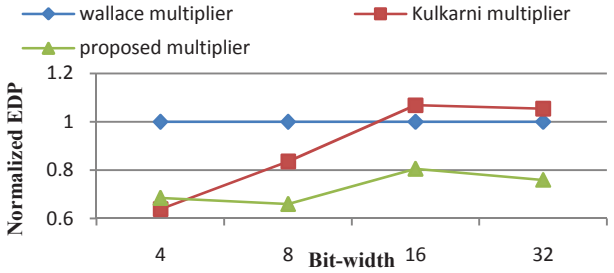

Fig. 15. Energy delay production comparison

TABLE II. shows the area overhead, power overhead and the increase delay of our proposed multiplier with EDC. The area overhead increased from 3.81% to 7.13%, and the power overhead increased from 4.24% to 6.44%. Fig. 16 compares the average latency(T_{ave}) of the proposed multiplier with EDC to the normal Wallace multiplier from 4bits to 32bits. T_{ave} is given by

$$T_{ave} = T_{cri}(pass\ rate) + 2T_{cri}i(1 - pass\ rate). \quad (6)$$

T_{cri} denotes the critical path delay of circuits. The average latency of our proposed multiplier is close to that of the Wallace multiplier when our proposed multiplier is in accurate mode due to the high pass rate. When the bit-width is increased to 32, the average latency is 6% faster than that of the Wallace multiplier. Fig. 17 compares the power consumption of the proposed multiplier with EDC to the normal Wallace multiplier from 4bits to 32bits. The power consumption is lower than that of the Wallace multiplier since our inaccurate basic 4×4 block saves enough power to obtain this result.

TABLE II. THE OVERHEAD WITH EDC

Bit width	Area overhead	Power overhead
4	3.81%	4.24%
8	7.13%	6.64%
16	5.40%	5.77%
32	3.86%	4.04%

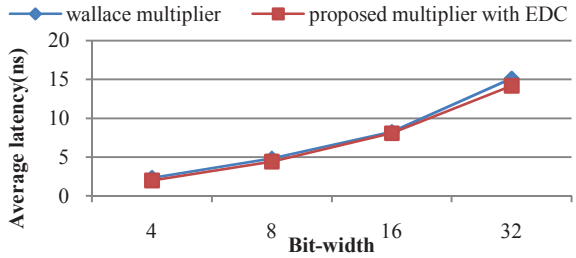

Fig. 16. Average latency comparison of multiplier with EDC

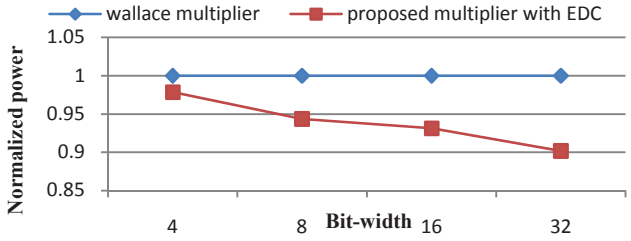

Fig. 17. Power comparison of multiplier with EDC

V. CONCLUSIONS

In this paper, we propose a novel inaccurate 4:2 counter with lower delay and power that can be used to build a basic power efficient inaccurate 4×4 Wallace multiplier. We build an arbitrarily large multiplier by using this inaccurate 4×4 Wallace multiplier. Compared to the normal Wallace multiplier, our proposed multiplier can reduce power consumption by 10.74% on average and can reduce delay by 9.8% on average. Compared to other approximate multipliers, our proposed multiplier has higher pass rate and greater accuracy. Moreover, we propose an efficient error detection and correction circuit with only 6% power overhead on average. The power consumption and the average latency are better than that of the Wallace multiplier when our proposed multiplier is in accurate mode.

ACKNOWLEDGMENTS

The authors would like to thank master Yi-Ming Yang and Yu-Hung Cho of National Cheng Kung University for their help in setting up the experimental framework.

REFERENCES

[1] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan and K. Roy, "IMPACT: IMPrecise adders for low-power Approximate CompuTing" in Low Power Electronics and Design (ISLPED) International Symposium on pp. 409-414, 2011.

[2] A.B. Kahng and S. Kang, "Accuracy-Configurable Adder for Approximate Arithmetic Designs" in Design Automation Conference (DAC), pp. 820-825, 2012.

[3] K.Y. Kyaw, W.L. Goh, and K.S. Yeo, "Low-Power High-Speed Multiplier For ErrorTolerant Application" in Electron Devices and Solid-State Circuits (EDSSC), pp. 1-4, 2010.

[4] P. Kulkarni, P. Gupta and M. Ercegovac, "Trading Accuracy for Power with an Underdesigned Multiplier Architecture" in VLSI Design, pp. 346-351, 2011.

[5] I. Koren, Computer Arithmetic Algorithms, 2nd ed. pp. 149-150, A.K. Peters, 2002.

[6] S.-L. Lu, "Speeding Up Processing with Approximation Circuits", in IEEE Computer 37(3) pp. 67-73, 2004.

[7] J. Ma, K. Man, T. Krilavicius, S. Guan, and T. Jeong, "Implementation of High Performance Multipliers Based on Approximate Compressor Design", in International Conference on Electrical and Control Technologies (ECT), 2011.

[8] International Technology Roadmap for Semiconductors, 2009, http://www.itrs.net.

[9] B.J. Phillips, D.R. Kelly and B.W. Ng, "Estimating adders for a low density parity check decoder", F. T. Luk, Ed., vol. 6313, no. 1. SPIE, 2006, p.631302.[Online].Available:http://link.aip.org/link/?PSI/6313/631302/1

[10] D. Shin, and S. K. Gupta, "Approximate logic synthesis for error tolerant applications" in Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 957-960, 2010.

[11] D. Shin, and S. K. Gupta, "A new circuit simplification method for error tolerant applications" in Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1-6, 2011.

[12] S. K. Sangjin, S. Hong, M. C. Papaefthymiou, and W. E. Stark, "Low power parallel multiplier design for dsp applications through coefficient optimization," in International ASIC/SOC Conference pp. 286-290, 1999.

[13] A.K. Verma, P. Brisk and P. Ienne, "Variable Latency Speculative Addition: A New Paradigm for Arithmetic Circuit Design", in Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1250–1255, 2008.

[14] C.S. Wallace, "A Suggestion for a Fast Multiplier", in IEEE Transaction on Electronic Computers, pp. 14-17, 1964.

[15] N. Zhu, W.L. Goh and K.S. Yeo, "An Enhanced Low-Power High-Speed Adder For Error-Tolerant Application" in Intl. Symposium on Integrated Circuits, pp. 69–72, 2009.

[16] N. Zhu, W.L. Goh, W. Zhang, K.S. Yeo and Z.H. Kong, "Design of Low-Power High-Speed Truncation-Error-Tolerant Adder and Its Application in Digital Signal Processing" in Very Large Scale Integration(VLSI) Systems, pp. 1225–1229, 2010.

[17] N. Zhu, W.L. Goh and K.S. Yeo, "Ultra low-power high-speed flexible Probabilistic Adder for Error-Tolerant Applications" in International SoC Design Conference (ISOCC), pp. 393-396, 2011.