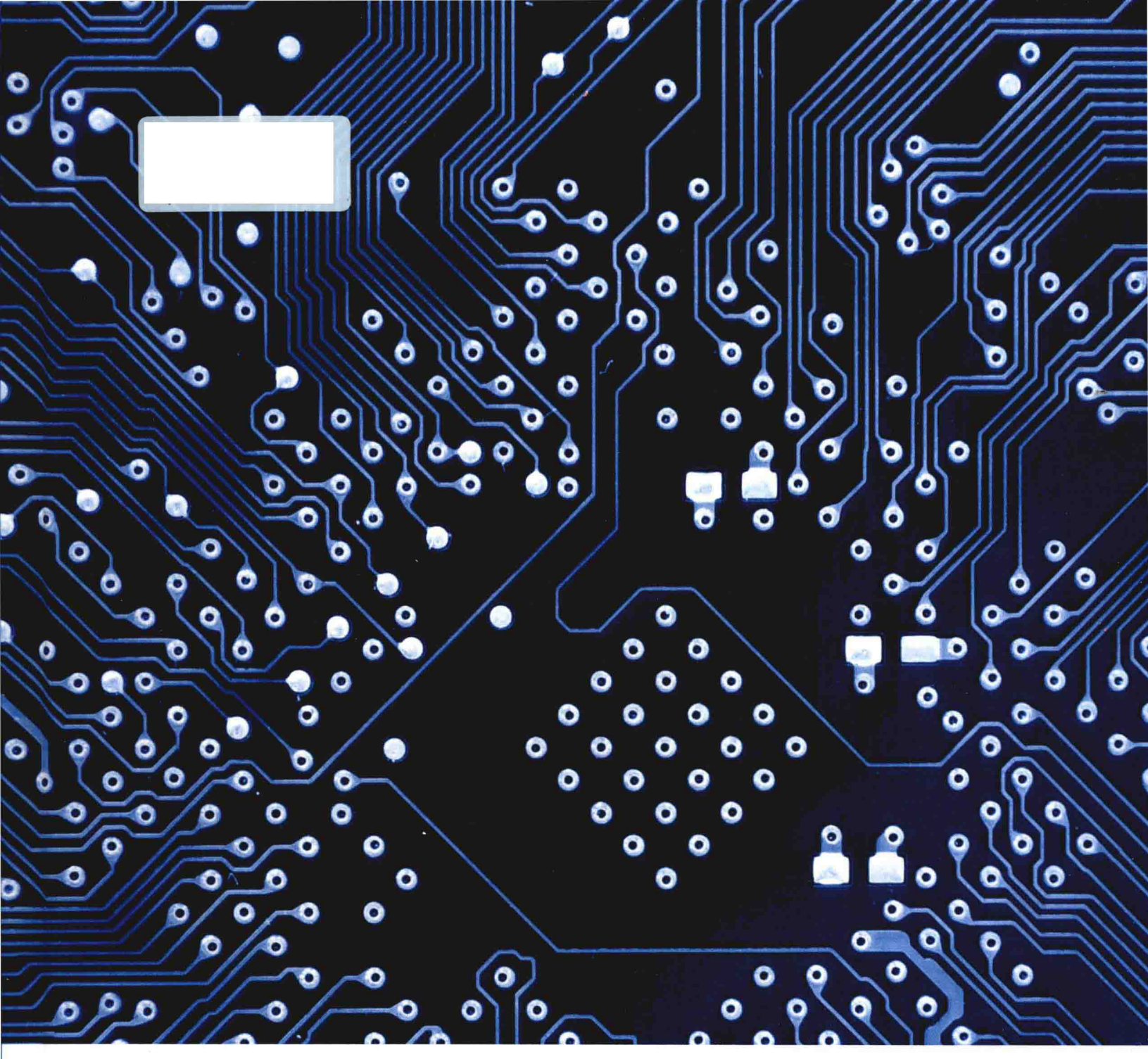

VLSI Design Handbook

Martin Limestone

Volume II

VLSI Design Handbook
Volume II

Edited by **Martin Limestone**

LANRYE
INTERNATIONAL
New Jersey

Published by Clanrye International,
55 Van Reypen Street,
Jersey City, NJ 07306, USA
www.clanryeinternational.com

VLSI Design Handbook: Volume II
Edited by Martin Limestone

International Standard Book Number: 978-1-63240-520-3 (Hardback)

Printed in China.

VLSI Design Handbook

Volume II

Preface

The abbreviation VLSI stands for Very Large Scale Integration. Integrated circuit technology allows billions of transistors to be fabricated into a single chip. The development of this technology only occurred in the twentieth century, somewhere in the mid-1920s, when numerous people tried to create devices which intended to convert solid-state diodes into triodes by controlling current. However, it was only in 1947, with the creation of transistors at Bell Labs that vacuum tubes were replaced by solid-state devices.

Factually the Moore's Law was always validated for prediction of exponential complexity growth and advancement in the performance of integrated circuits. Most semiconductor based industries, face extreme problems in maintaining all aspects of production process during designing of the chip. These issues range from scientific research in discovering novel materials and devices to advanced technology developments and finding new killer applications. This book has been compiled in order to emphasize the latest developments in the vast field of VLSI design.

The contributors have made no attempt to be comprehensive on the topics. Instead, they tried to provide some promising concepts, such as problems and challenges for the introduction of new-generation electronic design automation tools, optimization, modeling and simulation methodologies, thermal and power reduction and management, parasitic interconnects, etc.

I would like to thank all the authors for their excellent contributions in different applications of VLSI. Despite the rapid advances in the field, I believe that the examples provided here will allow us to look through some main researches. I hope that this book will prove to be a worthy contribution in the field of VLSI. I also wish to thank the publisher and the publishing team for their outstanding support at every level of the editing process. Lastly, I wish to convey my regards to my friends and family for supporting me in every endeavor of my life.

Editor

Contents

Faster and Energy-Efficient Signed Multipliers

B. Ramkumar and Harish M. Kittur

VLSI Division, School of Electronics Engineering, VIT University, Vellore 632014, Tamilnadu, India

Correspondence should be addressed to B. Ramkumar; bramvlsi@gmail.com

Academic Editor: Juan Sanchez-Garcia

We demonstrate faster and energy-efficient column compression multiplication with very small area overheads by using a combination of two techniques: partition of the partial products into two parts for independent parallel column compression and acceleration of the final addition using new hybrid adder structures proposed here. Based on the proposed techniques, 8-b, 16-b, 32-b, and 64-b Wallace (W), Dadda (D), and HPM (H) reduction tree based Baugh-Wooley multipliers are developed and compared with the regular W, D, H based Baugh-Wooley multipliers. The performances of the proposed multipliers are analyzed by evaluating the delay, area, and power, with 65 nm process technologies on interconnect and layout using industry standard design and layout tools. The result analysis shows that the 64-bit proposed multipliers are as much as 29%, 27%, and 21% faster than the regular W, D, H based Baugh-Wooley multipliers, respectively, with a maximum of only 2.4% power overhead. Also, the power-delay products (energy consumption) of the proposed 16-b, 32-b, and 64-b multipliers are significantly lower than those of the regular Baugh-Wooley multiplier. Applicability of the proposed techniques to the Booth-Encoded multipliers is also discussed.

1. Introduction

High-speed multiplication is a primary requirement of high-performance digital systems. In recent trends, the column compression multipliers are popular for high-speed computations due to their higher speeds [1, 2]. The first column compression multiplier was introduced by Wallace in 1964 [3]. He reduced the partial product of N rows by grouping into sets of three-row set and two-row set using (3,2) counter and (2,2) counter, respectively. In 1965, Dadda altered the approach of Wallace by starting with the exact placement of the (3,2) counter and (2,2) counter in the maximum critical path delay of the multiplier [4]. Three-dimensional minimization- (TDM-) based column compression approach was proposed in 1996 to perform fast multiplication [5]. Since the 2000s, a closer reconsideration of Wallace and Dadda multipliers has been done and proved that the Dadda multiplier is slightly faster than the Wallace multiplier and the hardware required for Dadda multiplier is lesser than the Wallace multiplier [6, 7]. The HPM-based column compression was developed in 2006, and it has standard layout structure than Eriksson et al.'s multiplier [8]. The detailed case for HPM-based Baugh-Wooley multiplier against the Booth-Encoded multipliers has been described in [9]. In this work, we implement the proposed techniques with the W, D, H based Baugh-Wooley multipliers, and the improved performance is compared with that of the same regular multipliers.

The Baugh-Wooley (BW) algorithm is a relatively straightforward way of doing signed multiplications [10]; Figure 1 illustrates the algorithm for an 8-bit case, where the partial-product bits have been reorganized as specified by Själander and Larsson-Edefors in his work [11]. The creation of the reorganized partial-product array comprises three steps: (i) the most significant bit (MSB) of the $N - 1$ partial-product rows and all bits of the last partial-product row, except its MSB, are inverted; (ii) a "1" is added to the Nth column; (iii) the MSB of the final result is inverted. The total delay of the multiplier can be split up into three parts: due to the partial-product generator (PPG), partial-product summation tree (PPST), and final CPA [12]. Of these, the dominant components of the multiplier delay are due to the PPST and the final adder. The relative delay due to the PPG is small. Therefore, a significant improvement in the speed of the multiplier can be achieved by reducing the delay in

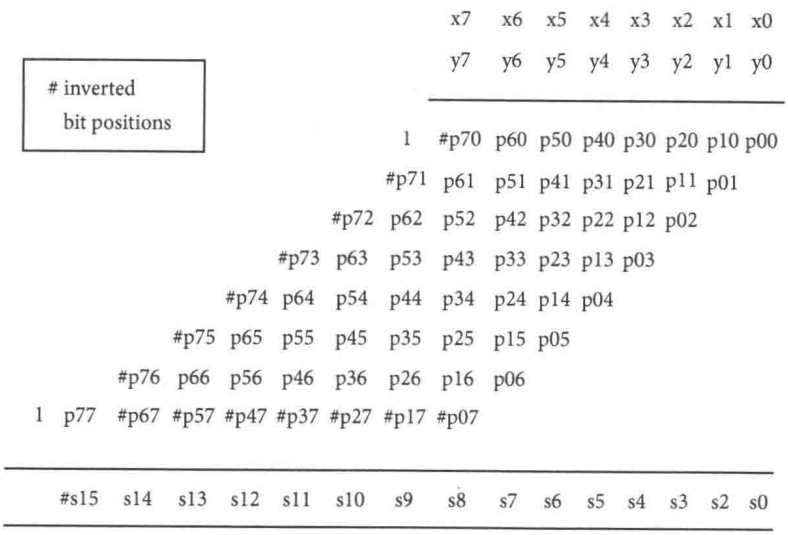

FIGURE 1: Illustration of an 8-bit Baugh-Wooley multiplication.

the PPST and the final adder stage of the multiplier. In this work, the delay of PPST is reduced by using two independent structures in the partial products. The proposed hybrid CPA, based on arrival profile aware design [12, 13] and the BEC (Binary to Excess-1 Converter) Logic [14, 15], computes the final products much faster. Arrival profile aware hybrid adders have been reported earlier [12, 13]. Recently, further investigations on the same are reported in [16].

This paper is structured as follows. Sections 2 and 3 describe the design of parallel structures for the PPST and the design of hybrid final adder structure, respectively. Section 4 reports the ASIC implementation details and the simulation results. Finally, Section 5 summarizes the result analysis. Throughout the paper, it is assumed that the number of bits in the multiplier and multiplicand is equal.

2. Design of Parallel Structures

The multiplication process begins with the generation of all partial products in parallel using an array of AND gates. The next major steps in the design process are partitioning of the partial products and their reduction process. Each of these steps is elaborated in the following subsections.

2.1. Partitioning the Partial Products. We consider two n-bit (8-bit) operands of Baugh-Wooley multiplier partial products which form a matrix of n rows and $2n$ columns as shown in Figure 1. Initially for the partial product of Baugh-Wooley multiplication, we assign an integer as shown in Figure 2(a); for example, p00 is given an index 0, p10 an index 1, and so on. For convenience, we rearrange the partial products as shown in Figure 2(b). The two longest columns in the middle of the partial products contribute to the maximum delay in the PPST. Therefore, in this work, we split up the PPST into two parts as shown in the Figure 2(c), in which both parts share equal number of columns. That is, part0 consists of n columns and part1 also consists of n columns. We then proceed to sum up each column of the two parts in parallel. The summation procedure adopted in this work is described in the next section.

2.2. The W, D, H Based Reduction. Next, the partial products of each part are reduced to two rows by the using (3,2) and (2,2) counters based on the W, D, H reduction algorithm. The HPM-based reduction is shown in Figures 3 and 4. The grouping of 3-bits and 2-bits indicates (3,2) and (2,2) counters, respectively, and the different colors classify the difference between each column. The bit positions s0, 22, and 29 are added using (3,2) counter to generate sum s2 and carry c2. The final two rows of each part are summed using a carry lookahead adder (CLA) to perform fast addition, and it forms the partial final products of a height of one-bit column, which is indicated at the bottom of Figures 3 and 4.

The two parallel structures in Figures 3 and 4 based on the HPM method are shown in Figure 5, where HA, FA, p0, p1, and p denote half adder ((2,2) counter)), full adder ((3,2) counter), partial final product from part0, partial final product from part1, and final product, respectively. The numerals residing on the HA and FA indicate the position of partial products. The outputs of part0 and part1 are computed independently in parallel, and those values are added using a high-speed hybrid final adder to get the final product.

However, before we proceed to carry out the final addition with the proposed hybrid adder, we first carry out the final addition with the faster adder of CLA for both the unpartitioned W, D, H Baugh-Wooley multiplier and the partitioned W, D, H Baugh-Wooley multiplier. This enables us to evaluate and analyze the effect of partitioning the PPST into two parts. The simulation results and their comparison are listed in Tables 1, 2, and 3, in these tables a negative percentage indicates overhead and a positive percentage indicates a reduction/improvement with reference to the compared multiplier. The comparison shows the percentage improvement

Table 1: Partitioned Wallace performance.

Regular Wallace multiplier with CLA				Partitioned Wallace multiplier with CLA			Performance comparison (%)		
Word size	Delay (ns)	Area (μm^2)	Power (μW)	Delay (ns)	Area (μm^2)	Power (μW)	Delay	Area	Power
16	1.84	3,983	0.66	1.76	4,093	0.69	4.34	−2.76	−4.54
32	2.52	14,398	3.21	2.30	14,665	3.33	8.73	−1.85	−3.73
64	3.29	53,896	16.86	2.92	54,449	17.34	11.24	−1.02	−2.84

Table 2: Partitioned Dadda performance.

Regular Dadda multiplier with CLA				Partitioned Dadda multiplier with CLA			Performance comparison (%)		
Word size	Delay (ns)	Area (μm^2)	Power (μW)	Delay (ns)	Area (μm^2)	Power (μW)	Delay	Area	Power
16	1.69	3,843	0.61	1.66	3,986	0.65	1.77	−3.72	−6.55
32	2.41	13,804	3.14	2.23	14,083	3.26	7.46	−2.02	−3.82
64	3.11	51,885	16.12	2.86	50,517	16.58	8.03	2.63	−2.85

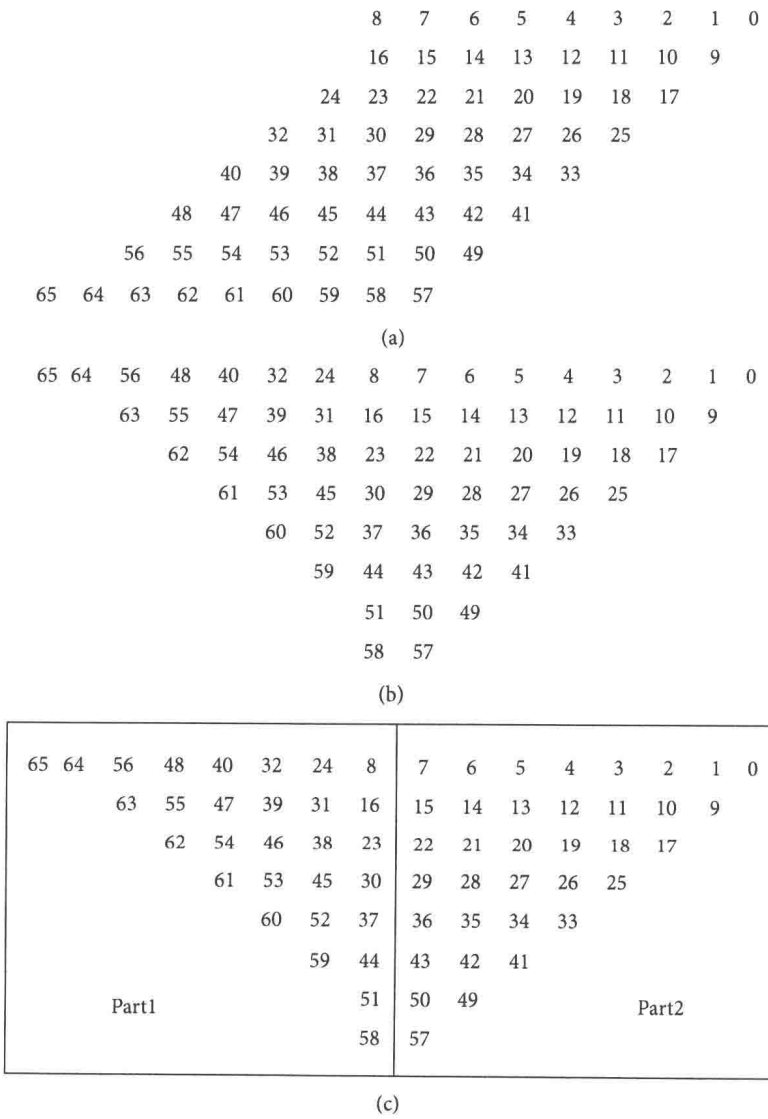

Figure 2: Partitioning the partial products: (a) partial-product array diagram for 8 * 8 multiplier, (b) an alternative representation, and (c) partitioned structure of multiplier showing part0 and part1.

Table 3: Partitioned HPM performance.

Regular HPM multiplier with CLA				Partitioned HPM multiplier with CLA			Performance comparison (%)		
Word size	Delay (ns)	Area (μm^2)	Power (μW)	Delay (ns)	Area (μm^2)	Power (μW)	Delay	Area	Power
16	1.75	3,848	0.63	1.72	4,025	0.67	1.71	−4.59	−6.34
32	2.48	13,817	3.16	2.36	14,173	3.33	4.80	−2.57	−5.37
64	3.18	51,912	16.35	2.99	50,673	16.72	5.97	2.38	−2.26

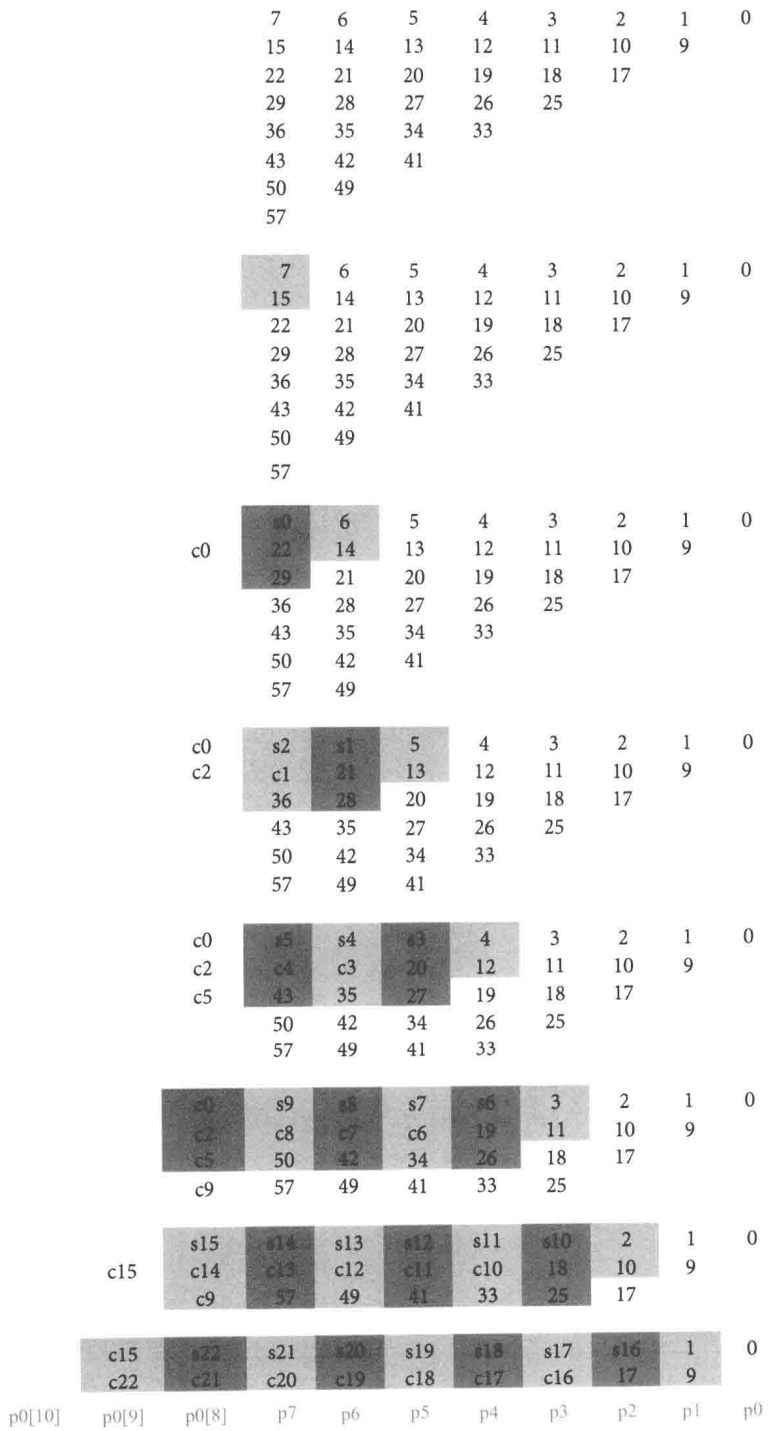

Figure 3: Reduction of the partial products of part0 based on the HPM reduction approach.

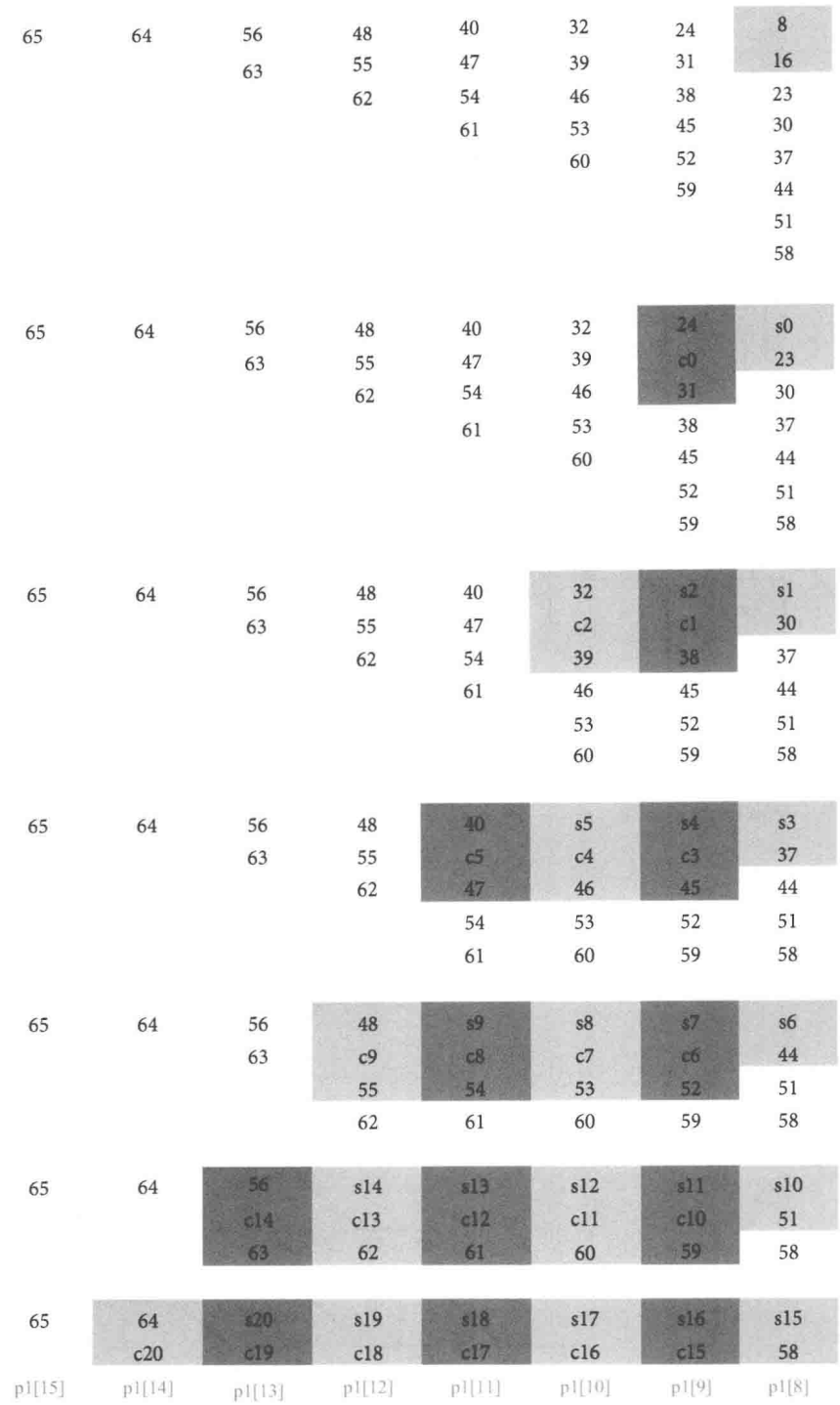

FIGURE 4: Reduction of the partial products of part1 based on the HPM reduction approach.

and overhead in delay, area, and power of the partitioned multipliers with respect to the unpartitioned multiplier.

It can be seen that there is 4.3% improvement in the speed for 16-b and 11.2% for 64-b size. The speed limitation in lower bit size multipliers is due to the greater difference between input arrival profile to the final CPA from part0 and part1. But with the increase in the word size, this difference becomes lesser and the improvement in the speed of the partitioned multipliers increases. There is maximum of 11%, 8%, and 6% speed improvement for 64-b W, D, H Baugh-Wooley multipliers with 1% area overhead. Having clearly demonstrated the reduction in the delay of the multipliers due to the partitioning of the partial products, we now proceed to further enhance the speed of the proposed multiplier. There is maximum of 6% to 7% power overhead in W, D, H based Baugh-Wooley multiplier, and this is due to the use of CLA as CPA in each part. But this power overhead is interestingly reduced by proposed hybrid CPA which is elaborated in the next section.

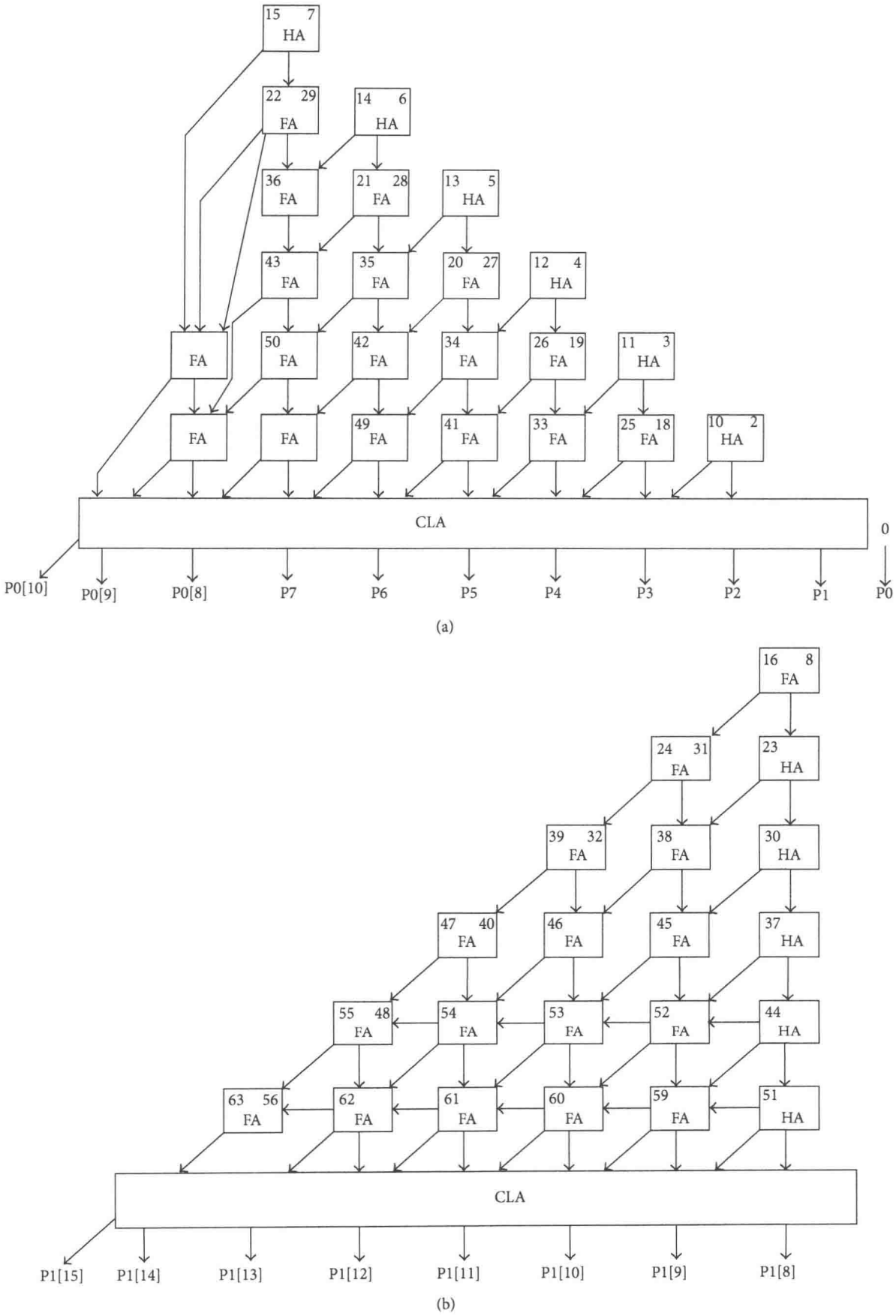

FIGURE 5: The HPM-based implementation: (a) implementation of part0; (b) implementation of part1.

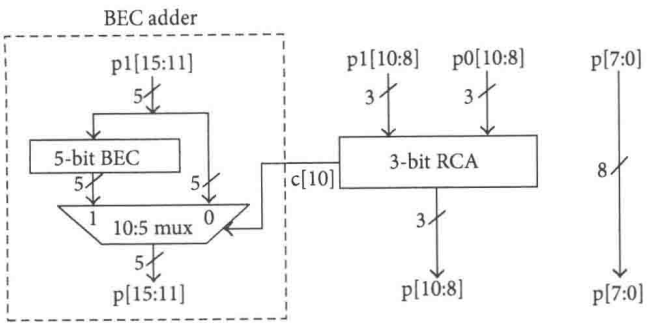

FIGURE 6: Hybrid final adder of 8-b multiplier.

TABLE 4: Function table of 5-BIT BEC and BECWC.

Input	BEC without carry	BEC with carry	
b[4:0]	x[4:0]	cy	x[4:0]
00000	00001	0	00001
⋮	⋮	⋮	⋮
11110	11111	0	11111
11111	00000	1	00000

3. The Hybrid Final Adder Design

In previous works, the hybrid final adder designs used to achieve the faster performance in parallel multipliers were made up of CLA (carry lookahead adder) and CSLA (carry select adder) [12, 13]. But due to the structure of the CSLA, it occupies more chip area and power than other adders. Thus to achieve the optimal performance, the proposed hybrid adder in this work uses BEC logic for fast summation of uneven input arrival time of the signals originating from the PPST. The BEC adder provides faster performance than carry save adder (CSA) and it consumes less area, low power than the carry select adder (CSLA) [14, 15].

3.1. Hybrid Adder for 8-b Multiplier. Once each part of the partial products has been reduced to height of one bit column, we get the final partial products as follows:

p0[10] p0[9] p0[8]

p7 p6 p5 p4 p3 p2 p1 p0

p1[15] p1[14] p1[13] P1[12] p1[11] p1[10] p1[9] p1[8].

The p0[10:8] are the exceeding carry bits of part0 and p1[15] is the carry bit of part1. The p[7:0] of part0 are directly assigned as the final products. To find the remaining p[15:8], we use the RCA and the BEC as shown in Figure 6.

The p0[10:8] and p1[10:8] are added using 3-bit RCA which finds p[10:8]. To obtain the remaining p[15:11], the p1[15:11] are assigned to the input of 5-bit BEC, which produce the two partial results p1[15:11] with Cin of "0" and the 5-bit BEC output with the Cin of "1." Depending on the Cout of RCA (c[10]), the mux provides the final p[15:11] without having to ripple the carry through p1[15:11].

The 8-bit multiplier uses a 5-bit BEC in the final adder, but for the large bit sized multipliers requires multiple BEC, and each of them requires the selection input from the carry output of the preceding BEC. Therefore, to generate the carry output from the BEC, an additional block is developed which is called BECWC. The detailed structures of the 5-bit BEC without carry (BEC) and with carry (BECWC) are shown in Figures 7(a) and 7(b). The BEC gets n inputs and generates n output; the BECWC gets n input and generates $n + 1$ output to give the carry output as the selection input of the next stage mux used in the final adder design of 16-b, 32-b, and 64-b multipliers. The function table of BEC and BECWC is shown in Table 4.

3.2. Variable-Size Hybrid Adder. The variable size of adder blocks always leads to faster performance than a fixed-size block adder [2, 17]; we, therefore, break down the ripple of gates in the BEC into variable-size groups according to the $\log_2 n$ method. Based on this approach, the final adder designs for 16-b, 32-b, and 64-b multipliers are shown in Figure 8.

In BECWC, the mux is getting n-bits of data input as it is input for selection input "0" side and $n + 1$-bits of data input from the BECWC output for selection input "1" side. Thus to make equal size of the inputs to the mux, the one-bit "0" is appending with the n-bits of the data input as "MSB" (most significant bit).

To analyze independently the effect of the proposed hybrid adder, the partitioned multiplier with CLA final adder is compared with the partitioned multiplier along with the proposed hybrid adder. The simulation results of partitioned W, D, H Baugh-Wooley multipliers with hybrid CPA are listed as first column in Tables 5, 6, and 7. The performance of hybrid CPA (comparison between the partioned multipliers with CLA and partitioned multipliers with hybrid CPA) and overall performance of proposed techniques (comparison between unpartitioned multiplier with CLA and partitioned multiplier with hybrid CPA) are listed as second column and third column, respectively, in Tables 5 to 7. The result analysis clearly shows that the speed increases with the word size of the multiplier. The hybrid CPA improves the speed of the W, D, H Baugh-Wooley multipliers by 19%, 20%, 15%, respectively, for 64-b size without area and power overhead. The overall improved performance is elaborated in result summary.

4. ASIC Implementation and Simulation Results

The ASIC implementation of the proposed design follows the cadence design flow. The design has been developed using Verilog-HDL and synthesized in Encounter RTL compiler using typical libraries of TSMC 65 nm technology. The Cadence SoC Encounter is adopted for Placement & Routing (P&R) (Encounter User Guide 2008). Parasitic extraction is performed using Encounter Native RC extraction tool. The extracted parasitic RC (SPEF format) is back annotated to Common Timing Engine in Encounter Platform for static timing analysis.

Table 5: Improved performance by hybrid CPA and overall performance in Wallace multiplier.

Partitioned Wallace multiplier with hybrid CPA				Performance of hybrid CPA			Overall performance		
Word size	Delay (ns)	Area (μm^2)	Power (μW)	Delay	Area	Power	Delay	Area	Power
16	1.58	4,137	0.71	10.22	−1.07	−2.89	14.13	−3.86	−7.57
32	1.98	14,758	3.39	13.91	−0.63	−1.80	21.42	−2.50	−5.60
64	2.35	54,225	17.28	19.52	0.41	0.34	28.57	−0.61	−2.49

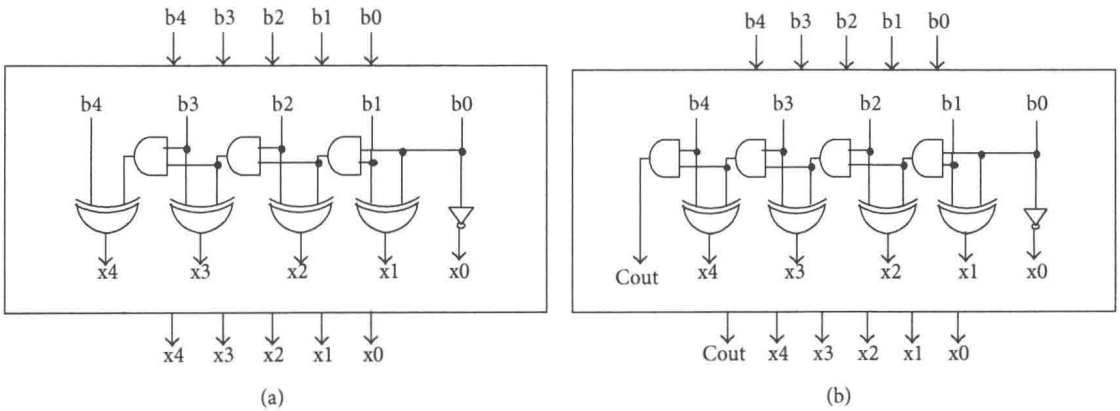

Figure 7: The 5-bit Binary to Excess-1 Code Converter: (a) BEC (without carry); (b) BECWC (with carry).

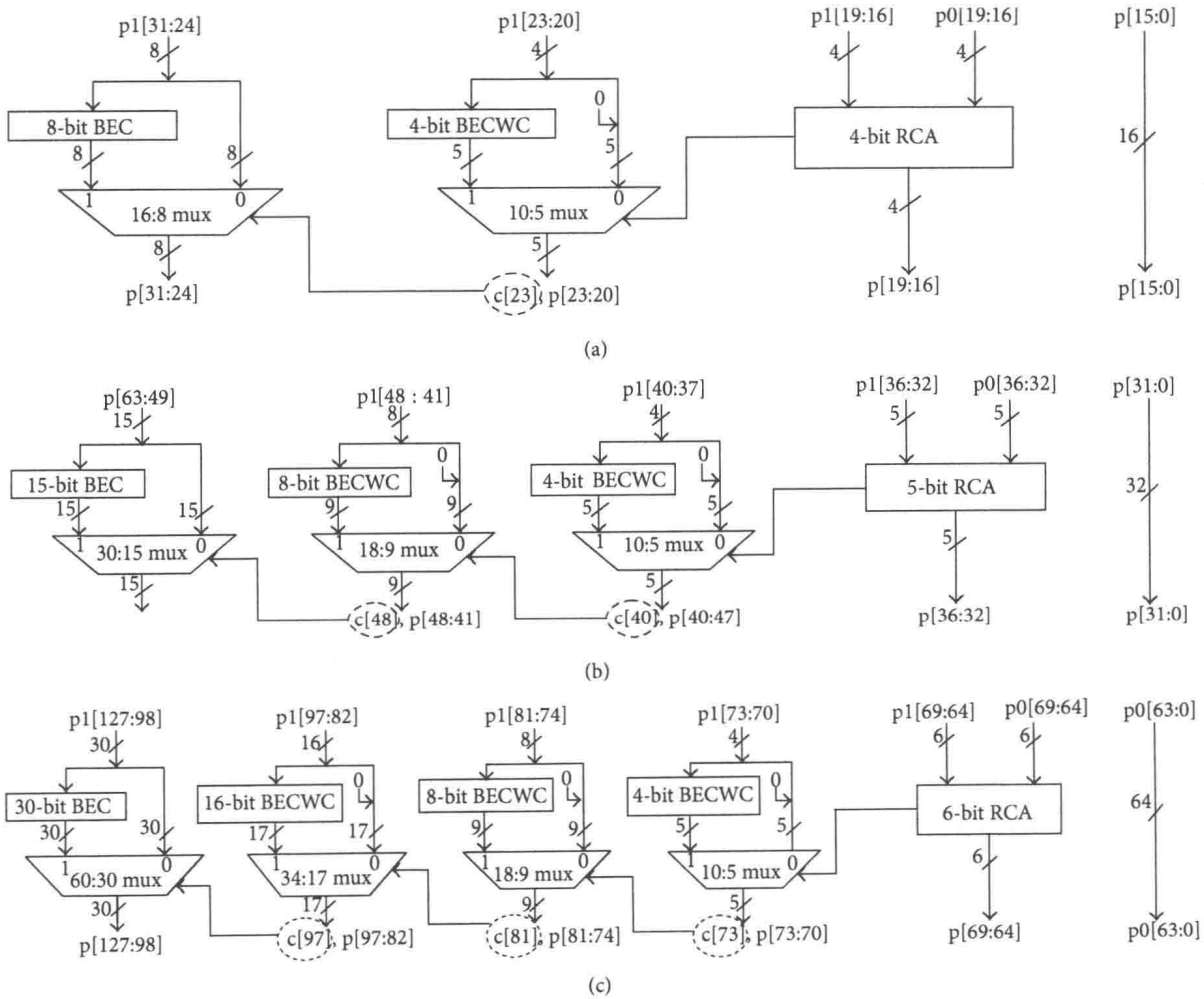

Figure 8: Hybrid final adder: (a) for 16-b multiplier, (b) for 32-b multiplier, and (c) for 64-b multiplier.