A Design Methodology for Carry-Lookahead Adders with Reconfigurability

Jin-Fu Li and Yu-Jane Huang
Advanced Reliable Systems (ARES) Laboratory
Department of Electrical Engineering
National Central University
Jungli, Taiwan 320, R.O.C.

Abstract

Fast addition plays an important role in advanced digital systems. Recently, reconfigurable adders have been widely employed to achieve real time processing of media signals. This paper presents a design-for-reconfigurability (DFR) technique for carry lookahead adders (CLAs). The DFR scheme only incurs a small amount of area cost and delay penalty. Experimental results show that the delay of an 64-bit reconfigurable CLA is only about 1.5ns with the $0.18\mu m$ technology. Compared with the original 64-bit CLA, the area overhead and delay penalty for realizing the DFR scheme in an 64-bit CLA are only about 4.7% and 2.7%, respectively.

1 Introduction

Fast addition is an essential arithmetic function for most advanced digital systems. It heavily impacts the overall performance of digital systems. Various adder structures can be used to execute addition [1, 2], such as serial and parallel structures. Most research works of adders are focused on the design of high-speed, low-area, or low-power adders [3–5].Recently, design of reconfigurable adders has received significant attentions. Reconfigurable adders usually are employed to achieve real-time processing of media signals [6–8]. Moreover, future systems will shift toward more programmable and reconfigurable integrated system on chips (SOCs) [9].Thus fast and reconfigurable adders for arithmetic computing are needed.

Several reconfigurable adder design methodologies have been reported in [10–13]. The PowerPC microprocessor has a reconfigurable ripple carry adder using additional bits for partitioning, such that multiple smaller adders are obtained [10]. For example, the Add/Compare block of the microprocessor can execute separate 8-bit, 16-bit, and 32-bit additions with a 36-bit reconfigurable adder. The adder has four 9-bit segments and each segment consists of 8-bit operand data and an additional partition bit. Each partition bit determines that the carry of the corresponding segment addition is blocked or propagated. This partition scheme is simple but it causes large delay penalty and area cost. In [11], a partition scheme for Brent-Kung carry lookahead adders is proposed. The partitioning is controlled by the propagate signals. In [12], a reconfigurable adder based on an optimized carry-skip scheme is reported. The partition approach does not incur significant area cost, power dissipation, and delay. In [13], a reconfigurable carry-skip adder has been proposed, which minimizes the product of energy and product. The proposed partition scheme enables an 64-bit adder to execute one 64-bit (64), two 32-bit (32,32), four 16-bit (16,16,16,16), or eight 8-bit (8,8,8,8,8,8,8,8) additions.

This paper proposes a design-for-reconfigurability (DFR) scheme for carry-lookahead adders (CLAs). The DFR scheme incurs only a small amount of area cost and additional delay. Experimental results show that the speed of an 64-bit reconfigurable CLA is about 1.5ns based on the $0.18\mu m$ technology. Compared with the 64-bit CLA without reconfigurability, the additional delay and area of the reconfigurable 64-bit CLA are only about 2.7% and 4.7%.

2 Carry-Lookahead Adder

Most of fast adders are based on being able to calculate the carry propagation much faster without having to wait for it to ripple through each bit of the adders. The carry-lookahead technique is the most commonly used scheme for accelerating carry propagation. A CLA calculates the carry-out for a block of bits in parallel to, and separately from, calculating the sum outputs of the block. Consider the addition of two operands $A = \{a_{n-1}, \cdots, a_0\}$ and $B = \{b_{n-1}, \cdots, b_0\}$. The carry lookahead algorithm introduces carry generate (G_i) and propagate (P_i) signals to reduce the carry propagation delay. The two signals can be defined as [1]

$$G_i = a_i b_i \text{ and } P_i = a_i + b_i. \quad (1)$$

As a result, the carry-out at bit position i can be expressed as a recursive equation:

$$c_{i+1} = G_i + P_i c_i. \quad (2)$$

Thus the carries of $A + B$ can be calculated in parallel according to each bit of operands. The carries are

$$\begin{aligned} c_1 &= G_0 + P_0 c_0 \\ c_2 &= G_1 + G_0 P_1 + P_1 P_0 c_0 \\ &\vdots \\ c_n &= G_n + P_n G_{n-1} + P_n P_{n-1} G_{n-2} + \cdots \\ &\quad + P_n P_{n-1} \ldots P_0 c_0. \end{aligned}$$

It is clear that the carry propagation delay is still long if the number of operands is large. Multilevel CLA networks can be used to cope with this problem by breaking the entire length of the operands into smaller blocks. That is, we may divide the n stages into blocks and have a separate carry-lookahead in each block. Then we may further reduce the delay of carry propagation by providing a carry-lookahead over blocks in addition to the internal lookahead within the block. For example, if the block size is 4, the block generate (G_0^*) and block propagate (P_0^*) signals of the first block can be obtained by

$$\begin{aligned} G_0^* &= G_3 + G_2P_3 + G_1P_2P_3 + G_0P_1P_2P_3, \\ P_0^* &= P_0P_1P_2P_3. \end{aligned} \quad (3)$$

Figure 1 depicts an example of 16-bit two-level CLA with 4-bit blocks, with G_0^*, G_1^*, G_2^*, G_3^* and P_0^*, P_1^*, P_2^*, P_3^*. Then the carries c_4, c_8, c_{12} and c_{16} can be generated by

$$\begin{aligned} c_4 &= G_0^* + c_0P_0^*, \\ c_8 &= G_1^* + G_0^*P_1^* + c_0P_0^*P_1^*, \\ c_{12} &= G_2^* + G_1^*P_2^* + G_0^*P_1^*P_2^* + c_0P_0^*P_1^*P_2^*, \\ c_{16} &= G_3^* + G_2^*P_3^* + G_1^*P_2^*P_3^* + G_0^*P_1^*P_2^*P_3^* \\ &\quad + c_0P_0^*P_1^*P_2^*P_3^*. \end{aligned} \quad (4)$$

As Fig. 1 shows, the two-level 16-bit CLA is comprised of three components: propagate-generate (PG) unit, carry-lookahead (CLA) unit, and sum generation unit (not shown in the figure). The sum generation unit can generate the sum with the expression: $s_i = a_i + b_i + c_i$. Figure 2 depicts an example of gate-level implementation of the level-1 CLA unit. The level-2 CLA unit can be implemented with the similar approach.

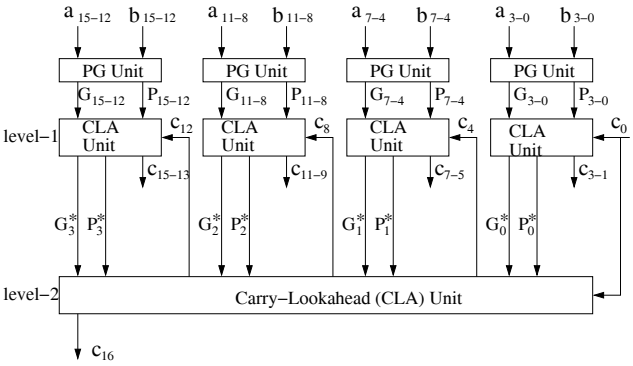

Figure 1: A 16-bit two-level CLA.

3 Design-for-Reconfigurability Technique

In this section we present a design-for-reconfigurability (DFR) technique for CLAs. We first use the 16-bit two-level CLA with 4-bit blocks shown in Fig. 1 as an example to explain the proposed DFR technique. Assume that two configurations: one 16-bit addition (16) and two 8-bit additions (8,8) are needed. Also, let c_8' be the carry input of the 8-bit adder on the left. Therefore, the following conditions must be satisfied when the configuration of two 8-bit adders

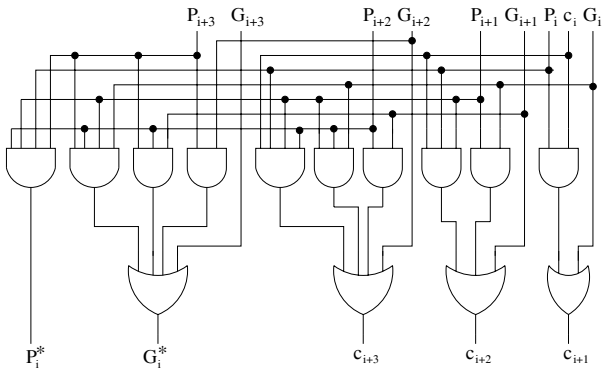

Figure 2: The level-1 CLA unit.

is set.

$$\begin{cases} c_8 = c_8' \\ c_{12} = G_2^* + c_8'P_2^* \\ c_{16} = G_3^* + G_2^*P_3^* + c_8'P_2^*P_3^* \end{cases} \quad (5)$$

Subsequently, we will show that the principle of the proposed DFR technique. Let α be the partition control signal, and $\alpha = 1$ or $\alpha = 0$ denote that the 16-bit CLA is not partitioned or partitioned, respectively. According to Eq. (1), we have the following expression

$$P_iG_i = G_i. \quad (6)$$

By replacing G_3 with P_3G_3, we can rewrite the G_0^* as follows:

$$\begin{aligned} G_0^* &= P_3G_3 + G_2P_3 + G_1P_2P_3 + G_0P_1P_2P_3, \\ &= P_3(G_3 + G_2 + G_1P_2 + G_0P_1P_2). \end{aligned}$$

Similarly, we can express the G_1^* and P_1^* as follows:

$$\begin{aligned} G_1^* &= P_7G_7 + G_6P_7 + G_5P_6P_7 + G_4P_5P_6P_7 \\ &= P_7(G_7 + G_6 + G_5P_6 + G_4P_5P_6) \\ &= P_7U_1 \\ P_1^* &= P_7P_6P_5P_4 \\ &= P_7V_1 \end{aligned}$$

Now we modify the G_1^* and P_1^* into two reconfigurable block propagate (rP_1^*) and block generate (rG_1^*) signals. The rG_1^* and rP_1^* are shown as follows:

$$\begin{aligned} rG_1^* &= P_7G_7 + G_6P_7 + G_5P_6P_7 + G_4P_5P_6P_7 \\ &= P_7(G_7 + G_6 + G_5P_6 + G_4P_5P_6) \\ &= \alpha P_7U_1 + \overline{\alpha}c_8' \\ rP_1^* &= P_7P_6P_5P_4 \\ &= \alpha P_7V_1 \end{aligned}$$

Therefore, the c_8, c_{12}, and c_{16} can be expressed as follows

$$\begin{aligned} c_8 &= rG_1^* + G_0^*rP_1^* + c_0P_0^*rP_1^*, \\ c_{12} &= G_2^* + rG_1^*P_2^* + G_0^*rP_1^*P_2^* + c_0P_0^*rP_1^*P_2^*, \\ c_{16} &= G_3^* + G_2^*P_3^* + rG_1^*P_2^*P_3^* + G_0^*rP_1^*P_2^*P_3^* \\ &\quad + c_0P_0^*rP_1^*P_2^*P_3^*. \end{aligned}$$

When $\alpha = 1$, the three equations are the same as those shown in Eq. (4). However, when $\alpha = 0$, the three equations are as follows

$$\begin{aligned} c_8 &= c'_8, \\ c_{12} &= G^*_2 + c'_8 P^*_2, \\ c_{16} &= G^*_3 + G^*_2 P^*_3 + c'_8 P^*_2 P^*_3. \end{aligned}$$

The three equations are the same as those expressed in Eq.(5). Thus the 16-bit CLA can be partitioned into two 8-bit CLAs when $\alpha = 0$. To realize the reconfigurable 16-bit CLA, we need one modified CLA block which can generate rG^*_1 and rP^*_1. Figure 3 shows an example of gate-level implementation of the modified CLA. As the figure shows, if $\alpha = 0$, $P^*_i = 0$ and $G^*_i = c'_j$. Therefore, the 16-bit two-level CLA shown in Fig. 1 can be modified as a reconfigurable 16-bit CLA with two possible configurations by replacing the second level-1 CLA unit with the modified CLA unit.

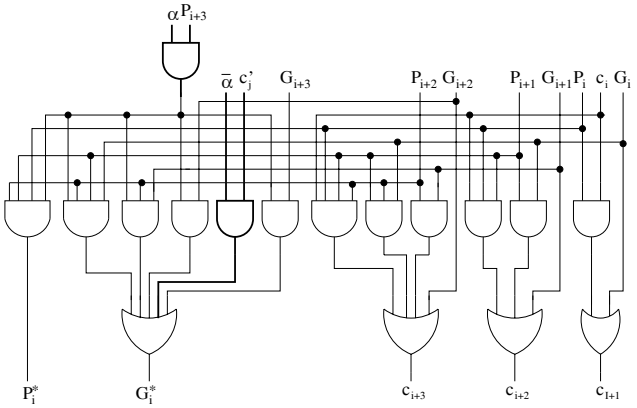

Figure 3: Modified level-1 4-bit CLA unit.

Another example is illustrated to show that the DFR scheme can be extended to CLAs with wider widths. Consider a 32-bit CLA with 4-bit blocks. A reconfigurable 32-bit CLA with three partitions, one 32-bit CLA (32), two 16-bit CLAs (16,16) and four 8-bit CLAs (8,8,8,8), can be obtained by replacing the second, fourth, and sixth level-1 CLA units with modified CLAs. Figure 4 shows the CLA network of the reconfigurable 32-bit CLA, where the shaded boxes denote the modified CLAs as shown in Fig. 3. The partitioning control signals are α and β. Table 1 lists the various partitions with respect to the state of the two control signals. When (β,α)=(0,0), the CLA is configured into four 8-bit CLAs with carry inputs c_0, c'_8, c'_{16}, and c'_{24}. When (β,α)=(0,1), the CLA is partitioned into two 16-bit CLAs with carry inputs c_0 and c'_{16}. When (β,α)=(1,1), the CLA is unchanged.

Table 1: Adder partitions with respect to control signals.

β	α	Partitions
0	0	Four 8-bit adders
0	1	Two 16-bit adders
1	1	One 32-bit adder

Subsequently, we want to show the reconfigurable 32-bit CLA can work as described above. As Fig. 4 shows, the P^*_i, G^*_i, P^{**}_i, and G^{**}_i are generated with the Boolean equations similar to those in Eq. (3). Also, the carry outputs of level-1 and level-2 CLAs are calculated with the Boolean equations similar to those in Eq. (4). The carry outputs c_{16} and c_{32} of the level-3 CLA are as follows

$$\begin{aligned} c_{16} &= G^{**}_0 + c_0 P^{**}_0, \\ c_{32} &= G^{**}_1 + G^{**}_0 P^{**}_1 + c_0 P^{**}_1 P^{**}_0. \end{aligned}$$

We first show that the configuration (8,8,8,8) is correct. When (α, β)=(0,0), $G^*_1 = c'_8$, $G^*_3 = c'_{16}$, $G^*_5 = c'_{24}$, and $P^*_1 = P^*_3 = P^*_5 = 0$. Therefore, we can obtain the following expressions:

$$\begin{aligned} G^{**}_0 &= c'_{16}, \\ P^{**}_0 &= 0, \\ G^{**}_1 &= G^*_7 + G^*_6 P^*_7 + c'_{24} P^*_6 P^*_7, \\ P^{**}_1 &= 0, \\ c_{24} &= G^*_5 = c'_{24}. \end{aligned} \tag{7}$$

By replacing G^{**}_0, P^{**}_0, G^{**}_1, and P^{**}_1 into the c_{16} and c_{32}, we obtain the following expressions:

$$\begin{aligned} c_{16} &= c'_{16}, \\ c_{32} &= G^{**}_1 = G^*_7 + G^*_6 P^*_7 + c'_{24} P^*_6 P^*_7. \end{aligned} \tag{8}$$

Also, $c_8 = c'_8$ can be shown the same as that for the 16-bit reconfigurable CLA. Therefore, we conclude that the configuration (8,8,8,8) is correct according to Eqs. (7) and (8).

Finally, we show that the configuration (16,16) can work correctly. When (α,β)=(1,0), $G^*_3 = c'_{16}$ and $P^*_3 = 0$. We can obtain the following expressions:

$$\begin{aligned} G^{**}_0 &= c'_{16}, \\ P^{**}_0 &= 0, \\ G^{**}_1 &= G^*_7 + G^*_6 P^*_7 + G^*_5 P^*_6 P^*_7 + G^*_4 P^*_5 P^*_6 P^*_7, \\ P^{**}_1 &= P^*_4 P^*_5 P^*_6 P^*_7 \end{aligned} \tag{9}$$

By replacing the G^{**}_0, P^{**}_0, G^{**}_1, and P^{**}_1 of c_{16} with the values described in Eq. (9), we obtain the c_{16} and c_{32} as follows:

$$\begin{aligned} c_{16} &= c'_{16}, \\ c_{32} &= G^{**}_1 + G^{**}_0 P^{**}_1 \\ &= G^*_7 + G^*_6 P^*_7 + G^*_5 P^*_6 P^*_7 + G^*_4 P^*_5 P^*_6 P^*_7 \\ &\quad + c'_{16} P^*_4 P^*_5 P^*_6 P^*_7. \end{aligned}$$

As the equations shown above, we see that the 32-bit CLA can correctly be partitioned into two 16-bit CLAs with $\alpha = 1$ and $\beta = 0$.

4 Analysis and Comparison

We have demonstrated the proposed reconfigurable design methodology on an 64-bit CLA with synthesizable Verilog RTL. Note that the UMC $0.18\mu m$ CMOS standard cell library is used to simulate the following results. Simulation results show that the speed of an 64-bit CLA without reconfigurability is about 1.46ns. Also, the area is about $17052\mu m^2$. On the other hand, the speed of the proposed 64-bit reconfigurable CLA is about 1.5ns when the 64-bit addition is executed. The area of the reconfigurable adder

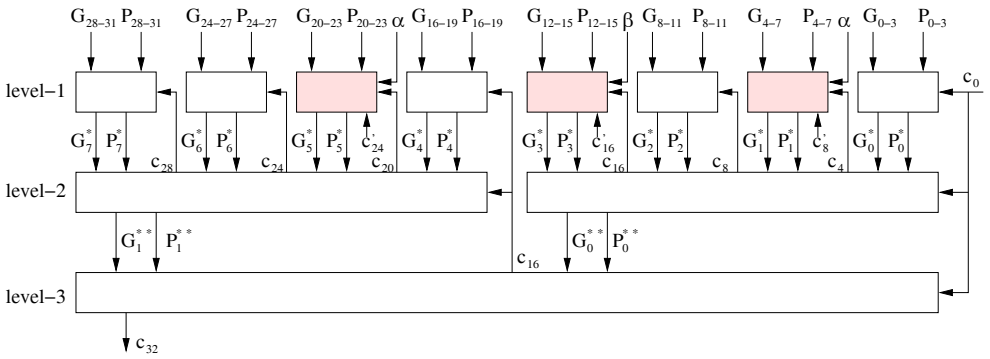

Figure 4: A 32-bit reconfigurable CLA with 4-bit blocks.

is about $17861\mu m^2$. Therefore, area overhead and delay penalty for the design-for-reconfigurability scheme are only about 4.7% and 2.7%, respectively.

Finally, we compared the proposed reconfigurable adder with the reconfigurable adders reported in the previous works. Table 2 summarizes the comparison results of the proposed 64-bit reconfigurable CLA and the adders reported in [11–13]. In the table, the BK and CSK denote the Brent-Kung CLA and carry skip adder, respectively. Data reported in the second column to fourth column are referred in [13], where AMS $0.35\mu m$ is assumed. As the table shows, the delay of the proposed 64-bit reconfigurable CLA is about 1.5ns. Although the proposed CLA is simulated with $0.18\mu m$ technology, the delay is only about 2.9ns by scaling the original delay with a factor of $0.35/0.18$. On the other hand, the proposed reconfigurable CLA dissipates only about 60pJ. Again, the energy of the proposed reconfigurable CLA is only about 117pJ by scaling the original energy with a factor of $0.35/0.18$. Compared with the previous works, the energy dissipation is lower than that of the other adders.

Table 2: Comparison with previous works.

64-bit addition	[11]	[12]	[13]	Proposed
Adder structure	BK	CSK	CSK	CLA
Reconfigurable	Y	Y	Y	Y
Technology	$.35\mu m$	$.35\mu m$	$.35\mu m$	$.18\mu m$
Supply Voltage	3.3V	3.3V	3.3V	1.8V
Area (μm^2)	74242	46404	64335	17861
Delay (ns)	4.9ns	6.5ns	4.9ns	1.5ns
Energy (pJ)	226	148	181	60

5 Conclusions

This paper has presented a design methodology of reconfigurable CLAs. A DFR scheme has proposed to divide a large CLA into multiple separate small ones. The DFR scheme only incurs a small amount of delay and area penalty. Experimental results show that the delay of an 64-bit reconfigurable CLA is only about 1.5ns based on UMC $0.18\mu m$ technology. Also, the 64-bit reconfigurable CLA has very low product of energy and delay. Furthermore, the delay penalty and area overhead of the DFR scheme are only about 2.7% and 4.7%, respectively.

Acknowledgment

This work was supported in part by the National Science Council, R.O.C., under Contract NSC 93-2215-E-008-017.

References

[1] I. Koren, *Computer Arithmetic Algorithms*, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 07632, 1993.

[2] J.-C. Lo, "A fast binary adder with conditional carry generation", *IEEE Trans. Computers*, vol. 46, no. 2, pp. 248–253, Feb. 1997.

[3] A. Tyagi, "A reduced-area scheme for carry-select adders", *IEEE Trans. Computers*, vol. 42, no. 10, pp. 1163–1170, Oct. 1993.

[4] C. Nagendra, M. J. Irwin, and R. M. Owens, "Area-time-power tradeoffs in parallel adders", *IEEE Trans. Circuits and Systems II: Analog and Digital Signal Processing*, vol. 43, no. 10, pp. 689–702, Oct. 1996.

[5] T.-Y. Chang and M.-J. Hsiao, "Carry-select adder using single ripple-carry adder", *Electronics Letters*, vol. 34, no. 22, pp. 2101–2103, Oct. 1998.

[6] A. Peleg and U. Weiser, "MMX technology extension to the Intel architecture", *IEEE Micro*, vol. 16, no. 4, pp. 42–50, Aug. 1996.

[7] M. Tremblay, J.-M. O'Connor, V. Narayanan, and H. Liang, "VIS speeds new media processing", *IEEE Micro*, vol. 16, no. 4, pp. 10–20, Aug. 1996.

[8] R.-B. Lee, "Subword parallelism with MAX-2", *IEEE Micro*, vol. 16, no. 4, pp. 51–59, Aug. 1996.

[9] R. Hartenstein, "Reconfigurable computing: a new business model and its impact on SoC design", in *Euromicro Symp. on Digital Systems Design*, Sept. 2001, pp. 103–110.

[10] M. S. Schmookler, M. Putrino, A. Mather, J. Tyler, and H. V. Nguyen, "A low-power, high-speed implementation of a PowerPC^{TM} microprocessor vector extension", in *IEEE Symposium on Computer Arithmetic*, 1999, pp. 12–19.

[11] A. A. Farooqui, V. G. Oklobdzija, and F. Chechrazi, "64-bit media adder", in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS)*, Orlando, May 1999.

[12] S. Perri, P. Corsonello, and G. Cocorullo, "A 64-bit reconfigurable adder for low power media processing", *Electronics Letters*, vol. 38, no. 9, pp. 397–399, Apr. 2002.

[13] S. Perri, P. Corsonello, and G. Cocorullo, "A high-speed energy-efficient 64-bit reconfigurable binary adder", *IEEE Trans. VLSI Systems*, vol. 11, no. 5, pp. 939–943, Oct. 2003.