# Low-Power 4-2 and 5-2 Compressors

*Karuna Prasad*

Morphics Technology Inc.
Campbell,California, USA
kprasad@morphics.com

*Keshab K. Parhi*

Department of Electrical and Computer Engineering
University of Minnesota, Minneapolis, USA
parhi@ece.umn.edu

## Abstract

*This paper explores various low power higher order compressors such as 4-2 and 5-2 compressor units. These compressors are building blocks for binary multipliers. Various circuit architectures for 4-2 compressors are compared with respect to their delay and power consumption. The different circuits are simulated using HSPICE. A new circuit for 5-2 compressor is then presented which is* 12% *faster and consumes* 37% *less power.*

## 1: Introduction

Multiplication is the basic arithmetic operation which is important in several microprocessors and digital signal processing applications. Microprocessors use multipliers within their arithmetic logic units, and digital signal processing systems require multipliers to implement DSP algorithms such as convolution and filtering. In most systems, the multiplier lies directly within the critical path due to which, the demand for high-speed multipliers is continuously increasing. However, due to portability and reliability issues, the power consumption of the multipliers has become equally important. All this has resulted in the development of novel circuit design techniques, with the aim of reducing the power dissipation of multipliers without compromising their speed performance.

A multiplication process essentially consists of generating the partial products' matrix, reducing the matrix to two rows followed by the final carry propagation addition. Modified Booth recoding and various other multi-bit recoding algorithms have proved to be useful in reducing the number of partial products generated. In the next step, speeding up the adding operation is highly critical for reducing the partial products' matrix. This step usually contributes the most to the delay, power and area of the multiplier. For this purpose, use of higher order compressors instead of the conventional 3:2 compressors have been explored. This reduces and simplifies the interconnections because they can sum up the partial products in the form of a binary tree. In the final step, a fast carry propagate adder generates the multiplier's final output by adding the two rows of partial products.

This work was supported in part by the National Science Foundation under grant number CCR-9988262.

As far as the circuit implementations are concerned, pass-transistor logic is emerging as an attractive replacement of the conventional satic CMOS logic, especially in the design of arithmetic units such as adders and multipliers. Fewer transistors are required by the pass-transistor logic to implement basic logic functions, which translates into lower input gate capacitance and lower power dissipation as compared with conventional static CMOS [1][2]. In this paper, we explore low power circuits, based on pass-transistor logic, for 4:2 and 5:2 compressors.

## 2: Partial Product Accumulation

Compressors are the fundamental building blocks used for accumulating the partial products during the multiplication process. Therefore, improving the power efficiency of these architectures can lead to significant savings of the power consumed by the entire multiplier. The compressors are combined to form a Wallace tree or a Dadda tree structure. A Wallace tree is an implementation of an adder tree designed for minimum propagation delay. Rather than completely adding the partial products in pairs like the ripple adder tree does, the Wallace tree uses carry save form to sum up all the bits of the same weights in a merged tree, in order to reduce the number of partial products bits to two rows. The advantage of the tree is that there is speed increases with log of the operand length, while this increase is linear in the case of iterative arrays. Dadda tree is a generalized form of Wallace tree adder. The number of adders needed in the Dadda tree is less than the Wallace tree but the overall interconnections are more irregular in the Dadda tree, making it difficult to layout in VLSI design [3].

Previously, full adders or 3:2 compressors were used for accumulation, in which 3 equally weighted bits were combined to produce two bits: one (the carry) with weight of n+1 and the other (the sum) with weight n. Each layer of the tree therefore reduced the number of vectors by a factor of 3:2. Now, with the development of fast 4:2 compressors, they are being used for the same purpose. The Wallace trees have a delay behavior of approximately $\Theta(log_{4/2}(mn))$, where m is the number of multiplicand bits, n is the number of encoded multiplier bits and 4/2 is the compression ratio of the 4:2 compressor used. Wallace Tree suffers from irregular routing and from its non-unique overall structure, i.e., there are several ways of building a particular Wallace tree. For example, a carry generated in one column may be introduced in the next more significant column at different places, e.g. close to the tree root or close to the output. The tree has as many layers as is necessary to reduce the number of vectors to two (a carry and a sum). 16 X 16 bit multiplier would need 2 levels of cascaded 4:2 compressors and 32 X 32 bit multiplier would need 3 levels of cascaded 4:2 compressors. Hence, on an average the output of each compressor would either be fed to another compressor or it'll go as an input to the final carry propagation adder.

## 3: Compressor Architectures

Figure 1 shows the block diagram of a 3:2 compressor, also known as a full-adder. It has three input bits of equal weight and two output bits, sum and carry. Sum has the same weight as the input bits but the carry has one greater binary bit weight. This compressor has a maximum of two XOR delays.

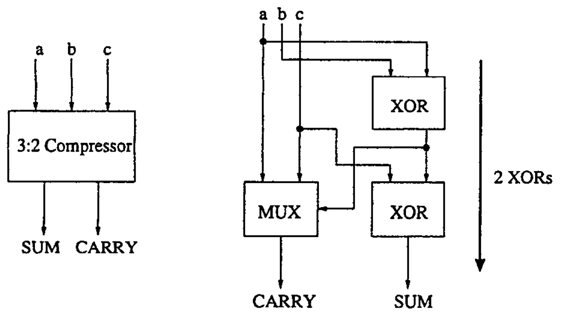

Figure 1: A 3:2 compressor

Figure 2 shows the block diagram of a 4:2 compressor. It has five inputs including a carry-in from the neighbouring cell of one binary bit lower significance. It has three outputs including a carry-out to the one greater significance cell. A 4:2 compressor can be built using 3:2 compressors. It consists of two 3:2 compressors in series and involves a critical path delay of 4 XORs. An alternative implementation is shown in Figure 3. This implementation is better and involves a critical path delay of 3 XORs, hence reducing the critical path delay by 1 XOR [4][5].

Figure 4 shows the block diagram of a 5:2 compressor.

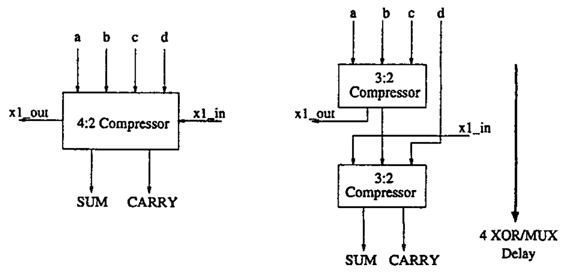

Figure 2: A 4:2 compressor

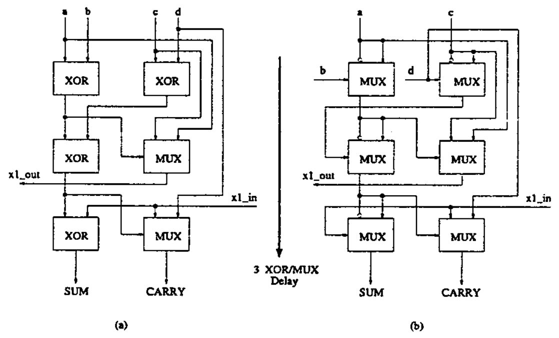

Figure 3: An alternative implementation of 4:2 compressor

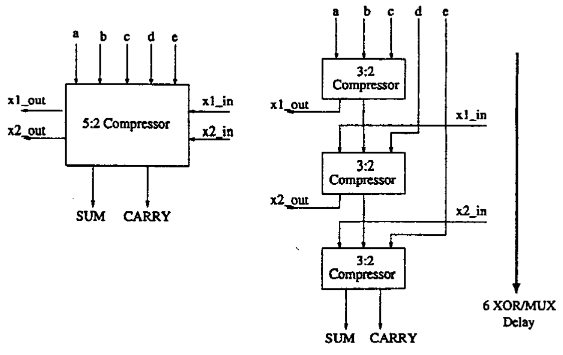

Figure 4: A 5:2 compressor

It has 5 direct inputs and 2 additional carry-in bits, from a neighbouring one-lower significant cell. It has four outputs, among which two of them are carry-out bits to the one greater significant cell, carry bit is of one greater significance and last is the sum bit. A 5:2 compressor can be built using three 3:2 compressors, in which case it involves a critical path delay of 6 XORs.

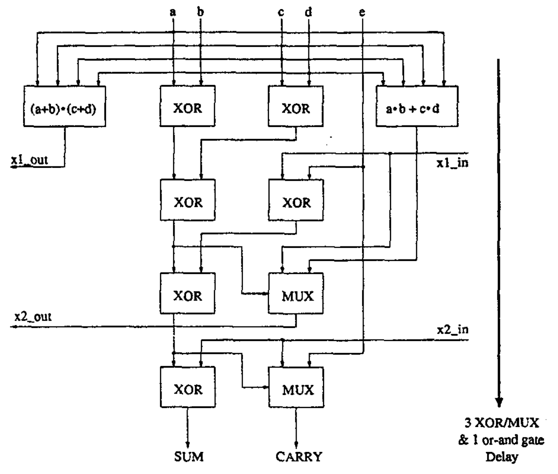

Figure 5: 5:2 compressor - Implementation 1

An implementation of 5:2 compressor was proposed in [6]. Figure 5 shows the block diagram of this 5:2 compressor implementation. At the first glance, it can be assumed that this implementation would have a critical path delay of 4 XORs. But a detailed analysis reveals that the critical path delay consists of 3 XORs + 1 andor gate delay.

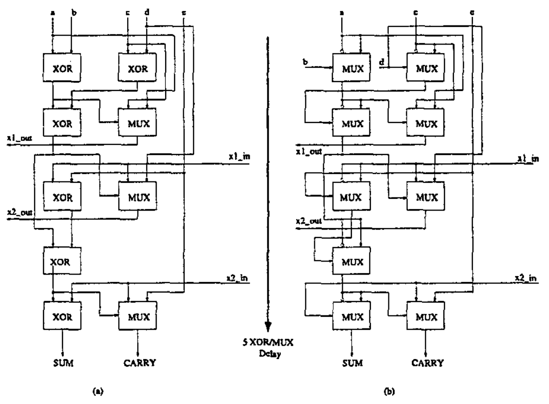

Figure 6: 5:2 compressor - Implementation 2

Figure 6 shows a block diagram of the second implementation of the 5:2 compressor. This implementation is derived in a similar fashion as that shown in Figure 3. It, basically, consists of a 4:2 compressor followed by a 3:2 compressor and has a critical path delay of 5 XORs. As it will be shown in the next section, although this circuit has 5 XORs delay it is faster and more power efficient as compared to the implementation shown in Figure 5. The detailed equations are written below.

$$
\begin{aligned}
sum &= a \oplus b \oplus c \oplus d \oplus e \oplus x1in \oplus x2in &(1)\\
carry &= (a \oplus b \oplus c \oplus d \oplus e \oplus x1in) \cdot x2in &(2)\\
&\quad + \overline{(a \oplus b \oplus c \oplus d \oplus e \oplus x1in)} \cdot e &\\
x1out &= (a \oplus b) \cdot c + \overline{(a \oplus b)} \cdot a &(3)\\
x2out &= (a \oplus b \oplus c \oplus d) \cdot x1in + \overline{(a \oplus b \oplus c \oplus d)} \cdot d &(4)
\end{aligned}
$$

## 4: Simulation Results and Analysis

All the simulations for computing delay and power consumption are done using **HSPICE**, $0.35\mu$m technology and at 3.3V. The outputs of all the simulated circuits were loaded appropriately using dummy NMOS gates. Diffusion load was assumed to be 20% of the gate load. The transistors were sized for equal rise and fall delay and an overall minimum delay. Matlab was used to generate 1024 random inputs, with an equal probability of 0.5 for 0 and 1, for the power simulation. Average power consumption was computed after feeding these 1024 random inputs. Delay was computed as the time interval between the time at which the input signal was at 50% of its full value, i.e., $1.65V$ and the time at which the output signal was also at $1.65V$. Worst case rise delay or fall delay for any of the outputs are recorded as the delay of the respective circuit. The x1in and/or x2in are not fed seperately to the compressor. The x1out and/or x2out generated by the circuit are fed as x1in and/or x2in. This should not make any difference as far as delay or power consumption of the compressor is concerned, since the input and output signals are seeing the same load as they actually would. Although, the arithmetic value of result would not be correct.

As might be evident from the previous section, all the compressors have XORs and MUXs on their critical path. Hence, these two basic blocks need to be implemented efficiently. Figure 7 shows the two different implementations of an XOR gate. Implementation $(a)$ has been proved to have the best performance for a CMOS XOR [7]. An XOR can also be implemented as a mux as in implementation $(b)$. Figure 8 shows two different efficient implementations of the MUX [1][4][5] using pass-transistor logic. There is one more efficient implementation based on the non-full swing nature of pass-transistor logic[4]. But, this MUX would suffer from severe signal degradation if two 4:2 compressors are cascaded in series. Hence, all our simulations involve use of full swing MUXs.

In order to obtain the optimum 4:2 compressor circuit, Figure 3 was simulated using the three different XOR/MUX

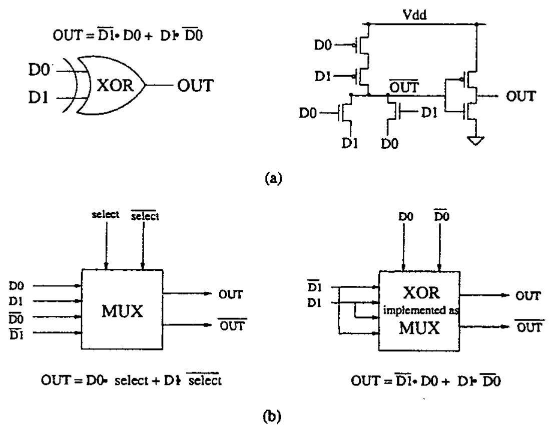

Figure 7: Two implementations of the XOR

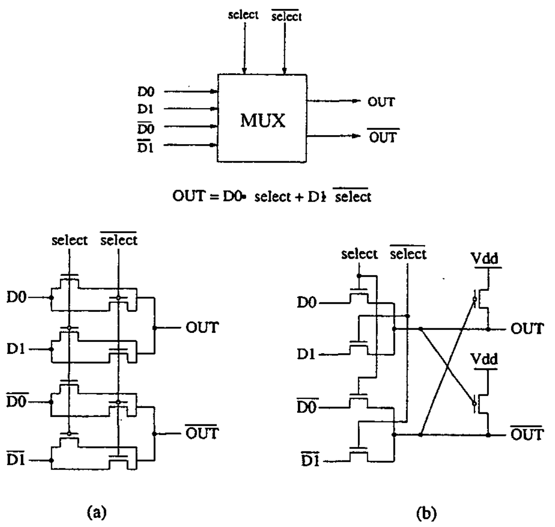

Figure 8: Two implementations of the MUX

implementations shown in the Figures 7(a), 8(a) and (b). The loading at the output of the compressor is another compressor. Hence, the sum and carry outputs of the compressor are appropriately loaded. The time delays and power consumption parameters for the four different implementations of the 4:2 compressor are tabulated in Table 1. The first implementation has XOR implemented as in Figure 7(a) and MUX as in Figure 8(a). The second implementation has the same XOR with the MUX as in Figure 8(b). The third implementation implements XOR and MUX both as in Figure 8(a) and the fourth one implements both the XOR and MUX as in Figure 8(b). All the circuits were simulated at a clock rate of 1.2ns. Transistor sizing, in each case, was done to achieve the minimum optimum delay.

The results in Table 1 indicate that implementation 4 is the best implementation. It not only has the least time delay but it also has the least power consumption. This also reveals that the XOR/MUX implemented as in Figure 8 (b) is optimum for the compressor circuits. Hence, for further simulation of 5:2 compressors, XORs and MUXs are implemented as in Figure 8(b).

In the case of 5:2 compressors, Implementation 1 shown in Figure 5 and Implementation 2 shown in Figure 6 (b) are simulated. The compressor is loaded with another compressor and hence, the two outputs are loaded accordingly. The time delays and power consumption parameters for the two different implementations of the 5:2 compressor are tabulated in Table 2. All the circuits were simulated at a clock rate of 1.2ns. In case of implementation 1, i.e. Figure 5, the two functions $\overline{(a \cdot b) + (c \cdot d)}$ and $\overline{(a + b) \cdot (c + d)}$ were implemented as $(\bar{a} + \bar{b}) \cdot (\bar{c} + \bar{d})$ and $(\bar{a} \cdot \bar{b}) + (\bar{c} \cdot \bar{d})$, respectively. By implementing these functions as negative logic functions we avoid the two inverters needed at the output of the otherwise positive logic functions and hence, we reduce the critical path delay. Minimum sized transistors were found to be optimum for implementing these two functions.

The results in Table 2 indicate that the second implementation outperforms the first. In terms of time delay it achieves 11.67% speed improvement and in case of power consmuption it achieves 37.02% of improvement.

Table 1: A Comparison between the different implementations of the 4:2 compressor

| | Time Delay ($10^{-9} secs$) | Power Consumption ($10^{-4} watts$) |
|---|---|---|
| Implem. 1 | .736 | 8.362 |
| Implem. 2 | .600 | 4.875 |
| Implem. 3 | .392 | 4.901 |
| Implem. 4 | .338 | 2.718 |

Table 2: A Comparison between the two implementations of the 5:2 compressor

| | Time Delay ($10^{-9} secs$) | Power Consumption ($10^{-4} watts$) |
|---|---|---|
| Implem. 1 | .540 | 7.360 |
| Implem. 2 | .477 | 4.635 |

## 5: Conclusion

This paper has described various circuit implementations of 4-2 and 5-2 compressors, followed by the simulation results of these circuits. A new faster circuit for a 5-2 compressor is presented and is proved to achieve better performance for both delay and power consumption. It achieves 11.67% improvement in speed and 37.02% improvement in power consumption, hence obtaining an overall improvement of 44.37% in the power-delay product.

## 6: References

[1] K. Yano, T. Yamanaka, T. Nishida, M. Saito, K. Shimohigashi, and A. Shimizu, "A 3.8ns cmos 16x16-b multiplier using complementary pass-transistor logic", *IEEE Journal of Solid-State Circuits*, vol. 25, pp. 388–395, April 1990.

[2] Neil H.E. Weste and K.Eshraghian, *Principles of CMOS VLSI Design: A systems Perspective*, Addison-Wesley Publishing Company, 1993.

[3] Keshab K. Parhi, *VLSI Digital Signal Processing Systems*, John Wiley and Sons, Inc., 1999.

[4] C.F.Law, S.S.Rofail, and K.S.Yeo, "Low power circuit implementation for partial-product addition using pass-transistor logic", *IEE Proc-Circuits Devices Syst.*, vol. 146, pp. 124–129, June 1999.

[5] J.M. Wang, S.C. Fang, and W.S. Feng, "A 4.4-ns cmos 54x54-b multiplier using pass-transistor multiplexer", *IEEE 1994 Custom Integrated Circuits Conference*, pp. 599–602, 1994.

[6] O.Kwang, K. Nowaka, and E.E. Swartzlander, "A 16-bit x 16-bit mac design using fast 5:2 compressors", pp. 235–243, 2000.

[7] J.M.Wang, S.C.Fang, and W.S.Feng, "New efficient designs for xor and xnor functions on the transistor level", *IEEE Journal of Solid-State Circuits*, vol. 29, pp. 780–786, July 1994.