Multiplexer Based Adder for Media Signal Processing

Aamir A. Farooqui[1], Vojin G. Oklobdzija[1,2], Farzad Chechrazi[1]

[1]LSI Systems Laboratory,
SONY US Research Laboratories,
San Jose, CA 95134.
e-mail: aamirf@lsi.sel.sony.com.

[2] Integration Corp.,
Berkeley, CA 94708.
email: vojin@nuc.berkeley.edu.
http://www.integr.com.

Abstract

This paper presents the design of a highly re-configurable adder, which has been optimized for speed and area. Since pass transistor based multiplexer is the fastest element in standard CMOS logic, we designed the adder using only multiplexers and 2-input inverted logic gates. This adder is the hybrid of binary carry lookahead adder of Brent, and carry select adder. By using the hybrid approach, the area and wiring of the adder is reduced by 1/2, keeping the adder delay proportional to O (log n). The critical path of the 64-bit partitioned adder consists of 7 two-to-one multiplexers and 1 XOR gate. The adder can be partitioned to support a variety of data formats, it can add two 64-bit operands, four 32-bit operands, eight 16-bit operands, or sixteen 8-bit operands. The adder can be used for multi-media applications, and is well suited for VLIW processors. The adder is described in Verilog, and synthesized using Synopsys tool. The critical path delay of the 64-bit adder is 0.9 ns at typical conditions in standard cell 0.25 um technology.

Introduction

Addition is an important operation affecting the speed and total performance of digital signal processors. High-speed adders can be realized using most widely used carry look-ahead (CLA) [1], carry select [2], or binary carry look-ahead [3,4] techniques. CLA adders can be realized in two gate levels provided there is no limit on fan in/out. With the shrinking VLSI dimensions and low voltage operation, it is not cost effective to have more than few transistors in series, which limits the efficiency of CLA adders. The carry select adders reduces the computation time by pre-computing the sum for all possible carry bit values (i.e., '0' and '1'). After the carry becomes available the correct sum is selected using a multiplexer. Carry select adders (CSA) are in the class of fast adders, but they suffer from fan-out limitation since the number of multiplexers that need to be driven by the carry signal increases exponentially. In the worst case, a carry signal is used to select n/2 multiplexers in an n-bit adder. The binary carry lookahead adder is a variant of CLA. It computes the carry in a binary tree fashion (using an associative operator), and the sum is generated using the intermediate carries. The Brent's binary carry lookahead adder produces a highly regular structure with high speed (proportional to *O(log n)*), at the expense of increased wiring and area. The Brent adder has not been in widespread use because of the additional delay introduced by the exponentially growing interconnection complexity.
In this paper we have presented the design of a fast 64-bit adder using multiplexers. This approach is the hybrid of Brent [3] and carry select [2] method. The adder has been implemented, using the '*o*' operator as described by [3,4] using four-bit groups, instead of binary. By using four-bit groups, the number of wires and hardware has been reduced by *1/2*, keeping the adder delay proportional to *O(log n)*. The carry select method [2] works in parallel with the carry generation, and calculates the two sums based on $C_{in} = 0$, and $C_{in} = 1$ for a group. When the actual carry for a group becomes available via fast carry generation, the correct sum is selected using a multiplexer. Pass-transistor multiplexers, have been used in the carry and sum generation as in [6]. The critical path of the 64-bit partitioned adder is equivalent to 7 two-to-one multiplexers and 1 XOR gate.
Multi-media processors with the wider and varying word lengths, require highly re-configurable adders, which can be reconfigured for variety of data formats. The proposed adder has been designed for multi-media applications. It can add two double word, four word, eight half-word, or sixteen byte operands, along with the generation of carry out and Overflow (OV) for saturation arithmetic. Due to its highly structured design, and generation of carries in groups of four, the adder fits well for multi-media applications as a partitioned adder.
The paper is organized as follows. Section 1 provides the mathematical analysis based on Brent's '*o*' operator for the four bit carry. A simple solution for partitioning an n-bit adder into n/w (w = word length of the data type) blocks is provided in Section 2. The VLSI implementation, and simulation results are presented in Section 3 and 4 respectively.

1. Mathematical Analysis:

Let, $A = a_{n-1}, a_{n-2} \ldots a_1, a_0$, and $B = b_{n-1}, b_{n-2}, \ldots b_1, b_0$ be the two input operands, with a_{n-1} and b_{n-1} be the most significant bits. The generate and propagate signal at bit position "i" are given by; $g_i = a_i \bullet b_i$, and $p_i = a_i \otimes b_i$,(where: $\bullet$ = AND operation and $\otimes$ = XOR operation). The Carry out from bit position "i" is given by; $C_i = g_i + g_{i-1} \bullet p_i$ (where + = OR operation) provided $C_0 = 0$. The "o" operator as defined by [3] is given as follows:

$$(g, p) \text{ o } (g', p') = (g + (p \bullet g'), p \bullet p') \tag{1}$$

The group Generate (G) and Propagate (P) are given by:

$$(G_i, P_i) = (g_0, p_0) \text{ if } i = 0 \ \& \ (g_i, p_i) \text{ o } (g_{i-1}, p_{i-1}) \text{ if } 0 < i < n \tag{2}$$

In [3,4,5], using (1), the generate and propagate signals for each level (k) of the adder are generated using the following combination:

$$(G_{i+2}k, P_{i+2}k) = (g_{i+2}k, p_{i+2}k) \text{ o } (g_i, p_i) \text{ for } 0 < k < \log n \tag{3}$$

In the proposed implementation for 'n' bits, at k = 0 (first level) n/2 generate and propagate signals are produced using the following combination:

$(G_{2i+1}, P_{2i+1}) = (g_{2i+1}, p_{2i+1}) \circ (g_{2i}, p_{2i})$ for $0 < i < n/2$ (4)

At the second level n/4 signals are produced (by grouping the signals generated at the first level) using (4) but limiting i to n/4. These signals are the four-bit group generate and propagate signals, their value for 4-bit case is given below, and their grouping is shown in Fig. 1.

$(g_{10}, p_{10}) = (g_1, p_1) \circ (g_0, p_0)$ and

$(g_{32}, p_{32}) = (g_3, p_3) \circ (g_2, p_2)$ at k = 0 (at first level) (5)

$(G_{30}, P_{30}) = (g_{32}, p_{32}) \circ (g_{10}, p_{10})$ (at second level) (6)

In this realisation no (g_2, p_2) or intermediate even carry is generated, because these are generated within the conditional sum adders. Once we have the 4-bit group carries, the carries in multiplies of 4 are generated using (2). Fig. 1 shows the generation of carries for a 16-bit adder. This technique results in minimum wiring and area, for n bits, approximately $2n/2^k$ signals are generated at each level of the adder in contrast to [3, 5] which requires $2(n-2^k)$ signals. A comparison of [5], and the proposed adder implementation using 0.25 um standard cell library is given in Table-1. It is clear from Table-1 that the wiring and area of Brent's adder [3,5] increases exponentially with the increase in the number of bits. In contrast the proposed adder offers linear increase in wiring and area, keeping the delay equivalent to Brent's adder.

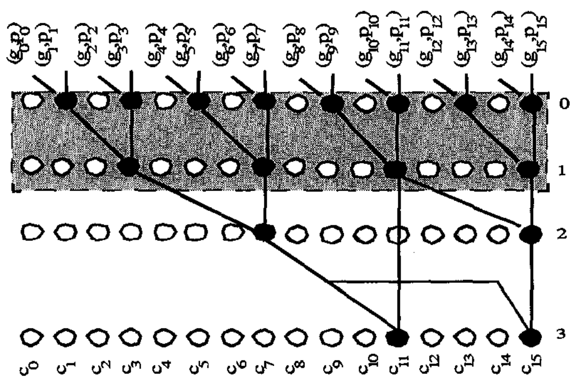

Fig. 1. Carry generation scheme for the proposed 16-bit adder.

Adder	Proposed			Brent and Kung [5]		
	Delay (ps)	Area	# of Nets	Delay (ps)	Area	# of Nets
8-Bit	625	194	181	560	190	174
16-Bit	665	460	428	670	458	421
32-Bit	710	850	764	767	1155	1136
64-Bit	903	1517	1316	869	2646	2554

Table 1. Comparison of delay, cell area, and # of wires in Brent & Kung[5] and the Proposed adder in 0.25um standard cell library at 2.5V and 25oC.

2. Partitioning:

The main motivation behind the design of this adder is to calculate multiple independent additions and their associated flags/conditions (OV, C_{out}) with different word-lengths using only one adder. This is the key requirement for Media processors. It seems simple to control the partition of an adder, by controlling the carry signal of each partitioned adder. In reality, this is not feasible because the carry chain is on the critical path of the adder and if we insert the control gate on the carry chain then the delay of the largest word size increases in proportion to the number of partitions. The other problem is that, in an adder we need a block carry (carry out) signal at three places:

1. Generation of Sum
2. Calculation of rest of the carries (in case of CLA of other fast adders), and
3. Ceneration of C_{out}, OV conditions.

One solution to this problem (as used in Motorola's "AltiVec" [8]) is to insert one extra bit for each partition, by forcing this bit to '0' ('1') partition (no partition) of the adder is achieved. However, this is not a good solution because it not only increases the adder size, but also results in higher delay. In this paper we provide a simple solution for partitioning the adder without producing any delay in the critical path and producing all the three carries mentioned above. In the following, we will show the partition of a 16-bit adder into four 4-bit adders, but it can be extended to any partition in multiples of 4. Consider a 16 - bit adder (Fig. 2) formed using the 4-bit groups. The group generate and propagate signals are G_{0-3}, P_{0-3}, for the first group, G_{4-7}, P_{4-7} for the second group and so on.
Now if we want to divide this adder into 4 adders of 4-bit each, then we need to make C_3 equal zero for the block generating Sum_{4-7}, C_7 equal zero for the block generating Sum_{8-11}, and C_{11} equal zero for the block generating Sum_{12-15}. At the same time we need these carries for the generation of C_{out} and OV conditions. If we just make $C_3 = 0$ then, we get wrong C_7, C_{11}, C_{15} due to their dependency on C_{0-1} and P_{2-3}, and the same applies for the other carries. The only way to solve this problem is to make $P_{4-7} = 0$ (P_{8-11}, P_{12-15} for the other partitions). By making this propagate signal zero, all the carries dependent on this propagate signal will become '0' and at the same time the C_{out} generated by a block remains valid for the calculation of OV condition. Since propagate signal is not on the critical path (less loading then generate and carry signals), its control is easy and does not require any significant delay. The only problem is the correct selection of the sum due to carry in of '0' since the OV condition requires a delay of an XOR gate after the generation of the carry signal. Therefore, making the carry in = 0 for sum selection does not cost any further delay, and at the same time reduces the load on carry signal.

3. Implementation:

Based on the analysis presented in Sections 1 and 2, we have designed a 64-bit partitioned adder. Fig. 2 shows the 16-bit block of this adder. The partition of the adder is performed

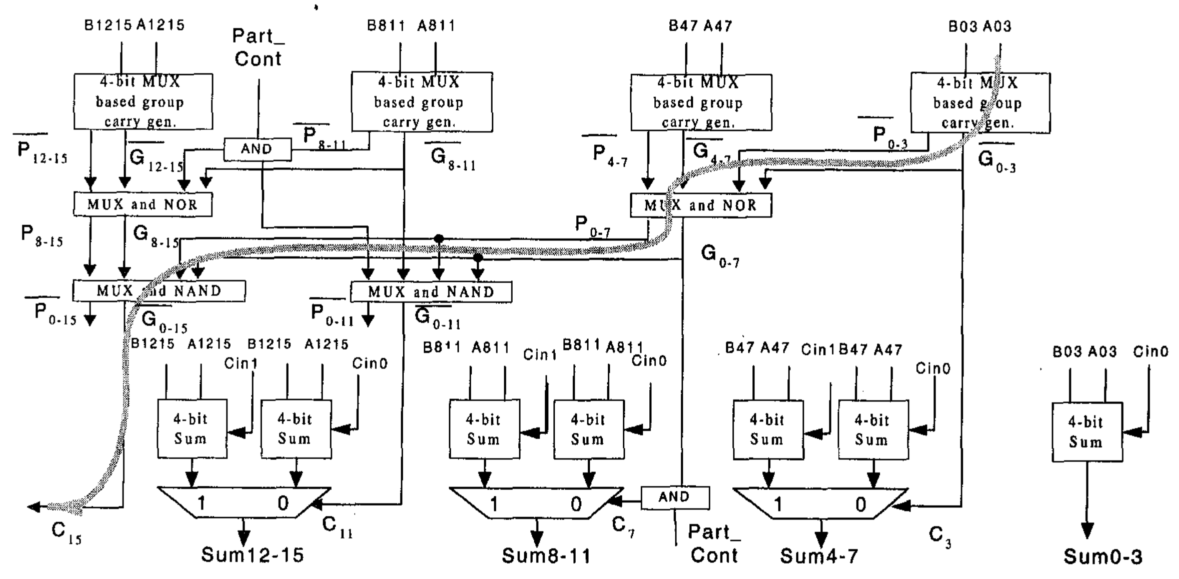

Fig. 2. 16-Bit block of the proposed adder.

according to Table-2. In order to get the highest speed using static CMOS standard cells, we restrict our design to 2-input NAND, NOR, XNOR and two-to-one multiplexers. The reason for using these gates is that, the delay of NAND, NOR gates in static CMOS is less than that of the non-inverting gates (AND, OR etc.), and in current technology multiplexers are realised using transmission gates, and inverters which offer the delay comparable to a single gate [6, 7].

Part1	Part0	Adder Operation
0	0	Byte
0	1	Half-word (16-bit)
1	0	Word (32-bit)
1	1	Double word (64-bit)

Table 2. Adder operation controlled by Part0 and Part1 controls.

The four bit generate and propagate logic is shown in Fig. 3. The circuit operation is simple, $g_{01} = a_1 \bullet b_1$, if $a_1 \otimes b_1 = 0$; and $g_{01} = a_0 \bullet b_0$ if $a_0 \otimes b_0 = 1$, taking advantage of the property that g_1=1 and p_1=1 can never occur. Once we have the two bit generate and propagate signals the 4-bit (and higher) group generate and propagate signals are calculated using one level of a two-one multiplexer and NAND/NOR gate respectively. Although we could have used the And-Or-Invert (AOI) cells instead of the two-one multiplexer, we have found that the delay of the AOI cells in our library is higher than the delay of the multiplexer, in addition it requires a buffer to drive more than two gates.

The final sum is calculated based on the carry signal. Since, the four bit carry select adders are not on the critical path, they could be designed using two sets of four bit ripple carry adders, with C_{in} = 0 for one set , and C_{in} =1 for the other set. However, we have found that it is possible to reduce the hardware by merging the two adders together. Fig. 4 shows the schematic diagram of the 4-bit merged carry select adder (CSA). The hardware of this merged CSA is approximately 40% smaller then the hardware required by two separate four bit ripple carry adders. In VLSI implementation, the 4-bit CSA is combined with the 4-bit carry generate circuit. Once this block is ready, the final goal in the realisation of high order adder is the cascading of all carry-sum blocks together using only multiplexers and NAND/NOR gates (Fig. 2 circled block) according to Equation (1).

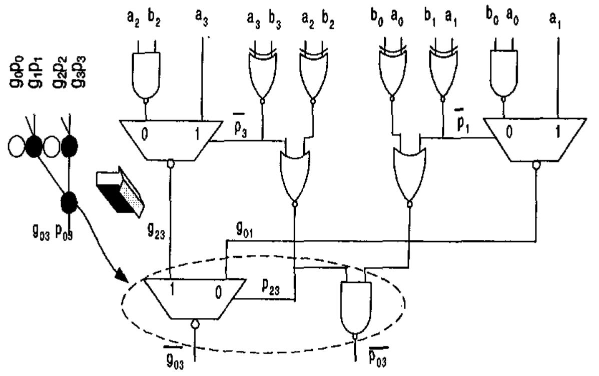

Fig. 3. Four bit carry and propagate generate circuit, using multiplexers.

4. Critical Path:

As one can see from Fig. 3, the four bit group carry (propagate) signal is generated in 1-XOR and 2-multiplexer (NOR) delays. After the first two levels, one multiplexer delay is required at each level of the adder. The critical path of adder is very predictable, starting from a_0,b_0 and going onto C_{0_35},.... C_{0_63}. The number of gates in this path are equivalent to 6 multiplexers (log 64) and 1-XOR gate. The correct sum is produced in 7-multiplexer delays including the delay of the CSA multiplexer.

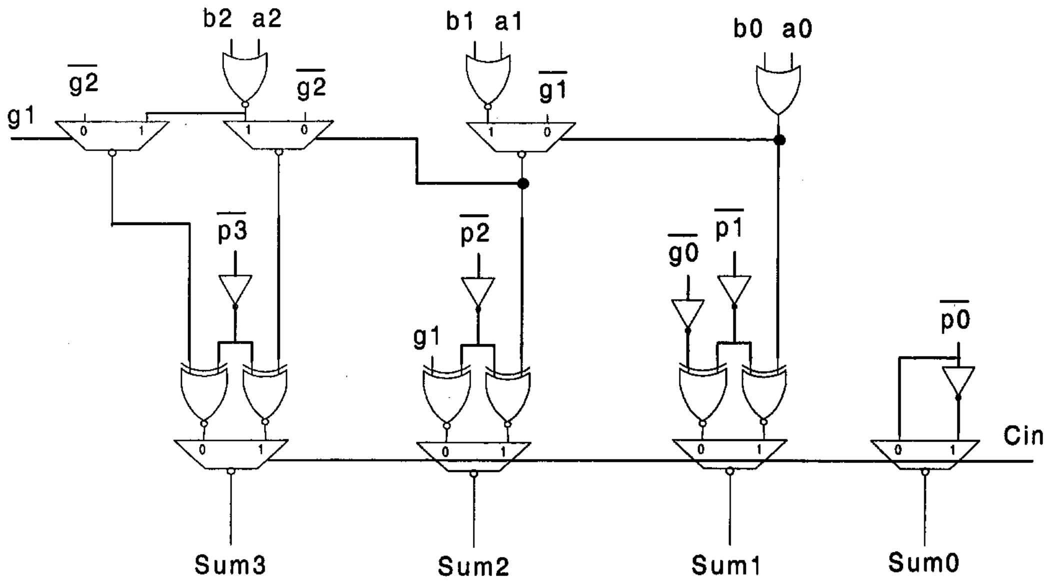

Fig. 4. Schematic diagram of the 4-bit fused carry select adder.

5. Simulation Results:

The partitioned adder has been implemented in Verilog, (using Cadence schematic capture tool) and synthesised in 0.25um standard cell library. Due to its modular nature, designing a high order adder using the blocks mentioned above is simple. We have synthesised adders of 8-bit, 16-bit, 32-bit and 64-bit separately. During these syntheses, a maximum fan-out of 4 has been used. The critical path delay for all of these adders at 2.5V and 25°C, using typical conditions of 0.25 um standard cell library are shown in Table-1. This table shows the critical path delay of 903 ps for a 64-bit partitioned adder, with minimum area and wiring.

Conclusion:

In this paper, the design and implementation of a fast 64-bit partitioned adder in standard CMOS has been presented. This adder has been design for Multi-media applications, but can also be used in general purpose, VLIW processors without performance degradation. The proposed adder can add two 64-bit operands, four 32-operands, eight 16-operands, or sixteen 8-bit operands, along with the generation of C_{out}, and OV conditions for all the cases. The adder has been implemented in Verilog, and synthesised in Synopsys using standard cell library. The critical path delay of the 64-bit adder as reported by Synopsys is 0.9 ns at typical conditions in 0.25 um technology. Further speed improvement can be achieved by using full custom design techniques.

Acknowledgments:

The authors are grateful to B. Nikolic for his assistance, and, SONY US Research Laboratories., for supporting this research.

References:

1. Weinberger and J. L. Smith, "A logic for high speed addition", National Bureau of Standards Circular 591, pp. 3-12, 1958.
2. Kai Hwang, "Computer Arithmetic: Principles Architecture and Design", John Wiley and Sons, 1979.
3. Brent R., "On the addition of binary numbers", IEEE Trans. Computers, C-19, 758 (1970).
4. Brent, R.P.; Kung, H.T., "A regular layout for parallel adders", IEEE Transactions on Computers, vol.C-31, (no.3), March 1982. p.260-4.
5. Dozza, D.; Gaddoni, M.; Baccarani, G. "A 3.5 ns, 64 bit, carry-lookahead adder", 1996 IEEE International Symposium on Circuits and Systems. Circuits and Systems, p.297-300 vol.2.
6. Oklobdzija, V.G., "Simple and efficient CMOS circuit for fast VLSI adder realisation". 1988 IEEE International Symposium on Circuits and Systems, p.235-8 vol.1. 3 vol. 2915 pp. 22.
7. Quach, N.T.; Flynn, M.J., "High-speed addition in CMOS" IEEE Trans. Computers, vol.41, Dec. 1992. p.1612-15.
8. Martin S. Schmookler, et. al., "A Low Power , High Speed Implementation of a PowerPC Microprocessor Vector Extension", to be presented at 14th Symposium on Computer arithmetic; 1999.