# Parallel Saturating Fractional Arithmetic Units

Navindra Yadav and Michael Schulte
EECS Department
Lehigh University
Bethlehem, PA 18015, USA

John Glossner
Advanced DSP Compilers and Architectures
Lucent Technologies, Inc.
Allentown, PA 18103, USA

## Abstract

*This paper describes the designs of a saturating adder, multiplier, single MAC unit, and dual MAC unit with one cycle latencies. The dual MAC unit can perform two saturating MAC operations in parallel and accumulate the results with saturation. Specialized saturation logic ensures that the output of the dual MAC unit is identical to the result of the operations performed serially with saturation after each multiplication and each addition*

## 1. Introduction

A characteristic of many DSP applications is that they are computationally intensive and execute millions of saturating arithmetic operations. For example, the Global System for Mobile communication (GSM) enhanced full rate speech coder and decoder each require over 128 million saturating arithmetic operations [2]. To be compliant with the GSM standard, the results produced must be identical to results produced when the operations are performed serially.

To achieve high performance, DSPs have multipliers, adders, and multiply and accumulate (MAC) units with one or two cycle latencies [4]. Typically, saturating arithmetic operations require two cycles, with saturation performed in the second cycle. More complicated operations, such as a MAC operation with saturation after both the multiplication and the addition, often require a greater number of cycles. The challenge is to design circuits that perform parallel saturating arithmetic operations in a single cycle and still produce the same results as when the operations are performed serially.

In this paper, designs of a fractional saturating adder, multiplier, single MAC unit, and dual MAC unit with single cycle latencies are presented. The dual MAC unit can perform two MAC operations in parallel and accumulate their results. The output of the dual MAC unit is identical to the result of the operations performed serially with saturation after each multiplication and each addition. The designs are coded with Verilog, and synthesized using the Synopsys Module Compiler [8] and a 0.25 micron CMOS standard cell library. Estimates for the critical path delay and area are presented for 16-bit arithmetic units.

## 2. Saturating Multiplier

Saturation occurs for two's complement fractional multiplication only when $-1.0 \times -1.0$ is computed. Since the true result of $1.0$ cannot be represented as a two's complement fractional number, the product is saturated to the largest positive fractional number. To detect when saturation needs to be performed, saturation detection logic (SDL) is used. If the input operands are

$$\begin{aligned} X &= x_{n-1}.x_{n-2}\dots x_1x_0 \\ Y &= y_{n-1}.y_{n-2}\dots y_1y_0 \end{aligned}$$

the logic equation for the SDL is

$$S = x_{n-1}\cdot\overline{x_{n-2}}\cdot\dots\cdot\overline{x_1}\cdot\overline{x_0}\cdot y_{n-1}\cdot\overline{y_{n-2}}\cdot\dots\cdot\overline{y_1}\cdot\overline{y_0}$$

When $X = 1.00\dots00$ and $Y = 1.00\dots00$, $S$ becomes 1 to indicate that saturation has occurred.

Figure 1 shows the multiplication matrix for a two's complement $n$-bit multiplier, where $P = X\cdot Y$ [1]. Except in the case of $-1.0\times-1.0$, $p_{2n-1}$ and $p_{2n-2}$ are identical and the product has the form

$$P = p_{2n-1}p_{2n-2}.p_{2n-3}\dots p_1p_0$$

To put the result back in fractional form, the product is shifted left one position and a zero is inserted into the least significant bit. When the multiplication of $-1.0\times-1.0$ occurs, the product that is generated is $P = 01.0\dots00$. Since this is not a valid two's complement fractional number, the result should be saturated to $0.11\dots11$.

Figure 2 shows the modifications that need to be made so that multiplication yields a saturated result. In this figure, $S$ is the saturation bit, which is output by the SDL. To produce a result in fractional form, the multiplication

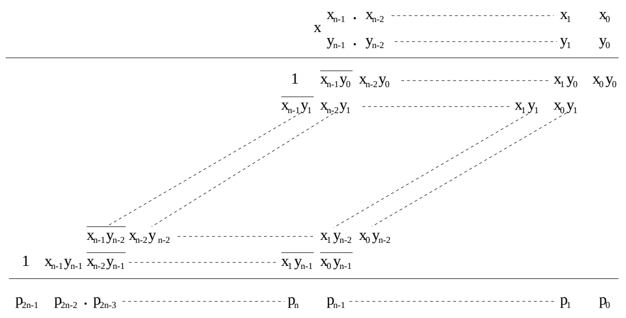

**Figure 1. Non-Saturating Multiplication.**

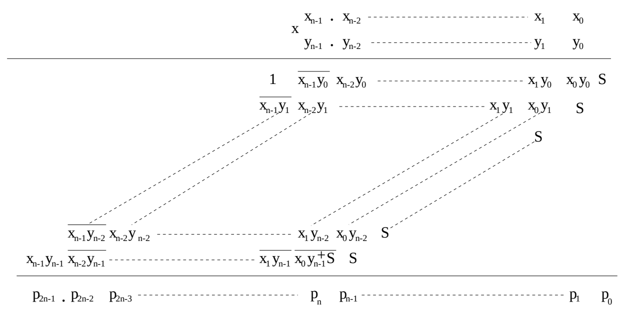

**Figure 2. Saturating Multiplication.**

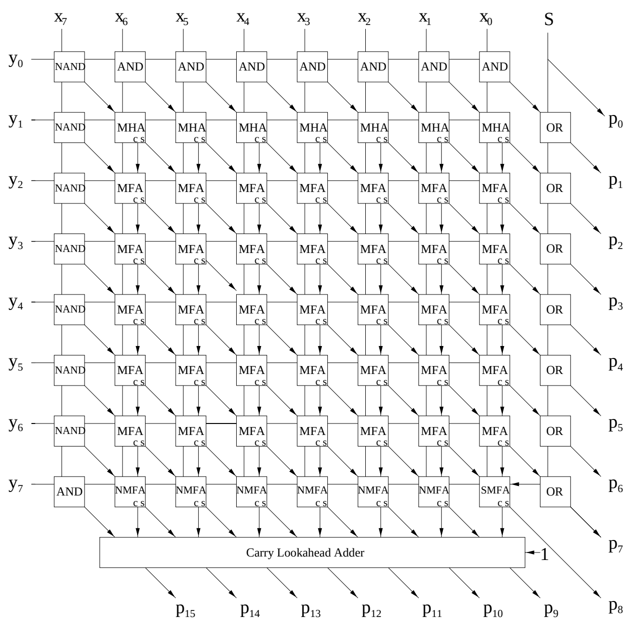

**Figure 3. 8-Bit Saturating Array Multiplier.**

matrix is shifted left one position and bits to the left of column $2n - 1$ are omitted. To handle saturation, $S$ is added to the $n$ least significant columns of the multiplication matrix, and the partial product $\overline{x_0 \cdot y_{n-1}}$ is replaced by $\overline{x_0 \cdot y_{n-1} + S}$. When $-1.0 \times -1.0$ is performed, $S = 1$, $\overline{x_0 \cdot y_{n-1} + S} = 0$, and the multiplier produces $0.11\ldots11$. In all other cases, $S = 0$ and the product does not saturate.

Figure 3 shows a block diagram of an 8-bit two's complement array multiplier [5] that has been modified to produce a saturated product. In this diagram, a modified half adder cell (MHA) consists of an AND gate and a half adder. Similarly, a modified full adder (MFA) consists of an AND gate and a full adder. In the leftmost column and bottom row of the array, NAND gates are used to produce partial products with either $x_{n-1}$ or $y_{n-1}$. Modified full adders that use NAND gates to generate the partial product are labeled with NMFA. To perform saturation, $p_0$ is set to the $S$ bit, and the $S$ bit is ORed with outputs from the rightmost column of the array to produce $p_1$ to $p_{n-1}$. The SMFA cell in the bottom-right corner of the array produces the partial product $\overline{x_0 \cdot y_{n-1} + S}$ and adds it with sum and carry bits from the previous row to produce $p_n$ and a carry bit. A carry look-ahead adder (CLA) [6] adds the sums and carries from the bottom row of the array to produce $p_{n+1}$ to $p_{2n-1}$.

Using the technique described previously, the saturation bit $S$ is computed early in the multiplication process. Since $\overline{x_0 \cdot y_{n-1} + S}$ is ready before the sum and carry bits that come into the SMFA and the computation of partial product bits $p_0$ to $p_{n-1}$ is not on the critical path, adding saturation to the multiplier does not increase its critical path delay. The only increase in area is due to the extra OR gates for saturation and changing one of the NMFAs to a SMFA. The technique presented here can easily be extended to other parallel multiplier implementations, such as tree multipliers and Booth-encoded multipliers [9], [7].

## 3. Saturating Adder

For two's complement addition, saturation only occurs if the two operands have the same sign and the sign of the result is different. A fast method for detecting this is to take the exclusive-or (XOR) of the carry-in and carry-out of the most significant bit position, which are denoted as $c_{n-1}$ and $c_n$, respectively. If the XOR of these two bits is one, the result saturates; otherwise, it does not.

It is also necessary to determine the value of the saturated result. If both operands are positive and the sum saturates, the saturated result is $0.11\ldots11$. If both operands are negative, and the sum saturates, the saturated result is $1.00\ldots00$. When performing $< S > = < A + B >$, the value used to saturate the result can be computed as

$$V = a_{n-1}.\overline{a_{n-1}}\ldots\overline{a_{n-1}}\,\overline{a_{n-1}}$$

where $a_{n-1}$ is the sign-bit of $A$ and $< K >$ indicates that $K$ is saturated.

The design of a saturating CLA is shown in Figure 4. The CLA [6], [3] is easily modified to produce a saturated result. If the output of $c_n \oplus c_{n-1}$ is 1, the sum needs to be saturated and $V$ is selected as the result; otherwise, the sum

$S$ from the CLA is selected. Compared to a non-saturating CLA, additional logic is needed to compute $V$, perform $c_n \oplus c_{n-1}$, and select between $V$ and $S$. Since $c_n \oplus c_{n-1}$ is available at the same time as $s_{n-1}$, the only delay added to the critical path is the delay of the multiplexor. This technique for saturating addition can also be applied to other adder implementations.

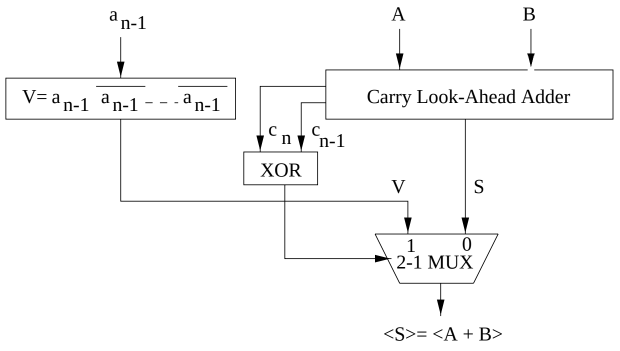

**Figure 4. Saturating Adder.**

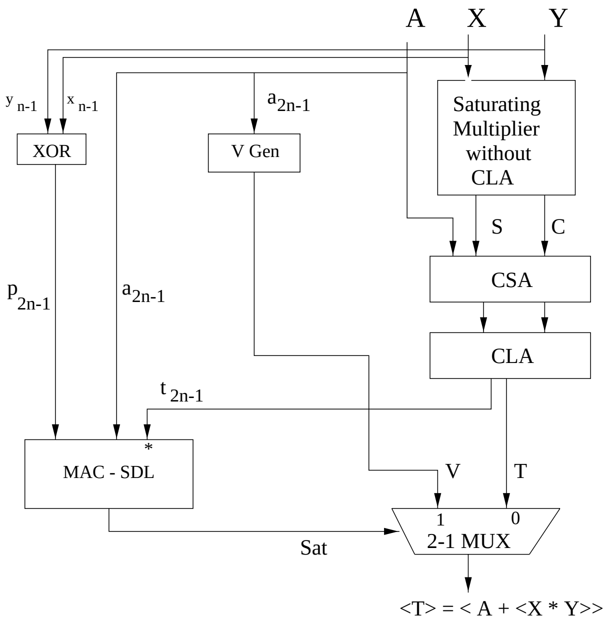

**Figure 5. Saturating MAC Unit.**

## 4. Saturating MAC Unit

Figure 5 shows the design of the saturating MAC unit, which computes $< T >=< A+ < X \cdot Y >>$. One approach for computing $< T >$ is to have $X$ and $Y$ be inputs to a saturating multiplier and then to add the saturated product $< X \cdot Y >$ to $A$ using a saturating adder. The main disadvantage of this approach is that there are two CLAs on the critical path.

To decrease the critical path delay, the saturating multiplier is modified so that it does not add the sum and carry bits generated in the last row of the array. This is accomplished by simply replacing the $(n-1)$-bit CLA, shown at the bottom of Figure 3, by a $(n-1)$-bit carry save adder (CSA). The modified saturating multiplier, produces a $2n$-bit sum vector $S$ and an $(n-1)$-bit carry vector $C$, such that $S + C =< X \cdot Y >$. The values $S$, $C$, and $A$ are then used as inputs to a $2n$-bit CSA, which combines $S$, $C$, and $A$ to produce two $2n$-bit vectors that are summed using a CLA to yield $T$.

MAC-saturation detection logic (MAC-SDL) is used to detect if the sum of the three vectors A, S and C saturates. The MAC-SDL uses the following logic equation to detect saturation.

$$Sat = a_{2n-1} \cdot p_{2n-1} \cdot \overline{t_{2n-1}} \; + \; \overline{a_{2n-1}} \cdot \overline{p_{2n-1}} \cdot t_{2n-1}$$

where $a_{2n-1}$ is the sign bit of $A$, $p_{2n-1} \; = \; x_{n-1} \oplus y_{n-1}$ is the sign bit of $< X \cdot Y >$, and $t_{2n-1}$ is the sign bit of $T$. Saturation occurs only when the signs of $A$ and $< X \cdot Y >$ are equal to each other, but not equal to the sign of $T$. If $Sat$ is one, the output of the VGen unit is selected as the final result; otherwise, $T$ is selected. The VGen unit generates the correct saturated value

$$V = a_{2n-1}.\overline{a_{2n-1}} \ldots \overline{a_{2n-1}}\;\overline{a_{2n-1}}$$

## 5. Saturating Dual MAC Unit

Figure 6 shows the block diagram of a saturating dual MAC unit. This unit performs two MAC operations in parallel, with saturation after each addition and each multiplication. The dual MAC uses two saturating MAC units plus additional hardware, so that it can combine the results of the two MAC operations and produce a result that is equivalent to the result that would be obtained if the operations had been performed serially. This computation can be expressed as

$$< T >=<< A_1 + < X_1 \cdot Y_1 >> + < X_2 \cdot Y_2 >>$$

where $A_1$ is the accumulator value, $X_1$ and $Y_1$ are operands for the first multiplication, and $X_2$ and $Y_2$ are operands for the second multiplication. The values $S_1, C_1$ and $S_2, C_2$ represent the saturated sum and carry vectors from the first and second saturated multipliers, respectively, where

$$\begin{aligned} S_1 + C_1 &= < X_1 \cdot Y_1 > \\ S_2 + C_2 &= < X_2 \cdot Y_2 > \end{aligned}$$

The final result is computed as

$$< T > \;\; = \;\; << A_1 + S_1 + C_1 > + S_2 + C_2 >$$

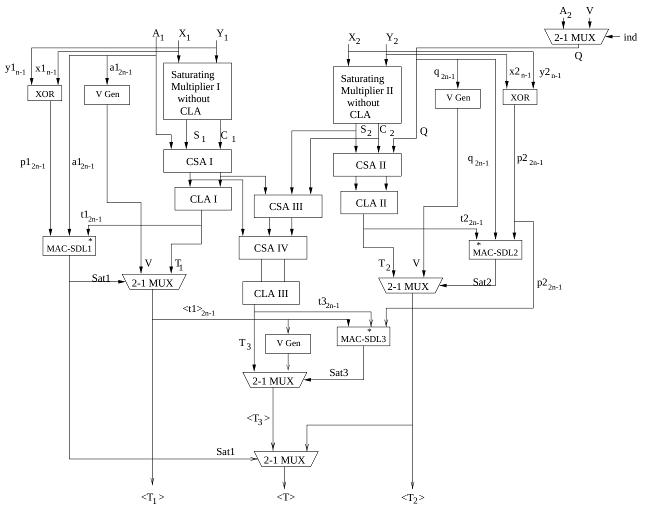

**Figure 6. Saturating Dual MAC Unit.**

To obtain the same result as when two MAC operations are executed serially, three summations are performed in parallel.

1. $< T_1 >=< A_1 + S_1 + C_1 >$
2. $< T_2 >=< Q + S_2 + C_2 >$
3. $< T_3 >=< A_1 + S_1 + C_1 + S_2 + C_2 >$

where $Q$ is $0.11 \ldots 11$ when $A_1$ is positive, and $1.00 \ldots 00$ when $A_1$ is negative. If $T_1$ saturates, $< T >=< T_2 >$; otherwise, $< T >=< T_3 >$. By performing the three summations in parallel and then selecting the proper result, the only increase in delay compared to a single saturating MAC is one CSA and one 2-to-1 multiplexor.

The dual MAC unit can also perform two independent saturating MAC operations in parallel, where the results produced correspond to $< T_1 >=< A_1 + < X_1 \cdot Y_1 >>$ and $< T_2 >=< A_2 + < X_2 \cdot Y_2 >>$. This is accomplished by setting the $ind$ control bit to 1, so that $Q = A_2$. With minor modifications, the dual MAC unit can also perform parallel saturating multiply and subtract operations, as well as non-saturating arithmetic operations.

## 6. Results and Conclusions

16-bit designs for saturating and non-saturating arithmetic units were implemented using Verilog. The Synopsys Module Compiler [8] and a 0.25 micron CMOS standard cell library were used to synthesize each design. Estimates of the area and critical path delay are shown in Table 1. Areas are reported in grid units and the critical path delay is given in nanoseconds for a supply voltage of 2.5 Volts. One unexpected result in Table 1 is the non-saturating MAC unit requires less area than the saturating MAC unit. This may have resulted from the synthesis tool's ability to exchange area for delay. The area and delay estimates shown for the non-saturating dual MAC unit correspond to two non-saturating MAC units that can operate in parallel, but cannot combine their results in the same cycle.

| | Saturation | | No Saturation | |
|---|---|---|---|---|
| Unit | Delay | Area | Delay | Area |
| Adder | 1.1 | 2373 | 1.0 | 2323 |
| Multiplier | 6.6 | 16695 | 6.6 | 16608 |
| MAC | 7.5 | 18567 | 7.2 | 18853 |
| Dual MAC | 8.1 | 42982 | 7.2 | 37706 |

**Table 1. Synthesis Results for 16-Bit Units**

On many DSPs, it takes two or more cycles to perform saturating arithmetic operations. In comparison, the arithmetic units presented in this paper perform saturating arithmetic operation in a single cycle, yet require only a small increase in area and delay. For many DSP applications, which require millions of saturating arithmetic operations, these units can provide significant performance improvements.

## Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant No. MIP-9703421, and by a grant from Lucent Technologies and the Pennsylvania Infrastructure Technology Alliance under Project No. AMD-003.

## References

[1] K. Bickerstaff, M. J. Schulte, and E. E. Swartzlander, Jr. Parallel Reduced Area Multipliers. *Journal of VLSI Signal Processing*, 9:181–192, April 1995.

[2] European Telecommunication Standards Institute. *Digital Cellular Telecommunications System: ANSI-C Code for the GSM Enhanced Full Rate (EFR) Speech Code*, 1997.

[3] I. Koren. *Computer Arithmetic and Algorithms*. Prentice Hall, 1993.

[4] P. Lapsley. *DSP Processor Fundamentals: Architectures and Features*. IEEE Press, 1997.

[5] J. C. Majithia and R. Kita. An Iterative Array for Multiplication of Signed Binary Numbers. *IEEE Transactions on Computer*, C-20:28–33, February 1971.

[6] T. F. Ngai, M. J. Irwin, and S. Rawat. Regular, Area-Time Efficient Carry-Lookahead Adders. *Journal of Parallel and Distributed Computing*, 3:92–105, 1986.

[7] H. Sam and A. Gupta. A Generalized Multibit Recoding of Two's Complement Binary Numbers and Its Proof with Application in Multiplier Implementations. *IEEE Transactions on Computers*, 39(8):1006–1015, 1990.

[8] Synopsys. *Synopsys Module Compiler Features*. Synopsis, Inc, 1998.

[9] C. S. Wallace. Suggestion for a Fast Multiplier. *IEEE Transactions on Electronic Computers*, EC-13:14–17, 1964.