High-Speed, Energy-Efficient 2-Cycle Multiply-Accumulate Architecture

Tung Thanh Hoang, Magnus Själander, and Per Larsson-Edefors
VLSI Research Group, Department of CSE, Chalmers University of Technology
SE-412 96 Gothenburg, Sweden. Email: {hoangt,hms,perla}@chalmers.se

***Abstract*—**
We propose a high-speed and energy-efficient 2-cycle Multiply-Accumulate (MAC) architecture. Our architecture is based on two's complement representation, it uses guarding bits to efficiently support longer MAC loops, and it includes output saturation. By performing carry propagation only in the second stage of the MAC pipeline, multiplication and accumulation have similar delays. But in contrast to previous MAC architectures that propose to only use one carry-propagation stage, our architecture requires no extra cycles to produce the final result. Instead it correctly produces the sum of the accumulated value and the product in each cycle. Our place-and-route evaluation shows that the proposed architecture, averaged across several operand sizes, offers a 33% improvement in speed and a 37% reduction of energy over a conventional 2-cycle MAC architecture.

I. Introduction

The Multiply-Accumulate (MAC) unit is a common accelerator that is extensively used in microprocessors and digital signal processors for data-intensive applications. For example, many filters, orthogonal frequency-division multiplexing, channel estimation etc. require FIR or FFT/IFFT computations that can be efficiently accelerated by dedicated MAC units.

In general, MAC architectures are based on a multiplier, whose output is added to the previous MAC output result by an accumulate adder (Fig. 1). For the purpose of high speed, the multiplier is typically comprising a chain of a partial-product unit (the *PP* unit) and a carry-propagate adder (the *final adder*).

To increase MAC performance, we can cut down the critical path delay by inserting an extra pipeline register, either inside the PP unit or between the PP unit and the final adder [1], leading to a 3-cycle MAC architecture (Fig. 1). However, pipelining comes at the cost of increased latency and an overhead in terms of energy and area[1].

Since the multiplier is much more complex than the accumulate adder, many design techniques have focused on reducing multiplier delay, either in the PP unit or in the final adder. Inside the PP unit, we may target the Partial-Product Generation (PPG) by using, e.g. the modified-Booth algorithm [2] or some of its successors [3]. As for the Partial-Product Reduction Tree (PPRT) of the PP unit, high-speed compressors[2] [4] and speed-optimized structures [5] have been proposed. In [6], Ladner *et al.* proposed a sparse-tree carry look-ahead adder for fast addition of the PP outputs, while Liu *et al.* introduced a hybrid adder to reduce delay over the design that assumed equal arrival time on all adder inputs [7].

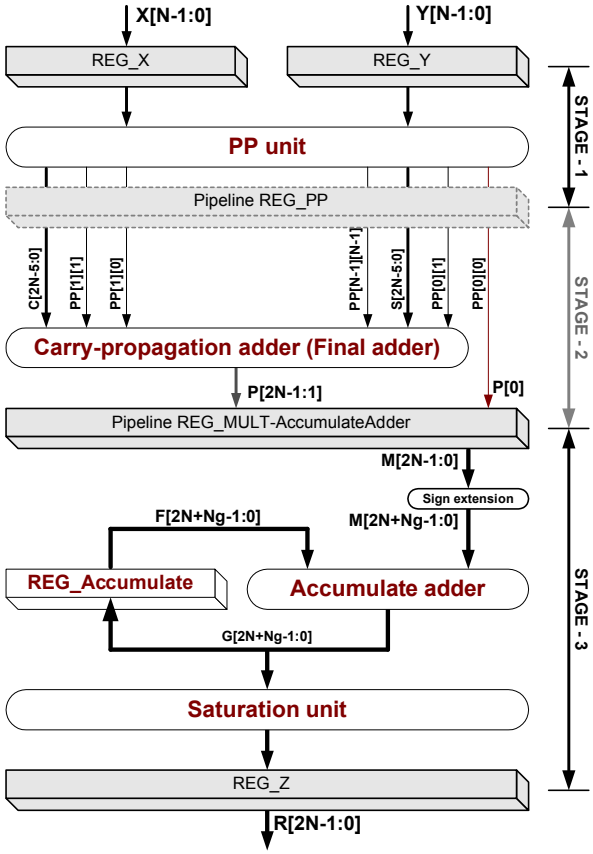

Fig. 1: Block diagram of a general MAC architecture. Here, the register between the PPRT and the final adder is removed/*inserted* to obtain the 2/*3*-cycle MAC architecture.

By feeding back the multiplier output bits to the top of the PPRT unit, the adder that traditionally handles accumulation can be removed [8], [9], [10]. This way the accumulation is handled by the final adder of the multiplier. The problem with this approach is that it is only applicable to 1-cycle MACs. If a pipeline register would be inserted, the MAC output will no longer produce the correct result in each cycle. In fact, to get the final MAC result, we would have to add an extra, empty cycle after the final MAC cycle of a loop.

[1]If very low energy is the primary target, 1-cycle MAC operations could be considered. However, the long delay of a 1-cycle MAC unit is a very limiting factor in most applications.

[2]The simplest 3:2 compressor being the full adder.

We propose to combine the following two, somewhat conflicting features:

- The accumulation should take place in the second stage of a 2-cycle MAC unit.
- The carry should be propagated only once in a MAC pipeline, thus, in the second stage.

As Sec. II will show, we can obtain a number of advantages with the new approach:

- Carry propagation only takes place in the second stage, which means that the multiplier's final adder is eliminated, leading to higher speed and lower energy.
- Since accumulation takes place inside the second stage
  - A pipeline register located before the accumulation stage has no impact on functionality. Regardless of pipelining, our MAC unit will produce the correct result in each cycle, and no extra cycles need to be added at the end of the loops.
  - Interconnects are localized, which simplifies routing, decreases delay, and reduces energy dissipation.
- Because of the above advantages, we can support several guarding bits, making longer loops feasible without any overflow problems. The use of guarding bits in an approach, where the accumulated value is fed back to the PPRT's input, would most certainly have a negative impact on hardware complexity.

The remainder of this paper is organized as follows: Sec. II describes our proposed MAC architecture and contrasts it to a conventional architecture. Next, Sec. III provides an evaluation with respect to performance, power/energy, and area. Finally, we conclude this paper in Sec. IV.

II. Proposed MAC Architecture

The proposed MAC architecture is shown in Fig. 2. The final adder has been removed, and a carry-save adder has been inserted after the pipeline registers. The maximum delay of the carry-save adder is only that of a single full adder, which means that the MAC's critical path delay still depends on the PP unit.

Fig. 3 shows the conventional way of performing a multiply-accumulate operation, assuming that the Baugh-Wooley algorithm [11][3] is used. First we compute the multiplication of the two inputs. Then the result from this multiplication is sign extended, so that it has the same size as the accumulate adder. Finally, the sign extended product is added to the stored accumulated value. The disadvantage of this conventional scheme is that $P[2N-1]$ in Fig. 3 needs to be computed and used as sign extension for the second, accumulative addition.

In the carry-save adder of our scheme we do not need to sign extend the multiplier output. We instead use a row of '1' to perform the sign extension, as shown in the circles in Fig. 4.

[3]A modified-Booth scheme such as that in [3] can indeed be used, but it offers no gain in terms of timing, but rather incurs a significant power dissipation overhead [12].

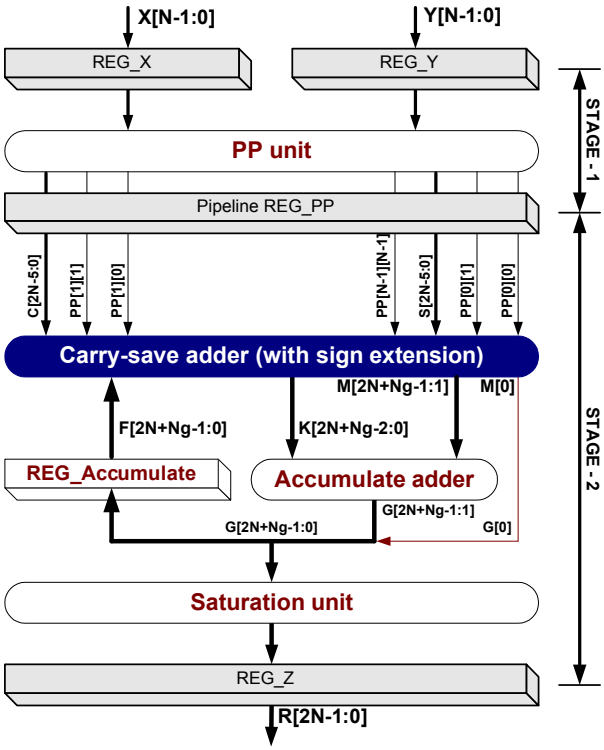

Fig. 2: Block diagram of the proposed MAC.

Our MAC architecture offers a number of advantages, in terms of latency, speed, area, and energy:

- If we compare to a 2-cycle MAC (Fig. 1), the proposed MAC architecture does not need the final adder.
- If we compare to a 3-cycle MAC (Fig. 1), our MAC architecture also allows us to not only remove the final adder but also to remove one pipeline register level — and the corresponding clock energy—without degrading speed.
- Because our MAC architecture is smaller than the conventional MAC architectures, the smaller footprint will lead to shorter interconnects.

III. Evaluation

A. Evaluated architectures

We consider three architectures that share the same structure for the PP unit, the final adder and the accumulate adder:

- **MAC-2C** : This conventional 2-cycle MAC has a critical path that goes through the PP unit and the final adder.
- **MAC-3C** : This conventional 3-cycle MAC has a critical path that is located inside the PP unit.
- **MAC-NEW**: Our proposed 2-cycle MAC exploits the fact that the delay of the accumulate adder is shorter than the delay of the PP unit, by at least an amount corresponding to the delay of a full-adder cell.

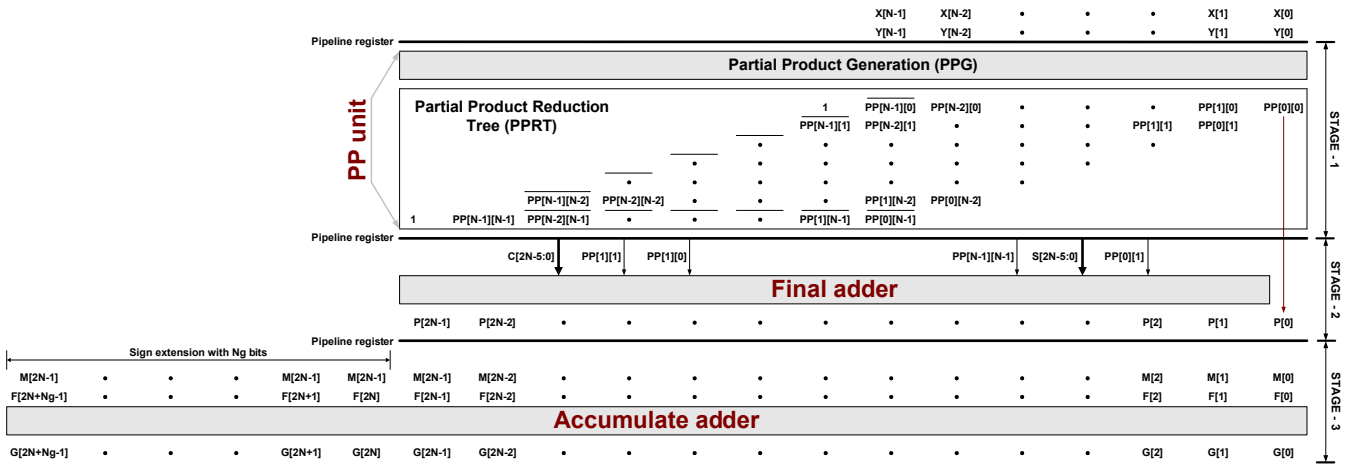

Fig. 3: A MAC operation using inputs X and Y, assuming the 3-cycle MAC of Fig. 1. The MAC operation starts with the generation (assuming the Baugh-Wooley algorithm) and reduction of partial products. Carry propagation of the sums and carries, produced by the PP unit, is handled by the final adder. Finally, the accumulate adder adds the pipelined products of the final adder (i.e., M) to the accumulated result (i.e., F), producing the new MAC result G.

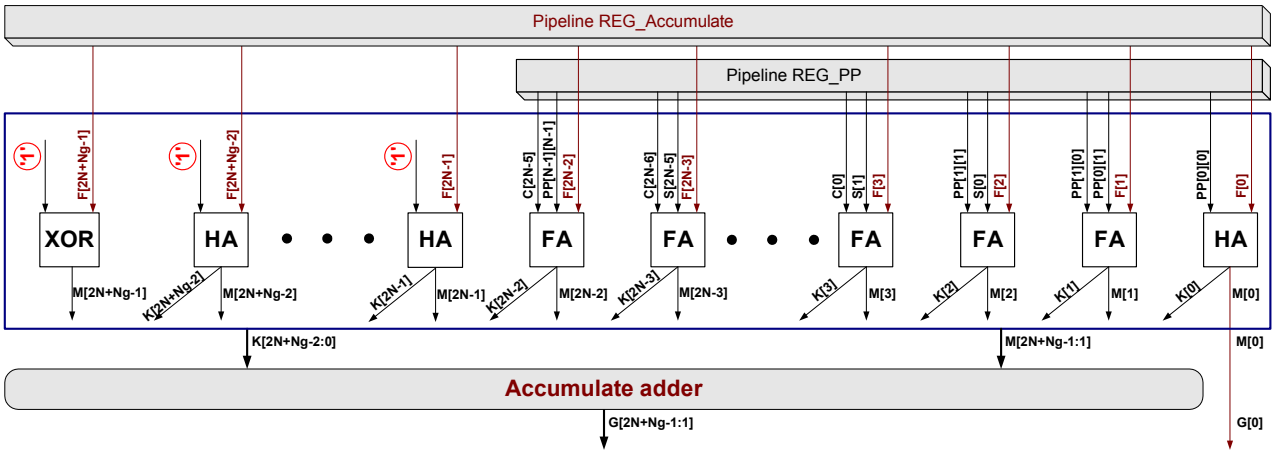

Fig. 4: Logic structure of the carry-save adder.

Concerning the comparison of the MAC critical path delay, we notice that MAC-2C and MAC-3C represent architectures that put an upper and a lower bound, respectively, on critical path delay.

B. Evaluation methodology

All PP units use Baugh-Wooley for partial-production generation and the HPM approach for partial-production reduction tree [13]. The accumulate adder is of Sklansky type [6] and has an extension of eight guarding bits. Finally, the final adder in [14] is used to support fast addition of the PP outputs in the case of MAC-2C and MAC-3C.

The VHDL codes were developed for MAC-2C, MAC-3C and MAC-NEW, using several different sizes of the input data. We synthesized the VHDL codes using Synopsys Design Compiler, using a commercial 65-nm standard-V_T library, at 1.1 V and with the worst-case corner. To avoid biasing the evaluation, we use a bottom-up synthesis method, meaning that the PP unit, the final adder and the accumulate adder are synthesized individually. Since the MAC-NEW turns out to have the smallest area, probably this design is at a disadvantage in our evaluation.

All MAC netlist files were subsequently verified by using logic simulation, after which they were taken through Place and route in Cadence SoC Encounter. Delay of individual units of each architecture was extracted using Synopsys PrimeTime. Power dissipation was estimated through a value change dump (VCD) analysis involving 20,000 random test vectors using RC extracted data from SoC Encounter.

C. Evaluation results and discussion

Table I presents the detailed results of our evaluation. The average values at the bottom rows of the tables are also presented as normalized values, using MAC-2C as reference.

Since the critical path is through the PP unit for all three designs and our proposed architecture uses pipeline registers at the bottom of the PP unit, MAC-NEW obviously can operate at the same speed as MAC-3C, while its performance on average—for various operand sizes— is 33% faster than MAC-2C. As far as power dissipation is concerned, because the final

TABLE I: Evaluation results of three architectures for 16, 32, 48, and 64 bits in operand size.

Operand size	Architecture	Delay of units (ps)			Critical path delay (ps)	Power (mW)	Energy (pJ)**	Cell area (μm^2)
		PP	Final adder	Final MAC stage*				
16	MAC-2C	1317	597	1039	1914	6.9	13.2	12937
	MAC-3C	1312	621	1039	1312	10.3	13.5	13711
	MAC-NEW	1312	*NA*	1247	1312	8.2	10.7	12014
32	MAC-2C	1671	701	1110	2372	17.1	46.7	42216
	MAC-3C	1664	711	1110	1664	21.2	35.2	43545
	MAC-NEW	1664	*NA*	1318	1664	19.8	32.9	39357
48	MAC-2C	1976	884	1089	2860	30.4	87.0	84894
	MAC-3C	1931	925	1089	1931	38.3	74.0	86898
	MAC-NEW	1931	*NA*	1297	1931	32.3	62.3	82696
64	MAC-2C	2090	824	1242	2914	54.6	159.1	149121
	MAC-3C	2082	848	1242	2082	62.1	129.3	152179
	MAC-NEW	2082	*NA*	1450	2082	41.6	86.6	142942
Average	MAC-2C				2605 (100%)	27.3 (100%)	71.0 (100%)	72292 (100%)
	MAC-3C				1747 (67%)	33.0 (121%)	57.6 (81%)	74083 (102%)
	MAC-NEW				1747 (67%)	25.5 (93%)	44.5 (63%)	69252 (96%)

(*) Here, the delay of the final MAC stage is the cumulative delay of the extended accumulate adder and the saturation unit for MAC-2C and MAC-3C. For MAC-NEW we here include the 239-ps delay of the carry-save adder, which is equal to the delay of a single full-adder.
(**) Energy dissipation is here defined per cycle, i.e. as the product of critical path delay and power dissipation.

adder is replaced by the simple carry-save adder, MAC-3C on average dissipates 29% more power than MAC-NEW for the same operating frequency and timing constraint.

We use energy dissipation to simultaneously capture power and performance for the considered architectures: averaged across the four operand sizes, MAC-NEW dissipates 37% and 23% less energy than MAC-2C and MAC-3C, respectively.

In terms of cell area, the proposed MAC architecture on average results in 4% and 7% smaller footprint than MAC-2C and MAC-3C, respectively.

Finally, we notice that the delay ratio between the PP unit and the accumulate adder increases for larger operand sizes (the ratio being in the range of 1.05–1.44), so extra guarding bits could be accommodated for large operands to support longer iteration computations without any performance degradation.

IV. Conclusion

We have described a novel high-speed and energy-efficient 2-cycle Multiply-Accumulate (MAC) architecture. We proposed removing the final adder of the multiplier, and instead use a simple carry-save adder inside the accumulate stage. The evaluation shows that our 2-cycle architecture is both faster and more efficient in terms of area and energy than a conventional 2-cycle MAC unit. Also, the evaluation shows that our architecture, while only requiring two cycles for completing the MAC computation, still performs the MAC operation at the same top operating frequency as a 3-cycle MAC unit, at a lower energy dissipation.

V. Acknowledgment

The authors wish to thank members of VLSI research group at Chalmers University of Technology who have reviewed and commented to this work

References

[1] S. Kim, C. H. Ziesler, and M. C. Papaefthymiou, "Fine-grain real-time reconfigurable pipelining," *IBM J. Research and Development*, vol. 47, no. 5-6, pp. 599–609, September 2003.

[2] O. L. MacSorley, "High-speed arithmetic in binary computers," *Proc. of IRE*, vol. 49, January 1961.

[3] W.-C. Yeh and C.-W. Jen, "High-speed Booth encoded parallel multiplier design," *IEEE Trans. on Computers*, vol. 49, no. 7, pp. 692–701, July 2000.

[4] M. R. Santoro and M. A. Horowitz, "SPIM: A pipeline 64x64 bit iterative multiplier," *IEEE J. Solid-State Circuits(JSSC)*, vol. 2, no. 1, pp. 487–493, April 1989.

[5] V. G. Oklobdzija, D. Villeger, and S. S. Liu, "A method for speed optimized partial product reduction and generation of fast parallel multipliers using an algorithmic approach," *IEEE Trans. on Computers*, vol. 45, no. 3, pp. 294–306, March 1996.

[6] R. Ladner and M. Fisher, "Parallel prefix computation," *J. of the Association for Computer Machinery*, vol. 27, no. 4, pp. 831–838, October 1980.

[7] J. Liu, S. Zhou, H. Zhu, and C.-K. Cheng, "An algorithmic approach for generic parallel adders," in *Proc. of IEEE International Conference on Computer Aided Design (ICCAD)*, December 2003, pp. 734–740.

[8] P. F. Stelling and V. G. Oklobdzija, "Implementing multiply-accumulate operation in multiplication time," in *Proc. of 13th International Symposium on Computer Arithmetic (ARITH)*, July 1997, pp. 99–106.

[9] J. Großschädl and G.-A. Kamendje, "A single-cycle (32x32+32+64)-bit multiply/accumulate unit for digital signal processing and public-key cryptography," in *Proc. of IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, December 2008, pp. 739–742.

[10] A. Abdelgawad and M. Bayoumi, "High speed and area-efficient Multiply Accumulate (MAC) unit for digital signal prossing applications," in *Proc. of IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2007, pp. 3199–3202.

[11] C. R. Baugh and B. A. Wooley, "A two's complement parallel array multiplication algorithm," *IEEE Trans. on Computers*, vol. 22, pp. 1045–1047, December 1973.

[12] M. Själander and P. Larsson-Edefors, "High-speed and low-power multipliers using the Baugh-Wooley algorithm and HPM reduction tree," in *Proc. of IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, August 2008, pp. 33–36.

[13] H. Eriksson, P. Larsson-Edefors, M. Sheeran, M. Själander, D. Johansson, and M. Schölin, "Multiplier reduction tree with logarithmic logic depth and regular connectivity," in *Proc. of IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2006, pp. 4–8.

[14] S. K. Mathew, M. A. Anders, B. Bloechel, T. Nguyen, R. K. Krishnamurthy, and S. Borkar, "A 4-GHz 300-mW 64-bit integer execution ALU with dual supply voltages in 90-nm CMOS," *IEEE J. Solid-State Circuits (JSSC)*, vol. 40, no. 1, pp. 44–51, January 2005.