An Efficient Twin-Precision Multiplier

Magnus Själander, Henrik Eriksson, and Per Larsson-Edefors
VLSI Research Group, Department of Computer Engineering
Chalmers University of Technology, SE-412 96 Göteborg, Sweden

Abstract

We present a twin-precision multiplier that in normal operation mode efficiently performs N-b multiplications. For applications where the demand on precision is relaxed, the multiplier can perform N/2-b multiplications while expending only a fraction of the energy of a conventional N-b multiplier. For applications with high demands on throughput, the multiplier is capable of performing two independent N/2-b multiplications in parallel. A comparison between two signed 16-b multipliers, where both perform single 8-b multiplications, shows that the twin-precision multiplier has 72% lower power dissipation and 15% higher speed than the conventional one, while only requiring 8% more transistors.

1. Introduction

Recent development at the micro architecture level shows that there is an increasing interest in datapath components that are capable of performing computations with variable operand size, e.g. adders capable of doing both N and N/2-b additions [1]. By using only a part of the datapath component for computation, it has been demonstrated [2] that reductions in the total power dissipation can be effected. Datapath components that can perform both one N, one single N/2, or two N/2-b operations give the designer the opportunity to design a system which can adapt to changing modes, such as low-power, high-throughput, or high-precision operation. Such a datapath component could be used for dynamic power reduction in the same way as described by Abddollahi *et al*. [2]; by using the same kind of logic for detecting if the effective bit rate is within N/2-b precision, it is possible to control at what precision the datapath component should be operating. This versatile type of datapath component is also suitable for systems in which several applications, having quite different requirements on precision and/or throughput, are executed [3]. Furthermore, such a datapath component could prove useful in processors that can support several instruction sets. In a processor that combines x86-32 and x86-64, a flexible datapath could be used for 64-b operations as well as for Single Instruction Multiple Data (SIMD) instructions, where two 32-b operations are performed in parallel.

It has been shown [4] that it is relatively straightforward to partition an array multiplier, so as to obtain a multiplier that can perform multiplications with varying operand size[1]. In comparison to tree multipliers, however, an array multiplier is slow and power hungry which makes it a poor design choice when a fast and efficient multiplier is needed [5]. It was claimed, but not substantiated, that the power-reduction techniques used for array multipliers [2] can be applied also to tree multipliers. It is certainly not straightforward to transfer the proposed technique to tree multipliers. Mokrian *et al.* presented a reconfigurable multiplier, which is constituted by several smaller tree multipliers [6]. However, the recursive nature of this multiplier is, due to an addition of reduction stage(s), likely to have a large impact on the delay for the N-b multiplication, compared to the multiplier proposed in this paper.

In the following we explore the possibility of combining N and N/2-b multiplications in the same N-b tree multiplier: we call this a twin-precision multiplier. The key challenges in designing a twin-precision multiplier are to limit the impact of flexibility on power dissipation, delay, and area. The proposed twin-precision multiplier efficiently performs either one N-b multiplication, one single N/2-b multiplication, or two N/2-b multiplications in parallel.

2. Design Exploration

Based on a simple representation of an array multiplier, Figure 1, it is obvious that if the partial product bits not being used in a low-precision multiplication are set to zero, the array multiplier will produce the correct result without the need of any additional logic. The 2-input AND gates corre-

1 This was done by gating parts of the array of carry-save adders and by using multiplexers to read out the data from a low-precision multiplication.

Proceedings of the IEEE International Conference on Computer Design (ICCD'04)
1063-6404/04 $ 20.00 IEEE

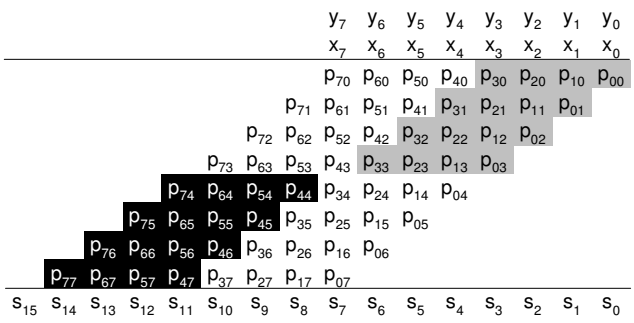

Figure 1: Partial product representation of a 4-b multiplication in an 8-b multiplier.

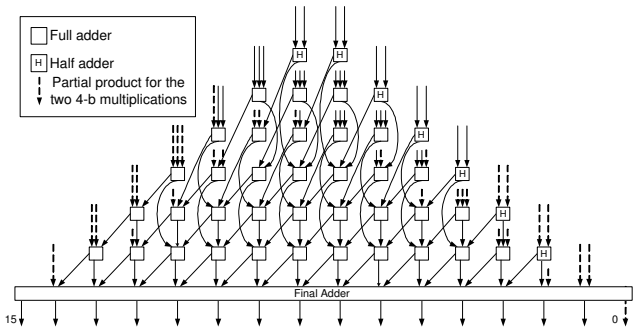

Figure 2: Partitioned tree of an 8-b multiplier.

sponding to the partial product bits that are not being used in the low-precision multiplication can be replaced by 3-input AND gates to force those bits to zero[2].

When doing an *N*/2-b multiplication within an *N*-b multiplier only one quarter of the logic is being used, as seen in grey in Figure 1. This makes it possible to use the multiplier for two parallel and independent *N*/2-b multiplications. We can partition the partial product bits of the *N*-b multiplier, such that an *N*/2-b multiplication can be performed in the Least Significant Part (LSP) of the multiplier in parallel with another *N*/2-b multiplication in the Most Significant Part (MSP), without using any additional logic in the partial product reduction tree, as seen in grey and black, respectively, in Figure 1. To be able to switch between *N*, *N*/2, or two *N*/2-b multiplications, the 2-input AND gates used to create the partial products need to be replaced with 3-input AND gates and two control signals for selecting the operating mode of the multiplier need to be introduced.

2.1. Tree Multiplier

Until now we have implicitly used the array multiplier to demonstrate the twin-precision feature. The array multiplier is, however, slow and power dissipating in comparison to a logarithmic tree multiplier. The implementation of the twin-precision feature in an *N*-b tree multiplier is similar to that of the array multiplier; all that is needed is to set the partial products bits not being used to zero and to partition the partial products bits of the two multiplications into the respective LSP and MSP of the tree. To reduce the critical path for the *N*/2-b multiplications the partial products bits used during the computation are moved as far down the tree as possible, Figure 2. In this paper we use a tree multiplier with regular connectivity [7].

To further reduce the critical path of the *N*/2-b multiplications it is possible to move the partial products even further down the tree by adding multiplexers on lower levels[3]. This makes it possible to select either the carry and sum from higher levels, when doing the *N*-b multiplication, *or* the partial products bits, when doing the *N*/2-b multiplication. This introduces multiplexers in the critical path of the *N*-b multiplier, which significantly increases the delay of the *N*-b multiplication. Thus, this alternative has not been considered here, since our goal is to find a good design tradeoff between the delay of *N*-b multiplications and *N*/2-b multiplications, respectively.

2.2. Signed Multiplication According to Baugh-Wooley

We used the Baugh-Wooley algorithm [8] to investigate the impact of the twin-precision feature on delay and power of a signed tree multiplier[4]. Here, signed multiplication is performed by first inverting all partial product bits that are results of the most significant bit (MSB) of exactly one of the operands, Figure 3. Second, for each executed multiplication, a logical one (framed) is added to column *N* (column 0 is to the far right in Figure 3) and, third, the MSB of the product is inverted. This is directly mapped onto the tree multiplier as shown in Figure 4.

To be able to generate the inverted partial product bits, we chose to replace the AND gates corresponding to the inverted bits with NAND gates followed by XOR gates. The option to either invert or not invert the signal from the NAND gates makes it possible to switch between signed and unsigned multiplication

The inversion of the MSB of the product is also done with an XOR gate. The insertion of the logical one to column *N* of the multiplication is straightforward for the *N*-b and the *N*/2-b multiplication in the LSP by changing the half adder of that column to a full adder and adding the log-

2 When performing only one *N*/2-b multiplication it is possible to set the most significant bits of the operands to zero instead.

3 Moving further down in the tree implies approaching the final adder.

4 Modified Booth does not impose any fundamental problems to the twin-precision concept. It has been evaluated, but is not included in this paper because of space constraints.

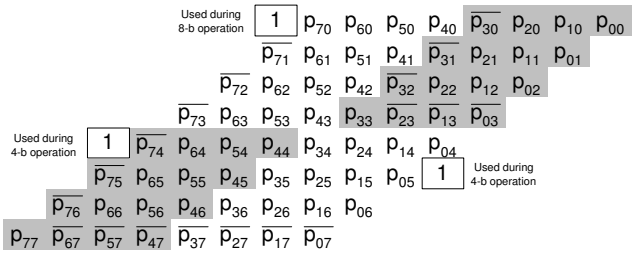

Figure 3: Example showing the inverted partial product bits of two signed 4-b multiplications within a signed 8-b multiplication.

ical one to the new adder. For the *N*/2-b multiplication in the MSP there is no half adder that can be replaced, but an extra level of half adders has to be added, seen at the far left in Figure 4. This added level of half adders does not increase the delay for the *N*-b multiplication, since none of the half adders are in the critical path.

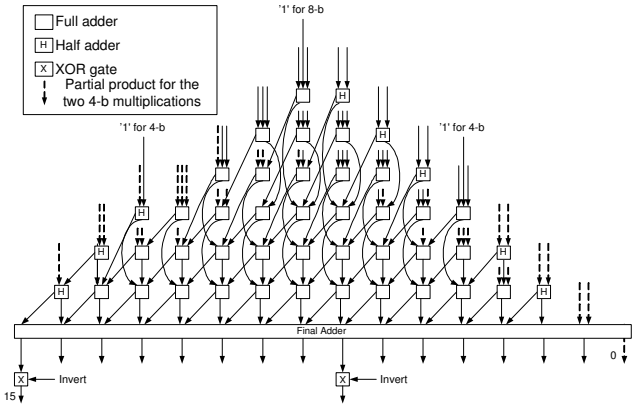

Figure 4: Signed 8-b multiplier capable of doing two signed 4-b multiplications using Baugh-Wooley.

3. Final Adder

The choice of final adder is very important in order to get short delay for both *N* and *N*/2-b multiplications. The recommendations given by Oklobdzija *et al.* [9] are not directly applicable in our twin-precision multiplier, since the delay profile of the multiplier varies with the multiplication precision. It would be possible to use the adder scheme presented by Oklobdzija *et al.* to reduce the delay for the *N*-b multiplication, but this could introduce long delays for the two independent *N*/2-b multiplications. In order to not increase the delay too much for the *N*/2-b multiplications, it is therefore important to have a final adder that is fast for both *N* and *N*/2-b multiplications. Mathew *et al.* [1] presented a sparse-tree carry-lookahead adder that is capable of doing both fast 64-b and fast 32-b additions. This adder scheme has been adapted to the appropriate word length in order to obtain short delays for both *N* and *N*/2-b multiplications, Figure 5.

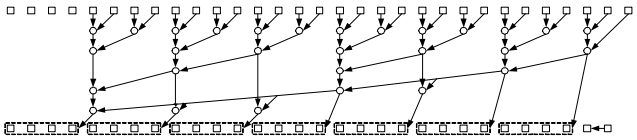

Figure 5: Example of 31-b final adder for a 16-b twin-precision multiplier.

4. Simulation Setup and Results

To evaluate delay and power dissipation, simulations have been performed in a commercially available 0.13-μm technology. The simulated circuit is a 16-b twin-precision tree multiplier, which is capable of performing two 8-b multiplications in parallel, and which uses the fast final adder of Section 3. As reference we use a conventional 8-b and 16-b multiplier, respectively, which both use a Kogge-Stone as final adder. For signed multiplication the Baugh-Wooley algorithm [8] has been implemented for both the conventional and the twin-precision multiplier. All simulations have been done using Spice transistor netlists including estimated wire capacitances. All logic has been implemented as static logic and designed to resemble what could be expected to be found in a standard-cell library. The implemented version of the multiplier cancels the inactive partial products by forcing the AND gates to zero. The impact of using sleep-mode techniques on power and delay has not been investigated[5]. For power simulation 50 random input vectors were applied to HSpice at 500 MHz, a supply voltage of 1.2 V, and an operating temperature of 25 °C. Delay was obtained using PathMill.

Table 1: Reference values normalized to the conventional 8-b multiplier.

Reference	Delay	Power
16 bit	1.40	4.06
8 bit	1.0	1.0

Table 1 lists the values for the conventional 8-b and 16-b multipliers used as comparison references. Table 2 lists delay and power for the twin-precision multiplier. All values have been normalized to the 8-b reference multiplier.

With a conventional 16-b multiplier as reference, the delay of the 16-b twin-precision multiplier operating in 16-b

5 We expect no fundamental problems in introducing sleep-mode techniques in the twin-precision multiplier.

Table 2: Simulation results for a twin-precision 16-b multiplier, where columns 2 and 3 are normalized to the conventional reference 8-b multiplier. Columns 4 to 7 are comparisons against the conventional reference multipliers given in Table 1.

Mode	Delay	Power	Compared to 8-b		Compared to 16-b	
			Delay	Power	Delay	Power
16-b	1.52	4.10			9.0%	0.9%
2x8-b	1.29	2.34	29.0%	16.9%	-7.5%	-42.4%
8-b	1.18	1.13	18.2%	13.3%	-15.3%	-72.1%

mode was 9.0% larger whereas the power dissipation was less than 1.0% larger. When using the 16-b twin-precision multiplier in single 8-b mode, the power dissipation is only 28% of the reference 16-b multiplier. The reason for the power reduction is that in single 8-b mode about two thirds of the multiplier tree is kept at constant zero, eliminating the dynamic power in these parts. The additional decrease in power comes from the reduction of glitches in the multiplier.

With a conventional 8-b multiplier as reference, the delay of the 16-b twin-precision multiplier operating in single 8-b mode was 18.2% larger whereas the power dissipation was 13.3% larger.

When doing two 8-b multiplications in parallel the power dissipation increases by about 4% and the delay is increased with 10% compared to a single 8-b multiplication. The increase in the delay is due to an increased logic depth—the 8-b multiplication in the MSP of the multiplier tree has a longer critical path than the 8-b multiplication in the LSP has, Figure 4. Additionally the logical depth of the final adder is one gate deeper for the MSP which contributes to a longer critical path for the 8-b multiplication computed in the MSP of the tree.

The power dissipation for driving the control signals to set the mode of the multiplier was not included in the power simulation. The control signal used to set the partial product bits to zero, when doing two $N/2$-b multiplications, is connected to the input of N AND gates. In order to cancel out the second $N/2$-b multiplication the control signal is connected to the input of $N/2$ AND gates. It has been shown that it is realistic to expect the multiplier to operate in the same mode for longer durations [2]. Since the control signals only toggle when the mode of the multiplier is changed, the power dissipation for these signals is negligible when the multiplier stays in one mode for longer durations.

5. Conclusion

The twin-precision multiplier presented in this paper offers a good tradeoff between precision flexibility, area, delay and power dissipation by using the same multiplier for doing N, $N/2$ or two $N/2$-b multiplications. In comparison to a conventional 16-b multiplier, a 16-b twin-precision multiplier has 8% higher transistor count and 9% longer delay. The relative transistor count overhead decreases for larger multipliers, since the number of AND gates needed to set the partial products to zero does not grow as fast as the number of adders in the tree.

References

[1] S. Mathew, M. Anders, B. Bloechel, T. Nguyen, R. Krishnamurthy, and S. Borkar. A 4GHz 300mW 64b Integer Execution ALU with Dual Supply Voltages in 90nm CMOS. In *Proceedings of the International Solid State Circuits Conference*, pages 162–163, 2004.

[2] A. Abddollahi, M. Pedram, F. Fallah, and I. Ghosh. Precomputation-based Guarding for Dynamic and Leakage Power Reduction. In *Proceedings of the 21st International Conference on Computer Design*, pages 90–97, 2003.

[3] J. Hughes, K. Jeppson, P. Larsson-Edefors, M. Sheeran, P. Stenström, and L. J. Svensson. FlexSoC: Combining Flexibility and Efficiency in SoC Designs. In *Proceedings of the IEEE NorChip Conference*, 2003.

[4] Z. Huang and M. D. Ercegovac. Two-Dimensional Signal Gating for Low-Power Array Multiplier Design. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, pages I–489–I–492 vol.1, 2002.

[5] T. K. Callaway and E. E. Swartzlander, Jr. Optimizing Multipliers for WSI. In *Proceedings of the Fifth Annual IEEE International Conference on Wafer Scale Integration*, pages 85–94, 1993.

[6] P. Mokrian, M. Ahmadi, G. Jullien, and W. Miller. A Reconfigurable Digital Multiplier Architecture. In *Proceedings of the IEEE Canadian Conference on Electrical and Computer Engineering*, pages 125–128, 2003.

[7] H. Eriksson. *Efficient Implementation and Analysis of CMOS Arithmetic Circuits*. PhD thesis, Chalmers University of Technology, 2003.

[8] C. R. Baugh and B. A. Wooley. A Two's Complement Parallel Array Multiplication Algorithm. *IEEE Transactions on Computers*, 22:1045–1047, December 1973.

[9] 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 Transactions on Computers*, 45(3):294–306, March 1996.