Adv. Radio Sci., 6, 113–118, 2008
www.adv-radio-sci.net/6/113/2008/
© Author(s) 2008. This work is distributed under
the Creative Commons Attribution 3.0 License.

Advances in Radio Science

Comparison of reconfigurable structures for flexible word-length multiplication

O. A. Pfänder[1], R. Nopper[1], H.-J. Pfleiderer[1], S. Zhou[2], and A. Bermak[2]

[1]Microelectronics Department, University of Ulm, Germany
[2]Electrical and Computer Engineering Department, The Hong Kong University of Science and Technology, Hong Kong S.A.R.

Abstract. Binary multiplication continues to be one of the essential arithmetic operations in digital circuits. Even though field-programmable gate arrays (FPGAs) are becoming more and more powerful these days, the vendors cannot avoid implementing multiplications with high word-lengths using embedded blocks instead of configurable logic. But on the other hand, the circuit's efficiency decreases if the provided word-length of the hard-wired multipliers exceeds the precision requirements of the algorithm mapped into the FPGA. Thus it is beneficial to use multiplier blocks with configurable word-length, optimized for area, speed and power dissipation, e.g. regarding digital signal processing (DSP) applications.

In this contribution, we present different approaches and structures for the realization of a multiplication with variable precision and perform an objective comparison. This includes one approach based on a modified Baugh and Wooley algorithm and three structures using Booth's arithmetic operand recoding with different array structures. All modules have the option to compute signed two's complement fix-point numbers either as an individual computing unit or interconnected to a superior array. Therefore, a high throughput at low precision through parallelism, or a high precision through concatenation can be achieved.

1 Introduction

Reconfigurable hardware is becoming more and more popular today, thanks to the continuous evolution and improvement of FPGA devices. FPGA-based systems can be reprogrammed after production by overwriting memory contents and thus adapted to new requirements or even completely different application scenarios. One beneficial effect on the economic side is the potential to save the development and/or production costs of an application-specific integrated circuit (ASIC), specifically for low and medium volume production. FPGAs achieve their high flexibility using programmable lookup tables and reconfigurable routing. However, implementing algorithms with high word-lengths and complex arithmetics on FPGAs can consume large hardware resources and obtain comparatively slow clock frequencies.

On the other hand, arithmetic operations such as addition, multiplication and combinations such as multiply-accumulate (MAC) are required in almost every DSP algorithm, e.g. digital filter designs or Fast Fourier Transform (FFT) implementations (Parhami, 2000). Furthermore, special applications make use of multi-precision arithmetic, i.e. working at different levels of precision during their execution time. Examples for this type of applications are iterative approximation of non-linear functions (Lachowicz and Pfleiderer, 2008), neural networks with different precision in training phase and operation phase (Bermak and Martinez, 2003), and also adaptive filter designs with variable input or coefficient definitions. Since multiplication is crucial in all these situations, optimizing this particular operation can help to save hardware cost and power dissipation. In our approach, we utilize embedded fixed-wired multiplier structures in FPGAs as dedicated hardware. Above that, we upgrade these multipliers with data exchange interfaces and also a sign handling capability. Hence, not only the operand number system, but also the word-length of the multiplication can be directly controlled at run-time, substituting the re-programming step by setting a number of appropriate control signals.

Correspondence to: O. A. Pfänder
(oliver.pfaender@uni-ulm.de)

Published by Copernicus Publications on behalf of the URSI Landesausschuss in der Bundesrepublik Deutschland e.V.

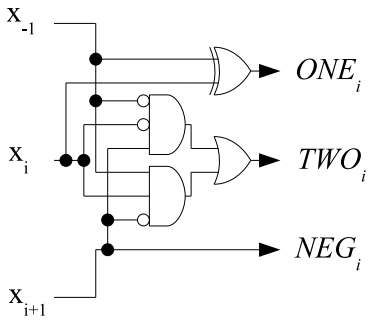

Fig. 1. Realization of a periodical radix-4 Booth encoder, overlapping 3 input bits.

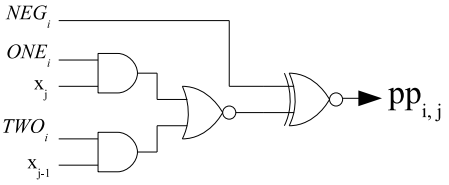

Fig. 2. Partial product generation for non-MSB bits from Booth-recoded signals.

This paper is organized as follows: Sect. 2 describes the basic concept of concatenating several individual multiplier blocks to form a superior multiplier, also covering the improvements over our previous designs, Sect. 3 compares the different approaches in terms of hardware usage and speed, and Sect. 4 presents a novel structure for a block-serial multiplier, followed by a conclusion.

2 Basic concept

In order to overcome the high area and routing expenses when realizing higher word-length multiplication in FPGAs, it is to the best advantage to embed multiplier blocks into the fabric and connect them to the routing network. This is already state-of-the-art in modern commercially available FPGA devices such as the Xilinx Virtex-II Pro and the Virtex-IV with its DSP48 blocks. Both examples feature 18×18-bit multipliers, while our multiplier elements are designed as fully functional $n\times n$-bit blocks that can be combined to form a superior multiplier array. For this work, the parameter n was set to 8 because the percental hardware overhead for reconfiguration would be too large for a lower number (Pfänder et al., 2004), and a larger number would constrain the word-length flexibility too much. By concatenating $m\times m$ uniform blocks, a superior $(m\cdot n)\times(m\cdot n)$-bit multiplier can be realized. Since every element is a fully functional multiplier itself and can work separately at its basic word-length, a high parallelism can be achieved.

There are several degrees of freedom for the multiplier elements' inner structure. Firstly, there is the choice of multiplication algorithm, i.e. parallel, serial, serial/parallel etc., and secondly, the number system must be assigned, in order to handle signed numbers. We have shown in Pfänder and Pfleiderer (2005) that a modified Baugh and Wooley algorithm can be efficiently applied to a parallel and a serial/parallel multiplier that accepts two's complement signed numbers. In both cases, this leads to a regular array structure – also permitting an asymmetric operand scheme – and it also shows a small overhead compared to an unsigned and stand-alone multiplier. Reconfigurable interfaces and configuration options include the exchange of carry and sum input signals with neighboring elements and also the inversion of partial products in certain array positions. In addition, introducing pipeline stages is also possible with the Baugh and Wooley scheme, as shown by Di and Yuan (2003). New improvements of our previous designs include two major updates, namely the the adoption of Booth's arithmetic recoding of input operands (Booth, 1951) and the use of a carry-save instead of a carry-ripple adder array. While the Booth recoding is aimed specifically at reducing the number of partial products to save hardware costs, while still retaining the full precision in contrast to e.g. truncation (Kidambi et al., 1996) both measures help to further accelerate the multiplication.

The radix-4 Booth algorithm encodes three overlapping digits of the multiplier into one new digit (Booth, 1951). The number of multiplier digits and thus the number of partial products is reduced by 50%, thus accelerating the multiplication significantly thanks to a shorter critical path. Although additional effort is needed for the encoding circuit, it only consists of a few logic gates as depicted in Fig. 1. Also, the partial products are one bit larger than the original multiplicand and some additional effort is needed for the partial product generation (PPG). But since their number is divided by two and the addition array is considerably smaller, the encoding can be compensated and the hardware usage is reduced. Lee (2005) presents a dedicated circuit to handle different word-lengths in this context, but its scalability is limited and this method results in a large overhead. Figure 2 shows the logic circuit for our PPG.

Our previous designs, e.g. the configurable parallel/parallel multipliers presented in Pfänder et al. (2004), were based on a carry-ripple adder array. This approach can be combined with a Booth recoding scheme as we have shown in Zhou et al. (2007) which already leads to the speed improvement shown in Table 1 over a standard array multiplier. The arbitrary array size is still possible, while the PPG is realized separately. Control signals manipulate the carry propagation and also the sign handling functionality in certain perimeter cells at run-time. For further acceleration thanks to a shorter critical path, we have re-shaped the inner structure of the partial product addition (PPA) stage and applied a carry-save addition technique instead. This enables us to design the PPA independently from the final adder, which

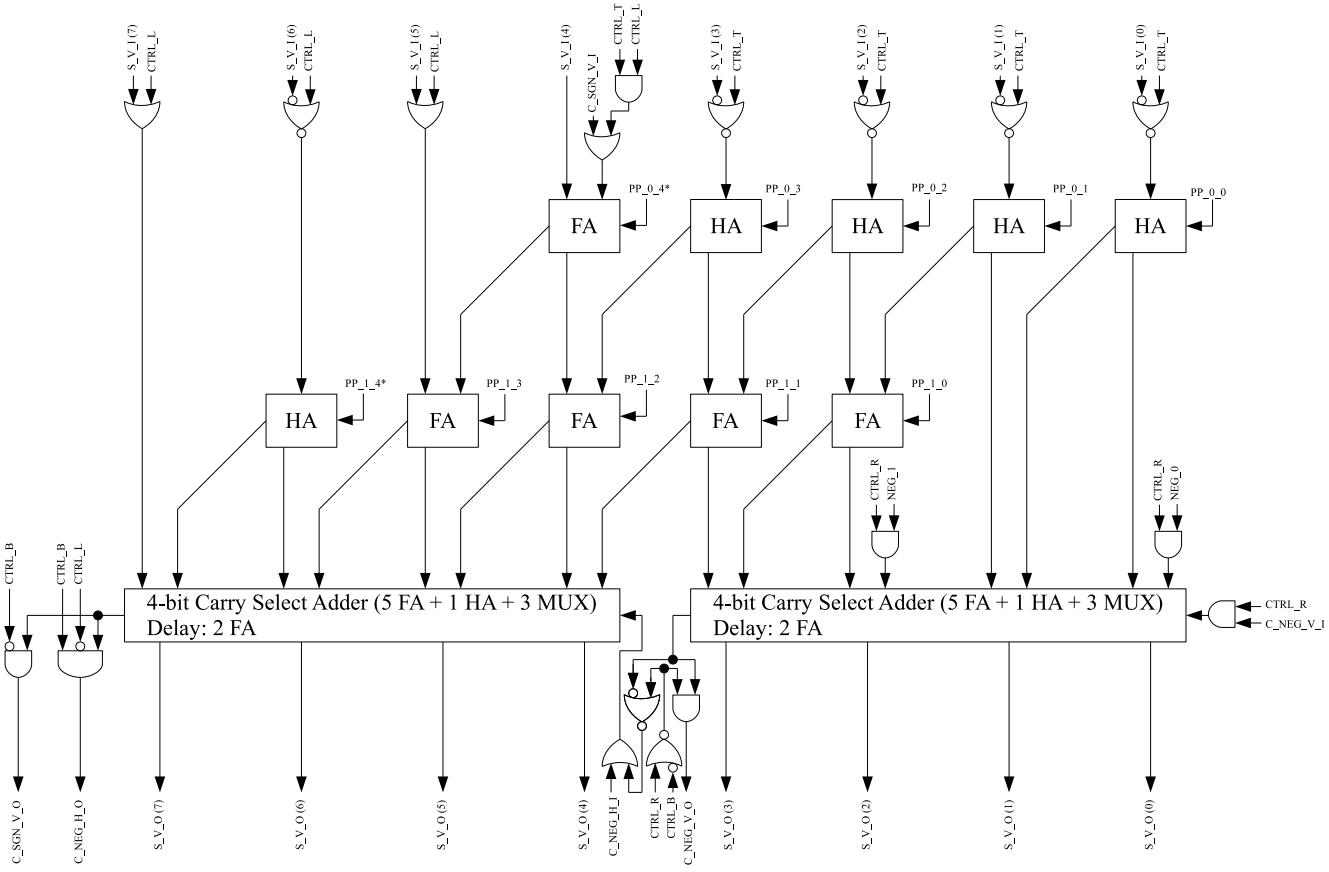

Fig. 3. Reconfigurable Booth multiplier with carry-save partial product generation and accelerated carry-select final adder.

can either be a standard carry-ripple adder row or an accelerated carry-select adder as pictured in Fig. 3.

3 Comparison

In order to perform a meaningful assessment in terms of hardware complexity and data throughput, the different structures for flexible word-length multiplication are compared with a selection of previous designs. We have adopted two benchmarks for comparison: The theoretical and hardware-independent approach is explained in Sect. 3.1 and the implementation results of an abstract hardware description in VHDL targeting a Xilinx Virtex-II Pro FPGA device is summarized in Sect. 3.2. Although this method does not lead to the same results as when using a semi-custom design flow toward an ASIC, it helps to quantify the designs and estimate the implementation costs on a common basis, i.e. the number of occupied slices in the FPGA device is assumed to scale up with the area usage. From this data, we derive a combined and normalized figure of merit in order to compare the multiplier concepts, display them as vertical-bar charts and discuss the results.

Table 1. Speed improvement of Booth recoding scheme over parallel array multiplier and percental overhead for reconfiguration (Zhou et al., 2007).

Word-length (bit)	Speed Advantage	Overhead for Reconfiguration
16	13.8%	2.0%
24	21.5%	1.7%
32	22.5%	1.6%

3.1 Theoretical hardware and speed comparison

Our theoretical hardware and speed comparison takes into account the number of required transistors and the number of full adder delays for a multiplier element. While the transistor numbers as depicted in Fig. 4 calculated for one exemplary implementation, e.g. assuming 24 transistors for one full adder, show no significant differences for the hardware usage, the multiplication times are more interesting, because they represent a hardware-independent figure for comparing the speed. Figure 5 shows the estimated delay in terms of full adders over the multiplier element's word-length.

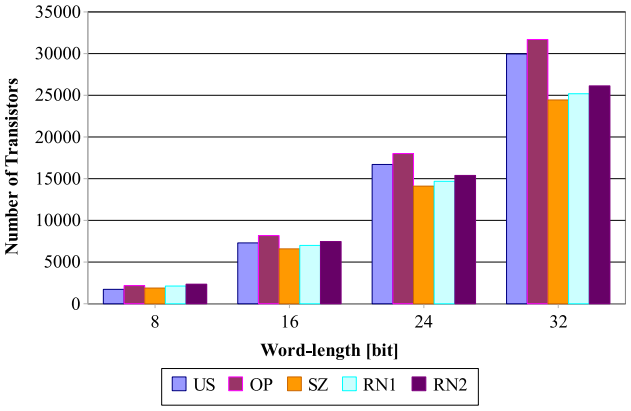

Fig. 4. Theoretical hardware usage for different word-lengths, expressed in terms of transistor numbers.

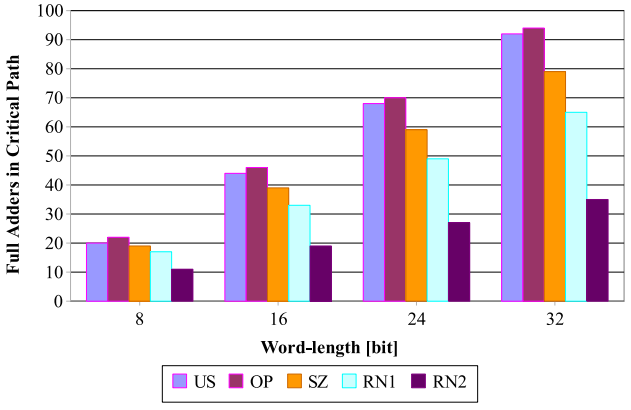

Fig. 5. Theoretical multiplication delay for different word-lengths, expressed in terms of the number of full adders in the critical path.

The diagram legends in Figs. 4–7 contain the following abbreviations: US stands for an unsigned parallel/parallel array. OP is one of our previous designs from Pfänder et al. (2004) using a modified Baugh and Wooley algorithm (Baugh and Wooley, 1973) and a parallel/parallel array. SZ is our first approach to introduce a Booth recoding scheme with a carry-ripple adder array, as covered in Zhou et al. (2007). RN1 incorporates a carry-save adder array and a carry-ripple final adder, while RN2 uses a carry-select final adder instead, as pointed out in the previous section. The numbers used for the diagrams in Figs. 4–5 represent the hardware usage of a basic multiplier element scaled to different word-lengths.

3.2 FPGA synthesis hardware and speed comparison

In order to get actual numbers for the hardware usage of our multiplier elements, we use the synthesis results from mapping the VHDL structural description onto a Xilinx Virtex-II Pro FPGA using the vendor's Integrated Synthesis Environment (ISE) software. More precisely, the number of occupied slices in the FPGA device is used as a measure for area usage,

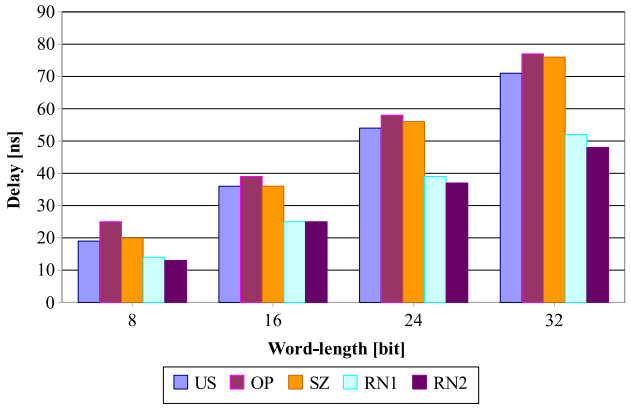

Fig. 6. Estimated delay for a cascaded superior multiplier assembled from 8×8-bit basic elements, without interconnect influence.

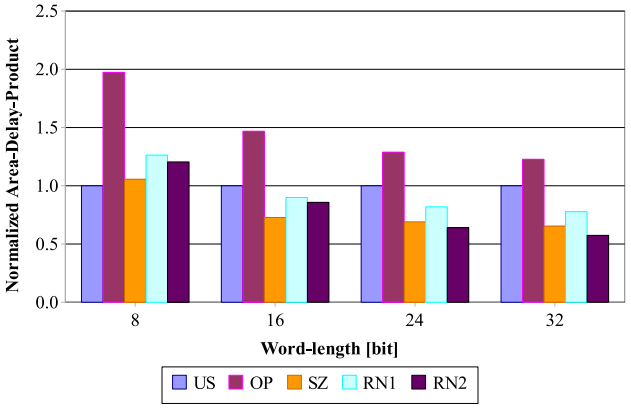

Fig. 7. Combined assessment criterion for scaled multiplier elements with different word-lengths, normalized with respect to an unsigned parallel multiplier.

and the critical path delay from the synthesis report is interpreted as a measure for multiplication time. Our experiments have shown that the synthesis results reflect the trends of the hardware-independent comparison and therefore confirm the theoretical prognosis.

The estimated multiplication time in nanoseconds for a cascaded multiplier array that was constructed from 8×8-bit basic elements is taken from the synthesis reports and shown in Fig. 6. Moreover, the influence of the interconnect and routing network has to be considered, but this should be done using a semi-custom ASIC design flow instead.

3.3 Combined comparison of area and delay

In addition to the comparisons summarized in the previous subsections, we have adopted the product of area and delay as a cost function normalized to the respective values of an n-bit unsigned parallel multiplier:

$$\text{Cost} = \frac{\text{No. of transistors} \times \text{No. of full adder delays}}{\text{Unsigned reference multiplier}} \quad (1)$$

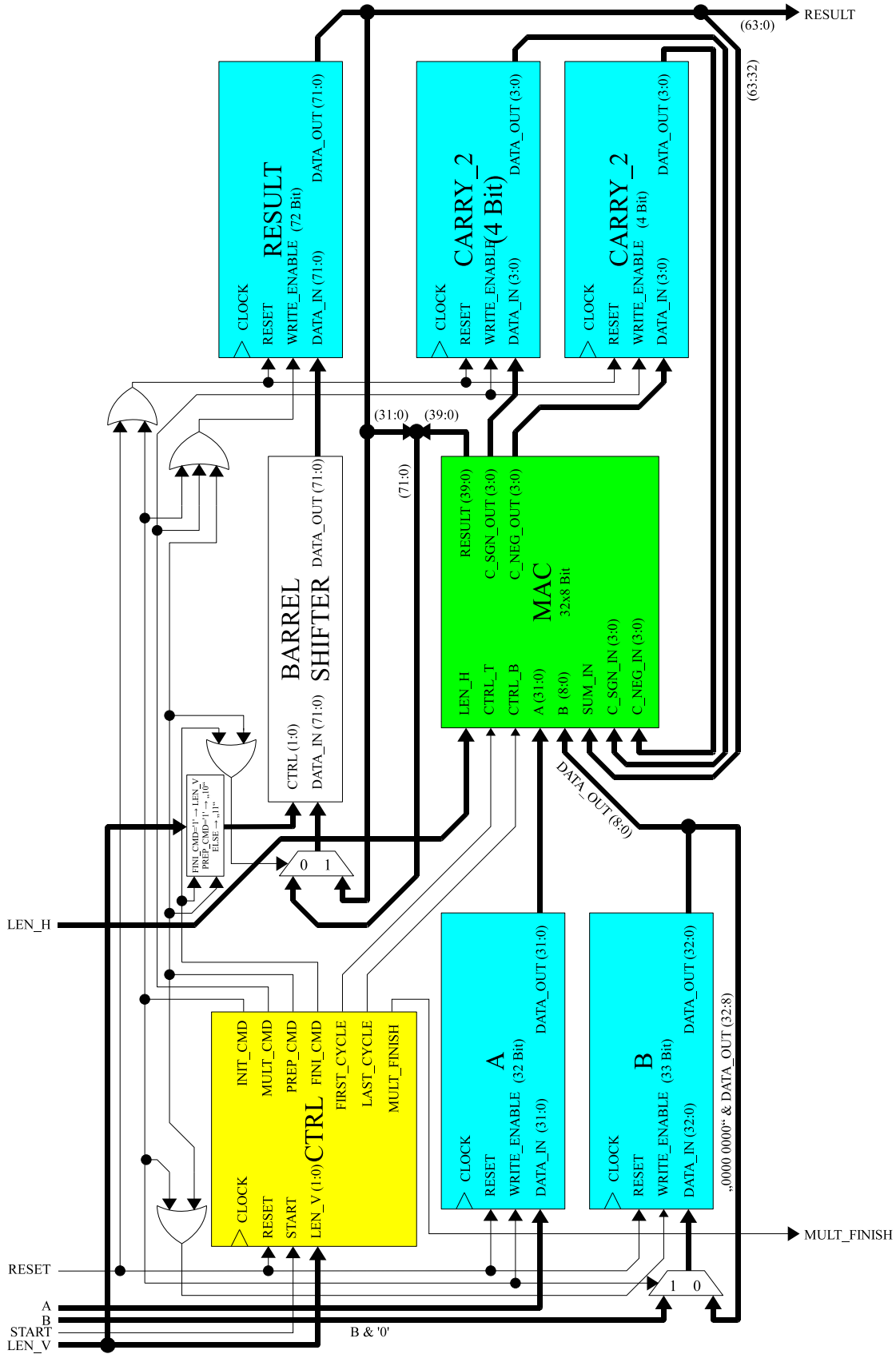

Fig. 8. Block-serial multiplication scheme with control circuit, registers, shifter and a configurable 32×8-bit MAC core.

Figure 7 plots the cost function over different word-lengths for scaled multiplier elements and clearly shows the significantly higher efficiency due to the Booth encoding scheme.

4 Novel block-serial approach

Assembling arbitrary large multiplier structures through the concatenation of configurable basic blocks leads to high routing efforts and a complex interface design communicating with the circuit periphery. In order to avoid this hardware overhead a priori, we have come up with an approach to provide a configurable precision multiplier with less operating expense by introducing a half-serial computation of the multiplier operand. This means that only one of the PPA rows is actually implemented in hardware – in the case of an 8-bit basic block this is equivalent to a 32×8-bit multiplier-accumulator (MAC). The other cells are replaced by buffered intermediate results that are fed back to the MAC sum input. After an certain number of computation steps, the multiplication result is valid at the output with the required precision that can be 64-bit maximum. With this arrangement, multiplications of 8×8-bit up to 32×32-bit can be realized, where both input operands are varied independently in steps of 8-bit. The major attribute of this block-serial structure is the fact that in contrast to a fully parallel multiplier, a number of clock cycles depending on the word-length n of the multiplier is required for computing the product. Since it is segmented in blocks of 8-bit, the number of computation steps including multiplication, shifting and addition is $n/8$. The multiplier's word-length is given by the 2-bit input LEN_V as a positive number in the range of $[0\ldots3]$ where $n=(\mathsf{LEN\_V}+1)\cdot8$. The principle design, composed of a control circuit, a shifter registers for the operands, a 32×8-bit MAC and feedback lines for the intermediate results to the sum input is shown in Fig. 8 on page 117. CTRL incorporates a finite state machine that receives and interprets LEN_V and addresses the registers and the MAC block accordingly. The outstanding feature of this design is its possibility to compute at a precision that is configurable at run-time, performing $(8\cdot x)\times(8\cdot y)$-bit up to 32×32-bit multiplications and using 60% less hardware than a fully parallel 32×32-bit array multiplier with sign handling capability and data exchange interfaces.

5 Conclusions

In this work, we have described and compared a number of four different run-time reconfigurable multiplier architectures in terms of hardware usage and speed. Starting from a previous design based on a parallel/parallel carry-ripple array, we have improved the architecture by a Booth recoding scheme, a carry-save adder array and an accelerated final adder. The overhead for a reconfigurable module over a static unsigned carry ripple multiplier varies between 24% and 37% at a module word length of 8-bit, but the critical path can be reduced by up to 45%, therefore accelerating the design significantly. For an 8-bit basic building block, our new approach requires 50% less hardware and decreases the delay by about 60% according to our synthesis experiments targetting a Xilinx Virtex-II Pro FPGA device. Moreover, we have applied a normalized area-delay product for fair comparison and verified the significant benefit of the Booth encoder, while still retaining the possibility to change the multiplier's precision at run-time. We have also presented a novel block-serial multiplication approach which turned out to be noticeably more area-efficient than a full 32×32-bit reconfigurable multiplier, since it requires 60% less hardware. We are currently working on the implementation of our multiplier architectures in standard-cell ASIC technology in order to avoid the FPGA synthesis limitations and make further improvements, and also developing a detailed routing and interface design.

Acknowledgements. The work described in this paper is supported by a grant from the Research Grant Council of Hong Kong S.A.R. and the German DAAD (Project No. G-HK019/05).

References

Baugh, C. R. and Wooley, B. A.: A Two's Complement Parallel Array Multiplication Algorithm, IEEE Trans. Computers, C-22, 1045–1047, 1973.

Bermak, A. and Martinez, D.: A compact 3D VLSI classifier using bagging threshold network ensembles, IEEE Trans. Neural Networks, 14, 1097–1109, 2003.

Booth, A. D.: A Signed Binary Multiplication Technique, Quart. J. Mechanics and Appl. Mathematics, 4, 236–240, 1951.

Di, J. and Yuan, J. S.: Run-time Reconfigurable Power-aware Pipelined Signed Array Multiplier Design, in: International Symposium on Signals, Circuits and Systems, 2, 405–408, 2003.

Kidambi, S. S., El-Guibaly, F., and Antoniou, A.: Area-efficient multipliers for digital signal processing applications, IEEE Trans. Circuits and Systems II, 43, 90–95, 1996.

Lachowicz, S. W. and Pfleiderer, H.-J.: Fast Evaluation of Nonlinear Functions using FPGAs, Adv. Radio Sci., 6, 2008.

Lee, H.: Power-Aware Scalable Pipelined Booth Multiplier, IEICE Trans. Fundamentals, E88-A, 3230–3234, 2005.

Parhami, B.: Computer Arithmetic: Algorithms and Hardware Designs, Oxford University Press, Oxford/New York, 2000.

Pfänder, O. A. and Pfleiderer, H.-J.: Dynamische Rekonfiguration von arithmetischen Einheiten auf Bitebene, Adv. Radio Sci., 3, 319–323, 2005, http://www.adv-radio-sci.net/3/319/2005/.

Pfänder, O. A., Hacker, R., and Pfleiderer, H.-J.: A Multiplexer-Based Concept for Reconfigurable Multiplier Arrays, in: International Conference on Field-Programmable Logic and Applications, pp. 938–942, Antwerp, Belgium, 2004.

Zhou, S., Pfänder, O. A., Pfleiderer, H.-J., and Bermak, A.: A VLSI Architecture for a Run-Time Multi-Precision Reconfigurable Booth Multiplier, in: International Conference on Electronics, Circuits and Systems, Marrakech, Morocco, 2007.