Fixed-Point Arithmetic in FPGA

M. Bečvář, P. Štukjunger

Arithmetic operations are among the most frequently-used operations in contemporary digital integrated circuits. Various structures have been designed, utilizing different features of IC architectures. Nevertheless, there are very few studies that consider the design of arithmetic operations in Field Programmable Gate Arrays (FPGAs), a re-programmable type of digital integrated circuit. This text compares the results achieved when implementation of basic fixed-point arithmetic units in FPGA.

Keywords: arithmetic, fixed-point, FPGA.

1 Introduction

Nowadays, arithmetic operations are among the most common operations in digital integrated circuits. Even the simplest circuit adds, subtracts or multiplies something. Therefore there are high requirements on arithmetic operations. The computation should be fast and the area consumed by the arithmetic units should be small. These are two basic requirements, which are unfortunately contradictory.

Numbers in digital integrated circuits can be represented in various ways. The most widely known and most frequently used is fixed-point addition and multiplication representation.

As mentioned above, the question of efficient fixed-point arithmetic operations is interesting when implementing a digital circuit. This is not a new question. Many studies have been done and many articles and books have been published on the topic of arithmetic. However, these have only rarely considered this question in conjunction with FPGAs (Field Programmable Integrated Circuits). In the design of an arithmetic unit in FPGA, it was common to use the corresponding operator of an HDL (Hardware Description Language) and rely on a synthesis tool. No thorough study and comparison of different structures of basic fixed-point arithmetic units has yet been published.

This paper shows the results of our study [7], based on implementing and then comparing of fixed-point arithmetic units. Virtex-II FPGA was chosen for implementation since it possesses arithmetic-support features common for contemporary FPGAs. The paper is divided into five sections. After the introduction, the second section describes the selected implementation platform. The third section outlines the implantation of arithmetic units. The fourth section consists of a discussion and a comparison of the results of the measurements. The last section concludes the text with a summary of the results.

2 Platform

As shown in Table 1, contemporary FPGAs possess special features to support of arithmetic operations. The support features differ from one FPGA vendor to another. It was not the goal of the study to compare the different FPGAs. We decided to use Xilinx [8] Virtex-II FPGAs as the implementation platform. This is a very popular implementation platform in contemporary designs, and it has special support features for both addition and multiplication.

Table 1: Arithmetic support features in FPGAs

Vendor	Chip	Features
Altera	Stratix II	Adder logic, carry chain, DSP support.
Atmel	AT40KAL	Vector Multipliers. The AT40KAL's patented 8-sided core cell with direct horizontal, vertical and diagonal cell-to-cell connections implement ultra fast array multipliers without using any busing resources.
Lattice	IspXPGA	Dedicated carry chain and Booth multiplication logic (extra two-input AND).
QuickLogic	Eclipse-II	ECU blocks (up to 8-bit MAC functions) for DSP (8b multiplier, 16b adder), 12 ECUs in the largest device.
Xilinx	Virtex-II	Dedicated fast carry chain and carry propagation logic, embedded 18x18 bit signed multipliers with associated RAM blocks.

Cad tools are another important part of the implementation platform. The choice of tools affects the results of the implementation. Again, a comparison of CAD (Computer Aided Design) tools was not the goal of our study. The tools were chosen according to their availability in our department.

The following tools were used in the relevant implementation steps:

- *Synthesis:* Leonardo Spectrum 2001, Mentor Graphics [4]
- *P&R, Timing Analysis:* Xilinx ISE 5.2i Foundation, Xilinx [8]
- *Simulation:* ModelSim 5.5f, Mentor Graphics [4]

3 Implementation

This section lists the implemented adders and multipliers. They were chosen from different sources [1-3][5-6] to meet the objectives of the study: to explore the characteristics of arithmetic unit structures when implemented in FPGA, and to examine the FPGA architecture itself including the features that support arithmetic operations.

Classes of Implemented Fixed-point Adders:

- **Carry-Ripple Adder (CRA):** A basic adder structure based on full adders connected into a chain. This is the simplest structure for implementation and also the slowest for operation. Nevertheless, it is important for comparison with the other structures.
- **Generated Adder (GA):** An adder generated by a synthesis tool. In contrast to CRA it utilises the dedicated carry chains of Xilinx Virtex II FPGAs.
- **Carry-Lookahead Adder (CLA):** The carry-Lookahead adder uses two special signals G(enerate) and P(ropagate) to predict the propagation of a carry signal without using a carry chain. Several CLAs were implemented with different building-block types and sizes.
- **Carry-Skip Adder:** Based on CLA Carry-Skip, this adder uses only the P(ropagate) signal, and for this reason, it is much easier to implement. It can be seen as a compromise between the complexity of CLAs and the long delay of CRAs.
- **Carry-Select Adder:** Two sets of adders are used, one with the carry-in signal driven to logic 0 and the second with the carry-in signal driven to 1. The result is selected by multiplexers controlled by the actual carry-in signal. This structure is suspected to have the largest consumption, which should however compensate its performance.

Fixed-point multipliers can be divided into two groups by the type by their operation: serial mode multipliers and parallel mode multipliers. A serial mode multiplier computes bits of the result one by one in each computation cycle. On the other hand, a parallel mode multiplier computes all bits of the result at once. Several multiplier structures from the two groups were selected for implementation. In the case of the parallel mode, not only conventional versions but also pipelined versions of multipliers were implemented.

As they were implemented, the multipliers take n-bit operands and produce a 2n-bit result. No adjustments of the result, such as radix-point position or length truncation, were considered in the implementation.

Classes of Implemented Fixed-point Multipliers:

Serial mode multipliers:

- **Classical Multiplier:** The most basic serial mode multiplier structure that computes multiplication in a human-like way, i.e., going through the bits of one operand and simultaneously accumulating the second operand and shifting the sum.
- **Booth Multiplier:** A serial mode multiplier that uses Booth recoding. This enables it to compute signed number multiplication and, for radices greater than two, also increases its speed by computing several bits of the result in one computational step.
- **CSA Multiplier:** A serial mode multiplier that uses Carry-Save Adders, adders that have 3N inputs and 2N outputs. Carry-Save Adders have a shorter delay than a normal full adder, which shortens the length of a single step of the multiplier.
- **Booth + CSA Multiplier:** Combining the concepts of Booth and CSA multipliers, this multiplier uses radix-4 Booth recoding to reduce the number of computational steps and CSA adders to shorten the duration of a single computational step.

Parallel mode multipliers:

- **Array Multiplier:** A parallel mode multipliers with a structure based on the basic formula for computation of a multiplication. It consists of N2 cells connected into an array-like structure.
- **Wallace Multiplier:** A parallel mode multiplier that uses a tree of CSA adders. Thanks to the CSAs at each level of the tree, the number of operands is reduced by a factor of 3:2.
- **Generated Multiplier (Generated HW):** A parallel mode multiplier generated by a synthesis tool. It utilises 18-bit hardware multipliers embedded in Xilinx Virtex II FPGAs.
- **Combinative Generated Multiplier (Generated C):** A parallel mode multiplier also generated by a synthesis tool, but with disabled usage of the embedded multipliers. This multiplier was implemented for the purposes of comparison with other structures and also for investigating the pipelining abilities of the synthesis tool.

VHDL hardware description language was used for implementation; however there is no reason it should have any impact on the results of the experiments. Different arithmetic unit structures were implemented with a common interface, so that they can be used interchangeably (i.e. an adder can be replaced by another adder, and a multiplier by another multiplier). The units are parametrized, which allows the user to choose the width of the operands (and consequently of the result). However, this it can be done only statically before the start of the implementation process. For more details on implementation, see [7].

4 Results

The results of the experiments with implemented arithmetic structures are shown below. The target FPGA chip was Xilinx Virtex-II, specifically XC2V500 in the FG456-6 package. There was no particular reason for choosing exactly this part rather than some other member of the family. It was discovered later that some of the 64-bit multiplier structures did not fit in this chip, so a larger (XC2V1500) chip had to be used instead. Place and route reports were used as a source for the evaluation.

4.1 Fixed-point addition

Tab. 2 shows the results of measurements carried out on the implemented adder structures of 16, 32 and 64 bit-width. The results show that the best adder structure in terms of both time and area is the one that uses dedicated carry chains as depicted in figure Fig. 1. None of the techniques that try to speed up the addition by cutting the carry chain, applied to this structure, brought any improvement in speed, but led to an increase in the occupied area. Therefore, the results of such structures are not shown.

As mentioned above, the speed-up techniques cripple the performance of the Generated Adder, and the following observations are only valid for adders based on the CRA adder.

Table 2: Results of fixed-point adders

	N = 16		N = 32		N = 64	
	Area (Slices)	**Delay** (ns)	**Area** (Slices)	**Delay** (ns)	**Area** (Slices)	**Delay** (ns)
CRA	23	13.52	47	23.99	95	40.27
Generated	**8**	**4.05**	**16**	**5.09**	**32**	**8.38**
CLA	29	7.97	61	19.06	127	24.84
Carry-Skip	23	11.54	47	15.67	95	18.02
Carry-Select	27	13.27	67	16.47	135	23.88

CLA: CLA occupies the largest area among adders with a speed-up technique applied, without bringing better speed improvement than the other techniques. The regularity of CLA with blocks of size 2 proved to have an advantage over other CLAs in smaller area consumption.

Carry-Skip Adders: No extra area was occupied by the additional carry propagation logic – a very favourable result. With no need for extra area, the Carry-Skip Adder accelerates CRA to less than half delay at 64 bits.

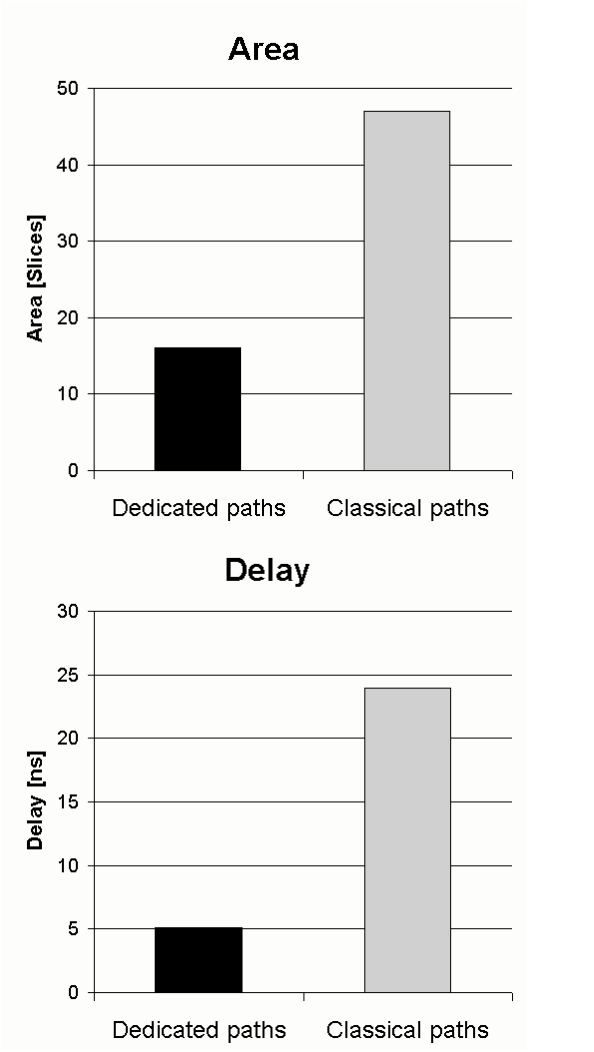

Fig. 1: Fixed-point adders: comparison of the delay and area consumption of an adder with dedicated carry chains vs. an adder utilising classical routing paths (an example for 32-bit data width)

Carry-Select Adders: The carry-Select Adder requires extra area. In addition, it does not shorten the delay of computation as much as the Carry-Skip Adder.

4.2 Fixed-point multiplication

Table 3 shows the results of measurements carried out on different multiplier structures. 16, 32 and 64-bit multipliers were implemented.

As mentioned above, the implemented multipliers can be divided according to their operation mode into two groups: serial and parallel mode multipliers. The multipliers within each group have similar characteristics, and the results can often be generalized to the whole group.

As in the case of adders, two parameters were examined: area and delay. Before dealing with the delay of the multiplier, it is necessary to state clearly what it means. Two measures have to be distinguished: delay and response time. *Response time* refers to the total time required for computation of the result. On the other hand, the *delay* of a sequential circuit is defined as the reverse value of its clock frequency. Note that the delay of the parallel mode multipliers is equal to their response time, but they response times of serial mode multipliers and pipelined multipliers are *multiples* of their delays. These time measures are both important, and were therefore the object of our examination.

4.2.1 Area

Fig. 2 depicts the area consumption of selected multiplier structures with 32-bit operand width. This figure shows, and it is also valid in general, that serial mode multipliers consume considerably a smaller area than parallel mode multipliers. The implementation of wide (larger than 64 bits) parallel mode multipliers approaches the limits of technology – not only the limits of FPGA, but also the capacity of the synthesis tool. The results are much better when embedded hardware multipliers are used, but the number on one chip is limited [8] and consumption grows exponentionally with operand width. Note that the area consumption depicted by the figure does not include the area of the embedded multipliers.

4.2.2 Delay

Two time measures were observed on the multiplier structures: delay and response time. The response time gives an overview of the overall performance of the unit. The delay limits the maximum clock frequency of the system on which the given multiplier structure will be used. Parallel mode structures have a shorter response time, while serial mode

Table 3: Results of fixed-point multipliers

	N = 16		N = 32		N = 64	
	Area (Slices)	**Delay** (ns)	**Area** (Slices)	**Delay** (ns)	**Area** (Slices)	**Delay** (ns)
Serial Mode Multipliers:						
Classical	40	4.60	71	4.94	136	8.75
Booth-2 (radix 2)	41	4.63	73	4.89	138	7.99
Booth-4 (radix 4)	**55**	**4.93**	**104**	**5.37**	**202**	**9.09**
CSA	62	3.46	119	3.67	234	4.54
Booth-4 + CSA	101	4.62	191	4.86	370	5.45
Parallel Mode Multipliers:						
Array *	364	28.96	1536	53.45	4187	99.97
Wallace *	390	19.81	1542	27.18	6164	37.00
Generated	**0**	**14.87**	**49**	**18.54**	**295**	**28.94**
Generated C	134	20.32	533	26.27	2109	29.97
Pipelined Multipliers:						
Array [P2] *	274	14.95	1059	29.98	4193	59.92
Wallace [P]	446	4.64	1654	4.95	6367	9.9
Generated C [P4] *	188	4.95	635	9.94	3317	11.61

Note1: [Pn] denotes a pipelined multiplier with a given number of stages, where appropriate.

Note2: 64-bit versions of multipliers marked * did not fit in an XC2V500 chip. An XC2V1500 was used instead.

Note3: Area consumption does not include the area of embedded multipliers. Generated multipliers of 16, 32 and 64 bit-width utilize 1, 4, and 16 embedded multipliers, respectively.

multipliers have a better delay criterion. This is an expected result, thanks to the philosophy of the operation mode. In parallel mode the computation is done in a single step. The step is long, longer than one step of the serial mode multiplier, but shorter than the sequence of steps that has to be carried out by a serial mode multiplier to compute the result.

The pipelining technique was applied on parallel mode multipliers to see if it brings any improvements. This technique could not be applied when embedded multipliers were used, since the multipliers cannot be divided to allow the insertion of registers for the pipeline. The advantages and disadvantages of pipelining were observed on the implemented

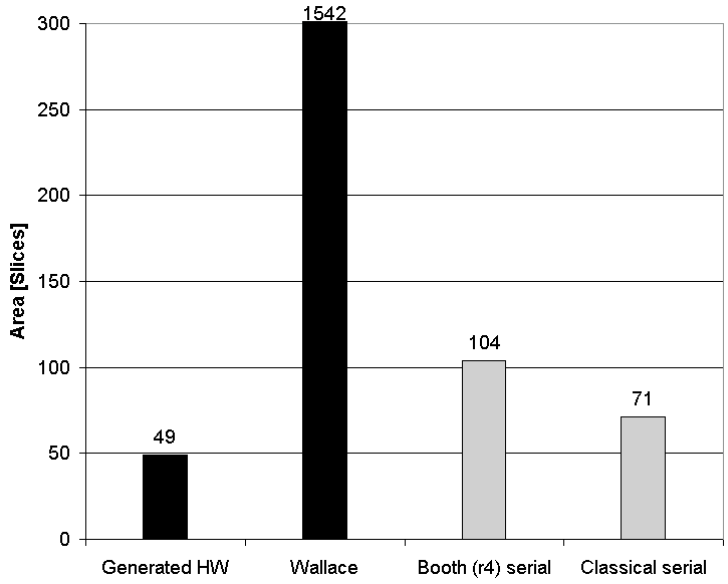

Fig. 2: Area of fixed-point multipliers: selected types of fixed-point multipliers are shown, two parallel mode multipliers on the left and two serial mode multipliers on the right (all with 32-bit data width)

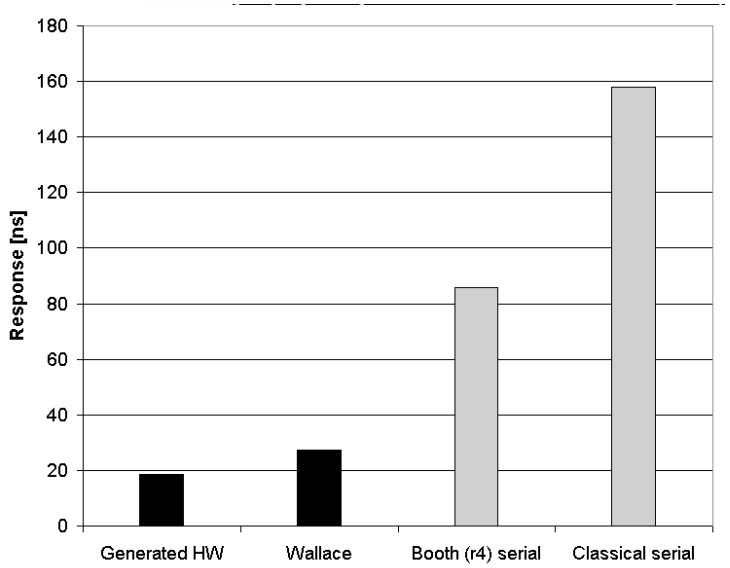

Fig. 3: Response time of fixed-point multipliers: selected types of fixed-point multipliers are shown, two parallel mode multipliers on the left and two serial mode multipliers on the right (all with 32-bit data width)

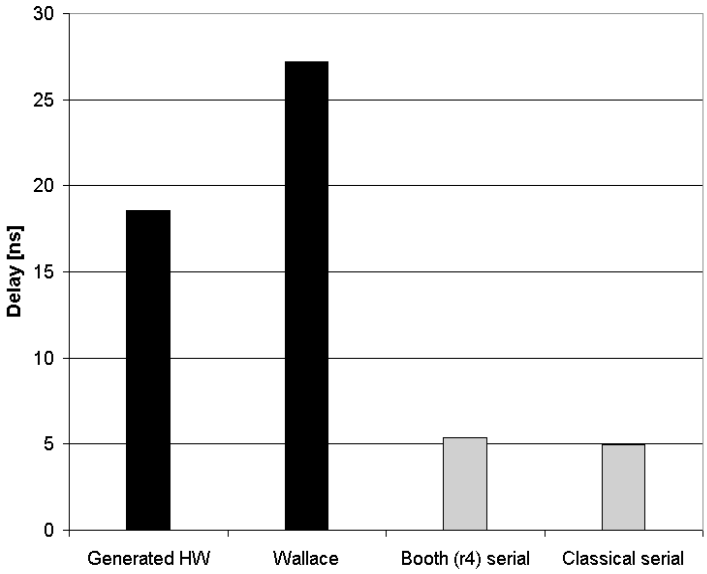

Fig. 4: Delay of Fixed-point Multipliers: selected types of fixed-point multipliers are shown, two parallel mode multipliers on the left and two serial mode multipliers on the right (all with 32-bit data width)

structures. However, pipelining did not show any considerable improvements. When a parallel mode multiplier is to be implemented, embedded multipliers are the best option.

5 Conclusion

Arithmetic operations in contemporary digital integrated circuits have an important role. Our study [7] provides a comparison of the structures of fixed-point adders and multipliers implemented in FPGA. From a number of well-known [1–3], [5–6] implementation architectures, several structures were chosen to observe their characteristics when they are implemented in FPGA and, at the same time, to investigate the possibilities of the FPGA architecture itself. The Xilinx Virtex-II FPGA family was chosen as the implementation platform, because it provides special features to support arithmetic operations, a trend that seems to be common at the present time for FPGA chips.

Nine fixed-point adder structures were implemented in the study. The results show that a structure using Virtex's dedicated fast carry chains has the best results in terms of both area and delay criteria. None of structures utilizing a speed-up technique showed better results. If no dedicated carry chain is available, the Carry-Skip Adder is the best option, because it doubles the speed of an adder with a normal carry chain without any extra area consumption.

Ten fixed-point multipliers were implemented. They can be divided according to their operation mode into two groups: serial and parallel mode multipliers, respectively. The parallel mode structures provide a shorter response time,

but they also require a significantly larger area in terms of CLB slices. A parallel multiplier structure that utilizes embedded hardware multipliers proved to have the shortest response time. However, the embedded hardware multipliers are consumed very rapidly for bit-widths larger than 18 and, furthermore, the number on a chip is limited [8]. Serial mode structures proved to have advantages for applications that require small area usage with a short clock cycle. They provide a reasonable response time even for relatively large bit-widths (32 bits and more), on a very small area. This gives the opportunity to exploit the parallelism between a large number of serial-mode multipliers in a single FPGA.

References

[1] Ercegovac, M. D., Lang, T.: *Digital Arithmetic,* Morgan Kaufmann Publishers, San Francisco, 2003.

[2] Hennessy, J. L., Patterson, D. A., Goldberg, D.: *Computer Architecture: A Quantitative Approach*: Appendix H Computer Arithmetic, Elsevier Science, 1995.

[3] Koren, I.: *Computer Arithmetic Algorithms,* Prentice-Hall, Englewood Cliffs, New Jersey, 1993.

[4] MENTOR GRAPHICS, web page, http://www.mentor.com/.

[5] Omondi, A. R.: *Computer Arithmetic Systems* (Algorithms, Architectures and Implementations), Prentice-Hall International (UK) Limited, 1994.

[6] Pluháček, A.: *Projektování logiky počítačů* (Designing Computer Logic), CTU Publishing House, Prague, 1992.

[7] Štukjunger, P.: *Arithmetic in FPGA*, Diploma thesis, CTU in Prague, 2004.

[8] XILINX, web page, http://www.xilinx.com.

Ing. Miloš Bečvář
e-mail: becvarm@fel.cvut.cz

Ing. Petr Štukjunger
e-mail: stukjup@fel.cvut.cz

Department of Computer Science & Engineering,

Czech Technical University in Prague
Faculty of Electrical Engineering
Karlovo nám. 13
121 35 Prague, Czech Republic

