Multi-input multiplexers and selectors as basic shifter blocks in arithmetic devices operating in floating point: comparative analysis

I V Ushenina
Penza State Technological University, 1a pr. Baidukova, Penza, 440039, Russia
E-mail: ivl23@yandex.ru

Abstract. In case of the Field Programmable Gate Array implementation of arithmetic devices operating in floating point, the implementation of shifters is associated with some challenges. This work compares two approaches to the formation of basic shifter blocks: as selectors using carry chains and as multi-input multiplexers. Both approaches use exclusively a FPGA programmable logic. The work shows that basic blocks as multiplexers require more than twice fewer FPGA logic slices and are nottable for 10-20% better performance when compared to those based on selectors.

1. Introduction
Today, real-time floating-point arithmetic operations are in great demand in many digital signal processing applications, including high-precision scientific computations, etc. A Field Programmable Gate Array (FPGA) is often used to implement such calculations.

One of the problems in the FPGA implementation of floating-point calculators is the implementation of shifters used to normalize the result of arithmetic operations. Shifter adders (subtractions) are also used to adjust the orders of summable operands.

The presence of shifters has a considerable effect on the performance and resource intensity of an arithmetic device. This is because hardware shifters are not present among FPGA logic resources. Logarithmic shifters [1] or shifters based on carry chains [2] can be implemented on the programmable logic of modern FPGAs.

A logarithmic shifter is an array of multiplexers. N of N-input multiplexers whose control inputs are supplied with a variable indicating the width of a shift is required to shift a N-bit operand. Multi-input multiplexers on FPGAs are several stages of multiplexers with fewer inputs. Each stage shifts by an amount equal to degree 2. If a multi-input multiplexer is implemented on 2:1 multiplexers and a maximum shift width is \( M \), the total number of stages is determined by \( \log_2 M \) rounded to the higher integer [1]. Multi-input multiplexers are implemented on the basis of tabular converters and 2:1 multiplexers that are part of FPGA logic slices [2].

An alternative to multi-input multiplexers is selectors [2]. Selectors are based on FPGA carry logic chains that make small delays in signal propagation and therefore can ensure the higher speed of a shift.

This work compares the performance and resource intensity characteristics of multi-input multiplexers and selectors as basic shifter blocks implemented on a FPGA programmable logic. 24-bit and 53-bit multiplexers and selectors are considered, which corresponds to arithmetic operations
according to the IEEE 754 standard with single and double precision. All studies were conducted with the use of the Xilinx XC7A200T-3 FPGA. The characteristics of the devices were designed and analyzed in the Vivado 2017.2 system.

2. Related work
The works dedicated to the FPGA implementation of arithmetic devices and their shifters are mainly related to logarithmic and cyclic shifters [3-11]. In [3-8] there is insufficient adaptation of FPGA resources to implement shifters. At the same time, [3] assesses the effect of complementation of FPGA logic slices with hardware shifters. [4] notes that the effectiveness of implementation of cyclic shifters on a FPGA would be increased by having a sufficient number of tri-state buffers on them. [5] discusses the possibility of decreasing the chip area occupied by the shifter by using the multiplexers of FPGA trace resources.

A designing result for a device on a FPGA depends not only on the choice of a chip, but also on the method of description of that device, as well as on project implementation settings. [6-8] shows that the implementation results of shifters may be considerably different depending on how they are described. So, for example, the resource intensity of the shifter described in VHDL, in behavioral style and with the use of the case operator is more than twice that of the shifter for which the use of 2:1 multiplexers [6] is clearly indicated. [7] compares the implementation results of shifters when using the VHDL for and if operators. The use of the if operator is shown to expand the cycle of shift operations and increase the shift speed by 2-3 times. [8] notes that during the implementation of logarithmic shifters the chip area is most effectively used when 4:1 multiplexers are implemented on FPGA 4-input tabular converters. [2] shows that for the most effective implementation of multi-input multiplexers and selectors it is necessary to carry out their descriptions in structural style, i.e. by clearly indicating chip resources in use and relations between them.

The performance of a device implemented on a FPGA depends on how it is placed on the chip. Resource-intensive devices are often divided into blocks. Each block has a separate area on the chip [12]. This allows us to optimize the placement of each block independently of the others and stabilize their performance. Therefore, the effect of the chip area limitation on the performance of basic shifter blocks should be assessed.

This work considers the possibility of implementation of currently available basic shifter blocks on FPGAs. All devices are described in structural style and implemented in two variants: placement without limitation and limited chip areas.

3. Basic shifter blocks
3.1. Selector with a unitary-coded control bus
Figure 1 (right) shows a selector chart presented in [2]. A remarkable feature of this chart is the representation of the select[23:0] control bus in a unitary code. A logical unit is shown as the bit number of the d[23:0] data bus to be passed to the selector output.

Each tabular converter included in the selector is supplied with three bits of the select bus and three bits of the d bus. If all three bits of the select bus that come to the tabular converter are equal to zero, a logic unit appears at the output of that converter and the associated MUXCY multiplexer connects its output to the ci input. If select[i]=1, d[i] is inverted and appears at the tabular converter output.

In each FPGA logic slice [13] it is possible to implement a selector with inputs of up to 12. If a selector with a greater number of inputs is needed, it is implemented on several slices that are placed on the chip strictly one above the other.
Figure 1. (left): Bit generator of the selector control bus from the 8-bit shift amount representation; (right): 24:1 selector with a unitary-coded control bus.

In arithmetic devices, the shift amount is represented in a binary code, that is why for the selector shown in figure 1 it is necessary to have the sv[7:0] shift amount converter from a binary code into a unitary one. Figure 1 (left) shows the bit control bus generator that processes the 8-bit binary representation of the shift amount.

Figure 1 shows the 24-bit selector. The 53:1 selector can be prepared in a similar manner.

3.2 Selector with a binary-coded control bus
A binary-to-unitary control bus converter requires a significant number of logic slices and occupies an additional place on the chip. In addition, the use of this converter takes away from a selector its main advantage: small propagation delays for interconnects because programmable connections of general purpose are used in addition to specified ones between carry chain links [2]. Figure 2 (left) shows the 24:1 selector that does not require a control bus to be converted into a unitary code. Each selector link consists of a tabular converter and a carry chain multiplexer controlled by it. If sv[4:0] coincides with the bit number of the (i) data bus, the d(i) bit is transmitted via the carry chain to the selector output. The upper selector link analyzes the high-order bits of the sv bus and provides a logic zero at the selector output if sv[7:0] > 23. The 53:1 selector requires 53 links to select one of the d[53:0] bus bits and two additional links to check the sv[10:0]<52 condition.

3.3 Multi-input multiplexer
Figure 2 (right) shows the 24:1 multiplexer with an 8-bit control bus. Its 16:1 and 8:1 multiplexers are built in accordance with the recommendations given in [2]. If sv[7:0]< 23, the tabular converter passes to the output one of the signals coming from the 16:1 and 8:1 multiplexers. If sv[7:0]>23, a logic zero is formed at the tabular converter output. The 53:1 multiplexer requires four 16:1 multiplexers, one 4:1 multiplexer and one sv[10:6] bus analyzer. The 4:1 multiplexer and the sv[10:6] bus analyzer are implemented on tabular converters. The sv[10:6] bus analyzer is involved in the control of the 4:1 multiplexer.
4. Methods
The description of every device studied in the work is made in VHDL, in structural style. The device description and implementation of related projects were carried out in the Vivado 2017.2 system for XC7A200T-3 FPGA.

The performance of the basic blocks was assessed according to the results of a temporary project analysis. To start the temporary analysis, the input and output signals of the analyzed devices were synchronized with a clock signal for the period of which a limitation was imposed [14]. As a result, the time analyzer assessed signal propagation delays within the basic blocks and calculated a maximum clock rate for each of them. In order to assess performance more accurately, the implementation of the basic blocks was carried out with several versions of the clock signal period. The chip area (selected area) available for the placement of the basic blocks was limited using the PBlock tool [12].

5. Results and discussion

| Bits | Resource intensity | Maximum clock rate for different placement methods, MHz |
|------|-------------------|------------------------------------------------------|
|      | LUTs  | Slices | Without limitations | With limited chip area |
| 24 bits | 8     | 2      | 561                | 583                |
| 53 bits | 18    | 5      | 549                | 522                |

Table 2. Performance and resource intensity of the 24:1 selector with the control bus converter from a binary code into a unitary one taken into account. When specifying resource intensity, the FPGA resources occupied by synchronization elements are not taken into account.
Table 3. Performance and resource intensity of the selectors and multiplexers (24:1 and 53:1). When specifying resource intensity, the FPGA resources occupied by synchronization elements are not taken into account.

| Device          | Resource intensity | Maximum clock rate spread for different placement methods, MHz |
|-----------------|--------------------|---------------------------------------------------------------|
|                 | LUTs   | Slices | Without limitations | With limited chip area |
| 24:1 selector   | 25     | 7      | 468-481             | 458-466                |
| 24:1 multiplexer| 7      | 3      | 483-544             | 504-522                |
| 53:1 selector   | 55     | 14     | 329-365             | 342-354                |
| 53:1 multiplexer| 16     | 6      | 439-464             | 432-451                |

Predictably, the selector with the control bus converter presented in figure 1 shows low performance. Based on the Vivado temporary analyzer reports, the control bus converter takes up to 50% of the total data processing time. At the same time, the selector presented in figure 1 (right) shows the best performance among all the devices under consideration and its resource intensity is comparable to that of the multi-input multiplexer. But since in arithmetic devices the shift amount is represented in a binary code, the selector presented in figure 1 is not useful.

The analysis of the implementation reports related to the selector and multiplexer presented in figure 2 shows that propagation delays for programmable connections within the selector are considerably lower (about 0.4 ns for 53-bit basic blocks). But due to the fact that selectors are long chains of logic elements (25 and 55 links), signal propagation delays for selector logic elements are higher (about 1 ns for 53-bit basic blocks) and multi-input multiplexers have an overall performance advantage.

The limitation of the chip area intended to place and trace basic blocks has two consequences: 1) a slight decrease in the maximum clock rate (by 2-5%) due to the limitation of the variants considered by means of placement and tracing; 2) stabilization of placement and tracing results expressed in the decreased spread between the minimum and maximum clock rates of basic blocks.

6. Conclusion
In case of the modern FPGAs implementation of shifters included in floating-point arithmetic devices, it is appropriate to use multi-input multiplexers built on tabular converters and 2:1 multiplexers as basic blocks. When compared to selectors based on carry logic, they are characterized by lower resource intensity and higher performance.

References
[1] Rabaey J M, Chandrakasan A P and Nikolić B 2003 Digital Integrated Circuits (N.J., USA: Prentice Hall)
[2] Chapman K 2014 Multiplexer design techniques for datapath performance with minimized routing resources Xilinx Application Note
[3] Beauchamp M J et al. 2008 Architectural modifications to enhance the floating-point performance of FPGAs IEEE T. Vlsi. Syst. 16.2 177-87
[4] Jian L, Tessier R and Mencer O 2003 Floating point unit generation and evaluation for FPGAs Proc. 11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, FCCM (Napa, CA, USA: IEEE)
[5] Yehdhih O M M et al. 2012 Reducing the cost of floating-point mantissa alignment and normalization in FPGAs Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays (Monterey, California, USA: ACM)

[6] Malik A and Ko S-B 2006 A study on the floating-point adder in FPGAs Proc. Canadian Conference on Electrical and Computer Engineering, CCEC (Ottawa, ON, Canada: IEEE)

[7] Shirazi N, Walters A and Athanas P 1995 Quantitative analysis of floating point arithmetic on FPGA based custom computing machines Proc. Symposium on FPGAs for Custom Computing Machines (Napa Valley, CA, USA: IEEE) pp 155-62

[8] Lee B and Burgess N 2002 Parameterisable floating-point operations on FPGA Proc. Conf. Record of the Asilomar Conference on Signals, Systems and Computers 2 1064-8

[9] Shirke M, Chandrababu S and Abhyankar Y 2017 Implementation of IEEE 754 compliant single precision floating-point adder unit supporting denormal inputs on Xilinx FPGA Proc. IEEE International Conference on Power, Control, Signals and Instrumentation Engineering, ICPCSI (Chennai, India: IEEE) pp 408-12

[10] Yiannacouras P, Rose J and Gregory Steffan J 2005 The microarchitecture of FPGA-based soft processors Proc. Int. Conf. on Compilers, Architecture, and Synthesis for Embedded Systems (San Francisco, CA, USA: ACM) pp 202-12

[11] Chen J et al. 2018 New Hardware and Power Efficient Sporadic Logarithmic Shifters for DSP Applications IEEE Transactions on CAD 37(4) 896-900

[12] Floorplanning Methodology Guide 2013 (San Jose, California, US: Xilinx Inc)

[13] 7 series FPGAs Configurable Logic Block 2016 (San Jose, California, US: Xilinx Inc)

[14] Timing Closure User Guide 2012 (San Jose, California, US: Xilinx Inc)