Complex Multiplier suited for FPGA structure

Keiichi Satoh[1], Jubee Tada[2], Kenta Yamaguchi[3], and Yasutaka Tamura [4]
[1]Yamagata University Graduate School of Science and Engineering, Department of System and Information Engineering
[2,3,4]Yamagata University Graduate School of Science and Engineering, Department of Informatics
4-3-16, Johnan, Yonezawa, Yamagata, 992-8510, Japan
E-mail: [1]keiichisato1@mail.goo.ne.jp

Abstract: In this paper, we propose complex multiplier suited for FPGA structure to achieve higher performance and lower cost. The complex multiplier is based on LUT (Look-Up-Table) and carry-chain from FPGA structure, we utilize Booth algorithm for partial product generation and Wallace tree utilizing effectively LUTs and carry-chains in the FPGA structure for the partial products compression to design it. We design Wallace trees of various types utilizing LUTs and carry-chains, the complex multipliers implemented the trees are synthesized by synthesis tool. Consequently, the proposed complex multipliers are superior to one synthesized by operator ('*', '+', and '-') from VHDL description for both the path delay and the scale.

1. Introduction

FPGA (Field Programmable Gate Array) is used in various applications such as signal processing, communication, and control. In these applications, complex multiplications are used to process in algorithms such as FFT[1], DCT[2], and convolution in the frequency domain[3], and so on. These computations require more and more computational power to process various signal processing in higher speed. In addition, it is desirable for hardware implementation in FPGA to implement more compact and higher performance complex multipliers for effective parallel operation.

Complex multiplication is composed of four multiplications, one addition, and one subtraction. Thus, four multipliers and two adders are required for a complex multiplier. Therefore, a complex multiplier requires more circuit resources and becomes longer path delay than a normal multiplier.

In this paper, first, we represent main structure in Xilinx Virtex-5 FPGA. Second we represent complex multiplication computation, subsequently, we represent the structure for partial product generation and partial product compression suited for the FPGA. Fourth, we discuss the comparison result from the performance and the scale for the designed complex multipliers of various types. Finally, we conclude from the result.

2. FPGA structure

FPGA (Xilinx Co.) basic structure is composed of CLBs (Configurable Logic Block) and wire matrixes, also the CLB is roughly composed of LUTs, carry-chain, and FFs (Flip Flop) [4]. The CLB structure is shown in **Figure 1**. Arbitrary combinational logics are realized by the LUT's data input and output pattern. Also, carry-chain is dedicated wire connected to LUT output ports to perform high-speed carry propagation, which is used to generate operational and logic circuits.

Virtex-5 FPGA has six-inputs and one-output LUTs; the single LUT can output one-bit data from six-input data. If this resource is properly used for the circuit design, we may be possible to obtain more compact and higher performance circuit. Thus, it is important for circuit suited for FPGA to consider the structure; we remark the structure according to the consideration. Subsequently, we select Virtex-5 as the target device and attempt to design 18-bit complex multiplier.

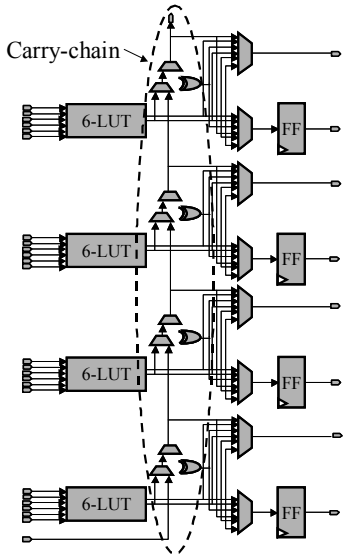

Figure 1. CLB structure in Virtex-5 FPGA.

3. Complex multiplier structure

With complex data *(A + jB)* and *(C + jD)* (*j* : imaginary number), the complex multiplication is represented as follows.

$$(A + jB)(C + jD) = AC - BD + j(AD + BC). \quad (1)$$

AC – BD and *AD + BC* are the product for real and imaginary part, respectively. Also, schematic diagram of complex multiplier is shown in **Figure 2**. As four the products are generated, complex multiplier is required of four multipliers and two adders. **Figure 3.** is concrete block diagram for complex multiplier. A description will be given in due order below about partial product generation and compression operation to explain the structure in **Figure 3**.

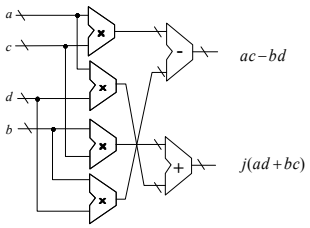

Figure 2. The schematic diagram of complex multiplier.

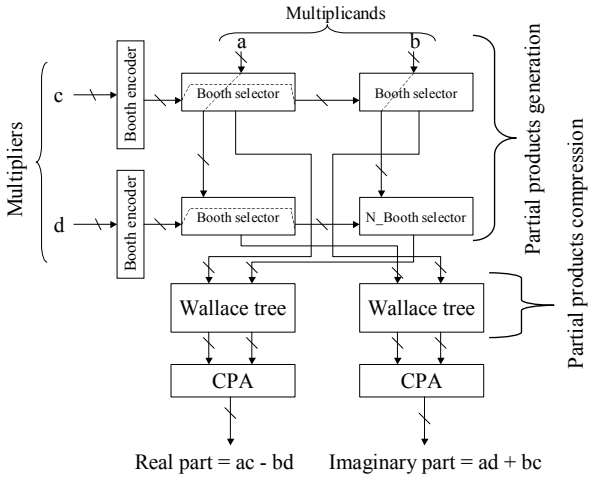

Figure 3. Block diagram for complex multiplier.

3-1. Partial product generation

For the partial products generation, we introduce Booth algorithm[5] to reduce the number of partial products for above "*ac*", "*bd*", "*ad*", and "*bc*" by roughly one half, respectively. For multiplication of 2's complement numbers, the two-bit encoding using this algorithm scans a triplet of bits. When multiplier B is divided into groups of two bits, and the algorithm is applied to this group of divided bits. The multiplier B in 2's complement representation is expressed by

$$B = -b_{n-1}2^{n-1} + \sum_{j=0}^{n-2}(b_j 2^j) \qquad (2)$$

$$= (b_{n-3} + b_{n-2} - 2b_{n-1})2^{n-2} + (b_{n-5} + b_{n-4} - 2b_{n-3})2^{n-4} + \cdots$$
$$+ (b_{n-k-1} + b_{n-k} - 2b_{n-k+1})2^{n-k} + \cdots + (b_{-1} + b_0 - 2b_1)2^0$$
$$= \sum_{k=0}^{n/2-1}(b_{2k-1} + b_{2k} - 2b_{2k+1})2^{2k}.$$

Where, n is an even number, b_{n-1} represents the sign bit b_s , and b_{-1} =0. The product Z(=AB) is then given by

$$Z = AB = \sum_{j=0}^{n/2-1} A(b_{2j-1} + b_{2j} - 2b_{2j+1})2^{2j}, \qquad (3)$$

where, replace $b_{2j-1} + b_{2j} + b_{2j+1}$ with Pj,

$$= \sum_{j=0}^{n/2-1} AP_j 2^{2j} . \qquad (4)$$

P_j has a value of 0,±1, and ±2 by Table 1., which depends on the values of the adjacent three bits on the multiplier b_{2j-1}, b_{2j}, and b_{2j+1}. The Booth encoders generate the three bits.

The multiplicand A is computed by the P_j, Booth selectors generate the partial products; however, the products "*bd*" is regarded as negative number from equation (1). Thus, the Booth selector corresponding "*bd*" is improved to allow generating of negative partial product bits. It is N_Booth selector in **Figure 3.** Also the Booth encoding table is shown in Table 2.

Table 1. 2-bit Booth encoding.

b_{2j+1}	b_{2j}	b_{2j-2}	P_j	$A \cdot P_j$
0	0	0	0	0
0	0	1	1	A
0	1	0	1	A
0	1	1	2	2A
1	0	0	−2	−2A
1	0	1	−1	−A
1	1	0	−1	−A
1	1	1	0	0

Table 2. Booth encoding for negative partial products.

b_{2j+1}	b_{2j}	b_{2j-2}	P_j	$A \cdot P_j$
0	0	0	0	0
0	0	1	−1	−A
0	1	0	−1	−A
0	1	1	−2	−2A
1	0	0	2	2A
1	0	1	1	A
1	1	0	1	A
1	1	1	0	0

Booth algorithm is applied to equation (1); the representation is described following as:

$$AC - BD + j(AD + BC) \qquad (5)$$
$$= \sum_{j=0}^{n/2-1} A(c_{2j-1} + c_{2j} - 2c_{2j+1})2^{2j} + \sum_{j=0}^{n/2-1} B(-d_{2j-1} - d_{2j} + 2d_{2j+1})2^{2j}$$
$$+ j(\sum_{j=0}^{n/2-1} A(d_{2j-1} + d_{2j} - 2d_{2j+1})2^{2j} + \sum_{j=0}^{n/2-1} B(c_{2j-1} + c_{2j} - 2c_{2j+1})2^{2j}).$$

From above the equation, the number of partial products for either real or imaginary part becomes generally approximately N+2 when the data length is N-bit. However, as the two of sign extension bit series[5] are constant, these can become form of single sign extension bit series by summing in advance. Thus, the number of partial products is N+1, respectively. **Figure 4.** is the concrete partial products layout map for imaginary part's computation when the data length is 6-bit, where c_i and c_j indicate a partial product bit introduced to add 1 at the LSB position in the j-th partial product if the encoded result is a negative value. Also, c_i and c_j are decided by the Booth encoders, these are summed by Half Adders in the Booth selectors after the multiplier bits were encoded by the Booth encoders, subsequently the sum(c_s) and carry(c_c) bits are generated, respectively.

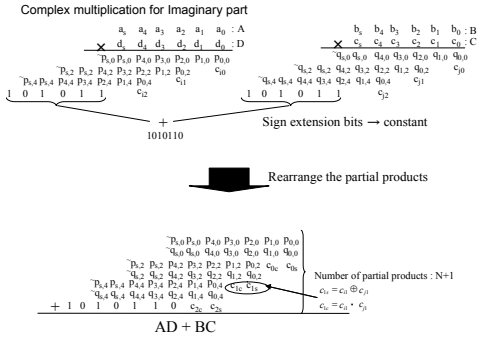

Figure 4. partial products layout map for 6-bit complex multiplication.

3-2. Compression operation

It is efficient for compression operation of the generated partial products to utilize Wallace tree[5]. Generally Wallace tree utilized 4:2-compressors[5] are known an approach to effectively reduce partial products in VLSI. However, utilizing 4:2-compressors in FPGA may be inefficient because the FPGA structure is based on six-input LUT. In other words, the FPGA resources may not be effectively consume to utilize 4:2-compressors when the characteristics of FPGA structure were considered. As an approach to design Wallace tree suited for FPGA, we propose utilization of 6:3-compressor and carry-chain. The schematic diagram of 6:3-compressor structure is shown in

Figure 5. Also, **Figure 6.** is the compression process when six 6-bit partial products are compressed by six 6:3-compressors.

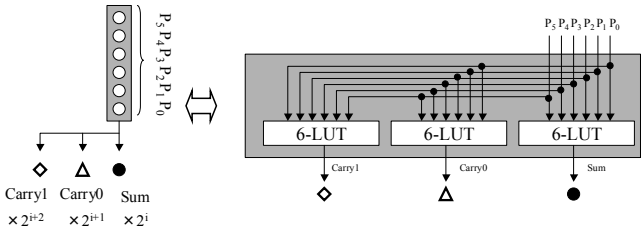

Figure 5. 6:3-compressor (left: schematic diagram for partial products compression, right: the internal structure).

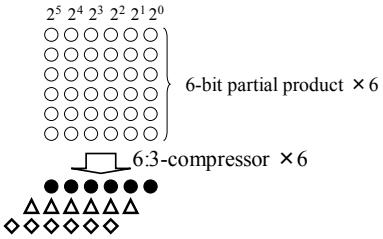

Figure 6. compression process by 6:3-compressors.

The 6:3-compressor is based on the six inputs LUT; it has six input ports and three output ports. Concretely, it contains the three output bit patterns corresponding to six input bit patterns. Thus, 6:3-compressor is composed of three six-input LUTs. By utilizing the unit, six partial product bits input from the same bit position can be added without carry from the first lower bit. Thus, the compression operation is performed by data access to the LUT. Consequently, the path delay reduces efficiently. In addition, the FPGA resources are effectively utilized every the LUT, the resource utilization cost will be lower.

On the other hand, we utilize carry-chain to generate an adder, the carry-chain is usually utilized by HDL description of '+' operator.

We utilize 6:3-compressors and carry-chains and design Wallace trees of various types. The designed tree types are classified into following three types; first type is the tree composed of 6:3-compressors only (***6:3 type***). Second type is the tree composed of carry-chains only (***Carry-chain type***). And the third type is the tree utilized both 6:3-compressors and carry-chains (***Hybrid type***). Concretely, ***Hybrid type*** is the tree to compress the partial products by utilizing 6:3-compressors and carry-chains alternately. In addition, this is classified into the two types that the tree compresses the generated partial products by the order of beginning 6:3-compressors and next carry-chains (***Hybrid-former type***) and the order of beginning carry-chains and next 6:3-compressors (***Hybrid-latter type***). The schematic diagrams of abovementioned Wallace trees for 18-bit complex multiplier are shown into **Fig.7**. In the tree of a) and d), Full Adders are utilized because the number of partial products in the after-compression stage tree stage is three. Also, CPA stands for Carry Propagate Adder; carry-chain is utilized to construct it.

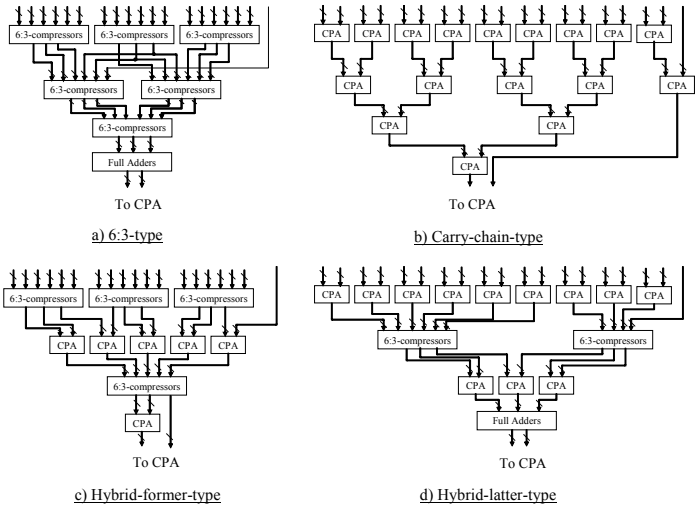

Figure 7. The schematic diagram of the proposed type's Wallace trees.

4. Design

We design the complex multipliers according to **Fig.3** by implementing abovementioned trees. And the design environment is as follows.

- HDL (Hardware Description Language) : VHDL
- Design tool : Xilinx, ISE version 9.1.03i
- Simulator : Mentor Graphics, ModelSimXE 6.2c
- Target device : Virtex-5 - XC5VLX220-2FF1760

5. Experimental results

Subsequently, we synthesize the complex multipliers by the parameters setting of "Optimization Goal: Speed" and "Optimization Effort: Normal" in the synthesis tool. And the path delay and the scale are compared from the synthesis results. First, we compare complex multipliers implementing ***6:3-type*** tree and tree composed of 4:2-compressors only to confirm the effectiveness of 6:3-compressor in the FPGA. 4:2 compressor is a compressor which is often utilized in ASIC design. The result is shown in Table 3.

Table 3. Comparison of 6:3 and 4:2-compressor only tree.

Complex multiplier structure	Number of LUTs	Path delay[ns]
6:3 compressor	1605	5.62
4:2 compreossor	1749	9.68

The path delay and the scale (number of LUT) for ***6:3-type*** complex multiplier is approximately 42% shorter and 9% smaller than another one. Thus, 6:3-compressor type tree is effectively utilized the FPGA resources because 6:3-compressor is the matched structure for the FPGA, additionally 6:3-compressor doesn't utilize the carry from the first lower bit. On the other hand, the path delay for 4:2 compressor's one is longer. 4:2 compressor contains five-input and three-output ports[5] (see Figure 8) because it utilizes the carry ($Carry_{in}$) from the first lower bit to perform the compression operation. However, the carry is dependently generated by the first lower partial product bits (W_0~Z_0) . Thus, the structure placed two the LUTs in series

are required because 4:2-compressor can not struct with one-stage's LUT in the FPGA. Consequently, longer path delay is necessarily generated in the FPGA.

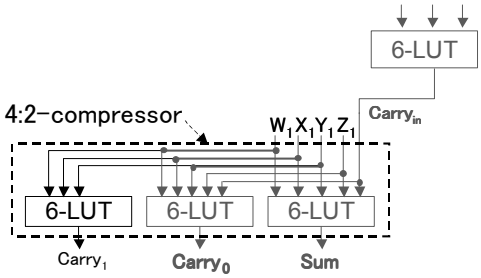

Figure 8. 4:2 compressor implemented in the FPGA

Next, we compare abovementioned type's complex multipliers and ones generated utilizing logic blocks and embedded multipliers (DSP48Es) in FPGA by the '+' and '*' operators in HDL description. The comparison for the scale and the path delay are shown in **Figure 9.** and **Figure 10.**, respectively.

Consequently, the path delay for ***6:3-type*** complex multiplier is the shortest and the scale for ***Hybrid-latter type*** complex multiplier is the smallest in the compared with all the complex multipliers, respectively. Next, the path delay and the scale for the complex multipliers designed utilizing the proposed type's trees are relatively compared with ones generated by the '+' and '*' operators. Consequently, ***6:3 type*** complex multiplier are approximately 42% shorter and 20% smaller than logic block type (no use DSP48Es), moreover the path delay is approximately 16% faster than DSP48 type, respectively. ***Carry-chain type, Hybrid-former type,*** and ***Hybrid-latter type*** complex multipliers are approximately 5% shorter and 36% smaller, 19% shorter and 24% smaller, and 18% shorter and 42% smaller than logic block type, respectively.

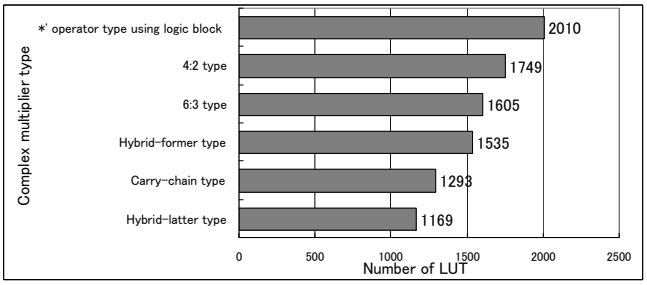

Figure 9. Comparison for the scale (Number of utilized LUTs).

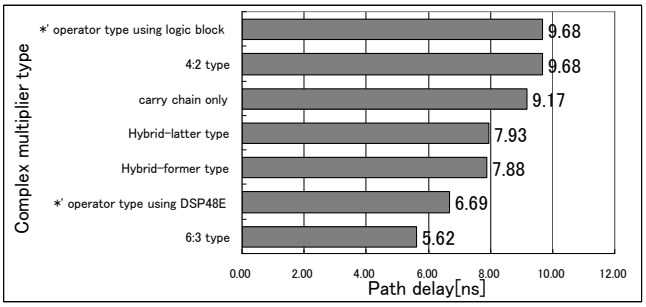

Figure 10. Comparison for the path delay.

In addition, FPGA resources are effectively utilized every the LUT, the resource utilization cost is expected to become lower. From abovementioned, we mention the following considerations.

6:3 type complex multiplier becomes larger scale in order to utilize the LUTs for all the units to construct the tree, whereas, the LUT-to-LUT connection from the tree structure accelerates to reduce the path delay because 6:3-compressors match the FPGA structure. Consequently, the complex multiplier can obtain the highest performance.

In the ***Carry-chain type*** complex multiplier, it is considered that many the LUTs are relatively utilized to construct the adder containing the carry-chain synthesized by '+' operator. Consequently, the scale is no minimum in the other types. Also, the "carry-chain"-to-"carry-chain" connection is inefficient because carry signal generated by the adder is transferred to the carry-chain in the next adder. Consequently longer carry propagation is performed, and the path delay may be longer.

In the ***Hybrid type***, the LUTs and the carry-chains are alternately utilized. However, the different is the frequency to utilize the LUT. From **Figure 9.**, ***Hybrid-former type's*** LUT utilization is more than another one. Thus, it becomes larger scale than another one. On the other hand, as ***Hybrid-latter type's one*** utilizes carry-chain and less the LUTs comparing with another one, the path delay and the scale is balanced. Thus, low-cost complex multiplier can construct.

6. Conclusion

We consider complex multiplier suited for FPGA contained six-input LUTs and design paying attention to resources in the FPGA structure.

From above the result, utilizing LUT based 6:3-compressors and carry-chains are efficient to construct Wallace tree suited for FPGA. This is a result that FPGA resources are effectively utilized because the designed structures fitted for FPGA's structure. Also, when the complex multipliers utilized 6:3-comporessors and carry-chains are utilized in the FPGA, in case of performance serious consideration and in case of cost serious consideration are suitable to utilize ***6:3-type*** and ***Hybrid-latter-type*** complex multiplier for designing higher performance and lower cost HW architecture for the FPGA.

References

[1] Steven W. Smith ,"Digital Signal Processing: A Practical Guide for Engineers and Scientists - chapter9 Applications of the DFT", p180-184, Newnes, Sept. 2002.

[2] X. Shao and S. G. Johnson, "Type-II/III DCT/DST algorithms with reduced number of arithmetic operations," arXiv.org e-Print archive, p. arXiv:cs.DS/0703150, Mar. 2007.

[3] K.Satoh, J.Tada, H.Yanagida, and Y.tamura, "Parallel Image Reconstruction Operation by Dedicated Hardware for Three-Dimensional Ultrasound Imaging", pp.1522-1525, Proc. of IEEE UFFC, Nov. 2007

[4] Xilinx Co.," Xcell journal vol.58.59", 2007 Spring.

[5] Vojin G. Oklobdzija, "THE COMPUTER ENGINEERING HANDBOOK", CRC PRESS.