Collaboration Projects

Implementation and Evaluation of RLS Adaptive Array Using FPGA

We implemented RLS-based adaptive array algorithm in FPGA. By improving processing speed and optimizing the circuit configuration with systolic array architecture, we realized a dramatic reduction of circuit scale. This research was conducted jointly with the Arai laboratory (Professor Hiroyuki Arai), Department of Electrical and Computer Engineering, Graduate School of Engineering, Yokohama National University.

Fumio Kira and Keizo Cho

1. Introduction

Recursive Least Square (RLS) is one of the algorithms that can be used to update array weights in adaptive array antennas. Although the calculation load is large, it achieves fast convergence and is thus effective under fading environments; it shows great promise in mobile communication applications [1]. We first constructed a fully functional testbed in which the RLS algorithm is implemented using Field Programmable Gate Arrays (FPGA), which are programmable integrated circuits. Then, we proposed a new architecture of systolic arrays [2] that allows a significant reduction of the number of processing clock cycles required for updating the array weights. We demonstrated that it is possible to reduce the circuit scale dramatically without impairing the processing speed by reusing circuits, making use of the symmetric property of the internal cells constituting the systolic arrays.

2. Implementation of RLS Algorithm in FPGA

The principle of adaptive array antennas is to weigh and combine signals of multiple antenna elements for the purpose of aiming the beams of an antenna in the directions of desired waves and forming nulls in the directions of interferences (**Figure 1**). An adaptive array antenna adopting a Minimum Mean Square Error (MMSE) algorithm [1] allows eliminating delayed waves without requiring information of the arrival direction of the desired waves; it is thus possible to achieve high-quality communication in multipath environments as well. Least Mean Square (LMS) and RLS algorithms are the most commonly used for weight optimization, and the RLS algorithm

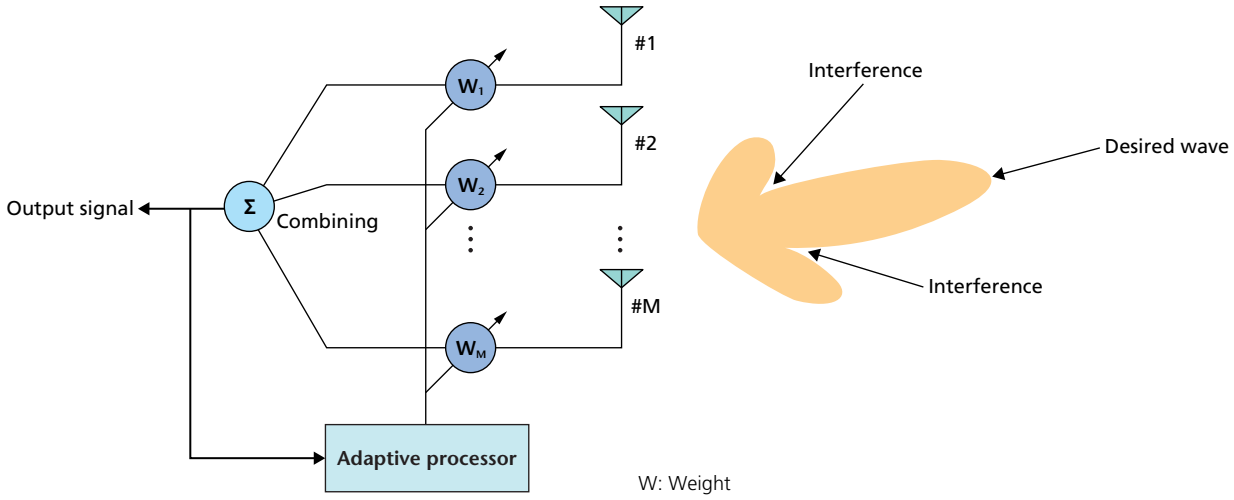


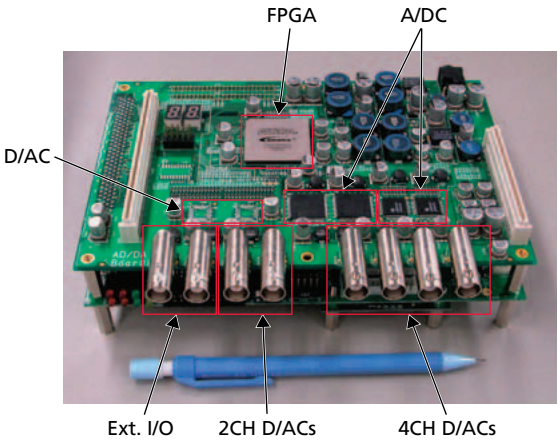
Figure 1 Principle of adaptive array antenna

is typically used in fading environments that change over time due to its fast convergence.

Photo 1 shows the signal processing hardware of the RLS adaptive array testbed, which consists of four elements implemented using FPGA. **Table 1** shows the specifications of the testbed; 69% (approximately 410,000 gates) of total area was required for the implementation in the FPGA board. **Figure 2** shows Bit Error Rate (BER) characteristics measured in an anechoic chamber when one desired wave and one interference wave come to the testbed simultaneously. A significant improvement can be observed compared to the case where only one antenna is used. Note that the difference between the computer simulation (theoretical values) and experimental data is caused by factors other than the signal processing (antenna elements, measurement jig, cables and so on). As the results of evaluations by using the testbed, we confirmed that the number of updates required before weight convergence was in the order of 10 to several tens, and the processing required for one weight update was approximately 400 clock cycles.

3. Implementation of Systolic Arrays

Systolic arrays represent a calculation architecture in which



A/DC: Analog to Digital Converter
D/AC: Digital to Analog Converter

Photo 1 Signal processing hardware

Table 1 Specifications of signal processing hardware

A/D converter	Number of channels	4
	Resolution	12 bits
	Sampling rate	40 MHz
D/A converter	Number of channels	2
	Resolution	14 bits
	Sampling rate	40 MHz
FPGA	System gates	600,000

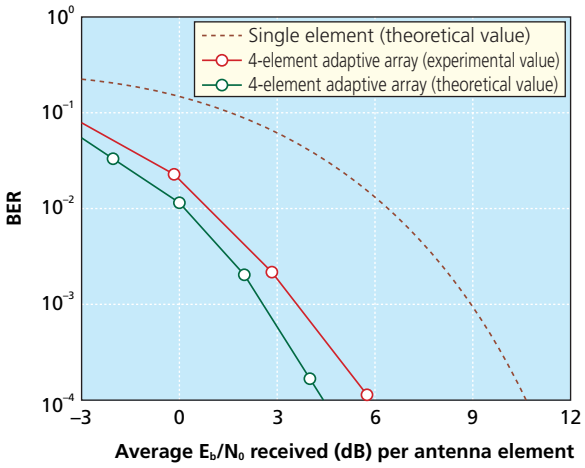


Figure 2 BER performance

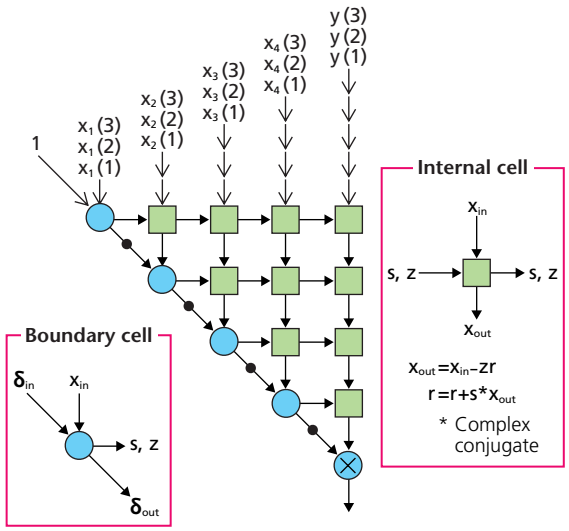


Figure 3 Architecture of RLS systolic array

calculations are performed in parallel by arranging circuits (cells) that perform individual, simple calculations in a regular pattern and pouring in the data required for the calculation in a pipelined manner. This parallel approach can improve the processing speed to a large extent and the uniform structure provides excellent expandability. Moreover, since the majority of the cells are only connected directly to adjacent cells and the data exchange is local, they are highly suited for implementation using Large Scale Integration circuits (LSI). **Figure 3** shows the architecture used for implementing the RLS algorithm using systolic arrays [3]. The figure shows a case with four array elements. $x_1(i)$, $x_2(i)$, $x_3(i)$ and $x_4(i)$ represent the i'th sample point of the signals of each element and y(i) represents the i'th sample point of the reference signal used for identifying the desired signal. The systolic arrays are mainly configured using two types of cells, boundary cells and internal cells. All

cells operate under the control of a single clock, and the data propagates through the cell arrays in synchronization with this clock, so that the processing is performed in a manner similar to the way blood circulates in a human body in synchronization with the contraction of the heart. Indeed, the word systolic is an adjective meaning "heart contraction." Each cell receives input data as indicated by the arrows in Fig. 3 and then performs a certain pre-determined processing. Each data point is a complex number, and two multiplications of complex numbers are included in the internal cell processing. **Figure 4** shows a specific circuit configuration of internal cells composed from adders, subtracters and multipliers. fpadd, fpsub and fpmul indicate adders, subtracters and multipliers, respectively. Complex number multiplication is resolved using real number calculation by splitting each complex number into its real and imaginary parts. If the number of antenna elements is M, the number of internal cells, as can be seen from Fig. 3, is M×(M+1)/2 and thus increases with the square of the number of elements M. Systolic arrays speed up the processing due to the parallel calculation, but have the issue that the circuit scale becomes extremely large if the number of elements is large. Multipliers and dividers consume the greater part of an area of digital circuits (circuit scale) and the possibility of making the circuit scale small thus depends on how the number of multipliers and dividers can be reduced. Note that, in general, multipliers and dividers are constructed using a number of adders proportional to the square of the number of digits of data handled (bit length), and they thus become very large compared to addition and subtracters. Moreover, multipliers and dividers also require significant processing time (delay) in digital circuits.

It can be seen that the internal cells shown in Fig. 4 have almost the same circuit block structure in the first half (left part) as in the latter half (right part). Subtracters can be replaced with adders by reversing the sign of the input data in advance. We thus used a selector circuit to branch between the processing of the first and latter halves, thereby reducing the circuit scale of the internal cells to approximately half of the original size. Since the processing of the latter half of the internal cells uses

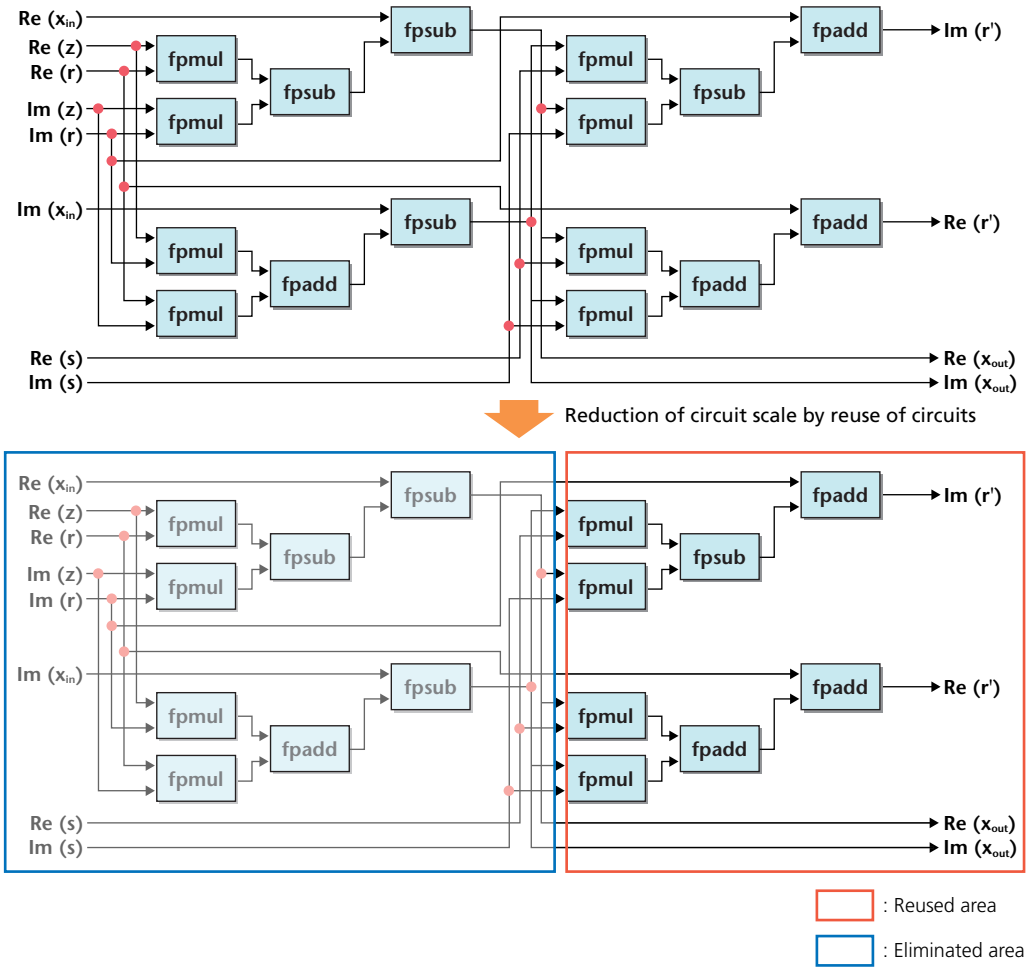


Figure 4 Internal cell circuit configuration

the processing result of the first half as input signal, it is possible to reduce the circuit scale into approximately half, almost entirely without impairing the processing speed.

4. Implementation Evaluation

Table 2 shows the result of implementing the RLS algorithm in FPGA using systolic arrays adopting the reusable configuration proposed above for the internal cells. For the cases of two and four antenna elements, the required number of sampling data processing clock cycles was 20, which indicates that the weights are updated using a very small number of clock cycles. The total numbers of clock cycles required before convergence for the two and four antenna element cases were 320 and 540, respectively. **Figure 5** shows a comparison between the beam patterns after convergence. As can be seen from the figure, a highly precise null was formed in the interference wave direction. The circuit scales were 440,000 gates and 1,130,000 gates, respectively. Conventionally, 19 Application-Specific Integrated Circuits (ASIC), each with 900,000 gates, were required in order to implement the RLS algorithm using systolic arrays for an antenna with 10 elements. With the proposed configuration, there is a strong chance of achieving the same performance using only 13 FPGAs (900,000 gates).

Table 2 Systolic array implementation results

Number of elements	Circuit scale	Number of clock cycles (per sample point)	Total number of clock cycles (for convergence)
2	440,000 gates	20	320
4	1,130,000 gates	20	540

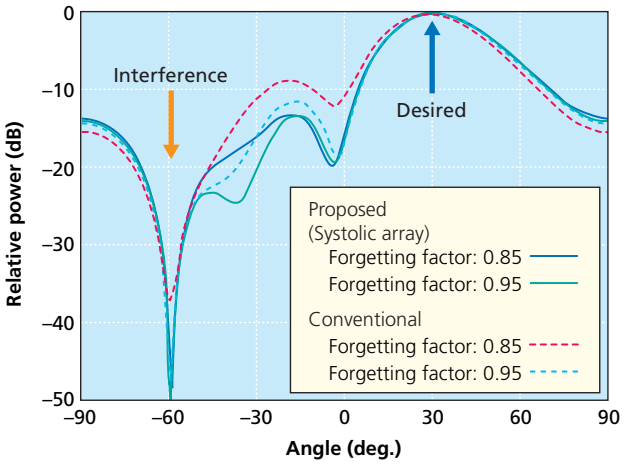


Figure 5 Beam patterns after convergence (4 elements)

5. Conclusion

This article described the implementation and evaluation result of the RLS algorithm using FPGA technology. The newly proposed systolic array architecture can reduce the circuit scale into half without impairing the processing speed. In the future, we intend to investigate antenna elements and array configurations, as well as examine application of Multiple Input Multiple Output (MIMO) technologies, for the purpose of expanding the transmission capacity.

REFERENCES

[1] N. Kikuma: "Adaptive Signal Processing with Array Antenna," Science and Technology Publishing Company, Inc., 1999 (in Japanese).
[2] S. Haykin: "Adaptive Filter Theory, 2nd Ed.," Prentice Hall, Englewood Cliffs, NJ, 1991.
[3] T. Asai and T.Matsumoto: "A Systolic Array RLS Processor," IEICE Trans.Commun, Vol. E84-B, No. 5, pp. 1356-1361, May 2001.

ABBREVIATIONS

ASIC: Application Specific Integrated Circuit
BER: Bit Error Rate
FPGA: Field Programmable Gate Array
LMS: Least Mean Square
LSI: Large Scale Integration circuit
MIMO: Multiple Input Multiple Output
MMSE: Minimum Mean Square Error
RLS: Recursive Least Square