FPGA IMPLEMENTATION OF RLS SYSTOLIC ARRAY

Yoshiaki YOKOYAMA[a)], Minseok KIM, and Hiroyuki ARAI
Department of Electrical and Computer Engineering, Yokohama National Univ.
79-5 Tokiwadai, Hodogaya-ku, Yokohama 240-8501, Japan.
a) E-mail: yokoyama@arailab.dnj.ynu.ac.jp

1 Introduction

The adaptive array antennas suppress the harmful effect of the interferences and delay components, which has been studied to realize the high speed wireless communication. MMSE (Minimum Mean Square Error) technique is widely used in the mobile radio communication system. In the MMSE adaptive array antenna, the weight is determined in order to minimize mean square error between the output signal combined with the signals of each antenna element and the reference signal known in advance at receiver. There are some well-known optimization techniques to obtain weight vector, RLS (Recursive Least Squares) algorithm is known as faster convergence property than LMS (Least Mean Square) algorithm, but the complexity increases in proportion to the square of the number of parameters to be estimated [1]. To overcome this problem, a systolic array processor, which computes the RLS algorithm using of high order of parallelism and pipeline architecture, has been implemented [2]. However, it is critical problem of huge circuit resource by 19 ASICs exhaustively. Hence it is necessary to examine carefully the circuit scale for the practical application.

This paper presents the FPGA (Field Programmable Gate Array) implementation of systolic array processor exploiting parallel pipeline processing of the RLS algorithm and the reduction of the circuit scale in the systolic array will be investigated.

2 FPGA Implementation of RLS Systolic Array

The square-root-free algorithm was derived for RLS systolic array [3]. The implementation of a 4-element RLS systolic array based on the square-root-free algorithm is shown in Fig. 1, where $\mathbf{x}$ and $y(n)$ denote the input vector and the reference signal at time n, respectively. Each cell within the basic triangular array stores trianglar matrix $\mathbf{R}(n)$. It is initialized to be zero before starting the least-square processing and then updated every clock cycle. Cells in the right-hand column store the vector $\mathbf{u}(n)$ which is also intialized to be zero and updated every clock cycle.

Each row of cells in the array performs a basic Givens rotation between the first row with the stored triangular matrix and a vector of the data received by the input data stream so that the leading element of the received vector is eliminated. The boundary cells compute the appropriate rotation parameters and passes them to the neighboring cell at the next clock cycle. In the internal cells, the similar rotation is applied subsequently to all other elements of the received data vector. Since the rotaion parameters are delayed by one clock cycle in each cell, it is necessary to impose corresponding time skew on the input data vectors. Having the next row of the array and so on. This architecture ensures that current triangular matrix $\mathbf{R}(n)$ is obtained from the previous one by applying the sequence of Givens rotations as each row data of $\mathbf{x}^H(n)$ of the matrix $\mathbf{X}$ goes down through the array. When all elements are flushed the triangular matrix $\mathbf{R}(n)$ is updated. Similarly the current vector $\mathbf{u}(n)$ is obtained by the sequence of Givens rotations from the previously one as each element of the vector $\mathbf{y}$ goes down through the cells in the right-hand column of the array.

ISBN: 89-86522-78-0 94460©KEES

Figure 2 shows an input data configuration for extracting the weight vector. The corresponding estimation error to be obtained as the RLS systolic array output is

$$e(n) = y(n) - \mathbf{w}^H(n)\mathbf{x}(n), \tag{1}$$

where $\mathbf{w}(n)$ is the weight vector. Input vector consists of zeros, except for the i-th element seting as unity

$$\mathbf{x}^H(n) = [0, \ldots, 0, 1, 0, \ldots, 0], \tag{2}$$
$$y(n) = 0. \tag{3}$$

The estimation error can be expressed as

$$e(n) = -\omega^*{}_i(n), \tag{4}$$

thus the i-th weight appear as the RLS systolic array output. To extract the entire weight vector, the updating process of all the stored values should be suspended, and input the identity matrix whose diagonal and off-diagonal terms are unity and zeros, respectively, as shown in Fig. 2. Additionally, it is necessary to set the input parameter δ_{in} in final cell by unity.

Configuration of implemented boundary and internal cell is shown in Figs. 3 and 4. According to IEEE-754 standard, 22-bit floating point arithmetic was used for internal arithmetic. Since the circuit scale becomes usually huge in systolic array processor, it is possible to reduce the circuit scale of the internal cell by about 1/2 eliminating redundancy from the symmetry of the circuit as shown in Fig. 5. RLS systolic array in M elements needs M boundary and $1/2M(M+1)$ internal cells, respectively. The proposed circuit can saves around $2.7M(M+1) \times 10^5$ gates in the FPGA. In ref. [2], RLS systolic array with up to 10 parameters has been developed with 19 ASICs, each having $900,000$ gates, however proposed system can reduce the circuit scale to 13 ASICs and can save 6 ASICs.

3 Performance evaluation

The accuracy of the RLS systolic array with 22-bit(sing:1bit, exponent:6bit, mantissa:15bit) floating point arithmetic was compared with that of floating point arithmetic of double precision. The simulation parameters are shown in Table 1 and the results are presented in Figs. 6 and 7. The simulation results confirmed that proposed system with 22-bit floating point arithmetic had comparable performance to double precision floating point arithmetic processor. The result on estimated gate count and computation performance in the 2- and 4-element configurations is presented in Table 2. Maximum operating frequency was about 14 MHz in both 2- and 4-element cases. The weight component is updated by 20 clock cycles, because it is independent of the number of elements. The weight vector becomes 20 clock cycles in 2-element case and 80 clock cycles in 4-element case, which depends on the number of elements because additional clock cycles are required for weight flushing. From Fig. 6, it can be seen that the error level converges after around 15 steps, and the convergence time can be roughly estimated by 39μs because totally 540 clock cycles are required in 4-element case.

This system uses the floating point division operation which leads to limit the maximum operating frequency to 14 MHz. It may be improved by enhancing pipeline architecture but required number of clock cycles usually increase. In addition replacing exhaustive the floating point division operation by the CORDIC (COordinate Rotation DIgital Computer) algorithm will be another solution in the future.

4 Conclusion

In this paper, we described the FPGA implementation of systolic RLS processor considering the reduction of the circuit scale eliminating redundancy from the symmetry of the circuit.

The synthesis results were 440K and 1,130K equivalent gates in case of 2- and 4-element cases, respectively. The estimated convergence time after 15 iterations was about 39μs.

Acknowledgment

The authors wouuld like thank Dr.Kenzo Cho, NTT Docomo Inc. for his helpful discussion.

References

[1] N. Kikuma, *Adaptive Signal Processing with Array Antenna*, Science and Technology Publishing Company, Inc., 1999 (in Japanese).

[2] T. Asai, T. Matsumoto, "A Systolic Array RLS Processor," *IEICE Trans. Commun*, Vol.E84-B, No.5, pp.1356–1361, May 2001.

[3] S. Haykin, J. Litva, T. Shepherd, *Radar Array Processing*, Springer-Verlag, pp.153–247, 1993.

Table 1: Simulation parameters.

Antennas	4 elements
Array topology	linear array equi-spaced by half wavelength
Number of incident signals	2 (uncorrelated)
DOAs	30°(User), -60°(Interferer)
SNR	same power of 20 [dB]
Fogetting factor	$\lambda = 0.875$
Iterations	100

Table 2: Processor performance.

	Estimated gate count	Required Number of clock cycles	Maximum operation clock freq.	Convergence time
2-element	440,000	20 clks / sample	14 [MHz]	23 [μs]
4-element	1,130,000	20 clks / sample	14 [MHz]	39 [μs]

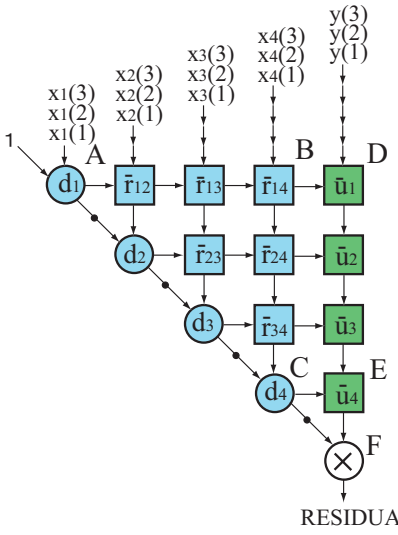

Figure 1: Block diagram of RLS systolic array.

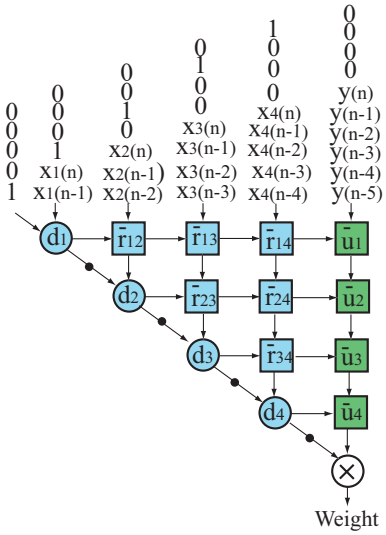

Figure 2: Serial weight flushing.

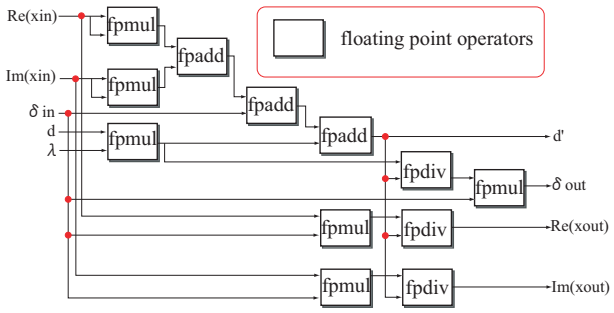

Figure 3: Configuration of boundary cell.

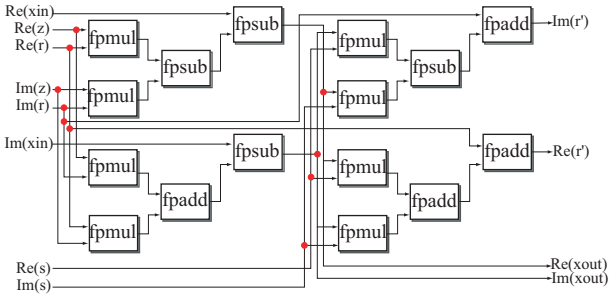

Figure 4: Configuration of internal cell.

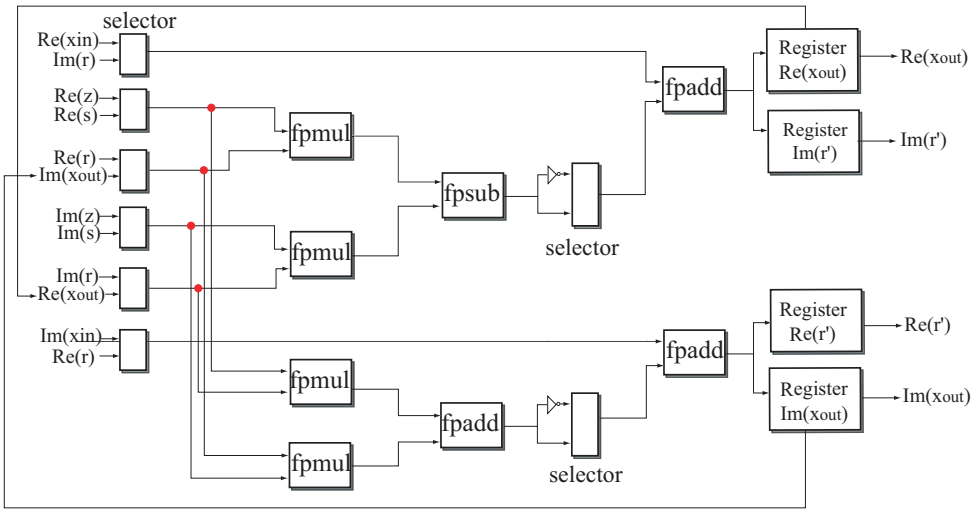

Figure 5: Improvement of internal cell.

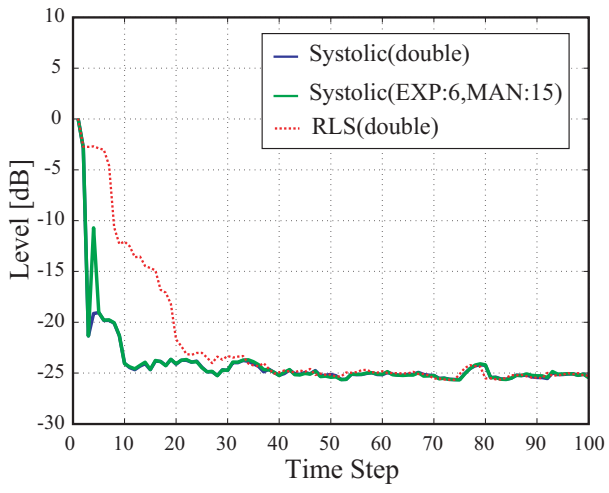

Figure 6: Comparison of convergence performance.

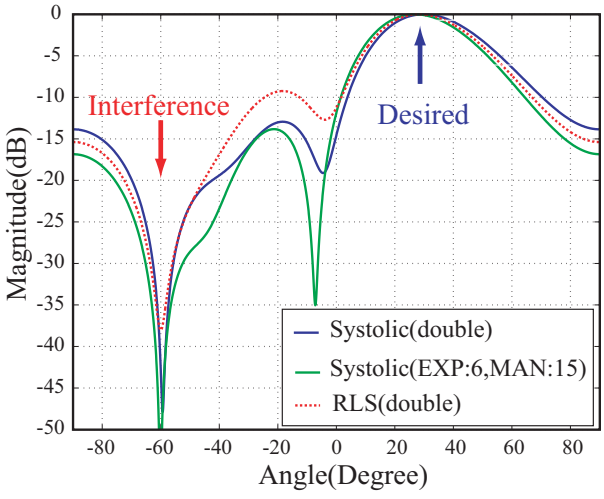

Figure 7: Resulting beampatterns.