POZNAN UNIVERSITY OF TECHNOLOGY ACADEMIC JOURNALS
No 80 Electrical Engineering 2014

Robert SMYK*
Maciej CZYŻAK*

ON CONFIGURATION OF RESIDUE SCALING PROCESS IN PIPELINED RADIX-4 MQRNS FFT PROCESSOR

Residue scaling is needed in pipelined *FFT* radix-4 processors based on the Modified Quadratic Residue Number System (*MQRNS*) at the output of each butterfly. Such processor uses serial connection of radix-4 butterflies. Each butterfly comprises *n* subunits, one for each modulus of the *RNS* base and generates four complex residue numbers. In order to prevent the arithmetic overflow in the succesive stage, every number has to be scaled, *i.e.* divided by a certain constant. The dynamic range of the processed signal increases due to the summation within the butterfly and the transformation of coefficients of the *FFT* algorithm to integers. The direct approach would require eight residue scalers that would be highly ineffective regarding that such a set of scalers had to be placed after each butterfly. We show and analyze a structure which uses parallel-to-serial transformation of groups of numbers so that only two scalers are needed.

KEYWORDS: Fast Fourier Transform, residue number system, modified quadratic residue number system, *FFT* pipelined processor

1. INTRODUCTION

The calculation of the Discrete Fourier Transform (*DFT*) is usually performed with one of the Fast Fourier Transform algorithms [1]. As the computing environment serve general purpose computers, digital signal processors or specialised *FFT* processors. These processors are used when high requrements are imposed with respect to the time of calculation. That may be as short as single microseconds. If there is a constant inflow of the input signal, the pipelined mode of operation can be applied, because pipelining mechanisms provide high data throughput. However, the sophisticated control of data flow between stages of the *FFT* processor is needed. The form of this control may depend on the type of arithmetic that is used and the given form of the *FFT* algorithm.

The *FFT* processors may use fixed point arithmetic, block floating point arithmetic [3] or Residue Number Systems (*RNS*) [2]. The *RNS* allows to attain the highest degree of granulation of the *VLSI* circuit. This also means that highest

* Gdansk University of Technology.

frequencies of operation can be achieved due to the fact that within the architecture of the processor there are no blocks that would enforce the reduction of pipelining rate. However, the *RNS FFT* processor requires certain number of processing channels and scalers between the butterflies. In particular, the Modified Quadratic Residue Number System (*MQRNS*) [4, 5] is used because it allows to implement complex multiplication using only three real multiplications.

When the *RNS* is used, the scaling operation is needed because the *RNS* is an integer number system but *FFT* algorithm coefficients are fractional numbers. For this reason they have to be transformed to integers by a multiplication by a suitable constant K and rounded off. After each multiplication the product has to be scaled by K in order to avoid the arithmetic overflow. This takes place when the number range of the *RNS* is fully utilized. The scaling operation is fairly complex with regard to the needed hardware amount. At the output of the *FFT* processor butterfly in dependence of the type of algorithm, usually radix-2 or radix-4, there are two or four complex numbers which must be scaled. In the direct approach four or eight residue scalers would be required that seems to be highly ineffective because of the hardware amount needed. In this paper we show that using an intermediary structure that delays and multiplexes the radix-4 butterfly output numbers, the number of scalers can be brought down to two .

2. TYPICAL FFT PIPELINED STRUCTURES

A typical pipelined *FFT* processor consists of a number of butterfly stages that implement butterfly operations. For $N = 1024$, for radix-2 ten stages are needed and five stages for radix-4. The form of butterflies and their interconnections determine the structure of the *FFT* processor. The butterfly input data has to properly ordered before the butterfly computations may start. This reordering is performed using commutators. Commutators provide for the proper connection between the subsequent butterfly stages in the pipeline. The form of these commutators influences the structure of the *FFT* processor. There are three most typical hardware commutator architectures used in the pipelined *FFT* processors: Multi-path Delay Commutator (*MDC*), Single-path Delay Feedback (*SDF*) and Single-path Delay Commutator (*SDC*) [6, 7]. The single-path architectures can process two data signals being the real and imaginary part of the signal, while multi-path architectures can process N data signals at a time.

The *MDC* is the most classical approach to pipeline implementation of radix-2 and radix-4 *FFT*s (Fig. 1). Here the input sequence is broken into several parallel data streams flowing forward. Input signal samples flowing to the radix-4 butterfly (*BF4*) must be in a proper order and scheduled by proper delays. In the radix-2 *FFT* the butterflies and the multipliers have 50% utilization and in radix-4 *FFT*

utilization is 25% of all components. This can be compensated only in some special applications where four *FFT*s are being processed simultaneously.

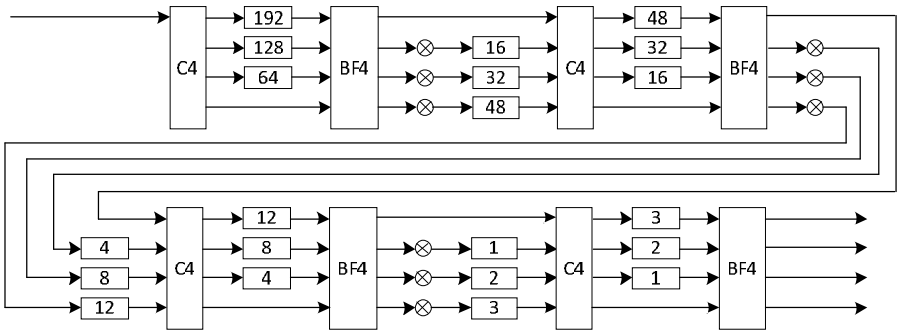

Fig. 1. The block diagram of *R4MDC*

Table 1. Comparison of required hardware for commutator schemes in pipelined *FFT*

The pipeline scheme	multipliers	butterflies	registers
R2MDC	$\log_2(N-2)$	$\log_2 N$	$3/2N$
R2SDF	$\log_2(N-1)$	$\log_2 N$	$N-1$
R4SDF	$\log_4(N-1)$	$\log_4 N$	$N-1$
R4MDC	$3\log_4 N$	$\log_4 N$	$5/2N-4$
R4SDC	$\log_4(N-1)$	$\log_4 N$	$2(N-1)$

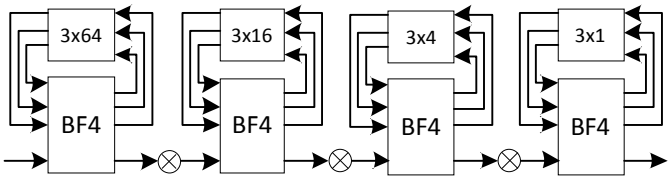

Fig. 2. The block diagram of *R4SDF*

Among several pipelined *FFT* architectures, the *R4SDC* is widely used, owing to its high utilization of multipliers, butterfly elements and memory blocks. A pipelined *N*-point radix-4 *FFT* processor based on this architecture, shown in Fig. 3, has $\log_4 N$ stages. Each stage produces one output number within each word cycle. Each stage contains a commutator (*C4*), a butterfly (*BF4*) element and a complex multiplier. The sequential outputs at each stage should be ordered. Here

the butterfly element performs only the summation that can be realized by six programmable adder/subtractors with an additional control circuit. Three complex adder/subtractors (each comprising of a real and an imaginary element) are used instead of eight complex adders. Control signals can be stored in *LUT*s (Look-Up Table), that selects the data fed into add/subs modules. This butterfly architecture generates *N* outputs consecutively in *N* word cycles, compared to the *R4MDC* butterfly which generates *N* outputs in *N*/4 word cycles, with *N*/3 word cycles idle.

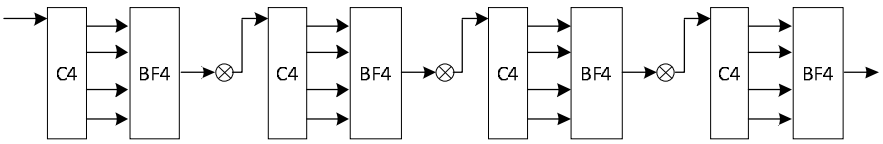

Fig. 3. The block diagram of *R4SDC*

In the *R4SDF* the utilization of multipliers is on the level of 75% by storing three *BF*4 outputs. In the radix-4 butterfly utilization is only 25% and the butterfly is fairly complicated because of utilization of at least 8 complex adders.

It could be concluded, that delay feedback solutions are typically more efficient in classical *FFT* processors than those using delay-commutators in terms of memory utilization, since butterfly outputs share inputs with storage elements.

3. GENERAL STRUCTURE OF RADIX-4 FFT PIPELINED PROCESSOR BASED ON MQRNS

The radix-4 *FFT* processor that uses the *MQRNS* consists of $\log_4 N$ stages, where each stage comprises *n* butterflies, one for each modulus of the *RNS* base. The flow of *MQRNS* calculations has been given previously in [5]. As stated above after each stage the residue scaling is needed. This scaling is performed in a block of residue scalers [8]. As each butterfly generates four residue numbers in the Complex Residue Number System (*CRNS*) form in total eight real residue numbers have to be scaled. The direct approach would require eight residue scalers after each stage of butterflies. This would lead to excessive number of scalers (in this case 40 scalers), that would make the implementation of the processor impractical. This results from the fact that a single scaler is a fairly complex circuit. The number of scalers can be reduced by applying the special switching matrix (Fig. 5) in which the butterfly outputs are suitably delayed, so that only two scalers instead of eight can be used.

At an instatnt t_0 eight real residue numbers emerge at the output of the each butterfly. The numbers are stored in the first stage of registers. In the next cycle they are written to the next level. After four cycles two numbers from Re_0 and Im_0

registers are directed using the multiplexer *MUX*1 and *MUX*2 to the respective inputs of the scalers *SCALER*1 and *SCALER*2. During the next cycles the properly delayed data enters the scalers.

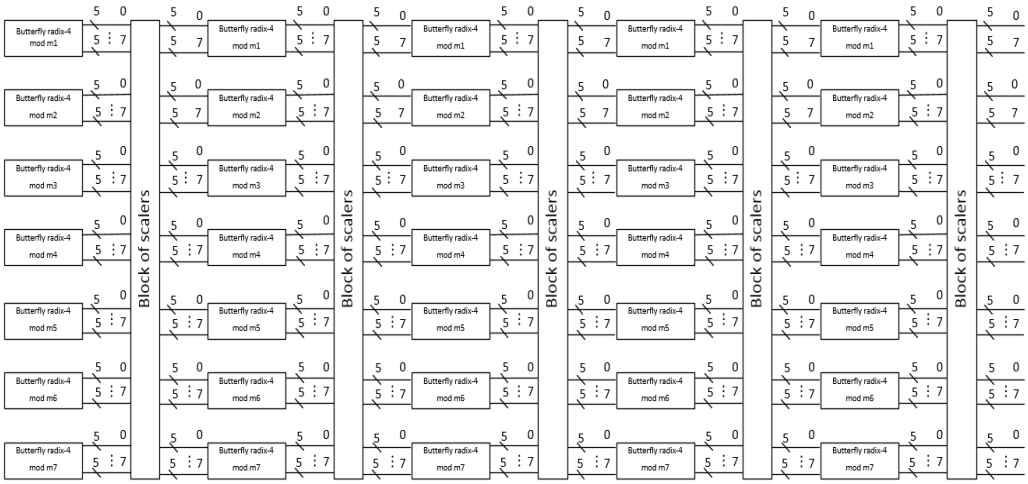

Fig. 4. The block diagram of radix-4 *MQRNS FFT* processor

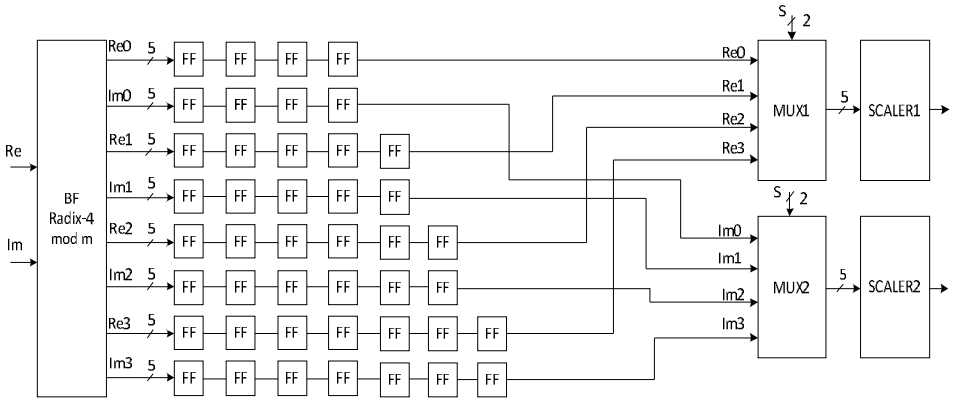

Fig. 5. The structure of the parallel-to-serial converter at the butterfly output

SUMMARY

We have presented an approach to the problem of reduction of the number of scalers within the radix-4 *FFT MQRNS* processor. The use of the *MQRNS* allows to attain high pipelining rates limited by the latch delay or full adder delay. However, the price paid is the need to scale residue numbers after each stage of butterflies. As there are eight residue numbers to be scaled, in the direct approach eight scalers

after each stage would be needed. This would render this solution impractical. We propose an intermediary structure between butterfly stages that by a suitable redirection and multiplexing allows to use only two scalers after each stage. This at the cost of introducing certain delay which is, however, not meaningful as compared to the overall delay of the pipeline.

REFERENCES

[1] Oppenheim A.V., Schafer R.W., Discrete-Time Signal Processing, Third Edition, Prentice-Hall, 2009.

[2] Soderstrand M.A. *et al.*, Residue Number System Arithmetic: Modern Applications in Digital Signal Processing, IEEE Press, Piscataway, NJ, 1986.

[3] Rabiner L.R., Gold B., Theory and Application of Digital Signal Processing. Prentice-Hall 1975.

[4] Krishnan R., Jullien G.A., Miller W.C., The modified quadratic residue number system (MQRNS) for complex high-speed signal processing, IEEE Transactions on Circuits and Systems, Volume 33, Number 3, Pages 325-327, 1986.

[5] Czyżak M., Smyk R., Radix-4 DFT butterfly realization with the use of the modified quadratic residue number system, Poznan University of Technology Academic Journals, Electrical Engineering, Number 63, Pages 39–51, 2010.

[6] Shousheng H., Torkelson M., A new approach to pipeline FFT processor, Proceedings of the 10th International Parallel Processing Symposium IPPS '96, Pages 766-770, 1996.

[7] Wold E., Despain A., Pipeline and parallel-pipeline FFT processors for VLSI implementations, IEEE Transactions on Computers, Volume C-33, Number 5, Pages 414-426, May 1984.

[8] Czyżak M., Smyk R., Ulman Z., Pipelined scaling of signed residue numbers with the mixed-radix conversion in the programmable gate array. Poznan University of Technology Academic Journals. Electrical Engineering, Number 76, Pages 89–99, 2013.