Abstract—In this paper, we propose a new architecture for the implementation of the N-point Fast Fourier Transform (FFT), based on the Radix-2 Decimation in Frequency algorithm. This architecture is based on a pipeline circuit that can process a stream of samples and produce two FFT transform samples every clock cycle. Compared to existing implementations the architecture proposed achieves double processing speed using the same circuit complexity.

Keywords—Digital signal processing, systolic circuits, FFT algorithm.

I. INTRODUCTION

Modern telecommunication systems are based more than ever before on digital signal processing. High-speed digital telecommunication systems such as OFDM (Orthogonal Frequency Division Multiplexing) and DSL (Digital Subscriber Lines) need real-time high-speed computation of the Fast Fourier Transform.

Many different hardware architectures have been proposed for the implementation of FFT algorithms [1-8]. A first approach for these implementations concerns time non-critical applications and has small hardware requirements, but it needs a significantly large number of clock cycles to compute a full FFT. For example, in [1] one butterfly unit is used for all computations and N+Nlog2N clock cycles are required for the computation of the FFT. A second implementation approach is for speed demanding applications, where one butterfly unit is used for each decimation stage of a radix-2 FFT. A pipeline architecture based on the constant geometry radix-2 FFT algorithm, which uses log2N complex-number multipliers (more precisely butterfly units) and is capable of computing a full N-point FFT in N/2 clock cycles, using N.log2N complex multipliers, 2N.log2N hardware utilization and can compute the N-point FFT in N/2 clock cycles, using N.log2N complex multipliers, 2N.log2N complex adders and N+2log2N-2 delay elements.

II. ARCHITECTURE FOR RADIX-2 DECIMATION IN FREQUENCY FFT

The N/2 butterfly operations included in every stage can be performed, sequentially, by the same Butterfly Unit. Because the two inputs of a Butterfly Unit of a stage are generated from the output of the Butterfly Unit of the previous stage at different time points, and the two inputs of a Butterfly Unit of a stage are obtained from either the upper or the lower output of the Butterfly Unit of the previous stage, a Shuffling Unit is inserted between two successive Butterfly Units in order to route these outputs to the corresponding inputs.

The block diagram of the implementation proposed for the N-point DIF FFT, where N is a power of two, is shown in Fig. 1. This circuit is an expansion of the circuit of Fig. 4, since additional are added in order to implement the additional stages of the FFT. The circuit consists of the FFT Processor, which performs the operations needed for the computation of the FFT (Butterfly and Shuffling Units) and two adaptation units; the Input Adaptation Unit, which converts the input steam of samples to two separate ones in order to provide the inputs of the FFT Processor, and the Output Adaptation Unit, which converts the two streams of outputs of the FFT Processor to one stream of samples in natural order.

The FFT Processor, in turn, consists of n Butterfly Units, one for each stage of Fig. 1 and n-1 Shuffling Units, where n=2^N. The Shuffling Units are responsible for the rearrangement of the data between two successive Butterfly...
The Shuffling Units are implemented using the circuit shown in Fig. 2. This circuit consists of $2w$ delay elements for the synchronization of the outputs of the previous Butterfly Unit and two multiplexers that perform the routing of the outputs to the corresponding inputs of the next Butterfly Unit.

The control signal $C$ of the multiplexers takes the value “0” for another $w$ clock cycles so that the lower output of the Butterfly Unit of stage $s$, which has already been delayed $w$ clock cycles is routed to the lower input of the Butterfly Unit of stage $s+1$ while at the same time, the first step of the procedure described above is repeated. For each stage $s$ this procedure is repeated $N/2$ times. If $f$ is the operating frequency of the circuit then the frequency of the control signal is $f/2w$ and the latency of the Shuffling Unit is $w$.

Most modern communication systems need the computation of the FFT of a stream of samples in natural order ($\ldots xN \ldots x1x0$). Therefore, an Input Adaptation Unit is needed so the architecture proposed can process such a stream of samples. Furthermore, an Output Adaptation Unit is needed in order to convert the output sequence of the last Butterfly Unit, which is given in bit-reverse order, to one stream of samples in natural order. The reverse operation for the formation of the output stream of samples is performed by the Output Adaptation Unit. Detailed memory-based implementations of these circuits are included in final form of the paper.
I. BUTTERFLY UNIT

To obtain high-speed operation parallel bit-level pipelined implementation is adopted.

Also bit-skew number format is selected in order to avoid arithmetic circuits for numbers in Redundant representation, which are in general more complex. In addition, such a representation will double the hardware of the Shuffling Units.

Let us consider the individual parts of the Butterfly Unit (B.U.) of Fig.3. The bit-skew form of a number \( x \) is represented by the symbol \( \hat{x} \). In Fig.4 the implementation of a bit-skew adder is shown. By inserting inverters at the inputs \( u \) and \( \text{cin} \) a similar circuit can be used for subtraction. The inputs \( d \) and \( u \) and the outputs \( (d+u, d-u) \) of the adder and the subtracter are all in bit-skew form.

The outputs are obtained with one clock cycle latency. In general the inputs \((u,d)\) and the outputs \((U,D)\) of the B.U. are complex numbers. For a more elegant presentation, the concept of complex binary digits is used [9].

The complex bit-skew form output of the subtracter enter to a bit-level pipelined carry-save complex array multiplier and the result is also in complex bit-skew form. Array multiplier form is selected instead of a Wallace Tree because of its canonical structure permitting higher operation speed. Detailed design (Full-Adder level pipelined and two’s complement number format) of this multiplier will be included in the final form of the paper. There is no need to de-skew the B.U. outputs since Shuffling Units can handle the numbers directly in bit-skew form. Notice that the switches of the Shuffling Units must be control by similar bit-skew signal that can be easily produced using a shift register inside each unit.

I. PERFORMANCE

If the Butterfly Unit is implemented using a complex bit-skew number carry-save array pipeline multiplier [5] instead of combinational then the circuit of Fig. 1 has an additional latency of \( r \) clock cycles, equal to the number of the pipeline stages, for every Butterfly Unit. In this case, additional \( r \) delay elements must be added to the upper line of the butterfly unit, as shown in Fig. 5, in order to synchronize the output of the complex multiplier with the output of the upper adder.

As can be seen in Fig. 1, for the implementation of DIF FFT Processor when using pipeline multipliers, every Shuffling Unit is preceded by a Butterfly Unit. In this case, the delay elements of each Shuffling Unit are preceded by the multiplier of the Butterfly Unit, as is also shown in the figure. Therefore, the latency of the multiplier can play the role of some of the \( w \) delay elements of the Shuffling Unit.

With this architecture a 100% hardware use is achieved, since all the Shuffling and Butterfly Units are utilized in every clock cycle. Therefore, this circuit can process two samples per clock cycle and compute a full N-point FFT in N/2 clock cycles. Since the circuit is fully pipelined, we can provide the samples of a new FFT right after the last sample (\( x_{N-1} \)) of the previous FFT and get the results right after the last result (\( X_{N-1} \)) of the previous FFT.

TABLE I illustrates the characteristics, hardware usage and performance, of various implementations of the FFT. The
The architecture proposed has the same speed performance with the architecture in [2] but requires significantly less delay elements (N-2 instead of Nlog2N). In addition, the latency of the circuit of the architecture proposed is significantly smaller since the architecture in [2] is based on the constant geometry radix-2 FFT algorithm. Furthermore, by using less hardware than [3] and almost the same hardware with [1] it can compute the FFT in half the clock cycles needed in [1] and [3], since 100% hardware use is achieved. Consequently, the architecture proposed yields a double speed performance having almost half the latency.

A great advantage of the architecture proposed for the DIF FFT is that it can operate with pipeline instead of combinational multipliers, with a slight impact on the latency of the circuit. For N=1024, in the architecture in [1] and [3] the latencies are 1033 and 1035 clock cycles respectively. For implementation with pipeline multipliers the latencies become 1133 and 1135 clock cycles respectively. In the architecture proposed, the corresponding latencies are 521 and 556 clock cycles.

II. CONCLUSIONS

A novel architecture has been proposed for the efficient implementation of the N-point FFT, where N is a power of two, based on the Radix-2 Decimation In Frequency (DIF). The circuit requires log2N Butterfly Units, namely log2N complex-number multipliers and 2.log2N complex-number adders, and N+2log2N-2 delay elements. It is capable of processing two samples every clock cycle, therefore it requires N/2 clock cycles for the computation of the N-point FFT. This architecture is suitable for applications where computation of the FFT on a high bit-rate stream of data is required. The operation speed is limited by the delay of a gated Full-Adder. In addition, a special bit-skew array pipeline multiplier that is efficient in terms of latency and delay elements is proposed. Our research team prepares on Synopsys design tool an FFT core ASIC implementation (~ 4M Transistors) for GHz signals real-time spectral analysis. Results will be given in the final form of the paper.

REFERENCES