A Normal I/O Order Radix-2 FFT Architecture to Process Twin Data Streams for MIMO

Abstract:
Nowadays, many applications require simultaneous computation of multiple independent fast Fourier transform (FFT) operations with their outputs in natural order. Therefore, this brief presents a novel pipelined FFT processor for the FFT computation of two independent data streams. The proposed architecture is based on the multipath delay commutator FFT architecture. It has an N/2-point decimation in time FFT and an N/2-point decimation in frequency FFT to process the odd and even samples of two data streams separately. The main feature of the architecture is that the bit reversal operation is performed by the architecture itself, so the outputs are generated in normal order without any dedicated bit reversal circuit. The bit reversal operation is performed by the shift registers in the FFT architecture by interleaving the data. Therefore, the proposed architecture requires a lower number of registers and has high throughput. The proposed architecture of this paper analysis the logic size, area and power consumption using Xilinx 14.2.

Enhancement of the project:

Existing System:
There are FFT architectures, which can handle multiple independent data streams. However, all the data streams are processed by a single FFT processor. In four independent data streams are processed one by one. Similarly, eight data streams are processed at two domains. Thus, the outputs of multiple data streams are not available in parallel. In order to simultaneously process the data streams, more than one FFT processors need to be employed. In one to four data streams are processed using multiple data paths for wireless local area network application. Data of different data streams are interleaved to process them simultaneously. In low complexity FFT
architectures are proposed but these architectures can process only real-valued signals (signals only with real part). Moreover, they generate two outputs per clock cycle and these outputs are not in natural order. Thus, most of the recent architectures require bit reversal structures to generate the outputs in natural order.

Disadvantages:

- Performance is low
- Usage of hardware element is high

Proposed System:

The idea of computing an N-point FFT using two N/2-point FFT operations with additional one stage of butterfly operations is shown in Fig. 1, which is not the exact architecture but provides the methodology. The reordering blocks in Fig. 1 are merely present to state that the N/2 odd samples (x(2n + 1)) are reordered before the N/2-point DIT FFT operation and N/2 even samples (x(2n)) are reordered after the N/2-point DIF FFT operation. In order to compute the N-point DIT FFT from the outputs of two N/2-point FFTs, additional one stage of butterfly operations are performed on the results of the two FFTs. Thus, the outputs generated by additional butterfly stage are in natural order.
Fig. 1. Idea of the proposed method.

For the purpose of simplicity, the proposed 16-point FFT architecture in Fig. 2 is explained. It has two eight-point MDC FFT architectures to process two data streams. The delay commutator units present at the left side of SW1 dissociate the odd and even samples. The shift registers in the delay commutator units, which receive inputs, are used to bit reverse the odd input samples. These shift registers are called reordering shift registers (RSRs).

Fig. 2. Proposed 16-point radix-2 FFT architecture with outputs in natural order.

Instead of using radix-2 FFTs, as shown in Fig. 2, any higher radix FFTs architecture can be used. In Fig. 3, two radix-23 64-point FFTs are used to realize 128-point FFT whose multiplier complexity is $4(\log_8(N/2) - .5)$ and working is almost the same as the 16-point FFT. The multiplier complexity of N-point radix-k FFT algorithm is $4(\log_k (N/2) - .5)$.
Fig. 3. Proposed 128-point radix-23 FFT architecture.

**Advantages:**

- throughput high
- high performance
- reduce the usage of hardware element

**Software implementation:**

- Modelsim
- Xilinx ISE