4

FFT ARCHITECTURES

4.1 General

Not only have several variations of the FFT algorithm been developed after the Cooley-Tukey’s publication but also various implementations. Generally, the FFT can be implemented in software, general-purpose digital signal processors, application specific processors or algorithm-specific processors.

The implementations with software on general-purpose computer can be found in literature and still being explored in some projects, for instance, the FFTW project in the Laboratory for Computer Science at MIT [74]. Software implementations are not suitable for our target application as the power consumption is too high.

Since it is hard to summarize all other implementations, we will concentrate on algorithmic-specific architectures and only give a brief overview on some FFT architectures.

4.2 General-Purpose Programmable DSP Processors

Many commercial programmable DSP processors include the special instructions for the FFT computation. Although the performance varies from one to another, most of them belong to the Harvard architecture from the architecture point of view. A processor with Harvard architecture has separate busses for data and control.

A typical programmable DSP processor has on chip data and program memory, address generator, program control, MAC, ALU, and I/O interfaces, as illustrated in Fig. 4.1.
The computation of FFT with general-purpose DSP processor does not differ too much from the software computation of FFT in a general-purpose computer.

To compute the FFT with a general-purpose DSP processor requires three steps: first the data input, then the FFT/IFFT computation, and finally the data output. In some DSP processors, for instance TI’s TMS320C3x, bit-reverse addressing is available to accelerate the unscrambling for the data output. Typical FFT/IFFT execution times are about 1 ms [75] [76] [77], which is far from the implementation using more specialized implementations. The implementation with general-purpose programmable DSP processor is therefore not applicable due to the throughput requirement.

4.3 Programmable FFT Specific Processors

Several programmable FFT processors have been developed for the FFT/IFFT computations. These processors are 5 to 10 times faster than the general-purpose programmable DSP processors.

The programmable FFT-specific processors have specific butterflies and at least one complex multiplier [78]. The butterfly is usually radix-2 or radix-4. There is often...
an on-chip coefficient ROM, which stores sinus and cosine coefficients. This type of programmable FFT-specific processors are often provided with windowing functions in either time or frequency domain.

The Zarlink’s (former Plessey) PDSP16515A processor performs decimation in time, radix 4, forward or inverse Fast Fourier Transforms [78]. Data are loaded into an internal workspace RAM in normal sequential order, processed, and then read-out in correct order. The processor has two internal workspace RAMs, one output buffer, and one coefficient ROM.

![Figure 4.2 FFT-specific processor PDSP16515A.](image)

Although the PDSP1615A processor accelerates the FFT computation, it is still hard to meet the throughput requirement with a single processor due to the slow I/O. The processor requires 98 ms to perform 1024-point FFT with a system clock of 40 MHz. Using multiple processor configuration can achieve a higher throughput, but the power consumption is then substantially higher.
A recent released FFT specific processor from DoubleBW systems B. V. has higher throughput (100 Msamples/s) [79], but consumes 8 W at 3.3 V.

### 4.4 Algorithm-Specific Processors

Non programmable algorithm-specific processors can also be designed for the computation of FFT algorithms. The processors are designed mostly for fixed-length FFTs. The architecture of an algorithmic-specific FFT processor is therefore optimized with respect to memory structure, control units, and processing elements.

There are mainly three types of algorithm-specific processors: fully parallel FFT processors, column FFT processors, and pipelined FFT processors.

All three types of algorithm-specific processors represent different mapping of the signal-flow graph for FFT to hardware structures. The hardware structure in a fully parallel FFT processor is an isomorphic mapping of the signal-flow graph [80]. For example, the signal-flow graph for an 8-point FFT algorithm is shown in Fig.4.3. The 8-point fully parallel FFT processor requires 24 complex adders and 5 complex multipliers. The hardware requirement is excessive, and, hence, is not power efficient.

![Figure 4.3 Signal-flow graph for an 8-point FFT.](image)
To reduce the hardware complexity, a column or a pipelined FFT processor can be used. A set of process elements in a column FFT processor [21] compute one stage at a time. The results are fed back to the same set of process elements to compute the next stage. For the long transform length, the routing for the processing elements is complex and difficult.

For a pipelined FFT processor, each stage has its own set of processing elements. All the stages are computed as soon as data are available. Pipelined FFT processors have features like simplicity, modularity and high throughput. These features are important for real-time, in-place applications where the input data often arrive in a natural sequential order. We therefore select the pipeline architecture for our FFT processor implementation.

The most common groups of the pipelined FFT architecture are

- Radix-2 multipath delay commutator (R2MDC)
- Radix-2 single-path delay feedback (R2SDC)
- Radix-4 multipath delay commutator (R4MDC)
- Radix-4 single-path delay commutator (R4SDC)
- Radix-4 single-path delay feedback (R4SDF)
- Radix-2² single-path delay commutator (R2²SDC)

We will discuss these pipeline architectures in more detail.

4.4.1 Radix-2 Multipath Delay Commutator

The Radix-2 Multipath Delay Commutator (R2MDC) architecture is the most straightforward approach to implement the radix-2 FFT algorithm using a pipeline architecture [82]. An 8-point R2MDC FFT is shown in Fig. 4.4.
When a new frame arrives, the first four input data are multiplexed to the top-left delay elements in the figure and the next four input data directly to the butterfly. In this way the first input data is delayed by four samples and arrives to the butterfly simultaneously with the fourth input sample. This completes the start-up of the first stage of the pipeline. The outputs from the first stage butterfly and the multiplier are then fed into the multipath delay commutator between stage 1 and stage 2. There are two paths (multipath) with delay elements and one switch (commutator). The multipath delay commutator alleviates the data dependency problem. The first and second outputs from the upper side of the butterfly are fed into the two upper delay elements. After this, the switch changes and the third and fourth outputs from the upper output of the first butterfly are sent directly to the butterfly at stage 2. However, the first and second outputs from the multiplier at the first stage are now delayed by the upper delay elements, which make the first and second outputs from the multiplier of the first stage arrive together with the fifth and sixth outputs from the top.

The butterfly and the multiplier are idle half the time to wait for the new inputs. Hence the utilization of the butterfly and the multiplier is 50%. The total number of delay elements is $4 + 2 + 2 + 1 + 1 = 10$ for the 8-point FFT. The total number of delay elements for an $N$-point FFT can be derived in similar way and is $\frac{N}{2} + \frac{N}{2} + \frac{N}{4} + \ldots + 2$, i.e., $3\frac{N}{2} - 2$. Each stage (except the last one) has one multiplier and the number of multipliers is $\log_2(N) - 1$. 

Figure 4.4 8-point DIF R2MDC architecture.
4.4.2 Radix-2 Single-Path Delay Feedback

Herbert L. Groginsky and George A. Works introduced a feedback mechanism in order to minimize the number of delay elements [83]. In the proposed architecture one half of outputs from each stage are fed back to the input data buffer when the input data are directly sent to the butterfly. This architecture is called Radix-2 Single-path Delay Feedback (R2SDF). Fig. 4.5 shows the principle of an 8-point R2SDF FFT.

![Figure 4.5 8-point DIF R2SDF FFT.](image)

The delay elements at the first stage save four input samples before the computation starts. During the execution they store one output from the butterfly of the first stage and one output is immediately transferred to the next stage. Thus, in the new interim half frame when the delay elements are filled with fresh input sample, the results of the previous frame are sent to the next stage. The butterfly is provided with a feedback loop. The modified butterfly is shown in the right side of Fig. 4.5. When the mux is 0, the butterfly is idle and data passes by. When the mux is 1, the butterfly processes the incoming samples. Because of the feedback mechanism we reduce the requirement of delay elements from \(3N/2\) to \(N - 1 (N/2 + N/4 +... + 1)\) which is minimal. The number of multiplier is exact the same as R2MDC FFT.
architecture, i.e., \( \log_2(N) - 1 \). The utilization of multiplier and butterflies remains the same, namely 50%.

### 4.4.3 Radix-4 Multipath Delay Commutator

This architecture is similar to R2MDC. Input data are separated by a 4-to-1 multiplexer and \( 3N/2 \) delay elements at the first stage. A 4-path delay commutator is used between two stages. Computation is taking place only when the last 1/4 part of data is multiplexed to the butterfly. The utilization of the butterflies and the multipliers is 25%. The length of the FFT has to be \( 4^n \). A length-64 DIF Radix-4 Multipath Delay Commutator (R4MDC) FFT is shown in Fig. 4.6.

![Figure 4.6 64-point DIF R4MDC FFT.](image)

Each stage (except the last stage) has 3 multipliers and the R4MDC FFT requires in total 3. \( (\log_4(N)-1) \) multipliers for an \( N \)-point FFT which is more than the R2MDC or R2SDF. Moreover the memory requirement is \( 5N/2 - 4 \), which is the largest among the three discussed architectures. From the view of hardware and utilization, it is not a good structure.

### 4.4.4 Radix-4 Single-Path Delay Commutator

To increase the utilization of the butterflies, G. Bi and E. V. Jones [84] proposed a simplified radix-4 butterfly. In the simplified radix-4 butterfly, only one output is produced in comparison with 4 in the conventional butterfly. To provide the same four outputs, the butterfly works four times instead of just one. Due to this modification the butterfly has a utilization of 4.25 % or 100%. To accommodate this change we must provide the same four data at four different times to the
butterfly. A few more delay elements are required with this architecture. Furthermore, the simplified butterfly needs additional control signals, and so do the commutators. The number of multipliers is $\log_4(N)-1$, which is less than the R4MDC FFT architecture. The utilization of the multiplier is 75% due to the fact that at least one-fourth of the data are multiplied with the trivial twiddle factor 1 (no multiplication is needed). The structure of a 16-point DIF Radix-4 Single-Path Delay Commutator (R4SDC) FFT is shown below.

![Figure 4.7 16-point DIF R4SDC FFT.](image)

The main benefit of this architecture is the utilization improvement for butterflies. The cost for R4SDC is the increase amounts of delay elements.

### 4.4.5 Radix-4 Single-Path Delay Feedback

Radix-4 single-path delay feedback (R4SDF) [85] [86] is a radix-4 version of R2SDF. Since we use the radix-4 algorithm we can reduce the number of multipliers to $\log_4(N) – 1$ compared to $\log_2(N) – 2$ for R2SDF. But the utilization of the butterflies are reduced to 25%. The radix-4 SDF butterflies also become more complicated than the radix-2 SDF butterflies. A64-point DIF R4SDF FFT is illustrated in Fig. 4.8.
The radix-4 SDF butterfly is shown in Fig. 4.9. The data are sent to the butterfly for processing when the mux is 1, otherwise the data are shifted into a delay-line with a length of $3N/4$ (first stage).

4.4.6 Radix-$2^2$ Single-Path Delay Commutator

The Radix-$2^2$ Single-path Delay Commutator (R2$^2$SDC) architecture [3] uses a modified radix-4 DIF FFT algorithm. It has the same butterfly structure as the radix-2 DIF FFT, but places the multipliers at the same places as for the radix-4 DIF
FFT. Basically two kinds of radix-2 SDF butterflies are used to achieve the same output (but not of the same order) as a radix-4 butterfly. By reducing the radix from 4 to 2 we increase utilization of the butterflies from 25% to 50%. We reduce the number of multipliers compared to the conventional radix-2 algorithm. This approach is based on a 4-point DFT.

![Diagram of 64-point DIF R2²SDC FFT](image)

The outputs are bit-reversed instead of 4-reversed as in a conventional radix-4 algorithm.

### 4.5 Summary

In this chapter, several FFT implementation classes are discussed. The programmable DSP or FFT-specific processors cannot meet the requirements in both high throughput and low power applications. Algorithm-specific implementations, especially with pipelined FFT architectures are better in this respect.