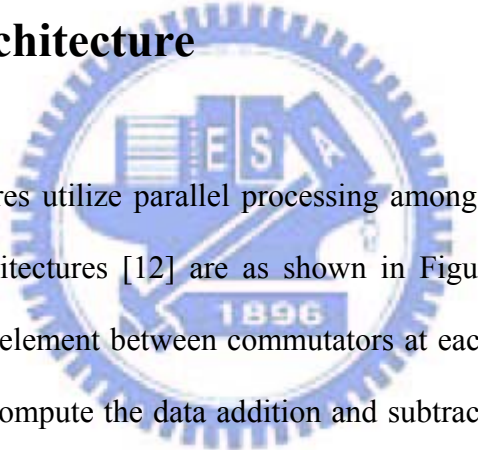


# Chapter 4

## Classification of FFT Processor Architectures

There are numerous architectures for realizing FFT computations in real time applications. In this chapter, we will only discuss two main classes: *pipeline architecture* and *shared memory-based architecture*. We also compare different architectures with respect to hardware requirement cost and throughput.

### 4.1 Pipeline Architecture



Pipeline architectures utilize parallel processing among the stages. The general structures of these architectures [12] are as shown in Figure 4.1. These structures consist of one butterfly element between commutators at each stage. The function of butterfly element is to compute the data addition and subtraction. The commutator is like a switch to rearrange the data from the butterfly element in order to perform subsequent calculations more conveniently.

Pipeline architecture is widely used to attain high-speed operation. The problem with this architecture is the relatively large die area required by the  $\log_r^N$  butterfly elements. Pipeline architecture can be roughly classified into three types:

- *single path delay feedback* (SDF) architecture,
- *multiple path delay commutator* (MDC) architecture and
- *single path delay commutator* (SDC) architecture.

These types have different properties, each with advantages and disadvantages. We will describe these properties in the following subsections.

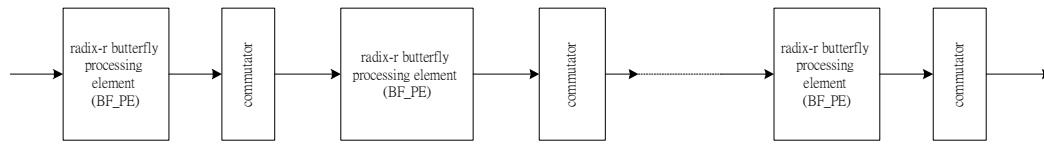


Figure 4.1 Pipeline architecture

### 4.1.1 Single Path Delay Feedback (SDF) Architecture

In the general single path delay feedback architecture, the input data sequences pass through one single path. The butterfly processing element performs the computations on the data. This structure uses shift registers or RAM to store the output data from the butterfly elements. As a result of the single output path, only one complex multiplier is required.

Here, we give two examples to illustrate the principle of single path delay feedback architecture. One example is a 16-point R2SDF and the other is a 16-point R4SDF.

#### (1) Radix-2 single delay feedback (R2SDF)

Figure 4.2 is a 16-point radix-2 single path delay feedback architecture[24]. The procedural flow of this architecture is as follows.

- (a) In the beginning, the input data with indices 0 to 7 are stored in the shift register (D8 in Figure 4.2). The radix-2 butterfly elements operate on these data and on the remaining input data with indices 8 to 15.
- (b) The data resulting from the butterfly addition operations are passed to the second stage and the subtraction results are fed back to the shift register (D8).

- (c) After all the 8-point data addition results are output, the subtraction data from the registers are passed to the second stage with a multiply twiddle factor coefficient. The symbol  $A$  in Figure 4.2 and Figure 4.3 represents the last output data of the first stage.
- (d) The procedure flow of the latter three stages for the 16-point R2SDF are similar to the first stage. The routing of data for R2SDF is given in Figure 4.3.

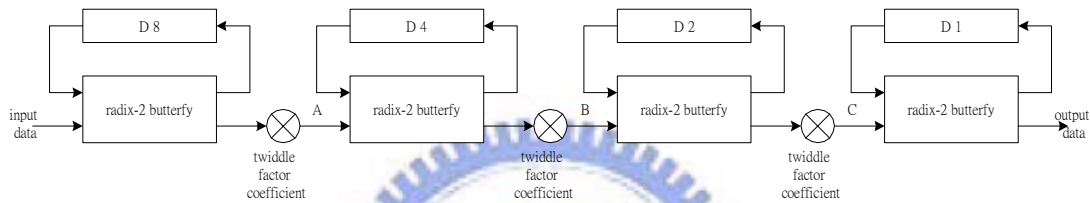


Figure 4.2 A 16- point radix-2 single path delay feedback architecture

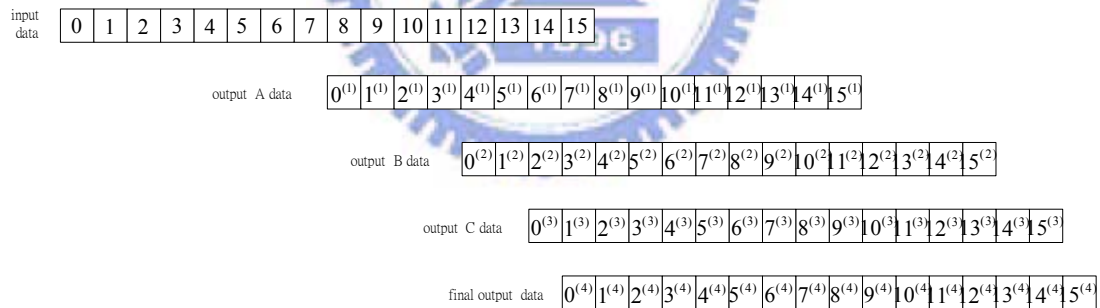


Figure 4.3 Routing of data for R2SDF (N=16)

## (2) Radix-4 single delay feedback (R4SDF)

Radix-4 single delay feedback (R4SDF) is a variant of R2SDF. In R4SDF, the butterfly operation is based on the radix-4 algorithm. The procedural flow of this architecture [25] is identical to that of R2SDF. Figure 4.4 is a block diagram of the radix-4 single delay feedback architecture. The schematic timing diagram is given in

Figure 4.5. The data output  $A$  in Figure 4.5 represents the data resulting from the 16-point radix-4 butterfly computation of the first stage. The output data in Figure 4.5 represents the data resulting from butterfly computation in the second stage.

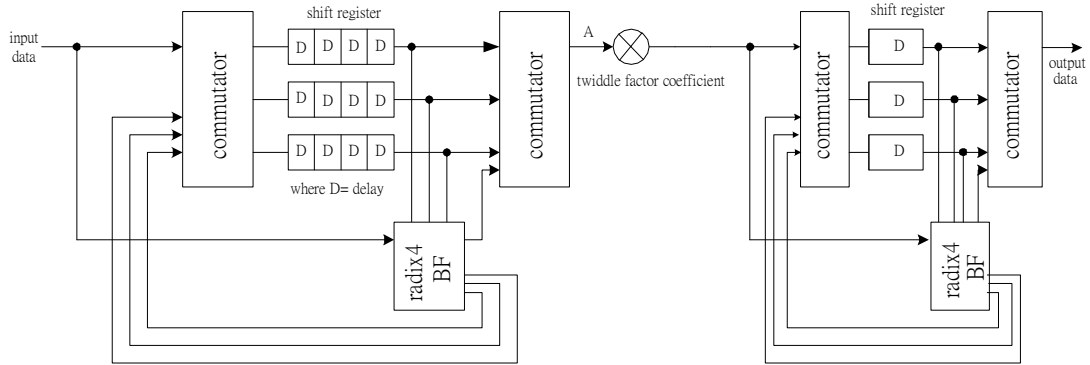


Figure 4.4 A 16- point radix-4 single path delay feedback architecture

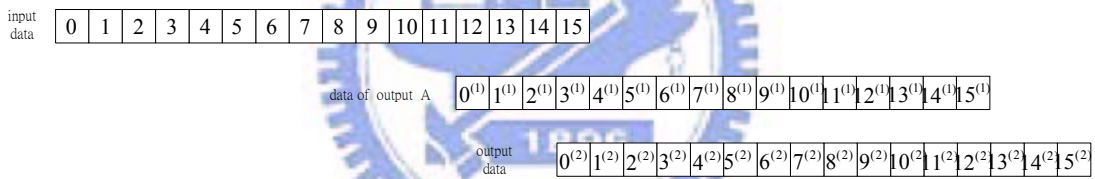


Figure 4.5 Timing diagram for R4SDF (N=16)

### 4.1.2 Multiple Path Delay Commutator (MDC) Architecture

In the general multiple path delay feedback architecture [14], the input data sequences are organized into a parallel data path entering the commutator by suitable delays and data summation is performed by the butterfly element at each stage. The intermediate data is not fed back to register. It has a higher throughput data rate than the single path delay feedback architecture.

Here, we give an example to illustrate the principles of the multiple path delay commutator architecture. The architecture of a 16-point Radix-4 multiple path delay commutator architecture (R4MDC) is as shown in Figure 4.6. The data routing diagram is depicted in Figure 4.7. The workflow for this architecture is

- (a) In the beginning, the input data from index 0 to 15 is decomposed into four parallel data streams of four points each after the first commutator.
- (b) These data are passed through different delay registers.
- (c) After the delay, the resulting data are operated on by radix-4 butterfly processing elements (BF4) and then multiplied by a twiddle factor.
- (d) The result data are stored in the delay register.
- (e) The function of the second commutator is to reorder the desired adjacent data.
- (f) Before the second butterfly operation, the ordered data are stored in an appropriate delay register. The first parallel data stream must be shifted by 3 delays. The second parallel data stream must be shifted by 2 delays. The third parallel data stream must be shifted by 1 delay and the last data stream is directly passed to the butterfly input in the second stage.

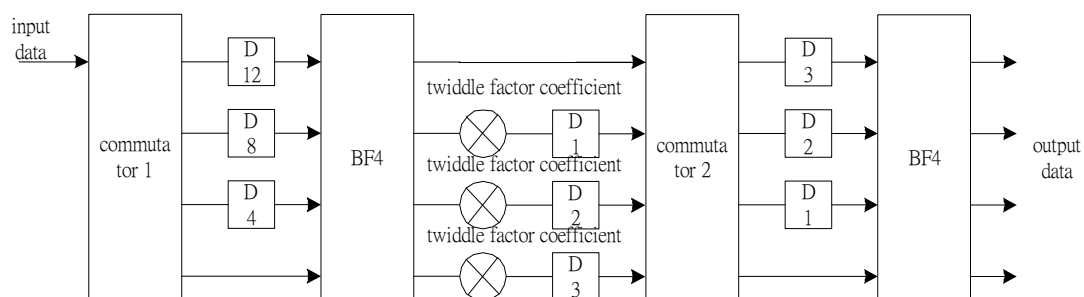


Figure 4.6 A 16-point radix-4 multiple path delay commutator architecture

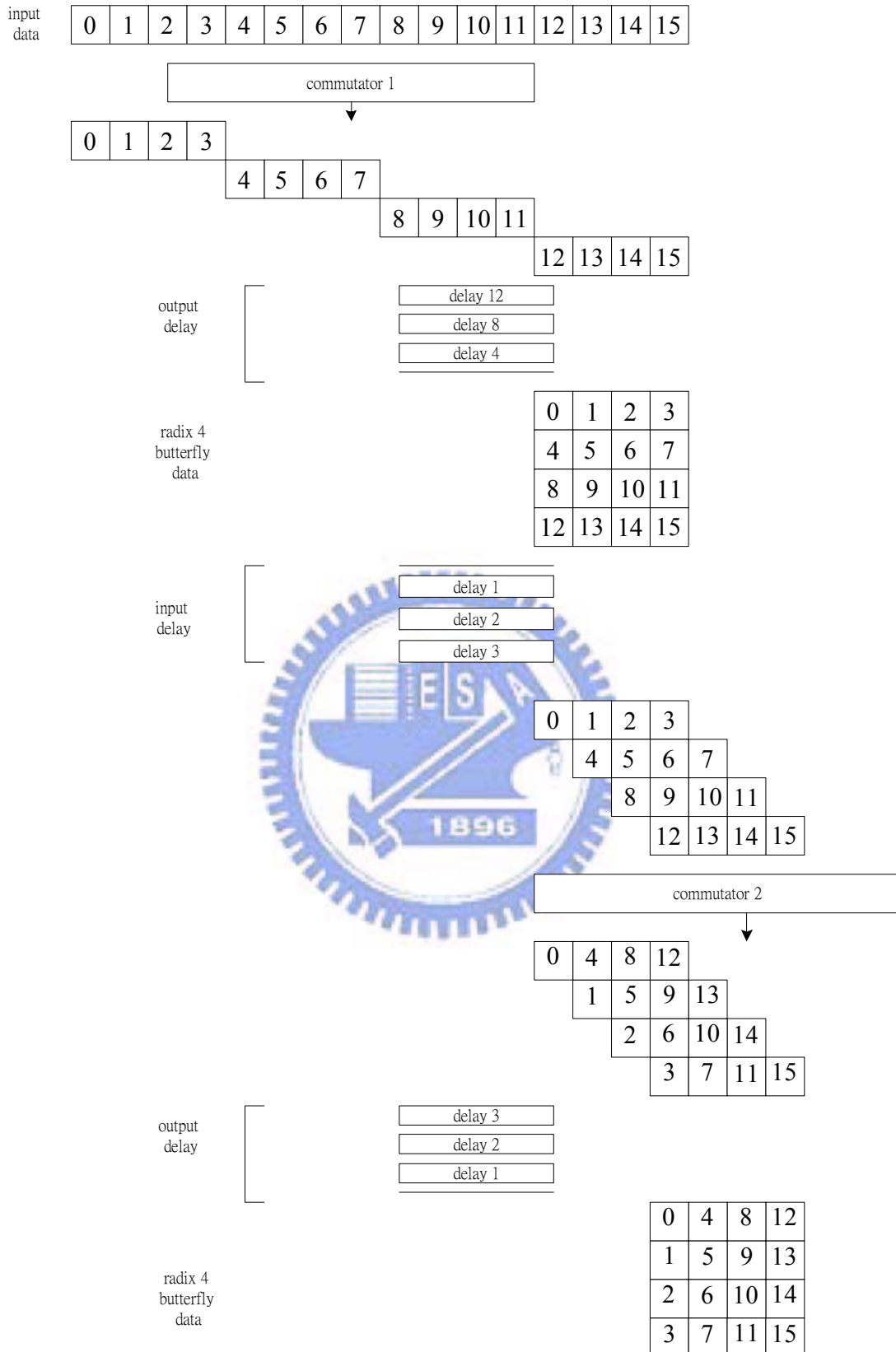


Figure 4.7 Timing diagram for R4MDC (N=16)

### 4.1.3 Single Path Delay Commutator (SDC) Architecture

The single path delay commutator (SDC) architecture [15][16] is based on the modified multiple path delay commutator. In each word cycle, each stage produces a single output rather than 4 outputs as in the multiple path delay commutator. Each stage requires one complex multiplier, a delay commutator to correct the order of the data, and a butterfly element. Here, we give an example to illustrate the principle of single path delay commutator architecture. The architecture of a 16-point radix-4 single path delay commutator (R4SDC) is as shown in Figure 4.8.

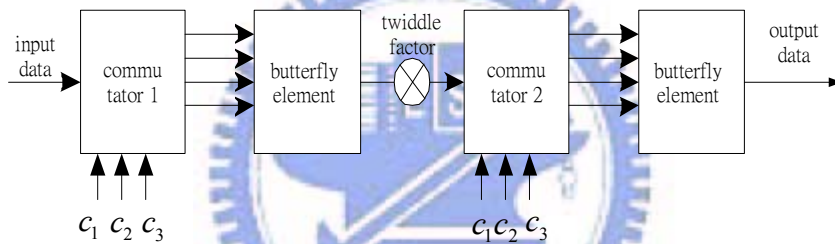


Figure 4.8 A 16-point radix-4 single path delay commutator architecture

Figure 4.9 is the commutator for the R4SDC at each stage. There are six shift registers. The symbol  $N_t$  represents the number of delays for each shift register. The multiplexer control parameters  $c_1, c_2$ , and  $c_3$  select the required data. The parameters  $m_1$  and  $m_2$  are switch control signals to reorder data. Its function is graphically charted in Figure 4.10 and Table 4.1. The timing diagram for the 16-point R4SDC is described in Figure 4.11.

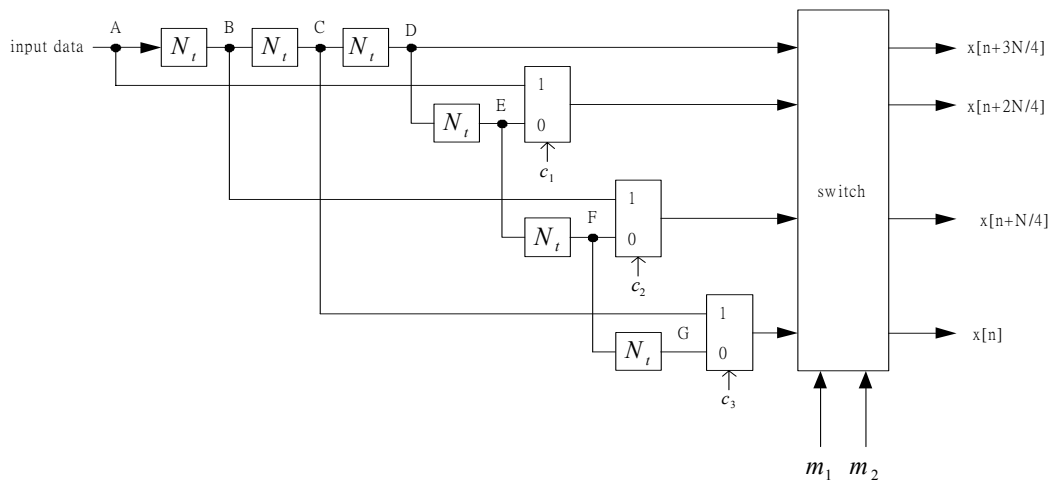


Figure 4.9 Delay commutator for R4SDC at each stage

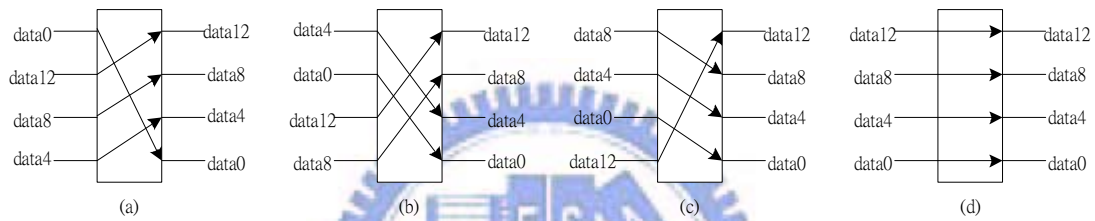


Figure 4.10 Action of switch in Figure 4.9 for 16-point FFT

Table 4.1 Control signals at each stage in Figure 4.9

$c_1$ $c_2$ $c_3$	$m_1$ $m_2$	Figure
1 1 1	0 0	Figure 4.10 (a)
0 1 1	0 1	Figure 4.10 (b)
0 0 1	1 0	Figure 4.10 (c)
0 0 0	1 1	Figure 4.10 (d)



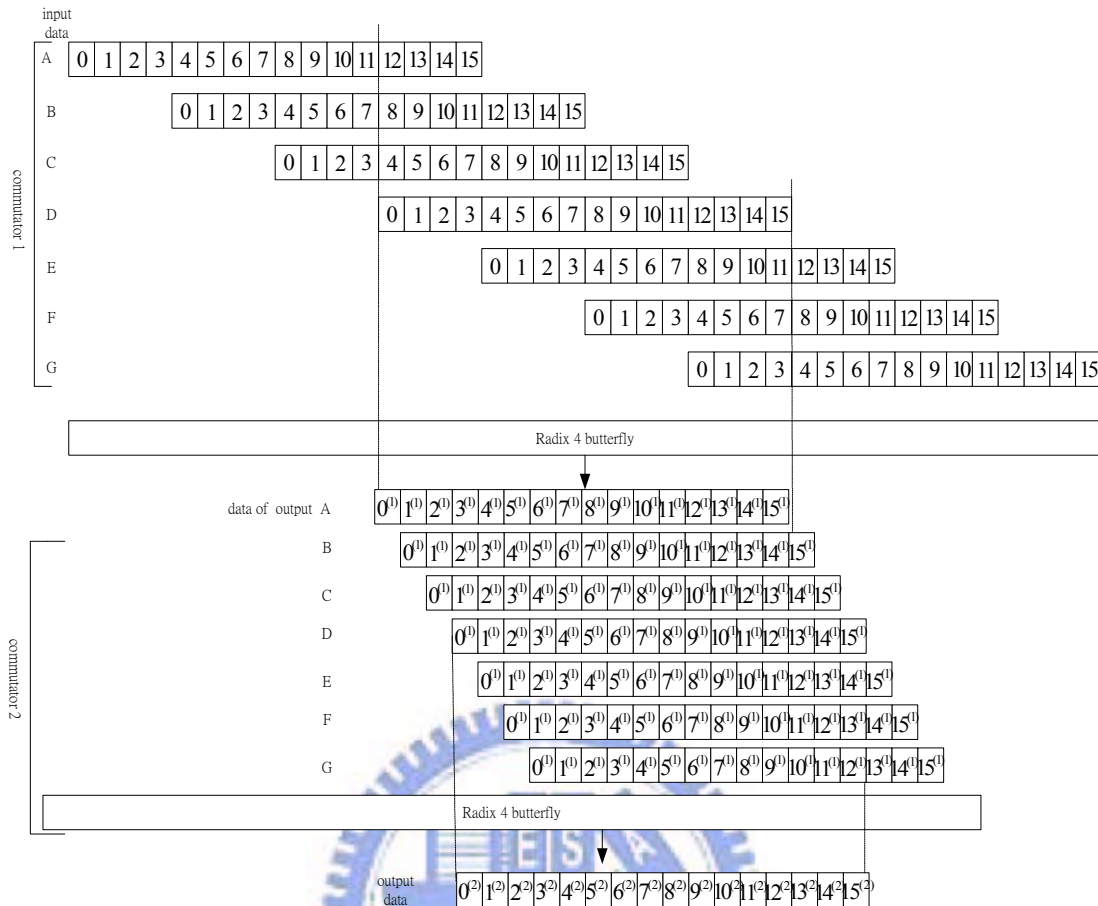


Figure 4.11 Timing diagram for 16-point R4SDC

## 4.2 Memory-Based Architecture

Memory based architectures, unlike pipelined architectures, generally use only one butterfly processor. Thus, their advantage is to save hardware cost. Because a single butterfly processing element can complete only one butterfly operation per clock cycle, it implies a loss in overall processing speed. We can compensate for this disadvantage by using the high radix algorithm [17]. Furthermore, we can increase the system clock rate to achieve the required speed.

Besides the butterfly processor, a memory based architecture as shown in Figure 4.12 also includes main memory, an address generator, and commutators. The

butterfly input and output are connected to main memory. The function of the address generator is to control the read/write addresses for memory access. The commutator arranges the data after reading from and writing back to the memory so as to generate the correct order for FFT computation.

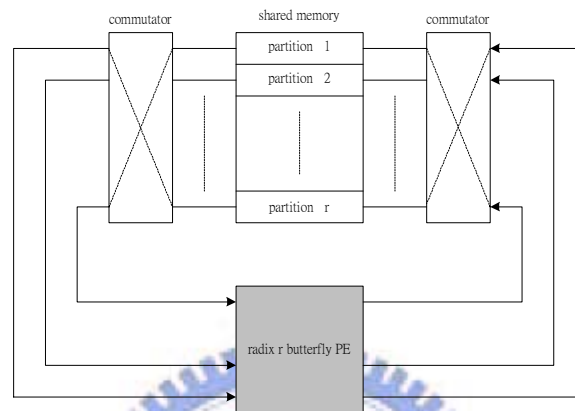


Figure 4.12 Schematic diagram of memory architecture

## 4.2.1 Memory-Based Processing Method

### (1) Single memory-based architecture

The simplest memory-based architecture is the single memory-based architecture [18]. It has only one memory, as shown in Figure 4.13. Hence it can be used in place of a memory addressing strategy that stores the new FFT operation data in the same location used by the previous FFT operation data.

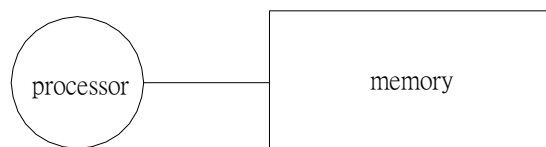


Figure 4.13 Single memory-based architecture block diagram

## (2) Dual memory-based architecture

Dual memory-based architecture [11] performs “ping-pong-like” actions to accomplish read-and-then-write operations. Each of the two memories alternates their functions between butterfly inputs and butterfly outputs. This means that it can read out input data from the memory and write the output results back to the memory simultaneously. Figure 4.14 is the schematic diagram of dual memory-based architecture.

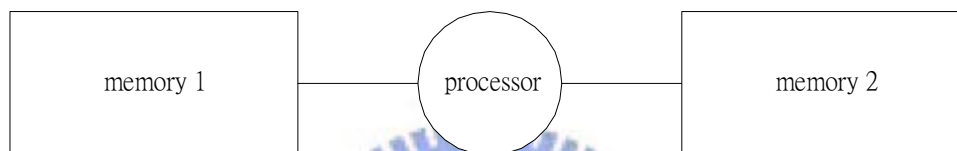


Figure 4.14 Dual memory-based architecture block diagram

## (3) Cache memory-based architecture

Cache memory-based architecture shown in Figure 4.15 consists of one processor, a small cache memory and a main memory. It is similar to the single memory architecture shown in Figure 4.15, but with a cache memory. The purpose of this architecture is to reduce the number of main memory accesses and to reduce power consumption even if the FFT length is large [11][19].

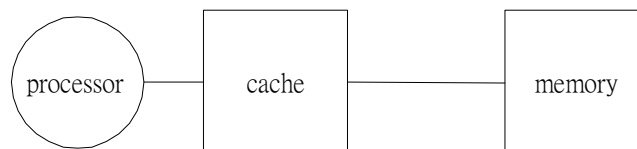


Figure 4.15 Cache memory-based architecture block diagram

## 4.2.2 Radix-2 Memory-Based Architecture

Reference [20] describes memory-based FFT implementations based on one radix-2 butterfly structure, as shown in Figure 4.16. It uses an in-place memory addressing scheme and a memory bank structure in order to increase the efficiency of memory access. In addition, [20] proposes using an address generator only for the fixed radix-2 algorithm. It partitions the main memory into two  $N/2$  location banks.

Here, we introduce the in-place memory addressing scheme that stores the butterfly output data, overwriting the memory locations being simultaneously read. This scheme provides the  $N$ -data points stored in an  $r$ -bank memory without conflicts in memory read/write access. The algorithm in [20] is as follows:

$$\begin{aligned}
 Data\_count &= [d_{n-1}, d_{n-2}, \dots, d_2, d_1, d_0]_r \\
 n &= \lceil \log_r N \rceil \\
 Bank\_index &= (d_{n-1} + d_{n-2} + \dots + d_2 + d_1 + d_0) \bmod r \\
 Address\_index &= [d_{n-1}, d_{n-2}, \dots, d_2, d_1]_r
 \end{aligned} \tag{4.1}$$

In Equation (4.1),  $Data\_count$  represents the original data index.  $Bank\_index$  expresses the appropriate value of *bank* after memory partition.  $Address\_index$  is the new address location in the assigned  $Bank\_index$ .

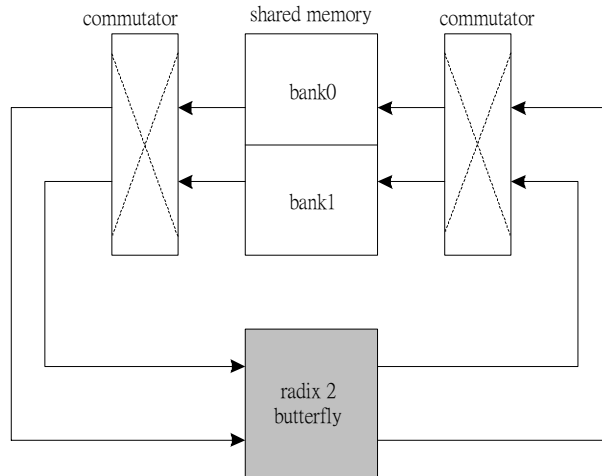


Figure 4.16 Radix-2 memory-based FFT processor

### 4.2.3 Radix-4 memory-Based Architecture

The architecture proposed in [18] uses a single memory. To achieve high speed, it uses the radix-4 algorithm, as illustrated in Figure 4.17. It applies the in-place memory addressing in Section 4.2.2, but modifies the address generation unit and control unit.

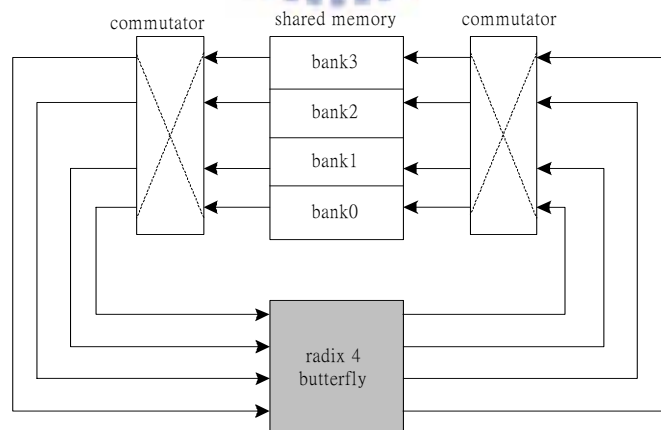


Figure 4.17 Radix-4 memory-based FFT processor

### 4.3 Comparison of Different Architectures

In this subsection, we summarize different architectures with different hardware requirements.

Table 4.2 Comparison hardware of different pipelined FFT processor

	No. of complex multipliers	No. of complex adders	throughput	No. of registers
R2SDF	$\log_2 N - 2$	$2 \log_2 N$	$T_{FFT} = T_{r,PE} \cdot N$	N-1
R4SDF	$\log_4 N - 1$	$8 \log_4 N$	$T_{FFT} = T_{r,PE} \cdot N$	N-1
R4MDC	$3 \log_4 N - 3$	$8 \log_4 N$	$T_{FFT} = T_{r,PE} \cdot \frac{N}{r}$	$(5/2)N-4$
R4SDC	$\log_4 N - 1$	$3 \log_4 N$	$T_{FFT} = T_{r,PE} \cdot N$	$\sum_{i=1}^{\log_4 N} \frac{6N}{4^i} = 2(N-1)$

Table 4.3 Comparison utilization of different pipelined FFT processor [13]

architecture	utilization of multipliers	utilization of adders	utilization of registers
R2SDF	50 %	50 %	100 %
R4SDF	25 %	25 %	100 %
R4MDC	25 %	25 %	25 %
R4SDC	75 %	100 %	100 %

Table 4.4 Comparison hardware of different memory based FFT processor

architecture	No. of complex multipliers	No. of computation cycles
radix-2 memory based architecture	1	$\frac{N}{2} \log_2 N + 2$
radix-4 memory based architecture	3	$\frac{N}{4} \log_4 N$