# EEC 281 — VLSI Digital Signal Processing
# Notes on the RRI-FFT

Bevan Baas

The attached description of the *RRI-FFT* is taken from a chapter describing the *cached-FFT* algorithm and there are therefore occasional references to it.

However, the purpose of this handout is to present a very simple and regular FFT algorithm, which is the RRI-FFT (our made up name). Section 4.6 provides all details needed to implement a very efficient and simple processor controller to generate the key processor signals:

- butterfly counter
- group counter
- butterfly input/output addresses (input and output addresses are the same)
- $W_N$ exponents (can be same as address for a $W_N$ lookup table)

Keep in mind a reasonable processor will be pipelined and therefore some additional circuits are required to properly time signals.

Here is an overview of the sections of the handout.

**Smit and Huisken (1995)**

Smit and Huisken propose using 32 and 64-word memories in the design of a 2048-point FFT processor. Unfortunately, they give no details or references regarding the algorithm.

**Stevens *et al.* (1998)**

Stevens *et al.* (1998) and Hunt *et al.* (1998) propose a mixed-radix FFT decomposition using $N_1$-point and $N_2$-point transforms, where $N = N_1 N_2$. They estimate the energy dissipation of their proposed 1024-point asynchronous FFT processor to be 18 $\mu$J per transform. In their comparison with other processors, they unfortunately cite Spiffee1's energy dissipation at a supply voltage of 3.3 V as 50 $\mu$J, when it is actually 25 $\mu$J. Although not noted in their papers, Spiffee1's measured energy dissipation at a supply voltage of 1.1 V is 3.1 $\mu$J per transform.

## 4.3 General FFT Terms

Some of the terms discussed in this section were used loosely in Ch. 2. They are now described more completely.

**Radix-$r$ FFT algorithms.** A few key characteristics of radix-$r$, FFTs are that they:

- Use only radix-$r$ butterflies, which have $r$ inputs and $r$ outputs
- Have $\log_r N$ *stages* (see *stage* definition below)
- Contain $N/r$ butterflies per stage

**Definition: Stage** A *stage* is the part of an FFT where all $N$ memory locations are read, processed by a butterfly, and written back once. □

In an FFT dataflow diagram, a stage corresponds to a column of butterflies. Figure 4.2 is a representative dataflow diagram which shows the six stages (0–5) of a 64-point radix-2 FFT. Additionally, the 64 memory locations are indicated along the vertical axis. By convention, memory locations are numbered with 0 at the top and $N - 1$ at the bottom.

**Definition: In-place** A butterfly is *in-place* if its inputs and outputs use the same memory locations. An *in-place FFT* uses only in-place butterflies. □

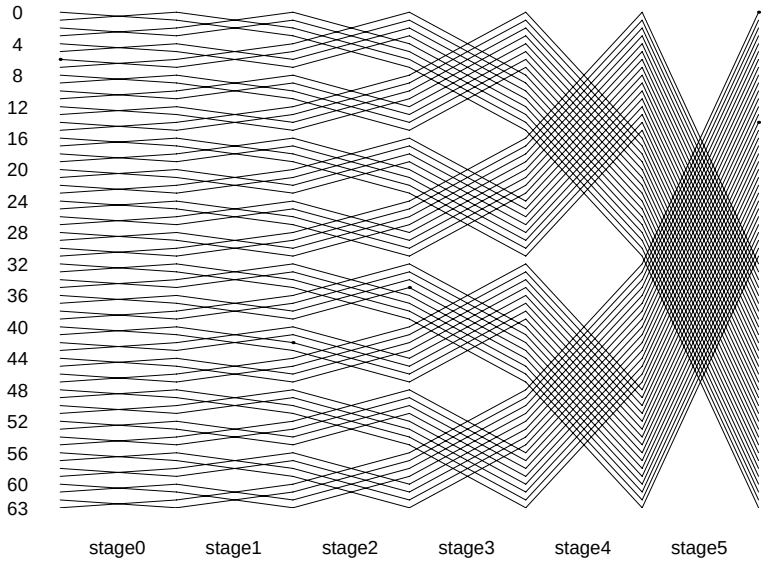


**Figure 4.2**: FFT dataflow diagram with labeled stages

**Definition: Stride** The *stride* of a butterfly is the distance (measured in memory locations) between adjacent "legs" or "spokes" of a butterfly. □

The stride is also the increment between memory addresses if memory words are accessed in increasing order. As an example, Fig. 4.3 shows a radix-2 butterfly with one leg reading memory location $k$ and the adjacent leg (the only other one in this case) reading location $k+4$. The stride is then $(k+4)-k=4$. Similarly, Fig. 4.4 shows a radix-4 butterfly with a stride of one between its input and output legs.

**Definition: Span** The *span* of a butterfly is the maximum distance (measured in memory locations) between any two butterfly legs. □

Using the two previous examples, the span of the butterfly in Fig. 4.3 is four and the span of the butterfly in Fig. 4.4 is three. For a radix-$r$ butterfly with constant stride between all adjacent legs pairs, the span can also be found by,

$$span = (r-1)stride. \tag{4.1}$$

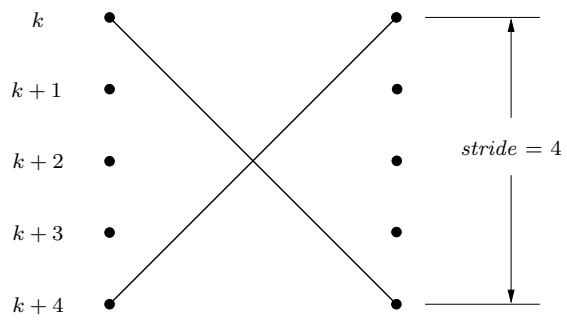


**Figure 4.3**: Radix-2 butterfly with $stride = 4$

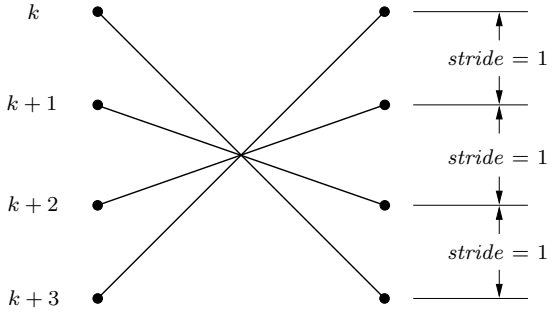


**Figure 4.4**: Radix-4 butterfly with $stride = 1$

## 4.4 The RRI-FFT Algorithm

**Definition: RRI-FFT** The *RRI-FFT* (Regular, Radix-$r$, In-place FFT) algorithm is a radix-$r$ in-place FFT with the following additional characteristics:

- The stride of all legs of a butterfly (input and output) are equal.
- The stride of all butterflies in a stage are equal.
- The stride of butterflies monotonically increases or decreases by a factor of $r$ as the stage number varies from 0 to $\log_r(N) - 1$.

□

### 4.4.1 Existence of the RRI-FFT

**Theorem 1** *It is always possible to draw an RRI-FFT dataflow diagram where the stride of butterflies increases by a factor $r$ through succeeding stages.*

**Proof** From the RRI-FFT definition, we assume a radix-$r$, in-place decomposition using butterflies with constant stride. Extending the FFT derivation given in Sec. 2.4, common-factor FFTs can be developed through the following procedure:

1. Decimate a 1-dimensional $M$-point data sequence by $r$, which results in a 2-dimensional $(M/r) \times r$ array
2. Perform DFTs of the columns (or rows)
3. Possibly multiply the intermediate data by twiddle factors
4. Perform DFTs of the rows (or columns)

The calculation of the DFT of a row (or column) longer than $r$ members normally involves the recursive treatment of that sequence. Figure 4.5 shows a common way of performing the decimation by $r$ for a sequence of length $N$. The sequence is decimated such that the index of $x(n)$ increases by one in one dimension (vertical in this case), and increases by $r$ in the other dimension. Radix-$r$ butterflies calculate the column-wise DFTs. The row-wise DFTs are, in general, longer than $r$ and therefore require further decimations.

During the decimation of the sequences, a single sequence is folded into $r$ sequences that are $1/r$ the length of the previous sequence. The final decimation ends with an $r \times r$ array where radix-$r$ butterflies calculate DFTs of both columns and rows.

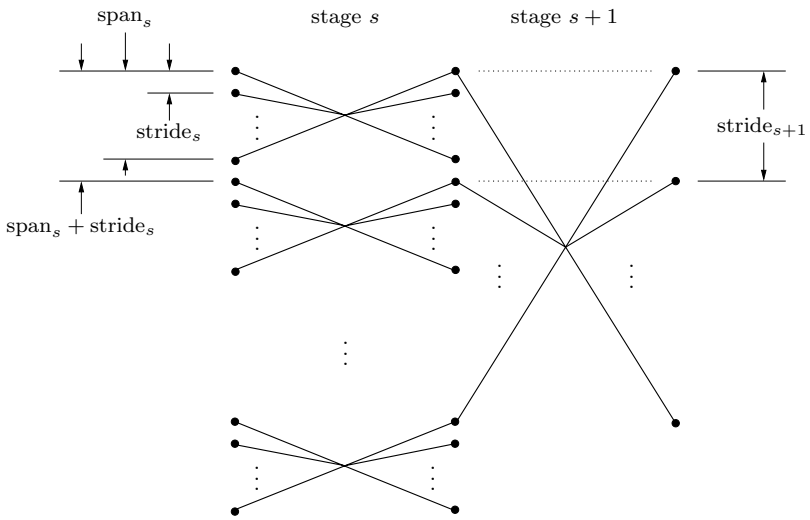


**Figure 4.14**: Extended diagram of the *stride* and *span* of arbitrary-radix butterflies

## 4.6 A General Description of the RRI-FFT

Descriptions of FFT algorithms in the literature typically consist of a dataflow diagram for a specific case, and a simple software program. This section provides a general description of the RRI-FFT which more readily provides insight into the algorithm and extensions to it.

We begin by closely examining the memory access pattern of the RRI-FFT dataflow diagram, shown in Fig. 4.2 on page 57, and recognizing that butterflies are clustered into "groups." For example, in stage 2, butterflies are clustered into eight "groups" of four butterflies each. Because of the unique clustering of butterflies into groups, a concise description of the memory accesses makes use of two counters:

- a *group* counter—which counts groups of butterflies
- a *butterfly* counter—which counts butterflies within a group

Table 4.2 contains a generalized listing of the number of groups per stage and the number of butterflies per group, as well as the number of digits required for each counter. The rightmost column contains the sum of the *group* counter bits and the *butterfly* counter bits and is constant across all stages. The constant sum is clearly necessary since both counters

| Stage number | Groups per stage | Digits in *group* counter | Butterflies per group | Digits in *butterfly* counter | Total counter bits |
|---|---|---|---|---|---|
| 0 | $N/r$ | $\log_r(N)-1$ | 1 | 0 | $\log_r(N)-1$ |
| 1 | $N/r^2$ | $\log_r(N)-2$ | $r$ | 1 | $\log_r(N)-1$ |
| 2 | $N/r^3$ | $\log_r(N)-3$ | $r^2$ | 2 | $\log_r(N)-1$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $\log_r(N)-2$ | $r$ | 1 | $N/r^2$ | $\log_r(N)-2$ | $\log_r(N)-1$ |
| $\log_r(N)-1$ | 1 | 0 | $N/r$ | $\log_r(N)-1$ | $\log_r(N)-1$ |

**Table 4.2**: *Butterfly* counter and *group* counter information describing the memory access pattern for an $N$-point, radix-$r$, DIT RRI-FFT

together address a single butterfly within a stage, and the number of butterflies per stage is constant ($N/r$).

To fully describe the RRI-FFT, it is necessary to specify the memory access patterns and the values of the $W_N$ butterfly coefficients. Table 4.3 shows the values of digits in the *group* and *butterfly* counters which are needed to generate memory addresses. The asterisks represent digit positions where different values in that position address the $r$ inputs and $r$ outputs of the butterflies. For example, in the radix-2 case, a "0" in place of the asterisk addresses the $A$ input and $X$ output of the butterfly, and a "1" in the asterisk's position addresses the $B$ input and $Y$ output. In the rightmost column of the table, $b_k$ digits generate exponents for the $W_N$ coefficients.

Note that the table does not specify a particular order in which to calculate the FFT. The subscripts of the counter digits $g$ and $b$ show only the relationship between address digits and $W_N$-generating digits. The digits can be incremented in any order or pattern—the only requirement is that all $N/r$ butterflies are calculated once and only once. As shown in Sec. 4.7, the development of the cached-FFT relies on this property. Of course, the order of calculation is not entirely arbitrary, as a butterfly's inputs must be calculated before it can calculate its outputs.

Although the radix-2 approach is easiest to visualize, the description given by the tables applies to algorithms with other radices as well. For use with radices other than two, the

| Stage number | Digits in *group* counter | Digits in *butterfly* counter | Butterfly addresses ( x = one digit) | $W_N$ exponents |
|---|---|---|---|---|
| 0 | $\log_r(N)-1$ | 0 | $\underline{g_{\log_r(N)-2}}\ \underline{g_{\log_r(N)-3}}\ \cdots\ \underline{g_2}\ \underline{g_1}\ \underline{g_0}\ \underline{*}$ | $000\cdots00$ |
| 1 | $\log_r(N)-2$ | 1 | $\underline{g_{\log_r(N)-3}}\ \underline{g_{\log_r(N)-4}}\ \cdots\ \underline{g_1}\ \underline{g_0}\ \underline{*}\ \underline{b_0}$ | $b_000\cdots00$ |
| 2 | $\log_r(N)-3$ | 2 | $\underline{g_{\log_r(N)-4}}\ \underline{g_{\log_r(N)-5}}\ \cdots\ \underline{g_0}\ \underline{*}\ \underline{b_1}\ \underline{b_0}$ | $b_1b_00\cdots00$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $\log_r(N)-2$ | 1 | $\log_r(N)-2$ | $\underline{g_0}\ \underline{*}\ \underline{b_{\log_r(N)-3}}\ \cdots\ \underline{b_2}\ \underline{b_1}\ \underline{b_0}$ | $b_{\log_r(N)-3}b_{\log_r(N)-4}\cdots b_00$ |
| $\log_r(N)-1$ | 0 | $\log_r(N)-1$ | $\underline{*}\ \underline{b_{\log_r(N)-2}}\ \underline{b_{\log_r(N)-3}}\ \cdots\ \underline{b_2}\ \underline{b_1}\ \underline{b_0}$ | $b_{\log_r(N)-2}b_{\log_r(N)-3}\cdots b_1b_0$ |

**Table 4.3**: Addresses and base $W_N$ exponents for an $N$-point, radix-$r$, DIT RRI-FFT. The variables $g_k$ and $b_k$ represent the $k^{th}$ digits of the *group* and *butterfly* counters respectively. Asterisks ($*$) represent digit positions where the $r$ possible values address the $r$ butterfly inputs and $r$ outputs.

counter digits (*e.g.,* $g_1$, $*$, $b_0$,...) are interpreted as base-$r$ digits instead of binary digits ($\in [0, 1]$) as in the radix-2 case. Higher-radix algorithms generally require more than one $W_N$ coefficient, so although Table 4.3 gives only one $W_N$, it serves as a base factor where other coefficients are normally multiples of the given value.

**Example 4** $N = 64$ **Radix-2 RRI-FFT**

Table 4.4 details the generation of butterfly addresses and $W_N$ coefficients for a 64-point, radix-2, DIT RRI-FFT. Because $r = 2$, there are $\log_2 64 = 6$ stages. Since the example uses a radix-2 decomposition, all counter places are binary digits. ◁

| Stage number | Butterfly address digits ( x = one digit) | $W_N$ butterfly coefficients |
|---|---|---|
| stage 0 | $g_4$ $g_3$ $g_2$ $g_1$ $g_0$ $*$ | $W_{64}^{00000}$ |
| stage 1 | $g_3$ $g_2$ $g_1$ $g_0$ $*$ $b_0$ | $W_{64}^{b_0 0000}$ |
| stage 2 | $g_2$ $g_1$ $g_0$ $*$ $b_1$ $b_0$ | $W_{64}^{b_1 b_0 000}$ |
| stage 3 | $g_1$ $g_0$ $*$ $b_2$ $b_1$ $b_0$ | $W_{64}^{b_2 b_1 b_0 00}$ |
| stage 4 | $g_0$ $*$ $b_3$ $b_2$ $b_1$ $b_0$ | $W_{64}^{b_3 b_2 b_1 b_0 0}$ |
| stage 5 | $*$ $b_4$ $b_3$ $b_2$ $b_1$ $b_0$ | $W_{64}^{b_4 b_3 b_2 b_1 b_0}$ |

**Table 4.4**: Addresses and $W_N$ coefficients for a 64-point, radix-2, DIT FFT

## 4.7 A General Description of the Cached-FFT

As previously mentioned, the derivation of the cached-FFT is greatly simplified by viewing the cached-FFT as a modified RRI-FFT. To further simplify the development, we initially consider only *balanced* cached-FFTs. Section 4.7.2 addresses the handling of unbalanced cached-FFTs. Further, we consider only DIT decompositions, and note that the development of DIF variations begins with a DIF form of the RRI-FFT and is essentially identical.

First, a review of two key points regarding the counter digits in Table 4.3: (i) the sum of the number of digits in the *group* and *butterfly* counters is constant, and (ii) the counter