Progettazione e sintesi di circuiti digitali

Fast Fourier Transform – VLSI implementation
– FFT algorithms revisited
– Radix-\(r\) Cooley – Tukey algorithm
– Serial architecture
– Pipelined architecture
– VLSI examples
FFT algorithms – index mappings

• The FFT algorithms can be described through a transformation of the index maps of the input and output sequences. Given the N-point DFT:

\[ X(k) = \sum_{n=0}^{N-1} x(n)W_N^{nk} \]

• the index transformations are defined as:

\[ N = N_1 \cdot N_2; \quad A, B, C, D \in \mathbb{Z} \]

\[ n = (n_1 \cdot A + n_2 \cdot B) \mod N \quad \text{with} \quad n_{1(2)} = 0, \ldots, N_{1(2)} - 1 \]

\[ k = (k_1 \cdot C + k_2 \cdot D) \mod N \quad \text{with} \quad k_{1(2)} = 0, \ldots, N_{1(2)} - 1 \]

• If \( N_1 \) and \( N_2 \) have a common factor, i.e. \( \gcd(N_1, N_2) > 1 \), then the algorithm is called Common Factor Algorithms (CFA)
  • Cooley – Tukey

• If \( N_1 \) and \( N_2 \) are coprime, i.e. \( \gcd(N_1, N_2) = 1 \), then the algorithm is called Prime Factor Algorithm (PFA)
  • Good – Thomas, Winograd
Cooley – Tukey (CT) algorithm revisited

• The Cooley – Tukey (CT) is the most general CFA since it allows any factorization of \( N \). The CT index transformations are defined as:

\[
N = N_1 \cdot N_2; \quad A = N_2; \quad B = 1; \quad C = 1; \quad D = N_1;
\]
\[
n = n_1 \cdot N_2 + n_2; \quad n_1 = 0,\ldots,N_1 - 1; \quad n_2 = 0,\ldots,N_2 - 1;
\]
\[
k = k_1 + k_2 \cdot N_1; \quad k_1 = 0,\ldots,N_1 - 1; \quad k_2 = 0,\ldots,N_2 - 1;
\]

• that gives:

\[
\hat{X}(k_1,k_2) = \sum_{n_2=0}^{N_2-1} W_{N_2}^{n_2 k_2} \left[ \sum_{n_1=0}^{N_1-1} x(n_1,n_2) W_{N_1}^{n_1 k_1} \right] = \sum_{n_2=0}^{N_2-1} W_{N_2}^{n_2 k_2} \hat{y}(n_2,k_1)
\]
\[
\hat{y}(n_2,k_1) = W_{N_1}^{n_1 k_1} \sum_{n_1=0}^{N_1-1} x(n_1,n_2) W_{N_1}^{n_1 k_1};
\]
\[
\hat{x}(n_1,n_2) = x(n_1 \cdot N_2 + n_2);
\]
\[
\hat{X}(k_1,k_2) = X(k_1 + k_2 \cdot N_1);
\]
Cooley – Tukey (CT) algorithm revisited

- The N-point DFT factorized according to the CT algorithm requires to compute:
  - the $N_1$-point DFT of each of the $N_2$ input samples subset:
    \[ \{x(n_1N_2+n_2): n_1=0, ..., N_1-1\} \Rightarrow \sum_{n_1=0}^{N_1-1} x(n_1, n_2)W_{N_1}^{n_1k_1}; \]
  - $N_1 \times N_2$ twiddle-factor multiplications $\Rightarrow W_{N}^{n_2k_1}$
  - the $N_2$-point DFT of each of the $N_1$-point subset obtained by picking the results of the multiplications with the same $k_1$ index value:
    \[ \Rightarrow \hat{X}(k_1, k_2) = \sum_{n_2=0}^{N_2-1} W_{N_2}^{n_2k_2} \hat{y}(n_2, k_1) \]
Cooley – Tukey (CT) algorithm revisited

\[
x(n_1N_2) \rightarrow \text{DFT}\ N_1 \rightarrow W_N^0 \rightarrow \text{DFT}\ N_2 \rightarrow \text{DFT}\ N_2 \rightarrow X(k_2N_1)
\]

\[
x(n_1N_2+1) \rightarrow \text{DFT}\ N_1 \rightarrow W_N^{k_1} \rightarrow \text{DFT}\ N_2 \rightarrow \text{DFT}\ N_2 \rightarrow X(k_2N_1+1)
\]

\[
x(n_1N_2+N_2-1) \rightarrow \text{DFT}\ N_1 \rightarrow W_N^{k_1(N_2-1)} \rightarrow \text{DFT}\ N_2 \rightarrow \text{DFT}\ N_2 \rightarrow X(k_2N_1+N_1-1)
\]
Radix-\(r\) Cooley – Tukey algorithm

- Even if the CT algorithm works with any factorization of the number of samples \(N\), the most popular realizations are those where \(N = r^Q\).
- These algorithms are referred to as \textit{radix-\(r\)} algorithms. We have already seen two versions (DIT and DIF) of the radix-2.
- The radix-2 CT DIT algorithm is then obtained by choosing the following index transformation:

\[
N_1 = N/2, \quad N_2 = 2, \quad n_1 = 0, \ldots, N/2, \quad n_2 = 0,1, \quad A = N_2 = 2, \quad B = 1
\]
Radix-2 DIT CT algorithm revisited

• With the above definitions we have:

\[
X(k) = \sum_{n=0}^{N-1} x(n)W_N^{nk} = \sum_{n_2=0}^{N/2-1} \sum_{n_1=0}^{N/2-1} x(n_1 \cdot N_2 + n_2)W_N^{k(n_1 \cdot N_2 + n_2)}
\]

\[
X(k) = \sum_{n_2=0}^{1} \left[ \sum_{n_1=0}^{N/2-1} x(2 \cdot n_1 + n_2)W_{N/2}^{kn_1} \right]W_N^{kn_2}
\]

\[
X(k) = \sum_{n_1=0}^{N/2-1} x(2 \cdot n_1)W_{N/2}^{kn_1} + W_N^{k} \sum_{n_1=0}^{N/2-1} x(2 \cdot n_1 + 1)W_{N/2}^{kn_1}
\]

- which is the same result obtained by dividing the input samples in even- and odd-index samples

- In a similar way, the radix-2 DIF CT algorithm can be obtained with the following choices:

\[N_1 = N/2; \quad N_2 = 2; \quad n_1 = 0, \ldots, N/2 - 1; \quad n_2 = 0, 1; \quad A = 1; \quad B = N_1 = N/2;\]
Radix-4 DIT CT algorithm

• The radix-4 CT algorithm is also a popular one because the basic cell requires no multipliers.

• The basic cell is that corresponding to the DFT of length equal to the radix (N=r). For the radix-2 CT it is the butterfly cell:

\[
X(0) = x(0) + x(1)
\]

\[
X(1) = x(0) - x(1)
\]

• The basic cell of the radix-4 algorithm can be derived applying the index transformation to a N=4 DFT:

\[
N = 4, \quad N_1 = 1, \quad N_2 = 4, \quad n_1 = 0, \quad n_2 = 0,1,2,3, \quad A = N_2 = 4, \quad B = 1
\]

\[
X(k) = \sum_{n_2=0}^{3} x(n_1 \cdot N_2 + n_2) \cdot W_N^{k(n_1 \cdot N_2 + n_2)} = \sum_{n_2=0}^{3} x(n_2) \cdot W_N^{kn_2}
\]

\[
X(k) = x(0) + x(1) \cdot W_4^k + x(2) \cdot W_4^{2k} + x(3) \cdot W_4^{3k}
\]
Radix-4 DIT CT algorithm

• Since with N=4 then k=0,1,2,3, we obtain:

\[
\begin{align*}
X(0) &= x(0) + x(1) + x(2) + x(3) \\
X(1) &= x(0) - j \cdot x(1) - x(2) + j \cdot x(3) \\
X(2) &= x(0) - x(1) + x(2) - x(3) \\
X(3) &= x(0) + j \cdot x(1) - x(2) - j \cdot x(3)
\end{align*}
\]

• whose corresponding signal flow graph is:
Cooley-Tukey algorithm recipe

Given the transform size decomposition $N = N_1 \cdot N_2$, then:

1. Arrange the one-dimensional data $x(n)$ into a 2D array with $N_1$ rows and $N_2$ columns.
2. Perform a $N_1$-DFT on each column individually.
3. Multiply each term on the resultant array by the twiddle factor:
   $$W_N^{k_1 \cdot n_2}$$
   where $k_1$ is the row index and $n_2$ is the column index.
4. Perform a $N_2$-DFT on each row of the resultant array.

This procedure can be applied recursively to the transforms in 2) and 4)
VLSI architectures for FFT processors

Summary:
• Serial radix-r
• Parallel radix-r
• Pipeline radix-r
  – Multi-path Delay Commutator (R2MDC, R4MDC)
  – Single-path Delay Feedback (R2SDF, R4SDF)
• Recent VLSI realization examples
  – Variable-length (128-1024) FFT processor
  – High-speed (2.4 Gs/s) FFT processor
  – FFT processor for OFDM wireless LAN transceiver
  – FFT/IFFT processor for multi-band OFDM UWB
Serial radix-\(r\) architecture

\[(N/r)\cdot\log_r(N)\) iterations to complete an \(N\)-point DFT
Processing element

- The radix-\( r \) FFT processing element is made of a basic DFT cell (a butterfly, for \( r=2 \)) and a twiddle-factor multiplier

- Basic DFT cell:
  - \( r=2 \) \( \Rightarrow \) two complex adders (i.e., four real adders); one complex sign change (i.e., two real sign changes)
  - \( r=4 \) \( \Rightarrow \) 12 complex adders (i.e., 24 real adders); four complex sign change; four \( \pm j \) multiplications (i.e., four real sign changes)

- Twiddle factor multiplier

\[
W_R = \text{Re}(W_N^k); \quad W_I = \text{Im}(W_N^k); \quad x_R = \text{Re}(x); \quad x_I = \text{Im}(x); \quad y_R = \text{Re}(y); \quad y_I = \text{Im}(y);
\]

\[
y = W_N^k \cdot x \Rightarrow \begin{cases} y_R = W_R \cdot x_R - W_I \cdot x_I \\ y_I = W_R \cdot x_I + W_I \cdot x_R \end{cases}
\]

- four multipliers, one adder, one subtractor
- multiplications dominate the TDF complexity
Efficient twiddle factor multiplier

- Based on the fact that read-only data and adders are less expensive than multipliers.
- Instead of using the real and imaginary part of the TF’s ($W_R$ and $W_I$), precompute and store the following coefficients:
  $$W_R; \quad W_\Sigma = W_R + W_I; \quad W_\Delta = W_R - W_I;$$
- Then, compute the intermediate data:
  $$E = W_R \cdot (x_R - x_I);$$
- Finally, compute the real and imaginary part of the result:
  $$y_R = W_\Delta \cdot x_I + E = (W_R - W_I) \cdot x_R + W_R \cdot (x_R - x_I) = W_R \cdot x_R - W_I \cdot x_I$$
  $$y_I = W_\Sigma \cdot x_R - E = (W_R + W_I) \cdot x_R - W_R \cdot (x_R - x_I) = W_R \cdot x_I + W_I \cdot x_R$$
- The algorithm uses three multiplications, one addition, two subtractions, and an additional storage location for each TF value.
VLSI architectures for FFT processors

• Parallel radix-\(r\) architecture
  – direct realization of the FFT signal flow graph
  – every operation is bound to a its own physical resource (TF multiplier, adder, sign change)
  – computations can be performed in a single clock cycle
  – viable only for short FFT’s \((N \leq 16)\); often used in large FFT’s as sub-processing block

• Example: radix-2, \(N=16\)
  – \(8\) non-trivial (i.e., excluding \(\pm 1, \pm j\)) TF multiplications \(\Rightarrow\) using the efficient TFM seen before this requires \(8 \times 3 = 24\) multipliers and \(8 \times 3 = 24\) adders/subtractors
  – \((N/2) \cdot \log_2(N) = 32\) butterflies with \(4\) adders/subtractors each \(\Rightarrow 128\) adders/subtractors
VLSI architectures for FFT processors

- **Pipeline radix-$r$ architectures**
  - real-time, non-stopping processing as the data sequence enters the processor
  - less hardware with respect to a fully parallel implementation, but higher latency

- **Practical examples of pipeline FFT processors**
  - Radix-2 Multi-path Delay Commutator – R2MDC (example with $N=16$)
    - shift registers
    - commutators
    - requires $\log_2 N - 2$ TF multipliers, $\log_2 N$ butterflies, $(3/2)N - 2$ registers
    - multipliers and butterflies effective usage is 50%
VLSI architectures for FFT processors

- Practical examples of pipeline FFT processors
  - Radix-2 Single-path Delay Feedback – R2SDF (example with N=16)

![Diagram of R2SDF FFT processor](image)

- same number of multipliers and butterflies as R2MDC but less registers
  (only N-1, the minimum possible for an N-point FFT)
Radix-2 Single-Path Delay Feedback explained

$t=0$

$x(N-1) \ldots x(1) \times x(0)$

$t=\frac{N}{2}$

$x(0) \times x(1) \times x(\frac{N}{2}-1)$

$t=N$

$x(0)+x(\frac{N}{2}) \times x(1)+x(\frac{N}{2}+1) \times x(N-1)$

$t=N+N/2$

$0 \times 0 \times 0$

$t=0$

$y(N-1) \ldots y(1) \times y(0)$

$W_N^0 \times \eta_0 = 0, \ldots, \frac{N}{2} = 1$
VLSI architectures for FFT processors

- Practical examples of pipeline FFT processors
  - Radix-4 Single-path Delay Feedback – R4SDF (example with N=256)

  - employs CORDIC to realize TF multiplications
  - $3 \times M$, with $M=64, 16, 4, 1$, means 3 shift registers of depth $M$
  - requires $\log_4 N - 1$ multipliers, $\log_4 N$ radix-4 butterflies, $N-1$ registers
  - resources usage is 75% for multipliers, 25% for butterflies
VLSI architectures for FFT processors

• Practical examples of pipeline FFT processors
  – **Radix-4 Multi-path Delay Commutator** – R4MDC (example with N=256)

  – radix-4 version of the R2MDC
  – requires a larger number (5/2N-4 vs N-1) of registers then R4SDF, but a simpler control unit
  – low (25%) utilization rate of all components
  – requires $3 \cdot \log_4 N$ multipliers, $\log_4 N$ radix-4 butterflies
VLSI architectures for FFT processors

• Practical examples of pipeline FFT processors
  – Radix-4 Single-path Delay Commutator – R4SDC (example with N=256)

  – uses a modified radix-4 algorithm with programmable radix-4 butterflies
  – achieves a 75% utilization rate of multipliers
  – the combined Delay-Commutators (DC6xM, M=64, 16, 4, 1) reduce the memory requirements to 2N-2 registers
  – on the down side, the programmable butterflies and the delay-commutators are relatively complicated; the control unit is also more complex then in the architectures seen earlier
VLSI architectures for FFT processors

• Practical examples of pipeline FFT processors
  – Radix-2² Single-path Delay Feedback – R²2SDF
  – mixed approach between radix-2 and radix-4
  – multiplier structure and utilization similar to radix-4 algorithm
  – butterfly structure of radix-2 algorithm
  – algorithm based on a 3-dimensional index mapping:

\[ N = N_1 \cdot N_2 \cdot N_3; \quad N_1 = N_2 = 2; \quad N_3 = N / 4; \]
\[ n = \left[ n_1 \cdot \left( N / 2 \right) + n_2 \cdot \left( N / 4 \right) + n_3 \right] \mod N; \quad n_1 = 0, 1; \quad n_2 = 0, 1; \quad n_3 = 0, \ldots, N / 4 - 1; \]
\[ k = \left[ k_1 + k_2 \cdot 2 + k_3 \cdot 4 \right] \mod N; \quad k_1 = 0, 1; \quad k_2 = 0, 1; \quad k_3 = 0, \ldots, N / 4 - 1; \]

resulting in a set of 4 DFT of length N/4:

\[ X(k_1 + 2k_2 + 4k_3) = \sum_{n_3 = 0}^{\frac{N}{4} - 1} \left[ H(k_1, k_2, n_3) W^{-n_3(k_1+2k_2)} N \right] W_{\frac{N}{4}}^{n_3k_3} \]
Radix-2\(^2\) Single-path Delay Feedback – (continues)

- the function \(H(k_1, k_2, n_3)\) can be expressed as the combination of two radix-2 butterflies:

\[
H(k_1, k_2, n_3) = \left[ x(n_3) + (-1)^{k_1} x(n_3 + \frac{N}{2}) \right] + (-j)^{(k_1+2k_2)} \left[ x(n_3 + \frac{N}{4}) + (-1)^{k_1} x(n_3 + \frac{3}{4}N) \right]
\]

- the algorithm signal flow diagram (for \(N=16\)) is the following:
VLSI architectures for FFT processors

- Radix-2\(^2\) Single-path Delay Feedback – (continues)
  - the actual implementation of the butterflies includes some programmability:

  - for N=256 the processor architecture is the following:
Real examples of VLSI FFT processors

- Energy optimized, 8 – 16 bit, 128 – 1024 point FFT processor
    - Serial architecture with ultra-low-voltage circuit design
  - chip fabricated in 0.18 μm CMOS technology; contains 627,000 transistors
  - performs 128, 256, 512 and 1024 point DFT with 8 or 16 bit precision
  - energy dissipation is 155 nJ for a 1024 point DFT at $V_{DD} = 350 \text{ mV}$
Real examples of VLSI FFT processors

• High speed FFT processor for MIMO OFDM communications
    – Parallelized radix-2^4 multimode multipath-delay-feedback architecture

  – performs one 256 point DFT at 2.4 Gs/s or 8 256 point DFT at 300 Ms/s
  – 90 nm CMOS, 1.8x1.8 mm^2, 10 bit prec., 132 mW power consumption

Andrea Neviani – P.S.C.D.
Real examples of VLSI FFT processors

- FFT processor for wireless LAN using OFDM
  - block diagram of the physical layer of an 802.11a compatible modem
  - data rate ranges from 1 to 54 Mb/s
  - requires a 64-point FFT/IFFT processor
  - an OFDM symbol duration, i.e., the time available to complete an FFT/IFFT, is 4 μs
  - WLAN clients may be battery-operated, thus minimization of power consumption is a key requirement
Real examples of VLSI FFT processors

- FFT processor for wireless LAN using OFDM (continues)
  - 64-point FFT realized by decomposition into a two-dimensional structure of 8-point FFTs
  - the 8-point FFT units are fully parallel realizations of the DIT signal flow graph
  - computation of the 8-point FFT is carried out in a single clock cycle
Real examples of VLSI FFT processors

- FFT processor for wireless LAN using OFDM (continues)
  - The multiplier unit contains 49 non-trivial twiddle factors, but only eight sets of them are unique.
  - Only the unique TF multipliers are actually realized using a CSD representation.
  - The shuffle network at the input routes the 8 complex data coming from the first 8-point FFT to the right TF multiplier.
  - The shuffle network at the output puts the data back in the right order.
Real examples of VLSI FFT processors

- FFT processor for wireless LAN using OFDM (continues)
  - Chip fabricated in a 0.25 μm BiCMOS technology
  - Requires 23 clock cycles to complete an FFT
  - At 20 MHz clock rate this means 1.15 μs
  - Power consumption is 41 mW at 1.8 V supply
  - Data precision is 16 bit
  - Area: 6.8 mm²
  - Breakdown of area usage:
Real examples of VLSI FFT processors

- FFT/IFFT processor for multi-band OFDM ultrawideband (UWB) applications
    - ADC and DAC sampling rate is up to 528 Msample/s in MB-OFDM UWB
    - the time available for the 128-point FFT/IFFT is only 312.5 ns
    - power consumption and hardware cost should also be minimized
Real examples of VLSI FFT processors

- FFT/IFFT processor for multi-band OFDM ultrawideband (UWB) applications (continues)
  - four-data-path pipelined FFT architecture, called mixed-radix multipath delay feedback (MRMDF)
  - combines the features of the SDF (lower complexity) and MDC (higher throughput) architectures

```plaintext
124 120 ...  8 4 0
125 121 ...  9 5 1
126 122 ... 10 6 2
127 123 ... 11 7 3

The order of the input sequence
```

```plaintext
31 15 ...  8 16 0
95 79 ...  72 80 64
63 47 ...  40 48 32
127 111 ... 104 112 96

The order of the output sequence
```
Real examples of VLSI FFT processors

- FFT/IFFT processor for multi-band OFDM ultrawideband (UWB) applications (continues)
  - Module 1: register that can store 64 pieces of complex data
  - one butterfly unit (BU)
  - two complex multipliers
  - two ROMs
  - some multiplexers
Real examples of VLSI FFT processors

- FFT/IFFT processor for multi-band OFDM ultrawideband (UWB) applications (continues)
  - Module 2: four BU 8 structures in parallel and one modified complex multiplier
  - the BU_8 architecture is directly mapped from the three-step radix-8 (or radix-$2^3$) algorithm
Real examples of VLSI FFT processors

- FFT/IFFT processor for multi-band OFDM ultrawideband (UWB) applications (continues)
  - Module 3: realizes a radix-8 FFT algorithm
  - View of the integrated circuit:
    - Summary of circuit performance:

<table>
<thead>
<tr>
<th>Items</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT size</td>
<td>128 point</td>
</tr>
<tr>
<td>Process Technology</td>
<td>0.18 μm 1p6m</td>
</tr>
<tr>
<td>Package</td>
<td>144 CQFP</td>
</tr>
<tr>
<td>Supply Voltage</td>
<td>3.3 V/1.8V</td>
</tr>
<tr>
<td>Max Work Frequency</td>
<td>250 MHz</td>
</tr>
<tr>
<td>Throughput rate</td>
<td>1G sample/s</td>
</tr>
<tr>
<td>Core Power Consumption</td>
<td>175 mW @ 250 MHz</td>
</tr>
<tr>
<td></td>
<td>77.6 mW @ 110 MHz</td>
</tr>
</tbody>
</table>