PIPELINE IMPLEMENTATIONS OF THE FAST FOURIER TRANSFORM (FFT)

• Consulted work:

DISCRETE FOURIER TRANSFORM (DFT)

• Consider a block of $N$ samples of a (possibly complex-valued) data stream:

$$x[n], n = 0, 1, \ldots N - 1$$

• The discrete Fourier transform of this block is given by:

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j \frac{2\pi k}{N} n}, k = 0, 1, \ldots N - 1$$

INVERSE DFT (IDFT)

• The inverse DFT is almost the same computation as the DFT:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k]e^{j \frac{2\pi k}{N} n}, n = 0, 1, \ldots N - 1$$
TWIDDLE FACTORS

- Define: $W_N = e^{-j \frac{2\pi}{N}}$

- Then DFT becomes: $X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}$

- $W_N^{kn}$ is called a twiddle factor (it is a number on the unit circle in the complex plane).

MATRIX REPRESENTATION OF DFT

- The DFT can be expressed in matrix form:

$$X = \begin{bmatrix}
1 & 1 & 1 & \ldots & 1 \\
1 & W_N & W_N^2 & \ldots & W_N^{N-1} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & W_N^{N-1} & W_N^{2(N-1)} & \ldots & W_N^{(N-1)(N-1)}
\end{bmatrix} \cdot \begin{bmatrix}
x[0] \\
x[1] \\
x[2] \\
x[3] \\
x[4] \\
x[5] \\
x[6] \\
x[7]
\end{bmatrix}$$

- Number of complex multiplications involved (including trivial ones with 1, $j$, etc.): $N^2$

DFT VISUALIZATION

- The DFT basically matches frequencies.
- The following picture originates from Wikipedia (entry: “DFT matrix”):

DECIMATION-IN-TIME FFT (1)

- Split odd and even terms in DFT definition:

$$X[k] = \sum_{m=0}^{N/2-1} x[2m] W_N^{2mk} + \sum_{m=0}^{N/2-1} x[2m+1] W_N^{(2m+1)k}$$

for $k = 0, 1, \ldots, N - 1$
DECIMATION-IN-TIME FFT (2)

- Consider first half of outputs: \( k = 0, 1, \ldots \frac{N}{2} - 1 \)
- Rewrite, using \( W_{N}^{2mk} = W_{N/2}^{mk} \):

\[
X[k] = \sum_{m=0}^{N/2-1} x[2m]W_{N/2}^{mk} + \sum_{m=0}^{N/2-1} x[2m + 1]W_{N/2}^{mk}
\]

DECIMATION-IN-TIME FFT (3)

- DFT has been expressed in terms of half-size DFTs:

\[
X[k] = \sum_{m=0}^{N/2-1} x[2m]W_{N/2}^{mk} + \sum_{m=0}^{N/2-1} x[2m + 1]W_{N/2}^{mk}
\]

DECIMATION-IN-TIME FFT (4)

- Now consider second half of outputs:

\[
X\left[ k + \frac{N}{2} \right] = \sum_{m=0}^{N/2-1} x[2m]W_{N/2}^{m(k+N/2)} + \sum_{m=0}^{N/2-1} x[2m + 1]W_{N/2}^{m(k+N/2)}
\]

\[
k = 0, 1, \ldots \frac{N}{2} - 1
\]

DECIMATION-IN-TIME FFT (5)

- Make use of the following identities:

\[
W_{N/2}^{m(k+N/2)} = W_{N/2}^{mN/2}W_{N/2}^{mk} = W_{N/2}^{mk}
\]

\[
W_{N}^{k+N/2} = W_{N}^{N/2}W_{N}^{k} = -W_{N}^{k}
\]
DECIMATION-IN-TIME FFT (6)

- The expression for the second half becomes the same as for the first except for a minus sign:

\[
x[k + \frac{N}{2}] = \sum_{m=0}^{N/2-1} x[2m]W_N^{mk} + \sum_{m=0}^{N/2-1} x[2m + 1]W_N^{mk}
\]

DECIMATION-IN-TIME FFT (7)

- One decimation-in-time step defines an \( N \)-point DFT in terms of two \( N/2 \)-point DFTs. They are combined by means of butterflies.

![Diagram of a butterfly](image)

- Applying the principle \( recursively \), results in a computation that consists of butterflies only.

DECIMATION-IN-TIME BUTTERFLY

- elementary FFT operation with two inputs and two outputs
- consisting of one complex multiplication, one complex addition and one complex subtraction:

![Diagram of a butterfly](image)
8-POINT DECIMATION-IN-TIME FFT

\[
\begin{align*}
 & x[0] \quad W_0^0 \quad X[0] \\
 & x[4] \quad W_0^0 \quad X[1] \\
 & x[2] \quad W_0^0 \quad W_0^2 \quad X[2] \\
 & x[6] \quad W_0^0 \quad W_0^2 \quad W_0^4 \quad X[3] \\
 & x[1] \quad W_0^0 \quad W_1^0 \quad X[4] \\
 & x[5] \quad W_0^0 \quad W_1^0 \quad W_1^2 \quad X[5] \\
 & x[3] \quad W_0^0 \quad W_1^0 \quad W_1^2 \quad W_1^4 \quad X[6] \\
 & x[7] \quad W_0^0 \quad W_1^0 \quad W_1^2 \quad W_1^4 \quad X[7]
\end{align*}
\]

COMPLEXITY REDUCTION

- Number of computations has been reduced:
  - From $O(N^2)$ for the DFT
  - To $O(N \log N)$ for the FFT

IMPLEMENTATION ISSUES

- Buffering of input is needed because of block-based nature.
- One could e.g. use a ping-pong memory:
  - While the input is filling one memory, the FFT could consume samples of the other memory.
  - After processing one block of samples, the memories change roles.
- Note the bit reversal of the addresses:
  - Address order can be found from increasing binary addresses read in reverse order $4 = 100_2$ e.g. becomes $1 = 001_2$.

DECIMATION-IN-FREQUENCY FFT (1)

- Split input block in first and second half and consider the outputs with even index:

\[
X[2m] = \sum_{n=0}^{N/2-1} x[n]W_N^{2mn} + \sum_{n=N/2}^{N-1} x[n]W_N^{2mn} \quad m = 0, 1, \ldots \frac{N}{2} - 1
\]
DECIMATION-IN-FREQUENCY FFT (2)

- Shift index in second sum:

\[ X[2m] = \sum_{n=0}^{N/2-1} x[n]W_N^{2mn} + \sum_{n=0}^{N/2-1} x[n+N/2]W_N^{2m(n+N/2)} \]

\[ m = 0, 1, \ldots \frac{N}{2} - 1 \]

DECIMATION-IN-FREQUENCY FFT (3)

- Using simplification rules mentioned earlier:

\[ X[2m] = \sum_{n=0}^{N/2-1} (x[n] + x[n + N/2])W_N^{mn} \]

\[ m = 0, 1, \ldots \frac{N}{2} - 1 \]

DECIMATION-IN-FREQUENCY FFT (4)

- This is a half-size DFT applied to an input stream consisting of pairs of the original input stream:

\[ X[2m] = \sum_{n=0}^{N/2-1} (x[n] + x[n + N/2])W_N^{mn} \]

\[ m = 0, 1, \ldots \frac{N}{2} - 1 \]

DECIMATION-IN-FREQUENCY FFT (5)

- Now consider the outputs with odd index:

\[ X[2m + 1] = \sum_{n=0}^{N/2-1} x[n]W_N^{n(2m+1)} + \sum_{n=0}^{N/2-1} x[n+N/2]W_N^{(2m+1)(n+N/2)} \]

\[ m = 0, 1, \ldots \frac{N}{2} - 1 \]
DECIMATION-IN-FREQUENCY FFT (6)

- With the usual type of simplifications:
  \[ X[2m + 1] = \sum_{n=0}^{N/2-1} (x[n] - x[n + N/2]) W_N^m W_N^{mn/N} \]
  \[ m = 0, 1, \ldots, \frac{N}{2} - 1 \]

- This is a half-size DFT applied on a sequence obtained by taking the difference of pairs of the input and multiplying them with factor \( W_N^m \).

DECIMATION-IN-FREQUENCY FFT (7)

8-POINT DECIMATION-IN-FREQ. FFT

- Similar to the decimation-in-time butterfly, but location of multiplication is now at the output side.

Decimation in time
Decimation in frequency
SUMMARY FFT BASICS

- **Decimation-in-time (DIT) FFT:**
  - “Divide and conquer” approach to DFT, based on grouping even and odd inputs in DFT definition
  - Butterfly: multiply before add/subtract
- **Decimation-in-frequency (DIF) FFT:**
  - “Divide and conquer” approach based on grouping even and odd outputs
  - Butterfly: multiply after add/subtract
- Both are of **radix-2** type: problem size is reduced by 2 at each stage

FFT IMPLEMENTATIONS

- FFTs can be realized on all kind of platforms:
  - Programmable processors
  - Dedicated hardware
- Here attention is paid to one type of implementation, viz. the **FFT pipelines**. An FFT pipeline transforms a computation with a block processing nature into one of a streaming nature.
- Issues of concern:
  - Area
  - Speed
  - Power
  - Memory access

RADIX-2 MULTI-PATH DELAY COMMUTATOR (R2MDC)

- Pipeline solution:
  - “Stream-like” processing of block-based algorithm.
- Examples based on 8-point FFT:
  - solutions scale for higher powers of 2.
- Originally proposed in:
**R2MDC EFFICIENCY**

- If implemented as presented, the pipeline has a 50% hardware utilization:
  - The first butterfly is idle half of the time until it can process new pairs of inputs.
  - All hardware operates at input sample frequency
- Situation can be easily improved by duplicating input buffer and operating it in ping-pong fashion:
  - Hardware utilization jumps to 100%.
  - All hardware operates at half the sample frequency.
- As there are no feedback paths, hardware can be further pipelined to cope with possibly long computational delays.

**RADIX-2 SINGLE-PATH DELAY FEEDBACK (R2SDF)**

- Alternative pipeline solution, with optimized memory size
- Originally proposed in:
**R2SDF ELEMENT**

- Element:
  - Either shifts first half of input in delay buffer and second half of output out of delay buffer (blue);
  - Or shifts second half of input into butterfly together with first half from delay buffer, first half of output to next stage and second half output into delay buffer (red).

**SYMBOLIC REPRESENTATION OF R2SDF ELEMENT**

**R2SDF FOR 8-POINT FFT**

**R2SDF STEP-BY-STEP**

A: \[\ldots 0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7\]
B: \[0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7\]
C: \[0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7\]
D: \[\ldots 0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7\]
R2SDF EFFICIENCY

- It needs to operate at same speed as input sample rate.
- Hardware utilization is 50%.
- Number of delay elements for N-point FFT:
  - R2SDF: \( N - 1 \)
  - R2MDC: \( \frac{3}{2}N - 2 \)
- R2SDF and R2MDC have the same number of (complex) adders and multipliers.

DELAY-BUFFER IMPLEMENTATION

- The straightforward implementation of a delay buffer in hardware would be a shift register.
- Such an implementation is not convenient (think of e.g. a 1024-point FFT):
  - Memory elements implemented by D-flipflops are large!
  - Shifting all data at every clock cycle consumes a lot of energy!
- Much better idea to keep the data in RAM and shift the address pointer(s) instead.

DUAL-PORT RAM DELAY BUFFER

- Dual-port RAM:
  - One read port with its own data and address
  - One write port with its own data and address
- Cyclic buffer: \( L+1 \) locations to store \( L \) items.

DELAY BUFFER WITH TWO SINGLE-PORT RAMS

- Idea is to alternatively read from and write to the two RAMs.
- Two single-port RAMs are cheaper than one dual-port RAM.
- Use LSB of address to connect to “write enable” (wen).
SPECIAL-PURPOSE RAMS

- Replace address decoder of RAM by single-bit shift register.
- Only one D-flipflop has output 1.
- Not much switching activity in shift-register (no waste of power).

RADIX-4 FFT

- Has both “decimation in time” as “decimation in frequency” variants.
- Idea is to express DFT as the combination of 4 DFTs whose sizes are one fourth of the original DFT.
- Takes advantage of the following symmetries:
  \[ W_N^{nk+N/4} = -W_N^{nk+3N/4} = -jW_N^{nk} \]
- This leads to a reduction of the number of multipliers (multiplication by \( j \) can be performed without multiplier).

RADIX-4 DIF (1)

\[
X[4m] = \sum_{n=0}^{N/2-1} (x[n] + x[n + N/4] + x[n + N/2] + x[n + 3N/4])W_N^{mn}
\]

\[
X[4m + 1] = \sum_{n=0}^{N/2-1} (x[n] - jx[n + N/4] + -x[n + N/2] + jx[n + 3N/4])W_N^{mn}W_N^n
\]

\( m = 0, 1, \ldots \frac{N}{4} - 1 \)

RADIX-4 DIF (2)

\[
X[4m + 2] = \sum_{n=0}^{N/2-1} (x[n] - x[n + N/4] + x[n + N/2] - x[n + 3N/4])W_N^{mn}W_N^{2n}
\]

\[
X[4m + 3] = \sum_{n=0}^{N/2-1} (x[n] + jx[n + N/4] + -x[n + N/2] - jx[n + 3N/4])W_N^{mn}W_N^{3n}
\]

\( m = 0, 1, \ldots \frac{N}{4} - 1 \)
**RADIX-4 DIF BUTTERFLY**

Scaling with $+1$, $-1$, $+j$, $-j$

**MULTIPLIER & ADDER COMPLEXITY**

- **Radix-2 decomposition:**
  - Number of stages: $\log_2 N$
  - Multiplications per stage: $N/2$ in total: $\frac{N}{2} \log_2 N$
  - Additions per stage: $N$ in total: $N \log_2 N$

- **Radix-4 decomposition:**
  - Number of stages: $\log_4 N = \frac{1}{2} \log_2 N$
  - Multiplications per stage: $3N/4$ in total: $\frac{3N}{4} \log_2 N$
  - Additions per stage: $3N$ in total: $\frac{3N}{2} \log_2 N$

**RADIX-2² BUTTERFLY**

**RADIX-2² BUTTERFLY EVALUATION**

- Radix-2² has the same number of multipliers as radix-4.
- It has fewer additions: $N \log_2 N$

- In a similar way, a radix-8 butterfly can be decomposed in a radix-2² butterfly.
RADIX-4 MULTI-PATH DELAY COMMUTATOR (R4MDC) PIPELINE FOR 16-POINT FFT

RADIX-4 SINGLE-PATH DELAY FEEDBACK (R4SDF) PIPELINE FOR 16-POINT FFT

EVALUATION RADIX-4 PIPELINES

- The multi-path structure:
  - Has a high memory overhead;
  - Has 100% multiplier utilization;
  - Can operate at one quarter of sample frequency.

- The single-path structure:
  - Has minimal memory;
  - Has a 75% multiplier utilization (multiplier drawn outside butterfly!)
  - Operates at sample frequency.