An Efficient Architecture for Ultra Long FFTs in FPGAs and ASICs

- Architecture optimized for Fast Ultra Long FFTs
- Parallel FFT structure reduces external memory bandwidth requirements
- Lengths from 32K to 256M
- Optimized for continuous data FFTs
- Architecture reduces the algorithm to two smaller manageable FFT engines
- Key Features
  - Uses 2 short manageable FFT engines ($N = N_1 \times N_2$)
  - QDR SRAM, reduce IC count, simultaneous read/write
  - CORDIC to generate rotation twiddle factors
  - Matrix transpose address sequence
  - Structure similar to 2D FFT or mixed radix FFT

Computing N = N$_1$ x N$_2$

The N$_1$ x N$_2$ FFT can be computed as:

$$X[k_1 N_2 + k_2] = \sum_{n_1=0}^{N_1-1} [e^{-j\frac{2\pi n_1 k_2}{N}} (\sum_{n_2=0}^{N_2-1} x[n_2 N_1 + n_1] e^{-j\frac{2\pi n_2 k_2}{N_2}})] e^{-j\frac{2\pi n_1 k_1}{N_1}}$$

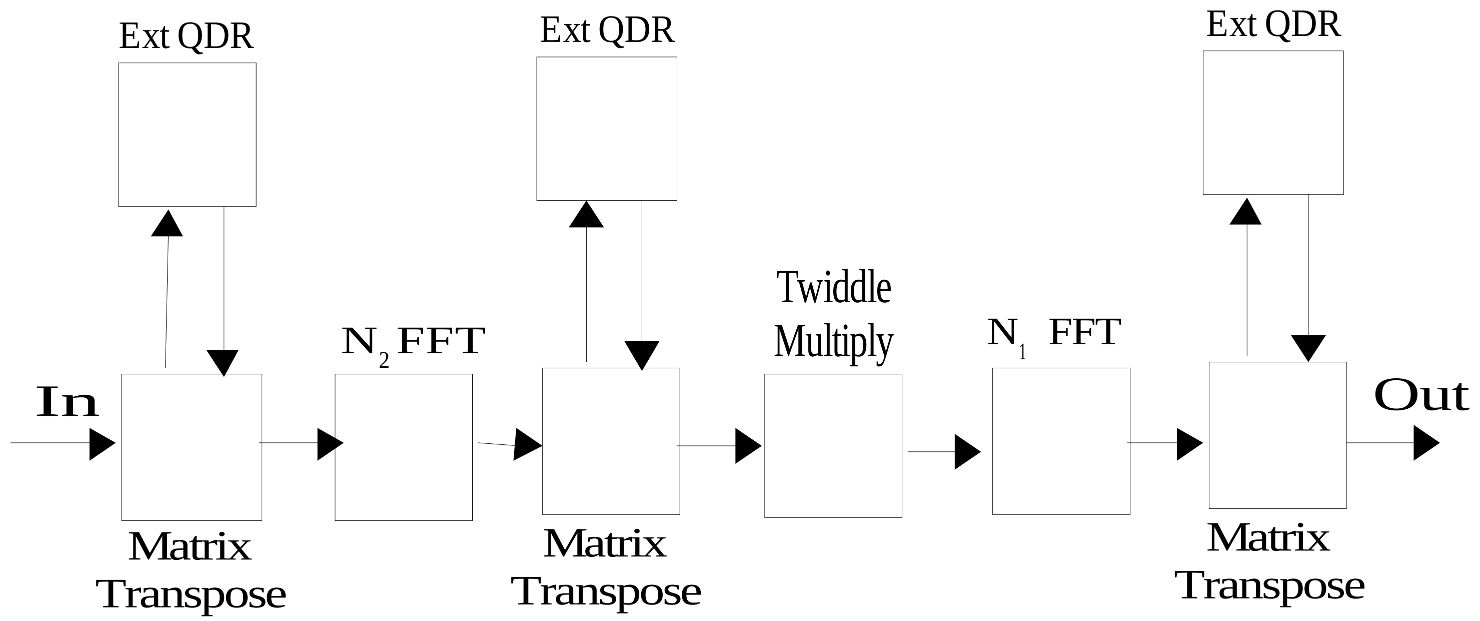
Computing this for:

$$0 \leq k_1 \leq N_1 - 1 \quad \text{and} \quad 0 \leq k_2 \leq N_2 - 1$$

Results in:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi nk}{N}} \quad \text{for} \quad 0 \leq k \leq N-1, \quad \text{as desired}$$

$N = N_1 \times N_2$ Architecture



- Three banks of external QDR Memory (single copy each)
- Two continuous data FFTs (N_1, N_2) inside FPGA
- Twiddle Multiply provides vector rotation between N_2 and N_1 FFTs.
- Final matrix transpose for normal order output.