AN14807

Accelerate FFT Computation with PowerQuad

Rev. 1.0 — 3 October 2025 Application note

Document information

Information	Content
Keywords	AN14807, MCXN947, LPC55S69, PowerQuad, FFT
Abstract	This application note describes how to accelerate FFT computation with PowerQuad DSP.

1 Introduction

In high-performance signal processing applications, the Fast Fourier Transform (FFT) plays a critical role. To enhance efficiency, the LPC55 Series and MCX N Series microcontrollers integrate a hardware accelerator called PowerQuad. This application note presents an approach to accelerate FFT computation using PowerQuad while addressing its limitation of a maximum FFT length of 512 points.

By decomposing larger FFT problems into smaller segments that fit within the 512-point constraint, PowerQuad efficiently performs 1024-point and 2048-point FFT and IFFT operations. This method maintains reliable performance and accuracy.

2 PowerQuad FFT overview

PowerQuad serves as a dedicated hardware accelerator peripheral that offloads and speeds up complex mathematical computations. PowerQuad also optimizes operations such as triangle calculations, matrix manipulations, and digital signal processing (DSP) tasks.

For FFT functionality, PowerQuad integrates a built-in Radix-8 butterfly engine. PowerQuad offers the following advantages:

- High-throughput FFT computation: PowerQuad performs FFTs faster than software implementations, making it ideal for time-sensitive applications, such as audio processing, communications, and control systems.
- Low power consumption: As a hardware accelerator, PowerQuad consumes less energy. This energy efficiency makes it more suitable than software-based FFT solutions, which is essential for battery-powered or thermally constrained environments.
- Data format support: PowerQuad accepts both INT16 and INT32 fixed-point input formats. However, during FFT computations, PowerQuad uses only the bottom 27 bits of the input, which affects precision and scaling considerations.
  Figure 1 shows the Radix FFT butterfly computation in PowerQuad.

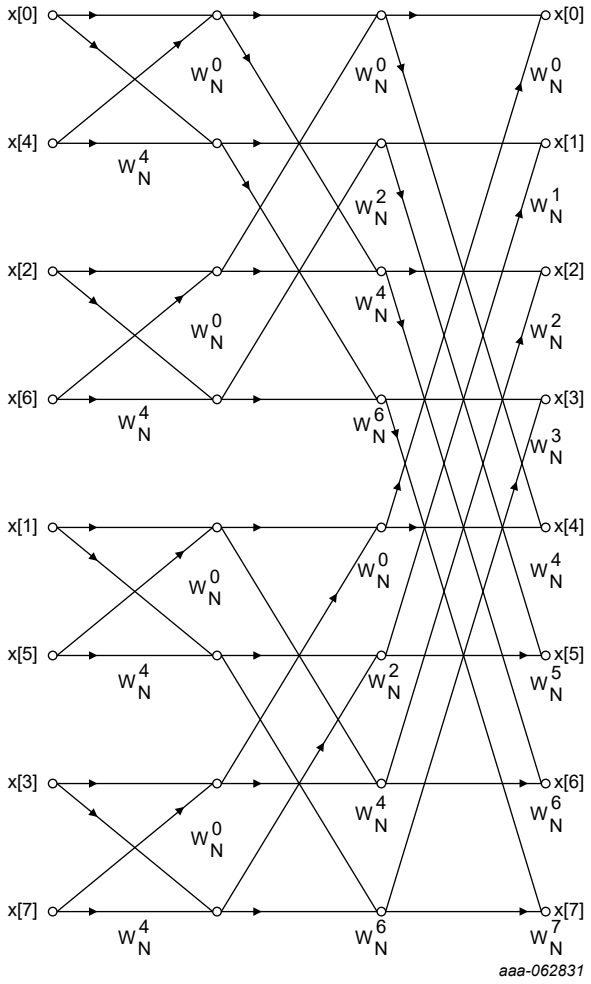

Figure 1. FFT Radix computation

The PowerQuad supports a maximum FFT size of 512 points. To compute larger FFTs, apply a decomposition strategy. By orchestrating multiple 512-point FFTs and combining their results using the Cooley-Tukey Radix-2 decimation-in-time (DIT) algorithm, we extend the capabilities of PowerQuad to support 1024-point, 2048-point, and even larger FFTs.

3 512 points FFT using PowerQuad

PowerQuad provides native support for FFT computations of lengths 16, 32, 64, 128, 256, and 512 points. All these sizes are powers of two and align with its internal Radix-8 FFT engine. When operating within these supported sizes, PowerQuad fully offloads the FFT computation to hardware, delivering maximum performance with minimal software overhead.

PowerQuad accepts both INT16 and INT32 formats for input and output. Internally, it uses only the bottom 27 bits of each input value. Therefore, users must scale or shift their data appropriately to prevent overflow or loss of precision.

PowerQuad also indirectly supports floating-point data. Users can apply its matrix conversion functions to transform float values into fixed-point format before performing FFT computations. This conversion process enables seamless integration with float-based signal processing pipelines.

To perform a 512-point FFT using PowerQuad, follow the steps below:

1. To compute an FFT using PowerQuad, users must first prepare their data first. When using the INT32 input format with PowerQuad, it is essential to understand how the hardware processes the data. It uses only the bottom 27 bits of each 32-bit input value during FFT computation and ignores the upper 5 bits. If the input values are not properly scaled, this limitation can result in overflow or incorrect results.
   If the INT32 input data spans the full 32-bit range, such as data from a high-resolution ADC, external audio input, or a float-to-fixed-point conversion, the upper 5 bits can contain significant data. Since PowerQuad ignores these bits, users must shift them into the usable 27-bit range. Right-shifting by 5 bits preserves the most significant bits within the 27-bit range and prevents overflow or clipping during FFT computation.
   The following code demonstrates how to apply a right shift to INT32 input data to align it with the 27-bit processing range of PowerQuad.

```
for (uint32_t i = 0; i < 512 * 2; i++)
    {
        output[i] = input[i] >> 5;
    }
```

2. Alternatively, when converting from floating-point data, users must first transform the float values into fixed-point format using matrix conversion functions of PowerQuad.

```
    pq_config_t pqConfig = {
        .inputAFormat = kPQ_Float,
        .inputAPrescale = 0,
        .inputBFormat = kPQ_32Bit,
        .inputBPrescale = 0,
        .outputFormat = kPQ_32Bit,
        .outputPrescale = 0,
        .tmpFormat = kPQ_Float,
        .tmpPrescale = 0,
        .machineFormat = kPQ_Float,
        .tmpBase = (uint32_t *)0xE0000000,
    };
    PQ_SetConfig(POWERQUAD, &pqConfig);
    PQ_SetConfig(POWERQUAD, &pqConfig);
    const uint32_t pqLen = POWERQUAD_MAKE_MATRIX_LEN(16, 16, 16);
    PQ_MatrixScale(POWERQUAD, pqLen, 1.0f, (void *)&input[0], (void
 *)&input[0]);
    PQ_WaitDone(POWERQUAD);
    PQ_MatrixScale(POWERQUAD, pqLen, 1.0f, (void *)&input[256], (void
 *)&input[256]);
    PQ_WaitDone(POWERQUAD);
    PQ_MatrixScale(POWERQUAD, pqLen, 1.0f, (void *)&input[512], (void
 *)&input[512]);
    PQ_WaitDone(POWERQUAD);
    PQ_MatrixScale(POWERQUAD, pqLen, 1.0f, (void *)&input[768], (void
 *)&input[768]);
    PQ_WaitDone(POWERQUAD);
```

3. After preparing the input data, users must set the input and output formats, configure the memory addresses, and trigger the FFT computation.

```
const pq_config_t pqConfig = {
        .inputAFormat = kPQ_32Bit,
        .inputAPrescale = 0,
        .inputBFormat = kPQ_32Bit,
        .inputBPrescale = 0,
        .outputFormat = kPQ_32Bit,
        .outputPrescale = 0,
        .tmpFormat = kPQ_32Bit,
        .tmpPrescale = 0,
        .machineFormat = kPQ_32Bit,
```

```
        .tmpBase = (uint32_t *)0xE0000000,
    };
    PQ_SetConfig(POWERQUAD, &pqConfig);
    PQ_TransformCFFT(POWERQUAD, 512, (void *)output, (void *)output);
    PQ_WaitDone(POWERQUAD);
```

4 Larger FFT with PowerQuad

Although PowerQuad supports a maximum FFT length of 512 points, we can compute a 1024-point FFT by decomposing the problem using the Cooley-Tukey Radix-2 DIT algorithm. This method breaks a large FFT into smaller segments that PowerQuad handles directly, followed by a recombination step using twiddle factors.

The Cooley-Tukey Radix-2 DIT algorithm efficiently computes the Discrete Fourier Transform (DFT) of a sequence whose length is a power of two. It recursively divides the input sequence into smaller subsequences, computes FFTs on each, and then combines the results. This recursive decomposition significantly reduces computational complexity from O (N^2) to O ($N\log_2 N$).

To compute a 1024-point FFT, we follow this decomposition and recombination approach, enabling PowerQuad to process larger transforms while maintaining performance and accuracy.

4.1 Split the input sequence

To compute a 1024-point FFT using PowerQuad, follow the process below:

To begin the FFT computation using PowerQuad, the first step involves splitting the input sequence into two 512-point sequences. One contains the even-indexed samples and the other containing the odd-indexed samples. The application then computes two 512-point FFTs using PowerQuad.

In the following code snippet, the application defines two 512-point complex buffers. The application copies the even and odd samples into their respective buffers. It then applies a right shift to each sample to fit within the 27-bit input width of PowerQuad.

```
q31_t *Xe = &tmp[0];    // length 512 * 2
q31_t *Xo = &tmp[1024]; // length 512 * 2

for (size_t i = 0; i < 512 * 2; i += 2)
{
    Xe[i + 0] = input[2 * (i + 0) + 0] >> 5;
    Xe[i + 1] = input[2 * (i + 0) + 1] >> 5;
    Xo[i + 0] = input[2 * (i + 1) + 0] >> 5;
    Xo[i + 1] = input[2 * (i + 1) + 1] >> 5;
}
PQ_TransformCFFT(POWERQUAD, 512, Xe, Xe);
PQ_WaitDone(POWERQUAD);
PQ_TransformCFFT(POWERQUAD, 512, Xo, Xo);
PQ_WaitDone(POWERQUAD);
```

4.2 Apply twiddle factors and combine the results

Twiddle factors are complex exponential coefficients used in FFT algorithms to combine the results of smaller FFTs into a larger one. These coefficients represent phase shifts that correctly align frequency components during recombination.

For a 1024-point FFT, the twiddle factor is defined as:

$$W_N^k = e^{-2\pi ik/1024} \tag{1}$$

Where, k = 0,1,2,…,511

These twiddle factors rotate the output of the FFT performed on the odd-indexed samples before combining them with the FFT results of the even-indexed samples. For each output index k € [0, 511]:

$$X[k] = X_E[k] + W_N^k \times X_O[k] \quad (2)$$

$$X[k+512] = X_E[k] - W_N^k \times X_O[k] \quad (3)$$

Where:

- $X_E[K]$ is the FFT result of the even-indexed input sequence.
- $X_O[K]$ is the FFT result of the odd-indexed input sequence.

To save computation time, the application precomputes twiddle factors and stores them in a lookup table. In this example, a Python script generates the twiddle factors, which are then defined in a separate *C* source file using Q0.31 fixed-point format.

The following code snippet applies twiddle factors to the FFT result of the odd-indexed samples and combines them with the even-indexed FFT output using a matrix function of PowerQuad. The code snippet then converts the result from a fixed-point number to a floating-point number using the scaling feature of PowerQuad.

```
for (uint32_t i = 0; i < 512; i++)
    {
        q31_t a = PQ_TWIDDLE_FACTOR_FFT1024[2 * i + 0];
        q31_t b = PQ_TWIDDLE_FACTOR_FFT1024[2 * i + 1];
        q31_t c = Xo[2 * i + 0];
        q31_t d = Xo[2 * i + 1];

        q31_t ac = __SSAT(((q63_t)a * c) >> 32, 31);
        q31_t ad = __SSAT(((q63_t)a * d) >> 32, 31);
        q31_t bc = __SSAT(((q63_t)b * c) >> 32, 31);
        q31_t bd = __SSAT(((q63_t)b * d) >> 32, 31);

        q31_t ac_sub_bd = __QSUB(ac, bd);
        q31_t ad_add_bc = __QADD(ad, bc);

        Xo[2 * i + 0] = ac_sub_bd;
        Xo[2 * i + 1] = ad_add_bc;
    }

    const uint32_t pqLen = POWERQUAD_MAKE_MATRIX_LEN(16, 16, 16);
    pqConfig.inputAFormat = kPQ_32Bit;
    pqConfig.inputAPrescale = -17;
    pqConfig.inputBFormat = kPQ_32Bit;
    pqConfig.inputBPrescale = -16;
    pqConfig.outputFormat = kPQ_Float;
    pqConfig.outputPrescale = 0;
    pqConfig.tmpFormat = kPQ_Float;
    pqConfig.tmpPrescale = 0;
    pqConfig.machineFormat = kPQ_Float;
    PQ_SetConfig(POWERQUAD, &pqConfig);

    PQ_MatrixAddition(POWERQUAD, pqLen, &Xe[0], &Xo[0], &output[0]);
    PQ_WaitDone(POWERQUAD);
    PQ_MatrixAddition(POWERQUAD, pqLen, &Xe[256], &Xo[256], &output[256]);
    PQ_WaitDone(POWERQUAD);
    PQ_MatrixAddition(POWERQUAD, pqLen, &Xe[512], &Xo[512], &output[512]);
    PQ_WaitDone(POWERQUAD);
    PQ_MatrixAddition(POWERQUAD, pqLen, &Xe[768], &Xo[768], &output[768]);
```

```
    PQ_WaitDone(POWERQUAD);

    PQ_MatrixSubtraction(POWERQUAD, pqLen, &Xe[0], &Xo[0], &output[1024 + 0]);
    PQ_WaitDone(POWERQUAD);
    PQ_MatrixSubtraction(POWERQUAD, pqLen, &Xe[256], &Xo[256], &output[1024 +
256]);
    PQ_WaitDone(POWERQUAD);
    PQ_MatrixSubtraction(POWERQUAD, pqLen, &Xe[512], &Xo[512], &output[1024 +
512]);
    PQ_WaitDone(POWERQUAD);
    PQ_MatrixSubtraction(POWERQUAD, pqLen, &Xe[768], &Xo[768], &output[1024 +
768]);
    PQ_WaitDone(POWERQUAD);
```

5 Performance compares with CMSIS-DSP

The PowerQuad and Cortex microcontroller software interface standard – digital signal processing (CMSIS-DSP) software implementations support complex FFT, real FFT, and inverse FFT. Table 1 presents the performance results of PowerQuad and CMSIS-DSP implementations for CFFT, RFFT, and IFFT functions using fixed-point data. All tested functions use fixed-point number input and output, and the evaluation takes place in MCUXpresso IDE v 24.12 with Optimization Level 3 enabled.

Table 1. PowerQuad performance comparison table

Function	PowerQuad (LPC55 S69)	PowerQuad (MCXN947)	CMSIS-DSP (LPC55 S69)	CMSIS-DSP (MCXN947)
CFFT 512	77 µs	77 µs	806 µs	443 µs
CFFT 1024	239 µs	238 µs	1481 µs	833 µs
CFFT 2048	758 µs	654 µs	3760 µs	2083 µs
RFFT 512	48 µs	47 µs	594 µs	312 µs
RFFT 1024	206 µs	206 µs	1384 µs	727 µs
RFFT 2048	686 µs	584 µs	2636 µs	1400 µs
IFFT 512	71 µs	77 µs	782 µs	441 µs
IFFT 1024	238 µs	238 µs	1479 µs	825 µs
IFFT 2048	751 µs	654 µs	3654 µs	2073 µs

The absolute error and relative error are also tested. Where ***a*** is the reference result calculated by numpy.

$$Err_{absolute} = a - b \tag{4}$$

$$Err_{relative} = (a - b) / b \tag{5}$$

Table 2 shows the maximum absolute and relative errors for PowerQuad CFFT, RFFT, and IFFT computations.

Table 2. PowerQuad computation max error

Function	$Err_{absolute}$	$Err_{relative}$
CFFT 512	< 3e-5	< 1e-5
CFFT 1024	< 6e-5	< 3e-5
CFFT 2048	< 2e-4	< 3e-5
RFFT 512	< 3e-5	< 2e-5

Table 2. PowerQuad computation max error*...continued*

Function	$Err_{absolute}$	$Err_{relative}$
RFFT 1024	< 7e-5	< 4e-5
RFFT 2048	< 2e-4	< 4e-5
IFFT 512	< 4e-8	< 2e-5
IFFT 1024	< 5e-8	< 3e-5
IFFT 2048	< 5e-8	< 4e-5

6 Note about the source code in the document

The example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2025 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN

ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

7 Revision history

Table 3 summarizes revisions to this document.

Table 3. Revision history

Document ID	Release date	Description
AN14807 v1.0	3 October 2025	Initial public release

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Limiting values — Stress above one or more limiting values (as defined in the Absolute Maximum Ratings System of IEC 60134) will cause permanent damage to the device. Limiting values are stress ratings only and (proper) operation of the device at these or any other conditions above those given in the Recommended operating conditions section (if present) or the Characteristics sections of this document is not warranted. Constant or repeated exposure to limiting values will permanently and irreversibly affect the quality and reliability of the device.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

No offer to sell or license — Nothing in this document may be interpreted or construed as an offer to sell products that is open for acceptance or the grant, conveyance or implication of any license under any copyrights, patents or other industrial or intellectual property rights.

Quick reference data — The Quick reference data is an extract of the product data given in the Limiting values and Characteristics sections of this document, and as such is not complete, exhaustive or legally binding.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile — are trademarks and/or registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved.

Contents

1 Introduction 2
2 PowerQuad FFT overview 2
3 512 points FFT using PowerQuad 3
4 Larger FFT with PowerQuad 5
4.1 Split the input sequence 5
4.2 Apply twiddle factors and combine the results 5
5 Performance compares with CMSIS-DSP 7
6 Note about the source code in the document 8
7 Revision history 8
Legal information 9