

Research Article

Engineering: Open Access Dr. Keith Jones, Consultant Mathematician (Retired) Dr. Keith Jones, Consultant Mathematician (Retired)

Schemes for Resource-Efficient Generation of Twiddle Factors for Fixed-Radix FFT Algorithms

Factors for Fixed-Radix FFT Algorithms

requirement, as expressed in terms of the amount of random access memory (RAM) required for constructing the

derived from the sine function and the other from the cosine function. Examples are provided which illustrate the

Keith Jones* *The paper describes schemes for the resource-efficient generation of twiddle factors for the fixed-radix*

Consultant Mathematician (Retired), Weymouth, Dorset, UK versionding Author versionaling at a parallel version (Fig. 1) The paper describes schemes for the resource-efficient generation of twiddle factors for the fixed-radix

Corresponding Author

version of the ubiquitous factority (retired), Weymouth, Dorset, External *Keith Jones, Consultant Mathematician (Retired), Weymouth, Dorset,* UK. *implementation of the FFT, provide one with the facility for trading off arithmetic complexity, as expressed in implementation of the FFT, provide one with the facility for trading off arithmetic complexity, as expressed in*

Submitted: 2024, Jun 25; **Accepted**: 2024, Jul 26; Published: 2024, Jul 30 *terms of the required numbers of multiplications and additions (or subtractions), against the memory requirement, as expressed in terms of the amount of random access memory (RAM) required for constructing the*

Citation: Jones, K. (2024). Schemes for Resource-Efficient Generation of Twiddle Factors for Fixed-Radix FFT Algorithms. *Eng OA, 2*(3), 01-08. *derived from the sine function and the other from the cosine function. Examples are provided which is a function of the sine function.* F_{F} **F**

Abstract *advantages and disadvantages of each scheme – which are very much dependent upon the length of the FFT to be advantages and disadvantages of each scheme – which are very much dependent upon the length of the FFT to be*

The paper describes schemes for the resource-efficient generation of twiddle factors for the fixed-radix version of the ubiquitous $\mathbb{E}[\mathbf{r}_1, \mathbf{r}_2]$ *fast Fourier transform (FFT) algorithm. The schemes, which are targeted at a parallel implementation of the FFT, provide one with* ast router transform (PPT) argortum. The schemes, which are targeted at a parallel implementation of the PPT, provide one with
the facility for trading off arithmetic complexity, as expressed in terms of the required numbe (or subtractions), against the memory requirement, as expressed in terms of the amount of random access memory (RAM) required (or subtractions), against the memory requirement, as expressed in terms of the amount of random access memory
for constructing the look-up tables (LUTs) needed for the storage of the two twiddle factor components – one co derived from the sine function and the other from the cosine function. Examples are provided which illustrate the advantages and derived from the sine function and the other from the cosine function. Examples are provided which titustrate
disadvantages of each scheme – which are very much dependent upon the length of the FFT to be computed – for bot level and multi-level LUTs, highlighting those situations where their adoption might be most appropriate. More specifically, it is seen that the adoption of a multi-level LUT scheme may be used to facilitate significant reductions in memory – namely, from $O(N)$ to an $O(\sqrt[6]{N})$ requirement, for the case of an N-point FFT, where $\beta \geq 2$ corresponds to the number of distinct angular resolutions used – at a relatively small cost in terms of increased FFT latency and arithmetic complexity. θ $\frac{1}{2}$ resour *computed – for both the single-level and multi-level LUTs, highlighting those situations where their adoption*

Keywords: Butterfly, Complexity, FFT, LUT, Parallel, Twiddle Factor $\frac{d}{dx}$ repetitive arithmetic arithmis repetitive arithmetic arit **1. Introduction** *might be most appropriate. More specifically, it is seen that the adoption of a multi-level LUT scheme may be butterfly*, the computational engine used for the computational engine used for the computational engine used for θ *used to facilitate significant reductions in memory in memory in memory in memory in the set of the*

1. Introduction

The fixed-radix version of the ubiquitous fast Fourier transform \tilde{C} (FFT) algorithm [1,2] provides one with an efficient means of solving the discrete Fourier transform (DFT) [1,2], as given for the case of the N-point transform by the expression N-point transform by the expression N-point transform by the expression **Fourier in the interpretative one control**
The fixed-radix version of the ubiquitous fast Fourier trans **1. Introduction** $\frac{1}{2}$ as $\frac{1}{2}$ for $\frac{1}{2}$ for the using fast $\frac{1}{2}$ for $\frac{1}{2}$ *b*utterfly this being the computational engine used for the computational engine used for the computational engine used for the computational engine used of the computational engine used of the computational engine used o

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

for $k = 0, 1, ..., N-1$, where the inputs/outputs are complex-valued and

$$
W_N = \exp(-i2\pi/N), \quad i = \sqrt{-1},
$$
 (2)

the primitive Nth complex root of unity [3]. The complex exponential terms, W_N^{nk} each comprise two trigonometric coefficient mer components – with each pair being more commonly referred to carry out the no as *twiddle factors* – that are required to be fed into each instance after all, the w of the FFT's *butterfly*, this being the computational engine used for carrying out the algorithm's repetitive arithmetic operations [1,2]. Thus, an efficient implementation of the fixed-radix FFT particularly for the processing of large and ultra-large data sets – invariably requires an efficient mechanism for the generation components, or
the primitive Nth complex root of unity [3]. The complex cosine function

Introduction of the twiddle factors which, for a decimation-in-time (DIT) type sform FFT design, with digit-reversed inputs and naturally-ordered ns of outputs, are applied to the butterfly inputs, whilst for a decimation- $\lim_{\varepsilon \to 0} \frac{d\zeta}{dt}$ in-frequency (DIF) type FFT design, with naturally-ordered inputs and digit-reversed outputs, are applied to the butterfly outputs $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ [1,2]. Note that a fixed-radix FFT such as this could also be used $\begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix}$. Note that a fixed-radix 11 i such as this could also be used
to some effect as one component of a prime factor FFT algorithm, where the lengths of the individual small-FFT components are constrained to be relatively prime $[3,4]$. $\frac{1}{2}$ and $\frac{1}{2}$ repetitive

> The twiddle factor requirement, more exactly, is that for a radix-2 FFT algorithm there will be one non-trivial twiddle factor to (2) be applied to each butterfly. The twiddle factor possesses two ϵ , and ϵ outputs to the butterfly outputs ϵ and ϵ the butterfly outputs ϵ components, one defined by the sine function and the other by the compensative, one contract by the state radiation and the either symplex cosine function, which may be either retrieved directly from the netric coefficient memory or generated on-the-fly in order to be able to red to carry out the necessary processing for the FFT butterfly which is, stance after all, the workhorse for the fixed-radix $FFT -$ as is used today the individual small-FFT components are components and the individual small-FFT components are compone processing (DSP) and image processing applications, in real time
exercise to be relatively pression of the EET bowever where R is fashion. With a radix-R version of the FFT, however, where R is an arbitrary integer greater than one, there will be R-1 non-trivial twiddle factors to be applied to each butterfly, rather than just one. 1 $\overline{1}$ $\frac{11}{11}$ a decimation-in-frequency (DIF) type $\frac{11}{11}$ type $\frac{1}{11}$

Thus, the results to be described in this paper – which are targeted, for ease of analysis, at a radix-2 formulation of the FFT – will need to be amended to cater for the increased coefficient memory needed for the generation of the R-1 non-trivial twiddle factors, particularly if a highly-parallel solution to the twiddle factor generation (whereby all the non-trivial twiddle factors are generated and applied simultaneously), and thus to the FFT, is to be achieved.

A radix-R version of the N-point FFT involves a total of log_R (N) stages in the *temporal* domain – where the processing for a given stage can only commence once that of its predecessor has been completed – with each stage involving the computation of N/R

radix-R butterflies in the *spatial* domain. Being independent, in terms of distinct input data sets, enables multiple butterflies to be computed in parallel in the spatial domain via the use of *single-instruction multiple data* (SIMD) type parallel processing techniques [5]. For a fixed-radix version of the FFT such as this a ctors are single butterfly design is required, with its name deriving from the radix-2 design's resemblance to that of a butterfly, as illustrated in Figure 1 – although for a radix-4 algorithm its design more closely resembles that of a *dragonfly* or, for a radix-8 algorithm, that of a spider! Clearly, a mixed-radix version of the FFT [1,2], involving a combination of different radices, such as one exploiting both radix-2 and radix-4 components, would require a commensurate number of distinct butterfly designs. sinusoidal component and one for the LUT address of \cdots is a section \cdots oth component and one for the LUT address of the LUT address of the LUT address of the LUT address of the LUT a the cosinus of contract the contract of the LUT needs to contract the LUT needs to component \mathcal{L}

Figure 1: Illustration of Butterfly for DIT Version of Radix-2 FFT Algorithm - Twiddle Factor Applied to Butterfly Input

For the assessment of these senemes assames the avantasting
of parallel computing equipment, such as that provided by means of a field programmable gate array (FPGA) device, enabling the efficient mapping of the twiddle factor generation – and thus of the *pipelines* for optimum implementational efficiency [6,5]. requirement for the case of Schemes are to be described which enable a simple trade-off in 2. Single-Level LUT Scheme computational complexity to be made between the arithmetic requirement, as expressed in terms of the number of arithmetic operations – denoted C_{M} for multiplications and C_{A} for additions (or subtractions) – required for obtaining the twiddle factors when one or more suitably sized look-up tables (LUTs) are used for their storage, and the memory requirement, as expressed in terms of the amount of random access memory (RAM) [6] – denoted C_{LUT} – required for constructing the one or more suitably sized LUTs. The assessment of these schemes assumes the availability associated fixed-radix FFT – onto suitably defined computational

Summarizing, when just one LUT is used for the twiddle factor between the sine and cosin based upon the adoption of a *single-level* LUT, whereas when more $\cos(x) = \sin(x)$ 3 – the scheme is said to be based upon the adoption of a *multi*- as well as the periodic nature of LUT schemes, the relative advantages and disadvantages of each, storage – as is discussed in Section 2 – the scheme is said to be than one LUT is used for their storage – as is discussed in Section *level* LUT, composed essentially of multiple single-level LUTs [7]. Following these descriptions of the single-level and multi-level which are very much dependent upon the length of the FFT to be computed, are discussed in some detail in Section 4 together with examples highlighting those situations where the adoption of the single-level, two-level and three-level LUT schemes might be most appropriate. Finally, a brief summary and conclusions is provided in Section 5.

2. Single-Level LUT Scheme

the arithmetic As already stated, each twiddle factor comprises two trigonometric components: one sinusoidal and the other cosinusoidal. To emponents. The simulation and the other essimulations. To minimize the arithmetic requirement for the generation of the twiddle factors, a single LUT may be used whereby the sinusoidal and cosinusoidal components are read from a sampled version of the sine function with argument defined from 0 up to 2π radians. As a result, the LUT may be accessed by means of a single, easy to access to compute address which may be updated from one access to another via simple control logic and one addition using a fixed increment – that is, the addresses form an *arithmetic sequence*. $\frac{1}{\pi}$ To a minimum Γ and $\frac{1}{2}$ ι of ms. $\frac{1}{10}$ **3. Multi-Level LUT Schemes** $\frac{1}{2}$ and $\frac{1}{2}$ is strong and one and one addition t ficht $-$ that is, the addresses form an *urinment sequen* arithmetic requirement, for the addressing of the addressing of the addressing of the $L_{\rm H}$ $T₀$ and $T₀$ the such as will be seen in the seen in th $\frac{1}{\sqrt{2}}$ is sufficiently local sections, when $\frac{1}{\sqrt{2}}$ u ms. asy **3. Multi-Level LUT Schemes**

> To achieve a memory-efficient implementation of the fixed-radix FFT, however, it should be noted that the coefficient memory requirement for the case of an N-point transform can be reduced from N to just $N/4$ memory locations by exploiting the relationship between the sine and cosine functions, as given by the expression $\frac{d}{dx}$ ory requirement at the experiment at the experiment at the experiment at the experiment arise of increased are the
increased are the experimental and are the experiment are the experiment are the experiment are the experiment on please twide factors are obtained from the twide factors are obtained from the twide essentially, involves the exploitation of multiple one- δ red at the experiment at the experiment at the experiment at \mathbf{r} $\frac{m}{\rho}$

$$
\cos(x) = \sin\left(x + \frac{1}{2}\pi\right),\tag{3}
$$

as well as the periodic nature of each, as given by the expressions as well as the periodic nature of each, as given by the as well as the periodic nature of each, as given by the $\cos \theta$ expressions $\sum_{n=1}^{\infty}$

$$
\sin(x + 2\pi) = \sin(x) \tag{4}
$$

$$
\sin(x + \pi) = -\sin(x). \tag{5}
$$

These properties enable the twiddle factors to be obtained from a pre-computed trigonometric function defined over a single quadrant of just $\pi/2$ radians rather than over the full range of 2π radians. α defined over a single quadrant of α *coarse-resolution* angular region and to the angle defined over a *fine-resolution* angular region. In achieving such a reduction in the memory requirement ant or just $n/2$ radians rather than over the full range **com** 2π

Thus, for the case of an N-point FFT based upon the adoption of a $\frac{1}{2}$. single LUT, the arithmetic requirement is given by Thus, for the case of an I-point FFT based upon the case of an I-point FFT based upon requirement is given by \mathbf{r} the trigonometric components: Γ omi_t

$$
C_M = 0 \& C_A = 2 \tag{6}
$$

that is, two additions for the generation of each twiddle factor one for the LUT address of the sinusoidal component and one for the LUT address of the sinusoidal component – whilst the LUT α $\frac{1}{\sqrt{1-\epsilon}}$ and the storage of the arithmetic requirement and $\frac{1}{\sqrt{1-\epsilon}}$ and $\frac{1}{\sqrt{1-\epsilon}}$ are the storage of the storage of the twide factors so as to keep the twide factors so as to keep the storage of twide fac requirement of \mathbf{r} th appropriate. Finally, a brief summary and conclusions \mathbf{a} other cosinusoidal. To minimize the arithmetic $r_{\rm th}$ nee rec cosinusoidal components are read from a sampled the sine function with a single s nee accessed by means of a single, easy to compute

$$
C_{LUT} = \frac{1}{4}N\tag{7}
$$

words. This single-quadrant scheme, which exploits a single-level LUT, would seem to offer a reasonable compromise between the arithmetic requirement and the memory requirement, using are antimized requirement and are include the arithmetic, although an approach, although an approximation, although an approximation of memory required for the storage of the twiddle factors so as to keep the arithmetic requirement, for the addressing of the LUT, to a minimum. Most FFT algorithms would invariably adopt such an approach, although as will be seen in the following sections, when the FFT is sufficiently long a multi-level scheme based upon the exploitation of multiple small LUTs might prove more attractive. other cosinus of W $rac{1}{t}$ single LUT may be used when m cosinusoidal components are read from a sampled \mathbf{M} 10 up to 21 accessed by means of a single, easy to compute accessed by means of a single, easy to compute by measure \overline{u} the another via since the control one and o \overline{a} to alt $\frac{5a}{\pi}$ \overline{t} $\frac{1}{2}$ t for \mathbf{r} the coefficient memory M_0 $\frac{a_1}{a_2}$ \int of r \mathbf{r} the expense of increased arithmetic at the expense of increased arithmetic arith

3. Multi-Level LUT Schemes $\overline{\mathbf{3}}$. $\frac{3}{2}$

The aim of the multi-level schemes – which, essentially, involves $\frac{1}{2}$ the exploitation of multiple one-level $LUTs - is$ to reduce the requirement at the expense of increased arithmetic of the state trigonometric identity of the standard trigonometric identity of the standard trigonometric identity of the standard trigonometric in the standard trigonometr complexity. The twiddle factors are obtained from the contents of cale the multiple LUTs through the repeated application of the standard
trigonometric identities trigonometric identities using a fixed increment – that is, the addresses form an $\overline{\text{co}}$ $\frac{1}{\sqrt{1-\frac{1}{2}}}$ th e th between tot ¹ cos x sin x , (3) $\frac{1}{2}$ the as well as the periodic nature of α $\frac{1}{2}$ $\frac{c}{\sqrt{1 + \left(1 + \frac{1}{2}\right)^2}}$ to $\frac{c}{\sqrt{1 + \left(1 + \frac{1}{2}\right)^2}}$ to $\frac{c}{\sqrt{1 + \left(1 + \frac{1}{2}\right)^2}}$

$$
\cos(\theta + \phi) = \cos(\theta) \times \cos(\phi) - \sin(\theta) \times \sin(\phi)
$$
 (8)

$$
\sin(\theta + \phi) = \sin(\theta) \times \cos(\phi) + \cos(\theta) \times \sin(\phi)
$$
 (9) is s

as will be applied directly for the two-level case, where θ corresponds to the angle defined over a *coarse-resolution* angular corresponds to the angle defined over a *coarse-resolution* angular
region and φ to the angle defined over a *fine-resolution* angular and
region. In achieving such a reduction in the memory requirement region and φ to the angle defined over a *fine-resolution* angular and
region. In achieving such a reduction in the memory requirement it is necessary, given M_R different angular resolutions – where the mth resolution is represented by LUT(s) of length S_m – that the *product parameter*, P, obtained from the product of the M_R LUT lengths, is such that re as well as the periodic nature of $\frac{1}{4}$ $\frac{p}{4}$ as r_{eq} $re\epsilon$ $\frac{1}{\sqrt{2}}$ T_{eff} α parameter, α , columned from the product of the α $\frac{d}{d}$ cor $T = \frac{1}{p}$ mth CM = 0 & CA = 2 (6) $\sum_{m=1}^{\infty}$ are the product parameter. P, obtained from the product of the N $\ddot{\ }$. directly for the two-level case, where θ
gle defined over a *coarse-resolution* angular $\frac{1}{1}$ R er. P. obtained from the product of the M LUT worker. α coefficient memory requirement is minimized. For α recuy for $\frac{1}{2}$ directly for the two-level case, where θ I differently for the two-lever case, where σ and the lunch the sum of σ each LUT-based scheme, the parameter α t section that for the multi-level case, where \det each LUT-based scheme, the parameter 1 is clearly scheme, the parameter 1 is clear

$$
P = \prod_{m=1}^{M_R} S_m = \frac{1}{4} N
$$
 (10)
reg
Stimesed by Edt 7, so that the required angular resolution (which is a constant)

as expressed by Eqtn. 7, so that the required angular resolution (is achieved, whilst at the same time ensuring that the *summation* of
narameter. S. obtained from the sum of all the LUT lengths is *parameter*, S, obtained from the sum of all the LUT lengths, is result that such that $\frac{a}{i}$ \mathbf{S} u \overline{a} $\frac{1}{1}$ is a pa $\frac{1}{\pi}$ that $\frac{1}{\pi}$ the LUT lengths, is such that $\frac{1}{\pi}$ is such t

$$
S = \sum_{m=1}^{M_R} \alpha_m \times S_m \text{ is minimized,}
$$
 (11)

where α_m represents the number of LUTs required by the mth angular $\frac{\mathbf{w}}{2}$ **3.1 Two-level Scheme of Lives i** the number of LUTs required by the n μ is and control to μ is required by the μ angular over

The first multi-level scheme involves the adoption

3.1 Two-Level Scheme

of a resolution region, so that the total coefficient memory requirement is minimized. For each LUT-based scheme, the parameter α_1 is clearly equal to one, as there is only one LUT to consider for the ciently equal to one, as there is only one EUT to consider for the coarse-resolution region, whilst it will be seen in this section that for the multi-level case, where $m > 1$, each parameter α_m is equal for the multi-level case, where $m > 1$, each parameter α_m is equal to two as there are two identically sized LUTs that need to be considered for each fine-resolution region - namely, one for the sine function and one for the cosine function. \det s equal categories – two cater for the sine and is, two to cater for the sine and the sine and the sine and the sine a
Executive sine and the sine and

3.1 Two-Level Scheme radians, and one fine-resolution region of length L for

3.1 Two-Level Scheme
The first multi-level scheme involves the adoption of a two-level LUT, this comprising one coarse-resolution region of length N/4L catering for both the sine and cosine functions, covering 0 up to $\pi/2$ radians, and one fine-resolution region of length L for each of the sine and cosine functions, covering 0 up to $\pi/2L$ radians. The required twiddle factors may then be obtained from the contents of the two-level LUT through the application of the standard trigonometric identities, as given by Eqtns. 8 and 9, where θ corresponds to the angle defined over the coarse-resolution region and φ to the angle defined over the fine-resolution region. Ω level $\frac{1}{\pi}$ ancient parallel solution $\frac{1}{2}$ region

By expressing the combined size of the two-level LUT for the sine function as having to cater for

$$
f(L) = \frac{N}{4L} + L \tag{12}
$$

words, where the LUTs are assumed for ease of analysis to be each of length L, it can be seen from the application of the differential of length L, it can be seen from the application of the differential calculus that the optimum LUT length is obtained when the derivative $\frac{1}{\sqrt{8}}$ that the differential calculus $\frac{8}{\sqrt{8}}$ that the differential calculus $\frac{8}{\sqrt{8}}$ that the differential calculus $\frac{8}{\sqrt{8}}$ calculus $\frac{8}{\sqrt{8}}$ calculus $\frac{8}{\sqrt{8}}$ calculus $\frac{8}{\sqrt{8}}$ calculus χ each χ ϵ enthe. To achieve the problem is the problem of problem is the prob

$$
\frac{\text{df}}{\text{d}L} = 1 - \frac{N}{4L^2} \tag{13}
$$

is set to zero, giving \overline{a} of zero, giving

namely

$$
L = \frac{1}{2}\sqrt{N} \tag{14}
$$

and resulting in a total O $\left(\!\sqrt{N}\right)$ memory requirement of where $\mathcal{L}(\Box)$ and resulting in a total $O(N)$ memory requirement of

$$
C_{LUT} = \frac{3}{2} \times \sqrt{N} \tag{15}
$$

words – that is, $\sqrt{N}/2$ to cater for both the sine and cosine functions defined over the coarse-resolution region and $\sqrt{N}/2$ to cater for each of the sine and cosine functions defined over the fine-resolution region [8]. words – that is, $\sqrt{N}/2$ to cater for both the sine and cosine functions of the sine and cosine functions defined over the fine-resolution of the sine and cosine functions defined over the fine-resolution $\sqrt{181}$ σ there is one LUT to consider the is one LUT to consider for σ

This scheme therefore yields a reduced memory requirement (when compared to that for the single-level scheme) for the storage *aation* of the twiddle factors at the expense of an increased arithmetic requirement, namely $T_{\rm eff}$ scheme therefore yields a reduced memory σ requirement (when compared to that for the singleeach fine-resolution region \mathbf{r} This scheme therefore yields a reduced memory required that for the multi-level case, where $\frac{1}{\sqrt{2}}$ function and one for the cosine function.

$$
C_M = 4 \& C_A = 6 \tag{16}
$$

where four of the additions are for generating the LUT addresses where four of the additions are for generating the LOT addresses –
that is, two to cater for both the sine and cosine functions defined
over the coarse-resolution region and two to cater for the sine and over the coarse-resolution region and two to cater for the sine and $\frac{1}{2}$ $\frac{1}{\pi}$ the coarse-resolution region and two to cater for the over the coarse-resolution region and two to cater for the sine and over the coarse-resolution region and two to cater for the sine and \bullet four of the addresses are for gener The first multi-level scheme involves the adoption where four of the additions are f $\mathcal{C} = \mathcal{A} + \mathcal{I}$ and $\mathcal{I} = \mathcal{I}$ and $\mathcal{I} = \mathcal{I}$ where $\frac{1}{2}$ is a distinct and $\frac{1}{2}$ are for generating the LUT $\frac{1}{2}$ ne and cosme ranchons defined

cosine functions, one per LUT, defined over the fine-resolution region. region.

The two-level LUT thus consists of three separate single-level LUTs, each of length $\sqrt{N}/2$, rather than a single LUT, where an efficient parallel solution to the FFT requires that: a) two locations need to be accessed simultaneously from the coarse-resolution LUT; and b) two locations need to be accessed simultaneously from the two fine-resolution LUTs, one per LUT. In addition, for the $\lceil \ln(1) \rceil$ $\mathcal{L}^{(1)}$ $t_{\rm max}$ section that for the multi-level case, where m

efficient mapping of the FFT onto parallel computing equipment it will be necessary for the twiddle factor generation to be carried out which be necessary for any extended actor generation to be carried out.
by means of a suitably defined computational pipeline. To achieve this, the problem must first be decomposed into a number of independent tasks to be performed in the specified temporal order – the solution here involving three independent tasks, as outlined in Figure $2 -$ so that a new twiddle factor may be produced on the completion of the final task.

 $T_{\rm eff}$ scheme therefore yields a reduced memory $T_{\rm eff}$ requirement (when compared to that for the single-Compute LUT addresses and access corresponding trigonometric terms Task 2: T_{cal} 2. where four of the additions are for generating the LUT Combine trigonometric product pairs additively to produce pair of twiddle factor components – one sinusoidal $\&$ one cosinusoidal component – see Eqtns. 8-9 region and two to cater for the sine and cosine Note: parallel processing required for producing simultaneous outputs from each task fine-resolution region. Task 1: Compute set of four trigonometric products from Task 1 outputs – see Eqtns. $8-9$ Task 3: temporal order – the solution here involving three solutions in the solution $\mathcal{L}_\mathbf{t}$ Task 1: $\text{Task } 3$: designed around the clock cycle of the chosen $\mathcal{L}^{\mathcal{L}}$) and $\mathcal{L}^{\mathcal{L}}$ (i.e. $\mathcal{L}^{\mathcal{L}}$) and $\mathcal{L}^{\mathcal{L}}$ cos A B sin C D (19) $\frac{1}{2}$ $\frac{1}{2}$

identities, as \mathbf{r} and 9, where $\overline{\text{tnc}}$ be efficiently carried out by means of a suitably defined internal expressive about the two-level of the three short pipelines device. This, in turn, enables each task to be carried out with a $\mathbf g$ CIOCK C latency, as represented by the combined duration in clock cycles of the three short pipelines. Note, however, that with a flexible computing device, such as an FPGA, each of the arithmetic operations within each task may pipeline, designed around the clock cycle of the chosen computing given latency, as expressed in terms of the required number of clock cycles, with a new twiddle factor being thus produced with every clock cycle at the cost of a time delay, due to the overall h_{tot} a new twide factor being that produced with at the cost of a time delay, due to the overall completion of the final task.

$J₀$ is obtained when the derivative when the derivative $J₀$ is $J₀$ in $J₀$ is a set of $J₀$ is a **3.2 Three-Level Scheme** λ homever, that with a flexible computing computing computing computing computing computing computing computing computing λ note, however, that with a flexible computing computing computing \mathcal{L}

N/4L² for the sine function, covering 0 up to $\pi/2$ radians, and two The next multi-level scheme involves the adoption of a three-
w. level LUT, this comprising one coarse-resolution region of length fine-resolution regions, each of length L, covering 0 up to $\pi/2L$ radians and 0 up to $\pi/2L^2$ radians, respectively, for each of the sine and cosine functions. The required twiddle factors may then be obtained from the contents of the three-level LUT through the double application of the standard trigonometric identities, as
given by Eqtns 8 and 9 so that given by Eqtns. 8 and 9, so that Γ $\pi/2L^2$ radians, respectively, for each of the $\frac{1}{2}$ suitably defined in the substitution, each of length L, covering 0 up to $\pi/2L$ of the standard trigonometric identities, as \ddot{C} to m 22 Tutting, respectively, for each of the \mathbf{p}

$$
\cos(\theta + (\phi_1 + \phi_2)) = \sin(\theta) \times \sin(\phi_1 + \phi_2)
$$
\n
$$
\cos(\theta) \times \cos(\phi_1 + \phi_2) - \sin(\theta) \times \sin(\phi_1 + \phi_2)
$$
\n
$$
\sin(\theta + (\phi_1 + \phi_2)) = \sin(\theta) \times \cos(\phi_1 + \phi_2) + \cos(\theta) \times \sin(\phi_1 + \phi_2),
$$
\n(18)

adoption of a three-level \mathcal{A} three-level LUT, this comprising one comprising one

adoption of a three-level LUT, this comprising one comprising one comprising one comprising one comprising one

an where θ corresponds to the angle defined over the coarse-resolution region and φ_1 and φ_2 to the angles defined over the first and second free resolution, regions, respectively. These equations, may be fine-resolution regions, respectively. These equations may be ing expanded and expressed as

$$
\begin{array}{ll}\n\text{c} \text{ of} & \cos(\theta + (\phi_1 + \phi_2)) = \\
\text{with} & \cos(\theta) \times (A - B) - \sin(\theta) \times (C + D) \\
\text{is} & \sin(\theta + (\phi_1 + \phi_2)) = \\
& \sin(\theta) \times (A - B) + \cos(\theta) \times (C + D) \\
\text{(20)}\n\end{array}
$$

where where where $\frac{1}{2}$ and $\frac{1}{$ \mathfrak{g} where

$$
\text{gth} \qquad \qquad \mathbf{A} = \cos(\phi_1) \times \cos(\phi_2) \tag{21}
$$

$$
B = \sin(\phi_1) \times \sin(\phi_2)
$$
 (22)

$$
C = \sin(\phi_1) \times \cos(\phi_2)
$$
 (23)

the
as
$$
D = cos(\phi_1) \times sin(\phi_2)
$$
. (24)

By expressing the combined size of the three-level LUT for the L_y expressing the comonical size of the time fever L_y sine function as having to cater for cosine functions defined over the coarse-resolution $B₁$ xpressing the combined size of the three-level LU

$$
f(L) = \frac{N}{4L^2} + 2L
$$
 (25)

words, where the LUTs are assumed for ease of analysis to be each of length L, it can be seen that the optimum LUT length is obtained when the derivative when the derivative when the derivative $\mathbf W$ and the each of length of length and the seen that the optimum LUT length is optimum Lut length is obtained when the derivative derivative derivative derivative derivative derivative deriv

³ 2L

2 N

 $\overline{}$

$$
\frac{df}{dL} = 2 - \frac{N}{2L^3}
$$
 where six
other six
that is, two
over the

is set to zero, giving $\frac{1}{18}$ for the sine \ldots \ldots t to zero, giving $\frac{1}{2}$ respectively. is set to zero, giving $\overline{\text{d}}$

 $\overline{}$

The next multi-level scheme involves the

overall latency, as represented by the combined

function, covering 0 up to / 2 radians, and two fine-

$$
L = \sqrt[3]{N/4}
$$
 region an
LUT, defi

and resulting in a total O $\sqrt[3]{N}$ memory requirement of respectively, for each of the sine and cosine functions. and 2 d resulting in a total O $\left(\sqrt[3]{\mathrm{N}}\right)$ memory requirement of $\mathcal{L}^{\mathcal{L}}$ $\mathcal{L}^{\mathcal{L}}$ respectively. For each of the sine sine sine sine \mathbf{a}_1 from the three-level LUT three-level LUT three-level LUT through the standard trigonometrical standard trigonom
The standard trigonometrication of the standard trigonometrication of the standard trigonometrical trigonometr $\overline{\mathbf{a}}$ resulting in a total $O(VN)$ memory requirement The required twiddle factors may then be obtained and resulting in a total $O(\sqrt[3]{N})$ memory requirement of The LUT

$$
C_{\text{LUT}} = 5 \times \sqrt[3]{N/4}
$$
 (28)

words – that is, $\sqrt[3]{N/4}$ to cater for both the sine and cosine functions defined over the coarse-resolution region and $\sqrt[3]{N/4}$ to cater for from each each of the sine and cosine functions defined over each of the two the state of the comparative control the control of the c double application of the standard trigonometric from the three-level lutter of the three double application of the standard trigonometric $\overline{\mathcal{L}}$ (1) $\overline{\mathcal{L}}$ (38) $\overline{\mathcal{L}}$ (38) double application of the standard trigonometric rds – that is, $\sqrt[3]{N/4}$ to cater for both the sine and cosine is double application of the standard trigonometric $d\epsilon$ region and ³ N / 4 to cater for each of the sine and α α regions. double application of the standard trigonometric words – that is, $\sqrt[3]{N/4}$ to cater for both the sine and cosine fur words that $f(x)$, $\gamma(x)$ from the three-level α each of the sine and cosine functions defined over each of the two in a s – that is, $\sqrt[3]{N/4}$ to cater for both the sine and cosine functions LUT region and 3 N $_{\rm equi}$ $\frac{1}{2}$ cu over the coarse-resolution region and $\sqrt{N/4}$ to

This scheme therefore yields a reduced memory requirement (when compared to that for the single-level and two-level schemes) for the probl the storage of the twiddle factors at the expense of an increased outlined in arithmetic requirement, namely ¹ ² ¹ ² cos cos sin sin (17) $\overline{\mathbf{a}}$ 1 **c**c therefore the twiddle factors at the expense of an increased to the single- ¹ ² ¹ ² cos cos sin sin (17) 1 2 sin cos **1** α 1 \mathbf{r} is the single-that for the single-the single-theoretical compared to the single-theoretical compared to the singl $\overline{\textbf{r}}$ corresponds to the company of the control over the control of the control of the control of the angle $\frac{1}{2}$ and 2 to the angle $\frac{$ $\rm c$ cos $\rm c$ where t the angle defined over the angle defined over the angle t coarse-resolution and 1 and 2 to the angles of $ar \overline{a}$ σ scheme therefore yields a reduced memory requirement (when σ and σ suitables arithmetic requirement, namely 101 fine-resolution regions. level and two-level schemes) for the storage of the a single LUT, who are an efficient parallel solution to the continuum of the continuum of the continuum of the $\sum_{i=1}^{n}$ for $\sum_{i=1}^{n}$ the set of better to be accessed to be ac

$$
C_M = 8 \quad \& \quad C_A = 10 \tag{29}
$$

where six of the additions are for generating the LUT addresses – (26) (26) that is, two to cater for both the sine and cosine functions defined over the coarse-resolution region, two to cater for the sine and cosine functions, one per LUT, defined over the first fine-resolution region and two to cater for the sine and cosine functions, one per LUT, defined over the second fine-resolution region. and a seen that the best of length L, it can be seen that the seen t optimum Lut length is obtained when the derivative when the derivative \cos

optimum LUT length is obtained when the derivative

The three-level LUT thus consists of five separate single-level LUTs, each of length $\sqrt[3]{N/4}$, rather than a single LUT, where an efficient parallel solution to the FFT requires that: a) two locations need to be accessed simultaneously from the coarse-resolution LUT; and b) two locations need to be accessed simultaneously $\frac{1}{2}$ from each of the two pairs of fine-resolution LUTs, one per LUT. In addition, for the mapping of the FFT onto parallel computing equipment it will be necessary, as with the two-level scheme, for the twiddle factor generation to be carried out by means of a suitably defined computational pipeline. To achieve this, s) for the problem is first decomposed into five independent tasks, as outlined in Figure 3, so that a new twiddle factor may be produced on the completion of the final task. The internal pipelining of the arithmetic operations within each task then enables a new twiddle factor to be produced with every clock cycle at the cost of a time delay, due to the overall latency, as represented by the combined duration in clock cycles of the five short pipelines. function of the three $f(x) = f(x)$ and $f(x) = f(x)$ are solutions. t twide factors at the expense of an increased of an increased t α era α era α era α era α where six of the additions are for generating the additions are for generating the LUT of the LUT of the LUT o ϵ (we define all $\frac{d\mu}{dt}$ cascu cuillicu in Figure 3, so that a first twidule resolution

Figure 3: Twiddle Factor Generation Using Three-Level LUT Scheme

3.3 Arbitrary K-Level Scheme

Finally, the results obtained above for the two-level and three-level r many, an extended to the computer of the computation of the computer schemes may be straightforwardly extended to the general case of an arbitrary K-level scheme. By expressing the combined size of the K-level LUT for the sine function as having to cater for raigniforwardly extended to the general case of

$$
f(L) = \frac{N}{4L^{K-1}} + (K-1)L
$$
 is set to zero, giving
(30)

The three-level LUT thus consists of $\mathcal{L}_\mathcal{A}$

The three-level LUT thus consistent of the three-level LUT thus consists of five separate separate separate se

 \mathbf{F} words, where the LUTs are assumed for ease of analysis to be each $\frac{1}{2}$ analysis to be each $\frac{1}{2}$

of length L, it can be seen that the optimum LUT length is obtained level when the derivative

$$
\frac{dF}{dL} = (K - 1)\left(1 - \frac{N}{4L^{K}}\right)
$$
 (31)

is set to zero, giving

accessed simultaneously from the coarse-resolution

$$
L = \sqrt[K]{N/4}
$$
 (32)

(since $\mathbf{K} > 1$) and resulting in a total $\mathrm{o}(\sqrt[\kappa]{N})$ memory requirement of uirement of 3.4 Discussion $T_{\rm eff}$ the three-level $T_{\rm eff}$ \sqrt{S} (x) and requiring in a total (x) means on require requirement of (since **K** > 1) and resulting in a total $O(\sqrt[k]{N})$ memory requirement of 3.4 Discussion

$$
C_{LUT} = (2K - 1) \times \sqrt[K]{N/4}
$$
 Note
setio
pineli

words – that is, $\sqrt[x]{N/4}$ to cater for both the sine and cosine functions generation which is defined over the coarse-resolution region and $\sqrt[\kappa]{N/4}$ to cater for upon t each of the sine and cosine functions defined over each of the K-1 version fine-resolution regions. the FFT requires that \mathbf{W} accessed since
Since since si \mathbf{f} ww
de
each
fin \mathbf{w} requires that: \mathbf{w} accessed simultaneously from the coarse-resolution \mathbf{e} simultaneously from each of the two pairs of fine-resolution regions. region and K N / 4 to cater for each of the sine and the s
The sine and the si

The computational cost of adopting such a scheme, however, for of N $K > 3$ would increase to fine-resolution LUTs, one per LUTs, $\frac{1}{2}$ the mapping of the \mathbf{F}_{max} equipment it will be necessary, as with the two-level

$$
C_M = 4K-4 \& C_A = 4K-2 \qquad (34)
$$
 of the fixed-radix

where 2K of the additions are for generating the LUT addresses that is, two to cater for both the sine and cosine functions defined assign over the coarse-resolution region and two to cater for the sine and cosine functions, one per LUT, defined over each of the K-1 fine-
resolution regions.
The K-level I UT thus consists of 2K-1 separate single level I UTs. resolution regions. scheme, for the twiddle factor generation to be carried factor generation to be carried factor generation to b
The twide factor generation to be carried factor generation to be carried factor generation to be carried fact
 $\overline{\mathbf{a}}$ pipeline. This is fixed that problem is fixed that \mathbf{c} or $\overline{\mathbf{a}}$ TH

K

W

th

ov

co

re

TH pipeline. To achieve this, the problem is first decomposed into five independent tasks, as outlined in Figure 3, so that a new twiddle factor may be r the coarse-resolution region and two to cater for the sine and a mege μ and μ $\frac{1}{2}$ defined one per lutter equals the $\frac{1}{2}$ $s_{\rm eff}$ the twiddle factor generation to be carried where $2K$ of the additions are for generating the LUT addresses $-$ Also, with each decomposed into five independent tasks, as outlined in Figure 3, so that a new twide factor may be computed for the compute functions of $\frac{m}{2}$, $\frac{m}{2}$ and $\frac{m}{2}$ or $\frac{m}{2}$. produced on the completion of the final tasks. The final tasks of the final tasks of the final tasks. The final task of the final tasks of the fin region and two to contain the sine and cost of the sine and contain α

The K-level LUT thus consists of $2K-1$ separate single level LUTs, discueach of length $\sqrt[\kappa]{N/4}$, rather than a single LUT, where an efficient parallel solution to the FFT requires that: a) two locations need to resolution LUT, of length $\sqrt[3]{N}$, and henceforth the sense of the sense be accessed simultaneously from the coarse-resolution LUT; and length b) two locations need to be accessed simultaneously from each of parameter, P, will still be the K-1 pairs of fine-resolution LUTs, one per LUT. In addition, addition for the mapping of the FFT onto parallel computing equipment it approximately $3.16 \times \sqrt[3]{N}$ will be necessary, as with the two-level and three-level schemes, fixedfor the twiddle factor generation to be carried out by means of a suitably defined computational pipeline. To achieve this, the 4. Com problem is first decomposed into $2K-1$ independent tasks so that a new twiddle factor may be produced on the completion of the final requirement, for task. The internal pipelining of the arithmetic operations within schem the method popertung of the arithmetic operations within each task then enables a new twiddle factor to be produced with every clock cycle at the cost of a time delay, due to the overall every clock cycle at the cost of a time delay, due to the overall latency, as represented by the combined duration in clock cycles of the $2K-1$ short pipelines. Γ ea $\sum_{i=1}^{\infty}$ each then benefit then \mathbf{b} produced with every clock cycle at the cost of a time delay, due to the overall latency, as represented by the combined duration in the five short- Finally, the results obtained above for the twolevel and the schemes may be a set of the schemes may be a set of the schemes may be a set of the schemes may
of the schemes may be a set of the schemes may be a set of the schemes may be a set of the schemes may be a se straightforwardly extended to the general case of an arbitrary Constantined Scheme. By expressing the contraction of \mathbf{S} $\frac{1}{1}$ onto parallel computing of the FFT onto parallel computing of the FFT only parallel computing $\frac{1}{1}$ ea
pa
be b
b)
th Finally, the results obtained above for the two $rac{1}{\pi}$ straighted to the general case of any $\tan \theta$ earbitra. By expressing the combined scheme. By expressing the combined scheme. By expressing the combined sch $\begin{array}{c} \text{lat} \\ \text{of} \end{array}$ raction the coarse-resolution Lutter-resolutional pipeline. To achieve
em is first decomposed into 2K-1 independent tasks
widdle factor may be produced on the completion of problem is first decomposed into 2K-1 independent tasks so that a To illu each task then enables a new twiddle factor to be produced with a range each task then enables a new twiddle factor to be produced with every clock cycle at the cost of a time delay, due to the overall latency, as represented by the $\frac{1}{\sqrt{2}}$ f level and three-level schemes may be \mathbf{f}_t arbitrary K-level scheme. By expressing the combined \mathbf{r} the sine function as \mathbf{r} $\frac{1}{2}$ \sim α analysis to be $\mathbf{1}_6$ \mathbf{f} the final task. The internal task of the internal task. The internal task. The internal task. The internal task. The internal task of the internal task. The internal task of the internal task. The internal task o define define α and the definition problem. $\frac{p}{2}$ $t_{\rm eff}$ so that a new twide factor may be produced on $t_{\rm eff}$

3.4 Discussion suitably defined computational pipeline, there is a

Note that with each of the multi-level schemes discussed in this (33) section, which involves the use of a suitably defined computational pipeline, there is a latency associated with the twiddle factor generation which is dependent upon the length of the FFT and thus which is dependent upon the length of the FFT and $\frac{1}{N/4}$ to cater for upon the length of the pipeline. With regard to the case of a radix-R extract of the K-1 version of the N-point FFT, regardless of how it is implemented - whether via the adoption of a pipeline or a memory-based $\frac{1}{2}$ architecture – the latency has to account for the computation me, however, for of $N_{R} \times log_{R} N$ radix-R butterflies, so that the effect of the additional latency due to the twiddle factor generation on the overall latency ϵ of the fixed-radix FFT will be expected to be minimal [9]. thus upon the length of the pipeline. With regard to the $\frac{1}{2}$ implementation in singlections generation which is dependent upon the length of the FFT and thus of the fixed-radix FFT will be expected to be minimal [9]. pipeline, there is a latency associated with the twiddle factor er via the adoption of a pipeline or a memory-based $\frac{1}{2}$ architecture – the latency has to account for the computation owever, for of $N_{R} \times log_{R} N$ radix-R butterflies, so that the effect of the additional $\frac{1}{2}$ latency due to the twiddle factor generation on the overall latency the fixed-radix FFT will be expected to be minimal [9]. one or other of the multi-level LUT schemes discussed Also, with each such scheme it is possible that the scheme listed in the table. Ways of reducing these $\frac{1}{2}$ fixed to the multi-level and the multi-level architecture in processes discussed architecture or a memory-based architecture or a memory-based architecture in processing or a memory-based of the FFT and thus of the K-1 version of the N-point FFT, regardless of how it is implemented $\frac{1}{2}$ $\left(\frac{1}{2} \right)$

latency associated with the twiddle factor generation

which may each be tacked with a suitably defined with a suitably defined with a suitably defined with a suitably defined with α

For implementation in silicon of both long and

astronomical data) and 230 (as might be encountered in

LUT addresses - Also, with each such scheme it is possible that the fixed length functions defined assigned to each LUT may not necessarily prove to be a positive er for the sine and integer, as is required, so that one or more of the LUT lengths h of the K-1 fine- may need to be modified in order for integer LUT lengths to be obtained that still satisfy the product and summation constraints of Eqtns. 10 and 11. For example, with the three-level scheme ingle level LUTs, discussed in Section 3.2, if rather than constraining all LUTs to be where an efficient of length $\sqrt[3]{N/4}$ (as given by Eqtn. 27), one used instead a coarselocations need to resolution LUT, of length $\sqrt[3]{N}$, and fine-resolution LUTs, each of T; and length $\sqrt[3]{N}/2$, then the constraint on the product (or multiplicative) parameter, P, will still be met whilst the size of the summation (or LUT. In addition, additive) parameter, S, will actually be marginally reduced from approximately $3.16 \times \sqrt[3]{N}$ (which is clearly not an integer), for the be-level schemes, fixed-length case, to just $3 \times \sqrt[3]{N}$. of Eqtns. 10 and 11. For example, v

us consists of 2K-1 separate single level LUTs, discussed in Section 3.2, if rather than obtained that still satisfy the product and summation constraints ting equipment it approximately $3.16 \times \sqrt[3]{N}$ (which is clearly not an integer), for the $N_{\rm eff}$ arithmetic the number of arithmetic t operations in various combinations in \mathbf{r} usly from each of parameter, P, will still be met whilst the size of the summation (or el schemes, fixed-length case, to just $3 \times \sqrt[3]{N}$. so, with each such scheme it is possible that the fixed length $\frac{1}{\sqrt{2}}$ he sine and integer, as is required, so that one or more of the LUT lengths the K-1 fine-
than constrained that still estisfy the product and summation constraints
of $\frac{1}{\sqrt{2}}$ required number of hardware multipliers – which, with signed to each LUT may not necessarily prove to be a positive $\mathbf d$ cost $\mathbf d$ in such large reductions in $\mathbf d$ $\frac{1}{1}$ and the equivalent to the equiv t an constrained, with each such scheme it is possible that the fixed length $\frac{1}{2}$. $\frac{1}{2}$ and $\frac{1}{2}$ $\frac{1}{2}$ $\frac{1}{2}$, then the constraint on the position (or constraint) eed to be modified in order for integer LUT lengths to be number of multiplications would be equivalent to the

achieve this, the 4. Complexity Results for LUT-Based Schemes $\frac{1}{\sqrt{2}}$

To illustrate the trade-off of arithmetic complexity against memory requirement, for both the single-level and multi-level LUT schemes, a set of results is provided – see Table 1 – which deal with a range of radix-2 FFT lengths: 2^{10} (1024), 2^{20} (1,048,576) and 2^{30} $(1,073,741,824)$ which may be regarded as close approximations to 10^3 , 10^6 and 10^9 , respectively, and which may each be tackled with a suitably defined radix- 2^K algorithm such as a radix-2 or radix-4 FFT.

Note: word length adopted for silicon sizing $= 24$ bits

Table 1: Resource Requirements for Different LUT-Based Twiddle Factor Generation Schemes as Required by Radix-2 FFT **Table 1** – resource requirements for different LUT-based twiddle factor **Algorithm**

For implementation in silicon of both long and ultra-long $FFTs - as$ are becoming of increasing interest with the trend in large scale, *big data* applications - such as those transforms of approximate lengths 2^{20} (as might be encountered in processing of astronomical data) and 2^{30} (as might be encountered in processing a of cosmic microwave data), respectively, considerable resources will inevitably be required, as is evidenced from the memory requirements obtained via the single-level LUT scheme listed in the table. Ways of reducing these requirements, therefore, such as via the adoption of one or other of the multi-level LUT schemes discussed here, need to be carefully considered, as the increased arithmetic complexity and pipeline delay (as will be required for a real-time parallel implementation) may be a cost worth paying for such large reductions in memory – namely, from O(N) to $\frac{1}{2}$ an $O(N)$ requirement, for the case of an N-point FFT, where $\beta \geq 1$ 2 corresponds to the number of distinct angular resolutions used [10]. \mathcal{L} concept \mathbf{F} and \mathbf{F} and \mathbf{F} \mathcal{C}

with every clock cycle at time delay, due to a time d

Note that Table 1 lists the number of arithmetic operations involved for various combinations of FFT length and LUT-based scheme. With a fully-parallel hardware implementation of the FFT, however, the number of multiplications would be equivalent to the required number of hardware multipliers - which, with the availability of fast embedded multipliers as provided by an FPGA manufacturer, are particularly resource and energy efficient - namely one hardware multiplier per multiplication, whilst the number of additions (or subtractions) would, in turn, be equivalent to the required number of hardware adders, namely one hardware to the required number of hardware adders, namely one hardware r_{adder} per addition (or subtraction). $\frac{1}{1}$ and $\frac{1}{2}$ constraints of Eqtns. 10 and 11. For example, with the

> Thus, with a fully-parallel hardware implementation, the numbers of arithmetic operations also defines the associated hardware complexity which, with an FPGA, may be expressed very simplistically in terms of the required number of 'slices' of programmable logic, where a slice comprises a number of LUTs (where an LUT in this context is a collection of logic gates hard wired on the device), flip-flops and multiplexers. With the adoption of L-bit fixed-point processing, an L-bit adder may be implemented with just $L/2$ slices and an $(L$ -bit) $\times (L$ -bit) multiplier – whose size equates, essentially, to that of L adders – with approximately $L^{2}/2$ slices. With regard to memory, an L-bit word of single-port RAM (as required for the single-sample addressing of the fine-resolution LUTs) may be implemented with L/2 slices and an L-bit word of dual-port RAM (as required for the double-sample addressing of the coarse-resolution LUTs) with L slices.

> Based upon these sizing figures the number of logic slices needed for the combined resource requirements of arithmetic and memory (but excluding associated control logic) may be expressed as in Table 1, for various combinations of FFT length and LUT-based scheme, where a wordlength of 24 bits has been assumed for purely illustrative purposes. The results highlight the potential benefits to be obtained through the adoption of one or other of the multilevel LUT schemes, particularly for implementation in silicon of both long and ultra-long FFTs. When compared to the single-level

ultra-long scheme, the two-level scheme (for the long FFT example) offers an he trend in approximate reduction in the total silicon sizing of $O(10^2)$ whilst insforms of the three-level scheme (for the ultra-long FFT example) offers a ocessing of reduction of $O(10^5)$ – these results holding true regardless of the adopted word length.

radix-2K algorithm such as a radiations and Conclusions

against memory requirement, for both the single-level α

me listed in The paper has described schemes for the resource-efficient ore, such as generation of twiddle factors for the fixed-radix version of the IT schemes FFT algorithm. The schemes, which are targeted at a parallel e increased implementation of the FFT, provide one with the facility for trading equired for off arithmetic complexity, as expressed in terms of the required orth paying numbers of multiplications and additions (or subtractions), against $m \ O(N)$ to the memory requirement, as expressed in terms of the amount of where $\beta \geq \text{RAM required for constructing the LUTs needed for the storage of }$ utions used the two twiddle factor components – one component being derived from the sine function and the other from the cosine function. Examples have been provided which illustrate the advantages and operations disadvantages of each scheme – which are very much dependent LUT-based upon the length of the FFT to be computed $-$ for both the singletion of the level and multi-level LUTs, highlighting those situations where equivalent their adoption might be most appropriate. More specifically, it which, with has been seen that the adoption of a multi-level LUT scheme may ided by an be used to facilitate significant reductions in memory – namely, gy efficient from O(N) to an $O(N)$ requirement, for the case of an N-point , whilst the FFT, where $\beta \geq 2$ corresponds to the number of distinct angular resolutions used – at a relatively small cost in terms of increased $\frac{1}{2}$ FFT latency and arithmetic complexity [14].

Note that for a radix-R version of the FFT, there will be R-1 nontrivial twiddle factors to be applied to each butterfly, rather than d hardware just one, so that the results obtained and discussed in this paper definition of the FFT, but the FFT countries of the FFT and the FFT.
I very sim- – which have been targeted, for ease of analysis, at a radix-2 formulation of the FFT – will need to be amended to cater for the of program-
formulation of the FFT – will need to be amended to cater for the If program commutation of the FFT with need to be unrelated to called the memory. hard wired for the generation of the R-1 non-trivial twiddle factors. This replication of replication of resources will be necessary, regardless of the LUTmate processes on the two schemes with the twisted of the 2011 speed of the control of the twiddle factor generation (whereby all the twiddle factors are generated and applied simultaneously), and thus to the FFT, is to be achieved. 7

> Finally, note that such techniques as those discussed here for dealing with the fixed-radix FFT could also be used to the same effect with the design of fast solutions to other commonly used orthogonal transforms [11]. This includes the design and implementation of fast algorithms for the efficient computation of the discrete cosine transform (DCT) and the discrete Hartley transform (DHT) where, for the case of the DHT, the regularized version of the fast Hartley transform (FHT) [7,9] involves the design of a single large double butterfly, with eight real-valued inputs/outputs, for its efficient parallel computation. Nearly all such fixed-radix transforms, except those based upon the adoption of non-standard arithmetic techniques – such as CORDIC arithmetic [7,14] – will rely upon the use of a pre-computed trigonometric function for their implementational efficiency [12,13].

References

- 1. [Brigham, E. O., & Morrow, R. E. \(1967\). The fast Fourier](https://doi.org/10.1109/MSPEC.1967.5217220) transform. *[IEEE spectrum, 4](https://doi.org/10.1109/MSPEC.1967.5217220)*(12), 63-70.
- 2. Chu, E., & George, A. (1999). *[Inside the FFT black box: serial](https://books.google.co.in/books?hl=en&lr=&id=30S3kRiX4xgC&oi=fnd&pg=IA1&dq=E.+Chu+%26+A.+George,+%E2%80%9CInside+the+FFT+Black+Box%E2%80%9D,+CRC+Press,+2000&ots=uInuCOcYSP&sig=dqfVhAuyTKwV6IxMEhvd0jMSyhk&redir_esc=y#v=onepage&q=E.%20Chu%20%26%20A.%20George%2C%20%E2%80%9CInside%20the%20FFT%20Black%20Box%E2%80%9D%2C%20CRC%20Press%2C%202000&f=false) [and parallel fast Fourier transform algorithms](https://books.google.co.in/books?hl=en&lr=&id=30S3kRiX4xgC&oi=fnd&pg=IA1&dq=E.+Chu+%26+A.+George,+%E2%80%9CInside+the+FFT+Black+Box%E2%80%9D,+CRC+Press,+2000&ots=uInuCOcYSP&sig=dqfVhAuyTKwV6IxMEhvd0jMSyhk&redir_esc=y#v=onepage&q=E.%20Chu%20%26%20A.%20George%2C%20%E2%80%9CInside%20the%20FFT%20Black%20Box%E2%80%9D%2C%20CRC%20Press%2C%202000&f=false)*. CRC press.
- 3. [Birkhoff, G., & Mac Lane, S. \(2017\).](https://doi.org/10.1201/9781315275499) *A survey of modern algebra.* [AK Peters/CRC Press.](https://doi.org/10.1201/9781315275499)
- 4. [McClellen, J. H., & Rader, C. M. \(1979\).](https://dl.acm.org/doi/abs/10.5555/578523) *Number theory in [digital signal processing.](https://dl.acm.org/doi/abs/10.5555/578523)* Prentice Hall.
- 5. Akl, S. G. (1989). *[The design and analysis of parallel](https://dl.acm.org/doi/abs/10.5555/63471) algorithms.* [Prentice-Hall, Inc..](https://dl.acm.org/doi/abs/10.5555/63471)
- 6. Maxfield, C. (2004). *[The design warrior's guide to FPGAs:](https://books.google.co.in/books?hl=en&lr=&id=dnuwr2xOFpUC&oi=fnd&pg=PP1&dq=C.+Maxfield,+%E2%80%9CThe+Design+Warrior%E2%80%99s+Guide+to+FPGAs%E2%80%9D,+Newnes+(Elsevier),+2004&ots=YoEcdAb8_t&sig=Ne35fIbYoZqDP1euRN-_XJFUcHY&redir_esc=y#v=onepage&q&f=false) [devices, tools and flows](https://books.google.co.in/books?hl=en&lr=&id=dnuwr2xOFpUC&oi=fnd&pg=PP1&dq=C.+Maxfield,+%E2%80%9CThe+Design+Warrior%E2%80%99s+Guide+to+FPGAs%E2%80%9D,+Newnes+(Elsevier),+2004&ots=YoEcdAb8_t&sig=Ne35fIbYoZqDP1euRN-_XJFUcHY&redir_esc=y#v=onepage&q&f=false)*. Elsevier.
- 7. [Jones, K. \(2010\). The regularized fast Hartley transform.](https://link.springer.com/book/10.1007/978-3-030-68245-3) *[Netherlands: Springer.](https://link.springer.com/book/10.1007/978-3-030-68245-3)*
- 8. Hardy, G. H. (1928). *A course of pure mathematics*. Cambridge University Press.
- 9. [Jones, K. J. \(2023\). A Comparison of Two Recent Approaches,](https://www.researchsquare.com/article/rs-3092888/v1) [Exploiting Pipelined FFT and Memory-Based FHT](https://www.researchsquare.com/article/rs-3092888/v1) [Architectures, for Resource-Efficient Parallel Computation of](https://www.researchsquare.com/article/rs-3092888/v1) [Real-Data DFT. Journal of Applied Science and Technology](https://www.researchsquare.com/article/rs-3092888/v1) [\(Open Source\), Vol. 1, No. 2, pp. 46-55, July 2023.](https://www.researchsquare.com/article/rs-3092888/v1)
- 10. [Bennett, C. L. \(2007\). Wilkinson microwave anisotropy](http://www.scholarpedia.org/article/Microwave_anisotropy_probe) probe. *[Scholarpedia, 2](http://www.scholarpedia.org/article/Microwave_anisotropy_probe)*(10), 4731.
- 11. [Ahmed, N., & Rao, K. R. \(2012\).](https://books.google.co.in/books?hl=en&lr=&id=F-nvCAAAQBAJ&oi=fnd&pg=PA1&dq=N.+Ahmed+%26+K.R.+Rao,+%E2%80%9COrthogonal+Transforms+for+Digital+Signal+Processing%E2%80%9D,+Springer,+2012&ots=D9E5O3QXB1&sig=vH_Sv5_dTPpX6XXWo0uZRsa79dk&redir_esc=y#v=onepage&q=N.%20Ahmed%20%26%20K.R.%20Rao%2C%20%E2%80%9COrthogonal%20Transforms%20for%20Digital%20Signal%20Processing%E2%80%9D%2C%20Springer%2C%202012&f=false) *Orthogonal transforms for digital signal processing.* [Springer Science & Business Media.](https://books.google.co.in/books?hl=en&lr=&id=F-nvCAAAQBAJ&oi=fnd&pg=PA1&dq=N.+Ahmed+%26+K.R.+Rao,+%E2%80%9COrthogonal+Transforms+for+Digital+Signal+Processing%E2%80%9D,+Springer,+2012&ots=D9E5O3QXB1&sig=vH_Sv5_dTPpX6XXWo0uZRsa79dk&redir_esc=y#v=onepage&q=N.%20Ahmed%20%26%20K.R.%20Rao%2C%20%E2%80%9COrthogonal%20Transforms%20for%20Digital%20Signal%20Processing%E2%80%9D%2C%20Springer%2C%202012&f=false)
- 12. [Rao, K. R., & Yip, P. \(2014\).](https://books.google.co.in/books?hl=en&lr=&id=fWviBQAAQBAJ&oi=fnd&pg=PP1&dq=K.R.+Rao+%26+P.C.+Yip,+%E2%80%9CDiscrete+Cosine+Transform:+Algorithms,+Advantages,+Applications%E2%80%9D,+Academic+Press,+1990&ots=1uIKGe4mFB&sig=w65zBQ7UVtAc4qQZ-36_wS4QSwg&redir_esc=y#v=onepage&q=K.R.%20Rao%20%26%20P.C.%20Yip%2C%20%E2%80%9CDiscrete%20Cosine%20Transform%3A%20Algorithms%2C%20Advantages%2C%20Applications%E2%80%9D%2C%20Academic%20Press%2C%201990&f=false) *Discrete cosine transform: [algorithms, advantages, applications](https://books.google.co.in/books?hl=en&lr=&id=fWviBQAAQBAJ&oi=fnd&pg=PP1&dq=K.R.+Rao+%26+P.C.+Yip,+%E2%80%9CDiscrete+Cosine+Transform:+Algorithms,+Advantages,+Applications%E2%80%9D,+Academic+Press,+1990&ots=1uIKGe4mFB&sig=w65zBQ7UVtAc4qQZ-36_wS4QSwg&redir_esc=y#v=onepage&q=K.R.%20Rao%20%26%20P.C.%20Yip%2C%20%E2%80%9CDiscrete%20Cosine%20Transform%3A%20Algorithms%2C%20Advantages%2C%20Applications%E2%80%9D%2C%20Academic%20Press%2C%201990&f=false)*. Academic press.
- 13. [Bracewell, R. N. \(Ed.\). \(1986\).](https://dl.acm.org/doi/abs/10.5555/5536) *The Hartley transform.* Oxford [University Press, Inc..](https://dl.acm.org/doi/abs/10.5555/5536)
- 14. [Volder, J. E. \(1959\). The CORDIC trigonometric computing](https://doi.org/10.1109/TEC.1959.5222693) technique. *[IRE Transactions on electronic computers](https://doi.org/10.1109/TEC.1959.5222693)*, (3), [330-334.](https://doi.org/10.1109/TEC.1959.5222693)

Copyright: *©2024 Keith Jones. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.*