CORRELATION AND CONVOLUTION OF IMAGE DATA USING FERMAT NUMBER TRANSFORM BASED ON TWO'S COMPLEMENT

Lars Rockstroh, Michael Klaiber, Sven Simon

University of Stuttgart, Institute of Parallel and Distributed Systems, Germany

ABSTRACT

The fast fermat number transform (FNT) enables fast correlation and fast convolution similar to fast correlation based on fast fourier transform (FFT). In contrast to fixed-point FFT with dynamic scaling, FNT is based on integer operations, which are free of rounding error, and maintains full dynamic range for convolution and correlation. In this paper, a technique to calculate FNT based on two's complement (TFNT) is presented and the correctness of the technique is proven. The TFNT is data flow driven without conditional assignments, which enables high performance pipelined implementations on digital signal processors and field programmable gate arrays. By taking the example of 2D correlation and based on a Radix-4 algorithm, it is shown that TFNT requires less operations than fixed-point FFT as well as less operations than FNT based on the previously presented diminished-1 approach.

***Index Terms*—** fermat number transform, two's complement

1. NOMENCLATURE

$\mathbb{N}_a$**:** set of natural numbers that are representable with $a+1$ bits.

$\mathbb{T}_a$**:** set of two's complements that are representable with $a+1$ bits.

x_z**:** z^{th} bit of x, where $x \varepsilon \mathbb{N}_a$ and $x_z \varepsilon \mathbb{N}_0$ or $x \epsilon \mathbb{T}_a$ and $x_z \epsilon \mathbb{N}_0$, $z \varepsilon \mathbb{N}$ with $a \geq z \geq 0$.

$x_{z1,z2}$**:** $z1^{th}$ bit down to the $z2^{th}$ bit of x, where $x \epsilon \mathbb{N}_a$ and $x_{z1,z2} \epsilon \mathbb{N}_{z1-z2}$ or $x \epsilon \mathbb{T}_a$ and $x_{z1,z2} \epsilon \mathbb{N}_{z1-z2}$, $z1, z2 \epsilon \mathbb{N}$ with $a \geq z1 \geq z2 \geq 0$.

2. INTRODUCTION

Floating point units require significant amounts of hardware resources and as a consequence, many technologies such as many digital signal processors and field programmable gate arrays (FPGA) provide arithmetic units for integer and fixed point processing only. Calculating FFTs on these fixed point architectures requires scaling the interim results to avoid overflows, which worsens the dynamic range of the correlation results.

Fermat number transform (FNT) satisfies the cross-correlation and the convolution theorem which enables fast correlation and fast convolution based on FNT similar to fast correlation based on the fast fourier transform (FFT). In contrast to the FFT, fast FNT is based on modular arithmetic, which enables efficient execution on integer architectures without any loss in accuracy or dynamic range.

Table 1 shows the word length for fixed point FFT and fast FNT that is required to maintain full dynamic range for two scenarios. The numbers shown are based on the analysis of the data flow only and do not include the loss of dynamic range due to roundoff noise that is inherent to fixed-point FFTs.

The first scenario is the correlation of binary image data, which is required in various imaging measurement techniques such as presented in [1]. The second scenario represents the correlation of a 16x16 window, which is a typical interrogation window size in particle image velocimetry calculations [2]. Table 1 shows that the FNT achieves full dynamic range with 17 bits while the FFT requires up to 33 bits to guarantee full dynamic range. Dynamic scaling during fixed-point FFT may reduce the required word length for some input data but does not relax the requirements for input data with many elements of high value.

Table 1. Dynamic range of 2D correlation based on FFT and FNT.

Dynamic range of input data	$0...2^1$	$0...2^4$
Dimension of data block	256x256	16x16
FFT: 1D (row-by-row)	$0..2^8$	$0..2^8$
FFT: 1D (column-by-column)	$0..2^{16}$	$0..2^{12}$
FFT: element-by-element mult.	$0..2^{32}$, 33 bit	$0..2^{24}$, 25 bit
FNT: 1D (row-by-row)	$0..2^8$	$0..2^8$
FNT: 1D (column-by column)	$0..2^8$	$0..2^8$
FNT: element-by-element mult.	$0..2^{16}$, 17 bit	$0..2^{16}$, 17 bit

Modular arithmetic that FNT is based on, requires executing modulo operations on a regular basis in order to maintain the values of interim results in a range that can be represented by the architectures word length. However, for FNT, where the modulus is chosen of the form $m = 2^n + 1$ or $m = 2^{2^n} + 1$, the modulo operation can be implemented with some additional operations such as demonstrated by the diminished-1 approach [3]. In this paper an approach to calculate FNT based on two's complement (TFNT) is presented which requires less arithmetic and logical operations than diminished-1. Since TFNT can replace diminished-1, all FNT datagraphs that are representable in diminished-1, can be adapted to TFNT, including the utilization of the chinese remainder theorem.

An optimization for FNT that was often discussed, is choosing the primitive root of unity in the form $\omega_{N,m} = 2^x$ so that all multiplications with all roots of unity can be replaced by simple shifts. However, there are no more than $2 \cdot \log_2 m$ unique numbers inside the quotient ring $m = 2^n + 1$ that are representable as a power of two (m positives and m negatives). As a consequence, this optimization drastically limits the transform length to $2 \cdot \log_2 m$ due to the fact that the amount of unique roots of unity must match the transform length. In order to avoid this limitation, general multiplications with arbitrary roots of unity are considered here.

The remainder of this paper is organized as follows: Section 3 introduces TFNT by taking the example of 2D correlation. In Section 4, the operations used in Section 3 are explained and their functionality is proven. Section 5 provides performance analysis of

TFNT, FNT based on diminished-1 as well as FFT and the analysis results are compared. Finally, Section 6 concludes this paper.

3. FNT AND CORRELATION BASED ON TWO'S COMPLEMENT

In this section FNT based on the two's complement operations (TFNT) is presented (Subsection 1). Then, the TFNT is used to design a fast correlation including inverse transform (Subsection 2). Without loss of generality modulus is chosen of the form $m = 2^n+1$ and the transform length is I.

3.1. FNT based on Two's Complement (TFNT)

The datapath of FFT and FNT consists of multiplications and additions. Regarding FNT, which is calculated in modular arithmetic, supplementary modulo functionality is required to limit the value range of interim results. By using two's complement for FNT, this modulo operation is equivalent to one single subtraction and during transform, this subtraction is required only once before and after each multiplication. In particular, the results of a multiplication with a root of unity are of the set $\mathbb{T}_{2n}$. Next, a modulo operation is executed and the set is reduced to $\mathbb{T}_{n+1}$. Then, up to 2^{n-1} of those modulo results may be accumulated before the range of the result of the accumulation reaches $\mathbb{T}_{2n}$, which are sufficient additions to implement radix-128 architectures with $n = 8$ or even larger radix butterflies with bigger n. After accumulation, another modulo operation is executed so that the range of the accumulation results is again in $\mathbb{T}_{n+1}$ and the next multiplication with a root of unity follows.

TFNT consists of the following operations:

1. Converting of the roots of unity and of the input data into two's complement representation: The conversion of input data takes place without computational cost. Afterwards, input data is represented as $a(i) \epsilon \mathbb{T}_{n+1} \forall\, i = 0, \ldots, I-1$ and the roots of unity are represented as $\omega(i) \epsilon \mathbb{T}_n$ with $-2^{n-1} \leqq \omega(i) \leqq 2^{n-1} \forall\, i = 0, \ldots, I-1$.
2. Additions inside butterfly operators $z = r + t$ with $r \epsilon \mathbb{T}_{s3}$, $r \epsilon \mathbb{T}_{s1}$ and $t \epsilon \mathbb{T}_{s2}$, where $s1,\ s2,\ s3, \epsilon \mathbb{N}$ and $s3 = s1+s2+1$.: Overflow and underflow are prevented by choosing the word length of the adder accordingly. The operation requires an adder with an accuracy of $s3 + 1$ bits.
3. Multiplications with roots of unity inside butterfly operators $z = r \cdot t$, where $z \epsilon \mathbb{T}_{2n}$, $t \epsilon \mathbb{T}_n$ with $-2^{n-1} \leqq t \leqq 2^{n-1}$ and $r \epsilon \mathbb{T}_{n+1}$ with $-2^n+1 \leq r \leq 2^{n+1}-1$: This multiplication requires a two's complement multiplier with an accuracy of $2n + 1$ bits.
4. Modulo operation $k \, mod \, 2^n+1 = l$, where $k \epsilon \mathbb{T}_{2n}$, $l \epsilon \mathbb{T}_{n+1}$, $-2^n+1 \leq l \leq 2^{n+1}-1$: The modulo operation requires one two's complement subtracter with n bits accuracy. In order to restrict the range of interim results, the modulo operation is executed once after each multiplication and once after each addition that follows a multiplication.

Except from converting the results at the end of an inverse transform from two's complement back into binary representation, the technique consists solely of two's complement operators without any conditional executions. As a consequence, the processing of all butterfly operations of the transform and its inverse are solely based on multiplications, additions and subtractions in two's complement. This enables implementing pipelined designs based on dedicated integer multipliers and adders on graphics card processing units, FPGAs or application specific integrated circuit (ASIC) designs.

3.2. Correlation based on Two's Complement FNT

Based on the TFNT presented in the previous subsection, this subsection deals with the processing steps of a correlation:

1. TFNT transform of both signals that are to be correlated: Two TFNTs are processed.
2. Element-by-element multiplication of the TFNT results $z = r \cdot t$, where $z \epsilon \mathbb{T}_{2n+2}$,$t \epsilon \mathbb{T}_{n+1}$ with $-2^n+1 \leq t \leq 2^{n+1}-1$ and $r \epsilon \mathbb{T}_{n+1}$ with $-2^n+1 \leq r \leq 2^{n+1}-1$: This multiplication requires a two's complement multiplier with an accuracy of $2n + 2$ bits.
3. Modulo operation $k \, mod \, 2^n + 1 \equiv l$, where $k \epsilon \mathbb{T}_{2n+2}$ and $l \epsilon \mathbb{T}_{n+1}$: The operation is partitioned into the following two modulo operations:
   (a) Modulo operation $k \, mod \, 2^n+1 \equiv l1$, where $k \epsilon \mathbb{T}_{2n+2}$ and $l1 \epsilon \mathbb{T}_{n+3}$. The modulo operation requires one two's complement subtracter with $n + 3$ bits accuracy.
   (b) Modulo operation $l1 \, mod \, 2^n+1 \equiv l2$, where $l1 \epsilon \mathbb{T}_{n+3}$ and $l2 \epsilon \mathbb{T}_{n+1}$. The modulo operation requires one two's complement subtracter with $n + 1$ bits accuracy.
4. Inverse TFNT: The normalization of the results, which is achieved by multiplying with the inverse of the transform length $1/I$ is hidden by scaling the roots of unity used for inverse transform.
5. Converting the results of the inverse TFNT from two's complement into natural numbers that cover the range of the input data of the correlation: $b = c$ with $0 \leq b \leq 2^n$, $b \epsilon \mathbb{N}_n$ and $-2^n+1 \leq c \leq 2^{n+1}-1$, $c \epsilon \mathbb{N}_{n+1}$. This operation converts data in two's complement representation back to binary representation that has the value range of the input data. The operation requires one adder and two conditional assignments. However, the conditions are purely based on the sign bit (Algorithm 1) and thus, condition checking does not require a subtraction or addition. The operation requires one adder.

Input: c , where $-2^n+1 \leq c \leq 2^{n+1}-1$ with $c \epsilon \mathbb{T}_{n+1}$
Output: b, where $b \equiv c$ and $0 \leq b \leq 2^n$ with $b \epsilon \mathbb{N}_n$
begin
  if $c < 0$ then $temp = m$; else $temp = -m$
  $res_{interim} = c + m$
  if $res_{interim} < 0$ then $res_{final} = c$
  else $res_{final} = res_{interim}$
  return res_{final}
end

Algorithm 1: Conversion of two's complement into binary representation.

4. FNT OPERATIONS IN TWO'S COMPLEMENT

This section discusses the TFNT operations that were introduced in the previous section. The modulus is chosen of the form $m = 2^n+1$ and the transform length is I. In the following subsections, these operations are described and their functionality is proven.

4.1. Converting of Input Data and Roots of Unity into Two's Complement

The conversion of input data into two's complement consists of adding a leading zero which does not require computational effort. Regarding the roots of unity, all values bigger than 2^{n-1} are subtracted by m in order to save one bit for representation and during calculations. The formal description of both conversions is as follows:

1. Representation of all input data in two's complement: $x_i = y_i \, \forall \, i = 0, \ldots, I-1$, where
   - $y(i)$ are the initial input data with $0 \leqq y(i) \leqq 2^n$ and $y(i) \, \epsilon \, \mathbb{N}_n \; \forall \, i = 0, \ldots, I-1$.
   - $x(i)$ are the input data in two's complement with $0 \leqq x(i) \leqq 2^n$ and $x(i) \, \epsilon \, \mathbb{T}_{n+1} \forall \, i = 0, \ldots, I-1$.
2. Conversion of roots of unity into two's complement: The conversion is only required once and can be prepared in advance:
   $$\omega(i) = \begin{cases} w(i) & w(i) \leqq 2^{n-1} \\ w(i) - m & else \end{cases} \quad \forall \, i = 0, \ldots, I-1, \text{ where}$$
   - w_i are the roots of unity represented in binary representation with $0 \leqq w(i) \leqq 2^n$ and $w(i) \, \epsilon \, \mathbb{N}_n \; \forall \, i = 0, \ldots, I-1$.
   - ω_i are the roots of unity converted into two's complement with $-2^{n-1} \leqq \omega(i) \leqq 2^{n-1}$ and $\omega(i) \, \epsilon \, \mathbb{T}_n \; \forall \, i = 0, \ldots, I-1$.

4.2. Modulo operation

The modulo operation is based on the following approach: $k \, mod \, 2^n + 1 \equiv l$ with $k \, \epsilon \, \mathbb{T}_{2n+q}$, $l \, \epsilon \, \mathbb{T}_{n+q}$ and $q \, \epsilon \, \mathbb{N}$. In order to implement this operation, the parameter q is chosen as follows:

1. For TFNT and inverse TFNT: $k \, mod \, 2^n + 1 \equiv l$ with $k \, \epsilon \, \mathbb{T}_{2n}$, $l \, \epsilon \, \mathbb{T}_{n+1}$, $-2^n + 1 \leq l \leq 2^{n+1} - 1 \Rightarrow q = 0$.
2. For correlation, Section 3.2, 3.(a): $k \, mod \, 2^n + 1 \equiv l1$ with $k \, \epsilon \, \mathbb{T}_{2n+2}$, $l1 \, \epsilon \, \mathbb{T}_{n+3} \Rightarrow q = 2$.
3. For correlation, Section 3.2, 3.(b): $l1 \, mod \, 2^n + 1 \equiv l2$, with $l1 \, \epsilon \, \mathbb{T}_{n+3}$, $l2 \, \epsilon \, \mathbb{T}_{n+1} \Rightarrow q = 0$. In addition, the sign bit of $l1$ is duplicated so that $l1 \, \epsilon \, \mathbb{T}_{2n}$.

Next, the implementation of the modulo operation is presented and based on this implementation, the correctness of the method and the range of l is proven.

1. Implementation:

   Set $x = k_{n-1,0}$ with $x \, \varepsilon \, \mathbb{N}_{n-1}$
   and $y_{n+q,0} = k_{2n+q,n}$ with $y \, \epsilon \, \mathbb{T}_{n+q}$.

   Then $l = x - y$.

2. Proof of correctness:

   From $x = k_{n-1,0}$ and $y_{n+q,0} = k_{2n+q,n}$ follows

   $$\begin{aligned} k \equiv & -2^{2n+q} \cdot k_{2n+q} + k_{2n+q-1,n} + k_{n-1,0} \equiv \\ & - 2^{2n+q} \cdot y_{n+q} + 2^n \cdot y_{n-1+q,0} + x \equiv \\ & 2^n \cdot (-2^{n+q} \cdot y_{n+q} + y_{n-1+q,0}) + x \equiv \\ & 2^n \cdot y + x \end{aligned}$$

   $$2^n \cdot y + x \qquad / - y \cdot (2^n + 1) \equiv x - y.$$

3. Proof that the range of l is: $-2^{n+q} + 1 \leq l \leq 2^{n+q} + 2^n - 1$:

   Set $x \, \epsilon \, \mathbb{N}_{n-1}$, $y \, \epsilon \, \mathbb{T}_{n+1+q}$.

   Investigation for minima and maxima:

   If $x = 0$ and $y = 2^{n+q} - 1$ then $l = -2^{n+q} + 1$.

   If $x = 2^n - 1$ and $y = 2^{n+q} - 1$ then
   $l = -2^{n+q} + 2^n \geq -2^{n+q} + 1$.

   If $x = 0$ and $y = -2^{n+q}$ then $l = 2^{n+q}$.

   If $x = 2^n - 1$ and $y = -2^{n+q}$ then $l = 2^{n+q} + 2^n - 1$.

4.3. Two's complement multiplication $z = r \cdot t$

This subsection proves the value range of the results of the two's complement multiplications that were introduced in the previous section. Here, the first type of multiplication is used for TFNT transform and inverse TFNT transform. The second type is required to implement the element-by-element multiplication of TFNT-based correlations and convolutions.

1. Type 1 (FNT): $z \, \epsilon \, \mathbb{T}_{2n}$, $t \, \epsilon \, \mathbb{T}_n$ with $-2^{n-1} \leqq t \leqq 2^{n-1}$ and $r \, \epsilon \, \mathbb{T}_{n+1}$ with $-2^n + 1 \leq r \leq 2^{n+1} - 1$

   Investigation of multiplication with minima and maxima:

   If $t = -2^{n-1}$ and $r = -2^n + 1$ then
   $z = 2^{2n-1} - 2^{n-1} < 2^{2n} - 1$.

   If $t = -2^{n-1}$, $r = 2^{n+1} - 1$ then
   $z = -2^{2n} + 2^{n-1} > -2^{2n}$.

   If $t = 2^{n-1}$, $r = -2^n + 1$ then
   $z = -2^{2n-1} + 2^{n-1} > -2^{2n}$.

   If $t = 2^{n-1}$, $r = 2^{n+1} - 1$ then
   $z = 2^{2n} - 2^{n-1} < 2^{2n} - 1$.

   From $-2^{2n} < z < 2^{2n} - 1$ follows $z \, \epsilon \, \mathbb{T}_{2n}$.

2. Type 2 (correlation): $z \, \epsilon \, \mathbb{T}_{2n+2}$, $t \, \epsilon \, \mathbb{T}_{n+1}$ with $-2^n + 1 \leq t \leq 2^{n+1} - 1$ and $r \, \epsilon \, \mathbb{T}_{n+1}$ with $-2^n + 1 \leq r \leq 2^{n+1} - 1$

   Investigation of multiplication with minima and maxima:

   If $t = -2^n + 1$ and $r = -2^n + 1$ then
   $z = 2^{2n} - 2^{n+1} + 1 < 2^{2n+1} - 1$.

   If $t = -2^n + 1$ and $r = 2^{n+1} - 1$ then
   $z = -2^{2n+1} + 2^{n+1} + 2^n - 1 > -2^{2n+1}$.

   If $t = 2^{n+1} - 1$ and $r = -2^n + 1$ then
   $z = -2^{2n+1} + 2^{n+1} + 2^n - 1 > -2^{2n+1}$.

   If $t = 2^{n+1} - 1$ and $r = 2^{n+1} - 1$ then
   $z = 2^{2n+2} - 2^{n+2} + 1 < 2^{2n+2} - 1$.

   From $-2^{2n+1} < z < 2^{2n+2} - 1$ follows $z \, \epsilon \, \mathbb{T}_{2n+2}$.

4.4. Addition and Conversion into Binary Representation

The accumulations inside TFNT butterflies are typical two's complement additions. Overflow and underflow must be prevented to guarantee accurate results. This is achieved by choosing the word length of an adder with respect to the maximum range of the input values so that overflow and underflow do not occur. The conversion from two's complement into binary representation, which is applied after inverse TFNT, was discussed in Section 3.2.

5. PERFORMANCE ANALYSIS

The performance of TFNT is analyzed by comparing its operation count with those of fixed-point FFT and the diminished-1 approach by taking the example of 2D correlation with 17 bit word length. The operation count $|Ops|$ consists of additions and multiplications, where an addition counts as one $|Ops_{add} = 1|$. The effort of a multiplication is approximated by an equivalence of 17 additions $|Ops_{mult} = 17|$ due to the word length of 17 bit.

The 2D correlation is based on a single-butterfly radix-4 algorithm used in various FFT core generators for industrial and research applications [4]. Such a butterfly is shown in Fig. 1. In order to implement FNT instead of FFT on this architecture, only the operations inside the radix-4 butterfly are replaced while the rest of the architecture including memory structure remains the same. This is an advantage of FNT in comparison to Winograd transform, which requires a specific order of processing and is more irregular than radix4 [5].

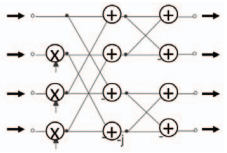

Fig. 1. Radix-4 butterfly.

A 2D correlation with N elements per dimension that is based on a radix-4 algorithm, requires in general the following operations:
$|Ops_{corr}| = 3 \cdot 2 \cdot N \cdot |Ops_{R4-1D}| + |Ops_{e-by-e}|$,
where $|Ops_{R4-1D}|$ is the number of operations required for 1D radix-4 algorithm and $|Ops_{e-by-e}|$ is the operation count for element-by-element multiplication. Three transforms are processed, two for each signal and one inverse transform after element-by-element multiplication.

The number of operations for a 1D radix-4 algorithm with transform length N is defined as
$|Ops_{R4-1D}| = N/4 \cdot \log_4(N)\ butterflies \cdot |ops|/butterfly$,
where the number of operations per butterfly $|ops|/butterfly$ depends on the selected transform [6].

For the **FFT**, a butterfly consists of 8 complex additions (16 real adds) and 3 complex multiplications (3 real adds and 3 real mults [7]). From this it follows that $|Ops|/butterfly =$
$8 \cdot 2 + 3 \cdot (3 + 3 \cdot 17) = 178$, $|Ops_{FFT\_1D}| = 178 \cdot N \cdot \log_4(N)$ and $|Ops_{e-by-e}| = N^2 \cdot (3 + 3 \cdot 17) = 54 \cdot N^2$. Due to the fact, that complex FFT can process two real-value FFTs in parallel, the operation count is divided by two and thus, the operation count is:
$|Ops_{corr\_FFT}| = 133.5 \cdot N^2 \cdot \log_4(N) + 27 \cdot N^2$.

Concerning the **diminished-1**, the analysis is based on the method discussed in [3]. Here, each diminished-1 multiplication is executed in binary representation while additions are executed in diminished-1 representation. As a consequence, a conversion between multiplication and addition as well as between addition and multiplication is required and this conversion consists of one addition. This conversion is required 8 times: 4 times before additions and 4 times after the additions. Also, each addition and multiplication is followed by an subsequent addition, and thus, the operation count per butterfly is: $|Ops|/butterfly = 3 \cdot 17 + 3 + 8 + 8 \cdot 2 = 78$. Element-by-element multiplication consists of the following operations: $|Ops_{e-by-e}| = 18 \cdot N^2$. Thus, the total operation count is:
$|Ops_{corr\_Dim}| = 117 \cdot N^2 \cdot \log_4(N) + 18 \cdot N^2$.

Regarding the **TFNT**, each multiplication of the butterfly requires an subsequent subtraction. Thus, the three multiplications inside the radix4-butterfly require $3 \cdot 17 + 3 = 54$ operations. The $-j$ multiplication can be replaced by an constant shift but requires an subsequent modulo operation (one subtraction). All additions require one operation and each of the four final results of the butterfly requires an subsequent modulo operation. Thus, the operation count is: $|ops|/butterfly = 54 + 1 + 8 + 4 = 67$ and
$|Ops_{FNT\_1D}| = 67 \cdot N \cdot \log_4(N)$. Regarding element-by-element multiplication, TFNT requires two subsequent modulo operations (see Section 3.2). From this it follows that the operation count is $|Ops_{e-by-e}| = N^2 \cdot (2 + 1 \cdot 17) = 19 \cdot N^2$.

In contrast to FFT and diminished-1, TFNT requires additional post-processing as shown in Algorithm 1 in order to back transform the range of results into the range of the input values. The algorithm consists of two comparisons and one addition. However, the comparisons are checked by investigating the MSB (Section 3.2). Thus, the operation count for post processing is approximated by $|Ops_{post}| = 1 \cdot N^2$. Then, the total count of operation is:
$|Ops_{corr\_TFNT}| = 100.5 \cdot N^2 \cdot \log_4(N) + 20 \cdot N^2$.
Based on the analysis of this section, the coefficients of the higher order terms $N^2 \cdot \log_4(N)$ of the operation counts are compared. Here, the operation count for TFNT shows an improvement of 24.7 Percent against FFT and 14.1 percent against the diminished-1 approach.

6. CONCLUSION

In this paper, fermat number transform based on two's complement (TFNT) was introduced. In contrast to fixed-point FFT with dynamic scaling, the TFNT enables convolutions and correlations free of rounding error and with full dynamic range independent from the input data. The technique can be used to utilize dedicated integer units on field programmable gate arrays as well as digital signal processors or integer hard macros during ASIC design. The functionality of the technique was proven and demonstrated by taking the example of fast 2D correlation with 17 bit word length. Based on this example, the performance of TFNT was evaluated and compared to FNT based on diminished-1 and to fast fourier transform (FFT). The comparison shows that TFNT requires 24.7 percent less operations than FFT and 14.1 percent less operations than FNT based on diminished-1.

7. REFERENCES

[1] L. Rockstroh, T. Lefeure, M. Wroblewski, S. Wahl, N. Fortier, and S. Simon, "Simultaneous characterization of particle velocities and sizes based on autocorrelation of filtered motion blurring," in *Liquid Atomization and Spray Systems Europe*, 2011.

[2] C. Gray, "High-speed piv using high-frequency diode-pumped solid state laser and multi-frame ccd," in *Instrumentation in Aerospace Simulation Facilities, ICIASF, Cleveland*, 2001.

[3] M. Benaissa, A. Bouridane, S.S. Dlay, and A.G.J. Holt, "Diminished-1 multiplier for a fast convolver and correlator using the fermat number transform," in *IEE Proceedings G Electronic Circuits and Systems*, 1988, vol. 135(5), pp. 187–193.

[4] Inc. Xilinx, *LogiCore IP Fast Fourier Transform v7.1*, 2011.

[5] S. Winograd, "On computing the discrete fourier transform," in *Math. Computation*, 1978, vol. 32(141), pp. 175–199.

[6] C.S. Burrus, "Unscrambling for fast dft algorithms," in *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 1988.

[7] S.G. Krantz, *Handbook of Complex Variables*, Birkhaeuser Boston, 1999.