Microelectronics Journal 45 (2014) 1311–1318

Contents lists available at ScienceDirect

Microelectronics Journal

journal homepage: www.elsevier.com/locate/mejo

An area-efficient Radix 2^8 FFT algorithm for DVB-T2 receivers

M. Turrillas, A. Cortés *, I. Vélez, J.F. Sevillano, A. Irizar

Department of Electronics & Communications, CEIT and Tecnun (University of Navarra), Manuel de Lardizábal 15, 20018 San Sebatián, Spain

ARTICLE INFO

Article history:
Received 15 February 2013
Received in revised form
5 September 2013
Accepted 30 October 2013
Available online 13 November 2013

Keywords:
Application specific integrated circuits (ASIC)
Digital video broadcasting – second generation terrestrial (DVB-T2)
Fast Fourier transform (FFT)
Orthogonal frequency division multiplexing (OFDM)
Tensor product

ABSTRACT

This paper presents an area-efficient variable-length FFT algorithm for DVB-T2 receivers. A matrix-based approach is used to achieve a novel radix 2^8 algorithm that fulfils the DVB-T2 specifications. Several implementation techniques are proposed to apply in order to reduce the FFT core area, such as a variable datapath scaling approach, a memoryless CORDIC algorithm and an efficient FIFO implementation. The layout of the FFT processor is designed in XFAB 0.18 μm CMOS technology. The proposed variable-length processor occupies a layout area of 6.75 mm^2. Compared with the DVB-T2 designs in the literature, the proposed FFT processor presents the most area-efficient implementation. Furthermore, it provides a good power efficiency in the lower modes.

1. Introduction

The Fast Fourier Transform (FFT) and the inverse fast Fourier transform (IFFT) are key components in orthogonal frequency division multiplexing (OFDM) systems, such as very high-speed digital subscriber line (VDSL), digital audio broadcasting (DAB) and digital video broadcasting-terrestrial (DVB-T). These applications require large point FFT processing for multiple carrier modulation. A second generation digital video broadcasting-terrestrial (DVB-T2) standard [1] has been approved by the European telecommunications standards institute (ETSI) as a new platform for high definition television (HDTV) transmissions for fixed and portable devices. One of the main differences between DVB-T2 and its predecessor, DVB-T, is the number of points of the FFT. DVB-T2 proposes four additional modes (1 K/4 K/16 K/32 K) to the two DVB-T modes (2 K/8 K). Thus, a 1 K/2 K/4 K/ 8 K/16 K/32 K-points FFT is required in DVB-T2.

Several FFT cores can be found for DVB-T [2–11]. To our best knowledge, only three FFT cores have been proposed in the literature for DVB-T2 receivers [12–14]. Refs. [2,3] use monoprocessor architectures, [14,4,5] implement parallel architectures, and [12,13,6–11] present pipeline architectures for the FFT processing.

When the length of the FFT is high, the number of twiddle factors to be stored and the number of twiddle factor multipliers grow with $\log_r(N)$. In order to reduce the number of twiddle factor multiplications, high radices FFT cores have been proposed, e.g., mixed radix 2/4 in [6,3], mixed radix 2/4/8 in [8] and radix 8 in [4]. However, this makes the butterfly more complex. In order to solve this problem, radix 2^2 algorithms have been proposed in [9,2], radix 2^3 and 2^4 in [11], and radix 2^5 in [12,13]. Ref. [15] presents general expressions for designing single-path delay feedback (SDF) pipeline radix r^k algorithms. The analysis in [15] is useful to deduce the hardware complexity of the r^k algorithms at a high level using a matrix-based approach. It allows the designer to compare the algorithms and discard those with a higher complexity. Nevertheless, this high level comparison is not enough to make a final decision.

The large number of points of the highest carrier modes in applications such as DVB-T2 or DVB-T also makes the memory requirements a crucial parameter to minimize. In the literature, different dynamic scaling techniques are proposed in order to increase the signal-to-quantization-noise ratio (SQNR) and to reduce the wordlength in FFT processors with respect to the fixed-point format. [3] use block floating-point (BFP) scaling technique and [4,7] present new floating-point approaches. These methods need additional internal buffers and control. Ref. [16] presents another way to achieve a high SQNR increasing the wordlength of each butterfly in one bit, which is called variable datapath. In [13], this variable datapath is optimized obtaining the minimum wordlength in each butterfly stage to comply with the system

* Corresponding author. Tel.: +34 943 212800.
E-mail addresses: mturrillas@ceit.es (M. Turrillas), acortes@ceit.es (A. Cortés), ivelez@ceit.es (I. Vélez), jfsevillano@ceit.es (J.F. Sevillano), airizar@ceit.es (A. Irizar).

0026-2692/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.mejo.2013.10.016

specifications. This method does not need additional internal buffers or control. Nevertheless, it can be enhanced analyzing in which butterfly stage is necessary to scale data.

In [17], a memoryless COordinate Rotation DIgital Computer (CORDIC) algorithm for the FFT computation is proposed. This approach requires no memory, and thus, the area hardly depends on the number of points, which permits the calculation of FFTs of a large number of points. This algorithm has been developed only for fixed-length radix r algorithms.

The objective of this paper is to achieve the most area-efficient radix r^k algorithm for DVB-T2 receivers. The paper presents a further low level analysis to refine the first algorithm selection performed according to [15]. In order to optimize even more the selected r^k algorithm, this paper proposes a memoryless CORDIC for variable-length radix r^k algorithms and a variable-datapath technique by analyzing in which butterflies data need to be scaled.

In order to accommodate variable-length operation in the FFT processor, the different FFT lengths are analyzed using the matrix-based expressions in [15]. This analysis enables us to design a variable-length implementation that reuses the hardware required for the 32 K mode reducing the number of multiplexors.

The remainder of this paper is organized as follows. Section 2 presents the selection process of the most efficient FFT algorithm. A matrix-based and implementation level analysis are combined in order to find out the most appropriate algorithm for DVB-T2. In Sections 3 and 4, some implementation techniques are presented to reduce core area. The FFT processor implementation is presented in Section 5. Section 6 compares our proposal with previous FFT designs in the literature. Finally, some conclusions are given in Section 7.

2. Selection of the 2^k Algorithm

This section presents the procedure followed to select an area-efficient algorithm that meets the DVB-T2 requirements. First, a high level analysis is carried out using the matrix-based representation and methodology proposed in [15]. Secondly, a more detailed analysis is performed. In this additional proposed analysis, FFT algorithms are considered at implementation level and their performance in a complete DVB-T2 receiver is assessed.

2.1. High level analysis

In this section, the family of pipeline-SDF radix 2^k FFT algorithms is analyzed for the highest mode of DVB-T2, 32 K, which is the one with the highest hardware complexity. In [15], the discrete Fourier transform (DFT) matrix factorization method based on the Kronecker product is proposed to express the family of pipeline-SDF radix r^k decimation-in-frequency (DIF) FFT architectures.

When the number of points of the FFT, N, is a power of r^k ($N=r^{kn}$), $l=0$ and the architecture is composed by n radix r^k stages. Nevertheless, when N is not a power of r^k ($N=r^{kn+l}$), $0<l<n$ and there are n radix r^k stages and a special radix r^l stage.

Each stage is represented in terms of four types of operators: **B**, **M1**, **M2** and **M3. B** operators represent the butterflies. The radix 2 butterflies are implemented using two complex adders, that is, four real adders. Operators **M1**, **M2** and **M3** define the multiplications by the twiddle factors. These twiddle factor multiplier operators are implemented using one complex multiplier. We consider that a complex multiplier is composed by four real multipliers, two real adders, and a look-up table (LUT) from where the twiddle factors are read sequentially. When the number of different twiddle factors corresponding to a twiddle factor operator is 8 or 16, their implementation can be improved as described in [18] and in [19] respectively. When the length of the twiddle factor operator is 8, it can be implemented using two real constant multipliers and two real adders. When the length of the twiddle factor operator is 16, it can be implemented using four real constant multipliers and two real adders. In the case of the **M2** operator, there is only two different twiddle factors, 1 and $-j$. Thus, the complex multiplier can be replaced by a simple swap of the real and imaginary parts of its input data. A more detailed implementation of these devices can be found in [15].

Using these matrix-based expressions, the implementation of the radix 2^k algorithms for $N=32$ K with $k=1,2,\ldots,15$ has been analyzed. Table 1 summarizes the hardware complexity of all the radix 2^k algorithms for $N=32$ K. This table shows the number of typical stages, the value of l, the number of real multipliers, real constant multipliers, twiddle factors stored in LUTs and real adders for each operator **B**, **M1** and **M3**.

As can be seen in Table 1, the r2, $r2^2$, $r2^{12}$, $r2^{13}$, $r2^{14}$ and $r2^{15}$ algorithms are directly discarded due to their large number of twiddle factors in LUTs. The minimum number of real multipliers is achieved by the $r2^3$, $r2^4$ and $r2^6$ algorithms. The minimum number of twiddle factors in LUTs is obtained by the $r2^8$ algorithm. The $r2^6$ algorithm saves 1536 LUT twiddle factors and four real constant multipliers with respect to the $r2^4$ algorithm, at the expense of four additional real multipliers. Similarly, the $r2^8$ algorithm saves 160 LUT twiddle factors and four real constant multipliers with respect to the $r2^6$ algorithm, at the expense of four additional real multipliers. If the saving in ROM twiddle

Table 1
Hardware complexity comparison.

Algorithm	n	l	Real multipliers			Real constant multipliers			Twiddle factors in LUTs			Real adders			
			M1	M3	Total	M1	M3	Total	M1	M3	Total	M1	M3	B	Total
$r2^1$	15	0	44	–	44	6	–	6	65,504	–	65,504	26	–	60	86
$r2^2$	7	1	24	–	24	2	–	2	43,680	–	43,680	14	–	60	74
$r2^3$	5	0	16	–	16	–	10	10	37,440	–	37,440	8	10	60	78
$r2^4$	3	1	12	–	12	–	14	14	34,944	–	34,944	6	8	60	74
$r2^5$	3	0	8	12	20	–	6	6	33,792	96	33,888	12	4	60	76
$r2^6$	2	3	8	8	16	–	10	10	33,280	128	33,408	10	4	60	74
$r2^7$	2	1	8	16	24	–	4	4	33,024	320	33,344	12	4	60	76
$r2^8$	1	7	4	16	20	–	6	6	32,768	480	33,248	12	2	60	74
$r2^9$	1	6	4	16	20	–	6	6	32,768	704	33,472	12	2	60	74
$r2^{10}$	1	5	4	16	20	–	6	6	32,768	1376	34,144	12	2	60	74
$r2^{11}$	1	4	4	16	20	–	6	6	32,768	2720	35,488	12	2	60	74
$r2^{12}$	1	3	4	16	20	–	6	6	32,768	5429	38,197	12	2	60	74
$r2^{13}$	1	2	4	20	24	–	2	2	32,768	10,912	43,680	12	2	60	74
$r2^{14}$	1	1	4	20	24	–	4	4	32,768	21,824	54,592	12	2	60	74
$r2^{15}$	1	0	–	24	24	–	2	2	–	43,680	43,680	–	14	60	74

factors compensates the additional arithmetic complexity, the $r2^6$ or $r2^8$ algorithms will be the most appropriate pipeline architectures. Furthermore, if the saving in real multipliers compensates the extra memory in LUT, the $r2^4$ algorithm will be the most area-efficient pipeline architecture. As can be seen, it is difficult to have a decision for the optimum algorithm. Therefore, a more detailed second analysis is needed.

2.2. Implementation level analysis

The objective of the analysis at this level is the selection of an area-efficient 32 K FFT algorithm that fulfils the DVB-T2 specifications among the survivors of the former high level analysis.

For the hardware implementation, the following design rules are stated.

2.2.1. Rule 1

Due to the large number of points of the studied FFT, the memory used for the FIFOs of the butterfly modules has a huge length. Thus, the implementation of these FIFOs is directly related to the area of the FFT cores. From FIFO of size 64×28 bits to FIFO of size 1×32 bits, the memories are implemented using registers. However, one single-port memory (SPRAM) of half length and double word width is the most area-efficient option for larger memories [20].

2.2.2. Rule 2

When the number of different twiddle factors corresponding to a twiddle factor operator is 8 or 16, its implementation is improved as described in [18] and in [19] respectively. In other cases, a complex multiplier and a LUT are used. All the LUTs are implemented using combinational logic. The coefficient memory reduction scheme proposed in [21] has been applied, so that ROM tables of only $(N/8+1)$ coefficients are implemented. Therefore, the size of the largest LUT in our design is reduced from 32 K to 4 K+1.

2.2.3. Rule 3

In this work, the memory to reorder the FFT outputs has been not considered in the area results because the outputs can be reordered by the following modules in the DVB-T2 receiver. This is a common assumption in DVB-T receivers [9,6,8,10,3].

The surviving pipeline-SDF $r2^k$ algorithms have been implemented using the above rules. Note that, novel algorithms such as pipeline-SDF $r2^6$, $r2^7$, $r2^8$, $r2^9$, $r2^{10}$ and $r2^{11}$ have been implemented for the first time in this work to carry out this analysis.

In order to select the most area-efficient algorithm with the optimum values of the data bitwidth (*dbw*), and the twiddle bitwidth (*tbw*), a bit accurate model of the algorithm has been integrated in a complete DVB-T2 receiver model developed by the system designers, as shown in Fig. 1. *dbw* and *tbw* are constant throughout the whole FFT. The channel is modeled as a Rayleigh channel (DVBT-P). To calculate the carrier-to-noise ratio (CNR), 25 frames are transmitted. These frames are modulated in 256-quadrature amplitude modulation (256-QAM) with 1/8 guard interval, which accounts for 60 OFDM symbols for a 32 K-points FFT and 8 MHz bandwidth [1]. A code rate of 2/3 and a PP2 pilot pattern are selected. This configuration is mentioned in the NorDig specification [22], which is one of the most strict specifications for the design of DVB-T2 receivers.

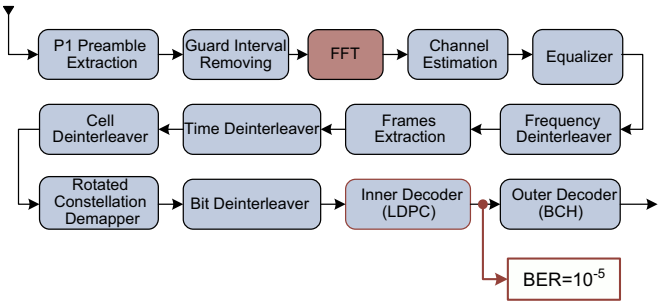

Fig. 1. Scheme of the considered DVB-T2 receiver model.

Several requirements are imposed to the FFT core by the DVB-T2 system designers. A 32 K FFT with 10-bit data input and 12-bit data output is required. The period of the OFDM symbol for this system configuration, when the FFT size is 32 K, is 4032 μs [1]. This timing specification is not the most critical timing for 32 K mode. However, it is the most critical configuration to evaluate the degradation in the CNR. The NorDig specification [22] indicates that for this configuration (1/8 guard interval, PP2 and FFT size of 32 K) a maximum of 2 dB can be lost in the CNR for a Bit Error Rate (BER) of 10^{-5} after Low-Density Parity Check (LDPC) decoding, owing to real channel estimation, imperfect LDPC decoding and the fixed-point implementation of all the receiver modules. In this work, it has been considered that a maximum of 0.1 dB CNR loss is reasonable for the fixed-point computation of the FFT module.

Fig. 2 represents the effect of each (*dbw*, *tbw*) configuration in the area and CNR for each FFT algorithm. The cell area results presented have been obtained by synthesizing the FFT cores for the XFAB 0.18 μm technology, with a clock frequency of 10 MHz and a typical process working at 1 V and 25 °C. The objective of this analysis is to evaluate the degradation that the quantization of the FFT introduces in the CNR for a fixed BER of 10^{-5} after LDPC decoding. First, a double floating-point FFT has been considered in the double floating-point model of the receiver developed in Matlab, and the CNR needed for a BER of 10^{-5} has been obtained. Then, the same analysis has been done with the designed fixed-point model of the FFT algorithms. The *x*-axis indicates the degradation in the CNR relative to the floating point FFT of the fixed point FFT, and the *y*-axis the synthesis area of the 32 K-points FFT cores in mm^2.

The *dbw*=15 bits configuration has to be discarded because the degradation in CNR is above 0.1 dB. The focus has been on *dbw*=16 bits, where a trade-off between CNR and synthesis area can be achieved. It can be observed that, for the same value of (*dbw*, *tbw*), the $r2^6$, $r2^8$ and $r2^9$ algorithms need the smallest synthesis area. Moreover, the (*dbw*, *tbw*)=(16, 8) is the most

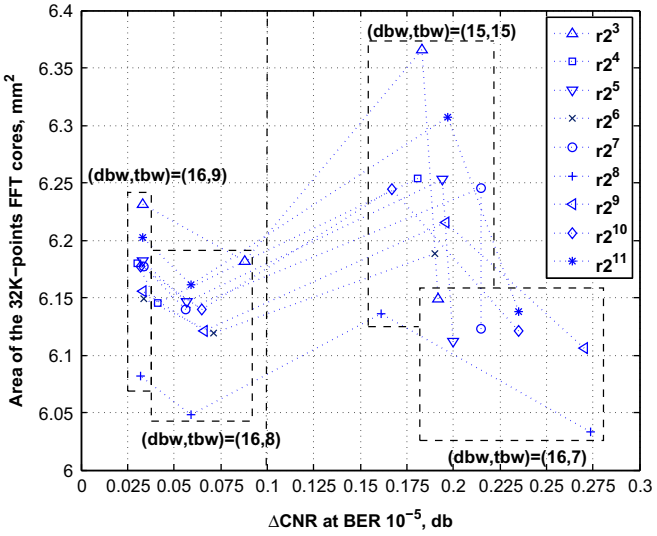

Fig. 2. Δ CNR for BER=10^{-5} and area of pipeline-SDF radix 2^k DIF 32 K-points FFT core.

area-efficient configuration that fulfils the CNR requirement for all the algorithms. Within this configuration, the $r2^4$ algorithm has the better system performance and the $r2^8$ algorithm requires the smallest silicon area. Thus, the pipeline-SDF radix 2^8 DIF architecture with a $(dbw, tbw)=(16, 8)$ and a synthesis area of 6.05 mm^2 is the most area-efficient core for a 32 K-points FFT.

3. Memoryless CORDIC optimization

The CORDIC algorithm [23] decomposes the desired rotation angle, θ, into a sum of a set of M predefined angles, α_i, with $i=0,\ldots,M-1$:

$$\theta=\sum_{i=0}^{M-1}\alpha_i+\varepsilon, \tag{1}$$

where ε is the error of approximation, and,

$$\alpha_i=\pm tg^{-1}(2^{-i}). \tag{2}$$

Thus, the rotation is accomplished iteratively according to

$$\begin{aligned} x_{i+1}&=x_i-y_i\delta_i 2^{-i}\\ y_{i+1}&=y_i+x_i\delta_i 2^{-i}, \end{aligned} \tag{3}$$

where δ_i indicates the direction of the so called micro-rotation.

In a conventional CORDIC, $\delta_i\in\{-1,1\}$. It means that all the rotations are accomplished, what leads to a constant gain of the rotator K.

$$K=\prod_{i=0}^{M-1}\cos(\alpha_i)=\prod_{i=0}^{M-1}\cos(tg^{-1}(2^{-i}))\approx 0.6073. \tag{4}$$

Thus, in a conventional CORDIC-based FFT processor a memory bank to store the necessary twiddle factor angles for the rotation is needed. In [17], a new memoryless CORDIC algorithm for the FFT computation is proposed. This is achieved calculating the direction of the micro-rotations from a control counter of the FFT. Nevertheless, the memoryless CORDIC algorithm proposed in [17] is focused on the radix r algorithms. In this paper, this algorithm has been extended to the radix r^k algorithms.

Fig. 3 shows the architecture of the designed CORDIC. First, the angle generator calculates the rotation angle θ. Then, the rotation generator produces the rotation sequence. As in a conventional CORDIC, $\delta_i\in\{-1,1\}$. The micro-rotations are calculated by a pipeline where each stage rotates the input data a certain angle. The number of stages depends on the desired precision. It must be noted that the rotation of 45°, that corresponds to the α_0 angle, is not considered. In this design, this first micro-rotation is achieved by the 180° and 90° rotations, which reduce the intrinsic gain of the CORDIC. Finally, the output data is scaled in order to compensate the gain of the CORDIC rotator.

In the following, the performance of each block is explained in detail.

- Angle generator.

The number of twiddle factors for a $\mathbf{M1}^{(a,b)}$ operator in an N-point FFT is N/r^a, where a depends on the k of the r^k algorithm and the processing stage, and b depends only on the k. The angle sequence is generated by concatenating r^b subsequences:

$$0,p,2p,\ldots,\left(\frac{N}{r^{a+b}}-1\right)p, \tag{5}$$

where p goes from 0 to r^b-1 in a bit-r reversed order. Fig. 4 shows the architecture of the angle generator block. It can be seen how these sequences can be obtained from 0 to N/r^a-1 counter. p is the value of the $\log_2(r^b)$ most significant bits of the counter, and the rest of the bits counts from 0 to $N/r^{a+b}-1$. Then, the angle is obtained by multiplying the value of the counter of p in a bit-r reversed order, and the value of the rest of the bits.

The generated sequence represents the rotation angle if the circumference is divided into N/r^a identical angles, i.e., if $\theta\in[0,\ldots,N/r^a-1]$ is one of the angles of the sequence,

$$\theta(rad)=-\frac{2\pi}{N/r^a}\theta. \tag{6}$$

- Rotation generator

Given the rotation angle, θ, the rotation generator calculates the vector δ that indicates the direction of the micro-rotations. Fig. 5 shows a scheme of the rotation generator block.

First, the three more significant bits of the angle θ are checked to determine if the 180° and 90° micro-rotations must be fulfilled. These two first micro-rotations correspond to the first rotation angle α_0. Their computation is defined by the two first bits of the rotation vector δ, that is, δ_0 and δ_1. The angle range goes from 0 to N/r^a-1, so these three bits of θ divide the circumference in eight sectors.

Fig. 6 shows the relationship between the angle θ and the defined sectors.

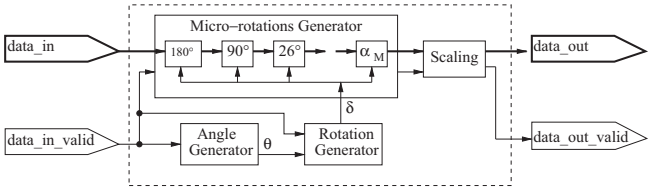

Fig. 3. Architecture of the designed CORDIC.

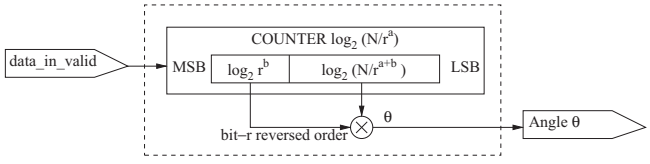

Fig. 4. Architecture of the angle generator module for any $\mathbf{M1}^{(a,b)}$ operator.

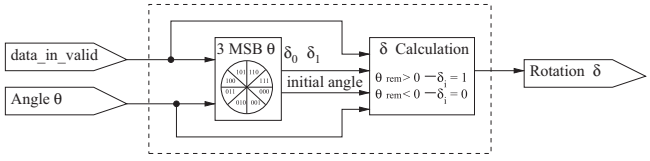

Fig. 5. Architecture of the rotation generator module.

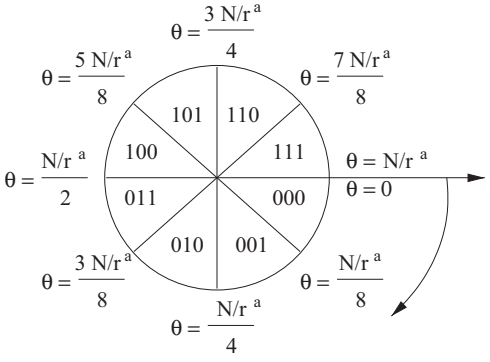

Fig. 6. Relationship between the three more significant bits of the angle θ and the eight sectors of the circumference.

In order to determine the two first bits of the vector δ, five cases must be differentiated depending on the three more significant bits of the angle θ. In the first case, the three more significant bits of the angle θ are "000". As can be seen in Fig. 6, this means that the rotation angle is between 0° and 45°, and that the initial angle is 0. Neither the 180° nor the 90° micro-rotations should be processed, and thus, the 2 first bits of the rotation vector δ, δ_0 and δ_1, are set to 0. The second case corresponds to the "001" and "010" states. The rotation angle is between 45° and 135° as shown in Fig. 6. The 90° micro-rotation should be processed, and thus, the second bit of the vector δ, δ_1, is set to 1, whereas the first one, δ_0, is 0. In this case, the initial angle is $N/r^a/4$. In the third case, these three bits of θ are "011" or "100". The rotation angle is between 135° and 225° as seen in Fig. 6. This supposes that the 180° micro-rotation should be processed, and thus, the first bit of the vector δ, δ_0 is set to 1, whereas the second one is 0. In this case, the initial angle is equal to $(N/r^a)/2$. The fourth case corresponds to the "101" or "110" states. The rotation angle is between 225° and 315° as shown in Fig. 6. Both the 180° and 90° micro-rotations should be processed, and thus, the two first bits of the vector δ, δ_0 and δ_1, are set to 1. In this case, the initial angle is set to $(3N/r^a)/4$. Finally, the fifth case is the "111" state. The rotation angle is between 315° and 360°. No micro-rotation should be processed, and thus, the two first bits of the vector δ, δ_0 and δ_1, are set to 0 as in the first case. However, in this case the initial angle must be N/r^a-1 in order to achieve that next δ values indicate the correct direction of the micro-rotations.

Table 2 summarizes the characteristics of each case.

Once we have the two first bits of the vector δ, the rest of the bits are calculated as in a conventional CORDIC. It is checked if the remainder angle is positive or negative to decide the next bit of the vector δ. When negative, $\delta_i=0$. When positive, $\delta_i=1$. Then adds or subtracts $\alpha_i(2)$ from this angle. This is repeated as many times as the selected number of stages for the desired precision.

For high values of i, it can be considered that $\alpha_i/\alpha_{i+1}\approx 2$ as is demonstrated in [17]. Thus, from this index i, the rest of the rotations can be calculated considering this approximation. If the remainder angle is normalized by the minimum rotation angle α_M, that corresponds to the last micro-rotation, the bits of the result offer the rest of the bits of δ.

There must be a trade-off between the precision of the calculation and the hardware resources. Thus, the number of micro-rotation stages, the number of bits used in the calculations of the rotation vector, and the selection of the first angle are parameters of the circuit.

- Micro-rotation generator

Given the rotation vector, δ, the direction of each micro-rotation is defined. The micro-rotation generator carries out the rotation of the input data with the addition and subtraction of each micro-rotation. Fig. 7 shows a scheme of the micro-rotation generator block.

Table 2
Values of the two first bits of the vector δ and initial angle depending on the three more significant bits of the angle θ.

θ	δ_0	δ_1	Initial angle
"000"	0	0	0
"001" or "010"	0	1	$\frac{N/r^a}{4}$
"011" or "100"	1	0	$\frac{N/r^a}{2}$
"101" or "110"	1	1	$\frac{3N/r^a}{4}$
"000"	0	0	N/r^a-1

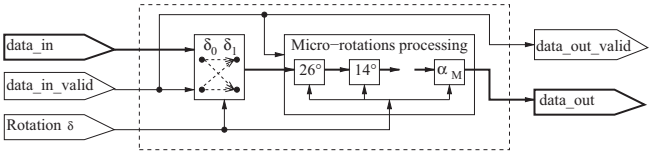

Fig. 7. Architecture of the micro-rotation generator module.

Table 3
Real and imaginary parts of the input data depending on δ_0 and δ_1.

δ_0	δ_1	R, I
0	**0**	R, I
0	**1**	I, −R
1	**0**	−R, −I
1	**1**	−I, R

The two first bits of the vector δ, δ_0 and δ_1 indicate the processing of the two first micro-rotations of 180° and 90°. These rotations have been easily calculated by interchanging the vector components and/or changing their sign. Table 3 shows the switching of the real and imaginary parts of the input data depending on δ_0 and δ_1.

The 180° and 90° micro-rotations correspond to the first rotation angle $\alpha_0=45°$. Thus, the third micro-rotation corresponds to the second rotation angle $\alpha_1\approx 26°$, and so on. Fig. 8 shows how the CORDIC micro-rotations are accomplished according to Eq. (3). This hardware is repeated as the selected number of micro-rotation stages, M in Fig. 7.

This micro-rotations processing is based on the idea that each micro-rotation is accomplished by an addition and a subtraction. According to this, the switch decides which data component must be added and which subtracted.

- Scaling

As has been said, in a conventional CORDIC all the rotations are accomplished, what leads to a constant gain of the rotator $K\approx 0.6073$, given by Eq. (8). Other CORDIC designs allow skipping some micro-rotations, and thus, the scale factor is not a constant.

In this design, the 45° rotation has been removed, and thus,

$$K=\prod_{i=1}^{M}\cos\left(tg^{-1}(2^{-i})\right)\approx 0.8588. \tag{7}$$

This can be achieved just by using two adders, considering that

$$K=0.8588\approx 0.8594=1-2^{-3}-2^{-6}. \tag{8}$$

4. Variable datapath optimization

In the analyzed fixed *dbw* pipeline-SDF radix 2^8 DIF FFT core proposed in the previous sections, almost 90% of the total area is occupied by the FIFOs. Therefore, minimizing *dbw* is important. Using the same *dbw* throughout whole FFT gives poor performance since the data has to be shifted down after each butterfly to maintain the same wordlength although overflow is not produced.

The shuffling of the butterflies is achieved using a delay feedback of N/r^i, being N the number of points of the FFT, r the radix and i the number of butterfly stage. The fact that the first stages in a DIF FFT contain the largest FIFOs has been taken into account to find a better solution. Thus, the main goal should be to minimize *dbw* in the first stages. If the bitwidth is reduced at the input and then allowed to increase towards the output, the

required memory, as well as the design area, will decrease significantly.

Fig. 9 shows the internal wordlength in each butterfly using a fixed datapath and a variable datapath for the selected design. The *dbw* of each butterfly has been reduced to the minimum value that complies with the DVB-T2 CNR requirement as in [13,24]. The FFT outputs are truncated from 16 bits to 12 bits which is the output bitwidth specification imposed by the system designers.

Additionally, this method has been enhanced evaluating in which butterfly stages are better not to divide by 2. Normally, to process an *N*-points FFT core, data are divided by two after the butterfly operations in each stage to avoid overflow and at the end of the FFT processing the outputs are multiplied by *N*. When data are not divided by two, the overflow is controlled. Analyzing the scaling of the data in each butterfly stage, we conclude that the degradation in the CNR is reduced when not dividing in the first, the second and the fourth butterflies.

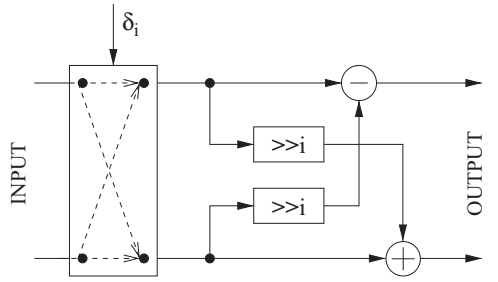

Fig. 8. CORDIC micro-rotations processing [17].

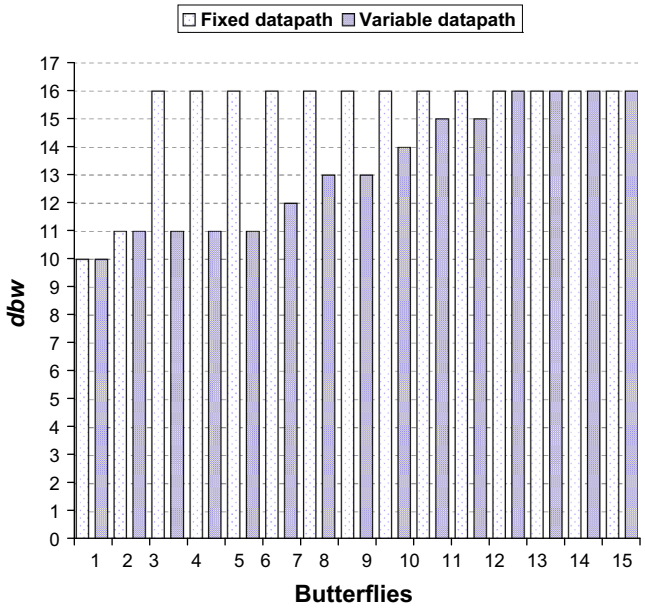

Fig. 9. Internal butterfly wordlengths of a 32 K-points pipeline-SDF radix 2^8 DIF FFT core.

Table 4
Stages of the pipeline-SDF radix 2^8 FFT core in variable-length operation.

Mode	1 K	2 K	4 K	8 K	16 K	32 K
Typical stages	1	1	1	1	1	1
l	2	3	4	5	6	7

5. Variable–length architecture

DVB-T2 systems can work with different operation modes where the number of sub-carriers change on-line. In DVB-T2 receivers, the FFT core must be able to work in 1 K/2 K/4 K/8 K/16 K/32 K modes. Table 4 presents the number of typical stages and the value of *l* of the pipeline-SDF radix 2^8 FFT core for each DVB-T2 operation mode. All the modes are composed by one radix 2^8 stage and one special radix 2^l stage. Thus, the processing elements of this special stage depend on the operation mode.

The proposed variable-length architecture has the special radix 2^l stage at the end of the architecture as can be seen in Fig. 10. Thus, a final multiplexing is required to determine the final processing elements for each working mode. The memoryless CORDIC and the variable datapath optimizations explained in Sections 3 and 4 respectively have been applied to the proposed variable-length FFT cores checking that the core complies with the system requirements for the six operation modes of DVB-T2 receivers.

The largest LUT in the proposed radix 2^8 algorithm has been replaced by this CORDIC. Thus, the first ROM table that stored 4097 twiddle factors and its corresponding complex multiplier disappear.

Table 5 shows the synthesis area of the proposed pipeline-SDF radix 2^8 DIF 32 K-points FFT core with and without the memoryless CORDIC optimization. An area saving of 7% is obtained in the core area applying this memoryless CORDIC to the largest LUT of the design.

Table 6 shows the synthesis area of the proposed pipeline-SDF radix 2^8 DIF 32 K-points FFT core for fixed and variable datapath configurations. An area saving of 10% is obtained in the FIFOs area applying this variable datapath configuration compared with the fixed *dbw* FFT core. It can also be observed that the core area is reduced with the variable datapath configuration. Applying the optimized variable datapath technique, a synthesis area of 5.39 mm^2 is obtained. This implies an area saving of almost 11% compared with the fixed *dbw* FFT core.

In the following, we present how this variable-length architecture is addressed.

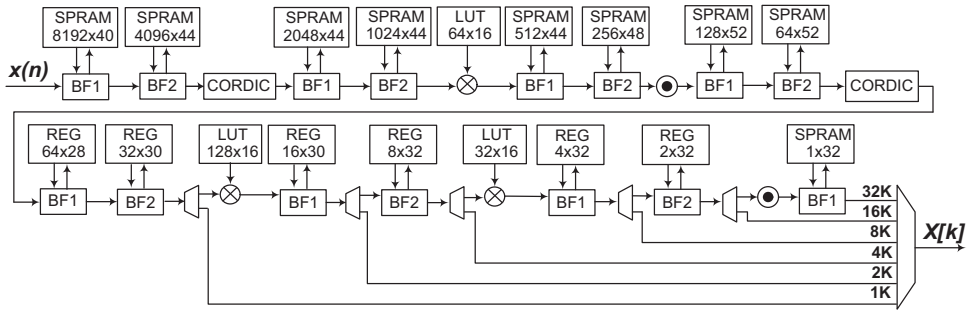

Fig. 10. Architecture of the proposed pipeline-SDF radix 2^8 DIF variable-length FFT with final multiplexing.

Table 5
Synthesis area of pipeline-SDF radix 2^8 DIF 32 K-points FFT core with and without the memoryless CORDIC optimization.

dbw (bits)	**tbw** (bits)	**Memoryless CORDIC**	**FIFOs** (mm^2)	**Core** (mm^2)	**Total** (mm^2)
16	8	No	5.33	0.71	6.04
16	8	Yes	5.33	0.66	5.99

Table 6
Synthesis area of pipeline-SDF radix 2^8 DIF 32 K-points FFT core for fixed and variable datapath configurations.

dbw (bits)	**tbw** (bits)	**FIFOs** (mm^2)	**Core** (mm^2)	**Total** (mm^2)
Fixed (16)	8	5.33	0.66	5.99
Variable	8	4.80	0.59	5.39

Fig. 10 shows the proposed variable-length architecture with final multiplexing. This proposal bypasses the last processing elements of the architecture.

BF1 devices are mapped to a device that implements the hardware needed to perform the arithmetic operations of the butterflies (**B**) and the required shuffling. **BF2** devices are similar to that of **BF1** devices. However, some of the outputs of the butterflies are multiplied by $-j$ (**M2**). These multiplications are implemented by swapping the real and imaginary parts and changing the sign of the real part of the input data. The $\otimes$ symbol represents a general complex multiplier and the $\odot$ symbol a constant complex multiplier.

The size of the required FIFOs depends on the FFT length N. For an (N_{max}/m)-points FFT, the length of these FIFOs is $1/m$ of the needed ones for an (N_{max})-points FFT where $N_{max}=32$ K and m can be 2, 4, 8, 16 or 32. Thus, depending on the working mode, only a part of the FIFO memories is needed for the shuffling. Therefore, each butterfly uses memory blocks larger than they need in lower modes.

Ref. [9] presents a 2 K/4 K/8 K-points FFT. In [12] a 1 K/2 K/4 K/8 K/16 K/32 K-points FFT core is implemented. Both propose to use multiplexors to select the FIFOs used by the butterflies in each mode. However, we propose to change the maximum value of the counter that address the FIFOs employing always the same memories for the different modes. This is a local solution which facilitates the routing phase. Therefore, our proposal will be more area-efficient.

Regarding the twiddle factors, it is known that the twiddle factors of the maximum number of points, in this case 32 K-points, contain the values of the twiddle factors of the lower length FFTs. In order to read the appropriate value stored in the LUT, two nested counters are used. The first counter controls easily the reading repetition and the second counter generates the address of the LUT to be read. The maximum value of each counter depends on the operation mode.

6. Comparison with other FFTs in the literature

The layout of the proposed variable-length FFT core with the final multiplexing, the variable datapath and the CORDIC optimizations has been carried out for XFAB 0.18 μm technology. The first eight FIFOs have been implemented using SPRAMs of half length and double word width and the last seven FIFOs, using registers. All the operation modes comply with DVB-T2 timing requirements. For instance, the execution time to compute the 32 K-points FFT at 10 MHz is 3276.8 μs, which is enough to meet

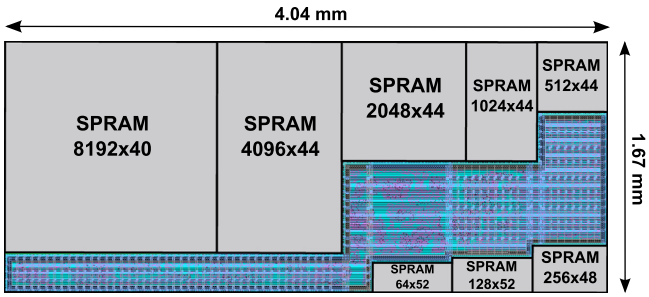

Fig. 11. Layout of the proposed variable-length pipelined FFT processor.

the most critical timing specification (3612 μs). The core size is 6.75 mm^2 as is shown in Fig. 11.

To our best knowledge, only three DVB-T2 FFT cores have been presented in the literature [12–14]. Ref. [12] presents only FPGA results. Refs. [13,14] provide standard cell results. However, only [12] proposes a variable-length core for DVB-T2.

Table 7 presents the performance features of previous DVB-T and DVB-T2 works and the proposed core to process a 32 K-points and a 8 K-points FFT. To compare fairly with the previous works, core area and power consumption are normalized to 0.18 μm as proposed in [4,25].

$$A_{norm}=\frac{\text{Area}}{(\text{Technology}/0.18\ \mu\text{m})^2} \tag{9}$$

$$\frac{\text{FFTs}}{\text{Energy}}=\frac{\text{Technology}/0.18\ \mu\text{m}}{\text{Power}\times\text{ExecutionTime}\times 10^3} \tag{10}$$

where (10) means the number of N-points FFTs that can be computed within 1 J of energy.

Our core can process all the modes of the DVB-T2 applications while the other two DVB-T2 proposals can only process a 32 K-points FFT. In spite of this fact, the proposed core is the most area-efficient DVB-T2 implementation which is the main goal of this proposal.

Due to the few DVB-T2 proposals, our core has been also compared with DVB-T receivers whose FFT maximum size is 8 K-points. As can be seen in Table 7, our core can be considered as an area-efficient design since the increase of area with respect to DVB-T receivers is due to the necessary extra hardware resources for a 32 K-points FFT.

Although the main goal of this proposal is the area efficiency, the power consumption of the 8 K-points mode is also presented in Table 7 in order to complete the comparison. The 8 K-points mode is selected since this is the maximum size of the FFT in a DVB-T system and this mode is supported by both DVB-T and DVB-T2 receivers. This power consumption result indicates that our proposal achieves a good power efficiency for the 8 K-points working mode. In order to get a fair comparison, the number of FFTs per Energy is also calculated since our core works at a low clock frequency. In spite of this fact, our core processes more 8 K-points FFTs per Energy than the other DVB-T proposals.

7. Conclusion

The family of pipeline-SDF radix 2^k DIF architectures have been proposed for the design of the DVB-T2 variable-length FFT core. Analyzing the performance of each algorithm in a complete DVB-T2 system, it has been concluded that the pipeline-SDF radix 2^8 DIF architecture is an area-efficient design.

On the one hand, a memoryless CORDIC has been proposed in order to remove the largest LUT of the design. An area saving of 7% has been obtained in the core area with this optimization. On the

Table 7
Performance comparison with 32 K-points DVB-T2 FFT designs.

	Lin [4]	Lee [7]	Wang [8]	Cortés [9]	Lin [14]	Turrillas [13]	Proposed
FFT size (K)	8	1, 2, 4, 8	512, 1, 2, 4, 8	2, 4, 8	32	32	1, 2, 4, 8, 16, 32
Application	DVB-T	DVB-T	DAB, DVB-T/H	DVB-T/H	DVB-T2	DVB-T2	DVB-T2
Wordlength (bits)	2 × 11	2 × 12	2 × 12	2 × 16	2 × 12	2 × 10	2 × 10
Algorithm	Radix 8	Balanced binary tree	Mixed-radix 2/4/8	Radix 2^2	Radix 2	Radix 2^5	Radix 2^8
Architecture	Parallel	Pipeline-SDF	Pipeline-SDF	Pipeline-SDF	Parallel	Pipeline-SDF	Pipeline-SDF
Process (μm)	0.18	0.18	0.18	0.35	0.09	0.09	0.18
Voltage (V)	1.8	1.8	1.8	3.3	1.0	1.0	1.8
Clock rate (MHz)	20	20	20	9.143	25	80	10
Area (mm^2)	4.84	3.52	2.9	18.7	2.5	2.88	6.75
A_{norm} (mm^2)	4.84	3.52	2.9	4.95	10	11.52	6.75
Power, 8 K-points (mW)	25.2	40.7	279	114.65	N.A.	N.A.	19.94
Execution time, 8 K-points (μs)	717.35	410.2	N.A.	450.1	N.A.	N.A.	819.2
Normalized FFTs/energy	55.32	59.9	N.A.	37.68	N.A.	N.A.	61.22

other hand, a variable datapath optimization has been applied to achieve a large reduction in area. More than the 80% of the design area is occupied by the FIFOs, whose area is directly related with the data wordlength. Reducing the data wordlength in each butterfly to the minimum possible value, an area saving of 11% has been obtained without CNR degradation.

Additionally, the selected architecture has been adapted to accommodate the 1 K/2 K/4 K/8 K/16 K/32 K variable-length FFT processor needed in DVB-T2 receivers. An area-efficient variable-length design has been achieved multiplexing the input data to the architecture depending on the working mode.

The layout of the proposed variable-length FFT core has been designed in XFAB 0.18 μm technology. The core complies with the DVB-T2 timing specifications at the low clock frequency of 10 MHz. Furthermore, it takes an area of 6.75 mm^2. The proposed DVB-T2 processor improves the chip area with respect to the previous DVB-T2 cores presented in the literature. Furthermore, it provides a good power efficiency in the lower modes.

Acknowledgments

This work has been partially supported by the subprogram Avanza of the Ministery of Industry, Tourism and Commerce under the project Furia-2: Futura Red Integrada Audiovisual. TSI-020301-2009-33. This work has been possible thanks to the cooperation with the University of Seville for their support in the DVB-T2 model simulations. Thus, the fixed–point models of the FFT algorithms presented in this paper were integrated in the DVB-T2 receiver in order to select the appropriate data and twiddle factor bitwidths of the FFT core to fulfill the required BER.

References

[1] ETSI Standard EN 302 755 V1.1.1, Framing structure, channel coding and modulation for a second generation digital terrestrial broadcasting system (DVB-T2) (September 2009).
[2] C.-P. Hung, S.-G. Chen, K.-L. Chen, Design of an efficient variable-length FFT processor, in: Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), vol. 2, Vancouver, BC, Canada, 2004, pp. II–833–6.
[3] X. Li, Z. Lai, J. Cui, A low power and small area FFT processor for OFDM demodulator, IEEE Trans. Consum. Electron. 53 (2) (2007) 274–277.
[4] Y.-W. Lin, H.-Y. Liu, C.-Y. Lee, A dynamic scaling FFT processor for DVB-T applications, IEEE J. Solid-State Circuits 39 (11) (2004) 2005–2013.
[5] C.-L. Wey, W.-C. Tang, S.-Y. Lin, Efficient memory-based FFT architectures for digital video broadcasting (DVB-T/H), in: Proceedings of the IEEE International Symposium on VLSI Design, Automation and Test (VLSI-DAT), Hsinchu, Taiwan, 2007, pp. 1–4.
[6] C.-C. Wang, J.-M. Huang, H.-C. Cheng;, A 2 K/8 K mode small-area FFT processor for OFDM demodulation of DVB-T receivers, in: Proceedings of the IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, 2005, pp. 165–166.
[7] H.-Y. Lee, I.-C. Park, Balanced binary-tree decomposition for area-efficient pipelined FFT processing, Circuits Syst. Regul. 54 (4) (2007) 889–900.
[8] S.-S. Wang, C.-S. Li, An area-efficient design of variable-length fast Fourier transform processor, J. Signal Process. Syst. 51 (3) (2008) 245–256.
[9] A. Cortés, I. Vélez, I. Zalbide, A. Irizar, J.F. Sevillano, An FFT core for DVB-T/DVB-H receivers, VLSI Des. 2008 (2) (2008) 1–9.
[10] H.-G. Kim, K.-T. Yoon, J.-S. Youn, J.-R. Choi, 8k-point pipelined FFT/IFFT with compact memory for DVB-T using block floating-point scaling technique, in: Proceedings of the IEEE International Symposium on Wireless Pervasive Computing (ISWPC), Melbourne, Australia, 2009, pp. 1–5.
[11] K. Jung, H. Lee, Low-cost variable-length FFT processor for DVB-T/H applications, in: IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), 2010, pp. 752–755.
[12] M. Turrillas, A. Cortés, I. Vélez, J.F. Sevillano, A. Irizar, An FFT core for DVB-T2 receivers, in: Proceedings of the IEEE International Conference on Electronics, Circuits and Systems (ICECS), Hammamet, Tunisia, 2009, pp. 120–123.
[13] M. Turrillas, A. Cortés, J.F. Sevillano, I. Vélez, C. Oria, A. Irizar, V. Baena, Comparison of area-efficient FFT algorithms for DVB-T2 receivers, Electron. Lett. 46 (15) (2010) 1088–1089.
[14] S.-Y. Lin, C.-L. Wey, M.-D. Shieh, Low-cost FFT processor for DVB-T2 applications, IEEE Trans. Consum. Electron. 56 (4) (2010) 2072–2079.
[15] A. Cortés, I. Vélez, J.F. Sevillano, Radix r^k FFTs: matricial representation and SDC/SDF pipeline implementation, IEEE Trans. Signal Process. 57 (7) (2009) 2824–2839.
[16] T. Lenart, V. Owall, A 2048 complex point FFT processor using a novel data scaling approach, in: Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), vol. 4. 2003, pp. IV–45–IV–48.
[17] M. Garrido, J. Grajal, Efficient memoryless cordic for FFT computation, in: IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP 2007, vol. 2, 2007, pp. II-113–II-116.
[18] J.-Y. Oh, M.-S. Lim, New radix-2 to the 4-th power pipeline FFT processor, IEEE J. Solid State Circuits E88-C (8) (2005) 1740–1746.
[19] J.-Y. Oh, M.-S. Lim, Area and power efficient pipeline FFT algorithm, in: Proceedings of the IEEE Workshop on Signal Processing Systems Design and Implementation, 2005, pp. 520–525.
[20] T. Lenart, V. Owall, Architectures for dynamic data scaling in 2/4/8 K pipeline FFT cores, Very Large Scale Integr. Syst. 14 (11) (2006) 1286–1290.
[21] D.-B. K. Huirae Cho, Myung-Soon Kim, J.-U. Kim, R2^2SDF FFT implementation with coefficient memory reduction scheme, in: IEEE Vehicular Technology Conference (VTC-2006), 2006, pp. 1–4.
[22] NorDig-T2, Requirements to NorDig-T2 compliant IRDs addendum to the NorDig Unified Requirements (ver 2.1) for Integrated Receiver Decoders for use in cable, satellite, terrestrial and IP-based networks (July 2009).
[23] J.E. Volder, The CORDIC trigonometric computing technique, IRE Trans. Electron. Comput. EC-8 (3) (1959) 330–334.
[24] C.-L. Hung, S.-S. Long, M.-T. Shiue, A low power and variable-length FFT processor design for flexible MIMO OFDM systems, in: Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), Taipei, Taiwan, 2009, pp. 705–708.
[25] B.M. Baas, A low-power, high performance, 1024-point FFT processor, IEEE J. Solid-State Circuits 34 (3) (1999) 380–387.