Hindawi Publishing Corporation
VLSI Design
Volume 2008, Article ID 610420, 9 pages
doi:10.1155/2008/610420

Research Article

An FFT Core for DVB-T/DVB-H Receivers

A. Cortés, I. Vélez, I. Zalbide, A. Irizar, and J. F. Sevillano

Department of Electronic and Communication, CEIT and Tecnun, Univerisity of Navarra, 20018 Donostia-San Sebastian, Spain

Correspondence should be addressed to A. Cortés, acortes@ceit.es

Received 27 April 2007; Revised 15 November 2007; Accepted 23 January 2008

Recommended by Jean-Baptiste Begueret

This paper presents the design and implementation of a 2K/4K/8K multiple mode FFT core for DVB-T/DVB-H receivers. The proposed core is based on a pipeline radix-2^2 SDF architecture. The necessary changes in the radix-2^2 SDF architecture to achieve an efficient FFT implementation are detailed. Quantization effects and timing design parameters are analyzed for DVB-T/DVB-H. Area and power results are provided for the proposed core.

1. INTRODUCTION

DVB-H adapts the successful DVB-T standard for digital terrestrial television to the specific requirements of mobile, handheld, and battery-powered receivers [1–3]. Both standards rely on an orthogonal frequency division multiplexing (OFDM) modulation scheme to achieve high-data rates in multipath environments. OFDM uses an inverse fast Fourier transform (IFFT) to modulate the signal and a fast Fourier transform (FFT) to demodulate it. One of the main differences between both standards is the number of points of the FFT: DVB-H that proposes an additional mode (4K) to the two DVB-T modes (2K and 8K). This new mode is a tradeoff between reception quality in movement and network size.

As DVB-H is an extension to DVB-T, it is possible to introduce DVB-H services in the bandwidth of DVB-T. One operator can offer two DVB-T services and one DVB-H service to its subscribers. Therefore, digital terrestrial television receivers should be able to receive both DVB-T and DVB-H signals. In these receivers, the FFT processor must be able to work in 2K/4K/8K multiple mode. Moreover, this module must have a high throughput in order to achieve the high-data rates required by both standards. Some 2K/4K/8K pipelined FFT architectures have been proposed in the literature [4].

The algorithm and architecture for the FFT core should be chosen trading off its processing speed, area, and power. Monoprocessor architectures such as [5] have to be discarded, as they are not able to fulfill the timing specifications. The throughput can be increased by using either parallel [6, 7] or pipeline architectures [8–14]. Pipeline architectures present smaller latency and lower-power consumption [8, 9], which makes them suitable for mobile devices such as DVB-T/DVB-H receivers.

Basically, two types of pipeline architectures can be distinguished: single-path delay feedback (SDF) architectures [10–12] and multipath delay commutator (MDC) architectures [9]. SDF architectures use registers more efficiently, since the outputs of the butterflies can be stored in shift registers. In MDC architectures, the input sequence is divided into several parallel lines that feed the butterfly applying the appropriate delays to the data. SDF architectures use less memory than MDC. On the other hand, MDC architectures obtain a slightly higher throughput than SDF architectures. The optimal choice depends on the application [9]. In the case of DVB-T/DVB-H receivers, the SDF architecture can achieve the required throughput and needs less area than the MDC architecture.

In addition to the pipeline structure, the radix of the algorithm also influences the complexity of the implementation. A radix-2 algorithm needs more products and gets a lower throughput than a radix-4 algorithm. However, a radix-4 algorithm can only process FFTs with a number of points that is a power of 4, and the butterfly is more complex.

In order to maintain the simplicity of the radix-2 butterfly, radix-2^2 (r2^2) algorithms have been proposed [11, 14]. The r2^2 algorithm is well-suited for DVB-T/DVB-H applications since it can work both with a number of points that is a power of 4 and with a number of points that is a power of 2.

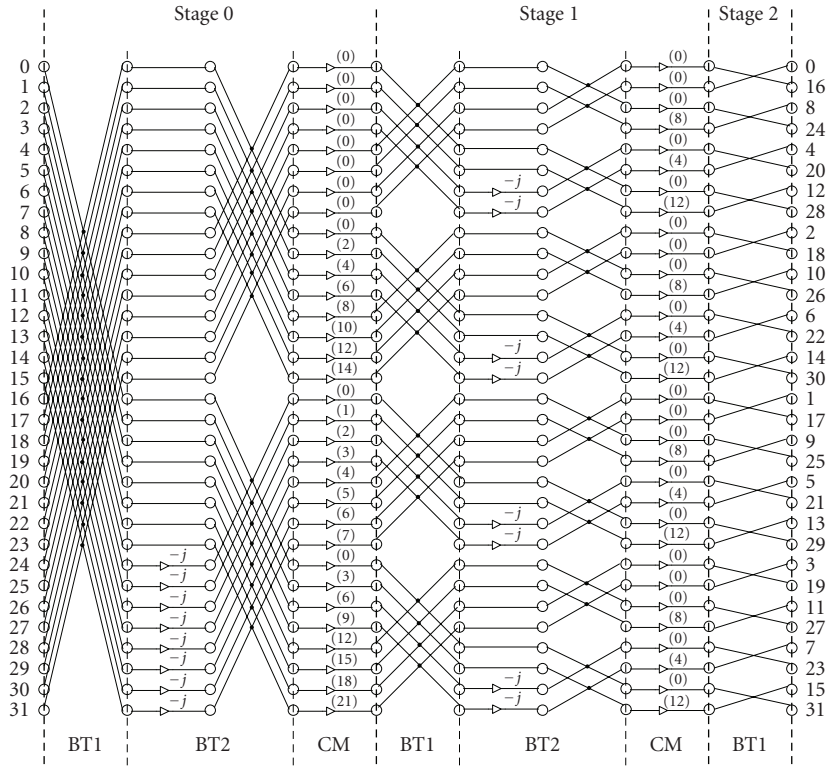


Figure 1: Flow graph of the r2^2 SDF 32-point FFT algorithm.

However, the implementation of the r2^2 FFT in [11] should be modified to achieve a multiple mode operation.

This paper presents a new 2K/4K/8K FFT core for DVB-T/DVB-H receivers. The r2^2-SDF pipeline architecture in [11] is adapted to achieve a multiple mode 2K/4K/8K FFT. The effect of the quantization errors of the FFT processor is studied, and design parameters are analyzed according to the DVB-T/DVB-H requirements.

This paper is organized as follows. Section 2 explains the modifications done to the r2^2 SDF algorithm so that it can operate in multiple mode. Section 3 describes the pipeline r2^2-SDF architecture proposed for a DVB-T/DVB-H receiver. Section 4 gives signal-to-noise-ratio (SNR), area, timing, and power results of the proposed FFT core. Finally, Section 5 summarizes the conclusions of this work.

2. r2^2 -SDF ALGORITHM

In this section, the r2^2-SDF algorithm presented in [11] will be modified by us to enable it to work in a multiple mode operation.

Figures 1 to 3 show the flow graph of the radix-2^2 algorithm when the length of the FFT is 32, 16, and 8, respectively. In these flow graphs, the index of the input and output data is shown. The black circles represent butterflies, and the arrows are the twiddle factor multiplications, where the number in brackets next to each arrow is the index m of the twiddle factor, $W_N^{(m)} = e^{j2\pi m/N}$, of an FFT of length N.

Three different operations can be distinguished in each stage: BT1, which is the butterfly of type 1 defined in [11]; BT2, which is the butterfly of type 2 defined in [11]; and CM, which performs complex multiplications. We can observe that, for BT2, some input data of the butterflies are multiplied by j. The multiplication by j can be implemented by exchanging the real and imaginary parts.

The twiddle factors of the CM at the kth stage, with $k = 0, 1, \ldots, \text{ceil}(\log_4 N) - 2$, are given by the vector $A_k = \{a_i\},\ i = 1, 2, \ldots, N/2^{2k}$, where $a_i = e^{-j2\pi p/N}$. The index p is defined as

$$p = \begin{cases} 0, & 0 \le i < m, \\ 2 \cdot 2^{2k} \cdot (i - m), & m \le i < 2m, \\ 2^{2k} \cdot (i - 2m), & 2m \le i < 3m, \\ 3 \cdot 2^{2k} \cdot (i - 3m), & 3m \le i < 4m, \end{cases} \tag{1}$$

where m is equal to $N/2^{2+2k}$.

The r2^2 algorithms reduce considerably the number of arithmetical operations performed by the FFT. The only restriction to process the r2^2-FFT algorithm is that the length of the FFT is a power of 2. Thus, the last stage of the algorithm is different according to the size of the FFT. If the number of points of the FFT is a power of 4, the last stage is composed of BT1 and BT2, as it can be seen in Figure 2. However, if the number of points of the FFT is only a power of 2, the last stage will be formed by BT1, as can be observed in Figures 1 and 3.

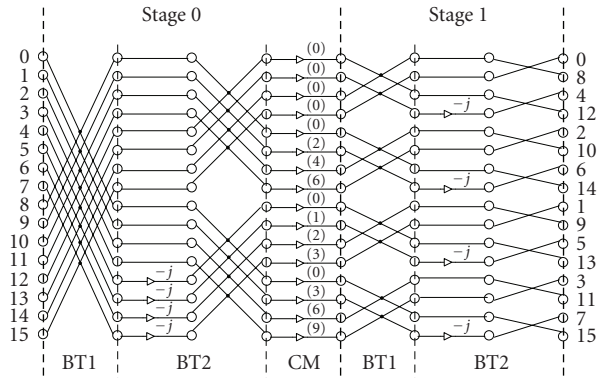


Figure 2: Flow graph of the r2^2 SDF 16-point FFT algorithm.

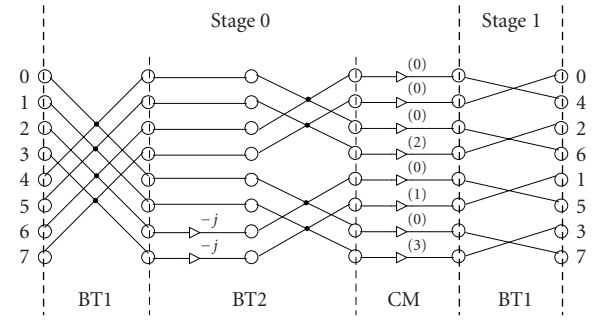


Figure 3: Flow graph of the r2^2 SDF 8-point FFT algorithm.

2.1. Multiple mode operation

In this section, the $2^{2(a-1)-1}/2^{2(a-1)}/2^{2a-1}$ multiple mode of the r2^2-SDF algorithm will be derived where a represents the number of stages of an FFT of $N_{\max} = 2^{2a-1}$ points. The resources needed to process the FFT of the largest number of points, $N_{\max}$, will be implemented.

An FFT of $2^{2(a-1)-1}$ points can be easily obtained from a 2^{2a-1} points FFT. The first stage of the $N_{\max}$ points FFT does not need to be processed in order to calculate the $2^{2(a-1)-1}$ points FFT. Additionally, in the following stages of the $N_{\max}$ points FFT, the twiddle factors are the same as the ones needed for the $2^{2(a-1)-1}$ points FFT, as $W_{N_{\max}}^{(4i)} = W_{N_{\max}/4}^{(i)}$. This multiple mode implementation can be deduced by analyzing the flow graphs of a 32-points FFT and an 8-points FFT in Figures 1 and 2, respectively, where we have considered $a = 3$.

Using the resources of an FFT of length $N_{\max}$, a $2^{2(a-1)}$ points FFT can be obtained. In the latter FFT, $a-1$ stages are needed. The first $a-1$ stages of the $N_{\max}$ points FFT can be reused to process the $2^{2(a-1)}$ points FFT, if only the operations in the even positions are carried out. Thus, half the operations are done in each BT1, BT2, and CM. Moreover, the twiddle factors of the CM in stage $a-1$ of the $2^{2(a-1)}$ points FFT are always one. Thus, the operations of the last CM can be omitted, and the final stage of the $2^{2(a-1)}$ FFT will only contain a BT1 and a BT2. This multiple mode implementation can be easily concluded by inspection of Figures 1 and 2, taking into account that $W_{N_{\max}}^{(2i)} = W_{N_{\max}/2}^{(i)}$.

3. PIPELINE r2^2 -SDF ARCHITECTURE IN MULTIPLE MODE

The proposed FFT core receives the input data DATA_IN in natural order, and it generates the output DATA_OUT in bit-reversed order. This is not a problem as the reordering can be performed by subsequent modules of the DVB-T/DVB-H receiver (e.g., deinterleaver) with no additional cost. Input data arrives at clock rate. All the data needed to compute each FFT arrives as a block. In order to ease the integration of the FFT core within a DVB-T/DVB-H receiver, a validation signal, DATA_IN_VALID, will be set high during the arrival of valid data at the input of the FFT core. Similarly, when valid output data are ready at the output of the FFT core, a validation signal DATA_OUT_VALID is set high.

The FFT processor employs fixed-point arithmetic. Input and output data are represented using *dbw* bits. The twiddle factors have been quantized with *tbw* bits. The core scales data appropriately during internal operations to avoid overflow.

In the following, the basic building blocks of the r2^2-SDF architecture [11] are described and their implementation detailed. Then, the required modifications to achieve a multiple mode 2K/4K/8K FFT are explained. The section finishes with a summary of the main features of the proposed multiple mode FFT core.

3.1. Basic building blocks

The FFT processor is a distributed system where every module generates the control signals for the next module in the pipe, as shown in Figure 4. Shadowed lines represent data, and white lines control signals. The FFT core has ceil ($\log_4 N$) stages, where N is the number of FFT points. A typical stage of the architecture consists of three processing elements: B1, B2, and CM; and three memory elements: ROM, FIFO1, and FIFO2. B1 and B2 carry out the processing of the two types of butterflies of the r2^2 algorithm (BT1 and BT2). FIFO1 and FIFO2 are used to achieve the required data shuffling for proper operation of the butterflies. In stage k, the depth of FIFO1 is $N/2^{k+1}$, and the depth of FIFO2 is $N/2^{k+2}$. CM computes the complex multiplications between the outputs of B2 and the twiddle factors stored in the corresponding ROM memory. The last stage of the FFT processor does not need the complex twiddle factor multiplication.

As explained before, the r2^2-SDF architecture can work with a number of points that is a power of 4 and with a number of points that is only a power of 2. When N is just a power of 2, in the last stage, data are only processed by B1.

3.1.1. Module B1

Figure 5 shows the structure of B1. The DATA_IN input port comes from the previous component in the pipe, normally a CM module. The DATA_OUT output port is connected to the next component in the pipe, usually a B2 module. The

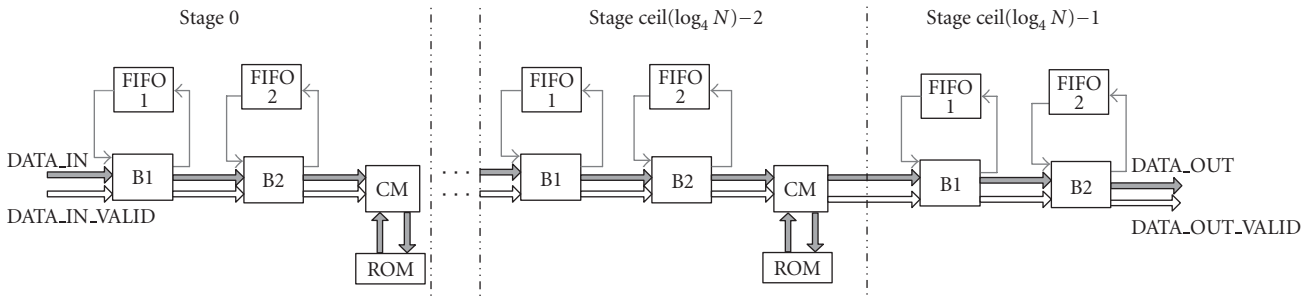


Figure 4: Typical pipeline r2^2-SDF architecture.

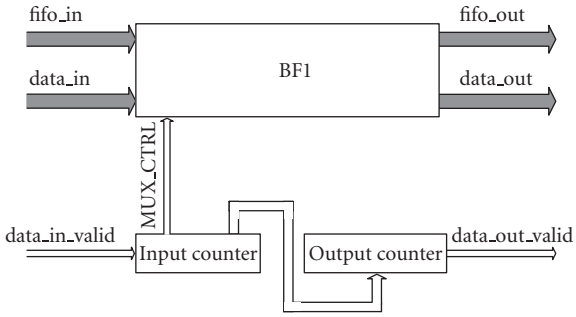


Figure 5: B1 module structure.

FIFO_IN and FIFO_OUT ports connect B1 with FIFO1. The size of the two counters of B1 is $2(\log_4 N - k)$, where k is the stage. The implementation of BF1, the butterfly of type 1, is detailed in Figure 6.

Initially, the multiplexers MUX are in position 0, and FIFO1 is empty. During the arrival of the first $N/2^{k+1}$ data, FIFO1 is filled. Then, the multiplexers change to position 1, and the butterfly operations can be performed using the input data at port DATA_IN and the data stored in FIFO1. One of the butterfly outputs, X1, is output, whereas the other one, X2, is stored in FIFO1. After other $N/2^{k+1}$ cycles, multiplexers switch back to position 0. Data for the next computation is stored in FIFO1, and the results X2 of the previous butterfly operations are sent out.

The selection signal of the multiplexers, MUX_CTRL, is generated by the input counter. This counter increments its value when DATA_IN_VALID is high. When the input counter arrives to half of its count, DATA_OUT_VALID is set high, and the output counter is started. The output counter will count until FIFO1 is emptied of valid output data. When the output counter finishes its count, DATA_OUT_VALID is set to zero.

3.1.2. Module B2

Figure 7 shows an internal diagram of component B2, which is formed by a butterfly of type 2, BF2, and some control logic. The DATA_IN input port of B2 comes from the previous component in the pipe, a B1 module. The DATA_OUT output port is connected to the next component in the pipe,

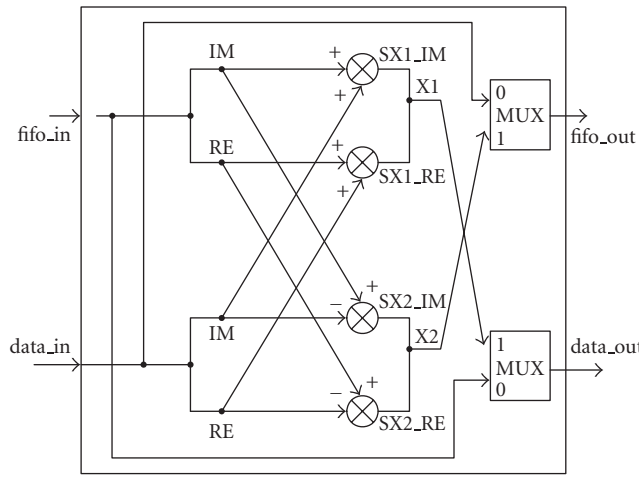


Figure 6: Arithmetic operations in the butterfly of type 1 (BF1).

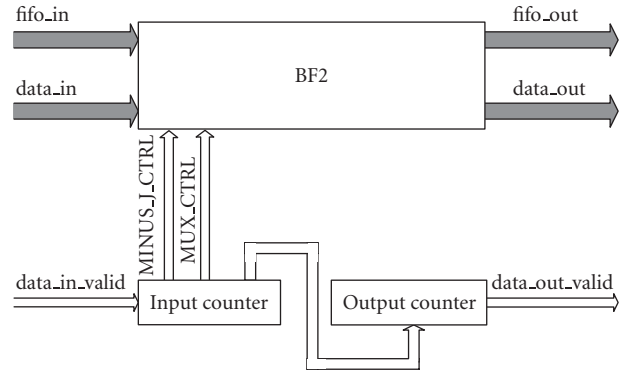


Figure 7: B2 module structure.

usually a CM module. The FIFO_IN and FIFO_OUT ports are connected to FIFO2.

Figure 8 details the implementation of BF2. Its structure is similar to BF1. FIFO2 is filled with the first $N/2^{k+2}$ data. Then, the multiplexers MUX change to position 1, and the input data at port DATA_IN and the data stored in FIFO2 are used to perform the required butterfly operations. The butterfly output X1 is sent out, and X2 is stored in FIFO2. After $N/2^{k+2}$ cycles, the multiplexers MUX switch back to position

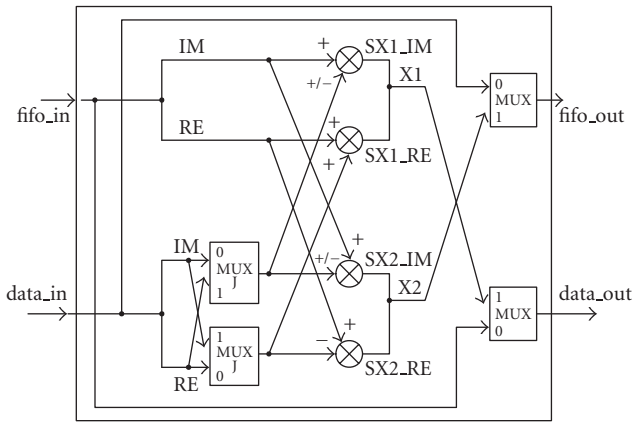


Figure 8: Arithmetic operations in the butterfly of type 2 (BF2).

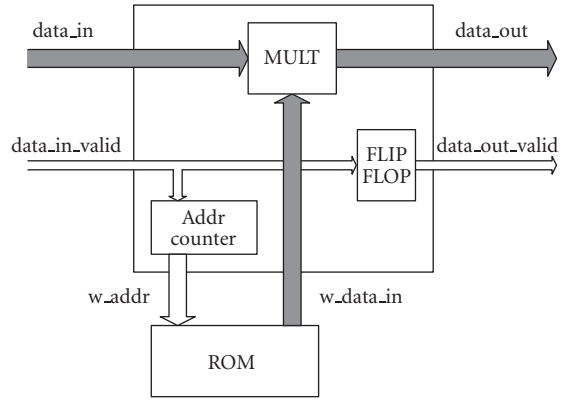


Figure 9: CM module structure.

0. Data for the next computation is stored in FIFO2, and the results X2 of the previous butterfly operations are output.

The multiplexers MUX J are used to handle efficiently the multiplications by $-j$ needed in a butterfly of type 2. Whenever a multiplication by $-j$ must be carried out, the multiplexers MUX J are set to position 1. The signal MINUS_J_CTRL controls the behavior of the multiplexers MUX J.

The input counter generates the signals that control the multiplexers MUX and MUX J. This counter increments its value when DATA_IN_VALID is high. Its size is $2(\log_4 N - k)$. The second most significant bit of this counter's value is used to generate MUX_CTRL. When the input counter is making the last quarter of its count, MINUS_J_CTRL is set to high.

When the input counter arrives to a quarter of its count, DATA_OUT_VALID is set to high, and the output counter starts to count. The output counter will count until FIFO2 is emptied of valid output data. When the output counter finishes its count, DATA_OUT_VALID is set to zero. The size of the output counter is $2(\log_4 N - k) - 1$.

3.1.3. Module CM

The internal structure of CM is shown in Figure 9. This component carries out the twiddle factor multiplications. A one clock cycle complex multiplier has been implemented to perform the complex multiplications between the input data and the twiddle factors. The twiddle factors are read from a synchronous ROM. A counter of size $2(\log_4 N - k)$, addr counter, is used to generate the ROM addresses appropriately. A flip-flop is used to synchronize the DATA_OUT_VALID signal with DATA_OUT.

3.2. Multiple mode operation

In order to accommodate the 2K/4K/8K multiple mode, some extra elements are needed in the r2^2 SDF architecture. The proposed architecture is depicted in Figure 10. As can be seen, the resources needed to process the FFT of the largest number of points, $N_{\max} = 8192$, have been implemented.

Table 1: Memory requirements.

FIFO	ROM
$2dbw \cdot (N_{\max} - 1)$	$2dbw \cdot \sum_{k=0}^{\text{ceil}(\log_4 N_{\max})-2} \frac{N_{\max}}{2^{2k}}$

Thus, there are $a = 7$ stages, and the twiddle factors have been calculated for $N_{\max}$.

When an 8K point FFT is to be calculated, the multiplexers shown in Figure 10 are configured so that the core works as described above. Multiplexers M4K and M2K are in position 0. When multiplexers are in position 0, they select the signal connected to the upper port.

In order to calculate a 2K point FFT, the first stage of the 8K FFT, stage 0 in Figure 10, does not have to be processed. Thus, multiplexers M2K are set to position 1 to bypass stage 0. Multiplexers M4K remain in position 0.

For the 4K points FFT, six complete stages (with both types of butterflies) are needed. In order to reuse the existing hardware, B1 of stage k uses FIFO2 of stage k, and B2 of stage k uses FIFO1 of stage $k+1$. Additionally, CM of stages 5 and 6 is bypassed. The former is achieved by setting M4K to position 1 and M2K to position 0. In each stage, CM needs half the twiddle factors of the 8K points FFT: those with an even address in the ROM memory. A control signal configures CM for proper operation according to the number of points of the FFT to be calculated.

3.3. Features of the proposed FFT core

Table 1 summarizes the memory requirements of the core. The table shows the total number of memory bits used in the FIFOs and in the twiddle factor ROMs. In order to achieve more compact memories, the real and imaginary parts of each complex number are stored in the higher and lower part of the same memory position. Table 2 is a summary of the arithmetic operators needed in the core. It can be noted that the memory and arithmetic modules needed in the proposed multiple mode architecture are the same as those needed in a single mode 8K FFT.

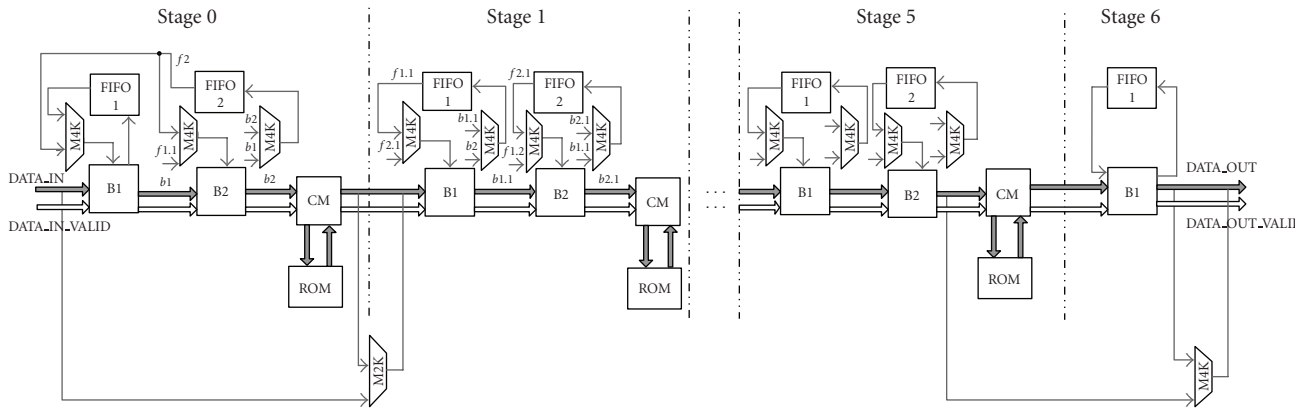


Figure 10: Architecture of the proposed 2K/4K/8K FFT.

Table 2: Arithmetic operators.

	Input bitwidth	Output bitwidth	Quantity per module	Quantity in the core
Adders (B1)	*dbw*	*dbw*	2	$2 \cdot \text{ceil}(\log_4 N_{\max})$
Subtractors (B1)	*dbw*	*dbw*	2	$2 \cdot \text{ceil}(\log_4 N_{\max})$
Adders (B2)	*dbw*	*dbw*	2	$2 \cdot \text{floor}(\log_4 N_{\max})$
Subtractors (B2)	*dbw*	*dbw*	2	$2 \cdot \text{floor}(\log_4 N_{\max})$
Multipliers (CM)	*dbw* and *tbw*	*dbw+tbw*	4	$4 \cdot \text{ceil}(\log_4 N_{\max} - 1)$
Adders (CM)	*dbw+tbw*	*dbw*	1	$\text{ceil}(\log_4 N_{\max}) - 1$
Subtractors (CM)	*dbw+tbw*	*dbw*	1	$\text{ceil}(\log_4 N_{\max}) - 1$

Once the clock frequency f_{clk} has been selected, the processing time t_{proc} of the FFT module can be determined using

$$t_{\text{proc}} = \frac{1}{f_{\text{clk}}} \cdot \left(\frac{N}{2} + 3 \cdot \text{ceil}(\log_4 N) - 2\right). \quad (2)$$

4. RESULTS

4.1. Analysis of the signal-to-noise ratio of the FFT

For an appropriate operation of the receiver, the degradation introduced in the signal due to the fixed-point computation of the FFT must be controlled. Monte Carlo simulations have been carried out comparing a fixed-point model of the proposed architecture with a floating-point FFT. The signal-to-noise-ratio (SNR) has been used to measure the degradation.

Figure 11 shows the SNR for the 2K/4K/8K FFT when the data are quantized, and the twiddle factors are left at floating point. It can be observed that both the data bitwidth and the number of points of the FFT influence the SNR. As N increases, the number of arithmetical operations grows and, thus, the SNR decreases. The SNR increases in 6 dB per bit of *dbw*.

The effect of the quantization of the twiddle factors in the SNR has been studied for 2K (Figure 12(a)), 4K (Figure 12(b)), and 8K (Figure 12(c)) FFTs. These figures show the SNR for different values of *dbw* and *tbw*. It can be seen that for a given value of *dbw* and of N, increasing *tbw* above a certain value does not improve the performance.

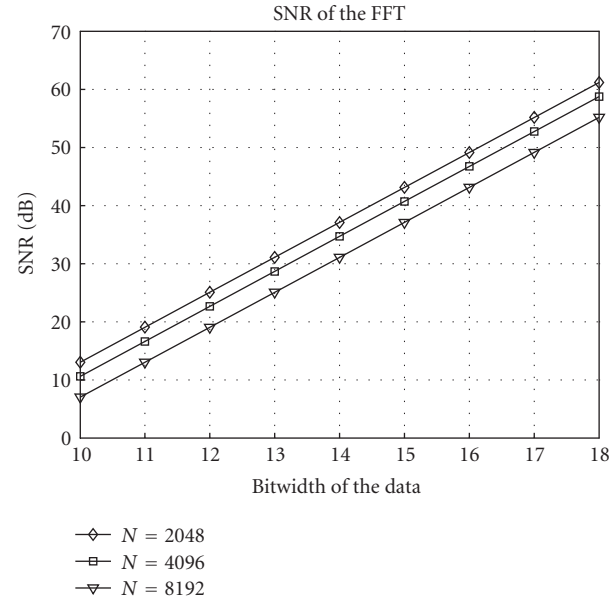
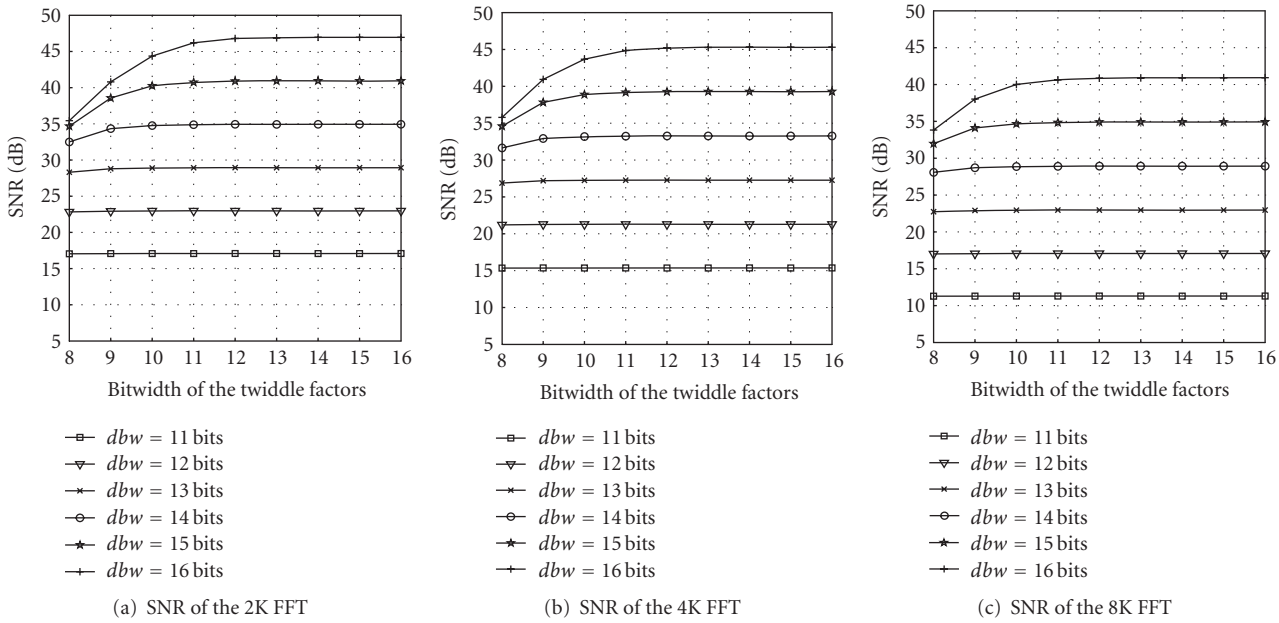


Figure 11: SNR for different data bitwidths (*dbw*) and number of points of the FFT (N). The twiddle factors are not quantized.

Figures 11 and 12 can help the designer in the selection of *dbw* and *tbw*. In a multiple mode FFT processor, *dbw* and *tbw* must be selected for the maximum number of points of the FFT (8K in a DVB-T/DVB-H receiver). A SNR of at least

Table 3: DVB-T/DVB-H timing specifications and processing time of the FFT core.

	6 MHz (f_s = 48/7 MHZ)			7 MHz (f_s = 8 MHZ)			8 MHz (f_s = 64/7 MHZ)		
	2K	4K	8K	2K	4K	8K	2K	4K	8K
$t_{\text{OFDM-SYMBOL}}$ (μs)	308	616	1232	264	528	1056	231	462	924
t_{proc} (μs)	151.6	301	600.1	148.6	294.8	587.8	113.7	225.7	450



(a) SNR of the 2K FFT (b) SNR of the 4K FFT (c) SNR of the 8K FFT

Figure 12: SNR for different values of data bitwidth (dbw) and twiddle factor bitwidth (tbw) for (a) N = 2K, (b) N = 4K, and (c) N = 8K.

40 dB is sufficient for terrestrial TV broadcasting [15]. In order to guarantee a SNR of 40 dB for the 8K FFT, $dbw = 16$ and $tbw = 11$ are needed.

4.2. Analysis of the timing of the FFT for DVB-T/DVB-H

For low-power applications, such as a DVB-T/DVB-H receiver, a slow-clock frequency is preferable: for example, equal to the sample rate of the FFT. For DVB-T/DVB-H, the minimum sample rate at the FFT (f_s) can be 48/7 MHz for a 6 MHz channel, 8 MHz for a 7 MHz channel, and 64/7 MHz for an 8 MHz channel [2]. However, the FFT processor shall be able to compute the FFT within the duration of an OFDM symbol plus the duration of the guard interval ($t_{\text{OFDM-SYMBOL}}$).

Table 3 presents the maximum allowed time to compute the FFT in DVB-T/DVB-H and the processing time of the proposed FFT core. Results are given for the different bandwidth channels (6 MHz, 7 MHz, and 8 MHz) and for the three lengths of the FFT (2K, 4K, and 8K). The value of $t_{\text{OFDM-SYMBOL}}$ requirement given in the table considers the worse case scenario: a guard interval with duration of 1/32 of the OFDM symbol period. The processing time of the FFT core, t_{proc}, has been calculated for a clock frequency equal to the corresponding sampling frequency f_s. It can be observed that the proposed FFT core is able to meet the timing requirements in all cases.

4.3. Area and timing comparison

Table 4 compares the proposed core with reported FFTs that could be used within DVB-T/DVB-H applications. The FFTs presented in [6, 10] have been designed for DVB-T, and they do not implement the 4K mode. The work in [7] only provides results for 8K.

In [4], a 2K/4K/8K FFT architecture is proposed. The twiddle factor multiplication is carried out using a CORDIC, and no ROM is needed for the twiddle factors. The CORDIC carries out 17 iterations to guarantee good performance. For comparison, we will relate the number of iterations to the precision of the twiddle factors. Following [16], we can estimate the precision in the rotated angle as $\delta_\theta \approx 2^{-(N_i-1)}$, where N_i represents the number of iterations. A quantization error in the twiddle factors can be seen as an error in rotating an angle. It can be shown that $\max(\delta_\theta) \approx 2^{-tbw}/(1-2^{-tbw})$. Thus, we can perform the following approximation:

$$tbw \approx \log_2\left((1+2^{-(N_i-1)})/2^{-(N_i-1)}\right) \approx N_i - 1. \quad (3)$$

Table 4: Comparison with other FFTs in the literature.

	dbw	*tbw*	f_{clk} (MHz)	Area (mm^2)	t_{proc} (μs)	AT (μs $\cdot$mm^2)	*N*
[4]	16	16$^{(*)}$	30	66	273	18018	2/4/8K
[6]	8	8	64	28.39	897	25465	2/8K
[7]	11	11	20	18.29	717.35	13120.33	8K
[10]	8	8	16	33.75	—	—	2/8K
Ours	16	11	9.143	18.7	450	8415	2/4/8K

$^{(*)}$This value is an approximation.

Table 5: Hardware complexity of FFT cores for DVB-T/H.

Architecture	Radix	Multipliers	Adds/Subs
Parallel [6]	2	64	96
Pipeline-SDF	2	48	76
(Ours)	2^2	24	64

The area and timing results shown in Table 4 for the proposed multiple mode FFT core are given for the selected *dbw* and *tbw*. They have been obtained using the 0.35 μm XFAB 4-ML technology. The area value given for the proposed FFT processor is an estimation of the core area after layout, making the assumption that the layout area is twice the cell area. For a fairer comparison, the area of [6, 7] has been normalized to 0.35 μm using the same approach as [7]. In [4], the number of equivalent gates is provided. The value given in Table 4 has been estimated for a 0.35 μm technology using that number.

Table 4 shows the parameter AT as well. AT is the product between the area and the processing time t_{proc}. This parameter can be used to assess the efficiency of different cores. The table shows that the proposed core is the most efficient.

In addition, Table 5 presents a comparison of the computational complexity of [6], a pipeline-SDF r2 FFT design, and our proposal. As can be observed, our design requires less multipliers and adders. Thus, our FFT core presents a more efficient implementation.

4.4. Area and timing results for FPGA implementation

The FFT core has been prototyped in an FPGA virtex 2V6000FF1517. Table 6 presents a comparison of our core with other FFT cores for DVB-T in the literature. Only those proposals that give data about an FPGA implementation have been considered in the comparison. The table shows the working clock frequency, the total number of occupied slices, the necessary block RAMs, and the number of multipliers. One can observe that our FFT core presents the most efficient implementation for an FPGA.

4.5. Area, timing, and power results for ASIC implementation

The layout of the FFT core, with $dbw = 16$ and $tbw = 11$, has been carried out for the 0.35 μm AMS 3-ML technology. The FIFOs of stages 0 to 3 have been implemented using single port RAMs and some additional control logic, whereas the FIFOS of stages 4 to 6 have been implemented using standard cells. A detailed summary of the features of the proposed FFT core is presented in Table 7. The power consumption of the proposed FFT processor has been calculated at synthesis level. The switching activity has been extracted from simulations operating in the 8K mode. A chip photo of the layout of the FFT core for 0.35 μm AMS 3-ML technology is shown in Figure 13.

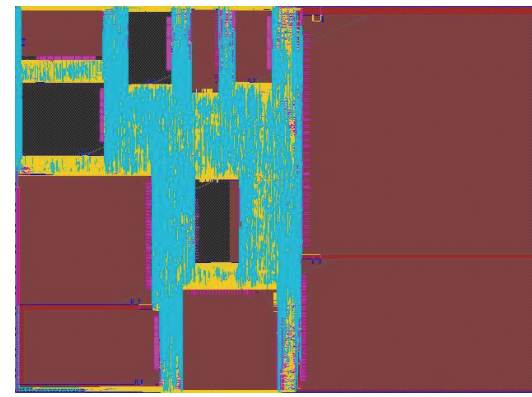


Figure 13: Layout of the 2K/4K/8K complex-point FFT core fabricated in a 0.35 μm technology, 4-ML CMOS process. The core size is 18.7 mm^2.

To sum up, our core simplifies the twiddle factors and, thus, reduces the number of multiplications. Therefore, our FFT proposal for a DVB-T/DVB-H system results in a more efficient ASIC and FPGA implementation than the proposals found in the literature.

5. CONCLUSION

An FFT core for DVB-T/DVB-H receivers has been designed and implemented. The core implements a pipeline r2^2 SDF architecture. This architecture can be adapted to achieve an efficient 2K/4K/8K multiple mode FFT processor. The extra hardware needed for multiple mode operation is minimal. In order to guarantee a SNR of 40 dB in all modes of operation, 16 bits and 11 bits are needed for the data bitwidth and the twiddle factor bitwidth, respectively. The architecture of the proposed FFT processor makes it possible to achieve the FFT processing time requirements of DVB-T/DVB-H working at the lowest-possible clock frequency. The proposed core is an

Table 6: Area and timing results in an FPGA virtex 2V6000FF1517.

	dbw	*tbw*	f_{clk} (MHz)	Occupied slices	BRAMs	Multipliers
[6]	16	11	64	17305 (51%)	96 (66%)	16 (11%)
Ours	16	11	9.143	6066 (17%)	19 (13%)	24 (16%)

Table 7: Chip summary of our FFT processor.

Items	Specification
FFT size	2K/4K/8K
Clock frequency	64/7 MHz
Data bitwidth (*dbw*)	16 bits
Twiddle factor bitwidth (*tbw*)	11 bits
Signal-to-quantization-noise ratio (SQNR) for $N_{max} = 8K$	40.6 dB
Process technology	0.35 μm XFAB 4-ML
Supply voltage	3.3 V
Execution time (clock cycles) for $N = 2K$	1040 clock cycles
Execution time (clock cycles) for $N = 4K$	2064 clock cycles
Execution time (clock cycles) for $N_{max} = 8K$	4115 clock cycles
Core power consumption for $N_{max} = 8K$	114.65 mW
Core size	18.7 mm^2

efficient implementation well suited for DVB-T/DVB-H receivers.

ACKNOWLEDGMENTS

This research is supported in part by the *Ministerio de Industria, Turismo y Comercio* Grant no. FIT330100-2006-43 and by the Basque Government. A. Cortés holds the Torres Quevedo Grant no. PTQ05-02-02455, which was awarded by the Spanish Ministry of Education and Science, by the European Regional Development Fund and by the European Social Fund.

REFERENCES

[1] G. Faria, J. A. Henriksson, E. Stare, and P. Talmola, "DVB-H: digital broadcast services to handheld devices," *Proceedings of the IEEE*, vol. 94, no. 1, pp. 194–209, 2006.

[2] ETSI EN 300744, "Digital video broadcasting (DVB); framing structure, channel coding and modulation for digital terrestrial television," 2004.

[3] U. H. Reimers, "DVB-The family of international standards for digital video broadcasting," *Proceedings of the IEEE*, vol. 94, no. 1, pp. 173–182, 2006.

[4] S. Y. Park, N. I. Cho, S. U. Lee, K. Kim, and J. Oh, "Design of 2K/4K/8K-point FFT processor based on cordic algorithm in OFDM receiver," in *Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM '01)*, vol. 2, pp. 457–460, Victoria, BC, Canada, August 2001.

[5] J. F. Sevillano, A. Mtz de Gereñu, M. Leyh, P. Nagel, and A. Irizar, "An FFT parametrizable core," in *Proceedings of the 15th Conference on Design of Circuits and Integrated Systems (DCIS '00)*, pp. 230–234, Montpellier, France, November 2000.

[6] A. Cortés, I. Vélez, J. F. Sevillano, and A. Irizar, "An approach to simplify the design of IFFT/FFT cores for OFDM systems," *IEEE Transactions on Consumer Electronics*, vol. 52, no. 1, pp. 26–32, 2006.

[7] Y.-W. Lin, H.-Y. Liu, and C.-Y. Lee, "A dynamic scaling FFT processor for DVB-T applications," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 11, pp. 2005–2013, 2004.

[8] T. Sansaloni, A. Pérez-Pascual, V. Torres, and J. Valls, "Efficient pipeline FFT processors for WLAN MIMO-OFDM systems," *Electronics Letters*, vol. 41, no. 19, pp. 1043–1044, 2005.

[9] Y. Jung, H. Yoon, and J. Kim, "New efficient FFT algorithm and pipeline implementation results for OFDM/DMT applications," *IEEE Transactions on Consumer Electronics*, vol. 49, no. 1, pp. 14–20, 2003.

[10] C.-C. Wang, J.-M. Huang, and H.-C. Cheng, "A 2K/8K mode small-area FFT processor for OFDM demodulation of DVB-T receivers," *IEEE Transactions on Consumer Electronics*, vol. 51, no. 1, pp. 28–32, 2005.

[11] S. He and M. Torkelson, "A new approach to pipeline FFT processors," in *Proceedings of the 10th International Parallel Processing Symposium (IPPS '96)*, pp. 766–770, Honolulu, Hawaii, USA, April 1996.

[12] J.-Y. Oh and M.-S. Lim, "New radix-2 to the 4th power pipeline FFT processor," *IEICE Transactions on Electronics*, vol. E88-C, no. 8, pp. 1740–1746, 2005.

[13] T. J. Ding, J. V. McCanny, and Y. Hu, "Rapid design of application specific FFT cores," *IEEE Transactions on Signal Processing*, vol. 47, no. 5, pp. 1371–1381, 1999.

[14] C.-P. Hung, S.-G. Chen, and K.-L. Chen, "Design of an efficient variable-length FFT processor," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '04)*, vol. 2, pp. 833–836, Vancouver, BC, Canada, May 2004.

[15] E. Bidet, D. Castelain, C. Joanblanq, and P. Senn, "A fast single-chip implementation of 8192 complex point FFT," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 3, pp. 300–305, 1995.

[16] Y. H. Hu, "The quantization effects of the CORDIC algorithm," *IEEE Transactions on Signal Processing*, vol. 40, no. 4, pp. 834–844, 1992.