# HARDWARE IMPLEMENTATION OF A POLYPHASE FILTER BANK FOR MP3 DECODING

*María Dolores Valdés, María José Moure*

Department of Electronic Technology
Institute of Applied Electronic
University of Vigo
Vigo, Spain
email: mvaldes@uvigo.es, mjmoure@uvigo.es

*Javier Diéguez, Santiago Antelo*

Technological Automotive Centre in Galicia
(CTAG)
Porriño, Spain
email: javier.dieguez@ctag.com,
santiago.antelo@ctag.com

## ABSTRACT

MP3 decoding is usually implemented in the software due to the complexity of its hardware solution. The aim of this work is the hardware implementation of the polyphase filter bank, the most computationally intensive operation in the MP3 decoder, in order to improve the decoder operation speed while saving power. The Altera's Stratix EP1S10F780C6ES FPGA has been used as hardware support taking advantage of its architecture oriented to DSP applications. DSP Builder design tools have been used together with Matlab and Quartus II in order to simplify the design and simulation tasks. As a result, a synthesis polyphase filter bank, working in real time, was designed and tested.

## 1. INTRODUCTION

MPEG-1 Audio Layer 3, more commonly referred to as MP3, is an audio encoding format corresponding to the standard ISO/IEC 11172-3, *"Coding of moving picture and associated audio for digital storage media at up to about 1,5 Mbit/s"* [1]. Its high compression ratio and good audio fidelity makes MP3 a commercial success.

Nowadays, MP3 decoding is typically done by software, or, in the case of portable devices, by hardware which uses digital signal processors (DSP) [2] or general purpose high-performance microprocessors. For example, hardware solutions like the popular MintyMP3 [3] or the one proposed in [4] combines a high power 8-bit microcontroller with ASICs. Other solutions are based on RISC architecture platforms, optimized for MP3 decoding [5], and many others focus on the software optimization [6]. Nevertheless, the power consumption of processor-based solutions is often higher than for ASIC based hardware. In the case of MP3 portable devices low power consumption is especially important, so specialized hardware could be a good solution in order to save power.

Recently, there have been some works published, which are oriented to the hardware acceleration of audio coding algorithms using FPGAs. An example is proposed in [7], where software and ASIC solutions are compared. The FPGAs which include DSP oriented resources, enable the development of traditional DSP implemented algorithms, thereby allowing higher speed and reconfigurability. The aim of this work is the hardware implementation of the polyphase filter bank, the most computationally intensive operation in the MP3 decoder, so that its operation speed can be improved with a low power consumption.

## 2. INTRODUCTION TO MP3

Fig. 1 shows a diagram of the decoding process for an audio encoded bitstream, according to the standard ISO/IEC 11172-3 [1]. A polyphase filter bank consists of a set of parallel filters where each one is centred on a specific band, resulting in a unique filter which works over a wider band. In the case of the MPEG decoder, the filter bank is made up of 32 uniform bands; so all filters have the same transference function but they are shifted in the frequency axis. Starting from a prototype filter $h_o(n)$, valid for the first band, the rest of the filters are shifted in frequency in order to centre them on their corresponding bands.

Fig. 2 shows the algorithm for the synthesis polyphase filter, as is described in the standard ISO/IEC 11172-3 [1]. Input encoded audio samples are processed in blocks of 32 elements. The 32 input samples, S0 … S31, are multiplied by a $N_{ik}$ matrix defined as in Eq. (1). The resulting values are organized in a $U_i$ vector of 512 elements. The $U_i$ vector is windowed using 512 coefficients ($D_i$), whose distribution is represented in Fig. 3(a). A new vector $W_i$ is obtained as a result. The 32 reconstructed PCM output samples are obtained through consecutive additions of the $W_i$ elements.

$$N_{ik} = \cos\left[(16+i)(2k+1)\frac{\pi}{64}\right] \quad 0 \le i \le 63, 0 \le k \le 31 \quad (1)$$

Fig. 3(b) represents the time domain filter response where the coefficients of Fig. 3(a) are used. In Fig. 3(c) the

978-1-4244-1992-0/08/$25.00 ©2008 IEEE

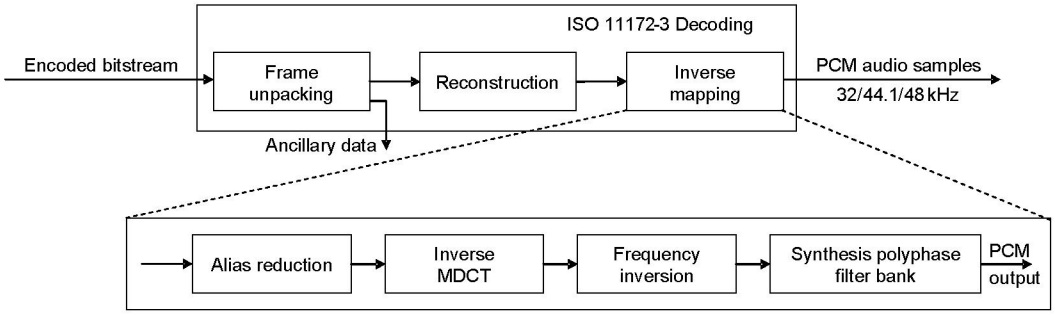


**Fig. 1.** Decoding process for an MP3 audio encoded bitstream.

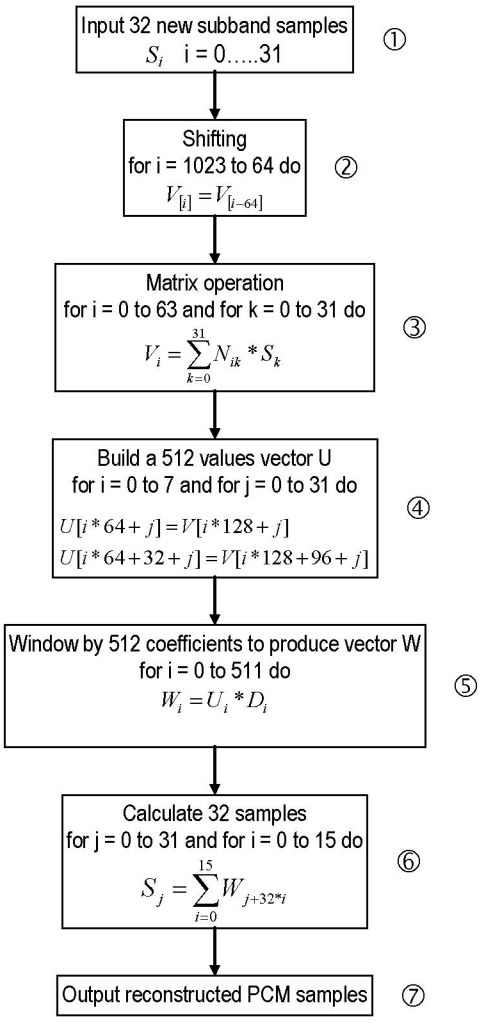


**Fig. 2.** Algorithm of the synthesis polyphase filter.

frequency response of the first four filters of the synthesis polyphase filter bank is shown.

From the analysis of Fig. 2, it can be inferred that software implementation of the algorithm is feasible. Nevertheless, if the objective is its hardware implementation, the algorithm of Fig. 2 is not suitable due to the excessive memory (about 4672 positions of 24 bits)

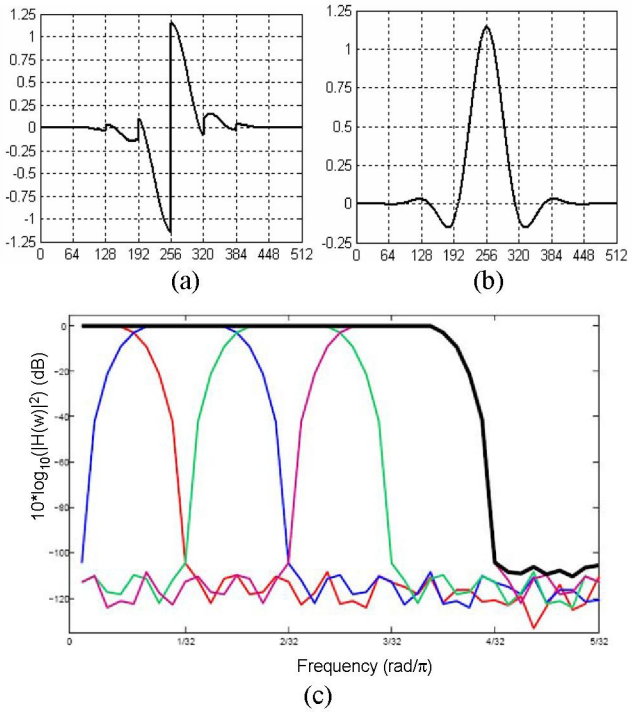


**Fig. 3.** (a) Coefficients Di. (b) Impulse response of the filter. (c) Frequency response of the first four filters of the filter bank.

and the computational capacity required (about 2560 24-bits multipliers and 2464 24-bits adders). For this reason we looked for an equivalent algorithm, requiring less processing capacity, and we named it *fast algorithm*.

## 3. THE FAST ALGORITHM

In [8], Sung Hee Park proposes a fast algorithm based on the results obtained by Konstantinos Konstantinides [9].

This algorithm replaces phases ③ to ⑥ of the polyphase filtering process shown in Fig. 2 by two independent and concurrent processes: a 32-point DCT-II and a windowing function. As a result, the required computations are reduced to 696 additions and 464 products.

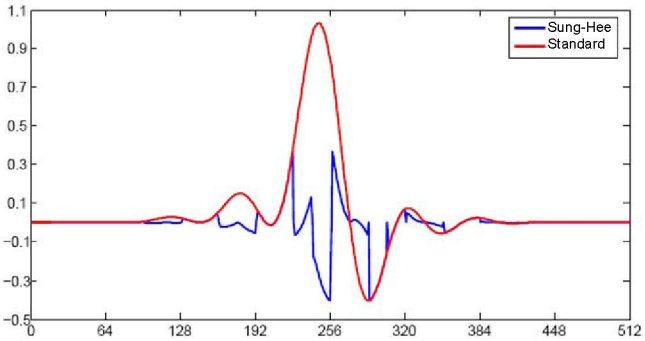


**Fig. 4.** Impulse response of the Standard algorithm and the Sung Hee Park's algorithm using a delta function as input.

Despite of the advantage of this algorithm, the results obtained when executing the algorithm proposed in [8] , as shown in Fig. 4, are not the same as those obtained with the standard ISO/IEC 11172-3 [1]. Bearing in mind this limitation we have developed a new algorithm which modifies some operations of the algorithm proposed by Sung Hee Park in [8]. The new algorithm, refered to as *modified fast algorithm*, can be analyzed in [10]. In this case the results of the modified fast algorithm and the standard algorithm are identical.

## 4. HARDWARE IMPLEMENTATION

To implement the polyphase filter bank, the 32-point DCT-II and the windowing functions have been designed and implemented separately and then interconnected. As the filter bank is intended to be implemented on an specific device (Altera Stratix EP1S10F780C6ES FPGA [11]), some design constraints regarding logic resources availability and timing, imposed by the device architecture, had to be taken into account.

Moreover, due to the architecture of the FPGA, floating arithmetic can not be used. Therefore, a *signed-fractional* format has been chosen to represent data and coefficients. This fixed-point format allows setting the number of bits for the integer and for the fractional parts, respectively.

In a first approach of the system implementation, 24 bits were used to represent data and coefficients but finally this was reduced to 16, due to the high number of logic resources required. The conclusions of this work demonstrate that 16 bits are enough to obtain a good resolution.

The coefficients are stored as independent constants allowing different signed fractional formats. A 2.14 format (2 bits for the integer part and 14 for the fractional part) has been selected for coefficients lower than unity. To represent coefficients greater than unity, the number of bits of the integer part should be increased whilst reducing those of the fractional part.

Input and output data use one bit for the sign and one for the integer part. Input data is normalized, so it is always lower than unity.

In the next paragraph, the hardware design for the different parts of the filter bank is analyzed.

The DSP Builder Tool [12] was used to describe the different parts of the system. This tool is oriented to the design of DSP applications and allows a high level description from Simulink (Matlab). It was easy to use this tool and quick to make comparisons between the expected results of the standard algorithm ISO/IEC 11172-3 implemented in Matlab [10] and the simulation results of our design.

### 4.1. 32-point DCT-II

There are many fast algorithms for the computation of a type-II Discrete Cosine Transform (DCT-II). In this case we started from the algorithm proposed by Byeong Gi Lee [13] to implement an 8-point DCT-II and extended it to get a 32-point DCT-II. Our main reason for selecting this algorithm was its simplicity.

The direct implementation of this algorithm leads to a totally parallel architecture allowing the computation of the 32 output samples of the DCT in only one operating cycle. Nevertheless, as indicated in [10], this solution requires 80 multipliers and 209 adders of 32-bit data, which makes its implementation unfeasible in the EP1S10F780C6ES device.

After analysing three different implementation alternatives (*Recursive DCT*, *Distributed arithmetic* and *Residue Number System (RNS)* [10]) without relevant results, we chose to modify the algorithm proposed by Byeong Gi Lee in [13]. A semi-parallel structure, based on the reuse of some functional blocks of the DCT, has been implemented. Besides, the number of bits used to represent the coefficients was reduced from 24 to 16. We have found that the precision obtained is acceptable. An error analysis with respect to a theoretical DCT, performed in Matlab, reports that the Overall Mean Square Error (OMSE) is $1.6677 \times 10^{-6}$, the Mean Square Error (MSE) is $7.2577 \times 10^{-8}$ and the Peak Signal to Noise Rate (PSNR) is 67.054 dB. A comparison with other documented solutions [14] shows that such errors are acceptable.

Table 1 and Fig. 5 show a comparison between the solution proposed by Byeong Gi Lee in [13] and the final implemented solution (named *Modified Lee*). Fig. 6 shows the results obtained with the 32-point DCT-II, implemented in the FPGA using Altera's Development Tools (Quartus II), and those obtained from a theoretical DCT modelled in Matlab. As can be seen, there are no significant differences between both output vectors. The error magnitude is in the order of $10^{-4}$.

**Table 1.** Comparison regarding resource usage: Lee vs. Modified Lee.

| Item \ Solution | Byeong Gi Lee | Modified Lee |
|---|---|---|
| Multipliers | 80 | 36 |
| Adders | 209 | 104 |
| Registers | 0 | 64 |
| Muxes | 0 | 24 |
| Coefficients | 31 | 31 |
| Operating cycles | 1 | 5 |

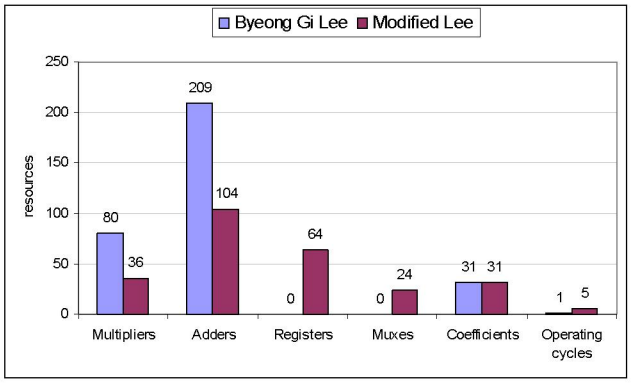


**Fig. 5.** Comparison regarding resource usage: Lee vs. Modified Lee.

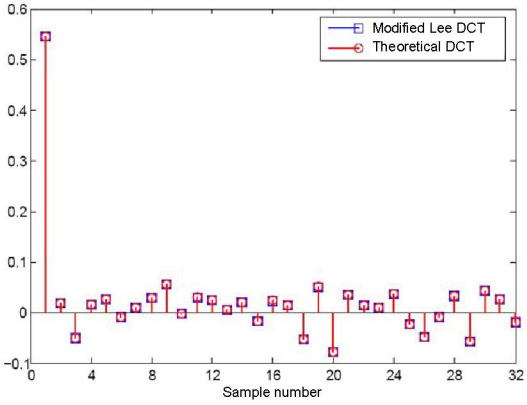


**Fig. 6.** Comparing the output vector of a theoretical 32-point DCT-II with the one implemented.

### 4.2. Windowing function

The windowing function proposed by Sung Hee Park [14] can be assimilated to a complex product, like the one represented in Fig. 7. Its result is accumulated to a previous one to generate the output samples of the filter. This complex product allows for the simultaneous computation of two output samples in only one operating cycle, using two input samples and three coefficients (Dsub, Dsum and D2 in Fig. 7). This implies the parallel operation of two complementary banks of the filter in each

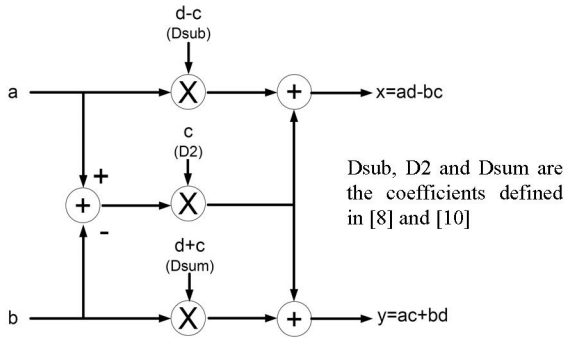


**Fig. 7.** Scheme of the simplified complex product.

cycle. There are 16 banks and each one generates 17 samples (16 are generated by means of the complex product and one using a simple product).

The coefficients D2, Dsum and Dsub are derived from the coefficients in vector D (Fig. 3a), as defined in standard ISO/IEC 11172-3. The algorithm to calculate D2, Dsum and Dsub is included in [10].

The hardware implementation of this block is not based on any previous work. To optimize the operating frequency the following design decisions were undertaken [10]:

- Double-port memories allowing the reading of two data in the same clock cycle were used to store the input samples coming from the 32-point DCT-II.
- The coefficients D2, Dsum and Dsub are stored in ROM memories. Each memory has 136 positions of 24 bits.
  These memories are read linearly, and in order to enhance operational efficiency, the coefficients are stored according to their order of use.
- A double-port memory is also used to store the output matrix comprising 16 rows, each one corresponding to a bank and 16 columns, which correspond to the 16 output samples of each bank. This arrangement provides two output samples, one for each bank, in only one operating cycle.
- DSP blocks available in the EP1S10F780C6ES device were used to implement the multiplications and additions involved in the computation of the complex product. In some cases, distributed logic resources were used to support these functions.

The windowing function implemented was simulated using Quartus II and the results were compared with those coming from a model of the *Modified fast algorithm* described in Matlab [10].

Once the 32-point DCT-II and the windowing function were designed and tested, they were interconnected to obtain the complete polyphase filter. Additional logic was required to implement phase ①, ② and ⑦ of the algorithm proposed in Fig. 2, as well as to guarantee the concurrent operation of the DCT-II and the windowing function.

## 5. IMPLEMENTATION RESULTS

The correct operation of the implemented polyphase filter has been fully verified. Several timing simulations have been carried out using the Quartus II design tools. Random values were used as test data. Timing simulation results have been compared to those obtained from a theoretical model of the standard ISO/IEC 11172-3 (implemented in Matlab [10]) using the same test data. In all cases the output vectors of both solutions were the same as [10].

Table 2 shows these compilation results using Quartus II design tool. Such compilation has been configured to obtain an optimum balance between the silicon area used and the speed.

**Table 2.** Logic resources usage.

| Family | Stratix |
|---|---|
| Device | EP1S10F780C6ES |
| Total logic elements | 8092 / 10570 (76%) |
| Total pins | 71 / 427 (16%) |
| Total memory bits | 25600 /920448 (2%) |
| DSP block 9-bit elements | 48 / 48 (100%) |

A maximum operating frequency of 17.43 MHz has been obtained, which allows for the real time operation of the filter. This is validated by the fact that the 32-point DCT requires 53 cycles to generate the input samples for the windowing function, and this one needs 272 cycles to generate the output samples. Although both modules run concurrently and valid results are obtained every 272 cycles, the initial system latency amounts to 325 cycles (53 cycles + 272 cycles). Assuming that a 10 MHz clock is used (a cycle period of 100 ns), the operating period of the filter is 32.5 µs (325 cycles * 100 ns), whereas the MPEG standard [1] specifies a maximum sampling rate of 48000 samples/s, which means that new input samples are available every 666.66 µs (32 samples / 48000 samples/s). By this reason, it is possible to lower the clock frequency to almost 1MHz, and still comply with real-time constraints.

## 6. CONCLUSIONS AND FUTURE WORK

This work evidences the potential of new FPGA architectures to accelerate some digital signal processing algorithms which traditionally are performed in software. Combining the parallelism supported by configurable logic devices with the availability of specific DSP resources makes this design alternative feasible and attractive. Compared to a processor-based decoder, the power consumption can be reduced by using a specialized hardware solution. In the case of the MP3 decoder, our future work is focused on the design of a SoC system connecting the developed synthesis polyphase filter with the Nios® II embedded soft processor. The processor would execute the rest of the stages involved in the inverse mapping function of the MP3 decoding working in parallel with the polyphase filter bank. It is expected that a bigger FPGA will be required due to the resources consumed by both, the Nios® II processor and the polyphase filter, as well as the glue logic required to connect them.

## 7. REFERENCES


[1] ISO/IEC. Information technology - coding of moving pictures and associated audio for digital storage media at up to about 1,5 mbit/s. ISO/IEC 11172, 1993.

[2] J. Eilert, A. Ehliar and Dake Liu, "Using low precision floating point numbers to reduce memory cost for MP3 decoding", *IEEE 6th Workshop on Multimedia Signal Processing*, pp. 119 – 122, 2004.

[3] MintyMP3, http://www.ladyada.net/make/minty/index.html

[4] Yang Jan-Ti and Jun-Ming Yu, "A novel hardware implementation of MP3 decoder for low power and minimum chip size", *6th International Conference On ASIC (ASICON)*, vol. 2, pp. 1085 – 1088, 2005.

[5] Chang-Hung Yang, Chin-Yu Huang, Tsui-Ying Hung, Tung-Ju Chiang and Yung-Ruei Chang, "Software and Hardware co-design for MP3 Decoder", *IEEE International Region 10 Conference (TENCON)*, pp. 1 – 4, 2006.

[6] Yingbiao Yao, Qingdong Yao, Peng Liu and Zhibin Xiao, "Embedded Software Optimization for MP3 Decoder Implemented on RISC Core", *IEEE Transactions on Consumer Electronics*, vol. 50, No. 4, Nov. 2004.

[7] F. Mayer, D. Dalquen and T. Dettbarn, "Hardware accelerating audio coding algorithms", *International Conference on Consumer Electronics (ICCE)*, pp. 283 – 284, 2006.

[8] Sung Hee Park, "Fast algorithm on MPEG/audio subband filtering", *99th Convention of the Audio Engineering Society (AES)*, New York, Oct. 1995.

[9] K. Konstantinides, "Fast subband filtering in MPEG audio coding", *IEEE Signal Processing Letters*, 1(2), Feb. 1994.

[10] J. Diéguez and M. D. Valdés, "Estudio, Diseño e Implementación en una FPGA de un Banco de Filtros Polifase para un Decodificador de MPEG 1 Layer 3", *Masters Thesis*, Universidad de Vigo, 2006.

[11] Altera Corporation, "Stratix Device Handbook", Altera, 2rd edition, 2003.

[12] Altera Corporation. "DSP Builder Reference Manual", Altera, 2rd edition, 2004.

[13] Byeong Gi Lee, "A new algorithm to compute the discrete cosine transform", *IEEE Transactions on acoustics, speech and signal processing*, 32(6), 1984.

[14] Domagoj Babic, "Discrete cosine transform algorithms for FPGA devices", *Masters Thesis,* Faculty of Electrical Engineering and Computing of Zagreb University, 2003.