IOSR Journal of Electrical and Electronics Engineering (IOSR-JEEE)
e-ISSN: 2278-1676,p-ISSN: 2320-3331, Volume 14, Issue 6 Ser. I (Nov. – Dec. 2019), PP 01-07
www.iosrjournals.org

Implementation of MP3 Decoder with Parallel Technique

Esther Ong Tze Shin[1], Zaini Abdul Halim[2]
[1](School of Electrical and Electronic Engineering, University Science Malaysia, Penang, Malaysia)
[2](School of Electrical and Electronic Engineering, University Science Malaysia)

Abstract: *This paper describes the mp3 decoding algorithm implemented in Matlab with parallel technique. It focuses on software-based, with different number of tasks. Three decoding techniques are used in the implementation including sequential decoding, two tasks parallel decoding and three tasks parallel decoding. The decoding algorithm is implemented based on 'spmdlabindex' coding provided in Matlab. The time taken to decode the mp3 files by using these three techniques are compared and their performance are evaluated. From the experimental results, it is found that two tasks parallel decoding has improved the decoding time by 14.18% with speedup of 1.165 while three tasks parallel decoding has 5.25% improvement in the computation time with speedup of 1.055. It can be concluded that by using spmdlabindex coding provided in Matlab, the time taken to decode the mp3 files can be improved and there exists some optimal number of tasks to perform the mp3 decoding process using parallel technique.*

Date of Submission: 24-10-2019 Date of Acceptance: 09-11-2019

I. Introduction

In mp3 decoding process, input file of the decoder is the mp3 bitstream. When decoder completes decoding for each frame, it will output the pulse code modulation (PCM) and the output will be written into waveform audio file format (wav). The wav format is important for its quality. The mp3 file has quality loss when it is compressed whereas the wav file is lossless and uncompressed. The mp3 will never sound better than the wav despite what kbps it is at because it is still lossy. Although the wav formats are much larger than the mp3 format, storage nowadays is no longer a problem, hence the quality can generally be afforded.

Generally, there are three approaches in the implementation of audio decoding system: software implementation, hardware implementation and hardware / software codesign implementation. The advantages of software audio decoder based on the personal computer (PC) include flexible design, short development cycle and little restriction of the platform. This is due to the latest PC hardware resources and the powerful software development tools. While for disadvantage there is high demand in the processor frequency of these systems, which cause higher power consumption [1].

On the other hand, the hardware audio decoder could have lower frequency and lower power consumption due to long development cycle. However, it is hard to modify or extend these decoders [2, 3]. The implementation of hardware and software co-design audio decoder which is able to meet requirements of flexibility, higher speed, extensible architecture and lower power consumption are proposed in the previous research [4, 5, 6, 7]. This method requires the audio decoder to be separated into software modules and hardware modules, thus the two audios standard's main function and computation complexity of each decoding module need to be analyzed.

With the development of computer science and technology, parallelism and pipelining are two key factors for improving the speed of audio decoding. However, most of the parallelism and pipelining are implemented on hardware. Since a minimum discussion is presented for implementation of mp3 decoder in software, a parallelism on mp3 decoder in software is thus be explored.

II. Material And Methods

Communication between two blocks requires a transfer of data from an output to an input. There needs to be a sender and receiver of the data for the complete communication process to take place. Sequential and parallel communications are both ways of transferring data over networks. Parallel connections have multiple route running in parallel and can transmit data on all the paths simultaneously. Sequential connections, on the other hand, uses a single path to transfer the data bits one at a time.

A. OVERVIEW OF SEQUENTIAL MP3 DECODING

The basic block diagram of the mp3 decoder is illustrated in Figure 1. It operates on the encoded bitstream to rebuild the PCM audio data. The architecture of the MPEG 1 Layer III bitstream are arranged into frames. Each frame consists of the essential information to regenerate the original PCM audio samples. The

design illustrates the decoder operating in a sequential manner. Data is transferred from the beginning until the end in serial and then restart again for a new frame.

The mp3 decoding process can be divided into nine blocks as shown in Figure 1 and each block executes independently from each other. Initially, the starting of a new frame is detected by the synchronization block. The frame is divided into two granules with each granule consists of 32 subband blocks. Each subband block contains 18 frequency lines which means each granule has 576 frequency lines. This operation is implemented by the Huffman and scalefactor decoding, requantization and reordering. Joint stereo decoding will be executed if frames are in stereo mode. After the alias reduction, IMDCT and synthesis filterbank, it will rebuild the PCM output. The music data is transformed from the frequency domain into the time domain.

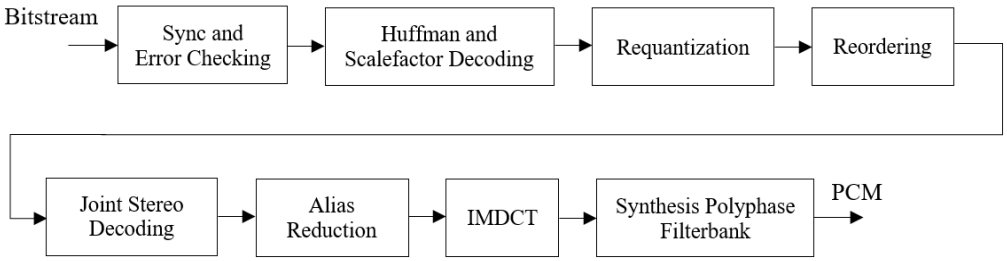


Figure 1 MP3 Decoding Process

B. OVERVIEW OF TWO TASKS PARALLEL MP3 DECODING

Figure 2 demonstrates the mp3 decoder blocks that are divided into two parts and assigned to two tasks which are labindex1 and labindex 2. Labindex 1 consists of 'Sync and Error Checking; Huffman and Scalefactor Decoding; Requantization and Reordering'. While labindex 2 contains 'Joint Stereo Decoding; Alias Reduction; IMDCT; Frequency Inversion and Synthesis Polyphase Filterbank'.

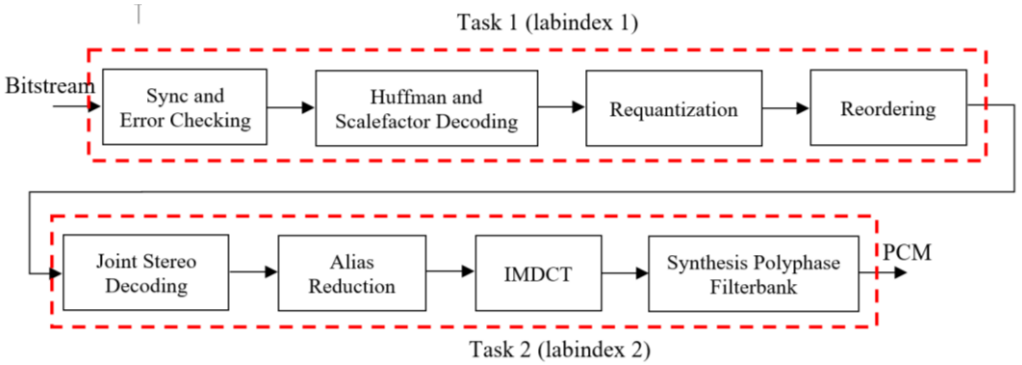


Figure 2 Block diagram of two tasks parallel mp3 decoding

Two tasks have been formed in the parallel computing program denoted labindex 1 and labindex 2. As shown in Table 1, the input is the bitstream and output is the PCM. It is a pipeline processing with two labindex. Initially, Data 1 is processed by labindex 1 and then passed to labindex 2. At the same time, Data 2 is read by labindex 1. The current data and the following data are being decoded simultaneously by both of the labindex. The process is repeated until the bitstream is completely decoded. Figure 3 demonstrate the two labindex pseudo code.

	labindex 1	labindex 2	
	Data 1		
Bitstream →	Data 2	Data 1	→ PCM
	Data 3	Data 2	
	Data 4	Data 3	

Table 1 Pipeline processing two labindex

Algorithm 1 Parallel processing with two labs

Input: Mp3 File
Output: PCM Waveform, Decoded Audio File

```
fid = fopen('sunshine.mp3', 'r', 'b');
y = uint8(fread(fid, 'ubit1')');

poolobj = parpool (2);
spmd
iflabindex == 1
while (next_bit ~= numel(y))
if (next_bit + 31 >numel(y))
labSend(2, []);
break
end
end
labSend(next_bit,2);
labSend(data,2);

else %labindex 2
while true
next_bit = labReceive(1);
data = labReceive(1);
if (next_bit + 31 >numel(y))
break
end
end
end
end %spmd end
delete(gcp);
end
```

Figure 3 Pseudo code for two tasks parallel mp3 decoding

C. OVERVIEW OF THREE TASKS PARALLEL MP3 DECODING

In three tasks parallel technique, the same process is implemented by adding one more task which is labindex 3. Figure 4 shows the mp3 decoder blocks are divided into three parts and distributed to three tasks (labindex) accordingly. Labindex 1 consists of 'Sync and Error Checking; Huffman and Scalefactor Decoding'. Labindex 2 includes 'Requantization and Reordering'. While labindex 3 contains 'Joint Stereo Decoding; Alias Reduction; IMDCT; Frequency Inversion and Synthesis Polyphase Filterbank'.

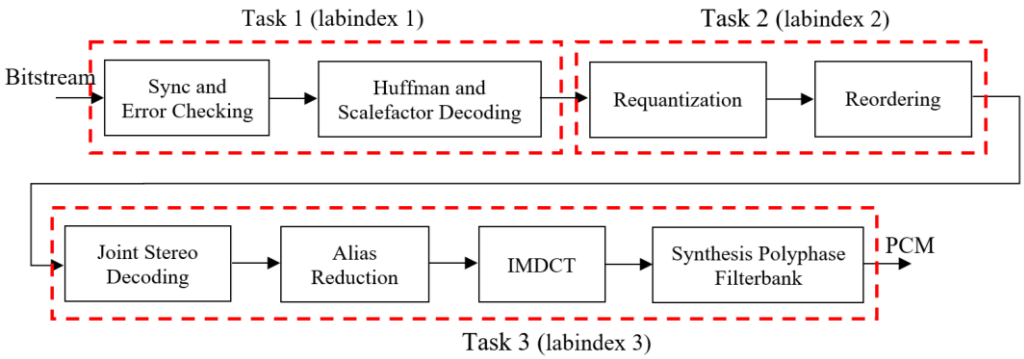


Figure 4 Block diagram of three tasks parallel mp3 decoding

According to Table 2, Data 1 is read by labindex 1 and then passed to labindex 2. Meanwhile, Data 2 is read by labindex 1. After that, Data 1 is passed to labindex 3 while Data 2 and Data 3 are read by labindex 2 and labindex 1 respectively. The process is repeated until the bitstream is completely decoded. Figure 5 illustrate the pseudo code for three labindex parallel computing.

Table 2 Pipeline processing with three labindex

	labindex 1	labindex 2	labindex 3	
Bitstream →	Data 1			
	Data 2	Data 1		→ PCM
	Data 3	Data 2	Data 1	
	Data 4	Data 3	Data 2	

Algorithm 2 Parallel processing with three labs

Input: Mp3 File
Output: PCM Waveform, Decoded Audio File

```
fid = fopen('sunshine.mp3', 'r', 'b');
y = uint8( fread(fid, 'ubit1')');
poolobj =parpool (3);
spmd
    if labindex == 1
        while (next_bit ~= numel(y))
            if (next_bit + 31 >numel(y))
labSend(2, []);
                break
            end
        end
labSend(next_bit,2);
labSend(data,2);
    elseif labindex == 2
        while true
next_bit = labReceive(1);
            data = labReceive(1);
                if (next_bit+31 >numel(y))
labSend(3, []);
                    break
                end
        end
labSend(next_bit,3);
labSend(data,3);
    else %labindex 3
        while true
next_bit = labReceive(2);
            if (next_bit + 31 >numel(y))
                break
            end
        end
    end %labindex1
end %spmd end
delete(gcp);
end
```

Figure 5 Pseudo code for three tasks parallel mp3 decoding

III. Results

One of the songs used in the decoding test is 'Reflection' by Lea Salonga. It is in mp3 file format which is the input of the audio decoding process. The input bitstream waveform is shown in Figure 6. The mp3 file size is 3.27MB. The mp3 file is decoded by three methods including sequential decoding, two tasks parallel decoding and three tasks parallel decoding. The PCM outputs are shown in Figure 7, Figure 8 and Figure 9 respectively. It is saved in wav file format and the file size is 12.3MB.

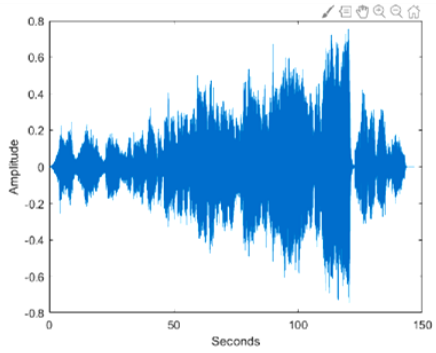


Figure 6 'Reflection.mp3' file waveform

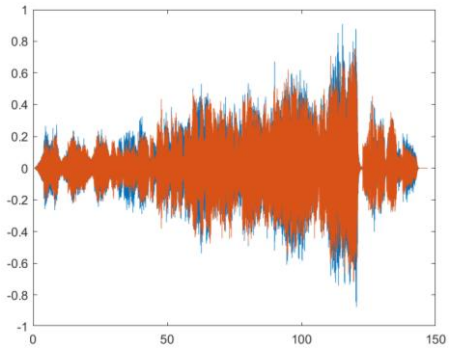


Figure 7 Sequential decoding PCM output for 'Reflection.wav'

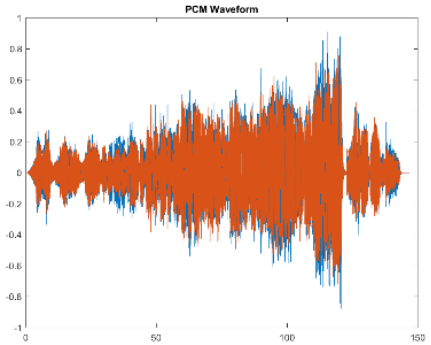


Figure 8 Two tasks parallel decoding PCM output for 'Reflection.wav'

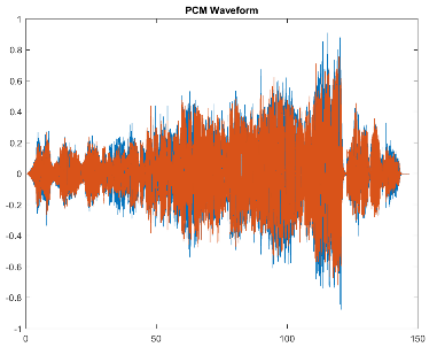


Figure 9 Three tasks parallel decoding PCM output for 'Reflection.wav'

Equality of two variances based on Bonett's method and T-test analysis have been employed to check the validity of the output pattern compared to the input pattern. Theresults of both tests can be observed from the p-value. It shows that the samples have equal variances and identical mean.

IV. Discussion

Table 3 shows the compilation of the test results. There are seven mp3 files with file size from 3.36 MB to 3.8 MB. As shown in the table, larger mp3 file takes longer decoding time in the simulation. The percentage of reduction in the decoding time is approximately 14.18% for two tasks parallel decoding with speedup 1.165 and 5.25% for three tasks parallel decoding with speedup 1.055.

Table 3 Test results for percentage of reduction in decoding time and speedup

No	Song Title (.mp3)	File Size (MB)	Percentage of Improvement (%)		Speedup	
			2tasks	3 tasks	2 tasks	3 tasks
1	Reflection	3.36	13.09	3.94	1.151	1.041
2	Fly me to the moon	3.54	13.85	5.15	1.161	1.054
3	Know Who You Are I Am	3.59	13.32	4.49	1.154	1.047
4	A whole new world	3.60	15.43	6.35	1.182	1.068
5	Everything at once	3.72	13.73	5.66	1.159	1.060
6	Beauty and the beast	3.77	14.36	5.51	1.168	1.058
7	Summer sunshine	3.89	15.46	5.63	1.183	1.060
		Mean Value	**14.18**	**5.25**	**1.165**	**1.055**

The speedup curve of two tasks parallel decoding and three tasks parallel decoding is shown in Figure 10. It is similar to a typical speedup curve [8]. As the number of task increases, speedup also increase until a saturation point is reached. Beyond this point, having more tasks will not further improve the performance due to the communication overhead.

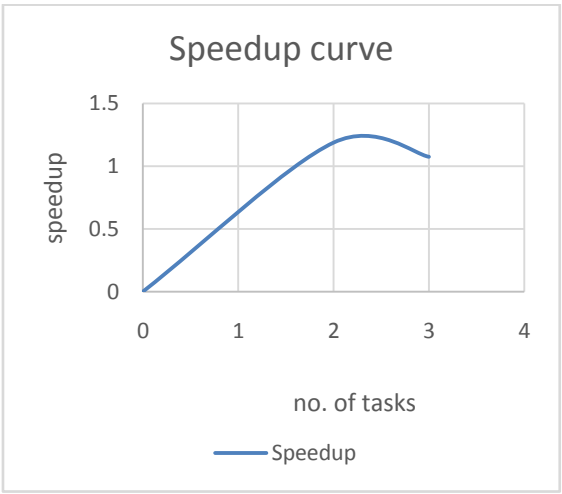


Figure 10: Speedup curve of two tasks parallel decoding and three tasks parallel decoding

Figure 11, Figure 12 and Figure 13 contains the Bonett's test and T-test analysis results between the input bitstream and PCM output in the mp3 decoder. The p-value obtained from the Bonett's test is 1. It means the null hypothesis is accepted, the variances between the two sets of data are the same. The two sets of data are then used in the T-test analysis.The p-value obtained in the T-test analysis is 1 which means the null hypothesis is accepted. The mean for both sets of data are the same. Hence, the validity between the input pattern and output pattern is verified.

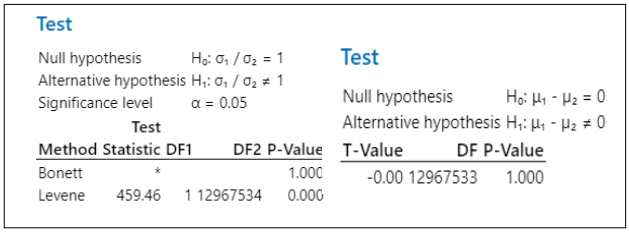
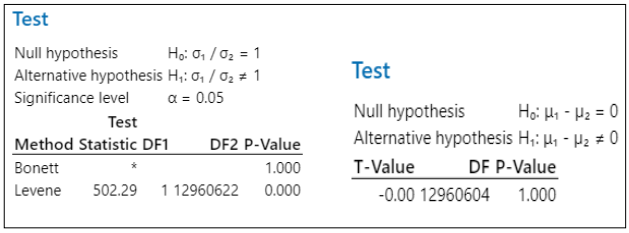


Figure 11: Bonett's test and T-test results between input bitstream and PCMoutput of sequential decoding



Test

Null hypothesis $H_0: \sigma_1 / \sigma_2 = 1$
Alternative hypothesis $H_1: \sigma_1 / \sigma_2 \neq 1$
Significance level $\alpha = 0.05$

Test

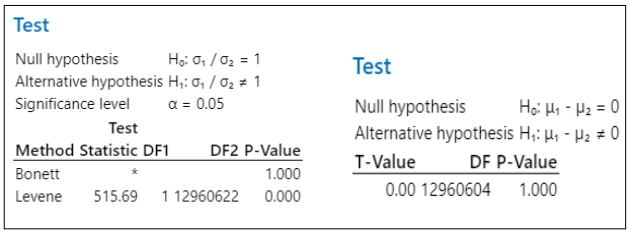
Method	Statistic	DF1	DF2	P-Value
Bonett	*			1.000
Levene	502.29	1	12960622	0.000

Test

Null hypothesis $H_0: \mu_1 - \mu_2 = 0$
Alternative hypothesis $H_1: \mu_1 - \mu_2 \neq 0$

T-Value	DF	P-Value
-0.00	12960604	1.000

Figure 12: Bonett's test and T-test results between input bitstream and PCMoutput of two tasks parallel decoding



Test

Null hypothesis $H_0: \sigma_1 / \sigma_2 = 1$
Alternative hypothesis $H_1: \sigma_1 / \sigma_2 \neq 1$
Significance level $\alpha = 0.05$

Test

Method	Statistic	DF1	DF2	P-Value
Bonett	*			1.000
Levene	515.69	1	12960622	0.000

Test

Null hypothesis $H_0: \mu_1 - \mu_2 = 0$
Alternative hypothesis $H_1: \mu_1 - \mu_2 \neq 0$

T-Value	DF	P-Value
0.00	12960604	1.000

Figure 13: Bonett's test and T-test results between input bitstream and PCM output of three tasks parallel decoding

V. Conclusion

Parallel processing can effectively reduce the decoding timerelative to the serial processing. The parallel mp3decoderis based on the multitask approach. The methods are compared through experimental results. The two tasks paralleldecoding has improved by 14.18% of the computation time while three tasks parallel decoding has only improved by 5.25% of the decoding time. Based on the performance evaluation, two tasks parallel decoding has the best improvement in the decoding time. This is because as the number of tasks increases, so does the overhead which comes from the communication during the parallel processing [9]. Hence, it shows that there exists some optimal number of tasks to perform the mp3 decoding process.

References

[1]. Uzelac, T. and Kovac, M. (1998) 'A Fast MPEG Audio Layer III Software Decoder', IEEE, 0-7803-4391- 3/98.
[2]. Lee, K. S. et. al. (1999) 'A VLSI Implementation of MPEG-2 AAC Decoders System', IEEE, 0-7803-5728-0/99.
[3]. Tsai, T. H. et. al. (2003) 'A Pure-ASIC Design Approach for MPEG-2 AAC Audio Decoder', IEEE, 0-7803-8185-8/03.
[4]. Yang, C. H. et. al. (2006) 'Software and Hardware co-design for MP3 Decoder', IEEE, 1-4244-0549-1/06.
[5]. Zhou, D. et. al. (2007) 'An SoC Based HW / SW Co-Design Architecture for Multi-Standard Audio Decoding', IEEE Asian Solid-State Circuits Conference, 1-4244-1360-5/07.
[6]. Tsai, T. H. et. al. (2008) 'A Platform-based SoC Design for a Multi-Standard Audio Decoder', IEEE, 978-1-4244-2064-3/08.
[7]. Zhang, T. et. al. (2010) 'MP3 / AAC Audio Decoder Implementation Based on Hardware and Software Co-design', 2010 3rd International Congress on Image and Signal Processing, IEEE, 978-1-4244-6516-3/10.
[8]. Singh, I. (2013) 'Review on Parallel and Distributed Computing', Scholars Journal of Engineering and Technology (SJET), ISSN 2321-435X, Scholars Academic and Scientific Publisher.
[9]. Kim, H., Mullen, J. and Kepner, J. (2005) 'Introduction to Parallel Programming and pMatlabv2.0', Mit Lincoln Laboratory, Lexington, MA.

Esther Ong Tze Shin. "Implementation of MP3 Decoder with Parallel Technique ". IOSR Journal of Research & Method in Education (IOSR-JRME) , vol. 9, no. 6, 2019, pp. 01-07.