International Journal of Electrical and Computer Engineering (IJECE)
Vol. 5, No. 6, December 2015, pp. 1336~1346
ISSN: 2088-8708

An SoC Architecture for Real-Time Noise Cancellation System Using Variable Speech PDF Method

Trio Adiono, Aditya F. Ardyanto, Nur Ahmadi, Idham Hafizh, Septian G. P. Putra
School of Electrical Engineering and Informatics, Institut Teknologi Bandung

Article Info

Article history:

Received Apr 14, 2015
Revised Sep 11, 2015
Accepted Sep 29, 2015

Keyword:

FPGA
LEON Processor
Noise Cancellation
System-on-Chip
Variable Speech PDF

ABSTRACT

This paper presents the architecture and implementation of system-on-chip (SoC) for realtime noise cancellation system which exploits variable speech probability density function (PDF) and maximum a posteriori (MAP) estimation rule as noise cancelling algorithm. The hardware software co-design approach is employed to achieve real-time performance while considering ease of implementation and design flexibility. The software module utilizes LEON SPARC-v8 and FPU co-prosessor as processing unit. The AMBA based Hanning Filter and FFT/IFFT are utilized as processing accelerator modules to increase system performance. The FFT/IFFT module employs custom Radix-2^2 Single Delay Feedback (R22SDF). In order to deliver high data transfer rate between buffer and hardware accelerators, the DMA controller is incorporated. The overall system implementation utilizes 18,500 logic elements and consumes 21.87 kB of memory. The system takes only 0.69 ms latency which is appropriate for real-time application. An FPGA Altera DE2-70 is used for prototyping with both algorithms and the noise cancellation function have been verified.

Corresponding Author:

Trio Adiono,
School of Electrical Engineering and Informatics,
Bandung Institute of Technology,
Jl. Ganesha No 10, Bandung 40132, Indonesia.
Email: tadiono@stei.itb.ac.id

1. INTRODUCTION

Noise is a common phenomenon that occurs in the audio signal, including the voice signal, which may distort or conceals any information contained in the signal. Therefore, the audio signal enhancement methods continue to be developed over time for a wide range of noise types and application conditions. Several methods exist on the literature to deal with this matter [1]. A spectral substraction (SS) method proposed by Boll [2] estimates the spectral of clear speech signal by subtracting noise spectra from noisy speech spectra. The subtraction process has to be carried out carefully to avoid any distortion. Too little substraction will keep the intefering noise, while too much substraction will lead to important information being removed [3]. Modification over SS method called as generalized spectral substraction (GSS) was proposed to improve noise cancellation performance while keeping the computational simplicity [4]. Another method utilizing a minimum mean-square error (MMSE) of the short-time spectral amplitude (STSA) was proposed by Ephraim-Malah [5] which provides an improved performance result. Lotter and Vary [6] proposed an efficient method based on maximum a posteriori (MAP) estimation and super-Gaussian statistical model, which does not require the use of tables for special complicated functions as Ephraim-Malah method does. A modified MAP estimation using a variable speech spectral distribution was also proposed, which approximate noise probability density function (PDF) more like a Delta function [7].

We perform simulation over several existing methods by using a reference sound sample which is added with Additive White Gaussian Noise over various SNR values. The performance of each method can

be seen by comparing the Segmental SNR (SegSNR) of the output signal. SegSNR can provide more objective parameter to compare the sound quality with the reference, because SegSNR performs calculation of average error on each segment [8]. The results of the simulation are shown in Figure 1. The waveform of input signal and processed signal is shown in Figure 2. The required processing time is also calculated to compare each algorithm performance as shown in Table 1.

The simulation results show that there is a tendency of better performance by using Variable Speech PDF using MAP method. This method eliminates almost all noises in non-speech segment, while maintaining good quality in speech segment as shown in Figure 2. However, Variable Speech PDF using MAP algorithm method requires the processing time about twice of SS or GSS method. Thus, there is trade-off between SegSNR performance and processing time. In this case, seeking to better performance and consistency, the Variable Speech PDF using MAP method will be used in proposed System-on-Chip (SoC) architecture.

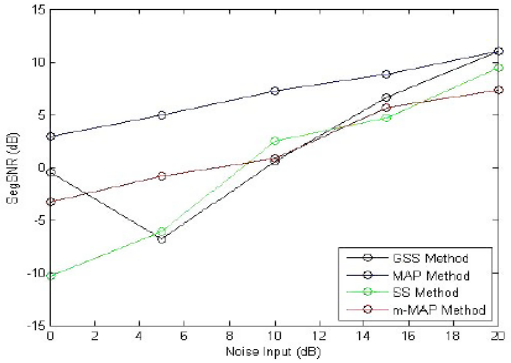


Figure 1. SegSNR comparison of various algorithms

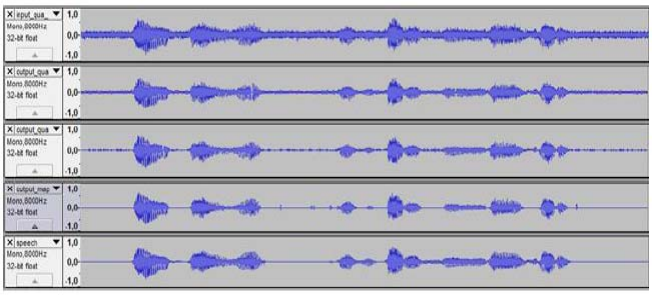


Figure 2. The output signal using various noise cancellation algorithms in simulation, noise SNR 5 dB (from top to bottom : noisy speech, output SS, output GSS, output MAP, output m-MAP, clear speech)

Table 1. Processing time in simulation of various algorithms

Method	Processing Time *(μs)*
SS	17.699
GSS	18.782
MAP	35.748
Modified MAP	32.191

Noise cancellation system implementation can be realized using hardware or software. Hardware implementation can result in an extremely fast system performance; however the required time for system design is quite long and not flexible for algorithm and tuning parameter update. In contrast to hardware, software design implementation can be accomplished quickly with high flexibility, but significantly slower system performances [9]. In order to take advantages of both methods and reduce their drawback, we incorporate hardware-software co-design in the proposed system design and implementation.

The rest of this paper is organized as follows. Section 2 explains the proposed algorithm which uses Variable Speech PDF with MAP Estimation. Section 3 describes the proposed SoC architecture which incorporates several modules such as input/output buffer, window filter, FFT/IFFT, AHB connection, and software design. The FPGA implementation and performance evaluation is given in Section 4. Finally, conclusions are drawn in Section 5.

2. PROPOSED NOISE CANCELLATION ALGORITHM

We proposed variable speech probability density function (PDF) with maximum a posteriori (MAP) estimation as noise cancelling algorithm [6]. In a speech segment, the PDF of spectral distribution approaches Rayleigh density, while in a non-speech segment, it approaches exponential function. The PDF parameters of speech spectral distribution are deduced from large amounts of observations. Based on PDF, noise cancellation can be achieved by using MAP estimation.

After obtaining noise power spectral density $\lambda_n(k)$ from estimated noise spectra, a priori $\xi_n(k)$ and a posteriori γ_n^2 should be calculated.

$$\begin{aligned}\xi_n(k) &= \frac{\lambda_s^2(k)}{\lambda_n^2(k)} \\ \gamma_n^2 &= \frac{|X_n(k)|^2}{\lambda_n^2(k)}\end{aligned} \tag{1}$$

The value of $\lambda_s^2(k)$ cannot be derived straight-forwardly, so a priori value must be estimated. Estimation is done using decision-directed method proposed by Epraim dan Malah [5]

$$\hat{\xi}_n(k) = \alpha\gamma_{n-1}(k)G^2\left(\widehat{\xi_{n-1}}(k), \gamma_{n-1}(k)\right) + (1-\alpha)\max[\gamma_n(k)-1, 0] \tag{2}$$

With α is forgetting factor between 0 and 1. MAP estimation is used to estimate reconstruction gain. Based on observations of many speech samples, speech model is characterized with spectral distribution PDF $p(S)$ and $p(\angle S)$

$$\begin{aligned}p(S) &= \frac{\mu^{v+1}S^v}{\Gamma(v+1)\sigma_S^{v+1}}\exp(-\mu\frac{S}{\sigma_S}) \\ p(\angle S) &= \frac{1}{2\pi}\end{aligned} \tag{3}$$

Joint MAP Estimator then gives gain G and phase function G which provides speech amplitude and phase spectra that can maximize conditional joint PDF $p(S, \angle S|X)$.

$$\begin{aligned}G &= u + \sqrt{u^2 + \tfrac{v}{2\gamma}}, \quad u = \frac{1}{2} - \frac{\mu}{4\sqrt{\gamma\xi}} \\ \angle S &= \angle X\end{aligned} \tag{4}$$

When μ = 3.1, σ_s^2 = 1, and v varies [5], peak amplitude of $p(S)$ is very small if $v \to 0$, which is suitable for non-speech segment to eliminate noise. On the other hand, peak amplitude of p(S) approaches large value when $v \to 2{:}0$, which is suitable limited parameter that is used to define v.

$$\begin{aligned}\widehat{v_n}(k) &= 0.1(1+10\log R(n))F_k \\ v_n(k) &= \begin{cases} 2.0 & , \ \widehat{v_n}(k) > 2.0 \\ \widehat{v_n}(k), & 0.1 \le \widehat{v_n}(k) \le 2.0 \\ 0.1 & , \ \widehat{v_n}(k) < 0.1 \end{cases}\end{aligned}$$

With $R(n)$ is ratio between input power spectra and noise power spectra, and $F(k)$ is weighting factor. $R(n)$ is obtained by:

$$R(n) = \beta R(n-1) + (1-\beta)\left(\frac{\sum_{k=0}^{M-1}|X_n(k)|^2}{\sum_{k=0}^{M-1}\lambda_n(k)}\right)$$

With β is forgetting factor between 0 and 1. Constants used in simulation comes from [5] with a little modification based on trial and error, which are $\alpha = 0.95$, $\beta = 0.50$, $\mu = 3.1$, and $F_k = 1.3$.

3. SYSTEM-ON-CHIP ARCHITECTURE

Based on Variable Speech PDF noise cancellation method, we can draw the proposed system block diagram as illustrated in Figure 3. Since the noise cancelation processing is carried out in frequency domain, the 64 points FFT/IFFT is used to convert audio signal into frequency domain. Before hand, the hanning filter is utilized to refine the FFT input signal. Non-speech audio signal is assumed to be in first 4 frames of the input signal ($\varkappa(n, l)$). The first 4 frames of the input signal will be processed by half overlapping, hanning filter, FFT, noise spectral estimator, and IFFT. After the first 4 frames, noise estimator will generate an averaged estimated noise spectra. The process that occurs in the next frame is half overlapping, The Hanning filter, FFT, Variable Speech PDF method as noise suppressor, and IFFT. In the proposed system, we use half overlap method, hence the first 32 samples of the output signal is combined with the last 32 samples from the previous frame. Therefore the output signal is generated and has correlation with the neighboring frames.

In order to achieve real-time performance while maintaining the flexibility of system implementation, the noise cancelation system is partitioned into hardware and software [10]. The partitioning is based on processing time of tasks involved in the system. The processing time is obtained by profiling of each task using system simulation models as shown in Table 2.

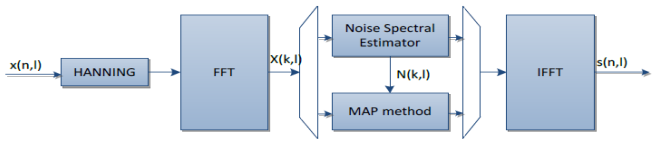
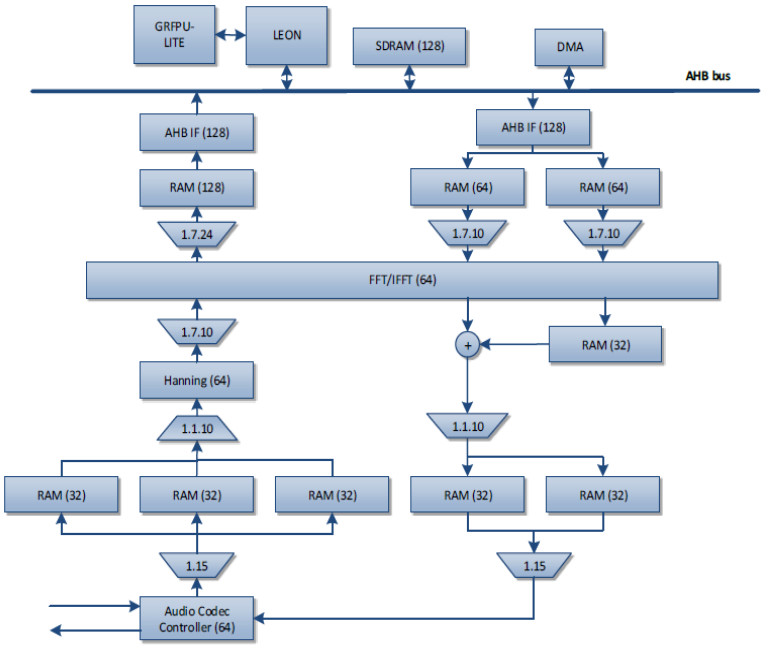


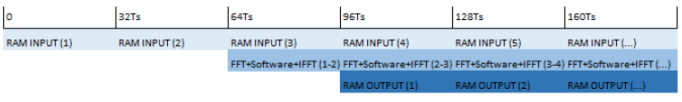
Figure 3. Proposed system srchitecture

Table 2. Profiling calculation result

Task	Processing Time (ms)	Percentage
Noise Sampling (D)	0.0175	0.0139%
Noise Cancellation (N)	0.0296	0.0235%
FFT + Hanning Filter	63.10	49.9847%
IFFT	63.09	49.9779%



(a) SoC architecture



(b) Timing diagram

Figure 4. The SoC architecture (a) and the timing diagram (b) of top-level system

Table 2 shows that the FFT/IFFT and Hanning Filter have the longest processing time compared to other tasks. The FFT/IFFT and Hanning Filter process are always executed due to conversion between time and frequency domain in noise cancellation algorithm. Moreover, the FFT/IFFT and Hanning Filter computation consist of simple and regular structures that are suitable for hardware implementation. On the contrary, other processing involves complex and irregular operations, such as divisions, exponentials, and logarithms that are suitable for software implementation. Based on those considerations, the FFT/IFFT and Hanning Filter are implemented in hardware as accelerator modules. On the other hand, the noise spectrum estimation and noise cancellation are implemented in software. In addition, the hardware software approach has the flexibility that allows future system performance improvement by introducing more advanced noise cancelation algorithms.

The proposed SoC architecture is designed using LEON SPARC-V8 32-bit processor as a host processor as illustrated in Figure 4(a). The host processor communicates with other modules through AMBA AHB/APB bus. The noise cancelation timing diagram is illustrated in Figure 4(b). It is shown that for 8 kHz sampling audio frequency, the available time to process the audio streaming data from Hanning Filter to IFFT is 4 ms.

3.1. The Input Buffer, Output Buffer, and Window Filter Module

In our proposed design, buffers are needed in input and output side, to bridge the gap between the low clock frequency of audio Codec module, and high clock frequency of the processor. There are two kinds of buffers need to be designed: a buffer to receive the input data from the Audio Codec (input buffer) and buffer to send the processed data to the Audio Codec (output buffer).

The FFT module requires 64 input data that is sent serially. Because the clock frequency of audio Codec is much slower than the processor, the input data needs to be buffered before it is sent into the FFT module. It utilizes three 32 words RAM with 16-bit width ports to implement dual clock buffer function. The main idea of this buffer is to continuously send 64 words of data while receiving 32 words of new data from audio Codec.

The computation result of IFFT process will be sent back to audio Codec. The output buffers are needed to combine half-overlapped data into serial output data. The output buffers consist of three 32 words RAM, two RAMs with different dual clock and one RAM (called RAM tail) as FIFO delay block to implement half overlap method.

Windowing Filter and the half overlap operations are used to alleviate discontinuities at the endpoints of each output block and increase relationship between frames [11]. In this proposed design, hanning window is used. The hanning window is specified as:

$$w(n) = \left\{ \begin{array}{ll} \frac{1}{2}\left(1 - \cos\frac{2\pi}{L-1}n\right), & 0 \le n \le L-1 \\ 0 \quad , & \text{otherwise} \end{array} \right\} \tag{7}$$

3.2. FFT/IFFT Module

The FFT/IFFT module is designed using a radix 2^2 with single-path delay feedback architecture. Radix 22 is an alternative form of radix-4. Radix-4 is chosen because of less nontrivial operation in contrast with radix-2, but the butterfly structure is still simple enough to be implemented in hardware [12]. Comparison of non-trivial operation in various FFT algorithms is shown in Table 3, while the butterfly structure is described in Figure 5(a) and 5(b).

Table 3. The addition and multiplication of real nontrivial operand in various DFT N-point computation

N	Real Multiplications				Real Additions			
	Radix-2	Radix-4	Radix-8	Split Radix	Radix-2	Radix-4	Radix-8	Split Radix
16	24	20		20	152	148		148
32	88			68	408			388
64	264	208	204	196	1032	976	972	964
128	712			516	2504			2308
256	1800	1392		1284	5896	5488		5380
512	4360		3204	3076	13566		12420	12292
1024	10248	7856		7172	30728	28336		27652

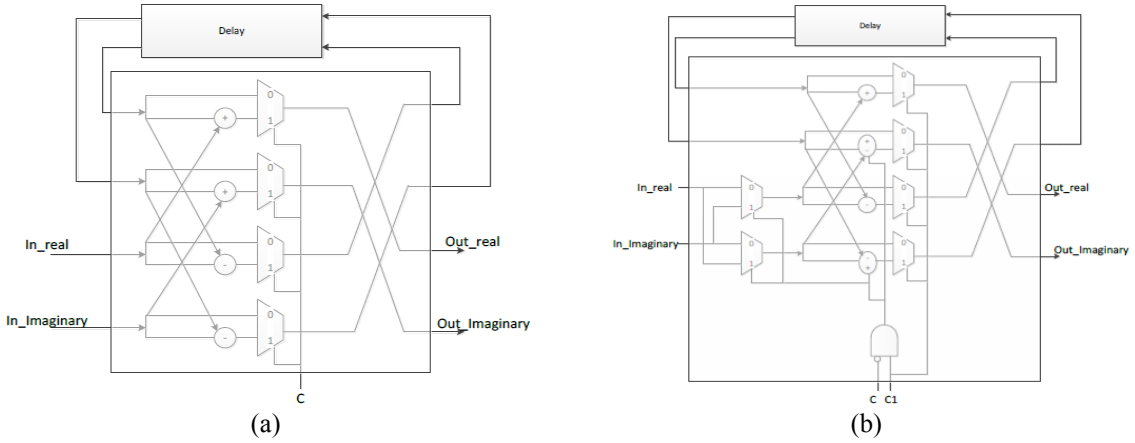


Figure 5. Butterfly structure of radix 22

The Single-path Delay Feedback (SDF) architecture is used in the proposed system because it has less multiplier, less memory bits, easier butterfly structure control, and no ordering in input signal. The comparison of R22SDF with other architecture is shown in Table 4. The R2^{2}SDF architecture can be described in Figure 6. Since the FFT processing is carried out in hardware and to optimize the utilized hardware resources, the data signal should be represented in fixed-point number with a limited bit width. The determination of bit width is performed by using simulation of bit precision FFT model. Based on the simulation results, the distortion of speech is difficult to be recognized when the utilized bit width in model is 10 bit fractional. Thus, we use the number format of Q1.10 (1 bit signed, 1 bit integer, and 10 bit fractional).

Table 4. The comparison of resource utilization in various FFT architectures

Architecture	# Multiplier	# Adder	Memory Size	Control Signal
R2MDC	$2(\log_4 N-1)$	$4\log_4 N$	$3N=2-2$	simple
R2SDF	$2(\log_4 N-1)$	$4\log_4 N$	$N-1$	simple
R4SDF	$\log_4 N-1$	$8\log_4 N$	$N-1$	medium
R4MDC	$3(\log_4 N-1)$	$8\log_4 N$	$5N=2-4$	simple
R4SDC	$\log_4 N-1$	$3\log_4 N$	$2N-2$	complex
R22SDF	$\log_4 N-1$	$4\log_4 N$	$N-1$	simple

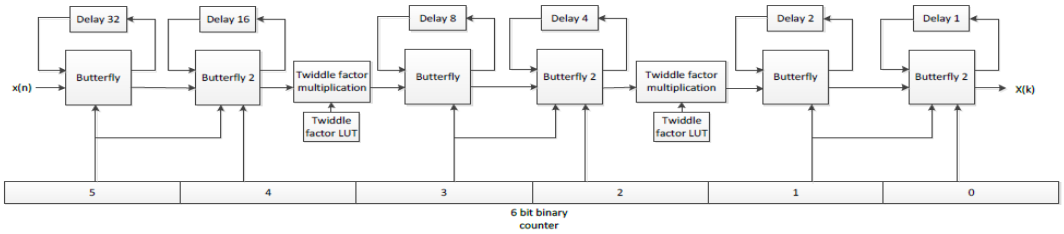


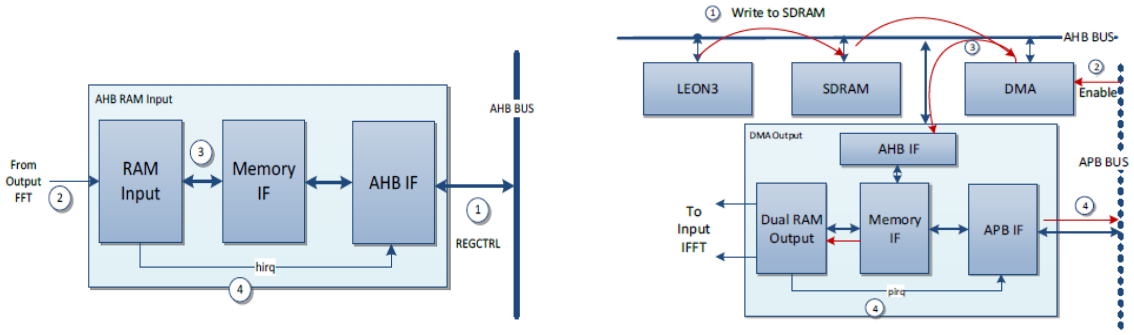
Figure 6. The R22SDF 64-point FFT architecture

Using the Q1.10 format, the obtained PSNR between input and output signal is 77.08 dB. The bit width of signal must be modified before entering FFT/IFFT module to prevent overflow when doing addition or multiplication in each stage of FFT. The broadening of bit width is adjusted with amount of stage, hence the number format used is Q7.10.

3.3. The AHB-connected Modules

The modules which are directly connected to the AHB bus are AHB RAM input and DMA output. The AHB RAM input has 128 words depth,32-bit wide, and an interface to the AHB bus as AHB slave. The RAM has such specification because in the frequency domain, the system has 128 data that consist of 64 real data and 64 imaginary data, whose width of the data follows the width of the AHB bus. The AHB RAM modules play a role to buffer FFT output data before accessed by the host processor. With the help of AHB

RAM module, the host processor can easily access the FFT output data by accessing a memory address mapped by the AHB interface. The host processors start accessing RAM immediately after receiving an interrupt signal from AHB RAM, as illustrated in Figure 7.



(a) Input process (b) Transfer process

Figure 7. The AHB buffer input process and DMA transfer process

In the next step, the host processor performs signal processing in the frequency domain. Once a processor finished the data processing, the processor will temporarily store the data in the SDRAM at the static address specified in the software. The special purpose DMA is designed in order to be able to perform high speed burst data transfer from SDRAM memory to the output buffer and from input buffer to the SDRAM as illustrated in Figure 7(b). The DMA module acts as an AHB Master that has privilege access to the SDRAM, without the help of the host processor.

The output buffer consists of dual 64-words RAM. The RAM buffer output is connected as APB Slave, which receives the read data and transmit DMA interrupt enable. The DMA process is carried out through a series of tasks. After the processor performs writing data to SDRAM, processors gives enable signal through APB, which triggers DMA to start working. After acquiring and locking access from the AHB arbiter, DMA starts to read data from SDRAM and the data is stored in the RAM output buffer. After reading all data in one frame, the interrupt signal is sent to the processor to indicate the data begins to be processed at the IFFT module. The buffer uses dual RAM because in the IFFT module, the real and imaginary inputs should be provided simultaneously. In addition, the enable signal is disabled to dismiss and release DMA access to SDRAM to wait for the next frame.

3.4. The Software Design

The frequency domain signal processing inside processor is done by using C-based software program. In the proposed SoC host ptocessor, the arithmetic operations can use both fixed point arithmetics and floating point unit GRFPU-Lite coprocessor. The utilization of co-processor depends on the computation complexity. The complex operation such as division and square root approximation are computed using floating point coprocessor. On the other hand, simple operation such as addition and multiplication is done in fixed point arithmatics.

When using fixed point, some approximation must be done to give similar result compared to floating point operation, for example in the logarithmic operation. In order to evaluate the logarithm of a fixed point number, x, we firstly divided it into mantissa, m, and exponent, e, that is similar to floating-point representation as:

$$x = m\,2^{e} \tag{8}$$

The logarithmic value of x can be expressed in terms of m and e as [13]:

$$y = \log_2 x = e + \log_2 m \tag{9}$$

The exponent is a signed integer and it is exactly the integer part of y. Since the exponent is to base 2, m is a fixed-point value in the range of [1:0; 2:0). Then, its logarithmic value is in the range of [0:0; 1:0),

i.e., the fractional part of v. The final logarithmic output is simply the sum of the integer part and the fractional part.

Since the logarithmic value ofmis always in the range of [0:0; 1:0), $\log_2 m$ can be expressed in binary format as:

$$\log_2 m = 0.b_{-1}b_{-2}\cdots b_{-i}\cdots \tag{10}$$

Where b_{-i} = f0; 1g for all integer i > 0. The weighting of each bit b_{-i} is 2^{-i}. Mathematically, the binary value can be expressed by the equation as follows:

$$\log_2 m = b_{-1}2^{-1} + b_{-2}2^{-2} + \cdots + b_{-i}2^{-i} + \cdots \tag{11}$$

Taking the anti-logarithm, we have:

$$m = 2^{b_{-1}2}2^{b_{-2}2^{-2}} \cdots 2^{b_{-i}2^{-i}} \cdots \tag{12}$$

The algorithm is to find out the bits, b_{-i}, by iterations starting from b_{-1}, then b_{-2} and so on.

Fixed point computation has less computing time in add, subtract, and multiply operation, while GRFPU Lite has less computing time in divide and square root operation. Therefore, in order to reduce processing time in software, the proposed design uses hybrid mode. The proposed design features fixed point method in add, subtract, multiply, and logarithmic operation, and GRFPU Lite in divide and square root operation.

4. IMPLEMENTATION AND PERFORMANCE EVALUATION

4.1. FPGA Implementation

In order to make sure that the proposed noise cancellation system can be implemented in FPGA Altera DE2-70, compilation and verification are performed. The system implementation on FPGA is carried out in several steps. The audio codec settings used in FPGA are 16-bit width ADC, 8 kHz sampling period. In the first test, we still only used bypassing with no algorithm used. This is done to test system capability whether it can be bypassed by an audio signal in a real-time manner with a certain sampling rate and observe whether significant distortion appears or not. The audio codec input is taken from the MIC channel, with bypass from LINE IN and MIC SIDETONE is deactivated. Tools arrangement used in the system implementation is shown in Figure 8.

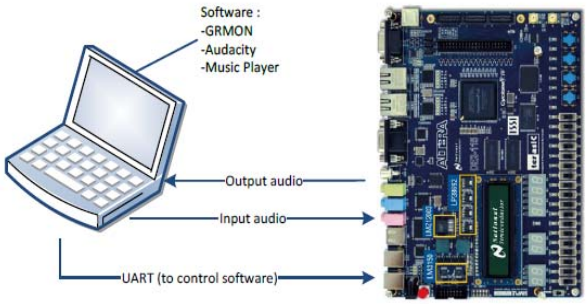


Figure 8. Tools arrangement

The output waveform from FPGA recorded on the Audacity software as well as the input waveform can be compared. The signal recorded from FPGA are bypassing audio codec signal, bypassing input-output buffer signal, bypassing FFT-IFFT signal, and bypassing LEON signal. The result is shown in Figure 9.

Qualitatively, there are still small observable distraction in speech signal. This occurs due to the use of noisy signal source or noisy audio receiver channel. The next test, we load the processor with Variable Speech PDF algorithm. The testing of noise canceling algorithm on FPGA is done in several ways, using full fixed point, full FPU utilization, and hybrid approach to compare processing unit performance and time using

the algorithm. The performance is measured by subjective listening test and using timestamps. The results are shown in Table 5.

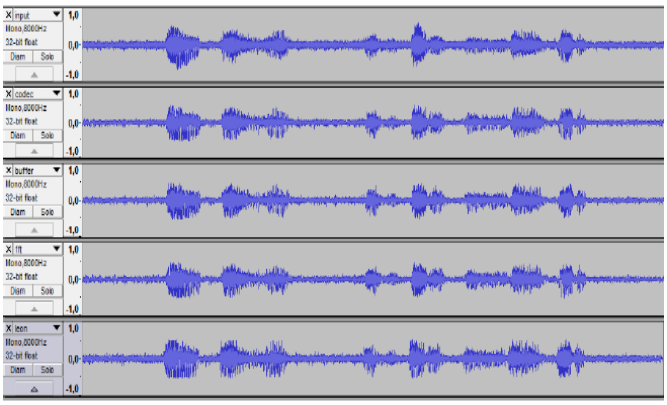


Figure 9. Comparison of input and output signal from FPGA without noise cancellation (from top to bottom: Bypass Codec, Bypass Buffer & Converter, Bypass FFT-IFFT, Bypass LEON, Clear Speech)

Table 5. Processing time using different computation techniques

Computation Technique	Processing Time (ms)	Relative Time Decrease
Fixed Point	3400	1.000
Floating Point	>4000	not feasible in real-time)
Hybrid	1600	2.125

Table 5 shows that the frequency domain processing is fastest when using both FPU and ALU function, while the differences in quality is unobservable by human ears. It is also shown that the Hybrid floating point and fixed point computation method significantly reduced the processing time while maintaining signal quality. The signal quality output of system is shown in Figure 10.

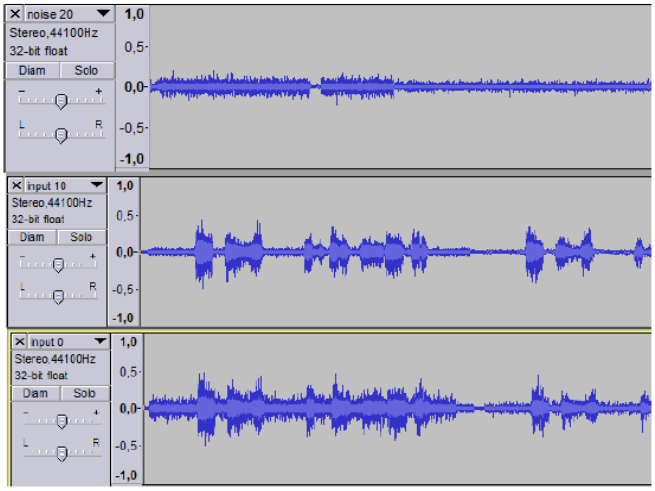


Figure 10. Comparison of input (Bypass Codec) and output signal (After MAP Method). From top to bottom: Noise Only 10 dB, Speech + Noise 10 dB, Speech + Noise 0 dB

4.2. Performance Evaluation

The measurement of performances, such as resource utilization of logic elements, memory bits, and multipliers, and maximum working frequency can be obtained in the report of design compilation. The

latency is obtained by dividing the amount of required cycles for processing of one frame using maximum frequency. The amount of cycles can be calculated by observing timing in RTL simulator. The overall system performance is shown in Table 6.

Table 6. Parameters of overall system performance

Components	Amount
Total Logic Elements (TLE)	18,500
Combinational Functions	17,050
Registers	5,152
Memory Bits	181,232
Embedded Multipliers	26
Fmax	59.48 MHz

Beside the overall system, we also analyze the performance of each blocks. The detailed resource utilization, such as LEs and memory bits, on each module is shown in Table 7.

Table 7. Resource utilization of each blok

Block	TLE	Memory Bits
LEON3 Processor	11,565 (62.5%)	145,664
FFT/IFFT	1,708 (9.23%)	4,464
Hanning Window	54 (0.29%)	0
AHB RAM Input	143 (0.77%)	4,096
DMA Output	238 (1.28%)	2,304
Input Buffers	98 (0.529%)	1,152
Output Buffers	94 (0.508%)	1,536
Audio Codec	442 (2.38%)	5,632

Other modules that utilize remaining resources are clock generator, AHB and APB controller, SDRAM controller, timer, Debug Support Unit, and UART controller for programming purpose. Proposed FFT/IFFT module performance is also compared to other FFT blocks from Quartus MegaCore function. The comparison is shown on Table 8. The Table 8 shows that our design utilizes significantly less LEs compared with Quartus MegaCore IP.

Table 8. Comparison of resource utilization on various FFT blocks

Blocks	LEs	Multipliers	Memory
Our custom FFT/IFFT	1,708	16	4,464
FFT from Quartus MegaCore function with 4 multipliers	4,373	24	9,984
FFT from Quartus MegaCore function with 3 multipliers	4,915	18	9,984

5. CONCLUSION

The proposed noise cancellation SoC system architecture using variable speech PDF method is designed by using hardware-software co-design approach. The time domain signal processing such as Hanning filter and FFT/IFFT are implemented in hardware to speed up the processing time, while the frequency-domain signal processing, such as MAP estimation, is implemented in software to increase the flexibility of future algorithm update. The design of SoC which based on the LEON processor and utilized FPU coprocessor, significantly improve system performance. The SoC uses custom AMBA based FFT/IFFT with R22SDF architecture. The system utilizes 18,500 logic elements, 181,232 memory bits, and can be run in real-time on 59.48 MHz system clock. The SoC has been successfully implemented and can run in real-time on FPGA Altera DE2-70 using speech signal input subjected with Gaussian noise.

REFERENCES

[1] JS Lim, AV Oppenheim. *Enhancement and bandwidth compression of noisy speech*. Proceedings of the IEEE. 1979; 67(12): 1586-1604.
[2] S Boll. Suppression of acoustic noise in speech using spectral subtraction. *IEEE Transactions on Acoustics, Speech and Signal Processing*. 1979; 27(2): 113–120.
[3] AR Fukane, SL Sahare. Different approaches of spectral subtraction method for enhancing the speech signal in noisy environments. *International Journal of Scientific & Engineering Research*. 1997; 2(5).

[4] BL Sim, YC Tong, JS Chang, CT Tan. A parametric formulation of the generalized spectral subtraction method. *IEEE Transactions on Speech and Audio Processing*. 1998; 6(4): 328–337.
[5] Y Ephraim, D Malah. Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator. *IEEE Transactions on Acoustics, Speech and Signal Processing*. 1984; 32(6): 1109–1121.
[6] T Lotter, P Vary. Speech enhancement by MAP spectral amplitude estimation using a super-Gaussian speech model. *EURASIP Journal on Applied Signal Processing*. 2005; 1110–1126.
[7] Y Tsukamoto, A Kawamura, Y Iiguni. Speech enhancement based on MAP estimation with a variable speech distribution. *International Symposium on Intelligent Signal Processing and Communications (ISPACS'06). IEEE*. 2006; 291–294.
[8] JH Hansen, BL Pellom. An effective quality evaluation protocol for speech enhancement algorithms. *In ICSLP*. Citeseer, 1998; 7: 2819–2822.
[9] G De Michell, RK Gupta. *Hardware/software co-design.* Proceedings of the IEEE. 1997; 85(3): 349–365.
[10] T Adiono, AA Purwita, R Haryadi, R Mareta, ER Priandana. *A hardware-software co-design for a real-time spectral subtraction based noise cancellation system.* International Symposium on Intelligent Signal Processing and Communications Systems (ISPACS'13). IEEE. 2013; 5–10.
[11] SV Vaseghi. Advanced digital signal processing and noise reduction. John Wiley & Sons. 2008.
[12] S He, M Torkelson. *A new approach to pipeline fft processor*. Proceedings of IPPS'96. The 10th International Parallel Processing Symposium. IEEE. 1996; 766–770.
[13] KE Larson. Floating point to logarithm converter. Dallas, Texas. 1994; 15.