FPGA Implementation of Spectral Subtraction for In-Car Speech Enhancement and Recognition

Jim Whittington, Kapeel Deo
Department of Electronic Engineering
LaTrobe University
Melbourne, Victoria, Australia
{j.whittington, kapeel.deo}@latrobe.edu.au

Tristan Kleinschmidt, Michael Mason
Speech and Audio Research Laboratory
Queensland University of Technology
Brisbane, Queensland, Australia
{t.kleinschmidt, m.mason}@qut.edu.au

***Abstract*—The use of speech recognition in noisy environments requires the use of speech enhancement algorithms in order to improve recognition performance. Deploying these enhancement techniques requires significant engineering to ensure algorithms are realisable in electronic hardware. This paper describes the design decisions and process to port the popular spectral subtraction algorithm to a Virtex-4 field-programmable gate array (FPGA) device. Resource analysis shows the final design uses only 13% of the total available FPGA resources. Waveforms and spectrograms presented support the validity of the proposed FPGA design.**

***Index Terms*—Field programmable gate arrays, speech enhancement, road vehicles.**

I. Introduction

A key challenge of deploying automatic speech recognition (ASR) in real-world environments is the requirement to perform well in the presence of high levels of noise. Most current speech recognition systems are trained for use in controlled scenarios (e.g. office environments or telephone-based systems). Since these recognisers are trained on "clean speech" they fail to produce satisfactory recognition performance under more adverse conditions such as in automotive environments.

There are a number of methods available for making speech recognition systems more robust. These include model compensation, the use of robust feature extraction and recognition algorithms, as well as speech enhancement. Enhancement techniques aim to remove (or at least reduce) the levels of noise present in the speech signals, allowing clean speech models to be utilised in the recognition stage. This is a popular approach as little-or-no prior knowledge of the operating environment is required for improvements in recognition accuracy.

Popular speech enhancement algorithms (e.g. filter-and-sum beamforming or spectral subtraction) have been designed primarily to improve intelligibility and/or quality of the speech signal without consideration of what effect that may have on other speech processing systems [1]. Despite being optimised using perception-based criteria, some of these techniques still produce improvements in ASR word accuracy rates, making them suitable for integration with speech recognisers.

The spectral subtraction method for the enhancement of noisy speech signals was originally proposed by Boll in 1979 [2]. Since this time there have been numerous reviews of the algorithm, although there are only limited examples where spectral subtraction has been specifically applied to noisy signals recorded in an automotive environment. Lockwood *et. al.* [3] and Wahab *et. al.* [4] have both concluded that spectral subtraction techniques can be successfully used to enhance speech signals in the presence of adverse automotive environment noise, although no particular hardware implementations were proposed.

Complete speech enhancement systems for use in automotive environments have been proposed by [5], [6]. Cheng *et. al.* [5] implement an adaptive beamformer with the majority of the processing performed on a PC, while Yu *et. al.* [6] propose the software implementation of a dual microphone least mean square (LMS) algorithm running on an Analog Devices Blackfin Digital Signal Processor (DSP). Neither of these provide a low cost, single chip solution, likely to be of greatest interest to automotive manufacturers.

Some examples of speech enhancement solutions on field-programmable gate arrays (FPGA) are provided in the literature [7], [8]. Yiu *et. al.* [7] have implemented a multi-microphone subband adaptive beamforming algorithm for speech enhancement in a high-end Virtex-4 FPGA. The high-end FPGA implementation was necessary to achieve real-time performance. The focus of this work was not to produce a stand-alone system, but rather a hardware accelerator for a large speech processing system centred on a CPU. This system showed "very similar" enhancement performance to a floating-point implementation and a very large improvement in processing performance. No automotive test data was included in this work, nor was any suggestion made that the system may be suitable for automotive applications.

Halupka *et. al.* [8] implemented a dual-microphone, phase-based time-frequency masking speech enhancement system on an Altera Stratix EP1S40 FPGA, and as a comparison, also on an off-the-shelf Freescale Semiconductor 16-bit DSP processor. An emphasis of this work was on producing a low power consumption solution potentially suitable for integration into hand-held devices. The FPGA and DSP implementations were tested against each other and an equivalent floating-point MATLAB implementation in terms of post-processing signal-to-noise ratio. The FPGA implementation produced similar real-time speech enhancement quality to the floating-point software, and used 3.5x less power than the non real-time

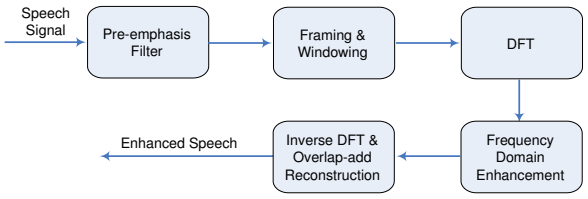

Fig. 1. *Block diagram of speech processing and enhancement in the frequency domain.*

DSP implementation. Test signals were artificially synthesised and no automotive data was tested nor was suitability for use in this environment discussed.

This paper focuses on the implementation of spectral subtraction optimised for speech recognition purposes into an FPGA design. Section II provides an overview of standard speech processing elements, followed by background on spectral subtraction theory and its application to speech recognition systems. Section III describes a fixed-point FPGA implementation of the algorithm. Experimental results using select samples from an in-car speech database verifying the FPGA design are presented in Section IV. Discussion of these results and possible improvements to the FPGA design are provided in Section V.

II. Speech Enhancement

A. Speech Processing Basics

Speech enhancement algorithms typically perform their primary processing in the frequency domain. The general approach to processing a speech signal in the frequency domain is presented in Fig. 1.

After a speech recording is acquired (via microphone), it is passed through a pre-emphasis filter which ensures a flatter spectrum of the signal by boosting the amplitude of the higher frequencies relative to lower frequencies. The signal is then decomposed into a series of frames of a set length (typically 32 ms – at 16 kHz this represents a frame length of 512 samples) and a Hamming window is applied to each frame. The frames are created using a sliding window with frame advances typically being 50% the length of the frame. This framing and windowing operation is followed by a Discrete Fourier Transform (DFT) which transforms the time-domain acoustic waveform to a discrete frequency representation.

The enhancement process operates on the frequency domain representation provided by the DFT, altering each frame's spectrum in an effort to improve the signal.

Following frequency domain enhancement, each frame is transformed back to the time domain using an inverse DFT, and adjacent frames are overlapped and added to resynthesise an enhanced time-domain signal. If the enhancement technique significantly alters the magnitude spectrum of the signal, a Hamming window can be re-applied prior to the overlap-add process in order to smooth discontinuities. The enhanced signal can then be used for playback or as an input to further speech processing such as automatic speech recognition.

A detailed description of the elements described in this section which comprise a standard speech processing system can be found in [9].

B. Spectral Subtraction

In a noisy environment, speech $s(n)$ is assumed to be corrupted by additive background noise $d(n)$ to produce corrupted speech $y(n)$ as follows:

$$y(n) = s(n) + d(n) \tag{1}$$

The signal is framed and the DFT is taken for each frame i as explained in Section II-A to produce the short-time frequency domain representation:

$$Y(i,\omega) = S(i,\omega) + D(i,\omega) \tag{2}$$

Generally in spectral subtraction algorithms, an estimate of the magnitude (or power) spectra of the noise signal $\hat{D}(\omega)$ is subtracted from the corresponding spectra of the noisy signal $Y(i,\omega)$ to give an estimate of the clean speech signal $\hat{S}(i,\omega)$:

$$|\hat{S}(i,\omega)|^\gamma = |Y(i,\omega)|^\gamma - |\hat{D}(\omega)|^\gamma \tag{3}$$

where γ is the exponent applied to the spectra, with $\gamma = 1$ providing magnitude spectral subtraction or $\gamma = 2$ for power spectral subtraction [10]. The noise estimate is determined by averaging frames deemed to be non-speech by a speech activity detector (or alternatively using a series of frames at the beginning of the utterance assuming the first 100-150 ms of a speech recording contains only noise). Using the latter method, around 8-10 frames are used to derive the noise magnitude spectrum estimate.

The phase component of the noisy speech signal is left unaltered and is kept for reconstruction into the time domain.

Should the subtraction in (3) give negative values (i.e. the noise estimate $|\hat{D}(\omega)|^\gamma$ is greater than the instantaneous signal $|Y(i,\omega)|^\gamma$) a flooring factor is introduced. This leads to the following formulation of spectral subtraction:

$$|\hat{S}(i,\omega)|^\gamma = \begin{cases} |Y(i,\omega)|^\gamma - |\hat{D}(\omega)|^\gamma & |\hat{D}(\omega)|^\gamma < |Y(i,\omega)|^\gamma \\ \beta|\hat{D}(\omega)|^\gamma & \text{otherwise} \end{cases} \tag{4}$$

where β is the noise floor factor, and $0 < \beta \ll 1$ [10]. Common values for this parameter range between 0.005 and 0.1 [10], [11].

Following enhancement of the magnitude spectrum, it is recombined with the original phase spectrum and reconstruction to the time domain is performed as described in Section II-A.

It should be noted here that in order to derive the rules for the two common spectra denoted in (3), two conflicting assumptions are made. If the clean speech and noise signals are assumed to be uncorrelated, the power spectral subtraction rule (i.e. $\gamma = 2$) results. Alternatively, if the two signals are assumed to be co-linear, the equation reduces to the magnitude spectral subtraction rule. In practice, neither of these assumptions is valid all the time.

C. Selection of Enhancement Parameters

Although common values for γ and β are those noted in the previous section, there is actually no limitation on the values that these parameters can take. The values of γ are typically used for their conceptual meanings as opposed to performance. Further, β is often chosen to optimise SNR given a particular value of γ. It was also previously established [12] that speech recognition performance differs greatly with various combinations of γ and β.

In order to reduce processing requirements in the FPGA implementation detailed in Section III, magnitude spectral subtraction (γ = 1) is chosen. Previous experiments in [12] showed that performing magnitude spectral subtraction provided better speech recognition accuracy than power spectral subtraction (if the β values were optimised for both values of γ).

Preliminary experiments were performed to determine the optimal value of β to use in the FPGA implementation. Using the first 5 experimental folds from the evaluation protocol for the AVICAR database [13], [14], values of β were varied in linear increments through the range [0, 1] with γ = 1. The results showed that a value of $\beta = 0.5$ provides best recognition performance and was therefore chosen for the FPGA implementation.

III. FPGA Implementation

The second part of this work involved adaptation of the spectral subtraction algorithm on to a realisable hardware platform suitable for use in an automotive environment. Xilinx, a leading FPGA vendor has developed the Xilinx Automotive (XA) product family specifically for automotive applications [15], [16]. With this in mind, Xilinx devices and development tools were chosen for this work since a clear pathway to a commercialisable platform is available. Cost is a key factor to eventual widespread adoption in the automotive field. The spectral subtraction algorithm relies on considerable DSP power. Thus, target devices must be cost effective while still providing relatively high performance DSP. With well over one million system gates, plus memory and XtremeDSP™ slices, Xilinx XA Spartan-3A DSP FPGAs fit this requirement well [16], [17], [18].

The Xilinx XA Spartan-3A DSP FPGA is a lower cost member of the Xilinx XtremeDSP™ Device portfolio, which also contains larger and higher performance Virtex-4 SX and Virtex-5 SXT devices. Due to similarities in their architecture, particularly the XtremeDSP™ slices, designs can be transported between XtremeDSP™ devices in a reasonably straight-forward manner. This feature enables designs to be initially developed in a high-end device and gradually reworked towards a lower-end device solution [19].

A. Design Process

Moving from an algorithmic description to a quality, cost effective FPGA solution is anything but trivial. The spectral subtraction algorithm was originally developed as MATLAB scripts using high precision, complex floating-point arithmetic. A one-to-one conversion to an equivalent FPGA implementation cannot be achieved as many of the complex operations cannot be directly or easily implemented in an FPGA with reasonable (ideally minimal) resource utilisation. Often options for the implementation of such operations involve use of approximations, either formulae-based or through the use of look up tables, both of which can introduce error to the system. Also, the precision of data in the FPGA implementation is limited to a fixed number of bits (fixed-point representation) which results in the addition of quantization noise to the system. This necessitates a multi-step process with considerable testing at each stage.

The first stage of this process involved identifying the main functions of the floating-point algorithm that would become key blocks in the hardware design (see Fig. 2). These were then converted to a fixed-point (data and operations) MATLAB implementation. MATLAB 7.3 with the fixed-point tool box were used in this process. The fixed-point implementation was then tested against the floating-point version.

Following validation of the fixed-point implementation, a Xilinx System Generator™ (XSG) model was developed. Xilinx System Generator™ is an FPGA hardware DSP development environment that sits above MATLAB and Simulink software packages [19]. The XSG package contains predefined blocks that can be readily compiled into a hardware description language (HDL) and subsequently synthesised for specific Xilinx FPGAs. Designers can also incorporate their own HDL descriptions into the model. An XSG model does not necessarily provide an optimised FPGA solution, but is certainly the fastest way to implement the complex functions of the algorithm, such as the fast Fourier transform (FFT) and obtaining the argument of complex numbers, which are examples of readily available functions. XSG provides a full simulation environment where signals can be analysed by simply dragging a 'scope' block on to the model and connecting the signal to it. Signal input/output can also be performed with the MATLAB workspace or file(s) as required. The development of the spectral subtraction XSG model was conducted block-by-block based on the fixed-point MATLAB implementation. Each block was tested after it was completed to ensure correct operation before the next block was developed.

Once the entire XSG model was complete, its performance was checked against the equivalent floating-point and fixed-point MATLAB models by comparing output waveforms. The verified hardware design was then synthesised, using Xilinx ISE 9.2 tools, and implemented on a high-end Xilinx Virtex-4 SX FPGA. For initial hardware development work the ML 402 development board (containing a Virtex-4 SX35 device) was used. While not unexpected in early versions of an FPGA design, the initial Virtex-4 implementation used more FPGA resources than desirable. This was attributed to the predefined XSG blocks that are often provided more as a rapid design development solution than a highly optimised one. The direct XSG model was then analysed and tested block-by-block to determine where resource inefficiencies lay and refined to use more appropriate resources. This was achieved through

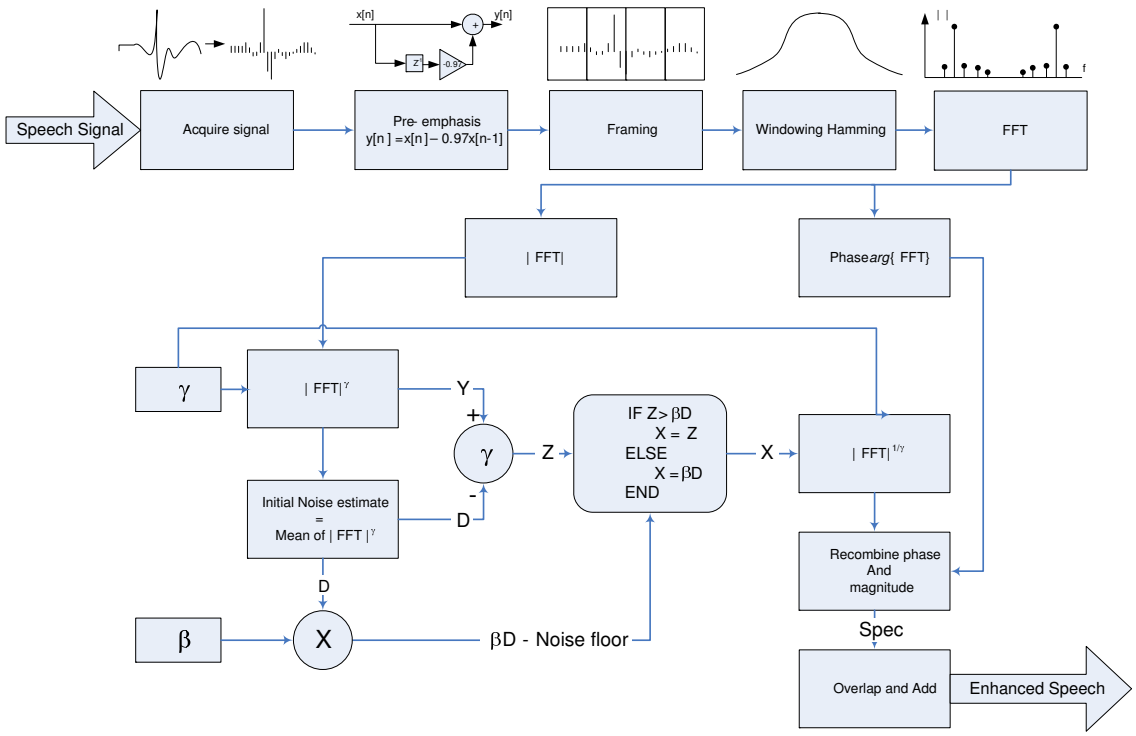

Fig. 2. *Block diagram of hardware implementation of spectral subtraction algorithm.*

specific coding of key sections of a number of blocks.

B. Hardware Perspective of Spectral Subtraction Algorithm

A block diagram of the spectral subtraction algorithm is provided in Fig. 2. Input signals consist of 16-bit speech waveforms sampled at 16 kHz. For this specific implementation, a frame size of 512 samples was used with 50% overlap.

The pre-emphasis filter is a basic design consisting of a delay, constant multiplication and sum. Framing and windowing can be readily achieved using an addressable shift register (buffer), predefined Hamming function, multiplication, and appropriate control logic.

For implementation of the DFT, an XSG forward and inverse FFT block can be used. This block provides both real and imaginary data outputs. As it is not used continuously, with appropriate control logic the same block can be used to perform the IFFT required after spectral subtraction. To generate frequency domain magnitude and phase data from the FFT block output, an XSG cordic arctan block is used.

At this point the algorithm calls for the magnitude data set to be raised to the power γ. This is potentially a very complex hardware operation, so as outlined in Section II-C, $\gamma = 1$ is used as it greatly simplifies the design yet has no effect on performance.

The initial noise magnitude estimate is calculated using a circular buffer, an addition block and division. Each incoming frequency bin data word is added to the previously accumulated value for that frequency. To obtain an average, the final sum must be divided by the number of frames used in the calculation. Restricting the number of frames to a power of two greatly simplifies the division operation. Therefore, this implementation uses the initial 8 frames of data to calculate the noise magnitude spectrum estimate.

The essence of the spectral subtraction technique occurs through subtracting the stored noise magnitude estimate $|\hat{D}(\omega)|$ from the subsequent frequency magnitude for each frame $|Y(i,\omega)|$ in the speech recording. The resulting frame Z is compared with a scaled version of the average noise magnitude by the factor β which is known as the noise floor. An element of the resultant frame Z is retained if it is greater than that of the noise floor, otherwise the noise floor element is retained. If a noise estimate is not available (i.e. the current frame is one of the first 8 frames of the signal), the incoming data frame is ignored. Furthermore, $\beta = 0.5$ was used in conjunction with $\gamma = 1$, as it provides best recognition performance [12]. Being a multiplication factor, $\beta = 0.5$ could be very simply implemented using a hardware shift operation as wiring between two registers. However, to maintain flexibility in the design during evaluation, an allowance for different β values was maintained by using a multiplication block for the noise flooring operation.

This point in the algorithm requires the enhanced frames X to be raised to the inverse power of γ. Once again potentially complex hardware is avoided by using $\gamma = 1$.

Before a time-domain frame can be generated, the new magnitude and previously retained phase frames must be combined and converted to real and imaginary cartesian coordinates required for input to the IFFT process (the XSG forward and inverse FFT block discussed earlier). This is performed with

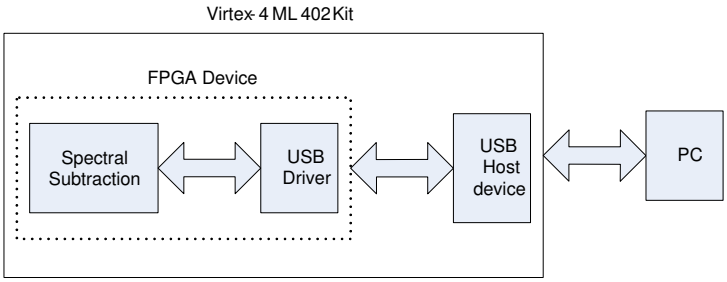

Fig. 3. *PC-USB test bench for FPGA implementation of spectral subtraction.*

an XSG cordic sin-cos block and two multipliers.

Finally, the resulting time-domain frames must be appropriately overlapped and added to produce the final reconstructed speech signal. This reverses the framing process performed at the start of the algorithm. This is implemented using an addressable shift register and an addition block.

IV. Verification of FPGA Design

A. Testing Xilinx Virtex-4 FPGA Implementation

To test the FPGA implementation, a method was required to accept standard speech waveforms from a PC – where they can also be passed through speech recognisers – and collect the corresponding output data and pass it back to the PC for storage and comparison. This requirement was met through the development of a USB test harness. The basic block diagram of this test harness is shown in Fig. 3.

The operation of the test harness is as follows.

1) A selected speech file is converted to a raw binary file suitable for sending to the FPGA.
2) The raw binary file is sent to the FPGA via the test bench PC application, through USB.
3) The USB driver/interface (an FPGA design) accepts the incoming binary data, buffers it and feeds it to the spectral subtraction implementation at the correct sampling rate.
4) At the same time output data is separately buffered by the USB driver/interface and sent to the PC via USB link.
5) The output from the FPGA is collected by the test bench PC application and stored as a raw binary file.
6) The file is then converted back to WAV format.
7) The resulting WAV file can then be played, plotted and compared to the equivalent XSG model simulation results or other versions of the spectral subtraction algorithm.

B. Test Results

Initial testing of the effectiveness of the Virtex-4 FPGA implementation was performed by comparing the output generated by the hardware implementation against equivalent outputs produced by the floating-point (MATLAB) implementation (of the spectral subtraction algorithm) and the Xilinx System Generator model. Test inputs comprised in-car speech signals from the AVICAR database [13]. As an example, a typical AVICAR speech sample under the 35mph with windows down noise condition is shown in Fig. 4(a). The corresponding spectral subtraction output from the floating-point algorithm is shown in Fig. 4(b), while the XSG model output is shown in Fig. 4(c). The Virtex-4 FPGA hardware output is shown in Fig. 4(d). Corresponding spectrogram plots for each of these waveforms are provided in Fig. 5.

Observing the waveforms and spectrograms it is clear that all three implementations of the spectral subtraction algorithm produce similar and noticeable reinforcement of the speech content within the signal. The background noise signal apparent in the original waveform (Fig. 4(a)) is noticeably reduced in the processed waveforms. The greater contrast present in the processed spectrograms reinforces this observation.

A synthesised waveform is required in order to analyse the sample-by-sample performance of the FPGA implementation. An amplitude-modulated chirp signal was corrupted by a Gaussian white noise signal to produce a peak signal-to-noise ratio of approximately 18 dB. This noisy reference signal is shown in Fig. 6(a). The corresponding spectral subtraction output from the floating-point implementation is shown in Fig. 6(b), whilst output from the Virtex-4 FPGA implementation is shown in Fig. 6(c). A sample-by-sample difference between the two implementations is provided in Fig. 6(d). The magnitude of the difference plot shows there is good correlation between the two implementations, with peak quantisation noise due to the fixed-point implementation of approximately -33 dB.

C. FPGA Resource Utilisation

The initial Xilinx Virtex-4 FPGA implementation – a direct synthesis of the XSG model – used approximately 29% of the total resources available. Table I shows a summary of the key FPGA resources used in this implementation. The following paragraph explains the terms used.

DSP48 slices are dedicated specialised hardware arithmetic blocks specifically tailored for the efficient implementation of complex mathematical and DSP functions. Digital Clock Manager (DCM) blocks are used to manage clock generation, distribution and to minimise clock skew. Block RAM (BRAM) are flexible blocks of RAM embedded in the FPGA fabric that can be utilised in a wide variety of ways, e.g. dual port or FIFO

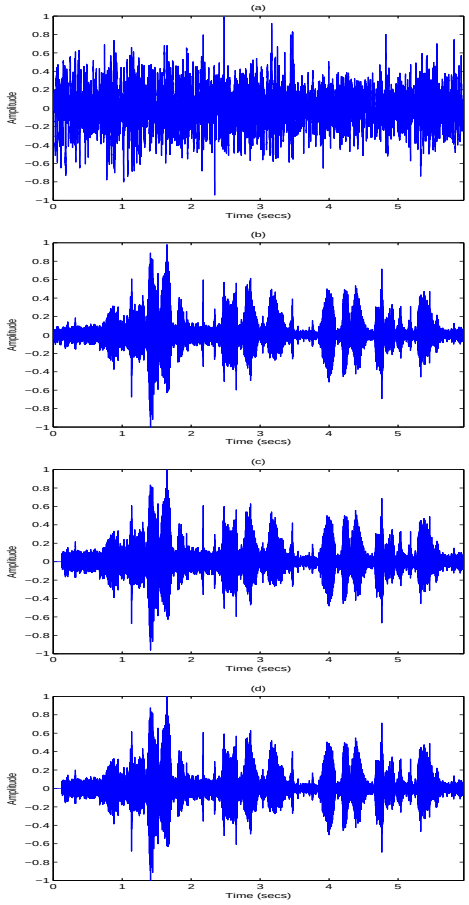

Fig. 4. *(a) Noisy speech signal from AVICAR database. Output of (b) floating-point MATLAB implementation of spectral subtraction, (c) Xilinx System Generator implementation, and (d) Virtex-4 FPGA implementation.*

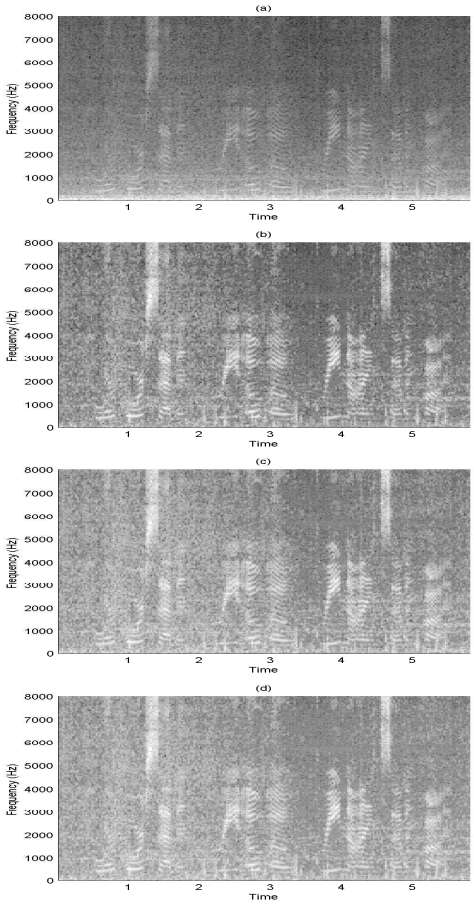

Fig. 5. *Spectrograms of (a) noisy speech signal from AVICAR database, output of (b) floating-point MATLAB implementation of spectral subtraction, (c) Xilinx System Generator implementation, and (d) fixed-point Virtex-4 FPGA implementation.*

TABLE I
INITIAL VIRTEX-4 FPGA RESOURCE USAGE SUMMARY.

Resource Type	Used	Available	Usage (%)
Slices	4505	15360	29.33
BRAM	8	192	4.17
DCM	1	8	12.50
DSP48	27	192	14.06

buffers, shift registers, large look-up tables (LUTs) etc. Slices represent the basic FPGA fabric, which consists of two 4-input LUTs, two flip-flops (FFs) plus interconnecting circuitry.

While this implementation fits comfortably into the Virtex-4 SX device, our aim is to produce a lower cost solution than a Virtex-4 implementation can provide. To identify potential resource savings, an analysis of block-by-block resource usage was conducted. A summary of these results are shown in Table II.

In general, the Xilinx Core blocks including the FFT, cordic arctan and sin-cos blocks use a significant amount of resources. This is not unexpected as these blocks perform complex mathematical functions. Although these blocks are provided by Xilinx as a core for rapid design development and as such may not be highly optimised, significant effort would be required to produce a more efficient tailor-made solution. A more appropriate first candidate for resource reduction are the Addressable Shift Registers. In the initial implementation these blocks used more than half the 4-input LUTs in the entire design. These circuit elements can be alternatively implemented in BRAM, a resource initially under-utilised. Finally, a number of additional slices, mostly FFs have been used to implement pipelines for improved speed performance.

After modification of the design to implement Addressable Shift Registers in BRAM, overall resource usage dropped by over half to approximately 13% of the total resources available. Table III shows a summary of the key FPGA resources used in the modified design.

A block-by-block analysis of resource usage is provided in Table IV. This table illustrates the significant savings made

TABLE II
BLOCK BASED FPGA RESOURCE USAGE – INITIAL DESIGN.

Block	Slices	Flip Flops	BRAM	LUTs	DSP48
FFT Core	700	900	5	1000	12
Rescaling Blocks (2)	316	0	0	630	0
Embedded 3 cycle Multiplier	81	130	0	55	4
Embedded 4 cycle Multiplier	94	156	0	55	0
Cordic a-tan	702	137	0	1300	0
Cordic sin-cos	500	85	0	1000	0
Addressable Shift Register (512x4)	2000	0	0	4000	0
Addressable Shift Register (256)	256	0	0	500	0

TABLE III
VIRTEX-4 FPGA RESOURCE USAGE SUMMARY – MODIFIED DESIGN.

Resource Type	Available	Used		Usage (%)	
		Initial	Modified	Initial	Modified
Slices	15360	4505	2040	29.33	13.28
BRAM	192	8	13	4.17	6.77
DCM	8	1	1	12.50	12.50
DSP48	192	27	27	14.06	14.06

through conversion of the Addressable Shift Registers to a BRAM implementation.

V. DISCUSSION

The initial Virtex-4 implementation used approximately 29% of the total FPGA resources available. While this was a reasonable result for a first cut of the design, a cost effective solution demands a more efficient implementation. A block-by-block analysis of resource usage quickly identified an obvious path to significant resource savings. By redesigning the Addressable Shift Registers (used in the Framing and Hamming Window, Noise Floor Calculation, IFFT buffers and Overlap & Add blocks) so that they are constructed from Block RAM rather than general LUT logic fabric, a very significant drop in overall resource usage to approximately 13% was achieved. If required, further efficiencies may be found through redesign of the complex mathematical function blocks, although these are likely to require significant design effort which – given current positive results – is probably only justified for a final product design. Removing some of the various pipelines used to improve processing speed and reduce latency could be used for a modest reduction in resource usage, although significant care would be needed to ensure appropriate performance is maintained.

Whilst the general structure of the enhanced waveforms from the floating-point and FPGA implementations are very similar, the peak quantisation difference between the floating-point and FPGA implementations was found to be approximately -33 dB, representing 5 bits. The true test of the effectiveness of the FPGA implementation for use in in-car speech recognition will be to evaluate FPGA processed waveforms using speech recognition engines. In the future, continuous speech recognition experiments using the phone numbers task of the AVICAR database [13] will be performed. The results of these experiments may lead to alteration of the precision of some hardware blocks in order to improve the accuracy of the speech recognition. The current overall Virtex-4 resource usage of roughly 13% will easily accommodate adjustments to the word widths used in various sections of the design. Also, β values can be easily changed in the current FPGA as a check of their impact on recognition performance.

Finally, the relatively modest resource usage of the Virtex-4 FPGA implementation bodes well for a reasonably straightforward transition of the design to a Spartan-3A DSP FPGA. Once complete, the performance of the Spartan-3A DSP implementation will be tested in the same manner as the Virtex-4 implementation described in this paper. Being identical to the Xilinx Automotive Spartan-3A DSP except in performance specification, an implementation on this platform will provide a good indication of the potential to produce a low cost realisation of spectral subtraction for in-car speech recognition and other applications.

VI. CONCLUSION

The common frequency-domain spectral subtraction enhancement algorithm has been described both theoretically and practically with respect to integration with any arbitrary speech recognition engine. It has also been analysed with respect to hardware implementation on a Virtex-4 FPGA.

Design paths have been described which have shown an initial resource usage of 29% reduced to approximately 13% of the total available FPGA resources. Waveforms and spectrograms of the spectral subtraction outputs from both the floating-point and FPGA implementations show the hardware design effectively realises the enhancement algorithm.

Future directions include porting the Virtex-4 design onto a smaller, more cost-effective Spartan-3A FPGA as well as performing continuous speech recognition for both designs using data collected in automotive environments.

ACKNOWLEDGMENT

This work was supported in part by the Australian Cooperative Research Centre for Advanced Automotive Technology (AutoCRC).

TABLE IV
BLOCK BASED FPGA RESOURCE USAGE – MODIFIED DESIGN.

Block	Slices	Flip Flops	BRAM	LUTs	DSP48
FFT Core	700	900	5	1000	12
Rescaling Blocks (2)	316	0	0	630	0
Embedded 3 cycle Multiplier	81	130	0	55	4
Embedded 4 cycle Multiplier	94	156	0	55	0
Cordic a-tan	702	137	0	1300	0
Cordic sin-cos	500	85	0	1000	0
Addressable Shift Register (512x4)	44	38	4	74	0
Addressable Shift Register (256)	6	5	1	10	0

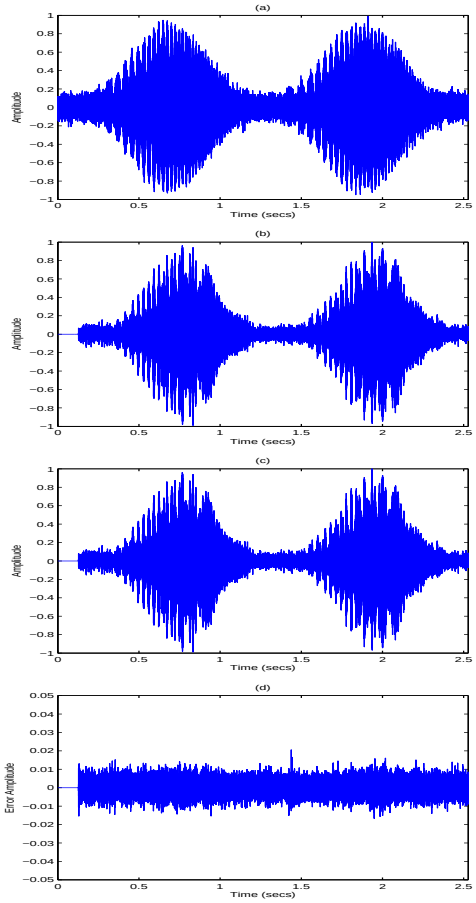

Fig. 6. *(a) Noise modulated chirp signal. Output of (b) floating-point MATLAB implementation of spectral subtraction, (c) fixed-point Virtex-4 FPGA implementation, and (d) the sample-by-sample difference between the floating-point and fixed-point implementations.*

REFERENCES

[1] M. Seltzer, B. Raj, and R. Stern, "Likelihood-maximizing beamforming for robust hands-free speech recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 12, no. 5, pp. 489–498, 2004.

[2] S. Boll, "Suppression of acoustic noise in speech using spectral subtraction," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 27, no. 2, pp. 113–120, 1979.

[3] P. Lockwood, J. Boudy, and M. Blanchet, "Non-linear spectral subtraction (NSS) and hidden Markov models for robust speech recognition in car noise environments," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, 1992, pp. 265–268.

[4] A. Wahab, E. C. Tan, and H. Abut, "CMAC spectral subtraction for speech enhancement," in *Sixth International Symposium on Signal Processing and its Applications*, vol. 2, 2001, pp. 707–710.

[5] C.-C. Cheng, W.-H. Liu, C.-H. Yang, and J.-S. Hu, "A robust speech enhancement system for vehicular applications using H_∞ adaptive filtering," in *IEEE International Conference on Systems, Man and Cybernetics*, vol. 3, 2006, pp. 2541–2546.

[6] S. Yu, "Hybrid speech enhancement and speech recognition system for car telematics platform for hands-free control GPS navigator and voice dialer for handphone," ASEAN Virtual Instrument Applications Contest Submission, 2006.

[7] K.-F. C. Yiu, Y. Lu, X. Shi, and W. Luk, "FPGA acceleration of a subband beamforming algorithm for speech enhancement," in *Congress on Image and Signal Processing*, vol. 5, 2008, pp. 742–746.

[8] D. Halupka, A. Rabi, P. Aarabi, and A. Sheikholeslami, "Low-power dual-microphone speech enhancement using field programmable gate arrays," *IEEE Transactions on Signal Processing*, vol. 55, no. 7, pp. 3526–3535, 2007.

[9] X. Huang, A. Acero, and H.-W. Hon, *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. Prentice Hall PTR, 2001.

[10] M. Berouti, R. Schwartz, and J. Makhoul, "Enhancement of speech corrupted by acoustic noise," in *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, 1979, pp. 208–211.

[11] R. Martin, "Spectral subtraction based on minimum statistics," in *EUSIPCO*, Edinburgh, 1994, pp. 1182–1185.

[12] T. Kleinschmidt, S. Sridharan, and M. Mason, "A modified LIMA framework for spectral subtraction applied to in-car speech recognition," in *1st International Conference on Signal Processing and Communication Systems*, Gold Coast, Australia, 2007, pp. 335–338.

[13] B. Lee, M. Hasegawa-Johnson, C. Goudeseune, S. Kamdar, S. Borys, M. Liu, and T. Huang, "AVICAR: Audio-visual speech corpus in a car environment," in *INTERSPEECH*, Jeju Island, Korea, 2004, pp. 2489–2492.

[14] T. Kleinschmidt, D. Dean, S. Sridharan, and M. Mason, "A continuous speech recognition protocol for the AVICAR database," in *1st International Conference on Signal Processing and Communication Systems*, Gold Coast, Australia, 2007, pp. 339–344.

[15] K. Kitagawa, "At the heart of consumer and automotive innovation," *XCell Journal*, no. 63, pp. 12–13, 2007.

[16] Xilinc Inc., "Xilinx Automotive - flexible solutions beyond silicon," 2007.

[17] D. Bagni and P. Zoratti, "Block matching for automotive applications on Spartan-3A DSP devices," *XCell Journal*, no. 63, pp. 16–19, 2007.

[18] V. Sardana, "Slash your total cost by up to 50% with Spartan-3 generation FPGAs," Xilinc Inc., Tech. Rep., 2008.

[19] Xilinx Inc., "XtremeDSP^{tm} solutions selection guide," 2008.