34th Annual International Conference of the IEEE EMBS
San Diego, California USA, 28 August - 1 September, 2012

Real-Time Source Separation on a Field Programmable Gate Array Platform

Valerie S. Hanson[1] and Kofi M. Odame[1]

Abstract— **In this paper, we present a real-time implementation of the ideal binary-mask algorithm, which is a promising approach for enhancing speech intelligibility. Our implementation is hardware efficient, making it suitable for embedded biomedical devices such as hearing aids and cochlear implants. We tested our algorithm implementation on an FPGA platform, and produced results that verify that it effectively performs source separation with 25 μs latency.**

I. INTRODUCTION

Hearing impaired individuals have more difficulty interpreting speech in the presence of noise than do individuals with normal hearing, even when corrective hearing devices are used. We are developing an algorithm for hearing prosthetic devices that improves speech intelligibility in noisy environments. The algorithm is based on beamforming, dynamic user feedback control and source separation, as shown in Fig. 1.

Our source separation scheme is a modified version of the ideal binary mask (IBM) algorithm that was introduced by Boldt [1]. With IBM-enhanced speech, normal hearing listeners have reported speech intelligibility scores of nearly 100%, in the presence of 4 overlapping speakers [2]. The IBM provides near 100% intelligibility even when the signal-to-noise ratio of the target speech (that is, the speech to which the listener is attending) to background noise is as low as -10 dB [3].

To date, the IBM algorithm has mostly been performed either offline or in simulation [1], [2]. Saruwatari et al. described an online IBM algorithm [4], but their implementation has a 3 s latency, a delay that is perceptible and objectionable to human listeners [5]. For the algorithm to be relevant in a practical hearing prosthetic device, it must be implementable in an embedded system and it must execute in real time, with sub-millisecond latency.

In this paper, we will describe our real-time implementation of the IBM algorithm, realized in hardware on a field programmable gate array (FPGA). The low latency, low power and low hardware utilization of our implementation make it practical for use in hearing aids and cochlear implants.

II. OVERVIEW OF IDEAL BINARY MASKING

A. Auditory Masking

Speech is very sparsely distributed in the time-frequency domain [6]. Hence, it is hardly ever the case that a competing noise (e.g. background clatter) would overlap with – or overwhelm – a time-frequency region that contains speech.

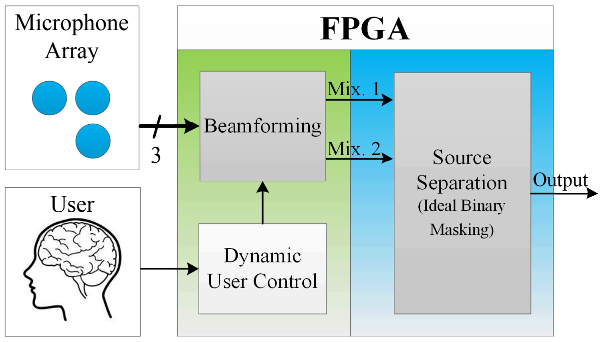

Fig. 1: Top level diagram of speech intelligibility-enhancing system. The FPGA must analyze user control signals, perform beamforming and output a source-separated signal in real time. This requires efficient use of hardware and careful timing of instructions.

However, competing sounds that occur close to a speech-containing time-frequency region can render the speech imperceptible, via the psychoacoustic phenomenon of auditory masking [7]. So, due to auditory masking, there is a loss of speech intelligibility even in moderate SNR conditions. The auditory masking effect – and loss of speech intelligibility – is much more pronounced in individuals with hearing impairment than in those with normal hearing [8].

B. Improving Speech Intelligibility

The idea of the IBM algorithm is to remove those portions of the time-frequency domain that contain competing sounds, as they could potentially cause auditory masking of the target speech [9].

The first step of the IBM algorithm is to perform a spectral analysis to map the input signal into the time-frequency domain. This is followed by a classification step, during which the target speech regions of the input signal are distinguished from competing noise regions. Finally, the algorithm removes the noise regions and retains the target speech regions. Fig. 3 illustrates the spectral analysis and classification phases of the IBM algorithm.

Forming the ideal binary mask requires access to the unmixed target and noise signals. As these signals are unobtainable in practice, they instead have to be estimated via beamforming as described in [1]. The estimated signals are mixtures of target and noise signals, with Mixture 1 consisting mostly of the target, and Mixture 2 consisting mostly of the noise. The mixtures are input to the IBM algorithm, as illustrated in Fig. 2.

[1]Authors with Thayer School of Engineering at Dartmouth College, Hanover, NH 03755 USA. (e-mail: valerie.s.hanson@gmail.com)

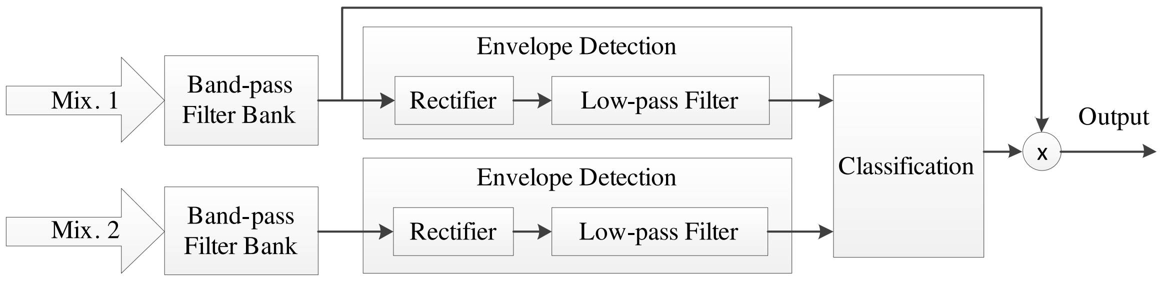

Fig. 2: Block diagram of the IBM algorithm data path.

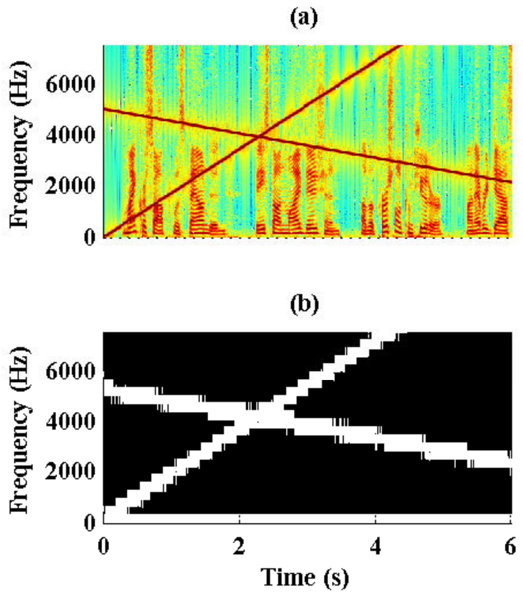

Fig. 3: (a) The spectral analysis phase and (b) the classification phase of the IBM algorithm for a speech sample masked by 2 chirp signals. The white pixels have been classified as noise and black pixels have been classified as target speech.

In the IBM algorithm, the sound mixtures are each mapped onto the time-frequency domain by performing spectral analysis with a bank of bandpass filters, followed by energy extraction with an envelope detector (see Fig. 2). Each time-frequency region is classified as either containing target sound or noise, depending on the relative amount of energy that is contributed by Mixture 1 and Mixture 2. The binary mask is generated by the classifier, and it is applied to Mixture 1 to separate target speech from noise [1].

III. IBM CHALLENGES

The performance of the IBM algorithm improves with increasing spectral resolution. For instance, speech intelligibility improves from 5% to over 65% when the number of spectral analysis frequency bands is increased from 6 to 32 for -5 dB SNR [10]. Also, additional studies have shown near 100% performance with -3 dB SNR for 128-count filter banks [2]. This requirement for high spectral resolution places a number of constraints on implementing the IBM in a real-time embedded system.

High spectral resolution requires a large number of narrow bandwidth, high-order bandpass filters. The IBM implementation must therefore have the capacity to store all of the associated filter coefficients in memory. Further, these coefficients must be stored at high resolution, which is necessary to prevent numerical instability in the narrow bandwidth filters. These considerations will all contribute to straining the system's memory resources.

During spectral analysis, the individual bandpass filtering operations can be performed sequentially, or – since there are no data dependencies between frequency bands – in parallel. In the following subsections, we outline the constraints faced with each of these approaches.

A. Latency Constraints

The IBM algorithm executes roughly 32,000 instructions for every input sample, assuming the spectral analysis is performed with a 128-channel filter bank.

With a sampling rate of 32 kSamples/second, and assuming each instruction can complete in a single clock cycle, a purely sequential implementation of the IBM algorithm would require at least a 1 GHz clock rate. A digital signal processor that ran at this clock rate would be incompatible with the low power requirements of hearing prosthetic devices [11], [12].

In theory, the entire chain of sequentially-executed instructions could be pipelined, which would allow the processor to run at a clock rate equal to the sampling rate. However, there would still be a latency of a few milliseconds, which is long enough to be perceptible and undesirable to users [5].

B. Hardware Constraints with Fully Parallel Implementation

For a completely parallelized implementation of the IBM algorithm, all 128 band-pass filters would be processed simultaneously for both mixture signals. Also, each individual filter would itself have an internally parallelized implementation, as many of its operations are independent of each other. With such aggressive parallelization, it is possible to execute the entire IBM algorithm in 24 clock cycles.

Limitations arise because parallelization requires a dedicated set of hardware for each component that is run simultaneously. For instance, to achieve a fully parallelized implementation of the 128-channel filter bank as described, 3072 units of 18-bit hardware multipliers are required. This design is unrealizable given that most FPGAs have the capacity for less than 100 18-bit multipliers [13].

IV. IBM Hardware Implementation

The computational complexity of the IBM algorithm forces us to make careful choices in our filter design and memory allocation. Also, it requires a data path that involves sequential, parallel and nested pipelined processing in order to fit the algorithm in a practical embedded system design.

The parallel and pipelined aspects of our novel data path are easiest to implement on an FPGA platform. Also, the level of parallelism in our data path can be easily expanded or contracted as needed to fit the available hardware resources of a given FPGA. Finally, FPGA implementation offers lower energy consumption than that of a DSP, since most of the computation is routed directly into hardware instead of requiring processing resources to shuffle instructions around. In this section, we describe the details of our FPGA implementation of the IBM algorithm.

A. Filter Structure

The filter structure implementation was chosen to optimize memory and hardware allocation. Filters are designed as 8th order IIR band-pass filters, formed by cascading two 4th order filters, which are formed by cascading two second order sections (SOS) in the direct form 2 implementation as illustrated in Fig. 4. Justification for this design is as follows:

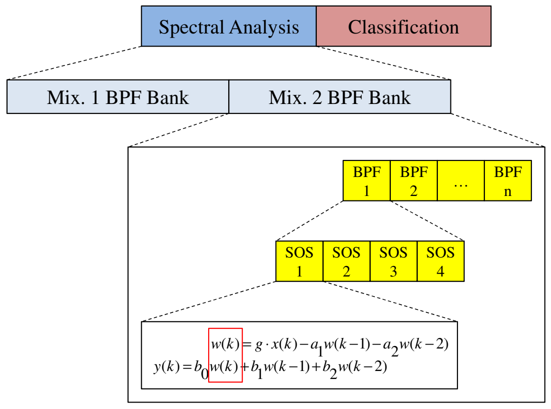

Fig. 4: Nested filter levels contained in the spectral analysis stage.

- An IIR design is chosen over FIR because IIR implementation meets the small pass-band and sharp roll-off constraints using far fewer history elements.
- An SOS implementation is chosen over the standard difference equation implementation in order to reduce the required word-length needed to maintain filter stability.
- Direct form 2 (DF2) implementation is chosen to reduce the total number of history elements required for processing. This implementation stores only intermediate histories as opposed to storing both input and output histories, halving the total number of histories required.
- Cascading two 4th order filters to form the 8th order filter is done for two reasons -Ű this halves the number of coefficients needed, and places the cutoff at -6dB instead of -3dB. This feature is important, because it ensures that when summing the signals from adjacent bands the magnitude at the cutoff frequency sums to 1.

B. Memory Allocation Design

This design optimizes the use of FPGA block RAM (BRAM) components through address partitioning and optimization of primitive concatenation. Hardware optimization comes as a trade-off to memory access time. Fastest data access is achieved by assigning all data for a given filter to one address. This approach can be realized by concatenating all data elements for a single filter into one single element. While this design is time efficient, enabling simultaneous access of all data, it prohibits efficient hardware allocation. When configuring BRAM primitives there is some flexibility in the address width to data width trade-off, however address width is always much larger than data width for all configurations. Therefore, implementation of large word-length data is achieved by concatenating multiple BRAM components until the desired data width is formed for a single address. When this design proved unrealizable on the given hardware a new design was developed to optimize memory allocation, by storing each piece of data as a single 32-bit entry. This design improves efficiency in setting the aspect ratio of memory allocation to better match that of the BRAM structure enabling deeper memory accessing through address partitioning, and by conforming data to fit within a pre-existing BRAM primitive (eliminating the need to concatenate BRAM components).

C. Data Path Design

The two main components of the IBM algorithm are spectral analysis and classification. For the spectral analysis, we use two parallel n-channel banks of bandpass filters, one bank for each mixture. Each of the two banks of filters is implemented by reusing the same filter hardware, with different coefficients, for n iterations of our primary data pipeline as outlined in Fig. 5.

The filters used for spectral analysis consist of 4 cascaded SOS structures, each of which requires 6 multiplications, as shown in Fig. 4. Due to a limited number of multipliers, each of the SOS structures is implemented by reusing the same multiplier for 6 iterations of a secondary, nested data pipeline. Pipelining the SOS is possible, because the only data dependency is between $w(k)$ and the $b_0 w(k)$ operation; placing the $b_1 w(k-1)$ and $b_2 w(k-2)$ operations between the generation of $w(k)$ and the execution of $b_0 w(k)$ is sufficient to prevent a data hazard.

After bandpass filtering the input to separate it into different frequency channels, spectral analysis is completed by performing envelope detection on each of these channels. The envelope detection hardware consists of a rectifier and a low-pass filter. These hardware components execute fast enough that they can be shared by Mixture 1 and Mixture 2 signals within the primary data pipeline (see Fig. 5).

After spectral analysis, the next stage in the primary pipeline is the classification stage. The classifier hardware consists of a comparator that determines which of the two mixture signals has the larger envelope in a given frequency channel. With the appropriate scaling parameters, the result of this comparison represents an accurate classification of the corresponding time-frequency region as either containing the target signal or noise [1]. The output of the classifer controls the enable pin of an accumulate-and-store block. If the classifier output is '1', then it enables data from the Mixture 1 channel to be accumulated and stored. If the classifier output is '0', the current time-frequency region has been identified as noise and the Mixture 1 data is discarded. This selective accumulation and storage of the Mixture 1 signal across all n frequency channels effectively reconstructs the target signal.

Our IBM implementation permits processing rates of 32 kHz for a 32 channel filter bank, 20 kHz for a 64 channel filter bank, and 10 kHz for a 128 channel filter bank assuming a 50 MHz clock. Additionally, this design leaves room for increased degrees of parallelization if using newer hardware with more multipliers.

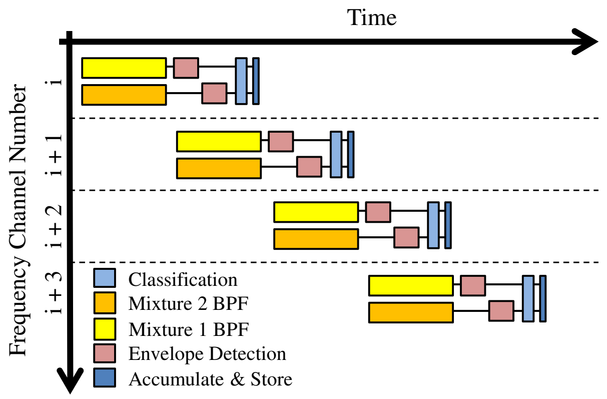

Fig. 5: Timing diagram of the IBM processing chain.

V. RESULTS AND CONCLUSION

We tested our IBM implementation on a Xilinx Spartan 3A-XC3S700A FPGA. The design was realized using a 32 channel filter bank, processing data at a rate of 32 kHz, with a 50 MHz clock. The processing latency was 25 μs, which is perceieved as real-time by the human ear [5]. Data obtained from real-time sampling of this system is shown in Fig. 6.

These results make it clear that real-time implementation of IBM algorithms on an FPGA is feasible. The benefit of an FPGA implementation is made clearer when looking at the overall resource utilization consumed by this design as outlined in Table I. This implementation consumes a small fraction of the total resources available, indicating that this algorithm can be successfully implemented in conjunction with other audio-processing algorithms currently used in hearing prosthetic devices.

TABLE I: FPGA resource utilization for IBM algorithm.

		Total	Used
Spartan 3 (XC3S700A)	BRAMs	20	4
	MULTs	20	13
	4-Input LUTs	11776	2856
Spartan 6 (XC6SLX25)	BRAMs	52	4
	MULTs	38	13
	Slice LUTs	15032	1512

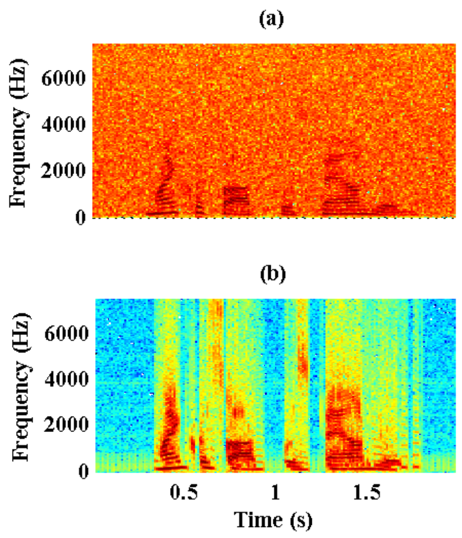

Fig. 6: Real-time IBM results: (a) 0 dB SNR input (b) Output with target speech separated from noise.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant No. 1128478, and by The Neukom Institute for Computational Science.

REFERENCES

[1] J. B. Boldt, U. Kjems, M. S. Pedersen, T. Lunner, and D. L. Wang, "Estimation of ideal binary mask using directional systems," in *Proc. Int. Workshop on Acoustics, Echo, Noise Control*, sept. 2008, vol. 16, pp. 229 – 238.
[2] D. Brungart, P. Chang, B. Simpson, and D. Wang, "Isolating the energetic component of speech-on-speech masking with ideal time-frequency segregation," *J. Acoust. Soc. Amer.*, vol. 120, pp. 4007 – 4018, 2006.
[3] N. Li and P. C. Loizou, "Factors influencing intelligibility of ideal binary-masked speech: Implications for noise reduction," *J. Acoust. Soc. Amer.*, vol. 123, pp. 1673 – 1682, 2008.
[4] H. Saruwatari, Y. Mori, T. Takatani, S. Ukai, K. Shikano, T. Hiekata, and T. Morita, "Two-stage blind source separation based on ica and binary masking for real-time robot audition system," in *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, aug. 2005, pp. 2303 – 2308.
[5] J. Agnew and J. M. Thornton, "Just noticeable and objectionable group delays in digital hearing aids," *Journal of the American Academy of Audiology*, vol. 11, pp. 330 – 336, 2000.
[6] S. Rickard and Y. Ozgur, "On the approximate w-disjoint orthogonality of speech," in *Acoustics, Speech, and Signal Processing (ICASSP)*, 2002.
[7] R. L. Wegel and C. E. Lane, "The auditory masking of one sound by another and its probable relation to the dynamics of the inner ear," *Phys. Rev.*, vol. 23, pp. 266 – 285, 1924.
[8] Q.-J. Fu and G. Nogaki, "Noise susceptibility of cochlear implant users: The role of spectral resolution and smearing," *Journal of the Association for Research in Otolaryngology*, vol. 6, pp. 1438 – 7573, 2005.
[9] D. Wang, "Time-frequency masking for speech separation and its potential for hearing aid design," *Trends in Amplification*, vol. 12, pp. 332 – 353, 2008.
[10] N. Li and P. C. Loizou, "Effect of spectral resolution on the intelligibility of ideal binary masked speech," *J. Acoust. Soc. Amer.*, vol. 123, pp. EL59 – EL64, 2008.
[11] M. Valente, *Hearing Aids: Standards, Options and Limitations*, Thieme, 2 edition, 2002.
[12] Texas Instruments, "Tms320c5504 fixed-point digital signal processor data sheet," 2012.
[13] Xilinx, "Spartan-3a fpga family: Data sheet," 2010.