Real-Time Embedded Implementation of the Binary Mask Algorithm for Hearing Prosthetics

Valerie Hanson, *Student Member, IEEE,* Kofi Odame, *Member, IEEE*

***Abstract*—In this paper, we present a real-time embedded implementation of the binary masking algorithm, which has been shown to significantly improve speech-in-noise intelligibility. Our real-time implementation relies on a balance of parallel processing and hardware pipelining. We have tested and evaluated our implementation on a Spartan 3A FPGA. The measured latency was 8.5 ms. The highest measured improvement in short-time objective intelligibility was 85%.**

***Index Terms*—binary mask, speech enhancement, real–time systems, embedded systems, hearing aids**

I. Introduction

IN a noisy environment, such as a busy street or a crowded restaurant, the normal human hearing system is able to discern and comprehend speech, despite the multitude of intruding, competing sounds [1], [2]. Unfortunately, the ability to solve this "cocktail party problem" is greatly diminished in individuals with sensorineural hearing loss [3], [4]. Even with the use of hearing prosthetics, hearing-impaired individuals often find it extremely difficult to comprehend speech, when it is presented in competition with everyday, background noise [3], [5]–[7].

Current speech enhancement algorithms improve speech quality, but not necessarily intelligibility [8]. While hearing-impaired listeners do benefit from improved speech quality, communication problems still exist if intelligibility is not improved. The ideal binary mask is one algorithm specifically shown to improve speech intelligibility. In [9], the speech intelligibility scores reported by normal hearing listeners increased from 12% to 100% after speech embedded in four-talker babble was processed by the ideal binary mask. Similarly, the ideal binary mask improved speech intelligibility from nearly 0% to 100% in the study described in [10].

We have previously illustrated that the binary mask algorithm is suitable for embedded applications such as the hearing prosthetic system illustrated in Fig. 1 [11]. In this work, we describe hardware design details and measurement results of our real-time implementation of the algorithm.

II. Binary Masking Overview

The idea behind the binary mask (BM) algorithm is as follows. Speech is sparse in the time-frequency domain. If we assume that noise is also sparse in this domain, then it very likely does not overlap with the speech. So, we can remove the noisy regions of the time-frequency plane (by applying the appropriate "binary mask"), which will leave us with intact, noise-free speech [5]. The algorithm is effective even if the noise is not sparse in the time-frequency domain; the overall signal-to-noise ratio (SNR) of the speech can be greatly improved by discarding those regions of the time-frequency plane whose SNR fails to exceed a specified threshold.

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

Manuscript received Month Day, Year; revised Month Day, Year.

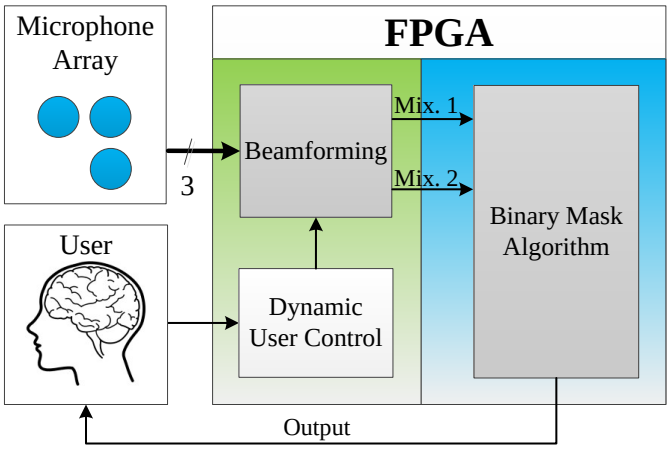

Fig. 1. A Top-level overview of how the real-time binary mask algorithm integrates with the complete hearing prosthetic system [11].

A practical implementation of the algorithm generally has three stages – spectral analysis, classification, and synthesis, as shown in Fig. 2. The spectral analysis stage uses the fast Fourier transform (FFT) or a filter bank to map the original, noisy signal from the time domain to the time-frequency (TF) domain. In the classification stage, each TF unit is either identified as belonging to class '1' (clean speech, a.k.a. "target"), or class '0' (noise). This classification creates a binary mask. In the synthesis stage, the TF-domain version of the original, noisy signal is multiplied by the binary mask, effectively removing all of the noise-containing portions of the signal. After the binary mask is applied, the TF units are then recombined to form a speech signal that is clean (or at least of higher SNR than before).

In [12], Boldt et al. introduced a target/noise classification stage that relies on spatial filtering. Assuming that the direction of the target signal is known, a delay-and-sum beamformer will provide the original, noisy signal, which is denoted $Target + Noise$. A reference noise signal, denoted $Noise$, is

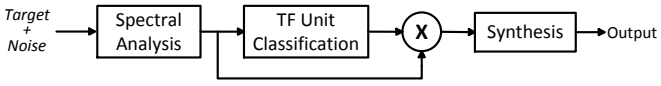

Fig. 2. High level block diagram of the binary mask algorithm.

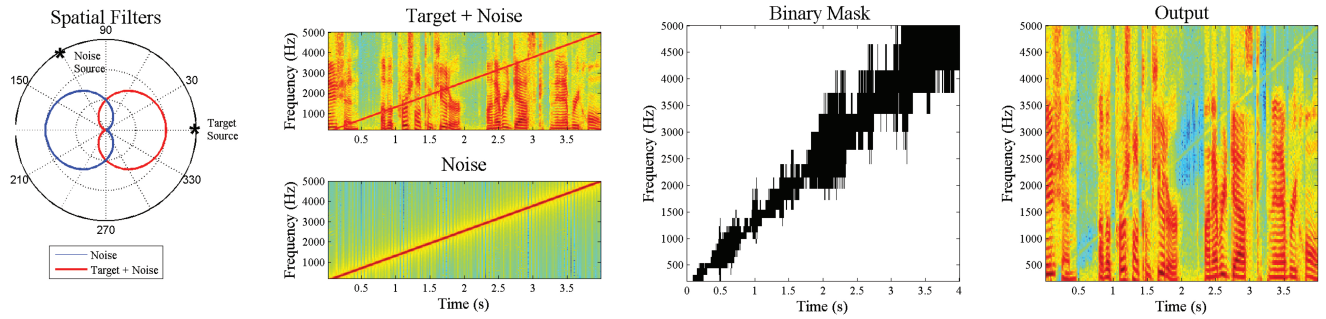

Fig. 3. A visual representation of the BM estimation algorithm. The first column shows the spatial filter cardioids used to create the $Target + Noise$ and $Noise$ signals. The second column shows the outputs of the spatial filters, the $Target + Noise$ and $Noise$ signals, after spectral analysis in the time-frequency domain. The third column shows the resulting binary mask formed. The black regions indicate noise to be removed. The fourth column shows the output after applying the binary mask.

formed by a beamformer that points $180°$ away from the target signal. Both $Target + Noise$ and $Noise$ signals are mapped into the TF domain. A TF unit of the $Target + Noise$ signal is classified as 1 (target) if it contains more energy than the corresponding TF unit in the $Noise$ signal. Otherwise, it is classified as 0 (noise) [1]. Figure 3 illustrates how the algorithm would process speech that has been corrupted by a chirp signal arriving from the $120°$ angle.

III. Real-time implementation details

As depicted in Fig. 4, our implementation of the binary masking algorithm uses a bank of band-pass filters to perform the spectral analysis. Also, classification is performed with a signal-to-noise (SNR) estimate and a comparator. Finally, an accumulator constitutes the synthesis stage. Following are the details of our design.

A. Spectral Analysis

1) Design: The spectral analysis stage of the BM algorithm was implemented as a bank of band-pass filters. The quality of the spectral analysis has a large impact on the overall algorithm performance, therefore the filter bank is designed accordingly.

BM algorithm performance improves with increasing spectral resolution. For instance, in [13], intelligibility scores of binary-masked speech improved from 5% to 65% when the number of filter bank channels increased from 6 to 32. We implemented our spectral analysis stage with a 28-channel bank of 8th order band-pass filters. This gives a moderate amount of spectral resolution without creating objectionably-long group delay. As shown in Fig. 5, our filter bank produces less than a 10 ms group delay over most frequencies of interest, which is in line with group delay exhibited by typical digital hearing aids [14], [15].

BM performance also improves as the frequency span of the filter bank increases. This results from a general widening in the signal processing bandwidth, which causes speech intelligibility to increase and noise sensitivity to decrease [16]–[19]. In a reverberant environment, for instance, increasing the bandwidth from 4 kHz to 7 kHz can increase word identification accuracy from 52% to 80% [19]. We designed the filter bank to span the range 200 Hz to 10 kHz. The upper limit of the filter bank is determined by the sampling rate of 32 kHz, while the lower limit is dictated by numerical stability and group delay concerns.

Audio processing algorithms that use a warped frequency scale to mimic how human hearing mechanisms process sound outperform algorithms based on a uniform scale [20]. Multiple

[1]In the more general version of the algorithm, classification is based on whether or not the ratio of $Target + Noise$ energy to $Noise$ energy for the TF units in question exceeds some threshold that is not necessarily equal to one [12].

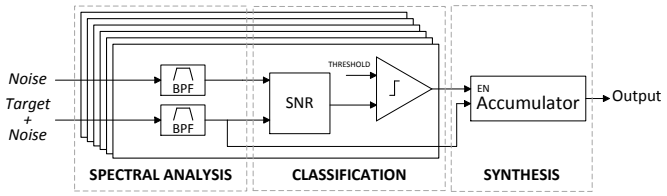

Fig. 4. Block-diagram of the hardware modules required for BM algorithm implementation. Spectral analysis is performed by a bank of 28 bandpass filters. Classification is performed on each time-frequency unit of the $Target + Noise$ signal by comparing its signal-to-noise ratio to some threshold. An accumulator is used to synthesize the enhanced speech.

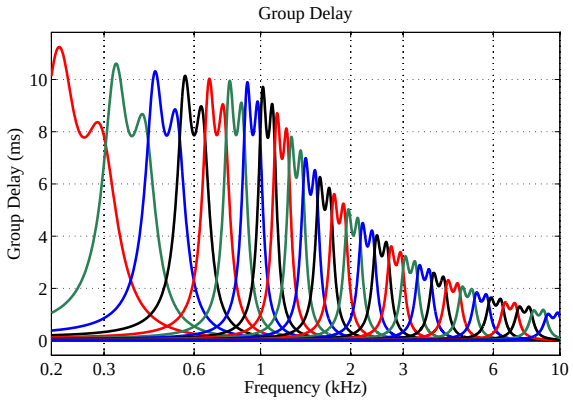

Fig. 5. Time delay caused by digital filtering is referred to as group delay. If the delay between when processed and unprocessed information reach the brain is large enough objectionable effects occur. These effects include echoes, pitch/timbre changes, and corruption of spatial information - all of which may cause potential discomfort and confusion for the user. The filter bank was designed to minimize this delay while maintaining the required frequency resolution, as shown in the above image.

audio-processing warping scale variations exist (Bark, Mel, ERB, etc.), but studies show that performance across scales is comparable [21]. Therefore, we chose to design our filter bank using a linear-log approximation of the Bark scale. The filter bank comprises 7 low-frequency linearly spaced filters, and 21 high-frequency logarithmically spaced filters. The linearly spaced filters span 200 Hz to 935 Hz, and exhibit a filter bandwidth of 105 Hz. The transition frequency and linear bandwidth features were chosen to keep group delay within acceptable levels. Figure 5 shows the group delay and frequency spacing associated with this filter bank. Filters were designed using Matlab's filter design tools. Coefficients for these filters were extracted directly from the filter structures generated.

2) Hardware: The filter bank could be realized with a highly parallel, low latency architecture, since each filter processes data independently of the others. However, such an implementation would require hundreds of multipliers. On the other hand, a purely sequential implementation would require close to 3000 clock cycles to process each input sample. In this section, we describe an implementation that uses — via a balance of parallelization and pipelining — only 4 multipliers and 812 clock cycles per input sample.

The spectral analyses of the *Target + Noise* mixture and the *Noise* signal are performed in parallel using two separate banks of 28 band-pass filters. Each band-pass filter is composed of a cascade of 4 Direct Form 2 (DF2) SOS filters of the form:

$$w(n) = g \cdot x(n) - a_1 \cdot w(n-1) - a_2 \cdot w(n-2) \quad (1)$$

$$y(n) = b_0 \cdot w(n) + b_1 \cdot w(n-1) + b_2 \cdot w(n-2) \quad (2)$$

where g, a_i, and b_i are the filter coefficients, $x(n)$ is the filter input, $y(n)$ is the output, and $w(n)$ is an intermediate variable. Also, the delayed versions of this variable, $w(n-1)$ and $w(n-2)$, are delay elements. From this equation the spectral analysis of each mixture requires $6 \cdot 4 \cdot 28 = 672$ multiplications per sample. To accommodate such a large number of multiplications, we reuse one multiplier block to implement an entire filter bank. Data hazard can be avoided by rearranging multiplication operations to ensure calculation of $w(n)$ is complete before the execution of $b_0 \cdot w(n)$.

The SOS implementation recycles the same multiplier hardware while sequentially changing operand values in order to compute all multiplications required by the DF2 filter equation, as shown in Fig. 6.

The routing pattern laid out in Fig. 6 interleaves equation 1 and equation 2 computations. This interleaved pattern was used because it minimizes the number of memory read instructions that must be performed, since each element must only be read once. In addition, accessing the $w(n-2)$ element before the $w(n-1)$ element permits overwriting the $w(n-2)$ element while simultaneously performing computations dependent on $w(n-1)$.

This SOS structure is then cascaded to form an 8th order filter, as shown in Fig. 7. A multiplexer is used to control the input to the SOS block – it routes the original input signal to the input for the first SOS filter and then routes the output of the previous SOS filter to the input for the next three SOS

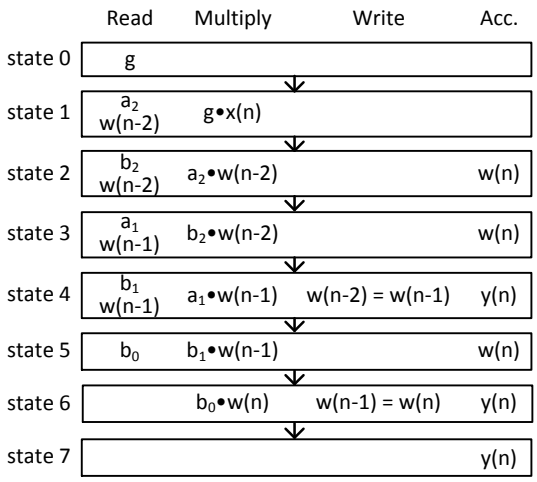

Fig. 6. The state machine for a single SOS block. Multiplication is performed sequentially because the filter is implemented using a single multiplier block.

filters. This 8th order filter block is then recycled sequentially in order to process all filters in the filter bank. While the same input signal is used for all filters, filter-specific coefficients and delay elements are required. Because this information is stored in the same memory blocks for all filters the only data path change required is filter-specific memory addressing. A filter-dependent state machine is used to update the memory addresses to reflect the region of memory corresponding the current filter.

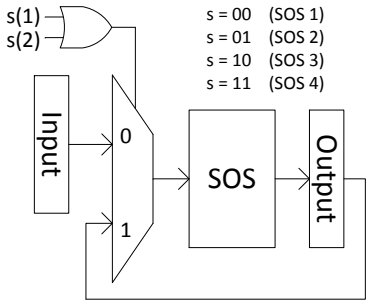

Fig. 7. Hardware diagram and data path for an 8th order band-pass filter. A single SOS filter is reused and cascaded four times to create the 8th order filter.

B. Classification

The binary mask is created by classifying individual TF units of the $Target + Noise$ mixture as target or noise. Classification is performed by comparing the local SNR for a given TF region to a specified threshold - 0 dB in this case. The local SNR is formed by taking the ratio of $Target + Noise$ power to $Noise$ power for a given TF region. Similar to spectral analysis, performing SNR calculations on 28 channels for two inputs would require more multiplications than we have multipliers. Therefore, this block is shared across channels and between mixtures, as shown in Fig. 8.

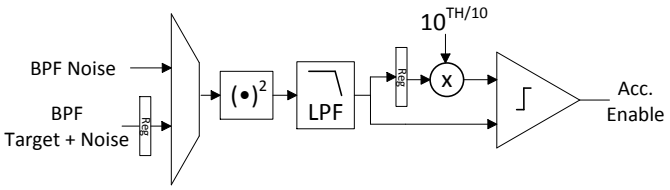

Fig. 8. High level block diagram of SNR estimation for the classification stage. SNR is calculated by extracting the envelope of the "signal" ($Target+Noise$) – via rectification and low-pass filtering – and comparing that to the envelope of the $Noise$. The envelope extraction hardware is multiplexed and shared between the $Target + Noise$ and $Noise$.

For a given sample interval binary mask values are computed for all 28 frequency channels. The binary mask value for one channel is computed by comparing the estimated local SNR to a specified threshold; SNR values above the threshold are classified as target whereas SNR values below the threshold are classified as noise. Local SNR estimation is formed by taking the ratio of $Target+Noise$ power to $Noise$ power in that frequency band. The power for each mixture is computed using an envelope detector. Envelope detectors track amplitude/power changes in a signal by first squaring the signal, then low-pass filtering the output for smoothing purposes. Filtering is performed using a single SOS section, because filter roll-off requirements are more relaxed compared to the spectral analysis roll-off requirements. The filter is implemented with the same SOS architecture as described in the spectral analysis section. A single set of filter coefficients is used for all channels.

As shown in Fig. 9, classification calculations are pipelined with spectral analysis calculations in order to reduce the total processing time. Classification of the channel n mixture is performed simultaneously with the spectral analysis of channel $n + 1$. The total number of operations required to perform classification in a single channel is much smaller than the number of operations required for a single band-pass filter, therefore power computations of $Target + Noise$ and $Noise$ are performed sequentially to save hardware resources, as shown in Fig. 8. The mask is then calculated once the second power computation completes. Once the mask for channel n is assigned this module idles until the current spectral analysis computation of channel $n + 1$ completes. The next classification computation begins in step with the next spectral analysis computation, as shown in the pipeline diagram illustrated in Fig. 9.

C. Synthesis

The binary mask computed in the classification stage identifies the "clean" $Target + Noise$ segments for all channels. A better estimate of the original target signal is formed by masking off $Target + Noise$ segments that are not marked as "clean". Masking is performed by recombining only $Target + Noise$ segments marked as clean. This is achieved by routing the $Target + Noise$ spectral analysis output to an accumulator, which is enabled or disabled by the output of the classification stage.

This component of the algorithm is implemented in the same pipeline stage as the SNR estimation, so the n channel portion of reconstruction is performed in step with the $n + 1$ channel spectral analysis. A breakdown of this pipelined implementation is shown in Fig. 9. This selective accumulate-and-store procedure effectively reconstructs a new signal with improved intelligibility by removing noise.

D. Data Representation

We used an integer representation as the internal data representation, because it requires less hardware, power, and latency than floating-point arithmetic [22], [23]. However, this representation is susceptible to numerical noise and instability. We minimized these nonidealities by scaling the operands and using a computational word length of 32 bits, which is long enough to avoid overflow during processing and to prevent compounded rounding error.

Separate scaling factors were chosen for the input signal and coefficients in order to maximize the number of bits used for computation while preventing scaling overflow. Scaling factors are based on powers of 2 so that re-scaling can be performed quickly using simple bit shifting arithmetic. The input signal is amplified prior to filtering so that when the output signal is scaled back to the original input scaling, rounding errors introduced by the algorithm are reduced by this factor.

In addition to input scaling, each set of filter coefficients is uniformly scaled as well. The coefficient scaling factor was chosen to maximize resolution of the coefficient with the

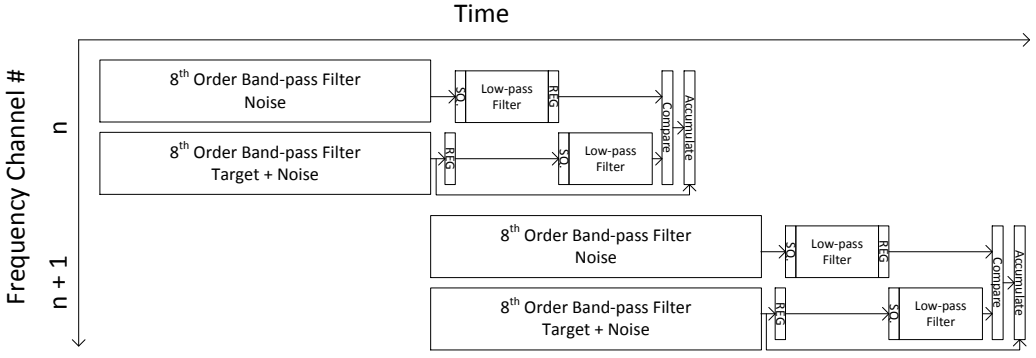

Fig. 9. Timing diagram snapshot showing data path pipeline for frequency channel n and frequency channel $n + 1$.

smallest magnitude (thereby reducing rounding error), while ensuring the coefficient with the largest magnitude does not exceed 2^{32-1}.

E. Memory Allocation

As described in the previous sections, we reuse the same multipliers for several different operations. The multipliers' operands (input samples, filter coefficients, delay elements, etc.) are not fixed, but instead have to be retrieved from memory and changed from one operation to another. In a given clock cycle, this requires simultaneous access to as many as 4 memory elements. So, the operands are stored in 4 separate RAM units. The coefficients for both filter banks are stored in a single RAM unit. The delay signals for the $Noise$ and the $Target + Noise$ filter banks are each stored in a separate RAM unit. Finally, a fourth RAM unit is used to store the delay elements for the low pass filter (the filter used during classification) for both the $Noise$ and the $Target + Noise$ signals.

One RAM unit stores the coefficients for the 28-channel bank of 8th order band-pass filters (this single set of coefficients is used to filter both the $Target + Noise$ and the $Noise$ signals). Each band-pass filter consists of two unique SOS filter structures, each of which are implemented twice for a total of four SOS structures. Each SOS structure is defined by 6 coefficients, so the total number of coefficients stored for the bank of band-pass filters is $28 \cdot 2 \cdot 6 = 336$. To form the memory address needed to access a given coefficient, we concatenate the channel index with the first bit of the SOS index followed by the coefficient index (see Fig. 10).

The $Target + Noise$ filter bank contains 28, 8th-order filters. Each 8th-order filter is made up of 4 SOS's, each of which requires the storage of 2 delay elements. In total, there are $28 \cdot 4 \cdot 2 = 224$ delay elements for the $Target + Noise$ filter bank, and they are all stored in a single RAM unit. Similarly, the 224 delay elements of the $Noise$ filter bank are stored in a single RAM unit. Figure 10 illustrates the addressing scheme for these memory elements.

One last RAM unit is used to store the delay elements of the low pass filter SOS (this is the filter used for energy extraction in the SNR estimation stage of the algorithm). This last RAM contains delay elements for both input signals, since the shared low pass filter SOS hardware only requires delay elements from one input signal at any given time. The RAM unit stores $2 \cdot 2 \cdot 28 = 112$ delay elements. Figure 10 illustrates the addressing scheme for this memory.

Since the coefficients are accessed independently and are never overwritten, it is sufficient to store them in single-port RAM. Memory used to access the filter delay elements requires a two-port configuration, because data from two separate memory addresses must be accessed in a single clock cycle. (The newer delay element is read while the older delay element is overwritten.) True dual-port ram permits simultaneous read and write access to separate memory addresses via two independent ports. While the same memory space is shared for true dual-port RAM port-specific read-to-write aspect ratios can be realized by splitting the memory space. Given that the port aspect ratios are symmetrical (both ports are used to access the same set of data), and simultaneous read access is not required the delay element memory can be configured as simple dual-port RAM instead of true dual-port RAM. This configuration comprises one port for reading and one port for writing.

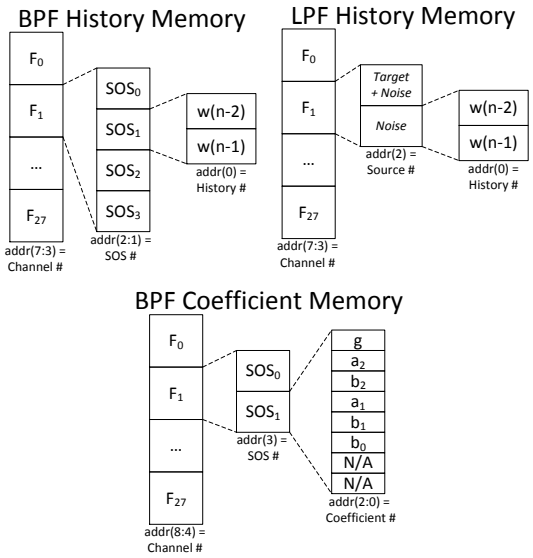

Fig. 10. Diagram showing the partitioning scheme used for memory addressing.

IV. Experimental Methods

We realized our algorithm implementation on a prototype FPGA platform. Test procedure details and results are presented in the following two sections.

A. Audio Databases

The target signals used for testing were speech samples obtained from the TIMIT Acoustic-Phonetic Continuous Speech Corpus [24]. This database, comprising 6300 sentences spoken by male and female speakers from eight American English dialect regions, was used to ensure that results are not biased towards a particular speaker gender, dialect, or pronunciation style. The noise signals used for testing were obtained from the AURORA database [25]. The specific sounds chosen include babble (speech shaped noise), restaurant noise, and white noise.

B. Speech-in-Noise Simulation

While this algorithm is designed for implementation in conjunction with delay-and-sum beamforming of raw microphone signals, algorithm performance is best verified under controlled test conditions. Therefore, in order to generate signals with precise SNR and directionality microphone recording, beamforming, and beam steering were simulated in Matlab.

1) Setting SNR: Noisy speech signals were generated in Matlab using a weighted sum of clean speech and noise. SNR levels were adjusted by changing the weight applied to the noise signal. SNR was measured using the global SNR output from the *snrseg* function in the *Voicebox* toolbox [26]. A binary search tree approach was used to determine the noise weighting required to obtain specific SNR values. The noisy speech signal was normalized to a maximum amplitude of 1 prior to outputting the signal to the FPGA in order to prevent the output from clipping.

2) Simulating Microphones: Three microphone signals were simulated for an L-shaped microphone array, as shown in Fig. 11. This configuration was used to enable two-dimensional spatial localization of sound (within the plane containing the microphones) via beamforming.

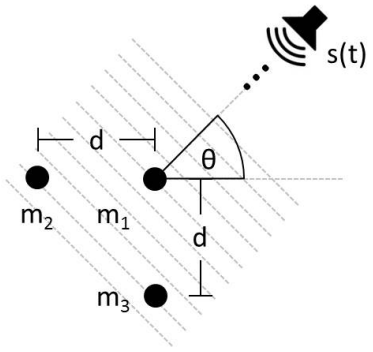

Fig. 11. Diagram of the hardware orientation assumed for simulating a three microphone array, where microphones are labeled as $m1$, $m2$, and $m3$. The sound source is labeled as $s(t)$, and the spatial positioning of the sound source relative to the array is labeled as θ. The spacing between microphones, d, was set to 0.4 cm.

For each sound source the three microphone signals were simulated by creating delayed versions of the original signal. Delays depend on the spatial positioning of the sound source relative to the array, as outlined in Eqn. 3, Eqn. 4. Spatial delays in the horizontal dimension were represented by the $m1$-$m2$ microphone pair, whereas spatial delays in the vertical dimension were represented by the $m1$-$m3$ microphone pair.

The horizontal delay is set by:

$$T_x = \frac{d}{c}\cos(\theta) \tag{3}$$

The vertical delay is set by:

$$T_y = \frac{d}{c}\sin(\theta) \tag{4}$$

In Eqn. 3 and Eqn. 4 d corresponds to the separation between $m1$ and $m2$, as well as between $m1$ and $m3$, c is the speed of sound, and θ is position of the sound source relative to the array. Using $s(t)$ to represent the sound source, simulated microphone signals can be written in terms of these delays, to produce the expressions given in Eqn. 5.

$$\begin{aligned} m_1(t) &= s(t) \\ m_2(t) &= s(t - T_x) \\ m_3(t) &= s(t - T_y) \end{aligned} \tag{5}$$

In order to realize negative delays $t = 0$ must be assumed to be some number of samples after start of each audio sample.

3) Creating Directional Filters: The two directionally filtered signals required as inputs to the binary masking algorithm are generated by applying delay and sum beamforming [27] and beam steering techniques [28] to the three simulated microphone signals in order to generate a simulated cardioid pointing in the direction of the target speech signal, and a simulated cardioid pointing 180° opposite the direction of the target speech signal.

C. Experimental Setup

Testing was carried out using Digilent's Spartan-3A development board, which features the Xilinx Spartan 3A XC3S700A FPGA. The clock frequency of the FPGA was 50 MHz. Simulated audio signals, sampled at 31.25 kS/s, were transmitted to the embedded system in the analog domain via the computer's sound card output jack. Digilent Inc.'s PMODAD1, featuring the AD7476A serial interface ADC, was used for signal conversion back to the digital domain. Additional hardware required include the on-board LTC2624 DAC (used to generate analog signals for latency testing), Agilent's MSO7014B oscilloscope (used to record analog signals for latency testing), and an FT232RL USB to serial bridge (used to record digital signals for intelligibility testing).

D. Test 1 - Latency

The latency of the algorithm was measured for 96 different noisy speech input signals. These input signals correspond to 32 unique speech signals masked by three types of noise (babble, restaurant, and white noise) at a mixture level of -5 dB SNR. The speech signals comprise a two-sentence utterance randomly selected from one male and one female speaker from each of the eight dialect regions laid out in the TIMIT corpus.

Latency as referred to here is defined as the time delay between when the original signal enters the system and when the processed output is made available. This delay was measured by outputting both the processed signal and the raw input in the analog domain so that an oscilloscope could be used to simultaneously record signals with high sampling resolution. The oscilloscope recorded approximately 1 s of data, which was then analyzed in Matlab. The *sigalign* routine obtained from the Matlab toolbox *Voicebox* was used to measure the delay. This function aligns clean and noisy data by maximizing the correlation coefficient between the two signals [26]. The total latency included delay contributions from the FPGA algorithm core, the data converters, and the I/O routing fabric. To determine the latency of the FPGA algorithm core alone, a signal was routed directly from an FPGA input pin to output pin (bypassing the algorithm core), and the measured delay was subtracted from the total latency.

E. Test 2 - Speech Intelligibility

The intelligibility improvement afforded by this algorithm was measured for 5 male and 5 female speakers randomly selected from 5 of the 8 dialect regions laid out in the TIMIT corpus masked by three types of noise (babble, restaurant, and

white noise). Intelligibility improvement for each speaker was measured for multiple noise angles – the angular separation between target and noise sources – and multiple SNR levels. Measurements were taken for noise angles ranging from $15°$ to $345°$ at $15°$ increments, and SNR levels ranging from -10 dB to 5 dB at 5 dB increments.

We used the short time objective intelligibility (STOI) index [29] to measure the amount of intelligibility improvement that our BM implementation provides. The STOI index is the average of the linear correlation coefficients between corresponding TF units of clean and processed speech samples [29]. The linear correlation coefficient, $d_{j,m}$, for a pair of corresponding TF units is given as

$$d_{j,m} = x_{j,m}^{\mathrm{T}} \cdot y_{j,m}, \tag{6}$$

where $x_{j,m}$ is the normalized, zero-mean short-time temporal envelope of the clean speech at time frame index m and frequency band j. The variable $y_{j,m}$ is similarly defined, but for the processed speech. Taal et al. provide more details in [29]. The attraction of the STOI index is that it displays a monotonic (albeit nonlinear) relation with speech intelligibility scores from human auditory task testing. In addition, the STOI index has been tested on English speech processed by time-frequency varying gain functions that are similar to those applied here. Also, STOI performs better than other commonly-used measures including the Dau Auditory Model, Coherence Speech-Intelligibility Index, Normalized Covariance Based Speech Transmission, Frequency-Weighted Segmental SNR, and Normalized Subband Envelope Correlation [29].

Two intelligibility measures were calculated for each condition tested - one corresponding to the intelligibility of the processed output relative to the original target speech, and the other corresponding to the intelligibility of the raw microphone signal relative to the original target speech. The percent intelligibility improvement between the processed and raw signals was measured. The percent improvement was used instead of absolute improvement values, because it allows for result comparison and averaging across conditions where baseline intelligibility measures differ.

All signals used for intelligibility calculations - target, raw microphone, and processed signal were transmitted through the same conversion chain to remove the effect of any mismatch/distortions that were not a result of the algorithm itself. Optimal STOI performance requires time alignment of both signals, therefore prior to STOI computation all signals were aligned using the *sigalign* function to remove shifts caused by system latency and alignment differences resulting from the data collection technique.

V. Results and Discussion

A. Latency

Results from latency test measurements are shown in Fig. 12. The latency is primarily due to the filter bank group delay, which changes with frequency, as was shown in Fig. 5. Since we tested various speech samples and types of noises that have different frequency compositions, the measured latency varied, with an average value of 8.48 ms and a standard deviation of 0.03 ms. This amount of delay is noticeable to listeners [30], and also does not take into account the delay contributed by other parts of the system.

We can reduce the latency of the algorithm by reducing the frequency resolution of the filter bank, particularly at the low frequency range. The trade-off is that lower frequency resolution makes the binary masking algorithm less effective at improving speech intelligibility [10].

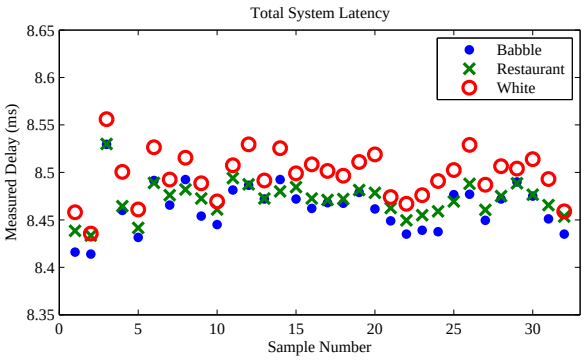

Fig. 12. Measured latency between input and processed output for speech masked by three types of noise at -5 dB SNR.

B. Speech Intelligibility

Results from the test measuring speech intelligibility are shown in Fig. 13. The values reported in Fig. 13 correspond to the average percent increase in STOI value relative to the raw baseline signal STOI value for all speakers tested for a given condition.

Figure 13 shows that this algorithm implementation results in intelligibility measure improvements for all three noise conditions for both male and female speakers, with improvement effects increasing with decreasing SNR levels. These results reveal that intelligibility improvement degrades when the noise angle is less than $90°$; however, this performance can be improved if an adaptive instead of fixed SNR threshold is used in the classification stage, as described in [12].

Spectrograms are also included to show improvements for a single test condition in the TF domain. Figure 14 shows improvements for a male speaker masked by babble noise at $105°$. One should note that the SNR as shown in the *Target + Noise* mixture is better than the condition stated due to directional filtering.

C. Hardware platform

Our algorithm implementation requires 4 multipliers, 4 adders, 1 comparator and 896 words of RAM. The memory must be able to accommodate simultaneous access to 4 different addresses. On the Spartan 3A FPGA, these components are easily accommodated and consume approximately 15 mW of power. However, the overhead cost (area and power consumption) of routing fabric and reconfiguration circuitry means that an FPGA can only be used as an experimental platform. To be useful in a practical hearing device, the algorithm must be realized in either a low-power digital signal processor (DSP) or as an application specific integrated circuit (ASIC).

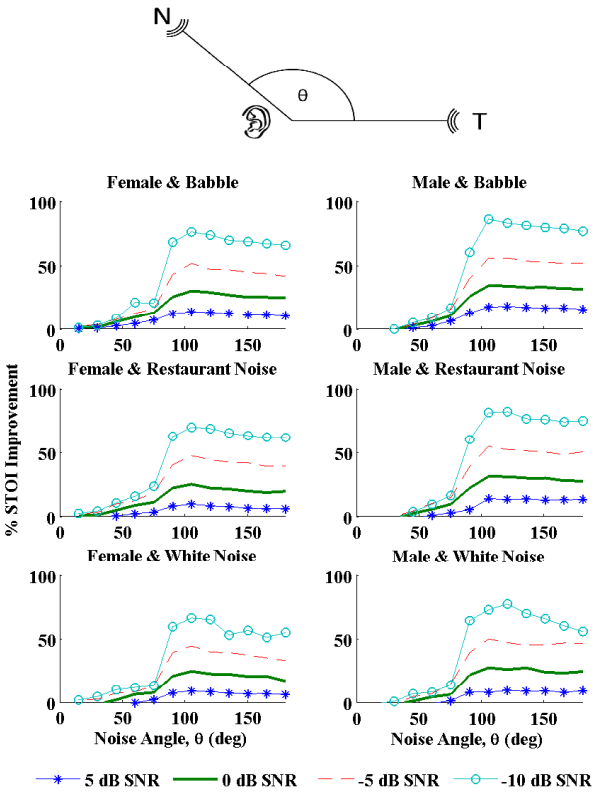

Fig. 13. 1) Top view of noise angle, with the source locations measured relative to listener. 2) Measured average percent intelligibility improvement from raw microphone signal to processed, binary masked signal under different test conditions.

For the algorithm to fit on current low-power processors, it might have to be modified slightly. For example, if we reduced the level of parallelization such that only 2 multipliers were needed simultaneously, then our implementation could be realized in the C55x series of low power DSPs (Texas Instruments, Dallas, TX) [31]. The reduced level of parallelization would increase the number of clock cycles needed to process one data sample from 812 to 1120. At a clock rate of 50 MHz, this corresponds to a power consumption of 7.5 mW and a maximum sample rate of 44 kHz. As another example, we could reduce the number of filters in the spectral analysis filter bank to 16, in order to use the BelaSigna 2xx series of audio processors (ON Semiconductor, Phoenix, AZ) [32]. For the BelaSigna, the maximum sample rate would have to be reduced to 16 kHz, but its filter bank hardware accelerator would ensure that power consumption is kept below 1 mW.

On an ASIC, the majority of the chip area and power budget will be used by the multipliers and the RAM. With the design introduced by [33], a single, 50 MHz 32x32-bit multiplier takes up an area of 0.06 mm^2 and consumes 0.16 mW of power on a 90 nm process. Meanwhile, a block of RAM sized 896 words by 32 bits — also implemented on a 90 nm process — takes up an area of 0.01 mm^2 and consumes less than 10 uW [34]. So, a conservative estimate of the requirements of the entire ASIC are 0.2 mm^2 of die area and 1 mW of power.

VI. Conclusion

The algorithm implementation presented in this work balances the hardware consumption benefits of a sequential implementation with the latency benefits of a parallel implementation. The implementation we designed completes processing in 812 clock cycles (compared to 3000 for a purely sequential

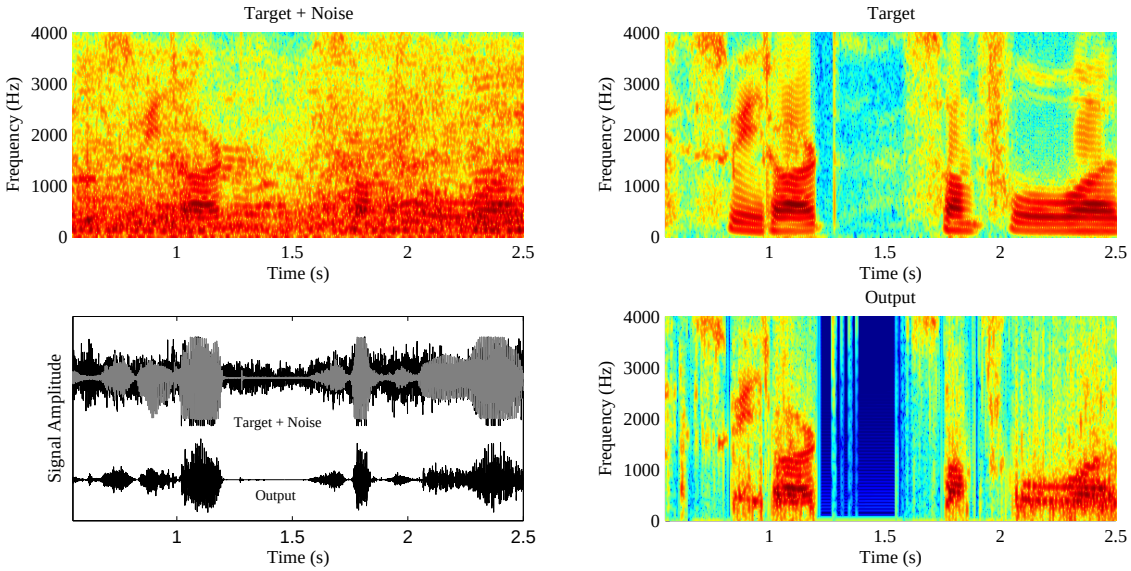

Fig. 14. Measurement results obtained from processing a male speaker's speech, $Target$, that has been masked by babble noise, $Noise$, at -10 dB SNR. The speaker and the babble noise source are in the same horizontal plane, separated by 105 degrees. The upper left spectrogram shows the spectral analysis of the $Target + Noise$ signal. The bottom right shows a spectrogram of our implemented binary mask output. The algorithm's output corresponds very closely to that of the ideal binary mask. A time series of $Target + Noise$ and the output of our binary mask implementation are shown in the bottom left panel.

design), and requires only 4 multipliers (compared to 112 for a highly parallelized design).

The algorithm latency was primarily due to filter bank group delay, and had an average value of 8.5 ms. The implemented algorithm was able to improve the intelligibility of noisy speech by as much as 85%, as measured by an objective intelligibility index.

The low resource requirements of our design make it appropriate for a hearing aid application, where it can be realized either in a low power DSP or in a low power ASIC.

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] E. C. Cherry, "Some experiments on the recognition of speech, with one and with two ears," *The Journal of the Acoustical Society of America*, vol. 25, no. 5, pp. 975–979, Sep. 1953.

[2] S. Haykin and Z. Chen, "The cocktail party problem," *Neural Computation*, vol. 17, no. 9, pp. 1875–1902, Sep. 2005.

[3] M. C. Anzalone, L. Calandruccio, K. A. Doherty, and L. H. Carney, "Determination of the potential benefit of time-frequency gain manipulation," *Ear and Hearing*, vol. 27, no. 5, pp. 480–492, Oct. 2006.

[4] N. Marrone, C. R. Mason, and G. Kidd, Jr., "The effects of hearing loss and age on the benefit of spatial separation between multiple talkers in reverberant rooms," *The Journal of the Acoustical Society of America*, vol. 124, no. 5, pp. 3064–3075, Nov. 2008.

[5] D. Wang, "Time-frequency masking for speech separation and its potential for hearing aid design," *Trends in Amplification*, vol. 12, no. 4, pp. 332–353, Dec. 2008.

[6] S. Kochkin, "MarkeTrak VIII: Why my hearing aids are in the drawer: the consumers' perspective," *The Hearing Journal*, vol. 53, no. 2, pp. 34–42, Apr. 2000.

[7] ——, "MarkeTrak VIII: Consumer satisfaction with hearing aids is slowly increasing," *The Hearing Journal*, vol. 63, no. 1, pp. 19–20, Jan. 2010.

[8] P. C. Loizou and G. Kim, "Reasons why current speech-enhancement algorithms do not improve speech intelligibility and suggested solutions," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 19, no. 1, pp. 47–56, Jan. 2011.

[9] D. S. Brungart, P. S. Chang, B. D. Simpson, and D. Wang, "Isolating the energetic component of speech-on-speech masking with ideal time-frequency segregation," *The Journal of the Acoustical Society of America*, vol. 120, no. 6, pp. 4007–4018, Dec. 2006.

[10] N. Li and P. C. Loizou, "Factors influencing intelligibility of ideal binary-masked speech: Implications for noise reduction," *The Journal of the Acoustical Society of America*, vol. 123, no. 3, pp. 1673–1682, Mar. 2008.

[11] V. Hanson and K. Odame, "Real-time source separation on a field programmable gate array platform," in *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, Aug. 2012, pp. 2925–2928.

[12] J. Boldt, U. Kjems, M. S. Pedersen, T. Lunner, and D. Wang, "Estimation of the ideal binary mask using directional systems," in *11th International Workshop on Acoustic Echo and Noise Control - IWAENC*, Sep. 2008, pp. 395–401.

[13] N. Li and P. C. Loizou, "Effect of spectral resolution on the intelligibility of ideal binary masked speech," *The Journal of the Acoustical Society of America*, vol. 123, no. 4, pp. 59–64, Apr. 2008.

[14] W. Cole, "Group delay and processing delay in hearing aids the facts and the fantasies," Audioscan, Dorchester,ON Canada, Tech. Rep., 2005.

[15] R. Herbig and J. Chalupper, "Acceptable processing delay in digital hearing aids," *The Hearing Review*, vol. 17, no. 1, pp. 28–31, Jan. 2010.

[16] H. Peeters, C.-C. Lau, and F. Kuk, "Speech-in-noise potential of hearing aids with extended bandwidth," *The Hearing Review*, vol. 18, no. 3, pp. 28–36, Mar. 2011.

[17] D. L. Beck and J. Olsen, "Extended bandwidths in hearing aids," *The Hearing Review*, vol. 15, no. 11, pp. 22–26, Oct. 2008.

[18] J. Rodman, "The effect of bandwidth on speech intelligibility," Polycom, San Jose, CA, Tech. Rep., Sep. 2006.

[19] P. Barnett, "Overview of speech intelligibility," *Institute of Acoustics*, vol. 21, pp. 1–15, 1999.

[20] H.-M. Park, S.-H. Oh, and S.-Y. Lee, "A bark-scale filter bank approach to independent component analysis for acoustic mixtures," *Neurocomputing*, vol. 73, no. 1, pp. 304–314, Dec. 2009.

[21] B. J. Shannon and K. K. Paliwal, "A comparative study of filter bank spacing for speech recognition," in *Microelectronic engineering research conference*, Nov. 2003.

[22] C. Inacio and D. Ombres, "The DSP decision: fixed point or floating?" *Spectrum, IEEE*, vol. 33, no. 9, pp. 72–74, Sep. 1996.

[23] A. V. Oppenheim, R. W. Schafer, J. R. Buck *et al.*, *Discrete-time signal processing*. Upper Saddle River, NJ: Prentice Hall, 1999, vol. 5.

[24] J. S. Garofolo, *TIMIT: Acoustic-phonetic Continuous Speech Corpus*. Linguistic Data Consortium, 1993.

[25] H.-G. Hirsch and D. Pearce, "The aurora experimental framework for the performance evaluation of speech recognition systems under noisy conditions," in *ASR2000-Automatic Speech Recognition: Challenges for the new Millenium ISCA Tutorial and Research Workshop (ITRW)*, Sep. 2000, pp. 29–32.

[26] M. Brookes *et al.*, "VOICEBOX: Speech processing toolbox for Matlab," *Software, available [Jun. 2013] from www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html*, 1997.

[27] G. W. Elko and A.-T. N. Pong, "A simple adaptive first-order differential microphone," in *Applications of Signal Processing to Audio and Acoustics, 1995., IEEE ASSP Workshop on*, Oct. 1995, pp. 169–172.

[28] ——, "A steerable and variable first-order differential microphone array," in *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, Apr. 1997, pp. 223–226.

[29] C. H. Taal, R. C. Hendriks, R. Heusdens, and J. Jensen, "An algorithm for intelligibility prediction of time-frequency weighted noisy speech," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 19, no. 7, pp. 2125–2136, Sep. 2011.

[30] 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, no. 6, pp. 330–336, Jun. 2000.

[31] Texas Instruments, "TMS320C5535, 'C5534, 'C5533, 'C5532 Fixed-Point Digital Signal Processors," Aug. 2011, datasheet, [Revised Mar. 2012].

[32] ON Semiconductor, "WOLA Filterbank Coprocessor: Introductory Concepts and Techniques," Apr. 2009, Appl. Note ABD8382/D.

[33] S. K. Hsu, S. K. Mathew, M. A. Anders, B. R. Zeydel, V. G. Oklobdzija, R. K. Krishnamurthy, and S. Y. Borkar, "A 110 GOPS/W 16-bit multiplier and reconfigurable PLA loop in 90-nm CMOS," *Solid-State Circuits, IEEE Journal of*, vol. 41, no. 1, pp. 256–264, Jan. 2006.

[34] M. Yamaoka, N. Maeda, Y. Shinozaki, Y. Shimazaki, K. Nii, S. Shimada, K. Yanagisawa, and T. Kawahara, "90-nm process-variation adaptive embedded SRAM modules with power-line-floating write technique," *Solid-State Circuits, IEEE Journal of*, vol. 41, no. 3, pp. 705–711, Mar. 2006.