A CASE STUDY OF MACHINE LEARNING HARDWARE: REAL-TIME SOURCE SEPARATION USING MARKOV RANDOM FIELDS VIA SAMPLING-BASED INFERENCE

Glenn G. Ko[1] *Rob A. Rutenbar*[2]

University of Illinois at Urbana-Champaign, Department of Electrical and Computer Engineering[1]
University of Illinois at Urbana-Champaign, Department of Computer Science[2]
{giko[1],rutenbar[2]}@illinois.edu

ABSTRACT

We explore sound source separation to isolate human voice from background noise on mobile phones, e.g. talking on your cell phone in an airport. The challenges involved are real-time execution and power constraints. As a solution, we present a novel hardware-based sound source separation implementation capable of real-time streaming performance. The implementation uses a recently introduced Markov Random Field (MRF) inference formulation of foreground/background separation, and targets voice separation on mobile phones with two microphones. We demonstrate a real-time streaming FPGA implementation running at 150 MHz with total of 207 KB RAM. Our implementation achieves a speedup of 20X over a conventional software implementation, achieves an SDR of 6.655 dB with 1.601 ms latency, and exhibits excellent perceived audio quality. A virtual ASIC design shows that this architecture is quite small (less than 10M gates), consumes only 69.977 mW running at 20 MHz (52X less than an ARM Cortex-A9 software reference), and appears amenable to additional optimization for power.

Index Terms— Machine learning, source separation, Markov Random Field, Gibbs sampling, real-time streaming hardware

1. INTRODUCTION

There is growing interest in the deployment of Machine Learning (ML) algorithms for applications that classify, categorize, label, and extract actionable intelligence from large and complex data sources. To date, this has been successful for enterprise applications, which use ML and data mining ideas in the context of data-to-action analytics. The research community has shown that many perceptual applications, such as those in computer vision and machine listening can similarly be addressed using ML ideas. Most ML algorithms are, however, computationally intensive, requiring either extreme operation counts, memory bandwidth, or both. For enterprise level tasks, this requires distributed software solutions in large clouds. But for mobile appliances, such solutions are infeasible. The new question is how to implement ML applications, especially perceptual tasks in a practical mobile form.

As a step toward answering this question, we focus on one such perceptual ML case study. We explore *sound source separation*, which refers to separating human voice from background noise. In the real audio world, humans listen to a mixture of multiple sound signals and the brain separates out the sound sources naturally. This is a classical signal processing problem called the *cocktail party problem*. However, this source separation problem remains difficult for computers. The typical issue in implementing separation is the trade-off between usable quality and computational complexity.

When minimal or no information is provided about the source or the mixing process, this problem is called Blind Source Separation (BSS) [1]. Assuming a situation where the number of mixture signals is less than the sources - two microphones on a cellphone - we focus on a recently introduced formulation for BSS that formulates the problem as *Maximum A Posteriori* (MAP) inference on a grid-connected Markov Random Field [2]. Roughly speaking, we build a 2-D "image" in the form of a spectrogram for sound mixtures obtained from each of the microphones. Each column of the image represents samples from one time point. Each pixel represents the ratio of energy of the desired sound source to the interfering sound source at a specific frequency. MRF inference solves iteratively for binary 0/1 labeling of this image, identifying which frequencies at which time points properly belong to either the desired sound source ("1") or the interfering sound source ("0").

In this work, we present a novel hardware implementation of sound source separation using Markov Random Fields. We are not the first to explore hardware implementations of sound source separation [3, 4, 5]. However, our preferred MRF model has some useful advantages, e.g., the ability to incorporate prior information like the smoothness of the interchannel level difference of the sound mixture spectrograms. Previous implementations have very long latency or do not discuss latency at all, which is critical for any practical implementations; our hardware has very small latency enabling real-time streaming source separation.

There are prior studies that focus solely on accelerating Gibbs sampling inference for Markov Random Fields, which we have used for source separation. One approach is to accelerate through incorporation of novel devices. [6, 7] suggest novel stochastic circuit architectures. [8] suggests using resonance energy transfer circuits, which is an outdated architecture. Neither approach seems practical in any real system. Another approach used algorithmic modifications and parallelization [9], but heavy control overhead appears to pose challenges to achieve real-time performance for mobile.

The reminder of the paper is organized as follows. Section 2 reviews the MRF inference model. Section 3 gives an overview of our real-time streaming hardware architecture, along with a discussion of trade-offs, such as the size of the MRF, and the number of sampling inference iterations to find an acceptable MAP labeling. Section 4 gives the details of test scenario and offers hardware implementation details and results. Section 5 offers concluding remarks.

This work was supported in part by Systems on Nanoscale Information fabriCs (SONIC), one of the six SRC STARnet Centers, sponsored by MARCO and DARPA, and the National Science Foundation under Grant No. CCF-1302641. The authors also acknowledge Paris Smaragdis, Minje Kim and Adel Ejjeh of University of Illinois at Urbana-Champaign for their input on MRF methods, and experimental power analysis.

2. SOURCE SEPARATION USING SAMPLING-BASED INFERENCE ON MARKOV RANDOM FIELDS

Sound source separation can be formed as an MAP inference problem, where a binary mask label $l \in \{0, 1\}$ that corresponds to the category of sound sources is assigned to each time-frequency point in the spectrograms obtained from taking the Short-Time Fourier Transform (STFT) of input sound mixtures from a microphone array. The goal is to find the most probable label assignments for all the points (or pixels) in the spectrograms. In this work, we assume that there are desired and interfering sources. This MAP inference problem can be formulated in terms of the parameters defined on an undirected grid graph (i.e. a grid MRF) with nodes ν and edges ε as

$$\arg\min_{l} E(l) = \arg\min_{l} \left\{ \sum_{s \in \nu} \theta_s(l_s) + \sum_{(s,t) \in \varepsilon} (l_s, l_t) \right\} \quad (1)$$

where $\theta_s(l_s)$ and $\theta_{st}(l_s, l_t)$ are parameters which penalize certain choices for a set of labels l and are called the *data cost* and *smoothness cost*, respectively [10]. The data cost $\theta_s(l_s)$ is related to the likelihood of a label l_s being assigned to the node s. We follow the Gaussian-based data cost formulation of our previous work [2] in which the data cost is defined as a function of two Gaussians,

$$\theta_i(x_i) = \begin{cases} \frac{(A_i - \mu_0)^2}{2\sigma_0^2} & \text{if } x_i = 0 \\ \frac{(A_i - \mu_1)^2}{2\sigma_1^2} & \text{if } x_i = 1, \end{cases} \quad (2)$$

where A_s is the Inter-channel Level Difference (ILD), the log ratio of the energy values seen at the s (time, frequency) point between the spectrogram of the two microphones. The means, μ_0 and μ_1, and the variances, σ_0 and σ_1, correspond to the mean and variance of each sound component of the energy ratio Gaussians. The smoothness cost $\theta_{st}(l_s, l_t)$ models the prior preference of two neighboring nodes, defined on edge (s, t), to encode spatial locality in frequency and time domain in the spectrogram as follows [2]:

$$\theta_{i,j}(x_i, x_j) = \|x_i - x_j\|^2. \quad (3)$$

The goal of the optimization problem (1) is to find the set of labels l among all possible per-pixel label choices that minimize the objective function. This overall sum is called *energy* and (1) is called an *energy minimization problem*.

In recent years, several algorithms have been proposed to solve the above MRF inference problem [11]. Existing algorithms cannot be run in real-time without hardware acceleration. In the domain of MAP inference, we argue that sampling-based algorithms inherently have the most potential for parallel computation, which is an important consideration for hardware implementations. The Gibbs sampler, a variant of the Markov Chain Monte Carlo (MCMC) sampler, was first introduced in the context of computer vision by Geman and Geman [12]. It is used to obtain a sequence of samples approximately derived from a specified distribution when direct sampling is intractable. The samples can be used to approximate the marginal distribution of one of the variables of the MRF. Suppose we have a joint distribution $P(x_1, ..., x_n)$ where x_i are the labels on each node as stated in the prior discussion. We can use the Gibbs sampler to sample from the $P(x_1, ..., x_n)$ distribution. In the case of an MRF, the conditional distribution of each variable is reduced to the conditional distribution given the Markov blanket of that variable.

Just as it is common for users to provide an initial seed for image segmentation [13], we can also provide initial observations or guesses about the underlying distribution. However, such user intervention is hard to impose on a real-time system. Thus, we perform an unsupervised learning where the Expectation-Maximization (EM) approach is used to estimate means and variances of the distribution as shown in Algorithm 1. We start with rough guesses of the Gaussian parameters for the data cost formulation based on the geometric orientation of microphones and sources. After one Gibbs sampling inference iteration on the graph, Gaussian variables are recalculated from the converged labels by grouping the nodes corresponding to each of the binary labels. The inference iterations are repeated using the updated data cost based on the new Gaussian parameters until convergence and the resulting labels are used to create the mask for separation. This is especially useful in scenarios where the sources are not stationary with respect to the microphones.

Algorithm 1 Creating masks for source separation via MCMC-EM

procedure SOURCESEPARATIONMASK(A_i, x)
  for each EM iteration **do**
    ConstructMRF
    (E-Step) GibbsSampling:
    for $t = 1$ to max iteration T **do**
      for each node $i = 1$ to I **do**
        $x_i^{(t+1)} \sim P(x_i | x_1^{(t+1)}, ..., x_{i-1}^{t+1}, x_{i+1}^{t}, ..., x_n^{(t)})$
      end for
    end for
    (M-Step) UpdateGaussians $\mu_{0,1}$ and $\sigma_{0,1}$
  end for
  return x
end procedure

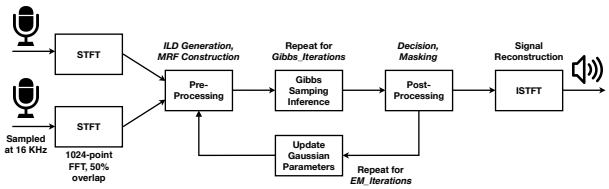

Fig. 1. Streaming source separation

3. REAL-TIME STREAMING IMPLEMENTATION

The flow of the algorithm and each corresponding step of the streaming source separation system is shown in Figure 1. The following subsections will describe each of our steps in detail, providing explanation for design choices made with the final goal of achieving a real-time streaming implementation.

3.1. Spectrogram generation

In the first step, the pair of audio input streams are converted to a pair of spectrograms by STFT. In order to construct a streaming version of source separation using MRFs, we have to decide on the temporal granularity of the source separation. In order to decrease the latency, we must use the finest granularity possible for each segment of input stream processed at a time. Here we assume that our system processes the input stream based on the size of the FFT, which is the finest granularity that can be taken with our implementation. We perform a 1024-point FFT with 50% overlap and a cosine window. The result of each 1024-point FFT will correspond to single column of the ILD matrix and the MRF generated from it.

3.2. ILD generation and MRF construction

As audio input is streamed and FFT is taken, it is appended as a new column to the ILD matrix which plots the ILD values for each (time, frequency) point. In order to take advantage of spatial locality in both the frequency and time domain, it is best to have the ILD matrix and the resulting MRF constructed to be as large as possible. At a higher level, the larger the MRF, the better the source separation will be. However, a larger MRF means longer inference runtime, and a larger memory to store the previous columns generated.

Initially, the Gaussian parameters for calculating the data cost of the MRF are approximated based on the location and the orientation of the sources relative to a pair microphones on a mobile phone. These are updated using EM once the inference is completed. The ILD generation and MRF construction is repeated with the updated parameters, and the process is repeated until convergence of the parameters. Note that a single column of the MRF does not contain much information about the approximated Gaussian mixtures. Although it will have a smoothness prior in the vertical (frequency) direction, it will not have horizontal (temporal) prior information. Instead of constructing a single column MRF, we store several previous columns of MRF from previous time samples of the input stream to build a larger MRF to take advantage of smoothness in the temporal direction. Since the 'vertical' spatial dimension of the MRF depends on the size (number of frequencies) of the spectrograms or the size of the FFT taken, it will be constant at 513. The width can be adjusted for quality and execution time trade-off.

3.3. Gibbs sampling MAP inference

The inference on the constructed MRF is analogous to how an FFT works in streaming fashion. The steady stream of newly constructed MRF columns can be understood as similar to input samples coming in for STFT block. The MRF is a sliding window that slides through streaming inputs, the MRF columns, that are constructed from previous stages with overlap. Another variable to determine is the number of Gibbs sampling inference iterations per MRF constructed. Ideally, the inference must be repeated until convergence of the resulting label values. The trade-off between the quality of the final source separation masking labels and the run-time must be considered when selecting the number of iterations to run.

3.4. Masking, updating Gaussians and output reconstruction

Once the inference is completed, the Gaussian parameters are updated using the resulting labels. The updated Gaussian parameters will be used to regenerate the MRF. Once we update the MRF with new Gaussian parameters, Gibbs sampling inference is run again to produce new labels and the Gaussian parameters. This process is repeated until it reaches the desired number of EM iterations. Once done, labels can be used to create a mask that will separate the sources from using the spectrogram created in the earlier steps. The separated sources are then reconstructed to an audible output audio stream using an ISTFT.

4. EXPERIMENTS AND RESULTS

4.1. Experimental audio setup

To test our implementation, we chose two speech signals, one female and one male, from the TIMIT corpus [14] as the input signals. These speech signals were sampled at 16,000 Hz and encoded with 16-bit PCM. For convenience, we modeled the audio input as a pair

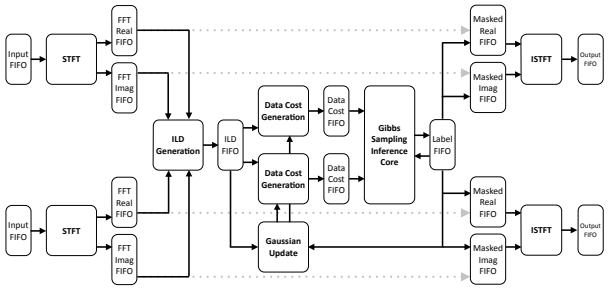

Fig. 2. Proposed Gibbs Sampling Hardware Architecture

of microphones separated by 15 cm, a distance equal to the length of a typical mobile phone. We created two convolutive mixtures with simulated room reverberations corresponding to each microphones by using the Roomsim Matlab toolbox [15]. A comprehensive mixture of a desired source and five interferences in the cellphone environment from [2] is used for the experiments. The qualitative performance of the implementation is measured by Signal-to-Distortion Ratio (SDR) [16].

4.2. Software Reference Architecture

We have implemented a software prototype of source separation using MRF via Gibbs sampling coded in C. Prior to considering the mobile use-case, we benchmarked the prototype running on an Intel Xeon X6550 2GHz. This high performance CPU took 31.695 ms to run sound source separation on 32 ms of input audio, which is the smallest granularity given 16 KHz sampling rate and 1024-point STFT. This essentially means that at least 64 ms of latency will be required. International Telecommunication Union's recommendation [17] regarding mouth-to-ear delay latency indicates that most users are "very satisfied" as long as latency does not exceed 200 ms. However, a typical latency or mouth-to-ear delay of LTE network is assumed to be approximately 160 ms [18], and delay of source separated voice on LTE network easily exceeds 200 ms. This unacceptable level of latency justifies our interest in custom hardware. Additionally, we need to understand the power consumption, using a more realistic mobile cpu. We benchmarked code on a simulated ARM Cortex-A9 CPU using GEM5 [19] and McPAT [20]. We estimate the peak power consumption of source separation on the ARM to be 3.661 Watts, which is unacceptable for mobile usage.

4.3. FPGA Implementation

Figure 2 is the top level architecture of our proposed streaming hardware. We currently have a fully synthesized Verilog version of our design running on a Convey HC-1 hybrid-FPGA platform [21]. The platform consists of Intel Xeon 5138 2.13 GHz dual-core host processor, and four Xilinx Virtex 5 (V5LX330) FPGA coprocessors running at 150 MHz each. The system has a single cache-coherent shared virtual memory which allows easier data transfers and communication between the host and the co-processors. Our design was able to fit on one Virtex 5 with resource utilization shown in Table 1.

Our system uses a 32-bit fixed-point arithmetic with 16 fractional bits (Q16). The number of fraction bits required to convert the floating point arithmetic of the source separation algorithm was determined by checking dynamic range and adequate amount of precision required to produce correct values within the system. The exception to this fixed-point number format is input and output stream values that are fractions with magnitude less than 1. To maximize

Table 1. FPGA Resource Utilization

Resource	Utilization
Slice Register	101302 / 207360 (48%)
Slice LUT	90280 / 207360 (43%)
Slice LUT FF	119300 / 207360 (57%)
BRAM	113/288 (39%)
DSP	36 / 192 (18%)

Table 2. Memory Instances

Memory Size	Number of Instances
512x32	8
1024x8	2
1024x16	2
1024x32	20
8192x1	4
8192x32	3
Total Size	207 KB

precision, input to STFT and ISTFT to output uses 31 fractional bits (Q31), with a sign bit. The iterative nature of the algorithm requires its intermediate results to be stored in buffers resulting in numerous FIFOs as shown in Figure 2. The size and number of memory instances have been listed in Table 2, with the total of 207 KB used. The hardware is fully parameterized and can be set to allow expansion to other larger inference applications. Figure 3 shows the Gibbs sampler node, which corresponds to the sampling done on a single node in the constructed MRF. A node takes labels of connected neighbors in the MRF required to calculate the smoothness cost, along with the data cost information corresponding to that node for each of the possible binary labels, 0 and 1. The sums of the costs corresponding to each label is then passed through fixed-point exponential units [22] and a divider to create a probability distribution. Each node also contains a fixed-point pseudo random number generator based on LFSR [23], which generates a random number from a uniform distribution to compare to the probability distribution found from the total costs to determine the label. The log function used to calculate ILD is implemented in fixed-point based on [24] and [25].

The pipeline structure containing the node is a module within the Gibbs sampling inference block, which contains the buffer for storing the data cost of the current EM iteration. The size of the data cost buffer is 513 frequency points by temporal (timed samples) width of the MRF, which is adjustable by a parameter. As discussed earlier, a larger MRF will produce better qualitative results with the sacrifice of the run-time. By experimenting with multiple widths with varying EM and sampling iterations and observing the resulting SDR values on our test case, we were able to decide on the EM and Gibbs sampling iterations of 4 each and the MRF block width of 8 time points on spectrogram, which corresponds to 256 ms of audio stream. Each MRF constructed will contain information from input audio up to 256 ms in the past. A correlation to input data previous to that is preserved by the labels for the MRF block that are buffered and updated each iteration as well. The number of pipeline modules can easily be selected as well.

For our implementation, the goal was to meet the mouth-to-ear delay recommendation of 200 ms of the International Telecommunication Union (ITU) [17] as we envisioned its application on a cellphone. With the FPGA running at 150 MHz, we were able to achieve 1.601 ms latency from the input to the output stream. This is one 512 sample block of input which corresponds to 32 ms of audio. Even with 160 ms LTE network delay, we are able to limit the mouth-to-ear delay under 200 ms. The resulting implementation requires 64 bits of memory data width and 1.2 GB/s of bandwidth.

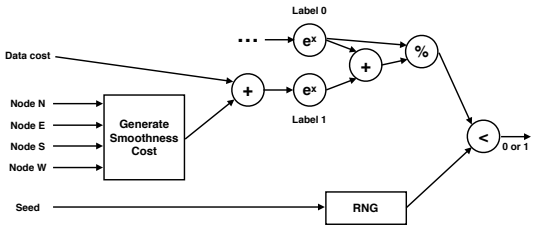

Fig. 3. A functional diagram of a Gibbs sampling node.

4.4. Preliminary ASIC Design Study

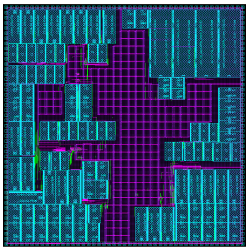

Fig. 4. The ASIC layout.

Although FPGA performance results show that source separation is feasible to run in real-time on the FPGA, it is not practical for a real appliance. We have used the IBM 45 nm library to synthesize and layout our design and find gate count, area and power estimates of a virtual ASIC prototype implementation. Our preliminary ASIC prototype shown in Figure 4, resulted in gate count of 9,126,222, area of 6.6 mm^2 and consumes 469.332 mW at 150 MHz. As expected, the overall design is attractively small, under 10 million gates. The total power consumption, however, at 469.332 mW, is a bit larger than expected - though still 8X better than our ARM reference model. We observe that SRAM memory power dominates the implementation with 403.836 mW of the 469.332 mW total. This appears to be a side-effect of our current technology library, which is not aimed at low-power mobile uses. We started with 150 MHz to synthesize and layout our design to match the operating frequency of our FPGA implementation. However, the analysis above show that we have more that enough room to tolerate longer latency while meeting the delay recommendation of ITU. When we resynthesized our design using a much lower frequency of 20 MHz, which still results in a satisfactory latency, we achieved a significantly lower power of 69.977 mW. We are currently working to resynthesize using a mobile library, and also optimize our design for lower memory usage. Nevertheless, even this early result suggests that our sampling based, MRF inference based architecture can be viable for the mobile use case.

5. CONCLUSION

We presented a novel real-time streaming architecture for sound source separation using MRFs and Gibbs sampling inference. An FPGA implementation confirms real-time feasibility, runs at 150 MHz, requires only 207 KB RAM, and achieves an SDR of 6.655 dB with 1.601 ms latency. A rough, preliminary ASIC design running at 20 MHz requires fewer than 10 million gates, with power consumption of 69.977 mW - 52X better than an ARM Cortex-A9 software reference design - but shows the need for more ASIC optimization using a lower-power technology library and design optimization for lower memory usage. Overall, we believe our design study shows that our sampling inference-based architecture can be viable for real-time audio separation for the mobile use case.

6. REFERENCES

[1] J-F Cardoso, "Blind signal separation: statistical principles," *Proceedings of the IEEE*, vol. 86, no. 10, pp. 2009–2025, 1998.

[2] M. Kim, P. Smaragdis, G.G. Ko, and R.A. Rutenbar, "Stereophonic spectrogram segmentation using markov random fields," in *International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2012.

[3] C. Charoensak and F. Sattar, "A single-chip fpga design for real-time ica-based blind source separation algorithm," in *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, may 2005, pp. 5822 – 5825 Vol. 6.

[4] Charayaphan Charoensak and Farook Sattar, "Design of low-cost fpga hardware for real-time ica-based blind source separation algorithm," *EURASIP J. Appl. Signal Process.*, vol. 2005, pp. 3076–3086, Jan. 2005.

[5] K.-K. Shyu, M.-H. Lee, Y.-T. Wu, and P.-L. Lee, "Implementation of pipelined fastica on fpga for real-time blind source separation," *Neural Networks, IEEE Transactions on*, vol. 19, no. 6, pp. 958 –970, june 2008.

[6] Vikash K. Mansinghka, Eric M. Jonas, and Joshua B. Tenenbaum, "Stochastic digital circuits for probabilistic inference," Tech. Rep., Massachussets Institute of Technology, 2008, MITCSAIL-TR-2008-069.

[7] Vikash K. Mansinghka and Eric Jonas, "Building fast bayesian computing machines out of intentionally stochastic, digital parts," *CoRR*, vol. abs/1402.4914, 2014.

[8] Siyang Wang, Xiangyu Zhang, Yuxuan Li, Ramin Bashizade, Song Yang, Chris Dwyer, and Alvin R. Lebeck, "Accelerating markov random field inference using molecular optical gibbs sampling units," in *Proceedings of the International Symposium on Computer Architecture*, 2016, ISCA '16.

[9] Dongzhen Piao, *Speeding Up Gibbs Sampling in Probabilistic Optical Flow*, PhD dissertation, Carnegie Mellon University, 2014.

[10] V. Kolmogorov, "Convergent tree-reweighted message passing for energy minimization," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 10, pp. 1568–1583, 2006.

[11] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother, "A comparative study of energy minimization methods for markov random fields with smoothness-based priors," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 30, no. 6, pp. 1068–1080, 2008.

[12] S. Geman and D. Geman, "Stochastic relaxation, gibbs distributions, and the bayesian restoration of images," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, , no. 6, pp. 721–741, 1984.

[13] Yuri Y Boykov and M-P Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in nd images," in *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*. IEEE, 2001, vol. 1, pp. 105–112.

[14] J.S. Garofolo, L.F. Lamel, W. M. Fisher, J.G. Fiscus, D.S. Pallett, N.L. Dahlgren, and Zue V., "Timit acoustic-phonetic continuous speech corpus," *Linguistic Data Consortium*, 1993.

[15] D. Campbell, K. Palomaki, and G. Brown, "A matlab simulation of" shoebox" room acoustics for use in research and teaching," *Computing and Information Systems*, vol. 9, no. 3, pp. 48, 2005.

[16] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in blind audio source separation," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 14, no. 4, pp. 1462–1469, 2006.

[17] Telecommunication Standardization Sector of ITU, "International telephone connections and circuits - general recommendations on the transmission quality for an entire international telephone connection," *ITU-T Recommendation G.114*, 2003.

[18] H. Holma and A. Toskala, *LTE for UMTS: Evolution to LTE-Advanced*, Wiley, 2011.

[19] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R Hower, Tushar Krishna, Somayeh Sardashti, et al., "The gem5 simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.

[20] Sheng Li, Jung Ho Ahn, Richard D Strong, Jay B Brockman, Dean M Tullsen, and Norman P Jouppi, "Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 2009, pp. 469–480.

[21] Convey Computer, "The convey hc-1 computer, architecture overview," nov. 2008.

[22] Jérémie Detrey and Florent de Dinechin, "Parameterized floating-point logarithm and exponential functions for fpgas," *Microprocessors and Microsystems*, vol. 31, no. 8, pp. 537–545, 2007.

[23] Thomas E. Tkacik, "A hardware random number generator," in *Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, London, UK, UK, 2003, CHES '02, pp. 450–453, Springer-Verlag.

[24] John N Mitchell, "Computer multiplication and division using binary logarithms," *IRE Transactions on Electronic Computers*, , no. 4, pp. 512–517, 1962.

[25] Roberto Gutierrez and Javier Valls, "Low cost hardware implementation of logarithm approximation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 12, pp. 2326–2330, 2011.