A Low-Power Speech Recognizer and Voice Activity Detector Using Deep Neural Networks

Michael Price, *Member, IEEE*, James Glass, *Fellow, IEEE*, and Anantha P. Chandrakasan, *Fellow, IEEE*

***Abstract*— This paper describes digital circuit architectures for automatic speech recognition (ASR) and voice activity detection (VAD) with improved accuracy, programmability, and scalability. Our ASR architecture is designed to minimize off-chip memory bandwidth, which is the main driver of system power consumption. A SIMD processor with 32 parallel execution units efficiently evaluates feed-forward deep neural networks (NNs) for ASR, limiting memory usage with a sparse quantized weight matrix format. We argue that VADs should prioritize accuracy over area and power, and introduce a VAD circuit that uses an NN to classify modulation frequency features with 22.3-μW power consumption. The 65-nm test chip is shown to perform a variety of ASR tasks in real time, with vocabularies ranging from 11 words to 145 000 words and full-chip power consumption ranging from 172 μW to 7.78 mW.**

***Index Terms*— CMOS digital integrated circuits, deep neural networks (DNNs), speech recognition, voice activity detection (VAD), weighted finite-state transducers (WFSTs).**

I. Introduction

AUTOMATIC speech recognition (ASR) figures prominently in the speech interfaces that are now used for a variety of human/machine interactions. Real-time ASR is computationally demanding, and the requirements often increase as researchers identify improved modeling techniques.

Circuit designs tailored to specific tasks reduce the overhead of general-purpose architectures, such as x86 or ARM, reducing the energy cost of those tasks by up to 100× [1]. There are also opportunities to modify algorithms and runtime parameters for favorable power/performance tradeoffs. However, the large difference in convenience and energy cost between on-chip (small) and off-chip (large) memories complicates hardware/software co-design efforts.

The speech interface on any given device is typically used infrequently, so its memory must be non-volatile. Flash and other non-volatile memories have a higher energy-per-bit than DRAM (typically 100 pJ/bit for MLC flash [2]), so there is a strong incentive to minimize memory bandwidth. A wake-up mechanism is necessary to enable and disable the interface at appropriate times; we investigate the use of voice activity detection (VAD) to disable ASR and save power during non-speech input. We characterize three VAD algorithms at SNRs from 30 to −5 dB and provide a VAD design with less than 10% equal error rate (EER) down to 1-dB SNR on a challenging data set, reducing the energy losses associated with false alarms in noisy conditions.

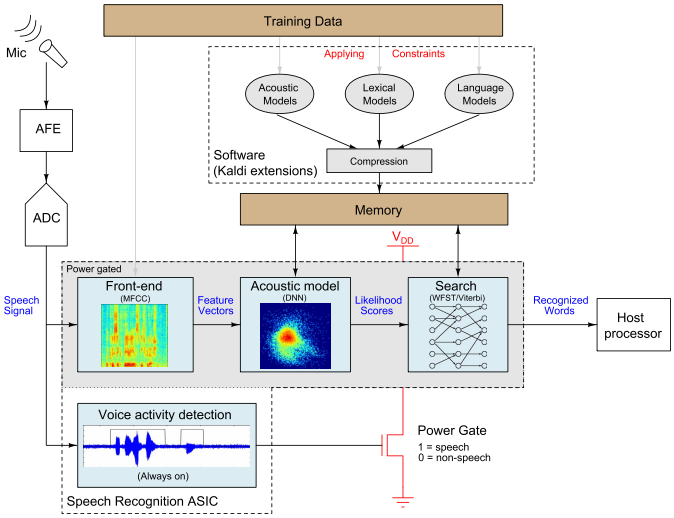

Fig. 1. Conceptual illustration of speech recognizer with voice-activated power gating. Training is performed off-line to prepare models stored in an external memory and used by search and acoustic modeling within the chip. This paper describes the digital ASIC and supporting software components surrounded by dashed boxes.

This paper demonstrates digital IC implementations of VAD and ASR, as shown in Fig. 1. Our implementation accepts audio samples from an ADC or digital microphone, labels regions of the waveform as speech/non-speech, and outputs text transcriptions by performing ASR with models stored in an external memory.

Section II provides background on the ASR problem and previous work. Sections III and IV describe the design of acoustic modeling and search subsystems that minimize memory bandwidth while preserving accuracy, programmability, and scalability. Section V describes the VAD algorithms and circuits in the context of minimizing system power, rather than the power of the VAD itself. Section VI describes the circuit-level techniques and measurement results, and Section VII concludes this paper.

II. Background

A. ASR Formulation

We provide a brief overview of the hidden Markov model (HMM) framework for ASR [3], [4]. Fig. 2 shows

Manuscript received May 3, 2017; revised July 18, 2017; accepted September 3, 2017. Date of publication October 25, 2017; date of current version December 26, 2017. This work was supported by Quanta Computer through the Qmulus Project. This paper was approved by Guest Editor Muhammad M. Khellah. *(Corresponding author: Michael Price.)*

The authors are with the Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: price@alum.mit.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSSC.2017.2752838

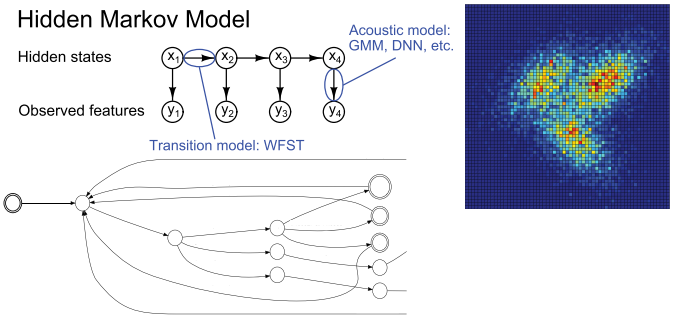

Fig. 2. HMM formulation of ASR. Searching for word hypotheses requires two types of statistical models: a WFST for transitions between hidden states, and an acoustic model for the observations (feature vectors) conditioned on each hidden state.

TABLE I
DATA SETS USED FOR ASR EXPERIMENTS [7]–[10]. THE LM PERPLEXITY CAN BE INTERPRETED AS A BRANCHING FACTOR, WITH HIGHER PERPLEXITY INDICATING FEWER CONSTRAINTS FOR SEARCH [11]

Name	Vocab.	Channel	LM Perplexity
Digits (TIDIGITS)	11	Telephone	10.8
Weather (Jupiter)	2k	Telephone	8.58
Food diary	7k	Wideband	23.7
News 1 (WSJ eval92-5k)	5k	Wideband	65.6
News 2 (WSJ dev93)	145k	Wideband	166

a speech HMM, which expresses the relationship between hidden variables x_t and observed variables y_t. Viterbi search is used to "decode" the most likely word sequence (based on x_t) from the feature vectors y_t. The feature vectors are extracted from the audio stream using a representation, such as mel-frequency cepstral coefficients (MFCCs) [5].

The HMM factorizes the joint probability of all observed and hidden variables as

$$p(x, y) = p(x_0) \prod_{t=0}^{T-1} p(x_{t+1}|x_t)\, p(y_t|x_t).$$

In this formulation, x_t is a discrete index to a state in a weighted finite-state transducer (WFST) [6]; y_t is a real-valued vector that is observed before decoding. The forward update of Viterbi search approximates the likelihood of all reachable states at the next time step

$$\begin{aligned} p(x_{t+1}) &= \sum_{x_t} p(x_t)\; p(x_{t+1}|x_t)\; p(y_{t+1}|x_{t+1}) \\ &\approx \max_{x_t}\; p(x_t)\; p(x_{t+1}|x_t)\; p(y_{t+1}|x_{t+1}). \end{aligned}$$

We follow the standard practice of training different recognizers for different tasks, each with its own vocabulary and acoustic properties. Some of the tasks we studied are listed in Table I. Having a range of tasks lets us evaluate the scalability of the ASR implementation.

B. Previous Work

Since the 1980s, there have been sporadic efforts to exploit special-purpose hardware for speech applications; please consult [4] for an overview. Diverse priorities and baselines have led to FPGA and ASIC demonstrations of ASR, whether real time or much faster [12], [13]. The architectures and frameworks used for ASR are still in flux.

Deep neural networks (DNNs) have become popular for ASR due to their improved accuracy [14]. The computer science community continues to develop new types of NNs and identify combinations that work well [15]. The circuits community has studied DNNs and developed efficient implementations, primarily for computer vision [16]. Other recent work has covered architectural exploration [17] and the application of code generation tools [18].

The wide range of algorithms available for VAD also provides opportunities to apply special-purpose architectures. Power can be minimized through a variety of techniques, whether minimum-energy operation in deeply scaled CMOS [19], or mixed-signal design with adaptive feature granularity [20]. These efforts have brought VAD power consumption down to a few μW.

C. Contributions

This paper builds on previous efforts in two major areas: 1) circuit implementations supporting modern algorithms and frameworks and 2) pushing down power consumption from a system-level perspective. The key results were presented in [21]; more details on all aspects of this paper are provided in [22]. Many degrees of freedom and their impacts are considered in pursuit of more accurate, programmable, and scalable embedded ASR capabilities.

III. ACOUSTIC MODELING

A. Choosing a Modeling Framework

HMM-based ASR can be broken down into feature extraction (front end), acoustic modeling, and search. The acoustic model has to evaluate the likelihood of input features $\mathbf{y}_t$ with respect to a set of distributions $p(\mathbf{y}|i)$, where i is the index of an acoustic state or senone. The features are typically 10–40 dimensions. While feature extraction is not a performance bottleneck for ASR, we implemented a configurable MFCC accelerator to avoid the overhead of adding a processor.

Experiments performed in [23] compared three acoustic modeling frameworks in the context of minimizing memory bandwidth: Gaussian mixture model (GMM), subspace GMM, and DNN. These comparisons made assumptions about the hardware architecture that would be used in each case. Overall, the DNNs offered the best tradeoff between bandwidth and accuracy. Bandwidth reductions can be achieved by using smaller networks (for example, 256–512 hidden nodes per layer) and keeping the width constant across layers to maximize the utilization of on-chip memory. These dimensionality changes are complemented by the scalar quantization of weights and the use of sparse weight matrices, both of which have minor impacts on accuracy. Our models are still large enough that the weight and bias parameters for each layer cannot be cached in on-chip memory.

B. DNN Accelerator Architecture

We evaluate feed-forward DNNs using a fixed-function SIMD architecture shown in Fig. 3. This architecture computes

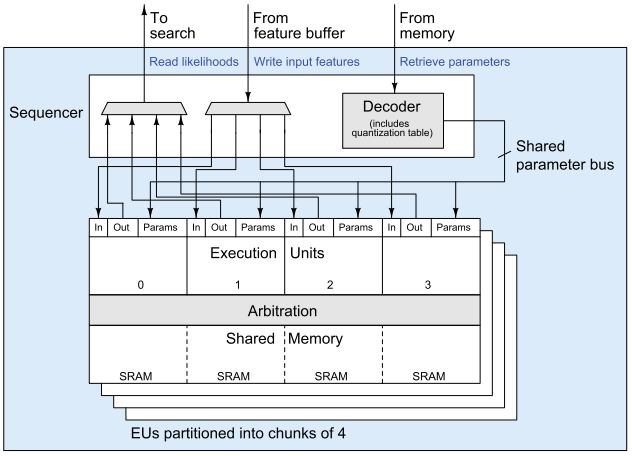

Fig. 3. Block diagram of SIMD NN evaluator, relying on a shared sequencer and 32 EUs. External memory access is read-only due to internal storage of activations.

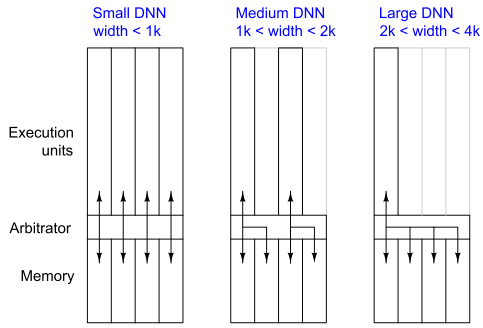

Fig. 4. EUs and local memory are grouped into chunks that can be reconfigured for different network sizes.

likelihood results for multiple frames simultaneously using the same model. The sequencer decodes a compressed parameter stream arriving from off-chip memory, and sends weight and bias coefficients to the execution units (EUs) where they are used and discarded. Each EU has local memory for storing the feature vector, intermediate results (activations), and log-likelihood outputs for one frame. This means the network parameters are the only data fetched from off-chip memory, and that the memory access is read-only.

Increasing the size of local memories would allow the use of wider networks (or more output targets), but it would also take up area that could otherwise be used for more EUs. Furthermore, the SRAMs have some area overhead from sense amplifiers and I/O logic, meaning that the area is not linearly proportional to depth. As a compromise, we group EUs into chunks, which each have access to a group of SRAMs. For small networks, each EU connects directly to one SRAM. For a network that would overflow one SRAM, every other EU is disabled, but the active EUs use two adjacent SRAMs. For a network that would overflow two SRAMs, three out of every four EUs are disabled, and the active EUs use four adjacent SRAMs. These options are shown in Fig. 4. Our implementation relied on clock gating to disable EUs, since leakage was small enough (on the order of 5 μA per EU) that power gating would provide little advantage.

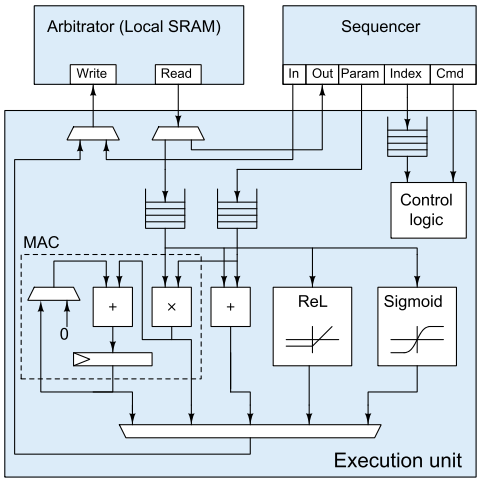

Fig. 5. EU block diagram (pipelining, addresses, and control signals not shown).

We opted to store intermediate results in 24-bit format, provide 32 EUs in eight chunks, and support up to 1024 nodes per layer without penalty (4096 nodes maximum). This arrangement requires 1.5 Mb of SRAM. Having 32 EUs means that small networks only need to be evaluated 3.125 times per second at 100 frames/s, reducing both memory bandwidth and clock frequency.

C. Sequencer and Execution Unit Design

The interface between the sequencer and the EU is a command FIFO, as well as a data FIFO that supplies (index, value) pairs. These interfaces are broadcasted to all active EUs. The sequencer interprets the NN data structure as it is streamed in, 1 byte per clock cycle. Before providing packed weight coefficients, each layer specifies a quantization lookup table (loaded into SRAM) and optionally a per-row binary sparsity pattern stored as its run length encoding (RLE). In the dense mode, the column index counter increases serially; in the sparse mode, the counter increases according to the RLE output.

The EU is shown in Fig. 5. A 24-bit multiply/accumulate is used to avoid extra reads/writes during the affine transformation. The output is saved to SRAM and can be fed through a sigmoid or rectified linear unit (ReLU) non-linearity afterward. The ReLU should be used with caution, because it does not suppress errors caused by quantized weights. We approximate the sigmoid operation $\sigma(x) = (1/1 + e^{-x})$ with a piecewise fifth-order fit evaluated using Horner's method.

In addition to affine layers, we support elementwise addition and multiplication operations. These are useful for feature transforms not handled by the front end shown in Fig. 1, such as cepstral mean and variance normalization.

IV. Viterbi Beam Search

The NN acoustic models provide a ripe opportunity for hardware acceleration due to their regular structure. Accelerating search requires a more complex architecture, but we believe it is equally important for energy efficiency. Our application scenarios require up to 1M hypotheses to be expanded

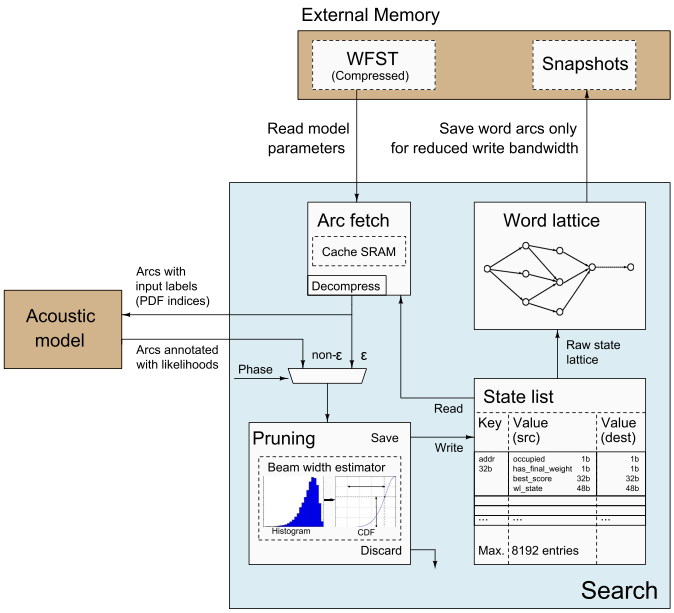

Fig. 6. Improved search architecture including WL and merged state lists.

and evaluated per second, which would require GHz clock frequencies on a general-purpose processor.

The Viterbi search algorithm creates hypotheses about the speech production process and propagates them forward in time, using observation likelihoods from the acoustic model. Pruning is used to limit the number of hypotheses and speed-up decoding. Through software studies [23], we developed several algorithmic techniques to improve the memory efficiency of Viterbi search algorithm operating on WFST transition models. Fig. 6 shows an architecture supporting these techniques: WFST compression and caching, predictive beamwidth control, a merged two-frame state list, and a word lattice (WL). This section provides details on the design of the state list and WFST fetch components. For information about the pruning and WL algorithms, please see [22].

A. State List

The search architecture, including state list and WFST data structures, has been designed to preserve memory locality. This is not only a benefit to energy efficiency, but a major convenience in a special-purpose digital design. It allows us to avoid having a shared memory hierarchy that connects to every module. Instead, each module has local memory for just the information it needs. There are only two external memory dependencies in the search subsystem: WFST fetch and the WL.

Fig. 7 shows the data structures stored in the state list and passed through each stage of the search pipeline.

1) *State List:* Every hypothesis refers to a particular WFST state and its log likelihood, along with a (state, time) pair for the most recent word label leading up to the current state. We call this the WL state.
2) *WFST Request:* To expand a state in the WFST, we need to know its address and length (in bytes). In response, we receive a sequence of arcs. Each arc that we expand has a destination state, input and output labels, and weight. After this, we no longer need to know the source state; the WL state is sufficient for recovering word sequences.
3) *Scoring:* Before search is run, the acoustic model has computed a likelihood for each senone (input label). For each unscored arc, we retrieve the appropriate likelihood and combine it with the source state score and arc weight to get the overall likelihood of the destination state. After this, we no longer need the input label.
4) *Pruning:* The scored arc has all the information necessary to save a new hypothesis in the state list, provided that its score exceeds the pruning cutoff (and that of any existing hypothesis at the same state). If the scored arc has a word label (i.e., non-ϵ output label), it will also be sent to the WL.

Bit fields have been sized conservatively to allow WFST sizes up to 4 GB, vocabularies up to 16M words, and acoustic models with up to 64k senones. These limits could be reduced to save area in a more restrictive implementation. The implementation uses hash tables similar to [4] that are dynamically resized to keep load factor between $1/16$ and $3/4$.

The state list is designed to perform a sequential scan (reading frame t states) while also performing reads, inserts, and updates of frame $t+1$ states. If it detects that the hash table has expanded, the scan must be restarted in order to ensure that all frame t states are read. It supports a "prune in place" operation that deletes all states from frame $t+1$ whose likelihoods fall below the specified cutoff; this is needed to prevent overflow. After each forward pass is completed, it performs an "advance" operation that moves information from frame $t+1$ fields to frame t fields, because t is being incremented. States that were present in frame t but not $t+1$ are deleted.

The state list is sized to trade between area and accuracy (high capacity is needed for larger recognition tasks). Our implementation has a capacity of 8192 states (223 bits per entry, or 1.8 Mb total). Merging the two frames and using single-ported hash tables allowed us to double capacity relative to [4] while using 11% less area.

B. WFST

The WFST encapsulates the time-varying aspects of the HMM. There are no simple rules for computing the size of the WFST; it is generally proportional to the number of N-grams in the language model (LM), rather than the size of the vocabulary.

To simplify interfaces, the WFST is treated as a read-only key/value store. Fig. 8 shows our WFST fetch architecture. Compression and caching techniques reduce the memory bandwidth of the millions of read queries generated while decoding each utterance.

As shown in [23], this search architecture delivers accuracy comparable to Kaldi. Amortized search throughput is approximately one hypothesis per 40 clock cycles.

V. Voice Activity Detection

A. Accuracy Influences Power Consumption

Accuracy (rather than power) of the VAD is the main driver of system power when the VAD is used for power gating.

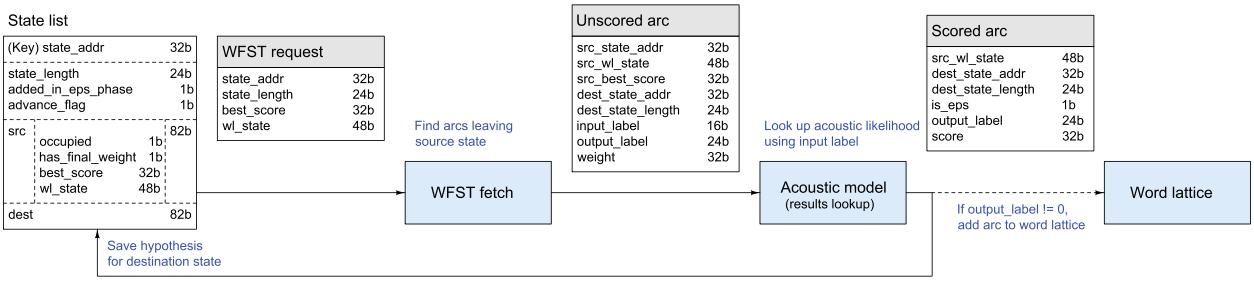

Fig. 7. Data fields for state list entries and in-flight hypotheses. The architecture is designed to preserve memory locality.

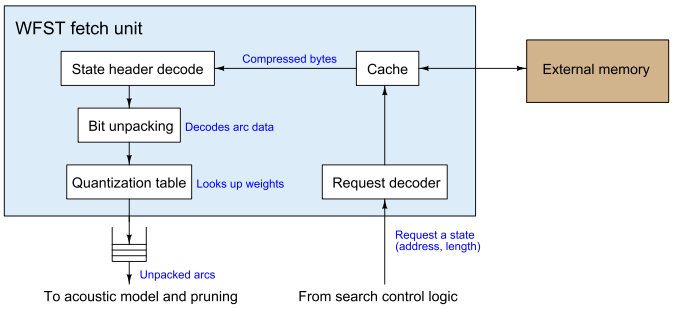

Fig. 8. WFST fetch architecture with caching and model compression.

A simple model for time-averaged system power is

$$P_{\text{avg}} = p_{\text{VAD}} + [(1 - p_M)D + p_{\text{FA}}(1 - D)]\, p_{\text{downstream}}$$

where D is the duty cycle of speech, p_M is the probability of misses, and p_{FA} is the probability of false alarms. The coefficient $(1 - p_M)D + p_{\text{FA}}(1 - D)$ reflects how often the downstream system is enabled—a duty cycle that can be far higher than D, if p_{FA} is significant. Differences in this contribution from the downstream system can far exceed the differences in power consumption between different VAD implementations.

A variety of research groups have presented hardware VADs, some with impressively low power consumption [19], [20], but there is a risk of reduced accuracy in low-SNR and non-stationary noise conditions. Our test chip allows direct comparison of three VAD algorithms that make tradeoffs between accuracy, power, and area. The VAD buffers the input samples (as the ASR is powered down until speech is detected), and can perform downsampling if the sample rates used by VAD and ASR differ. The latency of the VAD algorithm and any post-filtering of VAD decisions typically range from 100 to 500 ms. When ASR starts up, the recognizer must process the buffered audio, but it can be run faster than real time (by adjusting the clock frequency or beamwidth) as necessary to "catch up" to the incoming samples. Other ASR start-up overheads (including configuration from the external memory) are comparatively small, on the order of 1 ms.

B. Three VAD Algorithms

The block diagram of our VAD is shown in Fig. 9. Each algorithm works as follows.

1) Energy-Based: Thresholding a signal based on its short-term energy is a workable VAD approach in low-noise conditions. The energy-based (EB) algorithm we implemented tracks changes in energy over time that distinguish speech from many non-speech sounds [24]. These energy differences are weighted by an estimate of SNR, reflecting the decrease in confidence when SNR is low.[1] We compute the SNR by estimating noise energy as the minimum frame energy over the last 100 frames, plus 3 dB. To obtain clean decisions, the threshold is applied to a moving average of the scaled energy differences D_t over the previous 21 frames

$$E_t = 10 \log_{10} \sum_{n=0}^{N-1} x^2[kt + n]$$

$$\text{SNR}_{\text{post}} = \max\left(0,\, E_t - \left(\min_{t'=0}^{T} E_{t-t'} + 3\right)\right)$$

$$D_t = \sqrt{|E_t - E_{t-1}|\text{SNR}_{\text{post}}}.$$

2) Harmonicity: A simple but robust VAD technique is to threshold audio frames based on their harmonicity (HM), or the amplitude ratio between the periodic and non-periodic components of the signal. Not all speech is periodic: unvoiced phonemes (some fricatives, affricates, and stop consonants) consist of turbulence and silence. However, every utterance (except perhaps whispering) contains a mixture of unvoiced and voiced sounds.

Boersma [25] provided the mathematical underpinnings for accurate estimates of HM. Dividing out the autocorrelation of the fast Fourier transform (FFT) window gives us a normalized autocorrelation that allows accurate pitch extraction. In our experience, a threshold of 6–8 dB detects speech reliably

$$r'_{xx}[t] = \frac{r_{xx[t]}}{r_{ww}[t]}$$

$$H = 10 \log_{10} \max_{T_1 < t < T_2} \frac{r'_{xx}[t]}{r'_{xx}[0] - r'_{xx}[t]}.$$

3) Modulation Frequencies: The EB and HM algorithms are rule-based; it is difficult to design rules that cover every scenario, and these schemes fail to reject certain non-speech sounds (such as railroad noise or background music). These challenges motivate a supervised learning approach: transforming the signal into an appropriate representation and

[1] Use of the square root to compress energy differences was suggested in an unpublished successor to [24].

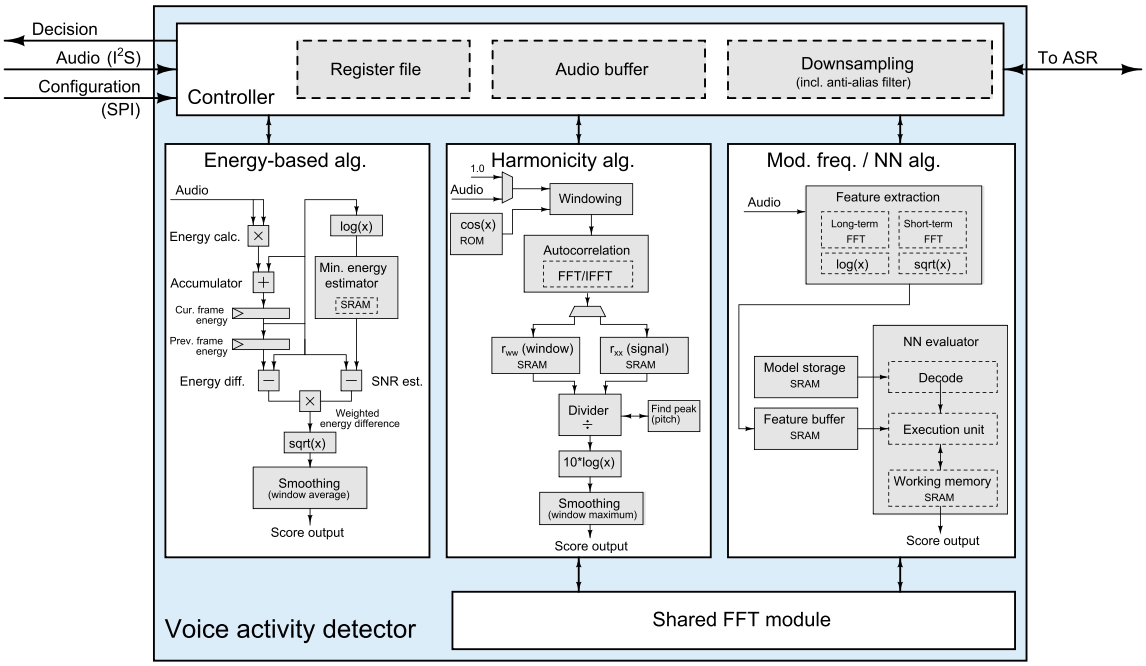

Fig. 9. Block diagram of VAD that can selectively enable EB, HM, and MF algorithms. Score computation pipelines for each module are shown.

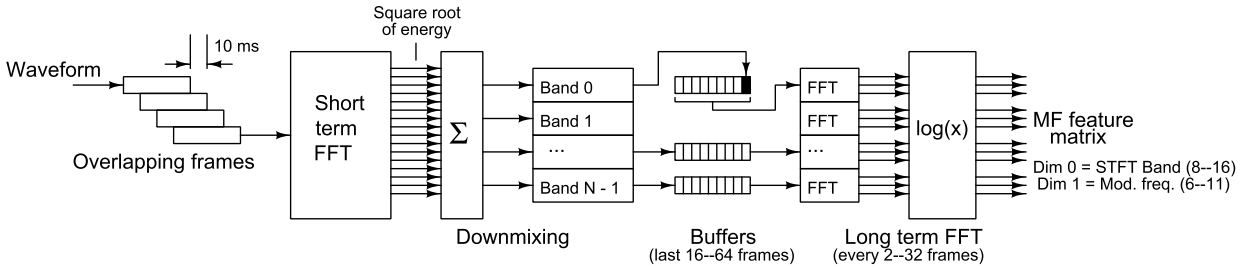

Fig. 10. MF feature extraction: the short-term FFT bins signal energy by band, and the long-term FFT bins signal energy by MF.

classifying based on labeled training data. However, representations that work well for speech recognition are not ideal for VAD. MFCCs capture the short-term spectral characteristics of a sound, which can guide HMM search and, over time, discriminate between words. Given the VAD's lack of temporal context, the brute force solution would be to concatenate several MFCCs, an approach that has been used for low-power keyword detection [26].

The modulation frequencies (MFs) representation is shown in Fig. 10. The history of energy within each frequency band is analyzed by a long-term FFT to obtain a power spectrum of temporal modulation. This can be viewed as a matrix where the rows represent different audio bands, and the columns represent MFs. MF features are very effective at resolving common temporal characteristics of speech, even at 0-dB SNR (see Fig. 11). In [27] (on which this paper is based), MF features were fed into a group of SVM classifiers whose outputs were fused. Instead, we use an NN classifier, which performs nearly as well with many fewer parameters. The resulting networks are much smaller than those used for ASR: two to four layers of 32–64 nodes, small enough to fit entirely in our 24-kB on-chip memory.

Each MF feature is sent to the NN evaluator for classification as speech/non-speech; the difference between the two last layer outputs is used as a score. The model is trained using Kaldi on disjoint speech and non-speech segments (no overlap), which improves accuracy even though the audio stream contains overlapped frames.

C. Comparisons

We constructed two sets of performance tests for comparing the VADs' detection performance. Fig. 12 (left) shows the receiver operating characteristic (ROC) of each algorithm at 10-dB average SNR on the Aurora2 [28] task.[2] In Fig. 12 (right), we use our simple system model to transform the ROC into a plot of power consumption versus miss rate. At low miss rates (which are desirable for good ASR performance), average power is dominated by the downstream portion of the system. This is true even if we make conservative (high) estimates for the power used by the VADs, and an aggressive (low) estimate of the power used by the downstream system.

A more challenging and realistic scenario was constructed using noise waveforms collected for a robotic forklift

[2]We count frame-level errors by comparing with a reference segmentation (CTM) from a speech recognizer. This method results in higher error rates than other studies, but is necessary in order to estimate power consumption. False alarm and miss rates up to about 5% can be explained by minor differences in the start and end times of speech segments, even if they are detected correctly.

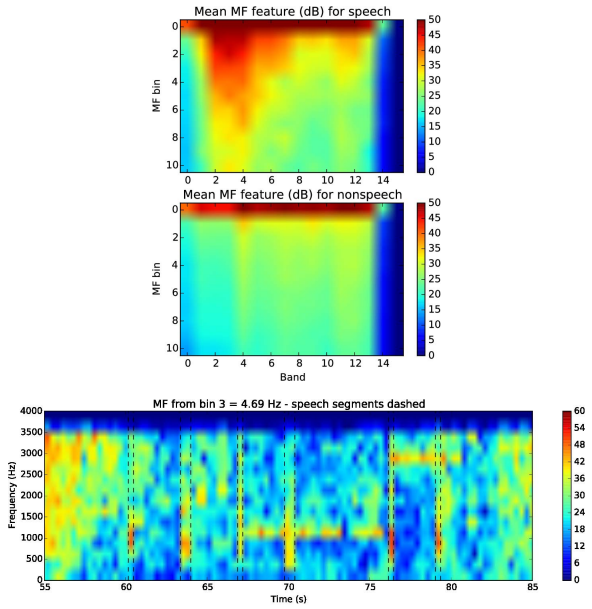

Fig. 11. Examples demonstrating the separability of MF features in noisy conditions: mean MF matrix for speech and non-speech (top), and 4-Hz MF spectra (bottom), at 0-dB SNR. Dashed vertical lines: speech segments.

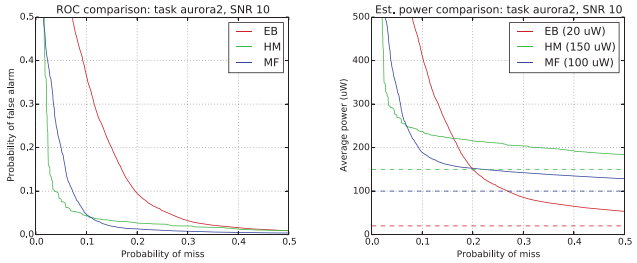

Fig. 12. Left: ROC of each VAD on Aurora2 task at 10-dB average SNR. Right: estimated system power derived from ROC at 5% speech duty cycle. This plot uses early (conservative) estimates of VAD power for each algorithm, as shown in the inset and dashed lines, and 1-mW downstream system power.

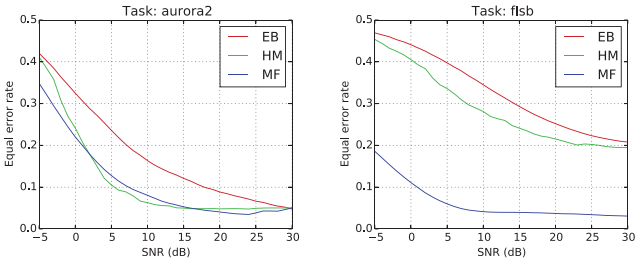

Fig. 13. EER versus average SNR on Aurora2 task (left) and Forklift/Switchboard task (right).

project [29], and conversational speech waveforms from the Switchboard data set [30]. Fig. 13 shows the EERs over a range of average SNR from +30 to −5 dB on both tasks. On the Aurora2 task, the HM and MF algorithms offer similar performance, both handling 5–10-dB lower SNR than the EB algorithm. On the Forklift/Switchboard task, the MF performance improves significantly, with little change in EER down to 5-dB SNR.

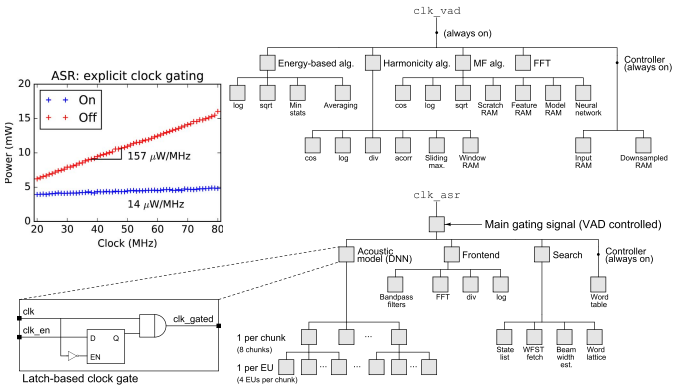

Fig. 14. Clock gating hierarchy: VAD clock domain (top) and ASR clock domain (bottom). The impacts of explicit clock gating are shown at left.

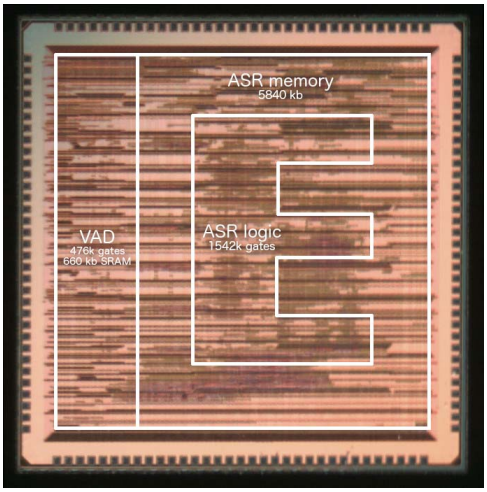

Fig. 15. Die photograph of ASR/VAD chip.

VI. IC Implementation

In this section, we present a test chip with the ASR and VAD functionality described earlier. The chip was designed for the TSMC 65-nm low-power logic process, using third-party IP libraries for standard cells, I/O, and memory compilers.

A. Clock Gating: Explicit and Implicit

To save power in the clock tree, we inserted latch-based clock gates at 76 locations in the design as shown in Fig. 14. Modules that employ clock gating generate a synchronous enable signal for each of their child clock domains. The clock gating is used both at high levels (enabling the front end, acoustic model, and search only when needed) and lower levels (enabling a subset of NN EUs, or arithmetic blocks in the VAD). These clock gates complement those that are automatically inserted at the lower levels of the clock tree by the synthesis tool.

B. Test Procedures

The test chip is shown in Fig. 15. Outer dimensions are 3.63×3.63 mm^2, with a core size of 3.1×3.1 mm^2. There are 132 I/O and supply pads, intended for wire bonding in an 88-pin QFN package with two pads per supply pin and all grounds downbonded to an exposed pad.

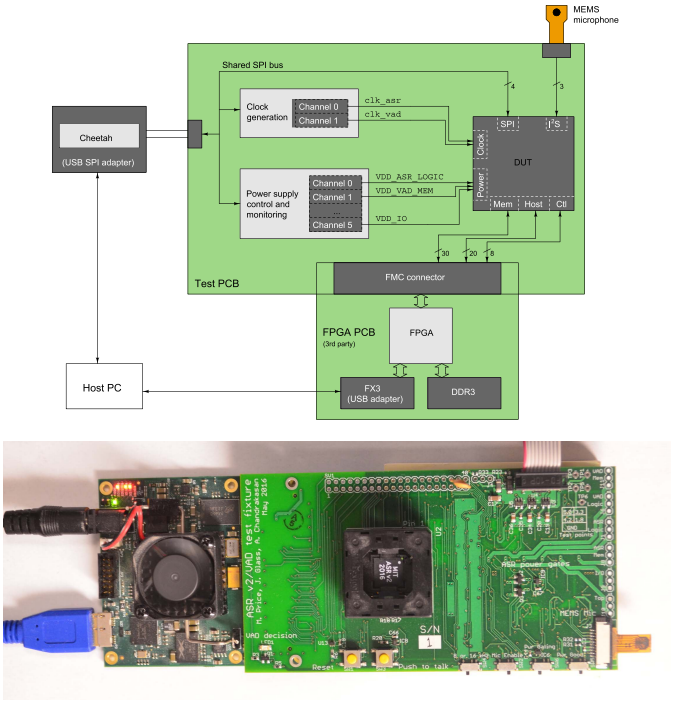

Fig. 16. Block diagram (top) and photograph (bottom) of test PCB stack integrating power supplies and functional test interfaces.

Our test setup is shown in Fig. 16. Through an SPI interface, the host computer has control over clocks and power supplies, and can measure the current draw of each supply in real time. A MEMS microphone connects to the chip directly via the I^2S protocol. An FPGA is used to translate the chip's generic host and memory interfaces to USB and DDR3, respectively.

C. Measured Performance

The chip performs VAD from audio samples to time-labeled decisions, and all stages of ASR transcription from audio samples to text. ASR is functional with logic supply voltages from 0.60 V (10.2 MHz) to 1.20 V (86.8 MHz), and VAD is functional from 0.50 V (1.68 MHz) to 0.90 V (47.8 MHz). For best performance, we operate the SRAM at 0.15–0.20 V above the logic supply (up to a limit of 1.20 V). Explicit clock gating resulted in a 30%–40% reduction in power when the core was saturated, and reduced ASR clock tree overhead from 157 to 14 pJ per cycle when running faster than real time.

An example time series of current measurements is shown in Fig. 17. The large pulses are NN evaluations (every 32 frames); between those pulses, search proceeds at a varying rate, since it evaluates different numbers of hypotheses for each frame. The search workload is more memory-intensive (blue trace) due to the large on-chip data structures, whereas acoustic modeling is more logic-intensive (red trace). Occasional WL snapshots cause peaks in I/O current as the data structure is written to external memory.

The acoustic model requires 16–56 pJ per non-zero neuron weight, depending on supply voltages. Search efficiency varies from 2.5 to 6.3 nJ per hypothesis, compared with 16 nJ in [4]. We expect these results to improve with CMOS scaling, with the caveat that search efficiency depends mostly on SRAM and acoustic model efficiency depends mostly on standard cells.

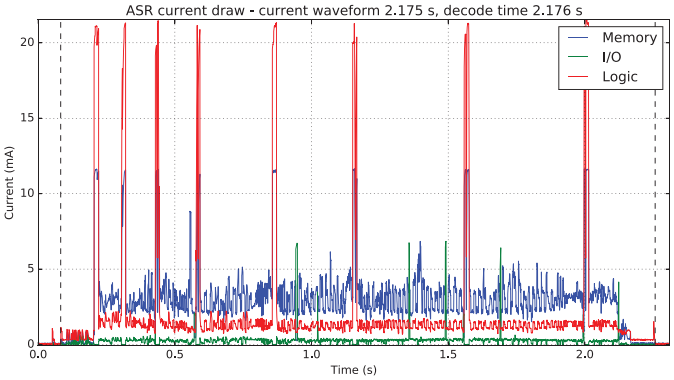

Fig. 17. Current measurement of each power supply during ASR decoding of an utterance from the food diary data set. Variations in search workload are visible as different lengths of decoding time between acoustic model evaluations (periodic spikes), which occur after every 32 frames (320 ms) of input.

TABLE II

SUMMARY OF MEASURED ASR AND VAD PERFORMANCE. ASR POWER SCALES BY 45× BETWEEN TASKS; VAD POWER IS RELATIVELY SIMILAR DESPITE LARGE DIFFERENCES IN NOISE ROBUSTNESS BETWEEN THE THREE ALGORITHMS

Automatic speech recognition (ASR)

Task	Vocab.	Clock	Mem. BW	WER	Power
Digits	11	3 MHz	0.11 MB/s	1.65%	172 μW
Weather	2k	23 MHz	10.1 MB/s	4.38%	4.70 mW
Food diary	7k	46 MHz	9.02 MB/s	8.57%	4.67 mW
News 1	5k	15 MHz	4.84 MB/s	3.12%	1.78 mW
News 2	145k	40 MHz	15.0 MB/s	8.78%	7.78 mW

Voice activity detection (VAD)

	SNR for 10% frame level EER			Power
Algorithm	White noise	Aurora2	Forklift	
Energy-based	−1 dB	18 dB	Fail	8.5 μW
Harmonicity	<-5 dB	5 dB	Fail	24.4 μW
Modulation frequencies	−2 dB	7 dB	1 dB	22.3 μW

A summary of ASR and VAD performance is shown in Table II. A variety of ASR tasks, with vocabularies ranging from 11 words to 145k words, can be run in real time on this chip. Different models and configuration parameters, such as the search beam width, can be selected at runtime, allowing host software to trade between accuracy and power consumption. With the representative configurations that we used, core power scales by 45× from the easiest to the hardest task, and memory bandwidth scales by 136×. On the Wall Street Journal eval92-5k task that was demonstrated by [4], we achieved 4.1× fewer word errors (3.12% versus 13.0%), 3.3× lower core power (1.78 versus 6.0 mW), and 12.7× lower memory bandwidth (4.84 versus 61.6 MB/s).

The power breakdown between search and acoustic modeling varies by task; neither appears as a consistent bottleneck. To illustrate this, we used our efficiency estimates (in conjunction with workload statistics collected by the chip) to roughly identify the proportions of core power used by each subsystem. On the weather query task, with a relatively low search workload (301k hypotheses/s) and moderately sized acoustic model (six hidden layers of 512 nodes each), we estimate that the acoustic model is using about 4.1× as much power as search. On the food diary task, with a search workload of 844k hypotheses/s and a smaller

acoustic model (six layers of 256 nodes), the situation is reversed: the acoustic model uses about 0.35× as much power as search. Hence, it is important to optimize both portions of the decoder in hardware designs and, ideally, consider these issues when training models.

The HM algorithm for VAD has slightly higher power consumption (24.4 μW) than the MF algorithm (22.3 μW). This is because the HM algorithm requires running two FFTs per frame in order to compute autocorrelation. While the MF algorithm is more complex, the long-term Fourier transform (LTFT) and NN operations are not as energy-intensive as the more frequent short-term Fourier transform (STFT). In our opinion, the additional power required by the MF algorithm is worthwhile in exchange for improved accuracy and noise robustness over EB approaches.

VII. Conclusion

A. Summary

In a portable or embedded system, the entire speech decoding chain from audio to text can be performed on a single chip without a general-purpose processor. To achieve good energy efficiency and scalability, our ASR approach draws techniques from the disciplines of algorithms, architectures, and circuits:

1) *Algorithm Techniques*: These techniques include acoustic model selection, quantization and sparse weight storage, the WL, and predictive beam search.
2) *Architectural Techniques*: These techniques include WFST caching, resizable hash tables, parallel NN evaluation, and a two-frame state list.
3) *Circuit Techniques*: These techniques include multi-voltage design, explicit clock gating, and variation-tolerant timing.

We explored the idea of using a VAD to power gate the ASR, and designed interfaces allowing the subsystems to work together on a single chip. We evaluated VAD performance in a broader context—considering its impact on system power and user experience—with more realistic (non-stationary) noise environments than previous hardware VAD studies.

B. Future Work

We envision a single-chip solution to a variety of speech processing problems from question–answering to translation, with enough scalability to be reused in applications from wearables to servers. Realizing this vision requires expanding and refining the set of algorithms we implemented and integrating additional components on-chip [22]. Future work should also leverage technological changes, such as die stacking and improved non-volatile memories, which could change the balance of design priorities (e.g., memory bandwidth versus logic complexity).

Meanwhile, advances in deep learning have raised the possibility that the HMM framework for ASR can be replaced by a single end-to-end model. Recognizers would not need a specialized front end, and could markedly simplify the Viterbi search architecture. Most resources would be devoted to NN evaluation, which could present an even larger memory bottleneck than we observed with HMM recognizers. This line of research should be watched carefully.

Acknowledgment

The authors would like to thank the TSMC University Shuttle Program for providing chip fabrication; Xilinx for providing FPGA boards; and Mentor Graphics, Synopsys, and Cadence for providing CAD tools.

References

[1] E. C. Lin, K. Yu, R. A. Rutenbar, and T. Chen, "In silico vox: Towards speech recognition in silicon," in *Proc. HOTCHIPS*, Aug. 2006, pp. 1–27.

[2] L. M. Grupp *et al.*, "Characterizing flash memory: Anomalies, observations, and applications," in *Proc. 42nd Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Dec. 2009, pp. 24–33.

[3] F. Jelinek, *Statistical Methods for Speech Recognition*. Cambridge, MA, USA: MIT Press, 1997.

[4] M. Price, J. Glass, and A. P. Chandrakasan, "A 6 mW, 5,000-word real-time speech recognizer using WFST models," *IEEE J. Solid-State Circuits*, vol. 50, no. 1, pp. 102–112, Jan. 2015.

[5] S. B. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-28, no. 4, pp. 357–366, Aug. 1980.

[6] M. Mohri, F. Pereira, and M. Riley, "Speech recognition with weighted finite-state transducers," in *Springer Handbook of Speech Processing* (Speech Communication). Berlin, Germany: Springer-Verlag, 2008.

[7] R. Leonard, "A database for speaker-independent digit recognition," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, vol. 9. Mar. 1984, pp. 328–331.

[8] V. Zue *et al.*, "JUPlTER: A telephone-based conversational interface for weather information," *IEEE Trans. Speech Audio Process.*, vol. 8, no. 1, pp. 85–96, Jan. 2000.

[9] M. Korpusik, C. Huang, M. Price, and J. Glass, "Distributional semantics for understanding spoken meal descriptions," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2016, pp. 6070–6074.

[10] D. B. Paul and J. M. Baker, "The design for the Wall Street Journal-based CSR corpus," in *Proc. Workshop Speech Natural Lang. (HLT)*, Stroudsburg, PA, USA, 1992, pp. 357–362. [Online]. Available: http://dx.doi.org/10.3115/1075527.1075614

[11] X. Huang, A. Acero, H.-W. Hon, and R. Reddy, *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. Englewood Cliffs, NJ, USA: Prentice-Hall, 2001.

[12] J. Johnston and R. Rutenbar, "A high-rate, low-power, ASIC speech decoder using finite state transducers," in *Proc. IEEE 23rd Int. Conf. Appl.-Specific Syst., Archit. Process. (ASAP)*, Jul. 2012, pp. 77–85.

[13] G. He, Y. Miyamoto, K. Matsuda, S. Izumi, H. Kawaguchi, and M. Yoshimoto, "A 40-NM 54-MW 3×-real-time VLSI processor for 60-kWord continuous speech recognition," in *Proc. IEEE Workshop Signal Process. Syst. (SiPS)*, Oct. 2013, pp. 147–152.

[14] G. Hinton *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.

[15] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2015, pp. 4580–4584.

[16] Y. H. Chen, T. Krishna, J. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Jan. 2016, pp. 262–263.

[17] O. A. Bapat, P. D. Franzon, and R. M. Fastow, "A generic and scalable architecture for a large acoustic model and large vocabulary speech recognition accelerator using logic on memory," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 12, pp. 2701–2712, Dec. 2014.

[18] Y. Lee, D. Sheffield, A. Waterman, M. Anderson, K. Keutzer, and K. Asanovic, "Measuring the gap between programmable and fixed-function accelerators: A case study on speech recognition," in *Proc. IEEE Hot Chips 25 Symp.*, Aug. 2013, pp. 1–2.

[19] A. Raychowdhury, C. Tokunaga, W. Beltman, M. Deisher, J. W. Tschanz, and V. De, "A 2.3 nJ/frame voice activity detector-based audio front-end for context-aware system-on-chip applications in 32-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 48, no. 8, pp. 1963–1969, Aug. 2013.

[20] K. M. H. Badami, S. Lauwereins, W. Meert, and M. Verhelst, "A 90 nm CMOS, 6μW power-proportional acoustic sensing frontend for voice activity detection," *IEEE J. Solid-State Circuits*, vol. 51, no. 1, pp. 291–302, Jan. 2016.
[21] M. Price, J. Glass, and A. P. Chandrakasan, "A scalable speech recognizer with deep-neural-network acoustic models and voice-activated power gating," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2017, pp. 244–245.
[22] M. Price, "Energy-scalable speech recognition circuits," Ph.D. dissertation, Dept. Elect. Eng. Comput. Sci., Massachusetts Inst. Technol., Cambridge, MA, USA, 2016.
[23] M. Price, A. Chandrakasan, and J. R. Glass, "Memory-efficient modeling and search techniques for hardware ASR decoders," in *Proc. INTERSPEECH*, 2016, pp. 1893–1897.
[24] Z.-H. Tan and B. Lindberg, "Low-complexity variable frame rate analysis for speech recognition and voice activity detection," *IEEE J. Sel. Topics Signal Process.*, vol. 4, no. 5, pp. 798–807, Oct. 2010.
[25] P. Boersma, "Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound," *Proc. Inst. Phonetic Sci.*, vol. 17, no. 1193, pp. 97–110, 1993.
[26] M. Shah, J. Wang, D. Blaauw, D. Sylvester, H.-S. Kim, and C. Chakrabarti, "A fixed-point neural network for keyword detection on resource constrained hardware," in *Proc. IEEE Workshop Signal Process. Syst. (SiPS)*, Oct. 2015, pp. 1–6.
[27] E. Chuangsuwanich and J. Glass, "Robust voice activity detector for real world applications using harmonicity and modulation frequency," in *Proc. INTERSPEECH*, 2011, pp. 2645–2648.
[28] H.-G. Hirsch and D. Pearce, "The Aurora experimental framework for the performance evaluation of speech recognition systems under noisy conditions," in *Proc. ASR-Autom. Speech Recognit., Challenges New Millenium ISCA Tutorial Res. Workshop (ITRW)*, 2000, pp. 181–188.
[29] S. Teller *et al.*, "A voice-commandable robotic forklift working alongside humans in minimally-prepared outdoor environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2010, pp. 526–533.
[30] J. J. Godfrey, E. C. Holliman, and J. McDaniel, "SWITCHBOARD: Telephone speech corpus for research and development," in *Proc. IEEE Int. Acoust., Speech, Signal Process. (ICASSP)*, vol. 1. Mar. 1992, pp. 517–520.

Michael Price (M'15) received the M.Eng. degree from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, in 2009, with a focus on high-speed serial links, and the Ph.D. degree from MIT in 2016, focusing on the work described in this paper.

From 2009 to 2011, he was an Electrical Engineer with Aurora Flight Sciences, Cambridge, MA, where he was involved in developing electronics and control systems for unmanned aerial vehicles and satellites. He is currently a Digital Design Engineer with Analog Devices, Cambridge, MA.

Dr. Price received the David Adler Thesis Award in 2009 and an MIT Presidential Fellowship in 2011.

James Glass (F'14) received the B.Eng. degree from Carleton University, Ottawa, ON, USA, in 1982, and the S.M. and Ph.D. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, in 1985 and 1988, respectively.

He is currently a Senior Research Scientist with MIT, where he heads the Computer Science and Artificial Intelligence Laboratory, Spoken Language Systems Group. His current research interests include the area of automatic speech recognition, unsupervised speech processing, and spoken language understanding. He has supervised over 60 student theses and published approximately 200 papers in these areas.

Dr. Glass is a member of the Editorial Board of *Computer, Speech, and Language*. He is an Associate Editor of the IEEE TRANSACTIONS ON AUDIO, SPEECH AND LANGUAGE PROCESSING.

Anantha P. Chandrakasan (F'04) received the B.S., M.S., and Ph.D. degrees from the University of California at Berkeley, Berkeley, CA, USA, in 1989, 1990, and 1994, respectively, all in electrical engineering and computer sciences.

He was the Director of Microsystems Technology Laboratories, Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, from 2006 to 2011, and the Head of the Electrical Engineering and Computer Science Department, MIT, from 2011 to 2017. Since 1994, he has been with MIT, where he is currently the Dean of Engineering. His current research interests include micropower digital and mixed-signal integrated circuit design, wireless microsensor system design, portable multimedia devices, energy efficient radios, and emerging technologies.

Dr. Chandrakasan received the 2009 Semiconductor Industry Association University Researcher Award and the 2013 IEEE Donald O. Pederson Award in Solid-State Circuits. He has been the ISSCC Conference Chair since 2010.