Low-Power Automatic Speech Recognition Through a Mobile GPU and a Viterbi Accelerator

The authors' automatic speech recognition system for low-power devices combines a mobile GPU for the deep neural network with a dedicated hardware accelerator for the Viterbi search. Their proposal outperforms traditional solutions running on the CPU by orders of magnitude. Compared to a GPU-only system, the authors' hybrid scheme improves performance by 5.25 times while reducing energy by 2.05 times.

Reza Yazdani
Albert Segura
José-María Arnau
Antonio González
Universitat Politècnica de Catalunya

Automatic speech recognition (ASR) has attracted the attention of the academic community[1–3] and industry[4] in recent years (see the "Related Work in Automatic Speech Recognition" sidebar). ASR is becoming a key feature in smartphones, tablets, and other energy-constrained devices such as smart watches. ASR technology is at the heart of popular voice-based user interfaces for mobile devices, such as Google Now, Apple's Siri, and Microsoft's Cortana. These systems deliver large-vocabulary, real-time, speaker-independent, and continuous speech recognition. Unfortunately, supporting fast and accurate speech recognition comes at a high energy cost, which is not affordable in most mobile devices.

An ASR pipeline comprises two stages: a deep neural network (DNN) and a Viterbi search. The DNN computes phonemes' probabilities for each frame (typically around 10 ms) of the input audio signal, whereas the Viterbi search uses these probabilities to generate the most likely sequence of words. Our profiling of Kaldi,[5] a speech recognition system widely used in academia and industry, on an Nvidia TX1 mobile platform shows that the Viterbi search is the main bottleneck, as it represents 65 percent of the execution time when running Kaldi on its ARM CPU and 81 percent when running it on its mobile GPU.

In recent years, much work has focused on using GPUs to boost DNN performance,[6–9] which achieves huge speedups and energy savings because DNN computation is easy to parallelize. Similarly, we obtained a speedup of 11.6 times when running the DNN implementation of Kaldi on a mobile GPU. On the other hand, the Viterbi search is difficult to parallelize,[10] and hence, previous work on GPU acceleration reported a more modest speedup of 3.74 times.[11] Our numbers are also consistent with

Published by the IEEE Computer Society

0272-1732/17/$33.00 © 2017 IEEE

Related Work in Automatic Speech Recognition

Prior work on hardware accelerators for ASR assumes severe constraints to simplify the hardware, mainly by reducing the size of the vocabulary and, hence, the accuracy of the system. The accelerator presented by Michael Price and colleagues supports a 5,000-word vocabulary dissipating 6 mW, including acoustic scoring and Viterbi search.[1] Our work is different as we focus on large-vocabulary systems. On the other hand, the Viterbi ASIC described by Jeffrey Johnston and Rob Rutenbar supports a vocabulary of 60,000 words dissipating 454 mW.[2] In comparison, our Viterbi accelerator supports a larger vocabulary of 125,000 words in real time dissipating about the same power (453 mW).

References

1. 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, 2015, pp. 102–112.
2. J.R. Johnston and R.A. Rutenbar, "A High-Rate, Low-Power, ASIC Speech Decoder using Finite State Transducers," *Proc. IEEE 23rd Int'l Conf. Application-Specific Systems, Architectures, and Processors* (ASAP), 2012, pp. 77–85.

previous findings, as we obtained a speedup of 5 times for the Viterbi search on a GPU after thoroughly optimizing and tuning the code.

In this article, we propose an ASR architecture for mobile devices that combines a mobile GPU and a novel hardware accelerator. In our system, the GPU performs the DNN computations, as mobile GPUs excel in this task,[12] whereas the Viterbi search runs on a dedicated accelerator specifically tailored to the needs of ASR systems. The GPU and the accelerator share the same address space in the main memory, so the results computed by the GPU—that is, the DNN output—can be efficiently accessed by the accelerator, avoiding additional memory copies from GPU memory to system memory. To further improve performance, the input audio frames are grouped in batches, and both the GPU and accelerator work in parallel in a pipelined manner: the GPU computes the DNN for the next batch while the accelerator performs the Viterbi search for the current batch.

CPU- and GPU-Based ASR System

ASR comprises three main components: feature extraction, acoustic scoring, and Viterbi search. The recognition process works as follows. First, the feature-extraction component splits the audio signal into frames of 10 ms of speech and computes several features for each frame using signal-processing techniques. ASR systems create acoustic models for subword units or phonemes. The production of sound corresponding to a phoneme is influenced by neighboring phonemes, so ASR systems typically employ context-dependent phonemes such as triphones to improve accuracy. After feature extraction, for each frame, the acoustic-scoring component computes the probabilities (also called *scores*) that the frame is part of a particular triphone, checking for all potential triphones. Finally, Viterbi search employs these sequences of probability sets (one set per frame, with as many elements as potential triphones) to find the most likely sequence of words. Acoustic scoring and Viterbi search take up the bulk of the execution time.

DNNs are the state-of-the-art approach for computing acoustic scores. DNNs for ASR comprise a sequence of fully connected layers, followed by a softmax output layer. The scores produced by the DNN are the inputs to the Viterbi search, which produces the sequence of words. The Viterbi search employs a graph-based recognition network to find the sequence of words with the maximum likelihood for the sequence of triphone probabilities. The most successful approach for representing the recognition network is the weighted finite state transducer (WFST),[13] which is a Mealy finite state machine that comprises a set of states and a set of arcs. The WFST is constructed offline during the training process using different knowledge sources, such as context-dependent phonemes, pronunciation, and grammar. For large-vocabulary ASR

Table 1. Execution time of Kaldi for decoding 127.4 seconds of speech on an Nvidia Tegra X1.

Automatic speech recognition	ARM Cortex A57 CPU (seconds)	GM204 GPU (seconds)
Deep neural network (DNN)	87.4	7.5
Viterbi search	161.5	32.2
Total	248.9	39.7

systems, the WFST contains millions of states and arcs.

In this work, we use Kaldi as our baseline ASR software solution.[5] Kaldi provides full support for acoustic scoring based on DNNs. Moreover, it implements the Viterbi search based on a WFST. We use the standard DNN and WFST for the English language as provided on Kaldi's website. The DNN comprises six connected layers followed by one softmax layer. It contains 41,095 neurons and 9 million parameters. The WFST is constructed from a large vocabulary of 125,000 words and contains more than 13 million states and more than 34 million arcs.

Table 1 shows Kaldi's execution time on an Nvidia Tegra X1 mobile platform.[14] We first evaluated Kaldi's performance on the mobile CPU of Tegra X1, a quad-core ARM Cortex A57. Kaldi does not achieve real-time performance when running on the mobile CPU, because it takes 1.95 seconds to decode each second of speech. The Viterbi search is the main bottleneck, because it takes 65 percent of the execution time.

Mobile platforms typically include a GPU that can be used to speed up the recognition process. Tegra X1 features a GM204 GPU, which is based on the Maxwell microarchitecture. Kaldi provides a CUDA implementation of the DNN, but it does not provide any GPU version for the Viterbi search because this algorithm is difficult to parallelize. We extended Kaldi with the GPU version of the Viterbi search presented by Kisun You and colleagues.[11]

Table 1 also shows the execution time of our GPU-accelerated version of Kaldi when running on Tegra X1. The GPU-based ASR system achieves real-time performance because it takes only 0.31 seconds to decode each second of speech. The GPU provides a speedup of 11.6 times for the DNN. The Viterbi search achieves a more modest speedup of 5 times, a bit higher than the 3.7 times reported in previous work.[11] Overall, the Viterbi search is the main bottleneck. It performs a search on a huge, irregular, directed graph to find the most likely word sequence for the input speech signal. Parallel graph traversal on large unstructured graphs is a well-known challenge for parallel architectures, especially in the context of ASR. The Viterbi search exhibits unpredictable memory accesses and poor data locality. These characteristics put significant pressure on the GPU memory subsystem and lead to substantial underuse of the GPU's functional units.

Hardware-Accelerated ASR System

Our GPU-accelerated ASR system achieves real-time performance on a mobile platform. However, further improving this ASR system's performance and energy efficiency can provide significant benefits. Higher performance will allow the use of larger and more sophisticated language models in real time, improving the system's accuracy. On the other hand, mobile devices are battery operated, and hence ASR systems running on those devices must consume as little energy as possible.

Because Viterbi search is the main bottleneck for ASR systems, our efforts to increase performance and reduce energy consumption focus on this component. In this article, we present a low-power accelerator for the Viterbi search. Figure 1 illustrates the architecture of the accelerator, which consists of a pipeline with five stages and several on-chip memories to speed up the access to the data required by the search process.

The Viterbi search algorithm expands the active states (of the WFST) at the current frame to create the new set of active states for the next frame. All the arcs departing from the active source states are considered during the search process. The cost of reaching the destination states at a given frame is computed using the acoustic likelihoods from the

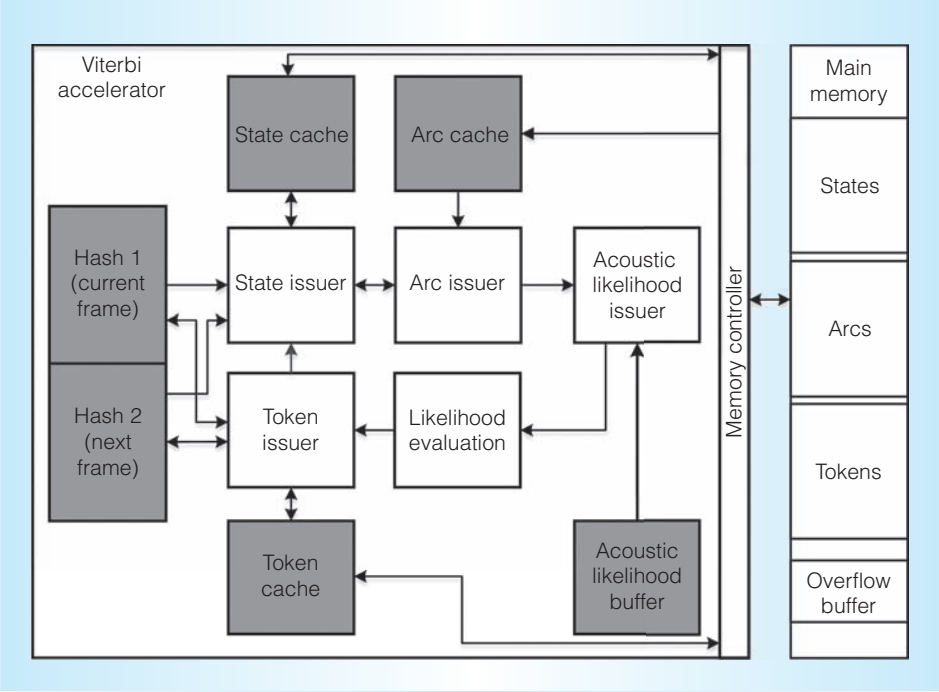

Figure 1. Architecture of the accelerator for Viterbi search. The accelerator comprises a five-stage pipeline (white rectangles) and several on-chip memories (gray rectangles).

DNN (computed in the GPU) and the weights of the arcs from the WFST. The active states at a given frame are also called *tokens.*

The accelerator's pipeline includes units for fetching states (state issuer), arcs (arc issuer), and acoustic scores (acoustic likelihood issuer) from main memory. In addition, the likelihood-evaluation unit computes the cost of reaching the destination states, using the information fetched from memory by the previous stages. Finally, the token issuer is used to store the information of the new active states in memory.

The WFST is typically quite large, so it cannot be stored in the on-chip memories. WFST states and arcs are stored in system memory. The accelerator includes a state cache and an arc cache to speed up the accesses to the states and arcs, respectively. On the other hand, the acoustic scores generated by the GPU are stored in main memory and consumed by the accelerator during the search process. Compared to the GPU-only system, moving the acoustic scores from the GPU to the accelerator is the only data movement overhead. However, the latency of this data movement can be completely hidden by overlapping the memory transfers with computations. To this end, the accelerator includes a double-buffered acoustic likelihood buffer that stores the acoustic scores for two frames of speech. The accelerator fetches from memory the scores for the next frame while it processes the current frame, hiding the memory latency incurred by bringing the acoustic scores from main memory.

The accelerator tracks the active states, or tokens, generated dynamically throughout the search. The information associated with the tokens is split into two parts, depending on whether the data must be kept until the end of the search or is required only for a given frame of speech. Persistent data is stored in main memory, leveraging the token cache, and temporary data is stored in the hash table. The accelerator includes two hash tables to store the tokens for the current and next frames of speech.

The result generated by the accelerator is the list of tokens expanded at every frame of speech. The tokens generated for all frames form a word lattice, which represents different alternative interpretations of the input

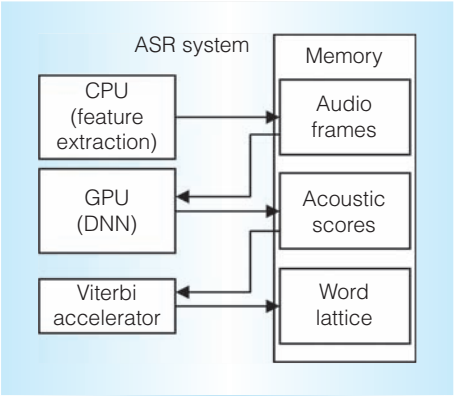

Figure 2. Architecture of the proposed automatic speech recognition (ASR) system. It comprises a CPU, GPU, and Viterbi accelerator. All these components have access to a shared address space in main memory.

speech. A backtracking algorithm is employed to select the most likely interpretation: the token with the maximum likelihood in the last frame is selected, and the backpointers saved by the accelerator are followed to reconstruct the most probable path. The backtracking requires a negligible amount of time and is executed on the CPU.

Figure 2 shows our overall ASR system for mobile devices. The CPU performs feature extraction to generate the audio frames in main memory. The GPU computes the DNN for those audio frames and generates the acoustic likelihoods. The accelerator performs the Viterbi search by using the acoustic scores to generate the word lattice in system memory. Finally, the CPU performs backtracking to obtain the most likely sequence of words.

In our system, the GPU and the accelerator work in parallel in a pipelined manner (see Figure 3). Speech frames are grouped in batches. The GPU computes the DNN for the next batch of frames while the accelerator performs the Viterbi search for the current batch. Therefore, we overlap the latency of the DNN and the Viterbi search.

Experimental Results

In this section, we present the evaluation methodology and experimental results for our proposed ASR system. Regarding the methodology, we run our GPU-accelerated version of Kaldi on an Nvidia Jetson TX1 platform that features a GM204 GPU with parameters shown in Table 2.[12] To measure GPU energy consumption, we read the registers of the Texas Instruments INA3221 power monitor included in the Jetson TX1 platform, which provides the actual energy by monitoring the GPU power rail as described by Merlin Friesen.[15]

On the other hand, we developed a cycle-accurate simulator that models the architecture of the Viterbi accelerator we discussed in the previous section. Table 3 shows the parameters we employed for the experiments in the accelerator. We performed a design-space exploration to find appropriate parameters for the different hardware structures and selected the configuration that provides the best tradeoff considering performance, power, and area.[16] We found that cache size and hash table size are the most critical parameters, because misses in these hardware structures are the main sources of pipeline stalls. According to our experiments, sizes bigger than those in Table 3 provide little reduction in the miss ratio, even for large WFSTs. Note that the arc cache is twice the size of the other caches to deal with the larger memory footprint required by arc fetching. On average, two arcs are fetched from memory for each token during Viterbi search.

To estimate the accelerator's energy consumption, we implemented the different pipeline components of the accelerator in Verilog and synthesized them using the Synopsys Design Compiler with a 28-nm cell library. In addition, we used CACTI to estimate the power of the caches included in the accelerator. All the energy numbers reported in this section include both static and dynamic energy. We used the delays provided by CACTI and Synopsys Design Compiler to set realistic latencies for the different hardware structures in the cycle-accurate simulator. Finally, by using both the execution time and the activity factor of each component generated by the simulator, we conducted an estimation of power dissipation according to the numbers collected from the synthesis tools.

For our datasets, we again used the DNN and the WFST for English language provided in the Kaldi toolset.[5] These models are trained

with 2,000 hours of speech and a large vocabulary of 125,000 words. We used audio files from LibriSpeech corpus[17] and ran the decoding for 127.4 seconds of speech.

Regarding the experimental results, we first focused on the Viterbi search and evaluated our Viterbi accelerator's performance and energy efficiency. Figure 4 shows our results, together with the energy and performance of a low-power GPU-only implementation. The Viterbi accelerator achieves real-time performance by a wide margin, as it takes 11 ms to decode each second of speech. The speedup achieved by the accelerator over the mobile GPU is 22 times. In addition to the performance improvement, the accelerator also provides a huge reduction (48 times) in energy consumption. Energy efficiency is drastically improved by using the hardware specifically tailored to the Viterbi search characteristics.

Table 4 shows the speedup achieved by our Viterbi accelerator over the GPU version for WFSTs of different sizes. The table also reports the word error rate as a measure of the accuracy achieved by the different WFSTs. Our Viterbi accelerator is designed for large-vocabulary ASR systems that deliver high recognition accuracy; hence, the larger the WFST, the bigger the speedup achieved.

Figure 5 shows execution time and energy consumption for the overall ASR system, including the DNN and Viterbi search. The GPU-only configuration corresponds to a system that runs the entire ASR pipeline on the mobile GPU (Figure 3a). The GPU+Accelerator configuration is the system that employs the GPU for the DNN computation and our accelerator for the Viterbi search, as shown in Figure 2. The GPU+Accelerator configuration provides 5.26 times speedup over the GPU-only system, which comes from two sources. First, the accelerator provides a large speedup for the main bottleneck of ASR, the Viterbi search, as shown in Figure 4. Second, the GPU+Accelerator configuration overlaps the execution time of the DNN and the Viterbi search, as shown in Figure 3, providing further performance improvements over the GPU-only configuration that has to serialize the two stages. In addition, GPU+Accelerator provides an energy reduction of 2.05 times.

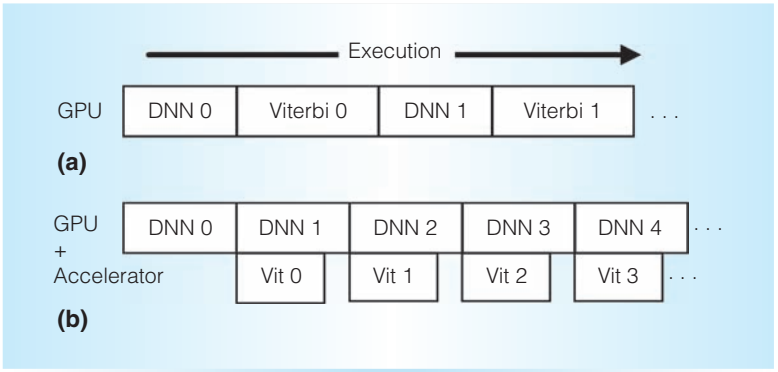

Figure 3. Execution of (a) the GPU-only ASR system and (b) our proposed architecture combining a GPU and the Viterbi accelerator. Overlapping DNN evaluation and Viterbi search further improves the performance of our ASR system.

Table 2. Mobile GPU parameters.

Parameter	Value
GPU	GM 204 (Nvidia Tegra X1)
Streaming multiprocessors	2 (2,048 threads/multiprocessor)
Technology	20 nm
Frequency	1.0 GHz
Level-2 cache	256 Kbytes

Table 3. Hardware parameters for the accelerator.

Parameter	Value
Technology	28 nm
Frequency	600 MHz
State cache	512 Kbytes, four-way, 64 bytes/line
Arc cache	1 Mbyte, four-way, 64 bytes/line
Token cache	512 Kbytes, two-way, 64 bytes/line
Acoustic likelihood buffer	64 Kbytes
Hash table	768 Kbytes
Memory controller	32 in-flight requests
Likelihood evaluation unit	4 floating-point adders, 2 floating-point comparators

In the proposed system (GPU+Accelerator), the main bottleneck for both performance and energy consumption is now the DNN running on the GPU, because the proposed Viterbi accelerator is extremely effective. If required, we can further improve the system by replacing the GPU with an accelerator for neural networks, such as the

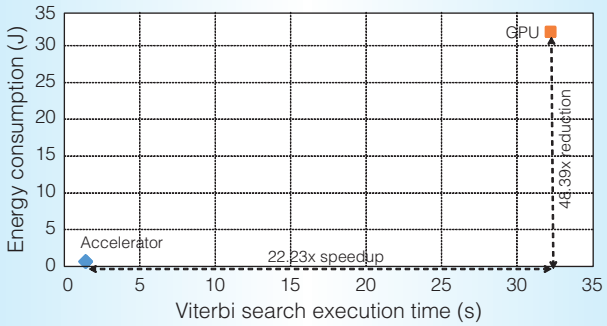

Figure 4. Energy consumption versus execution time for the Viterbi search, for decoding 127.4 seconds of speech. The Viterbi accelerator outperforms GPU implementation while dramatically reducing energy consumption.

Table 4. Speedup and word error rate for different vocabulary sizes.

Weighted finite state transducer	Size (no. of words)	Speedup (times)	Word error rate (%)
Voxforge	14,000	2.08	37.05
Fisher English	125,000	22.23	25.87
LibriSpeech	200,000	38.06	10.46

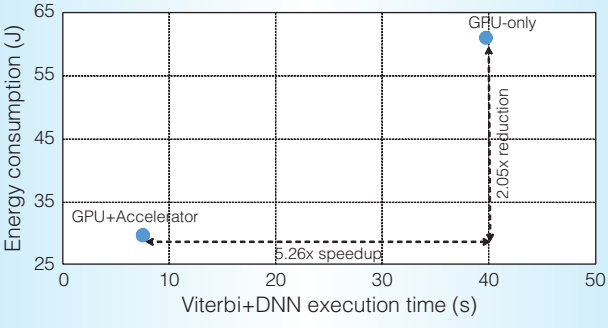

Figure 5. Energy consumption versus decoding time for the entire ASR system, for decoding 127.4 seconds of speech. Our proposal achieves high speedup and energy reduction compared to a GPU-only configuration.

one proposed by Zidong Du and colleagues.[18]

Our Viterbi accelerator supports any WFST, so it works for any language, acoustic model (basephones, triphones), and language model (bigrams, trigrams). The accelerator can also support more sophisticated future speech models just by extending the WFST. We believe speech recognition will be supported by most computing devices in the near future, and language models will evolve toward more complex ones for the sake of better accuracy, so the proposed accelerator's benefits will be even higher.

On the other hand, the Viterbi algorithm is also employed by other applications, such as statistical machine translation,[19] text-to-speech synthesis,[20] and computational biology,[21] which can also benefit from our accelerator. Thus, we are eager to extend our accelerator in order to provide a platform that can support different applications that make extensive use of the Viterbi search algorithm. MICRO

References

1. J. Choi, K. You, and W. Sung, "An FPGA Implementation of Speech Recognition with Weighted Finite State Transducers," *Proc. IEEE Int'l Conf. Acoustics Speech and Signal Processing* (ICASSP), 2010, pp. 1602–1605.
2. J.R. Johnston and R.A. Rutenbar, "A High-Rate, Low-Power, ASIC Speech Decoder Using Finite State Transducers," *Proc. IEEE 23rd Int'l Conf. Application-Specific Systems, Architectures, and Processors* (ASAP), 2012, pp. 77–85.
3. 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, 2015, pp. 102–112.
4. "IBM Watson Speech to Text," www.ibm.com/smarterplanet/us/en/ibmwatson/developercloud/speech-to-text.html.
5. D. Povey et al., "The Kaldi Speech Recognition Toolkit," *Proc. IEEE Workshop Automatic Speech Recognition and Understanding*, 2011, EPFL-CONF 192584.
6. A. Krizhevsky, I. Sutskever, and G.E. Hinton, "Imagenet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

7. G. Hinton et al., "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups," *IEEE Signal Processing Mag.*, vol. 29, no. 6, 2012, pp. 82–97.
8. Y. Jia et al., "Caffe: Convolutional Architecture for Fast Feature Embedding," preprint, 2014; https://arxiv.org/abs/1408.5093.
9. D. Povey, X. Zhang, and S. Khudanpur, "Parallel Training of DNNs with Natural Gradient and Parameter Averaging," preprint, 2014; https://arxiv.org/abs/1410.7455.
10. J. Chong, E. Gonina, and K. Keutzer, "Efficient Automatic Speech Recognition on the GPU," *GPU Computing Gems Emerald Edition*, Morgan Kaufmann, vol. 1, 2011, pp. 601–618.
11. K. You et al., "Parallel Scalability in Speech Recognition," *IEEE Signal Processing Mag.*, vol. 26, no. 6, 2009, pp. 124–135.
12. *GPU-Based Deep Learning Inference: A Performance and Power Analysis*, white paper, Nvidia; www.nvidia.com/content/tegra/embedded-systems/pdf/jetson_tx1_whitepaper.pdf.
13. M. Mohri, F. Pereira, and M. Riley, "Weighted Finite-State Transducers in Speech Recognition," *Computer Speech and Language*, vol. 16, no. 1, 2002, pp. 69–88.
14. *Nvidia Tegra X1*, white paper, Nvidia; http://international.download.nvidia.com/pdf/tegra/Tegra-X1-whitepaper-v1.0.pdf.
15. M. Friesen, "Linux Power Management Optimization on the Nvidia Jetson Platform," www.socallinuxexpo.org/sites/default/files/presentations/scale14x_r27_final.pdf.
16. R. Yazdani et al., "An Ultra-Low Power Hardware Accelerator for Automatic Speech Recognition," *Proc. 49th Ann. IEEE/ACM Int'l Symp. Microarchitecture*, 2016, pp. 552–563.
17. V. Panayotov et al., "LibriSpeech: An ASR Corpus Based on Public Domain Audio Books," *Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing* (ICASSP), 2015; doi:10.1109/ICASSP.2015.7178964.
18. Z. Du et al., "ShiDianNao: Shifting Vision Processing Closer to the Sensor," *Proc. Int'l Symp. Computer Architecture* (ISCA), 2015; doi:10.1145/2749469.2750389.
19. Y. Deng and W. Byrne, "HMM Word and Phrase Alignment for Statistical Machine Translation," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 16, no. 3, 2008, pp. 494–507.
20. M.L. Padmesh and P.S. Kumar, "Implementation of Viterbi Coder for Text to Speech Synthesis," *Proc. IEEE Int'l Conf. Computational Intelligence and Computing Research* (ICCIC), 2015; doi:10.1109/ICCIC.2015.7435659.
21. R. Šrámek, B. Brejová, and T. Vinař, "On-Line Viterbi Algorithm for Analysis of Long Biological Sequences," *Algorithms in Bioinformatics*, LNCS 4645, Springer, 2007, pp. 240–251.

Reza Yazdani is a PhD student at the Universitat Politècnica de Catalunya. His research interests include the design of high-performance and low-power accelerators for cognitive computing architectures, fault-tolerant design, embedded systems, and VLSI design. Yazdani received an MS in computer architecture from the University of Tehran. Contact him at ryazdani@ac.upc.edu.

Albert Segura is a PhD student at the Universitat Politècnica de Catalunya. His research interests include cognitive computing on GPU architectures. Segura received an MS in computer architecture from UPC. Contact him at asegura@ac.upc.edu.

José-María Arnau is a postdoctoral researcher at the Universitat Politècnica de Catalunya. His research interests include low-power architectures for cognitive computing. Arnau received a PhD in computer architecture from UPC. Contact him at jarnau@ac.upc.edu.

Antonio González is a full professor in the Computer Architecture Department at the Universitat Politècnica de Catalunya. His research interests include computer architecture, compilers, and parallel processing. González received a PhD in computer engineering from UPC. He is an IEEE Fellow. Contact him at antonio@ac.upc.edu.