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

Reza Yazdani, Albert Segura, Jose-Maria Arnau, Antonio Gonzalez
Computer Architecture Department, Universitat Politecnica de Catalunya
{ryazdani, asegura, jarnau, antonio}@ac.upc.edu

***Abstract*—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.**

***Index Terms*—Automatic Speech Recognition, Accelerator, Viterbi Search**

I. Introduction

Automatic Speech Recognition (ASR) has attracted the attention of the academic community [1], [2], [3] and industry [4], [5], [6], [7] in recent years. ASR is becoming a key feature in smartphones, tablets and other energy-constrained devices such as smartwatches. ASR technology is at the heart of popular voice-based user interfaces for mobile devices such as Google Now, Apple Siri or Microsoft 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 [8], a speech recognition system widely used in academia and industry, on a 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 GPUs to boost DNN performance [9], [10], [11], [12], 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 hard to parallelize [13] and, hence, previous work on GPU acceleration reported a more modest speedup of 3.74 times [14]. Our numbers are also consistent with previous findings, as we obtained a speedup of 5 times for the Viterbi search on a GPU after thoroughly optimizing and tuning of the code.

In this artcle, 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 [15], 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 main memory, so the results computed by the GPU, i. e. 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 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.

II. CPU and GPU-Based ASR System

ASR comprises three main components: Feature Extraction, Acoustic Scoring and Viterbi Search. The recognition process works as follows. Firstly, 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 sub-word units or phonemes. The production of sound corresponding to a phoneme is influenced by neighboring phonemes, so context-dependent phonemes or triphones are typically employed by ASR systems to improve accuracy. After Feature Extraction, for each frame, the Acoustic Scoring computes the probabilities (also called *scores*) that the frame is part of a particular triphone, checking for all potential triphones. Finally, the Viterbi Search employs these sequence of probability sets (one set per frame with as many elements as potential triphones) to find the most likely sequence of words. The Acoustic Scoring and the Viterbi Search take up the bulk of execution time.

DNNs are the state-of-the-art approach for computing acoustic scores. DNNs for ASR consist of 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 maximum likelihood for the sequence of triphone's probabilities. The most successful approach for representing the recognition network is the Weighted Finite State Transducer (WFST) [16], 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

TABLE I
EXECUTION TIME OF KALDI FOR DECODING 127.4 SECONDS OF SPEECH ON AN NVIDIA TEGRA X1.

	CPU - ARM Cortex A57	GPU - GM204
DNN	87.4 s	7.5 s
Viterbi Search	161.5 s	32.2 s
TOTAL	248.9 s	39.7 s

different knowledge sources such as context dependency of phonemes, pronunciation and grammar. For large vocabulary ASR systems, the WFST contains millions of states and arcs.

In this paper, we use Kaldi [8] as our baseline ASR software solution. Kaldi is a widely used toolkit for ASR that delivers state-of-the-art performance and accuracy. 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 English language as provided in Kaldi's website. The DNN comprises six fully-connected layers followed by one softmax layer, it contains 41,095 neurons and 9 million parameters. Regarding the WFST, it is constructed from a large vocabulary of 125,000 words and it contains more than 13 million states and more than 34 million arcs.

Table I shows the execution time of Kaldi on an NVIDIA Tegra X1 [17] mobile platform. We have first evaluated the performance of Kaldi when running 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, since it takes 1.95 seconds to decode each second of speech. The Viterbi search is the main bottleneck as 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 since this algorithm is challenging to parallelize. We have extended Kaldi with the state-of-the-art GPU version of the Viterbi search presented by Kisun You and colleagues [14].

Table I shows the execution time of our GPU-accelerated version of Kaldi when running on a Tegra X1. The GPU-based ASR system achieves real-time performance as it only takes 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 [14]. 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 underutilization of GPU's functional units.

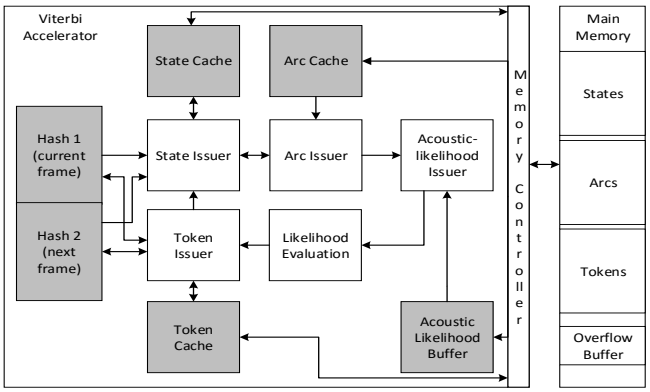

Fig. 1. Architecture of the accelerator for Viterbi search.

III. 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 the Viterbi search is the main bottleneck for ASR systems, our efforts to increase performance and reduce energy consumption focus on the Viterbi search 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 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,

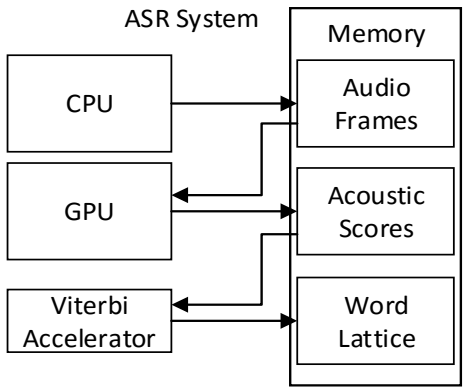

Fig. 2. Architecture of the proposed ASR system.

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 of bringing the acoustic scores from main memory.

The accelerator keeps track of 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 has to be kept until the end of the search or it is only required for a given frame of speech. Persistent data is stored in main memory leveraging the Token Cache. Temporary data is stored in the Hash Table. The accelerator includes two hash tables to store the tokens for the current and the next frame 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 what is called a word lattice, which represents different alternative interpretations of the input speech. A backtracking algorithm is employed to select the most likely interpretation. The token with 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 it 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 in order to generate the word lattice in system memory. Finally, the CPU performs the backtracking to obtain the most likely sequence of words.

In our system, the GPU and the accelerator work in parallel in a pipelined manner as illustrated in 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.

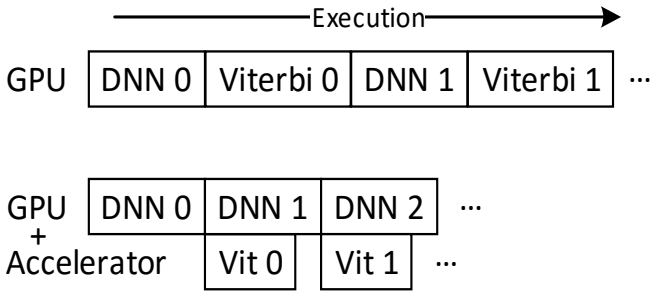

Fig. 3. Execution of the GPU-only ASR system (top) and our proposed architecture combining a GPU and the Viterbi accelerator (bottom).

TABLE II
MOBILE GPU PARAMETERS.

GPU	NVIDIA Tegra X1 - GM204
Streaming multiprocessors	2 (2048 threads/multiprocessor)
Technology	20 nm
Frequency	1.0 GHz
L2 cache	256 KB

IV. EXPERIMENTAL RESULTS

This section presents the evaluation methodology and the experimental results for the proposed ASR system. Regarding the methodology, we run our GPU-accelerated version of Kaldi on an NVIDIA Jetson TX1 platform [15] that features a GM204 GPU with parameters shown in Table II. To measure GPU energy consumption, we read the registers of the TI 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 [18].

On the other hand, we have developed a cycle-accurate simulator that models the architecture of the Viterbi accelerator presented in Section III. Table III shows the parameters employed for the experiments in the accelerator. We have performed a design space exploration to find appropriate parameters for the different hardware structures, selecting the configuration that provides the best trade-off considering performance, power and area [19]. We found that the sizes of the caches and the hash tables are the most critical parameters, as misses in these hardware structures are the main sources of pipeline stalls. According to our experiments, sizes bigger than the ones shown in Table III provide little reduction in miss ratio, even for large WFSTs. Note that the size of 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 state/token during Viterbi search.

To estimate energy consumption of the accelerator, we have 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 use 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 use 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

TABLE III
HARDWARE PARAMETERS FOR THE ACCELERATOR.

Technology	28 nm
Frequency	600 MHz
State Cache	512 KB, 4-way, 64 bytes/line
Arc Cache	1 MB, 4-way, 64 bytes/line
Token Cache	512 kB, 2-way, 64 bytes/line
Acoustic Likelihood Buffer	64 KB
Hash Table	768 KB
Memory Controller	32 in-flight requests
Likelihood Evaluation Unit	4 fp adders, 2 fp comparators

TABLE IV
SPEEDUP AND WORD ERROR RATE FOR DIFFERENT VOCABULARY SIZES.

WFST	Size (words)	Speedup	WER(%)
Voxforge	14,000	2.08	37.05
Fisher English	125,000	22.23	25.87
Librispeech	200,000	38.06	10.49

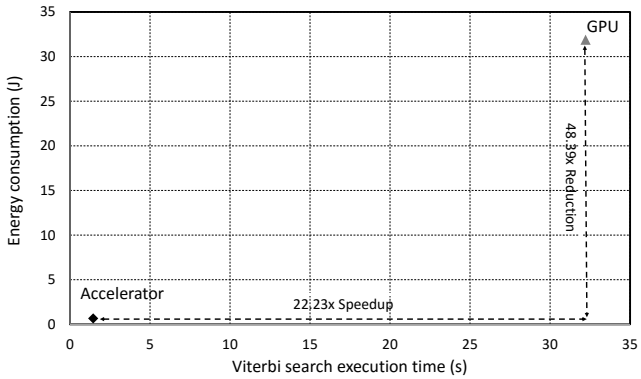

Fig. 4. Energy consumption versus execution time for the Viterbi search.

time and the activity factor of each component generated by the simulator, an accurate power estimation is conducted 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 [8]. These models are trained with 2000 hours of speech and a large vocabulary of 125,000 words. We used audio files from Librispeech corpus [20] and ran the decoding for 127.42 seconds of speech.

Regarding the experimental results, we first focus on the Viterbi search and evaluate the performance and energy-efficiency of our Viterbi accelerator. Results are shown in Figure 4, 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 characteristics of the Viterbi search.

Table IV 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 the Viterbi search. The GPU-only configuration corresponds to a system that runs the entire ASR pipeline on the mobile GPU (top of Figure 3). The GPU+ACC configuration is the system that employs the GPU for the DNN computation and our accelerator for the Viterbi search, as illustrated in Figure 2. The GPU+ACC 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+ACC configuration overlaps the execution time of the DNN and the Viterbi search, as illustrated in Figure 3, providing further performance improvements over the GPU-only configuration that has to serialize the two stages. Besides, GPU+ACC provides an energy reduction of 2.05 times.

In the proposed system (GPU+ACC), 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 one proposed by Zidong Du and colleagues [21].

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 speech models that may appear in the future, by just extending the WFST. We believe speech recognition will be a feature supported by the most computing devices in the near future, and language models will evolve towards more complex ones for the sake of better accuracy, so the benefits of the proposed accelerator will be even higher.

On the other hand, the Viterbi algorithm is also employed by other applications such as statistical machine translation [22], text to speech synthesis [23], and computational biology [24], which can 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.

V. RELATED WORK IN AUTOMATIC SPEECH RECOGNITION

Prior works on hardware accelerators for ASR assume 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 [3]. 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 in real-time dissipating about the same power (453 mW).

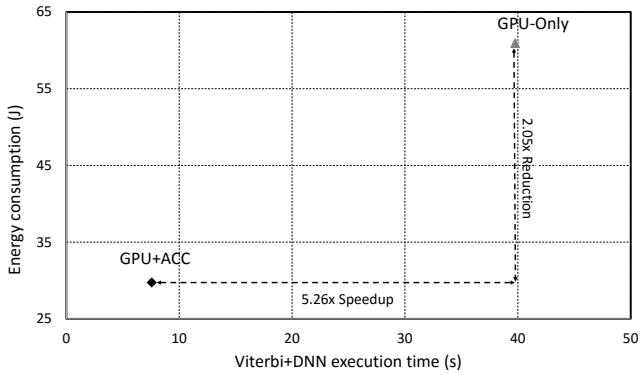

Fig. 5. Energy consumption versus decoding time for the entire ASR system.

REFERENCES

[1] J. Choi, K. You, and W. Sung, "An fpga implementation of speech recognition with weighted finite state transducers," in *ICASSP 2010*, 2010, pp. 1602–1605.

[2] J. R. Johnston and R. A. Rutenbar, "A high-rate, low-power, asic speech decoder using finite state transducers," in *ASAP 2012*, 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 Journal of Solid-State Circuits*, vol. 50, no. 1, pp. 102–112, 2015.

[4] Google Now, https://en.wikipedia.org/wiki/Google_Now.

[5] Apple Siri, https://en.wikipedia.org/wiki/Siri.

[6] Microsoft Cortana, https://en.wikipedia.org/wiki/Cortana_%28software%29.

[7] "IBM Watson Speech to Text," http://www.ibm.com/smarterplanet/us/en/ibmwatson/developercloud/speech-to-text.html.

[8] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The kaldi speech recognition toolkit," in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, 2011.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[10] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[11] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[12] D. Povey, X. Zhang, and S. Khudanpur, "Parallel training of dnns with natural gradient and parameter averaging," *arXiv preprint arXiv:1410.7455*, 2014.

[13] J. Chong, E. Gonina, and K. Keutzer, "Efficient automatic speech recognition on the gpu," *Chapter in GPU Computing Gems Emerald Edition, Morgan Kaufmann*, vol. 1, 2011.

[14] K. You, J. Chong, Y. Yi, E. Gonina, C. J. Hughes, Y. K. Chen, W. Sung, and K. Keutzer, "Parallel scalability in speech recognition," *IEEE Signal Processing Magazine*, vol. 26, no. 6, pp. 124–135, 2009.

[15] NVIDIA Corporation, "Gpu-based deep learning inference: A performance and power analysis," http://www.nvidia.es/content/tegra/embedded-systems/pdf/jetson_tx1_whitepaper.pdf.

[16] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech and Language*, vol. 16, no. 1, pp. 69 – 88, 2002.

[17] NVIDIA Corporation, "Nvidia tegra x1," http://international.download.nvidia.com/pdf/tegra/Tegra-X1-whitepaper-v1.0.pdf.

[18] Merlin Friesen, "Linux Power Management Optimization on the Nvidia Jetson Platform," http://events.linuxfoundation.org/sites/events/files/slides/Linux_Low_Power_ELC_SanDiego.pdf.

[19] R. Yazdani, A. Segura, J.-M. Arnau, and A. Gonzalez, "An ultra low-power hardware accelerator for automatic speech recognition," in *Proceedings of Micro'49*, 2016.

[20] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An asr corpus based on public domain audio books," in *ICASSP*, 2015.

[21] Z. Du, R. Fasthuber, T. Chen, P. Ienne, L. Li, T. Luo, X. Feng, Y. Chen, and O. Temam, "Shidiannao: Shifting vision processing closer to the sensor," in *Proceedings of ISCA*, 2015.

[22] Y. Deng and W. Byrne, "Hmm word and phrase alignment for statistical machine translation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 3, pp. 494–507, March 2008.

[23] M. L. Padmesh and P. S. Kumar, "Implementation of viterbi coder for text to speech synthesis," in *ICCIC*, 2015.

[24] R. Šrámek, B. Brejová, and T. Vinař, "On-line viterbi algorithm for analysis of long biological sequences," in *International Workshop on Algorithms in Bioinformatics*. Springer, 2007, pp. 240–251.

Reza Yazdani is a PhD student at UPC BarcelonaTech. He is pursuing his research under the supervision of Jose-Maria Arnau and Antonio Gonzalez, working on designing high-performance and low-power accelerators for cognitive computing architectures. His research interests also include fault-tolerant design, embedded systems and VLSI design. Contact him at ryazdani@ac.upc.edu.

Albert Segura is a PhD student at UPC BarcelonaTech. His doctorate research focuses on the area of Cognitive Computing on GPU Architectures. Albert is carrying out his doctorate under the supervision of Prof. Antonio Gonzalez and Dr. Jose-Maria Arnau. Contact him at asegura@ac.upc.edu.

Jose-Maria Arnau is a postdoctoral researcher at UPC BarcelonaTech. 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 Gonzalez (PhD 1989) is a Full Professor at the Computer Architecture Department of the Universitat Politcnica de Catalunya, Barcelona (Spain). His research has focused on computer architecture, compilers and parallel processing, with a special emphasis on microarchitecture and code generation. He is an IEEE Fellow. Contact him at antonio@ac.upc.edu.