AIX: A high performance and energy efficient inference accelerator on FPGA for a DNN-based commercial speech recognition

1st Minwook Ahn, Seok Joong Hwang, Wonsub Kim, Seungrok Jung, Yeonbok Lee, Mookyoung Chung
Woohyung Lim, Youngjoon Kim
SK Telecom
6, Hwangsaeul-ro, 258beon-gil, Bundang-gu, Seongnam-si, Gyeonggi-do, Korea, 13595
Email: {minwook.ahn, nzthing, wonsub79.kim, seungrok.jung, yeonbok.lee, mk.chung, w.lim, youngjoon.kim}@sk.com

***Abstract*—Automatic speech recognition (ASR) is crucial in virtual personal assistant (VPA) services such as Apple Siri, Amazon Alexa, Google Now and SKT NUGU. Recently, ASR has been showing a remarkable advance in accuracy by applying deep learning. However, with the explosive increase of the user utterances and growing complexities in ASR, the demands for the custom accelerators in datacenters are highly increasing in order to process them in real time with low power consumption. This paper evaluates a custom inference accelerator for ASR enhanced by a deep neural network, called AIX (Artificial Intelligence aXellerator). AIX is developed on a Xilinx FPGA and deployed to SKT NUGU since 2018. Owing to the full exploitation of DSP slices and memory bandwidth provided by FPGA, AIX outperforms the cutting-edge CPUs by 10.2 times and even a state-of-the-art GPU by 20.1 times with real time workloads of ASR in performance and power consumption wise. This improvement achieves faster response time in ASR, and in turn reduces the number of required machines in datacenters to a third.**

***Index Terms*—neural network, inference, accelerator, speech recognition, FPGA**

I. Introduction

Recent improvement in ASR accuracy has brought a significant growth in the VPA market with applying deep learning. On top of such higher accuracy, real time processing is critical to VPAs as customers normally do not have patience to VPAs' response. Moreover, exponential increase in users continuously requires more service channels. Though general-purpose computing machines such as high-end CPUs and GPUs have been widely used, their performance along with power consumption is occasionally unsatisfactory even in datacenter, as the power consumption of the machines in datacenters is enormous [18]. Therefore, demand for computing machines with improved computation capability and lower power consumption is increasing. To this end, domain specific accelerators such as Googles TPU [8] are drawing attention. However, such custom hardware chips cannot be easily applied to quite different types of applications and suffer from the huge non-recurring engineering (NRE) cost for fabrication.

This paper introduces a custom inference accelerator, called AIX, developed on a Xilinx FPGA in order to speed up the time-consuming neural network computations. Using FPGA, AIX is not only flexible for design updates for new applications but also requires lower power consumption comparing to the general purpose computing machines. The initial version of AIX aims to accelerate the ASR of SKT NUGU, which is the 1^{st} Korean VPA served by SK Telecom since 2016. NUGU employs a state-of-the-art deep learning based ASR for better accuracy in recognition. A neural network in the ASR more accurately computes the probabilities of phones from human voice signals depending on the structure of multilayer perceptron (MLP) with eight layers. The contributions of this paper are summarized as follows:

- Development of a custom inference accelerator, AIX, specialized to the acoustic model of ASR. In order to fully exploit the DSPs and memory bandwidth of FPGA with high clock frequency as possible, AIX organizes almost all the DSPs into a systolic array. At the same time, the number of columns in the systolic array is chosen to fully utilize memory bandwidth for fetching weights and biases from DRAM.
- Application of a FPGA based inference accelerator to a commercial ASR on Kaldi [15] in datacenter. For the application, a software framework is implemented for managing AIX deployed in the servers of datacenters. Also a thorough optimization is done by carefully scheduling different user utterances for batching. To the best of our knowledge, AIX is the 1^{st} FPGA based inference accelerator commercially used for ASR in VPAs.
- Comparison of performance and power consumption among AIX, CPUs and GPUs by using a real-world ASR workload. In the acceleration of the MLP in ASR, AIX is 4.5 times faster than Intel E5-2620 v4 and 6 times faster than Nvidia Pascal P100 GPU in the real workload of ASR. Xilinx KCU1500 FPGA board is used for AIX. As KCU1500 consumes only upto 75W, the performance per

This research was supported by ICT R&D program of MSIT/IITP [2017-0-00261, Intelligent Many-Core Processor and SW based on Low-Power Hypervisor], IT R&D program of MOTIE/KEIT [10076476, Scalable Machine Learning Acceleration Hardware Technology for Big-Data Servers], and ICT R&D program of MSIT/IITP [2018-0-00195, Artificial Intelligence Processor Research Laboratory

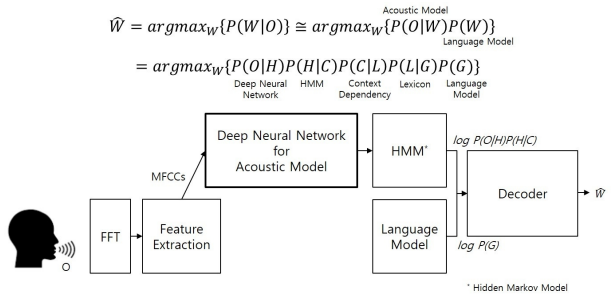

Fig. 1. Block diagram of speech recognition

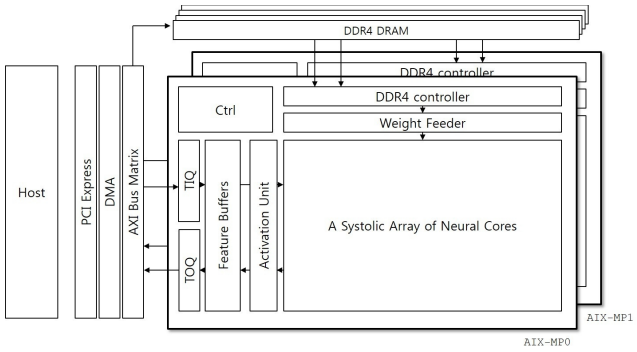

Fig. 2. AIX architecture

watt of AIX is 10.2 to 20.1 times better than E5-2620 v4 and P100. This performance improvement contributes to reduce the number of machines in datacenters to a third.

The remainder of this paper is organized as follows: Section II briefly explains a general ASR algorithm. Section III presents the architecture of AIX. Section IV describes the software framework of AIX for its commercial use. Section V shows the experimental results. Section VI summarizes the related works and Section VII concludes.

II. ASR ALGORITHM OVERVIEW

Before directly diving into the acceleration by AIX, we present the basic algorithm of ASR briefly. ASR converts human voice signal into its corresponding sequences of words in letter. Fig.1 shows the algorithmic flow of ASR. First, a human utterance is sampled and converted into a long sequence of speech features like MFCCs [14] or log Mel-spectral coefficients. With such 120 coefficients, each speech feature represents 25ms second speech signals in frequency domain with a mutual overlapping of 10ms between two consecutive features. The MLP in ASR takes the speech features as an input, and then computes the state probabilities of about 5k phones in a HMM modeling the transitions between phones. In order to augment the correlation among the consecutive speech features, the speech features within $[f_{i-n}, ...f_i, ...f_{i+n}]$ from the current speech feature f_i are combined as a single input with $120 \times (2n+1)$ elements to the MLP at a time. After the state probabilities of the HMM are calculated from the MLP, they are used to predict the most probable sequence of words with the language model in the end by using Viterbi decoding algorithm [5]. Using MLP like this as an acoustic model in ASR is advantageous as the neural network like deep MLP inherently models the non-linearity and correlations among the input features [17]. However, this advantage using MLP in ASR comes with a cost of the expensive computations in MLP. In our profiling, the time consumed in MLP based acoustic model is about up to 70% of the total execution time of ASR. This is one of the major motivations for developing AIX.

III. AIX ARCHITECTURE ON FPGA

Fig.2 shows the architecture of AIX. AIX is implemented on a Xilinx Kintex UltraScale KCU1500 board with KU115 FPGA and four 4GB DDR4-2400 DRAM each with a 64(or 72)-DQ channel. KU115 has 5520 DSP slices, and 75.9 Mb BRAM. The design goal of AIX is to achieve an optimal parallelism on top of FPGA architecture. Firstly, AIX consists of two symmetric multi-processors (AIX-MP0 and AIX-MP1) that can independently run for utilizing two independent silicon dies inside KU115. Each multi-processor consists of input/output queues, a feature buffer, a systolic array of neural cores, activation units, weight feeders, and their controller. As explained in Section I, the target neural network is MLP where the dominant operations are multiplication and accumulations (MACs) in matrix-matrix multiplication. The MACs are executed in the systolic array of neural cores. Therefore, the simultaneous executions of MACs on two multi-processors doubles the performance. Secondly, AIX maximally parallelizes the execution of MACs on each multi-processor. According to [12], [20], DSP slices in FPGA provide higher speed and superior energy efficiency for MACs on neural cores, compared to their general LUT-based design. Therefore, full exploitation of the DSPs with maximal parallelism and high clock frequency as possible is done to squeeze the maximum performance out of FPGA. This will be explained in detail below with the data flow from input to output inside AIX.

Data loading Before operation, AIX is configured and weight matrices and biases are uploaded to the DRAMs of KCU1500 from the host through PCIe bus by dynamic memory access (DMA). During operation, input speech features are pushed to the input queue through PCIe DMA. The input queue can stack up to 40 input features and transpose them so that the systolic array processes them efficiently as a batch. Then, the transposed input features are moved into the feature buffer and are continuously fed to the systolic array. For the succeeding layers of a neural network, it is not necessary to push inputs to input queue, since the feature buffer has two partitions that can work alternatively as input and output so that the output features of the preceding layer is directly used as input to its succeeding layer. Thus, off-chip memory access for loading inputs in succeeding layers can be removed. The weight feeder of each multi-processor reads two dedicated DDR channels, and feeds the parameters like weights and biases to the systolic array at 38.4 GB/s by synchronizing

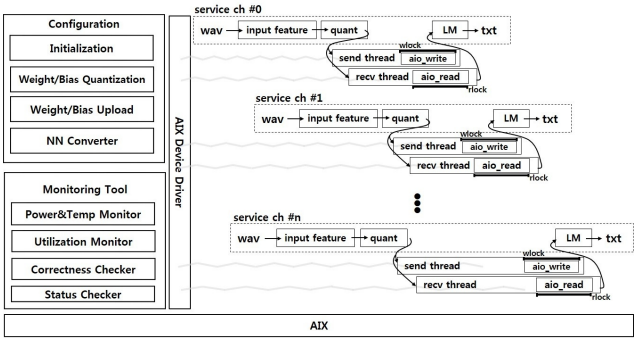

Fig. 3. Software modules for integrating AIX into ASR on datacenters

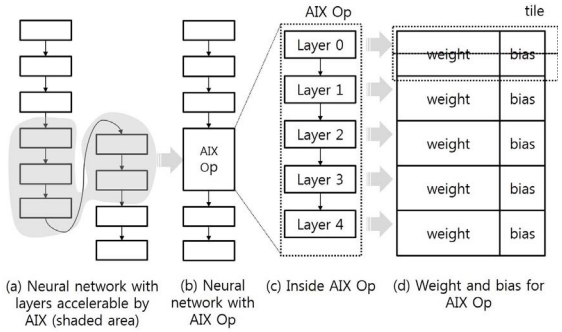

Fig. 4. NN converter for configuring AIX to a neural network

and assembling the read data from the two channels.

Computation Each multi-processor has a systolic array of 64x40 neural cores. The systolic structure removes the necessity of global wires among neural cores, so that clock frequency can be maximized. Also, the size of the column in the array is chosen to fully utilize memory bandwidth provided by two dedicated DDR channels for fetching parameters of weights and biases. Each neural core can execute 16-bit multiplication and 48-bit accumulation in every cycle in a pipelined manner. The inputs and weights are sequentially fed into the systolic array from the top-left neural core to the bottom-right one. After all necessary MACs are completed for the provided inputs, the outputs are sequentially transferred to the activation units.

Postprocessing The activation unit of AIX can emulate many kinds of activation functions popularly used in neural networks like sigmoid, hyper-tangent, and leaky ReLU. To achieve this, a lookup table is prepared where pre-calculated results of each activation function are stored. If an output is requested between two pre-calculated results, we get a more accurate value with interpolation of the two pre-calculated results. After the activation functions are applied, the results are written back to the feature buffer and used as input of the next layer or moved to the output transpose queue if it is the final layer. The output queue also transposes the output to rearrange the final results as the original input order. At last, the transposed data is sent to the host through PCIe DMA.

IV. Integration of AIX into a Commercial ASR

This section explains how to integrate AIX into a commercial ASR based on Kaldi [15] in datacenter. As described in Fig.3, three kinds of software modules are implemented for AIX in Kaldi: one for configuration, another for monitoring, and the others for processing service channels. Before the operation of AIX, configuration runs at once such as initializing AIX and preparing weights and biases of a target neural network. For such preparation, a major part of the neural network is transformed into a single AIX op, as shown in Fig.4. It is done by a tool called NN converter. NN converter automatically identifies the maximal consecutive layers that can run on AIX before the conversion. After the conversion, all the weights and biases needed for the AIX op are concatenated into a raw binary file and uploaded to the DRAM of KCU1500.

During the operation of AIX, every single human utterance turns into its corresponding text inside each service channel with AIX. As explained in Section II, a human utterance is converted into a sequence of speech features. In order not to impose a delay that customers can recognize, every eight speech features from the sequence are grouped into a batch. Then by just writing the batch into AIX, it is requested to process the batch. To fully exploit the data parallelism in AIX, it is desired to process as many human utterances as possible at a time, even from different persons' utterances (from different service channels) as shown in Fig.3. To efficiently achieve this, AIX has a hardware timer to delay the start of the computation in AIX so that the input queue gathers input speech features as many as possible. Aggressive waiting to make the queue full is able to reduce the computation time of all input speech features with more data parallelism. However, a long delay itself may increase the entire processing time despite of the reduction in computation time. Empirically setting 1ms to the hardware timer shows the best performance when there are 16 channels each with a batch of size 8. One disadvantage to this simultaneous processing of multiple input speech utterances is the necessity of mutexes among different service channels. There are multiple batches of input speech features from different service channels in out-of-order way whereas AIX processes the incoming inputs in-order way. After processing in AIX, it is necessary to identify from the inputs of which channel the outputs are generated. This is resolved by keeping the orders of the write transfer for input speech feature, and the read transfer for output with two global mutexes–wlock and rlock. When a DMA write transfer to AIX is needed, a thread, i.e., sender_thread, is assigned to the DMA write transfer with wlock. At the same time, a thread called receiver_thread is assigned for a corresponding DMA read transfer. Once the sender_thread acquires wlock, a nonblocking write (aio_write) is called. Right before the sender_thread releases wlock, the pairwise receiver_thread acquires rlock, calls a nonblokcing read (aio_read), and releases rlock. This way of using the mutexes can keep two kinds of the processing orders with minimizing the idle time in host: read-after-write (RAW) between a pairwise DMA write and read transfers, and first-

TABLE I
ACCURACY OF ASR FROM AIX

	Layer0		Other layers		Normalized SyIER	Normalized SER
	Input precision	Output precision	Input precision	Output precision		
SW-REF	fp32	fp32	i16(11m*)	i16(13m)	1.0000	1.0000
Q1 (AIX)	i16(11m)	i16(13m)	i16(11m)	i16(13m)	1.0240	1.0131
Q2 (AIX)	i16(11m)	i16(11m)	i16(11m)	i16(11m)	1.0135	1.0250
Q3 (AIX)	i16(6m)	i16(11m)	i16(11m)	i16(13m)	0.9925	0.9915
Q4 (AIX)	i16(11m)				0.9940	0.9908
Q5 (AIX)	i16(14m)				1.3448	1.2459

*11m means 11bit for fractional part.

in-first-out (FIFO) among the pairwise DMA transfers from different channels. Finally, the read output probabilities are used by the remaining procedures in ASR as shown in Fig.3.

As explained in Section III, AIX uses 16-bit integer MACs inside neural cores, so it needs to quantize not only the input speech features but also the weights and biases from 32-bit floating point to 16-bit integer. The output from AIX also should be de-quantized. In this quantization and de-quantization, it is critical to keep the accuracy of ASR the same or higher compared to the legacy in service point of view. For this, the accuracy of the ASR accelerated by AIX is evaluated for different configurations. SyIER (syllable error rate, percentage of wrongly predicted Korean letter) and SER (sentence error rate, percentage of wrongly dictated phases) are measured and normalized to SW-REF (a legacy) as shown in Table I. In some configurations using 16-bit integer shows better performance than SW-REF. As 16-bit integer MACs are used, it is initially expected that the quantized calculation in AIX gives an inferior performance. As expected, Q5 shows such a poor accuracy of 1.3448 in SyIER and 1.2459 in SER. On the other hand, Q4 shows the accuracy of 0.9940 in SyIER and 0.9908 in SER, even better than SW-REF. As reported in many previous works like [7], this is an improvement in accuracy after quantization by the effect of regularization in neural network. For the configuration like Q4, AIX has a function of differently configuring the precision of each layer in a neural network. Empirically shown in Table I, it is realized that no overflow of the calculated values in the first layer is more important as relatively many bits (4 bits) are used for integer part. On the other hand, wide dynamic range of precision in other layers is more important as more bits (13 bits in weight) are used for the fractional part.

In order to make a service in datacenter, stability is also important. Therefore, a long run test including all possible corner cases is done more than four months in the same working condition of datacenter. Also a monitoring tool is implemented in order to prevent the abnormal status of AIX from impacting on ASR service quality. The monitoring tool reports four kinds of information about AIX such as power consumption, temperature, resource utilization, and device status. Once an abnormality is found from the monitoring tool, the server with the AIX is immediately out of the service and can be replaced or repaired.

V. EXPERIMENTAL RESULTS

This section represents the experiments demonstrating the performance and power consumption of AIX with the same ASR algorithm as used in the commercial VPA, NUGU. As a configuration, n, explained in Section II, is set to four at the input of the neural network in ASR. For experiments, a proprietary Korean speech dataset is used with 1000 different samples of human utterances whose lengths are 2.045 seconds in average. Intel E5-2620 v4 CPU at 2.1GHz with 16 cores and 256GB memory, and Nvidia Pascal P100 GPU with CUDA 9.1 are prepared for comparison.

A. Implementation on FPGA

Table II shows the resource utilization of FPGA for AIX. Almost all DSP48E2 slices (94.42%) are used for the implementation of the neural cores. Compared to the usage in DSPs, the use of LUT as logic is somewhat low-only 44.14%. They are used to implement the other logics in the neural cores. The other utilization of LUTs for logic and memory comes from the activation units. It is as each activation unit requires a look-up table filled with the sampled data from an activation function, as explained in Section III. The pre and post logics of the look-up table including interpolator are implemented by LUTs and one DPS48E2 slice whereas the table itself is implemented by BRAM. BRAM, used only 46.39%, are mainly used to implement input/output transpose queues and feature buffer. In case of feature buffer, its maximum size should be bigger than the maximum size of activation in a neural network. The size of feature buffer is 1MB, which means AIX can show the best acceleration performance when there is a neural network maximally with 1 MB activation.

B. Performance and power consumption

The performance of AIX is compared with Intel E5-2620 v4 and Nvidia Pascal P100 GPU. Fig.5 compares the response times of ASR in E5-2620 v4, P100, and AIX. The response time means the average time of processing a single speech frame in ASR. As shown in Fig.5, in all range of batch sizes and channels, AIX shows the fastest response time. For example, in 32 channels with the batch size of 16, the response time of AIX is 1.54 ms/frame. This is 4.5 times faster than E5-2620 v4 and 6 times faster than P100. In the case of P100, the response time gets smaller when the batch

TABLE II
RESOURCE UTILIZATION FOR IMPLEMENTING FPGA

	LUT as logic	LUT as memory	FlipFlop	BRAM	DSP
Available	663360	293760	1326720	2160	5520
Used	292781	24910	491916	1002	5212
Utilization	44.14%	8.48%	37.08%	46.39%	94.42%

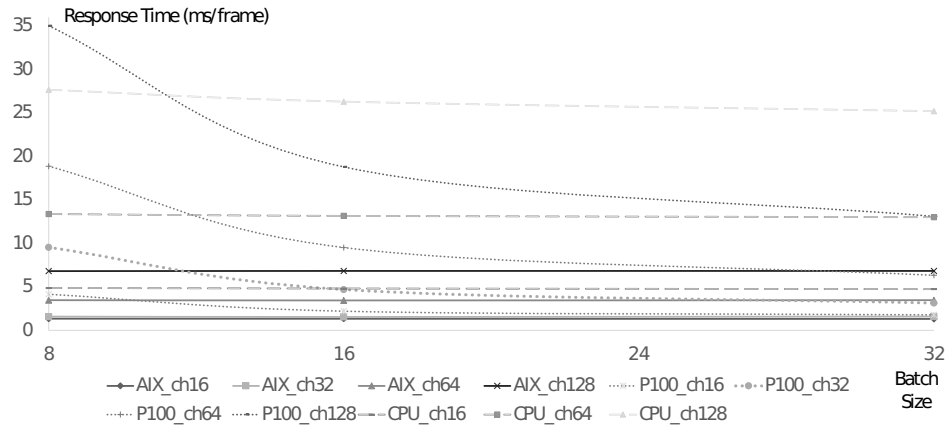

Fig. 5. Comparison of the response time in ASR among E5-2620 v4, P100, and AIX

size gets larger. With the batch size of 32 and 16 channels, the response time of P100 is very close to AIX. However, gathering more speech features into a single batch may delay the response of ASR as previously explained in Section IV. The practical range of a batch size is less than 16, where AIX shows a superior response time. Another advantage of AIX is that the response time of AIX is not rapidly growing with the increase of the channels. This superior comes from the use of input transpose queue implemented in hardware. On the other hand, the response time of E5-2620 v4 and P100 are rapidly growing with the increase of the channels. This means that if there is a plan to service more channels, AIX is the best choice among three. Due to these advantages, after deployed into SKT NUGU VPA service, AIX can reduce the number of servers for the ASR of NUGU in the datacenters to a third. From [1]–[3], the power consumption of two E5-2620 v4, P100 and AIX can be compared. The two physical cores in E5-2620 v4 consume 170W while P100 consumes 250W in their peak performance. AIX consumed only 75W, which is a third of GPU. If the performance (response time) and the power consumption are considered at the same time, AIX is superior to E5-2620 v4 by 10.2 times and to P100 by 20.1 times.

C. Scalability according to the size of neural network for ASR

It is a crucial issue to have scalability in an inference accelerator according to the size of neural networks as more complex deep neural networks are used for high quality services. There are two factors regarding the scalability in AIX for ASR. One is the amount of parameters such as weights and biases from the neural network. The other is the size of batch in the network input and activation internally generated during the network execution. As the weights and biases are uploaded into 16GB DRAM, a huge size of neural networks with a few hundred mega byte parameters can be executed on AIX. In case of input and activation, there is a limit in size as the input/output transpose queue and feature buffer are implemented on a part of 75.9Mb BRAM embedded in KCU115 FPGA. Currently up to 4KB input can be fed into AIX for acceleration. This is so enough in NUGU ASR service as its current neural network has about 100 MB parameters. Even though more complex neural network is used in ASR with a larger size of input, a smaller size of input batch can be used without programming a new bitstream into FPGA for AIX.

VI. RETROSPECT RELATED WORKS

There are many previous works about accelerating neural networks even though there are relatively small number of works regarding the acceleration of the neural networks in speech recognition on FPGA. [4], [7], [13] discuss the FPGA-based acceleration of neural networks that may be used in speech recognition. [7] presents a custom-designed recurrent neural network by using Xilinx XCKU060 FPGA running on 200 MHz with a power dissipation of 41W. It proposes a load-balance-aware pruning method that can compress the LSTM model size by 20 times with negligible loss of prediction accuracy. [13] presented a hardware accelerator for Gated Recurrent Network (GRU) on Stratix V and Arria 10 FPGAs. [4] presented a hardware implementation of LSTM network on Zynq 7020 FPGA from Xilinx with 2 layers and 128 hidden units in hardware, which is 21 times faster than the ARM Cortex-A9 CPU. Among these works, there are no works considering speech recognition system and their

commercial application on ASR. [11] uniquely deals with training acceleration in RNN, which is different from this work. It presents a FPGA implementation framework for RNN based language model accelerating training and improves the parallelism of RNN training scheme. [9], [21] are the previous works developing a speech recognition system. [9] presents accelerated LSTMs on FPGA using massively parallel PEs for low latency and high throughput on FPGA. One top of it, it proposes a speech recognition system where its acoustic model is accelerated by the FPGA while the language model runs on CPUs. It is similar to this work, however, the neural network used in this work is different and there is no consideration about its commercial usage. [14], [21], [22] are related to the acceleration of speech recognition on FPGA but not related to the acceleration of a neural network. [21] presents a hardware architecture for large vocabulary continuous speech recognition that conducts a search over a weighted finite state transducer network. [14] presents a FPGA-based FFT accelerator frequently used in speech recognition. [22] represents a hidden Markov model based 5000-word speaker independent continuous speech recognizer on FPGA.

VII. Summary and Conclusion

This paper introduces an inference accelerator AIX. Its performance and power consumption are evaluated with the modern CPU and GPU. Owing to the architectural novelty, AIX shows up to 20.1 times better than a CPU and GPU in performance and power wise. AIX is now commercially used in the ASR of SKT NUGU VPA. As a future work, the possibility of applying AIX to different kinds of neural network such as LSTM or GRU in natural language understanding domain will be investigated.

VIII. Acknowledgements

This work is done as a result of the co-work with SK Hynix. Special thanks to Yongsang Park and Youngjae Jin for their devotion.

References

[1] Intel. *www.intel.com*
[2] Xilinx. *www.xilinx.com*
[3] Nvidia. *www.nvidia.com*
[4] Andre Xian Ming Chang, Berin Martini, and Eugenio Culurciello. Recurrent neural networks hardware implementation on fpga. *arXiv preprint arXiv:1511.05552*, 2015.
[5] G David Forney. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.
[6] Yijin Guan, Zhihang Yuan, Guangyu Sun, and Jason Cong. Fpga-based accelerator for long short-term memory recurrent neural networks. In *Design Automation Conference (ASP-DAC), 2017 22nd Asia and South Pacific*, pages 629–634. IEEE, 2017.
[7] Song Han, Junlong Kang, Huizi Mao, Yiming Hu, Xin Li, Yubin Li, Dongliang Xie, Hong Luo, Song Yao, Yu Wang, et al. Ese: Efficient speech recognition engine with sparse lstm on fpga. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 75–84. ACM, 2017.
[8] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *Computer Architecture (ISCA), 2017 ACM/IEEE 44th Annual International Symposium on*, pages 1–12. IEEE, 2017.
[9] Minjae Lee, Kyuyeon Hwang, Jinhwan Park, Sungwook Choi, Sungho Shin, and Wonyong Sung. Fpga-based low-power speech recognition with recurrent neural networks. In *Signal Processing Systems (SiPS), 2016 IEEE International Workshop on*, pages 230–235. IEEE, 2016.
[10] Alfie Tan Kok Leong. A music identification system based on audio content similarity. *Oct-2003*, 2003.
[11] Sicheng Li, Chunpeng Wu, Hai Li, Boxun Li, Yu Wang, and Qinru Qiu. Fpga acceleration of recurrent neural network based language model. In *2015 IEEE 23rd Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 111–118. IEEE, 2015.
[12] Yufei Ma, Yu Cao, Sarma Vrudhula, and Jae-sun Seo. Optimizing the convolution operation to accelerate deep neural networks on fpga. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, (99):1–14, 2018.
[13] Eriko Nurvitadhi, Jaewoong Sim, David Sheffield, Asit Mishra, Srivatsan Krishnan, and Debbie Marr. Accelerating recurrent neural networks in analytics servers: Comparison of fpga, cpu, gpu, and asic. In *Field Programmable Logic and Applications (FPL), 2016 26th International Conference on*, pages 1–4. IEEE, 2016.
[14] Shing-Tai Pan, Chih-Chin Lai, and Bo-Yu Tsai. The implementation of speech recognition systems on fpga-based embedded systems with soc architecture. *International Journal of Innovative Computing, Information and Control*, 7(11):6161–6175, 2011.
[15] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*, number EPFL-CONF-192584. IEEE Signal Processing Society, 2011.
[16] Daniel Povey, Xiaohui Zhang, and Sanjeev Khudanpur. Parallel training of dnns with natural gradient and parameter averaging. *arXiv preprint arXiv:1410.7455*, 2014.
[17] Steve Renals. Neural networks for acoustic modeling part1, 2018.
[18] Arman Shehabi, Sarah Smith, Dale Sartor, Richard Brown, Magnus Herrlin, Jonathan Koomey, Eric Masanet, Nathaniel Horner, Inês Azevedo, and William Lintner. United states data center energy usage report. 2016.
[19] Lei Xie and Zhi-Qiang Liu. A comparative study of audio features for audio-to-visual conversion in mpeg-4 compliant facial animation. In *Machine Learning and Cybernetics, 2006 International Conference on*, pages 4359–4364. IEEE, 2006.
[20] Xilinx. *UltraScale Architecture DSP Slice*.
[21] Jungwook Choi, Kisun You and Wonyong Sung. An FPGA implementation of speech recognition with weighted finite state transducers. In *Acoustics, Speech and Signal Processing, 2010 International Conference on*, IEEE, 2010.
[22] Young-kyu Choi, Kisun You, and Wonyong Sung FPGA-based implementation of a real-time 5000-word continuous speech recognizer. In *16th European Signal Processing, 2008 International Conference on*, IEEE, 2008.