International Journal of Applied Engineering Research ISSN 0973-4562 Volume 10, Number 17 (2015) pp 38066-38069

Sharing Computation Resources for Large-Scale Recognition System-on-Chip (SoC)

Seung Eun Lee
Department of Electronic Engineering, Seoul National University of Science and Technology, 232 Gongreung, Seoul, Korea, seung.lee@seoultech.ac.kr

Abstract
In this paper, we investigate image and speech recognition system flows and propose a data-path for both of recognition engines, reducing the hardware cost and power consumption of the recognition System-on-Chip (SoC). Implementation results demonstrate 6.5% of area and 25.5% of power reductions in data-path, compared to the dedicated data-path designs using 90nm CMOS technology. The proposed architecture provides sharing of the integrated SRAM and the control logic in the recognition engines, reducing the area cost and power consumption.

Keywords: System-on-Chip (SoC), Hardware Accelerators, Mobile Augmented Reality, Speech Recognition, Low-power Design, Design Optimization

Introduction

Smart phones and mobile internet devices (MIDs) have gained widespread popularity by placing compute power and novel applications conveniently in the hand of end users. Moreover, emerging smart interfaces based on image recognition, motion/gaze tracking are quickly entering the mobile domain. For instance, Mobile Augmented Reality is an upcoming application [1] that enables users to point their handheld cameras to an object and the device recognizes the objects and overlays relevant metadata on the object. Automatic speech recognition is particularly attractive as an input method on hand-held devices enabling users to make voice calls, dictate notes and initiate searches.

In designing hand-held devices, area and power cost should be reduced while providing the desired functionality and performance. Sharing resources for the different applications or tasks is the simplest and efficient way to reduce the cost [2]. Sharing resources among different applications is required to keep the following things in key focus: (1) need to specify the common operation, (2) need to ensure the data types in the common operation, and (3) need to provide a control flow for shared resources. In particular, the application space such as accuracy, execution time, and throughput should be satisfied for the recognition applications.

In this paper, we investigate image and speech recognition system flows and propose a data-path for both of recognition system, achieving 6.5% of area and 25.5% of power reductions compared to the dedicated accelerator designs, providing the opportunity to share the integrated SRAM and control logic, reducing area and power consumption of the recognition accelerators. We assume that the GMM and MAR workload are not computed simultaneously in order to reuse the proposed data-path for both of workload. In this case, we can share the control logic, which is almost the half of the dedicated hardware accelerator, and integrated SRAM, which should be separated SRAMs for each hardware accelerator [6].

The rest of this paper organized as follows. Section 2 describes recognition workloads of interest, focusing on the computation intensive steps which should be accelerated by adopting hardware accelerators. We propose a data-path for the accelerator in Section 3 and discuss the implementation results including the performance impact on the system in Section 4. Finally, we conclude in Section 5 by outlining the direction for future works on this topic.

Recognition Engines

In this section, we present the recognition workload of interest, mobile augmented reality and speech recognition flows focusing on the computation intensive steps which should be accelerated in order to reduce user response time. Recognition engine enables the real-time operation of recognition application on hand-held platform by accelerating the computing intensive part in recognition workload on the dedicated hardware along with embedded processor.

A. Mobile Augmented Reality (MAR)

Mobile Augmented Reality is an upcoming application that enables users to point their handheld cameras to an object and the device recognizes the objects and overlays relevant metadata on the object. In order to achieve the usage model, it is required to compare query image against a set of pre-existing images in a database for a potential match, performing following three major steps (see Figure 1).

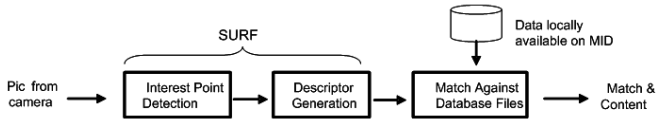

Fig.1. System flow of the mobile augmented reality

i. Interest-point detection identifies interest points in query image.
ii. Description generation creates descriptor vectors for interest points.
iii. Match against DB image: compares descriptor vectors of the query image against descriptor vectors of the DB images.

According to our analysis on the MAR workload, match process is the most computing intensive function, increasing the execution time of the MAR on handheld device [3]. In order to match two images (query and database), we use a brute force match algorithm that exhaustively compares a pair of interest point descriptor vectors from each image based on the Euclidean (a.k.a. L2) distance. Figure 2 describes how a query image from the camera is matched against a candidate image from the data-base. One descriptor represents one interest point. It should be noted that several other match algorithms are available, such as ANN match [4], but the brute force match is simple to implement from a hardware acceleration perspective. The key function is a simple loop, assuming 64 elements per descriptor. For each descriptor of the query image, calculating distances for all descriptors of a database image gives us the minimum and second minimum values of sum. This requires computation of the Euclidean distance square for every pair of descriptor vectors given by

$$sum = \sum_{i=1}^{64} (Q_i - D_i)^2 \quad (1)$$

, where the Q_i and D_i are the i-th descriptor vectors in query image and DB image, respectively.

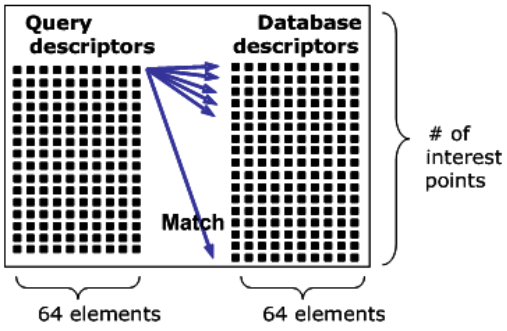

Fig.2. Brute force match of descriptor vectors between a query image and a database image

B. Speech recognition

Automatic speech recognition (ASR) is particularly attractive as an input method on hand-held systems enabling users to make voice calls, dictate notes and initiate searches. Future usage case projections for ASR on embedded systems include natural language navigation, meeting transcription, and web search, just to name a few. These applications are built on large vocabulary, continuous speech recognition (LVCSR) systems. The most widely researched LVCSR software is CMU Sphinx 3.0 [5].

Figure 3 illustrates the basic flow of Sphinx3, which requires three steps: (1) acoustic front-end, (2) GMM scoring, and (3) back-end search. Sphinx3 uses hidden Markov models (HMMs) as statistical models to represent sub-phonemic speech sounds. One key step of an HMM based ASR system is a Gaussian mixture probability density function evaluation which computes Gaussian mixture model (GMM) scores. GMM scoring consumes 82% of total execution time in Sphinx3 on Intel Atom based platform [6]. This GMM requires computation of the sum of weighted squares of distances between audio feature vectors and the mean values in the Gaussian table, followed by score generation. The computation intensive one is the sum of weighted squares of distances given by

$$sum = \sum_{i=1}^{39} (X_i - M_i)^2 \times V_i \quad (2)$$

, where X_i, M_i and V_i are the i-th audio feature vector, the mean and variance value in the Gaussian table, respectively.

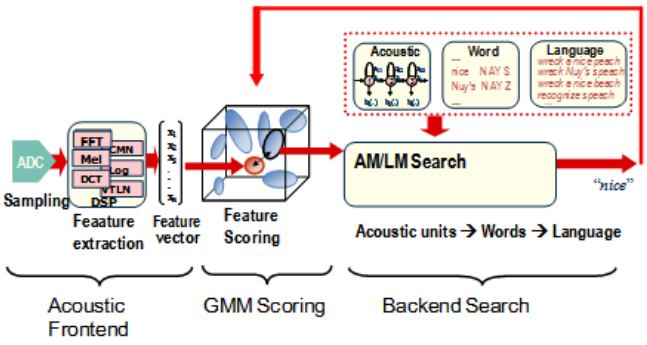

Fig.3. System flow of speech recognition (GMM)

Data-Path in Recognition Engines

In order to improve energy efficiency and execution time for the recognition based embedded system, we proposed and implemented a recognition server (CogniServe) based on heterogeneous architecture with hardware accelerators for the two recognition algorithms described in Section 2, image and speech recognition [6]. For the speech recognition, a GMM accelerator was proposed, calculating the sum of weighted square of the distances between audio feature vectors and the mean values in the Gaussian table, followed by score generation. For the image recognition, we designed two hardware accelerators: (1) a Match accelerator which computes distance calculations for matching descriptors from a query image to descriptors from a set of database image and (2) a Hessian accelerator that identifies the interest points in an input image. In this paper, we propose the hardware architecture which enables the sharing computing resources for both of recognition applications of our interest. Therefore, we focus on the GMM accelerator and the Match accelerator since they have the similar operations as shown in equation (1) and (2).

Figure 4 (a) and (b) show the data-path for MAR and GMM accelerators, respectively. The differences between two accelerators are (1) the GMM accelerator requires one more multiplication in addition to the Euclidean distance calculation between two variables, (2) the number of calculation units is 64 and 39 in the Match and GMM accelerators, respectively, and (3) types of input data (the Match accelerator takes 8bit data and the GMM accelerator takes 32bit data). Based on these observations, we propose a computation unit for both GMM and Match acceleration as shown in Figure 4(c). Four 8bit adders in the Match are used to implement one 32bit adder for the GMM. For the square computation for the MAR processing, one 32×32 and one 64×32 multipliers are used for concurrent calculation of four squares. Thus, four Euclidean distance square for MAR are calculated concurrently when

MAR data-path is activated, and one GMM scoring is computed when GMM data-path is activated. Therefore, initiating 16 elements of the proposed data-path in parallel (see Figure 4(c)) completes the computation of 64 Euclidean distance for MAR workload and the computation of 16 GMM scoring.

In order to complete the computation in equation (1) and equation (2), the accumulation of the results from the multiple instances of the data-paths are required. As described in our previous researches in [3][4], the accelerators have five-stage pipeline to compute the Euclidean distance calculation and GMM scoring. In the pipeline, two pipeline stages accumulate the result from the multiple data-paths. In this paper, we focus on the cost analysis of the data-paths because the accumulation stages are same to both of workload.

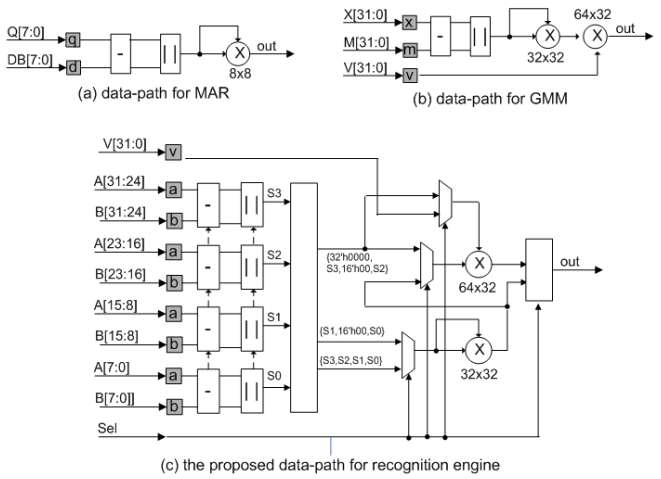

Fig.4. Data-Paths for MAR and GMM accelerators

Implementation

The data-paths were implemented using Verilog HDL and a logic description of our design was obtained by using the synthesis tool from the Synopsys Design Compiler using 90nm technology.

For the practical usage for recognition workloads, 64 instances of the data-path (see Figure 4(a)) and 39 instances of the data-path (see Figure 4(b)) in parallel could provide feasible execution time for MAR and Sphinx workloads, respectively [6]. In our design, we only compute 16 calculations of GMM scoring in parallel in order to share the data-path, SRAM, and control logic with the Match accelerator. Figure 5 summarizes the area and power consumption of data-paths which are required to complete entire computing. In this work, we extracted the physical characteristics by using the Synopsys Design Compiler, which has limitations such as not including the interconnection area in detail and dynamic power consumption with actual switching information under actual workloads. For the more precise comparison, the analysis should be performed by using area information obtained from layout and dynamic power consumption from the Synopsys Prime Time. However, we believe that our analysis provides a feasible comparison in terms of physical characteristics of our proposal. In case of dedicated design, 64 MAR and 16 GMM data-paths costs 0.88mm^2 and 16 instances of shared data-path (MAR+GMM) occupies 0.82mm^2, reducing 6.5% of area cost in data-path. In terms of power consumption, dedicated accelerator designs consume 198mW and the proposed design consumes 158mW, reducing 25.5% of power. The accelerators employ SRAM as a staging buffer in the address space, without which all subsequent values would need to be read from DRAM on at the time. Our approach also provides the opportunity to share SRAM and control logic, reducing the area and power consumption of the integrated memory.

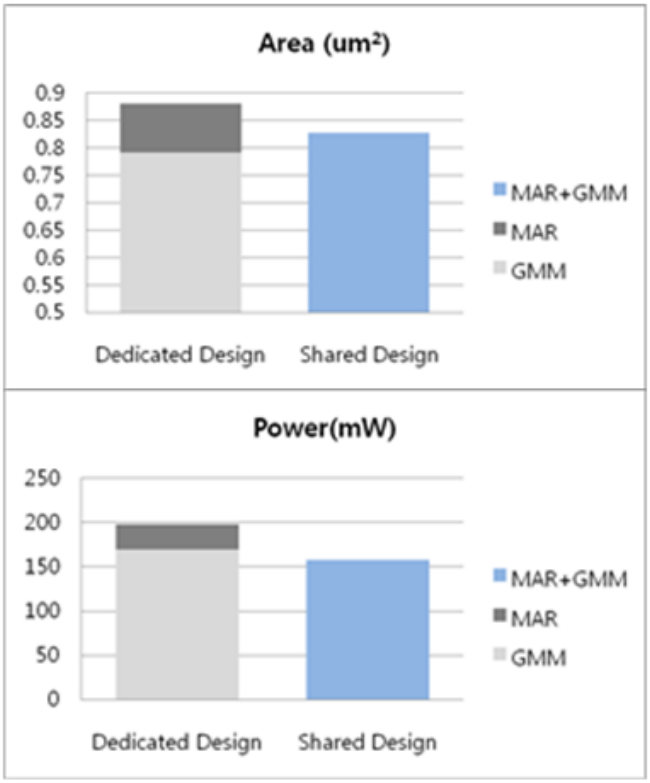

Fig.5. Area and power benefit of our proposal

In case of MAR, the operating frequency of the proposed data-path is reduced to almost half of the dedicated MAR accelerator. However, 400MHz -- which is our target frequency of the hardware accelerator along with a processor -- operation of the MAR accelerator provides reasonable user response time which is in milliseconds [3]. In case of the GMM scoring, only 16 calculations are processed in parallel in our design, requiring three iterations to finish the computations of entire 39 elements. Our initial analysis reveals that iterating three times for GMM calculation does not reduce the recognition time by adopting the grouping factor (*N*). In GMM calculation, instead of iterating over all senones for the current feature vector, the control unit loads and processes *N* feature vectors before loading the next senones. The *N* score's are written to system memory before the *M*, *V*, *LRD* and *W* data in the SRAM are reloaded (*LRD*: logarithmic reciprocal determinant, *W*: database vectors). Since the Gaussian table for the thousands of senones accounts for the vast majority of memory traffic, this audio

frame grouping technique can reduce memory bandwidth to 1/*N* of the original algorithm. Complexity of the control logic for this optimized processing order is the same as for the original algorithm. The only downside is an extra initial delay of *N* audio frames. In our default sampling granularity of 10ms per audio frame, *N*=8 will cause an initial delay of 80ms, which is hardly perceivable to the human being. This grouping technique was proposed by Mathew [7] and used in a number of GMM software optimizations and hardware accelerators [8, 9, 10].

Conclusion

In this paper, we investigated recognition algorithms and presented an opportunity to share the computation resources on MPSoC design by proposing the unified data-path, reducing area and power consumption. We plan to continue studying additional recognition workload and further refining the computation unit in order to reduce the hardware cost in MPSoC. We expect that sharing resources among different applications on embedded systems will bring forth a new spectrum of optimizations for emerging digital systems.

References

[1] Takacs, G. et. al, "Outdoors augmented reality on mobile phone using loxel-based visual feature organization," in 1st ACM international conference on Multimedia information retrieval, MIR '08, pp. 427-434, 2008.

[2] Lee, S. E., "Sharing computation resources in image and speech recognition for embedded systems," Communications in Computer and Information Science, vol. 215, pp. 150-156, 2011.

[3] Lee, S.E. et. al., "Accelerating mobile augmented reality on a handheld platform," in IEEE International Conference on Computer Design, pp. 419-426, 2009.

[4] ANN, http://www.cs.umd.edu/ mount/ANN

[5] CMU Sphinx: The carnegie mellon sphinx project, http://cmusphinx.sourceforge.net/html/cmusphinx.php

[6] Iyer, R. et. al, "Cogniserve: Heterogeneous server architecture for largescale recognition," in IEEE Micro, vol. 31, no. 3, pp. 20-31, 2011.

[7] Mathew, B., Davis, A., Fang, Z., "A low-power accelerator for the sphinx 3 speech recognition system, " in 2003 international conference on Compilers, architecture and synthesis for embedded systems, pp. 210-219, 2003.

[8] Ma, T., Deisher, M., "Novel ci-backoff scheme for real-time embedded speech recognition," in IEEE International Conference on Acoustics Speech and Signal Processing, pp. 1614-1617, 2010.

[9] You, K., Choi, Y.k., Choi, J., Sung, W. "Memory access optimized vlsi for 5000-word continuous speech recognition," Journal of Signal Processing Systems 63, pp. 95-105.

[10] Lin, E.C., Yu, K., Rutenbar, R.A., Chen, T.. "A 1000-word vocabulary, speaker-independent, continuous live mode speech recognizer implemented in a single fpga," Proceedings of the 2007 ACM/SIGDA 15th international symposium on Field programmable gate arrays, FPGA '07, pp. 60-68, 2007.