Speech Recognition on SPHERIC — An IC for Command & Control Applications

Dieter Geller, Markus Lieb, Wolfgang Budde, Oliver Muelhens, Manfred Zinke

Philips GmbH Forschungslaboratorien Aachen
P.O. Box 500145, D-52085 Aachen, Germany
E-mail: {geller,lieb,budde,muelhens,zinke}@pfa.research.philips.com

ABSTRACT

SPHERIC is a new IC that has been designed specially for automatic speech recognition applications in Consumer Electronics with a vocabulary of up to 126 words.
It allows real-time recognition of both speaker dependent and independent words spoken continuously or in an isolated way. Key word spotting and playback of coded messages and user trained words are additional features.

After a short system overview the hardware architecture and software structure are presented in this paper. The techniques for reducing computation time and necessary memory size are examined in more detail. Finally, the implemented speech recognition algorithm is described.

1. INTRODUCTION

In recent years several companies have introduced new ICs incorporating automatic speech recognition (ASR) for small vocabulary applications. Most of them cover vocabularies of less than 20 words and allow only speaker dependent recognition. Therefore, the experiences in the field of DSP development and all areas of speech recognition within Philips were brought together for developing SPHERIC – an IC capable of recognizing human speech with good performance, flexibility in application design and usability while simultaneously representing a low cost solution. The name SPHERIC is a shorthand for "Speech Recognizer Integrated Circuit". For both implemented algorithms and hardware parts the minimization of necessary memory and computational resources while preserving recognition performance and fulfilling the real-time requirement was the essential goal.
Examples of operation areas are controlling of mobile phones via digit or name dialling, car navigation or remote control of telephone answering machines.

2. SYSTEM OVERVIEW

A vocabulary size of up to 126 words being recognized simultaneously was considered sufficient for most command & control applications. By fast and simple vocabulary and context switching the total number of words that can be recognized by a SPHERIC application may be increased in a flexible way. The recognizable words are arranged in a finite state grammar structure defined by the application designer.
All program modules and data needed for speech references, grammars, vocabularies and parameters are stored outside the chip in ROM or Flash memory in order to allow for easy modification of applications without the need of redesigning the chip's hardware or firmware.
A training program module allows speaker dependent words to be trained. Additional program modules for speech coding and decoding enable playback of prerecorded messages or speaker dependent words. The program modules can be downloaded from external memory into the chip's program RAM just when needed, thus keeping the on-chip program memory size small.
For flexible speech I/O a variety of digital and analog audio interfaces is available on-chip. SPHERIC communicates with an external microcontroller via an I^2C-link (Inter-IC Communication). Many controller chips are already equipped with this world-wide accepted, serial 2-wire link or can at least emulate it by software. For this communication an application programming interface was developed and is part of the chip's software.

3. HARDWARE

3.1 General Description

All parts necessary for speech recognition except for

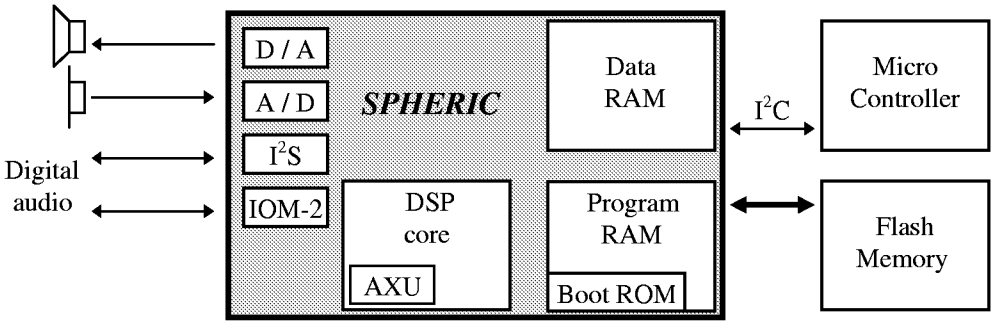

Figure 1: System Overview

the reference/program memory are integrated in the chip as shown in figure 1.
Around a Philips-internal 16x16 bit fixed-point DSP-core with double Harvard architecture [1] various interfaces for communication and an application specific execution unit (AXU) specially designed for ASR have been implemented. Beside analog I/O via A/D- and D/A-converters a world-wide accepted, digital I^2S-interface (Inter-IC-Sound), a serial 3-wire link, is available for externally connected audio converters. Optionally an IOM-2 interface (ISDN-Oriented Modular interface, rev. 2) can be used for digital telephone applications.
The I^2C-interface provides access to the on-chip program and data memories for system control with transfer rates of up to 400 kbit/sec.

3.2 Application Specific Execution Unit (AXU)

The bottleneck of an ASR task with good performance concerning computation time is mainly the log-likelihood distance calculation. In our case, Laplacian mixture densities were found to yield the best performance-to-effort ratio. The distance $D[j]$ of one of these density references $\vec{R}[j]$ to the current feature vector $\vec{X}$ with N components calculated for each speech frame is:

$$D[j] = \sum_{i=1}^{N} |X[i] - R[i,j]|$$

Without hardware accelerator, the pure calculation of one partial distance would require at least 3 cycles for calculation of difference, absolut value and accumulation including data fetching on DSPs comparable with the SPHERIC core. The implemented AXU reduces the necessary cycle budget for distance calculation in total to 11 %.
In a similar way the AXU supports the calculation of the resultant mixture density distances $M[k]$ with the help of density weights $W[i,k]$ and the density distances $D[i,k]$ contributing to this mixture for the Viterbi approximation:

$$M[k] \approx \min_i (W[i,k] + D[i,k])$$

As a third benefit, the AXU in SPHERIC allows for memory efficient and fast packing and unpacking of data structures used in the ASR algorithm. So no further calculation effort must be spent to separate data items within these structures.
The additional hardware in terms of silicon area for this AXU is nearly negligible compared to the other parts of the chip.

4. SOFTWARE OVERVIEW

Figure 2 gives a survey of the implemented software. It can be divided into the ASR application part, a rudimentary operating system and a toolset providing several utilities. Written in assembly language in a modular form, program blocks can easily be exchanged or modified, if necessary.
A bootstrap ROM program is responsible for initialization after power-up or reset. A self-test procedure and then program download – either from external memory or via the I^2C-link – are performed, and program execution is initiated.
For the communication between SPHERIC and an external controller an application programming interface (API) has been defined. It allows the controller to send single commands or command sequences the SPHERIC main control program has to execute like "START_RECOGNITION" or "DOWNLOAD_TRAINING_PROGRAM". These commands are directly written into a command buffer located in SPHERIC's data RAM via I^2C-link. Similarly, SPHERIC writes messages like "RECOGNIZED_WORD_N" into a message array. A parity protection mechanism prevents executing undesired commands in case of bit errors during transmission. The internal command and message buffers are polled or written to by the main program and the controller regularly; so, every 16 ms, the rate of speech frame processing, an information exchange is possible.

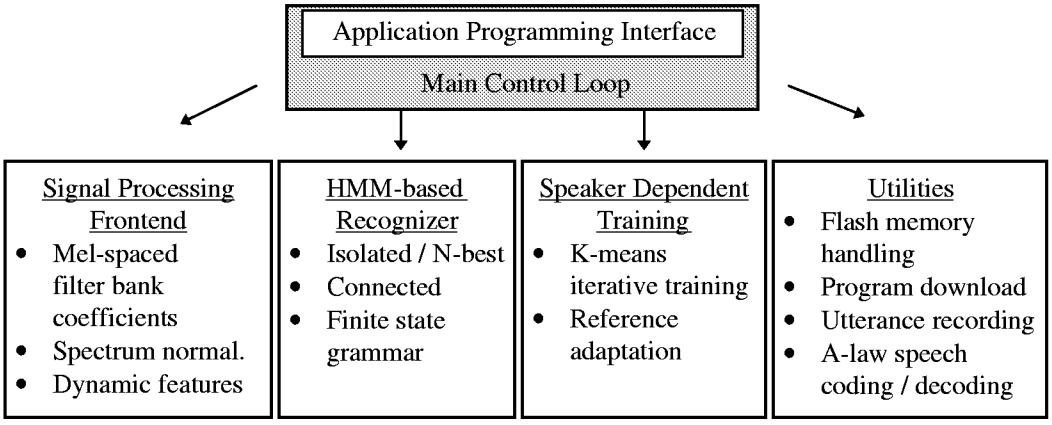

Figure 2: Software Structure

Not only the ASR functionalities, which are described in the following chapter in more detail, but also the utility toolbox is accessible via API commands :
The routines for supporting Flash memory handling perform erasing and writing this memory type, e.g. to store references of a new trained speaker dependent vocabulary word or to modify the grammar structure used for recognition.
The download utility allows the external controller to initiate downloading of a new program module from external into program memory whenever needed.
Speech coding and decoding facilities using A-law compression are available to store and playback samples of user trained words, e.g. for the verification of recognition results. Also, playback of prerecorded messages is supported.

The ASR software – the predominant software part – is able to perform the following tasks:

- Real-time speaker independent speech recognition with factory trained references
- Real-time speaker dependent speech recognition
- Isolated word recognition with N-best output
- Recognition of continuously spoken word sequences
- Key word spotting
- K-means training with 2 training utterances per word
- Reference adaptation to improve recognition of user trained words
- Playback of prerecorded messages and samples of user trained words

5. SPEECH RECOGNITION ALGORITHM

The whole ASR algorithm is divided into three major blocks, namely signal analysis, recognition and training. In the following, a compressed description of all three parts is given. For more detailed information refer to [2], [3] and [4].
The on-line recognition processing is frame based, that means after executing the signal analysis module for one speech frame the recognition module follows to do the search process for this frame. Another strategy is used for the training module: A recording phase in which feature streams of training utterances are extracted precedes the off-line training module.

5.1 Signal Analysis

The signal processing front-end provides a frame width of 32 msec., i. e. 256 samples at 8 kHz sampling frequency and an overlap factor of two, which leads to a frame rate of 62.5 per second. It is based on mel-spaced FFT filter bank coefficients, provides channel compensation via spectral normalization and includes dynamic features.
The first processing stage is a one tap FIR preemphasis filter to emphasize high frequency parts. The samples are then Hamming-windowed. A real valued 256-point Fast Fourier Transform is calculated in form of a complex 128-point FFT. The interleaved complex FFT output values are sorted and finally the squared power spectrum is calculated.
At several stages throughout this first part of the feature extraction process a scaling of the data is performed: the signal values are shifted left by a number of bits to assure the processor's numeric range being used optimally.
From the squared power spectrum values frequency band components are formed by convolution with mel-spaced triangular kernels. Then the logarithms are calculated from the filterbank outputs. Additionally, the previous scaling shifting operations are

corrected by subtraction in the logarithmic domain. Next processing steps contain clipping onto a minimum floor value, which can be regarded as background noise masking, and spectral sharpening; the latter meaning a special distortion of the triangular smoothing shapes, by subtracting the weighted neighbour bands from each filterbank output. Normalization is done in two senses: all components are normalized on the mean feature value, which in turn is added to the feature vector as an additional component representing the signal energy. Spectrum normalization for compensation of channel distortions and additive noise is done with a recursive filter of first order on each component[3]. Dynamic features of first order complete the feature vector which is finally scaled by a global pooled variance vector determined during offline speaker independent training.

5.2 Recognition

The HMM recognizer is based on Laplacian mixture densities for speaker independent and on single densities for speaker dependent vocabulary words. The structure of the HMM network is limited to the three possible state transitions "loop", "next" and "skip". The linear search procedure is based on a finite state grammar built up of whole word models. In isolated word recognition mode, an N-best list is output together with score differences against the word with highest probability, so that an application can include simple rejection strategies based on difference thresholds. During connected word recognition partial traceback is performed: word sequences are output as soon as surviving partial string hypotheses are unambiguous. Therefore, no explicit speech or utterence end detection is needed. To prevent from exceeding the numeric range, all active states are normalized onto the global best score with every frame processed. A program module for flexible and efficient pruning of hypotheses with low probabilities minimizes the resulting calculation effort. To achieve compact data representation and thus memory efficiency, much effort was spent on considerations on data quantization. To reduce, for instance, the data memory needed for the search, both the score and backpointer information for each HMM state of one active word are packed into one 16 bit data word.

5.3 Training

The procedure for training speaker dependent vocabulary words is split into two modules. The reference estimation phase is preceded by a recording phase in which the raw feature vector streams are extracted from the user utterances and stored for further processing. Typically, two utterances are sufficient to achieve good recognition performance. First processing step on the recorded speech patterns is an endpoint detection. In order to determine the pure speech part within the utterances, parameters of two Gaussian models – one for speech, one for silence – are estimated that maximize the segmentation likelihood [5]. The number of HMM states is calculated depending on these detected lengths. Initial linear segmentation within the resulting speech parts is followed by an initial reference parameter estimation and the k-means iteration process resulting in the final word model parameters. Single Laplacian densities are applied in speaker dependent vocabulary. In combination with the option of sharing one density model between several tied states, this leads to compact word references to store for recognition. The optional adaptation of a word model by additional training utterances copes with varying noisy background and speaking styles.

6. CONCLUSION

We have given a survey of SPHERIC, an integrated circuit specially suited for small vocabulary speech recognition tasks. The characteristics of both the chip hardware and the implemented software have been sketched, whereas the ASR modules have been considered in some more detail.

REFERENCES

[1] R. Woudsma, R.A.M. Beltman, G. Postuma et. al.: "EPICS, a Flexible Approach to Embedded DSP Cores", *Proc. of The Int. Conf. on Signal Processing Applications and Technology*, Vol. 1, pp. 506-511, Oct. 1994.

[2] D. Geller, R. Haeb-Umbach: "Improvements in Speech Recognition for Voice Dialling in the Car Environment", *Proc. ESCA Workshop on Speech Recognition in Adverse Conditions*, Cannes, France, pp. 203-206, Nov. 1992.

[3] R. Haeb-Umbach, D. Geller, H. Ney: "Improvements in Connected Digit Recognition Using Linear Discriminant Analysis and Mixture Densities", *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Proc.*, Minneapolis, MN, Vol. 2, pp. 239-242, April 1993.

[4] R. Haeb-Umbach, P. Beyerlein, D. Geller: "Speech Recognition Algorithms for Voice Control Interfaces", *Philips Journal of Research*, Vol. 49/4, Eindhoven, Netherlands, pp. 381-397, Dec. 1995.

[5] J. Bridle, N. Sedgwick: "A Method for Segmenting Acoustic Patterns with Applications to Automatic Speech Recognition", *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Proc.*, Hartford, Conn., pp. 656-659, May 1977.