IMPLEMENTATION OF A DYNAMIC TIME WARP INTEGRATED CIRCUIT FOR LARGE VOCABULARY ISOLATED AND CONNECTED SPEECH RECOGNITION

Riccardo Cecinati (*), Alberto Ciaramella (**), Luigi Licciardi (**), Giovanni Venuti (**)

(*) ELSAG - Via Puccini 6 - 16154 Genova (Italy)
(**) CSELT - Via G. Reiss Romoli 274 - 10148 Torino (Italy)

ABSTRACT
A new custom VLSI circuit for dynamic time warping is described; it can be used for both isolated and connected speech recognition, with both templates and discrete hidden Markov models (D.H.M.M.).

The chip has an Harvard architecture, with an internal program ROM horizontally microcoded and an external data memory up to 16 Mbytes: an internal cache relaxes speed requirements for the external data RAM.

A separate address generation block and an arithmetic unit specialized to the dynamic programming task complete the internal chip structure.

With a clock of 16 MHz, the microinstruction cycle is of 126 nsec.; we have evaluated a task speed up of about 5 in comparison to a DSP implementation.

The chip is implemented in a 3 micron 2 metal levels CMOS technology, with a complexity of 70K equivalent transistors and a die size of 8 mm. by 5.5 mm.; it is housed in 68 pins LCC package.

1. INTRODUCTION
Speech recognition heavily relies on dynamic time warping (DTW) algorithms, so an efficient DTW stage increases the task affordable by a minimal system or reduces the system complexity required by a demanding task.

The efficient computation of DTW requires the availability of some special hardware for data and address manipulations that is not normally present on today general purpose DSPs: hence some custom chips have been implemented, mainly in research laboratories, as for example [1,2].

Here is described a new DTW custom chip, named RIPAC, that is an acronym for Riconoscitore del Parlato Connesso, i.e. "connected speech recognizer".

In a speech recognition system the DTW algorithm can be split into two levels, where the lower level is charged with the recognition of whole words or subwords selected by the higher level, according to the grammar [3]: the higher level is in general less computational demanding, though less regular, while the low level is in general more computational demanding, but more regular in program control and in memory accesses; hence the low level is well suited to a custom implementation.

This work was partially supported by the Italian Ministry of Scientific and Technological Research in the framework of the National Microelectronics Plan.

2. ALGORITHM AND DATA MEMORY ORGANIZATION
Fig. 1 shows the functional block diagram of a RIPAC based recognition system: RIPAC accesses both the words/subwords description area and the working area, interchanging data with the master via broadcast and mailbox areas.

The high level grammar describes the words/subwords interconnection in the task: each arc in the grammar is associated with a word/subword.

At the low level each arc is expanded into a chain of ordered acoustical states regularly interconnected: in fact it is only possible to loop on a state, make a transition to the next state or skip a state.

We name active a state whose score is less than a given beam-search threshold (the score represents a distance in the templates case and a probability logarithm complement in the D.H.M.M. case); we name active a grammar arc with at least one active state. Active states and active arcs change frame by frame.

When the last state of an arc becomes active, in this frame the hypothesis that the corresponding word terminates can be issued: this is notified to the high level with a "pop" message, sent through the corresponding mailbox area. Based on this, the high level knows the arcs which will become active in the following frame, and notifies them to the low level with "push" messages, sent through the corresponding mailbox area.

Low level has the control of active arcs: an arc becomes inactive when the scores of all its states are over the threshold, becomes active when the high level sends a "push" message.

Dynamic programming on an active arc is performed in vector form for all states from the first to the last active; inactive states in this activation range are evaluated too, in order to maintain the algorithm vector organization: the saturated arithmetic allows not to differentiate these cases in the algorithm control.

DTW can be performed both for templates and for D.H.M.M.; the overall control in fact is the same and some difference exists only in evaluation of states scores [3].

In the following is summarized the DTW algorithm performed at the low level for each frame t: for each active arc do

if this arc is in the pushlist then

based on the pushlist information

set the score of the first state activate the first state of the arc cancel the arc from the pushlist endif

perform dynamic programming from the first to the last active state

update also the active states range

if all the states are below a threshold the arc becomes inactive enddo
for each arc in the pushlist not considered till now do
activate the arc based on the pushlist information set the score of the first state
activate the first state of the arc cancel the arc from the pushlist
enddo

The data memory is organized into static and dynamic blocks.

Static block contains the words/subwords description, which is organized into 4 tables: the first, named TDIR, is accessed by the word index and acts as a directory to the transition matrix A, spectral emission matrix B and energy emission matrix C [4,5]; these two last matrices are used only in the D.H.M.M. case.

In the templates case the matrix A contains also the spectral codebook symbol; in fact we have compared 4 cases, i.e. to quantize or not the input and to quantize or not the template and we have chosen to quantize the template only, since it does not show an appreciable recognition accuracy degradation in comparison to the more complete unquantized case and it greatly simplifies computations [3].

The most important table of the dynamic block is the active block, which essentially stores an information file for each active arc.

An active arc information file is composed by an head record, containing the arc identity (defined on it and by its occurrence in the grammar) and the positions of the first and last active state of this arc, followed by records containing the cumulated distance and the backpointer of ordered active states.

These files are organized as a circular list, of which the TOP and the FREE pointers define the used area; the file updating of an active arc cannot be performed in place, since the file length can increase, decrease or even disappear frame by frame: hence the updated file is transferred from the top to the bottom of the list, updating also TOP and FREE information, as shown in fig.4. This arrangement reduces the amount of WORK area needed; although for the WORK area a maximum amount of 32K words can be reserved, corresponding to about 16K active states for a maximum of 1K active arcs (and this seems adequate for demanding tasks also), a run time control on its occupancy is a prerogative of ordered active states.

In the case that the WORK area is full more than 75%, a warning interrupt is issued to the master, which for example could rise the beam-search threshold; a 100% WORK area full instead is notified to the master as a not recoverable error.

RIPAC knows the allocation of the data memory area used at run time: in fact the master first initializes a data area, called address block (ADB), containing the start addresses of each file in use, then writes into RIPAC the start address of ADB and finally issues the transfer of this area inside RIPAC.

After initialization, the master frame by frame activates RIPAC; for each activation RIPAC becomes the bus master, reads from external memory the present frame spectral and energy values and algorithm threshold, and then starts the DTW algorithm on the frame: at the end the bus is released and the master can read final information (as for example "posts" in suitable area).

3. RIPAC INTERNAL ARCHITECTURE
RIPAC internal architecture is Harvard like, with separate address circuits for program and data memory; its block diagram is shown in fig. 3.(*)

Program memory is contained in an internal 256 instructions ROM, 25% of which are dedicated to diagnostic routines; each instruction is coded on 43 bits, hence the chip is horizontally microprogrammed, with a microinstruction cycle of 125 ns.

An internal sequencer controls the microprogram ROM, allowing also conditional jumps and nested calls; the sequencer is also interfaced to the external bus so that the master can force the 8 bit microprogram starting address.

A command word can differentiate the execution of some of the microcoded routines:
as an example a bit in this word selects whether to perform the DTW for templates or for D.H.M.M.

Data memory is external, with a maximum address range of 16 Mbytes; this large address space can be useful especially for D.H.M.M. computations.

External data memory can be implemented by densely packed and low cost RAM chips with access time up to 200 nsec. without imposing wait states to the internal microprogram; maximum efficiency has been obtained by using an internal data cache of 64 words, where more heavily used information and/or intermediate results are stored, and by suitably pipelining external accesses. A dedicated address processor speeds up address computations for the data structures used in the external data memory: since these are organized as files, an internal RAM allocation table, loaded at initialization time, contains the base addresses BASE(M) and the RECORDLENGHT(M) information of each file M of the 16 used, then, using the record index I and the position J of the word in the record, the external RAM address is computed as:

\[
\text{RAMADDRESS}(M,I,J) = \text{BASE}(M) + \text{RECORDLENGHT}(M) \times I + J
\]

Since RECORDLENGHT(M) has been chosen as a power of 2, the computation RECORDLENGHT(M) \times I requires only a programmable functions, hitter.

Address processor can perform also other functions, for example work area circular addressing, as explained before.

An independent internal 12 bits unsigned saturated ALU is dedicated to the accumulate--compare operations typical of the dynamic programming: when an overflow occurs the saturated logic forces the result at the full scale value.

To fully exploit the horizontal microinstruction, internal blocks are interconnected with 216 bits internal buses.

A diagnostic unit supervises the component: in case of an hardware fault or in case of load critical situations an interrupt is issued from RIPAC to the master; a master readable status word allows to differentiate between different interrupt reasons; some dedicated I/O pins allow the off line checking of this unit as well.

In order to reduce the number of pins to 68 and also to simplify the system control, the chip is interfaced to a single external bus, 286 like; RIPAC, the high level processor and eventually also a dynamic RAM controller share the mastership of this bus to gain access to the data RAM.

The component has a complexity of 70K equivalent transistors and is implemented in CMOS 2 metal levels 3 micron technology with a die size of 5.5 mm. by 6 mm., a dissipations of less than 0.5 Watt, a clock cycle of 16 MHz. and an instruction time of 125 nsec.: fig. 4 shows the chip layout.

(*) Patent pending
4. THROUGHPUT EVALUATIONS

An estimation of the maximum RIPAC throughput can be given by the execution time of the dynamic programming loop for a single state. This loop takes 22 microinstructions of 125 nsec. each; hence we update a state each 2.75 usec.

This means that with a fixed frame rate of 10 msec. we could follow in real time a task with an average of 3.6k active states per frame, but this is an upper limit, since the frame DTW algorithm exhibits some overhead for the transition from an active to the next active arc, for the activation and deactivation of arcs and so on, as previously detailed; the overhead is not a constant, but depends mainly on the number of pushes.

Fig. 5 reports the evaluation of the maximum number of active states which can be supported in real time with a time frame of 10 msec. by one RIPAC; this number decreases with the increasing of the number of pushes. We have distinguished two limit cases: the line a in which all the pushes are performed on an already active arcs and the line b in which all pushes are performed on previously inactive arcs. Given the arc activation procedure this last case introduces in fact more overhead.

The number of active arcs which can be followed in parallel depends of course on the grammar description: assuming for example that the arcs represent words, and that words are described by 40 states in average and that the range of active states is 20 in average, a number of active states of the order of 2.6 K means that we can follow in real time 130 active arcs in parallel with a single chip; hence we can follow in real time a task described at a word level with a quite large branching factor.

Another possible use of the chip is in a large vocabulary recognition application, in which it is convenient to use arcs representing subwords of 3 or 4 states in general [5]; since for this case we have also a DSP implementation, on a TMS2020, a speed comparison between the two technologies can be done [6]. We found that DTW for a subword is executed in 50 us, as average with the DSP technology [8] and in 10 us. as average with the RIPAC, with 5 times speed up.

We point out however that this is a very cautious comparison, with some handicap on the RIPAC side, since:

a) the RIPAC can use low cost relaxed access time RAMs, instead the DSP must use high cost low access time RAMs for reaching this performance,

b) the DSP implemented algorithm has been optimized for the subword case, taking also into account some shortcuts, which increase the code, the RIPAC algorithm used is instead the general purpose vectorialized program.

5. CONCLUSIONS

The DTW implemented by the custom VLSI RIPAC is at least 5 times faster than a DTW implemented in DSP technology and hence its use can both raise the task complexity followed in real time by a minimal system and reduce the system size for a complex task, as for example a large vocabulary connected word recognizer.

Since utility and application programs are fixed in internal program ROM, these details are not in charge of the user; this architectural choice is also in favor of the speed. This choice does not imply a too rigid system however, since RIPAC functionality can be easily changed by writing the command word and other suitable data; this way for example we can set the DTW for templates or for D.H.M.M., which show only minimal computation differences.

The RIPAC architecture high throughput is due to the cooperation of 3 parallel units: the microprogram sequencer, the accumulate-compare optimized arithmetic unit and the external data memory address processor, with vectors and circular lists addressing capabilities.

The use of external RAMs of relaxed access time without adding time penalties allows the use of low cost and densely packaged RAMs, with an obvious simplification of the final system: in a complete recognition system in fact the RAMs are not to be overlooked.

ACKNOWLEDGE

A custom VLSI implementation involves many experts: hence we acknowledge R.Fossati, A.Pacchiotti, M.Paolini, R.Tasso, A.Torrielli M.Fassino (CSELT) and M.Silveri (ELSAG) which have detailed the design from the functional specification to the final electrical realization and C.Enrione and F.Goria (CSELT) which have designed the final layout. We acknowledge also C.Vicenzi (ELSAG) and L.Fissore (CSELT) for their early contributions to the RIPAC implemented algorithm and for accuracy simulations and G.Babini and M.Gandini for their technical management work.

REFERENCES


C.Gervesh, G.Wilson, J.Kumar
The Graph Search Machine (GSM): a VLSI Architecture for Connected Speech Recognition and Other Applications IEEE proceedings, vol 75, n.9, September 1987, pp.1172-1184


Synatx Driven Recognition of Connected Words by Markov Models Proc. ICASSP84, San Diego, 1984


Dynamic Programming with Hidden Markov Models choice of TMS32020 Digital Signal Processor EUSIPCO-88 (Grenoble, 6-8 September 1988)
Fig. 1 - System block diagram

Fig. 2 - Examples of circular file updating on an active arc

Fig. 3 - RIPAC block diagram

Fig. 4 - RIPAC layout

Fig. 5 - Average throughput of a RIPAC, with 10 ms frame