24-CHANNEL 32 Kb/s ADPCM TRANSCODER USING the CCITT RECOMMENDATION G.721

Sami Aly

BNR
P. O. Box 3511 station C
Ottawa Ont., Canada K1Y 4H7

ABSTRACT

The 32 kb/s ADPCM standard algorithm has been published in the CCITT recommendation G.721. Implementations for this algorithm have been proposed in the form of a per-channel transcoder. This paper presents an alternative implementation which is preferred for multi-channel applications. It uses a combination of CMOS EPROM lookup table and CMOS gate array processor technologies.

The paper presents the transcoder architecture which is based on a multiprocessor concept. The hardware block diagram and the firmware structure are discussed. System design and test results were obtained using a hardware prototype in which the functions of the CMOS gate array were temporarily implemented by ALS/Fast standard discrete logic components.

1. INTRODUCTION

The international standard 32 kb/s ADPCM algorithm has been approved by CCITT in Recommendation G.721 (1984). The algorithm describes transcoding A-/mu-law PCM to and from the proposed ADPCM. A general description for the algorithm is given in {1-2}. The performance of the coding scheme for speech, voiceband data and DTMF signals is presented in {3-4}. Moreover, a variation of the algorithm has been proposed by the T1Y1 committee as a North American standard. This one includes provisions for accommodating the T-carrier signal format.

Several features characterize the CCITT algorithm ; namely

- Adaptive backward predictor with 6 zeros and 2 stable poles using floating point arithmetics,
- Adaptive quantizer with two modes of adaptation ; fast for speech-like signals and slow for voiceband data and DTMF. Implementation is performed in the base-2-LOG domain.
- Synchronous code adjustment to avoid cumulative distortion in synchronous tandem coding.
- Algorithm implementation uses different word lengths to represent different signals.

Hardware implementations for the CCITT algorithm have been presented in {5-7}. In {5}, a single CMOS semi-custom chip is designed for a half-duplex transcoder (i.e. one chip per encoder or per decoder). That approach, although requiring a custom chip, features low power consumption and high circuit density. {6} offers a program for a single general- purpose Digital Signal Processor, TMS32010, which can work together with added off-the-shelf chips to realize a half-duplex transcoder. That approach offers a readily available solution at the expense of high power consumption and low circuit density. {7} gives a preliminary information on a CMOS VLSI chip implementing a full-duplex single channel transcoder.

The above approaches were optimized for single channel applications. Multichannel applications, however, cover most of the current market demand. For these applications, the approach presented in this paper offers several advantageous features ; namely relatively higher circuit density and reasonable power consumption without the need to develop fully custom chips. A 24-full-duplex-channel transcoder (24 encoders and 24 decoders) is discussed. It uses two identical special purpose processors implementable as CMOS gate arrays, together with CMOS EPROM lookup tables.

Section (2) of this paper describes the architecture used with its main features. Section (3) presents details on the hardware and firmware main blocks. Section (4) gives the test results obtained.

2. TRANSCODER ARCHITECTURE

In multichannel implementation, each operation is repeated for each channel. Therefore, a dedicated circuit can be efficiently built for each operation and

ICASSP 86, TOKYO CH2243-4/86/0000-0349 $1.00 © 1986 IEEE

still is fully utilized by time sharing it between the different channels. In a single channel application, however, the same circuit is used to perform a variety of operations. Therefore, efficiency is generally compromised.

The operations required to implement the CCITT ADPCM algorithm are arranged into two groups:
a-Digital filtering, limiting and general add/subtract.
b-Number representation conversion (fixed, floating points, base 2 logarithmic, A/mu law), floating point multiply and nonlinear functions.

A simple processor consisting of an ALU and a shifter can efficiently implement group "a" functions. Group "b" can best be implemented by special circuits. The approach followed is to design a dedicated processor for the first group of operations. The complexity of that processor is such that it can be integrated in a gate array. The second group of operations is implemented by lookup tables. The block diagram is shown in Figure (1).

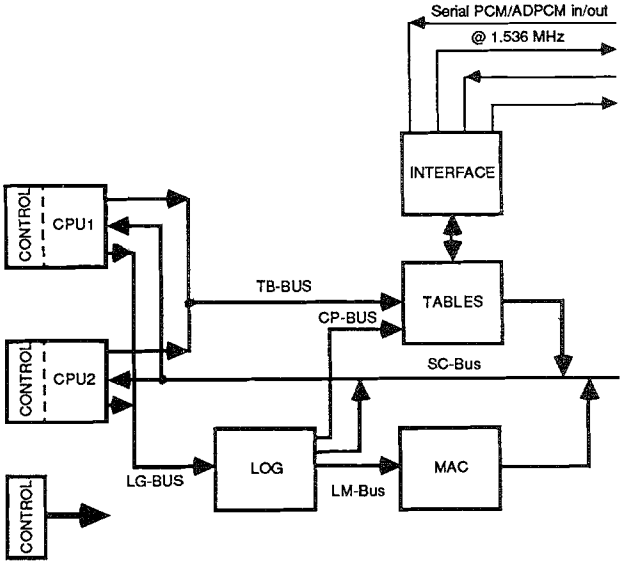

Figure 1 - 24 channel ADPCM transcoder block diagram

The CONTROL block, shown in Figure (1), generates the control time waveforms required for the whole system.

The LOG block performs the logarithm and similar operations. MAC is a floating point to fixed point multiplier- accumulator used to implement the predictor difference equation. TABLES does several functions such as mu/A-Law to linear conversion, antilog, quantization, synchronous code adjustment and nonlinear functions. Interface to the outside world is provided by four serial lines (PCM in/out, ADPCM in/out) via the INTERFACE block. Speed calculations for these blocks give a limit of 24 to the number of full-duplex channels handled. This is based on using 200 ns high density EPROMs.

The CPU performs group "a" operations. Its instruction set is optimized for those operations in the algorithm. With 3 micron CMOS technology, the presented design can handle 12 full duplex channels. As a result, two CPUs match the throughput requirements for the other blocks (called service circuits).

Three options were considered for the timing of this multiprocessor system:
1-To run the two CPUs in full synchronism
2-To run the two CPUs in synchronism but with a time offset
3-To run the two CPUs asynchronously.

Option 2 is found to be the most suitable because it avoids the need to use interrupts as in Option 3, and it eliminates the buffers required by the two CPUs to access the common circuit should Option 1 be used.

The transcoder architecture shown in Figure (1) has the following features:
- Inter-and intra-block parallel processing
- Inter-and intra-block pipeline architecture
- Whole system fully synchronous
- Multiprocessors time share common resources (circuits and buses)

3. HARDWARE/FIRMWARE BLOCK DIAGRAMS

3.1 THE CPU

The CPU block diagram is shown in Figure (2). It consists of the following:
- 16-bit ALU that performs A+B,A-B,B-A
- a shift-right barrel shifter
- 4:1 operand storage multiplexers
- register- and constant-file.
- serial 7x13 multiplier
- registers to store the sign bits of the difference signals for the current channel
- a PAL for creating the special instruction set used.
- RAM and pipeline address register

The CPU is designed as a linear machine with neither loop, nor branch nor interrupt instructions. It runs at 9.216 MHz except the register-file which runs at twice the speed. The design made use of the high internal speed of the CMOS process and accommodated for its slow I/O drivers. The control waveforms for the CPU are implemented on the chip with random logic , while those for the service

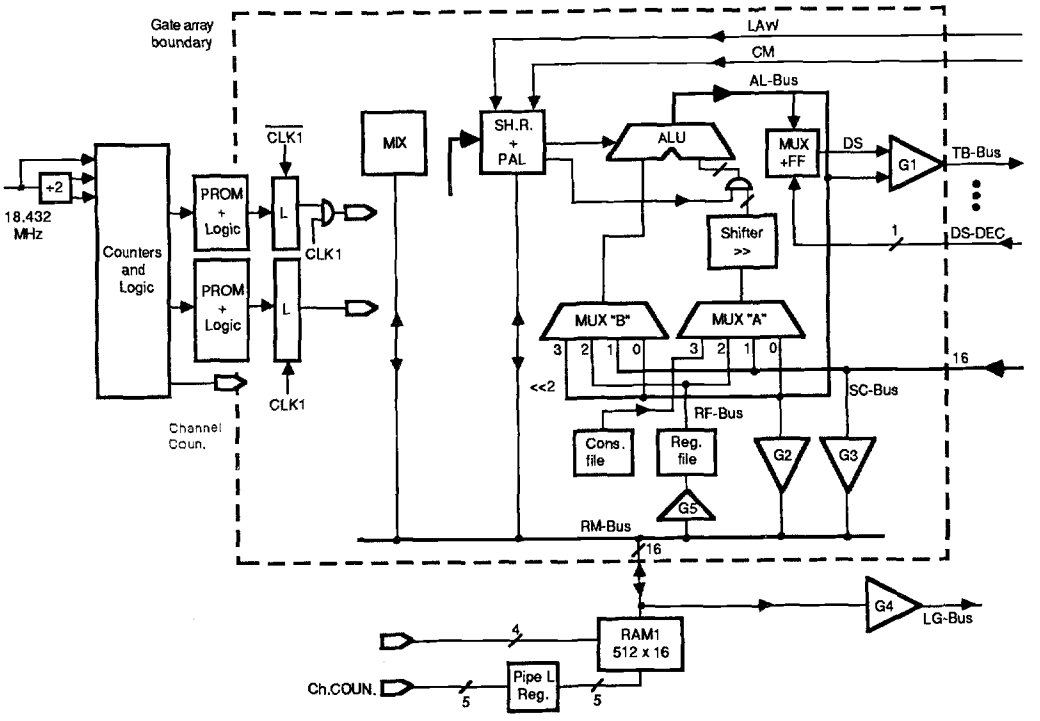

Figure 2 - The CPU block diagram

circuits are implemented with a mixture of PROMS and random logic.

3.2 THE LOG BLOCK

The LOG block converts 16-bit two's complement fixed point numbers to any of the following forms:

1- 11-bit sign magnitude floating point.
2- as in 1 with a division by 4 for the input signal.
3- 11-bit base-2-logarithm.
4- 8-bit mu-Law
5- 8-bit A-Law.

Direct lookup table implementation requires excessively large amounts of memory. Here, a novel patented approach is used. The memory is split in a pipeline architecture as shown in Figure (3). Memory requirements are thus reduced to 1/32 of those required by the direct implementation.

3.3 THE MAC BLOCK

This block implements the predictor difference equation. The predictor coefficients are updated by the CPU. Pipeline architecture is used between the multiply operation and the adjust and add operation as shown in Figure (4). To simplify the traffic on the common buses, a separate RAM is used to store old signal values. This RAM is addressed by an automatic address generator.

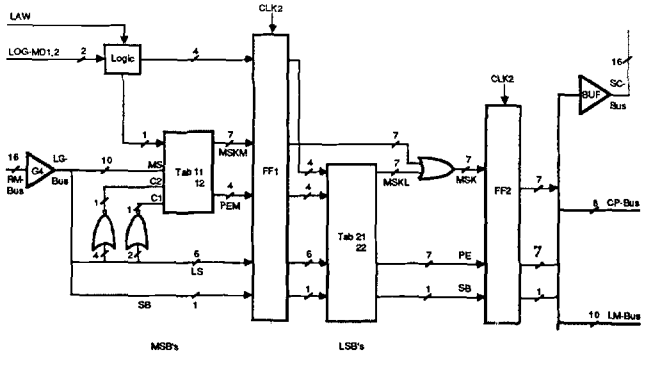

Figure 3 - The LOG block diagram

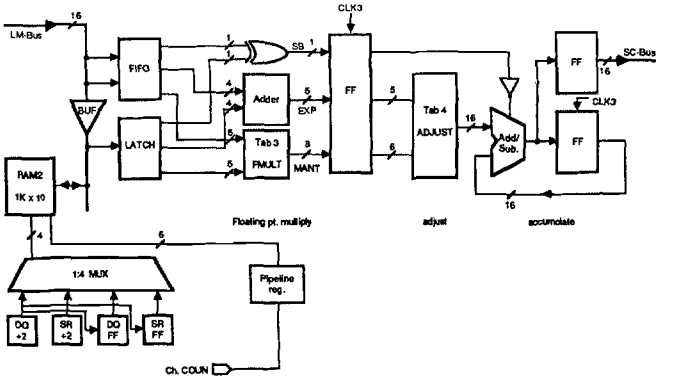

Figure 4 - The MAC block diagram

3.4 THE FIRMWARE

A 96-instruction program is developed to implement a CCITT-compatible encoder and decoder with options for the North American standards. The encoder and decoder parts are very similar except for the sequence of instructions and the synchronous code adjustment.

Due to the parallel processing used, a 93-bit instruction word is used. Each part of the circuit is controlled independently by a separate field in the instruction. The ALU field provides 14 functions ; some of them are : A+B,A-B,B-A,B±A,B+(A or 0), B-(A or 0), ±(A-B), and ABS(B).

The program development is relatively difficult because of the parallel synchronous and pipeline architecture, the resource sharing between the two CPUs, and the lack of development tools.

A part of the encoder program flow chart is described below:
After a serial PCM sample is received by the INTERFACE, it is converted to linear format (signal SL in the G.721 Recommendation) by the TABLES. Prior to that, the MAC calculates the signal SE. The CPU then gets the difference signal D by the operation SUBA. D is sent to the LOG. The coefficients B'i are updated by the CPU. When the LOG is finished, the CPU performs SUBB, and DLN is sent to the TABLES to be quantized.

In addition to the above, firmware includes the contents of all the lookup tables. These are developed on a mainframe computer.

4.SYSTEM DEVELOPMENT and TEST

System development has gone through the following phases:

1-Test patterns were generated to separately test the different blocks of the hardware.
2-After the integration of the firmware and hardware, the CCITT test tape was used for end-to-end testing. The tape contains 18K bytes of PCM and 34K nibbles of ADPCM to stimulate the transcoder. The corresponding correct output is also provided by the CCITT test tape. Each channel of the transcoder was stimulated by the input test sequence. The channel output was successfully compared to the correct one. During the transcoder development, a software simulation package implementing the CCITT algorithm was built. It was used for fault isolation purposes.
3-A signal to total distortion ratio test was performed purely in the digital domain. A 1004 Hz PCM tone was used. After looping back the ADPCM, the PCM output was captured and analyzed by a software program. Results are shown in Figure (5). They match those in {4}.
4-A qualitative analog speech test was performed successfully.

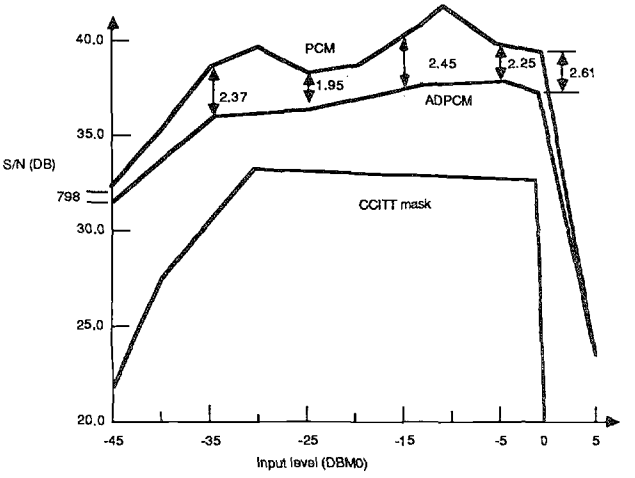

Figure 5 - Signal to total distortion ratio (1004 Hz)

REFRENCES

{1} W.R. Daumer, P.Mermelstein, X. Maitre and I Tokizawa, "Overview of the ADPCM Coding Algorithm", GlobeCom 84, Atlanta, U.S.A. pp774-777.
{2} M. Taka, R. Maruta and A. Le Guyader, "Synchronous Tandem Algorithm for 32 kbit/s ADPCM", GlobeCom 84, Atlanta, U.S.A. pp791-795.
{3} G. Williams and H. Suyderhoud, "Subjective Performance Evaluation of the 32-kbit/s ADPCM Algorithm", GlobeCom 84, Atlanta, U.S.A. pp778-785.
{4} J. Raulin, W. Belfield and T. Nishitani, "Objective Test Results for the 32kb/s ADPCM Coder", GlobeCom 84, Atlanta, U.S.A. pp786-790.
{5} M. Sato, Y. Ishikawa, T. Nishitani, T. Kato, H. Saita and Y. Aoki, "A Single Chip Signal Processor for CCITT Standard ADPCM codec", 1985 IEEE International Solid State Circuits Conference pp 192-193. Also see: T. Nishitani et al in, ICASSP85, pp 1425-1428.
{6} A. Charbonnier, X. Maitre and J. Petit, "A Digital Signal Processor Implementation of the CCITT 32 Kbit/s ADPCM Algorithm", IEEE Int. Conf. Comm., pp 1197-1201, 1985. Also see FYI, a T.I. update magazine, Vol.2, Issue 2, Feb. 85, P1.
{7} Motorola Semiconductors, Data sheet on the MC1454xx CMOS VLSI ADPCM TRANSCODER.