System Design Project: Multi-Channel ADPCM CODEC (MCAC)

Vineeta Singh, Student, *MSEE RIT*

***Abstract*— the multi-channel ADPCM with respect to CCITT recommendation G.726 provides a voice-compression algorithm defined by ITU. For the project, 8 channels are used. ADPCM plays an important role in encoding and decoding voice signals for various applications like digital cordless telephones, VOIP, wired and wireless telecommunications. For voice compression algorithms, low bit rates is an important concern. PCM at the sample rate of 8 KHz gives 64kbit/s. ADPCM helps for achieving lower bit rates: 40, 32, 24, 16 kbit/s. ADPCM varies the quantization bits adaptively based on the input sample and thus reduces number of bits required for representing data from 8 to 4. ADPCM encoder takes the input as PCM values and ADPCM decoder outputs PCM values. Test vectors generated by C model have been used for verification and they can distinguish between a-law and µ-law and also the different bit rates. Vector verification has been performed for all the modules to ensure that they are passing all test vectors. Synthesis reports help us to see the area, timing reports, total coverage of each block.**

Three architectures Bit-Serial, Pipelined and Single resource have been implemented. The main focus of this paper is on Bit-Serial architecture as the blocks FMULT, ACCUM and the control unit have been implemented like that.

***Index Terms*—MCAC, ADPCM, PCM, a-law, µ-law**

I. Introduction

The main goal of speech compression is to reduce number of bits required for representing speech signals so that stress on transmission bandwidth is reduced like when transmitting speech signal over mobile channels with limited capacity. In order for transmission, better quality, higher capacity, storage and digital processing, speech signals need to be converted from analog to digital signal which can be termed as speech signal digitization. The three phases for that are sampling, quantization and coding. Sampling enables converting the continuous analog signal into discrete digital signal. Quantization converts the amplitude of the signal from continuous to discrete. Coding then converts the discrete signal into series of binary bits. Quantization error is difference between actual signal amplitude and quantized amplitude. For reducing quantization error, the number of quantization steps can be increased but then the bandwidth also increases so there is always a tradeoff between the two[3]. PCM helps to convert the analog signals to digital using A/D converter and represents the codeword. PCM codec has got 8000 sampling rate where each sample is 8 bits and thus the transmission bit rate is 8000*8 = 64000 bit/s i.e 64kbit/s. 8bits are needed per sample for all international circuits and for that A-law and µ-law recommended by G.711 are used. µ-law is 14 bits, 84dB and used in North America whereas A-law is 13bits, 78 dB used by rest of the world. A-law and µ-law using logarithmic compression helps to convert PCM samples from 13/14 bit to 8bit codes [4]. The compression technique is called as companding which consists of compression and expansion. Every quantized value needs to be mapped into uniform PCM value and the mapping from A-law/µ-law to uniform PCM has been specified by the G.711 specs.

Adaptive differential PCM (ADPCM) uses the 8-bit uniform PCM value as the input and converts that to a 14-bit linear format from which the predicted value is subtracted generating a difference signal on which adaptive quantization is performed and a 4-bit value to be transmitted is formed. ADPCM converts 64kbit/s PCM channel to 40, 32, 24, 16kbit/s channels using trans-coding technique. The application for 16 and 24kbit/s channels is for overload channels carrying voice. 24kbit/s is used in digital response systems which are computer-controlled. 32kbit/s is can provide a telephone bandwidth of ‘transparent quality’ which is almost same as original source and used for tandem coding [3].40kbit/s is used to carry data modem signals in DCME (Digital circuit multiplication equipment) which is a voice compression equipment used at either ends of a long-distance link like for example submarines or communication satellite cables [6].

II. SYSTEM OVERVIEW

MCAC

The MCAC uses 8 channels for the implementation discussed throughout this paper. It consists of encoder and decoder blocks. The input to the encoder is basically a 8-bit PCM signal of 64kbit/s which gives the ADPCM output channels depending on the difference. For 40kbit/s, the difference is 5bits in which 4 bits are for magnitude and 1 bit for sign. For 32kbit/s, the difference is 4 bits out of which 3 bits are for magnitude and 1 bit for sign. For 24kbit/s, the difference is 3

Manuscript received May 16, 2014. This work was done as the class project for the course Advance Topics in Digital System Design.

Vineeta Singh is a student in the department of electrical and microelectronic engineering, Kate Gleason College of Engineering, Rochester Institute of Technology, NY 14623, USA phone: 585-351-8703; e-mail: vxs9946@rit.edu

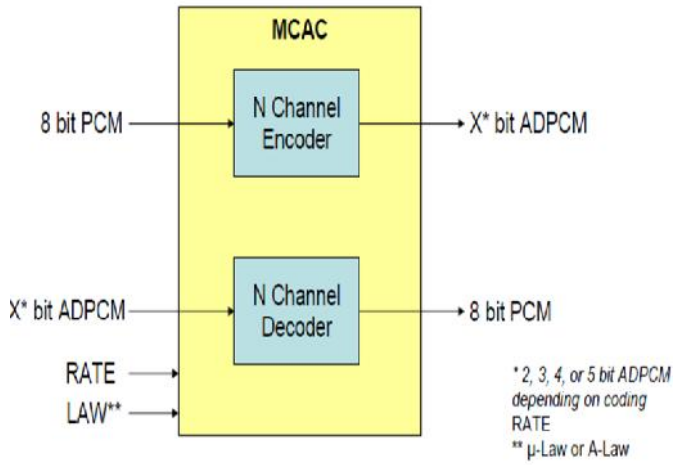

Fig. 1. MCAC Block diagram

bits for which 2 bits are for magnitude and 1 bit for sign. For 16kbit/s, the difference is 2 bits in which 1 bit is for magnitude and 1 bit for sign. The value of rate which is a 2-bit value will determine which channel will be selected. For the value of rate as 00: 40kbit/s is selected, for 01: 32kbit/s is selected, for 10: 24kbit/s is selected, for 11: 16kbit/s is selected. The value of LAW determines whether A-law or µ-law signal is selected. The module has been tested for both. For the value of LAW as 0:µ-law is selected and for value as 1: A-law is selected. The decoder again converts the ADPCM input to 8bit PCM signal.

A. Encoder

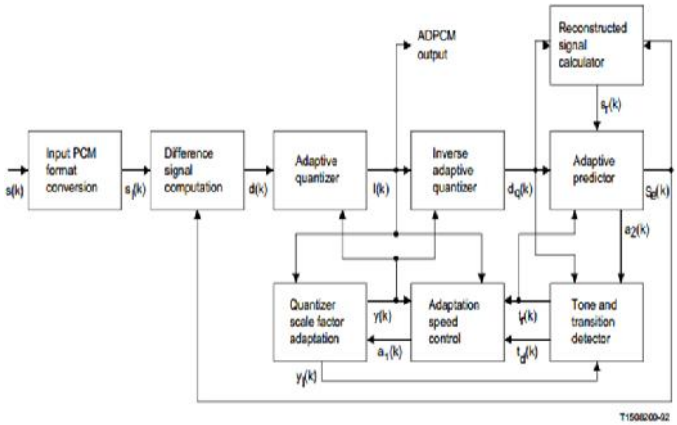

Fig. 2. Encoder Block diagram

The input PCM block coverts the A-law/ µ-law signal to a uniform PCM Sl (k) signal. The difference signal calculated by the Difference signal computation block is

d (k) = Sl(k) – Se (k)

Where Se (k) is the signal estimate generated by the adaptive predictor.

The adaptive quantizer quantizes the difference signal and generates the ADPCM output.

The Quantizer scale factor adaptation block converts the difference signal into logarithmic form and scaled by a suitable form and that happens before the actual quantization takes place. It basically produces the scaling factor for the quantizer and inverse quantizer.

The inverse adaptive quantizer again transforms the result from the logarithmic domain into linear form. It basically generates the dequantized value of difference signal.

Adaptation Speed control module value is derived from the rate at which the difference signal changes. For speech signals, it is 1 and for voice-band data signals, it is 0.

The Adaptive predictor and reconstructed signal module produces the estimate signal from quantized signal. It uses past 6 difference values which are dequantized and past 2 predicted values and calculates the weighted average. Original signal is reconstructed by adding the estimate signal to the difference signal. For reconstructing speech signal, basically the value generated by adaptive predictor is added to the dequantized signal.

Tone and Transition detector sets the prediction coefficient to 0 if a tone is present and calls the trigger_true routine. The ‘trans’ routine implements the transition detector and if the trigger value is 1 then the prediction coefficients are set to 0.

A. Decoder

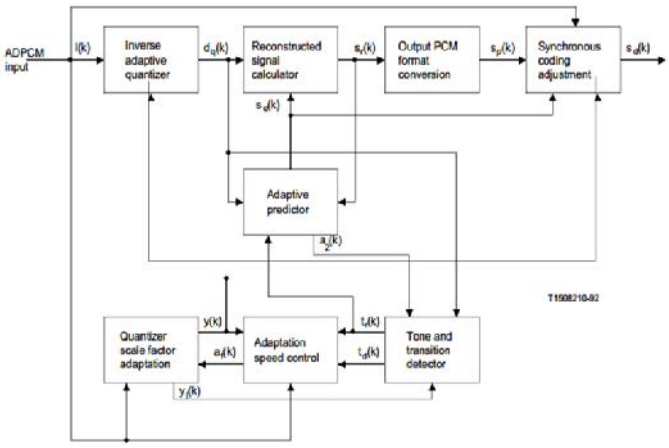

Fig. 3. Decoder Block diagram

The decoder contains majority of the same blocks as the encoder.

Reconstructed signal is computed as

Sr (k) = d (k) + Se (k)

The output PCM block converts the reconstructed signals back to A-law/µ-law PCM signal.

The synchronous coding adjustment block makes sure that there is no distortion of the speech signal.

The operation of both encoder and decoder is in synchronization so that automatically eliminates the need of

requiring any additional sideband data for synchronizing.

B. Bit-Serial Architecture

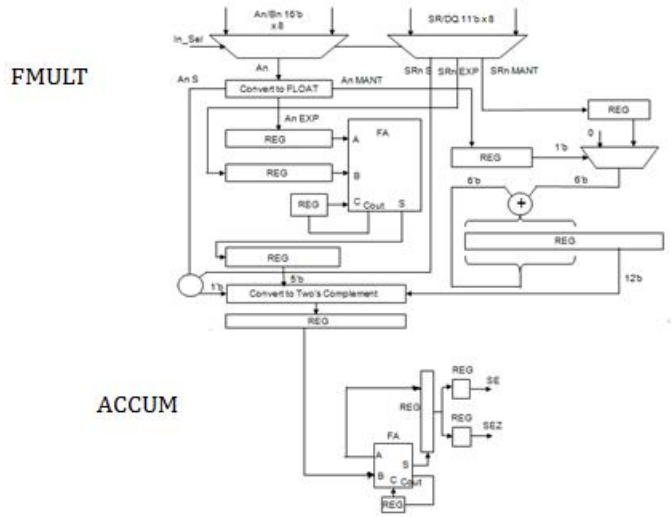

Fig. 4. FMULT_ACCUM Block diagram

FMULT-

For the FMULT block, two inputs are considered, out of which one is in floating point and other is in two's complement form. The two's complement value is converted to floating point. Then the exponent parts of both the inputs are stored in registers and then added in the Full adder Even the mantissa part of both the inputs are stored in separate registers and are multiplied in such a way that one is taken in bit-serial fashion and other in parallel. Then the mantissa, exponent and the sign bit are added and the result is converted back to 2's complement so that it can be used by the ACCUM block. FMULT requires 16 clock cycles for computing the result. As there are total 8 channels, 16*8 = 122 cycles are required by the FMULT.

ACCUM-

For the ACCUM block, 1-bit adder is used as the implementation is bit-serial. 16bit shift register is used. Initially it is reset, hence the value is 0. Then it takes the value from the FMULT and adds to the previous value and keeps updating the register accordingly. It takes overall 16 clock cycles to implement.

SE = SEZ + WA1 + WA2

SEZ = WB1 + WB2 + WB3 + WB4 + WB5 + WB6

FMULT_ACCUM takes overall 144 clock cycles for the complete operation

CONTROL UNIT-

The control Unit generates the necessary control signals for the smooth operation of the block. After the Start signal is activated, FMULT_ACCUM starts working. After FMULT_ACCUM has finished all the computations, it activates the done signal. When write signal is activated, results are written to the register. A total of 200 clock cycles are allocated out of which 180 clock cycles are actually used and remaining 20 are left idle.

III. TIMING ANALYSIS

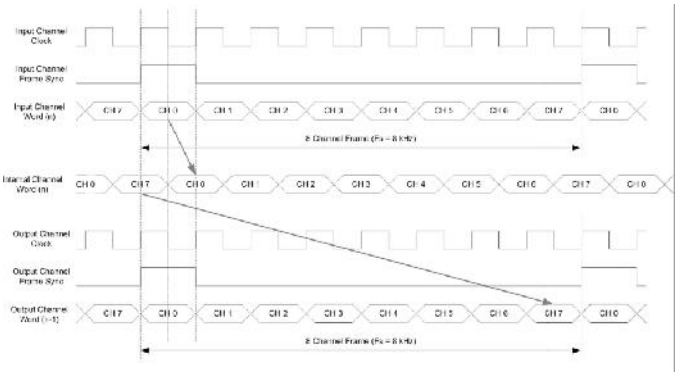

Fig. 5. Timing analysis of MCAC

Clock generator is mainly responsible for generating the necessary control signals, the frame sync, and the channel clock. The channel clock is 64 kHz and the time period is 15625µs. The frequency of the frame sync is 8 kHz. The frame sync synchronizes after every 8 channels and it is active only for the first channel activated.

The input word can be obtained during the negative edge of channel clock. A register file exists before and after the encoder. The encoder reads the value from the register file when the channel is active, then after 8 cycles, it is written to the register file. So when the next frame sync gets activated, data from the register file will go to the decoder. A total of 180 clock cycles is required for the whole system operation. For obtaining the SE of accumulator, we need 144 clock cycles and for SEZ 112 clock cycles.

IV. VERIFICATION METHODOLOGY

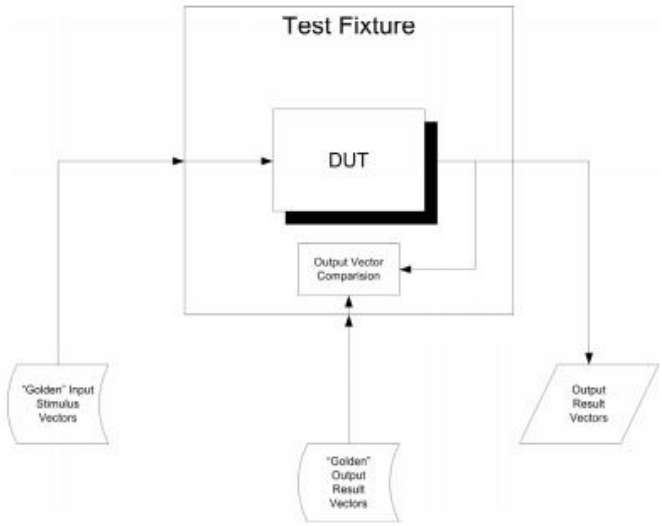

Fig. 6. MCAC module verification

For the verification purpose, official test vectors were already provided by ANSI or ITU for the ADPCM model. Therefore there is no need for using top level test suites for checking the operation of design.

The ADPCM codec model in C which was originally provided was only for 32kbit/s and thus was enhanced to check for 40, 32, 24, 16kbit/s. That was provided by ANSI standard Y1.301-1987.

ADPCM codec model provides Golden input stimulus vectors and Golden output stimulus vectors which are nothing but compliance test vectors for the RTL verification. The verification is completed only when the RTL and net-list pass all the compliance test vectors.

All the modules have been verified and the test vectors have passed which means that the output vectors provided by the codec model is equal after comparison to that of the output vectors passed by the modules. They are verified for all the 4 rates, for A-law and μ-law.

V. SYNTHESIS RESULTS

Encoder results

	Pre- scan	Post- scan
Total cell area	292969.35	327354.36
No. of cells	20	20
Gate count	29296	32735
Test coverage		96.39%
Timing analysis coverage		25270

Decoder results

	Pre- scan	Post- scan
Total cell area	307978.07	342506.11
No. of cells	15	15
Gate count	30797	34250
Test coverage		96.06%
Timing analysis coverage		25154

MCAC results

	Pre- scan	Post- scan
Total cell area	601300.02	669078.76
No. of cells	5	5
Gate count	60130	66907
Test coverage		97.7
Timing analysis coverage		50277

From the above results, it could be seen that the cell area increases after the post-scan insertion.

As the process has been successfully completed, net-list, timing file, and reports are generated.

The tools used for the project are NC Verilog which is a fast native simulator and supports Verilog 2001, Verilog 95, and SystemVerilog.

For the logic synthesis, syn.csh script was used and test insertion was performed with Synopsys Design Compiler.

The log file can be used to check for issues that need to be looked into.

Detailed timing analysis can be done using pt.csh using Synopsys PrimeTime.

SimVision is used for viewing the waveforms and all the design items.

The work area for the project was created in the Git repository.

For the implementation of the project, first according to the design specifications RTL code was written. Then they were tested using behavioral test-benches but for verifying it extensively, vector test-benches were used which used the test vectors generated by C model. If certain vectors were not passed, then immediate debugging was done. After that, the modules were synthesized.

VI. CONCLUSION

The design of architecture and other details were discussed by the individual groups and with the efforts and co-operation of all the team members, implementation of MCAC in bit-serial way was successfully done by simulating, synthesizing all the lower level modules and higher modules as well. Each of the modules has been successfully verified by using vector test-benches and all the vectors specified by ITU passed. The main goal of Bit-serial architecture was to optimize the area and cost. However, the area optimization could be achieved in a better way by using all the inputs in the FMULT serially or all in parallel and then use a parallel-to-serial converter. The total area of bit-serial MCAC is found to be more than the shared resource group.

We get the total test coverage of 97.7% for the MCAC. A total of 180 clock cycles are required for the overall implementation and the minimum operating frequency is 11.52MHz (180*64kbit/s) and the maximum frequency is 125MHz.

Some of the important problems were resolved during the testing of the modules. It is important to follow uniform naming conventions while doing a group project as it leads to errors if names are different for the same inputs. All the files of the lower-level modules need to be included in the higher level modules. The wire sizes needs to be declared correctly otherwise overflow problems may occur.

More channels can be used in future. The physical verification can be done.

ACKNOWLEDGMENT

The author would like to thank Mark Indovina for the feedback given for the different aspects of design project and comments pointed during the presentation of oral defense of project. The author would also like to thank David Herdzik for the help in debugging vector test-benches.

REFERENCES

[1] Bell System Technical Journal- ADPCM
[2] Digital Signal Processing Applications- Adaptive Differential Pulse Code Modulation
[3] Speech Compression
[4] ITU-REC-G.711-198811-E7107
[5] ITU-REC-G.726-199012-Spec-1990-E1951
[6] Digital circuit multiplication equipment Wikipedia
[7] MCAC timing diagram
[8] EE720 System Design Project information
[9] EDA Tutorial

Vineeta Singh received the BE degree in Electronics and Telecommunication from Pune University, India in 2012. She is currently enrolled as a graduate student in department of Electrical and Microelectronic engineering at Rochester Institute of Technology.

Her Research interests include Digital Design and Cryptography.