Design and Verification of Multi-Channel Adaptive Differential Pulse Code Modulation

Pratheep Joe Siluvai Iruthayaraj, *Student, Rochester Institute of Technology*

***Abstract*—Digital signal processing that can minimize the bit-rate without degradation in the signal quality is highly desirable. In telecommunication, speech compression is one of the main application where low bit rate is desired to reduce the bandwidth requirement. As per Recommendation G.711: Pulse Code Modulation (PCM) of voice frequencies by International Telecommunication Union (ITU), the standard PCM signal bit rate is 64kbits/s. In this paper, Recommendation G.726: 40, 32, 24, 16kbit/s Adaptive Differential Pulse Code Modulation (ADPCM) was used to achieve low bit rates for different application requirements. This paper details the design and verification methodology of Multi-Channel Adaptive Differential Pulse Code Modulation CODEC (MCAC). The design was targeted for processing 8 independent channels in a 125 microsecond frame. In order to achieve considerable reduction in system size and instantaneous power, the FMULT/ACCUM module in ADPCM algorithm was implemented as bit serial arithmetic operations with independent floating point multiplier and accumulator. A hierarchical bottom-up design and verification approach was followed resulting in gate level synthesis of MCAC, targeted to TSMC 0.18µm Process technology.**

***Index Terms*—Bit-serial, Digital Signal Processing, Floating Point Arithmetic, High level synthesis, System-level design.**

I. Introduction

PULSE code modulation (PCM) represents sampled analog signals digitally without taking advantage of redundancies in speech signals [3]. The PCM scheme referred in [2] uses either µ law (mu-law) PCM (used in North America and Japan) or A-law PCM (used in Europe and rest of the world) where the quantization levels vary as a function of amplitude [5]. Whereas, an Adaptive Differential Pulse Code Modulation (ADPCM) uses an adaptive predictor which predicts the future value based on the present signal value and quantizes the difference between them. In addition, it also adapts and varies the quantization step size based on the difference signal amplitude. Thus an ADPCM CODEC is highly desirable for message storage and retrieval which reduces the bandwidth requirement by a factor of two over the conventional 64kbit/s A-law or µ law PCM [4].

Manuscript received May 14, 2014. This work was supported in whole, or in part, by Department of Electrical Engineering, Rochester Institute of Technology.

The author is a graduate student from the department of Computer Engineering, Rochester Institute of Technology, Rochester, NY 14623 USA.

In 1990, International Telecommunication Union introduced the Recommendation G.726 [1], a voice-compression algorithm standard that supports transmission of voice signals at rates of 16, 24, 32 and 40kbit/s. Digital Signal Processor which can process multiple channels of voice data have been developed based on the ADPCM algorithm with different architectural approaches. In this paper, a bit-serial architecture of a multiplier and accumulator is utilized for the signal estimate calculations in the ADPCM codec algorithm for multi-channel applications. This paper is divided into three major sections. The first section gives a brief description about the ADPCM algorithm and its specifications. The second section details the architectural approach employed in the design with its unique characteristics and timing diagrams. The third section highlights the implementation methodologies to design, verify and synthesis the system that is developed.

II. System Overview

A. ADPCM Algorithm

The algorithm is based on the probability of high correlation seen in the consecutive voice samples. This characteristic is taken as an advantage to predict the future sample values based on the past sample value, then quantize the difference between the present and the predicted future sample [5]. In ADPCM this quantized difference sample is encoded, thus providing effective compression in the number of bits in each sample. Thus, the ADPCM encoded signal is of lower bit rate than PCM encoded signal.

B. Multi-Channel ADPCM

The Multi-Channel ADPCM CODEC (MCAC) is the top level block of the system which has a multi-channel encoder/decoder (CODEC) which follows the recommendations in [1]. The characteristics of the system include the conversion of a 64kbit/s A-law or µ-law pulse code modulation (PCM) data channel to a 16, 24, 32 or 40kbit/s data channel based on the requirement. Fig.1. shows the block diagram of the system in which the 8-Channel Encoder takes in 8 bit of PCM data (either A-law or µ-law PCM which is specified by the LAW input to the MCAC) in 8 channels and gives a 2, 3, 4, or 5 bit ADPCM transcoded data at the output. The number of bits in the ADPCM output is based on choice of the transcoding rate and the output channel capacity. The 8-Channel Decoder takes in 2, 3, 4, or 5 bit ADPCM data in 8 channels and gives 8 bit PCM data in each output channel.

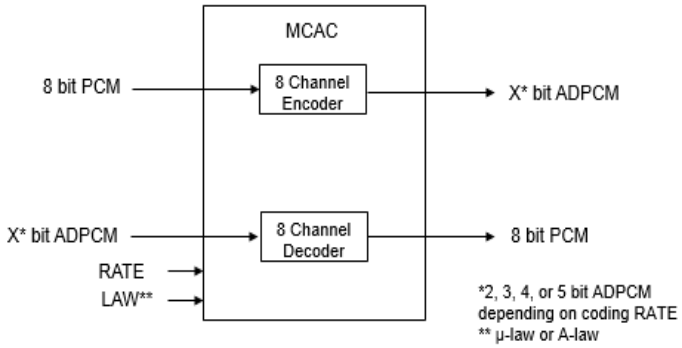

Fig.1. Multi-Channel ADPCM CODEC (MCAC) Block Diagram

C. *ADPCM Encoder*

In the encoder block diagram shown in Fig.2 the 8-bit PCM input s(k) is converted to 14-bit linear representation $s_l(k)$. The adaptive predictor produces the predicted value $s_e(k)$ which is computed by the FMULT and ACCUM functional blocks described in [1]. This particular FMULT and ACCUM functionality is done bit-serially which is explained in section 3 in detail.

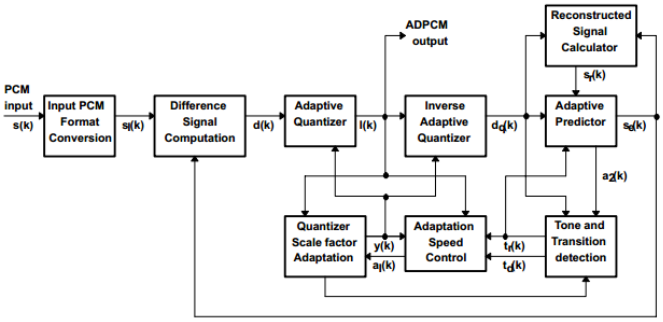

Fig.2. ADPCM Encoder Block Diagram [1]

The non-uniform adaptive quantizer, quantizes the difference signal d(k) to operate at 40, 32, 24, or 16kbit/s. Before quantization the difference signal is converted to base two logarithmic representation and scaled by the factor y(k) which is computed in quantize scale factor block. Then the values are normalized to give 2, 3, 4, or 5 bit quantized output. In the inverse adaptive quantizer, the scaled quantized values of the difference signal is normalized and transformed from the logarithmic domain to linear signal [1]. The tone and transition detector is used to improve the performance of the modem signals using two step detection process [1].

D. *ADPCM Decoder*

The decoder is very much identical to encoder in terms of structural representation but with additional uniform PCM to A-law or μ-law conversion and synchronous coding adjustment blocks. Fig.3

The additional synchronous coding adjustment block prevents the cumulative distortion that occurs on synchronous tandem coding (ADPCM-PCM-ADPCM, etc.). This distortion less communication is achieved by adjusting the output PCM codes such a way that it eliminates distortions in the next stage [1].

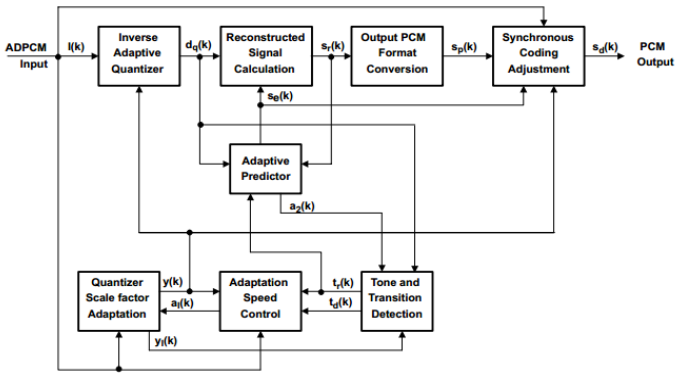

Fig.3. ADPCM Encoder Block Diagram [1]

III. Design Description

After examining the algorithm and the system specifications from [1], the design of MCAC was implemented as shown in Fig.4. In Fig.4, The encoder and decoder block is depicted along with its logic and required system blocks to control the flow of the encoder and decoder block in MCAC.

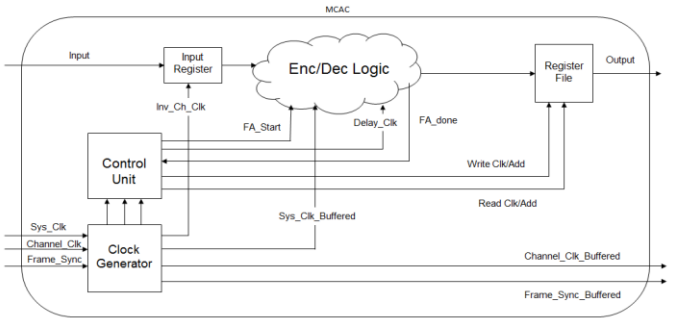

Fig.4. Encoder/Decoder Overview

The encoder and decoder blocks share four sequential logic blocks (Input Register, Control Unit, Clock Generator and Register File). The clock generator block takes in the system clock, channel clock and the frame sync signals that is generated off-chip. The system clock specifies the clock frequency of the MCAC system, channel clock specifies the frequency at which each input channel is captured inside the MCAC system, while frame sync specifies the frequency at which the frame of 8 channels are captured internally. These signals are buffered and wired to other sequential blocks in the system. Inv_Ch_Clk output is used to trigger the input register block. The buffered system clock is fed to the encoder/decoder logic as well as the control unit. The buffered channel clock and frame sync is fed as an output for the whole system which drives the output data. Control unit block simply implements a state machine which sends a FA_Start trigger to the FMULT/ACCUM block which triggers the operation and also receives FA_Done signal indicating the end of its operation. The FMULT/ACCUM block is explained in detail later in this section. The input register block is used to register the input

data that comes from off-chip at the falling edge of the Inv_Ch_Clk while at its negative edge, the registered data is accessed by the encoder/decoder logic. The output register is controlled by read/write clock signals along with read/write address for the register file. The write clock is used write the output data on the register file at its specified write address for each channel of data. The read clock is used to read the data from each channel and send as an output for the MCAC system. Fig.5 shows the internal timing of the MCAC system operation, where the input channel of data is captured at the negative edge of the channel clock and positive edge of frame sync indicates the channel0 of each frame. After 8 channel clocks at the raising edge of the frame sync, the output of each channel is written to the register file.

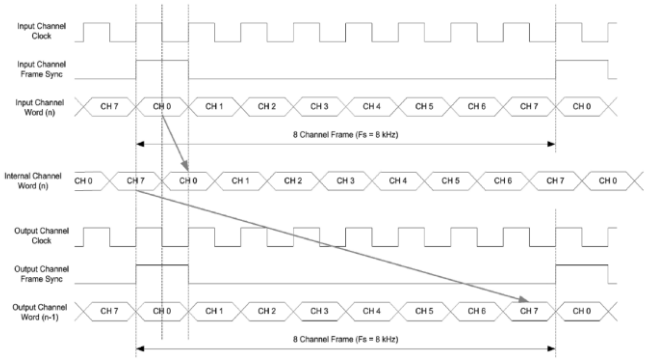

Fig.5. MCAC Internal Timing

A. *Fmult/Accum Architecture*

The design of Multi-Channel ADPCM involves the core design of the floating point multiplier and accumulator (FMULT/ACCUM). This paper discusses about the bit-serial implementation of the FMULT/ACCUM block of the system. The architecture of the multiplier and accumulator is designed based on the eight 16 bit inputs A_n/B_n and eight 11 bit floating point inputs S_R/D_Q. Initially, the 16 bit inputs are converted to floating point representation to perform the floating point multiplication over the two inputs. The mantissa and exponential of the inputs are obtained along with its signed bit. The mantissa part is multiplied using an array multiplier while the exponential part is added using a single bit full adder. In this bit-serial architecture, the mantissa value of one operand is bit serially shifted for each clock and fed as a control signal for the mantissa-multiplexer shown in Fig.6. The other operand is a parallel register which is added with the first 6 bits of the 12 bit shift register if the control is 1, else a zero is selected by the multiplexer and the 6bit addition is performed over the previous 6 bits of the 12 bit shift register. After 6 bit addition, the result is stored in the 12 bit shift register and it is shifted by one bit for every clock cycle. Meanwhile, the exponential part of the two inputs are added bit serially using a single full adder. Since the exponential part are 5bit wide, the result for 5bits bit-serial addition is obtained after 6 clocks including the carry propagate. In Fig.6, the output of the exponent and mantissa computations are stored in two shift registers. The signed bit from both the inputs are XOR-ed and stored in the output register for two's complement conversion. Once the outputs are obtained the content of the two registers are converted to two's complement and loaded parallel onto a 16 bit shift register. This shift register is shifted for every clock into a 1 bit full adder for accumulation. The architecture is designed such a way that the mantissa, exponent and accumulator calculations are done parallel. (1) Shows how the mantissa and exponents are calculated in the FMULT and converted to signed magnitude representation. Then the signal estimate is calculated using (2) which represents the ACCUM operation, where all the products are added and accumulated to get the signal estimate.

$$WA_nEXP = SR_nEXP + A_nEXP$$
$$WA_nMANT = ((SR_nMANT * A_nMANT) + 48) >> 4$$

$$WAnMAG = (WA_nMANT << 7) >> (26 - WA_nEXP)$$
When, $WA_nEXP \le 26$

$$WAnMAG = ((WAnMANT << 7) << (WAnEXP - 26)) \& 32767$$
When, $WAnEXP > 26$ (1)

$$SEZ = (((((((WB1 + WB2)\&65535) + WB3)\&65535) + WB4)\&65535) + WB5)\&655535) + WB6)\&65535)$$

$$SE = (((SEZ + WA2)\&65535) + WA1)\&65535 \quad (2)$$

In order to control this parallel operation, a finite state machine with 145 states is constructed.

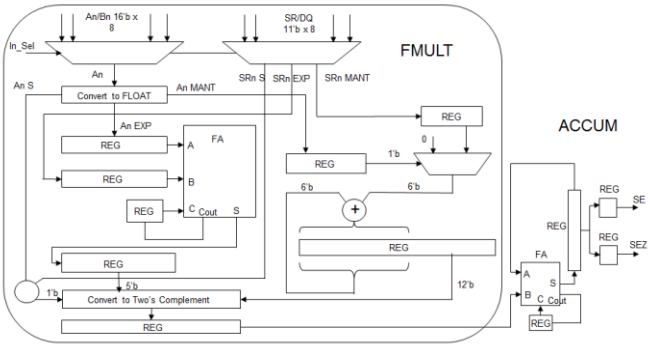

Fig.6. FMULT/ACCUM Architecture

B. *Timing Analysis*

The control unit generates FA_Start signal to the FMULT/ACCUM block after 16 clocks, till then the FMULT/ACCUM block remains idle. The 16 clocks is based on the time taken for the combinational logic that generates the inputs for the FMULT/ACCUM block. The block goes to compute state after receiving the clock signal. TABLE1 lists the number of clock cycles required in each state for the FMULT/ACCUM to compute the signal estimate for 1 channel of data.

TABLE I
CLOCK CYCLES FOR BIT-SERIAL FMULT/ACCUM

Steps	Clock Cycles	Operation
1	1	Input Select
2	2	Clear Output Registers
3	3-8	Exponent Addition
4	3-10	Mantissa Addition
5	10-14	Mantissa output shifting
6	15	Two's Complement Conversion
7	16	Parallel load to 16bit reg
8	16-112	Repeats the 1-7 steps for remaining 7 inputs
9	16-112	Accumulate and output SEZ
10	16-144	Accumulate and output SE

Total clocks for FMULT/ACCUM = 16x8 +16 = 144

The FMULT/ACCUM architecture takes totally 144 clock cycles to compute the signal estimate for the MCAC system. The computation of SE and SEZ values is done after 112 and 144 clock cycles respectively. After 144 clocks the state of the FMULT/ACCUM goes back to idle state and send a FA_done signal to the control unit.

Fig.8 shows the encoder and decoder timing specifying the clock cycles required for the combinational and sequential logic in the encoder and decoder blocks. The initial 16 clocks are required for the combinational logic to compute the inputs for the FMULT/ACCUM. The FMULT/ACCUM receives the start trigger and computes the signal estimates after 146 clocks (FA_Start + 144 + FA_done). Then the remaining combinational logic takes another 16 clocks to produce the outputs for the input channel data. The write clock is triggered after 178 clocks to write the outputs to the register file. The delay_strobe is the delay clock that is fed to fetch the next channel data.

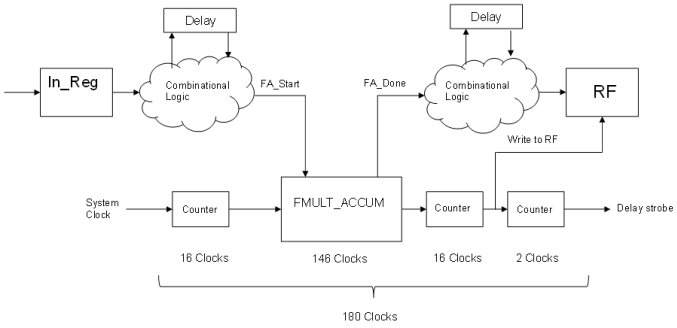

Fig.7. Encoder/Decoder Timing

Hence, from Fig.8 the encoder/decoder takes totally 180 clocks for converting the PCM to ADPCM in case of encoder and vice versa in case of decoder.

Total system clocks required = 16+146+16+2 = 180

As per the specification [1], the input channel clock is 64KHz and frame sync is 8KHz. The minimum system clock requirement is calculated using (1)

Minimum System clock frequency = 180x64 KHz = 11.5 MHz (1)

Therefore, the minimum system clock frequency required for the MCAC to operate is 11.5MHz.

IV. IMPLEMENTATION OVERVIEW

The whole system was initially modelled in C language to extract the test vectors for verification of the RTL design. The RTL design was done using the specification in [1] & [2]. The system was divided into several hierarchical modules and grouped to a single MCAC database. This database was source and version controlled using GitHub. Each module was coded in RTL and behavioral test benches where written for each lower level modules. Later the vector test benches were used to verify the functionality of each module.

A. Design Methodology

MCAC design methodology followed the simple ASIC design flow. The design follows a bottom-up approach where the low level blocks were first designed and moved up the hierarchy till the top level module. This system has four levels of hierarchy where simple combinational blocks formed the lowest level and the MCAC module forms the top level. Initially the system requirements was gathered form the specifications, then the functional blocks were coded using RTL and the functionalities were verified for each module. Moving up the hierarchy required low level modules were instantiated in the hierarchical modules based on the specification and algorithm. After verifying the functionality and logic for all modules, gate-level synthesis of the whole system from the top level is performed. The area and the timing analysis are captured for the system using the synthesis reports.

B. Verification Methodology

The C model generated vectors where used to verify the functionality of each module in system. ITU specified test vectors were used in the C model to generate the test vectors for each modules in the system. These test vectors were critical in verifying the RTL modules individually. The test vectors were used to verify the functionality of each module for different laws and rates.

C. Synthesis Methodology

After verifying the functionality of RTL the logic synthesis was performed using Synopsis to generate the gate level netlist. The netlist was generated for pre-scan and post-scan insertions. The RTL synthesis and Test was done using Synopsis tool. This synthesis tool was used to obtain the Automatic Test Pattern Generation (ATPG) coverage estimation for the system.

TABLE2 shows the area and test coverage results for encoder, decoder and the top level MCAC design.

TABLE 2
SYNTHESIS RESULTS

Block	Area (Pre Scan) µ meter square	Area (Post Scan) µ meter square	Test Coverage
Encoder	~292969	~327354	96.3%
Decoder	~327354	~342506	96.06%
MCAC	~601300	~669078	97.7%

D. Design Data

This section details the logical synthesis reports

- Total Cell Area for pre-scan netlist – 601300
- Total number of gates in the pre-scan netlist – 60130
- Total cell area for the post-scan netlist- 669078
- Total number of gates in the post-scan netlist- 66907
- Total timing analysis coverage for post scan netlist-
- Timing performance of pre-scan netlist- 7.623ns
- Timing performance of post-scan netlist- 7.863ns
- Minimum core clock frequency – 11.5MHz – 11.8MHz
- Maximum core clock frequency – 128MHz

V. PROJECT SUMMARY

The design of MCAC was done for eight channels with a bit-serial implementation. The minimum and maximum system clock frequency was found to be 11.5MHz and 128MHz with the total cell area of 669078 μm^2.

VI. CONCLUSION

An Application Specific Integrated Circuit was designed to perform eight channels of input data, compatible with the Recommendation G.726: 40, 32, 24, 16kbit/s Adaptive Differential Pulse Code Modulation (ADPCM). The design was done for the bit-serial architecture of the FMULT/ACCUM (floating point multiplier and accumulator). This architecture took 144 clock cycles for the signal estimate computation of single channel of data. The frequency of operation was found to be between 11.5MHz and 128MHz. The clock path with a minimum clock delay of 7.623ns was achieved at 128MHz. The bit-serial implementation reduced the instantaneous power since the computation for each clock is less. Hence, the bit-serial implementation of the Multi-Channel Adaptive Differential PCM targeted to TSMC .18µ Process technology consumed a total cell area of 669078 μm^2. The area is considerably small but it can be further reduced by implementing fixed point arithmetic instead of floating point arithmetic in the FMULT/ACCUM architecture. Additionally, the parallel to bit serial conversion of the input data contributed to the marginal increase in area, which can be avoided by performing a complete bit-serial manipulation over the multiplications performed for the computation of predictor coefficient in the ADPCM algorithm.

REFERENCES

[1] International Telecommunication Union, "*Recommendation G.726: Adaptive Differential Pulse Code Modulation (ADPCM) of voice frequencies by International Telecommunication Union*)," ITU-REC-G.726-199012-Spec-E1951.

[2] International Telecommunication Union, "*Recommendation G.711: Pulse Code Modulation (PCM) of voice frequencies by International Telecommunication Union*)," ITU-REC-G.711-198811-E7107.

[3] *Digital Signal Processing Applications - Adaptive Differential Pulse Code Modulation*.

[4] James D. Beatty, Richard D. Calder Jr., PerrySmith, Farazi, Daniel P.Kelly, and Dr. James L.Melsa"*Custom VLSI Design of A Single Chip Multi-Channel ADPCM Processor*"

[5] J. R. Booddie, J. D. Johnston, C. A. McGonegal, J.W. Upton, D. A. Berkley, R. E. Crochiere and J. L. Flanagan, "*Adaptive Differential Pulse-Code-Modulation Coding*" , The Bell Systems Technical Journal

Pratheep Joe Siluvai Iruthayaraj
This author was born in India on 11/16/1989. He holds a Bachelor's degree in Electronics and Communication Engineering from Anna University, Chennai, India in 2011.