FPGA implementation of Voice-over IP

Michel van den Braak and Stephan Wong
Computer Engineering Laboratory,
Electrical Engineering Department,
Delft University of Technology,
Delft, The Netherlands
{M.vandenBraak, J.S.S.M.Wong}@EWI.TUDelft.nl
http://ce.et.tudelft.nl/

Abstract— **Currently, the Internet provides a large number of services, e.g., e-commerce, file-sharing, and email, allowing people from all over the world to do business, exchange data, and communicate, respectively. A service that is gaining popularity is Voice-over-IP (VoIP) that allows people to communicate with each other via voice over the Internet without utilizing the plain old telephone service (POTS) – even though telephone lines may be used to carry the digital data. The perceived quality of this service greatly depends on the delay between capturing (subsequently, coding and transmission) of voice data and the playback (after reception and decoding) of the voice data. In this paper, we solely focus on the client-delay as we can not influence the delay caused by the Internet that carries the voice data. In this investigation, we are utilizing an FPGA platform that includes both a general-purpose processor (i.e., a PowerPC) with reconfigurable fabric surrounding it. First, we implemented VoIP in software. Second, we determined through profiling the most computationally intensive operation(s). The most time-consuming operations were found within the audio encoding and decoding, in particular, GSM en/decoding. In order to meet certain timing requirements, the coding accuracy was scaled back considerably resulting in a large degradation of the perceived audio quality. In order to improve the audio quality, we propose to implement these operations in the FPGA fabric surrounding the general-purpose processor.**

Keywords— **client, codec, delay, FPGA, GSM, reconfigurable hardware, VoIP**

I. Introduction

Nowadays, many services are running on top of the Internet allowing people to basically advertise, exchange, and retrieve information. An increasingly popular service is Voice-over-IP which allows people to talk to each other for free or at very low rates. A typical VoIP system is depicted in Figure 1 and a short explanation of the parts constituting this system is given in the following. First, at the sender side voice sound is sampled from a microphone. Second, this voice data is coded by an encoder to save bandwidth. Third, this coded bitstream is packed into IP packets and sent along an IP network, which in most cases is the Internet. Fourth, at the receiver side these packets are unpacked from IP and network headers. Fifth, the coded voice data from these packets and a decoder converts the voice data back into samples. Finally, these samples are put in a playback buffer. This buffer is needed to compensate for the variation of delay over the network, called jitter. Studies for buffer schemes can be found in [1], [2], and [3]. Basically, there are three locations in the overall system at which delay occurs: at the sender client, in the network, and at the receiver client.

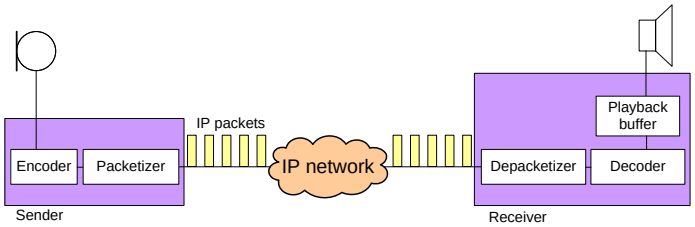

Fig. 1. Voice-over IP system

In this paper, we only focus on the client delay that is either the time between audio capture and actual transmission of the packets (containing the audio data) or the time between packet arrival and audio playback. One important delay factor (see Figure 1) is the audio codec being used in the encoder and the decoder. We have chosen to focus on the GSM codec that is utilized in cellular phones since it is similar to many other (linear predictive) codecs, e.g., G.723 and G.729. Another delay factor relates to the processing of the protocols used, e.g., SIP, SDP, RTP, UDP, etc. However, our initial results show that their impact on the client delay is minimal. Therefore, our main objective is to determine the most time consuming operations within the audio codec and subsequently implement these operation in reconfigurable hardware, i.e., field-programmable gate arrays (FPGAs). Consequently, we have implemented a software-alone VoIP solution on a FPGA platform that contains a general-purpose processor (GPP). More specifically, the software-alone solution is executed by this GPP. Subsequently, we profile this im-

plementation to determine the most time consuming operations. Afterwards, these operations are extracted from software and will be programmed in reconfigurable hardware. The reason for utilizing reconfigurable hardware is that it allows us to quickly change its functionality while at the same time the performance of the implemented operations are increased.

This paper is organized as follows. Section II gives an overview of related work and prior research with respect to delay in VoIP. Section III describes the used hardware, software and measure methodology. Section IV presents the results of our measurements. Section V will give conclusions about the results and recommendations for further research.

II. Related work

A lot of investigation has been done on measuring, characterizing, and reducing network delay and jitter, e.g., in [4]. The effects of bursty background network traffic on VoIP is investigated in [5]. However, less investigation has been performed on reducing client delay. A few performed research on real end-to-end delay. [6] and [7] used a metronome signal (pulses) to determine end-to-end delay by measuring the phase difference between the pulses. In addition to this research, [8] performed a large number of measurements on commercial VoIP phones (hardware and software) by calculating the delay of real sound. They used a local network to measure delays, which mean that network delay can be neglected because it is smaller than 1 ms. Client delay is measured and showed that hardware phones perform best with a minimum of 50 ms. Software phones have client delays of hundreds of milliseconds.

Our objective is to reduce client delay to a minimum. Other work has put the accent on minimizing delay in the playback buffer. [1], [2], and [3] work with adaptive playback buffers that adjusts the length of the playback buffer depending on certain QoS parameters, e.g., jitter, and packet loss. Our work extends this research, though, we are focusing on the processing time of encoding/decoding. Moreover, no research has, to our best knowledge, been performed on VoIP and reconfigurable hardware.

III. Experiment setup

In this section, we describe our implementation of VoIP. First, we describe the hardware. Second, we present the software implementation. Third, we explain our measurement methodology. Fourth, we briefly explain the GSM coding.

A. Hardware

The hardware we used, is the Xilinx XUP board. This board contains a Virtex II Pro and peripherals like Ethernet, Audio, VGA, and PS/2. The Virtex contains two PowerPC 405 embedded processor cores (of which one is used) which run at 100 MHz, 136 multipliers, 300 kB RAM cells, and an FPGA fabric. The FPGA fabric connects all blocks as depicted in Figure 2. These connections are made via two bus systems, PLB (Processor Local Bus) which is 64 bit wide, and OPB (On-Chip Peripheral Bus) which is 32 bit wide. These bus systems are IBM's CoreConnect compatible and connect internal blocks and peripherals on the XUP board to the processors. Figure 2 shows that all connections to the outside of the Virtex II Pro are made through FPGA blocks.

The Ethernet adapter and Audio AC97 codec are essential for VoIP. VGA and PS/2 are implemented to interact with the user. A PS/2 keyboard allows the user to select a contact and initiate a call, or terminate a call. A monitor connected to the VGA adapter, shows the status of the system and current delays. Due to the limited capacity of the VGA buffer, only text characters can be shown. The display is set to a resolution of 800x600 where 100x75 characters of 8x8 are shown.

When profiling is finished and time consuming operations are determined, they are implemented in hardware. The FPGA present on the Virtex is suitable to carry this implementation. The bus system allows us to communicate and exchange data from software to hardware parts.

B. Software

We built a VoIP implementation with minimum requirements. The software needed for a minimal VoIP application has the following protocols installed: SIP/SDP, and RTP/RTCP. SIP is a signaling protocol that is used to connect two or more people on the Internet. At startup, a REGISTER message is sent to the SIP server, in order to inform the SIP server of our location. At call setup, an INVITE message is send or answered, together with SDP (Session Description Protocol) messages. When both sides of the connection agree upon the session, a ACK signal is send to the side who first sent the INVITE signal. At call breakdown, a BYE message is send and answered by the other side of the connection. These four SIP commands (REGISTER, INVITE, ACK and BYE) are implemented in our software. SDP messages are embedded in SIP messages. They are used to negotiate which protocol and codec to use and exchange information on IP addresses and port numbers. When SIP has connected two endpoints, voice data is exchanged between them directly. This data is carried by

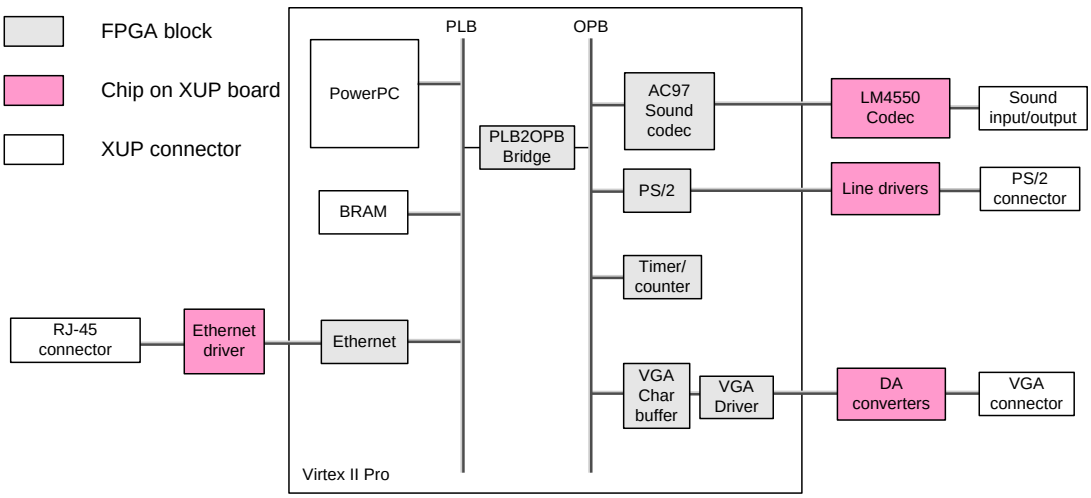

Fig. 2. Virtex II Pro Architecture

RTP (Real-time Transport Protocol). RTP also uses UDP to carry data over the Internet. In RTP, any codecs could be used to code voice data. Nearly all codecs have a 8 kHz 16 bit PCM signal as input. There are several popular sample based codecs, e.g., G.711 which compresses a 128 kbps bitstream into 64 bits/s, and G.726, which is a Adaptive Differential PCM codec and produces bitrates varying from 16-40 kbps. Other codecs are frame based codecs, e.g., G.723.1, G.729a, and GSM. We used an open source GSM 06.10 algorithm of C. Bormann and J. Degener from the Technical University of Berlin [9]. Since UDP is used, the sender will never know how many packets are received at the other side of the connection. Therefore, RTCP (Real-Time Control Protocol) informs a sender of RTP packets on the QoS of the network. An RTCP packet holds several kinds of QoS parameters, e.g., jitter and packet loss. RTP/RTCP always work on consecutive UDP port numbers: RTP on the even port number, RTCP on the odd ones. RTCP packets are not send as often as RTP packets, though RTCP can hold information about every RTP packet. Information on multiple RTP packets is collected and send in one RTCP packet. Sending no RTCP packet is also an option, but sender and receiver are not able to adapt to the network behavior.

C. *Timing measurement*

For time measurements, we use a hardware timer that is connected to the processor via the OPB bus. This is a hardware counter that runs at system clock (100 MHz). Because the counter is a hardware timer, it guarantees the accuracy of our measurements. Reading the counter over the OPB bus usually takes 10 cycles which does not significantly affect our accuracy. Accessing the counter needs the access time of the PLB bus plus the access time of the OPB bus, because of the PLB-to-OPB bridge. Each bus occupies 5 cycles to service a request. The value of the counter is read from software before and after an operation. The counter is 32 bits wide which enables us to measure a maximum time of 42.9 seconds, before the counter overflows.

We perform two types of measurement. The first type of measurement is determining the processor usage at three different states our system can be in: Idle, Ringing, and Talking. The second type of measurement is a more specific measurement of partial functions. The outcome of the first measurement is used to profile partial functions of time consuming operations. These operations are subject for implementation in hardware.

D. *GSM Coding*

GSM is a RPE-LPC (Regular Pulse Excitation - Linear Predictive Coder) coder [10] and contains a encoder and a decoder. We first describe the encoder and second, the decoder.

D.1 Encoder

The block diagram of the encoder is depicted in Figure 3. It consists of the following parts:

- Preprocess
- LPC Analysis
- Short term Analysis
- Long term Analysis
- RPE Encoding

The input for the encoder is a block of 160 samples. Only the 13 most significant bits of each 16 bit sample are used for encoding. The first stage in the encoding process is a offset cancellation and a first order pre-emphasis (low cut)

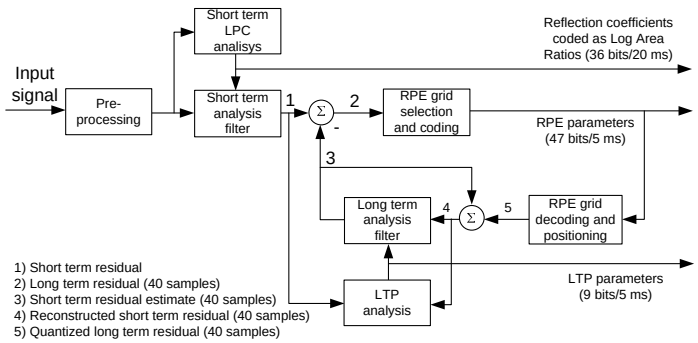

Fig. 3. GSM Encoder

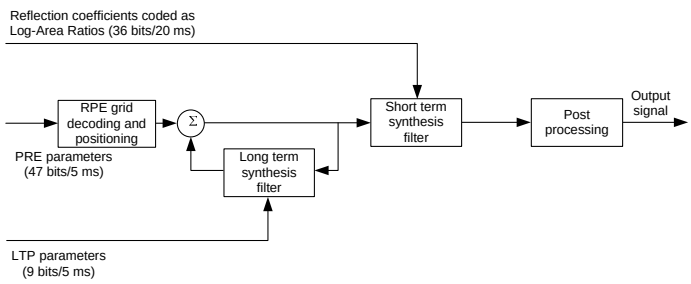

Fig. 4. GSM Decoder

filter. The second stage is calculating the Linear Prediction Coefficients (LPC), which are sometimes called reflection coefficients after the vocal tract model. The autocorrelation function is used to determine these parameters, that are used to filter the signal. Filtering is performed in the short term analysis. Because the filter gives a global characterization of the speech fragment, a residual signal remains. This signal is still 160 samples and is split in 4 equal segments of 40 samples. Four blocks of residual samples are fed to the Long term analysis and RPE Encoding to extract LTP (Long Term Prediction) and RPE (Regular Pulse Excitation) parameters. When these parameters are determined, the residual signal is simultaneously reconstructed for comparison with the original residual signal. Output of the encoder are 8 LPC coefficients, and for each 40 sample block a LTP Lag, LTP Gain, and 13 RPE pulses. Altogether, there are 76 parameters to send, which have been quantized to achieve compression.

D.2 Decoder

The block diagram of the decoder is depicted in Figure 4. It consists of the following parts:

- RPE Decoding
- Long term Synthesis
- Short term Synthesis
- Postprocess

The decoder contains the same feedback loop as the encoder. When the parameters are fed to the RPE decoder and long term synthesis, the reconstructed short term residual samples will be created. These samples are fed to the short term synthesis filter. Finally, the samples are postprocessed with a de-emphasis filter. The original signal is reconstructed.

IV. Results

We measured the processor utilization for the three states: Idle, Ringing, and Talking. As expected, the fist two states were not of interest. In the Idle state, the processor does nothing and in the Ringing state, the processor waits till the users has either accepted a incoming call, or a callee has accepted our outgoing call. The most interesting processor utilization was in the Talking state. We distinguish between the following tasks: Idle, SIP, Unpack, Unwrap RTP, GSM Decode, Playback, Record, GSM Encode, Wrap RTP, and Pack. The results of this measurement are depicted in Figure 5 and are acquired from a 32 seconds long call. We explain the tasks in the following. 'Idle' is the idle processor time. 'SIP' is the time needed to send, receive, compose or interpret SIP signals. While Talking, no SIP messages are exchanged. The next four tasks relate to receiving voice data. 'Unpack' is the time needed to strip the Ethernet and UDP/IP headers. 'Unwrap RTP' is the time needed to extract voice data from a RTP packet. 'GSM Decode' is the time needed to decode a GSM frame. A GSM frame holds 20 ms of voice data, which at 8 kHz samplerate corresponds to 160 samples. These samples are 13 bit wide and compressed to 33 bytes, containing 70 parameters. 'Playback' is the time needed for putting the voice data in the playback buffer. The next four tasks relate to transmitting voice data. 'Record' is the time needed to get the number of needed samples out of the record buffer. 'GSM Encode' is the time needed to compress a GSM frame. 'Wrap RTP' is the time needed to generate an RTP header along with the voice data. 'Pack' is the time needed to put the UDP/IP and Ethernet headers around the RTP packet. As can be observed in Figure 5, the GSM encoding and decoding together constitute more than 75% of the processor utilization.

The second measurement entails the determination of the different functions inside the encoding and decoding processes. For each function we measured the execution time in order to extract the most time consuming operations. The encoder functions are depicted in Figure 6 and the decoder functions are depicted in Figure 7. We observe that a 20 ms voice frame takes 9 ms to encode and 4 ms to decode. We must note that we tested two implementations. The first implementation used floating point operations and the second implementation used integer op-

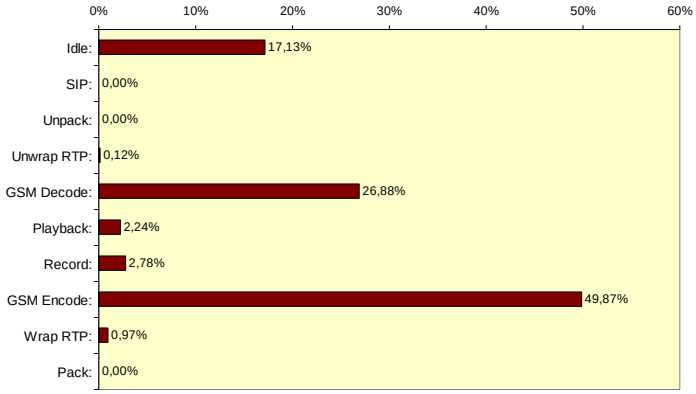

Fig. 5. Processor usage while Talking

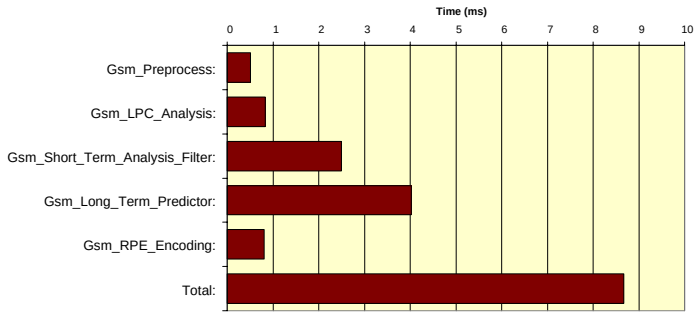

Fig. 6. Encoder time measurement

erations. However, when using floating point operations, voice encoding took 175 ms and decoding took 90 ms. It is obvious that the encoding/decoding should take less than the length of the speech frame, which is 20 ms. As a result, it is impossible to use the floating point version in software. A reason for this is the PowerPC 405 does not contain a dedicated floating point unit. A special library takes care for any floating point operation, which is time consuming. When comparing the voice quality by listening, difference can be heard between the encoder/decoder with integer and floating point operations. The integer version distorts the voice, due to rounding errors.

V. Conclusions and Further Research

Our objective was reducing client delay by using reconfigurable hardware. Our approach to this problem was first to build a pure software version of a VoIP implementation on the Xilinx XUP board. This software version runs on a PowerPC 405 embedded on a Virtex II Pro. We profiled this software version in order to determine the most time consuming operations. Our investigation yielded the most time consuming operations being GSM encoding and GSM decoding. Both operations combined utilized over 75% of the cpu time when in the Talking state.

We also observed that the processor was not powerful

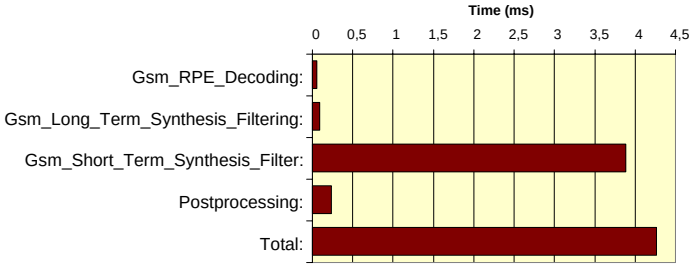

Fig. 7. Decoder time measurement

enough to process a floating point version of GSM encoding/decoding, because the PowerPC 405 does not contain a native floating point unit. Therefore, we used an integer version of GSM. However, this deteriorated the speech significantly. Consequently, these functions should be extracted from software and implemented into hardware, in order to speedup the encoding/decoding process. Hardware enables us at the same time to perform floating point operations. To maintain flexibility, an FPGA should be used to carry out the time consuming functions. As a result, we recommend for the future to implement the GSM encoding/decoding in the FPGA part of the Virtex II Pro.

References

[1] M. Narbutt and L. Murphy, "VoIP Playout Buffer Adjustment using Adaptive Estimation of Network Delays," in *18th Int. Teletraffic Congress ITC-18, Berlin, Germany*, pp. 1171–1180, Sept 2003.

[2] M. Narbutt and L. Murphy, "Improving Voice Over IP Subjective Call Quality," in *IEEE COMMUNICATIONS LETTERS*, pp. 308–310, May 2004.

[3] M. Narbutt and L. Murphy, "A New VoIP Adaptive Playout Algorithm," in *IEEE Trans. on Broadcasting, Vol. 50, No. 1*, pp. 1–10, Mar 2004.

[4] A. Charny and J.-Y. L. Boudec, "Delay Bounds in a Network with Aggregate Scheduling," in *Quality of Future Internet Services: First COST 263 International Workshop, QofIS 2000, Berlin, Germany*, Sep 2000.

[5] L. Zheng, L. Zhang, and D. Xu, "Characteristics of Network Delay and Delay Jitter and its Effect on Voice over IP (VoIP)," in *IEEE International Conference on Communications, Volume: 1*, pp. 11–14, Jun 2001.

[6] A. P. Calyam, "Performance Measurement and Analysis of H.323 Videoconference Traffic," Master's thesis, Department of Electrical Engineering, Ohio State University, Jun 2002.

[7] P. Schopis, "Summary Test H.323 Bounds Test Report," in *Technical report, Internet2 Technology Evaluation Center, Ohio*, Aug 2001.

[8] W. Jiang, K. Koguchi, and H. Schulzrinne, "Qos evaluation of voip end-points," in *Communications, 2003. ICC '03. IEEE International Conference on*, pp. 1917–1921, May 2003.

[9] C. Bormann and J. Degener, "ftp://ftp.cs.tu-berlin.de/pub/local/kbs/tubmik/gsm/gsm-1.0.10.tar.gz."

[10] *GSM: Digital Cellular Telecommications System(Phase 2); Full Rate Speech; part 2: Transcoding (GSM 06.10 version 4.2.0)*, Aug 2000.