VHDL IMPLEMENTATION OF TEXT TO SPEECH CONVERTER FOR SATELLITE RADIO RECEIVERS

R.Gunasundari, G.Sivaradje and V.Prithiviraj
Department of Electronics and Communication Engineering
Pondicherry Engineering College
Pondicherry.
Email: shivaradje@ieee.org

ABSTRACT

Text to speech (TTS) is synthetic, generated speech. Typed text is converted into speech using various approaches. This project work is about building a Text to Speech converter using a ***VLSI (Very Large Scale Integration) approach***. TTS conversion has numerous applications from enabling Robots to speak, to reading out to the blind. Specialized applications require innovative approaches for efficient user friendly, speech synthesis, rather than using the already developed speech engines. The motivation for this work is drawn from designing of a specialized TTS system to convert E-mail messages, received from a WorldSpace Satellite Radio Receiver to speech which is being undertaken as a value added service facility to the existing Digital Satellite Receivers. The work is implemented in VHDL and downloaded into the XILINX FPGA SPARTAN chip. The entire system is tested for numerous test input. It is observed that the overall feasibility of the TTS conversion has been well implemented incorporating very fast conversion rates.

1. INTRODUCTION

Text to speech (TTS) is synthetic or generated speech. Typed text is converted into speech using various algorithms. Human speech is difficult to produce artificially. At the simplest level, the entire vocal tract must be mimicked for a speech synthesizer to have the clarity and naturalness of real human speech.

During early attempts at synthetic speech, memory was extremely expensive. This affected the way early synthesis engines were designed. A memory efficient synthesis by rule system was popular, more commonly known as a Formant Engine. A Formant Synthesizer creates totally digitized or synthetic speech. No human recordings are used. They are similar to frequency synthesizers, whose output is similar to human voice. An advantage of the formant synthesizer is that the pitch and duration of words may be varied. However, in general the sound quality is inferior. Although this technique has been used with some success to speak numbers, many perceive this type of text-to-speech engine as sounding robotic[1-5].

A popular technique today is to store actual speech segments. This is known as Concatenative Synthesis. Phonemes are the smallest units of speech that distinguish one utterance from another. In this method of synthetic speech generation, the input to the speech engine is a phonetic spelling, obtained from the input text, and the output being the spoken version of the text. A two-phase process is usually employed (1) the text is converted into a phonetic representation with markers for stress and other pronunciation guides; and (2) the phonetic representation is spoken. The computation can be done on a DSP (Digital Signal Processor) or Microprocessor or both.

The objective of this paper is to design a specialized text-to-speech system, which would convert text input into speech. The text input is in the form of short messages as in any E-mail or *short messaging services (SMS)* [6]. The TTS system would automate E-Mail reading thus enabling the user to listen to his mails. Most E-Mail's do not extensively use all the vocabulary in English and hence the design of the TTS system is limited to a few optimum words.

2.IMPLEMENTATION OF TTS USING VHDL

Text to Speech conversion can be brought about, adopting a variety of approaches. In the software approach, a wide range of speech tools and speech engines exist, using which, the application can be given speech ability. Hardware approach is usually application specific. Hardware and software implementation of this project work is undertaken.

Software Implementation: The software implementation is done using Visual Basic 6.0(V.B 6.0), which is a Front End tool. It is an Integrated Development tool, which integrates all aspects of programming dictated by the application. The Microsoft voice text, Microsoft MAPI controls 6.0 are the important component controls being utilized for text to speech conversion. The body of the message is extracted from the input text file and stored in a rich text box. The text is then given to Voice Text control and the text input is read. The text input is now converted to speech.

Hardware Implementation: On the hardware platform text to speech synthesis is done using VHDL and after synthesis it is downloaded onto a FPGA chip. VHDL stands for ***VHSIC Hardware Description Language***. It is primarily intended for simulation, modeling of electronic systems. The behavior or functionality of any digital system can be coded, simulated, synthesized, routed and downloaded into an FPGA chip and its functionality verified. A FPGA chip consists of an array of logically uncommitted elements that can be interconnected to perform a specific application.

The system consists of an ***Analog Front End (AFE)*** and ***Xilinx FPGA chip*** both connected in parallel and a keyboard through which the input text is given. The design of the system consists of two phases.

- The first phase, involves the creation of a **S-RAM LUT** (Look Up Table), which contains the pre-recorded digital speech samples. Using the AFE words are first spoken and recorded via the microphone. The speech samples from the AFE are first fed to the Analog to Digital Converter (ADC), which digitizes them into bit streams of 14 bits in length. These digital samples are compressed from 14 bits to 8 bits using **m-law Companding** by the Encoder block. The noise components, which are inherent in the samples are removed and a Look Up Table is formed with the word as index and stored in the S-RAM. The block diagram of this process is illustrated in Fig.1 below.

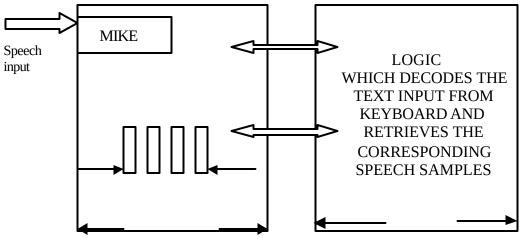


Fig.1 Block diagram of Design phase

- In the second phase the input text is given through a keyboard. Data from the keyboard is in ASCII (American Standard Character Information Interchange) form and is sent serially, along with necessary markers. These ASCII codes are sent to the Keyboard_Macro block where the 8-bit ASCII equivalent of the word is extracted and passed through a buffer sent to the Word_Framer block. The output of this block is the memory address of the speech samples. With this memory address, the corresponding chip of the S-RAM is selected, speech samples retrieved and sent to the AFE, where it is converted to voice and heard from the speaker attached to it. The block diagram for this process is illustrated in Fig.2.

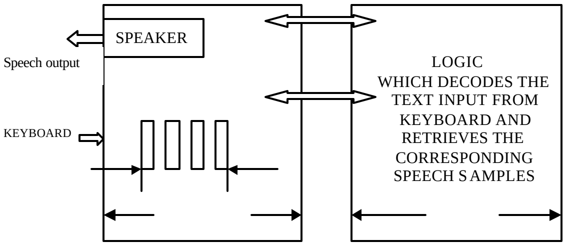


Fig.2 Block diagram of Runtime phase

The Central Block Diagram of the Text-To-Speech converter is illustrated in Fig.3. The blue lines represent the signal lines running from one block to another. Lines with thicker line widths represent Data Buses.

3. RESULTS AND DISCUSSION

The Text-To-Speech synthesizer is constructed using VHDL and downloaded into the XILINX FPGA chip. The following are the blocks that are constructed – Encoder, Decoder, Keyboard_Macro, Word_Framer, Speech_Store, Speech_Restore, Mem_R_W and Chip_Select. The blocks are tested, and their output waveforms verified. The following are the criterion based on which the testing is done.

- First it is checked whether the output is the desired output.
- Whether it is obtained at the correct Clock Pulse.
- Whether all the conditions are satisfied when the output is obtained.
- The state of all the signals during the time when the Reset signal is active is verified.

Then integration is done, starting with the interfacing of Encoder and Decoder blocks and verifying the correctness of the encoding and decoding process implemented. A VHDL model of S_RAM is created and then integrated with the Encoder and Speech_Store blocks. The testing is done thus verifying Phase One. Then Keyboard_Macro, Word_Framer, Speech_Restore, S_RAM and Encoder blocks are integrated and tested. Thus Phase Two is also verified. Finally all the blocks are integrated and the Top_Macro formed and verified.

A snapshot of the waveforms of each of the blocks is presented in the following paragraphs, accompanied by the explanations for important signals. The waveforms are obtained after testing and simulation using *Modelsim.*

Encoder

The 14-bit input signal In_Data in 2’s complement is compressed as a 8-bit code. µ-Law compression implemented as a Ram Table is used. The examples for input signal In_Data are taken as “00000000000000”, “11111111111110”, and “11111100111110”. They are compressed as “00000000”,”00000001” and “01000000” and sent on the output line Din. All operations are done and data obtained during the leading edge of the clock pulse.

Decoder

The 8-bit input to the decoder, Data, is decoded into 14 bit 2’s complement form. Data is the compressed speech samples obtained from the S_RAM. The examples taken are “00001100”, “00011100”, “0001101”. The decoded output are “00000010000000”, “11011111000111”, “1111111001101”. The output is in 2’s complement form. All operations are done and data obtained during the leading edge of the clock pulse.

Keyboard Macro

The serial input data, Data_Ser from pin 3 of the keyboard connector is trapped at every falling edge of the keyboard clock, Clk_Keyboard. The 8 required data bits is extracted and presented as a 8-bit parallel output, Data. The serial data given as an example is ‘0’,’0’,’0’,’0’,’1’,’1’,’1’,’0’,’0’,’0’,’0’. The 8-bit parallel output data is “00011100”.

Word-Framer

Data1 is the 8-bit input samples from the Keyboard_Buffer block, based on which the starting address of the speech samples of the input typed text is sent. Here as an example, input samples starting with “00011011” are given and the corresponding address, “000011000111101110101” is obtained on the output signal line Temp_Adr. All operations are done and data obtained during the leading edge of the clock pulse.

Speech Store

At the rising edge of every clock pulse, Clk, once Write signal from the Word_Framer block is high and Start_Stop signal from the Abs_value block is also high the samples from input signal line Datain, are sent to the Mem_R_W block through the output signal line Dataout. The input value “00010101” is given as an example and obtained on the signal line Dataout once the above mentioned conditioned is satisfied.

Speech Restore

This block routes the starting address of the samples from the Word_Framer block to the Mem_R_W and the samples read from the S_RAM to the Decoder block. As an example the input Din is given a value “10101010” and the Addr is given a value “ “ and the corresponding values are routed to the output signals Dac_Dout and Str_Adr.

Figure 4, 5 and 6 shows the waveforms obtained, for the explained input are enclosed in the following pages.

4. CONCLUSION

The Text to Speech converter is implemented using VLASI approach and the conversion carried out for various text input. The software implementation of the same is carried out in Visual Basic. In the hardware, implementation the designed system is successfully synthesized and downloaded into the XILINX chip and it’s working tested. Very fast conversion rates are inferred. The overall feasibility of the text to speech synthesis using a VLSI approach is established.

REFERENCES

[1] Douglas Perry, “Introduction to VHDL”, Mc Graw Hill, 1999, 3rd edition.

[2] J.Bhasker, "A VJDL Primer", Pearson Education Asia,1999, 3rd edition.

[3] S.D.Brown, R.J.Francis, "FPGAs", Boston, Kulwer Academic Publishers,1992.

[4] S.M.Trimberger, "FPGA Technology", Kulwer Academic Publishers,1994.

[5] A.Kaviani and S.D.Brown, "Hybrid FPGA architecture", Feb 1996.

[6] http://www.iec.org

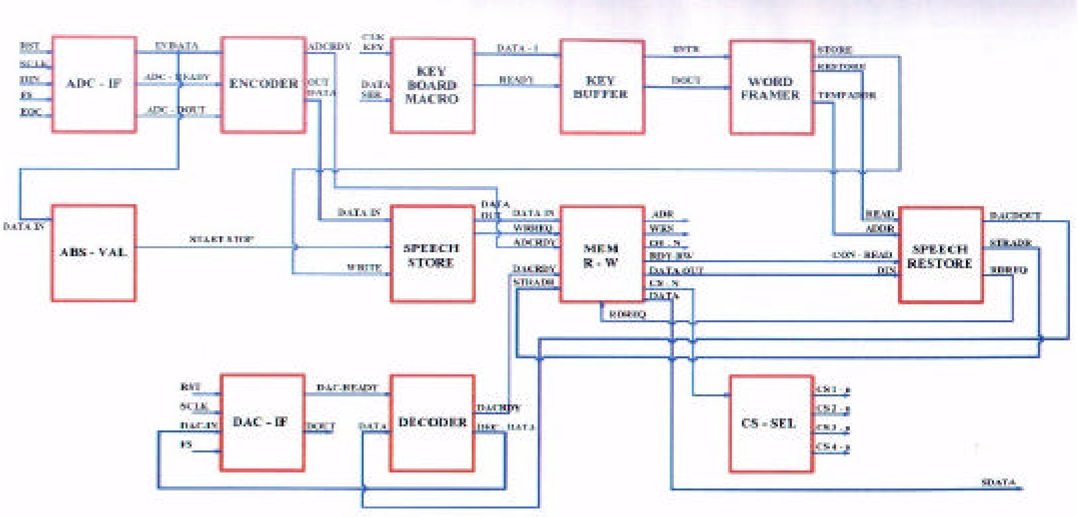
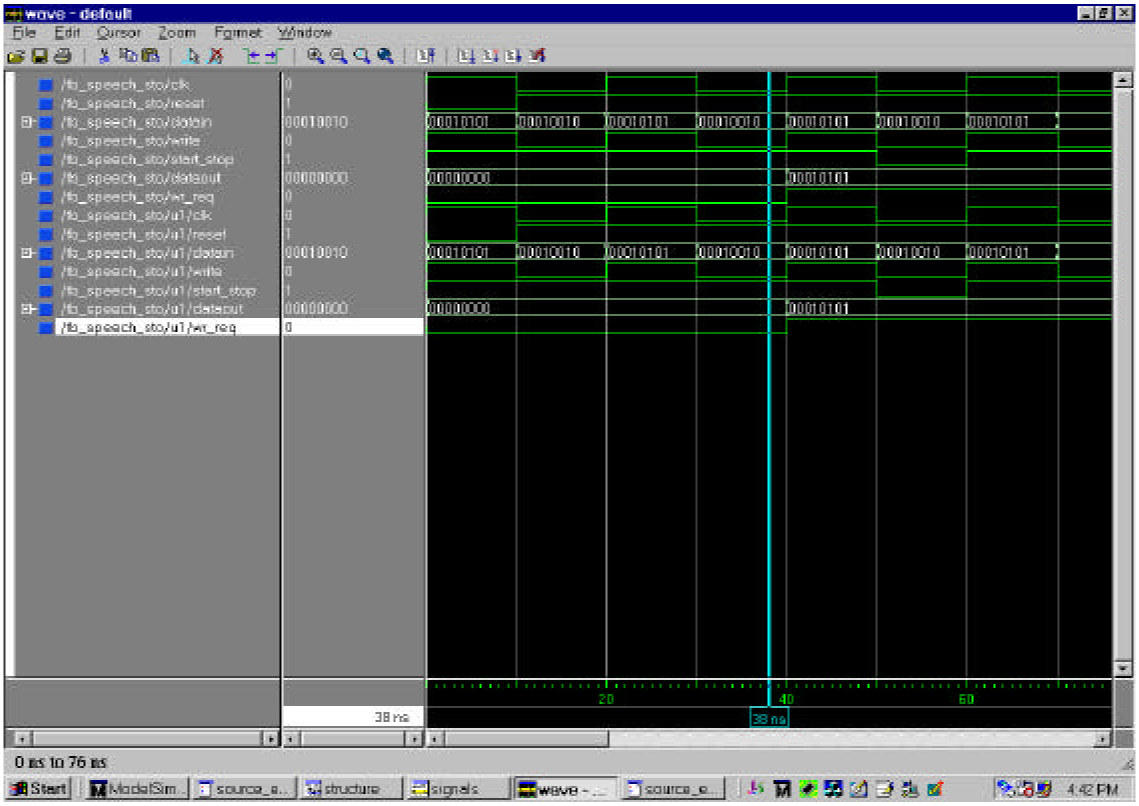


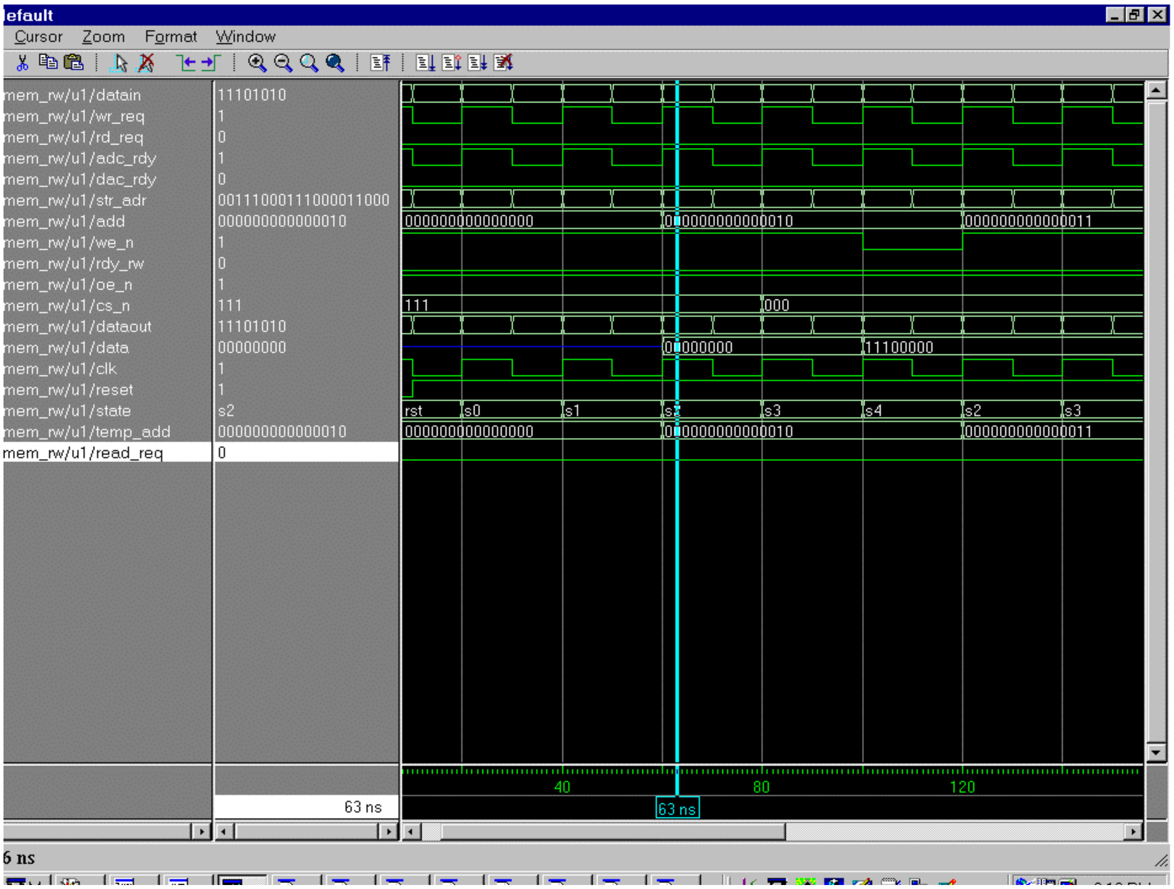
Fig. 3 Text to Speech Converter – Central Block Diagram



```
/tb_speech_sto/clk               0
/tb_speech_sto/reset             1
/tb_speech_sto/datain            00010010
/tb_speech_sto/write             0
/tb_speech_sto/start_stop        1
/tb_speech_sto/dataout           00000000
/tb_speech_sto/wr_req            0
/tb_speech_sto/u1/clk            0
/tb_speech_sto/u1/reset          1
/tb_speech_sto/u1/datain         00010010
/tb_speech_sto/u1/write          0
/tb_speech_sto/u1/start_stop     1
/tb_speech_sto/u1/dataout        00000000
/tb_speech_sto/u1/wr_req         0
```

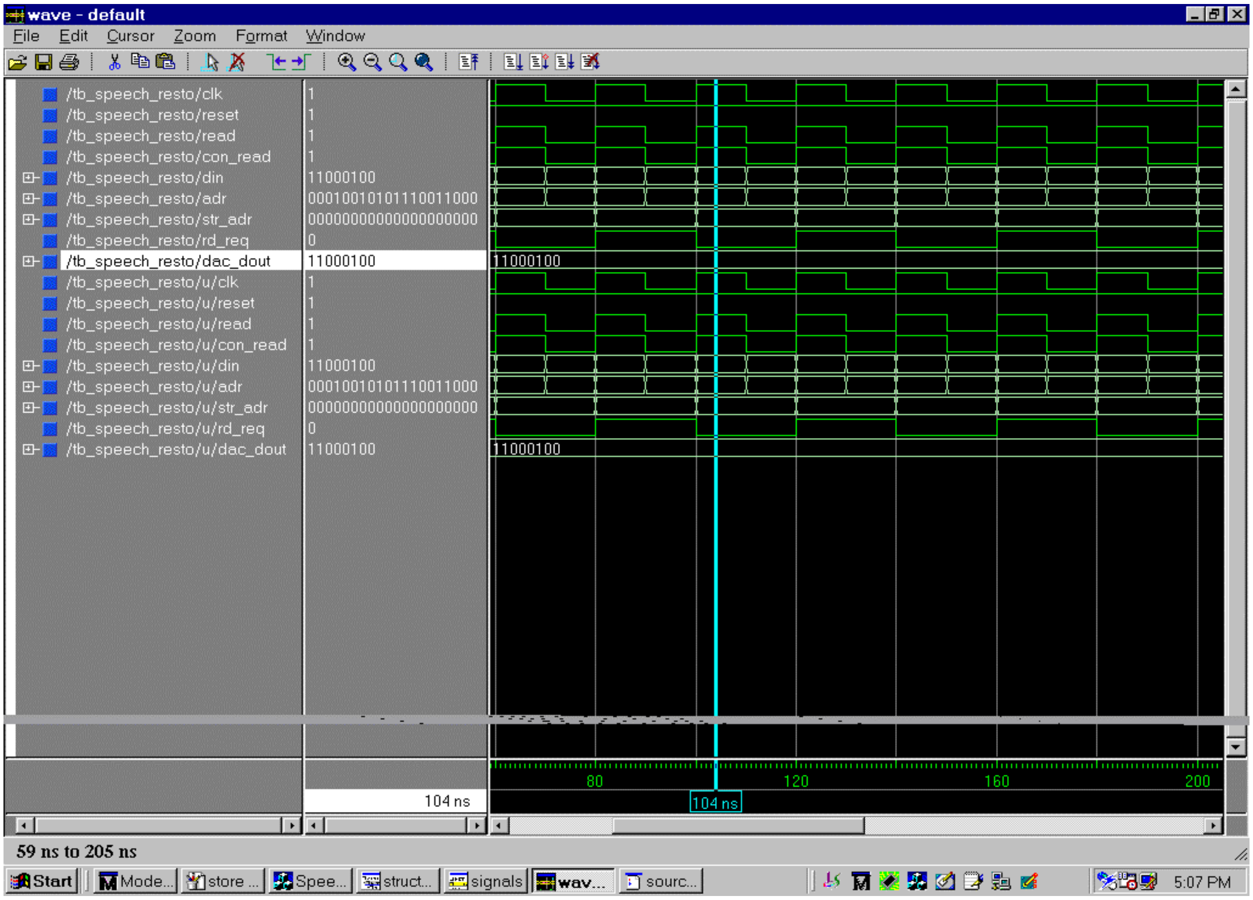
0 ns to 76 ns

Figure 4



mem_rw/u1/datain	11101010
mem_rw/u1/wr_req	1
mem_rw/u1/rd_req	0
mem_rw/u1/adc_rdy	1
mem_rw/u1/dac_rdy	0
mem_rw/u1/str_adr	0011100011100011000
mem_rw/u1/add	0000000000000010
mem_rw/u1/we_n	1
mem_rw/u1/rdy_rw	0
mem_rw/u1/oe_n	1
mem_rw/u1/cs_n	111
mem_rw/u1/dataout	11101010
mem_rw/u1/data	00000000
mem_rw/u1/clk	1
mem_rw/u1/reset	1
mem_rw/u1/state	s2
mem_rw/u1/temp_add	0000000000000010
mem_rw/u1/read_req	0

Figure 5



/tb_speech_resto/clk	1
/tb_speech_resto/reset	1
/tb_speech_resto/read	1
/tb_speech_resto/con_read	1
/tb_speech_resto/din	11000100
/tb_speech_resto/adr	00010010101110011000
/tb_speech_resto/str_adr	00000000000000000000
/tb_speech_resto/rd_req	0
/tb_speech_resto/dac_dout	11000100
/tb_speech_resto/u/clk	1
/tb_speech_resto/u/reset	1
/tb_speech_resto/u/read	1
/tb_speech_resto/u/con_read	1
/tb_speech_resto/u/din	11000100
/tb_speech_resto/u/adr	00010010101110011000
/tb_speech_resto/u/str_adr	00000000000000000000
/tb_speech_resto/u/rd_req	0
/tb_speech_resto/u/dac_dout	11000100

Figure 6