ARCHITECTURE OF AN APPLICATION SPECIFIC INSTRUCTION SET PROCESSOR FOR PARAMETRIC SPEECH SYNTHESIS


* Scientist, † Trainee
IC Design Group
Central Electronics Engineering Research Institute
Pilani – 333 031 (Rajasthan), India
chandra@ceeri.ernet.in

ABSTRACT

This paper analyses the parametric speech synthesizer by D. H. Klatt [1, 2] from the point of view of designing an Application Specific Instruction Set Processor (ASIP) chip for parametric speech synthesis. By analyzing Klatt’s code, the frequency of different computational operations in the code is estimated and constraints on speeds in performing these operations are derived. Next, the definition of a suitable instruction set for the ASIP and the hardware architecture for its implementation are proposed and analyzed. The architecture is verified using VHDL modeling and simulation. The strategy of implementation of the instruction set on the proposed hardware architecture is discussed and estimates of equivalent gate counts and RAM blocks needed for FPGA realization are given.

1. INTRODUCTION

The task of unlimited vocabulary text-to-speech conversion in different natural languages has been attempted by researchers using three broadly categoryisable approaches, namely:

- Articulatory Synthesis.
- Parametric Synthesis.
- Concatenative Synthesis.

While the relative strengths and weaknesses of each approach have been discussed in technical literature [3, 4, 5, 6], the parametric synthesis and concatenative synthesis approaches have been widely used by researchers.

In the context of Indian languages, parametric speech synthesis has been tried by the speech technology groups at various R&D laboratories and academic institutions in India. Parametric data base of syllables of Hindi language has been developed by CEERI (Delhi Centre).

Using this database and the associated set of layered rules, input text is parsed and converted into a sequence of parameter frames. The conversion of parameter frames to speech samples has been done using an adaptation of Klatt’s parametric speech synthesizer (C/C++ code running on a PC). This approach has produced good quality speech synthesis for Hindi language and can also be adapted for other Indian languages [7, 8].

In order to use this approach to develop a hand-held portable speech synthesizer or include speech synthesis capability in other hand-held devices such as pagers, PDAs etc., dedicated chip-level hardware / hardware-software solutions for the approach are required to be developed.

From the point of view of designing chips for the purpose, parametric approach to speech synthesis was partitioned into two logical parts to be mapped into their corresponding chip-level hardware / hardware-software solutions. One part com-
prises of designing a chip solution for converting parameter frames into speech samples i.e. implementing the Klatt's parametric speech synthesizer in a chip; and the other part comprises of designing a chip-solution for input text parsing and the application of layered set of rules to the text to convert it into parameter frames — to be passed on to the first chip for conversion into speech samples.

The following sections describe the design organization, hardware architecture design and instruction set design of an application specific instruction set processor chip (to be initially realized on a FPGA) for efficiently implementing the Klatt’s parametric speech synthesizer.

2. DESIGN METHODOLOGY

Given the real-time nature of the task, the complexity of the algorithm involved and the desirability of flexibility and enhanceability, an application specific instruction set processor based approach was chosen. Since the dynamic range of numerical quantities and expressions involved in the algorithm was relatively large, the IEEE-754 floating point (single precision) format was chosen to internally represent the numbers. Since the synthesizer code was built out of a number of different subroutines (each one performing a logical sub-task of the over-all task and not being too complex), it was decided to define an instruction set for the processor which comprised of instructions that performed the computational tasks corresponding to those of the different subroutines. Branching and looping instructions were added to the instruction set to provide flexibility and enhanceability.

A computational load analysis of the algorithm was done.

For one parameter frame, a count was taken of the number of times each subroutine was called. Besides, a count was also taken of the number of times each subroutine used the operations of addition-subtraction, multiplication and division. It was estimated from these counts that the synthesizer required 3419 floating-point additions-subtractions, 3774 floating-point multiplications and 154 floating-point divisions per parameter frame.

This count did not take into account the arithmetic operations embedded in the algorithms for computations of library functions such as exponential function, sine-cosine functions, square root function, db-to-linear conversion function and random number generation function.

Thus, for a parameter frame rate of 200 frames per second, one would require approximately 1 million floating-point addition-subtraction operations and around 1 million floating-point multiplications per second.

In addition to the operations, one would need time (clock cycles) to move operands from the registers/memories to functional units and the results from functional units to registers/memories. Assuming that around half the time is taken up by such data movements, one needs to complete about 1 million floating-point additions-subtractions and 1 million floating-point multiplications every half-a-second i.e. 2 million floating-point additions and 2 million floating-point multiplications per second. Accordingly, a design decision was taken to provide three separate functional units in the architecture:

1. Floating-point addition-subtraction unit.

2. Floating-point multiplication unit.

3. A multi-functional unit, for the remaining functions that are required.

From the design organization as well as FPGA implementation point of view, four separate memory blocks were provided in the architecture:

1. Program memory.

2. Parameter memory for storing the received parameters of a parameter-frame and their transformed values used by the algorithm.

3. Data memory for storing all the intermediate results (variables) of the algorithm.

4. Speech samples memory for storing and outputting the computed speech samples.

Out of these, the first three memories are single-ported synchronous memories, whereas the fourth
one is a dual-ported synchronous memory. The fourth memory is divided into two sub-blocks which are alternatively used for buffering the computed speech-waveform samples and for supplying speech samples to the external A/D converter at 10/12 KHz rate via the output controller.

The ASIP architectural diagram is given in Figure 1.

3. INSTRUCTION SET AND ITS IMPLEMENTATION

The complete computation of Klatt’s parametric speech synthesizer can be viewed as a three-tier algorithm. The complete synthesizer is the top-tier, individual subroutines in the synthesizer constitute the second-tier, and the mathematical operations and functions used by the subroutines constitute the third-tier.

A design decision was taken to embed the third-tier algorithms in the functional units of the architecture. Thus, the floating-point addition-subtraction unit, and the floating-point multiplication unit have their respective algorithms embedded in them. The third functional unit (multi-functional unit) has the algorithms for computation of floating-point division, exponential function, sine-cosine functions, random number generation, square root function and dB-to-linear conversion function embedded into it via a FSM controller.

The second-tier algorithms (subroutines given in Klatt’s code) are realized via individual instructions in the instruction set. For example, there is an instruction to compute the output of a resonator (with an implied addressing mode). Similarly, there is an instruction each to compute the output of each of the three voicing sources (the glottal source, the aspiration source and the friction source).

The execution of the instruction set is achieved by sequences of data moves into and out of functional unit input and output registers, and by triggering the functional units to start the computations. The controls for necessary data moves to/from the functional units and their triggering come from the control words stored in a Writable Control Store (WCS). The control words for instructions are sequenced by a state sequencer where starting state for each instruction is supplied by the instruction decoder.

The top-tier algorithm is implemented by storing the sequence of instructions in the program memory, and fetching and executing instructions from it based on the programme counter.

A VHDL-based behavioural simulation of the architecture and the implementation of instruction set on it has been used to validate the ASIP conceptualization. Detailed RT-Level design is currently underway. The RT-level design will be optimized for gate count — consistent with the desired speed of execution for the top-tier algorithm.

Currently, RT-Level synthesizable VHDL design has been completed for some of the architectural blocks. Using Xilinx Virtex series FPGA device, XC9400, as the target device and using a 25 MHz system clock, the floating-point adder-subtractor design requires approximately 5,000 equivalent gates and three clock cycles to compute its result, the floating-point multiplier design also needs approximately 5,000 equivalent gates and five clock cycles.

The size estimate of the parameter RAM, the Data RAM and the Speech Sample RAM is one single block (4K bits) each and that for the WCS 16K bits. Hardware flow-charting of instructions in the instruction set (corresponding to subroutines in Klatt’s code) has indicated that the architecture clocked at 25 MHz can conveniently handle a parameter frame rate of 200 frames/second (parameter update rate of 5 ms).

While it was possible to over-build a hardware architecture that would be much faster, it would have needed a larger number of logic gates. We chose an architecture that was designed to economize on the number of gates used — consistent with the performance necessary for the application.

4. CONCLUSION

This paper describes the architecture design of an Application Specific Instruction Set Processor (ASIP) to implement Klatt’s Parametric Speech Synthesizer that constitutes one part of the ASIP based implementation of the parametric speech
synthesis technology for Hindi language. Such a customized-processor based implementation of the technology can pave the way for the introduction of this technology in portable hand-held devices for mobile and low-power applications.

5. ACKNOWLEDGMENT

The authors wish to thankfully acknowledge the help and technical support that they received from Dr. S. S. Agrawal, Scientist Emeritus, CSIO (Delhi Centre), and all the colleagues of speech technology team at CEERI (Delhi Centre). They also wish to thank all the M. E. and B. E. thesis students, particularly under the CEERI-BITS collaborative M. E. (Microelectronics) programme, who have assisted the project by selecting one or the other aspect of it as a topic of their theses under the supervision of the authors.

REFERENCES


Figure 1: *ASIP Architecture Diagram.*