VLSI STRUCTURES FOR THE IMPLEMENTATION OF A FORMANT SPEECH SYNTHESIZER

C. D. Summerfield†

ABSTRACT

This paper describes a new approach to the implementation of the central signal processing functions of a parallel formant speech synthesiser using a bit-serial VLSI structure developed using the FIRST silicon compiler. The VLSI primitives have been arranged in a tightly looped structure with a high degree of computational concurrency to optimise the design for speed. This leads to a synthesis device which has a processing bandwidth far in excess of that required for real-time speech production. This excess may either be used for wide band speech synthesis or as the central signal processing element in a multiple channel formant speech synthesis device.

INTRODUCTION

A number of formant speech synthesisers have been implemented using general purpose signal processing devices, such as the TMS32010 device (Summerfield and Clark, 1986) and the NEC 7720 device (Quarmby and Holmes, 1982). In this paper a highly flexible bit-serial VLSI structure is described. This performs the central signal processing functions of a formant speech synthesiser. The bit-serial structure has been developed using the FIRST silicon compiler (Denyer and Renshaw, 1985). This compiler is primarily intended for the development of bit-serial signal processing devices and contains an arithmetic primitive library which is well suited to this application.

In the present design the VLSI structure has been optimised for processing speed. The excess bandwidth allows some interesting options to be considered, including wide band speech synthesis at higher sample rates and additional formant filters or the development of a multiple channel speech synthesis device.

FORMANT SYNTHESISER DESIGN

The synthesiser layout, shown in figure 1, is based on the parallel formant filter connection (Holmes, 1982). It consists of 6 formant filters F1 to F5 and FN connected in parallel. Each formant channel filter consists of a mixer circuit followed by a resonance filter. Formant channels F1 to F5 also contain an additional fixed filter which modify the formant filter skirt responses. For formant channels F2 to F5 the fixed filters contain a single zero at the origin whilst the F1 fixed filter consists of 2 zeros at -640Hz and 270Hz and a pole at -270Hz (Rye and Holmes, 1982).

The main difference between this design and previous parallel synthesiser designs reported by Holmes (1982) is in the high frequency formant filters F4 and F5. In the Holmes synthesiser the high frequency response is controlled by a cascade connection of 3 fixed resonance filters set at 3.1KHz, 3.5KHz and 3.9KHz. This arrangement crudely approximates the spectral characteristics of speech signal above 3KHz. Although this is necessary for obtaining the correct spectral tilt of the speech signal, it has been shown (Holmes, 1981) that the detailed spectral structure in this region contribute very little to the perceptually significant aspects of the signal. In the VLSI design these filters have been replaced by a parallel connection of 2 adjustable resonance filters. The main reason for this modification is to achieve a more structured parallel architecture. This simplifies the VLSI design and considerably reduces the complexity and size of the synthesizer device.

† Centre for Speech Technology Research, University of Edinburgh, 80 South Bridge, Edinburgh EH1 1HN
The bit-serial VLSI architecture used to perform the central signal processing functions in the formant speech synthesiser is shown in figure 2. This structure contains the input excitation function mixer; the resonance filter; the fixed differential filter and the formant filter coefficient generators. In the present design, the $F_1$ fixed filter has been implemented as a separate bit-serial VLSI device. The excitation function generators are also provided externally to the signal processing structure.

The structure has 6 input lines connected to multipliers 1, 2, 5 and 6. The input lines connected to multiplier 1 control the gain of the voiced excitation function. Similarly, the input lines connected to multiplier 2 control the gain of the fricative excitation function. The output from these multiplier primitives are combined in adder 1 to form the composite excitation function which is applied to the resonance filters.

The formant resonance filter is implemented as a direct form 2 second-order recursive digital filter using multiplier primitives 3 and 4; shift registers 1 and 2; adders 2 and 3 and a subtractor primitive (subtractor 1). This structure performs the difference calculation:

$$o(t) = i(t) + 2C_1o(t-\tau) - C_2o(t-2\tau)$$

Where $i(t)$ is the input composite excitation function sample value at the output of adder 1; $o(t)$ is the formant filter output sample value at the output of adder 2, and $o(t-\tau)$ and $o(t-2\tau)$ are the previous two output sample values stored in the shift register primitives 1 and 2, respectively. $\tau$ is the sampling rate interval (assumed to be 100$\mu$s).

The formant filter coefficients $C_1$ and $C_2$ control the frequency and bandwidth of the resonance filter. These are generated by multiplier 5 and 6 from the mapped formant filter frequency and bandwidth values $F_m$ and $B_m$, respectively:

$$C_1 = F_mB_n \quad C_2 = B_m^2$$

\(2a,b\)
The frequency and bandwidth mappings are provided externally to the VLSI signal processing structure using ROM or PLA look-up tables. The mapping functions are given by:

\[
F_m = \cos(2\pi f/\tau) \quad B_m = \exp(-\pi b/\tau) \quad (3a,b)
\]

Where \( f \) and \( b \) is the frequency and bandwidth of the formant filter in Hertz, respectively.

A second subtractor primitive (subtract 2) is used to produce the differential output by subtracting the \( o(t-\tau) \) value which is available at the output of shift register 1.

Sufficient shift register storage has been provided to allow the VLSI structure to be multiplexed 6 times. This enables all 6 formant filter calculations to be performed sequentially by a single formant filter structure. The design is optimised for processing speed and the complete synthesis structure operates at 100% duty cycle. For a 16 bit synthesis system all formant filters calculations are completed in 96 clock cycles. Thus, for real-time speech synthesis at 10KHz the VLSI clock rate is 960KHz. This is extremely low for most modern VLSI technologies and allows the extensions mentioned above to be incorporated into the basic VLSI structure by simply expanding the lengths of the shift registers.

Figure 2. Bit-Serial VLSI Formant Filter Structure

Figure 3 shows the impulse response of the VLSI structure for the nasal formant channel, FN, with a centre frequency of 500Hz and a bandwidth of 100Hz (the sample rate is assumed to be 10KHz). The transfer functions of the individual formant filters and the complete synthesiser are shown in figure 4(a) for a neutral vowel \( (F_1 = 500\text{Hz}, F_2 = 1500\text{Hz}, F_3 = 2500\text{Hz}, F_4 = 3500\text{Hz} \) and \( F_5 = 4500\text{Hz}) \). All formant bandwidths and gains are equally set to 100Hz and unity, respectively.
To verify the correctness of the VLSI design, this impulse response was compared against the transfer function from a floating point simulation of the synthesiser structure shown in figure 4(b). Both transfer function characteristics were computed from the respective impulse responses using a 512 point FFT.

![Graphs showing VLSI Synthesiser and Floating Point Simulation transfer functions.](image)

**Figure 4. Transfer Functions**
(a) VLSI Synthesiser  
(b) Floating Point Simulation

**COMMENTS AND CONCLUSIONS**

The similarity between these transfer functions indicate that the VLSI structure is operating correctly. Most of the discrepancies between the responses occur at very low signal levels where the quantisation resolution restricts the computational accuracy. The main factor limiting the device performance is the limit-cycle properties of the recursive structure introduced by truncation errors in the filter calculation. Close inspection of the output signal indicates that this property affects the 5 least significant bits of the calculation.

**REFERENCES**


