Composing With The York Polyphonic Real-Time Singing Synthesiser

Ian S. Gibson, David M. Howard, Andrew M. Tyrrell
Department of Electronics, University of York, Heslington, York YO10 5DD
Email: isg100@york.ac.uk

Abstract
To date, most singing synthesisers have been monophonic due to limitations of processing power. A real-time singing synthesiser has been developed which can synthesise four voices each with six formants. The hardware includes SHARC ADSP-21060 parallel processors, a Digital-To-Analogue converter and a Field Programmable Gate Array (FPGA). This paper presents a description of the synthesiser, methods of control and the results obtained.

1. Introduction
Existing singing synthesisers generally do not offer polyphonic output due to processing limitations of single processor systems. For this reason a parallel processing system was chosen to implement a polyphonic singing synthesiser (Gibson et al, 1998). Each voice is synthesised on a separate processor in real-time.

ADSP-21060 SHARC processors. Each has 4 Mbits of dual ported on-chip SRAM (organised as two blocks of 2Mbits each). The memory can be configured as 128K words of 32-bit data, 256K words of 16-bit data, 80K words of 48-bit instructions, or combinations of word sizes up to 4 megabits. Memory may be accessed with direct and indirect addressing. Up to 6 processors can access each others internal RAM and I/O registers at a rate of up to 240 Mbytes per second.

![Diagram of the Singing Synthesis System]

Figure 1: The Singing Synthesis System

If would be difficult to provide an intuitive interface which allows the performer to control every synthesis parameter in real-time. Therefore it was necessary to establish which parameters would be most desirable for a performer to change during a performance.

2. The Hardware
The design of the singing synthesis system is shown in figure 1. The system consists of several

There is a 25 ns instruction cycle time with an operating time of 40 MIPS. An on-chip instruction cache enables two operands and an instruction to be fetched for every cycle (a three-bus operation). The cache is selective, only storing instructions which conflict with program memory data accesses. A 1024-point complex FFT executes in 0.46ms. Both fixed and floating point data formats are supported.

Off-chip memory and peripherals may be accessed using the external port. An address space of 4-gigawords is provided in unified address space.
The host interface allows connection to 16 and 32-bit microprocessor busses. Four channels of DMA are available for the host interface.

The system uses the Apex-Comms software development environment that is specifically designed to support real-time applications running on networks of SHARC DSPs. Access to asynchronous communications is achieved using the Channel Application Programmer Interface. Two structures are provided for channel communication: CHAN_IN and CHAN_OUT. When either is called the processor idles until the other processor makes a matching call. It is also possible for a processor to check if another is ready to transmit data, thereby allowing a transfer to take place only when both processors are ready.

3. The Synthesis Model

A source/filter model (figure 2) (Fant G., 1980) is generally used for the synthesis of speech. A sound is produced when the lungs create an airstream which is passed through the glottis causing the vocal folds to vibrate. The acoustic result is known as the source. The articulators (e.g. jaw, tongue, pharynx and lips) modify the signal, and this is known as the filter. These change the volume of the cavities in the vocal tract and the acoustic responses (or formants) associated with them.

Existing speech and singing synthesis models connect the formants in series (Klatt D., 1980) or parallel (Holmes J.N., 1987). A parallel formant synthesiser requires control over the formant amplitudes, whereas a serial formant synthesiser automatically scales amplitudes. The latter has been chosen for the purposes of this research.

The wavetable for voiced speech is usually based upon the activity of the vocal folds. The filter section of the model provides a number of resonances with time varying centre frequencies and bandwidths. The remaining formants are often fixed. Some typical formant values of the male singing voice are shown in figure 3.

<table>
<thead>
<tr>
<th>Vowel</th>
<th>f1</th>
<th>f2</th>
<th>f3</th>
<th>f4</th>
<th>f5</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>609</td>
<td>1000</td>
<td>2450</td>
<td>2700</td>
<td>3240</td>
</tr>
<tr>
<td>E</td>
<td>400</td>
<td>1700</td>
<td>2300</td>
<td>2900</td>
<td>3400</td>
</tr>
<tr>
<td>IY</td>
<td>238</td>
<td>1741</td>
<td>2450</td>
<td>2900</td>
<td>4000</td>
</tr>
<tr>
<td>O</td>
<td>325</td>
<td>700</td>
<td>2550</td>
<td>2850</td>
<td>3100</td>
</tr>
<tr>
<td>OO</td>
<td>360</td>
<td>750</td>
<td>2400</td>
<td>2675</td>
<td>2950</td>
</tr>
<tr>
<td>U</td>
<td>415</td>
<td>1400</td>
<td>2200</td>
<td>2800</td>
<td>3300</td>
</tr>
<tr>
<td>ER</td>
<td>300</td>
<td>1600</td>
<td>2150</td>
<td>2700</td>
<td>3100</td>
</tr>
<tr>
<td>UH</td>
<td>400</td>
<td>1050</td>
<td>2200</td>
<td>2650</td>
<td>3100</td>
</tr>
</tbody>
</table>

Figure 3: Five formant frequencies (Hz) of eight vowels sung by a male

The York Singing Synthesiser is a series formant synthesiser. A lookup table is not used to hold the oscillator waveform. Instead it generates each sample based upon input from the voice source parameters. This allows voice source parameters (e.g. Open Quotient) to be changed without the need to generate or read a new lookup table.

3. The Interface

The synthesiser consists of a GUI (responsible for handling control data) and the SHARC network (responsible for sound generation). There are four SHARC processors each running a 6 formant synthesiser. The root SHARC processes control data from the GUI to the network. The synthesised sound data is passed back to the root SHARC.

A synthesis by analysis approach provides the primary method for driving the synthesiser. A sound sample of human singing is analysed by a software package which produces a file containing formant frequencies and bandwidths over time, and this data is converted into a Klatt synthesiser script. The Visual Basic GUI reads this script and passes the data to the root node of the SHARC network. There is one script for each voice.

The GUI allows the rate of synthesis parameter update to be changed as the synthesiser is running. If the rate of update is too slow then there is stepping effect in the resulting synthesised sound. If the rate of update is too fast (i.e. only a small number of samples are generated before the next control parameters are read) then there is too much data for the system to process successfully in real-time. For example, parameters may be updated every 125 samples at a sampling rate of 11025 Hz.

4. Composing With The Synthesiser

A survey of musicians was undertaken to establish their requirements for a real-time singing synthesiser. It also provided information relating to which parameters would be directly controllable.
in real-time and which would be controlled using some form of score or script. Typically, singing synthesis systems offer control over:

- fundamental frequency,
- voice quality (e.g. Open Quotient),
- the glottal spectrum,
- vibrato and
- formant frequencies.

Like all real-time digital sound synthesis methods, only a limited number of parameters may be controlled directly by a performer during a performance. The most important parameters which musicians required direct real-time control over (as opposed to real-time control using a script or score) were those relating to pitch (e.g. fundamental frequency, vibrato amplitude and frequency) and amplitude. These parameters are fundamental to singing synthesis, allowing the performer a great deal of expression during a live performance.

4.1. Script Control Of The Synthesiser

An audio sample is encoded into a 16 bit raw data format. This is analysed using a software package to produce files that contain information about the fundamental frequency, formants and voicing of the sample over time. Another software package is used to read and display these values graphically.

Usually, the analysis software does not produce very smooth formant trajectories, so these must be adjusted by hand. The areas where the formant values are most erratic occur in between periods of voicing, and therefore these need not be adjusted. Also, the periods of voicing are sometimes not accurately identified, so these must be manually corrected. An example of the first formant and bandwidth trajectories before and after manual correction are shown in figures 4, 5, 6 and 7. The data is used to produce a script which is read by the synthesis system.

4.2 Control Of The Synthesiser Using a GUI

A GUI allows the performer to override script parameters during a performance. Alternatively, the performer may choose to synthesise singing using only the GUI. The following synthesis parameters for each voice part may be controlled using on-screen sliders:

- fundamental frequency,
- formant frequencies,
- formant bandwidths,
- vibrato rate and
- vibrato amplitude.

A number of sliders are provided to allow mixing of each voice part and to scale their amplitudes.
5. Results

Figure 8 shows a spectrogram of an original and re-synthesised sound sample. The sampling rate used was 11025 kHz and the duration of the sample was approximately 7 seconds. It can be seen that the spectrograms look very similar to each other. The synthesised sound sample is slightly shorter in duration than the original. This is due to slight inaccuracies of the timer in Visual Basic. This problem can be overcome by making the root SHARC responsible for reading the input script rather than the Visual Basic GUI. Also, it can be seen that the parameters generated by the analysis software sometimes cause a stepwise change in values rather than a continuous change as seen in the original singing sample.

The onset and offset times of the synthesised sounds are shorter than the original. It was necessary to impose this feature to overcome the erratic tracking of voice parameters by the analysis software which was most noticeable at these times. Therefore the synthesis engine is not responsible for this effect. This problem may be overcome by carefully editing the script parameters to produce a more natural timbre at note onset and offset times.

6. Conclusions And Future Research

A real-time polyphonic sound synthesiser has been successfully designed and implemented using ADSP-21060 SHARC processors. Since the algorithm for each voice is identical, polyphony can easily be increased by adding further nodes to the network.

A more comprehensive user interface is to be implemented, the design of which is based upon results from a user survey. A GUI will allow mapping of control parameters to synthesis parameters through a graphical interface. Consideration will be given to designing the interface for use in real-time performance by musicians.

The real-time control of the system will be expanded. A performance will be controlled partly by a script which will update synthesis parameters in real-time. The performer will select certain parameters (e.g. fundamental frequency) for direct control using MIDI or the GUI. The MIDI interface will allow control of sound synthesis using a musical keyboard as well as more unusual methods of control (for example pressure sensors).

References


